TOPIC:SONG DATA VISUALIZATION

ROLL NO:
UCE2023530-DISHA JANVE
UCE2023531-ADITI JOSHI
UCE2023532-JUI KULKARNI
UCE2023546-ANJALI PATEL
BATCH-B2

SHORT FUNCTIONAL REQUIREMENT SYSTEM(SRS)

Project Overview: Spotify Data Visualizer

The Spotify Data Visualizer project is a Java-based application that provides users with insightful and interactive ways to explore music data. Utilizing a MySQL database, the project enables users to retrieve, analyze, and interact with song, artist, and user listening data. By combining backend processing with user input, the application delivers a streamlined experience for music enthusiasts and database administrators alike.

Key Objectives -

1. Provide users with insights into their music listening habits, including top songs, genres, and active listening hours.
2. Enable dynamic exploration of music data, such as artist statistics, track details, and historical song releases.
3. Offer a user-friendly way to manage music-related data, including adding new tracks to the database.
4. Serve as an educational tool showcasing SQL integration with Java for database querying and manipulation.

Core Functionalities -

Based on the functional requirements:

1. Track Analysis:
   - Identify the most-played tracks by country and user.
   - Highlight the user's most-listened-to song and favorite genre.
   - Display a random selection of songs from a specific time frame (e.g., 2011–2015).

2.Recommendations:
   - Suggest new songs based on followed artists.
    -Recommend tracks from unfamiliar artists for a more diverse listening experience.

3. Artist Insights:
Determine the top artist for a user by play count.
   - Showcase artists with the highest average number of tracks per album.

4. User Habit Analysis:
Identify peak listening hours.
Provide insights into users' most played songs and genres.

5.Data Management:
   - Allow artists to add new tracks with relevant metadata, such as title, genre, and duration.
   - Facilitate exploration of track details by specific criteria (e.g., titles starting with a given letter).

System Features -

1. Interactive User Queries:
   Users can query and retrieve specific details using intuitive console-based commands and inputs.

2. Dynamic Recommendations:
   Leveraging SQL joins and filters, the system provides personalized recommendations tailored to user preferences.

3. Real-Time Insights:
   Users receive instant feedback on their queries, such as top tracks or listening trends.

4. Data Integration and Management:

Tracks can be added dynamically to the database, ensuring the data remains up-to-date and relevant.

Technology Stack-

- **Programming Language:** Java
- **Database Management System:** MySQL
- **Development Environment:** Eclipse IDE
- **Database Connectivity:** JDBC (Java Database Connectivity)

System Architecture-

1. User Interface Layer:
Console-based application that accepts user input for queries and displays results in a formatted manner.

2. Logic Layer:
Java methods implement business logic for each functional requirement, such as retrieving top tracks, calculating averages, and inserting new tracks.

3. Database Layer: MySQL database serves as the storage for all music-related data, including tracks, artists, albums, and listening history.

Potential Use Cases -

1. A music enthusiast looking to explore their listening habits and discover new music.
2. A database administrator managing a music catalog and updating track details.
3. A developer learning to integrate SQL with Java for real-world applications.

Functional Requirements-

The project consists of **10 distinct operations** available to the user through the Java application, each mapped to specific SQL functionalities or queries.

In total, **12 SQL queries** have been incorporated to enable various database operations, leveraging advanced SQL concepts for optimal performance and functionality.

Clauses Used-

- Like
- Between
- Queries on dates
- Order by
- Group by
- Joins
- Subquery
- Views
- Trigger
- Procedure
- Cursor

QUERIES-

1]Software Requirements Specification for "Hot Hits by Country" Query

The "Hot Hits by Country" query retrieves the top 10 most-played tracks in a specified country, allowing users to view popular songs by region.The query pulls tracks from listening history, filters by country, calculates play counts, and returns the top 10 songs in descending order of popularity, showcasing regional music trends.

Clauses used: Join,GroupBY,OrderBY,Count,Limit

2]Software Requirements Specification for "Jump Back In" Query

The "Top Tracks by User" query retrieves the top 10 most-listened-to tracks for a specific user, enabling personalized recommendations by showing frequently played songs.
The query pulls tracks from listening history, filters by country, calculates play counts, and returns the top 10 songs in descending order of popularity, showcasing regional music trends.

Clauses used: Join,GroupBY,OrderBY,Count

3]Software Requirements Specification (SRS) for "Recommend Songs Based on Listening History and Preferred Artists" Query

The query recommends new songs from followed artists that a user has not yet listened to, enhancing discovery of similar music.
The query pulls tracks from listening history, filters by country, calculates play counts, and returns the top 10 songs in descending order of popularity, showcasing regional music trends.

Clauses used: Join,GroupBY,OrderBY,Limit

4]Software Requirements Specification (SRS) for "Discover New Songs" Query

The "Discover New Songs" query is designed to recommend songs that a user hasn't heard yet, focusing on tracks by artists the user doesn't follow. This feature helps users expand their music experience with fresh recommendations.
The query pulls tracks from listening history, filters by country, calculates play counts, and returns the top 10 songs in descending order of popularity, showcasing regional music trends.

Clauses used: Join,GroupBY,OrderBY,Limit

5]SRS(Software Requirements Specification) for Top Artist by User

The "Top Artist by User" query is designed to determine the artist that a specific user listens to the most. By analyzing the user's listening history, the system identifies the artist with the highest play count and returns that artist's information.

Clauses used: Join,GroupBY,OrderBY,Limit,Count

6]Software Requirements Specification (SRS) for "Top Song" Query

The purpose of this query is to identify the most-played song by a specific user based on their listening history. The system will return the song with the highest number of plays for the given user. This query will:

Identify the most-played song for a given user by analyzing the number of times each song appears in the user's listening history.

Return the trackID, title, and play count of the song that was played the most by the user.

Clauses used: Join,GroupBY,OrderBY,Limit,Count

7] Software Requirements Specification (SRS) for the `favouriteGenre` function:

The `favouriteGenre` function identifies the most frequently played genre for a specific user based on their listening history. This helps to personalize the user experience by understanding their preferences. It interacts with a database containing user listening history and track details. The output is the user's favorite genre along with the play count for that genre.

Clauses used: Join,GroupBY,OrderBY,Limit,Count

8]Software Requirements Specification (SRS) for the `mostActiveListeningHours` function:

The `mostActiveListeningHours` function determines the hour of the day when a user listens to the most tracks. This insight can be used for user behavior analysis and improving recommendations or engagement strategies.It processes listening history data to find the most active hour for a given user and outputs the hour and associated listening count.

Clauses used: GroupBY,OrderBY,Limit

9]Software Requirements Specification (SRS) for the `discoverSongs2011to2015` function:

The `discoverSongs2011to2015` function retrieves and displays a random selection of 10 songs released between the years 2011 and 2015. This functionality is useful for exploring tracks from a specific period, enriching user experience with tailored discovery features. It queries the database to find songs released during a given time range and displays their details in a tabular format.

Clauses used: Join,Limit,OrderBY,Between

10]Software Requirements Specification (SRS) for Average tracks of Artists

The **artistAverageTracks** function retrieves and displays the top 10 artists based on their average number of tracks per album. This functionality provides insights into artist productivity and helps users understand which artists produce the most content relative to their albums.The function is designed to process artist data from a pre-aggregated table (ArtistAverageTracks) in the database. It showcases artists' average track counts in descending order.

Clauses used: OrderBY,Limit,View,Subquery

11]Software Requirements Specification (SRS) for Track Details By Alphabet

The **trackDetailsByAlphabet** function retrieves details of tracks whose titles start with a specific alphabet. It displays the first 10 results in a formatted table, aiding users in browsing tracks based on alphabetical indexing.
It provides an easy way to explore tracks by title initials and is useful for users who want a quick glance at tracks that start with a specific letter.

Clauses used: Join,OrderBY,Limit,Like

12]Software Requirements Specification (SRS) for Publish A Track

This query defines a system for automatically updating the mood of tracks in the track table based on specific attributes (tempo and genre) using a stored procedure, a trigger,The UpdateMood stored procedure sets the mood of a specific track based on its tempo and genre.The after_track_insert trigger automatically adds new tracks to the mood_update_queue right after they're inserted into the track table.

Clauses used: PL/SQL

ERD AND NORMALIZATION

We have used the following tables in the database:

1) **User (UserID, Name, Email, Country)**
2) **Artist (ArtistID, Name, Country)**
3) **Album (AlbumID, Title, ReleaseDate, ArtistID)**
4) **Track (TrackID, Title, Genre, Duration, AlbumID, Tempo , Mood)**
5) **ListeningHistory (HistoryID, UserID, TrackID, ListenDate, ListenDuration)**
6) **UserFollowsArtist (UserID, ArtistID, FollowDate)**
7) **UserLikesTrack (UserID, TrackID, LikeDate)**

Analysis for 3NF

1) User (UserID, Name, Email, Country)

- **Primary Key**: UserID
- **Non-key Attributes**: Name, Email, Country
- **Dependencies**: Each non-key attribute depends only on UserID (the primary key).
- **3NF**: Satisfied, as there are no transitive dependencies. Name, Email, and Country are attributes describing UserID alone.

2) Artist (ArtistID, Name, Country)

- **Primary Key**: ArtistID
- **Non-key Attributes**: Name, Country
- **Dependencies**: Name and Country depend only on ArtistID.
- **3NF**: Satisfied, as there are no transitive dependencies. Name and Country only describe ArtistID.

3) Album (AlbumID, Title, ReleaseDate, ArtistID)

- **Primary Key**: AlbumID
- **Non-key Attributes**: Title, ReleaseDate, ArtistID
- **Dependencies**: Title and ReleaseDate depend only on AlbumID. ArtistID associates the album with its artist, but it doesn't introduce any transitive dependency.
- **3NF**: Satisfied. Title and ReleaseDate only describe AlbumID, and ArtistID is a foreign key to relate to the artist without introducing transitive dependencies.

4) Track (TrackID, Title, Genre, Duration, AlbumID, Tempo, Mood)

- **Primary Key**: TrackID
- **Non-key Attributes**: Title, Genre, Duration, AlbumID, Tempo, Mood
- **Dependencies**: All attributes depend on TrackID. AlbumID is a foreign key to the Album table.
- **3NF**: Satisfied. All non-key attributes are related to TrackID without transitive dependencies. AlbumID relates to Album but doesn't affect 3NF as it's not creating additional dependencies among non-key attributes.

5) ListeningHistory (HistoryID, UserID, TrackID, ListenDate, ListenDuration)

- **Primary Key**: HistoryID
- **Non-key Attributes**: UserID, TrackID, ListenDate, ListenDuration
- **Dependencies**: All attributes are dependent only on HistoryID.
- **3NF**: Satisfied. Each HistoryID entry represents a unique listening record, with UserID and TrackID providing foreign keys to User and Track, respectively. No transitive dependencies exist.

6) UserFollowsArtist (UserID, ArtistID, FollowDate)

- **Composite Primary Key**: UserID, ArtistID
- **Non-key Attribute**: FollowDate
- **Dependencies**: FollowDate depends on the entire primary key (both UserID and ArtistID).
- **3NF**: Satisfied. The composite primary key uniquely identifies each follow relationship, and FollowDate depends directly on it without transitive dependencies.

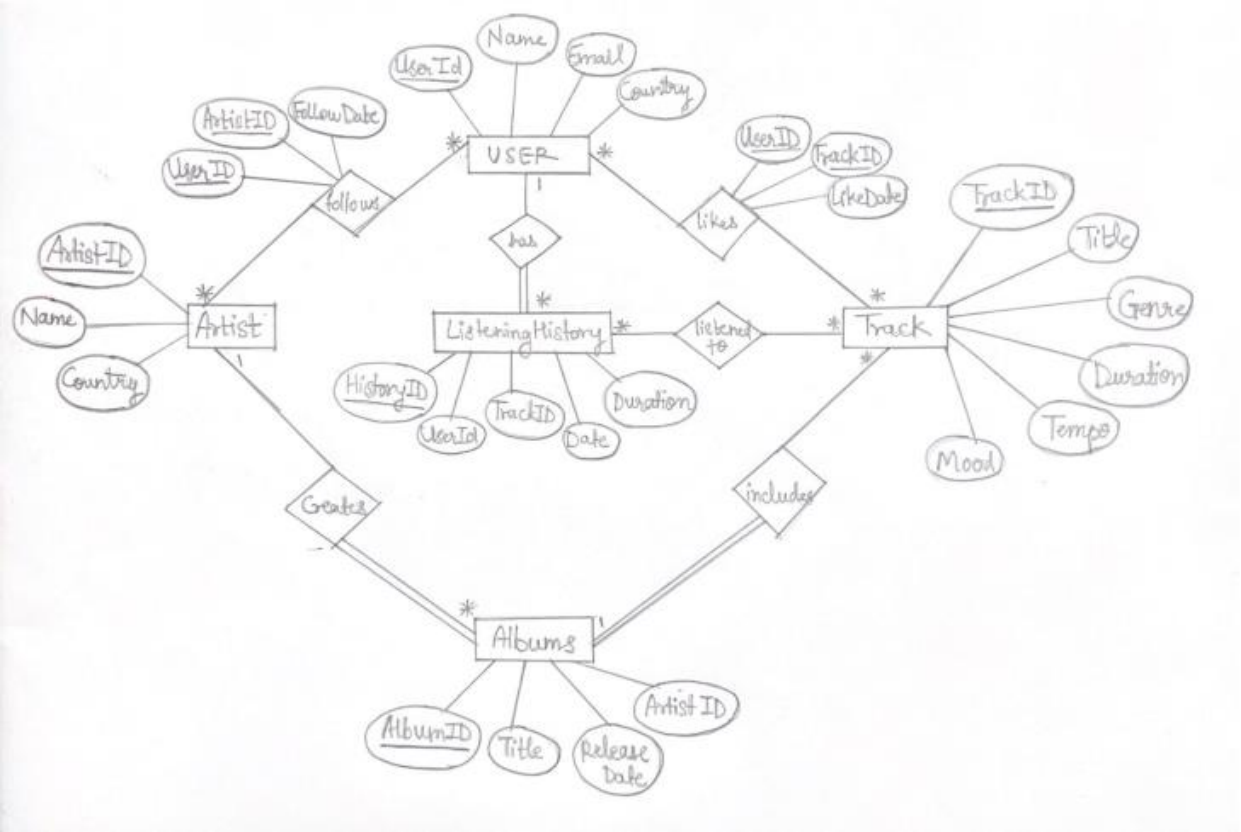
7) UserLikesTrack (UserID, TrackID, LikeDate)

- **Composite Primary Key**: UserID, TrackID
- **Non-key Attribute**: LikeDate
- **Dependencies**: LikeDate depends on the entire primary key (both UserID and TrackID).
- **3NF**: Satisfied. The composite primary key uniquely identifies each "like" relationship, and LikeDate depends directly on it without transitive dependencies.

Each table adheres to **3NF** as:

- Each table’s non-key attributes depend only on the primary key.
- Foreign keys are used to establish relationships between entities without creating transitive dependencies.

Since all table are in 3NF here is the ERD for the tables in the database:



CREATING TABLE

1.CREATE TABLE User (UserID INT PRIMARY KEY,Name VARCHAR(50),Email VARCHAR(100),country varchar(50));

2.CREATE TABLE Artist (ArtistID INT PRIMARY KEY,Name VARCHAR(50),Country VARCHAR(50));

3.CREATE TABLE Album (AlbumID INT PRIMARY KEY, Title VARCHAR(100), ReleaseDate DATE, ArtistID INT, FOREIGN KEY (ArtistID) REFERENCES Artist(ArtistID) ON DELETE CASCADE);

4.CREATE TABLE Track (TrackID INT PRIMARY KEY, Title VARCHAR(100), Genre VARCHAR(50), Duration TIME, Tempo INT, Mood VARCHAR(50), AlbumID INT, FOREIGN KEY (AlbumID) REFERENCES Album(AlbumID) ON DELETE CASCADE);

5.CREATE TABLE ListeningHistory (HistoryID INT PRIMARY KEY, UserID INT, TrackID INT, ListenDate TIMESTAMP, ListenDuration TIME, FOREIGN KEY (UserID)

REFERENCES User(UserID) ON DELETE CASCADE, FOREIGN KEY (TrackID) REFERENCES Track(TrackID) ON DELETE CASCADE);

6.CREATE TABLE UserFollowsArtist (UserID INT, ArtistID INT, FollowDate DATE, PRIMARY KEY (UserID, ArtistID), FOREIGN KEY (UserID) REFERENCES User(UserID) ON DELETE CASCADE, FOREIGN KEY (ArtistID) REFERENCES Artist(ArtistID) ON DELETE CASCADE);

7.CREATE TABLE UserLikesTrack (UserID INT, TrackID INT, LikeDate DATE, PRIMARY KEY (UserID, TrackID), FOREIGN KEY (UserID) REFERENCES User(UserID) ON DELETE CASCADE, FOREIGN KEY (TrackID) REFERENCES Track(TrackID) ON DELETE CASCADE);

QUERIES:

Q1) Determine the most listened songs in a country.

mysql> SELECT t.trackID, t.title, t.genre, COUNT(lh.trackID) AS play_count FROM listeninghistory lh JOIN user u ON lh.userID = u.userID JOIN track t ON lh.trackID = t.trackID WHERE u.country = 'India' GROUP BY t.trackID ORDER BY play_count DESC LIMIT 10;

+---------+----------------+----------+------------+

| trackID | title | genre | play_count |

+---------+----------------+----------+------------+

| 61 | Mile Ho Tum | Romantic | 3 |

| 72 | Sorry | Pop | 3 |

| 79 | No Pressure | R&B | 2 |

| 82 | Kiss It Better | Pop | 2 |

| 55 | Happier | Indie | 2 |

| 57 | Dive | Indie | 2 |

| 59 | What Do I Know? | Pop | 2 |

| 76 | The Feeling | Pop | 2 |

| 64 | Garmi | Dance | 2 |

| 75 | Company | Pop | 2 |

+---------+-----------------+----------+------------+

10 rows in set (0.00 sec)

Q2) What are the most listened to songs by a user?

mysql> SELECT t.trackID, t.title, t.genre, COUNT(lh.trackID) AS play_count FROM listeninghistory lh JOIN track t ON lh.trackID = t.trackID WHERE lh.userID =2 GROUP BY t.trackID ORDER BY play_count DESC LIMIT 10;

+---------+-------------------+-------+------------+

| trackID | title | genre | play_count |

+---------+-------------------+-------+------------+

| 72 | Sorry | Pop | 1 |

| 75 | Company | Pop | 1 |

| 76 | The Feeling | Pop | 1 |

| 77 | Purpose | Pop | 1 |

| 79 | No Pressure | R&B | 1 |

| 82 | Kiss It Better | Pop | 1 |

| 83 | Needed Me | R&B | 1 |

| 84 | Love on the Brain | Soul | 1 |

| 86 | Same Ol? Mistakes | R&B | 1 |

| 90 | Close to You | Pop | 1 |

+---------+-------------------+-------+------------+

10 rows in set (0.00 sec)

Q3) Recommend users songs based on their listening history.

mysql> SELECT t.trackID, t.title, t.genre, a.name AS artist_name FROM track t JOIN album al ON t.albumID = al.albumID JOIN artist a ON al.artistID = a.artistID JOIN userfollowsartist ufa ON ufa.artistID = a.artistID LEFT JOIN listeninghistory lh ON lh.trackID = t.trackID AND lh.userID = 2 WHERE ufa.userID = 2 AND lh.trackID IS NULL GROUP BY t.trackID ORDER BY t.genre, t.title LIMIT 10;

+---------+---------------------------+-----------+--------------+

| trackID | title | genre | artist_name |

+---------+---------------------------+-----------+--------------+

| 81 | Work | Dancehall | Rihanna |

| 39 | Try Me | Indie | The Weeknd |

| 15 | Wildest Dreams | Indie | Taylor Swift |

| 17 | All You Had to Do Was Stay | Pop | Taylor Swift |

| 36 | Alone Again | Pop | The Weeknd |

| 12 | Blank Space | Pop | Taylor Swift |

| 31 | Blinding Lights | Pop | The Weeknd |

| 85 | Desperado | Pop | Rihanna |

| 18 | How You Get the Girl | Pop | Taylor Swift |

| 33 | Save Your Tears | Pop | The Weeknd |

+---------+---------------------------+-----------+--------------+

10 rows in set (0.00 sec)

Q4) How would a user discover new songs?

mysql> SELECT t.trackID, t.title, t.genre, a.name AS artist_name FROM track t JOIN album al ON t.albumID = al.albumID JOIN artist a ON al.artistID = a.artistID LEFT JOIN userfollowsartist ufa ON ufa.artistID = a.artistID AND ufa.userID =2 LEFT JOIN listeninghistory lh ON lh.trackID = t.trackID AND lh.userID = 2 WHERE ufa.artistID IS NULL AND lh.trackID IS NULL GROUP BY t.trackID ORDER BY RAND() LIMIT 10;

+---------+----------------------+-----------+----------------+

| trackID | title | genre | artist_name |

+---------+----------------------+-----------+----------------+

| 30 | Hasi | Indie | Shreya Ghoshal |

| 50 | I Found a Boy | Pop | Adele |

| 67 | Sunny Sunny | Dance | Neha Kakkar |

| 27 | Saans | Romantic | Shreya Ghoshal |

| 46 | Don?t You Remember | Indie | Adele |

| 53 | Galway Girl | Pop | Ed Sheeran |

| 47 | Make You Feel My Love | Soul | Adele |

| 57 | Dive | Indie | Ed Sheeran |

| 62 | Aankh Marey | Bollywood | Neha Kakkar |

| 93 | The Greatest | Pop | Sia |

+---------+----------------------+-----------+---------------+

10 rows in set (0.01 sec)

Q5) Display the top artist of a user.

mysql> SELECT a.artistID, a.name AS artist_name, COUNT(lh.trackID) AS play_count FROM listeninghistory lh JOIN track t ON lh.trackID = t.trackID JOIN album al ON t.albumID = al.albumID JOIN artist a ON al.artistID = a.artistID WHERE lh.userID = 2 GROUP BY a.artistID ORDER BY play_count DESC LIMIT 1;

+----------+---------------+------------+

| artistID | artist_name | play_count |

+----------+---------------+------------+

| 8 | Justin Bieber | 5 |

+----------+---------------+------------+

1 row in set (0.00 sec)

Q6) Display the most listened song of the user.

mysql> SELECT t.trackID, t.title, COUNT(lh.trackID) AS play_count FROM listeninghistory lh JOIN track t ON lh.trackID = t.trackID WHERE lh.userID =2 GROUP BY t.trackID ORDER BY play_count DESC LIMIT 1;

+---------+-------+------------+

| trackID | title | play_count |

+---------+-------+------------+

| 72 | Sorry | 1 |

+---------+-------+------------+

1 row in set (0.00 sec)

Q7) Display the Favourite Genre of the user

```
mysql> SELECT t.genre, COUNT(lh.trackID) AS play_count FROM listeninghistory lh JOIN track t ON lh.trackID = t.trackID WHERE lh.userID = 2 GROUP BY t.genre ORDER BY play_count DESC LIMIT 1;

+-------+------------+

| genre | play_count |

+-------+------------+

| Pop   |          6 |

+-------+------------+

1 row in set (0.00 sec)
```

Q8) Display the most listening hours of the user.

```
mysql> SELECT HOUR(lh.listenDate) AS listen_hour, COUNT(lh.trackID) AS listen_count FROM listeninghistory lh WHERE lh.userID = 2  GROUP BY listen_hour ORDER BY listen_count DESC LIMIT 1;

+-------------+--------------+

| listen_hour | listen_count |

+-------------+--------------+

|           9 |            1 |

+-------------+--------------+

1 row in set (0.00 sec)
```

Q9) Display any 10 songs between the years 2011 and 2015.

```
mysql> SELECT t.trackID, t.title, t.genre, a.releasedate FROM track t JOIN album a ON t.albumID = a.albumID WHERE a.releasedate BETWEEN '2011-01-01' AND '2015-12-31' ORDER BY RAND() LIMIT 10;

+---------+--------------------------+-------+-------------+

| trackID | title                    | genre | releasedate |

+---------+--------------------------+-------+-------------+

|      47 | Make You Feel My Love    | Soul  | 2011-11-09  |

|      46 | Don?t You Remember       | Indie | 2011-11-09  |

|      45 | Turning Tables           | Pop   | 2011-11-09  |

|      18 | How You Get the Girl     | Pop   | 2014-10-27  |

|      71 | What Do You Mean?        | Pop   | 2015-11-13  |

|      77 | Purpose                  | Pop   | 2015-11-13  |

|      78 | Mark My Words            | R&B   | 2015-11-13  |

|      17 | All You Had to Do Was Stay | Pop | 2014-10-27  |

|      15 | Wildest Dreams           | Indie | 2014-10-27  |

|      42 | Someone Like You         | Soul  | 2011-11-09  |

+---------+--------------------------+-------+-------------+

10 rows in set (0.00 sec)
```

Q10) Display the total number of songs by every artist

```
mysql> CREATE VIEW ArtistAverageTracks AS SELECT a.artistID, a.name AS artist_name, ( SELECT AVG(track_count) FROM ( SELECT COUNT(t.trackID) AS track_count FROM album al JOIN track t ON al.albumID = t.albumID WHERE al.artistID = a.artistID GROUP BY al.albumID ) AS album_track_counts ) AS average_tracks_per_album FROM artist a;
```

Query OK, 0 rows affected (0.01 sec)

mysql> select * from artistaveragetracks;

```
+----------+---------------+-------------------------+
| artistID | artist_name    | average_tracks_per_album |
+----------+---------------+-------------------------+
|        1 | Arijit Singh   |                  10.0000 |
|        2 | Taylor Swift   |                  10.0000 |
|        3 | Shreya Ghoshal |                  10.0000 |
|        4 | The Weeknd     |                  10.0000 |
|        5 | Adele          |                  10.0000 |
|        6 | Ed Sheeran     |                  10.0000 |
|        7 | Neha Kakkar    |                  10.0000 |
|        8 | Justin Bieber  |                  10.0000 |
|        9 | Rihanna        |                  10.0000 |
|       10 | Sia            |                  10.0000 |
+----------+---------------+-------------------------+
```

10 rows in set (0.00 sec)

Q11) Search for a song based on their alphabet.

```
SELECT t.trackID, t.title, t.genre, a.name AS artist_name
FROM track t
JOIN album al ON t.albumID = al.albumID
JOIN artist a ON al.artistID = a.artistID
```

```
WHERE t.title LIKE "a%"

ORDER BY t.title LIMIT 10
```

```
+---------+--------------------------+-----------+--------------+
| trackID | title                    | genre     | artist_name  |
+---------+--------------------------+-----------+--------------+
|      62 | Aankh Marey              | Bollywood | Neha Kakkar  |
|      35 | After Hours              | R&B       | The Weeknd   |
|       2 | Agar Tum Saath Ho        | Rock      | Arijit Singh |
|      92 | Alive                    | Pop       | Sia          |
|      17 | All You Had to Do Was Stay | Pop       | Taylor Swift |
|      36 | Alone Again              | Pop       | The Weeknd   |
+---------+--------------------------+-----------+--------------+
6 rows in set (0.11 sec)
```

Q12) Insert a new track in the artist table and set its mood based on it’s tempo and genre.

```
mysql> DELIMITER //

mysql> CREATE PROCEDURE UpdateMood()
    -> BEGIN
    ->     UPDATE track
    ->     SET mood = CASE
    ->         WHEN tempo >= 120 AND genre IN ('pop', 'dance', 'hip-hop') THEN 'energetic'
```

```
    ->         WHEN tempo BETWEEN 80 AND 120 AND genre IN ('pop', 'rock') THEN 'happy'
    ->         WHEN tempo BETWEEN 60 AND 80 AND genre IN ('jazz', 'blues') THEN 'calm'
    ->         WHEN tempo < 60 AND genre IN ('classical', 'ambient') THEN 'melancholic'
    ->         ELSE 'neutral'
    ->     END;
    -> END //
Query OK, 0 rows affected (0.01 sec)

mysql>
mysql> DELIMITER ;
mysql> call updatemood();
Query OK, 100 rows affected (0.01 sec)

mysql> select * from track;
+---------+--------------------------+-----------+----------+-------+-----------+---------+
| TrackID | Title                    | Genre     | Duration | Tempo | Mood      | AlbumID |
+---------+--------------------------+-----------+----------+-------+-----------+---------+
|       1 | Tum Hi Ho                | Pop       | 00:04:35 |    70 | neutral   |       1 |
|       2 | Agar Tum Saath Ho        | Rock      | 00:05:10 |    85 | happy     |       1 |
|       3 | Raabta                   | Classical | 00:04:12 |    65 | neutral   |       1 |
|       4 | Channa Mereya            | Jazz      | 00:04:20 |    80 | calm      |       1 |
|       5 | Phir Le Aya Dil          | Pop       | 00:03:50 |    90 | happy     |       1 |
|       6 | Tum Jo Aaye              | Indie     | 00:04:40 |    95 | neutral   |       1 |
```

| 7 | Tum Hi Ho (Reprise) | Soul | 00:03:55 | 75 | neutral | 1 |

| 8 | Pachtaoge | Folk | 00:04:05 | 80 | neutral | 1 |

| 9 | Muskurane | Pop | 00:04:25 | 95 | happy | 1 |

| 10 | Tum Se Hi | Classical | 00:04:00 | 60 | neutral | 1 |

| 11 | Shake It Off | Pop | 00:03:39 | 120 | energetic | 2 |

| 12 | Blank Space | Pop | 00:03:51 | 100 | happy | 2 |

| 13 | Style | Pop | 00:03:51 | 115 | happy | 2 |

| 14 | Bad Blood | Rock | 00:03:30 | 130 | neutral | 2 |

| 15 | Wildest Dreams | Indie | 00:03:40 | 85 | neutral | 2 |

| 16 | Out of the Woods | Synth-Pop | 00:03:55 | 115 | neutral | 2 |

| 17 | All You Had to Do Was Stay | Pop | 00:03:00 | 105 | happy | 2 |

| 18 | How You Get the Girl | Pop | 00:03:30 | 110 | happy | 2 |

| 19 | This Love | Pop | 00:04:05 | 90 | happy | 2 |

| 20 | New Romantics | Synth-Pop | 00:03:30 | 120 | neutral | 2 |

| 21 | Sun Raha Hai | Classical | 00:05:30 | 60 | neutral | 3 |

| 22 | Dola Re Dola | Folk | 00:04:10 | 100 | neutral | 3 |

| 23 | Teri Meri | Romantic | 00:04:30 | 75 | neutral | 3 |

| 24 | Piyu Bole | Classical | 00:05:05 | 65 | neutral | 3 |

| 25 | Jab Tak | Indie | 00:04:25 | 95 | neutral | 3 |

| 26 | Tujh Mein Rab Dikhta Hai | Classical | 00:04:15 | 60 | neutral | 3 |

| 27 | Saans | Romantic | 00:04:20 | 85 | neutral | 3 |

| 28 | Sunn Raha Hai | Soul | 00:04:50 | 70 | neutral | 3 |

| 29 | Chikni Chameli | Pop | 00:03:30 | 125 | energetic | 3 |

30	Hasi	Indie	00:04:15	95	neutral	3
31	Blinding Lights	Pop	00:03:20	115	happy	4
32	Heartless	R&B	00:03:15	105	neutral	4
33	Save Your Tears	Pop	00:03:35	110	happy	4
34	In Your Eyes	Synthwave	00:03:55	110	neutral	4
35	After Hours	R&B	00:04:15	90	neutral	4
36	Alone Again	Pop	00:04:05	100	happy	4
37	Too Late	Pop	00:03:45	115	happy	4
38	Scared to Live	Soul	00:04:00	75	neutral	4
39	Try Me	Indie	00:03:50	95	neutral	4
40	Nothing Compares	R&B	00:04:20	80	neutral	4
41	Rolling in the Deep	Pop	00:03:50	100	happy	5
42	Someone Like You	Soul	00:04:40	60	neutral	5
43	Set Fire to the Rain	Pop	00:04:10	110	happy	5
44	Rumour Has It	Rock	00:03:45	120	happy	5
45	Turning Tables	Pop	00:04:05	85	happy	5
46	Don?t You Remember	Indie	00:03:50	95	neutral	5
47	Make You Feel My Love	Soul	00:03:30	70	neutral	5
48	One and Only	Pop	00:04:15	100	happy	5
49	Lovesong	Indie	00:03:55	95	neutral	5
50	I Found a Boy	Pop	00:04:00	100	happy	5
51	Shape of You	Pop	00:03:53	96	happy	6
52	Castle on the Hill	Rock	00:04:21	138	neutral	6

53	Galway Girl	Pop	00:02:50	112	happy	6
54	Perfect	Pop	00:04:23	63	neutral	6
55	Happier	Indie	00:03:28	94	neutral	6
56	New Man	Pop	00:03:09	108	happy	6
57	Dive	Indie	00:03:58	82	neutral	6
58	Shape of You (Reprise)	Soul	00:03:45	96	neutral	6
59	What Do I Know?	Pop	00:03:55	108	happy	6
60	Bloodstream	Rock	00:05:00	128	neutral	6
61	Mile Ho Tum	Romantic	00:04:10	68	neutral	7
62	Aankh Marey	Bollywood	00:03:35	125	neutral	7
63	Kala Chashma	Dance	00:03:50	127	energetic	7
64	Garmi	Dance	00:03:45	132	energetic	7
65	Dil Dhadakne Do	Pop	00:04:00	115	happy	7
66	Lahu Munh Lag Gaya	Romantic	00:04:25	75	neutral	7
67	Sunny Sunny	Dance	00:03:30	130	energetic	7
68	Laung Laachi	Folk	00:04:00	90	neutral	7
69	Cheez Badi	Dance	00:03:40	126	energetic	7
70	Tera Ban Jaunga	Romantic	00:04:15	72	neutral	7
71	What Do You Mean?	Pop	00:03:30	125	energetic	8
72	Sorry	Pop	00:03:20	100	happy	8
73	Love Yourself	Pop	00:03:53	79	neutral	8
74	Where Are Ü Now	EDM	00:04:10	108	neutral	8
75	Company	Pop	00:03:07	95	happy	8

76	The Feeling	Pop	00:03:10	102	happy	8
77	Purpose	Pop	00:03:30	95	happy	8
78	Mark My Words	R&B	00:03:25	80	neutral	8
79	No Pressure	R&B	00:04:00	75	neutral	8
80	Children	Pop	00:03:45	116	happy	8
81	Work	Dancehall	00:03:39	95	neutral	9
82	Kiss It Better	Pop	00:04:13	106	happy	9
83	Needed Me	R&B	00:03:11	92	neutral	9
84	Love on the Brain	Soul	00:04:23	65	neutral	9
85	Desperado	Pop	00:03:00	124	energetic	9
86	Same Ol? Mistakes	R&B	00:06:38	65	neutral	9
87	Never Ending	R&B	00:03:05	88	neutral	9
88	Yeah, I Said It	Pop	00:03:25	108	happy	9
89	Consideration	R&B	00:03:35	85	neutral	9
90	Close to You	Pop	00:03:40	88	happy	9
91	Cheap Thrills	Pop	00:03:31	95	happy	10
92	Alive	Pop	00:04:23	112	happy	10
93	The Greatest	Pop	00:03:29	107	happy	10
94	Elastic Heart	Pop	00:04:14	110	happy	10
95	Unstoppable	Pop	00:03:38	123	energetic	10
96	Bird Set Free	Pop	00:04:05	111	happy	10
97	Move Your Body	Dance	00:03:33	122	energetic	10
98	Reaper	Pop	00:04:10	122	energetic	10

```
|     99 | Space Between            | Pop       | 00:04:00 |   108 | happy     |      10 |
|    100 | Dressed in Black         | Pop       | 00:04:23 |   118 | happy     |      10 |
+--------+--------------------------+-----------+----------+-------+-----------+---------+
100 rows in set (0.00 sec)
mysql> CREATE TABLE mood_update_queue (    trackID INT);
Query OK, 0 rows affected (0.03 sec)

mysql> DELIMITER //
mysql>
mysql> CREATE TRIGGER after_track_insert
   -> AFTER INSERT ON track
   -> FOR EACH ROW
   -> BEGIN
   ->     INSERT INTO mood_update_queue (trackID)
   ->     VALUES (NEW.trackID);
   -> END //
Query OK, 0 rows affected (0.01 sec)

mysql>
mysql> DELIMITER ;
mysql> DELIMITER //
mysql>
mysql> CREATE EVENT update_mood_event
```

```
-> ON SCHEDULE EVERY 1 MINUTE
-> DO
-> BEGIN
->
->    DECLARE done INT DEFAULT 0;
->    DECLARE track_id INT;
->    DECLARE cur CURSOR FOR SELECT trackID FROM mood_update_queue;
->
->
->    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;
->
->    OPEN cur;
->
->
->    REPEAT
->       FETCH cur INTO track_id;
->       IF NOT done THEN
->
->          CALL UpdateMood(track_id);
->       END IF;
->    UNTIL done END REPEAT;
->
->    CLOSE cur;
```

```
    ->
    ->
    ->     DELETE FROM mood_update_queue;
    -> END //
Query OK, 0 rows affected (0.02 sec)

mysql>
mysql> DELIMITER ;
mysql> insert into track values(101,'abc','pop','00:03:50',120,null,10);
Query OK, 1 row affected (0.01 sec)

mysql> select * from mood_update_queue;
+---------+
| trackID |
+---------+
|     101 |
+---------+
1 row in set (0.00 sec)

mysql> select * from track;
+---------+--------------------------+-----------+----------+-------+-----------+---------+
| TrackID | Title                    | Genre     | Duration | Tempo | Mood      | AlbumID |
+---------+--------------------------+-----------+----------+-------+-----------+---------+
```

1	Tum Hi Ho	Pop	00:04:35	70	neutral	1
2	Agar Tum Saath Ho	Rock	00:05:10	85	happy	1
3	Raabta	Classical	00:04:12	65	neutral	1
4	Channa Mereya	Jazz	00:04:20	80	calm	1
5	Phir Le Aya Dil	Pop	00:03:50	90	happy	1
6	Tum Jo Aaye	Indie	00:04:40	95	neutral	1
7	Tum Hi Ho (Reprise)	Soul	00:03:55	75	neutral	1
8	Pachtaoge	Folk	00:04:05	80	neutral	1
9	Muskurane	Pop	00:04:25	95	happy	1
10	Tum Se Hi	Classical	00:04:00	60	neutral	1
11	Shake It Off	Pop	00:03:39	120	energetic	2
12	Blank Space	Pop	00:03:51	100	happy	2
13	Style	Pop	00:03:51	115	happy	2
14	Bad Blood	Rock	00:03:30	130	neutral	2
15	Wildest Dreams	Indie	00:03:40	85	neutral	2
16	Out of the Woods	Synth-Pop	00:03:55	115	neutral	2
17	All You Had to Do Was Stay	Pop	00:03:00	105	happy	2
18	How You Get the Girl	Pop	00:03:30	110	happy	2
19	This Love	Pop	00:04:05	90	happy	2
20	New Romantics	Synth-Pop	00:03:30	120	neutral	2
21	Sun Raha Hai	Classical	00:05:30	60	neutral	3
22	Dola Re Dola	Folk	00:04:10	100	neutral	3
23	Teri Meri	Romantic	00:04:30	75	neutral	3

| 24 | Piyu Bole | Classical | 00:05:05 | 65 | neutral | 3 |

| 25 | Jab Tak | Indie | 00:04:25 | 95 | neutral | 3 |

| 26 | Tujh Mein Rab Dikhta Hai | Classical | 00:04:15 | 60 | neutral | 3 |

| 27 | Saans | Romantic | 00:04:20 | 85 | neutral | 3 |

| 28 | Sunn Raha Hai | Soul | 00:04:50 | 70 | neutral | 3 |

| 29 | Chikni Chameli | Pop | 00:03:30 | 125 | energetic | 3 |

| 30 | Hasi | Indie | 00:04:15 | 95 | neutral | 3 |

| 31 | Blinding Lights | Pop | 00:03:20 | 115 | happy | 4 |

| 32 | Heartless | R&B | 00:03:15 | 105 | neutral | 4 |

| 33 | Save Your Tears | Pop | 00:03:35 | 110 | happy | 4 |

| 34 | In Your Eyes | Synthwave | 00:03:55 | 110 | neutral | 4 |

| 35 | After Hours | R&B | 00:04:15 | 90 | neutral | 4 |

| 36 | Alone Again | Pop | 00:04:05 | 100 | happy | 4 |

| 37 | Too Late | Pop | 00:03:45 | 115 | happy | 4 |

| 38 | Scared to Live | Soul | 00:04:00 | 75 | neutral | 4 |

| 39 | Try Me | Indie | 00:03:50 | 95 | neutral | 4 |

| 40 | Nothing Compares | R&B | 00:04:20 | 80 | neutral | 4 |

| 41 | Rolling in the Deep | Pop | 00:03:50 | 100 | happy | 5 |

| 42 | Someone Like You | Soul | 00:04:40 | 60 | neutral | 5 |

| 43 | Set Fire to the Rain | Pop | 00:04:10 | 110 | happy | 5 |

| 44 | Rumour Has It | Rock | 00:03:45 | 120 | happy | 5 |

| 45 | Turning Tables | Pop | 00:04:05 | 85 | happy | 5 |

| 46 | Don?t You Remember | Indie | 00:03:50 | 95 | neutral | 5 |

47	Make You Feel My Love	Soul	00:03:30	70	neutral	5
48	One and Only	Pop	00:04:15	100	happy	5
49	Lovesong	Indie	00:03:55	95	neutral	5
50	I Found a Boy	Pop	00:04:00	100	happy	5
51	Shape of You	Pop	00:03:53	96	happy	6
52	Castle on the Hill	Rock	00:04:21	138	neutral	6
53	Galway Girl	Pop	00:02:50	112	happy	6
54	Perfect	Pop	00:04:23	63	neutral	6
55	Happier	Indie	00:03:28	94	neutral	6
56	New Man	Pop	00:03:09	108	happy	6
57	Dive	Indie	00:03:58	82	neutral	6
58	Shape of You (Reprise)	Soul	00:03:45	96	neutral	6
59	What Do I Know?	Pop	00:03:55	108	happy	6
60	Bloodstream	Rock	00:05:00	128	neutral	6
61	Mile Ho Tum	Romantic	00:04:10	68	neutral	7
62	Aankh Marey	Bollywood	00:03:35	125	neutral	7
63	Kala Chashma	Dance	00:03:50	127	energetic	7
64	Garmi	Dance	00:03:45	132	energetic	7
65	Dil Dhadakne Do	Pop	00:04:00	115	happy	7
66	Lahu Munh Lag Gaya	Romantic	00:04:25	75	neutral	7
67	Sunny Sunny	Dance	00:03:30	130	energetic	7
68	Laung Laachi	Folk	00:04:00	90	neutral	7
69	Cheez Badi	Dance	00:03:40	126	energetic	7

70	Tera Ban Jaunga	Romantic	00:04:15	72	neutral	7
71	What Do You Mean?	Pop	00:03:30	125	energetic	8
72	Sorry	Pop	00:03:20	100	happy	8
73	Love Yourself	Pop	00:03:53	79	neutral	8
74	Where Are Ü Now	EDM	00:04:10	108	neutral	8
75	Company	Pop	00:03:07	95	happy	8
76	The Feeling	Pop	00:03:10	102	happy	8
77	Purpose	Pop	00:03:30	95	happy	8
78	Mark My Words	R&B	00:03:25	80	neutral	8
79	No Pressure	R&B	00:04:00	75	neutral	8
80	Children	Pop	00:03:45	116	happy	8
81	Work	Dancehall	00:03:39	95	neutral	9
82	Kiss It Better	Pop	00:04:13	106	happy	9
83	Needed Me	R&B	00:03:11	92	neutral	9
84	Love on the Brain	Soul	00:04:23	65	neutral	9
85	Desperado	Pop	00:03:00	124	energetic	9
86	Same Ol? Mistakes	R&B	00:06:38	65	neutral	9
87	Never Ending	R&B	00:03:05	88	neutral	9
88	Yeah, I Said It	Pop	00:03:25	108	happy	9
89	Consideration	R&B	00:03:35	85	neutral	9
90	Close to You	Pop	00:03:40	88	happy	9
91	Cheap Thrills	Pop	00:03:31	95	happy	10
92	Alive	Pop	00:04:23	112	happy	10

| 93 | The Greatest | Pop | 00:03:29 | 107 | happy | 10 |

| 94 | Elastic Heart | Pop | 00:04:14 | 110 | happy | 10 |

| 95 | Unstoppable | Pop | 00:03:38 | 123 | energetic | 10 |

| 96 | Bird Set Free | Pop | 00:04:05 | 111 | happy | 10 |

| 97 | Move Your Body | Dance | 00:03:33 | 122 | energetic | 10 |

| 98 | Reaper | Pop | 00:04:10 | 122 | energetic | 10 |

| 99 | Space Between | Pop | 00:04:00 | 108 | happy | 10 |

| 100 | Dressed in Black | Pop | 00:04:23 | 118 | happy | 10 |

| 101 | abc | pop | 00:03:50 | 120 | energetic | 10 |

+---------+--------------------------+-----------+----------+-------+-----------+---------+

101 rows in set (0.00 sec)

mysql> select * from mood_update_queue;

Empty set (0.00 sec)

VIDEO DEMONSTRATION

SAMPLE CODE WITH INPUT OUTPUT(ON ECLIPSE)

```
package miniproj;

import java.sql.*;

import java.util.*;

import java.sql.Date;
```

```
public class MyClass {


        public static void main(String[] args) throws ClassNotFoundException, SQLException {

                Class.forName("com.mysql.cj.jdbc.Driver");

                Connection
con=DriverManager.getConnection("jdbc:mysql://localhost:3306/minipro","root","Ramraya2308
$");

                Scanner scanner=new Scanner(System.in);

                int choice = 0;

      while (choice != 10) {  // Keep showing the menu until user chooses 10 to exit

        // Display menu

        System.out.println("\nChoose an operation to perform (1-9), or enter 10 to exit:");

        System.out.println("1. Hot Hits in your country");

        System.out.println("2. Jump Back In");

        System.out.println("3. Picked For You");

        System.out.println("4. Discover");

        System.out.println("5. Your Wrap");

        System.out.println("6. Best Of 2011 to 2015");

        System.out.println("7. View the Number of Songs of every Artist");

        System.out.println("8. Search A Song");

        System.out.println("9. Publish a new Track");

        System.out.println("10. Exit");
```

```
    // Get user's choice
    choice = scanner.nextInt();

scanner.nextLine(); // Consume the newline character

switch (choice) {
    case 1:
        // Query 1: Hot Hits in a specific country
        System.out.print("Enter the country: ");
        String country = scanner.nextLine();
        hotHitsByCountry(con, country);
        break;
    case 2:
        // Query 2: Jump Back In (Most listened songs by a specific user)
        System.out.print("Enter the userID: ");
        int userID = scanner.nextInt();
        jumpBackIn(con, userID);
        break;
    case 3:
        // Query 3: Recommend Songs Based on Listening History and Preferred Artists
        System.out.print("Enter the userID: ");
```

```
    userID = scanner.nextInt();
    recommendSongs(con, userID);
    break;
case 4:
    // Query 4: Discover New Songs
    System.out.print("Enter the userID: ");
    userID = scanner.nextInt();
    discoverNewSongs(con, userID);
    break;
case 5:
    // Query 5: Top Artist by user
    System.out.print("Enter the userID: ");
    userID = scanner.nextInt();
    topArtist(con, userID);
    topSong(con,userID);
    favouriteGenre(con, userID);
    mostActiveListeningHours(con, userID);
    break;

case 6:
    // Query 6: Songs Released Between 2011 and 2015
    discoverSongs2011to2015(con);
    break;
```

```
            case 7:
                // Query 7: Artist Average Tracks
                artistAverageTracks(con);
                break;
            case 8:
                // Query 8: Track Details Starting with Specific Alphabet
                System.out.print("Enter the alphabet to search for track names starting with: ");
                char alphabet = scanner.nextLine().charAt(0);
                trackDetailsByAlphabet(con, alphabet);
                break;
            case 9:
                // Query 9: View Mood Update Queue
                publishtrack(con);
                break;
            default:
                System.out.println("Invalid choice.");
            }
        }
    }
            private static void viewMoodUpdateQueue(Connection con) {
                    // TODO Auto-generated method stub

            }
```

```
        public static void hotHitsByCountry(Connection con, String country) {
            String query = "SELECT t.trackID, t.title, t.genre, COUNT(lh.trackID) AS play_count
" +
                    "FROM listeninghistory lh " +
                    "JOIN user u ON lh.userID = u.userID " +
                    "JOIN track t ON lh.trackID = t.trackID " +
                    "WHERE u.country = ? " +
                    "GROUP BY t.trackID " +
                    "ORDER BY play_count DESC LIMIT 10";
            try (PreparedStatement pstmt = con.prepareStatement(query)) {
                pstmt.setString(1, country);
                try (ResultSet rs = pstmt.executeQuery()) {
                    if (!rs.isBeforeFirst()) {
                        System.out.printf("No data found for country: %s%n", country);
                        return;
                    }
                    System.out.printf("%-10s %-30s %-20s %-15s%n", "Track ID", "Title", "Genre",
"Play Count");
                    System.out.println("----------------------------------------------------------------------
--------");
                    while (rs.next()) {
                        System.out.printf("%-10d %-30s %-20s %-15d%n",
                                rs.getInt("trackID"),
                                rs.getString("title"),
                                rs.getString("genre"),
```

```
                    rs.getInt("play_count"));
                }
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    // Query 2: Jump Back In (Most listened songs by a specific user)
    public static void jumpBackIn(Connection con, int userID) {
        String query = "SELECT t.trackID, t.title, t.genre, COUNT(lh.trackID) AS play_count
" +
                       "FROM listeninghistory lh " +
                       "JOIN track t ON lh.trackID = t.trackID " +
                       "WHERE lh.userID = ? " +
                       "GROUP BY t.trackID " +
                       "ORDER BY play_count DESC LIMIT 10";
        try (PreparedStatement pstmt = con.prepareStatement(query)) {
            pstmt.setInt(1, userID);
            try (ResultSet rs = pstmt.executeQuery()) {
                if (!rs.isBeforeFirst()) {
                    System.out.printf("No data found for user ID: %d%n", userID);
                    return;
                }
```

```
                System.out.printf("%-10s %-30s %-20s %-15s%n", "Track ID", "Title", "Genre",
"Play Count");
                System.out.println("----------------------------------------------------------------------------
--------");
                while (rs.next()) {
                    System.out.printf("%-10d %-30s %-20s %-15d%n",
                            rs.getInt("trackID"),
                            rs.getString("title"),
                            rs.getString("genre"),
                            rs.getInt("play_count"));
                }
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    // Query 3: Recommend Songs Based on Listening History and Preferred Artists
    public static void recommendSongs(Connection con, int userID) {
        String query = "SELECT t.trackID, t.title, t.genre, a.name AS artist_name " +
                "FROM track t " +
                "JOIN album al ON t.albumID = al.albumID " +
                "JOIN artist a ON al.artistID = a.artistID " +
                "JOIN userfollowsartist ufa ON ufa.artistID = a.artistID " +
```

```
                    "LEFT JOIN listeninghistory lh ON lh.trackID = t.trackID AND lh.userID =
? " +
                    "WHERE ufa.userID = ? AND lh.trackID IS NULL " +
                    "GROUP BY t.trackID ORDER BY t.genre, t.title LIMIT 10";
            try (PreparedStatement pstmt = con.prepareStatement(query)) {
                pstmt.setInt(1, userID);
                pstmt.setInt(2, userID);
                try (ResultSet rs = pstmt.executeQuery()) {
                    if (!rs.isBeforeFirst()) {
                        System.out.printf("No recommended songs found for user ID: %d%n",
userID);
                        return;
                    }
                    System.out.printf("%-10s %-30s %-20s %-30s%n", "Track ID", "Title", "Genre",
"Artist Name");
                    System.out.println("----------------------------------------------------------------------
----------");
                    while (rs.next()) {
                        System.out.printf("%-10d %-30s %-20s %-30s%n",
                                rs.getInt("trackID"),
                                rs.getString("title"),
                                rs.getString("genre"),
                                rs.getString("artist_name"));
                    }
                }
```

```
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
    // Query 4: Discover New Songs
        public static void discoverNewSongs(Connection con, int userID) {
            String query = "SELECT t.trackID, t.title, t.genre, a.name AS artist_name " +
                           "FROM track t " +
                           "JOIN album al ON t.albumID = al.albumID " +
                           "JOIN artist a ON al.artistID = a.artistID " +
                           "LEFT JOIN userfollowsartist ufa ON ufa.artistID = a.artistID AND
ufa.userID = ? " +
                           "LEFT JOIN listeninghistory lh ON lh.trackID = t.trackID AND lh.userID =
? " +
                           "WHERE ufa.artistID IS NULL AND lh.trackID IS NULL " +
                           "GROUP BY t.trackID ORDER BY RAND() LIMIT 10";
            try (PreparedStatement pstmt = con.prepareStatement(query)) {
                pstmt.setInt(1, userID);
                pstmt.setInt(2, userID);
                try (ResultSet rs = pstmt.executeQuery()) {
                    // Check if there are any results
                    if (!rs.isBeforeFirst()) {
                        System.out.printf("No new songs found for user ID: %d%n", userID);
                        return;
```

```
                }

                // Print table headers
                System.out.printf("%-10s %-30s %-20s %-30s%n", "Track ID", "Title", "Genre", "Artist Name");
                System.out.println("------------------------------------------------------------------------------");

                // Print each result row in formatted table style
                while (rs.next()) {
                    System.out.printf("%-10d %-30s %-20s %-30s%n",
                            rs.getInt("trackID"),
                            rs.getString("title"),
                            rs.getString("genre"),
                            rs.getString("artist_name"));
                }
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    // Query 5: Top Artist by user
    public static void topArtist(Connection con, int userID) {
```

```
        String query = "SELECT a.artistID AS artist_id, a.name AS artist_name,
COUNT(lh.trackID) AS play_count " +
                       "FROM listeninghistory lh " +
                       "JOIN track t ON lh.trackID = t.trackID " +
                       "JOIN album al ON t.albumID = al.albumID " +
                       "JOIN artist a ON al.artistID = a.artistID " +
                       "WHERE lh.userID = ? " +
                       "GROUP BY a.artistID " +
                       "ORDER BY play_count DESC " +
                       "LIMIT 1";
       try (PreparedStatement pstmt = con.prepareStatement(query)) {
           pstmt.setInt(1, userID);  // Setting userID dynamically
           try (ResultSet rs = pstmt.executeQuery()) {
               if (rs.next()) {
                       System.out.printf("%-12s %-20s %-12s%n", "Top ArtistID", "Top Artist", "Play
Count");
                       System.out.println("--------------------------------------------");
                       System.out.printf("%-12d %-20s %-12d%n", rs.getInt("artist_id"),
rs.getString("artist_name"), rs.getInt("play_count"));
                       System.out.println("--------------------------------------------");
                       System.out.println();
                       System.out.println();
               }else {
                  System.out.println("No top artist found for user ID: " + userID);
               }
```

```
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    // Query 6: Top Song
    public static void topSong(Connection con, int userID) {
        String query = "SELECT t.trackID, t.title, COUNT(lh.trackID) AS play_count " +
                       "FROM listeninghistory lh " +
                       "JOIN track t ON lh.trackID = t.trackID " +
                       "WHERE lh.userID = ? " +
                       "GROUP BY t.trackID " +
                       "ORDER BY play_count DESC " +
                       "LIMIT 1";
        try (PreparedStatement pstmt = con.prepareStatement(query)) {
            pstmt.setInt(1, userID);  // Setting userID dynamically
            try (ResultSet rs = pstmt.executeQuery()) {
                if (rs.next()) {
                    System.out.printf("%-12s %-20s %-12s%n", "Top Song ID", "Title", "Play
Count");
                    System.out.println("-----------------------------------------");
                    System.out.printf("%-12d %-20s %-12d%n", rs.getInt("trackID"),
rs.getString("title"), rs.getInt("play_count"));
```

```
                System.out.println("-----------------------------------------");
                System.out.println();
                System.out.println();
            } else {
                System.out.println("No top song found for user ID: " + userID);
            }
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
}

// Query 7: Favourite Genre
public static void favouriteGenre(Connection con, int userID) {
    String query = "SELECT t.genre, COUNT(lh.trackID) AS play_count " +
                   "FROM listeninghistory lh " +
                   "JOIN track t ON lh.trackID = t.trackID " +
                   "WHERE lh.userID = ? " +
                   "GROUP BY t.genre ORDER BY play_count DESC LIMIT 1";
    try (PreparedStatement pstmt = con.prepareStatement(query)) {
        pstmt.setInt(1, userID);
        try (ResultSet rs = pstmt.executeQuery()) {
            if (!rs.isBeforeFirst()) {
```

```
                System.out.printf("No favorite genre data found for user ID: %d%n", userID);
                return;
            }
            System.out.printf("%-20s %-10s%n", "Genre", "Play Count");
            System.out.println("------------------------------");
            if (rs.next()) {
                System.out.printf("%-20s %-10d%n",
                        rs.getString("genre"),
                        rs.getInt("play_count"));
                System.out.println("------------------------------");
                System.out.println();
                    System.out.println();
            }
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
}
// Query 8: Most Active Listening Hours
public static void mostActiveListeningHours(Connection con, int userID) {
    String query = "SELECT HOUR(lh.listenDate) AS listen_hour, COUNT(lh.trackID) AS
listen_count " +
                "FROM listeninghistory lh " +
                "WHERE lh.userID = ? " +
```

```
                "GROUP BY listen_hour " +
                "ORDER BY listen_count DESC " +
                "LIMIT 1";
        try (PreparedStatement pstmt = con.prepareStatement(query)) {
            pstmt.setInt(1, userID);  // Setting userID dynamically
            try (ResultSet rs = pstmt.executeQuery()) {
                if (rs.next()) {
                    System.out.printf("%-6s %-12s%n", "Hour", "Listen Count");
                    System.out.println("----------------");
                    System.out.printf("%-6d %-12d%n", rs.getInt("listen_hour"),
rs.getInt("listen_count"));
                    System.out.println("----------------");
                    System.out.println();
                    System.out.println();
                } else {
                    System.out.println("No listening data found for user ID: " + userID);
                }
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
    // Query 9: Songs Released Between 2011 and 2015
    public static void discoverSongs2011to2015(Connection con) {
```

```
        String query = "SELECT t.trackID, t.title, t.genre, a.releaseDate " +
                       "FROM track t " +
                       "JOIN album a ON t.albumID = a.albumID " +
                       "WHERE a.releaseDate BETWEEN '2011-01-01' AND '2015-12-31' " +
                       "ORDER BY RAND() LIMIT 10";
        try (Statement stmt = con.createStatement(); ResultSet rs = stmt.executeQuery(query)) {
            // Print table headers
            System.out.printf("%-10s %-30s %-20s %-20s%n", "Track ID", "Title", "Genre",
"Release Date");
            System.out.println("----------------------------------------------------------------------------------
-");

            // Print each result row in formatted table style
            while (rs.next()) {
                System.out.printf("%-10d %-30s %-20s %-20s%n",
                                  rs.getInt("trackID"),
                                  rs.getString("title"),
                                  rs.getString("genre"),
                                  rs.getDate("releaseDate"));
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
```

```
    // Query 10: Artist Average Tracks
    public static void artistAverageTracks(Connection con) {
        String query = "SELECT artistID, artist_name, average_tracks_per_album " +
                       "FROM ArtistAverageTracks " +
                       "ORDER BY average_tracks_per_album DESC LIMIT 10";
        try (Statement stmt = con.createStatement(); ResultSet rs = stmt.executeQuery(query)) {
            // Print table headers
            System.out.printf("%-10s %-30s %-30s%n", "Artist ID", "Artist Name", "Avg Tracks per
Album");
            System.out.println("---------------------------------------------------------------------");

            // Print each result row in formatted table style
            while (rs.next()) {
                System.out.printf("%-10s %-30s %-30.2f%n",
                                  rs.getString("artistID"),
                                  rs.getString("artist_name"),
                                  rs.getDouble("average_tracks_per_album"));
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
 // Query 11: Find the track by starting alphabet
```

```
    public static void trackDetailsByAlphabet(Connection con, char alphabet) {
        String query = "SELECT t.trackID, t.title, t.genre, a.name AS artist_name " +
                       "FROM track t " +
                       "JOIN album al ON t.albumID = al.albumID " +
                       "JOIN artist a ON al.artistID = a.artistID " +
                       "WHERE t.title LIKE ? " +
                       "ORDER BY t.title LIMIT 10";
        try (PreparedStatement pstmt = con.prepareStatement(query)) {
            pstmt.setString(1, alphabet + "%");
            try (ResultSet rs = pstmt.executeQuery()) {
                if (!rs.isBeforeFirst()) {
                    System.out.printf("No tracks found starting with the letter '%c'.%n", alphabet);
                    return;
                }
                System.out.printf("%-10s %-30s %-20s %-30s%n", "Track ID", "Title", "Genre",
"Artist Name");
                System.out.println("----------------------------------------------------------------------------
--------------");
                while (rs.next()) {
                    System.out.printf("%-10d %-30s %-20s %-30s%n",
                                      rs.getInt("trackID"),
                                      rs.getString("title"),
                                      rs.getString("genre"),
                                      rs.getString("artist_name"));
```

```
            }
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
}
// Query 12: Publish the track
public static void publishtrack(Connection con) {
    Scanner scanner = new Scanner(System.in);

    // Collecting track details
    System.out.print("Enter Track ID: ");
    int trackID = scanner.nextInt();
    scanner.nextLine();  // Consume newline

    System.out.print("Enter Title: ");
    String title = scanner.nextLine();

    System.out.print("Enter Genre: ");
    String genre = scanner.nextLine();

    System.out.print("Enter Duration (format: HH:MM:SS): ");
    String duration = scanner.nextLine();
```

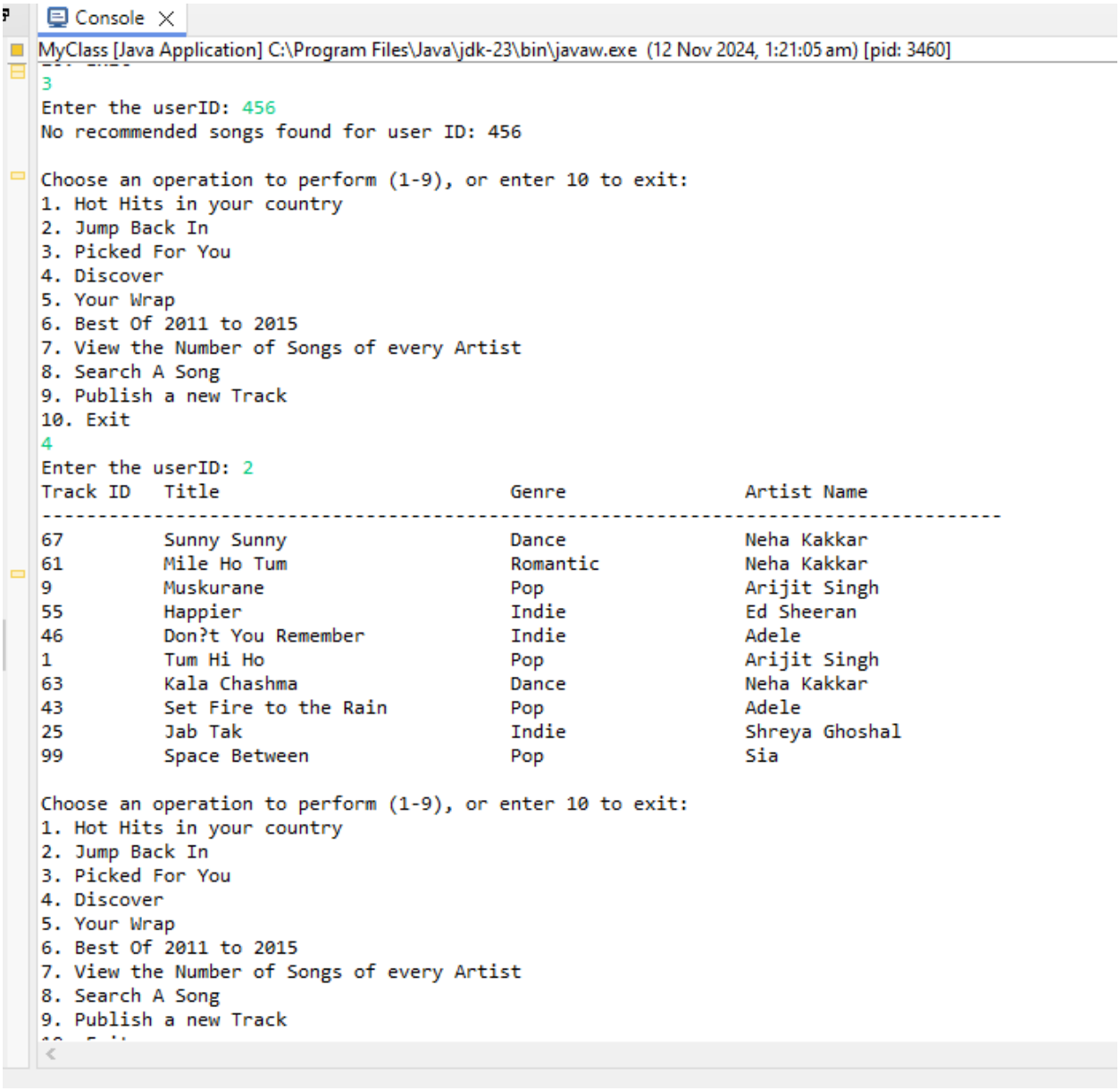
```
        System.out.print("Enter BPM: ");
        int tempo = scanner.nextInt();

        System.out.print("Enter Album ID (or NULL for no album): ");
        Integer albumID = scanner.hasNextInt() ? scanner.nextInt() : null;
        scanner.nextLine();  // Consume newline



        // Insert track into the track table
        String query = "INSERT INTO track (trackID, title, genre, duration, tempo, albumID)
VALUES (?, ?, ?, ?, ?, ?)";

        try (PreparedStatement pstmt = con.prepareStatement(query)) {
            pstmt.setInt(1, trackID);
            pstmt.setString(2, title);
            pstmt.setString(3, genre);
            pstmt.setString(4, duration);
            pstmt.setInt(5, tempo);

            if (albumID != null) {
                pstmt.setInt(6, albumID);
            } else {
                pstmt.setNull(6, java.sql.Types.INTEGER);
```

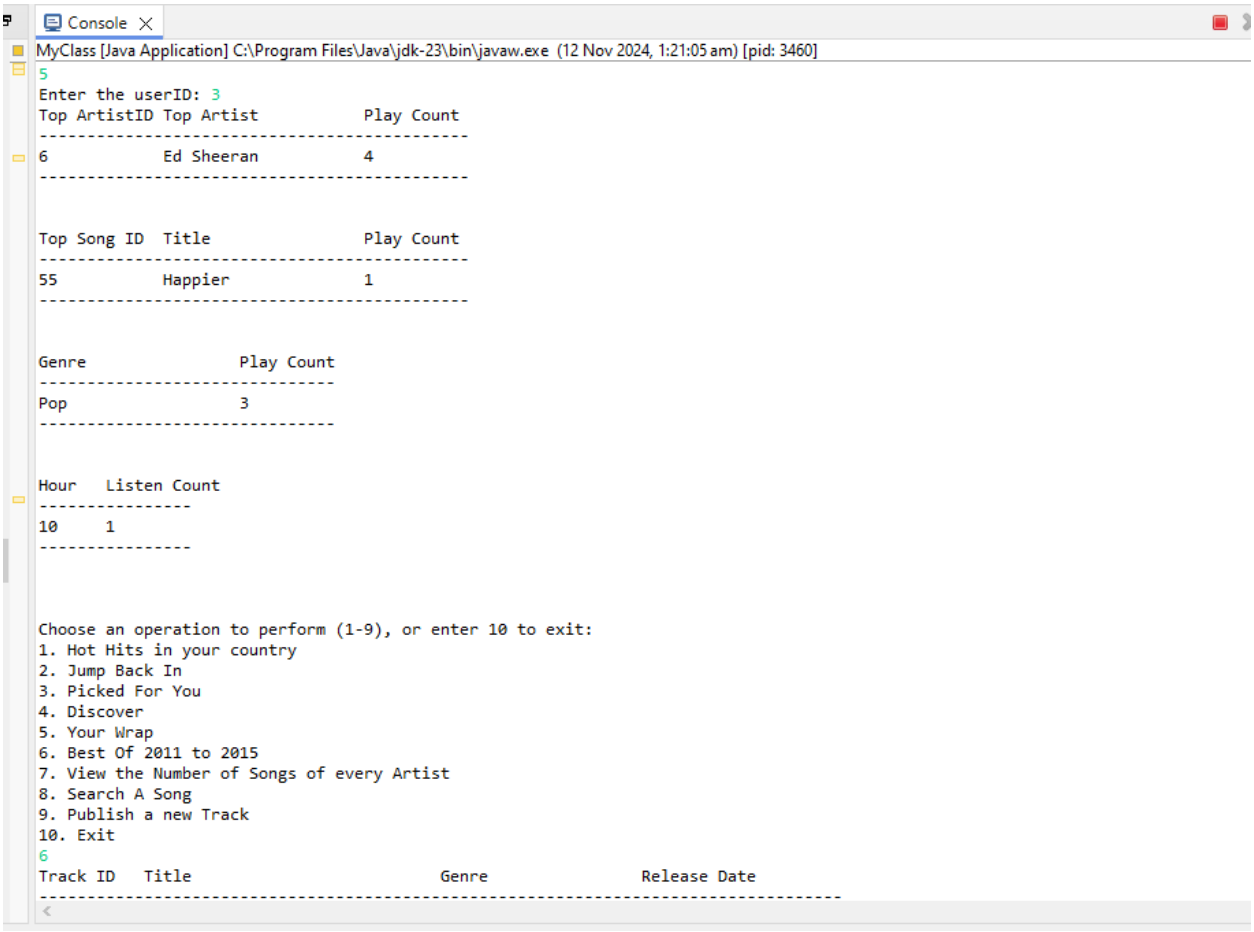
```
            }

            int rowsAffected = pstmt.executeUpdate();
            if (rowsAffected > 0) {
                System.out.println("Track inserted successfully. Trigger will add to
mood_update_queue.");
            } else {
                System.out.println("Track insertion failed.");
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
```

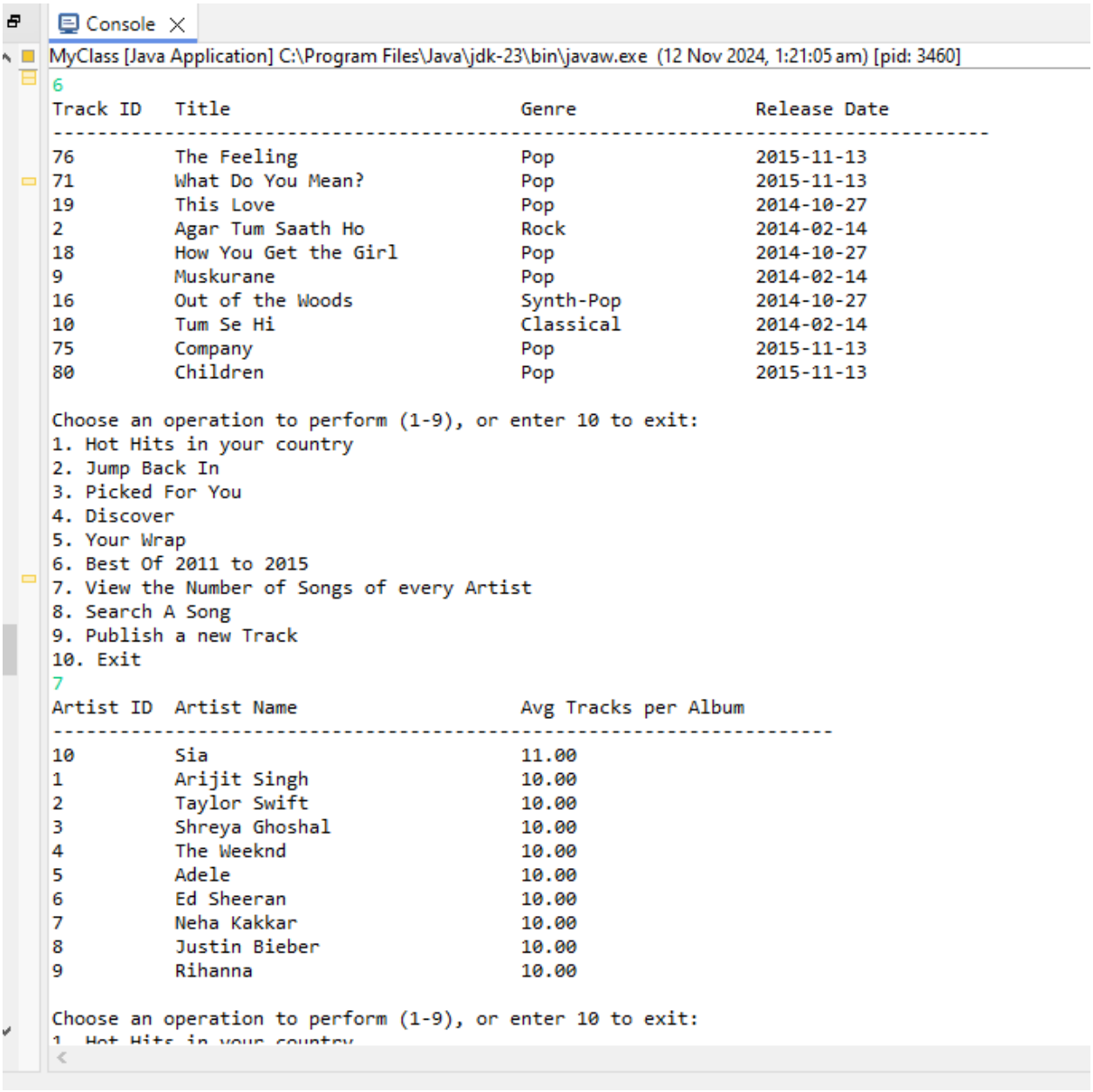
```
Choose an operation to perform (1-9), or enter 10 to exit:
1. Hot Hits in your country
2. Jump Back In
3. Picked For You
4. Discover
5. Your Wrap
6. Best Of 2011 to 2015
7. View the Number of Songs of every Artist
8. Search A Song
9. Publish a new Track
10. Exit
1
Enter the country: india
Track ID   Title                          Genre                Play Count
--------------------------------------------------------------------------------
61         Mile Ho Tum                    Romantic             3
72         Sorry                          Pop                  3
79         No Pressure                    R&B                  2
82         Kiss It Better                 Pop                  2
55         Happier                        Indie                2
57         Dive                           Indie                2
59         What Do I Know?                Pop                  2
76         The Feeling                    Pop                  2
64         Garmi                          Dance                2
75         Company                        Pop                  2

Choose an operation to perform (1-9), or enter 10 to exit:
1. Hot Hits in your country
2. Jump Back In
3. Picked For You
4. Discover
5. Your Wrap
6. Best Of 2011 to 2015
7. View the Number of Songs of every Artist
8. Search A Song
9. Publish a new Track
10. Exit
1
Enter the country: dfg
No data found for country: dfg
```

```
MyClass [Java Application] C:\Program Files\Java\jdk-23\bin\javaw.exe (12 Nov 2024, 1:21:03 am) [pid: 3400]
10. Exit
1
Enter the country: dfg
No data found for country: dfg

Choose an operation to perform (1-9), or enter 10 to exit:
1. Hot Hits in your country
2. Jump Back In
3. Picked For You
4. Discover
5. Your Wrap
6. Best Of 2011 to 2015
7. View the Number of Songs of every Artist
8. Search A Song
9. Publish a new Track
10. Exit
2
Enter the userID: 2
Track ID   Title                          Genre                Play Count
-------------------------------------------------------------------------------
72         Sorry                          Pop                  1
75         Company                        Pop                  1
76         The Feeling                    Pop                  1
77         Purpose                        Pop                  1
79         No Pressure                    R&B                  1
82         Kiss It Better                 Pop                  1
83         Needed Me                      R&B                  1
84         Love on the Brain              Soul                 1
86         Same Ol? Mistakes              R&B                  1
90         Close to You                   Pop                  1

Choose an operation to perform (1-9), or enter 10 to exit:
1. Hot Hits in your country
2. Jump Back In
3. Picked For You
4. Discover
5. Your Wrap
6. Best Of 2011 to 2015
7. View the Number of Songs of every Artist
8. Search A Song
9. Publish a new Track
```

```
MyClass [Java Application] C:\Program Files\Java\jdk-23\bin\javaw.exe  (12 Nov 2024, 1:21:05 am) [pid: 3460]
10. Exit
2
Enter the userID: 345
No data found for user ID: 345

Choose an operation to perform (1-9), or enter 10 to exit:
1. Hot Hits in your country
2. Jump Back In
3. Picked For You
4. Discover
5. Your Wrap
6. Best Of 2011 to 2015
7. View the Number of Songs of every Artist
8. Search A Song
9. Publish a new Track
10. Exit
3
Enter the userID: 2
Track ID   Title                          Genre                Artist Name
-----------------------------------------------------------------------------------------
81         Work                           Dancehall            Rihanna
39         Try Me                         Indie                The Weeknd
15         Wildest Dreams                 Indie                Taylor Swift
17         All You Had to Do Was Stay     Pop                  Taylor Swift
36         Alone Again                    Pop                  The Weeknd
12         Blank Space                    Pop                  Taylor Swift
31         Blinding Lights                Pop                  The Weeknd
85         Desperado                      Pop                  Rihanna
18         How You Get the Girl           Pop                  Taylor Swift
33         Save Your Tears                Pop                  The Weeknd

Choose an operation to perform (1-9), or enter 10 to exit:
1. Hot Hits in your country
2. Jump Back In
3. Picked For You
4. Discover
5. Your Wrap
6. Best Of 2011 to 2015
7. View the Number of Songs of every Artist
8. Search A Song
9. Publish a new Track
```


Console
MyClass [Java Application] C:\Program Files\Java\jdk-23\bin\javaw.exe (12 Nov 2024, 1:21:05 am) [pid: 3460]
3
Enter the userID: 456
No recommended songs found for user ID: 456
Choose an operation to perform (1-9), or enter 10 to exit:
1. Hot Hits in your country
2. Jump Back In
3. Picked For You
4. Discover
5. Your Wrap
6. Best Of 2011 to 2015
7. View the Number of Songs of every Artist
8. Search A Song
9. Publish a new Track
10. Exit
4
Enter the userID: 2
Track ID Title Genre Artist Name
67 Sunny Sunny Dance Neha Kakkar
61 Mile Ho Tum Romantic Neha Kakkar
9 Muskurane Pop Arijit Singh
55 Happier Indie Ed Sheeran
46 Don?t You Remember Indie Adele
1 Tum Hi Ho Pop Arijit Singh
63 Kala Chashma Dance Neha Kakkar
43 Set Fire to the Rain Pop Adele
25 Jab Tak Indie Shreya Ghoshal
99 Space Between Pop Sia
Choose an operation to perform (1-9), or enter 10 to exit:
1. Hot Hits in your country
2. Jump Back In
3. Picked For You
4. Discover
5. Your Wrap
6. Best Of 2011 to 2015
7. View the Number of Songs of every Artist
8. Search A Song
9. Publish a new Track


Console
MyClass [Java Application] C:\Program Files\Java\jdk-23\bin\javaw.exe (12 Nov 2024, 1:21:05 am) [pid: 3460]
5
Enter the userID: 3
Top ArtistID Top Artist Play Count
6 Ed Sheeran 4
Top Song ID Title Play Count
55 Happier 1
Genre Play Count
Pop 3
Hour Listen Count
10 1
Choose an operation to perform (1-9), or enter 10 to exit:
1. Hot Hits in your country
2. Jump Back In
3. Picked For You
4. Discover
5. Your Wrap
6. Best Of 2011 to 2015
7. View the Number of Songs of every Artist
8. Search A Song
9. Publish a new Track
10. Exit
6
Track ID Title Genre Release Date


Console
MyClass [Java Application] C:\Program Files\Java\jdk-23\bin\javaw.exe (12 Nov 2024, 1:21:05 am) [pid: 3460]
6
Track ID Title Genre Release Date
76 The Feeling Pop 2015-11-13
71 What Do You Mean? Pop 2015-11-13
19 This Love Pop 2014-10-27
2 Agar Tum Saath Ho Rock 2014-02-14
18 How You Get the Girl Pop 2014-10-27
9 Muskurane Pop 2014-02-14
16 Out of the Woods Synth-Pop 2014-10-27
10 Tum Se Hi Classical 2014-02-14
75 Company Pop 2015-11-13
80 Children Pop 2015-11-13
Choose an operation to perform (1-9), or enter 10 to exit:
1. Hot Hits in your country
2. Jump Back In
3. Picked For You
4. Discover
5. Your Wrap
6. Best Of 2011 to 2015
7. View the Number of Songs of every Artist
8. Search A Song
9. Publish a new Track
10. Exit
7
Artist ID Artist Name Avg Tracks per Album
10 Sia 11.00
1 Arijit Singh 10.00
2 Taylor Swift 10.00
3 Shreya Ghoshal 10.00
4 The Weeknd 10.00
5 Adele 10.00
6 Ed Sheeran 10.00
7 Neha Kakkar 10.00
8 Justin Bieber 10.00
9 Rihanna 10.00
Choose an operation to perform (1-9), or enter 10 to exit:

```
MyClass [Java Application] C:\Program Files\Java\jdk-23\bin\javaw.exe  (12 Nov 2024, 1:21:05 am) [pid: 3460]
Choose an operation to perform (1-9), or enter 10 to exit:
1. Hot Hits in your country
2. Jump Back In
3. Picked For You
4. Discover
5. Your Wrap
6. Best Of 2011 to 2015
7. View the Number of Songs of every Artist
8. Search A Song
9. Publish a new Track
10. Exit
8
Enter the alphabet to search for track names starting with: s
Track ID   Title                          Genre                Artist Name
-------------------------------------------------------------------------------------------
27         Saans                          Romantic             Shreya Ghoshal
86         Same Ol? Mistakes              R&B                  Rihanna
33         Save Your Tears                Pop                  The Weeknd
38         Scared to Live                 Soul                 The Weeknd
43         Set Fire to the Rain           Pop                  Adele
11         Shake It Off                   Pop                  Taylor Swift
51         Shape of You                   Pop                  Ed Sheeran
58         Shape of You (Reprise)         Soul                 Ed Sheeran
42         Someone Like You               Soul                 Adele
72         Sorry                          Pop                  Justin Bieber

Choose an operation to perform (1-9), or enter 10 to exit:
1. Hot Hits in your country
2. Jump Back In
3. Picked For You
4. Discover
5. Your Wrap
6. Best Of 2011 to 2015
7. View the Number of Songs of every Artist
8. Search A Song
9. Publish a new Track
10. Exit
9
Enter Track ID: 102
Enter Title: ahc
```

```
10. Exit
9
Enter Track ID: 102
Enter Title: ahc
Enter Genre: pop
Enter Duration (format: HH:MM:SS): 00:03:02
Enter BPM: 80
Enter Album ID (or NULL for no album): 10
Track inserted successfully. Trigger will add to mood_update_queue.

Choose an operation to perform (1-9), or enter 10 to exit:
1. Hot Hits in your country
2. Jump Back In
3. Picked For You
4. Discover
5. Your Wrap
6. Best Of 2011 to 2015
7. View the Number of Songs of every Artist
8. Search A Song
9. Publish a new Track
10. Exit
```

```
mysql> select * from mood_update_queue;
+---------+
| trackID |
+---------+
|     102 |
+---------+
1 row in set (0.00 sec)

mysql> select * from mood_update_queue;
Empty set (0.00 sec)
```

```
|      97 | Move Your Body             | Dance      | 00:03:33 |   122 | energetic |       10 |
|      98 | Reaper                     | Pop        | 00:04:10 |   122 | energetic |       10 |
|      99 | Space Between              | Pop        | 00:04:00 |   108 | happy     |       10 |
|     100 | Dressed in Black           | Pop        | 00:04:23 |   118 | happy     |       10 |
|     101 | aaa                        | pop        | 00:03:00 |    80 | happy     |       10 |
|     102 | ahc                        | pop        | 00:03:02 |    80 | happy     |       10 |
+---------+----------------------------+------------+----------+-------+-----------+----------+
102 rows in set (0.00 sec)
```