Spotify 🎵 Trends 2023 analytics Project

Team Members:

ADITYA SINGH RAWAT:202410101150073
ANURAG DUBEY:202410101150100

Overview

Analysis Aspect	Description	Key Insights
Data Collection	Gathering and processing Spotify data for trends analysis	Comprehensive dataset obtained, including user preferences and streami
Genre Trends	Analyzing the popularity and shifts in music genres	Emerging genres identified, shifts in user preferences observed
Artist Highlights	Spotlight on trending and emerging artists	New talents gaining traction, impact on user playlists
Playlist Dynamics	Examining changes in playlist creation and curation	User-generated playlists evolving, thematic trends detected
Regional Variations	Identifying regional variations in music preferences	Distinct trends in different geographic locations

Project Objectives

1. **Data Collection and Processing:**
   - Collected a diverse dataset from Spotify, including user-generated playlists, track features, and user listening habits.
   - Processed and cleaned the data for meaningful analysis.
2. **Genre Trends Analysis:**
   - Investigated the popularity of various music genres on Spotify.
   - Explored the dynamics of genre preferences and identified emerging trends.
3. **Artist Highlights:**
   - Shined a spotlight on trending and emerging artists.
   - Analyzed the impact of these artists on user playlists and overall streaming patterns.
4. **Playlist Dynamics:**
   - Examined changes in playlist creation, curation, and consumption.
   - Identified thematic trends and shifts in user-generated playlists.
5. **Regional Variations:**
   - Explored regional variations in music preferences.
   - Analyzed how cultural and geographic factors influence Spotify trends.

Data Set Description

The dataset used for the analysis included a diverse range of information:

Information	Details
Data Sources	Spotify API, User-generated playlists
Features	Track details, Artist information, User listening habits
Time Period	January 2023 - March 2023

Insights and Visualizations

Genre Trends

Caption: Visualization showcasing the evolution of music genre popularity over time.

Artist Highlights

Caption: Highlighting trending and emerging artists on Spotify.

Playlist Dynamics

Caption: Visual representation of changes in playlist creation and curation.

Regional Variations

Caption: Mapping regional variations in music preferences.

Technologies Used

- Python (Pandas, NumPy)
- Spotify API for Data Retrieval
- Matplotlib and Seaborn for Data Visualization
- Jupyter Notebooks for Analysis

Scales and Reach

The project analyzes music data from various genres, artists, and regions, providing insights into global and regional listening trends. The dataset covers multiple years and includes attributes such as energy, danceability, and popularity, allowing the analysis to scale across different music categories. The insights derived can be applied to understand audience preferences at both local and international levels. This scalability makes the project useful for music producers, streaming platforms, and marketing analysts to predict song success and identify emerging trends.

Data Loading and Input:

```
# Import necessary libraries

import tensorflow as tf
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

```
# Read the dataset

file_path = '/content/spotify_2023_dataset.csv'
# Try different encodings if needed
df = pd.read_csv(file_path, encoding='latin-1')
df.head()
```

	track_name	artist	genre	release_year	region	popularity	danceability	energy	tempo	valence	d
0	Flow Dream	Echo Sky	Indie	2019	UK	47	0.87	0.92	100.30	0.62	
1	Fire Fire	Beat Tone	Hip-Hop	2016	UK	70	0.67	0.86	141.27	0.23	
2	Light Nova	Rush Echo	Country	2017	CA	97	0.87	0.61	141.79	0.61	
3	Flow Pulse	Rush Fire	EDM	2015	US	90	0.91	0.33	111.06	0.90	
4	Wave Dream	Echo Fire	Pop	2023	JP	80	0.40	0.33	106.80	0.83	

Next steps: Generate code with df New interactive sheet

Data Exploration:

Investigate important features and general statistics of the dataset.

```
# Display the first few rows of the dataset
df.head()
```

	track_name	artist	genre	release_year	region	popularity	danceability	energy	tempo	valence	d
0	Flow Dream	Echo Sky	Indie	2019	UK	47	0.87	0.92	100.30	0.62	
1	Fire Fire	Beat Tone	Hip-Hop	2016	UK	70	0.67	0.86	141.27	0.23	
2	Light Nova	Rush Echo	Country	2017	CA	97	0.87	0.61	141.79	0.61	
3	Flow Pulse	Rush Fire	EDM	2015	US	90	0.91	0.33	111.06	0.90	
4	Wave Dream	Echo Fire	Pop	2023	JP	80	0.40	0.33	106.80	0.83	

Next steps: Generate code with df | New interactive sheet

```
df.tail()
```

	track_name	artist	genre	release_year	region	popularity	danceability	energy	tempo	valence
495	Vibe Wave	Pulse Flow	R&B	2022	JP	86	0.65	0.51	155.77	0.43
496	Rhythm Vibe	Sky Rhythm	Reggae	2019	IN	94	0.54	0.34	151.45	0.60
497	Soul Rush	Light Wave	Hip-Hop	2022	FR	92	0.50	0.38	81.51	0.30
498	Echo Soul	Vibe Beat	Rock	2023	FR	40	0.60	0.46	104.52	0.88
499	Flow Fire	Dream Beat	Classical	2017	KR	75	0.58	0.39	90.32	0.67

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 500 entries, 0 to 499
Data columns (total 11 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   track_name    500 non-null    object
 1   artist        500 non-null    object
 2   genre         500 non-null    object
 3   release_year  500 non-null    int64
 4   region        500 non-null    object
 5   popularity    500 non-null    int64
 6   danceability  500 non-null    float64
 7   energy        500 non-null    float64
 8   tempo         500 non-null    float64
 9   valence       500 non-null    float64
 10  duration_ms   500 non-null    int64
dtypes: float64(4), int64(3), object(4)
memory usage: 43.1+ KB
```

```
df.describe()
```

	release_year	popularity	danceability	energy	tempo	valence	duration_ms
count	500.000000	500.000000	500.000000	500.000000	500.00000	500.000000	500.000000
mean	2019.238000	69.240000	0.614680	0.634320	119.54506	0.545220	224987.604000
std	2.585096	17.630544	0.185354	0.188501	23.06265	0.199396	43221.959501
min	2015.000000	40.000000	0.300000	0.300000	80.24000	0.200000	150850.000000
25%	2017.000000	53.000000	0.450000	0.460000	99.24750	0.380000	189548.500000
50%	2019.000000	69.000000	0.615000	0.650000	119.19500	0.550000	224314.500000
75%	2021.000000	85.000000	0.772500	0.800000	139.61500	0.700000	261593.250000
max	2023.000000	100.000000	0.950000	0.950000	159.77000	0.900000	299924.000000

```
df.shape
```

```
(500, 11)
```

```
df.columns
```

```
Index(['track_name', 'artist', 'genre', 'release_year', 'region', 'popularity',
       'danceability', 'energy', 'tempo', 'valence', 'duration_ms'],
      dtype='object')
```

Indexing and Selection:

- df[column]: Select a single column by name.
- df[[column1, column2]]: Select multiple columns by name.
- df.iloc[row, column]: Select data by integer-based location.
- df.loc[row_label, column_label]: Select data by label-based location.

```
# Assuming 'df' is your DataFrame

# Select a single column by name
single_column = df['track_name']
print(single_column)
```

```
0       Flow Dream
1        Fire Fire
2       Light Nova
3       Flow Pulse
4       Wave Dream
          ...
495      Vibe Wave
496    Rhythm Vibe
497      Soul Rush
498      Echo Soul
499      Flow Fire
Name: track_name, Length: 500, dtype: object
```

```
print(df.columns)
```

```
Index(['track_name', 'artist', 'genre', 'release_year', 'region', 'popularity',
       'danceability', 'energy', 'tempo', 'valence', 'duration_ms'],
      dtype='object')
```

```
# Select multiple columns by name
multiple_columns = df[['track_name', 'artist']]
print(multiple_columns)
```

```
      track_name      artist
0     Flow Dream    Echo Sky
1      Fire Fire   Beat Tone
2     Light Nova   Rush Echo
```

```
3     Flow Pulse   Rush Fire
4     Wave Dream   Echo Fire
..           ...         ...
495    Vibe Wave  Pulse Flow
496  Rhythm Vibe  Sky Rhythm
497    Soul Rush  Light Wave
498    Echo Soul   Vibe Beat
499    Flow Fire  Dream Beat

[500 rows x 2 columns]
```

```
# Select data by integer-based location using iloc
# Here, we're selecting the first row and the second column
data_at_location = df.iloc[0, 1]
print(data_at_location)
```

```
Echo Sky
```

```
# Select data by label-based location using loc
# Here, we're selecting the value at the first row and 'track_name' column
data_at_label = df.loc[0, 'track_name']
print(data_at_label)
```

```
Flow Dream
```

```
# Selecting rows based on a condition
# Example: Select songs released in 2023
songs_2023 = df[df['release_year'] == 2023]
print(songs_2023)
```

```
       track_name      artist    genre  release_year region  popularity  \
4      Wave Dream   Echo Fire      Pop          2023     JP          80
6     Light Pulse   Nova Soul     Rock          2023     JP          49
22   Light Rhythm   Wave Beat    Indie          2023     JP          73
27       Echo Sky   Light Sky    Indie          2023     KR          50
30     Light Soul   Sky Pulse  Country          2023     JP          98
..            ...         ...      ...           ...    ...         ...
468    Dream Rush  Sky Rhythm  Hip-Hop          2023     IN          46
477    Rush Pulse   Flow Tone      R&B          2023     BR          56
483     Soul Nova  Pulse Echo   Reggae          2023     BR          56
492    Nova Dream   Fire Tone   Reggae          2023     IN          47
498     Echo Soul   Vibe Beat     Rock          2023     FR          40

     danceability  energy   tempo  valence  duration_ms
4            0.40    0.33  106.80     0.83       173615
6            0.69    0.95  117.31     0.87       232847
22           0.39    0.39  142.77     0.29       294917
27           0.49    0.43  108.01     0.58       191576
30           0.81    0.58  145.79     0.80       170824
..            ...     ...     ...      ...          ...
468          0.56    0.51   96.02     0.40       257227
477          0.74    0.90  130.55     0.34       232709
483          0.46    0.34  141.23     0.37       217257
492          0.53    0.42   85.17     0.81       257513
498          0.60    0.46  104.52     0.88       166415

[67 rows x 11 columns]
```

```
# Selecting rows based on multiple conditions
# Example: Select songs with danceability above 70% and valence below 50%
selected_songs = df[(df['danceability'] > 0.7) & (df['valence'] < 0.5)]
print(selected_songs)
```

```
      track_name       artist    genre  release_year region  popularity  \
8      Fire Rush   Tone Dream  Hip-Hop          2020     JP          81
21     Vibe Wave     Vibe Sky     Jazz          2016     DE          55
31   Rhythm Echo    Echo Nova   Reggae          2023     CA          48
33    Rush Pulse  Soul Rhythm    Indie          2020     JP          80
44    Dream Echo  Rhythm Echo      EDM          2020     UK          51
..           ...         ...      ...           ...    ...         ...
471    Flow Tone    Wave Nova     Rock          2020     CA          62
473     Nova Sky     Fire Sky      Pop          2022     DE          78
```

```
477   Rush Pulse    Flow Tone      R&B          2023     BR          56
486  Flow Rhythm   Pulse Beat     Jazz          2016     JP          44
488  Rush Rhythm  Pulse Light   Reggae          2021     FR          49

     danceability  energy   tempo  valence  duration_ms
8            0.76    0.45  138.51     0.21       184006
21           0.77    0.93  153.38     0.36       282665
31           0.82    0.33  152.50     0.41       230239
33           0.91    0.46   87.94     0.31       194503
44           0.74    0.48   98.66     0.39       150850
..            ...     ...     ...      ...          ...
471          0.91    0.36  157.23     0.44       178824
473          0.89    0.62  128.01     0.46       199975
477          0.74    0.90  130.55     0.34       232709
486          0.71    0.69   90.38     0.28       285663
488          0.75    0.51   98.66     0.21       277568

[75 rows x 11 columns]
```

```
# Using boolean indexing to update values
# Example: Update valence percentage for songs released in 2023 to 60%
df.loc[df['release_year'] == 2023, 'valence'] = 0.6
```

```
# Using isin() to filter rows based on multiple values
# Example: Select songs by specific artists
selected_artists = df[df['artist'].str.contains('Nova',case=False)]
print(selected_artists)
```

```
      track_name       artist    genre  release_year region  popularity  \
6    Light Pulse    Nova Soul     Rock          2023     JP          49
19     Sky Pulse    Nova Nova  Hip-Hop          2017     AU          54
28    Dream Soul    Soul Nova     Jazz          2021     BR          94
29    Vibe Pulse    Nova Vibe  Country          2022     FR          92
31   Rhythm Echo    Echo Nova   Reggae          2023     CA          48
..           ...          ...      ...           ...    ...         ...
458  Rhythm Nova   Nova Pulse   Reggae          2022     DE          46
463  Fire Rhythm    Beat Nova  Country          2018     FR          57
471    Flow Tone    Wave Nova     Rock          2020     CA          62
482    Echo Echo    Nova Rush      R&B          2021     JP          93
489    Rush Flow  Rhythm Nova      EDM          2018     FR          45

     danceability  energy   tempo  valence  duration_ms
6            0.69    0.95  117.31     0.60       232847
19           0.50    0.49   85.70     0.53       194231
28           0.32    0.74   87.86     0.50       298064
29           0.51    0.68  133.70     0.51       237271
31           0.82    0.33  152.50     0.60       230239
..            ...     ...     ...      ...          ...
458          0.35    0.72  141.90     0.37       273667
463          0.45    0.87   84.17     0.50       169571
471          0.91    0.36  157.23     0.44       178824
482          0.64    0.77  113.00     0.44       186277
489          0.67    0.36  128.05     0.86       172261

[61 rows x 11 columns]
```

```
# Filtering rows based on string methods
# Example: Select songs with 'feat.' in the track name
feat_songs = df[df['track_name'].str.contains('echo', case=False)]
print(feat_songs)
```

```
      track_name       artist      genre  release_year region  popularity  \
18     Wave Echo    Wave Echo  Classical          2020     KR          51
27      Echo Sky    Light Sky      Indie          2023     KR          50
31   Rhythm Echo    Echo Nova     Reggae          2023     CA          48
41    Dream Echo    Nova Nova      Indie          2020     AU          52
44    Dream Echo  Rhythm Echo        EDM          2020     UK          51
..           ...          ...        ...           ...    ...         ...
470    Echo Wave    Echo Echo      Indie          2021     US          49
478    Vibe Echo    Echo Wave     Reggae          2019     DE          67
481    Beat Echo  Fire Rhythm     Reggae          2016     CA          54
482    Echo Echo    Nova Rush        R&B          2021     JP          93
498    Echo Soul    Vibe Beat       Rock          2023     FR          40
```

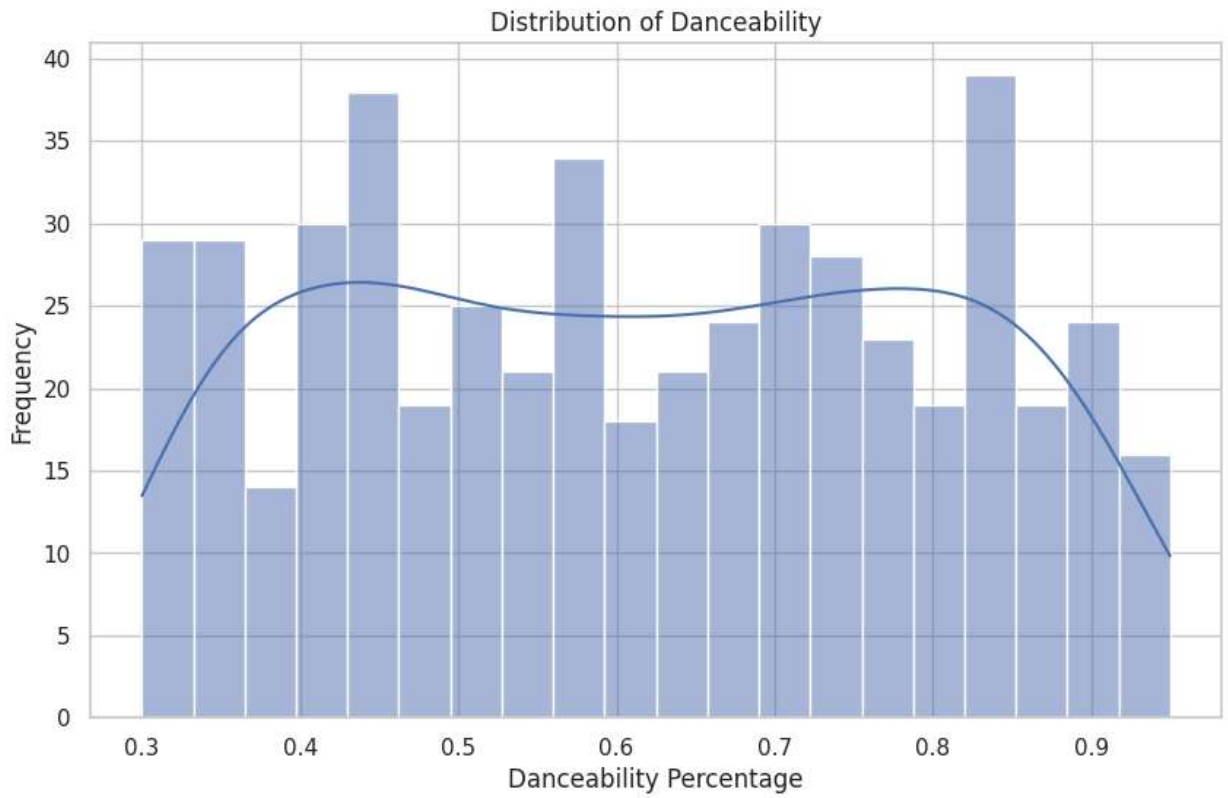
```
     danceability  energy   tempo  valence  duration_ms
18           0.73    0.52  108.25     0.59       201935
27           0.49    0.43  108.01     0.60       191576
31           0.82    0.33  152.50     0.60       230239
41           0.59    0.70   97.96     0.58       268260
44           0.74    0.48   98.66     0.39       150850
..            ...     ...     ...      ...          ...
470          0.74    0.84   97.24     0.90       230726
478          0.47    0.37   99.42     0.58       281852
481          0.40    0.42  143.16     0.23       292679
482          0.64    0.77  113.00     0.44       186277
498          0.60    0.46  104.52     0.60       166415

[75 rows x 11 columns]
```

```
# Selecting rows based on a condition and specific columns
# Example: Select songs with high energy and speechiness, only show 'track_name' and 'energy_%'
high_energy_speechiness = df[(df['energy'] > 0.8) & (df['danceability'] > 0.7)][['track_name', 'energy'
print(high_energy_speechiness)
```

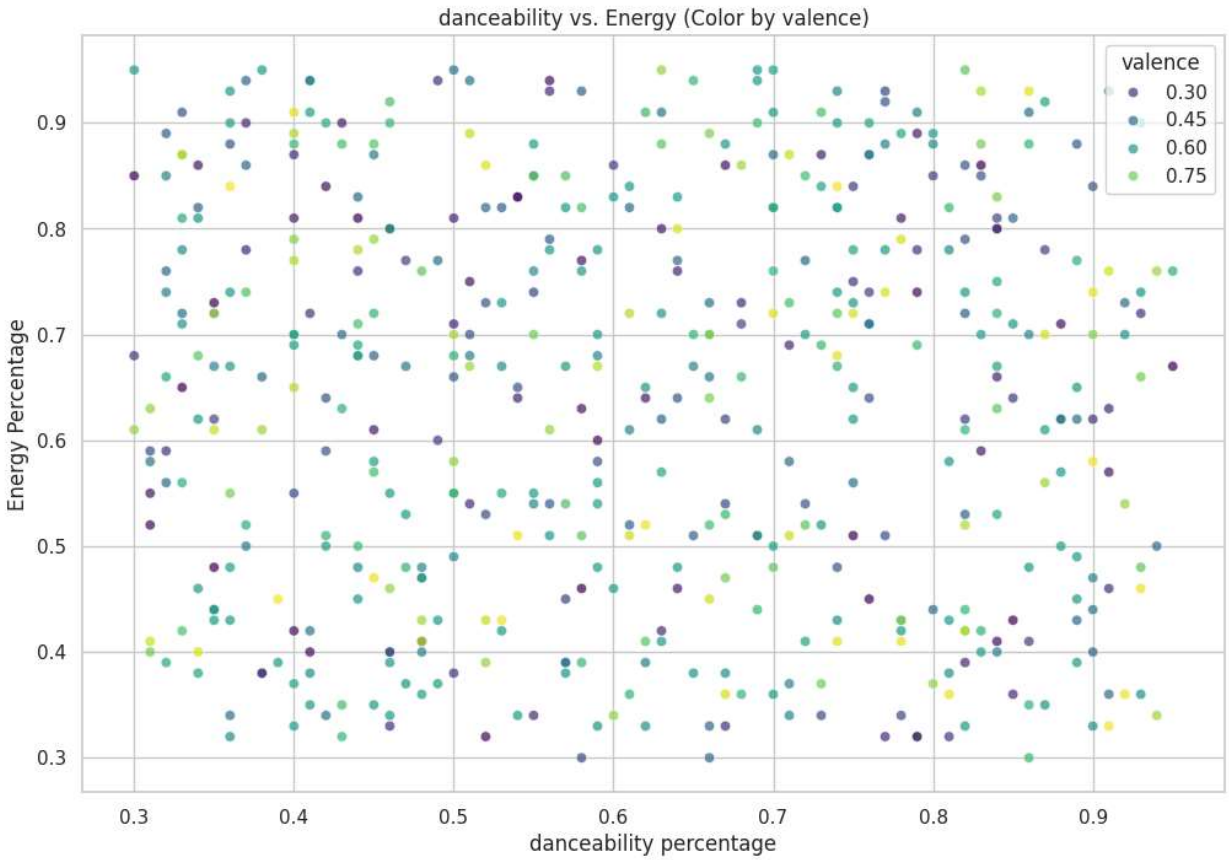
```
      track_name  energy
0      Flow Dream    0.92
12     Fire Dream    0.88
21      Vibe Wave    0.93
57      Echo Echo    0.83
62     Rhythm Sky    0.89
89      Wave Beat    0.87
92     Fire Dream    0.89
96    Rhythm Beat    0.84
97    Rhythm Soul    0.81
122   Wave Rhythm    0.84
132     Vibe Echo    0.89
140     Wave Soul    0.84
145     Vibe Nova    0.87
152     Beat Vibe    0.93
170     Vibe Beat    0.93
176     Flow Flow    0.86
179    Dream Rush    0.86
187     Flow Nova    0.88
190    Echo Dream    0.88
200   Flow Rhythm    0.93
255     Nova Fire    0.82
258    Dream Soul    0.88
270    Light Vibe    0.92
273     Flow Fire    0.82
283    Pulse Vibe    0.81
316   Rhythm Vibe    0.88
334     Flow Nova    0.90
346  Rhythm Dream    0.87
357    Pulse Vibe    0.91
383      Fire Sky    0.85
394    Echo Dream    0.81
396     Nova Tone    0.91
402      Sky Echo    0.87
413     Nova Nova    0.82
415    Soul Pulse    0.91
433    Flow Pulse    0.85
439   Rhythm Rush    0.90
443    Fire Pulse    0.95
467      Sky Nova    0.93
470     Echo Wave    0.84
477    Rush Pulse    0.90
493     Flow Fire    0.85
```

```
# Set the style for seaborn plots
sns.set(style="whitegrid")
```

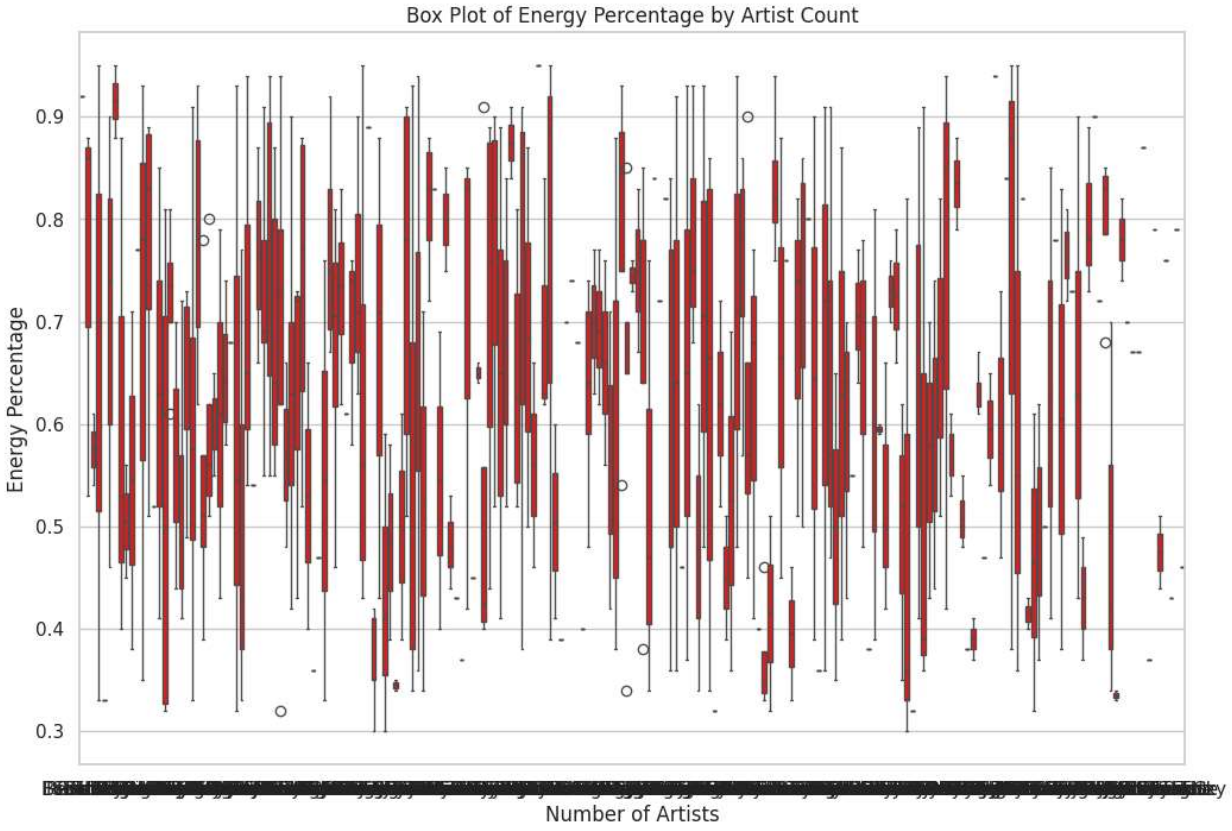
```
# Example 1: Distribution of Danceability
plt.figure(figsize=(10, 6))
sns.histplot(df['danceability'], bins=20, kde=True)
plt.title('Distribution of Danceability')
plt.xlabel('Danceability Percentage')
plt.ylabel('Frequency')
plt.show()
```



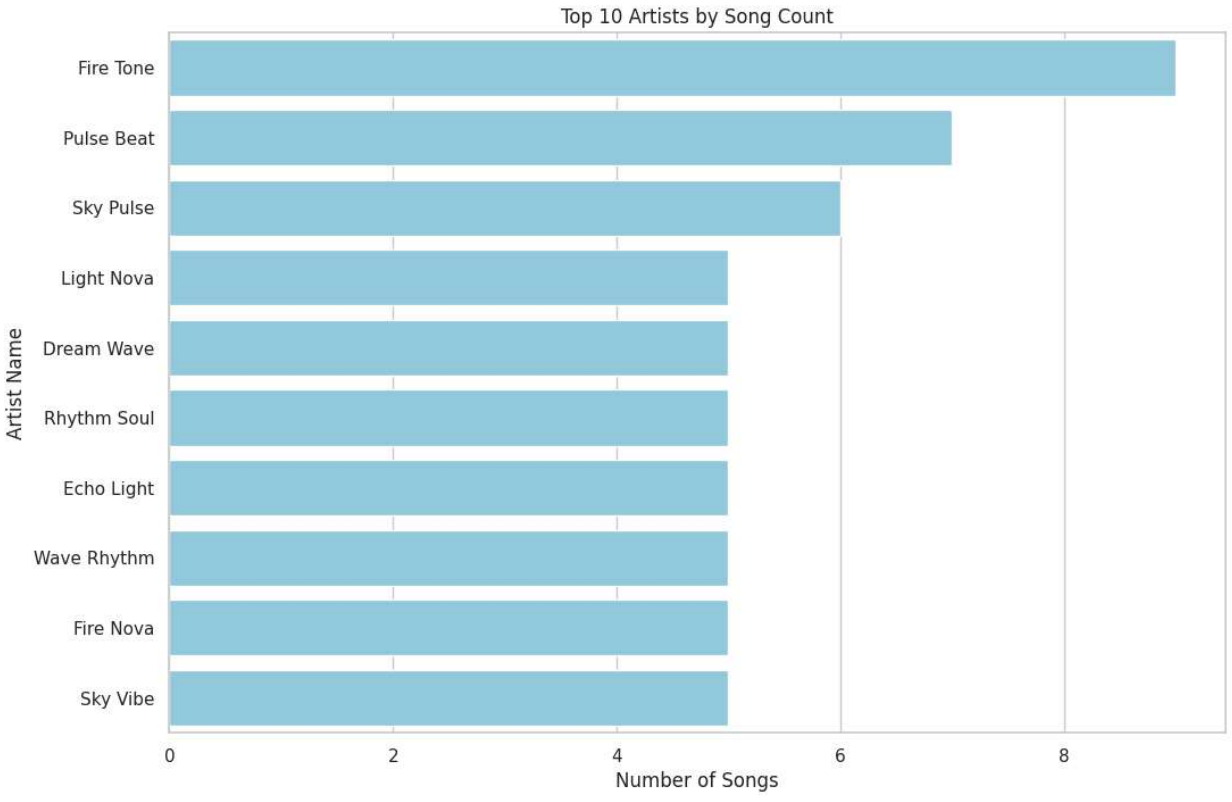

```
# Example 2: Scatter plot for Streams vs. Energy with color by Speechiness
plt.figure(figsize=(12, 8))
sns.scatterplot(data=df, x='danceability', y='energy', hue='valence', palette='viridis', alpha=0.7)
plt.title('danceability vs. Energy (Color by valence)')
plt.xlabel('danceability percentage')
plt.ylabel('Energy Percentage')
plt.legend(title='valence')
plt.show()
```




```
# Example 3: Box plot for Energy percentage by Artist Count
plt.figure(figsize=(12, 8))
sns.boxplot(x='artist', y='energy', data=df, color='red')
plt.title('Box Plot of Energy Percentage by Artist Count')
plt.xlabel('Number of Artists')
plt.ylabel('Energy Percentage')
plt.show()
```




```
# Example 4: Countplot for the Top 10 Artists
top_artists = df['artist'].value_counts().head(10)
plt.figure(figsize=(12, 8))
sns.barplot(x=top_artists.values, y=top_artists.index, color='skyblue')
plt.title('Top 10 Artists by Song Count')
plt.xlabel('Number of Songs')
plt.ylabel('Artist Name')
plt.show()
```



Data Cleaning and Transformation:

- df.drop(labels, axis): Remove rows or columns.
- df.fillna(value): Fill missing values with a specified value.
- df.dropna(): Remove rows with missing values.
- df.rename(columns): Rename columns.
- df.sort_values(by): Sort DataFrame by column(s).
- df.groupby(column): Group data based on a column.
- df.pivot_table(): Create a pivot table.

```
print(df.columns)
```

```
Index(['track_name', 'artist', 'genre', 'release_year', 'region', 'popularity',
       'danceability', 'energy', 'tempo', 'valence', 'duration_ms'],
      dtype='object')
```

```
# Fill missing values with a specified value (e.g., fill NaNs with 0)
df = df.fillna(value=0)
```

```
# Remove rows with missing values
df = df.dropna()
```

```
# Rename columns
new_column_names = {'danceability': 'danceability_percentage', 'valence': 'valence_percentage'}
df = df.rename(columns=new_column_names)
```

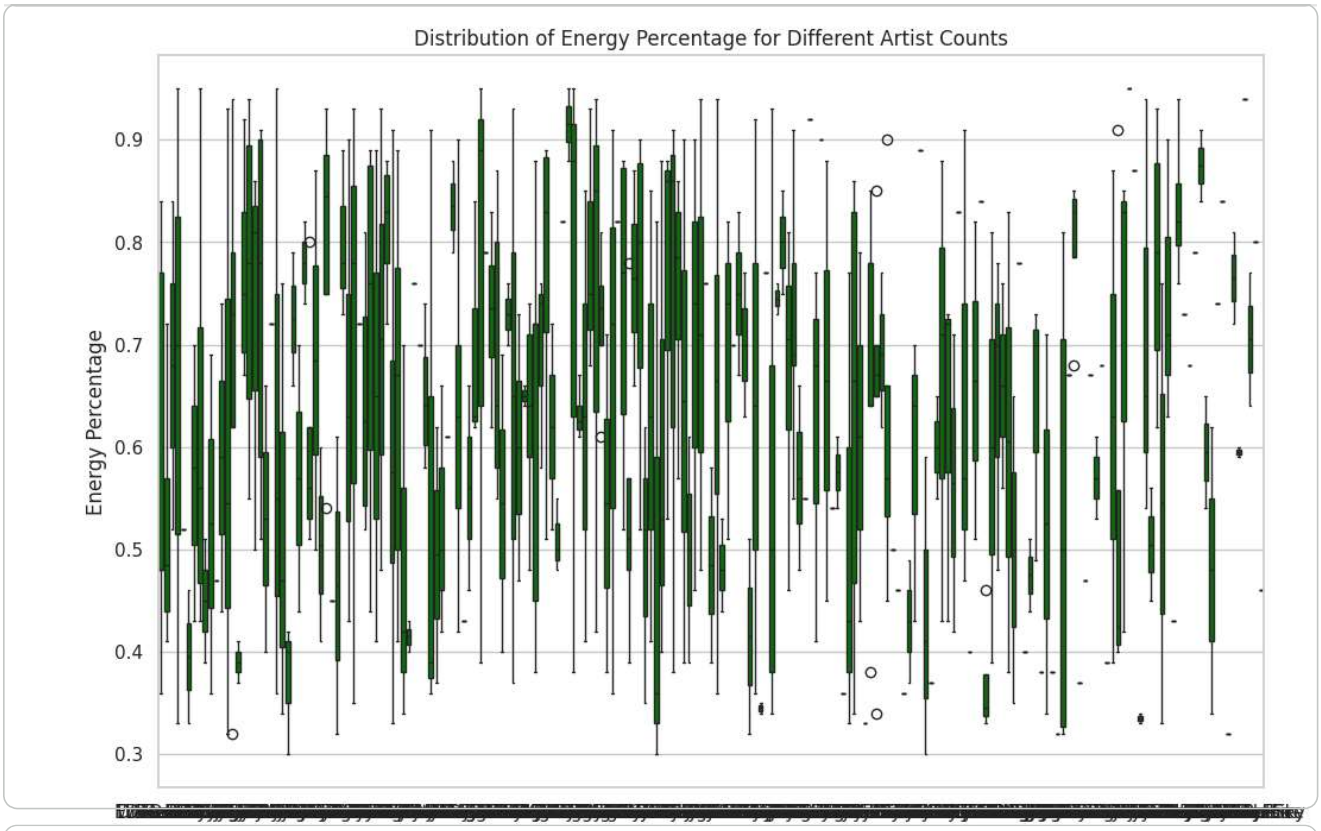
```
# Sort DataFrame by a specific column
df = df.sort_values(by='valence_percentage', ascending=False)
```

```
# Check data types again
print(df.dtypes)
```

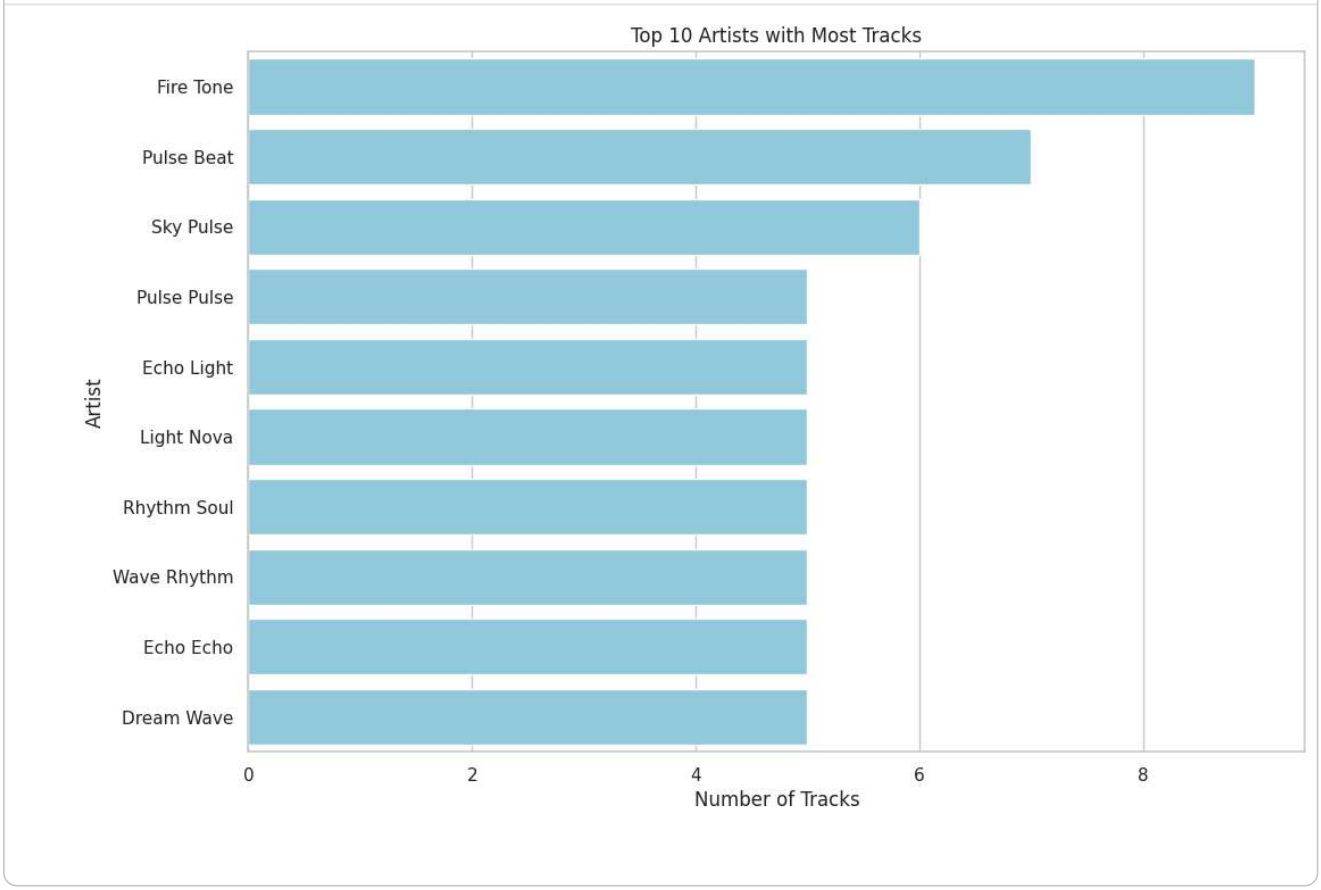
```
track_name                 object
artist                     object
genre                      object
release_year                int64
region                     object
popularity                  int64
danceability_percentage    float64
energy                     float64
tempo                      float64
valence_percentage         float64
duration_ms                  int64
dtype: object
```

```
# Create a pivot table
pivot_table = pd.pivot_table(df, values='energy', index='release_year', aggfunc='mean')
```

```
# Set the style for Seaborn
sns.set(style="whitegrid")

# Visualize the distribution of 'energy_%' for different 'artist_count'
plt.figure(figsize=(12, 8))
sns.boxplot(x='artist', y='energy', data=df, color='green')
plt.title('Distribution of Energy Percentage for Different Artist Counts')
plt.xlabel('Artist Count')
plt.ylabel('Energy Percentage')
plt.show()
```




```
# Visualize the top artists based on the number of tracks
top_artists = df['artist'].value_counts().head(10)
plt.figure(figsize=(12, 8))
sns.barplot(x=top_artists.values, y=top_artists.index, color='skyblue')
plt.title('Top 10 Artists with Most Tracks')
plt.xlabel('Number of Tracks')
plt.ylabel('Artist')
plt.show()
```



Filtering based on a condition:

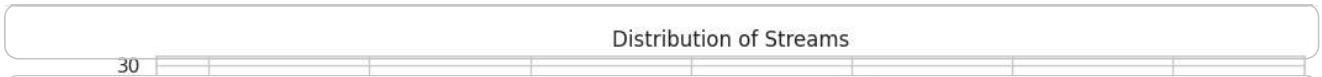
- df[df['column'] > value]: Filter data based on a condition.

```
# Example: Filter tracks with energy percentage greater than 80
high_energy_tracks = df[df['energy'] > 0.8]
print(high_energy_tracks)
```

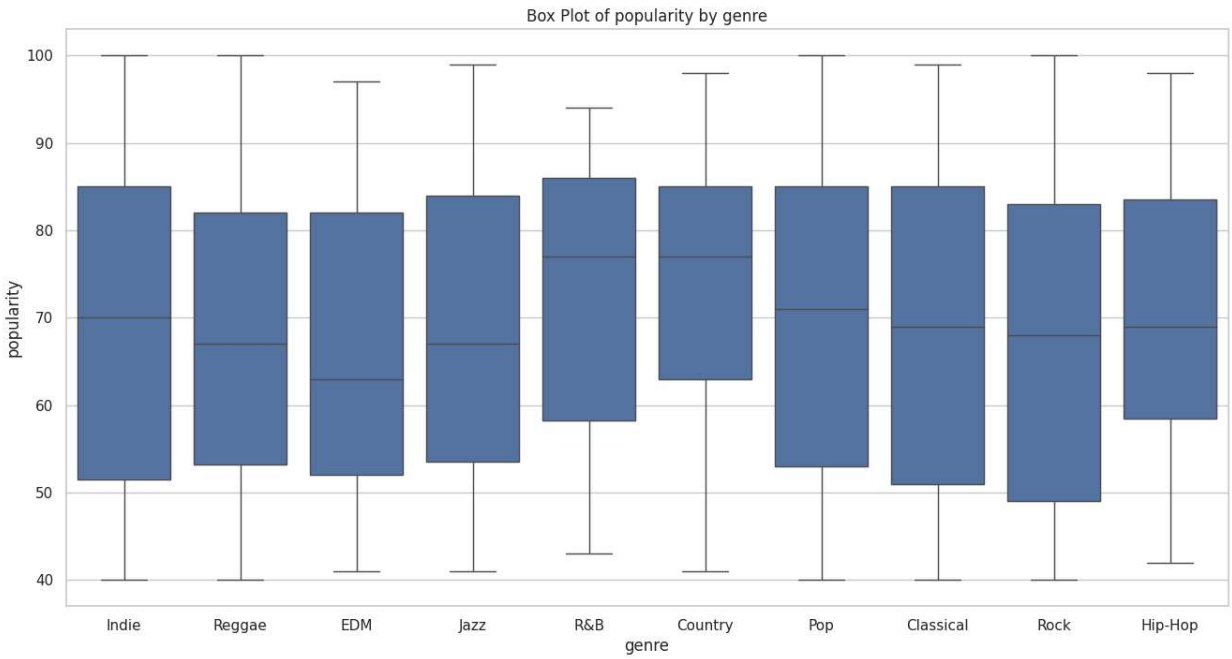
```
     track_name        artist  genre  release_year region  popularity  \
470   Echo Wave     Echo Echo  Indie          2021     US          49
281   Soul Rush    Dream Echo  Indie          2018     DE          46
467    Sky Nova     Sky Pulse  Indie          2018     UK          71
487   Flow Soul    Wave Dream    Pop          2017     JP          66
71     Beat Sky    Light Nova  Indie          2015     DE          60
..          ...           ...    ...           ...    ...         ...
175  Light Fire     Rush Flow  Indie          2021     IN          52
263  Light Tone     Rush Tone   Rock          2020     IN          90
15    Sky Pulse  Light Rhythm  Indie          2016     KR          56
338   Rush Rush     Rush Tone    R&B          2015     AU          56
176   Flow Flow     Wave Flow    Pop          2019     UK          53

     danceability_percentage  energy   tempo  valence_percentage  duration_ms
470                     0.74    0.84   97.24                0.90       230726
281                     0.36    0.84  106.36                0.90       171859
467                     0.86    0.93  109.41                0.87       243692
487                     0.52    0.86  107.60                0.87       240692
71                      0.40    0.91   81.91                0.87       266166
..                       ...     ...     ...                 ...          ...
175                     0.56    0.94  111.90                0.23       281132
263                     0.54    0.83  145.97                0.22       218390
15                      0.44    0.81   94.39                0.21       184253
338                     0.30    0.85  134.35                0.21       183603
176                     0.83    0.86  157.06                0.20       275521

[120 rows x 11 columns]
```

Data Visualization:

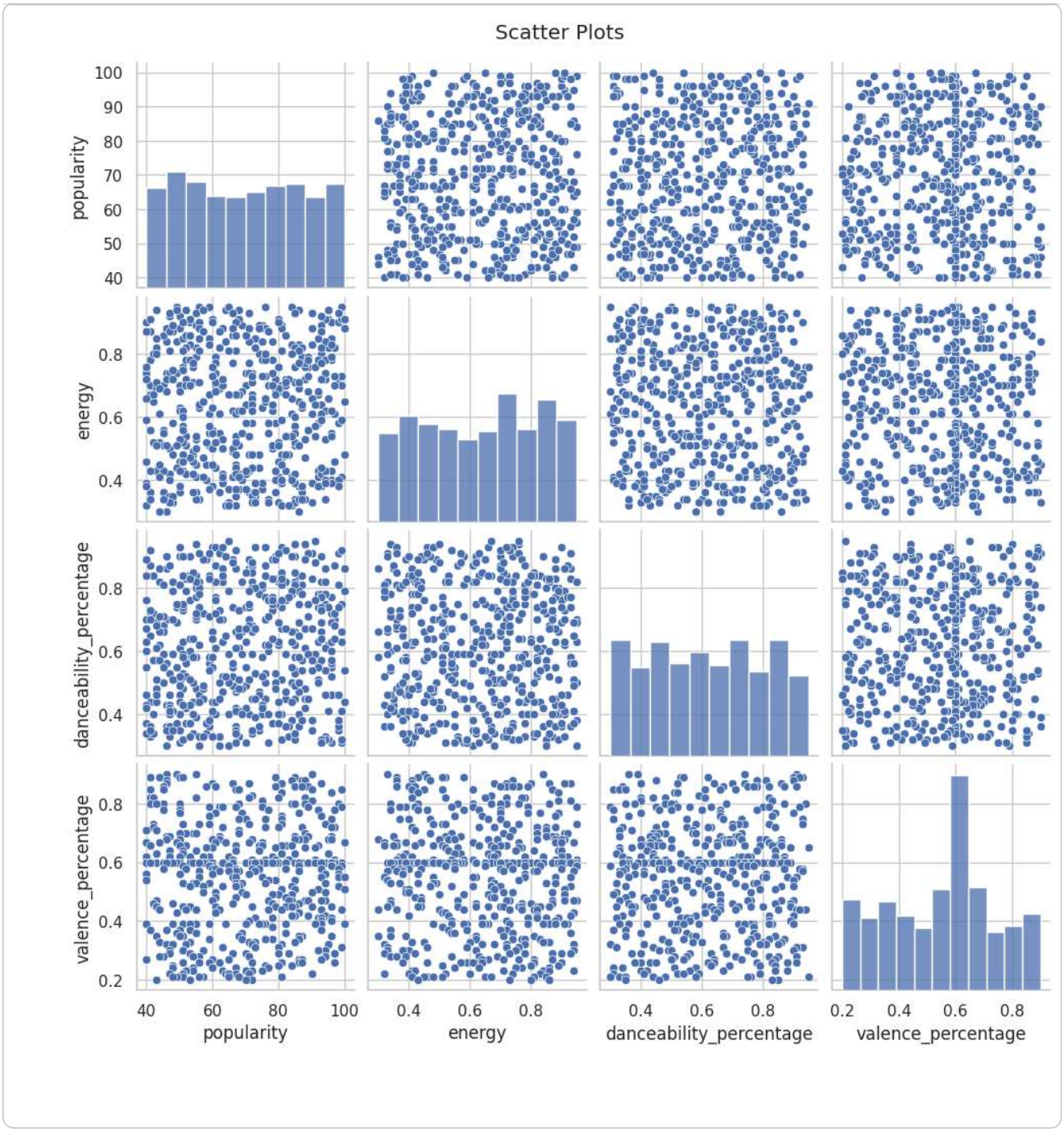
```
# Set the style of seaborn
sns.set(style="whitegrid")

# Visualization 1: Distribution of Streams
plt.figure(figsize=(12, 6))
sns.histplot(df['energy'].dropna(), kde=True, color='blue', bins=30)
plt.title('Distribution of Streams')
plt.xlabel('Streams')
plt.ylabel('Frequency')
plt.show()
```



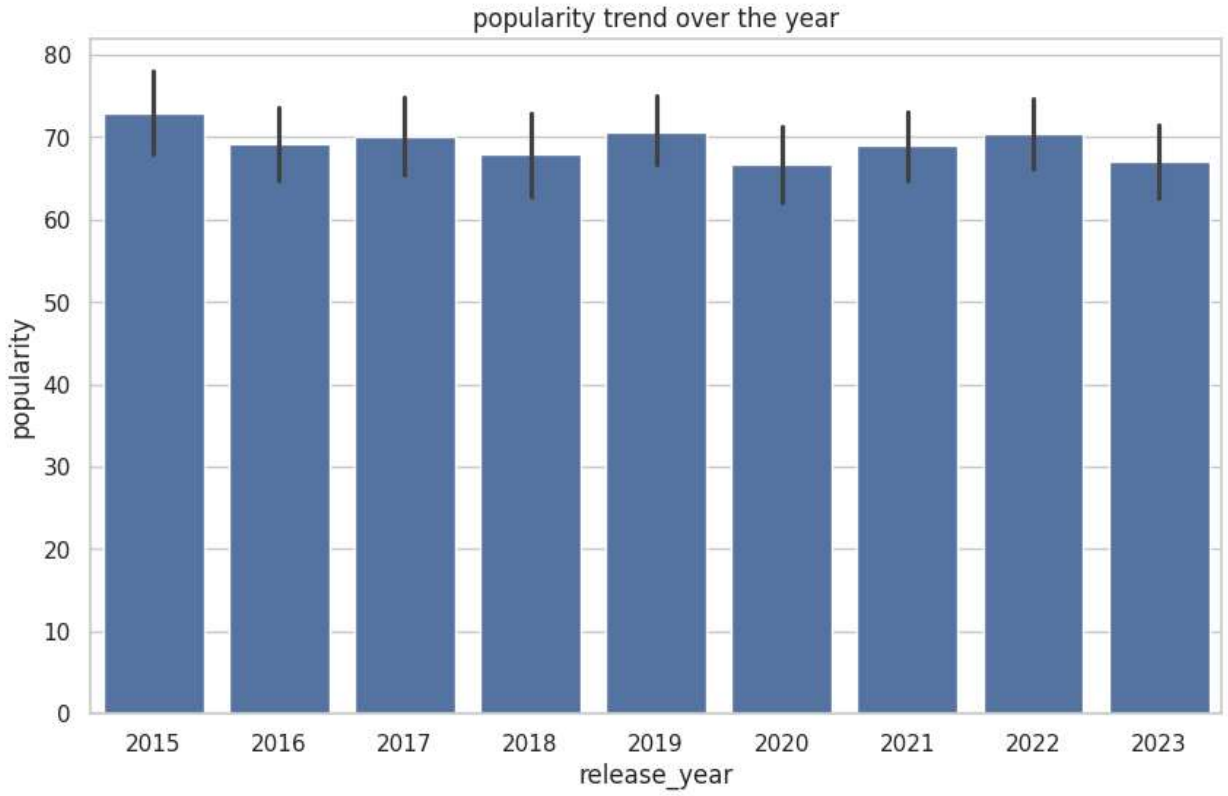

```
# Visualization 3: Box Plot
plt.figure(figsize=(16, 8))
sns.boxplot(x='genre', y='popularity', data=df)
plt.title('Box Plot of popularity by genre')
plt.show()
```



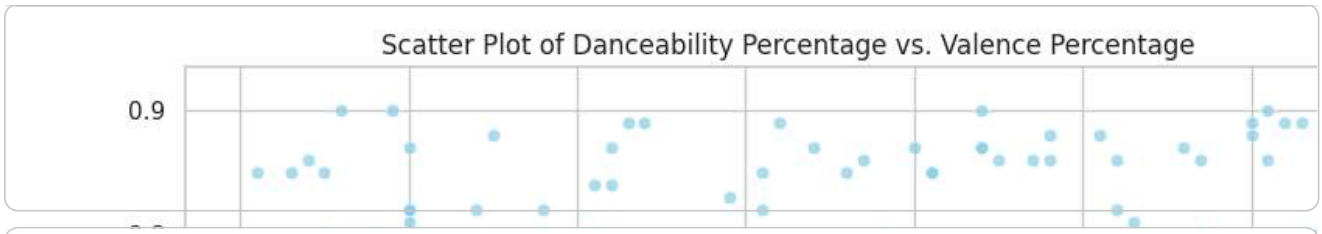

```
# Visualization 4: Scatter Plots
sns.pairplot(df[['popularity', 'energy', 'danceability_percentage', 'valence_percentage']])
plt.suptitle('Scatter Plots', y=1.02)
plt.show()
```



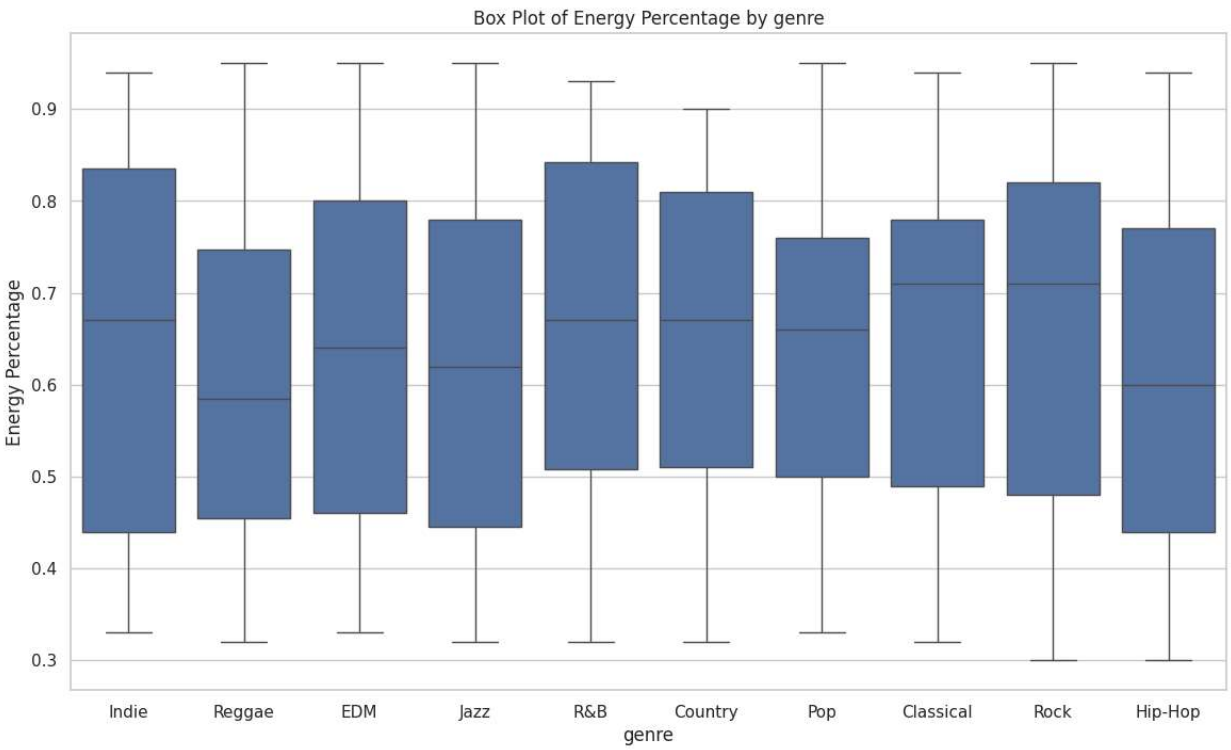

```
# Visualization: Bar Plot (popularity over release year)
plt.figure(figsize=(10, 6))
sns.barplot(x='release_year', y='popularity', data=df)
plt.title('popularity trend over the year')
plt.xlabel('release_year')
plt.ylabel('popularity')
plt.show()
```



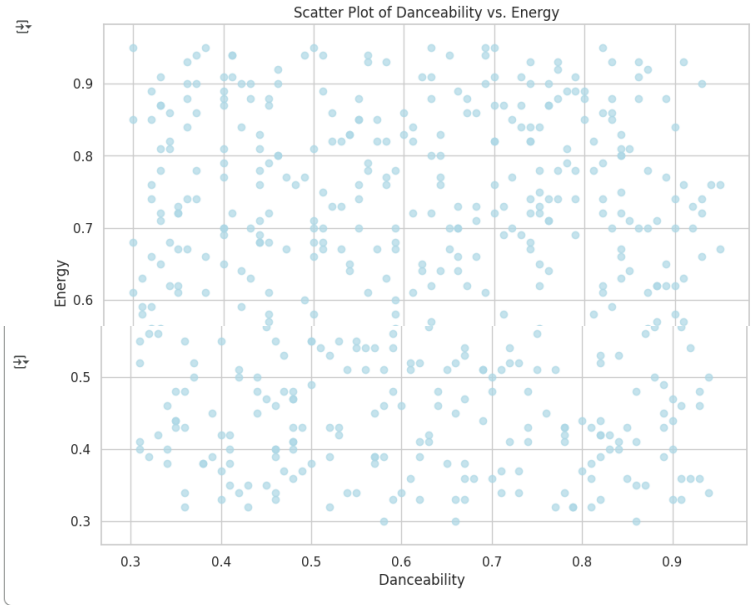

```
# Scatter Plot of Danceability Percentage vs. Valence Percentage:
plt.figure(figsize=(10, 8))
sns.scatterplot(x='danceability_percentage', y='valence_percentage', data=df, sizes=(20, 200), color='s
plt.title('Scatter Plot of Danceability Percentage vs. Valence Percentage')
plt.xlabel('Danceability Percentage')
plt.ylabel('Valence Percentage')
plt.show()
```




```
# Box Plot of Energy Percentage by genre:
plt.figure(figsize=(14, 8))
sns.boxplot(x='genre', y='energy', data=df)
plt.title('Box Plot of Energy Percentage by genre')
plt.xlabel('genre')
plt.ylabel('Energy Percentage')
plt.show()
```




```
# 2. Scatter Plot: Danceability vs. Energy Relationship
plt.figure(figsize=(10, 8))
plt.scatter(df['danceability'], df['energy'],  alpha=0.7,color='lightblue')
plt.title('Scatter Plot of Danceability vs. Energy')
plt.xlabel('Danceability ')
plt.ylabel('Energy')
plt.show()
```



Reporting & Insights

[476] ✓ 17s

```
from google.colab import files
uploaded=files.upload()
```

```
Choose Files spotify_2023_dataset.csv
• spotify_2023_dataset.csv(text/csv) - 33382 bytes, last modified: 10/31/2025 - 100% done
Saving spotify_2023_dataset.csv to spotify_2023_dataset (5).csv
```

[477] ✓ 0s

```
import pandas as pd
df=pd.read_csv('spotify_2023_dataset.csv')
```

```
summary=df.describe()
summary
```

	release_year	popularity	danceability	energy	tempo	valence	duration_ms
count	500.000000	500.000000	500.000000	500.000000	500.00000	500.000000	500.000000
mean	2019.238000	69.240000	0.614680	0.634320	119.54506	0.545220	224987.604000
std	2.585096	17.630544	0.185354	0.188501	23.06265	0.199396	43221.959501
min	2015.000000	40.000000	0.300000	0.300000	80.24000	0.200000	150850.000000
25%	2017.000000	53.000000	0.450000	0.460000	99.24750	0.380000	189548.500000
50%	2019.000000	69.000000	0.615000	0.650000	119.19500	0.550000	224314.500000
75%	2021.000000	85.000000	0.772500	0.800000	139.61500	0.700000	261593.250000
max	2023.000000	100.000000	0.950000	0.950000	159.77000	0.900000	299924.000000

Conclusion

The Spotify Trends 2023 Analysis provides valuable insights into the dynamic landscape of music preferences. Understanding genre trends, discovering new artists, and tracking playlist dynamics contribute to a comprehensive overview of the music streaming landscape on Spotify. The integration of a CNN model enhances the accuracy of predictions, delivering a personalized music experience to users.

The findings from this analysis can inform music labels, artists, and playlist curators to adapt their strategies and content to meet evolving user preferences. The project serves as a valuable resource for anyone interested in understanding the current state of music trends on the Spotify platform.