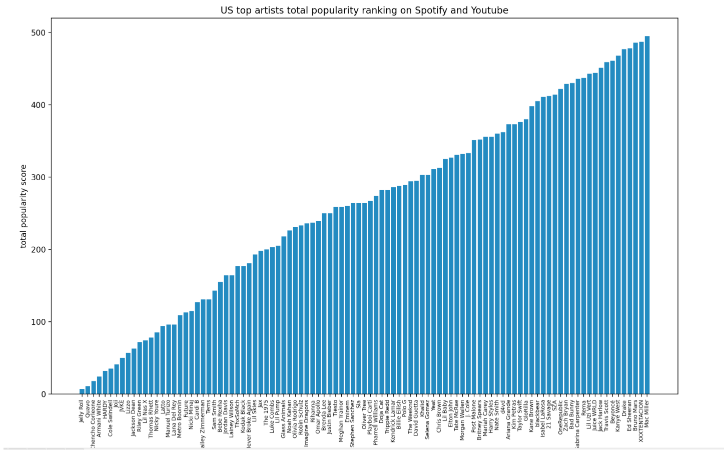
https://github.com/kakazzang/SI206_Final.git

1. The goals for your project (10 points)
    a. Our original goal was to outsource data from SoundCloud and youtube APIs and compare something along the lines of artist follower count/popularity from their first month on SoundCloud in comparison to their most recent months on youtube. The visualization would have included popularity per month by platform.
2. The goals that were achieved (10 points)
    a. We shifted to using a Spotify API. Instead of trying to see the degree to which artists 'started from the bottom' on SoundCloud vs their current youtube markings, we calculated youtube total views, subscriber rank, and average view rank and Spotify followers, popularity, and genre, making charts for all. We also made a total rank chart that shows US top artists' total popularity ranking on Spotify and Youtube.
3. The problems that you faced (10 points)
    a. Because there was a high volume of SoundCloud API requestors, the company postponed the outsourcing of their data. As a result, as mentioned above, we changed our second platform to Spotify.
4. Your file that contains the calculations from the data in the database (10 points)

https://docs.google.com/spreadsheets/d/1FcDdHPSDv6RnvIf7sicHyE9QsGuemTwTZsNMi61ZfZA/edit?usp=sharing

5. The visualization that you created (i.e. screenshot or image file) (10 points)

Just export the data as one tuple per line



6. Instructions for running your code (10 points)

Functions 1-3: Serve as retrievers of spotify and youtube data (API).
Functions 4-7: Serve as creators of the database and compilers of the retrieved musical data into the database.
Functions 8-13: Serve as the functions that perform the desired calculations to find the information we seek. We pull from data in our formerly created database.
Functions 14-21: Serve as creators of the visualization we created. Even though we only called certain functions, we created charts for all to further understand data.
Functions 21-24: Serve as the creators of the csv files containing the data from our calculations.

7. Documentation for each function that you wrote. This includes the input and output for each function (20 points)

Function 1: def get_spotify_data(list), this function gets spotify artist data based on unique ids. It takes a list of ids and returns the requested information from respective ids in a dictionary.

Function 2: def get_youtube_data(list), creates the youtube api request link. This function takes in a string id and returns a url of the youtube api request link.

Function 3: def get_youtube data(list) accesses the youtube channel api data with the specified id taking a list of ids as input and returning a list of dictionaries of the respective data from id token.

Function 4: def open_database(db_name), creates our database. This function takes a string database name as an input and returns the database desired.

Function 5: def add_name_data(artist_list, youtube_id_list, spotify_id_list, cur, con), takes a list of artist, a list of youtube ids, a list of spotify ids, as well as the cur conn of a database. Its return value is present within a new table inside our database: names.

Function 6: def add_spotify_data(data, cur, conn), takes a list of data, as well as the cur conn of a database. Its return value is present within a second table inside our database: spotify.

Function 7: def add_youtube_data(cur,conn), takes a list of data as well as  the cur conn of our database input. It returns a third table inside our database as output: youtube.

Function 8: def youtube_total_views_rank(cur, conn), takes the cur conn of our database as input and joins the view count from the youtube table and id and names from the names table returning a list of data in youtube_subscriber_rank.

Function 9: def youtube_subsribers_rank(cur, conn), takes the cur conn of our database as input and joins the subscriber count from the youtube table and id and names from the names table returning a list of data in youtube_subscriber_rank.

Function 10: def youtube_ave_views_rank(cur, conn), takes the cur conn of our database as input and joins the view and video count from the youtube table finding the average between the two and id and names from the names table returning a list of data in youtube_ave_views_rank.

Function 11: def spotify_followers_rank(cur, conn), takes the cur conn of our database as input and joins the followers from the spotify table and id and names from the names table returning a list of data in spotify_followers_rank.

Function 12: def spotify_popularity_rank(cur, conn): def spotify_followers_rank(cur, conn), takes the cur conn of our database as input and joins the popularity from the spotify table and id and names from the names table returning a list of data in potify_popularity_rank.

Function 13: def spotify_genres_followers_rank(cur, conn), takes the cur conn of our database as input and joins the followers and genres from the spotify table and id and names from the names table returning a list of data in genres_followers_rank.

Function 14: def youtube_total_views_rank_chart(data), takes a list of data as input, youtube_total_views_rank, and returns a visualization of the youtube total views data.

Function 15: def youtube_subscribers_rank_chart(data), takes a list of data as input, youtube_subscribers_views_rank, and returns a visualization of the youtube subscriber views data.

Function 16: def youtube_ave_views_rank_chart(data), takes a list of data as input, youtube_ave_views_rank, and returns a visualization of the youtube average views data.

Function 17: def spotify_followers_rank_chart(data), takes a list of data as input, spotify_followers_rank, and returns a visualization of the spotify followers data.

Function 18: def spotify_popularity_rank_chart(data), takes a list of data as input, spotify_popularity_rank, and returns a visualization of the spotify popularity data.

Function 19: def spotify_genres_followers_rank_chart(data), takes a list of data as input, spotify_genres_followers_rank, and returns a visualization of the genre follower data.

Function 20: def count_total-scores(data1, data2, data3, data4, data5), takes 5 list of data as input: youtube_total_views_rank, youtube_subsribers_rank, spotify_followers_rank, spotify_popularity_rank, and spotify_genres_followers_rank. It returns a singular list of total scores with all data complied.

Function 21: def total_rank_chart(data), takes a list of data as input, total_score_rank, and returns a visualization of the total score data.

Function 22: def write_csv_aveview(data, filename) takes the list of data in youtube_average_views_rank as input and returns a csv file of spotify genres

Function 23: def write_csv_genre(data, filename) takes the list of data in spotify_genres_followers_rank as input and returns a csv file of spotify genres

Function 24: def write_csv_total(data, filename), takes the list of data in total_score_rank as input and returns a csv file of total scores

8. You must also clearly document all resources you used. The documentation should be of the following form (20 points)

| **Date** | **Issue description** | **Location of resource** | **Result** |
|---|---|---|---|

| | | | |
|---|---|---|---|
| 12/05/2022 | Couldn’t get access to Spotify API | https://developer.spotify.com/documentation/general/guides/authorization/ | Resolved, had a function to request for a new token each time running the file and access each artist’s API with unique id |
| 12/05/2022 | Had trouble with accessing Youtube API | https://www.youtube.com/watch?v=MiEpJRoITsI | Resolved, had a function to access each channel’s API unique id |
| 12/05/2022 | Had trouble with limiting the number of data entering database each time | Office hour | resolved |
| 12/06/2022 | Had trouble extracting the information in data | Lecture | Resolved, it is due to some empty list in the data |