In [2]:
```
import os
import pandas as pd
import csv
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import matplotlib.pyplot as plt
from wordcloud import WordCloud
from dateutil.relativedelta import relativedelta
```

```
C:\Users\majum\Anaconda3\lib\site-packages\pandas\compat\_optional.py:138: UserWarning: Pandas requires version '2.7.0' or newer
of 'numexpr' (version '2.6.8' currently installed).
  warnings.warn(msg, UserWarning)
```

In [ ]:
```
print(os.getcwd())
```

In [ ]:
```
os.chdir('C:\\Users\\majum\\Documents\\Python scripts')
os.getcwd()
```

Data Setup

In [3]:
```
file1 = "C:\\Users\\majum\\Documents\\Python scripts\\userid-profile.tsv"
file2 = "C:\\Users\\majum\\Documents\\Python scripts\\userid-timestamp-artid-artname-traid-traname.tsv"
df1 = pd.read_csv(file1,sep='\t')

headers = ["userid", "timestamp","musicbrainz-artist-id", "artist-name","musicbrainz-track-id","track-name"]
df2 = pd.read_csv(file2,sep='\t', header = None, names=headers)
```

In [ ]:
```
# Explore the data
df1.head()
df2.head()
```

In [241]:
```
#If the number of distinct users is the same in both tables then we can assume that each user has their session activity in this d
ataset
if df1['#id'].nunique() == df2['userid'].nunique():

    print("The number of unique users is " + str(df2['userid'].nunique()))
```

```
The number of unique users is 992
```

Merge the User data with the Listening data using an inner join

In [4]:
```
df = pd.merge(df1, df2, left_on='#id', right_on ="userid")
```

Find the earliest data in the records, and assume this is the Launch Date

In [7]:
```
df1['registered'] = pd.to_datetime(df1['registered'])

df1['registered'].min()
```

Out[7]:
```
Timestamp('2002-10-29 00:00:00')
```

In [11]:
```
df2['timestamp'] = pd.to_datetime(df2['timestamp'])

df2['timestamp'].min()
```

Out[11]:
```
Timestamp('2005-02-14 00:00:07+0000', tz='UTC')
```

The launch date is assumed to be 2002-10-29

In [12]:
```
launch = pd.to_datetime('2002-10-29', format='%Y-%m-%d' )
```

In [14]:
```
### Format the data to appropriate datatype/ Convert to datetime

df['timestamp_date'] = pd.to_datetime(df['timestamp'])
df['timestamp_date'] = df['timestamp_date'].dt.strftime('%Y-%m-%d')
df['registered'] = pd.to_datetime(df['registered'])
```

Create a 'Day Since Launch' field

```
In [ ]: # Subtract launch date from timestamp dates to find the days that have elapsed since launch
        df['daysincelaunch'] = (df['timestamp_date'] -launch).dt.days
```

Create a 'Month Since Launch' field

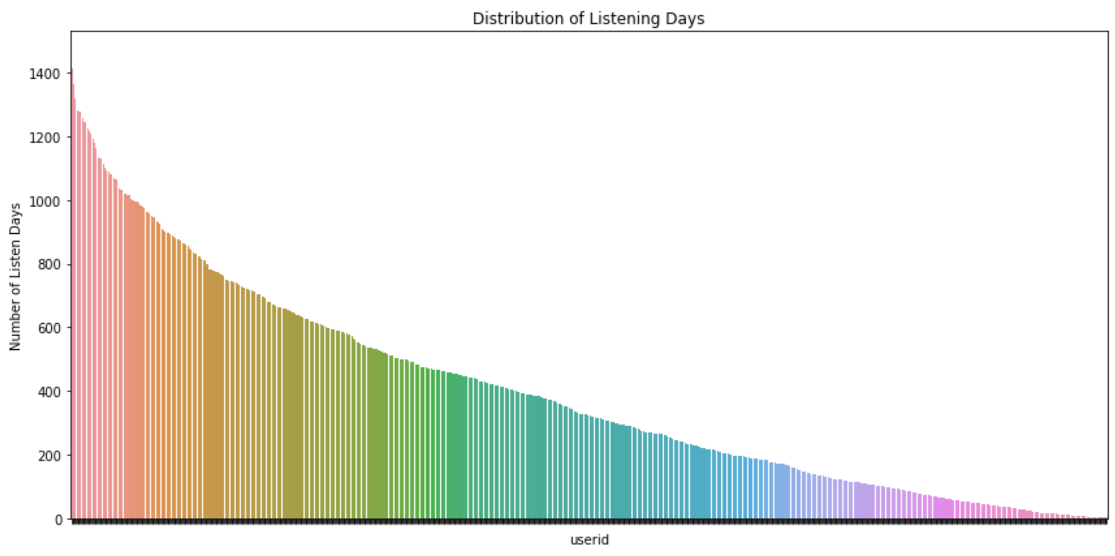
```
In [ ]: # df = df.drop(columns=['months_difference'])

        def calculate_months_difference(row):
            date_diff = relativedelta(row['timestamp_date'], launch)
            return date_diff.years * 12 + date_diff.months

        df['monthsincelaunch'] = df.apply(calculate_months_difference, axis=1)
```

```
In [ ]: def convert_to_int(value):
            if np.isfinite(value):
                return int(value)
            return value

        # Apply the custom function to the 'float_column'
        df['daysincelaunch'] = df['daysincelaunch'].apply(convert_to_int)
```

Explore the distribution of listening days to understand listening behaviour across the user base

```
In [ ]: a = df.groupby(['userid']).agg(sessiondays=('timestamp_date','nunique'))
        distr = a.sort_values(by='sessiondays', ascending=False).reset_index()
```

```
In [17]: plt.figure(figsize=(12,6))
         ax=sns.barplot(x='userid', y='sessiondays', data=distr)
         plt.title('Distribution of Listening Days')
         ax.set_xticklabels([])
         plt.ylabel('Number of Listen Days')
         plt.label('Number of Listen Days')
         plt.xticks(rotation=90)
         plt.tight_layout()
         plt.show()
```



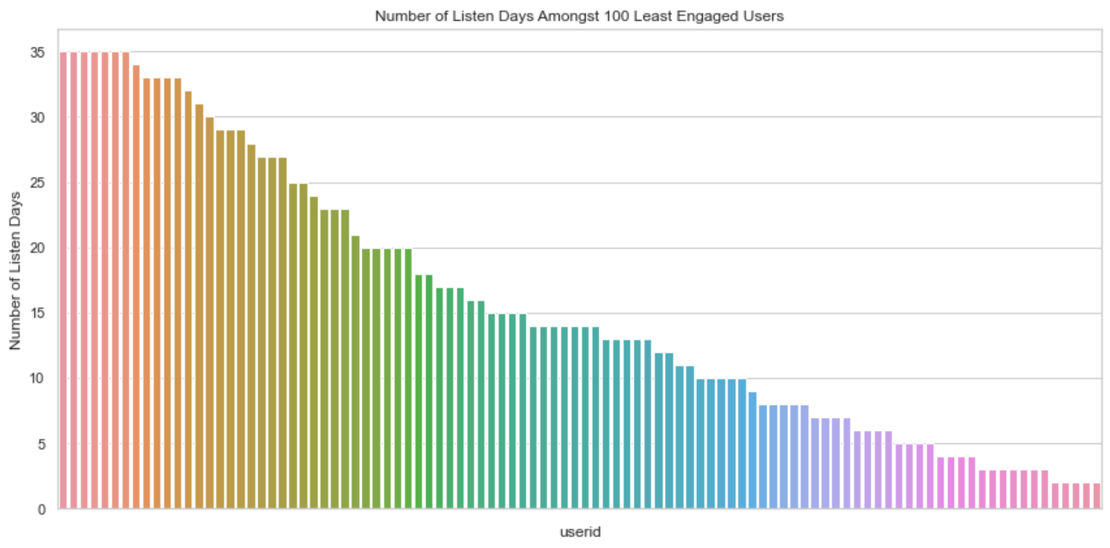
Since distribution is right skewed, median would be a good measure of central tendency

Median Number of Listening Days (Median Session Days) observed in Users

```
In [18]: sd =a.sort_values(by='sessiondays', ascending=False).tail(100).reset_index()

         # Most Common Sessions Per User (Mode)
         print("The Median number of Listening Days was " + str(a['sessiondays'].median()) + " before users stopped usage")

         The Median number of Listening Days was 322.5 before users stopped usage
```

```
In [228]: #To help us understand the lower end of engagement behaviour, plot the 100 least engaged users

          #Top Tracks
          plt.figure(figsize=(12,6))
          ax=sns.barplot(x='userid', y='sessiondays', data=sd)
          plt.title('Distribution of Listen Days Amongst 100 Least Engaged Users')
          ax.set_xticklabels([])
          plt.ylabel('Number of Listen Days')
          plt.xticks(rotation=90)
          plt.tight_layout()
          plt.show()
```

```
Users had a Median Listening Day of 322.5 before stopping usage
```



The Median Duration of Engagement was:

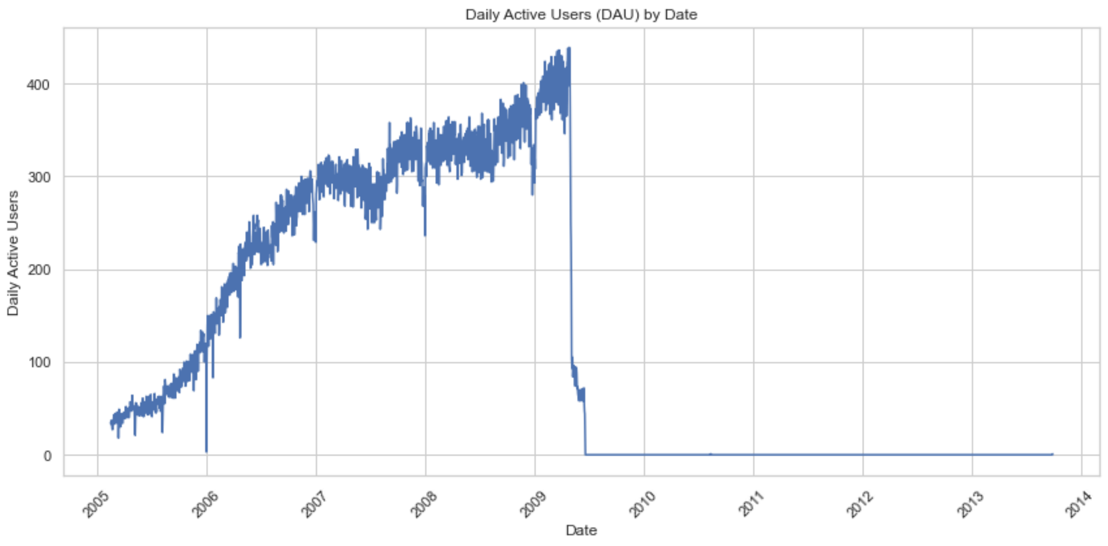
```
In [249]: # engdays = df.groupby('userid')['timestamp_date'].apply(lambda x: (x.max() - x.min()).days).reset_index()
          print ("The Median Duration of Engagement was " +
                 str(engdays['timestamp_date'].median()) + " days between first session and last session.")
```

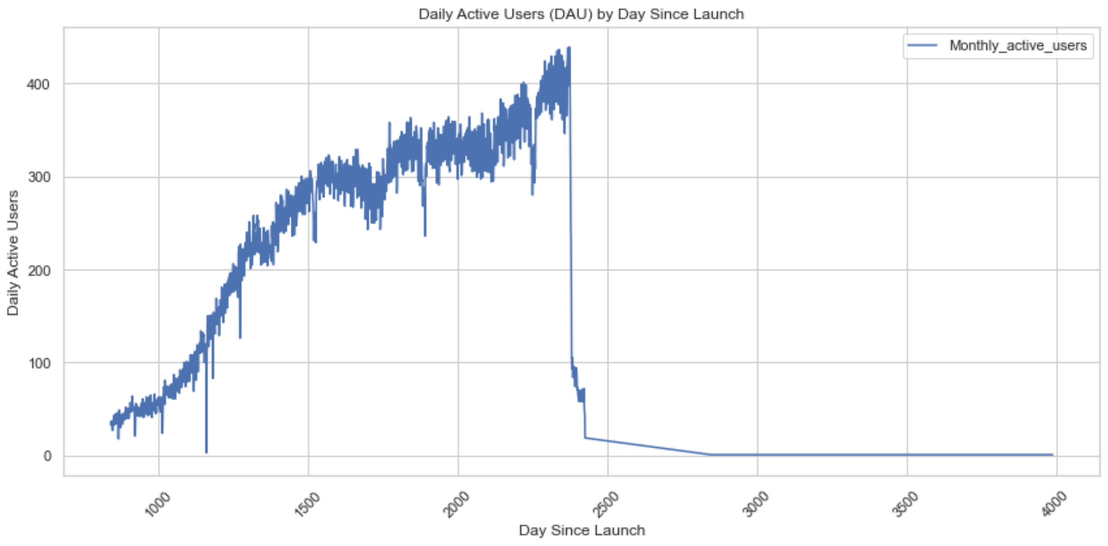
```
The Median Duration of Engagement was 920.0 days between first session and last session.
```

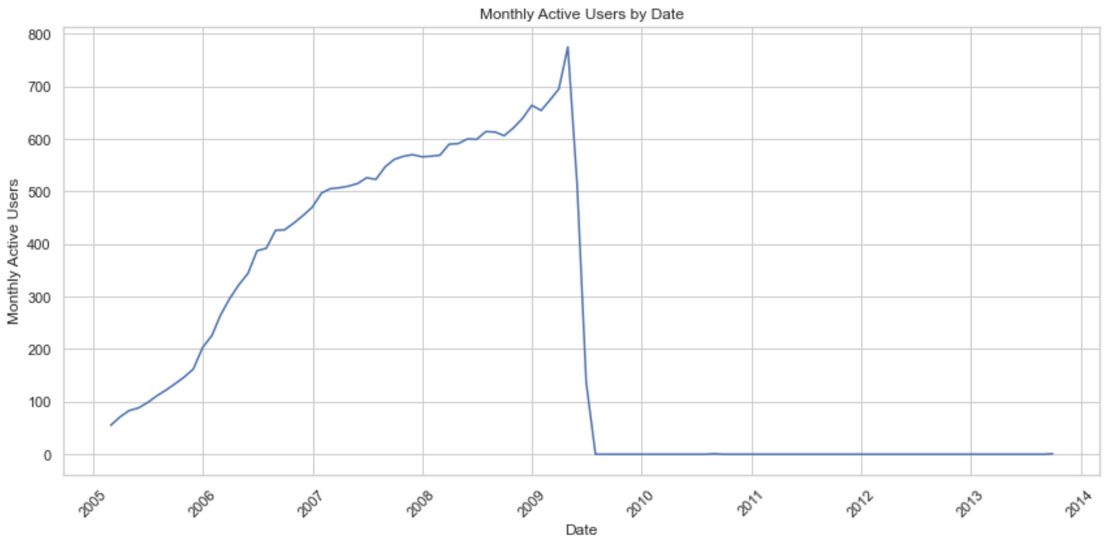
```
In [ ]: # Calculate daily active users (DAU)
        df_dau = df.set_index('timestamp_date').groupby(pd.Grouper(freq='D'))['userid'].nunique()

        # # Calculate daily active users (MAU) by day since launch
        df_dau2 = df.groupby(['daysincelaunch']).agg(Monthly_active_users =
                                   ('userid','nunique'))

        # Calculate monthly active users (MAU)
        df_mau = df.set_index('timestamp_date').groupby(pd.Grouper(freq='M'))['userid'].nunique()

        # # Calculate monthly active users (MAU) by month since launch
        df_mau2 = df.groupby(['monthsincelaunch']).agg(Monthly_active_users =
                                   ('userid','nunique'))

        # Create a DataFrame with MAU, DAU, and date indices
        result_df = pd.DataFrame({'MAU': df_mau, 'DAU': df_dau})

        # Calculate the ratio of DAU to MAU
        result_df['DAU/MAU'] = result_df['DAU'] / result_df['MAU']
        result_ratio = result_df['DAU/MAU']
```

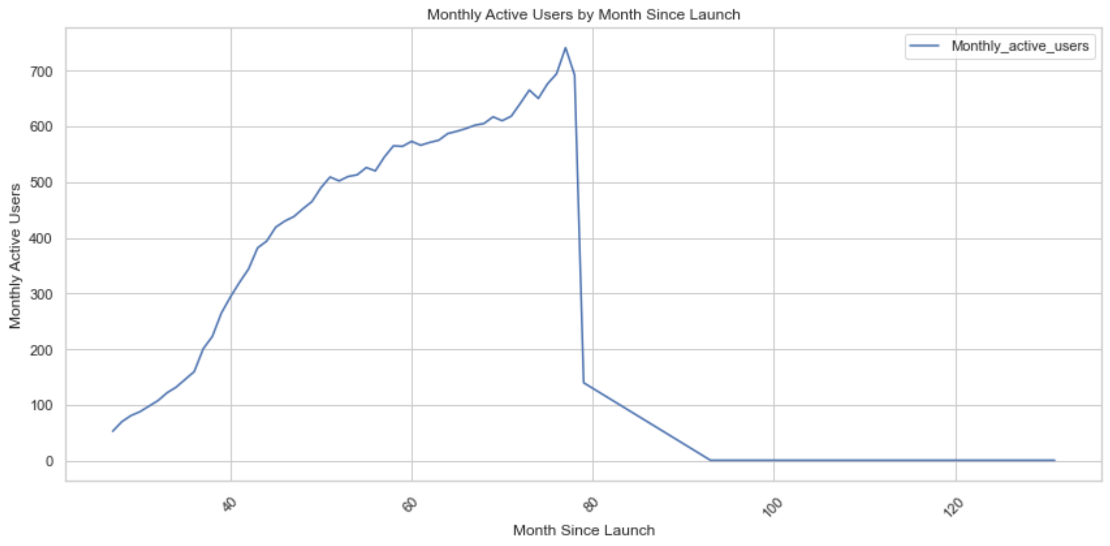
Daily Active Users (DAU), Monthly Active Users (MAU) and DAU to MAU ratio

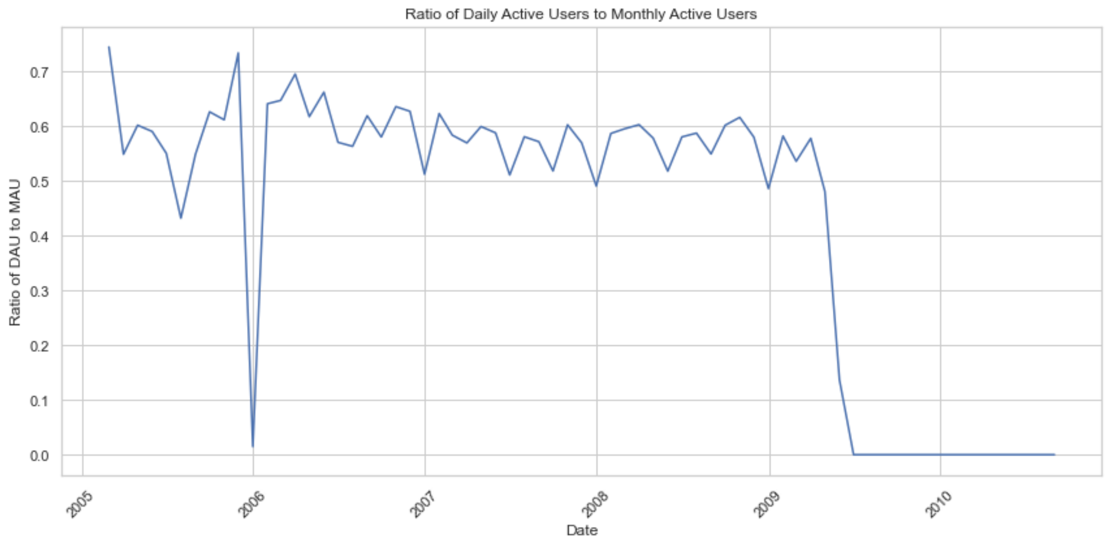
In [247]:
```
%matplotlib inline
pd.set_option('display.max_rows', None)

sns.set(style="whitegrid")


plt.figure(figsize=(12,6))
sns.lineplot(data=df_dau)
plt.title('Daily Active Users (DAU) by Date')
plt.xlabel('Date')
plt.ylabel('Daily Active Users')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()

plt.figure(figsize=(12,6))
sns.lineplot(data=df_dau2)
plt.title('Daily Active Users (DAU) by Day Since Launch')
plt.xlabel('Day Since Launch')
plt.ylabel('Daily Active Users')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()


# # Plot a line graph of monthly active users
plt.figure(figsize=(12, 6))
sns.lineplot(data=df_mau)#, color ='orange')
plt.title('Monthly Active Users by Date')
plt.xlabel('Date')
plt.ylabel('Monthly Active Users')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()

# # Plot a line graph of monthly active users
plt.figure(figsize=(12, 6))
sns.lineplot(data=df_mau2)#, color ='orange')
plt.title('Monthly Active Users by Month Since Launch')
plt.xlabel('Month Since Launch')
plt.ylabel('Monthly Active Users')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()


# # Plot a line graph of monthly active users
plt.figure(figsize=(12, 6))
sns.lineplot(data=result_ratio) #, color ='red')
plt.title('Ratio of Daily Active Users to Monthly Active Users')
plt.xlabel('Date')
plt.ylabel('Ratio of DAU to MAU')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```


Daily Active Users (DAU) by Date
Daily Active Users
Date
0
100
200
300
400
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014


Daily Active Users (DAU) by Day Since Launch
Monthly_active_users
Daily Active Users
Day Since Launch
0
100
200
300
400
1000
1500
2000
2500
3000
3500
4000


Monthly Active Users by Date
Monthly Active Users
Date
0
100
200
300
400
500
600
700
800
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014





```
In [ ]: # subset = df.sample(n=10000, random_state=42)
```

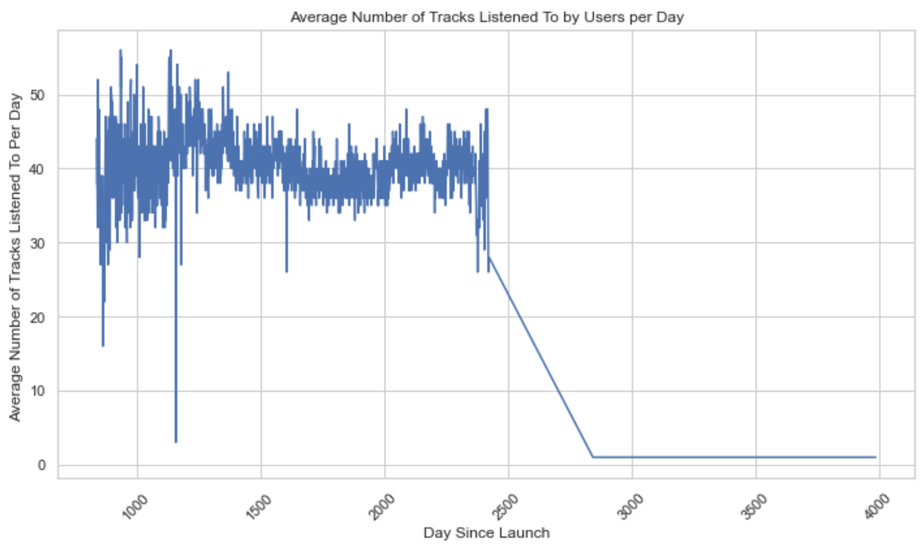
Median Number of Tracks Listened

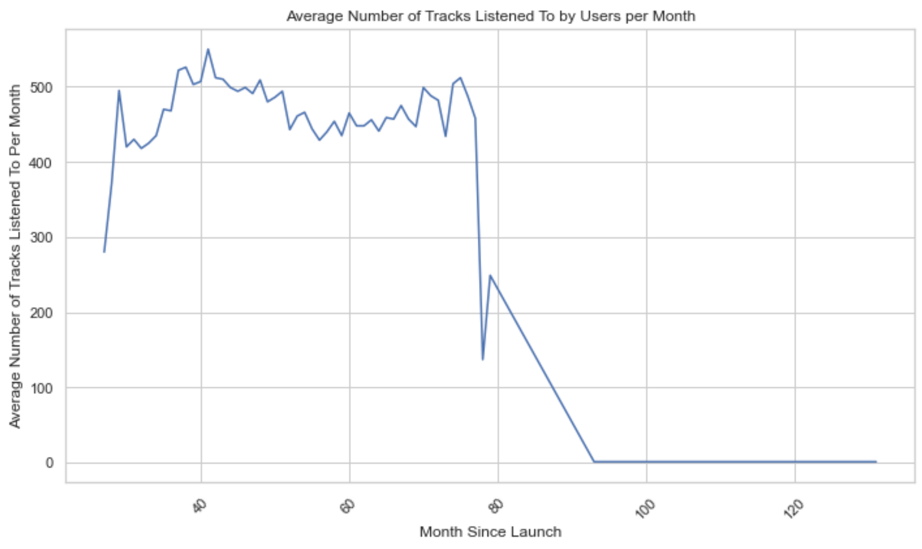
```
In [257]: # test = df.groupby(['daysincelaunch','userid']).agg(tracks =
          #                            ('track-name','nunique')).reset_index()

          # avtracks = test.groupby(['daysincelaunch']).agg(meantracks=('tracks','mean')).round(0)

          print('The Median Number of Tracks Listened to was: '+ str(test['tracks'].median()) + " days")
```

```
The Median Number of Tracks Listened to was: 26.0 days
```

Daily Average Number of Tracks Listened to and Monthly Average Number of Tracks Listened to by Users

In [258]:
```
sns.set(style="whitegrid")

# Create a line plot of the average session duration per day
plt.figure(figsize=(10, 6))
sns.lineplot(data=avtracks, x='daysincelaunch', y='meantracks')

plt.title('Average Number of Tracks Listened To by Users per Day')
plt.xlabel('Day Since Launch')
plt.ylabel('Average Number of Tracks Listened To Per Day ')
plt.xticks(rotation=45)
plt.tight_layout()

plt.show()


# testm = df.groupby(['monthsincelaunch','userid']).agg(tracks =
#                              ('track-name','nunique')).reset_index()

# avtracksm = testm.groupby(['monthsincelaunch']).agg(meantracks=('tracks','mean')).round(0)

# Create a line plot of the average session duration per day
plt.figure(figsize=(10, 6))
sns.lineplot(data=avtracksm, x='monthsincelaunch', y='meantracks')

plt.title('Average Number of Tracks Listened To by Users per Month')
plt.xlabel('Month Since Launch')
plt.ylabel('Average Number of Tracks Listened To Per Month')
plt.xticks(rotation=45)
plt.tight_layout()

plt.show()
```





Tracks and Artists Ranked by Listen Count

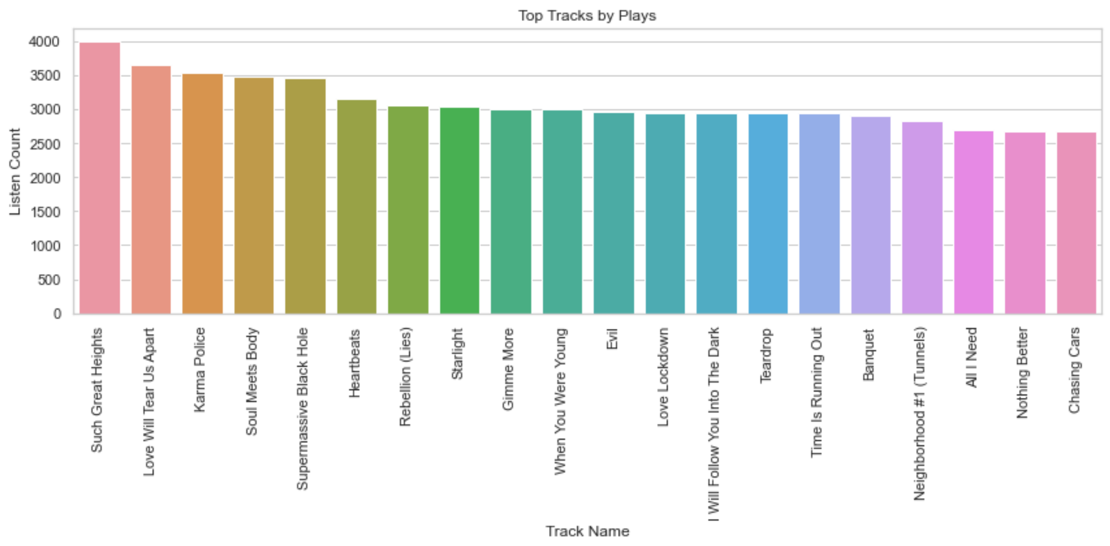
In [259]:
```
x= df.groupby(['track-name','artist-name']).agg(listencount=('userid','count'))

top = x.sort_values(by='listencount', ascending=False).head(100).reset_index()
top
```

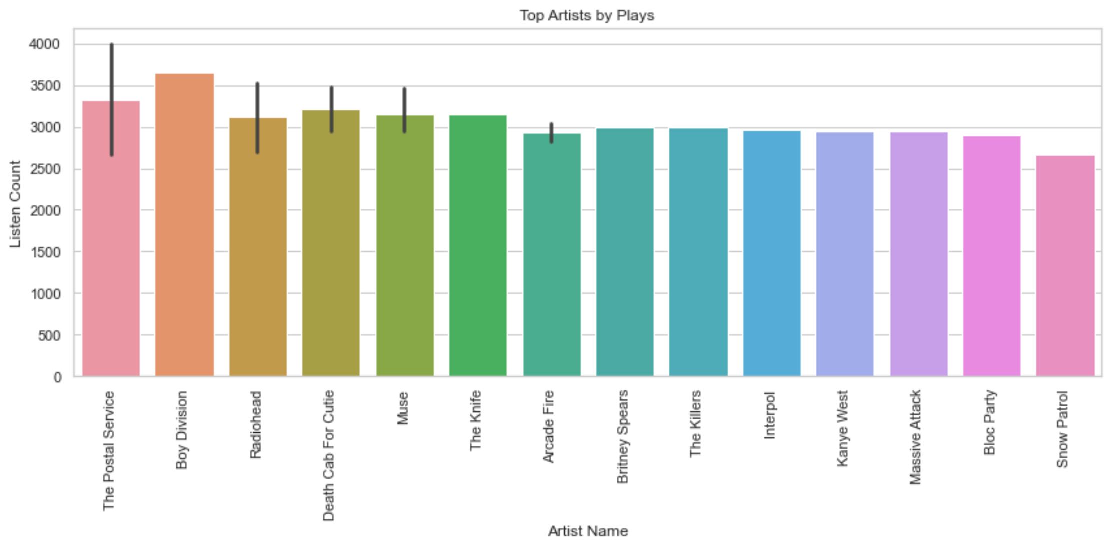
Out[259]:

	track-name	artist-name	listencount
0	Such Great Heights	The Postal Service	3991
1	Love Will Tear Us Apart	Boy Division	3651
2	Karma Police	Radiohead	3533
3	Soul Meets Body	Death Cab For Cutie	3479
4	Supermassive Black Hole	Muse	3463
5	Heartbeats	The Knife	3155
6	Rebellion (Lies)	Arcade Fire	3047
7	Starlight	Muse	3040
8	Gimme More	Britney Spears	3002
9	When You Were Young	The Killers	2997
10	Evil	Interpol	2962
11	Love Lockdown	Kanye West	2950
12	I Will Follow You Into The Dark	Death Cab For Cutie	2947
13	Teardrop	Massive Attack	2941
14	Time Is Running Out	Muse	2941
15	Banquet	Bloc Party	2903
16	Neighborhood #1 (Tunnels)	Arcade Fire	2826
17	All I Need	Radiohead	2694
18	Nothing Better	The Postal Service	2670
19	Chasing Cars	Snow Patrol	2667
20	Creep	Radiohead	2651
21	Heartless	Kanye West	2644
22	15 Step	Radiohead	2643
23	Womanizer	Britney Spears	2635
24	Nude	Radiohead	2635
25	Crazy	Gnarls Barkley	2600
26	Reckoner	Radiohead	2585
27	Time To Pretend	Mgmt	2584
28	Paranoid Android	Radiohead	2567
29	The Scientist	Coldplay	2563
30	I Bet You Look Good On The Dancefloor	Arctic Monkeys	2553
31	Wonderwall	Oasis	2519
32	Fake Plastic Trees	Radiohead	2490
33	Neighborhood #3 (Power Out)	Arcade Fire	2444
34	She'S Lost Control	Joy Division	2435
35	Jigsaw Falling Into Place	Radiohead	2416
36	Viva La Vida	Coldplay	2387
37	All These Things That I'Ve Done	The Killers	2384
38	Welcome To Heartbreak (Feat. Kid Cudi)	Kanye West	2376
39	Enjoy The Silence	Depeche Mode	2365
40	The District Sleeps Alone Tonight	The Postal Service	2365
41	Amazing (Feat. Young Jeezy)	Kanye West	2354
42	Take Me Out	Franz Ferdinand	2353
43	Kids	Mgmt	2333
44	Heartbeats	José González	2332
45	Slow Hands	Interpol	2322
46	Somebody Told Me	The Killers	2316
47	Wish You Were Here	Pink Floyd	2314
48	Obstacle 1	Interpol	2310
49	Maps	Yeah Yeah Yeahs	2308
50	Bodysnatchers	Radiohead	2292
51	New Slang	The Shins	2279
52	Everything In Its Right Place	Radiohead	2275
53	Knights Of Cydonia	Muse	2269
54	Paranoid (Feat. Mr. Hudson)	Kanye West	2261
55	Hysteria	Muse	2261

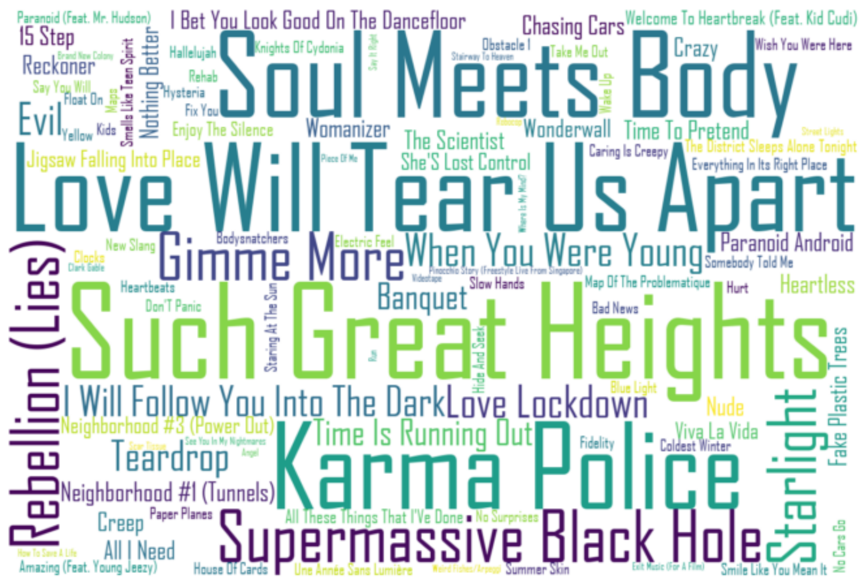
	track-name	artist-name	listencount
56	Map Of The Problematique	Muse	2257
57	Caring Is Creepy	The Shins	2245
58	Clocks	Coldplay	2245
59	House Of Cards	Radiohead	2239
60	Paper Planes	M.I.A.	2238
61	Hide And Seek	Imogen Heap	2236
62	Don'T Panic	Coldplay	2231
63	Say You Will	Kanye West	2229
64	Smells Like Teen Spirit	Nirvana	2227
65	No Surprises	Radiohead	2224
66	Blue Light	Bloc Party	2216
67	Fix You	Coldplay	2213
68	Staring At The Sun	Tv On The Radio	2209
69	Hallelujah	Jeff Buckley	2202
70	Coldest Winter	Kanye West	2201
71	No Cars Go	Arcade Fire	2194
72	Smile Like You Mean It	The Killers	2180
73	Float On	Modest Mouse	2175
74	Bad News	Kanye West	2163
75	Rehab	Amy Winehouse	2158
76	Yellow	Coldplay	2157
77	Electric Feel	Mgmt	2155
78	Summer Skin	Death Cab For Cutie	2115
79	Hurt	Johnny Cash	2112
80	Fidelity	Regina Spektor	2109
81	Une Année Sans Lumière	Arcade Fire	2109
82	Wake Up	Arcade Fire	2108
83	Pinocchio Story (Freestyle Live From Singapore)	Kanye West	2107
84	Robocop	Kanye West	2106
85	See You In My Nightmares	Kanye West	2092
86	Scar Tissue	Red Hot Chili Peppers	2091
87	Weird Fishes/Arpeggi	Radiohead	2088
88	Clark Gable	The Postal Service	2088
89	Exit Music (For A Film)	Radiohead	2087
90	Brand New Colony	The Postal Service	2076
91	Say It Right	Nelly Furtado	2074
92	Piece Of Me	Britney Spears	2069
93	How To Save A Life	The Fray	2064
94	Stairway To Heaven	Dread Zeppelin	2061
95	Angel	Massive Attack	2058
96	Videotape	Radiohead	2057
97	Street Lights	Kanye West	2051
98	Run	Snow Patrol	2045
99	Where Is My Mind?	Pixies	2030

In [260]:
```python
#Top Tracks
plt.figure(figsize=(12,6))
sns.barplot(x='track-name', y='listencount', data=top.head(20))
plt.title('Top Tracks by Plays')
plt.xlabel('Track Name')
plt.ylabel('Listen Count')
plt.xticks(rotation=90)
plt.tight_layout()
plt.show()


# Top Artists
plt.figure(figsize=(12,6))
sns.barplot(x='artist-name', y='listencount', data=top.head(20))
plt.title('Top Artists by Plays')
plt.xlabel('Artist Name')
plt.ylabel('Listen Count')
plt.xticks(rotation=90)
plt.tight_layout()
plt.show()
```

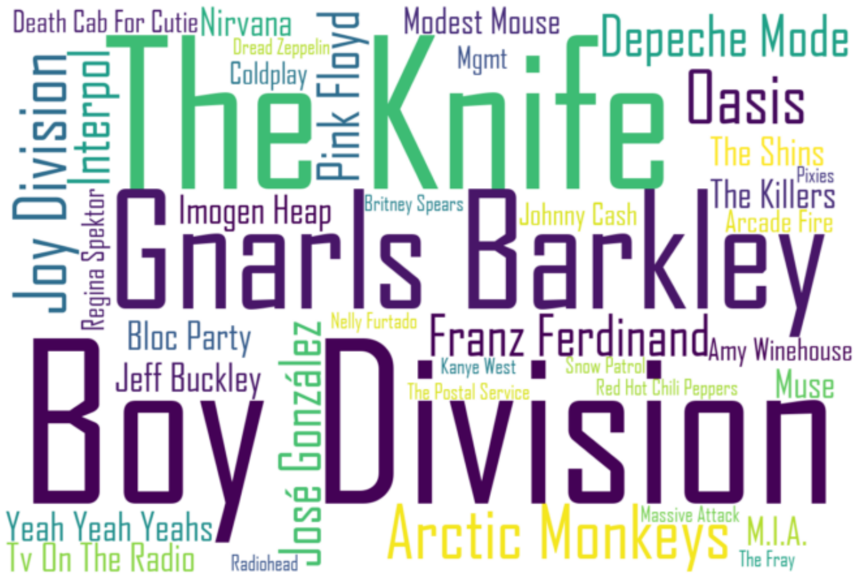




In [180]:
```python
# Create a dictionary from the top words
top_word_counts = dict(zip(top['track-name'], top['listencount']))

#Agency Font Path
agency_font_path = 'C:/WINDOWS/FONT/AGENCYR.TTF'

# Create a WordCloud object and generate the word cloud for the top 50 words
wordcloud = WordCloud(width=1200, height=800,
                      background_color='white',font_path=agency_font_path).generate_from_frequencies(top_word_counts)

# Display the word cloud using Matplotlib
plt.figure(figsize=(12, 8))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.show()
```



In [179]:
```python
import matplotlib.pyplot as plt
from wordcloud import WordCloud

# Create a dictionary from the top words
top_word_counts = dict(zip(top['artist-name'], top['listencount']))

#Agency Font Path
agency_font_path = 'C:/WINDOWS/FONT/AGENCYR.TTF'

# Create a WordCloud object and generate the word cloud for the top 50 words
wordcloud = WordCloud(width=1200, height=800,
                      background_color='white',font_path=agency_font_path).generate_from_frequencies(top_word_counts)

# Display the word cloud using Matplotlib
plt.figure(figsize=(12, 8))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.show()
```


Death Cab For Cutie
Nirvana
Pink Floyd
Modest Mouse
Depeche Mode
Dread Zeppelin
Coldplay
Mgmt
The Knife
Oasis
Joy Division
Interpol
The Shins
Pixies
The Killers
Imogen Heap
Britney Spears
Johnny Cash
Arcade Fire
Regina Spektor
Gnarls Barkley
Nelly Furtado
Bloc Party
Franz Ferdinand
Amy Winehouse
Jeff Buckley
Kanye West
Snow Patrol
José González
The Postal Service
Red Hot Chili Peppers
Muse
Boy Division
Yeah Yeah Yeahs
Tv On The Radio
Radiohead
Arctic Monkeys
Massive Attack
M.I.A.
The Fray