Jennie Tannenbaum and Maya Horowitz SI 206 Final Project
Link to Repository: https://github.com/jstbaum/Final-Project206

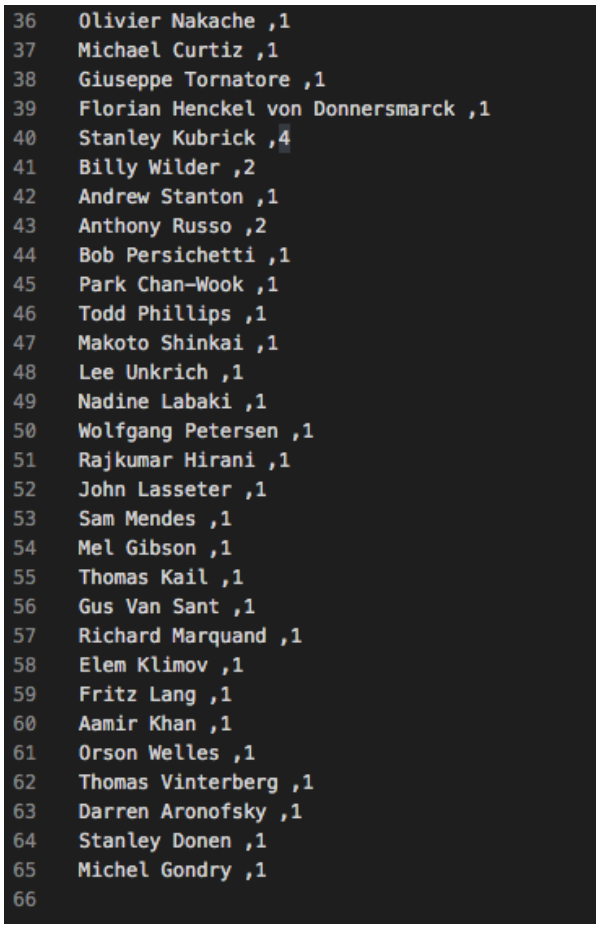
Goals
The main goal for our project was to examine the data from the IMDb API and ultimatemovierankings.com to see the top 100 movies. From this, we wanted to see which Directors had produced the most movies in the top rankings and which years the top movies were made in. Also, we wanted to see the top 100 movies that had the highest box office (adjusted for inflation). After gathering data between the IMDB API and ultimatemovierankings.com, we wanted to see if the top 100 best movies and the top 100 movies with highest box office had any overlap with each other. Considering a movie could be so highly ranked from the IMDB API, we thought that over half of the movies also generated one of the 100th highest box offices.

Achieved Goals
We gathered data from the top 100 movies and made a table in the database with each of these movies' year, rank, IMDb rating and director. We calculated the number of appearances each director had in the top 100. We made a table and Text file that has the Director and number of appearances. Also, we calculated the average IMDb rating (8.479/10). We made a dictionary of which years were in the top 100 movies where the key was the year and value was how many movies were made in this year. We saw that 1994 was the most common year to have a movie in the top 100 movies. We then gathered data from the top 100 movies with the highest box office. We calculated the average box office from these 100 movies which was $429 million. We then made intervals based on $100 million to see where most of the top 100 movies with the highest box office would fall into. Based on our bar graph, most of these movies had a box office between $200-$300 million. After joining the tables from the IMDB API and ultimatemovierankings.com, we made a text file that had all the movies that overlapped on the Box_Office table and Movies table. In this text file we also included their IMDB rank and Box Office (in millions of dollars). Turns out there were only 20 movies that were present in both tables. This was surprising to me because I thought a movie that is so highly ranked would also generate one of the 100th highest box offices.

Problems Faced
Upon completing the imdb api, we attempted to run the code on both computers and were surprised when it did not work. We were repeatedly getting an error saying “150 tries out of 100”. At first we did not know what this error meant, however we realized that the api key could only be used 100 times per day. We both had to make a free account on the imdb api so we could both get our own key. Another problem we faced was that it was difficult working on two separate computers so we did most of the work on Jennie’s. Maya and Jennie met up in person to do the project.

File with Data Calculation

```
You, 2 days ago | 1 author (You)
Director,Appearances
Frank Darabont ,2
Francis Ford Coppola ,3
Christopher Nolan ,6
Sidney Lumet ,1
Steven Spielberg ,3
Peter Jackson ,3
Quentin Tarantino ,4        You, 2 days ago
Sergio Leone ,3
David Fincher ,2
Robert Zemeckis ,2
Irvin Kershner ,1
Lana Wachowski ,1
Martin Scorsese ,2
Milos Forman ,2
Akira Kurosawa ,3
Jonathan Demme ,1
Fernando Meirelles ,1
Roberto Benigni ,1
Frank Capra ,1
George Lucas ,1
Hayao Miyazaki ,2
Bong Joon Ho ,1
Luc Besson ,1
Masaki Kobayashi ,1
Roman Polanski ,1
James Cameron ,2
Bryan Singer ,1
Alfred Hitchcock ,4
Roger Allers ,1
Charles Chaplin ,3
Tony Kaye ,1
Isao Takahata ,1
Damien Chazelle ,1
Ridley Scott ,2
Olivier Nakache ,1
Michael Curtiz ,1
Giuseppe Tornatore ,1
```



```
Olivier Nakache ,1
Michael Curtiz ,1
Giuseppe Tornatore ,1
Florian Henckel von Donnersmarck ,1
Stanley Kubrick ,4
Billy Wilder ,2
Andrew Stanton ,1
Anthony Russo ,2
Bob Persichetti ,1
Park Chan-Wook ,1
Todd Phillips ,1
Makoto Shinkai ,1
Lee Unkrich ,1
Nadine Labaki ,1
Wolfgang Petersen ,1
Rajkumar Hirani ,1
John Lasseter ,1
Sam Mendes ,1
Mel Gibson ,1
Thomas Kail ,1
Gus Van Sant ,1
Richard Marquand ,1
Elem Klimov ,1
Fritz Lang ,1
Aamir Khan ,1
Orson Welles ,1
Thomas Vinterberg ,1
Darren Aronofsky ,1
Stanley Donen ,1
Michel Gondry ,1
```

From the data in the database, we made calculations to see how many times each Director appeared in the top 100 movies from IMDB. Not all of the Directors could fit in this screenshot. On the left column you have the name of each director and on the right column you have the amount of appearances. We wanted to see which Directors were the most common.

```
Money Range in Millions,Frequency
100-200,13
200-300,31
300-400,21
400-500,12
500-600,6
600-700,3
700-800,3
800-900,3
900-1000,1
1000+,7
```

From the data in the database, we made calculations to see how many millions of dollars the top 100 movies with the highest box office typically fell. On the left column, you have the different ranges of box offices (in millions of dollars). On the right column, you have the amount of movies that were part of this range from their box office. We wanted to see where most of the movies fell under. After calculating this, we found that the majority of the top 100 movies with the highest database had a box office between $200-$300 million.

```
1   Movie Title,Rank from IMDB API,B.O. in Millions from Ultimate Movie Rankings Website
2   Amadeus,82.0,$173.50
3   American Beauty,81.0,$216.70
4   Apocalypse Now,53.0,$335.50
5   Casablanca,48.0,$344.30
6   Forrest Gump,12.0,$681.20
7   Gladiator,44.0,$293.60
8   Inception,13.0,$312.60
9   One Flew Over the Cuckoo's Nest,18.0,$448.20
10  Pulp Fiction,8.0,$223.00
11  Saving Private Ryan,25.0,$389.20
12  Schindler's List,6.0,$195.60
13  Star Wars: Episode IV - A New Hope,26.0,$1549.90
14  The Departed,45.0,$169.60
15  The Godfather,2.0,$686.30
16  The Godfather: Part II,3.0,$214.30
17  The Great Dictator,56.0,$301.10
18  The Lord of the Rings: The Fellowship of the Ring,10.0,$467.60
19  The Lord of the Rings: The Return of the King,7.0,$527.10
20  The Lord of the Rings: The Two Towers,14.0,$493.90
21  The Silence of the Lambs,21.0,$261.80
22
```

From the data in the database, we joined the Box_Office and Movies tables. We wanted to see how many movies from the Movies table was also on the Box_Office table. From here, we made a text file including all the movies that did overlap and included their rank from the IMDB API and their Box Office from ultimatemovierankings.com. There were 20 movies that overlapped.

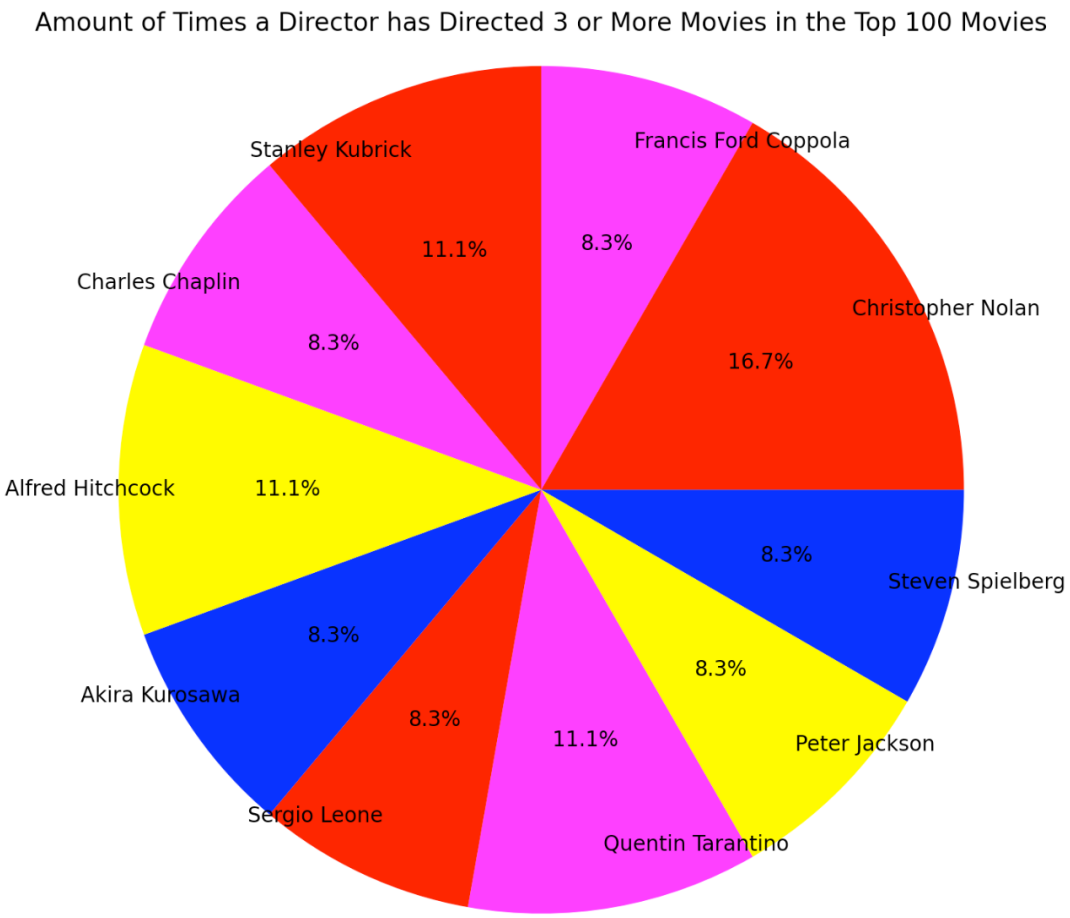
```
1   The average IMDB Rating for the top 100 movies from the IMDB API (out of 10),8.47899999999998
2
```

This file, calculations1.txt, includes the average IMDB rating out of ten that we calculated. The average is 8.479/10.

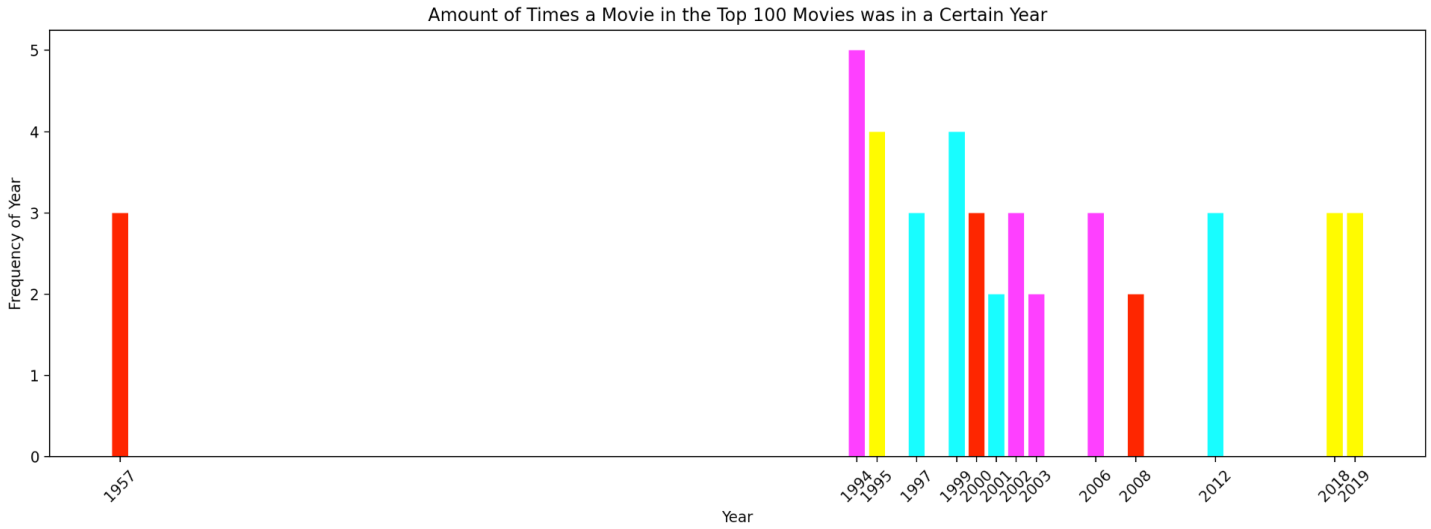
```
You, 3 minutes ago | 1 author (You)
1   The average Box Office (in millions of dollars) from Ultimate Movie Rankings top 100 movies with the highest BO,429.0760000000001
2
```

This file, calculations2.txt, includes the average Box Office (in millions) that we calculated. The average is $429.08 million.

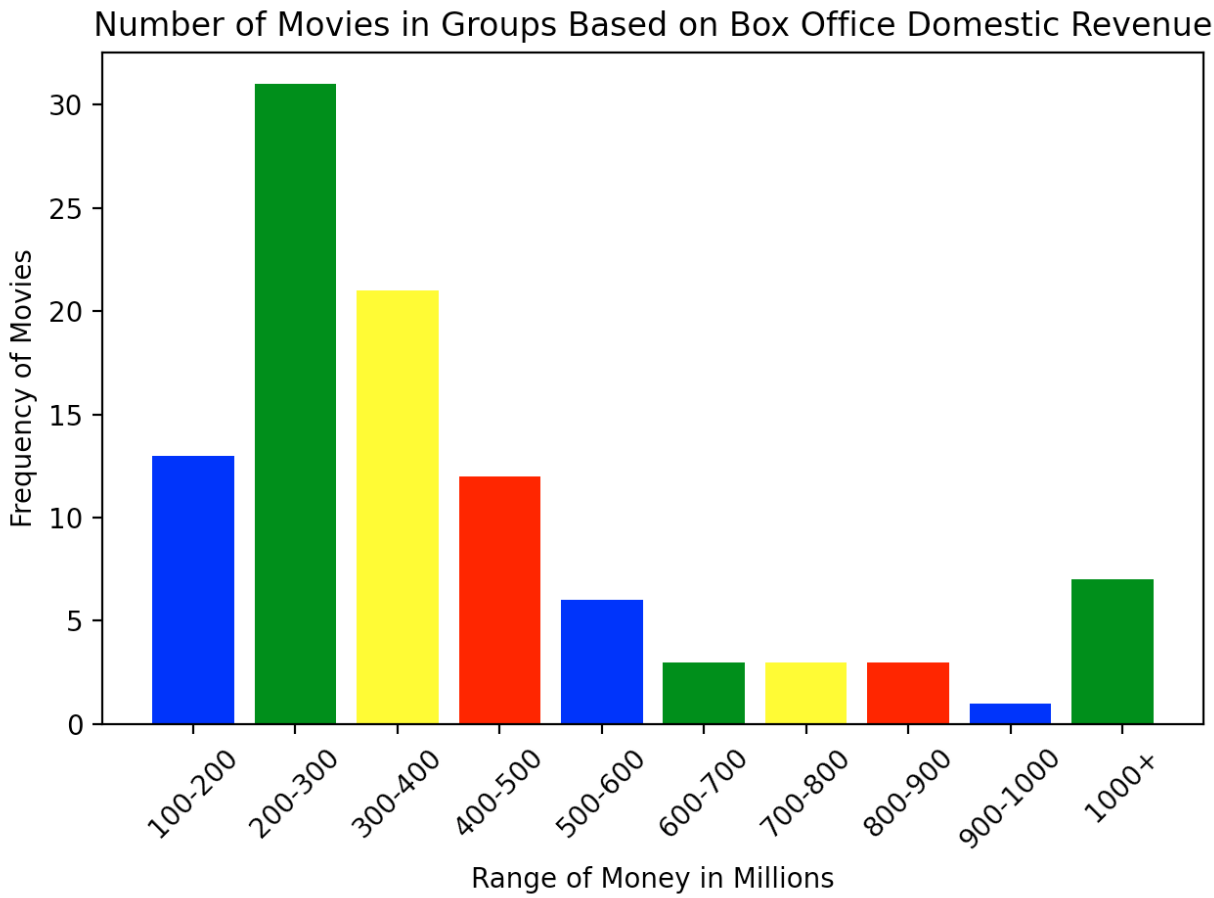
Visualizations



We gathered data on the most common directors of a movie from the IMDB API. We then examined the directors who directed three or more movies. From these directors, we made a pie chart and indicated the percentage of movies that each of these directors directed. As you can see from the pie chart, Christopher Nolan was the director to appear the most.



From the IMDB API, we gathered information about what year each movie was released. We then made a bar graph indicating which years were the most popular to have a movie be in the Top 100 movies. We only took the 14 most common years. Based on this bar graph, you can see that the most common year to have movies in the top 100 is 1994. 1994 had 5 movies make it to the top 100 movies.



From the Ultimate Movie Rankings website, we examined the top 100 movies with the highest box office (adjusted for inflation). We then made intervals of these movies where each interval ranged $100 million (except for the last interval which is just $1,000 million +). We wanted to see which interval the majority of movies fell under. Based on this graph, most of the movies with the highest box office fell in the category of $200-$300 million.

Instructions for Running the Code

1. First, you have to run imdb_api1.py four times. Each time you run this it will add 25 new entries in movies_final_project.db. In the database, there will be a table titles “Movies” and it will show you the top 100 movies according to the IMDB API. It will show you each movie and their rank, IMDB rating, year and director.
2. Next, you have to run imdb_api2.py. This will show you the pie chart of directors who were featured the most in the top 100 movies database. In addition, it will create a new table titled “Directors” in the movies_final_project.db where it will show you each director and the number of times they appeared in the top 100 movies.
3. Next, you have to run imdb_api3.py. This will create a text file called “Director_Frequency.txt” and it shows you all the directors and their appearances. Also,

running this file will show you a bar chart of the most common years featured in the top 100 movies. This will also create a file called “calculations1.txt” that has the average IMDB rating that we calculated out of 10.

4. Next, you have to run ultimateMovieRankings#1.py four times. Each time you run this, it will add 25 new entries in the movies_final_project.db. In the database, there will be a table titled “Box_Office” where it shows you the top 100 movies with the highest box office and the amount of money they generated in millions of dollars.
5. Next, you have to run ultimateMovieRankings#2.py. This will create a bar graph where it shows you different intervals of money in millions of dollars and how many times a movie fell under that interval in the top 100 movies with the highest box office. Then, there will be a text file created money_groups.txt where it shows you each money interval and the amount of times a movie fell under that interval. Also, there will be another text file called overlap.txt that shows which movies from the Box_Office table were also in the Movies table. It shows the title of the movie, their IMDB rank and their Box Office. This will also create a file called “calculations2.txt” that has the average box office that we calculated in millions of dollars.

Input and output for Each Function in the Code

IMDb Api Functions

1. Top100(key)
    a. This is in imdb_api1.py, imdb_api2.py and imdb_api3.py
    b. Inputs an api key generated by the IMDb website and returns a list of dictionaries of the first 100 items from the data. An example of a dictionary in the list is {'id': 'tt0111161', 'rank': '1', 'title': 'The Shawshank Redemption', 'fullTitle': 'The Shawshank Redemption (1994)', 'year': '1994', 'image': 'https://m.media-amazon.com/images/M/MV5BMDFkYTc0MGEtZmNhMC00ZDIzLWFmNTEtODM1ZmRlYWMwMWFmXkEyXkFqcGdeQXVyMTMxODk2OTU@._V1_UX128_CR0,3,128,176_AL_.jpg', 'crew': 'Frank Darabont (dir.), Tim Robbins, Morgan Freeman', 'imDbRating': '9.2', 'imDbRatingCount': '2503658'}

2. setUpDatabase(db_name)
    a. This is in imdb_api1.py, imdb_api2.py and imdb_api3.py
    b. Takes in the name of the database as a parameter (‘movies_final_project.db’) and sets up the database in db sq3lite

3. setUpMoviesTable(data, cur, conn)
    a. This is in imdb_api1.py

    b. Creates the movies database with the data returned from top100. It finds each variable we are searching for (title, rank, year, IMDB rating and director) and adds the information to the database. It adds 25 entries at a time so you have to run this file four times.

4. getDirectors(key)
    a. This is in imdb_api2.py and imdb_api3.py
    b. Inputs an api key generated by the IMDb website and calls upon the Top100 method. Loops through the list of dictionaries from the Top100 method and returns a list of directors from each movie in the Top 100.

5. countDirectors(directors)
    a. This is in imdb_api2.py and imdb_api3.py
    b. Inputs the list of directors that was returned from getDirectors. Returns a dictionary where the keys are all the directors and the values are the amount of times they appeared on the list.

6. setUpDirectorsTable(director_dict, cur, conn)
    a. This is in imdb_api2.py
    b. Inputs the dictionary that was returned from countDirectors. Creates a table within 'movies_final_project.db' where the left column shows you the names of directors and the right column shows you the amount of appearances they have in the top 100 movies.

7. director_pie(director_frequency)
    a. This is in imdb_api2.py
    b. Inputs the dictionary that was returned from countDirectors. Creates a pie chart that shows you all the directors in the director dictionary that have directed three or more movies. Out of all these directors, the pie chart shows you the percentage of movies they have directors from this amount of movies.

8. getAvgRating(data, filename, cur, conn)
    a. This is in imdb_api3.py
    b. Inputs the data found from the Top100 method and the table Movies in the database. Returns and calculates the average IMDB rating from the top 100 movies by using the SELECT function to find the IMDB ratings. The function also takes in filename and creates a file called calculations1.txt where it has the calculated average of the IMDB rating.

9. getTupleOfYears(data, cur, conn)

a. This is in imdb_api3.py
b. Uses SELECT to access all the movie data from the database including the year the movie was produced. Creates a dictionary where the keys are each year and the values are how many times they appeared in the database. Then, we sorted the dictionary from highest to lowest and returned the top 14 most common years. This function returns a list of 14 tuples where the first value in the tuple is the year and the second value in the tuple is how many times it appeared.

10. barchart_year_and_frequency(dictionary)
    a. This is in imdb_api3.py
    b. Takes in the list of tuples created in getTupleOfYears and creates a bar chart of the year and the amount of times a movie was in the top 100 from that year. Only shows the top 14 most common years.

11. director_freq_txt(dct, cur, conn, filename)
    a. This is in imdb_api3.py
    b. Takes in the dictionary returned from the countDirectors function. Also takes in cur, conn which will input the database information. Also takes in the filename we are about to write called "Director_Frequency.txt". Uses the data calculated from the number of appearances for directors and writes a file with this information. The left column is the director and the right column is the number of appearances.

Ultimate Movie Rankings Functions

1. getMovies()
    a. This is in ultimateMovieRankings#1.py and ultimateMovieRankings#2.py
    b. Uses the website url and Beautiful soup to return a dictionary with the keys being the movie name and the values being the money they made in millions. This dictionary shows you the top 100 movies with the highest box office.

2. setUpDatabase(db_name)
    a. This is in ultimateMovieRankings#1.py and ultimateMovieRankings#2.py
    b. Same as setting up database in IMDb api

3. setUpMoneyTable(money_per_movie_dict, cur, conn)
    a. This is in ultimateMovieRankings#1.py
    b. Uses the return value from getMovies (a dictionary of the top 100 movies with the highest box office where the key is the movie and the value is the money they made in millions) to create a table using the dictionary key value pairs. There will now be a new table in the database called "Box_Office" where the left column is

the movie and the right column is the money they made in millions of dollars. It adds 25 entries at a time so you have to run this file four times.

4. getAvgMoney(data, filename, cur, conn)
    a. This is in ultimateMovieRankings#2.py
    b. Takes in the dictionary generated from getMovies(). Also takes in cur, conn which will input the database information. Uses SELECT to get the values from the table with money, adds them all up and divides by the total number of elements in the list. It returns and calculates a float which has the average box office from the top 100 movies with the highest box office. The function also takes in filename and creates a file called calculations2.txt where it has the calculated average of the box office from the top 100 movies.

5. dctMoney(dct, cur, conn)
    a. This is in ultimateMovieRankings#2.py
    b. Takes in the dictionary created in getMovies. Also takes in cur, conn which will input the database information. Iterates through this dictionary and creates and returns a new dictionary. In this dictionary, the keys are different ranges of money in millions of dollars (for example: one key is '100-200' and another key is '200-300', etc). The value is how many times a movie from getMovies fell into this interval.

6. txtMoney(MoneyDict, filename, cur, conn)
    a. This is in ultimateMovieRankings#2.py
    b. Takes in the dictionary created from dctMoney. Also takes in cur, conn which will input the database information. Also inputs a filename "money_groups.txt" so when we run the function it will create a file. The function returns None, but the new text file shows the different money ranges on the left column in millions of dollars and the amount of times a movie fell into that interval on the right column.

7. barchart_frequency_and_money(dct)
    a. This is in ultimateMovieRankings#2.py
    b. Takes in the dictionary created from dctMoney. Creates a bar chart where the x-axis is the different money intervals and the y-axis is the amount of times a movie fell into this interval.

8. money_and_movie(cur, conn)
    a. This is in ultimateMovieRankings#2.py
    b. Uses the JOIN function to join together the data within different tables in the database. Selects the Movie column from the table Box_Office, the rank column

from the Movies table and the Money_in_Millions column from the table Box_Office. From Box_Office, joins with Movies where Box_Office.Movie is equal to Movies.title. The purpose of this was to try and find the overlap of movies that were ranked on UltimateMovieRankings.com and the IMDP API. The function returns a list of tuples where each tuple has three items: the movie title, the rank from the IMDB API and the box office in millions of dollars. This is returned sorted alphabetically by the movie title.

9. overlap_txt(sorted_data, filename)
    a. This is in ultimateMovieRankings#2.py
    b. This function has two inputs. The first input is the sorted data that was generated from the money_and_movie function. The second input is the file name we are going to write which is "overlap.txt". The function takes these two inputs and writes a file where the first column is the title of the movie, the second column is the rank of the movie according to the IMDB API and the third column is the Box Office of the movie in millions of dollars. The function returns None but it does write this new file. From the new file, there are 20 rows (excluding the header). This means that out of the Top 100 best movies ranked from the IMDB API, only 20 of them were also in the Top 100 movies ranked from ultimatemovierankings.com that had the highest box office in millions of dollars (adjusted for inflation).

Resources

Date Issue	Description	Location of Resource	Result (did it solve the issue?
12/5/21	We wanted to create a Bar Chart for our UltimateMovieRankings python file. We were stuck on how to go about this.	https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.bar.html	We used this link to help us and it definitely worked. It helped us understand the terminology of bar charts and what to fix to make all the data look nice and make sense.
12/7/21	We were having trouble getting our pie chart to show the percentages of each director.	https://matplotlib.org/stable/gallery/pie_and_polar_charts/pie_features.html	We used this link and it definitely helped. We were able to not only label each part of the pie chart with the director name but we were able to put the percentage they took of the chart.

12/7/21	We had trouble changing the colors of the bar graph and its elements	https://www.python-graph-gallery.com/3-control-color-of-barplots	We were able to create an appealing visual after looking through the examples on the website
12/7/21	We forgot how to get rid of the $ in the money without having it be a string	https://stackoverflow.com/questions/265960/best-way-to-strip-punctuation-from-a-string	We were able to get rid of the punctuation and cast the value to a float.
12/8/21	We had trouble finding the values for find_all in the Ultimate Movie Website.	https://tutorme.com/?utm_source=google&utm_medium=cpc&utm_campaign=br_tutorme&utm_term=tutor_me&utm_content=exact&utm_campaign=9211691181&utm_source=google&utm_medium=paid_search&utm_term=e_tutor%20me&adgroupid=92965763933&geoid=9016852&matchtype=e&device=c&gclid=Cj0KCQiAzMGNBhCyARIsANpUkzPEhir2Dq4N5GiaE1ymUAJOfo2ps8PgNmCWZT4JB49-xDXribXVMW0aAkddEALw_wcB	Yes, we were connected with a live tutor who helped us look through the html file to find the places where the title, director, and revenue were located.
12/8/21	We had trouble changing the values that were displayed on the pie chart as well as showing all of them	https://stackoverflow.com/questions/41088236/how-to-have-actual-values-in-matplotlib-pie-chart-displayed	We were able to display the percentages as well as the name of the director. We were also able to change the layout and colors of the chart.
12/8/21	We had trouble sorting the items in the dictionary that we wanted to add to our csv file	https://www.geeksforgeeks.org/how-to-sort-data-by-column-in-a-csv-file-in-python/	We were able to sort the dictionary by director appearances, with the most being at the top and least being at the bottom.
12/8/21	We had trouble inputting only the Director from the IMDB API into the database and excluding the rest of the cast and crew.	https://pythonspot.com/tag/sql/ And	Instead of having a problem with SQL, we realized we had to split the string at '(dir.)' and use an index of 0 to get

		https://www.w3schools.com/python/ref_string_split.asp	only the Director.