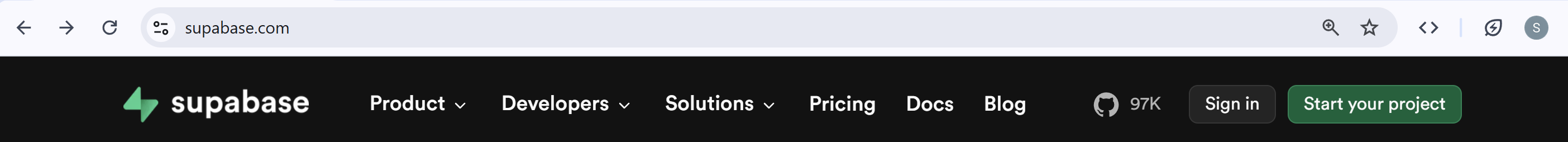
# Databases and Supabase – Exercises

**Exercises** and **homework** assignments for “Software Technologies with AI” course at SoftUni AI.

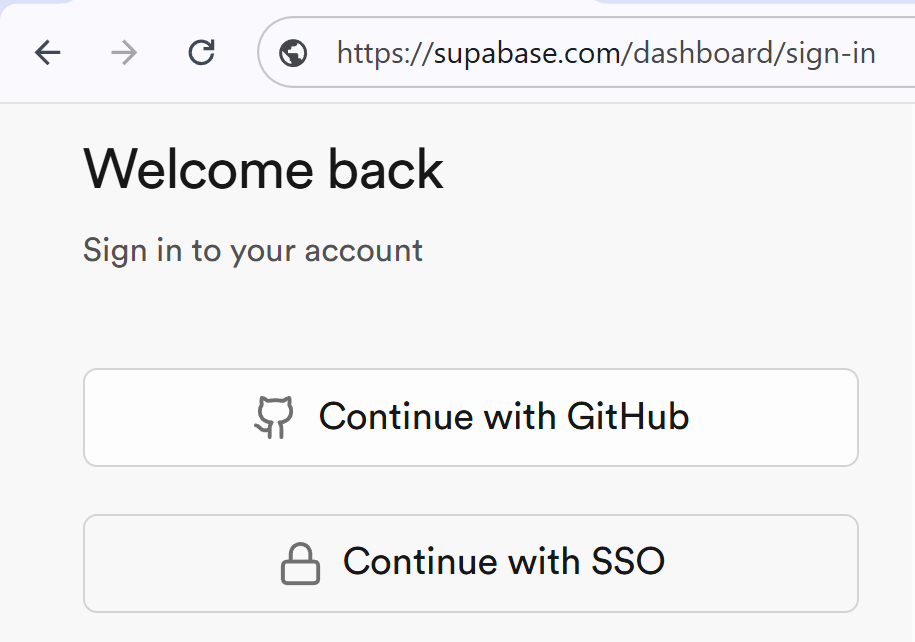
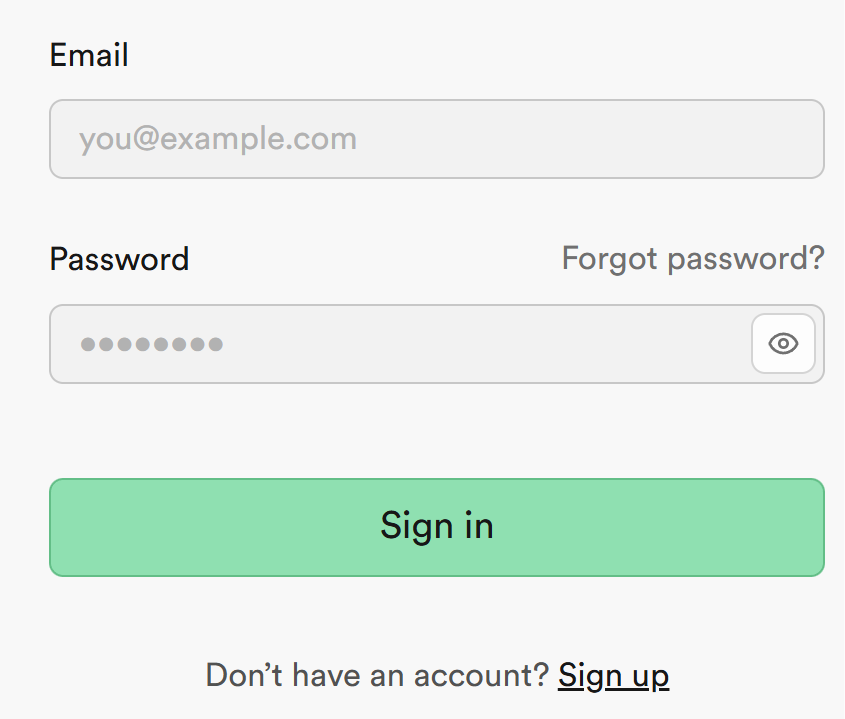
## Preparation

### Register a Supabase Account

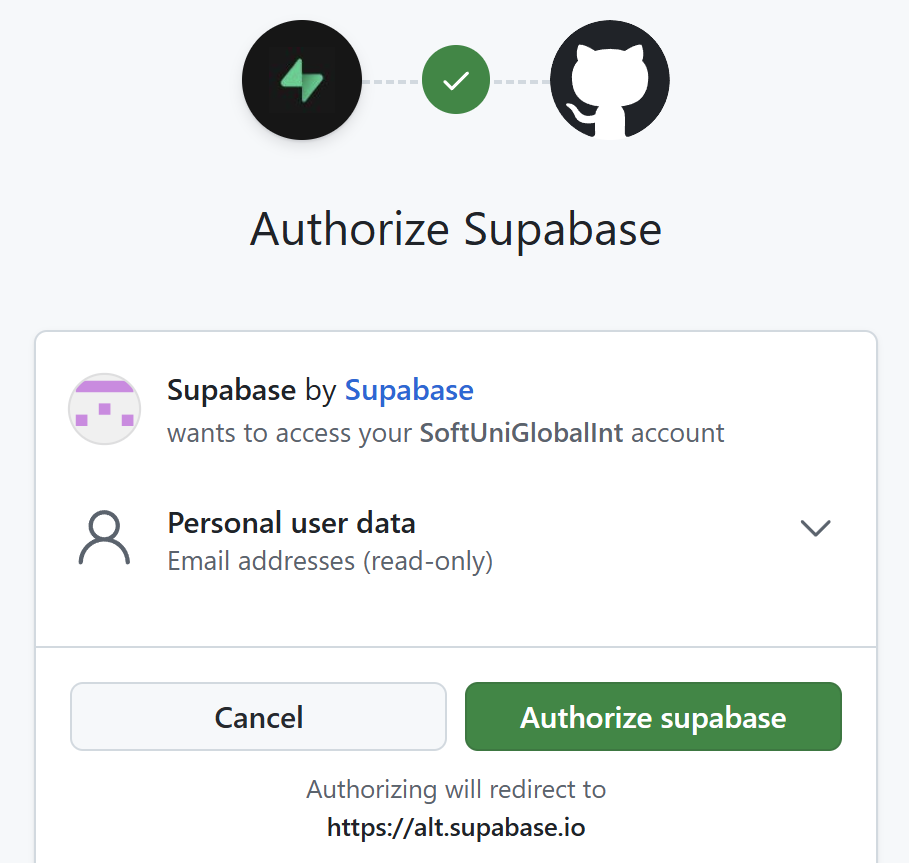
Visit the [https://supabase.com](https://supabase.com/) website and **log in** (if you have already created an account) or **create** a free **account**. Select [Sign In] from the navigation bar:



**Sign in** with your credentials or, if you haven’t got an account, select [Sign up] under the green [**Sign in**] button:

You can use your **GitHub credentials** or create an account with other credentials. **Authorize Supabase** if you have chosen **GitHub**:



### Create a New Visual Studio Code Project

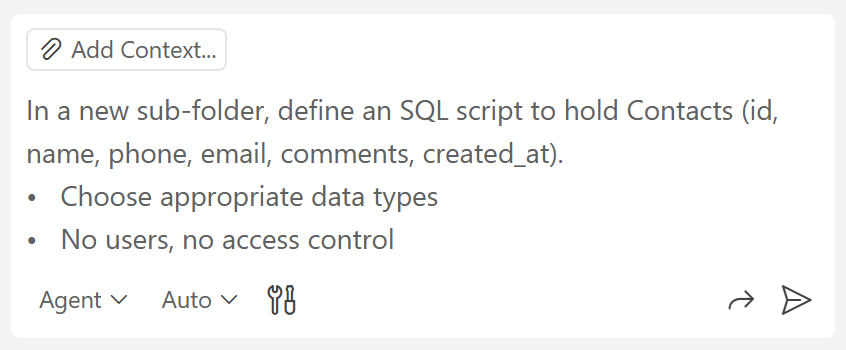
Create a **new project** in **Visual Studio Code** for this exercise, as we did in the previous exercises.

## Contacts DB Design

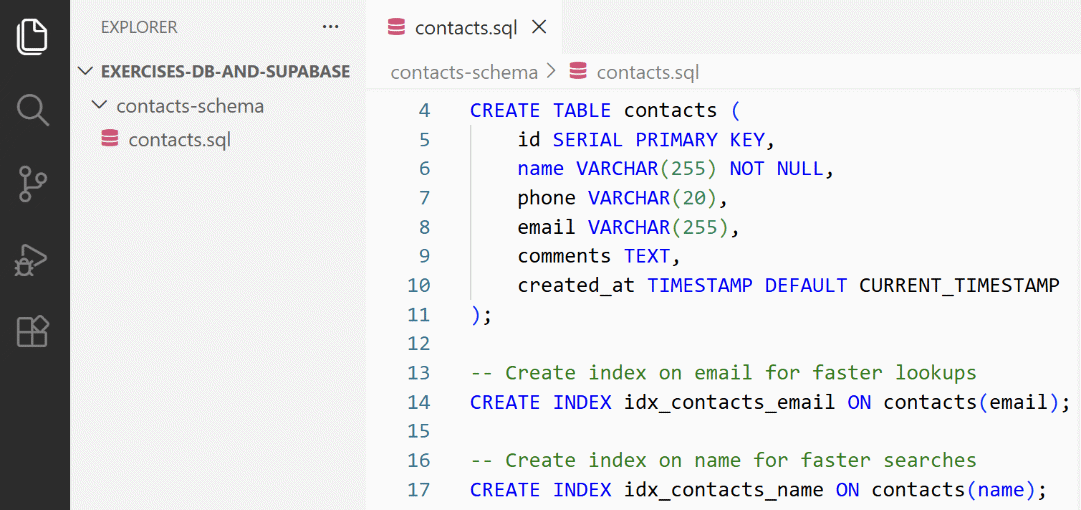
### Define a DB Schema as SQL

* **Define** a Supabase **table** (as **SQL script**) to hold **Contacts** (id, name, phone, email, comments, created\_at) using an **AI chatbot**.
* Choose appropriate **data types**.
* No users, **no access control**.

#### The AI Prompt



The result should be **new** .sql **file** containing an SQL script, which creates the contacts database table:

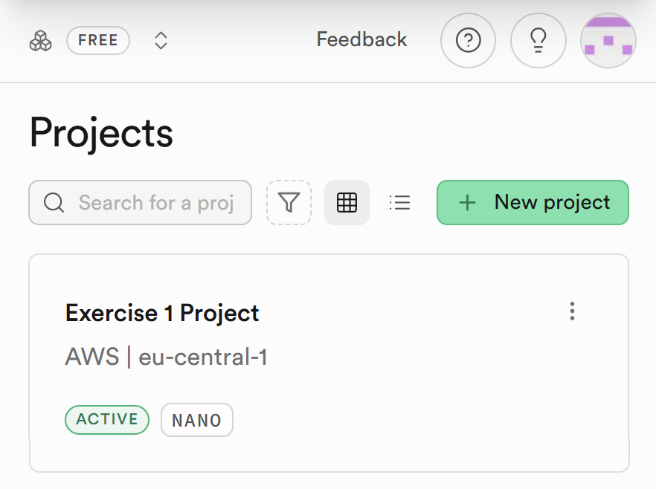


### Populate the DB Schema in Supabase

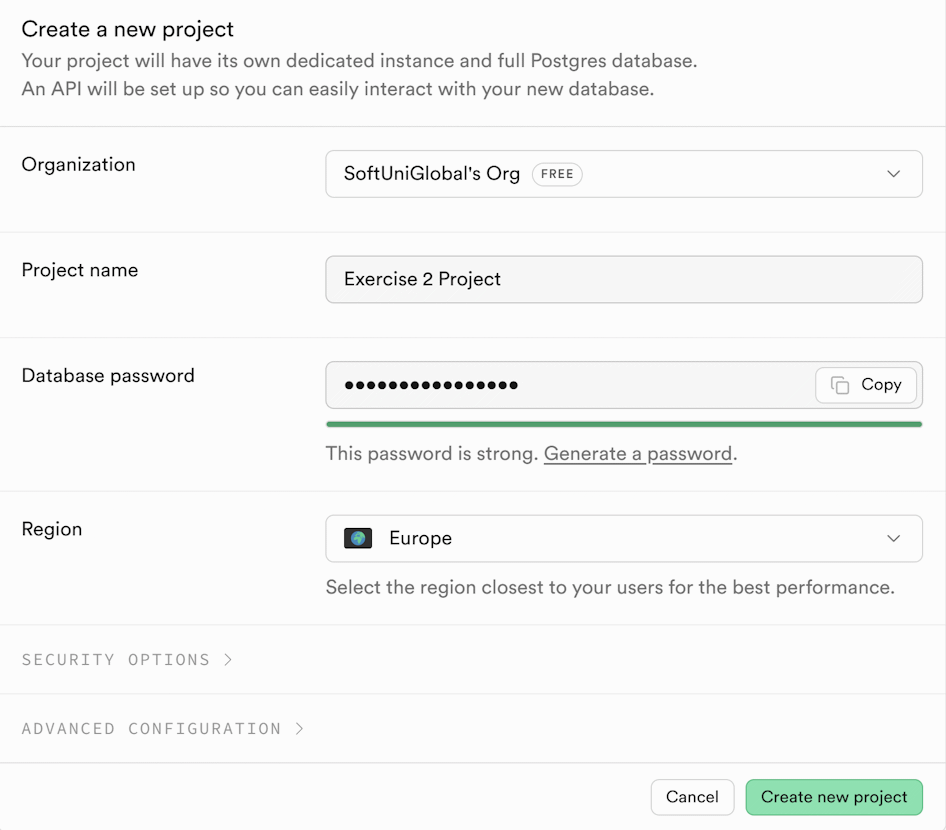
Now run the DB script in your Supabase DB to create the contacts table and its associated database objects.

#### New Supabase Project

Create a new Supabase project by clicking the [New Project] button:



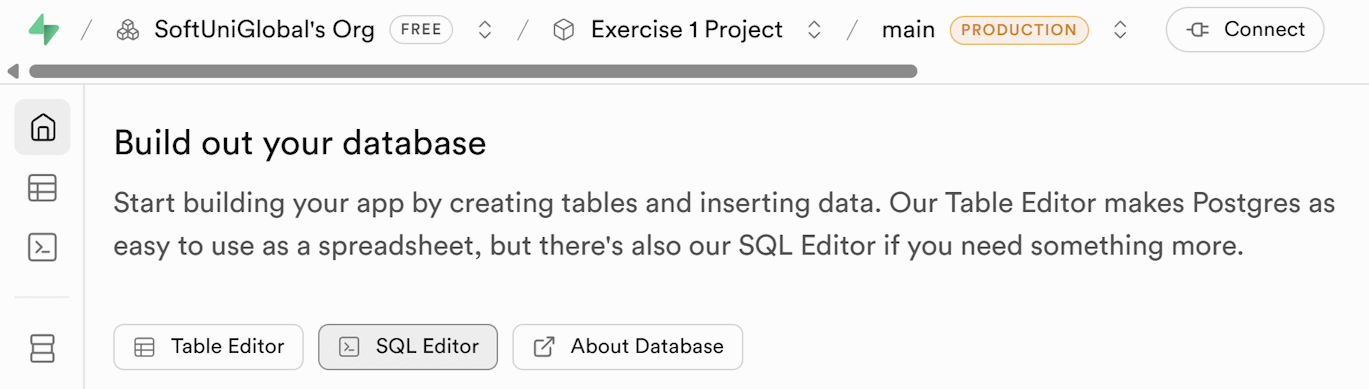
Set up a proper **name** and **password**. Set the **region** to be **Europe**. Click the [Create new project] button when you are ready:



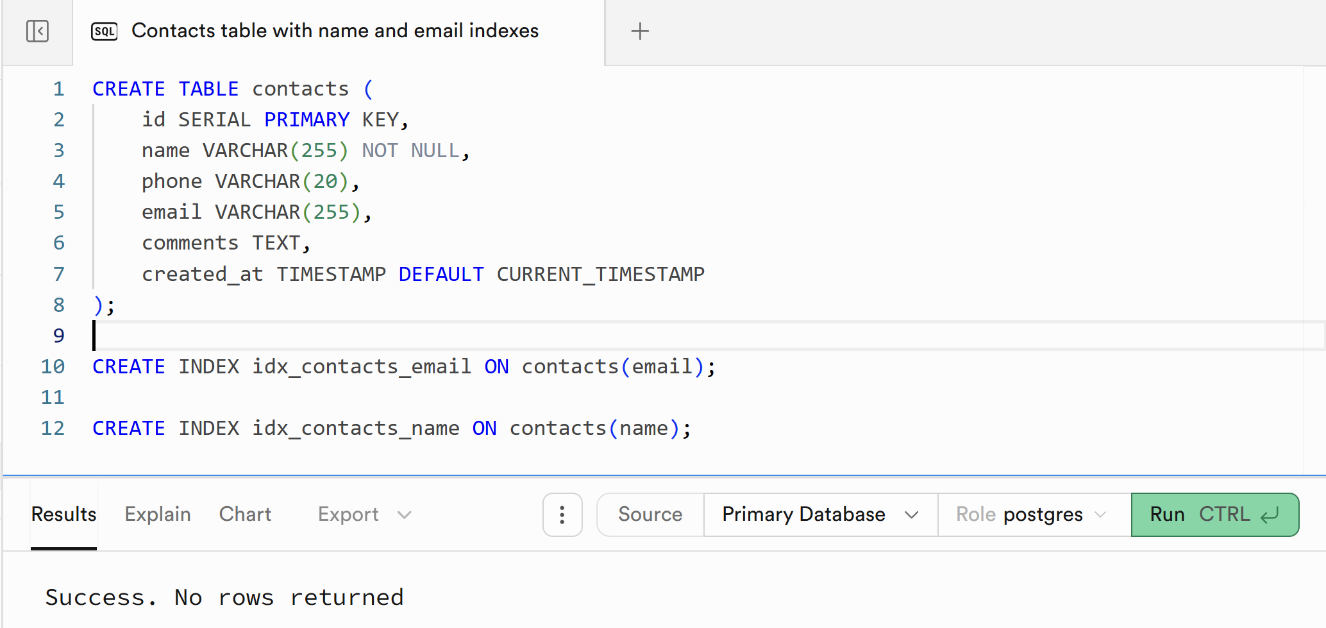
#### New Table

**Copy** the **script from Visual Studio Code** and **run** the **SQL script** in your **Supabase project** to **create** **the DB tables**.

**Open** **the project** you have created and select the [SQL Editor] at the bottom of the page:

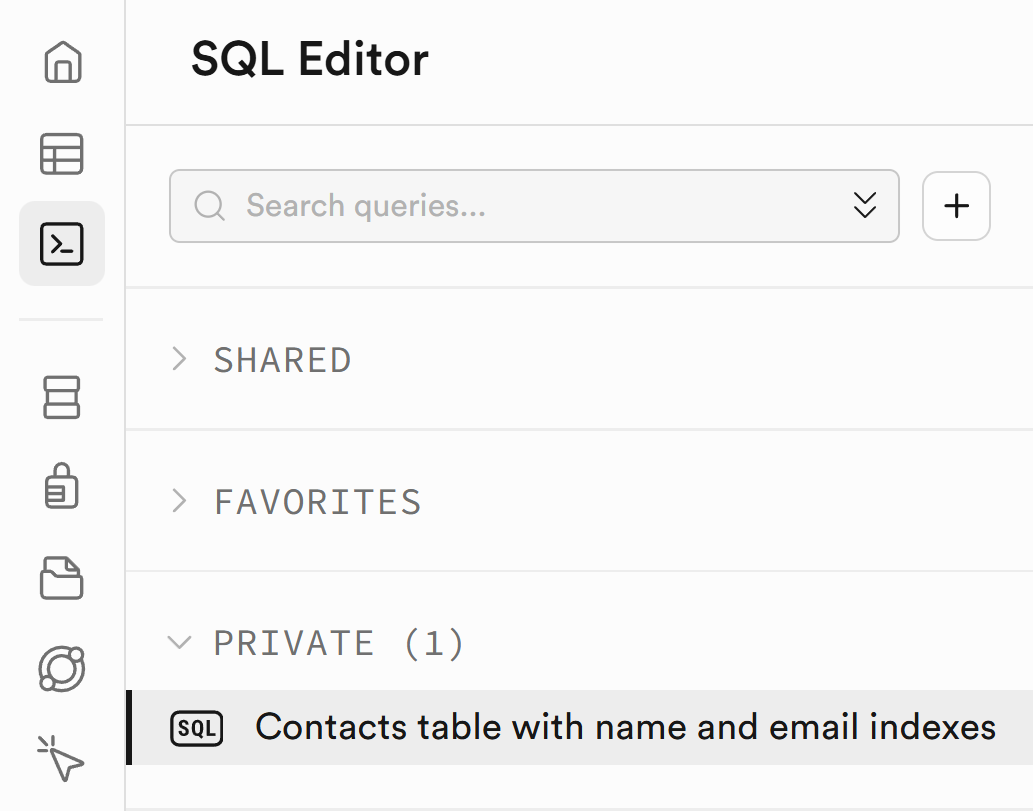
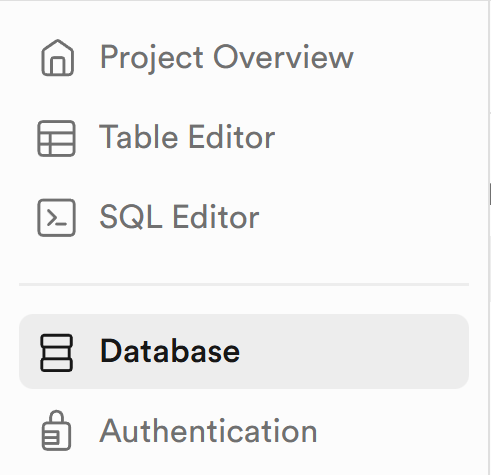


Paste the script and select the [Run] button:



If the script has been **run successfully**, you will see the message from the screenshot above in the **Results** section.

You can see the table you have created by selecting [Database] from the navigation bar to the left:

### Sample Data in Supabase

Insert **sample data** in the contacts table in **Supabase**.

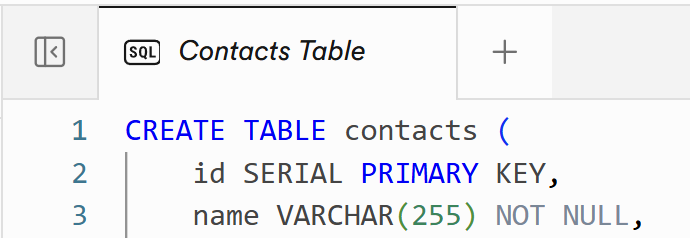
#### The AI Prompt

Using an AI chatbot, **generate sample data** (as SQL script)

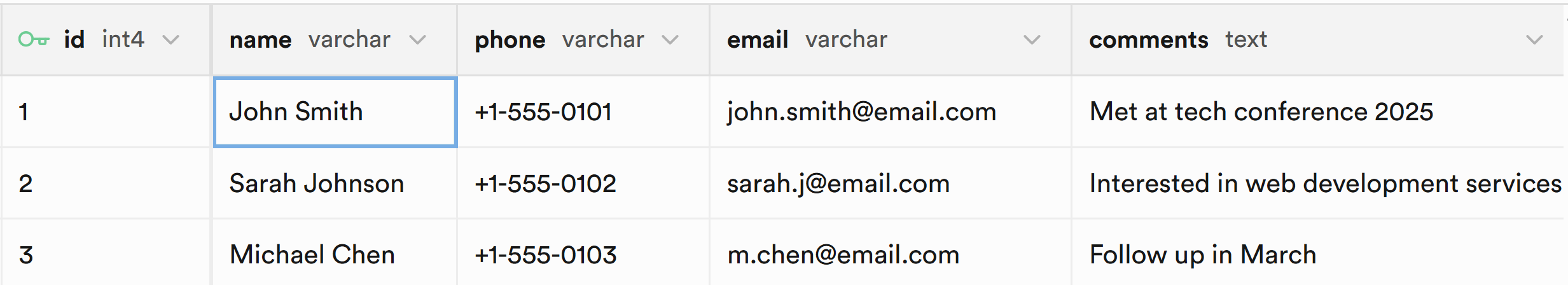


#### Seed the Database

**Run the SQL script** in your Supabase project to **fill the DB tables** with sample **data**. Inserting sample data in a database for testing purposes is known as "***seeding the database***". Open a **new script** from the [+] button next to the first script tab before doing so or you will get an error:



You can find the details you have inserted from **the Table Editor** menu on the **left navigation bar**. Your table should resemble this sample:



## Modify Contacts DB Schema

Using an AI chatbot, **add** a new table **column**. This operation will modify your **database schema**.

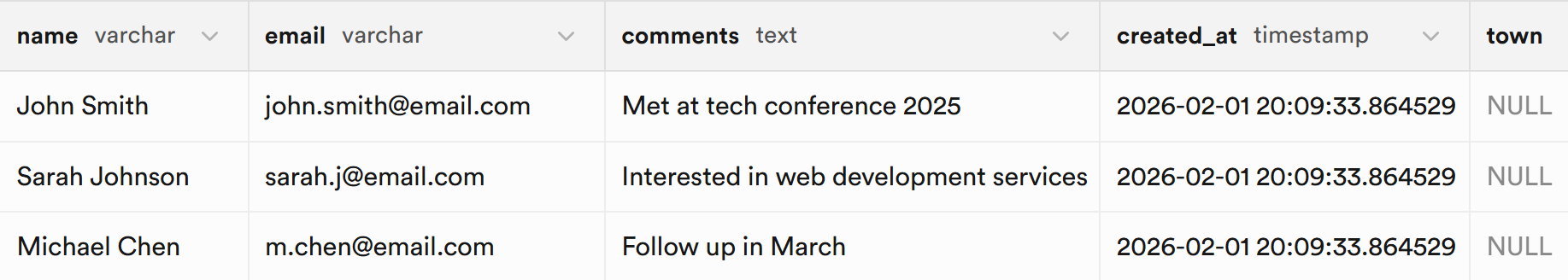
* + Modify the contacts table: add a new column "town".
* Leave empty the new column for all existing contacts.
  + Generate an SQL script for the DB schema change.
  + Run the SQL script in your Supabase project.

### The AI Prompt



### The Updated Table

Now you have a new column `town` inside the contacts table, holding empty (NULL) values.



## Insert Table Data

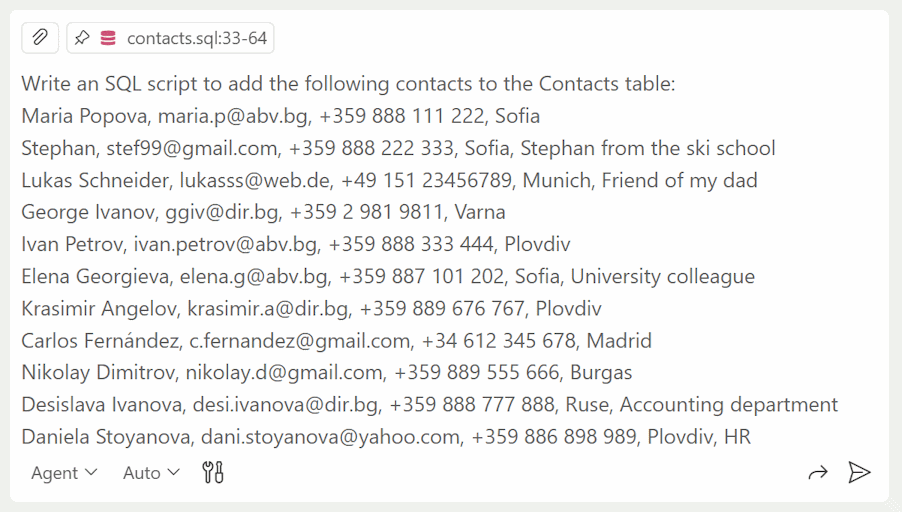
We shall insert **additional table data**, which we shall use during the SQL queries later.

Using AI chatbot, write an **SQL script** to **add** the following contacts to the contacts table:

|  |
| --- |
| - Maria Popova, maria.p@abv.bg, +359 888 111 222, Sofia  - Stephan, stef99@gmail.com, +359 888 222 333, Sofia, Stephan from the ski school  - Lukas Schneider, lukasss@web.de, +49 151 23456789, Munich, Friend of my dad  - George Ivanov, ggiv@dir.bg, +359 2 981 9811, Varna  - Ivan Petrov, ivan.petrov@abv.bg, +359 888 333 444, Plovdiv  - Elena Georgieva, elena.g@abv.bg, +359 887 101 202, Sofia, University colleague  - Krasimir Angelov, krasimir.a@dir.bg, +359 889 676 767, Plovdiv  - Carlos Fernández, c.fernandez@gmail.com, +34 612 345 678, Madrid  - Nikolay Dimitrov, nikolay.d@gmail.com, +359 889 555 666, Burgas  - Desislava Ivanova, desi.ivanova@dir.bg, +359 888 777 888, Ruse, Accounting department  - Daniela Stoyanova, dani.stoyanova@yahoo.com, +359 886 898 989, Plovdiv, HR department  - Dimitar Angelov, d.angelov@proton.me, +359 889 111 999, Veliko Tarnovo  - Petar Nikolov, petar.nikolov@abv.bg, +359 888 454 545, Varna  - Silvia Kirova, silvia.kirova@gmail.com, +359 887 232 323, Varna, Works in tourism  - Atanas Georgiev, atanas.g@gmail.com, +359 888 303 404, Varna, Friend from university  - Kalina Petrova, kalina.p@gmail.com, +359 888 010 203, Blagoevgrad, Marketing team  - Todor Marinov, t.marinov@abv.bg, +359 887 777 121, Sofia, Gym buddy  - Erik Svensson, erik.svensson@gmail.com, +46 70 123 45 67, Stockholm, Manager for Sweden  - Milena Todorova, milena.t@abv.bg, +359 887 121 212, Gabrovo  - Boyan Iliev, boyan.iliev@gmail.com, +359 889 808 707, Plovdiv, Startup founder  - Radostina Peeva, r.peeva@yahoo.com, +359 885 333 222, Sofia  - Agnieszka Nowak, a.nowak@yahoo.pl, +48 698 765 432, Kraków, Ski instructor  - Stoyan Vasilev, stoyan.v@gmail.com, +359 888 222 101, Bansko, Ski instructor  - Veselina Hristova, veselina.hr@abv.bg, +359 886 555 444, Varna, HR contact  - Sophie Dubois, sophdub@yahoo.fr, +33 6 12 34 56 78, Lyon, Met in Paris  - Georgi Stoyanov, g.stoyanov@yahoo.com, +359 886 123 456, Sofia  - Hristo Kolev, hristo.kolev@gmail.com, +359 885 909 090, Stara Zagora, Old classmate  - Yana Nikolova, yana.n@abv.bg, +359 887 444 555, Pleven |

### The AI Prompt

You can use a simple AI prompt like this:



**Run the SQL script** in your Supabase project. Refresh and see if the changes have been made in **Table Editor**.

## Modify Table Data

Use an AI chatbot to **write SQL scripts** **to perform the following DB changes** in the contacts table:

* + Add a comment for Stoyan Vasilev: "father of Radi".
  + Change Georgi Stoyanov's data -> Tel. +49 176 98765432, moved to Berlin.
  + Delete the email for Desislava Ivanova.
  + Remove phone number and email from contact Elena Georgieva.

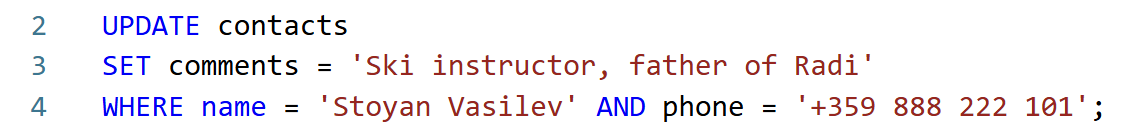
### Append Data

You can generate **SQL UPDATE** statement to **append data** in certain table column.

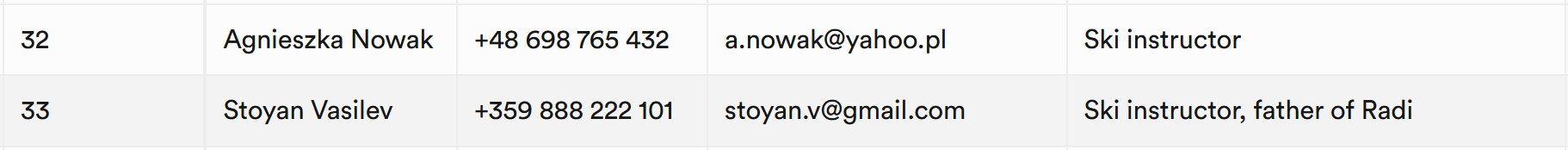
#### The AI Prompt



#### The Script



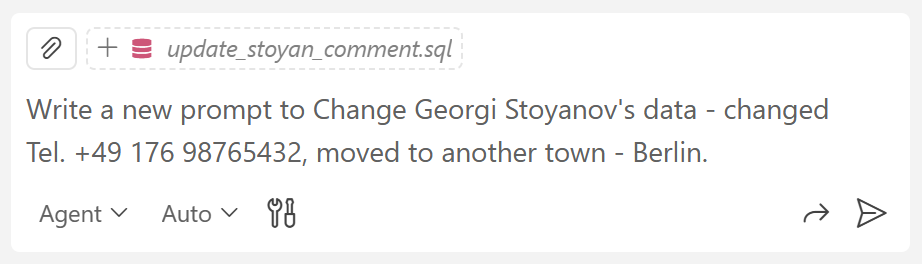
#### The Updated Comment



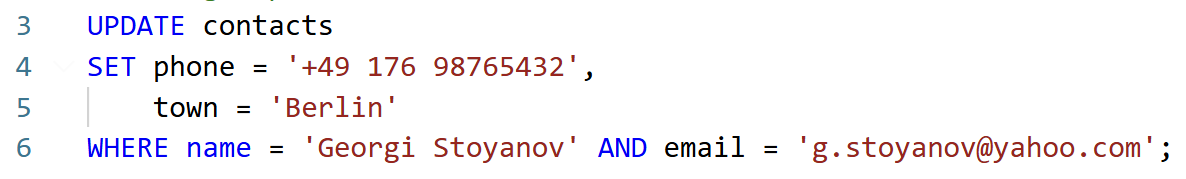
### Modify Data

You can generate **SQL UPDATE** statement to **modify existing data** in certain table column.

#### The AI Prompt



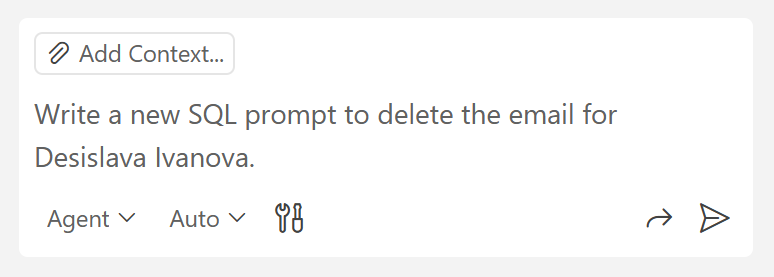
#### The Script



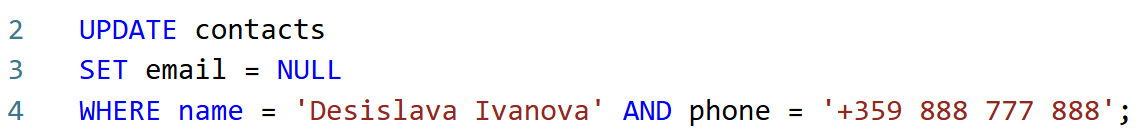
### Update / Delete Data

You can generate **SQL DELETE** statement to **delete existing data** rows from certain table.

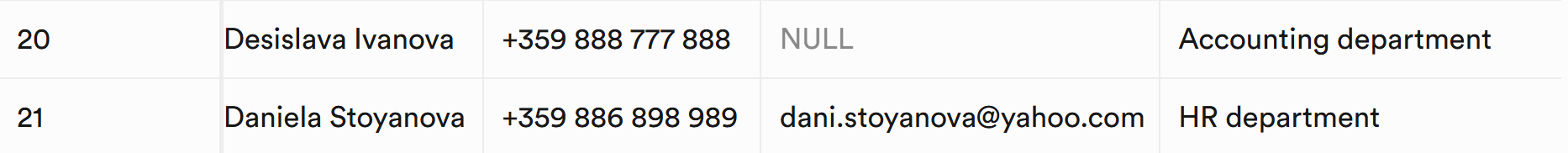
#### The AI Prompt



#### The Script



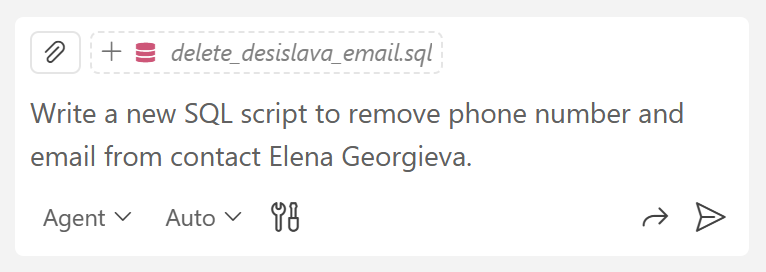
#### The Result



### Remove Data

You can generate **SQL UPDATE** statement to **remove existing data** (assign NULL values in certain table column).

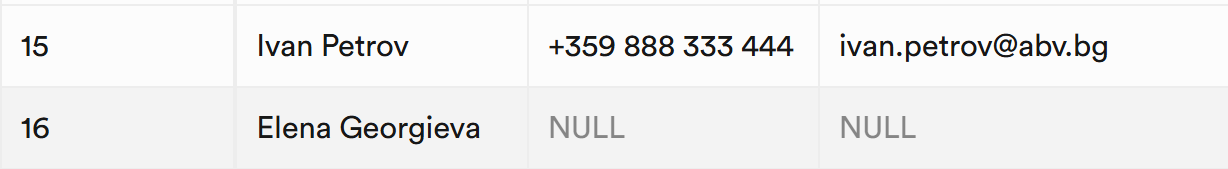
#### The AI Prompt



#### The Script



#### The Result

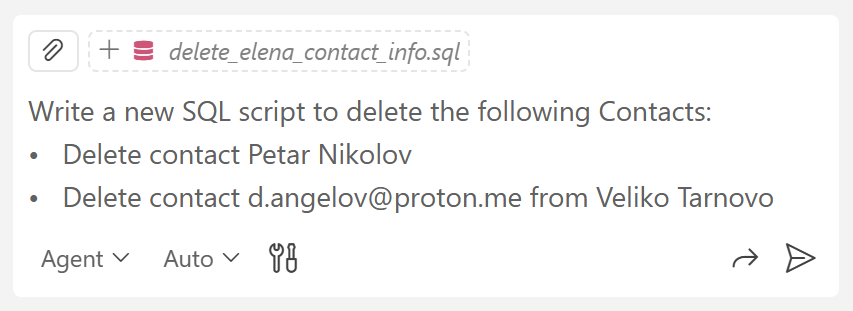


## Delete Table Rows

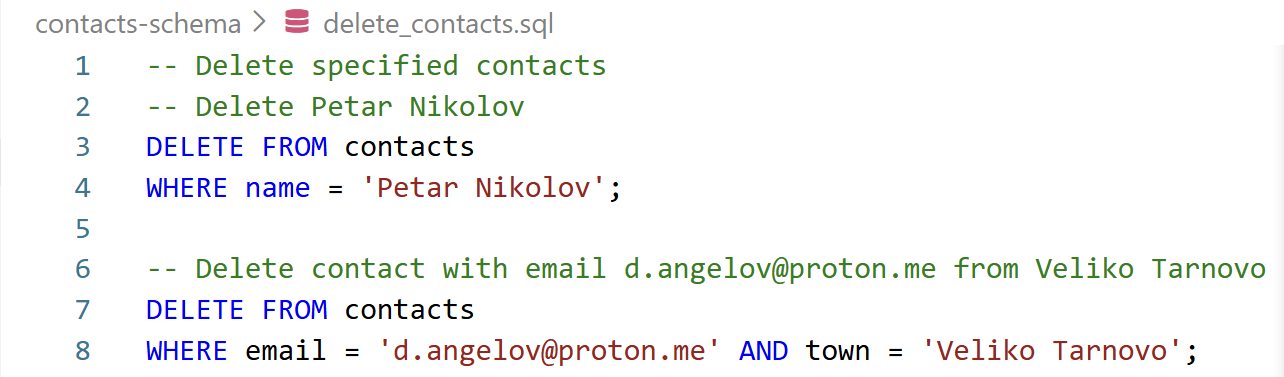
Use an AI chatbot to **write a new SQL script to** **delete** the following **contacts**:

* + Delete contact Petar Nikolov
  + Delete contact d.angelov@proton.me from Veliko Tarnovo

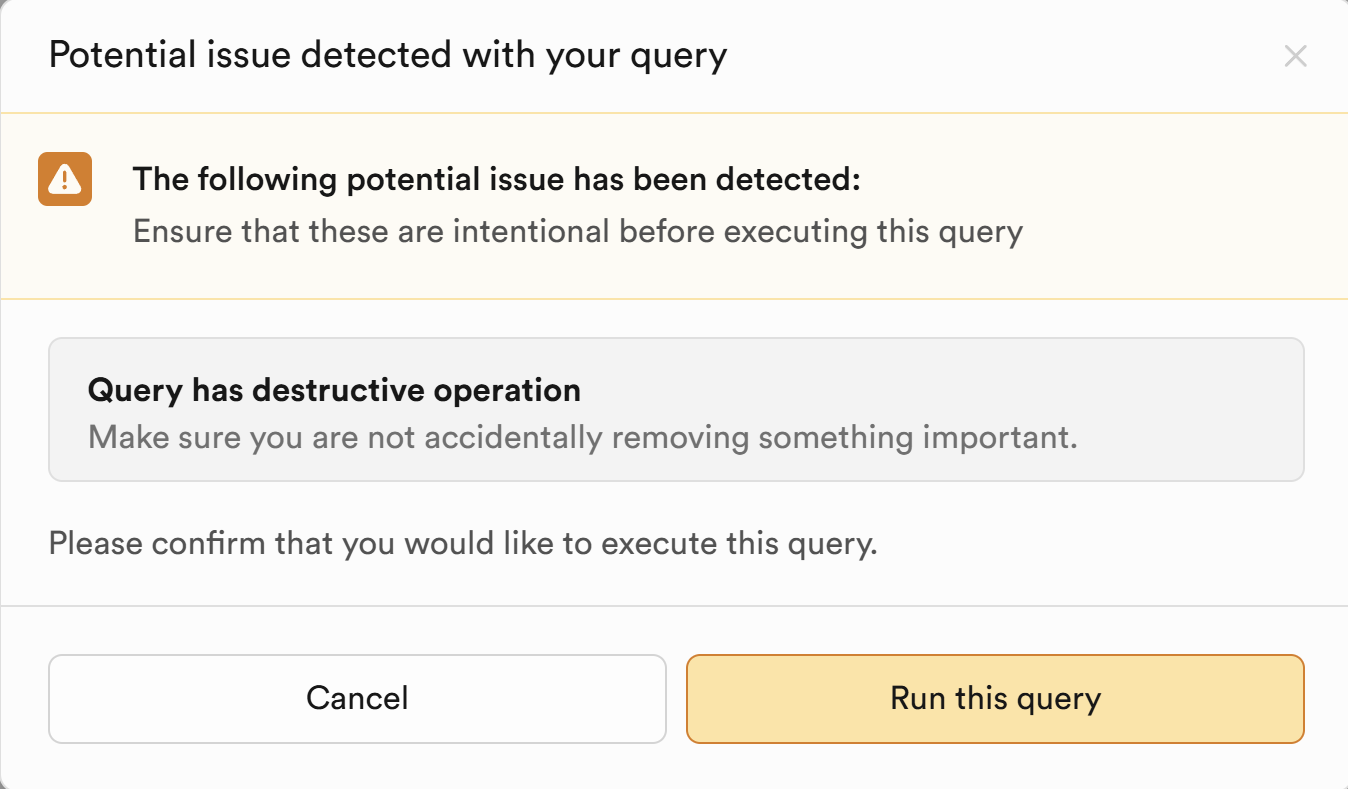
### The AI Prompt



### The Script



You may get the following **warning**. It says that we may delete many rows by mistake, and we should be careful. Review the script one more time and press the [Run this query] button:



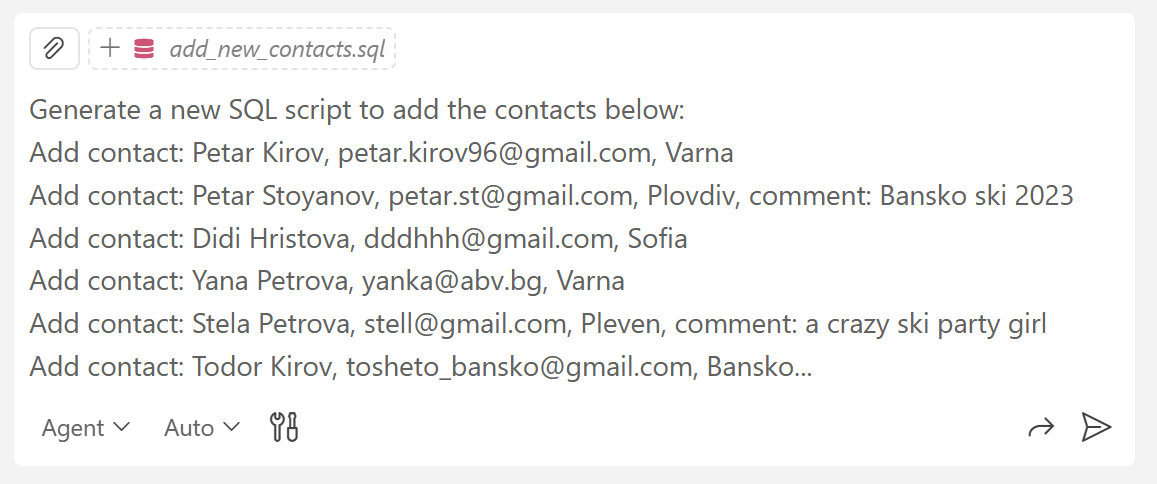
## Add More Contacts

Let’s **add more contact data** to the database, so we can run specific queries matching this data.

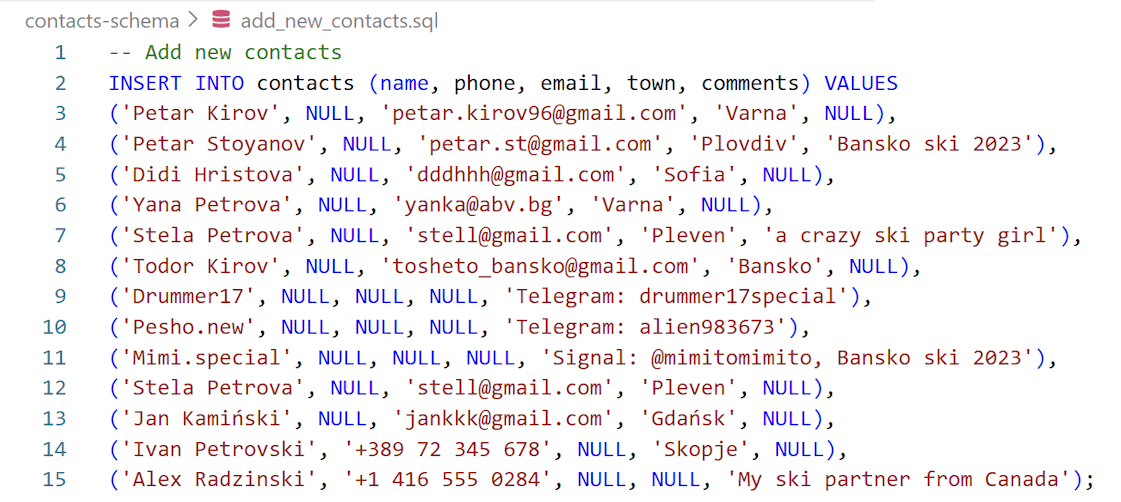
Using an AI chatbot, **generate** **a new SQL script** **to add the contacts below**:

|  |
| --- |
| - Add contact: Petar Kirov, petar.kirov96@gmail.com, Varna  - Add contact: Petar Stoyanov, petar.st@gmail.com, Plovdiv, comment: Bansko ski 2023  - Add contact: Didi Hristova, dddhhh@gmail.com, Sofia  - Add contact: Yana Petrova, yanka@abv.bg, Varna  - Add contact: Stela Petrova, stell@gmail.com, Pleven, comment: a crazy ski party girl  - Add contact: Todor Kirov, tosheto\_bansko@gmail.com, Bansko  - Add contact: Drummer17, no email, no phone, Telegram: drummer17special  - Add contact: Pesho.new, no email, no phone, Telegram: alien983673  - Add contact: Mimi.special, no email, no phone, Signal: @mimitomimito, comment: Bansko ski 2023  - Add contact: Stela Petrova, stell@gmail.com, Pleven  - Add contact: Jan Kamiński, jankkk@gmail.com, Gdańsk  - Add contact: Ivan Petrovski, +389 72 345 678, Skopje  - Add contact: Alex Radzinski, +1 416 555 0284, comment: My ski partner from Canada |

### The AI Prompt



### The Script



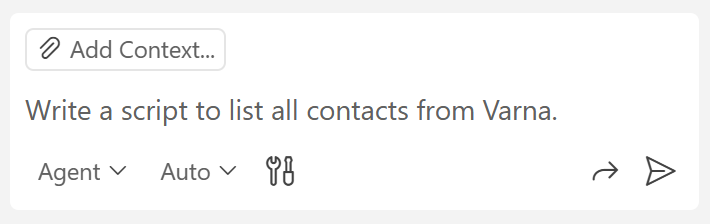
## Contacts DB Queries

Use an AI chatbot to **write a few simple SQL queries** on the contacts to filter them:

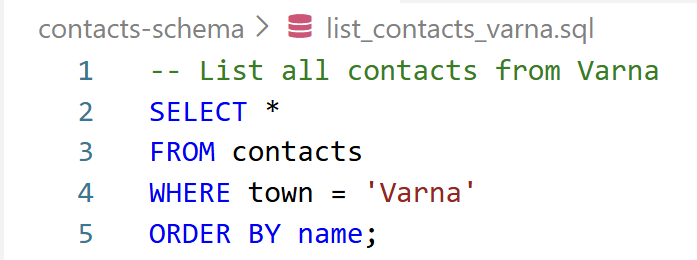
* + List all contacts from Varna.
  + List all contacts without a town and without an email.
  + List all contacts, which mention "ski" (as a substring).
  + List all contacts, which have a phone number from Bulgaria (+359 ...).
  + List all contacts (name, town, phone, email), order by towns alphabetically.
  + List all contacts from Sofia, Plovdiv, or Varna.
  + Find all contacts with Gmail accounts grouped by town.

### Example

#### The AI Prompt

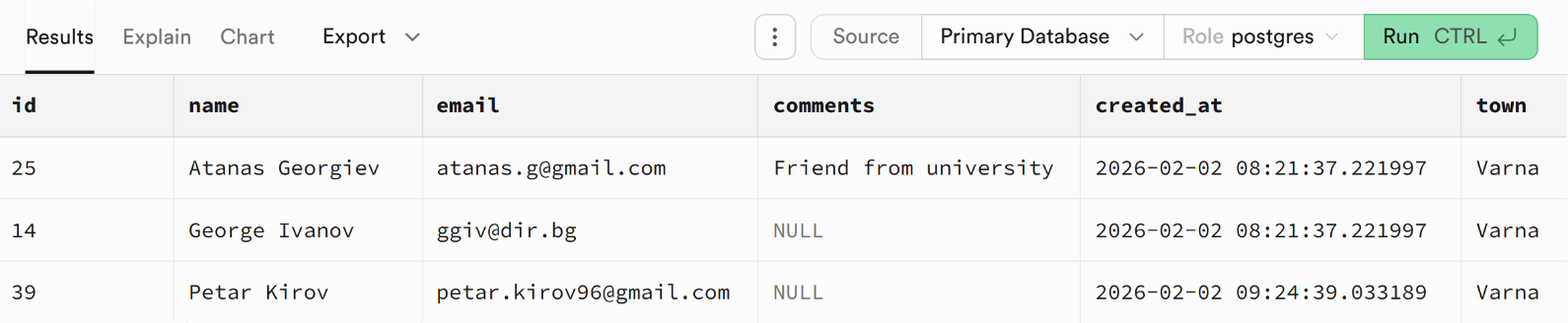


#### The Script



#### The Result

The results will be displayed in the **Results section** below the script:



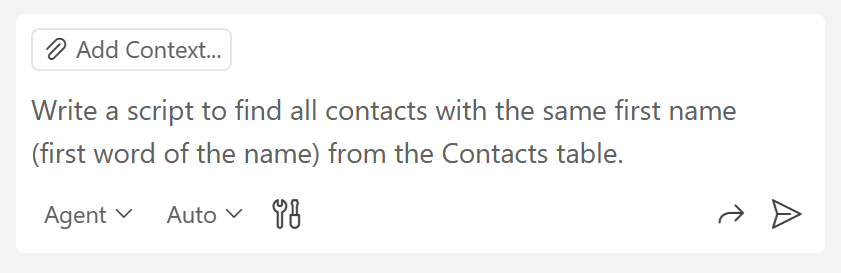
## Contacts Advanced DB Queries

Use an AI chatbot to **write a few more complex SQL queries** on the contacts:

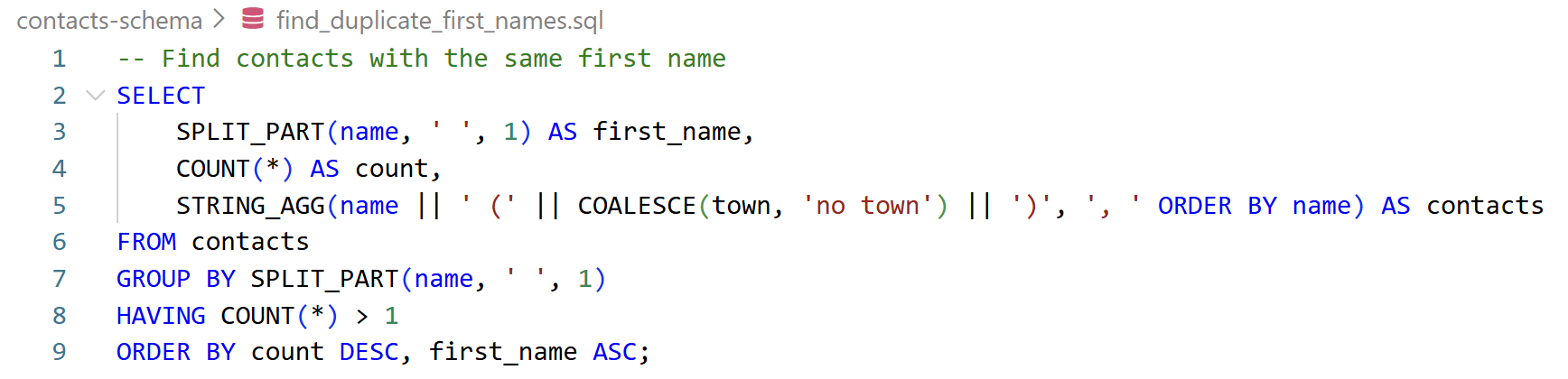
* + Find all contacts with the **same first name** (first word of the name)
    - Print first name | list of contacts (names), comma separated
  + List all **towns** and the **number of contacts** for each town
    - Also include the contacts with no town as town "[Unknown]"
  + List contacts with **incomplete information** (missing phone OR email).
  + Return name | phone | email | missing data
  + Missing data column: No contact info | Missing phone | Missing email
  + List all **email domains**, along with contacts (command separated, alphabetically)
    - Return domain | contacts\_count | contact\_names
  + Find **contacts per town**, with **rank** and **percentages**
    - Return town | rank | percentage | count | contact list (comma separated)
    - Include also contacts with no town and their percentage

### Contacts with the Same First Name

#### The AI Prompt



#### The Script

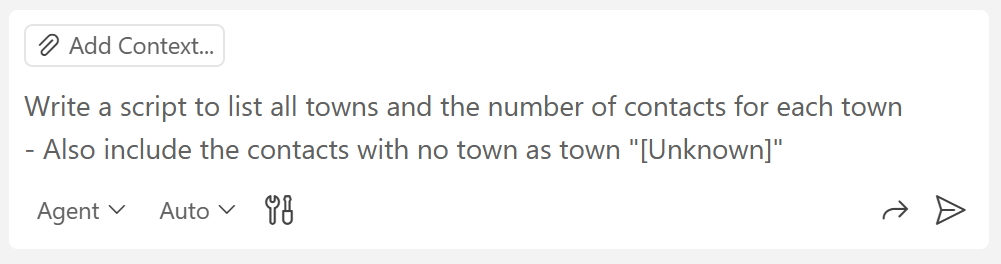


#### The Result

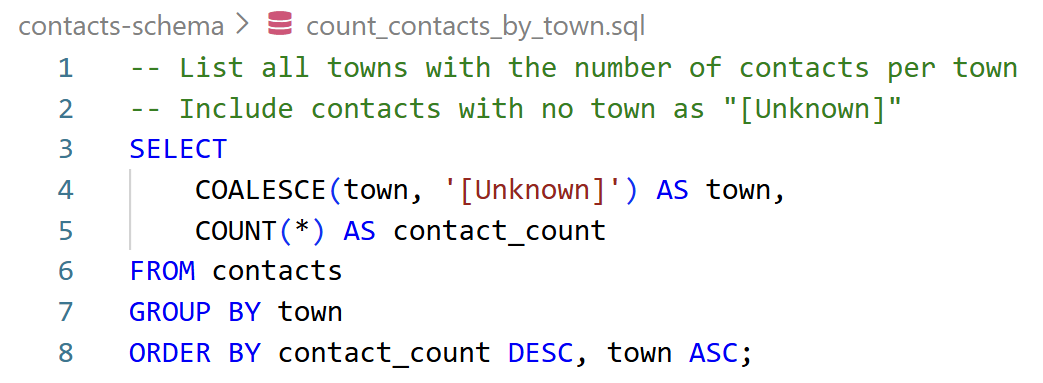


### Towns and Contacts from Towns

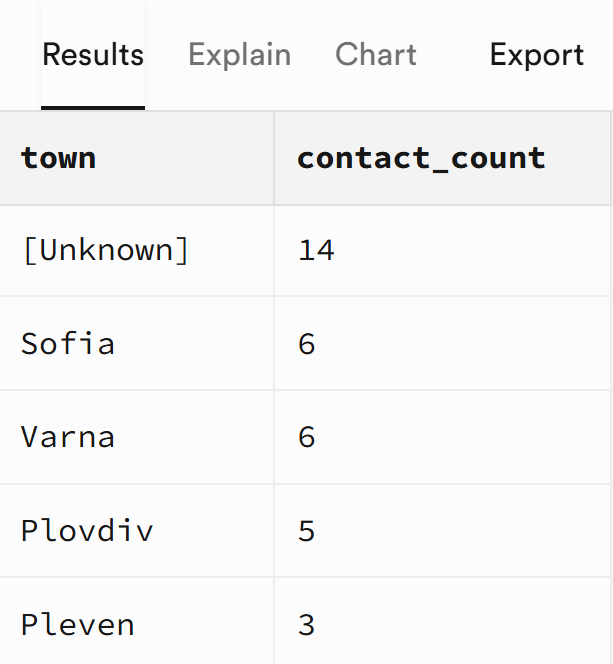
#### The AI Prompt



#### The Script



#### The Result



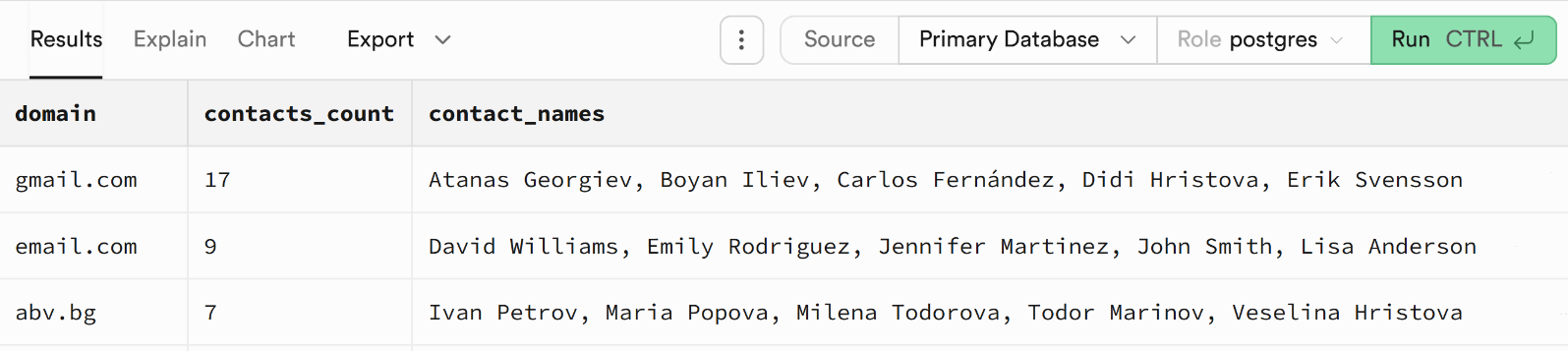
### Email Domains with Contacts

#### The AI Prompt



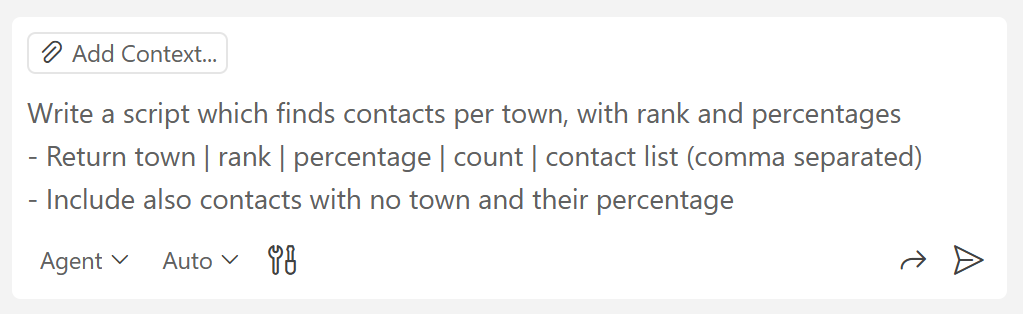
#### The Script

#### The Result

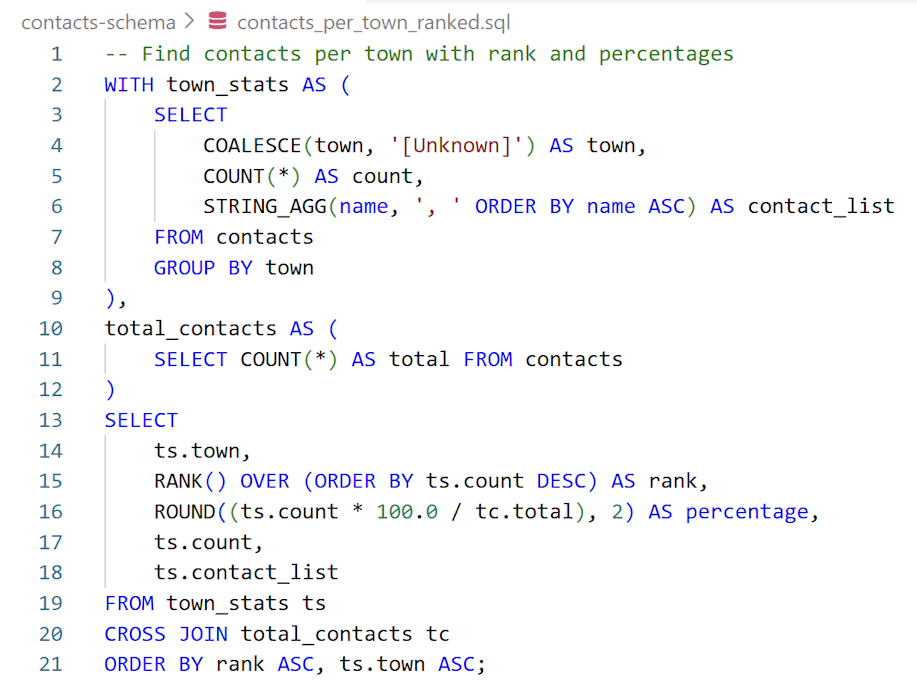


### Contacts per Town with Rank and Percentages

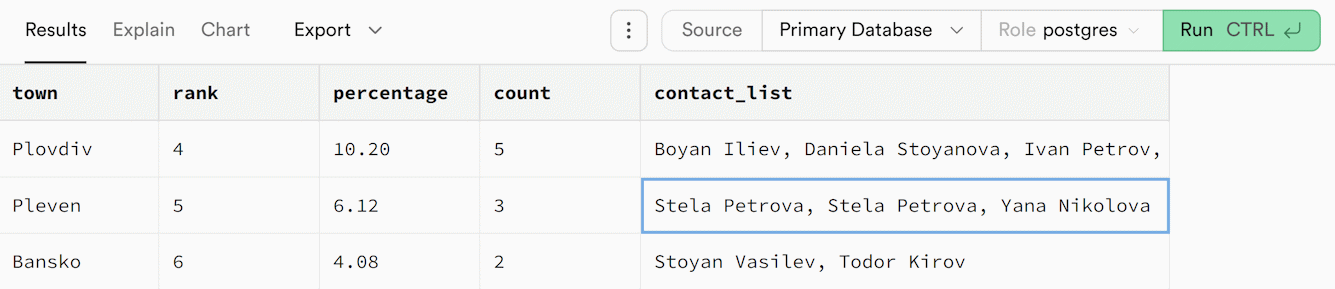
#### The AI Prompt



#### The Script



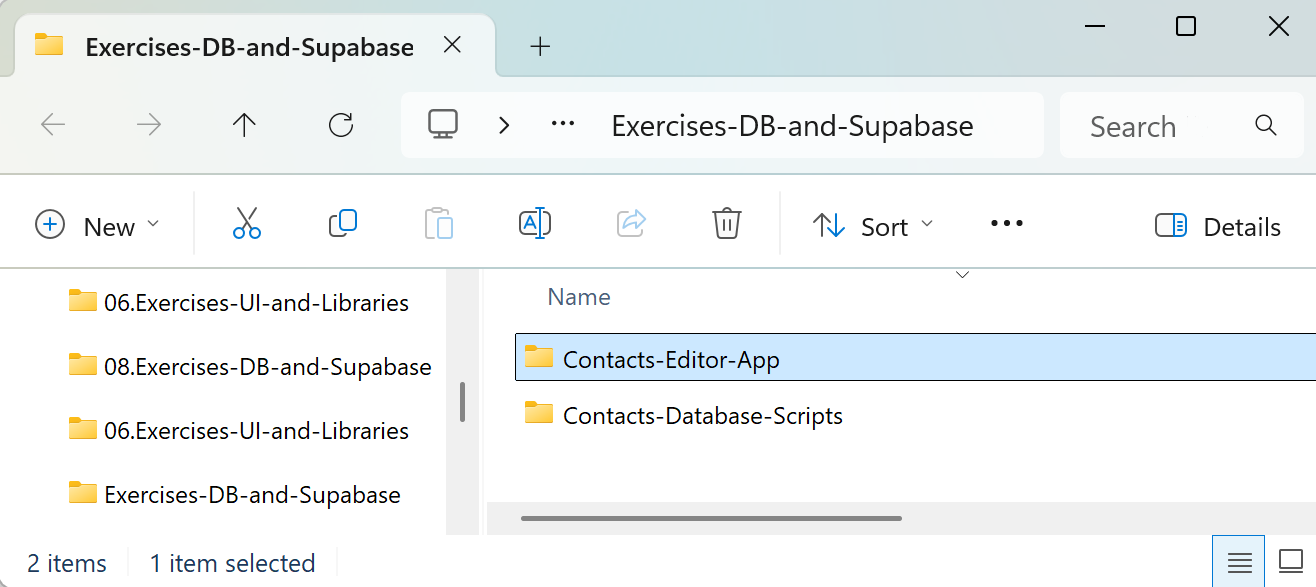
#### The Result

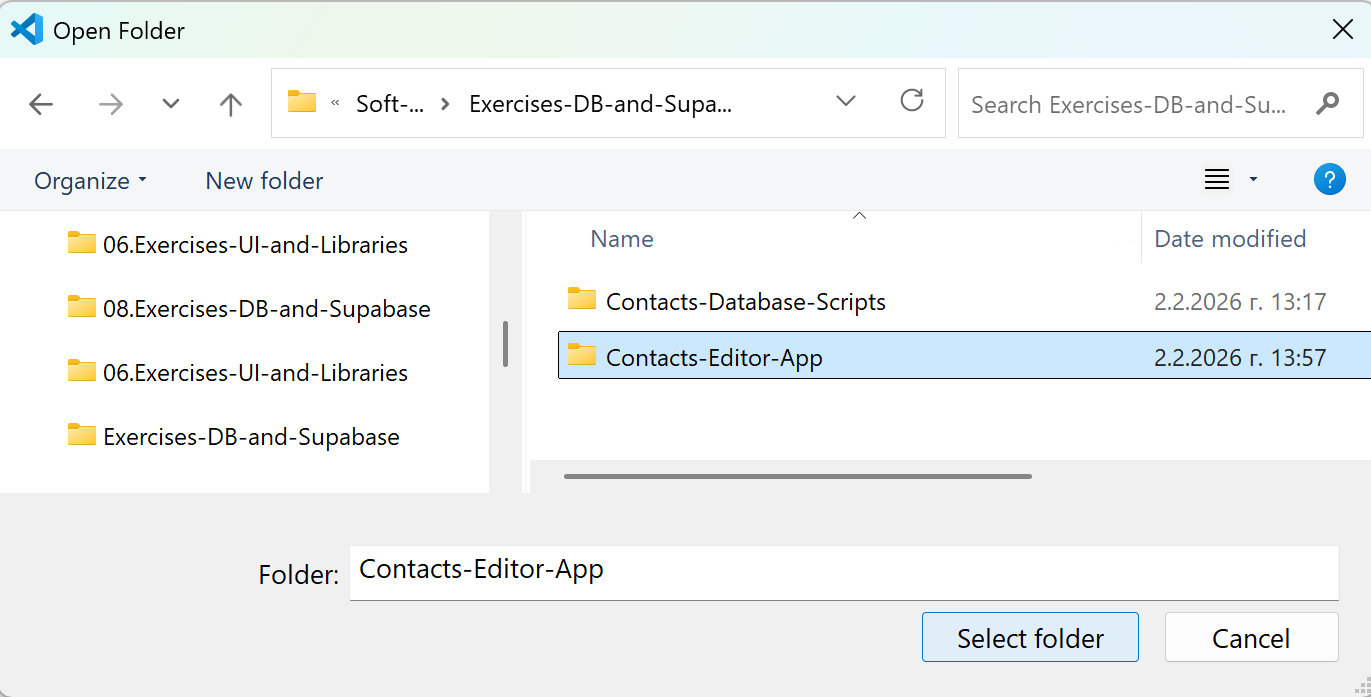


## Contacts Editor App

Your task is to **create a simple JS app to edit the existing contacts** **from the Supabase DB**. Do it **manually**, **without Supabase MCP**.

To avoid confusion with existing code and projects, **open a new Workspace in VS Code for this project**. Create a **new sub-folder** of the main folder for this exercise (Exercises-DB-and-Supabase (main folder for this lecture) -> Contacts-Editor-App (sub-folder to contain the new files which will be created during this task)):



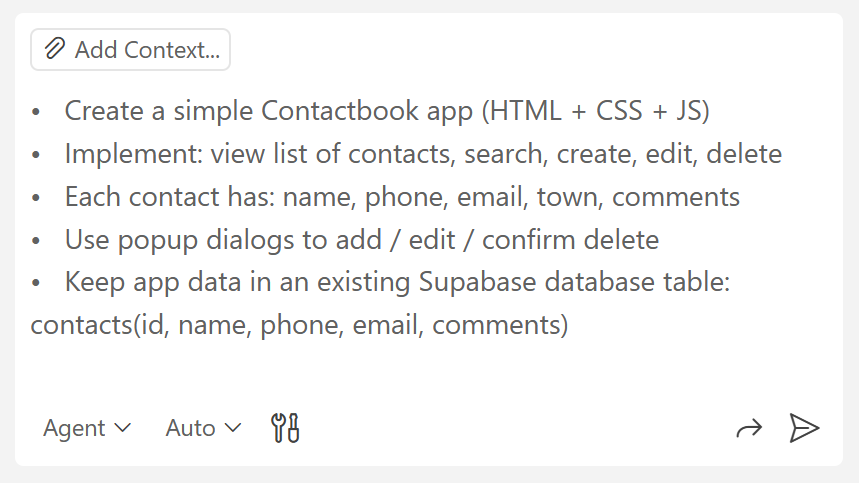


### Create the App

**Run** a **prompt in GitHub Copilot** (use Agent mode):

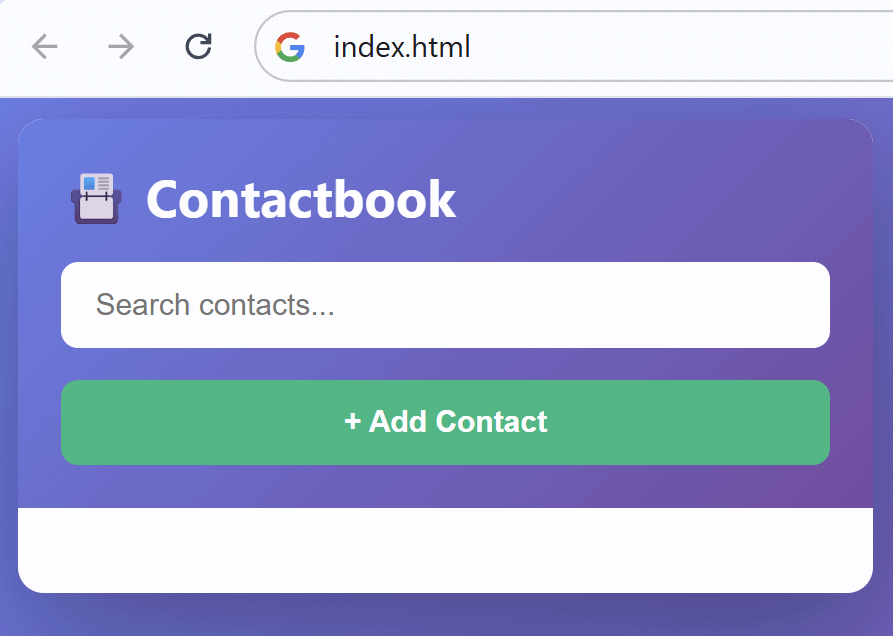
* + Create a simple **Contactbook app** (HTML + CSS + JS)
  + Implement: **view** list of contacts, **search**, **create**, **edit**, **delete**
  + Each **contact** has: **name**, **phone**, **email**, **town**, **comments**
  + Use **popup dialogs** to add / edit / confirm delete
  + Keep app data in the **existing Supabase database** table: contacts (id, name, phone, email, comments)

#### The AI Prompt



#### The App in the Browser

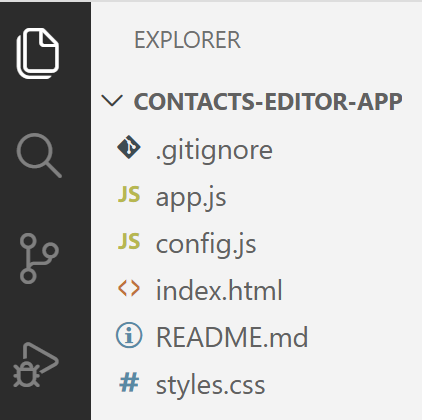
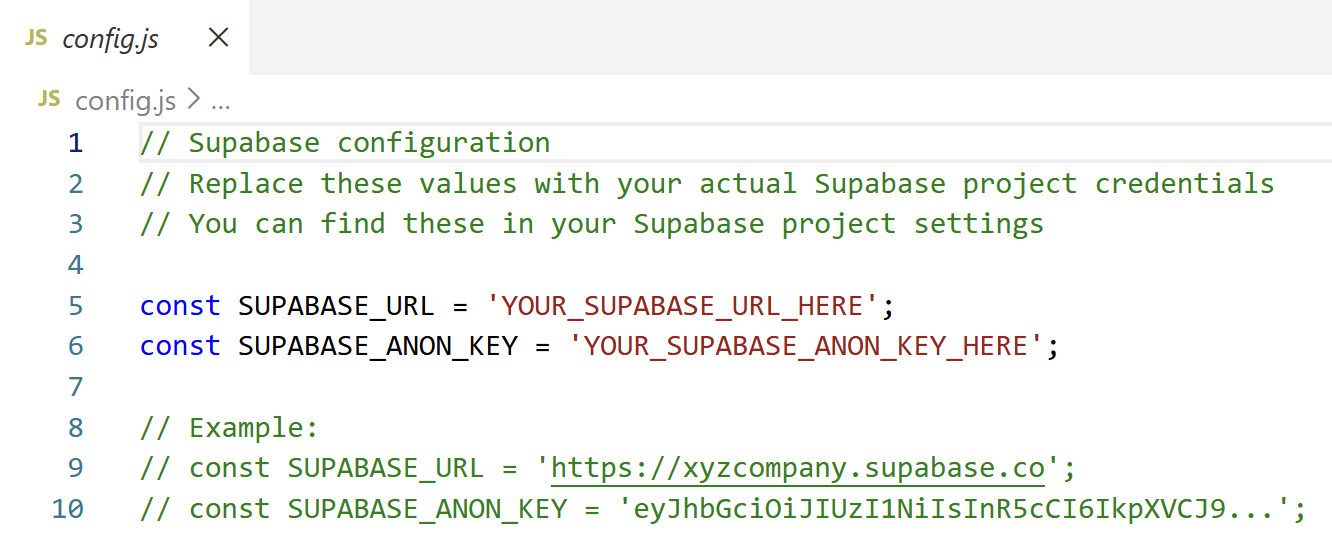
Right now, **the app is empty**. Contacts will appear when there is a **connection to the Database in Supabase**. Initially, the app will not run correctly:



**Insert the Supabase API Key**

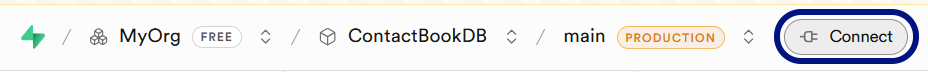
##### The Configuration File

**In Visual Studio Code** there should be a **file for configuration** and some instructions:

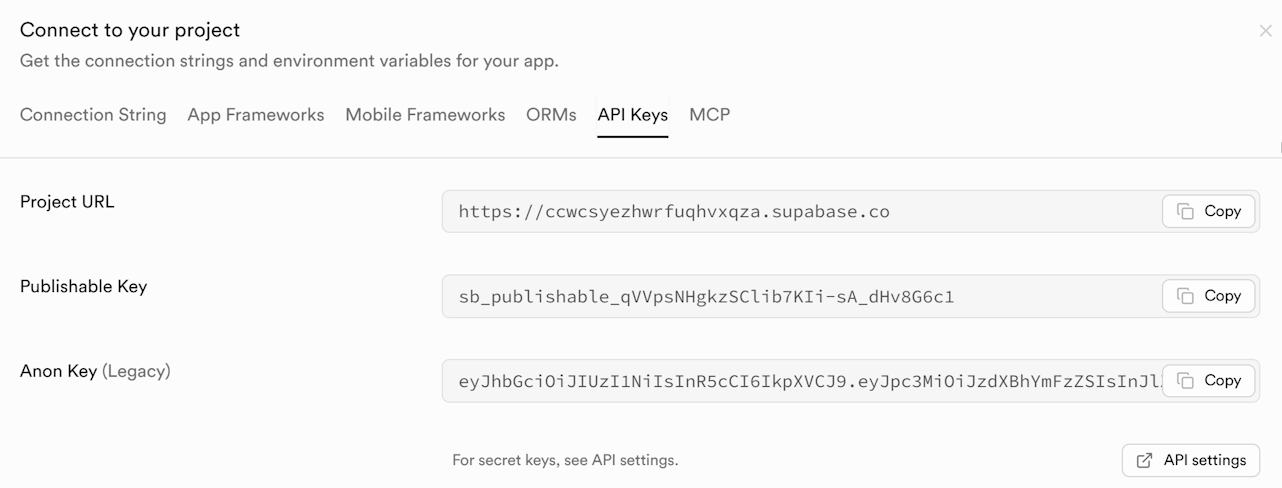
 

##### The Keys in Supabase

To get the necessary credentials, open your **database in Supabase** and click the [Connect] button **in the navigation**:

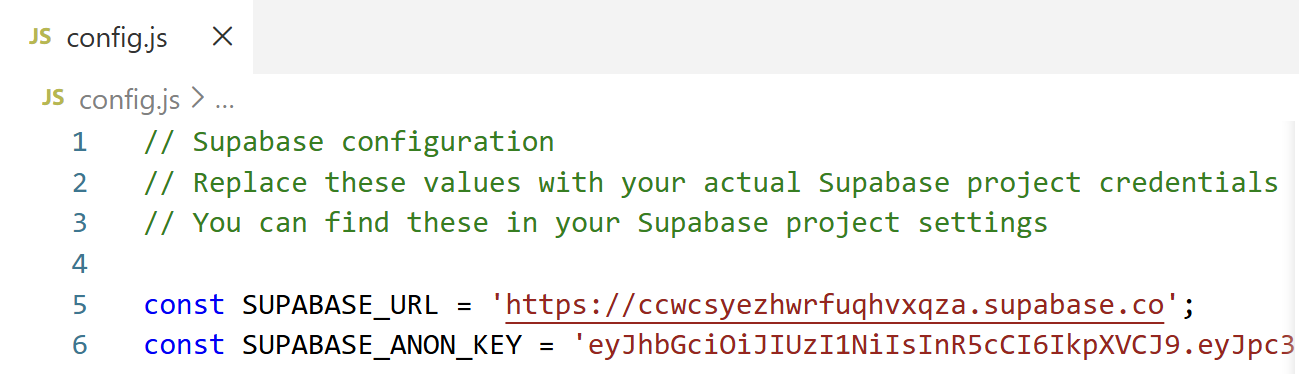


Go to the API Keys tab:



##### Linking the Database

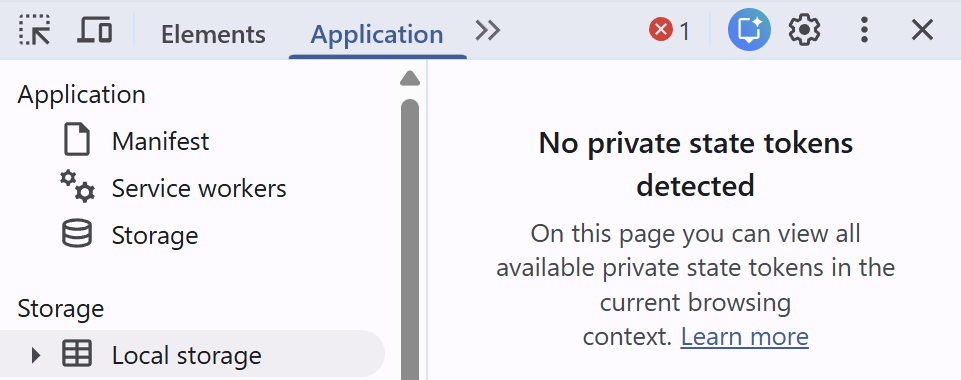
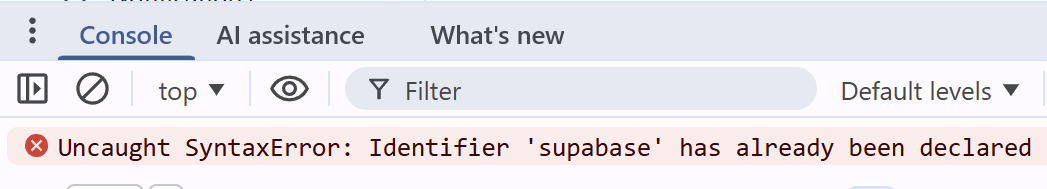
Now put the **Project URL** and the **Anon Key** in the configuration file:



##### Fixing Bugs

Run the index.html file and test if the app is working. You should be able to see the contacts from the database with the additional details from all columns. In any other case, there is a bug. To find what the issue is, open the index.html file in the browser and open the [Dev Tools] menu (with [F12] or [Ctrl] + [Shift] + [I]). Check if you see the following error:

Click the red [X] icon to see what the issue is:

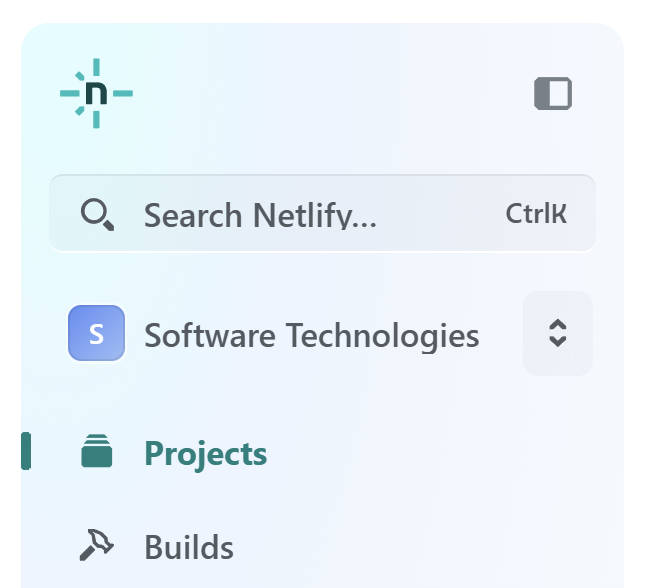
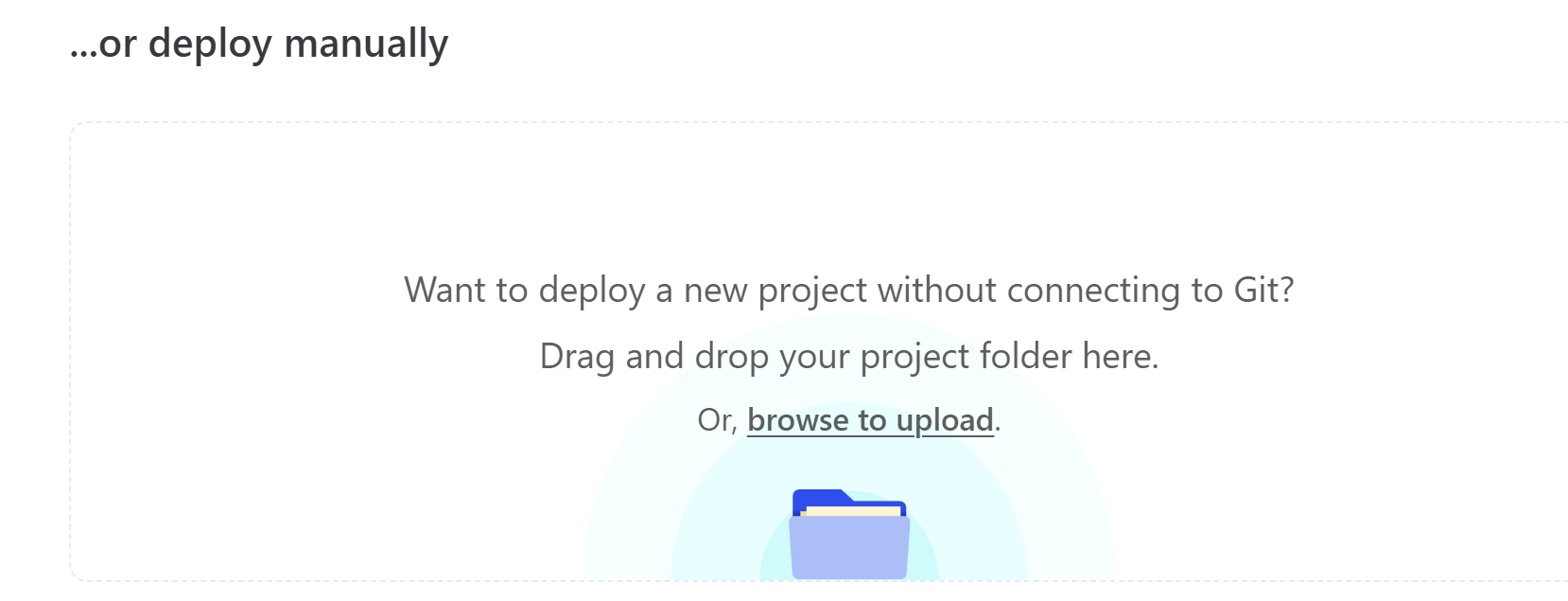
Copy the sentence or **paste a screenshot** in Copilot and ask it to fix it with a simple prompt.

***Note***: tell GitHub Copilot to **rename the `**supabase**` variable**. Otherwise, it may do stupid fixes, which will not work.

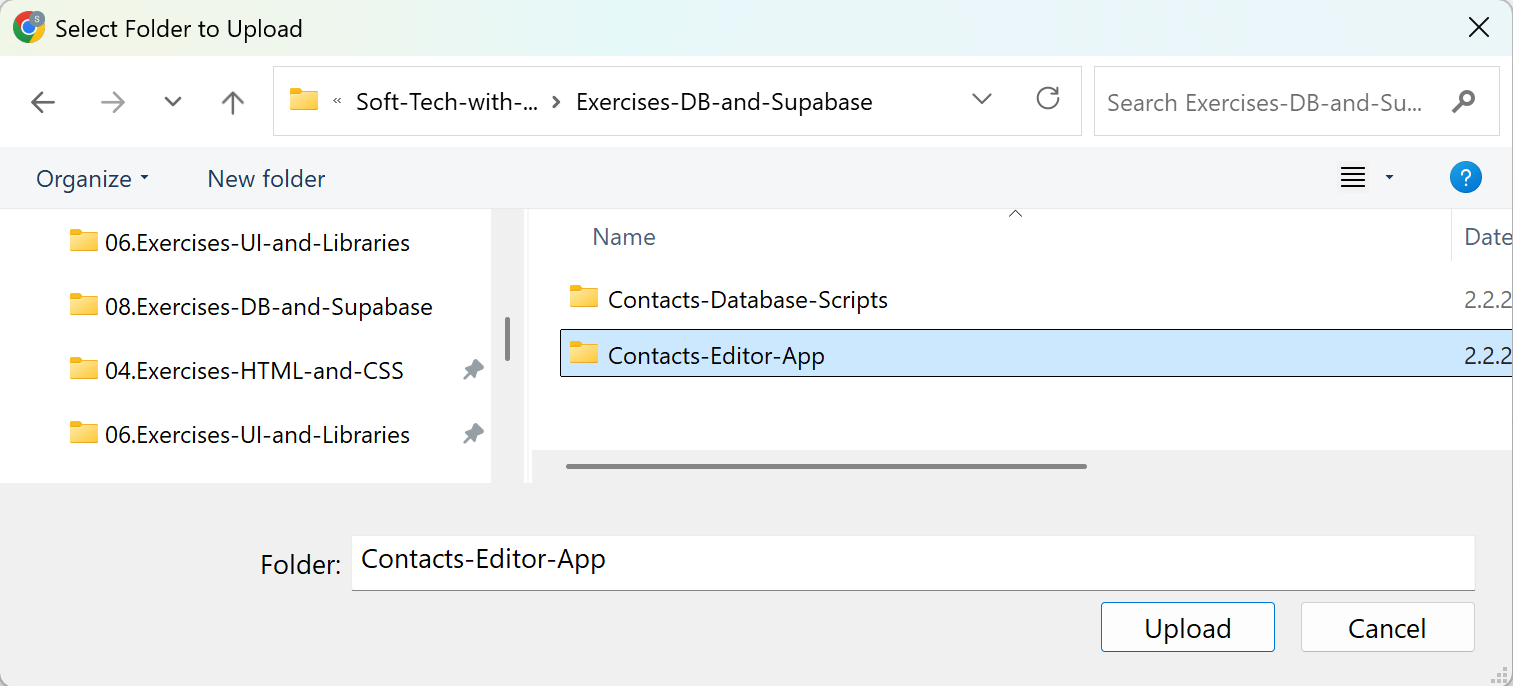
Fix any other bugs that may have appeared until you see the contacts display as expected.

### Deploy to Netlify

**Deploy your project to Netlify** manually:

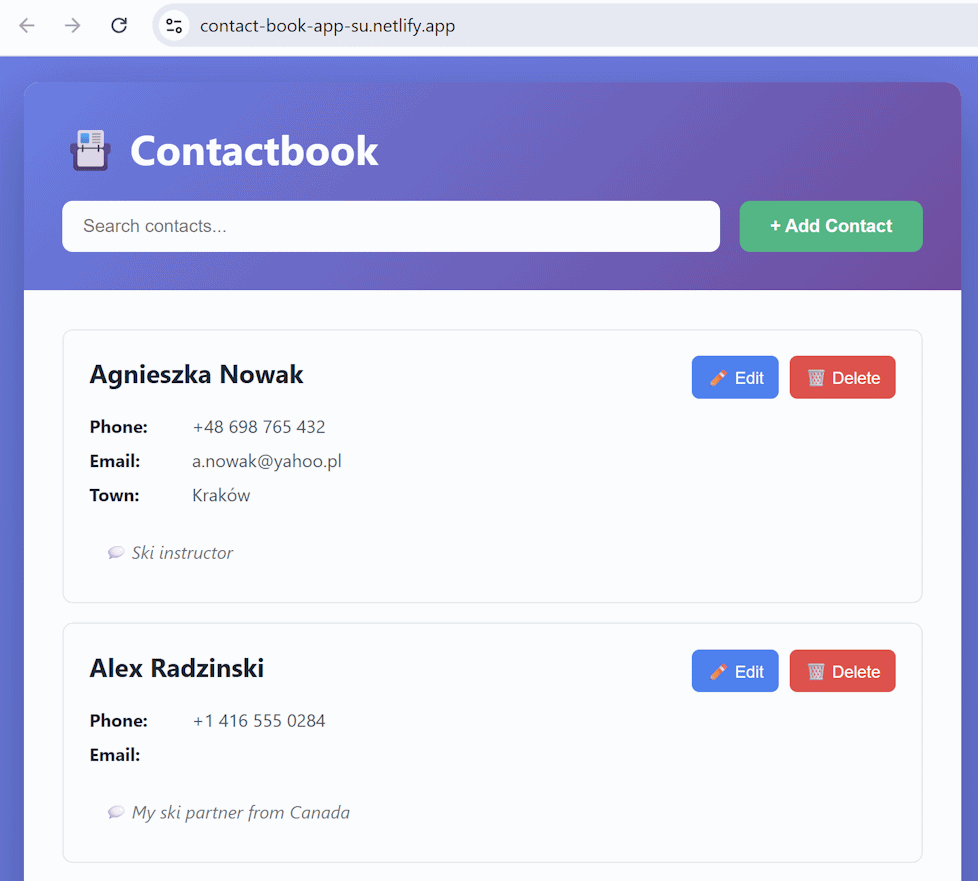
Select the folder with all the app files in it and click [Upload]:



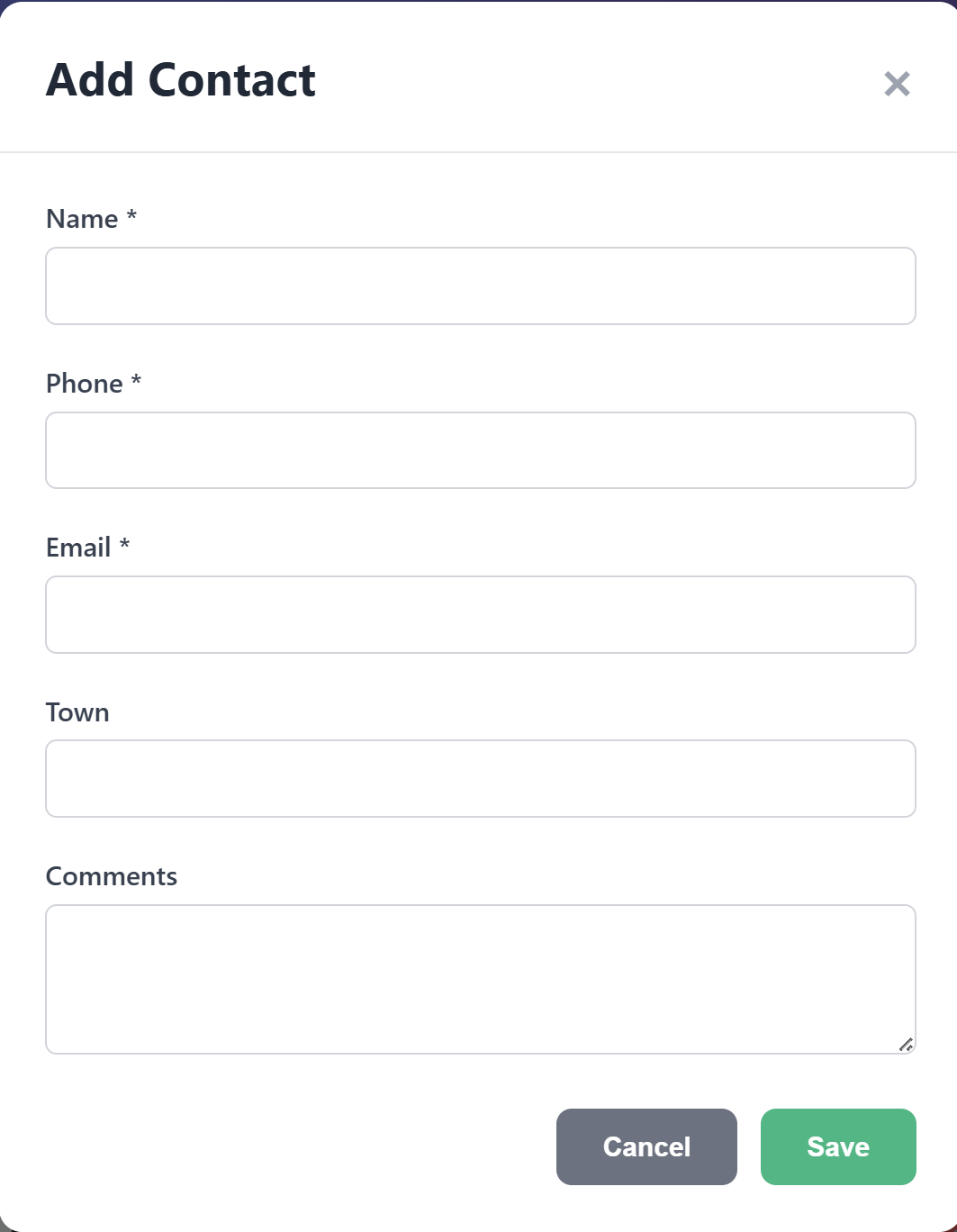
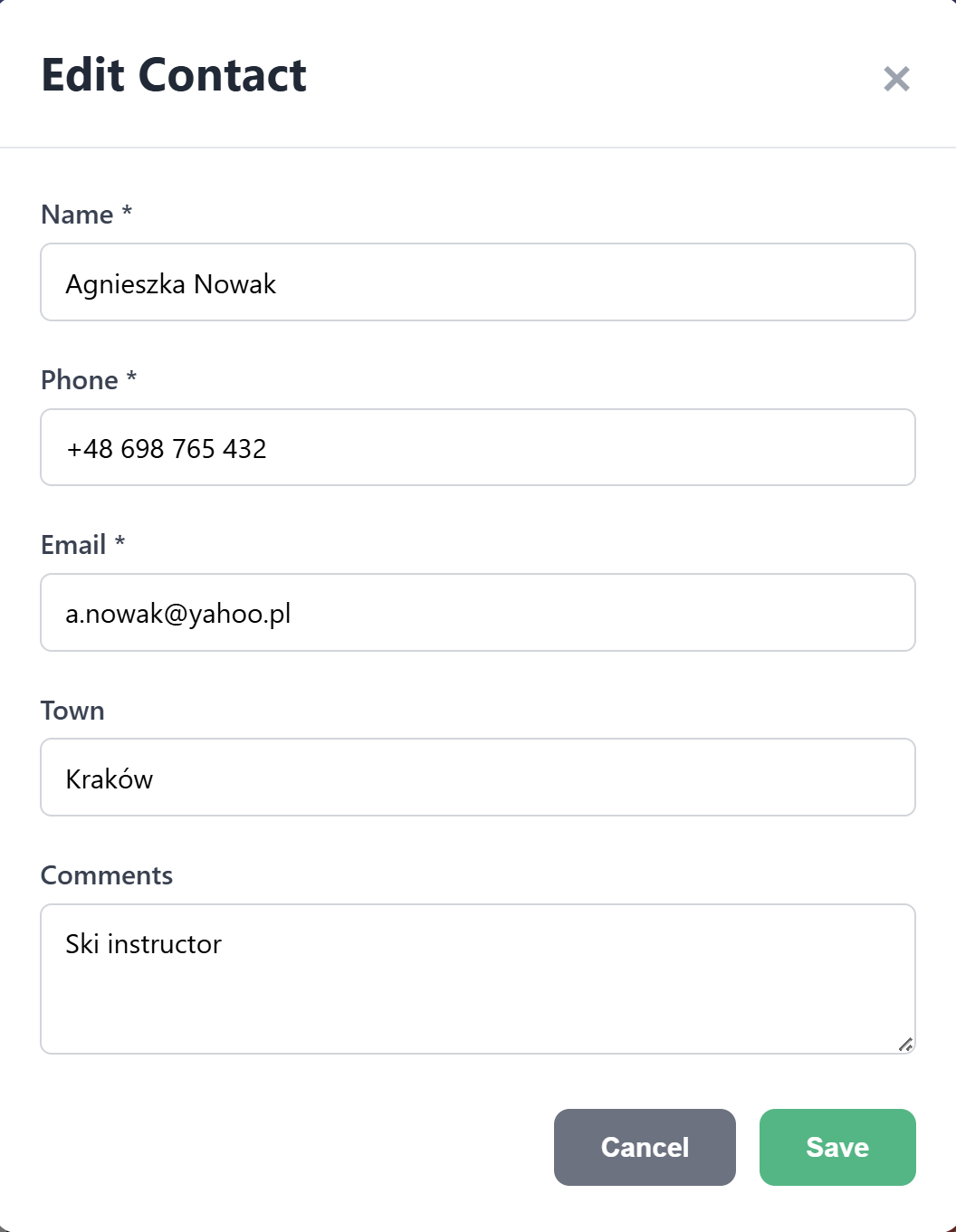
When the upload is ready, click on the [Quick Setup] button to change the project link and name to be suitable.

### The App in the Browser

This is an example of how your **Contactbook** **app** may look like:



#### The Add Contacts and Edit Contacts Forms

## Backup Database to SQL

In this exercise you will **back up your Supabase DB to SQL script**.

Why should you have backups? Because you can restore your DB and your business operations in case of data loss or corruption (e.g. accidental table drop or wrong DB migration).

### Install DBeaver

**DBeaver** is a **free, open-source, cross-platform** **database** **management tool** that lets you **connect to, browse, query, and manage databases** (like PostgreSQL, MySQL, SQLite, Oracle, and many more) through a graphical user interface.

In practice, people use **DBeaver** to:

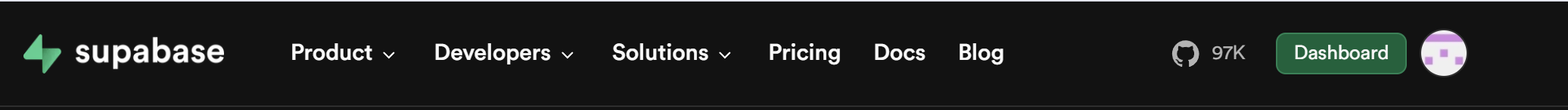
* Connect to databases (including Supabase/PostgreSQL)
* View and edit tables and data
* Run SQL queries
* Export / backup databases to SQL files
* Import / restore data from backups

**Install DBeaver** (community edition): <https://dbeaver.io/download/>:

### Connect DBeaver to Supabase

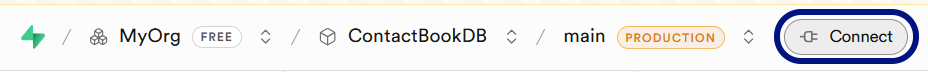
**Connect DBeaver to your Supabase database**. Follow these steps:

* **Supabase** -> **Dashboard** -> Open your **current project**

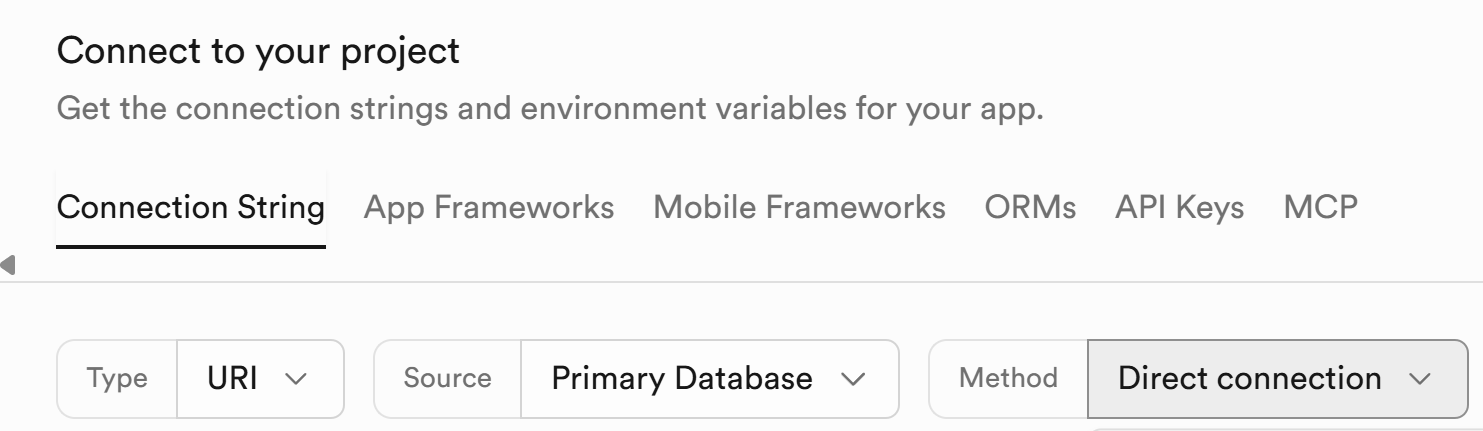




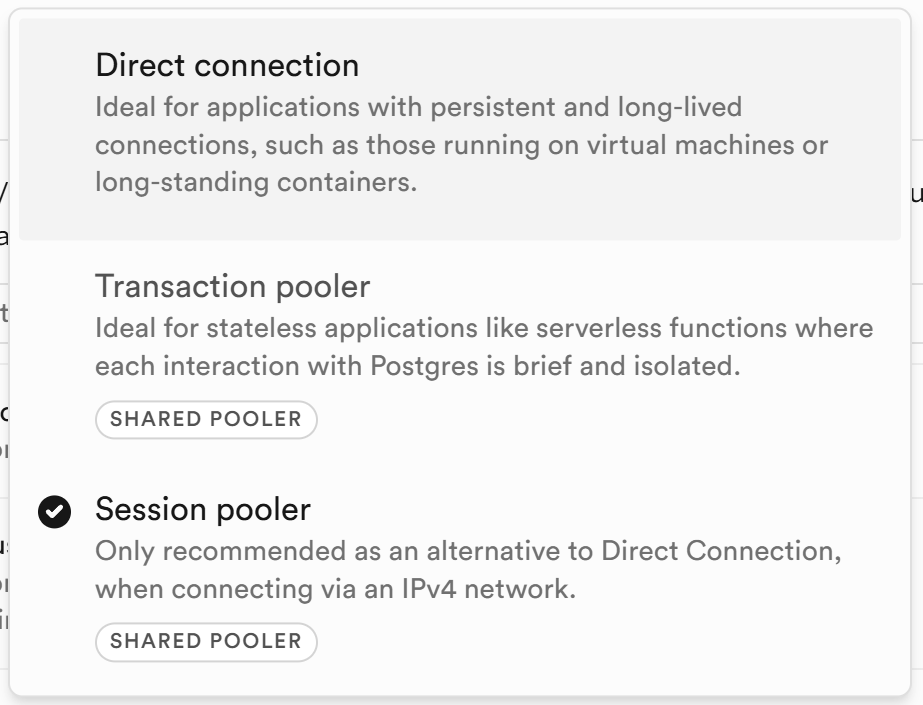
Select **Connect**:



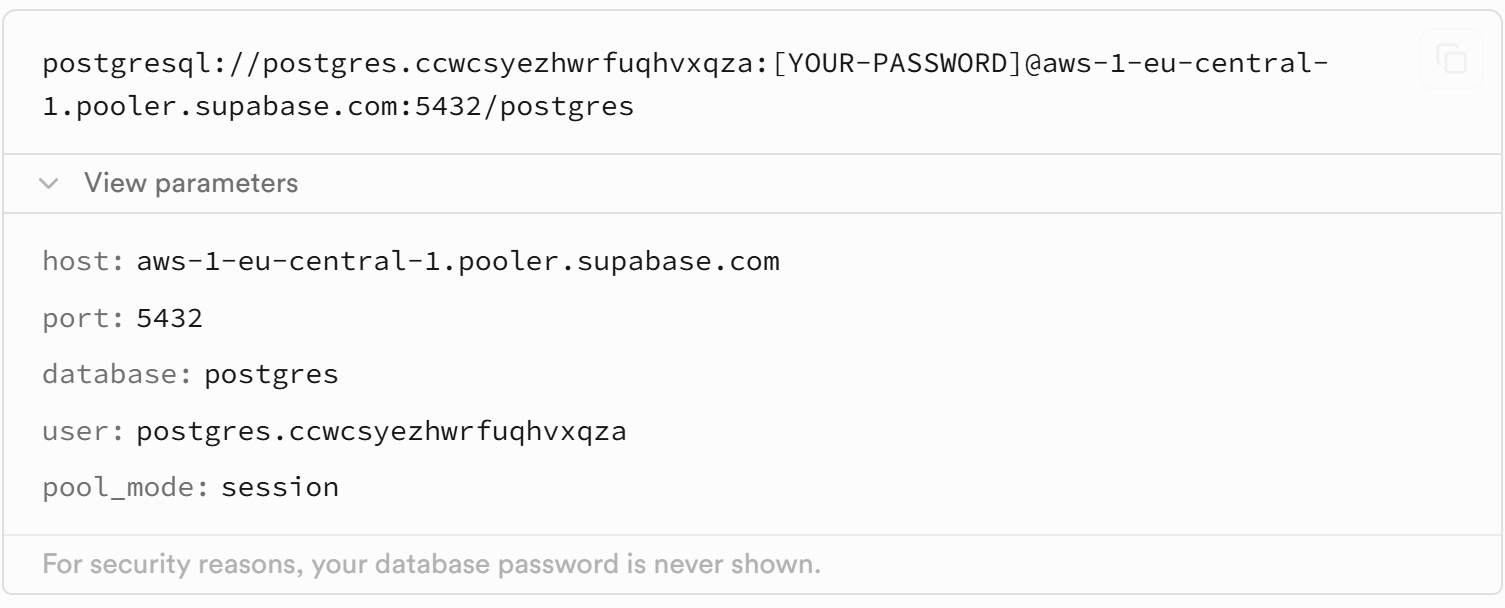
**Connection String** -> **Direct** **connection** -> **Session pooler**



Select [Session pooler]:

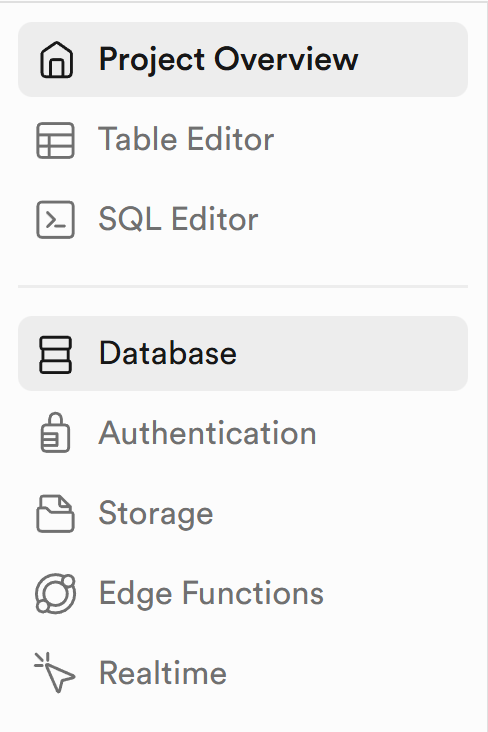
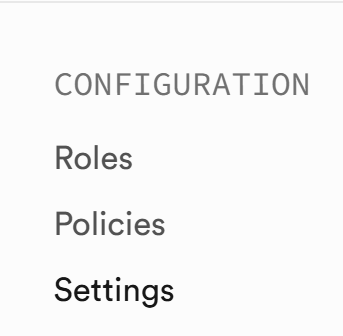


Click **View parameters**:

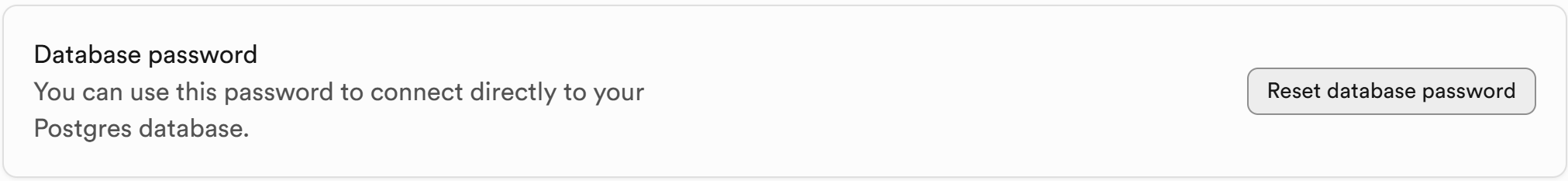


You will need your **Supabase DB** (project) **password**. You can **reset** it here:

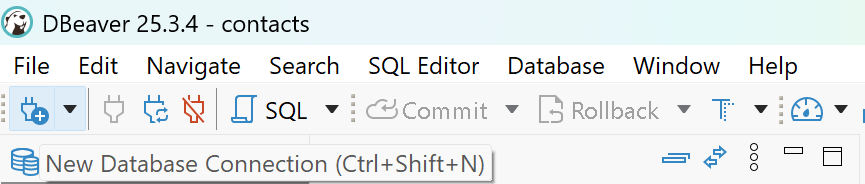
**Supabase** -> **Dashboard** -> **Database** -> **Configuration** -> **Settings**

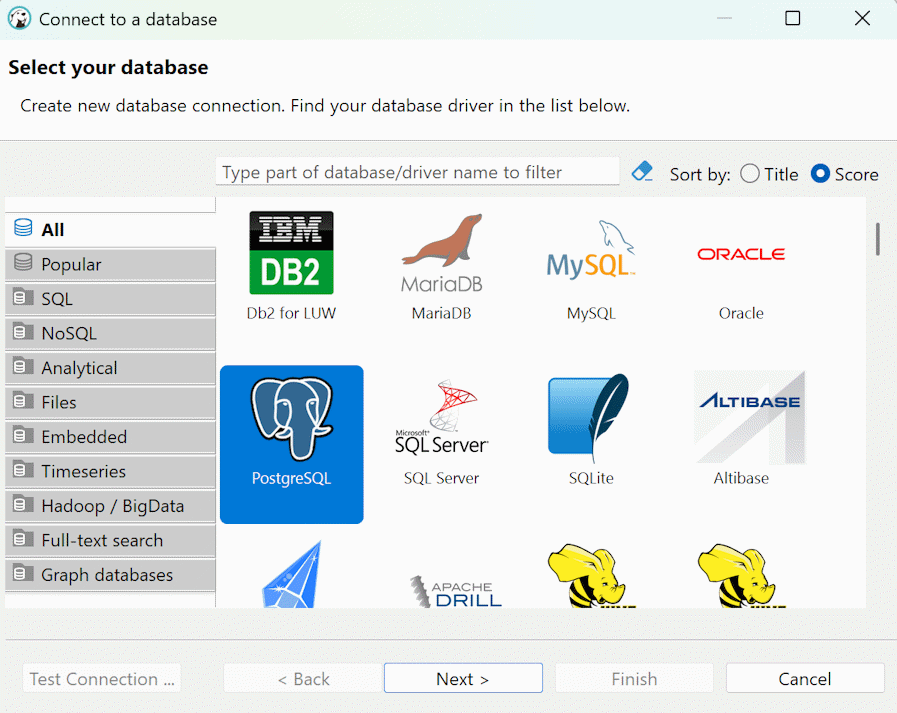
Click [Reset Password]



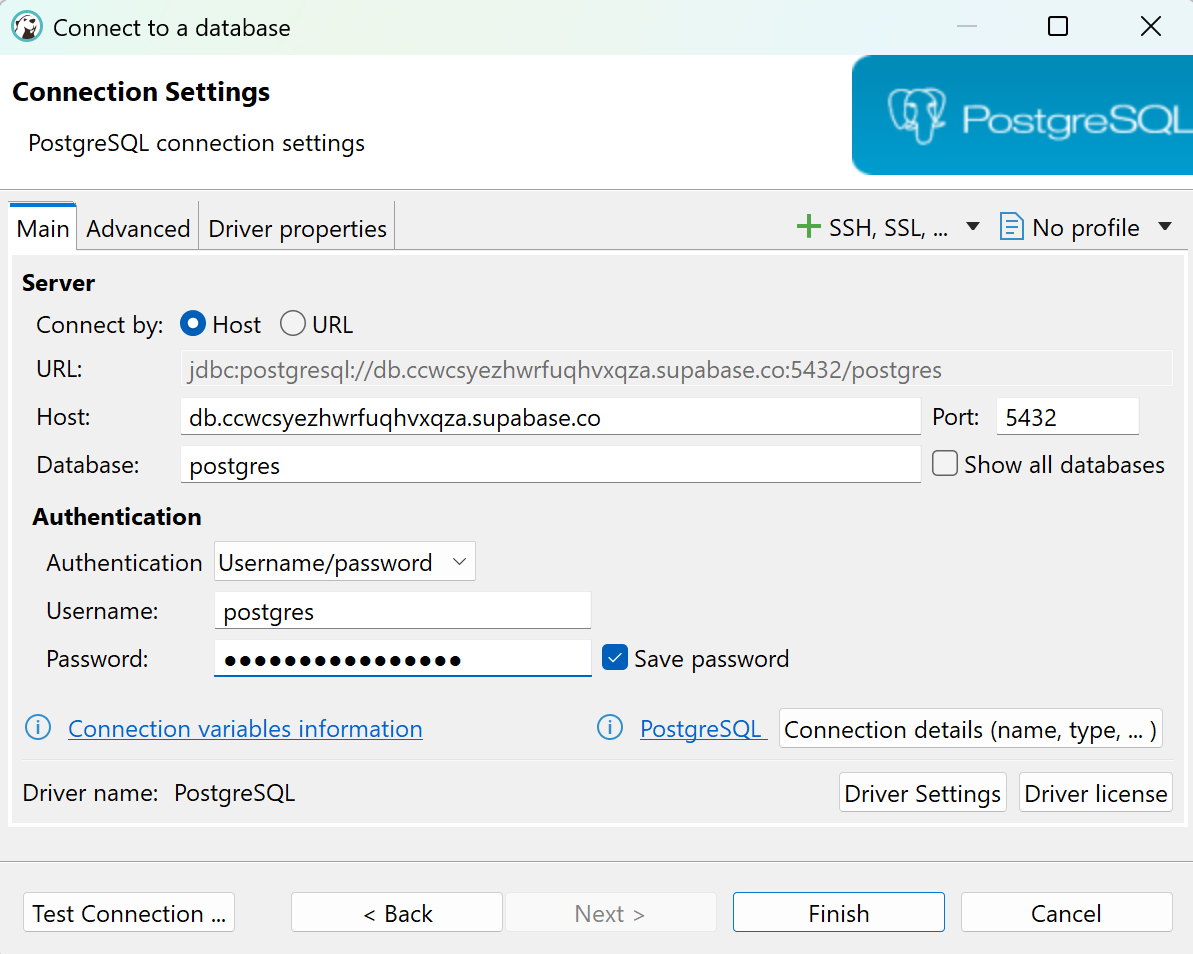
Open **DBeaver** and select the [New Database Connection] icon:



Choose **PostgreSQL**:

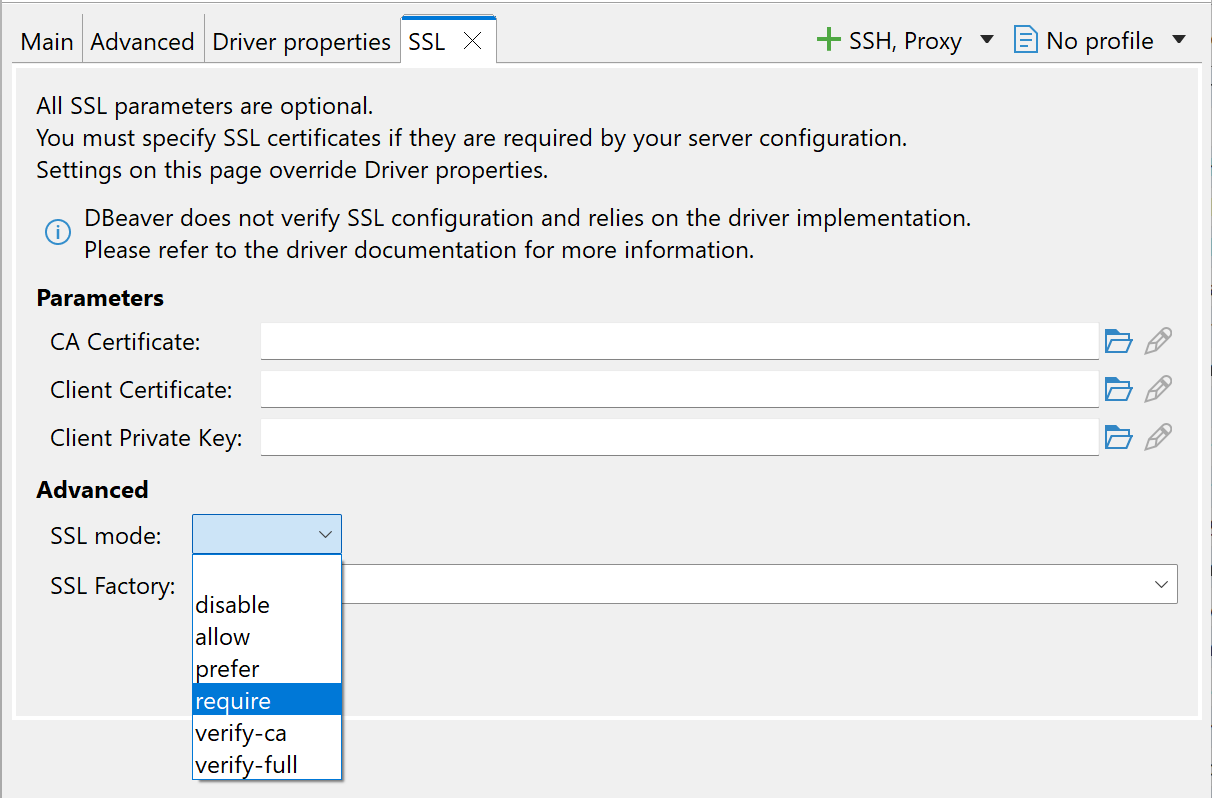


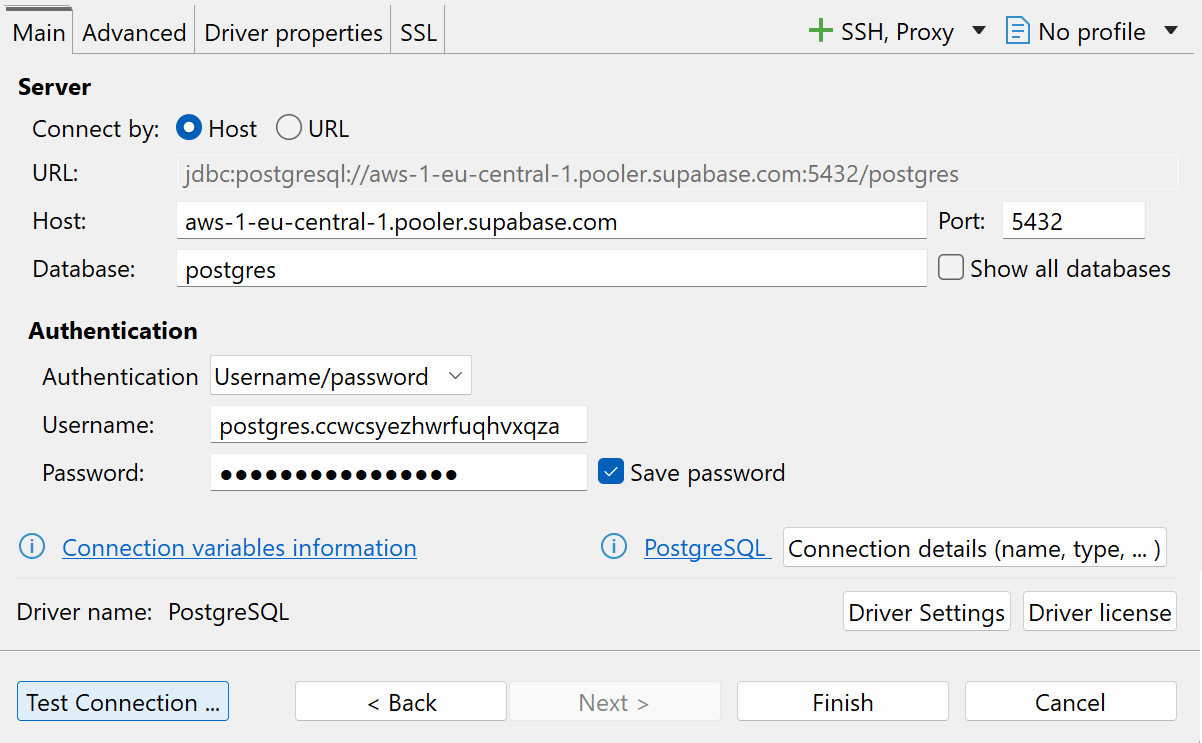
**Insert the information** you checked in Supabase in **Connection String** -> **Session Pooler**:

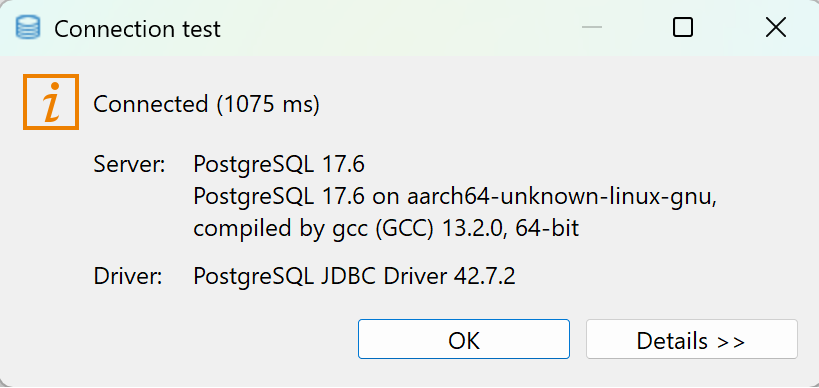


**Add SSL**:



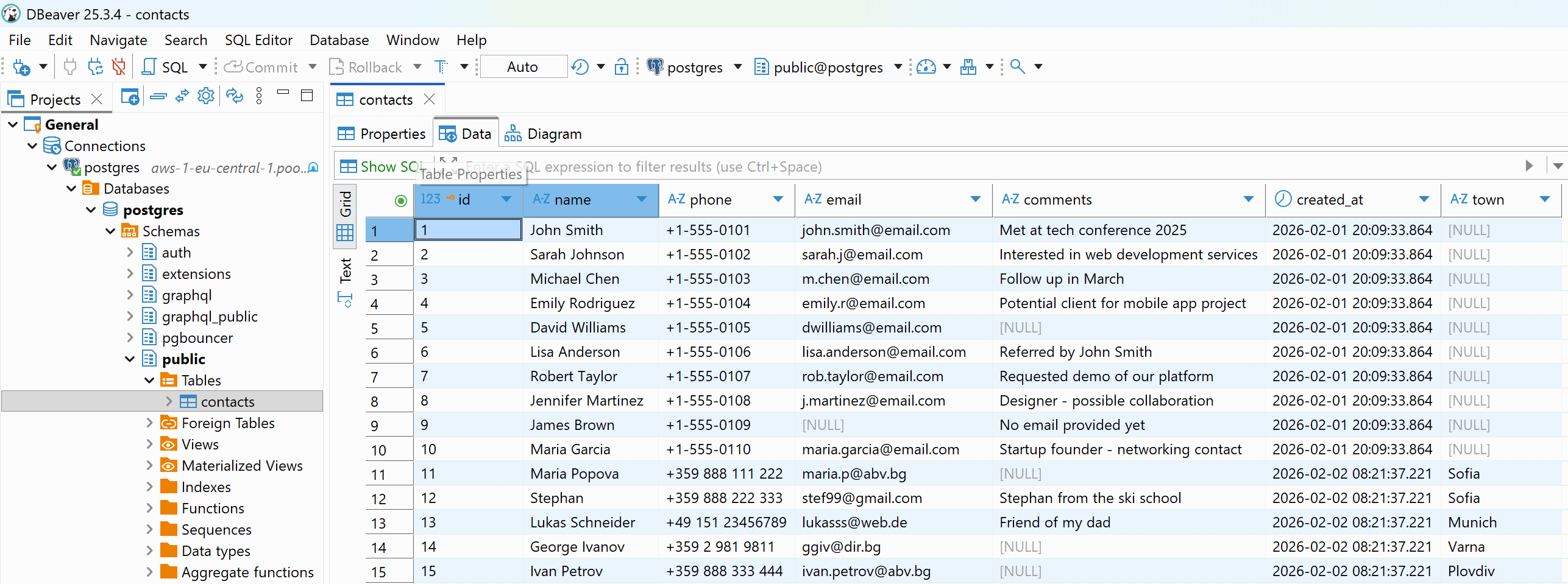
Open the **new SSL tab** and select [Require] for **SSL mode**:  


Return to the **Main tab** and click the [Test Connection] button:  


If everything is **correct**, you should receive the following result:  


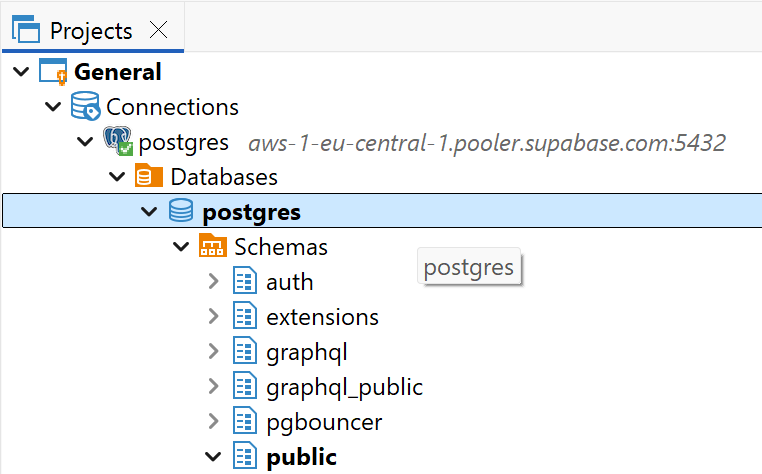
Click [OK] and then [Finish].

Now open the folders until you reach **the contacts table** and **double click it to open it**:

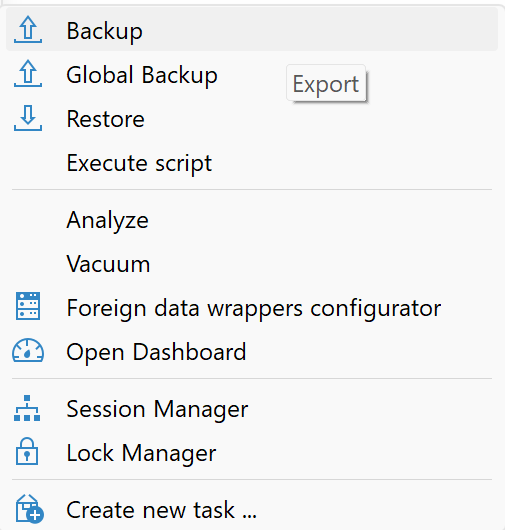


### Backup Your DB as SQL Script

We shall **export the schema** (table definitions and other DB objects) + **data** (table rows) for the following schemas: **public** (app tables), **auth** (app users and authentication data).

In DBeaver right click the postgres:  


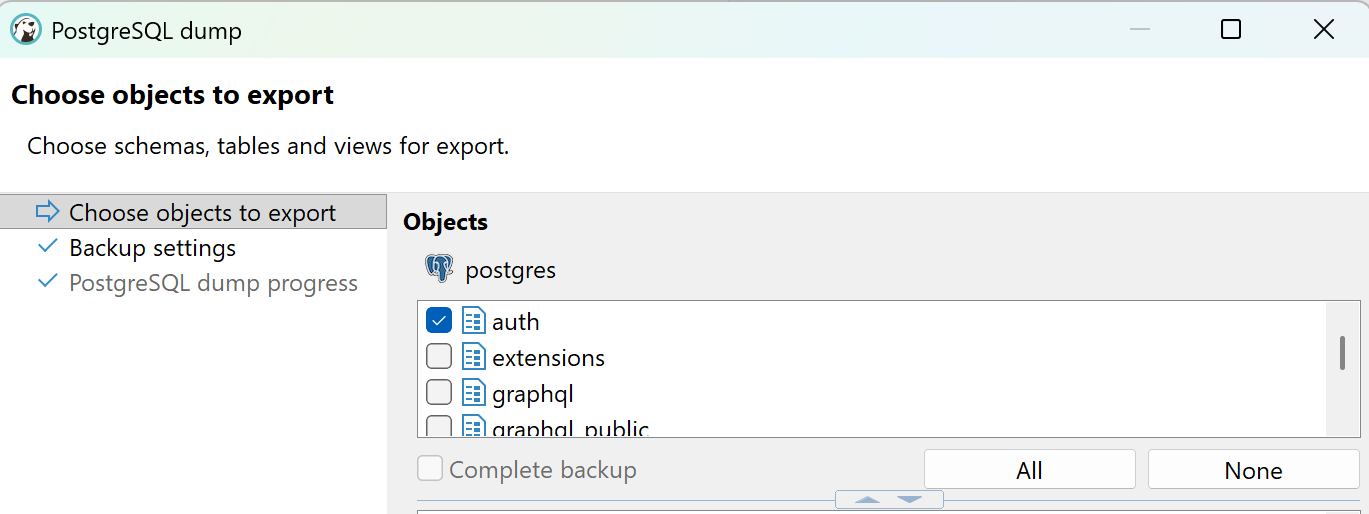
Select **Tools**, then **Backup**:

Your **app data** stays in the `public` schema (your tables and data inside these tables).

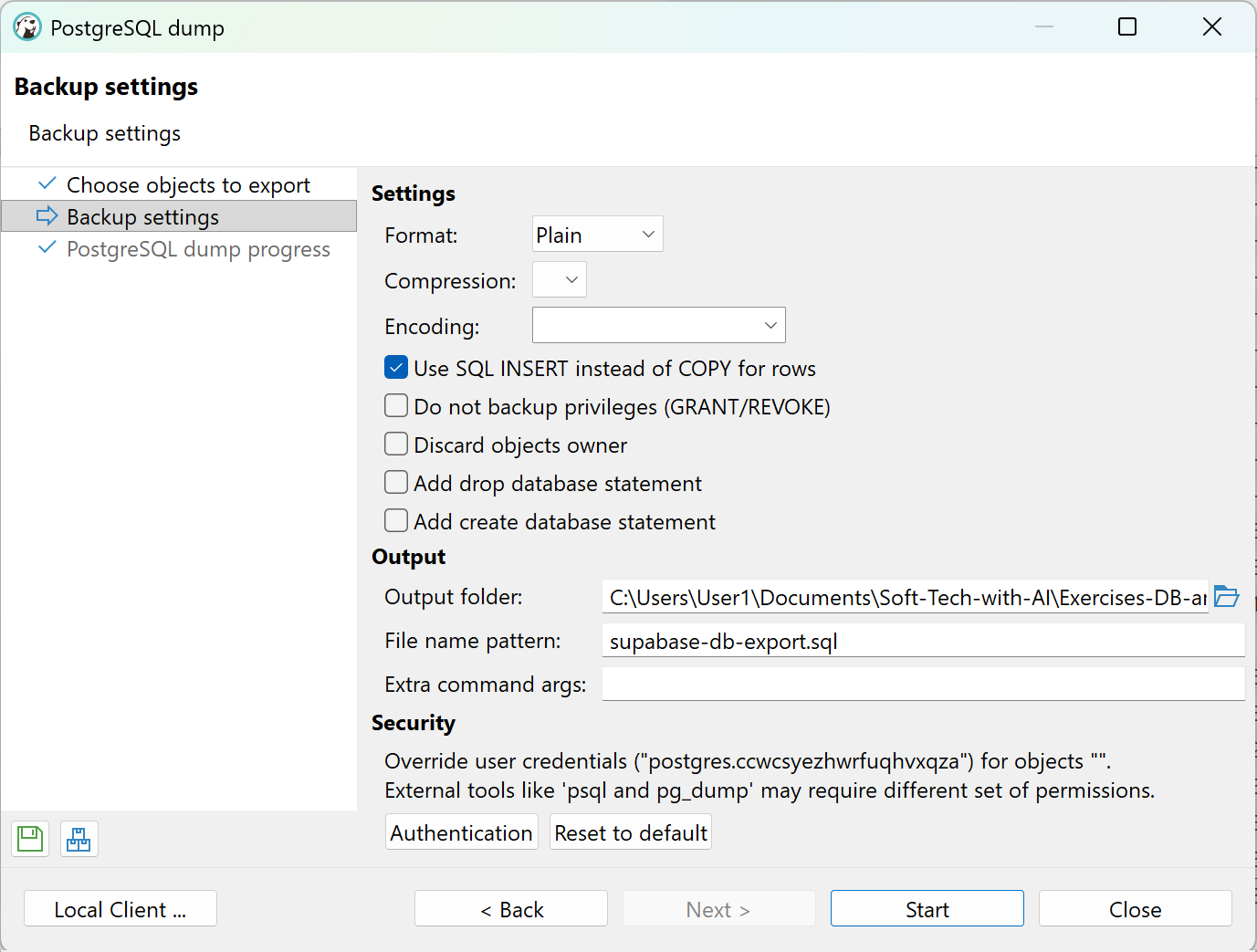
Your **app users** stay in the `auth` schema (users, passwords, auth methods, etc.)

Select only "public" and "auth":



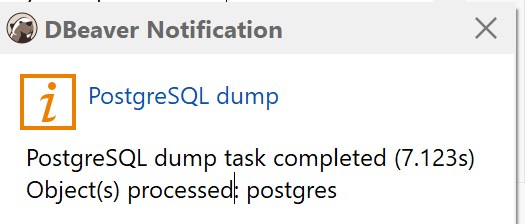
Select [Next].

Then set the **format** to be [Plain] and check the [Use SQL INSERT] checkbox. Select a proper name (e.g. `supabase-db-export.sql`):

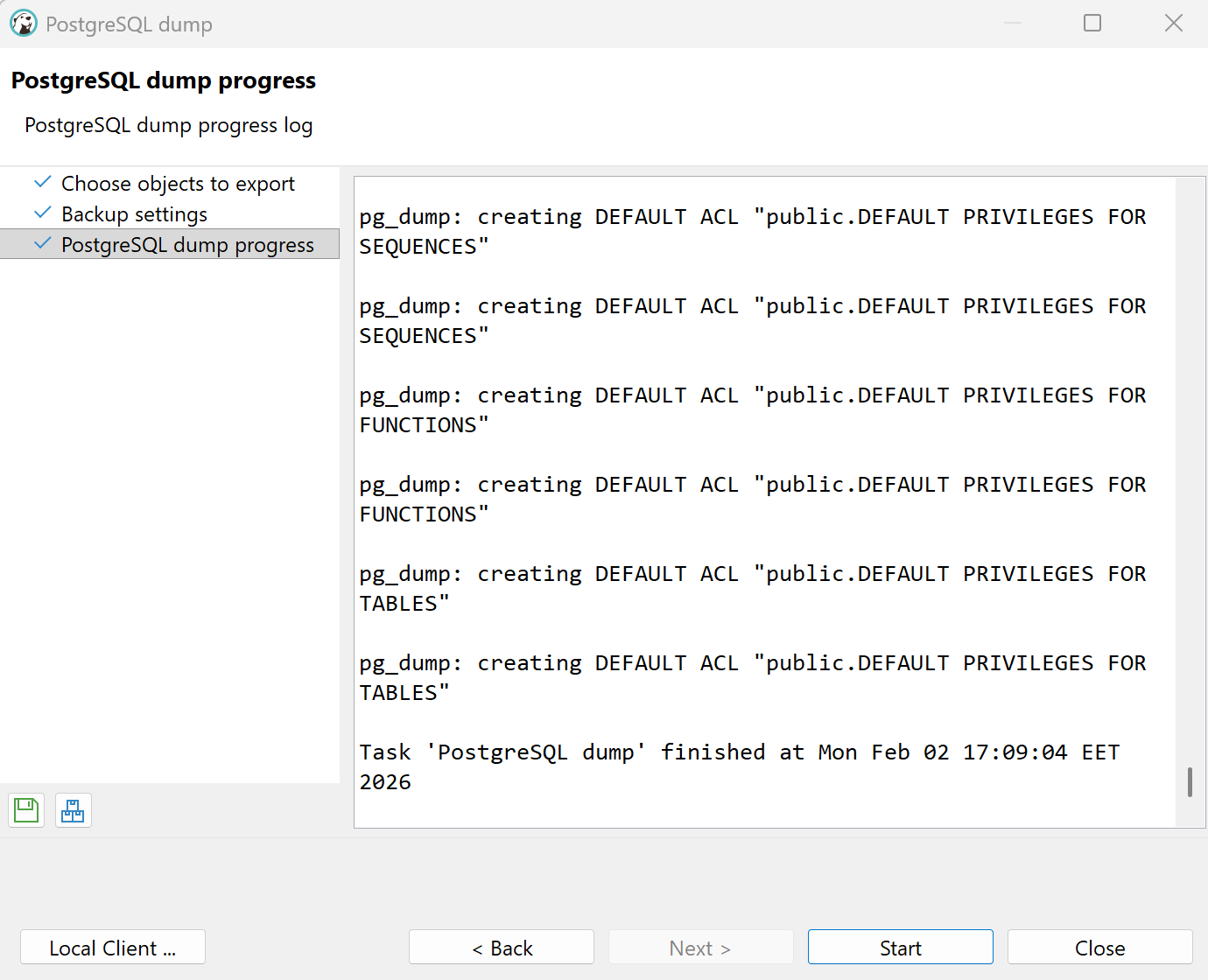


Select [Start].

You will receive a notification like this one:



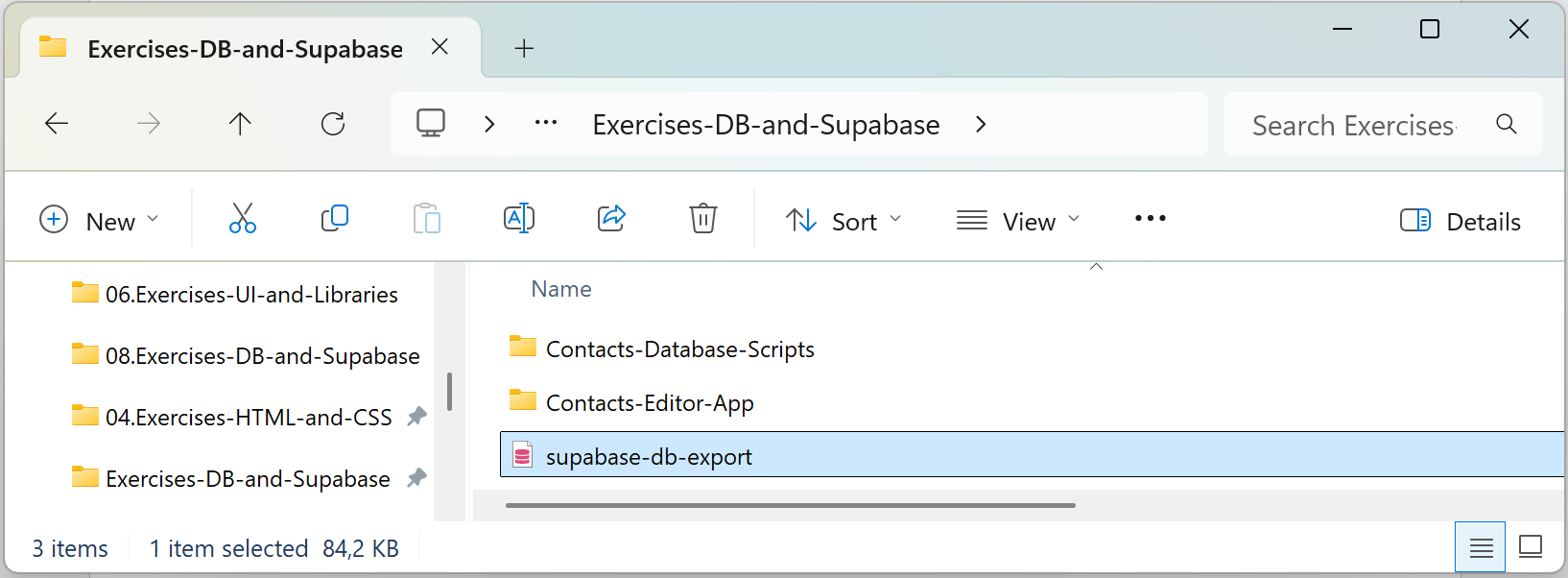
The backup processing screen will look like this:



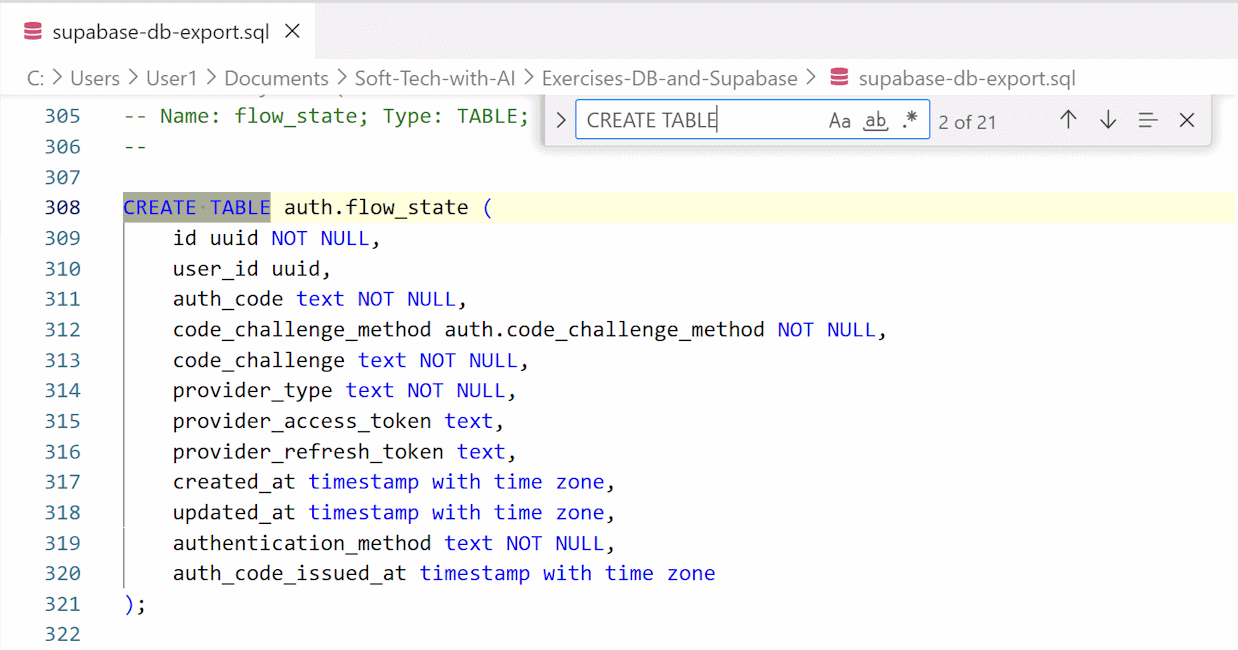
This means that it’s been successful and you can click [Close].

### View the SQL Script

Now we have **the SQL database backup script**, created by the pg\_sql export tool executed from **DBeaver**. Open the file in the folder you selected in the Backup settings with **Visual Studio Code**:



It should hold the data definition commands [CREATE TABLE] and data insertion commands [INSERT].

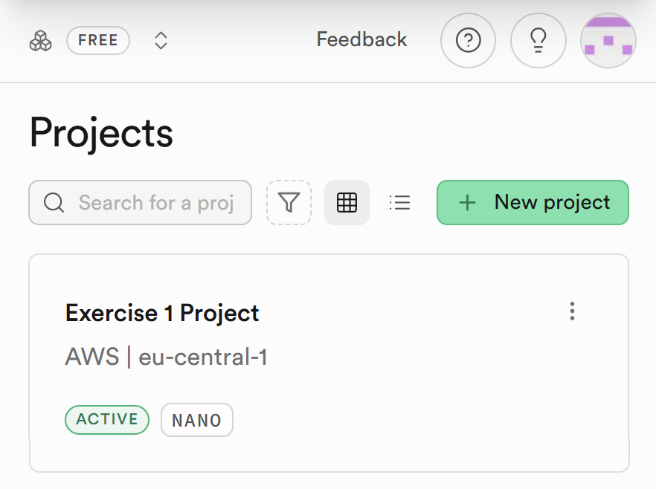


## Restore a Database from SQL

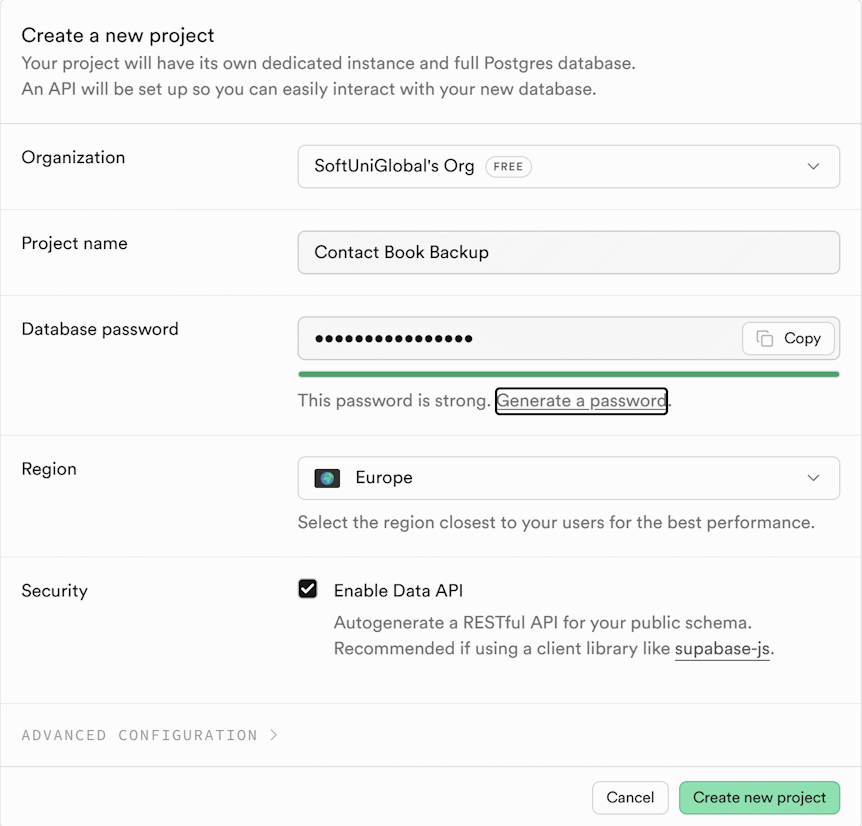
In this exercise we shall **restore a backup of Supabase DB** into a **new Supabase project**.

### New Supabase Project

Create a **new project**:



Fill in the details:



### Connect DBeaver to the Supabase Project

Connect the **new project** to **DBeaver** like you did the first time. Follow these steps **in Supabase**:

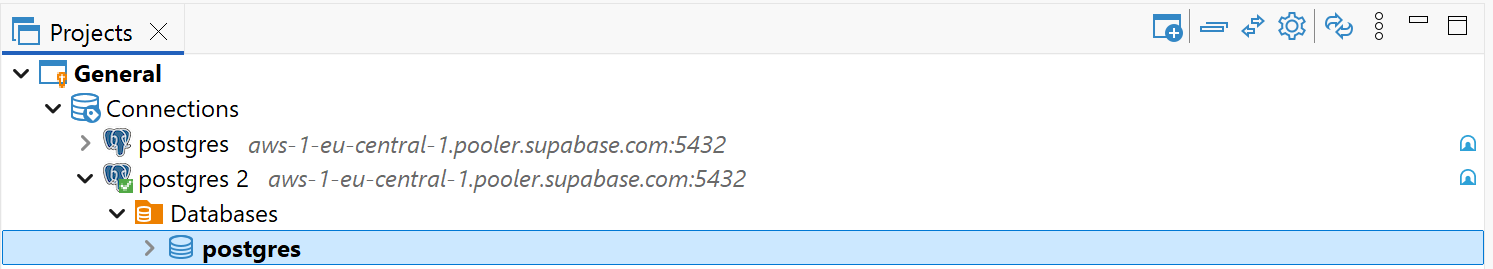
Open **Supabase** -> **Dashboard** -> **Connect** -> **Connection String** -> **Session pooler** -> **View parameters**

Then open **DBeaver** and connect it to the **new Supabase project** via the **parameters** and **the password** (The new project has a different password).

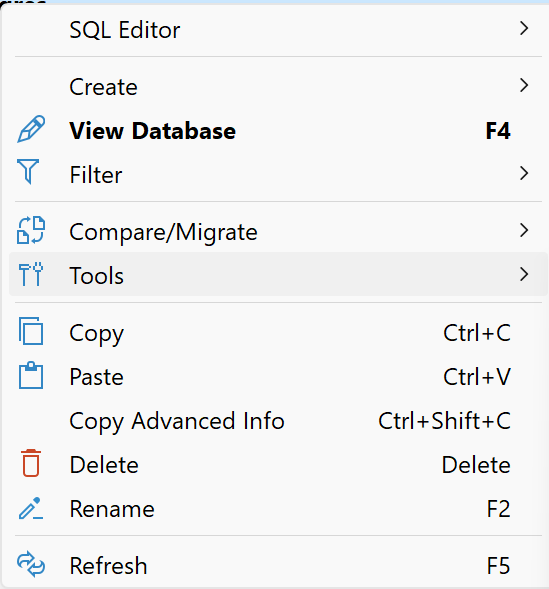
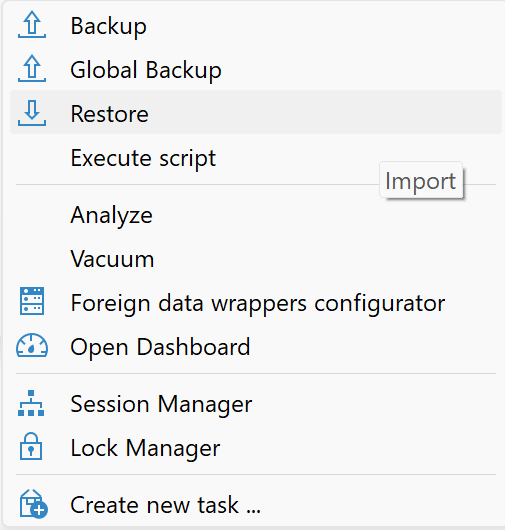
When the connection is ready, the next step is to **restore** **the** **database** **to the new project**.

### Restoration Way 1

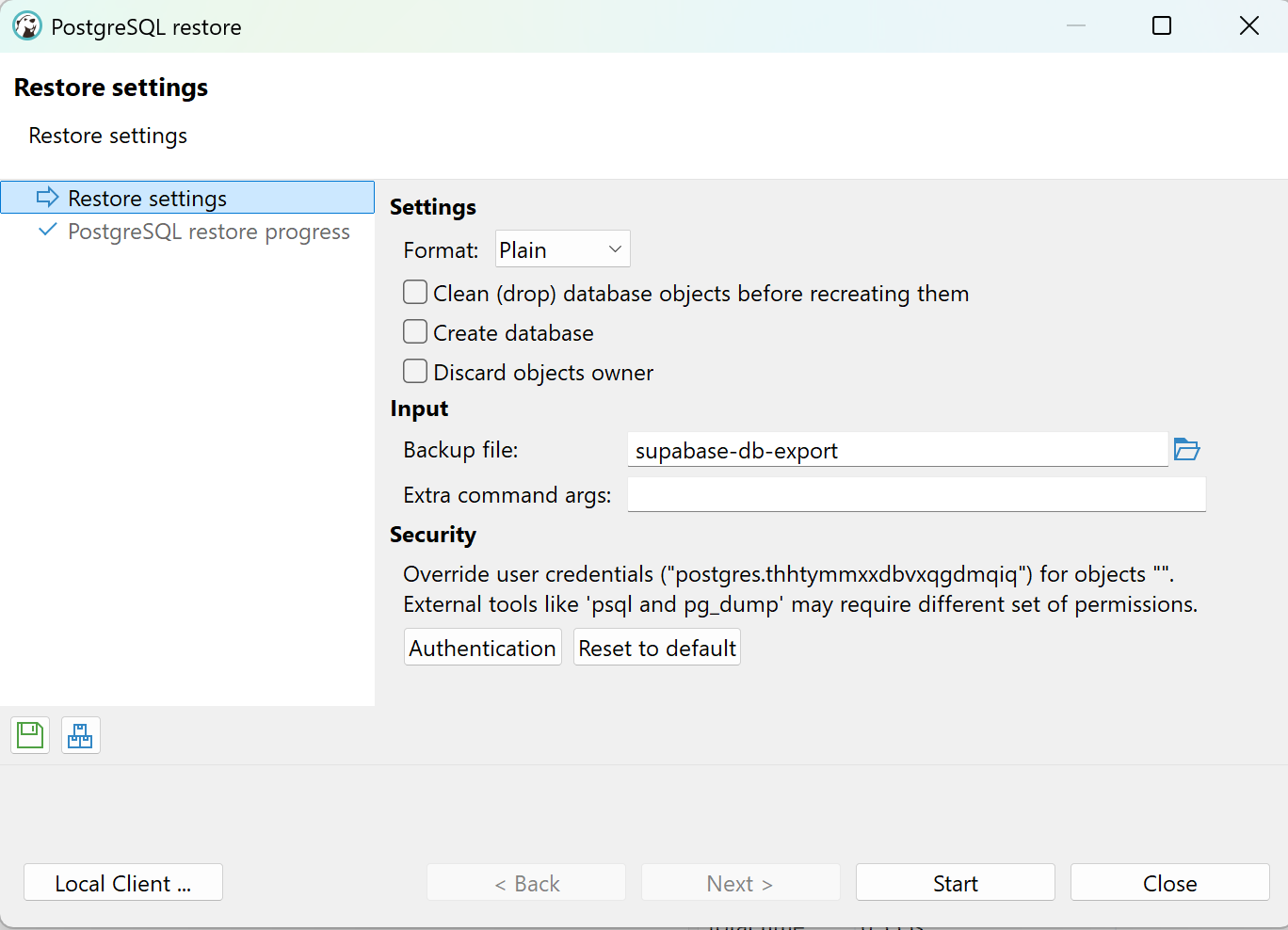
Open **DBeaver** -> **Databases** -> postgres



Then right click and select **Tools** -> **Restore**

Set the **Format: Plain** -> select your **SQL script** by browsing:



Click [Start].

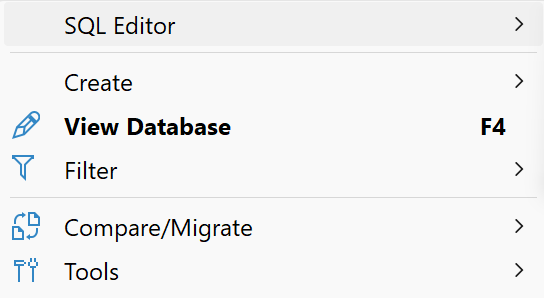
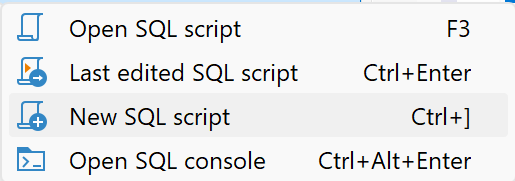
**Note**: in a **managed PostgreSQL** instance you have **limited access permissions**, so **you will get errors** when you restore your schema and data, but still, DB restoration will be **successful**.

Check if the restoration is successful by opening the project in Supabase. If the table is there with all contacts, then the restoration has passed successfully. However, if the project is empty, there has been an issue, and the restoration has not been completed.

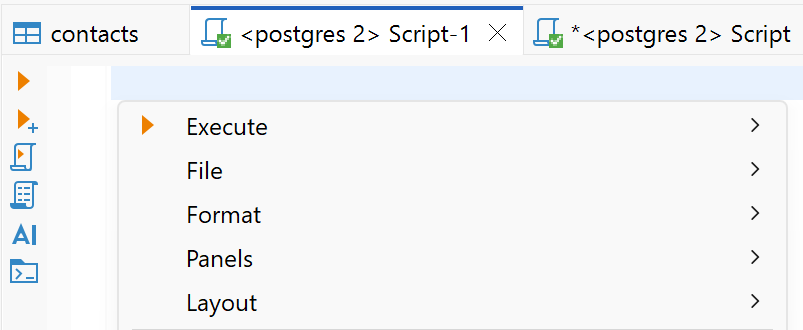
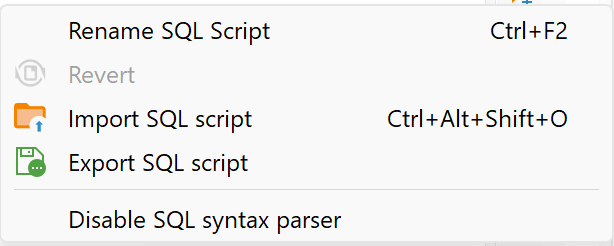
In this case, there is another way to restore the database.

### Restoration Way 2

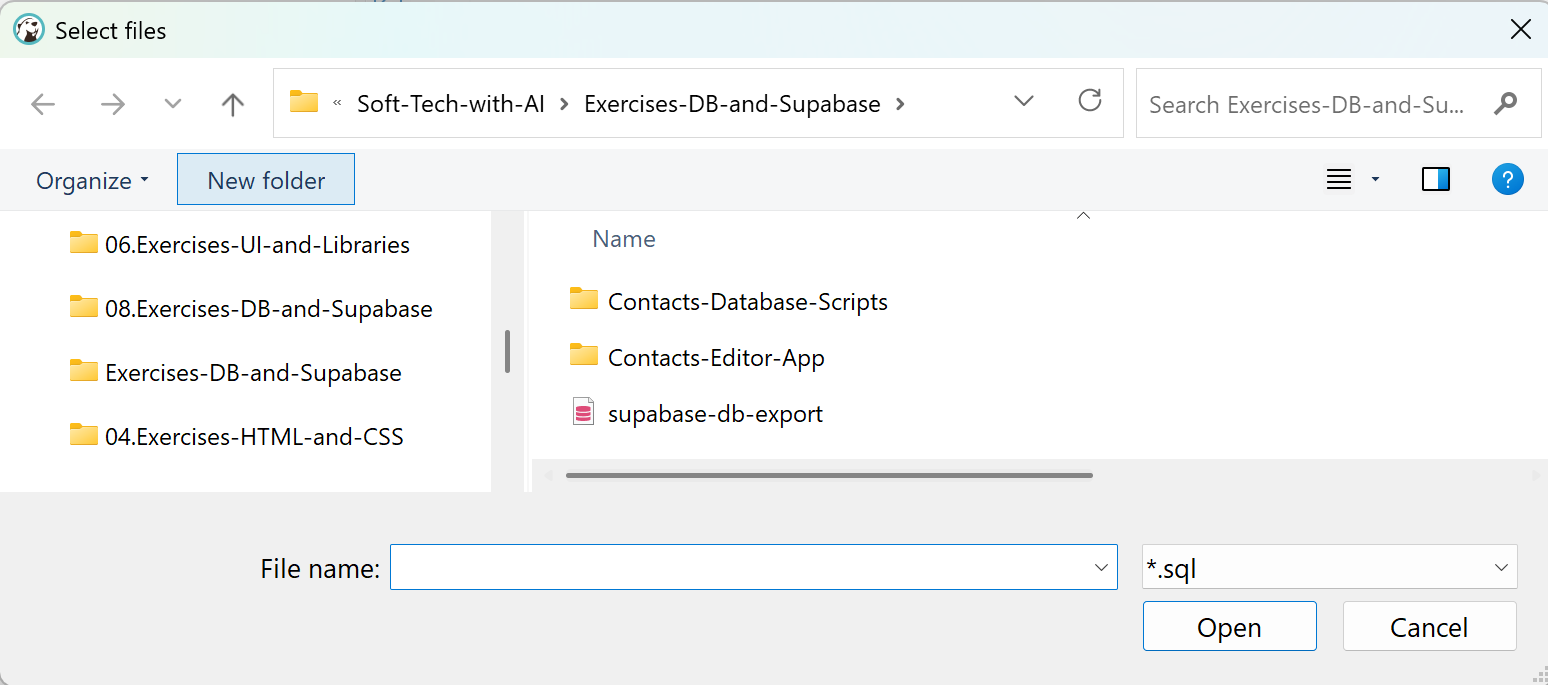
In **DBeaver** work with the **newly connected Supabase project** which was set up earlier in this exercise. Click **Databases**, then right click on postgres and select [**SQL Editor]**, then [New SQL Script]:

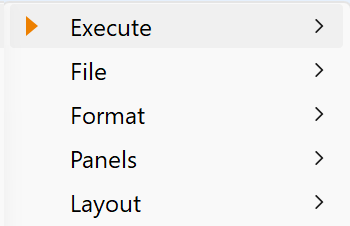
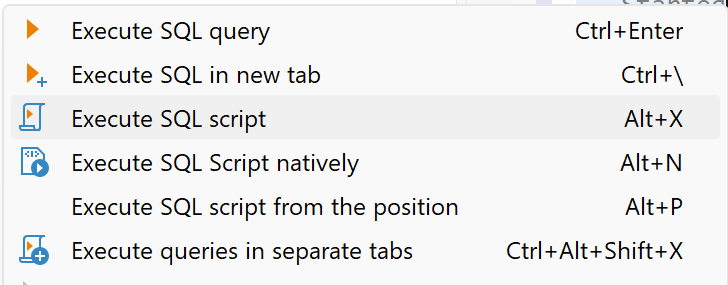
Open the backup file by right clicking anywhere in the newly opened space for script and selecting Import SQL script:

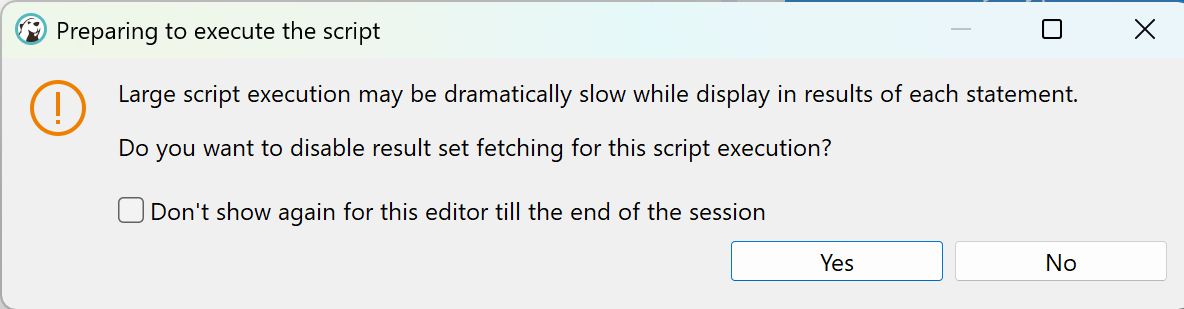
Then select the SQL file you created earlier:



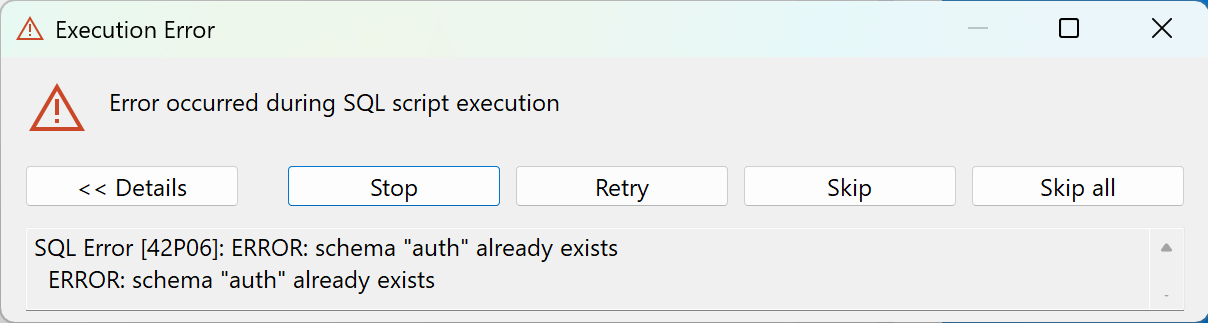
The script will be loaded. Select all the text ([Ctrl] + [A]), right click anywhere in text and select **[Execute]**, then **[Execute SQL script]**:

Answer with "Yes" to the next question:



You will receive the following error as well, select "Skip all":



When the project has been executed (this may take some time), review the project on Supabase. If the table is there, the restoration has been completed successfully. Congratulations!

## Employees Database

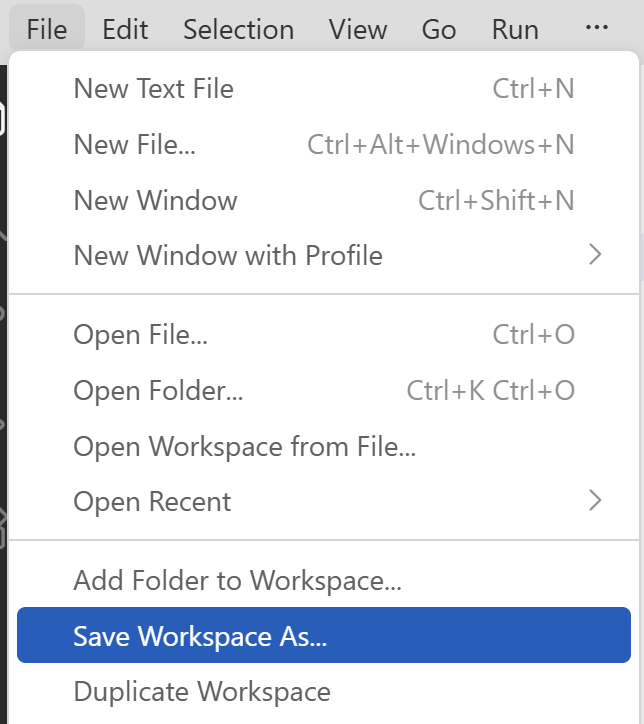
In this exercise we shall **build a Supabase Database to hold company employees**.

### Create Supabase Project

Pause one of your previous projects. **Create** a new Supabase **project** for your app, e.g. project named `EmployeesDB`.

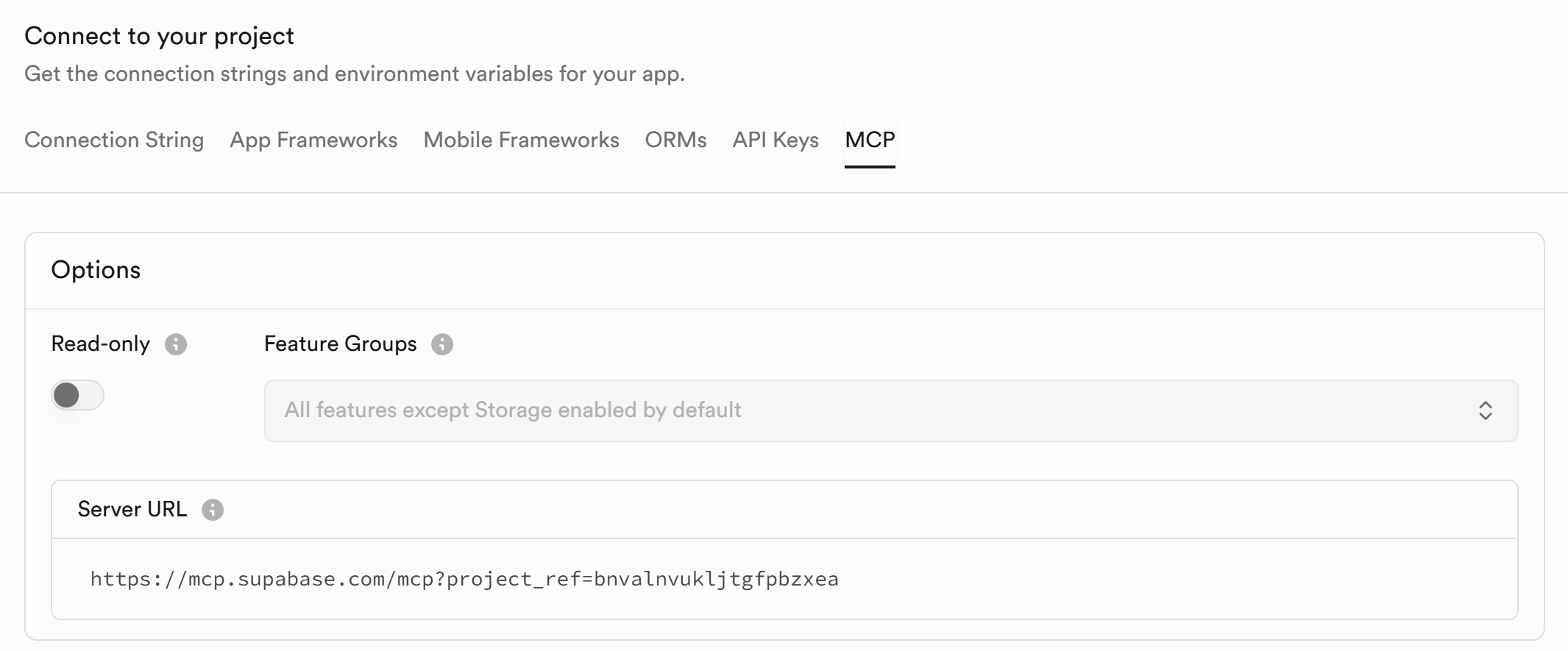
### Create VS Code Workspace

Create a new VS Code sub-folder for your app in your exercise working folder, e.g. `employees-app`. Then, from the **File** drop-down menu, select [Save Workspace As…]:

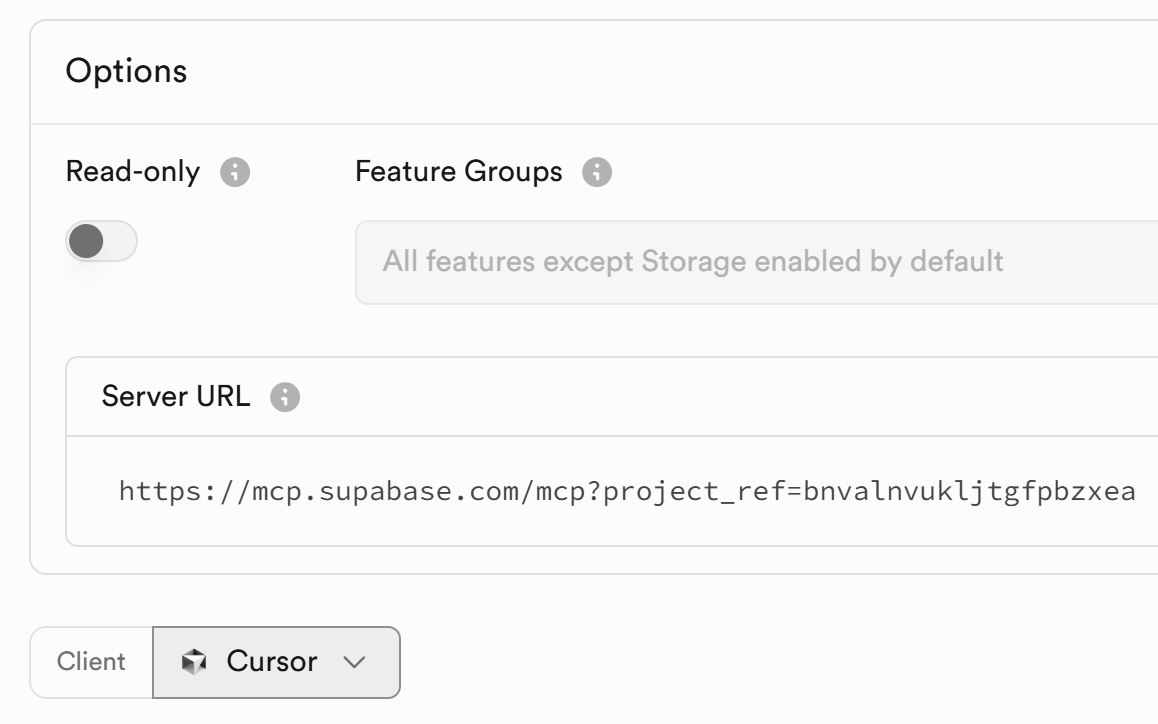


### Install Supabase MCP

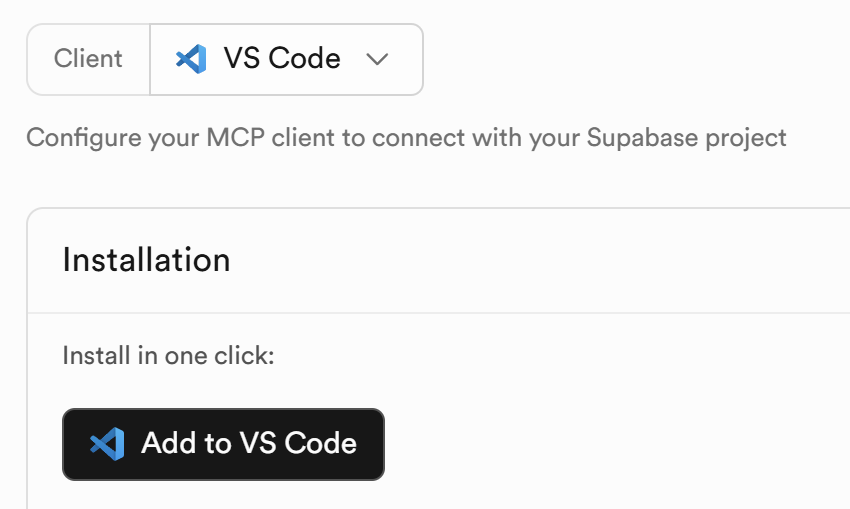
Go to your **Dashboard** in **Supabase** and select **Connect**, then the **MCP tab**:



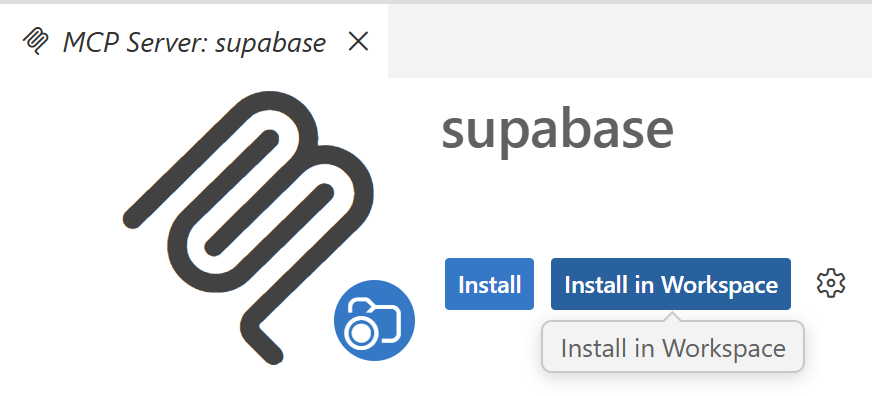
From the drop-down menu with current Cursor selection **choose VS Code**:

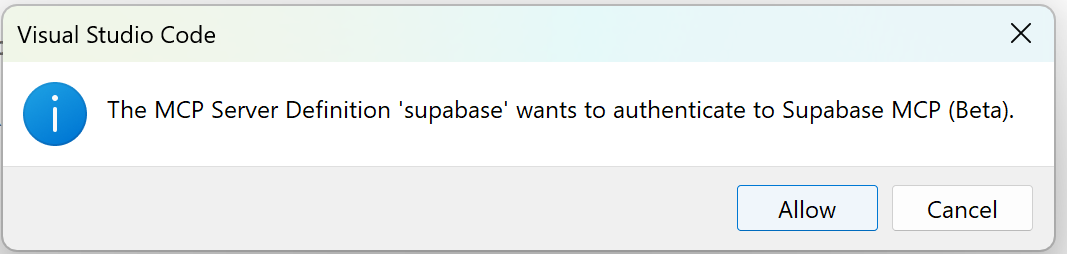
Press the [Add to VS Code] button:



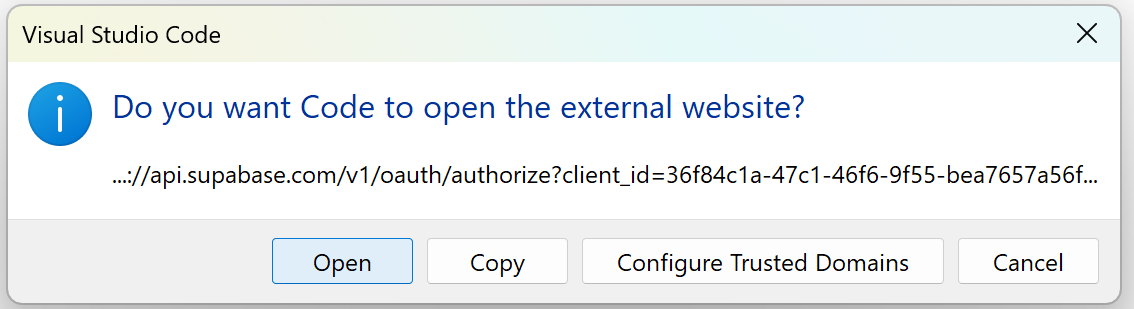
You will be forwarded to **Visual Studio Code**. Click the [Install in Workspace] button:



Click [Allow] for the following question:

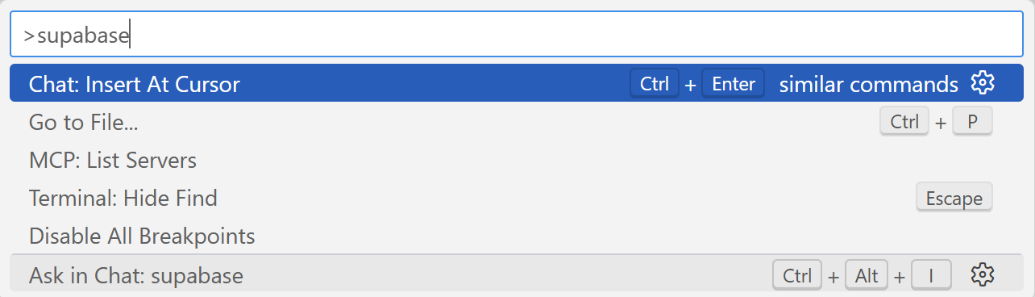


Authenticate the installed Supabase MCP server to your Supabase project. Select [Open]:



Then on the page that has opened, select [Authorize Visual Studio Code].

Check if the connection has been done correctly. Click [Ctrl] + [Shift] + [P]. Write "supabase" in the search bar and, if you see the **Ask in Chat:** supabase option, select it. This is all that needs to be done.



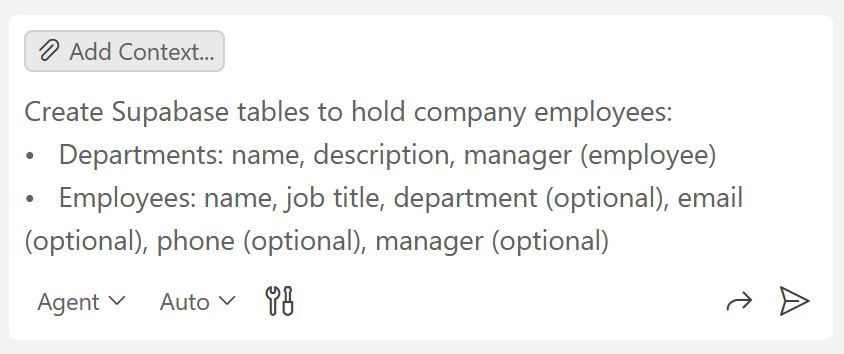
Supabase MCP works through ‘**Ask in Chat: Supabase**’ in **VS Code**.

### Create the Database Schema

**Create** Supabase **tables** to hold **company employees**:

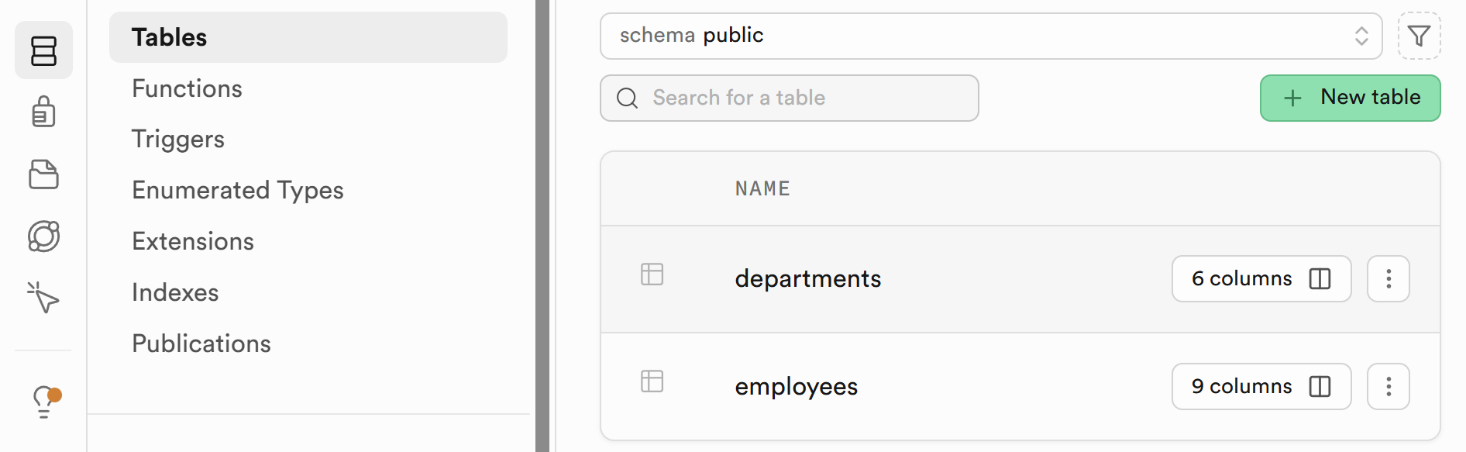
* **Departments**: name, description, manager (employee)
* **Employees**: name, job title, department (optional), email (optional), phone (optional), manager (optional)

#### The AI Prompt



#### The Tables in Supabase

View the database tables and DB diagram in Supabase, they will appear there this time.



### Insert Sample Data

**Insert sample employee data** into the Supabase database:

* **Departments**: Product, Marketing, Support, HR, Finance, Management
* **Employees**: 2-5 employees per department; management has only one employee, the CEO
* Employees shall have an **organizational hierarchy**: CEO -> Department Directors -> Regular Employees

See how the employees have been linked to the corresponding departments in the **Schema Visualizer** in Supabase:



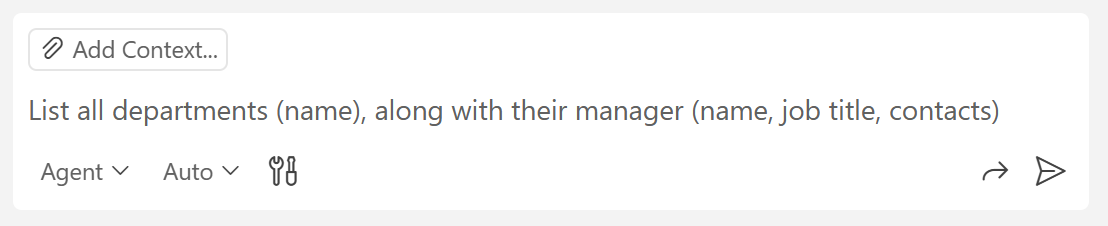
### Run SQL Queries

**Run** a few **database queries** through the **Supabase MCP** (in the GitHub Copilot chat windows in VS Code). The **results** will be retuned **in** **the chat**:

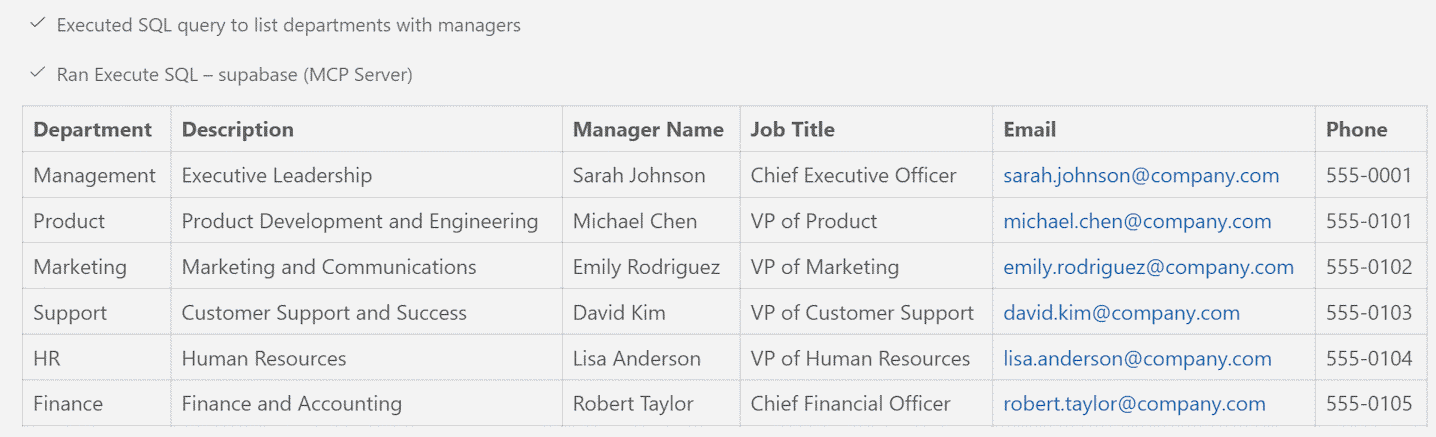
* **List all departments** (name), along with their manager (name, job title, contacts)
* **List all employees** (name, job title), along with their direct manager (name, job title)
* **List the departments**, along with the list of employees for each department (as JSON)
* \* Display the **full management hierarchy tree** with **hierarchical SQL query**.

#### Departments and Managers

##### The AI Prompt



##### The Result



### More Complex SQL Queries

Write a few **more complex SQL queries**:

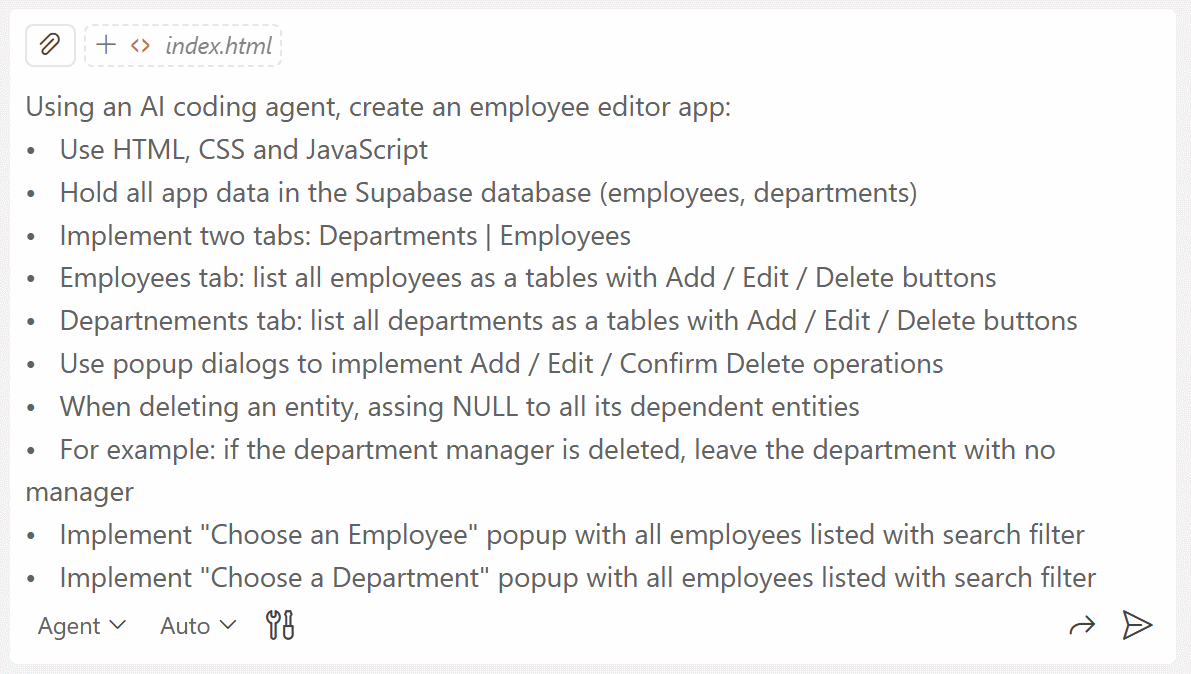
* Write an SQL query to list the **departments**, along with the **count of its employees** and a **list of the employees** (comma separated list of names) -> run the SQL from Supabase SQL Editor
* Write an SQL query to list the **managers**, along with their **directly managed employees** (comma separated list of names) -> run the SQL from Supabase SQL Editor

## Employees Editor App \*

Using an AI coding agent, create an **employee editor app**:

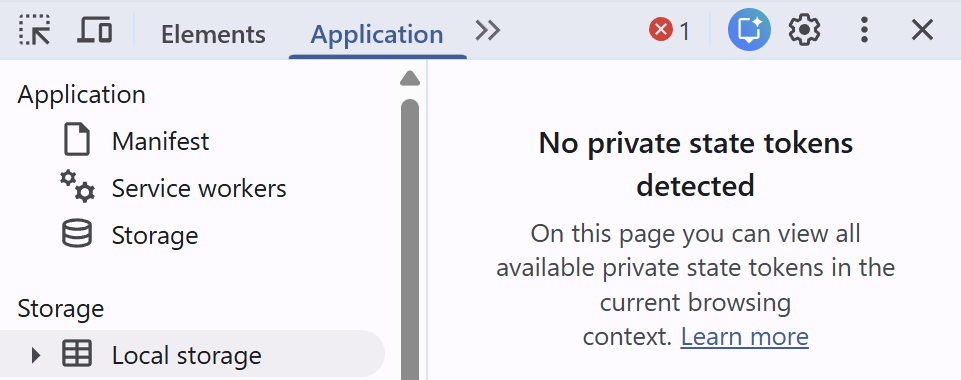
* Use **HTML**, **CSS** and **JavaScript**.
* Hold all app data in the **Supabase database** (employees, departments).
* Implement two **tabs**: **Departments** | **Employees**
  + **Employees tab**: list all employees as a table with Add / Edit / Delete buttons.
  + **Departments tab**: list all departments as a table with Add / Edit / Delete buttons.
* Use **popup dialogs** to implement **Add** / **Edit** / **Confirm Delete** operations.
* When **deleting** an entity, assign **NULL** to all its dependent entities.
  + For example: if the department manager is deleted, leave the department with no manager.
* Implement "**Choose an Employee**" **popup** with all employees listed with search filter.
* Implement "**Choose a Department**" **popup** with all departments listed with search filter.
* Use **icons**, **toast notifications**, and a **loading spinner** for improved UX.

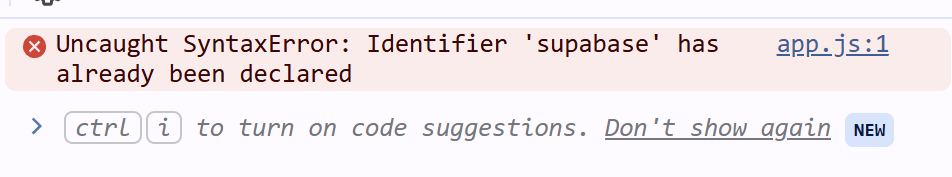
### The AI Prompt



### Debugging

You may encounter the **naming conflict error** for the `supabase` variable from earlier. Open [Dev Tools] and click the red  icon to see what the issue is:





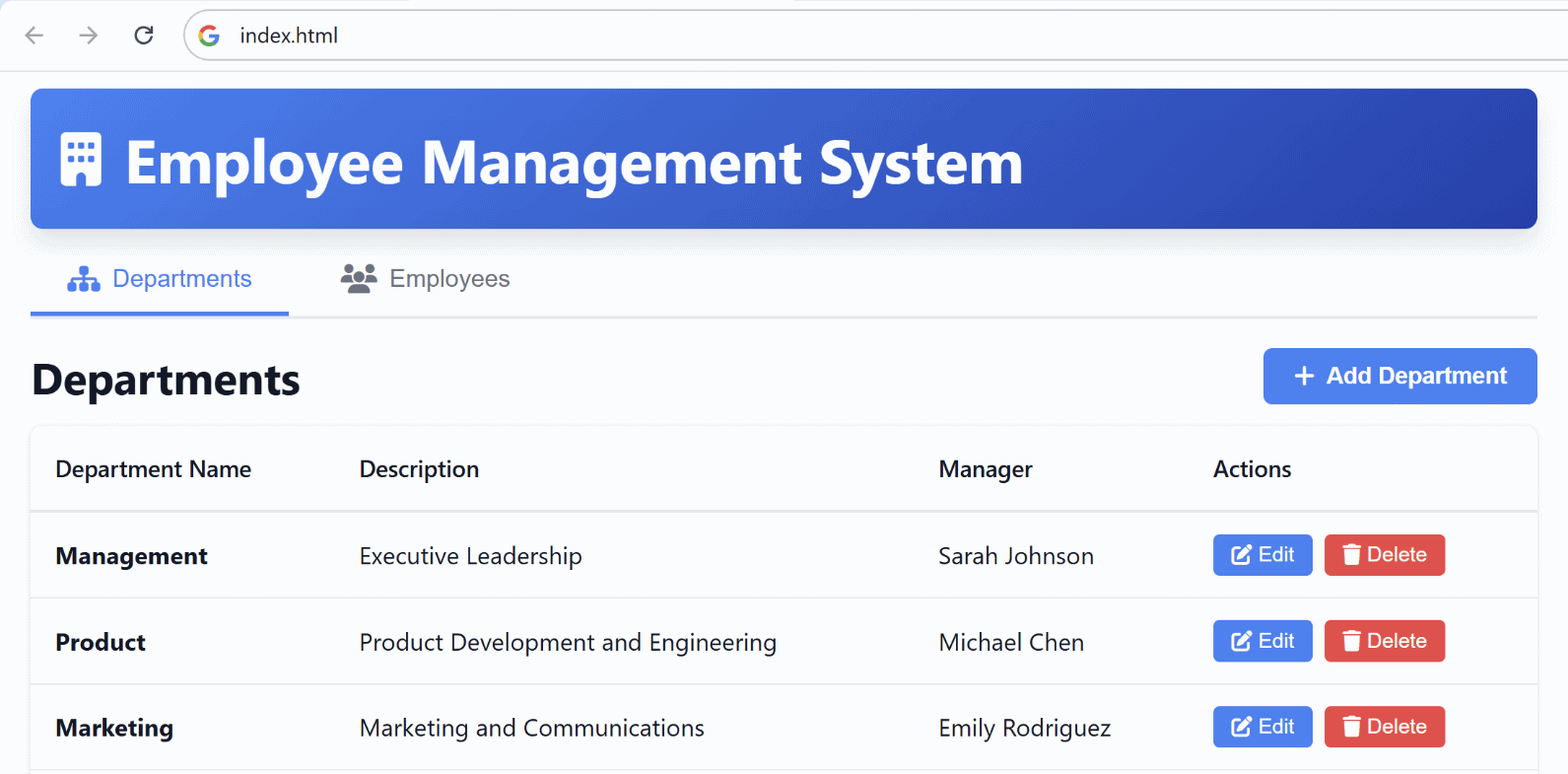
If this is the problem again, send a screenshot to the chat in VS Code and ask it to **rename the conflicting variable `**supabase**`**.

Fix any other bugs that may appear and test the functionalities of the app.

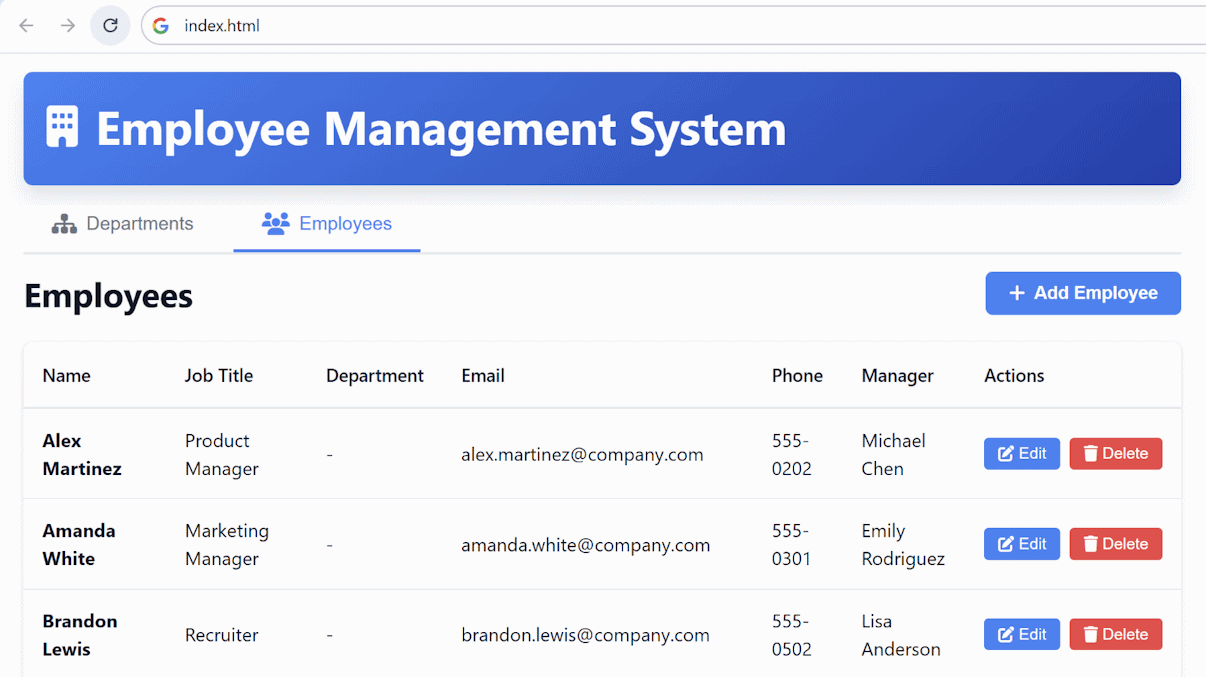
### The App in the Browser

This is how your app may look like at the end in your Web browser.

#### Departments Tab

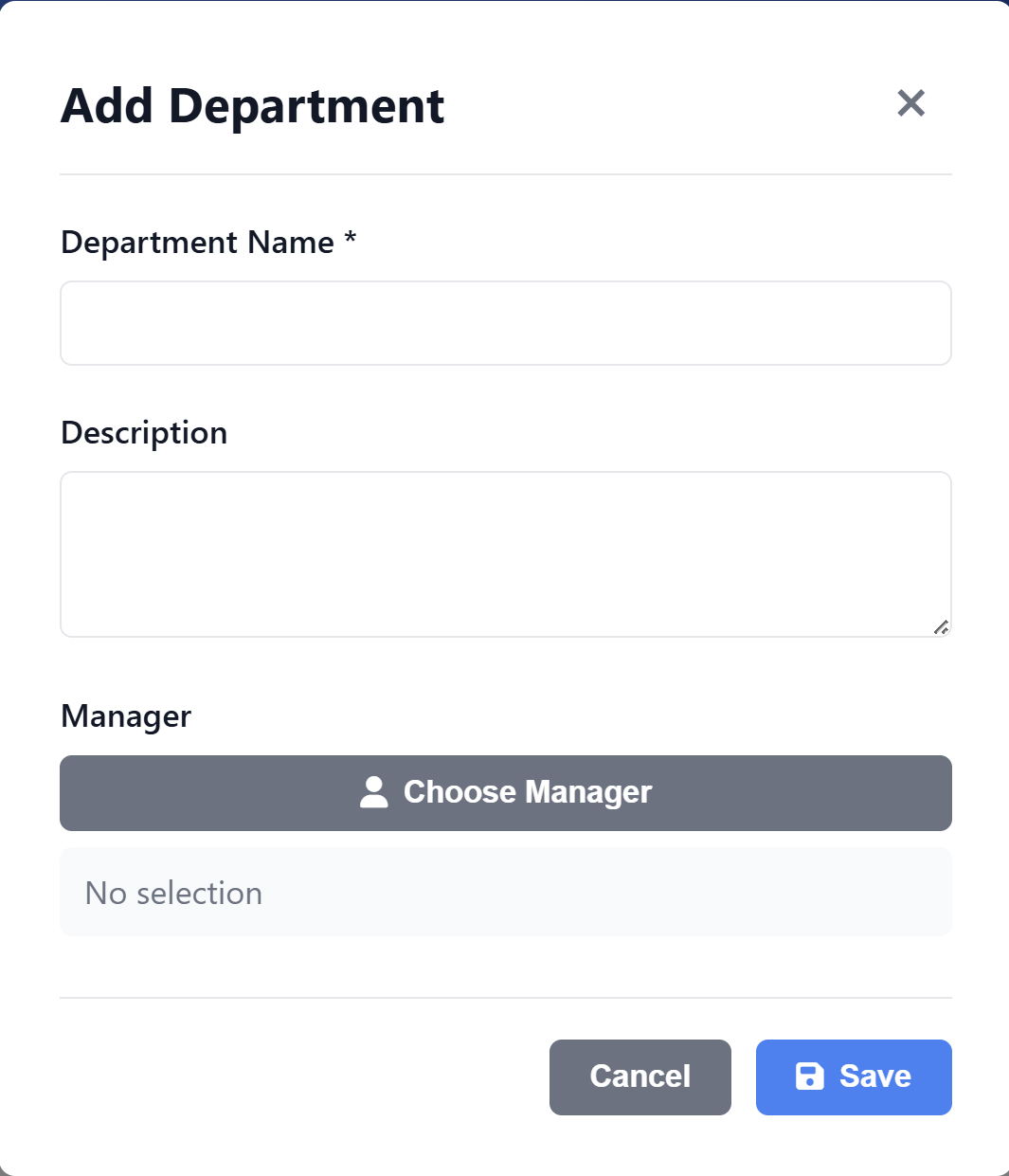
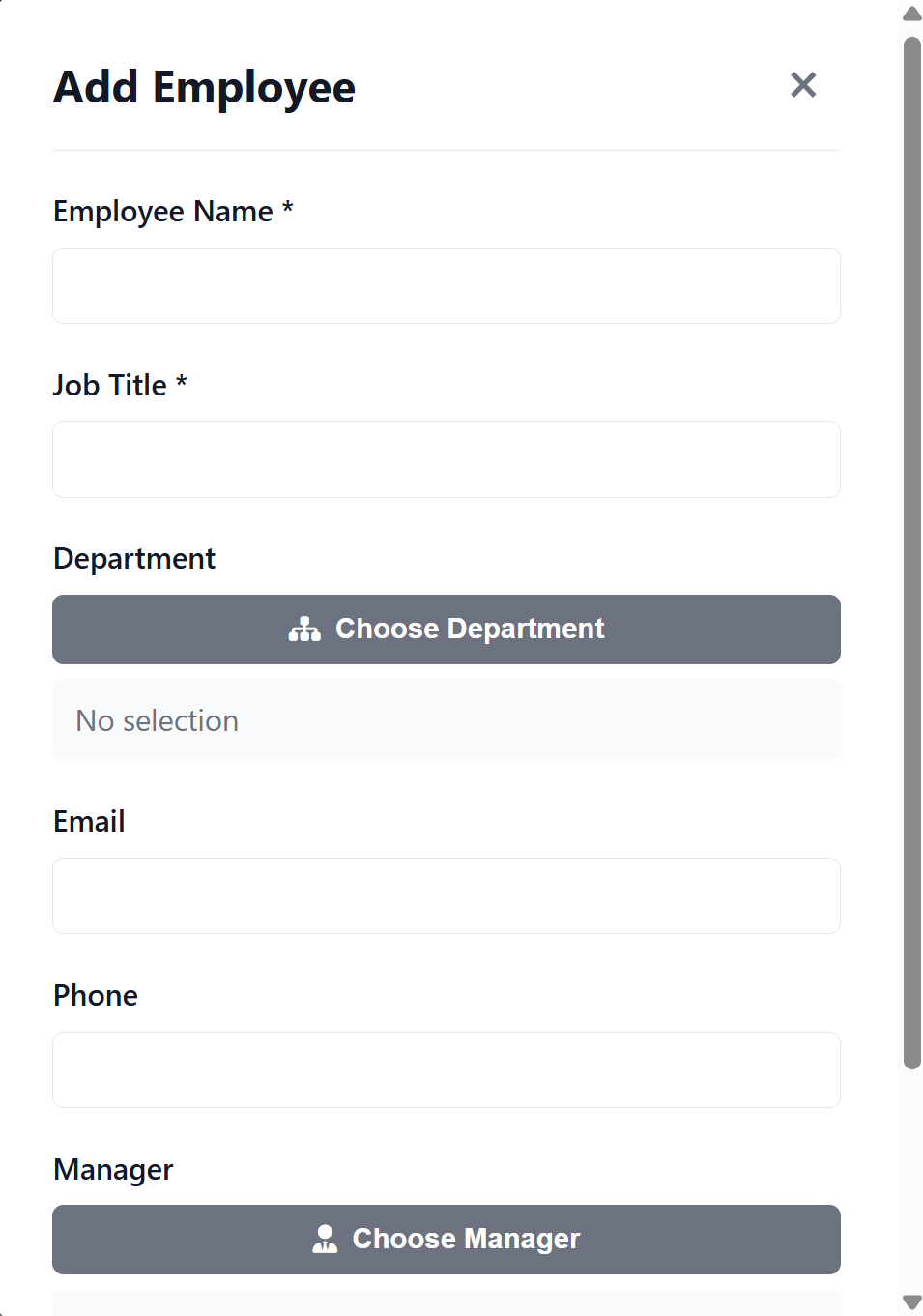


#### Employees Tab

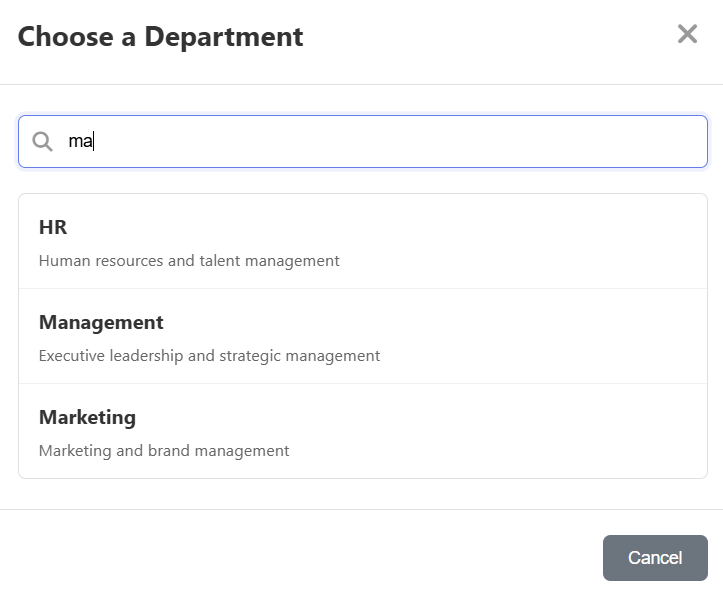
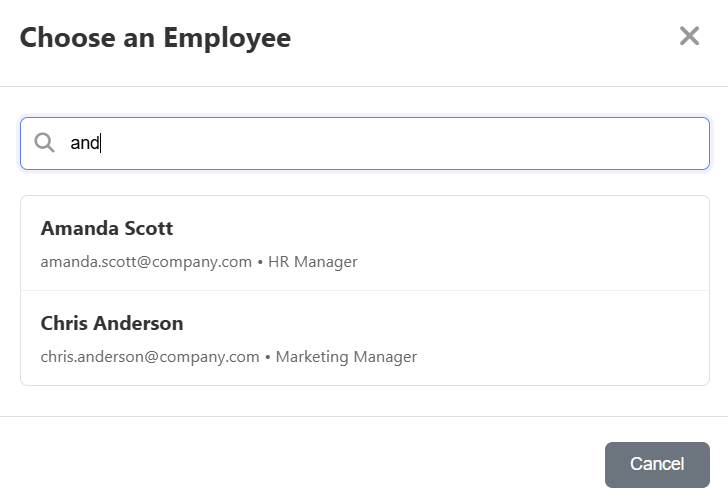


#### Add Department / Employee

Popup dialogs to add an employee / department.

Popup dialogs to **choose** an existing employee / department.

#### Edit Department / Employee

Popup dialogs to edit an employee / department.

