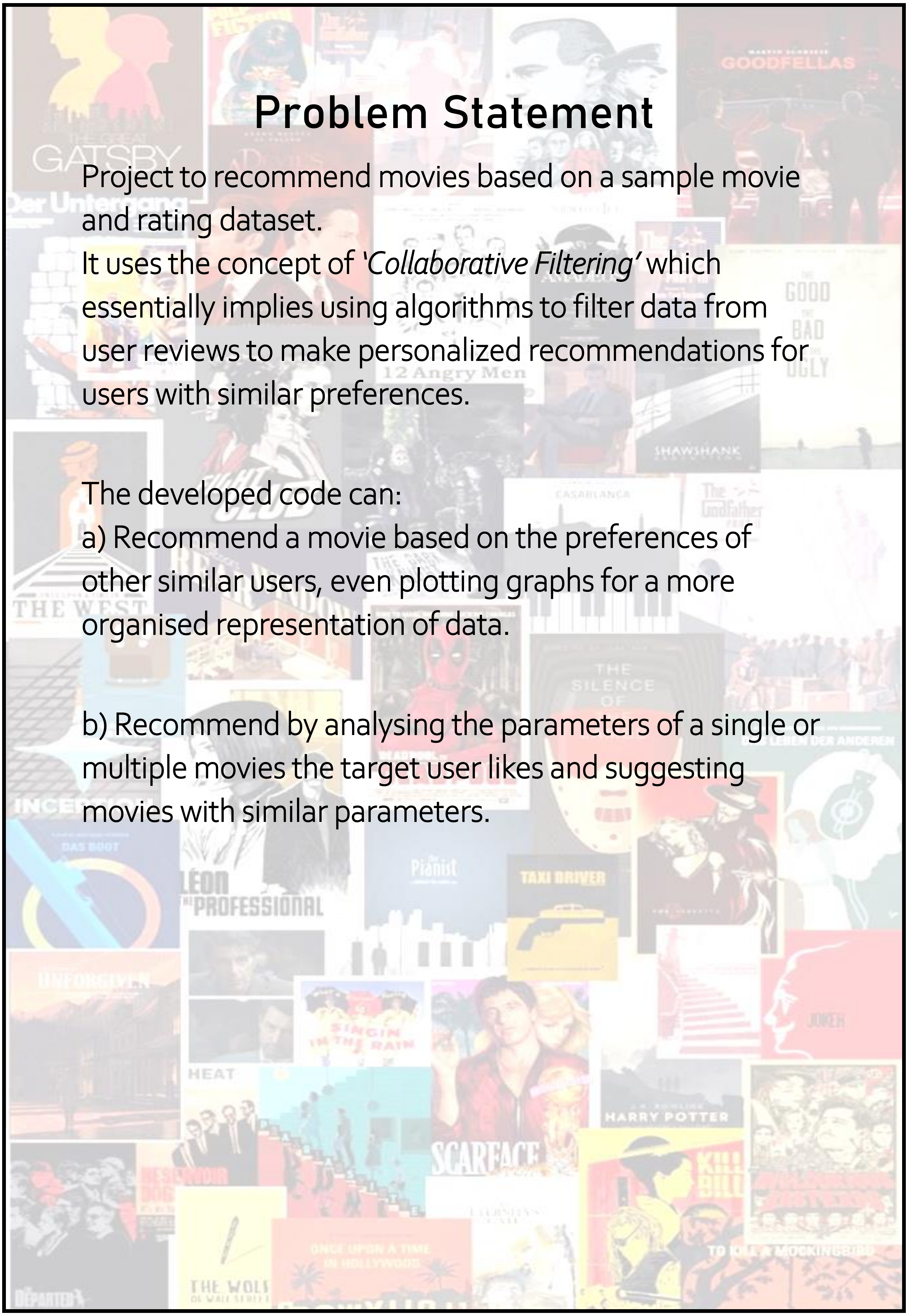


Problem Statement

Project to recommend movies based on a sample movie and rating dataset.
It uses the concept of *'Collaborative Filtering'* which essentially implies using algorithms to filter data from user reviews to make personalized recommendations for users with similar preferences.

The developed code can:
a) Recommend a movie based on the preferences of other similar users, even plotting graphs for a more organised representation of data.

b) Recommend by analysing the parameters of a single or multiple movies the target user likes and suggesting movies with similar parameters.

Procedure

I) User-User Based Filtering (parameters analysed using: ratings, count)

The code begins with inputting, by reading, both the .csv files. Modifying both these files by dropping columns which aren't useful to us and merge both these files on their common and unique column 'movieId' gives a third, new dataframe.

Then extracting the mean ratings and count of votes for each movie, the two primary parameters, one can display the top 5 results to the user for easy choosing.

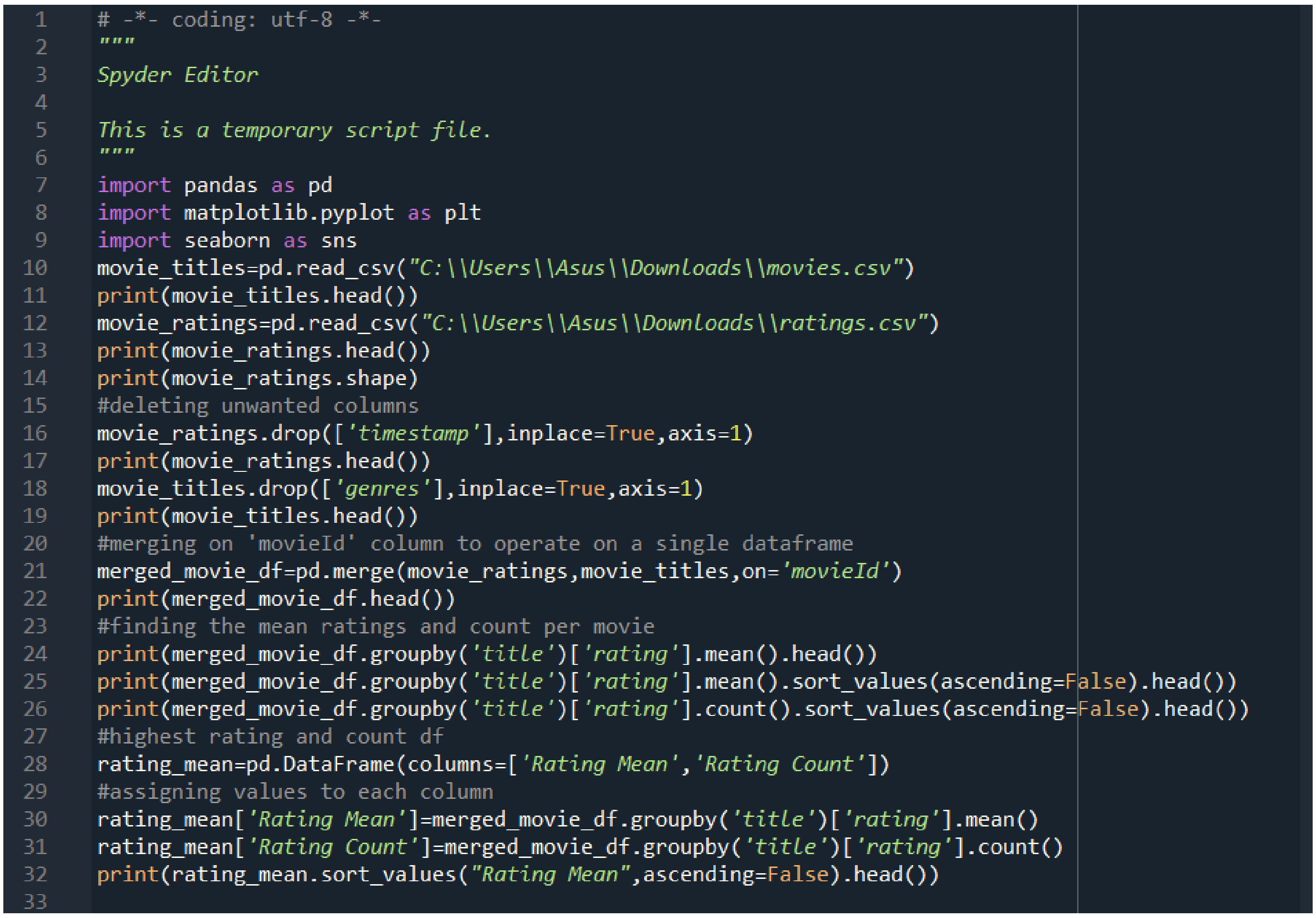
I also plotted three graphs depicting the mean rating, mean count and mean rating vs mean count in three graphs respectively.

II) Content-User Based Filtering (parameter analysed on: rating)

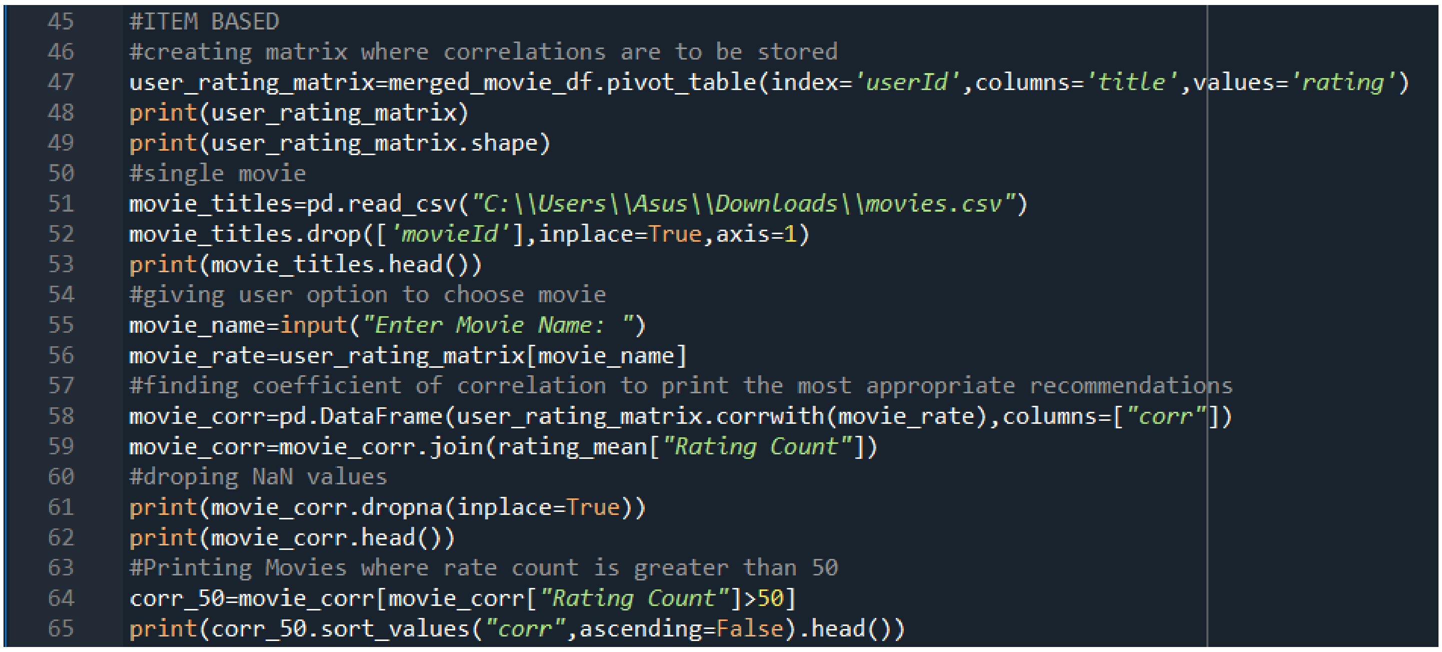
Beginning by creating a matrix of all the movie titles corresponding to the ratings given by each user, the code asks the user for their input on the movie similar to which they want recommendations. After this I find the coefficient of correlation of each movie to the input movie and display the most suitable results accordingly.

Next in this I take the input to be a set (dataframe) of movies and their ratings subsequently finding the movies which are most suitable.

Coding

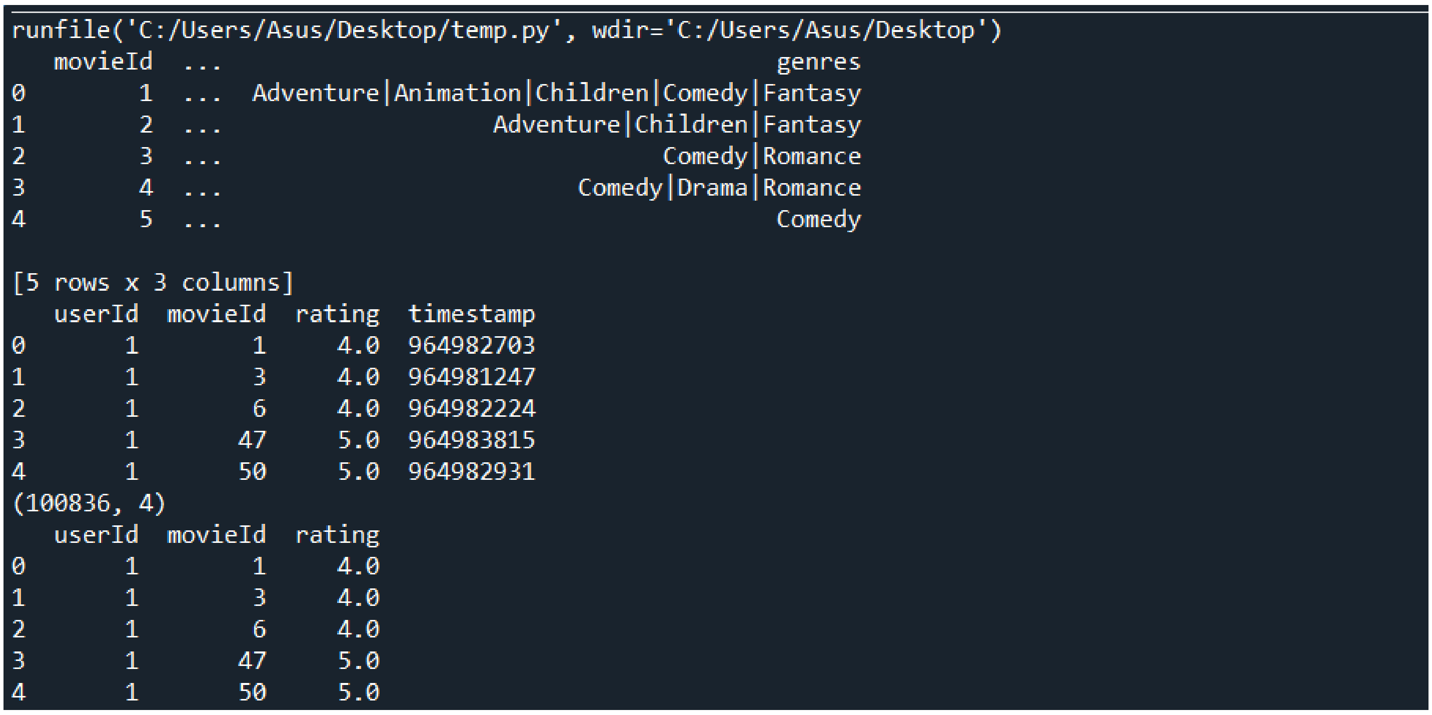


```python
# -*- coding: utf-8 -*-
"""
Spyder Editor

This is a temporary script file.
"""
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
movie_titles=pd.read_csv("C:\\Users\\Asus\\Downloads\\movies.csv")
print(movie_titles.head())
movie_ratings=pd.read_csv("C:\\Users\\Asus\\Downloads\\ratings.csv")
print(movie_ratings.head())
print(movie_ratings.shape)
#deleting unwanted columns
movie_ratings.drop(['timestamp'],inplace=True,axis=1)
print(movie_ratings.head())
movie_titles.drop(['genres'],inplace=True,axis=1)
print(movie_titles.head())
#merging on 'movieId' column to operate on a single dataframe
merged_movie_df=pd.merge(movie_ratings,movie_titles,on='movieId')
print(merged_movie_df.head())
#finding the mean ratings and count per movie
print(merged_movie_df.groupby('title')['rating'].mean().head())
print(merged_movie_df.groupby('title')['rating'].mean().sort_values(ascending=False).head())
print(merged_movie_df.groupby('title')['rating'].count().sort_values(ascending=False).head())
#highest rating and count df
rating_mean=pd.DataFrame(columns=['Rating Mean','Rating Count'])
#assigning values to each column
rating_mean['Rating Mean']=merged_movie_df.groupby('title')['rating'].mean()
rating_mean['Rating Count']=merged_movie_df.groupby('title')['rating'].count()
print(rating_mean.sort_values("Rating Mean",ascending=False).head())
```

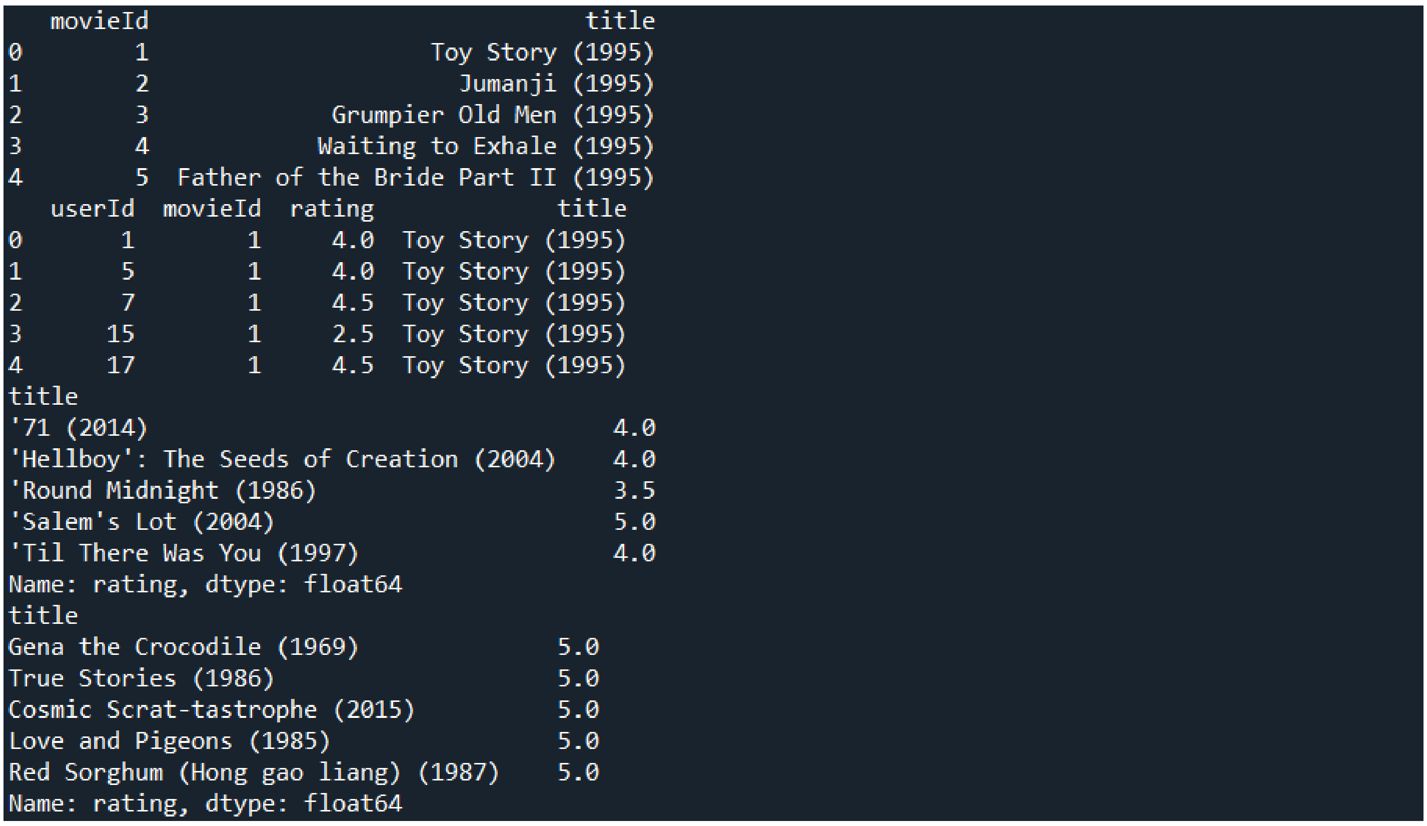


```
#ITEM BASED
#creating matrix where correlations are to be stored
user_rating_matrix=merged_movie_df.pivot_table(index='userId',columns='title',values='rating')
print(user_rating_matrix)
print(user_rating_matrix.shape)
#single movie
movie_titles=pd.read_csv("C:\\Users\\Asus\\Downloads\\movies.csv")
movie_titles.drop(['movieId'],inplace=True,axis=1)
print(movie_titles.head())
#giving user option to choose movie
movie_name=input("Enter Movie Name: ")
movie_rate=user_rating_matrix[movie_name]
#finding coefficient of correlation to print the most appropriate recommendations
movie_corr=pd.DataFrame(user_rating_matrix.corrwith(movie_rate),columns=["corr"])
movie_corr=movie_corr.join(rating_mean["Rating Count"])
#droping NaN values
print(movie_corr.dropna(inplace=True))
print(movie_corr.head())
#Printing Movies where rate count is greater than 50
corr_50=movie_corr[movie_corr["Rating Count"]>50]
print(corr_50.sort_values("corr",ascending=False).head())
```

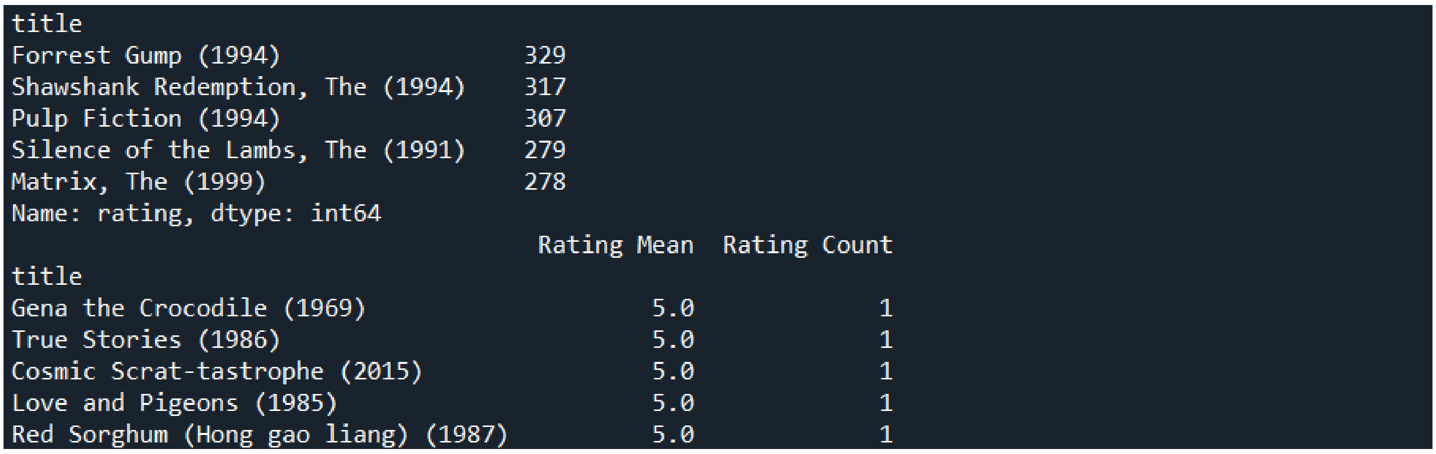
Outputs



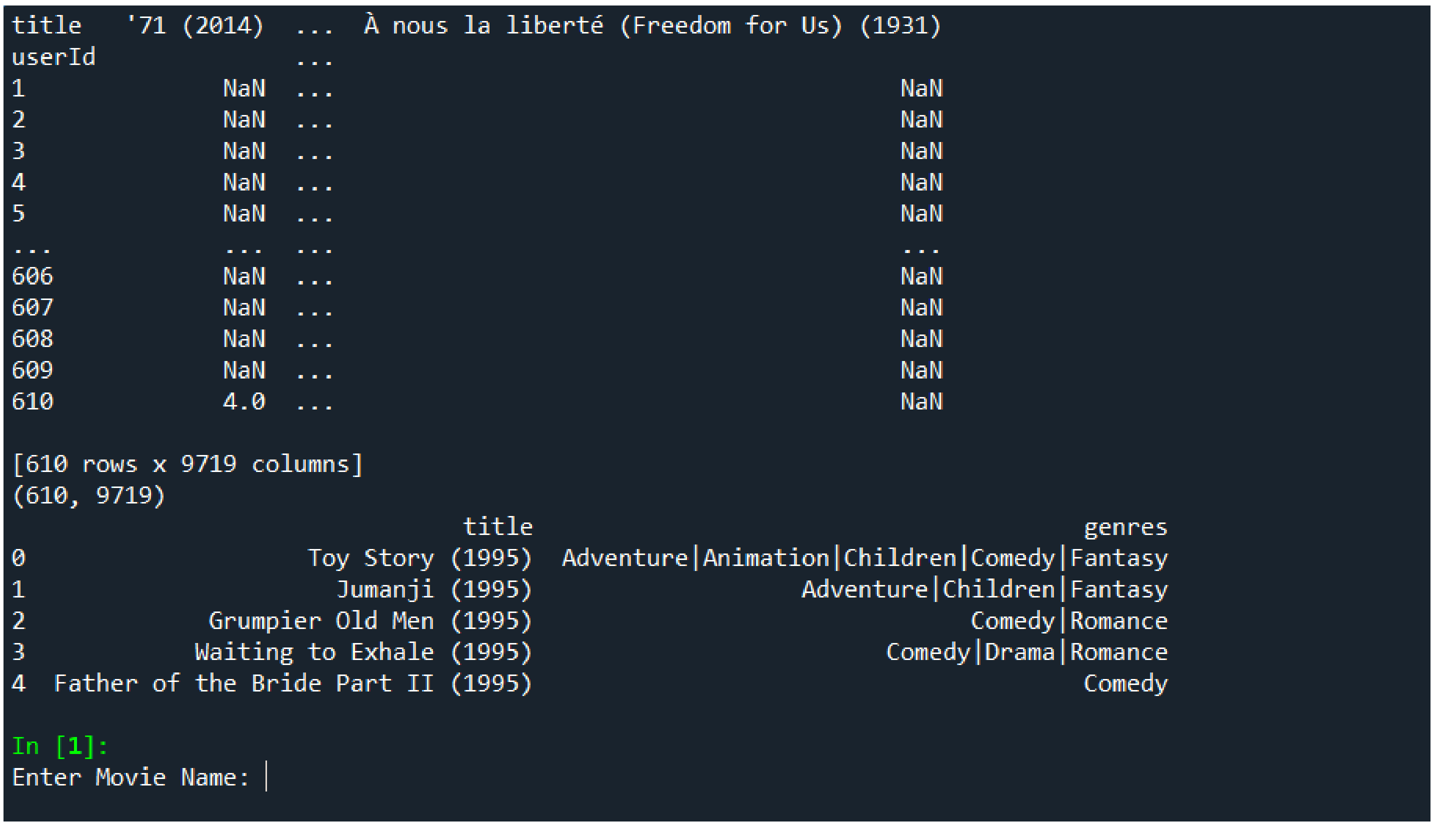
```
runfile('C:/Users/Asus/Desktop/temp.py', wdir='C:/Users/Asus/Desktop')
   movieId  ...                                       genres
0        1  ...  Adventure|Animation|Children|Comedy|Fantasy
1        2  ...                   Adventure|Children|Fantasy
2        3  ...                               Comedy|Romance
3        4  ...                         Comedy|Drama|Romance
4        5  ...                                       Comedy

[5 rows x 3 columns]
   userId  movieId  rating  timestamp
0       1        1     4.0  964982703
1       1        3     4.0  964981247
2       1        6     4.0  964982224
3       1       47     5.0  964983815
4       1       50     5.0  964982931
(100836, 4)
   userId  movieId  rating
0       1        1     4.0
1       1        3     4.0
2       1        6     4.0
3       1       47     5.0
4       1       50     5.0
```



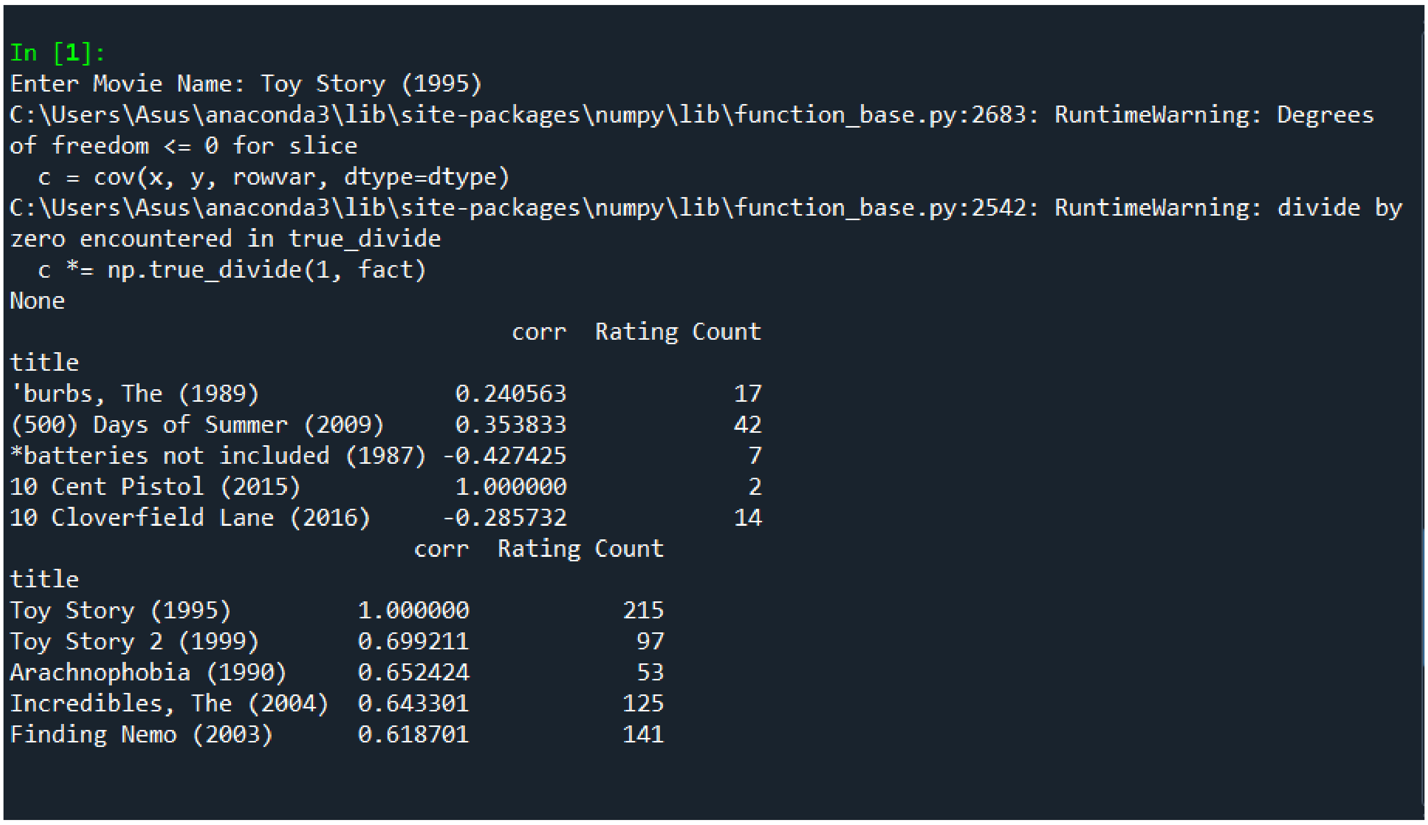
```
   movieId                               title
0        1                    Toy Story (1995)
1        2                      Jumanji (1995)
2        3             Grumpier Old Men (1995)
3        4            Waiting to Exhale (1995)
4        5  Father of the Bride Part II (1995)
   userId  movieId  rating             title
0       1        1     4.0  Toy Story (1995)
1       5        1     4.0  Toy Story (1995)
2       7        1     4.5  Toy Story (1995)
3      15        1     2.5  Toy Story (1995)
4      17        1     4.5  Toy Story (1995)
title
'71 (2014)                                 4.0
'Hellboy': The Seeds of Creation (2004)    4.0
'Round Midnight (1986)                     3.5
'Salem's Lot (2004)                        5.0
'Til There Was You (1997)                  4.0
Name: rating, dtype: float64
title
Gena the Crocodile (1969)              5.0
True Stories (1986)                    5.0
Cosmic Scrat-tastrophe (2015)          5.0
Love and Pigeons (1985)                5.0
Red Sorghum (Hong gao liang) (1987)    5.0
Name: rating, dtype: float64
```



```
title
Forrest Gump (1994)                 329
Shawshank Redemption, The (1994)    317
Pulp Fiction (1994)                 307
Silence of the Lambs, The (1991)    279
Matrix, The (1999)                  278
Name: rating, dtype: int64
                                     Rating Mean  Rating Count
title
Gena the Crocodile (1969)                    5.0             1
True Stories (1986)                          5.0             1
Cosmic Scrat-tastrophe (2015)                5.0             1
Love and Pigeons (1985)                      5.0             1
Red Sorghum (Hong gao liang) (1987)          5.0             1
```



```
title   '71 (2014)  ...  À nous la liberté (Freedom for Us) (1931)
userId              ...
1              NaN  ...                                       NaN
2              NaN  ...                                       NaN
3              NaN  ...                                       NaN
4              NaN  ...                                       NaN
5              NaN  ...                                       NaN
...            ...  ...                                       ...
606            NaN  ...                                       NaN
607            NaN  ...                                       NaN
608            NaN  ...                                       NaN
609            NaN  ...                                       NaN
610            4.0  ...                                       NaN

[610 rows x 9719 columns]
(610, 9719)
                                title                                       genres
0                    Toy Story (1995)  Adventure|Animation|Children|Comedy|Fantasy
1                      Jumanji (1995)                   Adventure|Children|Fantasy
2             Grumpier Old Men (1995)                               Comedy|Romance
3            Waiting to Exhale (1995)                         Comedy|Drama|Romance
4  Father of the Bride Part II (1995)                                       Comedy

In [1]:
Enter Movie Name: 
```



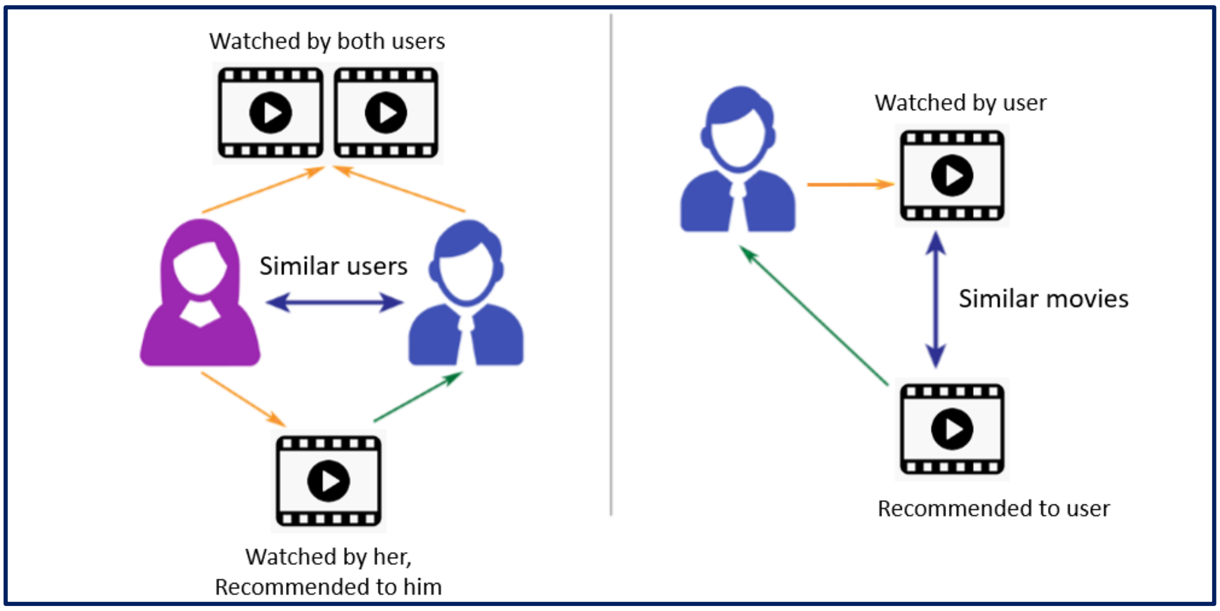
```
In [1]:
Enter Movie Name: Toy Story (1995)
C:\Users\Asus\anaconda3\lib\site-packages\numpy\lib\function_base.py:2683: RuntimeWarning: Degrees
of freedom <= 0 for slice
  c = cov(x, y, rowvar, dtype=dtype)
C:\Users\Asus\anaconda3\lib\site-packages\numpy\lib\function_base.py:2542: RuntimeWarning: divide by
zero encountered in true_divide
  c *= np.true_divide(1, fact)
None
                                    corr  Rating Count
title
'burbs, The (1989)              0.240563            17
(500) Days of Summer (2009)     0.353833            42
*batteries not included (1987) -0.427425             7
10 Cent Pistol (2015)           1.000000             2
10 Cloverfield Lane (2016)     -0.285732            14
                             corr  Rating Count
title
Toy Story (1995)         1.000000           215
Toy Story 2 (1999)       0.699211            97
Arachnophobia (1990)     0.652424            53
Incredibles, The (2004)  0.643301           125
Finding Nemo (2003)      0.618701           141
```

Conclusion

A movie recommendation system works on filtering or predicting the users' film preferences based on their past choices and behaviour. It's an *advanced filtration mechanism* that predicts the possible movie choices of the concerned user and their preferences towards a domain-specific item, aka movie.

It is designed to makes the lives of the users easier and more efficient and help them makes decisions more conveniently and systematically in order to choose the best movie of their choice.

In this recommendation system, we demonstrate two ways in which the user can choose a movie:
a) based on activity of other similar users
b) based on a movie they already like


Watched by both users
Similar users
Watched by her,
Recommended to him
Watched by user
Similar movies
Recommended to user