About Book Crossing Dataset

This dataset has been compiled by Cai-Nicolas Ziegler in 2004, and it comprises of three tables for users, books and ratings. Explicit ratings are expressed on a scale from 1-10 (higher values denoting higher appreciation) and implicit rating is expressed by 0.

Reference: http://www2.informatik.uni-freiburg.de/~cziegler/BX/ (http://www2.informatik.uni-freiburg.de/~cziegler/BX/)

Objective

This project entails building a Book Recommender System for users based on user-based and item-based collaborative filtering approaches.

Execute the below cell to load the datasets

In [1]:
```
# Importing the necessary packages

import pandas as pd
import numpy as np
```

In [2]:
```
#Loading data
books = pd.read_csv("books/books.csv", sep=";", error_bad_lines=Fal
books.columns = ['ISBN', 'bookTitle', 'bookAuthor', 'yearOfPublicat

users = pd.read_csv('books/users.csv', sep=';', error_bad_lines=Fal
users.columns = ['userID', 'Location', 'Age']

ratings = pd.read_csv('books/ratings.csv', sep=';', error_bad_lines
ratings.columns = ['userID', 'ISBN', 'bookRating']
```

```
b'Skipping line 6452: expected 8 fields, saw 9\nSkipping line 4366
7: expected 8 fields, saw 10\nSkipping line 51751: expected 8 fiel
ds, saw 9\n'
b'Skipping line 92038: expected 8 fields, saw 9\nSkipping line 104
319: expected 8 fields, saw 9\nSkipping line 121768: expected 8 fi
elds, saw 9\n'
b'Skipping line 144058: expected 8 fields, saw 9\nSkipping line 15
0789: expected 8 fields, saw 9\nSkipping line 157128: expected 8 f
ields, saw 9\nSkipping line 180189: expected 8 fields, saw 9\nSkip
ping line 185738: expected 8 fields, saw 9\n'
b'Skipping line 209388: expected 8 fields, saw 9\nSkipping line 22
0626: expected 8 fields, saw 9\nSkipping line 227933: expected 8 f
ields, saw 11\nSkipping line 228957: expected 8 fields, saw 10\nSk
ipping line 245933: expected 8 fields, saw 9\nSkipping line 251296
: expected 8 fields, saw 9\nSkipping line 259941: expected 8 field
s, saw 9\nSkipping line 261529: expected 8 fields, saw 9\n'
C:\ProgramData\Anaconda3\lib\site-packages\IPython\core\interactiv
eshell.py:3049: DtypeWarning: Columns (3) have mixed types. Specif
y dtype option on import or set low_memory=False.
  interactivity=interactivity, compiler=compiler, result=result)
```

Check no.of records and features given in each dataset

In [3]:
```
print(books.shape)
```

```
(271360, 8)
```

In [4]:
```
print(users.shape)
```

```
(278858, 3)
```

In [5]:
```
print(ratings.shape)
```

```
(1149780, 3)
```

Exploring books dataset

In [6]:
```
books.head()
```

Out[6]:

	ISBN	bookTitle	bookAuthor	yearOfPublication	publisher	
0	0195153448	Classical Mythology	Mark P. O. Morford	2002	Oxford University Press	http://images.ama
1	0002005018	Clara Callan	Richard Bruce Wright	2001	HarperFlamingo Canada	http://images.ama
2	0060973129	Decision in Normandy	Carlo D'Este	1991	HarperPerennial	http://images.ama
3	0374157065	Flu: The Story of the Great Influenza Pandemic...	Gina Bari Kolata	1999	Farrar Straus Giroux	http://images.ama
4	0393045218	The Mummies of Urumchi	E. J. W. Barber	1999	W. W. Norton & Company	http://images.ama

Drop last three columns containing image URLs which will not be required for analysis

In [7]:
```
books = books.drop(['imageUrlS', 'imageUrlM', 'imageUrlL'], axis =
```

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

Out[8]:

	ISBN	bookTitle	bookAuthor	yearOfPublication	publisher
0	0195153448	Classical Mythology	Mark P. O. Morford	2002	Oxford University Press
1	0002005018	Clara Callan	Richard Bruce Wright	2001	HarperFlamingo Canada
2	0060973129	Decision in Normandy	Carlo D'Este	1991	HarperPerennial
3	0374157065	Flu: The Story of the Great Influenza Pandemic...	Gina Bari Kolata	1999	Farrar Straus Giroux
4	0393045218	The Mummies of Urumchi	E. J. W. Barber	1999	W. W. Norton & Company

yearOfPublication

Check unique values of yearOfPublication

In [9]:
```
books['yearOfPublication'].unique()
```

Out[9]:
```
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, 19
74,
       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, 1955, 1946, 1936, 1930,
2011,
       1925, 1948, 1943, 1947, 1945, 1923, 2020, 1939, 1926, 1938,
2030,
       1911, 1904, 1949, 1932, 1928, 1929, 1927, 1931, 1914, 2050,
1934,
       1910, 1933, 1902, 1924, 1921, 1900, 2038, 2026, 1944, 1917,
1901,
       2010, 1908, 1906, 1935, 1806, 2021, '2000', '1995', '1999',
'2004',
       '2003', '1990', '1994', '1986', '1989', '2002', '1981', '19
93',
       '1983', '1982', '1976', '1991', '1977', '1998', '1992', '19
96',
       '0', '1997', '2001', '1974', '1968', '1987', '1984', '1988'
,
       '1963', '1956', '1970', '1985', '1978', '1973', '1980', '19
79',
       '1975', '1969', '1961', '1965', '1939', '1958', '1950', '19
53',
       '1966', '1971', '1959', '1972', '1955', '1957', '1945', '19
60',
       '1967', '1932', '1924', '1964', '2012', '1911', '1927', '19
48',
       '1962', '2006', '1952', '1940', '1951', '1931', '1954', '20
05',
       '1930', '1941', '1944', 'DK Publishing Inc', '1943', '1938'
,
       '1900', '1942', '1923', '1920', '1933', 'Gallimard', '1909'
,
       '1946', '2008', '1378', '2030', '1936', '1947', '2011', '20
20',
       '1919', '1949', '1922', '1897', '2024', '1376', '1926', '20
37'],
      dtype=object)
```

As it can be seen from above that there are some incorrect entries in this field. It looks like Publisher names 'DK Publishing Inc' and 'Gallimard' have been incorrectly loaded as yearOfPublication in dataset due to some errors in csv file.

Also some of the entries are strings and same years have been entered as numbers in some places. We will try to fix these things in the coming questions.

Check the rows having 'DK Publishing Inc' as yearOfPublication

In [10]:
```
books.loc[books['yearOfPublication'] == 'DK Publishing Inc', :]
```

Out[10]:

	ISBN	bookTitle	bookAuthor	yearOfPublication	
209538	078946697X	DK Readers: Creating the X-Men, How It All Beg...	2000	DK Publishing Inc	http://images.amazon.com/imag
221678	0789466953	DK Readers: Creating the X-Men, How Comic Book...	2000	DK Publishing Inc	http://images.amazon.com/imag

Drop the rows having 'DK Publishing Inc' and 'Gallimard' as yearOfPublication

In [11]:
```
books = books[books['yearOfPublication'] != 'DK Publishing Inc']
```

In [12]:
```
books = books[books['yearOfPublication'] != 'Gallimard']
```

In [13]:
```
books.yearOfPublication.unique()
```

Out[13]:
```
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, 19
74,
       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, 1955, 1946, 1936, 1930,
2011,
       1925, 1948, 1943, 1947, 1945, 1923, 2020, 1939, 1926, 1938,
2030,
       1911, 1904, 1949, 1932, 1928, 1929, 1927, 1931, 1914, 2050,
1934,
       1910, 1933, 1902, 1924, 1921, 1900, 2038, 2026, 1944, 1917,
1901,
       2010, 1908, 1906, 1935, 1806, 2021, '2000', '1995', '1999',
'2004',
       '2003', '1990', '1994', '1986', '1989', '2002', '1981', '19
93',
       '1983', '1982', '1976', '1991', '1977', '1998', '1992', '19
96',
       '0', '1997', '2001', '1974', '1968', '1987', '1984', '1988'
,
       '1963', '1956', '1970', '1985', '1978', '1973', '1980', '19
79',
       '1975', '1969', '1961', '1965', '1939', '1958', '1950', '19
53',
       '1966', '1971', '1959', '1972', '1955', '1957', '1945', '19
60',
       '1967', '1932', '1924', '1964', '2012', '1911', '1927', '19
48',
       '1962', '2006', '1952', '1940', '1951', '1931', '1954', '20
05',
       '1930', '1941', '1944', '1943', '1938', '1900', '1942', '19
23',
       '1920', '1933', '1909', '1946', '2008', '1378', '2030', '19
36',
       '1947', '2011', '2020', '1919', '1949', '1922', '1897', '20
24',
       '1376', '1926', '2037'], dtype=object)
```

Change the datatype of yearOfPublication to 'int'

In [14]:
```
books['yearOfPublication'] = books['yearOfPublication'].astype('int
```

In [15]:
```
books.dtypes
```

Out[15]:
```
ISBN                 object
bookTitle            object
bookAuthor           object
yearOfPublication     int32
publisher            object
dtype: object
```

Drop NaNs in 'publisher' column

In [16]:
```
books['publisher'].dropna(axis = 0, inplace = True)
```

Exploring Users dataset

In [17]:
```
print(users.shape)
users.head()
```

```
(278858, 3)
```

Out[17]:

	userID	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

Get all unique values in ascending order for column Age

In [18]:
```
print(sorted(users['Age'].unique()))
```

```
[nan, 0.0, 1.0, 2.0, 3.0, 4.0, 5.0, 6.0, 7.0, 8.0, 9.0, 10.0, 11.0
, 12.0, 13.0, 14.0, 15.0, 16.0, 17.0, 18.0, 19.0, 20.0, 21.0, 22.0
, 23.0, 24.0, 25.0, 26.0, 27.0, 28.0, 29.0, 30.0, 31.0, 32.0, 33.0
, 34.0, 35.0, 36.0, 37.0, 38.0, 39.0, 40.0, 41.0, 42.0, 43.0, 44.0
, 45.0, 46.0, 47.0, 48.0, 49.0, 50.0, 51.0, 52.0, 53.0, 54.0, 55.0
, 56.0, 57.0, 58.0, 59.0, 60.0, 61.0, 62.0, 63.0, 64.0, 65.0, 66.0
, 67.0, 68.0, 69.0, 70.0, 71.0, 72.0, 73.0, 74.0, 75.0, 76.0, 77.0
, 78.0, 79.0, 80.0, 81.0, 82.0, 83.0, 84.0, 85.0, 86.0, 87.0, 88.0
, 89.0, 90.0, 91.0, 92.0, 93.0, 94.0, 95.0, 96.0, 97.0, 98.0, 99.0
, 100.0, 101.0, 102.0, 103.0, 104.0, 105.0, 106.0, 107.0, 108.0, 1
09.0, 110.0, 111.0, 113.0, 114.0, 115.0, 116.0, 118.0, 119.0, 123.
0, 124.0, 127.0, 128.0, 132.0, 133.0, 136.0, 137.0, 138.0, 140.0,
141.0, 143.0, 146.0, 147.0, 148.0, 151.0, 152.0, 156.0, 157.0, 159
.0, 162.0, 168.0, 172.0, 175.0, 183.0, 186.0, 189.0, 199.0, 200.0,
201.0, 204.0, 207.0, 208.0, 209.0, 210.0, 212.0, 219.0, 220.0, 223
.0, 226.0, 228.0, 229.0, 230.0, 231.0, 237.0, 239.0, 244.0]
```

Age column has some invalid entries like nan, 0 and very high values like 100 and above

Values below 5 and above 90 do not make much sense for our book rating case...hence replace these by NaNs

In [19]:
```
users[(users['Age'] < 5) | (users['Age'] > 90)] = np.nan
```

Replace null values in column `Age` with mean

In [20]:
```
users['Age'] = users['Age'].fillna(users['Age'].mean())
```

Change the datatype of `Age` to `int`

In [21]:
```
users['Age'] = users['Age'].astype('int')
```

In [22]:
```
print(sorted(users.Age.unique()))
```

```
[5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22
, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38,
39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55
, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71,
72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88
, 89, 90]
```

Exploring the Ratings Dataset

check the shape

In [23]:
```
ratings.shape
```

Out[23]: (1149780, 3)

In [24]:
```
n_users = users.shape[0]
n_books = books.shape[0]
```

In [25]:
```
ratings.head(5)
```

Out[25]:

	userID	ISBN	bookRating
0	276725	034545104X	0
1	276726	0155061224	5
2	276727	0446520802	0
3	276729	052165615X	3
4	276729	0521795028	6

Ratings dataset should have books only which exist in our books dataset. Drop the remaining rows

In [26]:
```
ratings = ratings[ratings['ISBN'].isin(books['ISBN'])]
```

Ratings dataset should have ratings from users which exist in users dataset. Drop the remaining rows

In [27]:
```
ratings = ratings[ratings['userID'].isin(users['userID'])]
```

Consider only ratings from 1-10 and leave 0s in column `bookRating`

In [28]:
```
ratings = ratings[ratings['bookRating'] != 0]
```

Find out which rating has been given highest number of times

In [29]:
```
ratings['bookRating'].mode()
```

Out[29]:
```
0    8
dtype: int64
```

In [30]:
```
# Rating 8 has been given the highest number of times
```

Collaborative Filtering Based Recommendation Systems

For more accurate results only consider users who have rated atleast 100 books

In [31]:
```
counts = ratings['userID'].value_counts()
ratings = ratings[ratings['userID'].isin(counts[counts >= 100].inde
```

Generating ratings matrix from explicit ratings

Note: since NaNs cannot be handled by training algorithms, replace these by 0, which indicates absence of ratings

In [32]:
```
ratings_matrix = ratings.pivot(index = 'ISBN', columns = 'userID',
userID = ratings_matrix.index
ISBN = ratings_matrix.columns
ratings_matrix.head()
```

Out[32]:

userID	2033	2110	2276	4017	4385	5582	6242	6251	6543	6575	...	269566	270
ISBN													
0000913154	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	
0001046438	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	
000104687X	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	
0001047213	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	
0001047973	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	

5 rows × 447 columns

Generate the predicted ratings using SVD with no.of singular values to be 50

In [33]:
```
from scipy.sparse.linalg import svds
```

In [34]:
```
U, sigma, Vt = svds(ratings_matrix, k = 50)
```

In [35]:
```
sigma = np.diag(sigma)
predicted_ratings = np.dot(np.dot(U, sigma), Vt)
```

In [36]:
```
preds_df = pd.DataFrame(predicted_ratings, columns = ratings_matrix
preds_df.head()
```

Out[36]:

userID	2033	2110	2276	4017	4385	5582	6242	625
0	0.025351	-0.010009	-0.015075	-0.021511	0.002056	-0.002058	-0.015933	-0.01088
1	-0.002174	-0.003679	-0.015525	0.035632	-0.008016	0.018564	0.020214	-0.01013
2	-0.001449	-0.002452	-0.010350	0.023755	-0.005344	0.012376	0.013476	-0.00675
3	-0.002174	-0.003679	-0.015525	0.035632	-0.008016	0.018564	0.020214	-0.01013
4	-0.002174	-0.003679	-0.015525	0.035632	-0.008016	0.018564	0.020214	-0.01013

5 rows × 447 columns

Take a particular user_id

Lets find the recommendations for user with id 2110

Note: Execute the below cells to get the variables loaded

In [37]:
```
userID = 2110
```

In [38]:
```
user_id = 2 #2nd row in ratings matrix and predicted matrix
```

Get the predicted ratings for userID 2110 and sort them in descending order

In [42]:
```
print(sorted(preds_df.iloc[2110,:]))
```

```
[-0.059629629552703164, -0.05794245794598958, -0.04965008809218573
, -0.03576989494411327, -0.03491794506558848, -0.03231597647827785
, -0.03098722087649189, -0.02668282563225735, -0.02658558846694912
, -0.02607352158239924, -0.025216287668662622, -0.0250389638009405
5, -0.024570605676136087, -0.02406650774946766, -0.019130046035269
26, -0.01860162783168997, -0.01853185556852167, -0.01821496269826
909, -0.016186172402519426, -0.01277005582589141, -0.0121821679940
06994, -0.011674778845595707, -0.011543008208001285, -0.0113939472
66615178, -0.009421331610492169, -0.00906129249737606, -0.00850725
9088329026, -0.008287026074688666, -0.00806692030656917, -0.007446
```

136780829597, -0.006816024403976212, -0.006224950535765694, -0.005811559550955641, -0.005212563161589721, -0.004993407389226876, -0.00430450335044031, -0.004180632799587168, -0.004134941672937237, -0.003681743839775309, -0.0035107043067718236, -0.003481643597388173636, -0.00303155985304354, -0.0028366952799742987, -0.002444598444225234, -0.0018270423073145242, -0.001614410386626259, -0.0012095754168295304, -0.0011526607003120657, -0.0010692939198048343, -0.0008630698033017127, -0.0007201021980808455, -0.0007112386049946802, -0.0006868404654120295, -0.0004847798297865857, -0.0004032760986379623, -0.0003517218459406623, -0.0003071284276715859, -0.0002739938109506 71, -0.0002472158058255657, -0.0002325367355906204, -0.00022503466556605557, -0.00011651784217305422, -9.159375845470512e-05, -9.045931679951808e-05, -8.991975367496073e-05, -8.165585550708667e-05, -1.7880122733439143e-05, -1.657175476994649e-05, -2.8099154064617924e-06, 3.1448352551025457e-05, 4.097697478300075e-05, 5.794343978687321e-05, 7.714726299797584e-05, 0.00016548499389267986, 0.0001672676767781031612, 0.0001750274157285355 2, 0.00018004081380446153, 0.00024269678248125284, 0.0002544133252743388, 0.00026818098742572243, 0.00027486847010604686, 0.00030547854777264363, 0.0003112754840224199, 0.0003223404225889347, 0.00038652317173267723, 0.00039059029158371 35, 0.00041840748737232287, 0.00041973218350070874, 0.0005094535211400147, 0.0005236907613220253, 0.000565608111526177 3, 0.000603059626523668, 0.0006534658725719612, 0.0006888816334441317, 0.0007017021855409499, 0.0007107469583827958, 0.0007497326727610936, 0.0007989850676262946, 0.0008771130863933185, 0.0009027235219099777, 0.0009074128096884627, 0.0010294222473042 31, 0.0011319731602362454, 0.0011626013768050749, 0.0011700821390790016, 0.0011907992872879906 54, 0.0011951993775329293, 0.0012849333489333666, 0.0012878984670451823, 0.0013075055127599346, 0.0013491504914811792, 0.0014142272862869 77, 0.0014329958205652573, 0.0015052223925607 68, 0.0015072847455260328, 0.0016354128770357 99, 0.0017036326657420382, 0.0017121492539368083, 0.0017140087977431 71, 0.0017330212602641195, 0.0017354629472171104, 0.0017660671510911325, 0.0017875820329947081, 0.0017911977994880624, 0.0018035374050418 72, 0.0018077419484623027, 0.0018180221791272 36, 0.0018328142807925 38, 0.0018838922236996925, 0.0018991257407843 06, 0.0019234368590631754, 0.0019500628741490813, 0.0019546762111472917, 0.0019759134229502184, 0.0019988105621170007, 0.0020059775111332325, 0.0020093016934607825, 0.0020327288122496635, 0.0021108661881239334, 0.0021419308412722908, 0.0021825659203593 17, 0.0021898437366866287, 0.0022946129126433 35, 0.0022976683169582146, 0.0022979094942326 25, 0.0023021339288882 63, 0.0023160358829020056, 0.0023485769490099247, 0.0023837472629061897, 0.0024136766070638463, 0.0024659832063121 37, 0.0024709468770504482, 0.0025041139845512293, 0.0025484502079899125, 0.0026362200516290755, 0.0026390244942793473, 0.0026510363271894603, 0.0027318387121 6774 2, 0.0027940614823442657, 0.0028509943509364 39, 0.0029349732504231605, 0.0029731399213876923, 0.0030486175762815416, 0.0031326022241636 69, 0.0033496797112250627, 0.0034342782516310 73, 0.0034990950250860 59, 0.0035336192937844 1, 0.0035391449724353 82, 0.0036110363292605275, 0.0036509499054465096, 0.0036519204582502528, 0.0036659896705451 44, 0.0036826847805294 75, 0.0037347301255646467, 0.0037502544032422015, 0.0038277707988831665, 0.0039007144354898096, 0.0039739600479453 12, 0.004040347783298017, 0.0040612366114316 95, 0.0041067092211134 41, 0.004128630315942067, 0.0041562265031477984, 0.

004180550507464443, 0.00430239766037118, 0.004360332454446354, 0.0
04377695186843212, 0.0043913385828658795, 0.0043993022659583065, 0
.004466047511156453, 0.004520451378324816, 0.004527620803268827, 0
.004555848681314363, 0.004562009441271891, 0.004593237443416584, 0
.004637938301855009, 0.004772546932201328, 0.004831237529534712, 0
.004956450955907 8115, 0.004991459362594461, 0.005009978278917471,
0.005098545012093187, 0.005116589601 4388635, 0.005206289903335432,
0.005241619503582 6065, 0.005290192205306503, 0.005304914042169 0694
, 0.005448346001909756, 0.005455443401554504, 0.005652247113607732
, 0.005707502689820941, 0.005778376677328325, 0.005822732907213669
, 0.005825322265336119, 0.005861427698891794, 0.005868998390675202
, 0.005875312366344957, 0.005977376831560636, 0.006004892278465404
, 0.006150078718126927, 0.006193873594140 12, 0.006310601264206551,
0.006368052451298 9235, 0.006396059450829333, 0.006413693810412403,
0.006521243859193 6315, 0.006585754237079159, 0.006624230199355227,
0.006704087751033393, 0.006748768117304314, 0.006827443771523493,
0.006931506771207963, 0.007039996434055937, 0.007085200866021 0915,
0.007099251122692154, 0.007102134300816375, 0.007174124518326 7926,
0.007188595611531558, 0.007199754763211084, 0.007204157477471085,
0.007384537805918199, 0.007470400870556887, 0.007494505729303913,
0.007545637751977266, 0.007600911584795946, 0.00767592002681215, 0
.007701364803131164, 0.007731803449906934, 0.007779584314797176, 0
.007845622853960987, 0.007873614229498792, 0.007885917208088095, 0
.007935192522554925, 0.007999369079256824, 0.008050253344844388, 0
.008063034585899081, 0.008088800791051632, 0.00810533721636641, 0.
008106333552719709, 0.0081273760233692, 0.008129110145372037, 0.00
8157924791013762, 0.008399826313073048, 0.008412133631591398, 0.00
842686878289508, 0.008443472312136532, 0.008530445330995296, 0.008
596650240201079, 0.008608716296517192, 0.00862341501282785, 0.0086
7985007775526, 0.008759257663008713, 0.008955784825947903, 0.00896
2390522560844, 0.009067037660965747, 0.009073166952675443, 0.00915
0223718756716, 0.009343679455764064, 0.009344902737316649, 0.00944
8705916852235, 0.009498534034300038, 0.009655407668071013, 0.00971
0821181493367, 0.009874750901016043, 0.009922731429178279, 0.00994
5293438783618, 0.009962675538958429, 0.010093215287065532, 0.01011
4562786071408, 0.01011468373239389, 0.01019874630617083, 0.0102226
50457531528, 0.010297995301992569, 0.010365820408699227, 0.0103749
11777648498, 0.010425658363826572, 0.01043570811206787, 0.01045398
7184597965, 0.010509105111624485, 0.010519767899693609, 0.01071808
815230859, 0.010769229673815496, 0.010831878332826226, 0.010836742
565472129, 0.011012359520822692, 0.011023564621685064, 0.011030352
358776387, 0.011150141315680664, 0.011150791205516583, 0.011196860
686252174, 0.011256938481775 22, 0.011283865781649636, 0.0113055066
21522172, 0.011369206188496683, 0.011473461400791004, 0.0114764870
96830924, 0.011563275281093749, 0.011563782314912783, 0.0115962905
84542611, 0.011607889937849537, 0.011639282755670478, 0.0116918020
64517243, 0.0117184635590503, 0.011811429213428988, 0.011861519125
170564, 0.011869349907096342, 0.012185321164452633, 0.012625500265
636302, 0.012635987942616467, 0.012724095025969254, 0.012801094624
015833, 0.012921527378787802, 0.012928586266080547, 0.012971180454
133662, 0.013329502669443374, 0.013360981965628271, 0.013433957006
012827, 0.0136002698516639, 0.013636416750669322, 0.01367276499372
9665, 0.013876248149533452, 0.014043053873906907, 0.01407380096621

```
3126, 0.014111340037385463, 0.014124317588248819, 0.01429495571339
336, 0.014301389554804877, 0.014368610426068943, 0.014609981569180
45, 0.014665931013482899, 0.015190568578908822, 0.0152400058577440
35, 0.015258413771228605, 0.01536403696602367, 0.01568252858627257
, 0.01585912247126605, 0.015874447813851965, 0.01598355915136001,
0.016112239731415016, 0.016186902866972986, 0.016229257039376378,
0.016457242928152274, 0.016501665997637333, 0.01663594886982123, 0
.016641237568753815, 0.016643030677473927, 0.01672587779836002, 0.
01692270684619924, 0.016973648373247878, 0.01702242398888165, 0.01
722302915233335, 0.017565931104570665, 0.017735261518827074, 0.017
754906587986376, 0.017758496074848175, 0.01793715980978541, 0.0180
13312835555247, 0.01815050706216804, 0.018491042277008592, 0.01874
6461661814796, 0.018765856319433186, 0.018811689871706044, 0.01882
9843995388656, 0.01892435603905002, 0.01936497254909784, 0.0194302
284363729, 0.01943612953306097, 0.019460561918601505, 0.0202186485
92765533, 0.020288536310260778, 0.020323025816771104, 0.0203590623
97444887, 0.020689547501267377, 0.020965408137538007, 0.0210685322
00846105, 0.02114328268207967, 0.02152444271496611, 0.021574268481
859744, 0.021688843313591565, 0.0218474203619801, 0.02201324569847
3915, 0.0220435310426613, 0.023747790246422852, 0.0240376747476294
7, 0.025350483464549895, 0.02568176656170219, 0.026033612085428197
, 0.026088459799114068, 0.026642366619164498, 0.027163040383968943
, 0.027235419519964266, 0.027654452207526044, 0.02771925691011631,
0.027925763627856173, 0.028065367849831296, 0.029461263739887517,
0.029986812248922586, 0.030715846011240097, 0.030813194675605816,
0.031270306832574916, 0.032494016178096746, 0.03264275420612797, 0
.0355202270738676, 0.03570143816719244, 0.0372944149738631, 0.0401
3379606725132, 0.04292898345244319, 0.04397176073615682, 0.0442903
4053542206, 0.04609072984967164, 0.04635627388910159, 0.0465619672
96133565, 0.04674236549043242, 0.048828100124935325, 0.04956126601
930179, 0.052027271944685, 0.053478612287571836, 0.054279370197850
466, 0.05576340468054279, 0.060231413073970654, 0.0604350295554381
06, 0.06465468513812332, 0.06474221599751821, 0.07105113429678657,
0.09160138542792909, 0.13930471283661458]
```

Create a dataframe with name `user_data` containing userID `2110` explicitly interacted books

In [ ]:

In [67]:
```
user_data.head()
```

Out[67]:

	userID	ISBN	bookRating
14448	2110	0060987529	7
14449	2110	0064472779	8
14450	2110	0140022651	10
14452	2110	0142302163	8
14453	2110	0151008116	5

In [68]:
```
user_data.shape
```

Out[68]: (103, 3)

Combine the user_data and and corresponding book data(book_data) in a single dataframe with name user_full_info

In [ ]:

In [70]:
```
book_data.shape
```

Out[70]: (103, 5)

In [71]:
```
book_data.head()
```

Out[71]:

	ISBN	bookTitle	bookAuthor	yearOfPublication	publisher
246	0151008116	Life of Pi	Yann Martel	2002	Harcourt
904	015216250X	So You Want to Be a Wizard: The First Book in ...	Diane Duane	2001	Magic Carpet Books
1000	0064472779	All-American Girl	Meg Cabot	2003	HarperTrophy
1302	0345307674	Return of the Jedi (Star Wars)	James Kahn	1983	Del Rey Books
1472	0671527215	Hitchhikers's Guide to the Galaxy	Douglas Adams	1984	Pocket

In [ ]:

In [73]:
```
user_full_info.head()
```

Out[73]:

	userID	ISBN	bookRating	bookTitle	bookAuthor	yearOfPublication	publishe
0	2110	0060987529	7	Confessions of an Ugly Stepsister : A Novel	Gregory Maguire	2000	Regan Book
1	2110	0064472779	8	All-American Girl	Meg Cabot	2003	HarperTroph
2	2110	0140022651	10	Journey to the Center of the Earth	Jules Verne	1965	Pengu Book
3	2110	0142302163	8	The Ghost Sitter	Peni R. Griffin	2002	Puffin Book
4	2110	0151008116	5	Life of Pi	Yann Martel	2002	Harcou

Get top 10 recommendations for above given userID from the books not already rated by that user

In [ ]:

In [ ]: