

The background features decorative elements in the corners: top-left and bottom-right have overlapping light blue rectangles and diagonal lines; top-right and bottom-left have a grid of small light blue circles. The main title is centered in a large, bold, black sans-serif font.

SPOTIFY DATA ANALYSIS

GitHub Link

OVERVIEW

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), and optimizing query performance. The primary goals of the project are to practice advanced SQL skills and generate valuable insights from the dataset.

BASIC QUERIS

- Simple data retrieval, filtering, and basic aggregations.
1. Retrieve the names of all tracks that have more than 1 billion streams
 2. List all albums along with their respective artists
 3. Get the total number of comments for tracks where licensed = TRUE
 4. Find all tracks that belong to the album type single
 5. Count the total number of tracks by each artist

```
4
5  -- Retrieve the names of all tracks that have more than 1 billion streams.
6  ✓ EXPLAIN ANALYZE
7  SELECT * FROM spotify
8  WHERE stream >= 1000000000;
9
10 -- List all albums along with their respective artists.
11 ✓ SELECT DISTINCT album AS album, artist
12 FROM spotify;
13
14 -- Get the total number of comments for tracks where licensed = TRUE.
15 ✓ SELECT SUM(comments) AS total_comments
16 FROM spotify
17 WHERE licensed = 'true';
18
19 -- Find all tracks that belong to the album type single.
20 ✓ SELECT *
21 FROM spotify
22 WHERE album_type = 'single';
23
24 -- Count the total number of tracks by each artist.
25 ✓ SELECT artist, COUNT(track) AS number_of_tracks
26 FROM spotify
27 GROUP BY artist;
28
```

MEDIUM QUERIS

- More complex queries involving grouping, aggregation functions, and joins.
 1. Calculate the average danceability of tracks in each album
 2. Find the top 5 tracks with the highest energy values
 3. List all tracks along with their views and likes where `official_video = TRUE`
 4. For each album, calculate the total views of all associated tracks
 5. Retrieve the track names that have been streamed on Spotify more than YouTube

```
35
36 -- Calculate the average danceability of tracks in each album.
37 ✓ SELECT
38     album,
39     AVG(danceability) AS average_danceability
40 FROM spotify
41 GROUP BY album;
42
43 -- Find the top 5 tracks with the highest energy values.
44 ✓ SELECT
45     track,
46     MAX(energy) AS energy
47 FROM spotify
48 GROUP BY 1
49 ORDER BY 2 DESC
50 LIMIT 5;
51
52 -- List all tracks along with their views and likes where official_video = TRUE.
53 ✓ SELECT
54     track
55     SUM(views) AS total_views,
56     SUM(likes) AS total_likes
57 FROM spotify
58 WHERE official_video='true'
59 GROUP BY 1;
60
```

```
60
61 -- For each album, calculate the total views of all associated tracks.
62 ✓ SELECT
63     album,
64     track,
65     SUM(views) AS total_views
66 FROM spotify
67 GROUP BY 1,2
68 ORDER BY 3 DESC;
69
70 -- Retrieve the track names that have been streamed on Spotify more than YouTube.
71 ✓ SELECT *
72 FROM (
73     SELECT
74         track,
75         COALESCE(SUM(CASE WHEN most_played_on='Youtube' THEN stream END),0) AS stream_on_youtube,
76         COALESCE(SUM(CASE WHEN most_played_on='Spotify' THEN stream END),0) AS stream_on_spotify
77 FROM spotify
78 GROUP BY 1
79 )
80 WHERE stream_on_youtube < stream_on_spotify AND
81 stream_on_youtube<>0;
82
83
```

ADVANCE QUERIES

- Nested subqueries, window functions, CTEs, and performance optimization
1. Find the top 3 most-viewed tracks for each artist using window functions.
 2. Write a query to find tracks where the liveness score is above the average.
 3. Use a WITH clause to calculate the difference between the highest and lowest energy values for tracks in each album.
 4. Find tracks where the energy-to-liveness ratio is greater than 1.2.
 5. Calculate the cumulative sum of likes for tracks ordered by the number of views, using window functions.


```
88
89 -- Find the top 3 most-viewed tracks for each artist using window functions.
90 ✓ WITH ranking_artist AS(
91     SELECT
92         artist,
93         track,
94         SUM(views) AS total_views,
95         DENSE_RANK() OVER(PARTITION BY artist ORDER BY SUM(views) DESC) AS rank
96     FROM spotify
97     GROUP BY 1,2
98     ORDER BY 1,3 DESC
99 )
100
101 SELECT * FROM ranking_artist
102 WHERE rank<=3;
103
104 -- Write a query to find tracks where the liveness score is above the average.
105 ✓ SELECT *
106     FROM spotify
107     WHERE liveness > (
108         SELECT AVG(liveness) FROM spotify
109     );
110
```

```
111
112 -- Use a WITH clause to calculate the difference between the highest and lowest energy values for tracks in each album.
113 ✓ WITH track_energy AS(
114     SELECT
115         album,
116         MAX(energy) AS max_energy,
117         MIN(energy) AS min_energy
118     FROM spotify
119     GROUP BY 1
120 )
121
122     SELECT
123     album,
124     (max_energy-min_energy) as difference
125     FROM track_energy
126     ORDER BY 2 DESC;
127
```

```
128
129 -- Find tracks where the energy-to-liveness ratio is greater than 1.2.
130 ✓ SELECT
131     track,
132     energy / liveness AS energy_to_liveness_ratio
133 FROM Spotify
134 WHERE energy / NULLIF(liveness,0) > 1.2 ;
135
136
137 -- Calculate the cumulative sum of likes for tracks ordered by the number of views, using window functions.
138 ✓ SELECT
139     track,
140     SUM(likes) OVER (ORDER BY views) AS cumulative_sum
141 FROM Spotify
142 ORDER BY 2 DESC;
```
