**Spotify Advanced SQL Project and Query Optimization**

**Project Category**: Advanced

**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.

**Project Steps**

**1. Data Exploration**

Before diving into SQL, it’s important to understand the dataset thoroughly. The dataset contains attributes such as:

* **Artist**: The performer of the track.
* **Track**: The name of the song.
* **Album**: The album to which the track belongs.
* **Album\_type**: The type of album (e.g., single or album).
* Various metrics such as **danceability**, **energy**, **loudness**, **tempo**, and more.

**2. Querying the Data**

After the data is prepared, various SQL queries can be written to explore and analyze it. Queries are categorized into **easy**, **medium**, and **advanced** levels to help progressively develop SQL proficiency.

* **Easy Queries**: Simple data retrieval, filtering, and basic aggregations.
* **Medium Queries**: More complex queries involving grouping, aggregation functions, and joins.
* **Advanced Queries**: Nested subqueries, window functions, CTEs, and performance optimization.

**3. Query Optimization**

In advanced stages, the focus shifts to improving query performance. Some optimization strategies include:

* **Indexing**: Adding indexes on frequently queried columns.
* **Query Execution Plan**: Using EXPLAIN ANALYZE to review and refine query performance.

**15 Practice Questions**

**Easy Level**

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.

**Medium Level**

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.

**Advanced Level**

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.

**Query Optimization Technique**

To improve query performance, the following optimization process was carried out:

* **Initial Query Performance Analysis**: We began by analyzing the performance of a query using the EXPLAIN function to retrieve tracks based on the artist column. The initial performance metrics were:
  + Execution time (E.T.): **7 ms**
  + Planning time (P.T.): **0.17 ms**
* **Index Creation**: To optimize performance, an index was created on the artist column. This ensures faster retrieval of rows when the artist is queried.
* **Performance Analysis After Index Creation**: After creating the index, we ran the same query again and observed significant improvements in performance:
  + Execution time (E.T.): **0.153 ms**
  + Planning time (P.T.): **0.152 ms**

This optimization shows how indexing can drastically reduce query time, improving the overall performance of database operations.

**Technology Stack**

* **Database**: PostgreSQL
* **SQL Concepts**: DDL, DML, Aggregations, Joins, Subqueries, Window Functions
* **Tools**: pgAdmin 4 (or any SQL editor)

**How to Run the Project**

1. Install PostgreSQL and pgAdmin.
2. Set up the database schema and tables.
3. Insert the sample data into the respective tables.
4. Execute SQL queries to solve the listed problems.
5. Explore query optimization techniques for large datasets.

**Acknowledgments**

The dataset for this project was sourced from [**Kaggle**](https://www.kaggle.com/datasets/sanjanchaudhari/spotify-dataset) and was also highlighted by the YouTube channel [**Zero Analyst**](https://www.youtube.com/@zero_analyst) in one of their project videos. While the dataset idea originates from these sources, all SQL queries for the analysis were written independently for this project.

**Author**

* **Parveen Jalwal**
* **LinkedIn**: <https://www.linkedin.com/in/parveen-jalwal-201a2a302>