# GOAL: TO create a Recommender System using Item based Collabrative Filtering and Matrix Factorization on the given data.

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
```

In [3]:
```
df = pd.read_csv('/content/drive/MyDrive/BX-Book-Ratings.csv', sep= ';', encoding='I
```

In [4]:
```
df.head()
```

Out[4]:

| | 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 |

# EDA AND PREPROCESSING:

In [5]:
```
df.shape
```

Out[5]:
```
(1149780, 3)
```

In [6]:
```
df.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 [10]:
```
df.isnull().sum()
```

Out[10]:
```
User-ID        0
ISBN           0
Book-Rating    0
dtype: int64
```

In [11]:
```
df.isin(['?']).sum(axis=0)
```

Out[11]:
```
User-ID        0
ISBN           0
```

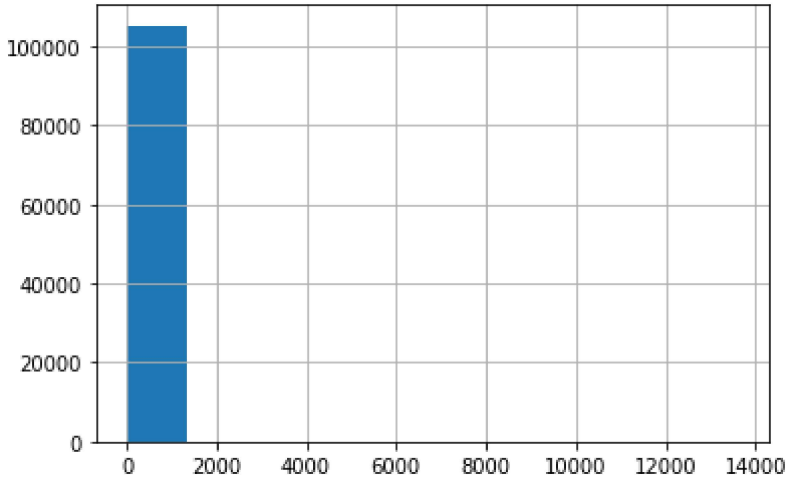
```
Book-Rating    0
dtype: int64
```

In [12]:
```
df['User-ID'].value_counts()
```

Out[12]:
```
11676     13602
198711     7550
153662     6109
98391      5891
35859      5850
          ...
158698        1
17920         1
277135        1
275086        1
187812        1
Name: User-ID, Length: 105283, dtype: int64
```

In [13]:
```
df['User-ID'].value_counts().hist()
```

Out[13]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f50be623550>
```




Statistically we cannot consider those users who have only rated one or two books hence, we are creating a dataset of users who have atleast rated 200 books.

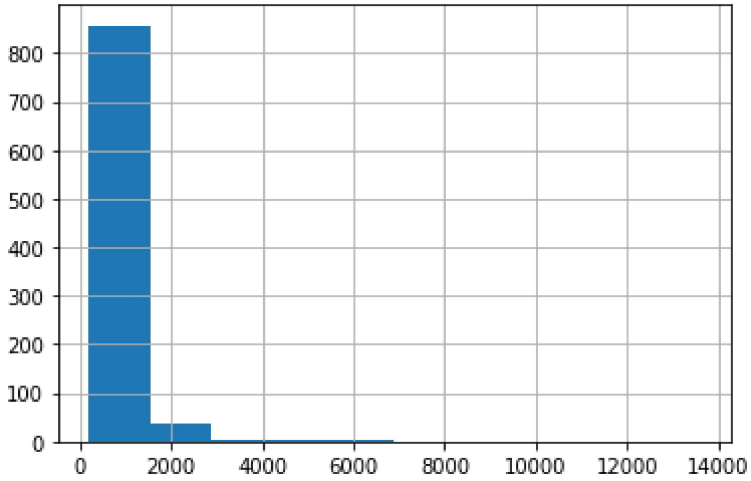
In [14]:
```
df =df[df['User-ID'].isin(df['User-ID'].value_counts()[df['User-ID'].value_counts()
```

In [15]:
```
df['User-ID'].value_counts()
```

Out[15]:
```
11676     13602
198711     7550
153662     6109
98391      5891
35859      5850
          ...
36554       200
83671       200
99955       200
26883       200
225595      200
Name: User-ID, Length: 905, dtype: int64
```

In [16]:
```
df['User-ID'].value_counts().hist()
```

Out[16]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f50c158a990>
```

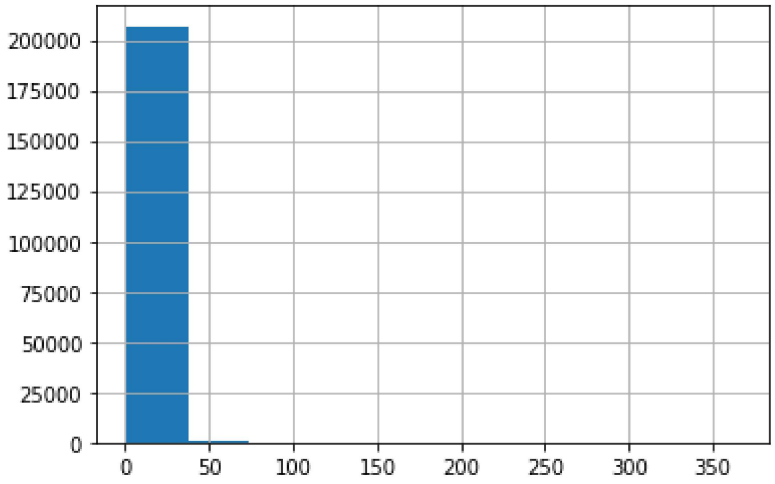



In [17]:
```
df['ISBN'].value_counts()
```

Out[17]:
```
0971880107    365
0316666343    272
0060928336    221
0440214041    218
0385504209    217
             ...
8437610680      1
0836261275      1
0374291977      1
0553110659      1
0698118413      1
Name: ISBN, Length: 207699, dtype: int64
```

In [18]:
```
df['ISBN'].value_counts().hist()
```

Out[18]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f50be4b45d0>
```




We can not consider the books which are rated only once or twice hence, we are considering those books which are rated atleast 50 times.

In [20]:
```
df =df[df['ISBN'].isin(df['ISBN'].value_counts()[df['ISBN'].value_counts() >= 50].in
```

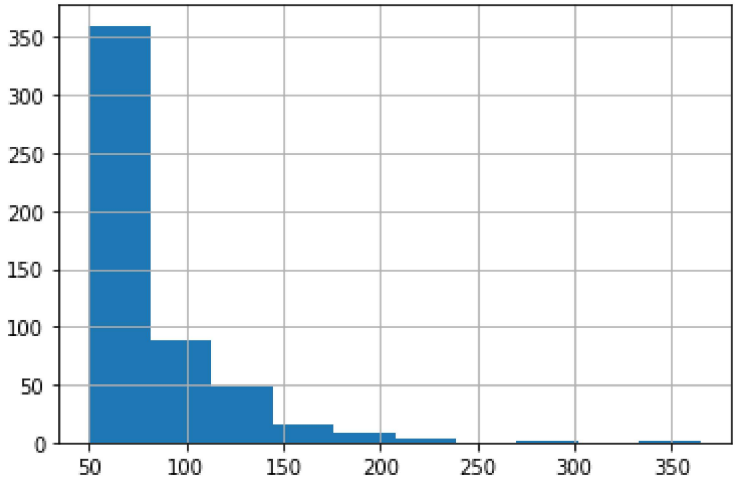
In [21]:
```
df['ISBN'].value_counts()
```

Out[21]:
```
0971880107    365
0316666343    272
0060928336    221
0440214041    218
0385504209    217
             ...
```

```
0345384350     50
0060959037     50
0812550706     50
0553250531     50
055356773X     50
Name: ISBN, Length: 526, dtype: int64
```

In [22]:
```
df['ISBN'].value_counts().hist()
```

Out[22]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f50bd54fe10>
```

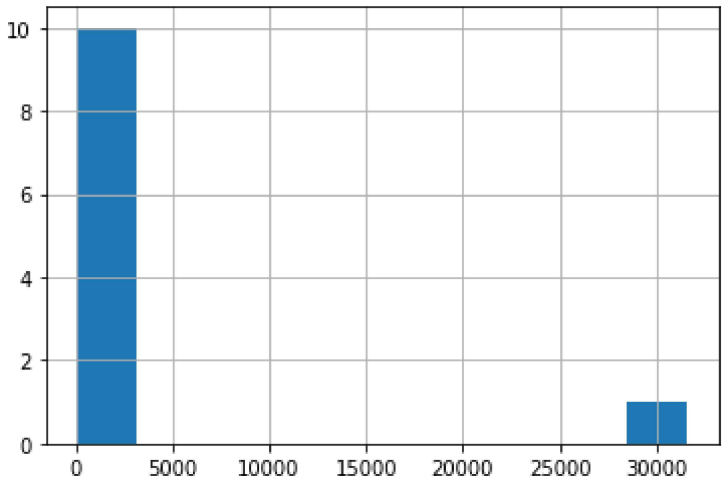



In [23]:
```
df['Book-Rating'].value_counts()
```

Out[23]:
```
0     31631
8      2600
10     2286
9      2129
7      1613
5       861
6       678
4       141
3        76
1        40
2        38
Name: Book-Rating, dtype: int64
```

In [28]:
```
df['Book-Rating'].value_counts().hist()
```

Out[28]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f50bd325f50>
```




We observe that maximum of our rating data lies at 0 this can result in a biased model hence , to avoid it we are creating a new modified ratings colums.

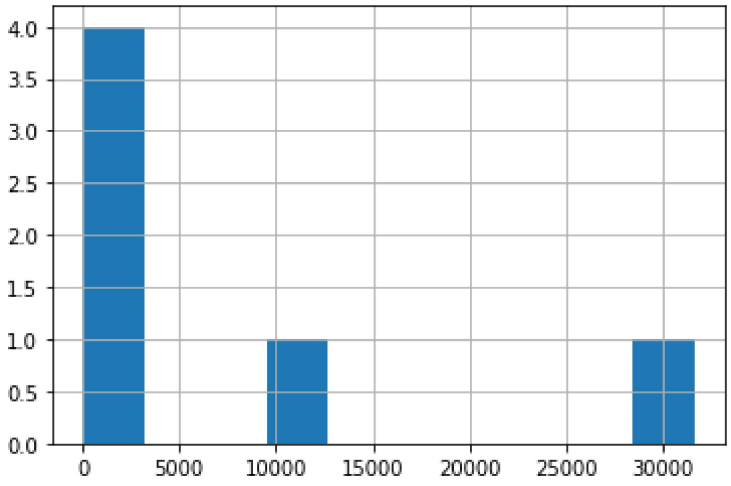
In [25]:
```
df.loc[(df["Book-Rating"] == 0, "Ratings")]=0
df.loc[(df["Book-Rating"] == 1, "Ratings")]=1
df.loc[(df["Book-Rating"] == 2, "Ratings")]=2
df.loc[(df["Book-Rating"] == 3, "Ratings")]=3
df.loc[(df["Book-Rating"] == 4, "Ratings")]=4
df.loc[(df["Book-Rating"] >= 5, "Ratings")]=5
```

In [26]:
```
df['Ratings'].value_counts()
```

Out[26]:
```
0.0    31631
5.0    10167
4.0      141
3.0       76
1.0       40
2.0       38
Name: Ratings, dtype: int64
```

In [27]:
```
df['Ratings'].value_counts().hist()
```

Out[27]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f50bd3b4090>
```




In [29]:
```
df.head()
```

Out[29]:

| | User-ID | ISBN | Book-Rating | Ratings |
|---|---|---|---|---|
| **1456** | 277427 | 002542730X | 10 | 5.0 |
| **1469** | 277427 | 0060930535 | 0 | 0.0 |
| **1471** | 277427 | 0060934417 | 0 | 0.0 |
| **1474** | 277427 | 0061009059 | 9 | 5.0 |
| **1484** | 277427 | 0140067477 | 0 | 0.0 |

In [30]:
```
df.isnull().sum()
```

Out[30]:
```
User-ID        0
ISBN           0
Book-Rating    0
Ratings        0
dtype: int64
```

In [31]:
```
df.drop(["Book-Rating"],axis=1, inplace=True)
```

In [32]:
```
df.head()
```

Out[32]:

| | User-ID | ISBN | Ratings |
|---|---|---|---|

| | User-ID | ISBN | Ratings |
|---|---|---|---|
| **1456** | 277427 | 002542730X | 5.0 |
| **1469** | 277427 | 0060930535 | 0.0 |
| **1471** | 277427 | 0060934417 | 0.0 |
| **1474** | 277427 | 0061009059 | 5.0 |
| **1484** | 277427 | 0140067477 | 0.0 |

# MATRIX FACTORIZATION:

In [33]:
```
!pip install surprise
from surprise import Dataset, SVD, accuracy, Reader
from surprise.model_selection import train_test_split
```

```
Collecting surprise
  Downloading https://files.pythonhosted.org/packages/61/de/e5cba8682201fcf9c3719a6f
dda95693468ed061945493dea2dd37c5618b/surprise-0.1-py2.py3-none-any.whl
Collecting scikit-surprise
  Downloading https://files.pythonhosted.org/packages/97/37/5d334adaf5ddd65da99fc65f
6507e0e4599d092ba048f4302fe8775619e8/scikit-surprise-1.1.1.tar.gz (11.8MB)
     |████████████████████████████████| 11.8MB 6.4MB/s
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.7/dist-package
s (from scikit-surprise->surprise) (1.0.1)
Requirement already satisfied: numpy>=1.11.2 in /usr/local/lib/python3.7/dist-packag
es (from scikit-surprise->surprise) (1.19.5)
Requirement already satisfied: scipy>=1.0.0 in /usr/local/lib/python3.7/dist-package
s (from scikit-surprise->surprise) (1.4.1)
Requirement already satisfied: six>=1.10.0 in /usr/local/lib/python3.7/dist-packages
(from scikit-surprise->surprise) (1.15.0)
Building wheels for collected packages: scikit-surprise
  Building wheel for scikit-surprise (setup.py) ... done
  Created wheel for scikit-surprise: filename=scikit_surprise-1.1.1-cp37-cp37m-linux
_x86_64.whl size=1617657 sha256=2690591aae8f16a4f93429a6e1917aadfb1839ee16f3d3f5d834
e656064a81ac
  Stored in directory: /root/.cache/pip/wheels/78/9c/3d/41b419c9d2aff5b6e2b4c0fc8d25
c538202834058f9ed110d0
Successfully built scikit-surprise
Installing collected packages: scikit-surprise, surprise
Successfully installed scikit-surprise-1.1.1 surprise-0.1
```

In [34]:
```
df["User-ID"] = "User" + df["User-ID"].astype("str")
```

We are converting the datatype of User-ID colum to object as required in Matrix Factorization.

In [35]:
```
df.head()
```

Out[35]:

| | User-ID | ISBN | Ratings |
|---|---|---|---|
| **1456** | User277427 | 002542730X | 5.0 |
| **1469** | User277427 | 0060930535 | 0.0 |
| **1471** | User277427 | 0060934417 | 0.0 |
| **1474** | User277427 | 0061009059 | 5.0 |
| **1484** | User277427 | 0140067477 | 0.0 |

In [61]:
```
#defining rating scale
reader = Reader(rating_scale=(1,5))
```

In [62]:
```
data = Dataset.load_from_df(df,reader)
```

In [63]:
```
trainset,testset = train_test_split(data,test_size=0.3)
```

In [64]:
```
svd = SVD(n_factors=1000)
```

In [65]:
```
svd.fit(trainset)
```

Out[65]:
```
<surprise.prediction_algorithms.matrix_factorization.SVD at 0x7f50b6339950>
```

In [66]:
```
predictions = svd.test(testset)
```

In [67]:
```
accuracy.rmse(predictions)
```

```
RMSE: 2.0544
```

Out[67]:
```
2.054406307692901
```

In [68]:
```
df.iloc[500]
```

Out[68]:
```
User-ID      User6251
ISBN       0440206154
Ratings             5
Name: 26052, dtype: object
```

In [69]:
```
user = "User6251"
book = "0440206154"
```

In [70]:
```
pred = svd.predict(user,book)
```

In [71]:
```
pred
```

Out[71]:
```
Prediction(uid='User6251', iid='0440206154', r_ui=None, est=4.678215101224084, detai
ls={'was_impossible': False})
```

# Conclusion: We observed that the prediction made by our model 4.6 is fairly close to the actual rating viz 5. We can say that our model is doing a good job in predicting the ratings.

# ITEM BASED COLLABRATIVE FILTERING:

In [72]:
```
ui_matrix = df.pivot_table(index="User-ID",columns="ISBN",values="Ratings")
```

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

Out[73]:

| ISBN | 002542730X | 006016848X | 0060391626 | 0060392452 | 0060502258 | 0060915544 | 00609 |
|---|---|---|---|---|---|---|---|
| **User-ID** | | | | | | | |
| **User100459** | NaN | NaN | NaN | 5.0 | NaN | NaN | |
| **User100644** | NaN | NaN | NaN | NaN | NaN | NaN | |
| **User100846** | NaN | NaN | NaN | NaN | NaN | NaN | |

| ISBN | 002542730X | 006016848X | 0060391626 | 0060392452 | 0060502258 | 0060915544 | 00609 |
|---|---|---|---|---|---|---|---|
| User-ID | | | | | | | |
| User100906 | NaN | NaN | NaN | NaN | NaN | NaN | |
| User101209 | NaN | NaN | NaN | NaN | NaN | NaN | |

5 rows × 526 columns

In [74]:
```
ui_matrix.fillna(0,inplace=True)
```

In [76]:
```
ui_matrix.head()
```

Out[76]:

| ISBN | 002542730X | 006016848X | 0060391626 | 0060392452 | 0060502258 | 0060915544 | 00609 |
|---|---|---|---|---|---|---|---|
| User-ID | | | | | | | |
| User100459 | 0.0 | 0.0 | 0.0 | 5.0 | 0.0 | 0.0 | |
| User100644 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | |
| User100846 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | |
| User100906 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | |
| User101209 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | |

5 rows × 526 columns

In [77]:
```
book_read = ui_matrix["0060928336"]
book_read
```

Out[77]:
```
User-ID
User100459    0.0
User100644    0.0
User100846    0.0
User100906    0.0
User101209    0.0
             ...
User98391     0.0
User9856      0.0
User98741     0.0
User98758     0.0
User99955     0.0
Name: 0060928336, Length: 889, dtype: float64
```

Above are the diffrent users and thr ratings they have given to this book.

In [78]:
```
similarity_scores = ui_matrix.corrwith(book_read)
similarity_scores
```

Out[78]:
```
ISBN
002542730X   -0.007908
006016848X    0.007949
0060391626   -0.027591
0060392452   -0.000088
0060502258    0.095631
                ...
1558744150   -0.040171
1558745157    0.001152
1559029838   -0.030028
1573225789    0.021201
```

```
1573229326    0.005531
Length: 526, dtype: float64
```

In [79]:
```
recommendation = similarity_scores.sort_values(ascending=False).head(10)
```

In [80]:
```
recommendation
```

Out[80]:
```
ISBN
0060928336    1.000000
014029628X    0.245245
0439139600    0.205316
0312195516    0.181140
0449212602    0.167960
0140119906    0.164463
0060959037    0.164463
0316284955    0.164339
0439064872    0.156516
0671041789    0.154121
dtype: float64
```

# Conclusion: Based on Pearson's corelation coefficient, above are the Top Ten Reommended books for book 0060930535.