SPOTIFY DATA ANALYSIS USING SQL

Data Cleaning and Insights on Song Popularity

INTRODUCTION

This project involves cleaning and analyzing Spotify data from two datasets. The datasets used are high_popularity_spotify_data and low_popularity_spotify_data. Music is universal, and this analysis aims to provide insights that resonate with anyone interested in Spotify's rich database of songs and artists.

DATASET OVERVIEW

Datasets Used:

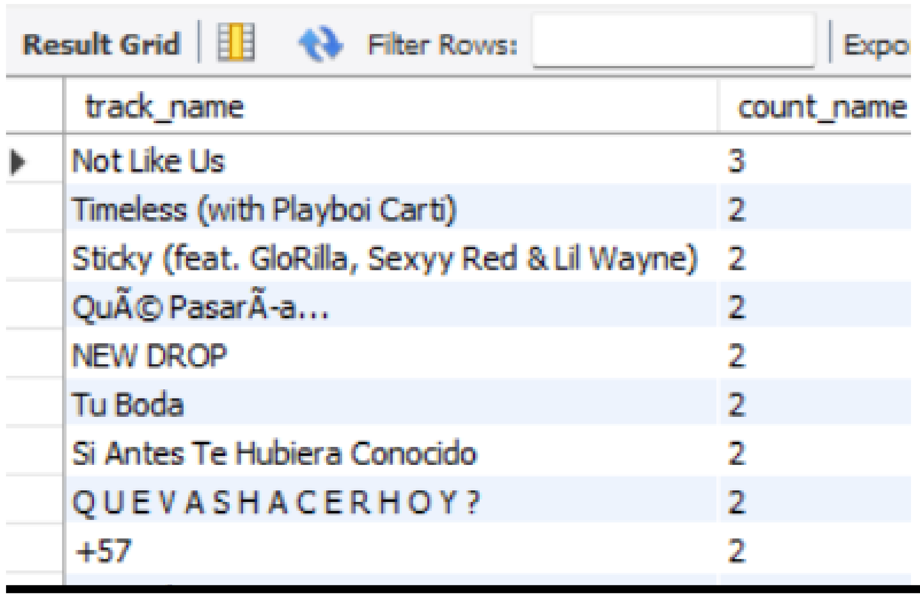
1. High Popularity Spotify Data
2. Low Popularity Spotify Data

Key Columns:
track_name (Song Name)
track_artist (Artist Name)
track_popularity (Popularity Score)
track_album_release_date (Release Date)
energy, danceability, key, etc.

STEP 1 - DATA CLEANING

Removing Duplicates

```
 SELECT
     track_name, COUNT(*) AS count_name
FROM
     music.high_popularity_spotify_data
GROUP BY track_name
HAVING COUNT(*) > 1
ORDER BY count_name DESC;
```

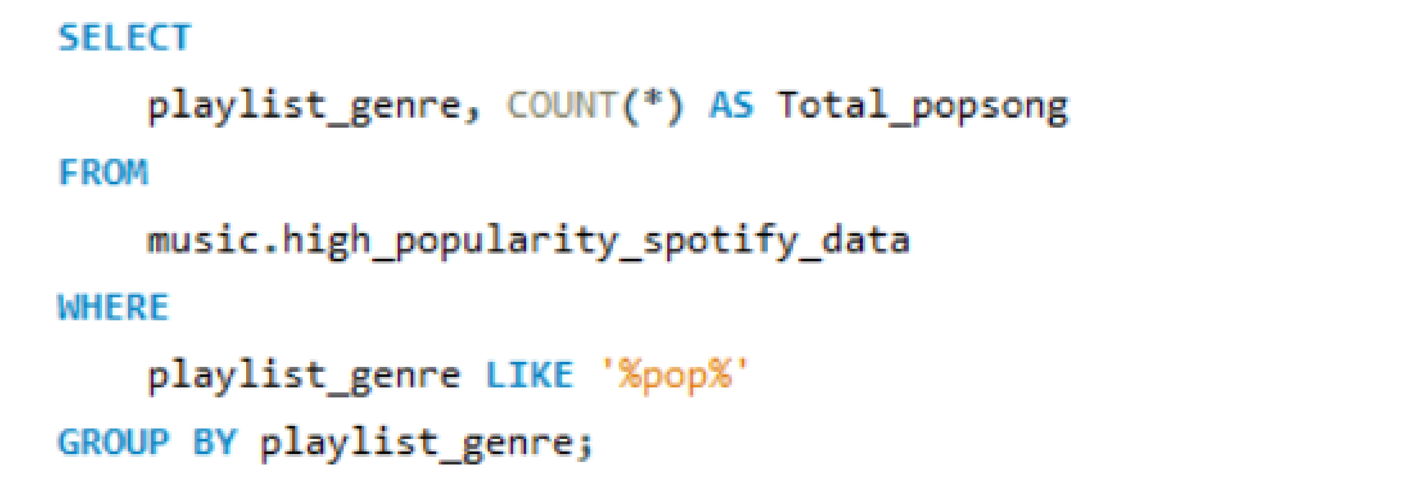


track_name	count_name
Not Like Us	3
Timeless (with Playboi Carti)	2
Sticky (feat. GloRilla, Sexyy Red & Lil Wayne)	2
QuÃ© PasarÃ-a...	2
NEW DROP	2
Tu Boda	2
Si Antes Te Hubiera Conocido	2
Q U E V A S H A C E R H O Y ?	2
+57	2

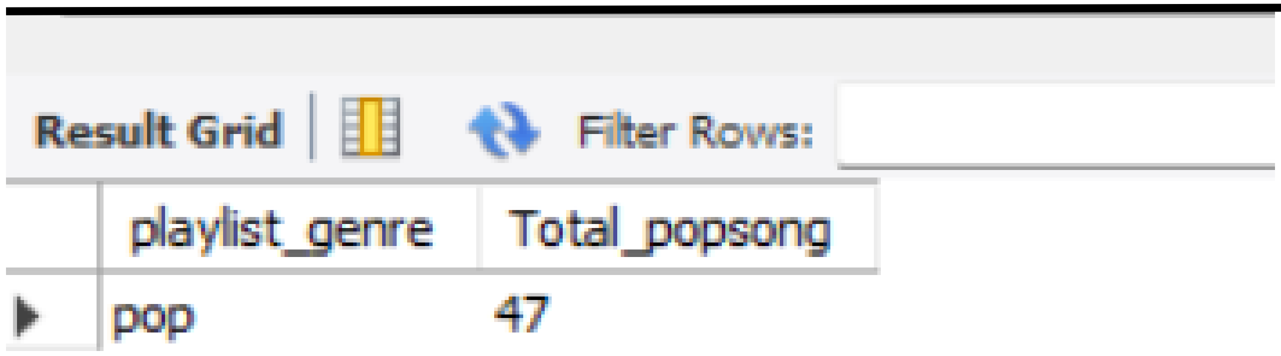
STEP 2 -DATA ANALYSIS

Basic Queries

1. Find the total number of Pop songs in the High Popularity dataset

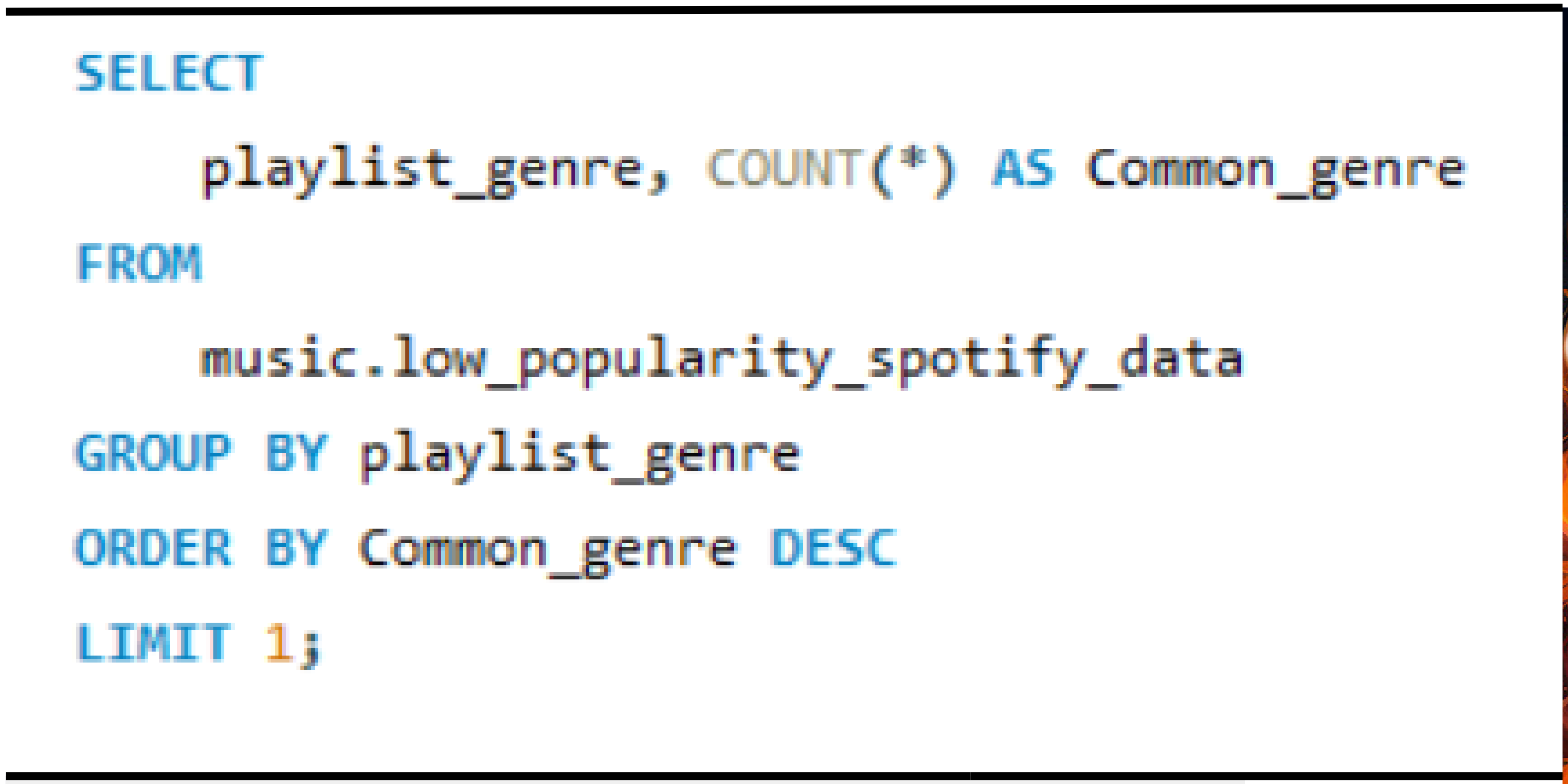


```
SELECT
    playlist_genre, COUNT(*) AS Total_popsong
FROM
    music.high_popularity_spotify_data
WHERE
    playlist_genre LIKE '%pop%'
GROUP BY playlist_genre;
```

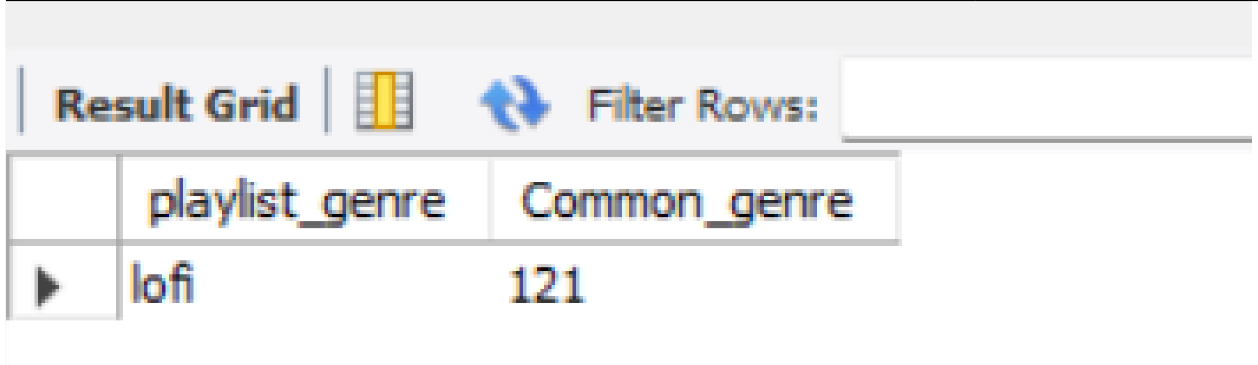


```
  playlist_genre   Total_popsong
▶ pop              47
```

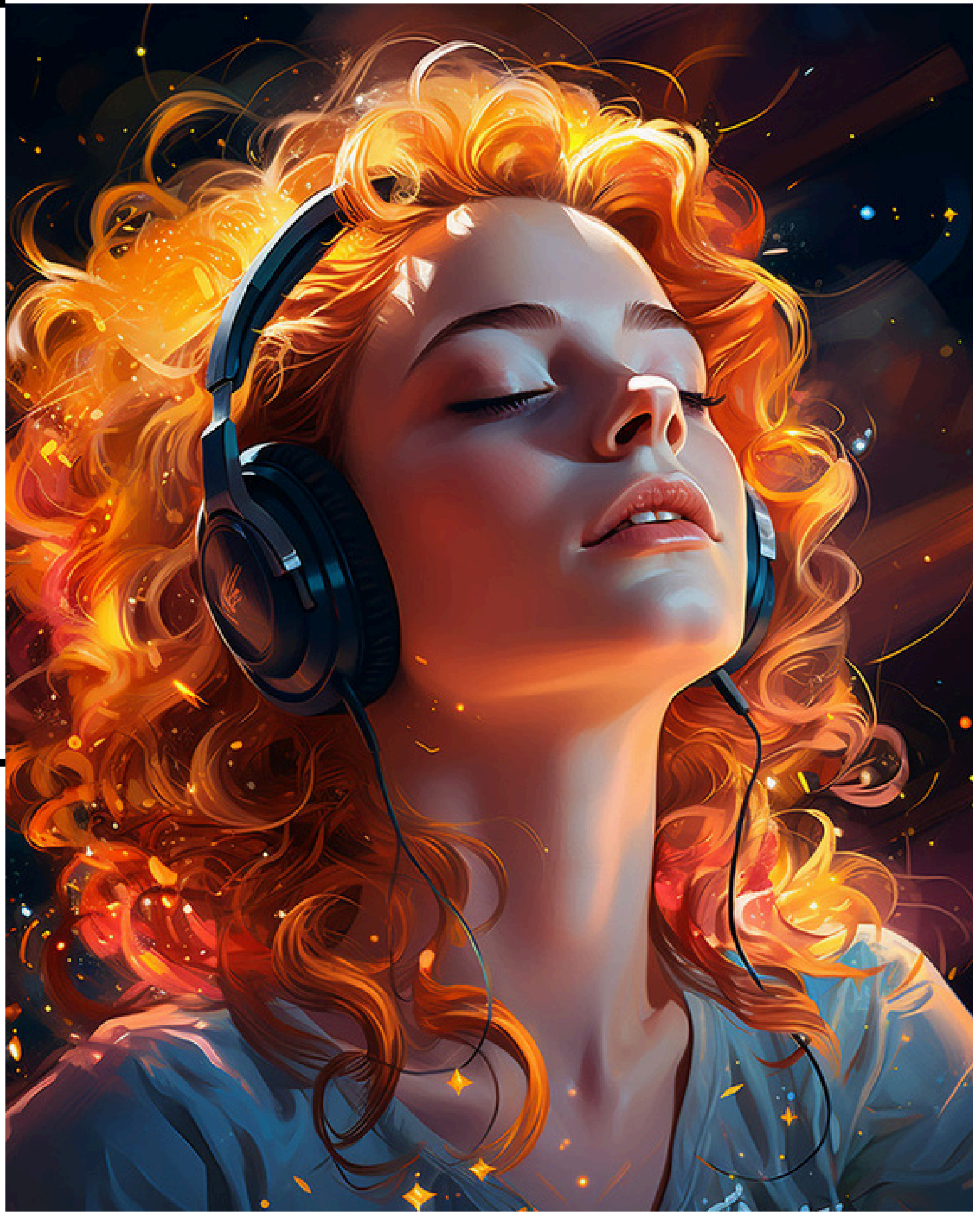
2. Identify the most common genre in the Low Popularity dataset



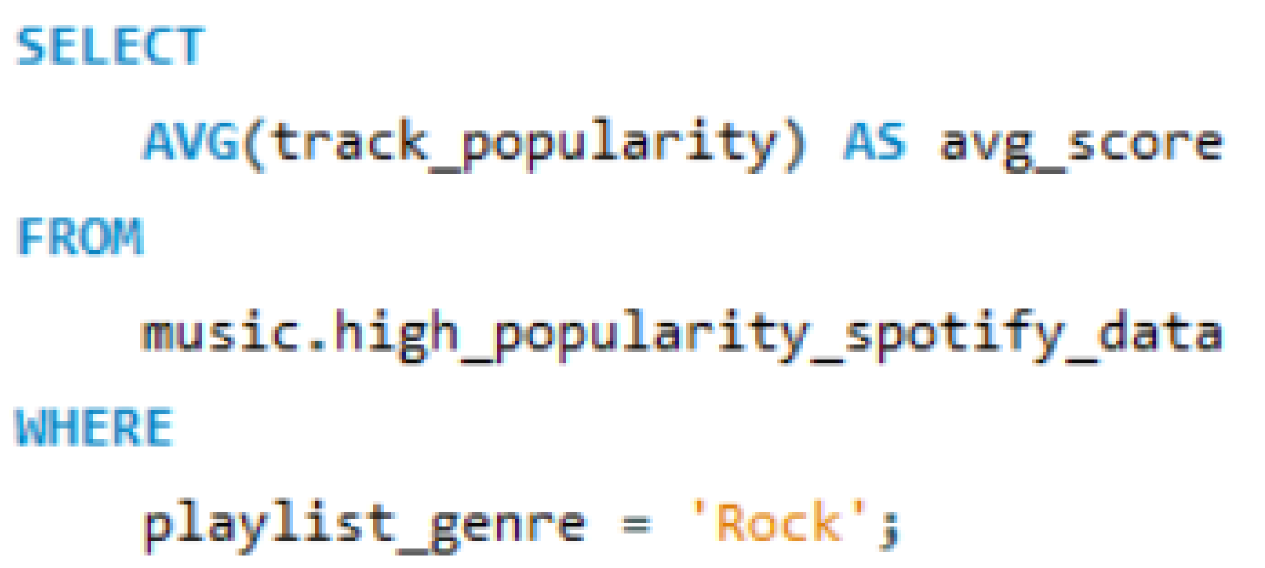
```
SELECT
    playlist_genre, COUNT(*) AS Common_genre
FROM
    music.low_popularity_spotify_data
GROUP BY playlist_genre
ORDER BY Common_genre DESC
LIMIT 1;
```



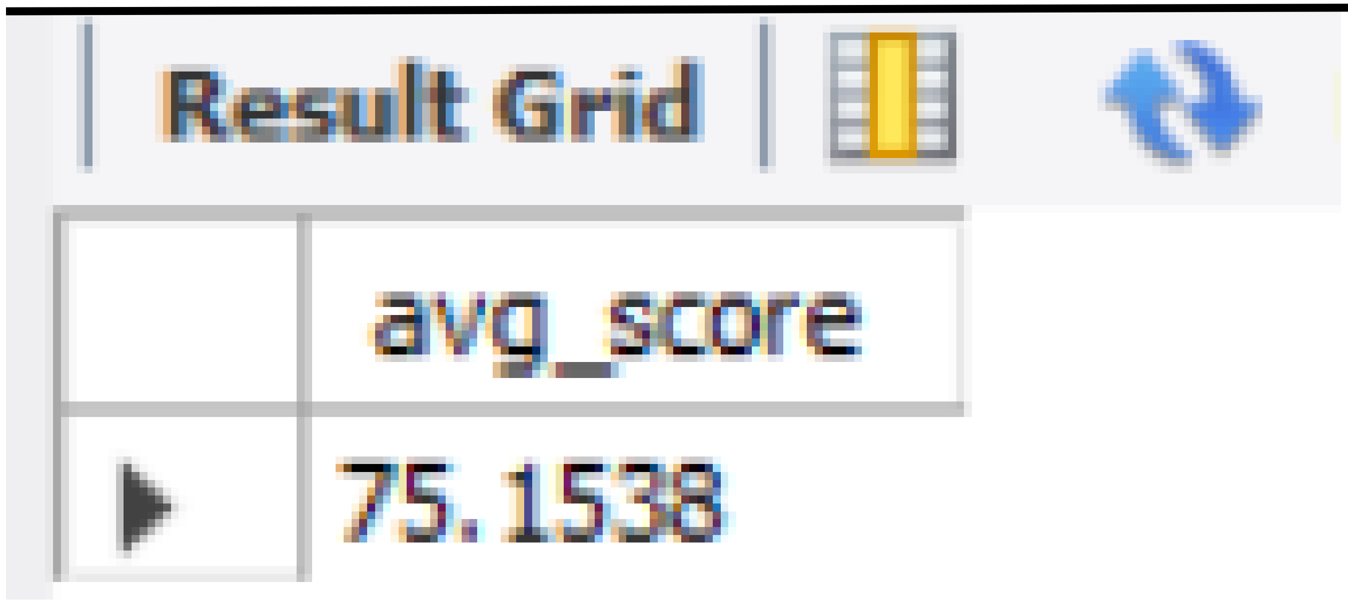
```
   playlist_genre   Common_genre
▶  lofi             121
```



3. Calculate the average popularity score of Rock song



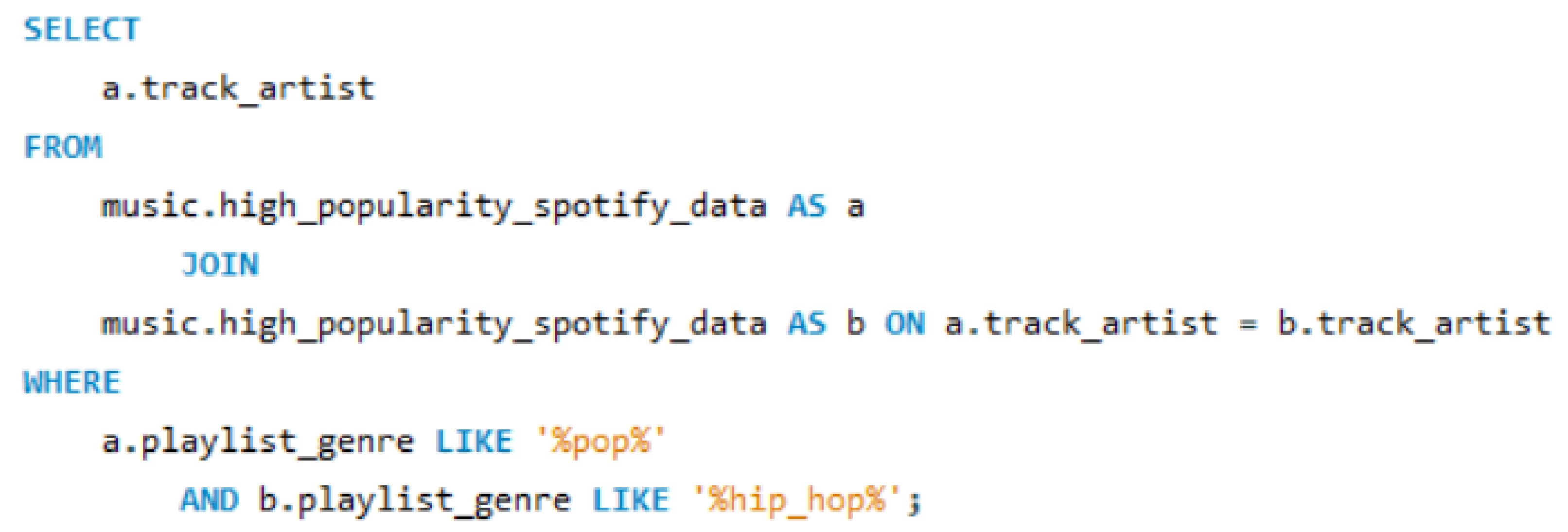
```sql
SELECT
    AVG(track_popularity) AS avg_score
FROM
    music.high_popularity_spotify_data
WHERE
    playlist_genre = 'Rock';
```



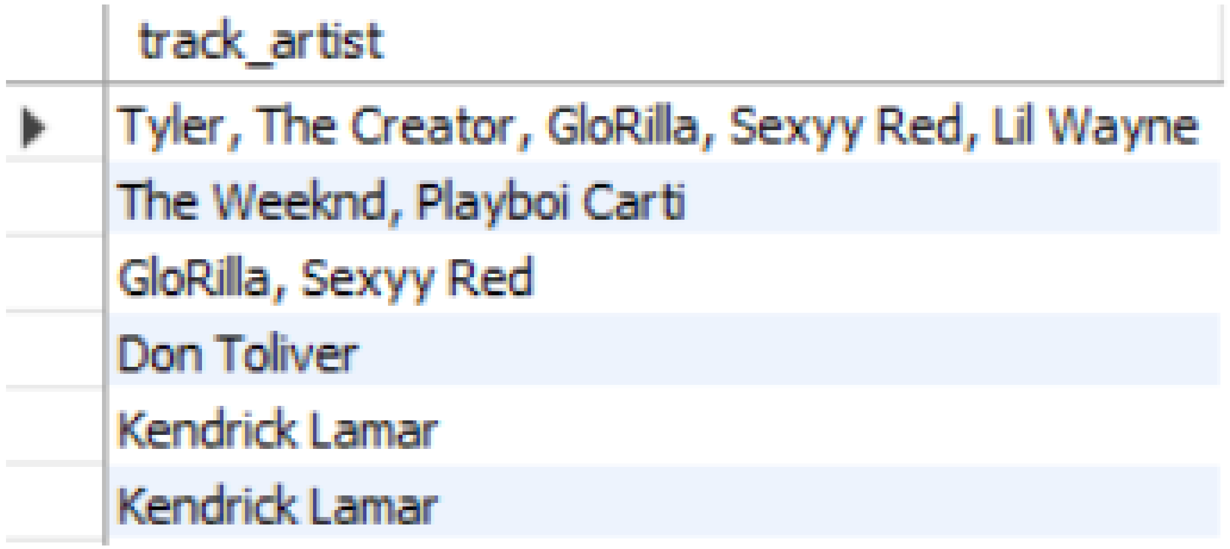
	avg_score
▶	75.1538

Intermediate Queries

1. List artists who have produced songs in both Pop and hip-hop genres



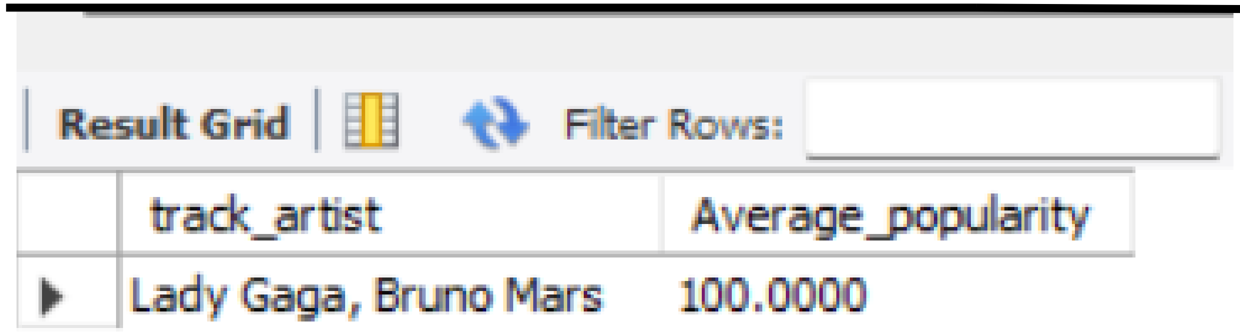
```
SELECT
    a.track_artist
FROM
    music.high_popularity_spotify_data AS a
        JOIN
    music.high_popularity_spotify_data AS b ON a.track_artist = b.track_artist
WHERE
    a.playlist_genre LIKE '%pop%'
        AND b.playlist_genre LIKE '%hip_hop%';
```



track_artist
Tyler, The Creator, GloRilla, Sexyy Red, Lil Wayne
The Weeknd, Playboi Carti
GloRilla, Sexyy Red
Don Toliver
Kendrick Lamar
Kendrick Lamar

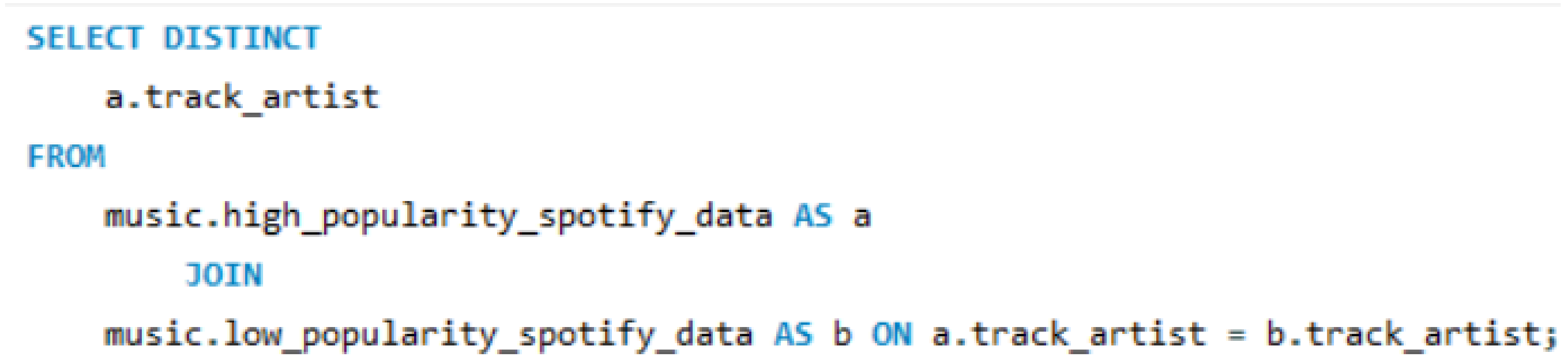
2.Find Artists with the Highest Average Popularity Across Both Tables

```sql
SELECT
    track_artist, AVG(track_popularity) AS Average_popularity
FROM
    (SELECT
        track_artist, track_popularity
    FROM
        music.high_popularity_spotify_data UNION ALL SELECT
        track_artist, track_popularity
    FROM
        music.low_popularity_spotify_data) AS Combined_data
GROUP BY track_artist
ORDER BY Average_popularity DESC
LIMIT 1;
```

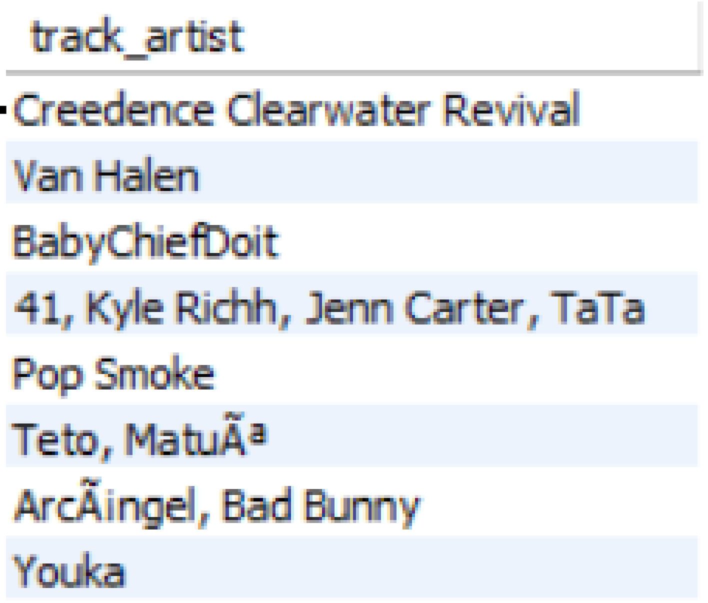


track_artist	Average_popularity
Lady Gaga, Bruno Mars	100.0000

3 .Find the Overlapping Artists in Both Datasets

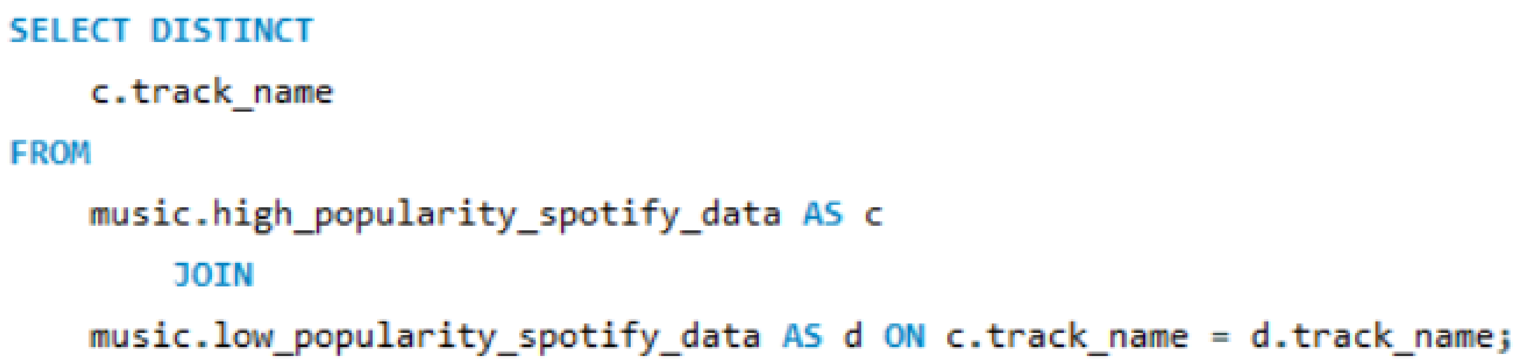


```
SELECT DISTINCT
    a.track_artist
FROM
    music.high_popularity_spotify_data AS a
        JOIN
    music.low_popularity_spotify_data AS b ON a.track_artist = b.track_artist;
```

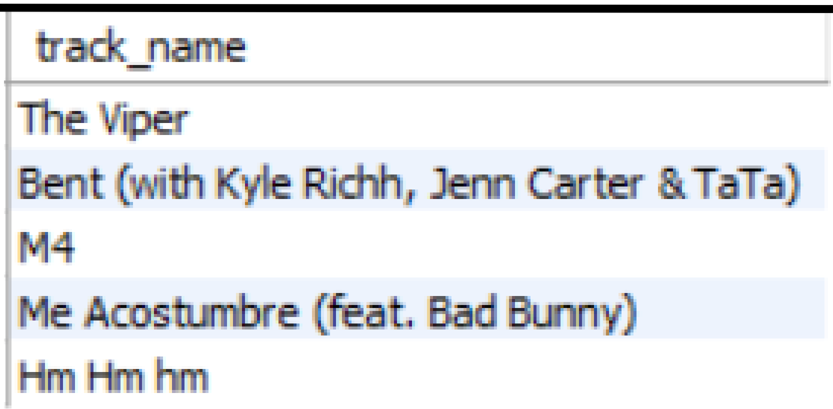


track_artist
Creedence Clearwater Revival
Van Halen
BabyChiefDoit
41, Kyle Richh, Jenn Carter, TaTa
Pop Smoke
Teto, MatuÃª
ArcÃingel, Bad Bunny
Youka

4 .list songs that appear in both high and low popularity datasets

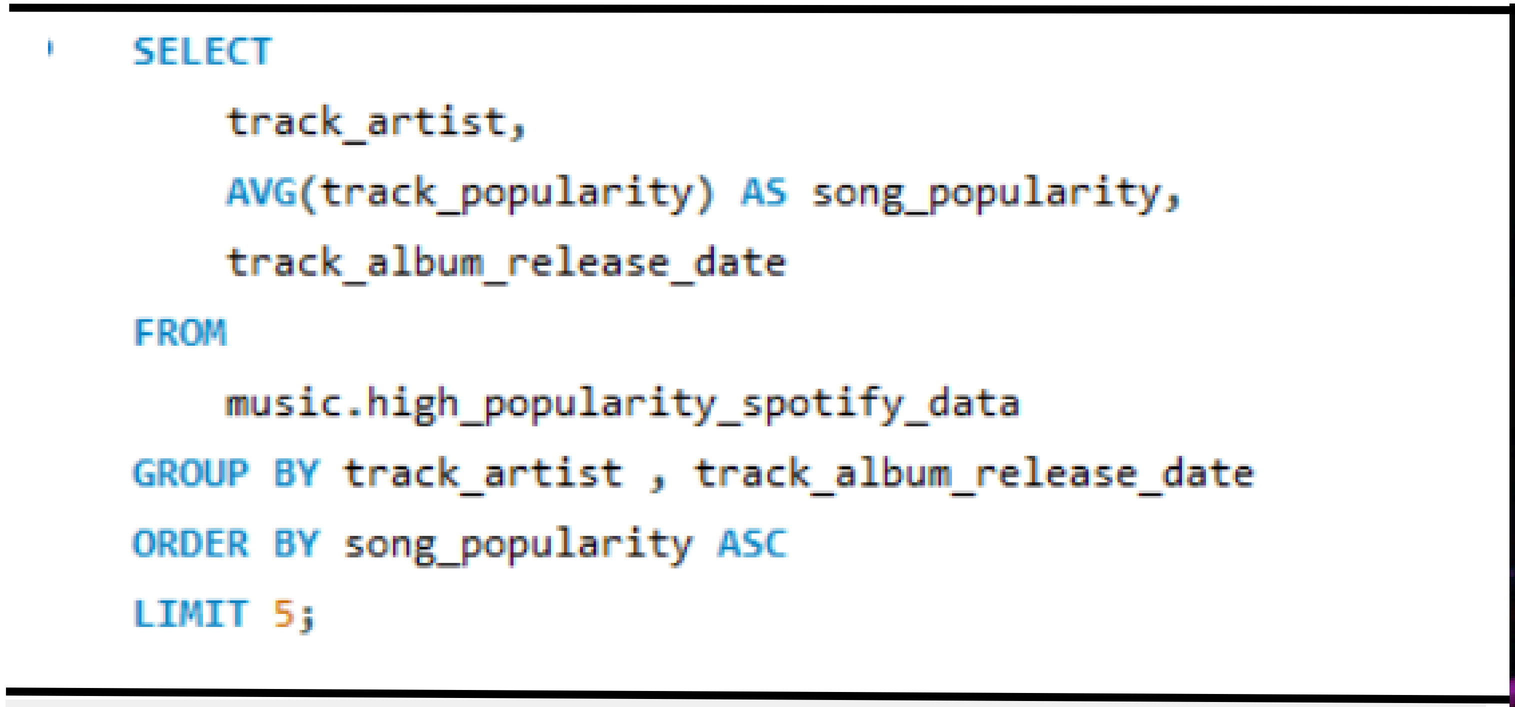


```
SELECT DISTINCT
    c.track_name
FROM
    music.high_popularity_spotify_data AS c
        JOIN
    music.low_popularity_spotify_data AS d ON c.track_name = d.track_name;
```

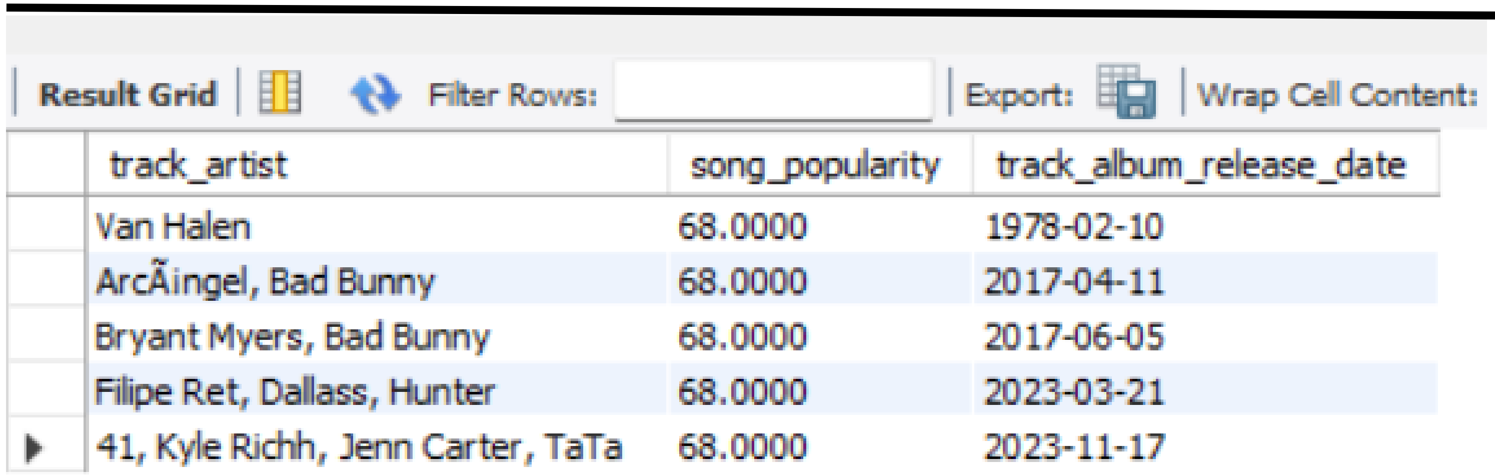


track_name
The Viper
Bent (with Kyle Richh, Jenn Carter & TaTa)
M4
Me Acostumbre (feat. Bad Bunny)
Hm Hm hm

5 .identify artists who improved their average song popularity over time



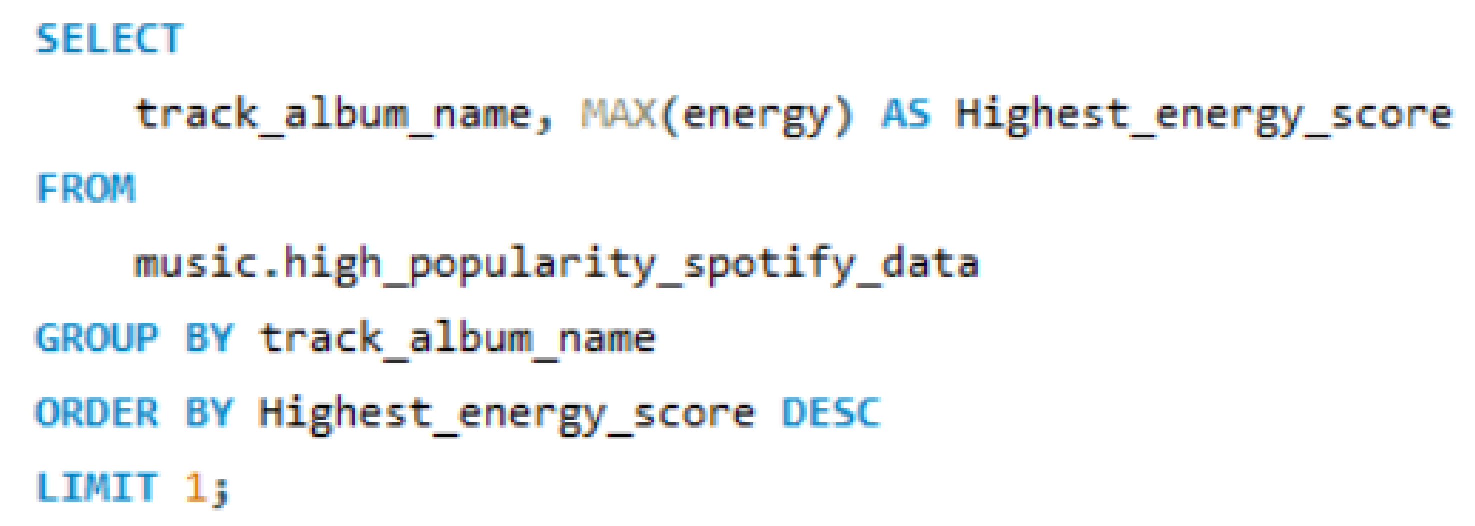
```
SELECT
    track_artist,
    AVG(track_popularity) AS song_popularity,
    track_album_release_date
FROM
    music.high_popularity_spotify_data
GROUP BY track_artist , track_album_release_date
ORDER BY song_popularity ASC
LIMIT 5;
```



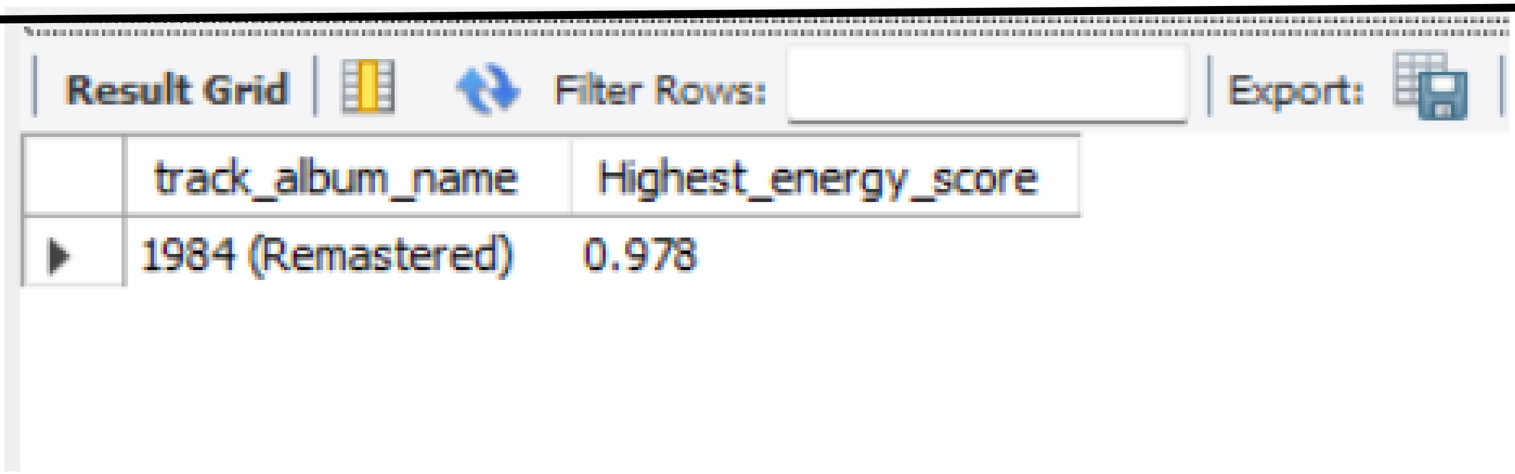
Result Grid | Filter Rows: | Export: | Wrap Cell Content:

track_artist	song_popularity	track_album_release_date
Van Halen	68.0000	1978-02-10
ArcÃ¡ngel, Bad Bunny	68.0000	2017-04-11
Bryant Myers, Bad Bunny	68.0000	2017-06-05
Filipe Ret, Dallass, Hunter	68.0000	2023-03-21
41, Kyle Richh, Jenn Carter, TaTa	68.0000	2023-11-17

6 .Which song has the highest energy score in the High Popularity dataset



```
SELECT
    track_album_name, MAX(energy) AS Highest_energy_score
FROM
    music.high_popularity_spotify_data
GROUP BY track_album_name
ORDER BY Highest_energy_score DESC
LIMIT 1;
```



Result Grid | Filter Rows: | Export:

	track_album_name	Highest_energy_score
▶	1984 (Remastered)	0.978

Conclusion

This portfolio demonstrates the process of cleaning, transforming, and analyzing Spotify data. By asking engaging and relatable questions, we provide insights into popular genres, and artists' performance. The queries are designed to appeal to a broad audience, showcasing the power of SQL in analyzing real-world datasets. Whether you're a music enthusiast or a data professional, these findings offer a glimpse into the fascinating world of Spotify data.