S. T. U. S. M's

SANGOLA COLLEGE, SANGOLA

A

PROJECT REPORT

ON

"MUSIC RECOMMENDATION SYSTEM"

SUBMITTED BY

Mr.Navale Aditya Malhari

Mr.Kashid Shubham Tanaji

UNDER THE GUIDENCE OF
Mr.Pailwan.G.A

SUBMITTED TO

PUNYASHLOK AHILYADEVI HOLKAR UNIVERSITY, SOLAPUR

IN PARTIAL FULFILLMENT

B.Sc.(Entire Computer Science)-III

2023-2024

submitted by

Mr. Navale Aditya Malhari

Mr.Kashid Shubham Tanaji

Of

Sangola College, Sangola

Guided By

Mr.Pailwan.G.A.

CERTIFICATE

Seat No. : - **Date:**

This is certify that *Mr.Navale Aditya Malhari* and *Mr.Kashid Shubham Tanaji* has contributed in successfully completed the project titled“MUSIC RECOMMENDATION SYSTEM” in satisfactory manner as a partial fulfillment for the course of B.Sc.(Entire Computer Science)-III for the session 2023-2024 of the university of Solapur.

[Project Guide] [Head of Department]

Examiner 1 **Examiner2**

ACKNOWLEDGEMENT

Before starting to project report we have to thanks to

Dept. Of Computer Science

For giving us to change to present ourselves with great potential and for providing us glorious platform towards our future career.

It gives me great pleasure to remain deeply inspected to our

Project guide Prof. Pailwan.G.A. Under whose guidance we completed the project.

The faith and confidence shown by him in our boosted our moral and motivated us to perform better in preparing this project.

We are thankful to all staff members for their valuable suggestion in completing this work.

We are also thankful to my all friends who help me directly or indirectly to carry this project successfully. Thanks must finally to our family especially to our parents, who are always with us.

Mr.Navale A.M and Mr.Kashid S.T.

INDEX

ABSTRACT

As streaming platforms have become more and more popular in recent years and music consumption has increased, music recommendation has become an increasingly relevant issue. Music applications are attempting to improve their recommendation systems in order to offer their users the best possible listening experience and keep them on their platform. For this purpose, two main models have emerged, collaborative filtering and content-based model. In the former, recommendations are based on similarity computations between users and their musical tastes. The main issue with this method is called cold start, it describes the fact that the system will not perform well on new items, whether music or users. In the latter, it is a matter of extracting information from the music itself in order to recommend a similar one.

It is the second method that has been implemented in this thesis. The state of the art of content-based methods reveals that the features that can be extracted are numerous. Indeed, there are low level features that can be temporal , spectral or even perceptual that require knowledge of physics and signal processing. There are middle level features that can be understood by musical experts like rhythm, pitch etc. Finally, there are higher level features, understandable by all mood, danceability. It should be underlined that the models identified during the paper readings step are also abundant.

Using the datasets from Kaggle , we will aim to first find the best model by focusing only on supervised models as well as their hyperparameters to achieve a relevant recommendation. On the other hand it is also necessary to determine the best subset of features to characterise the music while avoiding redundant and parasitic information. One of the main challenges is to find a way to assess the performance of our system.

STEPS FOR MAKING MUSIC RECOMMENDATION SYSTEM

- Problem Formulation

- Data Collection and Preprocessing

- Exploratory Data Analysis (EDA)

- Model Building

- Recommendation song

PROBLEM FORMULATION

- **Objective:**

The objective of the music recommendation system is to provide personalized recommendations to users based on their preferences and listening history, with the goal of enhancing user satisfaction and engagement on the music platform.

- **Scope:**

The recommendation system will focus on recommending songs or music tracks to users based on various factors such as genre preferences, artist preferences, mood, and user listening history.

The system will consider both explicit feedback like user ratings, likes, playlists and implicit feedback likes user interactions, skip patterns to generate personalized recommendations.

- **Target Audience:**

The target audience includes users of the music platform who are interested in discovering new music, exploring different genres, and creating personalized playlists.

➢ Data Sources:

The recommendation system will utilize data sources such as user profiles, listening history, song metadata e.g., genre, artist, release date, user interactions e.g., likes, skips, playlists and any additional contextual information e.g., time of day, location, device type.

➢ Problem Statement:

Develop a music recommendation system that can accurately predict and suggest songs or music tracks that users are likely to enjoy based on their individual preferences and listening behavior.

The system should address the cold start problem for new users by providing relevant recommendations even with limited user interaction data.

It should also ensure diversity and novelty in recommendations to encourage exploration and discovery of new music.

➢ Key Challenges:

1. Cold Start Problem: Providing relevant recommendations for new users with limited interaction history.
2. Sparsity and Scalability: Dealing with sparse user-item interaction data and efficiently scaling the recommendation system to handle a large volume of users and songs.
3. Diversity and Serendipity: Ensuring that recommendations are diverse and introduce users to new and unexpected music that aligns with their interests.
4. Real-Time Recommendation: Providing recommendations in real-time to accommodate user preferences and changes in listening behavior.

➢ Evaluation Metrics:

Performance of the recommendation system can be evaluated using metrics such as precision, recall, F1-score, mean average precision (MAP), normalized discounted cumulative gain (NDCG), and user engagement metric.

➢ **Success Criteria:**

The success of the music recommendation system will be determined by its ability to provide relevant, diverse, and engaging recommendations that enhance user satisfaction and retention on the music platform.

Improvement in user engagement metrics such as increased time spent on the platform, higher user retention rates, and increased interaction with recommended content.

By formulating the problem for the music recommendation system in this manner, we can establish clear objectives, scope, and success criteria for the development and evaluation of the recommendation system. This will guide the implementation of algorithms and techniques to address the challenges and meet the needs of the target audience effectively.

TYPES OF RECOMMENDATION SYSTEMS

Recommender systems are algorithms providing personalized suggestions for items that are most relevant to each user. With the massive growth of available online contents, users have been inundated with choices. It is therefore crucial for web platforms to offer recommendations of items to each user, in order to increase user satisfaction and engagement.

The following list shows examples of well-known web platforms with a huge number of available contents, which need efficient recommender systems to keep users interested.

1. Youtube:

   Every minute people upload 500 hours of videos, i.e. it would take 82 years to a user to watch all videos uploaded just in the last hour.
2. Spotify:

   Users can listen to ore than 80 million song tracks and podcasts.
3. Amazon:

   Users can buy more than 350 million different products.

All these platforms use powerful machine learning models in order to generate relevant recommendations for each user.

Recommender system can be classified according to the kind of information used to predict user preferences as Content-Based or Collaborative Filtering.

1. Content-Based Recommendation System:

Content-based filtering is a type of recommendation system that suggests items to users based on the characteristics or features of the items themselves and the user's preferences. It relies on analysing the content of items and recommending items that are similar to those the user has interacted with positively in the past. This approach is commonly used in various domains such as e-commerce, music streaming, and news recommendation.

content-based filtering typically works:

1. Item representation:

Each item in the system is represented by a set of features or attributes. These features can vary depending on the type of items being recommended.

For example, in the case of movies, features could include genres, actors, directors, and plot keywords. In the case of news articles, features could include keywords, topics, authors, and publication date.

2. User profile creation:

The system maintains a user profile that contains information about the user's preferences or interests. This profile is typically represented as a vector of weights assigned to different features based on the user's past interactions with items. For example, if a user has liked or rated several action movies highly in the past, the system might assign higher weights to features related to action genres or specific actors in the user's profile.

3. Similarity Calculation:

To generate recommendations, the system calculates the similarity between the user's profile and each item in the catalog. Various similarity metrics can be used for this purpose, such as cosine similarity, Jaccard similarity, or Euclidean distance. The similarity score indicates how closely an item matches the user's preferences based on their profile.

4. Ranking and Recommendation:

Once the similarity scores are calculated, the system ranks the items based on their similarity to the user's profile and recommends the top-ranked items to the user. The number of items recommended can be customized based on factors such as user preferences and system constraints.

Advantages of content-based filtering include:

• No Dependency on Other Users: Content-based filtering does not require information about other users' preferences or behaviors, making it suitable for new users or users with unique tastes.

• Transparency: Recommendations are based on explicit features of items and user preferences, making it easier to understand why certain items are recommended.

• Recommendations for Niche Items: Content-based filtering can recommend niche or less popular items based on their content characteristics, which might not be well-suited for collaborative filtering approaches.

However, content-based filtering also has limitations:

• Limited Serendipity: Recommendations are based on the user's past interactions and preferences, which can lead to a lack of serendipity or discovery of new items outside the user's known preferences.

• Limited Diversity: Since recommendations are based on item features similar to those the user has interacted with in the past, content-based filtering may struggle to recommend diverse or unexpected items.

• Cold Start Problem: Content-based filtering may face challenges when dealing with new users or items with insufficient data to generate meaningful recommendations.

Overall, content-based filtering is a valuable approach for providing personalized recommendations based on the intrinsic characteristics of items and user preferences. It can be used independently or in combination with other recommendation techniques to enhance recommendation quality and diversity.

2. Collaborative Filtering

Collaborative filtering is a recommendation system technique that makes predictions about the interests of a user by collecting preferences from many users (collaborating). It is based on the idea that users who have agreed in the past tend to agree again in the future. Collaborative filtering methods do not require item features or explicit ratings; they only rely on the user-item interaction data.

There are two main types of collaborative filtering:

1. **User-based Collaborative Filtering:**

In user-based collaborative filtering, recommendations are generated based on the similarity between users. The system identifies users who have similar preferences or behaviors to the target user and recommends items that those similar users have liked or interacted with.

The similarity between users is typically calculated using metrics such as Pearson correlation coefficient, cosine similarity, or Jaccard similarity.

For example, if user A and user B have rated several movies similarly in the past, and user A has not seen a particular movie that user B has rated highly, then the system may recommend that movie to user A.

2. **Item-based Collaborative Filtering:**

In item-based collaborative filtering, recommendations are generated based on the similarity between items. The system identifies items that are similar to the ones the user has liked or interacted with in the past and recommends those similar items.

The similarity between items is calculated based on the ratings given by users to both items, using metrics such as cosine similarity or Pearson correlation coefficient.

For example, if a user has positively rated a particular movie, the system identifies other movies that have been highly rated by users who liked the same movie and recommends those similar movies to the user.

Advantages of collaborative filtering include:

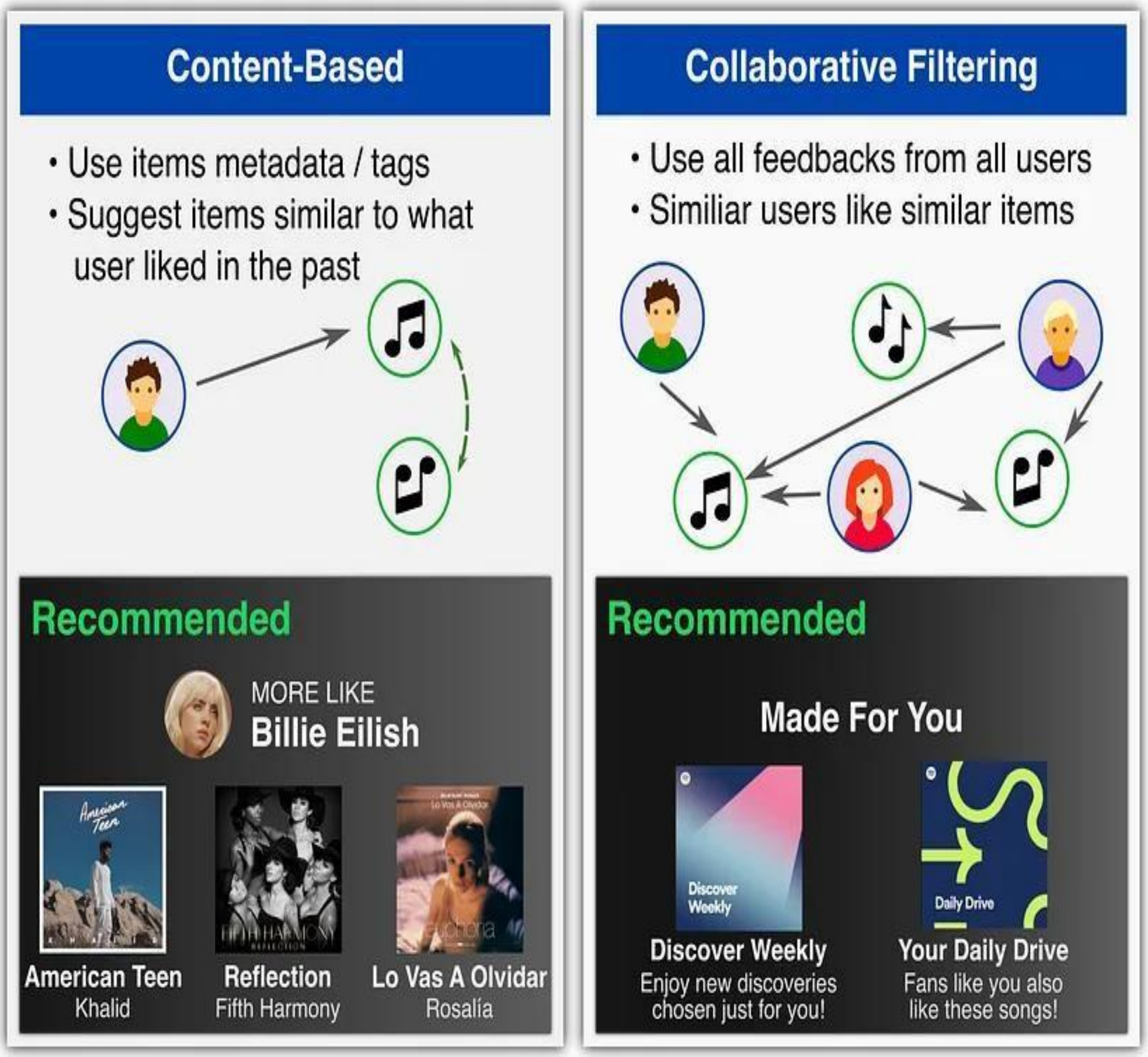
- No Need for Item Features: Collaborative filtering does not require explicit information about the items being recommended. It solely relies on user-item interaction data, making it suitable for scenarios where item features are unavailable or difficult to obtain.
- Serendipity: Collaborative filtering can recommend items that a user may not have discovered otherwise, based on the preferences of similar users.
- Scalability: Collaborative filtering can scale to large datasets and handle a high volume of users and items.

limitations:

- Cold Start Problem: It may struggle to provide accurate recommendations for new users who have not yet provided sufficient interaction data.
- Sparsity: If the user-item interaction data is sparse (i.e., many users have rated only a small fraction of items), collaborative filtering may struggle to identify similar users or items accurately.
- Popularity Bias: Collaborative filtering tends to recommend popular items more frequently, which can lead to a lack of diversity in recommendations.

Overall, collaborative filtering is a powerful technique for generating personalized recommendations based on the preferences of similar users or items. It can be used independently or in combination with other recommendation techniques to improve recommendation quality and diversity.

Difference between content based and collaborative based filtering :



Content-Based

- Use items metadata / tags
- Suggest items similar to what user liked in the past

Collaborative Filtering

- Use all feedbacks from all users
- Similiar users like similar items

DATA COLLECTION AND PREPROCESSING

Data collection

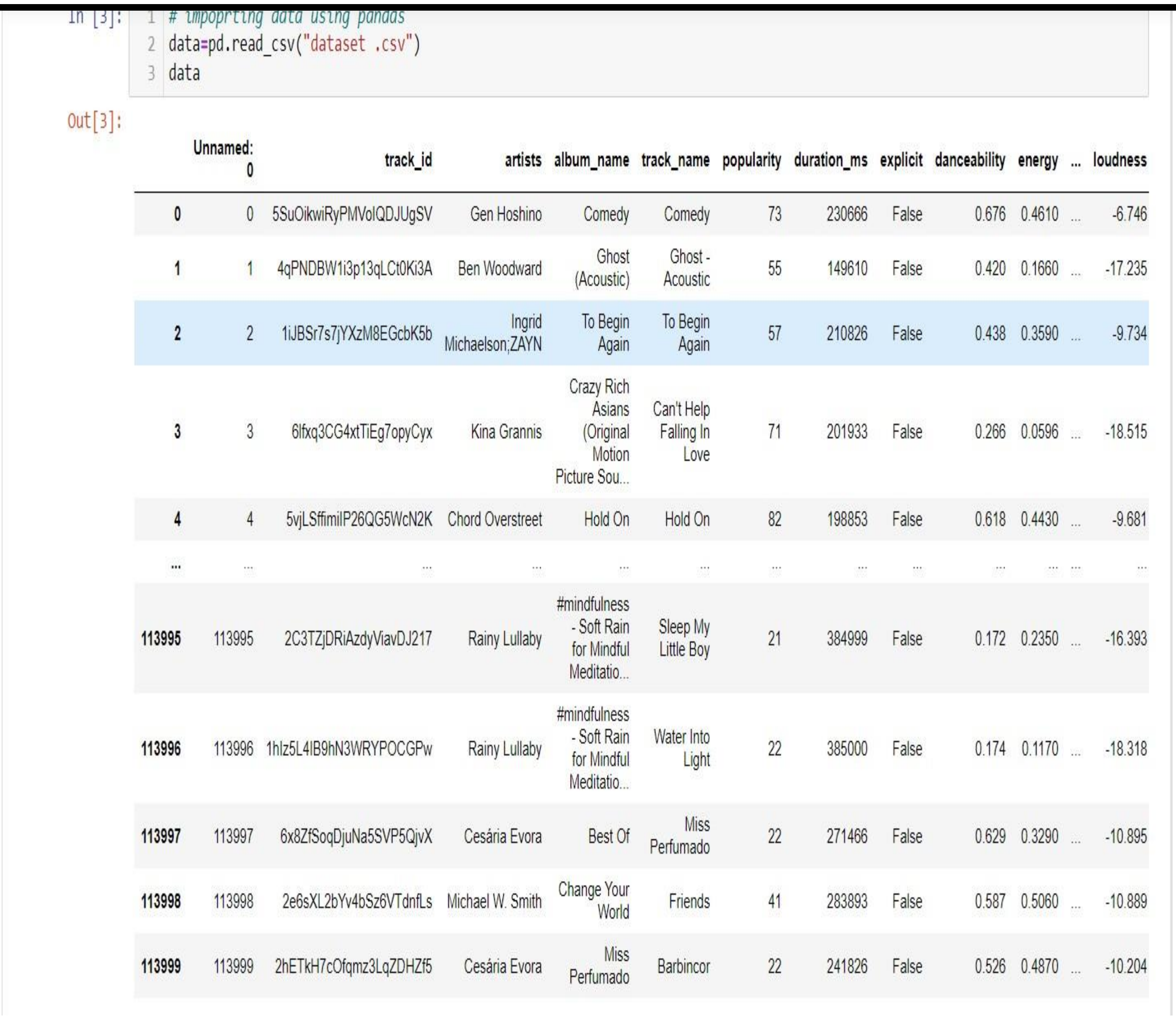
- We collected data set from Kaggle website . Our dataset consist of lot of data with we used usefull thinks like artists, album name, track name popularity and gener of song is also available.

Data preprocessing

1. For Import data .

```
import pandas as pd
    # Importing the dataset
  data = pd.read_csv("dataset.csv")
# Display brief information about the dataset
  print(data.info())
```

- We import the pandas library using the alias pd.
- We then use the pd.read_csv() function to read the dataset from the CSV file named "dataset.csv". Make sure to replace "dataset.csv" with the actual filename and path if it's located in a different directory.
- Finally, we use the info() method to display brief information about the dataset, including the number of rows and columns, column names, non-null counts, and data types of each column.



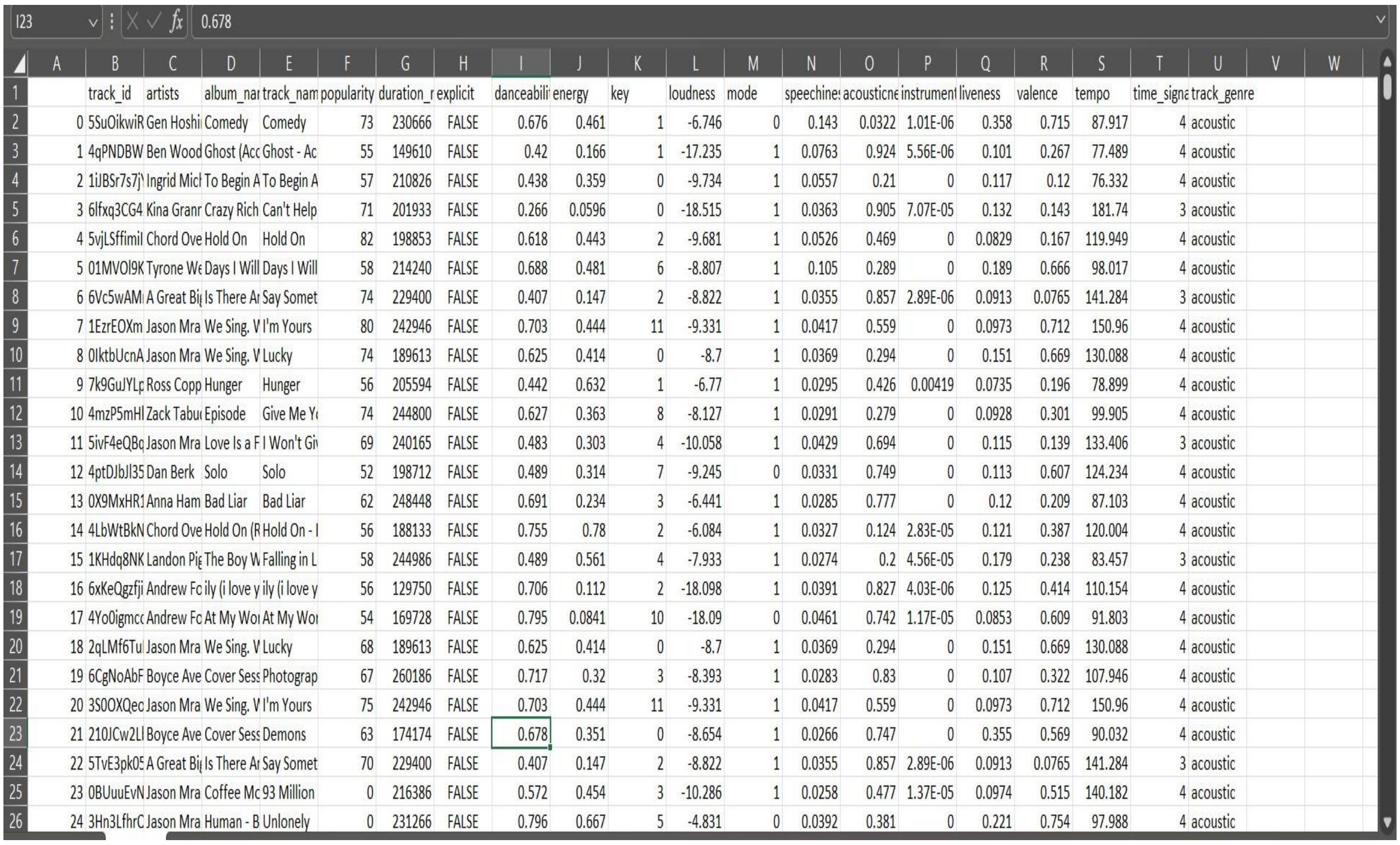
```
In [3]:  # impoprting data using pandas
         data=pd.read_csv("dataset .csv")
         data
```

Out[3]:

	Unnamed: 0	track_id	artists	album_name	track_name	popularity	duration_ms	explicit	danceability	energy	...	loudness
0	0	5SuOikwiRyPMVoIQDJUgSV	Gen Hoshino	Comedy	Comedy	73	230666	False	0.676	0.4610	...	-6.746
1	1	4qPNDBW1i3p13qLCt0Ki3A	Ben Woodward	Ghost (Acoustic)	Ghost - Acoustic	55	149610	False	0.420	0.1660	...	-17.235
2	2	1iJBSr7s7jYXzM8EGcbK5b	Ingrid Michaelson;ZAYN	To Begin Again	To Begin Again	57	210826	False	0.438	0.3590	...	-9.734
3	3	6lfxq3CG4xtTiEg7opyCyx	Kina Grannis	Crazy Rich Asians (Original Motion Picture Sou...	Can't Help Falling In Love	71	201933	False	0.266	0.0596	...	-18.515
4	4	5vjLSffimiIP26QG5WcN2K	Chord Overstreet	Hold On	Hold On	82	198853	False	0.618	0.4430	...	-9.681
...	...	...	...	...	...	...	...	...	...	...	...	...
113995	113995	2C3TZjDRiAzdyViavDJ217	Rainy Lullaby	#mindfulness - Soft Rain for Mindful Meditatio...	Sleep My Little Boy	21	384999	False	0.172	0.2350	...	-16.393
113996	113996	1hIz5L4IB9hN3WRYPOCGPw	Rainy Lullaby	#mindfulness - Soft Rain for Mindful Meditatio...	Water Into Light	22	385000	False	0.174	0.1170	...	-18.318
113997	113997	6x8ZfSoqDjuNa5SVP5QjvX	Cesária Evora	Best Of	Miss Perfumado	22	271466	False	0.629	0.3290	...	-10.895
113998	113998	2e6sXL2bYv4bSz6VTdnfLs	Michael W. Smith	Change Your World	Friends	41	283893	False	0.587	0.5060	...	-10.889
113999	113999	2hETkH7cOfqmz3LqZDHZf5	Cesária Evora	Miss Perfumado	Barbincor	22	241826	False	0.526	0.4870	...	-10.204

- This code will load the dataset into a pandas DataFrame named data and display a summary of its information. Make sure to replace "dataset.csv" with the actual filename and path of your dataset file

Dataset from Kaggle:

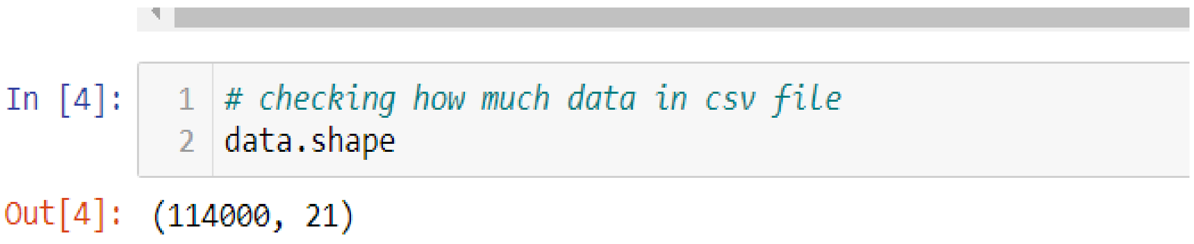


I23 0.678

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
1		track_id	artists	album_nar	track_nam	popularity	duration_r	explicit	danceabili	energy	key	loudness	mode	speechines	acousticne	instrument	liveness	valence	tempo	time_signa	track_genre		
2	0	5SuOikwiR	Gen Hoshi	Comedy	Comedy	73	230666	FALSE	0.676	0.461	1	-6.746	0	0.143	0.0322	1.01E-06	0.358	0.715	87.917	4	acoustic		
3	1	4qPNDBW	Ben Wood	Ghost (Acc	Ghost - Ac	55	149610	FALSE	0.42	0.166	1	-17.235	1	0.0763	0.924	5.56E-06	0.101	0.267	77.489	4	acoustic		
4	2	1iJBSr7s7j	Ingrid Mich	To Begin A	To Begin A	57	210826	FALSE	0.438	0.359	0	-9.734	1	0.0557	0.21	0	0.117	0.12	76.332	4	acoustic		
5	3	6lfxq3CG4	Kina Granr	Crazy Rich	Can't Help	71	201933	FALSE	0.266	0.0596	0	-18.515	1	0.0363	0.905	7.07E-05	0.132	0.143	181.74	3	acoustic		
6	4	5vjLSffimil	Chord Ove	Hold On	Hold On	82	198853	FALSE	0.618	0.443	2	-9.681	1	0.0526	0.469	0	0.0829	0.167	119.949	4	acoustic		
7	5	01MVOl9K	Tyrone We	Days I Will	Days I Will	58	214240	FALSE	0.688	0.481	6	-8.807	1	0.105	0.289	0	0.189	0.666	98.017	4	acoustic		
8	6	6Vc5wAM	A Great Bi	Is There Ar	Say Somet	74	229400	FALSE	0.407	0.147	2	-8.822	1	0.0355	0.857	2.89E-06	0.0913	0.0765	141.284	3	acoustic		
9	7	1EzrEOXm	Jason Mra	We Sing. V	I'm Yours	80	242946	FALSE	0.703	0.444	11	-9.331	1	0.0417	0.559	0	0.0973	0.712	150.96	4	acoustic		
10	8	0lktbUcnA	Jason Mra	We Sing. V	Lucky	74	189613	FALSE	0.625	0.414	0	-8.7	1	0.0369	0.294	0	0.151	0.669	130.088	4	acoustic		
11	9	7k9GuJYLp	Ross Copp	Hunger	Hunger	56	205594	FALSE	0.442	0.632	1	-6.77	1	0.0295	0.426	0.00419	0.0735	0.196	78.899	4	acoustic		
12	10	4mzP5mHl	Zack Tabu	Episode	Give Me Yo	74	244800	FALSE	0.627	0.363	8	-8.127	1	0.0291	0.279	0	0.0928	0.301	99.905	4	acoustic		
13	11	5ivF4eQBq	Jason Mra	Love Is a F	I Won't Giv	69	240165	FALSE	0.483	0.303	4	-10.058	1	0.0429	0.694	0	0.115	0.139	133.406	3	acoustic		
14	12	4ptDJbJl35	Dan Berk	Solo	Solo	52	198712	FALSE	0.489	0.314	7	-9.245	0	0.0331	0.749	0	0.113	0.607	124.234	4	acoustic		
15	13	0X9MxHR1	Anna Ham	Bad Liar	Bad Liar	62	248448	FALSE	0.691	0.234	3	-6.441	1	0.0285	0.777	0	0.12	0.209	87.103	4	acoustic		
16	14	4LbWtBkN	Chord Ove	Hold On (R	Hold On - I	56	188133	FALSE	0.755	0.78	2	-6.084	1	0.0327	0.124	2.83E-05	0.121	0.387	120.004	4	acoustic		
17	15	1KHdq8NK	Landon Pig	The Boy W	Falling in L	58	244986	FALSE	0.489	0.561	4	-7.933	1	0.0274	0.2	4.56E-05	0.179	0.238	83.457	3	acoustic		
18	16	6xKeQgzfji	Andrew Fo	ily (i love y	ily (i love y	56	129750	FALSE	0.706	0.112	2	-18.098	1	0.0391	0.827	4.03E-06	0.125	0.414	110.154	4	acoustic		
19	17	4Yo0igmcc	Andrew Fo	At My Wor	At My Wor	54	169728	FALSE	0.795	0.0841	10	-18.09	0	0.0461	0.742	1.17E-05	0.0853	0.609	91.803	4	acoustic		
20	18	2qLMf6Tul	Jason Mra	We Sing. V	Lucky	68	189613	FALSE	0.625	0.414	0	-8.7	1	0.0369	0.294	0	0.151	0.669	130.088	4	acoustic		
21	19	6CgNoAbF	Boyce Ave	Cover Sess	Photograp	67	260186	FALSE	0.717	0.32	3	-8.393	1	0.0283	0.83	0	0.107	0.322	107.946	4	acoustic		
22	20	3S0OXQeo	Jason Mra	We Sing. V	I'm Yours	75	242946	FALSE	0.703	0.444	11	-9.331	1	0.0417	0.559	0	0.0973	0.712	150.96	4	acoustic		
23	21	210JCw2Ll	Boyce Ave	Cover Sess	Demons	63	174174	FALSE	0.678	0.351	0	-8.654	1	0.0266	0.747	0	0.355	0.569	90.032	4	acoustic		
24	22	5TvE3pk05	A Great Bi	Is There Ar	Say Somet	70	229400	FALSE	0.407	0.147	2	-8.822	1	0.0355	0.857	2.89E-06	0.0913	0.0765	141.284	3	acoustic		
25	23	0BUuuEvN	Jason Mra	Coffee Mo	93 Million	0	216386	FALSE	0.572	0.454	3	-10.286	1	0.0258	0.477	1.37E-05	0.0974	0.515	140.182	4	acoustic		
26	24	3Hn3LfhrC	Jason Mra	Human - B	Unlonely	0	231266	FALSE	0.796	0.667	5	-4.831	0	0.0392	0.381	0	0.221	0.754	97.988	4	acoustic		

2. Checking how much data in csv file

data.shape
# (114000, 21)



```
In [4]:  # checking how much data in csv file
         data.shape
Out[4]:  (114000, 21)
```

Exploratory Data Analysis

- The code data.shape simply returns the shape of the DataFrame data, where the number of rows and columns is represented as a tuple (num_rows, num_columns).
- For instance, if you have a DataFrame data and you execute data.shape, it might return something like (100, 5), which means there are 100 rows and 5 columns in your DataFrame.

This function is useful for quickly understanding the size of your dataset.

EXPLORATORY DATA ANALYSIS

Exploratory Data Analysis (EDA) is a critical step in the data analysis process. It involves summarizing the main characteristics of a dataset, often using graphical and statistical techniques. EDA helps analysts understand the structure of the data, identify patterns, spot anomalies, formulate hypotheses, and select appropriate modeling techniques.

Here's a basic outline of steps typically involved in EDA:

Load Data: Import your dataset into your preferred data analysis environment (Python, R, etc.).

Understand the Data: Get a high-level understanding of the dataset's structure and content. Check the first few rows of the dataset using functions like head() (in pandas). View summary statistics using functions like describe() (in pandas)

Check data types, missing values, and unique values in each column.

Handle Missing Data: Identify and deal with missing values appropriately. Options include imputation, deletion, or modeling missingness.

Explore Distributions: Examine the distributions of individual variables through histograms, density plots, boxplots, etc. This step helps you understand the central tendency, spread, and skewness of each variable.

Analyze Relationships: Investigate relationships between variables.

Use scatter plots to visualize the relationship between continuous variables.

Employ boxplots or violin plots to compare distributions across different categories. Calculate correlation coefficients to quantify the strength and direction of linear relationships.

Explore Outliers: Identify and understand outliers in the dataset. Decide whether to remove, transform, or keep them based on domain knowledge and analysis goals.

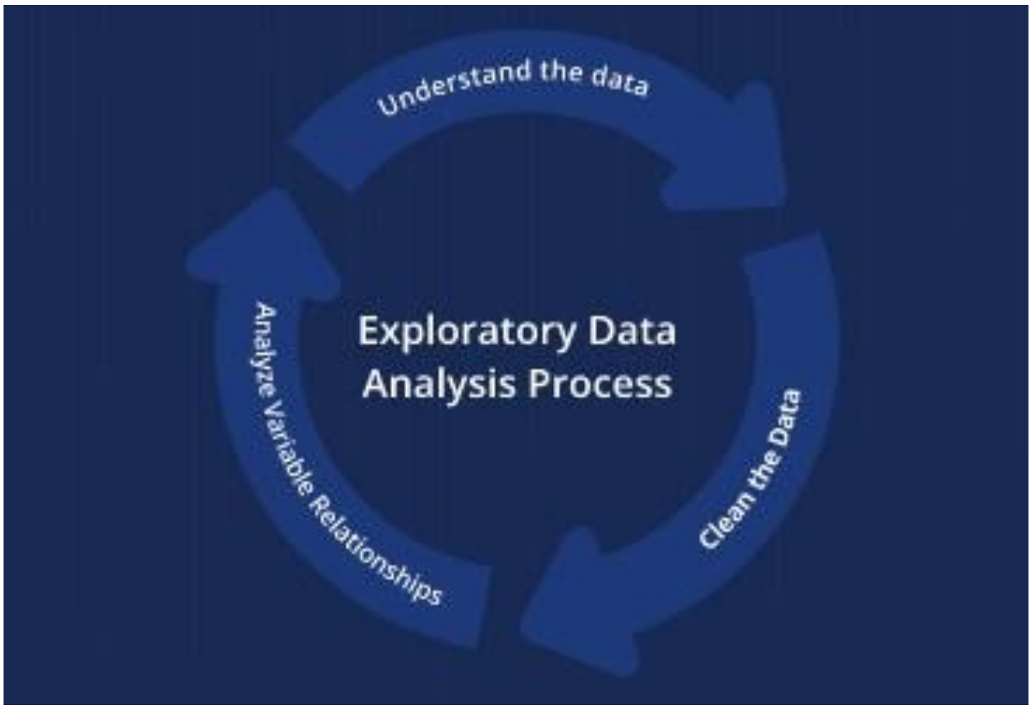
Visualize Trends and Patterns: Use various visualization techniques to uncover trends, patterns, and anomalies in the data. Techniques include time series plots, heatmaps, pair plots, and more.

Feature Engineering: Create new features from existing ones to improve model performance or uncover hidden patterns.

Dimensionality Reduction: Use techniques like PCA (Principal Component Analysis) or t-SNE (t-distributed Stochastic Neighbor Embedding) to reduce the dimensionality of the dataset and visualize high-dimensional data.

Summarize Findings: Summarize key insights and findings from the exploratory analysis. Document any important observations, patterns, or outliers discovered during the process.

Remember that EDA is an iterative process, and the steps mentioned above are not strictly sequential. Analysts often revisit previous steps as they gain more insights into the data. Additionally, the specific techniques and methods used in EDA may vary depending on the dataset and the analysis goals.

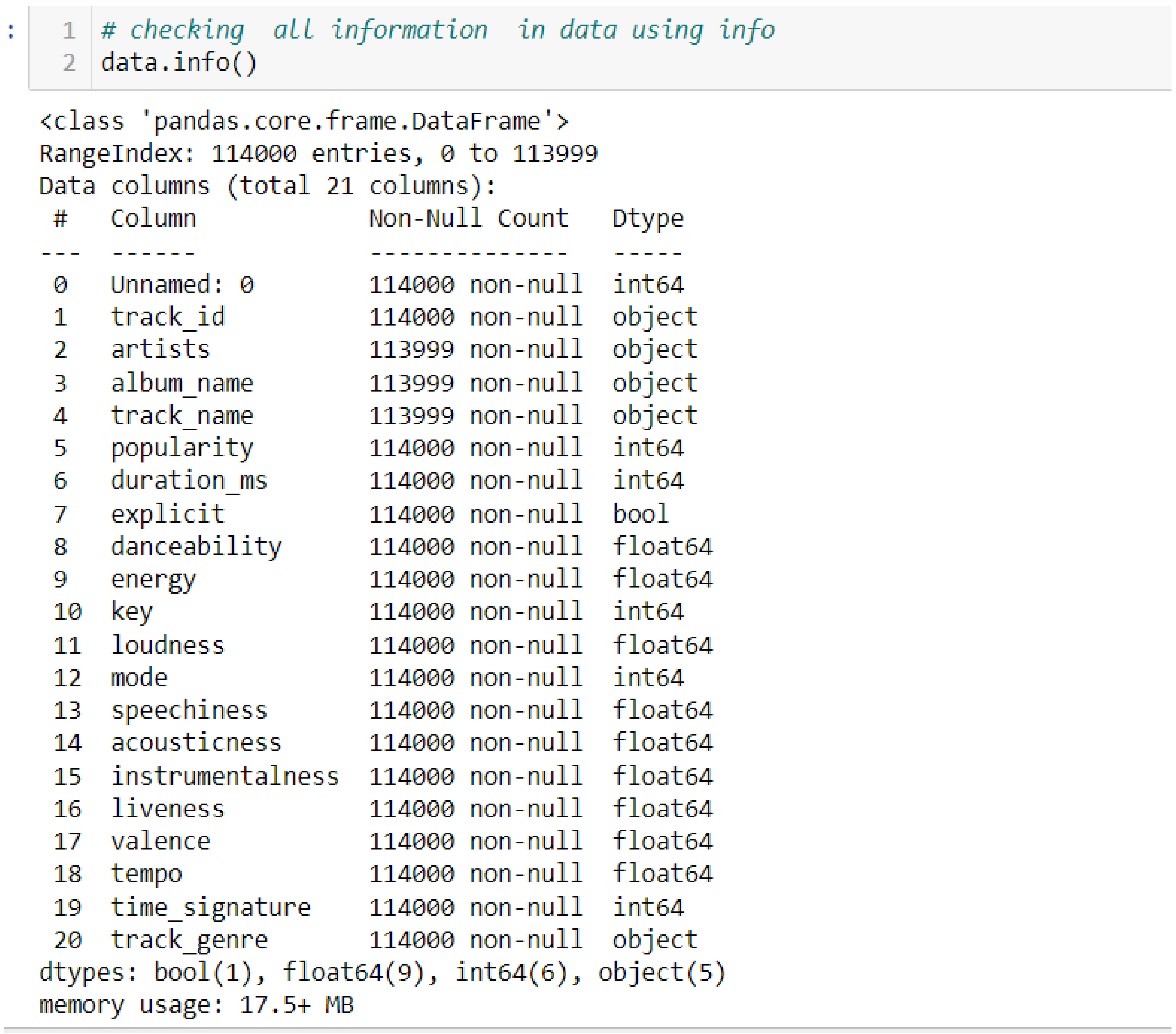

Exploratory Data Analysis Process
Understand the data
Clean the Data
Analyze Variable Relationships

➢ Data cleaning :

data.info()

checking all information in data using info

The code data.info() is a method in pandas DataFrame that provides a concise summary of the data, including the number of non-null values, data types, and memory usage. Here's a breakdown of what data.info() does:



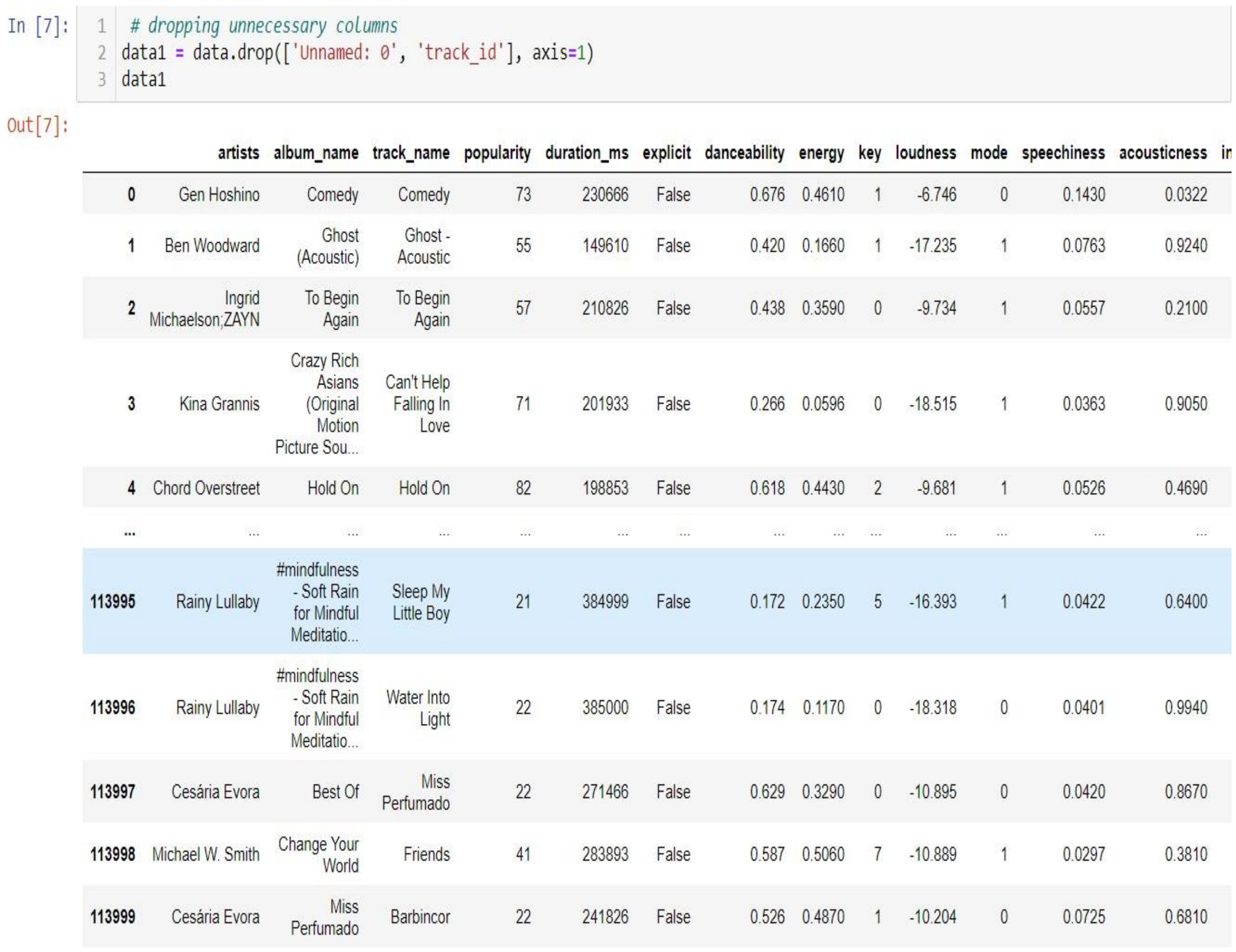
```
# checking  all information  in data using info
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 114000 entries, 0 to 113999
Data columns (total 21 columns):
 #   Column            Non-Null Count   Dtype  
---  ------            --------------   -----  
 0   Unnamed: 0        114000 non-null  int64  
 1   track_id          114000 non-null  object 
 2   artists           113999 non-null  object 
 3   album_name        113999 non-null  object 
 4   track_name        113999 non-null  object 
 5   popularity        114000 non-null  int64  
 6   duration_ms       114000 non-null  int64  
 7   explicit          114000 non-null  bool   
 8   danceability      114000 non-null  float64
 9   energy            114000 non-null  float64
 10  key               114000 non-null  int64  
 11  loudness          114000 non-null  float64
 12  mode              114000 non-null  int64  
 13  speechiness       114000 non-null  float64
 14  acousticness      114000 non-null  float64
 15  instrumentalness  114000 non-null  float64
 16  liveness          114000 non-null  float64
 17  valence           114000 non-null  float64
 18  tempo             114000 non-null  float64
 19  time_signature    114000 non-null  int64  
 20  track_genre       114000 non-null  object 
dtypes: bool(1), float64(9), int64(6), object(5)
memory usage: 17.5+ MB
```

- Data Type: It prints the data type of each column. This includes information such as whether the column contains integers, floats, strings, datetimes, or categorical data.
- Non-Null Count: It shows the number of non-null (non-missing) values present in each column. This information is crucial for understanding how complete the dataset is and identifying columns with missing data.
- Memory Usage: It displays the memory usage of the DataFrame. This is useful for assessing the memory footprint of the dataset, especially when dealing with large datasets.
- Total Number of Entries: It provides the total number of entries (rows) in the DataFrame.
- Column Names: It lists all column names present in the DataFrame.

➢ Dropping unnecessary columns:

```
data1 = data.drop(['Unnamed: 0', 'track_id'], axis=1)
data1
```



In [7]:
```
# dropping unnecessary columns
data1 = data.drop(['Unnamed: 0', 'track_id'], axis=1)
data1
```

Out[7]:

	artists	album_name	track_name	popularity	duration_ms	explicit	danceability	energy	key	loudness	mode	speechiness	acousticness	in
0	Gen Hoshino	Comedy	Comedy	73	230666	False	0.676	0.4610	1	-6.746	0	0.1430	0.0322	
1	Ben Woodward	Ghost (Acoustic)	Ghost - Acoustic	55	149610	False	0.420	0.1660	1	-17.235	1	0.0763	0.9240	
2	Ingrid Michaelson;ZAYN	To Begin Again	To Begin Again	57	210826	False	0.438	0.3590	0	-9.734	1	0.0557	0.2100	
3	Kina Grannis	Crazy Rich Asians (Original Motion Picture Sou...	Can't Help Falling In Love	71	201933	False	0.266	0.0596	0	-18.515	1	0.0363	0.9050	
4	Chord Overstreet	Hold On	Hold On	82	198853	False	0.618	0.4430	2	-9.681	1	0.0526	0.4690	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
113995	Rainy Lullaby	#mindfulness - Soft Rain for Mindful Meditatio...	Sleep My Little Boy	21	384999	False	0.172	0.2350	5	-16.393	1	0.0422	0.6400	
113996	Rainy Lullaby	#mindfulness - Soft Rain for Mindful Meditatio...	Water Into Light	22	385000	False	0.174	0.1170	0	-18.318	0	0.0401	0.9940	
113997	Cesária Evora	Best Of	Miss Perfumado	22	271466	False	0.629	0.3290	0	-10.895	0	0.0420	0.8670	
113998	Michael W. Smith	Change Your World	Friends	41	283893	False	0.587	0.5060	7	-10.889	1	0.0297	0.3810	
113999	Cesária Evora	Miss Perfumado	Barbincor	22	241826	False	0.526	0.4870	1	-10.204	0	0.0725	0.6810	

- The code data1 = data.drop(['Unnamed: 0', 'track_id'], axis=1) performs the operation of dropping two columns named 'Unnamed: 0' and 'track_id' from the DataFrame data and assigns the result to a new DataFrame data1.
- data.drop(['Unnamed: 0', 'track_id'], axis=1): This part of the code calls the drop() method on the DataFrame data.
- It takes a list of column names ['Unnamed: 0', 'track_id'] as the first argument, specifying which columns to drop.

- data1: This part of the code assigns the result of the operation (DataFrame with the specified columns dropped) to a new variable data1.

The resulting DataFrame data1 will be data with the columns 'Unnamed: 0' and 'track_id' removed. This can be useful for removing columns that are unnecessary for analysis or modeling.

➢ Checking duplicates values:

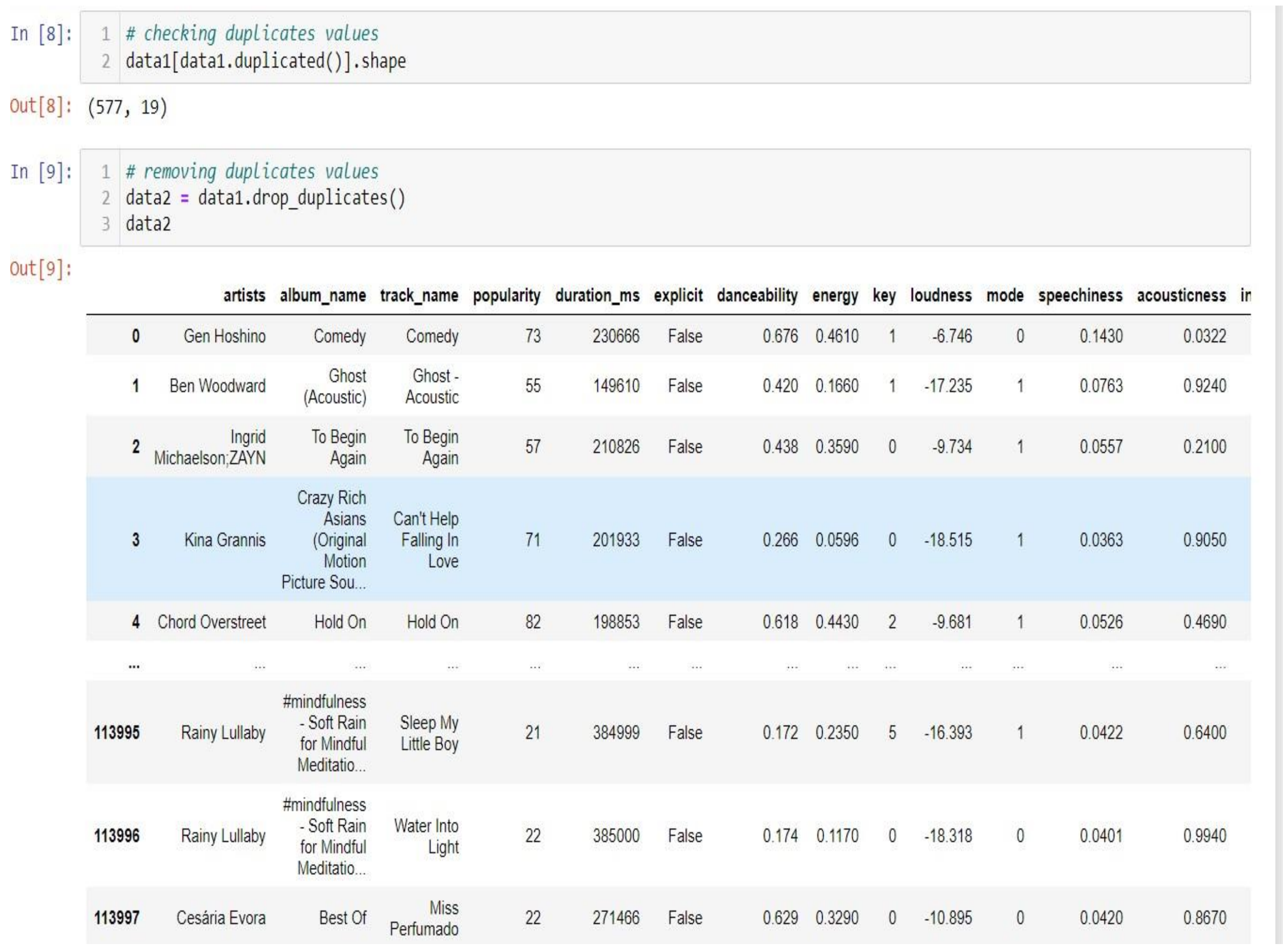
data1[data1.duplicated()].shape

removing duplicates values

data2 = data1.drop_duplicates()

data2

data1[data1.duplicated()].shape



```
In [8]:  # checking duplicates values
         data1[data1.duplicated()].shape

Out[8]: (577, 19)

In [9]:  # removing duplicates values
         data2 = data1.drop_duplicates()
         data2

Out[9]:
```

	artists	album_name	track_name	popularity	duration_ms	explicit	danceability	energy	key	loudness	mode	speechiness	acousticness	in
0	Gen Hoshino	Comedy	Comedy	73	230666	False	0.676	0.4610	1	-6.746	0	0.1430	0.0322	
1	Ben Woodward	Ghost (Acoustic)	Ghost - Acoustic	55	149610	False	0.420	0.1660	1	-17.235	1	0.0763	0.9240	
2	Ingrid Michaelson;ZAYN	To Begin Again	To Begin Again	57	210826	False	0.438	0.3590	0	-9.734	1	0.0557	0.2100	
3	Kina Grannis	Crazy Rich Asians (Original Motion Picture Sou...	Can't Help Falling In Love	71	201933	False	0.266	0.0596	0	-18.515	1	0.0363	0.9050	
4	Chord Overstreet	Hold On	Hold On	82	198853	False	0.618	0.4430	2	-9.681	1	0.0526	0.4690	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
113995	Rainy Lullaby	#mindfulness - Soft Rain for Mindful Meditatio...	Sleep My Little Boy	21	384999	False	0.172	0.2350	5	-16.393	1	0.0422	0.6400	
113996	Rainy Lullaby	#mindfulness - Soft Rain for Mindful Meditatio...	Water Into Light	22	385000	False	0.174	0.1170	0	-18.318	0	0.0401	0.9940	
113997	Cesária Evora	Best Of	Miss Perfumado	22	271466	False	0.629	0.3290	0	-10.895	0	0.0420	0.8670	

#This code checks for duplicate rows in the DataFrame data1.

- data1.duplicated() returns a boolean Series indicating whether each row is a duplicate of a previous row.
- data1[data1.duplicated()] filters the DataFrame to only include rows that are duplicates.
- .shape then returns the shape of this filtered DataFrame, which gives the number of duplicate rows and the number of columns.

- data2 = data1.drop_duplicates(): This line creates a new DataFrame data2 by removing duplicate rows from data1.
- drop_duplicates() is a method in pandas that removes duplicate rows from a DataFrame.
- By default, it keeps the first occurrence of each duplicated row and removes subsequent occurrences.
- data2: This simply displays the resulting DataFrame data2 after removing duplicate rows.

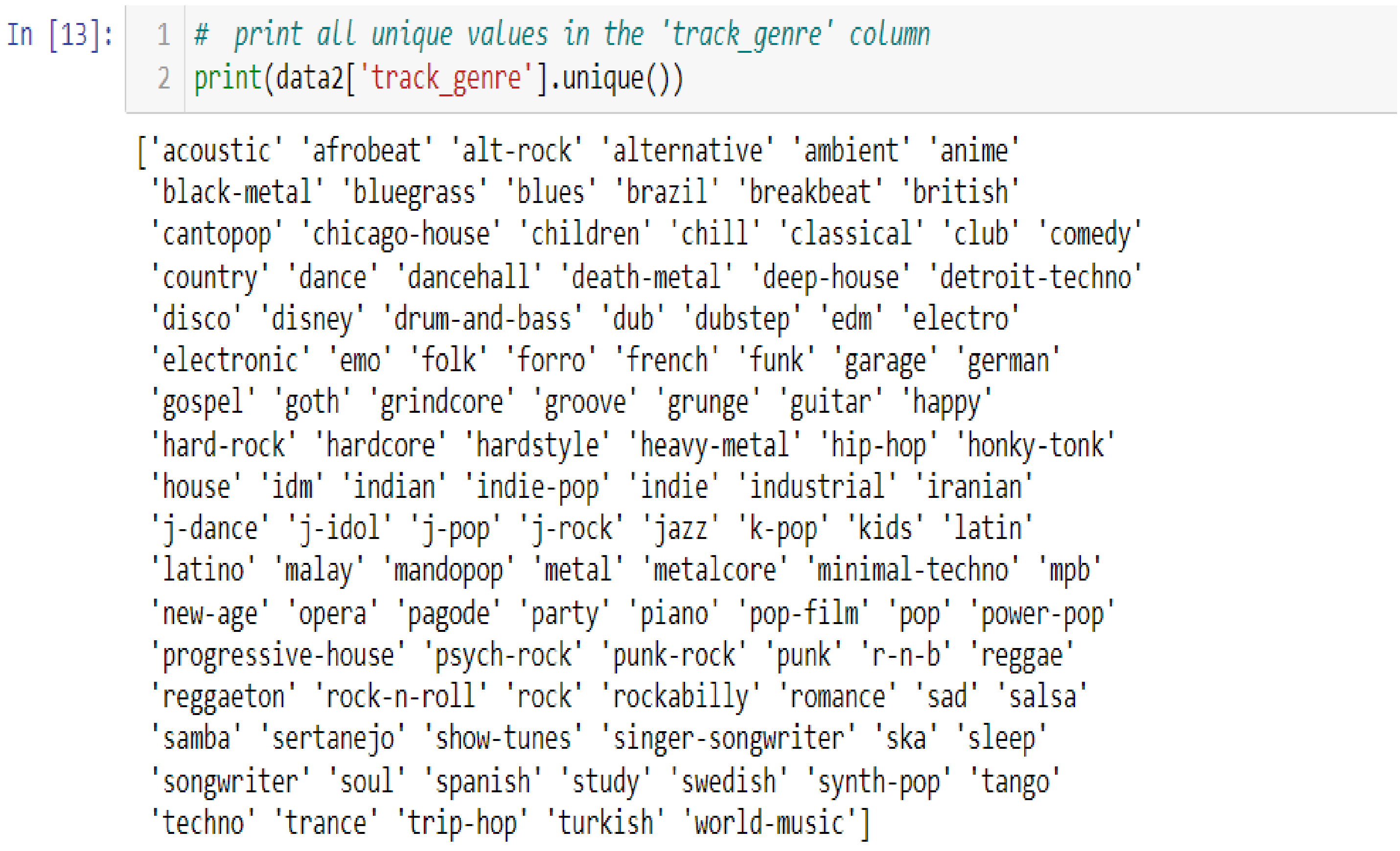
In summary, this code first checks how many duplicate rows are present in data1, then creates a new DataFrame data2 with those duplicate rows removed. This is useful for data cleaning to ensure that each row in the DataFrame is unique, which is often important for accurate analysis and modeling.

➢ All unique values in the 'track_genre' column :

print(data2['track_genre'].unique())

# used to obtain the counts of unique values in the 'track_genre' column of the DataFrame

data2['track_genre'].value_counts().head(20)



```
In [13]:  #  print all unique values in the 'track_genre' column
          print(data2['track_genre'].unique())
```

```
['acoustic' 'afrobeat' 'alt-rock' 'alternative' 'ambient' 'anime'
 'black-metal' 'bluegrass' 'blues' 'brazil' 'breakbeat' 'british'
 'cantopop' 'chicago-house' 'children' 'chill' 'classical' 'club' 'comedy'
 'country' 'dance' 'dancehall' 'death-metal' 'deep-house' 'detroit-techno'
 'disco' 'disney' 'drum-and-bass' 'dub' 'dubstep' 'edm' 'electro'
 'electronic' 'emo' 'folk' 'forro' 'french' 'funk' 'garage' 'german'
 'gospel' 'goth' 'grindcore' 'groove' 'grunge' 'guitar' 'happy'
 'hard-rock' 'hardcore' 'hardstyle' 'heavy-metal' 'hip-hop' 'honky-tonk'
 'house' 'idm' 'indian' 'indie-pop' 'indie' 'industrial' 'iranian'
 'j-dance' 'j-idol' 'j-pop' 'j-rock' 'jazz' 'k-pop' 'kids' 'latin'
 'latino' 'malay' 'mandopop' 'metal' 'metalcore' 'minimal-techno' 'mpb'
 'new-age' 'opera' 'pagode' 'party' 'piano' 'pop-film' 'pop' 'power-pop'
 'progressive-house' 'psych-rock' 'punk-rock' 'punk' 'r-n-b' 'reggae'
 'reggaeton' 'rock-n-roll' 'rock' 'rockabilly' 'romance' 'sad' 'salsa'
 'samba' 'sertanejo' 'show-tunes' 'singer-songwriter' 'ska' 'sleep'
 'songwriter' 'soul' 'spanish' 'study' 'swedish' 'synth-pop' 'tango'
 'techno' 'trance' 'trip-hop' 'turkish' 'world-music']
```

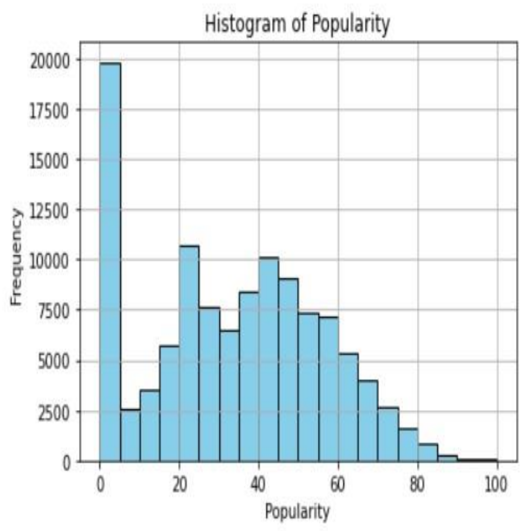
- The code data2['track_genre'].value_counts().head(20) is used to obtain the counts of unique values in the 'track_genre' column of the DataFrame data2, sorted in descending order, and then displays the top 20 most frequent genres.
- data2['track_genre']: This part of the code selects the 'track_genre' column from the DataFrame data2.
- .value_counts(): This method is applied to the 'track_genre' column, which counts the occurrences of each unique value in the column.
- .head(20): This method is used to select the first 20 rows (genres) from the resulting series, which corresponds to the 20 most frequent genres.

So, when you execute data2['track_genre'].value_counts().head(20), it will return a series where each index represents a unique genre, and the corresponding value represents the count of tracks belonging to that genre. This is helpful for understanding the distribution of genres in the dataset, especially focusing on the most common ones.

STATISTICAL OPERATIONS ON DATA

➢ HISTOGRAM:

```
# Import necessary libraries
import matplotlib.pyplot as plt
# Create a histogram with 20 bins, using the 'popularity' column
data2['popularity'].hist(bins=20, color='skyblue', edgecolor='black')

# Add title and labels to the plot
plt.title('Histogram of Popularity')
plt.xlabel('Popularity')
plt.ylabel('Frequency')

# Show the histogram
plt.show()
```



The purpose of bar plot is to provide a visual representation of how often each unique value appears in the 'popularity' column, giving you insights into the distribution of popularity values in your dataset.

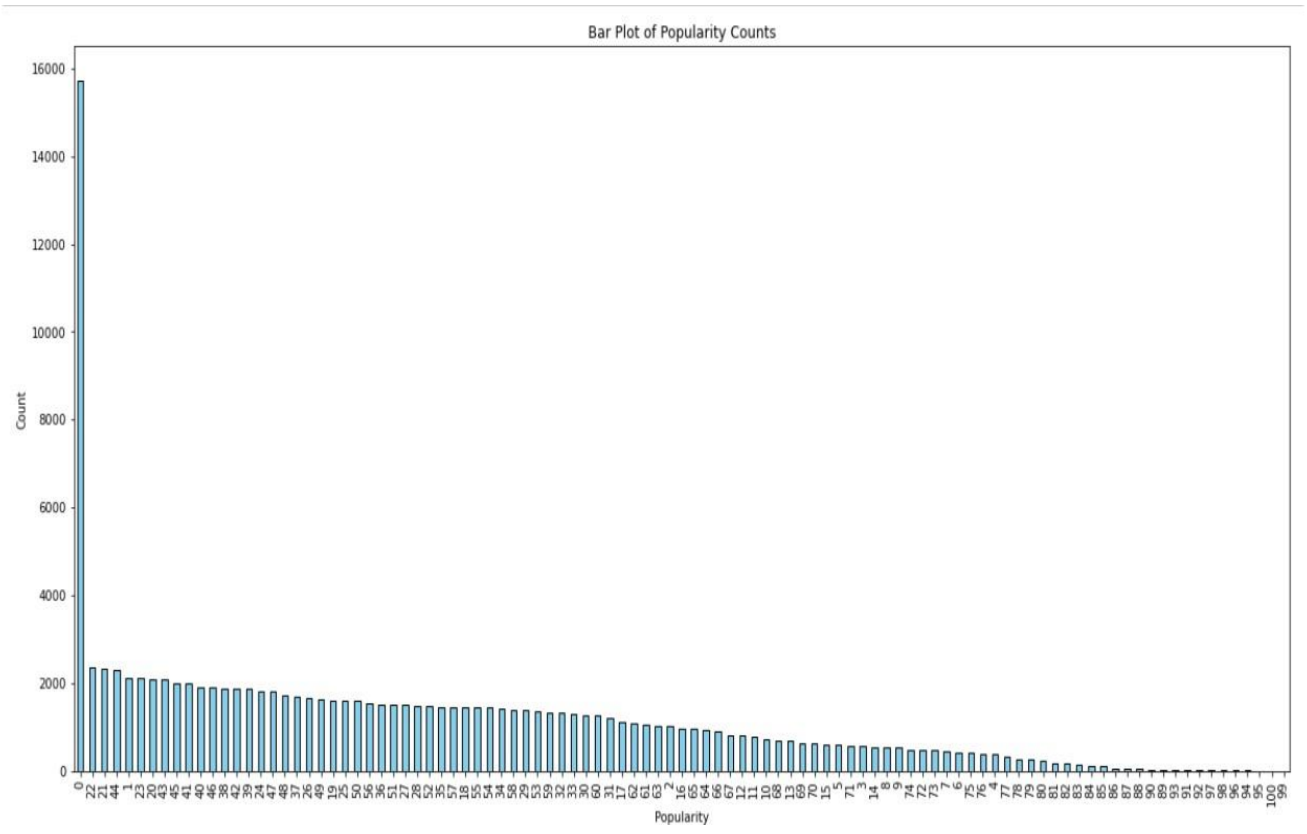
This code generates a histogram of the 'popularity' column in the DataFrame data2 using matplotlib, a plotting library for Python.

- Import Necessary Libraries: The code imports the matplotlib.pyplot module and assigns it the alias plt. This module provides a MATLAB-like plotting interface.
- Create Histogram: The data2['popularity'].hist(bins=20, color='skyblue', edgecolor='black') line creates a histogram of the 'popularity' column.
- bins=20 specifies the number of bins (bars) to divide the data into. In this case, it creates 20 bins.
- color='skyblue' sets the color of the bars in the histogram to sky blue.
- edgecolor='black' sets the color of the edges of the bars to black.
- Add Title and Labels: The plt.title('Histogram of Popularity'), plt.xlabel('Popularity'), and plt.ylabel('Frequency') lines add a title to the plot and label the x-axis and y-axis, respectively.
- Show the Plot: The plt.show() function displays the histogram plot.

Overall, this code generates a histogram that visualizes the distribution of popularity values in the dataset, with 20 bins to represent the data. The title and axis labels help to provide context for interpreting the plot.

➢ BAR PLOT:

```
data2['popularity'].value_counts().plot.bar(color='skyblue', edgecolor='black',
figsize=(20, 10))


# Add title and labels

plt.title('Bar Plot of Popularity Counts')

plt.xlabel('Popularity')

plt.ylabel('Count')


# Show the bar plot

plt.show()
```



This code generates a bar plot of the counts of each unique value in the 'popularity' column of the DataFrame data2 using matplotlib.

- Create Bar Plot: The data2['popularity'].value_counts().plot.bar(color='skyblue', edgecolor='black', figsize=(20, 10)) line creates a bar plot of the counts of unique values in the 'popularity' column.
- data2['popularity'].value_counts() calculates the counts of each unique value in the 'popularity' column.
- .plot.bar() creates a bar plot from the counts.
- color='skyblue' sets the color of the bars to sky blue.
- edgecolor='black' sets the color of the edges of the bars to black.
- figsize=(20, 10) specifies the size of the figure (plot) as 20 inches wide and 10 inches tall.
- Add Title and Labels: The plt.title('Bar Plot of Popularity Counts'), plt.xlabel('Popularity'), and plt.ylabel('Count') lines add a title to the plot and label the x-axis and y-axis, respectively.
- Show the Plot: The plt.show() function displays the bar plot.

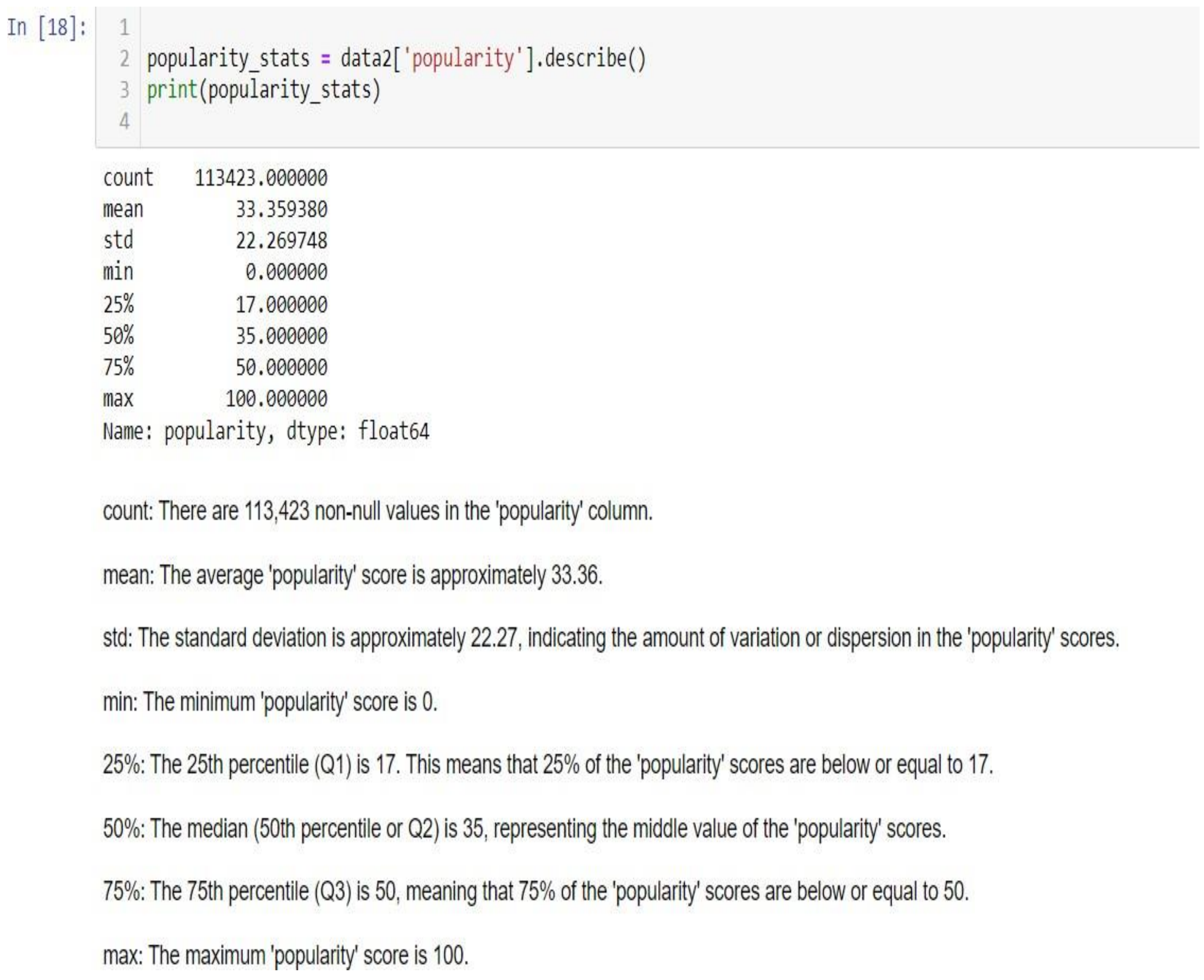
Overall, this code generates a bar plot that visualizes the counts of each unique popularity value in the dataset. The title and axis labels help provide context for interpreting the plot. The larger figure size ensures that the plot is easily readable.

➢ FOR STATISTICS:

```
popularity_stats = data2['popularity'].describe()

print(popularity_stats)
```

This code computes summary statistics for the 'popularity' column in the DataFrame data2 using the describe() method and assigns the result to a variable named popularity_stats.

Compute Summary Statistics: The data2['popularity'].describe() part of the code calculates summary statistics for the 'popularity' column.



```
In [18]:

popularity_stats = data2['popularity'].describe()
print(popularity_stats)
```

```
count    113423.000000
mean         33.359380
std          22.269748
min           0.000000
25%          17.000000
50%          35.000000
75%          50.000000
max         100.000000
Name: popularity, dtype: float64
```

count: There are 113,423 non-null values in the 'popularity' column.

mean: The average 'popularity' score is approximately 33.36.

std: The standard deviation is approximately 22.27, indicating the amount of variation or dispersion in the 'popularity' scores.

min: The minimum 'popularity' score is 0.

25%: The 25th percentile (Q1) is 17. This means that 25% of the 'popularity' scores are below or equal to 17.

50%: The median (50th percentile or Q2) is 35, representing the middle value of the 'popularity' scores.

75%: The 75th percentile (Q3) is 50, meaning that 75% of the 'popularity' scores are below or equal to 50.

max: The maximum 'popularity' score is 100.

The describe() method computes various descriptive statistics, including count, mean, standard deviation, minimum, 25th percentile (Q1), median (50th percentile or Q2), 75th percentile (Q3), and maximum.

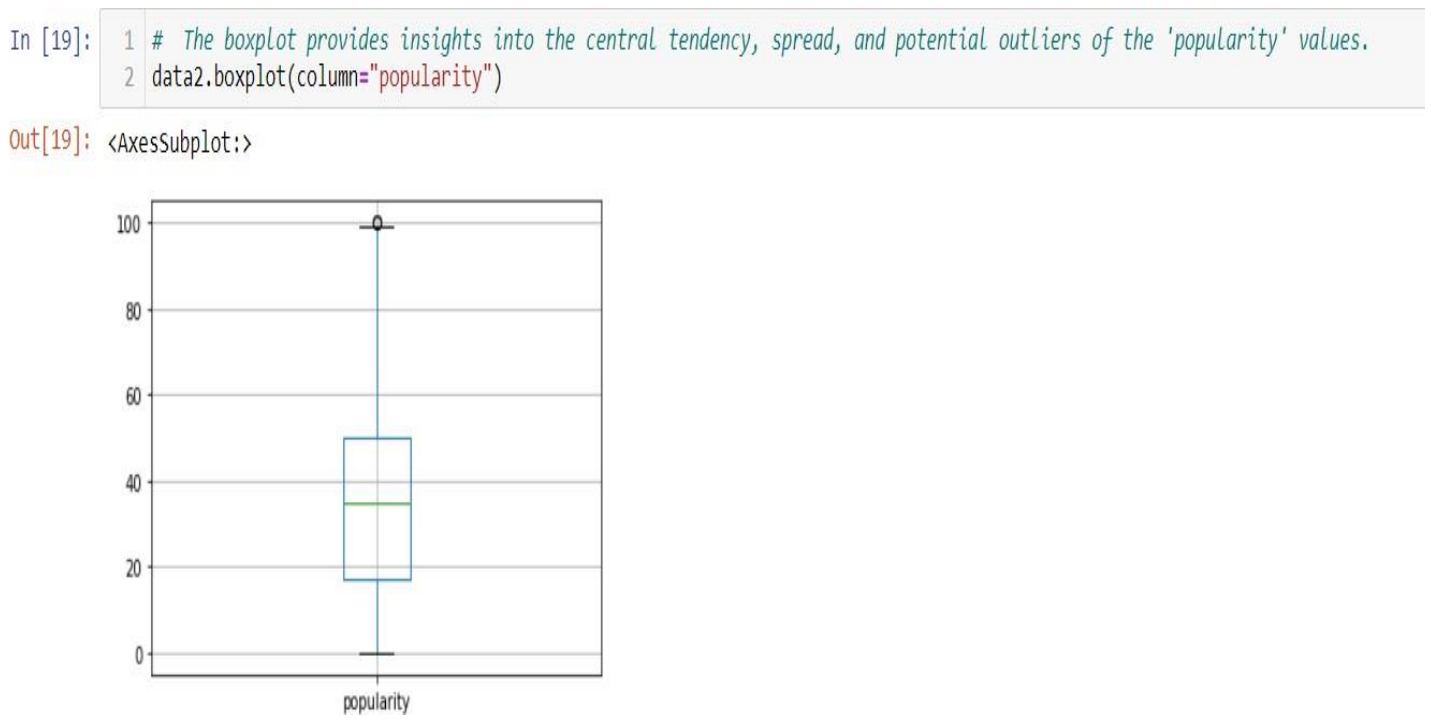
These statistics provide insights into the central tendency, dispersion, and shape of the distribution of 'popularity' values.

- Assign Summary Statistics to Variable: The result of the describe() method is assigned to a variable named popularity_stats. This variable now holds a pandas Series containing the summary statistics computed for the 'popularity' column.
- Print Summary Statistics: Finally, the print(popularity_stats) statement prints the summary statistics to the console.
- When you execute this code, it will print summary statistics such as count, mean, standard deviation, minimum, maximum, and quartile values for the 'popularity' column in the DataFrame data2. These statistics help you understand the distribution and characteristics of the 'popularity' values in your dataset.
- **count:** There are 113,423 non-null values in the 'popularity' column.
- **mean:** The average 'popularity' score is approximately 33.36.
- **std:** The standard deviation is approximately 22.27, indicating the amount of variation or dispersion in the 'popularity' scores.
- **min:** The minimum 'popularity' score is 0.
- **25%:** The 25th percentile (Q1) is 17. This means that 25% of the 'popularity' scores are below or equal to 17.
- **50%:** The median (50th percentile or Q2) is 35, representing the middle value of the 'popularity' scores.
- **75%:** The 75th percentile (Q3) is 50, meaning that 75% of the 'popularity' scores are below or equal to 50.
- **max:** The maximum 'popularity' score is 100.

➢ OUTLIER DETECTION:

The boxplot provides insights into the central tendency, spread, and potential outliers of the 'popularity' values.

data2.boxplot(column="popularity")



```
In [19]:  # The boxplot provides insights into the central tendency, spread, and potential outliers of the 'popularity' values.
          data2.boxplot(column="popularity")

Out[19]: <AxesSubplot:>
```

The provided code generates a boxplot for the 'popularity' values in the DataFrame data2 using matplotlib.

- **Create Boxplot:** The data2.boxplot(column="popularity") line creates a boxplot for the 'popularity' column. column="popularity" specifies the column to be used for creating the boxplot, which is 'popularity' in this case.
- **Boxplot Interpretation:**The box in the plot represents the interquartile range (IQR), with the central line representing the median (50th percentile or Q2).

- The "whiskers" extend from the box to the minimum and maximum values within 1.5 times the IQR from the first and third quartiles, respectively.
- Points beyond the whiskers are considered potential outliers and are plotted individually as points.
- **Central Tendency:** The median line gives insight into the central tendency of the 'popularity' values.
- **Spread:** The length of the box and the spread of the whiskers indicate the spread or variability of the 'popularity' values.
- **Outliers:** Individual points beyond the whiskers may represent potential outliers in the dataset.
- **Display the Plot:** The boxplot is displayed automatically after calling the data2.boxplot(column="popularity") line.

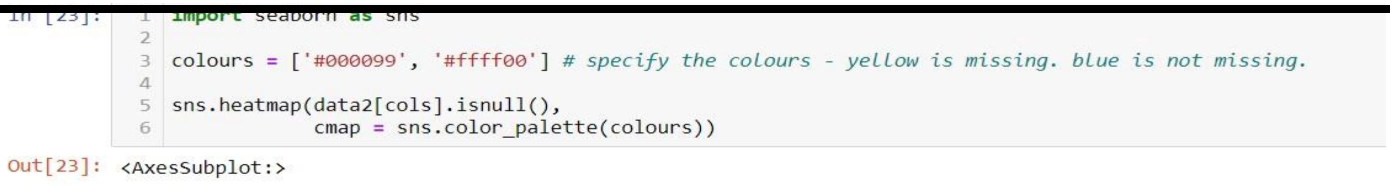
Boxplots are useful for visually summarizing the distribution of a dataset and identifying potential outliers. They provide a compact way to represent key statistical measures and visualize the spread of the data.

➢ **Checking null values:**

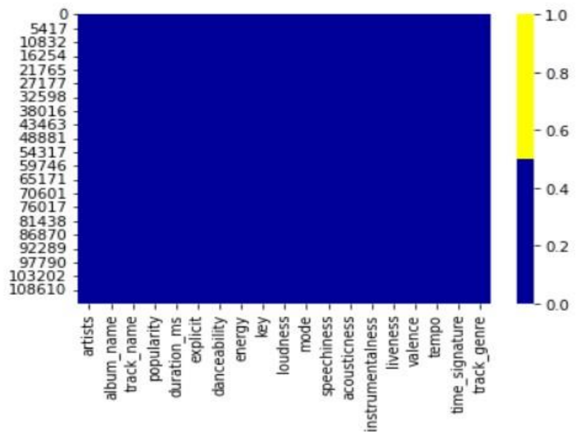
```
# used to count the number of missing (NaN) values in each column

data2.isna().sum()
```

- **data2.isna():** This part of the code returns a DataFrame of boolean values indicating whether each element in data2 is NaN (missing) or not. For each element in data2, it returns True if the value is NaN and False otherwise.
- **.sum():** This method is applied to the DataFrame generated by data2.isna() to calculate the sum along the axis (default axis is 0, which sums over rows).
- Since boolean values are treated as 1 for True and 0 for False when summing, the sum effectively counts the number of True values for each column.
- As a result, for each column, it returns the total count of NaN values.
- So, when you execute data2.isna().sum(), it returns a Series where the index represents column names, and the values represent the count of missing values (NaN) in each column of the DataFrame data2. This is a convenient way to quickly identify which columns contain missing data and how many missing values are present in each column.

```
import seaborn as sns


colours = ['#000099', '#ffff00']

        # specify the colours - yellow is missing. blue is not missing.


sns.heatmap(data2[cols].isnull(),

        cmap = sns.color_palette(colours))
```



```
In [23]: import seaborn as sns

         colours = ['#000099', '#ffff00'] # specify the colours - yellow is missing. blue is not missing.

         sns.heatmap(data2[cols].isnull(),
                     cmap = sns.color_palette(colours))
Out[23]: <AxesSubplot:>
```



This code generates a heatmap using Seaborn to visualize missing (NaN) values in the selected columns of the DataFrame data2.

- **Import Seaborn:** The code imports the Seaborn library with the alias sns. Seaborn is a data visualization library based on Matplotlib that provides a high-level interface for drawing attractive and informative statistical graphics.
- **Define Colors:** The colours list contains two color codes: '#000099' and '#ffff00'. These colors represent blue and yellow, respectively. In this context, blue indicates non-missing values, and yellow indicates missing values.
- **Create Heatmap:** The sns.heatmap() function is called to create the heatmap. data2[cols].isnull() generates a DataFrame with boolean values, where True represents missing (NaN) values and False represents non-missing values. cols is assumed to be a list of column names.
- cmap = sns.color_palette(colours) specifies the color palette to be used for the heatmap. The colors specified in the colours list are passed to sns.color_palette() to create a color palette.
- **Display Heatmap:** The generated heatmap is displayed automatically.

The heatmap provides a visual representation of missing values in the selected columns of the DataFrame. Blue cells represent non-missing values, while yellow cells represent missing values. This visualization helps to identify patterns of missingness across different columns and can be useful for data cleaning and exploration.

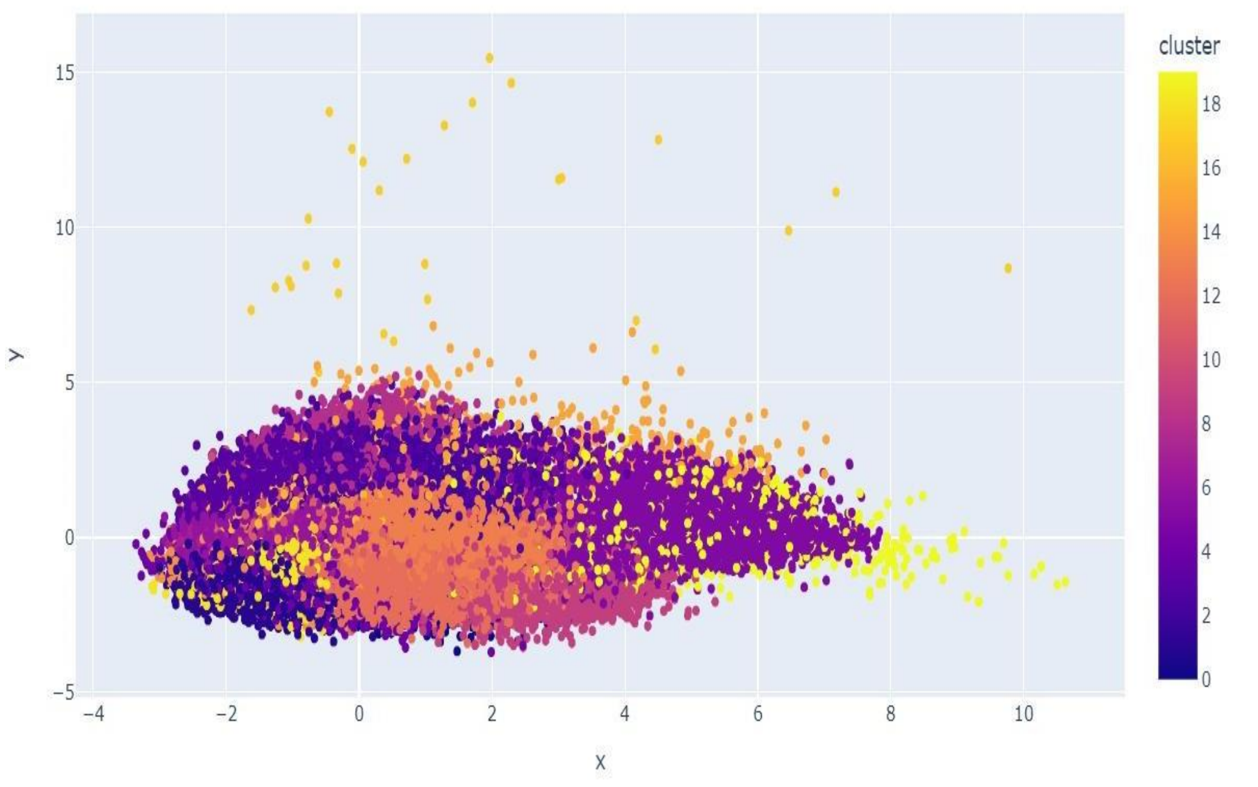
MODEL BUILDING

➢ K- means clustering algorithm:

- K-Means Clustering is an unsupervised learning algorithm that is used to solve the clustering problems in machine learning
- The goal of clustering is to divide the population or set of data points into a number of groups so that the data points within each group are more comparable to one another and different from the data points within the other groups. It is essentially a grouping of things based on how similar and different they are to one another.
- K-means clustering is a popular unsupervised machine learning algorithm used for partitioning a dataset into K distinct, non-overlapping clusters. The goal of K-means is to find clusters in such a way that the within-cluster variation (sum of squared distances from each point to the centroid of its assigned cluster) is minimized

K means algorithm typically works:

- **Initialization:** Choose K initial centroids randomly from the data points. These centroids will act as the center of the clusters.
- **Assignment:** Assign each data point to the nearest centroid, creating K clusters. The distance metric commonly used is Euclidean distance.
- **Update centroids:** Compute the new centroid for each cluster by taking the mean of all data points assigned to that cluster.
- **Repeat:** Iterate steps 2 and 3 until convergence. Convergence occurs when the centroids no longer change significantly or after a fixed number of iterations.
- **Termination:** Once the centroids stabilize or the maximum number of iterations is reached, the algorithm terminates, and the final clusters are obtained.



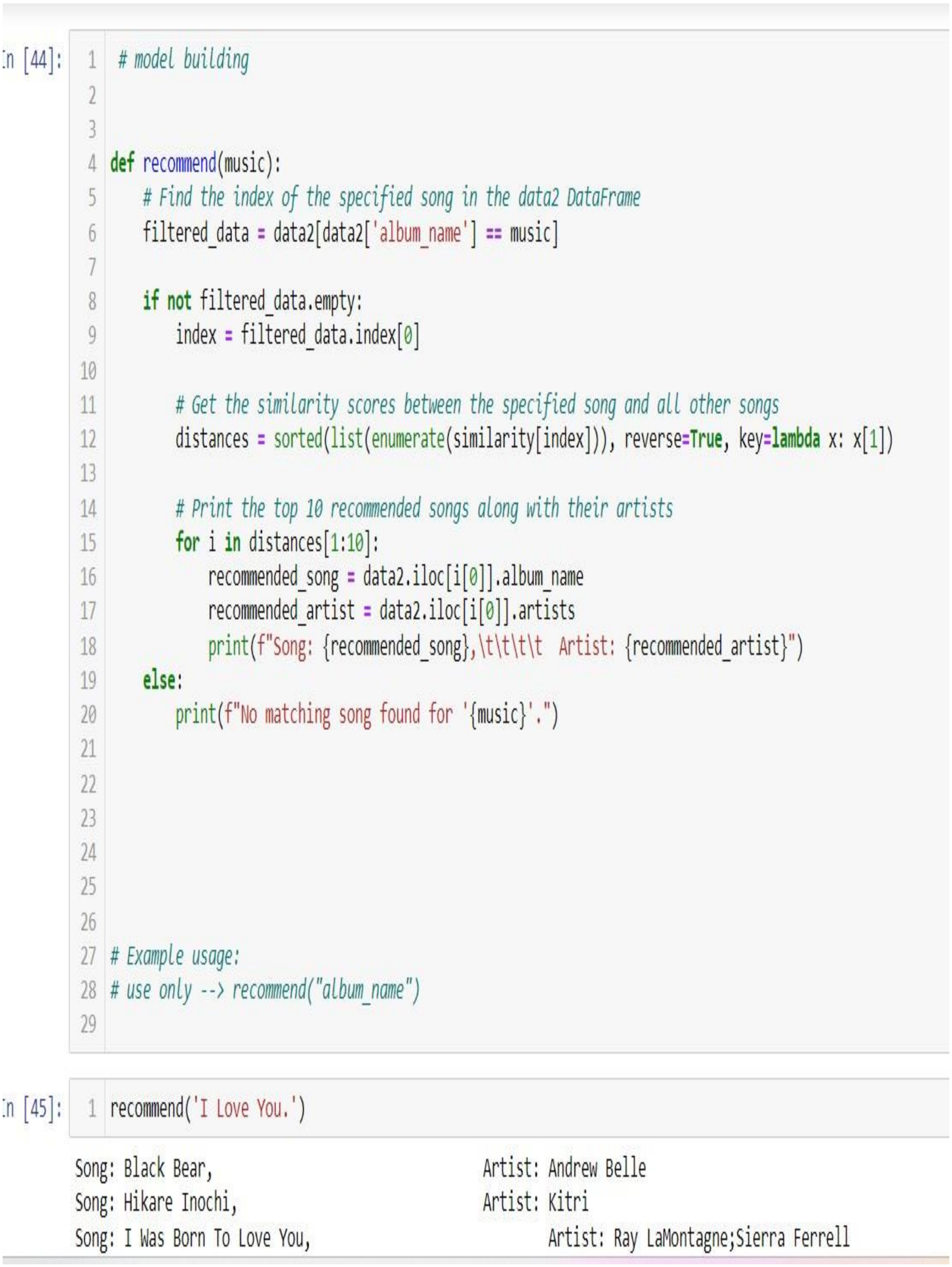
- It's worth noting that K-means clustering can be sensitive to the initial selection of centroids and may converge to a local minimum, which might not be the optimal solution. Therefore, it's common to run the algorithm multiple times with different initializations and choose the solution with the lowest within-cluster variation.
- K-means clustering is widely used in various fields, including image segmentation, customer segmentation, anomaly detection, and more. However, it has some limitations, such as its sensitivity to outliers and the requirement to specify the number of clusters (K) in advance, which can sometimes be challenging in practice.

In conclusion, K-means clustering is a powerful unsupervised machine learning algorithm for grouping unlabeled datasets. Its objective is to divide data into clusters, making similar data points part of the same group. The algorithm initializes cluster centroids and iteratively assigns data points to the nearest centroid, updating centroids based on the mean of points in each cluste

➢ CODE FOR MODEL BUILDING:

```
def recommend(music):
    # Find the index of the specified song in the data2 DataFrame
    filtered_data = data2[data2['album_name'] == music]

    if not filtered_data.empty:
        index = filtered_data.index[0]




# Get the similarity scores between the specified song and all other songs
        distances = sorted(list(enumerate(similarity[index])), reverse=True,
key=lambda x: x[1])

        # Print the top 10 recommended songs along with their artists
        for i in distances[1:10]:
            recommended_song = data2.iloc[i[0]].album_name
            recommended_artist = data2.iloc[i[0]].artists
            print(f"Song: {recommended_song},\t\t\t\t  Artist: {recommended_artist}")
    else:
        print(f"No matching song found for '{music}'.")
```

This function recommend(music) takes one argument, music, which is the name of an album. It aims to recommend similar songs based on the provided album's name.



```
In [44]:
# model building


def recommend(music):
    # Find the index of the specified song in the data2 DataFrame
    filtered_data = data2[data2['album_name'] == music]

    if not filtered_data.empty:
        index = filtered_data.index[0]

        # Get the similarity scores between the specified song and all other songs
        distances = sorted(list(enumerate(similarity[index])), reverse=True, key=lambda x: x[1])

        # Print the top 10 recommended songs along with their artists
        for i in distances[1:10]:
            recommended_song = data2.iloc[i[0]].album_name
            recommended_artist = data2.iloc[i[0]].artists
            print(f"Song: {recommended_song},\t\t\t\t  Artist: {recommended_artist}")
    else:
        print(f"No matching song found for '{music}'.")







# Example usage:
# use only --> recommend("album_name")

```

```
In [45]:
recommend('I Love You.')
```

```
Song: Black Bear,                              Artist: Andrew Belle
Song: Hikare Inochi,                           Artist: Kitri
Song: I Was Born To Love You,                        Artist: Ray LaMontagne;Sierra Ferrell
```

Here's a detailed explanation:

- **Finding the Index of the Specified Song:**

The function filters the DataFrame data2 based on the 'album_name' column to find rows where the album name matches the provided input music.

If there are matches, it retrieves the index of the first matching song. If there are no matches, it prints a message indicating that no matching song was found.

- **Calculating Similarity Scores:**

It uses the calculated similarity matrix named similarity. Presumably, this matrix contains pairwise similarity scores between songs.

It retrieves the similarity scores between the specified song and all other songs. It sorts these scores in descending order, preserving the index information.

- **Printing Recommendations:**

It loops through the top 10 similarity scores (excluding the first one, which would be the song itself).

For each of these similar songs, it retrieves the album name and artist from the DataFrame using the index information obtained earlier.

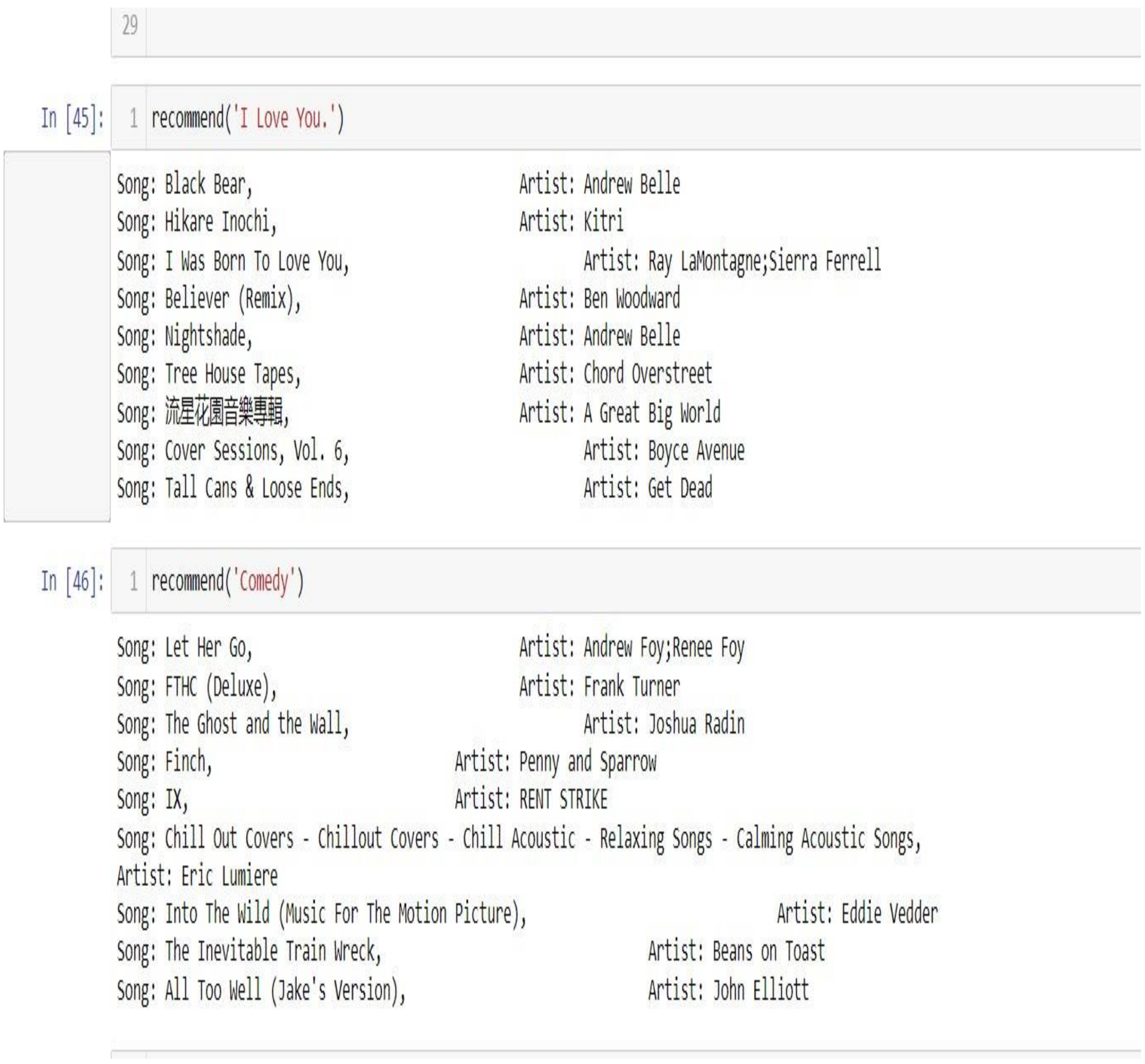
It prints out the recommended song's name and its artist.

- **Handling Cases with No Matches:**

If there are no matches for the provided album name in the DataFrame, it prints a message indicating that no matching song was found.

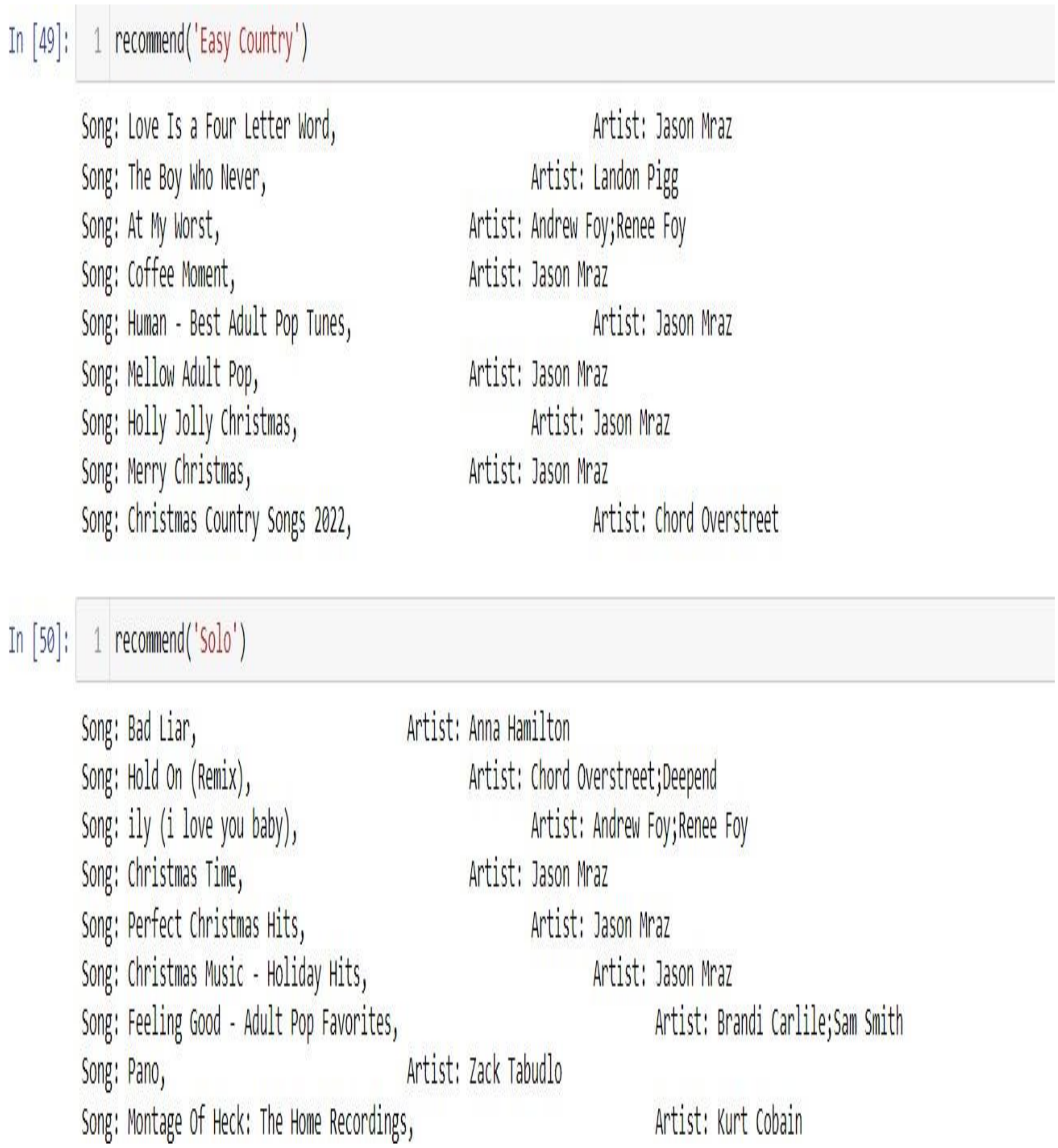
This function assumes the existence of a similarity matrix named similarity and a DataFrame named data2, where information about songs (such as album names and artists) is stored. Additionally, it's important to ensure that the similarity matrix and DataFrame are correctly populated and aligned for accurate recommendations

RECOMMENDED SONGS



```
In [45]:  recommend('I Love You.')

Song: Black Bear,                        Artist: Andrew Belle
Song: Hikare Inochi,                     Artist: Kitri
Song: I Was Born To Love You,                      Artist: Ray LaMontagne;Sierra Ferrell
Song: Believer (Remix),                  Artist: Ben Woodward
Song: Nightshade,                        Artist: Andrew Belle
Song: Tree House Tapes,                  Artist: Chord Overstreet
Song: 流星花園音樂專輯,                   Artist: A Great Big World
Song: Cover Sessions, Vol. 6,                      Artist: Boyce Avenue
Song: Tall Cans & Loose Ends,                      Artist: Get Dead
```

```
In [46]:  recommend('Comedy')

Song: Let Her Go,                        Artist: Andrew Foy;Renee Foy
Song: FTHC (Deluxe),                     Artist: Frank Turner
Song: The Ghost and the Wall,                      Artist: Joshua Radin
Song: Finch,                    Artist: Penny and Sparrow
Song: IX,                       Artist: RENT STRIKE
Song: Chill Out Covers - Chillout Covers - Chill Acoustic - Relaxing Songs - Calming Acoustic Songs,
Artist: Eric Lumiere
Song: Into The Wild (Music For The Motion Picture),                        Artist: Eddie Vedder
Song: The Inevitable Train Wreck,                  Artist: Beans on Toast
Song: All Too Well (Jake's Version),               Artist: John Elliott
```

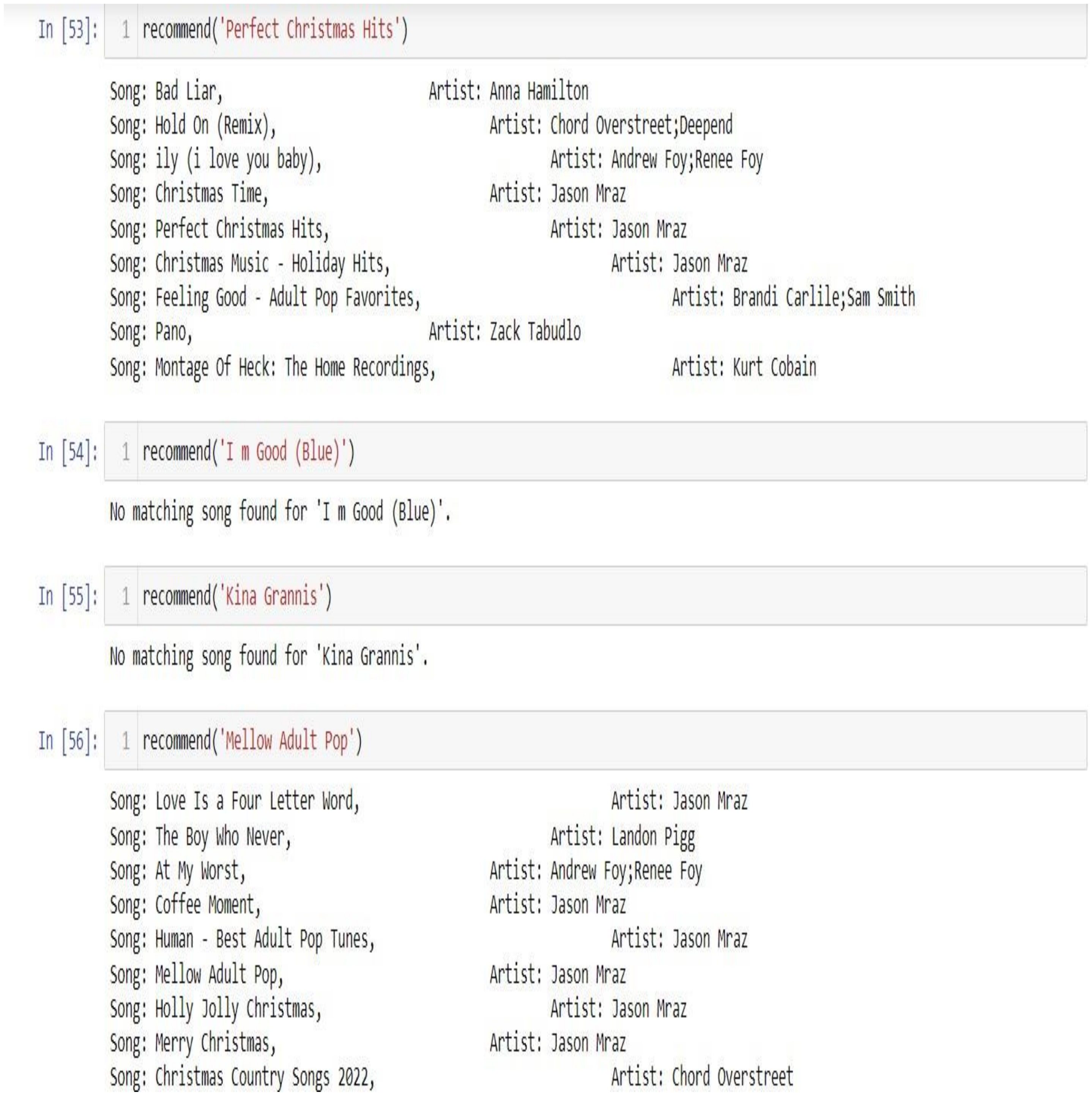


```
In [49]: recommend('Easy Country')
```

```
Song: Love Is a Four Letter Word,                                 Artist: Jason Mraz
Song: The Boy Who Never,                                  Artist: Landon Pigg
Song: At My Worst,                                Artist: Andrew Foy;Renee Foy
Song: Coffee Moment,                              Artist: Jason Mraz
Song: Human - Best Adult Pop Tunes,                               Artist: Jason Mraz
Song: Mellow Adult Pop,                           Artist: Jason Mraz
Song: Holly Jolly Christmas,                              Artist: Jason Mraz
Song: Merry Christmas,                            Artist: Jason Mraz
Song: Christmas Country Songs 2022,                               Artist: Chord Overstreet
```

```
In [50]: recommend('Solo')
```

```
Song: Bad Liar,                           Artist: Anna Hamilton
Song: Hold On (Remix),                            Artist: Chord Overstreet;Deepend
Song: ily (i love you baby),                              Artist: Andrew Foy;Renee Foy
Song: Christmas Time,                             Artist: Jason Mraz
Song: Perfect Christmas Hits,                             Artist: Jason Mraz
Song: Christmas Music - Holiday Hits,                             Artist: Jason Mraz
Song: Feeling Good - Adult Pop Favorites,                                 Artist: Brandi Carlile;Sam Smith
Song: Pano,                               Artist: Zack Tabudlo
Song: Montage Of Heck: The Home Recordings,                               Artist: Kurt Cobain
```



```
In [53]: recommend('Perfect Christmas Hits')
```

```
Song: Bad Liar,                           Artist: Anna Hamilton
Song: Hold On (Remix),                            Artist: Chord Overstreet;Deepend
Song: ily (i love you baby),                              Artist: Andrew Foy;Renee Foy
Song: Christmas Time,                             Artist: Jason Mraz
Song: Perfect Christmas Hits,                             Artist: Jason Mraz
Song: Christmas Music - Holiday Hits,                             Artist: Jason Mraz
Song: Feeling Good - Adult Pop Favorites,                                 Artist: Brandi Carlile;Sam Smith
Song: Pano,                               Artist: Zack Tabudlo
Song: Montage Of Heck: The Home Recordings,                               Artist: Kurt Cobain
```

```
In [54]: recommend('I m Good (Blue)')
```

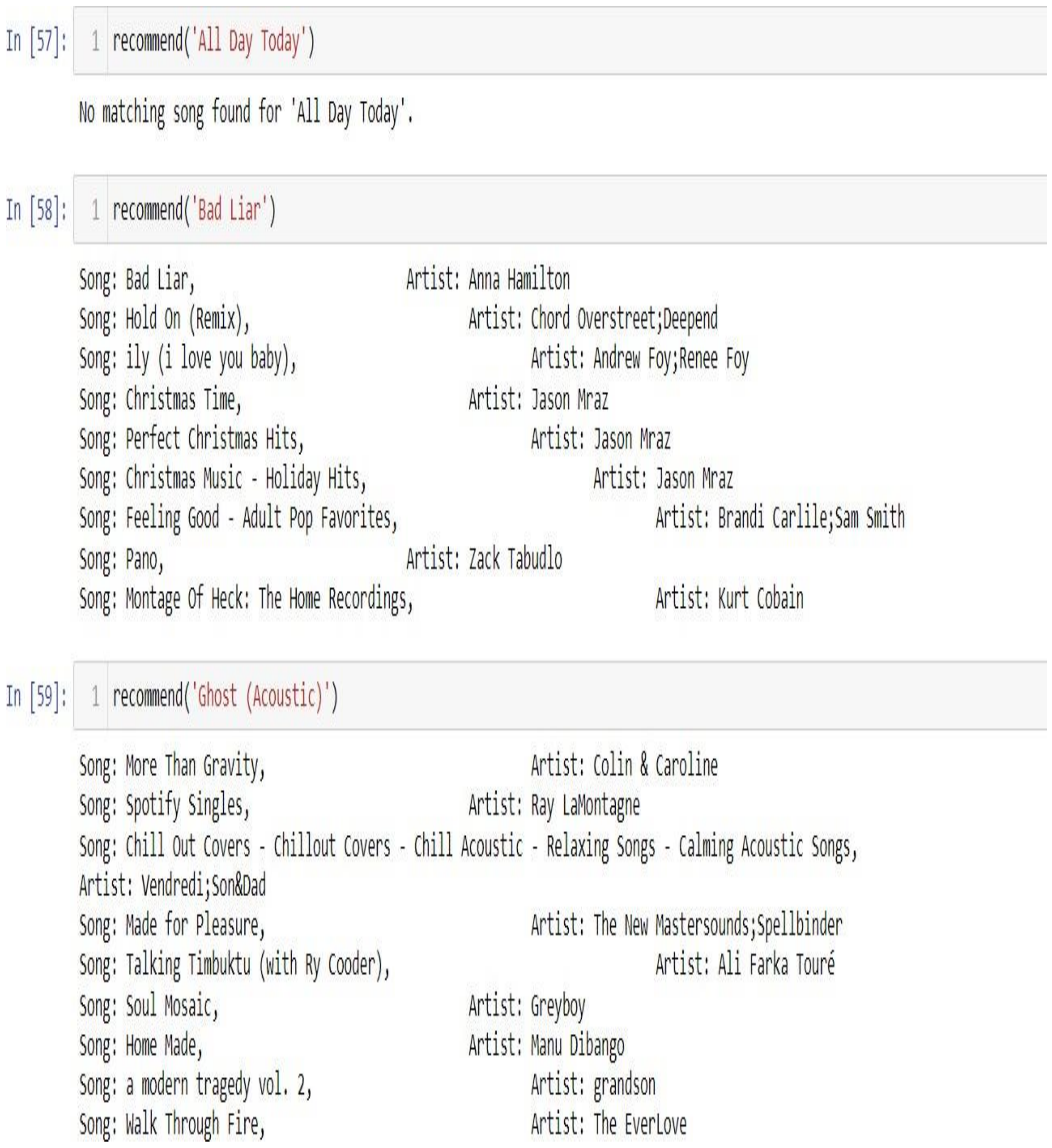
```
No matching song found for 'I m Good (Blue)'.
```

```
In [55]: recommend('Kina Grannis')
```

```
No matching song found for 'Kina Grannis'.
```

```
In [56]: recommend('Mellow Adult Pop')
```

```
Song: Love Is a Four Letter Word,                                 Artist: Jason Mraz
Song: The Boy Who Never,                                  Artist: Landon Pigg
Song: At My Worst,                                Artist: Andrew Foy;Renee Foy
Song: Coffee Moment,                              Artist: Jason Mraz
Song: Human - Best Adult Pop Tunes,                               Artist: Jason Mraz
Song: Mellow Adult Pop,                           Artist: Jason Mraz
Song: Holly Jolly Christmas,                              Artist: Jason Mraz
Song: Merry Christmas,                            Artist: Jason Mraz
Song: Christmas Country Songs 2022,                               Artist: Chord Overstreet
```



In [57]:
```
recommend('All Day Today')
```

```
No matching song found for 'All Day Today'.
```

In [58]:
```
recommend('Bad Liar')
```

```
Song: Bad Liar,                           Artist: Anna Hamilton
Song: Hold On (Remix),                            Artist: Chord Overstreet;Deepend
Song: ily (i love you baby),                              Artist: Andrew Foy;Renee Foy
Song: Christmas Time,                             Artist: Jason Mraz
Song: Perfect Christmas Hits,                             Artist: Jason Mraz
Song: Christmas Music - Holiday Hits,                             Artist: Jason Mraz
Song: Feeling Good - Adult Pop Favorites,                                 Artist: Brandi Carlile;Sam Smith
Song: Pano,                               Artist: Zack Tabudlo
Song: Montage Of Heck: The Home Recordings,                               Artist: Kurt Cobain
```

In [59]:
```
recommend('Ghost (Acoustic)')
```

```
Song: More Than Gravity,                                  Artist: Colin & Caroline
Song: Spotify Singles,                            Artist: Ray LaMontagne
Song: Chill Out Covers - Chillout Covers - Chill Acoustic - Relaxing Songs - Calming Acoustic Songs,
Artist: Vendredi;Son&Dad
Song: Made for Pleasure,                                  Artist: The New Mastersounds;Spellbinder
Song: Talking Timbuktu (with Ry Cooder),                                  Artist: Ali Farka Touré
Song: Soul Mosaic,                                Artist: Greyboy
Song: Home Made,                                  Artist: Manu Dibango
Song: a modern tragedy vol. 2,                            Artist: grandson
Song: Walk Through Fire,                                  Artist: The EverLove
```

CONCLUSION:

In conclusion, the development and implementation of a music recommendation system offer significant benefits to both users and music platforms alike. By content based recommendation system with k means algorithms and user data analysis, these systems can effectively personalize music suggestions, enhancing user satisfaction and engagement. Furthermore, they contribute to the discovery of new artists and genres, fostering diversity and innovation within the music industry. However, it is essential to address concerns regarding privacy, bias, and the potential homogenization of musical tastes. Overall, the continued refinement and responsible deployment of music recommendation systems hold promise for enriching the music listening experience while navigating the complexities of modern digital platforms.

REFERENCE:

1.Let Us Python ebook by Yashavant Kanetkar - Rakuten Kobo

2. fundamental of statistics by goon gupta , das gupta

3.https://www.kaggle.com/code/prashant111/recommender-systems-in-python

4. https://www.javatpoint.com/product-recommendation-machine-learning

5. Python Programming for the Absolute Beginner by Michael Dawson-Cengage learning.