We create a simple recommender system which takes n input from the user which is the name of the movie and recommends 10 movies based on the input

In [121…
```
import pandas as pd
import numpy as np

from sklearn.metrics.pairwise import cosine_similarity

pd.set_option('display.max_rows',None)
```

In [20]:
```
movies_df = pd.read_csv(r'C:\Solaris\MY COURSES\DATA ANALYTICS\DATA SETS\RECOMMENDER SYSTEMS\movies.csv')
ratings_df = pd.read_csv(r'C:\Solaris\MY COURSES\DATA ANALYTICS\DATA SETS\RECOMMENDER SYSTEMS\ratings.csv')
```

----------Exploratory Data Analysis----------

In [21]:
```
movies_df.head()
```

Out[21]:

	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 [22]:
```
ratings_df.head()
```

Out[22]:

	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

In [23]:
```
ratings_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100836 entries, 0 to 100835
Data columns (total 4 columns):
 #   Column     Non-Null Count   Dtype  
---  ------     --------------   -----  
 0   userId     100836 non-null  int64  
 1   movieId    100836 non-null  int64  
 2   rating     100836 non-null  float64
 3   timestamp  100836 non-null  int64  
dtypes: float64(1), int64(3)
memory usage: 3.1 MB
```

In [24]:
```
ratings_df.shape
```

Out[24]:
```
(100836, 4)
```

In [25]:
```
movies_df.shape
```

Out[25]:
```
(9742, 3)
```

Checking for null values

In [26]:
```
ratings_df.isnull().sum()
```

Out[26]:
```
userId       0
movieId      0
rating       0
timestamp    0
dtype: int64
```

In [27]:
```
movies_df.isnull().sum()
```

Out[27]:
```
movieId    0
title      0
genres     0
dtype: int64
```

----------Creating our Recommendation System----------

For our recommender system we use only those ids that have given ratings to more than 150 movies.

We make a list of all userIds with ratings for less than 150 movies.

Merge the two datasets ('movies_df and 'ratings_df')

Drop all rows where UserIds are in our 'discarded list'

In [28]:
```
rdf = ratings_df[['userId','rating']].groupby(['userId']).count()

rdf = rdf[rdf['rating'] < 150]

rdf.reset_index(inplace = True)

discarded = rdf['userId'].tolist()
```

In [29]:
```
# Merging the two data sets together to create our working data set

newdf = ratings_df.merge(movies_df,'left')

newdf.head()
```

Out[29]:

	userId	movieId	rating	timestamp	title	genres
0	1	1	4.0	964982703	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
1	1	3	4.0	964981247	Grumpier Old Men (1995)	Comedy|Romance
2	1	6	4.0	964982224	Heat (1995)	Action|Crime|Thriller
3	1	47	5.0	964983815	Seven (a.k.a. Se7en) (1995)	Mystery|Thriller
4	1	50	5.0	964982931	Usual Suspects, The (1995)	Crime|Mystery|Thriller

In [30]:
```
newdf.isnull().sum()
```

Out[30]:
```
userId       0
movieId      0
rating       0
timestamp    0
title        0
genres       0
dtype: int64
```

In [31]:
```
# deleting all rows with UserIds that gave ratings to less than 150 movies.



for i in discarded:
    discarded_df = newdf[(newdf['userId']==i)].index
    newdf.drop(discarded_df,inplace = True)
```

In [33]:
```
newdf.head()
```

Out[33]:

	userId	movieId	rating	timestamp	title	genres
0	1	1	4.0	964982703	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
1	1	3	4.0	964981247	Grumpier Old Men (1995)	Comedy|Romance
2	1	6	4.0	964982224	Heat (1995)	Action|Crime|Thriller
3	1	47	5.0	964983815	Seven (a.k.a. Se7en) (1995)	Mystery|Thriller
4	1	50	5.0	964982931	Usual Suspects, The (1995)	Crime|Mystery|Thriller

In [116…
```
newdf[['userId','rating']].groupby(['userId']).count().sort_values(by = 'rating')
```

Out[116]:

	rating
userId	
320	20
576	20
194	20
189	20
442	20
...	...
274	1346
448	1864
474	2108
599	2478
414	2698

581 rows × 1 columns

In [63]:
```
user_rating_table = newdf.pivot_table(index = 'userId', columns = 'title',values = 'rating',fill_value = 0)
user_rating_table = user_rating_table.transpose()
user_rating_table.reset_index(inplace = True)
user_rating_table.head()
```

Out[63]:

userId	title	1	4	6	7	18	19	20	21	28	…	596	597	599	600	603	605	606	607	608	610
0	'71 (2014)	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	…	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	4.0
1	'Hellboy': The Seeds of Creation (2004)	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	…	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	'Round Midnight (1986)	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	…	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	'Til There Was You (1997)	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	…	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	'Tis the Season for Love (2015)	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.5	0.0	…	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

5 rows × 177 columns

Creating a similarity matrix

We want to find how similar movies are to each other. We calculate the distance of each movie from each other using cosine-similarity, thereby creating a similarity matrix

In [91]:
```
similarity_df = user_rating_table.iloc[:,1:]

similarity= cosine_similarity(similarity_df)

similarity_matrix = pd.DataFrame(similarity)

similarity_matrix.head()
```

Out[91]:

	0	1	2	3	4	5	6	7	8	9	…	9368	9369	9370	9371	9372	9373	9374	9375	9376	9377
0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.173435	0.0	0.000000	…	0.0	0.0	0.394611	0.543305	0.707107	0.0	0.150542	0.327327	0.0	0.0
1	0.0	1.0	1.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.715542	…	0.0	0.0	0.000000	0.000000	0.000000	0.0	0.000000	0.000000	0.0	0.0
2	0.0	1.0	1.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.715542	…	0.0	0.0	0.000000	0.000000	0.000000	0.0	0.000000	0.000000	0.0	0.0
3	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.000000	0.0	0.000000	…	0.0	0.0	0.000000	0.000000	0.000000	0.0	0.000000	0.000000	0.0	0.0
4	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.000000	0.0	0.000000	…	0.0	0.0	0.000000	0.000000	0.000000	0.0	0.000000	0.000000	0.0	0.0

5 rows × 9378 columns

In [81]:
```
similarity_matrix.shape
```

Out[81]:
```
(9378, 9378)
```

Getting Similar movies

We make a function that takes the name of the movie as an input from the user and recommends 10 movies .

In [117…
```
def get_similar_movies(name):

    movie_list = []

    a = user_rating_table.index[user_rating_table['title']== name].values[0]
    df1 = similarity_matrix[a]
    df1 = df1.sort_values(ascending = False)

    count = 0

    for i in df1.index:
        count = count + 1
        t = user_rating_table.iloc[i,0]
        movie_list.append(t)

        if count == 11:
            break

    return movie_list
```

In [119…
```
movies =  get_similar_movies("'Round Midnight (1986)")
movies
```

Out[119]:
```
['Monsters (2010)',
 "'Hellboy': The Seeds of Creation (2004)",
 'Space Battleship Yamato (2010)',
 "'Round Midnight (1986)",
 'All the Right Moves (1983)',
 'Hidden Fortress, The (Kakushi-toride no san-akunin) (1958)',
 '...And Justice for All (1979)',
 'Battle of Algiers, The (La battaglia di Algeri) (1966)',
 'Kagemusha (1980)',
 'Sanjuro (Tsubaki Sanjûrô) (1962)',
 'Ghost Rider: Spirit of Vengeance (2012)']
```

In [ ]:
```

```