Student: Jennifer Lee

Project overview:

Utilizing a dataset: **[Most Streamed Spotify Songs 2023]** to answer and analyze Business Questions

Needed criteria:

- Problem Identification + Data Collection
- Derive Analytical Questions
- Collecting relevant data
- Coming up with analytical Questions related to dataset
- Collecting relevant data from dataset
- Data loading, cleaning, preprocessing, aggregating.
- Using 4 different types of visualizations
  - Histogram, Line charts, Bar Charts, Scatter Plots, etc.
- Did not implement regression modeling for this project. Project report link: https://docs.google.com/document/d/1iDREQXqZjm5X1nTkBqdq8cpvG8ZWx7I8c8givdY0xtA/edit

Overarching Business Problem: Analyze trends and patterns in Spotify Data to understand user preferences and optimize Spotify's machine learning music recommendations.

General Analytical Questions:

- What are the top 10 most popular songs of 2023?
- Who are the top 10 artists of 2023?
- What is the typical tempo range of songs in the dataset?
- Streams Vs. Year Released : Are there any trends?
- Is there any trends throughout the months with streaming?

```
#Importing needed libraries to start project.
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

```
!pip install chardet
```

```
Requirement already satisfied: chardet in /usr/local/lib/python3.10/dist-packages (5.2.6
```

```
import chardet
#errors with encoding. using chardet to fix.
```

```
with open('/content/spotify-2023.csv', 'rb') as f:
    encoding = chardet.detect(f.read())['encoding']
```

```
df_spotify = pd.read_csv('/content/spotify-2023.csv', encoding=encoding)
```

```
df_spotify.head(10)
#Displaying first top 10 songs.
```

	track_name	artist(s)_name	artist_count	released_year	released_month	released_day
0	Seven (feat. Latto) (Explicit Ver.)	Latto, Jung Kook	2	2023	7	14
1	LALA	Myke Towers	1	2023	3	23
2	vampire	Olivia Rodrigo	1	2023	6	30
3	Cruel Summer	Taylor Swift	1	2019	8	23
4	WHERE SHE GOES	Bad Bunny	1	2023	5	18
5	Sprinter	Dave, Central Cee	2	2023	6	1
6	Ella Baila Sola	Eslabon Armado, Peso Pluma	2	2023	3	16
7	Columbia	Quevedo	1	2023	7	7
8	fukumean	Gunna	1	2023	5	15
9	La Bebe - Remix	Peso Pluma, Yng Lvcas	2	2023	3	17

10 rows × 24 columns

Data Inspection + Cleaning

```
df_spotify.info()
#Data seems to be missing at in_shazam_charts (903)
#Data also seems to be missing at key (858)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 953 entries, 0 to 952
Data columns (total 24 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               953 non-null    object
 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  valence_%             953 non-null    int64
 19  energy_%              953 non-null    int64
 20  acousticness_%        953 non-null    int64
 21  instrumentalness_%    953 non-null    int64
 22  liveness_%            953 non-null    int64
 23  speechiness_%         953 non-null    int64
dtypes: int64(17), object(7)
memory usage: 178.8+ KB
```

```
df_spotify.isnull().sum()
#Missing values:
#In_shazam_charts: 50 missing
#Key: 95 missing.
```

```
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        50
bpm                      0
key                     95
mode                     0
danceability_%           0
valence_%                0
energy_%                 0
acousticness_%           0
instrumentalness_%       0
liveness_%               0
speechiness_%            0
dtype: int64
```

```
df_spotify_cleaned = df_spotify.dropna()
```

```
#A long string is found in row 576. replaced with actual value on spotify.
df_spotify.at[576, 'streams'] = 245472912
```

```
df_spotify_cleaned.info()
#Data is cleaned.
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 817 entries, 0 to 952
Data columns (total 24 columns):
 #   Column                Non-Null Count  Dtype 
---  ------                --------------  ----- 
 0   track_name            817 non-null    object
 1   artist(s)_name        817 non-null    object
 2   artist_count          817 non-null    int64 
 3   released_year         817 non-null    int64 
 4   released_month        817 non-null    int64 
 5   released_day          817 non-null    int64 
 6   in_spotify_playlists  817 non-null    int64 
 7   in_spotify_charts     817 non-null    int64 
 8   streams               817 non-null    object
 9   in_apple_playlists    817 non-null    int64 
 10  in_apple_charts       817 non-null    int64 
 11  in_deezer_playlists   817 non-null    object
 12  in_deezer_charts      817 non-null    int64 
 13  in_shazam_charts      817 non-null    object
 14  bpm                   817 non-null    int64 
 15  key                   817 non-null    object
 16  mode                  817 non-null    object
 17  danceability_%        817 non-null    int64 
 18  valence_%             817 non-null    int64 
 19  energy_%              817 non-null    int64 
 20  acousticness_%        817 non-null    int64 
 21  instrumentalness_%    817 non-null    int64 
 22  liveness_%            817 non-null    int64 
 23  speechiness_%         817 non-null    int64 
dtypes: int64(17), object(7)
memory usage: 159.6+ KB
```

```
print(df_spotify_cleaned['streams'].dtype)
#change streams from object to numeric.
```

```
object
```

```
df_spotify_cleaned = df_spotify_cleaned.reset_index(drop=True)
```

```
# Drop the rows containing the non-numeric value
df_spotify_cleaned = df_spotify_cleaned.drop(477)
df_spotify_cleaned = df_spotify_cleaned.drop(478)
```

```
df_spotify_cleaned['streams'] = pd.to_numeric(df_spotify_cleaned['streams'])
#Dropped 477 and 478 to be able to analyze.
```

```
df_spotify_cleaned.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 815 entries, 0 to 816
Data columns (total 24 columns):
 #   Column                Non-Null Count  Dtype 
---  ------                --------------  ----- 
 0   track_name            815 non-null    object
 1   artist(s)_name        815 non-null    object
 2   artist_count          815 non-null    int64 
 3   released_year         815 non-null    int64 
 4   released_month        815 non-null    int64 
 5   released_day          815 non-null    int64 
 6   in_spotify_playlists  815 non-null    int64 
 7   in_spotify_charts     815 non-null    int64 
 8   streams               815 non-null    int64 
 9   in_apple_playlists    815 non-null    int64 
 10  in_apple_charts       815 non-null    int64 
 11  in_deezer_playlists   815 non-null    object
 12  in_deezer_charts      815 non-null    int64 
 13  in_shazam_charts      815 non-null    object
 14  bpm                   815 non-null    int64 
 15  key                   815 non-null    object
 16  mode                  815 non-null    object
 17  danceability_%        815 non-null    int64 
 18  valence_%             815 non-null    int64 
 19  energy_%              815 non-null    int64 
 20  acousticness_%        815 non-null    int64 
 21  instrumentalness_%    815 non-null    int64 
 22  liveness_%            815 non-null    int64 
 23  speechiness_%         815 non-null    int64 
dtypes: int64(18), object(6)
memory usage: 159.2+ KB
```

Preprocessing: Encoding and Feature Scaling

```
df_spotify_cleaned.columns
#Calling the columns for easier references.
```

```
Index(['track_name', 'artist(s)_name', '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_charts', 'in_shazam_charts', 'bpm',
       'key', 'mode', 'danceability_%', 'valence_%', 'energy_%',
       'acousticness_%', 'instrumentalness_%', 'liveness_%', 'speechiness_%'],
      dtype='object')
```

```
from sklearn.preprocessing import LabelEncoder, StandardScaler
```

```
artist_label_encoder = LabelEncoder()


#encoding categorical spaces.
if 'artist' in df_spotify_cleaned.columns:
    # Encode the 'artist' category into numerical values
    df_spotify_cleaned['artist_encoded'] = artist_label_encoder.fit_transform(df_spotify_cle

    # Drop the original categorical column
    df_spotify_cleaned.drop('artist', axis=1, inplace=True)


#encoding mode and key as well
category_mode_mapping = {'Major': 0, 'Minor': 1}

df_spotify_cleaned['mode_encoded'] = df_spotify_cleaned['mode'].map(category_mode_mapping)







# Initialize StandardScaler
scaler = StandardScaler()

# Define the numerical features to be scaled
numerical_features = ['danceability_%', 'energy_%',
                      'speechiness_%', 'acousticness_%', 'instrumentalness_%',
                      'liveness_%', 'valence_%','artist_count','released_day',
                      'in_apple_playlists', 'in_apple_charts', 'in_deezer_playlists',
                      'in_deezer_charts','in_shazam_charts']

# Replace commas with an empty string and convert to numeric
df_spotify_cleaned[numerical_features] = df_spotify_cleaned[numerical_features].replace(',',
df_spotify_cleaned[numerical_features] = df_spotify_cleaned[numerical_features].astype(float

# Scale the numerical features
df_spotify_cleaned[numerical_features] = scaler.fit_transform(df_spotify_cleaned[numerical_f

# Display the preprocessed dataset
df_spotify_cleaned.head()
```

	track_name	artist(s)_name	artist_count	released_year	released_month	released_day
0	Seven (feat. Latto) (Explicit Ver.)	Latto, Jung Kook	0.492818	2023	7	0.02934
1	LALA	Myke Towers	-0.648224	2023	3	0.99874
2	vampire	Olivia Rodrigo	-0.648224	2023	6	1.75272
3	Cruel Summer	Taylor Swift	-0.648224	2019	8	0.99874
4	WHERE SHE GOES	Bad Bunny	-0.648224	2023	5	0.46018

5 rows × 25 columns

Summarizations and Aggregation.

```
#Summary for cleaned and encoded dataset
df_spotify_cleaned.describe()
```

	artist_count	released_year	released_month	released_day	in_spotify_playlists
count	8.150000e+02	815.00000	815.000000	8.150000e+02	815.000000
mean	4.141200e-17	2018.51411	6.022086	-3.487326e-17	4850.876074
std	1.000614e+00	10.70819	3.571936	1.000614e+00	7750.212435
min	-6.482242e-01	1930.00000	1.000000	-1.370913e+00	31.000000
25%	-6.482242e-01	2021.00000	3.000000	-9.400663e-01	829.000000
50%	-6.482242e-01	2022.00000	5.000000	-7.837190e-02	2035.000000
75%	4.928184e-01	2022.00000	9.000000	8.910343e-01	4882.500000
max	7.339074e+00	2023.00000	12.000000	1.860440e+00	52898.000000

8 rows × 21 columns

```
#frequency distribution
df_spotify_cleaned.value_counts()
```

```
track_name                                          artist(s)_name           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_charts  in_shazam_charts  bpm        key  mode   danceability_%  valence_%  
energy_%   acousticness_%  instrumentalness_%  liveness_%  speechiness_%  
mode_encoded  
'Till I Collapse                             Eminem, Nate Dogg         0.492818   
2002           5                1.321881     22923                0   
1695712020   0.239406           -0.071262        1.850776           -0.269741   
-0.375413          1.717578  C#   Major  -0.844234       -1.743087   1.282639   
-0.759170       -0.191624           -0.750010    0.925382      0             1  
Rauw Alejandro: Bzrp Music Sessions, Vol. 56  Rauw Alejandro, Bizarrap   0.492818   
2023           6               0.783322     871                32   
66902503    -0.468649           0.190990        -0.293918            0.471081   
0.197463          0.192133  B    Major   0.721994        0.331361   0.041056   
-0.641415       -0.191624            0.578652   -0.542097      0             1  
Question...?                                 Taylor Swift             -0.648224   
2022           10               0.783322     1608               0   
223064273   -0.669042           -0.938714        -0.311193           -0.454946   
-0.375413         -0.481900  G    Major   0.517703       -1.700751  -0.890131   
-0.248899       -0.191624            0.873910    0.631886      0             1  
Quevedo: Bzrp Music Sessions, Vol. 52        Bizarrap, Quevedo         0.492818   
2022           7               -0.832354     8506               45   
1356565093   0.453159           0.312030        -0.179903            2.137930   
0.770339          0.192133  D    Major  -0.367556        0.162018   0.848085   
-0.994680        0.150362            0.357208   -0.639929      0             1  
Quï¿½ï¿½ Ago                                  Yuridia, Angela Aguilar    0.492818   
2022           10              0.675611     660                15   
236857112   -0.548806           0.190990        -0.306011            0.471081   
-0.036895         -0.872131  B    Major   0.381510        1.559096  -0.455577   
1.164160       -0.191624           -0.971454   -0.835592      0             1  


..  
Hati-Hati di Jalan                           Tulus                    -0.648224   
2022           3               -1.155490     200                2   
202677468   -0.642323           -0.918540       -0.321558           -0.454946   
-0.375413         -1.794493  F#   Major  -0.231362        1.051068  -1.262606   
1.713684        0.834332           -0.454752   -0.639929      0             1  
Heart To Heart                               Mac DeMarco              -0.648224   
2019           5               -0.401507     1640               0   
244658767   -0.441930           -0.454555       -0.296509           -0.269741   
-0.368903          0.972594  G#   Minor   1.539157        0.543039  -3.124980   
1.595929        3.798203           -0.528567   -0.052937      1             1  
Heartless                                    Kanye West               -0.648224   
2008           1               -1.370913     17504              34   
887906111    0.039013           -0.212475        0.814275           -0.454946   
-0.362393         -1.226885  A#   Minor   0.790091        0.627711   0.041056   
-0.837673       -0.191624            0.504837    0.338390      1             1  
Heather                                      Conan Gray               -0.648224   
2020           3               0.675611     6170               7   
1301799902   0.292844           -0.979060       -0.122032           -0.454946   
-0.362393         -1.084983  F    Major  -1.389009       -1.108052  -1.324685   
1.399670       -0.191624            1.021539   -0.737760      0             1  
ýýýýýýýýýýýýýýýýýýýýý                        Fujii Kaze               -0.648224   
2020           5               0.675611     685                14   
403097450   -0.482009           0.897055        -0.313784           -0.454946   
-0.225684          1.256397  F#   Minor  -0.503750        0.035011   0.723927   
-0.366654       -0.191624            0.061950   -0.542097      1             1  
```

```
#Aggregating numerical columns
#Aggregating Streams
total_streams = df_spotify_cleaned['streams'].sum()
average_streams_per_track = df_spotify_cleaned['streams'].mean()
max_streams = df_spotify_cleaned['streams'].max()

total_streams
```

```
382125429011
```

```
average_streams_per_track
```

```
468865557.06871164
```

```
max_streams #highest streamed track?
```

```
3562543890
```

```
#aggregated bpm
bpm_summary = df_spotify_cleaned['bpm'].describe()
bpm_summary
```

```
count    8.150000e+02
mean    -2.005213e-16
std      1.000614e+00
min     -2.042821e+00
25%     -8.366551e-01
50%     -9.167034e-02
75%      6.710521e-01
max      2.959219e+00
Name: bpm, dtype: float64
```

```
#Aggregating categorical columns (mode, key)
mode_counts = df_spotify_cleaned['mode'].value_counts()
mode_counts
```

```
mode
Major    451
Minor    364
Name: count, dtype: int64
```

```
key_counts = df_spotify_cleaned['key'].value_counts()
key_counts
```

```
key
C#    115
G      91
F      87
G#     85
D      78
```

```
B      76
A      70
F#     69
E      59
A#     55
D#     30
Name: count, dtype: int64
```

```
songs_per_year = df_spotify_cleaned.groupby('released_year').size().reset_index(name='song_c
songs_per_year
#Aggregated song count
```

	released_year	song_count
0	1930	1
1	1942	1
2	1946	1
3	1950	1
4	1957	1
5	1958	2
6	1959	2
7	1963	3
8	1970	1
9	1971	1
10	1973	1
11	1975	1
12	1979	1
13	1982	2
14	1983	1
15	1984	2
16	1985	2
17	1986	2
18	1987	1
19	1991	2
20	1992	1
21	1995	1
22	1996	1
23	1997	1
24	1998	1
25	1999	5
26	2000	3
27	2002	6
28	2003	2
29	2004	3

30 2005 1

Answering the analysis goals.

33 2011 9

Top 10 most streamed songs of 2023

```
top_10_songs = df_spotify_cleaned.sort_values(by='streams', ascending=False).head(10)
#finding the top 10 songs
```

38 2016 17

```
top_10_songs
```

	track_name	artist(s)_name	artist_count	released_year	released_month	released_c
151	Shape of You	Ed Sheeran	-0.648224	2017	1	-0.8323
36	Sunflower - Spider-Man: Into the Spider-Verse	Post Malone, Swae Lee	0.492818	2018	10	-0.5092
137	One Dance	Drake, WizKid, Kyla	1.633861	2016	4	-1.0477
71	STAY (with Justin Bieber)	Justin Bieber, The Kid Laroi	0.492818	2021	7	-0.5092
122	Believer	Imagine Dragons	-0.648224	2017	1	1.8604
611	Closer	The Chainsmokers, Halsey	0.492818	2016	5	1.8604
41	Starboy	The Weeknd, Daft Punk	0.492818	2016	9	0.7833
120	Perfect	Ed Sheeran	-0.648224	2017	1	-1.3709
583	Seï¿½ï¿½o	Shawn Mendes, Camila Cabello	0.492818	2019	6	0.5678
283	Say You Won't Let Go	James Arthur	-0.648224	2016	9	-0.5092

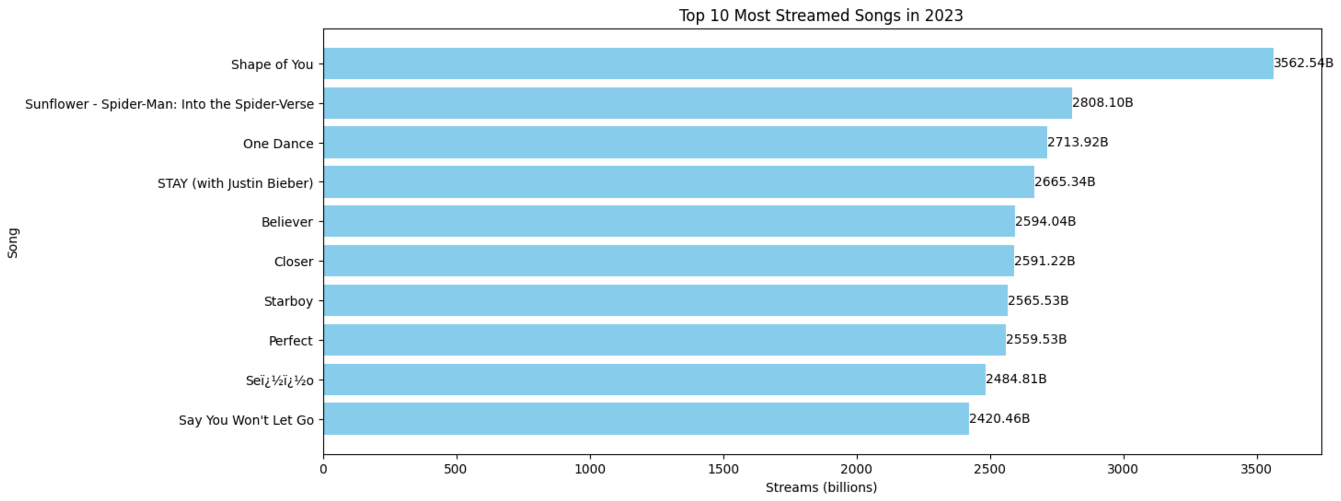
10 rows × 25 columns

```
# Convert streams to billions
streams_billions = top_10_songs['streams'] / 1_000_000

# Plotting the data
plt.figure(figsize=(14, 6))
plt.barh(top_10_songs['track_name'], streams_billions, color='skyblue')
plt.xlabel('Streams (billions)')
plt.ylabel('Song')
plt.title('Top 10 Most Streamed Songs in 2023')
plt.gca().invert_yaxis()  # Invert y-axis to display the song with the highest streams at tl

# Add value labels to the bars
for i, v in enumerate(streams_billions):
    plt.text(v + 0.1, i, f'{v:.2f}B', va='center')

plt.show()
```



Top 10 most streamed songs of 2023:

1. Shape of you by Ed Sheeran (3.6B Streams)
2. Sunflower- Spider-man: into the Spider-Verse by Post Malone
3. One Dance by Drake, Wiz Kid, Kyla
4. Stay (With Justin Beiber) by Justin Beiber and The Kid Laroi
5. Believer by Imagine Dragons
6. Closer by The Chainsmokers, Halsey
7. Starboy by The Weeknd, Daft Punk

8. Perfect by Ed Sheeran
9. Senorita by Shawn Mendes, Camila Cabello
10. Say you won't let go by James Arthur

Top 10 most popular artists of 2023: Bar graph

```
artist_streams = df_spotify_cleaned.groupby('artist(s)_name')['streams'].sum()
```

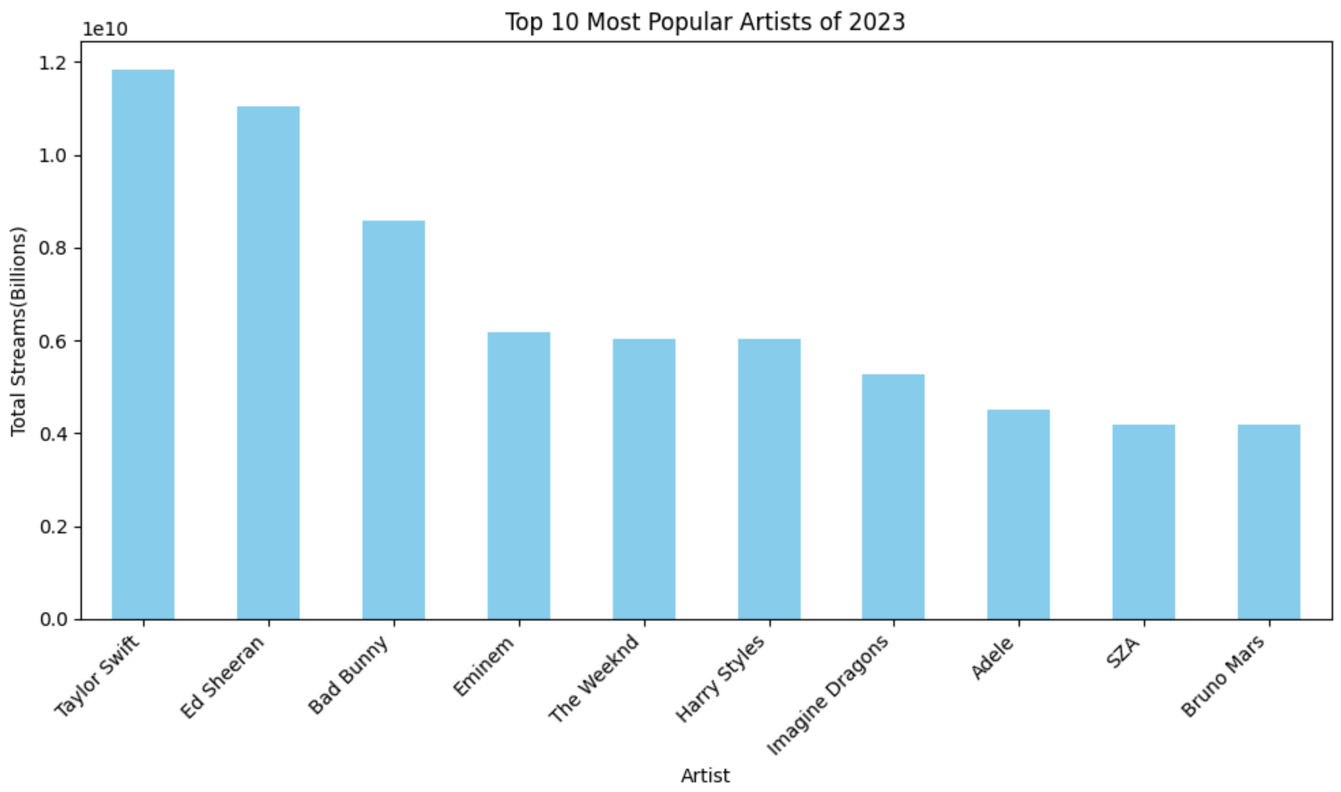
```
top_10_artists = artist_streams.sort_values(ascending=False).head(10)
```

```
top_10_artists
```

```
artist(s)_name
Taylor Swift       11851151082
Ed Sheeran         11051252012
Bad Bunny           8582384095
Eminem              6183805596
The Weeknd          6038640754
Harry Styles        6033490512
Imagine Dragons     5272484650
Adele               4508746590
SZA                 4197341485
Bruno Mars          4185733280
Name: streams, dtype: int64
```

```
top_10_artists = artist_streams.sort_values(ascending=False).head(10)

# Create a bar chart
plt.figure(figsize=(10, 6))
top_10_artists.plot(kind='bar', color='skyblue')
plt.title('Top 10 Most Popular Artists of 2023')
plt.xlabel('Artist')
plt.ylabel('Total Streams(Billions)')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()
```

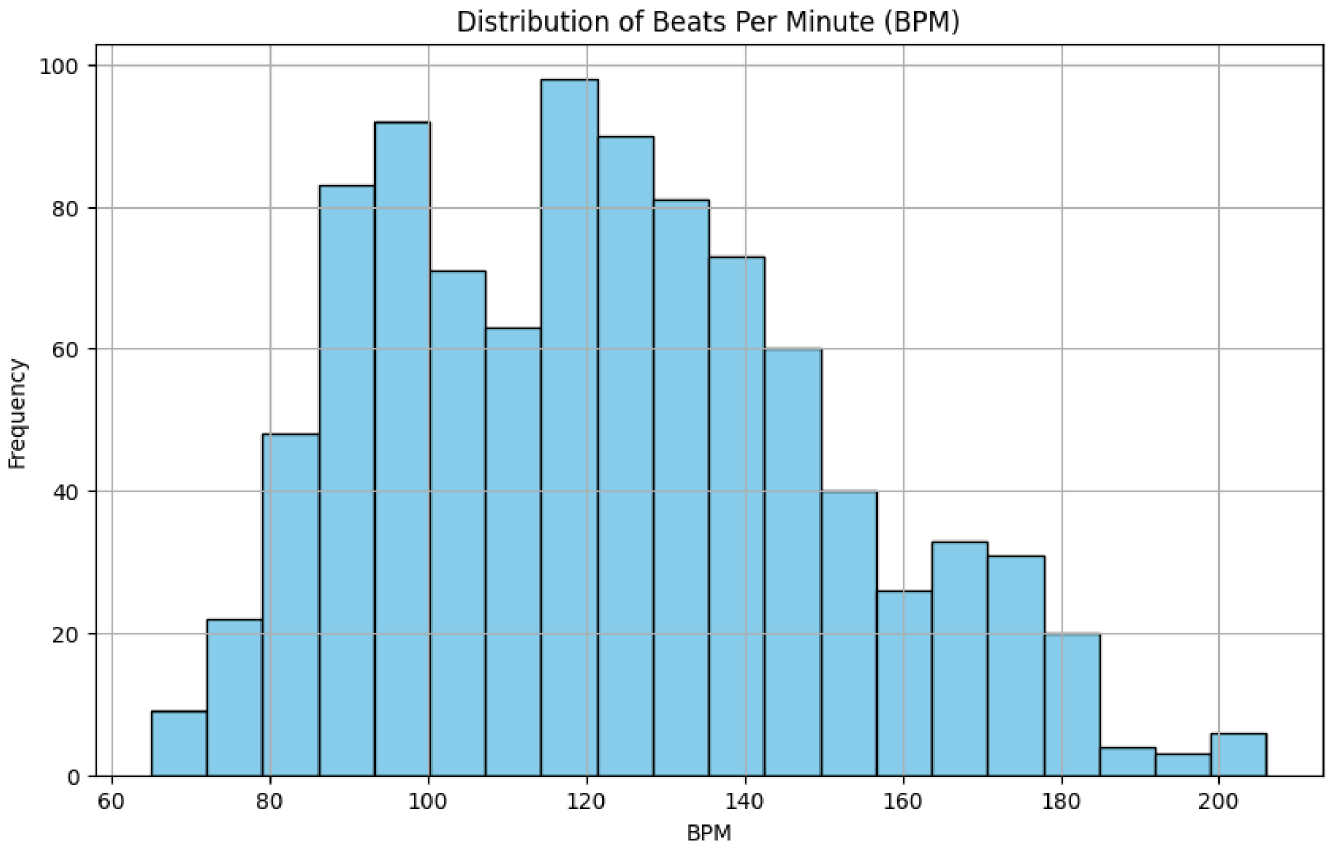


Top 10 Artists of 2023

1. Taylor Swift
2. Ed Sheeran
3. Bad Bunny
4. Eminem
5. The Weeknd
6. Harry Styles
7. Imagine Dragons
8. Adele
9. SZA
10. Bruno Mars

What is the typical tempo range of songs in the dataset?: Histogram

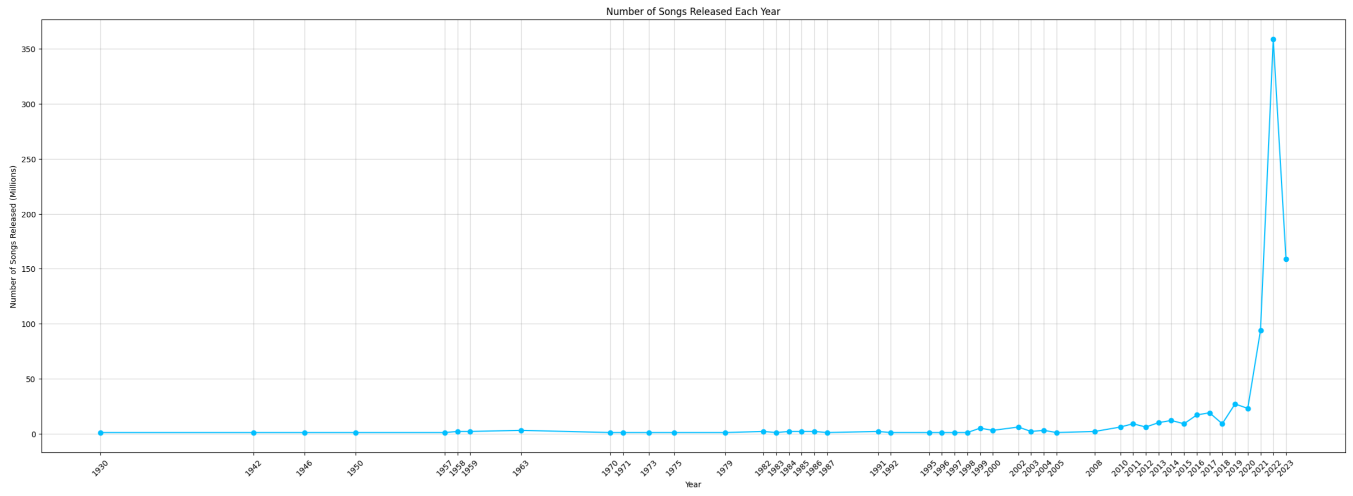
```
plt.figure(figsize=(10, 6))
plt.hist(df_spotify['bpm'], bins=20, color='skyblue', edgecolor='black')
plt.title('Distribution of Beats Per Minute (BPM)')
plt.xlabel('BPM')
plt.ylabel('Frequency')
plt.grid(True)
plt.show()
```



Number of songs released each year: Linechart

```
# Group the data by 'released_year' and count the number of songs released each year
year_counts = df_spotify_cleaned['released_year'].value_counts().sort_index()

# Create a line chart to visualize the number of songs released each year
plt.figure(figsize=(30, 10))
plt.plot(year_counts.index, year_counts.values, marker='o', color='deepskyblue', linestyle='
plt.title('Number of Songs Released Each Year' )
plt.xlabel('Year')
plt.ylabel('Number of Songs Released (Millions)')
plt.grid(True, alpha=0.5)
plt.xticks(year_counts.index, rotation=45)
plt.show()
```



Analysis:

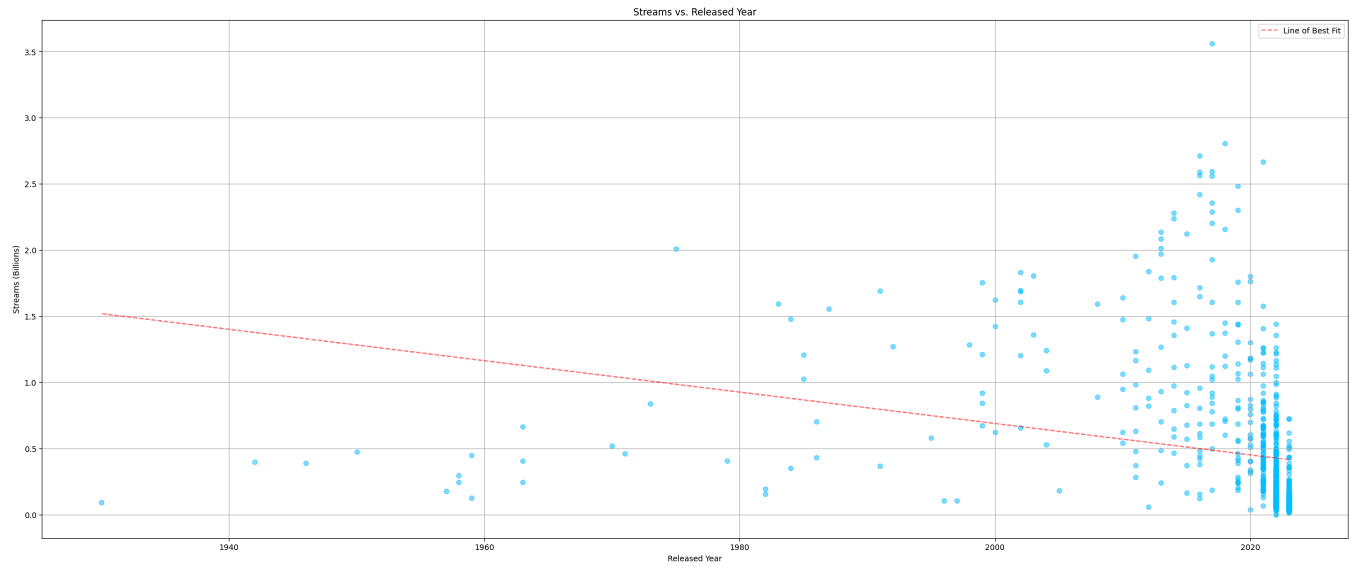
- Giant increase of released songs in 2021.
- Result of the Pandemic? Social Media?
- 2023: drops down (but note that the data cuts off from the 7th month).
-

Streams Vs. Year Released: Scatterplot

```
from scipy import stats
#A library that uses arrays and mathematical algorithms
#Used for the line of best fit.
```

```
# Create a scatter plot for 'Streams' vs. 'Released Year'
plt.figure(figsize=(30, 12))
plt.scatter(df_spotify_cleaned['released_year'], df_spotify_cleaned['streams'] / 1e9, color=
plt.title('Streams vs. Released Year')
plt.xlabel('Released Year')
plt.ylabel('Streams (Billions)')
plt.grid(True)

slope, intercept, r_value, p_value, std_err = stats.linregress(df_spotify_cleaned['released_
line = slope * df_spotify_cleaned['released_year'] + intercept
plt.plot(df_spotify_cleaned['released_year'], line, color='red', linestyle='--', label='Line
plt.legend()
plt.show()
```

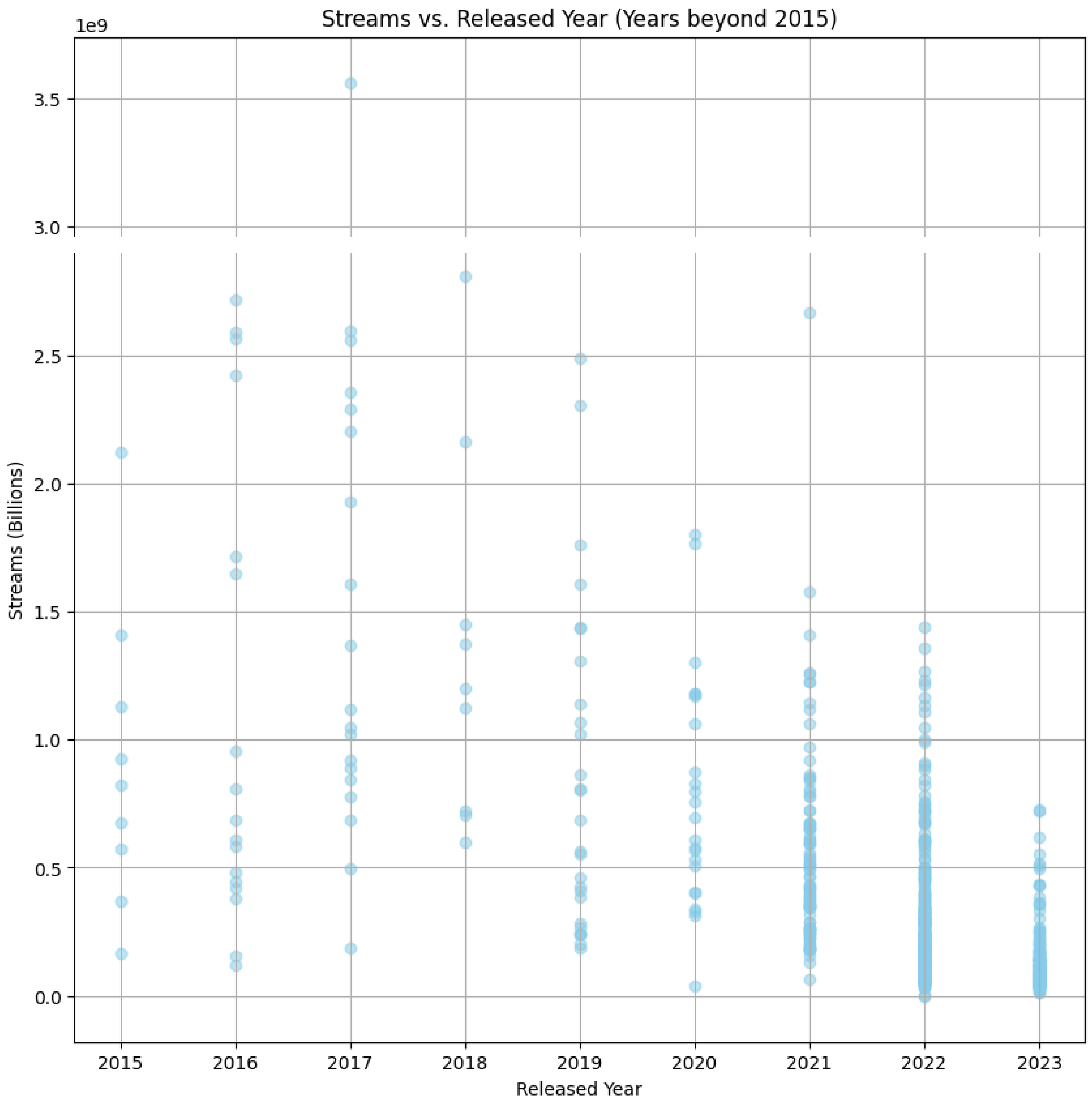


More Zoomed in chart with recent years:

```
# Filter the DataFrame to include only years beyond 2015
df_filtered = df_spotify_cleaned[df_spotify_cleaned['released_year'] > 2014]

# Create a scatter plot for 'Streams' vs. 'Released Year' for years beyond 2015
plt.figure(figsize=(10, 10))
plt.scatter(df_filtered['released_year'], df_filtered['streams'], color='skyblue', alpha=0.5
plt.title('Streams vs. Released Year (Years beyond 2015)')
plt.xlabel('Released Year')
```

```
    plt.ylabel('Streams (Billions)')
    plt.grid(True)
```



Analysis:

- Downward line of best fit, but important to note that before the 2000s, very little songs were released.
- Near the 2020s (around 2021 specifically), more songs were released
- More songs overall could mean less streams for everything because of the volume.

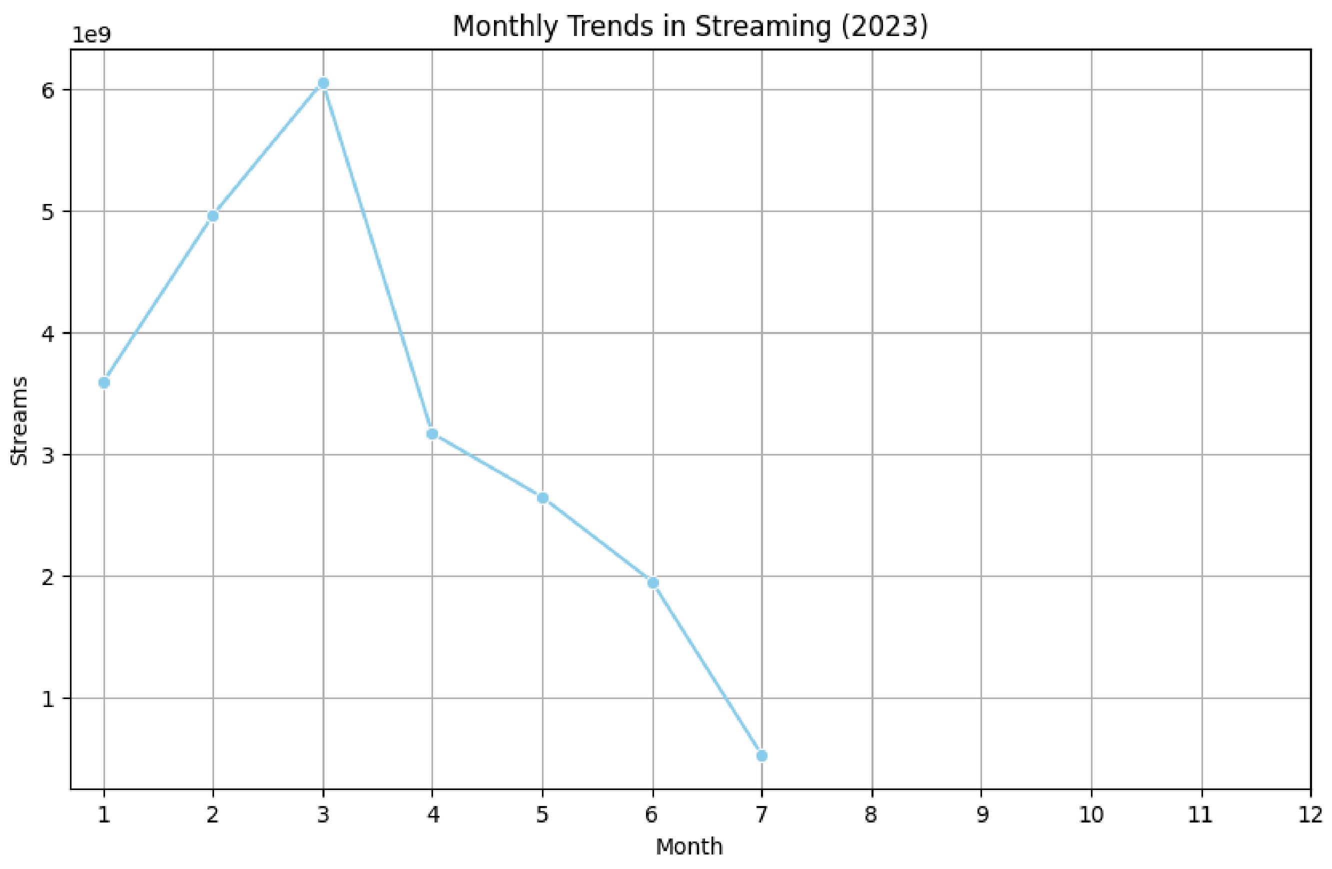
Monthly trends? (2023)

```
# Filter the DataFrame to include only the year 2023
df_2023 = df_spotify_cleaned[df_spotify_cleaned['released_year'] == 2023]

# Group the filtered data by month and sum up the streaming counts for each month
monthly_streams_2023 = df_2023.groupby('released_month')['streams'].sum()

# Reset index to convert the series to a DataFrame with columns
monthly_streams_2023 = monthly_streams_2023.reset_index()

# Create a line chart to visualize the monthly trends in streaming for 2023
plt.figure(figsize=(10, 6))
sns.lineplot(data=monthly_streams_2023, x='released_month', y='streams', marker='o', color='
plt.title('Monthly Trends in Streaming (2023)')
plt.xlabel('Month')
plt.ylabel('Streams')
plt.xticks(range(1, 13))  # Set x-axis ticks to represent months
plt.grid(True)
plt.show()
```



Comparing the data from 2020 onward.

```
# Filter the DataFrame to include only the years 2020, 2021, 2022, and 2023
df_filtered = df_spotify_cleaned[df_spotify_cleaned['released_year'].isin([2020, 2021, 202

# Group the filtered data by year and month and sum up the streaming counts for each month
monthly_streams = df_filtered.groupby(['released_year', 'released_month'])['streams'].sum(

# Reset index to convert the multi-index to columns
monthly_streams = monthly_streams.reset_index()

# Create a line chart to visualize the monthly trends in streaming for each year
plt.figure(figsize=(15, 10))
sns.lineplot(data=monthly_streams, x='released_month', y='streams', hue='released_year', ma
plt.title('Monthly Trends in Streaming (2020-2023)')
plt.xlabel('Month')
plt.ylabel('Streams (Billions)')
plt.xticks(range(1, 13))  # Set x-axis ticks to represent months
plt.grid(True)
plt.legend(title='Year', bbox_to_anchor=(1.05, 1), loc='upper left')
```