**1.The goals for your project including what APIs/websites you planned to work with and what data you planned to gather (10 points)**

- We originally intended to use SeatGeek's API and Billboard web scraping to retrieve the location of artists' concerts and the rank of popularity for designated artists. However, after encountering difficulties retrieving the API and Billboard information, we switched to using Spotify's and IMDB's API.

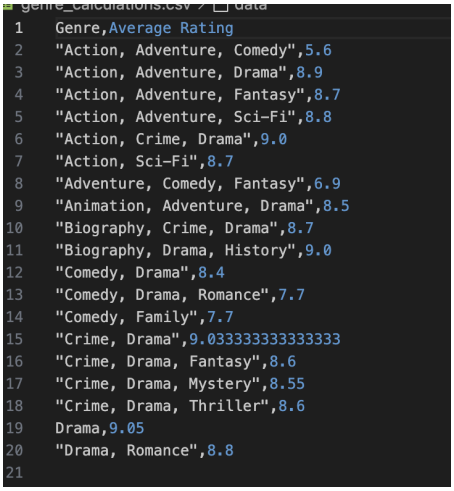
**2. The goals that were achieved including what APIs/websites you actually worked with and what data you did gather (10 points)**

- Data gathered from Spotify: Top tracks and their popularity level (1-100) of each song from a list of artists
- Data gathered from IMDB: Genre, year released, and imdb rating from a list of movies
- Using the Spotify and IMDB API, we calculated the average popularity of a list of artists from Spotify, the average rating of a list of movies from IMBD, and the average rating of each genre from IMDB.

**3. The problems that you faced (10 points)**

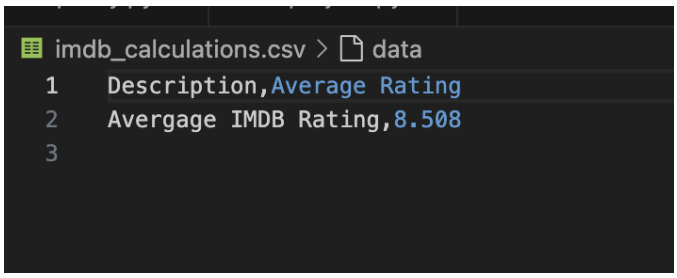
- Since neither of us have had much experience working with API's in the past, one of our main challenges was starting the project and gathering the data from the API's. Our biggest challenge was limiting our data to 25 at a time. It took us a couple of weeks to resolve this issue, but after a helpful Piazza post and a visit to office hours, we were able to successfully complete that portion of the project. The rest of the project went by smoothly other than a few debugging issues.

**4. The calculations from the data in the database (i.e. a screen shot) (10 points)**



```
Genre,Average Rating
"Action, Adventure, Comedy",5.6
"Action, Adventure, Drama",8.9
"Action, Adventure, Fantasy",8.7
"Action, Adventure, Sci-Fi",8.8
"Action, Crime, Drama",9.0
"Action, Sci-Fi",8.7
"Adventure, Comedy, Fantasy",6.9
"Animation, Adventure, Drama",8.5
"Biography, Crime, Drama",8.7
"Biography, Drama, History",9.0
"Comedy, Drama",8.4
"Comedy, Drama, Romance",7.7
"Comedy, Family",7.7
"Crime, Drama",9.033333333333333
"Crime, Drama, Fantasy",8.6
"Crime, Drama, Mystery",8.55
"Crime, Drama, Thriller",8.6
Drama,9.05
"Drama, Romance",8.8
```

From the IMBD data, we calculated the average rating of each movie, and the average rating of each genre



imdb_calculations.csv > data

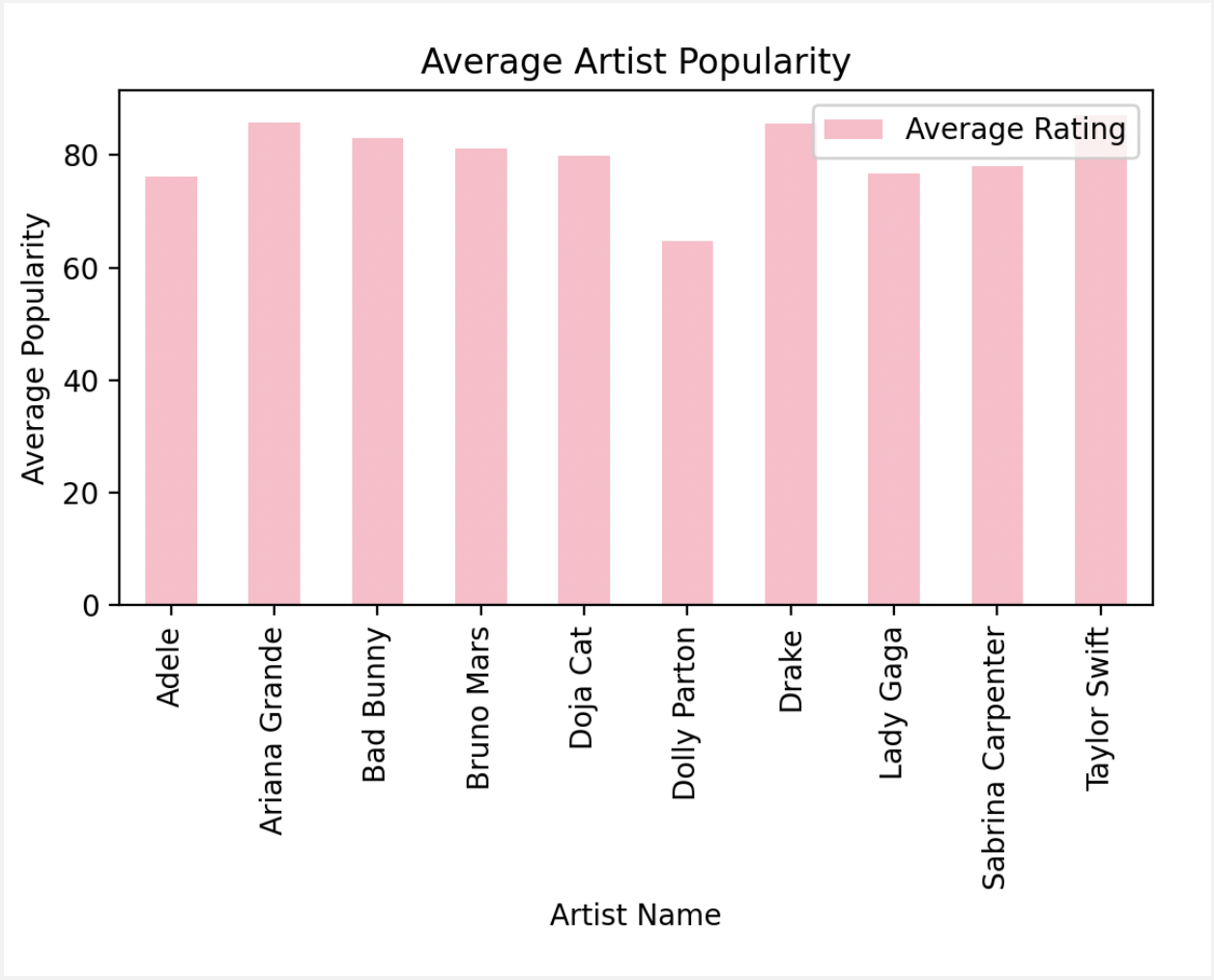
```
Description,Average Rating
Avergage IMDB Rating,8.508
```

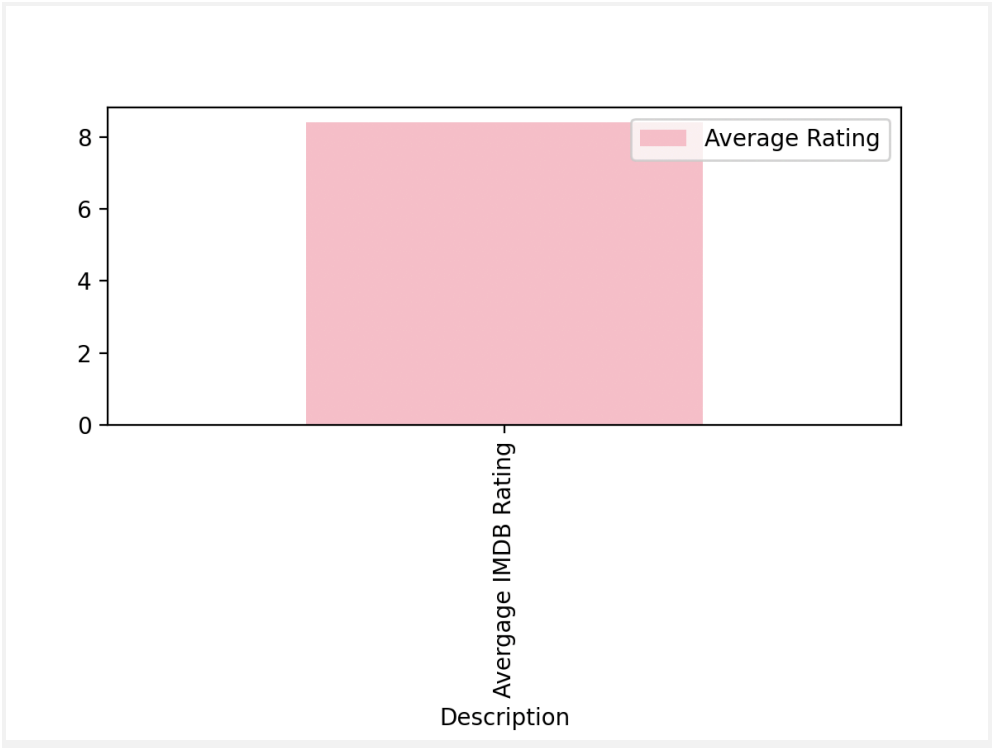
spotify_calculations.csv > data

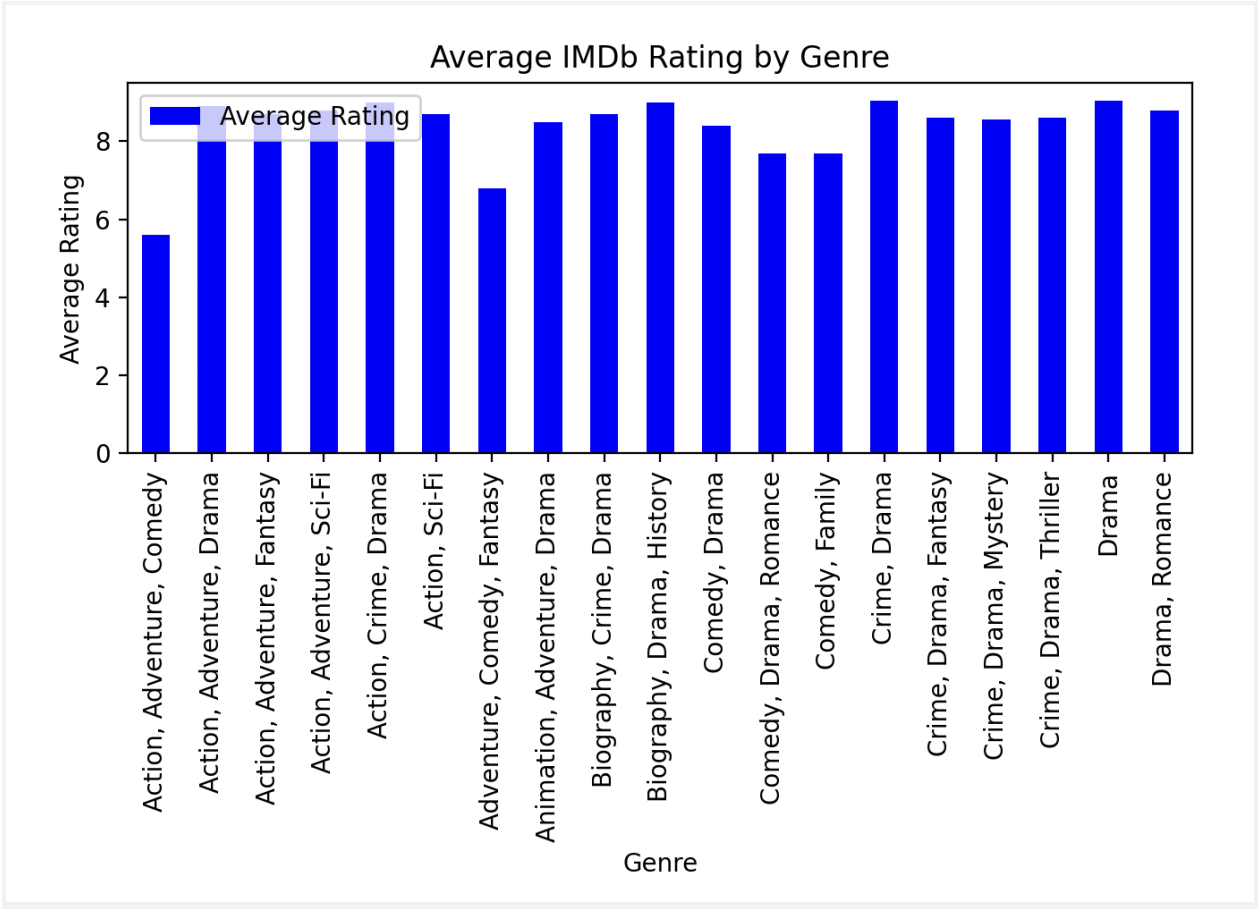
```
Artist Name,Average Rating
Adele,76.2
Ariana Grande,85.8
Bad Bunny,83.0
Bruno Mars,81.1
Doja Cat,79.8
Dolly Parton,64.8
Drake,85.7
Lady Gaga,76.8
Sabrina Carpenter,78.1
Taylor Swift,87.1
```

From the Spotify data, we calculated the average popularity rating of each artist

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




Average Rating
8
6
4
2
0
Avergage IMDB Rating
Description


Average IMDb Rating by Genre
Average Rating
Average Rating
8
6
4
2
0
Action, Adventure, Comedy
Action, Adventure, Drama
Action, Adventure, Fantasy
Action, Adventure, Sci-Fi
Action, Crime, Drama
Action, Sci-Fi
Adventure, Comedy, Fantasy
Animation, Adventure, Drama
Biography, Crime, Drama
Biography, Drama, History
Comedy, Drama
Comedy, Drama, Romance
Comedy, Family
Crime, Drama
Crime, Drama, Fantasy
Crime, Drama, Mystery
Crime, Drama, Thriller
Drama
Drama, Romance
Genre

**6. Instructions for running your code (10 points)**

These instructions apply to both "spotify.py" and "siproject.py"

1. Click on the file (either spotify.py or siproject.py)
    - **ensure that there isn't any code that is already running. If there is a database browser open, delete it**
2. Play the file in the upper right corner and let the code run
3. Repeat steps 1-2 for the second file (either spotify.py or siproject.py)
4. After running each file once, there will be a database titled "combined.db." To open it, double click on the database file and click "Reveal in Finder." Next, double click on "combined.db" in your finder, hover over "Open With" and click on "DB Browser for SQLite."
5. From there, you will have access to three databases. "Artists" (which contains the name and ID associated with each artists in spotify.py) and the first 25 parts of data for the other two databases, "TopTracks" and "movies."
6. Next, exit the database and run each file (spotify.py and siproject.py) an additional 3 times to get 100 rows of data from "TopTracks" and "movies". Reopen the database using the same instruction from step 4 to check the databases with the updated data.
7. Go to "part3.py" and run the file. This will open the calculation files, titled "spotify_calculations.csv", "idmb_calculations.csv", and "genre_calculations.csv"
    - **you will NOT be able to see the full calculations unless you run each file for a total of 4 times as listed in the directions from step 1-6.**
8. Go to "graph.py" and run the file. Three graphs will pop up one after another after your exit each one, you may have to minimize your tabs to see them depending on your device.Adjusting the sliders to your liking on the Subplot Configuration Tool (located on the left of the graph popup, on the left of the magnifying glass icon) will provide a better visual of the full graph. There is a large amount of titles for "Average IMDb Rating By Genre" graph, it is strongly recommended to adjust the "bottom" slider in the Subplot Configuration Tool to properly view the entire graph.

7. **"siproject.py"**

create_database():

- Input: None.
- Output: None.
- Description: This function connects to an SQLite database file named 'combined.db' (or creates it if it doesn't exist) and creates a table named 'movies' if it doesn't exist already. The table has columns for movie ID, title, year, IMDb rating, and genre.

2. fetch_movie_data(movie):

- Input: movie (string) - The title of the movie to fetch data for.
- Output: A dictionary containing movie data fetched from the OMDb API.
- Description: This function takes a movie title as input, encodes it for URL, constructs the API URL for the OMDb API, sends a GET request to the API, receives a JSON response, and returns the parsed data as a dictionary.

3. insert_movie_data(c, data):
    - Input: c (cursor object) - SQLite cursor object for executing SQL queries, data (dictionary) - Movie data retrieved from the OMDb API.
    - Output: None.
    - Description: This function takes the cursor object and movie data dictionary as input. If the data retrieval was successful, it extracts relevant information like title, year, IMDb rating, and genre, and inserts this data into the 'movies' table in the SQLite database. It handles duplicate entries by ignoring them.
4. main():
    - Input: None.
    - Output: None.
    - Description: This is the main function of the script. It connects to the SQLite database, calls create_database() to ensure the 'movies' table exists, fetches movie data from the OMDb API for a list of movie titles, and inserts the data into the database. It iterates through the list of movies, fetching data for each movie, and inserts it into the database. It stops after inserting data for 25 movies.

**“spotify.py”**

1. get_top_tracks(artist_name):
    - Input: artist_name (string) - The name of the artist to retrieve top tracks for.
    - Output: A list of tuples containing track names and their popularity.
    - Description: This function searches for an artist on Spotify using the Spotify API, retrieves the artist's top tracks, and returns a list of tuples where each tuple contains the track name and its popularity.
2. create_database():
    - Input: None.
    - Output: None.
    - Description: This function connects to an SQLite database file named 'combined.db' (or creates it if it doesn't exist) and creates two tables: 'Artists' to store artist information and 'TopTracks' to store top tracks data.
3. store_data(artist_list):
    - Input: artist_list (list) - A list of artist names.
    - Output: None.
    - Description: This function retrieves top tracks data for each artist in the artist_list, stores the data in the 'TopTracks' table of the SQLite database, and associates each track with its corresponding artist.

4. calculate_average_popularity_by_artist():
    - Input: None.
    - Output: A list of tuples containing artist names and their average track popularity.
    - Description: This function calculates the average popularity of tracks for each artist by querying the database and computing the average popularity for each artist.
5. write_to_file(data, filename):
    - Input: data (list of tuples) - Data to be written to the file, filename (string) - Name of the file to write the data to.
    - Output: None.
    - Description: This function writes the provided data to a file specified by the filename. Each tuple in the data list is written to a separate line in the file.
6. graph_spotify():
    - Input: None.
    - Output: None.
    - Description: This function reads data from a CSV file named 'spotify_calculations.csv', creates a pandas DataFrame, and plots a bar graph showing the average track popularity for each artist. The graph is displayed using Matplotlib.
7. main():
    - Input: None.
    - Output: None.
    - Description: This is the main function of the script. It calls create_database() to set up the database, retrieves and displays top tracks for each artist in the artist list, stores the data in the database using store_data(artist_list), and performs other necessary operations.

**“part3.py”**

1. calculate_average_imdb_rating():
    - Input: None.
    - Output: The average IMDb rating of all movies in the database.
    - Description: This function calculates the average IMDb rating of all movies stored in the SQLite database 'combined.db'.
2. calculate_average_popularity_by_genre():
    - Input: None.
    - Output: A list of tuples containing genre names and their average IMDb ratings.
    - Description: This function calculates the average IMDb rating for each genre by querying the database and grouping movies by genre.
3. calculate_movie_count_by_genre():
    - Input: None.

- Output: A list of tuples containing genre names and the count of movies belonging to each genre.
- Description: This function calculates the count of movies for each genre by querying the database and grouping movies by genre.

4. calculate_average_popularity_by_artist():
    - Input: None.
    - Output: A list of tuples containing artist names and their average popularity of top tracks.
    - Description: This function calculates the average popularity of top tracks for each artist by querying the database.
5. write_to_file(data, filename):
    - Input: data (list of tuples) - Data to be written to the file, filename (string) - Name of the file to write the data to.
    - Output: None.
    - Description: This function writes the provided data to a CSV file specified by the filename. Each tuple in the data list is written as a row in the CSV file.
6. main():
    - Input: None.
    - Output: None.
    - Description: This is the main function of the script. It calls each calculation function to calculate various statistics related to IMDb ratings and movie genres, then writes the results to CSV files. Finally, it prints the calculated statistics to the console.

**“graph.py”**

1. graph_spotify():
    - Input: None.
    - Output: Displays a bar graph showing the average popularity rating of top tracks for each artist.
    - Description: This function reads data from the CSV file 'spotify_calculations.csv', which contains information about the average popularity of top tracks for each artist. It then creates a bar graph using Matplotlib, with the x-axis representing artist names and the y-axis representing the average popularity rating.
2. graph_imdb():
    - Input: None.
    - Output: Displays a bar graph showing the average IMDb rating of all movies in the database.
    - Description: This function reads data from the CSV file 'imdb_calculations.csv', which contains information about the average IMDb rating of all movies in the database. It then creates a bar graph using Matplotlib, with the x-axis representing the description and the y-axis representing the average IMDb rating.

3. genre_imdb():
    - Input: None.
    - Output: Displays a bar graph showing the average IMDb rating for each movie genre.
    - Description: This function reads data from the CSV file 'genre_calculations.csv', which contains information about the average IMDb rating for each movie genre. It then creates a bar graph using Matplotlib, with the x-axis representing movie genres and the y-axis representing the average IMDb rating. It also sets the graph title, x-label, and y-label.
4. main():
    - Input: None.
    - Output: None.
    - Description: This is the main function of the script. It calls each graphing function to generate graphs for Spotify data, IMDb data, and IMDb ratings by genre. It displays the graphs one by one.

8. Resources:

| Date: | Issue Description | Location of Resource | Result |
|---|---|---|---|
| 3/28 - 4/19 | Debugging the entire project for almost every function. Alot of missing quotes, indentation errors, had to learn how to write sql. Helped debug the sql titles and add color to graphs | ChatGPT | Fixed a lot of errors, identified missing quotes,indentation and how to add color to graphs |
| 3/28-4/3 | API Implementation | Youtube: https://www.youtube.com/watch?v=2if5xSaZJlg | Use tips to retrieve API |
| 4/9-4/19 | Didn't understand SQL titles | W3schools: https://www.w3schools.com/sql/default.asp | Learned more about SQL and what titles do |
| | | | |

GITHUB LINK: https://github.com/sintiai/project