Final Reflection

Tarang Tondapu, Ambika Tripathi, Zahraa Elghoul
https://github.com/ttondapu/206final.git

Goals For Our Project

In our original project plan, we aimed to use public APIs from Spotify, SoundCloud, and Audiomack to compare the followers and streams of top artists on each respective platform. We wanted to determine the size of active followers on each platform, as well as whether the proportions between platforms remained consistent across popular genres. Because each of the platforms is different in their methods of access and uploads, we set out to use our skills in APIs and databases to find out the different characteristics of platforms and genres.

Goals We Achieved

While accessing data and reading API documentation further, we discovered several limitations of our metrics. Several of the APIs did not support endpoints to gather certain information, while workarounds led to time and space complexities beyond what was reasonable. We therefore shifted our focus from comparing three platforms, incorporating the Twitter API in place of audiomack or any third streaming platform. The SoundCloud API support was also not native, so we adapted the code to use BeautifulSoup in order to gather information. In doing so, we were able to find information beyond simply streaming numbers (such as online activity and popularity with respect to both music and social media), changing the scope of the project and ultimate calculations/visualizations. In the end, we successfully calculated information relating to follower counts, number of listings per platform, and average album lengths for each of the top few US artists.

Problems We Faced

The first problem we encountered was accessing the APIs for SoundCloud and Audiomack, which were either unavailable or protected by OAuth. We were unable to generate a valid token, and furthermore, could not find sufficient documentation for accessing our intended endpoints. We also felt that the data would be far too similar to compare anything other than averages or aggregates of identical metrics across platforms, forcing us to adapt. After the major hurdle of redesigning the original project plan, we faced several challenges while collecting data for various tables. SoundCloud's dynamic HTML and inconsistent tags proved to be a challenge, eventually resolved by the removal of the empty class tag in BeautifulSoup. Furthermore, learning to generate a new token for Spotify was made difficult at first due to an inaccurate error message (expired tokens showed up as local key errors).

Calculations

avg_followers.csv

```
artist name,average number of followers across platforms
Ed Sheeran,38676342
The Weeknd,22147918
Billie Eilish,23122535
Justin Bieber,58411999
Taylor Swift,47631849
Drake,35067282
Eminem,26219106
Post Malone,15131690
Kanye West,16370637
Juice WRLD,9376759
```

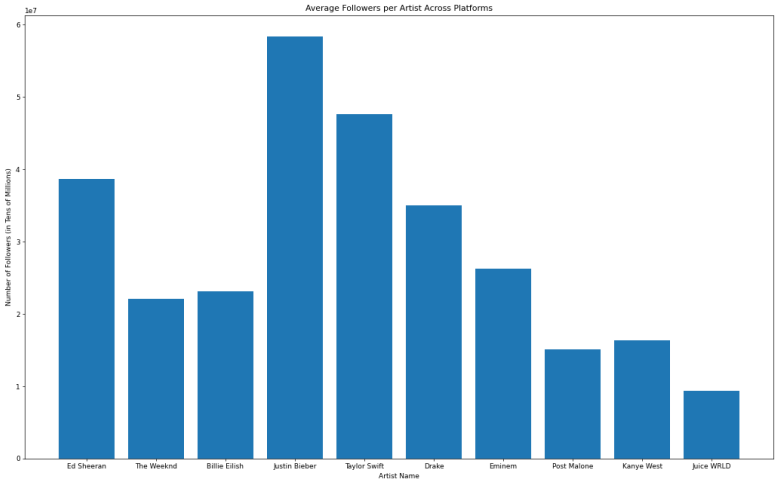
num_tracks_available.csv

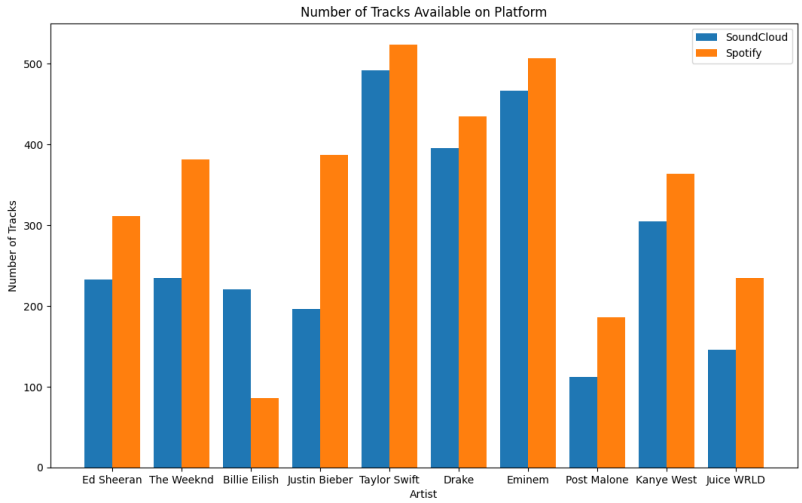
```
artist name,number of soundcloud tracks,number of spotify tracks
Ed Sheeran,233,311
The Weeknd,235,382
Billie Eilish,221,86
Justin Bieber,196,387
Taylor Swift,492,524
Drake,396,435
Eminem,467,507
Post Malone,112,186
Kanye West,305,364
Juice WRLD,146,235
```

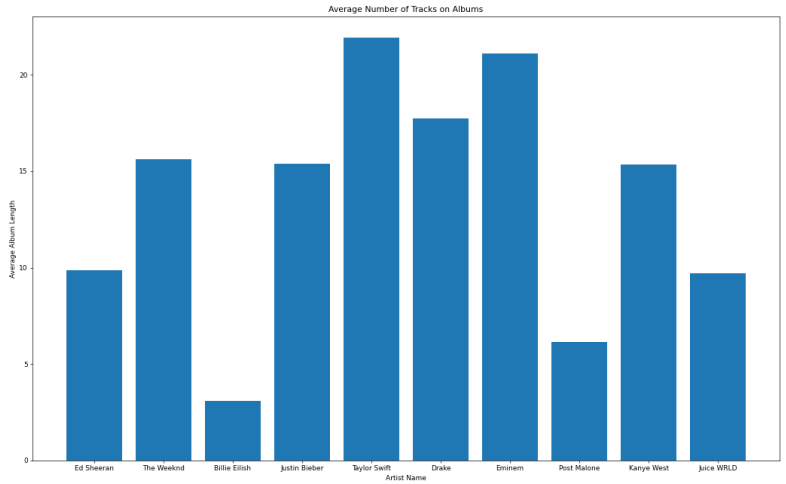
avg_album_length.csv

```
artist name,average number of tracks per album
Ed Sheeran,9.882352941176471
The Weeknd,15.636363636363637
Billie Eilish,3.090909090909091
Justin Bieber,15.384615384615385
Taylor Swift,21.9375
Drake,17.75
Eminem,21.11111111111111
Post Malone,6.157894736842105
Kanye West,15.357142857142858
Juice WRLD,9.692307692307692
```

Visualizations







Instructions for Running Code

spotify.py:

Note Please only run one table creation function in main() at a time. When the number of requests exceeds a certain threshold, the Spotify API will throw a random key error.

1. Generate a new token from the link in main() (https://developer.spotify.com/console/get-album/)
2. Paste this token into the 'token' variable (line 121 as of v1)

   If creating artists table:

   1. Comment out create_spotifyalbums_table with a hashtag at the beginning of the line and only leave create_spotifyartists_table
   2. Run the code and enter an artist name when prompted. The code will create an entry with the artist's information in the finalproj database's spotifysartists table

   If creating albums table:

   1. Comment out create_spotifyartists_table with a hashtag at the beginning of the line and only leave create_spotifyalbums_table
   2. Run code and enter an artist name when prompted. The code will create entries with all of the artist's albums in the finalproj database's spotifyalbums table

twitter.py

1. Simply run the code, and enter a number into the command line. The twitter table in the finalproj database will be populated with 25 artists at a time, and running it 4 times (with inputs 1, 2, 3, 4) should populate the table fully. No duplicates will be created.

soundcloud.py

Note for soundcloud.py, we used beautifulsoup and have pre-loaded htmls for offline access. The site was protected against scraping, meaning requests did not work.

If creating artists table:

1. Comment out setUpSoundcloudTrackTable with a hashtag at the beginning of the line and only leave setUpSoundcloudArtistTable
2. Run the code. The code will prompt you to enter one of the top artists included in the provided dictionary, and will create an entry with the artist's information in the finalproj database's soundcloud_artists table

If creating tracks table:

1. Comment out setUpSoundcloudArtistTable with a hashtag at the beginning of the line and only leave setUpSoundcloudTrackTable
2. Run the code. The code will prompt you to enter one of the top artists included in the provided dictionary, and will create entries with all of the artist's tracks in the finalproj database's soundcloud_tracks table

visuals.py

Run the code, and enter a number 1-3 into the command line as prompted. This will result in the creation of a csv containing the corresponding calculations in the current directory while displaying charts displaying the data.

Documentation

spotify.py

```
def createDB(filename):
   '''
   This function initializes the database (if not already created) to be used throughout this
project.
   filename.db will be found in the current working directory.
   '''
def create_spotifyartists_table(favartists, token, offset, cur, conn):
   '''
   This function takes in a list of artists, a token, an offset, and a database cur/conn.
   It then creates a table inside the database pointed to by cur/conn with an artist's
information
   which includes artist id, name, spotiy followers, and number of tracks available on
spotify.
   '''
def create_spotifyalbums_table(favartists, token, offset, cur, conn):
   '''
   This function takes in a list of artists, a token, an offset, and a database cur/conn.
   It then creates a table inside the database pointed to by cur/conn with each artist's
discography information
   which includes artist id, album name, spotify, length of project, and release date.
   '''
def artistalbumsurl(artistid):
   '''
   Given an artist ID, this function returns an API url to access all of their albums.
   '''
def albumurl(albumid):
   '''
   Given an album id, this function returns an API url to access a specific album.
   '''
def artisturl(artistid):
   '''
   Given an album id, this function returns an API url to access a specific artist.
   '''
def spot_data_two(artistid, token, offset):
   '''
   This function takes in an artist id and generates a list of tuples which will be
   used to populate the albums table with entries about each of the artist's releases.
   '''
def spot_data_one(artistid, token, offset):
   '''
   This function takes in an artist id and generates tuple about the artist's
   name, number of spotify followers, and total tracks available on spotify.
   '''
```

soundcloud.py

```
def createDB(filename):
   '''
   This function initializes the database (if not already created) to be used throughout this
project.
   filename.db will be found in the current working directory.
   '''
def setUpSoundcloudArtistTable(favartists, cur, conn):
   '''
   This function takes in a list of artists, a token, an offset, and a database cur/conn.
   It then creates a table inside the database pointed to by cur/conn with an artist's
information
   which includes artist id, name, soundcloud followers (and formerly number of tracks
available on soundcloud).
   '''
def setUpSoundcloudTrackTable(favartists, cur, conn):
   '''
   This function takes in a list of artists, a token, an offset, and a database cur/conn.
   It then creates a table inside the database pointed to by cur/conn with each track for each
artist,
   in the form of artist id, track name.
   '''
def get_url(artist_user):
   '''
   This function generates a url to access the html of an artist's tracks
   '''
def artist_followers(artist_html):
   '''
   This function returns the number of followers for a given artist from an artist's
SoundCloud page html.
   '''
def track_count(artist_html):
   '''
   This function returns the number of tracks on soundcloud for a given artist from an
artist's SoundCloud page html.
   '''
def all_tracks(artist_html):
   '''
   This function takes an artist's soundcloud html and returns a list of track titles for all
tracks available
   '''
```

twitter.py

```
def createDB(filename):
   '''
   This function initializes the database (if not already created) to be used
throughout this project.
   filename.db will be found in the current working directory.
   '''

def setUpTwitterTable(favartists, cur, conn):
   '''
   This function takes in a list of artists and a cur/conn pointing to a database.
   When run, it will populate the twitter table inside the database, inserting an
entry for each
   artist including information about their username, number of followers, and number
of tweets.
   '''

def create_url(user_id):
   '''
   This function takes in a twitter user id and returns a url to access their account
from the API.
   '''

def bearer_oauth(r):
   '''
   DO NOT EDIT: This function is from the twitter dev site to authorize our program.
   It utilizes the bearer token found at the top.
   '''

def get_artist_info(url):
   '''
   This function takes a twitter api url and returns the json response object.
   '''

def main(dbfilename):
   '''
   main() prompts the user for a number 1-4 and will populate the twitter table in the
database passed in.
   to fully populate, run 4 times and each time type in 1/2/3/4 in that order to fill
all 100 entries.
   '''
```

visuals.py

```
def graph_avg_followers(db_filename):
    '''
    Given the database, this function outputs a csv file to the current directory and
    displays a bar chart visualizing each artist's average follower count across
    twitter, spotify, and soundcloud.
    '''

def graph_num_tracks(db_filename):
    '''
    Given the database, this function outputs a csv file to the current directory and
    displays a bar chart visualizing the number of tracks available on soundcloud
    and spotify respectively for each artist.
    '''

def graph_avg_album_length(db_filename):
    '''
    Given the database, this function outputs a csv file to the current directory and
    displays a bar chart of each artist's average album length based on all of their
releases.
    '''

def write_to_csv(filename, headerlist, datalists):
    '''
    Given a desired output file name, a list of headers, and a list of data lists,
    this function writes a csv file where each entry contains values from the lists.
    This is used as a helper function inside the visualization/generation functions.
    '''

def main(db_filename):
    '''
    Main will prompt for a number representing the three visualization options.
    It will create/connect to the database passed in to access the five tables.
    Upon entering a number, a csv file will be created and a graphic will pop up.
    '''
```

Resources Used

Date	Issue Description	Location of Resource	Result
4/14	Needed access token for Spotify	https://developer.spotify.com/console/get-album/	Generated access token after logging in
4/14	Needed to know how to pull artist and album data from Spotify	https://developer.spotify.com/documentation/web-api/reference/#/	Accessed endpoints of different resource types successfully with url generator helper function
4/18	Needed to find twitter ids from username	https://tweeterid.com/	Created dictionary to pass into twitter.py and access each artist's info with API
4/25	Ran into trouble with accessing SoundCloud HTMLS with bs4	Meeting with Amanda	Got rid of class = ‘’, span successfully identified the data for all artists
4/25	Needed to know how to create and label bar chart	https://matplotlib.org/	Successfully created visualizations for our data
4/25	Needed to plot multiple bar charts in the same figure for the same artists	https://www.geeksforgeeks.org/plotting-multiple-bar-charts-using-matplotlib-in-python/	Successfully generated side-by-side bars for number of tracks graph