Final Project Report

Sarrah Ahmed and Katie Lyngklip

https://github.com/sarraha786/FinalProject.git

Original Goals: Initially we wanted to use geocoding apis to get the distance from North Quad to every other building on campus. However, when we went about this we found that it would be less analysis than the project was intended for. We also attempted to gather preliminary data for it and had trouble getting specific building data. So, we decided to analyze top movie ratings across IMDB and TMDB and their youtube trailer statistics instead

Goals Achieved: We were able to get the top movies from IMDB, their ratings, their release date, each trailer from Youtube (for a list of 100 movies that does not entirely match up to the first table, but outputs useful information regarding movie trailers nonetheless), their view count, and dislike counts. We also used the TMDB database and used the IMDB list of movies to retrieve TMDB ratings for the same movies along with their popularity, release date, language, and vote count.

Problems:

1.github merge errors, error pulling, pushing: We had untracked files in our repository, which prevented us from successfully pushing and pulling changes. After we learned how to track the files, we were able to commit, push, and pull. Later as bigger changes were made, we had issues pulling because the terminal claimed there were many versions of each file, so we had to meet up and decide which changes were valid because visual studio gives you the option to accept/reject changes. Eventually, it became easier to do main changes on one person's computer because we kept getting merge errors, otherwise.

2.Youtube API  quota: While gathering view counts, likes, and dislikes from the Youtube API was not especially difficult, the limit of 1000 requests per day was a major obstacle for our group. Additionally, since there are 250 movies, we had to space our requests over a week, so we decided to just take the top 100 from the file (which was sorted most recent to least recent in order to optimize youtube search).

3.After successfully getting youtube trailer information for the top 100 movies, we realized the top 250 movies had changed on IMDb. This threw a major wrench in our project because we found this out on the last day, and did not have enough time to redo all of our youtube requests. So, we downloaded the top 250 movies from that day as an

HTML file so it would not change, and used the TMDb API to get voter averages for the top 250 movies to supplement our calculations.

4. Limiting database: Since the original table for the top 250 movies only had “name”, “year”, and “rating”, we were finding it hard to load in 25 items at a time. In lecture, Dr. Ericson suggested that we should use SELECT MAX to make sure the data does not duplicate when run multiple times. So, we created ids to make this easier and the function increments the index as it loops through.

Calculation File:

youtube_calculations.csv

```
The average number of views for the movie trailers was 13261027.44
The average number of likes for the movie trailers was 176647.2
The average number of dislikes for the movie trailers was 13084380.24

```

tmdb_calculations.csv

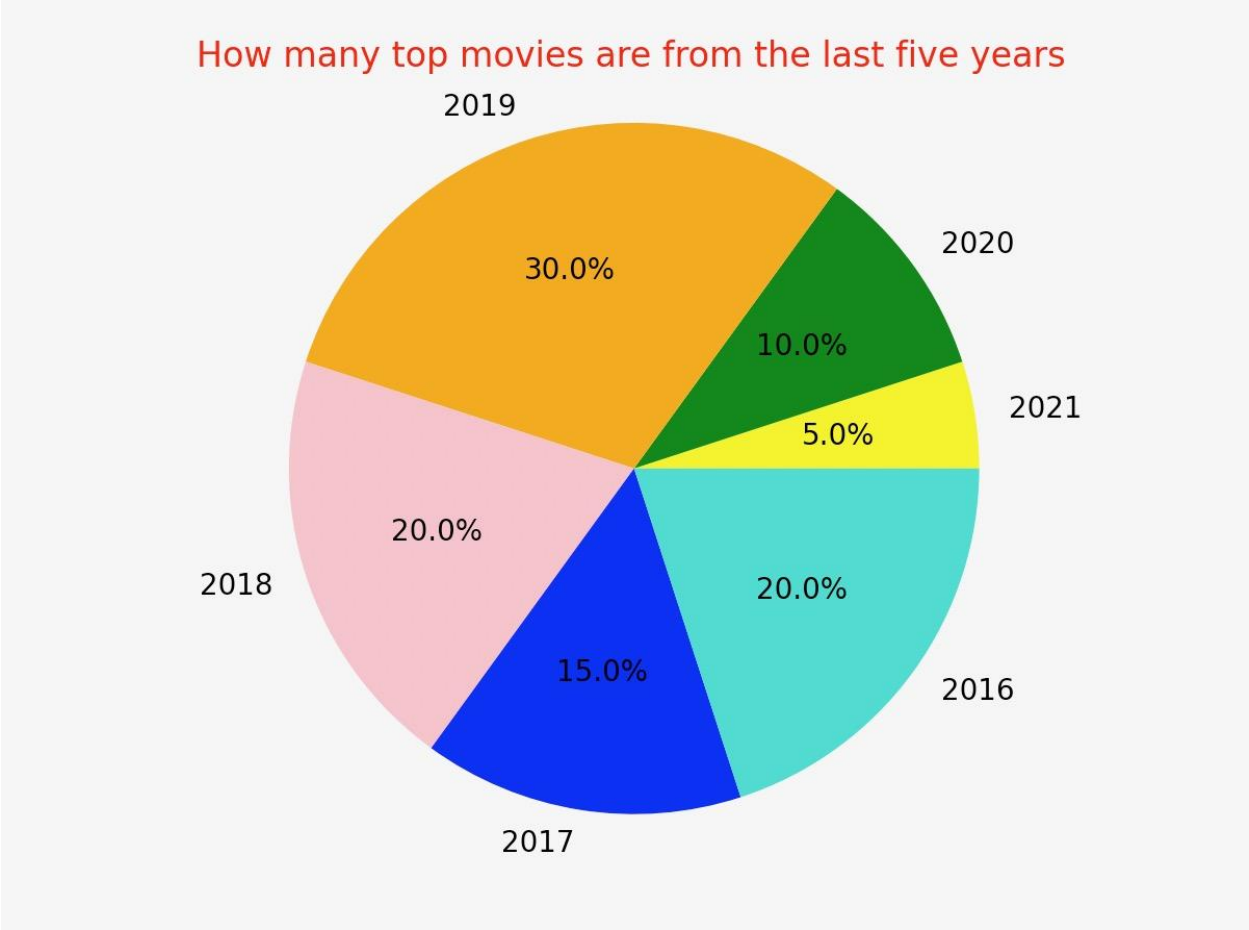
```
Movie Name,Absolute Value Difference Between Ratings from IMDB and TMDB
12 Years a Slave,0.2
1917,0.2
3 Idiots,0.3
A Beautiful Mind,0.3
A Separation,1.0
American Beauty,0.3
Amores perros,0.4
Amélie,0.3
Avengers: Endgame,0.0
Avengers: Infinity War,0.1
Batman Begins,0.5
Before Sunset,0.2
Capernaum,0.1
Catch Me If You Can,0.1
City of God,0.2
Coco,0.1
Dangal,8.2
Django Unchained,0.3
Downfall,0.9
Eternal Sunshine of the Spotless Mind,0.2
Fight Club,0.4
Finding Nemo,0.3
Ford v Ferrari,0
Gladiator,0.3
Gone Girl,0.2
Gran Torino,0.1
Green Book,0.0
Hachi: A Dog's Tale,0
Hacksaw Ridge,0.1
Hamilton,0.2
Harry Potter and the Deathly Hallows: Part 2,0.0
Hotel Rwanda,0.4
How to Train Your Dragon,0.3
Howl's Moving Castle,0.3
Incendies,0.1
Inception,0.3
Inglourious Basterds,0.1
Inside Out,0.2
Interstellar,0.2
```

```
Into the Wild,0.2
Joker,0.1
Kill Bill: Vol. 1,0.1
Kill Bill: Vol. 2,0.1
Klaus,0.2
Like Stars on Earth,0.2
Logan,0.2
Mad Max: Fury Road,0.5
Mary and Max,0.2
Memento,0.2
Memories of Murder,2.2
Million Dollar Baby,0.1
"Monsters, Inc.",0.3
My Father and My Son,0.5
No Country for Old Men,0.2
Oldboy,2.4
Pan's Labyrinth,0.5
Parasite,3.8
Pirates of the Caribbean: The Curse of the Black Pearl,0.2
Prisoners,0.0
Ratatouille,0.2
Requiem for a Dream,0.3
Room,1.6
Rush,1
Shutter Island,0.1
Snatch,0.4
Spider-Man: Into the Spider-Verse,0.1
Spider-Man: No Way Home,0.2
Spirited Away,0.5
Spotlight,0.2
The Dark Knight,0.5
The Dark Knight Rises,0.5
The Departed,0.3
The Father,0.0
The Grand Budapest Hotel,0.0
The Green Mile,0.1
The Handmaiden,0.3
The Help,0.2
The Hunt,1.7
The Incredibles,0.3
```

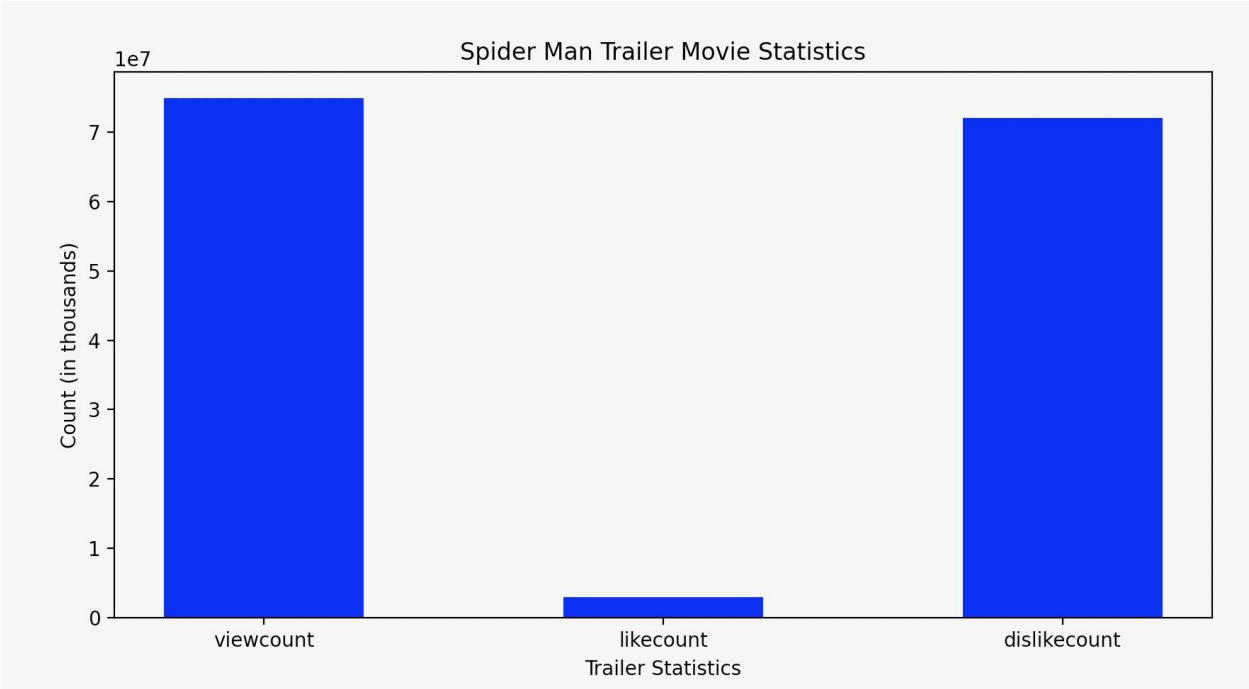
```
The Intouchables,0.2
The Lives of Others,0.3
The Lord of the Rings: The Fellowship of the Ring,0.4
The Lord of the Rings: The Return of the King,0.4
The Lord of the Rings: The Two Towers,0.3
The Matrix,2.0
The Pianist,0.1
The Prestige,0.3
The Secret in Their Eyes,0.1
The Sixth Sense,0.3
The Wolf of Wall Street,0.2
There Will Be Blood,0.1
"Three Billboards Outside Ebbing, Missouri",0.0
Toy Story 3,0.5
Up,0.3
V for Vendetta,0.2
WALL·E,0.3
Warrior,0.3
Whiplash,0.1
Wild Tales,0.3
Your Name.,8.3
```

Visualizations:

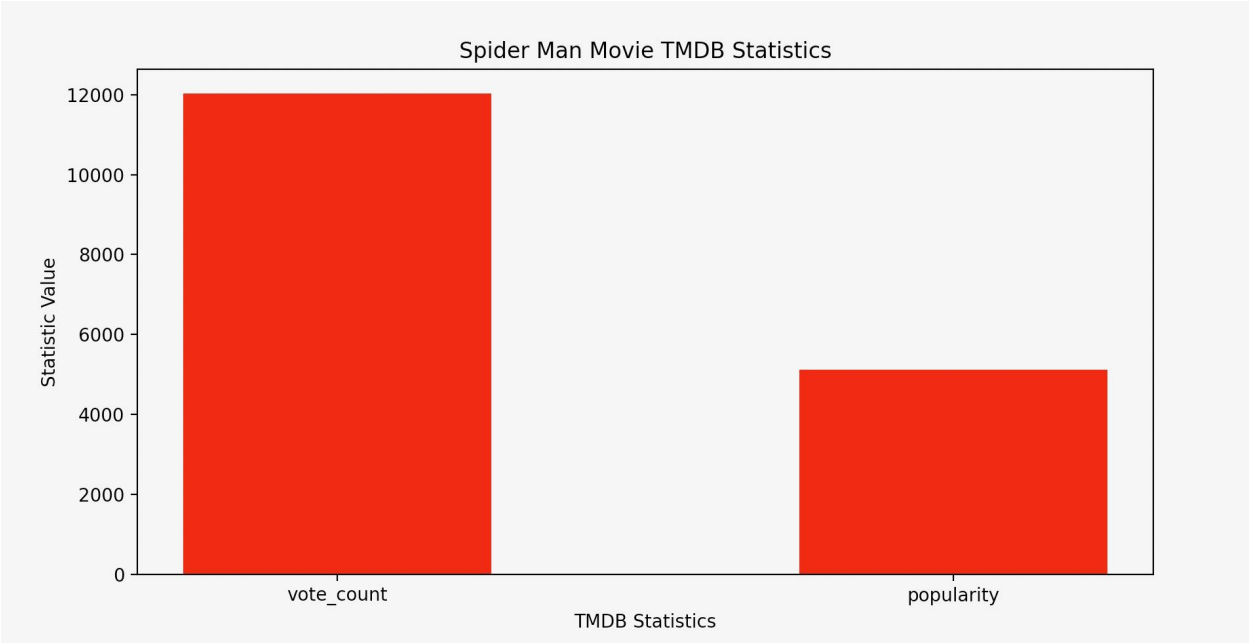
1. From IMDB Database, we isolated the last five years to see which years had more appearances on the top 250 list.



2. Youtube API visualization for Spider Man No Way Home trailer




3. TMDB visualization for Spiderman No Way Home



Instructions for running code: Run the code four times at least four times to get 100 rows in each table. If the code is run 10 times, the Movies table will have 250 rows.

To see visualizations: uncomment lines 317,324,and 329.

To use the youtube api to collect the necessary data for the Youtube_Statistics Database, you need to run the code three times and uncomment content in the requesting_yt_stats function. First, run it with only line 163 uncommented. Second, rerun it with only 164 uncommented. Third, run it with only line 165 uncommented. Youtube content will be written in a csv_file named yt_trailer_data.csv.

Code Documentation:

```
def setUpDatabase(db_name):
```

Takes in “final_project_db.db” as a parameter and returns cur, conn

```
def get_top_movies(html_file)
```

Takes in a downloaded html file of the IMDB top 250 movies, reads the file, creates a beautiful soup object, and scrapes the file for the top movies, their release date, and their rating. The soup puts each variable into a list and then outputs a zipped list of ids, movie names, release dates, and rating.

```
def movies_table(data,cur,conn):
```

Takes in the zipped list of tuples from the previous function with id, movie name, year, and rating.

This function creates the “Movies” table and inputs the tuple information 25 items at a time. This is done by keeping track of the id number and starting where the function left off in terms of loading in the tuples to avoid repetition. This function does not output anything.

**`def movie_viz(cur):`**

This function takes the cur, conn from the function that sets up the database. In this function we use SQL to get the rows with movies that were released after 2015. Then, we put these years into a dictionary, Counting the number of times each year in this subset appears. This information gets visualized in a pie chart , but the function itself does not return anything.

**`def get_youtube_info(movie_name):`**

This function takes in one movie name and uses the youtube API in two ways:

1. We use the API to search for the video ID for each movie’s trailer
2. We then use the API to get the movie trailer’s statistics

This function returns a list with the movie’s statistics [view count,like count, dislike count, favorite count, comment count]

**`def writecsv_w_ytdata(data):`**

This function takes in a list of trailer stats and puts them in a csv file because we will need to collect this data over the span of a few days due to youtube  api quotas. It writes a file and returns None.

**`def writing_movie_info(movie_lst,start_point,end_point):`**

This function takes in a list of 100 movies taken from get_top_movies(). It loops through each movie. The Starting point parameter is meant to help control how much data is written at a time because we must work within youtube quotas. In this function, we get trailer data for each movie in the list and send it back to the writecsv_w_ytdata function.

**`def requesting_yt_stats(movie_tuples):`**

This function uses the writing_movie_info function to request movie trailer info for each movie (separated in groups of 34 and 33 to get around query api overload. This function basically contains all the code used to call

functions to get youtube trailer information.

**def csv_reader(csv_file):**

This function reads in the existing csv file that contains the trailer stats and creates a list of lists. Each list contains the movie id, title, viewcount, likecount, dislike count. This function returns the list of lists.

**Def write_movietrailer_table(csv_data,cur,conn):**

This function takes in the data from the previous function and writes a new table. It loads the rows in 25 at a time and does not return anything.

**Def youtube_calc(cur):**

This function takes in the cursor from the database and computes the average views, likes, and dislikes Overall of the movies we have from IMDB. This function returns None.

**Def tmdb_api(movie_name):**

This function takes in one movie name from the output of def tmdb_database_prep(). It uses the TMDB API to search each movie and capture the average vote, release date, original language, vote count, and Popularity from TMDB. This function returns the aforementioned variables as a list.

**Def tmdb_database_prep(movie_tuples):**

This function takes in the tuple of movies from the last function and appends a list called output with the result of searching the movie in TMDB.

**Def tmdb_database(data,cur,conn):**

This function takes in the data from the TMDB API and writes it into a database called "TMDB" 25 rows at a time. Data is a list of lists, each representing the data for the first 100 movies from the IMDB Soup.

**Def tmdb_calculation(cur):**

This function takes in cursor from the database and joins the Movies table with the TMDB table on ratings and voter_avg, both are values out of ten. Then it computes the absolute value of the difference between both the numbers for each movie. Then, it writes it into the calculation CSV file.

**Def tmdb_viz(cur):**

This function takes in cur from the database and creates a bargraph of the average vote, vote count and Popularity of the Spiderman No Way Home movie

**Def youtube_viz(cur):**

This function takes in cur from the database and creates a bargraph of the number of views, Likes, and dislikes for the first movie on the IMDB list: Spiderman.

**Def main():**

This function calls each function and configures their parameters

Resources Used:

| Date | Issue Description | Location of Resource | Result |
|---|---|---|---|
| 4/20 | While scraping IMDB for the top 250 movies, the date released was in parentheses, so we needed a refresher on how .replace() works | https://www.w3schools.com/python/ref_string_replace.asp | We successfully removed the parentheses from the data the movie was released |
| 4/20 | Needed to reference the correct "CREATE" database statement | Lecture 18- DBs and APIs | We successfully created our first database table- top movies from IMDB |

| | | | |
|---|---|---|---|
| 4/20 | While creating the first visualization, which was a pie chart, whenever we ran plt.show(), the chart file showed up but without the actual data or pie pieces | https://www.w3schools.com/python/matplotlib_pie_charts.asp | We were able to use the correct code to configure the pie chart with different colors and got the text to show up too |
| 4/23 | Since we both were working on the project independently, we had major issues with github. We got errors that files were untracked, so we couldn't pull each others edits. | https://stackoverflow.com/questions/8470547/git-commit-a-untracked-files | We were able to track all the files and pull each other's changes. |
| 4/25 | We needed a refresher on how to write CSV files | https://www.pythontutorial.net/python-basics/python-write-csv-file/ | We created a function to write out the calculations |
| 4/25 | Youtube and TMDB required a number of additional libraries to be installed | https://packaging.python.org/en/latest/tutorials/installing-packages/ | We downloaded all packages associated with each API to reference in our code |
| 4/26 | In our calculations file, the results were being written with commas in between each item | https://stackoverflow.com/questions/15129567/csv-writer-writing-each-character-of-word-in-separate-column-cell | We were able to output a csv file with calculations in a readable format |
| 4/26 | We were having trouble using the TMDB API, so we looked up documentation | https://pypi.org/project/tmdbv3api/ | This resource was incredibly helpful and using it we were able to create a table with the |

| | | | |
|---|---|---|---|
| | examples | | values we wanted from the API |