Name:SHEIK PAREETH

Load the Libraries

In [1]:

```
# import the libraries
import pandas as pd
import numpy as np
```

Load the DataSet

In [2]:

```
# reading ratings file:
r_cols = ['user_id', 'movie_id', 'rating', 'unix_timestamp']
```

In [4]:

```
ratings = pd.read_csv('ml-100k/u.data', sep='\t', names=r_cols, encoding='latin-1')
```

In [5]:

```
ratings
```

Out[5]:

	user_id	movie_id	rating	unix_timestamp
0	196	242	3	881250949
1	186	302	3	891717742
2	22	377	1	878887116
3	244	51	2	880606923
4	166	346	1	886397596
...	...	...	...	...
99995	880	476	3	880175444
99996	716	204	5	879795543
99997	276	1090	1	874795795
99998	13	225	2	882399156
99999	12	203	3	879959583

100000 rows × 4 columns

In [6]:

```
n_users=ratings.user_id.unique().shape[0]
n_items=ratings.movie_id.unique().shape[0]
```

In [7]:

```
print("The number of user:",n_users)
print("The number of n_items:",n_items)
```

```
The number of user: 943
The number of n_items: 1682
```

Create pivot table for user and movie based on ratings

In [8]:

```
data=ratings.pivot_table(index='user_id',columns='movie_id',values='rating')
```

In [9]:

```
data
```

Out[9]:

movie_id	1	2	3	4	5	6	7	8	9	10	...	1673	1674	1675	167
user_id															
1	5.0	3.0	4.0	3.0	3.0	5.0	4.0	1.0	5.0	3.0	...	NaN	NaN	NaN	Na
2	4.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	2.0	...	NaN	NaN	NaN	Na
3	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	Na
4	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	Na
5	4.0	3.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	Na
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
939	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	5.0	NaN	...	NaN	NaN	NaN	Na
940	NaN	NaN	NaN	2.0	NaN	NaN	4.0	5.0	3.0	NaN	...	NaN	NaN	NaN	Na
941	5.0	NaN	NaN	NaN	NaN	NaN	4.0	NaN	NaN	NaN	...	NaN	NaN	NaN	Na
942	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	Na
943	NaN	5.0	NaN	NaN	NaN	NaN	NaN	NaN	3.0	NaN	...	NaN	NaN	NaN	Na

943 rows × 1682 columns

In [10]:

```
#Replace with nan with 0
data_matrix=data.replace(np.nan,0)
```

In [11]:

```
data_matrix
```

Out[11]:

movie_id	1	2	3	4	5	6	7	8	9	10	...	1673	1674	1675	1676	1677	167
user_id																	
1	5.0	3.0	4.0	3.0	3.0	5.0	4.0	1.0	5.0	3.0	...	0.0	0.0	0.0	0.0	0.0	0.
2	4.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.0	...	0.0	0.0	0.0	0.0	0.0	0.
3	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	0.0	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.
5	4.0	3.0	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.
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	.
939	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	5.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.
940	0.0	0.0	0.0	2.0	0.0	0.0	4.0	5.0	3.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.
941	5.0	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.
942	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.
943	0.0	5.0	0.0	0.0	0.0	0.0	0.0	0.0	3.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.

943 rows × 1682 columns

Find Cosine Similatity for user and Item

In [12]:

```
from sklearn.metrics.pairwise import pairwise_distances
user_similarity = pairwise_distances(data_matrix, metric='cosine')
item_similarity = pairwise_distances(data_matrix.T, metric='cosine')
```

In [13]:

```
user_similarity
```

Out[13]:

```
array([[2.44249065e-15, 8.33069016e-01, 9.52540457e-01, ...,
        8.51383057e-01, 8.20492117e-01, 6.01825261e-01],
       [8.33069016e-01, 3.33066907e-16, 8.89408675e-01, ...,
        8.38515222e-01, 8.27732187e-01, 8.94202122e-01],
       [9.52540457e-01, 8.89408675e-01, 0.00000000e+00, ...,
        8.98757435e-01, 8.66583851e-01, 9.73444131e-01],
       ...,
       [8.51383057e-01, 8.38515222e-01, 8.98757435e-01, ...,
        1.11022302e-16, 8.98358201e-01, 9.04880419e-01],
       [8.20492117e-01, 8.27732187e-01, 8.66583851e-01, ...,
        8.98358201e-01, 0.00000000e+00, 8.17535338e-01],
       [6.01825261e-01, 8.94202122e-01, 9.73444131e-01, ...,
        9.04880419e-01, 8.17535338e-01, 0.00000000e+00]])
```

In [14]:

```
item_similarity
```

Out[14]:

```
array([[0.00000000e+00, 5.97617822e-01, 6.69755213e-01, ...,
        1.00000000e+00, 9.52816933e-01, 9.52816933e-01],
       [5.97617822e-01, 0.00000000e+00, 7.26930825e-01, ...,
        1.00000000e+00, 9.21700637e-01, 9.21700637e-01],
       [6.69755213e-01, 7.26930825e-01, 8.88178420e-16, ...,
        1.00000000e+00, 1.00000000e+00, 9.03124947e-01],
       ...,
       [1.00000000e+00, 1.00000000e+00, 1.00000000e+00, ...,
        0.00000000e+00, 1.00000000e+00, 1.00000000e+00],
       [9.52816933e-01, 9.21700637e-01, 1.00000000e+00, ...,
        1.00000000e+00, 0.00000000e+00, 1.00000000e+00],
       [9.52816933e-01, 9.21700637e-01, 9.03124947e-01, ...,
        1.00000000e+00, 1.00000000e+00, 0.00000000e+00]])
```

Using formula for user and item we are calcuating the score value

In [15]:

```
def predict(ratings, similarity, type='user'):
    if type == 'user':
        mean_user_rating = ratings.mean(axis=1)
        #We use np.newaxis so that mean_user_rating has same format as ratings
        ratings_diff = (ratings - mean_user_rating[:, np.newaxis])
        pred = mean_user_rating[:, np.newaxis] + similarity.dot(ratings_diff) / np.array([r
    elif type == 'item':
        pred = ratings.dot(similarity) / np.array([np.abs(similarity).sum(axis=1)])
    return pred
```

In [16]:

```
# prediction Table
user_prediction = predict(data_matrix, user_similarity, type='user')
item_prediction = predict(data_matrix, item_similarity, type='item')
```

In [18]:

```
item_prediction
```

Out[18]:

	0	1	2	3	4	5	6	7	
user_id									
1	0.446278	0.475473	0.505938	0.443633	0.512667	0.547939	0.446243	0.463059	0.474
2	0.108544	0.132957	0.125589	0.124932	0.131178	0.129005	0.110883	0.122223	0.109
3	0.085685	0.091690	0.087643	0.089966	0.089658	0.089985	0.083492	0.089725	0.085
4	0.053693	0.059604	0.058114	0.058364	0.059356	0.061472	0.053374	0.058615	0.055
5	0.224739	0.229171	0.263280	0.226387	0.259973	0.296529	0.232710	0.237109	0.258
...	...	...	...	...	...	...	...	...	
939	0.092574	0.113870	0.110211	0.112040	0.112768	0.123140	0.098578	0.110839	0.098
940	0.164358	0.184894	0.196502	0.164884	0.195860	0.209652	0.162840	0.165606	0.171
941	0.032300	0.045024	0.042924	0.043223	0.047493	0.051077	0.032761	0.042646	0.039
942	0.157779	0.174095	0.189000	0.163514	0.186140	0.194151	0.164910	0.156970	0.167
943	0.247672	0.244892	0.282630	0.241440	0.279338	0.335121	0.250015	0.263325	0.273

943 rows × 1682 columns

In [17]:

```
user_prediction
```

Out[17]:

```
array([[ 2.06532606,  0.73430275,  0.62992381, ...,  0.39359041,
         0.39304874,  0.3927712 ],
       [ 1.76308836,  0.38404019,  0.19617889, ..., -0.08837789,
        -0.0869183 , -0.08671183],
       [ 1.79590398,  0.32904733,  0.15882885, ..., -0.13699223,
        -0.13496852, -0.13476488],
       ...,
       [ 1.59151513,  0.27526889,  0.10219534, ..., -0.16735162,
        -0.16657451, -0.16641377],
       [ 1.81036267,  0.40479877,  0.27545013, ..., -0.00907358,
        -0.00846587, -0.00804858],
       [ 1.8384313 ,  0.47964837,  0.38496292, ...,  0.14686675,
         0.14629808,  0.14641455]])
```

As per User based filtering ,first have to find similarity between the input user and others

In [19]:

```
#1. Select input user
input_user=34
```

In [20]:

```
#2. Convert the user_sim table into DataFrame
user_sim_table=pd.DataFrame(user_similarity)
```

In [21]:

```
user_sim_table
```

Out[21]:

	0	1	2	3	4	5	6	7	
0	2.442491e-15	8.330690e-01	0.952540	0.935642	6.215248e-01	0.569761	0.559633	0.680928	0.92
1	8.330690e-01	3.330669e-16	0.889409	0.821879	9.270210e-01	0.754157	0.892672	0.896656	0.83
2	9.525405e-01	8.894087e-01	0.000000	0.655849	9.787555e-01	0.927585	0.933863	0.916940	0.93
3	9.356422e-01	8.218788e-01	0.655849	0.000000	9.681958e-01	0.931956	0.908770	0.811940	0.89
4	6.215248e-01	9.270210e-01	0.978755	0.968196	4.440892e-16	0.762714	0.626400	0.751070	0.94
...	...	...	...	...	...	...	...	...	
938	8.819047e-01	7.714166e-01	0.973729	0.969862	9.285415e-01	0.888148	0.892973	0.904102	0.96
939	6.859280e-01	7.732100e-01	0.838110	0.803142	7.600453e-01	0.647551	0.670075	0.753117	0.87
940	8.513831e-01	8.385152e-01	0.898757	0.847959	8.604049e-01	0.855554	0.940007	0.853855	0.85
941	8.204921e-01	8.277322e-01	0.866584	0.829914	8.475026e-01	0.682672	0.717997	0.824678	0.90
942	6.018253e-01	8.942021e-01	0.973444	0.941248	6.860592e-01	0.723958	0.605636	0.700191	0.92

943 rows × 943 columns

In [22]:

```
#3. Find similarity user for 34 using cosine table
similar_input_user= user_sim_table[input_user].sort_values(ascending=True).head(5).index
```

In [23]:

```
similar_input_user
```

Out[23]:

```
Int64Index([34, 450, 852, 811, 509], dtype='int64')
```

In [24]:

```
#4.Convert in to list
similar_user_input=list(similar_input_user)
```

In [25]:

```
#5. Using similar_user_input,can select movie id from ratings table
similar_user_movieid_list=[]
for sim_user in similar_user_input:
    sim=list(ratings[ratings['user_id']==sim_user]['movie_id'])
    similar_user_movieid_list.append(sim)
```

In [26]:

```
len(similar_user_movieid_list)
```

Out[26]:

```
5
```

In [27]:

```
similar_user_movieid_list
```

Out[27]:

```
[[312,
  242,
  690,
  310,
  259,
  299,
  245,
  332,
  329,
  286,
  1024,
  324,
  294,
  292,
  990,
  289,
  898,
  899,
```

In [28]:

```
#6. Convert all the list as single
import itertools
similar_user_movieid_single_list=list(itertools.chain.from_iterable(similar_user_movieid_li
```

In [29]:

```
len(similar_user_movieid_single_list)
```

Out[29]:

```
663
```

In [30]:

```
#7. Unique movieid from the list
Unique_movieid_similar_user=set(similar_user_movieid_single_list)
```

In [31]:

```
len(Unique_movieid_similar_user)
```

Out[31]:

```
590
```

In [32]:

```
#8. Input user watched movie_list
input_user_watched_movieid=list(ratings[ratings['user_id']==input_user]['movie_id'].values)
```

In [33]:

```
input_user_watched_movieid
```

Out[33]:

```
[312,
 242,
 690,
 310,
 259,
 299,
 245,
 332,
 329,
 286,
 1024,
 324,
 294,
 292,
 990,
 289,
 898,
 899,
 288,
 991]
```

In [34]:

```
#9. Create a list which should have recom movieid to the input user
recom=[]
for per_id in Unique_movieid_similar_user:
    if(per_id in input_user_watched_movieid):
        pass
    else:
        recom.append(per_id)
```

In [35]:

```
len(recom)
```

Out[35]:

```
570
```

In [36]:

```
sorted(recom)
```

Out[36]:

```
[1,
 2,
 3,
 4,
 7,
 10,
 11,
 12,
 13,
 15,
 22,
 23,
 25,
 26,
 28,
 29,
 33,
 35,
```

In [39]:

```
# Checking the common movie list
list(set(Unique_movieid_similar_user)&set(input_user_watched_movieid))
```

Out[39]:

```
[1024,
 898,
 259,
 899,
 286,
 288,
 289,
 292,
 294,
 299,
 690,
 310,
 312,
 324,
 329,
 332,
 990,
 991,
 242,
 245]
```

In [40]:

```
user_pred=pd.DataFrame(user_prediction)
```

In [41]:

```
user_pred
```

Out[41]:

	0	1	2	3	4	5	6	7	8
0	2.065326	0.734303	0.629924	1.010669	0.640686	0.476150	1.784569	1.163032	1.513350
1	1.763088	0.384040	0.196179	0.731538	0.225643	0.003892	1.493597	0.876153	1.108467
2	1.795904	0.329047	0.158829	0.684154	0.173277	-0.035621	1.488230	0.835769	1.135426
3	1.729951	0.293913	0.127741	0.644932	0.142143	-0.062261	1.437010	0.796249	1.096663
4	1.796651	0.454474	0.354422	0.763130	0.359539	0.195987	1.547370	0.908904	1.292027
...	...	...	...	...	...	...	...	...	...
938	1.676950	0.346339	0.177518	0.689906	0.199740	0.003297	1.429565	0.830905	1.070986
939	1.822346	0.419125	0.286430	0.715605	0.294442	0.106633	1.514591	0.853050	1.195304
940	1.591515	0.275269	0.102195	0.624383	0.133762	-0.069553	1.320734	0.765529	1.035088
941	1.810363	0.404799	0.275450	0.726616	0.281316	0.087068	1.550310	0.850057	1.205745
942	1.838431	0.479648	0.384963	0.780521	0.388442	0.240998	1.564232	0.946704	1.289865

943 rows × 1682 columns

In [42]:

```
user_pred_Trans=user_pred.T
```

In [43]:

```
user_pred_Trans[34]
```

Out[43]:

```
0       1.740469
1       0.283366
2       0.114987
3       0.645096
4       0.127844
          ...
1677   -0.183977
1678   -0.182367
1679   -0.183172
1680   -0.181400
1681   -0.181306
Name: 34, Length: 1682, dtype: float64
```

In [44]:

```
# From recomd list select hightest rated film which would like by the user. Based on User p
highest_Rated=[]
input_user_pre=pd.DataFrame(user_pred_Trans[input_user])
input_user_pred=input_user_pre.T
for re in recom:
    value=input_user_pred[re].values
    if(value>=1):
        highest_Rated.append(re)
```

In [45]:

```
len(highest_Rated)
```

Out[45]:

27

In [46]:

```
# Now we give movieid respective movie list
i_cols = ['movie id', 'movie title' ,'release date','video release date', 'IMDb URL', 'unkr
'Animation', 'Children\'s', 'Comedy', 'Crime', 'Documentary', 'Drama', 'Fantasy',
'Film-Noir', 'Horror', 'Musical', 'Mystery', 'Romance', 'Sci-Fi', 'Thriller', 'War', 'Weste
items = pd.read_csv('ml-100k/u.item', sep='|', names=i_cols,
encoding='latin-1')
```

In [47]:

```
#Creating Movie List based on recom movieid
movie_title=[]
for movieid in highest_Rated:
    mov=items[items['movie id']==movieid]['movie title'].values
    movie_title.append(mov)
```

In [48]:

```
movie_title
```

Out[48]:

```
[array(['Seven (Se7en) (1995)'], dtype=object),
 array(['I.Q. (1994)'], dtype=object),
 array(['Santa Clause, The (1994)'], dtype=object),
 array(['Free Willy (1993)'], dtype=object),
 array(['Sleepless in Seattle (1993)'], dtype=object),
 array(['Aladdin (1992)'], dtype=object),
 array(['Dances with Wolves (1990)'], dtype=object),
 array(['Snow White and the Seven Dwarfs (1937)'], dtype=object),
 array(['Spitfire Grill, The (1996)'], dtype=object),
 array(['Private Benjamin (1980)'], dtype=object),
 array(['Empire Strikes Back, The (1980)'], dtype=object),
 array(['Princess Bride, The (1987)'], dtype=object),
 array(['Apocalypse Now (1979)'], dtype=object),
 array(['GoodFellas (1990)'], dtype=object),
 array(['Henry V (1989)'], dtype=object),
 array(['Sting, The (1973)'], dtype=object),
 array(['Unforgiven (1992)'], dtype=object),
 array(['Field of Dreams (1989)'], dtype=object),
 array(['Breaking the Waves (1996)'], dtype=object),
 array(['Men in Black (1997)'], dtype=object),
 array(['Chasing Amy (1997)'], dtype=object),
 array(['Sense and Sensibility (1995)'], dtype=object),
 array(["Marvin's Room (1996)"], dtype=object),
 array(['In & Out (1997)'], dtype=object),
 array(['Client, The (1994)'], dtype=object),
 array(['Aladdin and the King of Thieves (1996)'], dtype=object),
 array(['Some Like It Hot (1959)'], dtype=object)]
```

In [49]:

```
#Converting into pure list
movie_title_list=[]
for m in movie_title:
    print(m)
    mv=list(m)
    movie_title_list.append(mv)
```

```
['Seven (Se7en) (1995)']
['I.Q. (1994)']
['Santa Clause, The (1994)']
['Free Willy (1993)']
['Sleepless in Seattle (1993)']
['Aladdin (1992)']
['Dances with Wolves (1990)']
['Snow White and the Seven Dwarfs (1937)']
['Spitfire Grill, The (1996)']
['Private Benjamin (1980)']
['Empire Strikes Back, The (1980)']
['Princess Bride, The (1987)']
['Apocalypse Now (1979)']
['GoodFellas (1990)']
['Henry V (1989)']
['Sting, The (1973)']
['Unforgiven (1992)']
['Field of Dreams (1989)']
['Breaking the Waves (1996)']
['Men in Black (1997)']
['Chasing Amy (1997)']
['Sense and Sensibility (1995)']
["Marvin's Room (1996)"]
['In & Out (1997)']
['Client, The (1994)']
['Aladdin and the King of Thieves (1996)']
['Some Like It Hot (1959)']
```

In [50]:

```
#Converting into whole list
import itertools
Final_Recommend_movie=list(itertools.chain.from_iterable(movie_title_list))
```

In [51]:

```
# Checking the common movie list
list(set(recom)&set(input_user_watched_movieid))
```

Out[51]:

```
[]
```

In [52]:

```
def userbased(input_user,user_similarity,user_predictions,similar_user_count,thres):
    #Convert the user_sim table into DataFrame
    user_sim_table=pd.DataFrame(user_similarity)
    #Find similarity user for 78 using cosine table
    similar_input_user= user_sim_table[input_user].sort_values(ascending=True).head(similar
    #Convert in to list
    similar_user_input=list(similar_input_user)
    #Using similar_user_input,can select movie id from ratings table
    similar_user_movieid_list=[]
    for sim_user in similar_user_input:
        sim=list(ratings[ratings['user_id']==sim_user]['movie_id'])
        similar_user_movieid_list.append(sim)
    #Converting as a whole list
    import itertools
    similar_user_movieid_single_list=list(itertools.chain.from_iterable(similar_user_moviei
    #Unique movieid from the list
    Unique_movieid_similar_user=set(similar_user_movieid_single_list)
    #Input user watched movie_list
    input_user_watched_movieid=list(ratings[ratings['user_id']==input_user]['movie_id'].val
    #Create a list which should have recom movieid to the input user
    recom=[]
    for per_id in Unique_movieid_similar_user:
        if(per_id in input_user_watched_movieid):
            pass
        else:
            recom.append(per_id)
    #From recommendation list selecting only hightest rated(predicted) value
    highest_Rated=[]
    user_pred=pd.DataFrame(user_prediction)
    user_pred_Trans=user_pred.T
    input_user_pre=pd.DataFrame(user_pred_Trans[input_user])
    input_user_pred=input_user_pre.T
    for re in recom:
        value=input_user_pred[re].values
        if(value>=thres):
            highest_Rated.append(re)
    i_cols = ['movie id', 'movie title' ,'release date','video release date', 'IMDb URL', '
    'Animation', 'Children\'s', 'Comedy', 'Crime', 'Documentary', 'Drama', 'Fantasy',
    'Film-Noir', 'Horror', 'Musical', 'Mystery', 'Romance', 'Sci-Fi', 'Thriller', 'War', 'W
    items = pd.read_csv('ml-100k/u.item', sep='|', names=i_cols,encoding='latin-1')
    #Creating Movie List based on recom movieid
    movie_title=[]
    for movieid in highest_Rated:
        mov=items[items['movie id']==movieid]['movie title'].values
        movie_title.append(mov)
    #Converting into pure list
    movie_title_list=[]
    for m in movie_title:
        print(m)
        mv=list(m)
        movie_title_list.append(mv)
    #Converting into whole list
    import itertools
    Final_Recommend_movie=list(itertools.chain.from_iterable(movie_title_list))
    print("The common Movie in Recom & User:",list(set(recom)&set(input_user_watched_moviei
    return Final_Recommend_movie
```

In [53]:

```
#def userbased(input_user,user_similarity,user_predictions,similar_user_count,similar_user_
Recommended_movie=userbased(67,user_similarity,user_pred,5,1.5)
```

```
['Professional, The (1994)']
['Dances with Wolves (1990)']
['Snow White and the Seven Dwarfs (1937)']
['Princess Bride, The (1987)']
['Apocalypse Now (1979)']
['Men in Black (1997)']
["Marvin's Room (1996)"]
The common Movie in Recom & User: []
```

In [ ]:

```
len(Recommended_movie)
```