BIG DATA MANAGEMENT

Assignment - I

**Submitted To:**

Dr. Dip Shankar Banerjee

Associate Professor

Department of Artificial Intelligence and Data Engineering

Indian Institute of Technology – Jodhpur

**Submitted By:**

Name: Pranav Sei Vasthav Tenali Gnana

Roll Number: G24AI1114

Course: PGD-DE

Email id: g24ai1114@iitj.ac.in

Subject Code: AIL7520

Assignment Number: 1

IIT-Jodhpur

Music Data Analysis and Recommendation System

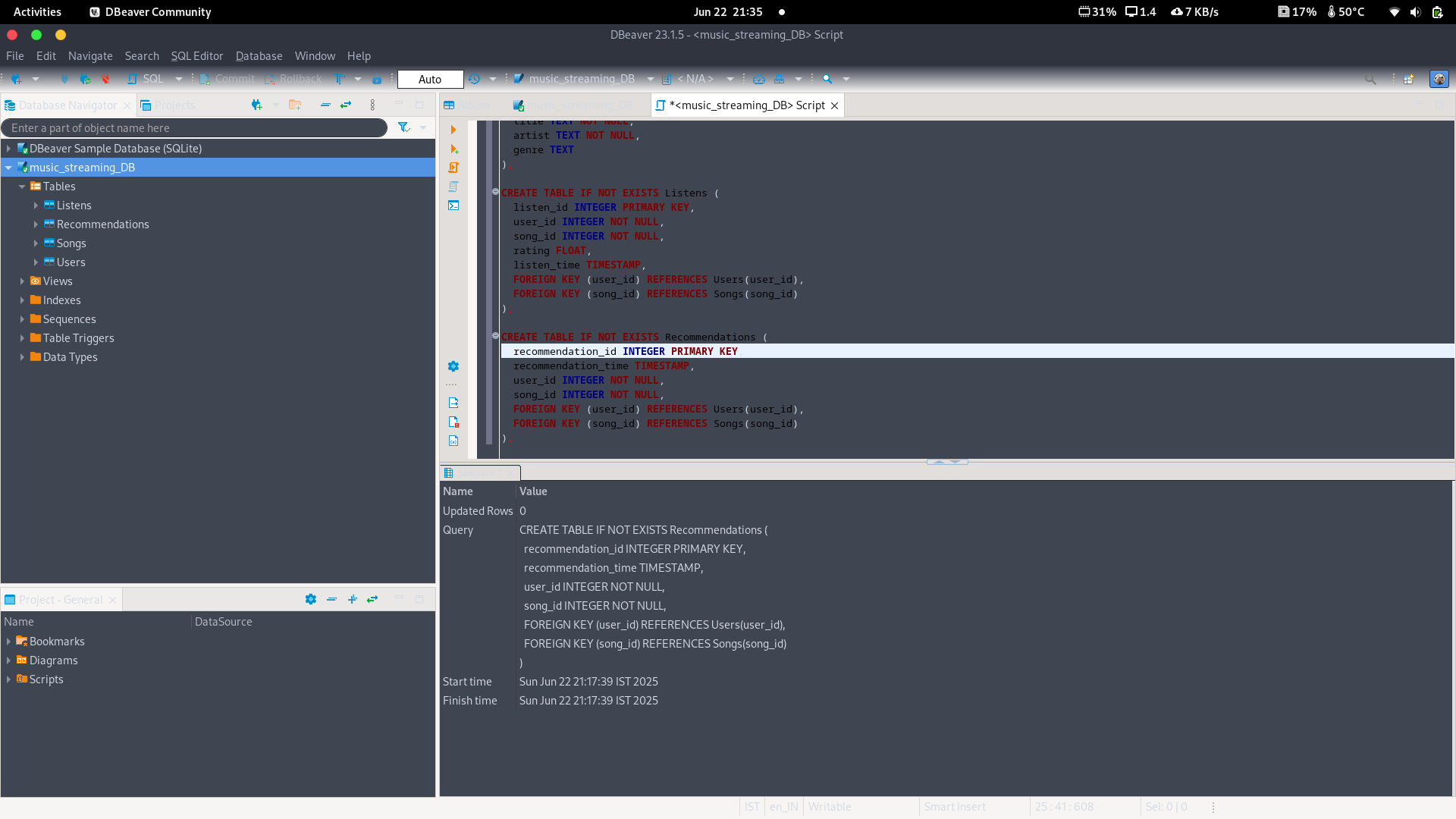
Platform: SQLite using Dbeaver

### ****1. Database and Table Creation****

The script connects to a MySQL server (configured using DBeaver on Parrot OS) and creates a database named music\_streaming\_DB. Within this database, it defines and creates the following four primary tables:

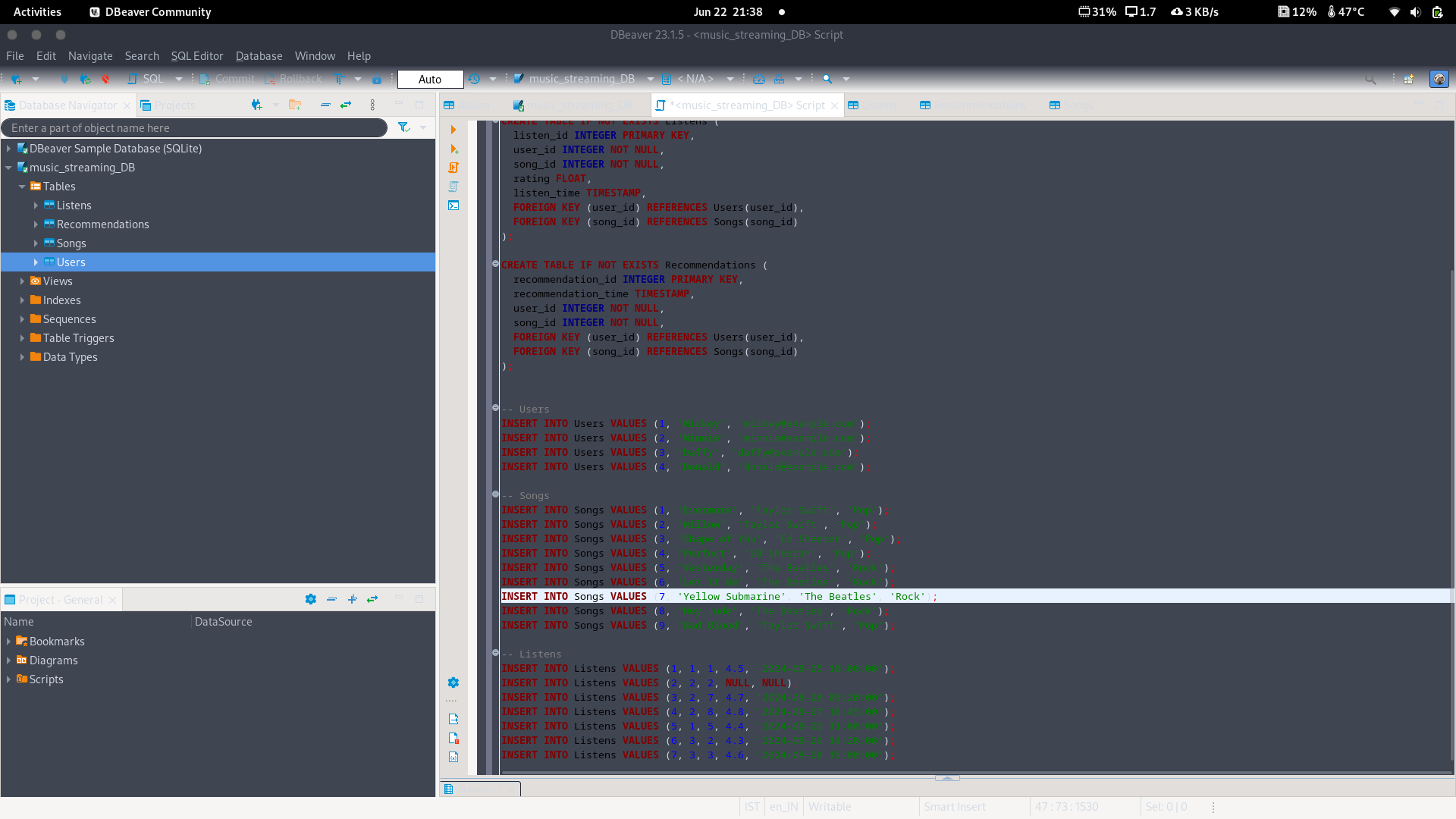
* **Users**: Stores user profile details including unique user ID, name, and email.
* **Songs**: Stores metadata about songs such as title, artist name, and genre (e.g., Pop, Rock).
* **Listens**: Logs each listening event including which user listened to which song, the rating given (if any), and the timestamp of listening.
* **Recommendations**: Contains personalized song recommendations generated by collaborative filtering, including the user, recommended song, and recommendation time.

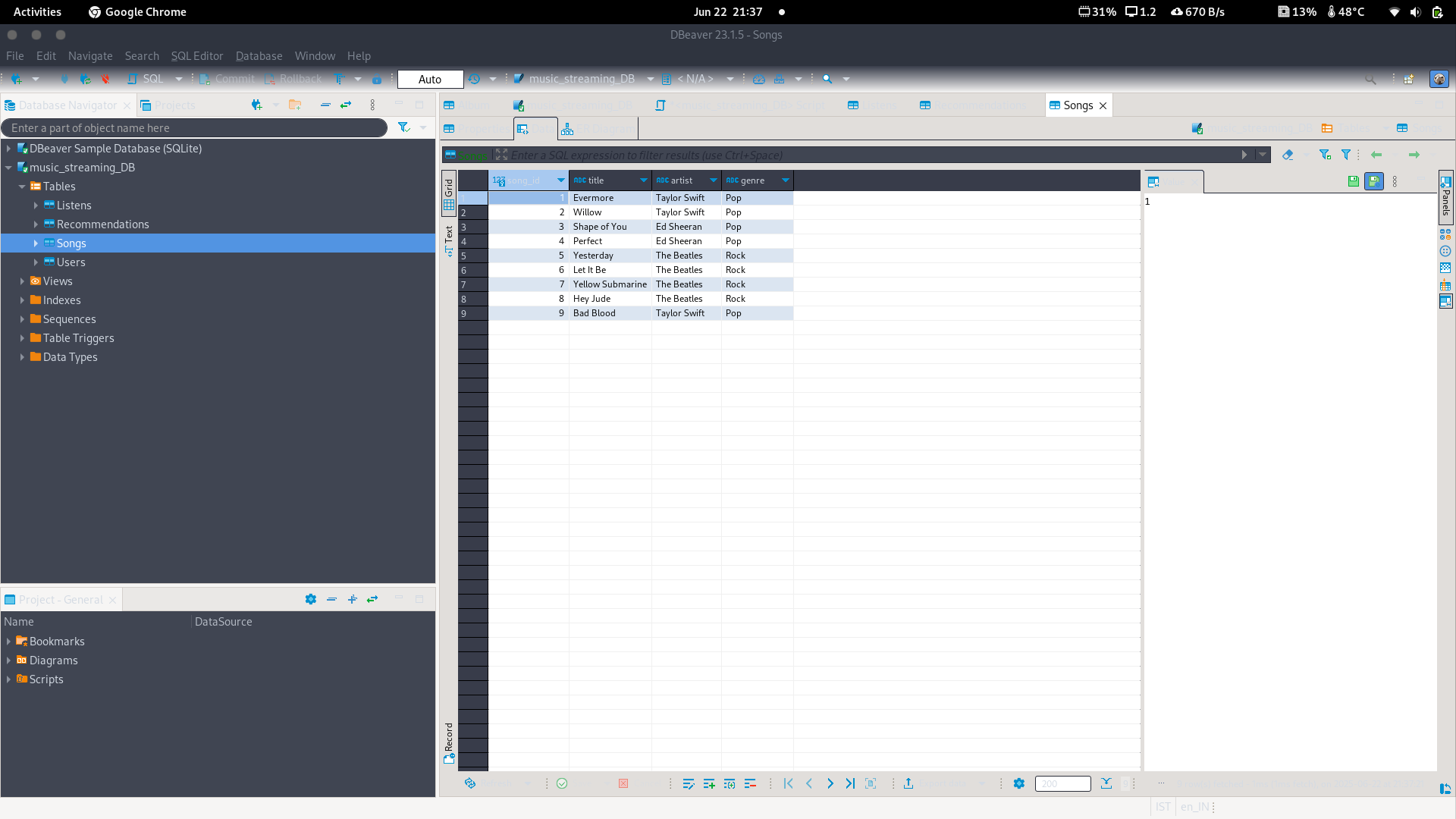
These tables form the foundation for tracking user behavior and implementing a basic recommendation system based on listening patterns and user-song interactions.



### ****2. Inserting Sample Data****

* Four fictional users — **Mickey**, **Minnie**, **Daffy**, and **Donald** — are inserted into the **Users** table with unique user IDs and email addresses.
* The **Songs** table is populated with tracks from popular artists including **Taylor Swift**, **Ed Sheeran**, and **The Beatles**. The songs are categorized into genres such as Pop and Rock.
* Sample listening records are inserted into the **Listens** table. These include:
  + The user who listened
  + The song they listened to
  + The rating they gave (if any)
  + The time of listening (where applicable)





### ****3. Filtering and Analytical Queries****

A series of SQL queries are executed on the database to perform meaningful analytics on the music data:

* **Filtering Rock genre songs** (referred to as classic in context) and Rock songs whose titles begin with "Ye" (e.g., Yellow Submarine).
* **Listing all available genres** from the Songs table, as well as **extracting distinct genres**.
* **Counting songs by each artist across different genres**, and performing a **cross-table join** between Users, Songs, and Listens to combine user activity with song metadata.
* **Identifying high-rated songs**, i.e., songs with a rating greater than 4.6 in the Listens table.
* **Calculating the average rating per song**, aggregated across all users who rated them.
* **Listing songs by specific artists**, such as Taylor Swift and Ed Sheeran, and querying songs that belong to either the Pop or Rock genres.
* **Finding songs that were listened to without a timestamp**, i.e., listens that occurred without a logged time, indicating potentially incomplete data.

### ****4. Collaborative Filtering: Song Recommendation Logic****

* A **common song listening pattern** is detected by identifying **song pairs** that have been listened to by the same user — but not necessarily at the same time.
* A **collaborative filtering approach** is applied to generate recommendations. For each user, the system recommends songs they haven't listened to but which have been listened to by others with overlapping listening histories.
* This is implemented using a **self-join on the Listens table** and filtering out already-heard songs per user.
* The final recommendation results — including the recommended song, user, and timestamp — are **inserted into the** Recommendations **table** using a WITH clause and a ROW\_NUMBER() window function for unique IDs.

### ****5. Personalized Recommendations****

This section focuses on generating **user-specific recommendations** using advanced filtering techniques, particularly for the user **Minnie (user\_id = 2)**. The goal is to refine recommendations beyond general collaborative filtering by considering listening behavior and recency.

#### ****Q2: Basic Personalized Recommendations****

* Recommendations are generated for **Minnie** by identifying other users with overlapping listening histories.
* The system suggests songs that Minnie has **not yet listened to**, but that appear in the listening records of similar users.
* The result is a list of personalized song suggestions based on shared interests.

#### ****Q3: Time-Filtered Recommendations****

* A more refined strategy is implemented where **only recent listens** (i.e., records with a non-null listen\_time) are considered.
* This ensures that the recommendations reflect **current listening trends** and **active user preferences**, rather than historical data.

#### ****Q4: Timestamp-Based Profile for Minnie****

* Minnie's **recent listening history** is analyzed to understand her current preferences.
* She has listened to tracks like **"Yellow Submarine"** and **"Hey Jude"** by The Beatles, indicating an inclination toward **classic rock**.
* This profile is then compared with the recent histories of other users to identify additional relevant tracks she hasn’t heard.

#### ****Q5: Static vs Time-Sensitive Recommendation Comparison****

| Aspect | Static Method | Time-Based Method |
| --- | --- | --- |
| **Data Used** | Full listens table | Only recent listens (non-null timestamps) |
| **Recency Considered** | ❌ Not considered | ✅ Yes, prioritized |
| **Behavior Modeled** | General user similarity | Current preferences |
| **Recommendations for Minnie** | "Shape of You", "Bad Blood" | "Shape of You" (more precise) |
| **Filtering Effect** | Broader but less relevant | Fewer, more relevant |
| **Timestamp Sensitivity** | ❌ Not used | ✅ Used to focus recommendations |

### ****Conclusion****

This Python–MySQL integration script successfully simulates the core functionalities of a music recommendation engine, using DBeaver as the interface on Parrot OS. It combines structured data modeling with SQL-based analytics to manage user behavior and song metadata effectively.

Through a series of filtering and collaborative queries, the system demonstrates how personalized content can be generated from basic user interactions. Personalized recommendations—both static and time-sensitive—highlight the power of analyzing listening patterns for deeper insights.

The modular structure and clear separation of data creation, insertion, analytics, and recommendation logic lay the groundwork for extending this system to more advanced use cases, such as:

* **User profiling**
* **Genre affinity modeling**
* **Real-time music recommendation using machine learning**

**Git Link: https://github.com/Pranav1114/Big\_data**