Spotify Wrapped: Exploratory Data Analysis Project

In this project, I have delved into my Spotify usage Spotify Wrapped. Spotify is renowned as one of the most popular and widely used music streaming platforms. The dataset utilized here represents my personal usage of this platform.

The dataset provides insights into the following aspects:

The songs I have played The artists behind those songs The duration of my usage, and more I obtained this dataset by downloading my personal usage data from Spotify's Privacy Setting section. Spotify allows users to access and download their personal usage data, and it typically takes up to 30 days to receive the complete dataset. Fortunately, I received my entire history within a maximum of 23 days, but generally, it may take the full 30-day duration. With this dataset at hand, I aim to analyze my streaming history and extract valuable insights from it.

0. Imports

In [8]:
```
%matplotlib inline
import pandas as pd
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
import seaborn as sns
from wordcloud import WordCloud
```

In [9]:
```
pip install wordcloud
```

```
Requirement already satisfied: wordcloud in c:\users\sawra\appdata\local\anaconda3\lib\s
ite-packages (1.9.2)
Requirement already satisfied: numpy>=1.6.1 in c:\users\sawra\appdata\local\anaconda3\li
b\site-packages (from wordcloud) (1.24.3)
Requirement already satisfied: pillow in c:\users\sawra\appdata\local\anaconda3\lib\site
-packages (from wordcloud) (9.4.0)
Requirement already satisfied: matplotlib in c:\users\sawra\appdata\local\anaconda3\lib
\site-packages (from wordcloud) (3.7.1)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\sawra\appdata\local\anaconda
3\lib\site-packages (from matplotlib->wordcloud) (1.0.5)
Requirement already satisfied: cycler>=0.10 in c:\users\sawra\appdata\local\anaconda3\li
b\site-packages (from matplotlib->wordcloud) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\sawra\appdata\local\anacond
a3\lib\site-packages (from matplotlib->wordcloud) (4.25.0)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\sawra\appdata\local\anacond
a3\lib\site-packages (from matplotlib->wordcloud) (1.4.4)
Requirement already satisfied: packaging>=20.0 in c:\users\sawra\appdata\local\anaconda3
\lib\site-packages (from matplotlib->wordcloud) (23.0)
Requirement already satisfied: pyparsing>=2.3.1 in c:\users\sawra\appdata\local\anaconda
3\lib\site-packages (from matplotlib->wordcloud) (3.0.9)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\sawra\appdata\local\anac
onda3\lib\site-packages (from matplotlib->wordcloud) (2.8.2)
Requirement already satisfied: six>=1.5 in c:\users\sawra\appdata\local\anaconda3\lib\si
te-packages (from python-dateutil>=2.7->matplotlib->wordcloud) (1.16.0)
Note: you may need to restart the kernel to use updated packages.
```

1. Downloading the Dataset

One can can download the ZIP file with a copy of most of the personal data by using the automated Download your data function on the Privacy Settings section of account page in Spotify.

Instructions for downloading the dataset

1. Go to the Privacy Settting Page of your Spotify Account.
2. Scroll to the bottom and you'll see a section called Download Your Data.
3. You'll see a three step process with instruction to download the data.
4. You have to to Request for your data the you'll get an confirmation email from Spotify to Confirm the request.
5. After collecting the required information, Spotify will create a Zip File and send you an email with the link to download it.
6. If you can't find the email, you can request again from your Privacy Settting Page.

For More Information: https://support.spotify.com/us/article/data-rights-and-privacy-settings/

In [ ]:

In [ ]:

1.1 Read the data

In [10]:
```
spotify_df=pd.read_json(r"C:\Users\sawra\spotify\Streaming_History_Audio_2023.json")
```

In [11]:
```
spotify_df.head()
```

Out[11]:

	ts	username	platform	ms_played	conn_country	ip_addr_decrypted	user_age
0	2023-01-06T14:15:42Z	31k4xbqhsw7unisqoe62lohjknma	ios	154275	IN	106.197.9.194	
1	2023-01-06T14:16:23Z	31k4xbqhsw7unisqoe62lohjknma	ios	39832	IN	106.197.9.194	
2	2023-01-06T14:19:00Z	31k4xbqhsw7unisqoe62lohjknma	ios	156680	IN	106.197.9.194	
3	2023-01-06T14:21:24Z	31k4xbqhsw7unisqoe62lohjknma	ios	140805	IN	106.197.9.194	
4	2023-01-06T14:24:29Z	31k4xbqhsw7unisqoe62lohjknma	ios	168642	IN	106.197.9.194	

5 rows × 21 columns

In [12]:
```
spotify_df.shape
```

Out[12]:
```
(2809, 21)
```

In [ ]:

2. Data Preparation and Cleaning

2.1 Prepare the data

In [13]:
```
spotify_df.to_csv(r"C:\Users\sawra\spotify\Streaming_History_Audio_2023.csv", index=Fals
```

In [14]:
```
spotify_df
```

Out[14]:

	ts	username	platform	ms_played	conn_country	ip_addr_decrypted	user_
0	2023-01-06T14:15:42Z	31k4xbqhsw7unisqoe62lohjknma	ios	154275	IN	106.197.9.194	
1	2023-01-06T14:16:23Z	31k4xbqhsw7unisqoe62lohjknma	ios	39832	IN	106.197.9.194	
2	2023-01-06T14:19:00Z	31k4xbqhsw7unisqoe62lohjknma	ios	156680	IN	106.197.9.194	
3	2023-01-06T14:21:24Z	31k4xbqhsw7unisqoe62lohjknma	ios	140805	IN	106.197.9.194	
4	2023-01-06T14:24:29Z	31k4xbqhsw7unisqoe62lohjknma	ios	168642	IN	106.197.9.194	
...	...	...	...	...	...	...	
2804	2023-09-15T16:07:16Z	31k4xbqhsw7unisqoe62lohjknma	ios	200373	IN	223.238.119.117	
2805	2023-09-15T16:17:49Z	31k4xbqhsw7unisqoe62lohjknma	ios	154554	IN	223.238.113.180	
2806	2023-09-15T16:18:46Z	31k4xbqhsw7unisqoe62lohjknma	ios	9056	IN	223.238.113.180	
2807	2023-09-15T16:22:18Z	31k4xbqhsw7unisqoe62lohjknma	ios	200373	IN	223.238.113.180	
2808	2023-09-15T16:26:44Z	31k4xbqhsw7unisqoe62lohjknma	ios	266075	IN	223.238.113.180	

2809 rows × 21 columns

In [15]:
```
spotify_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2809 entries, 0 to 2808
Data columns (total 21 columns):
 #   Column                              Non-Null Count  Dtype
---  ------                              --------------  -----
 0   ts                                  2809 non-null   object
 1   username                            2809 non-null   object
 2   platform                            2809 non-null   object
 3   ms_played                           2809 non-null   int64
 4   conn_country                        2809 non-null   object
 5   ip_addr_decrypted                   2809 non-null   object
 6   user_agent_decrypted                2809 non-null   object
 7   master_metadata_track_name          2808 non-null   object
 8   master_metadata_album_artist_name   2808 non-null   object
 9   master_metadata_album_album_name    2808 non-null   object
 10  spotify_track_uri                   2808 non-null   object
 11  episode_name                        1 non-null      object
 12  episode_show_name                   1 non-null      object
 13  spotify_episode_uri                 1 non-null      object
 14  reason_start                        2809 non-null   object
 15  reason_end                          2809 non-null   object
 16  shuffle                             2809 non-null   bool
 17  skipped                             2809 non-null   bool
 18  offline                             2809 non-null   bool
 19  offline_timestamp                   2809 non-null   int64
 20  incognito_mode                      2809 non-null   bool
dtypes: bool(4), int64(2), object(15)
memory usage: 384.2+ KB
```

In [16]:
```
spotify_df.nunique()
```

Out[16]:
```
ts                                  2783
username                               1
platform                               1
ms_played                           1811
conn_country                           1
ip_addr_decrypted                    297
user_agent_decrypted                   1
master_metadata_track_name           986
master_metadata_album_artist_name    498
master_metadata_album_album_name     911
spotify_track_uri                   1027
episode_name                           1
episode_show_name                      1
spotify_episode_uri                    1
reason_start                           9
reason_end                             7
shuffle                                2
skipped                                2
offline                                1
offline_timestamp                   2784
incognito_mode                         1
dtype: int64
```

2.2 Clean the data

We can see that we have a lot of columns, some of which are not useful anymore, so we'll make a new dataframe with the required colums.

In [17]:
```
spotify_stream_df = spotify_df[['ts', 'ms_played', 'master_metadata_track_name', 'master
spotify_stream_df.head()
```

Out[17]:

	ts	ms_played	master_metadata_track_name	master_metadata_album_artist_name
0	2023-01-06T14:15:42Z	154275	Zehen	Mitraz
1	2023-01-06T14:16:23Z	39832	Mehboob	Mitraz
2	2023-01-06T14:19:00Z	156680	Enna Sona	Mitraz
3	2023-01-06T14:21:24Z	140805	Teri Rahaan	Mitraz
4	2023-01-06T14:24:29Z	168642	Lage Nahi Man	Vismay Patel

In [ ]:

```
```

Convert the 'ts' column to datetime format

In [18]:

```
spotify_stream_df['ts'] = pd.to_datetime(spotify_stream_df['ts'])
spotify_stream_df['ts'] = spotify_stream_df['ts'].dt.strftime('%Y-%m-%d %H:%M')

spotify_stream_df.head()
```

Out[18]:

	ts	ms_played	master_metadata_track_name	master_metadata_album_artist_name
0	2023-01-06 14:15	154275	Zehen	Mitraz
1	2023-01-06 14:16	39832	Mehboob	Mitraz
2	2023-01-06 14:19	156680	Enna Sona	Mitraz
3	2023-01-06 14:21	140805	Teri Rahaan	Mitraz
4	2023-01-06 14:24	168642	Lage Nahi Man	Vismay Patel

In [ ]:

```
```

In [ ]:

```
```

In [19]:

```
len(spotify_stream_df["master_metadata_album_artist_name"].unique()) # Length of unique
```

Out[19]:

```
499
```

In [20]:

```
len(spotify_stream_df["master_metadata_track_name"].unique()) # Length of unique tracks
```

Out[20]:

```
987
```

2.3 Data formatting

In [21]:

```
spotify_stream_df["Play-Time"]= pd.to_datetime(spotify_stream_df["ts"]) # To create a ad
```

In [22]:

```
spotify_stream_df['year'] = pd.DatetimeIndex(spotify_stream_df["Play-Time"]).year
spotify_stream_df['month'] = pd.DatetimeIndex(spotify_stream_df["Play-Time"]).month
spotify_stream_df['day'] = pd.DatetimeIndex(spotify_stream_df["Play-Time"]).day
spotify_stream_df['weekday'] = pd.DatetimeIndex(spotify_stream_df["Play-Time"]).weekday
spotify_stream_df['time'] = pd.DatetimeIndex(spotify_stream_df["Play-Time"]).time
spotify_stream_df['hours'] = pd.DatetimeIndex(spotify_stream_df["Play-Time"]).hour
spotify_stream_df['day-name'] = spotify_stream_df["Play-Time"].apply(lambda x: x.day_nam
spotify_stream_df['Count'] = 1
```

In [23]:

```
spotify_stream_df["Time-Played (hh-mm-ss)"] = pd.to_timedelta(spotify_stream_df["ms_play
```

In [24]:
```
def hours(td):
    # To get the hour information
    return td.seconds/3600

def minutes(td):
    # To get the minutes information
    return (td.seconds/60)%60

spotify_stream_df["Listening Time(Hours)"] = spotify_stream_df["Time-Played (hh-mm-ss)"]
spotify_stream_df["Listening Time(Minutes)"] = spotify_stream_df["Time-Played (hh-mm-ss)
```

In [25]:
```
spotify_stream_df.head() # To check the newly formed dataset with additional columns
```

Out[25]:

	ts	ms_played	master_metadata_track_name	master_metadata_album_artist_name	Play-Time	year	month
0	2023-01-06 14:15	154275	Zehen	Mitraz	2023-01-06 14:15:00	2023	1
1	2023-01-06 14:16	39832	Mehboob	Mitraz	2023-01-06 14:16:00	2023	1
2	2023-01-06 14:19	156680	Enna Sona	Mitraz	2023-01-06 14:19:00	2023	1
3	2023-01-06 14:21	140805	Teri Rahaan	Mitraz	2023-01-06 14:21:00	2023	1
4	2023-01-06 14:24	168642	Lage Nahi Man	Vismay Patel	2023-01-06 14:24:00	2023	1

We can see that now we have a lot of columns, some of which are not useful anymore, so we'll drop few of them.

In [26]:
```
spotify_stream_df.drop(columns=["ts","Time-Played (hh-mm-ss)","ms_played"], inplace=True
```

In [27]:
```
spotify_stream_df.describe() # Final check for any abnormality
```

Out[27]:

	year	month	day	weekday	hours	Count	Listening Time(Hours)	Listening Time(Minutes)
count	2809.0	2809.000000	2809.000000	2809.000000	2809.000000	2809.0	2809.000000	2809.000000
mean	2023.0	5.271983	15.153791	3.250979	11.215379	1.0	0.035209	2.112138
std	0.0	2.160923	9.145344	2.082868	5.304029	0.0	0.028600	1.715374
min	2023.0	1.000000	1.000000	0.000000	0.000000	1.0	0.000000	0.000000
25%	2023.0	4.000000	6.000000	1.000000	8.000000	1.0	0.005000	0.283000
50%	2023.0	5.000000	15.000000	4.000000	12.000000	1.0	0.037000	2.217000
75%	2023.0	7.000000	24.000000	5.000000	14.000000	1.0	0.058000	3.500000
max	2023.0	9.000000	31.000000	6.000000	23.000000	1.0	0.180000	10.800000

In [ ]:
```

```

Note: Now we have a clean and properly formatted data we can go on with our analysis.

3. Exploratory Analysis and Visualization

In [28]:
```
sns.set_style('darkgrid')
plt.style.use('seaborn-darkgrid')

matplotlib.rcParams['font.size'] = 14
matplotlib.rcParams['figure.figsize'] = (9, 5)
matplotlib.rcParams['figure.facecolor'] = '#00000000'
```

```
C:\Users\sawra\AppData\Local\Temp\ipykernel_6644\1738326915.py:2: MatplotlibDeprecationW
arning: The seaborn styles shipped by Matplotlib are deprecated since 3.6, as they no lo
nger correspond to the styles shipped by seaborn. However, they will remain available as
'seaborn-v0_8-<style>'. Alternatively, directly use the seaborn API instead.
  plt.style.use('seaborn-darkgrid')
```

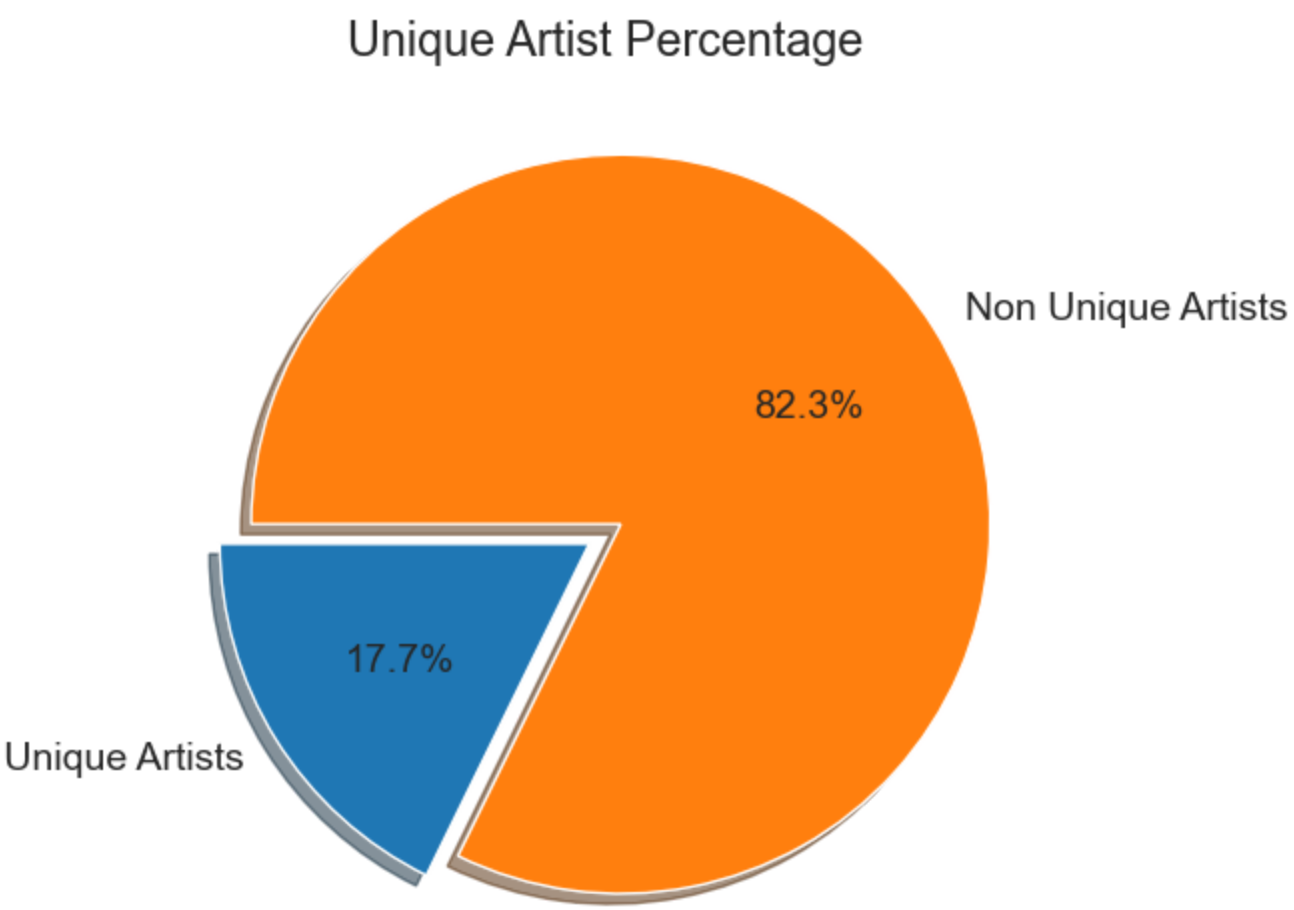
3.1 Artist Name (Exploration)

3.1.1 We can check what is the percentage of unique artist we have.

In [29]:
```
unique_artists = spotify_stream_df["master_metadata_album_artist_name"].nunique() # Coun
total_artists = spotify_stream_df["master_metadata_album_artist_name"].count() # Count t
unique_artist_percentage = unique_artists/total_artists*100 # Get the percentage of the
unique_artist_percentage
```

Out[29]:
```
17.735042735042736
```

In [30]:
```
unique_artist_list = np.array([unique_artists, total_artists-unique_artists])
unique_artist_list_labels = [" Unique Artists", "Non Unique Artists"]

fig, ax = plt.subplots(figsize=(12,6))
ax.pie(unique_artist_list, labels= unique_artist_list_labels, autopct='%1.1f%%',explode=
plt.title("Unique Artist Percentage")
plt.show()
```



3.1.2 We can also check the top 10 unique artist we have.

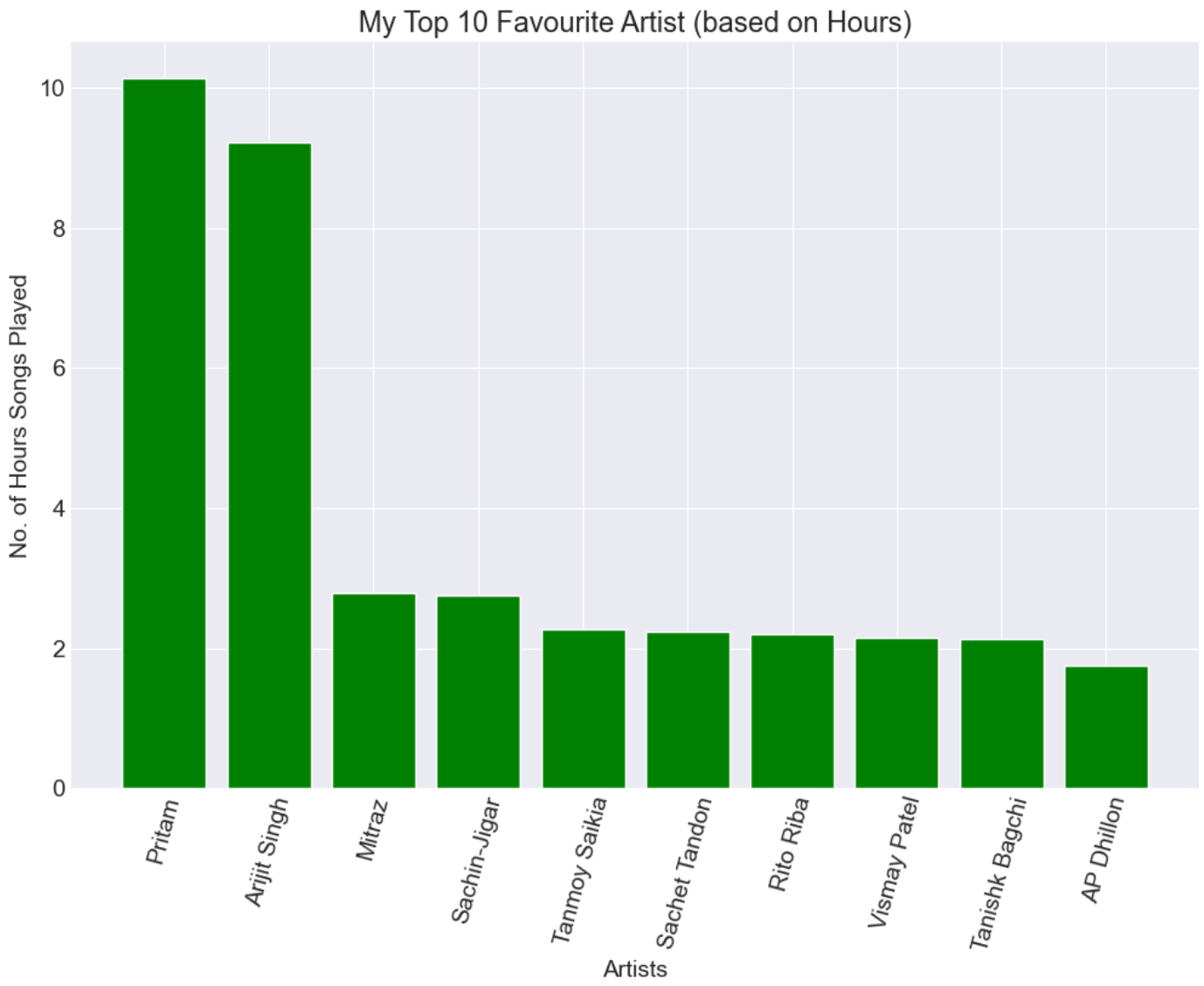
In [31]:
```
top_10_artist_df = spotify_stream_df.groupby(["master_metadata_album_artist_name"])[["Li
top_10_artist_df.head(10)
```

Out[31]:

master_metadata_album_artist_name	Listening Time(Hours)	Listening Time(Minutes)	Count
Pritam	10.147	609.081	213
Arijit Singh	9.230	553.781	181
Mitraz	2.797	167.705	90
Sachin-Jigar	2.757	165.175	61
Tanmoy Saikia	2.272	136.150	38
Sachet Tandon	2.238	133.929	59
Rito Riba	2.205	132.363	58
Vismay Patel	2.144	128.726	60
Tanishk Bagchi	2.130	127.268	44
AP Dhillon	1.744	104.486	56

In [32]:
```
fig,ax = plt.subplots(figsize=(12,8))
ax.bar(top_10_artist_df.head(10).index,top_10_artist_df["Listening Time(Hours)"].head(10
```

```
ax.set(title="My Top 10 Favourite Artist (based on Hours)",xlabel="Artists",ylabel="No.
plt.xticks(rotation=75);
```



3.1.3 Top 10 Unique Artist (count) : Based on the number of count

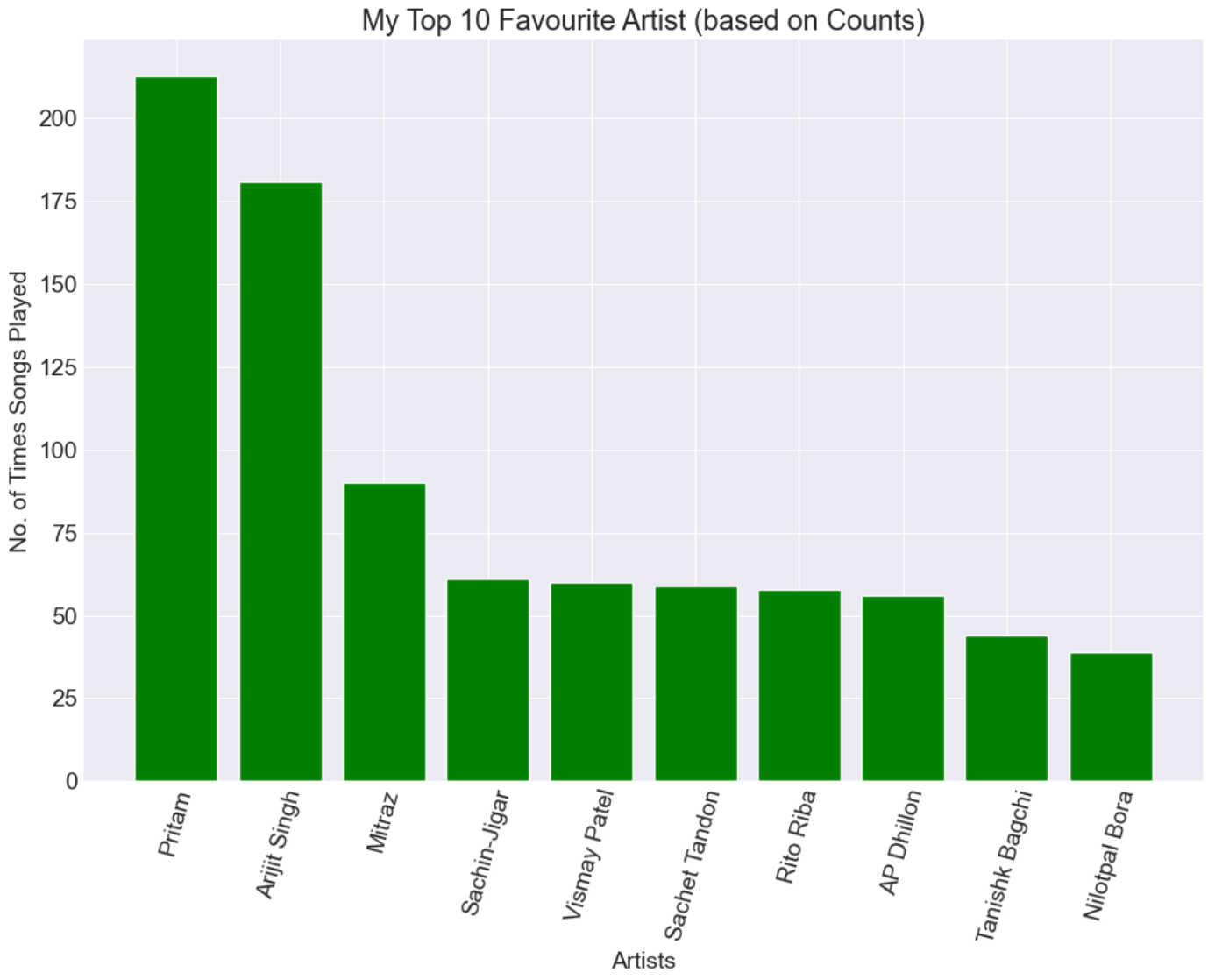
In [33]:
```
top_10_artist_count_df = spotify_stream_df.groupby(["master_metadata_album_artist_name"]
top_10_artist_count_df.head(10)
```

Out[33]:

master_metadata_album_artist_name	Listening Time(Hours)	Listening Time(Minutes)	Count
Pritam	10.147	609.081	213
Arijit Singh	9.230	553.781	181
Mitraz	2.797	167.705	90
Sachin-Jigar	2.757	165.175	61
Vismay Patel	2.144	128.726	60
Sachet Tandon	2.238	133.929	59
Rito Riba	2.205	132.363	58
AP Dhillon	1.744	104.486	56
Tanishk Bagchi	2.130	127.268	44
Nilotpal Bora	1.510	91.127	39

In [34]:

```
fig,ax = plt.subplots(figsize=(12,8))
ax.bar(top_10_artist_count_df.head(10).index,top_10_artist_count_df["Count"].head(10),co
ax.set(title="My Top 10 Favourite Artist (based on Counts)",xlabel="Artists",ylabel="No.
plt.xticks(rotation=75);
```



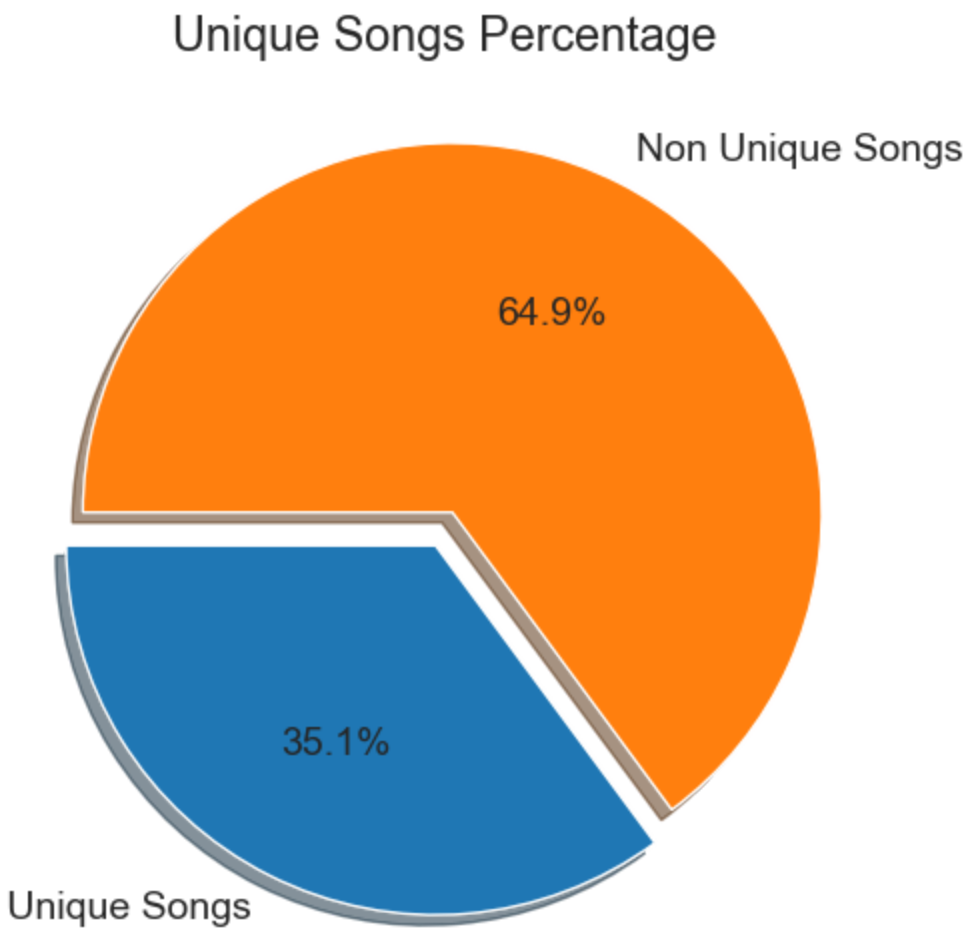
3.2 Song Tracks (Exploration)

3.2.1 We can check what is the percentage of unique songs we have

In [35]:
```
unique_songs = spotify_stream_df["master_metadata_track_name"].nunique()
total_songs = spotify_stream_df["master_metadata_track_name"].count()
unique_songs_percentage = unique_songs/total_songs*100
unique_songs_percentage
```

Out[35]:
```
35.11396011396011
```

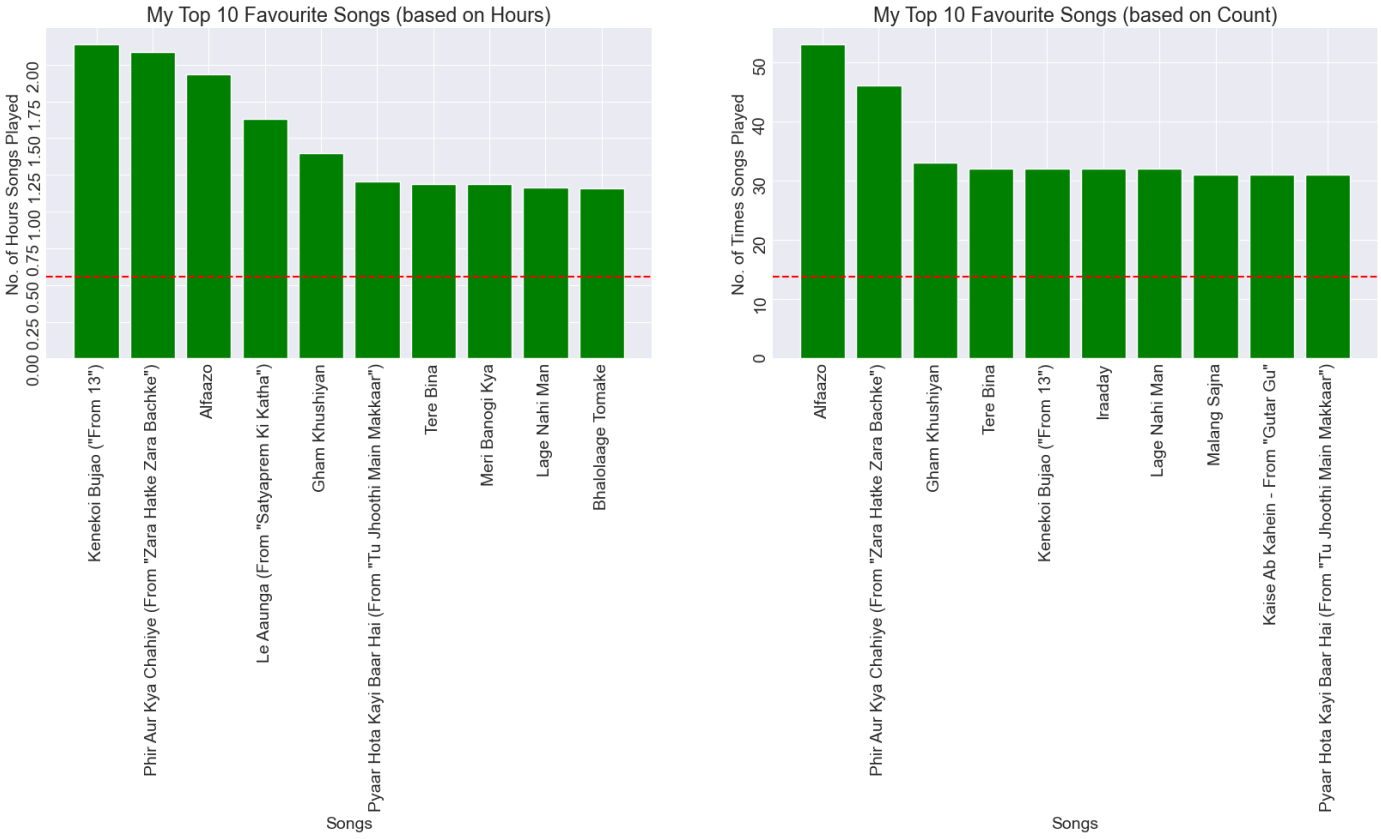
In [36]:
```
unique_songs_list = np.array([unique_songs, total_songs-unique_songs])
unique_songs_list_labels = [" Unique Songs", "Non Unique Songs"]

fig, ax = plt.subplots(figsize=(12,6))
ax.pie(unique_songs_list, labels= unique_songs_list_labels, autopct='%1.1f%%', explode=[
plt.title("Unique Songs Percentage");
```



3.2.2 We can also check the top 10 unique songs we have

In [37]:
```
top_10_songs_time_df = spotify_stream_df.groupby(["master_metadata_track_name"])[["Liste
top_10_songs_count_df = spotify_stream_df.groupby(["master_metadata_track_name"])[["List
```

In [38]:
```
fig, (ax1,ax2) = plt.subplots(1,2,figsize=(20,5))

# first graph
ax1.bar(top_10_songs_time_df.head(10).index,top_10_songs_time_df["Listening Time(Hours)"
ax1.set(title="My Top 10 Favourite Songs (based on Hours)",xlabel="Songs",ylabel="No. of
ax1.tick_params(labelrotation=90);
ax1.axhline(top_10_songs_time_df["Listening Time(Hours)"][:100].mean(), linestyle="--",

# second graph
ax2.bar(top_10_songs_count_df.head(10).index,top_10_songs_count_df["Count"].head(10), co
ax2.set(title="My Top 10 Favourite Songs (based on Count)",xlabel="Songs",ylabel="No. of
ax2.tick_params(labelrotation=90);
ax2.axhline(top_10_songs_count_df["Count"][:100].mean(), linestyle="--", color="r");
```

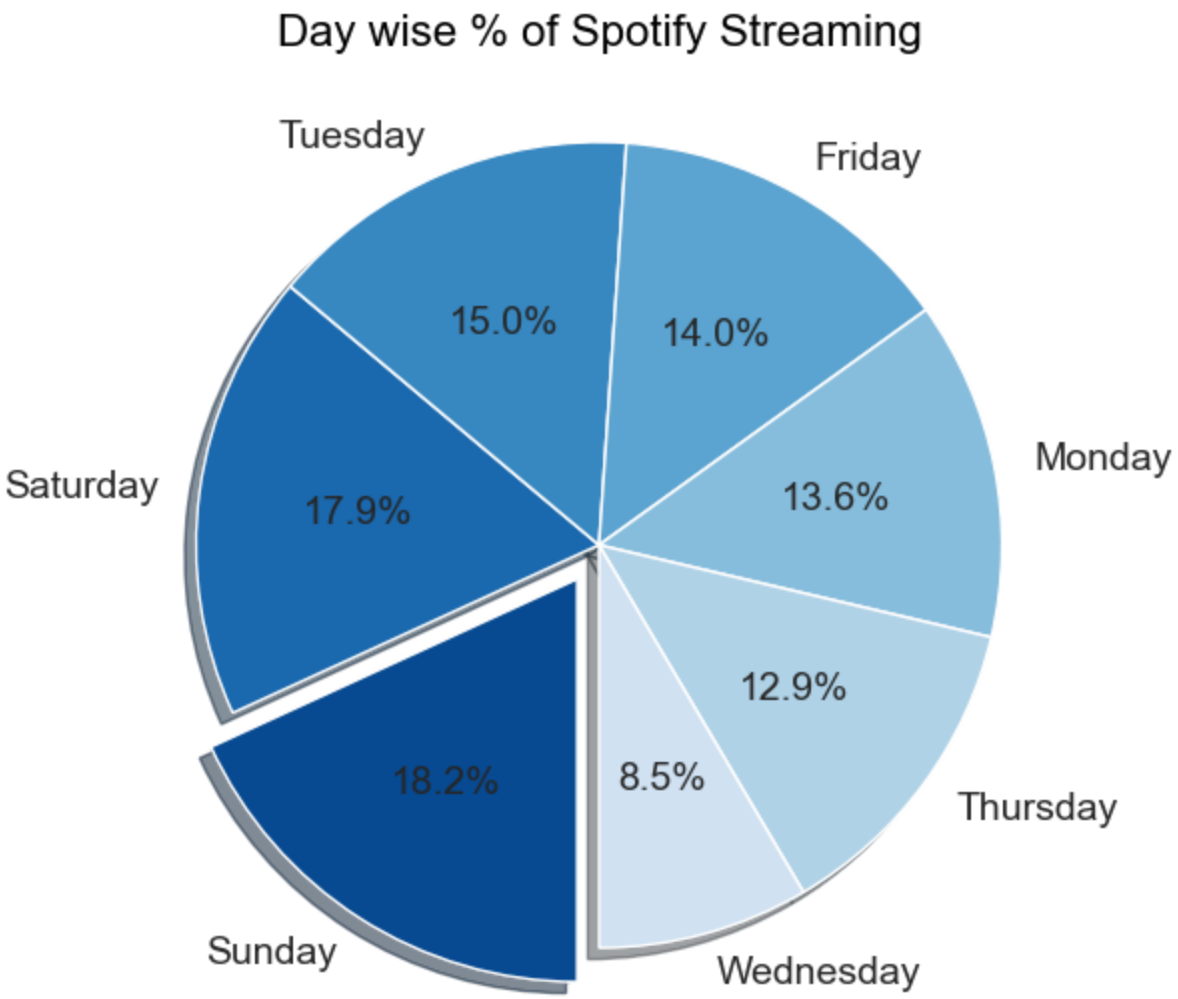


In [ ]:
```
```

3.3 Day Wise Usage (Exploration)

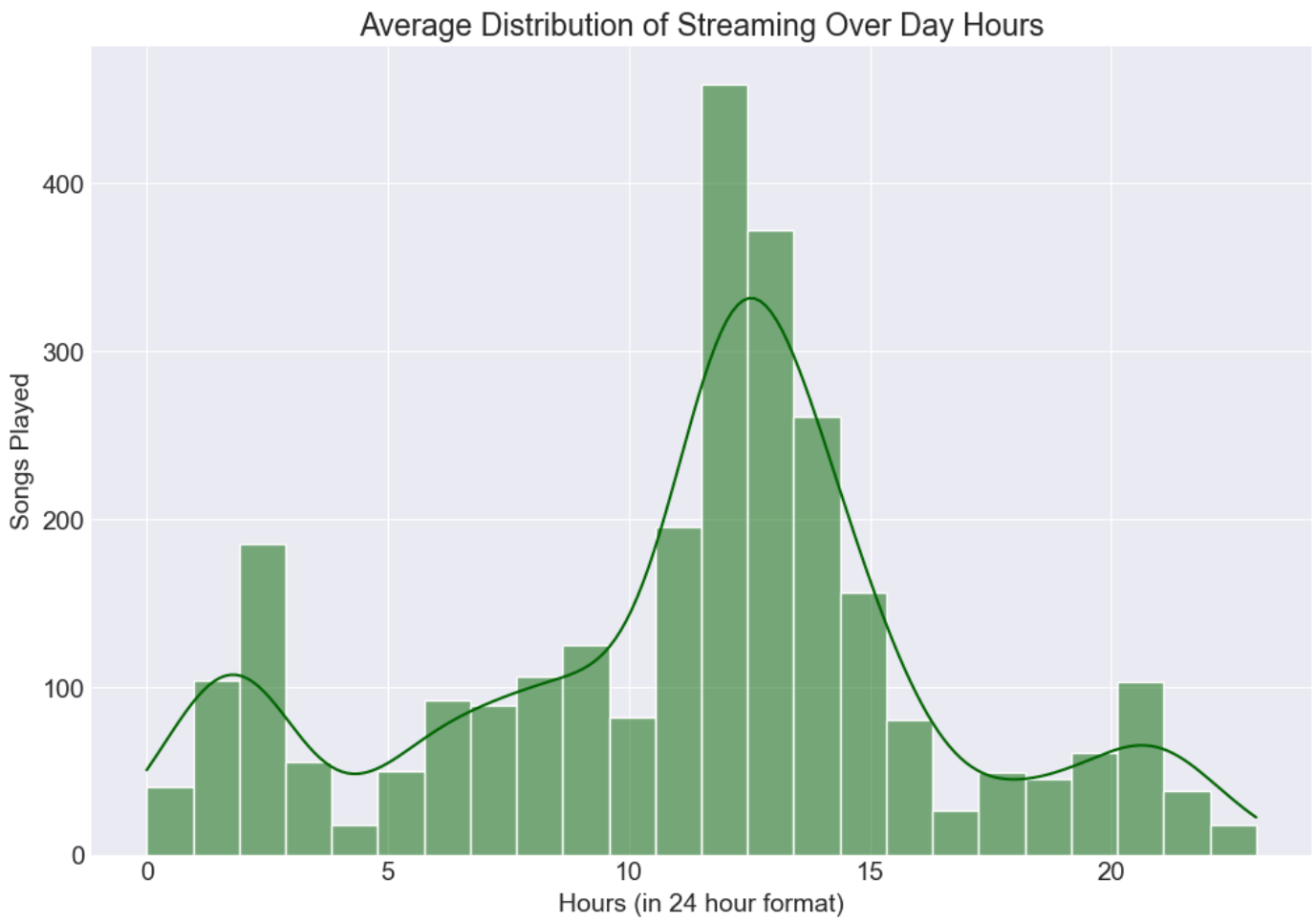
In [39]:
```
import matplotlib.cm as cm

day_name_counts = spotify_stream_df["day-name"].value_counts()
fig = plt.figure(figsize=(8, 6))
ax = fig.add_subplot(111)
colors = cm.Blues(np.linspace(0.9, 0.2, len(day_name_counts)))

# Get the maximum index
max_index = day_name_counts.argmax()

# Create an explode list
                * len(day_name_counts)
```

```
explode[max_index] = 0.1

# Plot the pie chart
ax.pie(day_name_counts, labels=day_name_counts.index, colors=colors, autopct='%1.1f%%',
       textprops={'fontsize': 14}, explode=explode, shadow=True, counterclock=False)

# Set the title and axis aspect ratio
ax.set_title('Day wise % of Spotify Streaming', pad=20, fontdict={'color': 'black', 'wei
ax.axis('equal')

plt.show()
```



3.4 Average Usage over a day (Exploration)

In [40]:
```
fig, ax = plt.subplots(figsize=(12,8))
ax.set(title="Average Distribution of Streaming Over Day Hours",xlabel="Hours (in 24 hou
sns.histplot(spotify_stream_df["hours"], bins=24,kde=True, color="darkgreen");
```



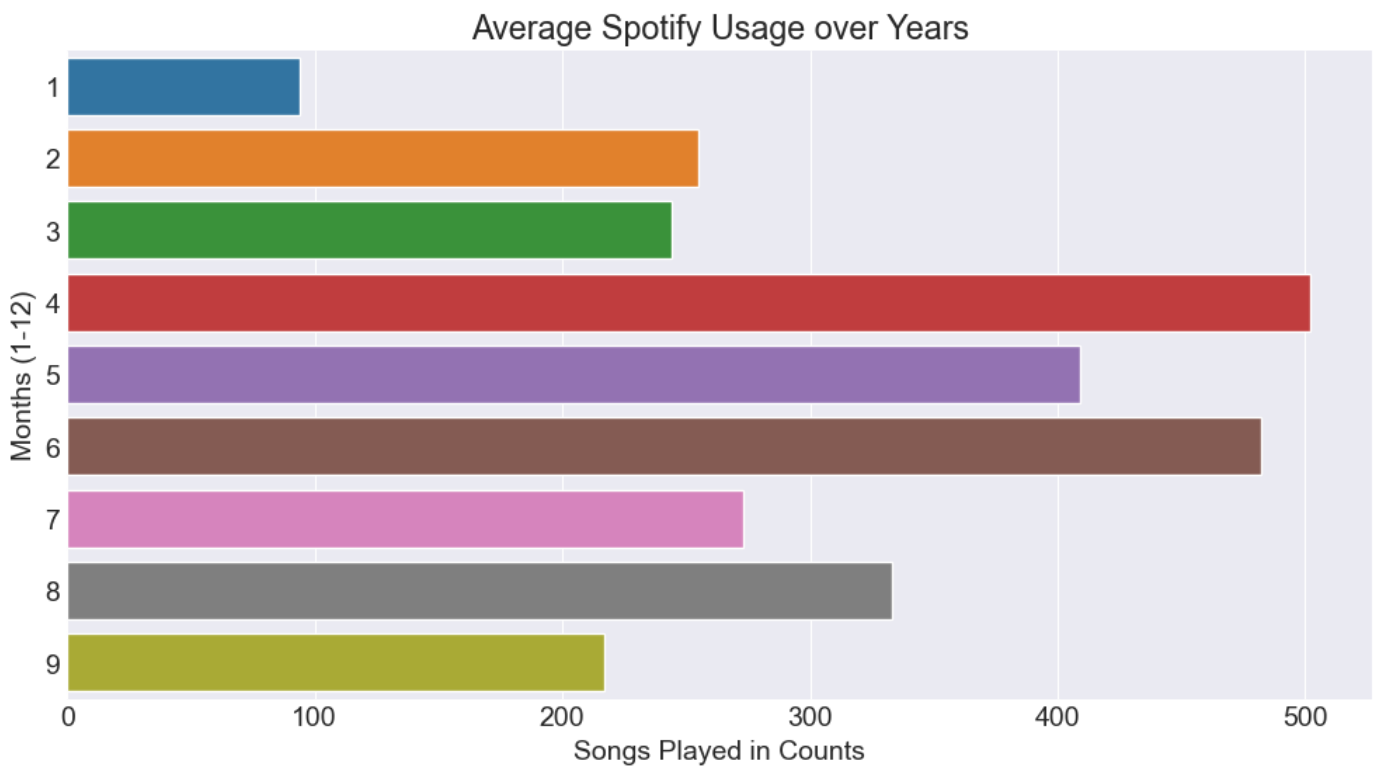
With this histogram graph we can see about my average usage:

1.Maximum around 12-2 PM hour mark

2.Minimum around 9-11 PM hour mark

3.5 Average Usage In a Year (Exploration)

In [41]:
```
fig, ax = plt.subplots(figsize=(12,6))

ax = sns.countplot(y=spotify_stream_df["month"], ax=ax)
ax.set(title="Average Spotify Usage over Years", xlabel="Songs Played in Counts", ylabel
```



How many hours did I spent on Spotify Streaming since the day I signed up for it

Here we want to know, how many hours I spend while streaming spotify since start.

In [42]:
```
time_spent_hours = spotify_stream_df["Listening Time(Hours)"].sum()
time_spent_hours
```

Out[42]:
```
98.90299999999999
```

For this we can simply do a summation of all the time I spent on listening to all songs.

-This comes out to be around **99 Hours.

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

What is actual usage in percentage compared to to the total possible

Here we want to know, what is the percentage of time I spend on spotify.

This question might seem bit odd, but here we want to know that out of maximum possible hours since the start, how much time I actually spent streaming Spotify and we want to calculate that in percentage.

In [43]:
```
date_df = spotify_stream_df["Play-Time"]
time_difference = (date_df.iloc[2808] - date_df.iloc[0]) / np.timedelta64(1,"D")
time_difference_hours = time_difference*24
time_difference_hours
```

Out[43]:
```
6050.183333333333
```

What is the average numbers of songs I played daily

In [44]:
```
total_songs = spotify_stream_df["master_metadata_track_name"].count()
```

In [45]:
```
time_difference
```

Out[45]:
```
252.09097222222223
```

In [46]:
```
average_songs_played_daily = (total_songs / time_difference).round()
average_songs_played_daily
```

Out[46]:
```
11.0
```

Here we can see that on an average I played 11 songs per day

In [ ]:
```

```

Some More Observations

On which day I played maximum number of songs via scatterplot

In [47]:
```
spotify_stream_df["date"] = spotify_stream_df["Play-Time"].dt.date # Creating a new colu
```

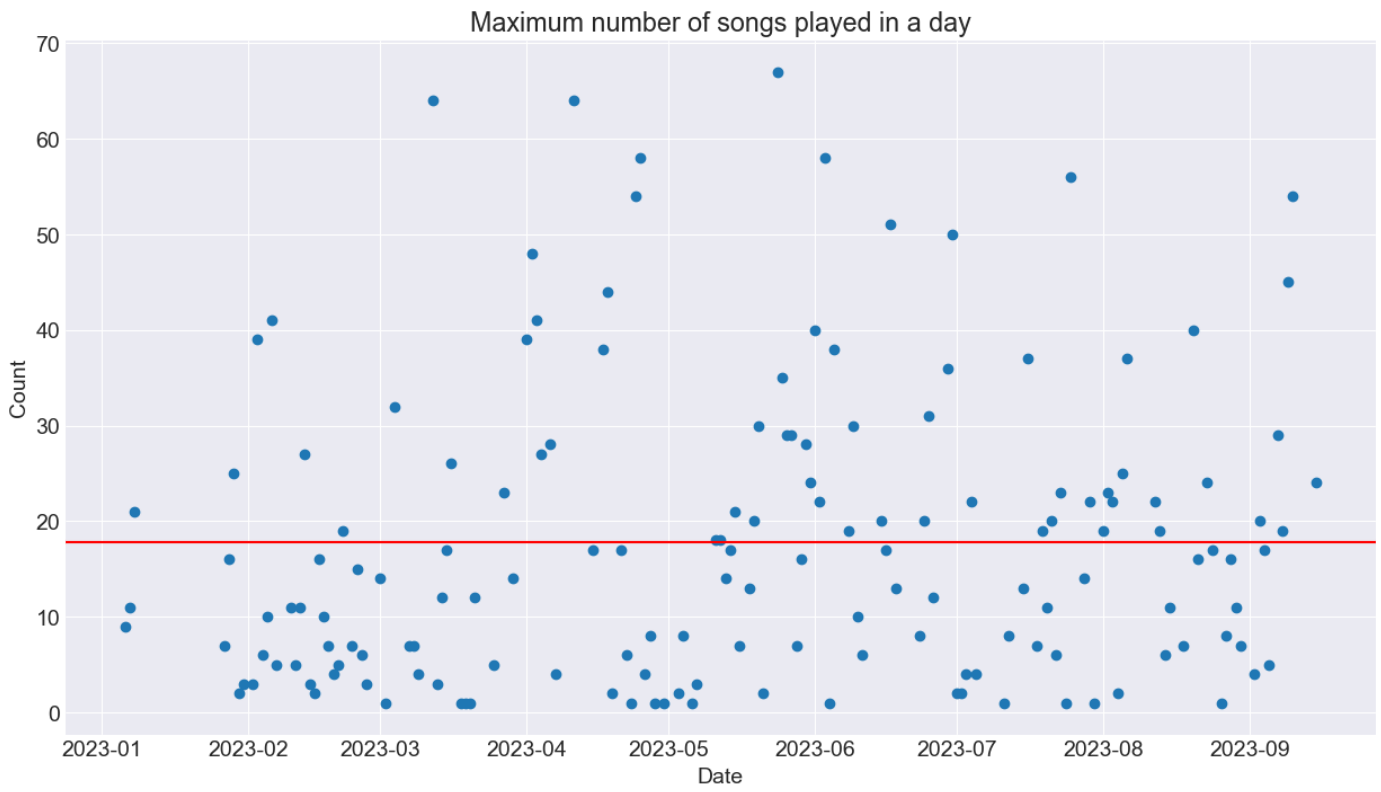
In [48]:
```
most_songs = spotify_stream_df.groupby(["date"])[["Count"]].sum().sort_values(by="Count"
most_songs.head(1)
```

Out[48]:

	Count
date	
2023-05-24	67

Here we can see that I played most songs which is 67 Songs on 24th May 2023

In [49]:
```
fig,ax = plt.subplots(figsize=(15,8))
ax.scatter(most_songs.index,most_songs["Count"]);
ax.set(title="Maximum number of songs played in a day",xlabel="Date",ylabel="Count");
ax.axhline(most_songs["Count"].mean(), linestyle="-", color="r");
```



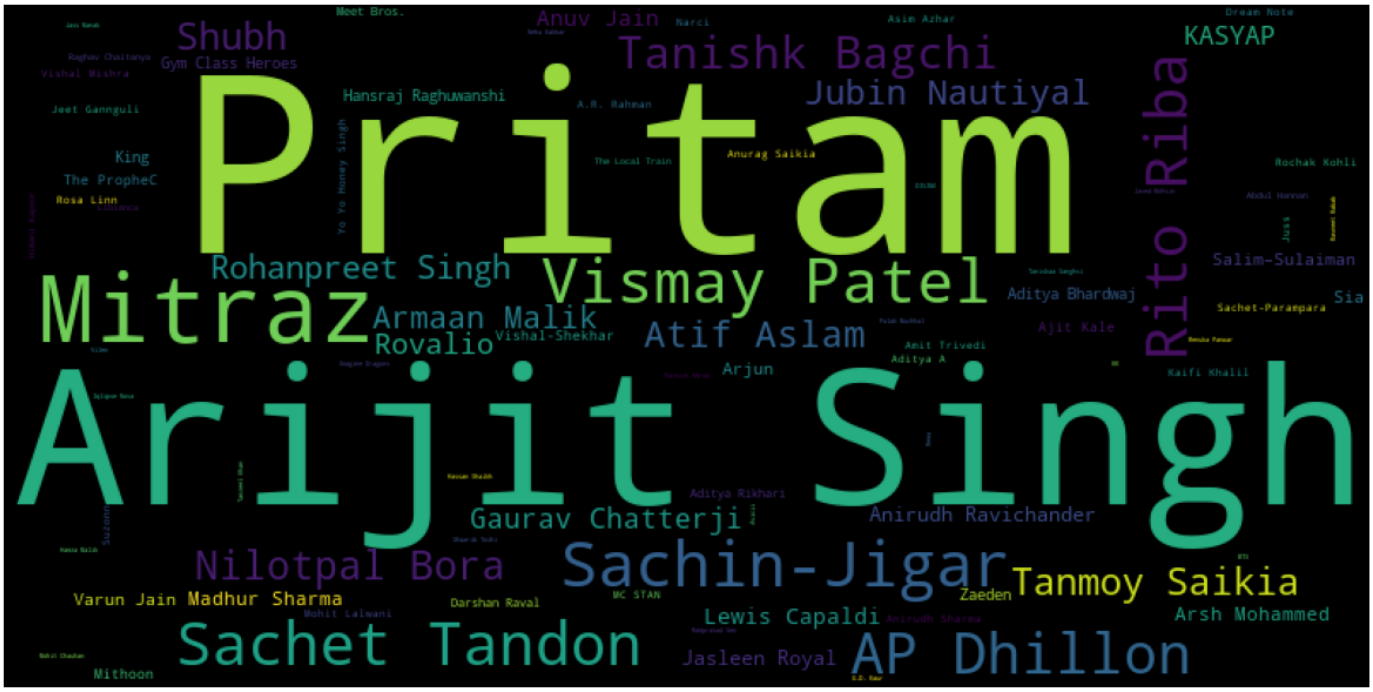
In [ ]:

My favourite 100 Artist in word could

In [50]:
```
fav_artist = spotify_stream_df.groupby(["master_metadata_album_artist_name"])["Count"].c
fav_artist.sort_values(ascending=False).head(100)
```

Out[50]:
```
master_metadata_album_artist_name
Pritam          213
Arijit Singh    181
Mitraz           90
Sachin-Jigar     61
Vismay Patel     60
                ...
Ed Sheeran        6
Bayaan            6
Ash King          6
Badshah           6
Anson Seabra      6
Name: Count, Length: 100, dtype: int64
```

In [51]:
```
def plot_cloud(wordcloud):
    fig = plt.figure(figsize=(15,8))
    plt.imshow(wordcloud)
    plt.axis("off");
wordcloud = WordCloud(width=800,height=400, max_words=100,relative_scaling=1,normalize_p
                      collocations=False).generate_from_frequencies(fav_artist)
plot_cloud(wordcloud)
```



My Favourite Artist playlist based on count of songs.

In [52]:
```
my_fav_artist_playlist = spotify_stream_df.groupby(["master_metadata_album_artist_name",
my_fav_artist_playlist
```

Out[52]:

		Count
master_metadata_album_artist_name	**master_metadata_track_name**	
Mitraz	**Alfaazo**	53
Sachin-Jigar	**Phir Aur Kya Chahiye (From "Zara Hatke Zara Bachke")**	46
Rohanpreet Singh	**Gham Khushiyan**	33
Rovalio	**Iraaday**	32
Tanmoy Saikia	**Kenekoi Bujao ("From 13")**	32
...	**...**	...
Arijit Singh	**Galtiyan**	6
Dikshant	**Aankhon Se Batana**	6
Mohit Lalwani	**Samay Samjhayega (Lofi)**	6
Pritam	**Tum Kya Mile (From "Rocky Aur Rani Kii Prem Kahaani")**	6
	What Jhumka ? (From "Rocky Aur Rani Kii Prem Kahaani")	6

100 rows × 1 columns

My Favourite 100 Songs In Word Cloud.

To see what are top 100 songs that I usually listen to.

In [53]:
```
fav_songs = spotify_stream_df.groupby(["master_metadata_track_name"])["Count"].count()
```

In [54]:
```
def plot_cloud(wordcloud):
    fig = plt.figure(figsize=(15,8))
    plt.imshow(wordcloud)
    plt.axis("off");
wordcloud = WordCloud(width=800,height=400, max_words=100,relative_scaling=1,normalize_p
                      collocations=False).generate_from_frequencies(fav_songs)
plot_cloud(wordcloud)
```

Most Usage (Active usage in a day over a week) via a heatmap.

In [55]:
```
active_usage = spotify_stream_df.groupby(['hours', 'day-name'])['master_metadata_album_a
active_usage_pivot = active_usage.pivot("hours", 'day-name', 'master_metadata_album_arti
active_usage_pivot.head()
```

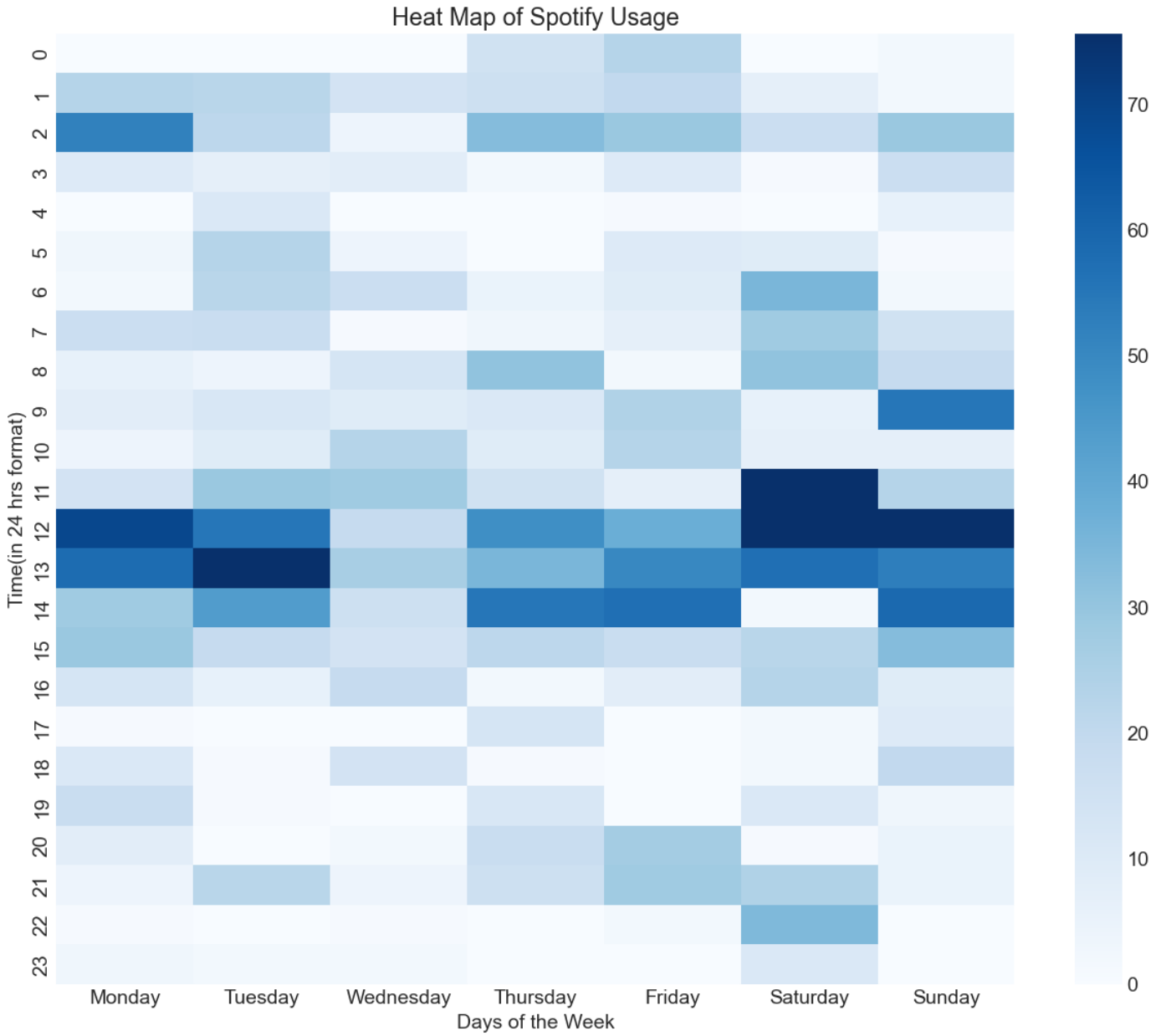
```
C:\Users\sawra\AppData\Local\Temp\ipykernel_6644\1012325590.py:2: FutureWarning: In a fu
ture version of pandas all arguments of DataFrame.pivot will be keyword-only.
  active_usage_pivot = active_usage.pivot("hours", 'day-name', 'master_metadata_album_ar
tist_name')
```

Out[55]:

day-name	Friday	Monday	Saturday	Sunday	Thursday	Tuesday	Wednesday
hours							
0	23.0	NaN	NaN	2.0	15.0	NaN	NaN
1	20.0	23.0	7.0	2.0	16.0	22.0	14.0
2	29.0	52.0	17.0	29.0	33.0	21.0	4.0
3	10.0	10.0	1.0	17.0	2.0	7.0	8.0
4	1.0	NaN	NaN	6.0	NaN	11.0	NaN

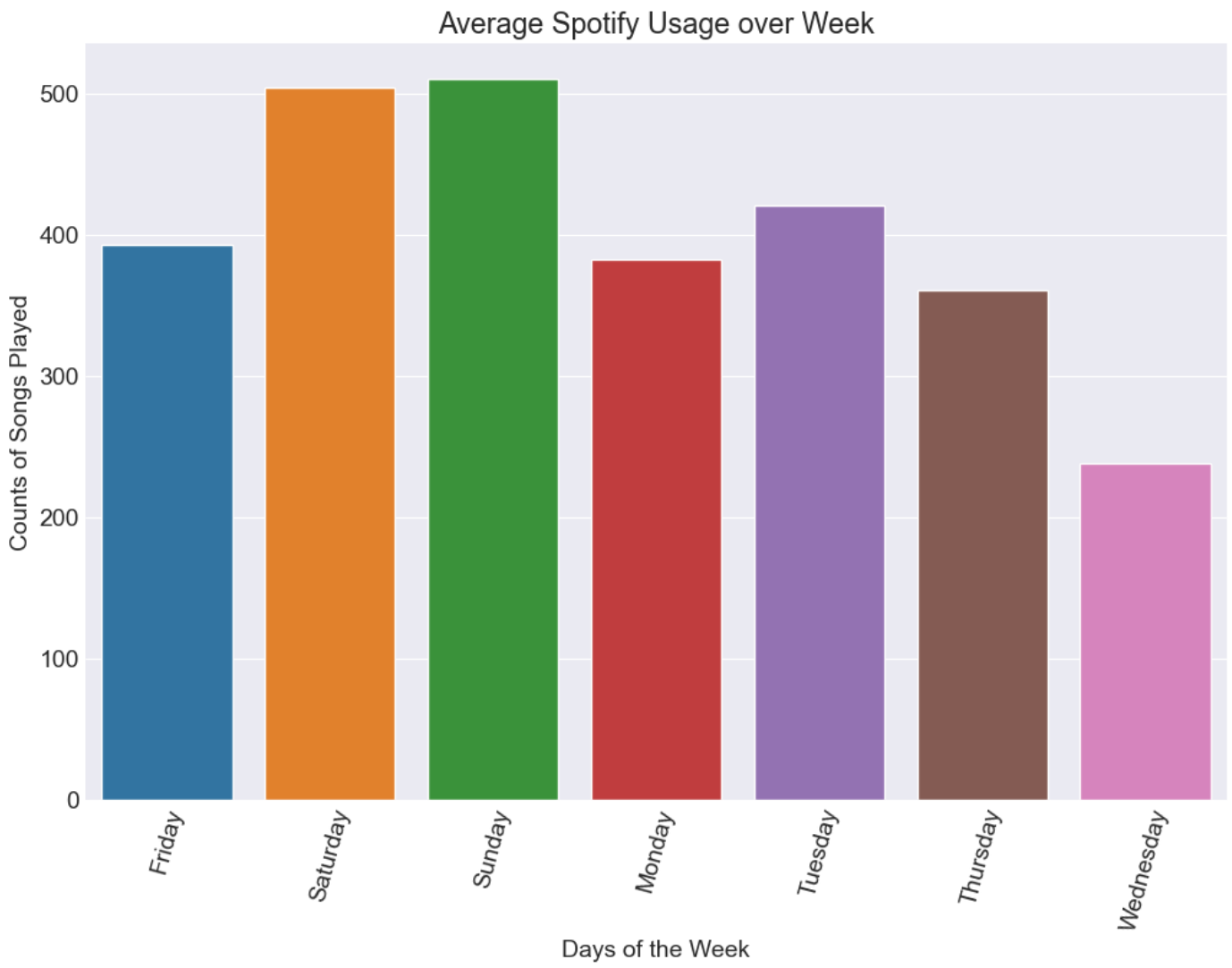
In [56]:
```
days = ["Monday", 'Tuesday', "Wednesday", "Thursday", "Friday", "Saturday", "Sunday"]

fig, ax = plt.subplots(figsize=(15,12))
```

```
ax = sns.heatmap(active_usage_pivot[days].fillna(0), robust=True, cmap="Blues", ax = ax)
ax.set(title="Heat Map of Spotify Usage", xlabel="Days of the Week",ylabel="Time(in 24 h
```



Usage Analysis over a week via countplot

In [57]:
```
fig, ax = plt.subplots(figsize=(12,8))
ax = sns.countplot(x=spotify_stream_df["day-name"],ax=ax)
plt.xticks(rotation=75);
ax.set(title="Average Spotify Usage over Week",xlabel="Days of the Week",ylabel="Counts
```



What is the percentage of usage distribution between Weekday and Weekend

In [58]:
```
extra_df = spotify_stream_df.copy()
extra_df['is_weekend'] = extra_df["day-name"].isin(['Sunday','Saturday'])
weekday_vs_weekend = extra_df.groupby(['is_weekend'])[['Count']].sum()
weekday_vs_weekend
```

Out[58]:

is_weekend	Count
False	1795
True	1014

In [59]:
```
weekday_vs_weekend["Percentage"] = weekday_vs_weekend["Count"]/weekday_vs_weekend["Count
weekday_vs_weekend
```

Out[59]:

is_weekend	Count	Percentage
False	1795	63.901744
True	1014	36.098256

In [60]:
```
fig, (ax1,ax2) = plt.subplots(1,2,figsize=(18,6))
ax1 = sns.barplot(x=["False","True"],y="Count",data=weekday_vs_weekend,ax=ax1)
ax1.set(title="Weekday vs Weekend",xlabel="Is it Weekend",ylabel="Counts of Songs Played

ax2 = sns.barplot(x=["False","True"],y="Percentage",data=weekday_vs_weekend,color="Olive
ax2.set(title="Weekday vs Weekend (Percentage)",xlabel="Is it Weekend",ylabel="Percentag
```

In [ ]:

In [ ]:

In [ ]:

In [ ]: