Data Management for Hackathons

Learn SQL, build a relational database, and deploy a live backend with Supabase

Why Bother with a Database?

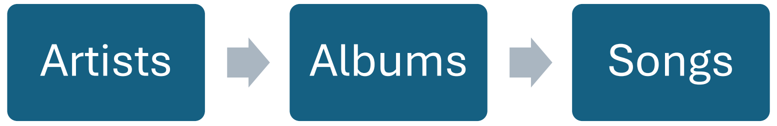
- What is it?
  - Your app's long-term memory.
- Why is it critical for a hackathon?
  - User Accounts: Storing profiles & login info.
  - Saving Progress: User-generated content (posts, scores, playlists).
  - Configuration: Saving settings & application state.
- *Without a database, every refresh wipes everything*. *It's the difference between a demo and an application*.

The Two Flavors: SQL vs. NoSQL

- SQL (Relational)
  - Analogy: A set of structured, linked Excel spreadsheets.
  - Structure: Rigid tables with predefined columns and rows.
  - Examples: PostgreSQL, SQLite.
  - Best for: Data with clear relationships (e.g., users, posts, comments).
- NoSQL (Non-Relational)
  - Analogy: A folder of flexible JSON files.
  - Structure: No rigid schema; data is often nested.
  - Examples: MongoDB, Firebase Firestore.
  - Best for: Unstructured data or data that changes a lot.
- Today's Focus: SQL. It's a foundational skill, and modern tools make it incredibly fast to set up.

The Relational Advantage: Why Use Multiple Tables?

- The Problem: Storing everything in one big table.
  - 'Queen', 'Bohemian Rhapsody', 'A Night at the Opera'
  - 'Queen', 'Another One Bites The Dust', 'The Game'
  - What if you misspell 'Queen'? You now have a "new" artist.
- The Solution: Normalization
  - Store related information in separate, linked tables.
  - `artists` Table → `albums` Table (links to artist ID) → `songs` Table (links to album ID)
- Key Benefits:
  - Reduces Redundancy: Store "Queen" only once.
  - Ensures Data Integrity: Impossible to misspell an artist on a new song.
  - Efficiency & Speed: Smaller, faster, and more professional.



The 4 Essential SQL Commands (CRUD)

- **Create**
  - INSERT INTO songs (title, album_id) VALUES ('New Song', 4);
- **Read**
  - SELECT * FROM songs;
- **Update**
  - UPDATE songs SET title = 'Corrected Title' WHERE id = 1;
- **Delete**
  - DELETE FROM songs WHERE id = 1;

The Superpower of SQL: `JOIN`

- Goal: How do we get data from all our linked tables at once?
- The JOIN command merges tables based on their linked IDs.

```
-- Get all songs by 'Queen' with their album title
SELECT
    songs.title AS song_title,
    albums.title AS album_title,
    artists.name AS artist_name
FROM songs
JOIN albums ON songs.album_id = albums.id
JOIN artists ON albums.artist_id = artists.id
WHERE artists.name = 'Queen';
```

Hackathon Superpower: Supabase

- What is it? A complete backend in minutes.
  - A powerful PostgreSQL Database (that's a type of SQL database).
  - User Authentication (Login, Sign up).
  - An instant API to access your data.
- Why is this a superpower?
  - It does all the boring, time-consuming backend setup for you, so you can focus on building your actual app.

Live Demo: Connecting an App to Supabase

- Create a Project on supabase.com.
- Import Data: Use the Table Editor to import the `artists.csv`, `albums.csv`, and `songs.csv` from the provided code.
- Get API Keys: Go to Project Settings -> API. We need the `Project URL` and the `anon public key`.
- Let's write the code!

Live Demo: The Code

Recap & Resources

- What We Covered:
  - Relational Model: The professional way to structure data.
  - SQL JOIN: The command to combine data from linked tables.
  - Supabase: Your instant backend (Database + Auth + APIs).
  - Supabase JS: The library to easily and securely query your database from a web app.
- Resources:
  - Database Generator Script: [Link to your GitHub repo or provided files]
  - Supabase JavaScript Docs: supabase.com/docs/reference/javascript/start
  - SQLZoo JOIN Tutorial: sqlzoo.net/wiki/The_JOIN_operation