Book Rental Recommendation

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from matplotlib.pylab import rcParams
rcParams['figure.figsize'] = 15,6
from sklearn.model_selection import train_test_split
```

Read the books dataset and explore it

In [2]:
```
# Loading of dataset with 10000 rows as Out Of Memory error can occur.
Data_Books_Ratings = pd.read_csv('BXBookRatings.csv',encoding ='Latin-1', nrows=10000)
```

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

Out[3]:

	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 [4]:
```
Data_Books_Ratings.tail()
```

Out[4]:

	user_id	isbn	rating
9995	243	425164403	0
9996	243	440224764	0
9997	243	440225701	0
9998	243	440226430	0
9999	243	440234743	0

In [5]:
```
Data_Books_Ratings.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 3 columns):
 #   Column   Non-Null Count  Dtype 
---  ------   --------------  ----- 
 0   user_id  10000 non-null  int64 
 1   isbn     10000 non-null  object
 2   rating   10000 non-null  int64 
dtypes: int64(2), object(1)
memory usage: 234.5+ KB
```

In [6]:
```
Data_Books_Ratings.shape
```

Out[6]:
```
(10000, 3)
```

In [7]:
```
Data_Users = pd.read_csv('BXUsers.csv',encoding ='Latin-1')
```

```
C:\Users\Lenovo\AppData\Local\Temp\ipykernel_9920\1092509303.py:1: DtypeWarning: Colu
mns (0) have mixed types. Specify dtype option on import or set low_memory=False.
  Data_Users = pd.read_csv('BXUsers.csv',encoding ='Latin-1')
```

In [8]:
```
Data_Users.head()
```

Out[8]:

	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

In [9]:
```
Data_Users.tail()
```

Out[9]:

	user_id	Location	Age
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

In [10]:
```
Data_Users.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 278859 entries, 0 to 278858
Data columns (total 3 columns):
 #   Column    Non-Null Count   Dtype  
---  ------    --------------   -----  
 0   user_id   278859 non-null  object 
 1   Location  278858 non-null  object 
 2   Age       168096 non-null  float64
dtypes: float64(1), object(2)
memory usage: 6.4+ MB
```

In [11]:
```
Data_Books = pd.read_csv('BXBooks.csv', encoding ='Latin-1')
```

```
C:\Users\Lenovo\AppData\Local\Temp\ipykernel_9920\429826054.py:1: DtypeWarning: Columns (3) have mixed types. Specify dtype option on import or set low_memory=False.
  Data_Books = pd.read_csv('BXBooks.csv', encoding ='Latin-1')
```

In [12]:
```
Data_Books.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 [13]:
```
Data_Books.head()
```

Out[13]:

	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

In [14]:
```
Data_Books.tail()
```

Out[14]:

	isbn	book_title	book_author	year_of_publication	publisher
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

In [15]:
```
Data_Books.shape
```

Out[15]:
```
(271379, 5)
```

Cleaning NAN Values

In [16]:
```
Data_Books.isnull().sum()
```

Out[16]:
```
isbn                 0
book_title           0
book_author          1
year_of_publication  0
publisher            2
dtype: int64
```

In [17]:
```
Data_Books.dropna(inplace=True)
```

In [18]:
```
Data_Books.isnull().sum()
```

Out[18]:
```
isbn                 0
book_title           0
book_author          0
year_of_publication  0
publisher            0
dtype: int64
```

In [19]:
```
Data_Users.isnull().sum()
```

Out[19]:
```
user_id          0
Location         1
Age         110763
dtype: int64
```

In [20]:
```
Data_Users.shape
```

Out[20]:
```
(278859, 3)
```

In [21]:
```
Data_Users1 = Data_Users.dropna()
```

In [22]:
```
Data_Users1.isnull().any()
```

Out[22]:
```
user_id     False
Location    False
Age         False
dtype: bool
```

In [23]:
```
Data_Books_Ratings.isnull().sum()
```

Out[23]:
```
user_id    0
isbn       0
rating     0
dtype: int64
```

In [24]:
```
Data_Books_Ratings.dropna(inplace=True)
```

In [25]:
```
Data_Books_Ratings.isnull().sum()
```

Out[25]:
```
user_id    0
isbn       0
rating     0
dtype: int64
```

In [26]:
```
Data_Books_Ratings.describe()
```

Out[26]:

	user_id	rating
count	10000.000000	10000.000000
mean	265844.379600	1.974700
std	56937.189618	3.424884
min	2.000000	0.000000
25%	277478.000000	0.000000
50%	278418.000000	0.000000
75%	278418.000000	4.000000
max	278854.000000	10.000000

Read the data where ratings are given by users

In [27]:
```
Data_Books_BooksRatings = pd.merge(Data_Books,Data_Books_Ratings, on ='isbn')
```

In [28]:
```
Data_Books_BooksRatings
```

Out[28]:

	isbn	book_title	book_author	year_of_publication	publisher	user_id	rating
0	195153448	Classical Mythology	Mark P. O. Morford	2002	Oxford University Press	2	0
1	2005018	Clara Callan	Richard Bruce Wright	2001	HarperFlamingo Canada	8	5
2	60973129	Decision in Normandy	Carlo D'Este	1991	HarperPerennial	8	0
3	374157065	Flu: The Story of the Great Influenza Pandemic...	Gina Bari Kolata	1999	Farrar Straus Giroux	8	0
4	393045218	The Mummies of Urumchi	E. J. W. Barber	1999	W. W. Norton & Company	8	0
...	...	...	...	...	...	...	...
8696	767907566	All Elevations Unknown: An Adventure in the He...	Sam Lightner	2001	Broadway Books	278851	5
8697	884159221	Why stop?: A guide to Texas historical roadsid...	Claude Dooley	1985	Lone Star Books	278851	7
8698	912333022	The Are You Being Served? Stories: 'Camping In...	Jeremy Lloyd	1997	Kqed Books	278851	7
8699	1569661057	Dallas Street Map Guide and Directory, 2000 Ed...	Mapsco	1999	American Map Corporation	278851	10
8700	345251547	Mister God This Is Anna	Fynn	1976	Ballantine Books	277187	0

8701 rows × 7 columns

In [29]:
```
Data_Books_BooksRatings.isnull().sum()
```

Out[29]:
```
isbn                 0
book_title           0
book_author          0
year_of_publication  0
publisher            0
user_id              0
rating               0
dtype: int64
```

In [30]:
```
Data_Books_BooksRatings.head()
```

Out[30]:

	isbn	book_title	book_author	year_of_publication	publisher	user_id	rating
0	195153448	Classical Mythology	Mark P. O. Morford	2002	Oxford University Press	2	0
1	2005018	Clara Callan	Richard Bruce Wright	2001	HarperFlamingo Canada	8	5
2	60973129	Decision in Normandy	Carlo D'Este	1991	HarperPerennial	8	0
3	374157065	Flu: The Story of the Great Influenza Pandemic...	Gina Bari Kolata	1999	Farrar Straus Giroux	8	0
4	393045218	The Mummies of Urumchi	E. J. W. Barber	1999	W. W. Norton & Company	8	0

In [31]:
```
Data_Books_BooksRatings.tail()
```

Out[31]:

	isbn	book_title	book_author	year_of_publication	publisher	user_id	rating
8696	767907566	All Elevations Unknown: An Adventure in the He...	Sam Lightner	2001	Broadway Books	278851	5
8697	884159221	Why stop?: A guide to Texas historical roadsid...	Claude Dooley	1985	Lone Star Books	278851	7
8698	912333022	The Are You Being Served? Stories: 'Camping In...	Jeremy Lloyd	1997	Kqed Books	278851	7
8699	1569661057	Dallas Street Map Guide and Directory, 2000 Ed...	Mapsco	1999	American Map Corporation	278851	10
8700	345251547	Mister God This Is Anna	Fynn	1976	Ballantine Books	277187	0

In [32]:
```
Data_Books_BooksRatings.shape
```

Out[32]:
```
(8701, 7)
```

Take a quick look at the number of unique users and books

In [33]:
```
Data_Books_BooksRatings.nunique()
```

Out[33]:
```
isbn                  8051
book_title            7850
book_author           4870
year_of_publication     59
publisher             1364
user_id                828
rating                  11
dtype: int64
```

In [34]:
```
unique_isbn = Data_Books_BooksRatings.isbn.nunique()
```

In [35]:
```
unique_user_id = Data_Books_BooksRatings.user_id.nunique()
```

In [36]:
```
print('Num. of Users: ' + str(unique_isbn))
print('Num of Books: '+str(unique_user_id))
```

```
Num. of Users: 8051
Num of Books: 828
```

Convert ISBN variables to numeric numbers in the correct order

In [37]:
```
#Convert and print length of isbn list
isbn_list = Data_Books_BooksRatings.isbn.unique()
print(" Length of isbn List:", len(isbn_list))
def get_isbn_numeric_id(isbn):
    #print ("  isbn is:" , isbn)
    itemindex = np.where(isbn_list==isbn)
    return itemindex[0][0]
# This is formatted as code
```

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

Convert the user_id variable to numeric numbers in the correct order

In [38]:
```
#Convert and print length of user_id list
userid_list = Data_Books_BooksRatings.user_id.unique()
print(" Length of user_id List:", len(userid_list))
def get_user_id_numeric_id(user_id):
    #print ("  isbn is:" , isbn)
    itemindex = np.where(userid_list==user_id)
    return itemindex[0][0]
```

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

Convert both user_id and ISBN to the ordered list, i.e., from 0...n-1

In [39]:
```
Data_Books_BooksRatings['user_id_order'] = Data_Books_BooksRatings['user_id'].apply(ge
```

In [40]:
```
Data_Books_BooksRatings['isbn_id'] = Data_Books_BooksRatings['isbn'].apply(get_isbn_nu
```

In [41]:
```
Data_Books_BooksRatings.head()
```

Out[41]:

	isbn	book_title	book_author	year_of_publication	publisher	user_id	rating	user_id_
0	195153448	Classical Mythology	Mark P. O. Morford	2002	Oxford University Press	2	0	
1	2005018	Clara Callan	Richard Bruce Wright	2001	HarperFlamingo Canada	8	5	
2	60973129	Decision in Normandy	Carlo D'Este	1991	HarperPerennial	8	0	
3	374157065	Flu: The Story of the Great Influenza Pandemic...	Gina Bari Kolata	1999	Farrar Straus Giroux	8	0	
4	393045218	The Mummies of Urumchi	E. J. W. Barber	1999	W. W. Norton & Company	8	0	

Re-index the columns to build a matrix

In [42]:
```
#Reindexing the columns
new_col_order = ['user_id_order', 'isbn_id', 'rating', 'book_title', 'book_author','ye
Data_Books_BooksRatings = Data_Books_BooksRatings.reindex(columns= new_col_order)
Data_Books_BooksRatings.head()
```

Out[42]:

	user_id_order	isbn_id	rating	book_title	book_author	year_of_publication	publisher	
0	0	0	0	Classical Mythology	Mark P. O. Morford	2002	Oxford University Press	19515
1	1	1	5	Clara Callan	Richard Bruce Wright	2001	HarperFlamingo Canada	200
2	1	2	0	Decision in Normandy	Carlo D'Este	1991	HarperPerennial	6097
3	1	3	0	Flu: The Story of the Great Influenza Pandemic...	Gina Bari Kolata	1999	Farrar Straus Giroux	37415
4	1	4	0	The Mummies of Urumchi	E. J. W. Barber	1999	W. W. Norton & Company	39304

Split your data into two sets (training and testing)

In [43]:
```
#Importing train_test_split model for splittig the data into train and test set
train_data, test_data = train_test_split(Data_Books_BooksRatings , test_size=0.30)
```

In [44]:
```
train_data
```

Out[44]:

	user_id_order	isbn_id	rating	book_title	book_author	year_of_publication	publisher	
7255	716	6607	6	Deutsche Geschichte. Ein Versuch 1. Von den An...	Herbert Rosendorfer	2000	Dtv	3423
6509	776	5869	8	Moon Missions: Mankind's First Voyages to Anot...	William F. Mellberg	1997	Plymouth Press, Ltd	1882
1904	17	1432	0	Justice for Some	Kate Wilhelm	1994	Fawcett Books	449
2108	253	1611	4	The Second Time Around : A Novel	Mary Higgins Clark	2003	Simon & Schuster	743
7877	7	7228	0	The Good Giants and the Bad Pukwudgies	Jean Fritz	1982	Putnam Pub Group (J)	399
...	...	...	...	...	...	...	...	
4572	260	3953	9	A Bright Shining Lie: John Paul Vann and Ameri...	Neil Sheehan	1988	Random House Inc	394
6870	58	6229	0	The Silver Chair (Chronicles of Narnia (Paperb...	C. S. Lewis	1986	Collier Books	20
2471	7	1948	0	Desperation Dinners!	Beverly Mills	1997	Workman Publishing	0761
3169	7	2604	0	Thanksgiving Cats (Read With Me (New York, N.Y...	Jean Marzollo	1999	Scholastic	590
102	51	56	9	A Soldier of the Great War	Mark Helprin	1992	Avon Books	380

6090 rows × 9 columns

In [45]:
```
test_data
```

Out[45]:

	user_id_order	isbn_id	rating	book_title	book_author	year_of_publication	publisher
5169	7	4541	0	To Be Or Not To Be (Harlequin Romance)	Sue Byfield	1983	Harlequin
6886	323	6245	0	Nomadentochter.	Waris Dirie	2002	Blanvalet Verlag GmbH
1763	123	1310	0	Whirlwind (Tyler, Book 1)	Nancy Martin	1992	Harlequin
5050	123	4425	0	A CORNER OF THE VEIL : A Novel	Laurence Cosse	1999	Scribner
3760	16	3168	0	Acquired Tastes	Peter Mayle	1993	Bantam
...	...	...	...	...	...	...	...
5115	701	4487	5	Media Control: The Spectacular Achievements of...	Noam Chomsky	2002	Seven Stories Press
4978	7	4353	0	Into the Land of the Unicorns (Unicorn Chronic...	Bruce Coville	1999	Apple Signature (Scholastic)
1927	5	1452	0	The Accidental Tourist	Anne Tyler	1994	Berkley Publishing Group
7115	2	6471	0	The Road to Compiegne (French Revolution Series)	Jean Plaidy	1988	Pan Macmillan
7968	7	7319	0	Lady Scandal	Rebecca Baldwin	1984	Fawcett Books

2611 rows × 9 columns

Creating train and test matrix

In [46]:

```
#Create user-book matrix for training
train_data_matrix = np.zeros((unique_user_id, unique_isbn))
for line in train_data.itertuples():
    train_data_matrix[line[1]-1, line[2]-1] = line[3]

#Create user-book matrix for testing
test_data_matrix = np.zeros((unique_user_id, unique_isbn))
for line in test_data.itertuples():
    test_data_matrix[line[1]-1, line[2]-1] = line[3]
```

In [47]:
```
#Importing pairwise_distances function
from sklearn.metrics.pairwise import pairwise_distances
user_similarity = pairwise_distances(train_data_matrix, metric='cosine')
item_similarity = pairwise_distances(train_data_matrix.T, metric='cosine')
```

In [48]:
```
user_similarity.round(3)
```

Out[48]:
```
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 [49]:
```
item_similarity.round(3)
```

Out[49]:
```
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.]])
```

Make predictions based on user and item variables

In [50]:
```
#Defining custom function to make predictions
def predict(ratings, similarity, type='user'):
    if type == 'user':
        mean_user_rating = ratings.mean(axis=1)
        #You 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.arr
    elif type == 'item':
        pred = ratings.dot(similarity) / np.array([np.abs(similarity).sum(axis=1)])
    return pred
```

In [ ]:
```
item_prediction = predict(train_data_matrix, item_similarity, type='item')
user_prediction = predict(train_data_matrix, user_similarity, type='user')
```

Use RMSE to evaluate the predictions

In [ ]:
```
#Importing RMSE function
from sklearn.metrics import mean_squared_error
from math import sqrt

#Defining custom function to filter out elements with ground_truth.nonzero
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 [ ]:
```
print('User-based CF RMSE: ' + str(rmse(user_prediction, test_data_matrix)))
print('Item-based CF RMSE: ' + str(rmse(item_prediction, test_data_matrix)))
```

User pridiction and item prediction we are getting same value