Data Analysis on Spotify Tracks on 2022-23

SQL project to analyze and answer interesting questions about Spotify. The data used in this project is provided by Spotify collecting from Kaggle. It contains about

1.47 Lakh records from January 2022 to May 2023.

**BUSINESS PROBLEMS**

**Q1. Give me your Top 10 Favorite Songs by Daily Ranking**

**Q2. Give me Top 10 Favorite Artist Name by Daily Ranking**

**Q3. Return Top 10 Song which I can recommend as Fitness Freak Q4. Return Top 5 Emerging artist of 2022 to current date**

**Q5. Recommend Top 10 Song for the Weekend**

**Q6. Return every week & respective trending song for that week**

**Q7. Return the Artist Name and 'Count of days' when they have two or more two songs in the top 200 song list.**

**Q8. Check if there is any association between No. of artist per song and Popularity of songs**

**Q9. Provide No. of artists in each continent and respective country**

**Q.10. Provide no. of songs belonging to each continent and country produced by individual singers**

**Raw Data –** [**Link**](https://drive.google.com/drive/folders/1R0rRAtyt83Lhw80aVMkel5vQEcRQr9L7?usp=drive_link) **Clean Data –** [**Link**](https://drive.google.com/drive/folders/1R0rRAtyt83Lhw80aVMkel5vQEcRQr9L7?usp=drive_link)

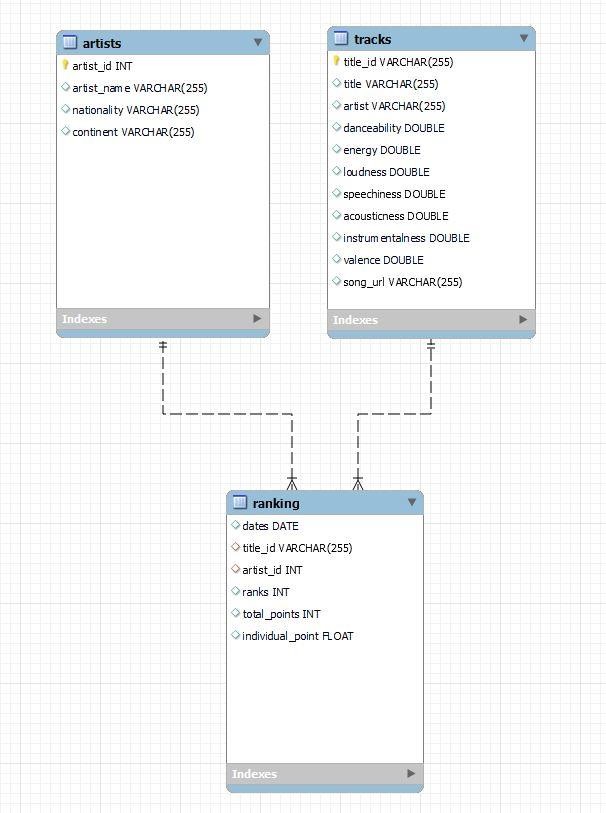
**Data Dictionary**

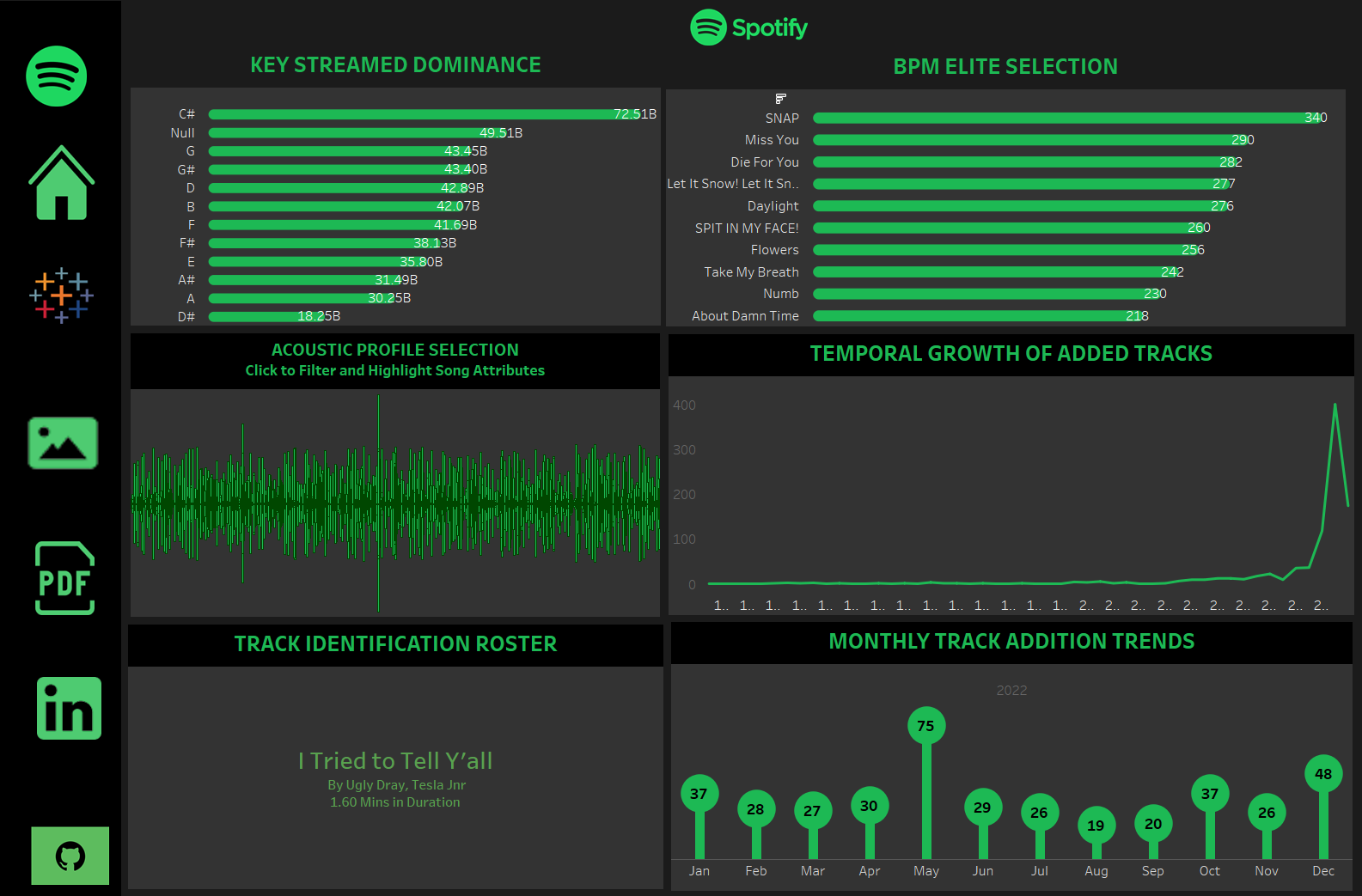
The dataset encapsulates 21 information-rich columns, meticulously cleaned and primed for analysis. The possibilities for insightful analysis utilizing this dataset are extensive.

This data set consists of about 181K records.

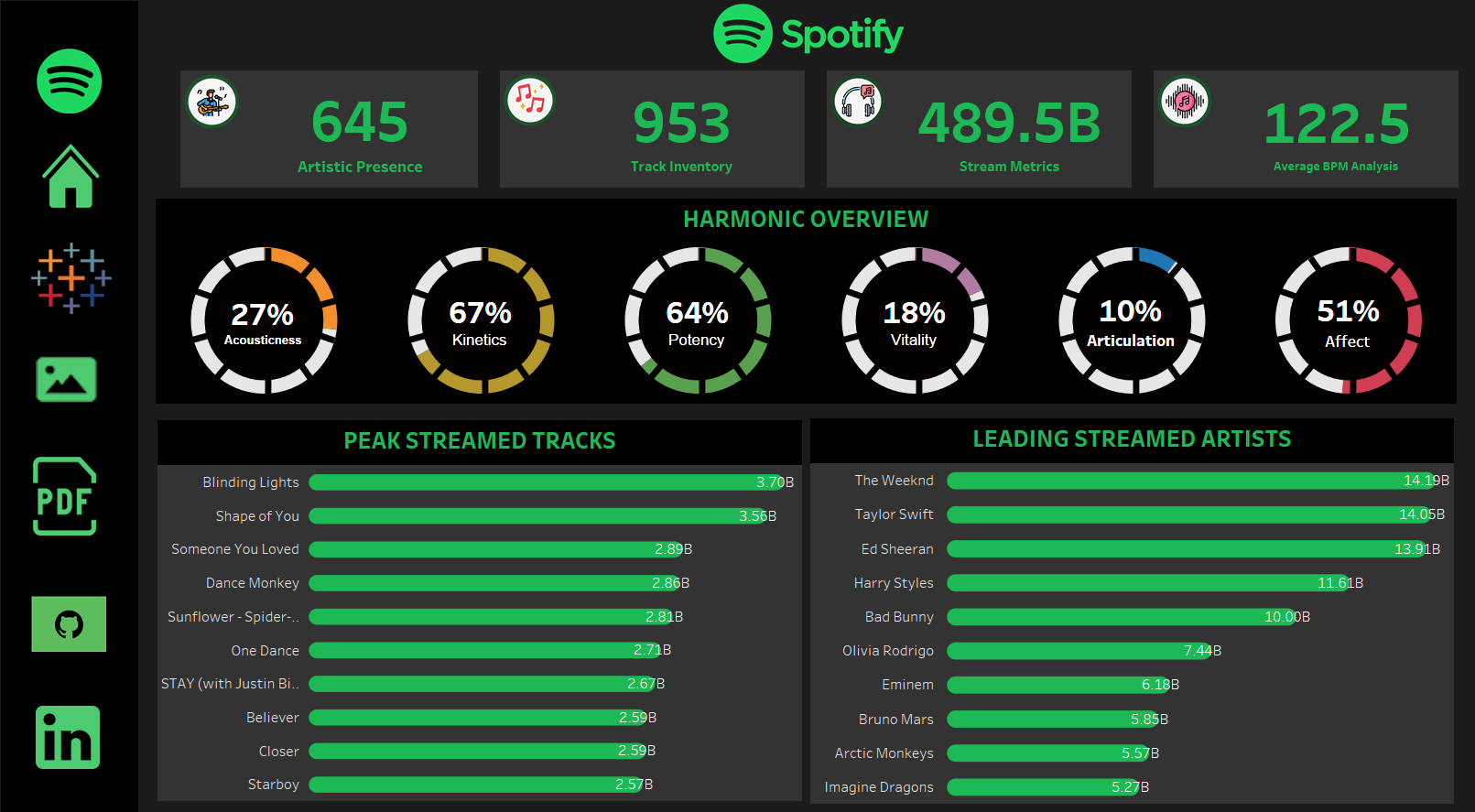
* **Song Rank:** Rank of that song out of 200 that day
* **Title of the Song:** Song name
* **Artist Name:** Singer of that song
* **Date:** Date consist of 1st Jan 2022 to 31st May 2023
* **Nationality of each artist:** From which Nation that Artist belongs
* **Artist's Continent:** From which Continent that Artist belongs
* **Unique Song ID:** Every song has a unique Song ID
* **URL Link to the Song:** Direct link of that song from Spotify
* **Danceability:** Describes how suitable a track is for dancing
* **Energy:** Represents a perceptual measure of intensity and activity. Energetic tracks feel fast, loud, and noisy.
* **Loudness:** The overall loudness of a track in decibels (dB)
* **Speechiness:** Detects the presence of spoken words in a track.
* **Acousticness:** Describes whether a track uses only or primarily instruments that produce sound through acoustic means.
* **Instrumentalness:** Predicts whether a track contains no vocals.
* **Valence:** Describes the musical positiveness of a track

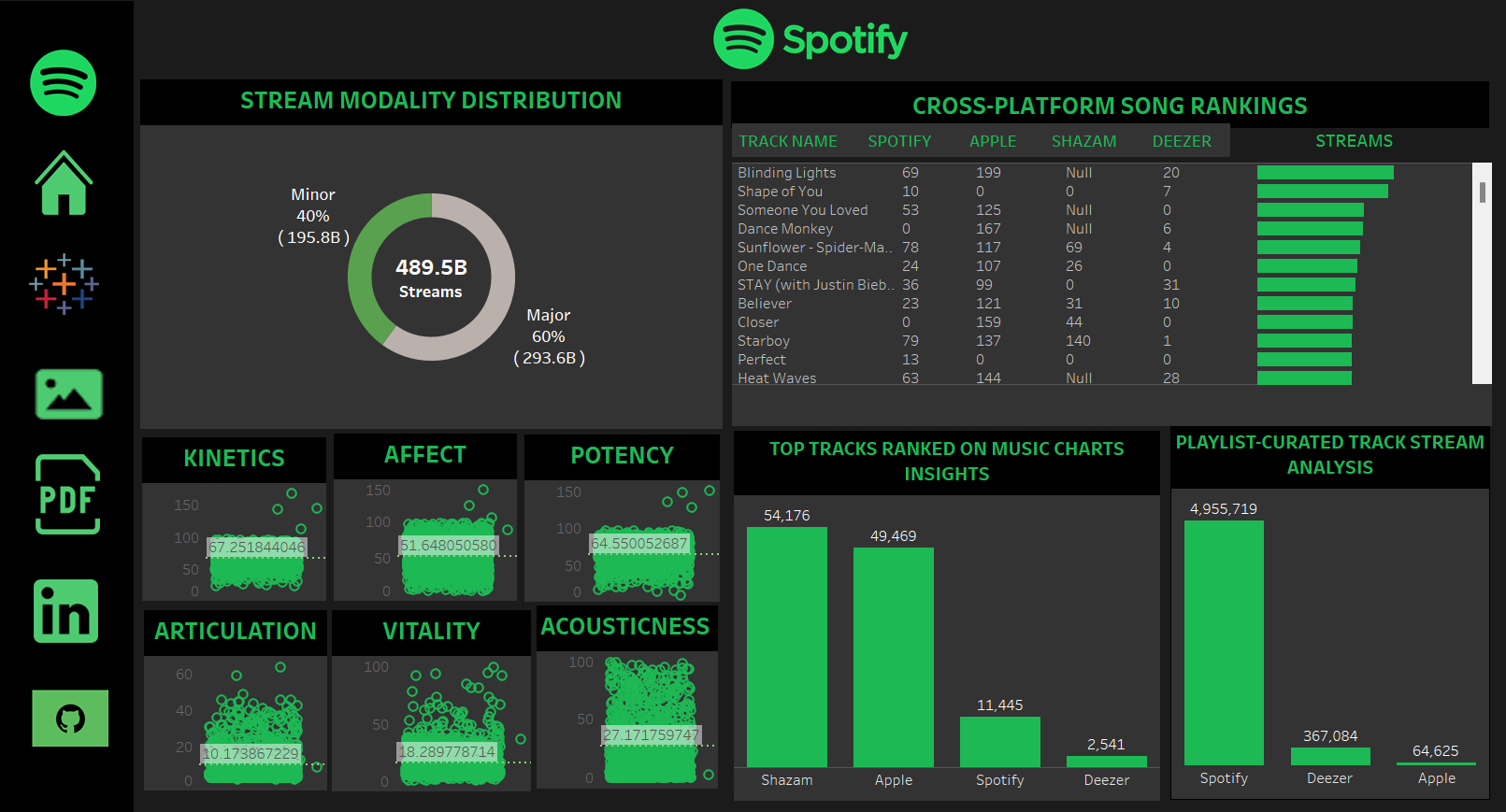
**SPOTIFY STAR SCHEMA**

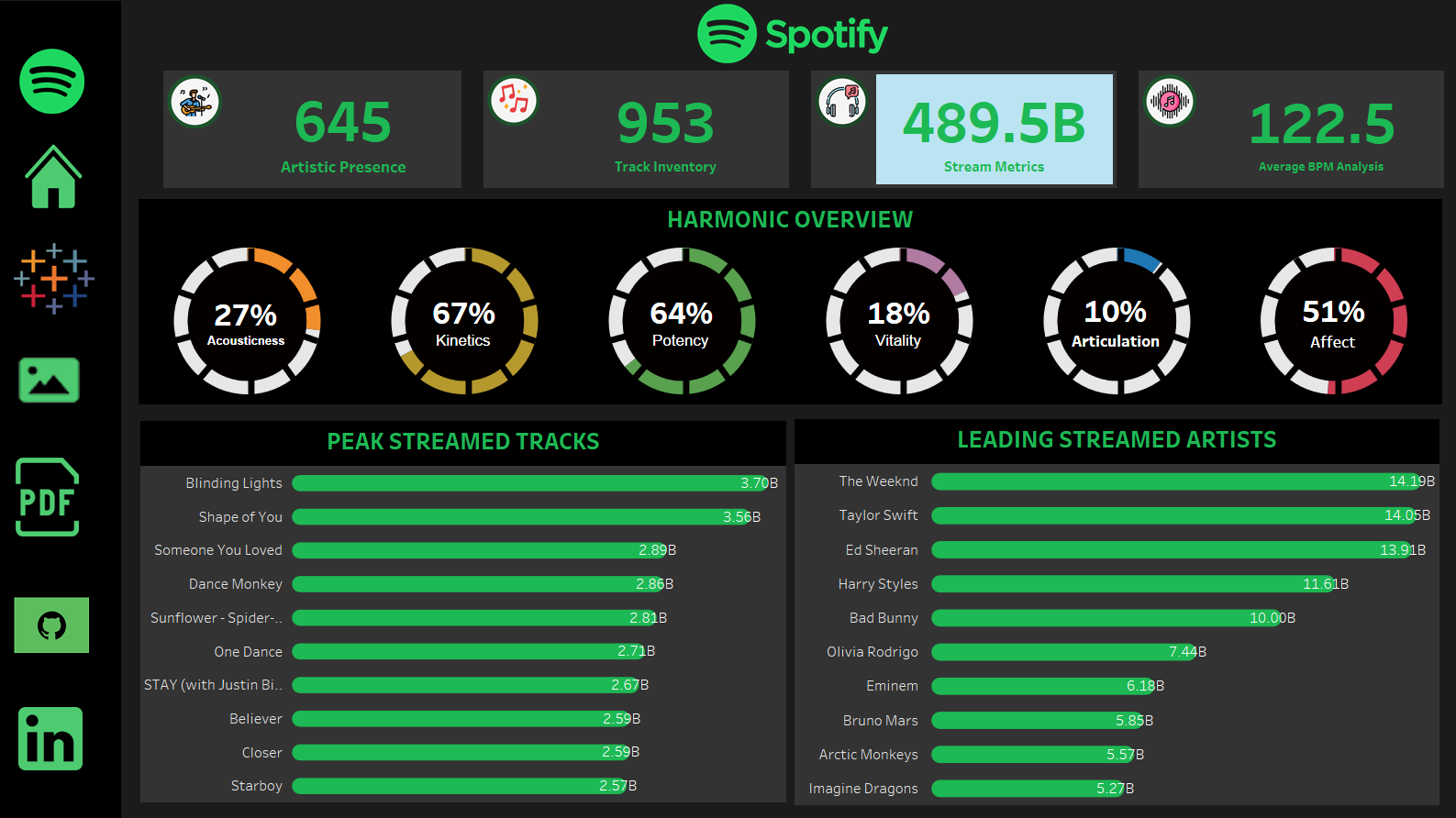


TABLEAU DATA VIZZES

1. Audio Analytics

2. Holistic Insights

3. Sonic Overview

****

4. Stream Metrics

**NORMALISED DATA IN TABLES FORMS**

**CREATING Song TABLE**

CREATE TABLE tracks (

title\_id VARCHAR(255) PRIMARY KEY, title VARCHAR(255),

artist VARCHAR(255), danceability FLOAT(25), energy FLOAT(25), loudness FLOAT(25), speechiness FLOAT(25), acousticness FLOAT(25),

instrumentalness FLOAT(25), valence FLOAT(25),

song\_url VARCHAR(255)

);

**CREATING artist TABLE**

CREATE TABLE artists( artist\_id int PRIMARY KEY, artist\_name VARCHAR(255), nationality VARCHAR(255), continent VARCHAR(255)

);

**CREATING ranking TABLE**

CREATE TABLE ranking( dates DATE,

title\_id VARCHAR(255), artist\_id INT,

ranks INT, total\_points INT,

individual\_point FLOAT,

FOREIGN key (title\_id) REFERENCES tracks(title\_id), FOREIGN key (artist\_id) REFERENCES artists(artist\_id)

);

# **SQL QUERIES**

**Q1. Give me Top 10 Favourite Song by Daily Ranking**

WITH cte AS ( SELECT

t.title,

COUNT(DISTINCT r.dates) AS song\_of\_the\_day FROM

ranking r

JOIN tracks t ON r.title\_id = t.title\_id WHERE

r.ranks = 1 GROUP BY

1

# ORDER BY

2 desc LIMIT

10

# ) SELECT

title FROM

cte;

**Q2. Give me Top 10 Favourite Artist Name by Daily Ranking**

WITH cte AS ( SELECT

a.artist\_name,

COUNT(DISTINCT r.dates) -- COUNT(DISTINCT r.dates) AS

song\_of\_the\_day FROM

ranking r

JOIN artists a ON a.artist\_id = r.artist\_id WHERE

r.ranks = 1 GROUP BY

1

# ORDER BY

2 desc LIMIT

10

# ) SELECT

artist\_name FROM

cte;

**Q3. Return Top 10 Song which I can recommend as Fitness freak**

# SELECT

t.title,

COUNT(DISTINCT r.dates),

MIN(r.ranks)

# FROM

ranking r

JOIN tracks t ON r.title\_id = t.title\_id WHERE

energy > 0.8

AND loudness > -500 GROUP BY

1

# ORDER BY

2 desc,

3

# LIMIT

10;

**Q4. Return Top 5 Emerging artist of 2022 to current date**

CREATE TEMPORARY TABLE monthly\_artists\_points SELECT

DATE\_FORMAT(r.dates, '%Y-%m') AS yearmonth, r.artist\_id,

SUM(r.individual\_point) AS points FROM

ranking r GROUP BY 1, 2

# ORDER BY 1 ASC, 3 DESC; SELECT

sub.yearmonth, sub.artist\_id, a.artist\_name

# FROM(

SELECT

yearmonth, artist\_id, points, RANK() OVER(

PARTITION BY yearmonth ORDER BY points desc

) AS rnk FROM

monthly\_artists\_points

) sub

JOIN artists a ON sub.artist\_id = a.artist\_id WHERE

rnk = 1;

**Q5. Recommend Top 10 Song for Weekend**

SELECT t.title

# FROM (

SELECT

title\_id,

COUNT(title\_id) AS counts, SUM(total\_points) AS all\_points

# FROM(

SELECT

CASE WHEN date\_format(dates, '%W') IN ('Saturday', 'Sunday') THEN "Yes"

ELSE "No"

END AS weekend, title\_id, total\_points

FROM ranking WHERE

# CASE

WHEN date\_format(dates, '%W') IN ('Saturday', 'Sunday') THEN "Yes"

ELSE "No" END = "Yes"

) sub GROUP BY 1

ORDER BY 3 desc

# LIMIT 10

) sub2

JOIN tracks t ON sub2.title\_id = t.title\_id;

**Q6. Return every week & respective trending song for that week**

# SELECT

start\_of\_week, t.title

FROM (

SELECT

\* FROM (

SELECT

\*,

# RANK() OVER(

PARTITION BY yearweek ORDER BY

total\_points desc

) AS rnk FROM

(

# SELECT

yearweek, title\_id,

MIN(dates) start\_of\_week, AVG(all\_points) AS total\_points

# FROM

(

# SELECT

\*,

SUM(total\_points) OVER( PARTITION BY yearweek, title\_id

# ORDER BY

yearweek

) AS all\_points FROM

(

concat('0', week(dates))

# SELECT

\*, concat(

YEAR(dates), '-',

# CASE

WHEN week(dates) < 10 THEN

ELSE week(dates) END

) AS yearweek FROM

(

# SELECT

DISTINCT dates, title\_id,

ranks,

total\_points FROM

ranking WHERE

dates > '2022-01-01'

AND dates <= '2023-05-27' ORDER BY

1,

3

) sub

) sub2

) sub3 GROUP BY

yearweek, title\_id

# ORDER BY 1,

2,

4 desc

) sub4

) sub5 WHERE

rnk = 1

) sub6

JOIN tracks t ON sub6.title\_id = t.title\_id ORDER BY

1;

**Q7. Return Artist Name and 'Count of days' when they have two or more than two songs in top 200 song list.**

# SELECT

a.artist\_name, COUNT(sub.dates) AS days

# FROM (

SELECT

dates, artist\_id,

COUNT(title\_id) AS cnt FROM

ranking GROUP BY

1,

2

# ORDER BY 1,

2

) sub

JOIN artists a ON sub.artist\_id = a.artist\_id WHERE

sub.cnt >= 2 GROUP BY

1

# ORDER BY

2 desc;

**Q8. Check if there any association between No. of artist per song and Popularity of songs**

SELECT

SUM(

CASE

WHEN cnt = 1 THEN 1

# ELSE 0 END

) AS '#of\_artists1', SUM(

# CASE

WHEN cnt = 2 THEN 1

# ELSE 0 END

) AS '#of\_artists2', SUM(

# CASE

WHEN cnt = 3 THEN 1

# ELSE 0 END

) AS '#of\_artists3', SUM(

# CASE

WHEN cnt = 4 THEN 1

# ELSE 0 END

) AS '#of\_artists4', SUM(

# CASE

WHEN cnt = 5 THEN 1

# ELSE 0 END

) AS '#of\_artists5', SUM(

# CASE

WHEN cnt = 6 THEN 1

# ELSE 0 END

) AS '#of\_artists6' FROM

(

# SELECT

dates, title\_id,

COUNT(artist\_id) AS cnt FROM

ranking WHERE

ranks <= 10 -- 5% of GROUP BY

1,

2

# HAVING

COUNT(artist\_id)

# ORDER BY 1

) sub;

**Q9. Provide No. of artist in each continent and respective country**

SELECT

continent, nationality,

COUNT(artist\_id) AS no\_of\_artists FROM

artists WHERE

continent <> 'Unknown' GROUP BY

1,

2

ORDER BY

1 asc,

2 asc;

**Q.10. Provide no. of songs belongs to each continent and country produced by individual singers**

# SELECT

continent, nationality,

COUNT(t.title\_id) AS no\_of\_songs FROM

tracks t

LEFT JOIN ranking r ON t.title\_id = r.title\_id LEFT JOIN artists a ON r.artist\_id = a.artist\_id

# WHERE

a.continent IS NOT NULL

AND a.nationality IS NOT NULL AND a.continent <> 'Unknown' AND a.nationality <> 'Unknown'

# GROUP BY 1,

2

# ORDER BY 1,

3 desc;