famBus

November 30, 2023

This dataset contains a comprehensive list of the most famous songs of 2023 as listed on Spotify. The dataset offers a wealth of features beyond what is typically available in similar datasets. It provides insights into each song's attributes, popularity, and presence on various music platforms. The dataset includes information such as track name, artist(s) name, release date, Spotify playlists and charts, streaming statistics, Apple Music presence, Deezer presence, Shazam charts, and various audio features.

Field

Description

track_name

Name of the song

artist(s)_name

Name of the artist(s) of the song

artist_count

Number of artists contributing to the song

released_year

Year when the song was released

released_month

Month when the song was released

released_day

Day of the month when the song was released

in_spotify_playlists

Number of Spotify playlists the song is included in

in_spotify_charts

Presence and rank of the song on Spotify charts

streams

Total number of streams on Spotify

in_apple_playlists

Number of Apple Music playlists the song is included in

in_apple_charts

Presence and rank of the song on Apple Music charts

in_deezer_playlists

Number of Deezer playlists the song is included in

in_deezer_charts

Presence and rank of the song on Deezer charts

in_shazam_charts

Presence and rank of the song on Shazam charts

bpm

Beats per minute, a measure of song tempo

key

Key of the song

mode

Mode of the song (major or minor)

danceability_%

Percentage indicating how suitable the song is for dancing

valence_%

Positivity of the song's musical content

energy_%

Perceived energy level of the song

acousticness_%

Amount of acoustic sound in the song

instrumentalness_%

Amount of instrumental content in the song

liveness_%

Presence of live performance elements

speechiness_%

Amount of spoken words in the song

Importing the Necessary Library

[ ]:
```
import numpy as np
import matplotlib.pyplot as plt
```

```
import matplotlib.ticker as ticker
import pandas as pd
import seaborn as sns
```

Loading the Data

```
[ ]: df_org=pd.read_csv("spt.csv",encoding= 'unicode-escape')
```

```
[ ]: df_org.info()
```

```
<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
```

Descriptive Statistics of the Dataframe

Quick overview of the distribution of the numerical data in the dataframe. This helps us in understanding key statistical measures for each colum

```
[ ]: df_org.describe()
```

[ ]:
```
       artist_count  released_year  released_month  released_day  \
count    953.000000     953.000000      953.000000    953.000000
mean       1.556139    2018.238195        6.033578     13.930745
std        0.893044      11.116218        3.566435      9.201949
min        1.000000    1930.000000        1.000000      1.000000
25%        1.000000    2020.000000        3.000000      6.000000
50%        1.000000    2022.000000        6.000000     13.000000
75%        2.000000    2022.000000        9.000000     22.000000
max        8.000000    2023.000000       12.000000     31.000000

       in_spotify_playlists  in_spotify_charts  in_apple_playlists  \
count            953.000000         953.000000          953.000000
mean            5200.124869          12.009444           67.812172
std             7897.608990          19.575992           86.441493
min               31.000000           0.000000            0.000000
25%              875.000000           0.000000           13.000000
50%             2224.000000           3.000000           34.000000
75%             5542.000000          16.000000           88.000000
max            52898.000000         147.000000          672.000000

       in_apple_charts  in_deezer_charts         bpm  danceability_%  \
count       953.000000        953.000000  953.000000       953.00000
mean         51.908709          2.666317  122.540399        66.96957
std          50.630241          6.035599   28.057802        14.63061
min           0.000000          0.000000   65.000000        23.00000
25%           7.000000          0.000000  100.000000        57.00000
50%          38.000000          0.000000  121.000000        69.00000
75%          87.000000          2.000000  140.000000        78.00000
max         275.000000         58.000000  206.000000        96.00000

        valence_%    energy_%  acousticness_%  instrumentalness_%  liveness_%  \
count  953.000000  953.000000      953.000000          953.000000  953.000000
mean    51.431270   64.279119       27.057712            1.581322   18.213012
std     23.480632   16.550526       25.996077            8.409800   13.711223
min      4.000000    9.000000        0.000000            0.000000    3.000000
25%     32.000000   53.000000        6.000000            0.000000   10.000000
50%     51.000000   66.000000       18.000000            0.000000   12.000000
75%     70.000000   77.000000       43.000000            0.000000   24.000000
max     97.000000   97.000000       97.000000           91.000000   97.000000

       speechiness_%
count     953.000000
mean       10.131165
std         9.912888
min         2.000000
25%         4.000000
50%         6.000000
```

```
75%          11.000000
max          64.000000
```

Cleaning Data

```
[ ]: #Renaming the column name of artist

df_org = df_org.rename(columns={'artist(s)_name': 'artists'})
```

```
[ ]: # Checking if there are any duplicate rows

print("Number of rows which are duplicated entirely: ", df_org.duplicated().
 ↪sum())
```

```
Number of rows which are duplicated entirely:  0
```

Finding number of NULL values in each column and corresponding percentage.

```
[ ]: df_null = pd.DataFrame(columns=['Total Null Values', 'Null Percentage'])

def check_null(df):
    df['Total Null Values'] = df_org.isnull().sum()
    df['Null Percentage'] = (df_org.isnull().sum() / len(df_org)) * 100
    return df

check_null(df_null)
```

[ ]:

	Total Null Values	Null Percentage
track_name	0	0.00000
artists	0	0.00000
artist_count	0	0.00000
released_year	0	0.00000
released_month	0	0.00000
released_day	0	0.00000
in_spotify_playlists	0	0.00000
in_spotify_charts	0	0.00000
streams	0	0.00000
in_apple_playlists	0	0.00000
in_apple_charts	0	0.00000
in_deezer_playlists	0	0.00000
in_deezer_charts	0	0.00000
in_shazam_charts	50	5.24659
bpm	0	0.00000
key	95	9.96852
mode	0	0.00000
danceability_%	0	0.00000
valence_%	0	0.00000
energy_%	0	0.00000
acousticness_%	0	0.00000

```
instrumentalness_%                    0           0.00000
liveness_%                            0           0.00000
speechiness_%                         0           0.00000
```

The columns 'in_shazam_charts' and 'key' have NULL values.

Finding rows with null value in in_shazam_charts column

```
[ ]: df_org[df_org['in_shazam_charts'].isnull()][['track_name', 'in_shazam_charts']]
```

```
[ ]:                              track_name in_shazam_charts
     14                               As It Was              NaN
     54                            Another Love              NaN
     55                         Blinding Lights              NaN
     71                              Heat Waves              NaN
     73                         Sweater Weather              NaN
     86                        Someone You Loved              NaN
     127                        Watermelon Sugar              NaN
     158                                   Ghost              NaN
     159                     Under The Influence              NaN
     180                           Night Changes              NaN
     243                             Unstoppable              NaN
     274                                 Shivers              NaN
     320                      Gangsta's Paradise              NaN
     392                               Calm Down              NaN
     395                              Space Song              NaN
     403                One Kiss (with Dua Lipa)              NaN
     410        INDUSTRY BABY (feat. Jack Harlow)              NaN
     429                              Bad Habits              NaN
     434                                   Woman              NaN
     440                                Payphone              NaN
     441           All I Want for Christmas Is You              NaN
     442                          Last Christmas              NaN
     443        Rockin' Around The Christmas Tree              NaN
     444                         Jingle Bell Rock              NaN
     446                           Santa Tell Me              NaN
     449                                 Snowman              NaN
     500                              ýýýabcdefu              NaN
     501                               Sacrifice              NaN
     504                             Out of Time              NaN
     506                 We Don't Talk About Bruno              NaN
     507                                   Pepas              NaN
     513                                good 4 u              NaN
     518                            Need To Know              NaN
     519           MONTERO (Call Me By Your Name)              NaN
     520                love nwantiti (ah ah ah)              NaN
     529                                   MONEY              NaN
     531                        Happier Than Ever              NaN
```

```
532       Moth To A Flame (with The Weeknd)              NaN
533                                 traitor              NaN
534                                   Toxic              NaN
535                         drivers license              NaN
549                    Love Nwantiti - Remix              NaN
554  Peaches (feat. Daniel Caesar & Giveon)              NaN
560                            Life Goes On              NaN
566                                Dynamite              NaN
584                 Mood (feat. Iann Dior)               NaN
620                            Dance Monkey              NaN
625                                  Arcade              NaN
727              Somebody That I Used To Know            NaN
927      I Really Want to Stay at Your House             NaN
```

Filling NULL values of in_shazam_charts with mean of in_spotify_charts, in_deezer_charts, in_apple_charts

[ ]:
```python
# Function to calculate the mean of non-null values in a row
def row_mean(row):
    non_null_values = row.dropna()
    if non_null_values.empty:
        return np.nan
    return int(non_null_values.mean())

# Applying the function to each row and filling null values in␣
↪'in_shazam_charts'
df_org['in_shazam_charts'] = df_org.apply(lambda row:␣
↪row_mean(row[['in_spotify_charts', 'in_apple_charts', 'in_deezer_charts']]),␣
↪axis=1)
print('Number of NULL values in \'in_shazam_charts\':',␣
↪df_org['in_shazam_charts'].isnull().sum())
```

```
Number of NULL values in 'in_shazam_charts': 0
```

Finding rows with NULL values in key column

[ ]:
```python
df_org[df_org['key'].isnull()][['track_name', 'key']]
```

[ ]:
```
                                              track_name  key
12                                               Flowers  NaN
17   What Was I Made For? [From The Motion Picture …   NaN
22                                       I Wanna Be Yours  NaN
35                                      Los del Espacio  NaN
44    Barbie World (with Aqua) [From Barbie The Album]  NaN
..                                                   …    …
899                                      Hold Me Closer  NaN
901                                          After LIKE  NaN
903                B.O.T.A. (Baddest Of Them All) - Edit  NaN
```

```
938                                          Labyrinth  NaN
940                                      Sweet Nothing  NaN

[95 rows x 2 columns]
```

For the null values in the key column, the key of the song can be converted into integers using the standard Pitch Class Notation. The null values for the key column will have a value of -1.

Tonal Counterparts	Pitch Class
-1	NULL
0	C
1	C#
2	D
3	D#
4	E
5	F
6	F#
7	G
8	G#
9	A
10	A#
11	B

[ ]:
```
# Mapping table
pitch_class_mapping = {
    np.nan:-1,
    'C': 0, 'C#': 1, 'D': 2, 'D#': 3,
    'E': 4, 'F': 5, 'F#': 6, 'G': 7,
    'G#': 8, 'A': 9, 'A#': 10, 'B': 11
}

df_org['key'] = df_org['key'].map(pitch_class_mapping)
```

Checking if there are still any NULL values present in any column

[ ]:
```
check_null(df_null)
df_null
```

[ ]:
```
                      Total Null Values  Null Percentage
track_name                            0              0.0
artists                               0              0.0
artist_count                          0              0.0
released_year                         0              0.0
released_month                        0              0.0
released_day                          0              0.0
in_spotify_playlists                  0              0.0
in_spotify_charts                     0              0.0
```

```
streams                                0              0.0
in_apple_playlists                     0              0.0
in_apple_charts                        0              0.0
in_deezer_playlists                    0              0.0
in_deezer_charts                       0              0.0
in_shazam_charts                       0              0.0
bpm                                    0              0.0
key                                    0              0.0
mode                                   0              0.0
danceability_%                         0              0.0
valence_%                              0              0.0
energy_%                               0              0.0
acousticness_%                         0              0.0
instrumentalness_%                     0              0.0
liveness_%                             0              0.0
speechiness_%                          0              0.0
```

In in_deezer_playlist (object type), there are comma values for integers greater than 999. Convert the entire column to integer

[ ]:
```
df_org['in_deezer_playlists'].info()
```

```
<class 'pandas.core.series.Series'>
RangeIndex: 953 entries, 0 to 952
Series name: in_deezer_playlists
Non-Null Count  Dtype
--------------  -----
953 non-null    object
dtypes: object(1)
memory usage: 7.6+ KB
```

Replacing commas with blank space using regex and converting to integer

[ ]:
```
df_org['in_deezer_playlists'] = df_org['in_deezer_playlists'].str.replace(',',
 ↪'', regex=True).astype('int64')
df_org['in_deezer_playlists'].info()
```

```
<class 'pandas.core.series.Series'>
RangeIndex: 953 entries, 0 to 952
Series name: in_deezer_playlists
Non-Null Count  Dtype
--------------  -----
953 non-null    int64
dtypes: int64(1)
memory usage: 7.6 KB
```

While analysing the data, we found discrepancy in streams column. It should be in int64 data type, while it is currently in object data type.

Checking the particular track with discrepancy

[ ]:
```
df_org[df_org['streams'].apply(pd.to_numeric, errors='coerce').isna()]
```

[ ]:
```
                           track_name           artists  artist_count  \
574  Love Grows (Where My Rosemary Goes)  Edison Lighthouse             1

     released_year  released_month  released_day  in_spotify_playlists  \
574           1970               1             1                  2877

     in_spotify_charts                                           streams  \
574                  0  BPM110KeyAModeMajorDanceability53Valence75Ener…

     in_apple_playlists  …  bpm  key   mode  danceability_%  valence_%  \
574                  16  …  110    9  Major              53         75

     energy_% acousticness_%  instrumentalness_%  liveness_%  speechiness_%
574        69              7                   0          17              3

[1 rows x 24 columns]
```

Change the stream value of the particular song which is Invalid

[ ]:
```
df_org.loc[df_org['track_name'] == "Love Grows (Where My Rosemary Goes)",␣
↪'streams'] = 211283228
df_org.loc[df_org['track_name'] == "Love Grows (Where My Rosemary Goes)"]
```

[ ]:
```
                           track_name           artists  artist_count  \
574  Love Grows (Where My Rosemary Goes)  Edison Lighthouse             1

     released_year  released_month  released_day  in_spotify_playlists  \
574           1970               1             1                  2877

     in_spotify_charts    streams  in_apple_playlists  …  bpm  key   mode  \
574                  0  211283228                  16  …  110    9  Major

     danceability_%  valence_%  energy_% acousticness_%  instrumentalness_%  \
574              53         75        69              7                   0

     liveness_%  speechiness_%
574          17              3

[1 rows x 24 columns]
```

Change the datatype of streams from Object to Int

[ ]:
```
df_org['streams']=df_org['streams'].astype('int64')
df_org['streams'].info()
```

```
<class 'pandas.core.series.Series'>
RangeIndex: 953 entries, 0 to 952
```

```
Series name: streams
Non-Null Count  Dtype
--------------  -----
953 non-null    int64
dtypes: int64(1)
memory usage: 7.6 KB
```

Checking track names to check for discrepancy

```
[ ]: unique_track = df_org['track_name'].unique()
     unique_track.sort()
     print(unique_track[:20])
     print(unique_track[-20:])
```

```
["'Till I Collapse" '(It Goes Like) Nanana - Edit'
 '10 Things I Hate About You' '10:35' '2 Be Loved (Am I Ready)' '2055'
 '212' '25k jacket (feat. Lil Baby)' '295' '505' '69'
 'A Holly Jolly Christmas - Single Version' 'A Tale By Quincy'
 'A Tu Merced' 'A Veces (feat. Feid)' 'ALIEN SUPERSTAR' 'AM Remix'
 'AMARGURA' 'AMERICA HAS A PROBLEM (feat. Kendrick Lamar)' 'AMG']
['jealousy, jealousy' 'love nwantiti (ah ah ah)' 'lovely - Bonus Track'
 'on the street (with J. Cole)' 'positions' 'psychofreak (feat. WILLOW)'
 'pushin P (feat. Young Thug)' 'sentaDONA (Remix) s2'
 "she's all i wanna be" 'this is what falling in love feels like'
 'thought i was playing' 'traitor' 'un x100to' 'vampire'
 'we fell in love in october' 'you broke me first' 'ýýý98 Braves'
 'ýýýabcdefu' 'ýýýýýýýýýýýý' 'ýýýýýýýýýýýýýýýýýýýýýý']
```

While analysing the dataset, we found that certain track names contains special characters. So we decided to replace it.

```
[ ]: #Checking rows with special characters
     characters_to_replace = ['ý', 'ï', '¿', 'Â', '½', 'Ã', '¯']

     def contains_special_character(cell):
         return any(char in cell for char in characters_to_replace)

     rows_with_special_characters = df_org[df_org.astype(str).apply(lambda row:␣
      ↪any(contains_special_character(cell) for cell in row), axis=1)]


     print("Number of rows with special characters: ", rows_with_special_characters.
      ↪shape[0])
     print("Rows with special characters:")
     rows_with_special_characters
```

```
Number of rows with special characters:  109
Rows with special characters:
```

[ ]:
```
                                             track_name  \
21   I Can See You (Taylorï¿½ï¿½ï¿½s Version) (From…
26                        Calm Down (with Selena Gomez)
36                       Frï¿½ï¿½gil (feat. Grupo Front
60                                               Tï¿½ï¿
63                                                 BESO
..                                                    …
887                                     ALIEN SUPERSTAR
913                                     XQ Te Pones Asï¿
915                                          Sin Seï¿½ï
918                                       THE LONELIEST
929                             Bamba (feat. Aitch & BIA)

                              artists  artist_count  released_year  \
21                       Taylor Swift             1           2023
26              Rï¿½ï¿½ma, Selena G              2           2022
36   Yahritza Y Su Esencia, Grupo Frontera         2           2023
60                dennis, MC Kevin o Chris             2           2023
63               Rauw Alejandro, ROSALï¿½            2           2023
..                                  …             …              …
887                          Beyoncï¿             1           2022
913                       Yandel, Feid            2           2022
915            Ovy On The Drums, Quevedo            2           2022
918                        Mï¿½ï¿½ne             1           2022
929                 Luciano, Aitch, Bï¿½            3           2022

     released_month  released_day  in_spotify_playlists  in_spotify_charts  \
21                7             7                   516                 38
26                3            25                  7112                 77
36                4             7                   672                 34
60                5             4                   731                 15
63                3            24                  4053                 50
..              …           …                   …                …
887               7            29                  2688                  0
913               9            13                   308                  0
915               7            22                  1097                  2
918              10             7                  1585                  5
929               9            22                   869                  7

       streams  in_apple_playlists  …  bpm  key   mode  danceability_%  \
21    52135248                  73  …  123    6  Major              69
26   899183384                 202  …  107   11  Major              80
36   188933502                  19  …  150    6  Major              61
60   111947664                  27  …  130   11  Major              86
63   357925728                  82  …   95    5  Minor              77
..         …                 …  …  …    …    …              …
887  171788484                  39  …  122   10  Minor              55
```

```
913   47093942                   6  …   92   10  Major              81
915  209106362                  18  …  118   11  Minor              82
918  225093344                  78  …  130    2  Major              52
929  146223492                  14  …  138   10  Major              80

     valence_%  energy_% acousticness_%  instrumentalness_%  liveness_%  \
21          82        76              6                   0           6
26          82        80             43                   0          14
36          39        73             37                   0          11
60          59        96             50                   1           9
63          53        64             74                   0          17
..         …       …            …                 …         …
887         46        64              0                   0          17
913         48        70             13                   0          15
915         75        85             33                   1          11
918         24        60              0                   0           8
929         82        81             14                   0          13

     speechiness_%
21               3
26               4
36               3
60               5
63              14
..             …
887             10
913              7
915              4
918              3
929             36

[109 rows x 24 columns]
```

There are 109 rows with such special characters.

Changing the track name where there are special characters

[ ]:
```
for char in characters_to_replace:
    df_org['track_name'] = df_org['track_name'].str.replace(char, '')

df_org.head(5)
```

[ ]:
```
                               track_name           artists  artist_count  \
0  Seven (feat. Latto) (Explicit Ver.)  Latto, Jung Kook             2
1                                   LALA       Myke Towers             1
2                                vampire    Olivia Rodrigo             1
3                           Cruel Summer      Taylor Swift             1
4                         WHERE SHE GOES         Bad Bunny             1
```

```
   released_year  released_month  released_day  in_spotify_playlists  \
0           2023               7            14                   553
1           2023               3            23                  1474
2           2023               6            30                  1397
3           2019               8            23                  7858
4           2023               5            18                  3133

   in_spotify_charts    streams  in_apple_playlists  …  bpm  key   mode  \
0                147  141381703                  43  …  125   11  Major
1                 48  133716286                  48  …   92    1  Major
2                113  140003974                  94  …  138    5  Major
3                100  800840817                 116  …  170    9  Major
4                 50  303236322                  84  …  144    9  Minor

   danceability_%  valence_%  energy_% acousticness_%  instrumentalness_%  \
0              80         89        83             31                   0
1              71         61        74              7                   0
2              51         32        53             17                   0
3              55         58        72             11                   0
4              65         23        80             14                  63

   liveness_%  speechiness_%
0           8              4
1          10              4
2          31              6
3          11             15
4          11              6

[5 rows x 24 columns]
```

While changing the track name, we came across songs whose track name is entirely special characters

[ ]:
```python
#Checking rows where track_name is empty or NULL

null_track_name_rows = df_org[df_org['track_name'].isnull() |
 (df_org['track_name'] == '')]
print("Rows with NULL or empty track name:")
null_track_name_rows
```

```
Rows with NULL or empty track name:
```

[ ]:
```
    track_name     artists  artist_count  released_year  released_month  \
174                YOASOBI             1           2023               4
374             Fujii Kaze             1           2020               5

     released_day  in_spotify_playlists  in_spotify_charts    streams  \
174            12                   356                 16  143573775
```

```
374              20                    685                  14  403097450

     in_apple_playlists  …  bpm  key   mode  danceability_%  valence_%  \
174                  35  …  166    1  Major              57         84
374                  24  …  158    6  Minor              60         52

     energy_% acousticness_%  instrumentalness_%  liveness_%  speechiness_%
174        94              11                   0          37              9
374        76              17                   0          19              5

[2 rows x 24 columns]
```

We change the track name by cross-referencing the artist and released date on the Internet

[ ]:
```python
#Replacing the track name with original

df_org.loc[374, 'track_name'] = 'Shinunoga E-Wa'
df_org.loc[174, 'track_name'] = 'Run Into The Night'
df_org.loc[[374, 174]]
```

[ ]:
```
              track_name     artists  artist_count  released_year  \
374       Shinunoga E-Wa  Fujii Kaze             1           2020
174  Run Into The Night     YOASOBI             1           2023

     released_month  released_day  in_spotify_playlists  in_spotify_charts  \
374               5            20                   685                 14
174               4            12                   356                 16

       streams  in_apple_playlists  …  bpm  key   mode  danceability_%  \
374  403097450                  24  …  158    6  Minor              60
174  143573775                  35  …  166    1  Major              57

     valence_%  energy_% acousticness_%  instrumentalness_%  liveness_%  \
374         52        76             17                   0          19
174         84        94             11                   0          37

     speechiness_%
374              5
174              9

[2 rows x 24 columns]
```

Checking artists with special characters

[ ]:
```python
total_artists_with_special_characters = 0

for char in characters_to_replace:
    char_present = df_org['artists'].str.contains(char)
```

```
    values_with_char = df_org['artists'][char_present]
    if not values_with_char.empty:
        total_artists_with_special_characters += len(values_with_char)

print('Total number of artists with special characters: ',␣
 ↪total_artists_with_special_characters)
```

```
Total number of artists with special characters:  136
```

Replacing special characters in artist name with #

[ ]:
```
for char in characters_to_replace:
    df_org['artists'] = df_org['artists'].str.replace(char, '#')

df_org.head(5)
```

[ ]:
```
                            track_name            artists  artist_count  \
0  Seven (feat. Latto) (Explicit Ver.)  Latto, Jung Kook             2
1                                 LALA        Myke Towers             1
2                              vampire     Olivia Rodrigo             1
3                         Cruel Summer       Taylor Swift             1
4                       WHERE SHE GOES          Bad Bunny             1

   released_year  released_month  released_day  in_spotify_playlists  \
0           2023               7            14                   553
1           2023               3            23                  1474
2           2023               6            30                  1397
3           2019               8            23                  7858
4           2023               5            18                  3133

   in_spotify_charts    streams  in_apple_playlists  …  bpm  key   mode  \
0                147  141381703                  43  …  125   11  Major
1                 48  133716286                  48  …   92    1  Major
2                113  140003974                  94  …  138    5  Major
3                100  800840817                 116  …  170    9  Major
4                 50  303236322                  84  …  144    9  Minor

   danceability_%  valence_%  energy_% acousticness_%  instrumentalness_%  \
0              80         89        83             31                   0
1              71         61        74              7                   0
2              51         32        53             17                   0
3              55         58        72             11                   0
4              65         23        80             14                  63

   liveness_%  speechiness_%
0           8              4
1          10              4
2          31              6
```

```
3           11              15
4           11               6

[5 rows x 24 columns]
```

While trying to split the artists from artists column, we encountered an error. On further analysing it, we found that the artists attribute of the song 'Nobody Like U - From "Turning Red" ' contains unwanted characters

```
[ ]: with pd.option_context('display.max_colwidth', None):
         print(df_org.loc[df_org['track_name'] == "Nobody Like U - From \"Turning␣
     ↪Red\""][['artists']])
```

```
                                          artists
759  Jordan Fisher, Josh Levi, Finneas O'Connell, 4*TOWN (From Disney and
Pixar#########s Turning Red), Topher Ngo, Grayson Vill
```

Fixing the artists of that track

```
[ ]: df_org.loc[df_org['track_name'] == "Nobody Like U - From \"Turning Red\"",␣
     ↪'artists']="Jordan Fisher, Josh Levi, Finneas O'Connell, 4*TOWN, Topher Ngo,␣
     ↪Grayson Vill"
     df_org.loc[df_org['track_name'] == "Nobody Like U - From \"Turning Red\""]
```

```
[ ]:                              track_name  \
     759  Nobody Like U - From "Turning Red"

                                                           artists  artist_count  \
     759  Jordan Fisher, Josh Levi, Finneas O'Connell, 4…             6

          released_year  released_month  released_day  in_spotify_playlists  \
     759           2022               2            25                   918

          in_spotify_charts    streams  in_apple_playlists  …  bpm  key   mode  \
     759                  0  120847157                  34  …  105    9  Minor

          danceability_%  valence_%  energy_% acousticness_%  instrumentalness_%  \
     759              91         73        72             13                   0

          liveness_%  speechiness_%
     759           9             15

     [1 rows x 24 columns]
```

Dealing with Duplicate Elements

Finding duplicate tracks by checking track name & artist. For duplicate tracks, we decided to keep the row with higher number of streams

[ ]:
```
duplicate = df_org.sort_values(by='streams', ascending=False)[df_org.
    ↪sort_values(by='streams', ascending=False).duplicated(['track_name',␣
    ↪'artists'], keep = 'first')]
duplicate
```

[ ]:
```
           track_name     artists  artist_count  released_year  \
764    About Damn Time       Lizzo             1           2022
873               SNAP   Rosa Linn             1           2022
482  SPIT IN MY FACE!     ThxSoMch             1           2022
512    Take My Breath  The Weeknd             1           2021

     released_month  released_day  in_spotify_playlists  in_spotify_charts  \
764               4            14                  9021                  0
873               3            19                  1818                  0
482              10            31                   573                  0
512               8             6                  2597                  0

       streams  in_apple_playlists  …  bpm  key   mode  danceability_%  \
764  723894473                 242  …  109   10  Minor              84
873  711366595                   3  …  170   -1  Major              56
482  301869854                   1  …  166    1  Major              70
512  130655803                  17  …  121   10  Minor              70

     valence_%  energy_% acousticness_%  instrumentalness_%  liveness_%  \
764         72        74             10                   0          34
873         52        64             11                   0          45
482         57        57              9                  20          11
512         35        77              1                   0          26

     speechiness_%
764              7
873              7
482              7
512              4

[4 rows x 24 columns]
```

Drop duplicate rows from the data

[ ]:
```
df_org.drop(duplicate.index, axis=0, inplace=True)
df_org.shape
```

[ ]:
```
(949, 24)
```

Resetting the index

[ ]:
```
df_org.reset_index(drop=True, inplace=True)
df_org
```

```
[ ]:                          track_name             artists  artist_count  \
    0    Seven (feat. Latto) (Explicit Ver.)    Latto, Jung Kook             2
    1                                   LALA         Myke Towers             1
    2                                vampire      Olivia Rodrigo             1
    3                           Cruel Summer        Taylor Swift             1
    4                         WHERE SHE GOES           Bad Bunny             1
    ..                                   …                   …           …
    944                         My Mind & Me        Selena Gomez             1
    945              Bigger Than The Whole Sky        Taylor Swift             1
    946                  A Veces (feat. Feid)   Feid, Paulo Londra             2
    947                        En La De Ella  Feid, Sech, Jhayco             3
    948                                Alone           Burna Boy             1

         released_year  released_month  released_day  in_spotify_playlists  \
    0             2023               7            14                   553
    1             2023               3            23                  1474
    2             2023               6            30                  1397
    3             2019               8            23                  7858
    4             2023               5            18                  3133
    ..             …              …            …                   …
    944           2022              11             3                   953
    945           2022              10            21                  1180
    946           2022              11             3                   573
    947           2022              10            20                  1320
    948           2022              11             4                   782

         in_spotify_charts    streams  in_apple_playlists  …  bpm  key   mode  \
    0                  147  141381703                  43  …  125   11  Major
    1                   48  133716286                  48  …   92    1  Major
    2                  113  140003974                  94  …  138    5  Major
    3                  100  800840817                 116  …  170    9  Major
    4                   50  303236322                  84  …  144    9  Minor
    ..                 …         …                  …  …  …  …  …
    944                  0   91473363                  61  …  144    9  Major
    945                  0  121871870                   4  …  166    6  Major
    946                  0   73513683                   2  …   92    1  Major
    947                  0  133895612                  29  …   97    1  Major
    948                  2   96007391                  27  …   90    4  Minor

         danceability_%  valence_%  energy_% acousticness_%  instrumentalness_%  \
    0                80         89        83             31                   0
    1                71         61        74              7                   0
    2                51         32        53             17                   0
    3                55         58        72             11                   0
    4                65         23        80             14                  63
    ..              …        …       …            …                 …
    944              60         24        39             57                   0
```

```
945                 42          7          24            83                     1
946                 80         81          67             4                     0
947                 82         67          77             8                     0
948                 61         32          67            15                     0

     liveness_%  speechiness_%
0             8              4
1            10              4
2            31              6
3            11             15
4            11              6
..          ...            ...
944           8              3
945          12              6
946           8              6
947          12              5
948          11              5

[949 rows x 24 columns]
```

Exporting the cleaned data to CSV and Making a copy of DataFrame to perform further analysis

[ ]:
```
df_org.to_csv('cleaned_data.csv')
df_copy = df_org.copy()
```

At this stage, the pre-processing is complete. Now we move on to identifying outliers and other analysis.

Identifying Outliers

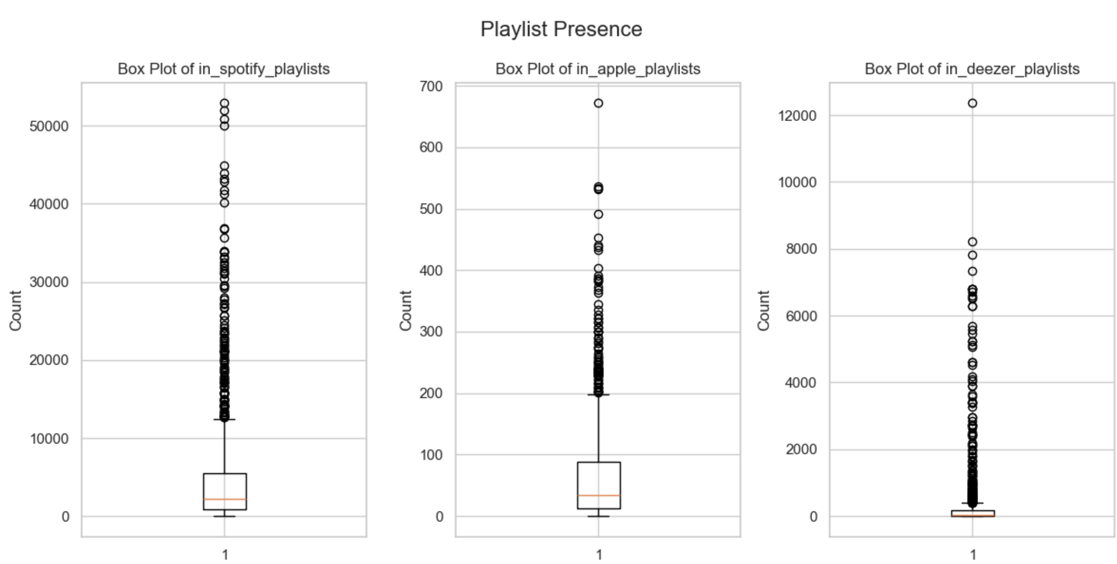
Removing columns with object type for correlation matrix

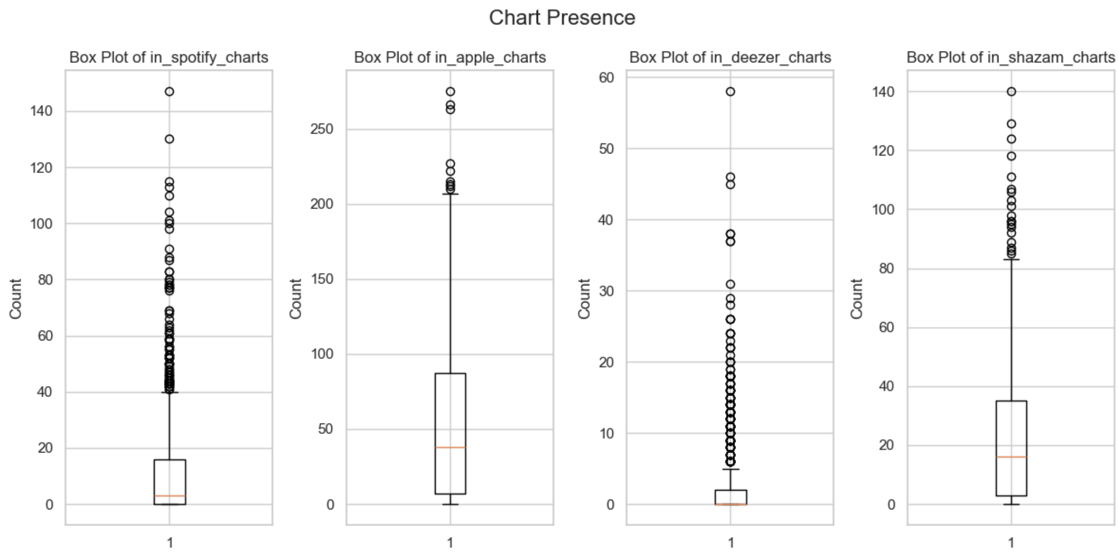
[ ]:
```
columns_to_drop = ['track_name', 'artists', 'mode']
out_check = df_copy.drop(columns=columns_to_drop)

columns = out_check.columns
```

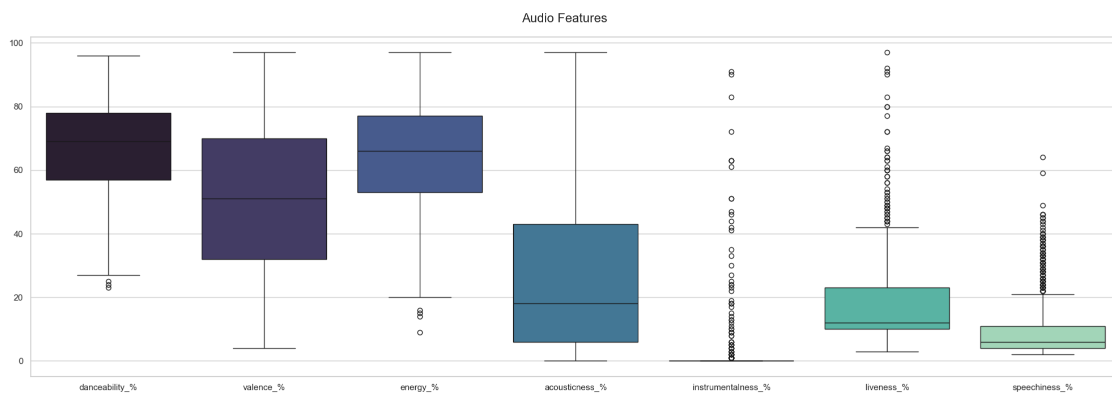
Box plots of Playlist Presence, Chart Presence and Audio Features

[ ]:
```
# Plot 1: Playlist presence

plt.figure(figsize=(12, 6))
plt.suptitle('Playlist Presence', fontsize=16)

for i, column in enumerate(['in_spotify_playlists', 'in_apple_playlists',␣
↪'in_deezer_playlists'], 1):
    plt.subplot(1, 3, i)
    plt.boxplot(out_check[column])
    plt.title(f'Box Plot of {column}')
    plt.ylabel('Count')
```

```python
plt.tight_layout()
plt.show()

# Plot 2: Chart presence
plt.figure(figsize=(12, 6))
plt.suptitle('Chart Presence', fontsize=16)

for i, column in enumerate(['in_spotify_charts', 'in_apple_charts',␣
 ↪'in_deezer_charts','in_shazam_charts'], 1):
    plt.subplot(1, 4, i)
    plt.boxplot(out_check[column])
    plt.title(f'Box Plot of {column}')
    plt.ylabel('Count')

plt.tight_layout()
plt.show()

# Plot 3: Audio features

plt.figure(figsize=(20, 7))
plt.suptitle('Audio Features', fontsize=16)

audio_features = ['danceability_%', 'valence_%', 'energy_%', 'acousticness_%',␣
 ↪'instrumentalness_%', 'liveness_%', 'speechiness_%']

# Use Seaborn's boxplot function to display multiple box plots
sns.boxplot(data=out_check[audio_features], palette='mako')
plt.tight_layout()
plt.show()
```







Correlation Analysis to find High Correlation

[ ]:
```
columns_to_keep = df_copy.columns.difference(['track_name','artists','mode'])
df_selected = df_copy[columns_to_keep]

df_selected.corr()
```

[ ]:
```
                 acousticness_%  artist_count       bpm  danceability_%  \
acousticness_%         1.000000     -0.103223 -0.015914       -0.236012
artist_count          -0.103223      1.000000 -0.036736        0.207960
bpm                   -0.015914     -0.036736  1.000000       -0.146076
danceability_%        -0.236012      0.207960 -0.146076        1.000000
energy_%              -0.577502      0.137882  0.026973        0.197616
```

in_apple_charts	-0.077860	-0.089804	0.034603	-0.025897
in_apple_playlists	-0.062320	-0.051329	0.027462	-0.028197
in_deezer_charts	-0.028517	-0.002622	0.031301	0.067460
in_deezer_playlists	-0.064188	-0.072244	-0.034531	-0.071539
in_shazam_charts	-0.078044	-0.074165	0.039972	-0.003990
in_spotify_charts	-0.057024	-0.020151	0.036597	0.030664
in_spotify_playlists	-0.065251	-0.102662	-0.017714	-0.107425
instrumentalness_%	0.044117	-0.049324	-0.004560	-0.089819
key	-0.018074	-0.000318	0.024242	0.028541
liveness_%	-0.048060	0.044970	-0.002786	-0.077736
released_day	-0.004992	-0.016583	-0.034405	0.049327
released_month	0.055019	0.038237	-0.039978	-0.046850
released_year	-0.123366	0.088508	-0.006323	0.187294
speechiness_%	-0.023848	0.119011	0.040275	0.185581
streams	-0.004561	-0.136456	-0.002054	-0.104962
valence_%	-0.082006	0.128441	0.041316	0.408211

	energy_%	in_apple_charts	in_apple_playlists	\
acousticness_%	-0.577502	-0.077860	-0.062320	
artist_count	0.137882	-0.089804	-0.051329	
bpm	0.026973	0.034603	0.027462	
danceability_%	0.197616	-0.025897	-0.028197	
energy_%	1.000000	0.104120	0.051658	
in_apple_charts	0.104120	1.000000	0.415088	
in_apple_playlists	0.051658	0.415088	1.000000	
in_deezer_charts	0.093434	0.385640	0.364570	
in_deezer_playlists	0.065069	0.173303	0.473100	
in_shazam_charts	0.111432	0.955245	0.415731	
in_spotify_charts	0.082617	0.552367	0.234163	
in_spotify_playlists	0.033533	0.270913	0.708634	
instrumentalness_%	-0.037390	-0.010930	-0.055613	
key	-0.005228	-0.064544	-0.055284	
liveness_%	0.116342	-0.017527	-0.050873	
released_day	0.052256	0.014234	0.027949	
released_month	-0.083468	-0.019188	0.001624	
released_year	0.095054	-0.035288	-0.199647	
speechiness_%	-0.004306	-0.152129	-0.108587	
streams	-0.026083	0.321526	0.773518	
valence_%	0.358206	0.048571	0.055200	

	in_deezer_charts	in_deezer_playlists	in_shazam_charts	\
acousticness_%	-0.028517	-0.064188	-0.078044	
artist_count	-0.002622	-0.072244	-0.074165	
bpm	0.031301	-0.034531	0.039972	
danceability_%	0.067460	-0.071539	-0.003990	
energy_%	0.093434	0.065069	0.111432	
in_apple_charts	0.385640	0.173303	0.955245	

in_apple_playlists	0.364570	0.473100	0.415731
in_deezer_charts	1.000000	0.066844	0.559931
in_deezer_playlists	0.066844	1.000000	0.162055
in_shazam_charts	0.559931	0.162055	1.000000
in_spotify_charts	0.604918	0.087783	0.766718
in_spotify_playlists	0.142981	0.826524	0.265623
instrumentalness_%	0.006890	-0.016406	-0.009792
key	-0.025870	-0.041766	-0.065792
liveness_%	-0.010407	-0.026113	-0.027853
released_day	0.074538	-0.084248	0.024084
released_month	-0.003110	-0.087894	-0.029814
released_year	0.095249	-0.306591	0.001835
speechiness_%	-0.080570	-0.062743	-0.147007
streams	0.228564	0.598337	0.335569
valence_%	0.073628	-0.013916	0.054106

	…	in_spotify_playlists	instrumentalness_%	key	\
acousticness_%	…	-0.065251	0.044117	-0.018074	
artist_count	…	-0.102662	-0.049324	-0.000318	
bpm	…	-0.017714	-0.004560	0.024242	
danceability_%	…	-0.107425	-0.089819	0.028541	
energy_%	…	0.033533	-0.037390	-0.005228	
in_apple_charts	…	0.270913	-0.010930	-0.064544	
in_apple_playlists	…	0.708634	-0.055613	-0.055284	
in_deezer_charts	…	0.142981	0.006890	-0.025870	
in_deezer_playlists	…	0.826524	-0.016406	-0.041766	
in_shazam_charts	…	0.265623	-0.009792	-0.065792	
in_spotify_charts	…	0.163980	-0.009125	-0.053430	
in_spotify_playlists	…	1.000000	-0.026920	-0.068419	
instrumentalness_%	…	-0.026920	1.000000	-0.001073	
key	…	-0.068419	-0.001073	1.000000	
liveness_%	…	-0.046688	-0.044285	-0.019352	
released_day	…	-0.078802	0.015024	-0.022257	
released_month	…	-0.104163	0.031378	-0.015290	
released_year	…	-0.392191	-0.015202	0.032095	
speechiness_%	…	-0.090188	-0.083158	0.036263	
streams	…	0.790053	-0.044053	-0.048590	
valence_%	…	-0.022439	-0.133835	0.031472	

	liveness_%	released_day	released_month	released_year	\
acousticness_%	-0.048060	-0.004992	0.055019	-0.123366	
artist_count	0.044970	-0.016583	0.038237	0.088508	
bpm	-0.002786	-0.034405	-0.039978	-0.006323	
danceability_%	-0.077736	0.049327	-0.046850	0.187294	
energy_%	0.116342	0.052256	-0.083468	0.095054	
in_apple_charts	-0.017527	0.014234	-0.019188	-0.035288	
in_apple_playlists	-0.050873	0.027949	0.001624	-0.199647	

```
in_deezer_charts        -0.010407    0.074538   -0.003110    0.095249
in_deezer_playlists     -0.026113   -0.084248   -0.087894   -0.306591
in_shazam_charts        -0.027853    0.024084   -0.029814    0.001835
in_spotify_charts       -0.045690    0.022953   -0.047569    0.070567
in_spotify_playlists    -0.046688   -0.078802   -0.104163   -0.392191
instrumentalness_%      -0.044285    0.015024    0.031378   -0.015202
key                     -0.019352   -0.022257   -0.015290    0.032095
liveness_%               1.000000    0.001970   -0.009670   -0.006911
released_day             0.001970    1.000000    0.079437    0.174095
released_month          -0.009670    0.079437    1.000000    0.076801
released_year           -0.006911    0.174095    0.076801    1.000000
speechiness_%           -0.021367   -0.015625    0.040163    0.134397
streams                 -0.049418    0.011319   -0.022795   -0.226132
valence_%                0.020793    0.041789   -0.118139   -0.059631

                      speechiness_%   streams  valence_%
acousticness_%            -0.023848 -0.004561  -0.082006
artist_count               0.119011 -0.136456   0.128441
bpm                        0.040275 -0.002054   0.041316
danceability_%             0.185581 -0.104962   0.408211
energy_%                  -0.004306 -0.026083   0.358206
in_apple_charts           -0.152129  0.321526   0.048571
in_apple_playlists        -0.108587  0.773518   0.055200
in_deezer_charts          -0.080570  0.228564   0.073628
in_deezer_playlists       -0.062743  0.598337  -0.013916
in_shazam_charts          -0.147007  0.335569   0.054106
in_spotify_charts         -0.082874  0.246172   0.035867
in_spotify_playlists      -0.090188  0.790053  -0.022439
instrumentalness_%        -0.083158 -0.044053  -0.133835
key                        0.036263 -0.048590   0.031472
liveness_%                -0.021367 -0.049418   0.020793
released_day              -0.015625  0.011319   0.041789
released_month             0.040163 -0.022795  -0.118139
released_year              0.134397 -0.226132  -0.059631
speechiness_%              1.000000 -0.112298   0.041048
streams                   -0.112298  1.000000  -0.042169
valence_%                  0.041048 -0.042169   1.000000

[21 rows x 21 columns]
```

[ ]:

```python
correlation_matrix = df_selected.corr()

high_correlation_matrix = correlation_matrix[(correlation_matrix.abs() > 0.7) &
 ↪(correlation_matrix.abs() < 1)]
high_correlations = (correlation_matrix.abs() >= 0.7) & (correlation_matrix.
 ↪abs() < 1)
```

```python
indices = [(i, j) for i in range(correlation_matrix.shape[0]) for j in range(correlation_matrix.shape[1]) if high_correlations.iloc[i, j]]

print("Indices of correlations greater than or equal to 0.7 or less than or equal to -0.7:")
print(indices)
```

```
Indices of correlations greater than or equal to 0.7 or less than or equal to
-0.7:
[(5, 9), (6, 11), (6, 19), (8, 11), (9, 5), (9, 10), (10, 9), (11, 6), (11, 8),
(11, 19), (19, 6), (19, 11)]
```

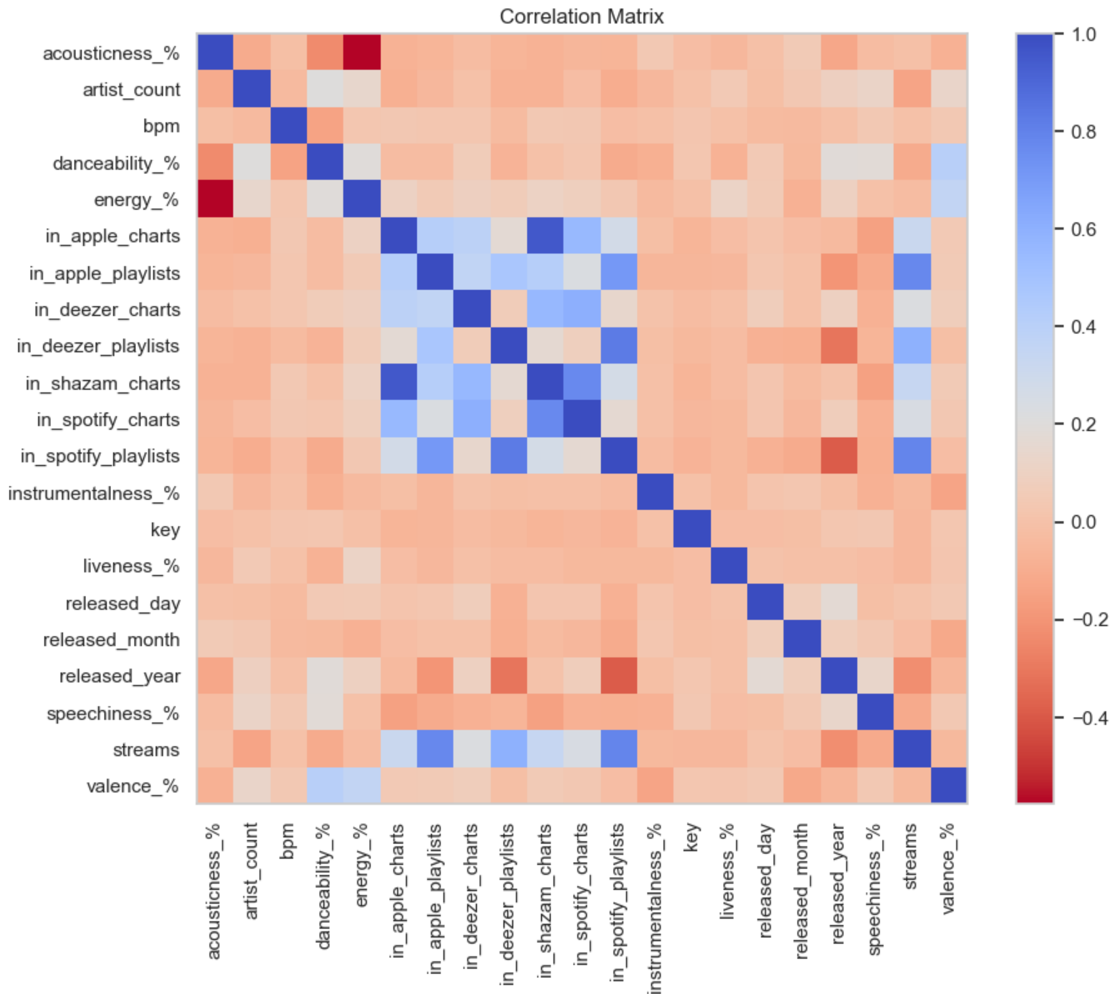
[ ]:
```python
non_nan_columns = high_correlation_matrix.dropna(axis=1, how='all').dropna(axis=0, how='all')
non_nan_columns = non_nan_columns.fillna('')

non_nan_columns
```

[ ]:

	in_apple_charts	in_apple_playlists	in_deezer_playlists	in_shazam_charts	in_spotify_charts	in_spotify_playlists	streams
in_apple_charts				0.955245			
in_apple_playlists						0.708634	0.773518
in_deezer_playlists						0.826524	
in_shazam_charts	0.955245				0.766718		
in_spotify_charts				0.766718			
in_spotify_playlists		0.708634	0.826524				0.790053
streams		0.773518				0.790053	

Heatmap of Correlation Matrix

[ ]:
```
plt.figure(figsize=(10, 8))
plt.imshow(correlation_matrix, cmap='coolwarm_r', interpolation='nearest')
plt.colorbar()
plt.title('Correlation Matrix')
plt.xticks(range(len(correlation_matrix.columns)), correlation_matrix.columns,␣
 ↪rotation=90)
plt.yticks(range(len(correlation_matrix.columns)), correlation_matrix.columns)
plt.grid(False)
plt.show()
```



Queries

Query 1: Density Distribution of Release Date of Songs

[ ]:
```
df_with_datetime = df_copy.copy()
```

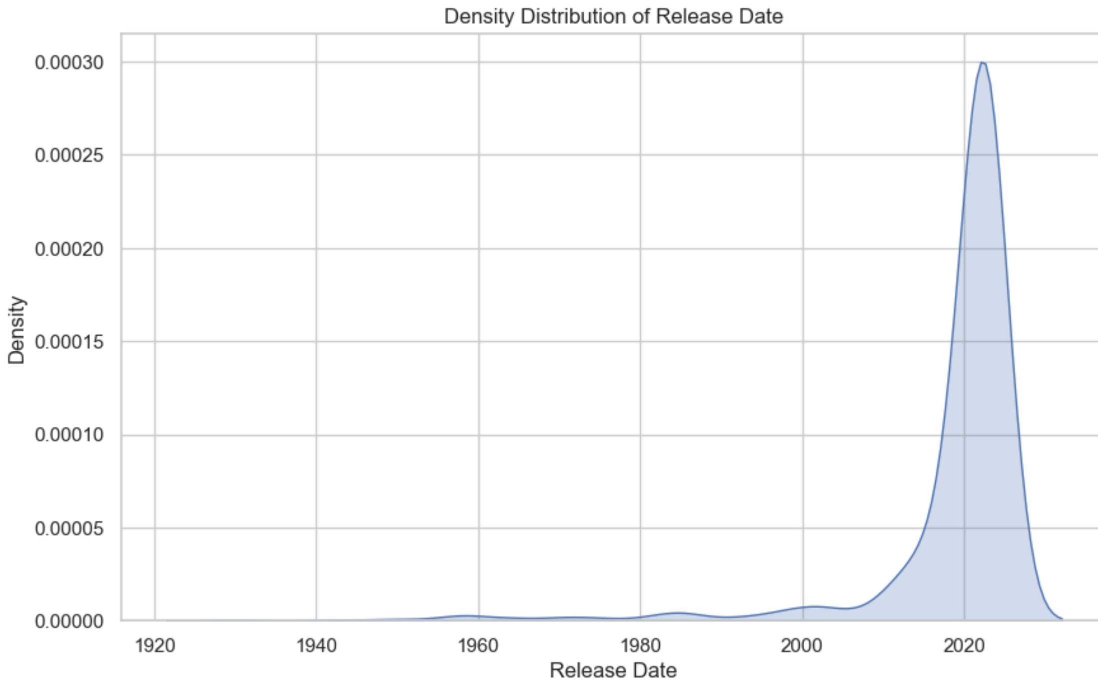
```
df_with_datetime['release_date'] = pd.
 ↪to_datetime(df_with_datetime[['released_year', 'released_month',␣
 ↪'released_day']].astype(str).agg('-'.join, axis=1), errors='coerce')
df_with_datetime[['track_name','release_date']]
```

[ ]:

```
                              track_name release_date
0    Seven (feat. Latto) (Explicit Ver.)   2023-07-14
1                                   LALA   2023-03-23
2                                vampire   2023-06-30
3                           Cruel Summer   2019-08-23
4                         WHERE SHE GOES   2023-05-18
..                                     …            …
944                           My Mind & Me   2022-11-03
945            Bigger Than The Whole Sky   2022-10-21
946                  A Veces (feat. Feid)   2022-11-03
947                         En La De Ella   2022-10-20
948                                  Alone   2022-11-04

[949 rows x 2 columns]
```

[ ]:

```
#Kernel Density Estimate

sns.set(style="whitegrid")

plt.figure(figsize=(10, 6))
sns.kdeplot(df_with_datetime['release_date'], fill=True)

plt.xlabel('Release Date')
plt.ylabel('Density')
plt.title('Density Distribution of Release Date')

plt.show()
```



The above graph gives the distribution of Release Date of Songs which became viral in 2023.

Newer songs tend to be streamed more, as shown in the graph, but older songs that dates back up to 1930 received a lot of streaming too.

The older songs that received a lot of streaming could be classic songs, or songs that were brought back to popularity due to some usage in social media.

Splitting the track for multiple artists

```
[ ]: df_split_artists = df_copy.assign(artists=df_copy['artists'].str.split(',')).
      ↪explode('artists')
     df_split_artists = df_split_artists.drop_duplicates(subset=['artists',␣
      ↪'track_name'])
     df_split_artists = df_split_artists.apply(lambda x: x.str.strip() if x.dtype ==␣
      ↪"O" else x)
     df_split_artists.reset_index(drop=True, inplace=True)
     df_split_artists.to_csv('artist_split.csv')
     df_split_artists
```

```
[ ]:                          track_name        artists  artist_count  \
     0     Seven (feat. Latto) (Explicit Ver.)          Latto             2
     1     Seven (feat. Latto) (Explicit Ver.)      Jung Kook             2
     2                                   LALA    Myke Towers             1
     3                                vampire  Olivia Rodrigo             1
     4                           Cruel Summer    Taylor Swift             1
```

```
…                                 …            …           …
1472                A Veces (feat. Feid)    Paulo Londra            2
1473                        En La De Ella           Feid            3
1474                        En La De Ella           Sech            3
1475                        En La De Ella         Jhayco            3
1476                                Alone      Burna Boy            1

      released_year  released_month  released_day  in_spotify_playlists  \
0              2023               7            14                   553
1              2023               7            14                   553
2              2023               3            23                  1474
3              2023               6            30                  1397
4              2019               8            23                  7858
…               …              …            …                    …
1472           2022              11             3                   573
1473           2022              10            20                  1320
1474           2022              10            20                  1320
1475           2022              10            20                  1320
1476           2022              11             4                   782

      in_spotify_charts    streams  in_apple_playlists  …  bpm  key   mode  \
0                   147  141381703                  43  …  125   11  Major
1                   147  141381703                  43  …  125   11  Major
2                    48  133716286                  48  …   92    1  Major
3                   113  140003974                  94  …  138    5  Major
4                   100  800840817                 116  …  170    9  Major
…                   …          …                 …  …   …  …    …
1472                  0   73513683                   2  …   92    1  Major
1473                  0  133895612                  29  …   97    1  Major
1474                  0  133895612                  29  …   97    1  Major
1475                  0  133895612                  29  …   97    1  Major
1476                  2   96007391                  27  …   90    4  Minor

      danceability_%  valence_%  energy_% acousticness_%  instrumentalness_%  \
0                 80         89        83             31                   0
1                 80         89        83             31                   0
2                 71         61        74              7                   0
3                 51         32        53             17                   0
4                 55         58        72             11                   0
…                …          …        …              …                  …
1472              80         81        67              4                   0
1473              82         67        77              8                   0
1474              82         67        77              8                   0
1475              82         67        77              8                   0
1476              61         32        67             15                   0

      liveness_%  speechiness_%
```

```
0             8             4
1             8             4
2            10             4
3            31             6
4            11            15
…          …          …
1472          8             6
1473         12             5
1474         12             5
1475         12             5
1476         11             5

[1477 rows x 24 columns]
```

To get the involvement of each artist in each track separately for the following analysis purposes, we split a track where multiple artists are present into new rows with single artist present in each

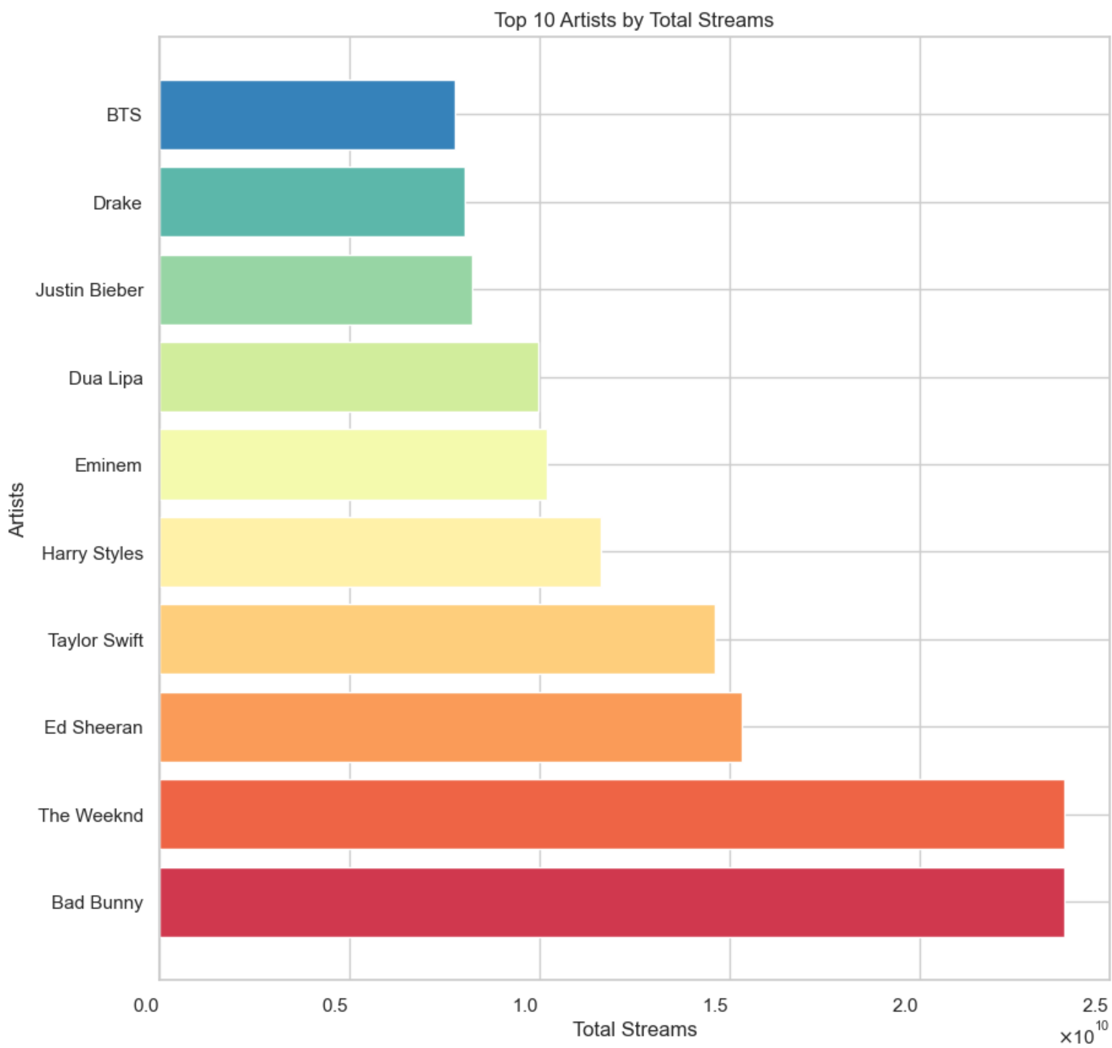
Query 2: Most Streamed Artists of 2023

[ ]:
```python
artist_streams = df_split_artists.groupby('artists')['streams'].sum()
artist_streams = artist_streams.sort_values(ascending=False)
df_artist_streams = pd.DataFrame(artist_streams)
df_artist_streams
```

[ ]:
```
                    streams
artists
Bad Bunny       23813527270
The Weeknd      23799104954
Ed Sheeran      15316587718
Taylor Swift    14630378183
Harry Styles    11608645649
…                         …
Toian              32761689
Beam               32761689
DJ 900             11956641
Sog                11599388
Sukriti Kakar       1365184

[699 rows x 1 columns]
```

[ ]:
```python
plt.figure(figsize=(10,10))
data = df_artist_streams.head(10)

colors = sns.color_palette("Spectral", n_colors=10)

plt.barh(data.index, data['streams'], color = colors)

plt.xticks(ha='right')
```

```
plt.gca().xaxis.set_major_formatter(ticker.ScalarFormatter(useMathText=True))


plt.ylabel('Artists')
plt.xlabel('Total Streams')
plt.title('Top 10 Artists by Total Streams')

plt.show()
```



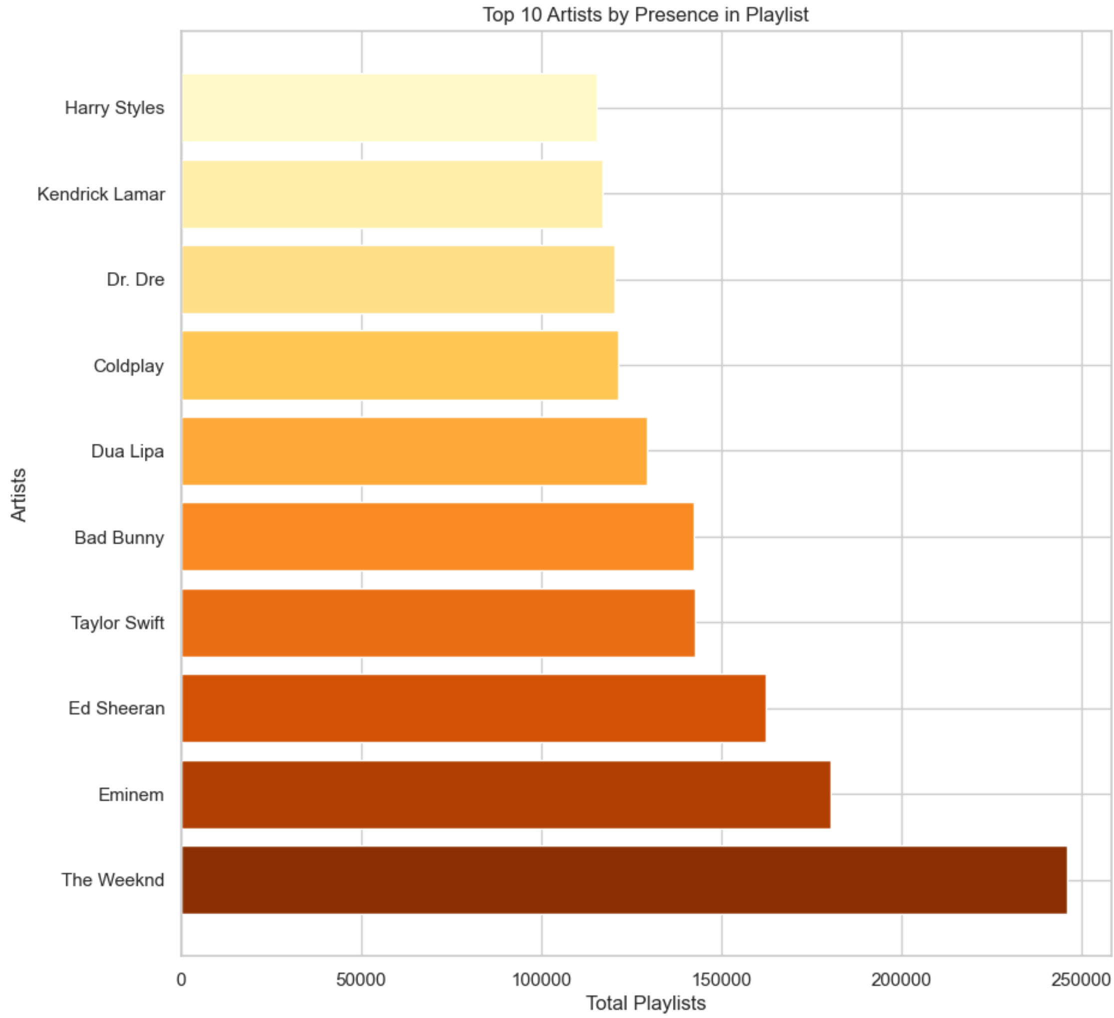
Bad Bunny is the most streamed artist of 2023, with 23813527270 (23.8 Billion) streams, followed closely by The Weeknd with 23799104954 (23.7 Billion) streams

Query 3: Artists present in most playlists

[ ]:
```python
artist_playlists = df_split_artists.groupby('artists')[['in_spotify_playlists',␣
↪'in_apple_playlists', 'in_deezer_playlists']].sum()

artist_playlists = artist_playlists.sum(axis=1).sort_values(ascending=False)

df_artist_playlists = pd.DataFrame({'Total_Playlists': artist_playlists})
df_artist_playlists
```

[ ]:
```
               Total_Playlists
artists
The Weeknd              245924
Eminem                  180355
Ed Sheeran              162567
Taylor Swift            142855
Bad Bunny               142461
…                          …
Sukriti Kakar              153
Mahalini                   138
Colde                      115
Shubh                       74
Jack Black                  34

[699 rows x 1 columns]
```

[ ]:
```python
plt.figure(figsize=(10, 10))

colors = sns.color_palette("YlOrBr_r", n_colors=10)

data = df_artist_playlists.head(10)

plt.barh(data.index, data['Total_Playlists'], color = colors)
plt.xlabel('Total Playlists')
plt.ylabel('Artists')
plt.title('Top 10 Artists by Presence in Playlist')
plt.show()
```



Even though Bad Bunny is the most streamed artist, The Weeknd is the artist present in most playlists. This indicates that The Weeknd has songs with higher repeat value and cover a wider genre

Query 4: Average Attributes of Songs of Top 15 Artists

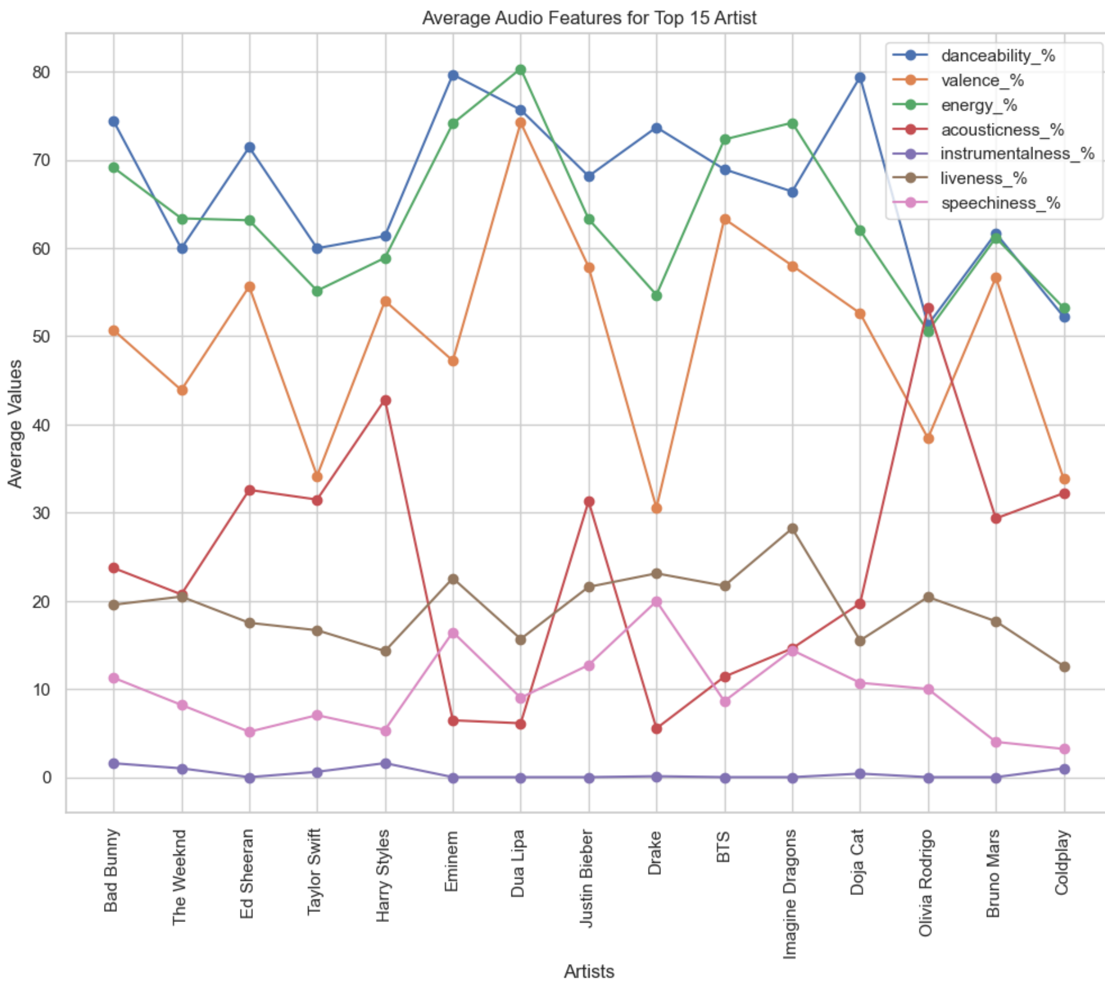
[ ]:
```
top_artists = list(df_artist_streams.reset_index().nlargest(15,␣
↪'streams')['artists'])

artists_data = df_split_artists[df_split_artists['artists'].isin(top_artists)].
↪copy()
artists_data['artists'] = pd.Categorical(artists_data['artists'],␣
↪categories=top_artists, ordered=True)
```

```
average_attributes = artists_data.groupby('artists',␣
↪observed=False)[['danceability_%', 'valence_%', 'energy_%',␣
↪'acousticness_%', 'instrumentalness_%', 'liveness_%', 'speechiness_%']].
↪mean()
average_attributes.sort_index()
```

[ ]:
```
                 danceability_%  valence_%   energy_%  acousticness_%  \
artists
Bad Bunny             74.425000  50.700000  69.125000       23.725000
The Weeknd            59.944444  43.888889  63.361111       20.722222
Ed Sheeran            71.428571  55.642857  63.142857       32.571429
Taylor Swift          59.973684  34.157895  55.157895       31.473684
Harry Styles          61.352941  54.000000  58.882353       42.823529
Eminem                79.666667  47.222222  74.111111        6.444444
Dua Lipa              75.666667  74.222222  80.333333        6.111111
Justin Bieber         68.142857  57.857143  63.285714       31.285714
Drake                 73.684211  30.526316  54.684211        5.526316
BTS                   68.923077  63.307692  72.307692       11.384615
Imagine Dragons       66.400000  58.000000  74.200000       14.600000
Doja Cat              79.400000  52.600000  62.000000       19.700000
Olivia Rodrigo        51.285714  38.428571  50.571429       53.285714
Bruno Mars            61.666667  56.666667  61.166667       29.333333
Coldplay              52.200000  33.800000  53.200000       32.200000

                 instrumentalness_%  liveness_%  speechiness_%
artists
Bad Bunny                  1.575000   19.550000      11.275000
The Weeknd                 1.000000   20.472222       8.194444
Ed Sheeran                 0.000000   17.500000       5.142857
Taylor Swift               0.605263   16.657895       7.026316
Harry Styles               1.588235   14.294118       5.352941
Eminem                     0.000000   22.555556      16.444444
Dua Lipa                   0.000000   15.666667       9.000000
Justin Bieber              0.000000   21.571429      12.714286
Drake                      0.105263   23.105263      19.947368
BTS                        0.000000   21.692308       8.615385
Imagine Dragons            0.000000   28.200000      14.400000
Doja Cat                   0.400000   15.500000      10.700000
Olivia Rodrigo             0.000000   20.428571      10.000000
Bruno Mars                 0.000000   17.666667       4.000000
Coldplay                   1.000000   12.600000       3.200000
```

[ ]:
```
average_attributes.plot(kind='line', marker='o', figsize=(12, 9))

plt.xlabel('Artists')
plt.ylabel('Average Values')
plt.title('Average Audio Features for Top 15 Artist')
```

```
plt.xticks(rotation = 90)
plt.xticks(range(len(average_attributes.index)), average_attributes.index)

plt.show()
```



The Top 15 Artists follow similar patterns for instrumentalness

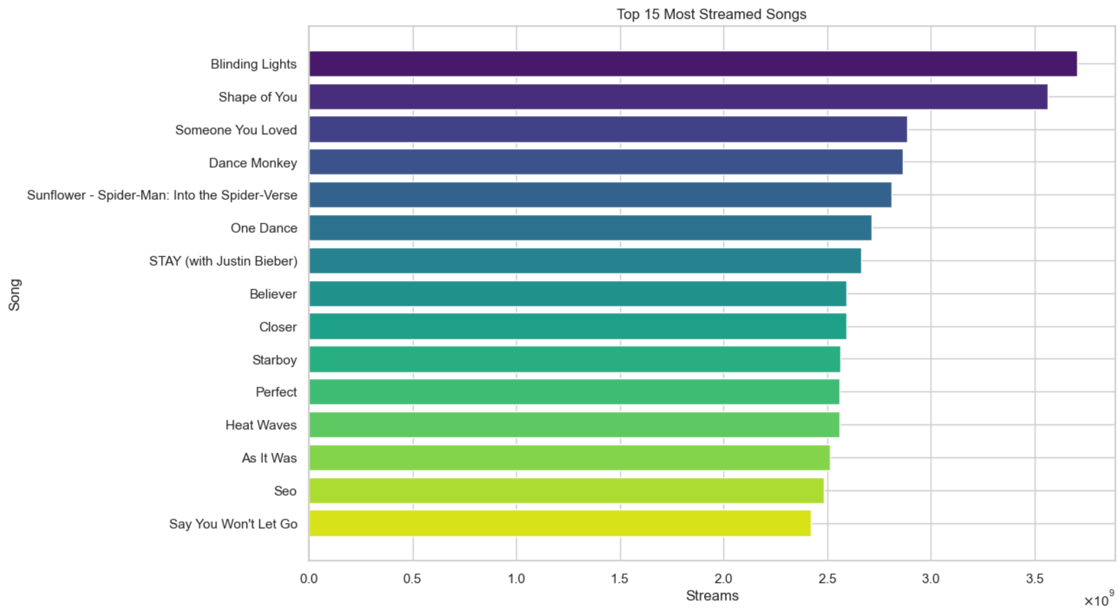
Query 5: Most Streamed Songs of 2023

[ ]:
```
top_songs_by_streams = df_copy.nlargest(15, 'streams')
top_songs_by_streams[['track_name', 'artists', 'streams']]
```

[ ]:
```
                                          track_name  \
55                                   Blinding Lights
179                                     Shape of You
86                                 Someone You Loved
618                                     Dance Monkey
```

```
41   Sunflower - Spider-Man: Into the Spider-Verse
162                                      One Dance
84                      STAY (with Justin Bieber)
140                                       Believer
723                                         Closer
48                                         Starboy
138                                        Perfect
71                                      Heat Waves
14                                       As It Was
691                                            Seo
324                              Say You Won't Let Go

                          artists     streams
55                     The Weeknd  3703895074
179                    Ed Sheeran  3562543890
86                  Lewis Capaldi  2887241814
618                   Tones and I  2864791672
41          Post Malone, Swae Lee  2808096550
162           Drake, WizKid, Kyla  2713922350
84   Justin Bieber, The Kid Laroi  2665343922
140               Imagine Dragons  2594040133
723      The Chainsmokers, Halsey  2591224264
48          The Weeknd, Daft Punk  2565529693
138                    Ed Sheeran  2559529074
71                  Glass Animals  2557975762
14                   Harry Styles  2513188493
691  Shawn Mendes, Camila Cabello  2484812918
324                  James Arthur  2420461338
```

[ ]:
```
plt.figure(figsize=(12, 8))

colors = sns.color_palette("viridis", n_colors=15)

plt.barh(top_songs_by_streams['track_name'], top_songs_by_streams['streams'],␣
 ↪color = colors)

plt.gca().xaxis.set_major_formatter(ticker.ScalarFormatter(useMathText=True))

plt.xlabel('Streams')
plt.ylabel('Song')
plt.title('Top 15 Most Streamed Songs')
plt.gca().invert_yaxis()
plt.show()
```



Blinding Lights by The Weeknd is the most streamed song of 2023 with 3703895074 (3.7 Billion) streams, followed by Shape of You by Ed Sheeran with 3562543890 (3.5 Billion) streams

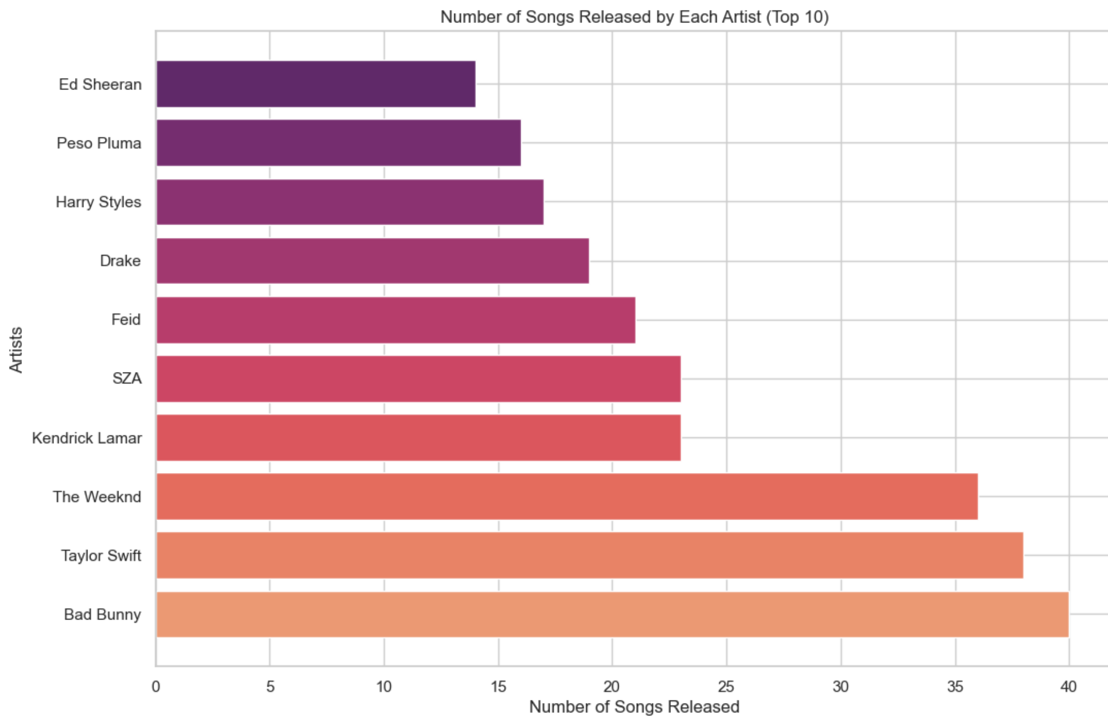
Query 6: Number of Songs Released by Each Artist

[ ]:
```
most_songs_artists = df_split_artists['artists'].value_counts()
most_songs_artists = pd.DataFrame(most_songs_artists)
most_songs_artists
```

[ ]:
```
                 count
artists
Bad Bunny           40
Taylor Swift        38
The Weeknd          36
Kendrick Lamar      23
SZA                 23
…                 …
La Joaqui            1
Steve Aoki           1
FIFA Sound           1
Beach House          1
Selena Gomez         1

[699 rows x 1 columns]
```

[ ]:
```
plt.figure(figsize=(12, 8))

colors = sns.color_palette("flare", n_colors=10)
```

```
data = most_songs_artists.head(10)

plt.barh(data.index, data['count'], color=colors)

plt.xlabel('Number of Songs Released')
plt.ylabel('Artists')
plt.title('Number of Songs Released by Each Artist (Top 10)')
plt.show()
```



- Bad Bunny is the artist whose songs became most popular in 2023 with 40 songs, followed by Taylor Swift with 38

Query 7: Top Songs by Playlist

[ ]:
```
N = 10

top_spotify_songs = df_copy.nlargest(N, 'in_spotify_playlists')
top_apple_songs = df_copy.nlargest(N, 'in_apple_playlists')
top_deezer_songs = df_copy.nlargest(N, 'in_deezer_playlists')

print("Top Spotify Songs:")
print(top_spotify_songs[['track_name', 'in_spotify_playlists']].
 ↪to_string(index=False))
```

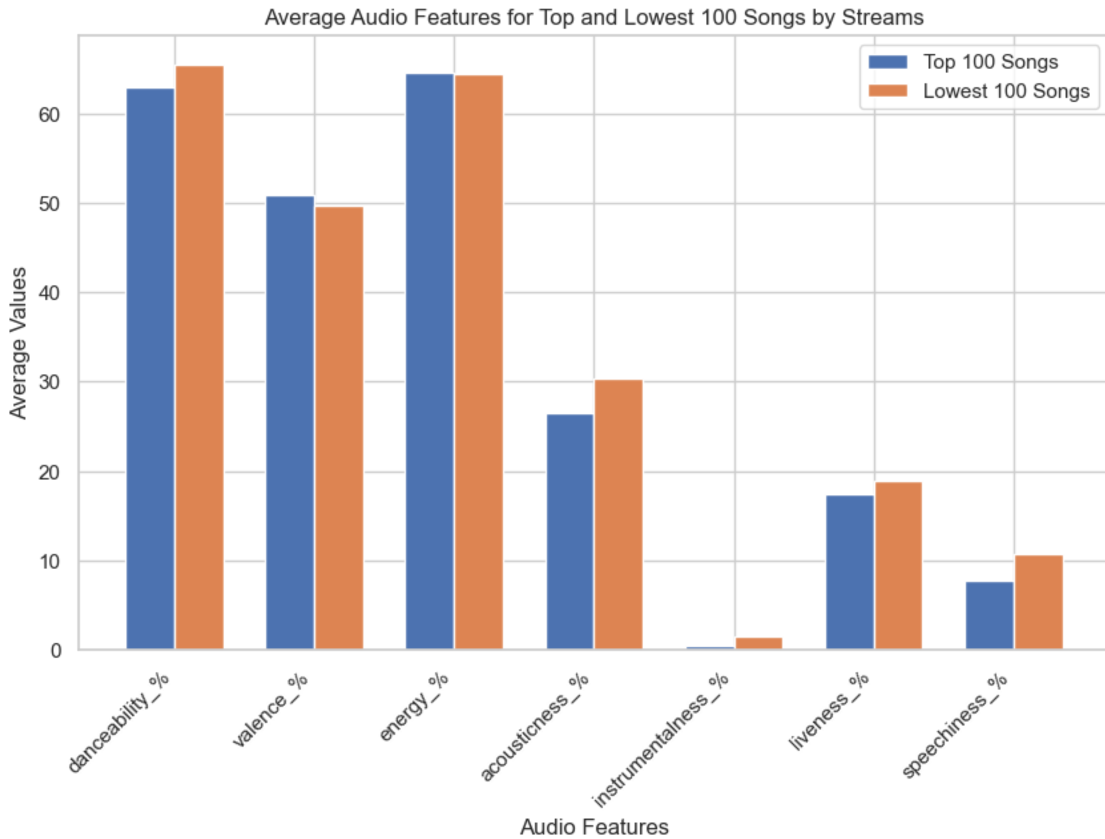
```
print("\nTop Apple Music Songs:")
print(top_apple_songs[['track_name', 'in_apple_playlists']].
 ↪to_string(index=False))

print("\nTop Deezer Songs:")
print(top_deezer_songs[['track_name', 'in_deezer_playlists']].
 ↪to_string(index=False))
```

```
Top Spotify Songs:
                              track_name  in_spotify_playlists
                   Get Lucky - Radio Edit                 52898
                           Mr. Brightside                 51979
                    Wake Me Up - Radio Edit               50887
Smells Like Teen Spirit - Remastered 2021                 49991
                               Take On Me                 44927
                          Blinding Lights                 43899
                                One Dance                 43257
              Somebody That I Used To Know                42798
       Everybody Wants To Rule The World                  41751
                        Sweet Child O' Mine               41231

Top Apple Music Songs:
               track_name  in_apple_playlists
          Blinding Lights                 672
 One Kiss (with Dua Lipa)                 537
             Dance Monkey                 533
          Don't Start Now                 532
STAY (with Justin Bieber)                 492
                      Seo                 453
        Someone You Loved                 440
         Watermelon Sugar                 437
                One Dance                 433
               As It Was                  403

Top Deezer Songs:
                              track_name  in_deezer_playlists
Smells Like Teen Spirit - Remastered 2021                12367
                   Get Lucky - Radio Edit                 8215
                           The Scientist                  7827
                                    Numb                  7341
                            Shape of You                  6808
                              In The End                  6808
                                   Creep                  6807
                      Sweet Child O' Mine                 6720
                            Still D.R.E.                  6591
         Can't Hold Us (feat. Ray Dalton)                 6551
```

- Blinding lights, One Dance, Get Lucky - Radio Edit are the tracks that are in more than 1 platforms playlist

Query 8: Average Audio Features for Top and Lowest 10 Songs by Streams

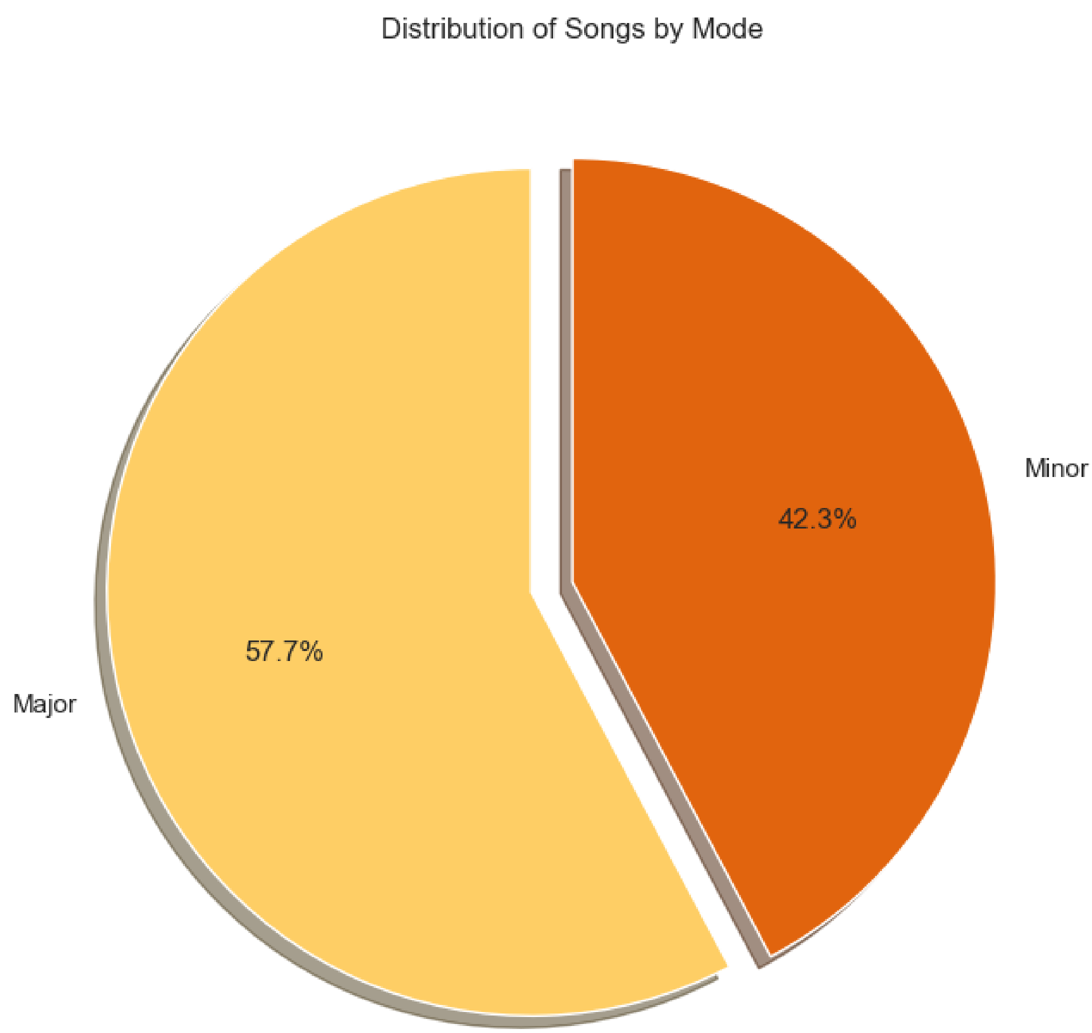
[ ]:
```
top_100_songs = df_copy.nlargest(100, 'streams')
lowest_100_songs = df_copy.nsmallest(100, 'streams')

columns_to_average = ['danceability_%', 'valence_%', 'energy_%',␣
 ↪'acousticness_%', 'instrumentalness_%', 'liveness_%', 'speechiness_%']

top_100_average = top_100_songs[columns_to_average].mean()
lowest_100_average = lowest_100_songs[columns_to_average].mean()

fig, ax = plt.subplots(figsize=(10, 6))

bar_width = 0.35

bar_positions_top = range(len(top_100_average))
bar_positions_lowest = [pos + bar_width for pos in bar_positions_top]

ax.bar(bar_positions_top, top_100_average, width=bar_width, label='Top 100␣
 ↪Songs')
ax.bar(bar_positions_lowest, lowest_100_average, width=bar_width, label='Lowest␣
 ↪100 Songs')

ax.set_xlabel('Audio Features')
ax.set_ylabel('Average Values')
ax.set_title('Average Audio Features for Top and Lowest 100 Songs by Streams')
ax.set_xticks([pos + bar_width / 2 for pos in bar_positions_top])
ax.set_xticklabels(columns_to_average, rotation=45, ha='right')
ax.legend()

plt.show()
```



Listeners prefer to stream tracks that consists of more singing than acoustincess, speech and liveness.

Query 9: Distribution of Songs by Mode

[ ]:
```
# Assuming df is your DataFrame
mode_counts = df_copy['mode'].value_counts()

# Define a bright color palette
colors = sns.color_palette('YlOrBr', n_colors=len(mode_counts))

# Explode a slice to highlight
explode = [0.1] + [0] * (len(mode_counts) - 1)  # explode the first slice

# Plot a pie chart with styling
plt.figure(figsize=(8, 8))
plt.pie(mode_counts, labels=mode_counts.index, autopct='%1.1f%%', startangle=90,
        colors=colors, explode=explode, shadow=True)
plt.title('Distribution of Songs by Mode')
plt.show()
```

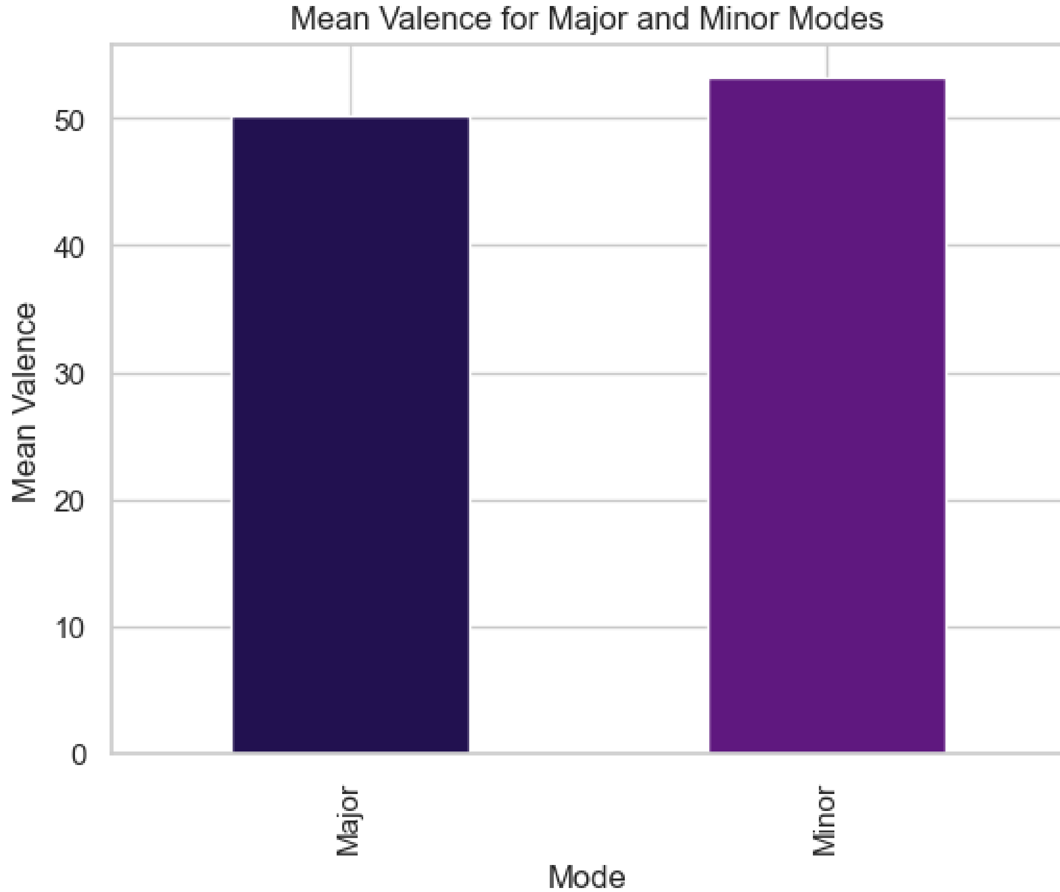


Most of the songs are compossed on Major mode

Query 10: Mean Valence for Major and Minor Modes

[ ]:
```
mode_valence_means = df_copy.groupby('mode')['valence_%'].mean()

colors = sns.color_palette("magma")

mode_valence_means.plot(kind='bar', color = colors)

plt.xlabel('Mode')
plt.ylabel('Mean Valence')
plt.title('Mean Valence for Major and Minor Modes')
```

```
plt.show()
```



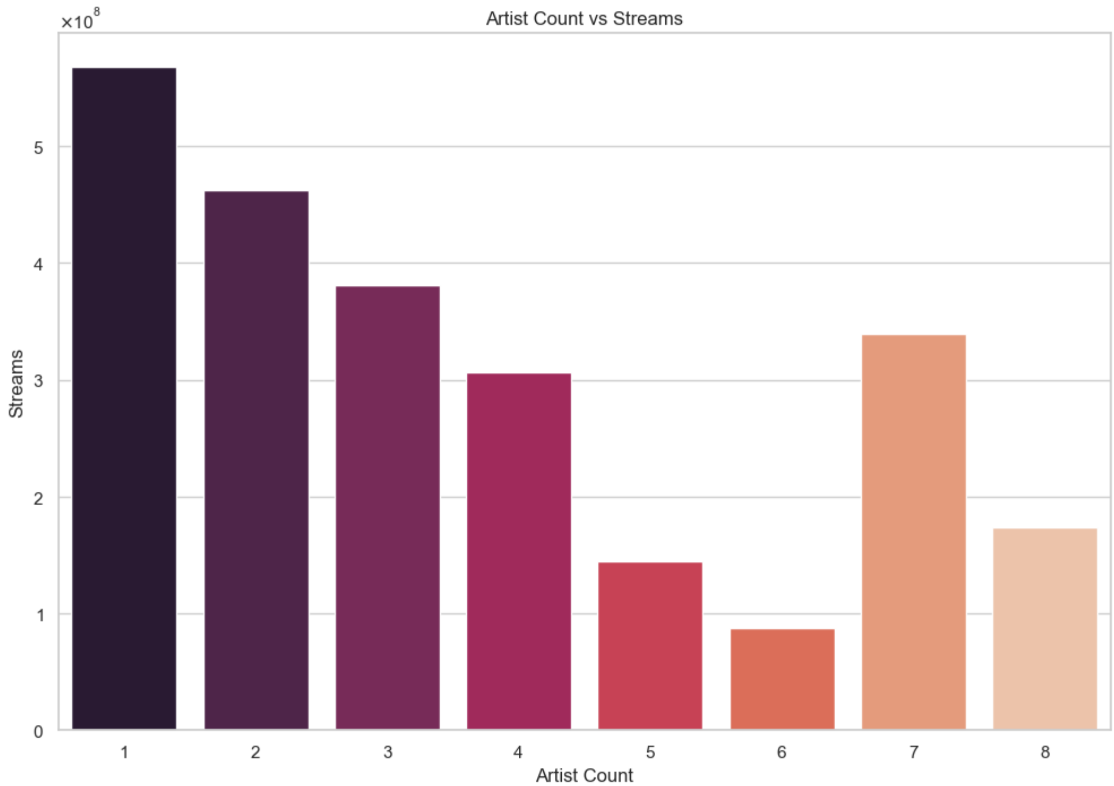
Assumption: It is commonly believed that songs in major mode are generally more positive.

Observation: Upon analyzing the graph, we found that songs in the minor mode exhibit a slightly higher level of valence. This contradicts the initial assumption.

Query 11: Single Artist v/s Multiple Artists

[ ]:
```
df_artist_count = df_copy.groupby('artist_count')['streams'].mean().
 ↪reset_index()

colors = sns.color_palette("rocket", n_colors=len(df_artist_count))


plt.figure(figsize=(12, 8))
sns.barplot(x='artist_count', y='streams', data=df_artist_count,␣
 ↪palette=colors, hue='artist_count', legend=False)
```

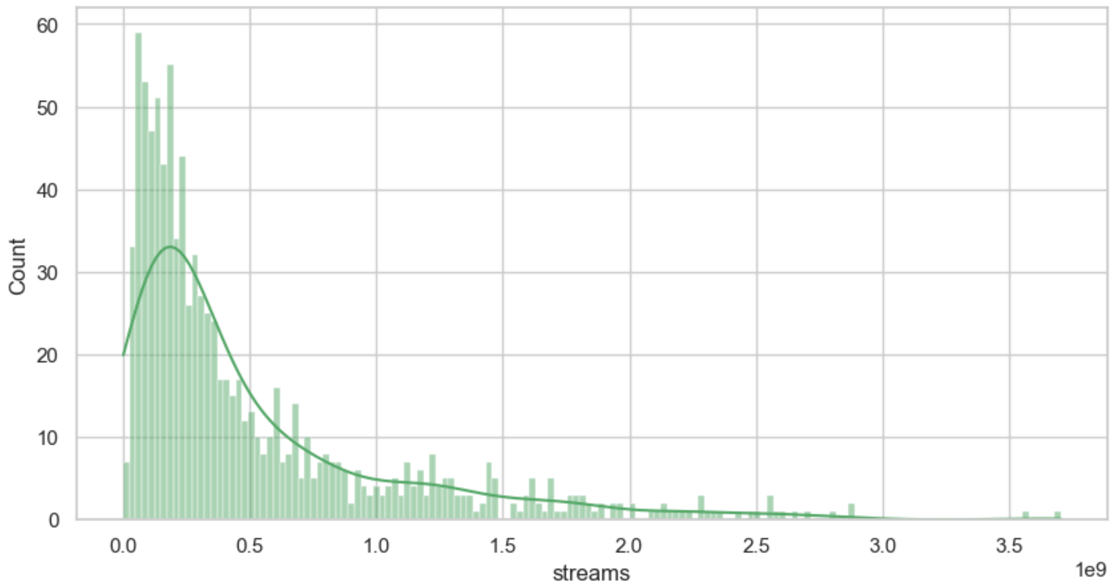
```
plt.gca().yaxis.set_major_formatter(ticker.ScalarFormatter(useMathText=True))

plt.xlabel('Artist Count')
plt.ylabel('Streams')
plt.title('Artisit Count vs Streams')

plt.show()
```



Tracks with only 1 artist seem to be more popular and streamed more

Query 12: Visualisation of Streams and Count

[ ]:
```
plt.figure(figsize=(10, 5))

sns.histplot(data=df_copy['streams'], color='g',bins=150, kde=True)

plt.show()
```



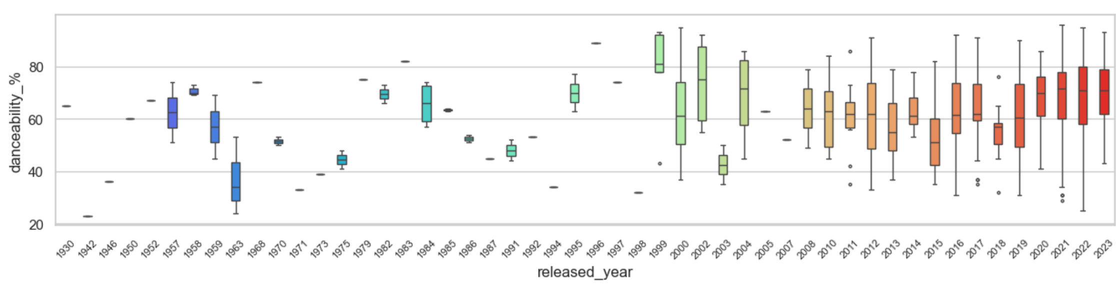
- Most of the songs have stream less than 0.5 Billion
- Songs with streams above 2.5 Billion are very rare

Query 13: Song Properties Throughout The Years

Danceability %

[ ]:
```
plt.subplots(1,1, figsize=(15,3))
colors = sns.color_palette("rainbow",n_colors=50)
sns.boxplot(data=df_copy, x='released_year',y='danceability_%', width=0.
 ↪4,fliersize=2,palette=colors ,hue='released_year',legend=False)
plt.xticks(rotation=45,fontsize=8)
plt.show
```

[ ]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```



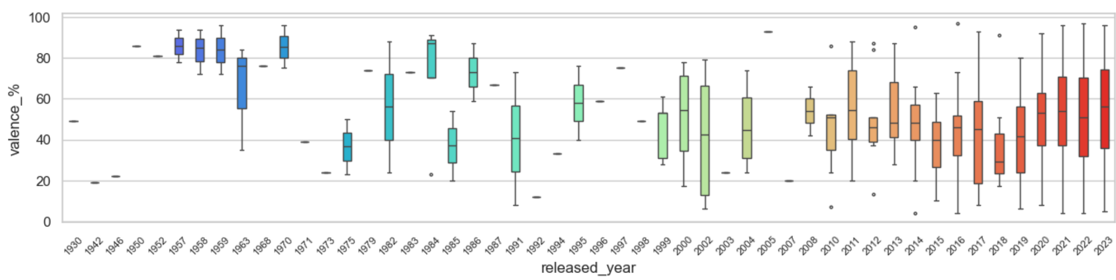
- Tracks that are released from the year 2020 to 2023 have almost similar median danceability, and almost similar interquartile range.

- Tracks that are released from 2008 to 2023 have wide range of danceability. It could be due to the majority of the top tracks were released in these years.
- The most danceable track in the top streamed songs was released in 2021.
- The least danceable track in the top streamed songs was released in 1942.

Valence %

[ ]:
```
plt.subplots(1,1, figsize=(15,3))
colors = sns.color_palette("rainbow",n_colors=50)
sns.boxplot(data=df_copy, x='released_year',y='valence_%', width=0.
 ↪4,fliersize=2,palette=colors ,hue='released_year',legend=False)
plt.xticks(rotation=45,fontsize=8)
plt.show
```

[ ]: <function matplotlib.pyplot.show(close=None, block=None)>

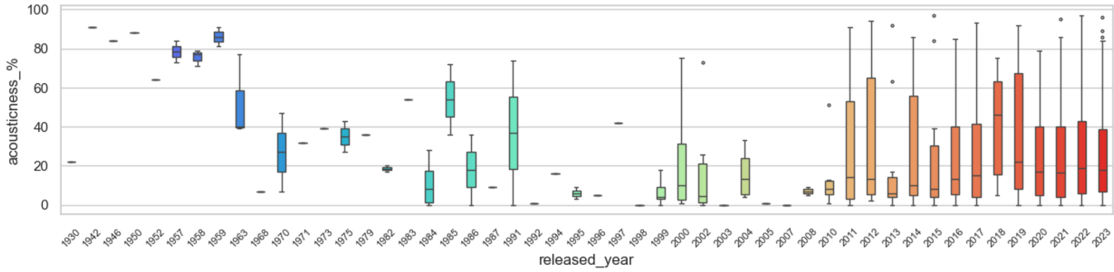


- Tracks from 2020 to 2023 shows wide variety of moods in the top streamed songs, long whiskers extending from low valence to high valence, and median values at approximately 50%.
- Tracks from 2011 to 2023 has median valence at approximately 40 to 50%, with the exception of 2018. The range is also at the middle of the chart, ranging 20 to 70%, which shows the neutrality of the mood in the top streamed songs.

Acousticness %

[ ]:
```
plt.subplots(1,1, figsize=(15,3))
colors = sns.color_palette("rainbow",n_colors=50)
sns.boxplot(data=df_copy, x='released_year',y='acousticness_%', width=0.
 ↪4,fliersize=2,palette=colors ,hue='released_year',legend=False)
plt.xticks(rotation=45,fontsize=8)
plt.show
```

[ ]: <function matplotlib.pyplot.show(close=None, block=None)>

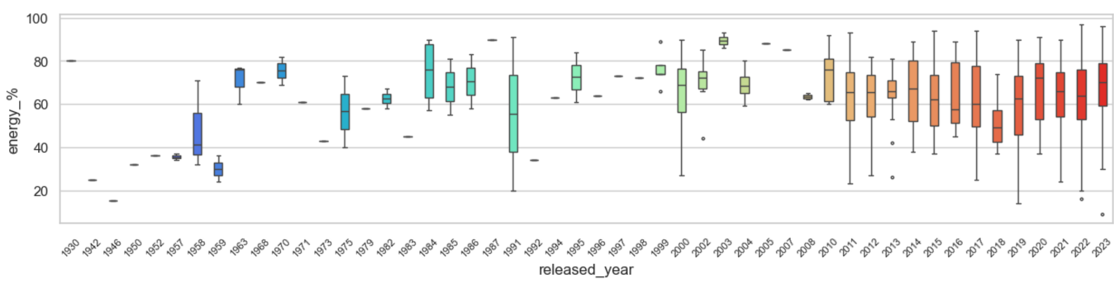


- Tracks from 2011 to 2023 contains high variety of songs with different acousticness values, as shown in the long whiskers and long interquartile range.
- Older tracks seem to fall under a small range of acousticness levels.

Energy %

[ ]:
```
plt.subplots(1,1, figsize=(15,3))
colors = sns.color_palette("rainbow",n_colors=50)
sns.boxplot(data=df_copy, x='released_year',y='energy_%', width=0.
↪4,fliersize=2,palette=colors ,hue='released_year',legend=False)
plt.xticks(rotation=45,fontsize=8)
plt.show
```

[ ]: <function matplotlib.pyplot.show(close=None, block=None)>



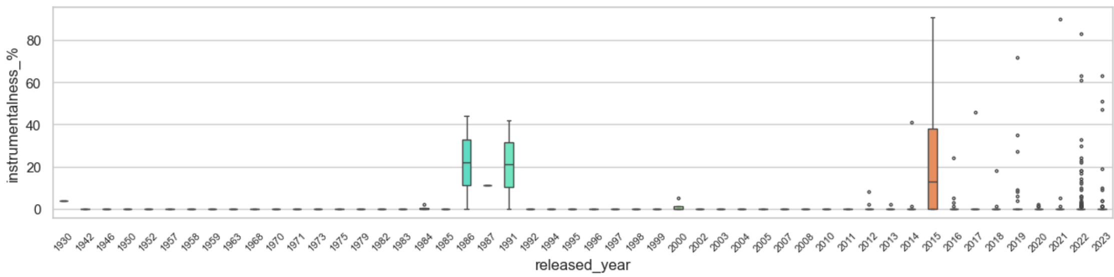
- Top tracks from 2011 to 2023 contains a wide range of tracks from energetic to less energetic, but with median values at the 50 to 60% energy percentage.
- The median values of the most streamed tracks are 50% or more, with the exception of 1958, 1959, and years with single tracks that made it to the most streamed. This shows that listeners prefer to listen to energetic tracks.

Instrumentalness %

[ ]:
```
plt.subplots(1,1, figsize=(15,3))
colors = sns.color_palette("rainbow",n_colors=50)
sns.boxplot(data=df_copy, x='released_year',y='instrumentalness_%', width=0.
↪4,fliersize=2,palette=colors ,hue='released_year',legend=False)
```

```
plt.xticks(rotation=45,fontsize=8)
plt.show
```

[ ]: <function matplotlib.pyplot.show(close=None, block=None)>

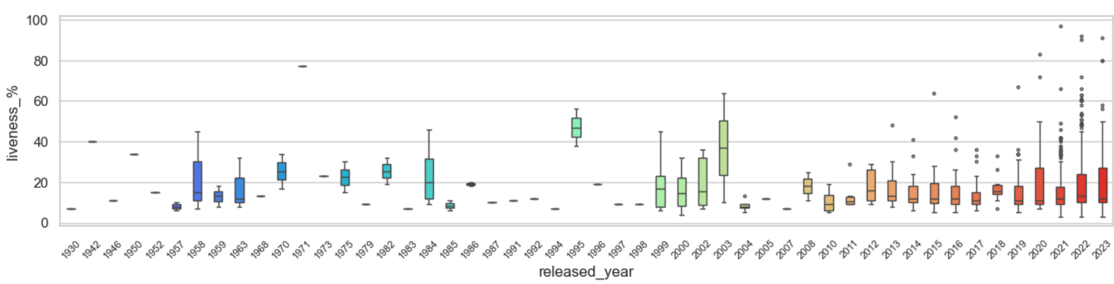


- The boxplot shows that majority of the most streamed tracks contains less instrumentalness levels, with some tracks that falls under the instrumental category identifies as outliers.
- Tracks released from 1986, 1991, and 2015, however, contains tracks that have considerably high instrumentalness levels, especially in 2015 where the boxplot whiskers reached the highest instrumentalness level.

Liveness %

[ ]:
```
plt.subplots(1,1, figsize=(15,3))
colors = sns.color_palette("rainbow",n_colors=50)
sns.boxplot(data=df_copy, x='released_year',y='liveness_%', width=0.
 ↪4,fliersize=2,palette=colors ,hue='released_year',legend=False)
plt.xticks(rotation=45,fontsize=8)
plt.show
```

[ ]: <function matplotlib.pyplot.show(close=None, block=None)>

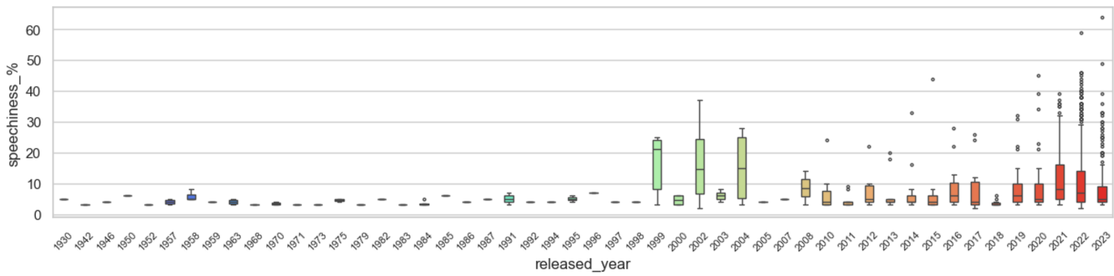


- A huge number of top streamed tracks have values of less than 50% liveness, with whiskers and interquartile range falling below 50%.
- Tracks which are performed live fall to the outliers, which means than listeners prefer to listen to recorded tracks.

Speechiness %

```
[ ]: plt.subplots(1,1, figsize=(15,3))
colors = sns.color_palette("rainbow",n_colors=50)
sns.boxplot(data=df_copy, x='released_year',y='speechiness_%', width=0.
 ↪4,fliersize=2,palette=colors ,hue='released_year',legend=False)
plt.xticks(rotation=45,fontsize=8)
plt.show
```

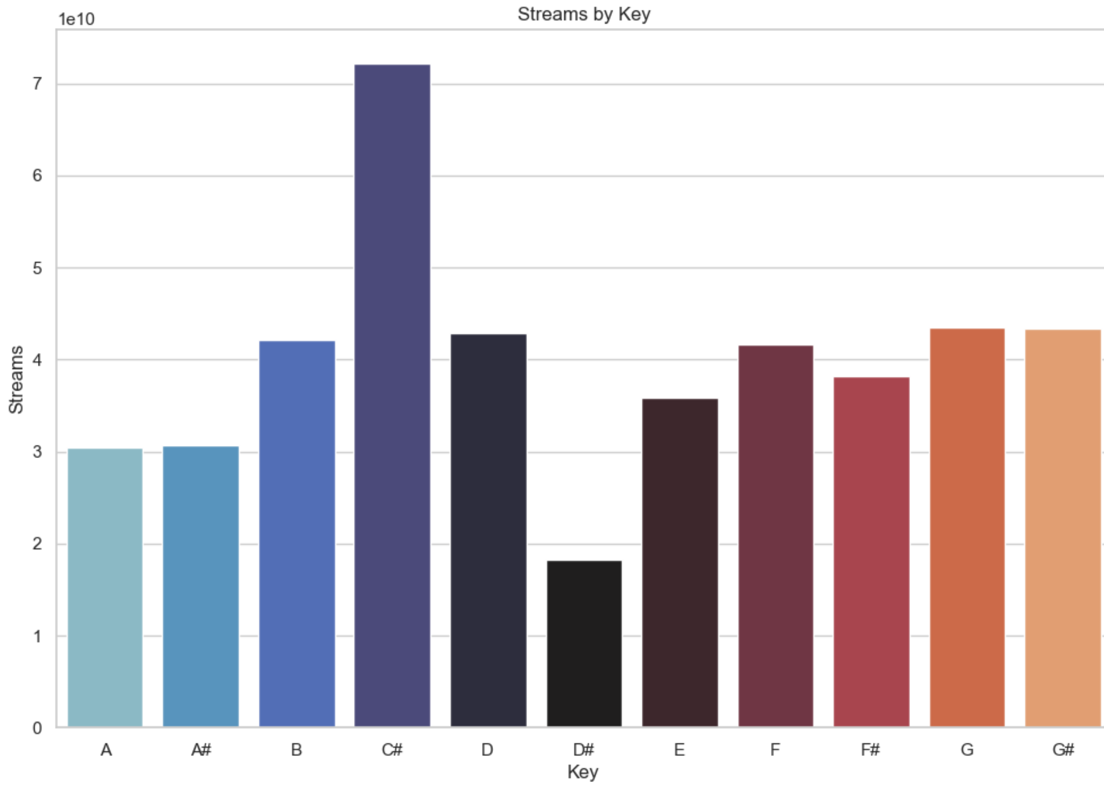
```
[ ]: <function matplotlib.pyplot.show(close=None, block=None)>
```



- Listeners prefer to listen to tracks with less speechiness or spoken words, as shown in the boxplot, where ticks and interquartile range fall below 30 to 40%, and outliers are rarely be seen above 50%.

Query 14: Most Streamed Key

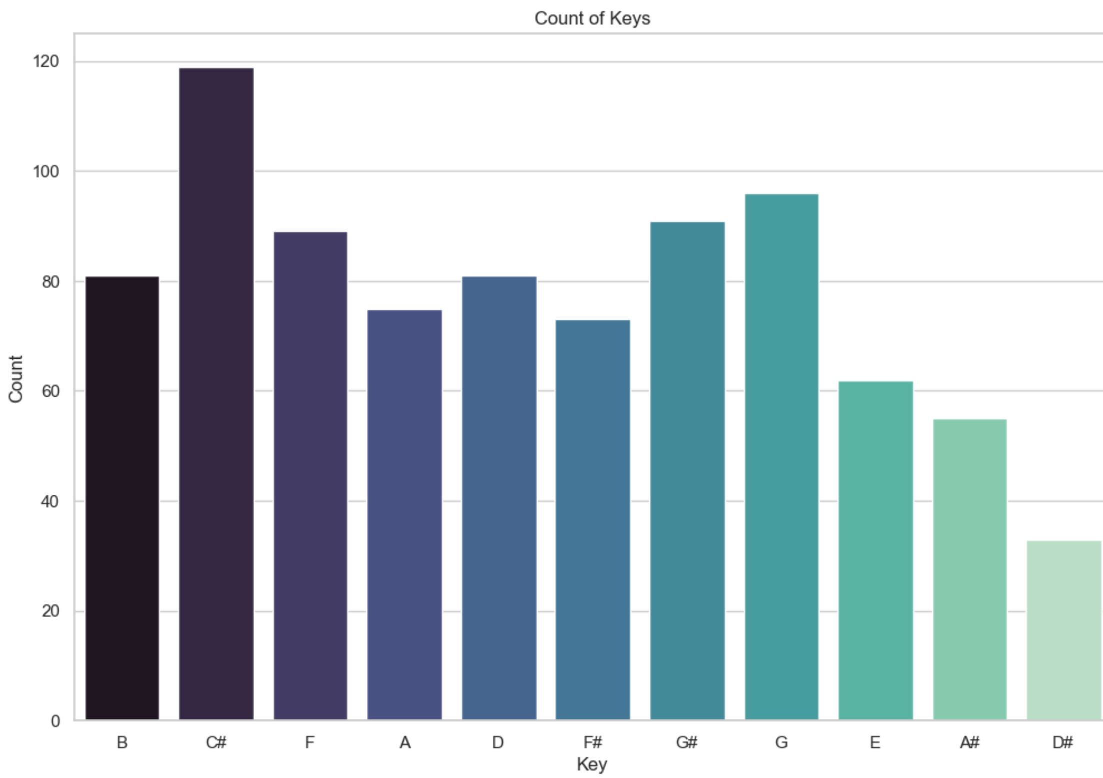
```
[ ]: plt.figure(figsize=(12, 8))
colorsKey = sns.color_palette("icefire", n_colors=11)

df_plot = df_copy.copy()
inverse_pitch_mapping = {v: k for k, v in pitch_class_mapping.items()}

df_plot['key'] = df_plot['key'].map(inverse_pitch_mapping)
df_key_sum = df_plot.groupby('key')['streams'].sum().reset_index()

sns.barplot(x='key', y='streams', data=df_key_sum, errorbar=None,␣
 ↪palette=colorsKey, hue='key', legend=False)

plt.xlabel('Key')
plt.ylabel('Streams')
plt.title('Streams by Key')
plt.show()
```



- C# is the most streamed key
- There are no streamed music with the key C
- The amount of streams for other keys have small variations

Query 14: Count of the occurrences of each key in the 'key' column of the DataFrame

[ ]:
```
plt.figure(figsize=(12, 8))
sns.countplot(x='key', data=df_plot, palette='mako', hue='key', legend=False)

plt.xlabel('Key')
plt.ylabel('Count')
plt.title('Count of Keys')

plt.show()
```


Count of Keys
120
100
80
60
40
20
0
Count
B
C#
F
A
D
F#
G#
G
E
A#
D#
Key