PROJECT - MOVIE RECOMMENDATION SYSTEM ON MOVIES DATASET

SUBMITTED BY - ROHITH RAHUL

RECOMMENDATION ENGINES

Recommendation Based on a customer's past purchasing behaviour, an engine can determine which products the customer is likely to be interested in or purchase. If the consumer is new, however, this strategy will not work because we have no prior information about them. Therefore, many approaches are employed to address this issue; for instance, frequently the most well-liked products are suggested. Given that they are universal for all new clients, these recommendations wouldn't be very accurate.

There are therefore many different kinds of recommendation engines:

- Collabrative Recommender System
- Content based Recommender System
- Demographic based Recommender System
- Utility based Recommender System
- Knowledge based Recommender System
- Hybrid Recommende System

We will import all of the necessary libraries first.

In [1]:
```
# import libraries
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
```

We're going to employ two datasets: one will be a movie dataset with names of movies, and the other will be a dataset with ratings and user ID.

In [2]:
```
#read csv file
df = pd.read_csv("C:/Users/rohit/OneDrive/Desktop/movies (7).csv")

# first few rows of dataset
df.head()
```

Out[2]:

	movieId	title	genres
0	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
1	2	Jumanji (1995)	Adventure|Children|Fantasy
2	3	Grumpier Old Men (1995)	Comedy|Romance
3	4	Waiting to Exhale (1995)	Comedy|Drama|Romance
4	5	Father of the Bride Part II (1995)	Comedy

In [3]:
```
df.shape
```

Out[3]:
```
(9742, 3)
```

We therefore have 9742 films. We are eliminating the genres section because we don't need it.

In [4]:
```
# drop genres column
df.drop(['genres'],axis=1,inplace=True)
```

We must import our additional dataset, which contains movie ratings.

In [5]:
```
# import rating dataset
rating = pd.read_csv("C:/Users/rohit/OneDrive/Desktop/ratings (7).csv")

# columns
rating.columns
```

Out[5]:
```
Index(['userId', 'movieId', 'rating', 'timestamp'], dtype='object')
```

In [6]:
```
# we need user id, movie id and rating
rating = rating.loc[:,["userId","movieId","rating"]]
rating.head()
```

Out[6]:

	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

In [7]:
```
#then merge movie and rating data
df = pd.merge(df,rating)
```

In [8]:
```
df.head()
```

Out[8]:

	movieId	title	userId	rating
0	1	Toy Story (1995)	1	4.0
1	1	Toy Story (1995)	5	4.0
2	1	Toy Story (1995)	7	4.5
3	1	Toy Story (1995)	15	2.5
4	1	Toy Story (1995)	17	4.5

One user has given one or more movies a rating, as seen here. This indicates that multiple users have given the same movie ratings.

In [9]:
```
df.shape
```

Out[9]:
```
(100836, 4)
```

We must subset our dataset, thus we will select 1M rows.

In [10]:
```
df = df.iloc[:1000000]
```

In [11]:
```
df.shape
```

Out[11]:
```
(100836, 4)
```

In [12]:
```
# basic stats
df.describe()
```

Out[12]:

	movieId	userId	rating
count	100836.000000	100836.000000	100836.000000
mean	19435.295718	326.127564	3.501557
std	35530.987199	182.618491	1.042529
min	1.000000	1.000000	0.500000
25%	1199.000000	177.000000	3.000000
50%	2991.000000	325.000000	3.500000
75%	8122.000000	477.000000	4.000000
max	193609.000000	610.000000	5.000000

DATA VISUALIZATION

Let's determine each movie's average rating.

In [13]:
```
df.groupby("title").mean()['rating'].sort_values(ascending=False)
```

Out[13]:
```
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
                                          ...
Don't Look Now (1973)                      0.5
Journey 2: The Mysterious Island (2012)    0.5
Joe Dirt 2: Beautiful Loser (2015)         0.5
Jesus Christ Vampire Hunter (2001)         0.5
Fullmetal Alchemist 2018 (2017)            0.5
Name: rating, Length: 9719, dtype: float64
```

Let's determine how many ratings a specific movie has received.

In [14]:
```
df.groupby("title").count()["rating"].sort_values(ascending=False)
```

Out[14]:
```
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
                                            ...
King Solomon's Mines (1950)                    1
King Solomon's Mines (1937)                    1
King Ralph (1991)                              1
King Kong Lives (1986)                         1
À nous la liberté (Freedom for Us) (1931)      1
Name: rating, Length: 9719, dtype: int64
```

Making a dataframe with a rating and number of ratings column is the next step.

In [15]:
```
ratings=pd.DataFrame(df.groupby("title").mean()['rating'])
ratings['number of ratings']=pd.DataFrame(df.groupby("title").count()["rating"])
print(ratings.head())
```

```
                                     rating  number of ratings
title
'71 (2014)                              4.0                  1
'Hellboy': The Seeds of Creation (2004)  4.0                  1
'Round Midnight (1986)                  3.5                  2
'Salem's Lot (2004)                     5.0                  1
'Til There Was You (1997)               4.0                  2
```

In [16]:
```
ratings.sort_values(by='rating', ascending=False)
```

Out[16]:

title	rating	number of ratings
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
...	...	...
Don't Look Now (1973)	0.5	1
Journey 2: The Mysterious Island (2012)	0.5	1
Joe Dirt 2: Beautiful Loser (2015)	0.5	1
Jesus Christ Vampire Hunter (2001)	0.5	1
Fullmetal Alchemist 2018 (2017)	0.5	1

9719 rows × 2 columns

In [17]:
```
ratings.describe()
```

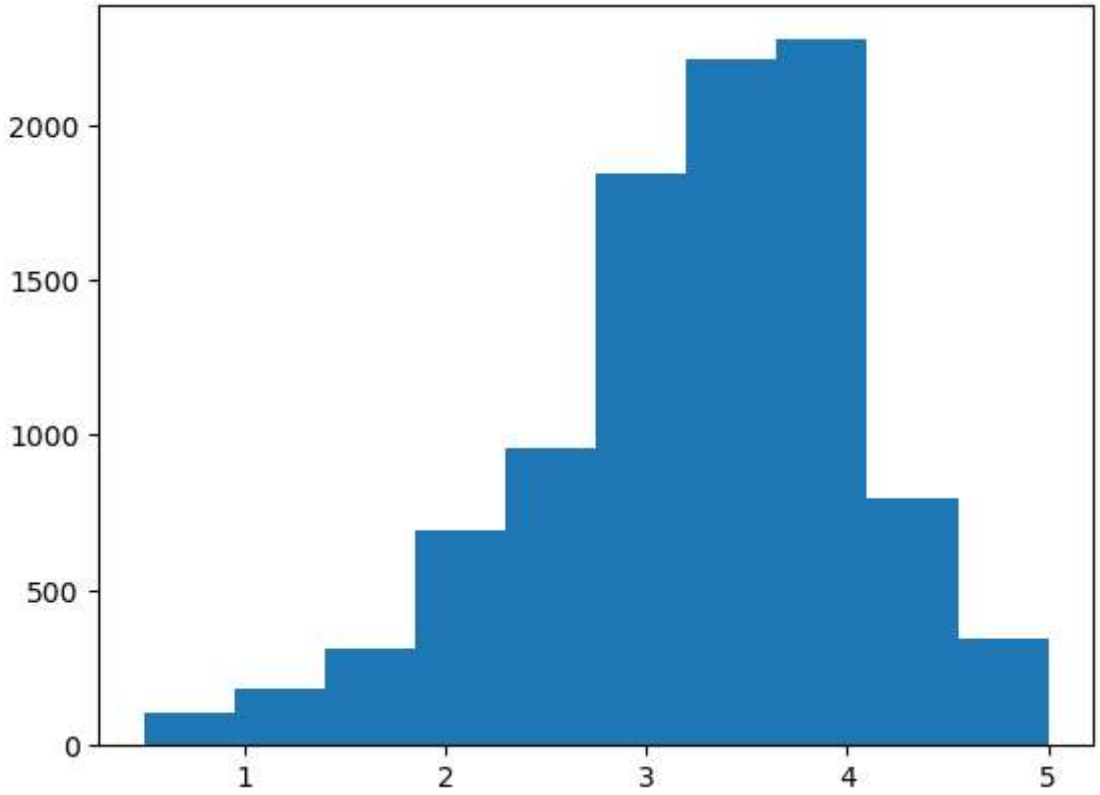
Out[17]:

	rating	number of ratings
count	9719.000000	9719.000000
mean	3.262388	10.375141
std	0.870004	22.406220
min	0.500000	1.000000
25%	2.800000	1.000000
50%	3.416667	3.000000
75%	3.910357	9.000000
max	5.000000	329.000000

Therefore, it is clear from the preceding that there are no films with a rating of 5.

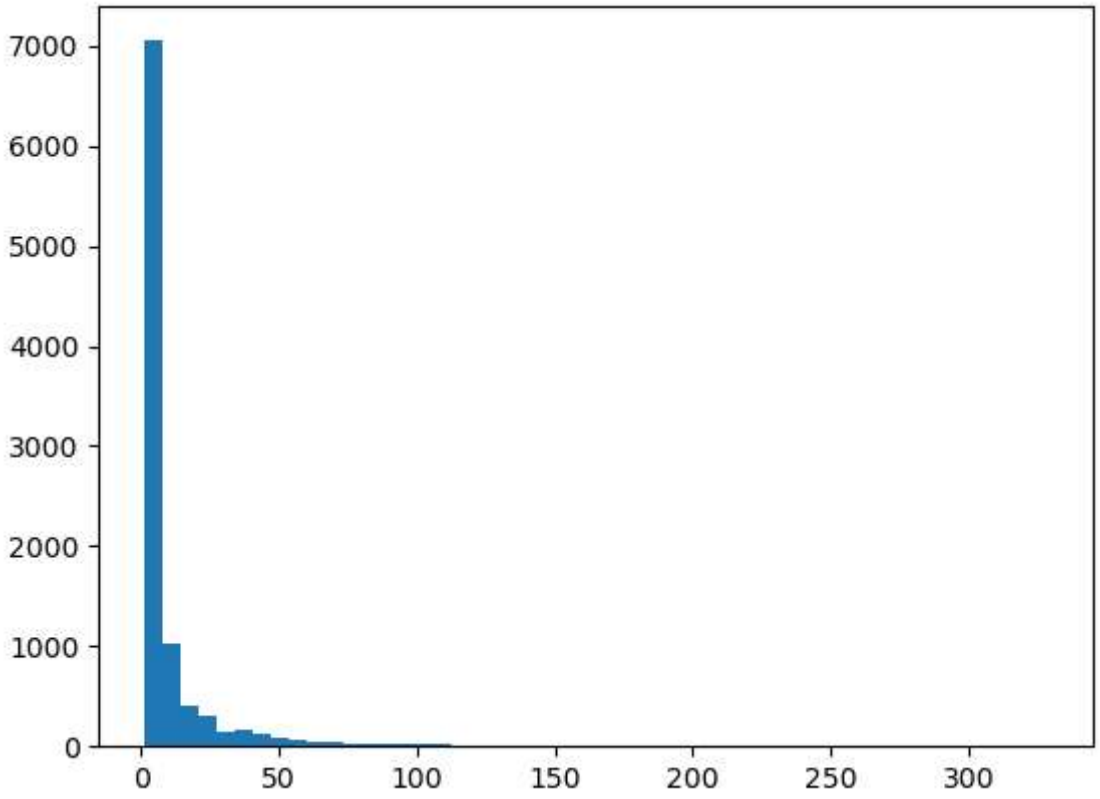
In [18]:
```
plt.hist(ratings['rating'])
plt.show
```

Out[18]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```



In [19]:
```
plt.hist(ratings['number of ratings'],bins=50)
plt.show
```

Out[19]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```



RECOMMENDER SYSTEM

In [20]:
```
# lets make a pivot table in order to make rows are users and columns are movies. And
pivot_table = df.pivot_table(index = ["userId"],columns = ["title"],values = "rating")
pivot_table.head(5)
```

Out[20]:

title	'71 (2014)	'Hellboy': The Seeds of Creation (2004)	'Round Midnight (1986)	'Salem's Lot (2004)	'Til There Was You (1997)	'Tis the Season for Love (2015)	'burbs, The (1989)	'night Mother (1986)	(500) Days of Summer (2009)	*batteries not included (1987)
userId										
1	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
5	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

5 rows × 9719 columns

In [21]:
```
pivot_table.shape
```

Out[21]:
```
(610, 9719)
```

Now we'll create a function that, using the correlation score, will suggest a movie. Be aware that the closer the correlation, the more closely the movies relate to one another.

In [22]:
```
def recommend_movie(movie):
    movie_watched = pivot_table[movie]
    similarity_with_other_movies = pivot_table.corrwith(movie_watched)  # find correlc
    similarity_with_other_movies = similarity_with_other_movies.sort_values(ascending=
    return similarity_with_other_movies.head()
```

In [27]:
```
recommend_movie('Silver Streak (1976)')
```

Out[27]:
```
C:\DEADPOOL\SOFTWARES\lib\site-packages\numpy\lib\function_base.py:2683: RuntimeWarni
ng: Degrees of freedom <= 0 for slice
  c = cov(x, y, rowvar, dtype=dtype)
C:\DEADPOOL\SOFTWARES\lib\site-packages\numpy\lib\function_base.py:2542: RuntimeWarni
ng: divide by zero encountered in true_divide
  c *= np.true_divide(1, fact)
title
Exorcist, The (1973)                      1.0
Good Morning, Vietnam (1987)              1.0
Birdcage, The (1996)                      1.0
Rudolph, the Red-Nosed Reindeer (1964)    1.0
Taxi Driver (1976)                        1.0
dtype: float64
```

In [29]:
```
recommend_movie('Waydowntown (2000)')
```

```
C:\DEADPOOL\SOFTWARES\lib\site-packages\numpy\lib\function_base.py:2683: RuntimeWarni
ng: Degrees of freedom <= 0 for slice
  c = cov(x, y, rowvar, dtype=dtype)
C:\DEADPOOL\SOFTWARES\lib\site-packages\numpy\lib\function_base.py:2542: RuntimeWarni
ng: divide by zero encountered in true_divide
  c *= np.true_divide(1, fact)
```

Out[29]:
```
title
eXistenZ (1999)                  1.0
Escape from New York (1981)      1.0
Good Will Hunting (1997)         1.0
Gone in 60 Seconds (2000)        1.0
Godfather: Part II, The (1974)   1.0
dtype: float64
```

In [30]:
```
recommend_movie('Amen. (2002)')
```

```
C:\DEADPOOL\SOFTWARES\lib\site-packages\numpy\lib\function_base.py:2683: RuntimeWarni
ng: Degrees of freedom <= 0 for slice
  c = cov(x, y, rowvar, dtype=dtype)
C:\DEADPOOL\SOFTWARES\lib\site-packages\numpy\lib\function_base.py:2542: RuntimeWarni
ng: divide by zero encountered in true_divide
  c *= np.true_divide(1, fact)
```

Out[30]:
```
title
Back to the Future Part III (1990)    1.0
Toy Story (1995)                      1.0
Rain Man (1988)                       1.0
Dead Poets Society (1989)             1.0
Shanghai Knights (2003)               1.0
dtype: float64
```

CONCLUSION

- We can get the conclusion that those who saw "Silver Streak (1976)" should watch "Exorcist, The (1973)"
- We can get the conclusion that those who saw "Waydowntown (2000)" should watch "eXistenZ (1999)"
- We can get the conclusion that those who saw "Amen. (2002)" should watch "Back to the Future Part III (1990)"