In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import os
%matplotlib inline
os.chdir('C:\\Users\\utpala mohapatra\\Documents\\python_folder\\python datasets')
```

In [7]:
```
books_df = pd.read_csv('BX-Books.csv')
books_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 271379 entries, 0 to 271378
Data columns (total 5 columns):
 #   Column              Non-Null Count   Dtype 
---  ------              --------------   ----- 
 0   isbn                271379 non-null  object
 1   book_title          271379 non-null  object
 2   book_author         271378 non-null  object
 3   year_of_publication  271378 non-null  object
 4   publisher           271377 non-null  object
dtypes: object(5)
memory usage: 10.4+ MB
```

```
C:\Users\utpala mohapatra\anaconda3\lib\site-packages\IPython\core\interactiveshell.py:3165: DtypeWarning: Columns (3) have mixed types.Specify dtype option on import or set low_memory=False.
  has_raised = await self.run_ast_nodes(code_ast.body, cell_name,
```

In [8]:
```
user_df = pd.read_csv('BX-Users.csv')
user_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 278859 entries, 0 to 278858
Data columns (total 3 columns):
 #   Column    Non-Null Count   Dtype  
---  ------    --------------   -----  
 0   user_id   278859 non-null  object 
 1   Location  278858 non-null  object 
 2   Age       168096 non-null  float64
dtypes: float64(1), object(2)
memory usage: 6.4+ MB
```

```
C:\Users\utpala mohapatra\anaconda3\lib\site-packages\IPython\core\interactiveshell.py:3165: DtypeWarning: Columns (0) have mixed types.Specify dtype option on import or set low_memory=False.
  has_raised = await self.run_ast_nodes(code_ast.body, cell_name,
```

In [42]:
```
ratings_df = pd.read_csv('BX-Book-Ratings.csv')
ratings_df.info()
```

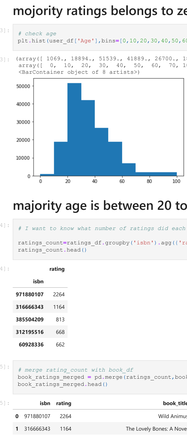
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1048575 entries, 0 to 1048574
Data columns (total 3 columns):
 #   Column       Non-Null Count    Dtype 
---  ------       --------------    ----- 
 0   user_id      1048575 non-null  int64 
 1   isbn         1048575 non-null  object
 2   rating       1048575 non-null  int64 
dtypes: int64(2), object(1)
memory usage: 24.0+ MB
```

In [11]:
```
print(books_df.shape,user_df.shape,ratings_df.shape)
```

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

In [19]:
```
ratings_df.rating.value_counts().plot(kind='bar')
```

Out[19]:
```
<AxesSubplot:>
```



mojority ratings belongs to zero

In [13]:
```
# check age
plt.hist(user_df['Age'],bins=[0,10,20,30,40,50,60,70,100])
```

Out[13]:
```
(array([ 1069.,  18894.,  51539.,  41889.,  26700.,  18811.,   6916.,
         1912.]),
 array([  0,  10,  20,  30,  40,  50,  60,  70, 100]),
 <BarContainer object of 8 artists>)
```

majority age is between 20 to 30

In [34]:
```
# I want to know what number of ratings did each book receive

ratings_count=ratings_df.groupby('isbn').agg({'rating':'count'}).sort_values('rating',ascending=False)
ratings_count.head()
```

Out[34]:

	rating
isbn	
971880107	2264
316666343	1164
385504209	813
312195516	668
60928336	662

In [35]:
```
# merge rating_count with book_df
book_ratings_merged = pd.merge(ratings_count,books_df,on='isbn')
book_ratings_merged.head()
```

Out[35]:

	isbn	rating	book_title	book_author	year_of_publication	publisher
0	971880107	2264	Wild Animus	Rich Shapero	2004	Too Far
1	316666343	1164	The Lovely Bones: A Novel	Alice Sebold	2002	Little, Brown
2	385504209	813	The Da Vinci Code	Dan Brown	2003	Doubleday
3	312195516	668	The Red Tent (Bestselling Backlist)	Anita Diamant	1998	Picador USA
4	60928336	662	Divine Secrets of the Ya-Ya Sisterhood: A Novel	Rebecca Wells	1997	Perennial

In [36]:
```
# add the average of in the rating_count df
ratings_count['Mean']=ratings_df.groupby('isbn').agg({'rating':'mean'})


ratings_count.head()
```

Out[36]:

	rating	Mean
isbn		
971880107	2264	1.032244
316666343	1164	4.457045
385504209	813	4.691267
312195516	668	4.326347
60928336	662	3.462236

In [43]:
```
#select ratings of users who have rated more than 200 ratings
count_users = ratings_df['user_id'].value_counts()
ratings_df = ratings_df[ratings_df['user_id'].isin(count_users[count_users >= 200].index)]

#We will also remove books with less than 100 ratings
count_books = ratings_df['rating'].value_counts()
ratings_df = ratings_df[ratings_df['rating'].isin(count_books[count_books >= 200].index)]

ratings_df.head()
```

Out[43]:

	user_id	isbn	rating
1456	277427	002542730X	10
1457	277427	26217457	0
1458	277427	0030986053X	8
1459	277427	30615321	0
1460	277427	60002050	0

In [67]:
```
ratings_df = ratings_df.drop_duplicates()
ratings_pivot = ratings_df.reset_index().pivot_table('rating',index='user_id',columns='isbn')
ratings_pivot.fillna(0,inplace=True)
ratings_pivot.head()
```

Out[67]:

isbn	9044924401X	'0551528325	0	0.00 612183 7	0.00 614496 2	0 7336 1053 6	0 907 062 008	0.330241664	000104687X	0001047999X	...	THEFLYINGACE	UNGRANDH
user_id													
254	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	
2276	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	
2766	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	
2977	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	
3363	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	

5 rows × 197767 columns

In [68]:
```
ratings_pivot[ratings_pivot.index == 254]
```

Out[68]:

isbn	9044924401X	'0551528325	0	0.00 612183 7	0.00 614496 2	0 7336 1053 6	0 907 062 008	0.330241664	000104687X	0001047999X	...	THEFLYINGACE	UNGRANDH
user_id													
254	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	

1 rows × 197767 columns

User - User BAsed collaborative filtering

In [71]:
```
from sklearn.metrics.pairwise import cosine_similarity
import operator

def similar_user(user_id,matrix,k=3):
    # create matrix for current user
    user = matrix[matrix.index == user_id]

    # create matrix for other users
    other_users = matrix[matrix.index != user_id]

    #calculate cosine similarity between each user with other user
    similarities =cosine_similarity(user,other_users)[0].tolist()

    #get the indices of other users
    indices = other_users.index.tolist()

    # create a dict of key as index and similarity as value
    index_similarity = dict(zip(indices,similarities))
    # print(index_similarity)

    # sort by similarity
    index_similarity_sorted = sorted(index_similarity.items(),key = operator.itemgetter(1))
    index_similarity_sorted.reverse()

    # grab k users of the top
    top_user_similarity = index_similarity_sorted[:k]
    users = [u[0] for u in top_user_similarity]

    return users
```

In [72]:
```
similar_user_indices = similar_user(277478,ratings_pivot,10)
print(similar_user_indices)
```

```
[141819, 12538, 42914, 102647, 80538, 239594, 76352, 81402, 44595, 203799]
```

In [74]:
```
def recommend_item(user_index,similar_user_indices,matrix,item = 3):
    # load vectors for similar users
    similar_users = matrix[matrix.index.isin(similar_user_indices)]

    # calc avg ratings across the 3 similar users
    similar_users = similar_users.mean(axis=0)

    # convert to dataframe so its easy to sort and filter
    similar_users_df = pd.DataFrame(similar_users,columns=['mean'])

    # load vector for the current user
    user_df = matrix[matrix.index == user_index]

    # transpose it so its easier to filter
    user_df_transposed = user_df.transpose()

    # rename the column as 'rating'
    user_df_transposed.rename(columns={user_index:'rating'},inplace = True)

    # remove any rows without a 0 value. Books not read yet
    user_df_transposed = user_df_transposed[user_df_transposed['rating']==0]

    # generate a list of Books the user has not read
    books_unseen = user_df_transposed.index.tolist()


    # filter avg ratings of similar users for only Books the current user has not read
    similar_users_filtered = similar_users_df[similar_users_df.index.isin(books_unseen)]

    # order the dataframe
    similar_users_ordered = similar_users_filtered.sort_values(by=['mean'],ascending=False )

    # grab the top n books
    top_n_books = similar_users_ordered.head(item)
    top_n_books_indices = top_n_books.index.tolist()

    # lookup these books in the other dataframe to find names
    book_info = book_ratings_merged[book_ratings_merged['isbn'].isin(top_n_books_indices)]

    return book_info

recommend_item(277478, similar_user_indices, ratings_pivot)
```

Out[74]:

	isbn	rating	book_title	book_author	year_of_publication	publisher
488	055321313X	117	Anne of Green Gables (Anne of Green Gables Nov...	L.M. MONTGOMERY	1982	Bantam Classics
626	452280621	99	Beloved	Toni Morrison	1998	Plume Books
32525	345383850	5	Garfield Fat Cat: Garfield at Large/Garfield G...	Jim Davis	1993	Ballantine Books

In [ ]:
```

```

In [ ]:
```

```

In [ ]:
```

```

In [ ]:
```

```

In [ ]:
```

```

In [ ]:
```

```

In [ ]:
```

```

In [ ]:
```

```