BIG DATA MANAGEMENT

Assignment - I

6/18/2025

Submitted To

Dr. Dip Shankar Banerjee

Associate Professor

Department of Artificial Intelligence and Data Engineering

Indian Institute of Technology – Jodhpur

Submitted By

Name: Purushothaman S

Roll Number: G24AI1042

Course: PGD-DE & 3rd Trimester

Email id: g24ai1042@iitj.ac.in

Subject Code: AIL7520

Assignment Number: 1

IIT-Jodhpur

Index

Music Recommendation System using MySQL and Python

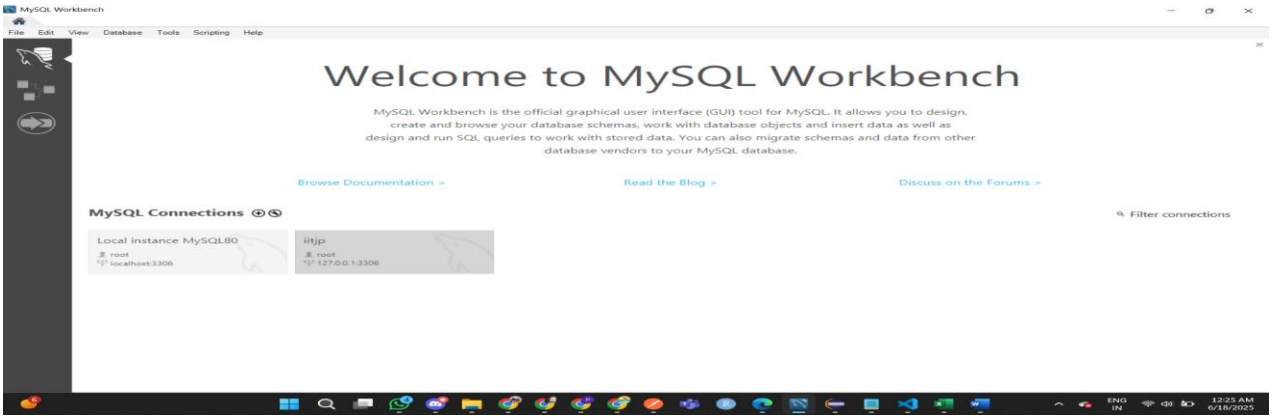
Introduction

This project implements a rudimentary music streaming and recommendation system using Python and MySQL. The system supports creating necessary database tables, inserting user, song, and listening data, and performing basic SQL queries to analyse listening behaviour. The core objective is to generate personalized song recommendations based on listening patterns, including collaborative filtering based on shared listening histories and listening timestamps. It simulates a lightweight backend for music data analytics and personalized content delivery.

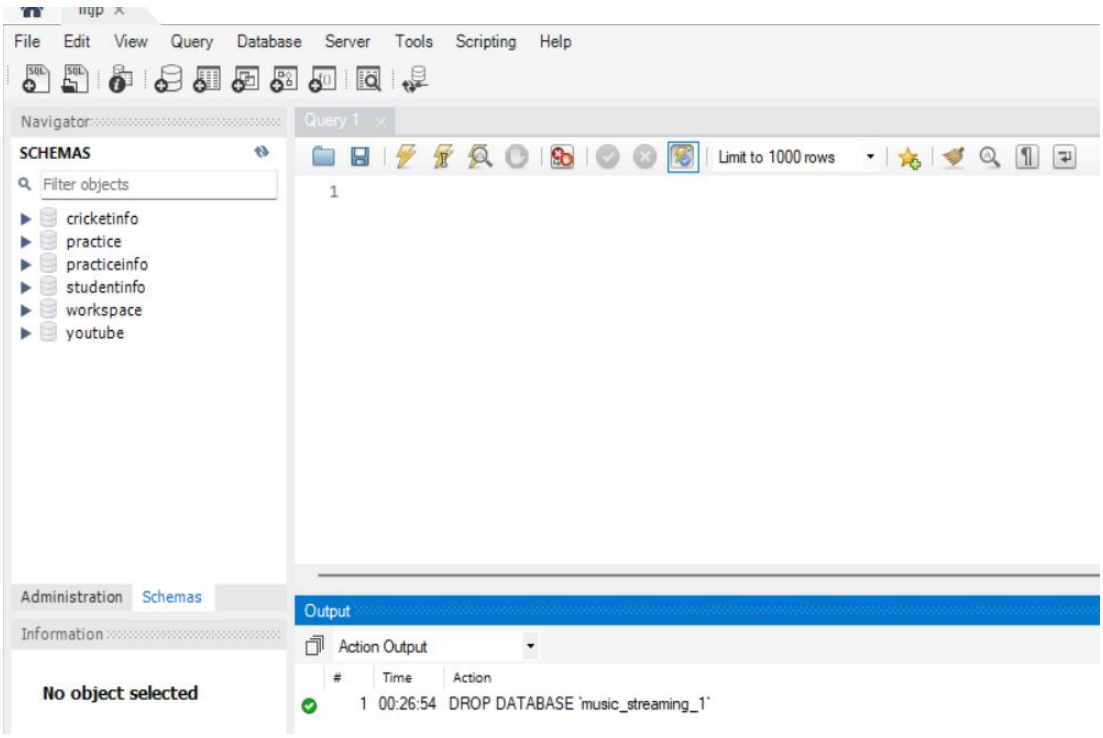
1. Database and Table Creation

The script begins by connecting to a MySQL server and creating a database named music_streaming_1. It defines and creates four primary tables:

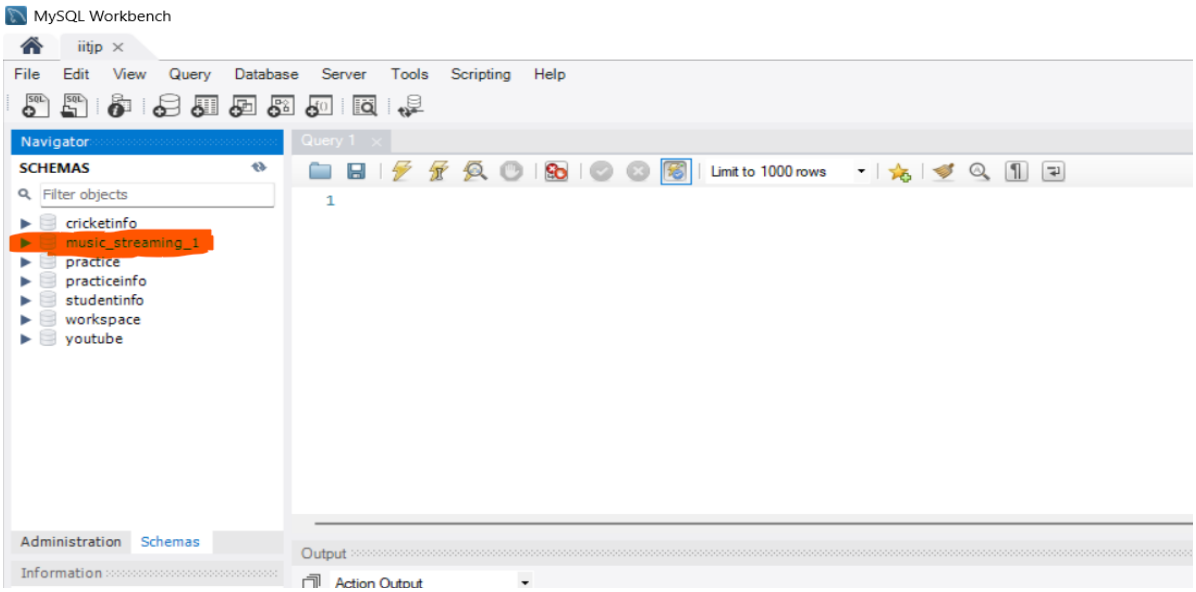
- Users: Stores user information.
- Songs: Stores song metadata including title, artist, and genre.
- Listens: Records listening events including ratings and timestamps.
- Recommendations: Will store the system-generated personalized song recommendations.



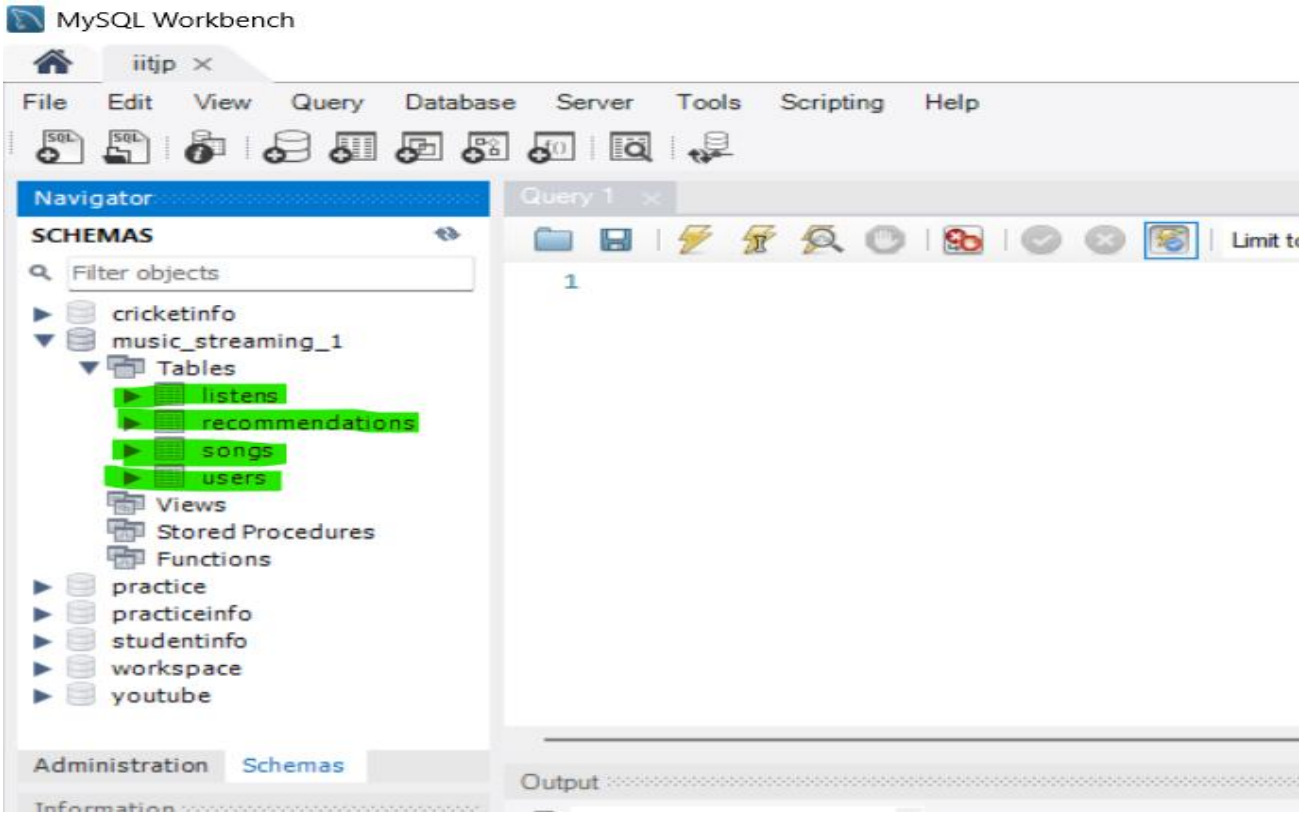
Before the Data Base Creation on the MySQL Server Manager



After the Data Base Creation on the MySQL Server Manager



The Tables Also has been Created



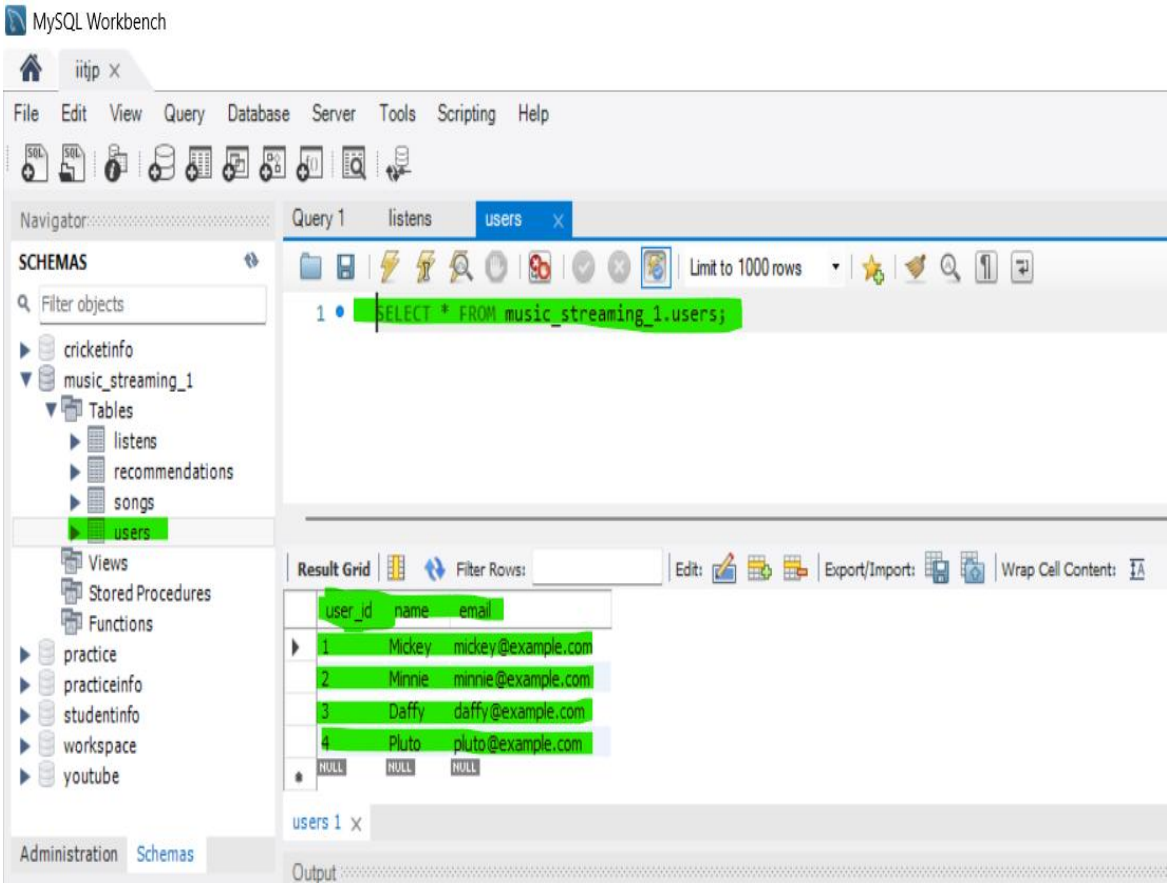
2. Inserting Sample Data

- User entries for four fictional 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

Each table is queried using a SELECT * command and the results are printed, ensuring data has been correctly inserted.

Here the Data has been Inserted into the table and displaying it



4. Filtering and Analytical Queries

Various SQL queries are executed to demonstrate analytics:

- Filtering classical songs and classical songs beginning with "Ye".
- Listing all genres and finding distinct genres.
- Counting songs per artist per genre and generating cross-table joins.
- Fetching high-rated songs (rating > 4.6).
- Calculating the average rating per song.

- Listing songs by Taylor Swift and Ed Sheeran, as well as Pop/Rock genres.
- Identifying songs without listening timestamps.

5. Collaborative Filtering: Song Recommendation Logic

- A Common Song Listening Pattern is used to identify song pairs shared among more than one user.
- Recommendations are generated for users based on songs they have not listened to but that others with similar tastes have.
- This logic is inserted into the Recommendations table.

6. Personalized Recommendations

- Q2: Recommendations for a specific user (Minnie) are generated based on the common listening history with other users.
- Q3: A more refined recommendation strategy is applied where only recent listens (i.e., those with a timestamp) are considered, assuming these reflect the most relevant preferences.
- Q4: Generate new recommendations for Minnie based on listen time
- Using the query from step #3, let's look at Minnie's listening history:
- From the Listens table, Minnie (user_id = 2) has recently listened to:

song_id	title	artist	listen_time
2	Willow	Taylor Swift	NULL (skipped)
7	Yellow Submarine	Beatles	2024-08-28 09:20:00
8	Hey Jude	Beatles	2024-08-27 16:45:00

So based on actual timestamps, Minnie's profile includes *Beatles classic songs*.

Let's compare that to others:

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

From the overlap, we find that:

- Daffy listened to "Willow" and "Shape of You"
- Mickey listened to "Yesterday" (shared artist)

Since Minnie hasn't listened to:

- Shape of You
- Bad Blood (from earlier static method)

We recommend these as fresh listens based on recent listening patterns.

Q5: What are the differences with the static method in #2 above?

Here's a comparison between the static method (step #2) and the listen-time-based method (step #4):

Aspect	Static Method (#2)	Listen-Time-Based Method (#4)
Basis of Recommendation	All listens (including null timestamps)	Only recent listens (non-null listen_time)
Recency Consideration	None	Yes — recent interactions are prioritized
Data used	Full ***Listens*** table	Filtered ***Listens*** with ***listen_time IS NOT NULL***
Behaviour Modeled	General user similarity across all time	Current/active user interests
Songs recommended to Minnie	***Shape of You, Bad Blood***	***Shape of You*** only
Filtering effect	Broader results	Stricter, fewer but more relevant recommendations
Timestamp sensitivity	Not sensitive to time	Sensitive to when songs were listened to

- The **static method** gives **more diverse recommendations**, including older or possibly outdated preferences.
- The **listen-time-based method** is **stricter**, reflecting **recent taste** — more precise but possibly fewer suggestions.

Conclusion

This Python-MySQL integration script successfully simulates the core functionalities of a music recommendation engine. By combining structured data storage with SQL analytics and filtering techniques, it demonstrates how music platforms can provide users with personalized content. The step-by-step recommendation queries lay the foundation for building more sophisticated machine learning models in the future, including user profiling, genre prediction, and real-time recommendation systems.

Note: Please Refer the code snippet on the git link below for the further Execution

Git Link : https://github.com/PurushothamanShanmugam/Big-Data