**Analyzing Spotify Data with SQL**

### **Introduction:**

This project focuses on analyzing Spotify's music data using SQL queries to uncover insights into track performance, artist engagement, and user preferences. The queries cover various aspects, such as identifying top tracks based on views and streams, calculating cumulative sums of likes, and exploring metrics like energy-to-liveness ratios. By applying SQL techniques like aggregation, window functions, and subqueries, this project aims to provide a deeper understanding of Spotify's catalog and user behaviour.

**Some Exploratory Data Analysis**

CREATE TABLE spotify (

artist VARCHAR(300),

track VARCHAR(300),

album VARCHAR(300),

album\_type VARCHAR(100),

danceability FLOAT,

energy FLOAT,

loudness FLOAT,

speechiness FLOAT,

acousticness FLOAT,

instrumentalness FLOAT,

liveness FLOAT,

valence FLOAT,

tempo FLOAT,

duration\_min FLOAT,

title VARCHAR(300),

channel VARCHAR(300),

views FLOAT,

likes BIGINT,

comments BIGINT,

licensed BOOLEAN,

official\_video BOOLEAN,

stream BIGINT,

energy\_liveness FLOAT,

most\_played\_on VARCHAR(100)

);

SELECT \* FROM spotify

SELECT COUNT(\*) FROM spotify;

SELECT COUNT(DISTINCT artist) FROM spotify;

SELECT COUNT(DISTINCT album) FROM spotify;

SELECT DISTINCT album\_type 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;

SELECT \* FROM spotify

WHERE duration\_min = 0;

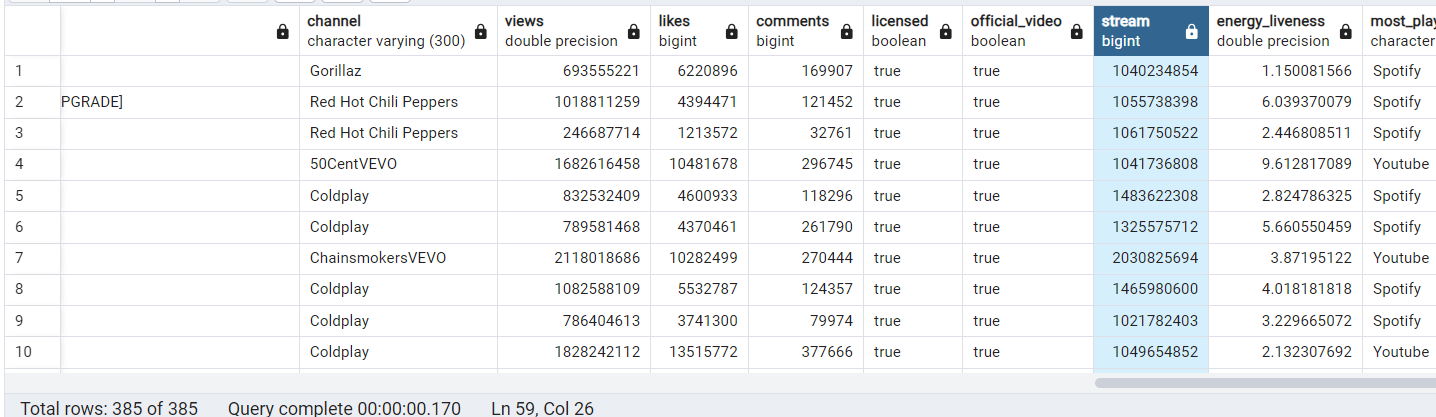
SELECT DISTINCT channel FROM spotify;

**Key Insights from Spotify Using SQL Queries**

**1. Retrieve the names of all tracks that have more than 1 billion streams.**

SELECT \* FROM spotify

WHERE stream > 1000000000;



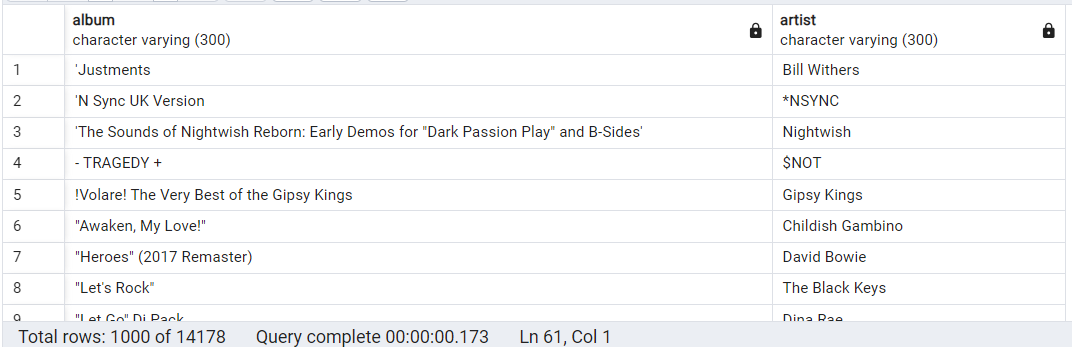
**2. List all albums along with their respective artists.**

SELECT

DISTINCT album, artist

FROM spotify

ORDER BY 1;



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

SELECT SUM(comments) AS total\_comments

FROM spotify

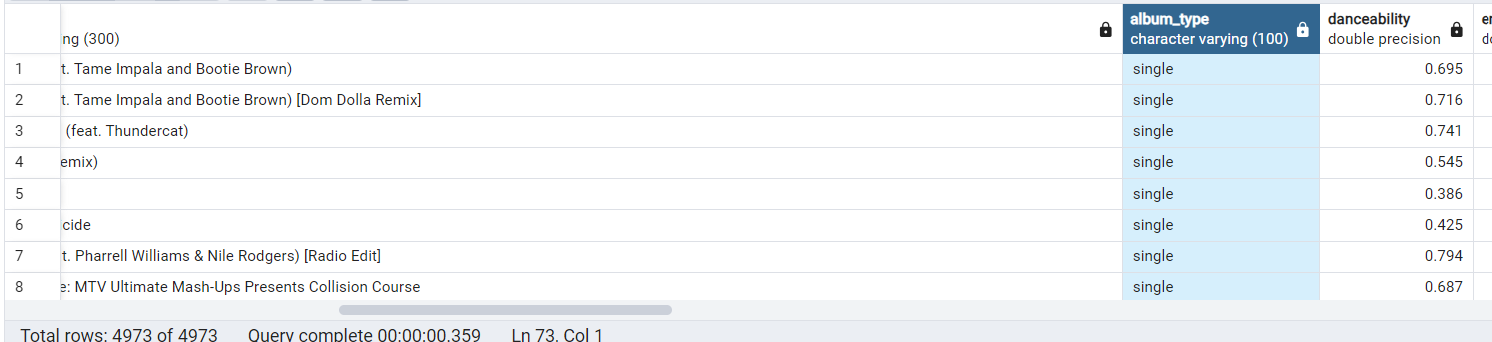
WHERE licensed = TRUE;



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

SELECT \* FROM spotify

WHERE album\_type = 'single';



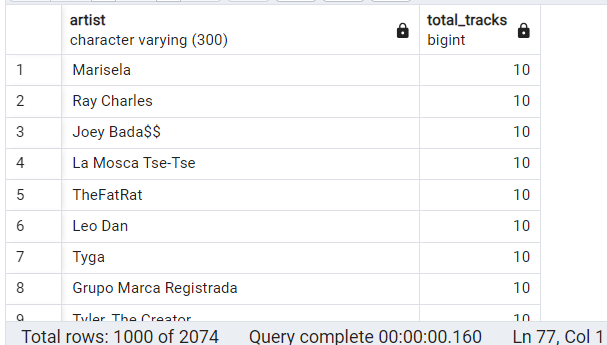
**5. Count the total number of tracks by each artist.**

SELECT artist, COUNT(\*) AS total\_tracks

FROM spotify

GROUP BY artist

ORDER BY total\_tracks DESC;



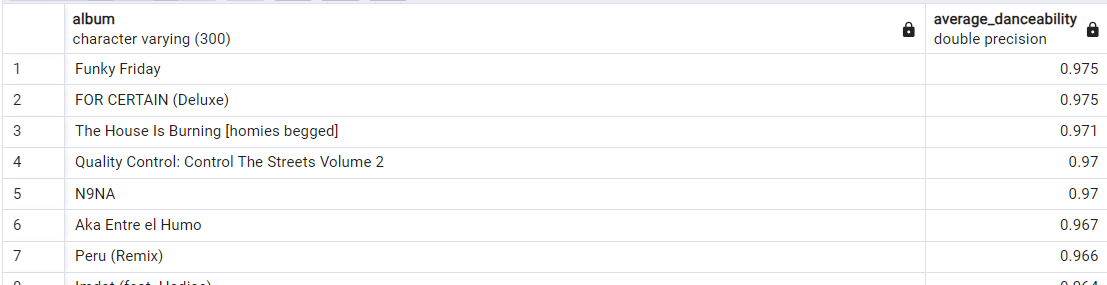
**6. Calculate the average danceability of tracks in each album.**

SELECT album, AVG(danceability) AS average\_danceability

FROM spotify

GROUP BY album

ORDER BY average\_danceability DESC;



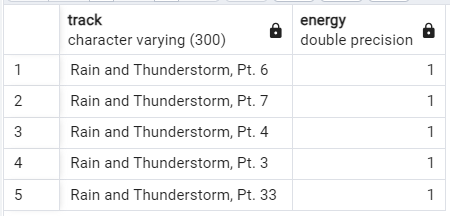
**7. Find the top 5 tracks with the highest energy values.**

SELECT track, energy

FROM spotify

ORDER BY energy DESC

LIMIT 5;



**8. List all tracks along with their views and likes where `official\_video = TRUE`.**

SELECT track,

SUM(views) AS total\_views,

SUM(likes) AS total\_likes

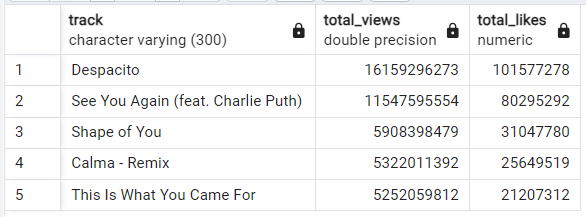
FROM spotify

WHERE official\_video = TRUE

GROUP BY track

ORDER BY total\_views DESC

LIMIT 5;



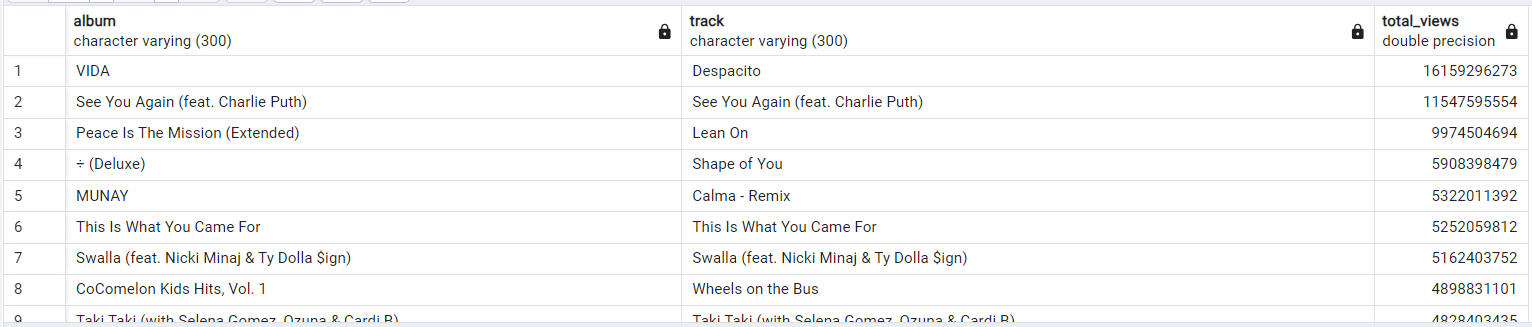
**9. For each album, calculate the total views of all associated tracks.**

SELECT album, track, SUM(views) AS total\_views

FROM spotify

GROUP BY album, track

ORDER BY total\_views DESC;



**10. 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

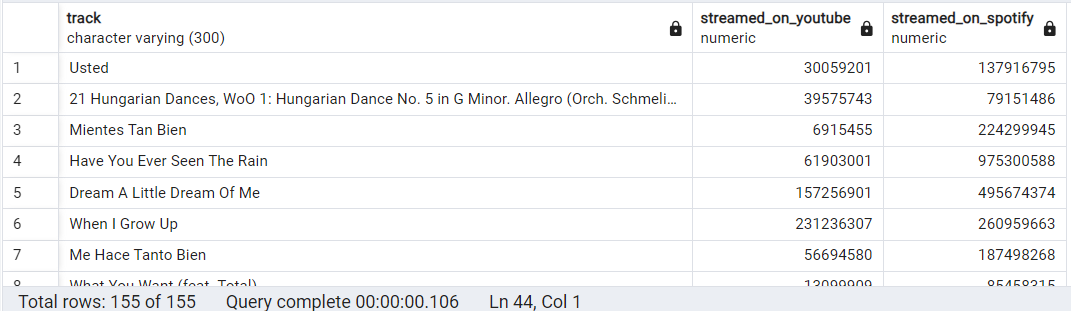
) AS t1

WHERE

streamed\_on\_spotify > streamed\_on\_youtube

AND

streamed\_on\_youtube <> 0;



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

WITH ranking\_artist

AS

(SELECT artist, track,

SUM(views) as total\_view,

DENSE\_RANK() OVER(PARTITION BY artist ORDER BY SUM(views) DESC) as rank

FROM spotify

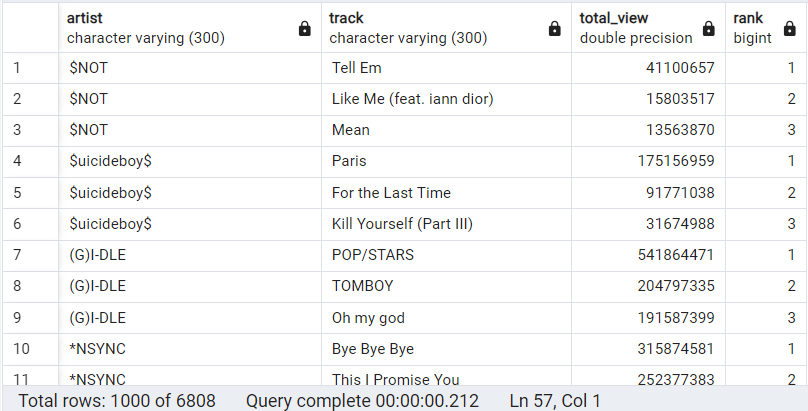
GROUP BY 1,2

ORDER BY 1, 3 DESC

)

SELECT \* FROM ranking\_artist

WHERE rank <=3;

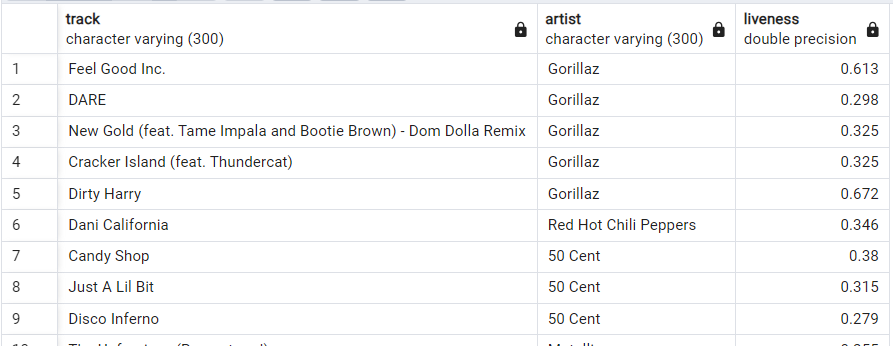


**12. Write a query to find tracks where the liveness score is above the average.**

SELECT track, artist, liveness

FROM spotify

WHERE liveness > (SELECT AVG(liveness) FROM spotify);



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

WITH energy\_stats AS (

SELECT

album,

MAX(energy) AS max\_energy,

MIN(energy) AS min\_energy

FROM spotify

GROUP BY album

)

SELECT

album,

max\_energy,

min\_energy,

(max\_energy - min\_energy) AS energy\_difference

FROM energy\_stats

ORDER BY energy\_difference DESC;

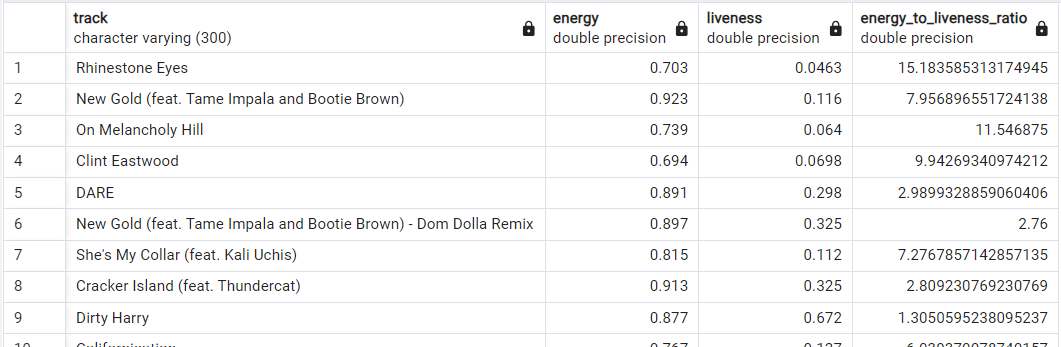


**14. Find tracks where the energy-to-liveness ratio is greater than 1.2.**

SELECT track, energy, liveness, (energy / liveness) AS energy\_to\_liveness\_ratio

FROM spotify

WHERE (energy / liveness) > 1.2;



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

SELECT track,

SUM(likes) OVER (ORDER BY views DESC) AS cumulative\_sum

FROM spotify

ORDER BY cumulative\_sum DESC;



### **Conclusion:**

This project demonstrates how SQL can be used to analyze large datasets from music streaming platforms like Spotify. By exploring key metrics and trends, the project offers valuable insights into track and album performance, which can help guide decisions for artists, labels, and streaming platforms. The use of advanced SQL techniques allows for efficient analysis of complex data, making it easier to uncover meaningful patterns and optimize strategies in the music industry.