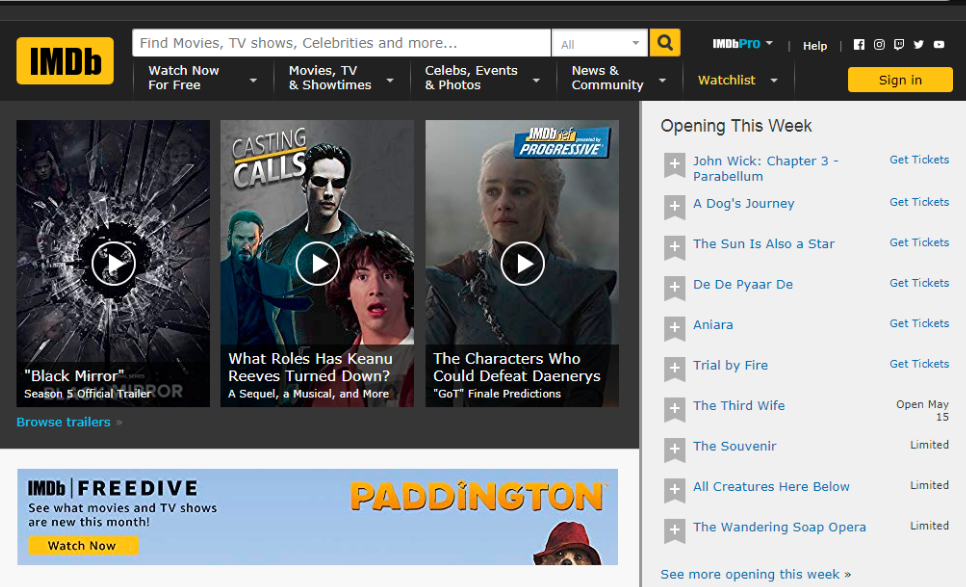
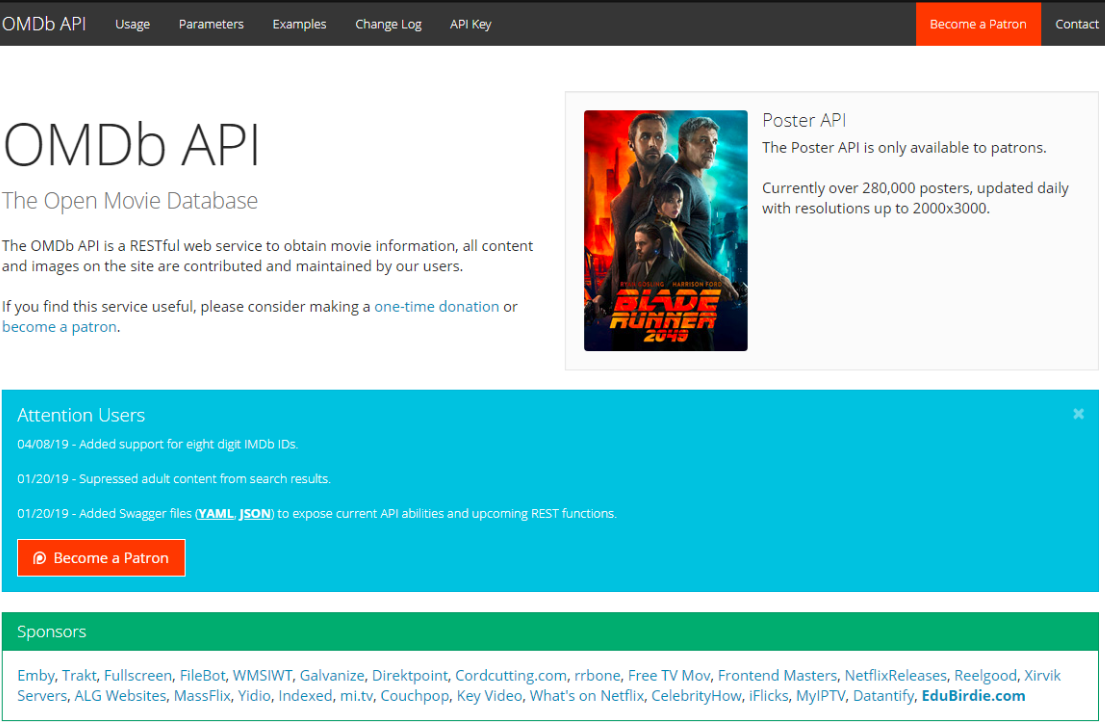
**ETL: Pam and Chris go to the movies**

Post Mortem

We picked the theme of movies because they appeal to a huge demographic, with many other forms of entertainment surrounding the industry itself.  We wanted to take a deeper look into the successes of the most popular movies of 2018 specifically. Our aim was to gather information that might give us insights and metrics to compare actor’s and director’s income from their blockbuster hits. We also wanted to include genres, parental ratings, and imdb scores to see which had an impact on the industry and possible future trends.

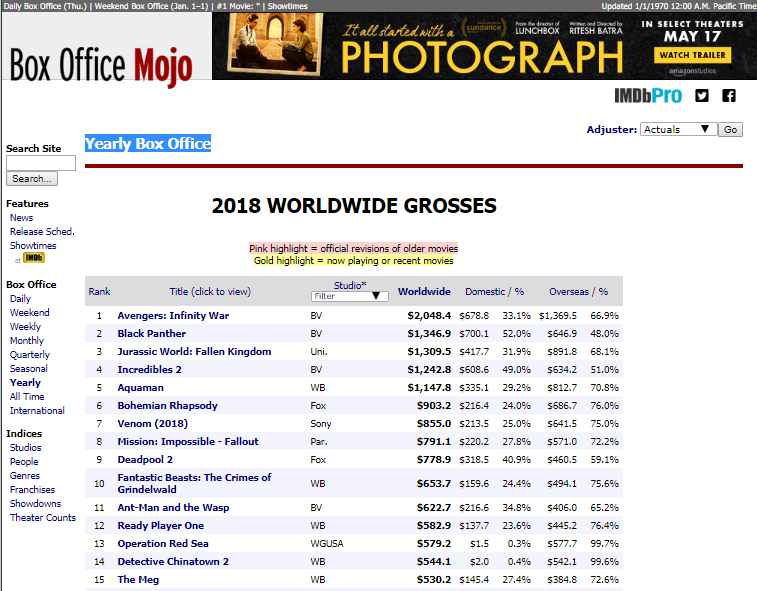
**Extraction:**

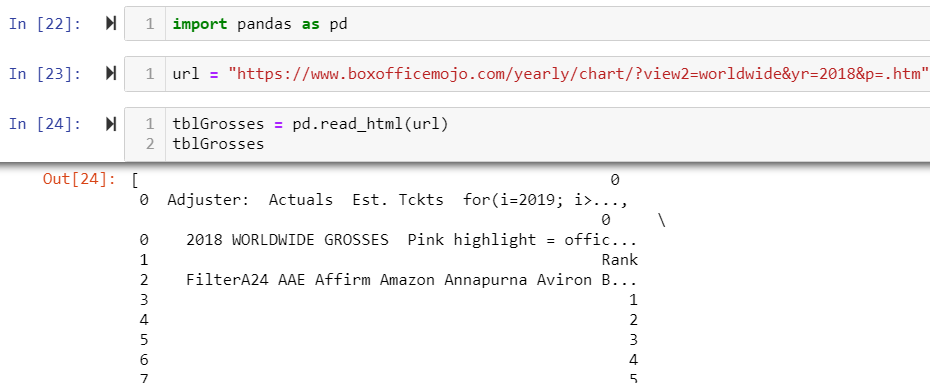
Our first task was to find a relevant dataset. I was able to find an api for the OMDB, The Open Movie Database, which describes itself as ”A RESTful web service to obtain movie information.” With the OMDB, we were able to query and return the info for (almost) any movie.  After donating to the creator’s Patreon, <https://www.patreon.com/join/omdb> , we were given a key to make 100,000 requests a day.  Now that we had access to a massive dataset, we came up with the idea to take the top grossing movies of 2018 as a limited dataset, and return additional information for them.

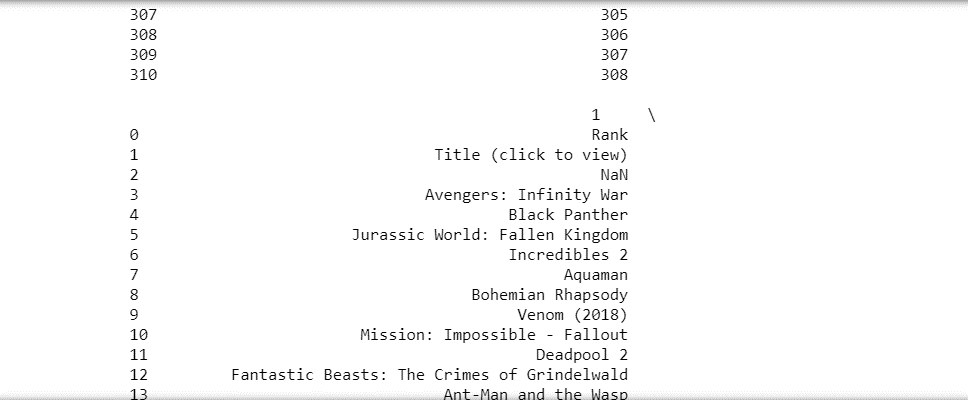
Scrape Box Office Mojo webpage – Pam Zhao

<https://www.boxofficemojo.com/yearly/chart/?view2=worldwide&yr=2018&p=.htm>

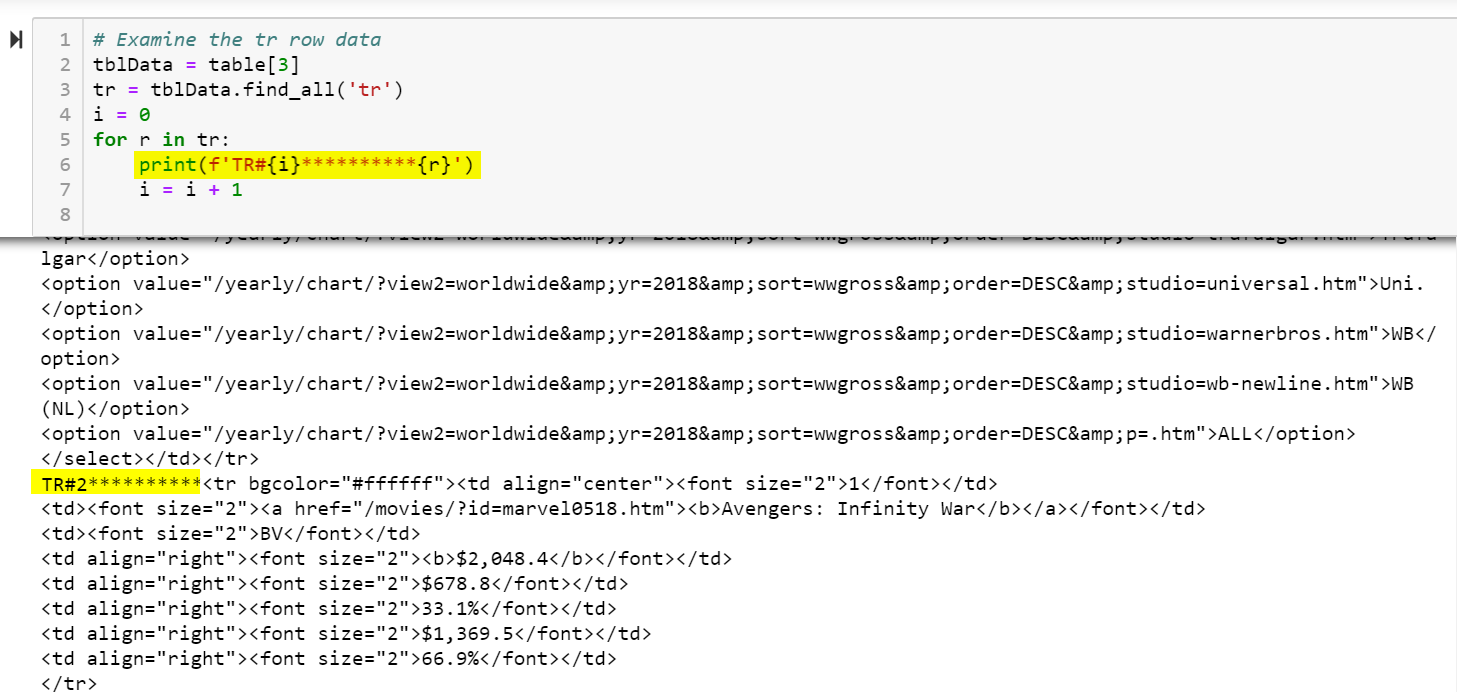


When I saw the table on the website (see above screenshot), I thought it would be very easy to scrape the table by using Panda scraping because there is only one big table in the webpage. However, the format was off a lot when I used Pandas scraping.





Then I inspected the HTML for the webpage and decided to use BeautifulSoup to scrape the table. There are more than one table on the webpage, and I was able to locate the table that has the movies grosses data by going through each table data. I examined the table by printing out the table row number in order to determine where to start to get the data.

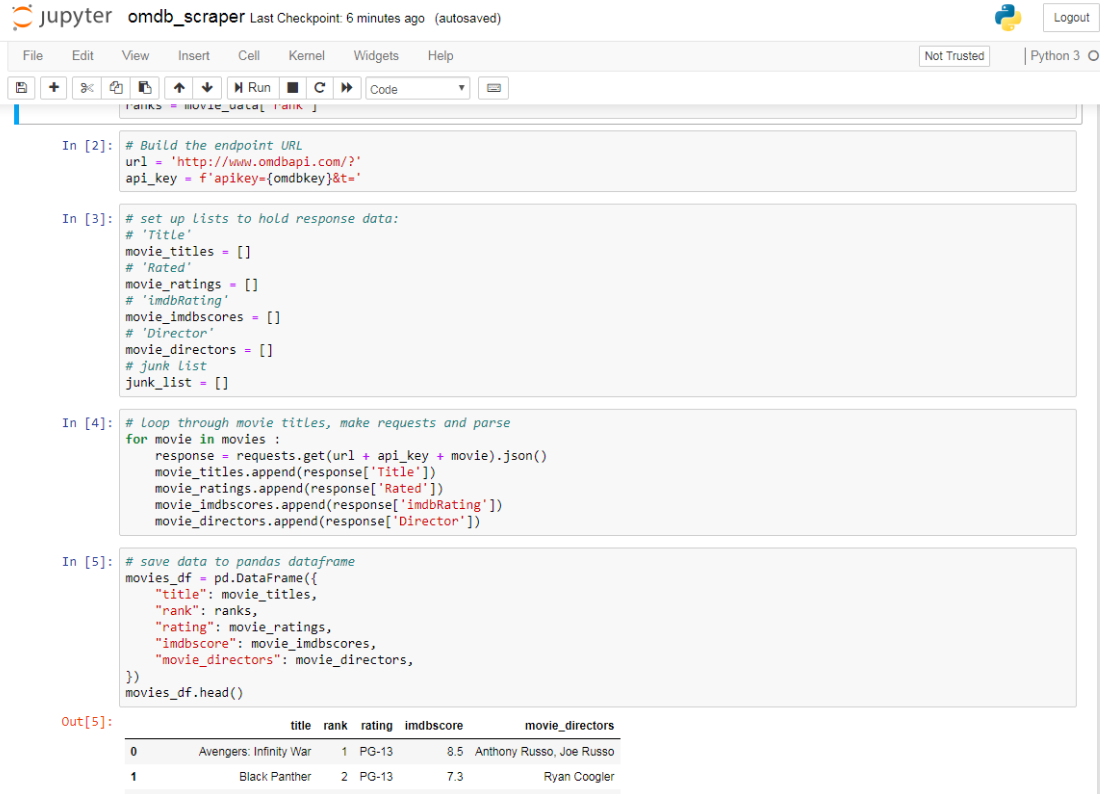


I iterated through the html table and append the data into a Pandas DataFrame. Then I saved the DataFrame to a CSV file. My teammate, Chris, used my list of movies title to pull the data from OMBD API.

After scraping the data, we put them into DataFrame and exported them into csvs. We then shared each other’s Github repos to make sharing the files easier, and we uploaded our work. We took our exported files and merged them, having the data easily available in MySQL.

**Transform Data**

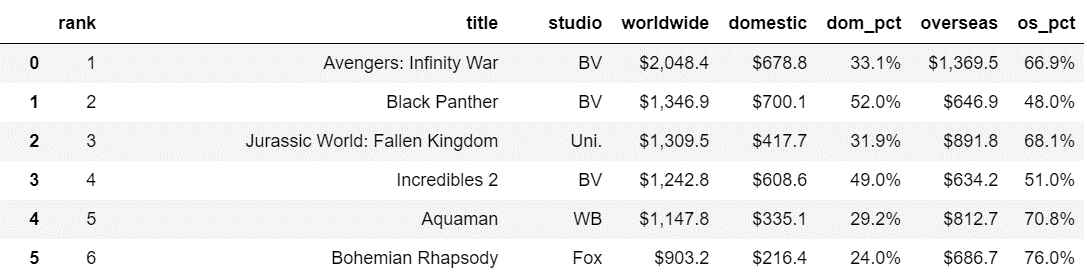
After Pam sent over the exported csv she created from scraping her site, I tried making a list from the titles in her dataset. I ran into the problem with the naming of the films, because I would constantly get an error about how a film wasn’t found. To solve this I went into Pam’s data and cleaned up the names, searching for each individually in imdb and copying over a corrected title for the OMDB API to search with. I had to delete a few films from the list as they were not in IMDB and would cause errors.



Transform Data-Pam Zhao

When I scraped the data from the webpage, all the data types are objects. I spent some time converted the gross from object to float.

This is original data:



This is the data after transform:



**Load Data**

Load Data-Pam Zhao

I loaded my DataFrame and Chris DataFrame into the MySQL database. We keep the movie rank as the link for two tables. I created a view by inner joining two table together. The reason that we use MySQL is that we have the structure for the tables so it easy to query.

