DESCRIPTION Book Rent is the largest online and offline book rental chain in India. They provide books of various genres, such as thrillers, mysteries, romances, and science fiction. The company charges a fixed rental fee for a book per month. Lately, the company has been losing its user base. The main reason for this is that users are not able to choose the right books for themselves. The company wants to solve this problem and increase its revenue and profit.

Project Objective: You, as an ML expert, should focus on improving the user experience by personalizing it to the user's needs. You have to model a recommendation engine so that users get recommendations for books based on the behavior of similar users. This will ensure that users are renting the books based on their tastes and traits.

Note: You have to perform user-based collaborative filtering and item-based collaborative filtering.

In [1]:
```
# import libraries
import pandas as pd
import numpy as np
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
#import data
user_data = pd.read_csv('BX-Users.csv', encoding='latin-1')
book_data = pd.read_csv('BX-Books.csv', encoding='latin-1')
rating_data = pd.read_csv('BX-Book-Ratings.csv', encoding='latin-1')
```

In [3]:
```
user_data.shape, book_data.shape, rating_data.shape
```

Out[3]:
```
((278859, 3), (271379, 5), (1048575, 3))
```

In [4]:
```
user_data.head()
```

Out[4]:

	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 [5]:
```
book_data.head()
```

Out[5]:

	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 [6]:
```
rating_data.head()
```

Out[6]:

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

Read the books dataset and explore it

In [7]:
```
book_data.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 [8]:
```
book_data.head()
```

Out[8]:

	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 [9]:
```
book_data.isna().sum()
```

Out[9]:
```
isbn                 0
book_title           0
book_author          1
year_of_publication  0
publisher            2
dtype: int64
```

In [10]:
```
book_data.duplicated().sum()
```

Out[10]:
```
0
```

In [11]:
```
book_data[book_data.publisher.isna()]
```

Out[11]:

	isbn	book_title	book_author	year_of_publication	publisher
128896	193169656X	Tyrant Moon	Elaine Corvidae	2002	NaN
129043	1931696993	Finders Keepers	Linnea Sinclair	2001	NaN

In [12]:
```
book_data[book_data.book_author.isna()]
```

Out[12]:

	isbn	book_title	book_author	year_of_publication	publisher
187700	9627982032	The Credit Suisse Guide to Managing Your Perso...	NaN	1995	Edinburgh Financial Publishing

In [13]:
```
book_data.shape
```

Out[13]:
```
(271379, 5)
```

since there are only 3 NANs , we can remove nulls. No imputing is necessary

In [14]:
```
book_data.dropna(inplace=True)
```

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

Out[15]:
```
(271376, 5)
```

In [16]:
```
# lets group by author name
bkgrp = book_data.groupby('book_author')
```

In [17]:
```
bkgrp.get_group
```

Out[17]:
```
<bound method BaseGroupBy.get_group of <pandas.core.groupby.generic.DataFram
eGroupBy object at 0x7fb3d0cd30a0>>
```

In [18]:
```
book_data[book_data.isbn == '2290048763']
```

Out[18]:

	isbn	book_title	book_author	year_of_publication	publisher
33915	2290048763	Mademoiselle Chambon	Ã?Â?ric Holder	1998	J'ai lu

In [19]:
```
# group by year
bkgrp1 = book_data.groupby('year_of_publication')
```

In [20]:
```
bkgrp1.first()
```

Out[20]:

	isbn	book_title	book_author	publisher	year_of_publication
0	091680013X	Masterpieces of Erotic Photography	David Bailey	Harbor House Publishers	
1901	671397214	JOY OF MUSIC P	Leonard Bernstein	Fireside	
1920	840724551	Agneatha and the Peacocks-Blank Book	Markings	Nelson Communications	
1929	073943828X	Murder at the Manor (Mystery Guild Lost Classi...	Agatha Christie	Dodd Mead & Company	
1930	684717999	Green Hills of Africa (Scribner Classic)	Ernest Hemingway	Collier Books	
...	...	...	...	...	
Luella Hill	096401811X	Solid as a rock \I\" stand: Inspirational poet...	short stories"	1998	
ROBERT A. WILSON	440500702	Schrodinger's Cat Trilogy : \The Universe Next...	\"The Homing Pigeons\""	1988	
Salvador de Madariaga	8423920143	GuÃa del lector del \Quijote\": Ensayo psicol...	14 : Ensayo)"	1976	
Stan Berenstain	039482492X	C is for Clown: A Circus of \C\" Words, (Brigh...	early books for beginning beginners)"	1972	
\"Freedom Song\""	330482750	Three Novels: \A Strange and Sublime Address\"	\"Afternoon Raag\"	Amit Chaudhuri	

202 rows × 4 columns

In [21]:
```
book_data.year_of_publication.unique()
```

Out[21]:
```
array(['2002', '2001', '1991', '1999', '2000', '1993', '1996', '1988',
       '2004', '1998', '1994', '2003', '1997', '1983', '1979', '1995',
       '1982', '1985', '1992', '1986', '1978', '1980', '1952', '1987',
       '1990', '1981', '1989', '1984', '0', '1968', '1961', '1958',
       '1974', '1976', '1971', '1977', '1975', '1965', '1941', '1970',
       '1962', '1973', '1972', '1960', '1966', '1920', '1956', '1959',
       '1953', '1951', '1942', '1963', '1964', '1969', '1954', '1950',
       '1967', '2005', '1957', '1940', '1937', 'John Peterman', '1955',
       '1946', '1936', '1930', '2011', '1925', '1948', '1943', '1947',
       '1945', '1923', '2020', '1939', '1926', '1938', '2030',
       '\\"Freedom Song\\""', '1911', '1904', '1949', 'Frank Muir',
       '1932', '1928', '1929', '1927', '1931', '1914', '2050', '1934',
       '1910', 'ROBERT A. WILSON', '1933', '1902', 'Karen T. Whittenburg',
       '1924', '1921', '1900', '2038', '2026', 'George H. Scherr', '1944',
       '1917', '1901', 'Salvador de Madariaga', '2010',
       'K.C. Constantine', 'Stan Berenstain', '1908', '1906', '1935',
       '1806', 'Francine Pascal', '2021', 'Luella Hill', '2012', '2006',
       'John Alderson Foote', 'DK Publishing Inc', 'Jules Janin',
       'Gallimard', '1909', '2008', '1378', ' &amp', ' Learning"', '1919',
       '1922', '1897', 'Isadora Duncan', '2024', 'Beatrix Potter', '1376',
       '2037', 'Bart Rulon', 'Alan Rich', 2000, 1982, 1983, 1989, 1993,
      1991, 1990, 1998, 1994, 1995, 1986, 1987, 1974, 1984, 0, 1977,
      1996, 1997, 1980, 1988, 2002, 2001, 1981, 1999, 1992, 2003, 2004,
      1972, 1976, 1985, 1978, 1979, 1970, 1962, 1975, 1901, 1973, 1955,
      1971, 1964, 1963, 1958, 1968, 1969, 1966, 1946, 1943, 1967, 1949,
      1965, 1961, 1960, 1930, 1951, 1957, 1959, 1952, 1953, 1956, 1950,
      1954, 1920, 2005, 1940, 1929], dtype=object)
```

In [22]:
```
bkgrp1.get_group('George H. Scherr')
```

Out[22]:

	isbn	book_title	book_author	year_of_publication	publisher
121766	894805959	The Best of the Journal of Irreproducible Resu...	Unfounded Findings\""	George H. Scherr	1989

In [23]:
```
bkgrp1.get_group('Jules Janin')
```

Out[23]:

	isbn	book_title	book_author	year_of_publication	publisher
220624	1874100055	\The Dead Donkey\" &amp	\"The Guillotined Woman\""	Jules Janin	0

The raw data is not very good. But Isbn numbers are unique and is what we most likely need. So lets leave the data as is

In [24]:
```
book_data.isbn.nunique()
```

Out[24]:
```
271376
```

In [25]:
```
book_data.shape
```

Out[25]:
```
(271376, 5)
```

In [26]:
```
# isbn is the primary key and no duplicates in that column.
```

In [26]:
```
book_data.isbn.unique()
```

Out[26]:
```
array(['195153448', '2005018', '60973129', ..., '006008667X', '192126040',
       '767409752'], dtype=object)
```

Read the data where ratings are given by users

In [27]:
```
rating_data.shape
```

Out[27]:
```
(1048575, 3)
```

In [28]:
```
rating_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1048575 entries, 0 to 1048574
Data columns (total 3 columns):
 #   Column   Non-Null Count    Dtype
---  ------   --------------    -----
 0   user_id  1048575 non-null  int64
 1   isbn     1048575 non-null  object
 2   rating   1048575 non-null  int64
dtypes: int64(2), object(1)
memory usage: 24.0+ MB
```

In [29]:
```
rating_data.user_id.isna().sum()
```

Out[29]:
```
0
```

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

Out[30]:

	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 [31]:
```
rating_data.describe()
```

Out[31]:

	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 [32]:
```
# lets take 200k rows as we are running out of memory
ratings = rating_data.head(10000)
```

In [33]:
```
ratings.shape
```

Out[33]:
```
(10000, 3)
```

In [34]:
```
# merge rating and books data frames
merged_data = pd.merge(ratings, book_data, on='isbn')
```

In [35]:
```
merged_data.head()
```

Out[35]:

	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
2	276727	446520802	0	The Notebook	Nicholas Sparks	1996	Warner Books
3	278418	446520802	0	The Notebook	Nicholas Sparks	1996	Warner Books
4	276729	052165615X	3	Help!: Level 1	Philip Prowse	1999	Cambridge University Press

In [36]:
```
merged_data.shape
```

Out[36]:
```
(8701, 7)
```

In [37]:
```
rateUsrGrpby = merged_data.groupby('user_id')
```

In [38]:
```
rateUsrGrpby.first()
```

Out[38]:

user_id	isbn	rating	book_title	book_author	year_of_publication	publisher
2	195153448	0	Classical Mythology	Mark P. O. Morford	2002	Oxford University Press
8	2005018	5	Clara Callan	Richard Bruce Wright	2001	HarperFlamingo Canada
9	440234743	0	The Testament	John Grisham	1999	Dell
10	1841721522	0	New Vegetarian: Bold and Beautiful Recipes for...	Celia Brooks Brown	2001	Ryland Peters & Small Ltd
12	1879384493	10	If I'd Known Then What I Know Now: Why Not Lea...	J. R. Parrish	2003	Cypress House
...	...	...	...	...	...	...
278846	60809833	8	Brave New World	Aldous Huxley	1989	Harpercollins
278849	380698439	9	Behind the Attic Wall (Avon Camelot Books (Pap...	Sylvia Cassedy	1985	HarperTrophy
278851	28630289	0	Frommer's 2000 California (Frommer's Californi...	Erika Lenkert	1999	Frommer's
278852	449907597	8	Dave Barry's Only Travel Guide You'll Ever Need	Dave Barry	1992	Ballantine Books

278854	042516098X	7	Hornet's Nest	Patricia Daniels Cornwell		1998	Berkley Publishing Group

828 rows × 6 columns

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

In [39]:
```
print('unique users', merged_data.user_id.nunique())
print('unique books', merged_data.isbn.nunique())
```

```
unique users 828
unique books 8051
```

Convert ISBN variables to numeric numbers in the correct order

In [40]:
```
# list of unique isbn numbers
isbn_ulist = merged_data.isbn.unique()
```

In [41]:
```
print('Number of unique isbns in final merged dataset' , isbn_ulist.size, is
```

```
Number of unique isbns in final merged dataset 8051 ['034545104X' '155061224
' '446520802' '052165615X' '521795028'
 '2080674722' '038550120X' '425115801' '449006522' '553561618']
```

In [42]:
```
# sort the list
isbn_ulist.sort()
```

In [43]:
```
print(isbn_ulist[:10])
```

```
['000225669X' '002043300X' '002542730X' '003008685X' '003021436X'
 '006008216X' '006015957X' '006016848X' '006019491X' '006020883X']
```

In [45]:
```
# a= a = np.array([1, 2, 3, 4, 8, 6, 7, 3, 9, 10])
#print("All index value of 3 is: ", np.where(a == 3)[0]) ---> prints [2,7]
 #print("First index value of 3 is: ",np.where(a==3)[0][0]) ---> prints 2
```

In [44]:
```
# function to get the index of a isbn
def convert_isbn_order(isbn):
    index = np.where(isbn_ulist==isbn)
    return index[0][0]
```

In [45]:
```
# list of unique user Ids numbers
userid_ulist = merged_data.user_id.unique()
print('Number of unique users in final merged dataset' , userid_ulist.size,
```

```
Number of unique users in final merged dataset 828 [276725 276726 276727 278
418 276729 276733 276744 276746 277427 278026]
```

In [46]:
```
# sort the array
userid_ulist.sort()
print('after sorted ',  userid_ulist.size, userid_ulist[:10])
```

```
after sorted  828 [ 2  8  9 10 12 14 16 17 19 20]
```

In [47]:
```
# function to get index of user
def convert_user_order(user):
    index = np.where(userid_ulist==user)
    return index[0][0]
```

In [48]:
```
# create both user id and isbn index columns
merged_data['user_index'] = merged_data['user_id'].apply(convert_user_order)
```

In [49]:
```
merged_data.size
```

Out[49]:
```
69608
```

In [50]:
```
merged_data.head()
```

Out[50]:

	user_id	isbn	rating	book_title	book_author	year_of_publication	publisher	us
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	
2	276727	446520802	0	The Notebook	Nicholas Sparks	1996	Warner Books	
3	278418	446520802	0	The Notebook	Nicholas Sparks	1996	Warner Books	
4	276729	052165615X	3	Help!: Level 1	Philip Prowse	1999	Cambridge University Press	

In [51]:
```
merged_data['isbn_index'] = merged_data['isbn'].apply(convert_isbn_order)
```

In [52]:
```
merged_data.head()
```

Out[52]:

	user_id	isbn	rating	book_title	book_author	year_of_publication	publisher	us
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	
2	276727	446520802	0	The Notebook	Nicholas Sparks	1996	Warner Books	
3	278418	446520802	0	The Notebook	Nicholas Sparks	1996	Warner Books	
4	276729	052165615X	3	Help!: Level 1	Philip Prowse	1999	Cambridge University Press	

In [53]:
```
merged_data.size
```

Out[53]:
```
78309
```

In [54]:
```
#### ReIndex the columns to build matrix
new_col_order = ['user_index', 'isbn_index', 'rating', 'book_title', 'book_a
merged_data = merged_data.reindex(columns= new_col_order)
merged_data.head()
```

Out[54]:

	user_index	isbn_index	rating	book_title	book_author	year_of_publication	publisher
0	84	112	0	Flesh Tones: A Novel	M. J. Rose	2002	Ballantine Books
1	85	755	5	Rites of Passage	Judith Rae	2001	Heinle
2	86	4481	0	The Notebook	Nicholas Sparks	1996	Warner Books
3	674	4481	0	The Notebook	Nicholas Sparks	1996	Warner Books
4	87	330	3	Help!: Level 1	Philip Prowse	1999	Cambridge University Press

In [55]:
```
####    Split your data into two sets (training and testing)
from sklearn.model_selection import train_test_split
train, test = train_test_split(merged_data, test_size=0.25)
```

In [56]:
```
print('train data size', train.size, ' test data size' , test.size, merged_d
```

```
train data size 58725  test data size 19584 78309
```

Make predictions based on user and item variables

Approach: We will use memory based filtering approach. 1. User- user filtering : We will recommend items that are loked by similar users 2. Item-item filtering: we will recommend items like by users who liked the item that we liked.

In [57]:
```
# unique users
uniq_users = merged_data.user_id.unique().shape[0];
# unique books
uniq_books = merged_data.isbn.unique().shape[0]
```

In [58]:
```
print('unique user count', uniq_users, 'unique book count', uniq_books)
```

```
unique user count 828 unique book count 8051
```

In [59]:
```
#lets build user book matrix for train data
train_matrix = np.zeros((uniq_users, uniq_books))
for line in train.itertuples():
        #[user_id index, movie_id index] = given rating.
        # print(line[0], line[1], line[2], line[3])
    train_matrix[int(str(line[1]))-1, int(str(line[2]))-1] = line[3]
```

In [60]:
```
# Create user-book matrix for testing
test_matrix = np.zeros((uniq_users, uniq_books))
for line in test.itertuples():
    test_matrix[int(str(line[1]))-1, int(str(line[2]))-1] = line[3]
```

In [61]:
```
train_matrix.shape, test_matrix.shape
```

Out[61]:
```
((828, 8051), (828, 8051))
```

In [62]:
```
#import pairwise model
from sklearn.metrics.pairwise import pairwise_distances
```

In [64]:
```
user_similarity = pairwise_distances(train_matrix, metric='cosine')
book_similarity = pairwise_distances(train_matrix.T, metric='cosine')
```

In [65]:
```
# prediction generic function
def predict(matrix, similarity, input_type):
    if input_type == 'user':
        mean_user_rating = train_matrix.mean(axis=1)[:, np.newaxis]
        ratings_diff = (train_matrix - mean_user_rating)
        pred = mean_user_rating + similarity.dot(ratings_diff) / np.array([n
    elif input_type == 'book':
        pred = matrix.dot(similarity) / np.array([np.abs(similarity).sum(axi
    return pred
```

In [66]:
```
book_prediction = predict(train_matrix, book_similarity, 'book')
```

In [67]:
```
user_prediction = predict(train_matrix, user_similarity, 'user')
```

In [68]:
```
print('book_prediction', book_prediction)
```

```
book_prediction [[0.00484472 0.00489393 0.00484472 ... 0.00484472 0.00484472
0.00484472]
 [0.00074534 0.00075291 0.00074534 ... 0.00074534 0.00074534 0.00074534]
 [0.         0.         0.         ... 0.         0.         0.        ]
 ...
 [0.00099379 0.00100388 0.00099379 ... 0.00099379 0.00099379 0.00099379]
 [0.00521739 0.00527038 0.00521739 ... 0.00521739 0.00521739 0.00521739]
 [0.         0.         0.         ... 0.         0.         0.        ]]
```

In [69]:
```
print('user prediction' , user_prediction)
```

```
user prediction [[ 0.00295997  0.03318972  0.00295997 ...  0.00295997  0.002
95997
   0.01021511]
 [-0.00114386  0.02908589 -0.00114386 ... -0.00114386 -0.00114386
   0.00611128]
 [-0.00189001  0.02833974 -0.00189001 ... -0.00189001 -0.00189001
   0.00536513]
 ...
 [-0.00089514  0.02933461 -0.00089514 ... -0.00089514 -0.00089514
   0.00636   ]
 [ 0.00333437  0.03356616  0.00333437 ...  0.00333437  0.00333437
   0.01059   ]
 [-0.00189001  0.02833974 -0.00189001 ... -0.00189001 -0.00189001
   0.00536513]]
```

In [70]:
```
# model evaluation
from sklearn.metrics import mean_squared_error
from math import sqrt
```

In [85]:
```
def rmse(prediction, testdata):
    return sqrt(mean_squared_error(testdata, prediction))
```

In [86]:
```
print('User-based Filtering RMSE: ' + str(rmse(user_prediction, test_matrix))
print('Item-based Filtering RMSE: ' + str(rmse(book_prediction, test_matrix))
```

```
User-based Filtering RMSE: 0.0705716829746829
Item-based Filtering RMSE: 0.07044161438779482
```

Both the approaches giving same performance(RMSE is low and is almost equal). So We can take any one of these approaches.

In [ ]:
```

```