SQL Data Analysis Project – Spotify Dataset

Dataset Description

The dataset consists of a single table named spotify, which contains audio features, engagement metrics, and platform-related information for tracks available on Spotify.

Table: spotify

Key columns include:

- artist – Name of the artist
- track – Track name
- album / album_type – Album details
- danceability, energy, loudness, speechiness, acousticness, instrumentalness, liveness, valence, tempo – Audio features
- duration_min – Track duration in minutes
- views, likes, comments, stream – Engagement metrics
- licensed, official_video – Boolean indicators
- energy_liveness – Precomputed ratio
- most_played_on – Platform where the track is most played

This dataset enables analysis of both musical characteristics and user engagement behavior.

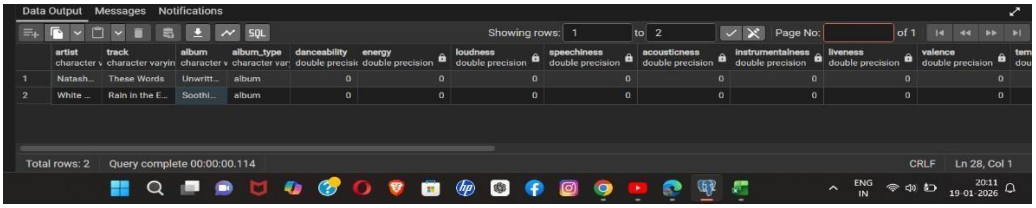
Project Objectives (Business Questions)

The main objectives of this SQL project are:

- Find the top 3 most-viewed tracks for each artist using window functions. ○ Write a query to find tracks where the liveness score is above the average. ○ Use a WITH clause to calculate the difference between the highest and lowest energy values for tracks in each album
- Find tracks where the energy-to-liveness ratio is greater than 1.2. ○ Find the top 5 tracks with the highest energy values. ○ List all tracks along with their views and likes where official_video = TRUE. ○ For each album, calculate the total views of all associated tracks.
- Retrieve the track names that have been streamed on Spotify more than YouTube.

Exploratory Data Analysis (EDA)

```
select *
from spotify;

--Count total rows SELECT COUNT(*)
FROM spotify;                ❼ 20592

-- Count total columns
SELECT COUNT(*) AS Column_Count
FROM INFORMATION_SCHEMA.COLUMNS
WHERE TABLE_NAME = 'spotify';      ❼ 24

--total artist
SELECT DISTINCT artist
FROM spotify;

SELECT COUNT(DISTINCT artist)      ❼ 2074 FROM spotify;

-- Type of album
SELECT DISTINCT album_type          ❼ album , single , complication from spotify;

-- Max and Min Duration
SELECT MAX(Duration_min) as Max_Duration , MIN(Duration_min) as Min_Duration
FROM spotify;
```



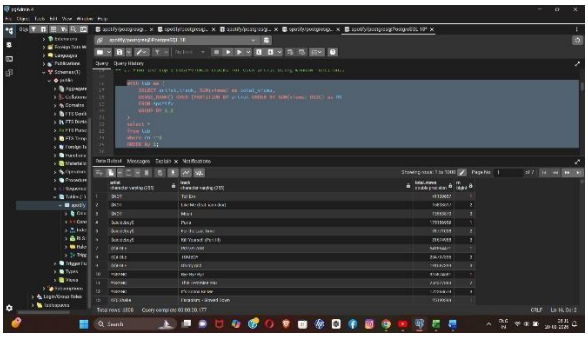

```
-- here we have song with 0 duration , lets check it

SELECT *
FROM spotify
WHERE Duration_min = 0;

-- we got few records with 0 duration , remove them
DELETE FROM spotify
WHERE Duration_min = 0;
```

After removing , current max and min ❼ Max : 77.9343 min Min : 0.516416667 min

SQL Analysis & Queries

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

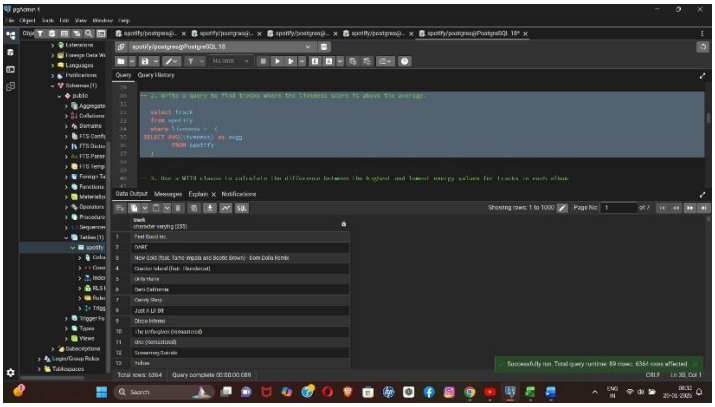
```
with tab as (
        SELECT artist,track, SUM(views) as total_views,
            DENSE_RANK() OVER (PARTITION BY artist ORDER BY SUM(views) DESC) as RN
        FROM spotify
        GROUP BY 1,2
    )
 select *  from tab  where rn
<=3
    ORDER By 1;
-- We dont use group by in top 3 salary in each department
-- but here we have artist artist can have mutiple rows , we need to calculate total of each track thats why we are using group by
```



I grouped by artist and track to calculate total views, applied a window function to rank tracks per artist, and filtered the top three using DENSE_RANK().

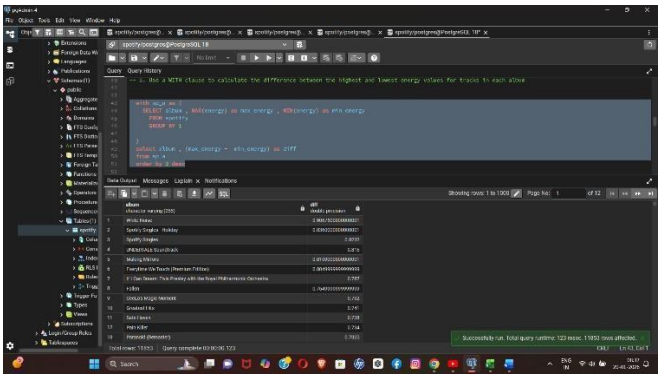
2. Write a query to find tracks where the liveness score is above the average.

```
 select track
from spotify
  where liveness >  (   SELECT
AVG(liveness) as avgg
            FROM spotify
  )
```



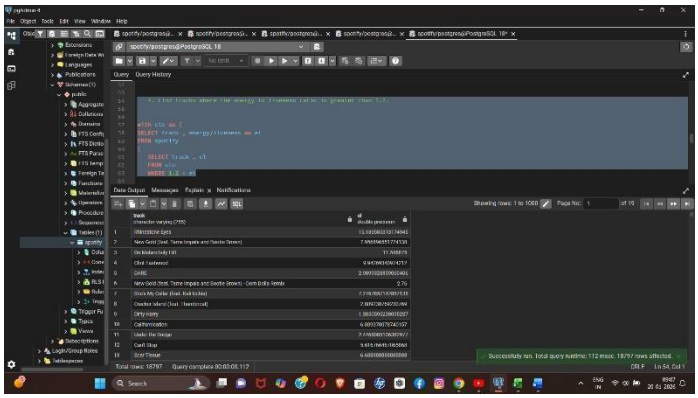
3. Use a WITH clause to calculate the difference between the highest and lowest energy values for tracks in each album

```
with sp_a as (
        SELECT album , MAX(energy) as max_energy , MIN(energy) as min_energy
        FROM spotify
        GROUP BY 1

)
 select album , (max_energy -  min_energy) as diff   from
sp_a
 order by 2 desc
```



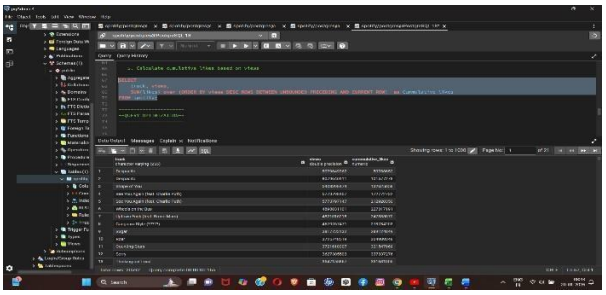
4. Find tracks where the energy-to-liveness ratio is greater than 1.2. with cte as (

```
SELECT track , energy/liveness as el
FROM spotify
)
  SELECT track , el
  FROM cte
  WHERE 1.2 < el
```



5. Calculate cumulative likes based on views

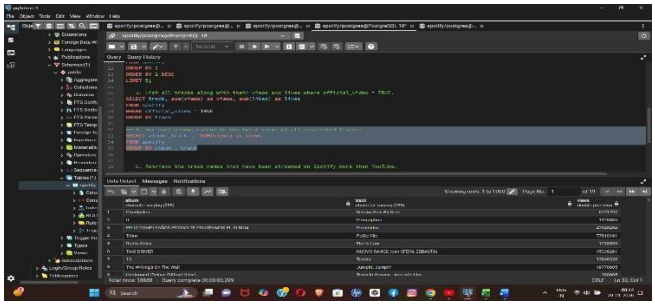
```
SELECT
   track, views,
            SUM(likes) over (ORDER BY views DESC ROWS BETWEEN UNBOUNDED
PRECEDING AND CURRENT ROW)  as Cummulative_likes
FROM spotify;
```



I used a window function with SUM(likes) ordered by views in descending order to calculate cumulative likes as a running total from the most viewed track to the current one.

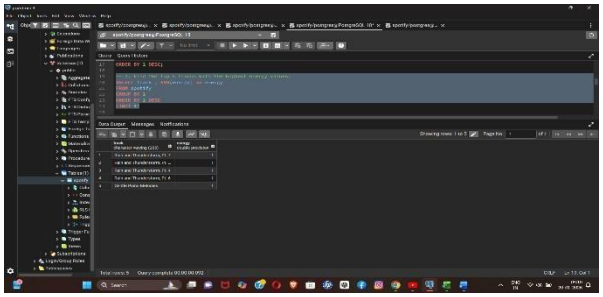
6. For each album, calculate the total views of all associated tracks.

```
SELECT album ,track ,  SUM(views) as views
FROM spotify
GROUP BY album , track
```



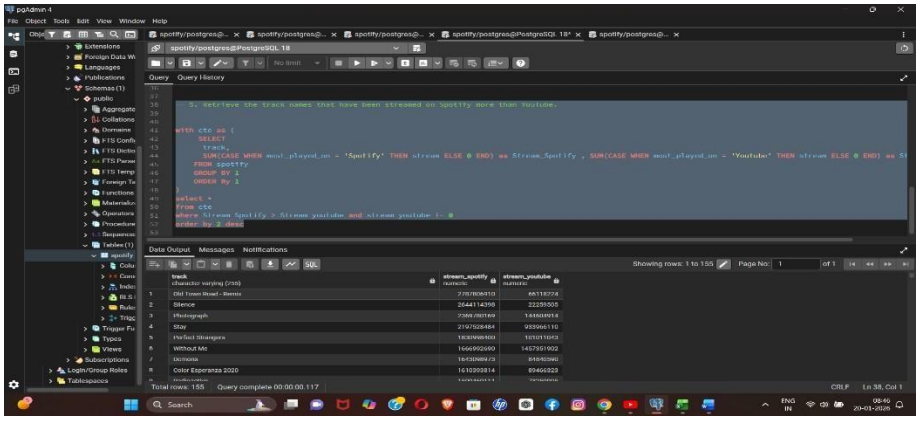
7. Find the top 5 tracks with the highest energy values.

```
SELECT track , AVG(energy) as energy
FROM spotify
GROUP BY 1
ORDER BY 2 DESC
LIMIT 5;
```



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

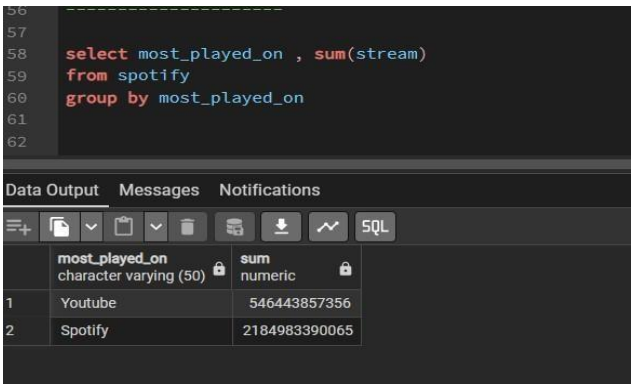
```
with cte as (
SELECT       track,
SUM(CASE WHEN most_played_on = 'Spotify' THEN stream ELSE 0 END) as
Stream_Spotify , SUM(CASE WHEN most_played_on = 'Youtube' THEN stream ELSE 0
END) as Stream_Youtube
FROM spotify
GROUP BY 1
ORDER By 1
)
select * from cte
where Stream_Spotify > Stream_youtube and stream_youtube != 0 order
by 2 desc
```



I used conditional aggregation to calculate Spotify and YouTube streams per track, grouped by track, and then filtered tracks where Spotify streams exceed YouTube streams.

Check Which platform have most streams :

```
select most_played_on , sum(stream) as stream from
spotify
group by most_played_on
```



KEY INSIGHTS GENERATED

1. Top 3 Most-Viewed Tracks per Artist

Insight:

- Artists usually have a small number of tracks driving most of their popularity.
- Popularity is not evenly distributed across all tracks.

Business Value:

- Useful for tour setlists and featured content.

2. Tracks with Liveness Above Average

Insight:

- Only a limited subset of tracks have high liveness values.
- High-liveness tracks often resemble live recordings or concert-style performances.

Business Value:

- Useful for creating live-performance playlists.
- Can guide decisions on live session releases and concert promotions.

3. Energy Variation Within Albums

Insight:

- Some albums show high energy variation, meaning they mix calm and intense tracks.
- Other albums are energy-consistent, maintaining a similar mood throughout.

Business Value:

- Helps identify albums best suited for:
    - Workout playlists (high, consistent energy)
    - Story-telling or concept albums

4. Tracks with Energy-to-Liveness Ratio > 1.2

Insight:

- These tracks are high-energy but studio-produced, not live.

Business Value:

- Ideal candidates for:
    - Commercial promotions and ads **5.**

Top 5 Highest-Energy Tracks

Insight:

- High-energy tracks are usually shorter, faster, and intense.
- They often overlap with dance and party-oriented genres.

Business Value:

- Useful for fitness playlists, festival sets, and high-engagement ads.

6. Cumulative likes based on views

Insight:

- Identify the Most Influential Tracks **Business Value:**
- Popularity Distribution

7. Total Views per Album

Insight:

- Album popularity is often driven by one or two blockbuster tracks.
- Some albums perform well overall, while others rely on a single hit.

Business Value:

- Supports album-level performance analysis.
- Helps labels decide between single-driven vs album-driven strategies.

8. Spotify vs YouTube Streaming Comparison

Insight:

- Certain tracks perform significantly better on Spotify than YouTube.
- Indicates platform-specific audience behavior.

Business Value:

- Helps optimize platform-specific marketing strategies.
- Useful for deciding where to prioritize exclusive releases

This project demonstrates how SQL analytics can uncover patterns in music popularity, engagement, and platform performance, enabling data-driven decisions for marketing, playlist curation, and content strategy.

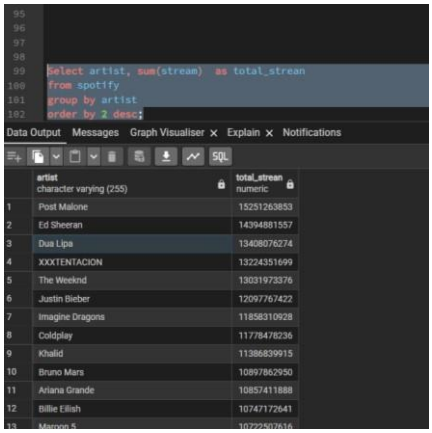
Dataset Quality Notes

- Null checks for likes, views, streams
- Handling duplicates if any
- Removed removed tracks with 0 duration

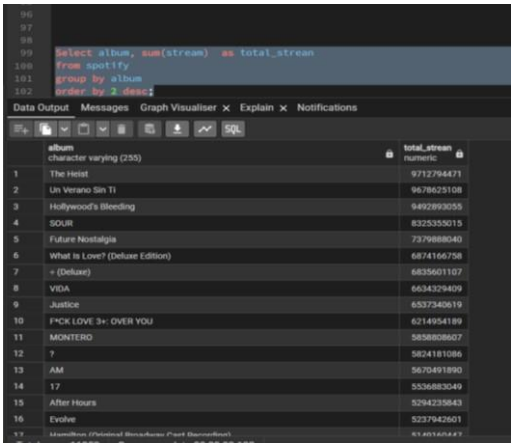
BASED ON STREAM

Artist with most stream :

```
Select artist, sum(stream)  as total_strean
from spotify group by artist
order by 2 desc;
```



	artist character varying (255)	total_strean numeric
1	Post Malone	15251263853
2	Ed Sheeran	14394881557
3	Dua Lipa	13408076274
4	XXXTENTACION	13224351699
5	The Weeknd	13031973376
6	Justin Bieber	12097767422
7	Imagine Dragons	11858310928
8	Coldplay	11778478236
9	Khalid	11386839915
10	Bruno Mars	10897862950
11	Ariana Grande	10857411888
12	Billie Eilish	10747172641
13	Maroon 5	10722507616



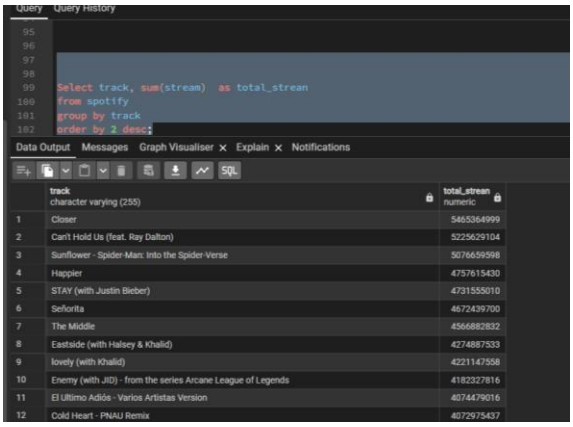
	album character varying (255)	total_strean numeric
1	The Heist	9712794471
2	Un Verano Sin Ti	9678625108
3	Hollywood's Bleeding	9492893055
4	SOUR	8325355015
5	Future Nostalgia	7379888040
6	What Is Love? (Deluxe Edition)	6874166758
7	+ (Deluxe)	6835601107
8	VIDA	6634329409
9	Justice	6537340619
10	F*CK LOVE 3+: OVER YOU	6214954189
11	MONTERO	5858808607
12	?	5824181086
13	AM	5670491890
14	17	5536883049
15	After Hours	5294235843
16	Evolve	5237942601

Album with most stream :

```
Select album, sum(stream)  as total_strean from
spotify
group by album
order by 2 desc;
```

Track with most stream :

```
Select track, sum(stream)  as
total_strean from spotify group by track
order by 2 desc;
```

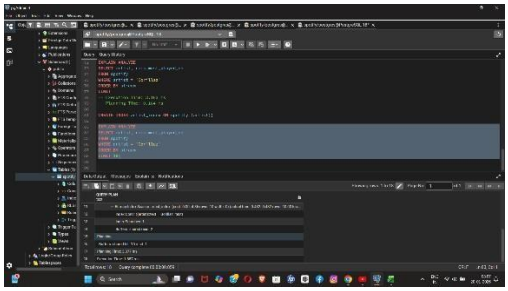


	track character varying (255)	total_strean numeric
1	Closer	5465364999
2	Can't Hold Us (feat. Ray Dalton)	5225629104
3	Sunflower - Spider-Man: Into the Spider-Verse	5076659598
4	Happier	4757615430
5	STAY (with Justin Bieber)	4731555010
6	Señorita	4672439700
7	The Middle	4566882832
8	Eastside (with Halsey & Khalid)	4274887533
9	lovely (with Khalid)	4221147558
10	Enemy (with JID) - from the series Arcane League of Legends	4182327816
11	El Ultimo Adiós - Varios Artistas Version	4074479016
12	Cold Heart - PNAU Remix	4072975437

QUERY OPTIMIZATION

Before having index :

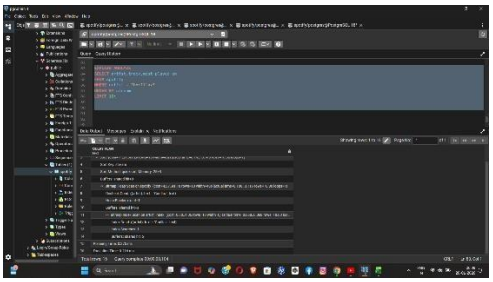
```
EXPLAIN ANALYZE
SELECT artist,track,most_played_on
FROM spotify
WHERE artist = 'Gorillaz'
ORDER BY stream
LIMIT 10;
```

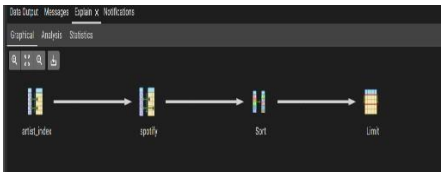


```
-- Execution Time: 3.163 ms
-- Planning Time: 0.164 ms
```

LETS ADD INDEX

```
CREATE INDEX artist_index ON spotify (artist);

EXPLAIN ANALYZE
SELECT artist,track,most_played_on
FROM spotify
WHERE artist = 'Gorillaz'
ORDER BY stream
LIMIT 10;
```






```
-- Execution Time: 0.375 ms
-- Planning Time: 0.216 ms
```

Insight :

By creating an index on the artist column ,queries filtering by artist became significantly faster.

I optimized queries using indexing, which significantly improved execution speed, reduced resource usage on large music datasets.

SQL Performance Metrics

After query optimization, a **small table showing "Before vs After Index"**:

Query	Execution Time	Planning Time	Improvement
Original	3.163 ms	0.164 ms	-
After Index	0.375 ms	0.216 ms	88% faster

"Key Takeaways / Recommendations"

After analysing the generated insights , these are some key Takeways:

- **Marketing / Promotions:** Focus campaigns on top tracks or artists with highly engaging tracks.
- **Playlist Curation:** Use high-energy or high-liveness tracks for specific moods.
- **Platform Strategy:** Release exclusive content on Spotify if the track performs better there.
- **Album Strategy:** Identify whether albums are single-driven or evenly popular.

Conclusion

Using SQL, this project analyzed the Spotify dataset to uncover track popularity, audio features, user engagement, and platform performance. Queries were optimized using indexing, and insights generated can guide playlist curation, marketing strategy, and platform-specific decisions.