Introduction to Supabase & PostgreSQL

Objective: To understand what Supabase is and why we use PostgreSQL for modern applications.

What is Supabase?

Think of Supabase as an "open-source alternative to Firebase." It provides a backend (database, authentication, real-time subscriptions) for your apps without you needing to manage a server.

The Engine: It is built on top of PostgreSQL, one of the world's most powerful relational database systems.

The Benefit: You get the power of SQL (structured query language) with an easy-to-use dashboard that looks like a spreadsheet.

Relational vs. Spreadsheet

Unlike Excel, where you can type anything anywhere, a Relational Database (like Postgres) requires structure.

Tables: Data is stored in strict tables (e.g., "Users", "Products").

Columns: Each column has a specific type (Text, Number, Date) that must be respected.

Step 1 – Creating Your Project

Signing Up and Setup

The first practical step is initializing the cloud environment.

- Go to Supabase.com: Sign in using GitHub (common for developers).
- "New Project": Click the green button to start.

Project Details:

- Name: Give it a clear name (e.g., My-IoT-Class-DB).
- Database Password: Crucial Step! You must create a strong password and save it immediately. Supabase will not show it to you again.
- Region: Select a server location closest to you (e.g., Singapore or Mumbai) for faster speed.

Step 2 – The Table Editor

Navigating the Dashboard

Once the project builds (takes ~2 minutes), you will see the dashboard. We will focus on the Table Editor.

- The Interface: On the left sidebar, look for the "Table" icon. This is your visual database manager.
- Creating a Table: Click "Create a New Table."
- Name: Use lowercase and plural names (standard convention), e.g., sensors or students.
- Primary Key: Supabase automatically adds an id column. Do not remove this. It is the unique fingerprint for every row.

Step 3 – Defining Columns (Data Types)

Designing Your Schema

A database needs to know what kind of data to expect. This is called the "Schema."

Add Columns:

- Name: temperature -> Type: float8 (Decimal number).
- Name: location -> Type: text (String of characters).
- Name: is_active -> Type: bool (True/False).
- Name: created_at -> Type: timestamptz (Time data).

Default Values: You can set created_at to now() so it automatically records the time data is entered.

Step 4 – Inserting Data (GUI vs. SQL)

Adding Data Manually (The Easy Way)

- In the Table Editor, click "Insert Row."
- A form appears. Type "Living Room" for location and "25.5" for temperature.
- Click Save. You have just created a database record!

Adding Data via SQL (The Code Way)

- Click the SQL Editor icon on the left.
- Type this command: INSERT INTO sensors (location, temperature) VALUES ('Kitchen', 28.0);
- Click Run. This is how your code will talk to the database later.

Connecting to Your App

API Keys and URLs

To connect your IoT device or website to this database, you need two things found in Settings > API:

- Project URL: The web address of your specific database.
- API Keys:
  - anon (Public): Safe to use in your frontend/website code. It follows security rules (RLS).
  - service_role (Secret): Bypasses all security. NEVER share this or put it in public code. It creates a "god mode" connection.