**ASSIGNMENT\_5 - SPOTIFY DATA ANALYSIS**

**Introduction**

Spotify is a digital music streaming service that provides users access to over 82 million songs, podcasts and audio books. The app was developed by Daniel Ek and Martin Lorenzton in 2006. This app has become a family name over the years and boasts over 457 million subscribers as of 2022, rivaling SoundCloud and Apple Music.

Spotify measures the popularity of its' artists based on their monthly listeners and number of streams they receive on songs produced. These streams are then multipled by (0.003) and paid to artists as "Royalties", it is a modernized system of monetizing digital sales from traditional album sales (100 streams = 1 album). Ed Sheeran was Spotify's most streamed artist in 2019, however, the rank placements change rapidly depending on album relases, EP's, mixtapes and so forth!

Spotify is a perfect dataset to measure the popularity of songs against various music elements, across a large set of songs throughout the decades. This analysis can be used to demonstrate how peoples music tastes have been translated throughout the past two decades!

I will be creating an exploratory analysis by creating data visualizations and conducting statistical analyses to investigate the relationship between the use of non-traditional musical elements and the popularity of Spotify hits from 2000 to 2019.

**Track Metadata**

|  |  |
| --- | --- |
| **Column** | **Description** |
| Track\_Name | Song title |
| Artist\_Name | Song Artist |
| Artist\_Genre | Song Genre Category |
| Year | Song Billboard chart entry year |

**Audio Numerical Quantitive Data**

|  |  |
| --- | --- |
| **Column** | **Description** |
| Loudness | How loud a song is (db) |
| Duration\_MS | How long the song is (seconds) |
| Tempo | How fast a song is (bpm) |

**Audio Qualitative Data**

|  |  |
| --- | --- |
| **Column** | **Description** |
| Energy | How energetic the song is |
| Dance\_Ability | How easy it is to dance to |
| Valence | How positive the mood of the song is |
| Acousticness | How acoustic sounding the song is |
| Speechiness | How much of a song is spoken word |
| Track\_Popularity | How popular a song is (as of time of data collection) |

**Table Structure:**

Create a table named PLAYLIST with the following structure:

CREATE OR REPLACE TABLE PLAYLIST

(

PLAYLIST\_URL VARCHAR(100),

YEAR\_NO INT,

TRACK\_ID VARCHAR(50),

TRACK\_NAME VARCHAR(100),

TRACK\_POPULARITY INT,

ALBUM VARCHAR(100),

ARTIST\_ID VARCHAR(30) ,

ARTIST\_NAME VARCHAR(50),

ARTIST\_GENRES VARCHAR(200),

ARTIST\_POPULARITY INT,

DANCE\_ABILITY DECIMAL(5,3),

ENERGY DECIMAL(6,4),

KEY\_ID TINYINT,

LOUDNESS DECIMAL(6,4),

MODE\_BIT TINYINT,

SPEECHINESS DECIMAL(6,4),

ACOUSTICNESS DECIMAL(10,8),

INSTRUMENTALNESS DECIMAL(15,10),

LIVENESS DECIMAL(6,4),

VALENCE DECIMAL(6,4),

TEMPO DECIMAL(7,4),

DURATION\_MS INT,

TIME\_SIGNATURE TINYINT,

PRIMARY KEY (TRACK\_ID, ARTIST\_ID)

);

**Task :**

1. **Check the entire dataset**

SELECT \* FROM SPOTIFY\_DATABASE.PUBLIC.PLAYLIST;

A screenshot of a computer

Description automatically generated

1. **Number of songs on Spotify for each artist**

SELECT ARTIST\_NAME,

COUNT(TRACK\_ID) AS TOT\_SONGS

FROM SPOTIFY\_DATABASE.PUBLIC.PLAYLIST

GROUP BY 1

ORDER BY 2 DESC;

A screenshot of a music list

Description automatically generated

1. **Top 10 songs based on popularity**

SELECT TRACK\_NAME AS SONGS

FROM SPOTIFY\_DATABASE.PUBLIC.PLAYLIST

ORDER BY TRACK\_POPULARITY DESC

LIMIT 10;

A screenshot of a music survey

Description automatically generated

1. **Total number of songs on spotify based on year**

SELECT YEAR\_NO AS YEAR,

COUNT(TRACK\_ID) AS TOT\_SONGS

FROM SPOTIFY\_DATABASE.PUBLIC.PLAYLIST

GROUP BY 1

ORDER BY 1;

A screenshot of a computer

Description automatically generated

1. **Top song for each year (2000-2022) based on popularity**

SELECT YEAR\_NO AS Year, Track\_Name

FROM PLAYLIST P1,(

SELECT YEAR\_NO AS Year,

MAX(TRACK\_POPULARITY) AS MOST\_POPULAR

FROM SPOTIFY\_DATABASE.PUBLIC.PLAYLIST

GROUP BY 1) T2

WHERE P1.TRACK\_POPULARITY = T2.MOST\_POPULAR AND P1.YEAR\_NO = T2.Year;

A screenshot of a computer

Description automatically generated

1. **Analysis based on Tempo :**

**tempo > 121.08 -> 'Above Average Tempo'**

**tempo = 121.08 -> 'Average Tempo'**

**tempo < 121.08 -> 'Below Average Tempo'**

**Note:**

**I have created a View here so that I can use this view to answer other queries related to this analysis.**

CREATE OR REPLACE VIEW PLAYLIST\_TEMPO\_ANALYSIS\_VIEW AS

SELECT TRACK\_NAME, ENERGY, TEMPO,

CASE

WHEN TEMPO > 121.08 THEN 'Above Average Tempo'

WHEN TEMPO = 121.08 THEN 'Average Tempo'

WHEN TEMPO < 121.08 THEN 'Below Average Tempo'

END AS TEMPO\_CATEGORY

FROM SPOTIFY\_DATABASE.PUBLIC.PLAYLIST

WHERE TEMPO IS NOT NULL;

SELECT \* FROM Playlist\_Tempo\_Analysis\_View ;

A screenshot of a computer

Description automatically generated

1. **Songs with Highest Tempo**

SELECT TRACK\_NAME, TEMPO, TEMPO\_CATEGORY

FROM SPOTIFY\_DATABASE.PUBLIC.PLAYLIST\_TEMPO\_ANALYSIS\_VIEW

ORDER BY 2 DESC

LIMIT 1;

A screenshot of a computer

Description automatically generated

1. **Number of Songs for different Tempo Range : track\_name, energy**

**Modern\_Music -> tempo BETWEEN 60.00 AND 100.00**

**Classical\_Music -> tempo BETWEEN 100.001 AND 120.00**

**Dance\_Music -> tempo BETWEEN 120.001 AND 150.01**

**HighTempo\_Music -> tempo > 150.01**

**Note:**

**I have created a View here so that I can use this view to answer other queries related to this analysis.**

CREATE OR REPLACE VIEW PLAYLIST\_TEMPO\_ANALYSIS\_VIEW\_2 AS

SELECT TRACK\_NAME, ENERGY, TEMPO,

CASE

WHEN TEMPO BETWEEN 60.00 AND 100.00 THEN 'Modern\_Music'

WHEN TEMPO BETWEEN 100.001 AND 120.00 THEN 'Classical\_Music'

WHEN TEMPO BETWEEN 120.001 AND 150.01 THEN 'Dance\_Music'

WHEN TEMPO > 150.01 THEN 'HighTempo\_Music'

END AS Music\_Type

FROM SPOTIFY\_DATABASE.PUBLIC.PLAYLIST

WHERE TEMPO IS NOT NULL;

SELECT \* FROM PLAYLIST\_TEMPO\_ANALYSIS\_VIEW;

A screenshot of a music report

Description automatically generated

* 1. **Number of songs of different TEMPO range**

SELECT Music\_Type, COUNT(TRACK\_NAME) AS Tot\_Songs

FROM PLAYLIST\_TEMPO\_ANALYSIS\_VIEW\_2

GROUP BY 1

ORDER BY 2 DESC;

A screenshot of a music chart

Description automatically generated

1. **Energy Analysis : TOP 10 track\_name, danceability, track\_popularity**

**energy > 0.64 -> 'Above Average Energy**

**energy = 0.64 -> 'Average Energy’**

**energy < 0.64 -> 'Below Average Energy’**

**energy BETWEEN 0.1 AND 0.3 -> 'Calm Music'**

**energy BETWEEN 0.3 AND 0.6 -> 'Moderate Music'**

**Energy >0.6 -> ‘Energetic Music'**

**Note:**

* **In this question, there is given 2 types of range on Energy so I have divided this question in 2 parts.**
* **I have created a View here so that I can use this view to answer other queries related to this analysis.**
  1. **Energy Analysis : TOP 10 track\_name, danceability, track\_popularity**

**energy > 0.64 -> 'Above Average Energy**

**energy = 0.64 -> 'Average Energy’**

**energy < 0.64 -> 'Below Average Energy’**

CREATE OR REPLACE VIEW PLAYLIST\_ENERGY\_ANALYSIS\_VIEW AS

SELECT TRACK\_NAME, DANCE\_ABILITY, TRACK\_POPULARITY, ENERGY,

CASE

WHEN ENERGY > 0.64 THEN 'Above Average Energy'

WHEN ENERGY = 0.64 THEN 'Average Energy'

WHEN ENERGY < 0.64 THEN 'Below Average Energy'

END AS Energy\_Type

FROM SPOTIFY\_DATABASE.PUBLIC.PLAYLIST

WHERE ENERGY IS NOT NULL;

SELECT \* FROM PLAYLIST\_ENERGY\_ANALYSIS\_VIEW;

A screenshot of a computer

Description automatically generated

* 1. **Energy Analysis : TOP 10 track\_name, danceability, track\_popularity**

**Energy BETWEEN 0.1 AND 0.3 -> 'Calm Music'**

**Energy BETWEEN 0.3 AND 0.6 -> 'Moderate Music'**

**Energy >0.6 -> ‘Energetic Music'**

CREATE OR REPLACE VIEW PLAYLIST\_ENERGY\_ANALYSIS\_VIEW\_2 AS

SELECT TRACK\_NAME, DANCE\_ABILITY, TRACK\_POPULARITY, ENERGY,

CASE

WHEN ENERGY BETWEEN 0.1 AND 0.3 THEN 'Calm Music'

WHEN ENERGY BETWEEN 0.3 AND 0.6 THEN 'Moderate Music'

WHEN ENERGY > 0.6 THEN 'Energetic Music'

ELSE 'Others'

END AS Music\_Type

FROM SPOTIFY\_DATABASE.PUBLIC.PLAYLIST

WHERE ENERGY IS NOT NULL;

SELECT \* FROM PLAYLIST\_ENERGY\_ANALYSIS\_VIEW\_2;

A screenshot of a computer

Description automatically generated

1. **Number of Songs for different energy ranges(above)**

**FOR 9.1**

SELECT ENERGY\_TYPE, COUNT(TRACK\_NAME) AS TOT\_SONGS

FROM PLAYLIST\_ENERGY\_ANALYSIS\_VIEW

GROUP BY 1

ORDER BY 2 DESC;

A screenshot of a computer

Description automatically generated

**FOR 9.2**

SELECT MUSIC\_TYPE, COUNT(TRACK\_NAME) AS TOT\_SONGS

FROM PLAYLIST\_ENERGY\_ANALYSIS\_VIEW\_2

GROUP BY 1

ORDER BY 2 DESC;

A screenshot of a music chart

Description automatically generated

1. **Danceability Analysis : Top 20 track\_name, danceability**

**danceability BETWEEN 0.69 AND 0.79 -> 'Low Danceability'**

**(danceability BETWEEN 0.49 AND 0.68) OR (danceability BETWEEN 0.79 AND 0.89) -> 'Moderate Danceability'**

**(danceability BETWEEN 0.39 AND 0.49) OR (danceability BETWEEN 0.89 AND 0.99) -> 'High Danceability'**

**danceability < 0.39 OR danceability > 0.99 -> 'Cant Dance on this one'**

**Note:**

**I have created a View here so that I can use this view to answer other queries related to this analysis.**

CREATE OR REPLACE VIEW PLAYLIST\_DANCE\_ABILITY\_ANALYSIS\_VIEW AS

SELECT TRACK\_NAME, DANCE\_ABILITY,

CASE

WHEN DANCE\_ABILITY BETWEEN 0.69 AND 0.79 THEN 'Low Danceability'

WHEN (DANCE\_ABILITY BETWEEN 0.49 AND 0.68) OR (DANCE\_ABILITY BETWEEN 0.79 AND 0.89) THEN 'Moderate Danceability'

WHEN (DANCE\_ABILITY BETWEEN 0.39 AND 0.49) OR (DANCE\_ABILITY BETWEEN 0.89 AND 0.99) THEN 'High Danceability'

WHEN DANCE\_ABILITY < 0.39 OR DANCE\_ABILITY > 0.99 THEN 'Cant Dance on this one'

ELSE 'Others'

END AS Dance\_Ability\_Category

FROM SPOTIFY\_DATABASE.PUBLIC.PLAYLIST

WHERE DANCE\_ABILITY IS NOT NULL;

SELECT \* FROM PLAYLIST\_DANCE\_ABILITY\_ANALYSIS\_VIEW;

A screenshot of a computer

Description automatically generated

1. **Number of Songs for different danceability ranges(above)**

SELECT DANCE\_ABILITY\_CATEGORY, COUNT(TRACK\_NAME) AS TOT\_SONGS

FROM PLAYLIST\_DANCE\_ABILITY\_ANALYSIS\_VIEW

GROUP BY 1

ORDER BY 2 DESC;

A screenshot of a music website

Description automatically generated

1. **Loudness Analysis : Top 20 track\_name, loudness,**

**loudness BETWEEN -23.00 AND -15.00 ->'Low Loudness'**

**loudness BETWEEN -14.99 AND -6.00 -> 'Below Average Loudness'**

**loudness BETWEEN -5.99 AND -2.90 -> 'Above Average Loudness'**

**loudness BETWEEN -2.89 AND -1.00 -> 'Peak Loudness'**

**Note:**

**I have created a View here so that I can use this view to answer other queries related to this analysis.**

CREATE OR REPLACE VIEW PLAYLIST\_LOUDNESS\_ANALYSIS\_VIEW AS

SELECT TRACK\_NAME, LOUDNESS,

CASE

WHEN LOUDNESS BETWEEN -23.00 AND -15.00 THEN 'Low Loudness'

WHEN LOUDNESS BETWEEN -14.99 AND -6.00 THEN 'Below Average Loudness'

WHEN LOUDNESS BETWEEN -5.99 AND -2.90 THEN 'Above Average Loudness'

WHEN LOUDNESS BETWEEN -2.89 AND -1.00 THEN 'Peak Loudness'

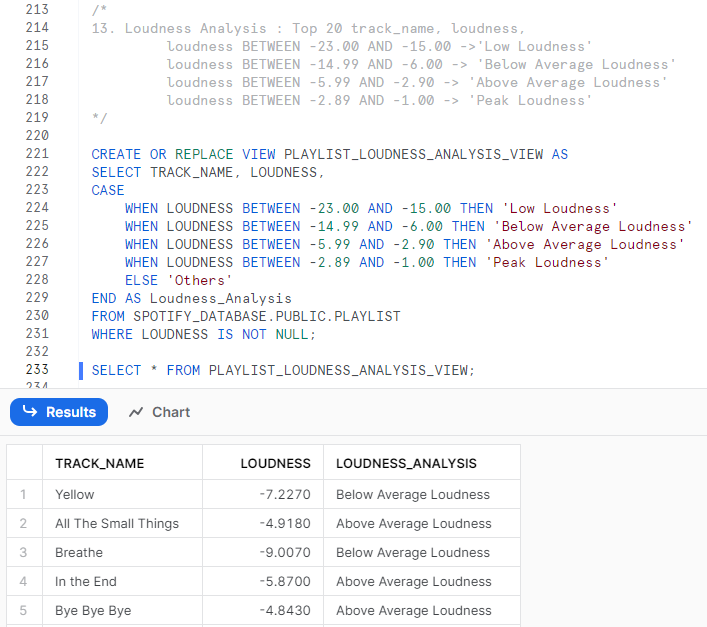
ELSE 'Others'

END AS Loudness\_Analysis

FROM SPOTIFY\_DATABASE.PUBLIC.PLAYLIST

WHERE LOUDNESS IS NOT NULL;

SELECT \* FROM PLAYLIST\_LOUDNESS\_ANALYSIS\_VIEW;



1. **Number of Songs for different loudness ranges(above)**

SELECT LOUDNESS\_ANALYSIS, COUNT(TRACK\_NAME) AS TOT\_SONGS

FROM PLAYLIST\_LOUDNESS\_ANALYSIS\_VIEW

GROUP BY 1

ORDER BY 2 DESC;

A screenshot of a computer

Description automatically generated

1. **Valence Analysis : Top 20 track\_name, valence, track\_popularity,**

**valence > 0.535 -> Above Avg Valence**

**valence = 0.535 -> Avg Valence**

**valence < 0.535 -> Below Average'**

**Note:**

**I have created a View here so that I can use this view to answer other queries related to this analysis.**

CREATE OR REPLACE VIEW PLAYLIST\_VALENCE\_ANALYSIS\_VIEW AS

SELECT TRACK\_NAME, TRACK\_POPULARITY, VALENCE,

CASE

WHEN valence > 0.535 THEN 'Above Avg Valence'

WHEN valence = 0.535 THEN 'Avg Valence'

WHEN valence < 0.535 THEN 'Below Avg Valence'

END AS Valence\_Analysis

FROM SPOTIFY\_DATABASE.PUBLIC.PLAYLIST

WHERE VALENCE IS NOT NULL;

SELECT \* FROM PLAYLIST\_VALENCE\_ANALYSIS\_VIEW;

A screenshot of a computer

Description automatically generated

1. **Number of Songs for different valence ranges(above)**

SELECT VALENCE\_ANALYSIS, COUNT(TRACK\_NAME) AS TOT\_SONGS

FROM PLAYLIST\_VALENCE\_ANALYSIS\_VIEW

GROUP BY 1

ORDER BY 2 DESC;

A screenshot of a computer

Description automatically generated

1. **Speechiness Analsis : Top 20 track\_name, speechiness, tempo,**

**speechiness > 0.081-> Above Avg Speechiness**

**speechiness = 0.081-> Avg Speechiness**

**speechiness < 0.081-> Below Speechiness**

SELECT TRACK\_NAME,TEMPO, SPEECHINESS,

CASE

WHEN SPEECHINESS > 0.081 THEN 'Above Avg Speechiness'

WHEN SPEECHINESS = 0.081 THEN 'Avg Speechiness'

WHEN SPEECHINESS < 0.081 THEN 'Below Avg Speechiness'

END AS Speechiness\_Analysis

FROM SPOTIFY\_DATABASE.PUBLIC.PLAYLIST

WHERE SPEECHINESS IS NOT NULL

LIMIT 20;

A screenshot of a computer

Description automatically generated

1. **Acoustic Analysis : DISTINCT TOP 25 track\_name, album, artist\_name, acousticness**

**(acousticness BETWEEN 0 AND 0.40000 -> 'Not Acoustic'**

**(acousticness BETWEEN 0.40001 AND 0.80000) ->'Acoustic'**

**(acousticness BETWEEN 0.80001 AND 1) ->'Highly Acoustic'**

SELECT DISTINCT TRACK\_NAME, ALBUM, ARTIST\_NAME, ACOUSTICNESS,

CASE

WHEN ACOUSTICNESS BETWEEN 0 AND 0.40000 THEN 'Not Acoustic'

WHEN ACOUSTICNESS BETWEEN 0.40001 AND 0.80000 THEN 'Acoustic'

WHEN ACOUSTICNESS BETWEEN 0.80001 AND 1 THEN 'Highly Acoustic'

ELSE 'Others'

END AS Acousticness\_Analysis

FROM SPOTIFY\_DATABASE.PUBLIC.PLAYLIST

WHERE ACOUSTICNESS IS NOT NULL

LIMIT 25;

A screenshot of a computer

Description automatically generated

**\*\*\*\*\*\*\*\*\*\*\*\*\*\* THANK YOU \*\*\*\*\*\*\*\*\*\*\*\*\*\*\***