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

In [2]:
```
movies_df=pd.read_csv("Amazon - Movies and TV Ratings.csv")
```

In [3]:
```
movies_df.head()
```

Out[3]:

	user_id	Movie1	Movie2	Movie3	Movie4	Movie5	Movie6	Movie7	Movie8	Movi
0	A3R5OBKS7OM2IR	5.0	5.0	NaN	NaN	NaN	NaN	NaN	NaN	N
1	AH3QC2PC1VTGP	NaN	NaN	2.0	NaN	NaN	NaN	NaN	NaN	N
2	A3LKP6WPMP9UKX	NaN	NaN	NaN	5.0	NaN	NaN	NaN	NaN	N
3	AVIY68KEPQ5ZD	NaN	NaN	NaN	5.0	NaN	NaN	NaN	NaN	N
4	A1CV1WROP5KTTW	NaN	NaN	NaN	NaN	5.0	NaN	NaN	NaN	N

5 rows × 207 columns

In [4]:
```
movies_df.columns
```

Out[4]:
```
Index(['user_id', 'Movie1', 'Movie2', 'Movie3', 'Movie4', 'Movie5', 'Movie6',
       'Movie7', 'Movie8', 'Movie9',
       ...
       'Movie197', 'Movie198', 'Movie199', 'Movie200', 'Movie201', 'Movie20
2',
       'Movie203', 'Movie204', 'Movie205', 'Movie206'],
      dtype='object', length=207)
```

In [5]:
```
#Analysis Task
#1.Exploratory Data Analysis:
```

user_id column is ignoorable as it is not needed for Exploratory Data Analysis:

In [6]:
```
movies_df.drop(['user_id'],axis=1,inplace=True)
```

In [7]:
```
movies_df.head()
```

Out[7]:

	Movie1	Movie2	Movie3	Movie4	Movie5	Movie6	Movie7	Movie8	Movie9	Movie10	...	Mo
0	5.0	5.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	
1	NaN	NaN	2.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	
2	NaN	NaN	NaN	5.0	NaN	NaN	NaN	NaN	NaN	NaN	...	
3	NaN	NaN	NaN	5.0	NaN	NaN	NaN	NaN	NaN	NaN	...	
4	NaN	NaN	NaN	NaN	5.0	NaN	NaN	NaN	NaN	NaN	...	

5 rows × 206 columns

In [8]:
```
#Which movies have maximum views/ratings?
```

Ans-The movie having maximum no. of views should have the minimum no. of null values in that perticular movie column

In [9]:
```
a=movies_df.isna().sum()
```

total no. of null values in a column defines the no. of user that did't rated that movie or did't wathced that movie and these values are saved in a new data set named as "a"

In [10]:
```
a.sort_values(ascending=True).head(1)
```

Out[10]:
```
Movie127    2535
dtype: int64
```

movie no. 127 has maximum views

In [11]:
```
#What is the average rating for each movie?
```

average rating of any movie can be taken by calculating the mean value of that movie's rating given by the all user

In [12]:
```
b=movies_df.mean()
b
```

Out[12]:
```
Movie1      5.000000
Movie2      5.000000
Movie3      2.000000
Movie4      5.000000
Movie5      4.103448
               ...
Movie202    4.333333
Movie203    3.000000
Movie204    4.375000
Movie205    4.628571
Movie206    4.923077
Length: 206, dtype: float64
```

mean rating of each movie is saved in a new data set named as "b"

In [13]:
```
#Define the top 5 movies with the maximum ratings?
```

ans-any movie is considered to be as maximum rated if it has maximum average rating and maximum no. of user who has given the rating for that movie means that movie should have minimum no. of null values in the column as well as maximum average rating/

In [14]:
```
z=pd.DataFrame()
z['rating']=b
z['no_of_null_values']=a
z
```

Out[14]:

	rating	no_of_null_values
Movie1	5.000000	4847
Movie2	5.000000	4847
Movie3	2.000000	4847
Movie4	5.000000	4846
Movie5	4.103448	4819
...	...	...
Movie202	4.333333	4842
Movie203	3.000000	4847
Movie204	4.375000	4840
Movie205	4.628571	4813
Movie206	4.923077	4835

206 rows × 2 columns

the dataframe "z" contains both average rating of the movie and total no. of nan values for that movie

In [15]:
```
z["no_of_null_values"][z["rating"]==5].sort_values(ascending=True).head()
```

Out[15]:
```
Movie186    4839
Movie191    4842
Movie188    4842
Movie12     4843
Movie101    4843
Name: no_of_null_values, dtype: int64
```

after applying the asumed constraints over the dataframe "z" we found that Movie186
Movie191
Movie188
Movie12
Movie101
are the top 5 movies with maximum rating

In [16]:
```
#Define the top 5 movies with the least audience.
```

To find the top 5 movies (the movies average rating must be maximum) but shuld have the least audiance(the movie should have the maximum no. of null values in it)

In [17]:
```
z["no_of_null_values"][z["rating"]==5].sort_values(ascending=False).head()
```

Out[17]:
```
Movie199    4847
Movie63     4847
Movie48     4847
Movie49     4847
Movie50     4847
Name: no_of_null_values, dtype: int64
```

by applying the assumed constraints we found that the movies

Movie199
Movie63
Movie48
Movie49
Movie50

are maximum average rating but have the least no. of audiance

In [18]:
```
#Recommendation Model: Some of the movies hadn’t been watched and therefore, a
re not rated by the users.
#Netflix would like to take this as an opportunity and build a machine learnin
g recommendation algorithm which provides the ratings for each of the users.
```

In [19]:
```
movies_df=pd.read_csv("Amazon - Movies and TV Ratings.csv")
```

creating association matrix

In [20]:
```
movies_df.drop(['user_id'],axis=1,inplace=True)
```

In [21]:
```
movies_df.columns
```

Out[21]:
```
Index(['Movie1', 'Movie2', 'Movie3', 'Movie4', 'Movie5', 'Movie6', 'Movie7',
       'Movie8', 'Movie9', 'Movie10',
       ...
       'Movie197', 'Movie198', 'Movie199', 'Movie200', 'Movie201', 'Movie20
2',
       'Movie203', 'Movie204', 'Movie205', 'Movie206'],
      dtype='object', length=206)
```

In [22]:
```
learningMatrix =movies_df
learningMatrix.fillna(0,inplace=True)
learningMatrix
```

Out[22]:

	Movie1	Movie2	Movie3	Movie4	Movie5	Movie6	Movie7	Movie8	Movie9	Movie10	...
0	5.0	5.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...
1	0.0	0.0	2.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...
2	0.0	0.0	0.0	5.0	0.0	0.0	0.0	0.0	0.0	0.0	...
3	0.0	0.0	0.0	5.0	0.0	0.0	0.0	0.0	0.0	0.0	...
4	0.0	0.0	0.0	0.0	5.0	0.0	0.0	0.0	0.0	0.0	...
...	...	...	...	...	...	...	...	...	...	...	...
4843	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...
4844	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...
4845	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...
4846	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...
4847	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...

4848 rows × 206 columns

In [23]:
```
#Divide the data into training and test data
```

In [24]:
```
from sklearn.model_selection import train_test_split
```

In [25]:
```
train_data_matrix, test_data_matrix=train_test_split(learningMatrix, test_size
=0.25 )
```

In [26]:
```
#Build a recommendation model on training data
```

apply Cosine Similarity Formula on Association Matrix

In [27]:
```
from scipy.spatial.distance import cosine
from sklearn.metrics import pairwise_distances
user_similarity =1-pairwise_distances(train_data_matrix, metric="cosine" )
np.fill_diagonal(user_similarity, 0 )
ratings_matrix = pd.DataFrame(user_similarity )
ratings_matrix
```

Out[27]:

	0	1	2	3	4	5	6	7	8	9	...	3626	3627	3628	3629	3630	3631	36
0	0.0	1.0	0.0	1.0	0.0	1.0	1.0	1.0	0.0	1.0	...	1.0	0.0	0.0	0.0	0.0	0.0	
1	1.0	0.0	0.0	1.0	0.0	1.0	1.0	1.0	0.0	1.0	...	1.0	0.0	0.0	0.0	0.0	0.0	
2	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	1.0	1.0	0.0	0.0	0.0	1.0	1.0	1.0	0.0	1.0	...	1.0	0.0	0.0	0.0	0.0	0.0	
4	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	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
3631	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	1.0	0.0	0.0	
3632	1.0	1.0	0.0	1.0	0.0	1.0	1.0	1.0	0.0	1.0	...	1.0	0.0	0.0	0.0	0.0	0.0	
3633	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	
3634	1.0	1.0	0.0	1.0	0.0	1.0	1.0	1.0	0.0	1.0	...	1.0	0.0	0.0	0.0	0.0	0.0	
3635	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	

3636 rows × 3636 columns

In [28]:
```
#Make predictions on the test data
```

In [29]:
```
mean_user_rating = train_data_matrix.mean(axis=1)[:, np.newaxis]
mean_user_rating
```

Out[29]:
```
array([[0.02427184],
       [0.02427184],
       [0.02427184],
       ...,
       [0.01941748],
       [0.00970874],
       [0.04854369]])
```

In [30]:
```
ratings_diff = (train_data_matrix - mean_user_rating)
ratings_diff
```

Out[30]:

	Movie1	Movie2	Movie3	Movie4	Movie5	Movie6	Movie7	Movie8	M
2720	-0.024272	-0.024272	-0.024272	-0.024272	-0.024272	-0.024272	-0.024272	-0.024272	-0.0
2922	-0.024272	-0.024272	-0.024272	-0.024272	-0.024272	-0.024272	-0.024272	-0.024272	-0.0
4501	-0.024272	-0.024272	-0.024272	-0.024272	-0.024272	-0.024272	-0.024272	-0.024272	-0.0
3196	-0.024272	-0.024272	-0.024272	-0.024272	-0.024272	-0.024272	-0.024272	-0.024272	-0.0
1131	-0.024272	-0.024272	-0.024272	-0.024272	-0.024272	-0.024272	-0.024272	-0.024272	-0.0
...	...	...	...	...	...	...	...	...	
4027	-0.004854	-0.004854	-0.004854	-0.004854	-0.004854	-0.004854	-0.004854	-0.004854	-0.0
1755	-0.019417	-0.019417	-0.019417	-0.019417	-0.019417	-0.019417	-0.019417	-0.019417	-0.0
585	-0.019417	-0.019417	-0.019417	-0.019417	-0.019417	-0.019417	-0.019417	-0.019417	-0.0
2215	-0.009709	-0.009709	-0.009709	-0.009709	-0.009709	-0.009709	-0.009709	-0.009709	-0.0
3865	-0.048544	-0.048544	-0.048544	-0.048544	-0.048544	-0.048544	-0.048544	-0.048544	-0.0

3636 rows × 206 columns

In [31]:
```
# User based collaborative filtering model:
user_pred = mean_user_rating + user_similarity.dot(ratings_diff) / np.array([n
p.abs(user_similarity).sum(axis=1)]).T
user_pred
```

```
<ipython-input-31-a0c78e4d4593>:2: RuntimeWarning: invalid value encountered
in true_divide
  user_pred = mean_user_rating + user_similarity.dot(ratings_diff) / np.array
([np.abs(user_similarity).sum(axis=1)]).T
```

Out[31]:
```
array([[ 0.00424275,  0.00424275,  0.00424275, ...,  0.00424275,
         0.00424275,  0.00424275],
       [ 0.00424275,  0.00424275,  0.00424275, ...,  0.00424275,
         0.00424275,  0.00424275],
       [ 0.00082664,  0.00082664,  0.00082664, ...,  0.00082664,
         0.00082664,  0.00082664],
       ...,
       [-0.00425446, -0.00425446, -0.00425446, ..., -0.00425446,
        -0.00425446, -0.00425446],
       [-0.01032874, -0.01032874, -0.01032874, ..., -0.01032874,
        -0.01032874, -0.01032874],
       [ 0.02594279,  0.02594279,  0.02594279, ...,  0.02594279,
         0.02594279,  0.02594279]])
```

In [32]:
```
#item based collaborative filtering
movie_similarity = pairwise_distances(train_data_matrix.T, metric='cosine')
movie_pred = train_data_matrix.dot(movie_similarity)/ np.array([np.abs(movie_s
imilarity).sum(axis=1)])

movie_pred
```

Out[32]:

	0	1	2	3	4	5	6	7	8
2720	0.024510	0.024510	0.024272	0.024390	0.024390	0.024390	0.024390	0.024390	0.024272
2922	0.024510	0.024510	0.024272	0.024390	0.024390	0.024390	0.024390	0.024390	0.024272
4501	0.024510	0.024510	0.024272	0.024390	0.024390	0.024390	0.024390	0.024390	0.024272
3196	0.024510	0.024510	0.024272	0.024390	0.024390	0.024390	0.024390	0.024390	0.024272
1131	0.024510	0.024510	0.024272	0.024390	0.024390	0.024390	0.024390	0.024390	0.024272
...	...	...	...	...	...	...	...	...	...
4027	0.004902	0.004902	0.004854	0.004878	0.004878	0.004878	0.004878	0.004878	0.004854
1755	0.019608	0.019608	0.019417	0.019512	0.019512	0.019512	0.019512	0.019512	0.019417
585	0.019608	0.019608	0.019417	0.019512	0.019512	0.019512	0.019512	0.019512	0.019417
2215	0.009804	0.009804	0.009709	0.009756	0.009756	0.009756	0.009756	0.009756	0.009709
3865	0.049020	0.049020	0.048544	0.048780	0.048780	0.048780	0.048780	0.048780	0.048544

3636 rows × 206 columns

although item based prediction are notasked in the question but lets check the knowlage if i am correct.

In [ ]:
```
```