Spotify Advanced SQL Project and Query Optimization

Unlocking Music Insights with PostgreSQL and High-Performance

- **Tool Used:** PostgreSQL
- **Focus Areas:** SQL Mastery | Query Performance | Data Insights

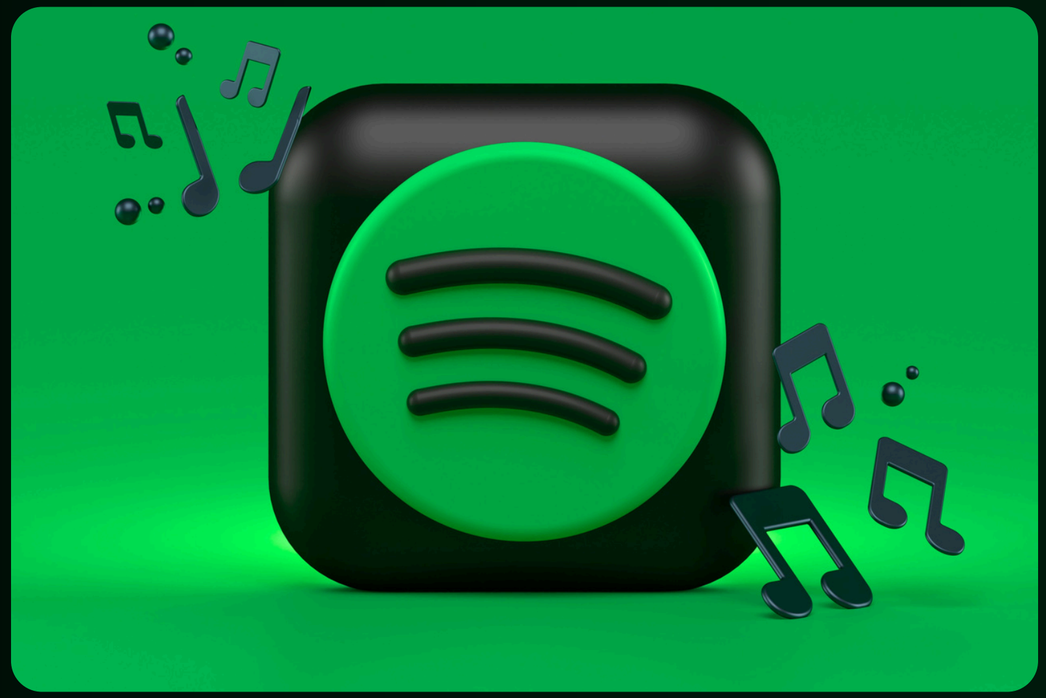
Project Overview

This project involves analyzing a denormalized Spotify dataset using PostgreSQL. It covers:

- Data normalization
- Writing SQL queries of varying complexity
- Query performance optimization

Primary Goals:

- Practice advanced SQL techniques
- Extract valuable insights from music data



Dataset Summary
Dataset Features

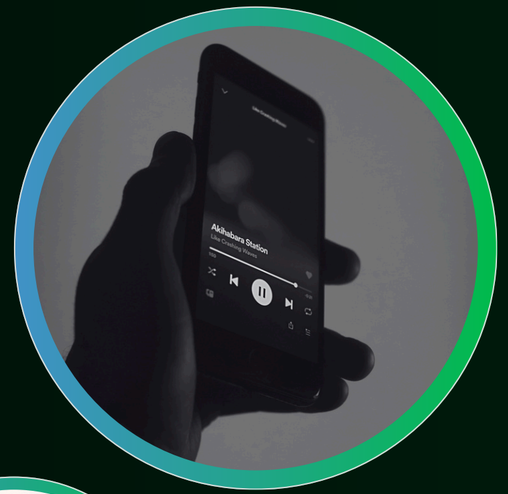
- **Artist:** Performer of the track
- **Track:** Name of the song
- **Album:** Album name
- **Album_type:** Single or full album
- **Metrics:** Danceability, Energy, Loudness, Tempo, etc.

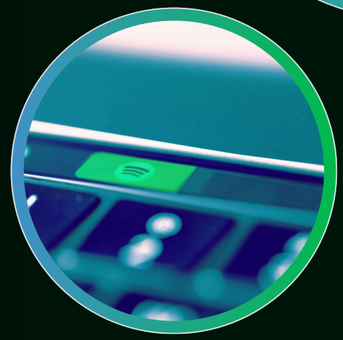
Purpose: Gain deep familiarity with data before querying.

SQL Query Categories
Query Difficulty Levels

1. **Easy:** Basic retrieval, filters, counts
2. **Medium:** Joins, aggregations, grouping
3. **Advanced:** Subqueries, CTEs, window functions

Each level builds deeper SQL fluency.





Easy Level Queries

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

```
SELECT * FROM spotify
WHERE stream > 1000000000
LIMIT 5;
```

	artist character	track character	album character	album_type character va	danceability double precis	energy double	loudness double pr	speechiness double precis	acousticness double precisi	instrumentalness double precision	liveness double pr	valence double precision	tempo double	duration_min double precis	title charact	channel character	views double	likes bigint	comments bigint	licensed boolean	official_vi boolean
1	Goril...	Feel ...	Demon...	album	0.818	0.705	-6.679	0.177	0.00836	0.00233	0.613	0.772	138.55	3.710666666	Gori...	Gorillaz	693555	622089	169907	true	true
2	Red ...	Calif...	Califor...	album	0.592	0.767	-2.788	0.027	0.0021	0.00165	0.127	0.328	96.483	5.49555	Red ...	Red H...	101881	439447	121452	true	true
3	Red ...	Und...	Blood ...	album	0.559	0.345	-13.496	0.0459	0.0576	0.000105	0.141	0.458	84.581	4.405116666	Red ...	Red H...	246687	121357	32761	true	true
4	50 C...	In D...	Get Ric...	album	0.902	0.72	-2.776	0.347	0.26	0	0.0749	0.805	90.059	3.22445	50 C...	50Cen...	168261	104816	296745	true	true
5	Cold...	Yellow	Parach...	album	0.429	0.661	-7.227	0.0281	0.00239	0.000121	0.234	0.285	173.372	4.446216666	Cold...	Coldpl...	832532	460093	118296	true	true

Total rows: 385 Query complete 00:00:00.121 CRLF Ln 65, Col 1

Easy Level Queries

2. List all albums along with their respective artists.

```
SELECT
    DISTINCT album, artist
FROM spotify
ORDER BY 1;

SELECT
    DISTINCT album
FROM spotify
ORDER BY 1;
```

	album character varying (255)
1	'Justments
2	'N Sync UK Version
3	'The Sounds of Nightwish Reborn: Early Demos for "Dark Passion Play" and B-Sides'
4	- TRAGEDY +
5	!Volare! The Very Best of the Gipsy Kings
6	"Awaken, My Love!"

Total rows: 11853 Query complete 00:00:00.200

Easy Level Queries

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

```sql
SELECT
    SUM(comments) AS total_comments
FROM spotify
WHERE licensed = 'true';
```

	total_comments numeric
1	497015695

Easy Level Queries

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

```
SELECT * FROM spotify
WHERE album_type = 'single';
```

	artist	track	album	album_type	danceability	energy	loudness	speechiness	acousticness	instrumentalness	liveness	valence	tempo	duration_min	title	channel	views	likes	comments	licensed	official_video	stream
	character	characte	characte	character var	double precisi	double p	double pre	double precis	double precisi	double precision	double pr	double p	double p	double precisi	characte	character	double p	bigint	bigint	boolean	boolean	bigint
1	Gorill...	New ...	New...	single	0.695	0.923	-3.93	0.0522	0.0425	0.0469	0.116	0.551	108.014	3.5858333333	Gori...	Gorill...	843505	282142	7399	true	true	6306
2	Gorill...	New ...	New...	single	0.716	0.897	-7.185	0.0629	0.012	0.262	0.325	0.358	127.03	4.5690333333	Gori...	Dom ...	451996	11686	241	false	true	1066
3	Gorill...	Crac...	Crac...	single	0.741	0.913	-3.34	0.0465	0.00343	0.103	0.325	0.643	120.012	3.5625	Gori...	Gorill...	244598	739527	20296	true	true	4267
4	50 C...	Best ...	Best...	single	0.545	0.64	-3.529	0.308	0.368	0	0.111	0.574	181.733	4.24155	50 C...	Light...	291023	5729	46	false	false	16003
5	Metal...	Lux ...	Lux ...	single	0.386	0.996	-2.96	0.0754	1.15e-05	0.000836	0.116	0.188	130.573	3.428	Met...	Metal...	149379	473751	46230	true	true	2006

Total rows: 4973 | Query complete 00:00:00.199 | CRLF | Ln 94, Col 8

Easy Level Queries

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

```sql
SELECT
    artist,
    COUNT(*) AS total_no_songs
FROM spotify
GROUP BY artist
ORDER BY 2;
```

	artist character varying (255)	total_no_songs bigint
1	Stars Music Chile	1
2	Bootie Brown	1
3	back number	1
4	Vaundy	2
5	Mrs. GREEN APPLE	2
6	Jimin	3

Total rows: 2074 Query complete 00:00:00.094

Medium Level Queries

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

```sql
SELECT
    album,
    AVG(danceability) AS avg_danceability
FROM spotify
GROUP BY 1
ORDER BY 2 DESC;
```

	album character varying (255)	avg_danceability double precision
1	FOR CERTAIN (Deluxe)	0.975
2	Funky Friday	0.975
3	The House Is Burning [homies begged]	0.971
4	Quality Control: Control The Streets Volume 2	0.97
5	N9NA	0.97
6	Aka Entre el Humo	0.967

Total rows: 11853 Query complete 00:00:00.145

Medium Level Queries

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

```
SELECT
    track,
    MAX(energy) AS max_energy
FROM spotify
GROUP BY 1
ORDER BY 2 DESC
LIMIT 5;
```

	track character varying (255)	max_energy double precision
1	Rain and Thunderstorm, Pt. 7	1
2	Rain and Thunderstorm, Pt. 33	1
3	Rain and Thunderstorm, Pt. 4	1
4	Rain and Thunderstorm, Pt. 6	1
5	Gentle Piano Melodies	1

Total rows: 5 Query complete 00:00:00.124

Medium Level Queries

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

```sql
SELECT
    track,
    SUM(views) AS total_views,
    SUM(likes) AS total_likes
FROM spotify
WHERE official_video = 'true'
GROUP BY 1
ORDER BY 2 DESC;
```

	track character varying (255)	total_views double precision	total_likes numeric
1	Despacito	16159296273	101577278
2	See You Again (feat. Charlie Puth)	11547595554	80295292
3	Shape of You	5908398479	31047780
4	Calma - Remix	5322011392	25649519
5	This Is What You Came For	5252059812	21207312

Total rows: 13650 Query complete 00:00:00.175

Medium Level Queries

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

```
SELECT
    album,
    track,
    SUM(views)
FROM spotify
GROUP BY 1, 2
ORDER BY 3 DESC;
```

	album character varying (255)	track character varying (255)	sum double precision
1	VIDA	Despacito	16159296273
2	See You Again (feat. Charlie Puth)	See You Again (feat. Charlie Puth)	11547595554
3	Peace Is The Mission (Extended)	Lean On	9974504694
4	÷ (Deluxe)	Shape of You	5908398479
5	MUNAY	Calma - Remix	5322011392

Total rows: 18680 Query complete 00:00:00.159

Medium Level Queries

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

```sql
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;
```

	track character varying (255)	streamed_on_youtube numeric	streamed_on_spotify numeric
1	Usted	30059201	137916795
2	21 Hungarian Dances, Wo...	39575743	79151486
3	Mientes Tan Bien	6915455	224299945
4	Have You Ever Seen The R...	61903001	975300588
5	Dream A Little Dream Of Me	157256901	495674374

Total rows: 155 Query complete 00:00:00.181

Advanced Level Queries

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

```sql
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;
```

	artist character varying (255)	track character varying (255)	total_view double precision	rank bigint
1	$NOT	Tell Em	41100657	1
2	$NOT	Like Me (feat. iann di...	15803517	2
3	$NOT	Mean	13563870	3
4	$uicideboy$	Paris	175156959	1
5	$uicideboy$	For the Last Time	91771038	2

Total rows: 6808 Query complete 00:00:00.324

Advanced Level Queries

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

```sql
SELECT
    track,
    artist,
    liveness
FROM spotify
WHERE liveness > (SELECT AVG(liveness) FROM spotify);
```

	track character varying (255)	artist character varying (255)	liveness double precision
1	Feel Good Inc.	Gorillaz	0.613
2	DARE	Gorillaz	0.298
3	New Gold (feat. Tame I...	Gorillaz	0.325
4	Cracker Island (feat. Th...	Gorillaz	0.325
5	Dirty Harry	Gorillaz	0.672

Total rows: 6364 Query complete 00:00:00.209

Advanced Level Queries

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

```sql
WITH cte
AS
(SELECT
    album,
    MAX(energy) AS highest_energy,
    MIN(energy) AS lowest_energy
FROM spotify
GROUP BY 1
)
SELECT
album,
highest_energy - lowest_energy AS energy_diffrent
FROM cte
ORDER BY 2 DESC;
```

	album character varying (255)	energy_diffrent double precision
1	White Noise	0.9067500000000001
2	Spotify Singles - Holiday	0.8360000000000001
3	Spotify Singles	0.8232
4	UNDERTALE Soundtrack	0.816
5	Making Mirrors	0.8109000000000001

Total rows: 11853 Query complete 00:00:00.149

Advanced Level Queries

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

```sql
SELECT
    artist,
    track,
    album,
    energy,
    liveness,
    (energy / NULLIF(liveness, 0)) AS energy_liveness_ratio
FROM
    spotify
WHERE
    (energy / NULLIF(liveness, 0)) > 1.2
ORDER BY
    energy_liveness_ratio DESC;
```

	artist character	track character	album character	energy double pre	liveness double prec	energy_liveness_ratio double precision
1	IVE	Take It	ELEVE...	0.934	0.0158	59.11392405063291
2	Juan ...	Veran...	Veran...	0.87	0.015	58
3	Bacilos	Salva...	Salvav...	0.836	0.0145	57.655172413793096
4	Dave ...	Ants ...	Under ...	0.856	0.0157	54.52229299363058
5	Fons...	Eres ...	Ilusión	0.93	0.0181	51.38121546961326

Total rows: 18797 Query complete 00:00:00.220

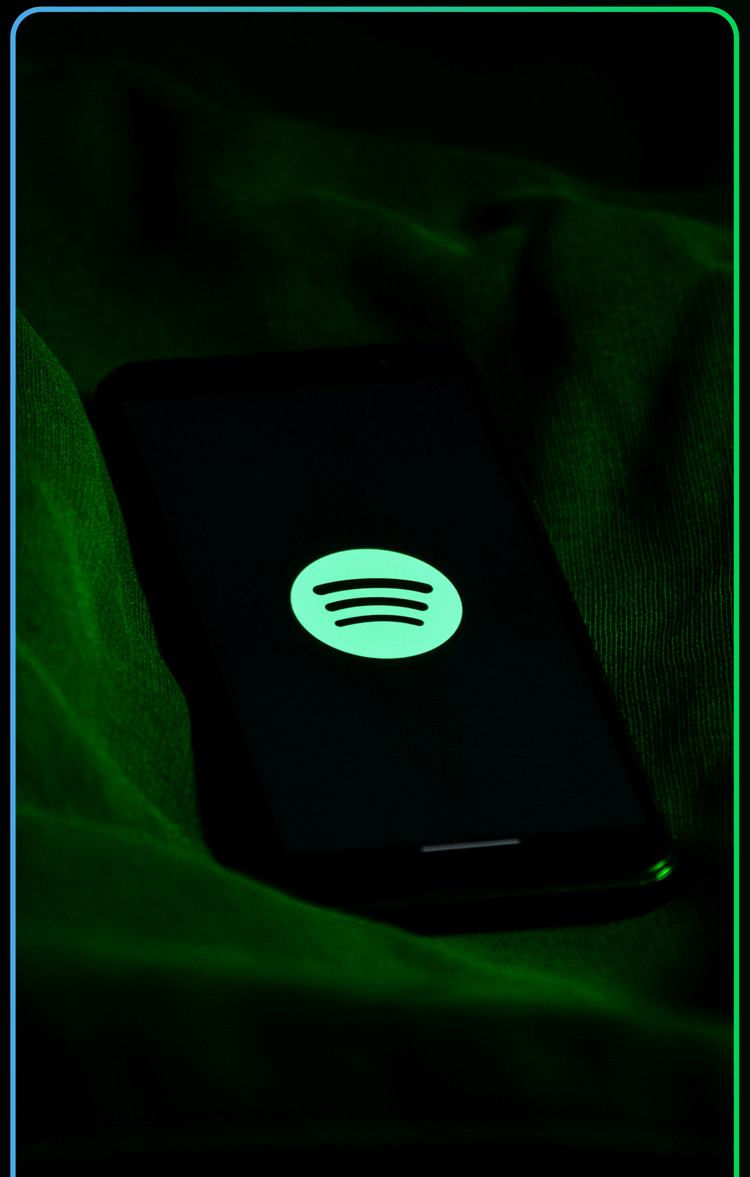
Advanced Level Queries

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

```sql
SELECT
    artist,
    track,
    views,
    likes,
    SUM(likes) OVER (ORDER BY views DESC) AS cumulative_likes
FROM
    spotify
ORDER BY
    Views DESC;
```

	artist character	track character	views double p	likes bigint	cumulative_likes numeric
1	Luis F...	Desp...	807964	50788652	50788652
2	Daddy...	Desp...	807964	50788626	101577278
3	Ed Sh...	Shap...	590839	31047780	132625058
4	Charli...	See Y...	577379	40147674	172772732
5	Wiz K...	See Y...	577379	40147618	212920350

Total rows: 20592 Query complete 00:00:00.256

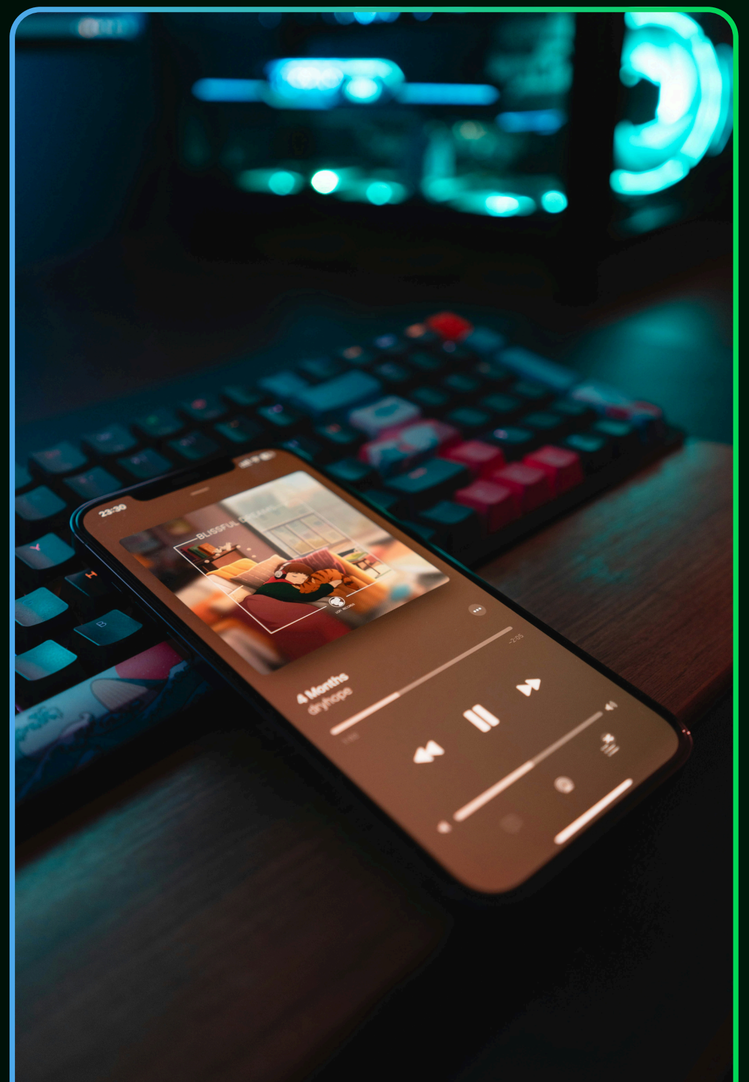


Business Insights & Key Findings

- Artists with the most tracks tend to dominate in both streams and engagement.
- Singles outperform albums in virality metrics like shares and comments.
- Danceability and energy are strong indicators of track popularity.
- Tracks with official videos tend to receive more likes and views.
- Performance tuning (e.g., indexing) dramatically improves response time for large datasets.

Final Conclusion

- PostgreSQL is a powerful tool for structured music data analysis.
- SQL querying can uncover critical business and listener behavior insights.
- Mastering query optimization leads to significant efficiency gains.
- The project successfully combines data engineering and analytical thinking.



Thank You

bhaskarpal.official@gmail.com