

TRACK 5

AI for Data Analysis

Generative AI for analysts: prompting, AI vs GenAI vs agentic AI, AI over SQL (Metabase, MinusX), Copilot and Gemini in Excel, and office agents.

4 lessons

Version 2026.10

Updated October 2026

Created by **Varun Kumar**
www.skillnavora.in

Contents

Every entry is a clickable link. The PDF also has bookmarks for each lesson and section.

- [01](#) Generative AI for Analysts: How It Works, Prompting & AI vs GenAI vs Agentic AI
- [02](#) AI with SQL: Text-to-SQL, Docker, Metabase & MinusX
- [03](#) AI in Excel & Office: Copilot, Gemini, Claude, Python in Excel & Agents
- [04](#) The AI-Assisted Analyst Workflow, Agents & Governance (Capstone)

Generative AI for Analysts: How It Works, Prompting & AI vs GenAI vs Agentic AI

Understand what generative AI is and how large language models behave (tokens, context, temperature, hallucination), how AI, machine learning, deep learning, GenAI and agentic AI differ, and how to write prompts that give analysts accurate, checkable results, with reusable templates and privacy rules.

● DVA with AI · class 1

✦ Models and tools as of 2026

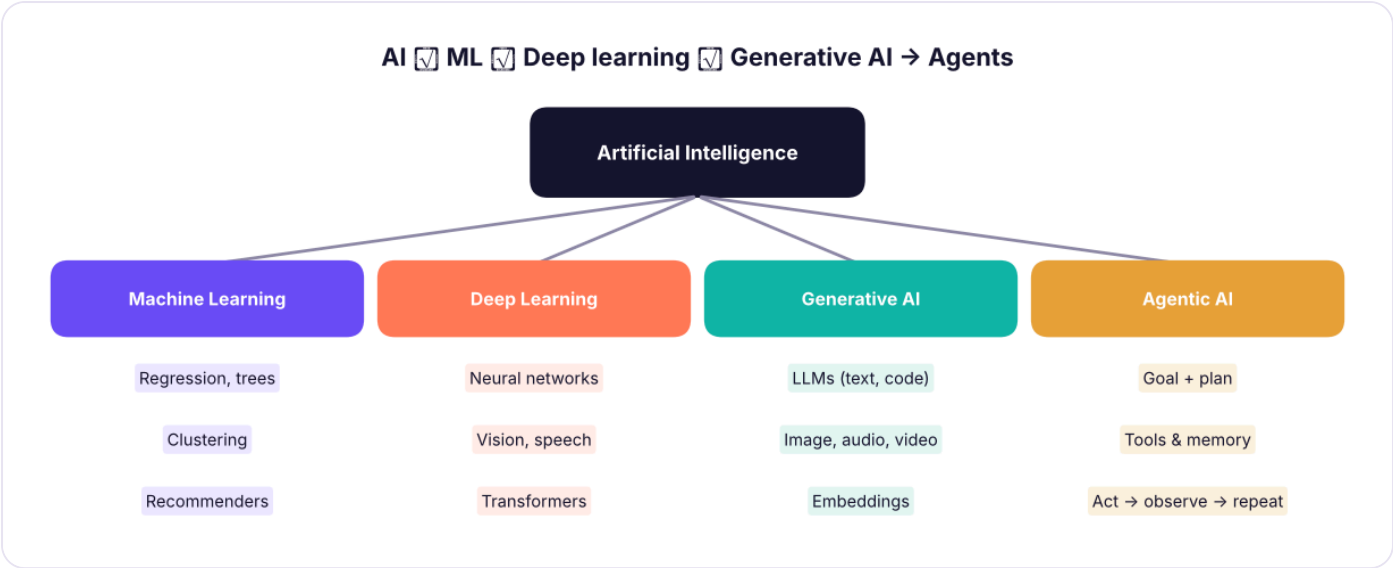
Beginner

🕒 110 min

By the end of this lesson you can

- Explain the relationship between AI, ML, deep learning, generative AI and agentic AI
- Describe in plain language how an LLM generates text and why it can be confidently wrong (hallucination)
- Write prompts using role, context, task, format and constraints, with examples and step-by-step reasoning
- Use prompt templates for common analyst tasks and know what never to paste into an AI tool
- Choose a suitable tool and check an AI answer before using it

The AI family tree



How the terms relate. Each ring is a subset of the one before it. Agentic AI uses generative models as its brain but adds goals, tools and loops.

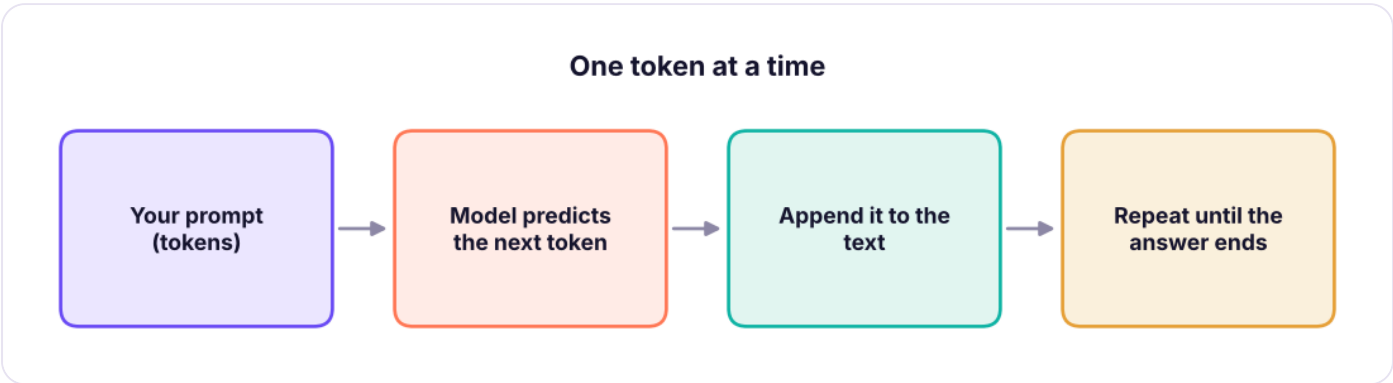
Term	What it means	Example
AI	Machines doing tasks that normally need human intelligence	Rule - based chess engine, spam filter
Machine learning (ML)	Systems that learn patterns from data instead of following hand - written rules	Predict loan default from past loans
Deep learning	ML with many - layered neural networks	Image recognition, speech - to - text
Generative AI (GenAI)	Models that create new content (text, images, code, audio)	ChatGPT, Claude, Gemini, Copilot, Midjourney
LLM	Large language model: the GenAI engine for text and code	GPT, Claude, Gemini, Llama
Agentic AI	An LLM plus goals, tools and a loop : it plans, calls tools (run SQL, search, edit files), checks results and continues until the goal is met	An analyst agent that explores a database and writes the report
AGI	Hypothetical AI at human level across tasks	Not here yet; ignore the hype

AI vs GenAI vs agentic AI

	Traditional ML	Generative AI	Agentic AI
Output	A prediction (number, label)	New content (text, code, image)	A completed multi-step task
Input	Structured data	A prompt (text, files, images)	A goal in plain language
Control	You train and deploy a model	You ask once, get one answer	The system decides the next step
Example	"Will this customer churn?" → 0.82	"Summarise these reviews" → paragraph	"Find why sales fell in Q3 and draft a memo" → runs queries, charts, writes
Main risk	Bias, drift	Hallucination	Hallucination plus actions : wrong tool calls, runaway cost, security

How an LLM works (without maths)

An LLM is trained on vast amounts of text to predict **the next token** (a word or word piece). Repeating that prediction one token at a time produces sentences, code and tables. It has learnt statistical patterns of language, facts and reasoning styles, but it does **not look things up** unless it is given a tool or documents.



Generating an answer: the prompt is split into tokens; the model repeatedly predicts the next token from everything so far.

Concept	Meaning	Why analysts care
Token	Roughly ¾ of an English word (about 4 characters)	Limits and costs are measured in tokens
Context window	How much text the model can consider at once (prompt + answer)	A long file may not fit; the model forgets what is outside
Temperature	Randomness of word choice	Low (0–0.3) for SQL, maths and extraction; higher for brainstorming
System prompt	Hidden instructions that set behaviour	Custom GPTs/Projects store your standing instructions
Training cut-off	The model's knowledge stops at some date	It may not know recent tools or prices; give it documents
Grounding / RAG	Supplying real documents or data in the prompt (Retrieval-Augmented Generation)	Cuts hallucination: "answer only from this file"
Tool use / function calling	The model asks the app to run a calculator, database query or code	Makes answers computed instead of guessed
Multimodal	Accepts images, PDFs, audio as well as text	Paste a chart screenshot and ask what it shows

WATCH OUT Hallucination

Because an LLM generates *plausible* text, it can state false things with complete confidence: invented statistics, non-existent functions, wrong column names, fake citations. **Never use an AI answer as a fact until you have checked it:** run the code, recompute the number, open the source.

Prompt engineering for analysts

A prompt is an instruction plus the material the model needs. The most reliable structure has five parts:

The R-C-T-F-C prompt

1. Role

Who should it act as? 'You are a senior retail data analyst.'

2. Context

Business, data and definitions: columns, grain, time period, what 'sales' means.

3. Task

One clear action: 'Find the three biggest drivers of the margin drop and quantify each.'

4. Format

Table, bullet list, SQL only, JSON; length; audience.

5. Constraints

'Use only the data given. If unsure, say so. Show assumptions. Do not invent numbers.'

+ Examples

One or two samples of the exact output you want (few-shot).

A prompt recipe: Role, Context, Task, Format, Constraints.

Weak vs strong prompt

Weak: "Analyse my sales data."

Strong:

Text

```
Role: You are a senior retail data analyst for an Indian omnichannel retailer.
Context: Table superstore(order_id, order_date, region, category, sub_category, sales,
discount, profit),
        one row per order line, Jan 2023-Dec 2025, amounts in rupees. Margin = profit /
sales.
Task: Explain why overall margin is only about 9%. Look at category, sub-category and
discount band.
Format: (1) a 3-bullet summary for the CF0, (2) a table of the 5 worst sub-categories
(sales, profit, margin),
        (3) the SQL Server queries you used.
Constraints: Use only the columns listed. State any assumption. If a number cannot be
computed from the table, say so.
```

The strong prompt names the grain, the metric definition, the output format and the rules for uncertainty. The result is far more likely to be right *and* checkable.

Core techniques

Technique	How	Use for
Be specific	Name columns, periods, units, audience	Everything
Few - shot examples	Show 2–3 input→output pairs	Classification, extraction, tone matching
Step by step	"First list the assumptions, then compute, then summarise"	Multi - step analysis; makes errors visible
Ask for checks	"Then list three ways this answer could be wrong"	Self - critique
Structured output	"Return JSON with keys..."	Feeding results into code
Delimiters	Put data in <code>"""triple quotes"""</code> or XML tags	Stops the model confusing instructions with data
Iterate	Treat the first answer as a draft; refine with follow - ups	Always
Give it the schema, not the data	Column names and 5 sample rows, not the whole file	Privacy, token limits

Prompt templates for common analyst jobs

Job	Template (replace brackets)
Explain a formula	"Explain this Excel formula step by step for a beginner and give a simpler alternative: [formula]"
Write SQL	"Dialect: SQL Server. Tables: [schema with keys and grain]. Write a query that [question]. Return only SQL, then list assumptions."
Debug code/SQL	"This [language] query gives [wrong result / this error]. Expected: [x]. Query: [code]. Find the cause and show the fix."
Clean data	"Here are 10 sample values of column [x]: [list]. Propose normalisation rules and a mapping table; flag ambiguous cases."
Interpret results	"Here is a result table: [table]. Write 3 insights for a non-technical manager, each with a number, and 2 caveats."
Choose a chart	"I want to show [message] to [audience] with [data]. Recommend a chart and what to label."
Draft a report	"Turn these findings into a one-page memo with: situation, key findings, recommendation, risks. Keep it under 250 words."
Interview practice	"Act as an interviewer for a data analyst role; ask one SQL question at a time and give feedback on my answer."

EXAMPLE

From vague to checkable

Prompt: "Explain this formula: `=IFERROR(INDEX($C:$C,MATCH(1,($A:$A=F2)*($B:$B=G2),0)), "Not found")`" A good answer says it looks up C where A equals F2 **and** B equals G2 (a two-condition lookup), returns "Not found" if none, and notes it needs array evaluation in older Excel. You can verify by testing it on 3 rows.



Choosing and using AI tools

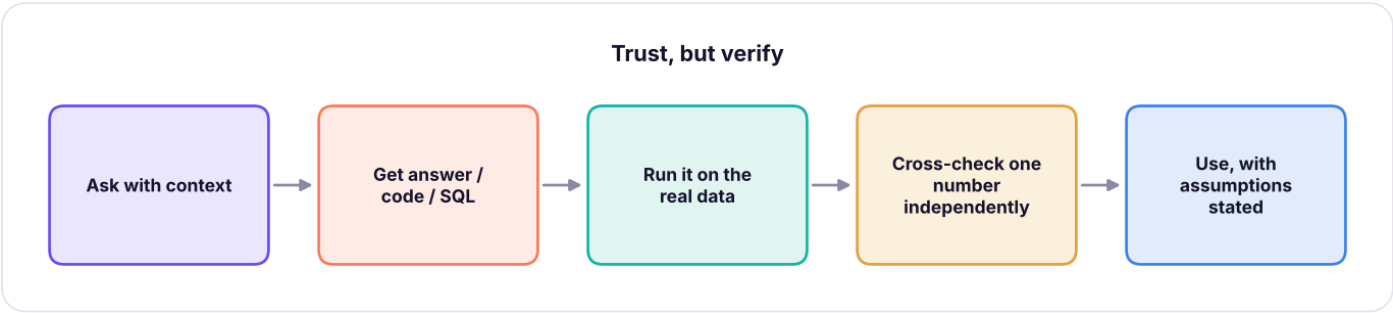
Tool type	Examples	Good for	Watch out
General chat assistants	Claude, ChatGPT, Gemini, Copilot, Grok, Perplexity	Explaining, drafting, code, brainstorming	Data privacy; hallucination
Search-grounded	Perplexity, Gemini with search, Copilot with web	Fresh facts with citations	Check the cited pages; sources vary in quality
Document/notebook tools	NotebookLM, Projects, Custom GPTs	Answering only from your files	Quality depends on the files
Embedded in apps	Copilot in Excel/Power BI, Gemini in Sheets, Metabase/MinusX, Claude in Excel	Work where the data is	Permissions; see next lessons
Coding assistants and agents	Claude Code, Cursor, GitHub Copilot, Codex	Writing and running code, analysis scripts	Review diffs; run tests; secrets

Never paste

- 1. **Personal data** (names with phone numbers, emails, Aadhaar/PAN, health, salary of identifiable people) unless your company's approved tool and policy allow it.
- 2. **Confidential business data** (unreleased financials, customer lists, contracts, source code, credentials, API keys) into public tools.
- 3. **Anything you are not allowed to share**: client data under NDA.

Use **enterprise versions** with data-protection terms, **mask or sample** the data (replace names with IDs, share the schema and 5 fake rows), and follow your organisation's AI policy. In India, personal data is also covered by the Digital Personal Data Protection Act, 2023.

A verification habit for every AI answer



The analyst's rule: generate with AI, verify with data.

Here is why verification matters. Suppose an AI writes the following query for "average order value per region". It looks right, runs without error and returns plausible numbers, but it averages order **lines**, not **orders**:

Python

```
import duckdb, pandas as pd
con = duckdb.connect()
con.execute("CREATE TABLE superstore AS SELECT * FROM
read_csv_auto('superstore_sales.csv')")
wrong = con.execute("SELECT region, ROUND(AVG(sales), 0) AS avg_order_value FROM superstore
GROUP BY region ORDER BY region").df()
right = con.execute("""SELECT region, ROUND(AVG(order_total), 0) AS avg_order_value
FROM (SELECT region, order_id, SUM(sales) AS order_total FROM
superstore GROUP BY region, order_id) GROUP BY region ORDER BY region""").df()
pd.DataFrame({"region": wrong.region, "AI query (per line)": wrong.avg_order_value,
"Correct (per order)": right.avg_order_value})
```

OUTPUT

	region	AI query (per line)	Correct (per order)
0	East	30,521	60,684
1	North	32,038	61,892
2	South	32,580	64,925
3	West	32,192	64,067

4 rows × 3 columns

The AI's version understates order value by roughly half, because an average order has about two lines. A one-line sanity check (total sales ÷ number of orders) would have caught it:

Python

```
tot = con.execute("SELECT SUM(sales) / COUNT(DISTINCT order_id) FROM
superstore").fetchone()[0]
print(f"Total sales / distinct orders = {tot:,.0f}")
```

OUTPUT

Total sales / distinct orders = 62,942
--

COMMON MISTAKES

Common AI-assistant failure modes

1. **Wrong grain:** counting lines as orders, double-counting after a join (Lesson 8 in SQL).
2. **Invented columns or functions:** the code references a field that does not exist.
3. **Silent assumptions:** it picks a date range, currency or definition without saying.
4. **Plausible but unverified numbers:** especially when asked to "estimate" or "recall" data it cannot see.
5. **Over-agreeing:** if you assert something wrong, it may follow you. Ask it to challenge your assumption.
6. **Stale knowledge:** new tool versions, prices, laws.

Responsible use

- **Accountability stays with you:** you sign off on the analysis, not the model.
- **Bias:** models mirror training data; check outputs for unfair assumptions about people or regions.
- **Transparency:** tell stakeholders when AI helped, and keep the prompt and verification notes.
- **Copyright and attribution:** do not present generated text or images as someone else's work; check licences.
- **Security:** treat prompts and uploaded files as data leaving your machine; beware prompt injection when an AI reads untrusted documents or web pages.

DO IT WITH AI

Do it with AI

Pick a task you did earlier in this course (for example the SQL discount analysis). Ask an AI assistant to produce the answer from the schema, run its SQL, and compare with the result in this handbook. Where it differs, work out *why* (grain? date range? definition?). This single exercise teaches more about AI limits than any lecture.

Summary

KEY TAKEAWAYS

- AI $\supset$ ML $\supset$ deep learning $\supset$ generative AI; **agentic AI** = an LLM with goals, tools and a loop.
- LLMs predict the next token; they do not look things up unless given documents or tools. They can **hallucinate**, so verify every answer.
- Good prompts have **role, context, task, format and constraints**, use examples and step-by-step reasoning, and give the schema (not the sensitive data).
- Use templates for SQL, formulas, cleaning, insights, memos and interview practice; iterate.
- Never paste personal or confidential data into public tools; mask or sample, and follow your company policy.
- The analyst's loop: ask with context $\rightarrow$ run on real data $\rightarrow$ cross-check one number $\rightarrow$ state assumptions.

Quick quiz

6 questions · tap an answer to check it

1. Which statement best describes agentic AI?

- A** A model that only classifies images
- B** An LLM-based system that plans, uses tools and repeats steps to achieve a goal
- C** Any chatbot
- D** A rule-based macro

Answer: B. Agents add goals, tools and a decision loop on top of a language model.

2. Why can an LLM state wrong facts confidently?

- A It lacks memory
- B It generates plausible text from patterns and does not verify facts unless given sources or tools**
- C It is connected to the internet
- D It always copies its training data

Answer: B. Fluency is not accuracy.

3. Which setting is best for generating SQL?

- A High temperature
- B Low temperature**
- C No context
- D Maximum length

Answer: B. Low temperature reduces random variation for precise tasks.

4. What makes a prompt more reliable for analysis?

- A Shorter wording
- B Role, context (schema and definitions), a clear task, an output format and constraints**
- C Asking for "everything"
- D Hiding the table names

Answer: B. Specifics reduce guesswork and make the answer checkable.

5. A teammate wants to paste a customer list with phone numbers into a public chatbot. Best advice?

A Go ahead

B Do not; mask or sample the data, or use an approved enterprise tool

C Delete the email column only

D Convert it to PDF first

Answer: B. Personal data should not go into tools not approved for it.

6. An AI query returns an "average order value" half the size of total sales ÷ orders. The likely error is:

A Wrong chart

B Averaging order lines instead of order totals (wrong grain)

C Too many decimals

D A missing index

Answer: B. Always sanity-check results against a simple independent calculation.

EXERCISE Prompt rewrite

Rewrite this weak prompt into a strong one and list three checks you will run on the answer: *"Tell me about churn in my telecom data."* The data is `telecom_churn.csv` with columns `customer_id`, `senior_citizen`, `tenure_months`, `contract`, `internet_service`, `payment_method`, `monthly_charges`, `total_charges`, `support_calls`, `churn` (Yes/No).

Show solution (worked answer)

Strong prompt: "Role: senior telecom data analyst. Context: table `telecom_churn(...columns...)`, one row per subscriber, `churn` = Yes/No. Task: find which contract types, tenure bands and support-call counts have the highest churn rate and quantify each. Format: a table (group, customers, churners, churn %), three bullets for a retention manager, and the SQL/pandas code used. Constraints: use only these columns, state assumptions, do not infer causes you cannot test, flag small groups (< 50 customers)." Checks: (1) recompute the overall churn rate by hand (count of Yes ÷ rows) and match it; (2) confirm each group's customer counts add up to the total; (3) spot-check one group with an independent query.

INTERVIEW CORNER Questions you may be asked

Q: Difference between AI, ML and generative AI? AI is the broad field; ML learns patterns from data to predict or classify; generative AI (a deep-learning subset) creates new content such as text, code and images.

Q: What is a hallucination and how do you reduce it? Confident but false output. Reduce it by grounding answers in provided documents or data (RAG), using tools that compute results, lowering temperature, asking for sources and assumptions, and verifying independently.

Q: How do you write a good prompt for a data task? State the role, context (schema, grain, definitions), a single clear task, the required output format, and constraints (use only given data, state assumptions); give examples; iterate.

Q: What is agentic AI? Systems where an LLM plans and executes multi-step tasks using tools, observes results and continues until a goal is met, with human oversight for risky actions.

AI with SQL: Text-to-SQL, Docker, Metabase & MinusX

Use AI to write, explain and debug SQL safely. Learn how text-to-SQL works and where it fails, give the model the right schema, protect your database with read-only access, install Docker, run Metabase on your own data, connect an AI assistant such as MinusX, and validate every generated query.

● DVA with AI · class 2

✦ Docker, Metabase and AI assistants, 2026

Intermediate

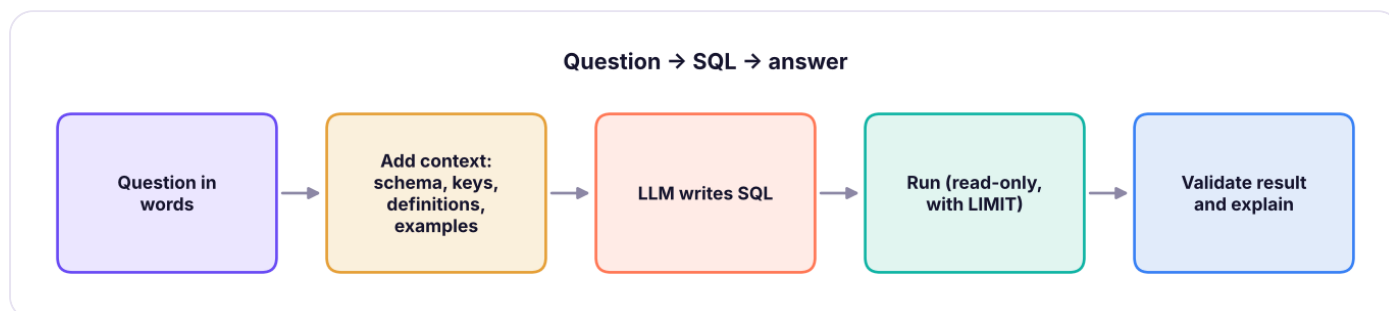
🕒 130 min

By the end of this lesson you can

- Explain how text-to-SQL works and why context (schema, definitions, examples) decides quality
- Write a schema-aware prompt and review generated SQL for the usual errors
- Install Docker and run Metabase locally; connect a database; build questions, models and dashboards
- Describe how an in-browser AI assistant such as MinusX works with Metabase, and its privacy trade-offs
- Apply guardrails: read-only roles, row limits, query review and result validation

What is text-to-SQL?

Text-to-SQL turns a business question in plain language into a SQL query: "Which region had the highest profit in 2025?" → `SELECT region, SUM(profit) ...`. LLMs are good at SQL syntax, but **correct SQL for your database** needs your context.



Text-to-SQL pipeline. Quality depends mostly on the context you supply, not on the model alone.

What goes wrong without context

Problem	Example	Prevention
Wrong tables/columns	Uses <code>customer_name</code> in <code>orders</code> , where it does not exist	Provide the schema (tables, columns, types, keys)
Wrong grain / fan-out	Joins orders to items and then <code>COUNT(*)</code> for "orders"	State the grain: "one row per order line"; add "count distinct order_id" rule
Wrong definitions	"Active customer", "net sales", "this year"	Provide a metric dictionary
Dialect mismatch	<code>LIMIT</code> instead of <code>TOP</code> in SQL Server	Name the database and version
Dates	Uses <code>YEAR(date) = 2025</code> (slow) or the wrong fiscal year	State calendar vs fiscal year and preferred patterns
Hallucinated values	<code>WHERE region = 'Northern'</code> when data says <code>North</code>	Provide sample values for categorical columns
Unsafe actions	<code>DELETE / DROP</code> suggestions	Read-only connection; review before running

A schema prompt that works

Text

```
You are a senior analytics engineer. Write SQL for Microsoft SQL Server 2022.

SCHEMA (all in dbo):
customers(customer_id PK, customer_name, segment, city, state, region)
  - segment in ('Consumer','Corporate','Home Office'); region in ('North','South','East','West')
orders(order_id PK, order_date DATE, ship_date DATE, ship_mode, customer_id FK → customers)
order_items(item_id PK, order_id FK → orders, product_id FK → products, quantity, discount, sales, profit)
  - one row per product line; sales and profit in rupees
products(product_id PK, product_name, category, sub_category, list_price)

DEFINITIONS:
- "Orders" = COUNT(DISTINCT orders.order_id). "Margin %" = SUM(profit) / SUM(sales).
- Financial year runs April-March. Use half-open date ranges (≥ start AND < next start).

QUESTION: For FY2025 (Apr 2025-Mar 2026), show margin % by category and by discount band (0, 1-20%, >20%).

RULES: Return only SQL; add comments; no SELECT *; list any assumptions after the query.
```

Notice the pieces: dialect, schema with keys, valid values, grain, definitions, date convention and output rules. This is the difference between a lucky answer and a reliable one.

TIP Keep a reusable "context file"

Save your schema, definitions and sample queries in one text file (or in an AI **Project** / custom instructions). Paste it into every session. Many BI tools (Metabase, Power BI Copilot, Snowflake Cortex, Looker) call this a **semantic layer** or "AI context": descriptions, synonyms and verified queries.

Reviewing AI-generated SQL: a checklist

1. **Does every table/column exist?** (Run it; fix any "invalid column name".)
2. **Is the grain right?** After each join, did rows multiply? Are distinct counts used for orders and customers?
3. **Are filters correct?** Date ranges half-open? Status filters? NULL handling?
4. **Do aggregates match the definition?** `SUM(profit)/SUM(sales)` not `AVG(margin)` .
5. **Does the total reconcile?** Compare with a simple query: `SELECT SUM(sales) FROM order_items` .
6. **Is it safe and cheap?** No writes; a `TOP` / `LIMIT` while testing; no accidental cross join.
7. **Is it readable?** CTEs, aliases and comments you could explain in an interview.

Let's see this in practice. A typical AI answer to "profit margin by discount band" that forgets `SUM(profit)/SUM(sales)` and averages row margins instead:

SQL

```
SELECT CASE WHEN discount = 0 THEN '0%' WHEN discount ≤ 0.2 THEN '1-20%' ELSE '>20%' END
AS discount_band,
      ROUND(AVG(profit / sales) * 100, 1) AS ai_margin_pct           -- average of row
      margins (unweighted)
FROM order_items
GROUP BY CASE WHEN discount = 0 THEN '0%' WHEN discount ≤ 0.2 THEN '1-20%' ELSE '>20%' END
ORDER BY discount_band;
```

RESULT

discount_band	ai_margin_pct
0%	18.2
1-20%	2.0
>20%	-20.5

3 rows × 2 columns

SQL

```
SELECT CASE WHEN discount = 0 THEN '0%' WHEN discount ≤ 0.2 THEN '1-20%' ELSE '>20%' END
AS discount_band,
        ROUND(100.0 * SUM(profit) / SUM(sales), 1) AS correct_margin_pct      -- weighted:
total profit / total sales
FROM order_items
GROUP BY CASE WHEN discount = 0 THEN '0%' WHEN discount ≤ 0.2 THEN '1-20%' ELSE '>20%' END
ORDER BY discount_band;
```

RESULT

discount_band	correct_margin_pct
0%	17.5
1-20%	1.2
>20%	-20.6

3 rows × 2 columns

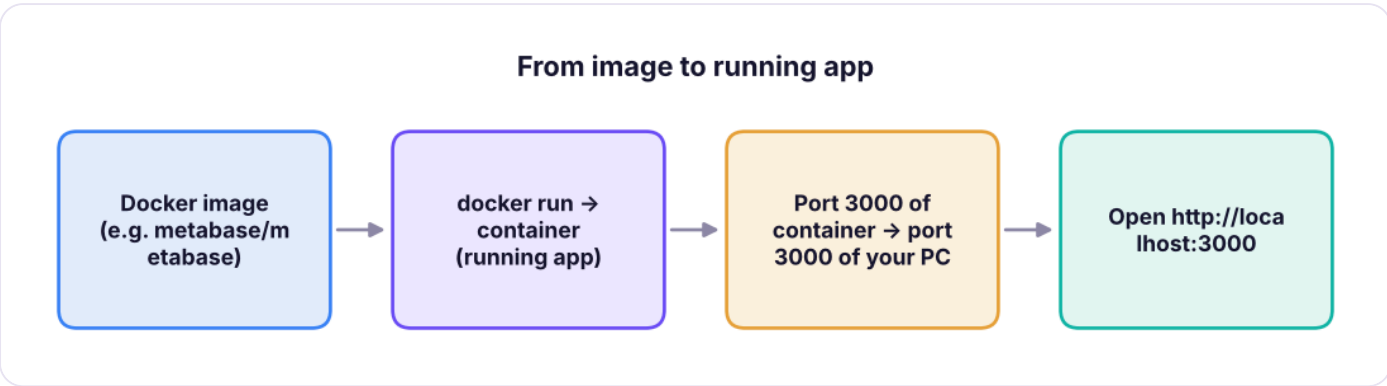
The two queries differ in every band (18.2% vs 17.5% and 2.0% vs 1.2% for the first two) because large order lines matter more in the weighted version than small ones. Which one the business wants must be written in the **definitions** part of your prompt. The checklist question "does the aggregate match the definition?" catches this.

Guardrails for letting AI near a database

Guardrail	How
Read-only login	Create a database user with SELECT only on approved views (GRANT SELECT ON ...); never give AI tools db_owner
Expose views, not raw tables	A vw_sales_for_ai view with clean names, no personal data columns
Row and time limits	Timeouts, TOP 1000, query cost limits
Masking	Hide phone, email, ID columns; use dynamic data masking
Human review	Show the SQL before running it; log queries
Verified questions	Keep a library of approved queries for key metrics
No secrets in prompts	Connection strings and passwords stay in the tool's secret store

Docker: running tools in containers

Docker packages an application and everything it needs into a **container**, so it runs the same on any machine. Instead of installing Java, a database and an app by hand, you run one command.



An image is a read-only template; a container is a running instance of it. Many containers can run from one image.

STEP BY STEP**Install Docker Desktop**

1. Download **Docker Desktop** for Windows, Mac or Linux from the Docker website and install it. On Windows it uses **WSL 2** (Windows Subsystem for Linux); the installer enables it, and you may need to restart.
2. Start Docker Desktop and wait until it says "Engine running".
3. Open a terminal (PowerShell, Terminal) and check: `docker --version` and `docker run hello-world`.
4. Note your machine needs virtualisation enabled in BIOS and about 4 GB of free RAM for small tools.

Command	Meaning
<code>docker pull metabase/metabase:latest</code>	Download an image
<code>docker run -d -p 3000:3000 --name metabase metabase/metabase</code>	Run it in the background (-d), map port 3000, name it
<code>docker ps / docker ps -a</code>	List running / all containers
<code>docker logs -f metabase</code>	Follow its logs (see start-up progress)
<code>docker stop metabase / docker start metabase</code>	Stop / start again
<code>docker rm metabase</code>	Delete the container (not the image)
<code>docker images</code>	List downloaded images
<code>docker compose up -d</code>	Start everything defined in a <code>compose.yaml</code> file

Metabase: open-source BI for everyone

Metabase is an open-source business-intelligence tool: connect a database, ask questions with a point-and-click **question builder** (no SQL needed), or write SQL, then save charts into **dashboards**. It is popular in start-ups because non-technical colleagues can self-serve.

STEP BY STEP

Run Metabase and connect your data

1. Start it: `docker run -d -p 3000:3000 --name metabase metabase/metabase`.
First start takes a minute or two (check `docker logs -f metabase`).
2. Open **<http://localhost:3000>**, create the admin account, and choose **Add your data**.
3. Pick your database type (PostgreSQL, MySQL, SQL Server, BigQuery, Snowflake, SQLite files via drivers, CSV upload in newer versions), enter host, port, database name, username and password. **Use the read-only user**. If the database also runs on your PC, use the host `host.docker.internal` from inside the container.
4. Metabase scans the tables. Browse them under **Browse data**.
5. **New → Question**: choose a table, pick columns, add **Filter**, **Summarize** (Sum of sales by region), choose a visualisation, **Save**.
6. **New → Dashboard**: add saved questions, add filters (date, region) and connect them to the question fields.

For anything you want to keep, run Metabase with a proper application database instead of the default embedded one, and mount a volume, for example:

YAML

```
# compose.yaml
services:
  metabase:
    image: metabase/metabase:latest
    ports: ["3000:3000"]
    environment:
      MB_DB_TYPE: postgres
      MB_DB_DBNAME: metabase
      MB_DB_PORT: 5432
      MB_DB_USER: metabase
      MB_DB_PASS: ${METABASE_DB_PASSWORD}      # from a .env file, never hard-coded
      MB_DB_HOST: postgres
    depends_on: [postgres]
  postgres:
    image: postgres:16
    environment:
      POSTGRES_USER: metabase
      POSTGRES_DB: metabase
      POSTGRES_PASSWORD: ${METABASE_DB_PASSWORD}
    volumes: ["pgdata:/var/lib/postgresql/data"]
volumes:
  pgdata:
```

Metabase concepts

Concept	Meaning
Question	A saved query (builder or SQL) with a visualisation
Model	A curated, cleaned table (a saved query with documented columns) for people to build on
Metric	A named, reusable calculation such as "Net revenue"
Dashboard	A page of questions with shared filters
Collection	Folders with permissions
Segment / filters	Reusable row filters
Permissions	Which groups see which databases and tables; row-level sandboxing (paid plans)
Metabot / AI features	Built-in AI assistance for questions and SQL in recent versions (availability depends on the edition and plan; check Metabase's current documentation)

Try it on the retail data: connect to your SQL Server RetailDB, then build three questions: *Sales by region*, *Monthly sales* (summarise by order_date: Month), and *Top 10 customers*. Then verify each one against this handbook's tables:

SQL

```
SELECT region, ROUND(SUM(sales), 0) AS sum_of_sales FROM superstore GROUP BY region ORDER BY sum_of_sales DESC;
```

RESULT

region	sum_of_sales
South	53,952,769
West	51,958,521
North	49,018,161
East	46,484,119

4 rows × 2 columns

MinusX: an AI assistant inside your BI tool

MinusX is an open-source AI assistant delivered as a **Chrome extension** that adds a chat sidebar to tools such as **Metabase** and **Jupyter notebooks**. Instead of copying your data into a chatbot, you work in Metabase and ask the sidebar things like *"Why did sales drop in March?"* or *"Write a query*

for monthly profit by category". According to its public description, it reads a simplified view of the page (metadata such as table and column names), then operates the tool by clicking, typing and writing queries, and lets you accept or reject its edits.

Aspect	How it behaves	What you should still do
Context	Sees the page, selected tables/columns and the query editor	Make sure the right database and tables are selected
Actions	Writes and runs SQL, edits questions, explains results	Read the SQL before accepting
Models	Supports several LLM providers (you may supply your own API key)	Check the provider's data policy
Privacy	Vendor says raw data is not sent; metadata and your prompts are	Verify the current policy; never expose sensitive tables
Cost	Free and paid plans; model API usage may cost	Check current pricing

STEP BY STEP

Using an AI assistant with Metabase

1. Install the Chrome extension from the Chrome Web Store (search "MinusX") and pin it; open your Metabase tab and open the sidebar.

2. Add your model/API key in settings if required.

3. Select the database/table in Metabase so the assistant knows the context.

4. Ask a precise question ("Monthly sales and profit by category for 2025, as a line chart; sales = SUM(sales), profit = SUM(profit)").

5. Review the generated SQL, run it, check totals against a simple query, then save it as a question.

6. Ask follow-ups: "Explain the biggest drop", "Break it down by region", "Why is Furniture negative?"

WATCH OUT

AI assistants and sensitive data

Anything the assistant can see, the model provider may receive. Use a **read-only role limited to non-sensitive views**, keep personal columns out of the connected schema, use an enterprise model agreement for company data, and keep a record of which prompts were run. If your company bans browser extensions that read pages, ask IT before installing any.

Other ways to use AI with SQL

Use	How
Explain a query	Paste the query (not the data) and ask for a line-by-line explanation and the business meaning
Optimise	Paste query + execution plan + table sizes; ask for rewrites and indexes; test each
Convert dialects	"Convert this SQL Server query to PostgreSQL; list functions that differ"
Generate test data	"Create INSERT statements for 20 fictional customers across four regions" (no real data)
Write documentation	Column descriptions, data dictionaries, comments
Data quality checks	"Write queries to find nulls, duplicates, orphans and out-of-range values for these tables"
Platform AI	Snowflake Cortex Analyst, BigQuery Gemini, Databricks Genie, SSMS Copilot, Azure Data Studio/VS Code Copilot, Fabric Copilot: all follow the same principle: context in, SQL out, you verify

2026 UPDATE **Trend: the semantic layer**

Teams that succeed with text-to-SQL do not rely on the raw schema: they define metrics (net sales, active customer) once in a **semantic layer** (dbt Semantic Layer, Cube, LookML, Power BI semantic models, Metabase models/metrics) and let AI query *that*. The AI then picks from governed metrics instead of reinventing definitions, which removes most wrong-number errors.

Summary

KEY TAKEAWAYS

- Text-to-SQL works when you provide **dialect, schema with keys, valid values, grain, metric definitions and date rules**.
- Review every query: columns exist, **grain** is right, filters correct, aggregates match the definition, totals reconcile, nothing writes.
- Protect the database with **read-only logins on curated views**, row limits and human review; never expose secrets or personal columns.
- **Docker** runs tools in containers (`docker run -d -p 3000:3000 metabase/metabase`); **Metabase** gives self-service questions, models and dashboards.
- An assistant such as **MinusX** adds an AI sidebar to Metabase; treat it as a fast junior analyst whose SQL you review, and check privacy terms.
- A governed **semantic layer** makes AI answers consistent.

Quick quiz

6 questions · tap an answer to check it

1. Why does giving the AI your schema and metric definitions improve results?

- A** It makes the model larger
- B** The model can use real table and column names and your definitions instead of guessing
- C** It speeds up the database
- D** It removes the need to review the SQL

Answer: B. Context is the main driver of text-to-SQL accuracy; review is still required.

2. Which command starts Metabase in a container on port 3000?

A `docker build metabase`

B `docker run -d -p 3000:3000 --name metabase metabase/metabase`

C `docker stop metabase`

D `metabase install`

Answer: B. `-d` runs detached and `-p 3000:3000` maps the port.

3. Which login is safest for an AI tool?

A The database owner

B A read-only user restricted to curated views

C The admin account

D An anonymous login

Answer: B. Least privilege limits damage if the AI makes a mistake.

4. A generated query averages profit/sales row by row to get a margin. Why is that risky?

A It is invalid SQL

B It is an unweighted average of ratios and may not match the business definition
 $\text{SUM}(\text{profit})/\text{SUM}(\text{sales})$

C It is slower

D It drops NULLs

Answer: B. Weighted vs unweighted aggregates differ; define the metric in the prompt.

5. What is a semantic layer?

- A A type of chart
- B A governed set of definitions for tables, metrics and relationships that BI and AI tools use**
- C A SQL function
- D A backup system

Answer: B. It standardises metrics so every query uses the same logic.

6. What should you do before saving an AI-generated Metabase question?

- A Nothing
- B Read the SQL and reconcile at least one total against a simple independent query**
- C Share it with everyone
- D Delete the database

Answer: B. Validation protects you from plausible but wrong numbers.

EXERCISE Schema prompt and review

(1) Write a schema prompt for the employees table (HR data: emp_id, gender, age, department, job_title, education, hire_date, years_at_company, monthly_income, job_satisfaction, overtime, performance_rating, city, attrition) including at least three definitions. (2) Ask any AI assistant for a query that returns attrition % by department and overtime. (3) Review the result with the checklist and reconcile the overall attrition % (25.9%) with a simple query.

Show solution (worked answer)

Definitions to include: "Attrition %" = employees with attrition = 'Yes' ÷ all employees; monthly_income is in rupees; "tenure band" = 0–2, 3–5, 6+ years. Reconciliation query: `SELECT ROUND(100.0 * SUM(CASE WHEN attrition = 'Yes' THEN 1 ELSE 0 END) / COUNT(*), 1) FROM employees;` should return 25.9. The weighted total from the grouped result (sum of leavers ÷ sum of employees across all groups) must also equal 25.9.

INTERVIEW CORNER Questions you may be asked

Q: How would you use AI to speed up SQL work safely? Provide schema, definitions and dialect; generate and explain queries; review for grain, filters and definitions; run read-only with limits; reconcile totals; keep verified queries in a library.

Q: What is Docker and why is it useful for data tools? A container platform that packages apps with their dependencies so the same setup runs anywhere; useful to spin up Metabase, databases or Jupyter in one command and share reproducible environments.

Q: What are the risks of connecting an AI assistant to a database? Data exposure to a model provider, destructive queries, wrong answers presented as facts, and prompt injection; mitigate with read-only access to curated views, masking, review and logging.

Q: What is Metabase? An open-source BI tool where users can ask questions with a visual builder or SQL, build dashboards and share them, with permission control by group.

AI in Excel & Office: Copilot, Gemini, Claude, Python in Excel & Agents

Use AI where analysts work every day. Work with Copilot in Excel (formulas, insights, charts, the COPILOT function), Gemini in Google Sheets and Excel add-ins, Claude and other assistants for formulas and files, Python in Excel, and office agents that perform multi-step tasks, and learn how to prompt, verify and stay safe.

● DVA with AI · class 3

✦ Microsoft 365 Copilot, Gemini and agent features as of 2026

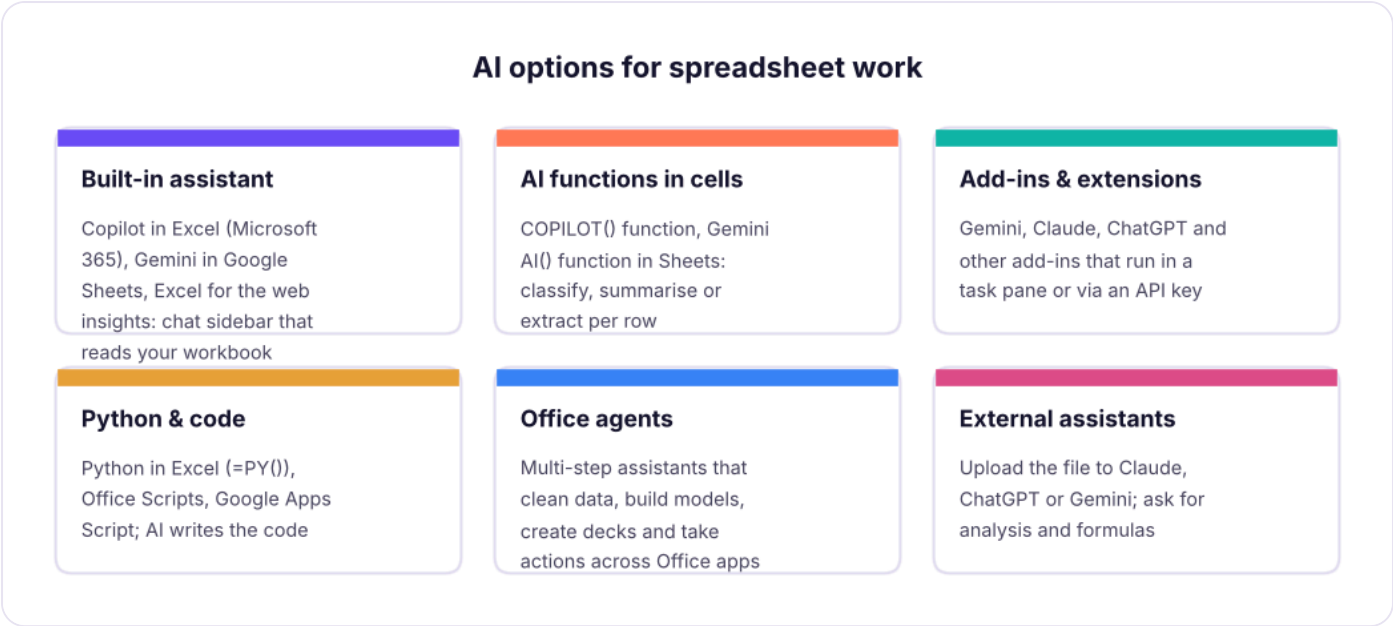
Intermediate

🕒 120 min

By the end of this lesson you can

- Prepare a workbook so AI tools understand it (Tables, clear headers, no merged cells)
- Use Copilot and Gemini-style assistants to explain, write and debug formulas, create pivots and charts, and summarise data
- Use the Excel COPILOT function and Python in Excel for text, classification and analysis inside cells
- Describe what office agents do and what controls they need
- Verify AI-generated formulas and numbers with independent checks

The AI-in-Excel landscape



Four ways AI shows up in spreadsheets.

Step 0: make your workbook AI-ready

AI assistants can only reason about data they can *see and parse*. The same rules that make pivot tables work (Excel Lesson 1) make AI work:

1. **Convert data to an Excel Table** (`Ctrl + T`) with a clear name (`tblSales`) and one header row.
2. **Descriptive column names** (`order_date` , `sales_inr`) instead of `Col1` ; add units.
3. No merged cells, blank rows or notes inside the data; keep totals outside.
4. One sheet per dataset; a small **Data dictionary** sheet that defines columns and metrics (AI reads it too).
5. Remove or mask sensitive columns before connecting an AI service.
6. Save to **OneDrive/SharePoint** (Copilot needs cloud-saved files with AutoSave on).

Copilot in Excel

Needs a Microsoft 365 Copilot licence and your data formatted as a Table or range with headers.

Task	Example prompt
Explain	"Explain what column H does and how it is calculated"
Formulas	"Add a column that shows profit margin %" / "Write a formula that looks up the manager's name from the Managers table"
Insights	"Show me insights about sales by region" → Copilot adds pivot tables and charts
Highlight	"Highlight rows where discount > 20% and profit is negative" (conditional formatting)
Sort / filter	"Sort by sales descending and filter to Technology"
Pivot & charts	"Create a pivot table of sales by category and year with YoY %"
Clean up	"Fix the inconsistent city names" / "Split full name into first and last name"
Analyse with Python	"Run a correlation between discount and profit and plot it" (uses Python in Excel)

STEP BY STEP

A safe Copilot workflow

1. Open your workbook (saved in OneDrive) → **Home** → **Copilot**.
2. Select the Table (or let Copilot detect it).
3. Ask one thing at a time, with column names: *"Add a column margin_pct = profit / sales, formatted as a percentage."*
4. **Read the proposed formula** and its explanation before clicking **Insert column**.
5. Check three rows by hand, and one total against a plain **SUM**.
6. Undo (**Ctrl + Z**) if wrong, and rephrase with more detail.

The COPILOT() function

Microsoft 365 also provides a **COPILOT** function that sends text and cell ranges to the model and returns results into cells that **spill and recalculate**:

Excel formula

```
=COPILOT("Classify this customer review as Positive, Neutral or Negative", D2:D100)
=COPILOT("Summarise the main complaint in 6 words or fewer", D2)
=COPILOT("Extract the city name from this address", A2)
```

Use it for text tasks (classify, summarise, extract, translate, tag) over **small to moderate** ranges, not for arithmetic: use normal formulas for numbers. Results can change between recalculations and

each call consumes model capacity; availability, limits and licensing change, so check Microsoft's current documentation. Convert important results to values (copy → paste values) once checked.

Here is the same classify-sentiment idea applied to the product reviews dataset, with the (non-AI) ground-truth labels already in the file, so you can see how to *measure* an AI's accuracy:

Python

```
import pandas as pd
rev = pd.read_csv("product_reviews.csv")
print(rev.shape)
rev[["review_id", "product", "rating", "review_text", "sentiment"]].head(6)
```

OUTPUT

(1200, 5)

	review_id	product	rating	review_text	sentiment
0	1	Speaker	4	Very comfortable to use all day	positive
1	2	Smartwatch	5	Very comfortable to use all day	positive
2	3	Smartphone	4	Delivery was quick and packaging was neat. great value for money, totally worth it	positive
3	4	Earbuds	1	Battery drains too fast	negative
4	5	Laptop	2	Delivery was late and the box was damaged	negative
5	6	Laptop	2	Refund process is very slow. heats up a lot while charging	negative

6 rows × 5 columns

Python

```
# How would you evaluate an AI classifier? Compare its labels with a trusted label set.
# Here a deliberately weak keyword rule plays the role of "the AI" so you can see the
# evaluation method.
# (These reviews are synthetic and templated, so real reviews will be much harder to
# classify.)
neg_words = ("stopped", "late", "poor", "crashing")          # a deliberately small word
list: an imperfect classifier
pos_words = ("excellent", "great", "amazing", "premium")
def guess(t):
    t = t.lower()
    n = sum(w in t for w in neg_words); p = sum(w in t for w in pos_words)
    return "negative" if n > p else ("positive" if p > n else "neutral")
rev["pred"] = rev.review_text.apply(guess)
acc = (rev.pred == rev.sentiment).mean()
print(f"Agreement with trusted labels: {acc:.1%}\n")
print(pd.crosstab(rev.sentiment, rev.pred, rownames=["actual"], colnames=["predicted"]))
```

OUTPUT

Agreement with trusted labels: 56.0%

	negative	neutral	positive
predicted			
actual			
negative	204	206	0
neutral	0	175	0
positive	0	322	293

The cross-tab (a **confusion matrix**, ML track Lesson 15) shows *where* a classifier goes wrong. Do the same with an AI column: take 100 reviews you labelled yourself, compare, and only then trust it on 10,000.

Gemini in Google Sheets and with Excel

- **Gemini in Google Sheets** (Workspace plans): a side panel that can *create tables, write formulas, summarise, generate charts and analyse trends*, plus an **AI()** **function** to run a prompt per row (classify, summarise, extract) similar to COPILOT().
- **Gemini and other LLM add-ins for Excel**: some marketplaces offer add-ins or custom functions (for example =GEMINI("prompt", A2)) that call a model through **your own API key**. They are useful for batch text tasks, but you must protect the key, watch the **cost per call**, and avoid sending sensitive data.
- **Apps Script / Office Scripts**: AI can write the script; you review and run it.

COMMON MISTAKES API-key hygiene for add-ins

Never paste an API key into a shared workbook or a cell others can read; use the add-in's secure settings, set a **spending limit** on the key, and rotate it if exposed. A key in a shared file is a billing and security incident waiting to happen.

Claude, ChatGPT and other assistants with your files

Upload a CSV or XLSX to a general assistant that can **run code** (analysis/code-execution tools). It reads the file, writes Python, runs it, and shows charts and tables: closer to an analyst than a chatbot that only talks.

A good first message with a file:

Text

```
I've attached superstore_sales.csv (one row per order line, Jan 2023-Dec 2025, amounts in rupees).  
1) Profile the data: row count, columns and types, missing values, duplicates, date range.  
2) Compute total sales, profit and margin by year and category.  
3) Find the three biggest drivers of low margin; quantify each.  
4) Show your Python code, and tell me which assumptions you made.  
Do not guess: if something cannot be computed from the file, say so.
```

Typical good behaviour: it prints the profile, shows code, makes charts. What you still do: spot-check numbers, make sure it used the right columns, and ask for the code so you can re-run it yourself. Assistants that run code produce **reproducible** analysis; those that only "think" may fabricate numbers.

Assistant strengths	Check
Writing and explaining formulas (even LET, LAMBDA, XLOOKUP, array formulas)	Test on sample rows
Cleaning plans and Power Query M code	Preview the result; keep a backup
Building DAX measures	Validate against a pivot
Summarising and drafting commentary	Verify each number quoted
Designing dashboards and chart choices	Compare with your audience's needs

Python in Excel

=PY(turns a cell into a Python editor (Formulas → Insert Python). It uses **pandas, matplotlib, seaborn, statsmodels and scikit-learn**, runs in Microsoft's secure cloud (no local Python needed), and reads Excel ranges with `xl()`.

Python

```
df = xl("tblSales[#All]", headers=True)          # Excel Table → pandas DataFrame
df.groupby("region")["profit"].sum().sort_values(ascending=False)
```

Ask Copilot: *"Use Python to show the correlation between discount and profit and draw a scatter plot."* Behind the scenes it writes the =PY() code. Check what it computed against this handbook's result:

Python

```
sup = pd.read_csv("superstore_sales.csv")
print(f"Correlation discount vs profit margin (lines): {sup.discount.corr(sup.profit /
sup.sales):.2f}")
print(sup.groupby(sup.discount.map(lambda d: f"{d:.0%}")).apply(lambda g:
round(g.profit.sum() / g.sales.sum() * 100, 1)).rename("margin_").to_string())
```

OUTPUT

Correlation discount vs profit margin (lines): -0.91

discount

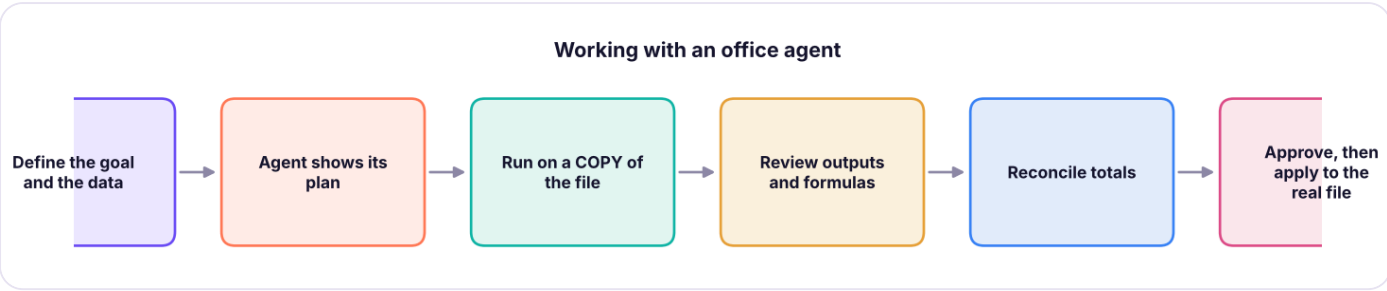
0%	17.5
10%	5.9
15%	0.9
20%	-4.6
30%	-19.0
40%	-26.3

A strong negative correlation, and margin turning negative from a 20% discount upwards: the same story as the SQL and Power BI lessons, arrived at a third way. Cross-checking tools against each other is how you build trust.

Office agents: the next step

Agents go beyond answering: given a goal ("Clean this export, build a pivot by region, add a chart, and draft a three-slide summary"), they **plan, use the spreadsheet or Office apps as tools, check results and iterate**, then present the outcome for your approval. Microsoft, Google, Anthropic, OpenAI and others have all released agent modes for office work; names and capabilities change quickly, so evaluate any specific product (including whichever "Frontier"-style agent your organisation provides) against the questions below.

Question to ask about any agent	Why it matters
What can it read and write ? (files, mail, calendar, web)	Scope of risk
Does it show its plan and steps and ask before acting?	Oversight
Can I undo or review changes (version history)?	Recoverability
Where does data go (tenant, region, retention, training use)?	Compliance
How does it handle untrusted content (prompt injection in emails/web pages)?	Security
Does it produce formulas and sources or just pasted values?	Auditability
What does it cost per task ?	Budget



A safe agent workflow: small steps, visible plan, human approval at risky points, independent checks.

What AI does well and badly in spreadsheets

Strong	Weak / risky
Explaining unfamiliar formulas and files	Large - scale precise calculation (use formulas/Python)
Drafting formulas, DAX, SQL, M code, VBA	Knowing your business rules unless told
Text tasks: classify, summarise, extract, translate	Consistent results every time (non-deterministic)
Suggesting charts and structure	Hidden assumptions (date ranges, definitions)
Debugging errors (#N/A, #SPILL!)	Anything involving sensitive data in the wrong tool
Documentation and commentary	Replacing review and sign-off

A checklist before you trust an AI-built spreadsheet

1. Totals reconcile with the source (SUM of raw vs summary).
2. Row counts match (no lost or duplicated rows).
3. Formulas are **visible** (not pasted values) and use absolute references correctly.
4. Spot-check five random rows by hand.

5. Edge cases: blanks, zero, negative, text in number columns, duplicates.
6. Units and currency are labelled; dates are real dates.
7. Assumptions are written on the sheet.
8. A second tool or person confirms the headline number.

Summary

KEY TAKEAWAYS

- Prepare data as **Tables** with clear headers, no merged cells, and a data dictionary; save in the cloud for Copilot.
- **Copilot** explains and writes formulas, creates pivots, charts, highlights and cleans data; `COPILOT()` and Sheets' `AI()` apply prompts per row for text tasks.
- General assistants that can **run code** make reproducible analyses from uploaded files; always request the code and assumptions.
- **Python in Excel** (`=PY()`) brings pandas into cells; AI can write it, you verify it.
- **Agents** plan and act across apps: work on a copy, require visible plans, approvals, undo, audit trail and clear data-handling terms.
- Verify with reconciliation, spot checks, edge cases and a second tool; protect keys and sensitive data.

Quick quiz

6 questions · tap an answer to check it

1. What should you do first to help an AI understand your Excel data?

- A** Merge headers
- B** Convert it to a clearly named Excel Table with descriptive headers and no merged cells
- C** Colour the cells
- D** Hide the columns

Answer: B. Structured, well-named data is easier for AI (and pivot tables) to use.

2. The COPILOT() function is best used for:

- A Precise financial arithmetic
- B Text tasks such as classifying, summarising or extracting over a range**
- C Replacing SUM
- D Sorting

Answer: B. Use normal formulas for numbers; use AI functions for language tasks.

3. Why ask an assistant to show its Python code?

- A To make the answer longer
- B So the analysis is reproducible and can be checked**
- C Because code is always correct
- D To avoid cleaning data

Answer: B. Visible code lets you verify and re-run the work.

4. Which is a sensible control for an office agent?

- A Let it edit the live file without review
- B Run it on a copy, review its plan and outputs, and keep version history**
- C Give it all passwords
- D Disable logging

Answer: B. Human oversight and recoverability are essential.

5. An AI classifier labels reviews; how do you measure its quality?

A Read three results

B Compare its labels with a trusted labelled sample and look at accuracy and the confusion matrix

C Use temperature 1.0

D Ask it if it is right

Answer: B. Evaluate against ground truth, not self-reports.

6. Where should an API key for an Excel add-in be stored?

A In a visible cell

B In the add-in's secure settings with a spending limit

C In the sheet name

D In an email

Answer: B. Keys in shared files can be abused and billed to you.

EXERCISE AI-assisted analysis, verified

Using `superstore_sales.csv`: (1) ask an AI assistant with code execution to profile the data and compute margin by category and discount band; (2) write the pivot table version yourself; (3) compare the two and list any differences; (4) create three `COPILOT()`-style prompts (or Gemini AI() prompts) for a column of customer comments, and say how you would evaluate them on 50 labelled examples.

Show solution (worked answer)

Expected margins by discount band: 0% → 17.5%, 1–20% → 1.2%, >20% → -20.6% (overall 9.1%). Differences usually come from averaging margins instead of total profit ÷ total sales, or from using order lines versus orders. Prompts: "Classify as Positive, Neutral or Negative (one word)"; "Extract the product mentioned (or 'none')"; "Summarise the complaint in under 8 words". Evaluation: label 50 comments yourself, run the prompt, compute accuracy and a confusion matrix, inspect errors, refine the prompt, and re-test on a fresh sample.

INTERVIEW CORNER Questions you may be asked

Q: How would you use AI tools to speed up Excel analysis responsibly? Prepare structured data, ask specific questions, review generated formulas and pivots, reconcile totals, document assumptions, keep sensitive data out of unapproved tools, and keep a human sign-off.

Q: What is Python in Excel? A feature that runs Python (pandas, matplotlib, scikit-learn) in cells via `=PY()`, in a secure cloud, reading Excel ranges with `xl()`, useful for analysis and charts that are hard with formulas.

Q: What is an AI agent in an office context and what are its risks? A system that plans and executes multi-step tasks using apps and files; risks include wrong actions, data leakage, prompt injection and lack of audit; mitigate with scoped permissions, approvals, copies and logs.

Q: How do you check an AI-generated formula? Test on a few rows including edge cases, compare totals with a simple independent calculation, read the formula's explanation, and make sure references and units are right.

The AI-Assisted Analyst Workflow, Agents & Governance (Capstone)

Combine everything into one repeatable workflow: frame a question, brief the AI, analyse with code, verify, visualise and write the story. See how an analyst agent works (plan, tools, observe, repeat) with a runnable toy example, learn MCP and tool-calling basics, and finish with governance, ethics and a telecom-churn capstone.

● DVA with AI · wrap-up

✦ 2026

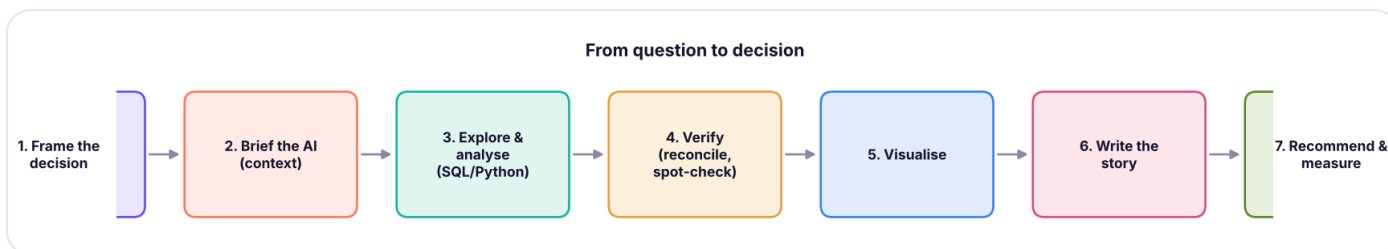
Intermediate

⌚ 140 min

By the end of this lesson you can

- Apply a seven-step AI-assisted analysis workflow from question to recommendation
- Explain the agent loop, tool calling and the Model Context Protocol (MCP) at a conceptual level
- Build a tiny agent-style loop that plans, runs SQL tools and reports, and describe where an LLM would sit
- Design human-in-the-loop controls, logging and evaluation for AI analytics
- Complete an end-to-end churn analysis and explain it to a business audience

The AI-assisted analyst workflow



Seven steps. AI accelerates several of them, but the analyst owns the first and the last two.

Step	What you do	Where AI helps	Your responsibility
1. Frame	Decision, audience, metric definitions, constraints	Brainstorm questions, critique your framing	Own the decision question
2. Brief	Schema, definitions, sample rows, rules	Turn notes into a clean brief	No sensitive data; correct definitions
3. Analyse	Profile, clean, aggregate, model	Draft SQL/Python, explain code, suggest checks	Run it; read it
4. Verify	Reconcile totals, test edge cases, second method	Write test queries, find anomalies	Sign off the numbers
5. Visualise	Choose and label charts	Suggest chart types, write plotting code	Honest, readable charts
6. Write	Insight, caveat, action	Draft commentary, simplify language	Every claim traceable to a number
7. Recommend	Action, owner, metric, date	Pre-mortem: "how could this fail?"	Business judgement

Case: why do telecom customers leave?

CASE STUDY Brief

A telecom company has 3,000 subscribers in `telecom_churn.csv` (tenure, contract type, internet service, payment method, charges, support calls, churn Yes/No). The retention head asks: **Who is most likely to churn, and what should we do about it?** We will follow the workflow and use AI-style assistance where it helps, checking everything.

Steps 1–2: frame and brief

Text

```
Decision: choose which customers get a retention offer next quarter (budget: 300 offers).
Metric: churn rate = churners / customers in the group. Report groups with at least 50
customers only.
Schema: telecom_churn(customer_id, senior_citizen 0/1, tenure_months, contract {Month-to-
month, One year, Two year},
        internet_service {Fiber, DSL, No internet}, payment_method, monthly_charges,
total_charges, support_calls, churn {Yes, No})
Rules: use only this table; state assumptions; do not claim causes; show code.
```

Step 3: profile and analyse

Python

```
import pandas as pd, numpy as np
tc = pd.read_csv("telecom_churn.csv")
tc["churned"] = (tc.churn == "Yes").astype(int)
print(f"{len(tc):,} customers | overall churn {tc.churned.mean():.1%} | missing values: {int(tc.isna().sum().sum())} | duplicate IDs: {int(tc.customer_id.duplicated().sum())}")
tc.describe(include="all").T[["count", "unique", "top", "mean", "min", "max"]].round(1).head(10)
```

OUTPUT

3,000 customers | overall churn 25.7% | missing values: 0 | duplicate IDs: 0

	count	unique	top	mean	min	max
customer_id	3000	3000	T00001	NaN	NaN	NaN
senior_citizen	3,000	NaN	NaN	0.17	0	1
tenure_months	3,000	NaN	NaN	36.06	0	72
contract	3000	3	Month-to-month	NaN	NaN	NaN
internet_service	3000	3	Fiber	NaN	NaN	NaN
payment_method	3000	4	Card	NaN	NaN	NaN
monthly_charges	3,000	NaN	NaN	67.07	20	131.74
total_charges	3,000	NaN	NaN	2,418.32	19.58	8,724.35
support_calls	3,000	NaN	NaN	1.28	0	6
churn	3000	2	No	NaN	NaN	NaN

10 rows × 6 columns

Churn by one factor at a time (the same GROUP BY thinking as SQL Lesson 7):

Python

```
def churn_by(col, bins=None, min_n=50):
    x = tc[col] if bins is None else pd.cut(tc[col], bins)
    g = tc.groupby(x, observed=True).agg(customers=("churned", "size"), churners=
("churned", "sum"))
    g["churn_%"] = (g.churners / g.customers * 100).round(1)
    return g[g.customers >= min_n]
print(churn_by("contract").to_string()); print();
print(churn_by("internet_service").to_string()); print()
print(churn_by("support_calls").to_string()); print(); print(churn_by("tenure_months",
bins=[-1, 6, 12, 24, 48, 72]).to_string())
```

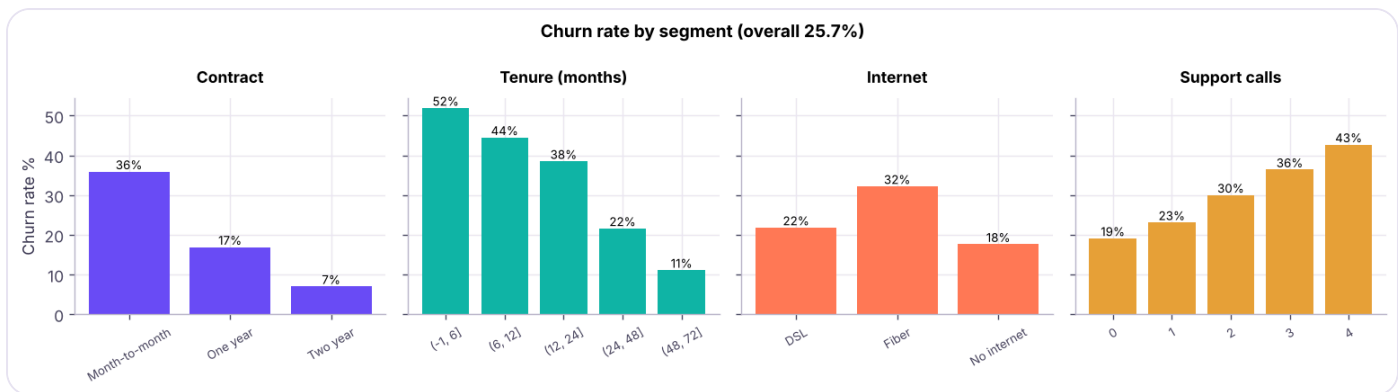
OUTPUT

	customers	churners	churn_%
contract			
Month-to-month	1690	607	35.9
One year	739	125	16.9
Two year	571	40	7.0

	customers	churners	churn_%
internet_service			
DSL	1047	228	21.8
Fiber	1368	441	32.2
No internet	585	103	17.6

	customers	churners	churn_%
support_calls			
0	822	157	19.1
1	1101	254	23.1
2	649	195	30.0
3	294	107	36.4
4	110	47	42.7

	customers	churners	churn_%
tenure_months			
(-1, 6]	318	165	51.9
(6, 12]	238	106	44.5
(12, 24]	452	174	38.5
(24, 48]	1011	218	21.6
(48, 72]	981	109	11.1



Churn rate by contract type, tenure band, internet service and support calls. Month-to-month customers, new customers, fibre users and frequent callers churn most.

Step 4: verify before you believe

Python

```
# 1. Reconcile: group counts add up; 2. a second method gives the same answer; 3. check
small groups
g = churn_by("contract", min_n=0)
print("Group customers sum to total :", g.customers.sum() == len(tc))
print("Group churners sum to total :", g.churners.sum() == tc.churned.sum())
import duckdb
sql = duckdb.sql("SELECT contract, COUNT(*) n, ROUND(100.0*AVG(CASE WHEN churn='Yes' THEN 1
ELSE 0 END),1) churn_pct FROM tc GROUP BY contract ORDER BY contract").df()
print("SQL cross-check:\n", sql.to_string(index=False))
print("\nCombinations with fewer than 50 customers are excluded from rankings; check:",
(tc.groupby(['contract', 'internet_service']).size() < 50).sum(), "such groups")
```

OUTPUT

```
Group customers sum to total : True
Group churners sum to total : True
SQL cross-check:
```

	contract	n	churn_pct
Month-to-month	1690	35.9	
One year	739	16.9	
Two year	571	7.0	

```
Combinations with fewer than 50 customers are excluded from rankings; check: 0 such
groups
```

Associations are **not causes**: month-to-month customers may also be newer or on fibre. To look at several factors together, combine them:

Python

```
tc["tenure_band"] = pd.cut(tc.tenure_months, [-1, 12, 72], labels=["≤12 months", ">12 months"])
combo = tc.groupby(["contract", "tenure_band"], observed=True).agg(customers=("churned", "size"), churn_pct=("churned", lambda s: round(s.mean() * 100, 1)))
combo[combo.customers ≥ 50]
```

OUTPUT

		customers	churn_pct
contract	tenure_band		
Month-to-month	≤12 months	306	67.0
	>12 months	1384	29.0
One year	≤12 months	151	33.8
	>12 months	588	12.6
Two year	≤12 months	99	15.2
	>12 months	472	5.3

6 rows × 2 columns

Steps 5–7: prioritise and recommend

With a budget of 300 offers, focus on the **highest-risk group that is also large enough to matter**. A simple **risk × size** table:

Python

```
seg = tc.groupby(["contract", "internet_service"], observed=True).agg(customers=("churned", "size"), churn_pct=("churned", "mean"), avg_charges=("monthly_charges", "mean"))
seg["expected_churners"] = (seg.customers * seg.churn_pct).round(0)
seg["churn_pct"] = (seg.churn_pct * 100).round(1); seg["avg_charges"] = seg.avg_charges.round(0)
seg = seg[seg.customers ≥ 50].sort_values("expected_churners", ascending=False)
seg.head(6)
```

OUTPUT

		customers	churn_pct	avg_charges	expected_churn
contract	internet_service				
Month-to-month	Fiber	743	43.5	87	323
	DSL	617	30.8	61	190
	No internet	330	28.5	31	94
One year	Fiber	341	26.7	86	91
	DSL	252	11.1	61	28
Two year	Fiber	284	9.5	87	27

6 rows × 4 columns

KEY TAKEAWAYS

A recommendation a manager can act on

1. **Target** the top segments by *expected churners*, not only by churn rate: they hold most of the preventable loss.
2. **Offer** a discounted switch to a 1-year contract for month-to-month customers in their first year, and a service check for fibre customers with 2+ support calls.
3. **Measure** with an A/B test: random half of the target list gets the offer; compare churn after 90 days (the test design is in the ML track, hypothesis-testing lessons).
4. **Limit:** this analysis is descriptive; it does not prove that contracts *cause* churn or estimate the offer's effect.

How an AI agent does analysis

An agent turns the workflow above into a loop that an LLM drives:



The agent loop. The model decides the next action, a tool executes it, and the result goes back into the model's context.

Piece	Meaning	Analytics example
Goal	What to achieve	"Find the top churn drivers and draft a memo"
Tools	Functions the model can call	<code>run_sql(query)</code> , <code>run_python(code)</code> , <code>plot(spec)</code> , <code>search_docs(text)</code>
Memory / context	What it has seen so far	Previous queries and results
Planner (LLM)	Chooses the next action	"Check churn by contract next"
Guardrails	Limits	Read-only DB, max queries, no network, human approval for writes
Evaluation	How you know it worked	Unit tests on metrics, golden questions, human review

A toy agent you can run (no LLM)

To make the loop concrete, here is a miniature "analyst agent" in which a hard-coded planner stands in for the LLM. It has **one tool** (read-only SQL) and a rule: stop when it has found the top three churn drivers.

Python

```
import duckdb
db = duckdb.connect(); db.execute("CREATE TABLE tc AS SELECT * FROM
read_csv_auto('telecom_churn.csv')")

def run_sql(q):
    # TOOL: read-only SQL with a row limit
    assert q.strip().lower().startswith("select"), "only SELECT statements are allowed"
    return db.execute(q).df().head(20)

def planner(memory):
    # stands in for the LLM: decides the next
    tool call
    todo = [c for c in ["contract", "internet_service", "payment_method", "support_calls"]
if c not in memory]
    if todo and len(memory) < 4:
        c = todo[0]
        return f"SELECT {c} AS grp, COUNT(*) AS n, ROUND(100.0*AVG(CASE WHEN churn='Yes'
THEN 1 ELSE 0 END),1) AS churn_pct FROM tc GROUP BY {c} HAVING COUNT(*) ≥ 50", c
    return None, None

memory, log = {}, []
while True:
    q, key = planner(memory)
    if q is None: break
    res = run_sql(q); memory[key] = res; log.append(f"ran query on '{key}' → best group:
{res.sort_values('churn_pct').iloc[-1].grp} ({res.churn_pct.max()}%)")
    print("\n".join(log))
    spread = {k: v.churn_pct.max() - v.churn_pct.min() for k, v in memory.items()}
    print("\nDrivers ranked by spread in churn rate (percentage points):")
    for k, v in sorted(spread.items(), key=lambda x: -x[1]): print(f" {k:18s} {v:5.1f}")
```

OUTPUT

```
ran query on 'contract' → best group: Month-to-month (35.9%)
ran query on 'internet_service' → best group: Fiber (32.2%)
ran query on 'payment_method' → best group: Net banking (28.5%)
ran query on 'support_calls' → best group: 4.0 (42.7%)
```

Drivers ranked by spread in churn rate (percentage points):

contract	28.9
support_calls	23.6
internet_service	14.6
payment_method	5.0

A real agent replaces `planner` with an LLM call that reads the memory and writes the next SQL, adds many more tools, and decides when to stop. Everything else (read-only guard, row limit, log) is the same safety scaffolding you should demand from any agent.

Tool calling and MCP

- **Tool (function) calling:** the model returns a structured request such as `{"tool": "run_sql", "arguments": {"query": "SELECT ..."} }`; your code runs it and sends the result back.
- **Model Context Protocol (MCP)** is an open standard for connecting AI applications to **tools and data sources** through a common interface, so one connector (a "server") for your database, files or SaaS app works with many AI clients. For analysts it means an assistant can query a **governed** database or semantic layer through an approved server instead of you pasting data around.
- **RAG (retrieval-augmented generation)** gives the model *documents* (metric definitions, past reports) to ground its answers (GenAI track).
- **Multi-agent systems** split work among specialised agents (a SQL agent, a chart agent, a reviewer); more moving parts mean more to test.

Evaluating AI analytics

What to evaluate	How
Correctness of numbers	A set of "golden questions" with known answers (total sales by year, churn by contract) run on every change
SQL quality	Execution accuracy (does the result match the reference result?), not just text similarity
Faithfulness of narrative	Every number in the text must appear in the tables; automatic check by extracting numbers
Safety	Attempted writes blocked, personal data not exposed, prompt-injection tests
Cost and latency	Tokens, tool calls and time per task
Usefulness	Do users make better decisions? Track adoption and feedback

Governance, ethics and your career

- **Policy:** know your organisation's approved tools, data-classification rules and retention.
- **Human in the loop** for decisions that affect people (credit, hiring, pricing, health): AI assists; accountable humans decide.
- **Fairness:** test whether results differ unfairly across groups (gender, age, region) before acting.
- **Explainability:** record the data, prompt, code and checks behind each insight; be able to re-run it.
- **Skills that stay valuable:** asking the right question, domain knowledge, data modelling, SQL/Python fluency to *read and verify* AI output, statistics and experimentation, storytelling and stakeholder management. The analysts who thrive are those who use AI for speed **and** can prove their answers.

Analytics agents are moving into the BI tools themselves (Copilot in Power BI/Fabric, Gemini in Looker and BigQuery, Snowflake Cortex, Databricks Genie, Metabase AI features). Vendors compete on accuracy with governed semantic layers, so expect "ask a question of certified metrics" to become a standard feature. Learn the concepts here and re-check each product's current capabilities, pricing and data-handling terms before adopting it.

Summary

KEY TAKEAWAYS

- Follow a **seven-step workflow**: frame, brief, analyse, verify, visualise, write, recommend; AI accelerates but you own framing, verification and judgement.
- Always **verify**: reconcile totals, cross-check with a second method, exclude tiny groups, and remember associations are not causes.
- An **agent** = goal + LLM planner + tools + memory + guardrails in a loop; tool calling and **MCP** connect models to tools and governed data.
- Evaluate AI analytics with golden questions, execution accuracy, faithfulness checks, safety tests and cost tracking.
- Keep humans accountable, log the work, check fairness and follow data policy.

Quick quiz

6 questions · tap an answer to check it

1. In the agent loop, which part executes a SQL query?

A The planner

B A tool

C The memory

D The prompt

Answer: B. The LLM plans; tools perform actions such as running SQL.

2. What is MCP?

- A A chart type
- B An open protocol for connecting AI applications to tools and data sources
- C A database engine
- D A prompt format

Answer: B. The Model Context Protocol standardises how AI clients use external tools and data.

3. Why report churn only for groups with at least 50 customers?

- A To shorten the report
- B Small groups give unstable percentages that can mislead
- C Because SQL requires it
- D To hide data

Answer: B. Small samples produce noisy rates.

4. Which statement about our churn findings is correct?

- A Contracts cause churn
- B Month-to-month customers churn more, but this is an association that needs testing before claiming cause
- C Fibre causes churn
- D Support calls prove dissatisfaction

Answer: B. Descriptive analysis shows patterns, not causes; use an experiment to test an intervention.

5. What is a "golden question" set?

- A** Questions that cost the most
- B** Questions with known correct answers used to test an AI analytics system after each change
- C** A list of dashboards
- D** Training data

Answer: B. They enable regression testing of answers.

6. Which control best limits damage if an agent writes a bad query?

- A** A longer prompt
- B** A read-only database role with row limits and logging
- C** A bigger model
- D** A louder warning

Answer: B. Technical guardrails beat instructions alone.

EXERCISE Run the workflow on a new question

Using `superstore_sales.csv`, follow the seven steps for: *"Which customer segment × region combinations should sales focus on next quarter?"* Write the brief, run the analysis (use `groupby` or SQL), verify with two methods, produce one chart and write a five-sentence recommendation that includes one limitation.

Show solution (worked answer)

Brief: define "focus" (for example profit, not just sales; minimum 100 order lines). Analysis: group by segment and region, compute sales, profit, margin and lines; filter groups with ≥ 100 lines. Verify: group totals equal the overall totals; repeat with SQL; check one group manually. Chart: bubble or bar of profit by segment × region. Recommendation example: "Focus on [top profit segment-region] (profit ₹X, margin Y%); protect [healthy group]; reduce discounts in [low-margin group]. Limitation: three years of data, no cost-to-serve or competitor information; profits include discounts that may not repeat."

INTERVIEW CORNER Questions you may be asked

Q: Walk me through how you use AI in your analysis workflow. I frame the decision, give the AI a clear brief with schema and definitions, use it to draft and explain SQL or Python, run everything myself, reconcile totals and cross-check with a second method, use AI for chart and narrative drafts, then I validate each number and take responsibility for the recommendation.

Q: What is an AI agent and how is it different from a chatbot? A chatbot responds once to a prompt; an agent pursues a goal across multiple steps by calling tools, observing results and deciding what to do next, so it needs stronger guardrails and evaluation.

Q: How do you stop an AI analytics tool from giving wrong numbers? Governed semantic layer, read-only access to curated data, golden-question tests, execution-based evaluation, human review of important outputs, and logging.

Q: A model says "contract type drives churn". What do you say? The data shows an association; to claim causation we would need an experiment or a causal method, controlling for tenure and other factors.