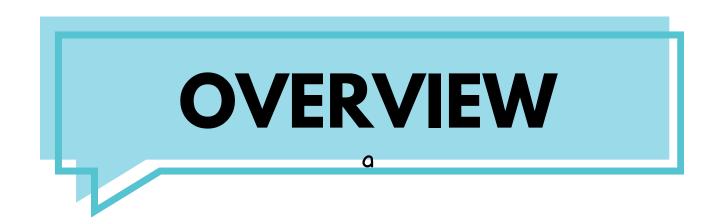


SPOTIFY DATA ANALYSIS

0000

GitHub Link



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.
- I. Retrieve the names of all tracks that have more than I billion streams

0000

0000

- 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
     -- Retrieve the names of all tracks that have more than 1 billion streams.
 6 V EXPLAIN ANALYZE
     SELECT * FROM spotify
     WHERE stream>=100000000;
 8
     -- List all albums along with their respective artists.
10
11 ✔ SELECT DISTINCT album AS album, artist
     FROM spotify;
12
13
     -- Get the total number of comments for tracks where licensed = TRUE.
14
15 	✓ SELECT SUM(comments) AS total_comments
     FROM spotify
16
     WHERE licensed = 'true';
17
18
     -- Find all tracks that belong to the album type single.
19
20 V SELECT *
     FROM spotify
21
     WHERE album_type = 'single';
22
23
     -- Count the total number of tracks by each artist.
24
25 V SELECT artist, COUNT(track) AS number_of_tracks
     FROM spotify
26
     GROUP BY artist;
27
28
```

MEDIUM QUERIS

More complex queries involving grouping, aggregation functions, and joins.

0000

- l. 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

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

```
00
     -- For each album, calculate the total views of all associated tracks.
62 SELECT
         album,
63
         track,
64
         SUM(views) AS total_views
65
     FROM spotify
66
     GROUP BY 1,2
67
     ORDER BY 3 DESC;
68
69
     -- Retrieve the track names that have been streamed on Spotify more than YouTube.
71 SELECT *
     FROM (
72
     SELECT
73
         track,
74
         COALESCE(SUM(CASE WHEN most_played_on='Youtube' THEN stream END),0) AS stream_on_youtube,
75
         COALESCE(SUM(CASE WHEN most_played_on='Spotify' THEN stream END),0) AS stream_on_spotify
76
     FROM spotify
77
     GROUP BY 1
78
79
     WHERE stream_on_youtube < stream_on_spotify AND</pre>
80
     stream_on_youtube<>0;
81
82
83
```

ADVANCE QUERIES

- Nested subqueries, window functions, CTEs, and performance optimization
- I. 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.

```
00
      -- Find the top 3 most-viewed tracks for each artist using window functions.
 90 ∨ WITH ranking_artist AS(
      SELECT
 91
          artist,
 92
          track,
 93
          SUM(views) AS total_views,
 94
          DENSE_RANK() OVER(PARTITION BY artist ORDER BY SUM(views) DESC) AS rank
 95
      FROM spotify
 96
      GROUP BY 1,2
 97
      ORDER BY 1,3 DESC
 98
 99
100
      SELECT * FROM ranking_artist
101
      WHERE rank<=3;
102
103
      -- Write a query to find tracks where the liveness score is above the average.
104
105 ∨ SELECT *
      FROM spotify
106
      WHERE liveness > (
107
      SELECT AVG(liveness) FROM spotify
108
109
      );
110
```

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

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