PRACTICUM

In [1]:

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics.pairwise import cosine_similarity
import math
```

```
C:\Users\venka\Anaconda3\lib\site-packages\statsmodels\tools\_testing.py:19: FutureWarning:
pandas.util.testing is deprecated. Use the functions in the public API at pandas.testing instead.
  import pandas.util.testing as tm
```

In [2]:

```
## Read the User Data
userColumns = ['userId', 'age', 'sex', 'occupation', 'zipCode']
userData = pd.read_csv('ml-100k/u.user', sep='|', names=userColumns)
userData.head()
```

Out[2]:

	userId	age	sex	occupation	zipCode
0	1	24	M	technician	85711
1	2	53	F	other	94043
2	3	23	M	writer	32067
3	4	24	M	technician	43537
4	5	33	F	other	15213

In [3]:

```
## Read the Ratings Data
ratingsColumns = ['userId', 'itemId', 'ratings', 'timeStamp']
ratingsData = pd.read_csv('ml-100k/u.data', sep='\t', names=ratingsColumns)
ratingsData
```

Out[3]:

	userId	itemId	ratings	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 [4]:

```
## Read the Movies Data
moviesColumns =
['movieId','movieTitle','releaseDate','videoReleaseDate','IMDbURL','unknown','Action','Adventure',
'Animation','Children','Comedy','Crime','Documentary','Drama','Fantasy',
'Film-Noir','Horror','Musical','Mystery','Romance','Sci-Fi','Thriller', 'War','Western']
moviesData = pd.read_csv('ml-100k/u.item', sep='|',names=moviesColumns, encoding='latin-1')
moviesData
```

Out[4]:

	movieId	movieTitle	releaseDate	videoReleaseDate	IMDbURL	unknown	Action	Adventure	Animation	Children
0	1	Toy Story (1995)	01-Jan-1995	NaN	http://us.imdb.com/M/title-exact?Toy%20Story%2...	0	0	0	1	1
1	2	GoldenEye (1995)	01-Jan-1995	NaN	http://us.imdb.com/M/title-exact?GoldenEye%20(...	0	1	1	0	0
2	3	Four Rooms (1995)	01-Jan-1995	NaN	http://us.imdb.com/M/title-exact?Four%20Rooms%...	0	0	0	0	0
3	4	Get Shorty (1995)	01-Jan-1995	NaN	http://us.imdb.com/M/title-exact?Get%20Shorty%...	0	1	0	0	0
4	5	Copycat (1995)	01-Jan-1995	NaN	http://us.imdb.com/M/title-exact?Copycat%20(1995)	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...
1677	1678	Mat' i syn (1997)	06-Feb-1998	NaN	http://us.imdb.com/M/title-exact?Mat%27+i+syn+...	0	0	0	0	0
1678	1679	B. Monkey (1998)	06-Feb-1998	NaN	http://us.imdb.com/M/title-exact?B%2E+Monkey+(...	0	0	0	0	0
1679	1680	Sliding Doors (1998)	01-Jan-1998	NaN	http://us.imdb.com/Title?Sliding+Doors+(1998)	0	0	0	0	0
1680	1681	You So Crazy (1994)	01-Jan-1994	NaN	http://us.imdb.com/M/title-exact?You%20So%20Cr...	0	0	0	0	0
1681	1682	Scream of Stone (Schrei aus Stein) (1991)	08-Mar-1996	NaN	http://us.imdb.com/M/title-exact?Schrei%20aus%...	0	0	0	0	0

1682 rows × 24 columns

In [5]:

```
ratings_matrix = ratingsData.pivot_table(index=['userId'],columns=['itemId'],values='ratings').rese
t_index(drop=True)
ratings_matrix.fillna( 0, inplace = True )
ratings_matrix.head()
```

Out[5]:

itemId	1	2	3	4	5	6	7	8	9	10	...	1673	1674	1675	1676	1677	1678	1679	1680	1681	1682
0	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	0.0	0.0	0.0	0.0	0.0
1	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	0.0	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	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	0.0	0.0	0.0	0.0	0.0
4	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	0.0	0.0	0.0	0.0	0.0

5 rows × 1682 columns

In [6]:

```
ratings_matrix_mean = ratings_matrix.mean(axis=1)
ratings_matrix_mean
```

Out[6]:

```
0      0.583829
1      0.136742
2      0.089774
3      0.061831
4      0.299049
         ...
938    0.124257
939    0.219976
940    0.052913
941    0.200357
942    0.340666
Length: 943, dtype: float64
```

In [7]:

```
ratings_matrix_centered = ratings_matrix - ratings_matrix_mean
ratings_matrix_centered = ratings_matrix_centered.where((pd.notnull(ratings_matrix_centered)),0)
ratings_matrix_noNull = ratings_matrix.where((pd.notnull(ratings_matrix)),0)
ratings_matrix_centered.head()
```

Out[7]:

	0	1	2	3	4	5	6	7	8	9	...	1673	1674	1675	1676	1677	1
0	0.0	4.863258	2.910226	3.938169	2.700951	2.543995	4.049941	3.866825	0.944114	4.539834	...	0.0	0.0	0.0	0.0	0.0	
1	0.0	3.863258	-0.089774	-0.061831	-0.299049	-0.456005	-0.950059	-0.133175	-0.055886	-0.460166	...	0.0	0.0	0.0	0.0	0.0	
2	0.0	-0.136742	-0.089774	-0.061831	-0.299049	-0.456005	-0.950059	-0.133175	-0.055886	-0.460166	...	0.0	0.0	0.0	0.0	0.0	
3	0.0	-0.136742	-0.089774	-0.061831	-0.299049	-0.456005	-0.950059	-0.133175	-0.055886	-0.460166	...	0.0	0.0	0.0	0.0	0.0	
4	0.0	3.863258	2.910226	-0.061831	-0.299049	-0.456005	-0.950059	-0.133175	-0.055886	-0.460166	...	0.0	0.0	0.0	0.0	0.0	

5 rows × 1683 columns

In [8]:

```
movieFeatures = moviesData.iloc[:,5:24]
movieFeatures
```

Out[8]:

	unknown	Action	Adventure	Animation	Children	Comedy	Crime	Documentary	Drama	Fantasy	Film-Noir	Horror	Musical	Myst
0	0	0	0	1	1	1	0	0	0	0	0	0	0	
1	0	1	1	0	0	0	0	0	0	0	0	0	0	
2	0	0	0	0	0	0	0	0	0	0	0	0	0	
3	0	1	0	0	0	1	0	0	1	0	0	0	0	
4	0	0	0	0	0	0	1	0	1	0	0	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1677	0	0	0	0	0	0	0	0	1	0	0	0	0	
1678	0	0	0	0	0	0	0	0	0	0	0	0	0	
1679	0	0	0	0	0	0	0	0	1	0	0	0	0	
1680	0	0	0	0	0	1	0	0	0	0	0	0	0	
1681	0	0	0	0	0	0	0	0	1	0	0	0	0	

1682 rows × 19 columns

In [9]:

```
user_Profile = np.dot(ratings_matrix_noNull,movieFeatures)
```

In [10]:

```
cosine = cosine_similarity(user_Profile,moviesData[moviesData.movieId==95].iloc[:,5:24])
```

In [11]:

```
print("Cosine Similarity for the User 200 is :", cosine[199])
print("Cosine Distance for the User 200 is :", 1-cosine[199])
print('*'*50)
print("Cosine Similarity for the User 15 is :", cosine[199])
print("Cosine Distance for the User 15 is :", 1-cosine[199])
```

```
Cosine Similarity for the User 200 is : [0.38745727]
Cosine Distance for the User 200 is : [0.61254273]
**************************************************
Cosine Similarity for the User 15 is : [0.38745727]
Cosine Distance for the User 15 is : [0.61254273]
```

The Movie 95 is suggested to the User 200 as he has the highest similartity and lowest distance

Problem 2.2

In [12]:

```
userData.head()
```

Out[12]:

	userId	age	sex	occupation	zipCode
0	1	24	M	technician	85711
1	2	53	F	other	94043
2	3	23	M	writer	32067
3	4	24	M	technician	43537
4	5	33	F	other	15213

In [13]:

```
ratingsData.head()
```

Out[13]:

	userId	itemId	ratings	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

In [14]:

```
ratings_matrix1 = ratingsData.pivot_table(index=['userId'],columns=['itemId'],values='ratings').res
et_index(drop=True)
ratings_matrix1.fillna( 0, inplace = True )
ratings_matrix1.head()
```

Out[14]:

itemId	1	2	3	4	5	6	7	8	9	10	...	1673	1674	1675	1676	1677	1678	1679	1680	1681	1682
0	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	0.0	0.0	0.0	0.0	0.0

itemId	1	2	3	4	5	6	7	8	9	10	...	1673	1674	1675	1676	1677	1678	1679	1680	1681	1682
1	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	0.0	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	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	0.0	0.0	0.0	0.0	0.0
4	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	0.0	0.0	0.0	0.0	0.0

5 rows × 1682 columns

In [15]:

```
ratings_matrix_mean1 = ratings_matrix1.mean(axis=1)
ratings_matrix_mean1
```

Out[15]:

```
0       0.583829
1       0.136742
2       0.089774
3       0.061831
4       0.299049
          ...
938     0.124257
939     0.219976
940     0.052913
941     0.200357
942     0.340666
Length: 943, dtype: float64
```

In [16]:

```
ratings_matrix_centered1 = ratings_matrix1 - ratings_matrix_mean1
ratings_matrix_centered1 = ratings_matrix_centered1.where((pd.notnull(ratings_matrix_centered1)),0
)
ratings_matrix_noNull1 = ratings_matrix1.where((pd.notnull(ratings_matrix)),0)
ratings_matrix_centered1.head()
```

Out[16]:

	0	1	2	3	4	5	6	7	8	9	...	1673	1674	1675	1676	1677	1
0	0.0	4.863258	2.910226	3.938169	2.700951	2.543995	4.049941	3.866825	0.944114	4.539834	...	0.0	0.0	0.0	0.0	0.0	
1	0.0	3.863258	-0.089774	-0.061831	-0.299049	-0.456005	-0.950059	-0.133175	-0.055886	-0.460166	...	0.0	0.0	0.0	0.0	0.0	
2	0.0	-0.136742	-0.089774	-0.061831	-0.299049	-0.456005	-0.950059	-0.133175	-0.055886	-0.460166	...	0.0	0.0	0.0	0.0	0.0	
3	0.0	-0.136742	-0.089774	-0.061831	-0.299049	-0.456005	-0.950059	-0.133175	-0.055886	-0.460166	...	0.0	0.0	0.0	0.0	0.0	
4	0.0	3.863258	2.910226	-0.061831	-0.299049	-0.456005	-0.950059	-0.133175	-0.055886	-0.460166	...	0.0	0.0	0.0	0.0	0.0	

5 rows × 1683 columns

In [17]:

```
ratings_matrix_New = ratings_matrix1.where((pd.notnull(ratings_matrix1)),0)
user_1 = ratings_matrix_New.iloc[:1,]
user_1
```

Out[17]:

itemId	1	2	3	4	5	6	7	8	9	10	...	1673	1674	1675	1676	1677	1678	1679	1680	1681	1682
0	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	0.0	0.0	0.0	0.0	0.0

1 rows × 1682 columns

In [18]:

```
ratings_matrix_Other = ratings_matrix_New.iloc[1:,]
ratings_matrix_Other.head()
```

Out[18]:

itemId	1	2	3	4	5	6	7	8	9	10	...	1673	1674	1675	1676	1677	1678	1679	1680	1681	1682
1	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	0.0	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	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	0.0	0.0	0.0	0.0	0.0
4	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	0.0	0.0	0.0	0.0	0.0
5	4.0	0.0	0.0	0.0	0.0	0.0	2.0	4.0	4.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

5 rows × 1682 columns

In [19]:

```
cosine1 = cosine_similarity(user_1,ratings_matrix_Other)
top10Users= np.argsort(cosine1[0])[-10:]
top10Values=[]
for i in top10Users:
    top10Values.append(cosine1[0][i])
```

In [20]:

```
count = 0
ratings = [0,0,0,0,0,0,0,0,0,0]
for i in range(len(top10Users)):
    ratings[i] = ratings_matrix_Other[508][top10Users[i]]
    if ratings[i] != 0.0:
        count+=1
sum = math.fsum(ratings)
mean = sum/count
print("Mean: ",mean)
```

```
Mean:  4.0
```

RECITATION

Problem 9.3.1 (C)

In [21]:

```
xp=np.array([[1,1,0,1,0,0,1,0],[0,1,1,1,0,0,0,0],[0,0,0,1,0,1,1,1]])
xp.T
```

Out[21]:

```
array([[1, 0, 0],
       [1, 1, 0],
       [0, 1, 0],
       [1, 1, 1],
       [0, 0, 0],
       [0, 0, 1],
       [1, 0, 1],
       [0, 0, 1]])
```

In [22]:

```
xpDF=pd.DataFrame(xp.T,columns=['A','B','C'])
```

In [23]:

```
xpDF
```

Out[23]:

	A	B	C
0	1	0	0
1	1	1	0
2	0	1	0
3	1	1	1
4	0	0	0
5	0	0	1
6	1	0	1
7	0	0	1

In [24]:

```
from scipy.spatial.distance import jaccard
from scipy.spatial.distance import cosine
```

In [25]:

```
print("Jaccard(A,B)",jaccard(xpDF['A'], xpDF['B']))
print("Jaccard(B,C)",jaccard(xpDF['B'], xpDF['C']))
print("Jaccard(A,C)",jaccard(xpDF['A'], xpDF['C']))
```

```
Jaccard(A,B) 0.6
Jaccard(B,C) 0.8333333333333334
Jaccard(A,C) 0.6666666666666666
```

Problem 9.3.1 (D)

In [26]:

```
print("cosine(A,B)",cosine(xpDF['A'], xpDF['B']))
print("cosine(B,C)",cosine(xpDF['B'], xpDF['C']))
print("cosine(A,C)",cosine(xpDF['A'], xpDF['C']))
```

```
cosine(A,B) 0.42264973081037416
cosine(B,C) 0.7113248654051871
cosine(A,C) 0.5
```

Problem 9.3.1 (E)

In [27]:

```
xp1=np.array([[4,5,0,5,1,0,3,2],[0,3,4,3,1,2,1,0],[2,0,1,3,0,4,5,3]])
xp1.T
```

Out[27]:

```
array([[4, 0, 2],
       [5, 3, 0],
       [0, 4, 1],
       [5, 3, 3],
       [1, 1, 0],
       [0, 2, 4],
       [3, 1, 5],
       [2, 0, 3]])
```

In [28]:

```
xpDF1=pd.DataFrame(xp1.T,columns=['A','B','C'])
xpDF1
```

Out[28]:

	A	B	C
0	4	0	2
1	5	3	0
2	0	4	1
3	5	3	3
4	1	1	0
5	0	2	4
6	3	1	5
7	2	0	3

In [29]:

```
for i in xpDF1.columns:
    xpDF1[i]=xpDF1[xpDF1[i]>0][i]-xpDF1[i].sum(axis=0)/xpDF1[xpDF1[i]>0].shape[0]
xpDF1=xpDF1.fillna(0)
```

In [31]:

```
xpDF1
```

Out[31]:

	A	B	C
0	0.666667	0.000000	-1.0
1	1.666667	0.666667	0.0
2	0.000000	1.666667	-2.0
3	1.666667	0.666667	0.0
4	-2.333333	-1.333333	0.0
5	0.000000	-0.333333	1.0
6	-0.333333	-1.333333	2.0
7	-1.333333	0.000000	0.0

Problem 9.3.1 (F)

In [30]:

```
print("cosine(A,B)",cosine(xpDF1['A'], xpDF1['B']))
print("cosine(B,C)",cosine(xpDF1['B'], xpDF1['C']))
print("cosine(A,C)",cosine(xpDF1['A'], xpDF1['C']))
```

```
cosine(A,B) 0.41569345253185686
cosine(B,C) 1.739573996953447
cosine(A,C) 1.1154700538379252
```