Importing Libraries

In [1]:

```
#Basic Libraries
import numpy as np
import pandas as pd

#Visualisation
import matplotlib.pyplot as plt
import seaborn as sns
import missingno as msno

import warnings
warnings.filterwarnings("ignore")
```

Loading Datasets

Books dataset

In [2]:

```
books = pd.read_csv('books.csv')
```

In [3]:

```
books.head(5)
```

Out[3]:

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

In [4]:

```
books[books['Book-Author']=='Agatha Christie']
```

Out[4]:

	ISBN	Book-Title	Book-Author	Year-Of-Publication	Publisher	
1855	0451200993	Sleeping Murder (Miss Marple Mysteries (Paperb...	Agatha Christie	2000	New American Library	http://images.amazon.com/imag
2012	0671555235	A Caribbean Mystery	Agatha Christie	1984	Pocket	http://images.amazon.com/imag
2355	0425173755	Murder on the Orient Express (Hercule Poirot M...	Agatha Christie	2000	Berkley Publishing Group	http://images.amazon.com/imag
2356	0451199871	The Body in the Library (Miss Marple Mysteries...	Agatha Christie	2000	Signet Book	http://images.amazon.com/imag
2569	067170463X	Murder on the Orient Express	Agatha Christie	1978	Pocket Books	http://images.amazon.com/imag
...	...	...	...	...	...	
270319	2702400787	Le couteau sur la nuque	Agatha Christie	1979	Librairie des Champs-ElysÃ? Â©es	http://images.amazon.com/imag
270519	0671494538	ENDLESS NIGHT	Agatha Christie	1983	Pocket	http://images.amazon.com/imag
271094	8427201079	El Misterio De Sittaford	Agatha Christie	0	Editorial Molino	http://images.amazon.com/imag
271095	8427285280	Poirot en Egipto	Agatha Christie	1996	Downtown Book Center	http://images.amazon.com/imag
271250	073943828X	Murder at the Manor (Mystery Guild Lost Classi...	Agatha Christie	1929	Dodd Mead & Company	http://images.amazon.com/imag

632 rows × 8 columns

In [5]:

```
books.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 271360 entries, 0 to 271359
Data columns (total 8 columns):
 #   Column               Non-Null Count   Dtype 
---  ------               --------------   ----- 
 0   ISBN                 271360 non-null  object
 1   Book-Title           271360 non-null  object
 2   Book-Author          271359 non-null  object
 3   Year-Of-Publication  271360 non-null  object
 4   Publisher            271358 non-null  object
 5   Image-URL-S          271360 non-null  object
 6   Image-URL-M          271360 non-null  object
 7   Image-URL-L          271357 non-null  object
dtypes: object(8)
memory usage: 16.6+ MB
```

In [6]:

```
books.describe(include='O')
```

Out[6]:

	ISBN	Book-Title	Book-Author	Year-Of-Publication	Publisher	
count	271360	271360	271359	271360	271358	
unique	271360	242135	102023	202	16807	
top	038075701X	Selected Poems	Agatha Christie	2002	Harlequin	http://images.amazon.com/images/
freq	1	27	632	13903	7535	

Users Dataset

In [7]:

```
users = pd.read_csv('users.csv')
```

In [8]:

```
users.head()
```

Out[8]:

	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 [9]:

```
users.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 278858 entries, 0 to 278857
Data columns (total 3 columns):
 #   Column    Non-Null Count   Dtype  
---  ------    --------------   -----  
 0   User-ID   278858 non-null  int64  
 1   Location  278858 non-null  object 
 2   Age       168096 non-null  float64
dtypes: float64(1), int64(1), object(1)
memory usage: 6.4+ MB
```

In [10]:

```
users.describe()
```

Out[10]:

	User-ID	Age
count	278858.00000	168096.000000
mean	139429.50000	34.751434
std	80499.51502	14.428097
min	1.00000	0.000000
25%	69715.25000	24.000000
50%	139429.50000	32.000000
75%	209143.75000	44.000000
max	278858.00000	244.000000

In [11]:

```
users.describe(include='O')
```

Out[11]:

	Location
count	278858
unique	57339
top	london, england, united kingdom
freq	2506

Ratings Dataset

In [12]:

```
ratings = pd.read_csv('ratings.csv')
```

In [13]:

```
ratings
```

Out[13]:

	User-ID	ISBN	Book-Rating
0	276725	034545104X	0
1	276726	0155061224	5
2	276727	0446520802	0
3	276729	052165615X	3
4	276729	0521795028	6
...	...	...	...
1149775	276704	1563526298	9
1149776	276706	0679447156	0
1149777	276709	0515107662	10
1149778	276721	0590442449	10
1149779	276723	05162443314	8

1149780 rows × 3 columns

In [14]:

```
ratings.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1149780 entries, 0 to 1149779
Data columns (total 3 columns):
 #   Column       Non-Null Count    Dtype
---  ------       --------------    -----
 0   User-ID      1149780 non-null  int64
 1   ISBN         1149780 non-null  object
 2   Book-Rating  1149780 non-null  int64
dtypes: int64(2), object(1)
memory usage: 26.3+ MB
```

In [15]:

```
ratings.describe()
```

Out[15]:

	User-ID	Book-Rating
count	1.149780e+06	1.149780e+06
mean	1.403864e+05	2.866950e+00
std	8.056228e+04	3.854184e+00
min	2.000000e+00	0.000000e+00
25%	7.034500e+04	0.000000e+00
50%	1.410100e+05	0.000000e+00
75%	2.110280e+05	7.000000e+00
max	2.788540e+05	1.000000e+01

In [16]:

```
ratings.describe(include='O')
```

Out[16]:

	ISBN
count	1149780
unique	340556
top	0971880107
freq	2502

Feature Selection

In [17]:

```
#updating column names in books dataset
books= books[['ISBN', 'Book-Title', 'Book-Author', 'Year-Of-Publication', 'Publisher','I
books.columns = ['ISBN', 'title', 'author', 'year', 'publisher','image']
```

In [18]:

```
#updating column names in users dataset
users.rename(columns = {"User-ID":"user"}, inplace=True)
```

In [19]:

```
#updating column names in ratings dataset
ratings.rename(columns = {"User-ID":"user", "Book-Rating": "rating"}, inplace=True)
```

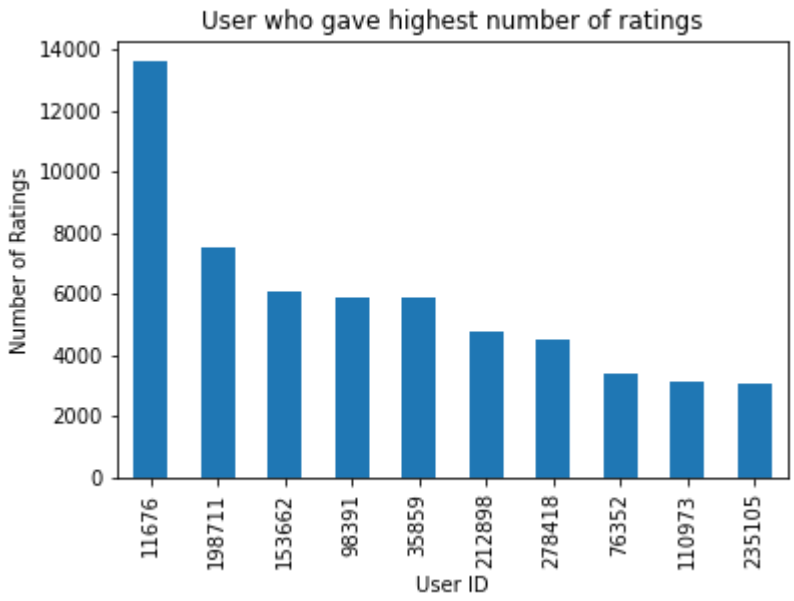
Selecting strategy for Recommender systems

Exploratory Data Analysis

Who gave the rating most number of times?

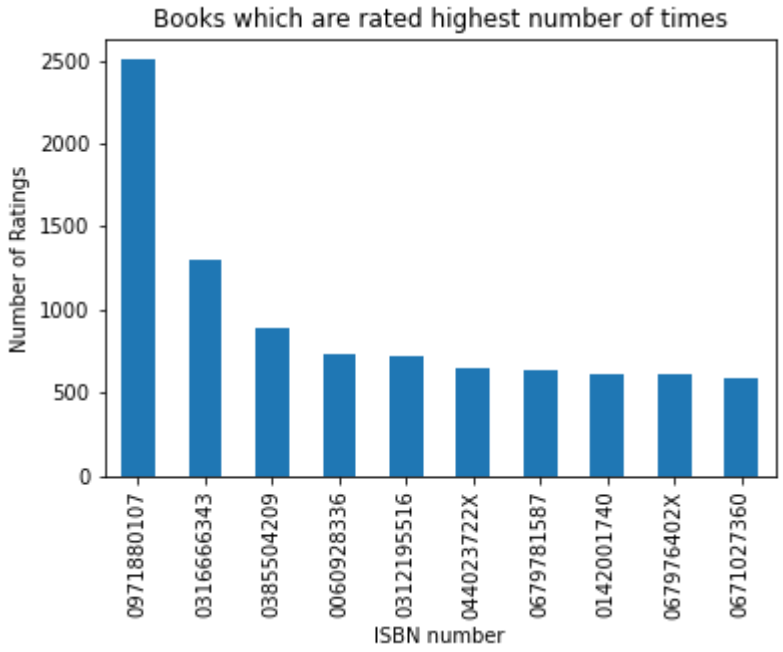
In [20]:

```
ratings.user.value_counts().head(10).plot(kind='bar')
plt.ylabel('Number of Ratings')
plt.xlabel('User ID')
plt.title('User who gave highest number of ratings');
```



In [21]:

```
ratings.ISBN.value_counts().head(10).plot(kind='bar')
plt.ylabel('Number of Ratings')
plt.xlabel('ISBN number')
plt.title('Books which are rated highest number of times');
```



In [22]:

```
books[books['ISBN']=='0971880107']
books[books['ISBN']=='0316666343']
```

Out[22]:

	ISBN	title	author	year	publisher	ir
408	0316666343	The Lovely Bones: A Novel	Alice Sebold	2002	Little, Brown	http://images.amazon.com/images/P/031666634

Books with highest average rating

In [23]:

```
average_ratings = ratings.groupby('ISBN')['rating'].mean().sort_values(ascending=False)
print(average_ratings)
```

```
ISBN
0874477050    10.0
561002010     10.0
0590939874    10.0
1570761914    10.0
56500624X     10.0
              ...
0866838937     0.0
0866839070     0.0
0866839100     0.0
0866839453     0.0
0439216397     0.0
Name: rating, Length: 340556, dtype: float64
```

In [24]:

```
average_ratings[average_ratings==10.0].shape
```

Out[24]:

```
(16762,)
```

There are 16762 books with average ratings 10

In [25]:

```
ratings['rating'].value_counts().sort_values(ascending=False)
```

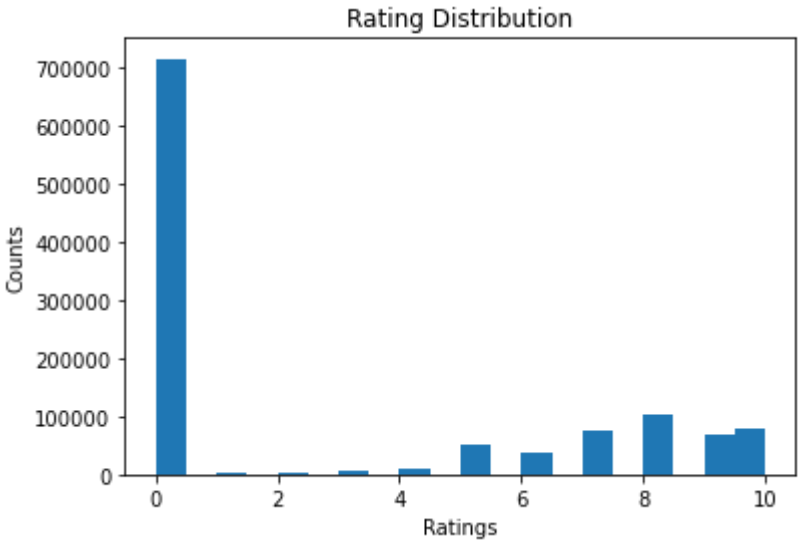
Out[25]:

```
0     716109
8     103736
10     78610
7      76457
9      67541
5      50974
6      36924
4       8904
3       5996
2       2759
1       1770
Name: rating, dtype: int64
```

Rating analysis

In [26]:

```
#Visualize the distribution of ratings
plt.hist(ratings['rating'], bins=20, align='mid')
plt.xlabel('Ratings')
plt.ylabel('Counts')
plt.title('Rating Distribution')
plt.show()
```



In [27]:

```
user_high_ratings = ratings.groupby('user')['rating'].mean().sort_values(ascending=False
print(user_high_ratings)
```

```
user
162951    10.0
21620     10.0
210063    10.0
250956    10.0
21611     10.0
          ...
198846     0.0
198838     0.0
60129      0.0
198835     0.0
2          0.0
Name: rating, Length: 105283, dtype: float64
```

In [28]:

```
user_high_ratings[user_high_ratings==10]
```

Out[28]:

```
user
162951    10.0
21620     10.0
210063    10.0
250956    10.0
21611     10.0
          ...
126212    10.0
153157    10.0
70        10.0
10840     10.0
1539      10.0
Name: rating, Length: 6443, dtype: float64
```

In [29]:

```
user_high_ratings[user_high_ratings==0]
```

Out[29]:

```
user
236012    0.0
252416    0.0
236015    0.0
238018    0.0
20592     0.0
         ...
198846    0.0
198838    0.0
60129     0.0
198835    0.0
2         0.0
Name: rating, Length: 27478, dtype: float64
```

Popular Books

In [30]:

```
ratings.head(2)
```

Out[30]:

	user	ISBN	rating
0	276725	034545104X	0
1	276726	0155061224	5

In [31]:

```
books.head(2)
```

Out[31]:

	ISBN	title	author	year	publisher	
0	0195153448	Classical Mythology	Mark P. O. Morford	2002	Oxford University Press	http://images.amazon.com/images/P/019
1	0002005018	Clara Callan	Richard Bruce Wright	2001	HarperFlamingo Canada	http://images.amazon.com/images/P/000

In [32]:

```
ratings_with_name = ratings.merge(books, on="ISBN")
```

In [33]:

```
num_ratings = ratings_with_name.groupby('title')["rating"].count().reset_index()
num_ratings.columns=['title', 'num_ratings']
num_ratings.head(3)
```

Out[33]:

	title	num_ratings
0	A Light in the Storm: The Civil War Diary of ...	4
1	Always Have Popsicles	1
2	Apple Magic (The Collector's series)	1

In [34]:

```
avg_ratings= ratings_with_name.groupby('title').mean()['rating'].reset_index()
avg_ratings.columns=['title', 'avg_ratings']
avg_ratings.head(3)
```

Out[34]:

	title	avg_ratings
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

In [35]:

```
popular_books = num_ratings.merge(avg_ratings, on="title")
popular_books.sort_values('num_ratings', ascending=False)
```

Out[35]:

	title	num_ratings	avg_ratings
234951	Wild Animus	2502	1.019584
196326	The Lovely Bones: A Novel	1295	4.468726
183573	The Da Vinci Code	898	4.642539
5303	A Painted House	838	3.231504
199237	The Nanny Diaries: A Novel	828	3.530193
...	...	...	...
147559	Real Love: The Truth About Finding Uncondition...	1	0.000000
147558	Real Love: The Drawings for Sean	1	10.000000
147557	Real Love or Fake (Camfield Novel of Love, No 78)	1	5.000000
63664	Fabulous Food for Family and Friends: Healthy ...	1	0.000000
168799	Suburban backlash: The battle for the world's ...	1	0.000000

241071 rows × 3 columns

In [36]:

```
popular_books=popular_books[popular_books["num_ratings"]>=100].sort_values('avg_ratings'
```

In [37]:

```
popular_books=popular_books.merge(books, on='title', how='left').drop_duplicates("title"
popular_books.columns
```

Out[37]:

```
Index(['title', 'num_ratings', 'avg_ratings', 'ISBN', 'author', 'year',
       'publisher', 'image'],
      dtype='object')
```

In [38]:

```
popular_books['avg_ratings']=popular_books['avg_ratings'].apply(lambda x: np.round(x,2))
```

In [39]:

```
popular_books=popular_books.drop(index=[40,166])
```

In [40]:

```
popular_books=popular_books.head(50)
```

In [41]:

```
popular_books["image"][43]
```

Out[41]:

```
'http://images.amazon.com/images/P/059035342X.01.MZZZZZZZ.jpg'
```

Approach

In [42]:

```
books.shape
```

Out[42]:

```
(271360, 6)
```

In [43]:

```
users.shape
```

Out[43]:

```
(278858, 3)
```

In [44]:

```
ratings.shape
```

Out[44]:

```
(1149780, 3)
```

In [45]:

```
# filtering users who gave 100 or more ratings

ratings_100=ratings['user'].value_counts()>=100
```

In [46]:

```
ratings_100[ratings_100]
```

Out[46]:

```
11676     True
198711    True
153662    True
98391     True
35859     True
          ...
65663     True
115692    True
160406    True
160414    True
49212     True
Name: user, Length: 1847, dtype: bool
```

In [47]:

```
# finding user IDs for users who gave 100 or more ratings

index_100 = ratings_100[ratings_100].index
index_100
```

Out[47]:

```
Int64Index([ 11676, 198711, 153662,  98391,  35859, 212898, 278418,  7635
2,
            110973, 235105,
            ...
             64015, 160407, 160410, 160416, 160415,  65663, 115692, 16040
6,
            160414,  49212],
           dtype='int64', length=1847)
```

In [48]:

```
# extracting ratings given by active users

ratings=ratings[ratings['user'].isin(index_100)]
ratings.head(3)
```

Out[48]:

	user	ISBN	rating
412	276925	0006511929	0
413	276925	002542730X	10
414	276925	0060520507	0

Merge Operations

In [49]:

```
ratings_merged=ratings.merge(books, on='ISBN')
```

In [50]:

```
ratings_merged.head(3)
```

Out[50]:

	user	ISBN	rating	title	author	year	publisher	
0	276925	002542730X	10	Politically Correct Bedtime Stories: Modern Ta...	James Finn Garner	1994	John Wiley & Sons Inc	http://images.amazon.com/ima
1	277427	002542730X	10	Politically Correct Bedtime Stories: Modern Ta...	James Finn Garner	1994	John Wiley & Sons Inc	http://images.amazon.com/ima
2	3363	002542730X	0	Politically Correct Bedtime Stories: Modern Ta...	James Finn Garner	1994	John Wiley & Sons Inc	http://images.amazon.com/ima

In [51]:

```
books_rating_qty= ratings_merged.groupby('title')['rating'].count().reset_index()
books_rating_qty.head(3)
```

Out[51]:

	title	rating
0	A Light in the Storm: The Civil War Diary of ...	3
1	Always Have Popsicles	1
2	Apple Magic (The Collector's series)	1

In [52]:

```
books_rating_qty.columns=['title', 'num_of_ratings']
```

In [53]:

```
books_rating_qty.head(3)
```

Out[53]:

	title	num_of_ratings
0	A Light in the Storm: The Civil War Diary of ...	3
1	Always Have Popsicles	1
2	Apple Magic (The Collector's series)	1

In [54]:

```
rating_with_books= ratings_merged.merge(books_rating_qty, on='title')
```

In [55]:

```
final_books= rating_with_books[rating_with_books['num_of_ratings']>=50]
```

In [56]:

```
final_books.shape
```

Out[56]:

```
(100811, 9)
```

Here, we want to avoid any duplicates for same user giving multiple ratings to the same book.

In [57]:

```
final_books=final_books.drop_duplicates(subset=['user', 'title'])
```

In [58]:

```
final_books.shape
```

Out[58]:

```
(98339, 9)
```

Matrix Factorisation

In [59]:

```
book_pivot = final_books.pivot_table(columns='user', index='title', values='rating')
```

In [60]:

```
book_pivot.fillna(0, inplace=True)
```

Model Creation

In [61]:

```
from sklearn.metrics.pairwise import cosine_similarity
```

In [62]:

```
similarity_score= cosine_similarity(book_pivot)
```

In [63]:

```
def recommend_books(book_name):
    '''function returns the name of the recommended books'''
    book_index= np.where(book_pivot.index==book_name)[0][0]
    suggested_items= sorted(list(enumerate(similarity_score[book_index])), key=lambda x:

    data = []
    for i in suggested_items:
        item =[]
        temp = books[books["title"]==book_pivot.index[i[0]]]
        item.extend(temp.drop_duplicates("title")['title'].values)
        item.extend(temp.drop_duplicates("title")['author'].values)
        item.extend(temp.drop_duplicates("title")['image'].values)

        data.append(item)

    return data
```

In [64]:

```
recommend_books('The Fellowship of the Ring (The Lord of the Rings, Part 1)')
```

Out[64]:

```
[['The Two Towers (The Lord of the Rings, Part 2)',
  'J.R.R. TOLKIEN',
  'http://images.amazon.com/images/P/0345339711.01.MZZZZZZZ.jpg'],
 ['The Return of the King (The Lord of the Rings, Part 3)',
  'J.R.R. TOLKIEN',
  'http://images.amazon.com/images/P/0345339738.01.MZZZZZZZ.jpg'],
 ['Harry Potter and the Prisoner of Azkaban (Book 3)',
  'J. K. Rowling',
  'http://images.amazon.com/images/P/0439136350.01.MZZZZZZZ.jpg'],
 ['The Hobbit : The Enchanting Prelude to The Lord of the Rings',
  'J.R.R. TOLKIEN',
  'http://images.amazon.com/images/P/0345339681.01.MZZZZZZZ.jpg'],
 ['Harry Potter and the Goblet of Fire (Book 4)',
  'J. K. Rowling',
  'http://images.amazon.com/images/P/0439139597.01.MZZZZZZZ.jpg']]
```

Deployment

In [65]:

```
import pickle
```

In [66]:

```
pickle.dump(popular_books, open("popular_books.pkl", "wb"))
```

In [67]:

```
pickle.dump(book_pivot, open("book_pivot.pkl", "wb"))
pickle.dump(books, open("books.pkl", "wb"))
pickle.dump(similarity_score, open("similarity_score.pkl", "wb"))
```