

Homework — Databases

Estimated time: ~50 min · Almost no code · A browser, a free account or two, and a notebook

Three tool-tour exercises. The point is to *touch* three very different shapes of database in your own browser: a classic SQL engine, a document store, and a vector / similarity-search world. By the end you'll have run an actual SQL query, opened a real JSON document inside MongoDB, and seen — without writing a line of code — what makes vector search different from keyword search.

Before you start

Keep a one-page notebook open (paper or a text file). For each task, write the short answers under (a)/(b)/(c). You are not turning this in — this is the muscle of *poking at* a database, not just reading about one. The only "code" in this homework is modifying *one* clause of one pre-written SQL query in Task 1.

The three tasks

Task 1 — Touch a SQL database in the browser

Open sqliteonline.com (or db-fiddle.com if you prefer Postgres/MySQL).

Pick the demo *Chinook* database, or any sample dataset that comes preloaded — most of these sites ship one. Run the example query that's already sitting in the editor; you should see a small table of results appear below it.

Now change **one thing**. You do not need to know SQL. Examples of a single-word change:

- add a `WHERE` clause, e.g. `WHERE Country = 'USA'`
- change `LIMIT 10` to `LIMIT 100`

- change the table name in `FROM` to a different table from the schema panel on the left

Re-run the query. Notice that the result table changes shape.

Self-check — answer all three:

- **(a)** How many rows did the *original* query return?
- **(b)** How many rows did your *modified* query return?
- **(c)** Take a screenshot of one query and its result table for your own notes.

✓ You do NOT need to know SQL — you only need to see that a query produced a table, and that changing one clause changed the table.

Task 2 — Tour MongoDB Atlas (free tier)

Go to `cloud.mongodb.com` and sign up. The **M0** free tier needs no credit card. Create a free cluster — call it `landscape` — and wait until it shows *Active* (usually 1–3 minutes).

Then click **Browse Collections** → **Load Sample Dataset**. Wait another ~2 minutes for it to import. Once it's done, open the `sample_mflix` database → `movies` collection. Click any movie document to expand it — you'll see a tree of fields like `title`, `year`, `cast`, `genres`. Click a few more documents and notice their fields are *not* identical — some have `awards`, some don't; some have `tomatoes`, some don't.

Self-check — answer all three:

- **(a)** One sentence: how does this differ from a SQL table? (Hint: think about the *shape* of each row.)
- **(b)** How many documents are in the `movies` collection? (Atlas shows the count in the collection header.)
- **(c)** Take a screenshot of one document expanded so its JSON shape is visible.

Cleanup

When you're done, you can delete the cluster from **Database** → ... → **Terminate**. The M0 free tier is fine to keep running — it never charges — but if you don't expect to come back, tear it down.

Task 3 — See vector / similarity search in action

Open claude.ai (or chat.openai.com or any chatbot you have access to). Ask:

"Find me five animals that are conceptually similar to an octopus but are not cephalopods."

Read the answer. Notice what just happened: the model didn't grep Wikipedia for the literal string *"octopus"*. It found things that are similar in *meaning* — intelligent, soft-bodied, aquatic, problem-solving. That's the core of how a vector database feels from the outside.

Then open github.com/pgvector/pgvector. Read just the **Getting Started** section of the README — don't try to install anything, just read. Pay attention to the `<->` operator in the example queries.

Self-check — answer all three:

- **(a)** Does the chatbot's answer feel like *keyword* search or like *similarity* search? One sentence on why.
- **(b)** What does the `<->` operator do in pgvector? (One short sentence from the README.)
- **(c)** Write one sentence on why a vector database is a natural fit for retrieval-augmented generation (RAG). Lesson 13 returns to this topic in depth.

What "done" looks like

You're done when, in your notebook, you have:

- three row-count numbers and one screenshot from Task 1 (SQL)
- one shape-difference sentence, one document count, and one screenshot from Task 2 (MongoDB)
- three short sentences from Task 3 (vector / similarity)

The aim was never to make you a DBA in 50 minutes. It was to make sure that the next time someone says *"we'll just shove it in Postgres"* or *"that needs a vector store"*, you have a picture in your head of what they're actually pointing at — because you've now opened all three.

Where this goes next

Module 3 puts data in motion: how it travels over the network (Lesson 10), how systems talk to each other (Lesson 11), and how all of this lives in the cloud (Lesson 12). Module 4 then comes back to databases through the AI lens — embeddings, vector indexes, and RAG — so Task 3 is a foretaste, not a deep dive.