Semi-Finals Output Part 1: SQL - DDL and DML Tasks

Create and execute an SQL script that performs the following:

- 1. Create a new MySQL database.
- 2. Select the database for use.
- 3. Create at least two tables with appropriate fields, data types, and primary key.
- 4. Insert a minimum of 10 records into each table.
- 5. Update at least **one existing record** in a table.
- 6. Delete at least **one record** from a table.
- 7. Write at least five SELECT statements that retrieve specific records based on different criteria (e.g., filtering, ordering, etc.).

The SQL code serves as the **foundation of the Playlist Manager system**, providing a structured and relational database that supports core functionalities such as user authentication, playlist creation, song cataloging, and many-to-many relationships between playlists and songs. It ensures data integrity through foreign key constraints and supports efficient data retrieval and manipulation.

Core Features and Design:

	User Management		Song Management		Playlist System	Many	r-to-Many Relationship (Playlist ↔ Songs)
•	The Users table stores essential account information (ID, username, password, email). Supports operations like registration, login, updates, and account deletion.	•	The Songs table contains the library of all available songs, including their title, artist, and genre. Enables adding, updating, and deleting songs.	•	The Playlist_Details table records all playlists created by users. Each playlist is linked to a specific user via a foreign key (user_id), with a	•	The Playlist_Songs junction table ensures playlists to contain multiple songs, and songs to belong to multiple playlists.
					timestamp to track creation.	•	Enforced through foreign keys referencing both Playlist_Details and Songs.

Relational Design Summary:

- 1 User → Many Playlists
- 1 Playlist → Many Songs
- 1 Song → Can exist in Many Playlists
- Cascading deletes ensure deleted records are automatically removed (e.g., deleting a user removes their playlists).

CRUD Operations and Queries:

The SQL code includes examples of:

- Creating: Tables, users, songs, playlists, and playlist-song relationships.
- Reading: Fetching songs by genre, playlist by user, or filtering emails.
- Updating: Modifying user emails, song genres, and playlist names.
- Deleting; Removing users, songs, or specific playlist-song entries.

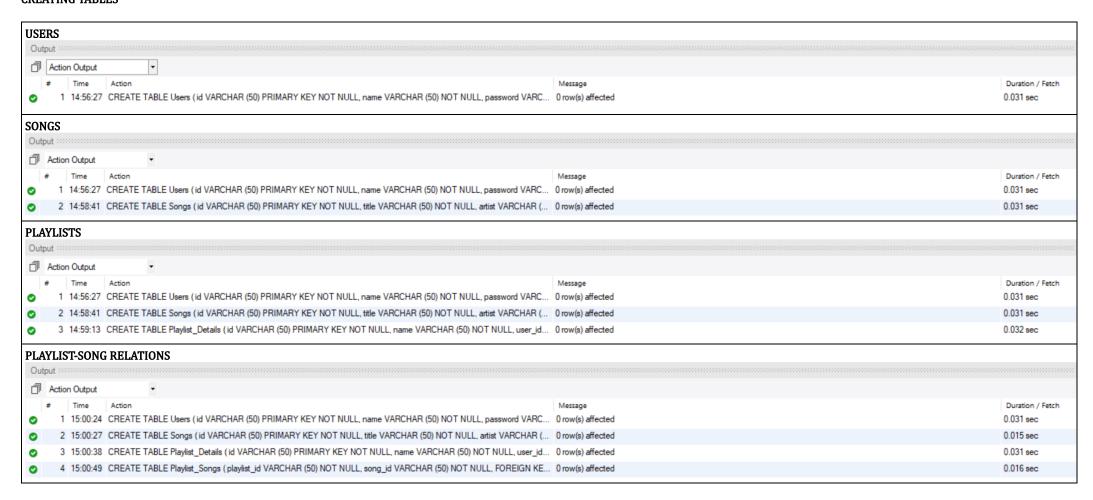
Uses primary keys for unique identification and foreign

Highlights:

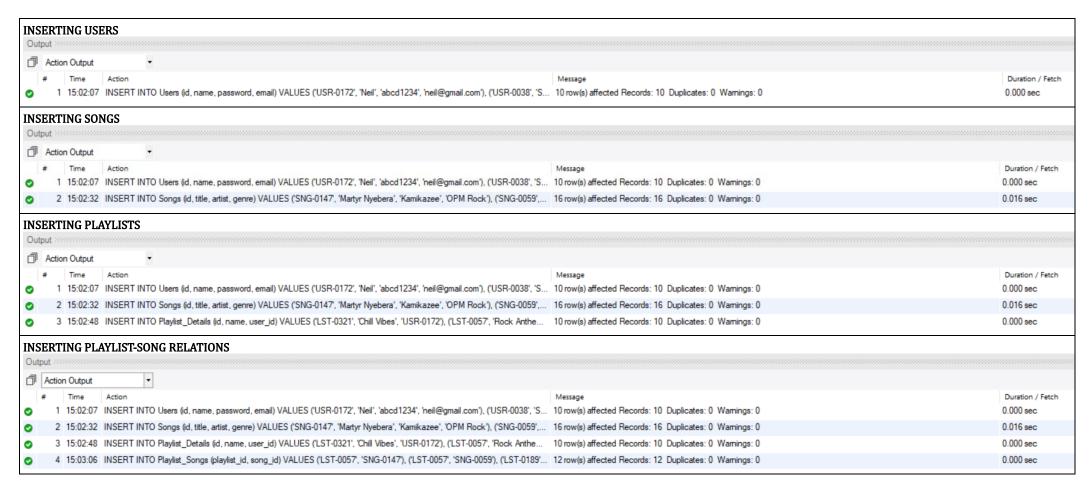
- keys for relational integrity.
 Implements ON CASCADE DELETE to simplify data
- Provides sample data for immediate testing and interaction.
- Includes query examples that demonstrate how to extract meaningful information (e.g., all songs in a playlist, all playlist of a user).

OUTPUT SCREENSHOTS:

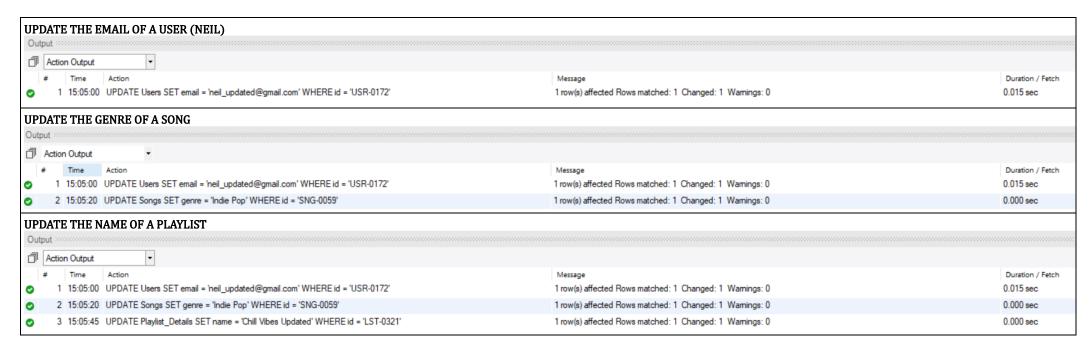
CREATING TABLES



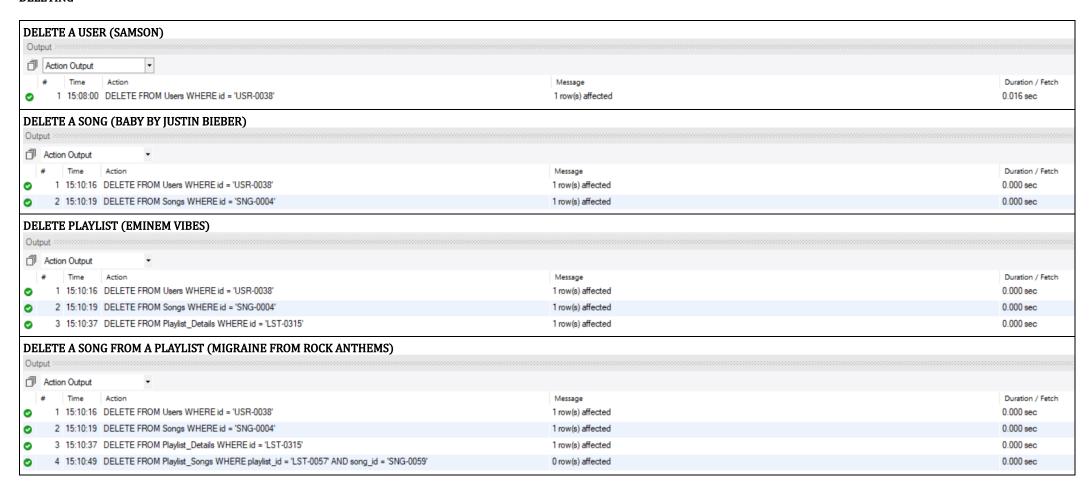
INSERTING



UPDATING



DELETING



SELECTING

