STA 141B Final Project: Analysis of Drake's Success and Music Career

Group 10

Group Members:

1. Aditya Mittal: 919336522
2. Olga Perez: 918845988
3. Sabrina Saypanya: 917363474
4. Sana Bharadwaj: 918016063

Project Outline

Attached below is an basic guideline of the structure project:

1. **Part One:** Motivations
2. **Part Two:** Visualization of Drake's popularity over the past decade
3. **Part Three:** Clustering Drake's music by song attributes
4. **Part Four:** Understanding people's preferences between old vs new Drake
5. **Part Five:** Sentiment analysis on Drake's rising popularity
6. **Part Six:** Conclusions

Part One: Motivations

On average, the career span of a musical artist is approximately between 1-3 years. The reason behind their short careers is simple: music tastes are constantly evolving through time and across different audiences, so it is extremely hard for artists to gain popularity and maintain sustained success. Most often, we see new artists come and pass after their "15 minutes of fame" as people move on to newer musicians. For example, hip-hop artist Fetty Wap dominated the 2015-2016 rap scene with his two hit singles "Trap Queen" and "1738". Now, as people have moved on to different artists, the numbers on his music streams have reduced dramatically. Artists must continue to meet the increasingly difficult expectations of their listeners if they want to maintain a successful long-term career. As such, many artists experience an extremely short span of popularity before it fades away and people move on.

That being said, Canadian rapper Drake is Spotify’s most streamed artist – his sustained success over the past decade is unparalleled as he continues to dominate the music industry. His Spotify achievements are endless: In 2016, his song, “One Dance” became the first song to reach one billion streams on the platform as he was awarded Spotify’s most streamed male artist. In the following year, he became the first artist to reach 10 Billion streams. In 2021, he continued to break records by reaching 50 Billion Streams and has currently surpassed over 75 Billion streams. When it comes to Drake, there is no comparison between artists in terms of long-term measured success.

With music as such a fundamental aspect of our society, it is extremely impressive for an individual to remain connected to millions of listeners globally for a span of over a decade. This ultimately piqued our curiosity – how can one artist continue to remain relevant for so long?

Our group addressed the following questions to gain a better understanding of Drake's musical career over the past decade:

- How can we better understand Drake's career in terms of popularity over the past decade?
- Can we group his music by song attributes, i.e. similar music? Is there an association between certain song attributes and Drake's most popular songs?
- Do people have preference between Drake's old music or Drake's new music?
- How has Drake's rising popularity been received by the broader audiences?

Import Modules

In [1]:
```
# Import necessary libraries
import pandas as pd
import requests
import numpy as np
import matplotlib.pyplot as plt
from bs4 import BeautifulSoup as bs
import json
from wordcloud import WordCloud, STOPWORDS, ImageColorGenerator
from PIL import Image
from nltk.sentiment.vader import SentimentIntensityAnalyzer
from googleapiclient.discovery import build
import nltk
from nltk import tokenize
import pycountry
import plotly.express as px
import plotly.graph_objects as go
import plotly.io as pio
import seaborn as sns
import matplotlib.pyplot as plt
from matplotlib import pyplot as plt
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
from sklearn import preprocessing
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans
from kneed import KneeLocator
from matplotlib import pyplot as plt
import plotly.graph_objects as go
```

```
from collections import Counter
import warnings
from collections import Counter
warnings.filterwarnings('ignore')
```

Part Two: Visualizing Drake's Popularity

Debut Album Streams in United States vs. Globally

In order to gain insight into Drake’s rising popularity and its reception among wider audiences, we web-scraped data from ChartMasters on Drake’s top debuted albums on Spotify between 2016 and 2022. In particular, our data focused on the albums 'Views,' 'More Life,' 'Scorpion,' 'Dark Lane Demo Tapes,' 'Certified Lover Boy,' 'Honestly, Nevermind,' and 'Her Loss,'; we included their overall rankings and the total streams by country.

First, we extracted data corresponding to each album using Requests and BeautifulSoup. The url is first requested and stored as a response, then the content of the response is parsed using BeautifulSoup to extract the HTML code of the webpage. We found the tables for each albums, extracted the data and headers from the corresponding tables, and then converted each of them into a pandas dataframes. Next, we merged the dataframes for each album into one larger dataframe. In order to determine how popular Drake's album debuts were in the U.S. and Globally, we used the merged dataset to extract the streams for each album:

In [2]:
```
### Web-Scraping Data from Chartmaster for Debut Album Streams

# Functions to extract data from ChartMasters
def extract_album_chart(table_object, album_name):
    '''This function returns a dataframe containing popularity information f
    to the beautiful_soup object containing the data. Album_name is the stri
    # get list of items from soup object
    header_table = [th.get_text().strip() for th in table_object.find_all('t
    rows_table = table_object.find_all('tr')[1:] # extract row data
    data_table = [[td.get_text().strip() for td in row.find_all('td')] for r
    album_data = [row.find_all('td')[5].get_text().strip() for row in rows_t

    # convert list items to a dataframe
    df = pd.DataFrame(data_table, columns=header_table).assign(Album=album_r
    df['Debut Sales'] = album_data # add column
    df = df.drop(['Average', 'Songs', 'Top'], axis=1) # drop columns
    return(df)

def get_country_code(country):
    '''This function returns the ISO alpha-3 code for a country name'''
    if country == 'United States of America':
        return 'USA'
    elif country == 'Global':
        return 'OWID_WRL'
    elif country == 'South Korea':
        return 'KOR'
    elif country == 'Czech Republic':
        return 'CZE'
    elif country == 'Taiwan':
        return 'TWN'
    else:
        try:
            return pycountry.countries.get(name=country).alpha_3
        except AttributeError:
            return None

# Web-scrape global chart data for Drake albums
headers = {'User-Agent': 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_2) Ap

url_song_data = 'https://chartmasters.org/spotify-top-album-debuts/?view=art
response = requests.get(url_song_data, headers=headers) # get response
bs_object = bs(response.content, 'html.parser') # create beautiful soup obje
table_views = bs_object.find_all('table')[0] # get 'views' data
table_more_life = bs_object.find_all('table')[1] # 'get more life' data
table_scorpion = bs_object.find_all('table')[2] # 'get scorpion' data
table_dldt = bs_object.find_all('table')[3] # get 'dark lane demo tapes' dat
table_clb = bs_object.find_all('table')[4] # get 'clb' data
table_hn = bs_object.find_all('table')[5] # get 'honestly, nevermind' data
table_hl = bs_object.find_all('table')[6] # get 'her loss' data

# create dataframe
views_df = extract_album_chart(table_views, "Views")
morelife_df = extract_album_chart(table_more_life, "More Life")
scorpion_df = extract_album_chart(table_scorpion, "Scorpion")
dldt_df = extract_album_chart(table_dldt, "Dark Lane Demo Tapes")
clb_df = extract_album_chart(table_clb, "Certified Lover Boy")
hn_df = extract_album_chart(table_hn, "Honestly, Nevermind")
hl_df = extract_album_chart(table_hl, "Her Loss")

# Data pre-processing
discography = pd.concat([views_df, morelife_df, scorpion_df, dldt_df, clb_df
duplicates = discography[discography.duplicated()] # Identify duplicates
discography = discography.drop_duplicates() # Drop duplicates

# Merge the concatenated dataframe on the 'Country', 'Streams', 'Album', and
merged = pd.merge(discography, discography, on=['Country', 'Streams', 'Album
del merged["_y"] # remove columns
del merged["_x"]

# Top streams from US and Global from each Album
subset = merged[['Country', 'Album', 'Streams']] # Subset the merged dataset
# Filter the dataset to include only rows where the 'Country' column is eith
filtered = subset[subset['Country'].isin(['United States of America', 'Globa
# Group the dataset by 'Country' and 'Album' and sort each group by 'Streams
grouped = filtered.groupby(['Country', 'Album']).apply(lambda x: x.sort_valu
top5 = grouped.groupby(['Country', 'Album']).head(5) # Select the top 5 rows

# Concatenate the results for the USA and Global groups
usa_top5 = top5[top5['Country'] == 'United States of America']
global_top5 = top5[top5['Country'] == 'Global']
```

```
tops = pd.concat([usa_top5, global_top5], ignore_index=True)
print(tops)
```

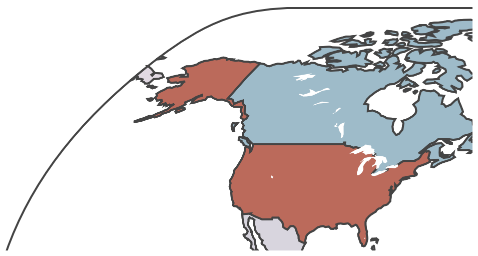
```
                     Country                Album      Streams
0   United States of America                Views   87,782,182
1   United States of America            More Life  161,699,615
2   United States of America             Scorpion  313,146,379
3   United States of America  Dark Lane Demo Tapes   80,872,881
4   United States of America   Certified Lover Boy  248,543,876
5   United States of America   Honestly, Nevermind   94,526,690
6   United States of America             Her Loss  190,708,458
7                     Global                Views  161,937,271
8                     Global            More Life  290,224,416
9                     Global             Scorpion  559,021,928
10                    Global  Dark Lane Demo Tapes  169,038,380
11                    Global   Certified Lover Boy  497,083,050
12                    Global   Honestly, Nevermind  212,108,781
13                    Global             Her Loss  381,665,545
```

From the results above, we can see that Drake's album streams charted very high numbers both in the United States and Globally. In particular, his 2018 album "Scorpion" had the highest streams with 313 Million streams in the US and almost 560 Million streams globally. Similarly, his newer albums "Certified Lover Boy" and "Her Loss" have charted extremely high debut streams both in US and globally as well. Upon initial glance, Drake's continued success is evident as each album from 2016 - 2022 (Views to Her Loss) has resulted in very high streaming numbers across the world.

Additionally, we've included a choropleth map to show the total number of debuts stream of the albums by country. The map has a color scale representing the stream numbers and uses country codes to display the data. The color bar ranges from 0 to 1.5 billion, with the midpoint being set at 100 million. Hovering over each country displays its corresponding stream count.

In [3]:
```
### Choropleth Map for Drake Debut Album Streams
merged['Streams'] = merged['Streams'].astype(str) # Convert the column to st
merged['Streams'] = merged['Streams'].str.replace(',', '').astype(int) # .st
tot = merged.groupby('Country')['Streams'].sum().reset_index() # finding the
total_streams = tot.reset_index() # convert back to dataframe
total_streams['Country Code'] = total_streams['Country'].apply(get_country_c

#Create a choropleth map using the 'choropleth' function
fig = px.choropleth(total_streams, locations='Country Code',
                        color='Streams',
                        color_continuous_scale='twilight',
                        range_color=[0,1500000000],
                        color_continuous_midpoint=100000000,
                        hover_name='Country',
                        title='Drake Total Debut Stream Popularity',
                        projection='natural earth')
fig.update_layout(coloraxis_colorbar=dict(
    tickmode='linear', # Set the tick mode to linear
    dtick=200000000,
    tick0=0
))
fig.show()
```

Drake Total Debut Stream Popularity

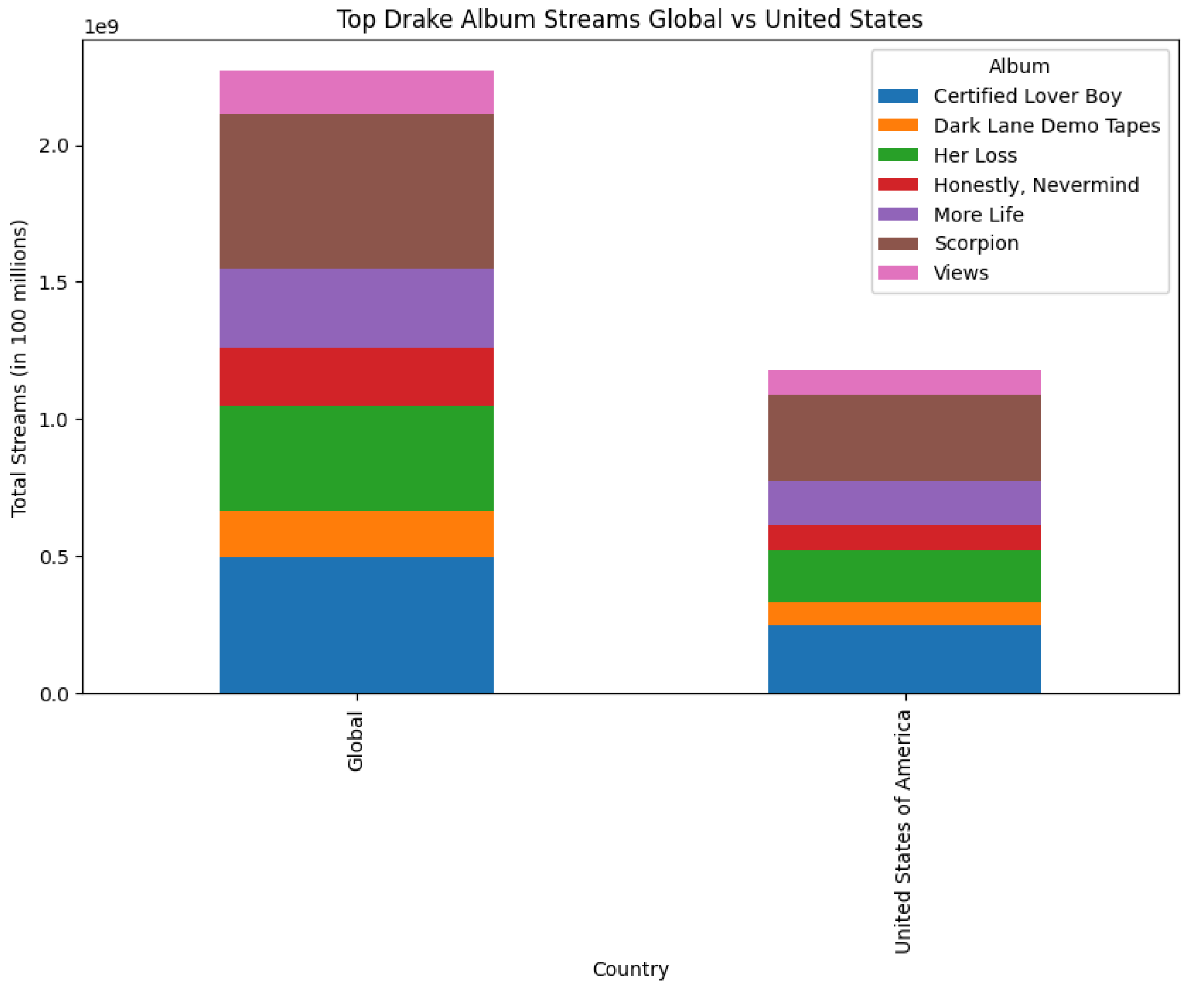


Based on our data, we found that Drake's music has been well received globally, with the highest streams in the North American Region. Furthermore, the 'Drake Total Debut Stream Popularity' map above displays that his music has accumulated a significant number of streams in parts of Europe, Asia, and Africa, suggesting that his music has been well received by a broader audience. His popularity beyond the U.S. has led to increased exposure and popularity, which has rendered into higher record sales and greater opportunities to refine his music. Drake's popularity with a broader audience has allowed him to dominate the music industry, and as a result, he has continued to undergo great success for so long.

In [4]:
```
### Box Plot for Drake Debut Album Streams
# add numbers to boxplot
tops['Streams'] = tops['Streams'].astype(str) # Convert the column to string
tops['Streams'] = tops['Streams'].str.replace(',', '').astype(int) # apply t

# Group and plot the data
grouped_data = tops.groupby(['Country', 'Album'])['Streams'].sum().unstack()
ax = grouped_data.plot(kind='bar', stacked=True, figsize=(10, 6))
```

```
ax.set_title('Top Drake Album Streams Global vs United States')
ax.set_xlabel('Country')
ax.set_ylabel('Total Streams (in 100 millions)')
```

```
Out[4]: Text(0, 0.5, 'Total Streams (in 100 millions)')
```



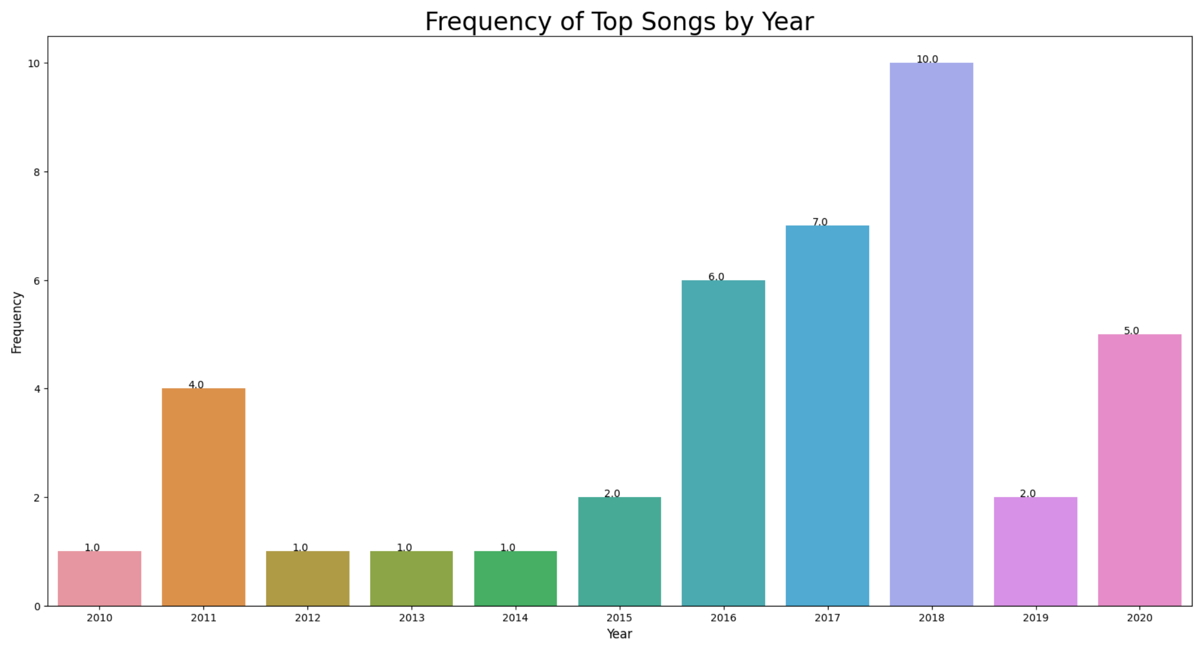
The amount of debut streams Drake received from each album reflects the popularity and success of his releases. By observing the barplot, it is apparent that the debut of each album was well received in the United States and Globally. The comparison between the two also indicates a high number of streams globally, suggesting Drake's increasing popularity beyond just the United States. We can also observe that Scorpion was among one of the most popular debut album's which will be dicussed further in a different section below.

Drake's Top 40 Streamed Songs

In addition to album streams, we web-scraped data from ChartMasters regarding Drake's top 40 most streamed songs over the course of his career. In particular, we are interested in finding out potential trends between Drake's career timeline and the release of his most streamed songs.

Using similar techniques as before, we used BeautifulSoup to extract a table from the website and then converted it into a dataframe. We then created a box-plot to get the frequency of to songs by year. From the plot below, we can see that the majority of his top 40 streamed songs were produced between 2016 and 2018. More specifically, 23/40 (more than half) of his top songs were produced during the span of these two years. These results indicate that Drake produced his best, most widely-accepted popular music during the span of these two years. This is further supported from our results above as the streams from Scorpion (2018) were highest both in the U.S and globally.

In [5]:
```
### Web-Scrape Data for Drake's Top 40 Songs
url = 'https://www.officialcharts.com/chart-news/drakes-official-top-40-most
page = requests.get(url)
soup = bs(page.text, "html.parser") # convert string into BeautifulSoup obje
table = soup.find_all('table') # using find_all method get all tables

# convert soup item to dataframe
res1 = [] # initialize empty list
for tr in table[0].find_all('tr'): # iterate over for loop
    td = tr.find_all('td') # find_all method for 'td'
    row = [tr.text.strip() for tr in td if tr.text.strip()] # strip our data
    if row:
        res1.append(row) # append data in our empty list

ranked_songs = pd.DataFrame(res1, columns=["Position", "Title", "Artist","Pe
ranked_songs = ranked_songs.drop(0) # drop row 0
ranked_songs['Title'] = ranked_songs['Title'].apply(str.lower) # lowercase s

ranked_songs.loc[25, 'Year'] = 2018 # fixing error from website

 # changing specific song names to match with spotify API data
ranked_songs.loc[16, 'Title'] = 'hold on, we\'re going home'
ranked_songs.loc[24, 'Title'] = 'don’t matter to me'
ranked_songs.loc[29, 'Title'] = 'wanna know remix'
ranked_songs.loc[32, 'Title'] = 'what\'s my name?'
ranked_songs.loc[36, 'Title'] = 'marvins room'
ranked_songs['Year'] = ranked_songs['Year'].astype(int) # convert year colum
ranked_songs = ranked_songs.sort_values(by=['Year'])

### Box-plot
fig, ax = plt.subplots(figsize=(20, 10)) # change plot size
p2 = sns.countplot(x=ranked_songs["Year"]) # create box plot
for p in p2.patches: # get counts for each year
    p2.annotate('{:.1f}'.format(p.get_height()), (p.get_x()+0.25, p.get_heig
plt.xlabel("Year" , size = 12) # set axis
plt.ylabel("Frequency" , size = 12)
plt.title("Frequency of Top Songs by Year" , size = 24)
plt.show()
```



Measuring Track Popularity From Spotify API

Next, we extracted Drake's entire discography using the Spotify API. Using a user curated playlist containing all of Drake's music, we extracted individual song attributes for each track across Drake's musical career.

These following features are included for each song:

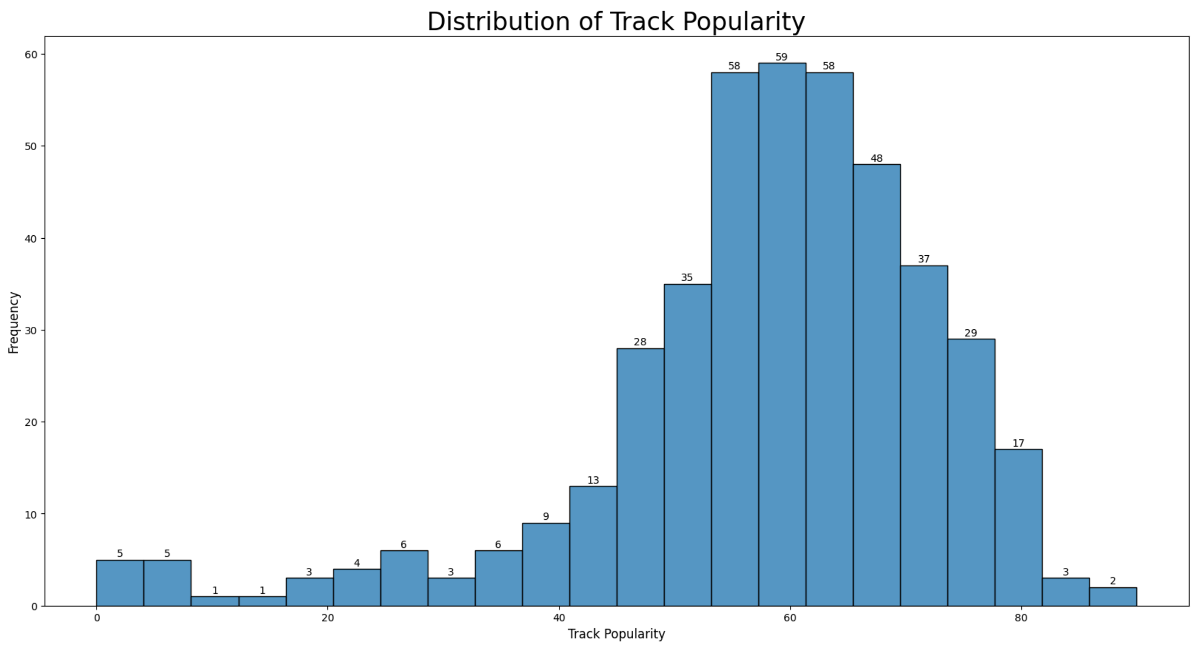
- Danceability: Danceability describes how suitable a track is for dancing based on a combination of musical elements including tempo, rhythm stability, beat strength, and overall regularity.
- Energy: Energy is a measure from 0.0 to 1.0 and represents a perceptual measure of intensity and activity. Typically, energetic tracks feel fast, loud, and noisy.
- Key: The key the track is in. Integers map to pitches using standard Pitch Class notation.
- Loudness: The overall loudness of a track in decibels (dB).
- Mode: Mode indicates the modality (major or minor) of a track, the type of scale from which its melodic content is derived.
- Speechiness: Speechiness detects the presence of spoken words in a track.
- Acousticness: A confidence measure from 0.0 to 1.0 of whether the track is acoustic.
- Instrumentalness: Predicts whether a track contains no vocals.
- Liveness: Detects the presence of an audience in the recording.
- Valence: Measure of Positivity
- Tempo: The overall estimated tempo of a track in beats per minute (BPM).
- Duration_ms: The duration of the track in milliseconds.
- Time_signature: The time signature (meter) is a notational convention to specify how many beats are in each bar (or measure).

- Track popularity: Popularity rating
- Track name: Track name
- Album_name: Name of the album
- Release_date: Release date
- Year: Release year

We then created a histogram for the all of the track_popularity measures from the Spotify API for Drake's entire discography:

In [6]:
```
### Extract data from Spotify API
# Create API keys
URL = 'https://api.spotify.com/v1/' # base url for spotify
MY_ID = '47dd696c104e49d1b6a9925ad26a829d' # client access ID
MY_SECRET = '0e52b5a013a64b058001e2f69d22ab96' # client secret ID
AUTH_URL = 'https://accounts.spotify.com/api/token' # authentication url
auth_response = requests.post(AUTH_URL, { # create authentication response
    'grant_type': 'client_credentials',
    'client_id': MY_ID,
    'client_secret': MY_SECRET,
})
my_key = auth_response.json()['access_token'] # get access token
headers = {
    'Authorization': 'Bearer {token}'.format(token=my_key) # create header f
}

# Functions to extract data from playlist
def extract_tracks(Playlist_ID):
    '''This function returns a list of tuples containing track name, id, pop
    Input requires the playlist ID from spotify.'''

    # Get first 50 results; Spotify API only returns the first 50 results at
    response_albums = requests.get(URL + 'playlists/' + Playlist_ID + '/trac
                                   headers=headers,
                                   params={'additional_types': 'track', 'limit':
    d1 = response_albums.json() # convert response into JSON list
    songs_info = list() # initialize empty list
    for song in d1['items']:   # add track information to the list
        songs_info.append((song['track']['name'], song['track']['id'], song[
                           song['track']['album']['release_date'], song['tra
    i = 0  # get data from next 50 entries
    while (i < d1['total']): # while loop to repeat code
        next_link = d1['next'] # extract URL for next 50 songs to extract
        if next_link is None: # conditional: if url is empty, end loop
            break
        response_albums = requests.get(next_link, headers=headers) # extract
        d1 = response_albums.json() # convert response into JSON
        for song in d1['items']: # append song info to our list
            songs_info.append((song['track']['name'], song['track']['id'], s
        i += 50 # add 50 to i and re-run loop
    return(songs_info) # return list

def extract_features(Playlist_ID):
    '''This function returns list of attributes for each song based on an in
    input_tuple = extract_tracks(Playlist_ID) # get track information from e
    data = list() # initialize empty list
    for i in range(0,len(input_tuple)): # run loop through song information
        response_track = requests.get(URL + 'audio-features/' + input_tuple[
        response_track = response_track.json() # convert response into json
        response_track.update({
            'track_popularity': input_tuple[i][2], # add track popularity fr
            'track_name': input_tuple[i][0], # add track name
            'release_date': input_tuple[i][3], # add album release date
            'album_name': input_tuple[i][4] # add album name
        })
        data.append(response_track) # append info into a list
    return(data)

playlist_id = '7b46c5syjtG86a77R7SnMs' # Drake playlist containing all the s
data_list = extract_features(playlist_id) # get list of song attributes base

df = pd.DataFrame(data_list) # convert to Pandas Dataframe

x = list() # empty list
for i in range(0,len(df)): # add year column
    x.append(df['release_date'][i][0:4])

df['Year'] = x # add 'Year' column
df['release_date'] = pd.to_datetime(df['release_date']) # change release_dat
df = df.sort_values(by='release_date') # sort by release date; oldest to new
df = df.drop(['type', 'id', 'uri','track_href','analysis_url'],axis = 1) # r
df['track_name'] = df['track_name'].apply(str.lower) # lowercase all track r
df['album_name'] = df['album_name'].apply(str.lower) # lowercase all album r
```

In [7]:
```
### Create Dataframe of just song attributes
song_att = df.drop(['track_name','release_date','album_name','Year'], axis =
# Distribution of track popularity
fig, ax = plt.subplots(figsize=(20, 10))
p1 = sns.histplot(data=song_att, x= 'track_popularity')
for p in p1.patches:
    ax.annotate(f'{p.get_height():.0f}\n',
                (p.get_x() + p.get_width() / 2, p.get_height()), ha='center'
plt.xlabel( "Track Popularity" , size = 12 )
plt.ylabel( "Frequency" , size = 12 )
plt.title( "Distribution of Track Popularity" , size = 24 )
plt.show()
```



From the histogram above, we can see the frequency of song's popularity measure through Drake's discography. We can see that the histogram is left skewed, signifying that quantitively Drake's songs are distributed amongst higher popularity scores. In particular, the peak of our histogram is at track_popularity of 60 with 59 songs, with several songs having extremely high popularity measures above 80. Furthermore, on the lower tail of our histogram, it is important to note the songs with lower popularity scores come from the beginning of his career before he became an established artist. In short, the results from this histogram indicate that Drake's music, in general, has been very well received through Spotify with many songs having high popularity scores.

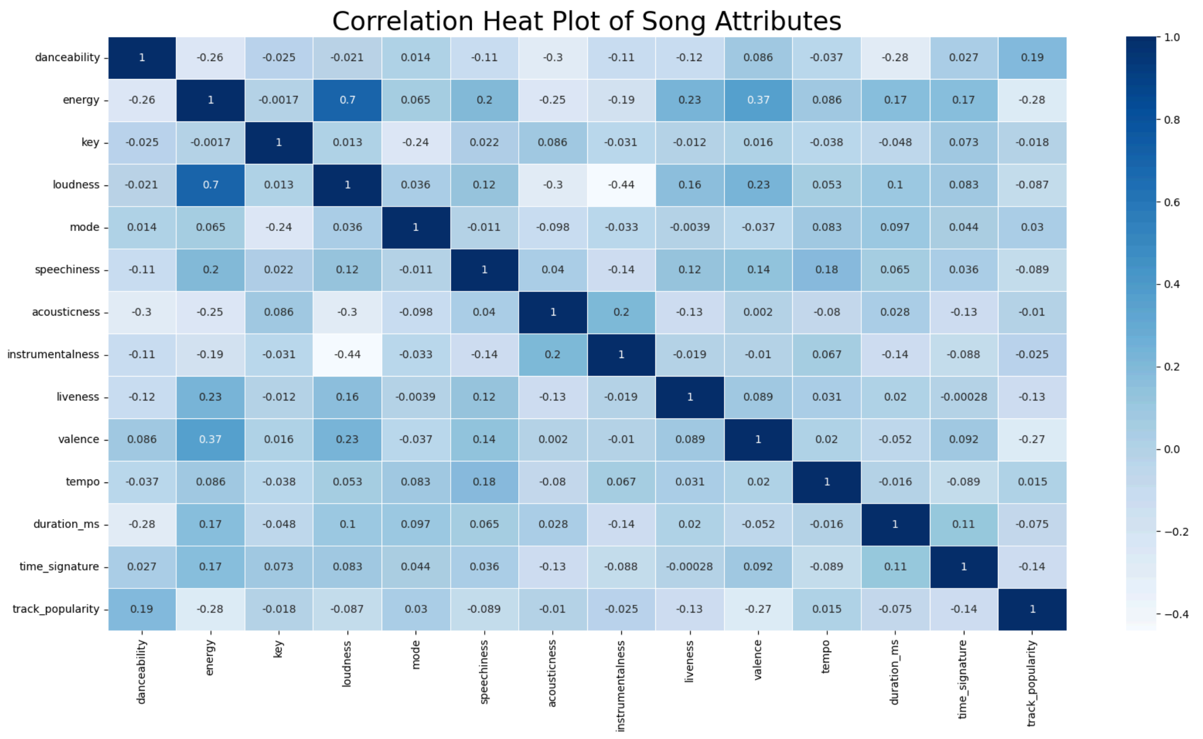
Based on our results until now, we can see Drake's sustained popularity comes from the fact his music charts high numbers globally and has allowed him to reach wider audiences. Furthermore, we saw that Drake produced his most popular music during the years 2016-2018, suggesting a possible timeline of the 'peak' of his career in terms of musical success. Ultimately, these visualization have helped us gain a better understanding of Drake's popularity over the past decade.

Part Three: Grouping Drake's Music by Song Attributes

Historically, Drake's music has been classified under the "Hip-Hop/Rap" genre. While this fact may have been true towards the beginning of his career, Drake's music has evolved significantly over the past decade and reached broader audiences than that of just hip-hop/rap. As a result, it is difficult to classify Drake's music under just one single genre as his record of music is extremely diverse. With this, our group was interested in learning more about the association between individual song attributes of his discography and his most popular songs; we're interested in creating clusters to label songs based on these attributes, i.e. "similar sounding music."

First, we created a correlation plot between all of the song attributes to see which features are most correlated with track popularity:

In [8]:
```
### Correlation Heat-Map between all song attributes
fig, ax = plt.subplots(figsize=(20, 10)) # set figure size
sns.heatmap(song_att.corr(), annot=True, linewidth=.5, cmap="Blues",ax = ax)
plt.title("Correlation Heat Plot of Song Attributes" , size = 24)
plt.show()
```



Based on our correlation plot above, we can see that there isn't much correlation between the song attributes. There appears to be a positive correlation of 0.7 between energy and loudness, which makes sense as energetic music often tends to be loud as well. Based on our initial results of song attributes in comparisions to track popularity, we can see that danceability and track_popularity seem to have slight positive correlation at 0.19, while both energy and valence (valence is the measure of positivity) are negatively correlated with track popularity. We will keep an eye on these attributes in particular when exploring our clusters of song groups.

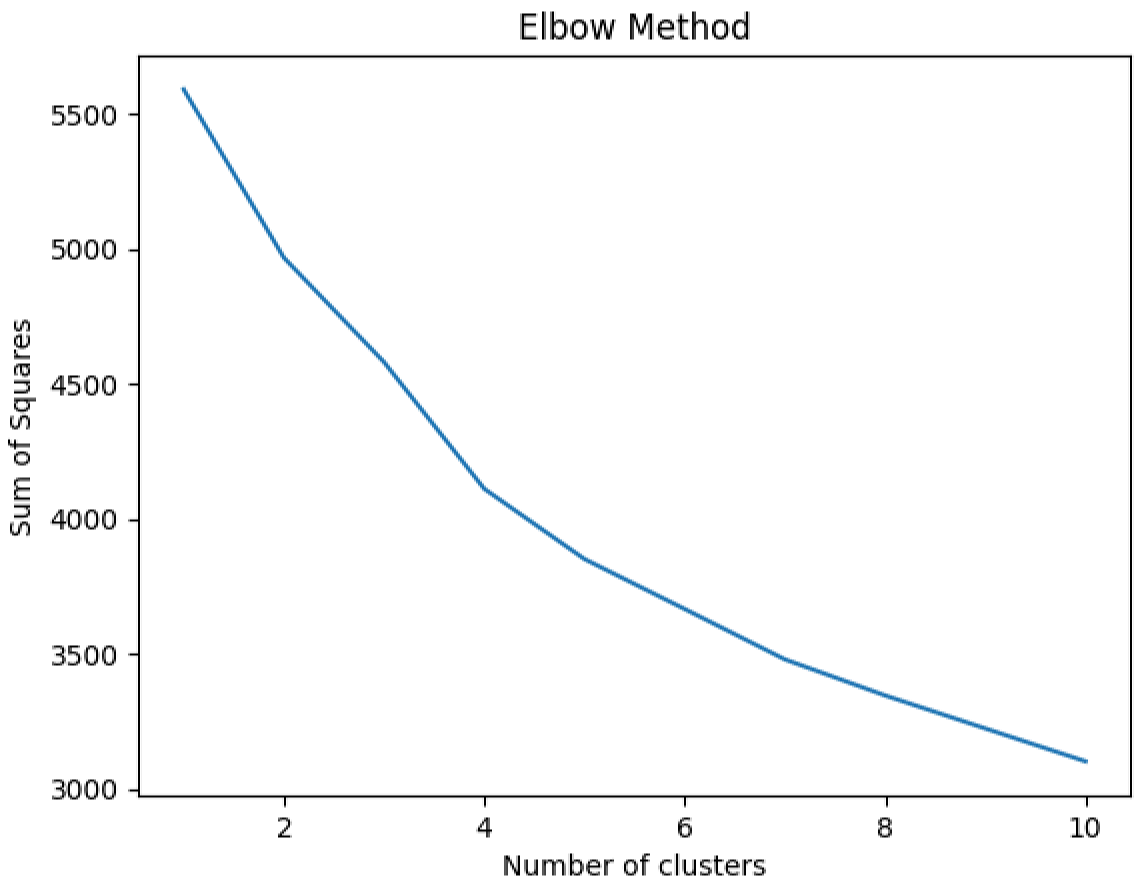
K-Means Clustering

Machine Learning techniques can be classified as either supervised or unsupervised. Unsupervised machine learning uses algorithms to analyze and cluster unlabeled datasets in order to identify hidden patterns or data groupings. For our purposes, we use K-means clustering to create groups based on the song attributes and find differences between these cluster to see if there's an association between a particular song attribute and his most popular music.

K-means clustering creates 'K' random centroids across the data and labels the data points. This is an iterative process that assigns each data point to a groups and slowly data points get clustered based on similar features. The objective of this algorthim is to minimize the sum of distances between the data points and the cluster centroid and identify the correct group each data point should belong to.

Our first step was to determine how many clusters to create. To do this, we used the Elbow Method: we plotted the explained variation as a function of the number of clusters and chose the "elbow" of the curve created to be the number of clusters for our data.

In [9]:
```
# Elbow Method
song_attribute_df = df.drop(['track_name','release_date','Year','track_popul

# standardize our data for k-means
scaler = StandardScaler()
X_std = scaler.fit_transform(song_attribute_df) # create data-frame
X_std = pd.DataFrame(X_std) # create dataframe

# Find number of clusters using Elbow Method
sse = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters=i, init='k-means++', max_iter=300, n_init=10,
    kmeans.fit(X_std)
    sse.append(kmeans.inertia_)
plt.plot(range(1, 11), sse)
plt.title('Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('Sum of Squares')
plt.show()
```



```
In [10]: k2 = KneeLocator(
             range(1, 11), sse, curve="convex", direction="decreasing"
         )
         k2.elbow
```

```
Out[10]: 4
```

From the plot above, we can see that there is slight "bend" at k = 4; from the 'KneeLocator' library, we can see there is concavity at k = 4 which corroborates our results from the plot. Thus, we will label our music in four different groups. Using the built-in function from the scikit-learn package, we can employ the K-means algorithm to our Spotify Dataset.

```
In [11]: # K-means clustering using scikit learn package
         kmeans = KMeans(n_clusters=4, init='k-means++', random_state=0)
         kmeans.fit(X_std)
         df['Cluster'] = kmeans.labels_ # add labels
         song_attribute_df['Cluster'] = kmeans.labels_ # add labels

         cluster_df = song_attribute_df.groupby("Cluster").mean() # get avg song attr
         cluster_df
```

Out[11]:

Cluster	danceability	energy	key	loudness	mode	speechiness	acousticness
0	0.782221	0.547267	4.533333	-7.570821	0.538462	0.196814	0.097601
1	0.586354	0.447458	5.770833	-9.612677	0.395833	0.175522	0.452088
2	0.295000	0.056200	2.500000	-26.407500	0.500000	0.046450	0.971500
3	0.554445	0.752803	5.211679	-5.513606	0.598540	0.283928	0.180275

Based on our initial results of song attributes by cluster, we can see that cluster 0 has significantly higher average danceability than the other three groups. Furthermore, cluster 3 has a much higher average energy, valence, and speechiness compared to the other groups. In terms of most features, it seems like cluster 0 and cluster 1 are most similar (with the exception of danceability and acousticness, cluster 1 has a much higher avg acousticness than cluster 0 and cluster 3).

One interesting result we got was through cluster 2, which had extremely low average speechiness and energy compared to the other groups while having extremely high instrumentalness and acousticness. Upon closer look, we found that this cluster actually included only two songs, "Intro" from *Her Loss* and "Outro" from *So Far Gone*. We found that both of these songs are just instrumental versions without any vocals, which indicates why they were labeled differently through their clusters.

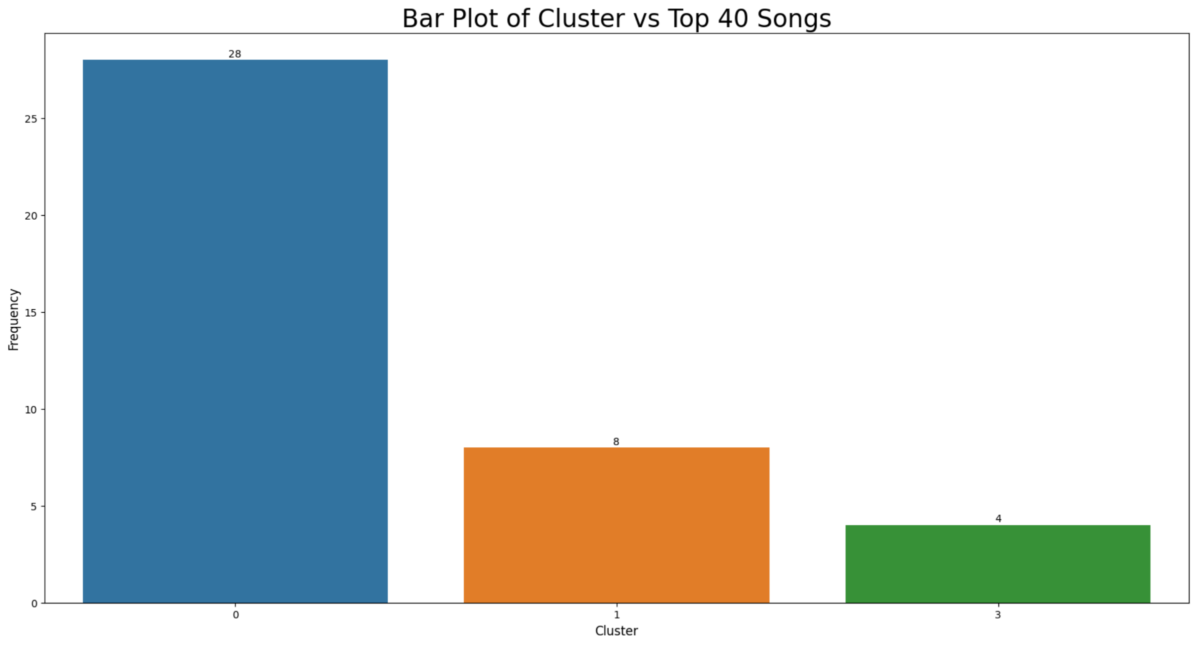
In essence, the result above indicates the broad diversity of Drake's musical career. For instance, songs in cluster 0 appear to be more of the 'pop' genre – provoking a more danceable feeling among its listeners and is meant for events with large audiences. Alternatively, songs in cluster 3 include significantly higher vocal elements and are meant for more live settings through its energetic feeling, such as concerts. We've already seen how cluster 2 is different from the rest of his discography. In short, although Drake has traditionally been classified as a rapper, we can see an extensively wide range of discography in terms of individual song attributes. Ultimately, this speaks to Drake's creativity as an artist in continually producing different sounds, allowing him to reach broader audiences and maintain his continued success.

Comparing Top 40 Songs by Cluster

Next, we're interested in comparing our results above with Drake's top 40 songs to see which cluster represents the highest proportion of his most popular music. To do this, we first merged the dataframe from our Spotify API with labeled clusters for each track and the web-scraped table from ChartMasters based on individual track names.

We then created a bar plot to get the frequency of clusters based on the top 40 songs:

In [12]:
```
### Create Histogram of Top 40 Songs by Cluster
def strip_footnote(x):
    """This function removes bracketed footnotes, such as '[1]'."""
    if pd.isna(x):
        return x
    return x.partition("(")[0]

txt = df['track_name'].apply(strip_footnote)
df['track_name'] = txt
df['track_name'] = df['track_name'].str.strip()

txt2 = ranked_songs['Title'].apply(strip_footnote)
ranked_songs['Title'] = txt2
ranked_songs['Title'] = ranked_songs['Title'].str.strip()

condensed_df = df[['track_name', 'Cluster', 'track_popularity']]
top_songs =  condensed_df.set_index('track_name').join(ranked_songs.set_inde
top_songs = top_songs.dropna()

fig, ax = plt.subplots(figsize=(20, 10))
p2 = sns.countplot(x=top_songs["Cluster"]) # frequency of 1's
for p in p2.patches:
    ax.annotate(f'{p.get_height():.0f}\n',
                (p.get_x() + p.get_width() / 2, p.get_height()), ha='center'
plt.xlabel( "Cluster" , size = 12 )
plt.ylabel( "Frequency" , size = 12 )
plt.title( "Bar Plot of Cluster vs Top 40 Songs" , size = 24 )
plt.show()
```



Based on our bar plot above, we can see that the overwhelming majority of his top 40 songs belong in cluster 0 with 28/40 songs included. Ultimately, this suggests that his most danceable music has traditionally been his most popular in terms of specific song features. This result was followed by cluster 1, which contains 8/40 of his most popular songs. In short, the results above have indicate that his song with high danceable, energetic attributes have traditionally been most well-receieved by the his audiences in terms of streams accumalated. This makes sense, as danceable music can be played in a variety of different settings and can help increase in overall streams.

Part Four: Reddit API

Drake's success in the music industry has been greatly influenced by the evolution of his discography. That being said, there is consensus among fans in general that a musician's songs become more commericialized, or "boring", once an artists gains popularity. With this, there has been debate between fans on whether the "Old" Drake or "New" Drake is better.

In order to determine people's preference on Old Drake vs New Drake, we extracted data from the Reddit API regardings the fans' prefences on Drake's music. The Reddit API allowed us to get people's opinions among a wide range of audiences; in particular, the following posts from the /r/Drizzy subreddit were analyzed based on the discussions from the users and the overall attention given to each of Drake's album:

- https://www.reddit.com/r/Drizzy/comments/u2xpzp/best_drake_album/
- https://www.reddit.com/r/Drizzy/comments/q2t6f0/ranking_drakes_discography/
- https://www.reddit.com/r/Drizzy/comments/92rvjc/what_are_your_top_5_songs_on_sco

In [13]:
```python
### get comments from reddit post
url1 = 'https://www.reddit.com/r/Drizzy/comments/u2xpzp/best_drake_album/.js
url2 = 'https://www.reddit.com/r/Drizzy/comments/q2t6f0/ranking_drakes_disco
url3 = 'https://www.reddit.com/r/Drizzy/comments/92rvjc/what_are_your_top_5_
params = {'limit': 100,}
headers = {'User-Agent': 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_2) Ap

def get_reddit_comments(url, params, headers):
    response = requests.get(url, params=params, headers=headers) # get respo
    data = response.json()
    comments = [] # Get the comments from the first set of comments
    for comment in data[1]['data']['children']:
        comments.append(comment['data']['body'])
    after = data[1]['data']['after'] # Get the 'after' parameter from the re

    while after is not None: # Loop through the next sets of comments until
        params['after'] = after  # Add the 'after' parameter to the paramete
        response = requests.get(url2, params=params, headers=headers)
        data = response.json()  # Extract the JSON data from the response
        for comment in data[1]['data']['children']: # Get the comments from
            comments.append(comment['data']['body'])
        after = data[1]['data']['after'] # Get the 'after' parameter from th
    return(comments)

c1 = get_reddit_comments(url1, params, headers) # get comments from subreddi
c2= get_reddit_comments(url2, params, headers) # get comments from subreddit
c1 = [w.lower() for w in c1] # lowercase comments
c2 =[w.lower() for w in c2]
```

```
In [14]: c1[10] # example of comment
```

```
Out[14]: 'take care'
```

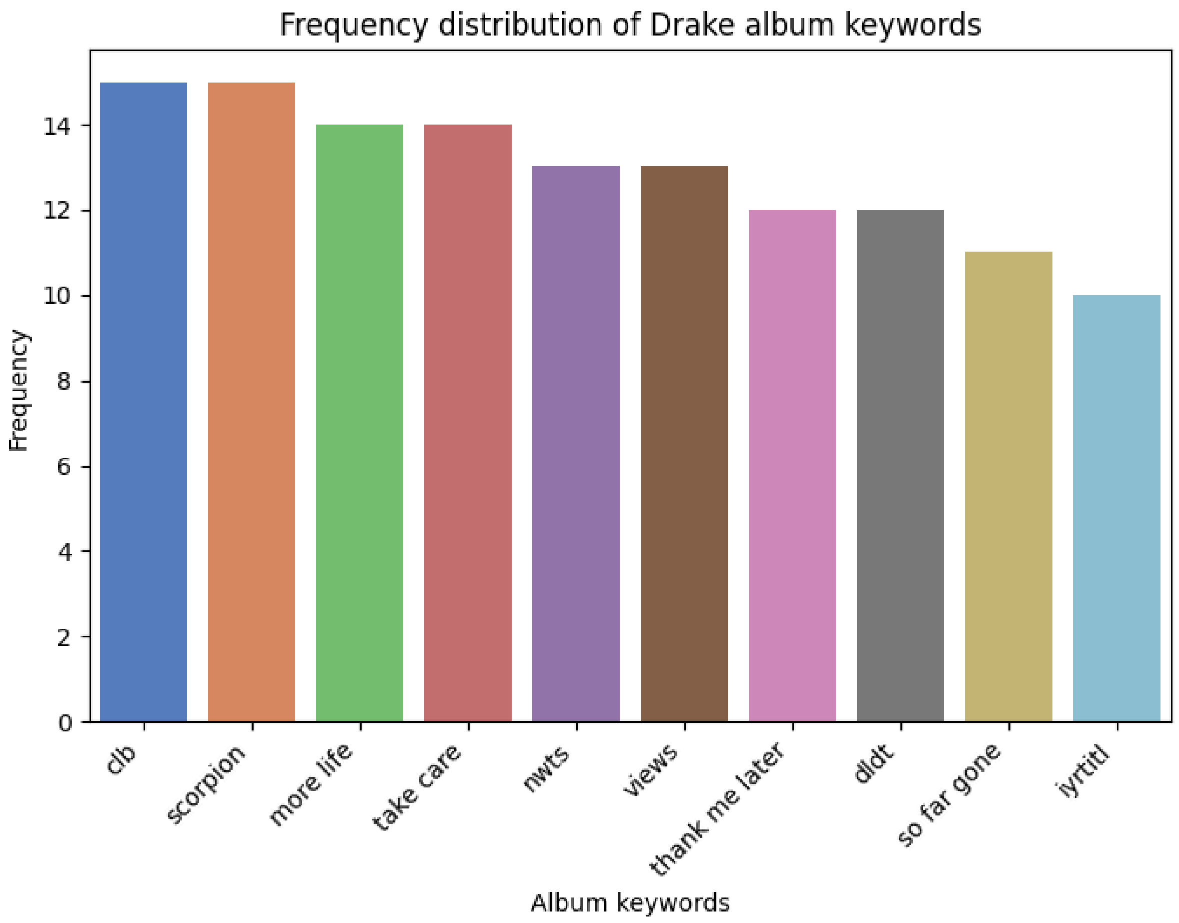
```
In [15]: c2[12] # example of comment
```

```
Out[15]: "clb should be higher. otherwise mine's similiar"
```

Key Word Counts

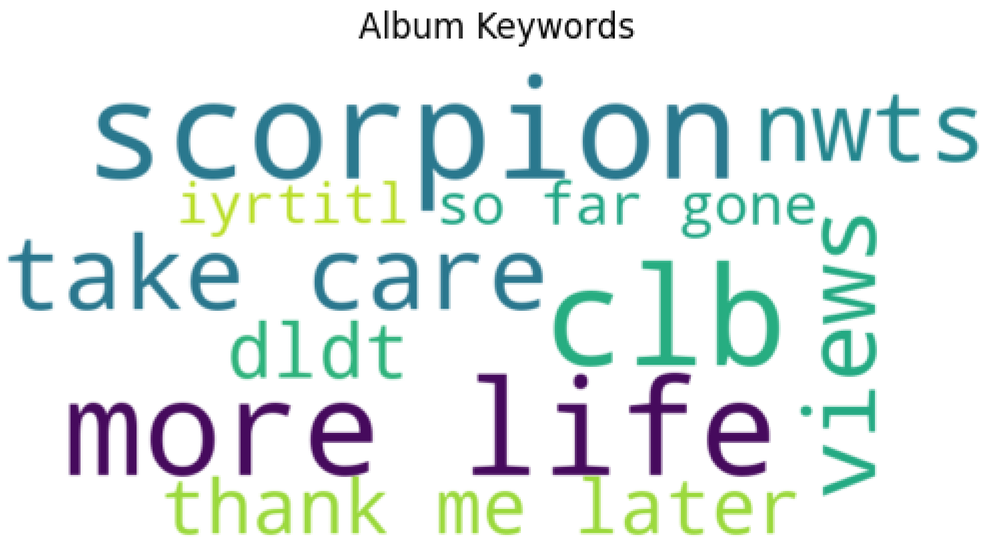
At this point, we've retrieved comments from Reddit posts discussing Drake's discography and identified which of his albums are most frequently mentioned within these comments. Using a list of pre-defined keywords representing the titles of Drake's albums, we iteratively searched through our retrieved comments to create a new list containing all keywords that we've found. This resulting list represents the frequency of the albums that are most commonly discussed in the comments, allowing us to create a bar plot and a WordCloud visualization:

```
In [16]: keywords = ['nwts','take care', 'iyrtitl', 'views', 'clb', 'more life', 'tha
         album_keywords = [] # album keyword list

         c = c1 + c2 # join lists
         for comment in c:
             for keyword in keywords:
                 if keyword in comment.lower():
                     album_keywords.append(keyword)

         keyword_counts = Counter(album_keywords)

         keyword_counts_sorted = dict(sorted(keyword_counts.items(), key=lambda item:

         # Create a bar plot of the frequency distribution with different colors for
         fig, ax = plt.subplots(figsize=(8, 5))
         sns.barplot(x=list(keyword_counts_sorted.keys()), y=list(keyword_counts_sort
         ax.set_title('Frequency distribution of Drake album keywords')
         ax.set_xlabel('Album keywords')
         ax.set_ylabel('Frequency')
         plt.xticks(rotation=45, ha='right')
         plt.show()
```



In [17]:

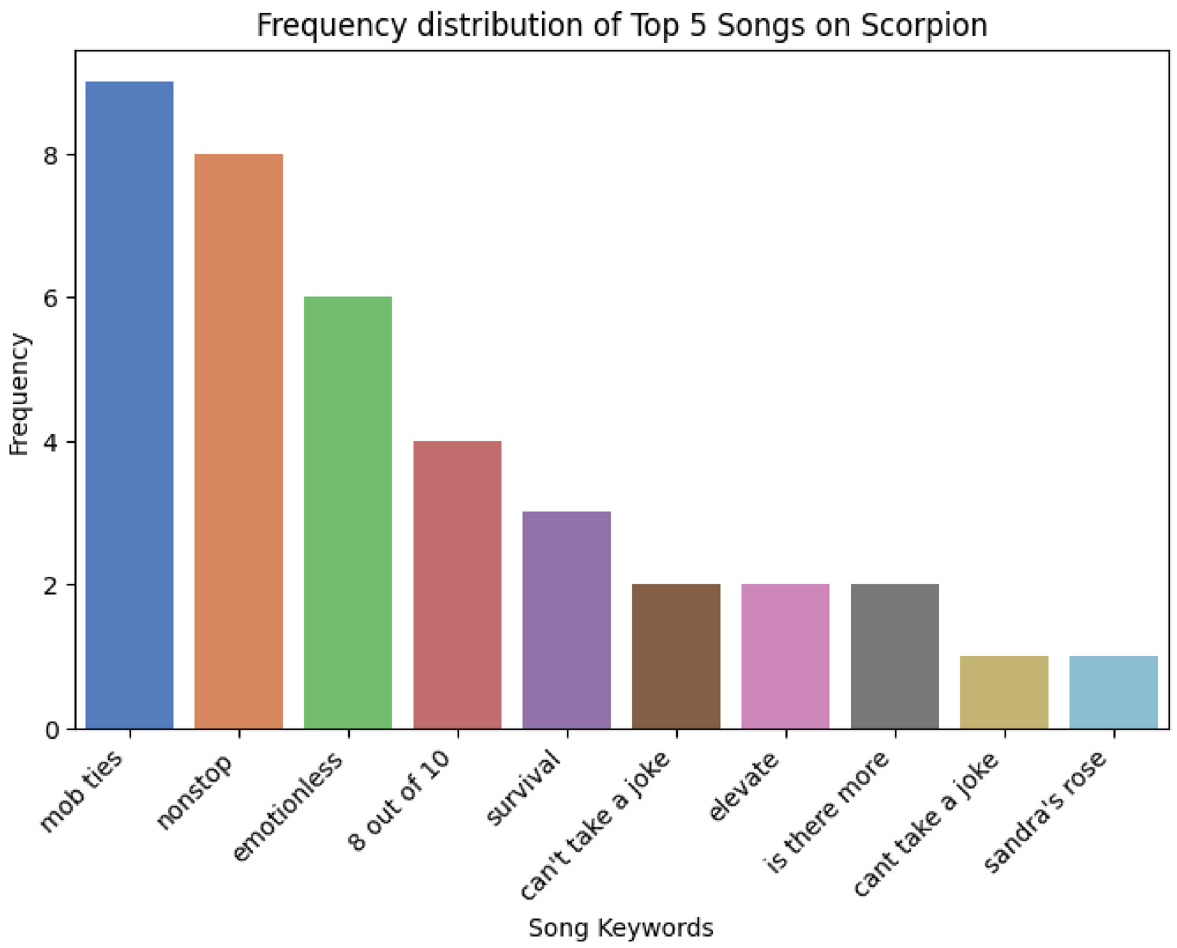
```
# Word Cloud
keyword_counts = Counter(album_keywords)
wordcloud = WordCloud(width=800, height=400, background_color='white').gener

# Display the word cloud
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.title('Album Keywords')
plt.show()
```



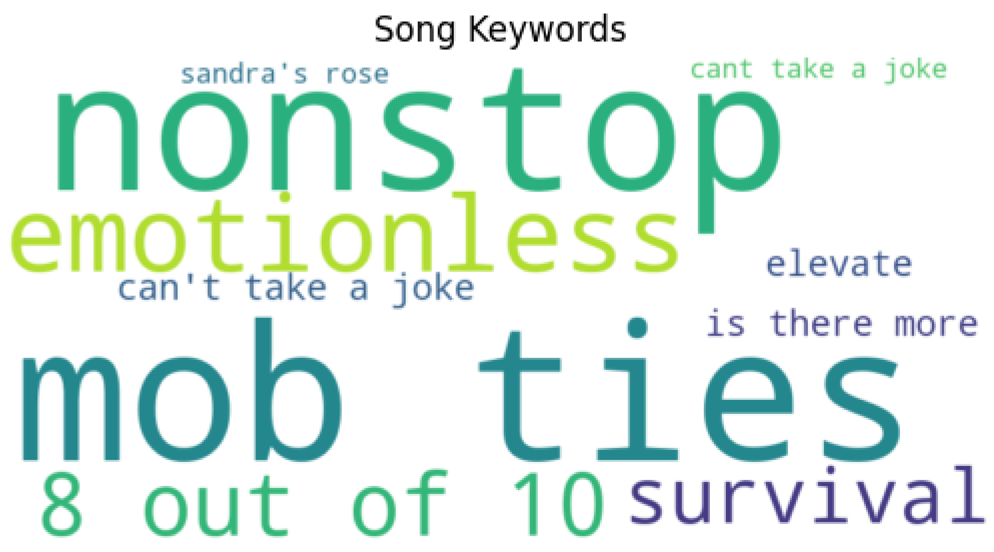
As we can see from the barplot and the WordCloud, his newer albums 'Certified Lover Boy' and 'Scorpion' were most frequently mentioned. Furthermore, we can see that his older albums 'Take Care' and 'Nothing Was the Same' also were mentioned equally as well. With this, the results above suggest that people's prefences regarding Drake's top albums have varied equally between Old and New Drake. Ultimately, this speaks to Drake's consistency in creating impactful music as he has built a unqiue discography that has continually garnered interest throughout his career.

Most Popular Songs on Scorpion

In [18]:
```
c3 = get_reddit_comments(url3, params, headers) # get comments from subreddi

keywords = ['survival', 'nonstop', 'elevate', 'emotionless', "god's plan", '
album_keywords = []

for comment in c3:
    for keyword in keywords:
        if keyword in comment.lower():
            album_keywords.append(keyword)


keyword_counts = Counter(album_keywords)
keyword_counts_sorted = dict(sorted(keyword_counts.items(), key=lambda item:

# Create a bar plot of the frequency distribution with different colors for
fig, ax = plt.subplots(figsize=(8, 5))
sns.barplot(x=list(keyword_counts_sorted.keys()), y=list(keyword_counts_sort
ax.set_title('Frequency distribution of Top 5 Songs on Scorpion')
ax.set_xlabel('Song Keywords')
ax.set_ylabel('Frequency')
plt.xticks(rotation=45, ha='right')
plt.show()
```



In [19]:
```
# Word Cloud
keyword_counts = Counter(album_keywords)
wordcloud = WordCloud(width=800, height=400, background_color='white').gener

# Display the word cloud
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.title('Song Keywords')
plt.show()
```



The bar plot and the WorldCloud above shows that the songs – "Nonstop," "Emotionless", "Mob Ties" were most liked among the *Scorpion* Album. From our k-means clustering analysis, we found that both "Non-stop" and "Emotionless" belong to Cluster 0, while "Mob Ties" belongs to cluster 3. In terms of individual features, all three of these songs exhibit high danceability and energy song attributes. Ultimately, this result reinforces our previous findings that his danceable, energetic music has traditionally been more well-received by the general public.

Overall, our analysis of Reddit posts about Drake was focused on two specific posts that we identified as being particularly relevant to our research questions. Our initial entry point was the Drake subreddit group, from which we began searching for relevant posts. After considering several options, we selected the two posts that we felt were best aligned with our research goals. We made this decision based on several factors, including the number of comments and upvotes on the posts, as well as the degree to which the posts addressed our research questions. By focusing our analysis on these two high-quality sources, we were able to obtain a robust and reliable picture of the opinions and preferences of Drake fans on Reddit.

Our findings indicate that "Nonstop", "Emotionless," and "Mob Ties" are the most liked songs on the Scorpion album, with "Nonstop" and "Emotionless" belonging to Cluster 0 and Mob Ties belonging to Cluster 3. This supports our previous conclusion that Drake's danceable, energetic music is more popular with the general public. Based on the bar plot, albums from *Scorpion* to *Take Care* have the high frequency counts, and our word cloud analysis suggests that people'preferences between old and new Drake varies equally. Ultimately, our findings reinforce that Drake's danceable and energetic music is highly appealing to his fans, and that his albums throughout his career have are extremely well-received.

Part Six: Youtube API

Music is so important as it gives artists a way to express their feelings with millions of fans being able to resonate with them. Drake recently recieved Artist Of The Decade Award and continues to dominate the charts, which is impressive as he has been in the game for more than a decade. Spotify also announced that he was the most streamed artist in the past decade. What is the overall sentiment of Drake's success? To answer this question, we extracted 100 comments through Youtube API's. Extracting all the comments from the youtube videos would take a long time and storage. To limit the irrelevant spam comments we extracted the top 100 comments since youtube's comment algorithm puts the most interacted, recent, and relevant comments at the top.

We used the following videos:

- 'Drake: The Biggest Fraud in Hip Hop': https://www.youtube.com/watch?v=-CXxiFSmBUg&t=3s
- 'DRAKE - ARTIST OF THE DECADE (Acceptance Speech)' https://www.youtube.com/watch?v=-HAuK8Z6q6I&t=3s

We choose to extract comments from "Drake-Artist Of The Decade(Acceptance Speech)" and "Drake: The Biggest Music Fraud". We choose these two videos as one highlights his successful career and the other critizes it.

Drake-Artist Of The Decade (Acceptance Speech)

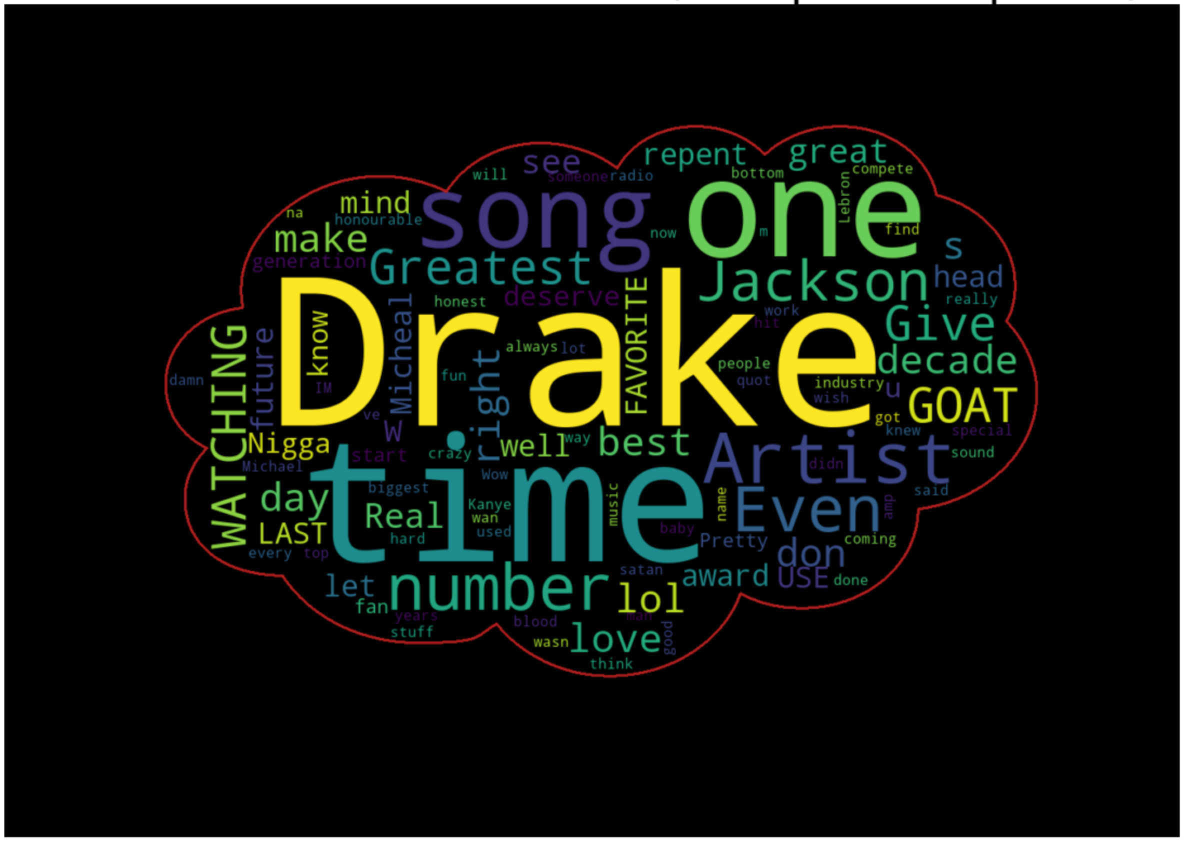
First, we built a resource in order to extract the top 100 comments. Afterwards, we used the built in function in python NLTK (Natural Launge Processing Toolkit) and we were able to get a compound score for each comment. The compound score is the sum of positive, negative, and neutral scores which is then normalized between -1 (most extreme negative) and +1 (most extreme positive). Afterwards we took the average for all the compound scores, but first we wanted to showcase and get a general idea of what words most frequently appeared in the comment section. We decided to lematize the words and run it through stopwords in order to decrease the number of useless words that would appear in the word cloud.

The youtube video, "Drake – Artist Of The Decade (Acceptance Speech)" hightlights his accomplishments and statistics throughout his career. They discuss how he had more weeks on the Hot 100 than Elvis Presley, Michael Jackson, Stevie Wonder, and various other artists. From the word cloud below, we observe a lot of positive comments such as favorite, love, great, and GOAT (greatest of all time) to indicate the support from his fans during his speech. However, we also appear to have words with negative connotations as well, suggesting that some people have criticizes his success as an artist.

In [20]:
```
#from nltk.sentiment.vader import SentimentIntensityAnalyzer
api_key = "AIzaSyCKhyPTPEHgjrobMjxXQOAq-Q-X5YyKRfc"
video_id = "-HAuK8Z6q6I"  #extracting from Drake's Accecptance Speech
```

```
#build a resource for youtube
resource = build('youtube', 'v3', developerKey=api_key)

#create a request to get 100 comments on the video
request = resource. commentThreads().list(
                            part="snippet",
                            videoId=video_id,
                            maxResults= 100,
                            order="orderUnspecified") #top comments

response = request.execute() #execute the request

items = response['items'][:100]
df=pd.json_normalize(items)
#analyzing sentiment analysis on raw data to get a more true score
analyzer = SentimentIntensityAnalyzer()
df['vid1_compound'] = [analyzer.polarity_scores(x)['compound'] for x in df['
vid1compound = df['vid1_compound']
avg_vid1_comp=(df['vid1_compound'].mean())


df=df['snippet.topLevelComment.snippet.textDisplay'] #extracting the comment
df.head()
text = df.to_csv() #converting the data frame to text file, in order to gene
words = tokenize.word_tokenize(text)
words = [w for w in words if w.isalpha()] #assures only words are in the out
stopwords = ["the", "a", "and", "or", "in", 'https', 'trump', 'is', 'you', '
final_words = [w for w in words if w not in stopwords]
fq = nltk.FreqDist (w for w in final_words if w.isalnum())
text = ' '.join(fq) #turning list back into string

# Create a word cloud image
cloud = np.array(Image.open("cloud.png")) #the mask for our cloud
wc = WordCloud(stopwords = STOPWORDS, background_color="black", max_words=10
               contour_width=3, contour_color='firebrick')

# Generate a wordcloud
wc.generate(text)

# show and adjusting sizes for visualization purposes
plt.figure(figsize=[20,10])
plt.imshow(wc, interpolation='bilinear')
plt.title('Drake Artist of The Decade (Acceptance Speech)', fontsize=30)
plt.axis("off")
plt.show()
```

Drake Artist of The Decade (Acceptance Speech)

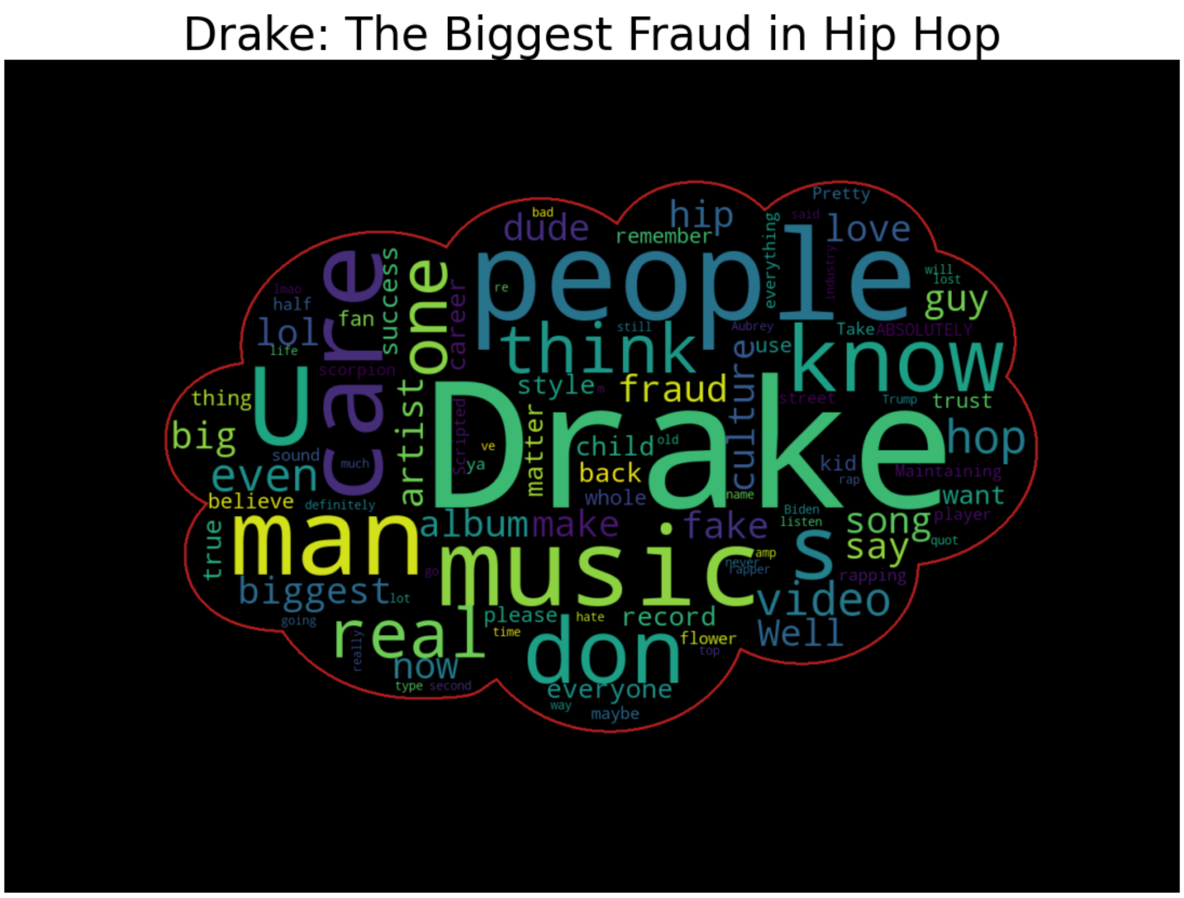


Drake: The Biggest Fraud in Hip Hop

Next, we will analyze the video, "Drake: The Biggest Fraud in Hip Hop" using the same steps described as the first video. The video gives a background of Drake's personal life and the narrator expresses his opinion about Drake's music career. In summarization, he discussed how Drake built his career on writing emotional and introspective music but then become more concerned with staying relevant rather than authentic. In the word cloud above we see a mixture of words such as fraud, fake, love and more – indicating a mixture of both positive and negative connotations.

```
In [21]: video_id = "-CXxiFSmBUg"  #extracting from Drake The Biggest Fraud in Hip Ho
         #build a resource for youtube
         resource = build('youtube', 'v3', developerKey=api_key)

         #create a request to get 100 comments on the video
         request = resource. commentThreads().list(
                                     part="snippet",
                                     videoId=video_id,
                                     maxResults= 100,
                                     order="orderUnspecified") #top comments

         response = request.execute() #execute the request

         items = response['items'][:100]
```

```
df=pd.json_normalize(items)
#analyzing sentiment analysis on raw data to get a more true score
analyzer = SentimentIntensityAnalyzer()
df['vid2_compound'] = [analyzer.polarity_scores(x)['compound'] for x in df['
avg_vid2_comp=(df['vid2_compound'].mean())


df=df['snippet.topLevelComment.snippet.textDisplay'] #extracting the comment
df.head()
text = df.to_csv() #converting the data frame to text file, in order to gene
words = tokenize.word_tokenize(text)
words = [w for w in words if w.isalpha()] #assures only words are in the out
stopwords = ["the", "a", "and", "or", "in", 'https', 'trump', 'is', 'you', '
final_words = [w for w in words if w not in stopwords]
fq = nltk.FreqDist (w for w in final_words if w.isalnum())
text = ' '.join(fq) #turning list back into string

# Create a word cloud image
cloud = np.array(Image.open("cloud.png")) #the mask for our cloud
wc = WordCloud(background_color="black", max_words=100, mask=cloud,
               contour_width=3, contour_color='firebrick')

# Generate a wordcloud
wc.generate(text)

# show and adjusting sizes for visualization purposes
plt.figure(figsize=[20,10])
plt.imshow(wc, interpolation='bilinear')
plt.title('Drake: The Biggest Fraud in Hip Hop', fontsize=30)
plt.axis("off")
plt.show()
```



Sentimental Comparison

Now that we have showed some of the words from the youtube comments, we wanted to take the compound average score for both videos by taking into consideration all the comments. From the bar plot below, it can be seen that the comments from Drake's acceptance speech got an average compound score of 20 and the comments from the second video, Drake: The Biggest Artist Fraud got an average compound score of 13. Actually, it is quite surprising that the second video got a positive compound score; we initially expected it to have a negative score since it mainly criticized Drake as an artist. If there was a negative score, it would entail that many negative words were used, however that wasn't the case. That being said, our results indicate to us that there has been an overall approving sentiment on Drake's rising popularity.

In [22]:
```
print(avg_vid1_comp)
print(avg_vid2_comp)
```

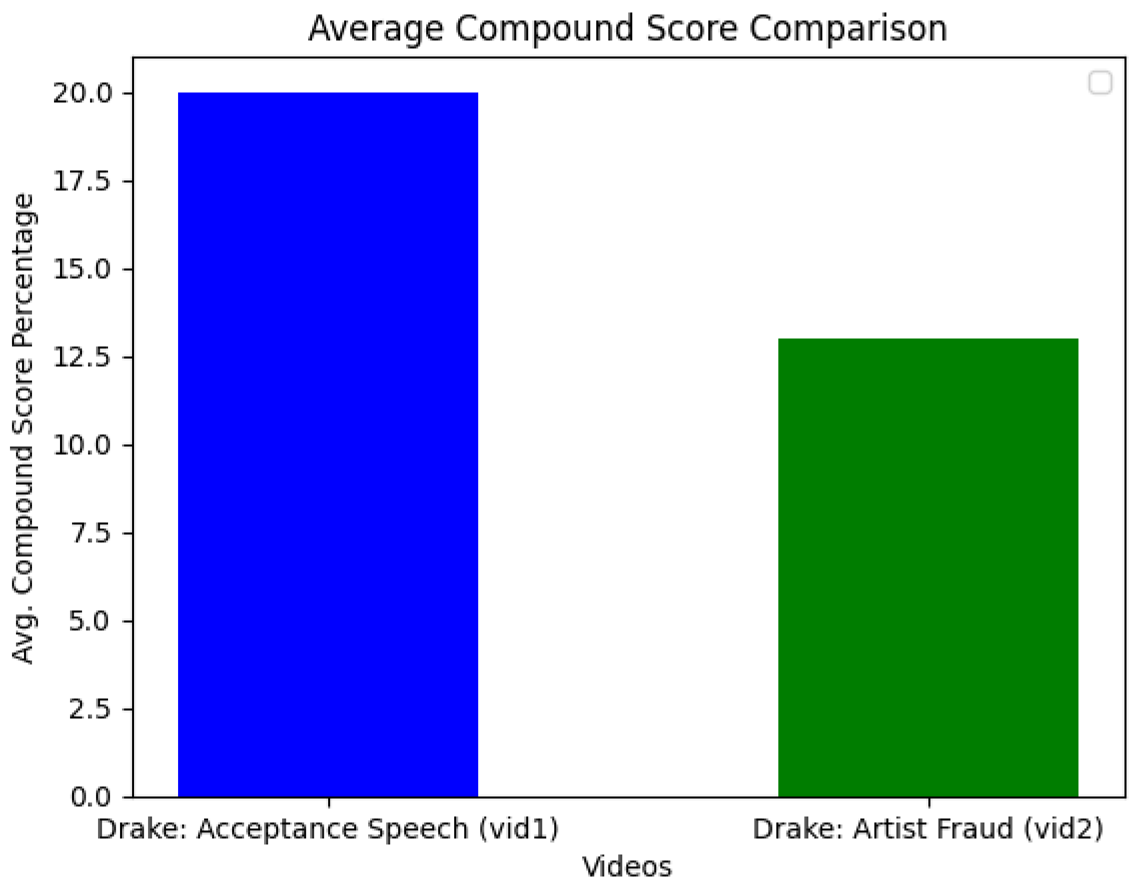
```
0.20373900000000003
0.12718100000000002
```

In [23]:
```
# Creating a plot to compare the compound score
left = [1, 2]
height = [20, 13]
tick_label = ['Drake: Acceptance Speech (vid1)', 'Drake: Artist Fraud (vid2)
```

```
plt.bar(left, height, tick_label = tick_label, width = .50, color = ['blue',
plt.xlabel('Videos')
plt.ylabel('Avg. Compound Score Percentage')
plt.legend()
plt.title('Average Compound Score Comparison')
plt.show
```

```
No artists with labels found to put in legend.  Note that artists whose lab
el start with an underscore are ignored when legend() is called with no arg
ument.
```

Out[23]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```



Conclusion

Our analysis of Drake's music and popularity has shed light on the factors contributing to his longstanding success in the music industry. Through our examination of various data API sources (Spotify, Reddit, and YouTube), we have identified several key elements that have helped Drake establish himself as one of the most influential musicians of our generation. While our analysis is not without limitations, we believe that the insights gained provide a valuable perspective on Drake's impact on the music industry and popular culture.

We visualized Drake's popularity and saw how he has been able to prolong his success. With each new album that Drake releases, there is a steady increase in his popularity. His music has achieved global popularity and the broad appeal of larger audiences, allowing Drake to dominate the music industry and witness prolonged success. We clustered our music by song attributes, finding that features such as danceability, energy, and valence were most pronounced in his most popular songs. Moreover, our analysis displayed that Drake has sustained his popularity and has accumulated a predominantly positive sentiment as an artist which could be due to his devoted and large fan base. Ultimately, addressing the questions at the beginning have helped us gain insight as to Drake's musical career has witnessed such prolonged success.

In terms of the limitations of this project, we didn't analyze Drake's entire discography and YouTube videos/comments due to time constraints. It is also important to note that the number of streams for a song may change daily; therefore, not all of the data presented may be up to date. Also, social media platforms like YouTube and Reddit may not provide reliable generalizations for subgroups, as they are more popular among Generation Z, who may have different music preferences than the general population.

In conclusion, Drake's success and longevity in the music industry are nothing short of impressive. While his music may not appeal to everyone, it's undeniable that he has made a significant impact on the industry and popular culture as a whole. As he continues to push the boundaries and innovate in his craft, it will be exciting to see what the future holds for Drake and his fans.

References

- Spotify API Documentation: https://developer.spotify.com/documentation/web-api/reference/#/
- ChartMaster Top 40 Songs: https://chartmasters.org/spotify-top-album-debuts/?view=artist&artist_name=drake&artist_id=
- ChartMaster Debut Album Sales: https://www.officialcharts.com/chart-news/drakes-official-top-40-most-streamed-songs__21625/
- Understanding K-means Clustering: https://towardsdatascience.com/understanding-k-means-clustering-in-machine-learning-6a6e67336aa1
- K-means clustering on Spotify Example: https://towardsdatascience.com/k-means-clustering-and-pca-to-categorize-music-by-similar-audio-features-df09c93e8b64
- Extracting Data from Spotify API: https://stmorse.github.io/journal/spotify-api.html
- Extracting Data through Reddit API: https://proxiesapi.com/blog/scraping-reddit-with-python-and-beautiful-soup.html.php
- WordCloud Visualization: https://www.analyticsvidhya.com/blog/2021/08/creating-customized-word-cloud-in-python/
- Extracting Data from Youtube API: https://developers.google.com/youtube/v3/docs/commentThreads/list
- Choropleth Map Visualization: https://plotly.com/python/choropleth-maps/

- ISO Country Codes: https://blog.devgenius.io/using-python-pandas-to-turn-iso-country-codes-into-a-string-to-use-as-values-for-a-sql-query-ad598bdd17d1

```
In [ ]:
```