

# Databases and Supabase

Database, SQL, Supabase, Connecting JS Apps with Supabase, Supabase MCP



**Svetlin Nakov, PhD**  
Co-founder @ SoftUni

# Agenda

1. **Software Technologies:** Front-End, Back-End, Databases
2. **Databases:** Tables, Rows, Columns, Relationships, DB Schema
3. **Supabase Intro:** Projects, Database, Schema, Tables, Table Editor
4. **Database Design:** DB Tables, Data Types, DB Schemas
5. **SQL Language:** Table Structure: CREATE / ALTER TABLE, CRUD Operations: SELECT, INSERT, UPDATE, DELETE, Joins
6. **Supabase Services:** Database, Auth, Storage, Edge Functions, Logs, Backup, API Keys, Row-Level Security (RLS)
7. **JS Apps with Supabase:** API Connection, CRUD Operations
8. **Supabase MCP:** Connecting GitHub Copilot with Supabase



# Sli.do Code

#Soft-Tech-AI

Join at

[slido.com](https://slido.com)

**#Soft-Tech-AI**



# Breaks

20:00 / 21:00



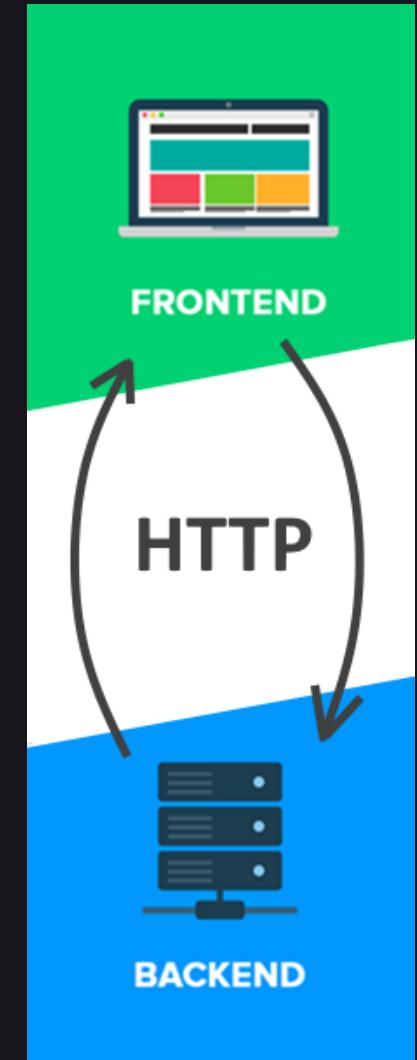
# Front-End and Back-End

## Software Architecture and Technologies, Front-End, Back-End, Databases



# Front-End and Back-End

- Front-end and back-end split modern apps into client-side components (UI) and server-side components (data & logic)
- Front-end == client-side components
  - Implement the user interface (UI)
- Back-end == server-side components (data and business logic + APIs)
  - Implements data storage and processing logic



# Front-End Technologies

- **Front-end technologies**
  - Implement app's **user interface** (UI)
  - HTML, CSS, JavaScript, TypeScript
  - React, Tailwind, Vite, React Router
- **Front-end developers** deal with UI, UX and front-end technologies and frameworks
- **AI-assisted dev tools** generate front-end using AI
  - GitHub Copilot, Cursor, Lovable, Bolt, Replit, Firebase Studio
  - Developers focus on the app logic and UI, not on the code



# Back-End Technologies

- Back-end technologies
  - Implement **server-side logic** and **data storage**
  - Technologies: Node.js, Express.js, Next.js
  - Store data in **databases** (like Supabase)
  - Expose **APIs** (access points for the front-end)
  - Server-side components are typically hosted in the **cloud**
- **Backend as a Service (BaaS)**, e. g. **Supabase**
  - Implement app's data storage, processing and authentication in the cloud, as a service



# Intro to Databases

Databases, Relational Model, Tables, Rows, Columns, Relationships, DB Schema



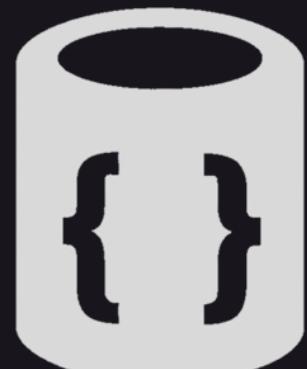
# What is a Database?

- A **database** is a collection of data, organized to be easily accessed, managed and updated
- Modern databases are managed by a Database Management Systems (**DBMS**), like **PostgreSQL**
  - Define database **structure**, e. g. tables, collections, columns, relations, indexes
  - **CRUD** operations: **Create / Read / Update / Delete** data (table rows / entities)
  - Execute **queries** (filter / search data)



# Databases

- **Databases** hold and manage data in the back-end systems
- **Relational databases** (RDBMS systems)
  - Hold data in **tables + relationships**
  - Use the **SQL** language to query / modify data
  - Examples: **PostgreSQL, Supabase, MySQL**
- **NoSQL databases**
  - Hold **document collections** or **key-value pairs**
  - Examples: **MongoDB, IndexedDB, Redis**



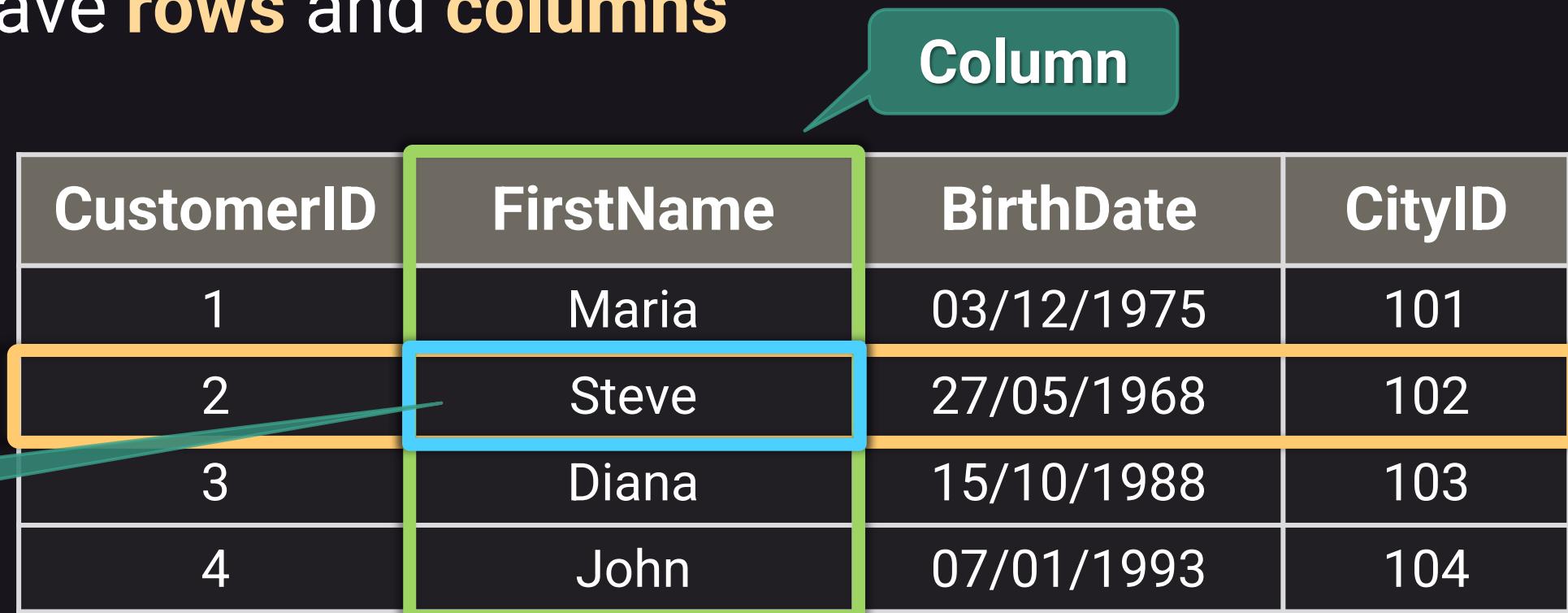
# Relational Databases (SQL Databases)

- Relational (SQL) databases organize data in **tables**
  - **Tables** have strict **structure** (columns of certain data types)
  - Can have **relationships** (connections) to other tables
- Relational databases use the **Structured Query Language (SQL)** to define / manipulate data
  - Extremely **powerful** for complex queries
- Relational databases are the **most widespread** data management technology



# Tables, Rows, Columns, Cells

- Relational databases (like Supabase) hold data in **tables**
- Tables have **rows** and **columns**



CustomerID	FirstName	BirthDate	CityID
1	Maria	03/12/1975	101
2	Steve	27/05/1968	102
3	Diana	15/10/1988	103
4	John	07/01/1993	104

- Columns define **data type** (e. g. text, number, date)

# SQL Language

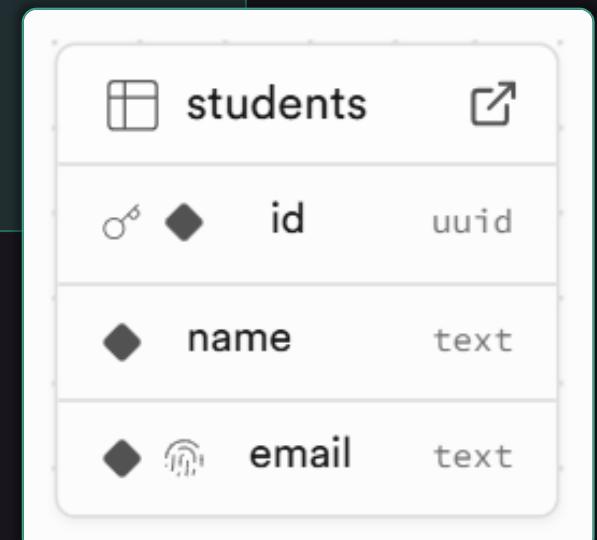
- SQL is a database-level language, used to **define** table structure, **query** and **modify** table data



```
CREATE TABLE students (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    name TEXT NOT NULL,
    email TEXT NOT NULL UNIQUE
)
```

```
INSERT INTO students(name, email) VALUES
('Steve', 'steve@gmail.com')
```

```
SELECT name, email FROM students
```



# Relationships between Tables

- Relational data is stored into database **tables** with
  - A **primary key** (unique key identifying each row) and
  - **Foreign keys** defining relationships (connections)



# Table Relationships – Example

**OrderItems**

ID	OrderID	Name	Quantity	Price
1	11	Table	1	250.00
2	11	Chair	4	160.00

**Customers**

ID	Name	Email
101	Peter	peter.j@yahoo.com
102	Jayne	jayne98@gmail.com

**Orders**

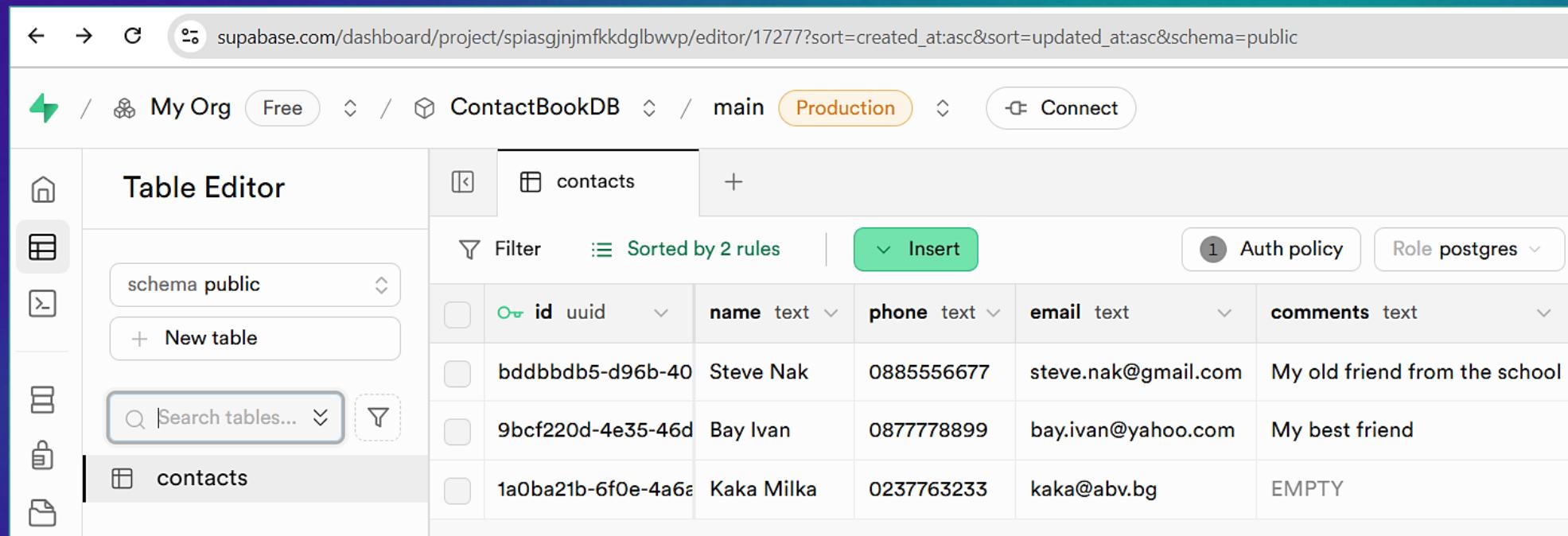
ID	CustomerID	Date	Total Price
11	101	1/07/2026	410.00
12	102	12/07/2026	570.30

# Database Servers

- **Database servers** (DBMS systems) **define, manipulate, retrieve** and **manage** data in a database
  - Local DB servers, cloud-bases DB servers
- Examples of DB servers:
  - **MySQL / MariaDB** – very popular, simple, open-source, for Web sites / apps
  - **PostgreSQL** – very powerful, open-source, for complex projects and apps
  - **Supabase** – hosted cloud instances of **PostgreSQL** + services
  - Others: **MS SQL Server, Oracle, MongoDB, Redis, SQLite, ...**

# Supabase Intro

Supabase BaaS, Projects, Databases,  
DB Schema, Table Editor, SQL Editor



The screenshot shows the Supabase Table Editor interface. The URL in the browser is `supabase.com/dashboard/project/spiasgjnjmfkkdglbwpp/editor/17277?sort=created_at:asc&sort=updated_at:asc&schema=public`. The navigation bar includes 'My Org' (Free), 'ContactBookDB', 'main', 'Production' (selected), and 'Connect'. The left sidebar has icons for Home, Tables, New table, and Search tables... The main area is titled 'Table Editor' for the 'contacts' table in the 'public' schema. It shows a grid of data with columns: id (uuid), name (text), phone (text), email (text), and comments (text). The first row contains 'bddbbbdb5-d96b-40', 'Steve Nak', '0885556677', 'steve.nak@gmail.com', and 'My old friend from the school'. The second row contains '9bcf220d-4e35-46d', 'Bay Ivan', '0877778899', 'bayivan@yahoo.com', and 'My best friend'. The third row contains '1a0ba21b-6f0e-4a6a', 'Kaka Milka', '0237763233', 'kaka@abv.bg', and 'EMPTY'. There are 'Insert' and 'Auth policy' buttons at the top of the table grid.

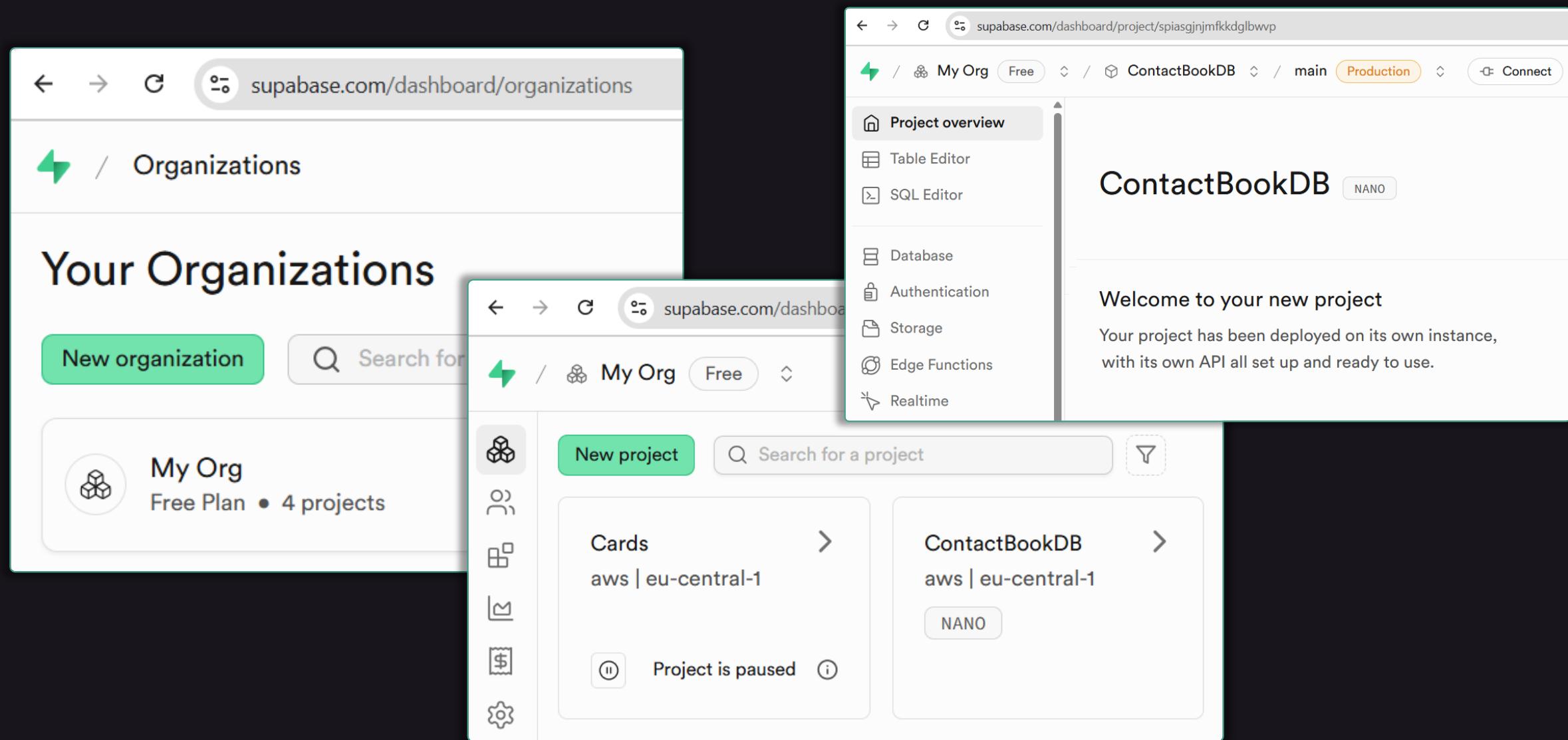
	id	name	phone	email	comments
	bddbbbdb5-d96b-40	Steve Nak	0885556677	steve.nak@gmail.com	My old friend from the school
	9bcf220d-4e35-46d	Bay Ivan	0877778899	bayivan@yahoo.com	My best friend
	1a0ba21b-6f0e-4a6a	Kaka Milka	0237763233	kaka@abv.bg	EMPTY

# Supabase

- Supabase is a popular open-source BaaS platform (Backend-as-a-Service)
  - PostgreSQL database (as a service in the cloud)
  - Built-in authentication (user management)
  - Serverless storage (images, files, documents, media)
  - Edge functions (serverless code in the cloud)
  - Real-time events, message queues, vector DB, cron jobs
  - <https://supabase.com>



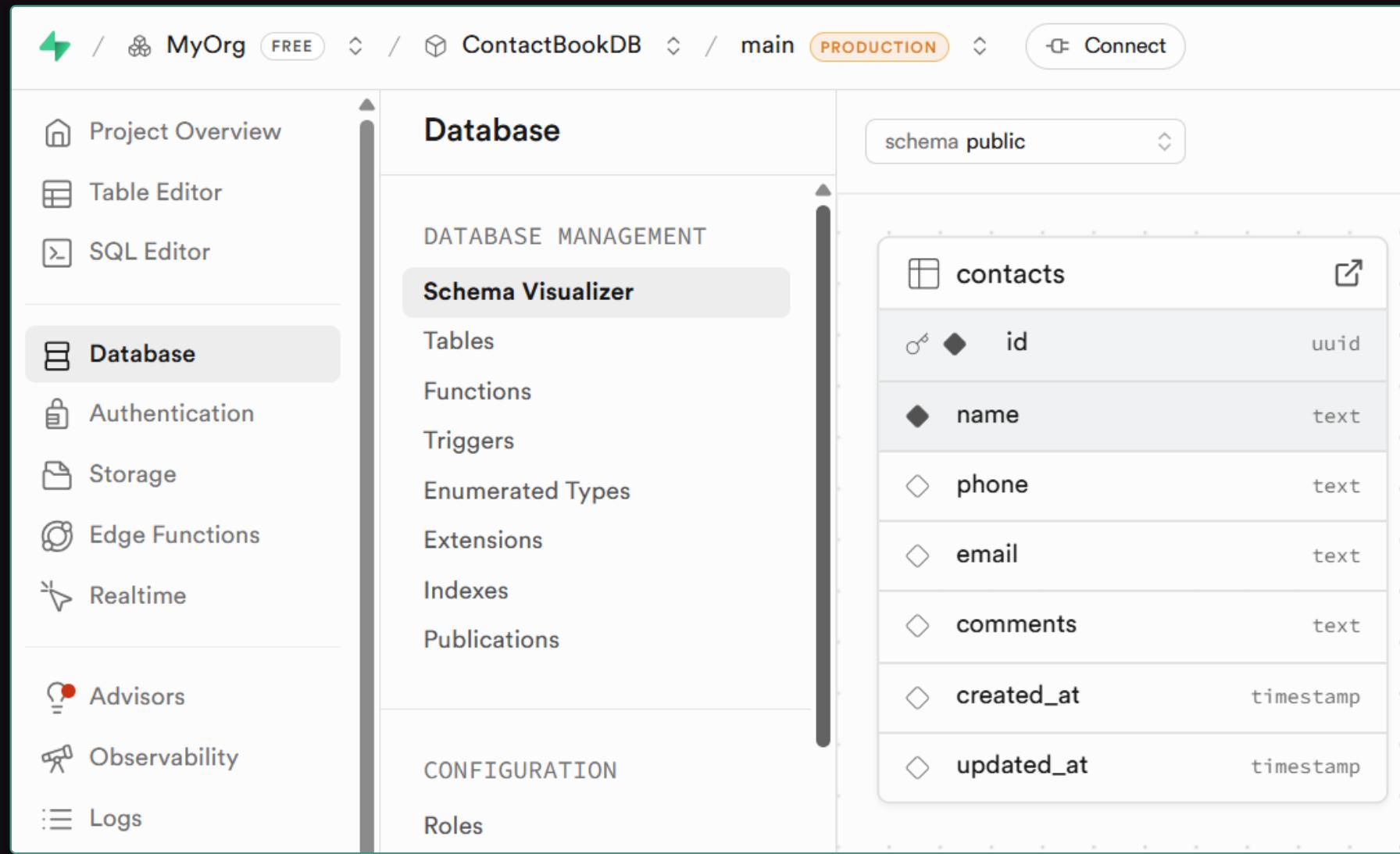
# Supabase Orgs and Projects



The image displays three screenshots of the Supabase dashboard:

- Top Left:** Shows the "Organizations" page with a "New organization" button and a "My Org" card indicating 4 projects.
- Middle Left:** Shows the "My Org" project page with a "New project" button and two existing projects: "Cards" (aws | eu-central-1) and "ContactBookDB" (aws | eu-central-1).
- Top Right:** Shows the "Project overview" for the "ContactBookDB" project, which is in "Production". It includes a "Welcome to your new project" message and deployment details.

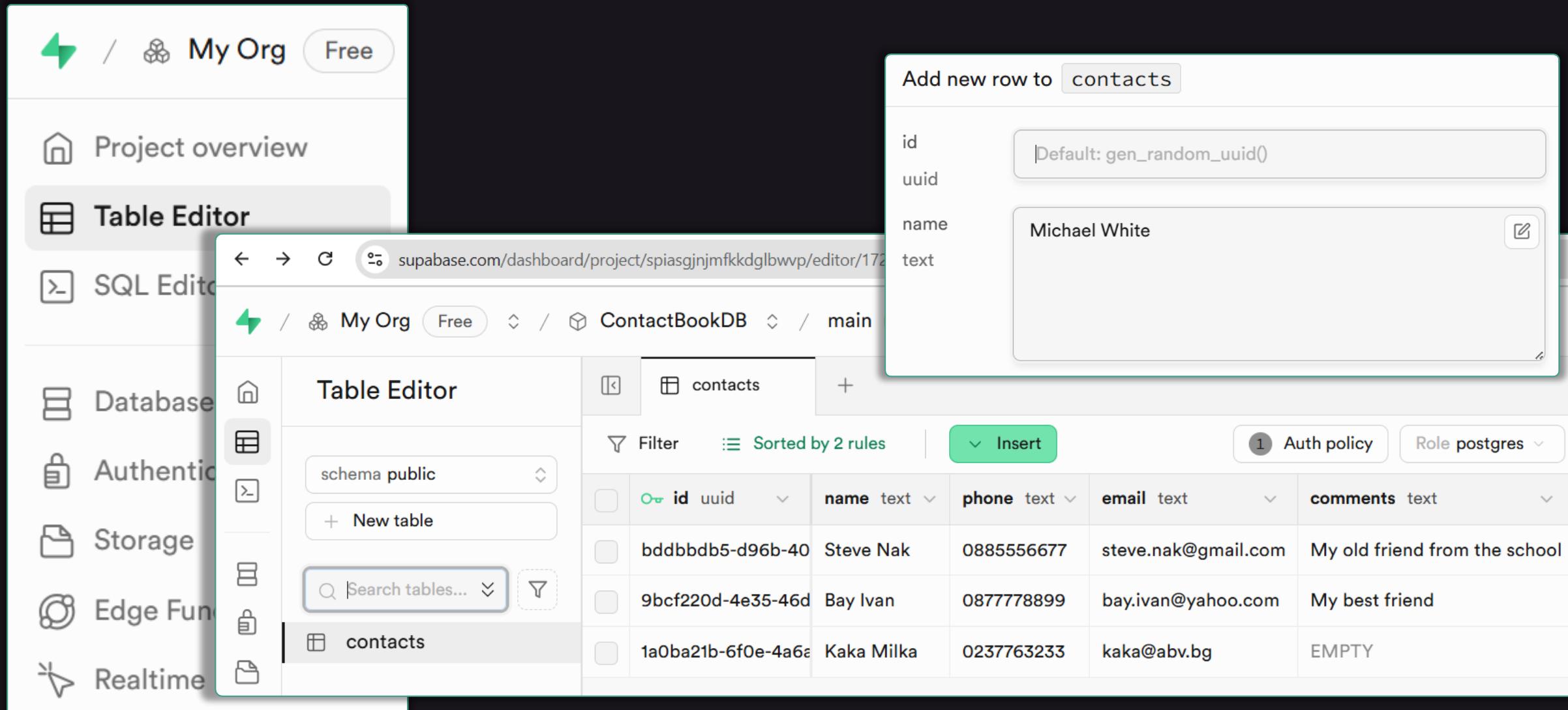
# Database Schema View



The screenshot shows a database schema visualization interface. At the top, the navigation bar displays: MyOrg (FREE) / ContactBookDB / main (PRODUCTION) / Connect. On the left, a sidebar menu includes: Project Overview, Table Editor, SQL Editor, Database (selected), Authentication, Storage, Edge Functions, Realtime, Advisors, Observability, and Logs. The main area is titled "Database" under "DATABASE MANAGEMENT". It features a "Schema Visualizer" section with tabs for Tables, Functions, Triggers, Enumerated Types, Extensions, Indexes, and Publications. Below this, under "CONFIGURATION", are Roles. A dropdown menu shows "schema public". The "contacts" table is listed, containing columns: id (uuid), name (text), phone (text), email (text), comments (text), created\_at (timestamp), and updated\_at (timestamp). The "id" column is marked as primary.

	Column Name	Type
id	id	uuid
	name	text
	phone	text
	email	text
	comments	text
	created_at	timestamp
	updated_at	timestamp

# Supabase Table Editor



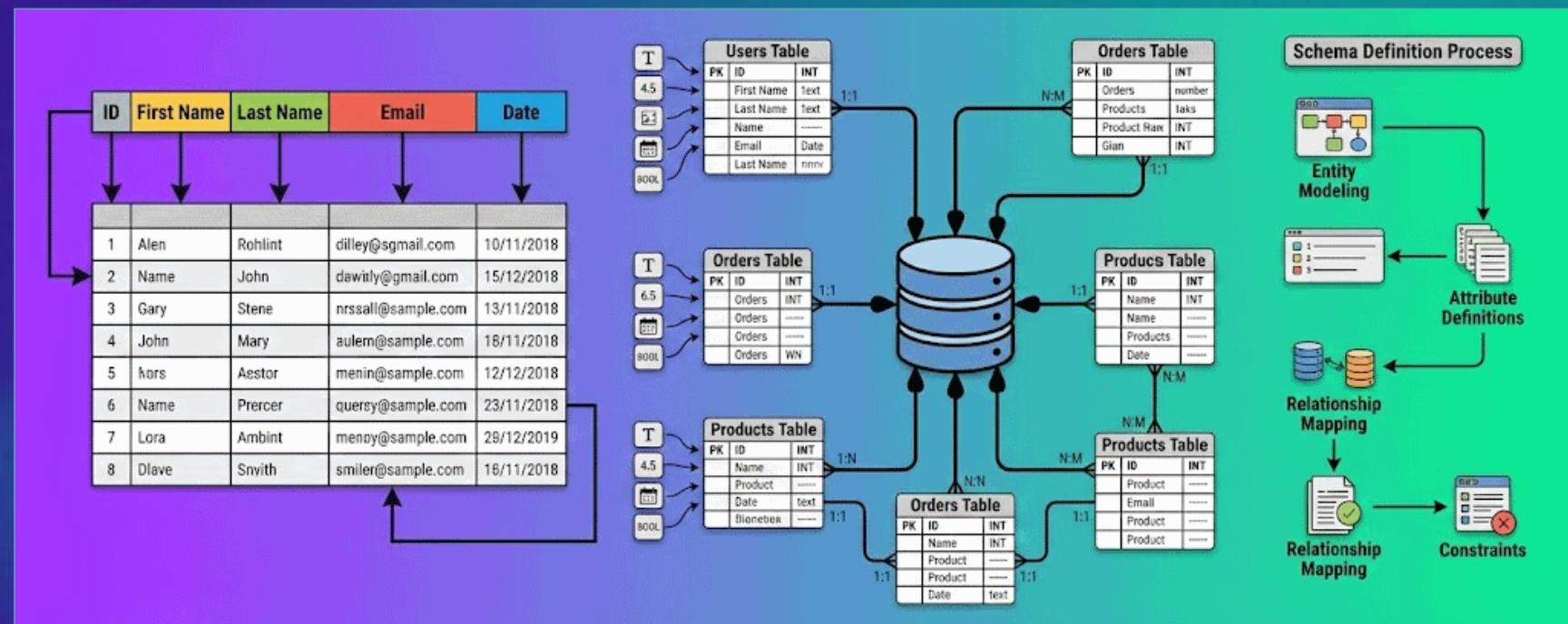
The screenshot shows the Supabase Table Editor interface. On the left, the sidebar includes Project overview, Table Editor (selected), SQL Editor, Database, Authentication, Storage, Edge Functions, and Realtime. The main area shows a sub-sidebar for the 'Table Editor' tab, which lists 'schema public', '+ New table', and a search bar for 'Search tables...'. The main workspace displays the 'ContactBookDB' database with the 'main' schema selected. A table named 'contacts' is shown with the following columns: id, name, phone, email, and comments. The table contains three rows of data:

	id	name	phone	email	comments
	bddbbdb5-d96b-40	Steve Nak	0885556677	steve.nak@gmail.com	My old friend from the school
	9bcf220d-4e35-46d	Bay Ivan	0877778899	bayivan@yahoo.com	My best friend
	1a0ba21b-6f0e-4a6a	Kaka Milka	0237763233	kaka@abv.bg	EMPTY

A modal window titled 'Add new row to contacts' is open, prompting for values for 'id' (with a note 'Default: gen\_random\_uuid()'), 'name' (set to 'Michael White'), and 'text'.

# Database Schema Design

Defining Table Structure,  
Relationships and Database Schema



# Table Structure

- DB table structure: **columns** of certain **data type**
- Example: table **Bookmarks**
  - id: **int** (primary key)
  - url: **text** (not null)
  - comments: **text**
  - created\_at: **date & time**

bookmarks		
  #	id	int8
 	url	text
 	comments	text
 	created_at	timestamptz

bookmarks			
		 Filter	 Sort
 id	int8	url	text
1		https://nakov.com	Nakov's Web site
2		https://softuni.bg	SoftUni Web site
3		https://wikipedia.org	NULL

# PostgreSQL Data Types

- **Columns** in the database tables have **data type**:
  - **integer / bigint**, e. g. 150
  - **uuid** (unique id), e. g. 373cc6ac-cf4e-470d-9ff4-f2f611b23958
  - **text / varchar**, e. g. "Steve Smith"
  - **timestamptz** (date + time), e. g. 2026-01-26 12:21:05.746115+00
  - **numeric(10, 2)** (money), e. g. 3.45
  - **jsonb** (JSON object), e.g. {"name": "Steve", "age": 28}
  - **text[]** (array of strings), ["news", "front-page", "news"]

# Exercise: Bookmarks DB Table

- Using an AI chatbot, define a Supabase table to hold **Bookmarks** (**`id`**, **`url`**, **`comments`**, **`created_at`**)
- Run the SQL to create the table in the DB
- Insert a few data rows

bookmarks	
◆	<code>id</code> <code>uuid</code>
◆	<code>url</code> <code>varchar</code>
◇	<code>comments</code> <code>text</code>
◇	<code>created_at</code> <code>timestamp</code>

SQL Bookmarks table with RLS and timestamp index

```

1 create table bookmarks (
2   id bigint primary key generated always as identity,
3   url text not null,
4   comments text,
5   created_at timestamp with time zone default now(),
6 );
7
8 -- Create an index on created_at for better query performance
9 create index idx_bookmarks_created_at on bookmarks(created_at);
10
11 -- Enable Row Level Security (RLS) for security
12 alter table bookmarks enable row level security;
  
```

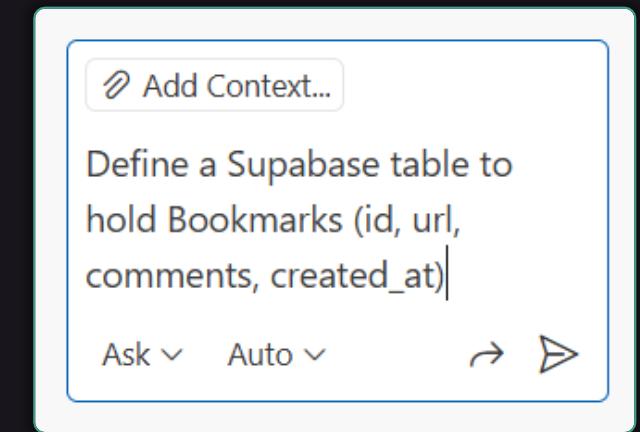
Results Explain Chart Export ✓ Run CTRL ⌘

Success. No rows returned

public.bookmarks

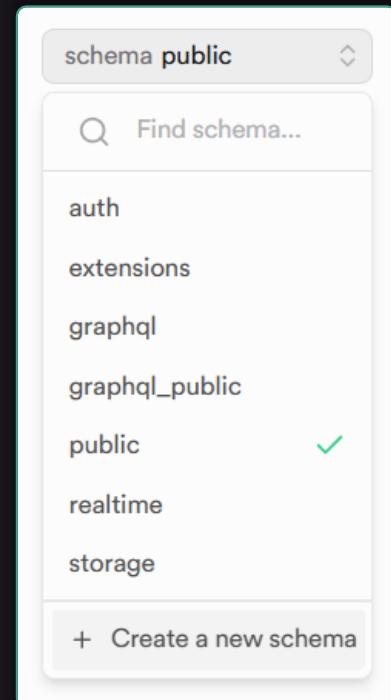
Filter Sort Insert Add RLS policy Index Advisor

	<code>id</code> int8	<code>url</code> text	<code>comments</code> text	<code>created_at</code> timestamp
1		https://nakov.com	Nakov's Web site	2026-01-26 12:21:05.746115+
2		https://softuni.bg	SoftUni Web site	2026-01-26 12:21:19.308172+
3		https://wikipedia.org	NULL	2026-01-26 13:37:30.946053



# Database Schema

- **Database schema** is the structure of your database
  - The structure of all **tables** (columns, data types)
  - **Relationships** (primary key / foreign key)
  - Unique keys, indices, triggers, functions
- **Supabase** databases hold **multiple schemas**:
  - **public** (default schema) – your app's tables
  - **auth** (Supabase schema) – users and auth
  - **storage** (Supabase schema) – buckets and files



# Exercise: Design a Database Schema

- Design a **DB schema** in Supabase to hold students and classes
  - **Students:** name and email
  - **Classes:** name, description, price, start date
  - **Enrollments:** student, class, enrollment date, price paid
  - Create **SQL script** to create the DB tables and relationships

Add Context...

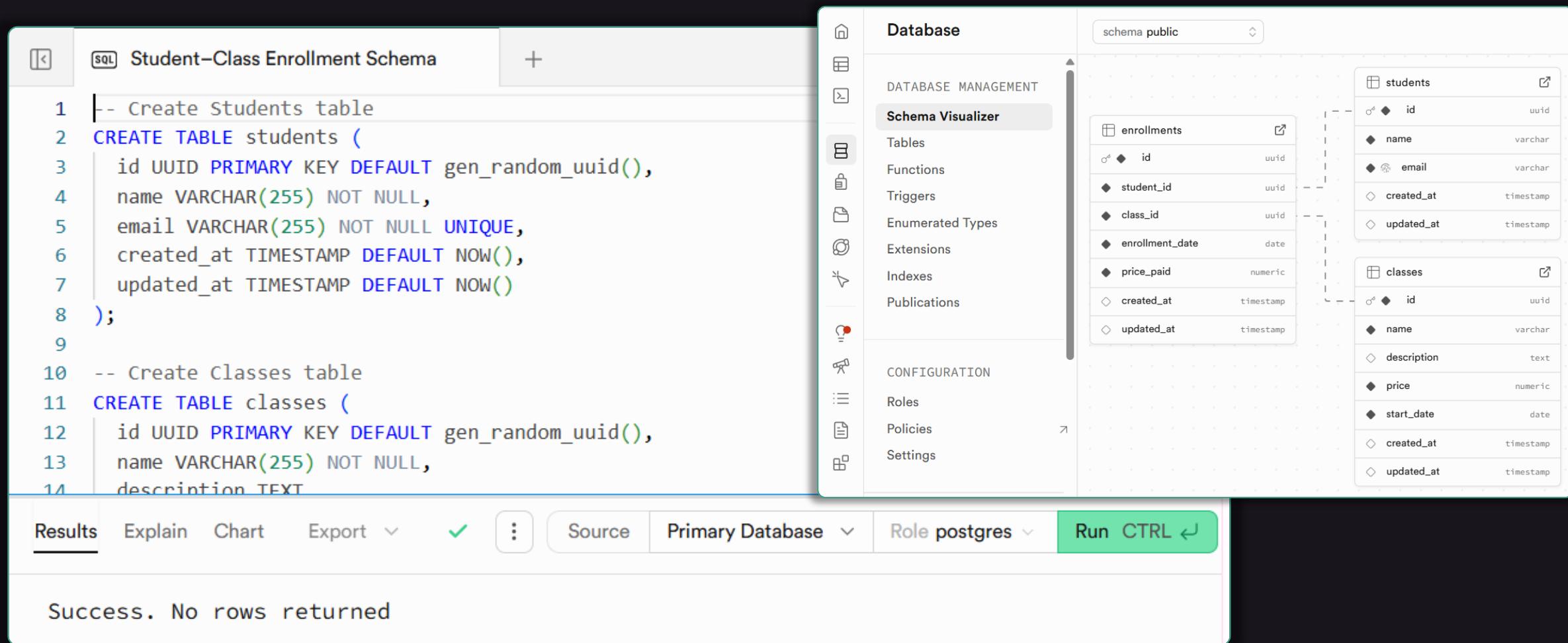
Design a DB schema in Supabase to hold students and classes  
Students: name and email  
Classes: name, description, price, start date  
Enrollments: student, class, enrollment date, price paid

Agent ▾ Auto ▾ ⚙️ ↗ ↘

```
schema.sql
3  -- Create Students table
4  CREATE TABLE students (
5    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
6    name VARCHAR(255) NOT NULL,
7    email VARCHAR(255) NOT NULL UNIQUE,
8    created_at TIMESTAMP DEFAULT NOW(),
9    updated_at TIMESTAMP DEFAULT NOW()
10 );
11
12 -- Create Classes table
13 CREATE TABLE classes (
14    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
15    name VARCHAR(255) NOT NULL,
16    description TEXT,
17    price DECIMAL(10, 2) NOT NULL,
18    start_date DATE NOT NULL,
19    created_at TIMESTAMP DEFAULT NOW(),
20    updated_at TIMESTAMP DEFAULT NOW()
21 );
```

# Exercise: Create DB Tables in Supabase

- Run the SQL script to create the tables in Supabase



The screenshot shows the Supabase Studio interface with the following components:

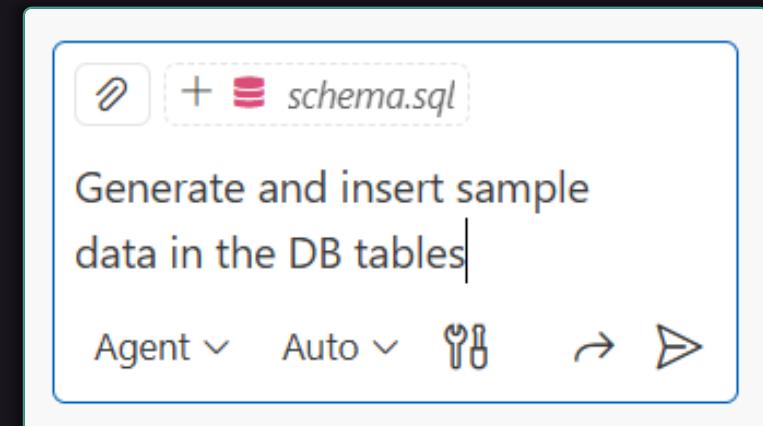
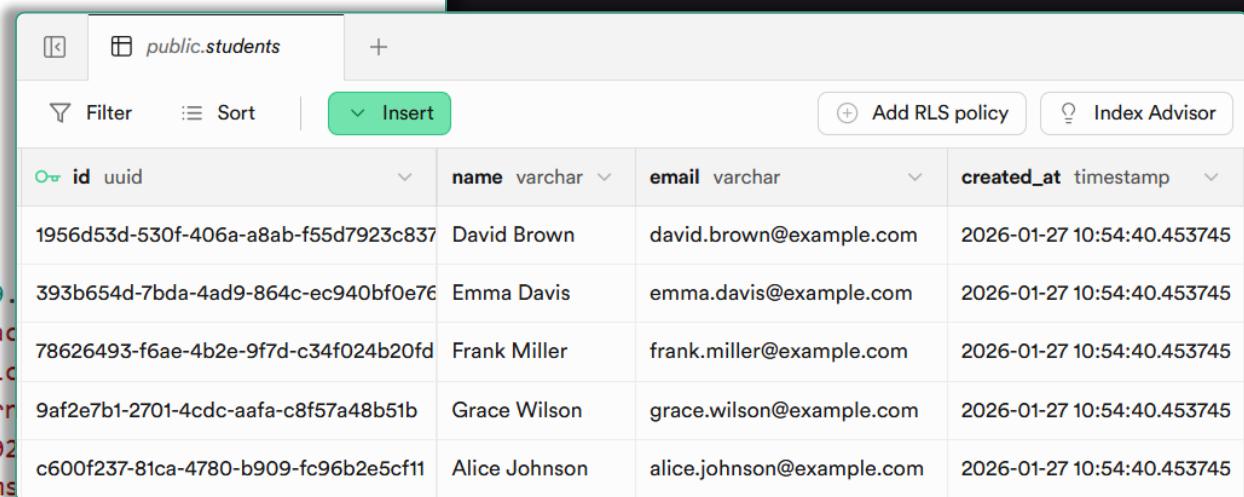
- Left Panel (SQL Editor):** Displays the SQL code for creating the 'students' and 'classes' tables.
- Middle Panel (Database Management):** Shows the 'Schema Visualizer' with the 'public' schema selected. It lists the 'enrollments' table and its columns: id, student\_id, class\_id, enrollment\_date, price\_paid, created\_at, and updated\_at.
- Right Panel (Table Details):** Shows two tables: 'students' and 'classes'. The 'students' table has columns: id (uuid), name (varchar), email (varchar), created\_at (timestamp), and updated\_at (timestamp). The 'classes' table has columns: id (uuid), name (varchar), description (text), price (numeric), start\_date (date), created\_at (timestamp), and updated\_at (timestamp).
- Bottom Navigation Bar:** Includes tabs for Results, Explain, Chart, Export, Source, Primary Database, Role, and Run (highlighted in green).
- Status Bar:** Shows the message "Success. No rows returned".

# Exercise: Insert Sample Data

- Generate and **insert sample data** in the DB tables

```
-- Insert sample students
INSERT INTO students (name, email) VALUES
('Alice Johnson', 'alice.johnson@example.com'),
('Bob Smith', 'bob.smith@example.com'),
('Carol Williams', 'carol.williams@example.com'),
('David Brown', 'david.brown@example.com'),
('Emma Davis', 'emma.davis@example.com'),
('Frank Miller', 'frank.miller@example.com'),
('Grace Wilson', 'grace.wilson@example.com'),
('Henry Moore', 'henry.moore@example.com');

-- Insert sample classes
INSERT INTO classes (name, description, price, start_date) VALUES
('Introduction to Python', 'Learn the basics of Python programming', 99.99, '2026-01-01'),
('Web Development with React', 'Build modern web applications using React', 129.99, '2026-02-01'),
('Data Science Fundamentals', 'Introduction to data science and analytics', 149.99, '2026-03-01'),
('Advanced JavaScript', 'Master advanced JavaScript concepts and patterns', 119.99, '2026-04-01'),
('SQL Database Design', 'Design and optimize SQL databases', 89.99, '2026-05-01'),
('Machine Learning Basics', 'Introduction to machine learning algorithms', 99.99, '2026-06-01'),
('Cloud Computing with AWS', 'Deploy applications on AWS', 179.99, '2026-07-01'),
('Mobile App Development', 'Create iOS and Android applications', 199.99, '2026-08-01');
```

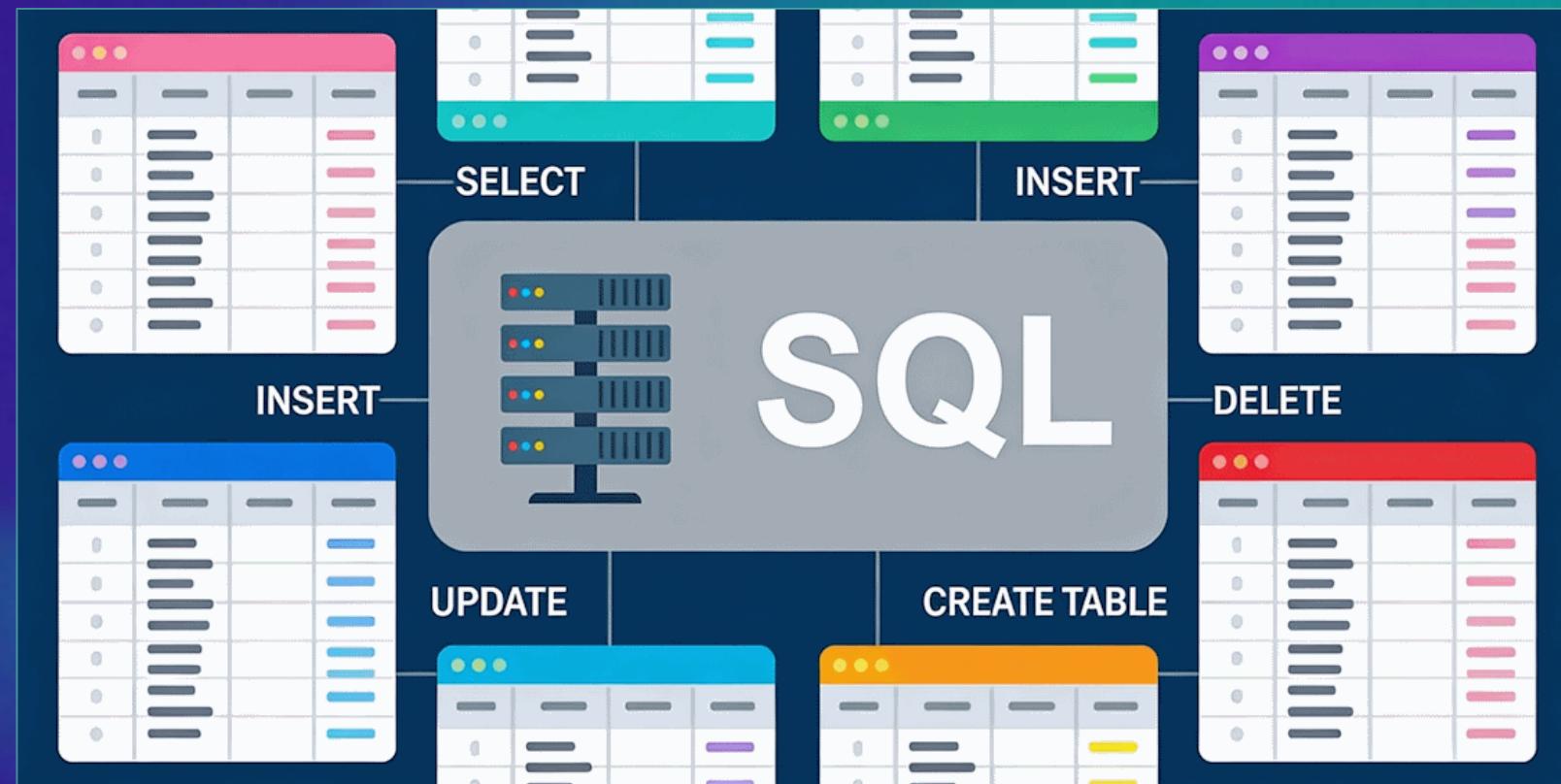



The screenshot shows a table viewer for the "public.students" table. The table has four columns: id (uuid), name (varchar), email (varchar), and created\_at (timestamp). There are 6 rows of data:

id	name	email	created_at
1956d53d-530f-406a-a8ab-f55d7923c837	David Brown	david.brown@example.com	2026-01-27 10:54:40.453745
393b654d-7bda-4ad9-864c-ec940bf0e76	Emma Davis	emma.davis@example.com	2026-01-27 10:54:40.453745
78626493-f6ae-4b2e-9f7d-c34f024b20fd	Frank Miller	frank.miller@example.com	2026-01-27 10:54:40.453745
9af2e7b1-2701-4cdc-aafa-c8f57a48b51b	Grace Wilson	grace.wilson@example.com	2026-01-27 10:54:40.453745
c600f237-81ca-4780-b909-fc96b2e5cf11	Alice Johnson	alice.johnson@example.com	2026-01-27 10:54:40.453745

# SQL Language and DB Schema

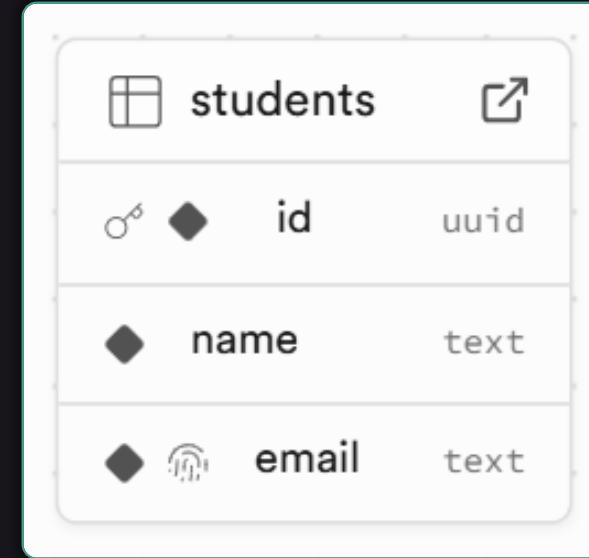
Table Structure: CREATE / ALTER / DROP TABLE



# What is SQL?

- **SQL (Simple Query Language)**
  - Database-level language for managing **DB schema** and **table data** in relational databases (RDBMS)
- Sample SQL code:

```
CREATE TABLE students (
    id UUID PRIMARY KEY
        DEFAULT gen_random_uuid(),
    name TEXT NOT NULL,
    email TEXT NOT NULL UNIQUE
)
```



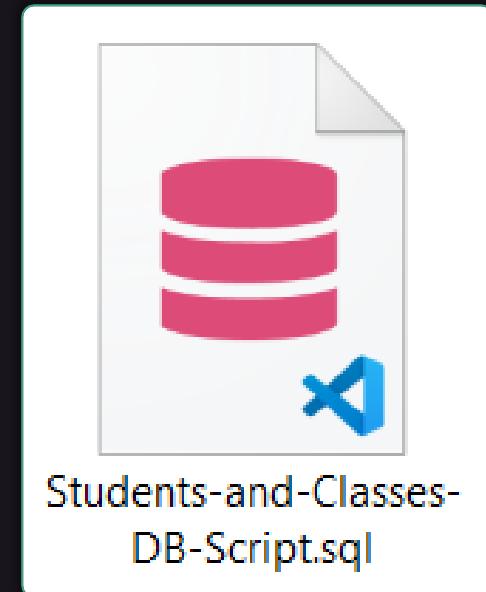
# DB Schema Definition with SQL

- **DDL** (Data Definition Language) in SQL
  - Define the table structure: **CREATE / ALTER / DROP TABLE**

```
CREATE TABLE enrollments (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    student_id UUID NOT NULL REFERENCES students(id) ON DELETE CASCADE,
    class_id UUID NOT NULL REFERENCES classes(id) ON DELETE CASCADE,
    enrollment_date DATE NOT NULL,
    price_paid DECIMAL(10, 2) NOT NULL,
    created_at TIMESTAMP DEFAULT NOW(),
    updated_at TIMESTAMP DEFAULT NOW(),
    UNIQUE(student_id, class_id) -- Prevent duplicate enrollments
);
```

# SQL DB Create Script

- You can create / backup / restore databases with a **SQL script** (holding schema + data)
- How to **restore a database** from SQL script?
  - Create a **new project** in Supabase
  - **Run the SQL script** (in the SQL Editor)
  - Check for **errors** / fix errors
- Example: run the attached SQL script
  - **Students-and-Classes-DB-Script.sql**

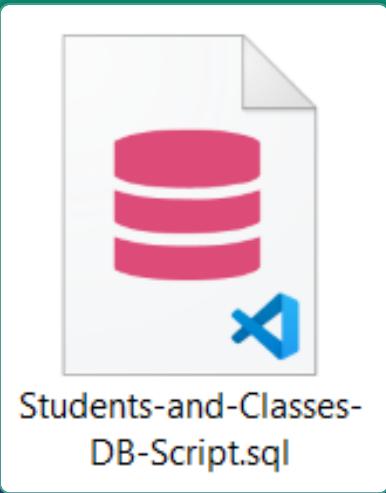


# Pause Unused Projects in Supabase



- The **free** tier in Supabase allows **up to 2 active projects**
- You can **pause unused projects**, to allow new projects

The image shows the Supabase Project Settings interface. The top navigation bar displays the organization name "MyOrg" (FREE), the project name "ContactBookDB", the branch "main", and the environment "PRODUCTION". The left sidebar lists "PROJECT SETTINGS" with options: General (selected), Compute and Disk, Infrastructure, Integrations, and Data API. The main content area is titled "Project availability" with the sub-section "Restart project". It states "Your project will not be available for a few minutes." with a "Restart project" button. Below it is the "Pause project" section, which states "Your project will not be accessible while it is paused." with a "Pause project" button. The "Pause project" button is highlighted with a blue border.

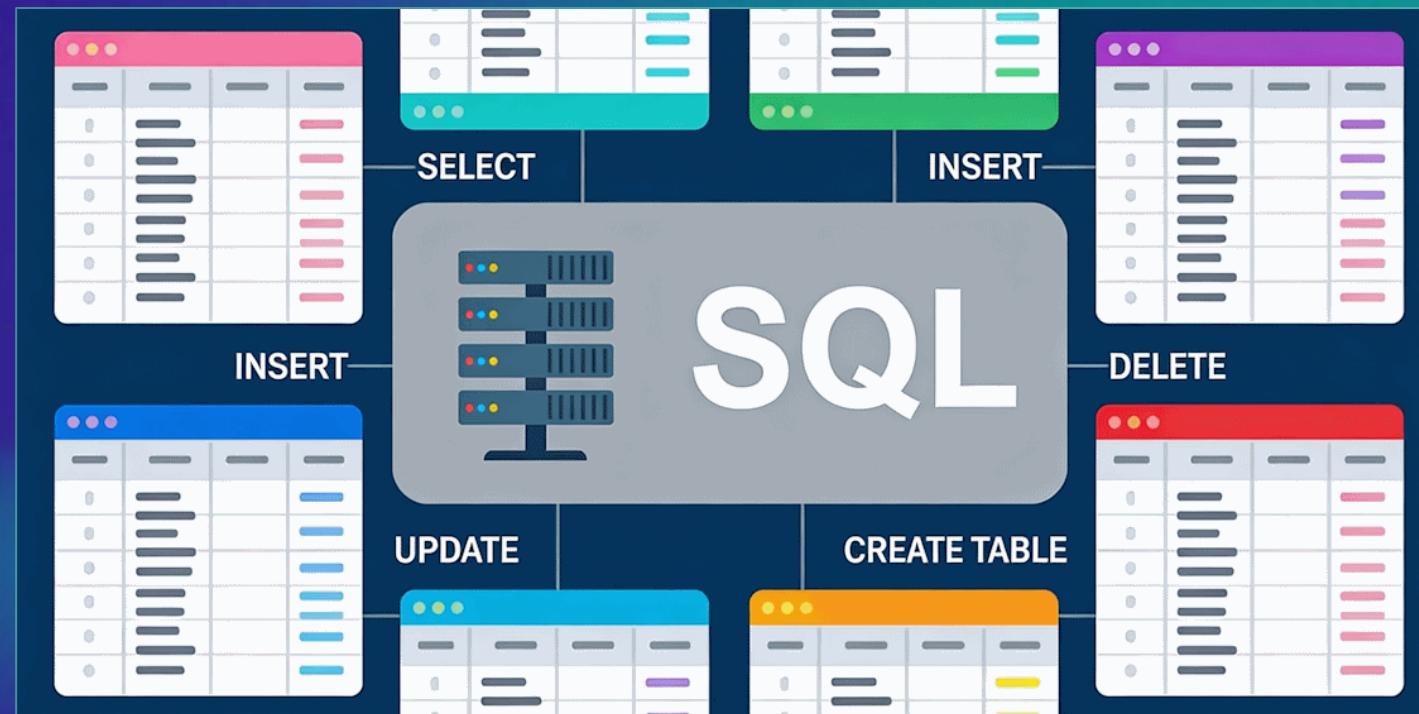


# Create a DB from SQL Script

Live Demo

# SQL Query and Edit Data

CRUD Operations, SELECT, INSERT, UPDATE, DELETE, Filtering, Joins, Aggregations, Grouping



# CRUD Operations with SQL

- DML (Data Manipulation Language)
  - CRUD Operations: **SELECT, INSERT, UPDATE, DELETE**
- SQL **queries**:
  - **SELECT** with **WHERE, ORDER BY, JOIN**, functions
  - **Joins, Grouping and Aggregations**
- Sample SQL commands:

```
SELECT name, price FROM classes
```

```
UPDATE classes SET price = price * 1.2
```

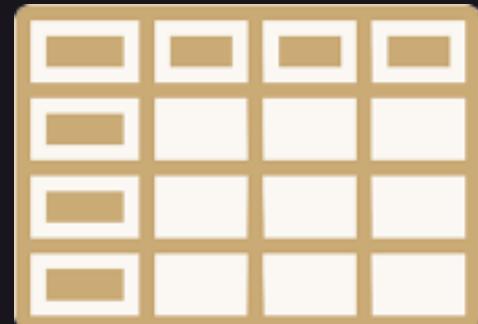
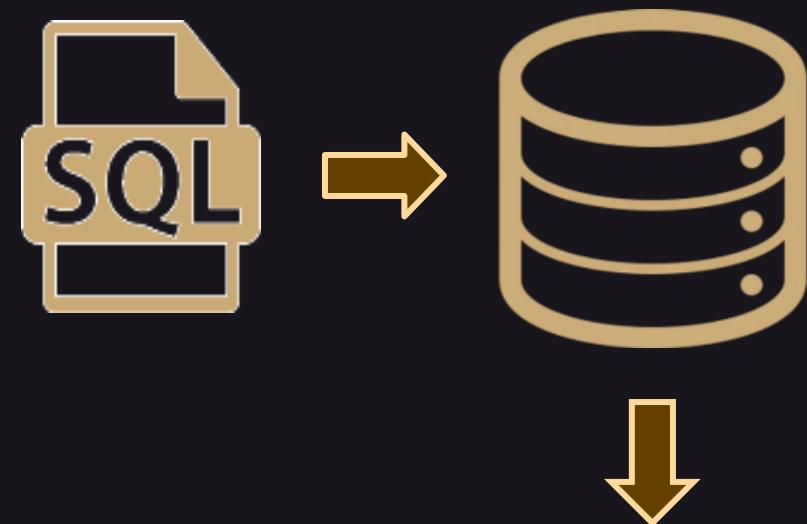
# SQL Query Execution

- Sample SQL query:

```
SELECT * FROM students
```

- The query is **executed** in the DB
  - It returns a sequence of **data rows**

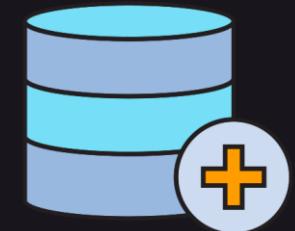
<b>id</b>	<b>name</b>	<b>email</b>	<b>created_at</b>
cd9ca4f4-7e24-4c34-8371-63430927d34e	Alice Johnson	alice.johnson@example.com	2026-01-27 11:27:40.746427
15b550ef-0537-4209-9fdb-688a1e007382	Bob Smith	bob.smith@example.com	2026-01-27 11:27:40.746427
89fb4905-876b-4d81-9e69-8af2977cd306	Carol Williams	carol.williams@example.com	2026-01-27 11:27:40.746427
6bb294b4-ac61-4e30-a518-1ffcd13b1558	David Brown	david.brown@example.com	2026-01-27 11:27:40.746427
3cacfd4f-8b03-404a-ba6c-13a914a3e174	Emma Davis	emma.davis@example.com	2026-01-27 11:27:40.746427
b7ad8fcf-1208-4a8d-8957-b917021b59e4	Frank Miller	frank.miller@example.com	2026-01-27 11:27:40.746427



# SQL INSERT, UPDATE, DELETE

- **Insert** a data row in a DB table:

```
INSERT INTO students(name, email)  
VALUES ('Steve New', 'steve.new@gmail.com')
```



- **Update** data rows in a DB table:

```
UPDATE students SET name = name || ' (old)'  
WHERE email='steve.new@gmail.com'
```



- **Delete** data rows from a DB table:

```
DELETE FROM students  
WHERE email='steve.new@gmail.com'
```

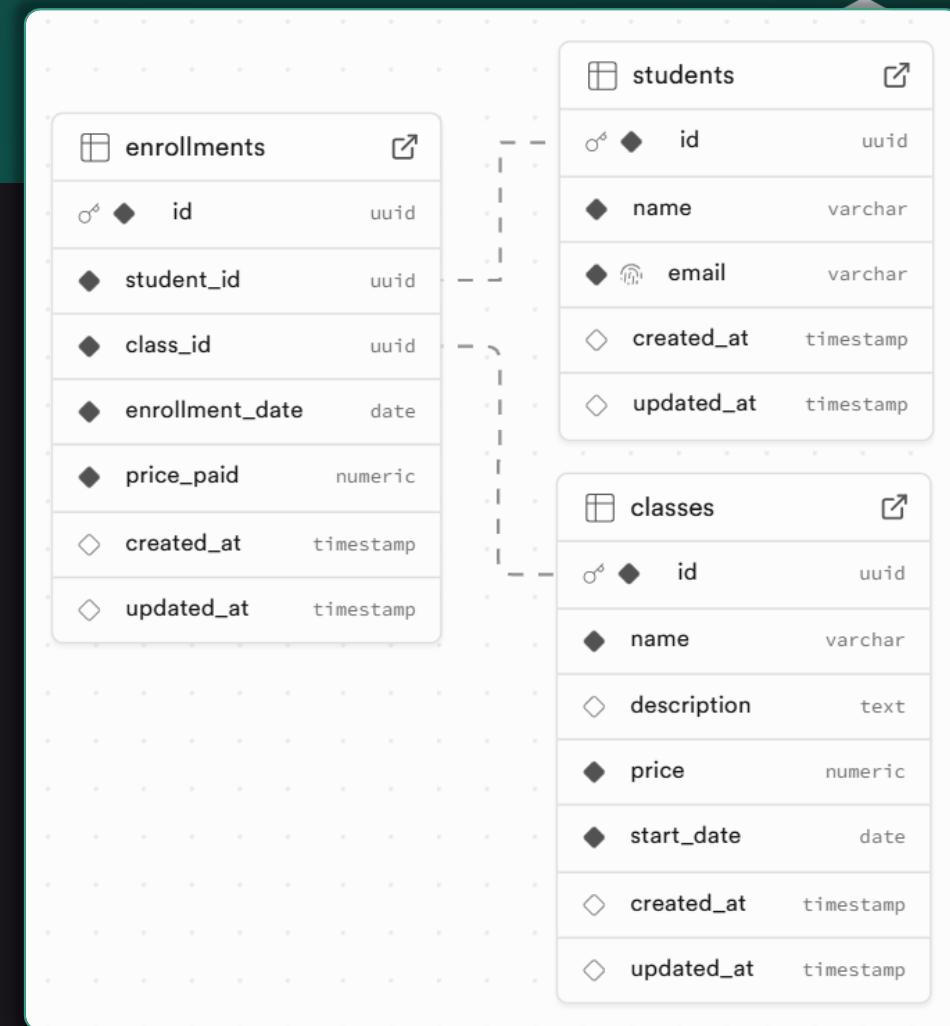


# SQL Queries with JOIN

- Joining tables (JOIN operator)

SELECT

```
s.name as student_name,  
s.email,  
c.name as class_name,  
e.enrollment_date,  
c.price,  
e.price_paid,  
(c.price - e.price_paid) as discount  
FROM enrollments e  
JOIN students s ON e.student_id = s.id  
JOIN classes c ON e.class_id = c.id  
ORDER BY s.name;
```



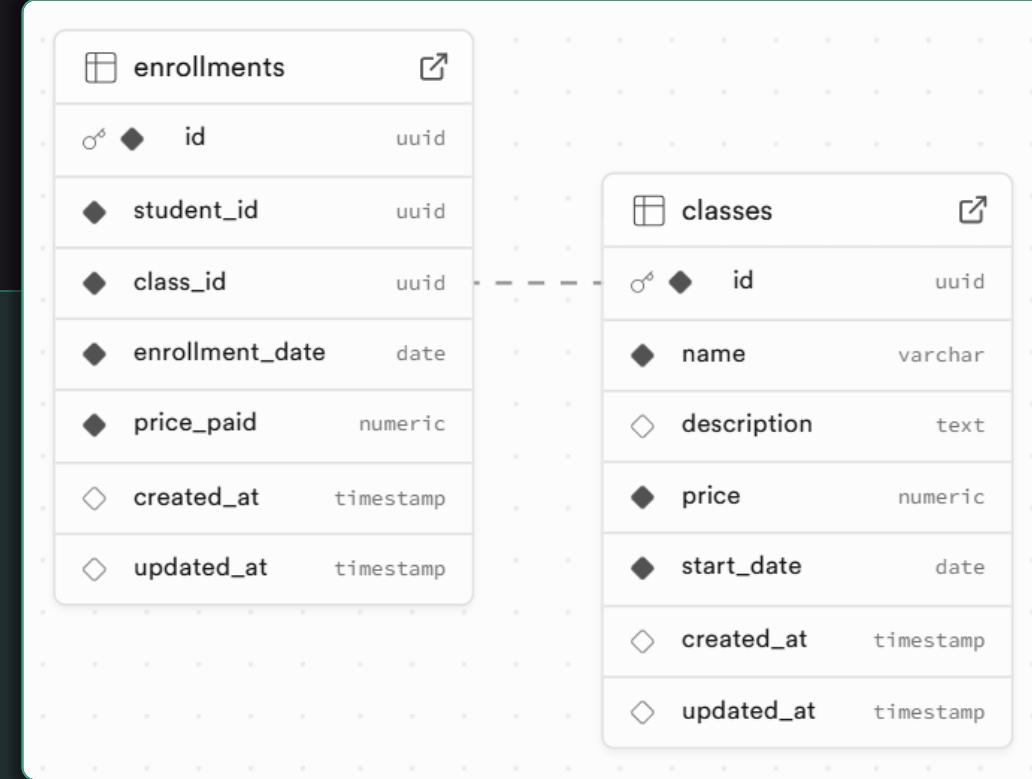
student_name	email	class_name	enrollment_date	price	price_paid	discount
Alice Johnson	alice.johnson@example	Introduction to Python	2026-01-15	119.99	89.99	30.00
Alice Johnson	alice.johnson@example	Web Development with R	2026-01-20	179.99	149.99	30.00
Bob Smith	bob.smith@example.com	Data Science Fundament	2026-01-10	239.99	199.99	40.00
Bob Smith	bob.smith@example.com	SQL Database Design	2026-01-18	107.99	79.99	28.00
Carol Williams	carol.williams@exampl	Advanced JavaScript	2026-01-12	155.99	129.99	26.00

# Grouping and Aggregations

- Group functions and aggregations with SQL

**SELECT**

```
c.name,
c.price,
COUNT(e.id) as enrollments,
SUM(e.price_paid) as revenue,
AVG(e.price_paid) as avg_price
FROM classes c
LEFT JOIN enrollments e
ON c.id = e.class_id
GROUP BY c.name, c.price;
```



name	price	enrollments	revenue	avg_price
Web Development with React	179.99	2	289.98	144.99
Cloud Computing with AWS	215.99	2	349.98	174.99
Machine Learning Basics	299.99	1	249.99	249.99

# Access Control with SQL

- **DCL** (Data Control Language)
  - Database users, roles and permissions: **GRANT / REVOKE**
- **RLS** (Row-level security)
  - **CREATE POLICY, ALTER POLICY, DROP POLICY**



```
CREATE POLICY "Users can view their own exams"
ON exams FOR SELECT
USING (
    (SELECT auth.uid()) = user_id
);
```

# Exercise: SQL Commands with AI

- Given the database "**Students, Classes and Enrollments**", perform the following tasks using **AI assistance**:
  - Add** a new class (SQL **INSERT**)
  - Enroll** several students for it (SQL **INSERT**)
  - Increase the price** for existing class (SQL **UPDATE**)
  - Delete** a student with its enrollments (SQL **DELETE**)
  - Print all **students**, their # of **enrollments** and **total fees paid**
  - Find a list of **all enrollments** in the **latest 2 courses**
  - Find all **enrollments** with **revenue by weeks** (W1-2026, W2-2026, ...)

# Supabase Services

Supabase Services (DB, Auth, Storage, Edge),  
Logs, Backup, API Keys, Row-Level Security

The screenshot shows the Supabase Project Overview dashboard. At the top left, there's a back arrow, the organization name "MyOrg", and a "FREE" badge. Below the header, there are several navigation links: "Project Overview" (which is highlighted with a grey background), "Table Editor", and "SQL Editor". On the right side, there are two columns of service icons and names: "Database", "Authentication", "Storage", "Edge Functions", "Realtime", "Advisors", "Observability", "Logs", "API Docs", and "Integrations". To the right of these, under the heading "DATABASE MANAGEMENT", is a vertical list of management features: "Schema Visualizer", "Tables", "Functions", "Triggers", "Enumerated Types", "Extensions", "Indexes", and "Publications".

Service	Management Options
Database	Schema Visualizer, Tables, Functions, Triggers, Enumerated Types, Extensions, Indexes, Publications
Authentication	
Storage	
Edge Functions	
Realtime	
Advisors	
Observability	
Logs	
API Docs	
Integrations	

# Supabase Services

- Supabase is a **backend** platform (**BaaS**)
  - Provides many **services** for devs, not just a database
- Supabase DB (tables and SQL)
  - PostgreSQL (cloud-based relational database instance)
  - Tables, relationships, views, enums, indexes, SQL, stored procedures / functions, triggers, row-level security (RLS)
- REST & GraphQL APIs for developers
  - JS apps connect to Supabase and DB via these APIs
  - Auth, CRUD operations, query, filter, sort, paginate, etc.



# Supabase Services (2)

- **Supabase Auth** (users and access control) 
  - **Authentication** and **authorization** system (access control), integrated with the database with **RLS** at database level
  - Built-in **user management**: sign-up, sign-in, sessions (with JWT tokens), OAuth (external login, e. g. Google login)
  - Auth methods: email + password, OAuth, magic links
- **Supabase Storage** (images, files, documents, videos) 
  - **Buckets** (root folders), **files** (with file paths, holding **folders**)
  - File **upload**, **download**, **share**, access control (with RLS)

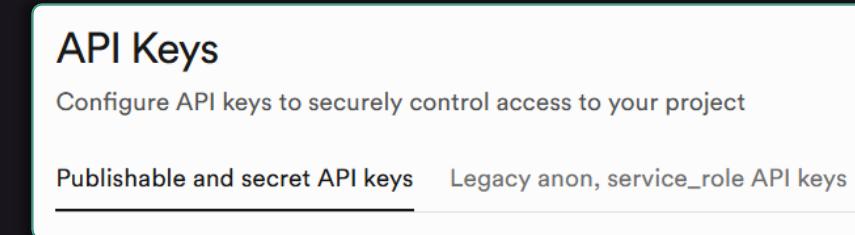
# Supabase Services (3)

- **Supabase Edge Functions** (serverless)
  - **Serverless functions**, acting as a lightweight back-end
  - Implemented in **JavaScript**, accessed through the API
  - Implement **payments**, **emails**, **webhooks** (Web callbacks)
  - Have **unrestricted DB access**, skipping the RLS policies
- **Supabase Realtime** (live stream DB changes)
  - DB emits **events** (change notifications) to **channels**, clients subscribe to channels, events are filtered by RLS
- **Supabase Logs** (track DB access, auth, storage, etc.)



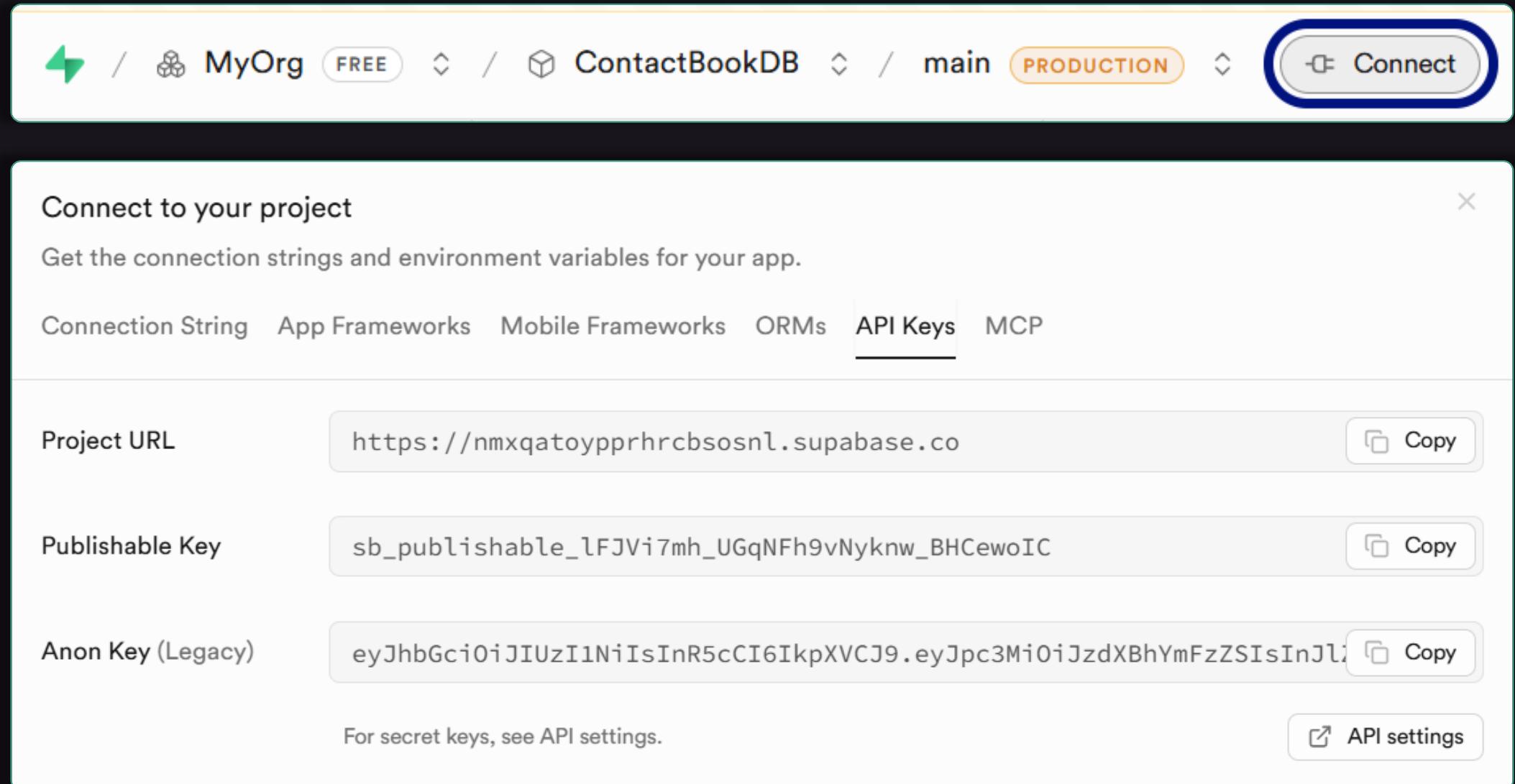
# API Keys in Supabase

- API keys in Supabase connect your app with your Supabase project



-  Publishable (anon) key
  - Restricted access key
  - For browsers / mobile apps
  - Safe to expose publicly
  - Access control based on RLS (Row-Level Security)
-  Secret (service) key
  - Full access key
  - Used in back-ends
  - Never expose publicly
  - Bypassed all access control policies (RLS)

# API Keys in Supabase – Example



The screenshot shows the Supabase Connect interface. At the top, there's a navigation bar with a green arrow icon, a folder icon labeled "MyOrg", a "FREE" badge, a dropdown menu, a cube icon labeled "ContactBookDB", another dropdown menu, and "main PRODUCTION". On the far right is a "Connect" button with a blue outline, which is circled in red in the screenshot. Below the navigation bar, a modal window titled "Connect to your project" is open. It contains the text "Get the connection strings and environment variables for your app." followed by a horizontal navigation bar with tabs: "Connection String", "App Frameworks", "Mobile Frameworks", "ORMs", "API Keys" (which is underlined and bolded), and "MCP". The main content area displays three API keys: "Project URL" with the value "https://nmxqatoypyprhrcbsosnl.supabase.co" and a "Copy" button; "Publishable Key" with the value "sb\_publishable\_lFJVi7mh\_UGqNFh9vNyknw\_BHCewoIC" and a "Copy" button; and "Anon Key (Legacy)" with the value "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJzdXBhYmFzZSIsInJl..." and a "Copy" button. At the bottom of the modal, there's a note "For secret keys, see API settings." and a "Copy" button for the API settings.

Connect to your project X

Get the connection strings and environment variables for your app.

Connection String App Frameworks Mobile Frameworks ORMs **API Keys** MCP

Project URL <https://nmxqatoypyprhrcbsosnl.supabase.co> Copy

Publishable Key [sb\\_publishable\\_lFJVi7mh\\_UGqNFh9vNyknw\\_BHCewoIC](#) Copy

Anon Key (Legacy) [eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJzdXBhYmFzZSIsInJl...](#) Copy

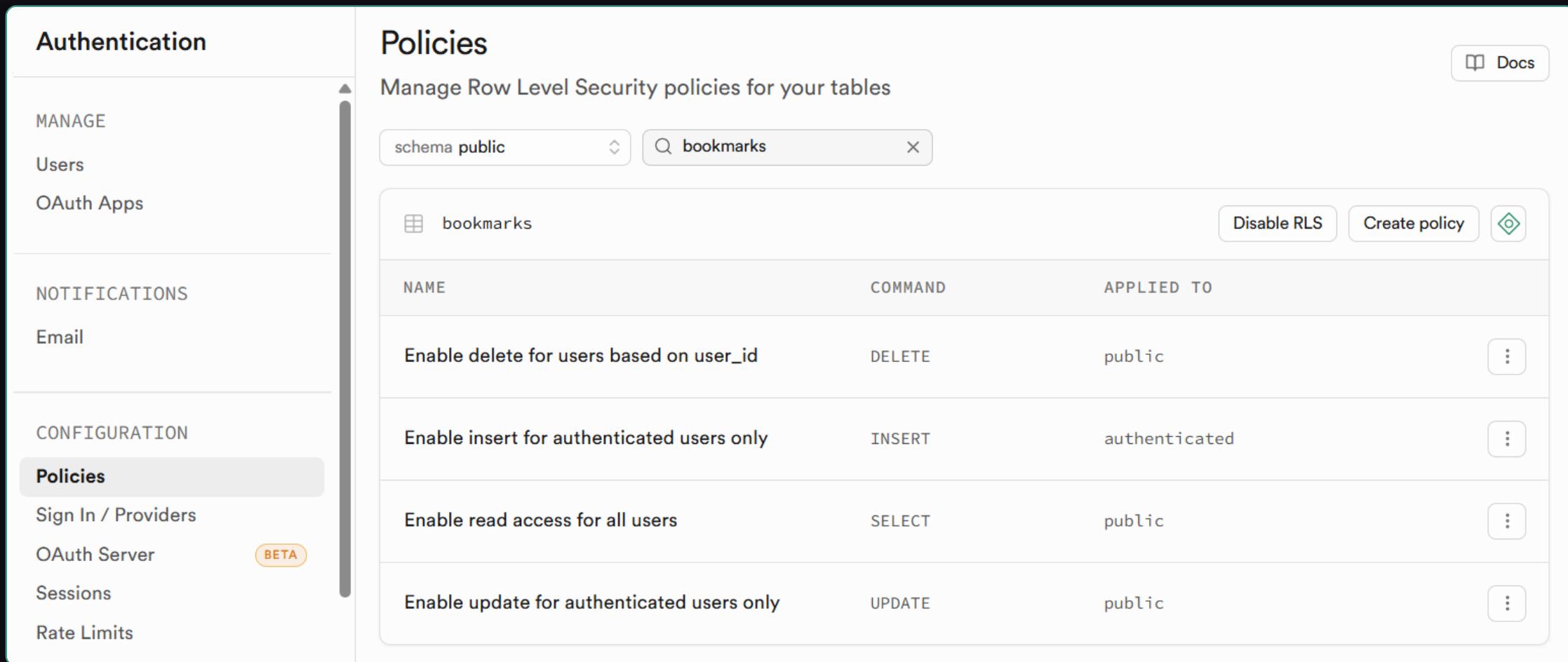
For secret keys, see API settings. Copy

# Row-Level Security

- **Row-Level Security** (RLS) define which rows a user can read or write (access control)
  - Defined by **security policies** in the database (SQL code)
  - Based on the **logged-in user**: anonymous / authenticated
- Typical RLS policy:
  - **Anonymous** users: can **read all public data**, e. g. products
  - **Authenticated users**: can read / write their **own data**, e. g. own products, own messages, own user profile
  - **Admin users**: can read / write **any data** in the DB

# Sample RLS Policies

- Sample RLS policies in Supabase:



The screenshot shows the Supabase Policies interface. On the left, a sidebar menu includes Authentication (Manage Users, OAuth Apps), Notifications (Email), Configuration (Policies, Sign In / Providers, OAuth Server, Sessions, Rate Limits). The Policies section is currently selected, indicated by a grey background. A 'BETA' badge is visible next to the OAuth Server item. The main area is titled 'Policies' with the subtitle 'Manage Row Level Security policies for your tables'. It shows a table for the 'bookmarks' schema with four rows of RLS policies.

NAME	COMMAND	APPLIED TO
Enable delete for users based on user_id	DELETE	public
Enable insert for authenticated users only	INSERT	authenticated
Enable read access for all users	SELECT	public
Enable update for authenticated users only	UPDATE	public

# Supabase Pricing and Free Tier

- Supabase has **free** and **paid** subscription plans
- Supabase **free plan** (for hobby projects):
  - Max **2 active projects**, with minimal computing resources
  - Projects automatically **pause** after ~1 week of inactivity
  - Manual backup (not easy)
- Supabase **paid plans** (for real projects):
  - \$25 / month for 1 project (more projects: \$10+ / month)
  - Supabase pricing: <https://supabase.com/pricing>

# Supabase Self-Hosting

- Supabase can be **self-hosted** on a VPS server
  - It's an **open-source project** → you can deploy your own Supabase instance for free
  - **Technical skills** are needed (Docker and Docker Compose)
  - **Highly reduced costs**, but manual setup + maintenance
  - You can find a VPS provider here:  
[https://www.vpsbenchmarks.com/best\\_vps/2026](https://www.vpsbenchmarks.com/best_vps/2026)
  - Or use template-based deployment, e.g.  
<https://hostinger.com/vps/supabase-hosting>

# JS Apps with Supabase

Connecting JS Apps with Supabase: API Keys, Supabase API, CRUD Operations

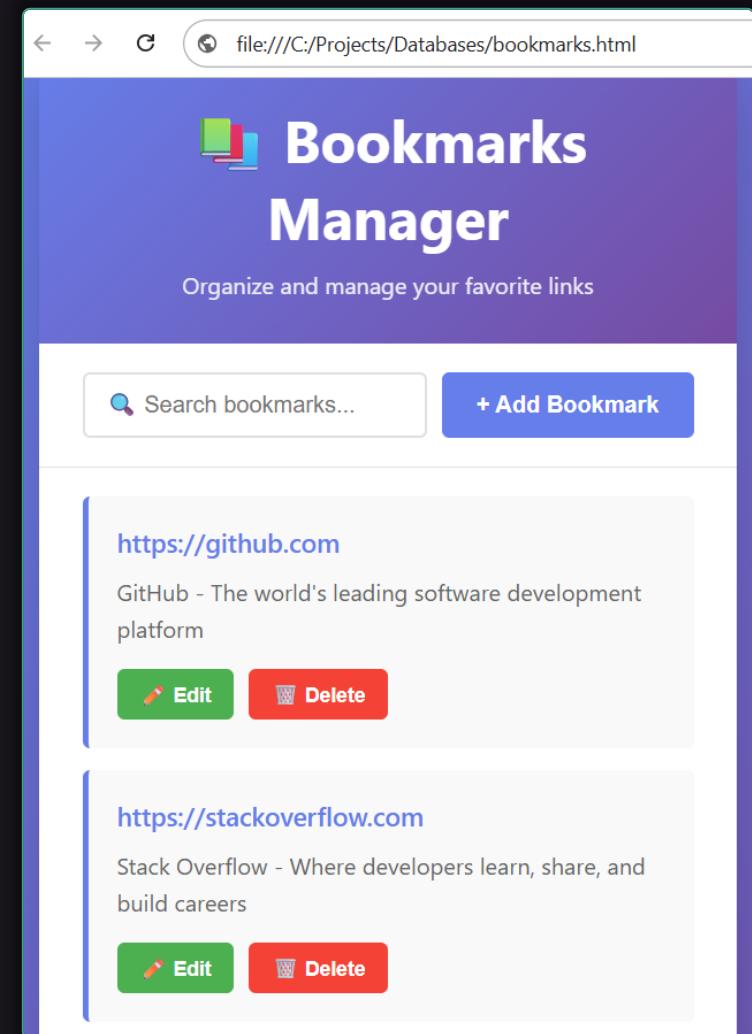


# Building an App with Database

- Steps to build an **app with a database** (DB):
  1. Build the **app functionality** without a database
    - Initially keep all app data in the memory
  2. Connect a database & **configure a DB connection**
    - Choose an existing DB / create a new DB
  3. Update the app code to **store app data in the DB**
    - Sample prompt: "*Update the app to store its data in DB*"
  4. **Run** and **test** the app DB operations
    - Check if app data is stored correctly in the DB tables

# Example: Bookmarks without a DB

- Step 1: Create the **app without DB**
  - Create a **Bookmarks app**: view list of bookmarks, search, create, edit, delete
    - Each bookmark has: **URL, comments**
    - Use **popup dialogs** to add / edit / confirm delete bookmarks
    - Keep app data in the **local storage**
      - Initially add a few sample bookmarks
      - Use simple **HTML + CSS + JS**



# Example: Bookmarks with DB

- Step 2: **Connect to Supabase**
  - Update the app to **store its data in Supabase**
  - Create Supabase **SQL script** to define the DB tables
    - Initially, insert a few **sample data** in the DB
    - Allow everyone to freely read / modify data **without auth**
    - **Note:** generally, this is **unsafe** (we'll add auth later)
  - Connect the **Supabase client library** with **API key**
  - Implement a "**loading**" **spinner** at the top of the screen, displayed for DB waits + **toast notifications**

# Example: Bookmarks SQL Script

- Step 3. Run the SQL script to create the tables in Supabase

**SQL** Bookmarks Table with RLS and Timestamps

```

1 -- Supabase Setup Script for Bookmarks App
2 -- This script creates the bookmarks table and its
3 -- associated RLS policy.
4 -- Create the bookmarks table
5 CREATE TABLE IF NOT EXISTS bookmarks (
6   id BIGSERIAL PRIMARY KEY,
7   url TEXT NOT NULL,
8   comments TEXT,
9   created_at TIMESTAMPTZ DEFAULT NOW(),
10  updated_at TIMESTAMPTZ DEFAULT NOW()
11 );

```

bookmarks

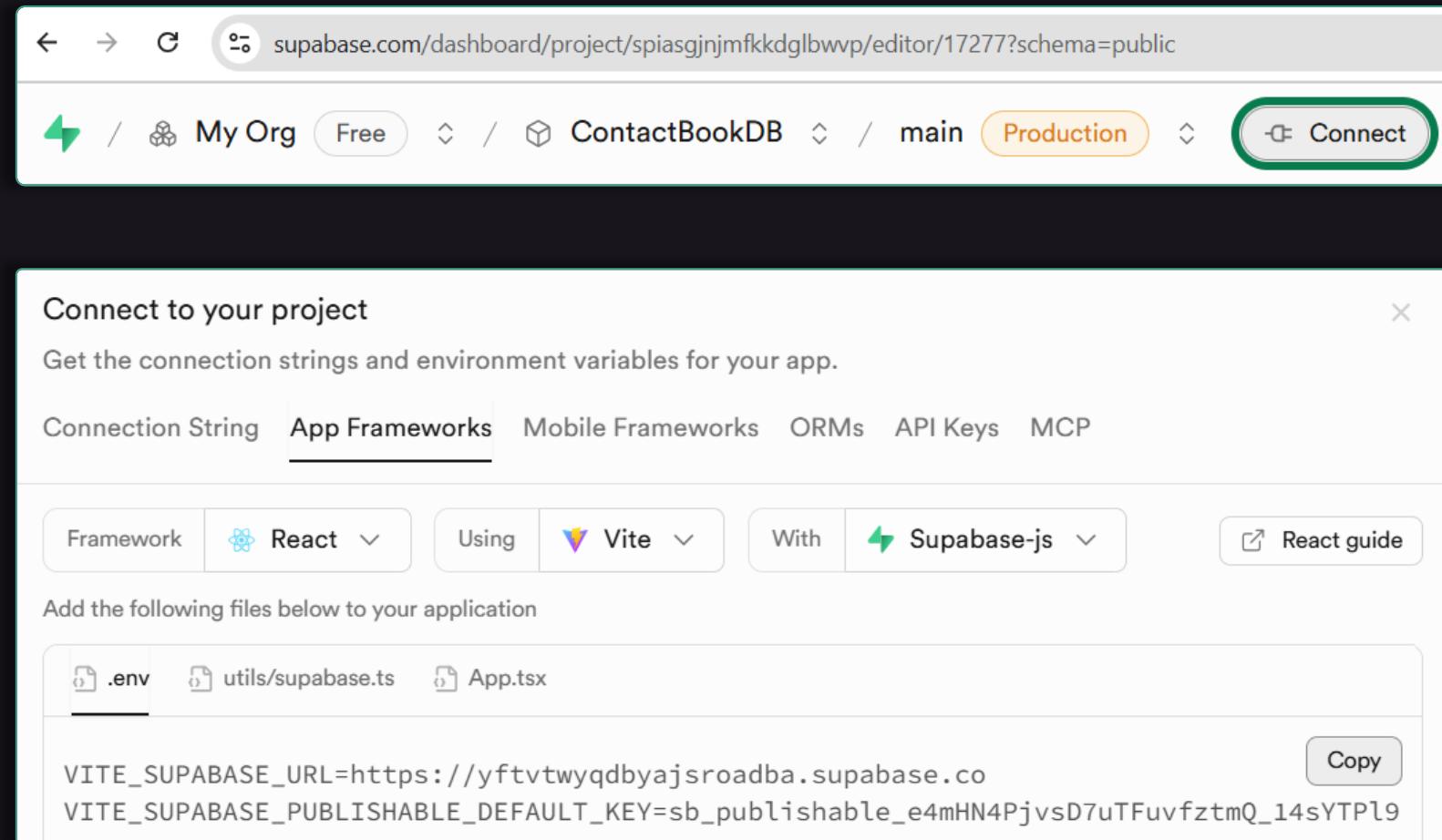
	id	url	comments
1	1	<a href="https://github.com">https://github.com</a>	GitHub - The world's leading software development community
2	2	<a href="https://stackoverflow.com">https://stackoverflow.com</a>	Stack Overflow - Where developers learn, share, and contribute
3	3	<a href="https://developer.mozilla.org">https://developer.mozilla.org</a>	MDN Web Docs - Resources for developers
4	4	<a href="https://www.w3schools.com">https://www.w3schools.com</a>	W3Schools - Online web tutorials
5	5	<a href="https://css-tricks.com">https://css-tricks.com</a>	CSS-Tricks - Tips, tricks, and techniques on CSS

Results Explain Chart Export ⋮ Source Primary Database Role postgres Run CTRL ⌘

Success. No rows returned

# Connecting Apps to Supabase

- To connect to Supabase BaaS project, you need an **API key**:
  - Server URL**
  - Publishable key**  
(anonymous API key)
- Dashboard  
→ Connect  
→ App Frameworks  
→ React  
→ Vite



The screenshot shows the Supabase Connect interface. At the top, there's a browser-like header with the URL `supabase.com/dashboard/project/spiasgjnfmkkdglbwvp/editor/17277?schema=public`. Below it is a navigation bar with tabs for `My Org`, `Free`, `ContactBookDB`, `main`, `Production` (which is highlighted), and a `Connect` button.

The main content area is titled "Connect to your project" with the sub-instruction "Get the connection strings and environment variables for your app." It features a tab bar with "Connection String" (disabled), "App Frameworks" (selected), "Mobile Frameworks", "ORMs", "API Keys", and "MCP".

Under "App Frameworks", there are dropdowns for "Framework" (React), "Using" (Vite), "With" (Supabase.js), and a "React guide" link.

A section titled "Add the following files below to your application" lists ".env", "utils/supabase.ts", and "App.tsx".

At the bottom, two environment variables are listed:  
`VITE_SUPABASE_URL=https://yftvtwyqdbya.sroadba.supabase.co`  
`VITE_SUPABASE_PUBLISHABLE_DEFAULT_KEY=sb_publishable_e4mHN4PjvsD7uTFuvfztmQ_14sYTP19`

A "Copy" button is located in the bottom right corner of the variable list.

# Example: Supabase API Key

- Step 4: Update Supabase connection settings
  - Update the Supabase URL + anon key in the JS code

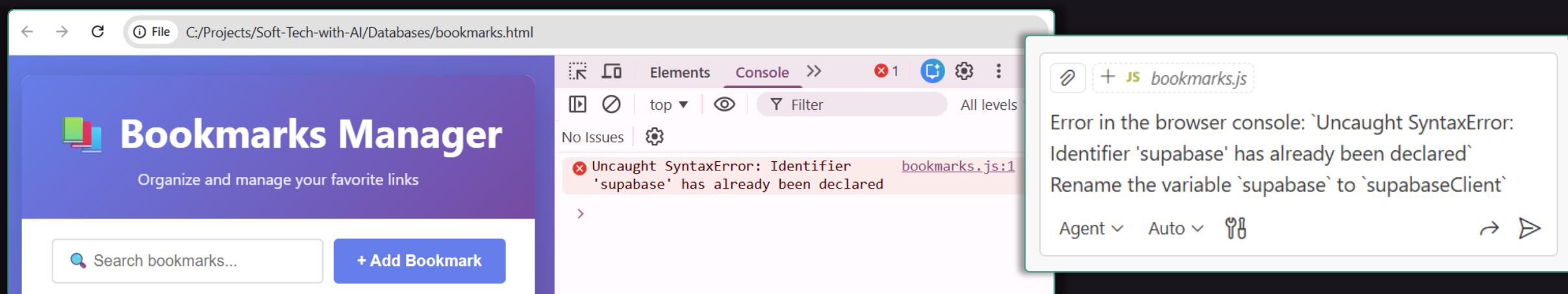
JS bookmarks.js > [🔗] SUPABASE\_URL

```
1 // =====  
2 // SUPABASE CONFIGURATION  
3 // =====  
4 // TODO: Replace these with your actual Supabase project credentials  
5 // Find these in your Supabase project settings -> API  
6 const SUPABASE_URL = 'YOUR_SUPABASE_URL_HERE';  
7 const SUPABASE_ANON_KEY = 'YOUR_SUPABASE_ANON_KEY_HERE';
```

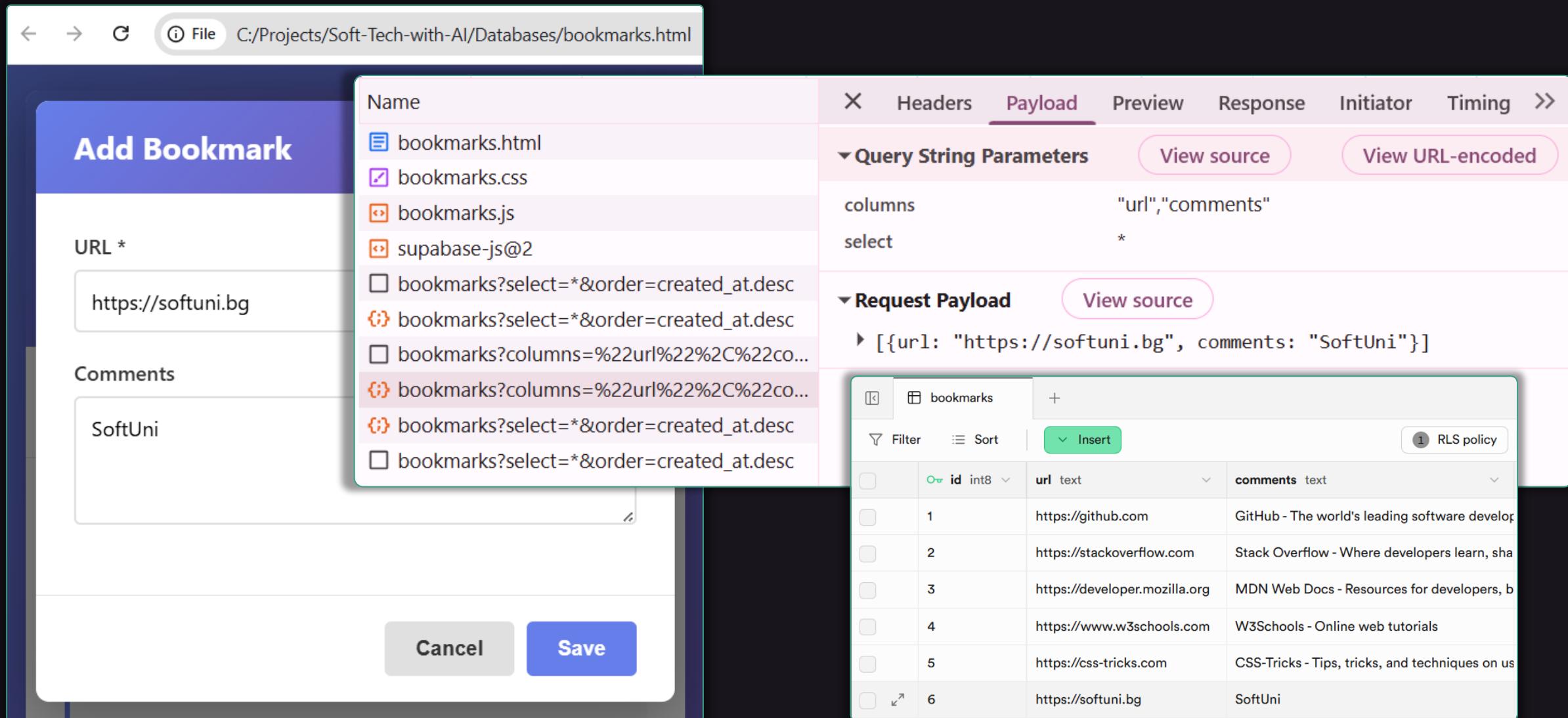
```
6 const SUPABASE_URL = 'https://yftvtwyqdbuja.sroadba.supabase.co';  
7 const SUPABASE_ANON_KEY = 'sb_publishable_e4mHN4PjvsD7uTFuvfztmQ_14sYTP19';
```

# Example: Testing and Bug Fixing

- Step 5. **Testing and bug fixing** the Supabase integration
  - Sometimes AI writes wrong JS code to connect to Supabase
  - Error in the browser console: `Uncaught SyntaxError: Identifier 'supabase' has already been declared`
  - Fix: rename the variable `supabase` to `supabaseClient`



# Example: Bookmarks App in Action



The screenshot illustrates a bookmarks application in action. On the left, a modal dialog titled "Add Bookmark" is open. It contains fields for "Name" (with suggestions like "bookmarks.html", "bookmarks.css", "bookmarks.js", and "supabase-js@2"), "URL \*" (set to "https://softuni.bg"), and "Comments" (set to "SoftUni"). At the bottom are "Cancel" and "Save" buttons. In the top right corner of the browser window, a Network tab from a developer tool shows a POST request to "bookmarks.html". The "Payload" section of the Network tab displays the JSON data being sent: 

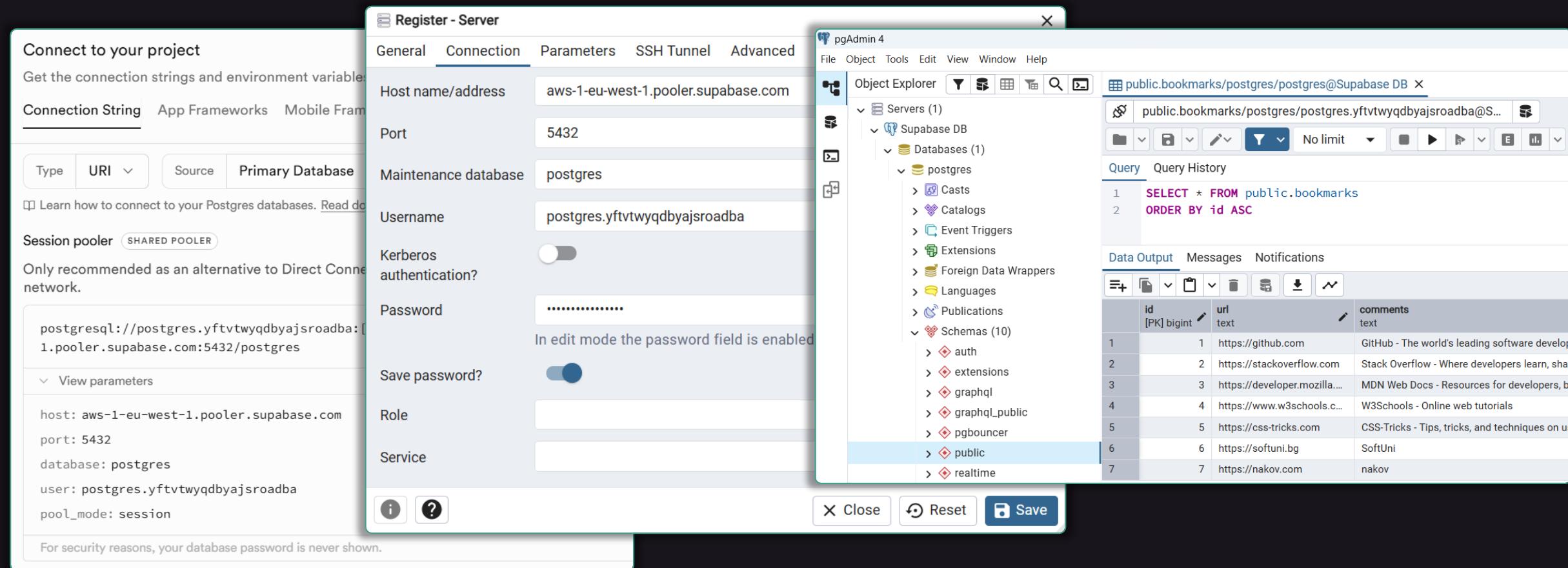
```
[{"url": "https://softuni.bg", "comments": "SoftUni"}]
```

. To the right of the Network tab, a database table named "bookmarks" is visible, showing the following data:

	id	url	comments
	1	https://github.com	GitHub - The world's leading software development platform
	2	https://stackoverflow.com	Stack Overflow - Where developers learn, share, and build things
	3	https://developer.mozilla.org	MDN Web Docs - Resources for developers, builders, and learners
	4	https://www.w3schools.com	W3Schools - Online web tutorials
	5	https://css-tricks.com	CSS-Tricks - Tips, tricks, and techniques on using CSS
	6	https://softuni.bg	SoftUni

# Direct Connection to Supabase DB

- You can connect to your Supabase DB with a standard **PostgreSQL admin tools** like **pgAdmin**, **HeidiSQL**, etc.



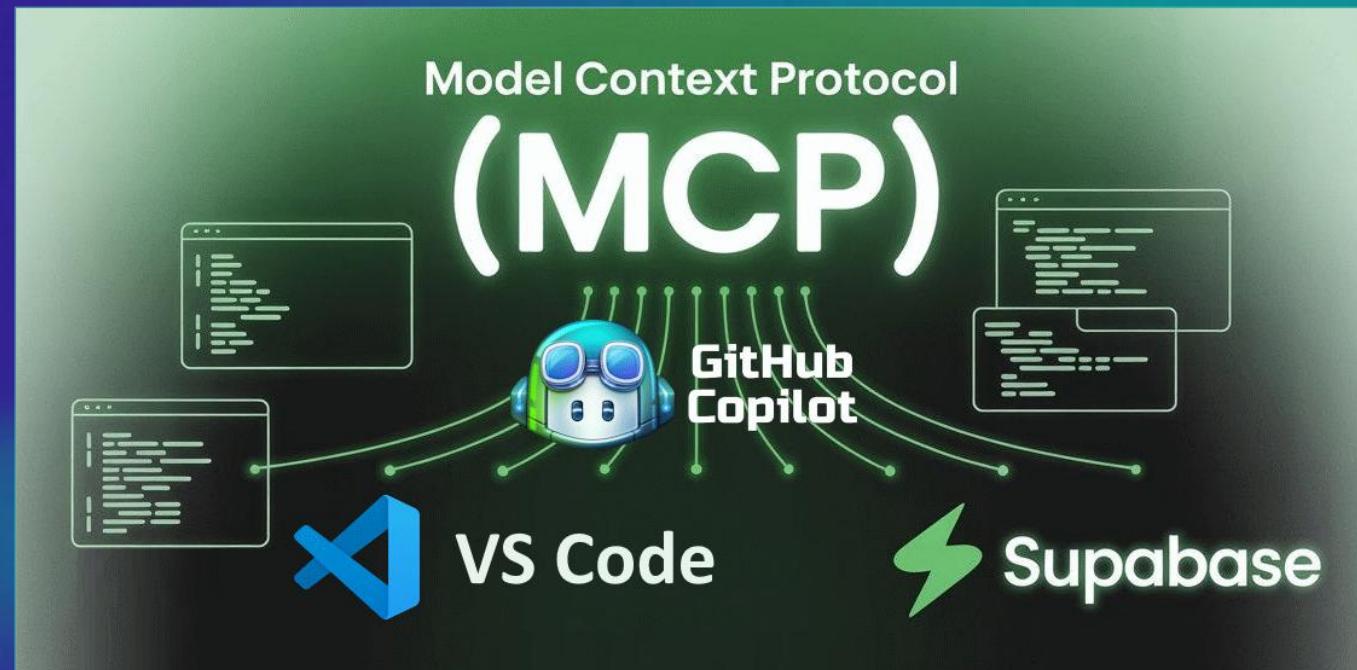
The image shows two windows side-by-side. On the left is a 'Register - Server' dialog box from pgAdmin 4. It has tabs for General, Connection, Parameters, SSH Tunnel, and Advanced. The Connection tab is selected, showing fields for Host name/address (aws-1-eu-west-1.pooler.supabase.com), Port (5432), Maintenance database (postgres), Username (postgres.yftvtywyqdbajsroadba), and Password (redacted). A note says 'In edit mode the password field is enabled'. There's also a 'Save password?' toggle and Role and Service fields. At the bottom are Close, Reset, and Save buttons. Below the dialog is a snippet of the connection string: postgresql://postgres.yftvtywyqdbajsroadba@aws-1-eu-west-1.pooler.supabase.com:5432/postgres. A note at the bottom says 'For security reasons, your database password is never shown.'

On the right is the pgAdmin 4 main interface. The Object Explorer shows a tree structure: Servers (1) > Supabase DB > Databases (1) > postgres > various objects like Casts, Catalogs, Event Triggers, Extensions, Foreign Data Wrappers, Languages, Publications, Schemas (10), auth, extensions, graphql, graphql\_public, pgbouncer, public, and realtime. The public schema is currently selected. The Query tab contains a SQL query: `SELECT * FROM public.bookmarks ORDER BY id ASC`. The Data Output tab shows the results of this query:

id [PK] bigint	url text	comments text
1	https://github.com	GitHub - The world's leading software development platform
2	https://stackoverflow.com	Stack Overflow - Where developers learn, share, and contribute
3	https://developer.mozilla.org	MDN Web Docs - Resources for developers, by developers
4	https://www.w3schools.com	W3Schools - Online web tutorials
5	https://css-tricks.com	CSS-Tricks - Tips, tricks, and techniques on CSS
6	https://softuni.bg	SoftUni
7	https://nakov.com	nakov

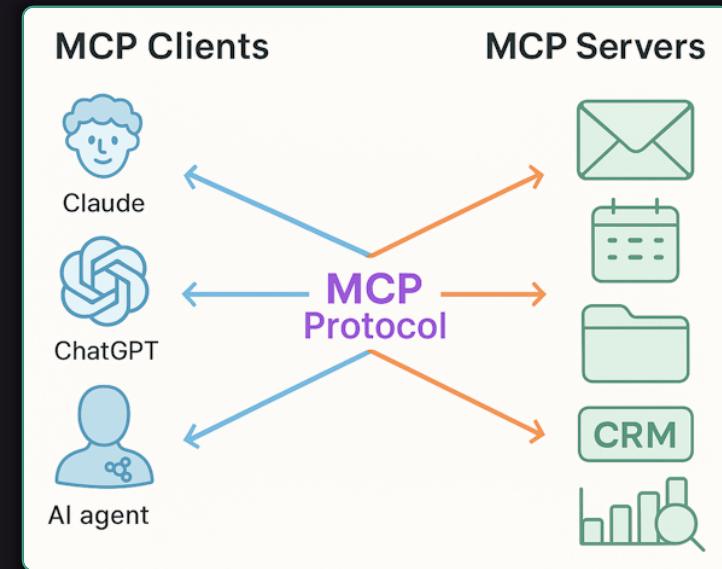
# Supabase MCP

MCP (Model Context Protocol), Install and Configure Supabase MCP for Certain Project



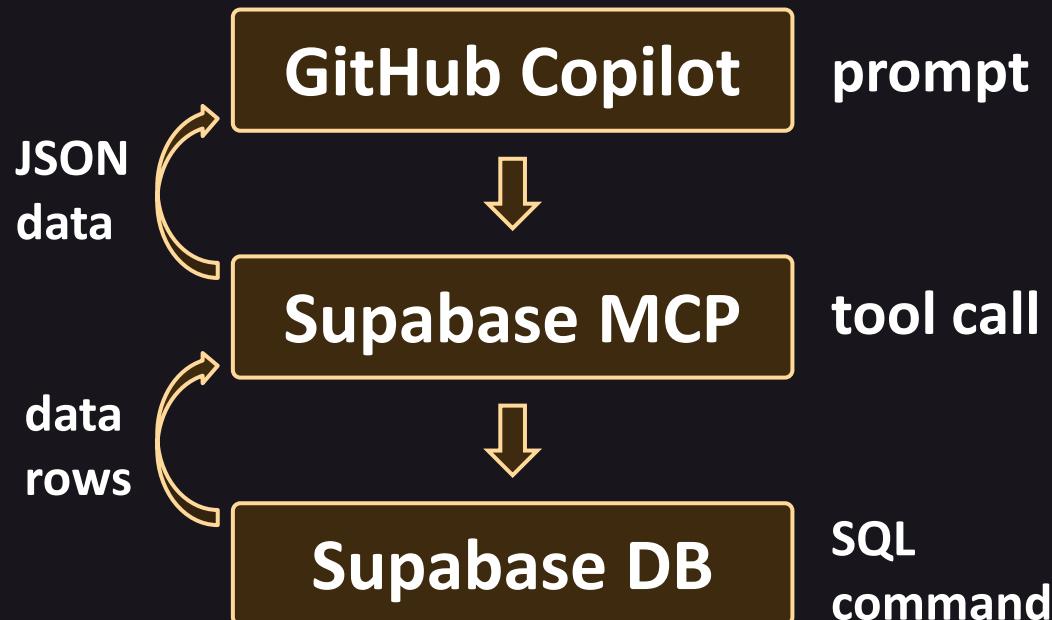
# Intro to MCP

- What is **MCP** (Model Context Protocol)?
  - Open standard for connecting AI models to **external tools, data sources** and **APIs**
- Examples of **MCP servers**:
  - Connect Claude Desktop with **Notion**
  - Connect GitHub Copilot with **Supabase** → browse the DB
  - Connect ChatGPT with **Booking.com** → find and book hotels
  - Connect ChatGPT with a **GMail** account → find / send emails
  - Connect an AI agent with your **internal CRM system**



# Supabase MCP

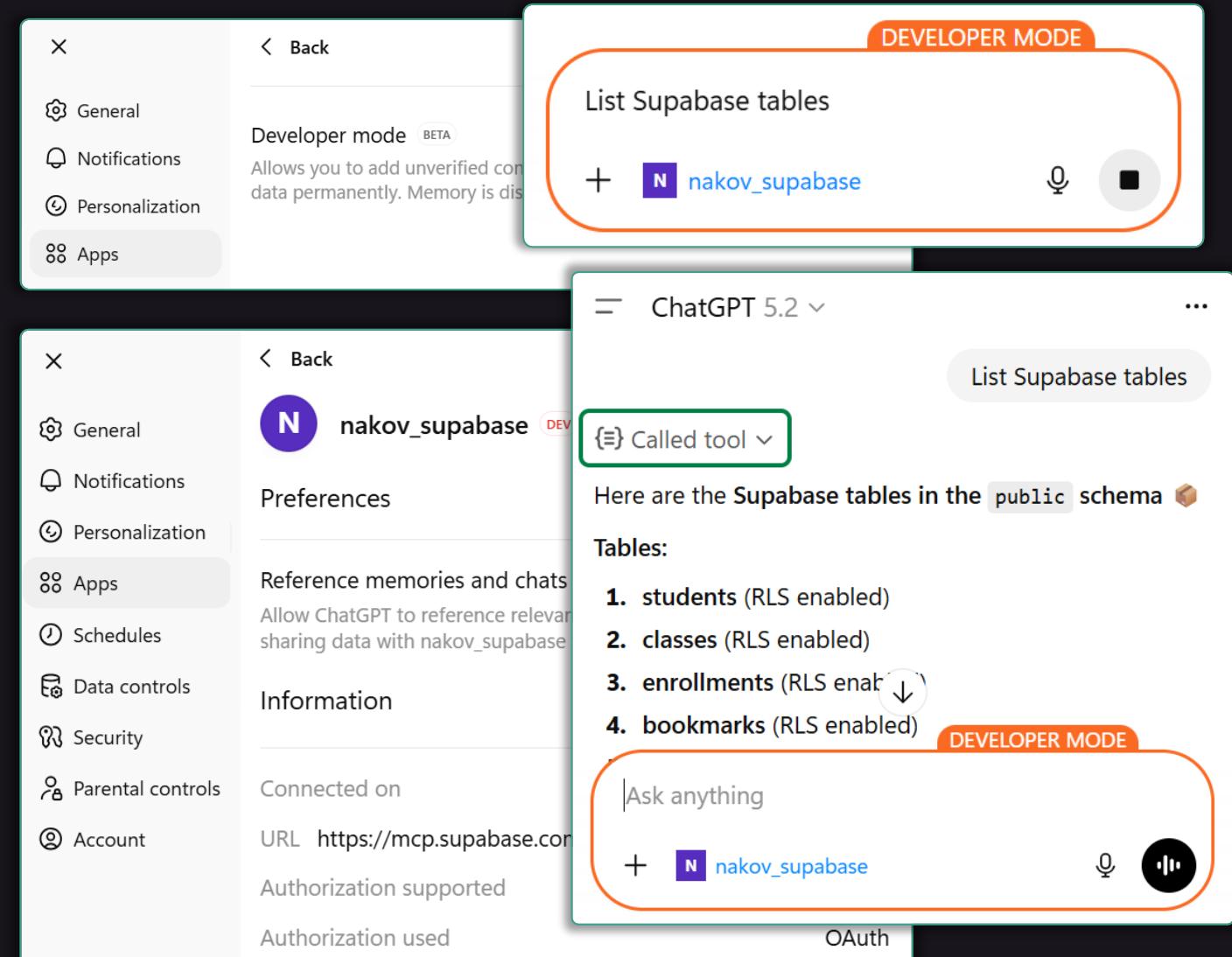
- Supabase MCP connects your **AI Dev Agent** (GitHub Copilot) to your **Supabase** instance via an MCP server



- Supabase provides an **MCP server** for your projects

# Supabase MCP in ChatGPT

- ChatGPT supports the **MCP protocol**
  - Can connect to your Supabase DB
- Step 1: Enable **dev mode**
- Step 2: **Create an app** (MCP configuration)
- Step 3: **AI prompt** with your app



The image displays a screenshot of the Supabase MCP configuration in ChatGPT. At the top left, there's a navigation bar with 'General', 'Notifications', 'Personalization', and 'Apps' (which is selected). Below it, a 'Developer mode' section is shown with a 'BETA' label, explaining that it allows adding unverified data permanently. To the right, a large orange-bordered box highlights the 'nakov\_supabase' app entry in the list of apps.

On the right side, the ChatGPT interface shows a message from 'ChatGPT 5.2' stating: 'Here are the Supabase tables in the public schema'. It lists four tables: 'students' (RLS enabled), 'classes' (RLS enabled), 'enrollments' (RLS enabled), and 'bookmarks' (RLS enabled). Below this, another orange-bordered box highlights the 'Ask anything' input field and the 'nakov\_supabase' app entry in the OAuth section.

# Installing Supabase MCP in VS Code



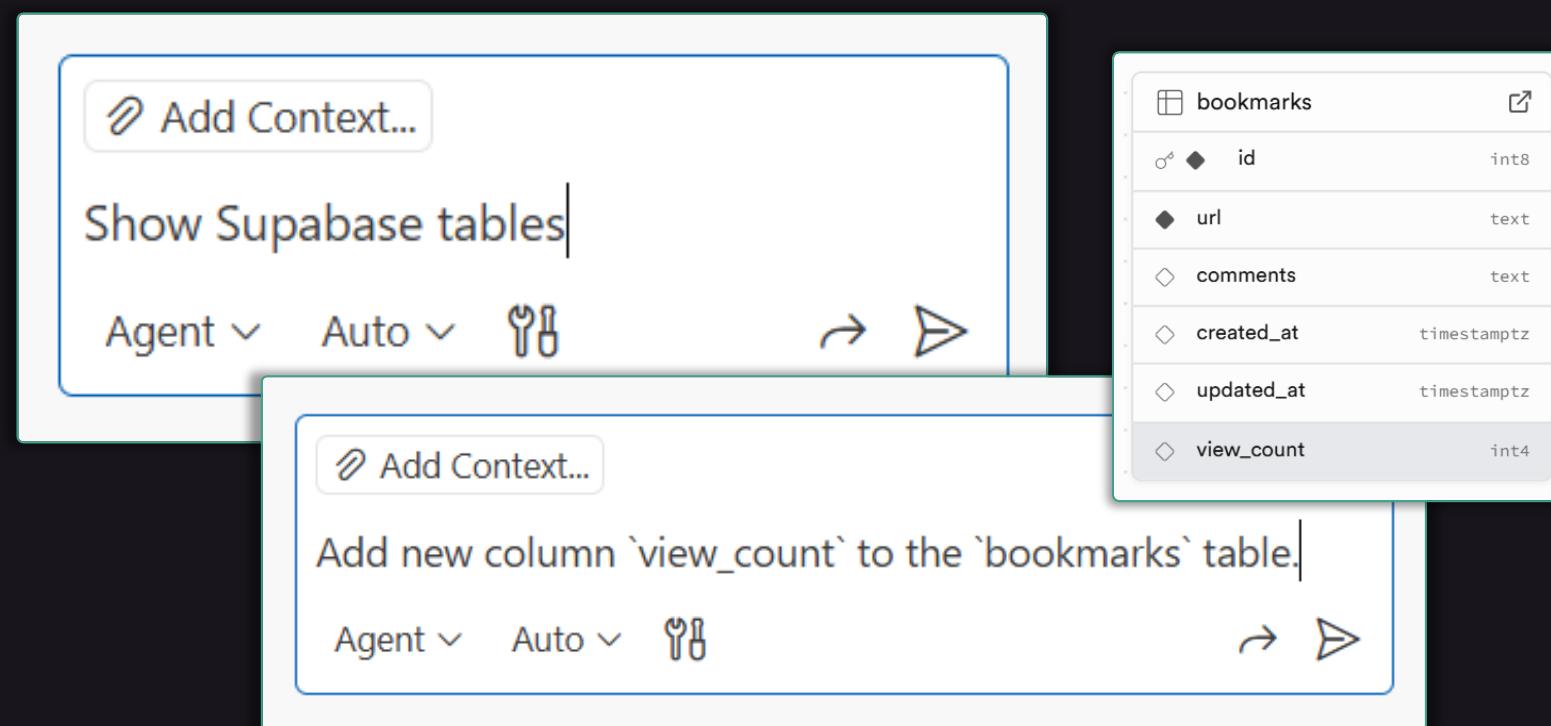
The image shows two side-by-side screenshots. On the left is the Supabase MCP (Metadata Configuration Platform) web interface. The URL in the browser is `supabase.com/dashboard/project/spiasgjnjmffkkdglbwvp/editor/17277?schema=public`. The project is named "ContactBookDB" and is set to "Production". A "Connect" button is highlighted with a green oval. Below this, a modal window titled "Connect to your project" is open, showing tabs for "Connection String", "App Frameworks", "Mobile Frameworks", "ORMs", and "MCP", with "MCP" selected. It displays "Options" and "Feature Groups" sections, with "All features except Storage enabled by default" selected. At the bottom, the "Server URL" is listed as `https://mcp.supabase.com/mcp?project_ref=nmxqatoypyprhrcbsosnl`. On the right is the "MCP Server: supabase" extension in the VS Code Marketplace. The extension icon is a black paperclip. The name "supabase" is listed, with "Install in Workspace" highlighted with a green oval and a green checkmark. Below this is the "CONFIGURATION" section with fields: "Name: supabase", "Type: http", and "URL: https://mcp.supabase.com/mcp?project\_ref=yftvtwyqdbajsroadba". A large green arrow points from the "Install in Workspace" button in the Marketplace to the "Install in Workspace" button in the configuration section. At the bottom of the configuration panel, there is sample JSON code for `.vscode/mcp.json`:

```
1  {
2    "servers": {
3      "supabase": {
4        "type": "http",
5        "url": "https://mcp.supabase.com/mcp?project_ref=nmxqatoypyprhrcbsosnl"
6      }
7    }
8  }
```

At the very bottom of the configuration panel, there is a link "Need help? View VS Code docs" with a green arrow pointing to it.

# Using the Supabase MCP

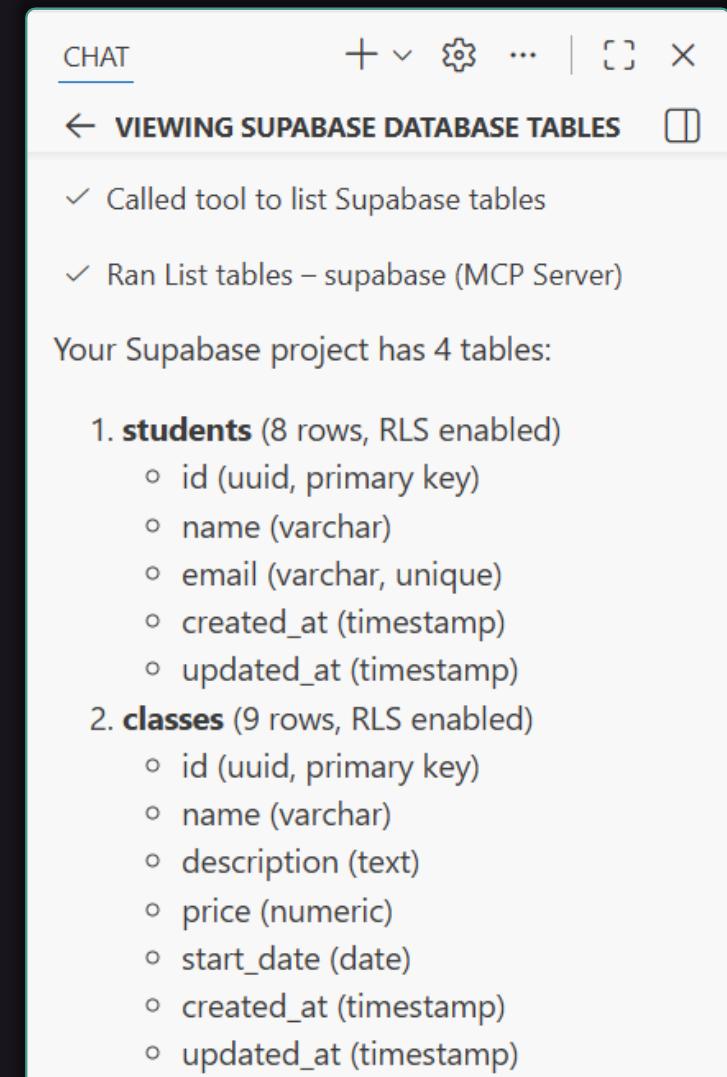
- Once the **Supabase MCP** is installed and connected, you can **ask the AI agent** to **interact with Supabase**



The image shows the Supabase MCP interface. On the left, there's a panel titled "Show Supabase tables" with an "Add Context..." button and dropdowns for "Agent" and "Auto". On the right, a modal window displays the schema for the "bookmarks" table:

	bookmarks	
◆ id	int8	
◆ url	text	
◇ comments	text	
◇ created_at	timestamptz	
◇ updated_at	timestamptz	
◇ view_count	int4	

Below the table, a message says "Add new column `view\_count` to the `bookmarks` table." with "Agent" and "Auto" dropdowns and a preview icon.



The image shows a "CHAT" interface titled "VIEWING SUPABASE DATABASE TABLES". The history includes:

- Called tool to list Supabase tables
- Ran List tables – supabase (MCP Server)

Your Supabase project has 4 tables:

- students** (8 rows, RLS enabled)
  - id (uuid, primary key)
  - name (varchar)
  - email (varchar, unique)
  - created\_at (timestamp)
  - updated\_at (timestamp)
- classes** (9 rows, RLS enabled)
  - id (uuid, primary key)
  - name (varchar)
  - description (text)
  - price (numeric)
  - start\_date (date)
  - created\_at (timestamp)
  - updated\_at (timestamp)

# Exercise: Polls App with MCP



- We want to create a **Polls app**: visitors view polls and vote
  - All app data stored in **Supabase** database, public access
- Step 1: Create a **Supabase project** and configure the **Supabase MCP**
  - Create a **new project** in Supabase
  - Install and configure the **Supabase MCP** server:
    - Supabase Dashboard → Connect → MCP → VS Code → Add to VS Code → Install in Workspace
    - Authorize API access for Visual Studio Code

# Exercise: Polls App with MCP (2)



- Step 2: **Create the app**, using Supabase through MCP
  - Create a **Polls app**: view polls, vote, view results
  - Use simple **HTML + CSS + JavaScript** code
  - Use **Supabase** as backend (use the Supabase MCP)
  - Each **poll** has a **question + answers** with **votes** for each answer
  - Initially, create a few **sample polls** with answers in the DB
  - **Voting is anonymous**: any visitor can view polls and vote

# Exercise: Polls App with MCP (3)

Add Context...

Create a Polls app: view polls, vote, view results

Use simple HTML + CSS + JavaScript code

Use Supabase as backend (use the Supabase MCP)

Each poll has a question + answers with votes for each answer

Initially, create a few sample polls with answers in the DB

Voting is anonymous: any visitor can view polls and vote

Agent ▾ Auto ▾   

Let me start by setting up the Supabase database:

 Run Apply migration – supabase (MCP Server) 

Applies a migration to the database. Use this when executing DDL operations. Do not hardcode [Show More](#)

Input

```
{ "name": "create_polls_schema", "query":
```

See more

Note that MCP servers or malicious users may attempt to misuse 'Code' through tools.

Good! Now let me insert sample poll data:

 Run Execute SQL – supabase (MCP Server) 

Executes raw SQL in the Postgres database. Use `apply_migration` instead for DDL operations [Show More](#)

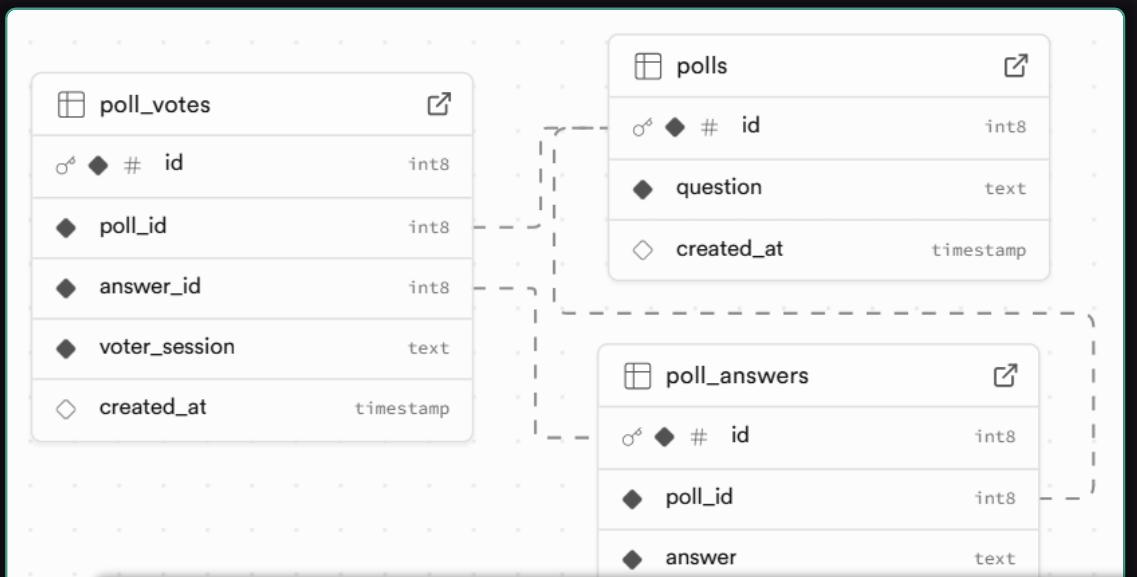
Input

```
{ "query": "\n-- Insert sample polls\ni
```

Note that MCP servers or malicious users may attempt to misuse 'Code' through tools.

# Exercise: Polls App with MCP (4)



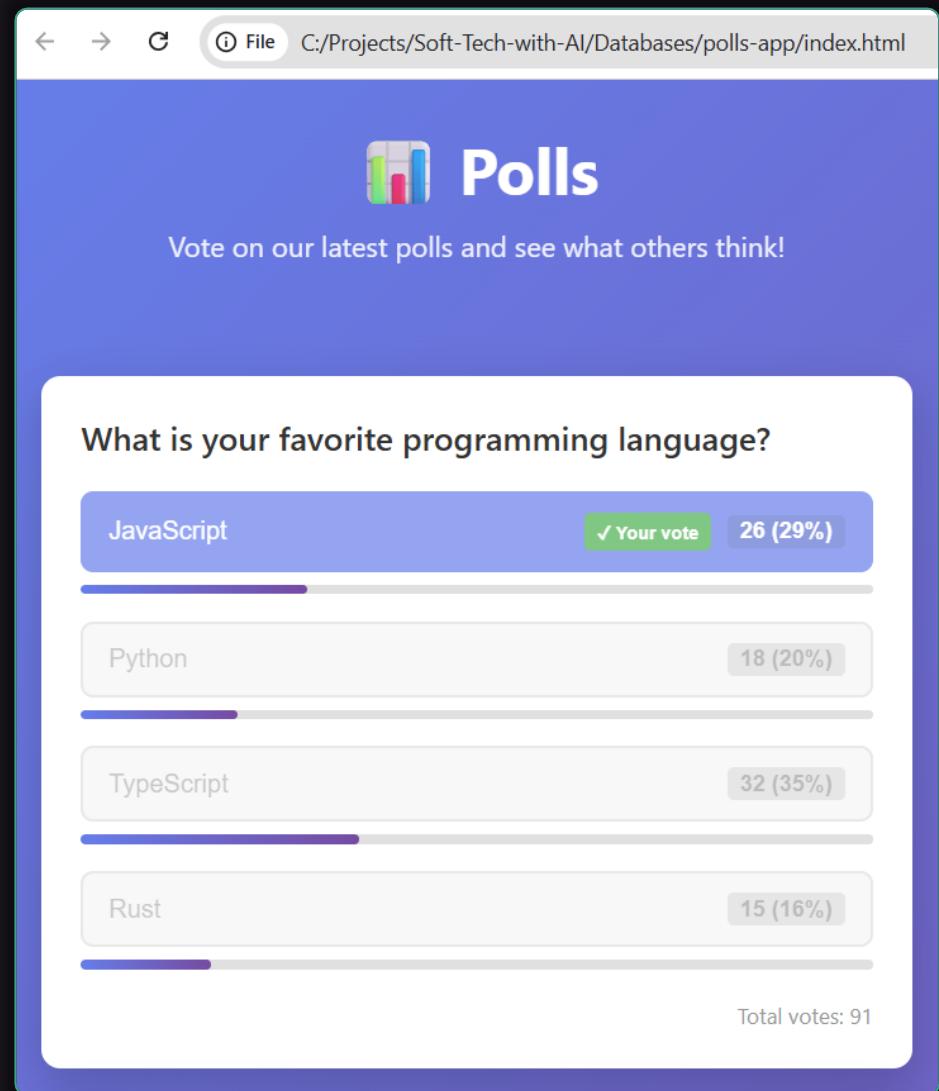
The screenshot shows a database schema diagram with three tables:

- poll\_votes**:
  - id (int8)
  - poll\_id (int8)
  - answer\_id (int8)
  - voter\_session (text)
  - created\_at (timestamp)
- polls**:
  - id (int8)
  - question (text)
  - created\_at (timestamp)
- poll\_answers**:
  - id (int8)
  - poll\_id (int8)
  - answer (text)

Below the schema, a table view displays the data for the **polls** table:

	id int8	question text	created_at timestamp
1	1	What is your favorite programming language?	2026-01-29 07:57:18.882831
2	2	Which frontend framework do you prefer?	2026-01-29 07:57:18.882831
3	3	Do you prefer light or dark mode?	2026-01-29 07:57:18.882831
4	4	What is your preferred work schedule?	2026-01-29 07:57:18.882831

Buttons at the bottom include **Insert**, **Filter**, **Sort**, and **RLS disabled**.



The screenshot shows a web browser displaying the **Polls** application at `C:/Projects/Soft-Tech-with-AI/Databases/polls-app/index.html`. The page has a purple header with the title "Polls" and a subtitle "Vote on our latest polls and see what others think!".

The main content is a poll titled "What is your favorite programming language?". It lists four options with their current vote counts and percentages:

Language	Your vote	Total votes
JavaScript	✓ Your vote	26 (29%)
Python		18 (20%)
TypeScript		32 (35%)
Rust		15 (16%)

A total of 91 votes are shown at the bottom right.

# Lesson Summary

- **Relational databases** (like PostgreSQL) hold **tables** (with rows and columns) and **relationships** (DB schema)
- **Supabase BaaS**: Projects, Database, Schemas, Tables, Table Editor, SQL Editor, Users and Auth, Storage, Edge Functions
- **SQL** language: table **structure** (CREATE / ALTER TABLE), **CRUD** operations (SELECT, INSERT, UPDATE, DELETE, Joins)
- **JS Apps with Supabase**: create SQL script → create tables in Supabase → API keys → JS code to interact with the DB
- **Supabase MCP**: connect VS Code with Supabase with MCP → create DB and JS app with an AI prompt

# Questions?



# Postbank – Exclusive Partner for SoftUni AI



- One of the leading **banking institutions** in Bulgaria
- Member of the Eurobank Group with € 103 billion assets
- Innovative trendsetter with next generation **beyond banking**, transforming today, empowering tomorrow
- Certified **Top Employer 2025** by the international Top Employers Institute
- Proven people care and **wellbeing initiatives**
- Benefits and unlimited access to professional, **personal and leadership trainings and programs**
- [www.postbank.bg](http://www.postbank.bg) / [careers.postbank.bg](http://careers.postbank.bg)



# Diamond Partners of Software University



# Diamond Partners of SoftUni Digital



**SUPER  
HOSTING  
.BG**



# Diamond Partners of SoftUni Digital



**HUMAN**

**NETPEAK**  
DIGITAL GROWTH PARTNER



Marmalab | Е-комерс агенция

**ETIEN YANEV**  
Break Your *Limits*. Live Your *Brand*.

**1FORFIT**



# Diamond Partners of SoftUni Creative



# Organization Partners of SoftUni Creative



THE  
BUCKS  
TOWN'S  
WORK

