Group member: Isabella Benjey, Joey Quick, Elaine Wu
Github link repo: https://github.com/ibenjey/Anime_Aggrigators.git

1: goals for the project

The goal for our project is to find correlation between lists of anime, anime quotes, and studio Ghibli. We worked on three different APIs under the anime topic. Joey did reddit anime and he is grabbing 100 posts from r/anime, Isabella did work on Spotify API by grabbing the studio ghibli track music, and Elaine worked on anime list API by grabbing the first 100 ranked animes from the ranking table. We are hoping to grab data from the three API and compare modern trends in anime with historical ones through the scope of Studio Ghibli. We chose Studio Ghibli as a point of comparison because in the United States it is a fairly popular anime company among a wide variety of demographics and we feel that older populations are more acclimated to anime made by Studio Ghibli compared to modern anime. One way we hope to contrast this is by comparing how often headlines from r/anime on reddit talk about studio ghibli vs. other animes.

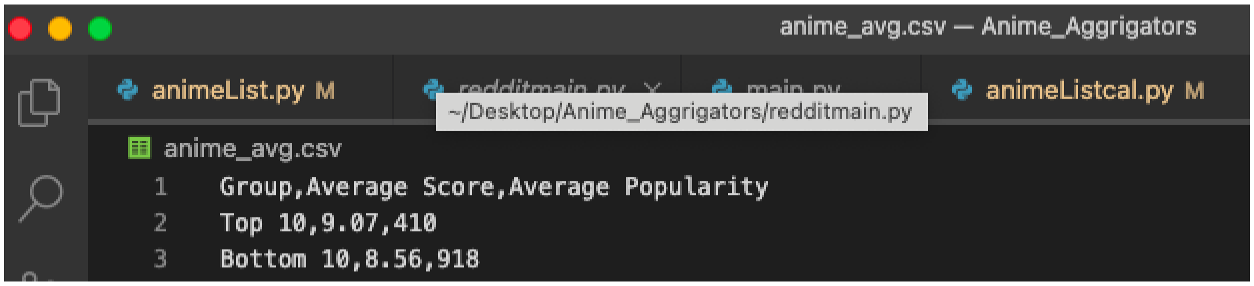
2. Goals that were achieved

Each of us successfully grabbed hundreds of data points and stored them into the database. From there we made average point calculations and tried to compare if there are correlations among these data. Unfortunately trying to compare data from different anime or even different production studios can prove to be tricky. Finding direct correlations between numbers was a challenge, however analyzing the data from a cultural perspective did bring anecdotes, information, and further questions that we did not have before. One of those questions being “is Studio Ghibli’s popularity waning?”, as it seems the popularity of those anime productions is cult in nature. One of the main goals we hoped to achieve was a comparison between modern anime and Studio Ghibli, but found that there is a lot more discussion around modern anime in many cases. Another goal we achieved was an analysis of the popularity of the soundtracks of Studio Ghibli. The scatter plot shows that there may be a trend in popularity leaning to the original ghibli soundtracks compared to the newer albums created by other artists.

3. Problems that we faced

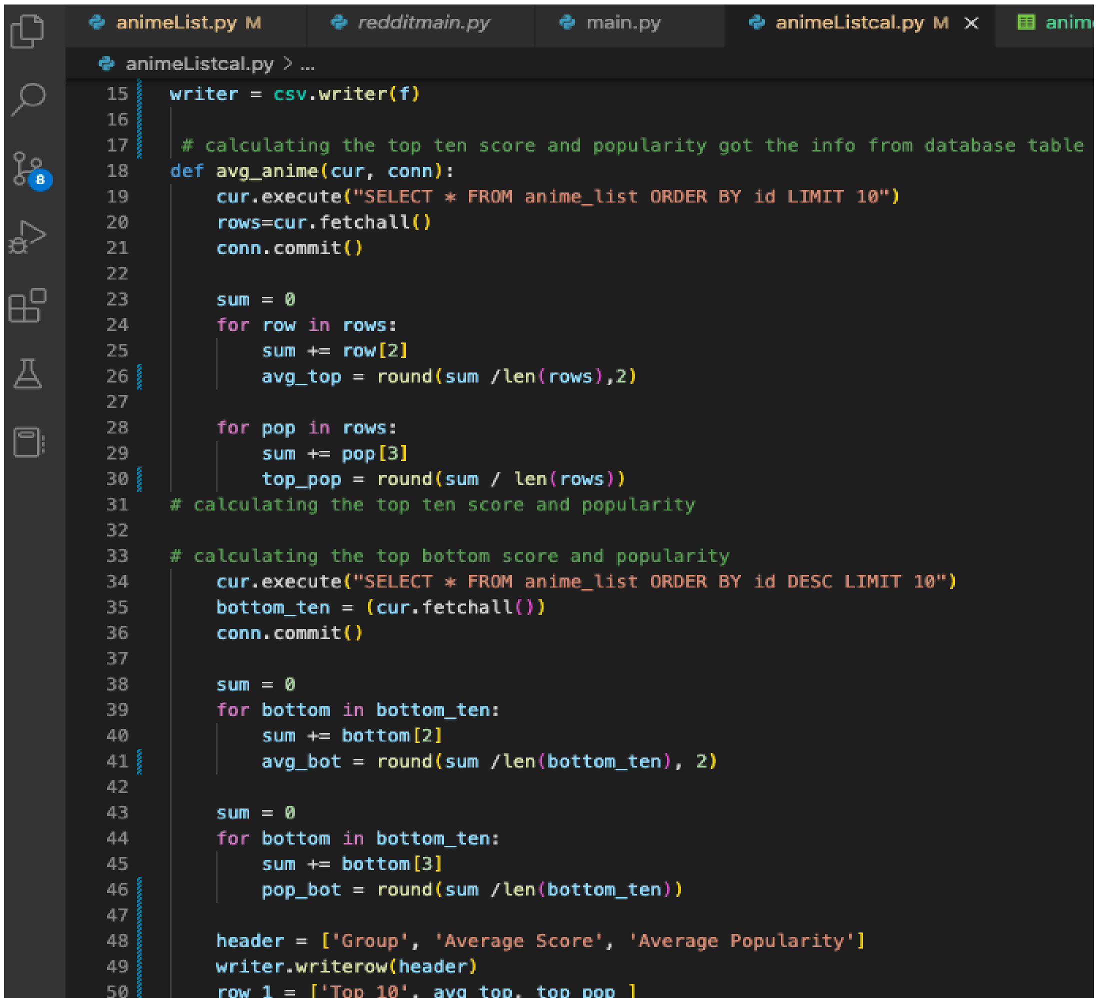
First, the original plan was diverted due to API failure. Our original plan was to grab anime facts, studio ghibli movies, and lists of different anime information. Yet, all of our API at some point fails to allow us to access the data points. We have no choice but to change our plan and select new APIs. Another problem would be getting access into the API and store to the database with limitations. It took our group a long time to find the right path and access the data straight from the API. Each of us individually dealt with issues in github code, console errors, etc. and spent a lot of time learning how to troubleshoot the issues. Github proved to give us more problems than we had anticipated when it came to merging files and following the procedure of group work on a github repository, it was a valuable but frustrating learning experience.

4. Files that contain calculation from the data
Elaine: screenshot of calculation file


anime_avg.csv — Anime_Aggrigators
animeList.py M
redditmain.py
main.py
animeListcal.py M
~/Desktop/Anime_Aggrigators/redditmain.py
anime_avg.csv

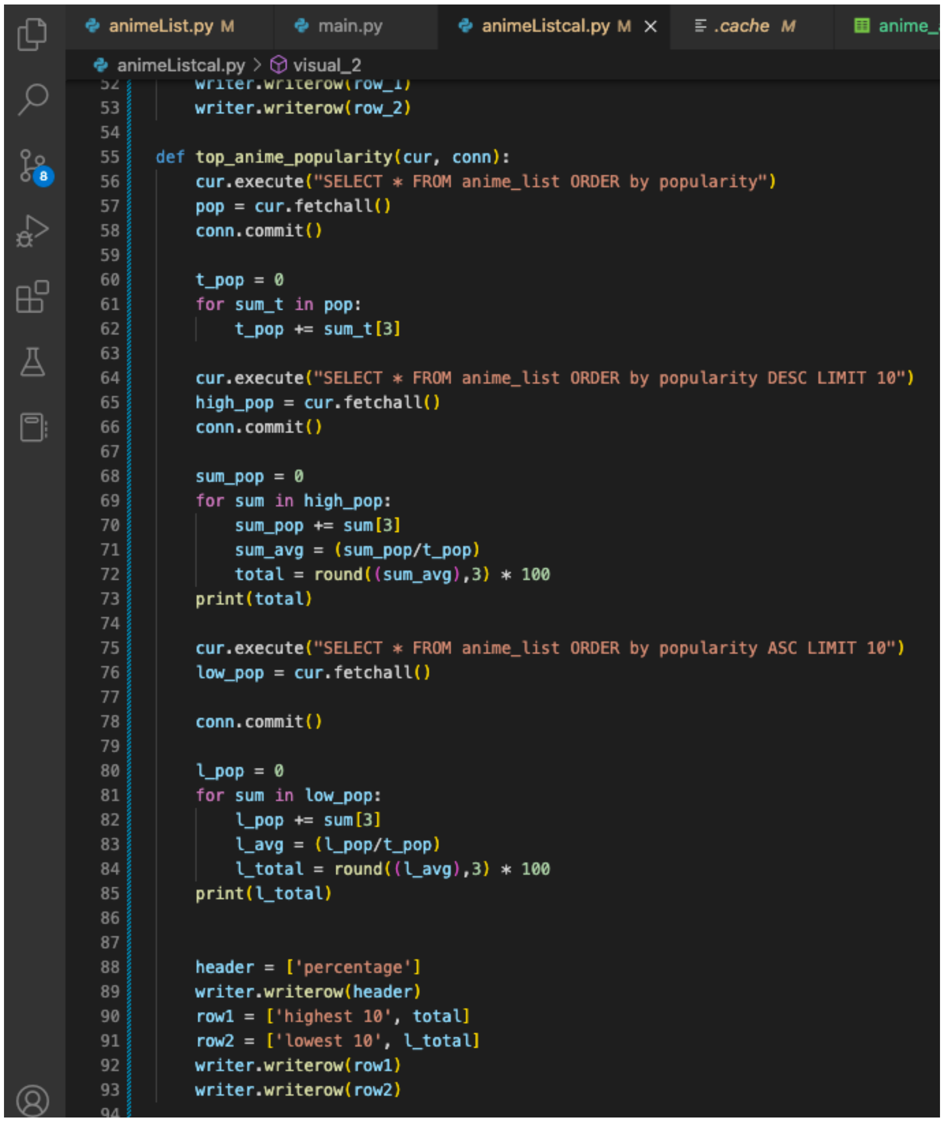
```
Group,Average Score,Average Popularity
Top 10,9.07,410
Bottom 10,8.56,918
```

Elaine: the functions that runs the calculation

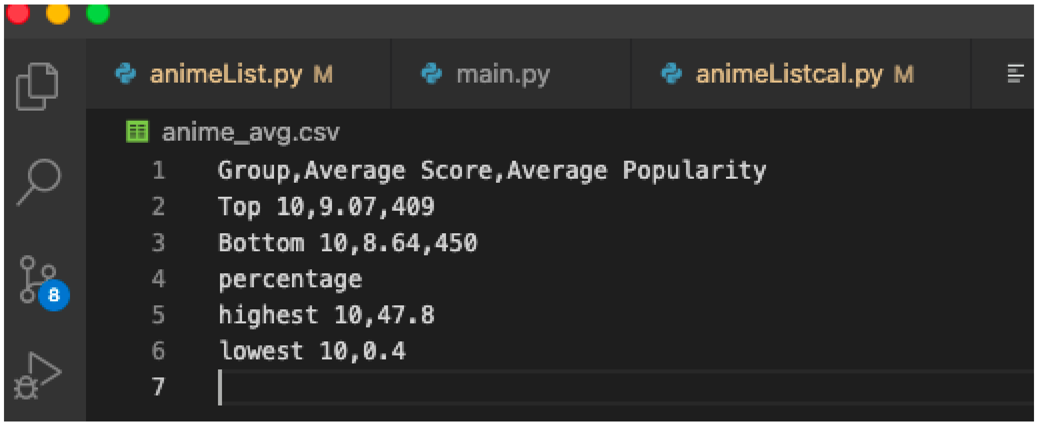


```
writer = csv.writer(f)

 # calculating the top ten score and popularity got the info from database table
def avg_anime(cur, conn):
    cur.execute("SELECT * FROM anime_list ORDER BY id LIMIT 10")
    rows=cur.fetchall()
    conn.commit()

    sum = 0
    for row in rows:
        sum += row[2]
        avg_top = round(sum /len(rows),2)

    for pop in rows:
        sum += pop[3]
        top_pop = round(sum / len(rows))
# calculating the top ten score and popularity

# calculating the top bottom score and popularity
    cur.execute("SELECT * FROM anime_list ORDER BY id DESC LIMIT 10")
    bottom_ten = (cur.fetchall())
    conn.commit()

    sum = 0
    for bottom in bottom_ten:
        sum += bottom[2]
        avg_bot = round(sum /len(bottom_ten), 2)

    sum = 0
    for bottom in bottom_ten:
        sum += bottom[3]
        pop_bot = round(sum /len(bottom_ten))

    header = ['Group', 'Average Score', 'Average Popularity']
    writer.writerow(header)
    row_1 = ['Top 10', avg_top, top_pop ]
```

Additional calculations: Elaine

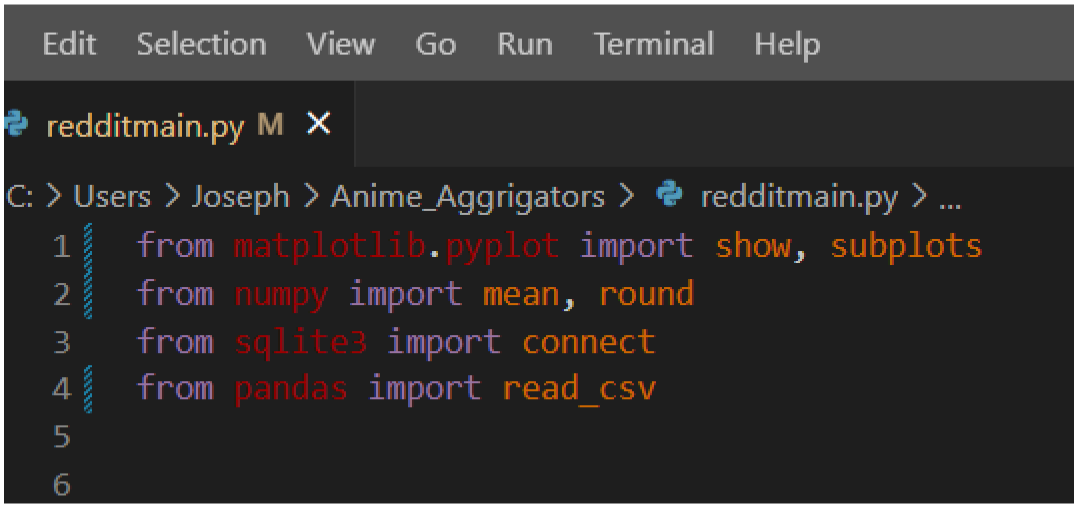


```python
    writer.writerow(row_1)
    writer.writerow(row_2)

def top_anime_popularity(cur, conn):
    cur.execute("SELECT * FROM anime_list ORDER by popularity")
    pop = cur.fetchall()
    conn.commit()

    t_pop = 0
    for sum_t in pop:
        t_pop += sum_t[3]

    cur.execute("SELECT * FROM anime_list ORDER by popularity DESC LIMIT 10")
    high_pop = cur.fetchall()
    conn.commit()

    sum_pop = 0
    for sum in high_pop:
        sum_pop += sum[3]
        sum_avg = (sum_pop/t_pop)
        total = round((sum_avg),3) * 100
    print(total)

    cur.execute("SELECT * FROM anime_list ORDER by popularity ASC LIMIT 10")
    low_pop = cur.fetchall()

    conn.commit()

    l_pop = 0
    for sum in low_pop:
        l_pop += sum[3]
        l_avg = (l_pop/t_pop)
        l_total = round((l_avg),3) * 100
    print(l_total)


    header = ['percentage']
    writer.writerow(header)
    row1 = ['highest 10', total]
    row2 = ['lowest 10', l_total]
    writer.writerow(row1)
    writer.writerow(row2)
```

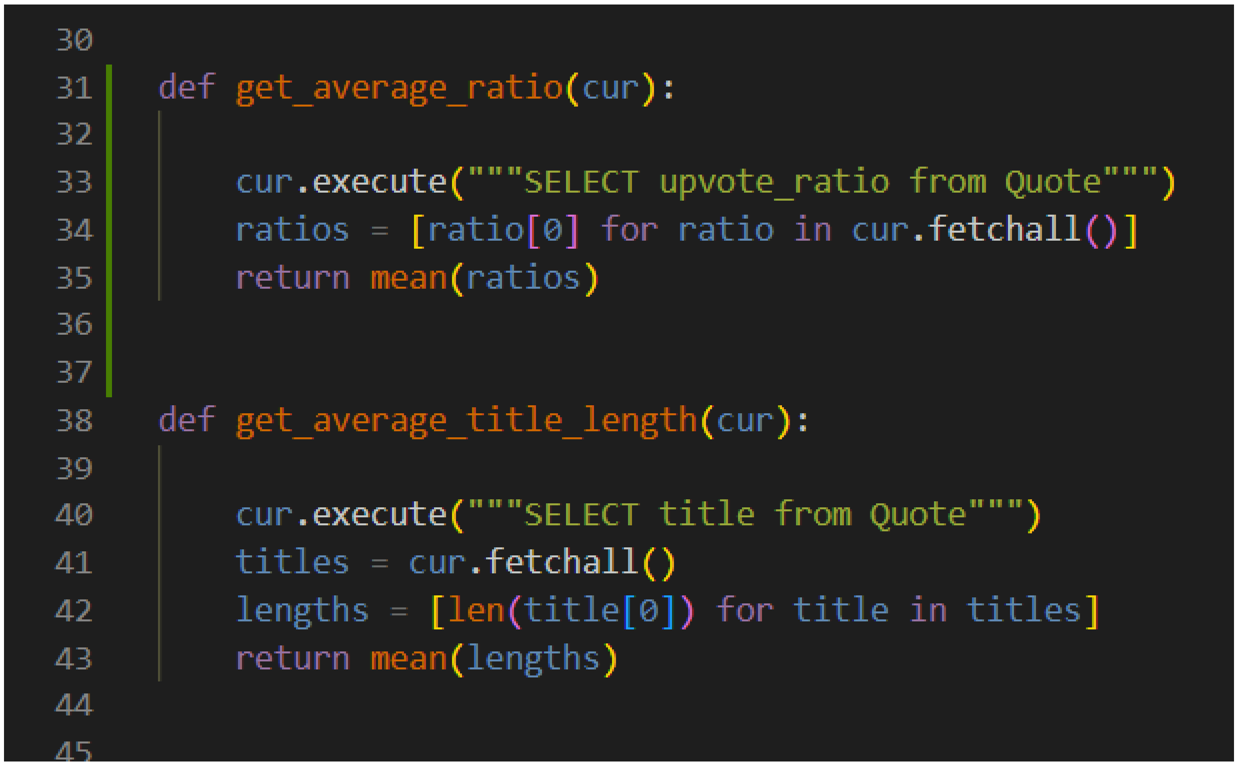


```
Group,Average Score,Average Popularity
Top 10,9.07,409
Bottom 10,8.64,450
percentage
highest 10,47.8
lowest 10,0.4
```

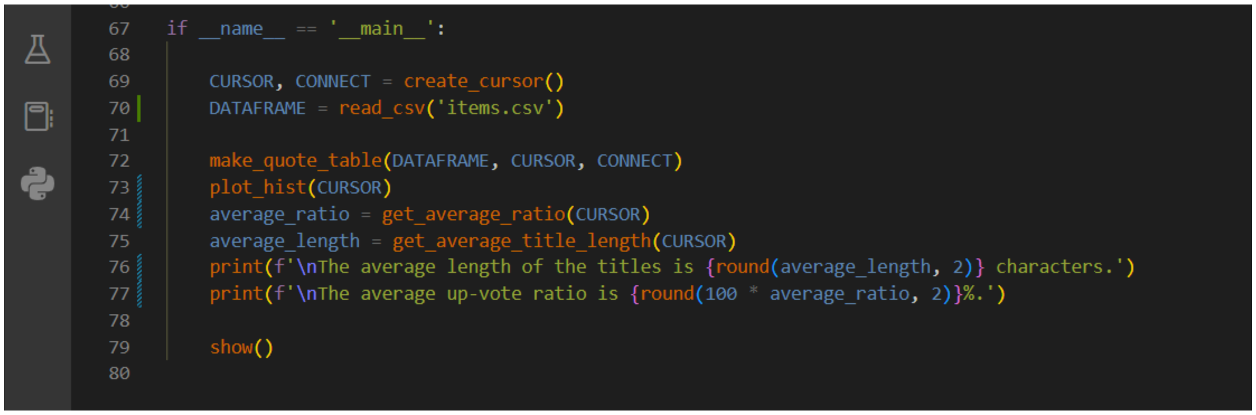
Joey: Calculation functions and output


Edit Selection View Go Run Terminal Help
redditmain.py M ×
C: > Users > Joseph > Anime_Aggrigators > redditmain.py > ...

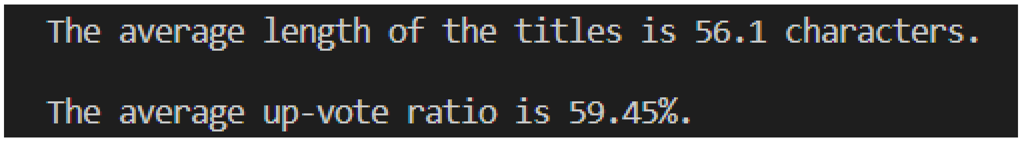
```python
from matplotlib.pyplot import show, subplots
from numpy import mean, round
from sqlite3 import connect
from pandas import read_csv


```



```python

def get_average_ratio(cur):

    cur.execute("""SELECT upvote_ratio from Quote""")
    ratios = [ratio[0] for ratio in cur.fetchall()]
    return mean(ratios)


def get_average_title_length(cur):

    cur.execute("""SELECT title from Quote""")
    titles = cur.fetchall()
    lengths = [len(title[0]) for title in titles]
    return mean(lengths)


```

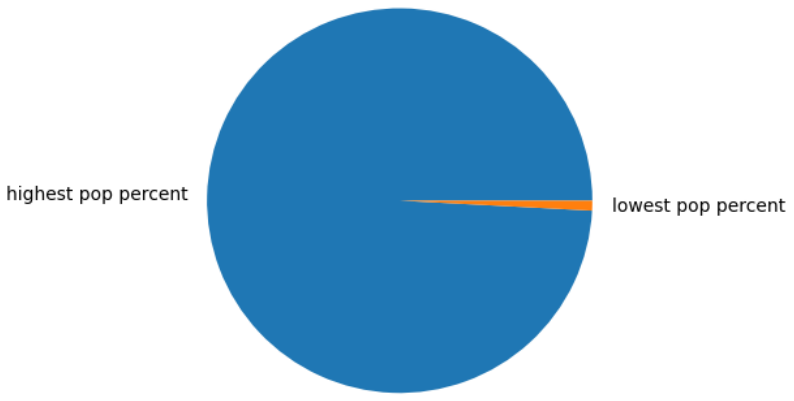


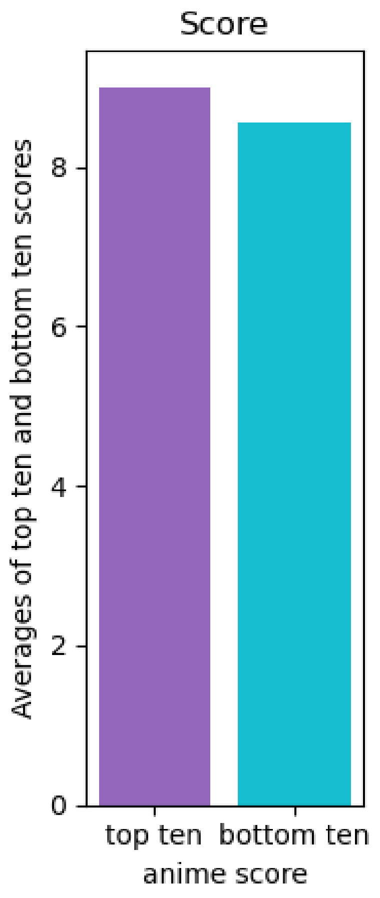
```
if __name__ == '__main__':

    CURSOR, CONNECT = create_cursor()
    DATAFRAME = read_csv('items.csv')

    make_quote_table(DATAFRAME, CURSOR, CONNECT)
    plot_hist(CURSOR)
    average_ratio = get_average_ratio(CURSOR)
    average_length = get_average_title_length(CURSOR)
    print(f'\nThe average length of the titles is {round(average_length, 2)} characters.')
    print(f'\nThe average up-vote ratio is {round(100 * average_ratio, 2)}%.')

    show()

```

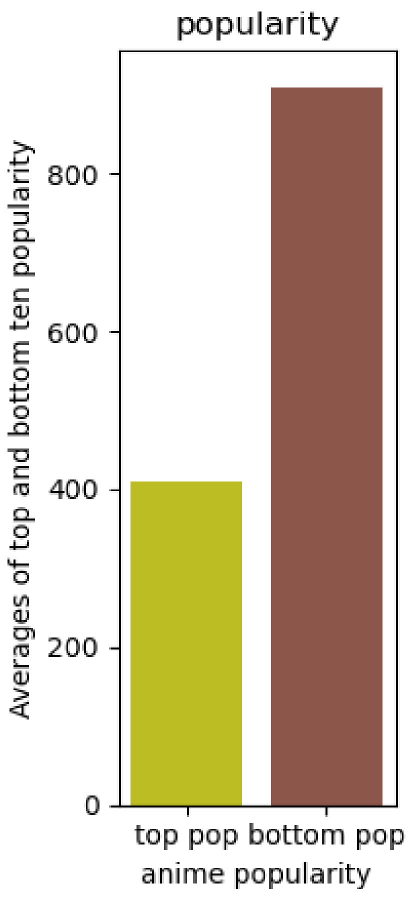


```
The average length of the titles is 56.1 characters.

The average up-vote ratio is 59.45%.
```

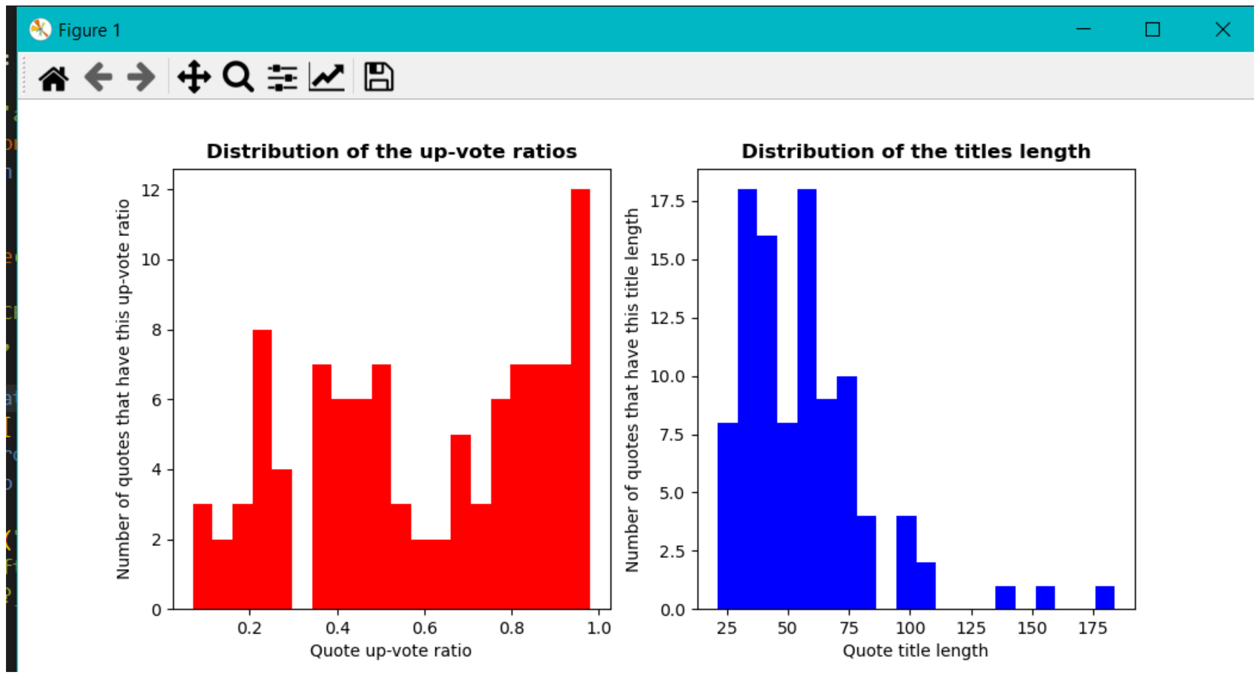
5. Visualization created
Elaine: Anime List visualization



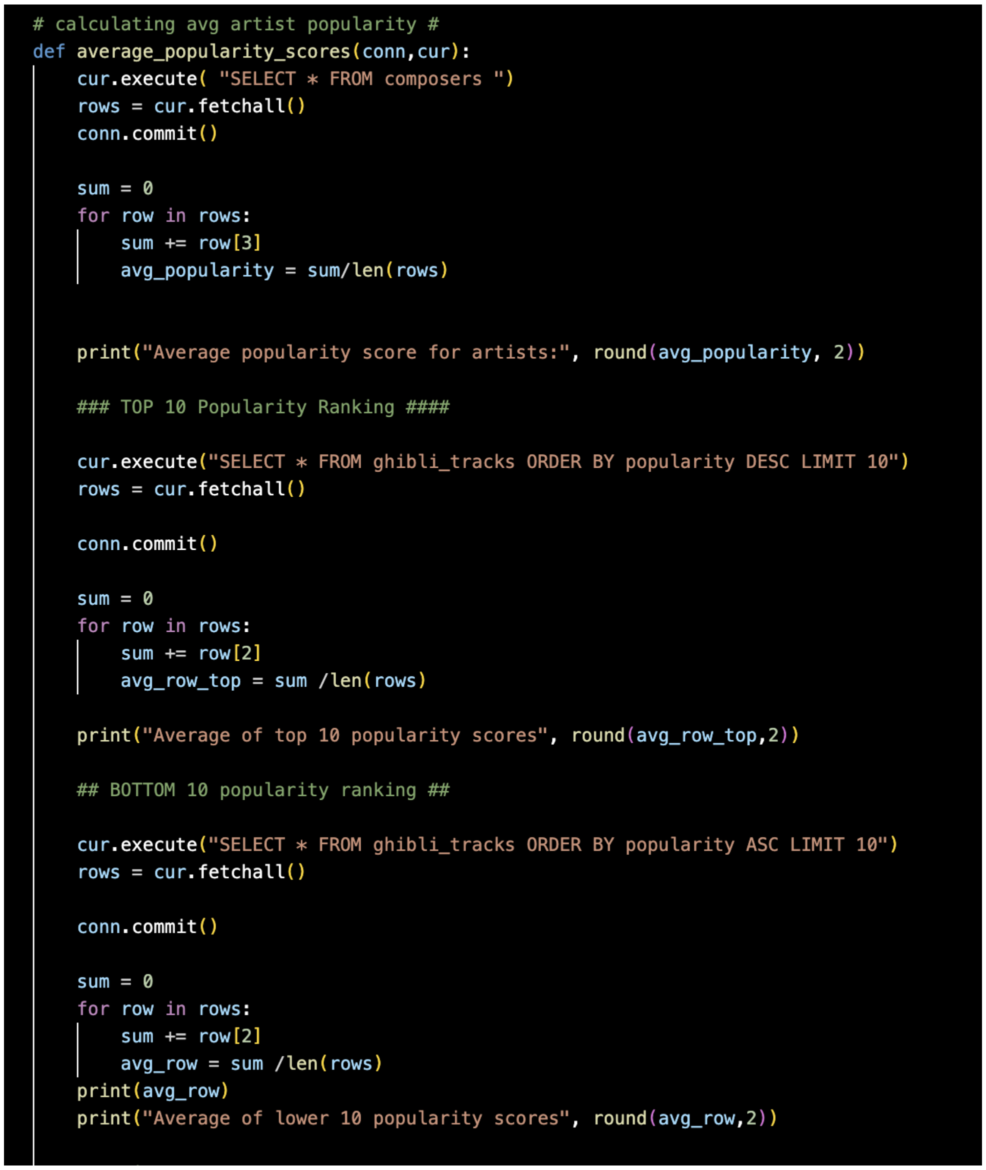




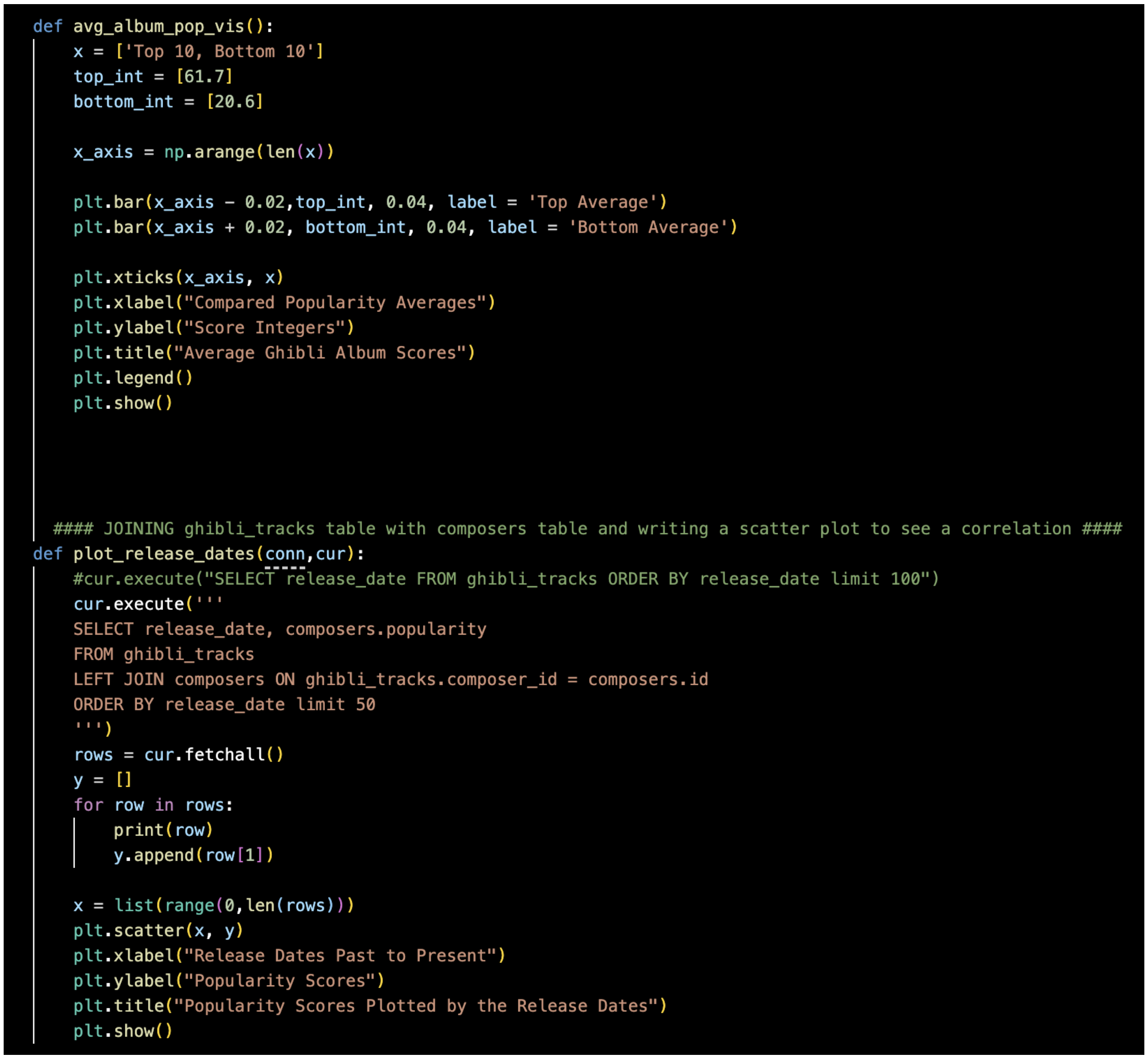
Joey: Reddit Anime visualization



Isabella calculation functions:

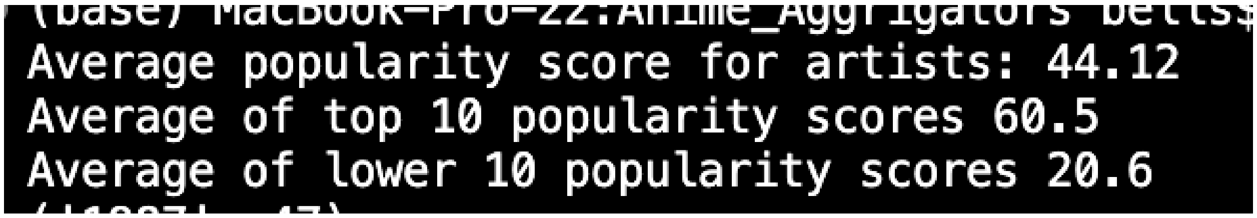


```python
# calculating avg artist popularity #
def average_popularity_scores(conn,cur):
    cur.execute( "SELECT * FROM composers ")
    rows = cur.fetchall()
    conn.commit()

    sum = 0
    for row in rows:
        sum += row[3]
        avg_popularity = sum/len(rows)


    print("Average popularity score for artists:", round(avg_popularity, 2))

    ### TOP 10 Popularity Ranking ####

    cur.execute("SELECT * FROM ghibli_tracks ORDER BY popularity DESC LIMIT 10")
    rows = cur.fetchall()

    conn.commit()

    sum = 0
    for row in rows:
        sum += row[2]
        avg_row_top = sum /len(rows)

    print("Average of top 10 popularity scores", round(avg_row_top,2))

    ## BOTTOM 10 popularity ranking ##

    cur.execute("SELECT * FROM ghibli_tracks ORDER BY popularity ASC LIMIT 10")
    rows = cur.fetchall()

    conn.commit()

    sum = 0
    for row in rows:
        sum += row[2]
        avg_row = sum /len(rows)
    print(avg_row)
    print("Average of lower 10 popularity scores", round(avg_row,2))
```



```python
def avg_album_pop_vis():
    x = ['Top 10, Bottom 10']
    top_int = [61.7]
    bottom_int = [20.6]

    x_axis = np.arange(len(x))

    plt.bar(x_axis - 0.02,top_int, 0.04, label = 'Top Average')
    plt.bar(x_axis + 0.02, bottom_int, 0.04, label = 'Bottom Average')

    plt.xticks(x_axis, x)
    plt.xlabel("Compared Popularity Averages")
    plt.ylabel("Score Integers")
    plt.title("Average Ghibli Album Scores")
    plt.legend()
    plt.show()




  #### JOINING ghibli_tracks table with composers table and writing a scatter plot to see a correlation ####
def plot_release_dates(conn,cur):
    #cur.execute("SELECT release_date FROM ghibli_tracks ORDER BY release_date limit 100")
    cur.execute('''
    SELECT release_date, composers.popularity
    FROM ghibli_tracks
    LEFT JOIN composers ON ghibli_tracks.composer_id = composers.id
    ORDER BY release_date limit 50
    ''')
    rows = cur.fetchall()
    y = []
    for row in rows:
        print(row)
        y.append(row[1])

    x = list(range(0,len(rows)))
    plt.scatter(x, y)
    plt.xlabel("Release Dates Past to Present")
    plt.ylabel("Popularity Scores")
    plt.title("Popularity Scores Plotted by the Release Dates")
    plt.show()
```

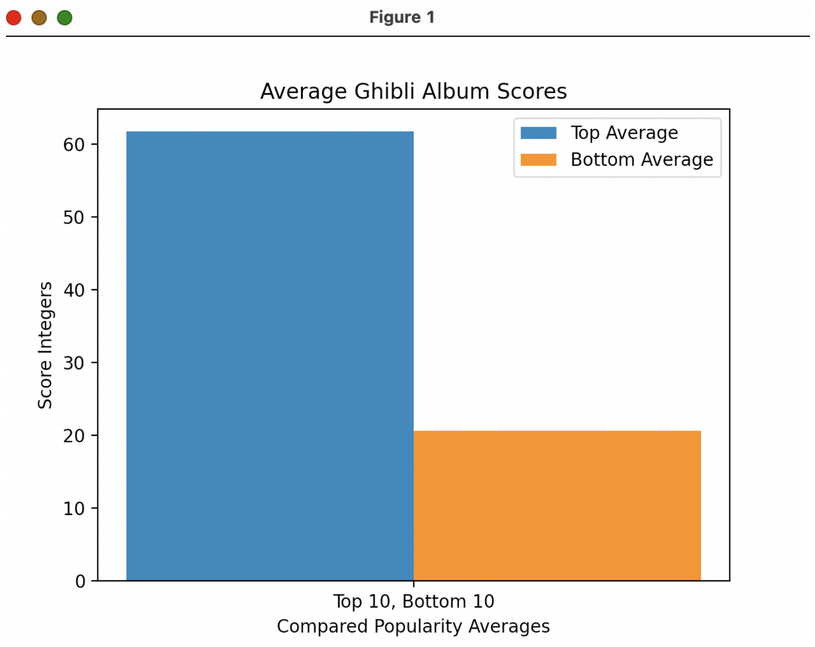
Isabella: Spotify average popularity scores for ghibli music, average of top 10, and average of bottom 10.



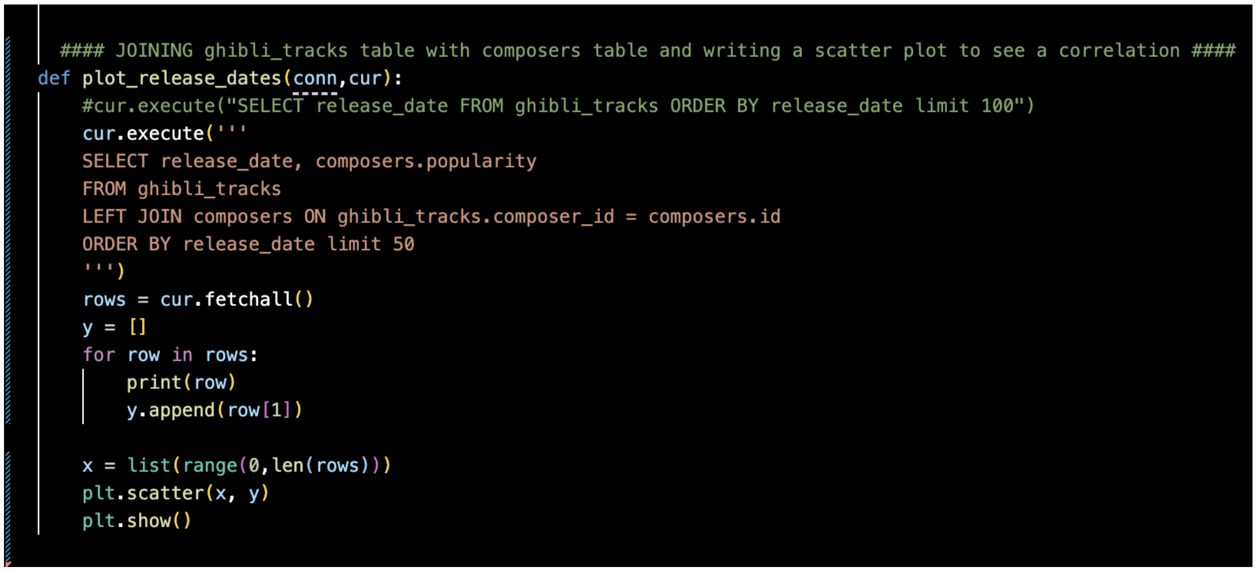
```
(base) MacBook-Pro-22:Anime_Aggrigators bells$
Average popularity score for artists: 44.12
Average of top 10 popularity scores 60.5
Average of lower 10 popularity scores 20.6
```

Isabella:

Bar graph of top and bottom average popularity scores

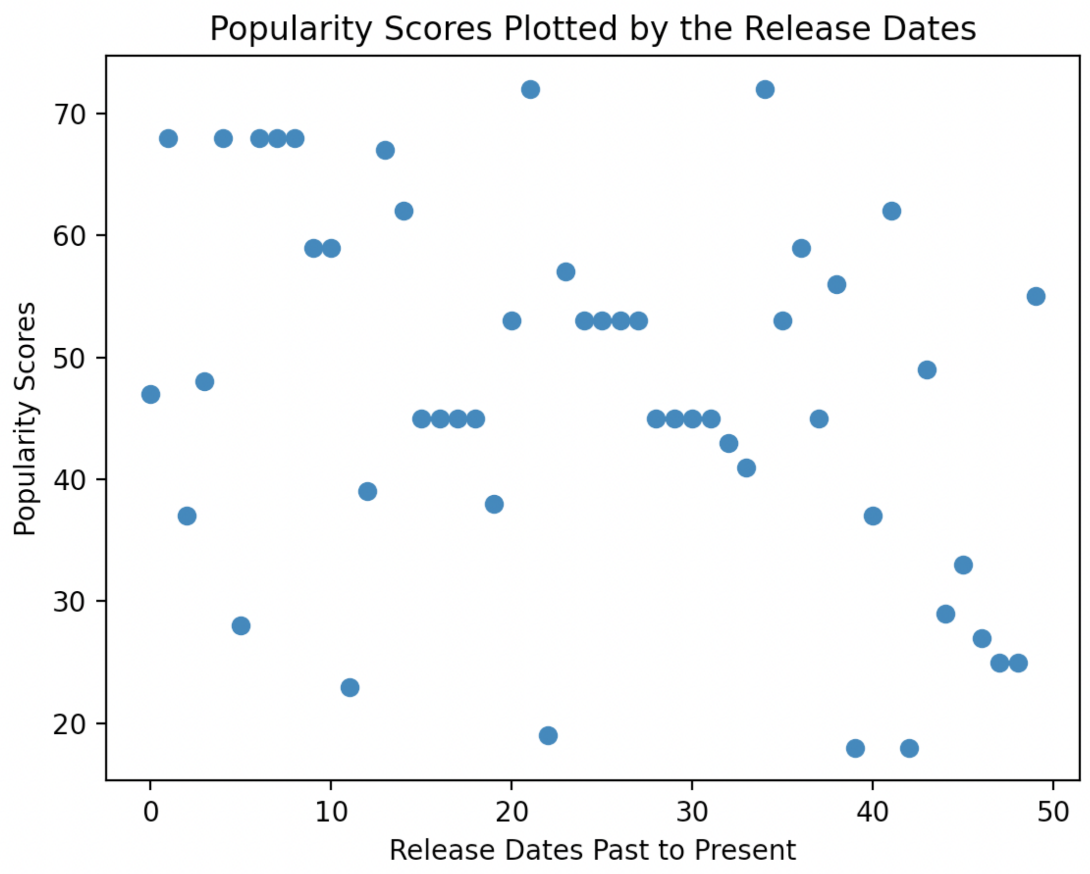


Isabella :
Plot_release_dates will select and join data from composers table and ghibli_tracks table and then it creates a scatter plot showing the relationship between release dates and popularity.



```
    #### JOINING ghibli_tracks table with composers table and writing a scatter plot to see a correlation ####
def plot_release_dates(conn,cur):
    #cur.execute("SELECT release_date FROM ghibli_tracks ORDER BY release_date limit 100")
    cur.execute('''
    SELECT release_date, composers.popularity
    FROM ghibli_tracks
    LEFT JOIN composers ON ghibli_tracks.composer_id = composers.id
    ORDER BY release_date limit 50
    ''')
    rows = cur.fetchall()
    y = []
    for row in rows:
        print(row)
        y.append(row[1])

    x = list(range(0,len(rows)))
    plt.scatter(x, y)
    plt.show()
```

Scatter plot created by plot_release_dates:



6.Instruction to run the code (p.s we set up the main python file and tried to connect our functions but it wasn't really working)

Isabella:

1. Run the file anime_spotify.py to create the database called anime.db. It will have to run 4 times to load all of the data required.

2. Then run anime_spotifycal.py to execute the calculations, visualizations, The Join statement, and the csv.

Joey:

1. First run through redditanime.py. This file contains the API url that grabs the data points with client ids and secret ids. As well as functions that process the data from the API information, and grab 25 items at a time to add into the database. (the result will be return from redditmain.py)

2. Second run is the redditmain.py. This file grabs the data from API and adds data into the database, runs the calculation, and creates visualization based on the calculation.

Elaine:

1. Run through the anime_list.py file. This file contains the function to grab data straight from the API as well as tables created and sent into the database file.

2. Second run would be the animelistcal.py. This is an anime list API calculation which I access straight from the table and grab top ten and bottom ten of scores and popularity. In addition it also included the visualization of the calculation.

7. Documentation of each function wrote

animereddit.py and redditmain.py (Joey):

def add_items: this function pulls data from the reddit api, it directly references the url for r/anime/new, which brings in new anime posts on the subreddit. This function limits the pull to 25 items for each time the code is run and will add on subsequently.

Input: data from reddit api url, auth token, last id used
Output: makes data-frame from r/anime, writes data to items.csv

def create_cursor: initializes/adds data into the anime.db file

Input: N/A, begins constructing a table
Output: anime.db database file

def make_quote_table: organizes the data into a table that can be used further for analysis, calculations, visualization, etc.

Input: data frame information from r/anime brought in by def_create_cursor
Output: a table named “quote” that shows title, selftext, and upvote ratio

def get_average_ratio(cur): Calculates the average upvote ratio for posts on r/anime

Input: takes data from upvote_ratio in quote for calculations

Output: The mean(average) of the upvote_ratio on r/anime posts

def get_average_title_length(cur): Calculates the average length (in characters) of the post titles on r/anime

Input: takes data from title in quote for calculations

Output: the mean(average) character length of the post titles in r/anime

def plot_hist(cur): Used this function to select specific data from the table and create bar graphs

Input: pulls from both ratios and titles, data and calculations

Output: a bar graph that shows the average character length in r/anime post titles vs the average upvote ratio of r/anime posts with appropriate captions and coloring

AnimeList.py and AnimeListcal.py(Elaine):

Def anime_process():

This function is getting access to the API for anime list databases. I looped through from page 1 to 5 and processed the data to only grab the title, score, and popularity.

Input: use the range function to loop through the url pages and made another for loop to process the data and grab what information I want

Output: The process data is put into the list of tuples.

Def database_setup(anime_name):

This function is to set up pathway for connecting to the conn and cur for databases

Input: connection of os.path way to conn and cur

Output: returns the cur and conn (doesn’t print anything)

Def anime_list_table(cur, conn, anime_info):

This function is creating the table into the database as well as limiting the 25 data per run into the database

Input: call in the parameter of cur, conn, and anime_info. Create the tables with column names. Loop through the range in anime_info that limits the data to grab at 25 items per run.

Output: conn commit it and the table in database will show it limits to 25 items per run.

Ps: the table in the anime list table, some anime names will look similar but there are differences by different punctuation behind therefore it is not a duplicate string.

Def avg_anime(cur, conn):

Cur execution to grab data points from the table and run calculation by doing averages of top ten and bottom ten of scores and popularity.

Input: select statement to grab from anime_list table in limit of 10 items, fetchall, and commit it. Next, simply calculation on both columns using average equation

Output: Return the average calculation into the csv file.

Calculation file

I also wrote the calculation into the csv file. After I completed the calculation, I wrote the csv file right under the function.

Output: calculate data points in csv file

Def top_anime_popularity(cur, conn)

This is calculation of percentage by grabbing the ten highest and lowest popularity number from the anime table

Input: cur execute and calculate the percentage

Output: two percentage shown of highest and lowest

Def visual ():

This is the visualization I graphed with my four data points calculation. I looked through matplotlib and came up to graph two different visualizations under one figure.

input: style the color of data points differently, and set up two different bar graphs under one function

Output: figure images show two different graphs under one function run.

Def visual2 ():

Second visualization that graphs two points of highest and lowest popularity number

Input: put it into pie chart in color of blue and orange

Output: pie chart images

Def main():

Calling all of my functions. The cur and conn along with avg_anime and top_anime_popularity. As well as the visual. In addition, it is the anime_process, database setup and anime_list_table function

Output: runs input of each function, making sure it worked

anime_spotify.py and anime_spotifycal.py by Isabella Benjey:
I initially modularized the file anime_spotify.py so that it could run through a main file but we ran out of time to configure all of our files so there are no functions in that file.

average_popularity_scores(conn,cur):
This function calculates the average of the popularity column in the ghibli_tracks table by selecting all of the rows from the composers table and then it takes the sum of the 4th row (popularity) and divides it by the length of the row. It also executes two select statements, one to pull the top 10 popularity scores and the other to select the bottom 10 popularity scores. Lastly, it writes a csv file with the top and bottom popularity averages.

avg_album_pop_vis():

This function creates a bar graph using matplotlib with the data from average_popularity_scores.

plot_release_dates(conn,cur):

This function executes a select statement that grabs the composers popularity scores from the composers table and the release dates from the ghibli_tracks table. It joins them using a left join composer on ghibli_tracks. With that data it creates a scatter plot using matplotlib. The purpose of this function was to see if there is a relationship between popularity and release dates.

8. Source cited

Date	Issue Description	Location of resource	resolve(was it resolved?)
11/28	Trying to get access into API	https://docs.api.jikan.moe/	Yes, able to get access into API and grab data from there
12/9	Having trouble with github push and pull files as one file path was diverted and unmerged	https://stackoverflow.com/questions/2452226/master-branch-and-origin-master-have-diverged-how-to-undiverge-branches Its stake overflow	No, trying to understand what is going on but didn't provide valuable information
12/10	Visualization points wasn't being apply to the graphs	https://matplotlib.org/stable/gallery/lines_bars_and_markers/bar_colors.html#sphx-glr-gallery-lines-bars-and-markers-bar-colors-py Matplotlib official website	Yes, following through the example and trying to understand how to implement the data points correctly.
12/09	I was receiving FutureWarnings about an append function	https://stackoverflow.com/questions/15777951/how-to-suppress-pandas-future-warning Article about pandas implementation	Yes, I found code to override this message and also found ways around using pandas
12/12	Using different colors on the visualization	https://matplotlib.org/stable/gallery/color/named_colors.html	Yes, I find the names of other colors I can use in matplotlib
12/10	I was wondering how to use .concat	https://www.wrighters.io/dont-append-r	Yes, the source showed me what I

	properly	ows-to-a-pandas-dataframe/	needed to use .concat in pandas
12/6	Was interested in the feasibility of the reddit API	https://www.youtube.com/watch?v=NRgfgtzIhBQ Video that demonstrates how reddit api could be used in python	Yes, I learned that the reddit api could in fact be used for a python project
12/8	Couldn't figure out the process for creating and utilizing a token	https://stackoverflow.com/questions/41354205/how-to-generate-a-unique-auth-token-in-python Stackoverflow information on creating auth tokens	Yes, this helped increase my understanding
12/12	Needed help with the structure of using matplotlib	https://stackoverflow.com/questions/8575062/how-to-show-matplotlib-plots Information and strategies for matplotlib	Yes, very useful resource that moved me forward
12/8	Was trying to find ways to process the data from API	https://blog.hubspot.com/website/python-enumerate#:~:text=What%20does%20enumerate%20do%20in,the%20collection%20easier%20to%20access. Taught me about the enumerate property	Yes

12/1	Authentication for Spotify Api "spotipy"	https://www.youtube.com/watch?v=3RGm4jALukM This is a really good tutorial that walks you through how to set up authentication for spotipy.	yes
12/1 - 12/10	Getting started / installing spotipy	https://spotipy.readthedocs.io/en/2.21.0/ This entire website is dedicated to supporting people interacting with spotipy. I came back to this site throughout the project.	yes
12/8-12/11	Git hub merge issues	https://docs.github.com/en/pull-requests/collaborating-with-pull-requests/addressing-merge-conflicts/resolving-a-merge-conflict-on-github Github has pretty good support and instructions on how to fix something. We used this site to find out more about	

12/11	Finding example of pie chart visualization	https://www.w3schools.com/python/matplotlib_pie_charts.asp	Yes, it worked
12/12	Setting the x and y labels for visualization	https://matplotlib.org/stable/gallery/lines_bars_and_markers/bar_colors.html#sphx-glr-gallery-lines-bars-and-markers-bar-colors-py Use matplotlib as reference	Yes, able to label it correctly.