SQL PROJECT SPOTIFY

spotfiy sql project

Objectives

This project involves analyzing a Spotify dataset with various attributes about tracks, albums, and artists using SQL. It covers an end-to-end process of normalizing a denormalized dataset, performing SQL queries of varying complexity (easy, medium, and advanced). The primary goals of the project are to practice advanced SQL skills and generate valuable insights from the dataset.

Data SCHEMA

```
CREATE TABLE spotify (
  Artist VARCHAR(255),
  Track VARCHAR(255),
  Album VARCHAR(255),
  Album_type VARCHAR(255),
  Danceability FLOAT,
  Energy FLOAT,
  Loudness FLOAT,
  Speechiness FLOAT,
  Acousticness FLOAT,
  Instrumentalness FLOAT,
  Liveness FLOAT,
  Valence FLOAT,
  Tempo FLOAT,
  Duration_min FLOAT,
  Title VARCHAR(255),
  Channel VARCHAR(255),
  Views FLOAT,
  Likes BIGINT,
  Comments BIGINT,
  Licensed BOOLEAN,
  official_video BOOLEAN,
  Stream BIGINT,
  EnergyLiveness FLOAT,
  most_played_on VARCHAR(50)
);
```

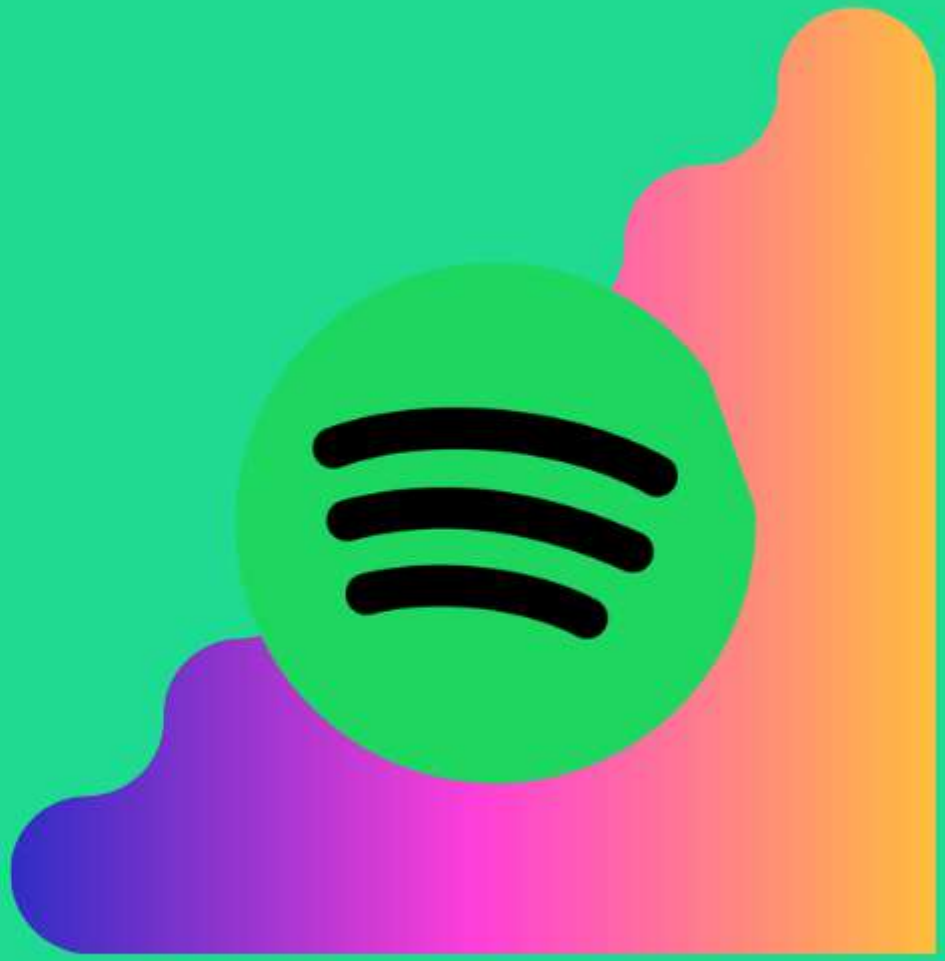
Data Exploration

Before diving into SQL, it's important to understand the dataset thoroughly. The dataset contains attributes such as:

- Artist: The performer of the track.
- Track: The name of the song.
- Album: The album to which the track belongs.
- Album_type: The type of album (e.g., single or album).
- Various metrics such as danceability, energy, loudness, tempo, and more.

Exploratory Data Analysis (EDA)

```
SELECT COUNT(*) FROM spotify;

SELECT COUNT(DISTINCT Artist)
FROM spotify;

SELECT COUNT(DISTINCT Album)
FROM spotify;

SELECT DISTINCT Album_type
FROM spotify;

SELECT COUNT(DISTINCT Title)
FROM spotify;

SELECT DISTINCT channel
FROM spotify;

SELECT DISTINCT most_played_on
FROM spotify;

SELECT MAX(duration_min)
FROM spotify;

SELECT MIN(duration_min)
FROM spotify;

SELECT *
FROM spotify
WHERE duration_min =0;

DELETE FROM spotify
WHERE duration_min =0;
```



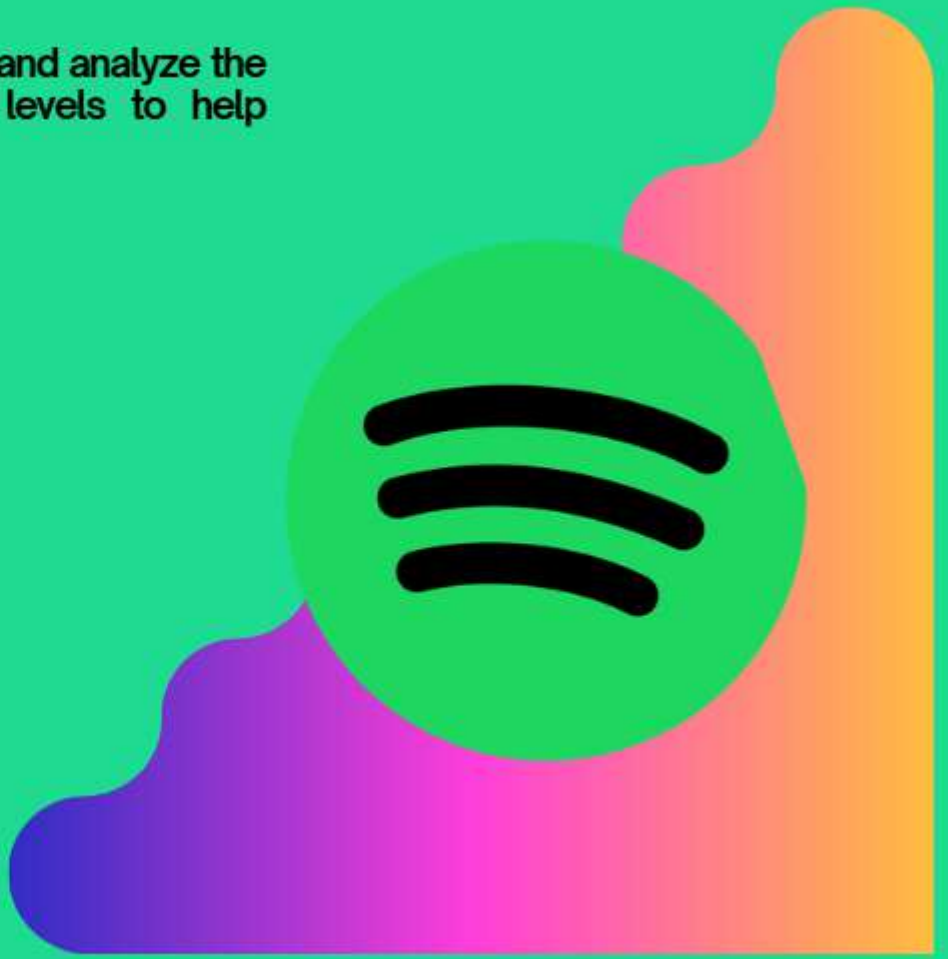
Querying the Data

After the data is inserted, various SQL queries can be written to explore and analyze the data. Queries are categorized into Basic, medium, and advanced levels to help progressively develop SQL proficiency.

Basic Queries
- Simple data retrieval, filtering, and basic aggregations.

Medium Queries
- More complex queries involving grouping, aggregation functions

Advanced Queries
- Nested subqueries, window functions, CTEs.



Basic Level Business Questions

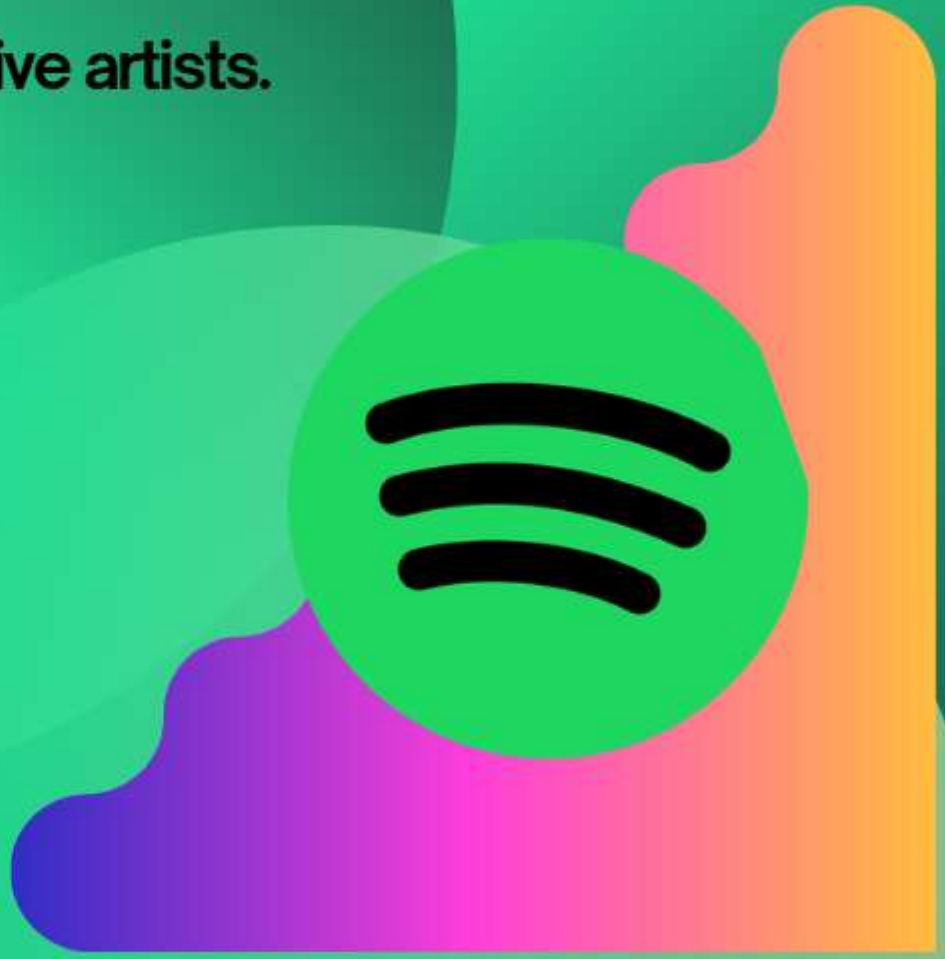
Q1: Retrevie the names of all tracks that have more than 1 billion streams.

```
SELECT *
FROM spotify
WHERE stream > 1000000000;
```

Basic Level Business Questions

Q2: List all albums along with their respective artists.

```
SELECT
   DISTINCT Album,
   Artist
FROM spotify
ORDER BY 1;
```



Basic Level Business Questions

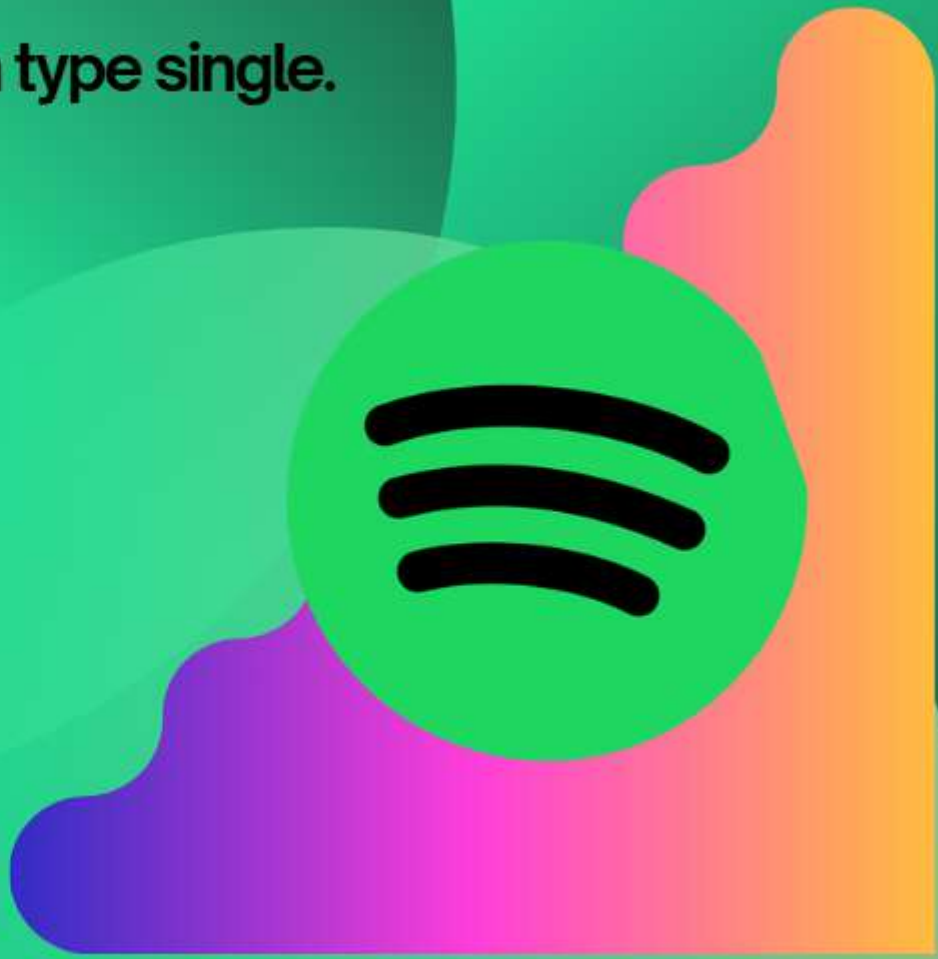
Q3: Get the total number of comments for tracks where licensed = True.

```
SELECT
    SUM(comments) AS total_comments
FROM spotify
WHERE licensed = 'true'
```

Basic Level Business Questions

Q4 :Find all tracks that belong to the album type single.

```
SELECT *
FROM spotify
WHERE album_type='single';
```



Basic Level Business Questions

Q5 :Count the total number of tracks by each artist.

```
SELECT
    Artist,
    COUNT(*) AS total_number
FROM spotify
GROUP BY 1
ORDER BY 2;
```

Medium Level Questions

Q1 :Calculate the average danceability of tracks in each album.

```
SELECT
	Album,
	AVG(danceability) AS avg_danceability
FROM spotify
GROUP BY 1
ORDER BY 2 DESC;
```

Medium Level Questions

Q2 : Find the top 5 tracks with the highest engery values.

```
SELECT
        track,
        MAX(energy) AS highest_energy
FROM spotify
GROUP BY 1
ORDER BY 2 DESC
LIMIT 5;
```

Medium Level Questions

Q3: List all tracks along with their views and likes where official_video = True

```
SELECT
      track,
      views,
      likes
FROM spotify
WHERE official_video = 'true'
```

Medium Level Questions

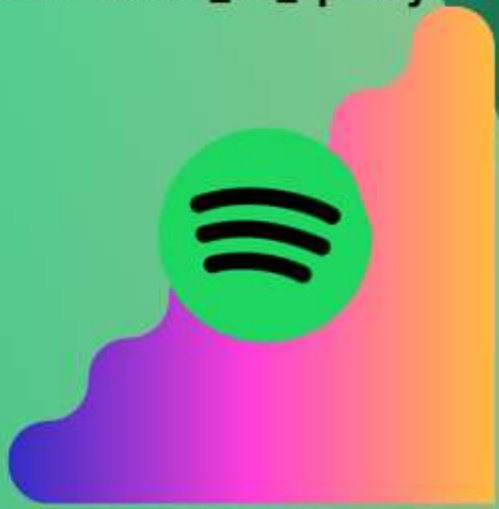
Q4: For each album, calculate the total views of all associated tracks.

```
SELECT
      Album,
      track,
      SUM(views) AS total_views
FROM spotify
GROUP BY 1,2
ORDER BY 3 DESC;
```

Medium Level Questions

Q5: Retrieve the track names that have been streamed on spotify more than youtube.

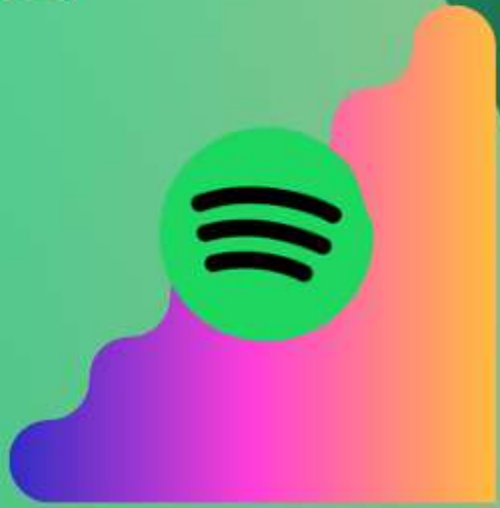
```
SELECT *
FROM(
      SELECT
      track,
      COALESCE(SUM(CASE WHEN most_played_on ='Youtube' THEN stream END),0) AS streamed_on_youtube,
      COALESCE(SUM(CASE WHEN most_played_on ='Spotify' THEN stream END),0) AS streamed_on_spotify
FROM spotify
GROUP BY 1
  ) t1
WHERE streamed_on_spotify>streamed_on_youtube
AND streamed_on_youtube <>0;
```



Advance Level Questions

Q1: Find the top 3 most-viewed tracks for each artist using window functions.

```
WITH ranking_artist AS
  (
      SELECT
       artist,
      track,
     SUM(views) AS total_views,
    DENSE_RANK() OVER(PARTITION BY artist  ORDER BY SUM(views) DESC)
AS rnk
FROM spotify
GROUP BY 1,2
ORDER BY 1,3 DESC
  )
SELECT *
FROM ranking_artist
WHERE rnk <=3;
```



Advance Level Questions

Q2: write a query to find tracks where the liveness score is above the average.

```
SELECT
    track,
    artist,
    liveness
FROM spotify
WHERE liveness> (SELECT AVG(liveness) FROM spotify);
```

Advance Level Questions

Q3: Use a With Clause to calculate the difference between the highest an lowest energy values for tracks in each album.

```
WITH CTE AS
 (
   SELECT
         Album,
         MAX(energy) AS highest_energy,
         MIN(energy) AS lowest_energy
FROM spotify
GROUP BY 1
 )
  SELECT
      album,
      highest_energy-lowest_energy AS energy_diff
  FROM CTE
  ORDER BY 2 DESC;
```

Advance Level Questions

Q4. Find tracks where the energy-to-liveness ratio is greater than 1.2.

```
with track_ratios AS
 (
    SELECT
          track,
          energy,
          liveness,
          (energy/liveness) AS energy_to_liveness
FROM spotify
 )
   SELECT *
   FROM track_ratios
   WHERE (energy/liveness) >1.2
   ORDER BY 3 DESC;
```

Advance Level Questions

Q5: Calculate the cumulative sum of likes for tracks ordered by the number of views, using window functions.

```
SELECT
      track,
      views,
      likes,
     SUM(likes) OVER(ORDER BY views DESC) AS cumulative_likes_artist
FROM spotify
ORDER BY Views DESC;
```

Technology Stack

- Database: PostgreSQL
- SQL Queries: DDL, DML, Aggregations, Joins, Subqueries, Window Functions
- Tools: pgAdmin 4 (or any SQL editor), PostgreSQL (via Homebrew, Docker, or direct installation)

Thank You