Exploratory Data Analysis of Top Spotify Songs in 2023

Step 1: Data Cleaning and Preprocessing

In [92]:
```
#import libraries and dataset

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

#ignore warnings
import warnings
warnings.filterwarnings('ignore')

df = pd.read_csv('spotify-2023.csv', encoding="latin-1")
```

In [93]:
```
#first look at the 5 first row of the dataset
df.head()
```

Out[93]:

	track_name	artist(s)_name	artist_count	released_year	released_month	released_day	in_spotify_playlists	in_spotify_charts	streams	in_apple_playlists	...	in_deezer_playlists	i
0	Seven (feat. Latto) (Explicit Ver.)	Latto, Jung Kook	2	2023	7	14	553	147	141381703.0	43	...	45	
1	LALA	Myke Towers	1	2023	3	23	1474	48	133716286.0	48	...	58	
2	vampire	Olivia Rodrigo	1	2023	6	30	1397	113	140003974.0	94	...	91	
3	Cruel Summer	Taylor Swift	1	2019	8	23	7858	100	800840817.0	116	...	125	
4	WHERE SHE GOES	Bad Bunny	1	2023	5	18	3133	50	303236322.0	84	...	87	

5 rows × 21 columns

In [94]:
```
#dataset information
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 953 entries, 0 to 952
Data columns (total 21 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   track_name            953 non-null    object
 1   artist(s)_name        953 non-null    object
 2   artist_count          953 non-null    int64
 3   released_year         953 non-null    int64
 4   released_month        953 non-null    int64
 5   released_day          953 non-null    int64
 6   in_spotify_playlists  953 non-null    int64
 7   in_spotify_charts     953 non-null    int64
 8   streams               952 non-null    float64
 9   in_apple_playlists    953 non-null    int64
 10  in_apple_charts       953 non-null    int64
 11  in_deezer_playlists   953 non-null    object
 12  in_deezer_charts      953 non-null    int64
 13  in_shazam_charts      903 non-null    object
 14  bpm                   953 non-null    int64
 15  key                   858 non-null    object
 16  mode                  953 non-null    object
 17  danceability_%        953 non-null    int64
 18  energy_%              953 non-null    int64
 19  acousticness_%        953 non-null    int64
 20  liveness_%            953 non-null    int64
dtypes: float64(1), int64(14), object(6)
memory usage: 156.5+ KB
```

In [95]:
```
#check for duplicates
df.nunique()
```

Out[95]:
```
track_name              943
artist(s)_name          645
artist_count              8
released_year            50
released_month           12
released_day             31
in_spotify_playlists    879
in_spotify_charts        82
streams                 948
in_apple_playlists      234
in_apple_charts         172
in_deezer_playlists     348
in_deezer_charts         34
in_shazam_charts        198
bpm                     124
key                      11
mode                      2
danceability_%           72
energy_%                 80
acousticness_%           98
liveness_%               68
dtype: int64
```

In [96]:
```
#missing values calculation
df.isnull().sum()
```

Out[96]:
```
track_name              0
artist(s)_name          0
artist_count            0
released_year           0
released_month          0
released_day            0
in_spotify_playlists    0
in_spotify_charts       0
streams                 1
in_apple_playlists      0
in_apple_charts         0
in_deezer_playlists     0
in_deezer_charts        0
in_shazam_charts       50
bpm                     0
key                    95
mode                    0
danceability_%          0
energy_%                0
acousticness_%          0
liveness_%              0
dtype: int64
```

In [97]:
```
#percentage of missing files
round((df.isnull().sum()/(len(df)))*100,2)
```

Out[97]:
```
track_name              0.00
artist(s)_name          0.00
artist_count            0.00
released_year           0.00
released_month          0.00
released_day            0.00
in_spotify_playlists    0.00
in_spotify_charts       0.00
streams                 0.10
in_apple_playlists      0.00
in_apple_charts         0.00
in_deezer_playlists     0.00
in_deezer_charts        0.00
in_shazam_charts        5.25
bpm                     0.00
key                     9.97
mode                    0.00
danceability_%          0.00
energy_%                0.00
acousticness_%          0.00
liveness_%              0.00
dtype: float64
```

In [98]:
```
#Now we deal with missing values

#Replacing missing values in the streams column with the median
df['streams'] = df['streams'].fillna(df['streams'].median())

#Replacing missing values in the key column with the mode
df['key'] = df['key'].fillna(df['key'].mode()[0])

#Cleaning the in_shazam_charts column to remove commas and converting it to integers
df['in_shazam_charts'] = df['in_shazam_charts'].astype(str)
df['in_shazam_charts'] = df['in_shazam_charts'].str.replace(',', '').astype(float)
df['in_shazam_charts'] = df['in_shazam_charts'].fillna(df['in_shazam_charts'].median()).astype(int)

#convert 'streams' column to int
df['streams'] = df['streams'].astype(int)

# Converting the 'in_deezer_playlists' column to integer
# First, we handle any non-numeric characters that might be present
df['in_deezer_playlists'] = df['in_deezer_playlists'].str.replace(',', '').astype(float)
df['in_deezer_playlists'] = df['in_deezer_playlists'].fillna(df['in_deezer_playlists'].median()).astype(int)

#check missing values again
df.isnull().sum()
```

Out[98]:
```
track_name              0
artist(s)_name          0
artist_count            0
released_year           0
released_month          0
released_day            0
in_spotify_playlists    0
in_spotify_charts       0
streams                 0
in_apple_playlists      0
in_apple_charts         0
in_deezer_playlists     0
in_deezer_charts        0
in_shazam_charts        0
bpm                     0
key                     0
mode                    0
danceability_%          0
energy_%                0
acousticness_%          0
liveness_%              0
dtype: int64
```

In [99]:
```
#chech datatypes again
df.dtypes
```

Out[99]:

```
track_name              object
artist(s)_name          object
artist_count             int64
released_year            int64
released_month           int64
released_day             int64
in_spotify_playlists     int64
in_spotify_charts        int64
streams                  int64
in_apple_playlists       int64
in_apple_charts          int64
in_deezer_playlists      int64
in_deezer_charts         int64
in_shazam_charts         int64
bpm                      int64
key                     object
mode                    object
danceability_%           int64
energy_%                 int64
acousticness_%           int64
liveness_%               int64
dtype: object
```

In [100…

```
# Renaming the columns 'track_name' to 'song' and 'artist(s)_name' to 'artist'
df.rename(columns={'track_name': 'song', 'artist(s)_name': 'artist'}, inplace=True)

df.head()
```

Out[100]:

	song	artist	artist_count	released_year	released_month	released_day	in_spotify_playlists	in_spotify_charts	streams	in_apple_playlists	...	in_deezer_playlists	in_deezer_ch
0	Seven (feat. Latto) (Explicit Ver.)	Latto, Jung Kook	2	2023	7	14	553	147	141381703	43	...	45	
1	LALA	Myke Towers	1	2023	3	23	1474	48	133716286	48	...	58	
2	vampire	Olivia Rodrigo	1	2023	6	30	1397	113	140003974	94	...	91	
3	Cruel Summer	Taylor Swift	1	2019	8	23	7858	100	800840817	116	...	125	
4	WHERE SHE GOES	Bad Bunny	1	2023	5	18	3133	50	303236322	84	...	87	

5 rows × 21 columns

In [119…

```
import datetime

# Function to determine the day of the week for a given date
def get_day_of_week(year, month, day):
    return datetime.date(year, month, day).strftime("%A")

# Applying the function to create a new column 'released_day_of_week'
df['released_day_of_week'] = df.apply(lambda row: get_day_of_week(row['released_year'], row['released_month'], row['released_day']), axis=1)
```

In [120…

```
#check if new column was added correctly
df.head()
```

Out[120]:

	song	artist	artist_count	released_year	released_month	released_day	in_spotify_playlists	in_spotify_charts	streams	in_apple_playlists	...	in_deezer_charts	in_shazam_cha
0	Seven (feat. Latto) (Explicit Ver.)	Latto, Jung Kook	2	2023	7	14	553	147	141381703	43	...	10	
1	LALA	Myke Towers	1	2023	3	23	1474	48	133716286	48	...	14	
2	vampire	Olivia Rodrigo	1	2023	6	30	1397	113	140003974	94	...	14	
3	Cruel Summer	Taylor Swift	1	2019	8	23	7858	100	800840817	116	...	12	
4	WHERE SHE GOES	Bad Bunny	1	2023	5	18	3133	50	303236322	84	...	15	

5 rows × 22 columns

Step 2: Descriptive Statistics

In [103…

```
#finding descriptive statistics, rounded to 2 decimals.
df.describe().round(2)
```

Out[103]:

	artist_count	released_year	released_month	released_day	in_spotify_playlists	in_spotify_charts	streams	in_apple_playlists	in_apple_charts	in_deezer_playlists	in_deezer_
count	953.00	953.00	953.00	953.00	953.00	953.00	9.530000e+02	953.00	953.00	953.00	
mean	1.56	2018.24	6.03	13.93	5200.12	12.01	5.139028e+08	67.81	51.91	385.19	
std	0.89	11.12	3.57	9.20	7897.61	19.58	5.666055e+08	86.44	50.63	1130.54	
min	1.00	1930.00	1.00	1.00	31.00	0.00	2.762000e+03	0.00	0.00	0.00	
25%	1.00	2020.00	3.00	6.00	875.00	0.00	1.417210e+08	13.00	7.00	13.00	
50%	1.00	2022.00	6.00	13.00	2224.00	3.00	2.905309e+08	34.00	38.00	44.00	
75%	2.00	2022.00	9.00	22.00	5542.00	16.00	6.738011e+08	88.00	87.00	164.00	
max	8.00	2023.00	12.00	31.00	52898.00	147.00	3.703895e+09	672.00	275.00	12367.00	

On average, songs featured about 1.56 artists; most were released around 2018, typically in June and around the middle of the month. These songs were quite popular, appearing on average in over 5200 Spotify playlists and 12 Spotify charts. They also achieved significant reach, with an average of around 513.9 million streams. Regarding other platforms, songs were found in an average of 67.81 Apple playlists, 51.91 Apple charts, and 385.19 Deezer playlists. The average tempo (BPM) of these songs was 122.54. These songs ' characteristics like danceability, energy, acousticness, and liveness varied, with average values of 66.97%, 64.28%, 27.06%, and 18.21% respectively, indicating a diverse range of song styles and moods.

In [104…

```
#finding the most frequent value (mode) for categorical columns

categorical_columns = ['artist', 'song', 'key', 'mode', 'released_day_of_week']
categorical_modes = df[categorical_columns].mode().iloc[0]

categorical_modes
```

Out[104]:

```
artist                       Taylor Swift
song                       About Damn Time
key                                     C#
mode                                 Major
released_day_of_week                Friday
Name: 0, dtype: object
```

Taylor Swift emerged as the most frequently occurring artist, indicating her significant popularity or presence in this dataset. The key of C# was the most common musical key among these songs, suggesting a preference or trend in popular music production during this period. Most songs were in the Major mode, often associated with a brighter, more upbeat sound. Interestingly, Friday was the most common day of the week for song releases, possibly reflecting strategic release timing to maximize listenership and chart performance. Lastly, the song titled "About Damn Time" appeared as the most frequent song title in the dataset, highlighting its prominence or repeated presence in the data collected.

Step 3: Data Visualization

In [114…

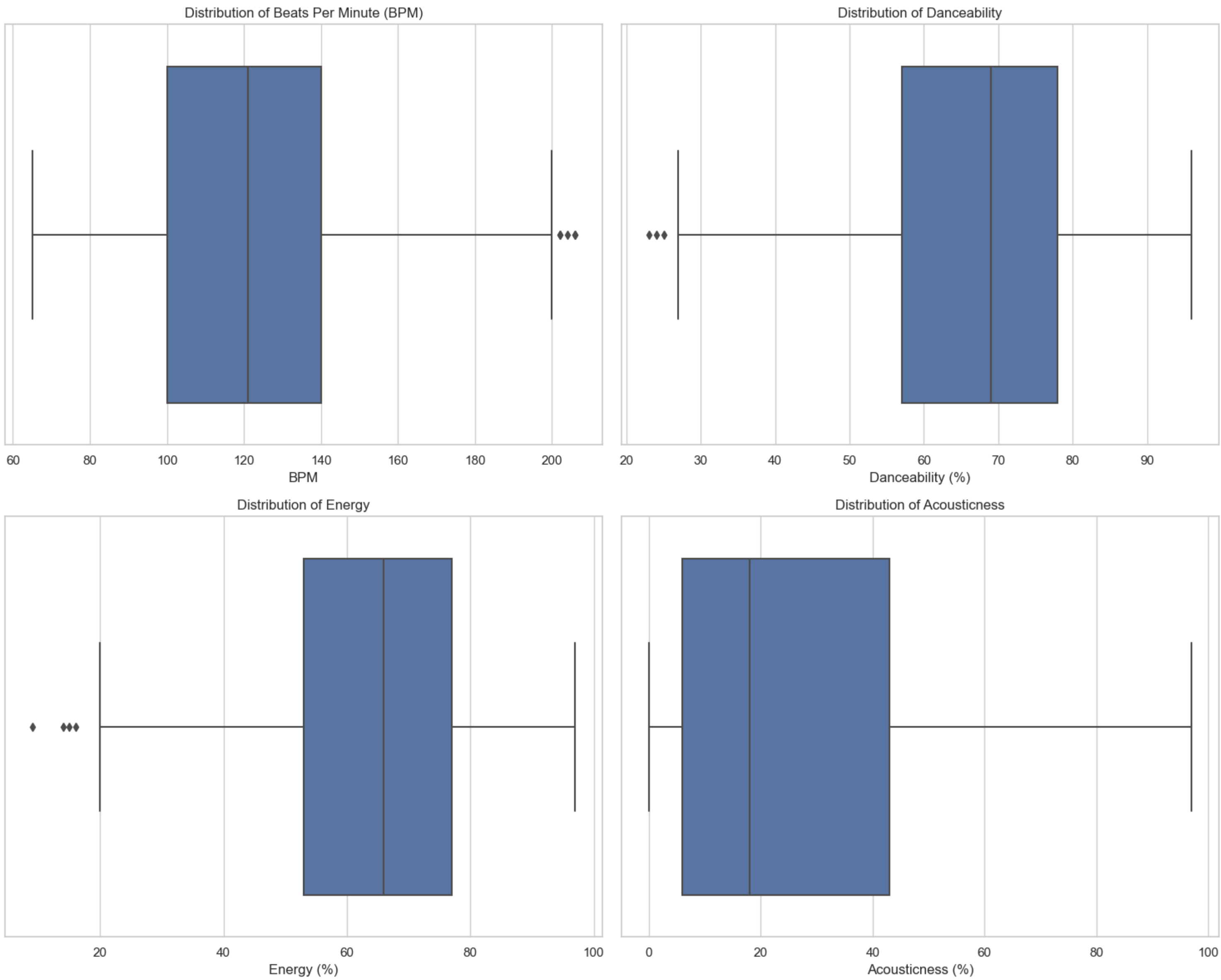
```
# Setting the aesthetic style of the plots
sns.set(style="whitegrid")

# Creating histograms for specified columns
fig, axes = plt.subplots(3, 2, figsize=(15, 15))

# top 20 artists
top_artists = df['artist'].value_counts().head(20)
sns.barplot(x=top_artists.values, y=top_artists.index, ax=axes[0, 0])
axes[0, 0].set_title('Top 20 Artists')
axes[0, 0].set_xlabel('Number of Songs')

# Histogram for 'released_year'
sns.histplot(df['released_year'], bins=30, kde=False, ax=axes[0, 1])
axes[0, 1].set_title('Distribution of Release Year')
axes[0, 1].set_xlabel('Release Year')

# Histogram for 'bpm'
sns.histplot(df['bpm'], bins=30, kde=False, ax=axes[1, 0])
axes[1, 0].set_title('Distribution of BPM')
axes[1, 0].set_xlabel('BPM')

# Count for 'key'
sns.countplot(x='key', data=df, ax=axes[1, 1])
axes[1, 1].set_title('Distribution of Key')
axes[1, 1].set_xlabel('Key')

# Count for 'released_day_of_week'
sns.countplot(x='released_day_of_week', data=df, ax=axes[2, 0])
axes[2, 0].set_title('Distribution of Release Day of Week')
axes[2, 0].set_xlabel('Day of Week')

# Adjusting layout to prevent overlap
plt.tight_layout()

# Hiding empty subplot
axes[2, 1].set_visible(False)

plt.show()
```

The bar chart of the top 20 artists reveals which artists are most dominant in the dataset, showing their prevalence in the music industry. The histogram of release years illustrates how song releases are distributed over time, providing a view into the popularity of songs from different years. The BPM histogram sheds light on the range of song tempos, indicating the typical beat speeds in these popular tracks. The count plot for musical keys reveals the most commonly used keys in these songs, offering a glimpse into the musical preferences in song production, and the count plot for the release day of the week shows which days are favored for releasing new music, hinting at strategic decisions in the music industry.

In [123…

```
# Creating boxplots for various features

# Setting up the figure for multiple boxplots
fig, axes = plt.subplots(nrows=2, ncols=2, figsize=(15, 12))

# BPM Distribution Boxplot
sns.boxplot(x=df['bpm'], ax=axes[0, 0])
axes[0, 0].set_title('Distribution of Beats Per Minute (BPM)')
axes[0, 0].set_xlabel('BPM')

# Danceability Distribution Boxplot
sns.boxplot(x=df['danceability_%'], ax=axes[0, 1])
axes[0, 1].set_title('Distribution of Danceability')
axes[0, 1].set_xlabel('Danceability (%)')

# Energy Distribution Boxplot
sns.boxplot(x=df['energy_%'], ax=axes[1, 0])
axes[1, 0].set_title('Distribution of Energy')
axes[1, 0].set_xlabel('Energy (%)')

# Acousticness Distribution Boxplot
sns.boxplot(x=df['acousticness_%'], ax=axes[1, 1])
axes[1, 1].set_title('Distribution of Acousticness')
```

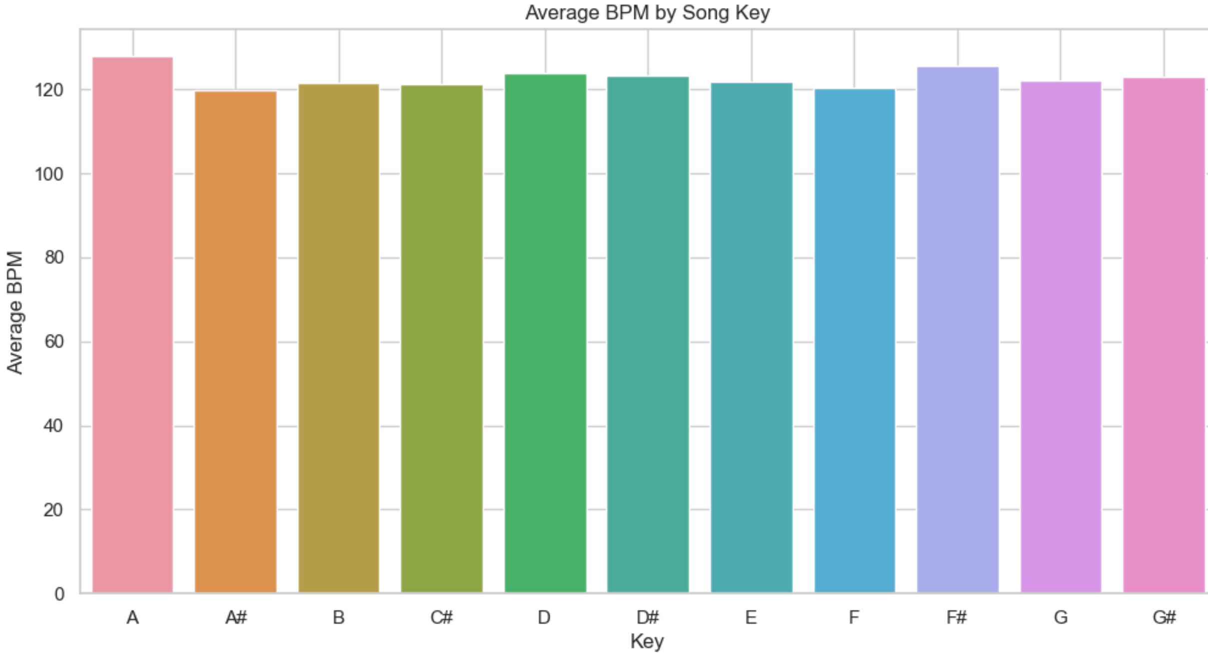
```python
axes[1, 1].set_xlabel('Acousticness (%)')

# Adjusting layout for better readability
plt.tight_layout()
plt.show()
```



The Beats Per Minute (BPM) distribution indicates a wide range of tempos in popular music, with a concentration in a specific range, suggesting a preference for certain tempos among listeners. The danceability measure shows that a large proportion of popular songs are quite danceable, highlighting a tendency towards rhythmically engaging music. In terms of energy, there is a varied distribution, indicating that both high-energy and more subdued tracks find their place in the top charts. Lastly, the acousticness distribution suggests a diversity in the use of acoustic elements, with some songs featuring high acousticness and others leaning towards more electronic sounds.

In [127…

```python
# Calculating the average BPM for each key
avg_bpm_by_key = df.groupby('key')['bpm'].mean()

# Creating a bar graph for Average BPM by Song Key
plt.figure(figsize=(12, 6))
sns.barplot(x=avg_bpm_by_key.index, y=avg_bpm_by_key.values)
plt.title('Average BPM by Song Key')
plt.xlabel('Key')
plt.ylabel('Average BPM')
plt.grid(True)
plt.show()
```



Each bar represents a different musical key, and the height of the bar indicates the average BPM of songs in that key. This visualization helps us understand how tempo (BPM) varies across different keys in popular music. Certain keys might be associated with faster or slower tempos, reflecting stylistic or genre tendencies.

Step 4: Correlation Analysis

In [128…

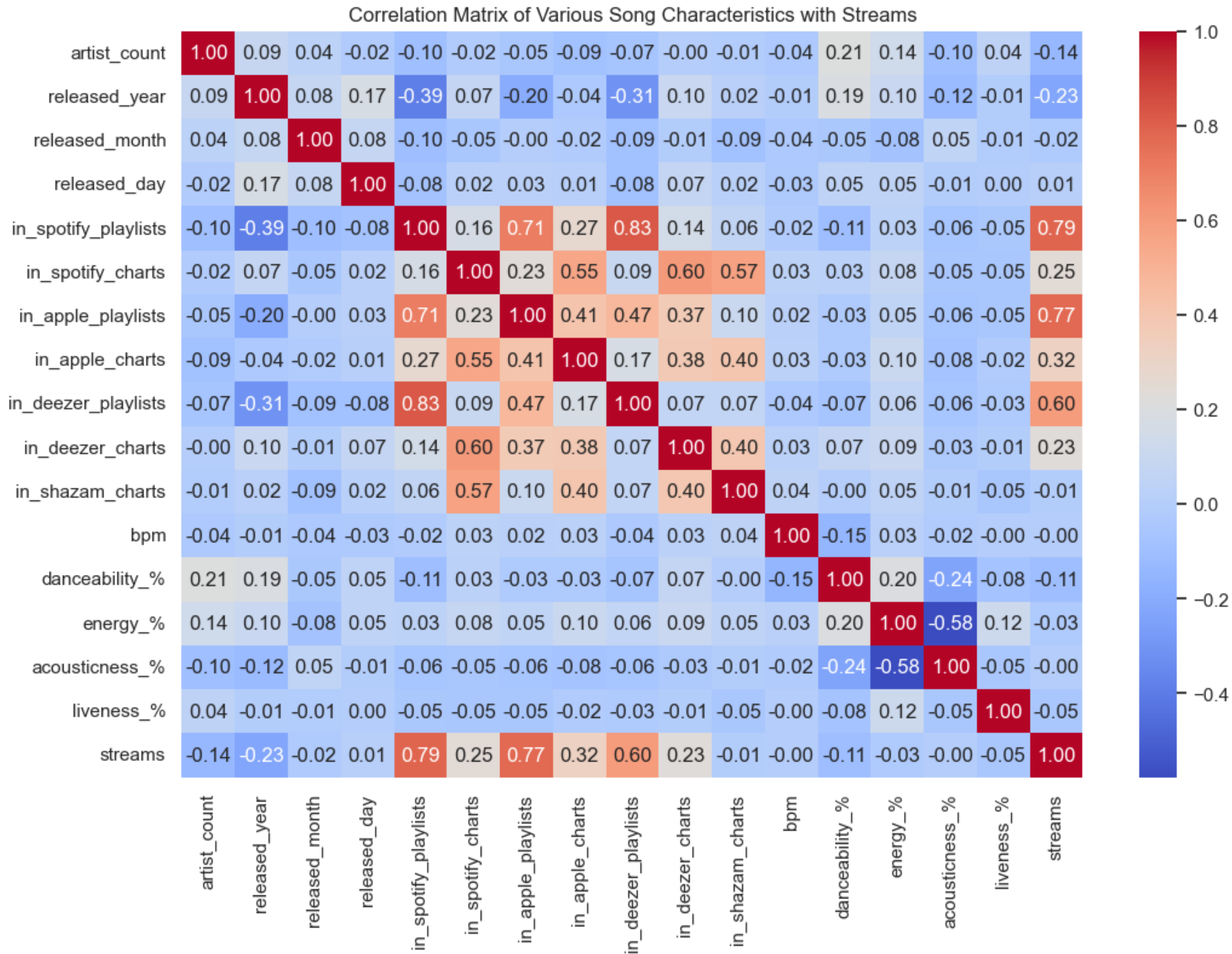
```
# Selecting relevant columns for correlation analysis
correlation_columns = [
    'artist_count', 'released_year', 'released_month', 'released_day',
    'in_spotify_playlists', 'in_spotify_charts', 'in_apple_playlists', 'in_apple_charts',
    'in_deezer_playlists', 'in_deezer_charts', 'in_shazam_charts',
    'bpm', 'danceability_%', 'energy_%', 'acousticness_%', 'liveness_%', 'streams'
]

# Calculating correlation matrix
correlation_matrix = df[correlation_columns].corr()

# Focusing on correlations with 'streams'
correlation_with_streams = correlation_matrix['streams'].sort_values(ascending=False)
correlation_with_streams
```

Out[128]:

```
streams                 1.000000
in_spotify_playlists    0.789844
in_apple_playlists      0.772103
in_deezer_playlists     0.598177
in_apple_charts         0.320456
in_spotify_charts       0.246007
in_deezer_charts        0.228739
released_day            0.011169
bpm                    -0.002252
acousticness_%         -0.004163
in_shazam_charts       -0.005330
released_month         -0.024325
energy_%               -0.026166
liveness_%             -0.048296
danceability_%         -0.105002
artist_count           -0.136166
released_year          -0.226689
Name: streams, dtype: float64
```

In [129…

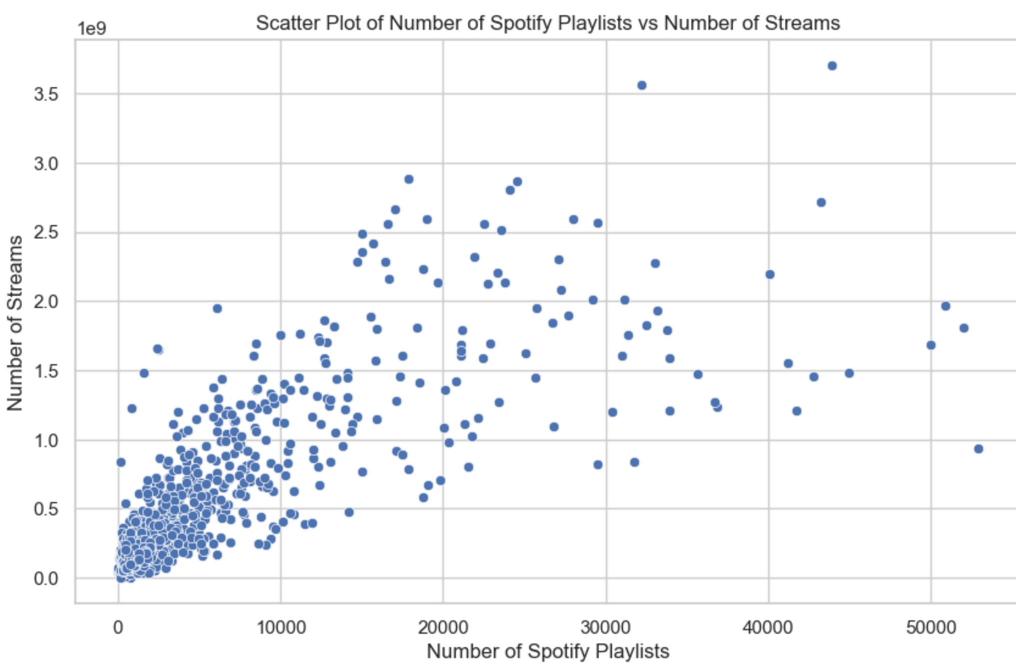
```
# Plotting the correlation matrix as a heatmap
plt.figure(figsize=(12, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.title("Correlation Matrix of Various Song Characteristics with Streams")
plt.show()
```



The strongest correlation with the number of streams on Spotify, as indicated by the correlation analysis, is with in_spotify_playlists, having a correlation coefficient of approximately 0.790.79. This suggests a strong positive relationship: songs featured in more Spotify playlists tend to have more streams. This could be due to greater visibility and accessibility to listeners through playlist inclusion, thereby driving up the stream counts.

In [130…

```
# Scatter plot of in_spotify_playlists vs Streams to visualize the strong correlation
plt.figure(figsize=(10, 6))
sns.scatterplot(x=df['in_spotify_playlists'], y=df['streams'])
plt.title("Scatter Plot of Number of Spotify Playlists vs Number of Streams")
plt.xlabel("Number of Spotify Playlists")
plt.ylabel("Number of Streams")
plt.show()
```



The scatter plot shows the relationship between the number of Spotify playlists a song is featured in and its number of streams. This visualization clearly illustrates the strong positive correlation (0.79) identified earlier. As the number of Spotify playlists a song is included in increases, there's a noticeable trend of increased streams. This supports the finding that being featured in more playlists is associated with higher streaming numbers, likely due to increased visibility and accessibility to listeners.