```
import numpy as np
import pandas as pd
```

```
books=pd.read_csv("BX-Books.csv",encoding='latin-1')
users=pd.read_csv("BX-Users.csv",encoding='latin-1')
ratings=pd.read_csv("BX-Book-Ratings.csv",encoding='latin-1',error_bad_lines=False)
```

```
/usr/local/lib/python3.8/dist-packages/IPython/core/interactiveshell.py:3326: DtypeWarning: Columns (3) have mixed types.Specify dtype option on import or set low_memory=False.
  exec(code_obj, self.user_global_ns, self.user_ns)
/usr/local/lib/python3.8/dist-packages/IPython/core/interactiveshell.py:3326: DtypeWarning: Columns (0) have mixed types.Specify dtype option on import or set low_memory=False.
  exec(code_obj, self.user_global_ns, self.user_ns)
/usr/local/lib/python3.8/dist-packages/IPython/core/interactiveshell.py:3326: FutureWarning: The error_bad_lines argument has been deprecated and will be removed in a future version.


  exec(code_obj, self.user_global_ns, self.user_ns)
```

```
users.head()
```

	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

```
ratings.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

```
print(books.shape)
print(ratings.shape)
print(users.shape)
```

```
(271379, 5)
(1048575, 3)
(278859, 3)
```

```
books.isnull().sum()
```

```
isbn                   0
book_title             0
book_author            1
year_of_publication    0
publisher              2
dtype: int64
```

```
users.isnull().sum()
```

```
user_id          0
Location         1
Age         110763
dtype: int64
```

```
ratings.isnull().sum()
```

```
user_id    0
isbn       0
rating     0
dtype: int64
```

```
books.duplicated().sum()
ratings.duplicated().sum()
users.duplicated().sum()
```

```
0
```

Popularity Based Recommender System

```
ratings_with_name = ratings.merge(books,on='isbn')
```

```
num_rating_df = ratings_with_name.groupby('book_title').count()['rating'].reset_index()
num_rating_df.rename(columns={'rating':'num_ratings'},inplace=True)
num_rating_df
```

	book_title	num_ratings
0	A Light in the Storm: The Civil War Diary of ...	4

```
avg_rating_df = ratings_with_name.groupby('book_title').mean()['rating'].reset_index()
avg_rating_df.rename(columns={'rating':'avg_rating'},inplace=True)
avg_rating_df
```

	book_title	avg_rating
0	A Light in the Storm: The Civil War Diary of ...	2.25
1	Always Have Popsicles	0.00
2	Apple Magic (The Collector's series)	0.00
3	Beyond IBM: Leadership Marketing and Finance ...	0.00
4	Clifford Visita El Hospital (Clifford El Gran...	0.00
...	...	...
230233	Ã?Â?l- Connection.	0.00
230234	Ã?Â?lpiraten.	0.00
230235	Ã?Â?rger mit Produkt X. Roman.	5.25
230236	Ã?Â?stlich der Berge.	4.00
230237	Ã?Â?thique en toc	4.00

230238 rows × 2 columns

```
popular_df = num_rating_df.merge(avg_rating_df,on='book_title')
popular_df
```

	book_title	num_ratings	avg_rating
0	A Light in the Storm: The Civil War Diary of ...	4	2.25
1	Always Have Popsicles	1	0.00
2	Apple Magic (The Collector's series)	1	0.00
3	Beyond IBM: Leadership Marketing and Finance ...	1	0.00
4	Clifford Visita El Hospital (Clifford El Gran...	1	0.00
...	...	...	...
230233	Ã?Â?l- Connection.	1	0.00
230234	Ã?Â?lpiraten.	2	0.00
230235	Ã?Â?rger mit Produkt X. Roman.	4	5.25
230236	Ã?Â?stlich der Berge.	2	4.00
230237	Ã?Â?thique en toc	2	4.00

230238 rows × 3 columns

```
popular_df = popular_df[popular_df['num_ratings']>=250].sort_values('avg_rating',ascending=False).head(50)
```

```
popular_df = popular_df.merge(books,on='book_title').drop_duplicates('book_title')[['book_title','book_author','num_ratings','avg_rating']]
```

Collaborative Filtering Based Recommender System

```
x = ratings_with_name.groupby('user_id').count()['rating'] > 200
padhe_likhe_users = x[x].index
```

```
filtered_rating = ratings_with_name[ratings_with_name['user_id'].isin(padhe_likhe_users)]
```

```
y = filtered_rating.groupby('book_title').count()['rating']>=50
famous_books = y[y].index
```

```
final_ratings = filtered_rating[filtered_rating['book_title'].isin(famous_books)]
```

```
pt = final_ratings.pivot_table(index='book_title',columns='user_id',values='rating')
pt.fillna(0,inplace=True)
pt
```

user_id	254	2276	2766	2977	3363	4017	4385	6251	6323	6543	...	249111	249628	249862	249
book_title															
1984	9.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	
1st to Die: A Novel	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	9.0	...	0.0	0.0	0.0	
2nd Chance	0.0	10.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	
4 Blondes	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	
A Bend in the Road	0.0	0.0	7.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
Year of Wonders	0.0	0.0	0.0	7.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	
You Belong To Me	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	
Zen and the Art of Motorcycle Maintenance: An Inquiry into Values	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	

```
from sklearn.metrics.pairwise import cosine_similarity
similarity_scores = cosine_similarity(pt)
```

```
similarity_scores.shape

     (603, 603)
```

```
def recommend(book_name):
    # index fetch
    index = np.where(pt.index==book_name)[0][0]
    similar_items = sorted(list(enumerate(similarity_scores[index])),key=lambda x:x[1],reverse=True)[1:5]

    data = []
    for i in similar_items:
        item = []
        temp_df = books[books['book_title'] == pt.index[i[0]]]
        item.extend(list(temp_df.drop_duplicates('book_title')['book_title'].values))
        item.extend(list(temp_df.drop_duplicates('book_title')['book_author'].values))

        data.append(item)

    return data
```

```
recommend('1984')

     [['Animal Farm', 'George Orwell'],
      ['Brave New World', 'Aldous Huxley'],
      ['The Vampire Lestat (Vampire Chronicles, Book II)', 'ANNE RICE'],
      ["The Handmaid's Tale", 'Margaret Atwood']]
```