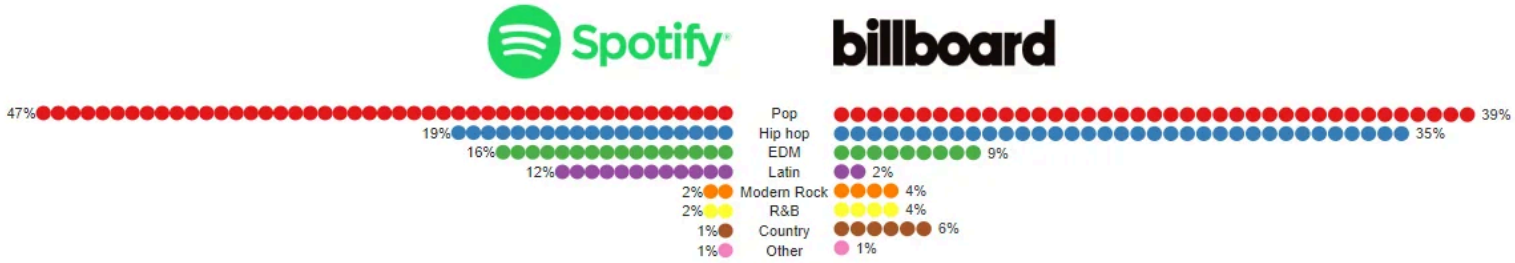
SPOTIFY DATA ANALYSIS AND VISUALIZATION

ABOT WALL CONSULTANTS™



INTRODUCTION:

With the shift in terms of how the average human consumes content in today's creator-driven economy, data driven insights are at the heart of identifying what drives success for these hedonic experiences. Collaboration has become more important than ever as it is quite common to see multiple artists collaborating on their tracks. Music and Short form videos are a natural synergy now as they mesh well together and create greater value through that association. Musicians these days are also focused on trying to create tracks that can go viral on social media as that virality further fuels their success in terms of views, repeats and reshares.

We at ABOT Wall Consultants are a Music Industry market research firm where clients come to us for insights so that they can leverage the market research studies and incorporate those in their decision making process to create successfull and viral music content.

Currently, we have a new client who is an upcoming artist in the industry by the name of "Lil Py" and requires our insights and help to aid his creative music decisions.

The aim of our project is to investigate and map the factors that make a song successful beyond the traditional metrics in music. Through our analysis, we aim to visualize the effect of these factors to potentially aid creative music decisions through the power of

data driven decision making.

DATA COLLECTION:

1. The first source of data collected for the project and analysis is from Kaggle. The Kaggle dataset describes songs from the period 1942 to 2021. Songs from various genres like pop, EDM, hip-hop, rap, latin, rock etc are included along with the artist's information, release dates, number of streams for each song and the number of followers the artist has along with some additional data. Some other insightful data columns in the data set contain song features such as its popularity relative to other songs, the energy, valence, acousticness, dancability which will be used to analyse the how these factors influence the song's success.

2. Getting real-time data from Spotify API or Spotipy library: Along with the aforementioned Kaggle dataset, we will analyse the songs released in 2022 to compare how the music preferences have changed throughout the years. In order to do so, the same song features mentioned above are required. These can be obtained by using Spotipy.

Song Features and what they mean:

Acousticness:

A confidence measure from 0.0 to 1.0 of whether the track is acoustic. 1.0 represents high confidence the track is acoustic.

Danceability:

Danceability describes how suitable a track is for dancing based on a combination of musical elements including tempo, rhythm stability, beat strength, and overall regularity. A value of 0.0 is least danceable and 1.0 is most danceable.

Liveness:

Detects the presence of an audience in the recording. Higher liveness values represent an increased probability that the track was performed live. A value above 0.8 provides strong likelihood that the track is live.

Loudness:

The overall loudness of a track in decibels (dB). Loudness values are averaged across the entire track and are useful for comparing relative loudness of tracks. Loudness is the quality of a sound that is the primary psychological correlate of physical strength (amplitude). Values typically range between -60 and 0 db.

Speechiness:

Speechiness detects the presence of spoken words in a track. The more exclusively speech-like the recording (e.g. talk show, audio book, poetry), the closer to 1.0 the attribute value. Values above 0.66 describe tracks that are probably made entirely of spoken words. Values between 0.33 and 0.66 describe tracks that may contain both music and speech, either in sections or layered, including such cases as rap music. Values below 0.33 most likely represent music and other non-speech-like tracks.

Valence:

A measure from 0.0 to 1.0 describing the musical positiveness conveyed by a track. Tracks with high valence sound more positive (e.g. happy, cheerful, euphoric), while tracks with low valence sound more negative (e.g. sad, depressed, angry).

Duration (ms):

The track length in milliseconds.

Chord/Key:

The key the track is in. Integers map to pitches using standard Pitch Class notation which is mentioned below

Notation	Chord
0	C
1	C♯, D♭
2	D
3	D♯, E♭
4	E
5	F
6	F♯, G♭
7	G
8	G♯, A♭
9	A
10	A♯, B♭
11	B

RATIONALE FOR HIGHER WEIGHTAGE:

We believe, in our humble opinion, that our project should be graded more heavily on the data processing side as we performed multiple operations that went above and beyond the requirements of traditional databases. In addition to just collecting and cleaning our databse, we collected real time data from the Spotipy API which had a lot of standardization and formatting issues. In addition to these, we also had to separately query the song ids for each song in the real time data to collect further information that was consistent with the columns of our master db. We also noticed several instances (~238) of songs where the data was corrupted or wrong. Such inconsistencies were tackled by using multiple queries that matched for ids and updated each column to remain consistent with the overall parent database.

DATA PROCESSING:

Data cleaning and managing missing data:

- Standardizing data columns: The columns had whitespaces that needed to be cleaned and standardized before they could be used for anaylsis. This was achieved using regex with `.replace()` function .
- Formatting column datatypes: In order to perform analysis and operations on the columns, we transformed the column datatypes into suitable float format using `astype()` .
- Dummy variable creation: The dataset contained some songs having multiple genres listed. In order to measure each genre's significance, we split the genre list, extracted and stored each genre category into seperate columns using `.str.contains()` , and created dummy variables.

In [1]:
```
# Libraries to be used
import numpy as np
import pandas as pd
import datetime as dt
import matplotlib.pyplot as plt
import seaborn as sns
import re
import plotly.express as px
import plotly.graph_objects as go
import spotipy
from spotipy.oauth2 import SpotifyClientCredentials
```

```
import spotipy.util as util
import time
from ipywidgets import interact, interactive, fixed, interact_manual
import ipywidgets as widgets

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

#Using a dummy client ID and Secret code to establish the connection
SPOTIPY_CLIENT_ID='37a25407afb64dd3b994d30536f8b12c'
SPOTIPY_CLIENT_SECRET='a74099f3c25b41fa97d2b750ce6fcc1e'

# Read Kaggle Database
df = pd.read_csv('Spotify_Kaggle.csv')

# Drop duplicates
df = df.drop_duplicates(subset='Song ID', keep='last')

# Standardise formats of columns
df['Streams'] = df['Streams'].replace(',','', regex=True)
df['Artist Followers'] = df['Artist Followers'].str.replace('\s','0')
df['Danceability'] = df['Danceability'].str.replace('\s','0')
df['Popularity'] = df['Popularity'].str.replace('\s','0')
df['Energy'] = df['Energy'].str.replace('\s','0')
df['Loudness'] = df['Loudness'].str.replace('\s','0')
df['Speechiness'] = df['Speechiness'].str.replace('\s','0')
df['Acousticness'] = df['Acousticness'].str.replace('\s','0')
df['Liveness'] = df['Liveness'].str.replace('\s','0')
df['Tempo'] = df['Tempo'].str.replace('\s','0')
df['Duration (ms)'] = df['Duration (ms)'].str.replace('\s','0')
df['Valence'] = df['Valence'].str.replace('\s','0')

# Update column data types for ease of processing
df['Streams'] = df['Streams'].astype(float)
df['Artist Followers'] = df['Artist Followers'].astype(float)
df['Danceability'] = df['Danceability'].astype(float)
df['Popularity'] = df['Popularity'].astype(float)
df['Energy'] = df['Energy'].astype(float)
df['Loudness'] = df['Loudness'].astype(float)
df['Speechiness'] = df['Speechiness'].astype(float)
df['Acousticness'] = df['Acousticness'].astype(float)
df['Liveness'] = df['Liveness'].astype(float)
df['Tempo'] = df['Tempo'].astype(float)
df['Duration (ms)'] = df['Duration (ms)'].astype(float)
```

```
df['Valence'] = df['Valence'].astype(float)

# Dummy variable creation for each genre
# remove the unnecessary quotation mark in this column
df['Genre'] = df['Genre'].str.replace('\'','')
#Create boolean for genres that belong to pop by using contains function and store in a new column
df['pop'] = df['Genre'].str.contains('\s*(pop)\s*')
#Create boolean for genres that belong to rap by using contains function and store in a new column
df['rap'] = df['Genre'].str.contains('\s*(rap)\s*')
#Create boolean for genres that belong to trap by using contains function and store in a new column
df['trap'] = df['Genre'].str.contains('\s*(trap)\s*')
#hip hop/ reggaeton/ latin/ electropop/ edm/ r&b/ country/ rock/
df['hip hop'] = df['Genre'].str.contains('\s*(hip hop)\s*')
df['reggaeton'] = df['Genre'].str.contains('\s*(reggaeton)\s*')
df['latin'] = df['Genre'].str.contains('\s*(latin)\s*')
df['electropop'] = df['Genre'].str.contains('\s*(electropop)\s*')
df['edm'] = df['Genre'].str.contains('\s*(edm)\s*')
df['r&b'] = df['Genre'].str.contains('\s*(r&b)\s*')
df['country'] = df['Genre'].str.contains('\s*(country)\s*')
df['rock'] = df['Genre'].str.contains('\s*(rock)\s*')
#fill the NaN by using 0
df.iloc[:,-11:] =  df.iloc[:,-11:].fillna(0)
#revise the boolean to dummy variable by using astype function
df.iloc[:,-11:] = df.iloc[:,-11:].astype(int)
```

For the Spotipy implementation we have done the following to access the data:

The client credential is first verified and then used to make valid search requests. The search request returns a **JSON metadata** about the track such album, unique song ID,song name, artists, artist ID popularity and release date of the track.To access the values of the same, it needs to be referenced by their corresponding keys.

Since the API limits the search request to 50 songs per search, we have provided the offset parameter to intialize the search after a batch of 50 songs.

After storing the details of the top 1000 songs of 2022, we then request the audio features of each song provided in the markdown listed above.

The sleep function is implemented to not exceed the rate limit of requests.

In [2]:
```
## SPOTIPY CODE - Commented out this code as we are using stored results of the API call to avoid data mismatch

# # Cleaned 235 songs with various artists
# #researching and verifying for the songs and their attributes that belonged to 'Various Artists'

# #Authorizing code flow for the client request
# auth_manager = SpotifyClientCredentials(client_id=SPOTIPY_CLIENT_ID, client_secret=SPOTIPY_CLIENT_SECRET)
# sp = spotipy.Spotify(auth_manager=auth_manager)

# #extracting the top 50 songs of the year 2020
# track_result = []
# #Creating an empty dataframe to store the features of the songs in the above dataframe
# features_df = pd.DataFrame()

# for i in range (0,1000,50):
#     track_data = sp.search(q='year:2022',type='track',limit=50,offset=i)
# #Enumerating through the item track and extracting song name, album name, track id, release date and popularity
#     for j,item in enumerate(track_data['tracks']['items']):
#         track = item['album']
#         track_id = item['id']
#         track_name = item['name']
#         popularity = item['popularity']
#         artist=item['artists']
# #Appending the extracted song attributes to the list track_result
#         track_result.append((track_id,track_name,track['artists'][0]['id'], track['artists'][0]['name'], track['name'],
# #Converting the list track_result into a dataframe
#         track_df = pd.DataFrame(track_result, index=None, columns=('Id','Song Name','Artist id','Artist', 'Album Name
#     time.sleep(0.5)
#     #Iterating through the dataframe to get the audio features
# for id in track_df['Id'].iteritems():
#     track_id = id[1]
#     audio_features = sp.audio_features(track_id)
#     local_features = pd.DataFrame(audio_features, index=[0])
#     features_df = features_df.append(local_features)
#     time.sleep(0.5)

# #Merging the track data frame with its corresponding feature dataframe
# final_df = track_df.merge(features_df,left_on='Id',right_on='id')

# #Sorting the song list based on the popularity and setting index to song name
# final_df = final_df.sort_values(by=['Popularity'], ascending=False)
```

After analyzing the dataframe , we found that some tracks had inconsitent data with respect to the artist and their respective album for eg:

The track ***Viva la Vida*** which originally belongs to the Artist ***'Coldplay'*** and the album ***Viva La Vida or Death and All His Friends***, had the artist as **Various Artist** and the album as **Billion Stream Hits**

So, to avoid the discrepancies regarding these tracks, we are searching for these songs again and appending with the valid song data to the original dataframe

The song artist followers and their respective genres has also been appended to the Spotify API data

Columns which did not contribute to the analysis were dropped and the final Spotify API data is stored into the .csv file

In [3]:
```
# #researching and verifying for the songs and their attributes that belonged to 'Various Artists'
# track_result=[]
# features_df = pd.DataFrame()
# for i in range (760,998):
#     track_data = sp.search(q='track:'+final_df["Song Name"][i],type='track',limit=1)
# #Enumerating through the item track and extracting song name, album name, track id, release date and popularity
#     for j,item in enumerate(track_data['tracks']['items']):
#         track = item['album']
#         track_id = item['id']
#         track_name = item['name']
#         popularity = item['popularity']
# #Appending the extracted song attributes to the list track_result
#         track_result.append((track_id,track_name,track['artists'][0]['id'], track['artists'][0]['name'], track['name']
# #Converting the list track_result into a dataframe
#     track_df = pd.DataFrame(track_result, index=None, columns=('Id','Song Name','Artist id','Artist', 'Album Name', 'R
#     time.sleep(0.5)
#     #Iterating through the dataframe to get the audio features
# for id in track_df['Id'].iteritems():
#     track_id = id[1]
#     audio_features = sp.audio_features(track_id)
#     local_features = pd.DataFrame(audio_features, index=[0])
#     features_df = features_df.append(local_features)
#     time.sleep(0.5)
# valid_df=track_df.merge(features_df,left_on='Id',right_on='id')
# valid_df=valid_df.sort_values(by=['Popularity'], ascending=False)
# valid_df

# #Slicing out the dataframe containing Artists as 'Various Artist'
# final_df=final_df[:761]
# #dropping duplicates if any
# valid_df.drop_duplicates(inplace=True)
```

```
# #merging the dataframe with the 2022 song dataframe
# result=pd.concat([final_df,valid_df])
# result

# #appending artist followers and genre
# followers=[]
# genre=[]
# for i in result["Artist id"]:
#     followers.append(sp.artist('spotify:artist:'+i)['followers']['total'])
#     genre.append(sp.artist('spotify:artist:'+i)['genres'])
# result['Artist_followers']=followers
# result['Genre']=genre

# #Sorting the dataframe by popularity
# result.sort_values(by='Popularity',ascending=False)
# pd.set_option('display.max_columns',None)
# result.drop(columns=['track_href',0,'uri','type','analysis_url','Artist id','Id','id','time_signature'],inplace=True)
# result

# #Saving the spotify's 2022 song dataframe as CSV
# result.to_csv('Spotipy.csv',index=False)
```

Finally, after updating, cleaning and obtaining the dataframe from our stored API results, we perform a final merge and cleaning process for the master database.

In [4]:
```
#spotipy
# pulled data from spotipy and stored in a file to merge with the master db
spotipy_df = pd.read_csv('Spotipy.csv')

# Update column data types for ease of processing
spotipy_df['Popularity']=df.Popularity.astype(float)
spotipy_df['Streams']=df.Streams.astype(float)

#Counting the number of unique values of popularity in descending order
popularity_count=df['Popularity'].value_counts().sort_index(ascending=False)

#Creating a dataframe to store mean value of streams for each unique value of popularity
stream=np.zeros(len(df.Popularity.unique()))
j=0
for i in np.sort(df['Popularity'].unique())[::-1]:
    stream[j]=stream[j]+sum(df[df['Popularity']==i]['Streams'])
    stream[j]=stream[j]/popularity_count[i]
    j+=1
stream.astype(int)
```

```
# Cleaning before merge

result=pd.read_csv('Spotipy.csv')
result.drop(columns=['Artist id','0','analysis_url','id','time_signature','track_href','type','uri'],inplace=True)
result.rename(columns={'Id':'Song ID','key':'Chord','Artist_followers':'Artist Followers','duration_ms':'Duration (ms)'
df.drop(columns='Index',inplace=True)

# Merge with Master
df=pd.concat([df,result])
df.sort_values(by=['Popularity','Release Date'],ascending=[False,True],inplace=True)

#dropping columns which don't contribute to the analysis
df.drop(columns=['instrumentalness','mode','Album Name','Highest Charting Position','Week of Highest Charting','Weeks C

#final.loc[final.Artist!='Various Artists','Streams']=final.loc[final.Artist=='Various Artists','Streams'].fillna(metho
df.loc[df.Artist=='Various Artist','Artist Followers']=df.loc[df.Artist=='Various Artists','Artist Followers']=0

#Searching for the songs with same popularity and replacing NaN values of streams with the corresponding stream values
j=0
for i in np.sort(df['Popularity'].unique())[::-1]:
    if(i==0):
        break
    else:
        df.loc[df.Popularity==i,'Streams']=df.loc[df.Popularity==i,'Streams'].fillna(value=stream[j])
        j+=1

#Replacing chord values with their notation
df.replace({'Chord':{'C':0,'C#/Db':1,'D':2,'D#/Eb':3,'E':4,'F':5,'F#/Gb':6,'G':7,'G#/Ab':8,'A':9,'A#/Bb':10,'B':11}},in

#filling NaN values of 'number of times charted' column with forward fill method
df['Number of Times Charted']=df['Number of Times Charted'].fillna(method='ffill',limit=5)

#Dropping duplicates with respect to song IDs
df.drop_duplicates(subset=['Song ID'],keep='first',inplace=True)
df.dropna(inplace=True)
```

DATA ANALYSIS:

Questions of Interest:

1. Does the artist with the highest followers have the most success in today's music industry?

2. Are there any influential metrics/factors when it comes to the success of a song?
3. Do people prefer listening to shorter songs over longer versions?
4. How much does collaboration with different artists help in terms of success (listens, streams, views)?

1. Does the artist with the highest followers have the most success in today's music industry?

In other words, we try to see if there is any correlation with artist followers and their success. To answer this question the following success metrics are considered from the dataset.

- Number of followers the artist has.
- The total number of streams the artists' songs have.
- The number of times an artist appears on top charts.

We will get the top 5 artists for each of these metrics and check if a relationship exists between them.

i. Artist Followers

Starting with a copy of the original dataframe, fetching the artists and their followers. "Followers" subscribe to the artists' content and get notified when new music/album is released.

ii. Artist Streams

Next, we check artists with the most streams for their songs. Spotify defines the number of streams as the number of times a song is played over 30 seconds. For downloaded songs, streams are calculated when the listener goes online.

To get the total number of streams for each artist, we group them based on artist names and calculate the overall stream count for all songs released by that artist.

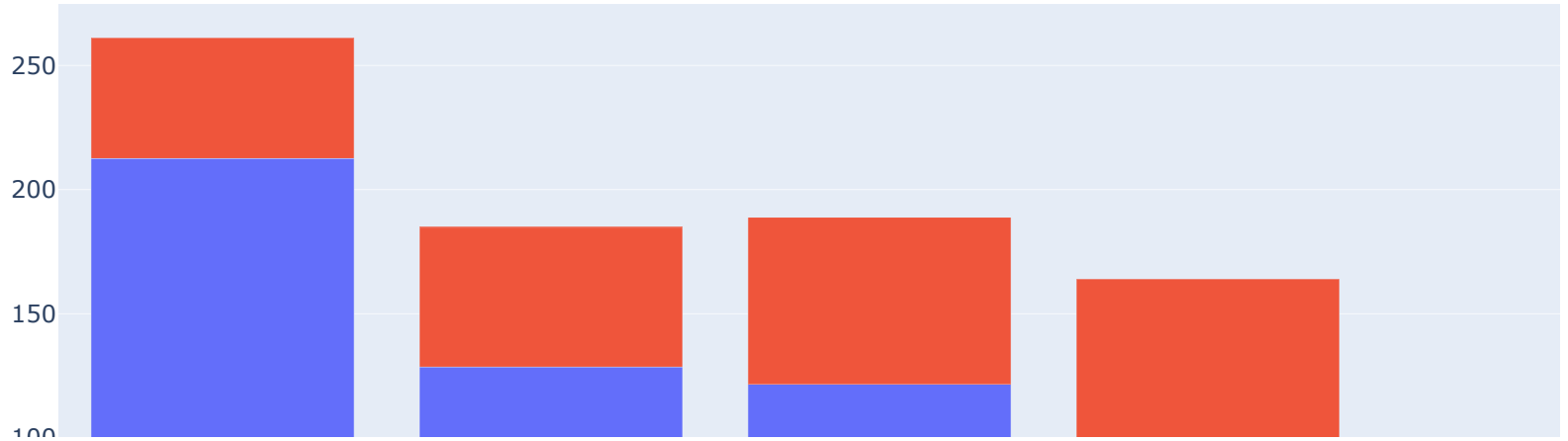
iii. Top Charts

Spotify charts are similar to the BillBoard Hot 100 charts. Spotify claims that these charts are made by listeners based on traction received by a song. Artists with great music often get charted many times and this is why we consider it a metric in our analysis.

We begin by grouping the data by Artist name and getting a total number of times the artist has been charted. We then get the artists with the maximum charting frequency.

In [5]:
```
#Creating a dataframe to get artists and their follower count
artist_followers = df[['Artist', 'Artist Followers']].copy()
artist_followers.sort_values(by=['Artist Followers'], ascending=False, inplace= True)
#Removing duplicate entries for artists
artist_followers = artist_followers.drop_duplicates(subset='Artist',keep='first')
#Setting the artist name as dataframe index
artist_followers.set_index('Artist', inplace=True)
#Dividing number of followers by 1000000 to get artist follower count in millions
artist_followers['Artist Followers'] = (artist_followers['Artist Followers']/1000000).round(2)

#Get total number of streams for all songs of an artist
artist_streams = df.groupby(by=['Artist'])
artist_streams = artist_streams.Streams.sum()
#Creating a dataframe to store the stream numbers
artist_streams= pd.DataFrame(artist_streams)
artist_streams.sort_values(by=['Streams'], ascending = False, inplace=True)
#Dividing number of streams by 1000000 to get streams in millions
artist_streams['Streams']= (artist_streams['Streams']/1000000).round(2)

#Fetching the number of times an artist was top charted
artist_charting_cnt = df.groupby(by=['Artist'])
artist_charting_cnt = artist_charting_cnt['Number of Times Charted'].sum()
#Storing into a dataframe
artist_charting_cnt= pd.DataFrame(artist_charting_cnt)
artist_charting_cnt.sort_values(by=['Number of Times Charted'], ascending = False, inplace=True)

#Creating new dataframe to get most popular artists
popular_artist = pd.merge(artist_followers,artist_streams,on='Artist')
popular_artist = pd.merge(popular_artist, artist_charting_cnt, on = 'Artist')

#Renaming the Artist stream and follower columns
popular_artist = popular_artist.rename(columns ={'Artist Followers': 'Artist Followers in Millions', 'Streams' : 'Strea

#Creating new dataframe to get most followed artists
pop_artist_follower = popular_artist.sort_values(by=['Artist Followers in Millions'], ascending=False)

#Creating new dataframe to get most streamed artists
pop_artist_stream = popular_artist.sort_values(by=['Streams in Millions'], ascending=False)

#Creating a interactive stacked bar plot to see the relationship between Streams and Followers of an artist
artist_stream_followers= popular_artist[['Artist Followers in Millions', 'Streams in Millions']].copy().head(7)
artist_stream_followers.reset_index(inplace=True)
artist_stream_followers= artist_stream_followers.sort_values(by=['Streams in Millions'], ascending=False)
```

```
#Creating a interactive stacked bar plot to see the relationship between Streams and Followers of an artist


stacked_bar = go.Figure(data=[go.Bar(name = 'Streams', x = artist_stream_followers['Artist'], y = artist_stream_followe
                              go.Bar(name = 'Followers', x=artist_stream_followers['Artist'], y = artist_stream_followe
                             ])

stacked_bar.update_layout(barmode='stack',title="Fig1: Artist's Followers vs Artist's Total Streams")
stacked_bar.show()

#Checking the correlation between Artist Followers, Number of streams and number of times the artist was charted
popular_artist_corr = popular_artist[['Artist Followers in Millions','Streams in Millions','Number of Times Charted']].
popular_artist_corr.corr()
```



Out[5]:

	Artist Followers in Millions	Streams in Millions	Number of Times Charted
Artist Followers in Millions	1.000000	0.080088	0.692482
Streams in Millions	0.080088	1.000000	0.758740
Number of Times Charted	0.692482	0.758740	1.000000

To compare the relationship between streams and followers, we have plotted a stacked bar chart.

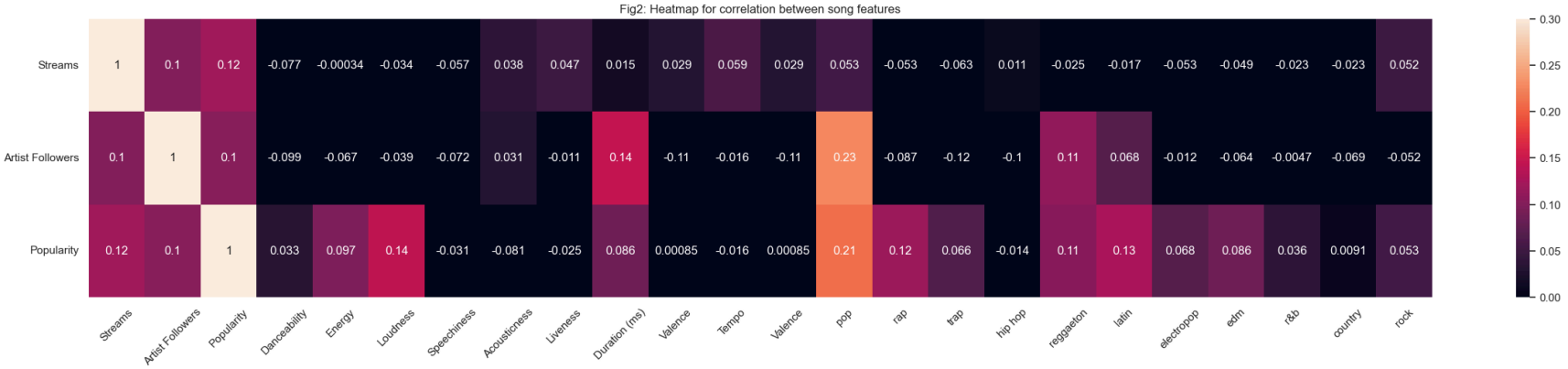
We see that Justin Bieber has the most streams but his follower count isn't very **high**. This suggests that there may not be a direct relationship between the two metrics. Similarly, we can see that although collaborations between artists may imply larger following but not necessarily translate to higher number of streams. Ed Sheeran has the most number of followers amongst all the listed artists but streams for his music are lower than Justin, Drake and Ariana.

The correlation matrix backs up the analysis as there is a very small correlation between Artist Followers and Artist Streams. However, the number of times an artist is charted has a significant influence on the artist's followers and streaming counts.

Thus we can infer that the number of artist followers does not impact their success.

2. Are there any influential metrics/factors when it comes to the success of a song?

In [6]:
```
df_corr = df[['Streams', 'Artist Followers', 'Popularity','Danceability','Energy','Loudness','Speechiness','Acousticnes
sns.set(rc = {'figure.figsize':(30,5)})
fig2 = sns.heatmap(df_corr.corr().head(3), annot=True, vmin=0, vmax=0.3)
plt.yticks(rotation=0)
plt.xticks(rotation=45)
fig2.set(title = "Fig2: Heatmap for correlation between song features")
plt.show()
```



We can observe and infer from the above visualization that clearly there are no significant correlations of any song attribute with the song's success metrics such as Streams, Followers and Popularity.

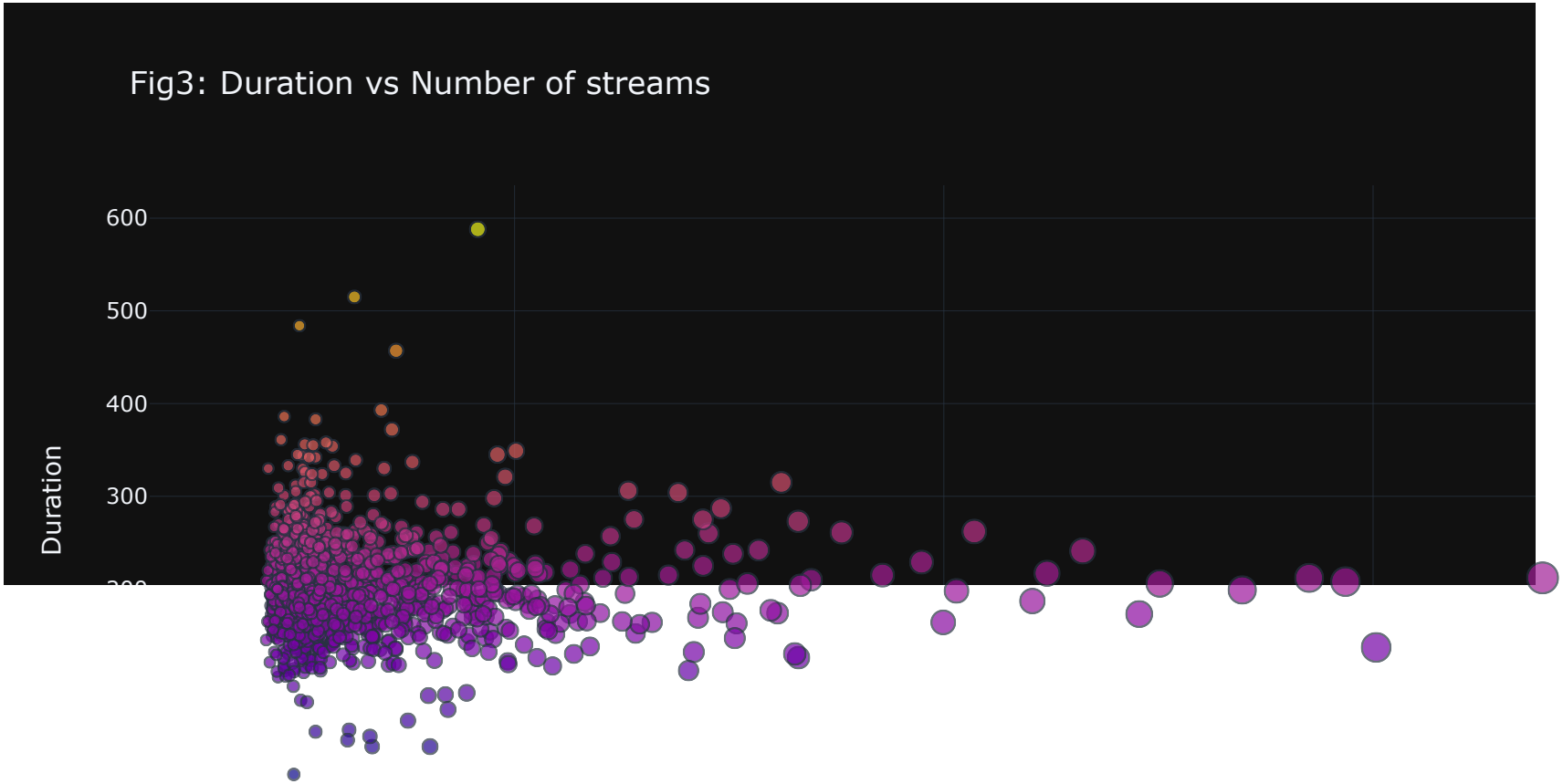
3. Do people prefer listening to shorter songs over longer versions?

Our Assumption:

When we began working on this project, we made the assumption that, given how hectic and stressful modern life is, most people prefer to listen to shorter songs than lengthier ones. To see if the most streamed songs support our hypothesis, we will create an **"interactive scatter"** plot.

Now let's look at the song lengths that different musicians have created over time. To proceed, the length of songs will be translated from milliseconds (ms) to seconds as mentioned below. To get an indication of how long the most streamed song is, we have compared the length of songs to the number of streams. To accomplish this, we used bubble chart and references from https://plotly.com/python/bubble-charts/ to become more familiar with the interactive visualizations produced by the plotly library. For a reference on how the intended visualization will seem, we referred: https://www.kaggle.com/code/varunsaikanuri/spotify-data-visualization.

In [7]:
```
#conversion of ms to second
df['Duration (ms)'] = df['Duration (ms)'].astype(float)
df['Duration']=df['Duration (ms)']//1000 #Floor division to get only the quotient
df.drop(['Duration (ms)'], axis = 1,inplace = True)

#plot duration of songs vs streams of songs
fig3 = px.scatter(df, x="Streams", y="Duration",title = "Fig3: Duration vs Number of streams", color="Duration" ,templa
fig3.show()
```



The observations we make from the **Duration VS Streams** visualisation are as follows:

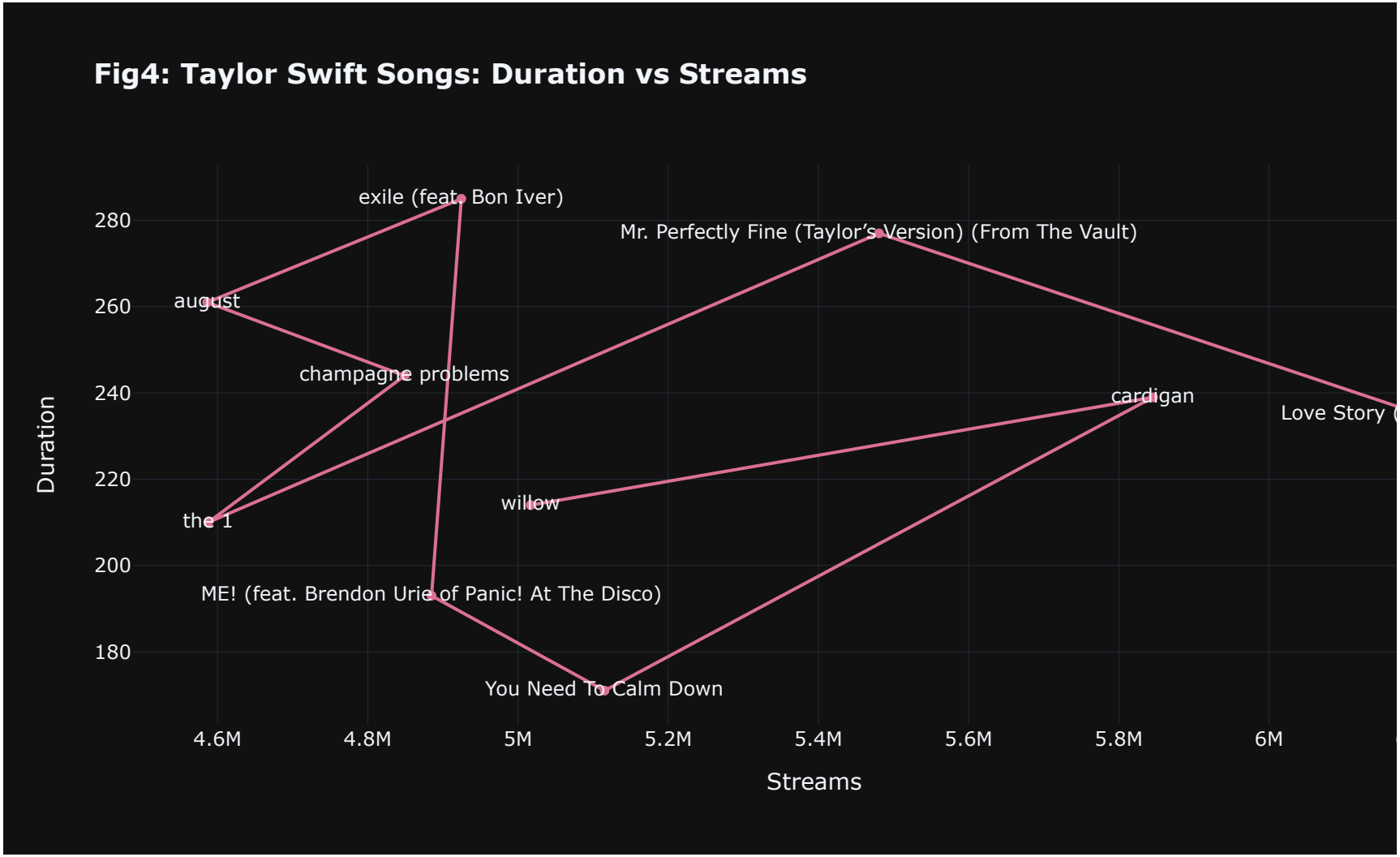
1. Songs with durations between 150 and 250 seconds and roughly 50 M are the most streamed.
2. We observe that just few songs are streamed by listeners the most.
3. The majority of the tracks range in length from 5 to 15 M streams.
4. Only a small number of outliers with durations on the extremes can be seen.

Considering the number of people who stream songs with a duration of about three minutes, it is clear from the observations above that people do indeed listen to shorter versions of music. This can make us think that the majority of individuals nowadays lead busy, hectic lives.

From the graph above, we can infer that the most streamed songs are also often approximately 200 seconds in length. This is an advice we would like to give 'Lil Py' (new producing artist) to use when creating a new song. Our most streamed artist, Taylor Swift, has the most streamed song, therefore let's use her statistics to further validate our methodology.

We understand that producing music costs a lot of money, therefore we don't want to put our customer at danger. Instead, we want to back up and further support our advice with the duration utilized in the most streamed song by the most popular artist.

In [8]:
```
#create a new dataframe for the most streamed artist
df_most_streamed_artist = df[df['Artist'] == 'Taylor Swift']

#plot a line plot to get insights
px.line(df_most_streamed_artist.head(10),
       x='Streams',y='Duration',labels={'song':'Total Songs'},width=1000,
       color_discrete_sequence=['palevioletred'],template='plotly_dark',text='Song Name',title='<b> Fig4: Taylor Swift
```



The most streamed song by Taylor Swift, who will continue to be our most popular artist through 2022, is **Love Story** which has a duration of about 235 seconds if you click on it. The curve for **Cardigan**, another song with a high stream volume, is comparable. Overall, based on our data, we would advise "Lil Py" now to have a song that is between 180 and 280 seconds in length.

As a result, this representation enables us to correctly predict our insight.

4. How much does collaboration with different artists help in terms of success (listens, streams, views)?

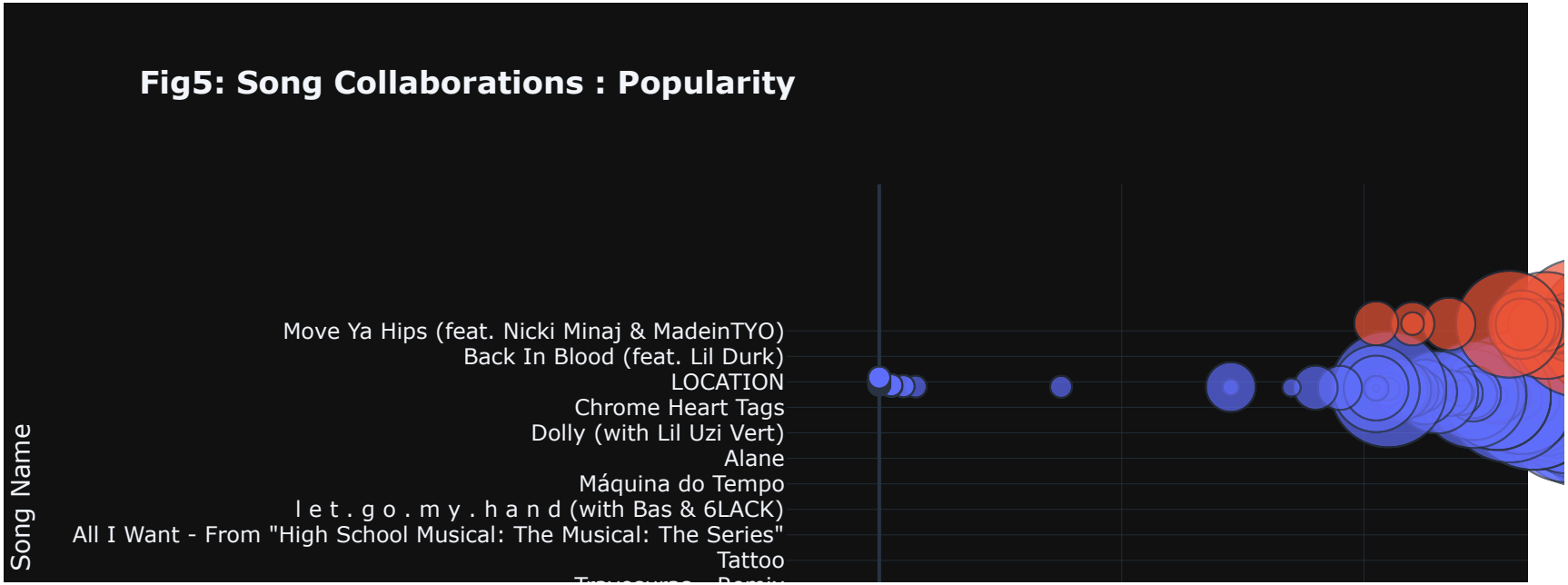
We now want to examine whether collaborations have any impact on a song's level of popularity. Our presumption is that since it combines many skills, styles, artist fans, and variety into a single song, it does have a favorable impact on popularity. Additionally, based on our own assumptions, songs these days tend to follow a genre like pop with a rap, which unquestionably involves collaboration.

A dummy variable column would be created as we move forward with this. It is assumed that the songs **with feat** in their name are **collaborations** and the tracks **without feat** are **singles.** Let's examine the popularity matrices for these two.

In [9]:
```
# creat a dummy variable of collaborations and use astype function to revise to boolean to int then to str
df['collaboration'] = df['Song Name'].str.contains("feat").astype(int).astype(str)

#change the 0's and 1's to True and False to make our figure better
df['collaboration'] = df['collaboration'].replace('0','False')
df['collaboration'] = df['collaboration'].replace('1','True')
```

Let's create a dynamic bubble chart for the songs that were cowritten and those which were singles. In order to determine whether a new artist who is generating music can be advised on which partnerships to pursue, we will set the size of the bubble to Artist Follower.

In [10]:
```
#bubble chart
fig = px.scatter(df,
      x="Popularity", y="Song Name", size="Artist Followers", color="collaboration",
      log_x=False, size_max=100,hover_name="Artist",
      template='plotly_dark', title="<b> Fig5: Song Collaborations : Popularity</b>")

fig.show()
```



The figure's blue color represents the singles that musicians have recorded, while the red bubbles represent collaborations. The song with the highest popularity score, "Unholy," is a collaboration, although the number of fans who follow the artist isn't that high. This reveals the fact that musicians with less followers can collaborate to create music, which raises their success measure. We can see that collaborations range in popularity from 41 to 100. The graph's X axis is designated as Song Name to provide readers a general notion of the song's title and featured performer. Let's examine the graph using only the singles that artists have generated.

The aforementioned graph also features the song "Begin'," which has a popularity score of 100. Even still, it is clear that in this instance the artist's follower is not the best. The popularity number for singles ranges from 1 to 100, which leads us to believe that our

presumption was accurate. Overall, songs with partnerships have greater success histories.

Our assumption is accurate, thus we strongly advise our client to move forward with partnerships in order to attain sky-high numbers and success in the music business.

Conclusions:

To summarize our results, we can infer some intriguing patterns and relationships that can influence the decision making of our client who is a an upcoming artist: 1) Artist followers and popularity clearly are not strong indicators of success or high stream count. Our client can safely assume that his low follower count, as a result of his early stage music career, will not have a significant impact on his stream count and success.

2) From the lense of genres, we can infer that no individual genre has a huge impact on the success of a song. The overall muscial attributes of a track such as acousticness, danceability etc. also do not showcase any significant impact for the popularity or virality of a track. This is a great insight as it allows our client to explore their creative side and produce music that is true to their experiences and music vibe.

3) When exploring the relationship between duration and the success, we could observe that the market demand for shorter duration of tracks has been on an uptrend and that the most streamed songs are typically between 140-210 seconds. Thus, our recommendation would be to produce tracks that are aligned with that range of duration to drive maximum success and streams.

4) Finally, while exploring the effect of collaboration with other artists, we observed that when artists collaborate, their default threshold for popularity is 40 base points. This is strikingly different for the case when artists do not collaborate as the base threshold is 1 basis point. When thinking about this intuitively, it adds up as the more the artists collaborate, the more searched and viral the track will be due to the interaction of their different listening audience.

These results are extremely meaningful as we believe that if our client focusses on collaboration with new artists while creating shorter duration tracks and exploring their musical creativity, they should be able to churn out bangers month after month!

KEEP CREATING AND STREAMING OR RATHER 'SPOTIFYING'!!

Thank you!

References:

1. Article title : Welcome to Spotipy!
   URL : https://spotipy.readthedocs.io/en/2.21.0/
   Website title : Welcome to Spotipy! - spotipy 2.0 documentation

1. Article title : Bubble
   URL : https://plotly.com/python/bubble-charts/
   Website title : Bubble charts in Python

2. Article title : Spotify Data Analysis with Python
   URL : https://blog.devgenius.io/spotify-data-analysis-with-python-a727542beaa7
   Website title : Medium
   Date published : March 25, 2022

3. Article title : Spotify Data Visualization
   URL : https://www.kaggle.com/code/varunsaikanuri/spotify-data-visualization
   Website title : Kaggle
   Date published : October 12, 2022

4. Article title : publiccode/1-Albums-tracks-audiofeatures.ipynb at master · prodramp/publiccode
   URL : https://github.com/prodramp/publiccode/blob/master/machine_learning/spotify-projects/1-Albums-tracks-audiofeatures.ipynb
   Website title : GitHub
   Date published : March 17, 2022