In [10]:

```
import numpy as np
import pandas as pd

import warnings
warnings.filterwarnings("ignore")

books = pd.read_csv("BX-Books.csv", sep=';', encoding="latin-1",on_bad_lines='skip')
users = pd.read_csv("BX-Users.csv", sep=';', encoding="latin-1", on_bad_lines='skip')
ratings = pd.read_csv("BX-Book-Ratings.csv", sep=';', encoding="latin-1", on_bad_lines='skip')

books.head()
```

Out[10]:

	ISBN	Book-Title	Book-Author	Year-Of-Publication	Publisher	Image-URL-S	
0	0195153448	Classical Mythology	Mark P. O. Morford	2002	Oxford University Press	http://images.amazon.com/images/P/0195153448.0...	http://images.amazon.com/ima
1	0002005018	Clara Callan	Richard Bruce Wright	2001	HarperFlamingo Canada	http://images.amazon.com/images/P/0002005018.0...	http://images.amazon.com/ima
2	0060973129	Decision in Normandy	Carlo D'Este	1991	HarperPerennial	http://images.amazon.com/images/P/0060973129.0...	http://images.amazon.com/ima
3	0374157065	Flu: The Story of the Great Influenza Pandemic...	Gina Bari Kolata	1999	Farrar Straus Giroux	http://images.amazon.com/images/P/0374157065.0...	http://images.amazon.com/ima
4	0393045218	The Mummies of Urumchi	E. J. W. Barber	1999	W. W. Norton & Company	http://images.amazon.com/images/P/0393045218.0...	http://images.amazon.com/ima

In [ ]:
```
# Preprocessing Data
# Now in the books file, we have some extra columns which are not required for our task like image U
# And we will rename the columns of each file as the name of the column contains space,
# and uppercase letters so we will correct as to make it easy to use.
```

In [11]:
```
books = books[['ISBN', 'Book-Title', 'Book-Author', 'Year-Of-Publication', 'Publisher']]
books.rename(columns = {'Book-Title':'title', 'Book-Author':'author', 'Year-Of-Publication':'year',
users.rename(columns = {'User-ID':'user_id', 'Location':'location', 'Age':'age'}, inplace=True)
ratings.rename(columns = {'User-ID':'user_id', 'Book-Rating':'rating'}, inplace=True)
```

In [ ]:
```
# when you run the above code we can see only 105283 peoples have given a rating among 278000.
# Now we will extract the user ids who have given more than 200 ratings and when we will have
# user ids we will extract the ratings of only this user id from the rating dataframe.
```

In [12]:
```
x = ratings['user_id'].value_counts() > 200
y = x[x].index  #user_ids
print(y.shape)
ratings = ratings[ratings['user_id'].isin(y)]
```

```
(899,)
```

In [ ]:
```
# So 900 users are there who have given 5.2 lakh rating and this we want.
# Now we will merge ratings with books on basis of ISBN so that we will get the
# rating of each user on each book id and the user who has not rated that book id the value will be
```

In [13]:
```
rating_with_books = ratings.merge(books, on='ISBN')
rating_with_books.head()
```

Out[13]:

	user_id	ISBN	rating	title	author	year	publisher
0	277427	002542730X	10	Politically Correct Bedtime Stories: Modern Ta...	James Finn Garner	1994	John Wiley & Sons Inc
1	3363	002542730X	0	Politically Correct Bedtime Stories: Modern Ta...	James Finn Garner	1994	John Wiley & Sons Inc
2	11676	002542730X	6	Politically Correct Bedtime Stories: Modern Ta...	James Finn Garner	1994	John Wiley & Sons Inc
3	12538	002542730X	10	Politically Correct Bedtime Stories: Modern Ta...	James Finn Garner	1994	John Wiley & Sons Inc
4	13552	002542730X	0	Politically Correct Bedtime Stories: Modern Ta...	James Finn Garner	1994	John Wiley & Sons Inc

In [ ]:
```
# Extract books that have received more than 50 ratings.
# Now dataframe size has decreased and we have 4.8 lakh because when we merge the dataframe,
# all the book id-data we were not having. Now we will count the rating of each book so we will
# group data based on title and aggregate based on rating.
```

In [14]:
```
number_rating = rating_with_books.groupby('title')['rating'].count().reset_index()
number_rating.rename(columns= {'rating':'number_of_ratings'}, inplace=True)
final_rating = rating_with_books.merge(number_rating, on='title')
final_rating.shape
final_rating = final_rating[final_rating['number_of_ratings'] >= 50]
final_rating.drop_duplicates(['user_id','title'], inplace=True)
```

In [ ]:
```
# Create Pivot Table
# As we discussed above we will create a pivot table where columns will be user ids,
# the index will be book title and the value is ratings. And the user id who has not
# rated any book will have value as NAN so impute it with zero.
```

In [15]:
```
book_pivot = final_rating.pivot_table(columns='user_id', index='title', values="rating")
book_pivot.fillna(0, inplace=True)
```

In [ ]:

```
# Modeling
# We have prepared our dataset for modeling. we will use the nearest neighbors algorithm which is
# the same as K nearest which is used for clustering based on euclidian distance.

# But here in the pivot table, we have lots of zero values and on clustering,
# this computing power will increase to calculate the distance of zero values so we will convert
# the pivot table to the sparse matrix and then feed it to the model.
```

In [16]:

```
from scipy.sparse import csr_matrix
book_sparse = csr_matrix(book_pivot)
```

In [17]:

```
from sklearn.neighbors import NearestNeighbors
model = NearestNeighbors(algorithm='brute')
model.fit(book_sparse)
```

Out[17]:

```
NearestNeighbors(algorithm='brute')
```

In [18]:

```
distances, suggestions = model.kneighbors(book_pivot.iloc[237, :].values.reshape(1, -1))

```

In [19]:

```
for i in range(len(suggestions)):
    print(book_pivot.index[suggestions[i]])
```

```
Index(['Harry Potter and the Chamber of Secrets (Book 2)',
       'Harry Potter and the Prisoner of Azkaban (Book 3)',
       'Harry Potter and the Goblet of Fire (Book 4)',
       'Harry Potter and the Sorcerer's Stone (Book 1)', 'Exclusive'],
      dtype='object', name='title')
```

In [ ]: