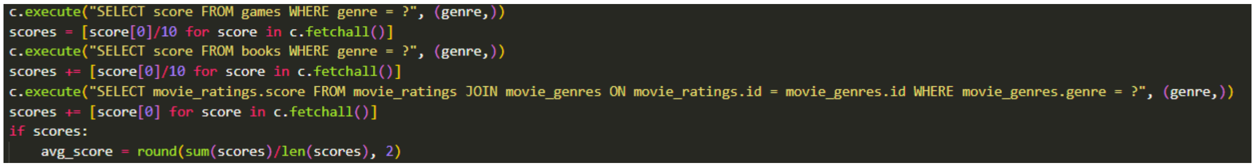
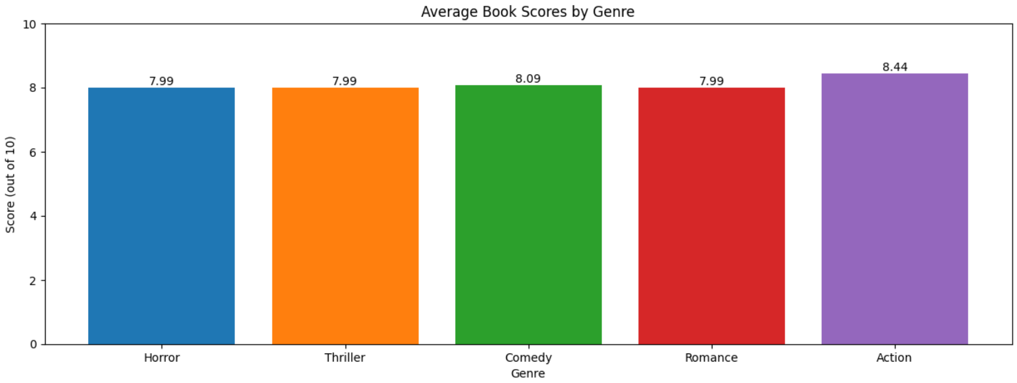
Team: Jonah Ableman, Adena Cartsonis, Angela Cheong
Github Repo: https://github.com/Jableman19/206FinalProject

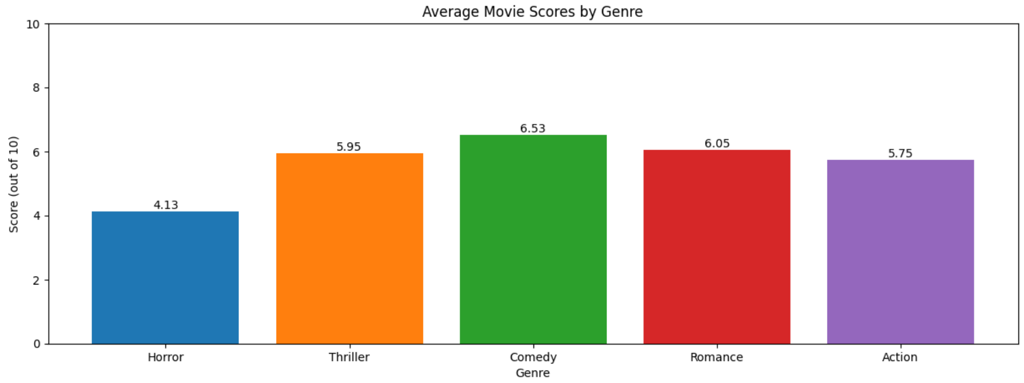
1. **The goals for your project including what APIs/websites you planned to work with and what data you planned to gather (10 points)**
   We wanted to see if there was a relationship between genre ratings over different media types. We wanted to see if there were more popular genres depending on the media type or if regardless of media type, genre popularities were similar. We planned on using GoodReads, MobyGames, and MovieDB to grab data like the movie/book/game, genre, and rating.
2. **The goals that were achieved including what APIs/websites you actually worked with and what data you did gather (10 points)**
   Grab the average ratings for each genre for each media type, generating a database, then using those tables in the databases to create barplot visualizations of our findings. We gathered the same data that we planned to use as well as the same APIs and websites we planned on using
3. **The problems that you faced (10 points)**
   Understanding the APIs and how they want you to specifically grab the data. Another issue we ran into was trying to get our project idea to fit to the specification of the project. APIs are typically built to be used a certain way, and the project introduces requirements or restrictions that were not always conducive to utilizing the API the way it should be. This forced us to have to think outside of the box and come up with our own solutions
4. **The calculations from the data in the database (i.e. a screen shot) (10 points)**
   Genre averages

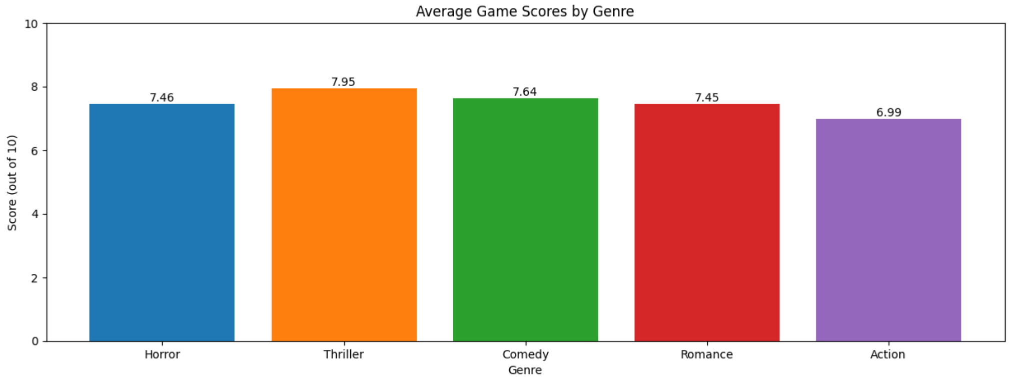


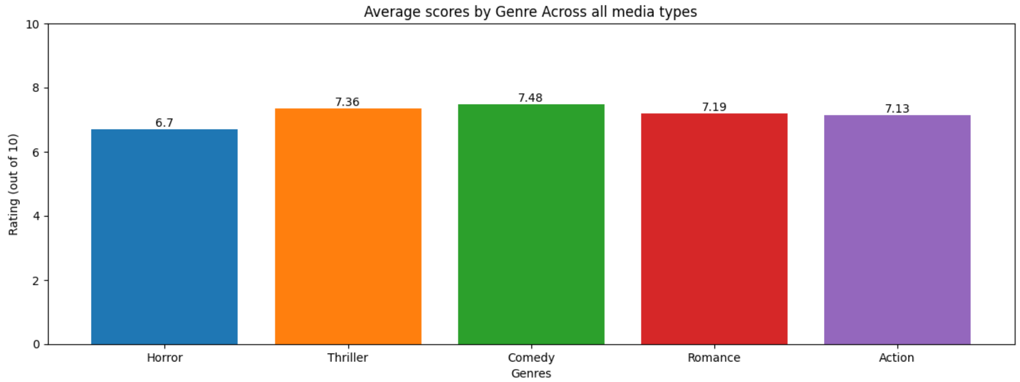
```
c.execute("SELECT score FROM games WHERE genre = ?", (genre,))
scores = [score[0]/10 for score in c.fetchall()]
c.execute("SELECT score FROM books WHERE genre = ?", (genre,))
scores += [score[0]/10 for score in c.fetchall()]
c.execute("SELECT movie_ratings.score FROM movie_ratings JOIN movie_genres ON movie_ratings.id = movie_genres.id WHERE movie_genres.genre = ?", (genre,))
scores += [score[0] for score in c.fetchall()]
if scores:
    avg_score = round(sum(scores)/len(scores), 2)
```

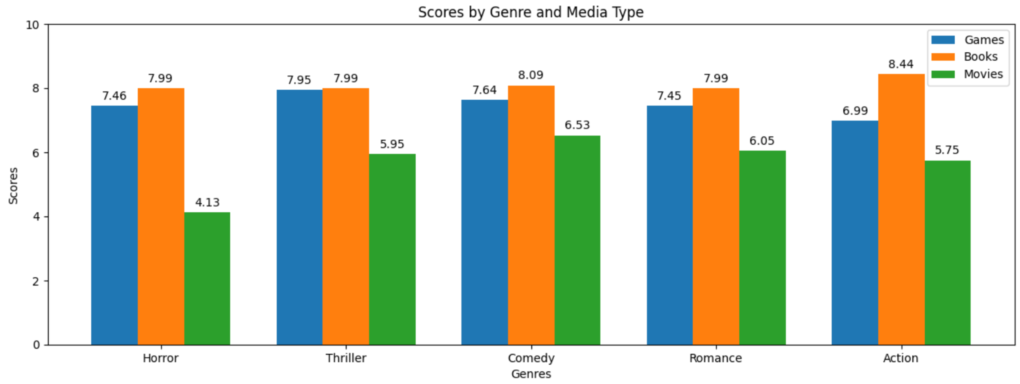
5. **The visualization that you created (i.e. screen shot or image file) (10 points)**




Average Movie Scores by Genre
Score (out of 10)
10
8
6
4
2
0
4.13
5.95
6.53
6.05
5.75
Horror
Thriller
Comedy
Romance
Action
Genre


Average Game Scores by Genre
Score (out of 10)
10
8
6
4
2
0
7.46
7.95
7.64
7.45
6.99
Horror
Thriller
Comedy
Romance
Action
Genre


Average scores by Genre Across all media types
Rating (out of 10)
10
8
6
4
2
0
6.7
7.36
7.48
7.19
7.13
Horror
Thriller
Comedy
Romance
Action
Genres


Scores by Genre and Media Type
Scores
10
8
6
4
2
0
Games
Books
Movies
7.46
7.99
4.13
7.95
7.99
5.95
7.64
8.09
6.53
7.45
7.99
6.05
6.99
8.44
5.75
Horror
Thriller
Comedy
Romance
Action
Genres

6. **Instructions for running your code (10 points)**
   To run our code, navigate to the root of the directory cloned by git and simply run the python script main.py 5 times. On the 5th and final run, the code will also pop up with the 5 different visualizations we created.
7. **Documentation for each function that you wrote. This includes describing the input and output for each function (20 points)**
   Mobygames.py
   - scrapeGenre(gameList): This function takes a list of game IDs as a parameter and scrapes returns a list of the corresponding moby games aggregated score as an output. This function obtains the score by scraping from each individual game page. This function has a runtime of at least 1 second as to avoid rate limiting
   - GetGameKist(APIKEY, genNum): This function takes in an API key and a number corresponding to the intended genre to scrape as input, and outputs a dictionary where the genre is the key and a list of games' unique moby games IDs as the value.

   Goodreads.py
   - main(): The main function for this file is used for testing purposes as it allows us to perform one scrape of goodreads without having to run all the other code. If using the main function, you will specify the genre number as your first command line argument.
   - scrapeBooks(genNum): The function takes in a genre number, which represents the genre that we want to scrape as input, and outputs a dictionary with the genre name as key values and a list of tuples containing the title and rating of each book in the respective genre. To use this function simply call scrapeBooks with an input parameter.

   Movies.py
   - get_movies(api_key, genreID): takes an api key and a genreID (which is a standardized version of the keys across all APIs/sites. The function requests the api by genreid and page- the code loops through each page adding movies that are not already found to a movies_per_genre and scores_nested dict. Movies_per_genre maps a genre name to the movies that are in the genre that is represented by a tuple (movie_id, name, vote_average)
   - main(): adds tables movie_ratings, and movie_genres to the ratings.db

   Barplot.py: Every function in this file requires that the database be filled with the expected items (ie run the main function 5 times). Do not call these functions if the requirements are not met
   - create_games_plot(): Takes in no input and returns a bar plot graph using matplotlib which displays the average rating/scores of games based on their genre
   - create_books_plot(): Takes in no input and returns a bar plot graph using matplotlib which displays the average rating/scores of books based on their genre

- create_movies_plot(): Takes in no input and returns a bar plot graph using matplotlib which displays the average rating/scores of movies based on their genre
- create_genre_ratings_plot(): Takes in no input and returns a bar plot graph using matplotlib which displays the average rating/scores of genres across all media types.
- create_media_types_plot(): Takes in no input and returns a barplot using matplotlib which displays the average genre scores of each media type, grouped together by genre.

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 (did it solve the issue?)**
04/11/2023	How to access the API/ getting API Key for moviesdb		https://www.themoviedb.org/documentation/api	Yes, once they gave me a key and the documentation, it made a lot more sense
04/11/2023	Receiving access to the MobyGames API	The moby games API requires a private key for use and is not publicly accessible	https://www.mobygames.com/info/api/	Yes, after requesting a key with our project detailed, we were granted access
04/16/2023	Matplotlib graph not showing		https://stackoverflow.com/questions/21688409/matplotlib-plt-show-isnt-showing-graph	Did not work, eventually went to office hours to figure it out
04/17/2023	Couldn't get multiple bar charts in one grouping	We were struggling to get multiple bar charts grouped together with a legend for the groups	https://pandas.pydata.org/docs/	Yes, by using pandas and doing some more research we were able to successfully create the intended visualization