Tabellum for desktop

User Guide

Version 1.0 · 08.11.2026

Contents

1 Getting started 4
1.1 The window 4
1.2 The demo database 5
1.3 Your workspace 5
2 Connections 7
2.1 Adding a connection 7
2.2 Cloud databases 8
2.3 Pasting a connection string 9
2.4 SQLite files 9
2.5 Editing a connection 10
2.6 SSH and WireGuard tunnels 10
2.7 Sharing connections 12
3 Browsing and editing data 15
3.1 The schema tree 15
3.2 A table’s rows 15
3.3 Editing cells 16
3.4 The inspector 17
4 Writing SQL 18
4.1 The query tab 18
4.2 Completion 18
4.3 Parameters 19
4.4 The query plan 19
4.5 The command palette 20
4.6 The approval dialog 20
5 Charts, dashboards and export 22
5.1 Charts 22
5.2 Dashboards 23
5.3 Exporting results 24
6 Diagrams, snapshots and comparing schemas 25
6.1 Schema diagrams 25
6.2 Snapshots 25
6.3 Comparing two schemas 25
7 The schema designer 27
7.1 Starting a design 27
7.2 The canvas 28
7.3 Reviewing and applying 28
7.4 When the database has changed 30
7.5 Full-screen editors 30
7.6 Creating a database 31
8 Views, users and roles 32
8.1 Making a view without writing SQL 32
8.2 Users and roles 32
9 The AI agent 33
9.1 Setting it up 33
9.2 Asking 34
9.3 What the agent can and cannot do 35
10 Backup, restore and sync 36
10.1 Backing up 36

10.2 Restoring 36
10.3 Syncing the workspace with git 36
11 Reference 38
11.1 Keyboard shortcuts 38
11.2 Databases 38
11.3 The Mac App Store version 38
11.4 This guide in Tabellum 38
11.5 If something goes wrong 38

1 Getting started

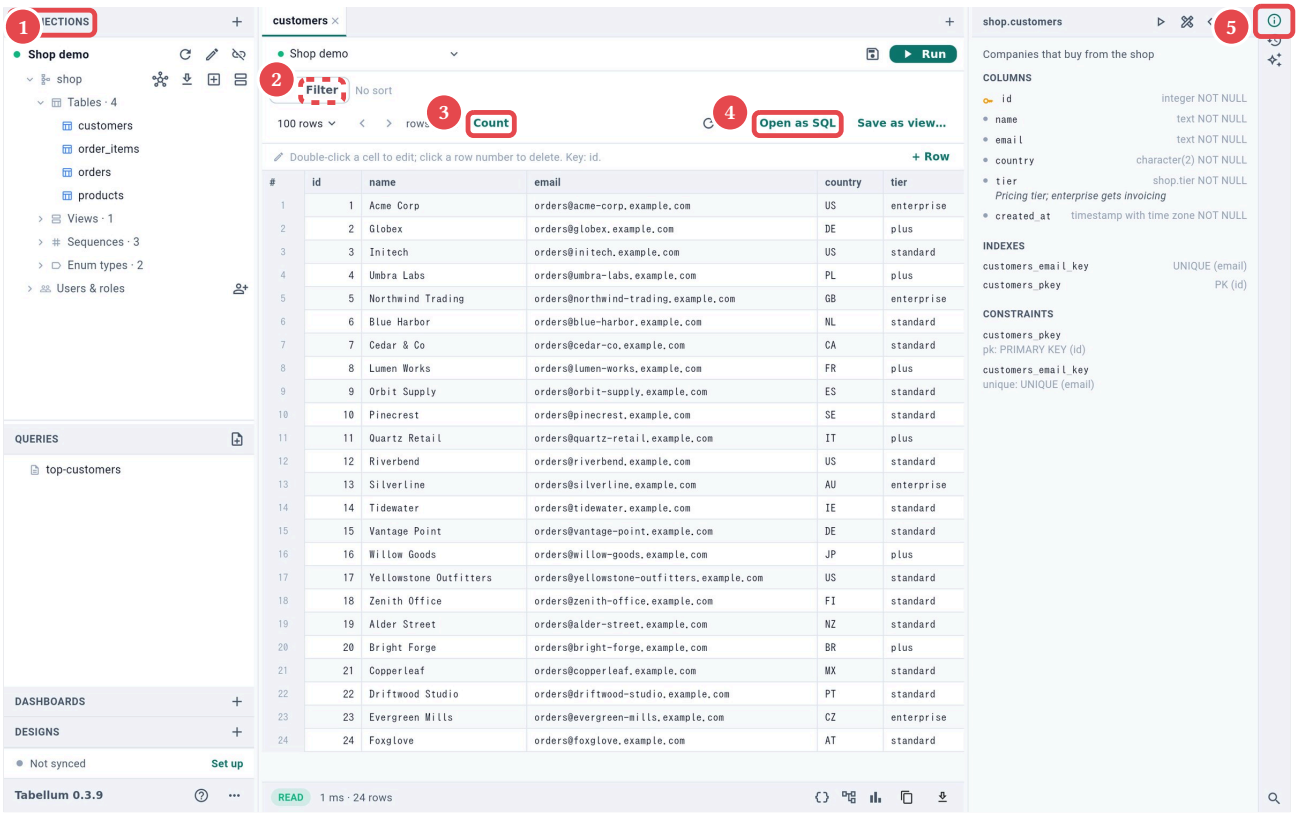
Tabellum is a database client for PostgreSQL, MySQL and MariaDB, SQL Server, BigQuery, SQLite and MongoDB. It browses and edits data, runs SQL, draws charts and dashboards, compares and designs schemas, and has an AI assistant that never writes to a database without your approval.

Everything you create — connections, saved queries, dashboards, designs, snapshots — is a plain file in your workspace folder. Passwords and API keys are not: they stay in the system's secure storage, the keychain on a Mac.

This guide covers Tabellum on macOS, Windows and Linux. The screenshots come from a Mac; on Windows and Linux the window is the same, and wherever the guide says `Cmd`, press `Ctrl`. The few things only a Mac has say so.

1.1 The window

The window has four parts: the sidebar on the left, tabs in the middle, the results under the editor, and panels on the right.



1. Connections, their schemas and objects
2. Filter the rows
3. Count every row of the table
4. The same rows as an editable query
5. Inspector, history and agent panels

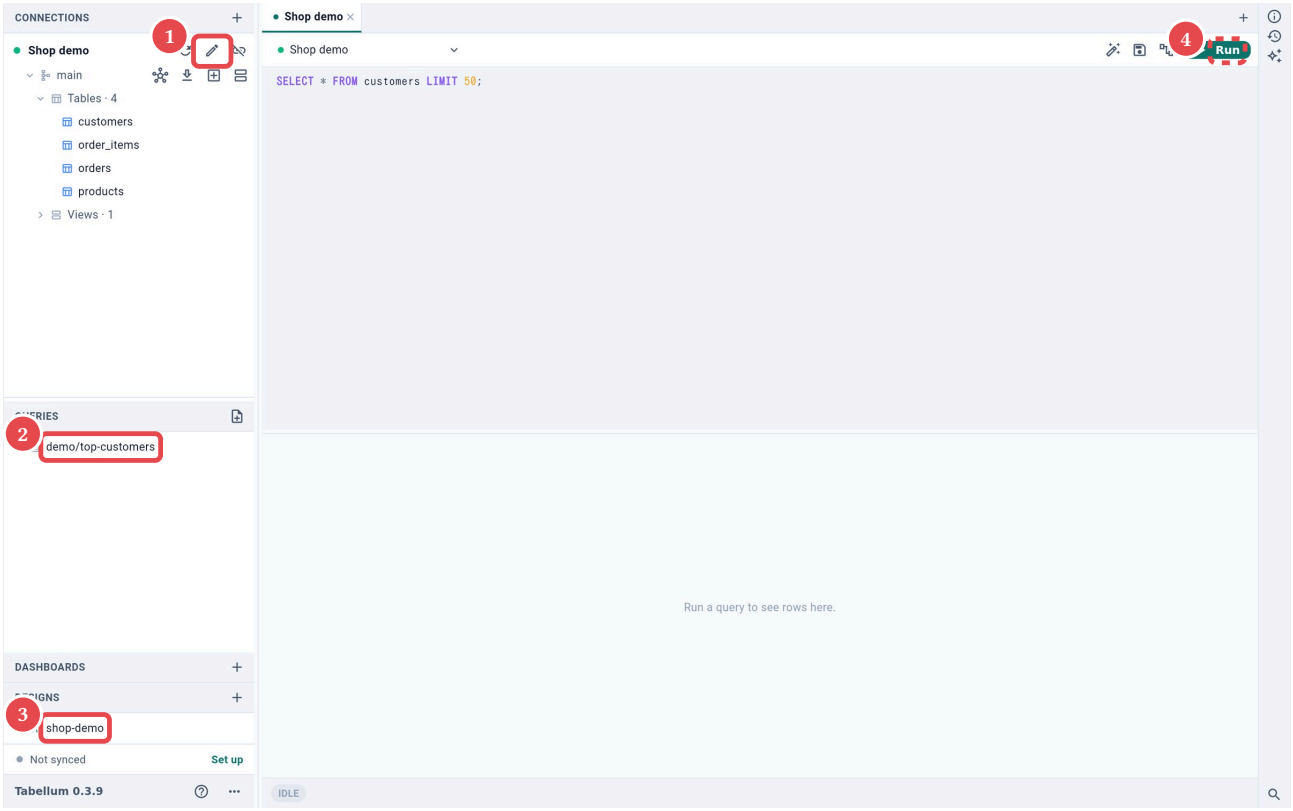
Figure 1: A table open in a tab, with the inspector beside it

- The sidebar 1 lists your connections. Open one to see its schemas, tables, views, functions, sequences, enum types, and users and roles. Under them are your saved queries, dashboards and designs.
- A click on a table opens its rows in a tab. **Filter** 2 narrows them, **Count** 3 counts every row, and **Open as SQL** 4 turns the same view into a query you can edit.

- The buttons on the right edge (5) open the inspector (the selected object's columns, indexes and constraints), the history of what ran, and the AI agent.

1.2 The demo database

The first time Tabellum starts, its workspace already holds a small demo: an invented shop in an SQLite file inside the app, a saved query, and a design of its schema. Nothing needs a network or a server, so it is the quickest way to try every feature. Deleted demo files stay deleted; to get them back, choose **Open the demo database** in the ··· menu at the bottom of the sidebar (on a Mac, also in the **File** menu).



1. Edit the connection
2. An example saved query
3. A design of the demo schema
4. Run the query

Figure 2: The demo connection, its example query and its design

1. Click **Shop demo** (1) in the sidebar to connect.
2. Open the example query (2) and click **Run** (4).
3. Open the design (3) to see the schema designer (see Section 7).

1.3 Your workspace

The workspace is a folder: `Tabellum` in your Documents folder, unless you open another one. Inside it, `connections/` holds the connections, `queries/` the saved queries, `dashboards/`, `designs/` and `snapshots/` the rest, and `sqlite/` the SQLite databases that belong to the workspace. The files are readable YAML, SQL and JSON, so they can be kept in a git repository and shared (see Section 10.3).

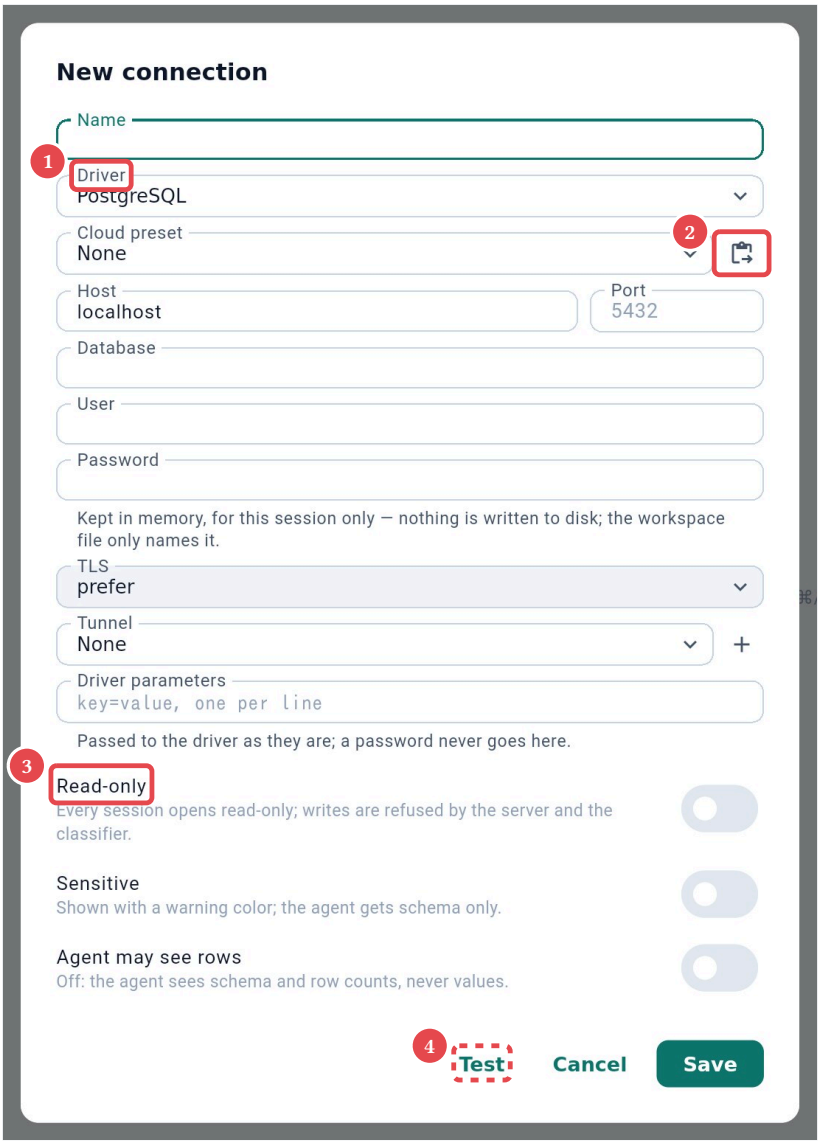
> **Note**
> A connection file never contains a password. The password is in the secure storage of the computer it was typed on; a teammate who opens the same workspace types their own.

2 Connections

A connection is the address of one database and how to log in to it. Tabellum keeps as many as you like and can have several open at once.

2.1 Adding a connection

1. Click `+` next to **CONNECTIONS** in the sidebar.
2. Pick the **Driver** (1): PostgreSQL, MySQL / MariaDB, SQL Server, BigQuery, SQLite or MongoDB.
3. Fill in the host, port, database, user and password.
4. Click **Test** (4). When the test passes, click **Save**.



1. The database type
2. Fill the form from a connection string
3. Refuse every write on this connection
4. Try the connection before saving

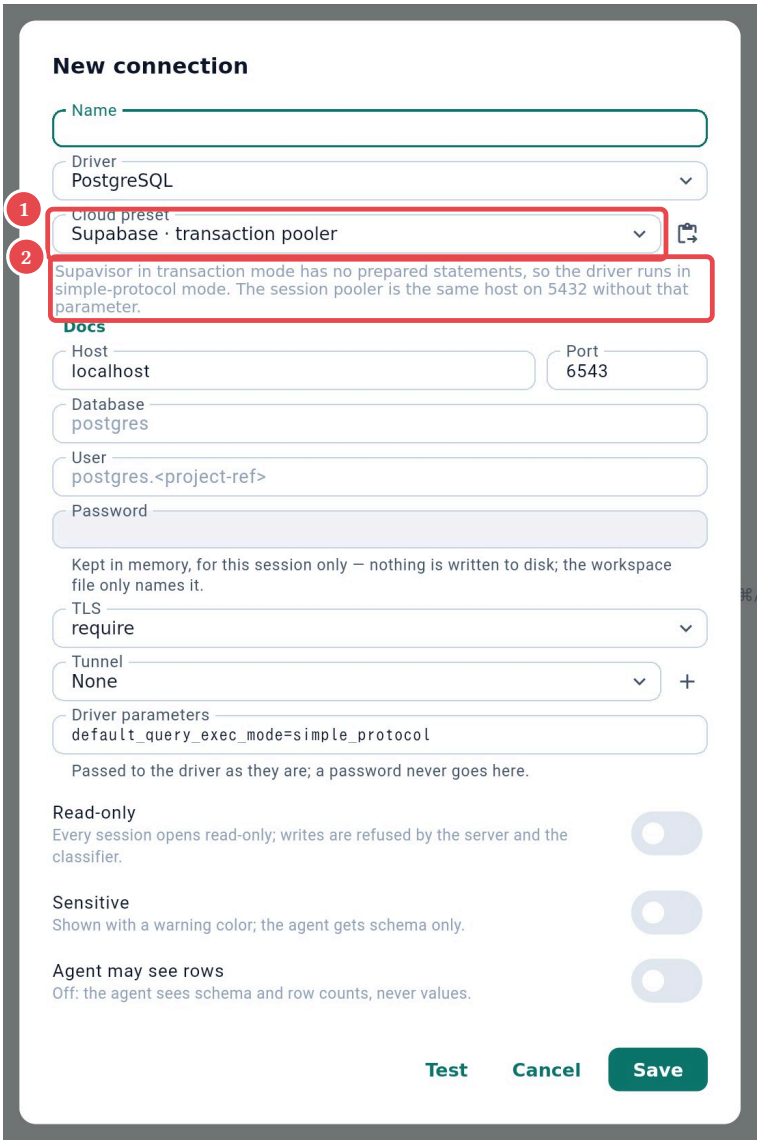
Figure 3: A new connection

Three switches decide what may happen on the connection:

- **Read-only** (3): every session opens read-only. The server refuses writes, and Tabellum refuses them before they reach it.
- **Sensitive**: the connection is drawn in a warning color, and the AI agent gets only its schema.
- **Agent may see rows**: off by default. While it is off, the agent sees tables, columns and row counts, never values.

2.2 Cloud databases

For a hosted database, pick its provider under **Cloud preset** (1). The preset fills in the port, TLS mode and driver settings the provider needs and explains what is special about it (2) — for example, that a transaction pooler has no prepared statements.

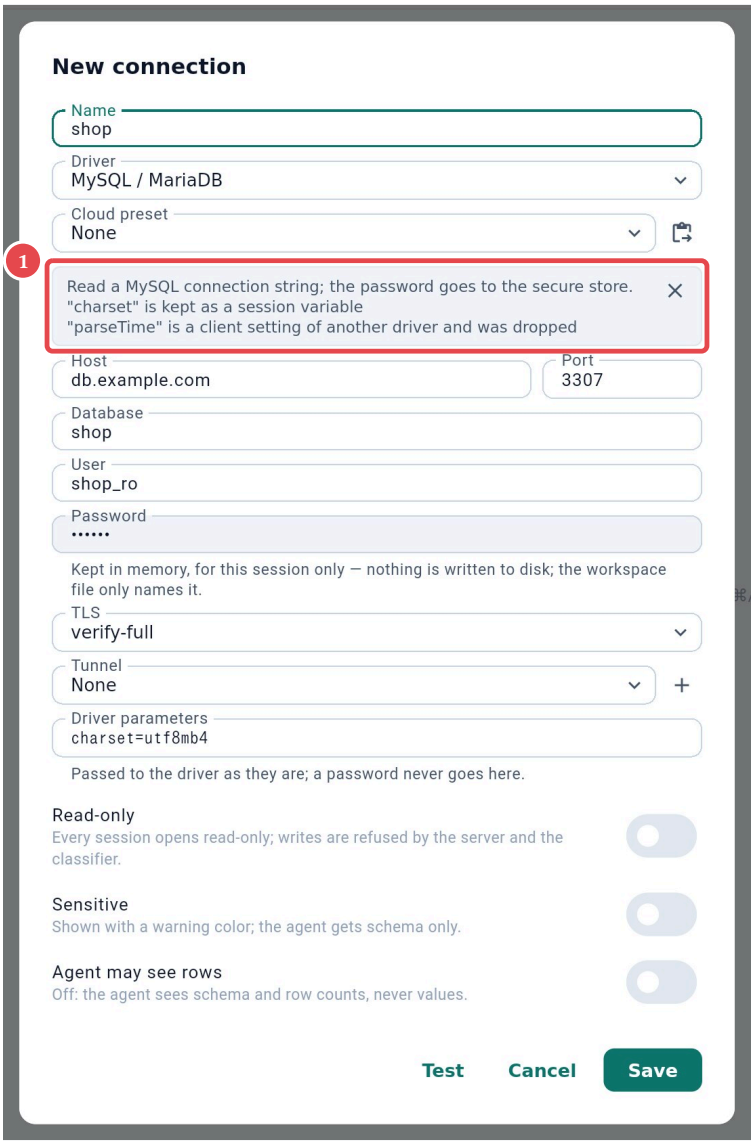


1. A cloud provider's settings
2. What the provider needs

Figure 4: A cloud preset

2.3 Pasting a connection string

If you have a connection string (`postgres://…`, `mysql://…`, `mongodb+srv://…`, a JDBC URL), click the clipboard button 2 in the first picture and paste it. Tabellum fills in the form, puts the password into the keychain and lists what it set and what it could not 1.

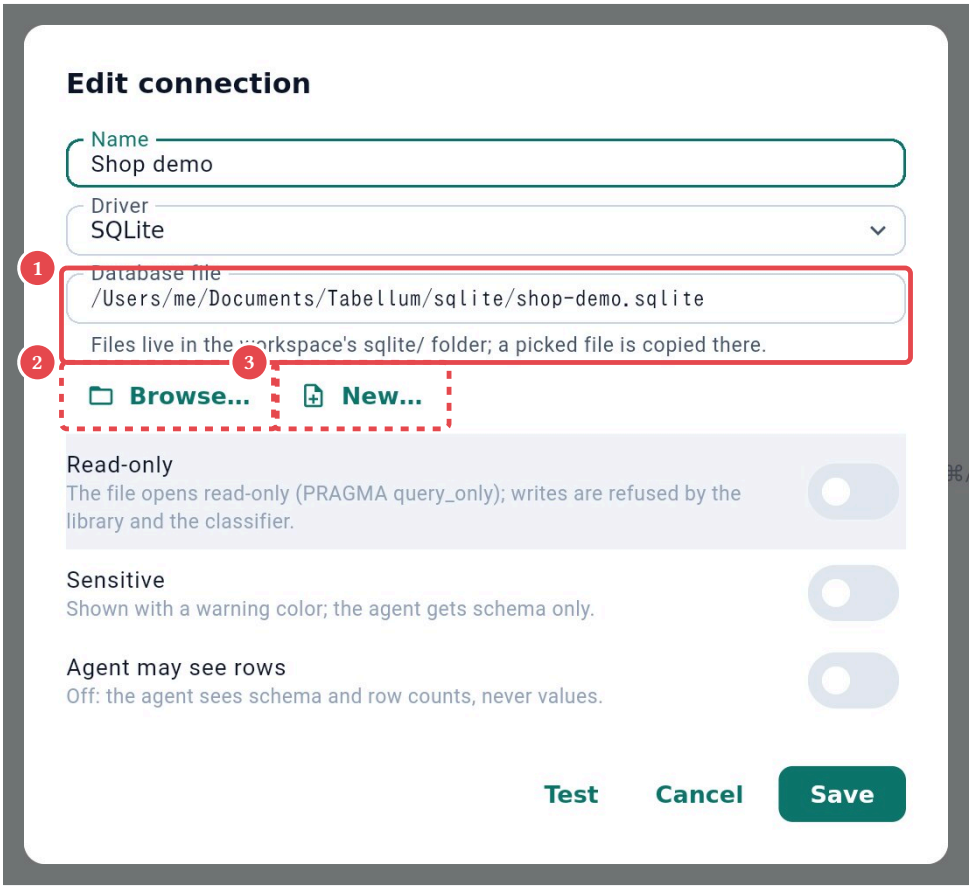


1 What the string set, and what it could not

Figure 5: The form filled from a pasted string

2.4 SQLite files

An SQLite database is a file, so its connection has no host or login. **Browse…** 2 picks an existing file and **New…** 3 creates an empty one. On the Mac App Store version a picked file is copied into the workspace's `sqlite/` folder, because the app may only keep access to its own folder.



1. The database file
2. Pick an existing file
3. Create a new database file

Figure 6: An SQLite connection

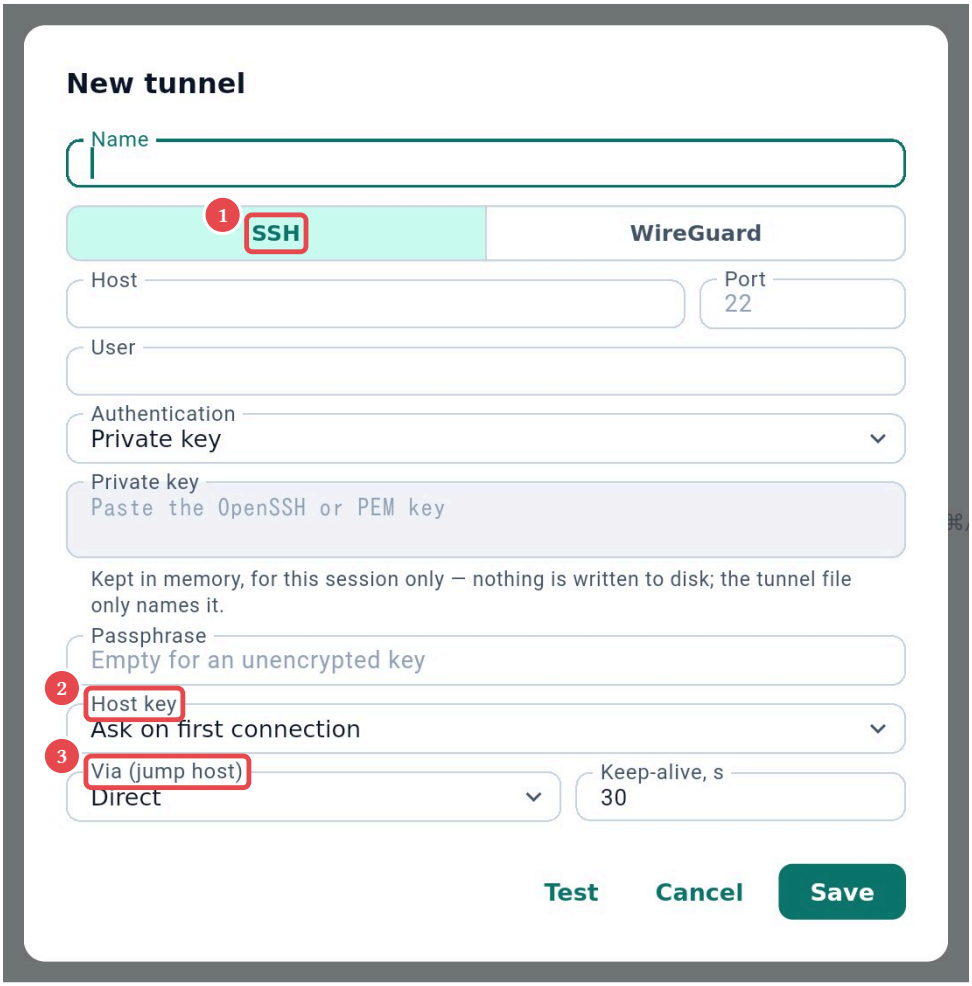
2.5 Editing a connection

Each connection in the sidebar has a pencil button, **Edit**, that opens the same form. The other buttons on the row refresh the schema and connect or disconnect. A connection that cannot connect shows one line saying what failed and what to check — a refused port, a timeout, a wrong password or a TLS problem. **Copy details** copies the full message.

After the computer sleeps, Tabellum reconnects the open connections by itself. A read that failed because the connection dropped is retried once; anything else offers **Run again**.

2.6 SSH and WireGuard tunnels

When a database is only reachable through a server or a VPN, add a tunnel and choose it in the connection's **Tunnel** field.



1. SSH, or a WireGuard peer
2. How the server's key is checked
3. Reach it through another tunnel

Figure 7: A new tunnel

- **SSH** (1) logs in with a private key (and its passphrase) or a password. **WireGuard** connects as a peer; paste a `wg-quick` configuration and the fields are filled in. Neither needs a system VPN or administrator rights.
- **Host key** (2) decides how the server's key is checked. With **Ask on first connection**, Tabellum shows the key's fingerprint the first time and remembers it.
- **Via (jump host)** (3) reaches the server through another tunnel.

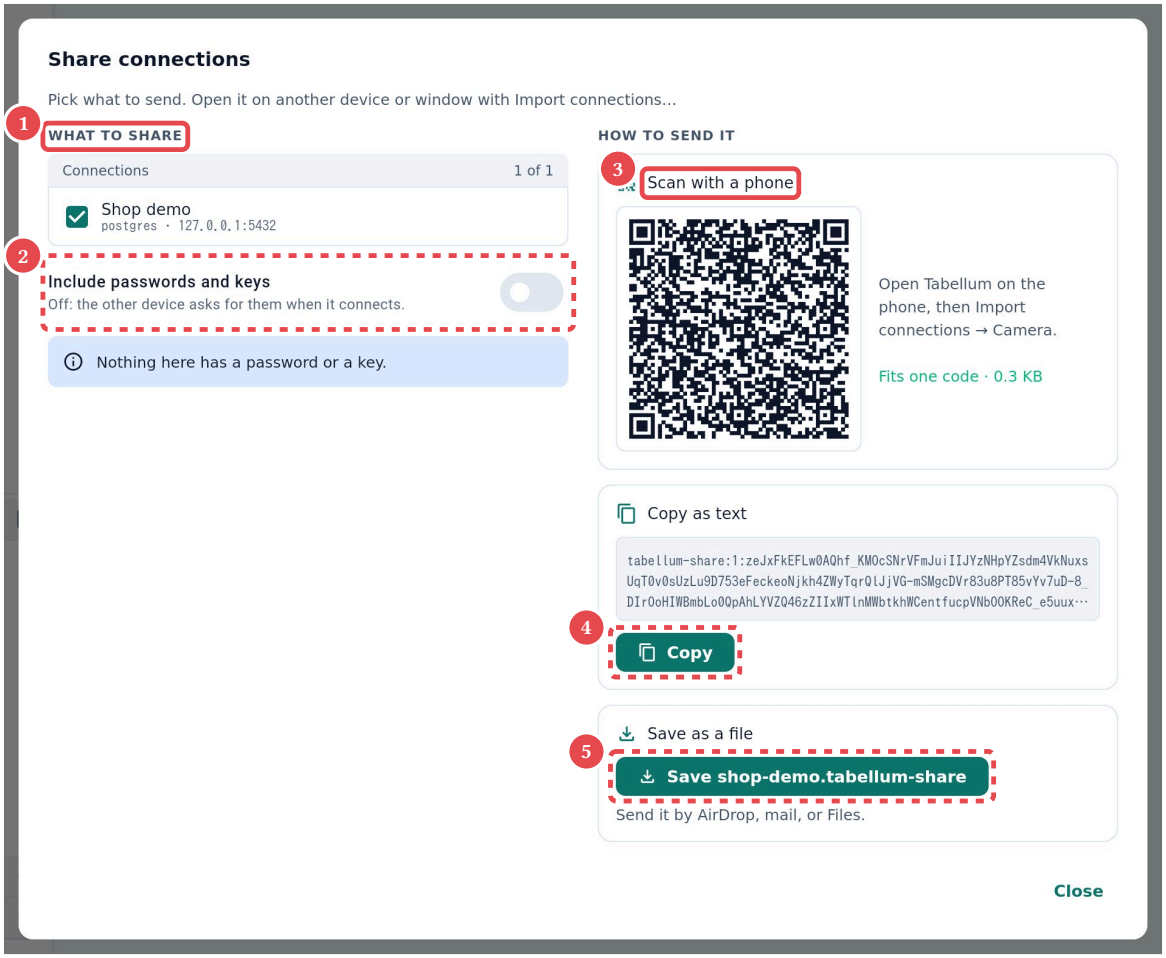


Figure 8: The first connection to an SSH server: compare the fingerprint, then trust it

> **Note**
> The Mac App Store version cannot use the SSH agent; paste the private key into the tunnel instead. It is kept in the keychain.

2.7 Sharing connections

Connections and tunnels move to another window or device with a code. Choose **Share connections…** in the Tabellum menu (⋯ at the foot of the sidebar, the File menu on a Mac) or the command palette, or **Share…** in a connection's menu or in **Tunnels…**.



1. Pick the connections; their tunnels come along
2. Passwords and keys, encrypted, only if you ask
3. A QR code for a phone's camera
4. The same code as text
5. Or saved as a file

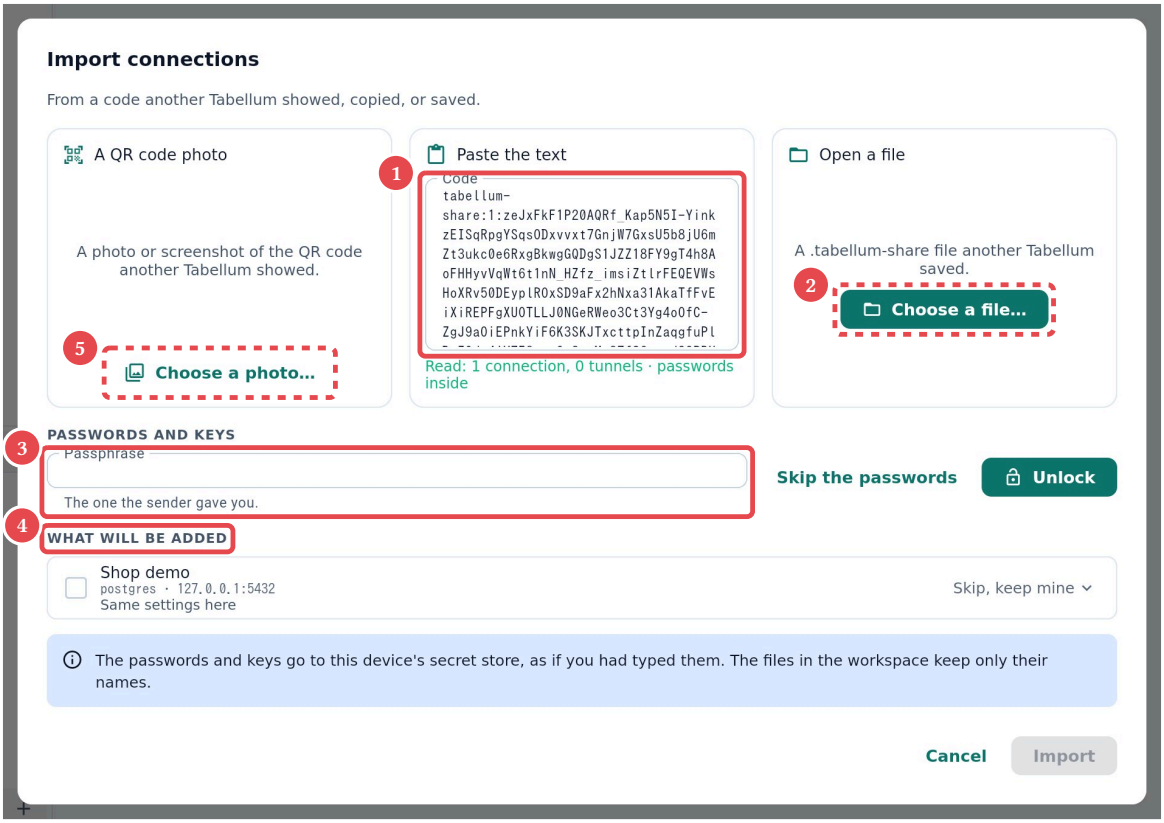
Figure 9: A connection as a QR code, a text, and a file

Tick what to send 1 a connection's tunnel comes along by itself. Then send it whichever way suits: scan the QR code 3 with a phone, copy it as text 4 into another window, or save it as a file 5 and send it by mail or AirDrop.

Passwords and keys stay on this computer unless you turn on **Include passwords and keys** 2. Then you type a passphrase twice, at least 6 characters, and the code carries them encrypted with it.

> **Warning**
> Anyone who has the code and the passphrase can reach those databases. Send the passphrase another way than the code: say it, don't paste it next to the code.

On the other side, choose **Import connections...**.



1. Paste the code
2. Or open the file
5. Or a photo or screenshot of the QR code
3. The passphrase, when the code carries passwords
4. What is added, and what to do with a name taken here

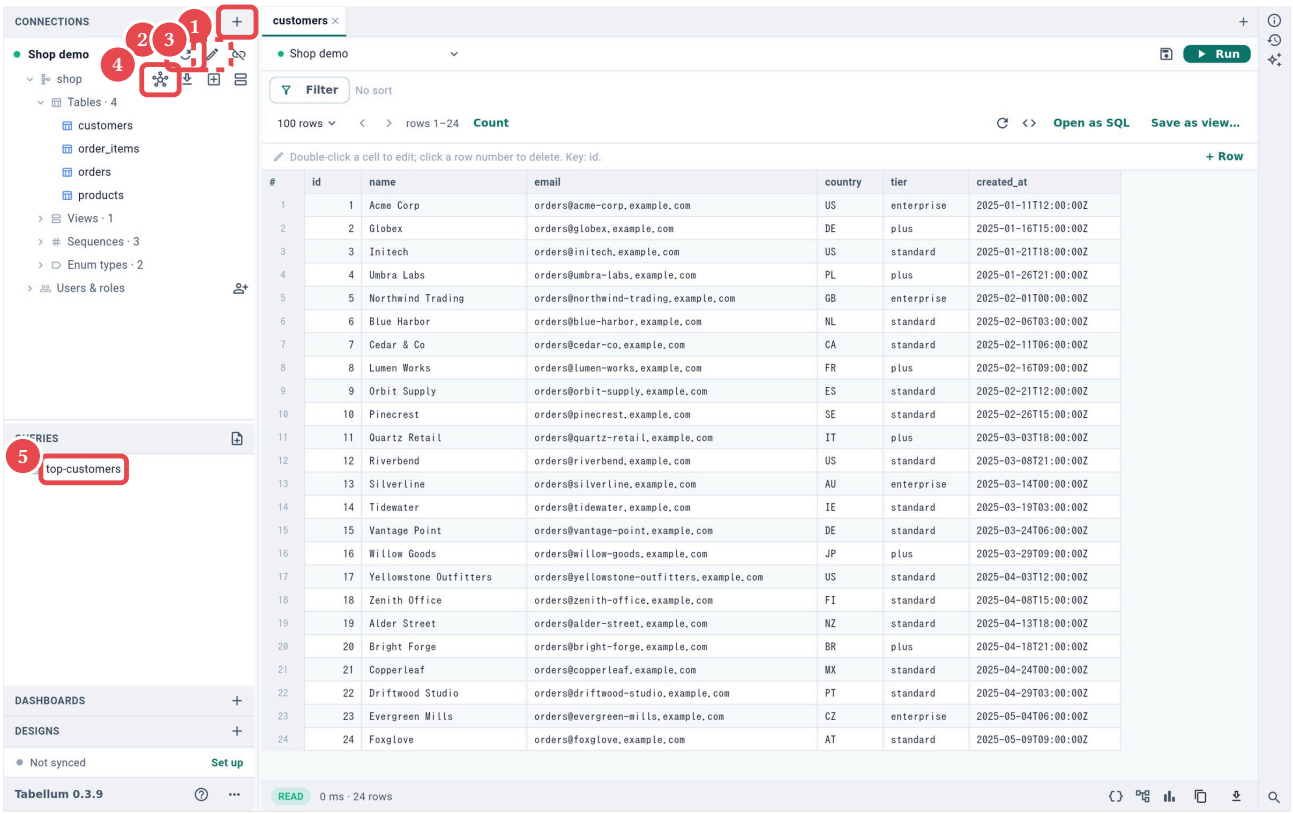
Figure 10: Importing a code that carries passwords

Paste the code 1 or open the file 2. The first panel reads the QR code: hold it up to the camera on a Mac, or choose a photo or screenshot of it 5. If the code carries passwords, type the passphrase 3 and choose **Unlock**, or **Skip the passwords** to type them later. The list 4 shows what will be added. When a connection or tunnel of that name is already here, choose **Import as a copy**, **Replace mine** or **Skip, keep mine** nothing is written until **Import**.

3 Browsing and editing data

3.1 The schema tree

Open a connection in the sidebar to see its schemas. Each schema groups its tables, views, functions, sequences and enum types; **Users & roles** lists the logins.



1. Add a connection
2. Read the schema again
3. Edit the connection
4. The schema as a diagram
5. Saved queries

Figure 11: The sidebar

- 1 adds a connection, 2 reads the schema again after someone changed it, 3 edits the connection.
- 4 draws the schema as a diagram (see Section 6.1).
- Saved queries 5 are files in `queries/`; a click opens one in a tab.

A right-click (or a long press) on an object opens its menu: export, import into a table, design, edit, and more.

3.2 A table's rows

A click on a table opens its rows in a tab, 100 at a time. Use the arrows to page, and click a column header to sort by it. **Filter** adds a condition; the value is passed to the database as a parameter, never pasted into the SQL.

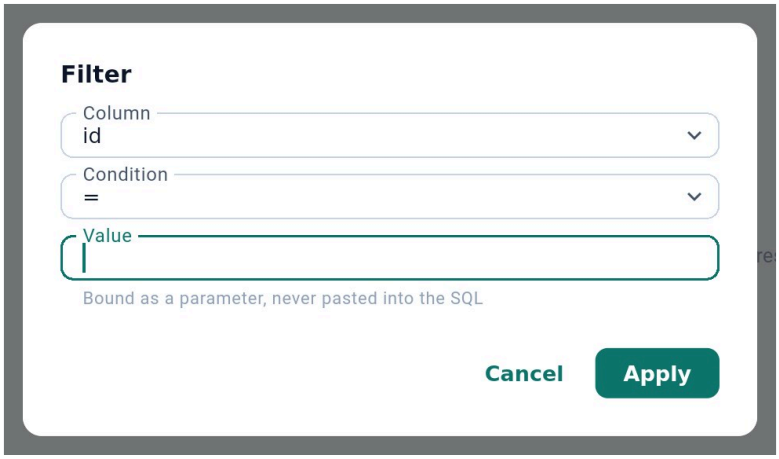
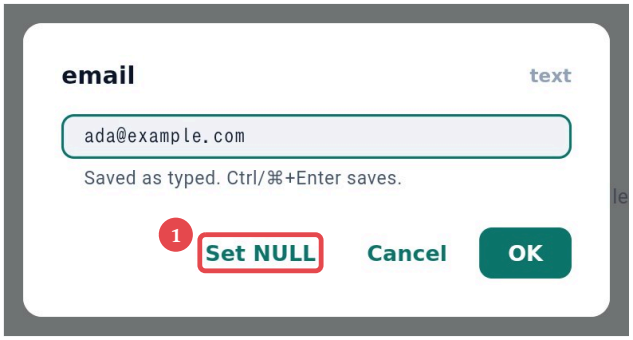


Figure 12: A filter on one column

The status bar under the rows shows how long the read took and how many rows came back. Its buttons switch the result between a table, JSON and a chart, copy it, and export it as CSV, JSON, SQL `INSERT` statements or an Excel workbook. **Export all matching rows…** exports every row the filter matches, not only the page on screen.

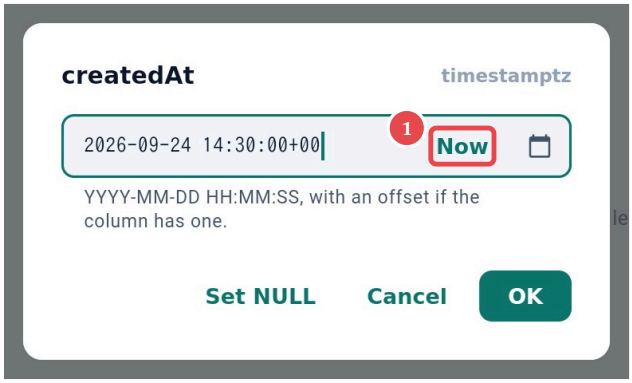
3.3 Editing cells

Double-click a cell to edit it. The editor fits the column's type: a date and time picker with **Now** (a time column takes only the time of day), a list for an enum, a check box for a boolean, a JSON editor, numbers and UUIDs checked as you type. **Set NULL** (1) stores NULL.



1 Store NULL instead of a value

Figure 13: Text



1 The current time

Figure 14: Date and time

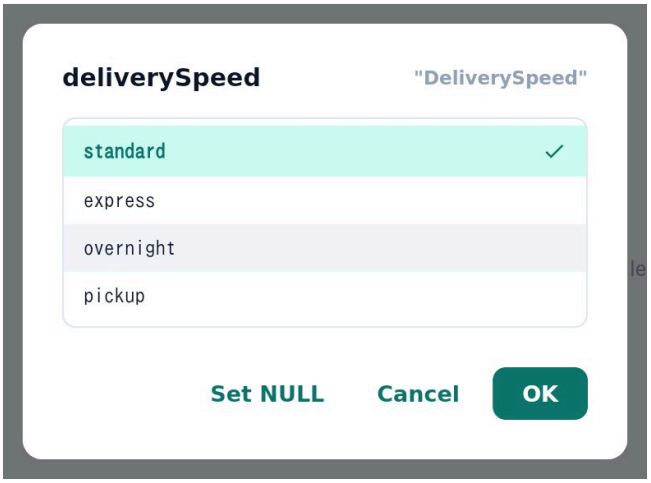


Figure 15: An enum column offers its values

Click a row number to mark the row for deletion, or **+ Row** to add one. Nothing is written until you save: the edits become SQL statements, and they run through the approval dialog (see Section 4.6) so you see exactly what changes.

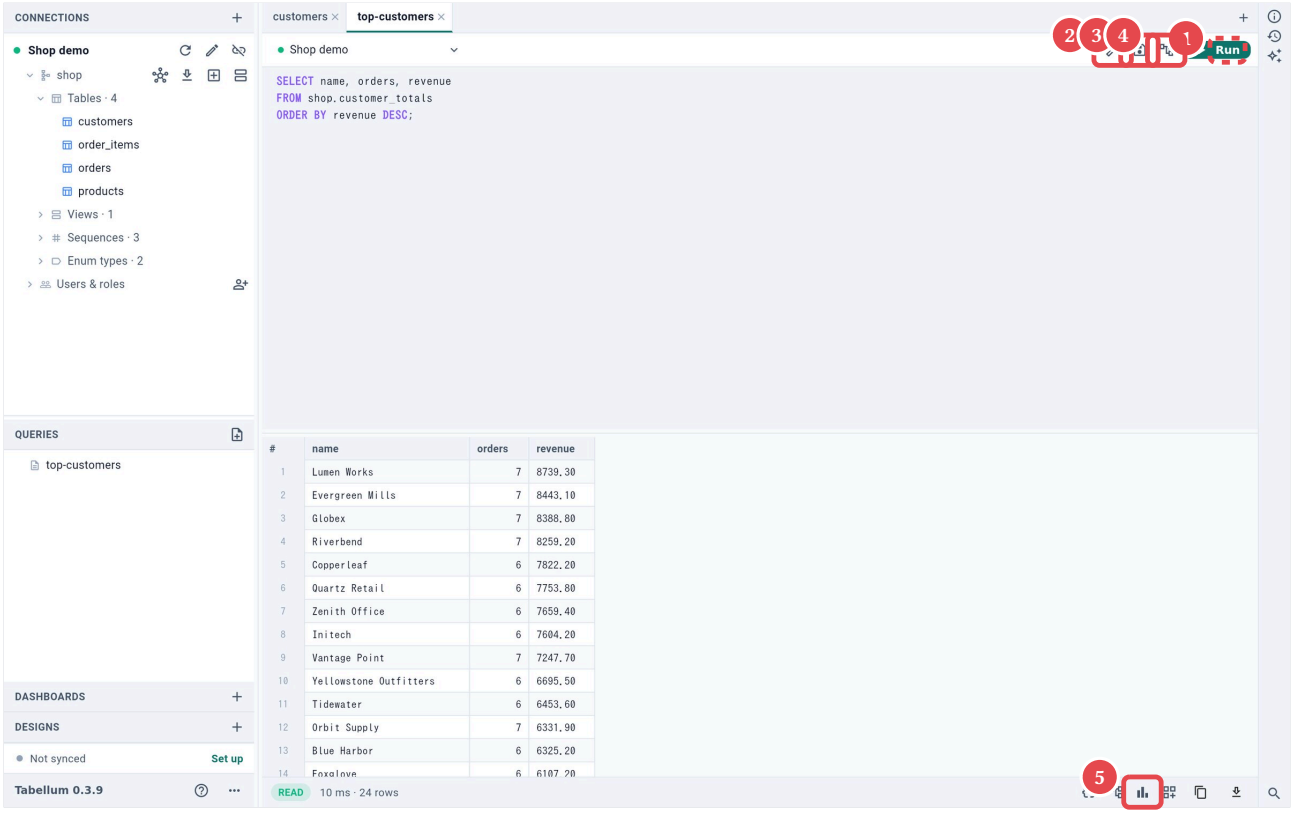
3.4 The inspector

With a table selected, the inspector on the right lists its columns with their types, its indexes and its constraints. **Copy DDL** copies the `CREATE` statement; **Browse rows**, **Design table** and **Edit definition** open the object in a tab.

4 Writing SQL

4.1 The query tab

New query tab (the `+` above the tabs, or the palette) opens an editor on the connection chosen at its top. Type SQL and press `Cmd+Enter` or click **Run** (1): Tabellum runs the selection if there is one, otherwise the statement under the caret.



1. Run the statement at the caret, or the selection
2. Format the SQL
3. Save as a query file
4. The query plan
5. Table, JSON or chart

Figure 16: A saved query and its rows

- (2) formats the SQL: keywords upper-cased, one clause per line.
- (3) saves the tab as a file in `queries/` (`Cmd+S`).
- (4) shows the query plan (see below).
- (5) switches the result between a table, JSON and a chart.

While a query runs, **Run** becomes a red **Stop** button that counts the seconds; `Cmd+.` or `Esc` stops it too.

4.2 Completion

The editor completes as you type, and it knows both the database's dialect and your schema: tables, columns of the tables the statement names (with their aliases), functions with their arguments, and whole join conditions along foreign keys.

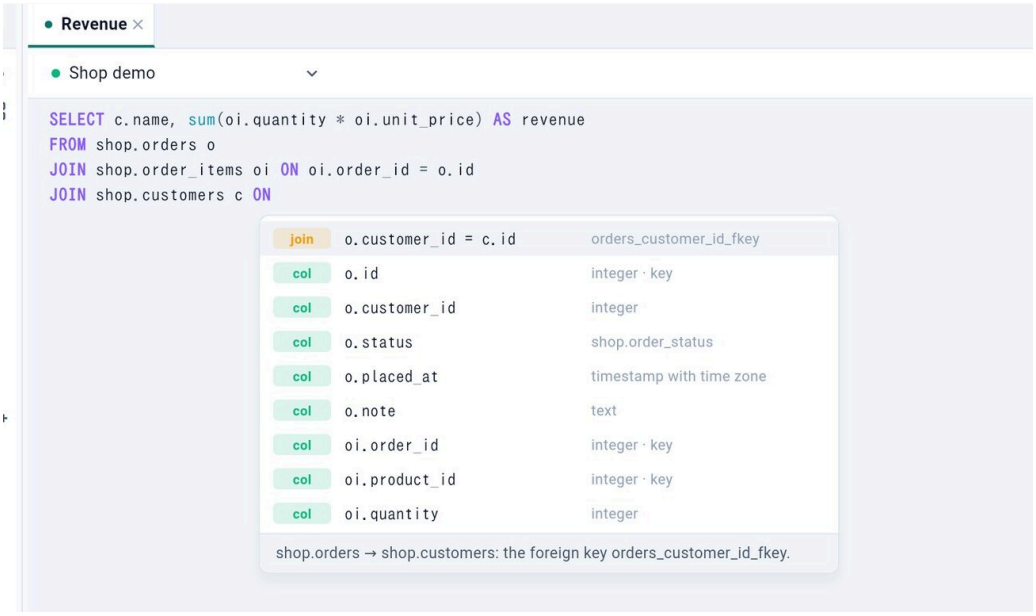


Figure 17: After `ON`, the joins the foreign keys allow come first

Press `Tab` or `Enter` to accept, the arrow keys to move, `Esc` to close.

4.3 Parameters

A query can have named parameters — `:since`, `$1` or `@name`, as the database writes them. Before it runs, Tabellum asks for their values and passes them to the database separately, so they are never pasted into the SQL.

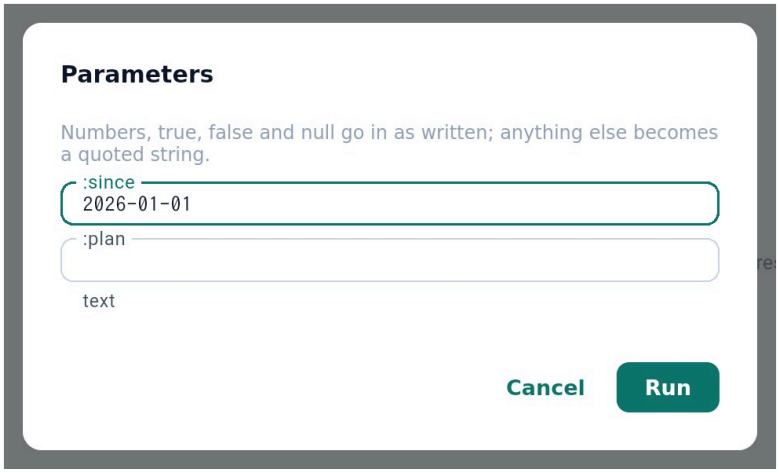


Figure 18: Values for a query's parameters

4.4 The query plan

Plan (`Cmd+Shift+E`) asks the database how it would run the statement and shows the answer as a tree. Each step has a bar for its share of the cost, the costliest steps are highlighted, and the database's own output is a click away. Press it again to go back to the rows without running anything.



Figure 19: A plan: each step with its rows and share of the time

4.5 The command palette

`Cmd+P` opens the palette: type a few letters of a connection, table, saved query or command and press `Enter`.

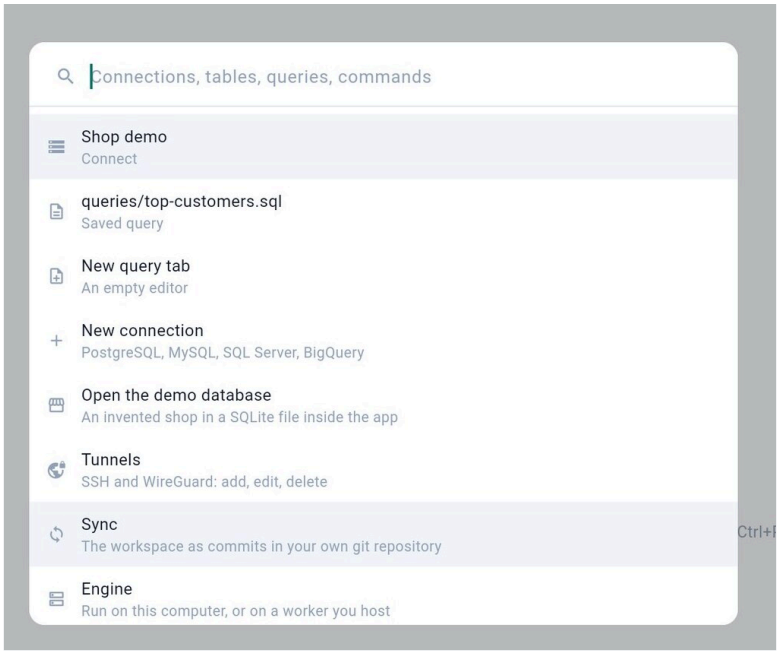
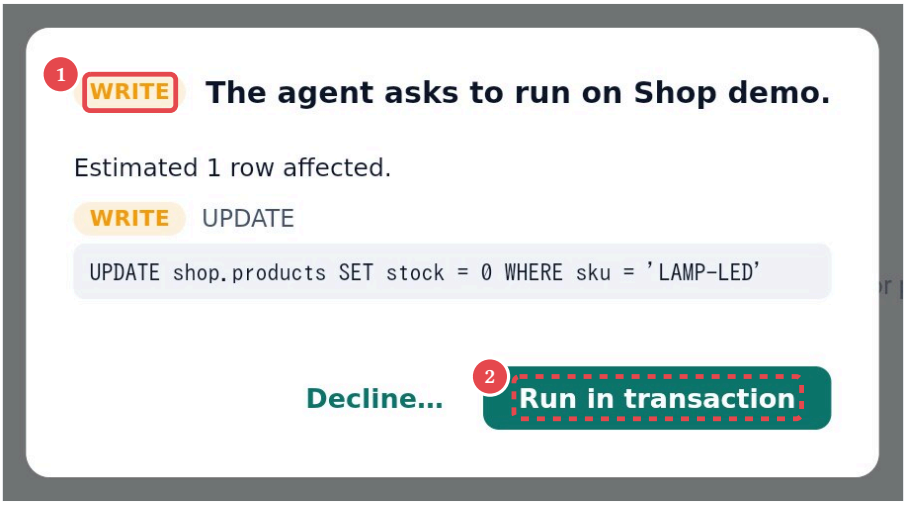


Figure 20: The command palette

4.6 The approval dialog

Tabellum reads every statement before it runs. Anything it cannot prove to be a read — `UPDATE`, `DELETE`, `INSERT`, `CREATE`, `ALTER`, `DROP` — stops at an approval dialog that shows the statements, their kind 1 and, where the database can tell, how many rows they would change.



1. What kind of statement it is
2. Run it, all or nothing

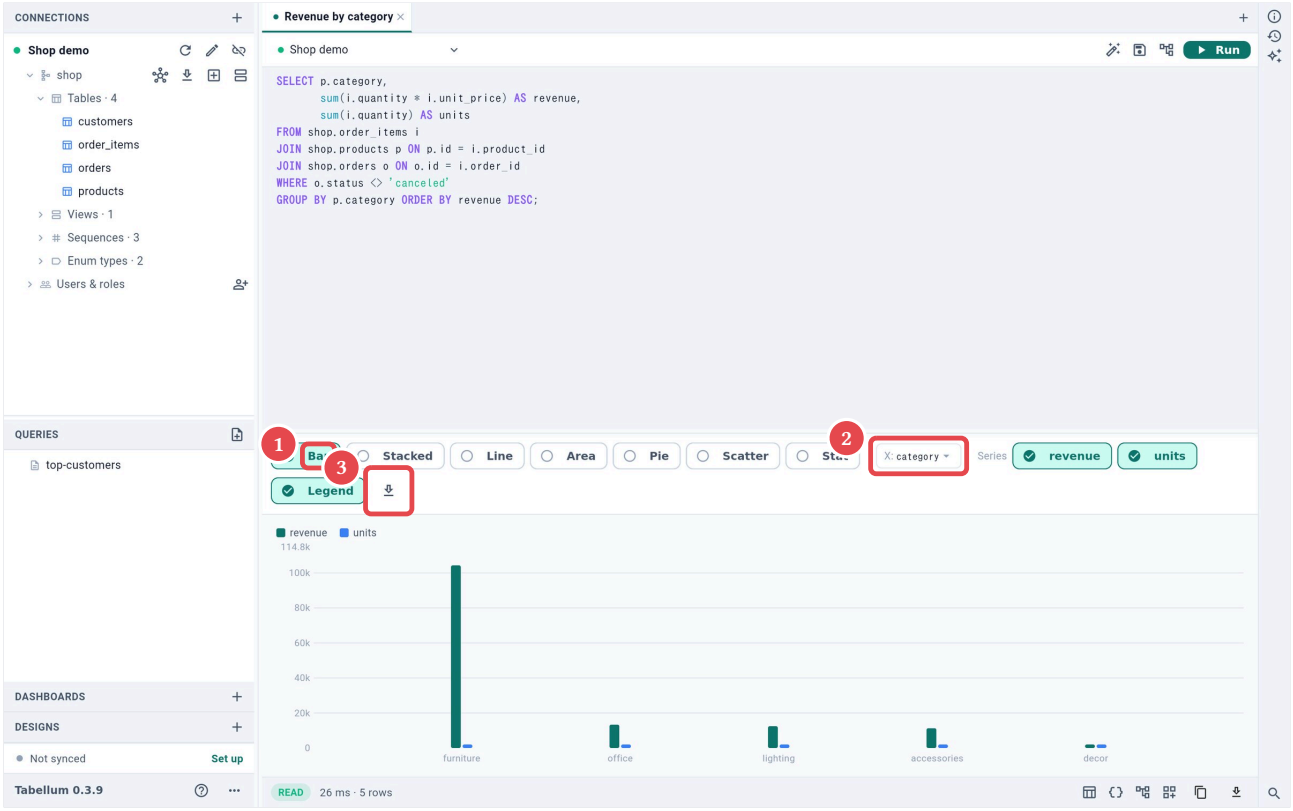
Figure 21: A write waits for your click

Run in transaction 2 runs them all or none. Before a statement that changes the schema, the dialog offers to back up first. This dialog is the same whether you typed the SQL, edited cells, applied a design or the AI agent proposed it.

5 Charts, dashboards and export

5.1 Charts

Any result can be a chart: click the chart button in the status bar. Tabellum suggests one from the columns; change the kind 1, the column along x 2 and the series to taste. 3 saves the chart as PNG or SVG.



1. The kind of chart
2. The column along x
3. Save as PNG or SVG

Figure 22: Revenue and units by category

The kinds are bar, stacked bar, line, area, pie, scatter and stat (one big number). A line or area over dates or numbers is drawn in their order, not the rows' order.

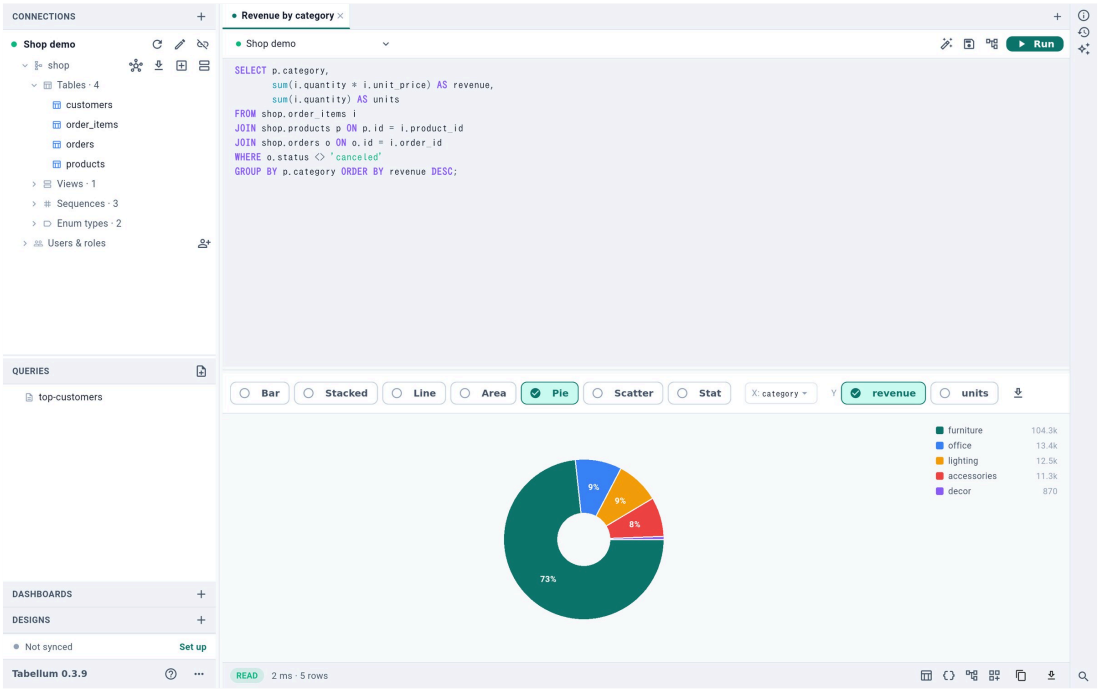
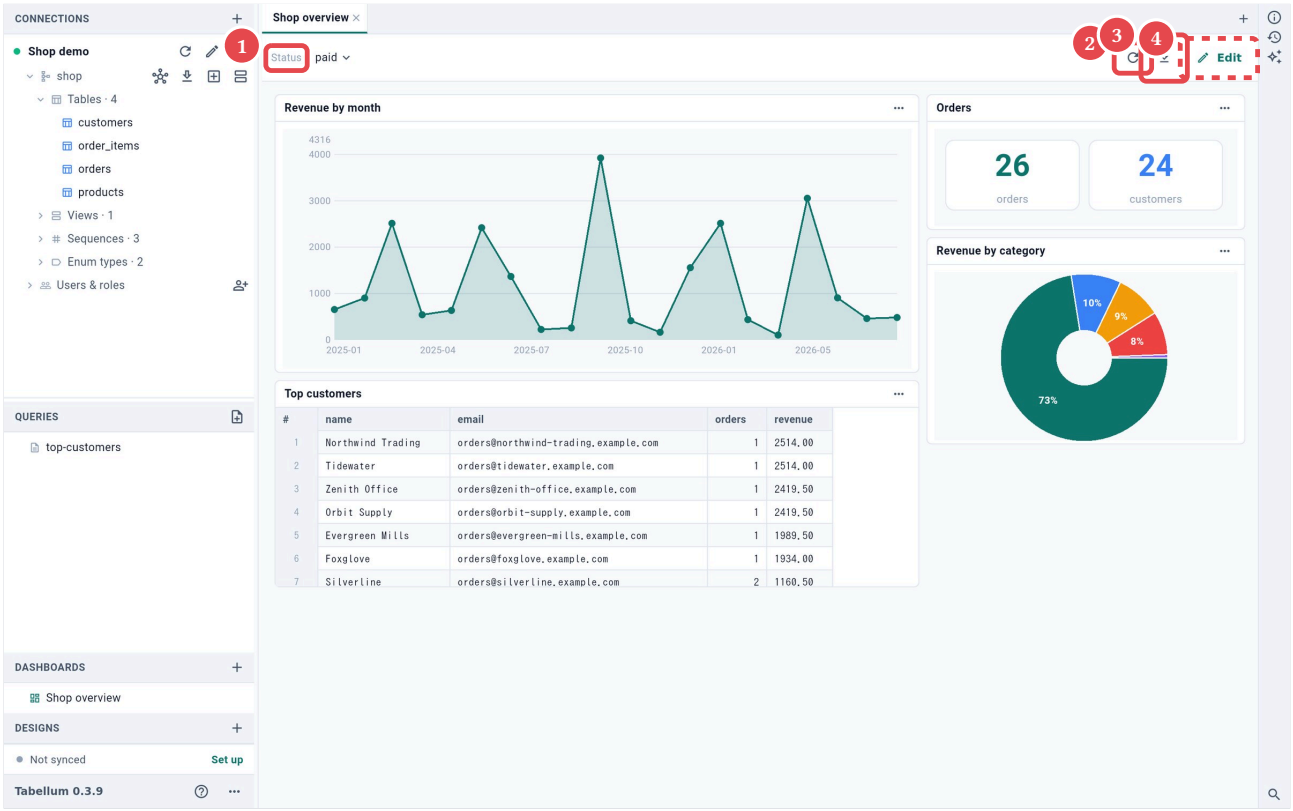


Figure 23: The same result as a pie

5.2 Dashboards

A dashboard is a page of panels — charts and tables — each with its own query, on any connection. It lives as a file in `dashboards/`.



1. A parameter every panel uses
2. Refresh every panel
3. Export as PNG or PDF
4. Edit panels and parameters

Figure 24: A dashboard with a parameter and four panels

- Parameters 1 are shared by every panel: change one and they all run again.
- 2 refreshes every panel; a panel can also rerun on its own timer while the dashboard is open.
- 3 exports the dashboard as a file, a PNG or a PDF.
- 4 edits the panels: their queries, their charts and their place on the grid.

To add a result you are looking at, use **Add to dashboard…** in its status bar.

> **Note**
> Dashboards only read. A panel's query is checked like any other, and a write is refused.

5.3 Exporting results

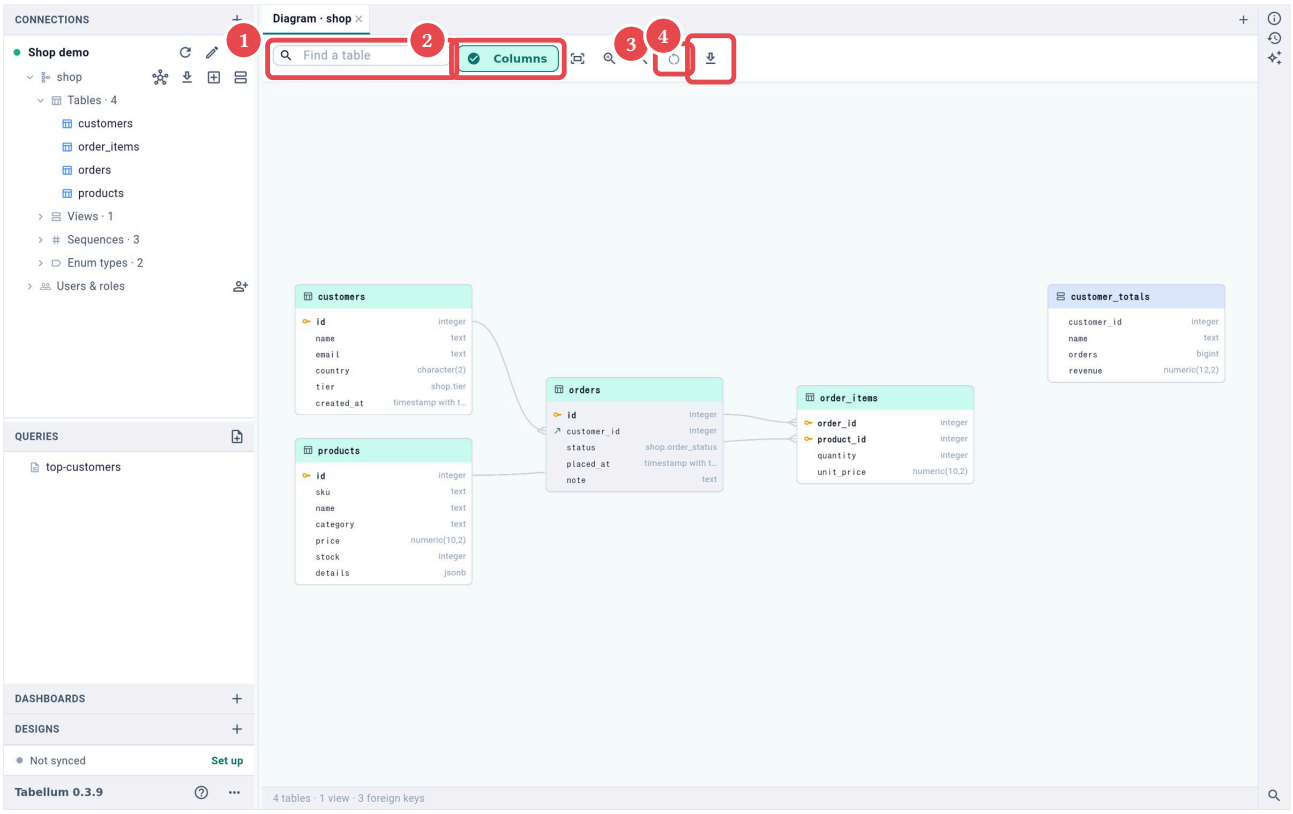
The **Export** button in the status bar saves a result as CSV, JSON, SQL `INSERT` statements or an Excel workbook (a large result is split across sheets). The **Copy** menu copies a cell, a row as JSON, a column, the column names, or the whole result as CSV, JSON or SQL.

To export a whole table, schema or database, use **Export table…**, the schema's export button or **Export database…** in the sidebar; **Import into table…** loads a CSV or JSON Lines file into a table, in one transaction.

6 Diagrams, snapshots and comparing schemas

6.1 Schema diagrams

The **Diagram** button next to a schema in the sidebar draws its tables and the foreign keys between them.



1. Find a table
2. Columns or names only
3. Put moved tables back
4. Save as SVG or PNG

Figure 25: The demo shop as a diagram

- 1 finds a table by name and centers it.
- 2 switches between full columns and table names only — the better view for a large schema.
- Drag a table by its card to move it; Tabellum remembers where you put it. 3 puts the moved tables back.
- 4 saves the diagram as SVG, or PNG while it is small enough.

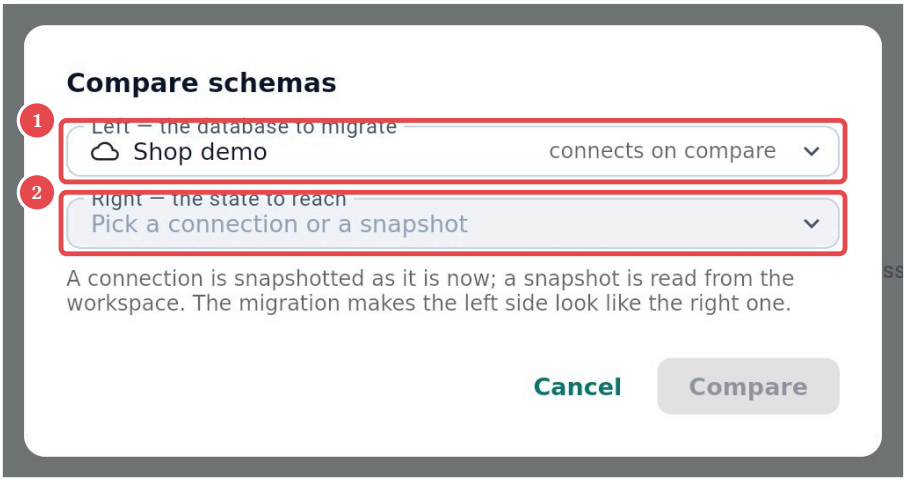
Scroll or drag the empty canvas to pan, pinch or use the zoom buttons to zoom. A click on a table selects it, lights up its foreign keys and opens the inspector; a click on empty canvas clears the selection.

6.2 Snapshots

A snapshot is a schema's structure — tables, columns, keys, indexes, views, functions — saved as a file in `snapshots/`, without any rows. Take one with **Save schema snapshot** in the palette or the connection's menu — before a risky change, for example — then compare against it later, or bring a database back to it.

6.3 Comparing two schemas

Compare schemas (in the palette or a connection's menu) compares two sides: the one to migrate 1 and the one to reach 2. Each side is a connection as it is now or a saved snapshot.

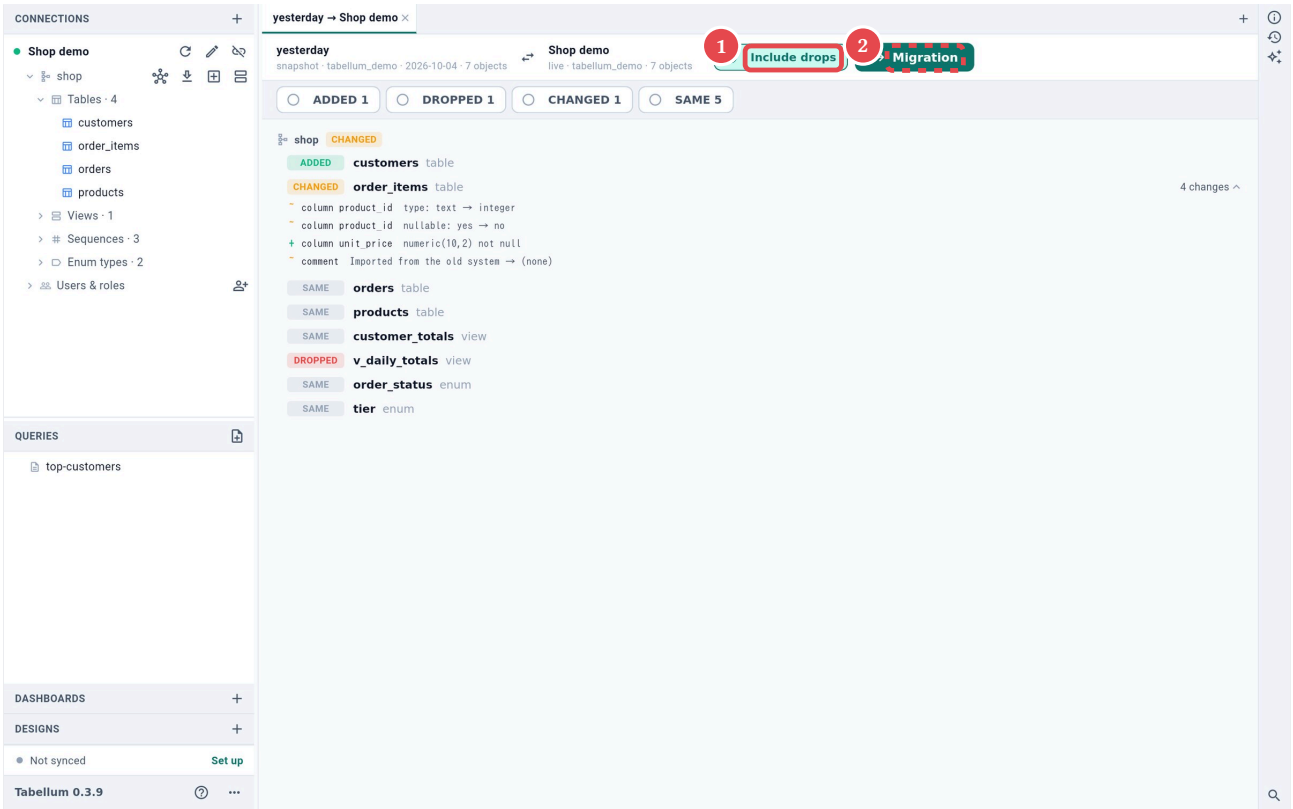


1 The database to migrate

2 The state to reach

Figure 26: Choosing the two sides

The result lists every object as added, dropped, changed or the same, with what changed in each.



1 Also drop what the target lacks

2 The migration as SQL

Figure 27: What differs between yesterday's snapshot and the database

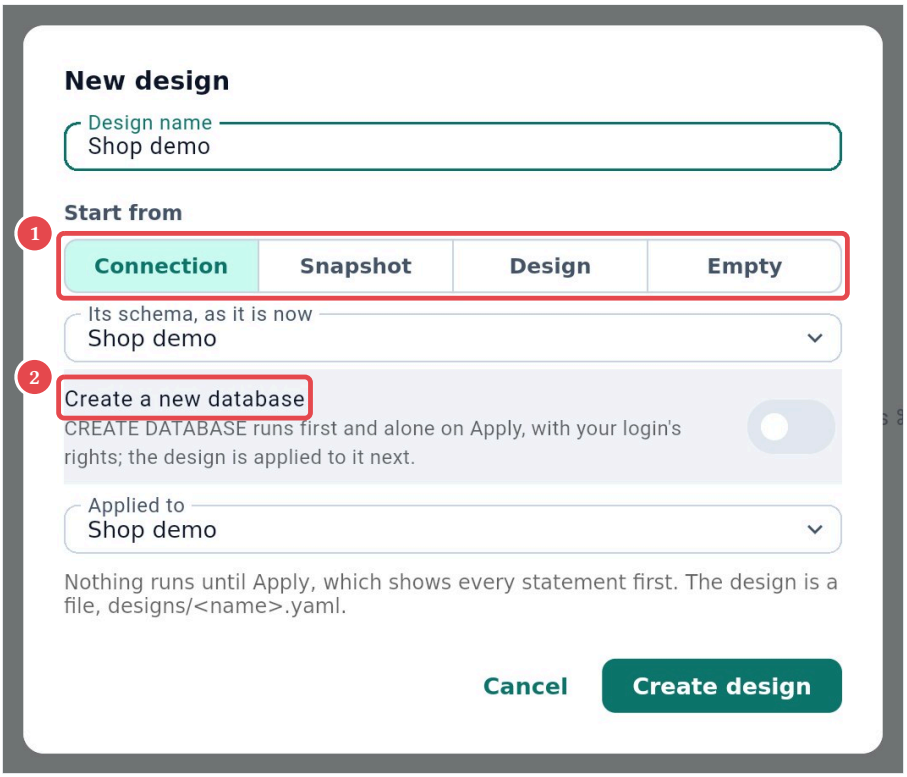
Migration 2 opens the SQL that makes the left side match the right in a new tab. It runs only when you run it, through the approval dialog. Drops are left out unless **Include drops** 1 is on.

7 The schema designer

The designer is a canvas for changing a schema the way you would draw it: add tables, drag foreign keys between columns, edit types and constraints. Nothing touches the database until you click **Apply**, and then you see every statement first. A design is a file in `designs/`.

7.1 Starting a design

Click **+** next to **DESIGNS** in the sidebar.



1. Where the design starts from
2. Create a new database instead

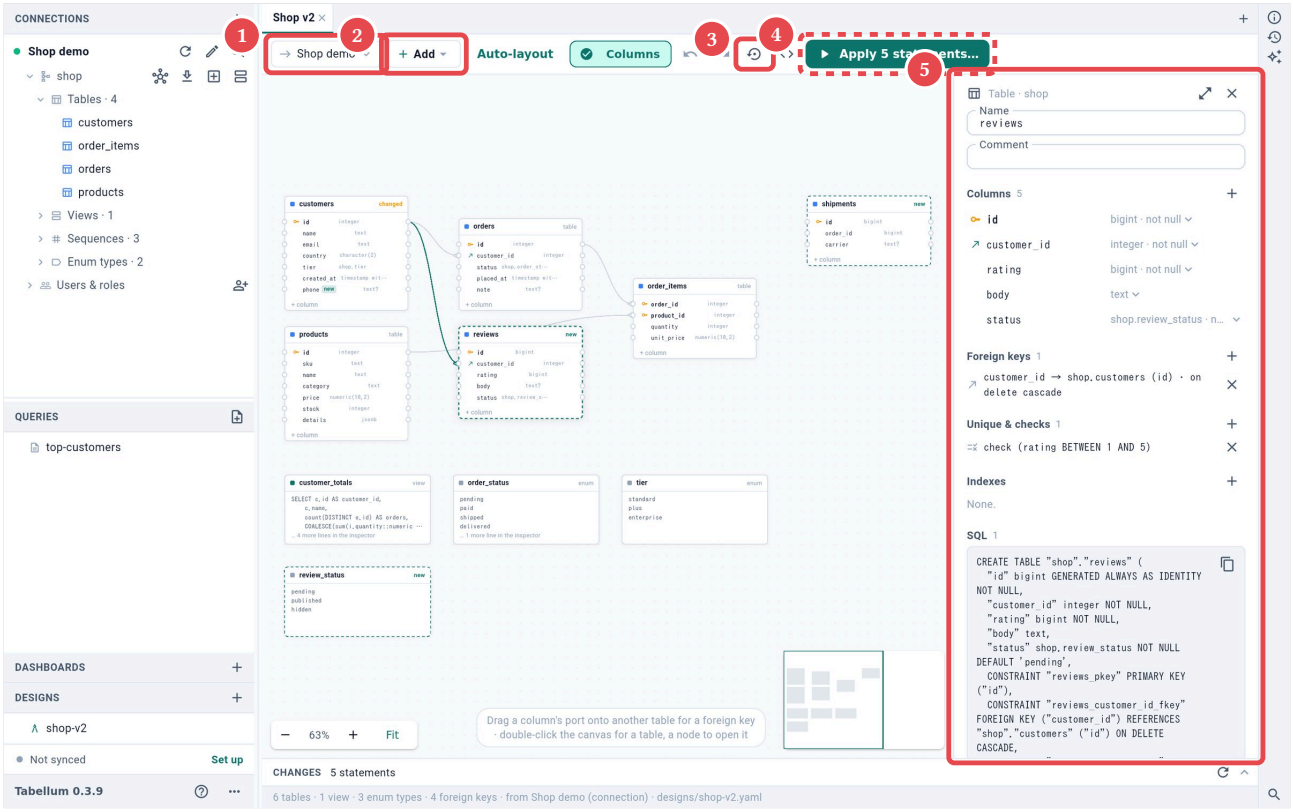
Figure 28: A new design

Under **Start from** 1, choose:

- **Connection** — the database's schemas as they are now. Applied back to the same database, nothing changes until you edit something.
- **Snapshot** — a saved snapshot, to build on it or bring a database back to it.
- **Design** — a copy of another design.
- **Empty** — a blank schema.

Create a new database 2 makes the design create its database first; you choose its name, the server it is created on and its options.

7.2 The canvas



1. The connection Apply runs on
2. Add a table, view, function…
3. Reload the design from the database
4. Apply the changes
5. The selected object

Figure 29: A design: the canvas, and the inspector for the selected table

- 1 is the connection **Apply** runs on.
- 2 adds a table, view, materialized view, function, procedure, sequence or enum type — what the database supports.
- Drag a table to move it. Drag from a column's port (the ring at its edge) onto another table's column to create a foreign key. Double-click empty canvas to add a table there, or a table to open it full-screen.
- The inspector 5 edits what you selected: name, comment, columns with their types and defaults, keys, foreign keys, checks and indexes, and shows the SQL for that object.
- `Cmd+Z` undoes and `Cmd+Shift+Z` redoes any change, including moves.
- **Auto-layout** arranges the canvas; **Columns** shows or hides the columns.

Tables marked **new** or **changed** differ from the database; a red dashed table is one the database has and the design does not, which **Apply** would drop.

7.3 Reviewing and applying

The **Changes** drawer under the canvas is always up to date: how many statements **Apply** would run, what it would drop, any warnings, and the statements themselves.

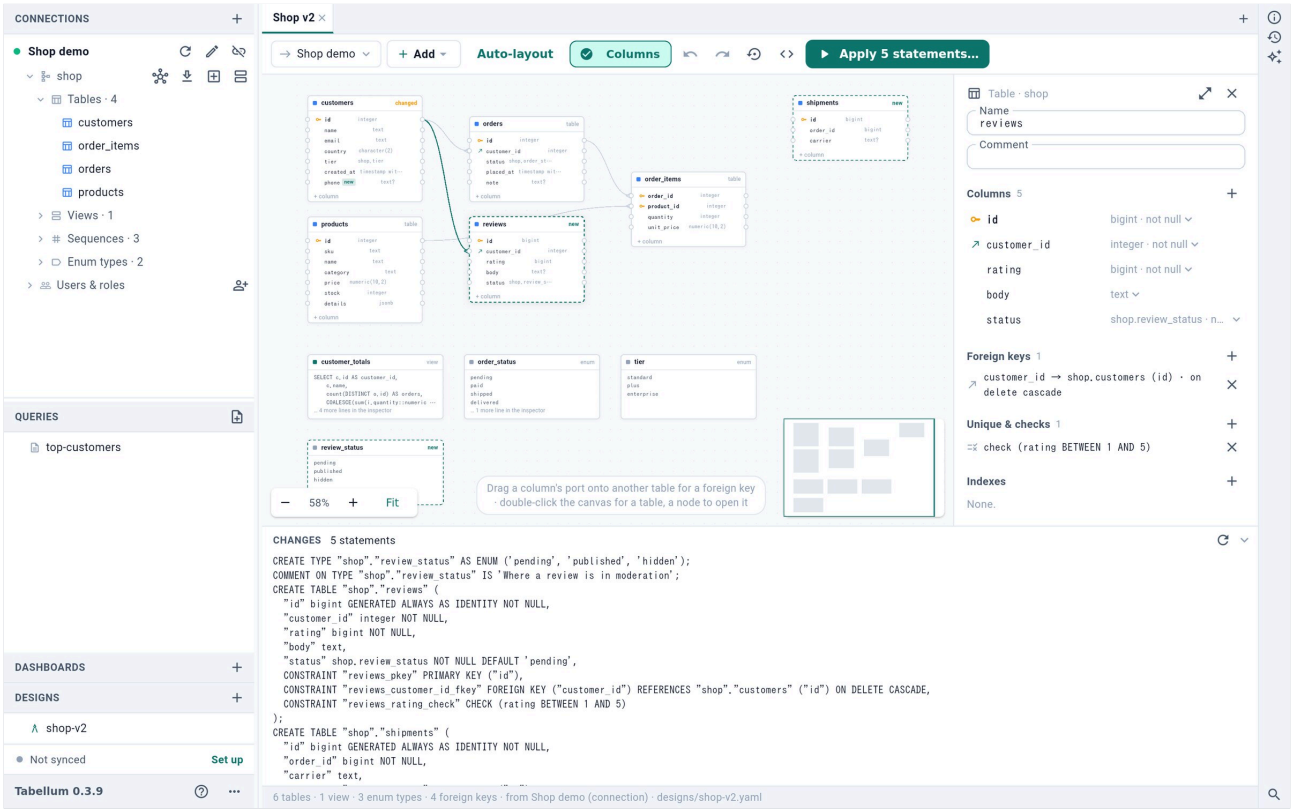


Figure 30: The Changes drawer, open

Click **Apply** 4 in the canvas picture. The statements open in the approval dialog and run in one transaction where the database allows it.

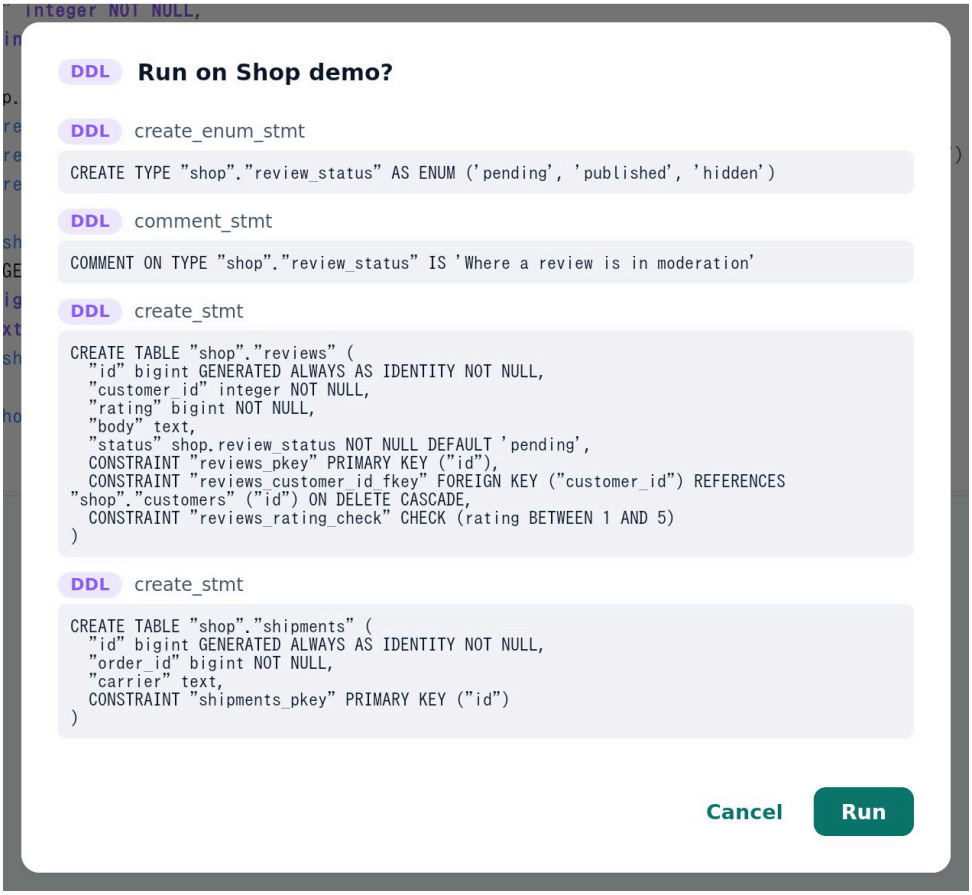


Figure 31: Applying a design

The code button next to **Apply** opens the same script in a query tab instead, to run it yourself or keep it.

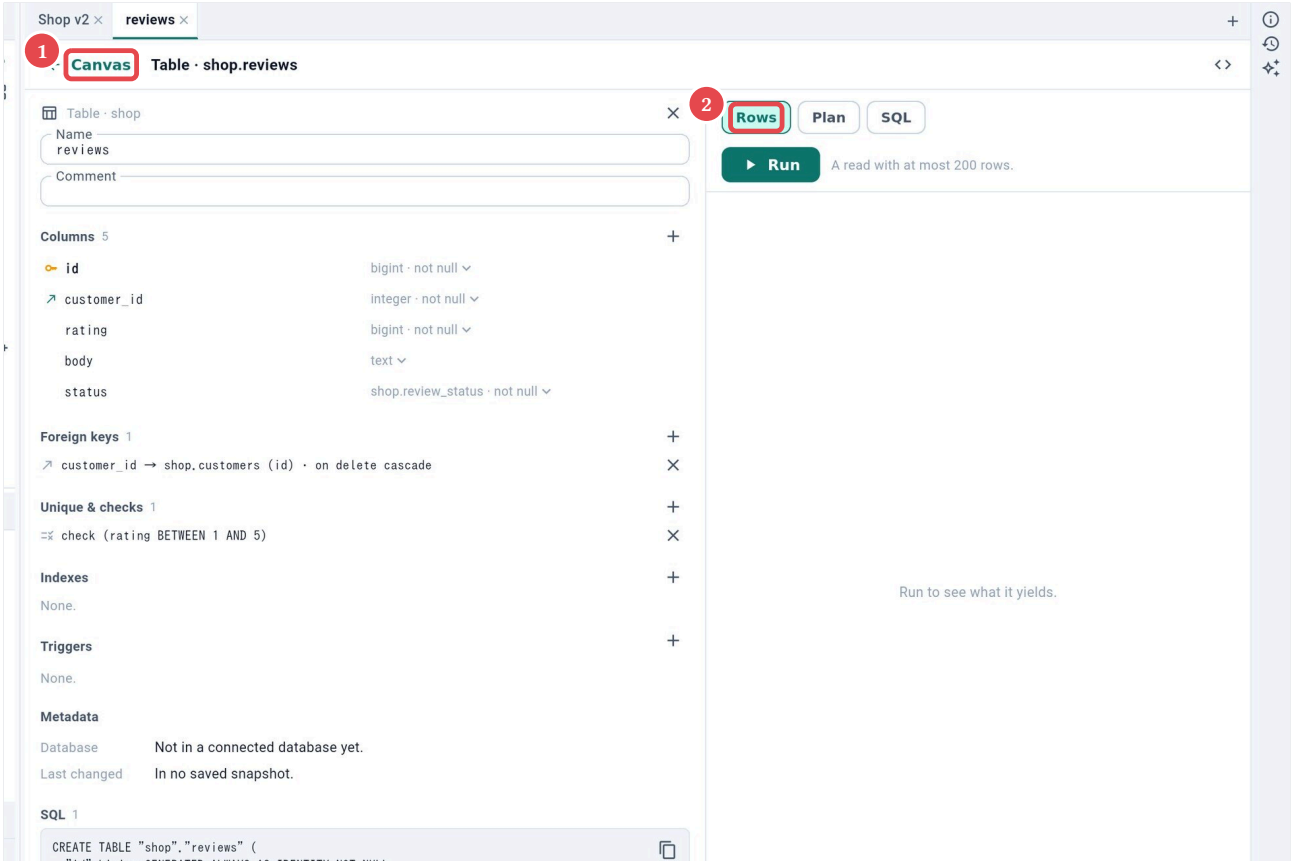
7.4 When the database has changed

If someone changed the database after you made the design — a migration, a new table — **Apply** would undo their change. Two buttons bring the design up to date:

- **Reload from the database** (3) reads the database again, lists what your design would lose, and replaces the design with the database's schema. Table positions are kept, and `Cmd+Z` brings your version back.
- **Take from the database**, in the inspector of a table marked **changed**, replaces that one table with the database's version and keeps the rest of your design.

7.5 Full-screen editors

Double-click an object (or use the arrow button in the inspector) to open it in its own tab. On the left is its definition, on the right a preview: sample rows, the query plan, or the SQL that would run.



1. Back to the canvas
2. Preview rows, plan or SQL

Figure 32: A table full-screen: definition and preview

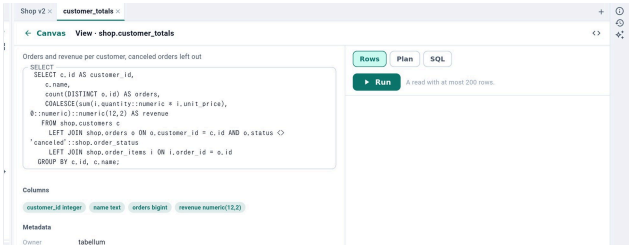


Figure 33: A view

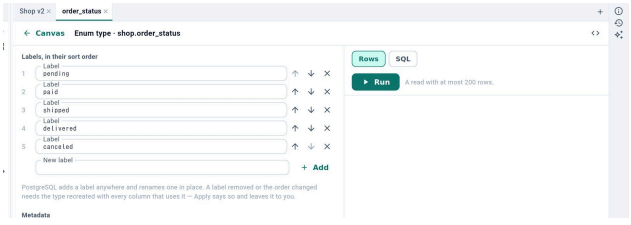


Figure 34: An enum type and its values

Canvas (1) goes back to the design. **Rows**, **Plan** and **SQL** (2) switch the preview.

7.6 Creating a database

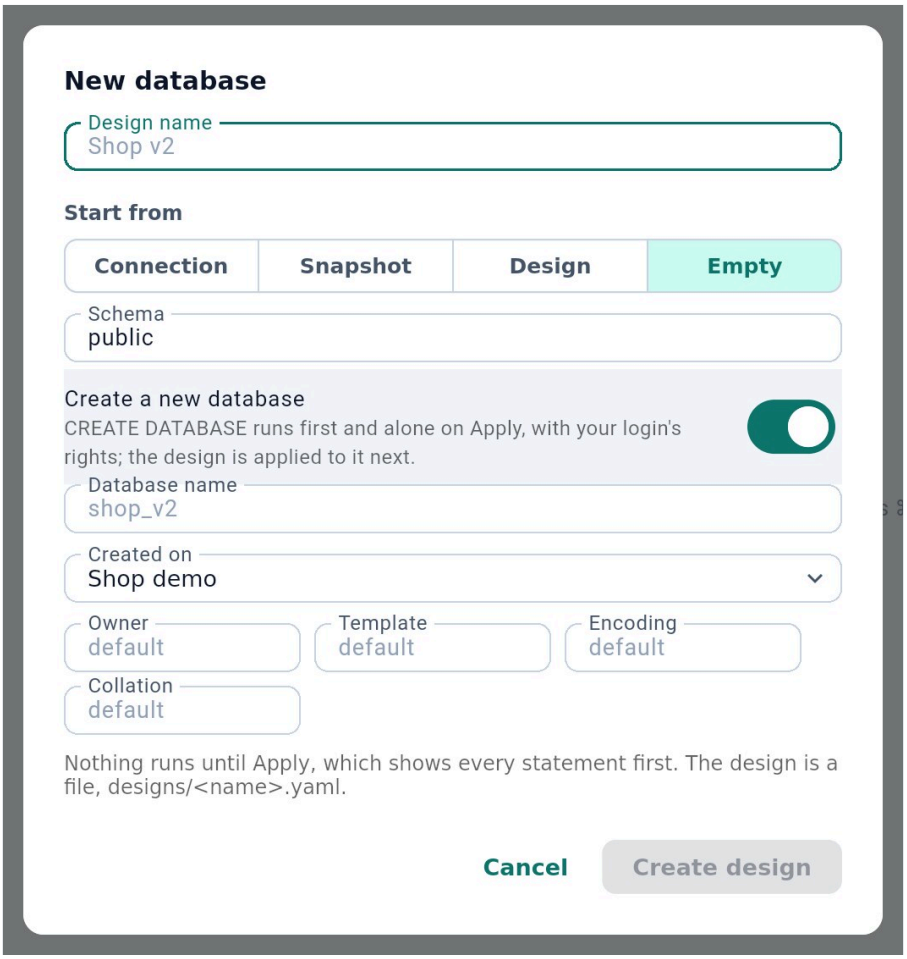


Figure 35: A design that creates its database

On PostgreSQL you choose the owner, template, encoding and collation; on MySQL the character set and collation; on SQL Server the owner and collation. Without a server, the new database is an SQLite file in the workspace.

8 Views, users and roles

8.1 Making a view without writing SQL

New view next to a schema in the sidebar (or **Save as view…** above a table's rows) opens the view designer: pick the source table, the columns to keep, conditions and a sort order, and whether the view is materialized. **Preview SQL** shows the `CREATE VIEW` statement, which runs through the approval dialog.

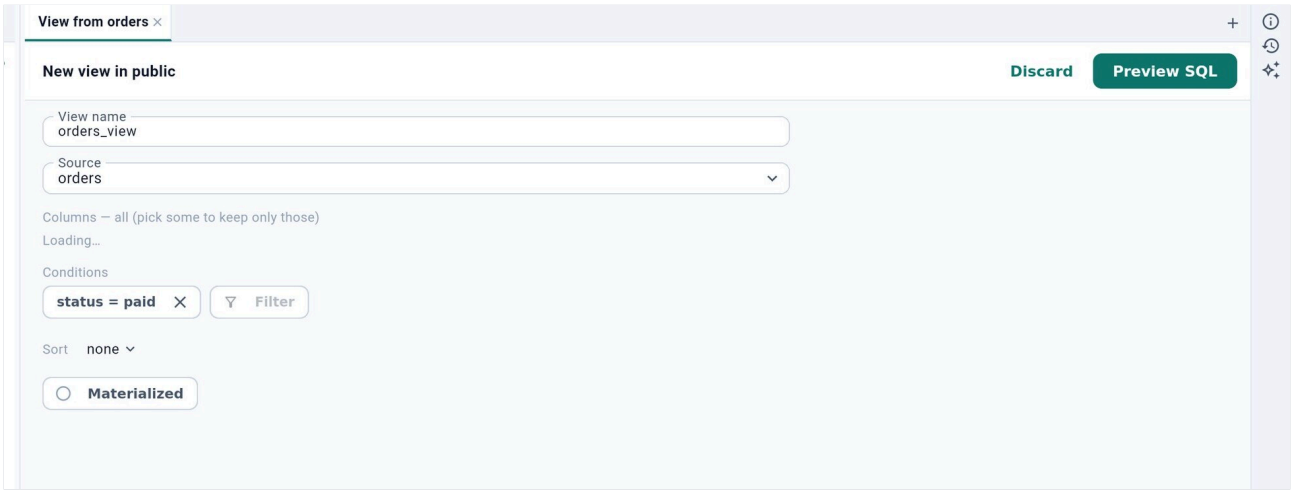


Figure 36: A view of the paid orders

8.2 Users and roles

Users & roles under a connection lists the logins and roles the connected user can see; a superuser is drawn in the warning color. **New user** creates one, a click on an existing one opens it.

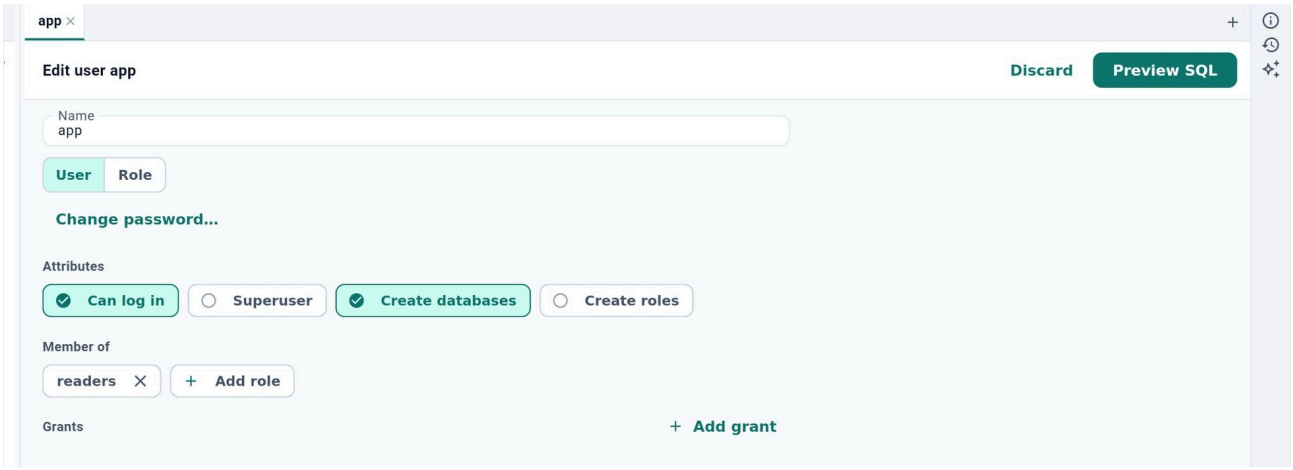


Figure 37: A user, its flags and its roles

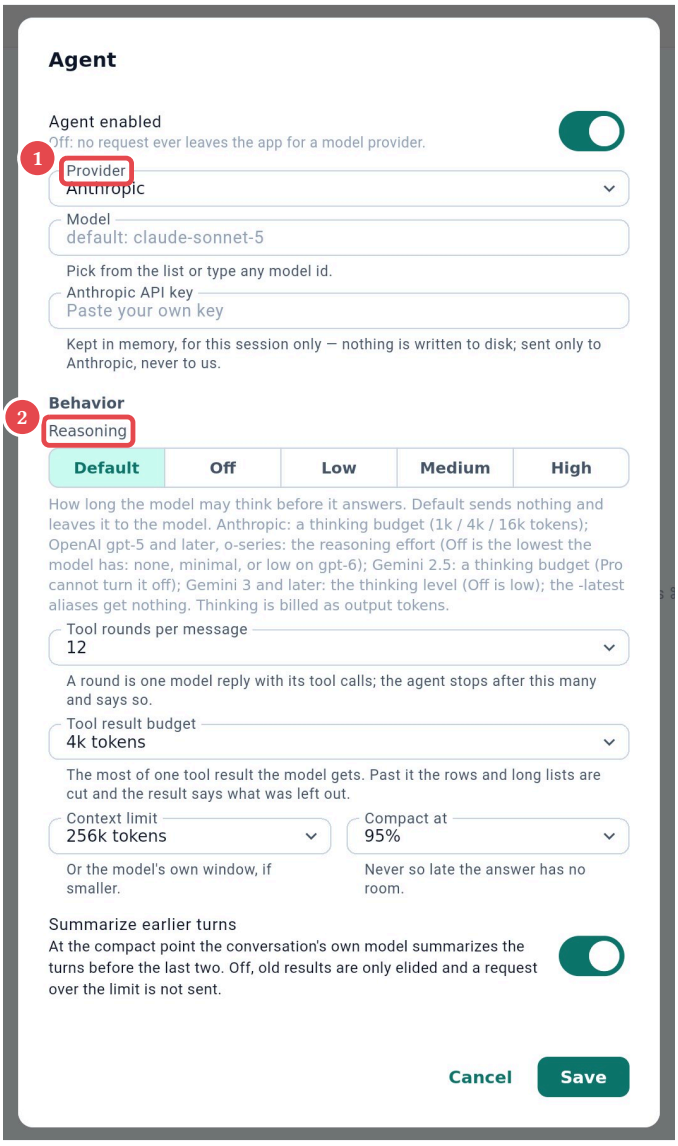
You can set the name, the password (**Change password…** on an existing user), whether it can log in, the database's own flags (on PostgreSQL **Superuser**, **Create databases** and **Create roles**), the roles it is a member of, and its grants. **Preview SQL** shows the statements; they run through the approval dialog. BigQuery has no users of its own — its access is managed in Google Cloud.

9 The AI agent

The agent answers questions about your databases in plain language: it reads the schema, runs read-only queries, explains plans, drafts SQL into the editor, builds dashboards and designs. It uses your own account with OpenAI, Anthropic or Google Gemini, and every write it proposes waits for your approval.

9.1 Setting it up

Click the agent button on the right edge, then **Agent settings**.



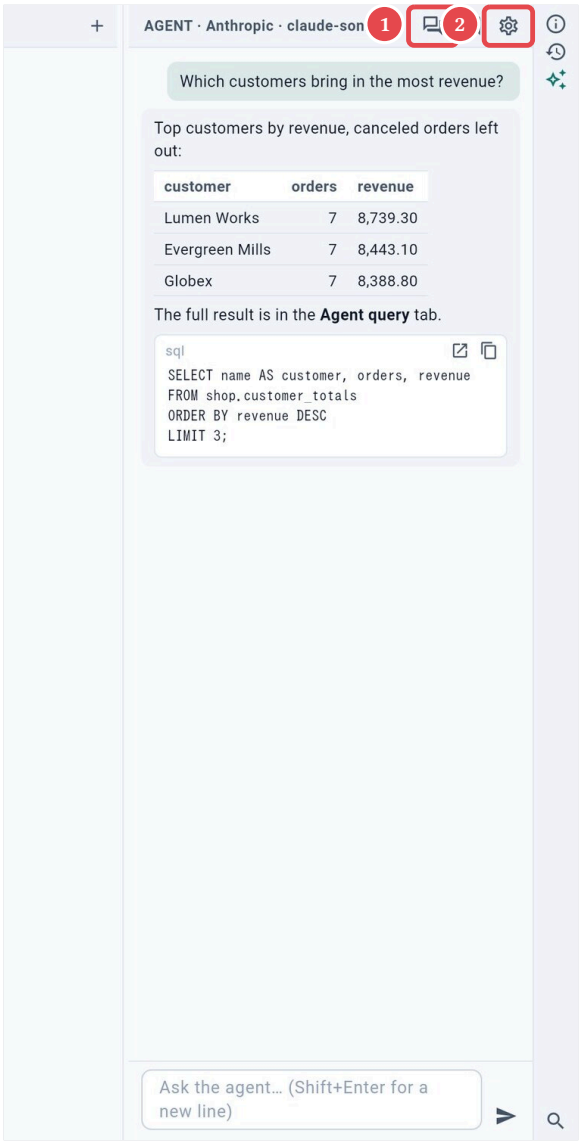
1. OpenAI, Anthropic or Gemini
2. How long the model may think

Figure 38: Agent settings

1. Turn on **Agent enabled**.
2. Choose the **Provider** 1 and a model, and paste your API key. The key is kept in the keychain and sent only to that provider.
3. Optionally change **Reasoning** 2 (how long the model may think), the number of tool rounds per message and the context limit.

9.2 Asking

Type a question in the panel and press `Enter` (`Shift+Enter` for a new line). The agent reads what it needs and answers with tables and SQL; a longer result opens in a query tab.



1. Earlier conversations
2. Provider, model and limits

Figure 39: An answer, with the query it ran

- 1 lists your earlier conversations; **New conversation** starts a fresh one.
- 2 opens the settings.
- Once a request has gone out, **What was sent** appears next to them and shows exactly what was sent to the model, request by request.

When a conversation gets long, the agent summarizes the earlier turns so it can keep going; the summary is marked in the conversation.

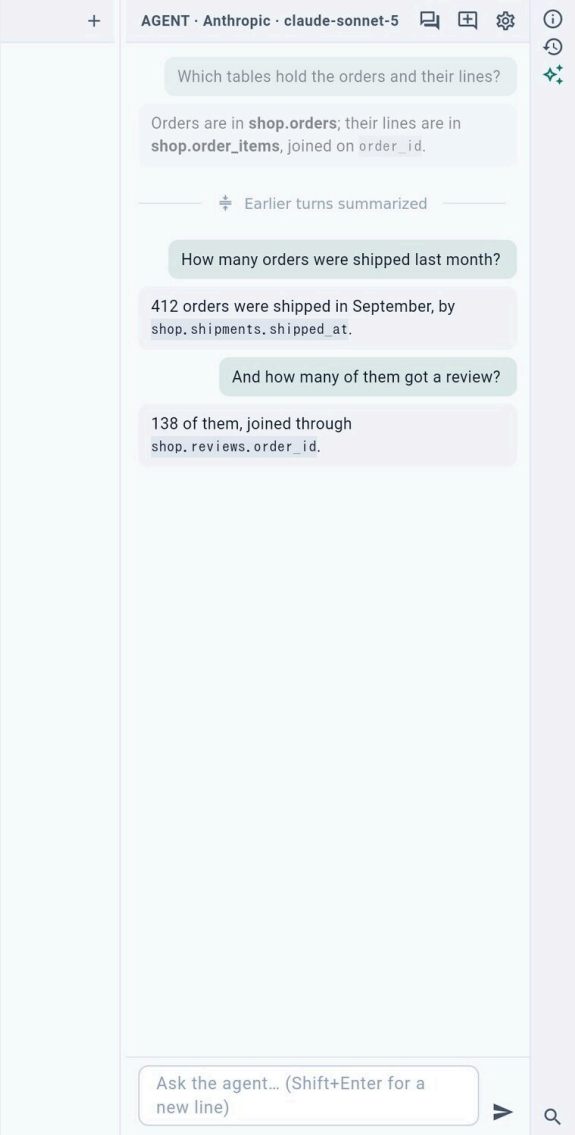


Figure 40: Earlier turns, summarized

9.3 What the agent can and cannot do

- It reads freely: schemas, row counts, plans, and the results of read-only queries.
- It never runs a write itself. An `UPDATE`, a `DELETE` or a schema change goes to the approval dialog (see Section 4.6), exactly as if you had typed it.
- It sees row values only on connections with **Agent may see rows** turned on. On every other connection it gets tables, columns and counts, never data. A **Sensitive** connection gives it the schema only.
- Your files can steer it: an `agent.md` in the workspace adds your own instructions to every conversation.

> **Note**
> Requests go straight from your computer to the provider you chose. Tabellum has no server of its own in between.

10 Backup, restore and sync

10.1 Backing up

Right-click a connection in the sidebar and choose **Back up…**.

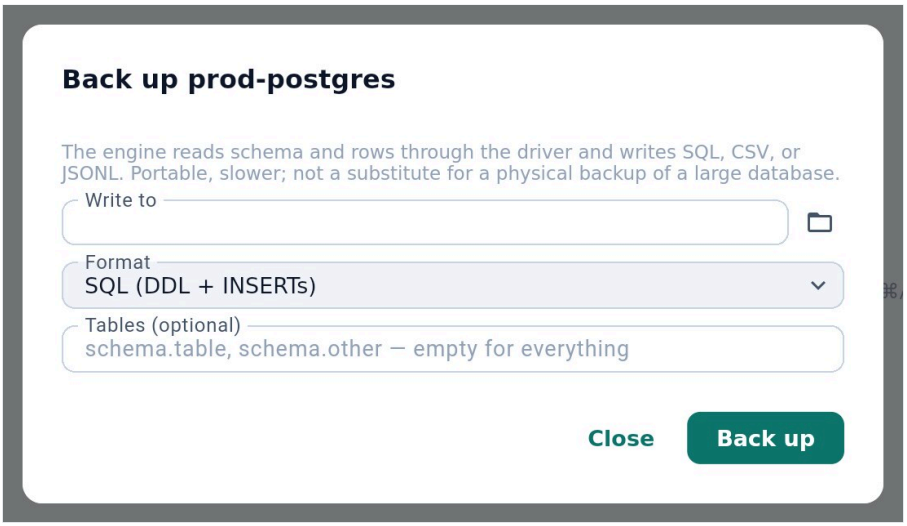


Figure 41: Backing up a database

The Mac App Store version backs up through the database connection itself: the schema and the rows are written as one SQL file (statements and `INSERT`s), as CSV or JSON Lines with one file per table, or as an Excel workbook with one sheet per table. **Tables** limits it to some tables. This is portable and works everywhere, but it is slower than the database's own backup tools and no substitute for them on a very large database.

10.2 Restoring

Restore…, in the same menu, runs an SQL file statement by statement, or loads CSV or JSON Lines rows into a table, in one transaction where the database supports it.

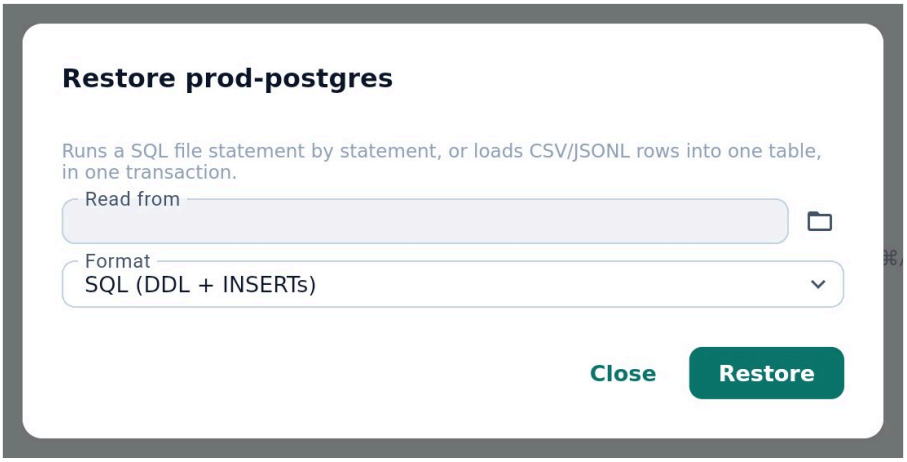
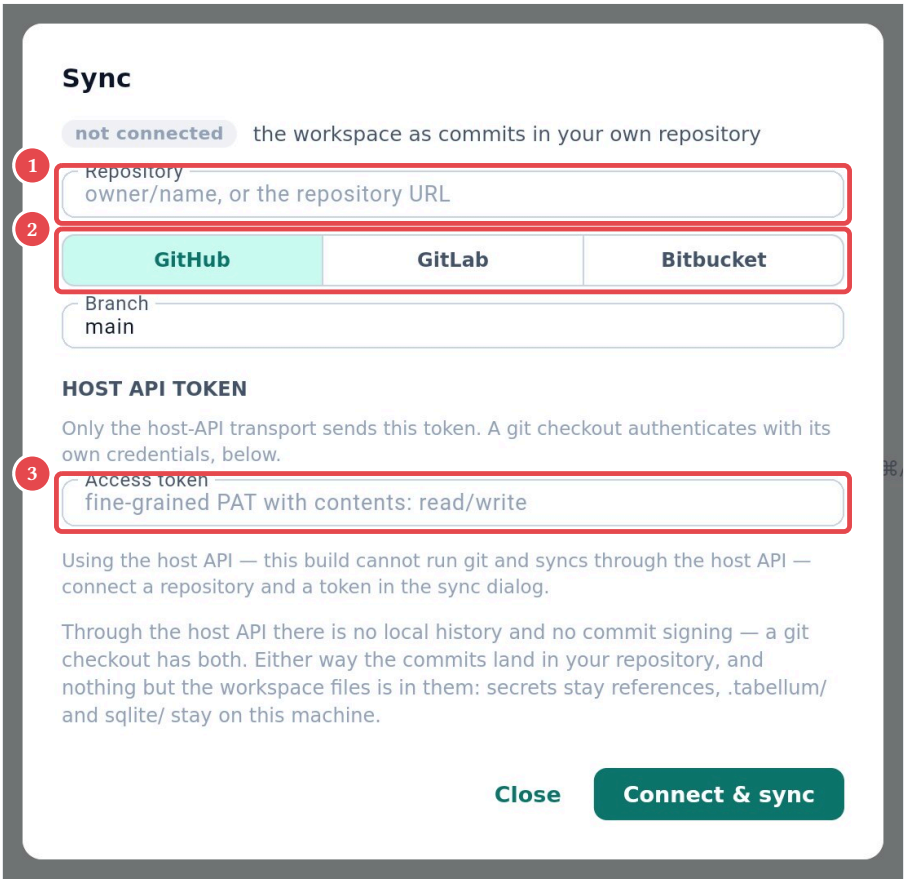


Figure 42: Restoring from a file

10.3 Syncing the workspace with git

Sync keeps the workspace in your own git repository on GitHub, GitLab or Bitbucket, so another computer — or your phone — sees the same connections, queries and dashboards.



1. Your repository
2. GitHub, GitLab or Bitbucket
3. An access token for that repository

Figure 43: Connecting the workspace to a repository

1. Choose **Sync** in the palette, or **Set up** at the bottom of the sidebar.
2. Enter the repository 1 and pick its host 2.
3. Paste an access token 3 that may read and write that repository.
4. Click **Connect & sync**.

Each sync commits your changes and brings in everyone else's. When two people edit the same connection, dashboard or tunnel, Tabellum merges them field by field; a real conflict is shown side by side for you to choose.

> **Note**
> Passwords, keys and tokens never go into the repository — only their names. On another device you type them once.

11 Reference

11.1 Keyboard shortcuts

On Windows and Linux, press `Ctrl` where the table says `Cmd`.

Keys	What it does
`Cmd+Enter`	Run the selection, or the statement at the caret
`Cmd+.` or `Esc`	Stop the running query
`Cmd+Shift+E`	Show the query plan, or hide it again
`Cmd+S`	Save the tab as a query file
`Cmd+P`	The command palette
`Tab`, `Enter`	Accept a completion
`Cmd+Z`, `Cmd+Shift+Z`	Undo and redo in a design

11.2 Databases

Database	Notes
PostgreSQL	Everything in this guide, including enum types, extensions, materialized views and plans
MySQL, MariaDB	Schemas are databases; users and roles with the server's own flags
SQL Server	Schemas, procedures, table-valued functions, users and roles
BigQuery	A dry run tells how many bytes a statement would process before it runs; access is managed in Google Cloud
SQLite	A file in the workspace; nothing to install
MongoDB	Collections as tables; find and aggregate pipelines in the editor

11.3 The Mac App Store version

The Mac App Store version runs in Apple's sandbox. Compared with the version downloaded from the website, it keeps its workspace in its own folder, backs up through the connection rather than with tools such as `pg_dump`, cannot use the SSH agent (paste the key instead), syncs through the GitHub, GitLab or Bitbucket API rather than a local `git`, opens one window, and has no local MCP server for external AI tools. Updates come from the App Store.

11.4 This guide in Tabellum

The same guide is inside the app, screenshots and marks included, and needs no network: choose **User guide** in the command palette (`Cmd+P`), the **User guide** button in Help (the **?** at the bottom of the sidebar), or, on a Mac, **Tabellum User Guide** in the **Help** menu. The chapter list switches to the mobile guide. On the website it is at tabellum.dudko.dev/manual, a page per chapter, with this PDF.

11.5 If something goes wrong

- **A connection fails.** The message under it says what failed and what to check; **Copy details** copies the full error. "Connection refused" means nothing listens on that host and port; a timeout usually means a firewall or a VPN.
- **A query takes too long.** Click the red **Stop** button or press `Esc`.
- **The agent does not answer.** Check its API key and model in **Agent settings**; **What was sent** in the agent's panel shows each request.

- **A design wants to undo someone else's change.** Use **Reload from the database** (see Section 7).

About Tabellum, in the ··· menu at the bottom of the sidebar (on a Mac, also in the **Tabellum** menu), shows the version. Support: tabellum.dudko.dev/support.