In [1]:
```
import os
import pandas as pd
import numpy as np
```

In [2]:
```
for each in os.listdir():
    print(each)
```

```
.ipynb_checkpoints
book rental datasets
book rental datasets.zip
Book_Recommendations-Copy1.ipynb
BOOK_RECOMMEND_SYSTEM-Copy1.ipynb
BOOK_RECOMMEND_SYSTEM.ipynb
BX-Book-Ratings.csv
BX-Books.csv
BX-Users.csv
proj_book_recommend_system.ipynb
Recommend.csv
rough.ipynb
Untitled.ipynb
```

In [3]:
```
rating = "BX-Book-Ratings.csv"
book = "BX-Books.csv"
users = "BX-Users.csv"
```

In [4]:
```
r = pd.read_csv(rating,encoding="latin-1")
b = pd.read_csv(book,encoding="latin-1")
u = pd.read_csv(users,encoding="latin-1")
```

```
C:\Users\david\AppData\Local\Temp\ipykernel_6568\2495360632.py:2: DtypeWarning: Columns (3) have mixed types. Specify dtype opti
on on import or set low_memory=False.
  b = pd.read_csv(book,encoding="latin-1")
C:\Users\david\AppData\Local\Temp\ipykernel_6568\2495360632.py:3: DtypeWarning: Columns (0) have mixed types. Specify dtype opti
on on import or set low_memory=False.
  u = pd.read_csv(users,encoding="latin-1")
```

In [5]:
```
r
```

Out[5]:

	user_id	isbn	rating
0	276725	034545104X	0
1	276726	155061224	5
2	276727	446520802	0
3	276729	052165615X	3
4	276729	521795028	6
...	...	...	...
1048570	250764	451410777	0
1048571	250764	452264464	8
1048572	250764	048623715X	0
1048573	250764	486256588	0
1048574	250764	515069434	0

1048575 rows × 3 columns

In [6]:
```
u
```

Out[6]:

	user_id	Location	Age
0	1	nyc, new york, usa	NaN
1	2	stockton, california, usa	18.0
2	3	moscow, yukon territory, russia	NaN
3	4	porto, v.n.gaia, portugal	17.0
4	5	farnborough, hants, united kingdom	NaN
...	...	...	...
278854	278854	portland, oregon, usa	NaN
278855	278855	tacoma, washington, united kingdom	50.0
278856	278856	brampton, ontario, canada	NaN
278857	278857	knoxville, tennessee, usa	NaN
278858	278858	dublin, n/a, ireland	NaN

278859 rows × 3 columns

In [7]:
```
# 1. Read the books dataset and explore it
b
```

Out[7]:

	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
2	60973129	Decision in Normandy	Carlo D'Este	1991	HarperPerennial
3	374157065	Flu: The Story of the Great Influenza Pandemic...	Gina Bari Kolata	1999	Farrar Straus Giroux
4	393045218	The Mummies of Urumchi	E. J. W. Barber	1999	W. W. Norton & Company
...	...	...	...	...	...
271374	440400988	There's a Bat in Bunk Five	Paula Danziger	1988	Random House Childrens Pub (Mm)
271375	525447644	From One to One Hundred	Teri Sloat	1991	Dutton Books
271376	006008667X	Lily Dale : The True Story of the Town that Ta...	Christine Wicker	2004	HarperSanFrancisco
271377	192126040	Republic (World's Classics)	Plato	1996	Oxford University Press
271378	767409752	A Guided Tour of Rene Descartes' Meditations o...	Christopher Biffle	2000	McGraw-Hill Humanities/Social Sciences/Languages

271379 rows × 5 columns

In [8]:
```
b.shape
```

Out[8]:
```
(271379, 5)
```

In [9]:
```
b.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 271379 entries, 0 to 271378
Data columns (total 5 columns):
 #   Column               Non-Null Count   Dtype 
---  ------               --------------   ----- 
 0   isbn                 271379 non-null  object
 1   book_title           271379 non-null  object
 2   book_author          271378 non-null  object
 3   year_of_publication  271379 non-null  object
 4   publisher            271377 non-null  object
dtypes: object(5)
memory usage: 10.4+ MB
```

In [10]:
```
b.describe().T
```

Out[10]:

	count	unique	top	freq
isbn	271379	271379	195153448	1
book_title	271379	242150	Selected Poems	27
book_author	271378	102042	Agatha Christie	632
year_of_publication	271379	202	2002	17145
publisher	271377	16823	Harlequin	7535

In [11]:
```
# 2. Clean up NaN values
```

In [12]:
```
u.isnull().sum()
```

Out[12]:
```
user_id          0
Location         1
Age         110763
dtype: int64
```

In [13]:
```
# Droping the nan Values.
u = u.dropna(axis=0)
```

In [14]:
```
u.isnull().sum() # Now there is no nan values.
```

Out[14]:
```
user_id     0
Location    0
Age         0
dtype: int64
```

In [15]:
```
# 3. Read the data where ratings are given by users
```

In [16]:
```
r.head()
```

Out[16]:

	user_id	isbn	rating
0	276725	034545104X	0
1	276726	155061224	5
2	276727	446520802	0
3	276729	052165615X	3
4	276729	521795028	6

In [17]:
```
r.describe()
```

Out[17]:

	user_id	rating
count	1.048575e+06	1.048575e+06
mean	1.285089e+05	2.879907e+00
std	7.421876e+04	3.857870e+00
min	2.000000e+00	0.000000e+00
25%	6.339400e+04	0.000000e+00
50%	1.288350e+05	0.000000e+00
75%	1.927790e+05	7.000000e+00
max	2.788540e+05	1.000000e+01

In [18]:
```
# merging the two data set books and rating.
```

In [19]:
```
df = pd.merge(r,b,on='isbn')
```

In [20]:
```
df=df.head(10000)
```

In [21]:
```
# 4. Take a quick look at the number of unique users and books.
```

In [22]:
```
n_users=df['user_id'].nunique()
n_users
```

Out[22]:
```
6292
```

In [23]:
```
n_books=df['isbn'].nunique()
n_books
```

Out[23]:
```
336
```

In [24]:
```
# 5. Convert the user_id variable to numeric numbers in the correct order.
```

In [25]:
```
# convert user_id into the numeric number.
list_userid = df.user_id.unique()
print("length of isbn list: ", len(list_userid))
```

```
length of isbn list:  6292
```

In [26]:
```
def userid_numeric(user_id):
    itemindex = np.where(list_userid==user_id)
    return itemindex[0][0]
```

In [27]:
```
# do the same with ISBN and it into the numeric number.
list_isbn = df.isbn.unique()
print("length of isbn list: ", len(list_isbn))
```

```
length of isbn list:  336
```

In [28]:
```
def isbn_numeric_id(isbn):
    itemindex = np.where(list_isbn==isbn)
    return itemindex[0][0]
```

In [29]:
```
# 6. Convert both user_id and ISBN to the ordered list, i.e., from 0...n-1
```

In [30]:
```
df['user_id_order'] = df['user_id'].apply(userid_numeric)
```

In [31]:
```
df['isbn_order'] = df['isbn'].apply(isbn_numeric_id)
```

In [32]:
```
df.head()
```

Out[32]:

	user_id	isbn	rating	book_title	book_author	year_of_publication	publisher	user_id_order	isbn_order
0	276725	034545104X	0	Flesh Tones: A Novel	M. J. Rose	2002	Ballantine Books	0	0
1	2313	034545104X	5	Flesh Tones: A Novel	M. J. Rose	2002	Ballantine Books	1	0
2	6543	034545104X	0	Flesh Tones: A Novel	M. J. Rose	2002	Ballantine Books	2	0
3	8680	034545104X	5	Flesh Tones: A Novel	M. J. Rose	2002	Ballantine Books	3	0
4	10314	034545104X	9	Flesh Tones: A Novel	M. J. Rose	2002	Ballantine Books	4	0

In [33]:
```
# 7. Re-index the columns to build a matrix.
```

In [34]:
```
ordered_col = ['user_id_order', 'isbn_order', 'rating', 'book_title', 'book_author', 'year_of_publication','publisher',
               'user_id', 'isbn']
df = df.reindex(columns = ordered_col)
```

In [35]:
```
df
```

Out[35]:

	user_id_order	isbn_order	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	0	5	Flesh Tones: A Novel	M. J. Rose	2002	Ballantine Books	2313	034545104X
2	2	0	0	Flesh Tones: A Novel	M. J. Rose	2002	Ballantine Books	6543	034545104X
3	3	0	5	Flesh Tones: A Novel	M. J. Rose	2002	Ballantine Books	8680	034545104X
4	4	0	9	Flesh Tones: A Novel	M. J. Rose	2002	Ballantine Books	10314	034545104X
...	...	...	...	...	...	...	...	...	...
9995	6288	335	0	Wild Animus	Rich Shapero	2004	Too Far	135847	971880107
9996	6289	335	0	Wild Animus	Rich Shapero	2004	Too Far	135865	971880107
9997	6290	335	3	Wild Animus	Rich Shapero	2004	Too Far	135880	971880107
9998	6291	335	9	Wild Animus	Rich Shapero	2004	Too Far	135911	971880107
9999	1093	335	0	Wild Animus	Rich Shapero	2004	Too Far	136010	971880107

10000 rows × 9 columns

In [36]:
```
# 8. Split your data into two sets (training and testing).
```

In [37]:
```
from sklearn.model_selection import train_test_split
train, test = train_test_split(df, test_size=.30, random_state = 10)
```

In [38]:
```
# 9. Make predictions based on user and item variables.
```

In [39]:
```
train_matrix = np.zeros((n_users, n_books))
for line in train.itertuples():
    train_matrix[line[1]-1, line[2]-1] = line[3]

test_matrix = np.zeros((n_users, n_books))
for line in test.itertuples():
    test_matrix[line[1]-1, line[2]-1] = line[3]
```

In [40]:
```
from sklearn.metrics.pairwise import pairwise_distances
user_correlation = pairwise_distances(train_matrix, metric= 'cosine')
```

```
item_correlation = pairwise_distances(train_matrix.T, metric= 'cosine')
```

In [41]:
```
user_correlation
```

Out[41]:
```
array([[0.        , 1.        , 0.61651751, ..., 1.        , 1.        ,
        1.        ],
       [1.        , 0.        , 1.        , ..., 1.        , 1.        ,
        1.        ],
       [0.61651751, 1.        , 0.        , ..., 1.        , 1.        ,
        1.        ],
       ...,
       [1.        , 1.        , 1.        , ..., 0.        , 1.        ,
        1.        ],
       [1.        , 1.        , 1.        , ..., 1.        , 0.        ,
        1.        ],
       [1.        , 1.        , 1.        , ..., 1.        , 1.        ,
        0.        ]])
```

In [42]:
```
def predict(ratings,correlation, type= 'user'):
    if type == 'user':
        mean_user_rating = ratings.mean(axis=1)
        rating_diff = (ratings - mean_user_rating[:,np.newaxis])
        pred = mean_user_rating[:, np.newaxis] + correlation.dot(rating_diff) / np.array([np.abs(correlation).sum(axis=1)]).T
    elif type == 'item':
        pred = ratings.dot(correlation) / np.array([np.abs(correlation).sum(axis=1)])

    return pred
```

In [43]:
```
user_prediction = predict(train_matrix, user_correlation, type = 'user')
item_prediction = predict(train_matrix, item_correlation, type = 'item')
```

In [44]:
```
item_prediction.shape
```

Out[44]:
```
(6292, 336)
```

In [45]:
```
# 10. Use RMSE to evaluate the predictions.
```

In [46]:
```
from sklearn.metrics import mean_squared_error
from math import sqrt
def rmse(prediction, actual):
    prediction = prediction[actual.nonzero()].flatten()
```

```
    actual = actual[actual.nonzero()].flatten()
    return sqrt(mean_squared_error(prediction, actual))
```

In [47]:
```
print('User-based CF RMSE: ' + str(rmse(user_prediction, test_matrix)))
print('Item-based CF RMSE: ' + str(rmse(item_prediction, test_matrix)))
```

```
User-based CF RMSE: 7.864992808743959
Item-based CF RMSE: 8.021869569575163
```

In [ ]:

In [ ]:

In [ ]:

In [ ]: