1. Read the books dataset and explore it

```
# 1. Read the books dataset and explore it
b
```

	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

2. Clean up NaN values

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

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

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

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

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

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

3. Read the data where ratings are given by users

```
# 3. Read the data where ratings are given by users
```

```
r.head()
```

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

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

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

```
6292
```

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

```
336
```

5. Convert ISBN variables to numeric numbers in the correct order

6.Convert the user_id variable to numeric numbers in the correct order.

```
# 5. Convert the user_id variable to numeric numbers in the correct order.
```

```
# 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
```

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

```
# 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
```

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

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

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

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

```
df.head()
```

	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

8. Re-index the columns to build a matrix

```
df
```

	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

9. Split your data into two sets (training and testing)

```
from sklearn.model_selection import train_test_split
train, test = train_test_split(df, test_size=.30, random_state = 10)
```

10. Make predictions based on user and item variables.

```
user_correlation
```

```
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.        ]])
```

```
item_prediction.shape
```

```
(6292, 336)
```

11.Use RMSE to evaluate the predictions.

```python
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))
```

```python
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
```