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

In [2]:
```
book_rating=pd.read_csv(r'C:\Users\sutharsan\Downloads\MACHINE LEARNING\project submission\book recom\book rental datasets\book rental datasets\BX-Book-Ratings
book=pd.read_csv(r'C:\Users\sutharsan\Downloads\MACHINE LEARNING\project submission\book recom\book rental datasets\book rental datasets\BX-Books.csv',encoding
user=pd.read_csv(r'C:\Users\sutharsan\Downloads\MACHINE LEARNING\project submission\book recom\book rental datasets\book rental datasets\BX-Users.csv',encoding
recom=pd.read_csv(r'C:\Users\sutharsan\Downloads\MACHINE LEARNING\project submission\book recom\book rental datasets\book rental datasets\Recommend.csv',encod
```

```
C:\Users\sutharsan\AppData\Local\Temp\ipykernel_11260\2868483394.py:2: DtypeWarning: Columns (3) have mixed types. Specify dtype option on import or set low_m
emory=False.
  book=pd.read_csv(r'C:\Users\sutharsan\Downloads\MACHINE LEARNING\project submission\book recom\book rental datasets\book rental datasets\BX-Books.csv',encod
ing='latin-1')
C:\Users\sutharsan\AppData\Local\Temp\ipykernel_11260\2868483394.py:3: DtypeWarning: Columns (0) have mixed types. Specify dtype option on import or set low_m
emory=False.
  user=pd.read_csv(r'C:\Users\sutharsan\Downloads\MACHINE LEARNING\project submission\book recom\book rental datasets\book rental datasets\BX-Users.csv',encod
ing='latin-1')
```

In [3]:
```
book.head(2)
```

Out[3]:

	isbn	book_title	book_author	year_of_publication	publisher
0	195153448	Classical Mythology	Mark P. O. Morford	2002	Oxford University Press
1	2005018	Clara Callan	Richard Bruce Wright	2001	HarperFlamingo Canada

In [4]:
```
user.head(2)
```

Out[4]:

	user_id	Location	Age
0	1	nyc, new york, usa	NaN
1	2	stockton, california, usa	18.0

In [5]:
```
book_rating.head(2)
```

Out[5]:

	user_id	isbn	rating
0	276725	034545104X	0
1	276726	155061224	5

In [6]:
```
recom.head(2)
```

Out[6]:

	196	242	3	881250949
0	186	302	3	891717742
1	22	377	1	878887116

In [7]:
```
print(book.shape)
print(user.shape)
print(book_rating.shape)  # 1.Read the books dataset and explore it
print(recom.shape)
```

```
(271379, 5)
(278859, 3)
(10000, 3)
(99999, 4)
```

In [8]:
```
print(book.isna().sum())
book=book.dropna()    # 2.Clean up NaN values
```

```
isbn                 0
book_title           0
book_author          1
year_of_publication    0
publisher            2
dtype: int64
```

In [9]:
```
print(book.isna().sum())
```

```
isbn                 0
book_title           0
book_author          0
year_of_publication    0
publisher            0
dtype: int64
```

In [10]:
```
print(user.isna().sum())
user=user.dropna()
```

```
user_id          0
Location         1
Age         110763
dtype: int64
```

In [11]:
```
print(user.isna().sum())
```

```
user_id     0
Location    0
Age         0
dtype: int64
```

In [12]:
```
print(book_rating.isna().sum())
```

```
user_id    0
isbn       0
rating     0
dtype: int64
```

In [13]:
```
print(recom.isna().sum())
```

```
196           0
242           0
3             0
881250949     0
dtype: int64
```

In [14]:
```
book_rating.head(10) # 3.Read the data where ratings are given by users
```

Out[14]:

	user_id	isbn	rating
0	276725	034545104X	0
1	276726	155061224	5
2	276727	446520802	0
3	276729	052165615X	3
4	276729	521795028	6
5	276733	2080674722	0
6	276736	3257224281	8
7	276737	600570967	6
8	276744	038550120X	7
9	276745	342310538	10

In [15]:
```
df=pd.merge(book_rating,book,on='isbn')  # 4.Take a quick look at the number of unique users and books
```

In [16]:
```
df.head(2)
```

Out[16]:

	user_id	isbn	rating	book_title	book_author	year_of_publication	publisher
0	276725	034545104X	0	Flesh Tones: A Novel	M. J. Rose	2002	Ballantine Books
1	276726	155061224	5	Rites of Passage	Judith Rae	2001	Heinle

In [17]:
```
nuser=df.user_id.nunique()
nbooks=df.isbn.nunique()
```

In [18]:
```
print('unique',(nuser))
print('unique',(nbooks))
```

```
unique 828
unique 8051
```

In [19]:
```
isbn_list = df.isbn.unique()
print(" Length of isbn List:", len(isbn_list))   # Convert both user_id and ISBN to the ordered list, i.e., from 0...n-1
def convert(isbn):
    itemindex = np.where(isbn_list==isbn)
    return itemindex[0][0]
```

```
 Length of isbn List: 8051
```

In [20]:
```
userid_list = df.user_id.unique()
print(" Length of user_id List:", len(userid_list))
def convert2(user_id):
    itemindex = np.where(userid_list==user_id)
    return itemindex[0][0]
```

```
 Length of user_id List: 828
```

In [21]:
```
df['user_id_order'] = df['user_id'].apply(convert2)
```

In [22]:
```
df['isbn_id'] = df['isbn'].apply(convert)
df.head()
```

Out[22]:

	user_id	isbn	rating	book_title	book_author	year_of_publication	publisher	user_id_order	isbn_id
0	276725	034545104X	0	Flesh Tones: A Novel	M. J. Rose	2002	Ballantine Books	0	0
1	276726	155061224	5	Rites of Passage	Judith Rae	2001	Heinle	1	1
2	276727	446520802	0	The Notebook	Nicholas Sparks	1996	Warner Books	2	2
3	278418	446520802	0	The Notebook	Nicholas Sparks	1996	Warner Books	3	2
4	276729	052165615X	3	Help!: Level 1	Philip Prowse	1999	Cambridge University Press	4	3

In [23]:
```
col_order=['user_id_order','isbn_id','rating','book_title','book_author','year_of_publication','publisher','user_id','isbn']
df=df.reindex(columns=col_order)          # Re-index the columns to build a matrix
df.head()
```

Out[23]:

	user_id_order	isbn_id	rating	book_title	book_author	year_of_publication	publisher	user_id	isbn
0	0	0	0	Flesh Tones: A Novel	M. J. Rose	2002	Ballantine Books	276725	034545104X
1	1	1	5	Rites of Passage	Judith Rae	2001	Heinle	276726	155061224
2	2	2	0	The Notebook	Nicholas Sparks	1996	Warner Books	276727	446520802
3	3	2	0	The Notebook	Nicholas Sparks	1996	Warner Books	278418	446520802
4	4	3	3	Help!: Level 1	Philip Prowse	1999	Cambridge University Press	276729	052165615X

In [25]:
```
# Split your data into two sets (training and testing)
from sklearn.model_selection import train_test_split
train_data,test_data=train_test_split(df,test_size=0.20)
```

In [28]:
```
train_data_matrix=np.zeros((nuser,nbooks))
for line in train_data.itertuples():
    train_data_matrix[line[1]-1, line[2]-1] =line[3]

test_data_matrix=np.zeros((nuser,nbooks))
for line in test_data.itertuples():
    test_data_matrix[line[1]-1,line[2]-1] =line[3]
```

In [49]:
```
from sklearn.metrics.pairwise import pairwise_distances
user_sim=pairwise_distances(train_data_matrix,metric='cosine')
item_sim=pairwise_distances(train_data_matrix.T,metric='cosine')
```

In [50]:
```
user_sim
```

Out[50]:
```
array([[0., 1., 1., ..., 1., 1., 1.],
       [1., 0., 1., ..., 1., 1., 1.],
       [1., 1., 0., ..., 1., 1., 1.],
       ...,
       [1., 1., 1., ..., 0., 1., 1.],
       [1., 1., 1., ..., 1., 0., 1.],
       [1., 1., 1., ..., 1., 1., 0.]])
```

In [51]:
```
item_sim
```

Out[51]:
```
array([[0., 1., 1., ..., 1., 1., 1.],
       [1., 0., 1., ..., 1., 1., 1.],
       [1., 1., 0., ..., 1., 1., 1.],
       ...,
       [1., 1., 1., ..., 0., 1., 1.],
       [1., 1., 1., ..., 1., 0., 1.],
       [1., 1., 1., ..., 1., 1., 0.]])
```

In [60]:
```
def predict(ratings, similarity, type='user'):
    if type == 'user':
        mean_user_rating = ratings.mean(axis=1)
        # We will 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([np.abs(similarity).sum(axis=1)]).T
    elif type == 'item':
        pred = ratings.dot(similarity) / np.array([np.abs(similarity).sum(axis=1)])
    return pred
```

In [63]:
```
#item_prediction = predict(train_data_matrix, item_sim, type='item')
user_pred = predict(train_data_matrix, user_sim, type='user')
```

In [64]:
```
user_pred
```

Out[64]:
```
array([[-0.00201151, -0.00201151,  0.00161606, ...,  0.0088712 ,
        -0.00201151, -0.00201151],
       [-0.00201151, -0.00201151,  0.00161606, ...,  0.0088712 ,
        -0.00201151, -0.00201151],
       [ 0.06290439,  0.06290439,  0.00653245, ...,  0.07378855,
         0.06290439,  0.06290439],
       ...,
       [-0.00201151, -0.00201151,  0.00161606, ...,  0.0088712 ,
        -0.00201151, -0.00201151],
       [-0.00201151, -0.00201151,  0.00161606, ...,  0.0088712 ,
        -0.00201151, -0.00201151],
       [-0.00201151, -0.00201151,  0.00161606, ...,  0.0088712 ,
        -0.00201151, -0.00201151]])
```

In [67]:
```
from sklearn.metrics import mean_squared_error
from math import sqrt

def rmse(prediction, ground_truth):
    prediction = prediction[ground_truth.nonzero()].flatten()
    ground_truth = ground_truth[ground_truth.nonzero()].flatten()
    return sqrt(mean_squared_error(prediction, ground_truth))
```

In [68]:
```
print('User pred RMSE: ' + str(rmse(user_prediction, test_data_matrix)))
```

```
User-based CF RMSE: 7.699249070558175
```

In [ ]:
```

```