Spotify SQL Analysis Project

Project Summary

This SQL-based project focuses on dissecting a Spotify music dataset containing detailed information on songs, albums, and artists. The dataset, initially in a flattened format, is explored and queried through a structured workflow. The project demonstrates the use of SQL to perform data retrieval, aggregation, transformation, and advanced analysis. Queries are organized by difficulty—from basic lookups to complex window functions and CTEs. An additional emphasis is placed on improving performance through query tuning techniques. The overarching aim is to deepen proficiency in SQL while extracting actionable insights from real-world music data.

Columns Overview:

- **Track – Song name**
- **Artist – Performer**
- **Album – Album title**
- **Album_type – Type of album (single, album, etc.)**
- **Views, Likes, Comments, Stream, etc.**
- **Audio features: Danceability, Energy, Loudness, Tempo, Liveness, etc.**
- **Flags: licensed, official_video**
- **Source: most_playedon (Spotify or YouTube)**

Business-Oriented SQL Queries

1. Retrieve the names of all tracks that have more than 10 million streams.

```
SELECT Track

FROM SP_Dataset

WHERE Stream > 10000000;
```

---

2. List all albums along with their respective artists.

```
SELECT DISTINCT Album, Artist

FROM SP_Dataset;
```

---

3. Get the total number of comments for tracks where licensed = TRUE.

```
SELECT Track, COUNT(Comments) AS cmt
```

```
FROM SP_Dataset
WHERE Licensed = 1
GROUP BY Track;
```

---

4. Find all tracks that belong to the album type single.

```
SELECT Track, Album_type
FROM SP_Dataset
WHERE Album_type = 'single';
```

---

5. Count the total number of tracks by each artist.

```
SELECT Artist, COUNT(Track) AS total_tracks
FROM SP_Dataset
GROUP BY Artist;
```

---

6. Calculate the average danceability of tracks in each album.

```
SELECT Album, AVG(Danceability) AS Average_Danceability
FROM SP_Dataset
GROUP BY Album;
```

---

7. Find the top 5 tracks with the highest energy values.

Note: Use LIMIT 5 if you're using MySQL or PostgreSQL, or TOP 5 for SQL Server.

```
SELECT Track, Energy
FROM SP_Dataset
ORDER BY Energy DESC
LIMIT 5;
```

---

8. List all tracks along with their views and likes where official_video = TRUE.

```
SELECT Track, Views, Likes, official_video
FROM SP_Dataset
WHERE official_video = 1;
```

---

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

```
SELECT Album, SUM(Views) AS Total_Views
FROM SP_Dataset
GROUP BY Album;
```

---

10. Retrieve the track names that have been streamed on Spotify more than YouTube.

```
SELECT Track, most_playedon
FROM SP_Dataset
GROUP BY Track, most_playedon
HAVING
   SUM(CASE WHEN most_playedon = 'Spotify' THEN 1 ELSE 0 END) >
   SUM(CASE WHEN most_playedon = 'Youtube' THEN 1 ELSE 0 END);
```

---

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

```
SELECT
   track_id,
   track_name,
   artist_id,
   artist_name,
   views
FROM (
   SELECT
      track_id,
      track_name,
      artist_id,
      artist_name,
      views,
      ROW_NUMBER() OVER (
         PARTITION BY artist_id
         ORDER BY views DESC
      ) AS rank
```

```
    FROM SP_Dataset
) AS ranked_tracks
WHERE rank <= 3;
```

---

12. Write a query to find tracks where the liveness score is above the average.

```
SELECT Track, Liveness
FROM SP_Dataset
WHERE Liveness > (
    SELECT AVG(Liveness)
    FROM SP_Dataset
);
```

---

13. Use a WITH clause to calculate the difference between the highest and lowest energy values for tracks in each album.

```
WITH cte AS (
    SELECT
        Album,
        MAX(Energy) AS highest_energy,
        MIN(Energy) AS lowest_energy
    FROM SP_Dataset
    GROUP BY Album
)
SELECT
    Album,
    highest_energy - lowest_energy AS energy_diff
FROM cte
ORDER BY energy_diff DESC;
```

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

```
SELECT
    Track,
    Energy / Liveness AS Energy_to_Liveness_Ratio
FROM SP_Dataset
WHERE Liveness != 0 AND Energy / Liveness > 1.2;
```

---

15. 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
FROM SP_Dataset;
```