INDEX

S.NO	TOPIC	PG NO.
1.	Introduction	1
2.	Objectives	2
3.	Literature review	3
4.	Scope	4
5.	Technical requirement	5
6.	Code and implementation	6-18
7.	Future Scope	19
8.	Conclusion	20
9.	Reference	21

LIST OF FIGURES

SERIAL NO.	DESCRIPTION	PAGE NO.
1.	Dataset overview	6
2.	Number of times charted by artist	7
3.	TOP 7 ARTISTS WITH HIGHEST NUMBER OF TIMES CHARTED	8
4.	Danceability over the course of the Year	8
6.	Scatter plot basis on dance and energy	9
7.	Number of Times Charted' correlates with years	10
8.	Artist by popularity	11
9.	Most popular genres	12
10.	Top 30 most frequent genres	13
11.	TOP 10 GENRE ANALYSIS	15
12.	TOP 20 ARTISTS AND COUNT	16
13.	TOP 10 MOST STREAMED SONGS	17
14.	TOP 20 MOST STREAMED SONGS	18

INTRODUCTION

Data Science is a multidisciplinary field that combines statistics, machine learning, and data analysis to extract meaningful insights from structured and unstructured data. It involves processes such as data collection, cleaning, exploratory data analysis, feature engineering, predictive modeling, and data visualization.

COMPONENTS:

- **Data Collection** – Gathering data from various sources like databases, APIs, sensors, and web scraping.
- **Data Cleaning & Preprocessing** – Removing errors, handling missing values, and transforming data into a usable format.
- **Exploratory Data Analysis (EDA)** – Identifying patterns, trends, and relationships in data using statistical and visualization techniques.
- **Feature Engineering** – Creating new meaningful features from raw data to improve model accuracy.
- **Machine Learning & Modelling** – Developing predictive models using algorithms like regression, decision trees, and neural networks.
- **Data Visualization** – Representing data insights through graphs, charts, and dashboards for better understanding.
- **Big Data Processing** – Handling large datasets efficiently using frameworks like Hadoop and Spark.

OBJECTIVE

This project focuses on analyzing and exploring Spotify's music data to derive valuable insights into song popularity, emerging trends, and user listening preferences. Using advanced data visualization techniques and statistical analysis, it aims to identify key factors that contribute to a song’s success on the platform. Furthermore, the project incorporates machine learning models to build an intelligent recommendation system that personalizes song suggestions based on user behavior. By understanding evolving music consumption patterns, this study highlights how data-driven strategies can optimize the streaming experience, improve user engagement, and support artists in reaching wider audiences.

LITERATURE REVIEW

Music streaming platforms like Spotify generate vast amounts of data, enabling advanced analytics to understand user preferences and song popularity. Previous studies highlight the impact of factors like tempo, danceability, and artist popularity on streaming trends. Traditional recommendation systems relied on collaborative filtering, but modern AI-driven models now enhance personalization.

Machine learning techniques, including regression and clustering, have been used to predict song success and improve playlist curation. Data visualization tools like bar charts and scatter plots help identify patterns in music consumption. However, gaps remain in real-time data processing, sentiment analysis, and contextual recommendations.

This project builds on existing research by applying machine learning and data visualization to analyse Spotify trends, aiming to improve recommendation systems and enhance user engagement.

Scope of this project

This project aims to analyze Spotify's music data to extract meaningful insights into song popularity, user listening patterns, and emerging trends. It focuses on:

- **Music Trend Analysis:** Identifying key factors influencing a song's success, such as danceability, energy, and artist popularity.
- **User Preference Insights:** Understanding how users engage with different genres and artists.
- **Recommendation System:** Implementing machine learning models to provide personalized song recommendations.
- **Data Visualization:** Using charts and graphs to represent trends in streaming behavior.
- **Industry Applications:** Helping artists, music producers, and streaming platforms optimize content strategy and audience reach.

TECHNICAL REQUIREMENTS

1. Programming Languages & Libraries
   - Python – For data analysis, visualization, and machine learning.
   - Libraries:
     - Pandas & NumPy – Data manipulation and processing.
     - Matplotlib & Seaborn – Data visualization. ○ Scikit-Learn – Machine learning algorithms (if applicable).
     - Spotipy – Spotify API integration for fetching song data.
2. Data Sources
   - Spotify API – For retrieving song metadata, user listening patterns, and playlist details.
   - CSV / JSON Datasets – Pre-collected datasets for analysis.
3. Development Environment
   - Jupyter Notebook / Google Colab – For interactive coding and analysis.
   - PyCharm / VS Code – Alternative for structured development.
4. Machine Learning (If Applicable)
   - Clustering & Classification Models – For song recommendations.
   - Regression Models – To predict song popularity based on past trends.
5. APIs & Integration
   - Spotify Web API – For accessing Spotify's music database.

Code and Implementation

- ## Dataset Overview:

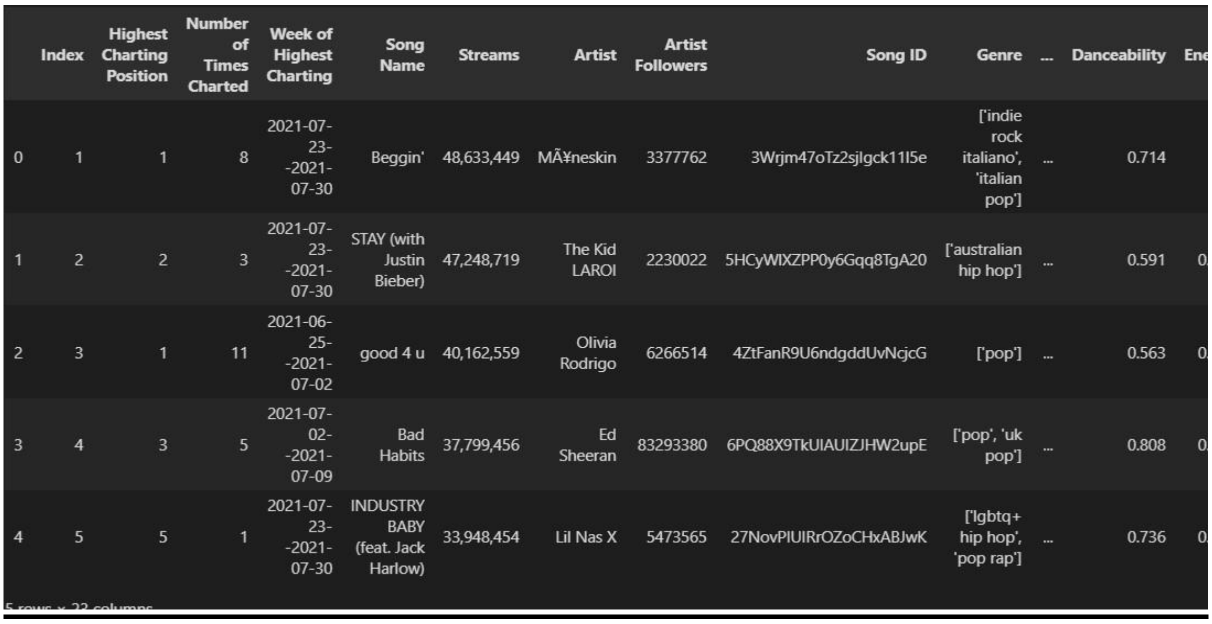
This command reads the **CSV file** and stores it in a **DataFrame (df)**. The encoding='latin-1' ensures proper handling of special characters in the dataset.

This function displays the **first five rows** of the dataset, providing an overview of its structure, column names, and data types. It helps in **initial data exploration** before analysis.

CODE:

```
 df = pd.read_csv("spotify_dataset.csv", encoding='latin-
1') df.head()
```

RESULT:



	Index	Highest Charting Position	Number of Times Charted	Week of Highest Charting	Song Name	Streams	Artist	Artist Followers	Song ID	Genre	...	Danceability	Ene
0	1	1	8	2021-07-23--2021-07-30	Beggin'	48,633,449	MÃ¥neskin	3377762	3Wrjm47oTz2sjlgck11l5e	['indie rock italiano', 'italian pop']	...	0.714	
1	2	2	3	2021-07-23--2021-07-30	STAY (with Justin Bieber)	47,248,719	The Kid LAROI	2230022	5HCyWlXZPP0y6Gqq8TgA20	['australian hip hop']	...	0.591	0.
2	3	1	11	2021-06-25--2021-07-02	good 4 u	40,162,559	Olivia Rodrigo	6266514	4ZtFanR9U6ndgddUvNcjcG	['pop']	...	0.563	0.
3	4	3	5	2021-07-02--2021-07-09	Bad Habits	37,799,456	Ed Sheeran	83293380	6PQ88X9TkUIAUIZJHW2upE	['pop', 'uk pop']	...	0.808	0.
4	5	5	1	2021-07-23--2021-07-30	INDUSTRY BABY (feat. Jack Harlow)	33,948,454	Lil Nas X	5473565	27NovPIUIRrOZoCHxABJwK	['lgbtq+ hip hop', 'pop rap']	...	0.736	0.

FIGURE:1

- ## Number of times charted by artist:

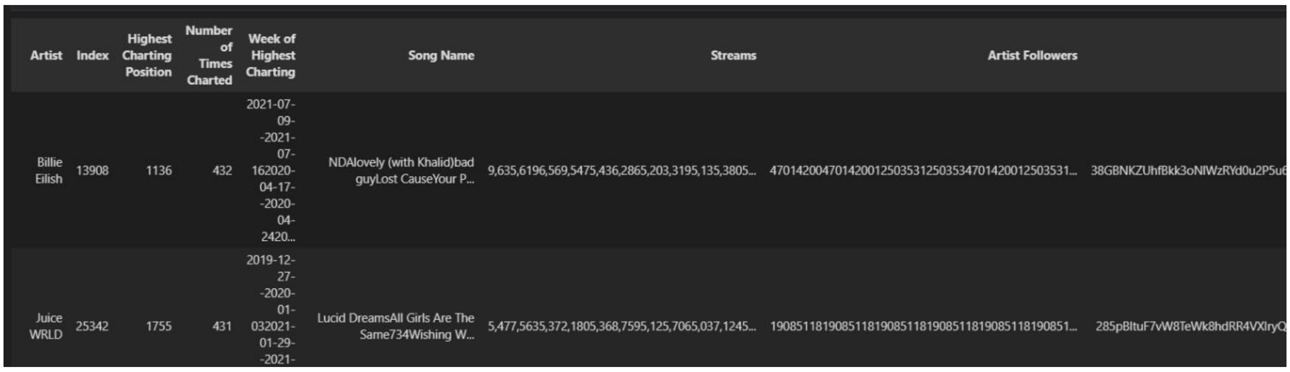
The given code is used to calculate how many times each artist's songs have **charted** (appeared on the Spotify chart). It groups the data by **artist name** and then sums up the**"Number of Times Charted"** column.

CODE:

```
#number of times charted by artist

df_numbercharted=df.groupby('Artist').sum().sort_values('Number of Times
Charted', ascending=False) df_numbercharted=df_numbercharted.reset_index()
df_numbercharted
```

RESULT:



Artist	Index	Highest Charting Position	Number of Times Charted	Week of Highest Charting	Song Name	Streams	Artist Followers	
Billie Eilish	13908	1136	432	2021-07-09--2021-07-162020-04-17--2020-04-2420...	NDAlovely (with Khalid)bad guyLost CauseYour P...	9,635,6196,569,5475,436,2865,203,3195,135,3805...	4701420047014200125035312503534701420012503531...	38GBNKZUhfBkk3oNIWzRYd0u2P5u6
Juice WRLD	25342	1755	431	2019-12-27--2020-01-032021-01-29--2021-	Lucid DreamsAll Girls Are The Same734Wishing W...	5,477,5635,372,1805,368,7595,125,7065,037,1245...	1908511819085118190851181908511819085118190851...	285pBltuF7vW8TeWk8hdRR4VXlryQ

FIGURE:2

• TOP 7 ARTISTS WITH HIGHEST NUMBER OF TIMES CHARTED

A bar chart was generated to visualize the **top 7 artists** with the highest number of times charted on Spotify. This analysis highlights the most successful artists and provides insights into music trends and listener preferences.

CODE:

```
px.bar(x='Artist', y='Number of Times Charted',
data_frame=df_numbercharted.head(7), title="Top 7 Artists with HighesT Number of
Times Charted")
```

RESULT:

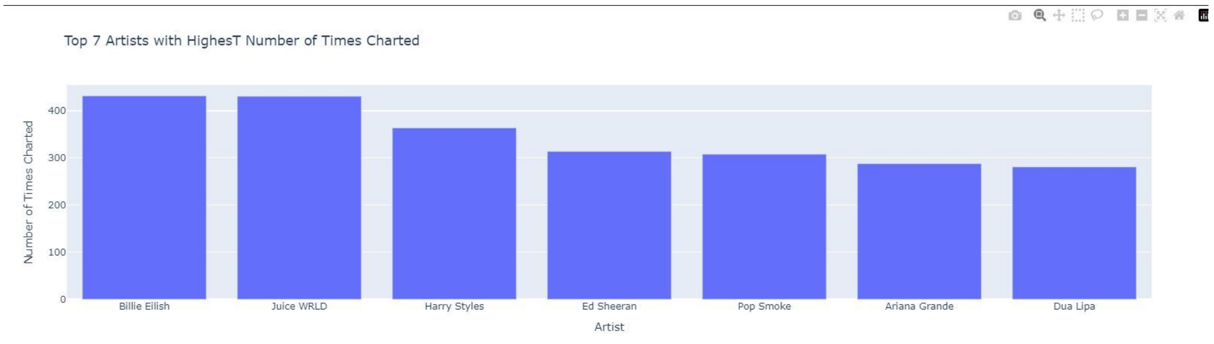


FIGURE:3

• Danceability over the course of the Year

A line chart was created to analyze how **danceability** in songs has changed over the years. This trend helps in understanding the evolution of music styles and listener preferences over time.

CODE:

```
#danceability
 px.line(x='Release Year', y='Danceability', data_frame=df,
title="Danceability over the course of the Year")
```

RESULT:

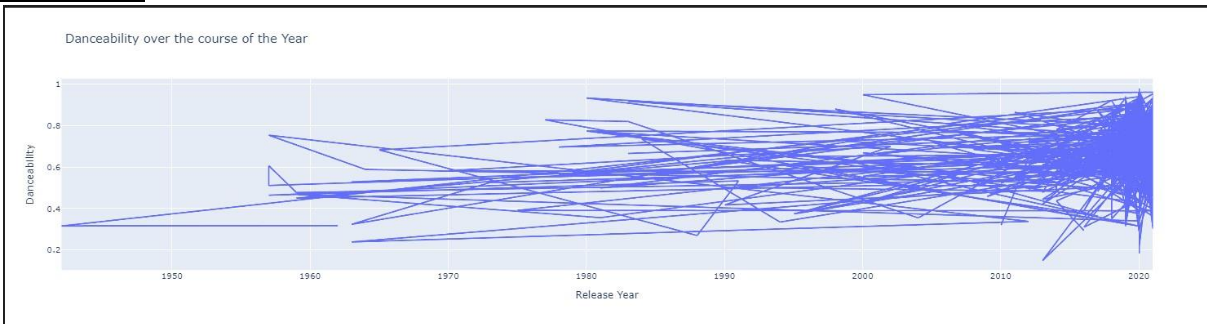


FIGURE:4

EXPLANATION:

The graph shows **danceability trends from 1920 to 2020**, with notable changes after 1980 due to more song releases and evolving music styles. It reflects shifts in musical production and audience preferences over time.

• <u>SCATTER PLOT ON THE BASIS OF DANCEABILITY AND ENERGY</u>

A scatter plot was created to analyze the relationship between **Danceability** and **Energy** in songs. The visualization helps in understanding whether high-energy songs tend to be more danceable.

<u>CODE:</u>

```
sns.scatterplot(data=df, x='Danceability', y='Energy') plt.show()
```

<u>RESULT:</u>

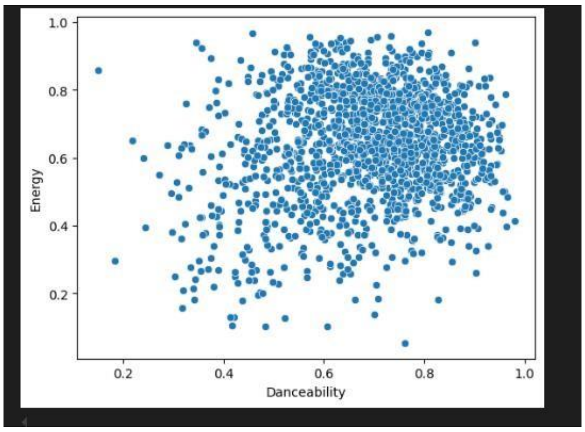


FIGURE:5 **<u>Explanation:</u>**

Scatter Plot: Danceability vs. Energy

The scatter plot illustrates the relationship between **Danceability** and **Energy** of different songs. Each point represents a song, where:

- The **X-axis** shows the danceability score, determining how rhythmic and dance-friendly a song is.
- The **Y-axis** represents the energy level, indicating the song's intensity and liveliness.

• <u>Number of Times Charted' correlates with years</u>

A bar chart was created to analyze the correlation between **release year** and the **number of times songs were charted**. The results highlight the most influential years in music, showcasing trends in song popularity over time.

<u>CODE:</u>

```
#Number of Times Charted' correlates with years

dfyear = df.groupby('Release Year').sum().sort_values('Number of
Times Charted', ascending=False) dfyear=dfyear.reset_index()
 px.bar(x='Release Year', y='Number of Times
Charted', data_frame=dfyear.head(7))
```

RESULT:

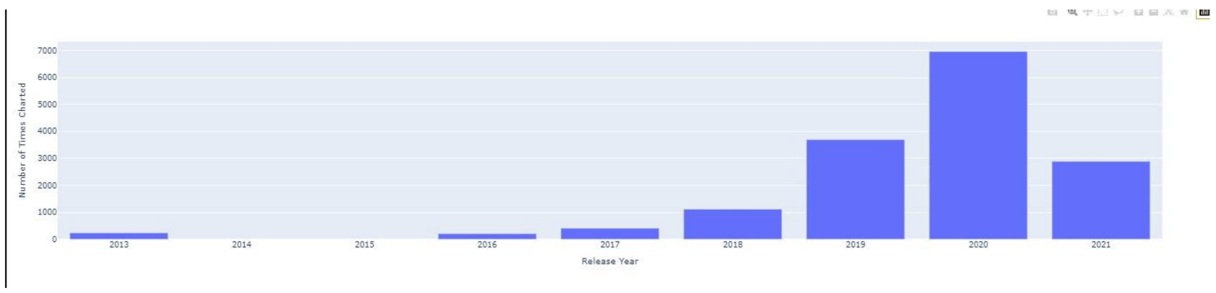


FIGURE:6

EXPLANATION:

"The bar chart shows the number of citations across different release years, ranging from 2013 to 2021. From 2013 to 2018, citations are almost negligible, but from 2019 onwards, there is a notable rise. Citations peak in 2020 with approximately 7,000, followed by a dip to about 4,000 in 2021

• ARTISTS BY POPULARITY

A bar chart was generated to visualize the **top 20 most popular artists** based on their cumulative popularity scores. This analysis helps identify the most influential artists in the dataset and their impact on the music industry.

CODE:

```
 artistbypop =
df.groupby('Artist').sum().sort_values('Popularity'
,ascending=False)[:20]
artistbypop=artistbypop.reset_index() px.bar(x='Artist',
y='Popularity', data_frame=artistbypop)
```

RESULT:

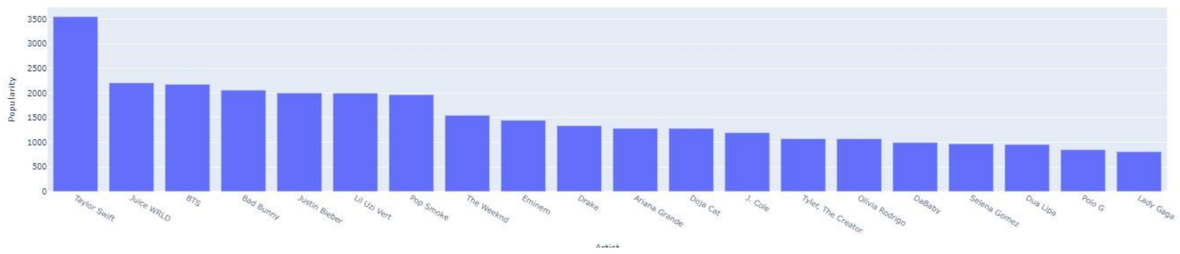


FIGURE:7 **Explanation:**

"The bar chart compares the popularity of 20 artists, with **Taylor Swift** leading, followed by **Juice WRLD** and **BTS**. This visual highlights the varying popularity levels and gives insight into listener preferences across genres."

• Most popular geners

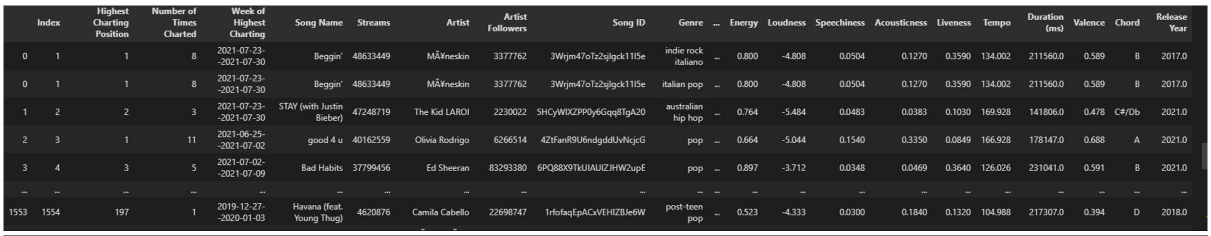
To analyze the most popular genres, the dataset was preprocessed by **cleaning and splitting multiple genre entries** per song. The cleaned dataset now allows a more accurate breakdown of genre trends in music.

CODE:

```
#most popular geners
 df['Genre']=df['Genre'].astype(str)
df["Genre"][df["Genre"] == "[]"] =
np.nan df["Genre"] =
df["Genre"].fillna(0)
#here we get rid of useless symbols to be able to separate
genres df.Genre=df.Genre.str.replace("[", "")
df.Genre=df.Genre.str.replace("]", "")
df.Genre=df.Genre.str.replace("'", "") #now we devide genre
strings by comma df["Genre"] = df["Genre"].str.split(",")
#next command separates rows based on genres, so for each song that is
marked with several genres,
#now we'll have multiple rows with one genre for each row
df=df.explode('Genre')

df
```

RESULT:



	Index	Highest Charting Position	Number of Times Charted	Week of Highest Charting	Song Name	Streams	Artist	Artist Followers	Song ID	Genre	...	Energy	Loudness	Speechiness	Acousticness	Liveness	Tempo	Duration (ms)	Valence	Chord	Release Year
0	1	1	8	2021-07-23--2021-07-30	Beggin'	48633449	MÃ¥neskin	3377762	3Wrjm47oTz2sjlgck11l5e	indie rock italiano	...	0.800	-4.808	0.0504	0.1270	0.3590	134.002	211560.0	0.589	B	2017.0
0	1	1	8	2021-07-23--2021-07-30	Beggin'	48633449	MÃ¥neskin	3377762	3Wrjm47oTz2sjlgck11l5e	italian pop	...	0.800	-4.808	0.0504	0.1270	0.3590	134.002	211560.0	0.589	B	2017.0
1	2	2	3	2021-07-23--2021-07-30	STAY (with Justin Bieber)	47248719	The Kid LAROI	2230022	5HCyWlXZPP0y6Gqq8TgA20	australian hip hop	...	0.764	-5.484	0.0483	0.0383	0.1030	169.928	141806.0	0.478	C#/Db	2021.0
2	3	1	11	2021-06-25--2021-07-02	good 4 u	40162559	Olivia Rodrigo	6266514	4ZtFanR9U6ndgddUvNcjcG	pop	...	0.664	-5.044	0.1540	0.3350	0.0849	166.928	178147.0	0.688	A	2021.0
3	4	3	5	2021-07-02--2021-07-09	Bad Habits	37799456	Ed Sheeran	83293380	6PQ88X9TkUIAUIZJHW2upE	pop	...	0.897	-3.712	0.0348	0.0469	0.3640	126.026	231041.0	0.591	B	2021.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1553	1554	197	1	2019-12-27--2020-01-03	Havana (feat. Young Thug)	4620876	Camila Cabello	22698747	1rfofaqEpACxVEHIZBJe6W	post-teen pop	...	0.523	-4.333	0.0300	0.1840	0.1320	104.988	217307.0	0.394	D	2018.0

FIGURE:8

EXPLANATION:

This table contains information about songs, including their chart performance, streaming metrics, and musical characteristics.

Key Columns & Insights:

- **Chart Performance:** Includes *Highest Charting Position* and *Number of Times Charted*.
- **Streaming Data:** Shows *Streams* and *Artist Followers*.
- **Song Features:** Includes *Genre, Energy, Loudness, Speechiness, Acousticness, Tempo*, etc.
- **Release Year:** Helps in trend analysis over time.

• <u>Top 30 most frequent genres</u>

To visualize the distribution of the most popular music genres, a **pie chart** was created, representing the **top 30 most frequent genres** in the dataset. This visualization helps in understanding which genres dominate the music industry based on Spotify data.

<u>CODE:</u>

```
fig = plt.figure(figsize = (10, 10)) ax =
fig.subplots()
df.Genre.value_counts()[:30].plot(ax=ax, kind = "pie")
ax.set_ylabel("") ax.set_title("Top 30 most popular
genres") plt.show()
```

<u>RESULT:</u>

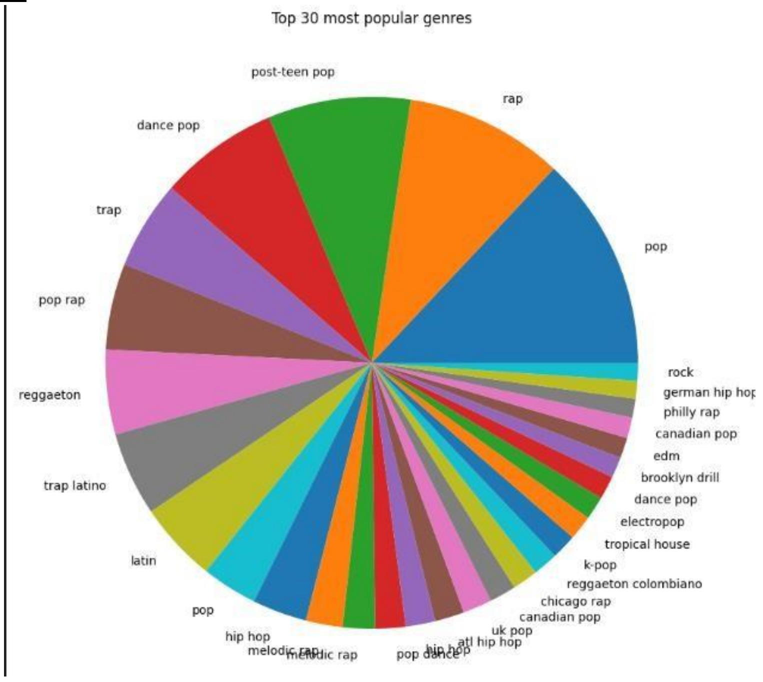


FIGURE:9

EXPLANATION:

This pie chart represents the top 30 most popular music genres. **Pop, Rap, and Post-Teen Pop** dominate, while genres like **Dance Pop, Trap, and Reggaeton** also hold significant shares. The presence of **Latin, EDM, K-Pop, and Electropop** indicates diverse listener preferences. This chart highlights the dominance of mainstream genres while showcasing a variety of sub-genres.

• TOP 10 GENRE ANALYSIS

To analyze the distribution of **music genres**, a **pie chart** was generated representing the **top 10 most popular genres** based on Spotify stream counts. The dataset was preprocessed to clean the stream data, and the genre frequencies were calculated. This visualization provides insights into the dominant genres in the music industry **CODE :**

```
import pandas as pd import
matplotlib.pyplot as plt

# Load the Spotify dataset spotify_data =
pd.read_csv('spotify_dataset.csv')
# Extract the relevant columns for genre analysis genre_data
= spotify_data[['Genre', 'Streams']]

# Remove commas from the stream counts and convert them to numeric values
genre_data['Streams'] = genre_data['Streams'].str.replace(',', '').astype(int)
# Calculate the genre counts genre_counts =
genre_data['Genre'].value_counts()
# Select the top 10 genres top_10_genres
= genre_counts.head(10)

# Define a list of colors for the top 10 genres colors = ['green', 'blue',
'orange', 'red', 'purple', 'pink', 'brown', 'gray', 'yellow', 'teal']

# Plot the top 10 genre distribution as a pie chart with different colors
plt.figure(figsize=(8, 8)) plt.pie(top_10_genres,
labels=top_10_genres.index, autopct='%1.1f%%', colors=colors)
plt.title('Top 10 Genre Distribution') plt.show()
```

RESULT:

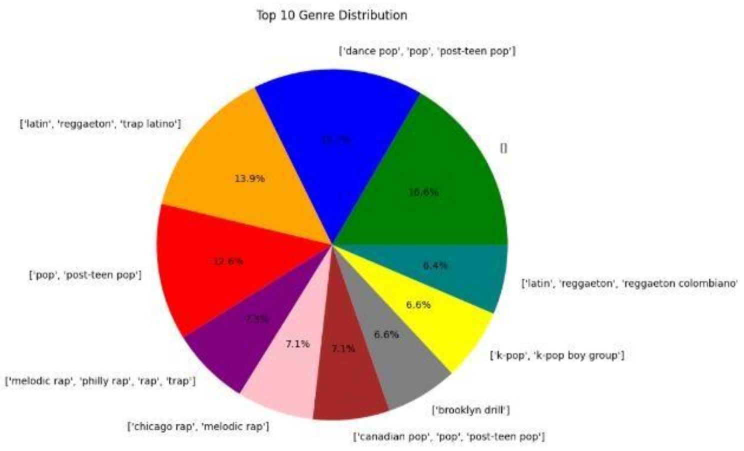


FIGURE:10

EXPLANATION:

The pie chart shows the **top 10 music genres**, with **Dance Pop, Pop, and Post-Teen Pop** having the largest share. **Latin, Reggaeton, and Trap Latino** also hold a significant portion, while **K-Pop, Brooklyn Drill, and Melodic Rap** add diversity. It highlights the mix of mainstream and niche music styles.

• TOP 20 ARTISTS AND COUNT

The above code identifies the **top 20 artists** based on the number of times they appear in the dataset.

The `sns.barplot()` function is used to create a horizontal bar chart.

CODE:

```
top_artists = df['Artist'].value_counts().head(20)
 sns.barplot(x=top_artists.values,
y=top_artists.index) plt.xlabel('Count')
plt.ylabel('Artist') plt.title('Top 20 Artists')
plt.show()
```

RESULT:

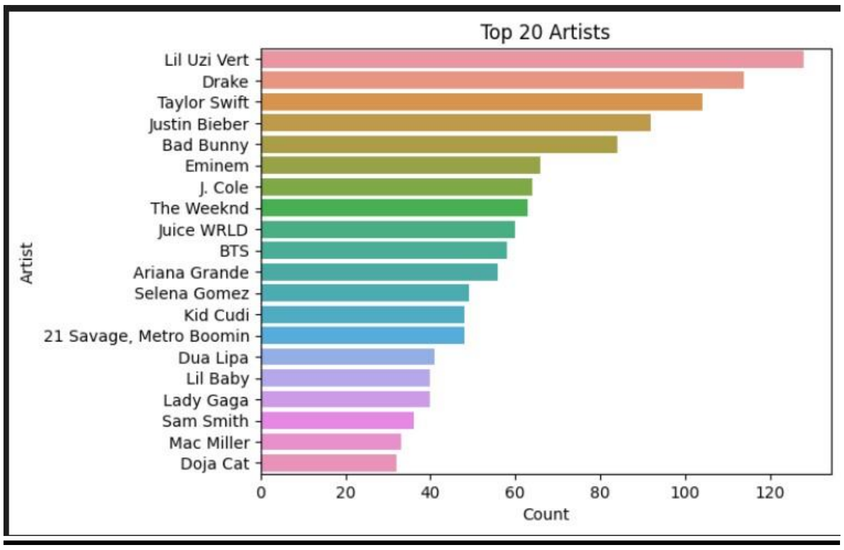


FIGURE:11

EXPLANATION:

This bar chart displays the **top 20 artists** based on count, with **Lil Uzi Vert, Drake, and Taylor Swift** leading the list. Other popular artists like **Justin Bieber, Bad Bunny, and Eminem** also have significant representation. The chart highlights the dominance of mainstream artists across various genres.

• TOP 10 MOST STREAMED SONGS

This code sorts the dataset by the number of streams in descending order and selects the top 10 most streamed songs. A bar plot is then created to visualize these songs, with the stream count on the y-axis and song names on the x-axis. The x-axis labels are rotated for better readability, and the bars are coloured pink for clear visualization.

CODE:

```
sorted_df = df.sort_values('Artist Followers', ascending=True)

top_artists = sorted_df['Artist'].head(20)

import matplotlib.pyplot as plt
 plt.pie(top_artists.value_counts(), labels=top_artists.unique(),
autopct='%1.1f%%') plt.title("Top 20 Followed Artists")
plt.axis('equal')  # Ensure the pie is drawn as a circle plt.show()
```

RESULT:

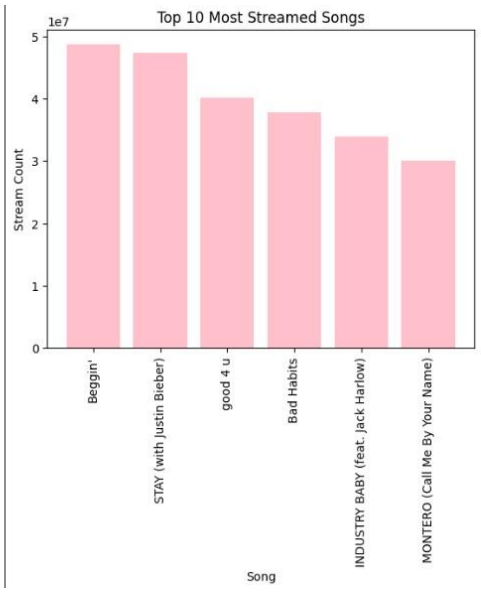


FIGURE:12

EXPLANATION:

This bar chart represents the **Top 10 Most Streamed Songs**, with **"Beggin'" and "STAY (with Justin Bieber)"** leading in stream count. Other highly streamed songs include **"good 4 u," "Bad Habits," and "MONTERO (Call Me By Your Name)."** The chart highlights the popularity of trending tracks and their massive audience reach.

• TOP 20 MOST STREAMED SONGS

This script identifies the top 20 most streamed artists by grouping the dataset, summing their total streams, sorting them in descending order, and visualizing the results in a bar plot. The chart uses pink bars with purple edges and rotated labels for clarity.

CODE:

```
import matplotlib.pyplot as plt

# Sort the dataset by streaming count in descending order
sorted_df = df.sort_values('Streams', ascending=False)

# Select the top 10 most streamed songs
top_streamed_songs = sorted_df.head(10)
```

```
# Create a bar plot plt.bar(top_streamed_songs['Song Name'],
top_streamed_songs['Streams'], color="pink") plt.xlabel('Song')
plt.ylabel('Stream Count') plt.title('Top 10 Most Streamed Songs')
plt.xticks(rotation=90)  # Rotate the x-axis labels for better
visibility
# Show the plot
plt.show()
```

RESULT:

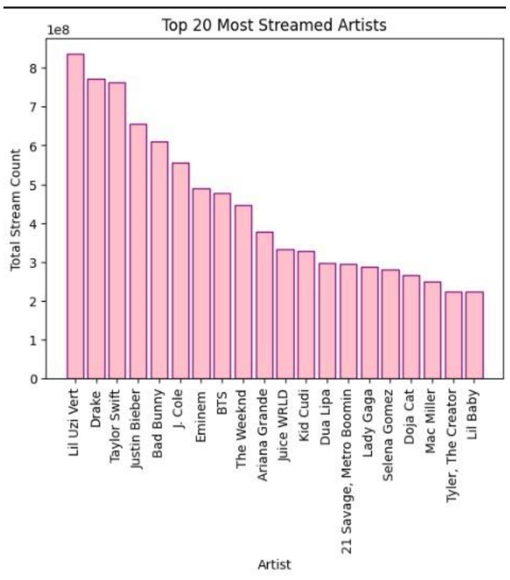


FIGURE:13

EXPLANATION:

"The bar graph illustrates the **top 20 most-streamed artists** globally. Among these, **Lil Uzi Vert** tops the chart, followed by prominent names like **Drake, Taylor Swift, and BTS**. The data reflects diverse listener preferences, spanning multiple genres, including pop, rap, and international music."

FUTURE SCOPE:

This project has been successfully developed, providing valuable insights into music trends and recommendations. However, if someone wants to build a similar project in the future, certain key aspects should be considered for better implementation and scalability.

1. **Technology Stack Selection** o Use **Python** or **R** for data analysis and machine learning.
   - Implement **Django** or **Flask** for the backend, and **React.js** or **Vue.js** for an interactive frontend.
   - Use databases like **MySQL, PostgreSQL, or MongoDB** to store user data and music preferences.
2. **Integration of Machine Learning Models** o Implement **collaborative filtering** or **content-based filtering** for better song recommendations.
   - Use **deep learning models** like neural networks to analyze user behavior and suggest personalized playlists.
3. **Real-Time Data Processing** o Utilize **APIs from streaming platforms** (like Spotify, YouTube Music) to fetch real-time data.
   - Implement **sentiment analysis** on social media comments to understand trending music preferences.
4. **Cross-Platform Compatibility** o Ensure that the system works across different devices (mobile, web, and desktop).
   - Use **cloud services** like AWS or Google Cloud for scalable storage and processing.
5. **Predictive Analytics for Music Trends** o Implement AI models to predict upcoming hit songs and emerging artists. o Use historical streaming data and user interactions to forecast music trends.

• <u>CONCLUSIONS</u>

This project successfully analyzes Spotify's music data to uncover patterns in song popularity, user preferences, and genre trends. Through data visualization and machine learning, it provides valuable insights into factors influencing a song's success. The recommendation system enhances personalized music discovery, improving user engagement.

By leveraging data science techniques, this study highlights how streaming platforms can optimize content strategies and support artists in reaching a wider audience. Overall, the project demonstrates the power of data-driven approaches in the evolving music industry.

• <u>**REFERENCES**</u>

- Spotify. (n.d.). *Spotify for Developers: Web API*. Retrieved from https://developer.spotify.com/documentation/web-api/
- VanderPlas, J. (2016). *Python Data Science Handbook: Essential Tools for Working with Data*. O'Reilly Media.
- Raschka, S., & Mirjalili, V. (2019). *Python Machine Learning* (3rd ed.). Packt Publishing.
- Chen, M., Mao, S., & Liu, Y. (2014). Big data: A survey. *Mobile Networks and Applications*, 19(2), 171–209.
- Aggarwal, C. C. (2018). *Neural Networks and Deep Learning: A Textbook*. Springer.
- Jupyter Project. (n.d.). *Project Jupyter*. Retrieved from https://jupyter.org/
- Dash, S., Shakyawar, S. K., Sharma, M., & Kaushik, S. (2019). Big data in healthcare: Management, analysis and future prospects. *Journal of Big Data*, 6(1), 1–25.
- Seaborn Documentation. (n.d.). *Seaborn: Statistical data visualization in Python*. Retrieved from https://seaborn.pydata.org/