

Data Analysis on Spotify tracks on 2022-23

SQL project to analysed 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 Top 10 Favourite Song by Daily Ranking**
- Q2. Give me Top 10 Favourite 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 Weekend**
- Q6. Return every week & respective trending song for that week**
- Q7. Return Artist Name and 'Count of days' when they have two or more than two songs in top 200 song list.**
- Q8. Check if there any association between No. of artist per song and Popularity of songs**
- Q9. Provide No. of artist in each continent and respective country**
- Q.10. Provide no. of songs belongs to each continent and country produced by individual singers**

Raw Data – [Link](#)

Clean Data – [Link](#)

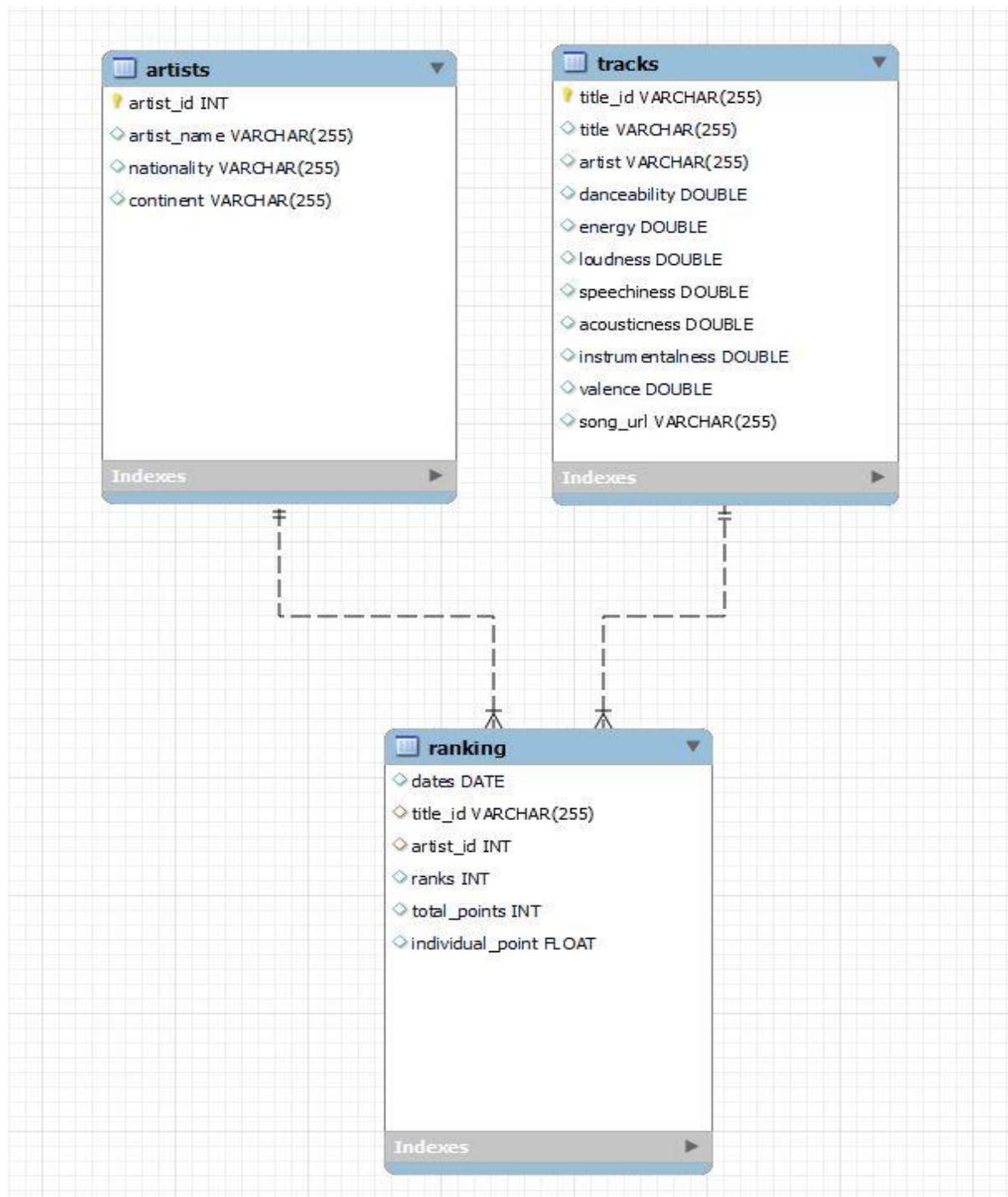
Data Dictionary

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 consist 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 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



NORMALIZED 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)  
);
```

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
            ) AS rk
        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
        (
            SELECT
                *,
                concat(
                    YEAR(dates),
                    '-',
                    CASE
                        WHEN week(dates) < 10 THEN
concat('0', week(dates))
                        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;
```