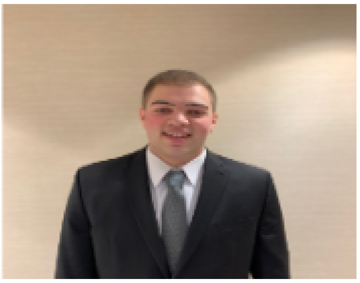
Table of Contents

1 Final Project Phase 3

1.1 Team Information

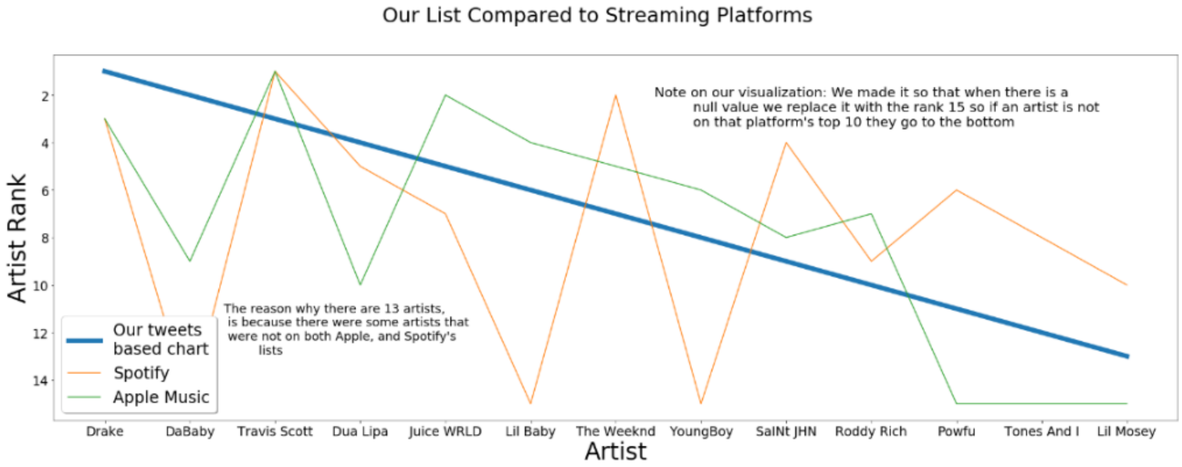
- Apple vs. Spotify - A Social Media Rivalrly Analysis
- Alex Bzdel - abzdel@bryant.edu (mailto:abzdel@bryant.edu)

- Zach Galante - zgalante@bryant.edu (mailto:zgalante@bryant.edu)
- Robert Mitchell - rmitchell2@bryant.edu (mailto:rmitchell2@bryant.edu)
- Group 2: Pied Piper



2 Abstract

We plan to explore an artist’s popularity on Spotify’s streaming platform in comparison to Apple Music's. We will first use the number of tweets about an artist to determine how many people are talking about them. Then we will pass that information into our equation (explained in detail in a later section) to get a rating for each tweet, which we can then sort into individual DataFrames for each artist. We will then make our own top 10 list based off of the data from both Apple Music and Spotify. This will then give us insights to our hypothesis to see which streaming platform twitter has a greater impact on. We will not be factoring in the sentiments of the tweets as we are simply testing whether being talked about more equates to having a higher rating on the Spotify and Apple top artists. This is due to the fact that, even if a user is tweeting negatively about an artist, the tweet will be seen by people who will stream their songs as a result of seeing the tweet (for a fun way of understanding this click here (https://youtu.be/BD5ha9mENzw)). We believe this would count the same as if someone was listening to the song because they heard something good about it. In terms of ranking tweets by importance, we will use a variety of factors. We plan to use the number of favorites and the number of followers that each follower has, in addition to the rank based on the number of tweets.

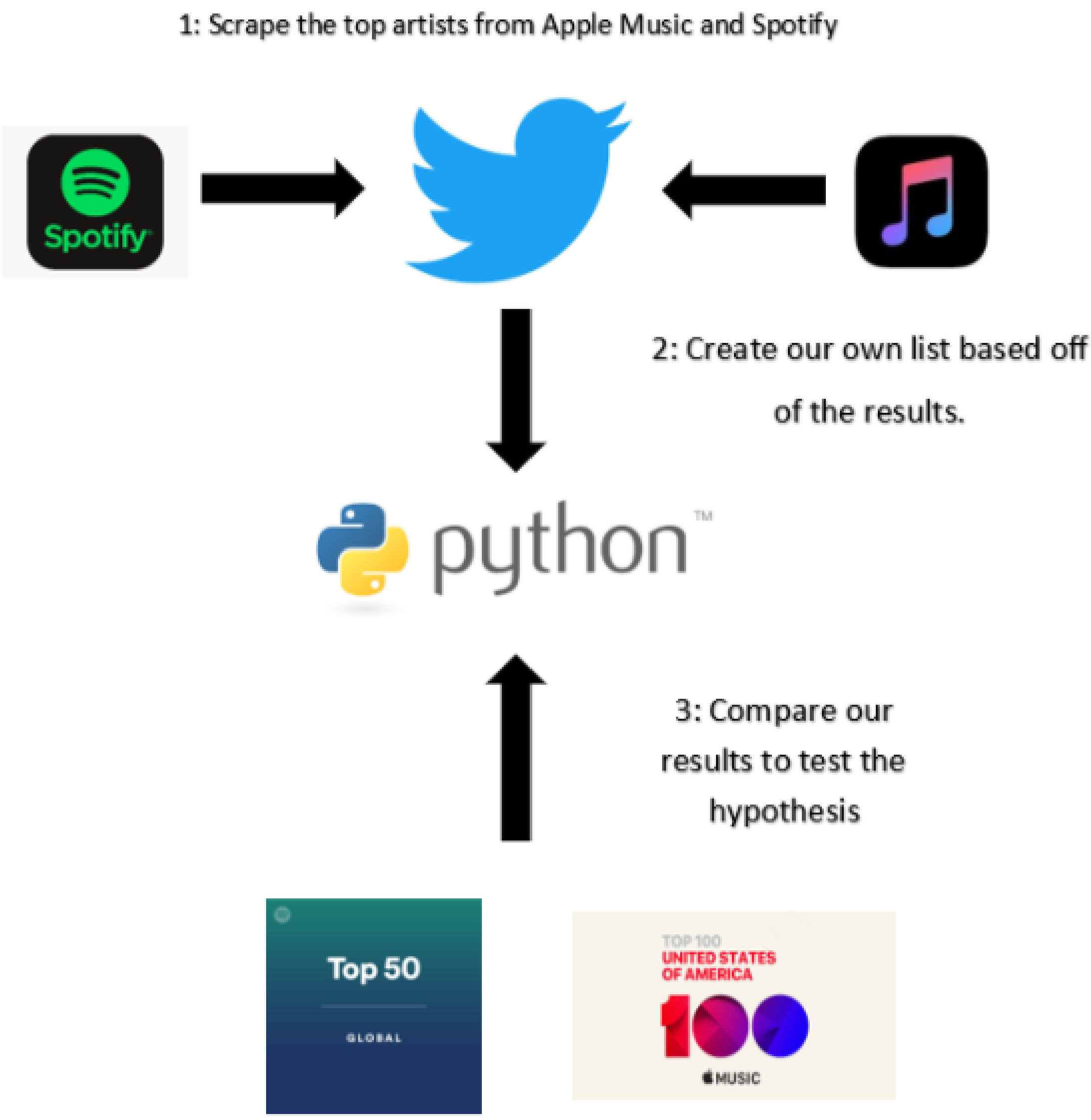


Note on our visualization: We made it so that when there is a null value we replace it with the rank 15 so if an artist is not on that platform's top 10 they go to the bottom

The reason why there are 13 artists, is because there were some artists that were not on both Apple, and Spotify's lists

2.1 Hypothesis

A higher quantity of tweets about an artist directly leads to a higher placement on the top charts. With our tweet rating metric we will ensure that, despite having a high quantity of tweets about an artist, poor tweets will be filtered out



1: Scrape the top artists from Apple Music and Spotify

2: Create our own list based off of the results.

3: Compare our results to test the hypothesis

In [27]:

```
#install twython
!pip install twython
```

```
Requirement already satisfied: twython in /usr/local/lib/python3.7/site-pac
kages (3.8.2)
Requirement already satisfied: requests-oauthlib>=0.4.0 in /usr/local/lib/p
ython3.7/site-packages (from twython) (1.3.0)
Requirement already satisfied: requests>=2.1.0 in /usr/local/lib/python3.7/
site-packages (from twython) (2.22.0)
Requirement already satisfied: oauthlib>=3.0.0 in /usr/local/lib/python3.7/
site-packages (from requests-oauthlib>=0.4.0->twython) (3.1.0)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /
usr/local/lib/python3.7/site-packages (from requests>=2.1.0->twython) (1.2
4.2)
Requirement already satisfied: chardet<3.1.0,>=3.0.2 in /usr/local/lib/pyth
on3.7/site-packages (from requests>=2.1.0->twython) (3.0.4)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python
3.7/site-packages (from requests>=2.1.0->twython) (2019.9.11)
Requirement already satisfied: idna<2.9,>=2.5 in /usr/local/lib/python3.7/s
ite-packages (from requests>=2.1.0->twython) (2.8)
```

3 Description of the data

3.1 Scraper(s) used

In [28]:

```
from twython import TwythonStreamer
import csv
import json
import codecs
import time
from random import seed
from random import randint
# seed random number generator
seed(1)

tweets_filename = "twitter_output_Spotify10"

consumer_key = "ND4wRUZRSgRS1NBtb9vRbm96v"
consumer_secret = "z0zxb6QObq4AUU2bA8BBclnIZbDr89vShniSZ9NjMWHUaIJ2AK"
access_token = "1250793114274598913-1mA5LzmdeGsu7PvTD1QpoZymfUPrPl"
access_token_secret = "cqNiCEcr5OsbS5G1t3uAGhZhqabgeu7XVHZluQiJIBVnf"

tweetabbr = []

# In this session we are using the Twitter IDs to gather tweets from those ac
AllPDs = ['561106229', '34296669', '974277346252423169', '121222566', '991525
          '35871927', '304847225']
# Filter out unwanted data for the CSV file. We are saving the entire JSON to
def process_tweet(tweet):
    d = {}
    d['hashtags'] = [hashtag['text'] for hashtag in tweet['entities']['hashta
    d['id'] = tweet['id']
    d['text'] = tweet['text']
    d['name'] = tweet['user']['name']
    d['user'] = tweet['user']['screen_name']
    d['user_loc'] = tweet['user']['location']
    d['user_desc'] = tweet['user']['description']
    d['user_followers'] = tweet['user']['followers_count']
    d['user_friends'] = tweet['user']['friends_count']
    d['user_listed'] = tweet['user']['listed_count']
    d['user_created'] = tweet['user']['created_at']
    d['user_favs'] = tweet['user']['favourites_count']
    d['user_statuses'] = tweet['user']['statuses_count']

    return d


# Create a class that inherits TwythonStreamer
class MyStreamer(TwythonStreamer):

    # Received data
    def on_success(self, data):

        # Save full JSON to file
        # TODO : save properly so we can load later directly
        # A tweet JSON record per line
        with open(f'{tweets_filename}.json', 'a') as jsonfile:
            json.dump(data, jsonfile)
            jsonfile.write("\n")

        # Only save tweets in English
        if data['lang'] == 'en':
            tweet_data = process_tweet(data)
            self.save_to_csv(tweet_data)

    # Problem with the API
    def on_error(self, status_code, data):
        print(status_code, data)
        self.disconnect()

    # Save each tweet to csv file
    def save_to_csv(self, tweet):
        with open(f'{tweets_filename}.csv', 'a', encoding="utf8") as file:
            writer = csv.writer(file)
            writer.writerow(list(tweet.values()))

while True:
    try:
        # Instantiate from our streaming class
        stream = MyStreamer(consumer_key, consumer_secret,
                            access_token, access_token_secret)

        # Start the stream - this would capture tweets generated by specific 
        # There are online tools to get the account number using the @ accour
        # stream.statuses.filter(follow=17169239) #Track uses comma separatec

        # Start the stream - this would capture tweets generated by these acc
        #stream.statuses.filter(follow=AllPDs) #Track uses comma separated li

        # Start the stream - This stream looks for specific terms (mentions) 
        #stream.statuses.filter(track='@VASenate2018,@MariaCantwell,@Susan_Hu
        stream.statuses.filter(track='Travis Scott, The Weekend, Drake, SAINt
        """
        the above tracks the Spotify top 10 artists - we used a separate scra
        the scraper for Apple Music looked like this:

        stream.statuses.filter(track='Travis Scott, Juice WRLD, Drake, Lil Ba
        """

    except (KeyboardInterrupt):
        print("Exiting")
        break
    except Exception as e:
        print("error - sleeping " + str(e))
        time.sleep(randint(30, 90)) #suspends (waits) execution of the currer
        continue
```

```
Exiting
```

3.2 Data description

- We scraped the following Tweet characteristics:
  - Hashtags
  - Tweet ID
  - Tweet text
  - User's display name

    - Username
    - User's location
    - User's description
    - Followers
    - Friends
    - Lists
    - Account creation date
    - Favorites
    - User total tweets
- We set the keywords to be the top 10 artists on both Spotify and Apple. There was a little bit of overlap, but some artists were unique to just one platform. We ended up with 13 artists total.

3.3 When and how long you scraped Twitter

- We scraped Twitter on 4/24, 4/25, and 4/26 for 4-5 hours each day, we were able to scrape almost 100,000 tweets

3.4 Alternative data source(s) used

- For alternative data sources, we used the Spotify and Apple Music Daily Top Charts for 4/24/2020. However, Spotify provided us with a csv file that included number of streams on that day while Apple Music only provided us with the top ranking songs.

3.5 Load in Data

In [3]:

```
import pandas as pd
import numpy as np
# if saving scraped tweets csv file in the same folder as this notebook, dele
Spotify = pd.read_csv("data_files/twitter_output_Spotify10.csv")
Spotify.columns= ['Hashtags','ID','Tweet_Text','Name','Username','User_Locati
Apple  = pd.read_csv("data_files/twitter_output_Apple_Artists.csv")
Apple.columns= ['Hashtags','ID','Tweet_Text','Name','Username','User_Locatior
df = Spotify.append(Apple, ignore_index =True)
df
```

Out[3]:

	Hashtags	ID	Tweet_Text	
0	[]	1.254237e+18	RT @BelindaJones68: Coronavirus sure has highl...	R
1	[]	1.254237e+18	i understand everyone has an opinion, but what...	ekata lanrete
2	[]	1.254237e+18	Wow! We had some super fast times today in th...	RHS
3	[]	1.254237e+18	RT @tdiizzlleee: Two weeks man that's all we h...	
4	[]	1.254237e+18	RT @DineshDSouza: Apparently the official prop...	QAnon

3.6 Drop Duplicates

In [6]:

```
length  = len(df)
new_length = len(df.drop_duplicates())
print(f" The current length of our Dataframe is {length} records")
print(f" The length of our Dataframe after dropping duplicates is {new_lengtł
```

```
 The current length of our Dataframe is 100065 records
 The length of our Dataframe after dropping duplicates is 99870 records
```

3.6.1 Apple Music vs. Spotify Top 10 Artists

- We've created DataFrames for the top daily streamed songs on 4/24/2020 - one represents Spotify's top 10 and one represents Apple Music's
- We will be analyzing tweets about the artists associated with each song
- A more interesting analysis would include number of streams per song, but only Spotify provides this information to external users

3.6.2 Loading top 10 lists

In [30]:

```
# if saving scraped tweets csv file in the same folder as this notebook, dele

# apple's top 10 artists as of 4/25
apple_top_10 = pd.read_csv('data_files/apple_top_10.csv')

# spotify top 10 artists as of 4/25
spotify_top_10 = pd.read_csv('data_files/spotify_top_10.csv')
```

In [31]:

```
apple_top_10
```

Out[31]:

	Track	Artist	Position
0	THE SCOTTS	Travis Scott	1
1	Righteous	Juice WRLD	2
2	Toosie Slide	Drake	3
3	Rockstar	Lil Baby	4
4	Blinding Lights	The Weeknd	5
5	Diamonds	YoungBoy Never Broke Again	6
6	The Box	Roddy Rich	7
7	Roses	SaINt JHN	8
8	JUMP	DaBaby	9
9	Don't Start Now	Dua Lipa	10

In [32]:

```
spotify_top_10=spotify_top_10.replace(to_replace="Roddy Ricch",value="Roddy R
spotify_top_10=spotify_top_10.replace(to_replace="SAINt JHN",value="SaINt JH
spotify_top_10
```

Out[32]:

	Track	Artist	Position
0	THE SCOTTS	Travis Scott	1
1	Blinding Lights	The Weeknd	2
2	Toosie Slide	Drake	3
3	Roses - Imanbek Remix	SaINt JHN	4
4	Don't Start Now	Dua Lipa	5
5	death bed (coffee for your head) (feat. beabad...	Powfu	6
6	Righteous	Juice WRLD	7
7	Dance Monkey	Tones And I	8
8	The Box	Roddy Rich	9
9	Blueberry Faygo	Lil Mosey	10

Now, we'll create one combined DataFrame, showing Apple Music's Top 10 vs Spotify's

In [33]:

```
all_rankings = apple_top_10.rename(columns={'Track': 'Track: Apple', 'Artist'
all_rankings = pd.merge(all_rankings, spotify_top_10, left_index=True, right_
all_rankings = all_rankings.rename(columns={'Track': 'Track: Spotify', 'Artis
all_rankings
```

Out[33]:

	Track: Apple	Artist: Apple	Position: Apple	Track: Spotify	Artist: Spotify	Position: Spotify
0	THE SCOTTS	Travis Scott	1	THE SCOTTS	Travis Scott	1
1	Righteous	Juice WRLD	2	Blinding Lights	The Weeknd	2
2	Toosie Slide	Drake	3	Toosie Slide	Drake	3
3	Rockstar	Lil Baby	4	Roses - Imanbek Remix	SaINt JHN	4
4	Blinding Lights	The Weeknd	5	Don't Start Now	Dua Lipa	5
5	Diamonds	YoungBoy Never Broke Again	6	death bed (coffee for your head) (feat. beabad...	Powfu	6
6	The Box	Roddy Rich	7	Righteous	Juice WRLD	7
7	Roses	SaINt JHN	8	Dance Monkey	Tones And I	8
8	JUMP	DaBaby	9	The Box	Roddy Rich	9
9	Don't Start Now	Dua Lipa	10	Blueberry Faygo	Lil Mosey	10

4 Our Tweet Importance Metric

For our tweet rating metric, we'll take the **number of followers** a user has and add it to the **number of favorites** on the given tweet. Then, we'll divide this figure by the **rank metric**.

The **rank metric** is determined by ranking each user based on how many tweets they have. The more tweets a user has, the lower their rank number will be. When the numerator is divided by this smaller denominator, the final result will be a larger number - indicating that it is more important. A user with only a few tweets will have a much higher rank number and, when combined with the numerator, will produce a lower tweet rating result.

$$\text{tweet\_rating} = \frac{f + fv}{r}$$

Where:

- f = number of followers the user has
- fv = number of favorites on the tweet

- r = rank metric (ranked based on number of tweets the user has)

In [34]:

```
df['Num_Tweets_Rank'] = df['User_Total_Tweets'].rank(ascending=False, method


df.sort_values('Num_Tweets_Rank', ascending=False)
```

Out[34]:

	Hashtags	ID	Tweet_Text	Name	Username
57004	['Fortnite']	1.254510e+18	RT @FortniteGLAT: Trailer - Concierto Travis S...	Eduardo Veléz (ForniteGamePlayer)	EduardoVelzFor1
81293	[]	1.254540e+18	Today’s plan: lying in bed and looking at the ...	irjdnev	irjdnev
25783	[]	1.254120e+18	@FortniteGame Fortnite u guys should do xxxte...	Sergio	Sergio19865538
92844	[]	1.254560e+18	@WORLDSTAR @DaBabyDaBaby Just for laughs but D...	danielmcjames	chromejoyd
52304	[]	1.254510e+18	Juice Wrld Righteous Reaction https://t.co/hVS...	MOFILMZ	mofilmz
...	...	...	...	...	...
22765	[]	1.254120e+18	@the1275GT Hi Steve, Thanks for those details....	Tesco	Tesco
76372	['CORONAVIRUS']	1.254540e+18	WHILE TRUMP IS ENCOURAGING PROTESTERS TO PROTE...	Tomthunkit™	TomthunkitsMind
66381	[]	1.254520e+18	RT @betsy_klein: March 9: political fundraiser...	Betty	missb62
54812	[]	1.254510e+18	Fans Pick Juice WRLD’s ‘Righteous’ as This Wee...	Fantasy Art: The Gifts	fantasysite
54817	[]	1.254510e+18	DaBaby Arrives at No. 1 on Billboard 200 Album...	Fantasy Art: The Gifts	fantasysite

99818 rows × 14 columns

In [35]:

```
numerator = df['Followers'] + df['Favorites'] # number of followers plus numb
rank_metric = np.round(numerator/df['Num_Tweets_Rank'], 4) # divide as per ou
rank_metric.sort_values(ascending=False)
# test to see highest final tweet rankings
```

Out[35]:

```
22765    119973.2000
28194     77753.0000
92119     43876.1429
54817     36251.0000
12828     30518.0370
             ...
29558         0.0000
72625         0.0000
81227         0.0000
38803         0.0000
92075         0.0000
Length: 99818, dtype: float64
```

In [36]:

```
rank_metric.value_counts()
```

Out[36]:

```
0.0000     477
0.0001     303
0.0002     299
0.0003     222
0.0004     210
          ...
0.4263       1
32.4797      1
5.6048       1
3.0349       1
0.9875       1
Length: 32647, dtype: int64
```

There are only 475 zeroes (<1% of tweets), so we will be able to drop these later without losing too much data

In [37]:

```
new_rank_df = pd.DataFrame(rank_metric.sort_index(ascending=False)) # sort th
new_rank_df

# we will sort our main dataframe by index as well to ensure the two join cor
```

Out[37]:

	0
99817	0.6805
99816	3.1593
99815	0.7115
99814	0.3494
99813	0.2721
...	...
4	2.9660
3	0.4211
2	0.0125
1	0.4499
0	1.4715

99818 rows × 1 columns

4.1 Join tweet ranks with tweets df

In [38]:

```
new_df = pd.merge(df.sort_index(ascending=False), new_rank_df, left_index=Tru
# sort main dataframe by index, merge the two to create a new dataframe
new_df.sort_values(0, ascending=False)
```

Out[38]:

	Hashtags	ID	Tweet_Text	Name	Username	Us
22765	[]	1.254120e+18	@the1275GT Hi Steve, Thanks for those details....	Tesco	Tesco	
28194	['SaturdayVibes', 'StayHome', 'weekend', 'Kind...	1.254480e+18	RT @barbiesway: #SaturdayVibes Love @ 1st sigh...	Paul Cude	paul_cude	
92119	[]	1.254560e+18	RT @mitchellreports: .@Yamiche: "We see a Pres...	Jean M. O'Brien	Oldlady12345	F
54817	[]	1.254510e+18	DaBaby Arrives at No. 1 on Billboard 200 Album...	Fantasy Art: The Gifts	fantasysite	
12828	[]	1.254498e+18	RT @GISH: YOU ARE NOW FREE TO SHARE YOUR ITEMS...	anodyne	DiChristine	
...	...	...	...	...	...	
29558	[]	1.254480e+18	@Luminosity Dababy	Dom	Dom59200289	
72625	[]	1.254530e+18	Former Hillary Adviser Calls On Joe Biden To D...	2020Ernie	2020Ernie	
81227	['ShowUsYourStack', 'books', 'bookstagram']	1.254540e+18	#ShowUsYourStack \n\nWhat are you reading duri...	Chapter One Bookstore	UBChapteronebks	D
38803	[]	1.254490e+18	@RogersBase She is pretty big compared to X Dr...	강성민	9WBnl0qW5mcW0DI	
92075	['HenryCavill', 'ManfromUncle']	1.254560e+18	RT @CavillsClique: Happy weekend everyone. Her...	Henry cavill	Henryca91803121	

99818 rows × 15 columns

In [39]:

```
new_rank_df = pd.DataFrame(rank_metric.sort_index(ascending=False)) # sort th
# we will sort our main dataframe by index as well to ensure the two join cor

new_df = pd.merge(df.sort_index(ascending=False), new_rank_df, left_index=Tru
# sort main dataframe by index, merge the two to create a new dataframe

# index 1396 as the first result matches our initial test two cells above, so
```

In [40]:

```
new_df = new_df.rename(columns={0: 'Tweet_Rank'}) # renaming column to Tweet_
```

In [41]:

```
df = new_df # overwriting our original dataframe
df.sort_values('Tweet_Rank', ascending=False)
```

Out[41]:

	Hashtags	ID	Tweet_Text	Name	Username	Us
22765	[]	1.254120e+18	@the1275GT Hi Steve, Thanks for those details....	Tesco	Tesco	
28194	['SaturdayVibes', 'StayHome', 'weekend', 'Kind...	1.254480e+18	RT @barbiesway: #SaturdayVibes Love @ 1st sigh...	Paul Cude	paul_cude	
92119	[]	1.254560e+18	RT @mitchellreports: .@Yamiche: "We see a Pres...	Jean M. O'Brien	Oldlady12345	F
54817	[]	1.254510e+18	DaBaby Arrives at No. 1 on Billboard 200 Album...	Fantasy Art: The Gifts	fantasysite	
12828	[]	1.254498e+18	RT @GISH: YOU ARE NOW FREE TO SHARE YOUR ITEMS...	anodyne	DiChristine	
...	...	...	...	...	...	
29558	[]	1.254480e+18	@Luminosity Dababy	Dom	Dom59200289	
72625	[]	1.254530e+18	Former Hillary Adviser Calls On Joe Biden To D...	2020Ernie	2020Ernie	
81227	['ShowUsYourStack', 'books', 'bookstagram']	1.254540e+18	#ShowUsYourStack \n\nWhat are you reading duri...	Chapter One Bookstore	UBChapteronebks	: D
38803	[]	1.254490e+18	@RogersBase She is pretty big compared to X Dr...	강성민	9WBnl0qW5mcW0DI	
92075	['HenryCavill', 'ManfromUncle']	1.254560e+18	RT @CavillsClique: Happy weekend everyone. Her...	Henry cavill	Henryca91803121	

99818 rows × 15 columns

5 EDA

5.1 Data Cleaning Tools

In [42]:

```
!pip install nltk
```

```
Requirement already satisfied: nltk in /usr/local/lib/python3.7/site-packag
es (3.4.5)
Requirement already satisfied: six in /usr/local/lib/python3.7/site-package
s (from nltk) (1.12.0)
```

In [43]:

```
import nltk
nltk.download('stopwords')
nltk.download('punkt')
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

Out[43]:

```
True
```

In [44]:

```
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
import re
stop_words = set(stopwords.words('english'))


def remove_url(txt):
    """Replace URLs found in a text string with nothing
    (i.e. it will remove the URL from the string).
    """
    return re.sub("([^0-9A-Za-z \t])|(\w+:\/\/\S+)", "", txt)

def remove_stop_words(txt):
    """lower case the text, and DROP stop words. """
    return " ".join([w for w in word_tokenize(txt.lower()) if not w in stop_w

def preprocess_tweet_text(txt):
    return remove_stop_words(remove_url(txt))
```

In [45]:

```
# Remove URLs and Stop Words
df['filtered_text'] = df.Tweet_Text.apply(preprocess_tweet_text)

#retweet
df['retweet_flags'] = df.Tweet_Text.str.startswith('RT')

# TODO Add code analyse URLs, their domains, categories etc.

df.head()
```

Out[45]:

	Hashtags	ID	Tweet_Text	Name	Username	User_L
99817	[]	1.254560e+18	Turns out I've been a fool for not listening t...	✨🔮 Crystal 🔮✨	_Crystalosaurus	The Af Be
99816	[]	1.254560e+18	RT @calebturner_10: The amount of snap stories...	s o f	itssofiaosmani	
99815	[]	1.254560e+18	RT @_DashawnJ_: JADED by Drake is a timeless s...	Lex ✨🌙	justlexi95	
99814	['TousEnsemble', 'StrongerTogether']	1.254560e+18	...Drake is still talking #TousEnsemble #Stron...	Erin Pepler	erinpepler	

5.2 Analyzing Data

Here, we will create DataFrames for each artist:

- DataFrames will be formed by a regular expression that searches for the artist name and song name (song that put the artist in the top charts).
- There will be two DataFrames per artist - one that searches 'Tweet_Text' (content of the tweet) and one that searches 'Hashtags'.
- Artists in both the Apple and Spotify top 10 will both be considered and all artists will be ranked
- These DataFrames will help us create our own top 10 ranking later on

In [46]:

```
import re
# dataframe for Travis Scott tweets and hashtags
travis_tweets = df[df['Tweet_Text'].str.contains("travis scott|the scotts", f
travis_hashtags = df[df['Hashtags'].str.contains("travisscott|thescotts", fla

# dataframe for The Weeknd tweets and hashtags
weeknd_tweets = df[df['Tweet_Text'].str.contains("weeknd|blinding lights", fl
weeknd_hashtags = df[df['Hashtags'].str.contains("weeknd|blindinglights", fla

# dataframe for Drake tweets and hashtags
drake_tweets = df[df['Tweet_Text'].str.contains("drake|toosie slide", flags=r
drake_hashtags = df[df['Hashtags'].str.contains("drake|toosieslide", flags=re

# dataframe for SAINt JHN tweets and hashtags
"""
Note that "roses" could return many results not related to the song. This is
songs on this list, but this one especially could yield some results we don't
with the expectation that this may not give us a ton of extra results as most
type this whole thing out. For hashtags, we'll just use #saintjhn
"""
saintjhn_tweets = df[df['Tweet_Text'].str.contains("saint jhn|roses - imanbek
saintjhn_hashtags = df[df['Hashtags'].str.contains("saintjhn", flags=re.IGNOR

# dataframe for Dua Lipa tweets and hashtags
dua_tweets = df[df['Tweet_Text'].str.contains("dua lipa|dont start now|don't
dua_hashtags = df[df['Hashtags'].str.contains("dualipa|dontstartnow", flags=r

# dataframe for Powfu tweets and hashtags
powfu_tweets = df[df['Tweet_Text'].str.contains("powfu|death bed|coffee for y
powfu_hashtags = df[df['Hashtags'].str.contains("powfu|deathbed", flags=re.IG

# dataframe for Juice WRLD tweets and hashtags
juice_tweets = df[df['Tweet_Text'].str.contains("juice wlrd|righteous", flags
juice_hashtags = df[df['Hashtags'].str.contains("juicewrld|righteous", flags=

# dataframe for Tones and I tweets and hashtags
tones_tweets = df[df['Tweet_Text'].str.contains("tones and i|dance monkey", f
tones_hashtags = df[df['Hashtags'].str.contains("tonesandi|dancemonkey", flag

# dataframe for Roddy Rich tweets and hashtags
roddy_tweets = df[df['Tweet_Text'].str.contains("roddy rich|the box", flags=r
roddy_hashtags = df[df['Hashtags'].str.contains("roddyrich|thebox", flags=re.

# dataframe for Lil Mosey tweets and hashtags
mosey_tweets = df[df['Tweet_Text'].str.contains("lil mosey|blueberry faygo",
# mosey_hashtags = df[df['Hashtags'].str.contains("lilmosey|blueberryfaygo",
# Lil Mosey didn't have any hashtag mentions, so we will only factor in tweet

# dataframe for Lil Baby tweets and hashtags
lilbaby_tweets = df[df['Tweet_Text'].str.contains("lil baby|rockstar", flags=
lilbaby_hashtags = df[df['Hashtags'].str.contains("lilbaby|rockstar", flags=r

# dataframe for Youngboy Never Broke Again tweets and hashtags
youngboy_tweets = df[df['Tweet_Text'].str.contains("youngboy|diamonds", flags
youngboy_hashtags = df[df['Hashtags'].str.contains("youngboy|youngboyneverbro
# dataframe for DaBaby tweets and hashtags
dababy_tweets = df[df['Tweet_Text'].str.contains("dababy|da baby|jump", flags
dababy_hashtags = df[df['Hashtags'].str.contains("dababy|jump", flags=re.IGN
```

5.3 Applying our Tweet Rating Metric

- For each artist, we will find the average Tweet rating for the tweet text DataFrame and the hashtag DataFrame

In [47]:

```
travis_tweets_rank = np.mean(travis_tweets['Tweet_Rank'])
travis_hashtags_rank = np.mean(travis_hashtags['Tweet_Rank'])
travis_avg_rank = (travis_tweets_rank + travis_hashtags_rank) / 2

print(f"Travis Scott's average tweet rank is: {np.round(travis_tweets_rank,2)
print(f"Travis Scott's average hashtag rank is: {np.round(travis_hashtags_ran
print("-"*50)

weeknd_tweets_rank = np.mean(weeknd_tweets['Tweet_Rank'])
weeknd_hashtags_rank = np.mean(weeknd_hashtags['Tweet_Rank'])
weeknd_avg_rank = (weeknd_tweets_rank + weeknd_hashtags_rank) / 2

print(f"The Weeknd's average tweet rank is: {np.round(weeknd_tweets_rank,2)}"
print(f"The Weeknd's average hashtag rank is: {np.round(weeknd_hashtags_rank,
print("-"*50)

drake_tweets_rank = np.mean(drake_tweets['Tweet_Rank'])
drake_hashtags_rank = np.mean(drake_hashtags['Tweet_Rank'])
drake_avg_rank = (drake_tweets_rank + drake_hashtags_rank) / 2

print(f"Drake's average tweet rank is: {np.round(drake_tweets_rank,2)}")
print(f"Drake's average hashtag rank is: {np.round(drake_hashtags_rank,2)}")
print("-"*50)

saintjhn_tweets_rank = np.mean(saintjhn_tweets['Tweet_Rank'])
saintjhn_hashtags_rank = np.mean(saintjhn_hashtags['Tweet_Rank'])
saintjhn_avg_rank = (saintjhn_tweets_rank + saintjhn_hashtags_rank) / 2

print(f"SaINt JHN's average tweet rank is: {np.round(saintjhn_tweets_rank,2)}
print(f"SaINt JHN's average hashtag rank is: {np.round(saintjhn_hashtags_rank
print("-"*50)

dua_tweets_rank = np.mean(dua_tweets['Tweet_Rank'])
dua_hashtags_rank = np.mean(dua_hashtags['Tweet_Rank'])
dua_avg_rank = (dua_tweets_rank + dua_hashtags_rank) / 2

print(f"Dua Lipa's average tweet rank is: {np.round(dua_tweets_rank,2)}")
print(f"Dua Lipa's average hashtag rank is: {np.round(dua_hashtags_rank,2)}"
print("-"*50)

powfu_tweets_rank = np.mean(powfu_tweets['Tweet_Rank'])
powfu_hashtags_rank = np.mean(powfu_hashtags['Tweet_Rank'])
powfu_avg_rank = (powfu_tweets_rank + powfu_hashtags_rank) / 2

print(f"Powfu's average tweet rank is: {np.round(powfu_tweets_rank,2)}")
print(f"Powfu's average hashtag rank is: {np.round(powfu_hashtags_rank,2)}")
print("-"*50)

juice_tweets_rank = np.mean(juice_tweets['Tweet_Rank'])
juice_hashtags_rank = np.mean(juice_hashtags['Tweet_Rank'])
juice_avg_rank = (juice_tweets_rank + juice_hashtags_rank) / 2

print(f"Juice WRLD's average tweet rank is: {np.round(juice_tweets_rank,2)}"
print(f"Juice WLRD's average hashtag rank is: {np.round(juice_hashtags_rank,2
print("-"*50)
```

```
tones_tweets_rank = np.mean(tones_tweets['Tweet_Rank'])
tones_hashtags_rank = np.mean(tones_hashtags['Tweet_Rank'])
tones_avg_rank = (tones_tweets_rank + tones_hashtags_rank) / 2

print(f"Tones and I's average tweet rank is: {np.round(tones_tweets_rank,2)}"
print(f"Tones and I's average hashtag rank is: {np.round(tones_hashtags_rank,
print("-"*50)

roddy_tweets_rank = np.mean(roddy_tweets['Tweet_Rank'])
roddy_hashtags_rank = np.mean(roddy_hashtags['Tweet_Rank'])
roddy_avg_rank = (roddy_tweets_rank + roddy_hashtags_rank) / 2

print(f"Roddy Rich's average tweet rank is: {np.round(roddy_tweets_rank,2)}")
print(f"Roddy Rich's average hashtag rank is: {np.round(roddy_hashtags_rank,2
print("-"*50)

mosey_tweets_rank = np.mean(mosey_tweets['Tweet_Rank'])
print(f"Lil Mosey's average tweet rank is: {np.round(mosey_tweets_rank,2)}")

print("-"*50)

lilbaby_tweets_rank = np.mean(lilbaby_tweets['Tweet_Rank'])
lilbaby_hashtags_rank = np.mean(lilbaby_hashtags['Tweet_Rank'])
lilbaby_avg_rank = (lilbaby_tweets_rank + lilbaby_hashtags_rank) / 2

print(f"Lil Baby's average tweet rank is: {np.round(lilbaby_tweets_rank,2)}")
print(f"Lil Baby's average hashtag rank is: {np.round(lilbaby_hashtags_rank,2
print("-"*50)

youngboy_tweets_rank = np.mean(youngboy_tweets['Tweet_Rank'])
youngboy_hashtags_rank = np.mean(youngboy_hashtags['Tweet_Rank'])
youngboy_avg_rank = (youngboy_tweets_rank + youngboy_hashtags_rank) / 2

print(f"Youngboy's average tweet rank is: {np.round(youngboy_tweets_rank,2)}"
print(f"Youngboy's average hashtag rank is: {np.round(youngboy_hashtags_rank,
print("-"*50)

dababy_tweets_rank = np.mean(dababy_tweets['Tweet_Rank'])
dababy_hashtags_rank = np.mean(dababy_tweets['Tweet_Rank'])
dababy_avg_rank = (dababy_tweets_rank + dababy_hashtags_rank) / 2

print(f"DaBaby's average tweet rank is: {np.round(dababy_tweets_rank,2)}")
print(f"DaBaby's average hashtag rank is: {np.round(dababy_hashtags_rank,2)}"
print("-"*50)
```

```
Travis Scott's average tweet rank is: 6.48
Travis Scott's average hashtag rank is: 1.48
--------------------------------------------------
The Weeknd's average tweet rank is: 6.33
The Weeknd's average hashtag rank is: 13.28
--------------------------------------------------
Drake's average tweet rank is: 8.76
Drake's average hashtag rank is: 4.78
--------------------------------------------------
SaINt JHN's average tweet rank is: 3.57
SaINt JHN's average hashtag rank is: 15.92
--------------------------------------------------
Dua Lipa's average tweet rank is: 18.63
Dua Lipa's average hashtag rank is: 5.32
--------------------------------------------------
Powfu's average tweet rank is: 26.0
Powfu's average hashtag rank is: 27.04
--------------------------------------------------
Juice WRLD's average tweet rank is: 92.45
Juice WLRD's average hashtag rank is: 1.99
--------------------------------------------------
Tones and I's average tweet rank is: 29.83
Tones and I's average hashtag rank is: 6.92
--------------------------------------------------
Roddy Rich's average tweet rank is: 7.83
Roddy Rich's average hashtag rank is: 1.56
--------------------------------------------------
Lil Mosey's average tweet rank is: 0.69
--------------------------------------------------
Lil Baby's average tweet rank is: 4.93
Lil Baby's average hashtag rank is: 0.99
--------------------------------------------------
Youngboy's average tweet rank is: 2.08
Youngboy's average hashtag rank is: 0.37
--------------------------------------------------
DaBaby's average tweet rank is: 28.82
DaBaby's average hashtag rank is: 28.82
--------------------------------------------------
```

5.4 Getting an Artist's Final Ranking

- To determine where an artist should fall on our ranking, we will multiply the following:
    - Number of tweets about the artist or song plus number of hashtags about the artist or song
    - 1/2 of the average of tweet and hashtag ratings

In [48]:

```
# totals - hashtag dataframes plus tweet dataframes
travis_total = len(travis_tweets)+len(travis_hashtags)
weeknd_total = len(weeknd_tweets)+len(weeknd_hashtags)
drake_total = len(drake_tweets)+len(drake_hashtags)
saintjhn_total = len(saintjhn_tweets)+len(saintjhn_hashtags)
dua_total = len(dua_tweets)+len(dua_hashtags)
powfu_total = len(powfu_tweets)+len(powfu_hashtags)
juice_total = len(juice_tweets)+len(juice_hashtags)
tones_total = len(tones_tweets)+len(tones_hashtags)
roddy_total = len(roddy_tweets)+len(roddy_hashtags)
mosey_total = len(mosey_tweets) # hashtags not included because he didn't hav
lilbaby_total = len(lilbaby_tweets)+len(lilbaby_hashtags)
youngboy_total = len(youngboy_tweets)+len(youngboy_hashtags)
dababy_total = len(dababy_tweets)+len(dababy_hashtags)
```

In [49]:

```
# final rankings - totals multiplied by 1/2 of the average tweet and hashtag
travis_final = travis_total*(.5*travis_avg_rank)
print(f"Travis Scott's final rating is {np.round(travis_final,2)}")

weeknd_final = weeknd_total*(.5*weeknd_avg_rank)
print(f"The Weeknd's final rating is {np.round(weeknd_final,2)}")

drake_final = drake_total*(.5*drake_avg_rank)
print(f"Drake's final rating is {np.round(drake_final,2)}")

saintjhn_final = saintjhn_total*(.5*saintjhn_avg_rank)
print(f"SaINt JHN's final rating is {np.round(saintjhn_final,2)}")

dua_final = dua_total*(.5*dua_avg_rank)
print(f"Dua Lipa's final rating is {np.round(dua_final,2)}")

powfu_final = powfu_total*(.5*powfu_avg_rank)
print(f"Powfu's final rating is {np.round(powfu_final,2)}")

juice_final = juice_total*(.5*juice_avg_rank)
print(f"Juice WRLD's final rating is {np.round(juice_final,2)}")

tones_final = tones_total*(.5*tones_avg_rank)
print(f"Tones and I's final rating is {np.round(tones_final,2)}")

roddy_final = roddy_total*(.5*roddy_avg_rank)
print(f"Roddy Rich's final rating is {np.round(roddy_final,2)}")

mosey_final = mosey_total*(.5*mosey_tweets_rank)
print(f"Lil Mosey's final rating is {np.round(mosey_final,2)}")

lilbaby_final = lilbaby_total*(.5*lilbaby_avg_rank)
print(f"Lil Baby's final rating is {np.round(lilbaby_final,2)}")

youngboy_final = youngboy_total*(.5*youngboy_avg_rank)
print(f"Youngboy's final rating is {np.round(youngboy_final,2)}")

dababy_final = dababy_total*(.5*dababy_avg_rank)
print(f"DaBaby's final rating is {np.round(dababy_final,2)}")
```

```
Travis Scott's final rating is 18057.49
The Weeknd's final rating is 1505.36
Drake's final rating is 50832.67
SaINt JHN's final rating is 711.22
Dua Lipa's final rating is 10561.02
Powfu's final rating is 344.81
Juice WRLD's final rating is 5831.17
Tones and I's final rating is 174.57
Roddy Rich's final rating is 668.92
Lil Mosey's final rating is 9.37
Lil Baby's final rating is 3346.18
Youngboy's final rating is 1453.8
DaBaby's final rating is 41916.41
```

6 Apple Versus Spotify: a contrastive study

6.1 One Big Artist DataFrame

contains:

- **Artist**
- **Total_Tweets**
  - total tweets about the artist or song
- **Total_Hashtags**
  - total tweets that have hashtags about the artist or song
- **Total_Tweet_Rating**
  - avg rating of total tweets about the artist or song
- **Total_Hashtags_Rating**
  - avg rating of total hashtags about the artist or song
- **Final_Rating**
  - factors in aforementioned equation to get a final rating

In [50]:

```
final_ranking = {'Artist':  ['Travis Scott','The Weeknd','Drake','SaINt JHN',
                'Tones And I','Roddy Rich','Lil Mosey','Lil Baby','YoungBoy N
        'Final_Rating': [travis_final,weeknd_final,drake_final,saintjhn_final
                        tones_final,roddy_final,mosey_final,lilbaby_final,you
        }

final_ranking = pd.DataFrame (final_ranking, columns = ['Artist','Final_Ratin

###########################################################################
# dataframe for us vs apple and us vs spotify
us_vs_apple = pd.merge(final_ranking,apple_top_10,on='Artist',how='inner')
us_vs_apple['Our_Position'] = us_vs_apple['Final_Rating'].rank(ascending=Fals

us_vs_spotify = pd.merge(final_ranking,spotify_top_10,on='Artist',how='inner'
us_vs_spotify['Our_Position'] = us_vs_spotify['Final_Rating'].rank(ascending=

final_ranking['Our_Rank'] = final_ranking['Final_Rating'].rank(ascending=Fals

###########################################################################

#final_ranking = pd.merge(final_ranking, all_rankings, left_index=True, right
#all_rankings = all_rankings.rename(columns={'Track': 'Track: Spotify', 'Arti
combined_df = pd.merge(spotify_top_10, apple_top_10, on='Artist', how='outer'
combined_df
final_ranking = pd.merge(final_ranking,combined_df,on='Artist',how='outer')
final_ranking = final_ranking.drop(columns={"Track_x", "Track_y"})
final_ranking = final_ranking.rename(columns={"Position_x": "Spotify_Positior
                                              "Position_y": "Apple_Position",
                                             "Our_Rank": "Our_Position"})

final_ranking = final_ranking.replace(to_replace="YoungBoy Never Broke Again'
us_vs_apple = us_vs_apple.replace(to_replace="YoungBoy Never Broke Again",val
us_vs_spotify = us_vs_spotify.replace(to_replace="YoungBoy Never Broke Again'



final_ranking.sort_values('Our_Position')
```

Out[50]:

	Artist	Final_Rating	Our_Position	Spotify_Position	Apple_Position
2	Drake	50832.673203	1.0	3.0	3.0
12	DaBaby	41916.413534	2.0	NaN	9.0
0	Travis Scott	18057.494768	3.0	1.0	1.0
4	Dua Lipa	10561.018171	4.0	5.0	10.0
6	Juice WRLD	5831.169321	5.0	7.0	2.0
10	Lil Baby	3346.176661	6.0	NaN	4.0
1	The Weeknd	1505.364304	7.0	2.0	5.0
11	YoungBoy	1453.799607	8.0	NaN	6.0
3	SaINt JHN	711.224786	9.0	4.0	8.0
8	Roddy Rich	668.920771	10.0	9.0	7.0

	Artist	Final_Rating	Our_Position	Spotify_Position	Apple_Position
5	Powfu	344.808804	11.0	6.0	NaN
7	Tones And I	174.565799	12.0	8.0	NaN
9	Lil Mosey	9.365050	13.0	10.0	NaN

In [51]:

```
artist_df = {'Artist':  ['Travis Scott','The Weeknd','Drake','SaINt JHN','Dua
                'Tones and I','Roddy Rich','Lil Mosey','Lil Baby','YoungBoy',
        'Total_Tweets': [len(travis_tweets),len(weeknd_tweets),len(drake_twee
             len(powfu_tweets),len(juice_tweets),len(tones_tweets),len(roddy_
             len(lilbaby_tweets),len(youngboy_tweets),len(dababy_tweets)],
        'Total_Hashtags': [len(travis_hashtags),len(weeknd_hashtags),len(drak
             len(powfu_hashtags),len(juice_hashtags),len(tones_hashtags),len(
             len(lilbaby_hashtags),len(youngboy_hashtags),len(dababy_hashtags
        'Total_Tweet_Rating': [travis_tweets_rank,weeknd_tweets_rank,drake_tw
                              powfu_tweets_rank,juice_tweets_rank,tones_tweet
                              lilbaby_tweets_rank,youngboy_tweets_rank,dababy
        'Total_Hashtags_Rating': [travis_hashtags_rank,weeknd_hashtags_rank,d
                                 dua_hashtags_rank,powfu_hashtags_rank,juice_
                                 roddy_hashtags_rank,mosey_tweets_rank,lilbab
                                 youngboy_hashtags_rank,dababy_hashtags_rank]
        'Final_Rating': [travis_final,weeknd_final,drake_final,saintjhn_final
                        tones_final,roddy_final,mosey_final,lilbaby_final,you
        }

artist_df = pd.DataFrame (artist_df, columns = ['Artist','Total_Tweets','Tota
                                                'Total_Hashtags_Rating','Fina
```

In [52]:

```
artist_df['Total_Tweets_Plus_Hashtags'] = artist_df['Total_Tweets']+artist_df
```

In [53]:

```
artist_df # final dataframe!
```

Out[53]:

	Artist	Total_Tweets	Total_Hashtags	Total_Tweet_Rating	Total_Hashtags_Rating	Final_
0	Travis Scott	8878	196	6.478274	1.481829	18057
1	The Weeknd	295	12	6.332809	13.281058	1505
2	Drake	14738	279	8.756969	4.783066	50832
3	SaINt JHN	136	10	3.569071	15.916540	711
4	Dua Lipa	1729	35	18.632370	5.315517	10561
5	Powfu	24	2	26.004508	27.043000	344
6	Juice WRLD	198	49	92.446000	1.985892	5831
7	Tones and I	18	1	29.826694	6.924000	174
8	Roddy Rich	282	3	7.830628	1.557733	668
9	Lil Mosey	27	27	0.693707	0.693707	9
10	Lil Baby	2217	41	4.933226	0.994456	3346
11	YoungBoy	2343	31	2.082491	0.367045	1453
12	DaBaby	2844	65	28.818435	28.818435	41916

So now we have two final DataFrames to work with - final_ranking and artist_df. We'll mainly be using final_ranking for visualizations as the data is more useful for our purposes.

6.2 Comparison using MSE

In [54]:
```
predicted = final_ranking["Our_Position"]
Apple_Actual = final_ranking["Apple_Position"]
Spotify_Actual = final_ranking["Spotify_Position"]
rr = predicted - Apple_Actual
sp = predicted - Spotify_Actual
sp.fillna(0, inplace = True)
rr.fillna(0, inplace = True)
rr.to_numpy
sp.to_numpy
MSE1 = np.power(rr,2)
MSE = np.power(sp,2)
size1 = sp.size
size = rr.size
MSE2 = np.sum(MSE1)/size
MSE3 = np.sum(MSE)/size1
print(f"The MSE comparing our ranking to Apple Music is {MSE2:.2f}")
print(f"The MSE comparing our ranking to Spotify is {MSE3:.2F}")
```

```
The MSE comparing our ranking to Apple Music is 9.54
The MSE comparing our ranking to Spotify is 8.77
```

7 Visual EDA

In [55]:
```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import seaborn as sns #Statistical Data Visualization
import matplotlib.pyplot as plt
%matplotlib inline
```

In [56]:
```
# add "rank" columns for visualization purposes.  Here, we are just reversing
# lower positions should have a higher rank in the barplot.
spotify_top_10["Rank"] = (10,9,8,7,6,5,4,3,2,1)
apple_top_10["Rank"] = (10,9,8,7,6,5,4,3,2,1)
```

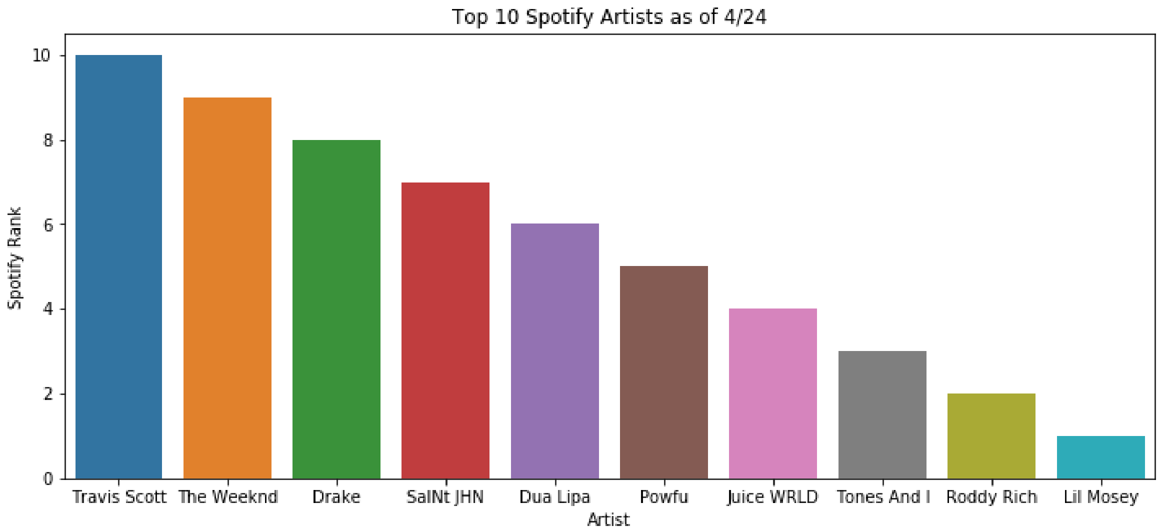
7.1 Top 10 Spotify artists

In [57]:

```
plt.figure(figsize=(12, 5)) # set size of barplot
ax = sns.barplot(x='Artist', y='Rank', data=spotify_top_10) # barplot w/ arti
plt.title('Top 10 Spotify Artists as of 4/24')
plt.xlabel('Artist')
plt.ylabel('Spotify Rank')
```

Out[57]:

```
Text(0, 0.5, 'Spotify Rank')
```



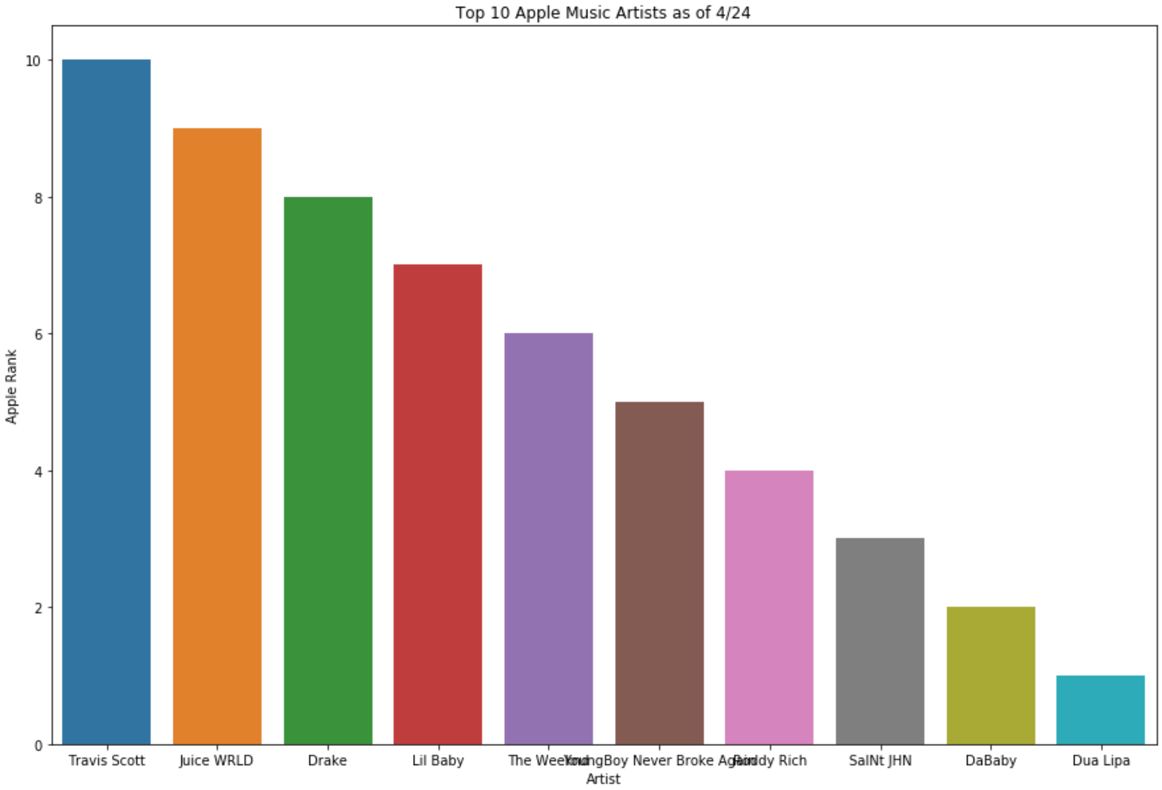
7.2 Top 10 Apple Music Artists

In [58]:

```
plt.figure(figsize=(15, 10)) # set size of barplot
ax = sns.barplot(x='Artist', y='Rank', data=apple_top_10) # barplot w/ artist
plt.title('Top 10 Apple Music Artists as of 4/24')
plt.xlabel('Artist')
plt.ylabel('Apple Rank')
```

Out[58]:

```
Text(0, 0.5, 'Apple Rank')
```



We will use position_swap in conjunction with apply to make two new series - Our_Rank and Apple_Rank - both of which are the inverse of their respective positions

Now, we'll create a new column called "rank" which is essentially just the inverse of the artist's top 10 rating. This will allow us to make a barplot where the most popular artists have the biggest bars.

To do this, let's make a function called position_swap which will swap the positions of the values 1-10

In [59]:

```
def position_swap(x):
    """function to swap values 1-10"""
    while True:
        if x==1:
            x=10
            break
        elif x==2:
            x=9
            break
        elif x==3:
            x=8
            break
        elif x==4:
            x=7
            break
        elif x==5:
            x=6
            break
        elif x==6:
            x=5
            break
        elif x==7:
            x=4
            break
        elif x==8:
            x=3
            break
        elif x==9:
            x=2
            break
        elif x==10:
            x=1
            break
    return x
```

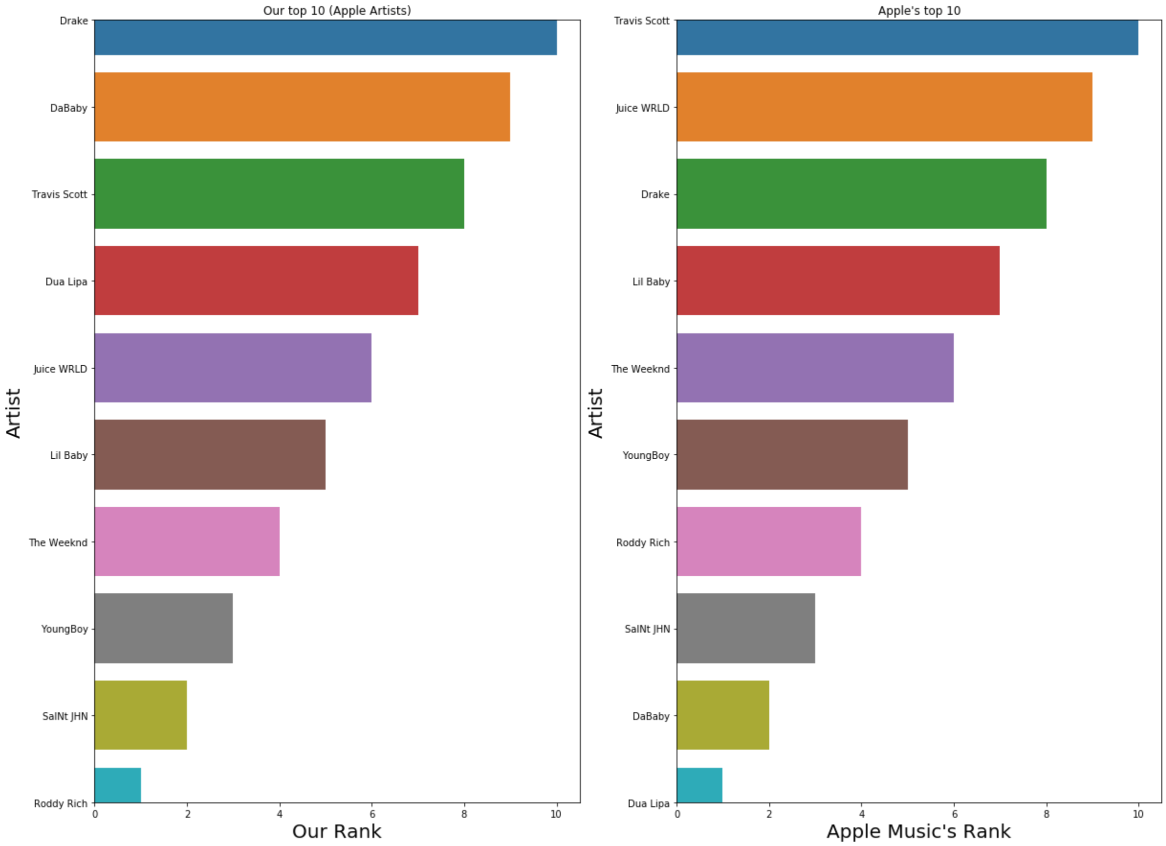
In [60]:

```
us_vs_apple['Our_Rank'] = us_vs_apple['Our_Position'].apply(position_swap) #
us_vs_apple['Rank'] = us_vs_apple['Position'].apply(position_swap) # apple's
```

7.3 Our Top 10 vs Apple Music's Top 10

In [61]:

```
f,ax=plt.subplots(1,2,figsize=(20,15))
sns.barplot('Our_Rank','Artist',data=us_vs_apple.sort_values('Our_Position'),
ax[0].set_title('Our top 10 (Apple Artists)')
ax[0].set_yticks(range(0,10))
ax[0].set_ylabel("Artist",fontsize = 20)
ax[0].set_xlabel("Our Rank", fontsize = 20)
plt.gca().invert_xaxis()
#ax[0].barh('Our_Rank', 10)


### Complete code here to explore Sex and Age vs Survived
sns.barplot('Rank','Artist',data=us_vs_apple.sort_values('Position'),orient='
ax[1].set_title("Apple's top 10")
ax[1].set_yticks(range(0,10))
ax[1].set_ylabel("Artist", fontsize =20)
ax[1].set_xlabel("Apple Music's Rank",fontsize = 20)
plt.gca().invert_xaxis()
plt.show()
```



7.4 Our Top 10 vs Spotify's Top 10

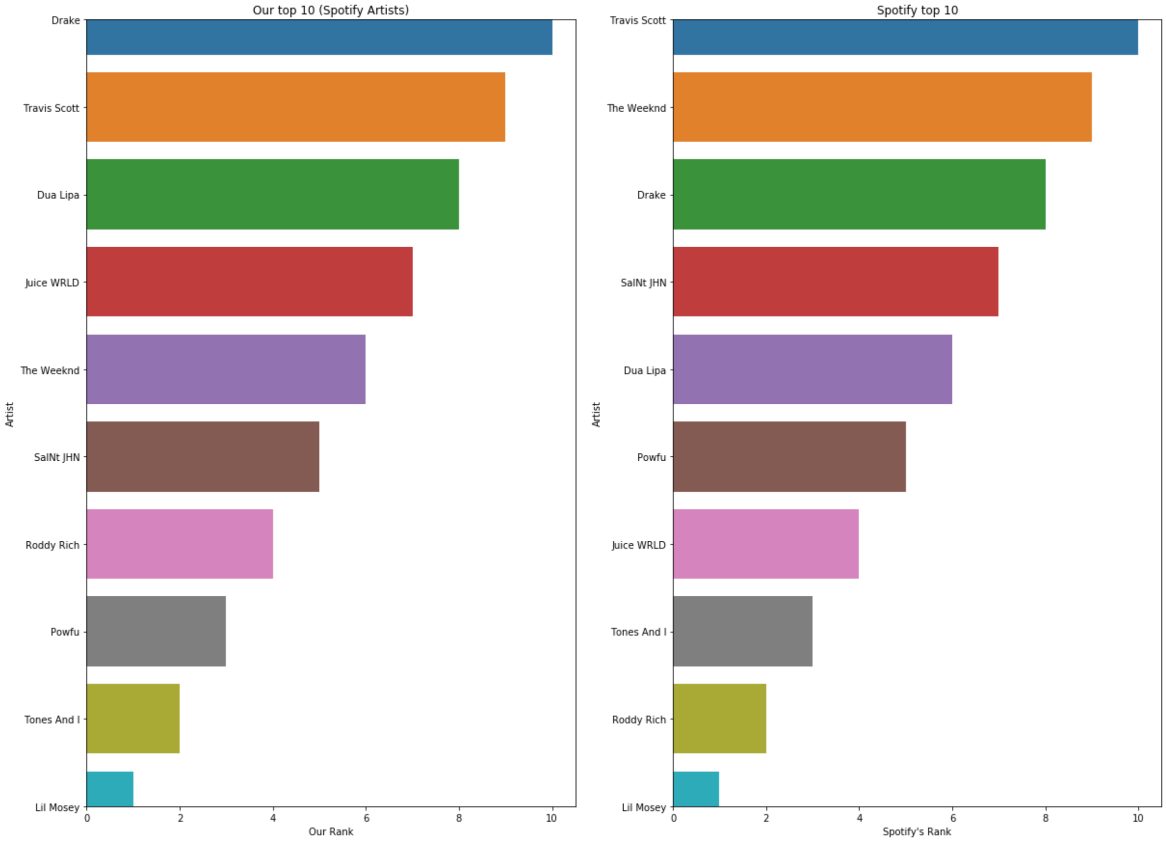
Let's do the same for Spotify

In [62]:

```
us_vs_spotify['Our_Rank'] = us_vs_spotify['Our_Position'].apply(position_swap
us_vs_spotify['Rank'] = us_vs_spotify['Position'].apply(position_swap)
```

In [63]:

```
f,ax=plt.subplots(1,2,figsize=(20,15))
sns.barplot('Our_Rank','Artist',data=us_vs_spotify.sort_values('Our_Position'
ax[0].set_title('Our top 10 (Spotify Artists)')
ax[0].set_yticks(range(0,10))
ax[0].set_ylabel("Artist")
ax[0].set_xlabel("Our Rank")
#plt.gca().invert_xaxis()
#ax[0].barh('Our_Rank', 10)


### Complete code here to explore Sex and Age vs Survived
sns.barplot('Rank','Artist',data=us_vs_spotify.sort_values('Position'),orient
ax[1].set_title('Spotify top 10')
ax[1].set_yticks(range(0,10))
ax[1].set_ylabel("Artist")
ax[1].set_xlabel("Spotify's Rank")
#plt.gca().invert_xaxis()
plt.show()
```



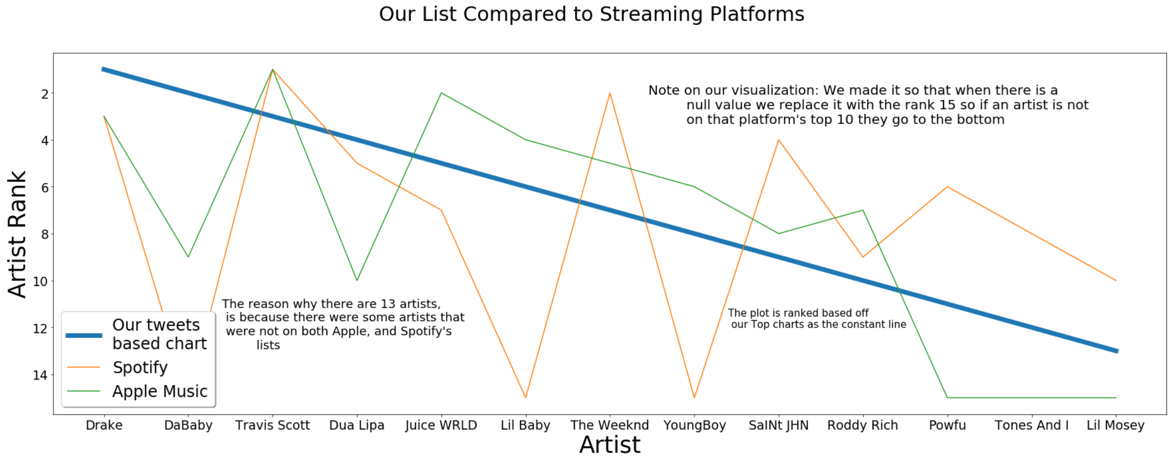
7.5 Final Line Plot

In [132]:

```
final_ranking = final_ranking.sort_values('Our_Position')
final_ranking.replace(to_replace = "YoungBoy Never Broke Again", value = "You
#Spotify12 = final_ranking["Spotify_Position"]
rank = np.arange(0,11)
Singers = final_ranking["Artist"]
Singers.replace(to_replace = "YoungBoy Never Broke Again", value = "Young Boy
Our_rank = final_ranking["Our_Position"]
Spotify = final_ranking["Spotify_Position"]
Apple = final_ranking["Apple_Position"]
Apple = Apple.fillna(15)
Spotify = Spotify.fillna(15)
plt.figure(figsize=[30,10])
plt.xticks(fontsize = 19)
plt.yticks(fontsize = 19)
plt.plot(Singers, Our_rank, label = "Our tweets\nbased chart", linewidth = 7)
plt.plot(Singers, Spotify, label = "Spotify")
plt.plot(Singers, Apple, label = "Apple Music")
plt.legend(loc = "lower left", shadow = True, prop = {"size": 24})
plt.xlabel("Artist", fontsize = 35)
plt.ylabel("Artist Rank", fontsize = 35)
plt.suptitle("Our List Compared to Streaming Platforms", fontsize = 30)
plt.gca().invert_yaxis()
plt.text(6.45,3.3,"""Note on our visualization: We made it so that when there
         null value we replace it with the rank 15 so if an artist is not
         on that platform's top 10 they go to the bottom""", fontsize = 20)
plt.text(1.4,12.9, """The reason why there are 13 artists,\n is because there
         lists""", fontsize = 18)
plt.text(7.4,12,"The plot is ranked based off\n our Top charts as the constar
```

Out[132]:

```
Text(7.4, 12, 'The plot is ranked based off\n our Top charts as the constan
t line')
```



7.6 Wordcloud Based Off Of Tweet Location

In [65]:

```
# install wordcloud library
!pip install wordcloud
```

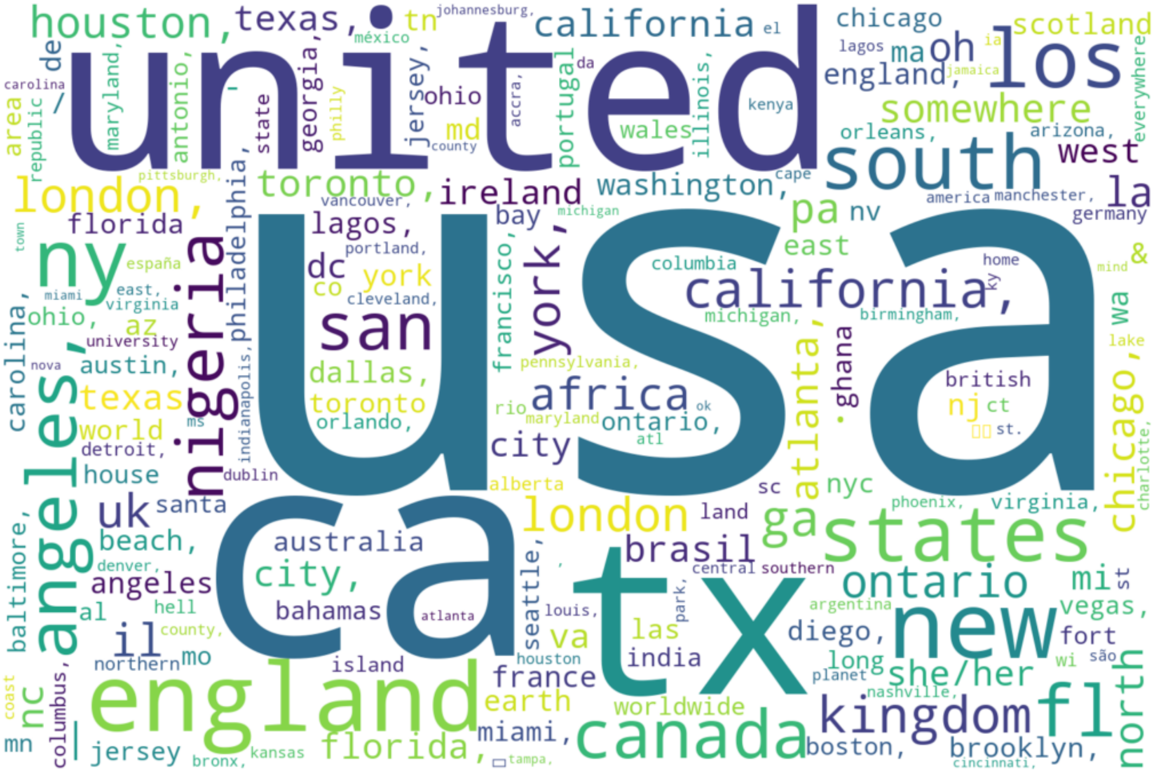
```
Requirement already satisfied: wordcloud in /usr/local/lib/python3.7/site-p
ackages (1.6.0)
Requirement already satisfied: numpy>=1.6.1 in /usr/local/lib/python3.7/sit
e-packages (from wordcloud) (1.17.2)
Requirement already satisfied: pillow in /usr/local/lib/python3.7/site-pack
ages (from wordcloud) (6.1.0)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.7/site-
packages (from wordcloud) (3.1.1)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.7/sit
e-packages (from matplotlib->wordcloud) (0.10.0)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.
7/site-packages (from matplotlib->wordcloud) (1.1.0)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in
/usr/local/lib/python3.7/site-packages (from matplotlib->wordcloud) (2.4.2)
Requirement already satisfied: python-dateutil>=2.1 in /usr/local/lib/pytho
n3.7/site-packages (from matplotlib->wordcloud) (2.8.0)
Requirement already satisfied: six in /usr/local/lib/python3.7/site-package
s (from cycler>=0.10->matplotlib->wordcloud) (1.12.0)
Requirement already satisfied: setuptools in /usr/local/lib/python3.7/site-
packages (from kiwisolver>=1.0.1->matplotlib->wordcloud) (41.2.0)
```

In [66]:

```
df["User_Location"].dropna(inplace = True)
from wordcloud import WordCloud, STOPWORDS

word_counts = {} # Initialize a Dictionary
for text in  df.User_Location:  #or the cleaned up text
    text = str(text).lower() # Convert to string
    for word in text.split(): # text.split()
        word_counts[word] = word_counts.get(word, 0) + 1


stopwords = set(STOPWORDS)
delete_present_stopwords=[key for key in word_counts if key in stopwords]
for key in delete_present_stopwords:
    del word_counts[key]
```

In [67]:

```
import matplotlib
import matplotlib.pyplot as plt
import seaborn as sns

%matplotlib inline
wordcloud = WordCloud(stopwords=stopwords, width = 1500, height = 1000,
                background_color ='white',
                min_font_size = 10).generate_from_frequencies(word_counts)

plt.figure(figsize = (14, 14), facecolor = None)
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.tight_layout(pad = 0)
plt.show()
```



8 Discussion

Throughout the process of this project, there were several aspects we had to think about before moving forward with our analysis. One major point of consideration for the group was to determine the key words to scrape for. At first, we scraped for an artist name, such as “Drake” followed by the streaming platform, “Apple Music”. When we analyzed these results, we saw that we didn’t have nearly enough data for an appropriate analysis. We then decided to change our keywords on our scapers to get the top 10 artists from Spotify and Apple Music’s top 10 charts, respectively. We also ran into an issue with the artist Lil Mosey - he did not have any hashtags during the time of our scraping. When we made our final equation to place our artists, we took the average rating of each artist’s total tweets and average rating of each artist’s hashtags. For Lil Mosey we simply used the average rating of his total tweets in our equation.

9 Conclusion

Given our results, we conclude that Spotify's chart is more highly affected by the number of relatively high-quality tweets there are about an artist. Our ranking had an MSE of 8.77 compared to Spotify, while the MSE for Apple Music was 9.54. One major driving factor we thought would cause a disparity is the fact that our ranking method placed the three artists that were unique to Spotify’s charts as ranks 11, 12, and 13 out of a total of 13 artists. In addition, we can also see from the word cloud that most of our tweets come from users located in the United States, and more specifically from the states of California and Texas.

10 Extra Analysis Attempt

Here we tried to scrape for the keywords such as 'track', 'new track', etc. to then make a list based off tweets where people mention an artist (so instead of scraping for the actual artist name, we "reversed engineered" the scraper to get keywords that might mention an artist name. We gave it a good attempt, but did not get too much data for the given time frame.

10.1 Keywords that were scraped

To create this list, we scraped for the following keywords for about 5 hours

- "Music, New Music, great music, just listened to, new release, new song, great track, track"

In [75]:

```
#stream.statuses.filter(track="Music, New Music, great music, just listened t
```

10.2 Our organic list based on tweets

In [99]:

```
import pandas as pd
import re

new_df = pd.read_csv("data_files/twitter_output_OurList.csv")
new_df.columns= ['Hashtags','ID','Tweet_Text','Name','Username','User_Locatio

col  = new_df.Tweet_Text
pages = col.to_string()
r = r"([A-Z][a-z]+ [A-Z][a-z]+)"
regex = re.compile(r)
match = regex.findall(pages)
match
x = pd.Series(match)
x.value_counts()
top10 = x.head(20)

#here we took what would match as a first name and last name, and got the cou
#but not sure how useful and accurate this would be, becuase the first artist
top_10  =pd.DataFrame(top10)
top_10.columns = ["Our Name"]
top_10
```

Out[99]:

	Our Name
0	Stone Hello
1	Music Re
2	Solar Won
3	Virtual Community
4	Doug Fords
5	Korean Music
6	Big Machine
7	The South
8	Atlantic Conferen
9	Rapper Hae
10	Agent Francis
11	York Morgan
12	Only Girl
13	In The
14	Adriana Lima
15	Daft Punk
16	Kazunari Ninomiya
17	Taeyeon Shares
18	The Container
19	Ship Greetje

In [69]:

```
x.iloc[15] #looks much different than the Apple and Spotify top 10.
```

Out[69]:

```
'Daft Punk'
```

In [12]:

```
new_df = pd.read_csv("data_files/AllArtists.csv")
artists = new_df["name"]
Artist_List = pd.DataFrame(artists)
Artist_List
```

Out[12]:

	name
0	Adele
1	Joey + Rory
2	Draaco Aventura
3	Justin Bieber
4	Peer van Mladen
...	...
2994	Crosby, Stills, Nash & Young
2995	CRU
2996	Crystal Waters
2997	Crazy Town
2998	Cynthia Fetty

2999 rows × 1 columns

11 Geo-Map

In [6]:

```
from mpl_toolkits.basemap import Basemap
```

In [7]:

```
from geopy.geocoders import Nominatim
```

In [33]:

```
location = df["User_Location"].value_counts()
new_loc = location.to_string()
location #Taking the frequency of each location
locc = pd.DataFrame(location)
locc
```

Out[33]:

	User_Location
United States	1195
Los Angeles, CA	950
California, USA	813
London, England	572
Houston, TX	520
...	...
BANGWINPINKONTREJO	1
My Location Unknow	1
Ahmadabad City, India	1
Des Moines, USA	1
East Helena, MT	1

26444 rows × 1 columns

Here we are taking the top values from the Location and then taking the top 300 locations. We are taking the top 300, becuase if we attempt to pass in all 100,000 tweets, our kernel will time out. Due to this, the most important 300 are used.

In [90]:

```
e = df.User_Location.dropna()
ee = pd.DataFrame(e)
zg = ee.User_Location.value_counts().head(300)
ZG = zg.index
ZLG= pd.DataFrame(ZG)
ZLG.columns = ["Location"]
ZLG
```

Out[90]:

	Location
0	United States
1	Los Angeles, CA
2	California, USA
3	London, England
4	Houston, TX
...	...
295	Ann Arbor, MI
296	Kenya
297	The Netherlands
298	Baltimore
299	The Bahamas

300 rows × 1 columns

In [54]:

```
e = df.User_Location.dropna()
ee = pd.DataFrame(e)
zg = ee.User_Location.value_counts().head(300)
count = pd.DataFrame(zg)
count
```

Out[54]:

	User_Location
United States	1195
Los Angeles, CA	950
California, USA	813
London, England	572
Houston, TX	520
...	...
Ann Arbor, MI	20
Kenya	20
The Netherlands	20
Baltimore	20
The Bahamas	20

300 rows × 1 columns

As far as we were able to get with the geo-plot. Were researching ways to plot, found geopy but ran out of time.

In [11]:

```
pip install geopy
```

```
Requirement already satisfied: geopy in /usr/local/lib/python3.6/site-packa
ges (1.21.0)
Requirement already satisfied: geographiclib<2,>=1.49 in /usr/local/lib/pyt
hon3.6/site-packages (from geopy) (1.50)
Note: you may need to restart the kernel to use updated packages.
```

In [12]:

```
import geopy
from geopy.geocoders import Nominatim
```

In [13]:

```
nom = Nominatim()
```

```
/usr/local/lib/python3.6/site-packages/ipykernel_launcher.py:1: Deprecation
Warning: Using Nominatim with the default "geopy/1.21.0" `user_agent` is st
rongly discouraged, as it violates Nominatim's ToS https://operations.osmfo
undation.org/policies/nominatim/ (https://operations.osmfoundation.org/poli
cies/nominatim/) and may possibly cause 403 and 429 HTTP errors. Please spe
cify a custom `user_agent` with `Nominatim(user_agent="my-application")` or
by overriding the default `user_agent`: `geopy.geocoders.options.default_us
er_agent = "my-application"`. In geopy 2.0 this will become an exception.
  """Entry point for launching an IPython kernel.
```

In [16]:

```
ZLG["Coordinates"] = ZLG["Location"].apply(nom.geocode)
```

In [17]:

```
ZLG.dropna()
```

Out[17]:

	Location	Coordinates
0	United States	(United States, (39.7837304, -100.4458825))
1	Los Angeles, CA	(Los Angeles, Los Angeles County, California, ...
2	California, USA	(California, United States of America, (36.701...
3	London, England	(London, Greater London, England, SW1A 2DX, Un...
4	Houston, TX	(Houston, Harris County, Texas, United States ...
...	...	...
295	Ann Arbor, MI	(Ann Arbor, Washtenaw County, Michigan, United...
296	Kenya	(Kenya, (1.4419683, 38.4313975))
297	The Netherlands	(Nederland, (52.5001698, 5.7480821))
298	Baltimore	(Baltimore, Maryland, 21203, United States of ...
299	The Bahamas	(The Bahamas, (24.7736546, -78.0000547))

297 rows × 2 columns

In [18]:

```
ZLG["Latitude"] = ZLG["Coordinates"].apply(lambda x: x.latitude if x != None
ZLG["Longitude"] = ZLG["Coordinates"].apply(lambda x: x.longitude if x != Nor
ZLG.dropna()
```

Out[18]:

	Location	Coordinates	Latitude	Longitude
0	United States	(United States, (39.7837304, -100.4458825))	39.783730	-100.445882
1	Los Angeles, CA	(Los Angeles, Los Angeles County, California, ...	34.053691	-118.242767
2	California, USA	(California, United States of America, (36.701...	36.701463	-118.755997
3	London, England	(London, Greater London, England, SW1A 2DX, Un...	51.507322	-0.127647
4	Houston, TX	(Houston, Harris County, Texas, United States ...	29.758938	-95.367697
...	...	...	...	...
295	Ann Arbor, MI	(Ann Arbor, Washtenaw County, Michigan, United...	42.268157	-83.731229
296	Kenya	(Kenya, (1.4419683, 38.4313975))	1.441968	38.431398
297	The Netherlands	(Nederland, (52.5001698, 5.7480821))	52.500170	5.748082
298	Baltimore	(Baltimore, Maryland, 21203, United States of ...	39.290882	-76.610759
299	The Bahamas	(The Bahamas, (24.7736546, -78.0000547))	24.773655	-78.000055

297 rows × 4 columns

In [89]:

```
Lat = ZLG["Latitude"]
Long = ZLG["Longitude"]
```

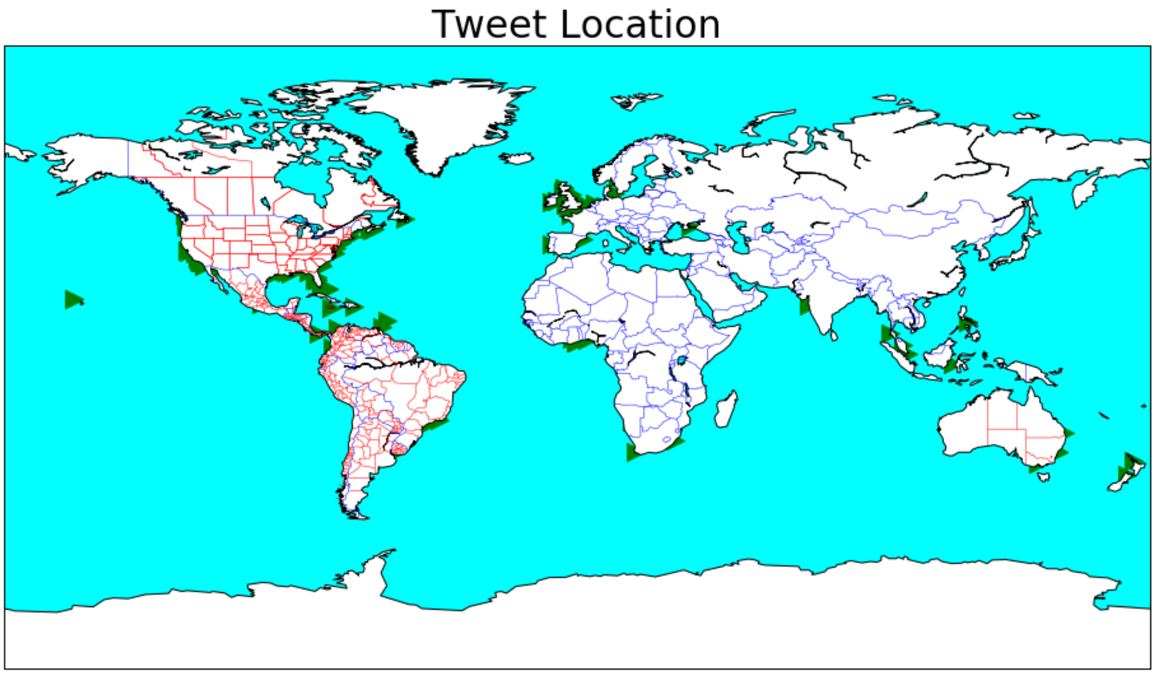
In [24]:

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import seaborn as sns #Statistical Data Visualization
import matplotlib.pyplot as plt
%matplotlib inline
```

In [88]:

```
from mpl_toolkits.basemap import Basemap
import matplotlib.pyplot as plt
fig = plt.figure(figsize=(20,9))
m = Basemap(projection='gall',
            resolution = 'c')
m.drawcoastlines()
m.drawcountries(color= 'blue')
m.drawstates(color= 'red')
m.drawmapboundary(fill_color='aqua')
m.fillcontinents(color='white',lake_color='aqua')
plt.title("Tweet Location",fontsize = 30)
Lat = list(ZLG["Latitude"])
Long = list(ZLG["Longitude"])
m.scatter(Long,Lat,latlon=True,color="green",marker = ">",s = 200)
plt.show()
```

```
/usr/local/lib/python3.6/site-packages/mpl_toolkits/basemap/__init__.py:478
8: RuntimeWarning: invalid value encountered in greater
  lonsin = np.where(lonsin > lon_0+180, lonsin-360 ,lonsin)
/usr/local/lib/python3.6/site-packages/mpl_toolkits/basemap/__init__.py:478
9: RuntimeWarning: invalid value encountered in less
  lonsin = np.where(lonsin < lon_0-180, lonsin+360 ,lonsin)
/usr/local/lib/python3.6/site-packages/mpl_toolkits/basemap/__init__.py:479
5: RuntimeWarning: invalid value encountered in greater_equal
  itemindex = len(lonsin)-np.where(londiff>=thresh)[0]
/usr/local/lib/python3.6/site-packages/mpl_toolkits/basemap/__init__.py:482
6: RuntimeWarning: invalid value encountered in less
  mask = np.logical_or(lonsin<lon_0-180,lonsin>lon_0+180)
/usr/local/lib/python3.6/site-packages/mpl_toolkits/basemap/__init__.py:482
6: RuntimeWarning: invalid value encountered in greater
  mask = np.logical_or(lonsin<lon_0-180,lonsin>lon_0+180)
```



In [ ]: