Stage-4

Merging Matched Data and Performing Analysis for Insights

Manjunath Shettar (shettar@wisc.edu)

Samhith Venkatesh (svenkatesh5@wisc.edu)

Jayashankar Tekkedatha (tekkedatha@wisc.edu)

In stage-3 we did entity matching for 'movies' between two websites Imdb [3050 Movies] and Metacritic [3000 Movies] dataset.

- Total Matched tuples were: **776**
- Now comes the part of Merging Matched Data and create the Table E, with following schema:

SCHEMA of TABLE - E
Name [Text]
Release Year [Number]
Certificate [Text]
Runtime [Number]
Genre [Text]
Director [Text]
Gross [Number]
Actors [Text]
Production House [Text]

- Matched CSV file having combined data of both tables: *matched_data.csv*
- We started stage 4 with matched_data.csv, which had combined data of both Imdb [columns appended with '_x'] and Metacritic [colomns appended with '_y'] symbolizing left and right table to pick the required tuples. So as create merged Table E.
- Now we wrote the Python Script to go through each tuple in both Table A and Table B and pick the appropriate data for Table E [as per the above schema].
- **MERGING**: Logic for choosing each schema attribute:
  - **Name**: We selected the longest movie name, as they provide more info pertaining to the movie. E.g. movie sequel, movie title's subtext etc.

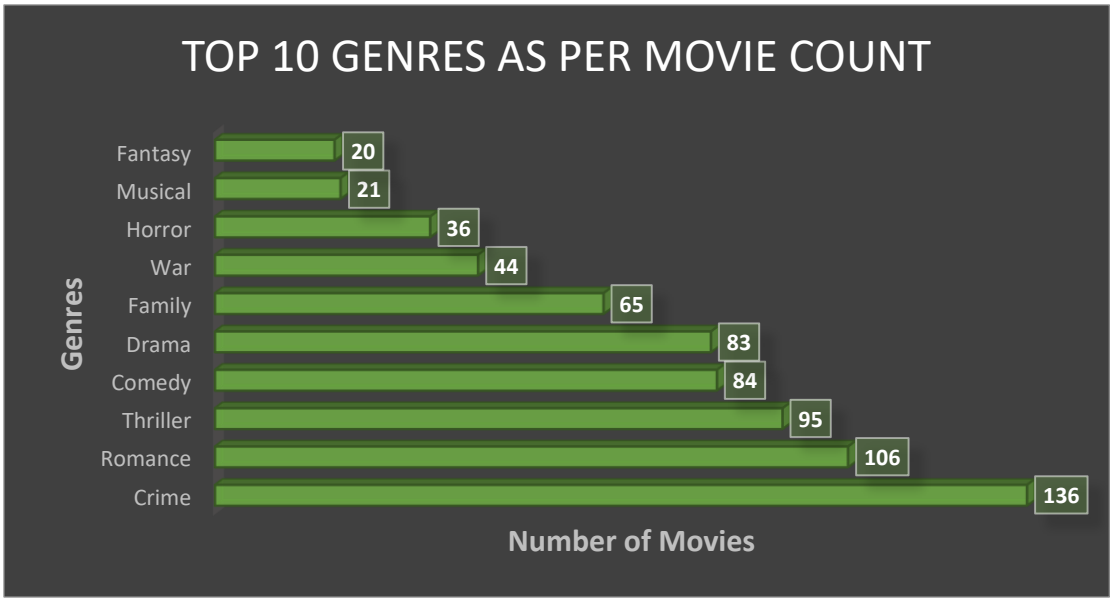
- **Release Year**: As movies get released at different time in different countries, so we chose the latest release year. To have a common convention for release year, as Imdb and Metacritic can have different convention.
- **Movie Certificate:** we observed in most occasion both website had same certificate info, but conflict arise when one website had some missing info or rating unavailable. So, we chose the smallest length certificate because certificate the usually 3-4 characters long. [PG-13, R, G]
- **Runtime:** Taking the convention we followed with release year, we chose the longer runtime as a convention, as today movies has post-credit scenes etc. **Runtime is in minutes**.
- **Genre:** We favored Imdb Genre classification over Metacritic as Imdb classified movies into one single genre whereas Metacritic associated multiple genre with one movie. This helped in doing better group by data analysis.
- **Director:** We chose longest string match for director name because we wanted to avoid any abbreviations and get data for movies directed by multiple persons.
- **Gross:** As box office collections we only collected from Imdb website, so directly chose from Table A. **The box office gross is in USD**.
- **Actors**: Again, keeping the theme of longest string match for all names, we chose the longest string among the both table to incorporate all the artists pertaining to that movie.
- **Production House**: Production house was only collected from Metacritic website, so we directly chose from Table B.

- **Note:**
  - We had to do some processing on certain attribute data to help with data analysis.
    - Movie Runtime datatype had to changed to Integer, as we had to perform operation on the runtime values later.
    - Initially actor names were separated with '_' delimiter and we processed them individually to perform Roll Up operations and analyze how successful are movies related to certain actors.
- Merged CSV file having Table E data: *merged_data.csv.*

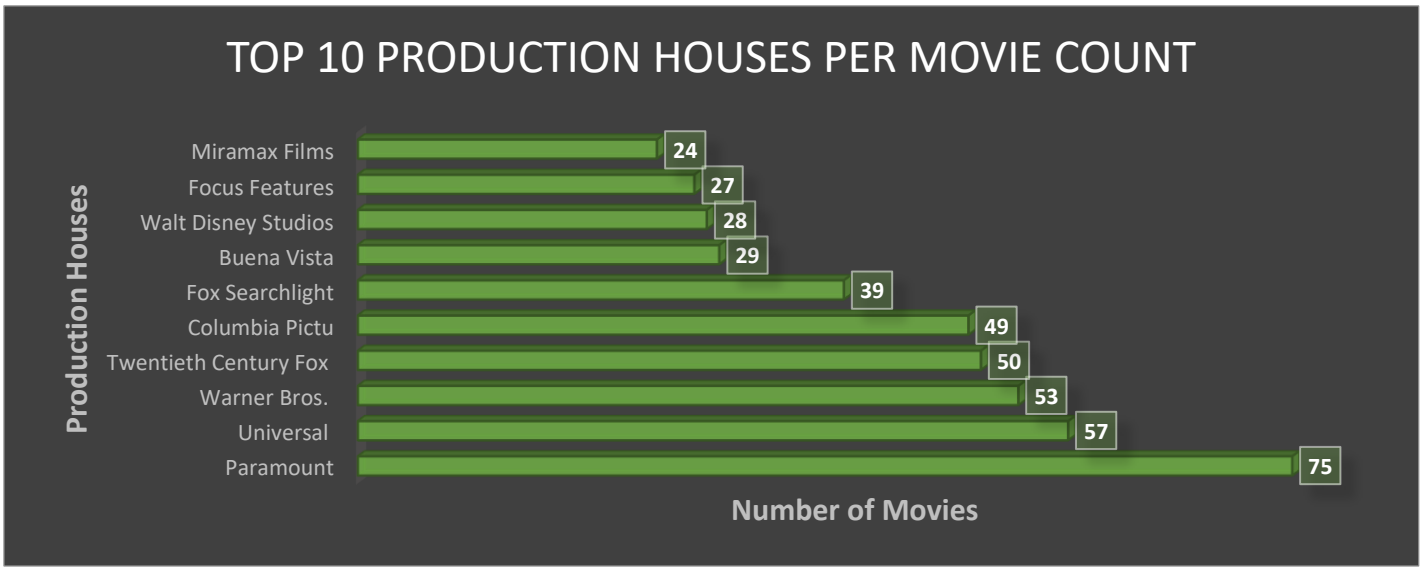
name	release_year	certificate	genre	director	actors	runtime	gross	production_house
Barton Fink	1991	R	Crime	_Ethan Coen_Joel Coen	_John Turturro_John Goodman_Judy Davis_Michael Lerner	116	6153939	Twentieth Century Fox Film Corporation
Toy Story	1995	G	Family	John Lasseter	_Tom Hanks_Tim Allen_Don Rickles_Jim Varney	81	191796233	Buena Vista Pictures
Twister	1996	PG-13	Thriller	Jan de Bont	_Helen Hunt_Bill Paxton_Cary Elwes_Jami Gertz	113	241721524	Warner Bros. Pictures

Sample Tuples form merged_data.csv

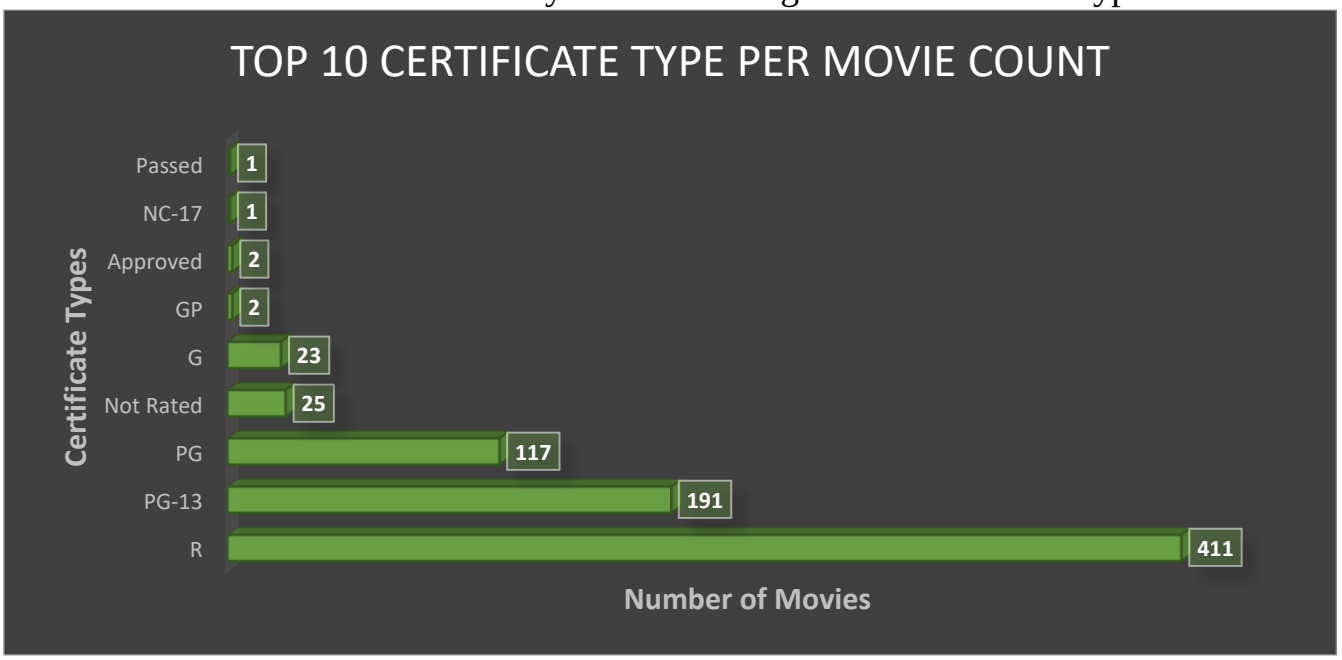
- **Perform Analysis and Obtaining Insights:** Operations performed:-
  - Roll Up
  - Drill Down
  - Slice and Dice
- Roll Up: Obtaining Movie Count along one dimension.
- 1st Dimension: **Movie Count.** We wrote our own Python function "**returnCount**" to calculate number of movies related to each attribute type.
  1) Of all the movie matched how many of them belong which genre, to get top movie genres



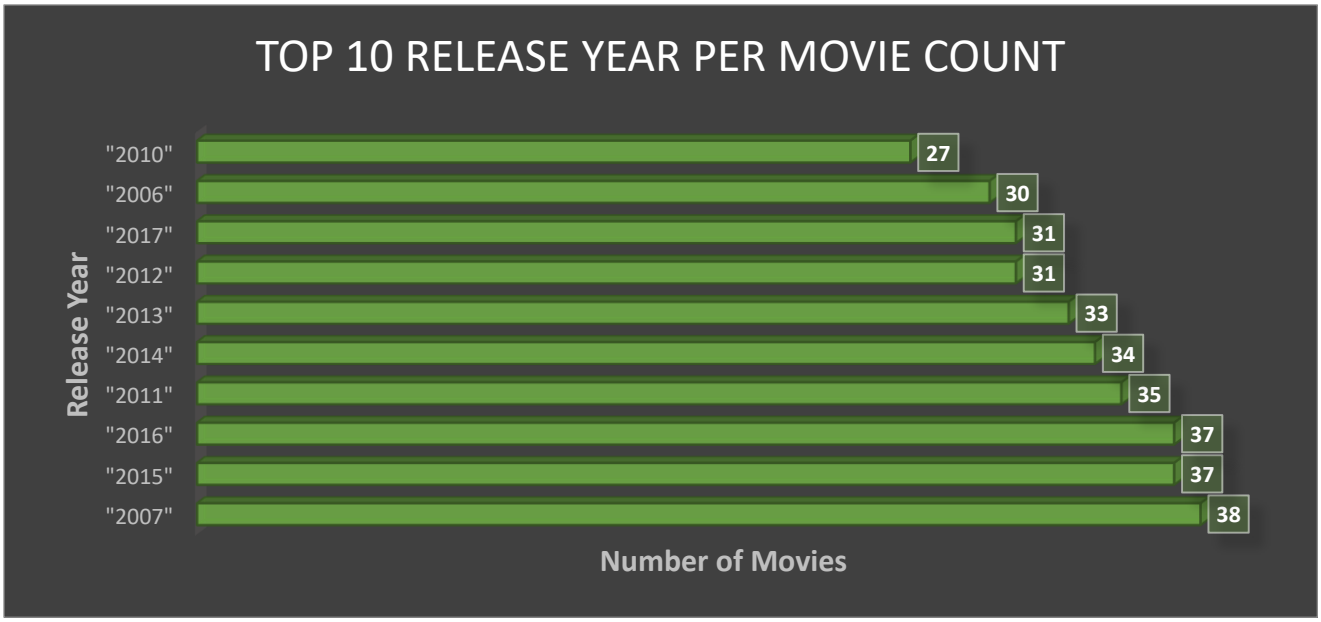
  2) Of all the movie matched how many of them belong to each production house. To get top movie producing companies.



3) Of all the movies matched how many of them belong to each certificate type.

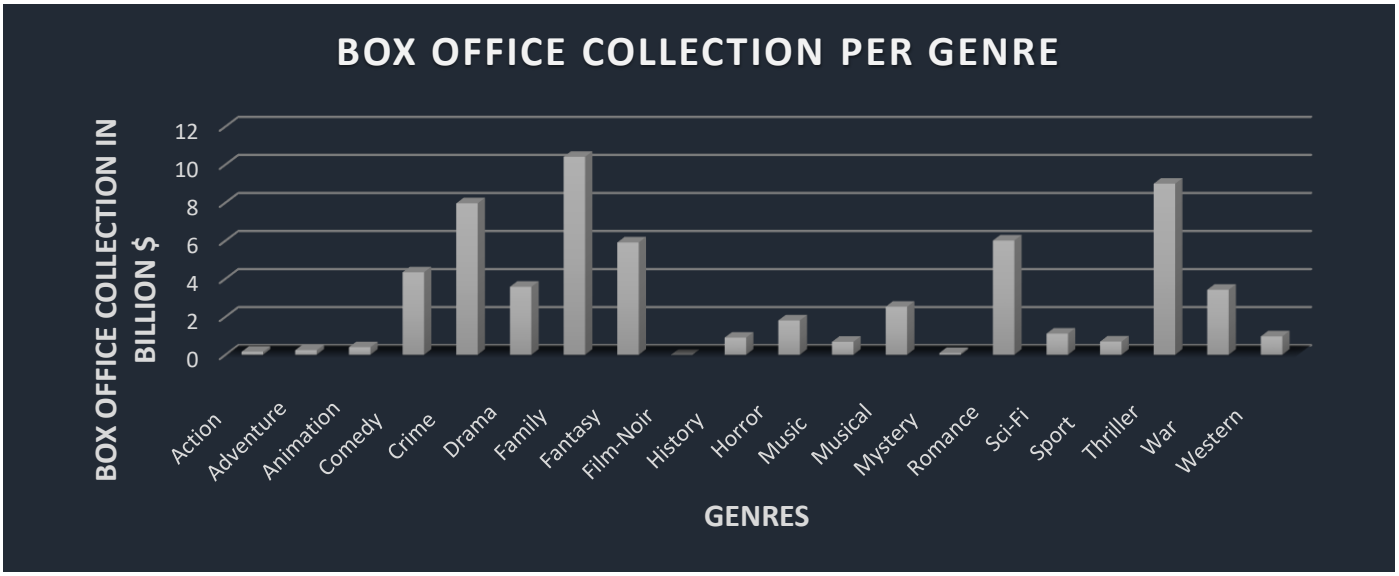


4) Of all the movies matched how many of them belong each release year.

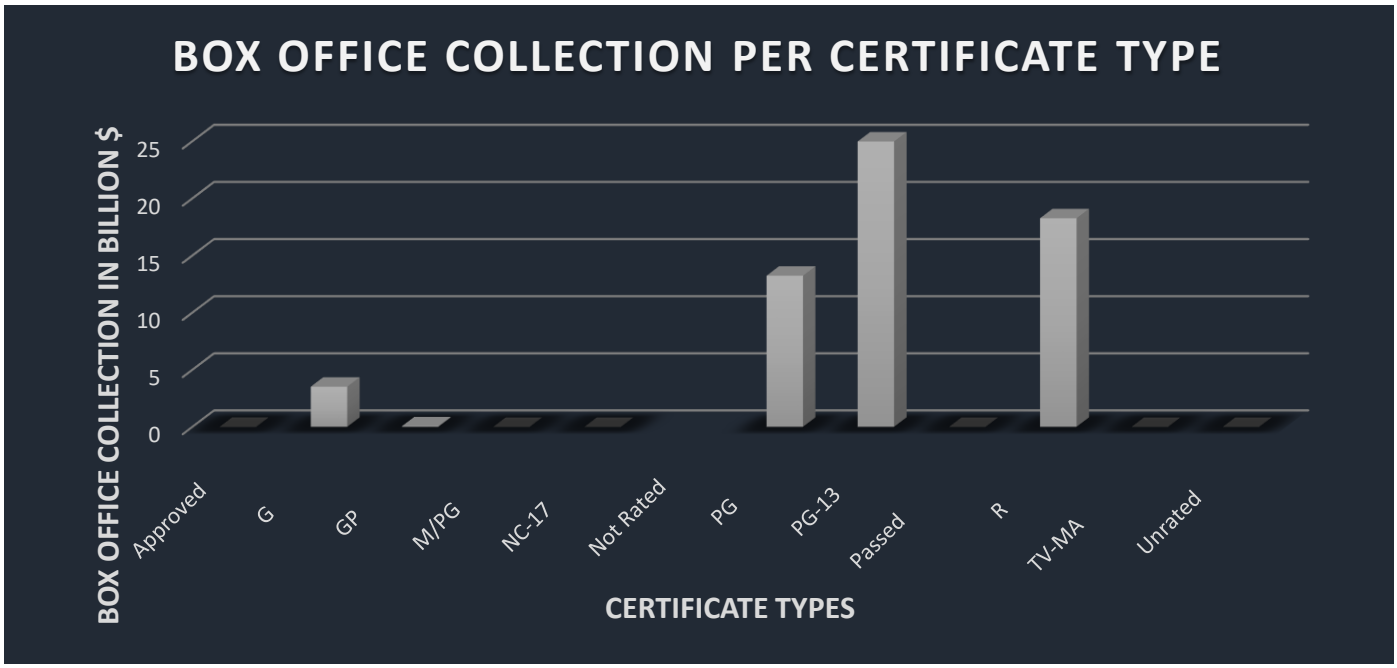


- 2nd Dimension: **Box Office Gross Collection.** We used **Pandas GroupBy** operation to group the data by Genre and Certificate and Calculating sum() over box office gross.

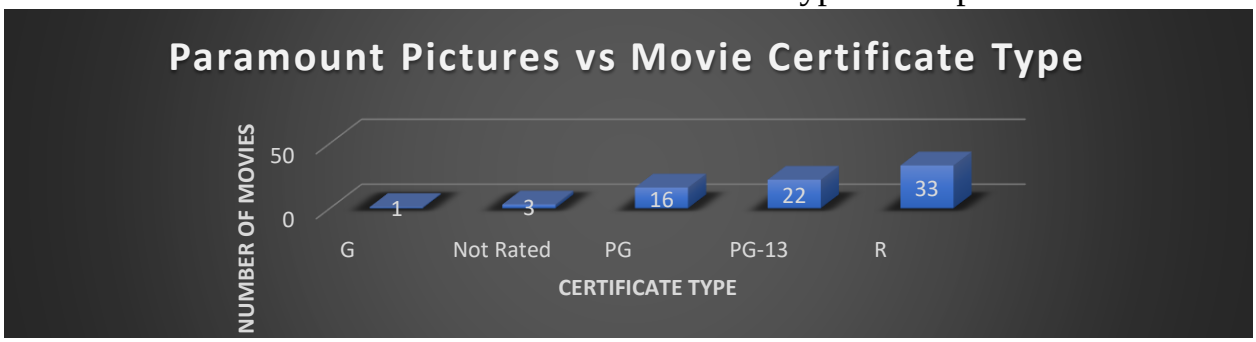
1) Aggregated Box Office Collection of each Genre.

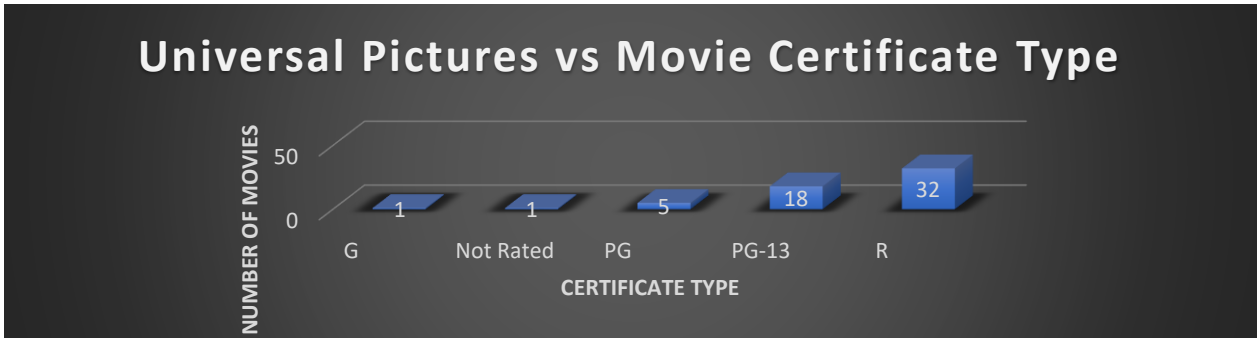


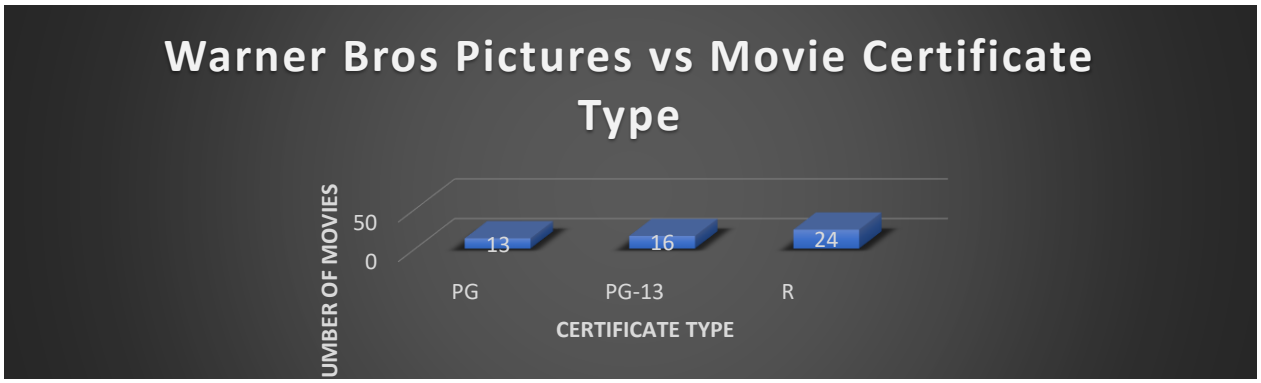
2) Aggregated Box Office Collection of each Certificate Type.



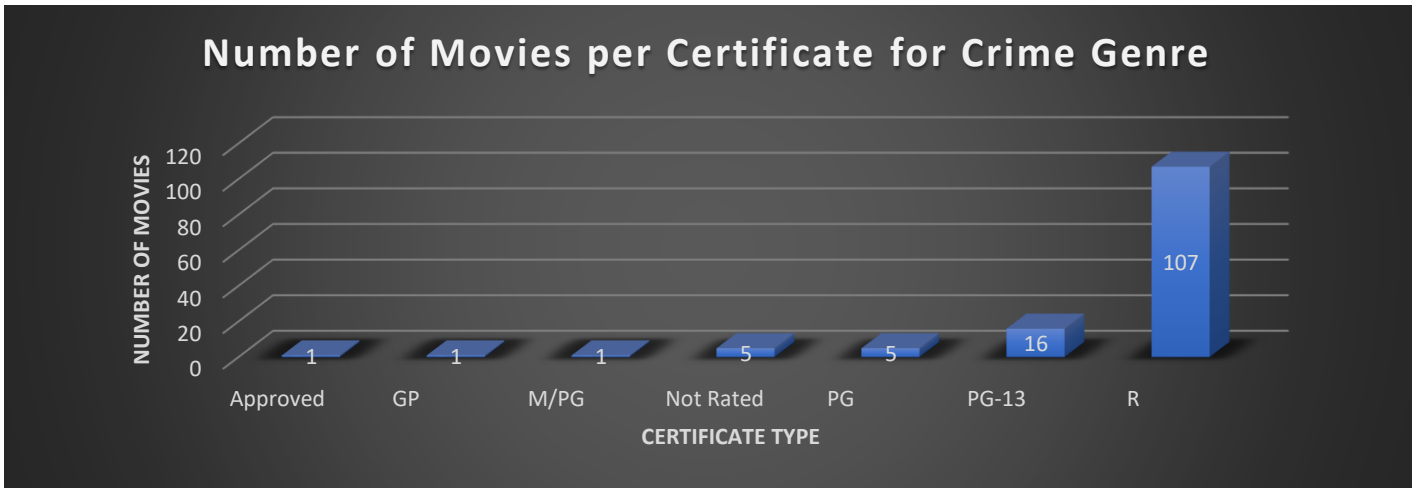
- Roll Up Analysis:
  - **Most Matched movies are produced by Movie Production Houses** such as Paramount, Warner Bros, Walt Disney etc.
  - Also, most movies matched are for Mature Audience i.e. 18 years and above because R-rated movies are highest.
  - And even most common genre are Crime and Thriller.
  - Most of **the movies matched are released in the past decade i.e. 2007-2017**.
  - On the other hand, box office collection tells a different story.
  - Even though **most matched movies are for mature audiences, but highest box office collection is observed for movies of Family genre**.
- Now we, Drill Down and perform further analysis by combining multiple dimensions.
  1) Drill Down on number of movies of each certificate type for Top 3 Prod. Houses.

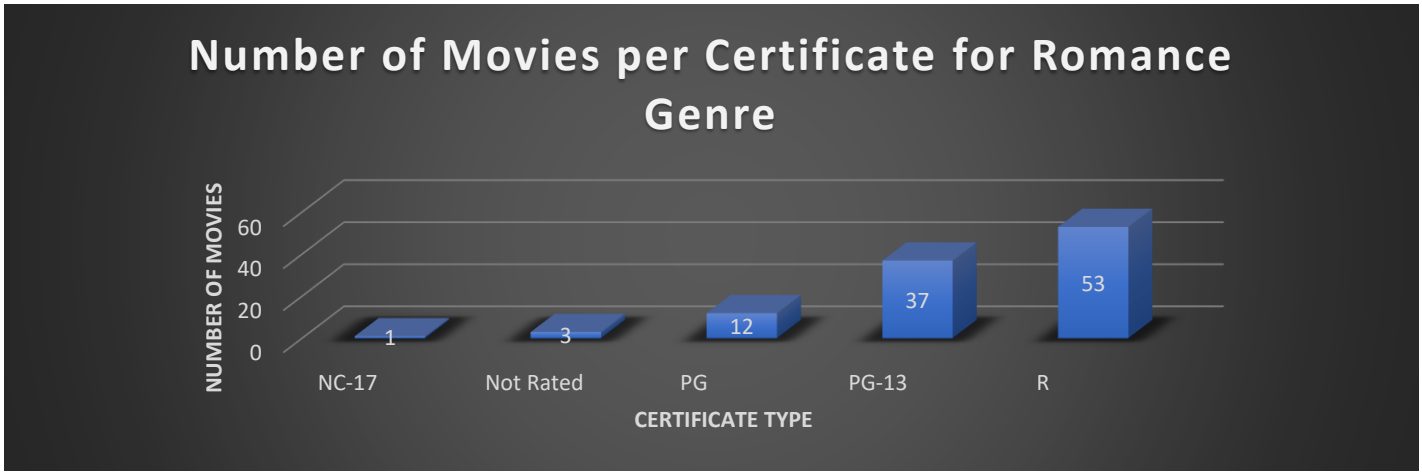


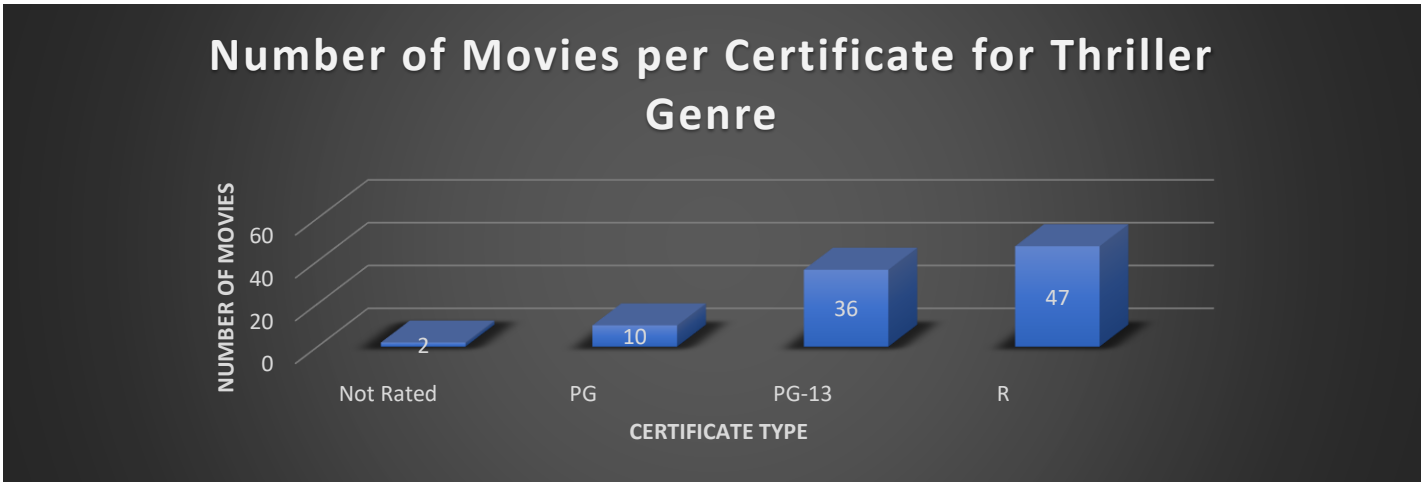




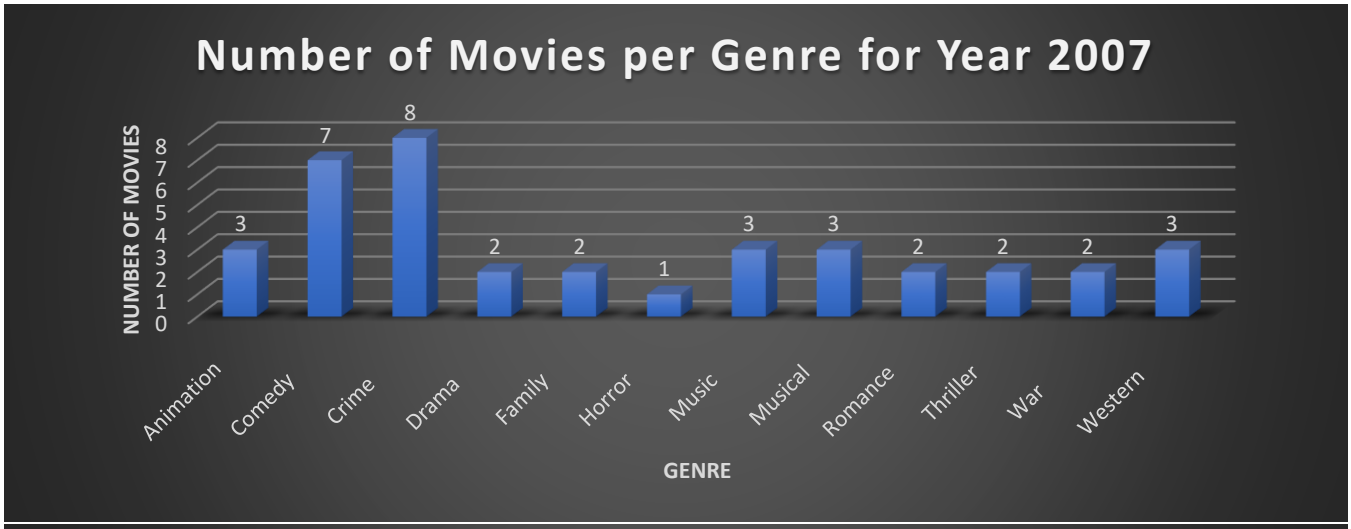
2) Here we used Groupby operation over genre and certificate attribute to Drill Down and check number of movies of each certificate type for Top 3 Genres.

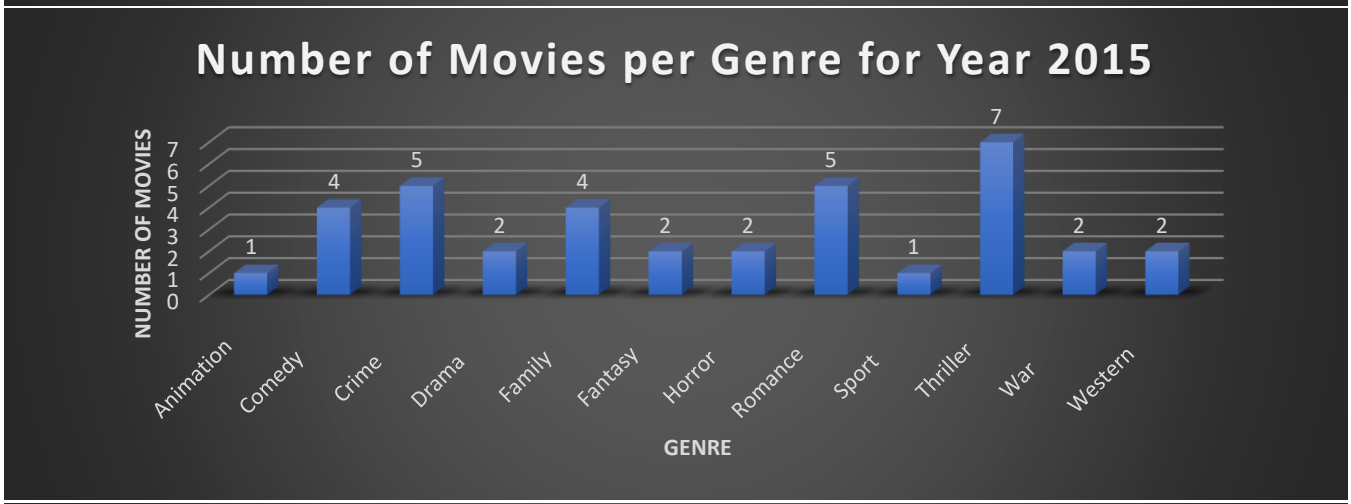


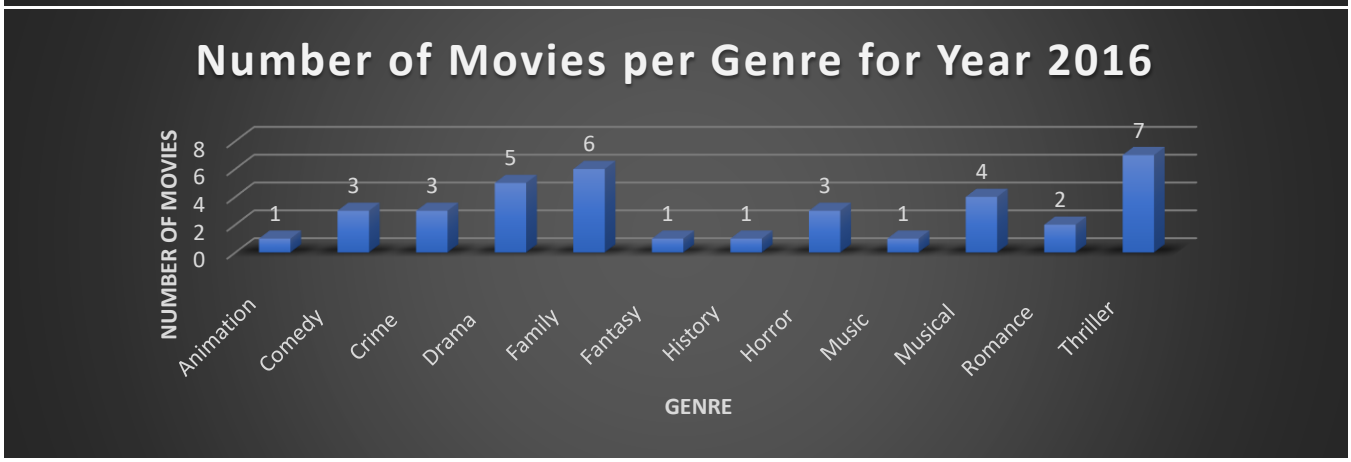




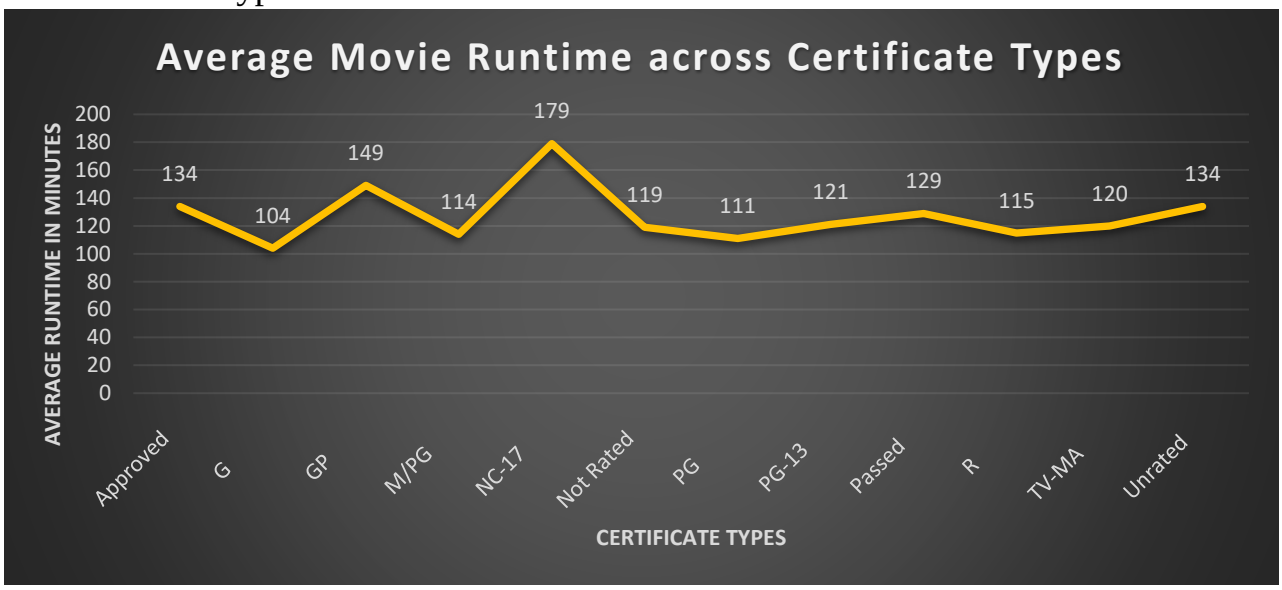
3) Here we used Groupby operation over release_year and genre attribute to Drill Down and check number of movies of each Genre type for Top 3 Release Year with most movies.

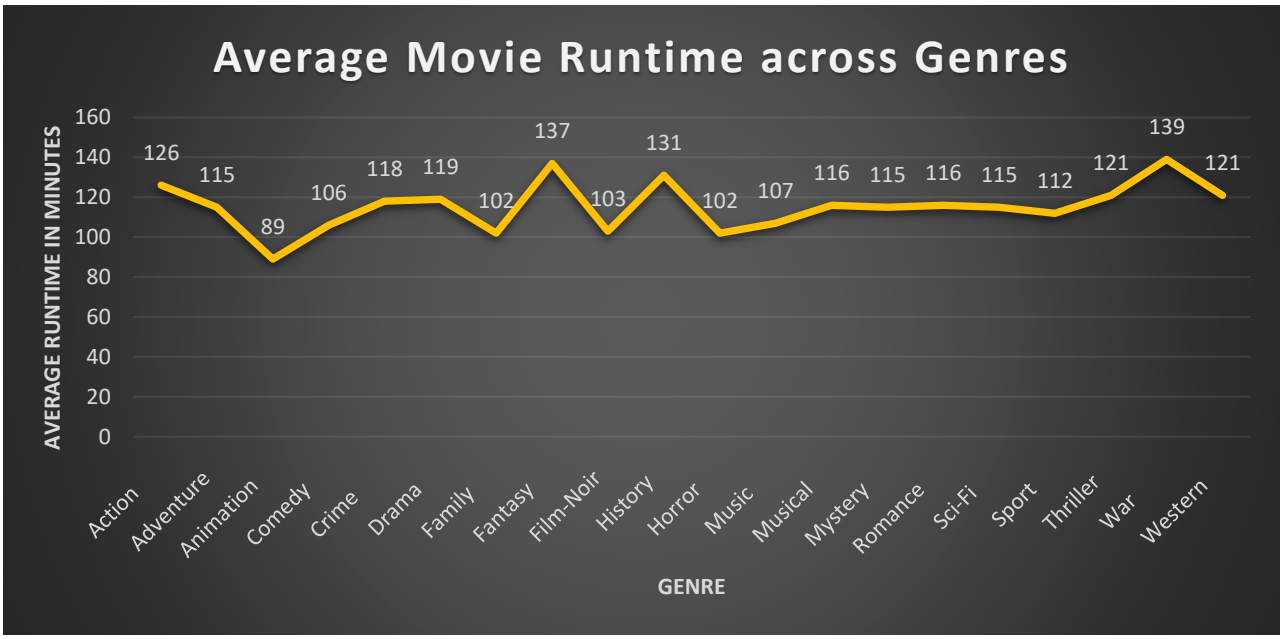






4) Here we used Groupby over genre attribute and performed the mean operation over runtime values to Drill Down and calculate average runtime of movies for each genre and certificate type.





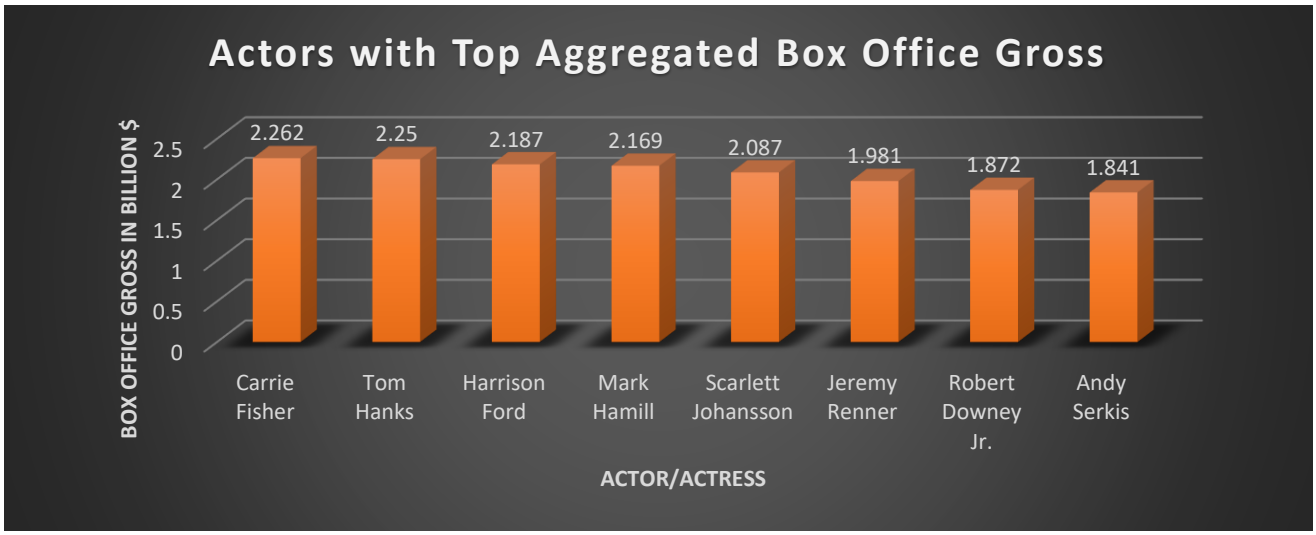
- Drill Down Analysis: After incorporating 2 or more dimensions such as genre, certificate type, production house, runtime etc. we observe the following:
  - Of the matched movies top production houses **have more movies for mature audience**. As we see the count of R-rated movies are highest for top production

houses. Same is the case with Top 3 Genres Crime, Romance which have R & PG-13 certification in the merged dataset.

- The release year show a different trend: the top 2 release years 2007 and 2015 have more movies of crime genre but **it's interesting to observe that 2016 had more Family genre movie releases than crime genre movies.**
- Also, **we see more variety of certification given to Crime genre** movies (seven different certifications given) but genre such as Romance & Thriller have four or five types of certifications.
- From runtime analysis, we observe that **most matched movies have runtime of around 2 Hours** (120 mins).
- We also see documentary type **movies such as of History & War Genre have the highest runtime**.

- Lastly, we wanted to do some broad analysis. Hence, we did Slice and Dice operation:

I. Most successful Actor/Actress. Using our custom function, we stored unique actors/actresses and calculated which artist's collective work has most box office gross:



- Its interesting to note that most of actor/actress are in one of the biggest Movie Franchises:
  - Carrie Fisher, Harrison Ford, Mark Hamill → Star Wars Trilogy
  - Scarlett Johansson, Robert Downey Jr, Jeremy Renner → Avengers Trilogy
  - Andy Serkis → Lord of the Rings Trilogy
- Another interesting thing to note is Tom Hanks is not involved in any of the major movie franchises, yet his movies aggregated box office is among the highest of the matched movies.

II. **Most Successful Movies:** Using DataFrames sort_values function we analyzed the highest grossing matched movies[their certificate type and production house].

name	gross	certificate	production_house
Star Wars: Episode VII - The Force Awakens	936662225	PG-13	Walt Disney Studios Motion Pictures
Avatar	760507625	PG-13	Twentieth Century Fox Film Corporation
Titanic	659325379	PG-13	Paramount Pictures
The Avengers	623357910	PG-13	Walt Disney Studios Motion Pictures
Star Wars: Episode VIII - The Last Jedi	619845623	PG-13	Walt Disney Studios Motion Pictures
Black Panther	612103128	PG-13	Walt Disney Studios Motion Pictures
The Dark Knight	534858444	PG-13	Warner Bros. Pictures
Finding Dory	486295561	PG	Walt Disney Studios Motion Pictures
Avengers: Age of Ultron	459005868	PG-13	Walt Disney Studios Motion Pictures
The Dark Knight Rises	448139099	PG-13	Warner Bros. Pictures

- **Problem/Issues Faced:**
  - Missing data: Tuples were checked for empty and None values.
  - For tuples with missing data we had to incorporate certain flags such as NA or 0.
  - As mentioned before while calculating average runtime for each genre and certificate type, the attribute datatype had to be transformed from string to numeric.
  - Also, calculating average runtime with movies with missing runtime value had to be managed properly.

- **Future Work**
  - If we had more time we would have collected more data (more than 3000) and performed deeper analysis
  - Also, there is more to learn about, how to scale operations on a larger data set.

- Python code for Merging Data from Table A and B to form Table E:

```
#Create column lists for creating new merged DataFrames
name = []
release_year = []
certificate = []
runtime = []
genre = []
director = []
gross = []
actors = []
production_house = []

actorsSet = set()
directorsSet = set()

#Loop over every row of the matched table to select the appropriate field for the merged
columns
for index, row in matched.iterrows():

    #Name: Select longest length name
    name1 = row['name_x']
    name2 = row['name_y']
    if(len(name1)>len(name2)):
        name.append(name1)
    else:
        name.append(name2)

    #Release Year: Select latest year
    year1 = row['release_year_x']
    year2 = row['release_year_y']
    if(year1>year2):
        release_year.append(year1)
    else:
        release_year.append(year2)

    #Certificate: Select smallest length certificate
    cert1 = row['certificate_x']
    cert2 = row['certificate_x']
    if(len(cert1)>len(cert2)):
        certificate.append(cert1)
    else:
        certificate.append(cert2)

    #Runtime: Select longest runtime
    run1 = row['runtime_x']
    run2 = row['runtime_y']
    if(run1>run2):
        if run1 == 'None':
            run1 = 0
        runtime.append(int(run1))
    else:
        if run2 == 'None':
            run2 = 0
        runtime.append(int(run2))

    #Genre: Select smaller length genre
    gen1 = row['genre_x']
    gen2 = row['genre_y']
    if(len(gen1)<len(gen2)):
        if '_' in gen1:
            gen1 = gen1.split('_')[1]
```

```
    genre.append(gen1)
else:
    if '_' in gen2:
        gen2 = gen2.split('_')[1]
    genre.append(gen2)

#Director: Select longest length director
dir1 = row['director_x']
dir2 = row['director_y']
if(len(dir1)>len(dir2)):
    director.append(dir1)
else:
    director.append(dir2)

#Gross: There is only one
gross.append(row['gross_x'])

#Actors: Select longest length actor
act1 = row['actors_x']
act2 = row['actors_y']
if(len(act1)>len(act2)):
    actors.append(act1)
else:
    actors.append(act2)

#Production house: There is only one
production_house.append(row['production_house'])
```

Conclusion:

- The whole project provided an insight in the world of data analysis: starting from data acquisition, transformation and performing various operations to obtain knowledge from raw data.
- Learn to use various tools and modules such as Pandas, Scikit learn, Magellan.