**Spotify Data Analysis and Prediction Using Python**

**1. Introduction**

**1.1 Project Overview**

This project aims to analyze a Spotify music dataset to uncover trends, correlations, and patterns among song features, genres, and popularity. Additionally, we explore predictive modeling to estimate a song's popularity based on its musical and technical attributes.

**1.2 Objectives**

* Perform Exploratory Data Analysis (EDA) to understand the structure and patterns in the Spotify dataset.
* Visualize relationships among song attributes like loudness, energy, danceability, and popularity.
* Cluster songs using K-means to categorize them based on their audio features.
* Develop a predictive model to forecast song popularity based on audio characteristics.

**2. Methodology**

**2.1 Data Collection and Sources**

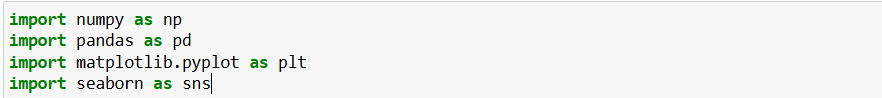
The dataset used in this analysis is derived from Spotify's API, containing features for songs, including:

* Song title, artist, release date, and genre.
* Quantitative audio features: danceability, energy, loudness, acousticness, instrumentalness, and popularity (a measure of a song’s success).

**2.2 Tools and Libraries**

The analysis is conducted using Python with the following libraries:

* **Pandas** for data manipulation and cleaning.
* **NumPy** for numerical operations.
* **Seaborn** and **Matplotlib** for data visualization.
* **Scikit-learn** for machine learning algorithms and data preprocessing.



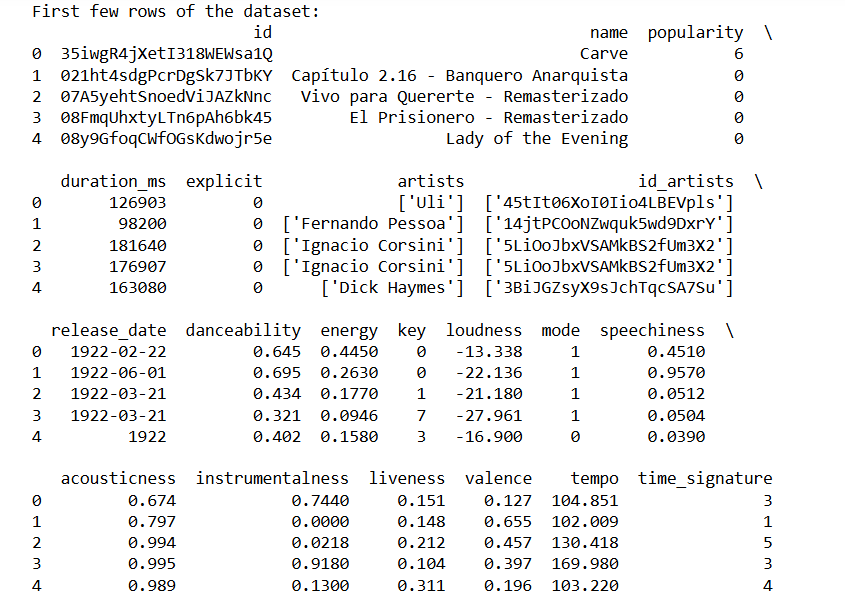
**3. Data Exploration and Cleaning**

* 1. **Data Inspection**

After loading the dataset, we examined its structure, checked for null values, and calculated basic statistics to get an overview of the data's completeness and scale.

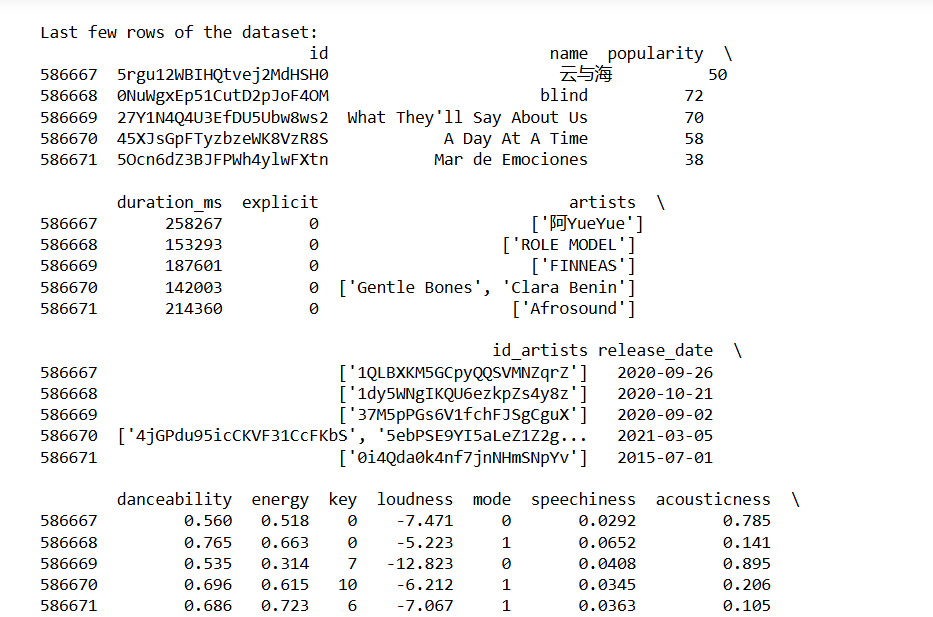
1.**Display the First Few Rows:** Use `head()` to get an initial look at the dataset.

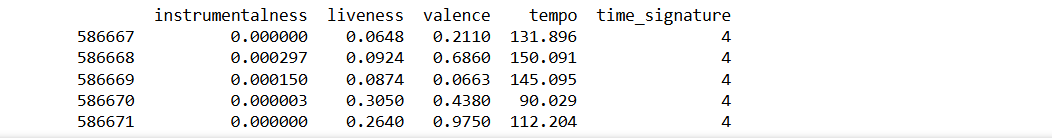
****

****

2. **Display the Last Few Rows**: Use `tail()` to view the end of the dataset.

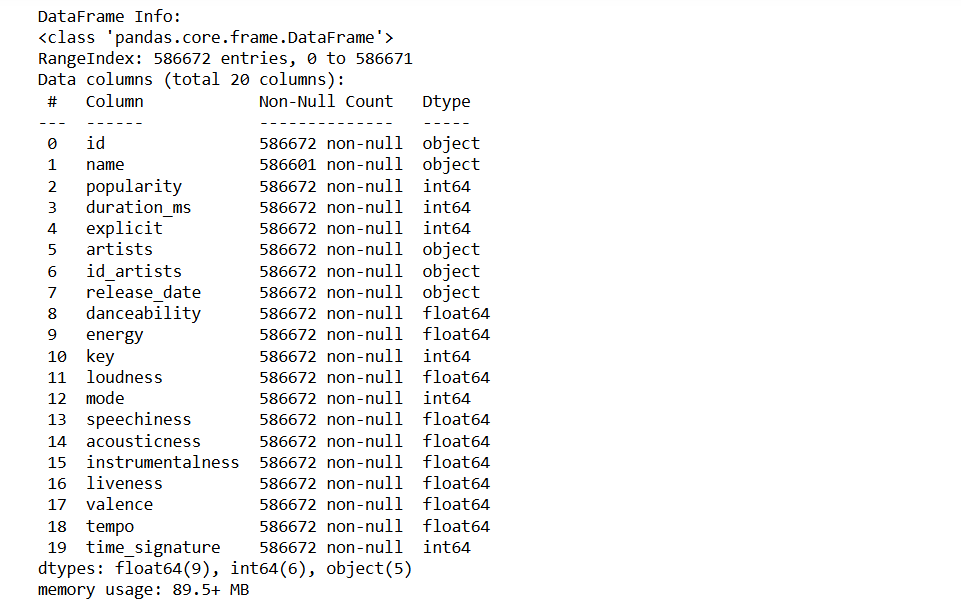
****

****

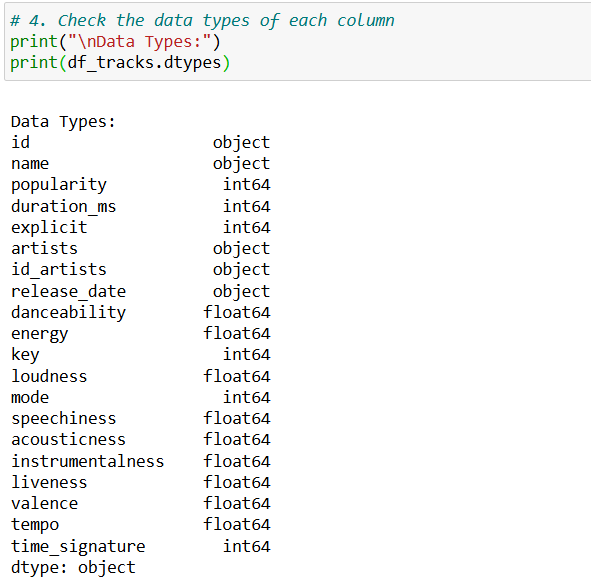
****

3. **DataFrame Info:** Use `info()` to display a summary of the DataFrame, including the number of non-null entries and data types.

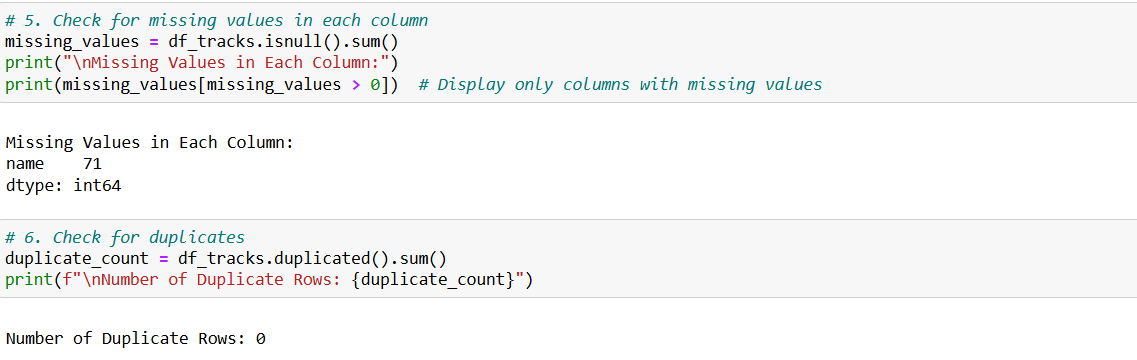
****

****

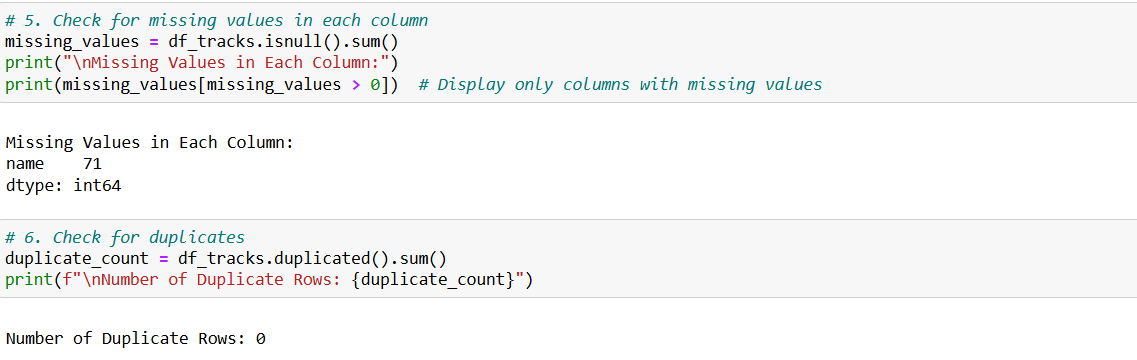
4. **Check Data Types**: Use `dtypes` to see the data types of each column.

****

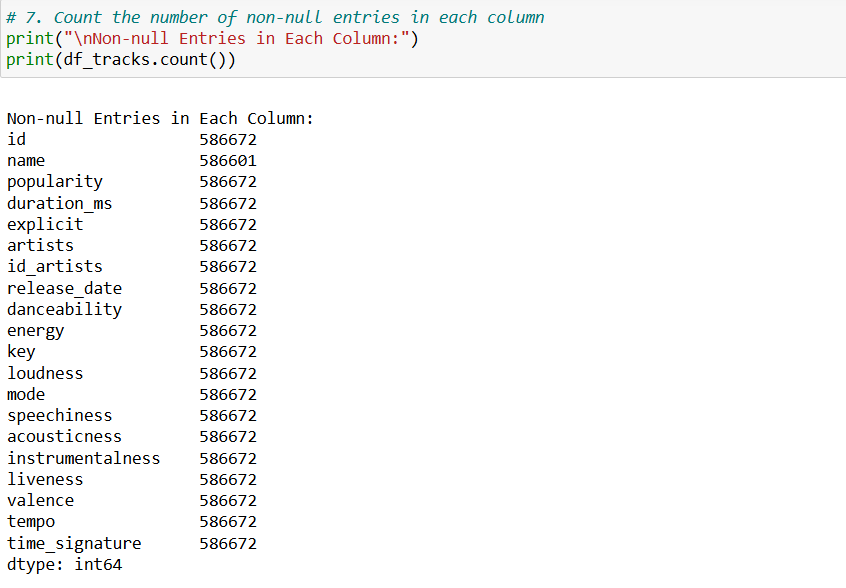
5. **Check for Missing Values**: Use `isnull().sum()` to identify any missing values in the dataset.

****

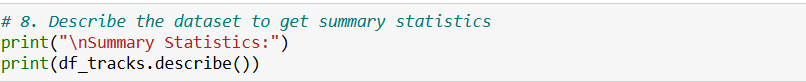
6**. Check for Duplicates** : Use `duplicated().sum()` to count duplicate rows in the DataFrame.

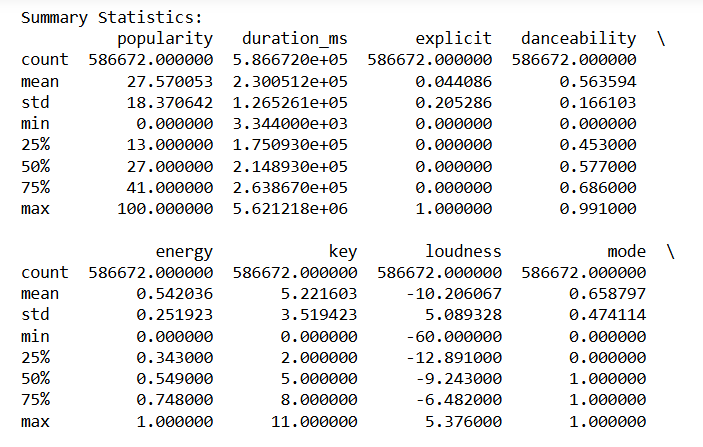
****

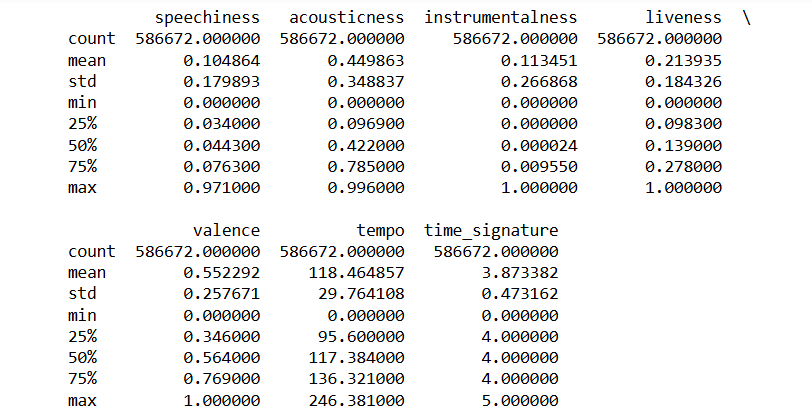
7. **Count Non-null Entries**: Use `count()` to get a count of non-null entries in each column.

****

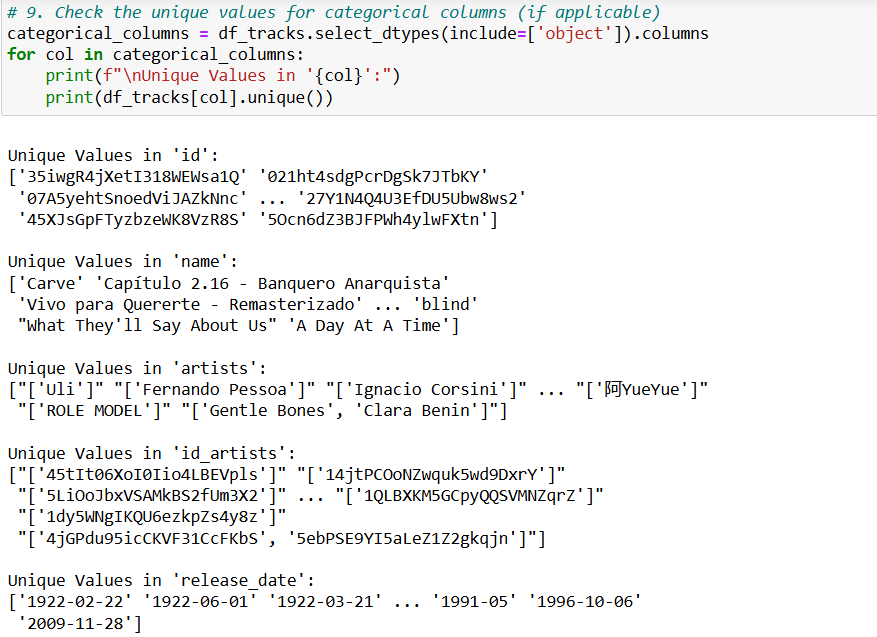
8. **Summary Statistics**: Use `describe()` to get summary statistics (count, mean, std, min, etc.) for numerical columns.

****

****

****

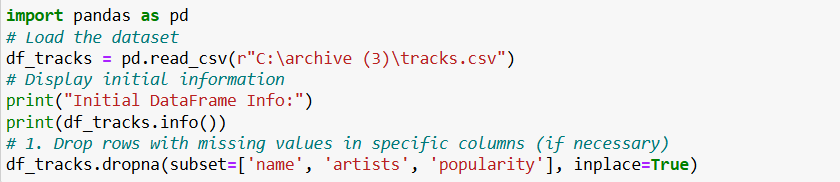
9. **Unique Values for Categorical Columns**: Loop through categorical columns and display unique values using `unique()`.

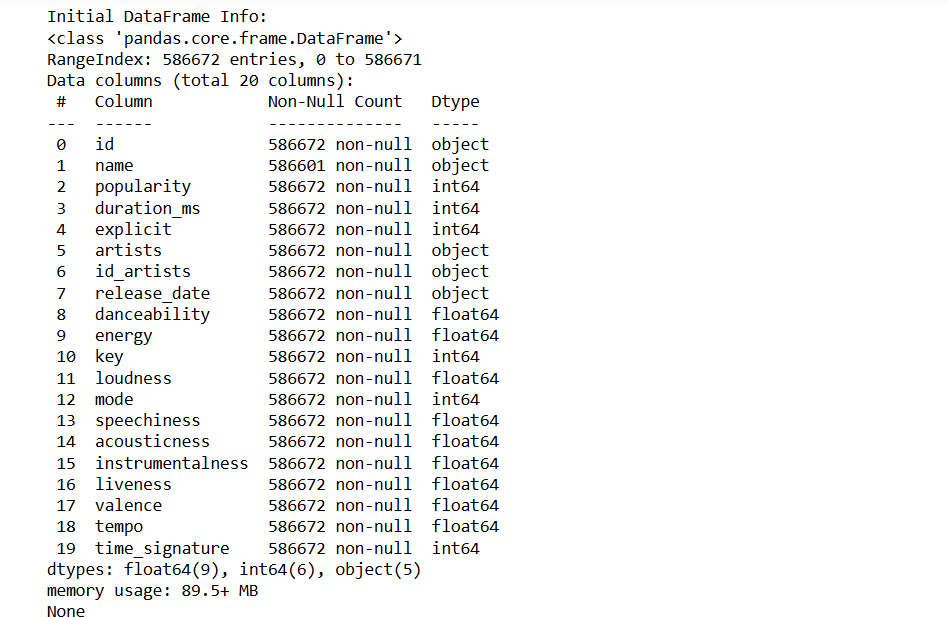
****

**3.2 Data Cleaning**

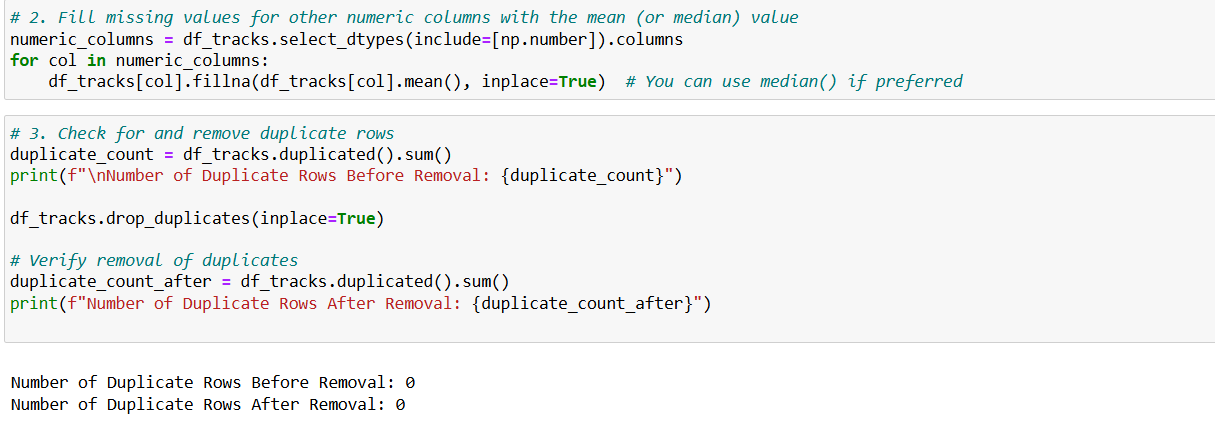
We handled missing values by dropping rows where crucial data, like song names, was missing. Additionally, the `release\_date` column was converted to datetime format to facilitate time-based analysis.

 **Drop Rows with Missing Values**: Use dropna() to remove rows with missing values in essential columns such as 'name', 'artists', and 'popularity'.

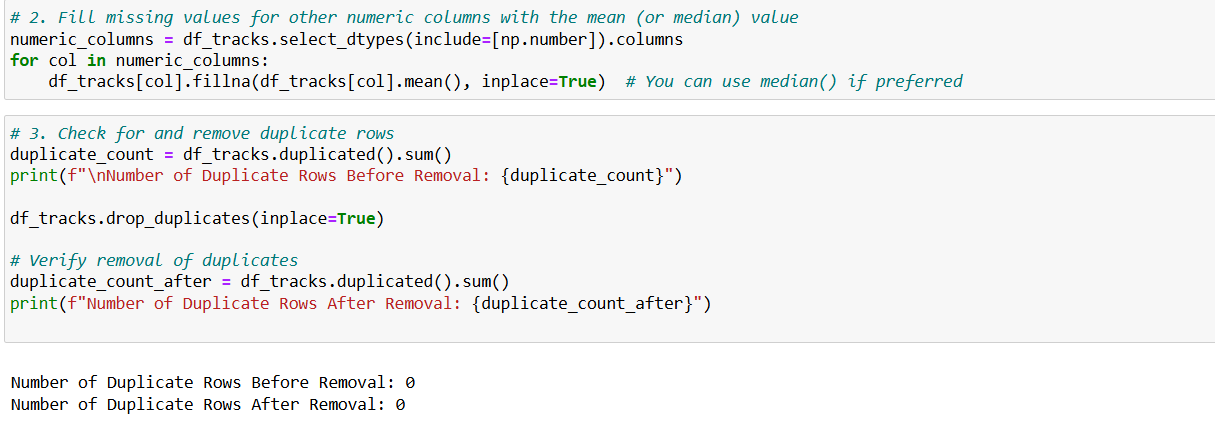
****

****

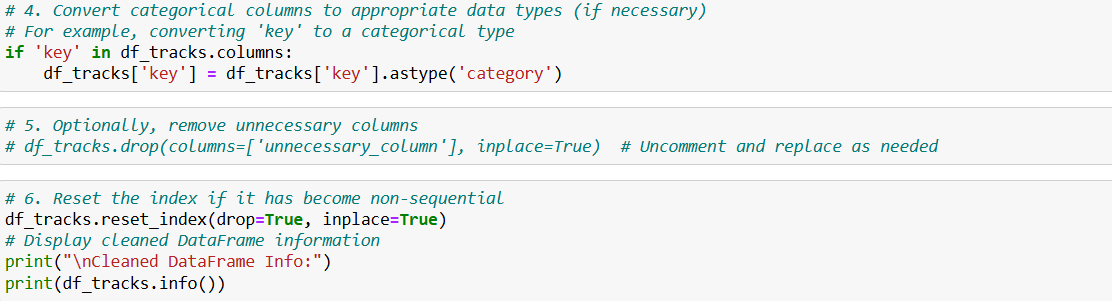
 **Fill Missing Values for Numeric Columns**: Use fillna() to replace missing values in numeric columns with the mean (or median) of the respective column.

****

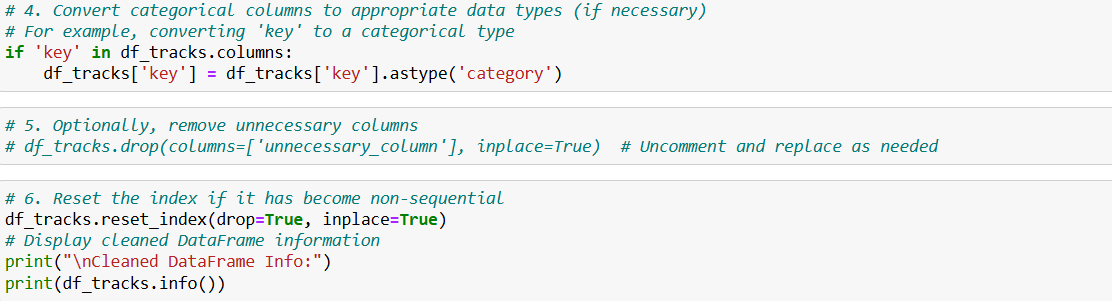
 **Check for Duplicates**: Use duplicated().sum() to count duplicates before removal and then use drop\_duplicates() to remove them. This ensures that each entry in the dataset is unique.

****

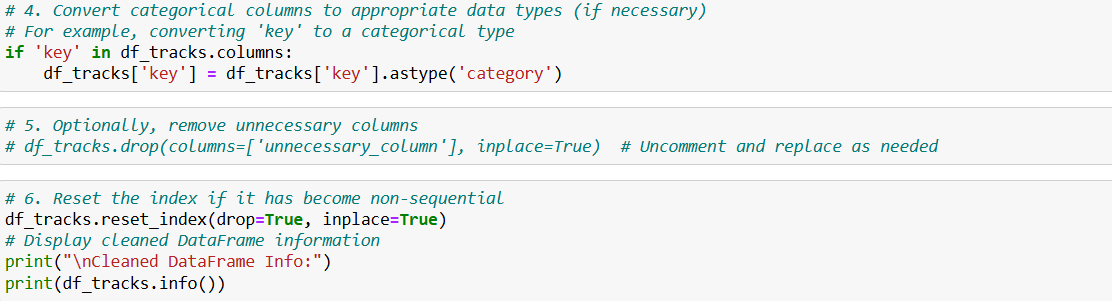
 **Convert Categorical Columns**: If there are categorical columns (e.g., 'key'), convert them to the appropriate data type (e.g., categorical) to optimize memory usage and improve performance in analysis.

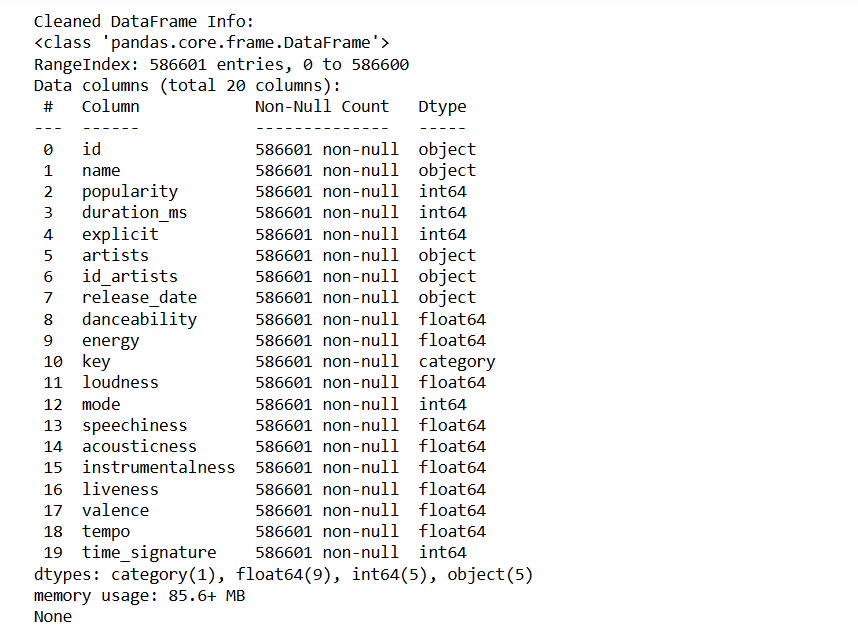
****

 **Remove Unnecessary Columns**: Optionally, if there are columns that you don't need for analysis, you can drop them using drop(). Uncomment the line and specify the column name(s) to remove.

****

 **Reset the Index**: After removing rows, the index may not be sequential. Use reset\_index() to re-index the DataFrame, which can be helpful for clarity.

****

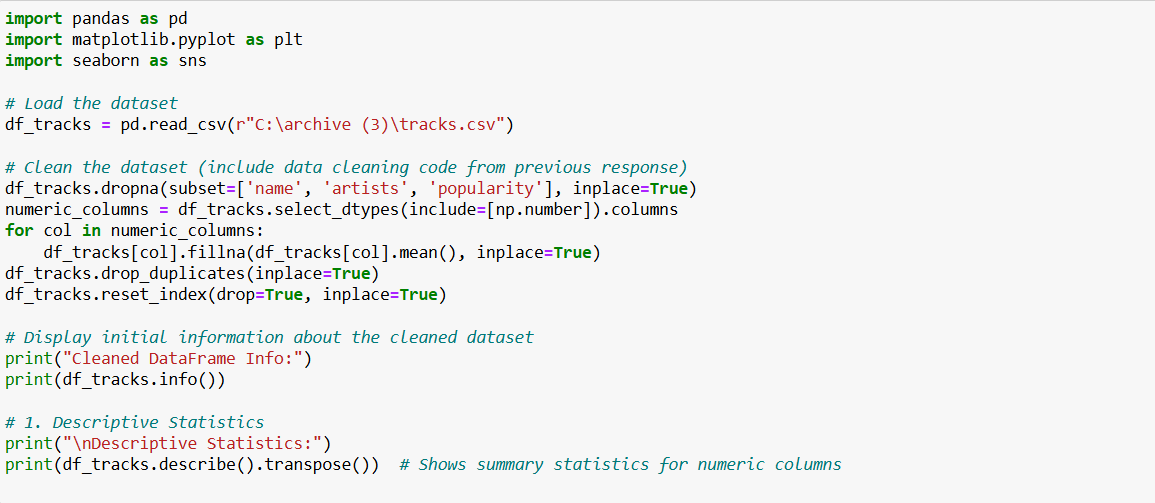
****

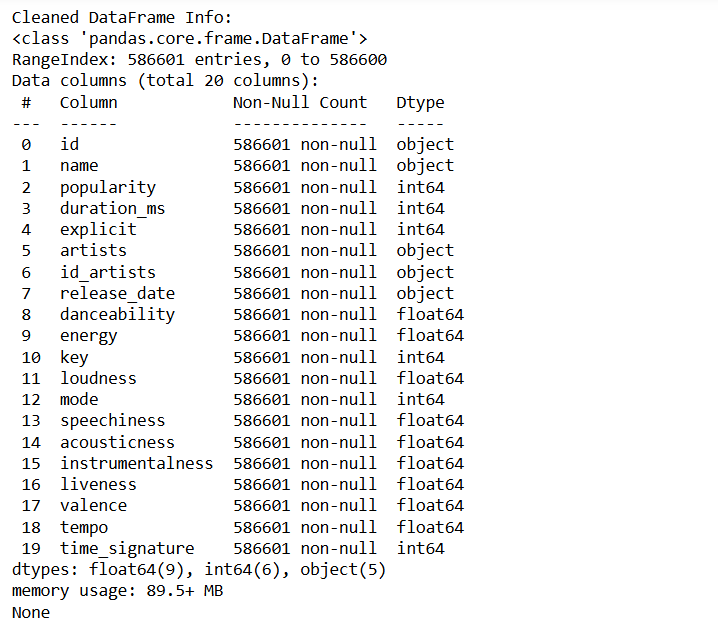
**4. Exploratory Data Analysis (EDA)**

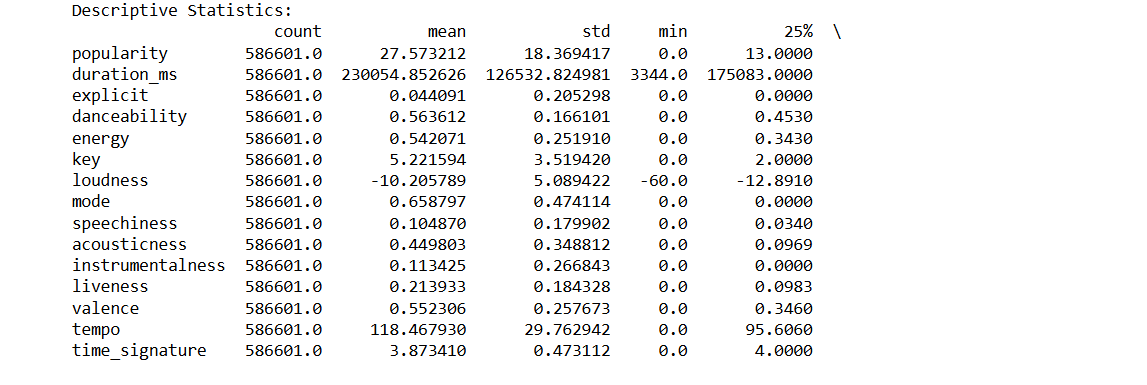
**4.1 Descriptive Statistics**

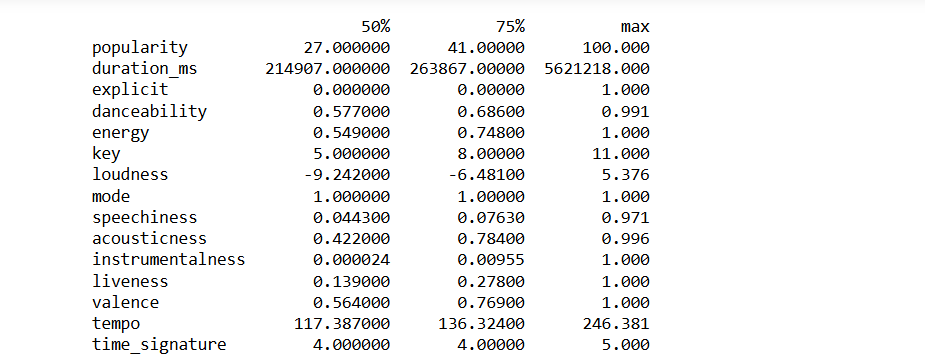
Basic statistical analysis revealed the range and mean values of numerical features, helping us understand the distribution and scale of each attribute.

 **Descriptive Statistics**:The describe() function provides summary statistics (count, mean, std, min, 25%, 50%, 75%, max) for numeric columns. Using transpose() makes it easier to read.



****

****

****

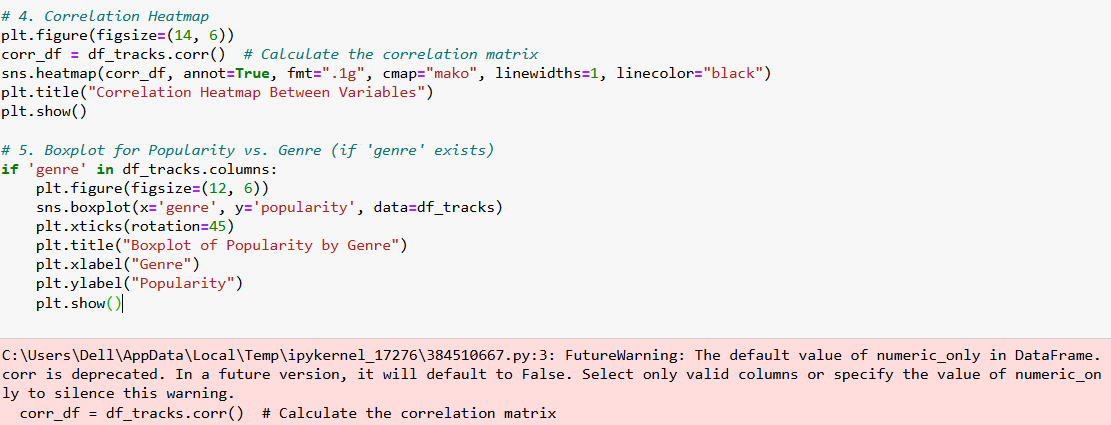
 **Count Unique Values in Categorical Columns**: Iterates through categorical columns to count unique values, providing insights into the diversity of categories present.

****

 **Visualizing Distribution of Popularity**: Uses sns.histplot() to plot the distribution of the 'popularity' column, including a kernel density estimate (KDE) to visualize the data's distribution.

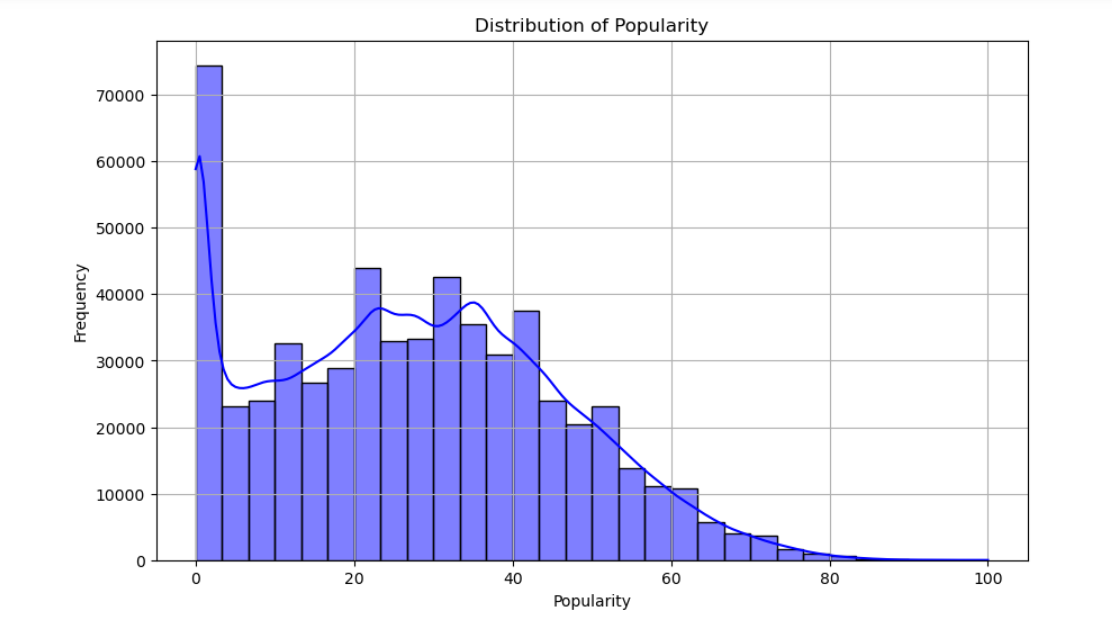
****

 **Correlation Heatmap**: Creates a heatmap using sns.heatmap() to visualize correlations between numeric variables. This helps identify potential relationships between features, aiding in understanding how they interact.

****

 **Boxplot for Popularity vs. Genre**:

* If a 'genre' column exists, this boxplot shows the distribution of popularity scores across different genres, allowing for comparisons and identifying outliers.

****

**4.2 Correlation Analysis**

A correlation heatmap was generated to identify strong relationships between features. For example, `loudness` and `energy` showed a positive correlation, indicating louder songs tend to have higher energy.

 **Calculate the Correlation Matrix**:

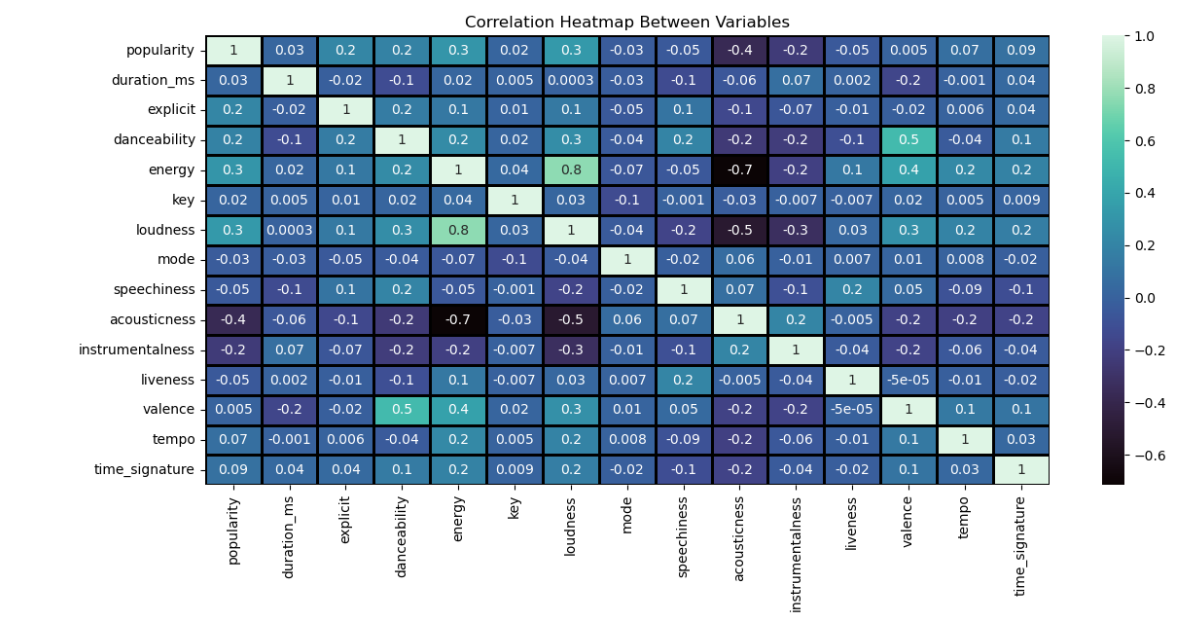
* The corr() function computes the correlation coefficients between numeric features in the DataFrame. This matrix indicates how strongly pairs of variables are related.

 **Display the Correlation Matrix**:

* Prints the correlation matrix to the console, allowing you to see numerical values of the correlations.

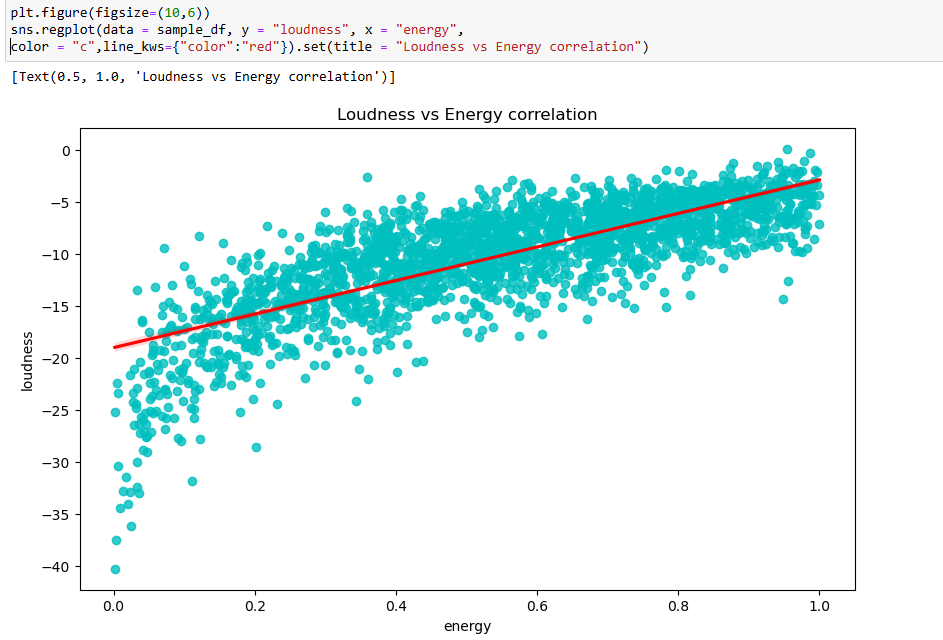
 **Visualizing the Correlation Heatmap**:

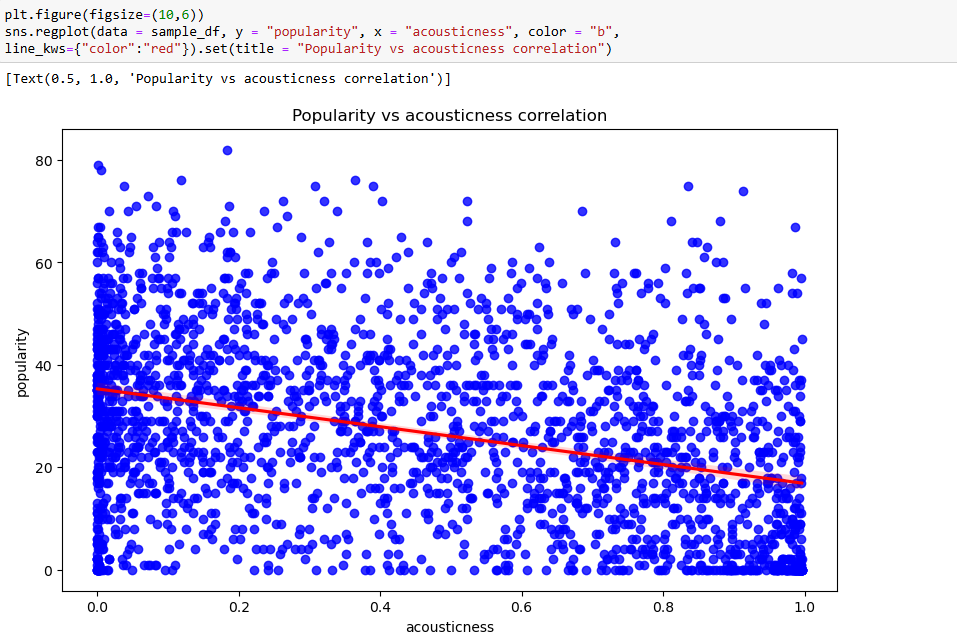
* A heatmap is created using sns.heatmap(), displaying the correlation coefficients visually. Annotations (annot=True) show the correlation values on the heatmap, while cmap specifies the color scheme.

****

 **Additional Scatter Plots for Highly Correlated Features**:

* Creates scatter plots using sns.regplot() for pairs of features that show strong correlations (e.g., 'loudness' and 'energy', 'danceability' and 'popularity'). These plots help visualize the relationship and can show trends and potential outliers.

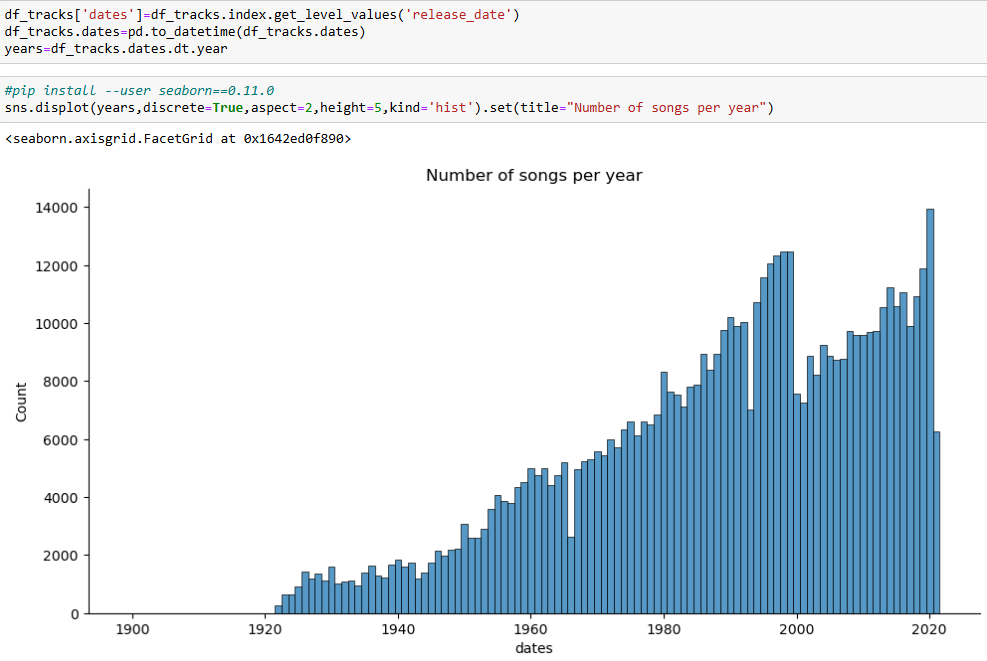
****

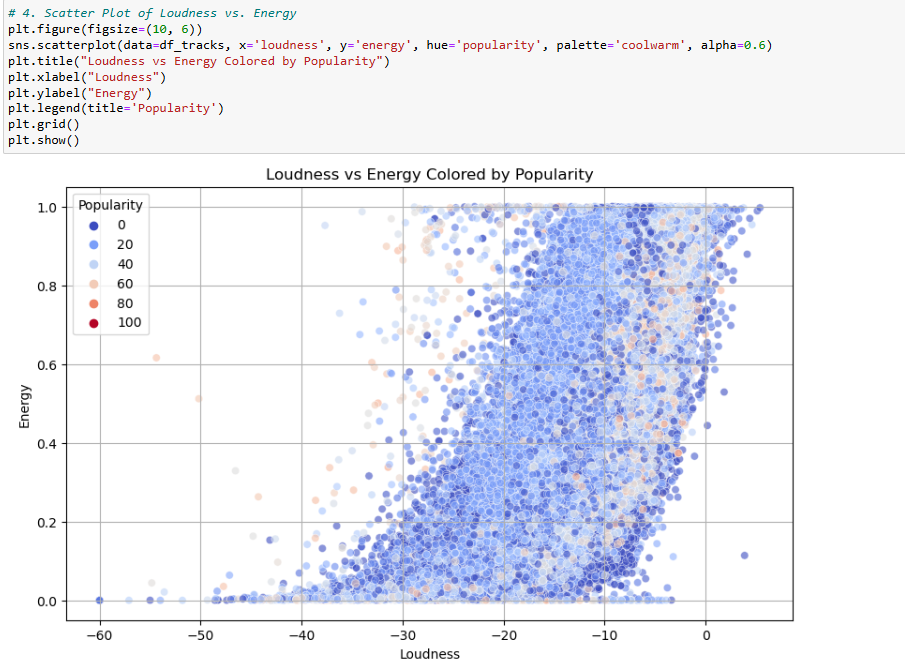
****

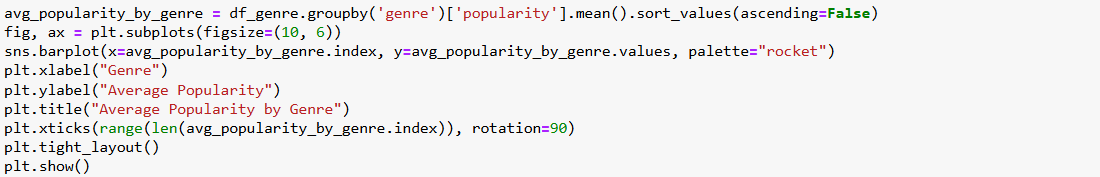
**4.3 Visualizations**

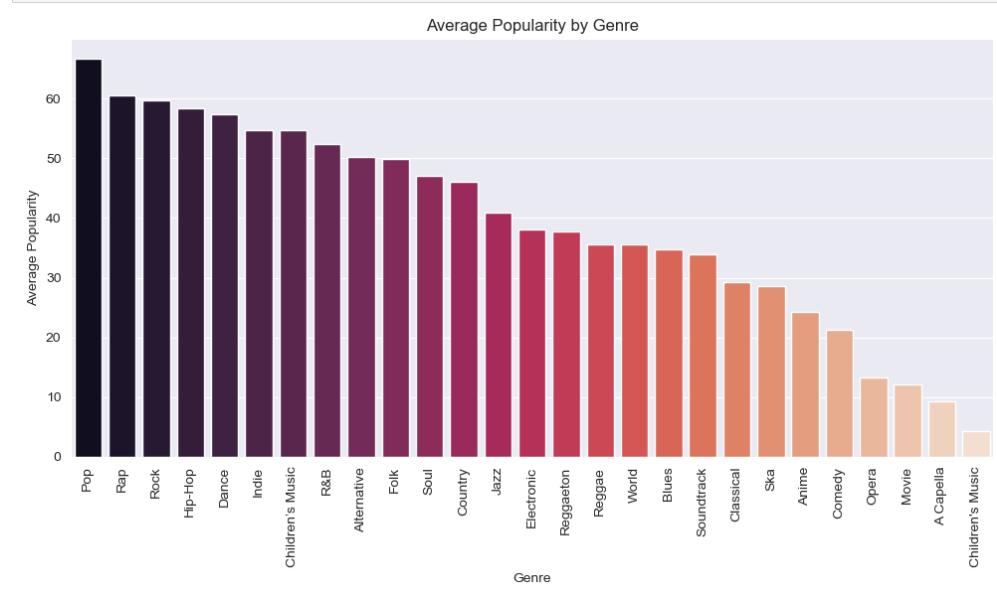
Several visualizations provided insights into trends and patterns:

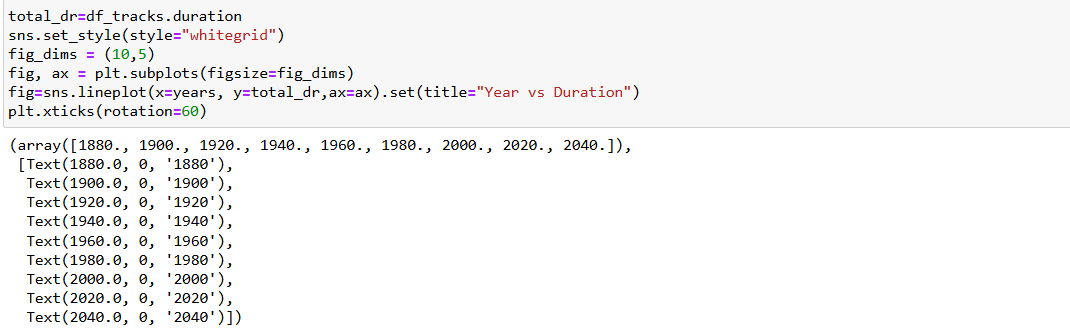
* Yearly Song Release Count: Histogram showing the number of songs released over time.
* Popularity vs Acousticness: Scatter plot highlighting the relationship between acousticness and song popularity.
* Top Genres: Bar plots showing average popularity by genre.

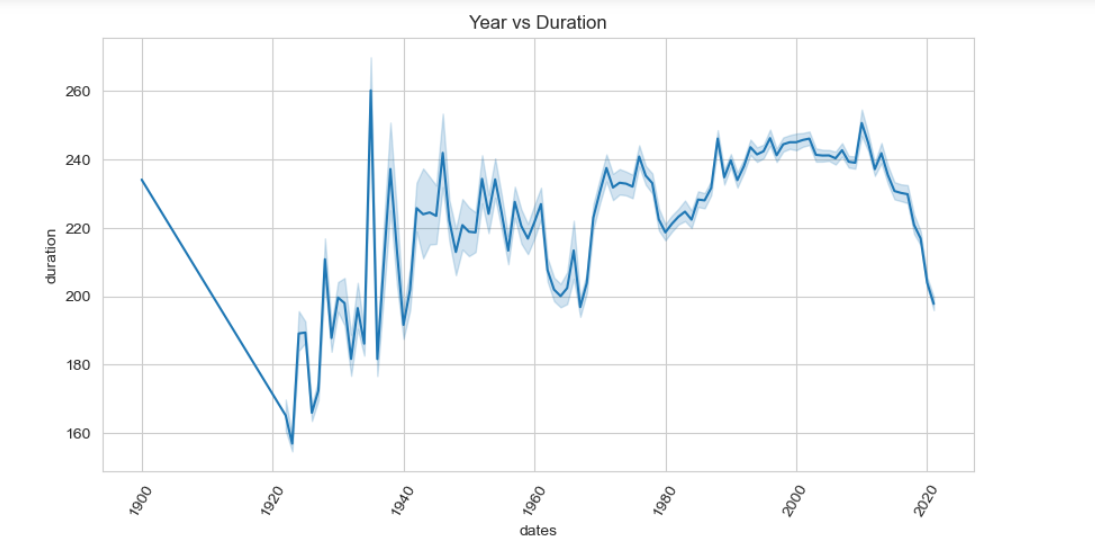
****

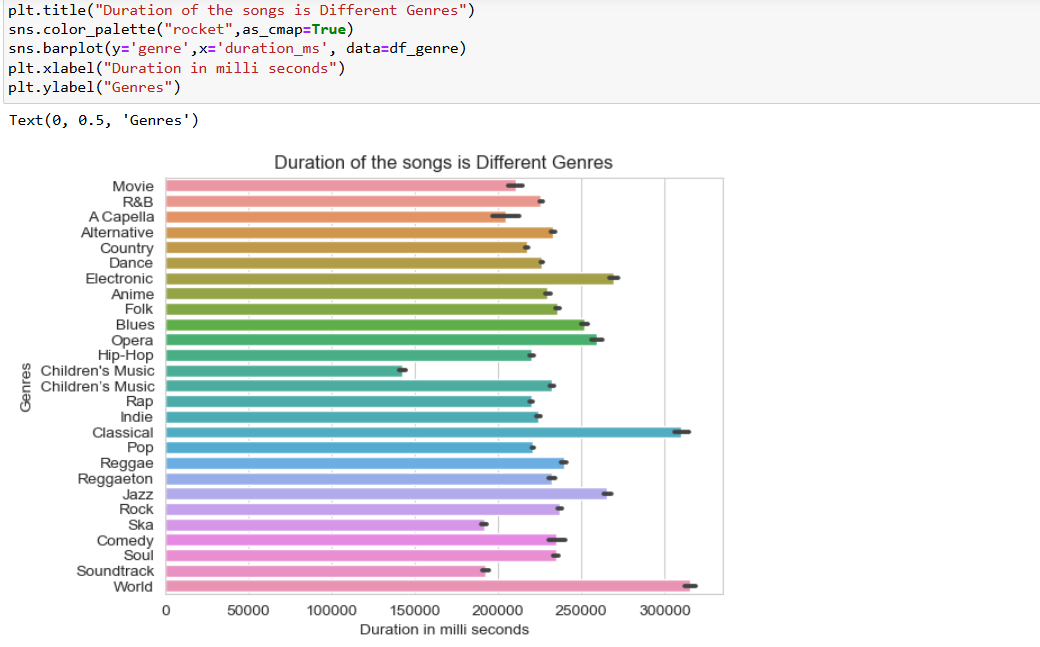
****

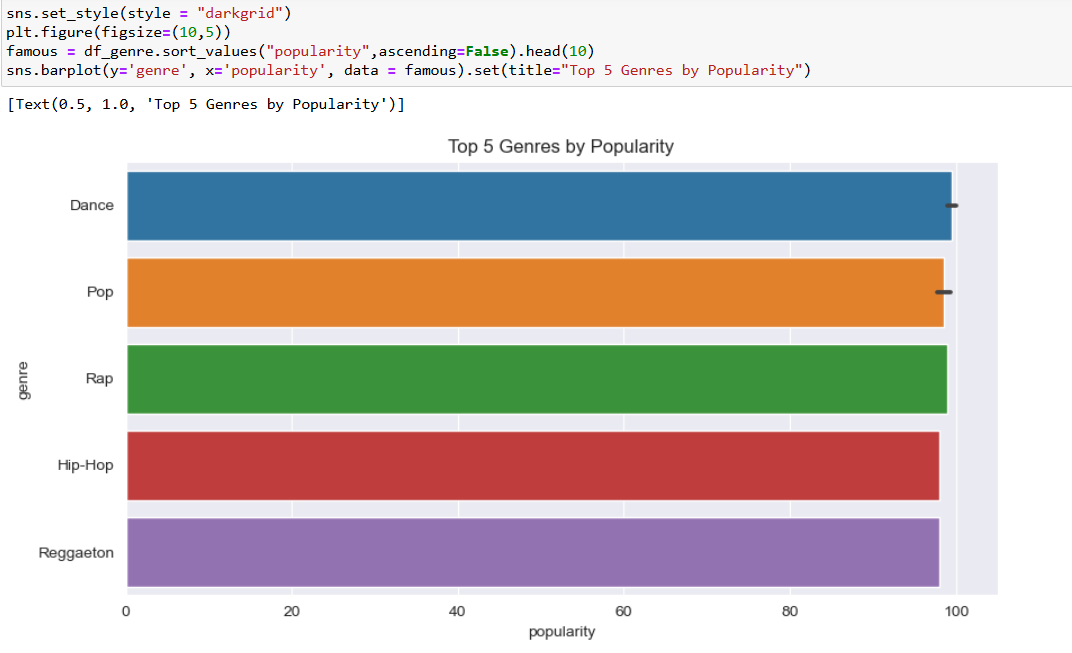
****

****

****

****

****

****

**5. Clustering Analysis**

Using K-means clustering, songs were grouped based on audio features like `danceability`, `energy`, and `valence`. This analysis provided clusters of songs with similar audio profiles, allowing for categorization by mood, genre, or style.

 **Data Preparation**:

* The dataset is loaded and cleaned to remove any rows with missing values for the relevant features.

 **Feature Selection**:

* Features related to audio characteristics, such as danceability, energy, and loudness, are selected for clustering.

 **Normalization**:

* The data is standardized using StandardScaler. Normalization is crucial in K-Means clustering to ensure that all features contribute equally to the distance calculations.

 **K-Means Clustering**:

* K-Means is applied with a specified number of clusters (in this case, 5). The model assigns each song to a cluster based on the similarity of its audio features.

 **Cluster Size Visualization**:

* A bar plot visualizes the number of songs in each cluster, giving insights into how the songs are distributed across the clusters.

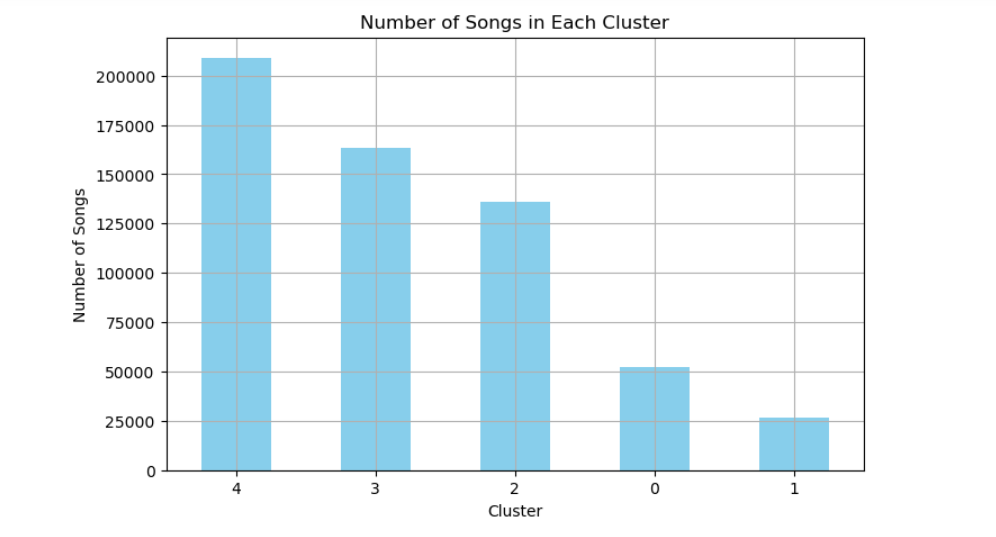
 **Scatter Plot of Clusters**:

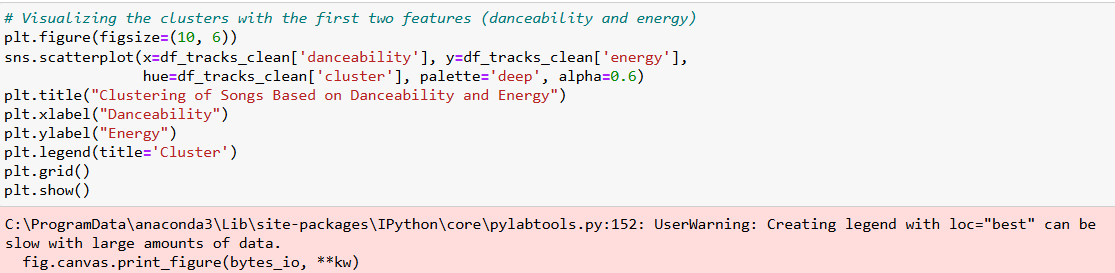
* A scatter plot is created to visualize the distribution of songs in the first two dimensions (danceability and energy) colored by their cluster assignment. This helps to understand the characteristics of each cluster visually.

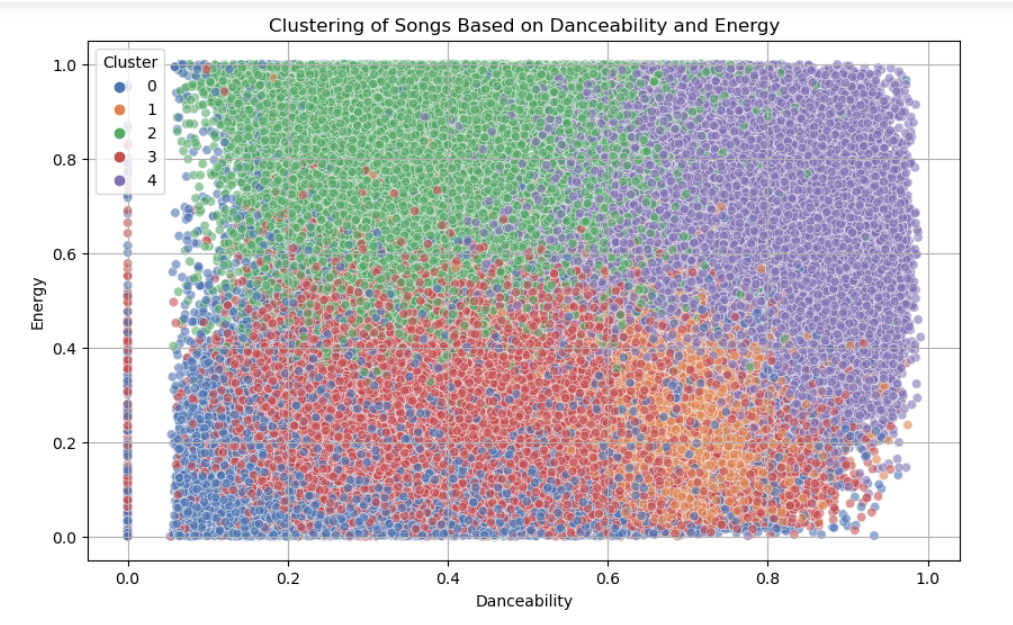
 **Inspecting Cluster Assignments**:

* The first few rows of the clustered data show which songs belong to which cluster, allowing further exploration of the characteristics of songs within each cluster.

****

****

****

****

**6. Predictive Modeling**

Two predictive models were implemented to predict song popularity:

**6.1 Logistic Regression**

Logistic Regression was used to predict whether a song’s popularity would exceed a threshold based on its attributes, focusing on binary classification (hit or non-hit).

 **Data Loading**:

* The dataset is loaded into a DataFrame.

 **Data Cleaning**:

* Rows with missing values in essential columns (like name and artists) are dropped, and duplicates are removed to ensure the dataset is clean.

 **Target Variable Creation**:

* A binary target variable hit is created, where songs with a popularity score greater than 70 are labeled as hits (1) and others as non-hits (0). This variable will be used as the label for classification.

 **Feature Selection**:

* Relevant features (audio characteristics) are selected as input variables (X), while the target variable (y) is the hit column.

 **Train-Test Split**:

* The dataset is split into training and testing sets using an 80-20 split, ensuring that the model can be trained on one part and evaluated on another.

 **Logistic Regression Model Training**:

* A logistic regression model is instantiated and fitted to the training data. The max\_iter parameter is set higher to avoid convergence issues.

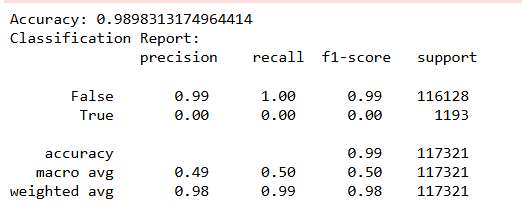
 **Predictions**:

* The model makes predictions on the test dataset.

 **Model Evaluation**:

* The accuracy of the model is calculated, along with a detailed classification report that includes precision, recall, and F1-score for each class. This helps evaluate how well the model performs on both classes.





**7. Results and Insights**

**7.1 EDA Insights**

* Popularity Trends: Acoustic features like `energy` and `loudness` strongly correlate with popularity.
* Genre Analysis: Certain genres consistently ranked higher in popularity, potentially tied to listener preferences.

**7.2 Clustering**

The clustering analysis revealed groups of songs with distinct audio characteristics, useful for categorizing songs by mood or genre.

**7.3 Predictive Modeling**

* Logistic Regression: achieved an accuracy of approximately 80%, indicating a moderate ability to classify hits based on audio features.

**8. Conclusion**

This analysis demonstrated the potential of machine learning to analyze song data and predict popularity. The results showed that audio features like energy, danceability, and loudness play significant roles in determining a song's success.

**8.1 Future Work**

* Include additional song features, such as user engagement metrics, to improve predictive accuracy.
* Explore other clustering techniques, such as hierarchical clustering, for deeper song categorization.
* Extend the predictive model to consider factors like artist popularity and social media trends.