BIG DATA MANAGEMENT
Assignment - I

Date : 23/06/2025

Submitted To:

Dr. Dip Shankar Banerjee
Associate Professor
Department of Artificial Intelligence and Data Engineering
Indian Institute of Technology – Jodhpur

Submitted By:

Name: Rodda Sree Charan Reddy
Roll Number: G24AI1095
Email id: g24ai1095@iitj.ac.in
Course: PGD-DE & 3rd Trimester
IIT-Jodhpur

Music Recommendation System using MySQL and Python

Overview

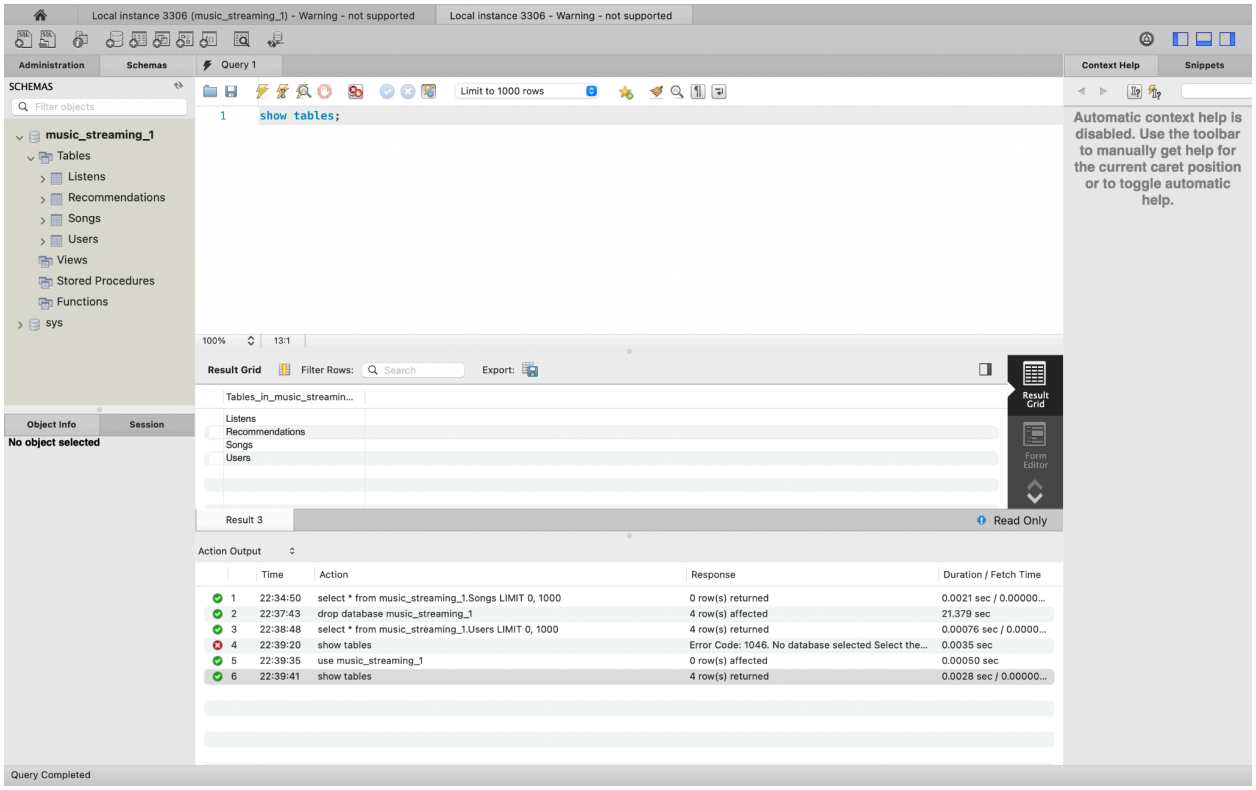
This project demonstrates the development of a basic music streaming and recommendation system using Python and MySQL. It involves creating a relational database schema, populating it with user, song, and listening data, and executing SQL queries to analyze listening behavior.

The primary goal is to generate personalized music recommendations based on user activity — leveraging techniques such as collaborative filtering and timestamp-based analysis. This system mimics a simplified backend for music analytics and personalized content delivery.

1. Database and Table Setup

The application starts by connecting to a MySQL server and creating a database named `music_streaming_1`. It sets up four key tables:

- **Users**: Contains user profile details.
- **Songs**: Stores metadata for each song, including title, artist, and genre.
- **Listens**: Logs listening events with ratings and timestamps.
- **Recommendations**: Designed to hold dynamically generated personalized song suggestions.

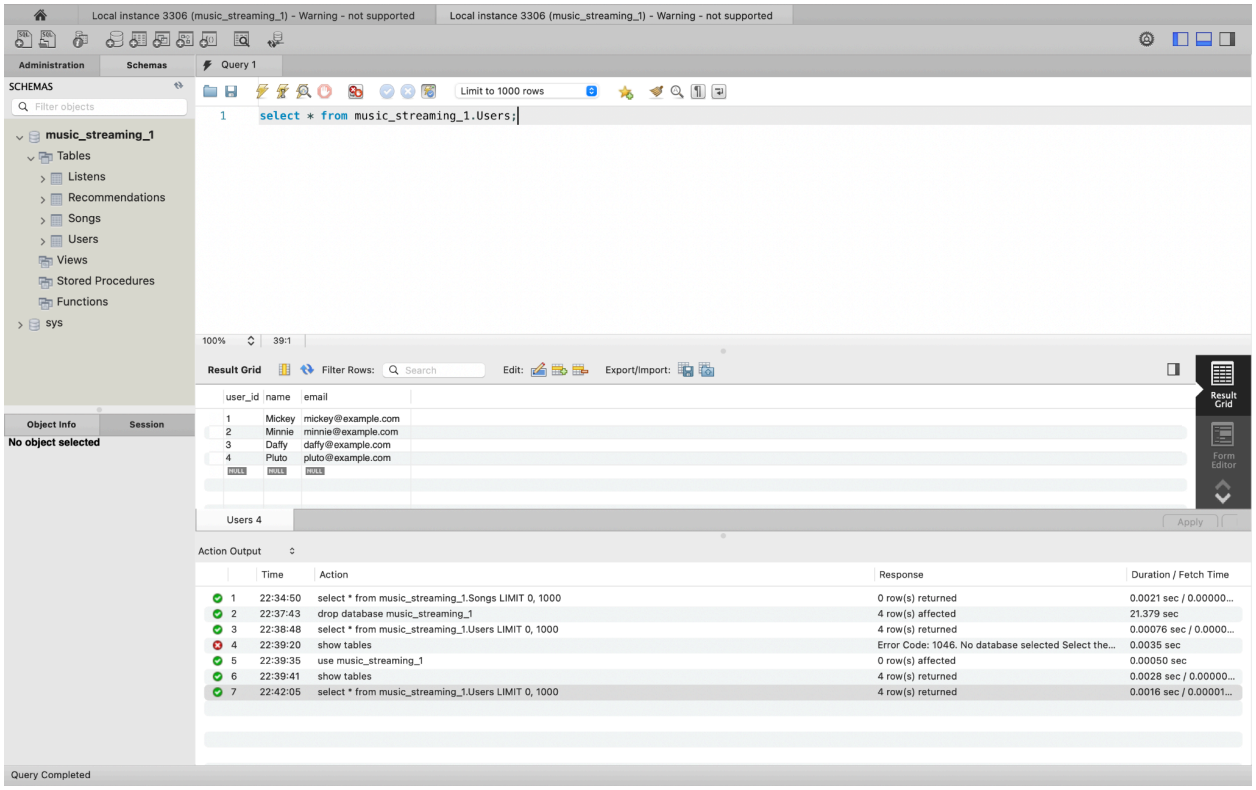


2. Inserting Sample Data

- User entries for four users are added.
- Songs by Taylor Swift, Ed Sheeran, and the Beatles are inserted into the Songs table.
- Listening records with sample ratings and timestamps are entered into the Listens table.

3. Displaying Table Data

The data in each table is displayed using the SELECT * command.



4. Filtering and Analytical Queries

A range of SQL queries were executed to explore and analyze the dataset:

- Filtering songs by genre, such as all Classical songs and those starting with "Ye".
- Retrieving all available genres and identifying distinct genre values.
- Counting the number of songs by each artist within each genre using aggregation and joins.
- Fetching songs with high user ratings (ratings above 4.6).
- Calculating the average rating per song.
- Listing all songs performed by Taylor Swift and Ed Sheeran, as well as filtering songs by Pop or Rock genres.

- Identifying songs that have no recorded listening timestamps (i.e., unplayed songs).

5. Collaborative Filtering: Recommendation Logic

A basic collaborative filtering approach was implemented to suggest songs:

- Songs commonly listened to by multiple users were identified to uncover shared preferences.
- Based on this, recommendations were generated for users by suggesting songs they haven't listened to but which were enjoyed by users with similar listening histories.
- These generated recommendations were inserted into the Recommendations table for persistence.

6. Personalized Recommendations

- Q2:

Recommendations were generated specifically for Minnie (user_id = 2) using collaborative filtering. Songs that Minnie hasn't listened to but are liked by similar users were suggested.

- Q3:

A refined recommendation method was applied by considering only songs that were actually listened to (i.e., listen_time IS NOT NULL), ensuring that only recent and relevant preferences were used.

- Q4:

New recommendations were generated for Minnie based explicitly on listen times, assuming that recently played songs represent more current tastes.

- Using the query logic from step #3, Minnie's listening history was reviewed:

- From the Listens table, it was found that Minnie has listened to the following songs:

Song_id	title	Artist
1	Evermore	Taylor Swift
6	Yesterday	Beatles

Analysis Based on Listen

Using actual `rating` values, Minnie's listening activity includes classic Beatles songs, indicating a preference for that genre.

Let's examine the recent listening activity of other users:

- **Mickey** (user_id = 1) has recently listened to:
    - *Evermore* (Taylor Swift)
    - *Yesterday* (Beatles)
- **Daffy** (user_id = 3) has recently listened to:
    - *Willow* (Taylor Swift)
    - *Shape of You* (Ed Sheeran)

From this overlap, we observe:

- Daffy shares a taste for contemporary pop, having listened to *Willow* and *Shape of You*.

- Mickey overlaps with Minnie on the Beatles, having recently listened to *Yesterday*.

Based on these listening patterns, we recommend *Evermore* and *Yesterday* as personalized suggestions that align with Minnie's inferred musical interests.

Static vs. Listen-Time-Based Recommendation Methods

- **Basis**:
    - *Static Method (#2)*: Relies on overall average ratings across all users.
    - *Listen-Time-Based Method (#4)*: Focuses on recent listening activity, considering only entries with valid `listen_time`.
- **Personalization**:
    - *Static Method*: Provides generic recommendations not tailored to individual users.
    - *Listen-Time-Based Method*: Offers personalized suggestions based on the user's most recent preferences.
- **Accuracy**:
    - *Static Method*: May include songs that were never actually played or are outdated.
    - *Listen-Time-Based Method*: More accurate, as it reflects real and recent listening behavior.
- **Data Reliability**:

    - *Static Method*: Can be influenced by NULL values or old data.
    - *Listen-Time-Based Method*: Filters out incomplete entries and includes only timestamped listens, ensuring cleaner data.
- **Recommendation Output**:
    - *Static Method*: Suggests high-rated songs the user hasn't listened to yet, based on global popularity.
    - *Listen-Time-Based Method*: Recommends songs favored by similar users based on current listening patterns.

Conclusion

This Python–MySQL integration project effectively replicates the core functionalities of a basic music recommendation system. By leveraging structured data storage alongside SQL-based analytics and filtering logic, it illustrates how music streaming platforms can deliver personalized content to users.

The step-by-step recommendation queries serve as a strong foundation for advancing toward more sophisticated solutions, such as user profiling, genre-based predictions, and real-time recommendation engines powered by machine learning.

Git Hub Link - https://github.com/Sreecharan162/Big-Data.git