ADDALY · THE AI CLASSROOM

Working In AI

The roles that exist, what they pay, and how people get in without a famous degree.

9 modules · 85 lessons · 33 figures · 9 case studies · 65,131 words · about 12 hours

Dr Mustafa Mun
Author and editor

ABOUT THIS BOOK

Working In AI, published by Addaly, 2026. Author and editor: **Dr Mustafa Mun**.

The manuscript was drafted with the assistance of a large language model and was edited and published under his name. That is stated plainly here rather than left to be discovered, because a reader is entitled to know how a book they are trusting was made.

This book is the printed form of the course of the same name at addaly.com/learn/ai-careers. The website is the living version and is corrected first; where the two differ, the website is right.

Free to read, free to download, free to print, and free to teach from. There is no paid edition and there never will be. If you paid for this, somebody has sold you something that was given away.

Every diagram in it is drawn from data by code, not generated as a picture, so the numbers on the page are the numbers in the argument. Answers to every exercise, test and examination question are at the back, with a note on why each wrong option attracts people — those notes are the teaching, not the marking.

Contents

1. The terrain: who hires, for what, and where you fit

Name the employer types that hire for AI work in your own market, decode a job advert into the role it really is, and commit to one lane with a plan sized to the hours you actually have.

2. The roles, one at a time

Describe one role's ordinary week, the artefact it produces, the stack it runs on and two realistic routes into it in enough detail that somebody doing that job recognises their own week in your description.

1. The data analyst
2. The analytics engineer
3. The data engineer
4. The machine learning engineer
5. The AI engineer
6. The platform and MLOps engineer
7. The applied scientist and research engineer
8. Trust, safety and the policy side of AI
9. The AI product manager
10. The forward-deployed and solutions engineer
11. Case study: The model that scored 0.94 in the notebook
12. Module test

3. The evidence: what gets tested, and how to prove you have it

Produce one written case study of your own work — with the mess you found, the decisions you rejected, and a number from a test set you built by hand — that a reviewer can judge in ninety seconds.

1. When the courses end and you still freeze
2. SQL, and the six query shapes that cover the job
3. The employable half of Python
4. The statistics that come up, and the ones that never do
5. Git, and what a reviewer sees before your code
6. The terminal, the container, and a cloud bill of zero
7. The spreadsheet skill nobody puts on a syllabus
8. Evaluation: deciding what "correct" means, in advance
9. A portfolio when you have never been paid for this
10. Where portfolio data comes from, and what you may do with it
11. Writing up a project as a decision record
12. Case study: The clinic's spreadsheet, and what could be published
13. Module test

4. Getting seen: the routes that put a person in front of your work

Run a search that puts your work in front of named people rather than into application queues, and identify the bridge role or first paid job you would take to get inside.

1. How people actually get in
2. Running the search as a funnel
3. A CV that survives a thirty-second read
4. The profile a stranger checks in ninety seconds
5. Writing to strangers, and what to say
6. Publishing, and the inbound it creates
7. A merged pull request as evidence
8. The job next to the data
9. The first client, and how not to lose money on them
10. Case study: Three hundred and forty-one applications
11. Module test

5. Using AI in your own search

Use a model for the mechanical parts of a search — tailoring, research, rehearsal, gap analysis — while stating correctly what an applicant tracking system does and does not do, and keeping your own and an employer's confidential material out of a third party's logs.

1. What it genuinely helps with, and what it quietly ruins
2. What an applicant tracking system actually does
3. Rewriting a CV against an advert, and the line
4. Why volume stopped working, on both sides
5. Your CV, their take-home, and the terms
6. Forty minutes of research, and how to check it
7. Recorded and machine-scored interviews
8. Rehearsing with a model that wants to please you
9. Building a project you can still defend
10. Case study: The take-home that arrived with somebody else's customers in it
11. Module test

6. The interview, from first screen to signed offer

Walk each stage of a data or AI hiring process knowing what it tests, and negotiate an offer without a competing one in hand.

1. The stages, and what each one is really testing
2. The first call, and the salary question
3. The take-home, and how not to lose on it
4. The live exercise, where silence is scored
5. "How would you build this?" answered well
6. "How would you measure it?"
7. Talking about your work without lying or shrinking
8. Your questions, and what the answers reveal
9. The offer conversation
10. Pay, rates, and who actually decides them
11. What a rejection tells you, and what it does not
12. Case study: Answer by ten tomorrow morning
13. Module test

7. The money, and the paperwork underneath it

Take an offer apart into cash, variable pay, employer contributions, equity and contract terms, compare two offers with explicit arithmetic rather than headline figures, and name the clauses and employment structures that materially change what you are being paid.

1. The headline number is the least useful one
2. Options, RSUs, and why most equity is worth nothing
3. Notice, bonds, IP and the clauses that follow you
4. Employee, contractor, or an employer of record
5. Getting paid by a company in another country
6. The shape of the routes, and why the rules keep moving
7. Two offers, compared with arithmetic
8. Raises, promotions and the market rate
9. Case study: Ninety days to find twenty-five lakh
10. Module test

8. Freelancing and contracting, as a business

Price your own work upwards from a target income and a realistic utilisation figure, write a proposal whose exclusions prevent scope creep, run a payment process that survives a slow client, and state the conditions under which you should take a salaried job instead.

1. Utilisation, and why a rate is not a salary divided by hours
2. Pricing the work, and raising the price
3. A pipeline, after the first three clients
4. Writing the thing that prevents the argument
5. Invoices, procurement, and the chase
6. Registration, tax, records and insurance
7. Repeatable offers, retainers and the ceiling of hours
8. The conditions under which you should take the job
9. Case study: Invoice 0042, and the purchase order nobody mentioned
10. Module test

9. Staying: the first ninety days and the ten years after

Earn trust in a new data role within ninety days, and decide what to learn, what to refuse and when to leave using evidence rather than anxiety.

1. The first ninety days
2. Landing in a codebase nobody documented
3. Being trusted with a number
4. Which parts of this work are most exposed
5. Keeping up without drowning
6. The things you will be asked to build
7. The second job, the plateau, and the ten-year view
8. The one-pager that gets a decision
9. Getting promoted, and the case somebody has to make
10. Case study: The postcode feature, asked for on a Thursday
11. Module test

Working In AI — final examination

The paper that opens the next course. Answers to everything follow it.

MODULE 1

The terrain: who hires, for what, and where you fit

Before you choose what to learn, look at the market you are actually in — the eight kinds of employer, how a job advert encodes the real job, and what a credential buys. This module comes first because most wasted years are caused by aiming at the smallest employer category with the wrong evidence.

> BY THE END OF THIS MODULE YOU CAN
>
> Name the employer types that hire for AI work in your own market, decode a job advert into the role it really is, and commit to one lane with a plan sized to the hours you actually have.

---

1 · 7 min

The jobs, described as they feel on a Tuesday

> THE ONE THING TO KEEP
>
> Most AI work is moving messy data and checking outputs, not training models.

Titles lie, work does not

"AI Engineer" means at least six different jobs depending on who wrote the posting. Read the responsibilities section, ignore the title. Here are the roles that actually exist, described as they feel on an ordinary Tuesday rather than in a brochure.

Data engineer

You move data from where it is created to where someone can use it. Pipelines, schemas, backfills, an alert at 03:40 because last night's job died on a row containing a comma.

A Tuesday: a dashboard shows Monday's orders as zero. You trace it to a payments provider that renamed a field from `order_id` to `orderId` on Saturday without telling anyone. You fix the parser, replay two days of data, and write a note so the next person loses ten minutes instead of three hours.

There are more of these jobs than any other kind on this list, in every country. Least discussed, most stable.

Data analyst

SQL, spreadsheets, dashboards, and the question behind the question. Someone asks for "signups by region". What they are actually deciding is whether to keep paying for the Nairobi campaign. Most of the skill is finding that out before you write the query, because the query takes four minutes and the wrong question wastes a week.

This is the most common first job in the field for people who did not study computer science.

ML engineer

You put models into products and keep them alive: serving, latency, retraining, monitoring, rolling back when the new version is quietly worse. Training models is a slice of it. Training one from scratch is rare outside research labs.

A Tuesday: recommendations got 400ms slower on Friday and nobody knows why. You find a feature lookup making one database call per item instead of one per request.

AI product manager

You decide what gets built and what "good enough" means — which in this field means writing down how the system will be measured before it exists.

Also: arguing with legal about which data may be used, cutting scope, and telling people no in a way they can accept.

Prompt and evaluation work

The newest and least settled corner. The durable half is not writing prompts, it is evaluation: assembling the test cases that decide whether a system ships, grading outputs, hunting failure modes, turning "this feels off" into a number someone can act on. Job titles here change every eighteen months. The work does not.

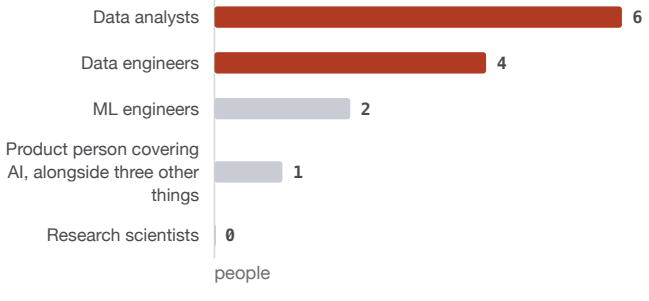
Research scientist

Reading papers, running experiments that mostly fail, writing up the few that do not. A small number of jobs, concentrated in a few dozen labs and universities worldwide, nearly all wanting a PhD. Keep it on your map. Do not plan your life around it.

The ratio nobody prints

Picture a 300-person fintech in São Paulo with a genuine data team: about four data engineers, six analysts, two ML engineers, one product person who covers AI alongside three other things, and zero researchers. Multiply that shape across thousands of companies and you have the actual job market.

One real data team, at a 300-person fintech



Thirteen people and nobody training models from scratch. Multiply this shape across thousands of companies and it is the job market: analyst and data engineer are the two wide doors.

If you are choosing where to aim, aim where the jobs are and where you can produce evidence fastest. Analyst and data engineer are the two widest doors. ML engineering is far easier to reach from a data engineering job than from a course.

What they all share

Every role above is mostly one activity: something is wrong with the data and you have to find out why. Courses skip this because it does not demo well. Employers pay for it because it is where the time goes.

BEFORE YOU MOVE ON

A 300-person logistics company in Jakarta hires its first AI person to build an assistant that answers staff questions about shipments. Three months in, where has most of that person's time gone?

A. Fine-tuning a language model on the company's shipment history.

B. Getting shipment data out of four systems that disagree with each other, and building a set of test questions to check the answers against.

C. Comparing model providers and negotiating a contract with one of them.

D. Rewriting the prompt until the answers read well.

2 · 8 min

The eight kinds of employer, and how many of each

THE ONE THING TO KEEP

Nearly every AI-adjacent job sits inside an ordinary organisation that uses software, not inside a company that builds models.

The picture in most people's heads is the smallest box

Ask someone where AI jobs are and they name four or five companies that train frontier models. Those organisations together employ a few thousand research staff worldwide, nearly all with doctorates, nearly all in a handful of cities. That is a real employer category. It is also the one you are least likely to work for and the one that shapes almost none of the market you will actually apply into.

Here is the fuller list. Read it as a map of where to spend your applications.

1. Frontier labs and big-tech research

Tiny, famous, PhD-heavy, publication-driven. Salaries at the extreme top. If you are not already publishing, this is not a plan.

2. Big-tech product teams

Large and real — the recommendation, search, ads, fraud and infrastructure teams. They hire steadily but skew senior, and their junior intake mostly comes through internships and graduate programmes with fixed windows. If you are outside that pipeline, entry is hard and lateral entry at three years' experience is much easier than entry at zero.

3. IT services and outsourcing

Infosys, TCS, Wipro, HCL, Cognizant, Accenture, Capgemini, Globant, and hundreds of smaller firms. In India, the Philippines, parts of Latin America and Eastern Europe this is the single largest employer of junior technical people, by a wide margin.

The economics are worth understanding because they change your strategy. These firms bill clients for your time and often bill more for certified staff, which is why a cloud certification has genuine commercial value here and almost none elsewhere. You may sit "on the bench" between projects. Training is real but generic. Pay starts low and rises slowly.

The honest read: it is a legitimate and very wide door, and a large number of working data engineers walked through it. Take the job, get two years, then move.

4. Product startups

Twenty to three hundred people. They read applications, they hire on evidence, and there is no qualifications filter because there is no HR department to run one. This is where a portfolio does the most work per hour you put into it. Against that: less mentorship, more chaos, and a real chance the company is gone in eighteen months.

5. Enterprises with internal data teams

Banks, insurers, telecoms, retailers, airlines, hospitals, manufacturers, logistics firms, utilities. This category is where most analysts and data engineers on earth actually work, and it is almost never discussed, because a regional insurer does not run a developer marketing campaign.

They have decades of data, real problems, formal processes, and hiring that runs on qualifications and references. Slower, steadier, and far less competitive per opening than a known technology brand.

6. Agencies and consultancies

Small firms doing project work for other companies. Broad exposure, many stacks, fast learning, and constant context-switching. Good for building range early. Hard to go deep.

7. Public sector, universities, NGOs, research institutes

Census bureaus, health ministries, transport authorities, agricultural research bodies, international organisations. Lower pay, slow processes, genuinely interesting data, and unusually low competition because almost nobody applies. Also unusually good for a first portfolio-driven hire, because these organisations publish their data and you can build something useful before you ever apply.

8. Yourself

Freelance and contract. Covered in its own lesson. Not a fallback — a category with its own economics.

The ratio nobody prints

For every organisation that trains its own foundation models, there are thousands that merely run software and have accumulated data they cannot use. Categories 3, 5 and 7 hold most of the jobs and generate the least conversation.

Eight employer types, ordered by how many beginners they take

3. IT services and outsourcing	the largest junior intake in India, the Philippines, Latin America, Eastern Europe
5. Enterprises with internal data teams	banks, insurers, telecoms, hospitals, logistics: where most analysts actually work
7. Public sector, universities, NGOs	low pay, published data, and almost nobody applies
4. Product start-ups	twenty to three hundred people; they read applications and hire on evidence
6. Agencies and consultancies	range early, depth rarely
8. Yourself	freelance and contract, with its own economics
2. Big-tech product teams	large but senior-skewed; juniors arrive through fixed graduate windows
1. Frontier labs	a few thousand research staff worldwide, PhD-heavy

Categories three, five and seven hold most of the jobs and generate the least conversation. The famous box at the bottom employs a few thousand people worldwide, nearly all with doctorates.

If you are optimising for a first job, apply where the jobs are, not where the news is.

The "AI team" that is not one

A caution you will meet quickly. Many companies now advertise AI roles while having no model in production and no data warehouse. The actual job is spreadsheets, one integration with a hosted model API, and a dashboard. This is not fraud; it is where those companies genuinely are. But go in knowing it, because it decides what you learn in your first year. The interview lesson on questions to ask exists mostly for this.

Finding these employers without paying anyone

The paid tools — LinkedIn Recruiter, Sales Navigator, Apollo, paid job aggregators — sell you a filtered list of companies. You can assemble the same list free:

- LinkedIn's free search, filtered by industry and employee count, in your city.

- Company career pages directly. Most enterprise roles are posted there and syndicated late or never.
- Your national employment portal, industry association member lists and chamber-of-commerce directories.
- Government tender and procurement portals, which name every firm doing data work for the state.
- Wellfound and Y Combinator's job board for startups; Kaggle and Hugging Face job boards for applied roles.

An afternoon spent building a list of forty local employers in categories 3, 5 and 7 is worth more than any subscription.

BEFORE YOU MOVE ON

Ayo lives in Lagos, has one deployed project and no degree. He spends his search on the careers pages of five frontier AI labs and three global technology brands. Ninety applications, no replies. What is the most useful diagnosis?

A. He needs a stronger portfolio before those companies will respond.

B. His applications are being rejected by resume-screening software he could beat with better keywords.

C. He is applying almost entirely into the two employer categories with the fewest openings and the highest bar, while the categories that hire most people at his level — services firms, enterprises with internal data teams, and public bodies — are absent from his list.

D. Lagos is the wrong market and he should focus only on remote roles with foreign companies.

3 · 10 min

Geography, time zones, and the office you have never heard of

THE ONE THING TO KEEP

Where you sit decides which employers can legally and practically hire you, and time-zone overlap filters candidates harder than any skill requirement in the advert.

Four ways to be paid by a company that is not in your city

The employer categories tell you who hires. This lesson tells you where the desk is, because that single fact changes your pay, your hours and how hard the door is to open.

1. A local company. Straightforward. Local pay, local law, local currency.

2. A services firm billing a foreign client. You are employed by an Indian, Filipino or Polish company and your work goes to a bank in London. Your pay is set by your employer's local market, not the client's.

3. A global capability centre. A multinational sets up its own engineering site abroad and employs people directly. This is the category almost nobody outside it knows exists, and it is enormous.

4. Remote for a foreign employer. You work from home for a company with no presence in your country. Highest ceiling, hardest door, and the one everybody aims at first.

The global capability centre

Also called a GCC, a captive centre, or an in-house development centre. A large company — an American insurer, a German carmaker, a British retailer — opens a wholly owned subsidiary in Bangalore, Hyderabad, Pune, Chennai, Gurugram, Kraków, Warsaw, Bucharest, Lisbon, Guadalajara, São Paulo,

Cairo, Nairobi, Manila or Ho Chi Minh City, and staffs it with hundreds or thousands of engineers, analysts and data people.

The difference from a services firm is the employment relationship. In a services firm you are billable inventory: your employer sells your hours and keeps the margin. In a GCC you are staff of the parent company's subsidiary, working on its own systems, with its own roadmap.

The practical consequences:

- Pay sits above local services firms and below the parent country. That gap is the whole reason the site exists.
- You get one system deeply rather than five shallowly, which is better for a technical career and worse for range.
- Hiring is often through the same local recruiters as everyone else, which means the bar is local, not head-office.
- These sites rarely market themselves. There is no "GCC" tag on a job board.

To find them: pick a large foreign company in an industry near you, search its name plus your city on LinkedIn, and look at the employee count. If a Dutch bank has 1,400 people in Chennai, that is a GCC. Then go to the company's own careers page and filter by country — those roles are often not syndicated to aggregators at all.

Time zone is a harder filter than skill

Most remote adverts carry a line like "must overlap four hours with Pacific time" or "European time zones only". People skim past it. It is usually the strictest filter in the advert.

Work the arithmetic once and it stops being abstract. Suppose a San Francisco team wants overlap from 09:00 to 13:00 Pacific:

- Lagos (UTC+1) — 17:00 to 21:00. An evening shift. Tiring but survivable.
- São Paulo (UTC-3) — 13:00 to 17:00. Ordinary working hours.
- Warsaw (UTC+2) — 18:00 to 22:00. Late, and legal working-time rules may apply.

- Bengaluru (UTC+5:30) — 21:30 to 01:30. This is a night job. Many people do it and burn out inside two years.

Now flip it. A London team wanting 10:00 to 14:00 UK time is 14:30 to 18:30 in Bengaluru and 11:00 to 15:00 in Lagos — comfortable for both. The lesson is not "remote is hard". It is that your continent determines which foreign markets are realistically open to you, and for South Asia that market is Europe and the Gulf far more than the US west coast.

The remote market has narrowed, and for a mechanism

Fully global remote hiring was wider in 2021 than it is now, and the reason is not fashion. When a company employs somebody in a country where it has no legal entity, it risks creating a *permanent establishment* — a taxable presence — and it inherits that country's employment law on notice, severance and leave. Finance and legal teams caught up, and the answer was to restrict hiring to countries where an entity already exists, or to route everyone through an employer-of-record service that charges a per-head fee.

That fee is the point. An employer of record typically costs a few hundred dollars a month per person. It is trivial next to a senior salary and material next to a junior one, which is exactly why remote junior roles are scarce and remote senior roles are not. If you are early, aim locally or at a GCC and treat global remote as the second job, not the first.

Moving within your own country is underrated

The highest-return relocation for most people is domestic: Lagos, Nairobi, Bengaluru, Hyderabad, Pune, São Paulo, Kraków, Cairo. Pay differences between a capital and a secondary city inside one country are frequently larger, in percentage terms, than the difference between two neighbouring countries, and there is no visa involved.

Before deciding, do the subtraction rather than comparing headline numbers. A 60% higher salary against 90% higher rent and a two-hour commute is a pay cut with extra steps. Write down monthly rent, transport, food and the hours you lose, then compare what is left.

BEFORE YOU MOVE ON

Priya is in Bengaluru with two years of analytics experience. She applies only to fully remote roles at US start-ups and gets nowhere. Which change is most likely to open doors, and why?

A. Target European and Gulf employers and the global capability centres of foreign companies already operating in India, because both give workable overlap hours and an existing legal entity to employ her through.

B. Improve her portfolio until it is strong enough to beat the US applicant pool, since remote roles are decided purely on evidence.

C. Accept a night shift, because overlap with Pacific hours is the standard cost of remote work and everyone pays it.

D. Register as a freelancer so US companies can hire her without employment restrictions.

4 · 9 min

The work that trains the models

THE ONE THING TO KEEP

Annotation and evaluation work is a real door into the field, but its value is the failure-mode intuition and data-quality vocabulary you take out of it, not a promotion track inside it.

An industry that does not appear in the industry's story

Every model you have used was shaped by people who were paid by the task. They wrote reference answers, ranked two responses against each other, flagged unsafe outputs, drew boxes around objects, transcribed speech, checked whether a citation was real, and argued in a comment thread about whether a guideline covered an edge case.

The names to search for are Scale AI and its worker platform Outlier, Surge AI, Appen, Toloka, Sama, TELUS Digital, Invisible Technologies, Mercor and Turing. Between them they engage hundreds of thousands of people. Almost none of that appears in how the field describes itself, and it is one of the widest doors into AI work that exists for someone with no job history.

What it actually pays, and how

Pay is per task or per hour, and it splits sharply by what the task requires:

- **General labelling and transcription** — the low end. In low-cost markets this is frequently a few dollars an hour, sometimes less once unpaid time is counted.
- **Language work in an under-served language** — better, because supply is short. Fluent Yoruba, Marathi, Vietnamese or Amharic is a genuine scarcity.
- **Domain expertise** — the high end. Qualified doctors, lawyers, accountants, translators and competitive programmers writing evaluation data are paid tens of dollars an hour, occasionally more.

The structural things nobody puts in the advert:

- **Qualification tests are often unpaid**, and can take hours.
- **Projects end without notice.** You may work forty hours one week and none the next.
- **Accounts get deactivated** for quality-score reasons with limited appeal. Read the platform's own community forums before committing.
- **Content moderation and safety work carries real psychological cost.** Reviewing violent or sexual material for months damages people. It is documented, it is not weakness, and if a role involves it you should know that going in and treat support as a requirement rather than a perk.

Be clear-eyed. Some of this work is well paid and interesting. Some of it is precarious piecework dressed in the language of a technology career.

Why it is still worth considering

Two reasons, and neither is the money.

First, it shows you what models get wrong. Nothing else gives you as many hours staring at bad model output next to a definition of what good would have been. That is the raw material of evaluation, which the evidence module treats as the scarcest hireable skill in the field. Somebody who has ranked four thousand responses has real intuitions about failure modes that a course cannot give.

Second, you learn the vocabulary of data quality from the inside. Annotation guideline, gold set, inter-annotator agreement, adjudication, label drift, edge-case memo. These are the exact words used by the teams that build training and evaluation data, and using them correctly in an interview marks you as someone who has done the work rather than read about it.

Turning the work into evidence

The failure mode is treating these platforms as a career ladder. They are largely flat by design: there is no promotion track from labeller to research engineer, and waiting for one wastes years.

The move that works is converting the experience into an artefact you own. All of these are things you can do without breaching a confidentiality agreement, because they describe your own process rather than the client's data:

1. **Write a guideline for a task of your own.** Take a public dataset, define a labelling task in one page, and state how to handle five edge cases.
2. **Label a sample twice, a week apart, and measure your agreement with yourself.** Report the percentage. Self-agreement below about 85% means the guideline is ambiguous, not that you are careless.
3. **Get one other person to label the same 100 items** and compute agreement between you. Then write up the disagreements and how you resolved them.
4. **Publish the guideline, the numbers and the adjudication notes.**

That package is a portfolio piece for any data-quality, evaluation or trust-and-safety role, and very few applicants have one.

What to check before signing up

- Is the pay stated per hour or per task, and what does the community say the effective hourly rate is?
- Is the qualification test paid?
- How does the platform pay out, in what currency, and what are the fees?
- Does the work involve harmful content, and is that disclosed up front?
- What are the terms about who owns work you produce and whether you may describe the project publicly? Usually you may not name the client. You can almost always describe your own methods.

BEFORE YOU MOVE ON

Blessing has spent eight months doing preference-ranking work on an AI platform and wants it to lead somewhere. What is the highest-value next step?

A. Apply for a team-lead position on the platform, since internal promotion is the natural route from labelling into full-time AI work.

B. Put the platform on her CV as AI experience and apply to machine learning engineering roles.

C. Build and publish her own labelling task on a public dataset — a written guideline, a double-labelled sample, a stated agreement figure and notes on how disagreements were resolved.

D. Move to a higher-paying platform and specialise in domain data, since rate is the only thing that compounds.

5 · 7 min

Reading a job advert for what it actually is

THE ONE THING TO KEEP

The tools list and the responsibilities tell you the real job; the title and the years-of-experience line usually do not.

Read the sections in the wrong order on purpose

A job advert has five parts: title, company blurb, responsibilities, requirements, benefits. Most people read top to bottom and decide from the title. Read it in this order instead: tools, responsibilities, requirements, title, blurb.

The tools list is the most honest section

Whoever wrote the advert may have been vague about the work. They were not vague about the stack, because the stack is what the team argues about. So the tools tell you the job:

- **Airflow, dbt, Snowflake or BigQuery, Kafka, Spark** — this is data engineering, whatever the title says.
- **SQL, Power BI or Tableau or Looker, Excel** — analytics. If Python appears once at the end, it is optional in practice.
- **PyTorch, CUDA, distributed training, experiment tracking** — genuine model work. Rare, and usually senior.
- **A hosted model API, a vector database, FastAPI or Node, Docker** — AI application engineering. This is the fastest-growing category and the most title-confused.
- **A CRM, a marketing platform, "prompt libraries"** — an operations or content role with AI in the title.

A posting titled "AI Engineer" with dbt and Snowflake in the requirements is a data engineering job. Apply to it as one.

The years-of-experience line

"3–5 years required" is usually a template inherited from the last requisition. It is a filter for the recruiter, not a law. The working rule most hiring managers will confirm privately: apply if you meet roughly sixty per cent of the requirements and can evidence the core two or three. Someone who meets a hundred per cent is applying to a job they have already outgrown, and interviewers can tell.

The exception is a hard gate written in regulation rather than preference — a security clearance, a professional licence, a visa rule that names a degree. Those are real. "Preferred: Master's degree" is not.

Reading the responsibilities for the shape of the day

Count them. Four to seven bullets describing one coherent activity is a real job someone has thought about. Fourteen bullets covering data engineering, dashboards, machine learning, stakeholder management and "supporting the marketing team's AI initiatives" is four jobs, one salary, and a person who will burn out in nine months.

Look for a first-ninety-days description. Its presence means the manager has planned for a human arriving. Its absence is not fatal but it is information.

Red flags, and the mechanism behind each

- **"Build our data infrastructure from scratch" in a junior role.** There is no senior person. You will make every beginner mistake with nobody to catch it, and the company will conclude data was a bad investment.
- **Machine learning expected, no warehouse or pipeline tool mentioned anywhere.** The data is in spreadsheets and application databases. Year one is plumbing, sold as ML.
- **"Fast-paced", "wear many hats", "self-starter" clustered together.** Usually true, and usually describing understaffing rather than energy.
- **The same advert reposted monthly for six months.** Either people keep leaving the role, or the specification is impossible and nobody will fix

it. Both are worth knowing.

- **No salary band in a market where bands are normally published.** In many European countries this is now a legal requirement; its absence is a signal.

Green flags

A named team and manager. A stated interview process with the number of stages. A described stack. A salary band. An honest sentence about what is hard about the role — rare, and almost always written by someone good.

The exercise worth doing this month

Open ten adverts you would plausibly apply to in your city or for remote roles you can legally take. Copy each requirements section into one text file. Count how often each tool appears.

That tally is your syllabus for the quarter. It is specific to your market, it costs nothing, and it beats any global ranking of "top AI skills" — those are written for traffic, and they will tell a person in Nairobi to learn the same things as a person in Seattle.

If you want to be slightly clever about it, save the postings as text and count with one command:

```
cat postings/*.txt | tr 'A-Z' 'a-z' | grep -oE
'sql|python|dbt|airflow|spark|tableau|power
bi|snowflake|docker|pytorch' | sort | uniq -c | sort -rn
```

Nothing paid is required for this. Paid market-intelligence products exist and sell you a national average; you have just measured the thing that applies to you.

The limitation

An advert describes what someone hoped for on the day they wrote it. Requirements soften during an interview, budgets change, and the role you

are offered is sometimes not the one advertised. Treat the advert as a strong hint about the team's stack and a weak hint about everything else.

BEFORE YOU MOVE ON

A posting is titled "Machine Learning Engineer". The responsibilities are building pipelines, maintaining a dbt project and supporting the analytics team; the required tools are SQL, Airflow, dbt and Snowflake; Python appears once, and no model or training framework is named anywhere. What should you conclude?

A. It is a data engineering role with a fashionable title, and you should prepare for it — and describe your experience for it — as data engineering.

B. The title is what matters for your CV and career path, so apply as an ML candidate and learn the pipeline tools later.

C. The company does not know what it wants, so the posting should be skipped.

D. It is bait for cheap labour, since ML work is being advertised at data engineering rates.

---

6 · 9 min

Who sits around you, and how work arrives

THE ONE THING TO KEEP

The organisation's shape and data maturity determine your daily work more than the job title does, and both can be diagnosed with one question in the interview.

The cast

Job adverts describe a role. They rarely describe the room. Most data and AI work happens inside a group that looks roughly like this:

- **A business stakeholder** who owns a number — revenue, churn, fraud losses, delivery times — and does not care how it is computed.
- **A product manager**, translating between that person and the team.
- **An analyst**, answering questions with SQL and charts.
- **An analytics engineer**, turning raw tables into clean, documented, tested models everyone else builds on.
- **A data engineer**, moving data from source systems into the warehouse and keeping the pipelines alive.
- **A machine learning or AI engineer**, when there is a model in production.
- **A platform or infrastructure person**, often shared across several teams.
- **A domain expert** — an underwriter, a clinician, a logistics planner — whose knowledge decides whether any of it is correct.

Small companies collapse several of these into one job, which is usually yours. That is not a warning. Doing four of them badly for a year teaches more than doing one of them well, and it is the fastest route to knowing which one you want.

Three shapes, three lives

How the group is arranged changes your day more than the tools do.

Centralised. One data team serves the whole company through a request queue. You see breadth and every part of the business, and you develop shallow domain knowledge because you are never anywhere long. Requests arrive as tickets. The team's constant problem is that it is a service desk with a backlog.

Embedded. Analysts and engineers sit inside product or business teams and report there. You get deep domain knowledge, real influence, and you are close enough to the decision to change it. The risks are isolation from other data people, no one senior to review your work, and career exposure if the product is cancelled.

Hub and spoke. Embedded people with a dotted line to a central team that owns standards, tooling and career development. Most organisations end up here after trying the other two. It is the best of the three for someone early, because you get a domain and a reviewer.

You can identify the shape in one interview question: *"If I need a number by Friday, who do I ask, and who checks it?"*

How work actually arrives

Almost never as a specification. It arrives as:

- A Slack message: "can you pull the numbers for the board deck".
- A ticket with a title and no body.
- A meeting where three people describe three different problems.
- An incident — a dashboard is wrong and somebody noticed in a leadership meeting.

The first, unteachable-sounding, entirely teachable skill of the job is converting a request into a question with a decision attached. The move is one sentence: *"What will you do differently depending on the answer?"* If nobody can answer that, the work is decoration and can wait. If they can, you have just learned what precision the answer needs. A decision to spend fifty thousand rupees needs a rougher number than a decision to change pricing for every customer.

The maturity ladder, and why it decides your year

Data functions sit on a ladder, and where a company sits determines what you will actually do:

1. **Spreadsheets.** Numbers live in files people email. Your job is extraction and reconciliation.
2. **A database or warehouse exists** but nothing is documented. Your job is archaeology and writing the same joins repeatedly.
3. **A transformation layer** — dbt or its equivalent — with tested, documented models. Your job becomes analysis rather than plumbing.

4. **A metrics layer**, so "active customer" means one thing everywhere. Arguments about definitions stop.
5. **Models in production**, with monitoring and retraining.

An advert for an AI engineer at a company on rung two is really an advert for a data engineer who will spend the first year at rung three. That is not a bad job. It is simply a different one from the one advertised, and taking it without knowing is how people end a year unable to explain what they learned.

Diagnose it with a single question in the interview: **"Where does the number on your main dashboard come from, end to end?"** A confident four-sentence answer means rung three or higher. Hesitation, or "it depends who you ask", means rung one or two. Both are worth joining. Only one of them matches the advert.

BEFORE YOU MOVE ON

In an interview, a candidate asks where the number on the company's main revenue dashboard comes from. The hiring manager says three teams maintain slightly different versions and finance reconciles them monthly in a spreadsheet. What does this most reliably tell the candidate?

A. The company is disorganised and the role should be avoided.

B. The team lacks a metrics layer and probably a tested transformation layer, so the first year will be plumbing and definition work regardless of what the advert called the role.

C. The role will involve building machine learning models on top of finance data.

D. The company needs better dashboard software and the candidate should propose a tool in the interview.

7 · 6 min

Who actually needs a degree

THE ONE THING TO KEEP

A degree opens doors in some markets; evidence that you can build is what keeps you inside.

The answer changes by role

Research scientist. A PhD is effectively required. Not because the maths is unlearnable elsewhere, but because the hiring signal labs use is published work, and publishing is what a PhD is for. This is the one role where the credential is close to a hard gate.

ML engineer. A degree helps and is not required. What is required is that you can write production code, reason about systems, and show something you shipped. Plenty of people arrive from backend engineering with no ML coursework at all.

Data engineer. The least credential-bound serious role in the field. It is software engineering with databases. Employers test it, and tests do not care where you studied.

Data analyst. Wide open. The gate is SQL and the ability to explain a number to someone who does not want to hear it.

AI product. Bound by experience, not education. Very hard as a first job, very reachable as a second or third.

What a degree actually buys

Three distinct things, and people confuse them constantly.

1. **Legal eligibility.** For a work visa in many countries, a recognised degree is not a preference but a requirement in the rules. A US H-1B, a German or

EU Blue Card, several Gulf work permits. If your plan involves relocating, check the actual regulation before you plan around a portfolio.

1. **Passing an HR filter.** Large banks, telecoms, government bodies and consulting firms filter applications on a qualifications field before any human looks. Your project never gets opened. This is not a judgement about your skill; it is a form field.

1. **Actual knowledge**, mostly for research: linear algebra, probability, optimisation, and the habit of reading papers. Real, and learnable outside a university with a great deal more discipline than most people have.

Notice that only the third is about learning.

Certificates and bootcamps, honestly

A cloud certification (AWS, Azure, GCP) has genuine currency in enterprise and consulting hiring in India, the Gulf, Nigeria and parts of Europe, because staffing firms bill clients partly on certified headcount. That is a real, narrow, commercial reason.

Outside that, a certificate mostly tells an employer that you paid for a course. It is not worthless — it is weak. It cannot carry an application by itself, and no volume of them adds up to one thing you built.

Bootcamps vary enormously and their outcome claims are marketing. If you consider one, ask for the placement rate calculated over everyone who enrolled, not everyone who graduated, and ask which employers hired last year's cohort by name.

If you do not have a degree

Aim at the places where the filter is not there: startups, small and mid-size companies, agencies, remote-first firms, and internal moves inside a company that already employs you. Skip the graduate schemes at large corporates; they exist to hire graduates and no amount of GitHub changes that.

And be precise about what happened when you get rejected. "Rejected in forty minutes with no interview" is a filter. "Rejected after a technical round" is

feedback. Only the second one is about you.

BEFORE YOU MOVE ON

Priya has no degree, a deployed tool used daily by a real clinic, and clean code on GitHub. A large bank rejects her forty minutes after she applies. Three startups interview her the same week. What is the most accurate reading?

A. Portfolios carry no weight in enterprise hiring; only credentials count there.

B. The bank's first filter is a rule about qualifications applied before any human reads anything, and the startups have no such rule.

C. Bank work requires deeper mathematics than startup work, and the degree is a proxy for that.

D. Startups have lower hiring standards, which is why they were willing to talk to her.

8 · 8 min

Choosing a lane you can defend, and changing it later

THE ONE THING TO KEEP

Choose the lane where you can produce evidence fastest, because lanes are adjacent and moving between them costs one project, not one degree.

Four lanes, not twenty

Beginners try to choose between fifteen job titles. There are four practical lanes for someone starting, and the titles sit inside them.

Analytics. SQL, spreadsheets, dashboards, and the ability to explain a number. Widest door, fastest to evidence, lowest ceiling early and a perfectly good ceiling later.

Data engineering. Pipelines, schemas, orchestration, reliability. More jobs than any other lane, more software engineering, longer runway to a first job unless you already program.

AI application engineering. Building products on top of models someone else trained: retrieval, APIs, evaluation, deployment. Newest lane, fewest settled conventions, most reachable if you can already write code and ship a web service.

AI-adjacent work that is not engineering. Product, operations, evaluation and annotation leadership, technical writing, solutions and customer-facing work, teaching. Real jobs, frequently overlooked, and often the best fit for someone with strong domain experience and no programming background.

The three questions that pick one

Where do you already have context? Ten years in pharmacy, insurance claims, school administration or freight is an asset that a computer science graduate cannot acquire quickly. It points at a domain, and the domain often points at a lane — claims work points at analytics, freight at data engineering.

What can you evidence in eight weeks? Not learn — evidence. A finished, deployed, written-up thing.

What is actually hired where you are, or where you can legally work remotely? You measured this in the last lesson. If your city has forty analytics openings and two ML ones, that is not a conspiracy against your interests; it is the shape of your market.

Honest time-to-first-evidence

Part-time, around six hours a week, starting from ordinary computer literacy:

- **Analytics:** eight to twelve weeks to a defensible project. SQL is learnable to working level in weeks.
- **AI application engineering:** three to four months if you already program, eight or more if you do not.

- **Data engineering:** four to six months, because it needs programming plus systems plus at least one orchestration tool.
- **Research:** years, and a different path entirely.

These are ranges from watching many people do it, not guarantees. Some move faster. Almost nobody moves faster than half of these.

The adjacency map

The reason this choice is less frightening than it feels: the lanes touch.

- Support, operations or finance → analyst. Very common.
- Analyst → analytics engineer → data engineer. Very common.
- Data engineer → ML engineer. Common, and much easier than the reverse.
- Backend engineer → AI application engineer. Short hop.
- Analyst → product. Common at three to five years.
- Anything → research. Rare without a formal route.

Notice what is missing: a direct jump from courses to ML engineering. It happens, but it is the exception people write posts about, which is exactly why it looks frequent.

What transfers when you move, and what does not

Transfers: the domain you learned, the habit of finishing, your written record of work, SQL, debugging, and the thirty people who have seen you work.

Does not transfer: certificates, course completions, and tool-specific fluency in a tool the next team does not use.

The practical consequence is that the switching cost is a project and a job change — six to eighteen months — not a restart. People treat a lane choice as though it were a degree subject, then delay for months rather than pick. The delay costs more than a wrong pick would have.

The failure mode that wastes a year

Studying three lanes at thirty per cent each. It feels prudent and it is the single most common way a promising year produces nothing hireable, because employers hire on depth in one thing and nobody is hired for breadth at zero years.

A workable rule: **one lane for six months.** Then reassess on evidence, not mood — did you finish something, did anyone use it, did anyone reply to you about it. Boredom in week three is not evidence.

What to do if you cannot get any job at all

Then the lane is chosen for you, and that is fine. Take the job that exists, then use the bridge-role method in the next module to move toward data from inside. Choosing a lane is a luxury of having options, and building options is the actual first task.

The limitation

None of this survives contact with an unusual opportunity. If someone offers you paid work in a lane you did not choose, take it. Paid work in the wrong lane beats unpaid preparation in the right one, every time, because it converts you from a candidate into someone with a reference.

BEFORE YOU MOVE ON

Rahel has eleven years in hospital administration, no programming, six hours a week, and wants out of admin within a year. She is deciding between teaching herself deep learning and teaching herself SQL and reporting. Which reasoning is soundest?

A. Deep learning, because AI roles pay more and hospital data is exactly what modern models are built for.

B. Neither yet — she should first complete a broad foundation covering statistics, programming, machine learning and data engineering.

C. SQL, but she should leave hospital work behind entirely, since her admin experience will not count in a technical role.

D. SQL and reporting, because it is the lane where she can produce a finished, defensible piece of work within about three months, and her eleven years of hospital context makes her a stronger candidate for health analytics than a generalist with better maths.

9 · 8 min

Levels, titles, and why "senior" means nothing on its own

THE ONE THING TO KEEP

Levels are set by how much ambiguity you absorb and how far your work travels, so negotiate the level rather than the salary and read adverts for scope rather than titles.

Two vocabularies, and only one of them is real

Every company has a **title** it prints on your email signature and a **level** it uses internally for pay, promotion and headcount. Titles are marketing. Levels are the machinery.

Large technology companies run numbered ladders — L3, L4, L5, or SDE I, II, III, or IC1 through IC6. Banks and insurers run grades. Consultancies run analyst, consultant, manager, principal. Start-ups often run nothing at all and hand out titles at hiring time because a title is free and salary is not.

The consequence is that a Senior Data Scientist at a thirty-person company and a Data Analyst II at a bank may be doing identical work, and the bank's analyst may have more responsibility. Neither title tells you anything until you know the scope behind it.

What actually separates levels

Not years. Not tools. Not how many algorithms you can name. Levels are defined by **how much ambiguity you absorb** and **how far the consequences of your work travel**.

- **Junior.** Given a well-defined task, you produce a correct result and say when you are stuck. Somebody else decided what to do and reviews what you did.
- **Mid.** Given a problem, you choose the approach, do it, and finish it without supervision. You are trusted to notice when the approach is wrong halfway through.
- **Senior.** Given a vague area, you define the problem, get other people to agree it is the right problem, and deliver something that keeps working after you stop looking at it. Much of your output is other people's changed behaviour.
- **Staff or principal.** You change what the organisation works on. You are measured on decisions avoided as much as work done.

Notice that only the first level is about producing correct code or correct numbers. Everything above it is about deciding what should be produced. That is why people plateau at mid-level with excellent technical skills, and why the fix is never another framework.

What actually separates levels

Staff or principal: changes what the organisation works on	measured on decisions avoided as much as work done
Senior: given a vague area, defines the problem and gets agreement	impact is largely other people's changed behaviour
Mid: given a problem, chooses the approach and finishes without supervision	trusted to notice halfway through that the approach is wrong
Junior: given a well-defined task, produces a correct result and says when stuck	somebody else decided what to do and reviews it

Only the first layer is about producing correct code or correct numbers. Everything above it is about deciding what should be produced, which is why people plateau at mid-level with excellent technical skills.

The manager fork is a fork, not a step

At roughly senior level most organisations offer two paths: keep building, or manage people. They are not up and further up. They are different jobs with different daily content — one-to-ones, hiring, performance conversations, planning — and the skill overlap is smaller than people expect.

Two things worth knowing before you take the fork. First, many companies pay the individual-contributor track and the management track equivalently for several levels, so managing is not a raise. Second, returning is possible but costs you time: a manager who stops writing code for three years and goes back typically re-enters a level below where they left.

Title inflation, and what it costs later

Small companies inflate titles because a title has no cash cost. A twenty-person start-up will happily call a person with three years of experience Head of Data.

This is fine right up until you apply somewhere else. Two things then happen. A large employer maps you to its own ladder using scope, not your title, and frequently offers you a level below what your title implied — this is called down-levelling, and it is routine, not an insult. Worse, an inflated title on a CV invites interviews for jobs that expect the scope the title claims, and those interviews end badly for everybody.

The same effect runs across markets. Moving from a services firm to a product company, or from one country to another, usually costs a level, because scope in the new place is measured differently. Expect it and it stops feeling personal.

Two practical uses

When reading an advert, translate the responsibilities into the ambiguity ladder above and ignore the title. "Partner with stakeholders to define measurement strategy" is senior scope whatever the header says. "Build reports according to specification" is junior scope even if the header says Senior Analyst. Apply to the scope you can actually hold, plus one step.

When negotiating, negotiate the level first. Salary bands are attached to levels, and a recruiter usually cannot move you far inside a band but can sometimes place you at the next one. Asking "which level is this role, and what distinguishes it from the one above?" is a normal question, it signals that you understand how the company works, and the answer tells you your realistic ceiling before you spend two weeks on interviews.

BEFORE YOU MOVE ON

Daniel has been Head of Data at a twenty-five person start-up for two years, managing nobody and building dashboards and pipelines himself. A 4,000-person company interviews him and offers a role at their mid-level engineer grade. What is the most accurate reading?

A. The offer is an insult and he should decline and hold out for a leadership title.

B. The company is mapping his actual scope — individual delivery, no organisational influence — onto its own ladder, which is normal, and the useful question is what distinguishes that grade from the one above it.

C. He was underpaid at the start-up and the new grade proves it.

D. He should ask to skip the grade because he already held a head-of title.

10 · 8 min

A year on six hours a week

THE ONE THING TO KEEP

Progress comes from a small weekly amount that survives bad weeks, because every gap longer than about ten days costs a session to re-enter.

Plan for the time you actually have

Most people reading this have a job, or a family, or both, and four to eight hours a week. A plan that assumes forty is not a plan; it is a reason to feel guilty. Here is what a year at six hours looks like when it works.

The year, in blocks

Weeks 1–8 — the tool. Three hours practising, three hours building something tiny with it. For analytics that is SQL against a real dataset. For application work it is one small service that runs.

Weeks 9–16 — the mess. One genuinely messy real dataset. Not a cleaned teaching set. Municipal budgets, transport timetables, exam results, your own bank exports. Clean it, and write down every decision you made about it as you go, because you will not remember in April what you decided in February.

Weeks 17–26 — ship it and find one user. Deploy it. Get one real person to use it weekly. One is enough; one changes what the project is.

Weeks 27–36 — the second project, with a measurement. In your domain. This time build a test set of thirty cases and report a number.

Weeks 37–44 — the search apparatus. CV, case study pages, the list of forty employers, twenty outreach messages. This is work, and it needs its own weeks rather than being squeezed around the edges.

Weeks 45–52 — interviews, and keep building. The building matters here because it is what you talk about, and because a search with nothing else in it is corrosive.

That is fifty-two weeks with roughly ten weeks of slippage built in, because you will lose weeks to illness, deadlines and family. The plan assumes it.

Three rules that do more than the plan

The same slot every week. A fixed time defended against everything else beats a flexible plan that loses to whatever else appears. Tuesday 20:00–22:00 and Sunday 07:00–11:00 is a real schedule. "When I get a chance" is not.

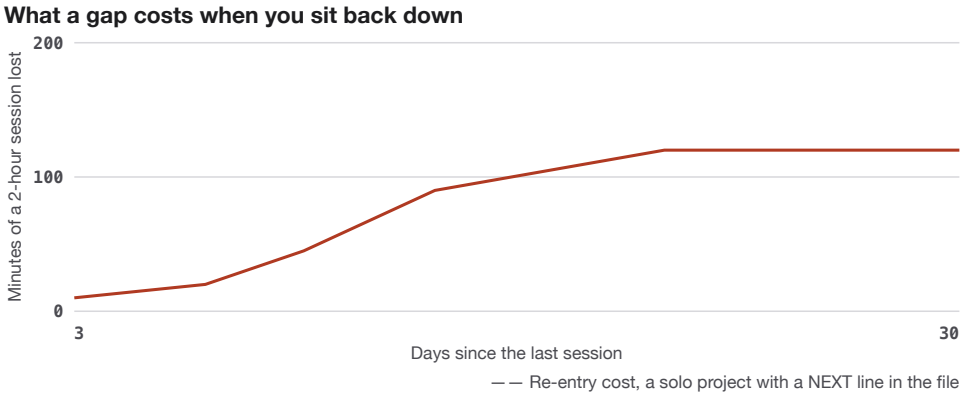
End every session by writing the next action. One line, in the file: `NEXT: the join is duplicating rows — check whether order_items has more than one row per order.` This is the single highest-return habit in the whole plan. Without it, the first twenty minutes of every session are spent re-building context.

Never start a session by reading news. Feeds are engineered to hold you, your two hours will become forty minutes, and you will feel as though you studied.

The re-entry cost, which is the real reason bursts fail

A three-week gap does not cost three weeks of progress. It costs a whole session, because you must re-read your own code and rediscover what you were doing before you can add one line. Two gaps a month and half your year is spent re-entering.

That is the mechanism. It is why two hours weekly beats twelve hours once a month, even though the monthly total looks similar. Memory of your own reasoning decays faster than memory of facts.



Past about ten days the whole of a two-hour session goes on re-reading your own code and rediscovering what you were doing. Two such gaps a month and half the year is spent re-entering, which is why two hours weekly beats twelve hours monthly.

If your only computer is a phone

A large share of the people reading this are on a phone. Be honest about what that allows.

Genuinely possible on a phone: reading; SQL practice on sites that run in a browser; writing your case study; drafting outreach messages; reviewing code on GitHub; Google Colab and Kaggle notebooks in a mobile browser, awkwardly.

Not realistically possible: building and debugging anything of size. Not because you lack discipline — because editing, running and reading errors in a 6-inch viewport costs you several times the keystrokes and most of your working memory.

So the plan needs a few keyboard hours a week from somewhere: a cybercafé, a public library, a friend's machine, your workplace computer at lunch, a college lab. Two focused keyboard hours plus four phone hours is a working week. Pretending a phone is enough is how a year disappears.

The free tools that make those keyboard hours count, because they need no installation and no payment:

- **Google Colab** — free Python with a browser, sessions time out after a period of inactivity and the disk is wiped, so push to GitHub before you close it.

- **Kaggle notebooks** — free, roughly thirty GPU hours a week, and datasets already attached.
- **GitHub Codespaces** — a real editor and terminal in a browser, on a free monthly allowance of core-hours.
- **DB Fiddle, SQLime, or DuckDB in the browser** — SQL with nothing installed.

The paid alternatives — a laptop, a cloud VM, a subscription notebook service — are better and are not required to start.

Measure finished things, not hours

Hours studied is a comforting and useless number. Count things finished: a query that answers a real question, a script that runs on a schedule, a page written, a message sent, a bug found. A month with zero finished things did not happen, no matter how many hours went into it.

The limitation

This plan produces a hireable beginner in a market that is hiring beginners. It does not defeat a market with no junior openings. If twelve months of honest work produces nothing, the problem is more likely the market or the employer categories you are targeting than your effort — go back to the employer map before you conclude anything about yourself.

BEFORE YOU MOVE ON

Tomás has four hours a week. He does nothing for three weeks while a work deadline runs, then puts in twelve hours over one weekend to catch up. After four months he feels he has learned almost nothing, though his total hours match a steady schedule. What is the main mechanism?

A. Twelve-hour sessions exceed his concentration limit, so most of that weekend was low-quality study.

B. Each long gap forces him to spend most of the next session rebuilding context — re-reading his own code and rediscovering his reasoning — so a large share of his total hours is spent returning to where he already was.

C. He is studying the wrong material, and a better curriculum would make the schedule irrelevant.

D. Four hours a week is simply too little, and there is no schedule that works at that level.

CASE STUDY · MODULE 1

Two offers, one title, and the question about the dashboard

Sneha, twenty-three, a commerce graduate in Nashik with eight months of evening SQL practice, holding two offers in the same week.

Offer A was from a fourteen-person edtech company in Pune. The title was **AI Engineer**. The money was ₹6,50,000 CTC. The founder had spent twenty minutes of the interview describing what they would build with language models next year.

Offer B was from the Pune office of a Dutch insurer — a global capability centre with about nine hundred people, a category Sneha had not known existed six weeks earlier. The title was **MIS Executive**. The money was ₹5,80,000 CTC.

Her mother had one question about each, and it was the same question: which one has AI in it.

What the adverts said when read backwards

Sneha had learned to read the tools list first. The startup's requirements named a hosted model API, Google Sheets, Zapier and "prompt libraries", and the responsibilities section ran to fourteen bullets covering dashboards, marketing support, customer research and "AI initiatives". The insurer's advert named SQL, dbt, Snowflake and Airflow, listed four responsibilities that described one coherent activity, gave the manager's name, and described the first ninety days.

She also noticed something she nearly did not mention to anybody: the start-up's advert had been reposted every month since April.

The question she asked in both interviews

Where does the number on your main dashboard come from, end to end?

The founder said it depended who you asked, and laughed, and said that was exactly the problem she would be hired to fix. The insurer's team lead answered in four sentences: the policy administration system lands in Snowflake by 04:00, twenty-six dbt models run against it, forty-one tests run after that, and if a test fails the dashboard shows yesterday's data with a banner rather than a wrong number.

One of those answers describes rung two of the maturity ladder. The other describes rung three or four.

The cost on both sides

Taking the startup meant twelve per cent more money, a title that would look serious on a CV, and touching a model API in her first month. It also meant being the only technical person on data, with nobody to review her work, in a company where the first year would be spreadsheets and one integration — and where the advert being reposted monthly suggested she would not be the first person to discover that.

Taking the insurer meant a title that sounded like a clerk's, no model anywhere near her desk for at least a year, and the slow, procedural culture of a large European employer. It also meant a named manager, a team of six, a reviewer for everything she wrote, and a system she could learn deeply.

The startup wanted an answer within twenty-four hours. The insurer had given her a week.

What she was actually choosing between

Not two jobs. Two first years. One of them would produce a person who had spent twelve months doing four things badly with nobody to correct her, holding a title that would cause a large employer to down-level her at the next application anyway. The other would produce a person who could say, in an interview, exactly where a number came from and what happens when the pipeline fails.

What she could not know was whether the startup would be the exception — the small, chaotic place where somebody learns five years of work in two.

STOP HERE

Before turning the page: what would you have done, and what would it have cost if you were wrong? Write it down. The point of the next section is to compare.

WHAT HAPPENED

She took the insurer, and asked for the offer in writing before telling the start-up. Her first eight months were SQL, dbt models and a rota for checking test failures — no models, no AI in the work at all. In month eleven she moved internally onto the data platform team after answering a question in a meeting with a number and a source, which is the only interview that move required. The startup's advert was reposted twice more that year. Two years later she interviewed at a product company and was mapped to their mid-level band on scope; the interviewer's only note about her CV was that she had a system she could describe end to end, which almost nobody applying at that level could.

Sneha's mother was right that the word AI matters to employers. Where in the process does a title actually help, and where does it stop helping?

A title helps at the top of a funnel — it gets a CV opened and it matches the words a recruiter is filtering on. It stops helping the moment somebody maps her to their own internal level, because large employers map on scope rather than on the title she held. An inflated title also invites interviews for jobs whose scope she cannot hold, and those interviews are miserable and consume the weeks she needed for the ones she could pass.

The founder said the vague answer about the dashboard was the problem she was being hired to solve. Why is that answer not enough on its own?

It is a truthful description of the job and it is also a description of rung one or two of the maturity ladder with no senior person on it. The work is real and worth doing, but the advert sold model work and the year would be extraction and reconciliation. The question is not whether the job is legitimate; it is whether she would end the year able to explain what she had learned, and with somebody having reviewed any of it.

What would have changed the decision in the startup's favour?

A senior technical person she would report to, an honest advert that described the plumbing rather than the models, and an advert that had not been reposted monthly for five months. Any one of those changes the risk materially; the first changes it most, because the defining cost of a junior role with no reviewer is a year of habits nobody corrects.

MODULE 1 · TEST

Check what stuck

6 questions, one right answer each. This one is for you, not for a record: answers and the reason behind each are at the back. If two questions from the same module go wrong, re-read that module before the exam rather than after.

1 You have forty applications to spend this month. Why do enterprise internal data teams and IT services firms usually give a better return per application than a well-known technology brand?

A. Far fewer people apply per opening, because those firms run no developer marketing.
B. They pay above the market, which makes candidates self-select away from them.
C. They hire purely on portfolios, so qualifications are not filtered at any stage.
D. They do not use tracking systems, so a human being reads every application that arrives.

2 A San Francisco team requires overlap from 09:00 to 13:00 Pacific time. For which candidate is that a night job rather than an inconvenience?

A. São Paulo, where it falls in the early afternoon
B. Lagos, where it falls in the evening
C. Bengaluru, where it falls between 21:30 and 01:30
D. Warsaw, where it falls late in the evening and working-time rules may apply

3 An advert is titled "AI Engineer". The requirements name dbt, Snowflake, Airflow and strong SQL, with Python mentioned once at the end. What job is being advertised?

A. A prompt and evaluation role, because AI is in the title
B. A research position, since Airflow is used to schedule training runs
C. An analytics role, because SQL appears before the other tools
D. A data engineering job, whatever the header says

4 Fully remote junior roles for foreign employers became scarcer while remote senior roles did not. What is the mechanism?

A. Junior work needs supervision that cannot happen over a video call.
B. An employer-of-record fee is a fixed per-head cost, so it weighs far more against a junior salary.
C. Immigration rules were tightened for remote workers in most countries.
D. Junior candidates apply in such overwhelming numbers that employers stopped advertising the roles publicly.

5 Why does two hours every week beat twelve hours once a month, even though the monthly total is similar?

A. Facts are retained better when they are revisited at spaced intervals rather than in one long block.
B. Twelve-hour sessions exceed the point at which concentration is productive.
C. A gap costs a whole session, because you must rediscover your own reasoning before adding a line.
D. Employers can see commit dates and prefer a steady contribution history.

6 Two people have the same tools and the same five years. One is levelled senior and one is levelled mid. What distinguishes them?

A. The senior one is given vague areas and defines the problem others then agree to.
B. The senior one writes code that is correct more often and needs less review.
C. The senior one knows more algorithms and more of the current tooling.
D. The senior one has managed people, which is the step above individual contribution everywhere.

MODULE 2

The roles, one at a time

One overview lesson cannot tell you whether you would enjoy being a data engineer. This module takes ten roles and gives each its own hour — the ordinary week, the thing you produce, the tools with their free equivalents, the two routes in, and the failure that ends people's first year in that seat. Read the three that sound closest and skim the rest, because knowing what you are not applying for is half of choosing.

> BY THE END OF THIS MODULE YOU CAN
>
> Describe one role's ordinary week, the artefact it produces, the stack it runs on and two realistic routes into it in enough detail that somebody doing that job recognises their own week in your description.

---

11 · 9 min

The data analyst

> THE ONE THING TO KEEP
>
> An analyst's value comes from owning what the words mean and being accountable for the number, not from producing charts faster.

The week

Monday, a message: "revenue looks off in the north region, can you check". Tuesday, you find the region field changed meaning in March when the sales team reorganised, and half the comparison is nonsense. Wednesday, you re-

build the comparison on a stable definition and it turns out revenue is fine but one large customer moved their billing date. Thursday, you write four paragraphs and a chart. Friday, somebody asks the same question about the south region.

That is the job. Not the chart. The chart takes twenty minutes. The three days before it are the work, and they are almost entirely about establishing what a word means.

What you produce

An answer somebody acts on, in writing, with the caveats attached. Secondarily, dashboards — which are answers to questions asked repeatedly, frozen so nobody has to ask you again.

The unit of value is a decision changed. An analyst who produces forty dashboards nobody opens is less valuable than one who produced three memos that stopped a bad launch. This is the hardest thing to believe early, because the dashboards are visible and the memos are not.

The stack

- **SQL.** Non-negotiable, and it is most of the job. Everything else is optional.
- **A BI tool** — Tableau, Power BI, Looker in industry. Free equivalents that teach the same concepts: Metabase, Apache Superset, Google Looker Studio, or plain matplotlib in Python. A hiring manager cares whether you can build a chart that answers a question, not which licence you held.
- **A spreadsheet.** Excel or Google Sheets. Do not sneer at this; see the spreadsheet lesson in the evidence module.
- **Python or R**, at the level of loading a table, cleaning it and plotting it. Pandas is enough for years. R with the tidyverse is entirely respectable and still dominant in pharmaceuticals, biostatistics and parts of academia.
- **Statistics** at the level of proportions, distributions, sampling error and what a confidence interval means.

Two routes in

From inside a company. The most common entry in the world. You are in operations, finance, support, marketing or logistics; you already know the domain; you learn SQL and take over the reporting nobody wants to do. Domain knowledge is the expensive half and you already have it. This route is invisible on job boards, which is why people underestimate how many analysts arrived by it.

From outside, with public data. Pick a dataset with a genuine question attached — your city's transport authority, a national statistics office, a public health registry — and publish three analyses that end in a recommendation. The bar is not sophistication. It is whether a reader can tell what you would do differently on Monday.

The failure that ends the first year

Answering the question you were asked. A stakeholder says "how many users churned last month". You produce the number. Then they say "but that includes trial accounts", and you produce it again, and then "that counts anyone who did not log in, but our contract runs annually", and you produce it a third time.

Three days gone because nobody wrote down what "churn" means. The senior version of you opens with a sentence: *"Before I run this — churn meaning contract not renewed, or meaning no activity for thirty days? They will give different answers and I want the one you will act on."*

Owning definitions is what separates an analyst who is trusted from one who is a query service. Write them down where others can find them. A one-page document listing the ten metrics your team argues about, each with a definition and the SQL that computes it, will do more for your standing than any dashboard.

The honest limitation

Analyst work is the most exposed part of this field to automation, and for a specific mechanical reason: much of it is text-to-SQL over a schema, and mod-

els are genuinely competent at that. What they are not competent at is knowing that the region field changed meaning in March, that the finance team's definition of revenue differs from the sales team's, or that the person asking will act on the answer in a board meeting where being confidently wrong is expensive.

The defence is not faster querying. It is being the person who owns the definitions and is accountable for the number. That part is not going anywhere, and it is a completely different daily practice from writing more queries.

How to tell a good version of this job from a bad one

Ask what proportion of requests arrive with a decision attached, and ask to see the team's metric definitions. If definitions live in one place and are argued about openly, the job is analysis. If they live in people's heads and each team has its own, you are joining an archaeology project — which can be excellent early-career work, but only if you know that is what you signed up for.

BEFORE YOU MOVE ON

A stakeholder asks an analyst for "last month's churn rate" and acts irritated when the analyst asks a question back. Which opening question is most likely to save time rather than waste it?

A. "Do you want this in Tableau or as a spreadsheet?"

B. "Should I use last calendar month or the trailing thirty days?"

C. "Churn meaning contract not renewed, or no activity for thirty days? Those give different answers and I want the one you will act on."

D. "I will pull it now and we can refine it if the number looks wrong."

12 · 9 min

The analytics engineer

THE ONE THING TO KEEP

Analytics engineering makes data trustworthy by treating transformations as idempotent, version-controlled, tested code, and its only real success metric is whether other people build on your tables.

A role that did not exist in 2015

Analytics engineering appeared because two jobs were failing at their boundary. Data engineers delivered raw tables and stopped. Analysts wrote the same 200-line join in fifteen different notebooks, each slightly different, and the company had fifteen answers to one question.

The analytics engineer sits in that gap. They take raw, ugly source data and turn it into a set of clean, tested, documented tables that everybody else builds on. They write almost no application code and almost no machine learning. They write SQL, in a disciplined way, with version control and tests.

The week

You own a directed graph of transformations. Raw orders, customers and payments come in at the top. At the bottom sit a handful of tables the whole company uses: `dim_customer`, `fct_orders`, `fct_subscription_events`. In between are dozens of intermediate models, each one doing one comprehensible thing.

An ordinary week: a source system adds a column and silently changes an enum value; a test fails at 6am and you find out before the sales team does; an analyst asks for a field that does not exist and you decide whether to add it here or tell them to compute it themselves; you rename something and spend an afternoon finding everything downstream that depended on the old name.

The idea that carries the role

A transformation should be idempotent and re-runnable. Run it once, run it ten times, delete the output and run it again — the result must be identical. That single property is what makes a data platform trustworthy, and it is why analytics engineers rebuild tables from source rather than updating them in place.

The second idea is that **data can be tested like code**. Not "does the query run", but assertions about the data itself:

```
-- every order must belong to a customer that exists
select o.order_id
from fct_orders o
left join dim_customer c on c.customer_id = o.customer_id
where c.customer_id is null
```

If that returns any rows, the build fails. A team with a hundred such tests finds out about broken data from a red build at 06:15, not from a director in a meeting at 11:00. This is the whole value proposition of the role, stated in one sentence.

The stack

- **dbt** is the industry standard, and dbt Core is free and open source; the paid Cloud product adds scheduling and a web interface you do not need to learn the craft. Alternatives: SQLMesh, or plain SQL files run by a scheduler.
- **A warehouse** — Snowflake, BigQuery, Databricks, Redshift. To practise free: DuckDB on your laptop reads CSV and Parquet files and speaks proper analytical SQL, and BigQuery and Snowflake both have free tiers.
- **Git**, seriously. This role lives in pull requests.
- **A scheduler** — Airflow, Dagster, Prefect, or a cron job. All have free open-source versions.

You can build a complete, credible analytics engineering portfolio project on a laptop with DuckDB, dbt Core and a public dataset, at zero cost. Very few ap-

plicants do, which is exactly why it works.

Two routes in

From analyst. The overwhelmingly common one. You are already writing the SQL; you add version control, tests and a transformation tool. Six focused months is realistic.

From software engineering. You already have Git, testing and code review; you learn the warehouse and the dimensional modelling ideas — facts, dimensions, grain, slowly changing dimensions. Kimball's dimensional modelling vocabulary is forty years old and still what these tables are named after.

The failure that ends the first year

Building a beautiful graph nobody uses. It is possible to spend four months producing 180 exquisitely tested models while analysts keep querying the raw tables directly, because your models do not contain the field they need and nobody asked them.

The role is a service role. Its output is other people's speed. If analysts are not migrating onto your models, that is the only metric that matters and everything else is decoration.

The honest limitation

This is a genuinely enjoyable job with a narrow ceiling in most companies. One or two analytics engineers can serve a large organisation, so there are fewer senior seats than in data engineering or analysis, and the progression is usually sideways — into data engineering, into platform work, or into leading a data team. Worth knowing before you specialise hard at 23.

BEFORE YOU MOVE ON

An analytics engineer has built 180 tested models over four months. Analysts still query the raw source tables directly. What is the correct conclusion?

A. The analysts need training on the new models and the fix is documentation and a workshop.

B. The project has failed at its actual purpose so far, because the only measure of the work is whether others build on it — and the first question is which fields analysts need that the models do not contain.

C. Access to raw tables should be revoked so analysts are forced onto the modelled layer.

D. The models are fine and the tests prove it; adoption will follow naturally as the team grows.

13 · 10 min

The data engineer

THE ONE THING TO KEEP

Data engineering is judged on availability and silence, so idempotency, backfills and loud failure matter more than any tool on the CV.

The job is availability, not cleverness

A data engineer is responsible for the fact that the data is there, on time, complete, and the same as it was yesterday. Nobody thanks you when this works. Everybody notices within eleven minutes when it does not.

The mental model that helps: you are running a small logistics company. Things arrive from suppliers you do not control, at times they choose, in formats they change without telling you, and your customers downstream have already promised their customers that the goods will be on the shelf by 07:00.

The week

Monday's pipeline failed at 03:40 because an upstream API returned a 500 for eleven minutes. Your job is not to be awake at 03:40; it is to have written the retry so that nobody had to be. Tuesday, marketing wants three years of history for a new source, so you write a backfill and discover that running it will cost more in compute than the analysis is worth. Wednesday, a source system starts sending timestamps in local time instead of UTC and nothing errors — the numbers are just quietly wrong for a day. Thursday, you add a schema check so that particular silence cannot happen again. Friday, capacity planning.

The ideas that carry the role

Idempotency. Re-running yesterday's job must produce yesterday's result, not double it. This is why data engineers think in partitions: you replace the 3 March partition rather than appending to a table.

Backfills are a first-class feature, not an emergency. Any pipeline you build should be runnable for an arbitrary past date range on day one, because you will need it in month two.

Failures are loud or they are worse. A pipeline that crashes is a good pipeline. A pipeline that silently writes half the rows is the one that costs a quarter of trust. Every stage gets a row-count check, a freshness check and a null-rate check.

Data contracts. An agreement with the source team about schema, meaning and notice before changes. Mostly social, occasionally enforced in code, and the difference between a calm team and a firefighting one.

The stack

- **Python and SQL.** Both, properly.
- **Orchestration** — Airflow, Dagster or Prefect. All free and open source; managed versions cost money and change nothing about the concepts.

- **Storage and compute** — Snowflake, BigQuery, Databricks in industry. Free at home: PostgreSQL, DuckDB, and Parquet files on disk.
- **Streaming**, in some jobs — Kafka, or the managed equivalents. Redpanda and Kafka itself run free locally.
- **Containers and some infrastructure** — Docker, a little Terraform, enough Linux to read a log.
- **Spark**, if you are near large volumes; PySpark runs free on a laptop, and increasingly does not need to, because DuckDB and Polars handle on one machine what needed a cluster in 2018.

Two routes in

From software engineering or IT. The shortest path. You already have code, Git and operational instincts; you learn warehouses, orchestration and the failure modes of moving data.

From analyst or analytics engineer. Also common. You know the data and what downstream users need; you learn Python properly, testing, containers and how to be on call.

The portfolio piece that works: a pipeline that runs on a schedule, pulls from a real public API, is idempotent, has data-quality checks, fails loudly, and has a README explaining what happens when the source is down. Hosted free on a small virtual machine or a scheduled GitHub Actions workflow. This is the closest thing in the field to a project that reliably converts to interviews, because it is exactly the job.

The failure that ends the first year

Building for a scale you do not have. It is very easy to reach for Kafka, Spark and a Kubernetes cluster for 40 million rows that DuckDB would handle in eight seconds on a laptop. You then spend your year operating infrastructure instead of delivering data, the costs are visible to finance, and the value is not.

The seniority signal is the opposite instinct: choose the smallest thing that works, and be able to say precisely at what volume it stops working.

The honest limitation

On call is real. Many data engineering jobs carry a rota, and being woken at 04:00 because a vendor's API changed is part of the compensation you are being paid. Ask about the rota, the frequency of pages in the last quarter, and whether there is time-off compensation. A team that cannot answer with numbers has not measured it, and unmeasured on call is usually bad on call.

BEFORE YOU MOVE ON

A nightly pipeline writes only 40% of the expected rows because a source API rate-limited it, but the job exits successfully and the dashboard updates. Why is this worse than the job crashing outright?

A. Because a crash would have triggered the retry logic, and retries always recover rate-limit failures.

B. Because incomplete data that looks complete is acted on, and the error is only discovered later when trust is already spent — whereas a crash is noticed immediately and blocks the bad numbers from being used.

C. Because partial writes corrupt the table's schema and require a full rebuild.

D. Because the pipeline used more compute than budgeted for a partial result.

14 · 10 min

The machine learning engineer

THE ONE THING TO KEEP

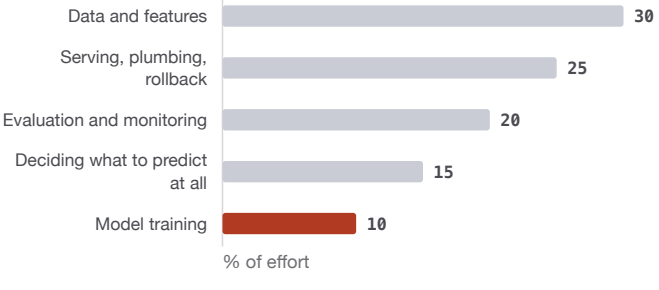
Most of a machine learning engineer's work is the system around the model, and the defining bug is a feature that exists in training data but not at prediction time.

The model is the small part

The title suggests a person who trains models. The job is mostly building and operating the system around a model: getting features to it reliably, serving predictions inside a latency budget, noticing when quality decays, and being able to roll back.

A useful proportion, borne out by most teams that have measured it: of the effort in a deployed machine learning system, model training is somewhere between five and fifteen per cent. The rest is data, plumbing, evaluation, monitoring, and the arguments about what should be predicted at all.

Where the effort goes in a deployed model



Training is somewhere between five and fifteen per cent of the work, and it is the part every course teaches. The rest is the system around the model, and that is the job.

The week

You are asked to reduce failed deliveries. Week one is not modelling; it is establishing what "failed" means in the data and discovering that the field was

only populated properly after last April. Week two you build a baseline — predict the customer's historical failure rate — and it is 80% as good as anything you will build later, which is worth knowing before you spend a month. Week three you add features and gain four points. Week four you find one of those features is not available at prediction time in production, only in the training table, and you delete it and lose the four points.

That last one has a name and it is the defining bug of the field.

Training-serving skew, and leakage

Leakage is when your training data contains information that will not exist when the prediction is actually made. The classic: predicting whether a customer will cancel, using a field that the operations team fills in *after* the cancellation call. Your offline accuracy is 0.94 and your live accuracy is 0.61, and everybody spends three weeks confused.

Training-serving skew is the same family: the feature is computed one way in the training pipeline and a slightly different way in the serving path — a different time zone, a different null handling, a different rounding. The model was trained on one distribution and sees another.

The structural defence is to compute features once, in one place, and use the same code in both paths. That is the entire reason feature stores exist. You do not need a feature store to get the benefit; you need one function.

The behavioural defence is a question you should ask of every feature: *"At the moment we make this prediction, does this value exist yet?"* Asking it about twenty features takes an hour and saves a month.

The stack

- **Python**, properly — not notebook Python. Modules, tests, typing, packaging.
- **scikit-learn** for anything tabular, which is most business problems. **XGBoost** or **LightGBM** for the rest; gradient-boosted trees still beat neural networks on ordinary tabular data and this surprises people every year.

- **PyTorch** where deep learning is genuinely needed. Free, and Google Colab and Kaggle both give free GPU hours, which is enough to learn on.
- **MLflow** (free, open source) or Weights & Biases (free tier) for tracking experiments, because in month two you will not remember which run produced the good number.
- **Docker**, a cloud, and enough CI to run tests on a pull request.

Two routes in

From software engineering. Learn the modelling. Faster, because the hard, unglamorous half — code quality, testing, deployment — is already there.

From analysis or research. Learn the engineering. This is the more common starting point and the slower path, and the gap that must be closed is not algorithms; it is writing code somebody else can run.

The failure that ends the first year

Optimising the metric instead of the decision. You improve AUC from 0.81 to 0.84 over six weeks, and the business outcome does not move, because the model's output feeds a process with a fixed capacity of 200 reviews a day and you never asked what happens to a prediction after you emit it.

Follow the prediction all the way to the human or system that acts on it, before you train anything. Ask what changes as a result. If nothing changes, you have found the real problem, and it is not a modelling problem.

The honest limitation

Fewer of these jobs exist than the internet implies, and most of them are at companies already at the top of the data maturity ladder. A great many advertised machine learning engineer roles are, on inspection, data engineering roles with one model in them. That is not a scam and it is often good work — but if you take one expecting to train models daily, you will be unhappy, and the interview question that reveals it is simply: *"How many models are in production today, and who retrains them?"*

BEFORE YOU MOVE ON

A churn model scores 0.94 AUC offline and 0.61 in production. The team confirms the code deployed is identical to the code trained. What should be checked first?

A. Whether the production traffic differs from the training population and the model needs retraining on recent data.
B. Whether the model is underfitting and a larger architecture would close the gap.
C. Whether the serving infrastructure is dropping requests or timing out under load.
D. Whether any feature used in training is unavailable or computed differently at prediction time — a value that only gets populated after the event being predicted would produce exactly this pattern.

15 · 10 min

The AI engineer

THE ONE THING TO KEEP

An AI engineer's core skill is separating retrieval failures from generation failures and trading quality, cost, latency and risk with actual numbers.

A role built on somebody else's model

The AI engineer builds products on top of models they did not train, usually reached through an API. Retrieval systems, assistants, document extraction, classification pipelines, agents that take actions. The job existed in a recognisable form from about 2023 and is now one of the most commonly advertised titles in the field.

It is closer to software engineering than to data science. If you can build and operate a web service and you understand evaluation, you are most of the way

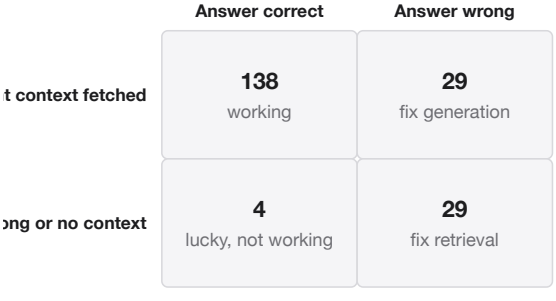
there.

The week

A support assistant answers customer questions from your documentation. Week one it demos beautifully. Week two, someone asks about a product you discontinued and it invents a refund policy. Week three you build a test set of 200 real questions with correct answers, and discover the assistant is right 71% of the time, which nobody knew because nobody had counted. Week four you find that half the failures are retrieval — the right document was never fetched — and the model was never the problem.

That split is the single most useful diagnostic in this job. **When the output is wrong, was the right context retrieved?** If no, fix retrieval: chunking, embeddings, the query itself. If yes, fix generation: the prompt, the model, the output format. Teams that do not separate these two spend months tuning prompts against a retrieval bug.

Two hundred real questions, sorted by where they failed



Seventy-one per cent right. Of the 58 failures, half sit in the bottom row: the right document was never fetched, and no amount of prompt tuning could have fixed them. The four lucky answers in the bottom-left are not to be trusted either.

The four constraints you are always trading

1. **Quality**, measured on a set you built, not on vibes.
2. **Cost**, measured per request in currency. A 40,000-token prompt on every call is a business decision, not a technical detail.
3. **Latency**, measured at the 95th percentile, not the average. Users experience the slow tail.

4. **Risk** — what happens when it is wrong, and what the system does to bound that.

Almost every design argument in this role is one of these traded against another. A smaller model is cheaper and faster and worse. More retrieved context is better and slower and dearer. Being able to say "this change costs 1.9 cents per request and buys nine points of accuracy" is what makes somebody senior in this work.

Non-determinism changes how you build

The same input can produce different output. That breaks the habits of ordinary software testing and it is why evaluation, not debugging, is the core discipline. You do not step through a failure; you measure a rate over a set of cases, change something, and measure again.

Practical consequences: pin model versions, because a provider's silent update will move your numbers; log the full prompt and response for every request so a failure can be reproduced; and keep a regression set that every change must pass before it ships.

The stack

- **Python** or **TypeScript**, and ordinary backend engineering — queues, caching, retries, rate limits, timeouts.
- **A model API** — the commercial providers, or open-weight models. Free paths that genuinely work: Ollama or llama.cpp running a small open-weight model on a laptop, and Hugging Face for models and datasets.
- **A vector store** — pgvector inside PostgreSQL is free, ordinary, and sufficient for most systems; the dedicated services matter at scale.
- **An evaluation harness.** You can and should write your own before adopting a framework; the free ones worth knowing are promptfoo, Ragas and DeepEval.
- **Observability.** Log every call with its cost and latency, or you will not be able to answer the first question finance asks.

Two routes in

From backend engineering. Short. Add retrieval, evaluation and cost awareness to what you already do.

From analysis, support, teaching or a domain job. Longer but real, and this route has one genuine advantage: you know what a good answer looks like in the domain, which is the scarce half. Build a small assistant over documents you actually understand, then measure it honestly and publish the number, including the failures.

The failure that ends the first year

Shipping on demos. A system that impresses in a meeting and has never been measured is not a product; it is a rehearsal. The person who says "we are at 71% on 200 real questions and here are the four failure categories" is trusted immediately, even when the number is bad. Especially when the number is bad.

The honest limitation

This is the least settled role in the field. The tooling changes every few months, half the frameworks that were essential two years ago are abandoned, and job titles are still inconsistent. What transfers is not the framework — it is retrieval, evaluation, cost control and system design. Invest there and the churn stops mattering.

BEFORE YOU MOVE ON

A retrieval-based assistant gives a wrong answer about a discontinued product. The team spends two weeks rewriting the system prompt with no improvement. What did they most likely skip?

A. Checking whether the correct document was retrieved at all before assuming the model's generation was at fault.

B. Upgrading to a larger model, which would have handled the ambiguity better.

C. Lowering the temperature so the output is deterministic and reproducible.

D. Adding a disclaimer so users know answers may be inaccurate.

16 · 9 min

The platform and MLOps engineer

THE ONE THING TO KEEP

Platform and MLOps work is judged by other teams' delivery speed and by adoption, so build one well-supported road with a real team before generalising it.

Measured on other people's speed

A platform engineer builds the road other teams drive on. Nobody outside your team can name what you shipped, and the honest measure of your work is a question about somebody else: how long does it take a data scientist here to go from an idea to a model serving live traffic?

If the answer is eleven weeks, you have work. If it is two days, you have done the job.

The week

Somebody's training run has been queued for four hours and they want to know why. A model is serving 300 milliseconds slower since Tuesday and nobody changed the model, so it is the infrastructure. Finance has sent the cloud bill with one line highlighted in yellow. A new starter needs access to five systems and the process for that is a document from 2023 that references a team which no longer exists. In between, you are building the thing that makes the next twenty deployments boring.

The idea that carries the role: paved roads

The temptation in platform work is to build a system so general it handles every possible case. The better instinct is a **paved road**: one well-supported way to do the common thing, documented, tested and fast, with an explicit exit for the rare team that genuinely needs something else.

A paved road for model deployment might be: put your model behind this interface, commit it here, and CI builds the container, runs the evaluation set, deploys behind a feature flag at 5% traffic, and rolls back automatically if error rate or latency crosses a threshold. Three teams using that beats twelve teams each inventing their own, even when each of their versions is slightly better suited to them.

The corollary is uncomfortable and important: **a platform nobody adopts is worse than no platform**, because it costs maintenance and splits the organisation. Adoption is the metric. If people are routing around you, they are telling you something true.

What MLOps adds beyond ordinary DevOps

Three things, and they are the reason the term exists at all.

- **Models decay without anybody changing the code.** The world moves and yesterday's model is quietly worse. Nothing in ordinary software deployment prepares you for an artefact that rots on its own, so you need monitoring of input distributions and output quality, not just uptime.

- **You must be able to answer "which data trained the thing currently serving?"** Reproducibility spans code, data and configuration, which means versioning the data too.
- **Rollback includes the model, not only the code.** A rollback that restores the previous container but keeps the new model is not a rollback.

The stack

- **Linux, networking and enough security to not be dangerous.** This half is unglamorous and does most of the work.
- **Containers and orchestration** — Docker, Kubernetes. Free to learn locally with kind, k3s or minikube on an ordinary laptop.
- **Infrastructure as code** — Terraform or OpenTofu, both free; Pulumi if the shop prefers code.
- **CI/CD** — GitHub Actions and GitLab CI both have free tiers generous enough to run a real pipeline.
- **Observability** — Prometheus and Grafana, free and open source, are what the paid products are usually wrapping.
- **Model-specific pieces** — MLflow or similar for a model registry, KServe, BentoML or plain FastAPI for serving.

You can build a genuinely credible portfolio here on one machine: a model served in a container, deployed by a pipeline on every commit, with metrics on a Grafana dashboard and an automatic rollback rule. It costs nothing and demonstrates the entire role.

Two routes in

From DevOps, systems administration or backend. Add the model lifecycle. This is the shortest path into a well-paid seat that exists in this field, and it is under-advertised because people who could take it do not think of themselves as AI people.

From data science. Learn the systems half properly. Slower, but the resulting engineer is unusually valuable because they understand what the scientists actually need.

The failure that ends the first year

Building the platform in isolation and unveiling it. Six months of work meets three teams who each say "that does not fit how we work" — and they are right, because nobody asked them.

The alternative is to take one team's real deployment, do it with them by hand, and only then generalise the parts that repeated. Platform work done as a service to a specific team first, and generalised second, almost always survives; the reverse almost never does.

The honest limitation

Your best days are invisible. When the platform works, nobody mentions it, and in performance reviews this is a genuine problem you must actively manage by measuring and reporting the things you removed: deployment time down from eleven days to two, cloud spend down 30%, four fewer incidents this quarter. If you do not count it, nobody will.

BEFORE YOU MOVE ON

A platform team spent six months building a general model-deployment system. Three product teams have each declined to use it and continued with their own scripts. What is the most productive response?

A. Mandate the platform through an engineering policy so the organisation stops fragmenting.

B. Take one team's real deployment, do it end to end with them using the platform, fix whatever breaks, and generalise only the parts that then repeat across a second team.

C. Add the missing features each team asked for, so the platform covers all three cases.

D. Publish better documentation and run an internal training session.

17 · 9 min

The applied scientist and research engineer

THE ONE THING TO KEEP

Applied research is run on time-boxed questions against a properly tuned baseline, and its most valuable habit is writing up what did not work.

Between a paper and a product

An applied scientist works on problems where the method is not yet known. Not "build a classifier" — that is engineering — but "we do not currently know how to rank these results, and the standard approaches do not work here."

The role concentrates in a small number of places: frontier labs, big technology research groups, quantitative finance, pharmaceutical and biotech companies, some large retailers and logistics firms with genuinely hard optimisation, and universities. It is the smallest of the roles in this module by a wide margin, and the most credential-sensitive.

The week

Mostly reading and failing. You read four papers, decide two are irrelevant, reimplement one and cannot reproduce its reported number. You build a small experiment that isolates a single question. It gives a negative result. You write down the negative result properly, because in three months somebody will propose the same idea and your note will save them a fortnight.

If you dislike the sentence "it did not work and here is why", this is not your role.

The discipline that separates good from bad

Time-boxing. Research can absorb infinite time, so the working version of the job runs on explicit budgets: "two weeks on this direction; if we cannot

beat the baseline by three points by the 14th, we stop and write it up." Teams without this drift for quarters.

A baseline first, always. Before any novel method, implement the simplest thing — a heuristic, a linear model, the current production rule — and measure it. A surprising share of research projects end when the baseline turns out to be within one point of the ambitious idea. Discovering that in week one is a success, not a disappointment.

Reading papers as a skill, not a chore. The efficient order is abstract, then figures and tables, then the experimental setup, then the limitations section, and only then the method. Most papers are eliminated at the tables. When a paper claims a large gain, the first question is what it was compared against and whether that baseline was tuned as carefully as the proposed method. Frequently it was not, and that is where a large share of reported improvements live.

The stack

- **Mathematics** that is actually used: linear algebra, probability, optimisation, and enough statistics to know when a difference is noise.
- **PyTorch** and the surrounding ecosystem; **JAX** in some groups.
- **Hugging Face** for models and datasets, and the free compute on Kaggle and Colab, which is enough for real small-scale experiments.
- **Experiment tracking** and an obsessive habit of recording seeds, configurations and dates.
- **Writing.** The output of this job is a document. People underestimate how much of the role is prose.

Free path for the mathematics: MIT OpenCourseWare's linear algebra course, and Octave or NumPy instead of MATLAB, which does the same work at zero cost.

Two routes in

The credentialled route. A PhD, or a master's with publications. For frontier research positions this is close to a requirement, and pretending other-

wise wastes people's years.

The side route, which does exist. Research *engineer* positions — building the infrastructure, running the experiments, reproducing results — are open to strong engineers without doctorates, and they sit in the same rooms. Public, careful reproductions of papers are the currency here: take a recent paper, reimplement it, report where your numbers differ and why. That artefact is rare, checkable, and takes you seriously into conversations that a certificate never will.

The failure that ends the first year

Chasing novelty over the problem. It is possible to spend a year on an interesting method that improves a metric nobody in the company acts on. Applied means applied: the work has to end somewhere a decision changes, and knowing which of your directions can actually land is the difference between a scientist a company renews and one it does not.

The honest limitation

The market for this role is small, geographically concentrated, and unusually sensitive to funding cycles. Research teams are cut first when budgets tighten — this happened visibly across the industry in 2023 and again since — because their value arrives on a longer horizon than a quarter. That is not an argument against the role. It is an argument for keeping engineering skills sharp alongside it, so that a contracting research market does not become a contracting career.

BEFORE YOU MOVE ON

A team wants to try a new architecture for a ranking problem. What should happen in week one?

A. Reimplement the most recent paper on the topic to establish state of the art.

B. Implement the simplest available approach — the current production rule or a linear model — and measure it on the evaluation set, so any later result has something honest to be compared against.

C. Secure enough GPU budget for the full training runs so the project is not blocked later.

D. Build the evaluation set after the first model exists, so it can be designed around the failure cases that appear.

18 · 9 min

Trust, safety and the policy side of AI

THE ONE THING TO KEEP

Trust and safety is a chain from policy to taxonomy to detection to appeals, and every threshold is an explicit choice about which of two harms you would rather cause.

The job that decides what a system is allowed to do

Every deployed model is governed by rules: what it must refuse, what gets removed, what gets shown to fewer people, what gets escalated to a human. Somebody writes those rules, somebody turns them into a taxonomy a classifier can be trained on, somebody measures whether enforcement matches the policy, and somebody handles the appeals from people who were wrongly caught.

That cluster of work is trust and safety. It employs far more people than research does, it is growing because regulation is growing, and it hires unusually

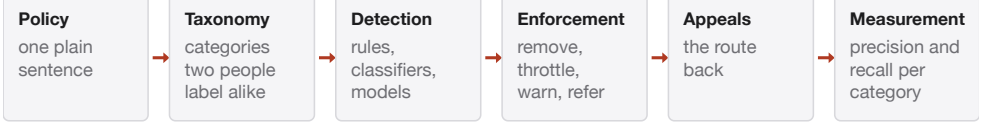
widely — lawyers, linguists, teachers, journalists, former moderators, and engineers.

The chain, in order

1. **Policy.** A sentence in ordinary language: "we remove content that harasses a named private individual."
2. **Taxonomy.** That sentence broken into categories with definitions and boundary cases, so two people reading it label the same post identically.
3. **Detection.** Rules, classifiers, models — whatever finds candidates.
4. **Enforcement.** What happens: removal, throttling, a warning, an account action, a referral.
5. **Appeals.** The route back for the people you got wrong.
6. **Measurement.** Precision and recall per category, and how both differ across languages and groups.

Almost every failure in this field is a break between two adjacent links. A good policy with no taxonomy produces inconsistent enforcement. Good detection with no appeals route produces a slow accumulation of injustice that surfaces as a news story.

The chain, in order



Almost every failure in this field is a break between two adjacent links: a policy with no taxonomy enforces inconsistently, and detection with no appeals route accumulates injustice until it becomes a news story.

The trade that never goes away

Every threshold sets a ratio of two harms. Raise recall and you catch more genuine abuse and also remove more innocent posts. Raise precision and you leave more abuse standing.

There is no setting that avoids both, and the mathematics will not choose for you. What makes somebody good at this job is being explicit: which of the two

errors is worse *for this category*, and saying so in writing.

The answer differs sharply by category, and the reasoning is the interesting part. For child sexual abuse material and for direct threats, a false negative is catastrophic and irreversible, so you accept a heavy review load. For political opinion, a false positive silences lawful speech, and the standard response is not removal but reduced distribution — a decision that is genuinely contested and that you should be able to argue on both sides. For self-harm, removal is often the wrong action entirely: the post is a person in distress, and the correct response is support resources and keeping them connected, not deletion.

The stack

- **Writing.** The primary tool. Policies, definitions, incident reports, transparency reports.
- **SQL and basic analysis**, because every argument here is settled with rates: appeals upheld per thousand actions, prevalence by category, enforcement rate by language.
- **The evaluation vocabulary** from the evidence module: confusion matrix, precision, recall, inter-annotator agreement, sampling for prevalence.
- **Regulatory literacy** — the EU's Digital Services Act, the EU AI Act, India's IT Rules, the UK Online Safety Act, and the fact that the rules differ per country and are still moving. You do not need to be a lawyer. You need to know which questions belong to one.

Two routes in

From moderation or support. Very common, and the domain knowledge is real. Move from applying policy to writing it by documenting edge cases and enforcement inconsistencies that nobody else has written down.

From data or engineering. Bring measurement. Teams frequently cannot answer "what is our precision on harassment in Hindi", and the person who can build that answer becomes central quickly.

The portfolio piece: take a public platform's published policy, build a taxonomy from it, label 200 real public posts against it, report per-category precision

and recall for a simple keyword baseline, and write up the boundary cases that broke your definitions. That is the job, done small.

The failure that ends the first year, and the honest limitation

The failure is treating enforcement as a technical problem. A classifier at 0.9 precision applied to twenty million posts a day is wrong about two million of them, and each one is a person. The work is designing the whole system — thresholds, human review, appeals, and what happens at 3am on a public holiday.

The limitation is exposure. Depending on the role you may see material that is genuinely distressing. Reputable teams limit exposure time, blur by default, rotate people out and provide real clinical support. Ask about all four in the interview. An employer that treats the question as unusual has answered it.

BEFORE YOU MOVE ON

A team raises the confidence threshold on its harassment classifier so that fewer innocent posts are removed. What has necessarily happened, and what should be reported alongside the change?

A. Overall accuracy has improved, and reporting the new accuracy figure is sufficient.

B. More genuine harassment now stays up, so the change should be reported as a pair — the fall in wrongful removals and the rise in missed abuse — with a stated view on which error is worse for this category.

C. The classifier has become more precise, so it should be retrained less often.

D. Appeals will rise, because more users will contest the remaining removals.

19 · 9 min

The AI product manager

THE ONE THING TO KEEP

An AI product manager's job is to state what decision changes, what being wrong costs, and what the fallback is, then insist on a measured number instead of an impression.

Deciding what should be built, and what should not

A product manager owns the problem, not the code. In AI products the job acquires a specific difficulty: the thing you are shipping is probabilistic, so "does it work" has no yes-or-no answer, and every feature has a failure rate you must decide is acceptable.

This changes the craft. An ordinary feature either logs the user in or does not. An AI feature summarises the document correctly 88% of the time, and your job is to decide whether 88% is shippable, what the other 12% costs, and what the product does to make being wrong survivable.

The week

You write a one-page problem statement and get three teams to agree it is the right problem. You sit with six support agents and watch them work, and discover the thing they actually hate is not the thing anybody proposed to automate. You spend a morning labelling 100 model outputs yourself, because you will otherwise be arguing about quality with no shared referent. You cut a feature that demos well because its failure mode is a customer receiving a wrong price, and no confidence score makes that acceptable.

The three questions that define the role

1. What decision changes? If the model's output does not alter what somebody does, the feature is decoration however impressive it looks.

2. What is the cost of being wrong, and who pays it? Wrong summary of an internal meeting: annoying. Wrong extraction of a dosage from a prescription: dangerous. Wrong refund decision: expensive and unfair. The design follows from this answer — how much human review, how the interface presents uncertainty, whether the system acts or only suggests.

3. What is the fallback? Every AI feature needs an answer to "what happens when it fails" that is not "the user notices". Show sources so the answer is checkable. Offer the manual route beside the automatic one. Fail into a human queue rather than into silence.

A specification that answers those three is more valuable than one containing model choices, and model choices are not your decision anyway.

The measurement half

You are the person who insists the number exists. Not "it feels better since we changed the prompt", but 200 real cases, labelled, with a number and a date. In practice you will often build the first evaluation set yourself, in a spreadsheet, because nobody else will and the argument cannot be settled without it.

You also own the second number that engineers rarely raise: cost per interaction. A feature at 4 cents a request used ten times a day by 50,000 users is $730,000 a year. That is a product decision wearing a technical costume, and it is yours.

The stack

- **Writing**, above everything. The core artefact is a document.
- **SQL**, enough to answer your own questions without queuing behind an analyst.
- **A spreadsheet**, for the evaluation set and the unit economics.
- **Enough technical literacy** to know what is hard: that latency and quality trade against each other, what retrieval is, why a model cannot cite a document it never saw, and why "just fine-tune it" is not a small ask.
- Free paths for prototyping: Streamlit or Gradio let a non-engineer put a working demo in front of a user in an afternoon, and both are free.

Two routes in

From a domain job. Support lead, operations manager, clinician, teacher, underwriter. You know the workflow being changed, which is the half that cannot be learned quickly. Add the measurement discipline.

From analysis or engineering. You have the numbers; you learn the customer research and the writing, and unlearn the instinct to answer "how" before "whether".

The failure that ends the first year

Writing "make it smarter" as a requirement. It is not actionable, it cannot be tested, and it hands the hardest decision — what good means — to whoever writes the code, who has less information than you do.

The replacement is a sentence with a number and a consequence in it: *"On the 300-question set, at least 90% of answers must cite a real source document, because the reviewer's whole job is checking the source and an uncited answer costs them more time than doing it manually."* Now an engineer can build, and disagree, on the same terms.

The honest limitation

Product management is difficult to enter from outside, because the role is largely judgement and judgement is hard to evidence in a portfolio. The most reliable route in is sideways, inside a company where people have already seen you make decisions. Plan for that rather than for a cold application, and be sceptical of anyone selling a certificate as the entry.

BEFORE YOU MOVE ON

A team proposes an assistant that auto-approves customer refunds under a threshold, at 93% accuracy on a labelled set. What should the product manager establish before agreeing to ship?

A. Whether accuracy can be raised to 97% first, since higher is always safer.

B. Which model the team plans to use and whether a larger one would be better.

C. What the 7% of errors actually look like — wrongly approved versus wrongly refused, who bears each cost, and what the fallback and appeal route is for a customer the system gets wrong.

D. How much engineering time the feature will take, so it can be prioritised against other work.

20 · 9 min

The forward-deployed and solutions engineer

THE ONE THING TO KEEP

Forward-deployed work makes a product work inside one organisation, and its career value is the loop back into the product — which fails the moment each customer gets a permanent fork.

Engineering, in the customer's building

Some companies sell software that does not work on arrival. Not because it is broken, but because it must be connected to a customer's data, configured to their process and argued about with their staff before it produces anything. The person who does that is called a forward-deployed engineer, a solutions engineer, an implementation engineer, or a field engineer, depending on the company.

This is one of the most open doors in the field for somebody without a conventional background, and one of the least discussed, because it does not sound

like an AI job. It is an AI job. You are the person who makes the model produce value inside a specific organisation, which is the part nobody else has solved.

The week

You are at a logistics company. Their data lives in three systems and one of them is a mainframe report emailed as a fixed-width text file at 06:00. You write the parser. You discover the depot codes in system A do not match those in system B, and nobody at the company knew, because the humans mentally translated them. You build the mapping. You sit with a dispatcher for two hours and learn that the model's recommendation is ignored every Friday, for a reason that is completely correct and appears nowhere in the specification. You go back to the product team with that.

Half the job is engineering. Half is anthropology.

What makes it different from consulting

A consultancy sells your hours. A forward-deployed engineer works for the product company and the hours are a cost, not the revenue. The consequence is important: your incentive is to make the thing work with less bespoke code each time, and to push what you learned back into the product so the next customer needs less of you.

That feedback loop is the career value. You see, at close range, why software fails in real organisations — and that is the knowledge product managers and platform teams pay for. A great many product managers and senior engineers in enterprise software arrived through this seat.

The skills

- **Enough engineering to be dangerous alone.** Python, SQL, APIs, some data plumbing, enough to build without waiting for anybody.
- **Reading somebody else's systems fast.** Legacy databases, undocumented exports, an API written in 2011.

- **Communication under pressure**, in a room where you are outnumbered and something is not working.
- **Written follow-up.** After every session: what we agreed, what is blocked, who owes what by when. This single habit makes people trust you disproportionately.
- **Restraint.** Not every request should be built. Learning to say "we can do that, and here is what it costs and what it delays" is the senior version of the job.

Two routes in

From support or implementation. If you are already the person customers ask for by name, you have the hard half. Add the engineering.

From engineering. If you can build and you do not mind rooms, this pays better than an equivalent internal role at many companies and moves your career faster, because you are exposed to decision-makers early.

The evidence that works in the interview is not a portfolio project. It is a story, told precisely: a system somebody else built, a customer or user with a problem, what you found when you looked, what you changed, and what happened afterwards. Two of those, told in four minutes each, are worth more here than a repository.

The failure that ends the first year

Becoming a bespoke development team. Each customer asks for something reasonable, you build it, and eighteen months later there are eleven forks of the product and nobody can upgrade anybody. The discipline is to keep asking which of these requests is the same request wearing different clothes, and to push the general version into the product.

The honest limitation

Travel, and the emotional load of being the face of a product when it disappoints somebody. There are weeks where you are apologising for a decision you did not make. Companies vary enormously in how much they protect field

staff from this, and it is worth asking directly: who talks to the customer when a deadline slips, and does the product team ever attend?

There is also a real risk of skill drift. Two years of parsing other people's exports is not two years of building product, and if you want to move into core engineering later, keep one deep technical thing going deliberately rather than assuming the job will supply it.

BEFORE YOU MOVE ON

A forward-deployed engineer has built custom parsers and workflow tweaks for six customers. Upgrades now break at least two deployments every release. What is the underlying mistake?

A. The release process needs better regression testing across all six deployments.

B. Customers should be charged for the customisation so the work pays for itself.

C. Each customer's reasonable request was built as bespoke code instead of being examined for the general version that belongs in the product, so the deployments have diverged into six forks.

D. The product is not mature enough to be sold yet and deployments should pause.

CASE STUDY · MODULE 2

The model that scored 0.94 in the notebook

A machine learning engineer's first real project at a four-hundred-person logistics company in Coimbatore, three weeks before a board demonstration.

The brief was to reduce failed deliveries. Around eleven per cent of consignments needed a second attempt, each one costing about ₹180 in fuel and driver time, and the operations director wanted to know which deliveries would fail before the van left the depot.

Week one produced no model at all. It produced the discovery that the `delivery_status` field had only been populated consistently since the previous April, and that before that the depots had used a free-text note. Week two produced a baseline: predict each customer's historical failure rate. It got about seventy-nine per cent of the ranking right and took an afternoon.

Week three produced the model everybody wanted. Gradient-boosted trees over thirty-one features, and an offline score of 0.94.

Nobody on the team had asked for a neural network and nobody needed one. This is ordinary tabular business data — a few hundred thousand rows, thirty columns, no images and no text — and gradient-boosted trees remain the correct tool for it, which surprises people who have spent a year reading about model architectures.

The question asked at the feature review

A senior engineer went through the feature list and asked one question about each: *at the moment we make this prediction, does this value exist yet?*

Thirty features survived. One did not. `resolution_code` was filled in by the depot clerk **after** a delivery attempt, describing how the attempt had been resolved. In the training table it sat innocently beside the target. At prediction time — 05:30, before any van has moved — it is empty.

Removing it took the score from 0.94 to 0.81. Three weeks of work, and the number went backwards in front of a board demonstration that was already in

the calendar.

The decision

Demonstrating 0.94 was available. The leaked column would not be visible in the slide, the number was real in the sense that the notebook had computed it, and there was a defensible version of the argument: the demo is about the idea, not the deployment.

The cost of that path was not abstract. The operations director would take the number to the board, the board would fund a rollout, and in production the model would score somewhere near 0.61 — at which point the failure would be discovered by the depot supervisors, who are the people whose trust the system needs and who have watched three previous head-office systems arrive and be quietly abandoned.

The cost of the honest path was three weeks of visible progress turning into a lower number, in a company where nobody on the board understood what leakage was and where "it went down" is not an easy sentence to say in a boardroom.

The second problem, which was larger

While arguing about the first, somebody asked what happened to a prediction after it was emitted. The answer: the depot could physically re-check about a hundred and twenty consignments a day out of roughly four thousand. Whatever the model produced, exactly a hundred and twenty deliveries would be looked at.

Which meant the metric everybody had been optimising — overall ranking quality across four thousand rows — was not the metric that mattered. What mattered was how many genuine failures appeared in the top hundred and twenty.

That is the second failure that ends a first year in this role, and it is quieter than leakage: improving a score for six weeks while the business outcome does not move, because nobody followed the prediction to the person who acts on it. A model whose output lands in a queue with a fixed capacity is not being

asked to rank four thousand things. It is being asked to fill a hundred and twenty slots well.

What the three weeks were actually for

The engineer's own summary afterwards was that roughly one week of the three had been modelling. The rest was establishing what the target field meant before April, arguing about whether a re-attempt on the same day counted as a failure, writing the feature computation so that the training path and the serving path used the same function, and finding out what happened to a prediction after it was emitted.

That proportion is not a sign of a badly run project. It is the job. Model training is somewhere between five and fifteen per cent of the effort in a deployed system, and a candidate who describes their week the way the internet describes this role has usually not deployed anything.

STOP HERE

Before turning the page: what would you have done, and what would it have cost if you were wrong? Write it down. The point of the next section is to compare.

WHAT HAPPENED

The feature was deleted. The demonstration showed 0.81 with a slide headed "the number we removed and why", explaining the leak in four sentences, and a second slide showing that against the historical-rate baseline of 0.79 the model was worth two points, not fifteen. The operations director's response was not disappointment; it was to ask what the model would do to the hundred and twenty daily checks. That reframed the project. The team switched to precision in the top hundred and twenty, which the baseline achieved at thirty-one per cent and the model at forty-four — a difference worth roughly ₹9 lakh a year in avoided second attempts, and a number the board could act on. Six months after rollout, live precision was within two points of the offline figure, which is the only result that ever mattered.

Why is leakage described as the defining bug of machine learning engineering rather than as a beginner's mistake?

Because it does not announce itself. Nothing errors, the code is correct, and the model appears to be excellent. It is caught only by a habit — asking of every feature whether the value exists at the moment the prediction is made — and it is created by the ordinary structure of business data, where operational systems record what happened after the event you are trying to predict.

The baseline scored 0.79 and the final model 0.81. Was the three weeks wasted?

No, but the honest headline is two points above the baseline rather than a model that works. The value of the three weeks was mostly the discovery of the leak, the fixed capacity constraint and the correct metric. Building the baseline first is what made all three visible; a team that had gone straight to the model would have shipped 0.94 and found out in production.

What does the hundred-and-twenty-checks constraint tell you about how to start a modelling project?

Follow the prediction all the way to the person or system that acts on it before training anything, and ask what changes as a result. Here the action was a fixed-capacity review queue, which means the correct metric was precision at a fixed cut-off rather than overall ranking quality. If nothing had changed as a result of the prediction, the real problem would not have been a modelling problem at all.

MODULE 2 · TEST

Check what stuck

6 questions, one right answer each. This one is for you, not for a record: answers and the reason behind each are at the back. If two questions from the same module go wrong, re-read that module before the exam rather than after.

1 An analytics engineer rebuilds a table from source every night instead of appending yesterday's new rows. Why?

A. Rebuilding is faster than appending on a columnar warehouse.
B. Appending is not supported by most transformation tools.
C. It keeps the warehouse bill predictable from one month to the next.
D. So the result is identical however many times the job runs.

2 A support assistant gives a customer a wrong refund window. What is the first thing an AI engineer establishes?

A. Whether the correct document was retrieved at all
B. Whether the prompt should be rewritten to be firmer
C. Whether a larger model would have answered correctly
D. Whether the temperature setting is too high for factual answers

3 Why does a data engineer prefer a pipeline that crashes to one that completes with half the rows?

A. A crash produces a stack trace, which is easier to debug than missing data.
B. A crash triggers the orchestration tool's retry, and partial writes do not.
C. A crash is visible; a partial write is believed for weeks and costs trust.
D. Partial writes corrupt the warehouse's statistics and slow later queries down.

4 A model scores 0.94 offline and 0.61 in production. Which question, asked of every feature beforehand, would most likely have caught it?

A. Is this feature correlated with any other feature in the set?

B. At the moment we make the prediction, does this value exist yet?

C. Is this feature available in the warehouse as well as in the source system?

D. Does this feature have an acceptable proportion of missing values in training?

5 A platform team ships a deployment system that is more capable than the three hand-rolled ones it replaces. Two teams keep routing around it. What follows?

A. The teams should be required to migrate, since the platform is measurably better.

B. A migration guide will fix it, because the objection is usually unfamiliarity.

C. The capability gap will close as the platform matures and adoption follows.

D. Adoption is the metric, and an unadopted platform costs more than none.

6 In trust and safety, why is a lawful political opinion throttled rather than removed, while a direct threat is removed?

A. Political content is harder for classifiers to detect accurately than direct threats are, in every language.

B. Removing political content attracts regulatory attention in more jurisdictions.

C. For each category, one of the two errors is worse, and here they point opposite ways.

D. Throttling is cheaper to operate at scale than removal and appeals together.

MODULE 3

The evidence: what gets tested, and how to prove you have it

Four skills carry almost every entry-level offer in this field — SQL, working Python, version control and evaluation — and one artefact converts them into a hire: a written project that shows judgement under mess. This module builds both, and covers the spreadsheet fluency no syllabus lists and every organisation runs on.

> BY THE END OF THIS MODULE YOU CAN
>
> Produce one written case study of your own work — with the mess you found, the decisions you rejected, and a number from a test set you built by hand — that a reviewer can judge in ninety seconds.

---

21 · 7 min

When the courses end and you still freeze

> THE ONE THING TO KEEP
>
> You learn to build by finishing small things alone, not by watching bigger things being finished.

Why this happens to almost everyone

A tutorial removes the two hardest parts of building software and leaves the easy part.

The hard parts are **deciding what to do next** and **getting unstuck from an error nobody has written about**. A tutorial has already decided everything, and its errors are anticipated. What is left is typing, and typing is the part you were never missing.

So the feeling of "I understood all of it and I can build none of it" is not a sign that you are slow. It is the accurate result of practising one skill and needing a different one.

The drill that fixes it

Take a project you already completed with a tutorial. Delete the code. Keep only a short description of what it did.

Rebuild it from an empty file. No video, no walkthrough. Documentation is allowed. Web search for specific errors is allowed.

```
Session 1 (90 min): get anything to run. One route, one
                    hardcoded value, printed to the screen.
Session 2 (90 min): make it real. Actual data, actual output.
Session 3 (90 min): make it survive bad input. Then stop.
```

It will be far harder than the tutorial and that difficulty is the entire point — you are finally practising the missing skill. Most people report the second rebuild feels different from the first in a way no additional course produced.

Shrink until it fits in your hands

Beginners choose projects five times too large, then interpret the collapse as lack of talent. A good first independent project is one you could finish over a weekend if nothing went wrong — which means it will take two weekends, because things go wrong.

Not "a social media app". A page that takes a CSV of your expenses and shows the three largest categories this month. Finish that. Then add one thing.

Learn to read errors

When something breaks, most beginners feel a rush of dread and scroll away from the message. The message is the lesson. Practise a fixed sequence:

1. Read the last line first. It names what failed.
2. Find the first line in the trace that points at *your* file, not the library's.
3. Say out loud what you believed was in that variable.
4. Print it. You are almost always wrong about it.
5. Only now search — with the exact error text, not a paraphrase.

Step 3 is where the learning is. Debugging is the repeated discovery that your mental picture of the program and the program itself were different.

Using an AI assistant without stalling

An AI coding assistant makes a capable programmer much faster and can freeze a beginner in place, because it removes exactly the struggle you need. Two habits keep it useful:

- **Predict before you accept.** Say what the code will do before you run it. When you are wrong, you have found a real gap.
- **Let it explain, not decide.** "Why does this fail?" builds you. "Write the whole thing" does not. Reading working code feels like learning and is not; producing code under uncertainty is.

A fair test: could you write today's code again tomorrow without help? If no, you got output, not skill.

Expect this to feel bad for a while

The gap between following and creating takes weeks of unpleasant, unimpressive work to cross. Nobody posts about that stretch, which is why it feels like a personal failing rather than the standard experience it is.

BEFORE YOU MOVE ON

Ade has finished three courses. He can follow any tutorial. Facing an empty file, he cannot start. Which change is most likely to move him?

A. Take a project-based course so the projects are larger and closer to real work.

B. Delete a project he already built with a tutorial and write it again from an empty file, using only a description of what it did.

C. Read the source of a well-known open source project to see how professionals structure code.

D. Have an AI assistant generate a complete app, then study the result line by line.

22 · 9 min

SQL, and the six query shapes that cover the job

THE ONE THING TO KEEP

SQL is tested in almost every data interview because it is fast to test and impossible to fake, and six query shapes cover nearly all of the work.

Why this one skill decides so many applications

SQL is the shared language of analytics, data engineering and a large part of AI application work. It is also the easiest thing in the field to test: an interviewer can put a schema on a screen and know within fifteen minutes whether you have used it. That combination — universally needed, cheaply verified — is why it appears in almost every process and why it is the highest-return eight weeks a beginner can spend.

The six shapes

Nearly every query you will write at work is one of these, or two of them stacked.

1. Filter and aggregate.

```
select date_trunc('month', created_at) as month,
       count(*)                        as orders,
       count(distinct customer_id)     as customers,
       sum(total_amount)               as revenue
from orders
where status <> 'cancelled'
group by 1
order by 1;
```

2. Join two tables and keep the rows that do not match. `left join` exists so that a customer with no orders still appears with a zero.

3. Group, then filter the groups. `having count(*) > 3` — filtering after aggregation, which is a different operation from `where`.

4. Bucket by time. Month, week, day, hour. Half of all business questions are a time bucket in disguise.

5. A window function. Running totals, rank within a group, comparing a row to the previous one.

```
select customer_id, order_date, total_amount,
       sum(total_amount) over (partition by customer_id
                               order by order_date) as running_-
total
from orders;
```

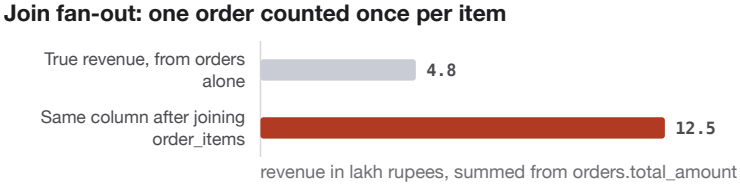
6. Second-highest, latest-per-group, first-touch. The classic interview family, solved with a window function or a correlated subquery.

If you can write all six against a schema you have not seen before, you can pass most SQL interviews.

The two bugs that cost real money

Join fan-out. You join `orders` to `order_items` to filter by product, then sum `orders.total_amount`. An order with three items now appears three times, and revenue is inflated by however many items the average order has. The query runs, the number looks plausible, and the dashboard is wrong for eight months.

The mechanism: a join multiplies rows whenever the right side has more than one match. Aggregating a left-table column after such a join counts it once per match.



A thousand orders averaging 2.6 items each. Join orders to order_items to filter by product, sum orders.total_amount, and every order is counted once per matching item. The query runs, the number looks plausible, and the dashboard is wrong for eight months.

The habit that prevents it: check row counts before and after every join. If the count changed and you did not expect it to, stop.

NULL is not a value. `where status <> 'cancelled'` silently drops every row where `status` is NULL, because comparing anything to NULL yields *unknown*, and `where` keeps only *true*. This is not a quirk to memorise; it is three-valued logic, and it will remove customers, orders and patients from your results without any error.

Write `where status is distinct from 'cancelled'` or `where (status <> 'cancelled' or status is null)` when NULL should be kept — and always know which you meant.

Practising without a cloud account

Job adverts name Snowflake, BigQuery and Databricks. You do not need any of them, and their free tiers all end. The SQL is the same.

The free path:

- **DuckDB** — one binary, no server, and it queries CSV and Parquet files directly. `duckdb -c "select count(*) from 'sales.csv'"` works on a file you downloaded five seconds ago.
- **SQLite** — already on your machine, and enough for everything in this lesson except window functions on very old versions.
- **PostgreSQL** — one install, and the dialect closest to what most warehouses use.

Dialects differ mainly in date functions and string handling. Learn one properly; the translation is an afternoon.

The practice method that works

Do not grind two hundred puzzle problems. Puzzle sites over-represent exotic window functions and under-represent the actual job, which is joining four tables that disagree with each other.

Instead: take one real dataset with several tables. Write twenty questions a person would genuinely ask of it — *which day of the week has the most cancellations, which customers stopped buying after March, is the average order value going up or down for repeat customers*. Answer each one. Then check every answer a second way: a different query, a spot-check of five rows, or a total that must match something you already know.

The second check is the part that turns SQL knowledge into employability, because it is the habit an employer is actually buying.

Where SQL stops

It is genuinely bad at some things, and knowing where it stops is part of knowing it. Anything requiring row-by-row procedural logic, complex string parsing, or calling an external service belongs in Python. Very deep nested subqueries become unreadable long before they become slow. And a query that is correct can still be unusable if it scans a hundred million rows every time someone opens a dashboard — at which point the answer is a materialised table, not a cleverer query.

BEFORE YOU MOVE ON

An analyst joins `orders` to `order_items` so she can filter to a product category, then reports `sum(orders.total_amount)` by month. Revenue looks about 2.4 times higher than finance's figure. What has happened?

A. The `order_items` table contains cancelled items that should have been filtered out first.

B. Finance is using a different currency conversion date, so the two figures are measuring different things.

C. The join produced one row per order item, so each order's total is counted once for every item it contains, and summing a left-table column after the join multiplies revenue by the average number of items per order.

D. Window functions are needed here, and the `group by` is aggregating at the wrong level.

23 · 9 min

The employable half of Python

THE ONE THING TO KEEP

The paid part of Python is reading other people's code, handling files and failures, and writing a script that somebody else can run.

What the job actually uses

Most Python courses spend their weeks on classes, inheritance and algorithm puzzles. Most Python jobs in this field spend their days on: read some data, reshape it, call an API, write the result somewhere, and make sure it does not fall over at 03:00.

The employable list is short.

- `pathlib` for files, because string paths break across operating systems.

- `csv` and `json` from the standard library, before reaching for anything bigger.
- `requests` or `httpx` for APIs, with a timeout on every single call.
- `pandas` or `polars` for tabular work.
- `argparse` so your script takes inputs instead of hard-coding them.
- `logging` instead of `print`, because a scheduled job with `print` statements produces output nobody sees.
- `try` / `except` naming a specific exception, never a bare `except:`.
- Virtual environments.

The bug that quietly destroys data

This one is worth more than a chapter of syntax.

```
df = pd.read_csv("customers.csv")
df["customer_id"].head()
# 0    7.0
# 1    42.0
```

Pandas inferred the column as a float because one row was empty. `007` is now `7.0`. A postcode of `01234` is now `1234`. An account number too long for a float loses its last digits silently, with no warning of any kind.

The fix is one argument, and typing it is a professional habit:

```
df = pd.read_csv(
    "customers.csv",
    dtype={"customer_id": "string", "postcode": "string"},
    keep_default_na=False,
)
```

The mechanism: identifiers look like numbers but are not numbers. Nothing arithmetic is ever done to them, and every operation that treats them as numeric loses information. This class of bug has broken payroll runs and lost hospital records, and no marketing page about learning Python will mention it.

"It worked in my notebook"

Notebooks keep state. Cells run in whatever order you clicked them, variables persist after you delete the cell that made them, and the thing that makes your code work may no longer exist anywhere in the file.

The discipline: **restart the kernel and run all, before you believe any result.** Do it before you show anyone anything. Half of the "I cannot reproduce it" conversations in data teams are this.

A script skeleton worth memorising

```
import argparse, logging, sys
from pathlib import Path
import pandas as pd

logging.basicConfig(level=logging.INFO,
                    format="%(asctime)s %(levelname)s %
(message)s")

def clean(df: pd.DataFrame) -> pd.DataFrame:
    before = len(df)
    df = df.drop_duplicates(subset=["order_id"])
    logging.info("dropped %d duplicate orders", before -
len(df))
    return df

def main() -> int:
    p = argparse.ArgumentParser()
    p.add_argument("--input", type=Path, required=True)
    p.add_argument("--output", type=Path, required=True)
    args = p.parse_args()

    df = pd.read_csv(args.input, dtype={"order_id": "string"})
    logging.info("read %d rows from %s", len(df), args.input)
    clean(df).to_csv(args.output, index=False)
    return 0

if __name__ == "__main__":
    sys.exit(main())
```

Every line of that is doing employable work: it takes arguments, it says what it did, it names its types, it returns an exit code a scheduler can read. A reviewer looking at this knows you have run something in production, or at least understand what production needs.

Environments, without the folklore

```
python -m venv .venv
source .venv/bin/activate      # Windows: .venv\Scripts\activate
pip install -r requirements.txt
```

That is the whole thing. `uv` does the same faster if you prefer. The point is that your project declares its dependencies in a file, so a colleague — or you in six months — can run it.

The free path covers everything: plain `venv` and `pip`, VS Code or VSCodium, Colab for anything needing a GPU. The paid options people assume they need — commercial Anaconda licences, PyCharm Professional — buy convenience, not capability.

The most underrated skill: reading someone else's code

You will spend more time reading code than writing it, and almost nobody practises it deliberately. A method that works on any unfamiliar repository:

1. Find the entry point — the file with `if __name__ == "__main__"`, or the route definitions, or whatever the README says to run.
2. Follow exactly one path from input to output. One request, one record, one file.
3. Ignore everything else, including the parts that look important.
4. Change one small thing and run it. What breaks tells you what depends on what.

An hour of this on a real open-source project teaches more than a week of tutorials, and it is precisely what your first month at a job will be.

Where pandas stops

Pandas holds everything in memory and copies aggressively, so on an ordinary laptop it starts struggling somewhere around a few gigabytes — and the effective ceiling is lower than your RAM, because an operation may need several copies at once. That is the honest threshold. Above it, the answers are `polars`, DuckDB over Parquet files, or moving the work into a database. Knowing the number is more useful than the vague claim that pandas is slow.

BEFORE YOU MOVE ON

A nightly script reads a CSV of member records with `pd.read_csv` and writes them to a database. Months later, staff find that members whose ID begins with a zero cannot be looked up, and a few long account numbers end in incorrect digits. The script has never raised an error. What happened?

A. Pandas inferred the ID column as numeric, so leading zeros were dropped and very long values lost precision — a silent conversion that produces no error, and is prevented by declaring the column as a string on read.

B. The database column is too narrow, so values were truncated on insert.

C. The CSV was written with the wrong encoding, corrupting some characters.

D. Duplicate member rows were silently overwritten during the load.

24 · 10 min

The statistics that come up, and the ones that never do

THE ONE THING TO KEEP

Precision improves only with the square root of sample size, so most business tests were never large enough to detect the effect they were run to find.

A short list, learned properly

Interviewers rarely ask you to derive anything. They ask whether you will notice that a number is meaningless. Six ideas cover almost every case, and each has an arithmetic form you can do on paper.

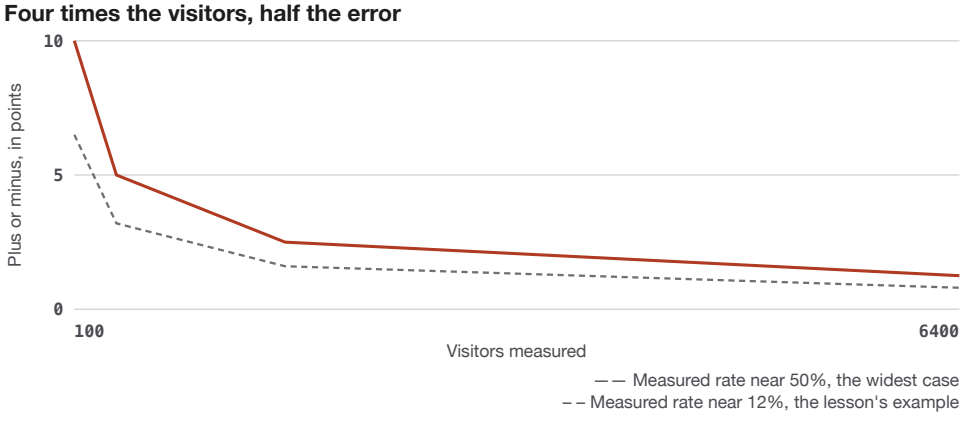
1. Uncertainty in a proportion

You measure a conversion rate of 12% on 400 visitors. How much could that move by chance alone?

```
standard error = sqrt( p * (1 − p) / n )
               = sqrt( 0.12 * 0.88 / 400 ) = 0.0162
95% interval  ≈ p ± 2 * SE = 12% ± 3.2%
```

So the true rate is somewhere around 8.8% to 15.2%. If someone reports that the rate "rose from 11% to 12%", nothing has been observed.

The consequence worth memorising: precision improves with the **square root** of sample size. Four times the data halves the error. That is why a survey of 100 people has a margin of roughly ±10 points and why "we asked 40 users" is almost never evidence of a small effect.



The 95 per cent interval around a measured proportion shrinks with the square root of sample size. From 400 to 1,600 visitors halves it; a survey of 100 people is roughly plus or minus ten points, which is why forty users is never evidence of a small effect.

2. Power, and why most tests cannot see what they are looking for

Before running a test, ask what size of difference it could detect. A serviceable rule for 80% power at the usual 5% threshold:

```
n per group ≈ 16 * p * (1 − p) / (difference)^2
```

Baseline conversion 5%, hoping for a 10% relative lift (5.0% → 5.5%, a difference of 0.005):

```
n ≈ 16 * 0.05 * 0.95 / 0.000025 ≈ 30,400 per group
```

Now the same baseline with a 1% relative lift (difference 0.0005): about **3 million per group**. Most companies do not have that traffic, which means most of their tests were never capable of detecting the effect they were run to find. Saying this politely, early, before the test runs, is one of the more valuable things a junior analyst can do.

3. Simpson's paradox

A pattern in every subgroup can reverse when the groups are combined. Two treatments for kidney stones, real published shape:

- Small stones: treatment A cures 81 of 87 (93%); treatment B cures 234 of 270 (87%).
- Large stones: A cures 192 of 263 (73%); B cures 55 of 80 (69%).
- Combined: A cures 273 of 350 (78%); B cures 289 of 350 (83%).

A wins both groups and loses overall, because doctors sent the hard cases to A. The lesson is not that aggregates lie. It is that a comparison is only meaningful once you know what determined who ended up in which group.

4. Regression to the mean

Take your worst-performing shops, intervene, and measure again. They improve. So would they have without you, because part of being the worst last month was bad luck, and luck does not repeat.

Any evaluation that starts by selecting the extreme of a distribution needs a control group selected the same way. Without one, you will report an effect that is partly an artefact of the selection, and the size of that artefact is not small.

5. Multiple comparisons

Slice a result by twenty segments at the conventional 5% threshold and you should expect about one "significant" finding from pure noise. Slice by a hundred and you expect five.

This is how dashboards produce a weekly false discovery. The fix is not sophisticated: decide the segments before you look, or state plainly that a post-hoc slice is a hypothesis rather than a finding.

6. Correlation, causation, and the third thing

Ice cream sales and drowning both rise in summer. The useful habit is not reciting the slogan but naming the candidate confounder out loud. When you cannot name one, say so; when you can, the conversation improves immediately.

The three routes to a causal claim, in descending order of strength: a randomised experiment; a natural experiment where something outside the system changed for reasons unrelated to the outcome; and adjustment for measured confounders, which only handles the ones you thought of.

What is not asked

Derivations. Obscure distributions. Proofs. In seven years of ordinary data work you may never use a t-test by hand, and no interviewer will ask you to integrate anything.

Free ways to learn this properly

OpenIntro Statistics is a full university textbook, free as a PDF, and it is better than most paid courses. Seeing Theory (Brown University) animates every idea above in a browser. For computation, `scipy.stats` and `statsmodels` in Python are free, R is free, and a spreadsheet handles most of it. If you want the mathematics behind the arithmetic, MIT OpenCourseWare publishes complete statistics courses at no cost.

The honest limitation

Statistical significance is not importance. A test on ten million users can return a real, robust, thoroughly significant improvement of 0.02% that is worth nothing and costs a month of engineering. The number that matters is the effect size and what it is worth in currency; the p-value only tells you whether you are looking at noise. Report both, always in that order.

BEFORE YOU MOVE ON

A team wants to test whether a new checkout screen improves conversion. Baseline conversion is 4%, they expect roughly a 5% relative improvement, and the site gets 9,000 visitors a week. What should be said before the test starts?

A. Run it for four weeks; 36,000 visitors is a substantial sample and will settle the question.

B. Detecting a difference of 0.2 percentage points at this baseline needs roughly 150,000 visitors per arm, so at this traffic the test would run for years — the honest options are to seek a larger change, use a more sensitive metric, or accept the decision cannot be made this way.

C. Use a one-sided test to increase sensitivity, which roughly halves the required sample.

D. Skip the test and ship it, since a 5% relative lift is small enough that the risk is low either way.

25 · 8 min

Git, and what a reviewer sees before your code

THE ONE THING TO KEEP

Your repository is read as evidence that you can work with other people, and the commit history is read before the code is.

The first thing opened, and what is looked for

A hiring manager clicks your GitHub link before they read your CV properly. They are not counting stars. In about forty seconds they are answering one question: *does this look like someone who has worked with others?*

What answers it:

- A README at the top of the repository that says what this is and how to run it.
- A commit history with more than one commit, and messages that are sentences.
- A `.gitignore`, so the repository is code and not a `.venv` directory and a 200MB CSV.
- No credentials anywhere.
- Branches and pull requests, even on a solo project — it shows you know the shape of team work.

What answers it badly: one commit called `final`, then `final2`, then `actually final`.

The commands that are ninety-five per cent of it

```
git status                  # what have I changed
git add -p                  # stage changes hunk by hunk, reading them
git commit -m "Fix timezone handling in the nightly load"
git push
git switch -c fix/timezones
git pull --rebase
```

`git add -p` is the one to adopt deliberately. It walks you through your own changes before you commit them, and it catches the debug print, the hard-coded path and the accidental password more often than any tool.

Commit messages: imperative mood, saying *why* where the *what* is not obvious. `Fix double-counting in the weekly report — orders were joined to items before summing` is worth more than `bug fix`, because the next person searching the history is you.

The secret that cannot be deleted

You commit a `.env` with an API key, notice, delete the file, commit again. The key is still in the repository, because git stores snapshots and every past snap-

shot is still there. On a public repository, automated scrapers index new commits within minutes, and several providers now scan public code themselves and revoke keys they find.

The order of the fix matters and people get it backwards:

1. **Revoke and rotate the key first.** Immediately. Assume it is already copied.
2. *Then* clean the history, with `git filter-repo` or the provider's support process.

Rewriting history first and rotating later leaves a live key in circulation while you do the harder task. Rotating first makes the exposed key worthless in a minute.

Prevent it: put `.env` in `.gitignore` on the first commit of every project, and keep a `.env.example` with the names of the variables and no values.

Pull requests as writing, not ceremony

A pull request description is the most-read thing you will write at work. The shape that works:

```
What this changes: the nightly load now retries on a
timeout instead of failing the whole run.

Why: the run died three times last week, always on the
same slow endpoint, and a rerun always succeeded.

How I checked: forced a timeout locally and confirmed
two retries then success; ran against last Tuesday's
data and the row counts match.

What I did not do: no backoff between retries yet —
worth adding if this recurs.
```

That takes four minutes and it is the difference between a review that takes an hour and one that takes five minutes. It is also, not incidentally, an audition for every promotion conversation you will ever have.

What to do with a large file

Do not commit the 400MB dataset. The repository keeps it forever, clones become painful, and some hosts refuse pushes over a size limit. Commit a small sample — a hundred rows — plus the script that fetches the real thing, and say in the README where the full data lives.

The free path

Git itself is free and always was. GitHub's free tier includes unlimited private repositories and CI minutes; GitLab and Codeberg are alternatives if you prefer. Nothing in this lesson needs a paid plan. GitHub Copilot and team plans are the paid products; neither affects how your history reads.

Why it feels arbitrary for a year

Git's commands feel inconsistent because they expose an underlying model rather than hiding it: git stores complete snapshots of your files, addressed by a hash of their content, and branches are just labels pointing at one of those snapshots. Once that clicks, `rebase` and `reset` stop being magic spells. Until it does, you will follow recipes and occasionally break things, and that is the normal experience of everyone who uses it — including people who have used it for a decade.

Keep one recipe handy for the situation that frightens beginners most: you have committed to the wrong branch, or you want to undo the last commit but keep the work.

```
git reset --soft HEAD~1     # undo the commit, keep the changes
staged
```

Nothing is lost. Almost nothing in git is ever lost, which is worth knowing at 23:00 when it appears to be.

BEFORE YOU MOVE ON

You realise you committed and pushed a `.env` containing a live API key to a public repository twenty minutes ago. What is the correct first action?

A. Delete the file and push a new commit removing it.

B. Make the repository private, then decide what to do.

C. Rewrite the history with `git filter-repo` to erase the file from every commit.

D. Revoke and rotate the key immediately, on the assumption it has already been copied, and only then clean the history.

26 · 10 min

The terminal, the container, and a cloud bill of zero

THE ONE THING TO KEEP

Cloud costs are dominated by resources that bill for existing rather than for working, so set a budget alert first and prefer things that scale to zero.

Where notebooks stop

Almost every course teaches you inside a notebook. Almost every job eventually puts you on a machine that has no notebook, no graphical interface and no mouse — a server, a container, a continuous-integration runner. The gap between those two worlds ends more early careers than any missing algorithm.

The good news is that the gap is small and closes in a fortnight.

Twenty commands cover most of it

`cd` `ls` `pwd` `cat` `less` `head` `tail` `grep` `find` `wc` `sort` `uniq` `cut` `sed` `awk` `chmod` `ssh` `scp` `curl` `tar`, plus `df` and `du` when a disk fills and

`ps` / `kill` when something will not stop.

The reason these are worth learning is that they compose. A single line answers a question that would otherwise be a twenty-minute script:

```
grep ERROR app.log | awk '{print $5}' | sort | uniq -c | sort -
rn | head
```

Read it right to left: find the error lines, take the fifth field, sort them together, count each distinct value, sort those counts descending, show the top ten. That is the most common diagnostic in operational work and it runs on a log file of any size that fits on disk.

Two habits make the shell survivable. Keep a plain text file of commands you had to look up — after two months you will stop consulting it, and the writing is what made that happen. And run `history | grep <word>` to find what you did last Tuesday, because you did not remember it either.

Environments, and why "it works on my machine"

Python's default behaviour installs packages globally, so two projects fight over one version of a library and eventually somebody's code stops running. The fix takes ten seconds:

```
python -m venv .venv
source .venv/bin/activate      # Windows: .venv\Scripts\acti-
vate
pip install -r requirements.txt
pip freeze > requirements.txt
```

`uv` is a newer, much faster, free tool that does the same thing and is worth adopting. Whichever you use, the rule is that a project states its dependencies in a file that is committed, and a stranger can reproduce your environment from that file alone. If they cannot, your portfolio project does not run for the reviewer, and a project that does not run is not evidence.

Containers, in one page

A container packages your code with its operating-system dependencies so it behaves identically wherever it runs. A complete, useful Dockerfile:

```
FROM python:3.12-slim
WORKDIR /app
COPY requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt
COPY . .
CMD ["python", "main.py"]
```

Two commands run it: `docker build -t myapp .` then `docker run myapp`. That is genuinely most of what a junior needs. Docker Desktop is free for personal use and small companies; Podman and Colima are free alternatives with no licence conditions at all.

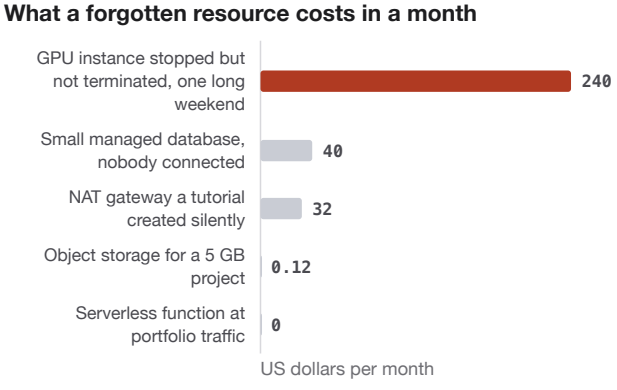
SSH, briefly

Generate a key once with `ssh-keygen -t ed25519`, put the public half on the server and on GitHub, and never type a password again. The private key never leaves your machine and never goes in a repository — a leaked key is an incident, and scanners find them within minutes of a public commit.

The free tier, and the bill that surprises people

You can do all of this at no cost. Cloud providers have free tiers; Oracle Cloud and Google Cloud both offer small always-free virtual machines, GitHub Actions gives free minutes on public repositories, and Colab and Kaggle give free GPU hours that are enough to train real models at small scale. A second-hand laptop running Linux, or WSL on Windows, is a complete environment.

The bills that catch people are not compute-hours spent doing work. They are resources that bill for **existing**:



The bill is for existing, not for working. Compute you actually used is rarely the surprise; a database nobody connects to and a GPU that was stopped rather than terminated are. Set a budget alert before creating anything, and prefer things that scale to zero.

- A managed database left running: a small one is roughly $15 to $60 a month, forever, whether or not anything connects to it.
- A NAT gateway, which many tutorials create silently: about $30 a month plus data charges.
- **Egress** — data leaving the cloud — charged per gigabyte, which is how a "small" export becomes a real number.
- A GPU instance you stopped using but did not terminate. At a few dollars an hour, a forgotten weekend is a few hundred dollars.

Two defences, both free and both taking five minutes. Set a billing alert and a budget on the day you create the account, before you create anything else. And prefer things that scale to zero — serverless functions, object storage, a container that runs on a schedule — over anything that runs continuously.

The honest limitation

None of this is intellectually difficult and all of it is unfamiliar, which is a worse problem, because unfamiliarity feels like inability. Expect two weeks of looking things up constantly. That is the normal cost, not a signal about you.

BEFORE YOU MOVE ON

A learner follows a tutorial, deploys a small demo, uses it twice, and receives a $94 bill the next month despite almost no traffic. What is the most likely explanation?

A. The demo was attacked or scraped, generating request volume they did not see.

B. Free tier limits were exceeded because free tiers only cover the first month.

C. The tutorial provisioned always-on resources — typically a managed database and a NAT gateway — which bill per hour for existing, so the cost is unrelated to how often the demo was opened.

D. Storage costs accumulated because the container images were never deleted.

27 · 7 min

The spreadsheet skill nobody puts on a syllabus

THE ONE THING TO KEEP

The organisation's real numbers live in spreadsheets, and being able to meet people there is what gets you access to the questions worth answering.

Where the numbers actually are

Every company has a data strategy document and a set of spreadsheets. The decisions are made from the spreadsheets.

Finance closes the month in one. Operations plans the rota in one. HR tracks headcount in one. The sales forecast that the board sees was assembled by hand on a Thursday. If you cannot work in that environment, you are cut off from the questions that matter, and you will spend your first year building dashboards nobody opens while the real work happens in a file called `FINAL_v7_updated_JK.xlsx`.

This is not a stepping stone to be endured. It is where you meet the business.

What is actually worth knowing

Six things, and they take a weekend:

1. **Pivot tables.** The fastest way to answer "by category, by month" for someone standing at your desk.
2. **`XLOOKUP`, or `INDEX` / `MATCH`.** Joining two sheets. `VLOOKUP` still appears everywhere and breaks whenever a column is inserted, which is why the others exist.
3. **Text to columns, and trim.** Because half of real spreadsheet data is one column that should be three, with trailing spaces.
4. **Conditional formatting to find anomalies.** Highlight the top and bottom one per cent and look at them. This finds more errors per minute than anything else in this lesson.
5. **Data validation.** So the person filling it in cannot type "N/a", "NA", "n/a" and "-" into the same column.
6. **Reading someone else's spreadsheet.** Which is a skill, and mostly consists of finding the hard-coded numbers hidden inside formulas.

The failure that has a famous example

Spreadsheets convert things that look like dates, silently, on paste.

Human gene symbols like `SEPT1` and `MARCH1` were being converted to dates by default settings, corrupting supplementary data in a substantial share of published genomics papers. The eventual fix was not to the software: in 2020 the naming committee renamed the genes. That is how hard this class of bug is to stamp out — an entire scientific field changed its vocabulary to work around a default.

The same mechanism eats your data every week in smaller ways. `1-10` becomes 1 October. A product code `5E3` becomes 5000, because it looks like scientific notation. A leading zero disappears from a phone number.

The defence: import as text, not paste. In Sheets and Excel, use the import dialog and set the column type explicitly, exactly as you set `dtype` in pandas. It is the same bug wearing a different coat.

Spotting a spreadsheet you cannot trust

- Numbers typed inside formulas: `=B4*1.18` — nobody knows what 1.18 is or when it changes.
- Merged cells, which break every attempt to read the file programmatically.
- Dates stored as text, which sort as `01/05`, `01/12`, `02/03` and look fine until you sum by month.
- Several sheets that should be one table, one per month, with slightly different column names.
- No row that reconciles against a source system.

When you find these, you have also found your first useful contribution.

The mistake juniors make with spreadsheets

You see the mess, build a proper pipeline and a dashboard, and present it. The team keeps using the spreadsheet.

The reason is not stubbornness. The spreadsheet lets them change an assumption and see the effect in two seconds, and your dashboard does not. Whatever replaces it must preserve the ability to ask a new question without asking you.

The move that works: leave the spreadsheet as the interface, and replace what feeds it. Automate the extraction and the cleaning, deliver a clean tab, and let them keep their formulas on top. Trust first, migration later.

The free path

Job adverts name Excel because organisations pay for it. Everything in this lesson works in **Google Sheets** or **LibreOffice Calc**, both free, and pivot tables, lookups and validation exist in all three with different names. If you are learning at home, learn in Sheets and translate; the concepts carry.

Know the ceilings, because knowing when to leave a spreadsheet is part of the skill. Google Sheets caps at ten million cells and becomes unpleasant well before that. Excel stops at 1,048,576 rows, which is exactly why a truncated export at that number should make you suspicious rather than satisfied. When a dataset approaches either limit, the answer is a database, and DuckDB will read the same file without complaint.

The interview question this prepares you for

"Here is a spreadsheet. How would you check whether it is right?" The strong answer is a routine, not an opinion: reconcile the total against the source system, check the row count, sort by each key column and look at the top and bottom five rows, count blanks per column, and confirm the dates are dates.

BEFORE YOU MOVE ON

You replace the operations team's weekly planning spreadsheet with a clean dashboard that shows the same figures more accurately. Two months later the team is still using the spreadsheet. What is the most likely reason?

A. They do not trust an automated system and need training on the new tool.

B. The spreadsheet lets them change an assumption and see the result immediately, and the dashboard only shows them numbers — so it answers the question they had last month but not the ones they have on Thursday.

C. Old habits take longer than two months to change, and the migration simply needs more time.

D. The dashboard is missing figures they rely on, so an audit of the fields is needed.

28 · 10 min

Evaluation: deciding what "correct" means, in advance

THE ONE THING TO KEEP

Anyone can produce an AI output; the scarce and durable skill is defining what counts as correct before you look, and then measuring it.

Why this is the skill worth having

Work that is well specified, abundantly exampled and quickly checkable is the work that gets automated first. Deciding what "good enough" means is none of those things. It is a judgement someone has to be accountable for, and organisations do not hand accountability to a tool.

That is the strategic argument. The practical one is simpler: almost no beginner shows evidence of having evaluated their own work, so doing it makes your portfolio unusual in the one way that matters.

Build the set by hand, and build it small

Thirty to fifty cases. Not three thousand. You are going to read every one of them personally, several times, and a set you will not read is a set you will not learn from.

Where the cases come from:

- **Real questions people actually asked.** Support tickets, search logs, the questions in a team's chat channel. These are worth ten invented ones.
- **The edge cases the team argues about.** Every team has three examples they disagree on. Those are the most valuable cases you own.
- **Three cases that should fail.** A question the system has no information to answer, a request it should refuse, and a question in another lan-

guage or dialect. What a system does when it should say "I don't know" is where most of the risk lives.

The rubric, and why three points beats five

For each case, write down what a correct answer must contain, before you see any output. Then grade.

Use binary correct/incorrect wherever you can. Where you cannot, use three levels: correct, partly correct, wrong. Resist the five-point scale — graders do not agree on the difference between a 3 and a 4, agreement between them collapses, and once two people scoring the same output produce different numbers, the average stops meaning anything. Three coarse levels that people agree on beat five fine ones they do not.

What a record looks like

```
case_id: 017
question: "What is the refund window for orders shipped to
Nigeria?"
must_contain: ["30 days", "counted from delivery, not order
date"]
answer: "You can request a refund within 14 days of ordering."
verdict: wrong
note: retrieved the UK policy page; the Nigeria page exists
      but is titled 'West Africa returns' and did not match
```

The `note` field is where the value is. "Wrong" is a number. "Retrieved the wrong document because the title uses a regional name" is a fix.

The two numbers to report, and the one to lead with

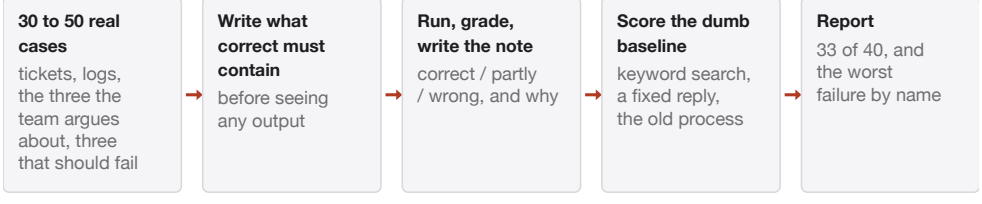
Report accuracy on the set — *33 of 40 correct* — and report the **worst failure** by name. A single named failure moves a decision more than any average, because a stakeholder can picture it. "It confidently gave a customer in Lagos the UK refund window" is a sentence that gets a bug fixed. "82 per cent accuracy" is a sentence people nod at.

The baseline almost nobody builds

Before you report that your system scores 82 per cent, score the dumb version on the same forty cases. Keyword search over the documents. Or a fixed reply saying "contact support". Or the previous process.

If plain keyword search scores 74, your honest headline is *eight points above search*, and that changes what the thing is worth. Reporting 82 with no baseline is not lying, but it lets everyone in the room assume the baseline was zero, which it never is.

An evaluation, start to finish



The rubric is written before any output is seen, and the dumb baseline is scored on the same forty cases. That is what turns 'eighty-two per cent' into 'eight points above keyword search', which is what the thing is actually worth.

Using a model as a judge, carefully

Grading by hand does not scale past a few hundred cases, so people have models grade outputs. It works, with two known biases: judges prefer longer answers, and they prefer text that reads like their own output. Both inflate scores in ways unrelated to correctness.

The discipline: hand-label about fifty cases, run the judge on the same fifty, and measure how often they agree. If agreement is poor, fix the rubric before you trust any large run. Re-check whenever the judge's model version changes, because a provider updating a model silently changes your measuring instrument.

The free path

You do not need an evaluation platform. A CSV of cases, a script that runs each one, and a column for the verdict is a complete evaluation harness, and it is what most teams actually use for the first year.

The named paid products — LangSmith, Braintrust, Weights & Biases Weave — buy you tracing, dashboards and collaboration when many people grade together. The free equivalents are **promptfoo** (open source, config-file driven), the open evaluation harnesses on Hugging Face, or sixty lines of Python. Start with the sixty lines; you will understand what the product is doing when you eventually need one.

The limitation that catches everyone

An evaluation set built once goes stale. You fix the cases in it, the score climbs to 95 per cent, and the system is no better in production — because you have optimised against forty fixed examples rather than against reality. This is overfitting to your test set, and it happens quietly.

The fix is a refresh habit: every month, sample fresh cases from real traffic and add them, retiring some old ones. Keep a small held-out set you never look at while developing. And treat any score above about 95 per cent on a hand-built set as a sign that the set has become too easy, not that the work is done.

BEFORE YOU MOVE ON

Your assistant scores 82 per cent on a 40-case set you wrote. Your manager asks whether it is ready to put in front of customers. Which response is strongest?

A. "Yes — 82 per cent is above the 80 per cent threshold we agreed as the bar for launch."

B. "Not yet — I need a much larger test set before the figure is statistically reliable."

C. "Yes, with a disclaimer on the interface telling customers the answers may be inaccurate."

D. "Plain keyword search scores 74 on the same cases, so we are eight points better than search. Of the seven failures, five are one bug — it retrieves the UK policy for African customers. I would fix that, re-run, and add twenty fresh cases sampled from real questions before deciding."

29 · 7 min

A portfolio when you have never been paid for this

THE ONE THING TO KEEP

A portfolio proves judgement under mess, not that you can finish someone else's tutorial.

What a reviewer actually does

Someone hiring gives your project a minute, maybe two. They open the repository, read the top of the README, glance at one file, and decide whether to spend more time. Almost nobody runs your code.

So the portfolio's job is not to demonstrate that the code works. It is to show, quickly, that you made decisions in a situation where the answer was not published anywhere.

What does not work

Titanic survival. MNIST digits. A sentiment classifier on a movie review dataset. A chatbot wrapper over a public model with no evaluation. These are not bad projects — they are fine as practice. They are bad *evidence*, because ten thousand people submitted the same thing and none of the decisions were yours. The dataset was clean, the target was given, the metric was chosen for you.

A notebook with forty cells and no README is also not evidence. Nobody reads it.

What works

One to three projects, each solving a problem that a specific real person or group actually had. Sources of such problems, in order of how easy they are to

reach:

- Your own repeated annoyance. Something you do by hand every week.
- A small organisation near you: a clinic, a school, a farmers' cooperative, a two-person shop, a student society. They all have data in spreadsheets and a question they cannot answer.
- A public dataset that is genuinely messy — municipal budgets, transport delays, rainfall, exam results — where cleaning it *is* the work.

The defining feature is mess. Mess is where judgement becomes visible.

Write it down like this

```
# Bus delay tracker — Lagos, route 57

What it does: tells the cooperative's drivers which departure
window is late most often, updated nightly.

Who uses it: 6 dispatchers, since March 2026.

The data problem: three of the eleven months had GPS pings
recorded in local time and eight in UTC. Nothing labelled
this. I found it because "delays" spiked exactly one hour.

What I decided and why:
  - Dropped the first two weeks entirely (device swap, no
    reliable timestamps) instead of interpolating.
  - Median, not mean — two breakdowns skewed everything.

What it gets wrong: no data for public holidays. A dispatcher
caught this before I did.

Run it: make dev
```

That page is worth more than the code beneath it. It shows you noticed something, chose between two defensible options, and were honest about a limitation.

Measure something

The single strongest signal in an AI-adjacent portfolio is that you evaluated your own work. Not "it produces good answers" — a table of thirty test cases you wrote by hand, what the system did on each, and how many it got wrong. Anyone can build something that works on the example they demoed. Almost no beginner shows what it does on the cases they did not choose.

Two practical rules

Deployed beats local, because a running link takes five seconds to check and a repository takes five minutes. A free hosting tier is enough.

Small and finished beats large and abandoned. A tool that does one thing for thirty people is stronger than an unfinished platform. If your project cannot be described in one sentence, cut it until it can.

BEFORE YOU MOVE ON

Two applicants. One has twelve finished course projects. One has a single tool that thirty people at a farmers' cooperative use every week, plus a written account of a bug that corrupted two days of data and how it was found. The second usually gets the interview. Why?

A. Because deploying and running software in the real world is the skill employers are paying for.

B. Because twelve similar projects suggest someone who cannot focus on anything long enough.

C. Because the second person worked on a problem with no published answer and showed how they decided between options.

D. Because working for a real organisation shows commitment and good character.

30 · 9 min

Where portfolio data comes from, and what you may do with it

THE ONE THING TO KEEP

A dataset's licence, its provenance and whether it describes people decide what you may do with it, and the absence of a licence means you have no permission rather than a free hand.

The question nobody asks until it is a problem

You need data for a project. The instinct is to take some. Three questions decide whether that is fine, awkward, or career-ending: where did it come from, does it describe identifiable people, and what does its licence permit.

Where good data actually lives

- **National statistics and open-government portals.** India's data.gov.in, the UK's data.gov.uk, the EU portal, the US data.gov, Nigeria's and Kenya's open data initiatives, Brazil's dados.gov.br. Census, health, transport, agriculture, budgets, procurement. Usually the most interesting data available anywhere, and almost nobody uses it, which is exactly why a project built on it stands out.
- **International bodies.** World Bank, WHO, UN, FAO, OECD. Cleanly licensed and well documented.
- **Research repositories.** UCI, Zenodo, OpenML, Hugging Face Datasets, Papers with Code.
- **Kaggle.** Enormous and convenient, with a caveat below.
- **Your own.** Data you generate, log or collect with consent. The strongest option for a portfolio, because nobody else has it.

Licences, in five minutes

The licence is a permission, not a formality:

- **CC0 / public domain** — do anything.
- **CC-BY** — do anything, credit the source.
- **CC-BY-SA / ODbL** — as above, but derived datasets must carry the same licence. OpenStreetMap is ODbL, and this catches people.
- **Non-commercial (CC-BY-NC)** — fine for a portfolio, not fine inside a product.
- **"Research use only"** — common on academic sets, and it means what it says.
- **No licence at all** — the default is that you have no permission. Absence of a licence is not permission.

The Kaggle caveat: many datasets there were uploaded by somebody who was not the rights holder, and the licence field was chosen by that uploader. A dataset marked CC0 that is plainly a scrape of a commercial site is not CC0. Trace it to the original source when the provenance looks thin.

Scraping: the honest state of it

Technically easy. Legally unsettled, and unsettled differently in different countries. The shape of the disagreement, which is the most anybody can honestly give you:

- Access to **publicly available** pages has been treated relatively permissively in some United States computer-misuse rulings, while a site's **terms of service** are a separate contractual question that those rulings did not resolve.
- The **European Union** has a database right that has no American equivalent, and separate text-and-data-mining provisions with an opt-out for commercial use. **Japan** and **Singapore** have broader mining exceptions.
- **Copyright in AI training data** is being litigated in several countries right now, with decisions pointing in different directions. Anybody who tells you this is settled is describing their preference.

None of this is legal advice, and this course cannot give any. What is defensible practice regardless of jurisdiction: respect `robots.txt`, rate-limit yourself, prefer a documented API where one exists, never take personal data, never redistribute the raw scrape, and be able to explain in one paragraph why you believed you were entitled to the data. If a project cannot survive that paragraph, choose different data — there is plenty.

Personal data is regulated even when it is public

If it identifies a person, a data protection law probably applies: the GDPR in Europe, India's Digital Personal Data Protection Act, Nigeria's Data Protection Act, Brazil's LGPD, and others. "It was on a public profile" is not an exemption in any of them.

For a portfolio there is no upside. Aggregate, anonymise properly, or pick a different subject.

Never your employer's data. Not once.

This ends careers, and the mechanism is worth stating plainly. The data is not yours; it is covered by your employment contract, usually by a customer contract underneath that, and frequently by regulation on top. Publishing a chart built from it can be a breach of all three at once, and a reviewer who spots it will assume you would do the same to them.

What to do instead: describe the problem and the method, and reproduce the *shape* of the data synthetically. `Faker`, `SDV` and twenty lines of NumPy generate a table with the same columns, cardinalities and rough distributions. Synthetic data is excellent for demonstrating a pipeline, a model or an interface. It is worthless for demonstrating a finding, and you should say which of the two you are showing.

The habit that costs one minute

Every project gets a `DATA.md` with the source URL, the licence, the download date and one line on any preprocessing. It takes a minute, it answers the re-

viewer's question before they ask it, and in a field where provenance is the live argument of the decade, it marks you as somebody who thinks about it.

BEFORE YOU MOVE ON

An applicant's portfolio project analyses customer complaints. In the interview they explain the data came from their current employer's support system, with names removed. What does a reviewer most reasonably conclude?

A. The work is acceptable because removing names satisfies data protection requirements.

B. The applicant is resourceful and has demonstrated access to realistic data, which is a point in their favour.

C. The work should be accepted if the employer has no explicit policy against it.

D. The applicant published data they did not own, likely breaching their employment contract and possibly a customer contract, and would do the same with this company's data.

---

31 · 8 min

Writing up a project as a decision record

THE ONE THING TO KEEP

Write the project as a record of decisions — the mess, the choice, the number, and what it still gets wrong — because that is what a reviewer is reading for.

The ninety-second reading order

A reviewer opens your project page and reads in this order: the title, the first sentence, whatever number is visible, and then — if you have earned it — the section admitting what is wrong. They almost never read the code.

So write for that order. Everything that proves something goes above the fold, and the honest limitations go near the top rather than buried at the bottom,

because their presence is itself the strongest signal on the page.

The template

```
# Bus delay tracker — Lagos, route 57

Tells a transport cooperative's dispatchers which departure
window runs late most often. Updated nightly.

**Used by** 6 dispatchers, weekly, since March 2026.

## The mess
Three of eleven months had GPS timestamps recorded in local
time and eight in UTC, with nothing labelling which. I found
it because "delays" spiked by exactly one hour on the day the
device firmware changed.

## Two decisions
- Dropped the first two weeks entirely rather than
  interpolating. The device swap meant I could not tell
  which clock any reading came from, and inventing a
  correction would have hidden that.
- Median rather than mean. Two breakdowns of four hours
  each were dragging every weekly average upward.

## Does it work
I wrote 30 test journeys with known outcomes from the
dispatchers' own paper log. It classifies 27 correctly.
Of the 3 misses, all are journeys crossing midnight.

## What it gets wrong
No handling of public holidays — a dispatcher found this
before I did. Midnight-crossing journeys, as above.

## Run it
`make dev`, then open localhost:8000. Sample data included.
```

That page takes ninety minutes to write and outperforms three more projects.

Why each section is there

The mess and how you found it. This is the only part of the page that cannot be faked by copying a tutorial. The detail that you noticed the spike was *exactly* one hour is worth more than any architecture description, because it shows how you think when something is wrong.

Two decisions, with the rejected alternative. A decision with no discarded option is not a decision. Naming what you did not do is what distinguishes judgement from following instructions.

A number. From the previous lesson. Thirty test cases and a count. Most beginner portfolios contain no number at all, and the ones that do stand out immediately.

What it gets wrong. Counter-intuitive and true: admitting a limitation raises your credibility on everything else on the page, because it tells the reader you know how to look for problems. A page with no limitations reads as a page written by someone who has not checked.

The page, in the order a reviewer reads it

Title and first sentence: what it does, for whom, how often	read by everyone
Used by: a real count, with dates	the visible number
The mess, and exactly how you found it	the part a tutorial cannot fake
Two decisions, each with the alternative you rejected	judgement, not instructions
Does it work: thirty cases and a count	27 of 30, and what the misses share
What it gets wrong	raises credibility on everything above it
Run it: one command	a project that does not run is not evidence

Ninety seconds: the title, the first sentence, whatever number is visible, then the section admitting what is wrong. They almost never read the code, which is why the mess and the number sit above it.

Publishing work you did for someone else

If the project was for a client or an employer, you cannot simply publish it.

- **Ask, in writing, before you publish anything.** An email saying "may I write publicly about the approach, with your name and figures removed?" is usually answered yes, and it protects you.
- **Never publish their data.** Not a sample, not an anonymised extract, not "just the schema" if the schema is distinctive.
- **Replace real figures with realistic invented ones and say so.** "Figures changed for confidentiality; proportions preserved" is honest and reviewers accept it.
- **A blurred screenshot is not anonymisation.** The underlying CSV may still be in your repository, and blur can sometimes be reversed. Regenerate the screenshot with fake data instead.
- **If you cannot publish it at all**, write a page about the *problem shape* with no client details. "How I found a timezone bug in a delivery dataset" needs no client name.

Show it moving

A forty-second screen recording of the thing working outperforms three paragraphs, because it proves the project exists in a way a repository does not. Free tools do this: **OBS Studio** on any platform, the built-in recorder on macOS and Windows, **Peek** on Linux. Loom is the paid product people reach for; nothing about a forty-second clip needs it.

Keep it short and unnarrated. Nobody watches minute two.

The mistake to avoid

Writing the page as a tutorial. "First, we import pandas. Then we load the data." The reviewer is not learning your project; they are making a decision about you, and every sentence that teaches them Python is a sentence not spent on evidence.

Write for a peer who could have built it themselves, and tell them the parts they would not have predicted.

The limitation

This only works if the project is real. Inventing a user count is checkable — a reviewer may ask which cooperative, or when the last commit was, or what the dispatchers complained about — and the conversation ends there. One genuinely small real project beats an impressive imaginary one, and it is much less exhausting to talk about.

BEFORE YOU MOVE ON

Which of these belongs on a project page, near the top, even though beginners usually leave it out?

A. A short, specific account of what the project still gets wrong, and how you know.

B. A detailed architecture diagram of every component and how they connect.

C. A step-by-step explanation of the code, so a reader can follow how it was built.

D. A list of every technology and library used, to match the keywords in job adverts.

CASE STUDY · MODULE 3

The clinic's spreadsheet, and what could be published

Farhan, a final-year student in Bhopal, building his first portfolio project from a two-doctor clinic's appointment records.

His aunt runs the reception desk at a clinic near Arera Colony. Fourteen months of appointments live in a spreadsheet called `appointments_master_-final.xlsx`, maintained by hand, and about a fifth of booked patients do not arrive. The doctors lose the slot; the patients who could have had it wait another week.

It is a genuinely useful problem and Farhan had the data on his laptop within a day. Then three separate questions arrived, in the wrong order.

The mess, which was the interesting part

The first thing the file taught him had nothing to do with prediction. Three months in the middle of the fourteen used day-first dates and the rest used month-first, with nothing marking the change. He found it because Tuesday appointments spiked implausibly in exactly those three months, which is what a day-month swap looks like when you plot it.

He had two defensible options and had to choose one. Re-parse those months with the other convention and keep the rows, accepting that any row where both numbers were twelve or lower could still be wrong and he would never know which. Or drop the three months entirely and lose a fifth of his data. He dropped them, and wrote down why, because inventing a correction would have hidden the fact that he could not tell.

The data question

Patient names, mobile numbers, ages, and a free-text `reason` column that in about four hundred rows contains a diagnosis. Nothing about that file is his. It belongs to the clinic, it describes identifiable people, and India's data protection regime does not contain an exemption for "my aunt gave it to me".

A blurred screenshot would not have helped either — the underlying file would still have been in the repository.

The measurement question

He built forty test cases by hand from the last two months, wrote down for each what a correct prediction would be, and scored his model: eighty-six per cent.

Then he did the thing almost nobody does and scored the dumb version on the same forty cases. Predicting that every patient shows up — a rule requiring no model, no Python and no laptop — was right eighty-two per cent of the time, because most people do turn up.

So the honest headline was four points above a constant. Reporting eighty-six per cent alone would not have been a lie. It would have let every reader assume the baseline was zero.

The decision

Three routes were open, and each cost something.

Publish the real analysis. Strongest evidence: real mess, real decisions, a real user. Also a breach of the clinic's trust, a probable breach of the law, and a reviewer who spots patient data in a public repository will assume he would do the same with theirs.

Publish with synthetic data. Safe, and honest if labelled. Also much weaker, because synthetic data demonstrates that a pipeline runs and demonstrates nothing about a finding. He would have a project that proved he could write code and proved nothing about judgement.

Ask, in writing, and publish what is permitted. Slowest, and it required a conversation with a doctor who had not asked for any of this and who might reasonably say no — in which case he would have spent three weeks and had nothing.

He also had to decide whether to report the eighty-two per cent baseline at all, three weeks before applications opened. Nobody would have known it existed.

He had computed it himself, in an afternoon, for his own benefit, and deleting one line from a page would have turned a four-point improvement into an eighty-six per cent result.

The reviewer he was actually writing for

He kept forgetting who the audience was. A reviewer gives a portfolio project between one and two minutes: the title, the first sentence, whatever number is visible, and — if the page has earned it — the section admitting what is wrong. Almost nobody runs the code, and nobody at all reads a notebook of forty cells.

So the page had to carry the mess, two decisions with the option he rejected for each, one number, and the limitations, in that order and above the fold. The model itself was fifteen lines of scikit-learn and the least interesting thing on the page.

> **STOP HERE**
>
> Before turning the page: what would you have done, and what would it have cost if you were wrong? Write it down. The point of the next section is to compare.

WHAT HAPPENED

He emailed the doctor asking whether he could write publicly about the approach, with the clinic unnamed, no patient-level data, and figures changed while proportions were preserved. The reply took nine days and was yes, with one condition: he was to show her the page before it went up. The published case study contains the mess (three months of the spreadsheet used a different date format, which he found because "Tuesday" appointments spiked), two decisions with the rejected alternative for each, the sentence "a rule that predicts everyone attends scores 82% on the same 40 cases; this scores 86%", and a list of what it gets wrong. A `DATA.md` records the source, the permission and the date. Two months later an interviewer at a health-claims company spent most of the first round on the baseline sentence, and told him after-

wards that it was the reason he was in the room — not because four points is impressive, but because almost nobody reports the number that makes their own work look smaller.

Why does reporting the 82% baseline make the project stronger rather than weaker?

Because it tells a reviewer what the work is actually worth and shows that Farhan knew to ask. A number with no baseline invites the reader to assume the comparison was zero, and a reviewer who has done this work will assume the baseline was never computed. Reporting it converts a possibly-inflated claim into a checkable one, and the habit it demonstrates — measure the dumb version first — is the thing being hired.

The clinic is his aunt's employer and the data was handed to him freely. Why is that not permission?

Because the data is not his aunt's to give. It belongs to the clinic and it describes identifiable patients, so the clinic's obligations and the law apply regardless of who copied the file onto whose laptop. Permission has to come in writing from somebody with the authority to give it, and it has to specify what may be published — which is exactly what he eventually obtained.

If the doctor had said no, what should he have published?

A page about the problem shape with no clinic details, the method described, and the pipeline demonstrated on synthetic data with that fact stated plainly. He would have to say which of the two he was showing: synthetic data is good evidence for a pipeline and worthless evidence for a finding. It is a weaker project and it is still publishable; what is not publishable is the real data.

MODULE 3 · TEST

Check what stuck

6 questions, one right answer each. This one is for you, not for a record: answers and the reason behind each are at the back. If two questions from the same module go wrong, re-read that module before the exam rather than after.

1 You join `orders` to `order_items` to filter by product, then sum `orders.total_amount`. Revenue comes out about three times too high. Why?

A. The join dropped orders with no items, so the remaining totals were rescaled.

B. Summing across two tables double-counts the tax component of each order.

C. The item table stores amounts in minor units, so the sum is inflated by rounding.

D. Each order appears once per item, so its total is counted once per matching row.

2 `where status <> 'cancelled'` returns 9,100 of 10,000 orders, but only 400 are actually cancelled. Where did the other 500 go?

A. They have a NULL status, and comparing anything to NULL is unknown rather than true.

B. They have a status with trailing whitespace, which fails an exact string comparison in SQL.

C. They were removed by the query planner because the column has no index.

D. They have a status in different letter case, and the comparison is case-sensitive.

3 After `pd.read_csv`, a customer ID of `007` reads as `7.0` and a postcode of `01234` as `1234`. What happened, and what prevents it?

A. The file was saved with a byte-order mark; open it with an explicit encoding argument instead.

B. The CSV writer stripped the quotes; re-export with quoting forced on.

C. One blank row made pandas infer a float column; pass an explicit dtype instead.

D. The index column was written into the file; pass index_col to skip it.

4 A conversion rate measured on 400 visitors has a 95% interval of about plus or minus three points. Roughly how many visitors are needed to halve that to plus or minus 1.5 points?

A. About 800, because the interval scales with the inverse of the sample

B. About 1,600, because precision improves with the square root of the sample

C. About 40,000, because intervals shrink only as samples approach the tens of thousands

D. About 600, because the improvement flattens quickly after a few hundred

5 You report that your assistant answers 82% of forty test questions correctly. Why is that number incomplete on its own?

A. Forty cases is too few for the figure to be reported as a percentage.

B. Accuracy should always be reported alongside precision and recall.

C. Without a baseline, everyone assumes the comparison started at zero.

D. The test set should have been generated from real traffic rather than written by hand.

6 A student's first cloud bill is $84 after a weekend of practice that used almost no compute. What is the most likely cause?

A. Storage of the practice datasets, which is charged per gigabyte for every hour it exists

B. The free tier expiring silently partway through the weekend

C. Repeated container builds, which are billed per build minute

D. Resources that bill for existing, such as a managed database left running

MODULE 4

Getting seen: the routes that put a person in front of your work

Applying cold is the most common strategy and the weakest one. Every route that works has the same shape — a human sees your work before, or instead of, an application landing in a pile. This module covers the paper, the message, the job beside the data, and the first person who pays you.

> BY THE END OF THIS MODULE YOU CAN
>
> Run a search that puts your work in front of named people rather than into application queues, and identify the bridge role or first paid job you would take to get inside.

---

32 · 8 min

How people actually get in

THE ONE THING TO KEEP

Most people enter this field sideways, from an adjacent job or through someone who vouches.

The uncomfortable base rate

Applying cold to public postings is the most common strategy and the weakest one. A single AI-adjacent posting at a known company can draw several hundred to a few thousand applicants, most auto-screened. Sending two hundred

applications and hearing back twice is not bad luck; it is the ordinary output of that method.

The methods that work all have the same shape: a human sees evidence of your work before, or instead of, an application landing in a pile.

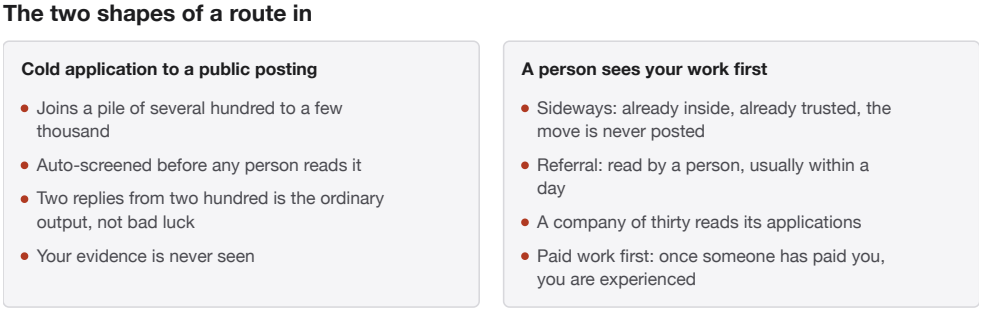
The four routes that actually carry people

Sideways from an adjacent job. The most reliable and least discussed. Get any job at an organisation that has data — support, operations, QA, finance, field work — and become the person who answers questions with numbers. You already have context nobody outside can buy, you are already trusted, and internal moves skip the entire screening machine. Large numbers of working analysts and data engineers arrived this way.

Referral. A referred application is read by a person, usually within a day. This does not require a network of executives. It requires being known by ten or fifteen people who do this work, which is a matter of months of showing up in the same places and being useful.

Small companies. A company of thirty reads its applications. A company of thirty thousand does not. Your careful cover letter is only ever read at the small one.

Paid work first. A short contract, a freelance job, a three-month project. Once someone has paid you, you are experienced, and every subsequent conversation starts differently.



Every method that works has the same shape: a person sees evidence of your work before, or instead of, an application landing in a pile.

The market you are in changes the shape

India. Campus placement and the large service companies remain the widest entry route, with aptitude and coding tests up front. Getting in at scale then moving internally into a data team is a completely legitimate path and a common one. Notice periods of 60 to 90 days are normal and affect how employers plan hiring.

Nigeria and Kenya. Community is the hiring channel. Developer groups, talent programmes, public work shared where people can see it. A large share of good jobs are remote contracts with foreign companies, which means your competition is global and your evidence must be public.

Latin America. Nearshore staffing firms placing engineers with US companies are a major employer. The real gate is often spoken English in meetings rather than technical depth.

Europe. Structured, slower processes, several rounds, formal references. Visa sponsorship narrows the list of employers sharply, and in Germany, France and Spain a working level of the local language quietly decides many mid-level roles.

United States. Referral-heavy, take-home exercises common, brutal application-to-interview ratios at known names. Non-obvious employers — hospitals, insurers, logistics firms, universities, state agencies — hire steadily with far less competition.

Southeast Asia. Regional scale-ups hire in cohorts, often with structured graduate intakes that behave more like the Indian model than the American one.

What to do this month

Pick six companies you could plausibly join, not sixty. Find one person at each who does the work. Read something they wrote, then send four sentences: what you built, one specific question about their problem, no request for a job. Some will not reply. Two or three will.

That is a worse-feeling and much higher-yield use of an evening than another fifty applications.

BEFORE YOU MOVE ON

Meera has sent 200 applications through job boards over six months and received two replies. She writes clearly and her projects are solid. Which change is most likely to help?

A. Rewrite the CV with more of the exact keywords from each posting so it survives automated screening.

B. Find routes where a person who can vouch for her sees her work before any application exists.

C. Apply more widely, including to roles that are a stretch, to increase the number of chances.

D. Add a recognised certification so that the CV clears the first screen.

---

33 · 9 min

Running the search as a funnel

THE ONE THING TO KEEP

A job search is a funnel with measurable conversion at each stage, and the route in changes the top-of-funnel rate by roughly ten times, which no amount of extra applications can compensate for.

Feelings are a bad instrument

Three weeks into a search everybody has the same two thoughts: *nothing is working* and *maybe I am not good enough*. Neither is information. A funnel is.

Every search has the same five stages, and the only useful question is which one is leaking.

1. Applications sent or approaches made
2. First conversations — recruiter or hiring manager
3. Technical stages — take-home or live exercise
4. Final stages
5. Offers

The arithmetic, worked

Rates vary enormously by market, month and role, so treat these as a starting estimate to be replaced by your own within thirty applications. Typical shapes in an ordinary market:

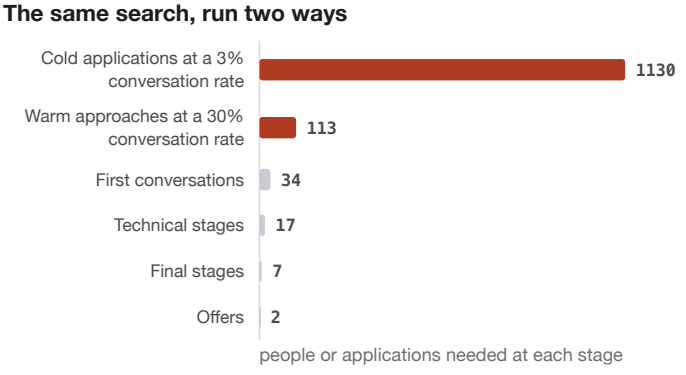
- **Cold applications → first conversation: 2% to 5%.** For heavily advertised remote roles it can be well under 1%, because a single posting can attract two thousand applicants.
- **Referred or directly-approached → first conversation: 20% to 50%.** This is the entire argument of this module, expressed as a number.
- **First conversation → technical: around 50%.**
- **Technical → final: around 40%.**
- **Final → offer: around 30%.**

Work backwards from two offers, which is the right target because one offer is not a negotiating position:

```
2 offers  ->  ~7 finals  ->  ~17 technicals  ->  ~34 first
conversations
```

Thirty-four first conversations at a 3% cold rate is **1,130 applications**. At a 30% warm rate it is **113 approaches**.

Those two numbers are the same search run two ways. One of them is possible for a person with a job and a life; the other is not. This is why the lessons on writing to strangers and on bridge jobs come before this one.



Working back from two offers at typical rates: 30% of finals convert, 40% of technicals reach a final, 50% of first conversations reach a technical. The top of the funnel is where the route decides everything, by a factor of ten.

Track it, in a spreadsheet, from application one

Seven columns: date, company, role, route (cold, referral, recruiter, direct message), stage reached, outcome, next action and date. Nothing more. Ten seconds per row.

After thirty rows you can answer the only question that matters — where does it break — and you will stop having to guess.

Reading your own funnel

Almost no first conversations. The problem is upstream of your ability: targeting, route or the top third of your CV. If you are applying cold to roles two levels above your scope, no amount of technical improvement moves the number. Change route before changing yourself.

Conversations but no technical stages. Something in the first fifteen minutes is not landing — usually a rambling answer to "tell me about yourself", or a salary expectation given badly. Both are fixable in an afternoon and neither is about your skills.

Technical stages that end there. Now it is skills, and now — only now — the answer is more practice. Ask for the reason. About a third of companies will tell you something specific.

Finals without offers. Rarely technical. Usually stories, evidence of judgement, or a stated concern nobody surfaced. Ask what the reservation was; at final stage people often answer honestly.

Volume is not the lever people think it is

The instinct when nothing happens is to send more applications. If your cold rate is 2%, doubling from 200 to 400 applications buys four extra conversations and about a hundred hours.

The same hundred hours spent making twenty warm approaches — a specific person, a specific reason, evidence attached — buys six. This is not motivational; it is division.

Two honest limitations

First, these rates move with the market. In a tight hiring year the cold rate can fall below 1%, and a funnel that would have worked in 2021 produces nothing. That is not a statement about you, and confusing the two is what makes long searches damaging.

Second, a funnel measures throughput and says nothing about fit. Someone who accepts the wrong job efficiently has optimised the wrong thing. Keep a line in the tracker for why you wanted each role, and read it back after twenty entries. If most of them say "it was open", the search has drifted.

BEFORE YOU MOVE ON

After 180 cold applications, Rahul has had four first conversations, three of which reached a technical stage and two of which reached a final. He has no offers. Where is his search actually leaking?

A. At the technical stage, since one of three conversations did not progress and that is where preparation pays off.
B. At the top: a 2% cold rate is normal, so the pipeline is starved rather than broken, and the fix is a warmer route — referrals and direct approaches — not better interviewing.
C. At the final stage, and he should focus on negotiation and closing.
D. Nowhere; he should send another 180 applications, since the conversions are healthy.

34 · 8 min

A CV that survives a thirty-second read

THE ONE THING TO KEEP

A CV is scanned for scope, outcome and a number; formatting exists only to make those findable fast.

What actually happens to it

Your CV gets well under a minute on the first pass, often nearer twenty seconds. In that time someone is answering four questions: what is this person now, have they done anything close to this, is there a number anywhere, and can I see something they made.

Everything below follows from that.

The structure that works

1. **Name, one line of positioning, and links.** The positioning line is not a summary paragraph. It is eight words: *Analyst — hospital operations data, SQL and Power BI.* The links are your deployed project first, GitHub second, LinkedIn third.
2. **Whichever is stronger: experience or selected work.** If you have no relevant jobs, put projects first and call the section **Selected work**, not "Personal projects" — the second phrasing invites the reader to discount it.
3. **Skills**, grouped and honest. Three groups, not a wall of forty words.
4. **Education last**, unless it is your strongest asset.

One page under about eight years of experience. Two only if the second page earns itself.

Bullets: what, how, result, number

This is the whole craft, and it is a rewriting exercise.

Before: *Responsible for data cleaning and analysis for the sales team.*

After: *Rebuilt the weekly sales report from three exported spreadsheets into one scheduled SQL job — cut preparation from four hours to fifteen minutes, and removed a double-counting bug that had inflated Q2 revenue by about eight per cent.*

The second is not a better-written version of the first. It contains information the first does not: what the situation was, what you did, and what changed. Every bullet should survive the question *so what happened?*

If you genuinely have no numbers, use scale instead: how many people, how many rows, how often it runs, how long it took. *Used by six dispatchers, weekly, since March* is a number.

Applicant tracking systems, without the folklore

An enormous amount of nonsense is sold around this. The reality is dull. Tracking systems are text extractors that pull your document into fields. What actually breaks them:

- Text inside an image, including a header rendered as a graphic. There is nothing to extract.
- Multi-column layouts, where extraction interleaves the columns into unreadable order.
- A PDF exported from a design tool with no text layer.
- Section headings the parser does not recognise — "My Journey" instead of "Experience".
- Sometimes tables, depending on the parser.

What does not break them: a normal PDF exported from a word processor, in one column, with standard headings.

So "ATS-optimised templates" sold for a fee are mostly charging you to avoid a problem you would not have had. Free path: Google Docs, LibreOffice Writer, or a plain LaTeX template. Paid: Canva Pro and subscription resume builders — and a Canva export can parse *worse* than a Word document, because of how the text layer is produced.

Keywords, honestly

Match the advert's vocabulary where it is true of you. If the posting says "Power BI" and you use Power BI, write "Power BI" — not "BI tooling", not "data visualisation platforms". A human is scanning for the same word the requisition used, and so is the parser.

That is matching. Stuffing — listing tools you have not used, or a white-text keyword block — is checkable in about one interview question, and it ends the process.

With no experience at all

Lead with **Selected work**. Three projects, three lines each: what it does and for whom, the hard part, the number. That is the same structure as the case study page, compressed.

Include non-technical work that shows you can be relied on. Two years in a call centre is evidence of dealing with people under pressure and following process, and a hiring manager for an analyst role reads it as such. Do not hide it; frame it.

The one-line tailoring rule

Do not rewrite your CV for every application; you will stop applying. Change two things: the positioning line at the top, and the order of your bullets so the most relevant sits first. That is fifteen minutes, not two hours, and it captures most of the benefit.

The limitation worth knowing

A good CV changes your outcome at the margin — perhaps a few percentage points of response rate. The *route* changes it by an order of magnitude: a referred application is read by a human within a day, and a cold one joins several hundred others. Spend an evening making the CV good, then spend the rest of your evenings on the routes in the other lessons of this module. People polish CVs for months because it is the only part of the search that feels controllable.

BEFORE YOU MOVE ON

Kwame has no paid experience in data. His CV opens with a three-sentence career objective, then education, then a section titled "Personal projects" listing four projects by name and technology. Which single change is likely to help most?

A. Convert the CV to a designed template with a sidebar and icons, so it stands out visually.

B. Add every relevant tool from the job advert to a skills section to improve keyword matching.

C. Replace the objective with an eight-word positioning line and links, move the work to the top under a heading like "Selected work", and give each project three lines saying who uses it, what was hard, and one number.

D. Remove the projects section entirely, since unpaid work is discounted by employers anyway.

35 · 8 min

The profile a stranger checks in ninety seconds

THE ONE THING TO KEEP

Three public surfaces each answer one question in ninety seconds, and a reviewer who compares them treats inconsistencies as reasons to stop reading.

Three surfaces, one job each

Before anybody speaks to you, someone opens two or three tabs. They are not reading. They are answering one question: *is this person plausibly the thing the advert describes?*

- **LinkedIn** answers "who is this and are they real".
- **GitHub** answers "do they actually build things".

- **One page of your own** answers "what have they done and what happened".

Each has a different job and none of them needs to be impressive. They need to be legible in ninety seconds.

LinkedIn: the top third is the whole thing

On a phone, a recruiter sees your photo, your name, your headline, your location and about two lines of the About section. Everything below is read only if that top third worked.

- **Headline.** Not your job title, which is already shown. Use the shape *what you do · with what · for whom*: "Data analyst · SQL and Python · retail and logistics · Nairobi". A student or career-changer writes what they are doing, not what they lack: "Learning data engineering · building public pipelines with dbt and DuckDB" beats "Aspiring data professional seeking opportunities", which says only that you are unemployed.
- **About.** Five sentences, first person, plain. What you do, one thing you have built, what you are looking for. No third-person biography, no adjectives about passion.
- **Open to work.** The recruiter-visible setting costs nothing. The green banner is a separate choice; it is visible to everyone including your current employer, and people have been caught out by that.
- **Location.** Set it to where you actually are. An inaccurate location wastes everybody's time and is discovered in the first minute of the first call.

On photographs, markets differ, and the difference is legal rather than aesthetic. In the United States and United Kingdom, CVs conventionally carry no photograph because employers actively avoid material that could support a discrimination claim. In much of continental Europe, South Asia and Latin America a photograph is normal. On LinkedIn, have one everywhere; on a CV, follow the local convention.

GitHub: the pinned six

The profile page shows six pinned repositories and a README you control. That README is the highest-leverage paragraph you own on the internet and almost nobody writes one. Four lines: what you work on, three links to your best repositories with one sentence each, and how to contact you.

What a reviewer actually checks, in order: does the top repository have a README that says what it does and how to run it; does the code run; are there commits over several days rather than one enormous dump; is there a test file anywhere.

The contribution graph is largely folklore. Nobody serious rejects a candidate for pale squares, and a wall of green from automated commits is transparent and slightly embarrassing. Three finished, documented projects beat four hundred days of streak.

Your own page

One page, no framework, no build step. A heading, two sentences about you, three projects with a paragraph each ending in what happened, and a contact line. GitHub Pages, Cloudflare Pages and Netlify all host it free on a custom domain; a domain costs around ten dollars a year and is optional.

The mistake is building a portfolio *site* instead of writing up the *work*. A beautiful site with three unexplained repositories is worth less than an ugly page with three honest write-ups, because the reviewer is buying judgement and only one of those shows any.

The consistency check nobody mentions

A recruiter with two tabs open compares them. Dates that disagree between CV and LinkedIn, a title that is grander in one place, a project that appears on the site and nowhere else — each is a small question mark, and three of them is a reason to move on. Make the three surfaces say the same thing, and put the most detail in the one you control.

The honest limitation

Returns diminish steeply. An afternoon on these three surfaces is genuinely worth it; a second week is procrastination wearing a productive hat. Profiles cannot make a case that the underlying work does not support, and the only reliable way to improve them further is to go and do something worth writing about.

BEFORE YOU MOVE ON

A career-changer's LinkedIn headline reads "Aspiring Data Scientist | Passionate about AI | Seeking Opportunities". What is the strongest reason to change it?

A. It is too long for mobile and will be truncated before the important words.

B. Recruiters filter on keywords, so it should list every tool they have used.

C. It states an ambition and a feeling rather than an activity, so a reader learns only that the person is unemployed — where naming what they are currently building would give the same reader something to judge.

D. "Aspiring" signals low confidence, and a bolder title such as "Data Scientist" would attract more recruiters.

---

36 · 8 min

Writing to strangers, and what to say

THE ONE THING TO KEEP

A short, specific message about their problem, sent to the person who does the work, outperforms fifty applications.

Why this works when applying does not

An application joins a pile. A message arrives in an inbox. The person doing the job is rarely contacted, has the problem you would be solving, and — cru-

cially — a hiring manager's internal recommendation moves through a company faster than any application ever will.

This is the highest-yield and worst-feeling activity in a job search. Most people do not do it, which is precisely why it works.

Who to write to

The person doing the work: a data engineer, an analyst, the engineering manager. Not the recruiter, who receives hundreds of these. Not the chief executive, unless the company has nine people.

Find them by looking at who speaks about the company's technical work — conference talks, blog posts, GitHub contributions, LinkedIn posts, a podcast, an answer on a forum.

The four-sentence message

```
Hi Ananya — I read your write-up on moving the reporting
stack off nightly CSVs. I built something much smaller with
the same problem: a delay tracker for a transport
cooperative, where three months of GPS timestamps were in
local time and eight in UTC with nothing labelling which
(https://…).

One question: when you backfilled, did you reprocess the
old rows or keep both clocks and correct at read time? I
went with reprocessing and I am not sure it was right.

No ask — I am moving into data engineering and reading
people who have done it.
```

What that message does: proves you read her actual work, shows your own work in one clause, asks a question only somebody who has tried the problem could ask, and removes the pressure of a request.

Reply rates on messages like this run somewhere in the region of one in five to one in three. Templated messages run near zero, and everybody can tell the difference instantly.

One message that gets a reply, and six that do not

What the four sentences do	What kills a message
• Prove you read their actual work, by naming it	• "I hope this finds you well"
• Show your own work in one clause, with a link	• A CV nobody asked for
• Ask a question only somebody who has tried the problem could ask	• "Any opportunities in your team?" as the first contact
• Remove the pressure of a request	• Flattery with no specifics
	• Four hundred words about your journey
	• Asking for a call before giving anything

Specific messages to the person who does the work run somewhere between one in five and one in three. Templated ones run near zero, and everybody can tell the difference instantly.

What kills a message

- "I hope this email finds you well."
- A CV attached that nobody asked for.
- "Any opportunities in your team?" as the first contact.
- Flattery with no specifics — "I admire your work" without naming any of it.
- Four hundred words about your journey.
- Asking for a call before you have given anything.

Volume, honestly

Fifteen to twenty-five well-targeted messages a month is the right scale. Not two hundred — at that volume you cannot be specific, and non-specific is the same as not sending. Two or three real conversations a month, sustained for six months, is a network.

The channels, and their real constraints

LinkedIn. A connection request carries a note with a hard 300-character limit on the free tier, which is roughly the four sentences above compressed. That constraint is useful discipline. Premium raises message limits and does not raise reply rates; the message does that.

Email. Company address formats are guessable and usually consistent — `first.last@`, `first@`, `finitial.last@`. Send one; a bounce costs nothing.

Communities. Discord and Slack groups for your language, city or tool; local meetup groups; the forum where people actually answer questions. Here the sequence is reversed: be useful in public for three months first, then a message is from someone they recognise.

GitHub. If you have fixed a bug or improved documentation in a project they maintain, you already have a conversation.

The follow-up rule

Once, after seven to ten days, adding something new — not "just bumping this". If there is still no reply, stop. Silence is usually workload, not judgement, and a third message converts neutral into negative.

The slow version that compounds

Answer questions in public for three months. In the community for your city or your tool, help people with the thing you learned last month. You are qualified to help anyone one step behind you, which is a much larger group than you think.

What this produces is not followers. It is fifteen to thirty people who have watched you be useful, and who will forward you a job when one appears — which is what "network" actually means, stripped of its unpleasant connotations.

The free path

LinkedIn free, community servers, and a guess at an email format cost nothing. Hunter and similar services have free tiers of a handful of lookups a month. The paid products — LinkedIn Premium, Sales Navigator, Apollo — are built for salespeople sending thousands of messages, which is the opposite of this method.

Where this does not work

In markets and employers where hiring is centralised and procedural — large IT services firms with campus intakes, public sector recruitment, graduate schemes — the process genuinely is the process, and a direct message will be politely redirected to the portal. Match the method to the employer category. Outreach is for startups, mid-size companies, agencies and remote roles; the application form is for institutions.

BEFORE YOU MOVE ON

Meera sends 40 messages a week to people at companies she wants to join. Each opens by praising the company, summarises her background, and asks whether they have openings. She gets almost no replies. What is the correct diagnosis?

A. Her background is not strong enough yet, so she should build more projects before reaching out.

B. At forty a week she cannot be specific, and a message that praises the company, recaps her CV and asks for openings gives the reader nothing to reply to — twenty specific messages a month about the recipient's own work would do better.

C. She is contacting the wrong channel and should apply through job boards instead.

D. She should follow up more persistently, since most people miss the first message.

37 · 9 min

Publishing, and the inbound it creates

THE ONE THING TO KEEP

Published writing works because it is found by strangers months later, which rewards narrow, specific answers to real problems over broad guides written from a week's knowledge.

The asymmetry

Writing to strangers is linear: one message, one chance. Publishing is not. A page you wrote in March is read in September by somebody you have never heard of who is hiring, and you did no work in September.

That is the entire mechanism, and it is worth being precise about it rather than mystical. Two things bring the reader: a search engine or a model answering a question you happened to have written the answer to, and a person forwarding your link to somebody else. Both reward the same thing — writing that is specific enough to be the best available answer to a narrow question.

What to write, in order of return

1. The thing that just took you two days. You spent nine hours on a Docker networking error, an obscure `dbt` incremental bug, a Hindi tokenisation problem. Write it up the following morning while you still remember the wrong turns. This is the highest-value writing on the internet and almost nobody does it, because once you have solved it, it feels obvious. It is not obvious. It was not obvious to you on Monday.

2. A comparison with numbers. "I ran the same 200-row extraction task through four approaches; here is the accuracy, the cost per thousand documents and the latency." Anybody can have an opinion. Almost nobody has run the test.

3. A decision record. The case-study format from the evidence module, published. What you were trying to do, what you rejected, what the number was.

4. Notes on something you are learning. Weakest, but not worthless, provided you write as a learner rather than as an authority. "Three things that confused me about embeddings, and what finally made them click" is honest and useful. "The Ultimate Guide to Embeddings", written by somebody who learned them last week, is neither.

The shape that gets read

- **Title it as the question somebody would type.** "Why my dbt incremental model silently dropped rows after a schema change" is a title. "Data Engineering Lessons" is not.
- **Answer in the first paragraph.** Readers who need the answer will take it and go. That is a success, not a theft; they will remember where they got it.
- **Show the error text, the command and the version numbers.** These are what search finds and what makes the page credible.
- **Say what you are unsure about.** A sentence admitting a limit does more for a reader's trust than three paragraphs of confidence.
- **Length is whatever the thing needs.** Six hundred honest words beat three thousand padded ones, and the padding is visible.

Where, and at what cost

Your own site first, because links you own accrue to you and a platform can disappear or paywall you. Free hosting on GitHub Pages, Cloudflare Pages or Netlify; free static generators — Hugo, Eleventy, Quarto, or plain HTML. Then cross-post where the readers are, with a link back: Dev.to, Medium, Hashnode, LinkedIn, Reddit's topic communities, Hacker News occasionally. On LinkedIn, paste the text into the post itself rather than posting a bare link, because the feed suppresses outbound links and a post nobody sees helps nobody.

The honest limitation, stated plainly

Most posts are read by about forty people. Many are read by four. The compounding is real but slow and unreliable: a fair expectation is six to twelve months of monthly writing before anything traceable happens, and a substantial number of people write for a year and get nothing but the writing itself.

That should change what you optimise for. Write things that are useful to *you* in six months — your own notes, findable — and treat the inbound as the second-order effect it is. People who write for the audience quit at post four. People who write to remember things keep going, and the audience arrives later.

One rule with teeth

Never publish your employer's data, code, incident details or customer names, and be careful with anything under an agreement. Write about the method, the tool, the general problem. This is not a formality: it is the one way that public writing can cost you a career rather than build one.

BEFORE YOU MOVE ON

Which of these posts is most likely to bring a stranger to a writer's profile six months after publication?

A. "A Complete Introduction to Large Language Models", written after finishing a course on the subject.

B. "Why my dbt incremental model silently dropped 12,000 rows after an upstream schema change", including the error output and the fix.

C. "Ten lessons from my first year learning data engineering", a reflective personal essay.

D. "The future of AI jobs in 2027", an opinion piece designed to be shared widely.

38 · 9 min

A merged pull request as evidence

THE ONE THING TO KEEP

A merged pull request proves you can work inside someone else's codebase and take review, and the reproduction of a bug is often a more welcome contribution than the fix.

What it actually proves

A portfolio project proves you can build something alone, on your own terms, with no one disagreeing. A merged pull request into somebody else's repository proves four different things, and they are the things employers are actually unsure about:

- You can read a codebase you did not write.
- You can follow conventions you did not choose.
- You can take review comments without defending yourself.
- Somebody with no obligation to you judged your work good enough to keep.

That last one is the only external validation available to a person with no job history. It is worth a great deal more than a certificate and it costs nothing but weeks.

The route that works, and the one that does not

The common advice — filter a famous repository by "good first issue" — mostly fails. Those issues are claimed within hours by hundreds of people doing the same thing, and maintainers of large projects are drowning.

The route that works starts from use:

1. **Use a mid-sized tool regularly.** Something with fifty to five thousand stars, an active maintainer and real users. The libraries around your own stack: a dbt package, a data-quality tool, a small Python client for an API you use.
2. **Hit a real problem.** You will, within weeks. The documentation is wrong, an error message is unhelpful, a function fails on a valid input, the installation instructions skip a step on Linux.
3. **Reproduce it minimally.** Cut it to the smallest script that shows the behaviour, with versions and the exact error. **This is the contribution.** Maintainers are short of time, not ideas, and a clean reproduction is often more valuable than the fix.
4. **Open an issue with that reproduction**, and ask whether a pull request would be welcome and how they would prefer it solved. Asking first prevents the most common heartbreak, which is a week of work rejected on design grounds.
5. **Send a small pull request.** One change. Tests if the project has them. The project's own formatting, not yours.

Documentation counts. A pull request fixing an install guide that is wrong on Ubuntu is a genuine contribution, is usually merged fast, and is a real name in a real repository.

Etiquette, learned the cheap way

- **Do not reformat files.** A diff of 400 lines where 3 are the fix will not be read.
- **Do not refactor on the way past.** It reads as a stranger rearranging someone's kitchen.
- **Match the surrounding style**, even where you disagree with it.
- **Reply to review comments within a day or two.** Abandoned pull requests are the most common outcome and the most annoying one for maintainers.
- **Accept a "no" gracefully.** Every maintainer remembers who argued, and it is a small world.

- **Read `CONTRIBUTING.md` first.** Roughly half of contributors do not, and it shows immediately.

Using it in a search

One merged pull request into a tool the interviewer has heard of is worth mentioning in the first minute of a conversation. Better still, describe the review: "the maintainer pushed back on my approach because it broke Windows paths, and the second version handled both." That single sentence demonstrates receiving criticism, which is the trait most interview processes are secretly probing for and the hardest to fake.

The honest limitations

Three, all real.

It is slow and outside your control. Maintainers are volunteers with jobs. A pull request can sit for three months and then be closed with "we are going a different direction". This is not personal and it is not rare.

Quantity is visible and worthless. The annual contribution events produced a flood of trivial pull requests, and maintainers now treat one-line typo fixes from strangers with weariness. Two substantive contributions to one project you actually use beat forty scattered ones, and reviewers can tell the difference instantly.

It does not convert directly into jobs. Almost nobody is hired because of a pull request. It works by making you credible in a conversation you obtained some other way, and by teaching you how professional code review feels before your first day rather than during it.

BEFORE YOU MOVE ON

A learner wants a first open-source contribution. Which approach has the best chance of ending in a merge?

A. Filter a well-known machine learning framework by "good first issue" and claim the first unassigned one.
B. Fork a popular library, refactor a module they find untidy, and submit it as an improvement.
C. Notice that a tool they use daily fails on a valid input, cut it to a minimal reproduction with versions, open an issue, and ask how the maintainer would prefer it fixed before writing code.
D. Submit typo corrections to twenty repositories to build a visible contribution history quickly.

39 · 8 min

The job next to the data

THE ONE THING TO KEEP

Take the job beside the data and become the person who answers with numbers, because an internal move skips the entire screening machine.

The most reliable route, and the least discussed

Large numbers of working analysts and data engineers did not get hired as analysts or data engineers. They were already inside a company, doing something adjacent, and they moved.

This route almost never appears in career advice because it is slow, unglamorous, and impossible to sell a course about. It is also the highest-probability path available to somebody with no relevant experience.

The bridge roles

Any job that sits near data and near a decision:

- Customer support and success — every ticket is a data point, and support has the clearest view of what is broken.
- Operations, dispatch, scheduling, warehouse coordination.
- Quality assurance and testing.
- Finance and back office, accounts receivable, reconciliation.
- Sales operations and CRM administration.
- Field data collection, survey work, clinical or laboratory administration.
- Business process outsourcing roles of any kind.
- Teaching and training, in an organisation with student data.

What they share: you touch the raw material, you learn the domain, and you find out which numbers people argue about.

Why the internal move is so much easier

Three mechanisms, all of them structural rather than about you.

You are a known quantity. The manager has watched you work for a year. Hiring you carries almost no risk, compared with an outsider who interviews well.

Many internal moves are never posted. A team that needs a person and knows one asks for them. There is no queue, no screening software and no comparison against three hundred applicants.

You already have the expensive part. Domain context — which table anyone trusts, why the Mexico numbers are late, what "active customer" means in this company — takes an outsider a year to acquire and cannot be bought. You have it for free.

The twelve-month play

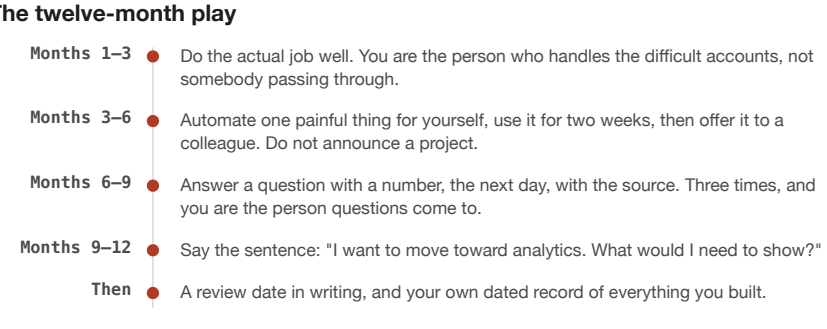
Months 1–3: do the actual job well. Nothing else works if this is missing, and everything else follows from the credibility it buys. You are not passing through; you are the person who handles the difficult accounts.

Months 3–6: automate one painful thing, for yourself, then share it. The weekly report someone assembles by hand. The rota that is copied between four sheets. Build it small, use it yourself for two weeks so it is proven, then offer it to a colleague. Do not announce a project.

Months 6–9: answer a question with a number. Someone in a meeting says "I think refunds are worse on the Lagos route". Come back the next day with the actual figure and where it came from. Do this three times and you become the person people bring questions to, which is the whole game.

Months 9–12: say the sentence. To your manager, directly: *"I want to move toward analytics. What would I need to show, and is there a project with the data team I could take?"*

This is the step people skip, and it is the one that works. Managers are not telepathic and generally cannot promote an ambition they have not been told about. Most of them will help, because a motivated internal person is cheaper than a hire.



The last step is the one people skip. Managers cannot promote an ambition they have not been told about, and a motivated internal person is cheaper than a hire.

What you can actually build from inside

Usually you cannot install anything, and there is no budget. The free tools that survive that constraint are more powerful than people expect:

- **Google Apps Script** — free with a work Google account, runs on a schedule, reads and writes Sheets, Gmail and Drive, and calls APIs. In a Workspace company this is the single highest-leverage skill available to someone in a bridge role.
- **Excel Power Query** — included in Excel, no admin rights needed, and it is a genuine ETL tool that most Excel users have never opened.
- **SQL access, if anyone will grant it.** Ask. Read-only access to a reporting replica is a small request that is frequently granted and rarely made.

The paid equivalents — Alteryx, premium Power Automate, a licensed integration platform — do the same work with a procurement process attached.

The risk, and the guard

The failure mode is real: you become the unofficial spreadsheet person for ever, doing two jobs, with the same title and the same pay. Some organisations will happily let this run for years.

Guard against it in two ways. First, agree a review date in writing — an email after a conversation saying *"as discussed, we will look at this again in six months"* is enough. Second, keep your own written record of everything you built, with dates and outcomes, because it becomes your Selected work section whether you move internally or leave.

And be willing to conclude that this employer will not move you. A year of building real things on real data is a portfolio, subject to what you are permitted to publish. That is not a wasted year — it is the year that makes you hireable somewhere else.

BEFORE YOU MOVE ON

Fatima works in customer support at a logistics firm. She wants to become an analyst. She has spent eight months teaching herself SQL in the evenings and applying to analyst roles at other companies, with no interviews. What is the strongest next move?

A. Keep applying externally but add a certification, to get past the screening filters.
B. Leave support for a junior technical role at any company, even at lower pay, to get a technical title.
C. Build a large public portfolio project on an open dataset to prove her SQL is real.
D. Use the support data she already touches: automate one report for her own team, bring a real number to a meeting that changes a decision, and then tell her manager plainly that she wants to move toward analytics and ask what she would need to show.

40 · 9 min

The first client, and how not to lose money on them

THE ONE THING TO KEEP

The first paid job converts you from a learner into someone with a reference, so make it small, write the scope down, and pick work you have already done once.

What the first one is for

It is not for the money. A first paid job is for three things: a reference, the right to say "I have done this for a client", and the discovery that selling and scoping are separate skills from building.

Price it accordingly — small, fast, finishable — and do not attempt to make it your income.

Where the first three clients come from

In order of how quickly they can be reached:

1. **People who already know you.** A relative's business, a former manager, someone from your old job. This is not cheating; it is how nearly every freelance career begins.
2. **Small local businesses with a visible spreadsheet problem.** A clinic, a school, a distributor, a two-branch retailer. They have an inventory sheet somebody maintains by hand for six hours a week.
3. **A community you are part of.** People post work in the places where people help each other.
4. **A subcontract from an overloaded freelancer.** The most underrated route. Established freelancers turn away small jobs constantly. Ask three of them whether they have anything they do not want.

Global platforms, honestly

Upwork, Fiverr and their equivalents work, with a cold-start problem that is structural rather than about your skill. Ranking depends heavily on completed jobs and reviews, so with zero of both you are invisible. Add a platform fee of roughly ten per cent and, on some platforms, a charge for submitting proposals.

The consequence: your first platform job has to be small, cheap and delivered perfectly, because you are buying a review rather than earning a wage. After three or four good reviews the economics change completely. Going in expecting your fourth-month rate in month one is how people conclude the platforms are a scam.

Local clients pay less per hour in many markets but find you faster, pay sooner, speak your language, and give you references that mean something in your city.

What a good first job looks like

Five to fifteen hours. One deliverable, describable in a sentence. Work you have already done once in a personal project.

Take the three monthly sales exports, combine them, and produce a one-page summary of top products and slowest-moving stock, updated weekly. That is a good first job.

Build an AI chatbot for my business for two hundred dollars is not a job; it is an unbounded liability with a number attached.

The one-page scope

Write it, even for your cousin. Especially for your cousin.

```
Deliverable: a weekly one-page PDF report of top 20 products
by revenue and stock older than 90 days, generated from the
three exports you already produce.

Not included: changes to your POS system, historical data
before January, any web dashboard.

Revisions: two rounds of changes to layout or metrics.
Further changes quoted separately.

You provide: the three export files, and access to last
year's data, by 14 March.

Timeline: 10 working days from receiving the files.

Payment: 50% to start, 50% on delivery.
```

The line that saves you most often is *you provide, by this date*. The commonest way a first project fails is that the client never sends the data, three weeks pass, and you are somehow the one who is late. Put their obligation in writing with a date and the conversation is easy.

Getting paid

- **Take 30–50 per cent before you start.** A client who refuses is telling you something useful, cheaply.
- **Bill in milestones**, not one payment at the end.
- **Do not hand over final files, credentials or a deployment before the final payment clears.** Show it working on your machine; transfer on payment.
- **Charge something.** Free work sets your price at zero and is treated as such — free projects get deprioritised by the client, not by you.

For cross-border payments, Wise and Payoneer are the usual routes. Watch the exchange-rate spread rather than the headline fee; the spread is where the cost hides and it is not itemised. Price with a margin for currency movement on a two-month project.

Invoicing needs nothing paid: a template in a word processor is a valid invoice if it has your details, the client's, a number, a date, the amounts and how to pay. **Wave** and **Invoice Ninja** are free if you want software. QuickBooks and FreshBooks are the subscriptions people buy before they have three clients.

What freelancing will not teach you

It teaches scoping, selling, dealing with clients and finishing — all genuinely valuable and rarely taught. It does not teach you to work inside somebody else's codebase, review other people's changes, or operate a system with other engineers, which is a large part of what an employer tests.

If your goal is employment, do both, or at least know which half you are missing and be honest about it in interviews. "I have shipped for clients but I have never worked in a team codebase" is a sentence that earns respect. Pretending otherwise does not survive the first technical conversation.

BEFORE YOU MOVE ON

Your first paying client stops responding after you send the scope. Three weeks later they message asking why the report is not ready — they say they sent the files "ages ago", but nothing arrived. What in the scope document would have prevented the argument?

A. A "you provide" line naming exactly which files the client must send and by which date, so the timeline visibly depends on something they can check.

B. A larger deposit, so they had more financial reason to engage with the project.

C. A clause limiting the number of revisions, so scope could not expand during the delay.

D. A penalty clause for late delivery, defining what happens if the deadline is missed.

CASE STUDY · MODULE 4

Three hundred and forty-one applications

Ifeoma, in Lagos, eleven months into a search for a first data role, with a spreadsheet of every application she has sent.

The spreadsheet is the only reason this case has numbers in it. She had started it in month two, on the advice of someone in a Discord server, and it had seven columns.

After eleven months:

- Applications sent: **341**
- First conversations: **6**
- Technical stages: **2**
- Final stages: **0**
- Offers: **0**

Six conversations from three hundred and forty-one applications is a rate of one point eight per cent. In the same eleven months she had finished two courses, built three projects and learned dbt.

What the funnel said

The leak was at the very top, and everything she had spent the year improving sat below it. Her SQL was not the reason nobody was calling; nobody was getting far enough to find out about her SQL. A funnel that breaks before the first conversation is a statement about targeting and route, not about ability — and it was the one diagnosis she had spent eleven months not making, because improving her skills felt productive and changing her method felt like admitting the year had been wasted.

Run the arithmetic forwards and it is worse. To reach two offers she needed roughly seven finals, seventeen technicals and thirty-four first conversations. At one point eight per cent that is close to **nineteen hundred applications**. At a warm-approach rate of thirty per cent it is a hundred and thirteen approaches.

Those two numbers describe the same search. One of them is possible for a person with rent to pay and a life to lead; the other is not, and no amount of stamina closes the gap, because the gap is division rather than effort.

The part of the funnel she could not see from inside

The tracker also had a column she had stopped filling in: why she wanted each role. Reading twenty rows back, most of them said some version of *it was open*. That is a different failure from a leaking funnel and it does not show up in a conversion rate. A search that has drifted into applying to whatever exists produces applications that are visibly generic to the person reading them, which lowers the rate further, which produces more applications.

Lagos also has a specific shape she had not adjusted for. A large share of the good roles are remote contracts with foreign companies, which means her competition is global and her evidence has to be public — and the hiring channel that actually works there is community rather than job board.

The decision, and what it cost

The proposal from a friend in the same server was blunt: stop applying for six weeks. Spend the same eight hours a week writing to twenty named people who do the work, with something specific attached, and publishing one write-up a month.

The costs were real and none of them were about strategy.

Sending applications is visible effort. Her uncle asked every Sunday how many she had sent, and "none, I wrote four emails" is a difficult sentence in a house where money is tight. Approaches also fail silently and slowly: an application produces a rejection email, which is at least an event, while a message to a stranger produces nothing at all for a fortnight.

And she had a third option she kept not taking: a customer support role at a payments company, twenty per cent below what she wanted, sitting next to the transaction data. Taking it would have meant admitting, out loud and to her family, that eleven months of study had ended in a support job — while

being, on the evidence of how most working analysts actually got in, the single highest-probability route available to her.

STOP HERE

Before turning the page: what would you have done, and what would it have cost if you were wrong? Write it down. The point of the next section is to compare.

WHAT HAPPENED

She did not stop applying, which was probably correct — she split the week: five tailored applications, five direct approaches, one piece of public work, half an hour on the tracker. In six weeks that produced twenty-seven approaches and eight replies, three of which became calls. One reply came from an engineer at a fintech whose write-up on timezone handling she had read and asked a specific question about; that conversation produced a six-week contract cleaning a transactions dataset, at a rate she would have refused a year earlier. The contract produced the reference, and the reference changed how every subsequent conversation started. Her cold rate over the whole search stayed near two per cent. Her warm rate over those six weeks was twenty-nine per cent — the same person, the same skills, a different route.

Ifeoma spent eleven months improving her skills. Was that the wrong response to the numbers?

It was the wrong response to these particular numbers. Skill improvement is the right fix when a funnel leaks at the technical stage; hers leaked before any human conversation, which points at route, targeting or the top third of the CV. The skills were not wasted — she needed them for the contract she eventually got — but eleven months of them did not move the stage that was actually broken.

Why is a warm approach worth roughly twenty cold applications rather than a few?

Because the conversion rates differ by an order of magnitude at the first stage: cold applications convert at one to five per cent and often under one per cent for advertised remote roles, while a referral or a direct approach to the person doing the

work converts at twenty to fifty per cent. The gap is not effort, it is that an application joins a pile of several hundred and a message arrives in one person's inbox.

What did the six-week contract change that three hundred and forty-one applications had not?

It converted her from a candidate into somebody with a reference and a paid engagement to describe. After it, conversations started with what she had delivered for a client rather than with what she had studied, and she had one person with no obligation to her who would answer a call about her work. Paid work in an unplanned direction beats unpaid preparation in the planned one, for exactly this reason.

MODULE 4 · TEST

Check what stuck

6 questions, one right answer each. This one is for you, not for a record: answers and the reason behind each are at the back. If two questions from the same module go wrong, re-read that module before the exam rather than after.

1 After 200 applications you have had three first conversations and no technical stages. What does the funnel say to change?

A. The technical preparation, since two of three conversations went nowhere

B. The route and the targeting, because the leak is above any assessment of skill

C. The volume, because 200 is too few to draw any conclusion from

D. The interview answers, because the first fifteen minutes are clearly not landing

2 Reaching two offers takes roughly thirty-four first conversations. Why does that make cold volume the wrong lever?

A. Cold applications are rejected by tracking software before a person sees them.

B. Employers penalise candidates who apply to many roles at the same company.

C. At a 3% cold rate it needs over a thousand applications; at 30% warm it needs a hundred.

D. The quality of a tailored application falls sharply once you are sending more than five in a week.

3 Which rewrite of "Responsible for data cleaning and analysis for the sales team" is doing the work a CV bullet is meant to do?

A. Performed advanced data cleaning and analysis to support key sales stakeholders
B. Owned the sales data pipeline end to end using SQL, Python and Power BI, partnering with commercial stakeholders to deliver accurate reporting throughout the quarter
C. Rebuilt the weekly sales report as one scheduled SQL job — four hours to fifteen minutes, and removed a double-count that had inflated Q2 by eight per cent
D. Cleaned and analysed sales data daily, working closely with the wider commercial team

4 You messaged an engineer about her write-up ten days ago and heard nothing. What is the correct next move?

A. Send one follow-up that adds something new, and stop if there is still no reply.
B. Send a short bump so the message returns to the top of her inbox.
C. Message a second person at the same company instead, since she has effectively declined.
D. Connect on another channel, because a different medium often gets a response.

5 You have found a genuine bug in a mid-sized open-source tool you use weekly. What is the most useful first contribution?

A. A pull request with the fix and a refactor of the surrounding file
B. A minimal reproduction in an issue, with versions and the exact error
C. A comment on an existing similar issue offering to take it on
D. A pull request adding a test that fails, leaving the fix to the maintainer

6 Why is moving into a data role from inside a company so much easier than being hired into one from outside?

A. Internal candidates are usually paid less, so the hire is cheaper to approve.

B. Companies are legally required to advertise every open role internally before posting it publicly.

C. Internal moves skip the technical assessment stages entirely.

D. You are a known quantity, many moves are never posted, and you already hold the domain context.

MODULE 5

Using AI in your own search

Both sides of the hiring table now use models, and most advice about it is sold by people with something to sell — CV scores, auto-apply subscriptions, interview coaching. This module is the mechanics: what an applicant tracking system actually does, why volume applying stopped working, where a model genuinely saves you hours, where using one destroys the evidence you were trying to produce, and what you should never paste into somebody else's server.

> BY THE END OF THIS MODULE YOU CAN
>
> Use a model for the mechanical parts of a search — tailoring, research, rehearsal, gap analysis — while stating correctly what an applicant tracking system does and does not do, and keeping your own and an employer's confidential material out of a third party's logs.

41 · 9 min

What it genuinely helps with, and what it quietly ruins

THE ONE THING TO KEEP

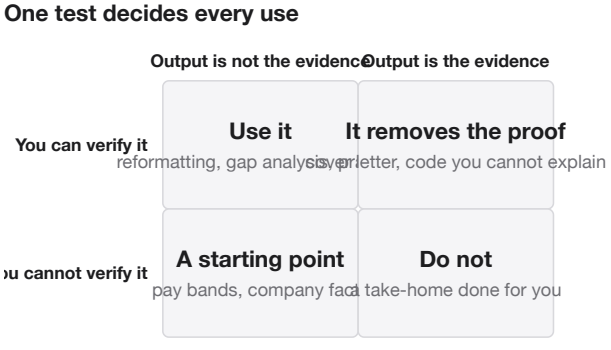
Use a model wherever you can verify the result and the result is not itself the evidence of your ability; the moment the output is the thing being judged, using one removes what you were trying to prove.

One test decides every use

Can you verify the output, and is the output itself the evidence?

Where you can check the result and the result is not what you are being judged on — reformatting, summarising, drafting a first version, generating practice questions — a model saves real hours. Where the output *is* the evidence of your ability, using one destroys the thing you were trying to produce, and everybody downstream finds out at the technical stage.

Everything below follows from that single test.



Where you can check the result and the result is not what you are being judged on, a model saves hours. Where the output is the evidence of your ability, using one destroys the thing you were trying to produce, and the technical stage finds out.

Where it saves hours

Vocabulary translation. The same job is called "Data Analyst" in one country, "MIS Executive" in another and "Business Intelligence Analyst" in a third. Paste three adverts and ask what terms they share that your CV does not use. This is genuinely useful, because you cannot see your own vocabulary.

Gap analysis against an advert. Paste the advert and your CV and ask: *"List each stated requirement and quote the line in my CV that evidences it. Mark the ones with nothing."* You will get a short honest list of holes, which is what you actually needed. Do not then ask it to fill them.

Decoding jargon. "Experience with medallion architecture and dbt semantic layers in a Databricks environment" is four things, and knowing which two you already do under different names changes whether you apply.

Research. Company, product, market, competitors, likely metrics — as a starting point to be checked, never as a fact.

Practice material. "Give me eight SQL questions on window functions at increasing difficulty, with answers" produces a decent exercise set in seconds, and you can check every answer by running it.

Rehearsal. Covered in its own lesson.

The first draft of an outreach message, which you then rewrite entirely, because the first draft's job is to stop the blank page, not to be sent.

Where it costs you

A cover letter written wholesale. Readers detect these at a rate that surprises people, and the tell is register rather than grammar: three-item lists, "I am particularly drawn to", "excited about the opportunity to leverage", a paragraph of enthusiasm containing no fact about the company. The specific damage is that a generated letter contains nothing only you could have written, which was the letter's only job.

A portfolio project you cannot explain. Covered in its own lesson. Briefly: the interview asks why, and you will not know.

Auto-apply services. They spend your money to worsen the ratio described in the volume lesson, and applying to roles you plainly do not fit is visible in the company's system next time.

Salary figures, hiring processes and company facts stated confidently. Models are fluent about pay bands and interview stages in your city and are frequently wrong, because that information is thin, local and changes yearly. Check against real postings and people.

"ATS score" tools. The next lesson explains why the number is invented.

The register problem, and the fix

Default model prose has a shape: balanced, enthusiastic, superlative, hedged, and structurally identical every time. It is exactly the register that a recruiter reading forty applications has learned to skip.

If you use a model in your writing at all, do the last pass yourself and cut anything that is true of any candidate. "I have a passion for using data to drive impact" survives being said by anybody. "I rebuilt our weekly stock report after finding the depot codes did not match between two systems" does not, and it is the only kind of sentence worth the paper.

A useful instruction when drafting: *"Write it in plain sentences. No adjectives about my enthusiasm. Every claim must be traceable to something in the notes I gave you."* Then read it and remove another third.

The honest limitation

None of this changes the funnel much. Tailoring and formatting move your conversion by a few percentage points at most; the route in moves it by a factor of ten. A model can make forty applications feel like productivity while the thing that would actually work — five people contacted properly — goes undone. Use it to buy time, then spend the time on the route.

BEFORE YOU MOVE ON

Which use of a model during a job search is most likely to backfire at a later stage of the process?

A. Asking it to list which requirements in an advert are not evidenced anywhere in your CV.
B. Asking it to generate eight practice SQL questions with answers before a technical screen.
C. Asking it to build the analysis and write-up for a portfolio project on an unfamiliar dataset, which you then publish under your name.
D. Asking it to summarise a company's product and recent announcements before an interview.

42 · 9 min

What an applicant tracking system actually does

THE ONE THING TO KEEP

An applicant tracking system parses, stores and filters — the automatic rejection happens on the form's knockout questions, not through a secret score applied to your CV.

A database with a workflow attached

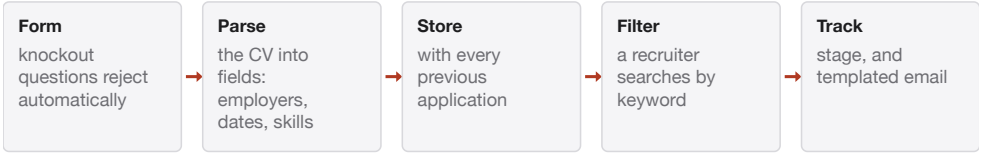
An applicant tracking system — Workday, Greenhouse, Lever, Ashby, Taleo, SuccessFactors, iCIMS, Zoho Recruit — is a database of applications with a workflow on top. It does four things:

1. **Parses your CV** into structured fields: name, contact, employers, dates, titles, education, a bag of skill keywords.

2. **Stores it**, permanently, along with every previous application you made to that company.
3. **Lets recruiters search and filter** — by keyword, by location, by answers to the form's questions.
4. **Tracks stage and sends templated email.**

That is the whole machine. It is unglamorous and it is not the villain of the story.

What an applicant tracking system does, in order



No mainstream system scores your CV against a threshold. The only automatic rejection is the form's knockout question; everything after it is a person choosing a filter, which is why the words matter and the percentage does not exist.

The score is a product, not a feature

You will be offered tools that "scan your CV like an ATS" and return a match percentage, with a subscription to improve it. No mainstream applicant tracking system assigns your CV a percentage and rejects you below a threshold. The number is invented by the company selling you the report.

What is real, and different in an important way:

- **Recruiters can sort and filter.** If a recruiter searches the pile for "dbt" and your CV never contains the word, you are not in the results. That is a human choosing a filter, not an automatic rejection — but the effect on you is the same, which is why the words matter.
- **Knockout questions are genuinely automatic.** The form asks whether you have the right to work in the country, whether you have five years of experience, whether you can be on site. Answer in the disqualifying direction and you are rejected without a human involved. This is where automatic rejection actually happens, and it happens on the form, not in your CV.

- **Some systems rank candidates** and surface a suggested order. Recruiters vary in how much they trust it, and regulators are increasingly interested in it — New York City requires bias audits of automated employment decision tools, and the EU AI Act treats recruitment systems as high risk. The rules here are still moving and differ by jurisdiction.

What parsing actually breaks on

Since the machine's real job is reading your document into fields, the practical advice is about being readable:

- **One column.** Two-column layouts frequently parse as interleaved nonsense — your job titles woven through your skills list.
- **No text in headers or footers.** Many parsers drop them entirely, which is how people submit a CV with no phone number.
- **No tables for layout**, no text inside images, no icons standing in for words. A skills chart of five-star ratings parses as nothing at all.
- **Standard headings.** "Experience", "Education", "Skills". Not "My Journey".
- **Dates in a consistent format**, ideally `Mar 2023 – Jun 2025`.
- **The file type the form asks for.** If it says PDF, send PDF; if it says .docx, send .docx.

You can check yours in thirty seconds, free, without any tool: open your PDF, select all, copy, and paste into a plain text editor. What you see is roughly what the parser sees. If your job titles are scattered and your contact details are missing, fix the layout — that test is worth more than any paid report.

The words, honestly

Use the advert's exact terms **where they are true of you**. If they say "ETL" and you wrote "data pipelines", write both once. If they say "stakeholder management" and you did it, use their phrase.

What does not work: keyword stuffing, white text on white background, or a hidden block of terms. These get you rejected by the human two minutes later,

and some systems flag them. The invisible-text trick in particular is recirculated every year and it has never worked.

The two things that matter more than any of this

Apply through the company's own careers page where one exists. Applications through aggregators sometimes arrive as an email attachment nobody opens, or never arrive at all.

Get a human to look. A referral, an email to the hiring manager, a comment from someone internal — any of these routes around the pile entirely, and that is the actual answer to "my CV is not getting through".

The honest limitation

The formatting advice above is worth doing once, in an hour, and then never thinking about again. Parsing is not what is stopping you. Nine hundred other applications are. Treat the ATS as a door you should not accidentally lock rather than as the reason it will not open.

BEFORE YOU MOVE ON

An applicant's CV uses a two-column layout with their contact details in the page header and a five-star skills chart. Why is this a problem?

A. The ATS will assign it a low match score and reject it before a human reads it.

B. The parser is likely to interleave the two columns, drop the header entirely and read nothing from the chart, so the stored record may lack contact details and have garbled job history — and recruiter searches run against that record, not the pretty document.

C. Recruiters find visual CVs unprofessional and discard them on sight.

D. Skills charts are dishonest because star ratings are unverifiable.

43 · 8 min

Rewriting a CV against an advert, and the line

THE ONE THING TO KEEP

Tailoring is translating true things into the employer's vocabulary, and the reason not to cross into invention is mechanical: titles and dates get verified and tools get tested.

Why tailoring works at all

Two companies with the same job describe it differently. One says "ETL", one says "data pipelines"; one says "stakeholder management", one says "working with the business". A reader scanning for their own words does not find yours, and concludes you have not done the thing you have done for three years.

Tailoring is translation, and translation is honest. Everything in this lesson turns on the difference between saying a true thing in their vocabulary and claiming a thing that is not true.

A twenty-minute workflow

1. Extract the requirements. Paste the advert into a model and ask for each stated requirement as a separate line, with must-have and nice-to-have marked. Adverts bury this in prose and the list is clarifying.

2. Ask for a gap analysis against your CV. *"For each requirement, quote the line in my CV that evidences it, or say NOTHING FOUND."* You now have three piles: evidenced, evidenced-but-in-different-words, and genuinely missing.

3. Fix pile two yourself. This is the actual tailoring, and it is usually five or six lines. "Built and maintained nightly data pipelines" becomes "Built and maintained nightly ETL pipelines (data pipelines)" if both terms appear in their advert. One word, true, findable.

4. Rewrite the top third. The summary at the top of your CV and the first two bullets of your most recent role are the only parts most readers reach. Those get rewritten per application. The rest stays fixed. This is the eighty per cent of the value for twenty per cent of the work.

5. Decide about pile three. Missing requirements are a decision, not a writing problem. Missing one or two of eight is normal and you apply. Missing the central ones means the advert is describing somebody else and your hour is better spent elsewhere.

Do not let the model write the final text. It knows nothing about what you did, so anything it adds is either recycled from your own words or invented, and the second kind is what ends interviews.

The line, stated concretely

On the honest side: reordering bullets so the relevant work is first; using their term for a thing you did; adding a project you had left off; quantifying a result you genuinely produced; describing a team's outcome and naming your part in it.

On the other side: listing a tool you have not used; changing a job title to one you were not given; moving dates to hide a gap; claiming a team result as your own work; converting six weeks of a course into "two years of experience".

The reason to stay on the honest side is not only ethical, and it is worth being specific about the mechanism, because "you might get caught" is vague. You get caught in three predictable places. The **technical screen**, where twenty minutes of questions on the tool you listed goes badly in the first three. The **project conversation**, where "walk me through a hard decision on that" has no answer if you did not make it. And the **reference or background check**, where titles and dates are the two things that get verified, and a discrepancy is treated as dishonesty rather than as an error — offers are withdrawn for this, routinely, after acceptance.

There is a fourth cost people underestimate: an exaggerated CV wins you interviews for jobs you cannot do, and those interviews are miserable and consume the weeks you needed for the jobs you can do.

Gaps and career changes

You do not have to hide either. A gap gets one line — "2024–2025: caring for a family member; part-time study in data engineering" — and stops being a mystery. Interviewers ask about gaps because the silence invites the question, not because a gap disqualifies anybody.

For a career change, the top of the CV should say what you are now, with the previous career reframed as domain knowledge rather than apologised for. A nurse moving into health data has something that no computer science graduate has, and burying it is the mistake.

The honest limitation

Tailoring moves your response rate a few points. It is worth twenty minutes and not worth two hours, and it cannot rescue an application to a role you are three levels away from. If your funnel shows no first conversations, the fix is the route in, not another rewrite.

BEFORE YOU MOVE ON

An applicant has used Postgres and MySQL, and the advert asks for Snowflake, which they have never touched. What is the defensible way to handle this?

A. Add Snowflake to the skills list, since the SQL is broadly similar and the difference can be learned in the notice period.

B. Leave the CV unchanged and hope the reviewer infers transferability.

C. Write "SQL warehousing on Postgres and MySQL; no Snowflake yet" and, if it matters enough, spend a weekend on Snowflake's free trial so the sentence can change honestly next month.

D. List it as "Snowflake (familiar)", which signals awareness without claiming expertise.

44 · 8 min

Why volume stopped working, on both sides

THE ONE THING TO KEEP

Generated applications made volume cheap, employers responded by raising the cost of being considered, and the result is that a warm approach is now worth roughly twenty cold applications rather than seven.

An arms race with a predictable end

Before 2023, a tailored application cost an hour, so people sent few and employers received few. When drafting one dropped to about thirty seconds, applications per posting rose sharply — LinkedIn and several large job platforms have reported application volume growing far faster than the number of jobs, and single postings for ordinary remote roles now routinely draw several hundred to several thousand candidates.

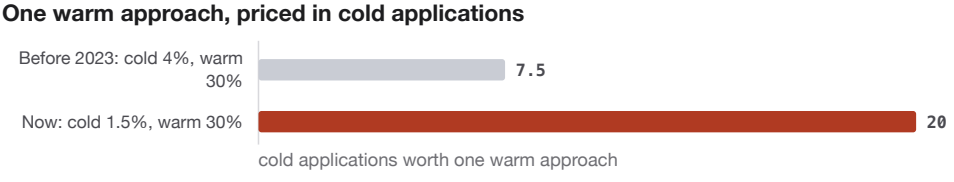
Employers responded exactly as you would expect if you had thought about it for a minute:

- **More knockout questions** on the form, which reject automatically.
- **More take-home exercises**, earlier, to impose a cost that a generated application cannot pay.
- **Heavier weighting on referrals**, because a vouched candidate skips the pile.
- **Postings closed early**, sometimes within 48 hours, once enough applications arrive.
- **Fewer replies**, because the arithmetic of replying to two thousand people does not work.

The net effect is that the cold conversion rate in the funnel lesson fell, for everybody, at once. Nothing about you changed.

The maths, since it is the whole argument

If cold conversion was 4% and is now 1.5%, and warm conversion held at 30%, then one warm approach is now worth twenty cold applications rather than seven. Every hour you spend automating volume is spent on the side of the ledger that got worse, while the other side got relatively better.



Cold conversion fell from about four per cent to about one and a half when generated applications made volume cheap; warm conversion held at thirty. Nothing about you changed, and every hour spent automating volume is spent on the side that got worse.

This is not a moral point about effort. It is division, and it is the reason this module sits after the one on getting seen rather than before it.

Auto-apply services, specifically

Products that fire hundreds of applications on your behalf are sold on exactly the wrong model of the problem. Three concrete costs:

- **They apply to roles you do not fit**, because they match on keywords. Your application history is stored in that company's tracking system, and a recruiter who sees six unsuitable applications from you in a year reads it as noise rather than enthusiasm.
- **Some systems detect and block them**, through submission-rate patterns or form telemetry.
- **They spend the hours you had.** The two hundred applications feel like work; the five people you did not contact are the search you did not run.

What replaces it

A weekly quota that a working person can actually hold:

- **Five real applications**, tailored in the twenty-minute way, to roles you can defend wanting.

- **Five direct approaches** — a named person, a specific reason, one piece of evidence attached.
- **One piece of public work** — a write-up, a merged pull request, an answer to somebody's question.
- **Thirty minutes updating the tracker** and reading what it says.

Eleven items a week. Roughly six hours. Sustained for three months, that is 60 applications, 60 approaches and 12 published artefacts, and on the conversion rates in the funnel lesson that produces enough conversations to reach offers. Two hundred auto-applications a week for the same three months produces a full inbox and no interviews.

Where volume genuinely does still work

Being honest requires naming the exceptions, because they are real and they matter for exactly the readers this course is written for.

- **Large services and outsourcing firms** hiring in batches. These processes are built for volume, run standardised aptitude tests, and applying widely is the correct strategy.
- **Graduate schemes and campus intakes**, which open in fixed windows with fixed processes.
- **Government and public-sector recruitment**, where the process is formal, scored against published criteria, and personal routes do not help.
- **Agency-driven contract markets**, where recruiters place people at speed and being in many agencies' databases is straightforwardly useful.

In those channels, apply to everything that matches. Outside them, volume is a way of feeling busy.

The honest limitation

Nobody knows where this settles. Employers are experimenting with in-person stages, live coding, work trials and stricter referral policies, and some of those are worse for candidates without networks — which is most people reading this. The reasonable response is to build the one thing that survives every

version of the process: a small number of people who have seen your work and will say so.

BEFORE YOU MOVE ON

Which candidate is using the current market correctly?

A. Amara, applying to a large IT services firm's batch intake and three government postings by their published processes, while also contacting two engineers at product companies she has researched.

B. Kwame, running an auto-apply subscription that submits 150 applications a week to every posting containing the word "data".

C. Ravi, refusing to apply to any advertised role and contacting only hiring managers directly, including at firms whose entire process is a standardised batch test.

D. Lena, spending three weeks perfecting one application to her single favourite company before sending anything else.

---

45 · 8 min

Your CV, their take-home, and the terms

THE ONE THING TO KEEP

Before pasting anything, establish whose data it is; for material that is not yours, a small model running locally removes the question rather than managing it.

Three different questions, usually collapsed into one

People ask "is it safe to use AI for this". That question has no answer until you split it:

1. **Is this data mine to share?**
2. **What will the provider do with it?**

3. **Would I mind if it appeared in someone else's output?**

Most of the trouble comes from skipping the first.

Not yours to share

- **An employer's take-home exercise** that the brief marks confidential. Some companies say so explicitly; assume it if the exercise is based on their real data or describes their internal problem.
- **Your current employer's code, data, documents, incidents or customer names.** Your contract almost certainly covers this, and pasting into a third-party service is disclosure.
- **Other people's personal data.** A referee's phone number, a colleague's name in a story you are polishing, a screenshot with an email address in it. Data protection law applies to you as an individual in several jurisdictions, and in all of them it applies to your employer.
- **Identity documents.** Passport, national ID, bank details. There is no job-search task that requires uploading these to a chatbot.

Your own CV is generally fine to paste. Strip your home address and phone number first — they are not needed for the task, and you have then removed the only sensitive parts.

What the provider does with it

Terms differ by provider and by tier, they change, and the last time you checked is not now. The stable shape, which has held across the major providers for a while:

- **Consumer free and low-cost tiers** may use your conversations to improve models, sometimes by default. There is normally a setting to turn this off, and it is normally not the default.
- **Business, enterprise and API tiers** typically do not train on customer content by contract. This is the difference that matters, and it is why an employer that has bought a licence usually wants you to use theirs rather than your personal account.

- **"We do not train on your data" is not "we do not keep it".** Retention for abuse monitoring is standard practice, commonly around 30 days, sometimes longer, and staff can review flagged content. That is reasonable operational practice and it is also a reason not to paste something you would not want a stranger to read.

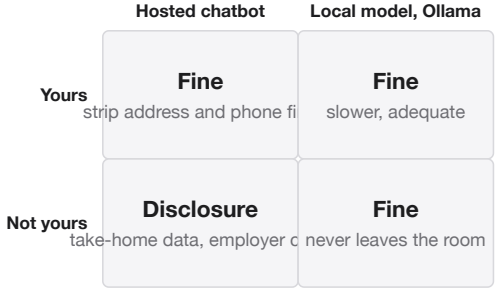
The practical instruction: open the settings on whichever tool you use, find the training or data-controls section, and set it deliberately. It takes two minutes and most people have never looked.

The free option that removes the question entirely

A small open-weight model running on your own machine sends nothing anywhere. **Ollama** and **llama.cpp** are free, install in minutes, and run usable models on an ordinary laptop with 8 to 16 GB of memory — no GPU required, though it is slower without one.

They are meaningfully weaker than the large hosted models at reasoning and long documents. For the tasks in this module — reformatting, summarising, drafting, rehearsing, and anything touching a confidential exercise — they are entirely adequate, and this is the honest answer to "how do I use AI on my employer's material". You do not. You use a model that never leaves the room.

Whose data, and where it runs



The first question is whose data it is; the second is where it goes. For material that is not yours, a small open-weight model on your own laptop sends nothing anywhere and removes the question rather than managing it.

Two rules for the take-home specifically

Read the brief's own instructions. An increasing number state their position on AI use directly: some forbid it, many now permit it and ask you to say what you used and why. Follow what it says, and if it says nothing, disclose briefly in the README. Nobody has ever been rejected for a sentence saying which parts were assisted; people are rejected for being unable to explain their own submission.

Never paste their proprietary data anywhere. If the exercise ships a real customer extract, work locally. This is one of the few places in a job search where a single decision can be a genuine breach rather than a bad look.

The honest limitation

Everything in this lesson is a snapshot of a moving target. Terms are rewritten, defaults change, regulators intervene, and a provider's policy in your country may differ from its policy elsewhere. Treat the principle as durable — *know whose data it is, and check the setting yourself* — and the specifics as something to verify each time it matters.

BEFORE YOU MOVE ON

A take-home exercise ships a CSV of the company's real transaction data and the brief says nothing about AI tools. What is the defensible approach?

A. Paste the data into a hosted assistant, since the brief does not forbid it and silence implies permission.

B. Use a hosted assistant but turn off model training first, which removes the risk.

C. Work on the data locally — with a local model if AI help is wanted — and use a hosted assistant only for general questions containing none of their data, noting in the README what was used.

D. Anonymise the identifiers and then paste it, since anonymised data is no longer personal data.

46 · 8 min

Forty minutes of research, and how to check it

THE ONE THING TO KEEP

Research is aimed at how the organisation makes money, where the role sits and what remains uncertain, and every fact a model gives you needs a source you actually open.

What you are actually trying to learn

Not trivia. Three things:

1. **How this organisation makes money**, because everything about the role follows from it.
2. **Where the role sits** — whose problem you would be solving, and what number would move if you did it well.
3. **What is uncertain** — the thing you would want to know before saying yes.

Forty minutes gets all three. Four hours gets you a fact about their 2019 funding round that impresses nobody.

The sequence

The product page and the pricing page (10 minutes). What do they sell, to whom, and what does it cost? Pricing tells you more than the About page: per-seat pricing means many small customers and churn matters; six-figure enterprise deals mean few customers, long sales cycles and a data team that supports sales.

All their open roles (5 minutes). This is the most underused source in a job search. Twelve engineering roles and no data roles tells you where you would sit. Three roles mentioning Snowflake and one mentioning Databricks

tells you a migration is happening. Hiring a Head of Data at the same time as your role means your manager may not exist yet.

Their engineering blog and GitHub organisation (10 minutes). The stack, the scale, and the problems they thought worth writing about. If there is nothing, that is also information.

Money and trajectory (5 minutes). Listed company: the annual report says the segments and the risks in their own words. Private: Crunchbase's free tier gives the funding stage and date. A company that raised in 2022 and not since is in a different mood from one that raised last quarter.

People (5 minutes). Your interviewers' backgrounds on LinkedIn — what they did before, how long they have been there. Long tenures across a team is a good sign; everyone arriving in the last eight months means something changed.

Reviews (5 minutes), read carefully. Glassdoor, AmbitionBox, Indeed. The selection is biased towards the delighted and the furious, so ignore the ratings and read only for repeated specifics. Four separate people mentioning the same manager or the same reorganisation is signal. Twenty people saying "great culture" is not.

Using a model without inheriting its errors

A model is good at compressing what you found and bad at knowing what is true about a private company in your city. It will produce a fluent paragraph about their funding, their customers and their technology stack, some of which will be invented, and none of which is labelled.

Two habits fix this:

- **Paste the source and ask about it**, rather than asking from memory. "Here is their careers page and their pricing page — what do these imply about the shape of the engineering team?" is a task it does well.
- **Demand sources for anything it asserts.** *"For each claim, give the URL. Mark anything you cannot source."* Then open two of them. When a

search tool is available, this converts most of the guessing into checkable links; when it is not, the honest output is a very short list.

Never repeat an unverified number in an interview. "I saw you raised a Series B last year" is a small, humiliating error when the round was a Series A and it was 2023.

What the research is for

Three questions you could not have asked otherwise. That is the whole return.

- "Your pricing suggests most revenue comes from a small number of large accounts — does the data team mostly serve sales and customer success, then?"
- "You are hiring a Head of Data alongside this role. Would I be reporting to that person once they start?"
- "Your engineering blog described moving off Redshift in 2024. Is that finished, or would I be joining mid-migration?"

Each shows you did the work, and — more usefully — each answer changes whether you want the job.

Two limits, one practical and one about manners

Research does not win interviews. It prevents the specific loss where you cannot say why you want this job rather than any job, and it stops you accepting something you would have refused with better information. That is worth forty minutes and not worth four hours.

And there is a line on people. Reading someone's professional background is normal preparation. Reading their personal social media, mentioning their marathon time, or naming their children is not research, it is surveillance, and it lands exactly as badly as you would expect. Stay on the professional side and stop there.

BEFORE YOU MOVE ON

A candidate is preparing for an interview at a private company. Which source most reliably reveals the shape of the team they would join?

A. The company's full list of currently open roles, read together.

B. A model's summary of the company, its stack and its recent history.

C. The overall star rating on Glassdoor or AmbitionBox.

D. The company's About page and mission statement.

47 · 9 min

Recorded and machine-scored interviews

THE ONE THING TO KEEP

A one-way interview is an elimination stage scored mostly from a transcript, so structure, audio quality and a first-sentence answer matter more than anything you would optimise in a live conversation.

Three different things wearing one name

The one-way video interview. You are sent a link, questions appear on screen, you record answers with a time limit and often one retake. HireVue, Spark Hire, Willo and others. There is no interviewer.

The chat screen. A bot asks knockout and basic screening questions in a messaging interface before a human is involved.

The live AI interviewer. Newer, less common, a synthetic voice conducting a conversation. Treat it as a one-way interview that interrupts.

All three are used for the same reason: a posting with 1,400 applicants and one recruiter. Knowing that is useful, because it tells you the stage's purpose is *elimination*, not selection.

What is actually scored, and what is no longer

Most systems transcribe your answers and score the text against a rubric of competencies — structure, relevance, whether you gave a specific example. Some also score pace and pauses.

Automated **facial analysis** was heavily marketed around 2018 to 2020 and has substantially retreated. HireVue publicly stopped using facial analysis in its assessments in 2021 following criticism and a complaint to the US Federal Trade Commission, and the research underpinning inferences from facial expression is contested. Some vendors still market variants of it. You cannot tell from the outside which you are facing.

The evidence base for the whole category is weaker than the marketing implies. Vendors publish validity studies; independent replication is thin. Treat confident claims in either direction with suspicion, including the claim that these systems are simply worthless — a structured, rubric-scored transcript is not obviously worse than a distracted human, and that is an uncomfortable thing to sit with.

The rules, which are real and moving

- **Illinois** requires notice, an explanation of how the AI works, consent, and deletion on request for AI-analysed video interviews.
- **New York City** requires bias audits of automated employment decision tools and notice to candidates.
- **The EU AI Act** classifies recruitment and selection systems as high risk, with obligations phasing in.
- Elsewhere, frequently nothing specific yet.

You are usually entitled to ask what is being assessed, whether a human reviews the result, and how long the recording is kept. Ask the recruiter by email. The answers are informative regardless of content.

How to do well in a recording

The reason people underperform is mechanical. Human conversation runs on feedback — a nod, a follow-up, a laugh — and with none of it, people speed up, wander and stop early. Compensate deliberately:

- **Announce the structure.** "Three things happened. First…" A rubric looking for structure now has it, and so does the human who skims the transcript.
- **Ninety seconds per answer.** Two minutes at most. Watch the timer once, then ignore it.
- **Answer the question in the first sentence**, then give the example. Do not build to a conclusion; there is nobody to hold.
- **Look at the lens, not at yourself.** Move your own preview window up near the camera if you can.
- **Use the practice question properly.** Almost every system offers one. Record it, watch it back, fix the audio and the framing before you start.
- **Sound quality beats picture quality.** Wired earphones with a microphone, a quiet room, and a hard surface behind you rather than a curtain. If the transcript is wrong, the score is wrong.
- **Plain background, decent front light.** A window in front of you, not behind.

Prepare four stories in advance — a conflict, a failure, a thing you shipped, a time you changed your mind — and expect to reuse them here.

Ask for what you need

If you have a speech difference, a disability, unreliable internet, or no quiet space, say so and ask for an alternative. In many jurisdictions employers have an obligation to consider reasonable adjustments, and a good number of companies will offer a live call instead without any obligation at all. The request is normal and is not held against you by anybody you would want to work for.

The honest limitation

These systems are documented to work less well for people with strong regional accents, atypical speech, poor bandwidth or noisy homes — which describes a substantial share of the people this course is written for. That is a genuine fairness problem and it is not yours to solve at 9pm the night before.

Two practical responses. Control what you can: audio, light, structure, timing. And weight the channel accordingly — a company whose entire first stage is an unexplained one-way recording has told you something about how it treats candidates, and that information is worth having early.

BEFORE YOU MOVE ON

A candidate has one retake per question in a recorded interview. They record a first answer that wanders for three minutes before reaching the point. What is the best use of the retake?

A. Re-record with the same content but faster, to fit inside the time limit.

B. Re-record answering the question in the first sentence, then naming a structure and giving one specific example, in about ninety seconds.

C. Keep the first take, since authenticity scores better than a rehearsed second attempt.

D. Re-record with more energy and stronger eye contact with the camera, since delivery is what these systems measure.

48 · 8 min

Rehearsing with a model that wants to please you

THE ONE THING TO KEEP

Interviews are lost in production rather than recognition, so rehearsal must be out loud, one question at a time, with the model instructed to press on vagueness rather than encourage.

Recognition is not production

You read an answer to "tell me about a time you disagreed with a colleague" and it makes sense. That feeling is recognition, and it is not the skill being tested. The skill is producing a clear answer, out loud, in ninety seconds, while nervous.

The gap between the two is where almost all interview failure lives, and it closes with production practice only. There is no version of this that happens in your head.

A rehearsal that is worth the hour

Give the model the advert and your CV, then:

> *"Conduct a 25-minute first-round screen for this role. Ask one question at a time and wait for my answer. Do not give feedback or hints during the interview. Ask at least two follow-up questions when an answer is vague. At the end, tell me the two weakest answers and exactly why."*

Three details matter and all three are usually skipped.

One question at a time. A model given a free hand produces a list of twelve questions, which you then read. That is recognition again.

No feedback during. Encouragement after every answer trains you for an interview that does not exist.

Follow-ups on vagueness. The follow-up is where real interviews are lost — "you said you improved the process, by how much?" — and it is the part you cannot rehearse alone.

Answer out loud, not in the chat box. Typing is a different skill and you will not be typing.

Making it stop flattering you

Models are agreeable by default, which is the single thing that makes them poor practice partners. Instruct against it explicitly:

> *"Now play a sceptical hiring manager who thinks I am too junior for this role. Give me the three objections you would raise, in the words you would use."*

> *"For each of my answers, name the follow-up question I would least like to be asked."*

> *"Which of my claims would you not believe without evidence, and what evidence would satisfy you?"*

The last one is the most useful prompt in this lesson. It surfaces the claims on your CV that are load-bearing and unevidenced, which is exactly what a good interviewer probes.

The free thing that beats every tool

Record yourself on your phone and watch it back. It is unpleasant, it takes fifteen minutes, and it is more useful than any coaching product sold to job seekers.

You will discover, in order: that you take forty seconds to reach the point; that you say "basically" eleven times; that you trail off at the end of answers instead of stopping; and that a story you thought was two minutes is five. All

four are fixable within a week, and none of them would have been visible from the inside.

A friend or a study partner is better still, because a person reacts and a recording does not. Peer practice costs nothing and both sides improve, since watching somebody else answer badly teaches you more about structure than answering well does.

Timing, concretely

- **Behavioural answer: 90 seconds.** Two minutes is the ceiling.
- **"Tell me about yourself": 60 to 90 seconds.** Present, past, why here. Nothing before university unless asked.
- **Technical explanation: 2 minutes**, then stop and ask whether they want more depth.

Time three answers with an actual timer once. Your internal estimate is wrong by roughly a factor of two, in the direction you would expect.

Two honest limitations

A model does not know the company's real bar. It invents plausible questions, not their questions. It cannot tell you whether your answer would pass at that firm, and if it says it can, it is guessing.

Over-rehearsal is visible. An answer delivered word-for-word sounds recited, and interviewers discount it. Rehearse the *structure* and the *facts* — the situation, the number, the decision, the outcome — and let the sentences be new each time. Aim to have said each story out loud four or five times, not twenty.

BEFORE YOU MOVE ON

Why does asking a model for "a list of likely interview questions for this role" do so little for a candidate's performance?

A. The questions it generates are usually inaccurate for the specific company.

B. It rehearses recognition — reading answers that seem sensible — while the interview tests production of a structured spoken answer under pressure, including the follow-up when an answer is vague.

C. Reading questions in advance makes answers sound rehearsed and interviewers discount them.

D. Interview questions are confidential, so any published list is out of date.

49 · 9 min

Building a project you can still defend

THE ONE THING TO KEEP

Type the parts a reviewer would ask "why" about, delegate the rest, and never move past generated code you cannot explain — because generated data code fails by being plausible rather than by breaking.

The interview that ends it

You submit a project. The interviewer opens it and says: *"Walk me through this function. Why a left join here rather than an inner one?"*

If a model wrote it and you accepted it, you will not know. There is no recovery from that moment, and it is not because anybody objects to the tool — it is because the entire purpose of the artefact was to show how you think, and it now shows how something else thinks.

The rule that survives contact

Anything you would be asked about in an interview, you type yourself.

That is a small set, and it is not the whole project. Concretely:

- **Type yourself:** the schema and the joins, the metric definitions, the evaluation, the decisions about missing data, anything you will describe as a judgement.
- **Delegate happily:** boilerplate, argument parsing, the matplotlib incantation for a second axis, a regular expression, a Dockerfile skeleton, converting a script into a module, writing the docstrings.
- **Delegate and then read carefully:** anything that touches correctness, because of the next section.

The dividing line is not difficulty. It is whether a reviewer would ask why.

Silently wrong is the real risk

Generated data code fails in a particular way: it runs, produces plausible output, and is wrong. Three examples that recur:

- **`df.groupby('region')` drops rows where `region` is null**, because `dropna=True` is the default. Your totals are quietly short and nothing errors.
- **A join that should be one-to-many is written as one-to-one**, so rows multiply and every sum is inflated. The code is fine; the arithmetic is not.
- **A deprecated argument** from a version two years old, which either warns or silently changes behaviour.

The defence is not vigilance, which fails. It is a check that runs: assert the row count after a join, assert that totals reconcile to a number you computed by hand on ten rows, print the null counts before and after. Twenty lines of assertions catch all three of the above and cost less time than finding one of them the hard way.

Learning while using it

There is a real tension here and it deserves a straight answer rather than a slogan.

Reading a correct solution feels like learning and mostly is not. What encodes a skill is the effort of producing something and being wrong. An assistant removes exactly that effort, which is why it can make you faster this month and no better in six.

The practical compromise that works:

1. **Attempt first, then compare.** Write your version, however clumsy, then ask for a critique. You keep the productive struggle and still get the correction.
2. **Never move on from code you do not understand.** When something appears that you cannot explain, stop and ask what each part does until you can. This is the single habit that separates people who learn with an assistant from people who do not.
3. **Keep a decisions file.** `DECISIONS.md`, one line per choice: what you did, what you rejected, why. Written as you go, it takes seconds; reconstructed later, it is fiction. It is also, conveniently, the raw material for the case-study page.
4. **Delete and rewrite one component from memory** at the end. If you cannot, you did not learn it, and now you know that before an interviewer does.

Disclosure

Say what you used. A line in the README — "Used an assistant for boilerplate and plotting; the schema design, join logic and evaluation are mine" — costs nothing, is increasingly expected, and pre-empts the suspicion.

Nobody competent will reject you for using the tools their team uses daily. They will reject you for not being able to explain your own submission, and those are different failures with different fixes.

The honest limitation

The evidence on assistants and learning is genuinely unsettled. Studies find real productivity gains on well-specified tasks and weaker or negative effects on retention and on debugging unfamiliar code, and the picture differs sharply between novices and experienced developers. Anybody telling you confidently that assistants either destroy learning or do not affect it is ahead of the evidence.

What is not in dispute is the interview: you will be asked to explain your work by somebody who is good at telling. Build so that you can.

BEFORE YOU MOVE ON

A learner's assistant-written pipeline runs cleanly and produces a regional sales report. Which check would most reliably catch the errors this kind of code typically contains?

A. Running a linter and type checker over the code to catch quality issues.

B. Asking the assistant to review its own output for bugs.

C. Asserting the row count before and after each join and reconciling the totals against ten rows computed by hand.

D. Comparing the output to last month's report to see whether the numbers look reasonable.

CASE STUDY · MODULE 5

The take-home that arrived with somebody else's customers in it

Marlon, in Manila, four days to complete a five-hour take-home that shipped with sixty thousand rows of a real client's transaction data.

The brief was ordinary: find the drivers of chargebacks in the attached extract, produce a short write-up, five hours. The zip file was not ordinary. It contained merchant names, partial card numbers, timestamps, and a free-text `complaint_note` column in which about nine hundred rows contained a customer's name and, in a few dozen, a phone number.

The brief said nothing at all about using AI.

The tempting ten minutes

Classifying nine hundred free-text notes into complaint categories is the kind of task a hosted model does well and a person does slowly. Pasting the column into a chat window would have taken ten minutes and produced a usable classification. Doing it by hand would take most of an evening he did not have, because he was doing this after a full shift.

He asked the three questions in order and the first one settled it. **Is this data mine to share?** No. It is not even the hiring company's to share carelessly; it is their client's. His own CV would have been fine to paste. This was not that.

The second and third questions — what the provider does with it, and whether he would mind seeing it in somebody else's output — did not need answering, because the first had already decided the matter. That order is the point: most of the trouble in this area comes from skipping straight to the provider's retention policy.

He checked the settings on his own account anyway, which took two minutes and which he had never done in a year of using it. The training toggle was on by default, as it usually is on a consumer tier. That would have mattered for his own CV. For somebody else's customer file it was beside the point — "we

do not train on your data" is not "we do not keep it", retention for abuse monitoring is standard, and the disclosure has already happened at the moment of the paste.

The decision

Paste it. Ten minutes, a better classification than he could produce by hand, and a submission that would almost certainly never be traced. The cost is that a disclosure of a client's customer data to a third-party service is a genuine breach rather than a bad look, and he would be starting a relationship with a company by doing to their client exactly what they would fire him for later.

Do it locally. A small open-weight model through Ollama on a laptop with sixteen gigabytes of memory. Free, private, and meaningfully worse — slower, weaker on long text, and an hour of setup out of a five-hour budget.

Skip the free-text column entirely and write in the README that he had left it out. Honest, fast, and a visibly smaller piece of work than the other candidates would submit.

He had four days, one of which was a Sunday, and a real chance that the extra hours would produce a worse submission than the shortcut.

The thing he could not resolve by choosing well

There was a fourth question underneath the other three, and it was about the company rather than about him. A take-home that ships a real client's identifiable data to unvetted candidates is a data-handling failure by the employer, whatever the candidate then does with it. He was being asked to be careful with material that had already been handled carelessly.

He decided that was worth raising, once, in the submission rather than in an email — a single sentence noting that the extract contained customer names and numbers, and describing how he had contained it. Raising it as an accusation before an offer exists costs a candidate the process; raising it as a description of what he did costs nothing and is checkable.

STOP HERE

Before turning the page: what would you have done, and what would it have cost if you were wrong? Write it down. The point of the next section is to compare.

WHAT HAPPENED

He ran a small model locally for a first pass over the complaint notes, hand-checked fifty of its classifications against his own reading and found it agreed on forty-one, reported that agreement rate in the write-up rather than presenting the categories as fact, and left the rest of the analysis in plain SQL and pandas. The README carried two extra lines: what he had spent (about five and a half hours, and what he cut), and one sentence saying that no client data had left his machine and which parts of the code were assisted. He was told at the technical round that the data-handling paragraph was what moved his submission to the top of the pile — not because the analysis was the best, but because the company had been arguing internally about that exact risk, and had added an explicit line about it to the brief the following month.

Why does the question 'is this data mine to share?' come before the question about the provider's terms?

Because if the data is not yours, no setting on the provider's side makes the disclosure permissible. Turning off training, using a business tier or trusting a thirty-day retention policy all address what happens to data you were entitled to send. Marlon was not entitled to send this, so the provider's terms were irrelevant to his decision.

The local model was weaker. How did he keep that from becoming a weaker submission?

By treating its output as a first pass rather than as a finding: he hand-checked fifty classifications, reported the agreement rate, and did not present the categories as established. That is the same discipline as checking a model judge against hand labels. A stated agreement rate on a sample is stronger evidence than a confident, unchecked classification of all nine hundred.

Should he have disclosed the assistance at all, given the brief was silent?

Yes, in one line. Disclosure costs nothing, pre-empts suspicion, and is increasingly expected; nobody competent rejects a candidate for using the tools their own team uses. What gets people rejected is being unable to explain their own submission. The line also let him state the more important fact — that the client's data never left his machine — which is what actually distinguished him.

MODULE 5 · TEST

Check what stuck

6 questions, one right answer each. This one is for you, not for a record: answers and the reason behind each are at the back. If two questions from the same module go wrong, re-read that module before the exam rather than after.

1 Where does automatic rejection actually happen in an applicant tracking system?

A. On the form's knockout questions, such as right to work or years of experience
B. On a match percentage the system computes from your CV against the advert
C. On a keyword threshold set by the recruiter when the requisition opens
D. On a duplicate check that discards a second application to the same company within a year

2 Which single test decides whether using a model on a job-search task is sensible?

A. Whether the provider's terms permit commercial use of the output
B. Whether the employer's brief explicitly allows AI assistance
C. Whether you can verify the output, and whether the output is itself the evidence
D. Whether the task is writing rather than code, since writing is checkable by reading

3 An exaggerated CV is risky for a mechanical reason rather than a moral one. Which pair of items is most reliably verified?

A. Your salary history and your reason for leaving each employer
B. Your job titles and your employment dates
C. Your listed tools and the size of the teams you worked in
D. Your university grades and your professional certifications

4 Cold conversion fell from about 4% to about 1.5% while warm conversion held near 30%. What follows for how you spend an hour?

A. One warm approach is now worth about twenty cold applications rather than seven.

B. Cold applications should be abandoned entirely in every channel.

C. Applications should be sent earlier in the day, since postings close within 48 hours.

D. Volume still works, provided each application is tailored in twenty minutes.

5 A take-home ships with a real customer extract. Which question settles whether you may paste it into a hosted model?

A. Whether the provider retains conversations, and for how long

B. Whether the brief forbids AI assistance anywhere in its instructions

C. Whether you would mind the content appearing in somebody else's output

D. Whether the data is yours to share in the first place

6 Why must interview rehearsal with a model be done out loud, one question at a time, with feedback withheld until the end?

A. Speaking aloud improves recall of prepared material more than reading does.

B. Reading a list of questions practises recognition; the skill being tested is production.

C. Models produce better questions when they are asked to generate one at a time.

D. Feedback given during the session biases the questions the model chooses to ask afterwards.

MODULE 6

The interview, from first screen to signed offer

Hiring processes in this field are five to seven stages, and each one tests something different. Knowing which stage you are in tells you what to say. This module walks the whole sequence — the take-home, the live exercise, the design conversation, the stories, the questions you ask, and the offer.

> BY THE END OF THIS MODULE YOU CAN
>
> Walk each stage of a data or AI hiring process knowing what it tests, and negotiate an offer without a competing one in hand.

---

50 · 7 min

The stages, and what each one is really testing

> THE ONE THING TO KEEP
>
> Every interview stage tests a different thing, and knowing which stage you are in tells you what to say.

The standard sequence

Most data and AI hiring processes are some subset of seven stages. Knowing what each is for stops you giving a round-four answer in round one.

1. Recruiter screen — 20 to 30 minutes. Tests three administrative facts: can you legally work there, are you in the salary band, and can you describe

your experience in plain language. It is not a technical conversation and trying to make it one is a common error. Give a two-minute version of your background, ask for the band, ask about the process.

2. Hiring manager screen — 30 to 45 minutes. Tests one thing: *would this person be useful on my team within a few months?* This is where relevance and communication decide everything, and where depth matters least. Talk about what you built and who used it.

3. Technical exercise — a take-home, or 45 to 60 minutes live. Tests whether you can actually do the work. Its own lessons follow.

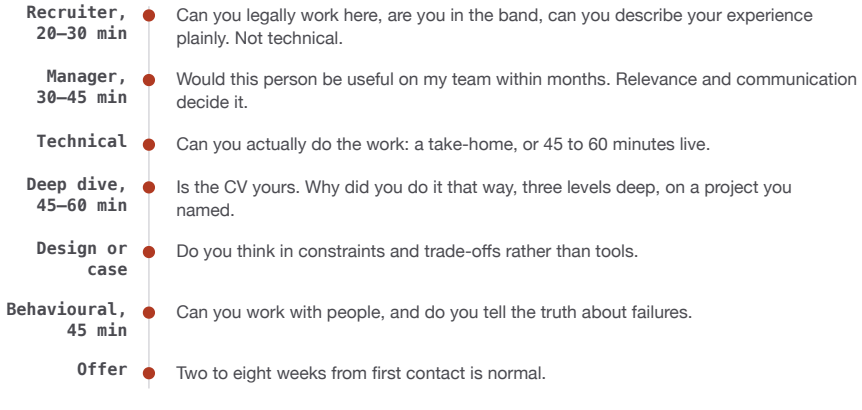
4. Deep dive on your own work — 45 to 60 minutes. Tests whether the things on your CV are yours. Expect "why did you do it that way" three levels deep on a project you named. Prepare the two projects you will be asked about; do not let them pick a third you barely remember.

5. Design or case conversation — 45 to 60 minutes. Tests whether you think in constraints and trade-offs rather than tools.

6. Behavioural — 45 minutes. Tests whether you can work with people and whether you tell the truth about failures.

7. Offer.

Seven stages, and what each is testing



Knowing which stage you are in tells you what to say. A round-four answer in round one is a common error, and the notes from each round travel forward to the next panel.

Timelines, and what they tell you

Two to eight weeks from first contact to offer is normal. Under two weeks usually means a small company or an urgent gap. Beyond about ten weeks, with long silences between rounds, you are learning something real about how that organisation makes decisions — and you will experience the same thing as an employee waiting for a decision about your work.

Ask at the recruiter screen: how many stages, and what is the expected timeline. A recruiter who cannot answer is a signal in itself.

The thing nobody tells candidates about feedback

Panels write down their assessment after each round, and those notes travel forward. The interviewer in round four has usually read what round two thought before they meet you.

Two practical consequences. First, a weak first round is not erased by a strong third; prepare the early rounds harder than instinct suggests, because they are cheap for the company and decisive for you. Second, if you gave a poor answer, you may raise it later — *"I have been thinking about the join question from Tuesday and I got it wrong; here is what I would do"* — and it is read as strength rather than weakness, because the notes are already there and you are showing you can revisit your own work.

Between the rounds

Send three lines afterwards, naming one concrete thing from the conversation. Not for politeness. It puts a written trace of you into the loop, in your own words, at the moment the panel is deciding.

```
Thanks for the time today. The point about the nightly load
overlapping with the batch export was the part I keep thinking
about — I had not considered that the retry window and the
export window could collide. Enjoyed the conversation.
```

Four sentences, specific, no request.

Rejection, and what you can learn from it

You will usually be given no reason. This is legal caution rather than rudeness; a specific reason is a document that can be argued with, and most companies have decided not to produce them.

What you can still learn is the **stage**. Rejected before any human conversation means the route or the paper is wrong — go back to the employer categories and the referral routes. Rejected after the technical means a skill gap you can name. Rejected after the final round repeatedly means you are close, and it is usually a small number of specific things; that is the point at which asking a friendly interviewer for one piece of feedback is most likely to succeed.

Practising without paying

Interview coaching costs from about a hundred to several hundred per hour, and much of what it sells is rehearsal with an audience. A peer who does the job — swapping mock interviews in a community, an hour each way — is free and often better, because they know the questions your market actually asks. Free mock-interview swaps exist in most large technical communities. Record yourself on a phone and watch it back once; it is unpleasant and it works.

BEFORE YOU MOVE ON

Twenty minutes into a first call, the recruiter asks what you are looking for in terms of compensation. What is the most useful answer?

A. Give a precise figure slightly above your target, to anchor the negotiation early.

B. Ask what band the role is budgeted at, and if pressed, give a range you would genuinely accept — because this stage exists partly to check you are affordable, and finding out now saves both of you weeks.

C. Say compensation is not important to you and that you care most about the work.

D. Defer the question entirely until an offer is on the table.

51 · 9 min

The first call, and the salary question

THE ONE THING TO KEEP

The recruiter screen tests availability, motivation and band rather than skill, and the two questions worth spending your last minutes on are the internal level and why the role is open.

Twenty-five minutes with somebody who is not technical

The first human contact is usually a recruiter. They are not assessing your SQL and it is unfair to judge them for not being able to. They are checking five things:

1. Are you real, and roughly what your CV says?
2. Can you explain your work to a non-specialist?
3. Are you actually interested in this role, or in any role?
4. Are you available — notice period, location, right to work?
5. Are you inside the band?

Most rejections at this stage come from three, four and five, not from anything technical. Prepare accordingly.

The opening ninety seconds

"Tell me about yourself" is answered in the order **present, past, why here**. Sixty to ninety seconds.

> *"I am a data analyst at a logistics firm, mostly SQL and reporting for the operations team. Before that I was in operations myself, which is how I ended up doing this. The thing I have been moving towards is pipeline work rather than reporting, and your advert is the first one I have seen that pairs that with a domain I already understand."*

Three sentences, no life story, and it ends by handing them a reason to keep talking. Do not start with where you were born.

The salary question

It arrives early and it is the part people handle worst. Three facts change how you should answer it.

First, in a growing number of places the band is disclosable and sometimes must be published. Several US states and cities require a salary range in the posting or on request. The EU Pay Transparency Directive obliges employers to give applicants pay information before the interview, with member states implementing by June 2026. Ask. In many jurisdictions you are entitled to the answer, and where you are not, asking is still normal.

Second, asking your current salary is prohibited in a number of places — California, New York City, Massachusetts and several other US jurisdictions ban it outright. Elsewhere it is legal and common. You are never obliged to answer, and answering anchors your next job to your last market, which is precisely the trap when you are moving from a services firm to a product company or from one country to another.

Third, whoever names a number first sets the frame. That is not a reason to refuse forever; it is a reason to try once to get theirs.

A sequence that works:

> ***Them:*** *"What are your salary expectations?"*

> ***You:*** *"I would rather understand the level first — what is the band for this role?"*

Most internal recruiters answer, because they have the band on their screen and a mismatch wastes their time too. If they press:

> *"Based on what I have seen for this level in this market, I am looking at somewhere between X and Y, and I would want to understand the whole package before being firm. Does that sit inside your band?"*

Rules for X and Y. Research first, from real postings and people rather than from a model, which is confidently wrong about local pay. Make X a number you would genuinely accept, because you may be offered it. Set the range about 15% wide. And never quote your current salary as the basis.

If they will not give a band and will not confirm you are in it, that is information about the company, not a puzzle to solve.

What to ask, and the one question that pays

Save two minutes at the end.

- **"What level is this role internally, and what would distinguish it from the level above?"**
- "What are the stages and the timeline?"
- "Who would I report to, and how large is the team?"
- **"Why is the role open?"**

That last one is the highest-yield question in the whole call, and recruiters usually answer it honestly. A new role from growth, a backfill after somebody left, a replacement for somebody who left after four months, or a role being rebuilt after a reorganisation are four very different jobs. The answer sometimes arrives with a pause, and the pause is also an answer.

Internal recruiter or agency

Worth establishing in the first minute, because the incentives differ. An **internal recruiter** works for the company and is measured on hires that stay. An **agency recruiter** is typically paid a percentage of your first-year salary on placement, which makes them genuinely useful — they want you placed and paid well — and also means they may push you towards roles that suit their pipeline rather than your plan.

Neither is a villain. But never let an agency submit you to a company without naming it first, because duplicate submissions from two agencies can get a candidate discarded to avoid a fee dispute.

The honest limitation

A recruiter screen is a low-information stage in both directions. They cannot tell whether you can do the job, and you cannot tell whether the job is good. Its only real function is to avoid wasting the next four hours, so answer clearly, get the band and the level, and do not read too much into warmth. Recruiters are warm professionally, and it means nothing about your chances.

BEFORE YOU MOVE ON

A recruiter asks a candidate what they currently earn. The candidate is moving from an IT services firm, where pay is well below product-company rates. What is the best response?

A. Give the current figure honestly, since being caught understating it later would be worse.

B. Inflate the figure moderately so the anchor sits closer to the target band.

C. Decline to answer entirely and end the discussion of pay.

D. Say that their current pay reflects a different segment of the market, ask for the band for this role, and give a researched range for this level if pressed.

---

52 · 9 min

The take-home, and how not to lose on it

THE ONE THING TO KEEP

A take-home is graded on judgement and communication, not on completeness, so scope it, time-box it, and write down what you left out.

What the grader is actually scoring

Almost never "did they finish everything". Usually, in this order:

1. **Does it run?** From a clean checkout, following the README, on their machine. A surprising share of submissions fail here, and it is an immediate no.
2. **Can I read the README in two minutes and know what you did?**
3. **Did you notice the mess?** Every well-made take-home dataset has a deliberate trap.
4. **Would I want to review this person's code every week?** Names, structure, comments where the reason is not obvious.
5. **Did you say what you would do next?**

Note that four of the five are communication.

Time-boxing, and saying so

If the brief says four hours, spend four to five, and write at the top of the README what you spent and what you cut.

```
Time spent: about 4.5 hours. With another day I would have
added tests for the currency parsing and looked at why 6%
of rows have no customer_id.
```

Spending twenty hours and not saying so is dishonest, and it also loses: the grader is comparing you against a four-hour bar, and a submission that is polished far beyond it reads either as a misjudged sense of scope or as a candidate who could not stop. Judgement about scope is part of what is being tested.

The trap is the test

Every good take-home dataset contains something planted: duplicate rows, mixed timezones, a currency column in three formats, an ID column with leading zeros, negative quantities that mean returns, or a target that leaks.

You do not have to fix all of them. You do have to **name them**. A README section headed *What I found in the data* with four bullets scores better than a silent perfect model, because it shows you looked.

Leakage is the classic and worth understanding properly. You are asked to predict which customers will request a refund. The dataset contains `refund_processed_at`. Any model using it scores near-perfect and is worth nothing, because at prediction time that column is empty — it only exists after the event you are predicting. The mechanism is that the feature is downstream of the target rather than upstream of it. Interviewers plant this deliberately, and catching it is worth more than any modelling choice you make.

The README that scores

Same skeleton as your project pages:

- How to run it. One command if possible.
- What I did, and why — the two decisions that mattered.
- What I found in the data.
- How I checked it — the number, however small.
- What I did not do, and what I would do next.
- Time spent.

The red lines

Some take-homes are not assessments.

- A brief that is a real business deliverable — "here is our actual dataset, produce the report we need this month".
- More than about eight hours of unpaid work.
- "Build a prototype of our product" before any technical conversation.
- A take-home given before anyone at the company has spoken to you at all.

Decline these politely and offer an alternative: *"I would rather not spend twenty hours before we have talked. I am happy to do a live 90-minute session, or to walk you through a project I have already built."* Some companies will accept. The ones that will not have told you how they will treat your time later, which is worth knowing for free.

Asking for a longer turnaround is different and almost always granted. If you have a full-time job, ask for a week rather than 48 hours. Refusal to accommodate that is itself information.

Deploying it for free

If the brief wants something running, you do not need to pay for hosting:

- **Hugging Face Spaces** — free CPU tier, good for a demo with an interface.
- **Fly.io** and **Render** — free or near-free small instances.
- **GitHub Pages** — for anything static.
- **Streamlit or Gradio** — a working interface in forty lines, both free and open source.

A deployed link takes the reviewer five seconds to check and a repository takes five minutes. That difference decides some processes.

The honest limitation

Take-homes systematically disadvantage people with a full-time job, caring responsibilities, or unreliable electricity and internet, because they convert the assessment into a test of available spare time. Nothing you do fixes that for the industry. What you can do is negotiate the deadline, state your time budget clearly, and treat a company that will not move on either as having answered a question about themselves.

BEFORE YOU MOVE ON

A take-home asks you to predict which orders will be returned. Your model scores 0.98 AUC on the first attempt, far above anything you expected. What should you do first?

A. Submit it, noting that the result is strong and the approach was straightforward.
B. Reduce the model's complexity to avoid overfitting, then re-run and report the lower score.
C. Add cross-validation to confirm the result is stable before submitting.
D. Examine the features for one that could only be known after a return happened — a processed-at timestamp, a refund flag, a status field — remove it, re-run, and write up both the discovery and the honest score.

53 · 8 min

The live exercise, where silence is scored

THE ONE THING TO KEEP

In a live exercise the interviewer is grading your narration, so saying where you are stuck scores better than being quietly stuck.

The setup

Forty-five to sixty minutes, a shared editor or a SQL sandbox, one or two problems, one or two interviewers watching. Sometimes on your own machine with your screen shared, which is better because you keep your own tools.

The four moves, in order

Restate the question. "So you want, for each customer, the date of their second order — and customers with only one order should still appear, with a

null." Thirty seconds, and it catches half of all misunderstandings before they cost you twenty minutes.

State your assumptions out loud. "I am assuming order_id is unique and that cancelled orders should be excluded — is that right?" Interviewers will usually tell you. Assumptions you keep to yourself become wrong answers.

Name the approach before typing. "I will use a window function to number each customer's orders by date, then filter to row number two." Now the interviewer can correct your direction before you spend fifteen minutes implementing it.

Check the answer against a case you can compute in your head. Take one customer with three orders and verify by eye. Candidates who do this are rare and it is noticed every time.

Why narration decides the outcome

The interviewer cannot see your thinking. They see keystrokes and hear words. A candidate thinking brilliantly in silence for four minutes and a candidate who is completely lost look identical from the other side of the screen.

So a candidate who says *"before I write this I want to check whether joining to order_items can duplicate rows"* scores above one who silently writes the same correct query — because the first has demonstrated the reasoning, and the reasoning is what transfers to the job.

What to say when you are stuck

Be precise about the boundary of what you know. This is a strong answer:

> *"I know I need the second highest value per group. I am deciding between a window function and a correlated subquery. I would use* `row_number()` *over a partition, but I do not remember the exact syntax for the frame clause. May I check the documentation?"*

That tells the interviewer you understand the problem, know two approaches, and have a normal professional relationship with reference material. Almost

every interviewer says yes to the documentation question, and a refusal tells you something about the job.

The only wrong response is freezing. If your mind empties, say so — *"I have lost the thread, give me twenty seconds"* — and then restate the question aloud. Restating restarts thinking more reliably than staring.

The exercises that actually come up

- Aggregate with a join, then a group-level filter.
- A cleaning question: "this date column has five different formats — what do you do?"
- "Explain what this query does" — reading, not writing.
- "What would you check before trusting this number?" — the reconciliation answer from the spreadsheet lesson.
- Debugging somebody else's broken query or script. This one is increasingly common, because it is closest to the job.

The pitfalls that cost points

- `count(*)` after a `left join`, which counts one for the null row too. Use `count(column_from_the_right_table)`.
- A `where` clause on a left-joined table, which quietly turns it into an inner join — the null rows fail the predicate and disappear. Move the condition into the `on` clause.
- Summing a left-table column after a one-to-many join. The fan-out again.
- Money in floating point. `0.1 + 0.2` is not `0.3`, and financial totals drift. Use a decimal or integer minor units.
- Assuming `null` behaves like zero or like an empty string. It behaves like neither.

Practising for free

You do not need a subscription. **DuckDB** with a downloaded CSV gives you a sandbox in one command. **SQLZoo**, **pgexercises** and **Kaggle**'s SQL courses

are free and cover more than most interviews test. For live conditions, the thing that helps most is doing it with somebody watching — swap sessions with a peer, since the difficulty is the observation and not the SQL.

The honest limitation

Live coding measures composure under observation at least as much as it measures skill. Competent working engineers fail these regularly, and people who have rehearsed pass while being no better at the job. That is a known weakness of the format, not a judgement about you.

The implication is practical: the fix for failing live exercises is rehearsal with an audience, not more study. If you can solve the problem alone in ten minutes and not in the room, another course will not help you. Ten mock sessions will.

BEFORE YOU MOVE ON

Halfway through a live SQL exercise you realise you cannot remember the syntax for a window function you know you need. What is the best thing to do?

A. Say which approach you want, why, and exactly what you cannot recall, then ask whether you may check the documentation.

B. Work around it with a correlated subquery you are confident in, without mentioning the window function.

C. Stay quiet and reason it out; interrupting the flow to talk will slow you down.

D. Say you are stuck and ask the interviewer for the answer so you can continue with the rest of the problem.

54 · 9 min

"How would you build this?" answered well

THE ONE THING TO KEEP

A design answer is a set of constraints, measurements and failure modes, not an architecture diagram.

The question you will get

> *"We have a help centre with about four hundred articles. How would you build something that answers customer questions from it?"*

This has become a standard interview question at almost every level, including entry, because it is cheap to ask and reveals a great deal. It is not a test of whether you know the current fashionable stack.

The five steps, in this order

1. Ask who uses it, how often, and what a wrong answer costs.

An internal tool for support agents who can see the source article is a completely different system from a public one answering questions about medication. The cost of a wrong answer sets everything else: how conservative the retrieval is, whether there is a human in the loop, whether it may answer at all when unsure. Beginners skip this and start naming tools. Ask it first, every time.

2. Say how you would measure it before you say how you would build it.

"Before building anything I would take fifty real questions from the support queue, write down what a correct answer must contain, and score whatever we build against them. I would also score plain keyword search on the same fifty, so we know what the improvement actually is."

This one paragraph moves you above most candidates, because it demonstrates the habit that separates people who ship working systems from people who ship demos.

3. Data, and who owns it.

Where do the articles live? How often do they change, and does anyone tell you when? Are they all current, or are a third of them from 2021 and contradicting each other? Is there anything in them that should not be shown to a customer? A system built over stale documents answers confidently and wrongly, and no model choice fixes that.

4. The simplest thing that could work.

Search the articles, retrieve the top few passages, have a model answer using only those passages, and show the source link with every answer. Citations are not a feature; they are what makes the thing checkable.

Say out loud that you would try plain keyword or full-text search as the baseline. Interviewers notice. It is frequently good enough on four hundred articles, and it is far easier to debug — when it retrieves the wrong document you can see exactly why.

5. Failure modes and cost.

What happens when retrieval finds nothing relevant? The answer should be "it says it does not know and offers to hand over to a person", not "it answers anyway". What happens at ten times the traffic? What does a query cost — a few thousand tokens in and a few hundred out is cents rather than dollars, but multiplied by fifty thousand queries a month it becomes a real budget line, and that arithmetic is worth doing out loud.

The senior tell: saying what you would not build

> *"I would not fine-tune a model on the articles. They change weekly, fine-tuning bakes them in, and fixing a wrong answer would mean retraining. With retrieval I fix it by editing the article."*

Explaining a rejected option with its mechanism is the strongest single move available in this conversation, and it works at any experience level because it is reasoning rather than knowledge.

What loses

- Naming five products in the first thirty seconds with no reason attached.
- Jumping straight to a vector database before establishing what is being measured.
- Never mentioning evaluation at all. This is the most common failure, and for AI roles it is close to disqualifying.
- Designing for a million users when the company has four hundred articles and six support agents.

Building the whole thing free

Everything in that design has a free path, which matters because the strongest possible answer is "I built a small version of this":

- **Search:** PostgreSQL full-text search, SQLite FTS5, or DuckDB. If you need vectors, **pgvector** or **Chroma**, both free and self-hosted.
- **The model:** a free API tier, or a small local model through **Ollama** on a laptop.
- **Hosting:** a free **Hugging Face Space** with **Gradio**.
- **Evaluation:** a CSV and a script.

The named paid products — Pinecone, Weaviate Cloud, managed retrieval services — buy operational convenience at scale. None of them is required to demonstrate that you understand the problem, and at four hundred articles none of them is required at all.

The limitation

Nobody expects a correct architecture in forty-five minutes, and there is no single right answer to grade against. What is being graded is order of thinking:

constraints before components, measurement before building, failure modes before scaling.

"I do not know how that behaves at ten million documents, and here is how I would find out" is a scoring answer. Confident invention is not, and interviewers who work on these systems can tell the difference immediately.

BEFORE YOU MOVE ON

In a design conversation about an internal question-answering tool, which opening move is strongest?

A. Describe the architecture: documents, embeddings, a vector store, retrieval, a model, an interface.
B. Ask which model provider the company already uses, so the design fits their stack.
C. Ask who uses it, how often, and what a wrong answer costs — then say how you would measure the system before describing how you would build it.
D. Propose starting with a small pilot on a subset of documents to learn before committing.

---

55 · 10 min

"How would you measure it?"

THE ONE THING TO KEEP

Answer a metrics case by naming the decision, one sensitive primary metric, guardrails and a counter-metric — and when diagnosing a sudden drop, check the instrumentation before the users.

A round with three shapes

The design conversation asks how you would build a system. The metrics case asks how you would know whether it worked. Analyst, data scientist and prod-

uct-facing roles nearly all have one, and it comes in three forms.

1. "Define success for this feature"

"We are adding saved searches to the app. How would you measure whether it worked?"

The bad answer starts naming metrics. The good answer starts with the decision.

- **Who is the user and what changes for them?** People who search repeatedly stop retyping.
- **What decision does the measurement inform?** Whether to invest further, or remove it.
- **One primary metric.** Something close to the behaviour: proportion of returning searchers who reuse a saved search within seven days.
- **Two or three guardrails** — things that must not get worse. Search latency. Overall search volume. Support tickets.
- **A counter-metric** — the way this could look good while being bad. If saved searches make people search *less* because they are stuck in narrow saved queries, discovery falls. Name it.

Then say what would make you kill the feature. Interviewers remember candidates who state a failure condition, because most do not.

The property that separates a good metric from a bad one

Three tests, and being able to name them is most of the round:

- **Sensitive** — it moves when the thing you changed moves. Monthly active users is insensitive to almost everything you can ship in a quarter.
- **Hard to game** — "tickets resolved" rewards closing tickets prematurely; "tickets resolved without reopening in 14 days" mostly does not.
- **Attributable** — you can tell whether your change caused it, which usually means it is close to the change in time and in the causal chain. Revenue is the thing everybody cares about and is almost never a usable feature metric, because forty other things move it.

The classic failure is a metric that rises when the product gets worse. Time-in-app for a support tool. Number of searches per session for a search engine. Say the words "this could go up for a bad reason, so I would pair it with…" and you have signalled more experience than a fluent framework does.

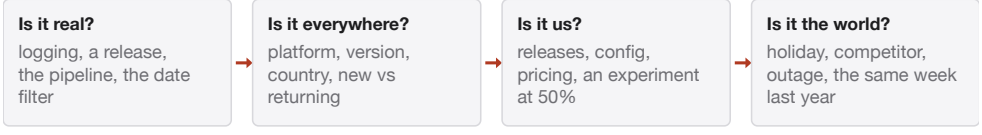
2. "This metric dropped 15% overnight. Diagnose it."

There is a correct order, and the first step is the one that marks experience.

1. **Is it real?** Before anything else: did the logging change, did a release stop firing an event, did the pipeline fail, did a data source stop delivering, is the dashboard's date filter doing something odd. A large share of overnight step-changes are instrumentation, and a candidate who investigates user behaviour before checking whether the number is trustworthy has revealed how much time they have spent doing this.
2. **Is it everywhere?** Segment: platform, app version, country, browser, new versus returning, paid versus organic. A drop confined to Android 14 is a bug; a drop everywhere is an event.
3. **Is it us?** Releases, config changes, pricing changes, an experiment ramped to 50%, a marketing campaign that ended.
4. **Is it the world?** Holiday, competitor launch, outage at a payment provider, a platform policy change, seasonality — and check the same week last year before claiming anything is unprecedented.

Say the order out loud as you go. The round is scored on the search strategy, not on guessing the answer.

A metric dropped fifteen per cent overnight



Check the instrument before the users. A large share of overnight step-changes are a release that stopped firing an event, a failed pipeline or a dashboard filter, and a candidate who investigates user behaviour first has revealed how much of this they have done.

3. "Design an experiment for this"

Six things, in this order, and none of them is optional:

- **Unit of randomisation** — user, session, device, or something coarser like a city when there are network effects. Getting this wrong invalidates everything after it.
- **Primary metric and guardrails**, chosen before the test.
- **Minimum detectable effect and sample size.** Use the arithmetic from the statistics lesson: roughly `16 * p * (1–p) / difference²` per arm. Then say how long that takes at their traffic. This is the single moment in the round where a real number wins it.
- **Duration** — at least one full weekly cycle, because Tuesday users are not Saturday users.
- **A stopping rule agreed in advance.** No peeking and stopping at significance, which manufactures false positives.
- **What you would do with each outcome**, including no difference. If the answer is "ship it anyway", the test was theatre and it is better to say so and skip it.

The honest limitation

This round rewards fluency, and fluency is memorisable. Interviewers know that, so a smooth framework recited without a number in it now reads as preparation rather than experience.

The discriminator is specifics: an actual sample-size calculation, an actual counter-metric for this product, an actual reason to reject the obvious metric. One concrete trade-off stated out loud is worth more than four tidy headings.

BEFORE YOU MOVE ON

A candidate is asked why a company's daily active users fell 15% overnight. Which opening move signals the most experience?

A. Segment by platform and country to find where the drop is concentrated.

B. Ask whether a release, experiment or pricing change went out the previous evening.

C. Check whether the number is real — logging changes, a failed pipeline, a release that stopped firing the event, a broken dashboard filter — before investigating user behaviour at all.

D. Compare against the same week last year to rule out seasonality.

56 · 8 min

Talking about your work without lying or shrinking

THE ONE THING TO KEEP

Prepare six stories with real numbers and tell them the same way every time; the failure story is the one that decides the round.

Six stories, prepared once

Most behavioural questions are variations on six situations. Write these out, once, properly, and you are ready for almost any version of them.

1. **Something you built.** The one you are proudest of, with who used it.
2. **Something that broke, and what you did in the following hour.**
3. **A disagreement** with a colleague or a manager, and how it resolved.
4. **A time you were wrong about data** — you reported a number and it was incorrect.
5. **A time you pushed back** or said no.

6. **Something you had to learn quickly** under real pressure.

Numbers 2 and 4 are the ones that decide rounds, and the ones almost nobody prepares.

The shape, without the acronym

One sentence of situation. What *you* did, specifically. The number. What you would do differently. Two minutes, not eight.

> *"The weekly revenue report was wrong for two months and I did not catch it. I had joined orders to order items to filter by category and then summed the order total, which counted each order once per item — revenue was inflated by about forty per cent. Finance found it, not me, which was the embarrassing part. I fixed it, republished the corrected figures with a note explaining the cause, and added a row-count check before and after every join in that pipeline. I now check counts on every join, and I would have caught it in week one if I had reconciled against the finance export at the start."*

That is under ninety seconds and it does five things: names a real error, quantifies it, admits how it was found, describes the systemic fix, and states the habit that came out of it.

Why the failure story matters most

They are hiring somebody who will one day break production. Everyone does. What they are buying is your behaviour in the hour afterwards — whether you notice, whether you say so immediately, whether you fix the cause or the symptom.

The failing answer is the deflection: "my weakness is that I care too much" or "I once worked too hard on a project". Interviewers hear these several times a week and score them as an unwillingness to be examined, which is exactly the trait that makes an incident worse.

The other failing answer is a failure that was somebody else's fault. If the story ends with a colleague being wrong, it is not a failure story.

"I" and "we", precisely

Say "I" for what you did and "we" for what the team did, and be exact about the boundary. Interviewers cross-check by asking follow-ups — *"how did you decide on the median?"*, *"who wrote the retry logic?"* — and an inflated claim collapses in one question. A candidate who says "the pipeline was mostly my colleague's; I built the validation layer on top of it" gains credibility, not loses it.

Interviewing in a second language

If you are interviewing in English for a nearshore, remote or overseas role, the bar is being understood, not sounding native. Nobody is scoring your accent; they are scoring whether a colleague could work with you across a video call.

Four things that measurably help:

- **Slow down by about twenty per cent.** Almost all comprehension problems on calls are speed, not vocabulary.
- **Short sentences.** One idea each. Long clauses are where second-language speech becomes hard to follow.
- **Ask them to repeat rather than guessing.** "Sorry, could you say that again?" is normal and costs nothing. Answering the wrong question costs the round.
- **Rehearse the six stories out loud until the words are automatic.** This is the mechanism that matters: composing an answer and translating it at the same time consumes all your working memory, and there is none left for thinking. Rehearsal removes the composing half, so the whole load goes to the content.

Rehearse aloud, not in your head — silent rehearsal does not produce the automaticity.

Have the numbers ready

"About thirty users." "Four hours down to fifteen minutes." "Forty test cases, thirty-three correct." "Three months, alone, alongside my job."

Specific numbers read as memory. Vagueness reads as invention, whether or not it is, because a person who did the work remembers its size.

Free practice

Record yourself on a phone answering the six questions, and watch it back once. It is unpleasant and it is the fastest correction available: you will hear the filler, the length, and the place where you stopped answering the question. Then swap mock interviews with a peer. Paid coaching does the same thing with a bill attached.

The limitation

Some interviewers grade for "fit" in ways that are about them rather than you — accent, background, school, confidence performed in a particular cultural register. You cannot fix that from inside the room, and trying to will make you worse. Treat one bad room as data about the room, and pay attention only when the same specific feedback appears three times.

BEFORE YOU MOVE ON

An interviewer asks about a time you made a significant mistake. Which answer is strongest?

A. "I am a perfectionist, so I sometimes spend too long polishing work before releasing it."

B. A specific error you made, what it cost, how it was found — including if someone else found it — what you changed so it cannot recur, and what you would do differently from the start.

C. A project that failed because requirements kept changing and stakeholders could not agree.

D. A small, safe error with no consequences, such as a typo caught in review before release.

57 · 8 min

Your questions, and what the answers reveal

THE ONE THING TO KEEP

Your questions decide whether the job is real: ask about where the data lives, what is in production, and why the role is open.

Why this matters more in this field than most

An "AI role" at a company with no data warehouse, no pipelines and nothing in production is a year of plumbing sold as machine learning. That is not always a bad job — early plumbing is how a lot of good careers start — but you should know which one you are accepting.

Your questions are the only instrument you have.

The seven questions

"Where does the data live today, and who owns it?" The strong answer names systems and people. A vague answer usually means it lives in application databases and spreadsheets, and that extracting it is your first year.

"What is in production that uses a model, and when did it last change?" The most diagnostic question in the list. "We are exploring several initiatives" means nothing is in production. A specific answer — "the fraud scoring model, retrained monthly, last deployed in July" — tells you there is a working system to learn from.

"How do you know when a change made things better?" If nobody can answer, there is no measurement culture, and your work will be judged on how it looks in a meeting.

"What did this team ship last quarter, and how long did it take from idea to live?" Cuts through everything. Six weeks means a functioning

team. "We are still waiting on approvals from a project started last year" means your work will queue.

"Who would I ask when something breaks at nine in the evening?" Tests whether there is anyone senior, whether there is an on-call arrangement, and whether it is written down.

"What does the first ninety days look like for this role?" A manager who has thought about you arriving has an answer. One who has not is hiring a pair of hands for an unspecified backlog.

"Why is this role open?" Growth, backfill, or a reorganisation. All three are fine and they are different jobs. If it is a backfill, ask — pleasantly — what the previous person went on to do. A predecessor who lasted seven months is not disqualifying, but it is a thread worth pulling.

Reading the answers

Hesitation matters more than content. Everything you are told is the best available version of the truth, so the reliable signal is behaviour: how quickly they replied to your emails, whether the process matched what they described, whether the interviewers had read your CV, whether anyone was late without saying so.

A company that runs a shambolic interview process runs shambolic projects. The interview is the sample.

What not to ask

Anything on the website. Holiday policy in round one — ask the recruiter later, when it costs nothing. "What does the company do." And avoid the performed question that is really a statement about how diligent you are; interviewers can tell, and it wastes the slot.

Asking about money

Ask the recruiter, early, plainly: "What is the band for this role?" It is a normal question, it is legally required to be answered in a growing number of places,

and it prevents four weeks of interviews for a job that cannot pay you. Nobody senior thinks less of a candidate for asking. The awkwardness is entirely in your head, and it costs real weeks.

Checking the answers independently

Free sources for pay and culture signals: **Levels.fyi**, **Glassdoor**, **AmbitionBox** in India, national salary surveys published by statistics offices and industry bodies, and public job adverts from the same company that do list a band.

Better than any of them: ask a current or former employee directly, using the outreach method from the previous module. A ten-minute conversation with someone who left eight months ago is worth more than every review site combined, and people are usually willing.

The question to ask yourself afterwards

Not "did I do well" but "would I want to spend two years learning from these people". A job where the work is dull but two colleagues are excellent teaches more than a job with a fashionable stack and nobody to learn from. At the start of a career, who you sit next to is the largest single factor in how fast you improve, and it is the one candidates weigh least.

BEFORE YOU MOVE ON

You ask what is currently in production using a model. The manager says the team is "exploring several AI initiatives and building capability for the year ahead". What is the most accurate read?

A. The company is early but well-resourced, and the role offers a rare chance to build from scratch.
B. It is a warning sign of an unserious employer, and the process should be ended.
C. The manager is protecting confidential information about systems they cannot discuss with candidates.
D. Nothing is in production. That is not automatically a bad job, but it means the first year is data plumbing rather than modelling — so ask about warehouse, pipelines and who owns the source systems before deciding.

58 · 9 min

The offer conversation

THE ONE THING TO KEEP

Almost everything is negotiable once, politely, in writing — and a first offer is rarely the top of the band.

The mechanics

Get it in writing before you respond to anything. Ask for the full picture: base, bonus and how it is determined, equity and the terms attached, probation length, notice period, working hours, remote policy, start date. A verbal offer is a conversation; the written one is the thing.

Then take two to five days. Nobody withdraws an offer because you asked for the weekend. Never accept on the call, even if you are certain, because the moment you say yes the conversation is over.

The sentence that works with no leverage

You have no competing offer. You still ask, once, calmly:

> *"I am really pleased with this and I want to accept. Based on the scope of the role and what I have seen in the market, I was hoping for X. Is there room?"*

Then stop talking. The silence after the question is the entire technique — most people fill it and negotiate against themselves.

Realistic outcome without a second offer: five to ten per cent, or a non-cash concession. Not forty. Anyone promising you forty is selling a course. Five per cent on a first salary compounds through every subsequent raise and every subsequent offer that asks what you currently earn, which is why the two-minute conversation is worth having even when the number is small.

What else is negotiable

More things than people ask for, and often easier than base:

- Start date — worth real money if you have a notice period to serve.
- A signing amount, which comes from a different budget than salary.
- Equipment or home-office budget.
- Remote or hybrid days, in writing rather than by custom.
- Title, which costs nothing today and affects your next application.
- A written salary review at six months with a defined band.
- A training or conference budget.
- Notice period length, in markets where it is long.

The contract clauses that matter in this field

Intellectual property assignment. Read whether it covers work done on your own time and equipment. In some jurisdictions a blanket claim over everything you create is legally unenforceable; in others it stands. If you have side projects or intend to publish, ask for a carve-out naming them. This is a

routine request and is usually granted when asked before signing, and never after.

Non-compete. Unenforceable or heavily restricted in some places, actively enforced in others. Read the duration and the geography. A twelve-month worldwide non-compete for a junior analyst is not normal.

Notice period. Sixty to ninety days is standard in India and parts of Asia and affects everything about your next search — employers plan around it and some will not wait. Know yours before you start looking.

Training bonds and service agreements. Common in some markets: you agree to repay a sum if you leave within a period. These are real financial instruments and people sign them without reading the amount. Check the sum, the duration, and whether it survives redundancy. If you would not be able to pay it, do not sign it.

The counteroffer from your current employer

You resign and they improve your terms. Two honest observations. First, the reason you were leaving usually still exists in six months, and if it was money, being underpaid until you threatened to leave is itself the information. Second, in some organisations a resignation permanently changes how you are viewed, and in others it does not — you know which one you are in.

Treat a counteroffer as evidence about what they could have paid you all along.

Exploding offers

"We need an answer by tomorrow morning." A company that manufactures pressure at the offer stage will manufacture it at every subsequent stage. Ask for forty-eight hours. A real employer says yes. If they refuse, you have learned something at no cost, and it was probably the most useful thing the process told you.

Where negotiation genuinely does not apply

Be realistic about your market. Campus placements, large IT services intakes, public sector grades and unionised pay scales have fixed bands that an individual cannot move, and pushing hard against them wastes goodwill for nothing. In those settings, negotiate the things that are actually variable: location, start date, project allocation, team.

Knowing which market you are in is most of the skill. The rest is one sentence and the willingness to be quiet afterwards.

BEFORE YOU MOVE ON

You receive an offer at the bottom of the band you were told, with no competing offer in hand. What is the best move?

A. Ask once, plainly, for a specific higher figure with a short reason — then stop talking and let them respond.

B. Accept it, and plan to negotiate at the six-month review when you have proved your value.

C. Say another process is at final stage to create competitive pressure.

D. Ask for the band's midpoint and list every reason you deserve it, in detail.

59 · 8 min

Pay, rates, and who actually decides them

THE ONE THING TO KEEP

Pay is set by employer type, country and your alternatives, far more than by your job title.

Read these numbers as shapes, not quotes

Salaries in this field vary by a factor of ten across countries and by a factor of three within a single city. Anyone quoting you one global number for "AI engineer salary" is selling something. What follows is rough, approximate for 2026, and should be checked locally before you act on it.

Junior analyst or junior data engineer, first role:

- India: ₹4–9 lakh per year
- Nigeria: ₦4–12 million per year
- Philippines: ₱30,000–70,000 per month
- Brazil: R$4,000–9,000 per month
- Poland: PLN 8,000–15,000 per month
- Germany: €45,000–60,000 per year
- United States: $65,000–95,000 per year

Mid-level ML engineer, three to five years: roughly two to three times the junior figure in the same market. In the US, $130,000–200,000 is a common band.

Two distortions to keep in mind. First, the widely shared enormous packages come from a handful of frontier labs and big tech firms and describe perhaps a few thousand jobs worldwide. They are the tail, not the field. Second, employer type moves pay more than title does: in the same city, a bank, a startup, an

outsourcing firm and a university will pay very different amounts for identical work.

Where the leverage is

Most people believe pay follows skill. It follows alternatives. The single largest lever in any negotiation is a second offer, and the second largest is not being the one who names a number first. "What is the band for this role?" is a complete and reasonable answer to "What are your expectations?"

If you are given equity in a startup, value it at zero when deciding whether the salary works. Most startup equity ends up worth nothing. Occasionally it is life-changing. Both facts are true and only the first should drive your rent.

Freelancing and contracting

Early rates on global platforms for data cleaning, automation, dashboards and scripting sit around $15–40 per hour, and experienced people with a clear niche charge $60–150. Local clients often pay less per hour but find you faster, pay sooner, and give you references in your own language.

A contract rate is not a salary. Do this arithmetic before comparing anything:

```
$30/hour × 40 hours × 52 weeks     = $62,400   (fantasy)

Realistic billable weeks              38
Realistic billable hours per week     25   (selling, admin,
gaps)
$30 × 25 × 38                      = $28,500
minus self-employment tax, own health cover,
unpaid sick days, unpaid holidays, one client
who never pays the last invoice
```

That is the honest comparison against a salary. Contracting can pay much better than employment, but the multiplier that makes it work is roughly two to three times the equivalent hourly salary, not the same number.

Protecting yourself

- Take 30–50% before starting. A client who refuses is telling you something.
- Bill in milestones, not one payment at the end.
- Put the scope in writing, including what counts as "done" and how many revisions are included.
- Do not hand over the final files, credentials or deployment before the final payment clears.
- Cross-border work means currency swings and slow transfers. Price with a margin for both.

None of this is cynical. It is the ordinary equipment of getting paid, and every experienced freelancer learned it by losing money once.

BEFORE YOU MOVE ON

Chidi contracts remotely from Lagos at $30 per hour. Lena is employed in Berlin at the equivalent of roughly $22 per salaried hour, doing the same work. Who is better off?

A. Chidi, because his headline rate is higher and he keeps more of what he bills.

B. Lena, because salaried employment in Europe is always safer than contract work.

C. It cannot be judged from the rates alone; the comparison is annual take-home after unpaid weeks, taxes, benefits and payment risk.

D. Chidi, because the same money buys much more in Lagos than in Berlin.

60 · 8 min

What a rejection tells you, and what it does not

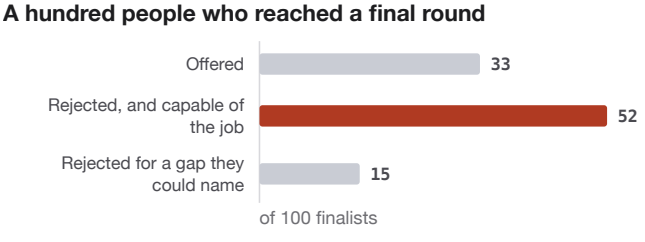
THE ONE THING TO KEEP

Most rejected finalists were hireable, because hiring ranks candidates under a budget rather than assessing them absolutely — so read your log in tens, not one at a time.

The arithmetic of losing

At the final stage, a strong candidate converts to an offer perhaps a third of the time. That means two out of every three people who reach a final round are rejected, and most of them were entirely capable of doing the job.

This is not consolation. It is the mechanism. Hiring is a **ranking against other candidates under a budget**, not an absolute judgement of you. A rejection is compatible with all of the following, none of which is about your ability:



About a third convert. Of the rest, most could have done the job: somebody was slightly better on the day, the internal candidate appeared, the headcount froze, the manager left. Hiring ranks candidates under a budget; it does not assess anyone absolutely.

- Somebody else was slightly better on the day, or knew someone on the panel.
- The internal candidate applied in week three.
- The headcount was frozen between your second and third interviews.
- The hiring manager left and the role was withdrawn.

- The team decided mid-process that they needed someone more senior.

Companies rarely explain which of these happened, so candidates supply the explanation themselves, and the one people reach for is always the same and always the least likely.

Asking for feedback so that you sometimes get it

Make it cheap to answer. A long email asking for a detailed review gets nothing; two lines gets a reply maybe a third of the time.

> *"Thanks for letting me know. If there was one specific gap that decided it, I would find that genuinely useful — a single line is plenty. Either way I appreciated the process."*

Two things to know. First, many companies have a policy against detailed feedback, on legal advice, because a specific reason can become evidence in a discrimination claim. Silence is often policy rather than disdain. Second, when you do get feedback, treat it as one data point. A single interviewer's view is noisy; the same comment from three companies is a finding.

The log

The tracker from the funnel lesson gets four extra columns for anything that reached a first conversation:

- **Stage reached.**
- **Their stated reason**, verbatim, or "none given".
- **Your own read**, written the same day, in one sentence.
- **One thing to change** — and it must be one thing, not a list.

Then review after every ten rows, not after every rejection. Reading it after each one is how a bad week becomes a theory about yourself. Reading ten together shows a pattern or shows that there is not one, and both are useful.

Patterns you will actually see: the same weak answer to the same question; falling out at the same stage every time; only applying when you feel confi-

dent, which correlates with roles below your level; four rejections all from companies whose process you disliked, which is a different problem.

Ghosting, and a rule to end it

You will be ignored, sometimes after a final interview. It is rude, it is common, and it is structural rather than personal — a recruiter with 1,400 applications and a frozen requisition sends nothing.

Have a rule so it stops occupying you: **two follow-ups, one week apart, then close the row and stop thinking about it.** Write the rule down before you need it, because the decision is harder in the moment.

Re-applying

Applying again after six to twelve months is completely normal. Their tracking system keeps your history, which cuts both ways: a strong previous process is remembered favourably, and six scattergun applications are also visible.

When a rejection says "please do apply for future roles", it is sometimes politeness and sometimes literal. You can tell the difference by whether they name a role type or a timeframe.

The part that is not tactics

Long searches are damaging, and the damage is not proportional to how tough anybody is. Three things reliably help, and none of them is more applications.

A fixed weekly quota — the eleven items from the volume lesson — so effort becomes a decision made on Sunday rather than a mood negotiated daily. On the days you do the quota, the day was a success regardless of any reply.

Other people. A study partner, a group, someone who is also searching. Isolation makes every rejection larger, and a search run entirely alone is the version that breaks people.

A separation you state out loud. The market's response is information about the market. It is not a measurement of your worth, and the two feel

identical from the inside after the fifteenth rejection. If it is affecting your sleep, eating or health, that is a signal to talk to somebody — a doctor, a friend, a helpline — and not a signal to send more applications.

BEFORE YOU MOVE ON

A candidate has been rejected after three final-round interviews at three companies, with no feedback given. What is the most defensible interpretation?

A. Something specific about their interviewing is failing at the last stage and they should overhaul their approach before applying again.

B. The lack of feedback shows all three companies found the same serious problem and are avoiding saying it.

C. Reaching three final rounds indicates the funnel is working, three losses at roughly a one-in-three conversion is an ordinary outcome, and the useful action is to log each one and look for a pattern once there are ten data points.

D. They should apply to less competitive companies where they are more likely to convert.

CASE STUDY · MODULE 6

Answer by ten tomorrow morning

Rahul, in Jaipur, holding one offer with a twenty-four-hour deadline and a final round scheduled for the following Tuesday.

The offer came by telephone at 16:40 on a Thursday: analytics consultancy, sixty people, ₹9,00,000 base, joining in three weeks. The recruiter added a sentence at the end. *We need your answer by ten tomorrow morning — the budget is being reallocated on Friday.*

The other process was a global capability centre. He had reached the final round, scheduled for Tuesday. At their recruiter screen four weeks earlier he had asked for the band and been told ₹9,50,000 to ₹12,00,000. He had not asked which level the role sat at, which turned out to matter.

How he had got here

The consultancy's process had run over five weeks: a recruiter screen, a hiring manager call, a four-hour take-home he had time-boxed to four and a half and said so in the README, and a deep dive on the take-home in which he had been asked why he chose a left join three separate times. He had answered the third one badly and had emailed the following morning saying he had thought about it and got it wrong, with the version he would write now.

He assumed that email had cost him. It had not — panel notes travel forward, the interviewer in the later round had already read the earlier one, and revisiting your own answer reads as strength rather than as weakness precisely because the note is already there.

What was actually being decided

Not whether to accept. Whether to ask for time — and everything about that felt asymmetric at 16:40 on a Thursday. He had one offer. He had watched a friend push back on a deadline in March and be told the position had gone to somebody else, which may or may not have been true and had been extremely persuasive anyway.

Accepting on the call was available and safe. The cost was that the moment he said yes, the conversation was over: no negotiation, no start date, no six-month review, and a Tuesday final round he would have to cancel while wondering for a year what the band would have produced.

Asking for time risked the offer. It also tested it. A deadline that survives a polite request for the weekend is a real constraint; one that evaporates within an hour was manufactured, and a company that manufactures pressure at the offer stage does it at every subsequent stage too.

The second thing he had not asked

The consultancy's title was Senior Analyst. Three years of experience, sixty-person firm, and a title that was free to give. He had no idea what level that mapped to internally, or whether "senior" there meant anything a larger employer would recognise later.

And he had no competing offer, so any negotiation would be one sentence with nothing behind it.

The list he wrote on the Thursday evening

Before replying to anything he wrote down what was actually negotiable, because the base salary is only the loudest item and often the stickiest:

- The start date, which was worth real money against a notice period.
- A joining amount, which in most companies comes from a different budget than salary and is therefore easier to say yes to.
- A written salary review at six months with a defined band, rather than a vague promise about the annual cycle.
- The title, which costs the employer nothing today and shapes his next application.
- Whether the two remote days were written into the contract or existed by custom.

He also wrote down the number he would genuinely accept, because a range you quote is a range you may be offered the bottom of, and the bottom of it

has to be a number you can live with.

STOP HERE

Before turning the page: what would you have done, and what would it have cost if you were wrong? Write it down. The point of the next section is to compare.

WHAT HAPPENED

He replied by email within the hour: he was very interested, he had one final-stage interview on Tuesday that he had committed to, and he asked for until Monday to give a proper answer. The recruiter agreed in fifty minutes, which told him what the Friday deadline had been worth. On the Monday call he asked two questions before any number: which level the role sat at internally, and what distinguished it from the level above. The answer put Senior Analyst one rung below what the title implied at any larger employer, which he found useful rather than insulting. Then he asked once — "I want to accept; based on the scope we discussed I was hoping for ₹9,80,000; is there room?" — and stopped talking. He got ₹9,50,000, a written salary review at six months with a stated band, and a start date four weeks out. The GCC rejected him after Tuesday's final round without a reason. The offer he had nearly panicked into accepting unchanged was still there, five per cent higher, with a review date in writing.

What did asking for time actually buy, given the GCC rejected him anyway?

Five per cent, a written six-month review, a later start date, and the knowledge that the Friday deadline was manufactured. None of that depended on the second process succeeding. The competing offer would have been leverage; the request for time was not leverage at all, it was simply the precondition for having any conversation, and the conversation is where the five per cent lived.

Why ask about the level before asking about the money?

Because salary bands are attached to levels. A recruiter usually cannot move you far inside a band but can sometimes place you at the next one, so the level is the

larger lever. The answer also told him what the title was worth outside the company, which affects his next application far more than ₹30,000 does.

He had no competing offer. Was the single ask worth the risk?

Yes, and the risk is smaller than it feels. A realistic outcome without a second offer is five to ten per cent or a non-cash concession, and offers are essentially never withdrawn because a candidate asked once, politely, and then stopped talking. Five per cent on a first salary also compounds through every subsequent raise and every future conversation that anchors on what he currently earns.

MODULE 6 · TEST

Check what stuck

6 questions, one right answer each. This one is for you, not for a record: answers and the reason behind each are at the back. If two questions from the same module go wrong, re-read that module before the exam rather than after.

1 A take-home dataset for a refund-prediction task contains a column called `refund_processed_at`. What is it doing there?

A. It is a planted leak, and naming it scores better than a model that uses it.
B. It is the label, and the task is to reconstruct it from the other columns.
C. It is a red herring with no signal, included to test feature selection.
D. It is a timestamp for joining, and should be used to order the events correctly in time.

2 In a live SQL exercise you cannot remember the syntax for a window frame clause. What scores best?

A. Working around it with a correlated subquery you are confident about
B. Writing the query as best you can and correcting it once it runs
C. Saying which two approaches you are choosing between, and asking to check the docs
D. Staying quiet while you reconstruct the syntax from a similar query you wrote last month

3 Asked how you would build an assistant over four hundred help-centre articles, what should come before naming any component?

A. The chunking strategy, since retrieval quality depends on it most
B. Who uses it, how often, and what a wrong answer costs
C. Whether the articles are in one language or several
D. The expected traffic, so the system is sized correctly from the start

4 "Sign-ups dropped fifteen per cent overnight. Diagnose it." What does a strong answer do first?

A. Segment by platform, country and app version to locate the drop
B. Check releases, experiments and pricing changes from the previous day
C. Compare against the same week last year to rule out seasonality
D. Check whether the number is real before investigating any behaviour

5 The recruiter asks your salary expectations in the first ten minutes. What is the best first move?

A. Give a range about fifteen per cent wide, anchored on your current salary
B. Say you are flexible and would rather discuss it once you know the role
C. Ask what the band is for the role, before naming any number
D. Name a single figure at the top of your research, so there is room to come down

6 You have been rejected after three final rounds. What does that most likely mean?

A. Hiring ranks candidates under a budget, so capable finalists are routinely rejected.
B. A specific technical gap is being found repeatedly at the last stage.
C. Your salary expectations are above the band and are being discovered late.
D. Your references are being contacted and are answering less warmly than you had expected.

MODULE 7

The money, and the paperwork underneath it

An offer is a package, and the headline number is the part that tells you least. This module takes the whole thing apart — what the components actually pay out, why most equity is worth nothing, which contract clauses change what the job is worth, how contractor and employer-of-record arrangements differ from employment, and how to compare two offers with arithmetic. Employment law differs by country and this is not legal advice; what is offered here is the shape of the questions and where to take them.

> BY THE END OF THIS MODULE YOU CAN
>
> Take an offer apart into cash, variable pay, employer contributions, equity and contract terms, compare two offers with explicit arithmetic rather than headline figures, and name the clauses and employment structures that materially change what you are being paid.

61 · 9 min

The headline number is the least useful one

THE ONE THING TO KEEP

Compare guaranteed annual cash against guaranteed annual cash; a bundled total figure mixes employer contributions and target variable pay into a number nobody receives.

Six components, and only one gets discussed

1. **Base salary** — the guaranteed cash.
2. **Variable pay** — bonus or commission, which may or may not arrive.
3. **Employer contributions** — pension, provident fund, gratuity accrual, social insurance.
4. **Benefits** — health cover, leave, allowances, equipment, learning budget.
5. **Equity**, which has its own lesson.
6. **Terms** — notice, probation, severance, working pattern.

People compare offers on the first item, or worse, on a total figure that bundles all six and is not cash.

Cost to company, and why it is not your salary

In India and several other markets an offer is quoted as **CTC** — cost to company. It includes things you will never see in your bank account: the employer's provident fund contribution, gratuity accrual, the premium on your insurance, sometimes a notional "flexible benefit" pool, and a variable component quoted at target.

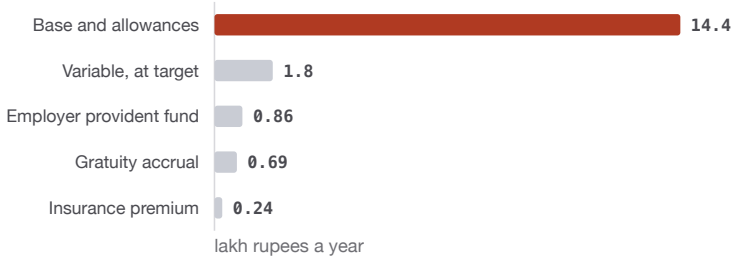
The shape of a ₹18,00,000 CTC offer is often something like:

- Base and allowances: ₹14,40,000
- Employer provident fund: ₹86,400

- Gratuity accrual: ₹69,300
- Insurance premium: ₹24,000
- Variable at target: ₹1,80,000

Two things follow. The **guaranteed cash before tax is ₹14,40,000**, not eighteen lakh — twenty per cent less than the number on the letter. And the variable is quoted at target, so if the team has paid out 60% for two years running, the realistic figure is another lakh short.

An eighteen-lakh offer, taken apart



Guaranteed cash before tax is 14.4 lakh, twenty per cent under the number on the letter. The variable is quoted at target; if the team has paid sixty per cent for two years, the realistic figure is another lakh short.

None of this is dishonest; it is a convention. But comparing a CTC figure against a base salary figure from another country or another company compares two different quantities, and people accept worse offers doing exactly that every day.

The equivalent traps elsewhere: American offers quoting "total compensation" with a four-year equity grant divided by four; European offers where a 13th or 14th month salary is customary and may or may not be included; commission-heavy roles where the "on-target earnings" figure has never been hit by anybody on the team.

The questions that resolve it

Ask these before accepting. All of them are ordinary and none is rude.

- **"Can I have the full breakdown — fixed, variable, employer contributions, and benefits?"** Most companies will send it; a refusal is informative.

- **"Is the bonus contractual or discretionary, and what did it actually pay out for this team in the last two years?"** A discretionary bonus is a hope. A 20% target that paid 40% twice is a 8% bonus with extra paperwork.
- **"What is the employer pension or provident fund contribution?"** In some countries this is 3% and in others 15%, and it is real money.
- **"Does the health cover include family, and what is the sum insured?"** In markets without public healthcare this is one of the largest non-cash items in the package, easily worth more than a 5% salary difference to somebody with dependants or a parent to insure.
- **"What is the notice period, in probation and after?"**
- **"Is there a joining bonus, and is it repayable if I leave within a year?"** Usually yes, and usually pro-rated. Read that clause.

The number to write down

For each offer, compute one figure and one list.

Guaranteed annual cash — base plus any contractual fixed allowances, before tax. Then, separately: expected variable using the historic payout rate rather than target, employer contributions, and the monetary value of benefits you would otherwise buy.

Keeping guaranteed cash separate from everything else is the discipline that matters, because guaranteed cash is what pays rent in a bad quarter.

Get it in writing before you resign

Verbal offers are withdrawn — after a hiring freeze, a reorganisation, a failed background check, or a manager changing their mind. It happens often enough that the rule is absolute: written offer, signed, with the start date, before you hand in your notice.

If you are asked to resign first "to show commitment", decline. No legitimate employer requires it.

The honest limitation

Tax turns gross into net and it does so differently in every country, in every regime within a country, and with every deduction you personally qualify for. This course cannot compute your net pay and neither can a model, confidently as it will try.

What you can do is find your own country's official tax calculator — most revenue authorities publish one, free — and run both offers through it before deciding. Two offers with the same gross can differ by a month's salary once allowances, regimes and deductions are applied, and it takes ten minutes to find out.

BEFORE YOU MOVE ON

A candidate holds two offers: ₹18,00,000 CTC at Company A, and a base salary of ₹15,60,000 plus a contractual 5% bonus at Company B. What is the first thing to establish?

A. Which company has the better long-term reputation, since the figures are close enough to be a tie.

B. Whether Company B will match the ₹18,00,000, since it is the higher figure.

C. The breakdown of A's CTC — how much is fixed cash, how much is employer contributions, and what the variable component has actually paid out — because A's guaranteed cash may be below B's.

D. The equity component at each company, since that is where the real upside sits.

62 · 10 min

Options, RSUs, and why most equity is worth nothing

THE ONE THING TO KEEP

An option is only worth the share value minus its strike, and the liquidation preference and the ninety-day exercise window decide most real outcomes long before the share price does.

Start from the base rate

Most start-ups do not produce a payout for employees. Most employee equity in private companies ends up worth zero, and this is the ordinary outcome rather than the sad exception.

That is not an argument against ever taking equity. It is the reason for one rule: **do not accept below-market cash for equity unless you can afford for the equity to be worth nothing.** If the grant is a lottery ticket you can afford, take it happily. If it is load-bearing for your rent, it is not compensation.

The four instruments

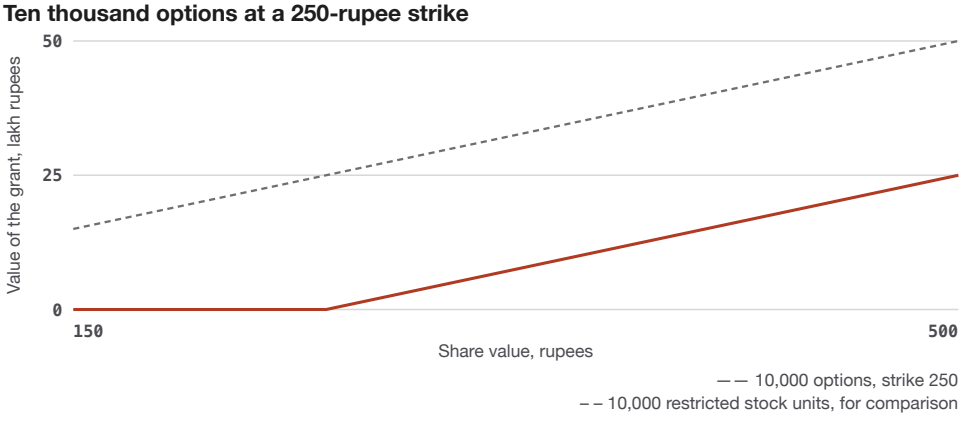
- **Share options (ISOs and NSOs in the US, ESOPs in India and much of Asia).** The right to buy shares at a fixed **strike price**. You pay to convert them.
- **Restricted stock units (RSUs).** Shares given to you when they vest. No purchase, and at a large public company they are close to cash on a delay.
- **Phantom shares and share appreciation rights.** A cash payment tracking share value, with no shares involved. Common where issuing real shares is legally awkward.
- **Direct shares**, occasionally, in very early companies.

The arithmetic

An option's value today is:

```
(current share value − strike price) * number of options
```

10,000 options at a strike of ₹250, when the company's current fair value per share is ₹250, are worth **zero today**. They are worth something only if the value rises. People hear "10,000 shares" and imagine a number; the strike is half the equation and is frequently omitted from the conversation.



At today's fair value of 250 the options are worth zero, and they stay at zero until the share value passes the strike. The strike is half the equation and is the half most often left out of the conversation.

Vesting is the schedule on which they become yours. Four years with a one-year cliff is the common shape: nothing for twelve months, then a quarter, then monthly or quarterly. Leave at month eleven and you have nothing.

Dilution is the reduction in your percentage every time new shares are issued in a funding round. Your number of shares does not change; the denominator grows. Two more rounds can halve a percentage.

The two clauses that decide the outcome

Liquidation preference. Investors typically get their money back before common shareholders get anything. If a company has raised $50 million with

a 1× preference and sells for $60 million, the investors take $50 million and $10 million is divided among everybody else. The press release says the company sold for sixty million dollars. Employees discover their share is small, and they were never told the preference stack.

Ask: *"What is the total liquidation preference, and is any of it participating?"*

The post-termination exercise window. In most option plans you have **90 days after leaving** to buy your vested options or lose them. If you have 10,000 vested options at a ₹250 strike, exercising costs ₹25,00,000 in cash, and in many countries you owe tax on the paper gain in the same year, for shares you cannot sell. This is how people leave a company having vested four years of equity and walk away with nothing, and it is entirely legal and disclosed in a document nobody read.

Some companies extend the window to seven or ten years. It is worth asking, and the answer tells you a great deal about how the company thinks about the people who leave.

Tax, in outline only

Genuinely differs by country and you should take advice. The shape:

- **India:** ESOPs are taxed as a perquisite at exercise, on the difference between fair market value and strike, whether or not you can sell. Eligible start-ups have a deferral. This catches people badly.
- **United States:** incentive stock options can trigger alternative minimum tax at exercise; RSUs are taxed as income at vest.
- **Most places:** a second tax event on eventual sale.

The pattern to notice is that tax is often due **before** any money arrives. Plan for that or do not exercise.

The four questions to ask

1. How many shares are outstanding on a fully diluted basis? (Without this you cannot compute a percentage, and a number of shares alone is meaningless.)

2. What is the strike price and the most recent valuation per share?
3. What is the total liquidation preference?
4. How long do I have to exercise after leaving?

A company that will not answer the first has told you the grant cannot be valued, and an unvaluable grant should be treated as worth zero in your comparison. Some genuinely well-run private companies decline on confidentiality grounds; you still cannot value it, and the arithmetic does not care about their reasons.

The honest limitation

Public-company RSUs are the exception to almost everything above: they are close to deferred cash with market risk, and they are straightforward to value. Nearly all the complexity here belongs to private companies, and nearly all the disappointment does too.

BEFORE YOU MOVE ON

A start-up has raised $50m with a 1x non-participating liquidation preference and is acquired for $60m. An employee holds 0.5% of common shares. What do they receive, roughly, before tax?

A. About $300,000, being 0.5% of the $60m sale price.

B. About $50,000, because investors recover their $50m first and only the remaining $10m is divided among common shareholders.

C. Nothing, because a liquidation preference means common shareholders are paid only if the sale exceeds twice the amount raised.

D. About $300,000, but only after a further four years of vesting following the acquisition.

63 · 10 min

Notice, bonds, IP and the clauses that follow you

THE ONE THING TO KEEP

Contract clauses vary enormously by jurisdiction and the only moment you can change one is before signing, so read for notice, bonds, IP assignment and outside-work restrictions and ask for carve-outs in writing.

The only moment you have leverage

Between the offer and your signature, a company that has spent six weeks choosing you would rather amend a clause than restart. After you sign, you are asking a favour. Read the contract in that window, once, properly. It takes forty minutes.

What follows is the shape of the common clauses and how they differ between countries. **It is not legal advice, employment law differs by jurisdiction, and for anything unusual or expensive you want an actual lawyer** — many bar associations and legal-aid clinics offer free or low-cost initial consultations.

Notice period

The gap between resigning and leaving. Two weeks to a month in much of the US and Europe at junior levels; **60 to 90 days is standard in Indian technology employment** and it is a real cost, because a company that wants somebody in three weeks will not wait three months, and you will lose offers to it.

Look for: notice during probation (usually much shorter), whether the employer can require **garden leave** — paid but not working, which protects them and freezes you — and whether there is a **buyout** option letting you pay

to leave earlier. Buyouts are common in India and the new employer sometimes covers them; ask.

Training bonds and service agreements

Common in Indian IT services and in some Gulf employment: you agree to stay for a period, or repay a sum, usually framed against training given.

The legal position is contested and jurisdiction-specific. In India, Section 27 of the Indian Contract Act voids agreements in restraint of trade, and courts have generally been willing to enforce a bond only to the extent of the employer's genuine, demonstrable costs — not as a penalty, and not where the amount is unconscionable. That is a general shape, decided case by case, and not a promise about your contract.

What to do before signing: ask what the sum covers, get the figure and the period in writing, and be clear that "we will hold your original certificates" is a practice you should refuse. Retaining original educational documents is not a legitimate security and in several jurisdictions is unlawful.

Non-compete

Highly jurisdictional, and the differences are stark:

- **India:** post-employment non-competes are generally void under Section 27. Confidentiality and non-solicitation clauses are treated differently and are more likely to hold.
- **California:** non-competes are void by statute, which is a substantial part of why Silicon Valley's labour market works as it does.
- **Germany:** enforceable only if the employer pays you at least 50% of your previous compensation for the duration. A non-compete with no compensation is not binding.
- **United Kingdom:** enforceable if reasonable in scope and duration, typically three to twelve months, and courts do strike down overreach.
- **United States generally:** the Federal Trade Commission's attempt at a national ban was blocked in court in 2024 and the position remains unsettled and state-by-state.

The practical point is that a clause existing in your contract does not mean it would be enforced, and that a clause you cannot afford to litigate constrains you regardless. Both things are true at once.

Intellectual property assignment

Almost every technology contract assigns to the employer the intellectual property in what you create during employment. The variation is in the edges, and the edges matter to anybody with a side project or an open-source habit.

Watch for language covering anything created "during the term of employment" without limiting it to work using company time, equipment or subject matter. Some jurisdictions limit this by statute — California's Labor Code §2870 protects inventions developed entirely on your own time without employer resources and unrelated to their business — and many do not.

What to do: list your existing projects in an annexe as **prior inventions** excluded from assignment, and ask for a written carve-out for personal open-source work. Reasonable employers agree. The request is normal and the moment to make it is before signing.

Moonlighting

Clauses restricting outside work have tightened considerably, particularly in India since 2022, where several large employers made dismissals over dual employment. Some contracts prohibit any paid outside work; some require written permission; some prohibit only competing work.

If you intend to freelance, teach, or take a second contract, read this clause specifically and get any exception in writing. "My manager said it was fine" survives exactly as long as your manager does.

Probation

Usually three to six months with shorter notice on both sides and sometimes reduced benefits. Confirm what happens at the end — is confirmation automatic, is there a review, can it be extended — because an extended probation is a signal that arrives too late to act on.

The one habit

Read the whole thing before signing, mark anything you do not understand, and ask. Asking three clear questions about a contract has never cost anybody an offer, and every experienced employer expects it.

BEFORE YOU MOVE ON

An offer contract assigns to the employer all intellectual property created "during the term of employment", with no limitation. The candidate maintains a popular open-source library. What is the right move?

A. Sign, since employers never enforce these clauses against personal open-source work.

B. Sign and stop contributing to the library while employed there.

C. List the library in an annexe as an excluded prior invention and ask for a written carve-out for personal work done on their own time and equipment, before signing.

D. Sign and rely on the local statute that protects work done on personal time.

64 · 9 min

Employee, contractor, or an employer of record

THE ONE THING TO KEEP

Employment carries 25% to 40% of salary in contributions and protections, so a contract at the same gross is a pay cut, and an employer of record gives you local employment status without the client needing an entity.

Three arrangements that look similar and are not

Employment. You are on the payroll. The employer withholds tax, contributes to pension or provident fund, owes you notice, leave, and in most countries severance and protection from unfair dismissal.

Independent contracting. You invoice. You handle your own tax, your own insurance, your own pension, and you have whatever notice your contract says — frequently a week or nothing. Higher gross, fewer protections.

Employer of record. A third company — Deel, Remote, Papaya, Velocity Global and others — legally employs you in your country and bills the company you actually work for. You get local employment status, a local contract, statutory benefits and a payslip. Your day-to-day is with the client company; your legal employer is the EOR.

The gross figures for these three are not comparable, and treating them as comparable is the most expensive mistake in this module.

What employment is actually worth

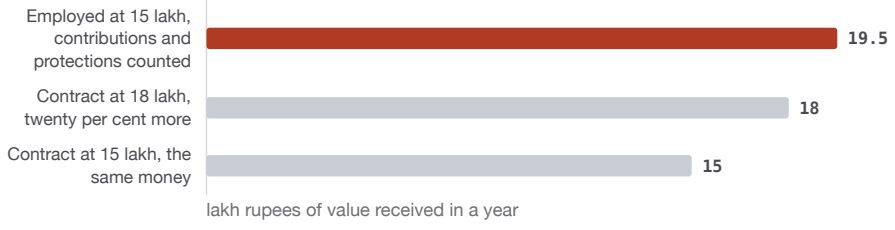
Before deciding a contract rate is better, count what you stop receiving. Depending on country, employment typically includes:

- Employer pension or provident fund contributions — commonly 3% to 15% of salary.

- Paid leave, public holidays, and sick pay.
- Health insurance, where the employer provides it.
- Notice, redundancy pay and protection from arbitrary dismissal.
- Parental leave.
- The employer's half of social security or payroll taxes.

Added together these are frequently 25% to 40% of salary in value. A contract at "the same money" is a large pay cut, and a contract at 20% more is roughly break-even before you account for unpaid weeks between engagements.

The same gross, three ways



Employment carries roughly 25 to 40 per cent of salary in contributions and protections. A contract at the same money is a pay cut, and a contract at twenty per cent more is about break-even before the unpaid weeks between engagements.

Misclassification, and why companies care

Calling somebody a contractor when the working relationship is employment is unlawful in most countries, and the tests are broadly similar even where the statutes are not: who controls how and when the work is done, whether you can send a substitute, whether you use their equipment, whether you work for others, how long the relationship has run, and whether you are integrated into their organisation.

The consequences fall mostly on the company — back taxes, penalties, and re-classification — which is why large employers are cautious and why an EOR exists at all. But it can also fall on you: unpaid tax and contributions are yours, and in some regimes the assessment lands years later.

If you sit in their standup daily, use their laptop, take their annual leave process, have a manager and work only for them, you are doing a job that most legal tests would call employment, whatever the contract says.

When each one is right

Take employment when you want stability, benefits, and legal protection, and when you are early enough that mentorship matters more than rate. This is nearly everybody's first job and should be.

Take a contract when the rate genuinely compensates for the loss — the freelancing module has the arithmetic — when you want several clients, or when it is the only structure available for work you want.

Take an EOR when a foreign company wants you and has no entity in your country. This is usually the best available option in that situation. Two things to check: the EOR's fee is paid by the client and should not be deducted from your salary, and your notice, leave and severance come from **your** country's law, not from the client's. Ask to see the local employment contract before accepting, not just the offer letter.

The questions to ask when an arrangement is not plain employment

- Who is my legal employer, and in which country?
- Which country's employment law governs this, and which courts?
- What notice applies, from each side?
- Who pays social contributions, and to which system?
- Am I covered for injury or illness, and by whom?
- What happens if the client ends the arrangement — do I get notice, or does it end that day?

An offer that cannot answer these has not been thought through by the employer either, and you will discover the gap at the worst moment.

The honest limitation

Contractor status is sometimes the only way to work for the employer you want, particularly across borders, and telling somebody to refuse it on principle ignores the market they are actually in. The point is not to refuse; it is to price it. If you take a contract, price it as a contract, keep three to six months

of expenses in reserve, and buy the protections you gave up — health cover and income protection — deliberately, rather than discovering their absence.

BEFORE YOU MOVE ON

A foreign company offers a candidate the "same money" as their local salaried job, on an independent contract, with the same daily hours, their manager's standup and the company's laptop. What is the most accurate assessment?

A. It is a straightforward improvement, since the gross figure is unchanged and remote work is more flexible.

B. It is a material pay cut once lost employer contributions, paid leave and notice are counted, and the working pattern described would fail most misclassification tests — so both the pricing and the structure need addressing, with an employer of record as the usual fix.

C. It is illegal and the candidate should decline outright.

D. It is fine provided the candidate registers as a business and pays their own tax correctly.

65 · 9 min

Getting paid by a company in another country

THE ONE THING TO KEEP

Foreign pay is reduced by an exchange spread nobody itemises, taxed where you are resident rather than where the client is, and usually benchmarked to your city — so compare it against your real local alternative, not against the employer's home market.

Four things change at once

When your employer or client is abroad, four things move together: how the money arrives, what it is worth after conversion, who taxes it, and how your

pay is set. People notice the first and are surprised by the other three.

Getting the money in

The routes, cheapest first for most corridors:

- **Wise** and similar platforms, which convert at close to the mid-market rate for a stated fee, typically well under 1%.
- **Payoneer**, common where clients pay into local receiving accounts.
- **Direct bank transfer (SWIFT)**, the default and usually the worst: a fixed fee at each end plus an exchange spread that banks do not itemise.

The spread is the cost people miss because it is invisible. A bank taking 2.5% against the mid-market rate on $3,000 a month costs about **$900 a year**, quietly, without a line item. The same transfers through a platform charging 0.6% cost roughly $216. That difference is a laptop, every year, for a change you make once.

Two practical rules. **Compare the amount that lands, not the fee** — a service advertising "zero fees" usually takes it in the rate. And check what your own country requires for inbound foreign payments: in India, for example, banks issue a foreign inward remittance advice for service exports, and you want those records from the first payment rather than reconstructed later.

Tax, in outline, because it cannot be more than that here

Two ideas do most of the work.

Tax residence. You are generally taxed by the country you are resident in, not the country your client is in. Residence usually turns on days present — the 183-day figure is a common threshold and a rule of thumb, not a universal rule, and several countries add tests about your home, family and centre of economic interest.

Double taxation treaties. Most country pairs have one, so the same income is not taxed twice; relief comes as a credit or an exemption. This is why "will I be taxed twice" almost always has the answer "no, but you must file correctly to prove it."

Beyond that, the specifics are genuinely complicated and country-specific: whether service exports attract sales tax or VAT in your country, whether a presumptive or simplified regime applies to independent professionals, whether you must register a business, what records you must keep and for how long. This course cannot answer any of that for you, and a model asked about it will produce a confident answer that may be two budget cycles out of date.

The proportionate response: read your own revenue authority's guidance for freelancers and exporters of services — every major one publishes it free — and once your foreign income is a meaningful share of your living, pay an accountant for one session. In most markets that costs less than a single month's mistake.

Working from a country you are not resident in

Spending three months working from another country on a tourist visa is common and is usually a breach of that country's immigration rules, and it can create tax obligations for you and a permanent-establishment risk for your employer. Employers increasingly ask where you are physically working and mean it. If you plan to move around, say so before signing rather than after.

How they set your pay

Companies take one of three positions, and asking which is a normal question:

- **Location-adjusted bands.** Most common. Your pay is benchmarked to your city, which is why the same role pays differently in Bengaluru, Warsaw and Berlin.
- **Region-adjusted.** Coarser tiers, often three or four.
- **A single global band.** Rare, loudly advertised where it exists, and the reason those roles attract thousands of applicants.

There is a real trade here and it is worth naming honestly. A location-adjusted offer from a foreign company is usually still well above your local market, be-

cause it is benchmarked against local employers, not against nothing. Comparing it to the San Francisco figure makes it look like exploitation; comparing it to the alternative you actually have makes it look like a good job. Both comparisons are available and only one of them is a decision.

Currency risk, which is real

If you earn in dollars and spend in naira, rupees, reais or zloty, your income moves with the exchange rate and you did not choose that exposure. Two defences that cost nothing: keep a buffer of three to six months of expenses so a bad quarter does not force a bad decision, and convert on a regular schedule rather than trying to time the rate, which nobody does successfully.

The honest limitation

Cross-border arrangements concentrate risk on the person with the least power, which is you. The client can end a contract with a week's notice from a jurisdiction where suing them is impractical. Price that in, keep the reserve, keep your own records of every invoice and payment, and do not let a single client become your entire income for longer than you can help.

BEFORE YOU MOVE ON

A freelancer receives $3,000 a month through their bank, which advertises no transfer fee. What is the most likely reality?

A. There is genuinely no cost, since the bank recovers it from the sending institution.

B. The cost is taken in the exchange rate rather than as a fee — a spread of a couple of per cent against the mid-market rate is roughly $900 a year on this volume, invisible on the statement.

C. The cost appears later as a tax liability on the converted amount.

D. The fee is charged to the client, so it reduces their cost rather than the freelancer's income.

66 · 9 min

The shape of the routes, and why the rules keep moving

THE ONE THING TO KEEP

Immigration rules move constantly and only official sources are current; the routes that reward planning are points-based systems and the internal transfer, and no legitimate party ever charges a candidate for a job offer.

Read this as a map, not as advice

Immigration rules change every year, sometimes twice. Thresholds rise, quotas close, categories are added and withdrawn. Anything specific written here may be out of date by the time you read it, and a model asked about visa rules will answer confidently from a training cut-off that predates the current rules.

Everything below is the **shape** of the routes so you know what to search for. The authoritative sources are free and official: gov.uk, uscis.gov, canada.ca, make-it-in-germany.de, ind.nl, immi.homeaffairs.gov.au and your destination's equivalent. Use those, and for anything complicated use a regulated adviser — in the UK an OISC-registered adviser or solicitor, in the US an immigration attorney, in Canada an RCIC.

The five routes

1. Employer-sponsored work visa. The company must be licensed to sponsor and the role must meet a salary and skill threshold.

- **United States, H-1B:** an annual cap with a lottery. Registration in March, selection by chance, employment starting in October. You cannot improve your odds and neither can your employer, which makes it the least plannable route in the world for something so consequential.
- **United Kingdom, Skilled Worker:** requires a licensed sponsor and meeting the general salary floor and the occupation-specific going rate.

The general threshold was raised substantially in 2024 and has continued to move — check the current figure, do not assume.

- **Germany, EU Blue Card:** a recognised degree or equivalent experience and a salary above a threshold that is reset annually, with a lower threshold for shortage occupations including IT.
- **Netherlands, highly skilled migrant:** a recognised sponsor and an age-banded salary threshold; administratively one of the simpler routes in Europe.

2. Points-based, no employer needed. Canada's Express Entry scores age, education, language test results and experience, with category-based draws that have included STEM occupations. Australia runs a comparable skilled list. These reward preparation — a higher language test score is worth real points, and the test costs a few hundred dollars against a life change.

3. Intra-company transfer. The most underrated route in this lesson. Join a multinational's office in your own country, do well for one to two years, and transfer internally. There is no lottery, the employer knows you, and the process is routine. A large share of the people you meet abroad arrived this way, and almost nobody plans for it deliberately.

4. Study, then work. A master's degree followed by a post-study work permit: the UK Graduate Route, Canada's post-graduation work permit, Germany's eighteen-month job-seeking period, the Netherlands' orientation year. Expensive and slow, and it converts a visa problem into a debt problem — be clear-eyed about which of the two you would rather have.

5. Job-seeker and freelance permits. Germany's Opportunity Card, and freelance or self-employment visas in several countries. Narrower, but real.

The costs nobody quotes

Ask the employer, in writing, which of these they cover, because practice varies from everything to nothing:

- Government application fees, per person, including dependants.

- Health surcharges — the UK charges an annual immigration health surcharge per person for the full visa duration, payable up front, which for a family is a large four-figure sum before you land.
- Legal fees, relocation, initial accommodation, and the deposit on a rental in a country where you have no credit history.
- The flights, and the month of double rent.

The trade you are accepting

A sponsored visa ties your right to remain to your employment. That changes the relationship: leaving is harder, negotiating is harder, and a redundancy becomes an immigration emergency with a short window to find a new sponsor. Some people find this entirely acceptable and it is a reasonable trade for what they gain. It is not a small one, and it should be a decision rather than a discovery.

One warning, because it costs people their savings

Nobody legitimate sells you a job offer. Paid "consultants" who guarantee employment abroad, ask for money to secure a work permit before any employer exists, or charge for a place on a "sponsored programme" are running the most common scam in this field, and the targets are exactly the people this course is written for. Employers pay recruiters; candidates do not pay for jobs. If money flows from you towards a job offer, stop.

BEFORE YOU MOVE ON

A candidate in Lagos wants to work in Europe within three years and has an offer from the local office of a European multinational. Which strategy has the most controllable path?

A. Decline it and apply directly to European companies advertising visa sponsorship, since that is the direct route.
B. Take the role, perform well, and pursue an internal transfer to a European office after one to two years, while also preparing points-based options such as language tests.
C. Take the role and rely on the US H-1B lottery, entering each March until selected.
D. Enrol in a European master's degree, which guarantees a post-study work permit.

67 · 10 min

Two offers, compared with arithmetic

THE ONE THING TO KEEP

Compare guaranteed cash first, then expected cash using historic payout rates, then contributions, avoided spending and the costs of the job — and enter equity in a private company at zero with the upside noted separately.

Do the sum on paper

Two offers feel like a choice between two vibes. They are mostly a choice between two spreadsheets, and the spreadsheet takes twenty minutes.

Here is one worked all the way through. The currency is rupees and the method is currency-agnostic; substitute your own.

The two offers

A — product start-up, fully remote. Quoted as ₹20,00,000 CTC: fixed cash ₹15,60,000; employer provident fund ₹86,400; gratuity accrual ₹75,000; insurance for self ₹18,000; variable at target ₹2,60,000. Six thousand share options at a strike of ₹180, current fair value ₹180.

B — global capability centre. Quoted as ₹18,50,000 CTC: fixed cash ₹14,04,000; employer provident fund ₹86,400; gratuity accrual ₹67,500; family insurance ₹96,000; contractual bonus of 8% of fixed, ₹1,12,320. Three days a week in the office, an hour each way.

A looks ₹1,50,000 better. Now do the work.

Step 1 — guaranteed cash

The money that arrives whatever happens.

```
A: 15,60,000
B: 14,04,000 + 1,12,320 (contractual)  = 15,16,320
```

Nearly identical. The headline gap has already almost vanished, because A's extra was mostly a target.

Step 2 — expected cash

Ask what the variable actually paid. Suppose A's team has paid 50% of target for two years:

```
A: 15,60,000 + 1,30,000 = 16,90,000
B: 15,16,320
```

A is ahead by about ₹1,74,000, and roughly ₹1,30,000 of that is contingent on a payout history you should verify rather than assume.

Step 3 — employer contributions

```
A: 86,400 + 75,000 = 1,61,400
B: 86,400 + 67,500 = 1,53,900
```

Real money, close to equal, and gratuity only vests after a qualifying period — five years in India — so treat it as long-term rather than annual.

Step 4 — benefits you would otherwise buy

A insures you. B insures your family. If comparable family cover would cost you ₹45,000 a year, B is **₹45,000 better** in avoided spending. For somebody with dependants or a parent to cover, this line is often larger than the salary difference and is the one people leave out.

Step 5 — costs of the job itself

B requires three days on site, an hour each way:

```
transport: ~3,500/month  = 42,000/year
time: 6 hours/week x 45 weeks = 270 hours
```

Money is easy to net off. The 270 hours are not money, and you should not convert them into money — write them down as hours and decide what they are worth to you, which depends entirely on your life.

Step 6 — equity

Six thousand options at a ₹180 strike when the fair value is ₹180 are worth **zero today**. They may be worth a great deal later. The honest entry in the comparison is: *expected value zero, with an unquantified upside I could not value because I do not have the share count or the preference stack.*

Never let equity break a tie you would otherwise decide the other way. It is a bonus if it lands.

The result

```
              A            B
guaranteed    15,60,000    15,16,320
expected      16,90,000    15,16,320
employer      1,61,400     1,53,900
benefits      —            +45,000
job costs     —            −42,000, −270 hours
equity        0 (upside)   0
```

A is ahead by roughly ₹1,20,000 of expected cash and 270 hours. B is ahead on certainty and family cover. That is a real decision rather than a difference in headline numbers, and you can now hold it in your head.

The column you cannot compute

Everything above is the easy half. The half that decides how the next two years actually go:

- **Who you would report to.** The largest single factor in whether a job is good, and impossible to quantify. Weight what you saw: did they listen in the interview, could they describe the team's failures, did they answer "why is this role open" without hedging?
- **What you would learn**, and what job it makes possible afterwards. Early in a career this frequently outweighs a 10% pay difference. Later, with dependants, it frequently does not. Both are correct positions and only you know which one you are in.
- **The company's stability.** A start-up's cash runway is a fair question and a good one will answer it.

The regret test

When the arithmetic is close, ask which rejection you would rather explain to yourself in a year. Turning down more money for a manager you trusted is a decision people rarely regret. Turning down a job you wanted for eight per cent more is a decision people regret constantly.

The honest limitation

You are choosing under bad information, and no amount of care fixes that. The interview showed you three people for four hours; the reality is forty people for two years. Do the arithmetic because it removes the errors you can remove, then accept that the residual uncertainty is genuinely large and that this is what everybody else is doing too.

BEFORE YOU MOVE ON

Two offers are within 2% of each other on expected cash. One insures the candidate's family, the other insures only the candidate; the second requires a commute costing 270 hours a year. What has the arithmetic actually established?

A. That the offers are equivalent and the decision should be made on which company is more prestigious.

B. That family cover and 270 hours are the real difference, and the remaining decision rests on the uncomputable column — manager, learning and stability — which should now be weighed deliberately rather than by feel.

C. That the candidate should ask both employers to improve their offers until one clearly wins on cash.

D. That the higher-cash offer wins, since benefits and commuting are personal preferences rather than compensation.

68 · 9 min

Raises, promotions and the market rate

THE ONE THING TO KEEP

Raises come from a small annual pool, promotions move your band, and job changes reset you to market — so if the band is the constraint, no amount of performance evidence will fix it internally.

Three mechanisms, three different sizes

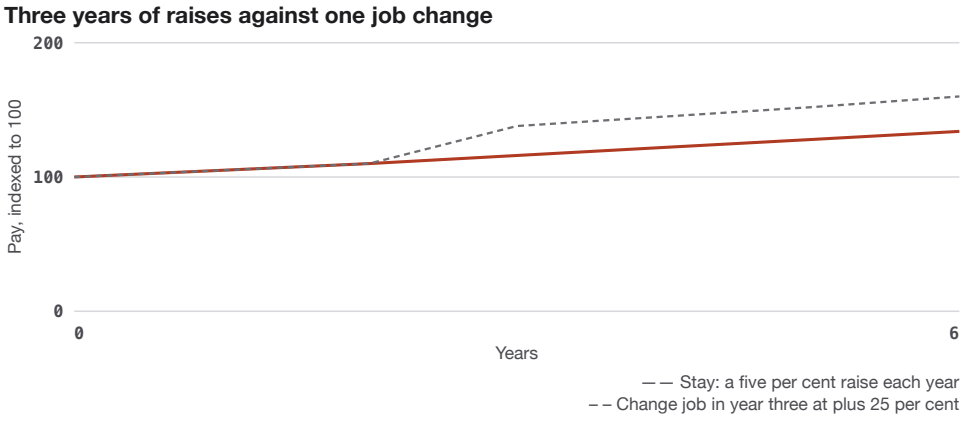
Pay moves in exactly three ways, and they are not interchangeable.

The annual raise. A company sets a merit pool — commonly 3% to 8% of payroll in ordinary conditions, sometimes zero — and managers divide it across their team. Even an outstanding year usually means a share of a small pool. Your manager is not being unhelpful; they are dividing a fixed amount.

The promotion. Moves you into a new band, and typically brings 8% to 15% at once. This is the only large internal step, and it is why the level lesson told you to negotiate the level rather than the number.

The job change. Resets you to the current market. Typically 15% to 30%, and considerably more when you also change segment — services firm to product company, or one country's market to another's.

The sizes are the argument. Three years of good raises might be 18% compounded; one job change can be 25% in a day. This is not a recommendation to leave, and the next section explains why.



A merit pool of three to eight per cent compounds to about eighteen per cent over three years; a change of job resets to market in a day. Not a recommendation to leave: the lesson goes on to what staying is worth.

Salary compression, which is a mechanism rather than an insult

A company hires at market rate. Market rates rise. Existing staff receive raises from a pool that is smaller than the rise in market rate. Within three years, new joiners doing the same job earn more than people who have been there five years and are better at it.

Every large employer has this problem and very few fix it retrospectively, because fixing it costs a great deal all at once. Knowing the mechanism is worth more than resenting it: it tells you that loyalty is not rewarded automatically, that the correction is either a promotion or an external offer, and that your manager probably agrees with you and cannot do much.

Getting a raise, practically

Know the cycle. Most companies decide pay one to three months before it is announced, and by then the pool is allocated. Ask your manager directly: *"When are compensation decisions made, and when should I have made my case?"* The number of people who ask is small and the answer is not secret.

Build the case over months, not in the conversation. Keep a running file of what you delivered with numbers attached — the report that replaced four hours a week of manual work, the pipeline failure rate that fell from weekly to none, the analysis that changed a decision. Managers write these

cases for their own managers and they are usually assembling them from memory. Handing over a page of evidence makes you the easy candidate.

Separate the two asks. A raise and a promotion run on different machinery: one is a pool, one is a level change with a committee or a calibration meeting. Ask which you are eligible for and what specifically is missing, and get the answer as a behaviour rather than a feeling: *"What would I need to be doing consistently for you to put me forward?"*

Ask early and ask specifically. "I would like to discuss my compensation before the cycle closes; here is what I have delivered" lands very differently from "I feel underpaid".

The counter-offer

Your employer matches an outside offer to keep you. It is tempting and it is usually the wrong choice, for a reason more solid than the folklore: **the conditions that made you look are unchanged.** If you were leaving over the work, the manager, the ceiling or the commute, more money changes none of them, and you will be having the same thought in seven months with your bargaining chip spent.

You are also now known to have interviewed elsewhere, which affects how you are staffed and how you are considered when the next redundancy list is drawn up.

You will read confident statistics about most counter-offer acceptees leaving within a year. Those figures circulate widely and are poorly sourced; treat them as folklore. The structural argument above stands on its own and does not need them.

The exception worth naming: if you were genuinely happy and only underpaid, and the counter-offer fixes the band rather than adding a one-off bonus, staying can be entirely rational. Ask which of the two it is, in writing.

What staying is actually worth

The case for not changing job every eighteen months is real and rarely made:

- **Scope compounds.** Senior work requires knowing a system and an organisation deeply, and you cannot demonstrate that in your first six months anywhere.
- **Equity vests**, if you have any that is worth anything.
- **Reputation accrues** among people who will hire you later — most senior roles are filled by people somebody already trusts.
- **A CV of six jobs in seven years** raises a question you will be asked in every interview, and you will need a good answer each time.

The workable pattern for most people is two to four years per role early on, with a deliberate move when the learning stops rather than when the frustration peaks.

The honest limitation

Pay bands are set mostly by market, geography and employer type, and only marginally by how good you are. Excellent performance inside a low band produces a low-band raise, and no amount of evidence changes a band that was set by the company's business model.

If you are near the top of your band and the band is the problem, the fix is a different employer category, a different city, or a different market — the terrain module, revisited with better information than you had the first time.

BEFORE YOU MOVE ON

An analyst is told they are a top performer but receives a 4% raise. They are near the top of their band. What follows from the mechanism?

A. Their manager undervalues them and the case should be made again with better evidence.

B. A 4% raise is a normal share of a merit pool, and being near the top of the band means further internal movement requires a promotion into a new band — or a different employer, since bands are set by market and business model rather than performance.

C. They should obtain an external offer and use it to secure a match.

D. Top performers are usually paid through bonus rather than base, so the bonus is where the recognition will appear.

CASE STUDY · MODULE 7

Ninety days to find twenty-five lakh

Priya, a data engineer in Pune, resigning after four years at a startup holding ten thousand vested share options.

She had thought of the grant as "ten thousand shares" for four years. Nobody had lied to her. The document said everything, and she read it properly for the first time in the week she resigned.

The numbers:

- Vested options: **10,000**, fully vested after four years with a one-year cliff.
- Strike price: **₹250** per share.
- Most recent valuation used for tax purposes: **₹410** per share.
- Post-termination exercise window: **90 days**.

So the paper gain was ₹16,00,000 and the cash required to realise any of it was ₹25,00,000. In India an ESOP is taxed as a perquisite at exercise, on the difference between fair market value and strike — so on top of the ₹25 lakh she would owe tax on ₹16 lakh of gain, in a year in which she could not sell a single share, because the company is private and there is no buyer.

She had ₹6,00,000 saved.

What she thought she had been paid

Four years earlier she had accepted ₹4,00,000 a year less than a competing offer from a bank, on the reasoning that the options made up the difference. That was the decision, and it was made before she knew what a strike price was.

The arithmetic of an option is not complicated. It is worth the current share value minus the strike, multiplied by the number of options — which means a grant issued at a strike equal to the then-current fair value is worth exactly zero on the day it is granted, and is worth something only if the value rises. Her grant had risen from ₹250 to ₹410 over four years, which sounds like a

good outcome and, once the exercise cost and the tax are placed beside it, is not obviously one.

There had also been two funding rounds in those four years. Her number of options never changed and her percentage roughly halved, because the denominator grew. Nobody had hidden this; dilution is what a funding round is.

The questions she asked too late

Four of them, and she asked them in week one of her notice period rather than at the offer stage four years earlier.

1. How many shares are outstanding on a fully diluted basis? — answered.
2. What is the strike and the latest valuation per share? — answered.
3. What is the total liquidation preference, and is any of it participating? — **declined, on confidentiality grounds.**
4. How long do I have to exercise after leaving? — ninety days, as written.

The third refusal was the important one. Without the preference stack she could not value the grant at all. Investors are paid before common shareholders, so a company that has raised heavily can sell for a headline number that leaves nothing for employees, and the press release will still say a large figure.

The decision

Borrow and exercise everything. ₹25 lakh of borrowed money and a tax bill, converted into an illiquid holding in a private company whose preference stack she was not permitted to see, with no timeline to any sale.

Let all ten thousand lapse. Four years of vesting worth nothing, and the certainty of watching a headline in three years and doing the arithmetic she chose not to do.

Something in between, sized to what she could lose without it changing her life.

She asked for the window to be extended — some plans allow seven or ten years, and asking is normal. The company declined, politely, and the fact that

it declined told her something about how it thought about people who leave.

She also, for the first time, ran the comparison she should have run four years earlier: guaranteed annual cash against guaranteed annual cash, with the equity entered at zero and the upside noted separately in a line of its own. On that arithmetic the bank had offered more every year for four years, and the difference compounded through every raise she had received since, because raises are percentages of a base.

STOP HERE

Before turning the page: what would you have done, and what would it have cost if you were wrong? Write it down. The point of the next section is to compare.

WHAT HAPPENED

She exercised 2,500 options for ₹6,25,000 and paid perquisite tax on ₹4,00,000 of notional gain, funded from savings she could afford to lose entirely. The remaining 7,500 lapsed. Two years later the company raised a down round at ₹300 per share with a one-times participating preference on roughly ₹300 crore of invested capital — which, had she exercised the full grant, would have left her having spent ₹25 lakh and a tax bill on a holding worth less than she paid. She now tells every junior engineer the same two things: ask the four questions at the offer stage rather than at the exit, and value private-company equity at zero when deciding what rent you can pay. The upside is a bonus if it lands and it is not compensation.

Why is the ninety-day exercise window, rather than the share price, the clause that decided this outcome?

Because it converts a paper holding into a cash demand at the worst possible moment — the point at which you are leaving and have no salary from the company. The share price only matters if you can hold the shares long enough for it to move, and the window is what determines whether you can. Four years of vesting can evaporate on a clause nobody reads, entirely legally and entirely as disclosed.

The company refused to disclose the liquidation preference. How should that refusal be treated in a comparison of offers?

As making the grant unvaluable, which means entering it at zero with the upside noted separately. Some genuinely well-run private companies decline for legitimate confidentiality reasons, and the arithmetic does not care about their reasons: without the preference stack and the fully diluted count you cannot compute what the grant is worth in any exit, so it cannot carry weight in a decision about salary.

What would asking the four questions four years earlier have changed?

Probably not the grant, and definitely the planning. She would have known the exercise cost was of the order of ₹25 lakh, that tax falls at exercise rather than at sale, and that she had ninety days from any departure. That turns a shock into a savings plan, and it might also have changed how much cash she accepted at the offer stage, because the rule is not to take below-market cash for equity you cannot afford to see go to zero.

MODULE 7 · TEST

Check what stuck

6 questions, one right answer each. This one is for you, not for a record: answers and the reason behind each are at the back. If two questions from the same module go wrong, re-read that module before the exam rather than after.

1 An offer letter says ₹18,00,000 CTC. Which figure should you compare against another offer?

A. The CTC, since it is what the employer actually spends on you

B. The CTC minus tax, since that is what reaches your account

C. The fixed cash before tax, excluding contributions and target variable pay

D. The CTC minus the insurance premium, which you would otherwise buy yourself

2 You hold 10,000 vested options at a ₹250 strike; the current fair value is ₹250. What are they worth today?

A. ₹25,00,000, which is the value of the underlying shares

B. Zero, because the value equals the share price minus the strike

C. ₹12,50,000, since options are conventionally valued at half the share price

D. It cannot be stated without knowing the total number of shares outstanding

3 You are offered a contract at the same gross figure as a salaried role you hold. What has changed?

A. Nothing material, since the gross is identical and tax is your own responsibility either way.
B. It is a modest improvement, because contractors can deduct business expenses
C. It is a large pay cut: employment carried 25% to 40% in contributions and protections.
D. It depends only on whether the contract is renewed at the end of its first twelve-month term.

4 A bank advertises zero fees on inbound foreign transfers of $3,000 a month. Why might that cost about $900 a year?

A. Because the receiving country levies a withholding charge on service exports
B. Because a monthly account fee applies once foreign currency is received
C. Because the exchange spread against the mid-market rate is not itemised
D. Because conversion happens on the day of arrival rather than at the rate on the invoice

5 You have side projects and the contract assigns everything created "during the term of employment". What is the move?

A. Sign, since such clauses are unenforceable against work done on your own equipment
B. Sign, and rely on your manager's assurance that personal projects are fine
C. Ask for a carve-out and list existing projects as excluded, before signing
D. Decline the offer, because a blanket assignment clause indicates a hostile employer

6 Three years of good annual raises compounded to about eighteen per cent. A single job change typically brings what, and why?

A. About the same, since new employers benchmark against your previous salary

B. Around five per cent, because a new joiner starts at the bottom of the band

C. Around fifteen to thirty per cent, because a change resets you to the current market

D. Nothing predictable, because pay depends entirely on how well an individual negotiates it

MODULE 8

Freelancing and contracting, as a business

The first paid job is covered in the module on getting seen. This module is what comes after it — the arithmetic that turns a salary into a rate, where a repeatable pipeline of clients comes from, how to write a scope that prevents the argument, how to get paid by a company whose finance department has never heard of you, and the honest conditions under which working for yourself is the wrong choice this year.

> **BY THE END OF THIS MODULE YOU CAN**
>
> Price your own work upwards from a target income and a realistic utilisation figure, write a proposal whose exclusions prevent scope creep, run a payment process that survives a slow client, and state the conditions under which you should take a salaried job instead.

69 · 9 min

Utilisation, and why a rate is not a salary divided by hours

THE ONE THING TO KEEP

Only half to two-thirds of a freelancer's hours are billable, and once lost employer contributions and unpaid risk are added, a sustainable rate is roughly two to two and a half times salary divided by working hours.

The mistake that ends most first years

Somebody earning ₹15,00,000 a year decides to freelance. They divide by 1,920 working hours, get about ₹780 an hour, add a bit for confidence, and quote ₹1,000. Twelve months later they have worked harder than they ever did as an employee and earned less.

The arithmetic below is why, and it is the single most useful page in this module.

You are not paid for every hour you work

Start with the year. Fifty-two weeks, minus four for holiday and illness, is forty-eight weeks at forty hours: **1,920 working hours**.

Of those, the hours you can actually invoice — your **utilisation** — are far fewer, because the rest go to work nobody pays for:

- Finding clients: outreach, calls, proposals that go nowhere.
- Scoping and estimating.
- Invoicing, chasing, bookkeeping, tax.
- Learning, which as an employee happened on someone else's time.
- Contract admin, tool costs, email.

Realistic figures, which agencies track closely because their business depends on them:

- **Established solo freelancer: 50% to 70% billable.**
- **First year: 25% to 40%**, because you are mostly selling.

At 60%, that is **1,150 billable hours a year**. Not 1,920.

What you have to replace

An employee on ₹15,00,000 of fixed pay costs their employer more than that, and receives more than that:

```
fixed salary                          15,00,000
employer provident fund + gratuity     1,60,000
insurance, equipment, software           50,000
                                      ----------
true cost of employing you            17,10,000
```

As a freelancer you also carry costs the employer used to absorb, and a premium for risk you now hold alone:

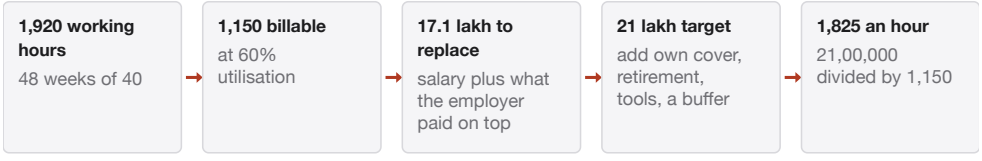
```
own health cover + income protection    80,000
own retirement contribution             90,000
laptop, software, accountant, phone     60,000
buffer for gaps and bad debt          1,60,000
                                      ----------
target annual revenue                 21,00,000
```

The rate

```
21,00,000 / 1,150 billable hours  =  ~1,825 per hour
```

Against the naive ₹780. **Roughly 2.3 times** the number people first reach for — and that multiple, somewhere between two and two and a half, holds across currencies and markets, which is why it is worth memorising rather than the figure.

From salary to rate, the honest way



Against the naive 780 an hour from dividing salary by working hours. The multiple, somewhere between two and two and a half, holds across currencies and markets, which is why it is worth memorising rather than the figure.

Run the same calculation with your own numbers before you quote anybody. It takes ten minutes and it is the difference between a business and an expensive hobby.

Cash flow is a separate problem from profit

You can be profitable on paper and unable to pay rent. Work done in March is invoiced on 31 March on 30-day terms, paid in late April if the client is prompt, and in June if the client is a large company with a procurement department.

Three consequences:

- **Keep three to six months of expenses in reserve** before you start, not as an aspiration afterwards.
- **Invoice on milestones**, not at the end. A three-month project paid entirely on completion is you lending the client money for three months.
- **Expect the first six months to earn less than your salary did.** People who plan for that survive it; people who do not, take the first bad contract they are offered in month four.

The pipeline paradox

When you are busy delivering, you are not selling. Three months later the work you did not sell fails to arrive, so you sell frantically, and three months after that you are busy again. This oscillation is the defining rhythm of a badly run freelance business and it is entirely avoidable.

The fix is boring and it works: **a fixed weekly slot for selling that you keep regardless of how busy you are.** Half a day, every week, forever. Two outreach messages, one follow-up with a past client, one piece of public work. Nothing more.

The honest limitation

Freelance income is more variable than almost anyone expects, and variance is worse than a lower average for most people's lives. A year of ₹18,00,000 arriving evenly is easier to live on than a year of ₹22,00,000 arriving in three lumps with a five-month gap.

If you have dependants, a loan, a visa tied to employment, or no reserve, that variance is a substantial risk and not a lifestyle preference. The last lesson in this module is about exactly that.

BEFORE YOU MOVE ON

An employee earning ₹15,00,000 wants to freelance at the same income. Why is ₹780 an hour — their salary divided by 1,920 working hours — badly wrong?

A. Because freelancers should always charge a premium for the flexibility they give clients.

B. Because roughly 40% of their hours will be unbillable selling and admin, and they must now fund the contributions, insurance and idle-time buffer their employer used to carry — which together push the sustainable rate to more than twice that figure.

C. Because clients expect round numbers and ₹780 signals inexperience.

D. Because taxes are higher for the self-employed, so the rate must be grossed up for tax.

70 · 9 min

Pricing the work, and raising the price

THE ONE THING TO KEEP

Set a floor from your own cost arithmetic, price fixed work with 20% to 50% contingency behind a paid discovery phase, and raise the rate on new enquiries rather than waiting for permission.

Four ways to charge, and what each one does to you

Hourly. Simple and universally understood. Two problems: you are penalised for being fast, so the better you get the less you earn per job, and the client watches the clock, which turns every conversation into a cost. Fine for open-ended or exploratory work.

Day rate. The same, in units nobody argues about. Better for contract-style engagements where you are effectively on the team for a period. Most contract markets run on day rates.

Fixed price per project. You quote a number for a defined outcome. You keep the gains when you are efficient, and you absorb the loss when your estimate was wrong — which it will be, early. Requires a real scope, which the next lesson covers.

Retainer. A fixed monthly amount for an agreed scope or a block of availability. The best of the four for cash flow and the hardest to sell first.

Most sustainable freelance businesses end up mixing fixed-price projects with one or two retainers, and use hourly only for work that genuinely cannot be specified.

Setting the first number

Two directions, and you need both.

Bottom-up, from the previous lesson: target revenue divided by realistic billable hours. This is your **floor**. Quoting below it is subsidising the client from your savings.

Top-down, from the market. Free sources: published rates on freelancing platforms for your skill and region; contract rates on job boards; asking two or three people who do what you do; and agency rates, which are typically 1.5 to 2 times what they pay the subcontractor, so an agency charging ₹4,000 an hour is paying somewhere near ₹2,000.

If your floor is above the market, the problem is not the price — it is either the market you are selling into or the value you are offering, and the geography section below is usually the answer.

Fixed price without losing money

Three rules.

Add contingency honestly. For work you have done before, 20%. For work with an unfamiliar system or an unfamiliar client, 30% to 50%. This is not padding; estimation error in software and data work is consistently in one direction and every experienced person prices for it.

Sell a paid discovery first. For anything substantial, phase one is a short paid piece — two to five days — producing a written assessment and a fixed quote for phase two. It protects you from quoting blind, it is genuinely useful to the client even if they stop there, and it converts at a high rate because they have already paid you once.

Never quote in the meeting. "I will send you a number tomorrow" costs nothing and has saved more freelancers than any other sentence in this module.

Value-based pricing, honestly

The idea: charge a share of the value created, not the hours spent. If your reporting work saves a client twenty hours a week of manual reconciliation, that

is roughly ₹10,00,000 a year, and charging ₹2,00,000 for it is defensible whether it took you four days or fourteen.

Where it genuinely works: when the client can quantify their own gain, when you have evidence from prior clients, and when you are talking to somebody who owns the budget and the outcome.

Where it does not: most first engagements, most small clients, and anywhere the buyer is a manager comparing you against a day rate they can benchmark. It is oversold by people who sell courses about pricing. Treat it as a technique available in a minority of engagements, not as the goal.

Geography is the largest lever

The same work, delivered by the same person, is priced against the client's market. A data pipeline built for a company in Berlin or Boston is worth several times what it is worth to a company in a low-cost market — for identical work.

This is the single biggest determinant of freelance income for somebody in India, Nigeria, the Philippines or Brazil, and it is worth being clear that it comes with costs: harder sales, longer trust-building, time-zone overlap, slower payment and currency risk. A sensible progression is local clients first to build proof, then regional, then international once you have references somebody abroad will believe.

Raising your rate

Nobody will do it for you. Two mechanics that work:

- **Raise on new clients first.** Quote the new number to the next enquiry. If they accept without pausing, you were too cheap. If three in a row accept instantly, raise again.
- **Give existing clients notice.** "From April my rate for new work will be X." Ten to twenty per cent a year is unremarkable. Some clients leave; in practice fewer than people fear, and the ones who leave over 15% were the difficult ones.

The discount that is never worth it

Working cheaply or free for "exposure", "a case study", or "lots of work coming later". The future work rarely arrives, and if it does it arrives at the price you set. A discount you must give — for a charity, for a first reference — should be shown on the invoice at full price with the discount stated as a line, so the anchor stays where you put it.

BEFORE YOU MOVE ON

A freelancer is asked to quote a fixed price for integrating a client's undocumented legacy system, which they have never seen. What is the best response?

A. Quote a fixed price with a 50% contingency, since the uncertainty is priced in.

B. Quote hourly instead, so the risk sits entirely with the client.

C. Propose a short paid discovery — a few days producing a written assessment and a fixed quote for the build — so the fixed price is set after the system has been seen.

D. Quote low to win the work, then raise a change request once the true complexity emerges.

---

71 · 9 min

A pipeline, after the first three clients

THE ONE THING TO KEEP

A repeatable pipeline comes from a sentence somebody can forward, referrals asked for specifically, agency subcontracting and a weekly selling slot — platforms are a ramp to your first references, not a business.

The first three came from people you knew

That route is covered in the module on getting seen, and it works. It also runs out. What follows is how a repeatable flow of work is built once you have ex-

hausted the people who already trust you.

Say what you do in one sentence

"Freelance data consultant" is unmemorable and unforwardable. Nobody hears it and thinks of a specific person with a specific problem.

The sentence that works names a **buyer**, a **problem** and an **outcome**:

> *"I fix broken reporting for e-commerce companies with under fifty staff — usually a warehouse, dbt models and a dashboard people actually trust, in about six weeks."*

That is narrower than feels comfortable, which is precisely why it works. Somebody who hears it can forward you to the right person without thinking. You can still take other work; the sentence is a door, not a fence.

The sources, ranked by how well they convert

1. Past colleagues and employers. Highest conversion of anything, by a distance. People who have seen you work do not need convincing. Tell every former colleague what you now do, once, without asking for anything.

2. Referrals from clients. Ask at the end of a good project, specifically: *"Do you know anyone else dealing with the reporting problem we just fixed?"* A general "let me know if you hear of anything" produces nothing.

3. Subcontracting from agencies and other freelancers. The most underrated source in this lesson. Agencies win more work than they can staff and regularly need a specialist for six weeks. You get steady work with no sales effort at a rate 30% to 50% below what they bill — which is a fair trade for them carrying the client relationship and the payment risk. Approach ten small agencies in your specialism and describe exactly what you can take off their hands.

4. Inbound from public work. Slow, compounding, and the highest quality when it arrives, because somebody who found you through a write-up has

already decided you know the subject. This is the freelancing payoff of the writing-in-public lesson.

5. Local business networks. Chambers of commerce, industry associations, trade groups. Unglamorous and full of companies with real problems and no data person.

6. Platforms. Useful, with the caveats below.

7. Cold outreach. Works when specific. A message naming something you noticed about their business, one concrete thing you would do, and no attachment. Ten thoughtful messages beat two hundred templated ones and take about the same time.

Platforms, with the numbers

Fee structures change, so verify before relying on any figure — but the shape is stable and it matters:

- **Upwork** charges freelancers a service fee on earnings, plus fees on the client side that reduce what clients are willing to offer.
- **Fiverr** takes a notably larger share, around a fifth.
- **Toptal** and similar curated networks screen hard, pay meaningfully better, and take their margin on the client side.
- **Contra**, **PeoplePerHour** and regional platforms sit in between.

Two honest things about them. The low end is a genuine race to the bottom, where you compete on price with people whose cost of living is lower than yours, and no amount of profile optimisation wins that race. And **you do not own the relationship** — the platform does, the terms usually forbid taking clients off it, and your income depends on an account that can be suspended.

The sensible use: platforms for the first handful of paid references when you have none, then a deliberate move to direct clients. Treat them as a starting ramp, not a business.

The weekly habit

Half a day a week, permanently:

- Two specific outreach messages.
- One reconnection with a past client or colleague, with no ask attached.
- One piece of public work — a write-up, an answer, a small tool.
- Update the pipeline list: who you spoke to, what you agreed, when you will follow up.

Follow-up is where most freelance work is actually won. A polite second message a week later converts a surprising share of silences, and almost nobody sends it.

The honest limitation

Selling is the job. Technical people consistently underestimate this and dislike it, and there is no arrangement in which good work markets itself early on. The first ten clients are much harder than the next fifty, the effort is front-loaded and mostly invisible, and this is the part where people quit — usually in month four, about three weeks before the pipeline they built in month two would have started producing.

BEFORE YOU MOVE ON

A freelancer has completed three projects for former colleagues and has no further work lined up. Which move is most likely to produce paid work within six weeks?

A. Optimise their profiles on two freelancing platforms and bid on twenty listings a week.

B. Approach small agencies in their specialism describing exactly what they can take off the agencies' hands, and ask each completed client for a named referral.

C. Build a professional website and wait for inbound enquiries to arrive.

D. Lower their rate by 30% to become competitive while rebuilding the pipeline.

72 · 9 min

Writing the thing that prevents the argument

THE ONE THING TO KEEP

A proposal exists to make the disagreement happen in week zero, and its most valuable sections are what is excluded, what acceptance means, and how many revisions are included.

Every bad freelance experience is a scope failure

Not a technical failure. Somebody expected something that somebody else did not intend to deliver, nobody wrote it down, and by the time the difference is visible there is money and pride attached to both readings.

A proposal is not a sales document. It is the artefact that makes the disagreement happen in week zero, cheaply, instead of in week eight, expensively.

The seven sections

Two pages, plain language, no design.

1. What I understand your problem to be. In *their* words, from the conversation. Three or four sentences. This section alone catches a startling number of mismatches, because clients read it and say "no, that is not really the issue".

2. What I will do. The approach, briefly.

3. What you will get. The deliverables, named as objects. Not "improved reporting" but: "a dbt project in your repository with twelve models and tests; three dashboards; a one-page handover document; a two-hour walkthrough."

4. What is not included. The most valuable paragraph in the document. Name the things a reasonable person might assume: ongoing maintenance,

training beyond the walkthrough, changes to source systems, mobile layouts, migrating historical data, support after handover.

5. What I need from you, and by when. Access, a named decision-maker, review turnaround. Write the consequence: *"Assumes read access to the warehouse by day two. Each week of delay moves delivery by a week."* This is not aggression; it is the only thing that prevents a delay caused by the client becoming a delay attributed to you.

6. Timeline and milestones, with payment attached to each.

7. Price, validity and how changes are handled. A quote should expire — thirty days is normal — so a client cannot accept in five months at last year's rate.

The seven sections, on two pages

1. What I understand your problem to be	in their words; catches a startling number of mismatches
2. What I will do	the approach, briefly
3. What you will get	deliverables named as objects: twelve models, three dashboards, a walkthrough
4. What is not included	maintenance, training, source-system changes, historical migration
5. What I need from you, and by when	access by day two; each week of delay moves delivery a week
6. Timeline and milestones	payment attached to each; acceptance defined; revisions numbered
7. Price, validity, how changes are handled	a quote expires in thirty days

The document exists to make the disagreement happen in week zero, cheaply, instead of in week eight. Section four is the most valuable paragraph in it.

Milestones that protect both sides

A shape that works for a six-week project:

```
Phase 1 — discovery and written assessment      30%  on start
Phase 2 — build, delivered for review           40%  on deliv-
ery
Phase 3 — revisions, handover, documentation    30%  on
acceptance
```

Two details do the work. **Acceptance must be defined** — "acceptance means the three dashboards match the agreed metric definitions on the test period; two rounds of revisions are included" — otherwise "done" is whatever the client feels on the day. And **revisions are numbered**. Unlimited revisions is not generosity; it is an unbounded liability, and it makes you resentful of a client who is behaving reasonably.

The change request, and the tone

Scope creep almost never arrives as a demand. It arrives as "while you are in there, could you also…", and each request is small and reasonable. Three of them is a week.

The move is always the same, and it is a yes:

> *"Yes, we can do that. It is outside the current scope, so I will send a short change note with the cost and the new date — probably about two days. Shall I?"*

You have agreed, priced it, and moved the date, in one sentence, without a confrontation. Clients respect this; what they resent is a freelancer who silently absorbs three changes and then delivers late.

Keep a written log of every change, even the ones you waive. "Included at no charge: two extra dashboards (worth ₹40,000)" on the final invoice is the most effective goodwill you can generate, and it prevents the free work from becoming the new baseline.

The kill clause

Both sides should be able to end it. A termination clause saying either party may end the engagement with two weeks' notice, with completed milestones

payable and work in progress billed at the day rate, protects the freelancer from a client who goes quiet and the client from a freelancer who is not working out.

Nobody enjoys writing this and everybody who has been through a bad engagement has one.

The honest limitation

Some clients read a detailed scope as rigidity, particularly smaller ones used to informal arrangements. The answer is tone, not substance: keep the document short and human, lead with the understanding of their problem rather than with your exclusions, and say out loud that the purpose is so neither of you is surprised.

If a client is genuinely unwilling to define what "done" means, that is not a document problem. That is the client, and the information is worth having before you start rather than in week eight.

BEFORE YOU MOVE ON

Three weeks into a fixed-price project, a client asks for "one more small dashboard, while you're in there". It is genuinely about two days of work. What is the best response?

A. Do it without charging, since two days is minor and goodwill will pay off in future work.

B. Decline politely and point to the signed scope.

C. Agree, and send a short change note stating the cost and the revised delivery date before starting it.

D. Do it and mention the extra effort in the final invoice as justification for a higher total.

73 · 9 min

Invoices, procurement, and the chase

THE ONE THING TO KEEP

Get onto the client's vendor system and obtain a purchase order before starting, chase on a fixed schedule rather than by mood, and keep something undelivered until the final payment clears.

The deposit is the beginning, not the system

Taking 30% to 50% up front is the rule from the first-client lesson and it stands. This lesson is what happens with the rest, particularly when the client is large enough to have a finance department that has never heard of you.

The invoice, and the field that stops it being paid

An invoice needs: your legal or trading name and address; your tax registration number if you have one; a **sequential invoice number**; the issue date; a **specific due date** rather than "net 30"; a description tied to the agreed milestone; the amount and currency; your bank details; and — the one people miss — the client's **purchase order number** where one exists.

In any company running procurement software, an invoice without a matching purchase order does not get rejected. It sits, unmatched, in a queue, until somebody notices. Your contact assumes it was paid. You assume they are ignoring you. Nine weeks pass.

Get set up as a vendor before you start work

For clients above roughly two hundred people, being paid requires you to exist in their supplier system: a vendor form, bank verification, possibly a tax form for cross-border payments, sometimes proof of insurance. This routinely takes two to six weeks and it is nobody's priority.

Start it on day one, not when you send the first invoice. Ask your contact directly: *"Who do I speak to about vendor onboarding and a purchase order, so the first invoice does not get stuck?"* Asking that question marks you as someone who has done this before, and it removes the most common cause of a two-month delay.

Payment terms are negotiable, and lateness has a price

Net 30 from the invoice date is the common default. Large companies impose net 45, 60 or 90, and often will not move. Small ones will frequently agree to net 14 if you ask before signing.

Several jurisdictions give you a statutory remedy. The EU's late payment rules entitle a business creditor to interest at a reference rate plus eight percentage points, plus a fixed recovery amount, and the UK has an equivalent regime. Where such a rule exists, put a line in your contract citing it. You will rarely charge it. Mentioning it in a third chase letter changes the conversation, because the person reading it knows it is real.

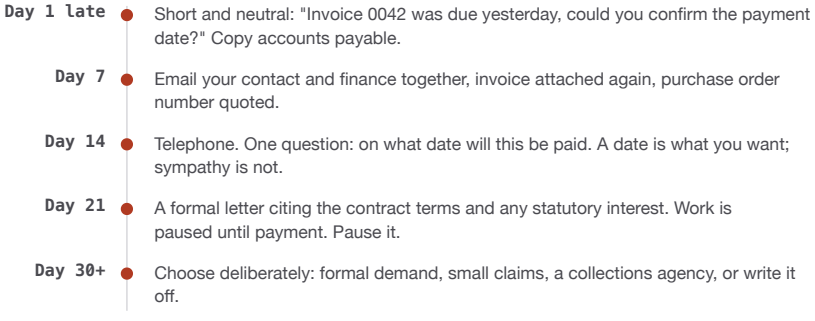
The chase, as a sequence

Decide this before you need it, because the decisions are harder when you are anxious and owed money.

- **Day 1 after due date.** Short, neutral, unbothered. "Invoice 0042 was due yesterday — could you confirm the payment date?" Copy accounts payable if you have the address.
- **Day 7.** Direct email to your contact *and* finance, with the invoice attached again and the purchase order number quoted.
- **Day 14.** Telephone. Ask one question: *on what date will this be paid?* A date is the thing you want; sympathy is not.
- **Day 21.** A formal letter referencing the contract terms and any statutory interest, and stating that work is paused until payment. Pause the work.
- **Day 30 and beyond.** Choose: a formal demand, a small-claims process where the amount qualifies, a collections agency taking a percentage, or writing it off. Choose deliberately rather than drifting.

Escalate on schedule and unemotionally. Most late payment is administrative rather than malicious, and the freelancers who are paid fastest are the ones who follow up predictably without becoming difficult.

The chase, on a schedule



Decided before it is needed, because the decisions are harder when you are anxious and owed money. Most late payment is administrative, and the freelancers paid fastest follow up predictably without becoming difficult.

Your leverage is unfinished work

The strongest position you will ever hold is before you hand over the final files, credentials, deployment or repository access. Afterwards you have only the contract and the law, and both are slow.

Structure engagements so that something meaningful remains undelivered until the final payment clears. Show the work running, on your machine or in your environment; transfer on payment. This is standard practice, it is not hostile, and stating it in the proposal prevents it from feeling like one later.

Bad debt, and the clients who cause it

Budget one to three per cent of revenue as bad debt. Treating it as a cost of business rather than a personal failure means you make better decisions when it happens.

The warning signs are visible before you start, and they repeat: no written agreement; refusal of any deposit; a very small company insisting "our process does not allow deposits"; urgency combined with vagueness about the outcome; and a first milestone paid late. That last one is the strongest predictor

there is. A client who pays the first milestone eleven days late will not improve, and the correct response is to pause and get the terms fixed while the amount at risk is still small.

Across borders

Everything above is harder internationally. Payment takes longer, conversion costs money, and litigation is usually impractical — the cost of pursuing a €4,000 debt in another country exceeds the debt.

So the structure has to do the work instead: a larger deposit for a first international client, smaller milestones, staged handover, and payment through a platform or escrow for the first engagement. Once somebody has paid you twice, you can relax the terms. Not before.

BEFORE YOU MOVE ON

A freelancer's invoice to a 900-person company is six weeks overdue. Their contact says it was approved weeks ago and does not understand the delay. What is the most likely cause?

A. The client is short of cash and is delaying suppliers deliberately.

B. The invoice is sitting unmatched in accounts payable — typically missing a purchase order number or submitted before vendor onboarding completed — so no human has consciously chosen not to pay it.

C. The contact lacks the authority to approve payments and has misled the freelancer.

D. Payment terms were net 60 and the invoice is not actually overdue.

74 · 9 min

Registration, tax, records and insurance

THE ONE THING TO KEEP

Move a fixed share of every payment into a separate tax account the day it arrives, keep records weekly, and treat professional indemnity insurance as a contract requirement rather than an optional extra.

The part that decides whether the good year survives

This is the least interesting lesson in the course and it prevents more damage than any other. It is also the most country-specific thing here, so what follows is the shape of the questions. **Verify everything against your own revenue authority's guidance, which is free, and pay an accountant once the numbers justify it.**

What structure to trade under

Two broad options nearly everywhere.

Sole trader, sole proprietorship, or the local equivalent. Cheap or free, minimal filing, and you and the business are the same legal person — which means business debts are your debts.

A limited company. Costs money to form and to maintain, requires accounts and filings, and separates your personal assets from the business. Some clients, particularly large ones and some agencies, will only contract with an incorporated supplier, which is often the actual reason people incorporate.

Reasonable default: start as a sole trader, incorporate when a client requires it, when the liability is real, or when the tax arithmetic in your country favours it. That last one changes with budgets and is exactly what an accountant is for.

Tax, and the one habit that matters

On the day money arrives, move a fixed percentage into a separate account and do not touch it. Twenty-five to thirty-five per cent, depending on your country and income. That account is not yours.

This single habit prevents the most common freelance disaster, which is not a failed business — it is a good year followed by a tax bill for money that was spent in good faith on the assumption that everything received was income.

Other things to establish early:

- Whether you must pay tax in instalments during the year rather than in one annual payment. Many countries require this, and the penalties for missing instalments are real.
- Whether you must register for **sales tax, VAT or GST**, and at what turnover threshold. Registering late is expensive; the liability usually applies from the date you crossed the threshold, not from when you noticed.
- How **exported services** are treated. Selling to a client abroad frequently has different sales-tax treatment from selling domestically, sometimes with a form or declaration required to benefit from it.
- What social security or pension contributions you owe as a self-employed person.

Records

Keep everything: contracts, invoices, receipts, bank statements, and a note of what each payment was for. Retention requirements are commonly five to seven years.

Free tools that are genuinely sufficient: a spreadsheet plus a folder per year works for a small practice; **Wave**, **Invoice Ninja** and **GnuCash** are free and do proper bookkeeping; **Zoho Books** has a free tier for small businesses in some markets. The paid standards are QuickBooks and Xero, and they are worth it once you have an accountant who wants to work in one.

Record income when it arrives and expenses when they occur, in the same place, weekly. Twenty minutes a week; a nightmare once a year.

Insurance

Four kinds, in order of how likely you are to need them:

- **Professional indemnity.** Covers claims that your work caused a loss. Increasingly required contractually — larger clients and agencies frequently ask for a certificate before signing, and not having one can cost you an engagement outright.
- **Public liability**, if you visit client sites.
- **Health cover**, which your employer used to buy and which is now a business cost you must price into your rate.
- **Income protection**, which pays if you cannot work. Expensive, dull, and the only thing standing between an illness and a catastrophe when you are the entire company.

Contracts

Have two documents. A **master agreement** covering the standing terms — payment, liability, intellectual property, confidentiality, termination — signed once per client. And a short **statement of work** per project, which is essentially the proposal from the previous lesson.

Free templates exist and are a reasonable starting point. Paying a lawyer once to review your template, rather than every contract, is among the best money a freelancer spends: a few hundred in local currency, once, covering every engagement afterwards.

When to get an accountant

Sooner than most people do. The triggers: foreign income, crossing a sales-tax threshold, incorporating, or simply reaching the point where an hour of your billable time costs more than an hour of theirs. In most markets a small practice charges less annually than one avoidable penalty.

The honest limitation

Everything here is jurisdictional and it changes with budgets. A model asked about your country's thresholds will answer confidently and may be describing last year's rules or another country's.

The durable skill is knowing which questions to ask: what structure, what taxes, what thresholds, what records, what insurance, what forms and when. Take that list to an official source or a professional and you will get the right answers in an hour.

BEFORE YOU MOVE ON

A freelancer has a strong year, spends what arrives, and is then assessed for tax on the full amount. What structural habit would have prevented this?

A. Incorporating a limited company, since companies are taxed at lower rates.

B. Charging a higher rate to cover the tax burden.

C. Transferring a fixed percentage of each payment into a separate account on the day it arrives, and treating that account as not theirs.

D. Filing quarterly rather than annually, so the amounts are smaller.

75 · 9 min

Repeatable offers, retainers and the ceiling of hours

THE ONE THING TO KEEP

Hours are capped, so income above a certain point comes from repeating a defined offer, from retainers that create a floor, or from assets you own — all of which require narrowing what you sell.

The ceiling is arithmetic

Freelance income is rate multiplied by billable hours. Hours are capped by the calendar and by your health. Rates rise, but slowly, and each rise is a negotiation with somebody who would rather not.

So a purely hourly practice has a ceiling, and most people meet it in year three, work harder, and find the number barely moves. The escapes are few and they are all forms of not being paid for time.

Escape one: the productised service

The same offer, at a fixed price, delivered by the same process, to the same kind of buyer.

> *"A two-week reporting audit for e-commerce companies. Fixed price. You get a written assessment of your data model, a prioritised list of what is broken, and a costed plan. Delivered in ten working days."*

Four things improve at once, and they compound.

The sale gets much easier. A defined thing with a price is a decision the buyer can make in one conversation. A bespoke engagement requires scoping calls, a proposal, revisions and a wait.

Your delivery cost falls with repetition. The first one takes ten days. By the fifth you have a checklist, a query pack, a report template and a set of things you always find, and it takes six. The price did not change, so your effective rate rose by two thirds without a single negotiation.

You get better at one thing instead of adequate at nine.

You can eventually delegate it, because a documented process is transferable and a bespoke engagement in your head is not.

The uncomfortable part is that productising means saying no to work outside the offer, or at least stopping advertising it. Narrowing feels like losing options. It is the mechanism.

Escape two: the retainer

A fixed monthly fee for an agreed scope. This is the single largest improvement available to a freelance business, because it converts feast-and-famine into a floor.

Three retainers at ₹80,000 a month is ₹28,80,000 a year that arrives whether or not you sold anything in March. That changes what you can refuse, which changes what you are paid.

What to sell as a retainer: maintenance of what you built, an agreed number of days a month, monitoring and monthly reporting, or being the person who answers when something breaks. What you are really selling is availability and continuity, and clients value both once they have depended on you.

Two traps, both common:

- **The retainer that fills with ad hoc work.** Write the scope as tightly as a project scope and use the same change-request sentence. Otherwise "two days a month" becomes ten.
- **Becoming an employee without the protections.** One client taking four days a week, with your day-to-day directed by them, is employment in everything but law — the contractor lesson in the previous module explains why that matters. Keep any single client under roughly half your

revenue, and if you cannot, consider whether you should be employed there instead.

Escape three: leverage

Subcontract. Take work larger than yourself and bring in another freelancer at a rate below what you charge. You now carry client risk and payment risk for both of you, which is a real job and not free money, but it is the only way to serve a client bigger than your calendar.

Build assets. Templates, a project skeleton, an internal package, a checklist, a standard report. These are what make repetition profitable and they belong to you rather than to any client — provided your contracts do not assign them away, which is why the intellectual property clause matters.

Sell a product. A course, a template pack, a small tool. Be sceptical here: this is a marketing business, not a technical one, and the people who succeed at it are mostly people who already had an audience. It is not passive, it does not scale on its own, and treating it as an escape from selling gets it exactly backwards.

The honest limitation

Not everybody should productise. Bespoke work is more varied, and variety is a legitimate reason to accept a lower ceiling — plenty of people are happier doing five different things a year at a good rate than the same audit forty times.

What is not legitimate is drifting. Choose: a bespoke practice with a rate you actively raise, or a narrowed offer you repeat and improve. Both work. The version that fails is doing bespoke work at a fixed rate for six years and being surprised that the income never moved.

BEFORE YOU MOVE ON

A freelancer has delivered eleven similar reporting audits at a fixed price, each taking about six days after early ones took ten. What has actually changed?

A. Nothing meaningful; they are simply working faster, which will plateau.

B. Their effective hourly rate has risen substantially without any negotiation, because the price stayed fixed while the delivery cost fell — and the templates and checklists that caused it are assets they own and can delegate from.

C. They should now reduce the price to win more work, since the job costs them less.

D. They are underpricing and should switch to hourly billing to capture the true value.

76 · 8 min

The conditions under which you should take the job

THE ONE THING TO KEEP

Freelancing early without anyone reviewing your work produces years of uncorrected habits, and selling is a permanent 30% to 40% of the job rather than a start-up cost.

An honest counterweight

Freelancing is sold hard, mostly by people who profit from selling it. This lesson exists because a course that only argues one way is advertising.

There are conditions under which working for yourself is the wrong choice right now, and recognising yourself in them is not a failure of ambition.

1. You are early and nobody reviews your work

This is the largest one. In your first two or three years, most of what makes you better comes from somebody more experienced looking at your work and telling you it is wrong: a code review, a design discussion, a senior colleague asking why you joined those tables that way.

Clients do not do this. Clients tell you whether they are happy, which is a different signal and frequently an inverse one — a client can be delighted by an analysis that is quietly wrong, and you will never learn that it was.

Freelancing early is how people acquire five years of habits nobody corrected. If you have no one reviewing your work, that alone is a strong reason to take a job first.

2. You have no reserve, and obligations that cannot wait

Rent, dependants, a loan, medical costs. Freelance income is not merely lower at first; it is **irregular**, and irregularity is harder to live with than a lower average. Three to six months of expenses is the entry cost. Without it, the first slow month forces you to accept a bad client, and bad clients consume the time you needed to find good ones.

3. Your immigration status depends on employment

A sponsored visa usually ties your right to remain to a specific employer. Freelancing may be prohibited outright or may jeopardise the visa. Check before, not after, and check with an official source.

4. You need health cover you cannot otherwise obtain

In markets without comprehensive public healthcare, employer cover is a large, real benefit — particularly with a family or an existing condition. Price the replacement honestly before deciding; it is often larger than expected.

5. You will not sell

Selling is 30% to 40% of the job, forever, and it does not become optional once you are good. If the thought of following up an unanswered email a second time makes you avoid your desk, this is disqualifying information rather than a detail to work on later. It is entirely respectable to want to do the work and not the business.

6. You want to work on big systems

The genuinely hard problems of scale, reliability and organisational complexity exist inside organisations, and they take years to see. A freelancer arrives, builds, and leaves. You rarely watch a decision play out over eighteen months, rarely carry a system through three reorganisations, and rarely learn what your own design did to the people who inherited it.

That last one is a real gap in a career, and it is not recoverable by doing more projects.

The hybrid, which is usually the right first move

Keep the salaried job and take one small engagement, five hours a week. You test the market, learn the sales and admin, and build two references at almost no risk.

One caution with teeth: **read your employment contract's outside-work clause first.** Moonlighting restrictions have tightened, particularly in India since 2022, and several large employers have dismissed people over dual employment. Get any permission in writing from someone whose authority survives their departure.

Coming back afterwards

A period of freelancing is not a black mark, provided it is legible. What raises questions is a CV saying "Self-employed consultant, 2024–2026" with nothing checkable underneath.

Make it checkable while you are doing it: name the client sectors even where you cannot name the clients, describe the outcome with a number, keep two referees who will answer a call, and publish a write-up of each engagement with the confidential parts removed. Then it reads as three years of varied delivery, which is what it was.

The test

Freelancing is a business, and the technical work is perhaps forty per cent of it. The other sixty is selling, scoping, invoicing, chasing, filing and deciding.

If that sounds like a reasonable trade for autonomy and a higher ceiling, it is a good life and this module is your manual. If it sounds like the tax you pay to do the real work, take the job — and freelance on the side, where the sixty per cent stays small.

BEFORE YOU MOVE ON

A developer with fourteen months of experience, no savings and a strong portfolio wants to freelance full time. What is the strongest argument against it right now?

A. Their portfolio is not yet strong enough to win clients at a sustainable rate.

B. Freelance rates are lower than salaries for junior people, so they would earn less.

C. With no reserve, an irregular income forces them to accept poor clients in the first slow month — and with fourteen months of experience, no one will be reviewing their work, so mistakes will go uncorrected for years.

D. Clients prefer established consultants and will not hire someone with fourteen months of experience.

CASE STUDY · MODULE 8

Invoice 0042, and the purchase order nobody mentioned

Vinod, a freelance analytics engineer in Kochi, sixty-one days after delivering phase two to a two-thousand-person retailer.

The engagement was ₹4,20,000 across three phases — thirty per cent on start, forty on delivery of the build, thirty on acceptance. The deposit had been paid on time, in eleven days, by his contact's own corporate card because "finance is slow".

That sentence was the warning and he did not hear it.

Phase two was delivered on 14 March. Invoice 0042, for ₹1,68,000, went out the same day. On 14 May it had not been paid, his contact was answering with "it's with finance", and Vinod had already done most of phase three.

What was actually wrong

He telephoned on day fourteen of the chase and asked one question — *on what date will this be paid* — and got the answer that mattered. There was no purchase order. In the retailer's procurement software an invoice without a matching PO does not get rejected; it sits in an unmatched queue until somebody notices. His contact assumed it had been paid. Vinod assumed he was being ignored.

Worse: he had never been onboarded as a vendor at all. That process — a supplier form, bank verification, a tax declaration and proof of professional indemnity insurance — takes two to six weeks at a company that size and is nobody's priority. It could have been started on day one of the project. It was started on day sixty-two.

The cash-flow arithmetic underneath it

He was profitable on paper and could not pay his own bills. Work delivered on 14 March, invoiced the same day on thirty-day terms, would have been paid in

mid-April by a prompt client. This client was not prompt and was not refusing; it was simply a two-thousand-person company with a procurement department, and those pay in six to ten weeks as a matter of course rather than as a matter of intent.

That is the difference between profit and cash flow, and it is why the reserve exists. He had two months of expenses rather than the three to six the arithmetic asks for, which is what turned an administrative delay into a decision made under pressure.

The squeeze

Meanwhile his contact asked, pleasantly, for repository access and the warehouse credentials, because a new analyst was starting on Monday and "it would be good for her to get going".

That was the decision.

Hand over. The client is the largest name on his site, a retainer had been mentioned twice, and refusing over money would be remembered. Once the repository, the credentials and the deployment are transferred, his only remaining instruments are a contract and the law, and both are slow enough to be theoretical for ₹2,94,000.

Hold the handover and pause the work. Protects the ₹1,68,000 outstanding and the ₹1,26,000 to come. Risks the retainer, the reference, and a reputation among the four people at that company who would recommend him.

There was also a smaller ledger he had been quietly absorbing: three "while you're in there, could you also" requests, worth about a week between them, none of which he had priced or logged. Each had been small and reasonable on the day it arrived, which is how scope creep always arrives; nobody had demanded anything. The proposal had said what was included and had not said clearly enough what was not, and it did not number the revision rounds.

He was also aware, uncomfortably, that the client had not behaved badly at any point. His contact had chased finance twice. Nothing here required a villain.

STOP HERE

Before turning the page: what would you have done, and what would it have cost if you were wrong? Write it down. The point of the next section is to compare.

WHAT HAPPENED

He ran the chase as a schedule rather than as a mood: a short neutral note the day after the due date, an email to accounts payable with the invoice re-attached at day seven, the telephone call at day fourteen that found the missing purchase order, and at day twenty-one a formal letter citing the contract's payment terms and stating that work was paused. He did not refuse the handover; he offered a call in which he demonstrated the dbt project and the three dashboards running in his own environment, and said the transfer would happen on the day the payment cleared. Nobody was offended — his contact used the letter internally to get the PO raised, which is frequently what such letters are for. Payment landed on day seventy-four. He took the retainer at a rate fifteen per cent higher, with net-thirty terms, a purchase order required before any work starts, two numbered revision rounds, and vendor onboarding as the first item on the plan. The three absorbed changes went on the final invoice at full price with the discount shown as a line, so the next request had a number attached to it.

Why was the missing purchase order a bigger problem than a client who did not want to pay?

Because nothing was being refused — the invoice was invisible to the system that pays invoices, so no human was ever going to act on it. That is administrative failure rather than bad faith, and it is fixed by a phone call and a PO rather than by pressure. It is also entirely preventable by asking on day one who handles vendor onboarding and purchase orders.

Vinod's leverage was the undelivered handover. Why is using it not hostile?

Because it is standard practice and because it was stated in the proposal before the work began. Showing the work running in his own environment demonstrates that it exists and is complete, so the client loses nothing but the transfer date. Once cre-

dentials, repositories and deployments have been handed over, a freelancer has only a contract and a legal process, which for a sum this size is not a real remedy.

What single earlier signal should have changed how he set the terms?

The deposit paid on the contact's personal corporate card because finance was slow. A first milestone that is paid awkwardly or late is the strongest predictor of later payment trouble there is, and the right response is to pause and fix the terms while the amount at risk is still small — get the vendor onboarding started, get the purchase order raised, and make the next milestone smaller.

MODULE 8 · TEST

Check what stuck

6 questions, one right answer each. This one is for you, not for a record: answers and the reason behind each are at the back. If two questions from the same module go wrong, re-read that module before the exam rather than after.

1 Somebody on ₹15,00,000 divides by 1,920 hours, gets ₹780, and quotes ₹1,000 an hour. Why do they earn less than before?

A. Clients negotiate the quoted rate down by roughly a third on most engagements.

B. Only half to two-thirds of hours are billable, and the employer's costs are now theirs.

C. Income tax is higher for the self-employed in most countries.

D. Fixed-price work is far more common than hourly work, so the quoted rate is rarely applied at all.

2 Which section of a proposal prevents the most arguments in week eight?

A. The section listing what is not included

B. The section describing your approach and methodology

C. The section listing your relevant previous engagements

D. The section setting out the tools and technologies you will use

3 Your invoice to a two-thousand-person client is sixty days late and your contact says it is with finance. What is the most likely cause?

A. The client is disputing the deliverable and has not said so directly.

B. The invoice does not match a purchase order, so it sits unmatched in a queue.

C. The payment terms in the contract are net sixty rather than net thirty.

D. The finance team is waiting for the final milestone before releasing any payment.

4 You want to raise your rate by twenty per cent. What actually works?

A. Announce the new rate to all clients at once, with three months' notice given

B. Wait until a client praises the work, then ask for an increase on that project

C. Quote the new number to the next enquiry, and give existing clients notice separately

D. Add a surcharge for urgent work, which raises the effective rate without a conversation

5 The fifth delivery of a fixed-price two-week audit takes six days instead of ten, at the same price. What happened?

A. The client's scope was smaller, which is common after the first few engagements.

B. Repetition built a checklist and a template, so delivery cost fell while the price held.

C. Experience allowed corners to be cut that the first few clients would certainly have noticed.

D. The estimate was padded originally, and the real duration has now become visible.

6 What is the strongest reason for somebody two years into their career to take a salaried job rather than freelance?

A. Freelance income is lower on average in the first three years.

B. Employers rarely hire people whose recent history is self-employment.

C. Clients tell you whether they are happy; nobody tells you when the work is wrong.

D. Selling consumes roughly a third of the working week, which reduces billable capacity.

MODULE 9

Staying: the first ninety days and the ten years after

Getting hired is the short problem. This module covers the long one — earning trust in a new team, being the person whose numbers are right, deciding what to learn while the field shouts, refusing the thing you should not build, and knowing a plateau when you are standing on one.

> BY THE END OF THIS MODULE YOU CAN
>
> Earn trust in a new data role within ninety days, and decide what to learn, what to refuse and when to leave using evidence rather than anxiety.

77 · 8 min

The first ninety days

> THE ONE THING TO KEEP
>
> Reproduce other people's numbers before you propose changing anything, because the mess usually encodes a reason you do not know yet.

The pattern that wastes two years

A new data hire arrives, reads the code, is appalled, and proposes a rewrite in week two. Everything they said was technically correct. Nobody agrees, the proposal dies, and they spend the next two years as the person who criticised things they did not understand.

The mechanism is simple and worth internalising: almost every piece of ugliness in a working system was put there deliberately by somebody solving a problem you have not met yet. The hard-coded exception for one customer, the pipeline that runs twice, the column called `amount_2` — each has a story, and the story is usually a person or a regulation rather than laziness.

You are not being asked to accept it for ever. You are being asked to find out first.

The read-only month

Spend your first four weeks reproducing the numbers the team already reports.

Pick the three figures leadership sees monthly. Work out, from source, how each is produced. Then reproduce them yourself and compare.

One of two things happens, and both are good. Your number matches, and you now understand the business's definitions — which is most of what a new analyst is missing. Or your number differs, and you have found either your own misunderstanding or a real bug. Bring the difference as a question, never as an accusation: *"I get 14,120 and the report says 14,200 — what am I missing?"* That sentence has started more good careers than any rewrite proposal.

The map to build in your head

By the end of month one you want to be able to answer:

- Where does the data come from, and which system is authoritative when two disagree?
- Which table do people actually trust, and which one is deprecated but still queried?
- What are the five questions leadership asks every month?
- Who do I ask about what? Names, not teams.
- Which dashboard do people actually open, and which ones does nobody look at?
- What broke most recently, and why?

That last question is the fastest route to understanding a system. Ask someone to tell you about the last incident.

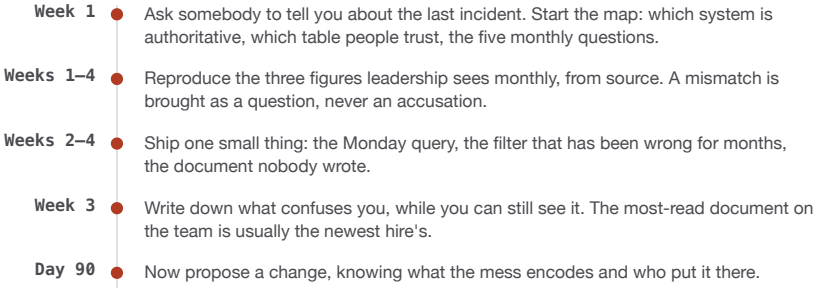
Small useful things, delivered early

Between weeks two and four, ship something small. Not a platform. Candidates:

- The query somebody runs by hand every Monday morning.
- A filter that has been wrong on a dashboard for months and everyone works around.
- Documentation of the thing nobody documented.

The purpose is not the value of the work. It is that you become somebody who finishes things, in the first month, when opinions are being formed.

The first ninety days



Reproduce before you propose. The ugliness in a working system usually encodes a person or a regulation you have not met yet, and the rewrite proposed in week two is what turns a new hire into the person who criticised things they did not understand.

The document that makes you valuable in week three

Write down what confuses you, as it confuses you. The acronyms, the table that is not what its name suggests, the two systems with similar names, the report that means something different from what it says.

Do it now, because in three months you will no longer be able to see any of it. That fluency is the whole problem: everybody who could write this document has forgotten what was hard.

Onboarding notes written by the most recent hire are usually the most-read document on a data team, and writing them is one of the cheapest ways to be visibly useful before you are technically productive. Keep it in the team's wiki, or a markdown file in the repository if there is no wiki. Notion and Confluence are the paid tools; a text file in git works and is searchable.

Asking questions well

The fear is looking stupid. The actual failure is guessing quietly for three days and producing something wrong.

Two rules. **Thirty minutes stuck, then ask** — and say what you already tried, so the answer is quick to give. And **ask in a public channel** rather than a direct message, so the answer is searchable by the next person and so your colleague answers once rather than four times. A new person asking good questions in the open reads as engaged, not as struggling.

When the read-only month is impossible

Sometimes you are hired because things are on fire and there is no month to spend reading. That is a legitimate situation, and the response is to say plainly what you are skipping: *"I am going to fix the nightly failure first, which means I will not have reproduced the revenue figures before I touch this pipeline. I will come back to that in month two."*

Naming what you are trading is what distinguishes a deliberate shortcut from an accident. It also means that when something breaks, the conversation is about a known risk rather than about you.

BEFORE YOU MOVE ON

Two weeks into a new data job, you find the main pipeline contains a hard-coded exception that treats one large customer's orders differently, with no comment explaining why. What should you do?

A. Remove it, since undocumented special cases are exactly the kind of technical debt a new hire should clear.
B. Leave it alone entirely; existing code is the team's decision and not a new person's business.
C. Find out who wrote it and why, ask them directly, and write the answer into a comment or the documentation — the reason is almost always a real constraint, and recording it is the useful contribution.
D. Write a proposal documenting the debt you have found and present it to the team as an improvement plan.

78 · 9 min

Landing in a codebase nobody documented

THE ONE THING TO KEEP

Trace one behaviour end to end rather than reading the codebase, and use git history to recover the reasons — the strange condition usually has a commit explaining the incident that caused it.

Week two, and 40,000 lines

The repository has no README worth reading, three directories whose names mean nothing, a `utils.py` of 1,800 lines, and the person who wrote most of it left in March. Somebody has asked you to change how one number is calculated.

This situation is the normal case, not the unlucky one, and being good at it is one of the most transferable skills in this field. Almost nobody is taught it.

Do not read from the top

The instinct is to open the first file and start reading. This fails because a codebase is not a book; it is a graph, and you have no idea which parts matter.

Start from a behaviour and trace it. Pick one thing the system visibly does — one endpoint, one scheduled job, one number on one dashboard — and follow it end to end. You will understand a thin slice completely, which is far more useful than understanding forty per cent of everything vaguely.

Find the entry point by searching for something you can see: the URL path, the table name, the text of an error message, the label on the dashboard. Then follow the calls outward, writing down each hop.

The tools that answer "why"

Git holds the history and almost nobody uses it for archaeology. Four commands do most of the work:

```
# history of one file, newest first
git log --oneline -- src/pricing/discount.py

# every commit that added or removed this string, anywhere
git log -S "customer_status" --oneline

# who wrote these lines, and in which commit
git blame -L 120,150 src/pricing/discount.py

# the full commit: message, diff, and often the ticket link
git show 4f2a91c
```

`git log -S` is the one to remember. It finds the commit where a magic constant, a strange flag or an odd condition first appeared, and the commit message frequently contains the entire explanation: *"Handle the Belgian VAT case — see TICKET-4412."*

Commit messages, pull request descriptions and the tickets they link to are the real documentation of most systems. They are also the reason the commit-

message convention in this course's guidelines asks for a body explaining *why*.

Tests are the specification

If tests exist, read them before the implementation. A test says what the authors believed the code should do, in executable form, and it usually says it more clearly than the code does. Running one test under a debugger, or with print statements inserted, teaches more in ten minutes than an hour of reading.

If there are no tests, writing one for the behaviour you are about to change is the safest first move you can make, and it converts your reading into something the team keeps.

The fence you should not remove

You will find a condition that makes no sense — a hard-coded date, an exception for one customer, a retry with an oddly specific timeout. The temptation is to clean it up.

Do not, until you know why it exists. Use `git log -S` on the strange value. Nine times out of ten there is a commit saying exactly what broke, and the tenth time you ask someone with the evidence in hand. Removing a condition somebody added at 2am after an incident is how a new joiner causes the same incident again, and it is remembered.

Write the map as you go

Keep a file as you trace: what calls what, which table holds what, which directory is dead, which name is a lie. Two weeks later, tidy it into a page and open a pull request adding it to the repository.

This is the highest-value first contribution a new joiner can make. It is genuinely useful, it is unarguably yours, and it makes your learning visible without anybody having to take your word for it.

Ask questions with your homework attached

There is an enormous difference between two questions.

> *"How does the pricing service work?"*

This costs somebody a meeting, and you will retain a third of it.

> *"I traced the discount calculation from the API handler to `apply_tier`, and it looks like the branch at line 214 exists to handle legacy contracts. Is that right, and is there anything else that path touches?"*

This costs thirty seconds, gets a precise answer, and tells the person you are worth answering. Do the tracing first, then ask about the one thing you could not resolve.

The honest limitation

Some code is genuinely unknowable — no history because it was imported in one commit, no tests, no one left who remembers. The professional response is not heroic archaeology; it is to **bound it**. Write tests capturing what it currently does, change only what you must, and treat it as a black box with a known interface.

An AI assistant is good at summarising what a file does and is confidently unreliable about *why* it does it, because the reasons live in tickets and in people's memories rather than in the code. Use it for the first pass, then use `git log` for the part that matters.

BEFORE YOU MOVE ON

A new joiner finds a hard-coded three-second timeout with no comment and wants to remove it as untidy. What is the most effective first step?

A. Run `git log -S "3"` on the file to find the commit that introduced the value, and read its message and linked ticket.
B. Remove it behind a feature flag so it can be restored quickly if something breaks.
C. Ask in the team channel whether anyone knows why the timeout is there.
D. Check whether any test covers the timeout, and if none does, treat it as dead configuration.

79 · 8 min

Being trusted with a number

THE ONE THING TO KEEP

Reputation in data work is built by catching your own mistakes before anyone else does.

The asymmetry that governs the job

One wrong number presented in a leadership meeting costs more trust than ten correct ones build. That is unfair and it is how people work: a correct number is invisible, and a wrong one is a story told for a year.

The response is not anxiety. It is a routine — a fixed sequence you run before anything leaves your hands, so that checking is not a decision you have to make while tired.

The checking list

1. Row counts before and after every join. Write them down. If the count changed and you did not expect it to, stop and find out why. This single habit prevents the most expensive class of error in analytics.

2. Reconcile against a second source. The finance export, the admin panel, last month's report, the count in the source system. Your number does not have to match exactly; you have to be able to explain the gap.

3. Look at five actual rows. Beginners never do this and it catches an astonishing share of bugs — the duplicated record, the test account, the row where the customer name is `asdf`.

4. Check the boundaries. Earliest and latest date. Minimum and maximum of every numeric column. Count of nulls per column. A max order value of 9,999,999 or a date in 1970 is a data problem announcing itself.

5. Ask what this number would look like if you were wrong in the most likely way. If you suspect a fan-out, what would revenue look like if every order were counted three times? If it looks like *that*, you have your answer.

Two bugs worth knowing by name

Fan-out, from the SQL lesson: joining to a one-to-many table before aggregating a one-side column multiplies it. Row counts catch it.

The daylight-saving gap. A job that runs daily at 00:30 local time will, twice a year, run twice or not at all, because local time is not monotonic — the clock moves backward in autumn and forward in spring. The symptom is one missing day and one doubled day, every year, in the same two weeks, and it is diagnosed hundreds of times a year by people who have never seen it before. Schedule in UTC; convert for display only.

Presenting a number with its uncertainty

The fear is that caveats make you sound unreliable. Done properly they do the opposite:

> *"Orders were 14,200 last month, up nine per cent. Two things to know: the Nigeria channel is missing three days because of the migration, so the true figure is slightly higher. And I have excluded test accounts, which the old report did not — that alone accounts for about two points of the change."*

That is a person who has checked. The version without caveats is either luckier or less careful, and everybody in the room eventually finds out which.

"I do not know yet"

A complete and professional answer, with a time attached: *"I do not know yet. I will have it by four."* It builds far more trust than a confident guess that turns out wrong, and it is available to you at any experience level.

The failure mode is the guess given to avoid looking uncertain in a meeting, which then travels into a slide, which then travels into a decision.

When you get it wrong

You will. The recovery is mechanical:

1. Say so immediately, in the same channel where the wrong number was given.
2. Give the corrected figure.
3. One sentence on the cause. Not a paragraph of apology.
4. One sentence on what stops it recurring.

Do not bury it, do not wait for the next meeting, and do not hope nobody noticed. Fast correction is the behaviour that makes people comfortable relying on you, because it means they will hear about problems from you first.

Automating the checks

The free path is a script with assertions, and it covers most of what expensive tooling does:

```
assert df["order_id"].is_unique, "duplicate order ids"
assert df["amount"].min() >= 0, "negative amounts present"
assert before == after, f"join changed row count: {before} ->
{after}"
```

dbt has tests built in and is free and open source. **Great Expectations** and **Soda Core** are free libraries for the same job. The paid observability products — Monte Carlo and similar — buy you automated anomaly detection across many tables, which matters at scale and not on your first pipeline. Job adverts name them; start with the assertions.

The limitation

No amount of checking survives a source system changing without telling you. A field renamed, a new status value, a supplier changing their export format — your assertions pass on old assumptions until the day they do not.

That is why monitoring counts *over time* matters more than any one-off check. A row count that has been 40,000 daily for a year and is 12,000 this morning is worth an alert, and setting that up is usually a day of work that pays for itself the first time.

BEFORE YOU MOVE ON

Ten minutes after presenting last quarter's figures to the leadership team, you realise you double-counted one region. What should you do?

A. Correct it in the next monthly report, with a footnote explaining the revision.
B. Say so immediately in the same channel: the corrected figure, one sentence on the cause, and one sentence on the check that will prevent it recurring.
C. Check the whole analysis thoroughly first, then send a full write-up of everything you found.
D. Mention it privately to your manager and let them decide whether it needs a wider correction.

80 · 7 min

Which parts of this work are most exposed

THE ONE THING TO KEEP

Well-specified work with fast verification goes first; being accountable for the answer goes last.

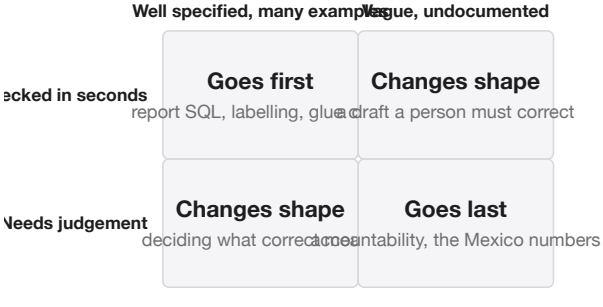
A test you can apply yourself

Forget predictions. Ask three questions about any task:

1. **Is it well specified?** Could you write down what a correct result looks like before starting?
2. **Are there many examples of it being done?** Millions of instances of this exact task, publicly available.
3. **Can correctness be checked in seconds?** By a test, a compiler, a rule, a quick glance.

Three yeses means heavily exposed. Three noes means durable for now. Most real work is a mixture, which is why jobs change shape rather than vanishing.

The three-question test, as a grid



Well specified, abundantly exampled, checked in seconds: that corner goes first, and it is exactly where beginners used to enter. Accountability for the answer and context that lives in people's heads go last.

The most exposed work is the newest work

This is the part people find surprising. Much of the work created by the first wave of AI is what the second wave absorbs: writing large numbers of prompt variations, simple labelling and annotation, routine data cleaning scripts, first-draft dashboards, boilerplate glue code, straightforward report queries.

Also exposed: the traditional bottom rung. "Write the SQL for this report." "Turn this spreadsheet into a chart." "Summarise these tickets." Well specified, abundantly exampled, quickly checked.

That is the honest problem for a beginner. The exposure lands hardest exactly where people used to enter.

What holds up

Accountability. Someone has to be answerable when the number is wrong, the model refuses a legitimate customer, or the pipeline reports a loss that did not happen. Organisations do not assign that to a tool. This is why evaluation work is more durable than prompt work: deciding what "good enough" means is a responsibility, not a task.

Undocumented context. Why the Mexico numbers are always late. Which of the three customer tables anyone actually trusts. Why finance recalculates revenue differently in Q4. This lives in people's heads and in three-year-old chat threads, and it is most of what makes a competent employee competent.

Work that touches the real organisation. Getting access approved. Persuading a team to change how they record something. Running an incident at 2am. Negotiating what data may lawfully be used.

Systems work under constraints. Latency, cost, permissions, failure modes, data that arrives late or twice. Specifying it correctly is nearly as hard as building it.

Say the honest thing about timing

Nobody knows the timeline. People confidently claiming mass unemployment within two years and people confidently claiming nothing will change are both

guessing, and both are usually selling something. What can be observed is direction: tasks meeting the three-yes test are getting cheaper quickly.

The realistic effect is not that a job disappears but that one person does what four or five did, which reduces hiring volume at the bottom before it touches anyone senior.

What to do about it

Do not try to out-type the tools. Move toward the parts they cannot own.

- Become the person who can tell whether an output is *correct*, not just whether it looks plausible. That skill needs domain knowledge and a habit of checking, and it is the scarcest thing on most teams.
- Take jobs where you sit near the mess and near the decision, even if the title is modest.
- Learn one domain properly — health, logistics, agriculture, lending, education. Context is the moat.
- Use the tools heavily. Being slower than a colleague who uses them well is a real risk, and a nearer one than the abstract fear.

And stay usefully sceptical. Anyone certain about this field in either direction is telling you about their business model, not the future.

BEFORE YOU MOVE ON

A company reports that AI now writes most of its internal SQL. A year later, the analytics team has not shrunk. What best explains this?

A. The generated SQL is wrong often enough that people must rewrite it, so nothing was really saved.

B. Answers became cheap, so demand for them rose, and the scarce work moved to choosing the right questions and vouching for the numbers.

C. Companies are slow to reduce headcount, so the cuts are simply still coming.

D. The team spent the year learning new tools, which consumed the time the tools saved.

81 · 7 min

Keeping up without drowning

THE ONE THING TO KEEP

Follow problems and primary sources rather than launches, because almost nothing announced this month changes what you should do this month.

Why it feels urgent

The volume is genuinely enormous and it is not slowing. But notice the mechanism behind the feeling: launch announcements are written to feel urgent, and the person posting is rewarded for your attention rather than for your career. Your sense of falling behind is the product being sold, and it is sold by people with model releases, courses and newsletters to promote.

Once you can see that, the noise becomes much easier to ignore without guilt.

The three-source rule

Pick three sources you actually read, and let everything else reach you by accident.

- **One primary.** The documentation and release notes of the tools you actually use. Model provider docs. The paper, when there is one, rather than the thread about the paper.
- **One practitioner.** Somebody who ships things and writes about what broke. Their failures are worth ten launch posts.
- **One local.** Your country's or language's community, because hiring, pay and regulation are local and the global feed will never tell you about them.

Three. Everything else you sample quarterly, which is often enough for anything that survives a quarter.

The test for whether something matters

Would this change a decision I am making this quarter?

If no, note the name and move on. You can learn any tool in a week when it becomes relevant to you. You cannot recover the six months spent skimming announcements about tools you never used.

This rule feels reckless the first time you apply it and is almost always right. The genuinely important developments do not disappear; they keep appearing, in your primary sources, attached to problems.

What is genuinely worth tracking

Four things, because they change decisions rather than opinions:

- **Prices.** They fall unevenly and they change build-versus-buy arithmetic. A feature that was uneconomic last year may be trivial now.
- **Context limits and capability changes** in the models you actually build on.
- **Deprecations.** This one bites hardest and gets the least attention: a provider retiring a model version breaks an application that has run untouched for a year. Providers publish deprecation schedules; check them for anything you have in production, and diarise the dates.
- **Licence changes on open models and libraries.** An open weight model whose licence changes can make a shipped product non-compliant overnight.

The half-life argument

Some knowledge decays fast and some barely decays.

Fast: the API surface of a specific framework, the current best model, the fashionable orchestration library, prompt tricks tied to one model version.

Slow: SQL. Statistics. Debugging. Writing clearly. How data actually gets messy. Your domain — how insurance claims work, how a hospital admits patients, how freight is priced.

Nearly everyone spends their learning time on the fast-decaying half because it is what gets announced. Deliberately spending most of yours on the slow half is the entire strategy of this lesson, and after five years the difference between two people who studied equally hard is almost entirely this allocation.

A monthly routine that fits a working life

Two hours, once a month:

1. **Thirty minutes:** release notes and deprecation pages for the things you actually run.
2. **Thirty minutes:** one long-form piece or paper, read properly rather than skimmed.
3. **One hour:** build something tiny with one new thing. Not a tutorial — the smallest possible real use.

That is it. It sounds inadequate against the volume of what exists, and it produces more durable capability than daily skimming, because the hour of building is the only part that becomes knowledge.

The free path

Everything you need is free and primary: provider documentation, arXiv, model cards on Hugging Face, project changelogs, and the community for your language or region. The paid layer — newsletters, courses, video subscriptions — mostly repackages those sources with a delay and a personality attached. Some of it is good. None of it is where the information originates, and paying for it does not reduce the volume problem.

The limitation, stated honestly

This approach trades novelty for depth, and it will occasionally make you late to something real. You will hear about a genuinely important shift a few months after the people who read everything.

That is the correct trade for a career, and the cost is smaller than it sounds — the real shifts persist for more than a quarter, so they reach you anyway, at the

point where they have stopped being speculation and started being something you can use.

BEFORE YOU MOVE ON

A significant new model is released with capabilities beyond anything you have used. You are three weeks from delivering a project on the current one. What is the sound response?

A. Evaluate switching immediately, since delivering on an outdated model wastes the work.

B. Deliver as planned and avoid the new model until it has been in the market for a year.

C. Add it as a fallback provider now, so the project benefits without changing the main path.

D. Finish the project, note the release, and after delivery run your existing evaluation set against both models — a measured comparison on your own cases is the only thing that should move a production decision.

---

82 · 8 min

The things you will be asked to build

THE ONE THING TO KEEP

You will be asked to build something you should not; decide your line before the meeting and put the objection in writing.

The requests are ordinary, which is what makes them hard

Nobody asks you to build a surveillance state. They ask, pleasantly, under deadline, for something that sounds reasonable in the room:

- Scrape a competitor's site, including the personal details of their customers or staff.

- Score job applicants on features that stand in for something you are not allowed to use — postcode, name, university, a photograph.
- Ship an assistant with no route to a human, on a service where some of the people using it are in distress.
- Reuse customer data collected for one purpose for a different one, because it is already in the warehouse.
- Publish a metric you know is wrong, because the deadline is fixed and the correct number arrives on Friday.
- Automate a decision that affects people, with no way to appeal it.

Each arrives from somebody you like, who is under pressure, who is not being malicious. That is the mechanism you have to prepare for, and it is why deciding in the meeting is so hard. Decide before.

What to do, in order

1. Ask a clarifying question that surfaces the risk without accusing anyone.

> *"Which lawful basis are we relying on for that data?"*

> *"What happens for the person the model gets wrong?"*

> *"If a customer asks why they were declined, what do we tell them?"*

Half the time the request changes here, because nobody had thought about it and the question is easier to answer with a change than with a defence.

2. Offer the version you can build. "I cannot do it with the applicant photographs. I can do it with the structured application fields, and here is what that costs us in accuracy." Being the person with an alternative is far more effective than being the person with an objection.

3. Put the concern in writing, once, plainly, to your manager. Not a manifesto. Three sentences naming the specific risk. This matters ethically — it makes the decision visible to someone accountable — and it matters for you, because a verbal objection did not happen.

4. Escalate to legal, privacy or the data protection officer if one exists. People assume these functions are obstacles. In practice they would far rather hear about a problem now than after it ships, and they carry authority you do not.

5. Decide whether you stay. If it ships anyway and it crosses your line, that is a decision you should already have thought about, in the calm, before you needed it.

The rules that exist, in outline

You do not need to be a lawyer. You need to know these questions exist and who to ask.

- **Purpose limitation.** Data collected for one purpose generally may not be silently reused for another. This is the core of GDPR-style regimes and of India's DPDP Act, and it is the rule most often broken by accident inside data teams.
- **A right to human review of significant automated decisions**, in several jurisdictions. If a model declines someone's loan or job application, there is often a legal requirement for a person to be reachable.
- **Sector rules.** Health, credit, employment, insurance and children's services all carry additional obligations that override general practice.
- **Scraping.** Public availability is not permission, particularly for personal data. The terms of the site, copyright, and data protection law all apply separately.

Regulators publish their own guidance, free and written for non-lawyers. Your national data protection authority's website is a better source than any paid summary, and citing it in a meeting is unusually effective.

The honest career cost

Refusing has a cost. Usually small: a slightly awkward meeting, a reputation for asking difficult questions, which in a decent organisation is an asset. Occasionally large: the project happens anyway and you are the person who objected.

People who navigate this well share three habits. They pick their hills — not every imperfection is a resignation. They are specific about the harm rather than moralising, because "this will decline legitimate customers in three states and we will not be able to explain why" moves a room and "this is unethical" does not. And they always bring the alternative.

Where this advice stops

None of it helps much if you are the only technical person and the founder wants the thing. There is no committee, no legal function and no ally. Then the question reduces to whether you can afford to leave.

Which is the practical, unromantic conclusion: three months of savings is the thing that lets you hold a line. It is a career asset, and almost nobody thinks of it as one.

BEFORE YOU MOVE ON

Your manager asks you to include applicant postcodes as a feature in a CV-screening model, saying it improves accuracy and is not a protected characteristic. What is the strongest first move?

A. Ask what happens for the applicants the model gets wrong and how you would explain a rejection, then offer the version built on structured application fields with the accuracy cost stated.

B. Refuse on the grounds that postcode is a proxy for race and class, and say the request is discriminatory.

C. Build it as asked and raise the concern if the model's outputs later show a disparity.

D. Include the postcode but add a fairness metric to the monitoring dashboard so any bias becomes visible.

83 · 8 min

The second job, the plateau, and the ten-year view

THE ONE THING TO KEEP

A plateau is a signal to change the work, not to work harder — and the durable assets are a domain, thirty people who have seen you work, and the habit of finishing.

The pattern most careers follow

Year one you learn the tools and feel permanently behind. Year two you become genuinely useful and the panic subsides. Year three you either deepen, broaden, or quietly stall.

The stall is the dangerous one because it feels like competence. Work is easy, you are respected, nothing is difficult. From the inside it is indistinguishable from having arrived.

Diagnosing a plateau, concretely

Three questions, asked honestly once a year:

- **Could I have done last quarter's work eighteen months ago?** If yes, you have been paid to repeat yourself.
- **Is there anybody here I am learning from?** If the answer is no, no amount of effort fixes it, because skill at this level is mostly absorbed from people.
- **Has anything I built been used by more people this year than last?** Scope growing is the clearest external signal that you are.

Three noes is not a crisis. It is a year's notice.

The three moves, and what each costs

Deeper. A specialism — data platform engineering, ML systems, a regulated domain. Costs: narrower job market, higher pay within it, and a real risk if the specialism fades.

Wider. Product, management, solutions work. Costs: your technical depth erodes within about two years and going back is hard. Worth doing when you want it, not as an escape from a technical problem.

Out. Contracting, a small consultancy, your own product. Costs: income variance, the selling half of the job, and no colleagues. Freedom that people romanticise and then find lonely.

Doing nothing is a fourth option, chosen by default rather than decided. Money and comfort compound, the cost of the plateau is invisible, and it presents its bill at the next interview, when you discover what your market rate actually is.

Gaps, caring years and non-linear paths

If you took two years out — for a child, a parent, illness, a country change — the advice is one plain sentence with no apology, then talk about the work. *"I was out of work for two years caring for my father. I kept my SQL current on a project for a local charity."* No elaboration, no defensiveness.

Employers who dwell disproportionately on a gap are telling you how they will treat any future human circumstance of yours. That is worth learning during the interview rather than after.

Burnout, in this field specifically

Data and AI work has three ingredients that produce it reliably. Pages at three in the morning, because pipelines run at night. A permanent, manufactured sense of being behind, from the previous lesson. And no natural end state — the data is never clean, the model is never finished, and there is always one more thing to check.

Two guards that actually work, both structural rather than attitudinal:

- **An on-call arrangement that is written down**, with a rota and a definition of what warrants waking someone. An informal expectation that you are always reachable is the most common cause of people leaving this work altogether.
- **A definition of done for your own projects.** Written before you start. Without one, every project expands until you stop from exhaustion rather than completion.

The asset almost nobody builds deliberately

Three months of expenses saved changes which jobs you can refuse, which requests you can decline, and how a negotiation feels. It is the most useful career asset available to most people and it is never described as one, because it is boring and it is not a skill.

If you are early and underpaid, this is genuinely hard, and it is still worth naming as the goal rather than pretending the choice is only about learning.

What survives ten years

Not the tools. Every framework named in this course will be replaced, and the replacements will be replaced.

What survives:

- **A domain.** Ten years of understanding how hospitals, freight, lending or agriculture actually work is not reproducible by anyone in a hurry.
- **About thirty people who have seen you work** and would take your call. This is what a network is, stripped of its unpleasant connotations, and it is built by being useful to people one step behind you, for years.
- **A written record of what you built and decided.** The habit from the case study lesson, kept up for a decade, is a career's evidence that no employer owns.
- **The habit of finishing.** Rarer than talent and more valuable.

The limitation, honestly

None of this is a guarantee. Markets contract, sectors are hit, well-run companies fail, and a good decision can still end badly through nothing you did. Career advice that promises otherwise is describing survivors.

What the evidence habit actually buys is not immunity. It is that your work is legible to somebody else when you need it to be — which is the difference between a bad year and a lost one.

BEFORE YOU MOVE ON

Three years into a data job, Sofia is respected, comfortable, and cannot remember the last time work was difficult. Her manager is pleased with her. What is the most accurate reading?

A. She has reached competence in the role, and the sensible move is to stay and consolidate.

B. She is likely underpaid relative to the market and should test it by interviewing elsewhere.

C. Ease is the signal to check three things — whether she could have done this work eighteen months ago, whether anyone there is teaching her anything, and whether her scope has grown — and to change the work rather than work harder if the answers are no.

D. Her manager's satisfaction confirms she is performing well, so the discomfort she feels is misplaced.

84 · 9 min

The one-pager that gets a decision

THE ONE THING TO KEEP

Lead with the recommendation, its cost and the decision needed by when — and attach to every number its definition, date, sample and uncertainty, because stating uncertainty is what makes a reader trust the rest.

Analysis is not a decision

A great deal of good work dies in a document that describes what was found and never says what to do. The reader — busy, three levels up, holding a budget — reaches the end without knowing what is being asked of them, and files it.

The skill that converts technical work into influence is the ability to write a page that produces a decision. It is learnable, it is rarer than it should be, and it is the most reliable route from doing good work to being trusted with more of it.

Put the ask first

Technical writing is trained backwards: method, results, then conclusion. Decision writing inverts it, because the reader may stop at any point and you want them to have the answer before they do.

Compare the same work in two openings.

> *"We analysed churn across the last four quarterly cohorts using a survival model, controlling for plan type and acquisition channel. The results indicate…"*

Nobody above your manager reads past that.

> *"**Recommendation:** move the annual renewal reminder from 7 days before expiry to 30 days. Expected effect: 1.5 to 3 percentage points of annual churn, worth ₹40 to ₹80 lakh a year. Cost: about two engineering weeks. **Decision needed by 20 October** to make the January cohort."*

Everything after that is evidence for a reader who wants it. Both documents contain the same work; only one of them will be acted on.

The structure

1. **Recommendation.** One or two sentences.
2. **What it costs and what it is worth**, with numbers.
3. **The decision needed, from whom, by when.**
4. **The evidence**, briefly, with the strongest thing first.
5. **What would change my mind**, or what we are uncertain about.
6. **Alternatives considered and why they were rejected** — this is what separates a recommendation from an opinion.
7. **Appendix** for the method, the queries, the caveats.

One page above the appendix. If it will not fit, the recommendation is not clear enough yet.

One page, in this order

1. Recommendation	one or two sentences
2. What it costs and what it is worth	with numbers: 1.5 to 3 points of churn, 40 to 80 lakh, two engineering weeks
3. The decision needed, from whom, by when	decision needed by 20 October
4. The evidence, strongest thing first	each number with its definition, date, sample and uncertainty
5. What would change my mind	or what we are uncertain about
6. Alternatives considered and why rejected	a recommendation, not an opinion
7. Appendix	method, queries, caveats

The reader may stop at any point, so the ask comes first. Section six is what separates a recommendation from an opinion, and if the page will not fit above the appendix the recommendation is not clear enough yet.

Number discipline

Every number in a decision document carries four things: what it measures, when, on what sample, and how uncertain it is.

> *Churn is 12%.*

versus

> *Churn was **12.1%** for the January cohort (n=400, ±3 points at 95%), where churn means no renewal within 30 days of contract end.*

The second is longer and it is the one people trust, because a reader can tell what would break it. Stating uncertainty raises your credibility rather than lowering it: you have shown you know the difference between what you measured and what you know, which is exactly what a decision-maker is trying to establish about you.

Anticipate three questions

Every senior reader asks the same three, so answer them before they do:

- **How do we know?** One line on the data and its limits.
- **What does it cost?** Money, time and opportunity.
- **What happens if we do nothing?** The most commonly omitted section and often the most persuasive one.

The meeting

A pre-read circulated 24 hours in advance beats a presentation, because people can think. Some organisations go further and read the document silently at the start of the meeting — a practice Amazon is known for — precisely because slides let a weak argument sound strong.

If you must present: the first slide is the recommendation, the last slide is the recommendation, and the middle is evidence you may skip if the room has already agreed. Never build to a conclusion. There is no suspense in a budget meeting.

Email, briefly

Same rules, smaller. The **subject line contains the ask**: "Decision needed by Friday: renewal reminder timing". The first line repeats it. The detail follows. If there is a deadline, it appears twice.

The honest limitation

Good writing does not defeat organisational politics. A perfectly argued memo proposing something a director does not want will sit unanswered, and you will not be told why.

Two things partly compensate. **Know whose decision it actually is** before writing — a document addressed to nobody in particular is decided by nobody in particular. And **find out what that person is measured on**, because a recommendation framed against their target is a different document from one framed against yours, even when the analysis is identical.

BEFORE YOU MOVE ON

An analyst's memo opens with three paragraphs of methodology before reaching a recommendation on the final page. Why does this reliably fail with senior readers?

A. Senior readers do not understand statistical methods and are alienated by them.

B. Methodology sections invite challenges to the analysis that distract from the conclusion.

C. A reader may stop at any point, so a document that withholds the ask until the end is one that most readers finish without knowing what is being requested of them.

D. The document is too long, and cutting it to one page would solve the problem.

85 · 9 min

Getting promoted, and the case somebody has to make

THE ONE THING TO KEEP

Promotion is argued for you in a room you are not in, using evidence you supplied, and it is granted for work already being done at the next level rather than as a reward for the current one.

The room you are not in

At most companies of any size, promotion is not decided by your manager. It is decided in a **calibration meeting** where several managers argue for their people against a limited number of slots, using a written case each has prepared.

You are not there. Your manager is, holding whatever they can remember or find. Everything below follows from that single fact: your job is to make their document easy to write and hard to argue with.

The thing almost everybody gets wrong

Promotion is generally granted for **work you are already doing at the next level**, not as a reward for doing your current level well.

This is the most commonly misunderstood mechanism in a career, and it explains the frustration of the person who has been excellent for three years and never moved. They have been excellent at their level. The case being made in the room has to say "she is already operating as a senior engineer and we should recognise it", and if the evidence only shows outstanding mid-level work, there is nothing to say.

The practical consequence is a sequence, not a request:

1. **Find out what the next level actually means here.** Many companies publish a levelling guide; ask for it. Where none exists, ask your manager to describe the difference in behaviours, and write down their answer.
2. **Do that work visibly for two or three quarters.** Take the ambiguous problem, run the project across teams, review other people's work, be the person who wrote the plan.
3. **Then ask**, with the evidence assembled.

The packet

Keep a running document from your first month. One dated entry per meaningful thing:

- What the situation was and what you did.
- What changed, with a number where one exists.
- Who benefited and who said so — a quoted sentence from a stakeholder is worth a paragraph of your own description.
- A link to the artefact: the pull request, the document, the dashboard, the incident review.

Ten minutes a fortnight. At review time you hand your manager a page and they write the case from it in twenty minutes instead of reconstructing your year from memory, which is what they would otherwise do and which systematically favours whoever was most recently visible.

Ask the two questions that get real answers

"What would I need to be doing consistently for you to put me forward, and what would the case say?" This forces a specific answer. "Keep doing what you are doing" is not one, and it is worth saying so politely: *"Could you name one or two things a senior person here does that I am not doing yet?"*

"When are promotion decisions made, and when should my case be ready?" Cycles usually close one to three months before anything is an-

nounced. Making your case a fortnight after the calibration meeting is a year lost to timing.

Visibility is not self-promotion

Many people, particularly those raised to believe that good work is noticed, find this section uncomfortable. It is worth separating two things: boasting, which nobody likes, and **making work legible**, which is part of the job.

Legibility looks like this:

- A short weekly or fortnightly written update to your manager: shipped, in progress, blocked. Three bullets.
- Presenting your own work at the team demo rather than letting somebody summarise it.
- Writing the document instead of leaving it to whoever volunteers.
- Naming other people's contributions explicitly, which costs you nothing, is honest, and makes you the person others want credit from.

The credit problem is structural: work done by people who do not talk about it gets attributed to people who do, and this falls hardest on those least comfortable with claiming it. The remedy is not to become louder. It is to write things down, with dates and names, so that the record does the claiming for you.

When it does not happen

After two cycles of "next time", ask directly and specifically: *"Is the constraint my performance, the level's requirements, or the budget?"* A good manager answers honestly, and the three answers imply completely different actions.

If it is performance, you now have something to work on. If it is the level's requirements, ask what evidence is missing. If it is budget or headcount — which it very often is — then no amount of further excellence changes it, and the honest options are to wait deliberately with an agreed date, or to move, using the arithmetic from the lesson on how pay moves.

The honest limitation

Promotion is partly a function of company growth. A company that is not adding scope has few senior slots, and they open when somebody leaves. In a flat or shrinking organisation, outstanding work and a perfect packet can both be true and produce nothing for two years.

That is not a reason to stop building the evidence — it travels with you and makes the case at the next employer — but it is a reason to distinguish between a problem you can solve and a queue you are standing in. Waiting for a slot is a legitimate choice. Waiting for one without knowing that is what you are doing is not a choice at all.

BEFORE YOU MOVE ON

An engineer has had three years of strong performance reviews at mid level and has been passed over twice. Their manager says "keep doing what you are doing". What is the most useful reading?

A. The manager is not advocating for them and they should escalate to skip-level management.

B. Consistent strong reviews at the current level are evidence of doing the current job well, which is not what the promotion case has to claim — so the next step is to establish what next-level scope looks like here and do it visibly before the next cycle.

C. Three years without promotion means the company undervalues them and they should leave immediately.

D. They should ask for a pay rise instead, since the level is clearly blocked.

CASE STUDY · MODULE 9

The postcode feature, asked for on a Thursday

Wanjiru, two years into an analyst role at a lending startup in Nairobi, four days before an investor demonstration.

Default rates had risen for two quarters. On Thursday afternoon the head of product asked for three features to be added to the credit model before Monday's demonstration: the applicant's postcode, the price band of the handset they applied from, and the applicant's first name, which somebody had noticed "seems to carry signal".

Nobody in the room was being malicious. Everybody was under pressure and the request took eleven seconds to make.

This is the shape these requests actually take. Nobody asks an analyst to build something obviously wrong. They ask, pleasantly, under a deadline, for something that sounds reasonable when said quickly — and the person being asked has to produce an objection in real time, in front of people they like, with no preparation. That is why the advice is to decide the line before the meeting, because the meeting is not where anybody thinks clearly.

Why it was hard, and it was hard

She knew that postcode in Nairobi is close to a proxy for both income and ethnic group; that handset price band is a proxy for income directly; and that a first name in Kenya carries ethnic information as plainly as it does anywhere in the world. She also knew that the system declines people with no explanation and no appeal route, and that nobody had ever measured the decline rate by ward.

Against that: she had been there two years, she had no savings to speak of, the demonstration was Monday, and refusing on Thursday makes you the person who blocked an investor meeting. "This is unethical" is a sentence that moves nobody in a room where everyone believes themselves to be reasonable.

What she did, in order

A clarifying question that surfaced the risk without accusing anyone. *"If a declined applicant asks us why, what do we tell them?"* The room went quiet for a moment, because nobody had an answer and everybody understood the question.

The version she could build. Structured application fields and repayment history, with the accuracy cost measured rather than asserted, delivered by Sunday.

Three sentences in writing to her manager, naming the specific risk — a decline decision the company cannot explain, on features that stand in for ethnicity — rather than writing a manifesto. A verbal objection did not happen; a written one is visible to somebody accountable.

Escalation to the one person with authority, a compliance lead hired two months earlier whom most of the team regarded as an obstacle and who turned out to be delighted to hear about a problem before it shipped rather than afterwards.

The measurement that ended it

She did not win the argument by arguing. Over Friday she measured both models and produced one table: adding the three features moved the ranking metric by nine tenths of a point, and the decline rate in two wards went from near-parity to nearly double.

That table ended an eleven-minute meeting.

It is worth noticing what the table cost her. It was a day of work she had not been asked to do, on a weekend before a demonstration, to produce evidence for a position she already held. The alternative — asserting the position without the numbers — was free and would not have worked, because a room under deadline pressure discounts an objection it cannot check and cannot discount a decline-rate breakdown by ward.

There was also a rule underneath all of this that nobody in the meeting had mentioned. Several jurisdictions require that a person be reachable when a

significant decision about somebody is automated, and the company's product declined applicants with no explanation and no appeal route at all. That was not a feature request; it was a gap that existed before Thursday and would have existed if the meeting had never happened.

STOP HERE

Before turning the page: what would you have done, and what would it have cost if you were wrong? Write it down. The point of the next section is to compare.

WHAT HAPPENED

The demonstration used the smaller model, and the nine-tenths of a point was never mentioned by anybody. The first-name feature was dropped permanently. Postcode returned six months later in a different form — distance to the nearest branch, with a written justification, and a decline-rate-by-ward dashboard that Wanjiru built and that is reviewed monthly. The compliance lead later said the three-sentence email was what allowed her to act, because it made the request a documented decision rather than a corridor conversation. Wanjiru's own account is less heroic than it sounds from outside: she was frightened on the Thursday, she had no reserve to fall back on, and the thing that made it survivable was having decided her line before the meeting rather than during it.

Why did the decline-rate table end the argument when the ethical objection had not?

Because it named a specific, checkable harm with numbers attached, which a room can act on, rather than a judgement, which a room can dispute. "This will decline people in two wards at twice the rate and we cannot explain why" is a statement about the product; "this is unethical" is a statement about the people in the room, and it invites defence rather than change.

What did putting the objection in writing achieve that the meeting did not?

It made the decision visible to somebody accountable and gave the compliance lead something to act on. It also protects the person raising it: a verbal objection is deniable and, in the version of this story where the feature ships anyway, the written record is the difference between having objected and having been present. Three sentences naming the specific risk is the right size; a long document is not.

She had no savings. How does that change what this lesson can honestly claim?

It changes how far the advice reaches. Asking the question, offering the buildable version and writing three sentences cost her nothing and were available regardless. Refusing outright, if the feature had shipped anyway, is a decision that depends on being able to afford to leave — which is why three months of expenses is described in this course as a career asset rather than a personal finance tip.

MODULE 9 · TEST

Check what stuck

6 questions, one right answer each. This one is for you, not for a record: answers and the reason behind each are at the back. If two questions from the same module go wrong, re-read that module before the exam rather than after.

1 What should the first four weeks in a new data role be spent on?

A. Documenting the gaps you find, so the team has a record of what is broken

B. Reproducing the numbers the team already reports, from source

C. Proposing the changes you can see are needed while your view is still fresh

D. Reading the codebase from the top so you understand the system as a whole

2 You find a hard-coded exception for one customer in a pricing function. What do you do before touching it?

A. Write a test capturing the current behaviour and then simplify the branch

B. Ask the team in a public channel whether anybody recognises the customer

C. Run `git log -S` on the strange value to find the commit that introduced it

D. Leave it and route around it, since undocumented conditions are rarely safe to change

3 A nightly job scheduled for 00:30 local time produces one missing day and one doubled day, in the same fortnight every year. Why?

A. The upstream source changes its export format at the start of each financial quarter, without notice.

B. Local time is not monotonic, so the daylight-saving change skips or repeats the hour.

C. The orchestration tool retries failed runs at a fixed offset from midnight.

D. Leap seconds are applied at midnight and shift the partition boundary by one day.

4 Applying the three-question test, which of these tasks is most exposed to automation?

A. Deciding what counts as a correct answer for a new customer-facing feature

B. Negotiating with a source team about how a field should be recorded

C. Writing the SQL for a well-specified weekly report from a documented schema

D. Establishing why the Mexico figures arrive three days later than everything else

5 Which of these is genuinely worth tracking monthly, because it changes decisions rather than opinions?

A. Deprecation schedules for the model versions your production system calls

B. Benchmark results published for newly released models in your general area of work

C. Funding announcements from companies building tools adjacent to yours

D. Framework releases that add features similar to ones you already use

6 Somebody has performed excellently at their level for three years and has not been promoted. What is the most likely explanation?

A. Their manager has not been asked directly and is waiting to be approached.

B. Promotion is granted for work already being done at the next level.

C. Their achievements were not communicated widely enough across the organisation.

D. The company's promotion cycle rewards tenure, and three years is usually too short.

EXAMINATION

Working In AI — final examination

This paper covers the whole course: the employer types and roles that exist and how to read an advert for the job it really is; what each role does in an ordinary week; the evidence a reviewer looks for and how to produce it; the routes that put a person in front of your work; using a model in your own search without destroying the thing being judged; every stage of a hiring process from the recruiter screen to the signed offer; what is actually in an offer, and the contract and equity terms underneath it; freelancing priced as a business; and the first ninety days and the ten years after.

You will be given 25 questions drawn at random from a larger pool, so no two attempts are the same paper. The pass mark is 70 per cent, and passing opens the next course in the track. There is no timer — take the time you need, and read each question twice. If you do not pass, you may retake after thirty minutes with a fresh set of questions, as many times as you need; nothing about a failed attempt is recorded against you. A teacher can open the next course for you at any point regardless of this paper, and that is recorded as a teacher's decision rather than disguised as a pass.

Every question tests a mechanism the course explains rather than a definition you could look up. After you submit, each question shows why the right answer is right, including for the ones you got right.

On the website a sitting is 25 questions drawn from this pool and the pass mark is 70%. There is no time limit, there and here. Passing opens the next course in the track.

1 `1. The terrain: who hires, for what, and where you fit`

Across data analysis, data engineering, ML engineering and AI engineering, what does the largest share of the working day have in common?

A. Something is wrong with the data and you have to find out why

B. Communicating results to stakeholders who lack technical background

C. Choosing between competing tools and frameworks for a given task

D. Writing and reviewing production code under version control

2 1. The terrain: who hires, for what, and where you fit

What is the durable value of a year of annotation or model-evaluation work on a platform like Outlier, Appen or Toloka?

A. A promotion track from labelling into research engineering at the same company
B. Failure-mode intuition and the vocabulary of data quality, taken elsewhere
C. A reference from a well-known AI company that recruiters recognise
D. Enough accumulated income to fund a period of full-time study afterwards

3 1. The terrain: who hires, for what, and where you fit

Which single interview question best reveals where a company sits on the data maturity ladder?

A. How many people are on the data team, and how is that team structured?
B. Which tools are you using for orchestration and transformation?
C. What are the biggest data challenges the team is facing this year?
D. Where does the number on your main dashboard come from, end to end?

4 1. The terrain: who hires, for what, and where you fit

A degree buys three separate things. Which of them is a genuine hard gate rather than a preference?

A. Legal eligibility, where immigration rules name a recognised qualification
B. The mathematics, which is difficult to acquire outside a university setting
C. Credibility with hiring managers at product companies and start-ups
D. Access to campus placement and graduate schemes at large employers

5 1. The terrain: who hires, for what, and where you fit

What is the most common way a promising year of part-time study produces nothing hireable?

A. Choosing a lane that does not match the local job market's actual openings
B. Spending too long on fundamentals before attempting anything practical
C. Studying three lanes at thirty per cent each, because it feels prudent
D. Building projects that are too small to demonstrate real technical ability

6 1. The terrain: who hires, for what, and where you fit

The same advert for a junior role has been reposted every month for six months. What does that most usefully tell you?

A. The company is growing quickly and hiring several people into the same role
B. The posting is being refreshed automatically to keep it near the top of listings
C. The salary band is below market and candidates are declining offers
D. Either people keep leaving the role, or the specification is impossible

7 1. The terrain: who hires, for what, and where you fit

What is the practical difference between working at a global capability centre and at a services firm billing the same foreign client?

A. The capability centre pays at the parent company's home-country rate
B. At the capability centre you are staff of the parent's own subsidiary
C. The services firm offers deeper work on a single system over several years running
D. The capability centre hires through the parent's head-office recruitment process

8 2. The roles, one at a time

A stakeholder asks how many users churned last month, and the number is rejected three times over three days. What was missing at the start?

A. An agreed definition of churn, written down before the query was run
B. Access to the subscription system, which holds the authoritative records
C. A dashboard, so the stakeholder could answer the question themselves
D. A statistical treatment of the uncertainty in a single month's figure

9 2. The roles, one at a time

Analyst work is described as exposed to automation for a specific mechanical reason. What is the defence?

A. Producing queries faster than a model can, using saved templates and snippets
B. Moving into dashboard design, which requires human judgement about layout
C. Owning the definitions and being accountable for the number
D. Specialising in a warehouse dialect that models handle less reliably

10 2. The roles, one at a time

An analytics engineer has built 180 tested, documented models in four months. Analysts are still querying the raw tables. What does that mean?

A. The models need much better documentation, so that analysts can discover what exists
B. The analysts need training, since the models are demonstrably more reliable
C. The migration will happen naturally once the raw tables are deprecated
D. The work has produced nothing, because the role's output is other people's speed

11 2. The roles, one at a time

An applied scientist reads a paper claiming a large gain over prior work. What is the first thing to establish?

A. What it was compared against, and whether that baseline was tuned as carefully
B. Whether the code and the trained weights have been released publicly
C. Whether the result has since been replicated by an independent group with no shared authors
D. How large the training compute budget was relative to a normal team's

12 2. The roles, one at a time

Which requirement written by an AI product manager is actually buildable and testable?

A. The assistant should feel noticeably more accurate than the current version
B. The assistant must be smarter about which document it chooses to answer from
C. On the 300-question set, 90% of answers must cite a real source document
D. The assistant should handle edge cases more gracefully than it does today

13 2. The roles, one at a time

Eighteen months into forward-deployed work there are eleven forks of the product and no customer can be upgraded. What went wrong?

A. The product was not mature enough to be deployed at several different customers at once
B. Each reasonable request was built bespoke instead of pushed back into the product
C. The team accepted customers whose data was too different to support properly
D. Engineering did not allocate enough capacity to support field deployments

14 2. The roles, one at a time

An assistant that passed every check last month is now failing them, and nobody on the team changed anything. What is the most likely cause?

A. Retrieval has degraded as more documents were added to the index
B. The evaluation set has drifted because new cases were appended to it
C. The provider updated the model version behind the same API name
D. Traffic growth has pushed latency up, causing timeouts on longer answers

15 3. The evidence: what gets tested, and how to prove you have it

You understood every minute of a course and cannot build anything from an empty file. What does the drill of deleting a finished tutorial project and rebuilding it fix?

A. Recall of the syntax, which fades quickly without deliberate spaced repetition of it
B. Confidence, by proving the material was genuinely understood the first time
C. Familiarity with the tooling, which the tutorial had configured in advance
D. The two skills a tutorial removes: deciding what to do next, and getting unstuck

16 3. The evidence: what gets tested, and how to prove you have it

What separates SQL practice that makes you employable from grinding two hundred puzzle problems?

A. Checking every answer a second way, against a total or a spot-check
B. Practising on the specific warehouse dialect named in local job adverts
C. Learning window functions thoroughly, since they appear in most interviews
D. Working through problems under a timer to build speed under pressure

17 3. The evidence: what gets tested, and how to prove you have it

A colleague cannot reproduce a result you obtained in a notebook. What is the most likely cause?

A. A different library version producing a slightly different numerical result
B. Cells were run out of order, so the state your code needs is not in the file
C. A random seed was left unset, so the sampling differs between machines
D. The source data changed between your run and theirs, without either of you noticing it

18 3. The evidence: what gets tested, and how to prove you have it

You have just realised you committed a `.env` file containing a live API key to a public repository. What is the correct order of actions?

A. Delete the file and commit, then rotate the key once the history is clean
B. Rewrite the history with git filter-repo, then rotate the key afterwards
C. Revoke and rotate the key immediately, then clean the history
D. Make the repository private first, then rotate the key and clean the history

19 3. The evidence: what gets tested, and how to prove you have it

You built a proper pipeline and a dashboard to replace a team's spreadsheet, and they kept using the spreadsheet. Why?

A. The team distrusts systems they did not build and needs time to adjust
B. The spreadsheet lets them change an assumption and see the effect in two seconds
C. The dashboard was not integrated into the tools they already have open
D. Nobody had been trained on the new dashboard before it was released to the whole team

20 3. The evidence: what gets tested, and how to prove you have it

You are using a model to grade outputs because hand-grading does not scale. What must you do before trusting a large run?

A. Use a larger model as the judge, since judging is harder than generating
B. Hand-label about fifty cases and measure how often the judge agrees
C. Average three separate judge runs to reduce variance in the scores
D. Ask the judge to explain each verdict, which improves grading accuracy

21 3. The evidence: what gets tested, and how to prove you have it

A dataset on a public repository has no licence file at all. What may you do with it?

A. Use it freely for a portfolio, since no restriction has been stated
B. Use it for anything non-commercial, which is the default for unlicensed work
C. Use it with attribution, which satisfies the normal expectation of the uploader
D. Nothing without permission, because absence of a licence is not permission

22 3. The evidence: what gets tested, and how to prove you have it

Why does a "what it gets wrong" section raise a reviewer's confidence in the rest of a project page?

A. It shows the writer knows how to look for problems, which is what is being bought
B. It pre-empts the criticism an interviewer would otherwise raise in the room
C. It demonstrates that the project was tested against realistic edge cases and failures
D. It shortens the review, because the reviewer can stop looking for weaknesses

23 4. Getting seen: the routes that put a person in front of your work

In which market is applying widely to advertised roles the correct strategy rather than a way of feeling busy?

A. Product start-ups of twenty to three hundred people, which read applications
B. Remote roles at foreign companies, where a written application is the only channel
C. Large services and outsourcing firms hiring in batches with standardised tests
D. Mid-sized enterprises with internal data teams and formal hiring processes

24 4. Getting seen: the routes that put a person in front of your work

Your tracker shows plenty of first conversations and almost no technical stages. What is most likely wrong?

A. Your CV overstates your level, so the conversation exposes a mismatch
B. Something in the first fifteen minutes is not landing, and it is quickly fixable
C. Your technical skills are below the bar for the roles you are targeting
D. You are applying to companies whose hiring processes are much longer than the average one

25 4. Getting seen: the routes that put a person in front of your work

Which LinkedIn headline does the job a headline is for?

A. Aspiring data professional seeking opportunities in analytics and AI
B. Data enthusiast passionate about turning data into actionable insight
C. Learning data engineering · building public pipelines with dbt and DuckDB
D. Graduate with certifications in SQL, Python, Power BI and machine learning

26 4. Getting seen: the routes that put a person in front of your work

Which piece of public writing produces the most return for the time it takes?

A. A comprehensive guide to a topic you have been studying for a few weeks
B. An opinion piece about where the field is heading over the next two years
C. A summary of the week's model releases, published on a regular schedule
D. The obscure error that cost you nine hours, written up the next morning

27 4. Getting seen: the routes that put a person in front of your work

What is the specific risk of the bridge-job route, and what guards against it?

A. Becoming the unofficial spreadsheet person for ever; agree a review date in writing
B. Learning tools nobody else uses; insist on the same stack the central data team uses
C. Being paid below market for the data work; ask for a salary review at six months
D. Losing technical depth while doing the operational job; keep a side project running

28 4. Getting seen: the routes that put a person in front of your work

Which line in a one-page scope for a first paid job saves you most often?

A. The line stating the total price and the payment schedule
B. The line describing the deliverable in a single sentence
C. The line listing what the client provides, and by what date
D. The line stating how many rounds of revision are included in the price

29 4. Getting seen: the routes that put a person in front of your work

In a community — a city meetup, a language Discord, a tool's forum — how does the outreach sequence change?

A. You introduce yourself formally to the moderators before contacting members
B. You send the same four-sentence message, but to several members of the group at the same time
C. You post your work publicly and let interested people contact you first
D. You are useful in public for months, so a later message comes from somebody they recognise

30 5. Using AI in your own search

What is the free thirty-second test that tells you whether your CV will parse correctly?

A. Upload it to a scanner tool and read the compatibility score it returns
B. Open the PDF, select all, copy, and paste into a plain text editor
C. Print it in greyscale and check that every section heading is still legible
D. Send it to yourself and open it on a phone to check the layout holds

31 5. Using AI in your own search

In the twenty-minute tailoring workflow, which part is worth most of the benefit?

A. Rewriting the top third: the summary and the first two bullets of the recent role
B. Reordering the skills section so the advert's tools appear at the front
C. Rewriting every bullet on the page in the advert's vocabulary rather than your own
D. Adding a cover letter that addresses each stated requirement in turn

32 5. Using AI in your own search

A service offers to fire two hundred applications a week on your behalf. What is the concrete cost beyond the fee?

A. A recruiter seeing six unsuitable applications from you reads it as noise
B. The service cannot answer knockout questions, so applications are incomplete
C. Applications are submitted with generic formatting that parses badly
D. Employers share application data, so the pattern is visible across companies

33 5. Using AI in your own search

In forty minutes of research before a call, which source is most underused and most informative?

A. The company's press coverage over the last twelve months
B. The founders' interviews and conference talks about strategy
C. Employee reviews on Glassdoor, AmbitionBox or Indeed
D. The company's full list of currently open roles

34 5. Using AI in your own search

You have been sent a one-way recorded interview. What should you optimise first?

A. Facial expression and eye contact, since some systems score them
B. Audio quality and answer structure, because the score comes from the transcript
C. Answer length, using the full time limit available for each question
D. Background and lighting, which shape the first impression of your professionalism

35 5. Using AI in your own search

Which line of generated data code is most dangerous, and why?

A. A regular expression you did not write, because it may not match every case

B. A Dockerfile skeleton, because it may pin an outdated base image version

C. A groupby that drops null keys by default, because totals are quietly short

D. A plotting incantation, because the resulting chart may mislead the reader visually

36 5. Using AI in your own search

Which sentence should be cut from an application, and what is the test?

A. "I rebuilt our stock report after finding the depot codes did not match."

B. "I have a passion for using data to drive impact" — it is true of anybody

C. "I taught myself SQL in the evenings while working full time in retail."

D. "I have not used your warehouse, but I have used a similar one for two years."

37 6. The interview, from first screen to signed offer

You gave a poor answer in round two and realised afterwards what the right one was. What should you do?

A. Nothing — raising it draws attention to a weakness the panel may have forgotten

B. Mention it briefly in the final round, if the same topic happens to come up again

C. Ask the recruiter whether the panel would like a written follow-up first

D. Send it in writing before the next round; panel notes already record the answer

38 6. The interview, from first screen to signed offer

Which take-home should be declined rather than completed?

A. A four-hour exercise with a deliberately messy dataset and an open brief
B. A two-hour exercise sent before any human conversation has taken place
C. An eight-hour exercise with a week's turnaround and a clear scoring rubric attached
D. A brief asking you to produce the report the company actually needs this month

39 6. The interview, from first screen to signed offer

In a design conversation, which move signals seniority most strongly at any experience level?

A. Explaining what you would not build, with the mechanism for rejecting it
B. Sketching the full architecture including caching, queuing and monitoring
C. Naming the specific vector database and embedding model you would use
D. Estimating throughput and storage requirements at ten times current traffic

40 6. The interview, from first screen to signed offer

Why is revenue almost never a usable metric for a specific feature?

A. It is measured monthly, which is too slow for a feature evaluation
B. It is not attributable, because forty other things move it at the same time
C. It is easy to game by shifting the timing of discounts and renewals
D. It is insensitive, because feature-level changes are small relative to totals

41 6. The interview, from first screen to signed offer

In an experiment design question, which decision invalidates everything after it if you get it wrong?

A. The unit of randomisation — user, session, device, or something coarser
B. The choice of primary metric, which determines what the test can conclude
C. The duration, which must cover at least one full weekly cycle
D. The stopping rule, which must be agreed before any data is collected

42 6. The interview, from first screen to signed offer

Which failure story would a good interviewer score highest?

A. A project that failed because a supplier changed a format without notice
B. A weakness framed as caring too much and working too hard on a project
C. A report wrong for two months because of a join, found by finance rather than by you
D. A deadline missed because the requirements were changed three times during that quarter

43 6. The interview, from first screen to signed offer

Which of your questions most reliably reveals whether an advertised AI role involves any AI?

A. How large is the data team, and how is it structured across the business?
B. What is in production that uses a model, and when did it last change?
C. What models and frameworks does the team use most often day to day?
D. What would success in this role look like at the end of the first year?

44 6. The interview, from first screen to signed offer

You have a written offer and want to negotiate, with no competing offer. What is the technique?

A. Name a number well above your target, so there is room to be negotiated down

B. Ask once, name your figure, and then stop talking

C. Ask for time to consider, then negotiate several items in a single email

D. Mention that you are in process elsewhere, without naming the company

45 7. The money, and the paperwork underneath it

A company that raised $50 million with a 1× preference sells for $60 million. What do employees share?

A. $60 million, divided by everybody's proportional shareholding

B. $50 million, with the remainder retained by the company

C. $30 million, because preferences are conventionally settled at half their value

D. $10 million, because investors are repaid before common shareholders

46 7. The money, and the paperwork underneath it

Why do people leave a company with four years of vested options and receive nothing?

A. Vesting schedules usually require a fifth year for the final tranche to qualify

B. Options are cancelled automatically when an employee resigns rather than being made redundant

C. The company repurchases vested options at the strike price on departure

D. The exercise window is usually ninety days, and exercising costs cash plus tax

47 7. The money, and the paperwork underneath it

A foreign company with no entity in your country wants to hire you through an employer of record. What must you check?

A. That your notice, leave and severance come from your own country's law
B. That the arrangement converts to direct employment within a fixed period
C. That the employer of record is registered in the same country as the client
D. That the client's own employment policies apply to you in the same way

48 7. The money, and the paperwork underneath it

You are engaged as a contractor. Which pattern would most legal tests treat as employment?

A. You invoice monthly and are paid on thirty-day terms like other suppliers
B. You work for three other clients and set your own hours around them
C. You attend their standup daily, use their laptop, and work only for them
D. The engagement has run for two years across several separate contracts

49 7. The money, and the paperwork underneath it

You live in Nairobi and your only client is in Berlin. Where is that income generally taxed?

A. In Germany, because that is where the paying entity is registered
B. In Kenya, because tax generally follows where you are resident
C. In both, at full rates, unless you register a company in one of them
D. In neither, until the total exceeds the German reporting threshold

50 7. The money, and the paperwork underneath it

Which relocation route rewards planning most and is most often overlooked?

A. The employer-sponsored lottery route, where early registration improves the odds
B. A master's degree abroad followed by a post-study work permit
C. A freelance or self-employment visa, which avoids needing a sponsor
D. Joining a multinational locally and transferring internally after a year or two

51 7. The money, and the paperwork underneath it

Comparing two offers, one insures only you and the other insures your family. How does that enter the arithmetic?

A. As avoided spending, at what comparable family cover would cost you
B. As part of the total package figure quoted by each employer
C. As a tie-breaker only, after guaranteed and expected cash are equal
D. It does not, because benefits vary too much to be compared meaningfully

52 8. Freelancing and contracting, as a business

A freelancer is profitable on paper and cannot pay rent in June. What is the mechanism?

A. Clients dispute invoices more often than employees expect them to
B. Tax instalments fall due before the corresponding income has arrived
C. March work is invoiced on 31 March and paid in late April at the earliest
D. Expenses are incurred at the start of a project and recovered only at the end

53 8. Freelancing and contracting, as a business

A freelancer oscillates between being fully booked and having no work three months later. What fixes it?

A. Taking longer engagements, so the gaps between projects are less frequent

B. Raising rates, so fewer billable hours are needed to cover the same costs

C. Keeping two or three clients at once, so no single ending empties the calendar

D. A fixed weekly selling slot kept regardless of how busy you are

54 8. Freelancing and contracting, as a business

Which introduction actually produces referrals?

A. "I am a freelance data consultant working across analytics and engineering."

B. "I fix broken reporting for e-commerce companies under fifty staff, in about six weeks."

C. "I help businesses become data-driven and make much better decisions with their own data."

D. "I am available for data engineering, analytics, dashboards and AI integration work."

55 8. Freelancing and contracting, as a business

A client asks, mid-project, for one more small thing. What is the move?

A. Say yes and absorb it, since goodwill on a first project is worth more

B. Say it is out of scope and cannot be done within the agreed timeline

C. Say yes, then send a short change note with the cost and the new date

D. Say you will look at it during the revision rounds already included

56 8. Freelancing and contracting, as a business

Which early signal most reliably predicts that a client will not pay on time later?

A. They negotiated the price down before agreeing to the proposal

B. They asked for a detailed breakdown of how the estimate was reached

C. They have no written process for approving supplier invoices

D. They paid the first milestone eleven days late

57 8. Freelancing and contracting, as a business

Which habit prevents the most common freelance financial disaster?

A. Moving a fixed share of every payment into a separate tax account on arrival
B. Incorporating a limited company as soon as the first client is signed
C. Buying accounting software before the first invoice is issued
D. Setting aside a fixed sum each month towards the annual tax bill when it arrives

58 8. Freelancing and contracting, as a business

How should a substantial fixed-price project be quoted when you have not seen the client's systems?

A. At an hourly rate instead, so the risk of a wrong estimate sits with the client
B. With a paid discovery phase first, producing an assessment and a fixed quote
C. With a range rather than a single figure, narrowed once work begins
D. At your best estimate plus a contingency of about ten per cent

59 9. Staying: the first ninety days and the ten years after

You are stuck on something in your second week. What is the professional way to handle it?

A. Keep working on it for a day; a new joiner is expected to find their own way
B. Message the person who wrote the code directly, so you do not disturb others
C. Try for thirty minutes, then ask in a public channel, saying what you tried
D. Collect several questions and raise them together in your weekly one-to-one

60 9. Staying: the first ninety days and the ten years after

You presented a number in a leadership meeting and have just discovered it was wrong. What do you do?

A. Correct it at the next meeting, once you have established the full cause
B. Say so immediately in the same channel, with the corrected figure and the cause
C. Tell your manager privately and let them decide how to communicate it
D. Publish a written post-mortem explaining the failure and the fix in careful detail

61 9. Staying: the first ninety days and the ten years after

Which part of a data job is most durable against automation, and why?

A. Writing correct SQL quickly, since it is the foundation of everything else
B. Building dashboards, which requires design judgement about what to show
C. Knowing the current tooling, which changes faster than models can learn it
D. Being accountable when the number is wrong, which is a responsibility

62 9. Staying: the first ninety days and the ten years after

Two people study equally hard for five years and end up very differently skilled. What most explains the gap?

A. How they allocated time between fast-decaying and slow-decaying knowledge
B. Whether they specialised early or kept their options open across several areas
C. How many tools and frameworks each of them learned over the period
D. Whether their employers gave them access to modern infrastructure

63 9. Staying: the first ninety days and the ten years after

You are asked to score job applicants using features you believe stand in for something unlawful. Which move works best in the room?

A. State plainly that the request is unethical and that you will not build it
B. Ask for the request in writing so that responsibility sits with the requester
C. Escalate to legal before responding, so the conversation happens with authority
D. Ask what you tell a declined applicant, and offer the version you can build

64 9. Staying: the first ninety days and the ten years after

Which question most reliably diagnoses a career plateau?

A. Am I being paid at the market rate for my level and location?
B. Could I have done last quarter's work eighteen months ago?
C. Have I learned a new tool or framework in the past twelve months?
D. Is my title consistent with what people at other companies hold?

65 9. Staying: the first ninety days and the ten years after

Which section of a one-page recommendation is most often left out and most persuasive?

A. What happens if we do nothing
B. The alternatives considered, and the reason each was rejected
C. The method and the queries, which let a reader verify the analysis
D. The uncertainty around the central number, stated explicitly

Answers

Each entry gives the correct option, then what each wrong one reveals about how somebody was thinking. The second part is worth more than the first: a wrong answer here is a named misreading, not a mistake.

Lesson checks

1. The jobs, described as they feel on a Tuesday — B

A — Assumes the hard part of AI work is training models, because that is what courses spend their weeks on.

C — Treats the job as vendor selection — a real task, but one that takes days, not months.

D — Mistakes the visible surface of the system for its main cost, because the prompt is the part you can watch changing.

2. The eight kinds of employer, and how many of each — C

A — Assumes the barrier is evidence quality, when the categories he chose filter mainly on credentials, referrals and fixed graduate intakes.

B — Treats a structural mismatch as a formatting problem, which is the most commonly sold and least useful explanation.

D — Skips over the large local employer categories in favour of the most globally competitive option available.

3. Geography, time zones, and the office you have never heard of — A

B — Assumes the barrier is candidate quality when the binding constraints are a nine-and-a-half hour offset and the absence of a legal entity to employ her.

C — Treats a genuinely damaging schedule as unavoidable when a different target market removes the problem entirely.

D — Solves the entity problem but silently converts a salaried role into contract work with no notice, benefits or severance, which is a large trade presented as a formality.

4. The work that trains the models — C

A — Assumes a ladder exists inside a workforce that is deliberately flat, which is how people lose two years waiting.

B — Relies on the label rather than the evidence; a reviewer cannot tell from a platform name whether she ranked four thousand items thoughtfully or clicked through them.

D — Improves income without changing employability, because a higher task rate produces no artefact anybody outside the platform can inspect.

5. Reading a job advert for what it actually is — A

B — Trusts the title over the stack, which is how people end up in interviews answering questions about a job that was never on offer.

C — Mistakes a common titling habit for organisational confusion, and discards a role that may be a strong entry point.

D — Reads a naming convention as bad faith, when the pay band for this work is often perfectly reasonable.

6. Who sits around you, and how work arrives — B

A — Reads a maturity level as a character judgement; most organisations are here, and it is where the highest-impact early work exists.

C — Assumes advertised work happens; a company reconciling revenue by hand is several rungs below production modelling.

D — Treats a definition and ownership problem as a tooling problem, which is the most common wrong diagnosis in this field.

7. Who actually needs a degree — B

A — Concludes the portfolio is worthless because it failed once, when in fact nobody at the bank ever opened it.

C — Assumes the rejection was a judgement about skill when it was a judgement about a form field.

D — Reads a faster process as a weaker one; startups often test harder, just later and by different means.

8. Choosing a lane you can defend, and changing it later — D

A — Chooses by destination salary rather than by time-to-evidence, and ignores that the deep learning route from zero takes years at six hours a week.

B — The breadth-first plan, which feels responsible and reliably produces a year with nothing finished and nothing hireable.

C — Discards the domain knowledge that is the least replaceable thing she owns.

9. Levels, titles, and why "senior" means nothing on its own — B

A — Treats down-levelling as a judgement of worth rather than the routine mapping of scope onto a ladder he has not previously been measured against.

C — Conflates level with pay; the two are related but the grade is set by scope, and the start-up may well have paid him more.

D — Uses the title as the argument when the ladder is explicitly built to ignore titles, so it will not persuade anybody in the loop.

10. A year on six hours a week — B

A — Blames fatigue within the session, which is real but secondary to what the gaps themselves cost.

C — Reaches for content quality, the explanation that leads to buying another course rather than changing the pattern that is actually costing the time.

D — Concludes the amount is the problem when the distribution is; steady four-hour weeks do produce progress.

11. The data analyst — C

A — Settles delivery format before establishing what is being measured, so the re-work happens anyway once the definition is challenged.

B — Pins down one parameter but leaves the expensive ambiguity — what counts as churned — untouched.

D — Uses plausibility of the result as the check, which means the definition is only discovered when somebody dislikes the answer.

12. The analytics engineer — B

A — Assumes the models are right and adoption is a communication problem, which is the most common self-flattering diagnosis in this role.

C — Forces adoption without fixing the gap, which produces workarounds and resentment rather than trust.

D — Confuses tests passing with the models being useful; a test only checks that the data matches its own assertions.

13. The data engineer — B

A — Assumes retries fix the underlying cause; rate limits often need backoff or a schedule change, not another attempt.

C — Describes a structural failure that did not happen; the schema is fine, and that is precisely why nothing alerted.

D — Names a real but minor cost while missing that decisions are now being made on numbers that are 60% short.

14. The machine learning engineer — D

A — Reaches for drift, which is gradual, to explain a gap that appeared immediately on deployment.

B — Proposes more capacity for a model that already fits the training data extremely well, so capacity is not the constraint.

C — Investigates availability, which would show up as errors and latency rather than as confidently wrong predictions.

15. The AI engineer — A

B — Spends more money on generation for a case where the evidence may never have reached the model.

C — Reduces variability without changing whether the necessary information was present in the context.

D — Manages the consequence rather than diagnosing the cause, and leaves the same failure in place for every other query.

16. The platform and MLOps engineer — B

A — Forces adoption of something unvalidated, which produces compliance rather than usage and hides the mismatch instead of fixing it.

C — Treats generality as the gap when the evidence is that the road was never paved with a real driver on it, so each added feature is another guess.

D — Assumes a knowledge problem, but the teams understand the platform well enough to have decided against it.

17. The applied scientist and research engineer — B

A — Starts from the literature's frame rather than the team's own problem, and delays finding out whether anything sophisticated is needed at all.

C — Commits resources before knowing whether the direction is worth resourcing, which is how research budgets are consumed by directions nobody time-boxed.

D — Lets the model define the test, which guarantees the number is flattering and uninformative.

18. Trust, safety and the policy side of AI — B

A — Uses a single aggregate on a rare-event problem, where accuracy is dominated by the vast majority of posts that were never going to be flagged.

C — Confuses a threshold change with a model improvement; the model is identical and its decay rate is unchanged.

D — Predicts the opposite of the likely effect, since fewer removals generally means fewer appeals.

19. The AI product manager — C

A — Chases a number without asking what the remaining errors do, which is the question that decides whether any threshold is acceptable.

B — Takes a decision that belongs to engineering while leaving the product question — the cost and shape of the 7% — unanswered.

D — Prioritises a feature whose acceptability has not been established, so the estimate is being applied to something that may not be shippable at all.

20. The forward-deployed and solutions engineer — C

A — Adds testing to contain a divergence that will keep growing, so the cost rises with every new customer rather than falling.

B — Prices the symptom; revenue from bespoke work makes the fork problem more attractive to continue, not less.

D — Treats early-stage enterprise software as the anomaly, when arriving incomplete is the normal condition this role exists to handle.

21. When the courses end and you still freeze — B

A — Assumes what is missing is instruction, when what is missing is practice at deciding.

C — Confuses recognising good structure with being able to produce any structure at all under uncertainty.

D — Assumes reading working code teaches the same thing as producing it, when the struggle that was removed was the lesson.

22. SQL, and the six query shapes that cover the job — C

A — A plausible data-quality guess that would shift the number slightly, not multiply it by roughly the average items per order.

B — Reaches for a reconciliation difference between teams, which is common in general but does not explain a clean multiplicative gap.

D — Names a more advanced tool as the fix, when the problem is the row count entering the aggregate, not the aggregate itself.

23. The employable half of Python — A

B — Blames the destination, which would usually raise an error or truncate uniformly rather than affecting only leading zeros and long values.

C — An encoding fault garbles text visibly; it does not selectively remove leading zeros while leaving everything else readable.

D — Explains missing records in general but not the specific pattern of leading zeros disappearing and long numbers changing their last digits.

24. The statistics that come up, and the ones that never do — B

A — Judges sample size by whether it feels large rather than against the difference being sought, which is the mistake the power calculation exists to prevent.

C — Buys a small factor by narrowing the question, leaving the test short by more than an order of magnitude.

D — Reaches a defensible action through the wrong reasoning, and gives up the chance to state what evidence would have been needed.

25. Git, and what a reviewer sees before your code — D

A — The intuitive fix, and the one that does nothing: git keeps every past snapshot, so the key remains readable in the history.

B — Reduces further exposure but leaves a key that was public and very likely already harvested, still valid.

C — The right eventual step, done in the wrong order — it takes time while the exposed key is still live and usable.

26. The terminal, the container, and a cloud bill of zero — C

A — Assumes usage drove the bill, when the pattern of a near-idle service costing tens of dollars points at resources charged by the hour regardless of use.

B — Misstates how free tiers work; most are ongoing monthly allowances, and exceeding them on two requests is implausible.

D — Names a real charge that is normally cents per month, far too small to explain the figure.

27. The spreadsheet skill nobody puts on a syllabus — B

A — Assumes the gap is confidence or skill, when the team may understand the dashboard perfectly and still find it unusable for their task.

C — Treats a design problem as an adoption timeline, which means waiting instead of fixing the thing that is missing.

D — A reasonable diagnostic, but it assumes the deficit is in the data shown rather than in what can be done with it.

28. Evaluation: deciding what "correct" means, in advance — D

A — Treats an agreed number as sufficient without asking what the errors were, which is how a system ships with a small number of severe failures.

B — Reaches for sample size, which matters, while ignoring that the existing failures are already readable and already decisive.

C — Moves the risk onto the customer rather than examining it, and a disclaimer does not reduce the harm of a confidently wrong answer.

29. A portfolio when you have never been paid for this — C

A — Treats deployment as the point; a deployed toy still loses to a messy problem handled thoughtfully.

B — Reads quantity as the red flag, when the real problem is that all twelve answer questions somebody else had already answered.

D — Converts a technical signal into a moral one; the cooperative matters because its data was messy, not because the cause was worthy.

30. Where portfolio data comes from, and what you may do with it — D

A — Treats name removal as anonymisation, when free-text complaints routinely re-identify people and the contractual problem is untouched either way.

B — Values the realism of the data over the judgement question the reviewer is actually being shown.

C — Assumes silence means permission, whereas employment and customer contracts normally assign the data to the employer by default.

31. Writing up a project as a decision record — A

B — Feels like the professional thing to include, but a reviewer deciding about you in ninety seconds gets nothing from it that the first sentence did not give them.

C — Writes for a learner rather than an assessor, and spends the most valuable space on the page teaching rather than evidencing.

D — Optimises for a keyword scan that a human reviewer is not performing, at the cost of the space where judgement would be shown.

32. How people actually get in — B

A — Optimises for a filter that is not the real bottleneck; a keyword-tuned CV still lands in a pile of nine hundred.

C — Treats hiring as a numbers game, when a near-zero conversion rate multiplied by more attempts is still mostly silence.

D — Assumes she was rejected for lacking a credential, when most of those applications were never read by anyone.

33. Running the search as a funnel — B

A — Reads a 75% pass rate as the weak point, when it is well above typical and the sample is three.

C — Draws a conclusion from two finals, where even a strong candidate converts around a third of the time.

D — Correctly reads the conversions but chooses the lever with the worst return, buying four more conversations for another several months.

34. A CV that survives a thirty-second read — C

A — Optimises for attention rather than evidence, and multi-column designs are exactly what text extraction handles worst.

B — Chases the parser instead of the reader, and adds tools he cannot defend in an interview.

D — Deletes the only evidence he has, on a belief about employers that is true of some large filters and false of the people who would actually hire him.

35. The profile a stranger checks in ninety seconds — C

A — Identifies a real formatting constraint but implies the fix is brevity, when the sentence would still say nothing at half the length.

B — Optimises for a keyword-matching behaviour that recruiters use for search, not for the human read that decides whether to open the profile.

D — Fixes the tone by claiming a role not yet held, which fails at the first conversation and costs more than the weak headline did.

36. Writing to strangers, and what to say — B

A — Assumes the silence is about her qualifications, when the messages give the reader nothing specific enough to evaluate her by.

C — Sends her back to the method with the worst response rate, when the problem is the content of the messages rather than the approach.

D — Doubles the volume of a message that was not working, and converts non-response into active irritation.

37. Publishing, and the inbound it creates — B

A — Competes with thousands of near-identical broad guides on a topic where the writer holds no advantage, so nothing directs a searcher to it.

C — Reads well for people who already follow the writer, but answers no question anybody types into a search box.

D — Optimises for a burst of attention that decays in days, rather than for being the answer to a recurring question.

38. A merged pull request as evidence — C

A — Enters the most heavily contested queue in open source, where issues are claimed within hours and maintainer attention is scarcest.

B — Sends an unrequested large diff, which maintainers decline on principle because they inherit the maintenance of a change they did not ask for.

D — Produces volume that maintainers now actively distrust, and demonstrates none of the four things a reviewer is trying to establish.

39. The job next to the data — D

A — Adds a weak credential to a route that is already failing, rather than changing the route.

B — Discards the domain context and internal trust that are her two strongest assets, in exchange for a title.

C — A reasonable use of time, but it competes with thousands of similar projects and ignores the far shorter path she is already standing in.

40. The first client, and how not to lose money on them — A

B — Money changes their incentive slightly but does not create the shared record of who owed what and when, which is what the dispute is actually about.

C — Solves a different and later problem; nothing here has been revised because nothing has been built.

D — Adds an enforcement mechanism to a deadline that was never the issue, and puts the risk on the wrong side of the relationship.

41. What it genuinely helps with, and what it quietly ruins — C

A — Produces a checkable list about your own document, and nothing generated reaches the employer.

B — Creates practice material you verify by running it, and it is never submitted as your work.

D — Risks confident errors, but those are caught by checking against the company's own pages, and the cost of being wrong is one awkward sentence.

42. What an applicant tracking system actually does — B

A — Attributes the failure to a scoring mechanism that mainstream systems do not have, which leads people to buy score-improvement products.

C — Substitutes a taste judgement for a mechanical one; the design may be fine on screen and still be unreadable to the parser.

D — Raises a fair objection to the format's content while missing that the parser extracts nothing from it either way.

43. Rewriting a CV against an advert, and the line — C

A — Claims a tool never used on the reasoning that it is learnable, which is exactly the claim a twenty-minute technical screen is designed to test.

B — Avoids invention but also avoids translation, so a reviewer filtering on warehouse experience never sees a relevant candidate.

D — Uses a hedge word that readers interpret as experience and the candidate intends as exposure, so it fails at the same screen while looking evasive afterwards.

44. Why volume stopped working, on both sides — A

B — Buys volume in the channel whose conversion collapsed, and accumulates unsuitable applications in the systems of companies he may later want.

C — Applies the warm-route rule correctly in general and misses the channels — batch, graduate and public-sector hiring — where the formal process is the only route that exists.

D — Treats a process with roughly a 30% conversion at its best stage as if a single attempt could be made reliable by preparation.

45. Your CV, their take-home, and the terms — C

A — Treats the absence of a rule as consent to disclose another organisation's data to a third party, which is the question the brief was never asked.

B — Addresses training while leaving the disclosure and retention untouched, and the data still leaves the candidate's machine.

D — Assumes column-level masking makes a transaction extract safe to share, when the commercial confidentiality of the dataset is untouched by removing names.

46. Forty minutes of research, and how to check it — A

B — Produces fluent, unlabelled claims about a private company where public information is thin, which is exactly the condition under which fabrication is most likely.

C — Aggregates a sample skewed towards the delighted and the furious, so the number moves with mood rather than with structure.

D — Describes how the company wishes to be seen, and is written to be true of almost any organisation in the sector.

47. Recorded and machine-scored interviews — B

A — Treats length as the defect when the problem is that the answer arrives last, which the transcript scoring reads as failure to address the question.

C — Assumes a human is weighing spontaneity, when the stage is an elimination filter reading a transcript against a rubric.

D — Optimises for the visual analysis that most vendors have moved away from, while leaving the transcript as unstructured as before.

48. Rehearsing with a model that wants to please you — B

A — True and worth knowing, but a candidate who practised badly on accurate questions would still fail for the same underlying reason.

C — Identifies a real risk of over-rehearsal, which comes from repeating fixed wording rather than from having seen the questions.

D — Implies the problem is the freshness of the list, when the format of the practice is what fails regardless of which questions are used.

49. Building a project you can still defend — C

A — Catches style and type problems in code that is already syntactically valid, while the failure mode here is arithmetic that runs perfectly.

B — Re-runs the same generator over the same assumptions, so a wrong join or a dropped null group is as likely to be endorsed as found.

D — Uses plausibility as the test, which is precisely what silently-wrong output is good at passing.

50. The stages, and what each one is really testing — B

A — Anchoring works when you know the band; naming a number first, before you know theirs, more often anchors you below it than above.

C — Sounds agreeable and reliably produces a lower offer, while also failing to answer the administrative question the call exists to settle.

D — Preserves negotiating room in theory, and in practice risks four weeks of interviews for a role that was never going to pay enough.

51. The first call, and the salary question — D

A — Treats a question they are not obliged to answer as a disclosure obligation, and anchors the offer to a market they are deliberately leaving.

B — Solves the anchoring problem with a statement that background checks and payslip requests can contradict, which costs the offer.

C — Protects the anchor but leaves both sides blind to a possible mismatch, which is the one thing this call exists to prevent.

52. The take-home, and how not to lose on it — D

A — Treats an implausibly good score as good news, which is the single most common way candidates fail this exercise.

B — Reaches for overfitting, a real phenomenon that does not explain a score this high on held-out data and leaves the actual cause untouched.

C — A good habit that will faithfully confirm the same inflated score, because the problem is in the columns rather than in the validation scheme.

53. The live exercise, where silence is scored — A

B — Produces a working answer while hiding the reasoning, and the interviewer scores what you showed rather than what you knew.

C — The most common instinct and the most costly, because silent thinking and being lost are indistinguishable from the other side of the screen.

D — Concedes the whole item instead of the one missing detail, and gives away the reasoning that was worth the most points.

54. "How would you build this?" answered well — C

A — Answers the question that was asked but skips the constraints that determine whether any of it is the right shape, which is what is being graded.

B — A sensible practical question that arrives several steps too early, before anyone has established what the system is for.

D — Sounds prudent but defers the thinking rather than doing it, and a pilot with no defined measurement teaches nothing.

55. "How would you measure it?" — C

A — A strong second step that wastes effort if the event was never logged correctly in the first place.

B — Reasonable and specific, but assumes the measurement is sound before that has been established.

D — Useful for gradual movements, but seasonality rarely produces a single-night step change of this size.

56. Talking about your work without lying or shrinking — B

A — The rehearsed non-answer, heard several times a week, and scored as an unwillingness to be examined rather than as a real weakness.

C — Describes a difficult situation rather than a mistake, and ends with other people at fault, which answers a different question than the one asked.

D — Technically responsive and reveals nothing, so the interviewer learns only that you would rather not answer.

57. Your questions, and what the answers reveal — D

A — Reads a hedge as opportunity; it may be true, but nothing in the answer supports it and the phrasing avoids the question asked.

B — Overreads a single answer, and discards jobs where early-stage plumbing is genuinely good experience.

C — A charitable reading that is occasionally right, and which you can test in one follow-up rather than assume.

58. The offer conversation — A

B — Postpones the only moment when your leverage is highest, and raises are typically calculated as a percentage of the number you accepted.

C — Invents leverage that can be checked, is sometimes called, and turns an ordinary negotiation into a question about your honesty.

D — Buries a simple request in justification, which invites the employer to argue with the reasons rather than answer the question.

59. Pay, rates, and who actually decides them — C

A — Compares rates rather than annual totals, ignoring that a contractor is not paid for gaps, illness or holidays.

B — Turns one genuine advantage — protection — into a blanket rule, when well-run contracting can pay considerably more.

D — Uses cost of living to settle a question about income stability; purchasing power does not help when a client stops paying.

60. What a rejection tells you, and what it does not — C

A — Draws a strong conclusion from three trials of a stage where roughly two thirds of good candidates are rejected anyway.

B — Reads a common legal policy against detailed feedback as concealment of a specific fault.

D — Lowers ambition in response to what is statistically an expected run of results, and gives up the level they have demonstrably reached.

61. The headline number is the least useful one — C

A — Treats the numbers as comparable when one is a bundled total and the other is a base, so the comparison has not been made yet.

B — Negotiates against a number that is not a salary, which can end with the candidate arguing themselves into the worse offer.

D — Jumps to the most uncertain part of the package before establishing the guaranteed part that pays rent.

62. Options, RSUs, and why most equity is worth nothing — B

A — Applies the percentage to the headline price, which is exactly how these announcements are read and exactly what the preference stack prevents.

C — Overstates the mechanism; a 1x non-participating preference takes the invested amount, not a multiple of it.

D — Correctly notes that acquisitions often impose retention terms while still valuing the stake against the headline price.

63. Notice, bonds, IP and the clauses that follow you — C

A — Relies on a custom rather than a term, which fails precisely when there is a dispute or the company is acquired.

B — Complies at the cost of the candidate's most valuable public asset, when a carve-out is routinely granted if asked.

D — Assumes a protection that exists in some jurisdictions and not others, and even where it exists it does not cover a project related to the employer's business.

64. Employee, contractor, or an employer of record — B

A — Compares two gross figures that represent different quantities, ignoring that one included contributions, leave and notice and the other does not.

C — Overstates the position: the exposure sits mainly with the company, and for many people cross-border contracting is the only available route.

D — Handles the compliance half while leaving the candidate with a large uncosted reduction in total compensation.

65. Getting paid by a company in another country — B

A — Accepts the advertised claim at face value, which is exactly what an unstated exchange spread is designed to allow.

C — Confuses a conversion cost with a tax obligation; both exist, but they are separate and the tax is not what the bank is taking.

D — Assumes the sending side absorbs everything, when the receiving conversion is where the largest deduction usually happens.

66. The shape of the routes, and why the rules keep moving — B

A — Chooses the most contested channel, where the candidate competes globally and the employer must both want them and hold a sponsor licence.

C — Builds a three-year plan on a random draw the candidate cannot influence, in a country they did not name as the goal.

D — Buys a real route at high cost and converts an immigration problem into a debt problem, when a lower-cost path is already in hand.

67. Two offers, compared with arithmetic — B

A — Discards the differences the arithmetic did surface and substitutes a signal that predicts very little about the next two years.

C — Assumes the cash gap can be widened enough to decide it, when a few per cent will not outweigh benefits and hours either way.

D — Treats avoided insurance spending as a preference when it is a quantified cash difference of a similar size to the salary gap.

68. Raises, promotions and the market rate — B

A — Assumes advocacy failed, when a manager dividing a small pool may have given them the largest share available.

C — May work once, but it spends their standing on a one-off correction that leaves the band — the actual constraint — unchanged.

D — Offers a plausible-sounding alternative mechanism that does not address being at the ceiling of a fixed range.

69. Utilisation, and why a rate is not a salary divided by hours — B

A — Reaches the right direction through a market-positioning argument rather than the cost structure that actually determines the floor.

C — Notes a real perception effect that would not change the outcome if the underlying arithmetic were wrong.

D — Names one real cost while missing the larger structural issue, which is that fewer than two thirds of the hours can be invoiced at all.

70. Pricing the work, and raising the price — C

A — Prices for uncertainty without reducing it, so the number is either uncompetitively high or still wrong in the same direction.

B — Transfers the risk but usually loses the work, because a client asking for a fixed price is asking for a bounded cost.

D — Wins the engagement by making a claim the freelancer knows to be unreliable, which destroys the relationship exactly when the difficult conversation arrives.

71. A pipeline, after the first three clients — B

A — Enters the most price-competitive channel available and competes on cost with people whose floor is lower.

C — Invests in a surface that converts traffic the freelancer does not yet have, and produces nothing in six weeks.

D — Solves a demand problem with a price cut, which sets an anchor that is hard to reverse with the same clients.

72. Writing the thing that prevents the argument — C

A — Sets the precedent that additions are free, which converts each subsequent reasonable request into another unpaid week.

B — Protects the scope at the cost of the relationship, when the client's request is reasonable and could simply be priced.

D — Bills for work the client never agreed to pay for, which is the version of this that ends in a dispute over the final payment.

73. Invoices, procurement, and the chase — B

A — Assumes intent where an approving contact who is confused points at a process failure rather than a decision.

C — Reads a common administrative failure as dishonesty, which leads to an escalation aimed at the wrong person.

D — Worth checking, but does not explain a contact who believes the invoice was approved and expects it to have been paid.

74. Registration, tax, records and insurance — C

A — Reaches for a structure change whose benefit varies by country and which does nothing about money already spent.

B — Increases revenue without changing the behaviour, so a bigger year produces a bigger unpaid bill.

D — Changes the payment schedule, which helps only if the money was set aside — otherwise it produces four crises instead of one.

75. Repeatable offers, retainers and the ceiling of hours — B

A — Treats accumulated templates and process as personal speed rather than as a reusable asset that raised the effective rate.

C — Passes the entire efficiency gain to clients, which is the one move that guarantees the ceiling stays where it was.

D — Reverses the mechanism, since hourly billing is precisely what converts becoming faster into earning less.

76. The conditions under which you should take the job — C

A — Questions their marketability when the stated portfolio is strong; the binding problems are elsewhere.

B — States something often true but framed as a pricing issue, when the same person could raise rates and still be in trouble.

D — Asserts a market rule that plenty of junior freelancers disprove, and misses that the risk is to their development rather than to their bookings.

77. The first ninety days — C

A — Acts on the correct general principle while the specific reason is still unknown, which is how new hires break invoicing for the company's biggest account.

B — Overcorrects into passivity, and the point of the read-only month is investigation, not silence.

D — Produces a critique before understanding, which is the specific move that costs new hires their credibility in the first month.

78. Landing in a codebase nobody documented — A

B — Manages the rollback while still shipping a change whose purpose is unknown, so the failure it prevented can recur before anyone connects the two.

C — Reasonable, but spends someone else's attention on a question the repository can usually answer in twenty seconds.

D — Uses test coverage as evidence of intent, when the least-documented behaviours are exactly the ones added under incident pressure without tests.

79. Being trusted with a number — B

A — Delays a correction people are already acting on, and a footnote a month later is found by nobody who saw the original.

C — Optimises for completeness while a wrong number is actively circulating; the corrected figure should go out now and the full account can follow.

D — Transfers a decision you can make, delays the correction, and puts your manager in the position of concealing something on your behalf.

80. Which parts of this work are most exposed — B

A — Assumes exposure is about accuracy, when queries can be perfectly correct and the job still changes shape.

C — Predicts staffing from the tool alone, ignoring that the work itself was redefined rather than removed.

D — Treats this as a one-off tooling transition rather than a lasting change in what the team is there to do.

81. Keeping up without drowning — D

A — Treats novelty as obsolescence, and reopens a working design three weeks from delivery on the basis of an announcement rather than a measurement.

B — Turns a sensible focus rule into a blanket avoidance, which is how teams end up two generations behind and paying for it.

C — Adds an untested dependency and a second code path to a project three weeks from delivery, in exchange for no measured benefit.

82. The things you will be asked to build — A

B — Correct on the substance and framed as an accusation, which turns a solvable design conversation into a dispute about your manager's motives.

C — Defers the objection until after the system exists and people have been screened by it, when changing it is far more costly and the harm has occurred.

D — Measures a harm rather than preventing it, and a dashboard nobody is accountable for does not stop a rejection being issued.

83. The second job, the plateau, and the ten-year view — C

A — Reads ease as mastery, which is exactly how the plateau disguises itself, and consolidation of unchanging work is repetition.

B — Possibly true and beside the point; the issue is what she is learning, and a higher salary for identical work extends the plateau rather than ending it.

D — Uses her manager's view as the measure, when a manager is generally pleased by exactly the reliability that a plateau produces.

84. The one-pager that gets a decision — C

A — Assumes incomprehension when the issue is attention and ordering; the same readers will read a method section placed in an appendix.

B — Suggests concealing the method, which is what destroys trust when the challenge eventually comes.

D — Fixes the symptom; a one-page document that still buries the recommendation on the last line fails the same way.

85. Getting promoted, and the case somebody has to make — B

A — Assumes a failure of advocacy before establishing whether there is a case to advocate with.

C — May end up being right, but acts before asking the one question — performance, requirements or budget — that distinguishes the three very different causes.

D — Substitutes the smaller mechanism for the larger one, and a merit-pool raise does not address being at a level ceiling.

Module test — 1. The terrain: who hires, for what, and where you fit

1. A

A regional insurer or a staffing firm does not run a developer marketing campaign, so almost nobody thinks of applying there. The competition per opening is a fraction of what a known brand attracts, and the work is often the same. Note that these employers usually do filter on qualifications and do use tracking systems — the advantage is the ratio of applicants to openings, not a friendlier process.

2. C

Bengaluru is UTC+5:30, so 09:00 Pacific is 21:30 local and the window ends at 01:30. That is not a late shift, it is a night shift, and people burn out of it inside two years. The arithmetic is why a South Asian candidate's realistic remote market is Europe and the Gulf far more often than the US west coast — a London team's 10:00 to 14:00 lands comfortably in both Bengaluru and Lagos.

3. D

The tools list is the most honest section of any advert. Whoever wrote it may have been vague about the work, but they were not vague about the stack, because the stack is what the team argues about. Airflow, dbt and a warehouse is data engineering. Apply to it as one, and expect the interview to test pipelines rather than prompts.

4. B

Hiring where a company has no legal entity risks creating a taxable presence and inherits that country's employment law, so companies route people through an employer of record, which charges a few hundred dollars a month per person. That fee is trivial beside a senior salary and material beside a junior one. It is an arithmetic filter, not a judgement about whether juniors can work remotely.

5. C

The cost of a gap is re-entry, not decay of the calendar. Memory of your own reasoning — why you chose that join, what you were halfway through fixing — fades much faster than memory of facts, so the first session back is spent rebuilding context rather than making progress. Two gaps a month and half the year goes on re-entry. Writing one NEXT line at the end of every session is the cheapest defence.

6. A

Levels are set by how much ambiguity you absorb and how far the consequences of your work travel. Only the junior level is about producing a correct result from a defined task; everything above is about deciding what should be produced and getting others to agree. This is why excellent technical people plateau at mid-level, and why management is a fork rather than a step — many companies pay both tracks equivalently for several levels.

Module test — 2. The roles, one at a time

1. D

Idempotency is the property that makes a data platform trustworthy: run it once, run it ten times, delete the output and run it again, and the answer is the same. An appending job that runs twice after a retry silently doubles rows and nothing errors. Rebuilding from source, or replacing a partition rather than adding to a table, removes that entire class of failure.

2. A

Separating retrieval failure from generation failure is the most useful diagnostic in the role. If the right passage was never fetched, no prompt change and no bigger model will fix it — the work is in chunking, embeddings or the query. Teams that skip this split spend months tuning prompts against a retrieval bug, which is why the split is worth stating out loud in an interview.

3. C

Nobody acts on a number they know is missing. Everybody acts on a number that is quietly wrong, and by the time it is discovered it has been in three dashboards and one board deck. That is why every stage gets a row-count check, a freshness check and a null-rate check: the goal is to convert silent wrongness into loud failure.

4. B

Leakage is training data containing information that will not exist when the prediction is actually made — typically a field an operations team fills in after the event you are predicting. Nothing errors and the offline number looks excellent. Asking the availability question of twenty features takes an hour and saves a month, and computing each feature once in code shared by the training and serving paths removes the related skew problem.

5. D

Platform work is measured by other teams' delivery speed, so a platform nobody uses is not neutral — it costs maintenance and splits the organisation in two. People routing around you are reporting something true about how they work. The instinct that survives is to take one team's real deployment, do it with them by hand, and generalise only the parts that repeated.

6. C

Every threshold sets a ratio between two harms, and the answer differs by category. For a direct threat a false negative is irreversible, so you accept a heavy review load. For lawful political speech a false positive silences somebody, so the standard response is reduced distribution rather than removal. What makes somebody good at this work is saying which error is worse for this category, in writing, rather than reaching for one global setting.

Module test — 3. The evidence: what gets tested, and how to prove you have it

1. D

A join multiplies rows whenever the right-hand side has more than one match. Aggregating a left-table column after such a join counts that column once per match — an order with three items contributes its total three times. The query runs, the number looks plausible, and the dashboard is wrong for months. The habit that prevents it is checking row counts before and after every join.

2. A

SQL uses three-valued logic. Comparing a NULL to anything yields unknown, and a WHERE clause keeps only rows that are true, so every NULL-status row disappears with no error and no warning. This silently removes customers, orders and patients from results. Write `is distinct from`, or add `or status is null`, whenever NULL should be kept — and always know which you meant.

3. C

Identifiers look like numbers and are not numbers: nothing arithmetic is ever done to them, and every operation that treats them as numeric loses information. A single empty cell is enough for pandas to infer a float column, after which leading zeros vanish and long account numbers lose their last digits with no warning at all. Passing `dtype={"customer_id": "string"}` is a professional habit, and it is the same bug as a spreadsheet turning a product code into a date.

4. B

The standard error of a proportion is the square root of p(1–p)/n, so error falls with the square root of sample size: four times the data halves the interval. This is why a survey of a hundred people carries a margin near ten points, and why most business tests were never large enough to detect the effect they were run to find. Saying so before the test runs is one of the more valuable things a junior analyst does.

5. C

Plain keyword search over the same documents might score 74, in which case the honest headline is eight points above search rather than 82 per cent. Reporting a score with no baseline is not lying, but it lets the room assume the previous process scored nothing, which it never did. Scoring the dumb version on the same cases takes an afternoon and changes what the work is understood to be worth.

6. D

The bills that surprise people are not compute-hours spent doing work. They are things charged for existing: a small managed database is roughly $15 to $60 a month whether or not anything connects to it, a NAT gateway created silently by a tutorial is about $30, and a GPU instance stopped but not terminated bills all weekend. Two free defences take five minutes — set a budget alert before creating anything, and prefer services that scale to zero.

Module test — 4. Getting seen: the routes that put a person in front of your work

1. B

A funnel that breaks before the first conversation is a statement about route, targeting or the top third of the CV — nobody has got far enough to assess your skill. Improving SQL cannot move a stage that skill is not being measured at. The diagnosis matters because the instinctive response, studying harder, feels productive and moves nothing.

2. C

It is division rather than motivation. Thirty-four conversations at a three per cent cold rate is more than eleven hundred applications; the same target at a thirty per cent warm rate is about a hundred and thirteen approaches. One of those is possible for a person with a job and a life. Doubling cold volume from 200 to 400 buys four extra conversations and costs about a hundred hours.

3. C

A bullet has to survive the question "so what happened?", which means it needs what the situation was, what you did, and what changed, with a number. The other three describe activity, and activity is what everybody claims. Where no number exists, use scale instead — how many people, how many rows, how often it runs. Length is not the signal: this is simply the only option that contains an outcome.

4. A

Follow up once, after seven to ten days, with something added rather than "just bumping this". Silence is usually workload rather than judgement, and a third message converts neutral into negative. The discipline matters because the whole method depends on being a specific person with something to say, and repeated contact turns you into the thing everybody's inbox is defended against.

5. B

Maintainers are short of time, not of ideas, so a clean reproduction cut to the smallest script that shows the behaviour is often more valuable than the fix. Opening with it, and asking whether a pull request would be welcome and how they would prefer it solved, also prevents the common heartbreak of a week's work rejected on design grounds. Refactoring on the way past reads as a stranger rearranging somebody's kitchen.

6. D

All three mechanisms are structural rather than about you. A manager who has watched you work for a year is taking almost no risk compared with an outsider who interviews well; a team that needs somebody and knows one simply asks; and domain context — which table anyone trusts, why the Mexico numbers are late — takes an outsider a year to acquire and cannot be bought. Internal moves are usually still assessed technically, and they are not required to be advertised.

Module test — 5. Using AI in your own search

1. A

No mainstream tracking system scores your CV as a percentage and rejects you below a threshold; that number is invented by whoever is selling you the report. What is real is the knockout question on the form, which rejects without a human, and a recruiter filtering the pile by a keyword — a person choosing a filter, which is why the exact words still matter even though the score does not.

2. C

Where you can check the result and the result is not what you are being judged on — reformatting, gap analysis, practice questions, a first draft — a model saves real hours. Where the output is the evidence of your ability, using one destroys the thing you were producing, and it comes apart at the technical stage. Everything else in this module follows from that single test.

3. B

Titles and dates are what a reference or background check confirms, and a discrepancy is treated as dishonesty rather than as an error — offers are withdrawn for it after acceptance, routinely. Tools get caught earlier, in the first three minutes of a technical screen. The fourth cost people underestimate is that an inflated CV wins interviews for jobs you cannot do, consuming the weeks you needed for the ones you can.

4. A

Thirty divided by one and a half is twenty; thirty divided by four is seven and a half. The side of the ledger that got worse is the one automation makes cheaper, and the side that held is the one that costs attention. Volume genuinely does still work in a few channels — large services firms hiring in batches, graduate schemes, public-sector recruitment, agency contract markets — and outside those it is a way of feeling busy.

5. D

If the data is not yours, no setting on the provider's side makes the disclosure permissible — retention policies and training toggles govern data you were entitled to send. Most trouble in this area comes from skipping straight to the second question. For material that is not yours, a small open-weight model running locally through Ollama or llama.cpp removes the question rather than managing it.

6. B

Reading an answer and finding it sensible is recognition. The tested skill is producing a clear answer aloud in ninety seconds while nervous, and that gap is where almost all interview failure lives. A model given a free hand emits twelve questions you then read, encouragement after each answer trains you for an interview that does not exist, and the follow-up on a vague answer is the part you cannot rehearse alone.

Module test — 6. The interview, from first screen to signed offer

1. A

The column only exists after the event you are predicting, so any model using it scores near-perfectly offline and is worth nothing in production, where the field is empty at prediction time. Interviewers plant traps deliberately — mixed timezones, duplicate rows, leading zeros, negative quantities meaning returns — and you do not have to fix them all. You have to name them, in a README section headed what I found in the data.

2. C

The interviewer cannot see your thinking; they see keystrokes and hear words, and a candidate thinking brilliantly in silence looks identical to one who is completely lost. Naming the boundary of what you know — I need the second value per group, I would use row_number over a partition, I do not recall the frame syntax — demonstrates the reasoning that transfers to the job. Almost every interviewer says yes to the documentation, and a refusal tells you about the job.

3. B

An internal tool for agents who can see the source article is a different system from a public one answering medication questions, and the cost of being wrong sets everything after it: how conservative retrieval is, whether a human is in the loop, whether it may answer at all when unsure. Saying next how you would measure it — fifty real questions, scored, with keyword search scored on the same fifty — moves you above most candidates before you have named a single tool.

4. D

A large share of overnight step-changes are instrumentation: an event stopped firing after a release, a pipeline failed, a source stopped delivering, a dashboard filter changed. A candidate who investigates user behaviour before establishing that the number is trustworthy has revealed how little time they have spent doing this. The other three steps follow — is it everywhere, is it us, is it the world — and the round is scored on the order, not the answer.

5. C

Whoever names a number first sets the frame, and internal recruiters usually have the band on screen and a reason to share it, since a mismatch wastes their time too. In a growing number of jurisdictions they must disclose it. If pressed, give a researched range whose bottom is a number you would genuinely accept — and never anchor on your current salary, which is exactly the trap when moving between segments or countries.

6. A

At final stage a strong candidate converts perhaps a third of the time, so two of three people who get there are rejected and most were entirely capable. The rejection is compatible with an internal candidate applying in week three, a headcount freeze, or a role being withdrawn. That is why the log is reviewed in tens rather than one at a time: ten rows show a pattern or show there is not one, and reading it after each rejection is how a bad week becomes a theory about yourself.

Module test — 7. The money, and the paperwork underneath it

1. C

A CTC figure bundles provident fund, gratuity accrual, an insurance premium and a variable component quoted at target — none of which is guaranteed cash in your account. Eighteen lakh of CTC is frequently about fourteen point four lakh of fixed pay, twenty per cent less than the headline. Guaranteed cash is what pays rent in a bad quarter, so it is kept separate and everything else is listed beside it.

2. B

An option is the right to buy at a fixed price, so it is worth the current value minus the strike, multiplied by the count. Granted at a strike equal to today's fair value, it is worth exactly nothing today and is worth something only if the value rises. People hear ten thousand shares and picture a number; the strike is half the equation and is frequently left out of the conversation.

3. C

Employment typically includes pension or provident fund contributions, paid leave and public holidays, sick pay, health cover, notice, redundancy protection and the employer's half of payroll taxes. Added together those are commonly a quarter to two fifths of salary. A contract at twenty per cent more is roughly break-even before you count unpaid weeks between engagements, which is why the freelancing arithmetic starts at two to two and a half times.

4. C

A bank taking two and a half per cent against the mid-market rate on $3,000 a month costs roughly $900 a year, invisibly, with no line item, while a platform charging 0.6 per cent costs about $216. The rule is to compare the amount that lands rather than the advertised fee — a service with zero fees usually takes it in the rate, and the difference is a laptop every year for a change you make once.

5. C

Some jurisdictions limit blanket assignment by statute and many do not, so whether it would be enforced is not something you can know in advance. What you can know is that the moment of leverage is between the offer and the signature: a company that has spent six weeks choosing you would rather amend a clause than restart. Listing prior projects in an annexe is a routine request, usually granted before signing and never after.

6. C

The three mechanisms are different sizes: an annual raise is a share of a merit pool of perhaps three to eight per cent of payroll, a promotion moves your band and brings eight to fifteen, and a job change resets you to market at fifteen to thirty or more. Salary compression is the mechanism behind it — market rates rise faster than internal pools, so new joiners overtake people who are better at the job, and very few employers fix it retrospectively.

Module test — 8. Freelancing and contracting, as a business

1. B

Utilisation is the missing term. Established solo freelancers bill fifty to seventy per cent of their hours and a first year is often twenty-five to forty, because selling, scoping, invoicing, chasing and learning are unpaid. Add lost employer contributions, own insurance and a buffer for gaps and bad debt, and a sustainable rate is roughly two to two and a half times salary divided by working hours.

2. A

A proposal exists to make the disagreement happen in week zero, cheaply. Naming what a reasonable person might assume — ongoing maintenance, training beyond the walkthrough, migrating historical data, support after handover — is what stops those assumptions surfacing when there is money and pride attached to both readings. The other two sections that do this work are a defined meaning for acceptance and a numbered limit on revisions.

3. B

In a company running procurement software an invoice without a matching purchase order is not rejected; it sits in a queue until somebody notices, while your contact assumes it was paid and you assume you are being ignored. Being set up as a vendor takes two to six weeks and is nobody's priority, which is why you start it on day one and ask directly who handles onboarding and purchase orders.

4. C

Nobody raises it for you, and new enquiries are where a new number costs you nothing to test: if they accept without pausing you were too cheap, and three instant acceptances in a row means raise it again. Existing clients get notice — ten to twenty per cent a year is unremarkable — and in practice fewer leave than people fear. Waiting for praise ties a business decision to somebody else's mood.

5. B

That is the mechanism of a productised service: the same offer, at a fixed price, to the same kind of buyer, delivered by a process that improves with each repetition. The effective rate rose by two thirds without a single negotiation. The sale also gets easier, because a defined thing with a price is a decision a buyer can make in one conversation — and the uncomfortable cost is having to stop advertising work outside the offer.

6. C

Most of what makes somebody better early is a more experienced person looking at their work and saying it is wrong. Clients do not do that — a client can be delighted by an analysis that is quietly incorrect, and you will never find out. Freelancing early is how people acquire five years of habits nobody corrected. The other options are real costs and none of them is permanent in the way an uncorrected habit is.

Module test — 9. Staying: the first ninety days and the ten years after

1. B

Either your number matches, and you have learned the business's definitions, which is most of what a new analyst is missing; or it differs, and you have found your own misunderstanding or a real bug. Bringing the difference as a question — I get 14,120 and the report says 14,200, what am I missing — has started more good careers than any rewrite proposal. Almost every piece of ugliness in a working system was put there deliberately by somebody solving a problem you have not met.

2. C

`git log -S` finds every commit that added or removed a string, and the commit message frequently contains the entire explanation — handle the Belgian VAT case, see ticket 4412. Nine times out of ten the reason is there; the tenth time you ask somebody with the evidence in hand. Removing a condition added at 2am after an incident is how a new joiner causes the same incident again, and it is remembered.

3. B

Clocks move backward in autumn and forward in spring, so a job pinned to a local wall-clock time either runs twice or does not run at all. The symptom is one missing day and one doubled day, in the same two weeks each year, and it is diagnosed hundreds of times a year by people who have never seen it before. Schedule in UTC and convert for display only.

4. C

The three questions are: is it well specified, are there many public examples, and can correctness be checked in seconds. A report query against a documented schema is three yeses. The other three are undocumented context, organisational work and accountability for what good means, all of which are one or more noes. The uncomfortable part is that the exposure lands hardest exactly where people used to enter the field.

5. A

A provider retiring a model version breaks an application that has run untouched for a year, and the schedules are published, so this is the item that bites hardest and gets the least attention. The others are worth sampling quarterly. The test for everything else is whether it would change a decision you are making this quarter — you can learn any tool in a week once it becomes relevant, and you cannot recover six months of skimming.

6. B

The case is argued in a calibration meeting you are not in, and it has to say that this person is already operating at the next level and should be recognised. Evidence of outstanding work at the current level gives the manager nothing to say. The sequence is therefore: find out what the next level means here, do that work visibly for two or three quarters, then ask with the evidence assembled — and check when decisions are actually made, since cycles close months before anything is announced.

Working In AI — final examination

1. A ***1. The terrain: who hires, for what, and where you fit***

Every role in the catalogue is mostly one activity: a number is wrong, a field changed meaning, a source stopped delivering, and somebody has to trace it. Courses skip this because it does not demonstrate well; employers pay for it because it is where the time goes. Recognising that is also what makes a career change between these roles far less frightening than it looks.

2. B ***1. The terrain: who hires, for what, and where you fit***

These platforms are largely flat by design and waiting for a promotion track wastes years. What you take out is real: hours spent staring at bad output beside a definition of what good would have been, plus the working vocabulary — annotation guideline, gold set, inter-annotator agreement, adjudication, edge-case memo. Convert it into an artefact you own by writing your own guideline and measuring your agreement with yourself.

3. D ***1. The terrain: who hires, for what, and where you fit***

A confident four-sentence answer naming the source system, the transformation layer and what happens on a test failure means rung three or higher. Hesitation, or "it depends who you ask", means rung one or two — numbers in spreadsheets and application databases. Both are worth joining and only one of them matches an advert promising model work; the question tells you which job you are actually accepting.

4. A ***1. The terrain: who hires, for what, and where you fit***

For several work visas a recognised degree is written into the regulation, not into a preference — so if your plan involves relocating, check the rule before planning around a portfolio. The three things a degree buys are legal eligibility, passage through an HR qualifications filter, and actual knowledge; only the last of those is about learning, and the middle one is avoidable by targeting employers that do not run such a filter.

5. C ***1. The terrain: who hires, for what, and where you fit***

Employers hire on depth in one thing, and nobody is hired for breadth at zero years. Splitting a year across analytics, data engineering and AI application work feels like keeping options open and produces no evidence in any of them. The workable rule is one lane for six months, then reassess on evidence — did you finish something, did anyone use it, did anyone reply — rather than on mood.

6. D ***1. The terrain: who hires, for what, and where you fit***

Both readings are worth knowing before you apply, and both are about the role rather than about you. The related red flags have their own mechanisms: "build our data infrastructure from scratch" in a junior role means there is no senior person to catch your mistakes, and machine learning expected with no warehouse or pipeline tool named anywhere means year one is plumbing, sold as modelling.

7. B ***1. The terrain: who hires, for what, and where you fit***

In a services firm you are billable inventory: your employer sells your hours and keeps the margin. In a capability centre you work on the parent's own systems with its own roadmap, which is why pay sits above local services firms and below the parent country — that gap is the reason the site exists. Hiring is usually through the same local recruiters, so the bar is local rather than head-office.

8. A ***2. The roles, one at a time***

Churn meaning contract not renewed and churn meaning no activity for thirty days give different answers, and nobody had said which they would act on. Owning definitions is what separates an analyst who is trusted from one who is a query service — the senior move is a single opening sentence asking which definition, before writing anything. A one-page document listing the ten metrics your team argues about does more for your standing than any dashboard.

9. C ***2. The roles, one at a time***

Much of the work is text-to-SQL over a schema and models are genuinely competent at that. What they are not competent at is knowing that the region field changed meaning in March, that finance and sales define revenue differently, or that the person asking will use the answer in a board meeting. That is a different daily practice from writing more queries, and it is the part that is not going anywhere.

10. D ***2. The roles, one at a time***

Analytics engineering is a service role and its only real success metric is whether other people build on your tables. It is entirely possible to spend four months producing an exquisite graph that does not contain the field anybody needed, because nobody was asked. If analysts are not migrating, that is the measurement, and everything else — test coverage, model count, documentation — is decoration.

11. A ***2. The roles, one at a time***

A large share of reported improvements live in the gap between a carefully tuned proposed method and a baseline nobody tuned. This is also why the discipline inside the job is a baseline first, always — implement the heuristic, the linear model or the current production rule and measure it, because a surprising number of research projects end when the baseline turns out to be within a point of the ambitious idea. Discovering that in week one is a success.

12. C ***2. The roles, one at a time***

"Make it smarter" is not actionable, cannot be tested, and hands the hardest decision — what good means — to whoever writes the code, who has less information than the product manager does. A sentence with a number, a set and a consequence lets an engineer build and disagree on the same terms. The other half of the role is the second number engineers rarely raise: cost per interaction, which at four cents used ten times a day by fifty thousand people is a real budget line.

13. B ***2. The roles, one at a time***

A consultancy sells your hours; a forward-deployed engineer works for the product company, where your hours are a cost rather than the revenue. The incentive is therefore to make each deployment need less bespoke code than the last, which means constantly asking which of these requests is the same request wearing different clothes. That feedback loop into the product is also the career value of the seat.

14. C ***2. The roles, one at a time***

Non-determinism and silent provider updates are why this role builds differently: pin model versions, log the full prompt and response for every request so a failure can be reproduced, and keep a regression set that every change must pass before it ships. The same applies to a model used as a judge — when its version changes, your measuring instrument changed, and any comparison across that boundary is unsound.

15. D ***3. The evidence: what gets tested, and how to prove you have it***

A tutorial has already decided everything and its errors are anticipated, so what is left to practise is typing — the part you were never missing. The feeling of understanding everything and building nothing is the accurate result of practising one skill and needing a different one. Rebuilding from an empty file, with documentation allowed and no walkthrough, is unpleasant precisely because it is finally the missing skill.

16. A ***3. The evidence: what gets tested, and how to prove you have it***

Puzzle sites over-represent exotic window functions and under-represent the actual job, which is joining four tables that disagree with each other. Taking one real multi-table dataset, writing twenty questions a person would genuinely ask, and verifying each answer a second way builds the habit an employer is buying. Dialects differ mainly in date and string functions; learning one properly makes the translation an afternoon.

17. B ***3. The evidence: what gets tested, and how to prove you have it***

Notebooks keep state. Cells run in whatever order you clicked them, variables persist after you delete the cell that created them, and the thing that makes your code work may no longer exist anywhere in the file. Half the "I cannot reproduce it" conversations in data teams are this, and the discipline is one habit: restart the kernel and run all before you believe any result or show it to anybody.

18. C ***3. The evidence: what gets tested, and how to prove you have it***

Git stores snapshots and every past snapshot is still there, so deleting the file changes nothing about the exposure. Automated scrapers index new public commits within minutes and several providers scan public code themselves. Rotating first makes the exposed key worthless in a minute; cleaning history first leaves a live key in circulation while you do the harder task. Prevention is `.env` in `.gitignore` on the first commit, with a `.env.example` holding names and no values.

19. B ***3. The evidence: what gets tested, and how to prove you have it***

It is not stubbornness. Whatever replaces a spreadsheet has to preserve the ability to ask a new question without asking you, and a fixed dashboard removes exactly that. The move that works is to leave the spreadsheet as the interface and replace what feeds it — automate the extraction and cleaning, deliver a clean tab, and let them keep their formulas on top. Trust first, migration later.

20. B ***3. The evidence: what gets tested, and how to prove you have it***

Judges have two known biases: they prefer longer answers and text that reads like their own output, and both inflate scores in ways unrelated to correctness. Measuring agreement against your own labels on a shared sample is what tells you whether the instrument works, and a poor agreement rate means fixing the rubric rather than trusting the run. Re-check whenever the judge's model version changes, because that silently changes the instrument.

21. D ***3. The evidence: what gets tested, and how to prove you have it***

The default is that you have no rights, not that you have all of them. The related trap is a dataset marked CCo on an aggregator that is plainly a scrape of a commercial site — the licence field was chosen by an uploader who was not the rights holder, so trace the provenance when it looks thin. A `DATA.md` recording source, licence and download date costs a minute and answers a reviewer's question before they ask it.

22. A ***3. The evidence: what gets tested, and how to prove you have it***

A page with no limitations reads as a page written by somebody who has not checked. The counter-intuitive result is that admitting a limitation raises credibility on everything else, which is why the section belongs near the top rather than buried at the bottom. The two other sections that carry a page are the mess and how you found it — the only part that cannot be faked from a tutorial — and a decision with its rejected alternative named.

23. C ***4. Getting seen: the routes that put a person in front of your work***

Volume genuinely works where the process is built for volume: services and outsourcing firms hiring in cohorts, graduate schemes and campus intakes with fixed windows, public-sector recruitment scored against published criteria, and agency-driven contract markets. Outreach is for start-ups, mid-sized companies, agencies and remote roles. Matching the method to the employer category matters more than the method itself.

24. B ***4. Getting seen: the routes that put a person in front of your work***

That stage tests availability, motivation, band and whether you can describe your work plainly. The usual culprits are a rambling answer to "tell me about yourself" or a salary expectation given badly, and neither is about your skills. Skill is the right diagnosis only when technical stages are where you stop — at which point asking for a reason is worth doing, because about a third of companies will say something specific.

25. C ***4. Getting seen: the routes that put a person in front of your work***

On a phone a recruiter sees the photo, name, headline, location and about two lines before deciding whether to read on, so the headline states what you do, with what, for whom. A student or changer writes what they are doing rather than what they lack — "aspiring" and "seeking opportunities" say only that you are unemployed. The related check nobody mentions: a recruiter with two tabs open compares them, and dates or titles that disagree are a reason to stop reading.

26. D ***4. Getting seen: the routes that put a person in front of your work***

Publishing works because a page written in March is found in September by somebody searching the exact error text, which rewards narrow, specific answers over broad guides. Once you have solved it, it feels obvious — it was not obvious to you on Monday, which is exactly why almost nobody writes it. Title it as the question somebody would type, answer in the first paragraph, and include the error text, the command and the version numbers.

27. A ***4. Getting seen: the routes that put a person in front of your work***

The failure mode is real: you do two jobs on one title and one salary, and some organisations let that run for years. Two guards work. An email after a conversation saying "as discussed, we will look at this again in six months" is enough to make it a decision rather than a drift, and keeping your own dated record of everything you built means it becomes your Selected work section whether you move internally or leave.

28. C ***4. Getting seen: the routes that put a person in front of your work***

The commonest way a first project fails is that the client never sends the data, three weeks pass, and you are somehow the one who is late. Putting their obligation in writing with a date makes that conversation easy rather than accusatory. Revisions and payment terms matter too — take thirty to fifty per cent before starting, and a client who refuses has told you something useful, cheaply.

29. D ***4. Getting seen: the routes that put a person in front of your work***

In a community the order reverses: the message lands only after you have been visible. Answering questions for three months produces fifteen to thirty people who have watched you be useful and will forward you a job — which is what a network actually is, stripped of its unpleasant connotations. You are qualified to help anybody one step behind you, which is a far larger group than most people assume.

30. B ***5. Using AI in your own search***

What you see pasted is roughly what the parser sees. If your job titles are interleaved with your skills or your contact details have vanished, the layout is the problem — usually two columns, text in a header or footer, a table used for layout, or a design-tool export with no text layer. This test is worth more than any paid report, and the whole formatting question is an hour's work once rather than an ongoing project.

31. A ***5. Using AI in your own search***

The top third is the only part most readers reach, so it earns the per-application rewrite while the rest stays fixed. The mechanism of the whole exercise is translation: two companies describe the same job as "ETL" and "data pipelines", and a reader scanning for their own words concludes you have not done the thing you have done for three years. Do not let a model write the final text — it knows nothing about what you did, so anything it adds is recycled or invented.

32. A ***5. Using AI in your own search***

These services match on keywords, so they apply to roles you plainly do not fit, and your history is stored permanently in each company's tracking system. Some systems also detect and block them through submission-rate patterns. The largest cost is the least visible: two hundred applications feel like work, and the five people you did not contact are the search you did not run.

33. D ***5. Using AI in your own search***

Twelve engineering roles and no data roles tells you where you would sit. Three adverts naming Snowflake and one naming Databricks tells you a migration is happening. Hiring a Head of Data alongside your role means your manager may not exist yet. Reviews are worth five minutes read only for repeated specifics — four people naming the same manager is signal, twenty people saying "great culture" is not.

34. B ***5. Using AI in your own search***

Most systems transcribe your answers and score the text against a rubric of competencies, so if the transcript is wrong the score is wrong. Wired earphones with a microphone, a quiet room, announcing the structure — "three things happened, first…" — and answering in the first sentence do more than anything visual. Automated facial analysis was heavily marketed around 2018 to 2020 and has substantially retreated, and the stage's purpose is elimination rather than selection.

35. C ***5. Using AI in your own search***

Generated data code fails by running, producing plausible output and being wrong. `df.groupby('region')` drops rows where region is null because dropna defaults to true; a one-to-many join written as one-to-one inflates every sum; a deprecated argument silently changes behaviour. The defence is not vigilance, which fails, but twenty lines of assertions: row counts after joins, totals reconciled against ten rows computed by hand, null counts printed before and after.

36. B ***5. Using AI in your own search***

The test is whether the sentence survives being said by any other candidate. Default model prose is balanced, enthusiastic, superlative and structurally identical every time, which is exactly the register a recruiter reading forty applications has learned to skip. If you draft with a model, do the last pass yourself and cut everything that is true of anybody — the specific, checkable sentence is the only kind worth the paper.

37. D ***6. The interview, from first screen to signed offer***

Panels write down their assessment after each round and those notes travel forward, so the interviewer in round four has usually read what round two thought before meeting you. The weak answer is already recorded; revisiting it reads as strength, because you are showing that you go back over your own work. The same fact means early rounds deserve harder preparation than instinct suggests — they are cheap for the company and decisive for you.

38. D ***6. The interview, from first screen to signed offer***

That is not an assessment, it is unpaid work. The other red lines are more than about eight hours, "build a prototype of our product", and a take-home given before anybody at the company has spoken to you. Decline politely and offer an alternative — a ninety-minute live session, or a walkthrough of something you have already built. A company that refuses has told you how it will treat your time later, for free.

39. A ***6. The interview, from first screen to signed offer***

"I would not fine-tune on the articles: they change weekly, fine-tuning bakes them in, and fixing a wrong answer would mean retraining — with retrieval I fix it by editing the article." That is reasoning rather than knowledge, which is why it works regardless of experience. What loses is naming five products in the first thirty seconds, and never mentioning evaluation at all, which for AI roles is close to disqualifying.

40. B ***6. The interview, from first screen to signed offer***

A good metric is sensitive, hard to game and attributable, and revenue fails the third: you cannot tell whether your change caused it. The pattern worth practising out loud is naming a counter-metric — the way the number could look good while the product got worse, such as time-in-app rising for a support tool because people cannot find anything. Stating a failure condition that would make you kill the feature is remembered, because most candidates do not.

41. A ***6. The interview, from first screen to signed offer***

If there are network effects between users, randomising by user leaks the treatment into the control and no amount of care afterwards recovers the estimate — which is why some tests randomise by city. The others are genuinely required and are recoverable by rerunning: choose the primary metric and guardrails in advance, run at least a weekly cycle because Tuesday users are not Saturday users, and agree a stopping rule so nobody peeks and stops at significance.

42. C ***6. The interview, from first screen to signed offer***

They are hiring somebody who will one day break production, and what they are buying is your behaviour in the hour afterwards. The strong version names a real error, quantifies it, admits how it was found, describes the systemic fix and states the habit that came from it — all in under ninety seconds. A story that ends with a colleague or a supplier being wrong is not a failure story, and the caring-too-much answer is scored as an unwillingness to be examined.

43. B ***6. The interview, from first screen to signed offer***

"We are exploring several initiatives" means nothing is in production. "The fraud scoring model, retrained monthly, last deployed in July" means there is a working system to learn from. The companion questions are where the data lives and who owns it, how the team knows a change made things better, what shipped last quarter and how long it took, and why the role is open. Hesitation matters more than content.

44. B ***6. The interview, from first screen to signed offer***

The silence after the question is the entire technique — most people fill it and negotiate against themselves. A realistic outcome without a second offer is five to ten per cent or a non-cash concession, and five per cent on a first salary compounds through every subsequent raise and every future conversation anchored on what you currently earn. Inventing a competing process you do not have is checkable and ends badly.

45. D ***7. The money, and the paperwork underneath it***

Investors typically get their money back before common shareholders get anything, so a headline sale price and an employee's share can be very different numbers. The press release says sixty million dollars. This is why the question to ask is what the total liquidation preference is and whether any of it is participating — and why a company that declines to answer has told you the grant cannot be valued, which means entering it at zero.

46. D ***7. The money, and the paperwork underneath it***

Exercising ten thousand options at a ₹250 strike costs ₹25 lakh in cash, and in several countries tax falls at exercise on the paper gain, for shares that cannot be sold. The pattern to notice is that tax is often due before any money arrives. Some companies extend the window to seven or ten years; asking is normal, and the answer tells you a great deal about how the company thinks about the people who leave.

47. A ***7. The money, and the paperwork underneath it***

The employer of record legally employs you in your country and bills the client, so your statutory protections come from where you are, not from where the client is. Also check that the provider's fee is paid by the client rather than deducted from your salary, and ask to see the local employment contract rather than only the offer letter. When a foreign company has no entity, this is usually the best available structure.

48. C ***7. The money, and the paperwork underneath it***

The tests are broadly similar across jurisdictions: who controls how and when the work is done, whether you can send a substitute, whose equipment you use, whether you work for others, and how integrated you are into their organisation. Consequences fall mostly on the company — back taxes, penalties, reclassification — which is why large employers are cautious and why employer-of-record providers exist at all. Unpaid tax and contributions can still land on you, years later.

49. B ***7. The money, and the paperwork underneath it***

Tax residence usually turns on days present — the 183-day figure is a common threshold rather than a universal rule — with additional tests about home, family and centre of economic interest. Most country pairs have a double taxation treaty, so the same income is not taxed twice, but you must file correctly to prove it. Read your own revenue authority's guidance, which is free, and pay an accountant for one session once foreign income matters.

50. D ***7. The money, and the paperwork underneath it***

There is no lottery, the employer already knows you, and the process is routine — a large share of people working abroad arrived this way and almost nobody plans for it deliberately. Points-based systems also reward preparation, because a higher language test score is worth real points. The study route converts a visa problem into a debt problem, and no legitimate party ever charges a candidate for a job offer.

51. A ***7. The money, and the paperwork underneath it***

If comparable family cover would cost ₹45,000 a year, the offer providing it is ₹45,000 better in money you do not have to spend. For somebody with dependants or a parent to insure, this line is frequently larger than the salary difference and is the one people leave out. The order that keeps the comparison honest is guaranteed cash, then expected cash at historic payout rates, then contributions, then avoided spending, then the costs of the job — with private equity entered at zero.

52. C ***8. Freelancing and contracting, as a business***

Profit and cash flow are different problems. A three-month project paid entirely on completion is you lending the client money for three months, and a large company with a procurement department may take six to ten weeks after that. The three consequences are a reserve of three to six months of expenses before starting, invoicing on milestones rather than at the end, and planning for the first six months to earn less than the salary did.

53. D ***8. Freelancing and contracting, as a business***

When you are busy delivering you are not selling, so three months later the work you did not sell fails to arrive. Half a day every week — two outreach messages, one reconnection with a past client, one piece of public work, and updating the pipeline list — is boring and it removes the oscillation. Follow-up is where most freelance work is actually won, and almost nobody sends the polite second message.

54. B ***8. Freelancing and contracting, as a business***

A sentence that names a buyer, a problem and an outcome can be forwarded without thinking. The others are unmemorable, so nobody hears them and pictures a specific person with a specific problem. It feels narrower than is comfortable, which is exactly why it works — and it is a door rather than a fence, since you can still take other work. The highest-converting sources are past colleagues, client referrals asked for specifically, and subcontracting from agencies.

55. C ***8. Freelancing and contracting, as a business***

Scope creep never arrives as a demand; it arrives as "while you are in there, could you also", and three reasonable requests is a week. Agreeing, pricing it and moving the date in a single sentence avoids the confrontation entirely. Clients respect this; what they resent is a freelancer who silently absorbs three changes and then delivers late. Log the ones you waive too, and show them at full price with the discount stated.

56. D ***8. Freelancing and contracting, as a business***

A client who pays the first milestone late will not improve, and the correct response is to pause and fix the terms while the amount at risk is still small. The other warning signs are visible before you start: no written agreement, refusal of any deposit, and urgency combined with vagueness about the outcome. Budget one to three per cent of revenue as bad debt so that when it happens it is a cost of business rather than a personal failure.

57. A ***8. Freelancing and contracting, as a business***

The disaster is not a failed business; it is a good year followed by a tax bill for money spent in good faith on the assumption that everything received was income. Twenty-five to thirty-five per cent, moved the day the payment lands, into an account you treat as not yours. The other early questions are whether instalments are required during the year, at what turnover you must register for sales tax, and how exported services are treated.

58. B ***8. Freelancing and contracting, as a business***

A short paid discovery of two to five days protects you from quoting blind, is genuinely useful to the client even if they stop there, and converts at a high rate because they have already paid you once. Estimation error in this work is consistently in one direction, so honest contingency is twenty per cent for familiar work and thirty to fifty for an unfamiliar system. And never quote in the meeting: "I will send you a number tomorrow" costs nothing.

59. C ***9. Staying: the first ninety days and the ten years after***

The fear is looking stupid; the actual failure is guessing quietly for three days and producing something wrong. Saying what you already tried makes the answer quick to give, and asking in the open means the answer is searchable by the next person and your colleague answers once rather than four times. A new person asking good questions publicly reads as engaged, not as struggling.

60. B ***9. Staying: the first ninety days and the ten years after***

The recovery is mechanical: say so immediately, in the same place the wrong number was given; give the corrected figure; one sentence on the cause; one sentence on what stops it recurring. Not a paragraph of apology. Fast correction is what makes people comfortable relying on you, because it means they will hear about problems from you first — and one wrong number costs more trust than ten correct ones build.

61. D ***9. Staying: the first ninety days and the ten years after***

Organisations do not assign accountability to a tool. That is why evaluation work is more durable than prompt work — deciding what good enough means is a responsibility rather than a task. The other durable categories are undocumented context, work that touches the real organisation, and systems work under constraints. The uncomfortable part of the same analysis is that exposure lands hardest on the traditional bottom rung.

62. A ***9. Staying: the first ninety days and the ten years after***

A framework's API surface, the current best model and prompt tricks tied to one version decay fast. SQL, statistics, debugging, writing clearly, how data actually gets messy and your domain barely decay at all. Nearly everyone spends their learning time on the fast half because that is what gets announced, and deliberately spending most of yours on the slow half is the whole strategy.

63. D ***9. Staying: the first ninety days and the ten years after***

Half the time the request changes at the clarifying question, because nobody had thought about it and it is easier to answer with a change than with a defence. Being the person with an alternative is far more effective than being the person with an objection. The written note to your manager and the escalation both follow — three sentences naming the specific risk, not a manifesto — and a specific harm moves a room where moralising does not.

64. B ***9. Staying: the first ninety days and the ten years after***

If yes, you have been paid to repeat yourself. The two companion questions are whether there is anybody here you are learning from, and whether anything you built was used by more people this year than last. Three noes is not a crisis; it is a year's notice. The stall is dangerous precisely because it feels like competence — work is easy, you are respected, and from the inside it is indistinguishable from having arrived.

65. A ***9. Staying: the first ninety days and the ten years after***

Senior readers ask the same three questions — how do we know, what does it cost, and what happens if we do nothing — and the third is the one writers omit. The document leads with the recommendation, its cost and worth, and the decision needed from whom by when, because the reader may stop at any point and should have the answer before they do. Stating uncertainty raises credibility rather than lowering it, and rejected alternatives are what separate a recommendation from an opinion.