Portland State
Computer Science

PROJECT

By

Mounika Javvaji

(956734921)

Velgonda Laasya

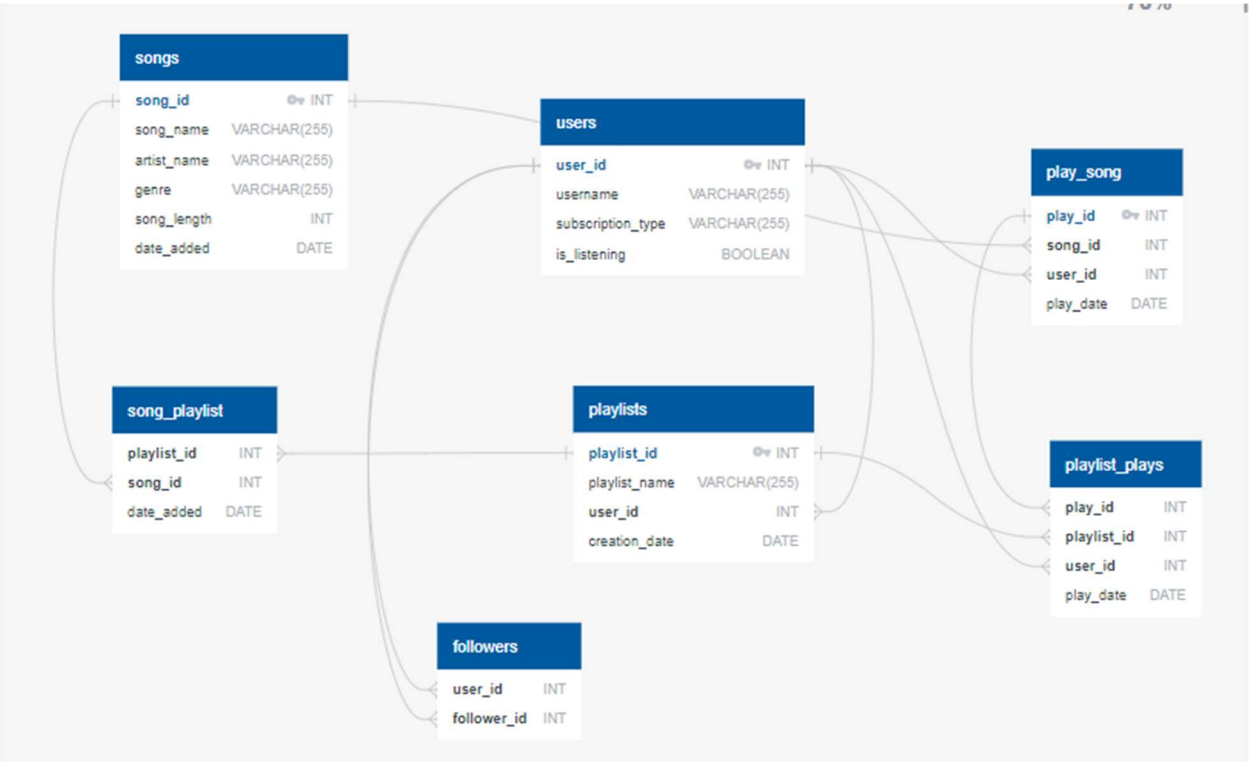
(957262603)

FINAL WRITE-UP

1. Full references:

https://www.kaggle.com/datasets/mrmorj/dataset-of-songs-in-spotify

2. ER diagram:



3. CREATE TABLE statements:

CREATE A SONGS TABLE

```
CREATE TABLE songs (
 song_id INT PRIMARY KEY,
 song_name VARCHAR(255),
 artist_name VARCHAR(255),
 genre VARCHAR(255),
 song_length INT,
 date_added DATE
);
```

CREATE A USERS TABLE

```
CREATE TABLE users (
 user_id INT PRIMARY KEY,
 username VARCHAR(255),
 subscription_type VARCHAR(255),
 is_listening BOOLEAN
);
```

CREATE A PLAYLISTS TABLE

```
CREATE TABLE playlists (
 playlist_id INT PRIMARY KEY,
 playlist_name VARCHAR(255),
 user_id INT,
 creation_date DATE,
 FOREIGN KEY (user_id) REFERENCES users(user_id)
```

```
);
```

CREATE A SONG_PLAYLIST TABLE

```
CREATE TABLE song_playlist (
 playlist_id INT,
 song_id INT,
 date_added DATE,
 FOREIGN KEY (playlist_id) REFERENCES playlists(playlist_id),
 FOREIGN KEY (song_id) REFERENCES songs(song_id)
);
```

CREATE A PLAY_SONG TABLE

```
CREATE TABLE play_song (
 play_id INT PRIMARY KEY,
 song_id INT,
 user_id INT,
 play_date DATE,
 FOREIGN KEY (song_id) REFERENCES songs(song_id),
 FOREIGN KEY (user_id) REFERENCES users(user_id)
);
```

CREATE THE FOLLOWERS TABLE

```
CREATE TABLE followers (
 user_id INT,
 follower_id INT,
 FOREIGN KEY (user_id) REFERENCES users(user_id),
 FOREIGN KEY (follower_id) REFERENCES users(user_id)
);
```

CREATE THE PLAYLIST_PLAYS TABLE

```
CREATE TABLE playlist_plays (
 play_id INT,
 playlist_id INT,
 user_id INT,
 play_date DATE,
 FOREIGN KEY (play_id) REFERENCES play_song (play_id),
 FOREIGN KEY (playlist_id) REFERENCES playlists(playlist_id),
 FOREIGN KEY (user_id) REFERENCES users(user_id)
);
```

4. Description:

We got all the CSV datasets from the website.

Each dataset's data was cleansed, and an entirely new set of csv files was obtained.

Here is an example Python program to clean up the raw data:

add pandas as the pd

Open the CSV file by entering df = pd.read csv("filename.csv").

Do not leave any blank spaces at the beginning or end of the column names. df.columns equals df.columns.str.strip ()

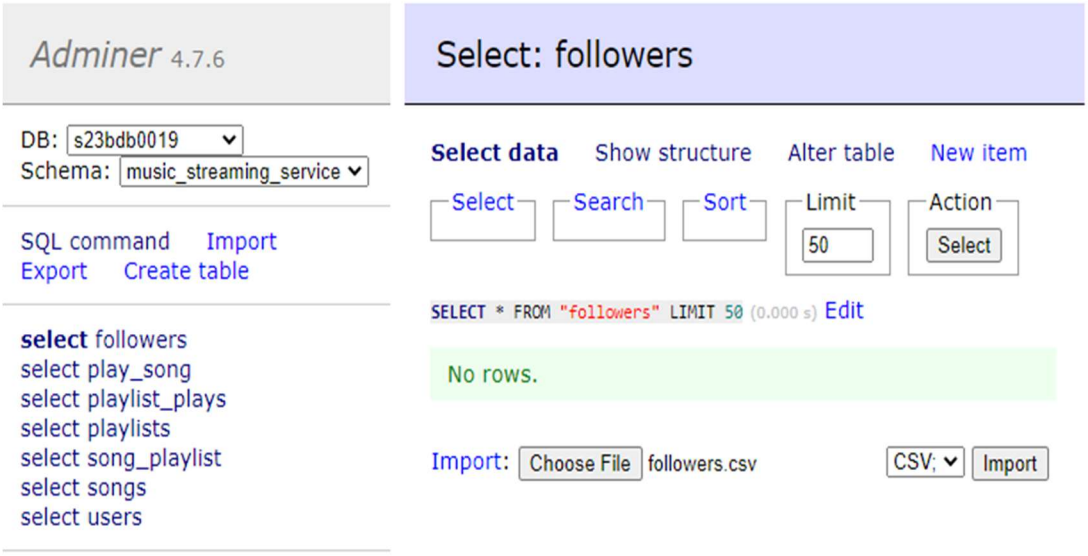
Remove any rows with blank values. df.dropna(inplace=True)

Change the necessary type for the attribute column. "attribute-name" = "attribute name," etc. astype(type)

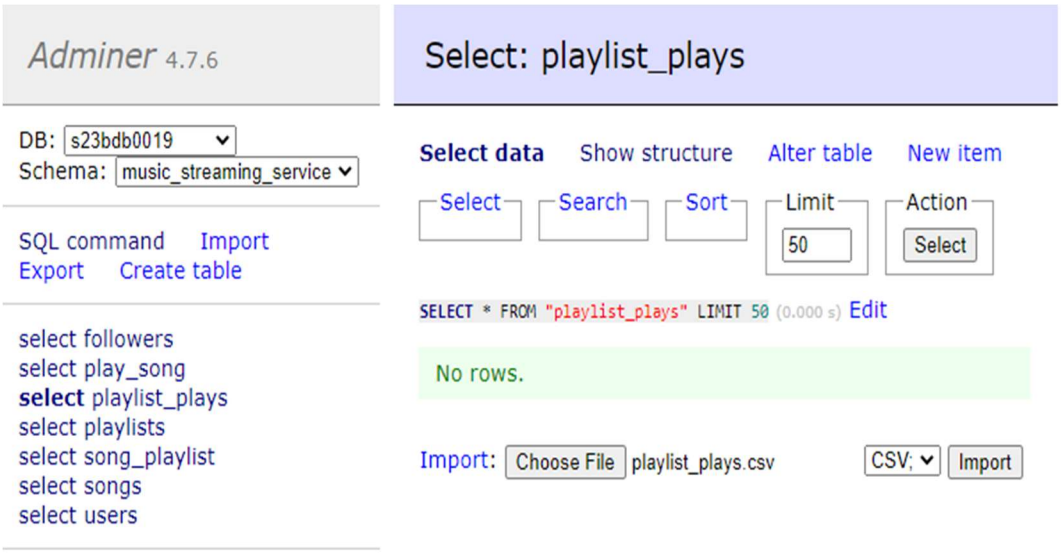
After dropping rows, reset the index. (drop=True, inplace=True) df.reset index

Use the command df.to csv("new filename.csv", index=False) to save the cleaned data to a new CSV file.

The necessary csv files were imported after all the tables had been created.




Adminer 4.7.6
DB: s23bdb0019
Schema: music_streaming_service
SQL command Import
Export Create table
select followers
select play_song
select playlist_plays
select playlists
select song_playlist
select songs
select users
Select: play_song
Select data Show structure Alter table New item
Select Search Sort Limit Action
50
Select
SELECT * FROM "play_song" LIMIT 50 (0.001 s) Edit
No rows.
Import: Choose File play_song.csv
CSV; Import


Adminer 4.7.6
DB: s23bdb0019
Schema: music_streaming_service
SQL command Import
Export Create table
select followers
select play_song
select playlist_plays
select playlists
select song_playlist
select songs
select users
Select: playlist_plays
Select data Show structure Alter table New item
Select Search Sort Limit Action
50
Select
SELECT * FROM "playlist_plays" LIMIT 50 (0.000 s) Edit
No rows.
Import: Choose File playlist_plays.csv
CSV; Import

Adminer 4.7.6

DB: s23bdb0019
Schema: music_streaming_service

SQL command Import
Export Create table

select followers
select play_song
select playlist_plays
select playlists
select song_playlist
select songs
select users

Select: playlists

Select data Show structure Alter table New item

Select | Search | Sort | Limit: 50 | Text length: 100 | Action: Select

```
SELECT * FROM "playlists" LIMIT 50 (0.000 s) Edit
```

No rows.

Import: Choose File playlists.csv CSV; Import

Adminer 4.7.6

DB: s23bdb0019
Schema: music_streaming_service

SQL command Import
Export Create table

select followers
select play_song
select playlist_plays
select playlists
select song_playlist
select songs
select users

Select: song_playlist

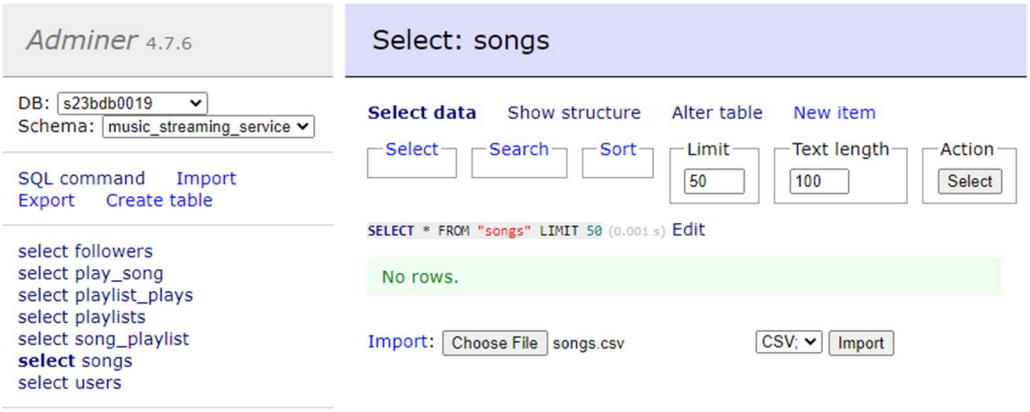
Select data Show structure Alter table New item

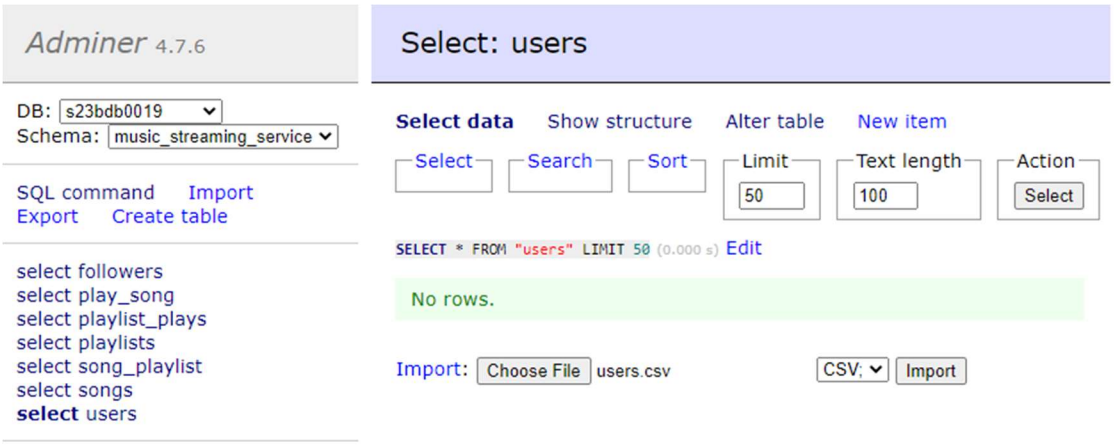
Select | Search | Sort | Limit: 50 | Action: Select

```
SELECT * FROM "song_playlist" LIMIT 50 (0.000 s) Edit
```

No rows.

Import: Choose File song_playlist.csv CSV; Import





5. Queries:

We have changed queries for 1,6,10,15,17,18,19 and 20 from the ones we previously submitted. We realized that this was not how we tested ourselves and that we could not complicate a straightforward inquiry by using many notions. Hence, we made the decision to edit the questions and replace them with ones that covered the principles we had learned in class.

1)

List the usernames of users who have created playlists but have never played any song.

```
SELECT u.username FROM users u LEFT JOIN playlists p ON u.user_id = p.user_id LEFT JOIN playlist_plays pp ON p.playlist_id = pp.playlist_id WHERE pp.play_id IS NULL;
```

```
SELECT u.username FROM users u LEFT JOIN playlists p ON u.user_id = p.user_id LEFT JOIN playlist_plays pp ON p.playlist_id = pp.playlist_id WHERE pp.play_id IS NUL
```

username
PopQueen
MelodySeeker
PopQueen
JaneSmith
JohnDoe
RockFan
RhythmMaster
CountryBoy

8 rows (0.003 s) Edit, Explain, Export

```
SELECT u.username FROM users u LEFT JOIN playlists p ON u.user_id = p.user_id LEFT JOIN playlist_plays pp ON p.playlist_id = pp.playlist_id WHERE pp.play_id IS
```

---

2) What is the most popular genre of music?

```
SELECT genre, COUNT(*) AS count FROM songs GROUP BY genre ORDER BY count DESC LIMIT 1;
```

```
SELECT genre, COUNT(*) AS count FROM songs GROUP BY genre ORDER BY count DESC LIMIT 1
```

genre	count
Pop	6

1 row (0.003 s) Edit, Explain, Export

```
SELECT genre, COUNT(*) AS count FROM songs GROUP BY genre ORDER BY count DESC LIMIT
```

---

3) How many users are currently listening to music?

```
SELECT COUNT(*) FROM users WHERE is_listening = true;
```

```
SELECT COUNT(*) FROM users WHERE is_listening = true
```

count
6

1 row (0.001 s) Edit, Explain, Export

```
SELECT COUNT(*) FROM users WHERE is_listening = true;
```

---

4) How many playlists does each user have?

```
SELECT user_id, COUNT(*) AS count FROM playlists GROUP BY user_id;
```

```
SELECT user_id, COUNT(*) AS count FROM playlists GROUP BY user_id
```

user_id	count
9	1
3	1
5	1
4	1
10	1
6	1
2	1
7	1
1	1
8	2

10 rows (0.002 s) Edit, Explain, Export

```
SELECT user_id, COUNT(*) AS count FROM playlists GROUP BY user_id;
```

---

5) What are the most popular songs of all time?

```
SELECT s.song_id, s.song_name, s.artist_name, COUNT(ps.play_id) AS play_count
FROM songs s
JOIN play_song ps ON s.song_id = ps.song_id
GROUP BY s.song_id, s.song_name, s.artist_name
ORDER BY play_count DESC
LIMIT 10;
```

```
SELECT s.song_id, s.song_name, s.artist_name, COUNT(ps.play_id) AS play_count
FROM songs s
JOIN play_song ps ON s.song_id = ps.song_id
GROUP BY s.song_id, s.song_name, s.artist_name
ORDER BY play_count DESC
LIMIT 10
```

song_id	song_name	artist_name	play_count
7	Uptown Funk	Mark Ronson ft. Bruno Mars	2
10	Rolling in the Deep	Adele	2
6	Despacito	Luis Fonsi	1
3	Show me the thumka	Sunidhi Chauhan	1
5	Shape of You	Ed Sheeran	1
2	Wildest Dreams	Taylor Swift	1
8	Thinking Out Loud	Ed Sheeran	1
11	Bad Guy	Billie Eilish	1
9	Hotline Bling	Drake	1

9 rows (0.004 s) Edit, Explain, Export

```
SELECT s.song_id, s.song_name, s.artist_name, COUNT(ps.play_id) AS play_count
FROM songs s
JOIN play_song ps ON s.song_id = ps.song_id
GROUP BY s.song_id, s.song_name, s.artist_name
ORDER BY play_count DESC
LIMIT 10;
```

---

6)

Find the most recent play date for each user.

```
SELECT user_id, MAX(play_date) AS recent_play_date FROM play_song GROUP BY user_id;
```

```
SELECT user_id, MAX(play_date) AS recent_play_date FROM play_song GROUP BY user_id
```

user_id	recent_play_date
3	2022-10-20
5	2023-02-28
4	2023-05-02
6	2023-04-10
2	2022-12-12
1	2023-03-15

6 rows (0.002 s) Edit, Explain, Export

```
SELECT user_id, MAX(play_date) AS recent_play_date FROM play_song GROUP BY user_id;
```

7) What is the average length of a song in the database?

```
SELECT AVG(song_length) FROM songs;
```

```
SELECT AVG(song_length) FROM songs
```

avg
3.5454545454545455

1 row (0.001 s) Edit, Explain, Export

```
SELECT AVG(song_length) FROM songs;
```

8) How many songs were added to the database in the last month?

```
SELECT COUNT(*) FROM songs WHERE date_added >= CURRENT_DATE - INTERVAL '1 MONTH';
```

```
SELECT COUNT(*) FROM songs WHERE date_added >= CURRENT_DATE - INTERVAL '1 MONTH'
```

count
1

1 row (0.002 s) Edit, Explain, Export

```
SELECT COUNT(*) FROM songs WHERE date_added >= CURRENT_DATE - INTERVAL '1 MONTH';
```

9) Which artists have the most songs in the database?

```
SELECT artist_name, COUNT(*) AS count FROM songs GROUP BY artist_name ORDER BY count DESC LIMIT 10;
```

```
SELECT artist_name, COUNT(*) AS count FROM songs GROUP BY artist_name ORDER BY count DESC LIMIT 10
```

artist_name	count
Ed Sheeran	2
Sunidhi Chauhan	1
Sid Sriram	1
Thulipili Bharathi	1
Drake	1
Billie Eilish	1
Adele	1
Luis Fonsi	1
Taylor Swift	1
Mark Ronson ft. Bruno Mars	1

10 rows (0.001 s) Edit, Explain, Export

```
SELECT artist_name, COUNT(*) AS count FROM songs GROUP BY artist_name ORDER BY count DESC LIM
```

---

10)

Retrieve the usernames of users who have played at least one song in all the playlists they follow.

```
SELECT u.username FROM users u JOIN playlist_plays pp ON u.user_id = pp.user_id JOIN playlists p ON pp.playlist_id = p.playlist_id JOIN followers f ON p.user_id = f.user_id WHERE u.user_id = f.follower_id GROUP BY u.user_id HAVING COUNT(DISTINCT pp.playlist_id) = (SELECT COUNT(*) FROM playlists WHERE user_id = u.user_id);
```

```
SELECT u.username FROM users u JOIN playlist_plays pp ON u.user_id = pp.user_id JOIN playlists p ON pp.playlist_id = p.playlist_id JOIN followers f ON p.user_id =
```

username
Mounika

1 row (0.005 s) Edit, Explain, Export

```
SELECT u.username FROM users u JOIN playlist_plays pp ON u.user_id = pp.user_id JOIN playlists p ON pp.playlist_id = p.playlist_id JOIN followers f ON p.user_id =
```

---

11) Which users have the most followers?

```
SELECT user_id, COUNT(*) AS count FROM followers GROUP BY user_id ORDER BY count DESC LIMIT 10;
```

```
SELECT user_id, COUNT(*) AS count FROM followers GROUP BY user_id ORDER BY count DESC LIMIT 10
```

user_id	count
3	2
6	2
2	2
1	2
8	1
5	1
4	1
7	1

8 rows (0.002 s) Edit, Explain, Export

```
SELECT user_id, COUNT(*) AS count FROM followers GROUP BY user_id ORDER BY count DESC LIMI
```

12) How many playlists are created per day?

```
SELECT TO_CHAR(creation_date, 'YYYY-MM-DD') AS date, COUNT(*) AS count
FROM playlists
GROUP BY date;
```

```
SELECT TO_CHAR(creation_date, 'YYYY-MM-DD') AS date, COUNT(*) AS count
FROM playlists
GROUP BY date
```

date	count
2021-08-09	1
2020-09-13	1
2021-09-30	1
2018-05-31	1
2022-03-17	1
2023-01-10	1
2019-02-28	1
2021-11-05	1
2023-05-01	1
2020-03-12	1
2022-07-20	1

11 rows (0.002 s) Edit, Explain, Export

```
SELECT TO_CHAR(creation_date, 'YYYY-MM-DD') AS date, COUNT(*) AS count
FROM playlists
GROUP BY date;
```

13) What are the most popular playlists of all time?

```
SELECT playlists.playlist_id, playlists.playlist_name, COUNT(playlist_plays.play_id) AS play_count
FROM playlists
JOIN playlist_plays ON playlists.playlist_id = playlist_plays.playlist_id
GROUP BY playlists.playlist_id, playlists.playlist_name
ORDER BY play_count DESC;
```

```
SELECT playlists.playlist_id, playlists.playlist_name, COUNT(playlist_plays.play_id) AS play_count
FROM playlists
JOIN playlist_plays ON playlists.playlist_id = playlist_plays.playlist_id
GROUP BY playlists.playlist_id, playlists.playlist_name
ORDER BY play_count DESC
```

playlist_id	playlist_name	play_count
1	party	4
2	Blues	3
3	Gym	2
4	Road Trip	2

4 rows (0.003 s) Edit, Explain, Export

```
SELECT playlists.playlist_id, playlists.playlist_name, COUNT(playlist_plays.play_id) AS play_c
FROM playlists
JOIN playlist_plays ON playlists.playlist_id = playlist_plays.playlist_id
GROUP BY playlists.playlist_id, playlists.playlist_name
ORDER BY play_count DESC;
```

14) How many songs are in each playlist?

```
SELECT playlists.playlist_id, playlists.playlist_name, COUNT(song_playlist.song_id) AS song_count
FROM playlists
LEFT JOIN song_playlist ON playlists.playlist_id = song_playlist.playlist_id
GROUP BY playlists.playlist_id, playlists.playlist_name;
```

```
SELECT playlists.playlist_id, playlists.playlist_name, COUNT(song_playlist.song_id) AS song_count
FROM playlists
LEFT JOIN song_playlist ON playlists.playlist_id = song_playlist.playlist_id
GROUP BY playlists.playlist_id, playlists.playlist_name
```

playlist_id	playlist_name	song_count
11	Acoustic	0
9	Summer Vibes	0
3	Gym	1
5	Chill Out	1
4	Road Trip	1
10	Indie	0
6	Throwback Hits	1
2	Blues	2
7	Workout	1
1	party	3
8	Romantic	1

11 rows (0.002 s) Edit, Explain, Export

```
SELECT playlists.playlist_id, playlists.playlist_name, COUNT(song_playlist.song_id) AS song_c
FROM playlists
LEFT JOIN song_playlist ON playlists.playlist_id = song_playlist.playlist_id
GROUP BY playlists.playlist_id, playlists.playlist_name;
```

15) How many songs have been played in total?

List the playlists that have never been played.

```
SELECT p.playlist_name FROM playlists p LEFT JOIN playlist_plays pp ON p.playlist_id = pp.playlist_id WHERE pp.play_id IS NULL;
```

```
SELECT p.playlist_name FROM playlists p LEFT JOIN playlist_plays pp ON p.playlist_id = pp.playlist_id WHERE pp.play_id IS NULL
```

playlist_name
Acoustic
Indie
Chill Out
Romantic
Throwback Hits
Summer Vibes
Workout

7 rows (0.001 s) Edit, Explain, Export

```
SELECT p.playlist_name FROM playlists p LEFT JOIN playlist_plays pp ON p.playlist_id = pp.playlist_id WHERE pp.play_id IS
```

16) Which songs have been played the most times?

```
SELECT s.song_name, COUNT(ps.play_id) AS play_count
FROM songs s
JOIN play_song ps ON s.song_id = ps.song_id
GROUP BY s.song_name
ORDER BY play_count DESC;
```

```
SELECT s.song_name, COUNT(ps.play_id) AS play_count
FROM songs s
JOIN play_song ps ON s.song_id = ps.song_id
GROUP BY s.song_name
ORDER BY play_count DESC
```

song_name	play_count
Uptown Funk	2
Rolling in the Deep	2
Show me the thumka	1
Bad Guy	1
Wildest Dreams	1
Thinking Out Loud	1
Despacito	1
Shape of You	1
Hotline Bling	1

9 rows (0.003 s) Edit, Explain, Export


```
SELECT s.song_name, COUNT(ps.play_id) AS play_count
FROM songs s
JOIN play_song ps ON s.song_id = ps.song_id
GROUP BY s.song_name
ORDER BY play_count DESC;
```

17)

List the playlists along with the number of songs and the total duration of songs in each playlist.

```
SELECT p.playlist_name, COUNT(sp.song_id) AS song_count, SUM(s.song_length) AS total_duration
FROM playlists p
JOIN song_playlist sp ON p.playlist_id = sp.playlist_id
JOIN songs s ON sp.song_id = s.song_id
GROUP BY p.playlist_id;
```

```
SELECT p.playlist_name, COUNT(sp.song_id) AS song_count, SUM(s.song_length) AS total_duration
FROM playlists p
JOIN song_playlist sp ON p.playlist_id = sp.playlist_id
JOIN songs s ON sp.song_id = s.song_id
GROUP BY p.playlist_id
```

playlist_name	song_count	total_duration
Gym	1	4
Chill Out	1	4
Road Trip	1	3
Throwback Hits	1	3
Blues	2	7
Workout	1	4
party	3	11
Romantic	1	3

8 rows (0.003 s) Edit, Explain, Export

```
SELECT p.playlist_name, COUNT(sp.song_id) AS song_count, SUM(s.song_length) AS total_dura
FROM playlists p
JOIN song_playlist sp ON p.playlist_id = sp.playlist_id
JOIN songs s ON sp.song_id = s.song_id
GROUP BY p.playlist_id;
```

18)

Retrieve the playlists created by users who are not currently listening to music.

```
SELECT p.playlist_name
FROM playlists p
INNER JOIN users u ON p.user_id = u.user_id
WHERE u.is_listening = false;
```

```
SELECT p.playlist_name
FROM playlists p
INNER JOIN users u ON p.user_id = u.user_id
WHERE u.is_listening = false
```

playlist_name
party
Blues
Road Trip
Workout

4 rows (0.002 s) Edit, Explain, Export


```
SELECT p.playlist_name
FROM playlists p
INNER JOIN users u ON p.user_id = u.user_id
WHERE u.is_listening = false;
```

19)

Retrieve the usernames of users who have never created a playlist.

```
SELECT username
FROM users
WHERE user_id NOT IN (SELECT user_id FROM playlists);
```

```
SELECT username
FROM users
WHERE user_id NOT IN (SELECT user_id FROM playlists)
```

username
CountryBoy

1 row (0.001 s) Edit, Explain, Export

```
SELECT username
FROM users
WHERE user_id NOT IN (SELECT user_id FROM playlists);
```

20)

Retrieve the users who have listened to songs from all genres.

```
SELECT u.username
FROM users u
JOIN songs s ON s.genre NOT IN (
  SELECT DISTINCT s.genre
  FROM songs s
  LEFT JOIN play_song ps ON s.song_id = ps.song_id AND ps.user_id = u.user_id
  WHERE ps.song_id IS NULL
);
```

```
SELECT u.username
FROM users u
JOIN songs s ON s.genre NOT IN (
  SELECT DISTINCT s.genre
  FROM songs s
  LEFT JOIN play_song ps ON s.song_id = ps.song_id AND ps.user_id = u.user_id
  WHERE ps.song_id IS NULL
)
```

username
Jaanu
JohnDoe
JohnDoe
JaneSmith

4 rows (0.076 s) Edit, Explain, Export

```
SELECT u.username
FROM users u
JOIN songs s ON s.genre NOT IN (
  SELECT DISTINCT s.genre
  FROM songs s
  LEFT JOIN play_song ps ON s.song_id = ps.song_id AND ps.user_id = u.user_id
  WHERE ps.song_id IS NULL
);
```