

Guide to Setting Up and Using Supabase

Introduction

Supabase is an open-source Backend-as-a-Service (BaaS) platform that provides features like real-time databases, authentication, and serverless functions. It simplifies backend development while offering robust integrations for modern applications.

Prerequisites

1. Basic knowledge of programming.
 2. A web browser.
 3. An account with [Supabase](#).
 4. A code editor like VS Code.
-

Setting Up Supabase

Step 1: Create a Supabase Account

1. Visit the [Supabase website](#) and click on **Sign Up**.
2. Use your email or GitHub account to create an account.

Step 2: Create a New Project

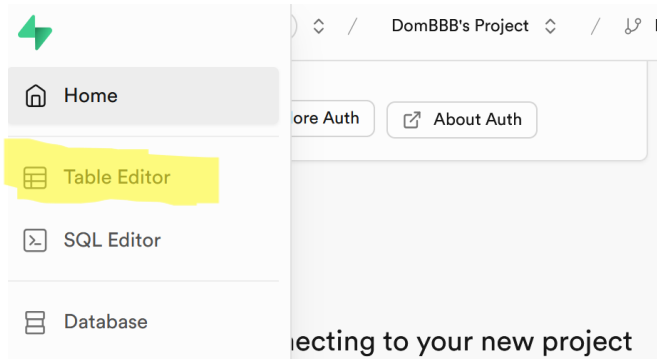
1. After logging in, go to the **Dashboard**: <https://supabase.com/dashboard/projects>
2. Click **New Project**.
3. Setup your **Organization** (just leave the default values and click **Create organization**).
4. Fill in the project details:
 - **Project Name**: A unique name for your project.
 - **Database Password**: Choose a strong password (**yourself** or click **Generate a password** and copy it to note it down).
 - **Region**: Select **Central Europe (Zurich)** or **Central EU (Frankfurt)**.
5. Click **Create New Project**.

Step 3: Wait for the Setup to finish.

Using Supabase

Step 4: Database

1. Navigate to the **Table Editor** tab.



2. Click **Create a new table**.
3. Fill in the **name** and the **columns** (specify for each column the type of data you want to store).
4. Click **Save**.
5. Example: To create a "users" table:

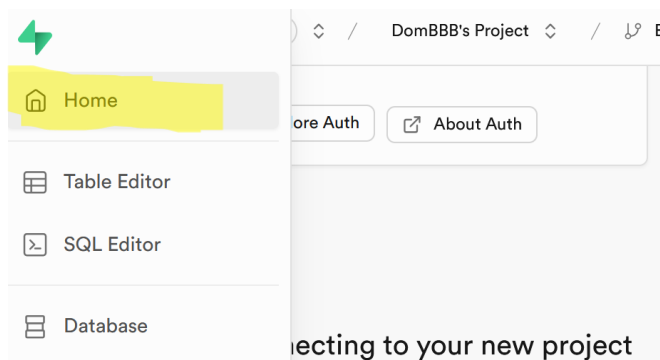
A screenshot of the 'Create a new table' form in Supabase. The form is for creating a table named 'users' under the 'public' schema. It has fields for 'Name' (users) and 'Description' (Optional). Below is a 'Columns' section with a table:

Name	Type	Default Value	Primary	Is Identity	Is Unique
id	# int8	NULL	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
username	T varchar	NULL	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
password	T varchar	NULL	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Two callout boxes are present: one for 'Is Identity' (Automatically assign a sequential unique number to the column) and one for 'Is Unique' (Enforce if values in the column should be unique across rows). Red arrows point from the 'Is Identity' and 'Is Unique' checkboxes in the table to their respective callout boxes.

Step 5: Access from Python

1. Navigate to the **Home** tab.



2. Scroll down to find your **Project URL** and **API Key** (click the yellow marked text to find the **SERVICE_ROLE SECRET** key and use this one)

Connecting to your new project

Interact with your database through the [Supabase client libraries](#) with your API keys.

More information about your project's keys can be found in your project's API settings.

[View API settings](#) [About APIs](#)

Project API

Your API is secured behind an API gateway which requires an API Key for every request. You can use the parameters below to use Supabase client libraries.

Project URL [Copy](#)

A RESTful endpoint for querying and managing your database.

API Key [Copy](#)

[anon](#) [public](#)

This key is safe to use in a browser if you have enabled Row Level Security (RLS) for your tables and configured policies. You may also use the service key which can be found [here](#) to bypass RLS.

Project API Keys

[service_role](#) [secret](#)

**** * **** * **** * **** * [Reveal](#)

This key has the ability to bypass Row Level Security. Never share it publicly. If leaked, generate a new JWT secret immediately.

3. Install the library using pip in your cmd/terminal: **pip install supabase**
4. Initializing Supabase in Python:

```
from supabase import create_client, Client

# Replace these with your Supabase project details
SUPABASE_URL = "https://<your-project-ref>.supabase.co"
SUPABASE_KEY = "<your-supabase-key>"

supabase: Client = create_client(SUPABASE_URL, SUPABASE_KEY)
```

5. Write data to Supabase:

```
def insert_user(username: str, password: str):
    try:
        response = supabase.table("users").insert({"username": username,
                                                    "password": password}).execute()
        return response.data
    except Exception as e:
        return e

insert_user("john_doe_5", "secure_password123")
```

6. Read data from Supabase:

Getting all users

```
def get_all_users():
    try:
        response = supabase.table("users").select("*").execute()
        return response.data
```

```
except Exception as e:  
    return e
```

```
get_all_users()
```

Getting specific user

```
def get_user_by_username(username: str):  
    try:  
        response = supabase.table("users").select("*").eq("username",  
            username).execute()  
        return response.data  
    except Exception as e:  
        return e
```

```
get_user_by_username("john_doe")
```

7. Update data to Supabase:

```
def update_password(username: str, new_password: str):  
    try:  
        response = supabase.table("users").update({"password":  
            new_password}).eq("username", username).execute()  
        return response.data  
    except Exception as e:  
        return e
```

```
update_password("john_doe", "new_secure_password456")
```

8. Delete data from Supabase:

```
def delete_user(username: str):  
    try:  
        response = supabase.table("users").delete().eq("username",  
            username).execute()  
        return response.data  
    except Exception as e:  
        return e
```

```
delete_user("john_doe")
```

Step 6: View your Data

You can always get a live view of your table data when going back to the table editor (Step 4) and select the table you are interested in.