BOOK RECOMMENDATION ENGINE

Mikaela Marcos
Applicant

Flow of Topics

- Problem Statement
- Data Summary
- Descriptive Analysis of data
- Data Preparation and Cleaning
- Recommended Books
- Conclusion
- Challenges encountered

Introduction

- Progress… development
- Entertainment
- More and highly personalized
- Very relative

Introduction

- Recommending algorithms
- Inclusive environment
- Consumer's perspective

Introduction

For this homework:

- An engine would be needed to determine which books can be suggested to readers of specific books
- To consider the reader's interest and the popularity of books in the given data set

Data Summary

The data sets that will be used is entitled: Books.csv, Ratings.csv, Users.csv

User:

- User-ID
- Location (city, state, country)
- Age
- Shape of data set (278858, 3)

Data Summary

Books Data set

- ISBN
- Book-Title
- Book-Author
- Year-Of-Publication
- Publisher
- Image-URL-S
- Image-URL-M
- Image-URL-L
- Shape of data set (271360, 8)

Data Summary

Ratings data set

- User-ID
- ISBN
- Book-Rating
- Shape of data set (1149780, 3)

Data Preparation and Cleaning
Disclaimer

It has to be noted that the presenter had limited to no background in using programming softwares and with the limited amount of time to create this engine, she had conducted related readings and adoptation of previously used source code.

Data Preparation and Cleaning
Disclaimer

Still it has to be taken into account that the outsourced code still didn't function well and encountered errors in running it on the program as it is, hence there are some alteration conducted by the presenter.

Data Preparation and Cleaning

Jupyter Notebook

- To conduct all process on making this engine we will import and process it in Jupyter Notebook
- This support different languages including Python which we will use in this set-up
- For this, we will utilize Ashima Garg, Ananya Tyagi, and Arun Abhishek Chowhan's source code.

Data Preparation and Cleaning

Jupyter Notebook

Libraries used that needs to be installed

- ipython-notebook
- sklearn
- seaborn
- matplotlib
- numpy
- pandas

Data Preparation and Cleaning
Jupyter Notebook

1. Importation of libraries

```
import re
import pickle
import operator
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from collections import Counter
from scipy.sparse import csr_matrix
from pandas.api.types import is_numeric_dtype
from sklearn.neighbors import NearestNeighbors
from sklearn.feature_extraction import DictVectorizer
from sklearn.metrics.pairwise import cosine_similarity
from sklearn.feature_extraction.text import TfidfVectorizer

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

Data Preparation and Cleaning

Books data set

1. Importation of data sets

```
books = pd.read_csv(r"Datasets/Books.csv", delimiter=';', on_bad_lines= 'warn'
users = pd.read_csv(r"Datasets/Users.csv", delimiter=';', on_bad_lines= 'warn'
ratings = pd.read_csv(r"Datasets/Book-Ratings.csv", delimiter=';', on_bad_line

print("Books Data:    ", books.shape)
print("Users Data:    ", users.shape)
print("Books-Ratings: ", ratings.shape)
```

```
Skipping line 6452: expected 8 fields, saw 9
Skipping line 43667: expected 8 fields, saw 10
Skipping line 51751: expected 8 fields, saw 9

Skipping line 92038: expected 8 fields, saw 9
Skipping line 104319: expected 8 fields, saw 9
Skipping line 121768: expected 8 fields, saw 9

Skipping line 144058: expected 8 fields, saw 9
Skipping line 150789: expected 8 fields, saw 9
Skipping line 157128: expected 8 fields, saw 9
Skipping line 180189: expected 8 fields, saw 9
Skipping line 185738: expected 8 fields, saw 9

Skipping line 209388: expected 8 fields, saw 9
Skipping line 220626: expected 8 fields, saw 9
Skipping line 227933: expected 8 fields, saw 11
Skipping line 228957: expected 8 fields, saw 10
Skipping line 245933: expected 8 fields, saw 9
Skipping line 251296: expected 8 fields, saw 9
Skipping line 259941: expected 8 fields, saw 9
```

Data Preparation and Cleaning
Books data set

Comparison to the outsourced code

```
error_bad_lines=False, encoding='ISO-8859-1', warn_bad_lines=False
error_bad_lines=False, encoding='ISO-8859-1', warn_bad_lines=False
ter=';', error_bad_lines=False, encoding='ISO-8859-1', warn_bad_li
```

Outsourced : Used error_bad_lines and warn_bad_lines in their syntax

```
oks.csv", delimiter=';', on_bad_lines= 'warn', encoding='ISO-8859-1')
ers.csv", delimiter=';', on_bad_lines= 'warn', encoding='ISO-8859-1')
Book-Ratings.csv", delimiter=';', on_bad_lines= 'warn', encoding='ISO-8859-1')

ape)
ape)
shape)
```

Presenter: Used on_bad_lines instead

Data Preparation and Cleaning

Books data set

2. Dropping the image columns

```
## Dropping URL Columns
books.drop(['Image-URL-S', 'Image-URL-M', 'Image-URL-L'], axis=1, inplace=True)
books.head()
```

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

Data Preparation and Cleaning
Books data set

3. Checking for null values for each column

```
## Check for the null values
books.isnull().sum()
```

```
ISBN                    0
Book-Title              0
Book-Author             2
Year-Of-Publication     0
Publisher               2
dtype: int64
```

Data Preparation and Cleaning

Books data set

4. Checking the specific null data.

```
books.loc[books['Book-Author'].isnull(),:]
```

	ISBN	Book-Title	Book-Author	Year-Of-Publication	Publisher
118033	0751352497	A+ Quiz Masters:01 Earth	NaN	1999	Dorling Kindersley
187689	9627982032	The Credit Suisse Guide to Managing Your Perso...	NaN	1995	Edinburgh Financial Publishing

```
books.loc[books['Publisher'].isnull(),:]
```

	ISBN	Book-Title	Book-Author	Year-Of-Publication	Publisher
128890	193169656X	Tyrant Moon	Elaine Corvidae	2002	NaN
129037	1931696993	Finders Keepers	Linnea Sinclair	2001	NaN

Data Preparation and Cleaning
Books data set

5. Changing the null into 'other' instead

```
## Changing those with null values with other instead
books.at[118033 ,'Book-Author'] = 'Other'
books.at[187689 ,'Book-Author'] = 'Other'

books.at[128890 ,'Publisher'] = 'Other'
books.at[129037 ,'Publisher'] = 'Other'
```

Data Preparation and Cleaning

Books data set

5. Checking for a different content within Year-Of Publication.

```
## Checking for a different content within Year-Of Publication
books['Year-Of-Publication'].unique()
```

```
array([2002, 2001, 1991, 1999, 2000, 1993, 1996, 1988, 2004, 1998, 1994,
       2003, 1997, 1983, 1979, 1995, 1982, 1985, 1992, 1986, 1978, 1980,
       1952, 1987, 1990, 1981, 1989, 1984, 0, 1968, 1961, 1958, 1974,
       1976, 1971, 1977, 1975, 1965, 1941, 1970, 1962, 1973, 1972, 1960,
       1966, 1920, 1956, 1959, 1953, 1951, 1942, 1963, 1964, 1969, 1954,
       1950, 1967, 2005, 1957, 1940, 1937, 1955, 1946, 1936, 1930, 2011,
       1925, 1948, 1943, 1947, 1945, 1923, 2020, 1939, 1926, 1938, 2030,
       1911, 1904, 1949, 1932, 1928, 1929, 1927, 1931, 1914, 2050, 1934,
       1910, 1933, 1902, 1924, 1921, 1900, 2038, 2026, 1944, 1917, 1901,
       2010, 1908, 1906, 1935, 1806, 2021, '2000', '1995', '1999', '2004',
       '2003', '1990', '1994', '1986', '1989', '2002', '1981', '1993',
       '1983', '1982', '1976', '1991', '1977', '1998', '1992', '1996',
       '0', '1997', '2001', '1974', '1968', '1987', '1984', '1988',
       '1963', '1956', '1970', '1985', '1978', '1973', '1980', '1979',
       '1975', '1969', '1961', '1965', '1939', '1958', '1950', '1953',
       '1966', '1971', '1959', '1972', '1955', '1957', '1945', '1960',
       '1967', '1932', '1924', '1964', '2012', '1911', '1927', '1948',
       '1962', '2006', '1952', '1940', '1951', '1931', '1954', '2005',
       '1930', '1941', '1944', 'DK Publishing Inc', '1943', '1938',
       '1900', '1942', '1923', '1920', '1933', 'Gallimard', '1909',
       '1946', '2008', '1378', '2030', '1936', '1947', '2011', '2020',
       '1919', '1949', '1922', '1897', '2024', '1376', '1926', '2037'],
      dtype=object)
```

Data Preparation and Cleaning

Books data set

6. Checking the details of the books which had their publisher in the Year-Of-Publication

```
books.loc[books['Year-Of-Publication'] == 'DK Publishing Inc',:]
```

	ISBN	Book-Title	Book-Author	Year-Of-Publication	
209538	078946697X	DK Readers: Creating the X-Men, How It All Began (Level 4: Proficient Readers)\";Michael Teitelbaum"	2000	DK Publishing Inc	http://images.amazon.com/image
221678	0789466953	DK Readers: Creating the X-Men, How Comic Books Come to Life (Level 4: Proficient Readers)\";James Buckley"	2000	DK Publishing Inc	http://images.amazon.com/image

```
books.loc[books['Year-Of-Publication'] == 'Gallimard',:]
```

	ISBN	Book-Title	Book-Author	Year-Of-Publication	
220731	2070426769	Peuple du ciel, suivi de 'Les Bergers\";Jean-	2003	Gallimard	http://images.amazon.com/images/P/

Data Preparation and Cleaning

Books data set

7. Correcting the details and assigning them on their respective column

```
## Correcting the details and assigning them on their respective column
books.at[209538 ,'Publisher'] = 'DK Publishing Inc'
books.at[209538 ,'Year-Of-Publication'] = 2000
books.at[209538 ,'Book-Title'] = 'DK Readers: Creating the X-Men, How It All Began
books.at[209538 ,'Book-Author'] = 'Michael Teitelbaum'

books.at[221678 ,'Publisher'] = 'DK Publishing Inc'
books.at[221678 ,'Year-Of-Publication'] = 2000
books.at[209538 ,'Book-Title'] = 'DK Readers: Creating the X-Men, How Comic Books (
books.at[209538 ,'Book-Author'] = 'James Buckley'

books.at[220731 ,'Publisher'] = 'Gallimard'
books.at[220731 ,'Year-Of-Publication'] = '2003'
books.at[209538 ,'Book-Title'] = 'Peuple du ciel - Suivi de Les bergers '
books.at[209538 ,'Book-Author'] = 'Jean-Marie Gustave Le ClÃ?Â©zio'
```

Data Preparation and Cleaning

Books data set

7. Turning the Year-Of- Publication in a number format and checking the content.

```
## Years-of -Publication will be turn into a number format
books['Year-Of-Publication'] = books['Year-Of-Publication'].astype(int)
```

```
##Checking of years if all are in numbers
print(sorted(list(books['Year-Of-Publication'].unique())))
```

```
[0, 1376, 1378, 1806, 1897, 1900, 1901, 1902, 1904, 1906, 1908, 1909, 1910, 1911, 19
14, 1917, 1919, 1920, 1921, 1922, 1923, 1924, 1925, 1926, 1927, 1928, 1929, 1930, 19
31, 1932, 1933, 1934, 1935, 1936, 1937, 1938, 1939, 1940, 1941, 1942, 1943, 1944, 19
45, 1946, 1947, 1948, 1949, 1950, 1951, 1952, 1953, 1954, 1955, 1956, 1957, 1958, 19
59, 1960, 1961, 1962, 1963, 1964, 1965, 1966, 1967, 1968, 1969, 1970, 1971, 1972, 19
73, 1974, 1975, 1976, 1977, 1978, 1979, 1980, 1981, 1982, 1983, 1984, 1985, 1986, 19
87, 1988, 1989, 1990, 1991, 1992, 1993, 1994, 1995, 1996, 1997, 1998, 1999, 2000, 20
01, 2002, 2003, 2004, 2005, 2006, 2008, 2010, 2011, 2012, 2020, 2021, 2024, 2026, 20
30, 2037, 2038, 2050]
```

Data Preparation and Cleaning
Books data set

8. Replacing the invalid years (those with 0 and those greater than 2023 with the year with the max count

```
##Replacing the Invalid years with the max year
count = Counter(books['Year-Of-Publication'])
[k for k, v in count.items() if v == max(count.values())]
```

```
[2002]
```

```
## Changing invalid years such as zero and those > 2023 with the max year
books.loc[books['Year-Of-Publication'] > 2023, 'Year-Of-Publication'] = 2002
books.loc[books['Year-Of-Publication'] == 0, 'Year-Of-Publication'] = 2002
```

Data Preparation and Cleaning
Books data set

9. Capitalized the aplhabets in ISBN and dropping duplicated information

```
## The alphabets in the ISBN should be capitalized
books['ISBN'] = books['ISBN'].str.upper()
```

```
##Duplicate entries or rows is to be dropped
books.drop_duplicates(keep='last', inplace=True)
books.reset_index(drop = True, inplace = True)
```

Data Preparation and Cleaning

Users data set

1. Checking for the Null Values.

```
##Null values for each column should be checked
print(users.isna().sum())
```

```
User-ID      0
Location     0
Age          110762
dtype: int64
```

Data Preparation and Cleaning

Users data set

2. Checking what age is present in the data set

```
]:
##Check what age is present on the age column
print(sorted(list(users['Age'].unique())))
```

```
[nan, 0.0, 1.0, 2.0, 3.0, 4.0, 5.0, 6.0, 7.0, 8.0, 9.0, 10.0, 11.0, 12.0, 13.0, 14.
0, 15.0, 16.0, 17.0, 18.0, 19.0, 20.0, 21.0, 22.0, 23.0, 24.0, 25.0, 26.0, 27.0, 28.
0, 29.0, 30.0, 31.0, 32.0, 33.0, 34.0, 35.0, 36.0, 37.0, 38.0, 39.0, 40.0, 41.0, 42.
0, 43.0, 44.0, 45.0, 46.0, 47.0, 48.0, 49.0, 50.0, 51.0, 52.0, 53.0, 54.0, 55.0, 56.
0, 57.0, 58.0, 59.0, 60.0, 61.0, 62.0, 63.0, 64.0, 65.0, 66.0, 67.0, 68.0, 69.0, 70.
0, 71.0, 72.0, 73.0, 74.0, 75.0, 76.0, 77.0, 78.0, 79.0, 80.0, 81.0, 82.0, 83.0, 84.
0, 85.0, 86.0, 87.0, 88.0, 89.0, 90.0, 91.0, 92.0, 93.0, 94.0, 95.0, 96.0, 97.0, 98.
0, 99.0, 100.0, 101.0, 102.0, 103.0, 104.0, 105.0, 106.0, 107.0, 108.0, 109.0, 110.
0, 111.0, 113.0, 114.0, 115.0, 116.0, 118.0, 119.0, 123.0, 124.0, 127.0, 128.0, 132.
0, 133.0, 136.0, 137.0, 138.0, 140.0, 141.0, 143.0, 146.0, 147.0, 148.0, 151.0, 152.
0, 156.0, 157.0, 159.0, 162.0, 168.0, 172.0, 175.0, 183.0, 186.0, 189.0, 199.0, 200.
0, 201.0, 204.0, 207.0, 208.0, 209.0, 210.0, 212.0, 219.0, 220.0, 223.0, 226.0, 228.
0, 229.0, 230.0, 231.0, 237.0, 239.0, 244.0]
```

Data Preparation and Cleaning

Users data set

3. Setting the valid range from 10-80 and getting the mean age of all users.

```
required = users[users['Age'] <= 80]
required = required[required['Age'] >= 10]
```

```
mean = round(required['Age'].mean())
mean
```

```
35
```

Data Preparation and Cleaning

Users data set

4. Setting the invalid ages into the mean of the valid ages within the data set and changing the data type as well.

```
users.loc[users['Age'] > 80, 'Age'] = mean     #those with age grater than 80
users.loc[users['Age'] < 10, 'Age'] = mean     #those with age less than 10 years a
users['Age'] = users['Age'].fillna(mean)       #filling null values with mean
users['Age'] = users['Age'].astype(int)        #changing Datatype to int
```

Data Preparation and Cleaning

Users data set

5. Splitting the location column into 3 columns of City, State, and Country.

```
list_ = users.Location.str.split(', ')

city = []
state = []
country = []
count_no_state = 0
count_no_country = 0

for i in range(0,len(list_)):
    if list_[i][0] == ' ' or list_[i][0] == '' or list_[i][0]=='n/a' or list_[i][0] == ',':  #removing invalid entries too
        city.append('other')
    else:
        city.append(list_[i][0].lower())

    if(len(list_[i])<2):
        state.append('other')
        country.append('other')
        count_no_state += 1
        count_no_country += 1
    else:
        if list_[i][1] == ' ' or list_[i][1] == '' or list_[i][1]=='n/a' or list_[i][1] == ',':   #removing invalid entrie
            state.append('other')
            count_no_state += 1
        else:
            state.append(list_[i][1].lower())

        if(len(list_[i])<3):
            country.append('other')
            count_no_country += 1
        else:
            if list_[i][2] == ''or list_[i][1] == ',' or list_[i][2] == ' ' or list_[i][2] == 'n/a':
                country.append('other')
                count_no_country += 1
            else:
                country.append(list_[i][2].lower())

users = users.drop('Location',axis=1)

temp = []
```

Data Preparation and Cleaning

Users data set

6. Checking how many in the countries and state column doesn't have a value

```
print(count_no_country)   #to show the number of countries didnt have any values
print(count_no_state)     #to show the states which didnt have any values
```

Data Preparation and Cleaning
Users data set

7. Dropping duplicate information

```
## Duplicate entries/ rows should also be dropped
users.drop_duplicates(keep='last', inplace=True)
users.reset_index(drop=True, inplace=True)
```

Data Preparation and Cleaning
Book-Ratings

1. Checking for the Null Values

```
## Null values should be checked for each column
ratings.isnull().sum()
```

```
User-ID        0
ISBN           0
Book-Rating    0
dtype: int64
```

Data Preparation and Cleaning
Book-Ratings

2. Checking i fthe data type for ratings and User-ID are in numbers

```
## the data type of all ratings should be numbers as well
print(is_numeric_dtype(ratings['Book-Rating']))
```

```
True
```

```
## The data type of User-ID should be numbers as well
print(is_numeric_dtype(ratings['User-ID']))
```

```
True
```

Data Preparation and Cleaning
Book-Ratings

3. Checking if the ISBN Column have aby special characters in it and removing it.

```
## Checking for any speacial characters within the ISBN column
flag = 0
k =[]
reg = "[^A-Za-z0-9]"

for x in ratings['ISBN']:
    z = re.search(reg,x)
    if z:
        flag = 1

if flag == 1:
    print("False")
else:
    print("True")
```

```
False
```

```
## These extra/special characters needs to removed in the ISBN column for this spe
bookISBN = books['ISBN'].tolist()
reg = "[^A-Za-z0-9]"
for index, row_Value in ratings.iterrows():
    z = re.search(reg, row_Value['ISBN'])
    if z:
        f = re.sub(reg,"",row_Value['ISBN'])
        if f in bookISBN:
            ratings.at[index , 'ISBN'] = f
```

Data Preparation and Cleaning
Book-Ratings

3. Upper casing the ISBN alphabets and removing duplicate information in the data set.

```
## Just like the Books data set, the alphabets in ISBN in this data set also needs
ratings['ISBN'] = ratings['ISBN'].str.upper()
```

```
## Duplicate records should also be removed
ratings.drop_duplicates(keep='last', inplace=True)
ratings.reset_index(drop=True, inplace=True)
```

Data Preparation and Cleaning

Merging the Tables

1. Inner joining the data sets:
- (Data set 1) Rating to Books through ISBN
- User to Data set 1 through User-ID

```
dataset = pd.merge(books, ratings, on='ISBN', how='inner')
dataset = pd.merge(dataset, users, on='User-ID', how='inner')
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1031609 entries, 0 to 1031608
Data columns (total 11 columns):
 #   Column               Non-Null Count    Dtype 
---  ------               --------------    ----- 
 0   ISBN                 1031609 non-null  object
 1   Book-Title           1031609 non-null  object
 2   Book-Author          1031609 non-null  object
 3   Year-Of-Publication  1031609 non-null  int32 
 4   Publisher            1031609 non-null  object
 5   User-ID              1031609 non-null  int64 
 6   Book-Rating          1031609 non-null  int64 
 7   Age                  1031609 non-null  int32 
 8   City                 1031609 non-null  object
 9   State                1031609 non-null  object
 10  Country              1031609 non-null  object
dtypes: int32(2), int64(2), object(7)
memory usage: 78.7+ MB
```

Data Preparation and Cleaning
Merging the Tables

2. Dividing the merge data to those rated of zero and those of greater than zero.

```
## Not equal to zero rating dataset
dataset1 = dataset[dataset['Book-Rating'] != 0]
dataset1 = dataset1.reset_index(drop = True)
dataset1.shape
```

```
(384074, 11)
```

```
## Equal to zero rating dataset
dataset2 = dataset[dataset['Book-Rating'] == 0]
dataset2 = dataset2.reset_index(drop = True)
dataset2.shape
```

```
(647535, 11)
```

Data Preparation and Cleaning

Merging the Tables

2. Dividing the merge data to those rated of zero and those of greater than zero.

```
##Check the condition of the data set
dataset1.head()
```

	ISBN	Book-Title	Book-Author	Year-Of-Publication	Publisher	User-ID	Book-Rating	Age	
0	0002005018	Clara Callan	Richard Bruce Wright	2001	HarperFlamingo Canada	8	5	35	timr
1	074322678X	Where You'll Find Me: And Other Stories	Ann Beattie	2002	Scribner	8	5	35	timr
2	0887841740	The Middle Stories	Sheila Heti	2004	House of Anansi Press	8	5	35	timr
3	1552041778	Jane Doe	R. J. Kaiser	1999	Mira Books	8	5	35	timr
4	1567407781	The Witchfinder (Amos Walker Mystery Series)	Loren D. Estleman	1998	Brilliance Audio - Trade	8	6	35	timr

Data Preparation and Cleaning

Merging the Tables

2. Dividing the merge data to those rated of zero and those of greater than zero.

```
dataset2.head()
```

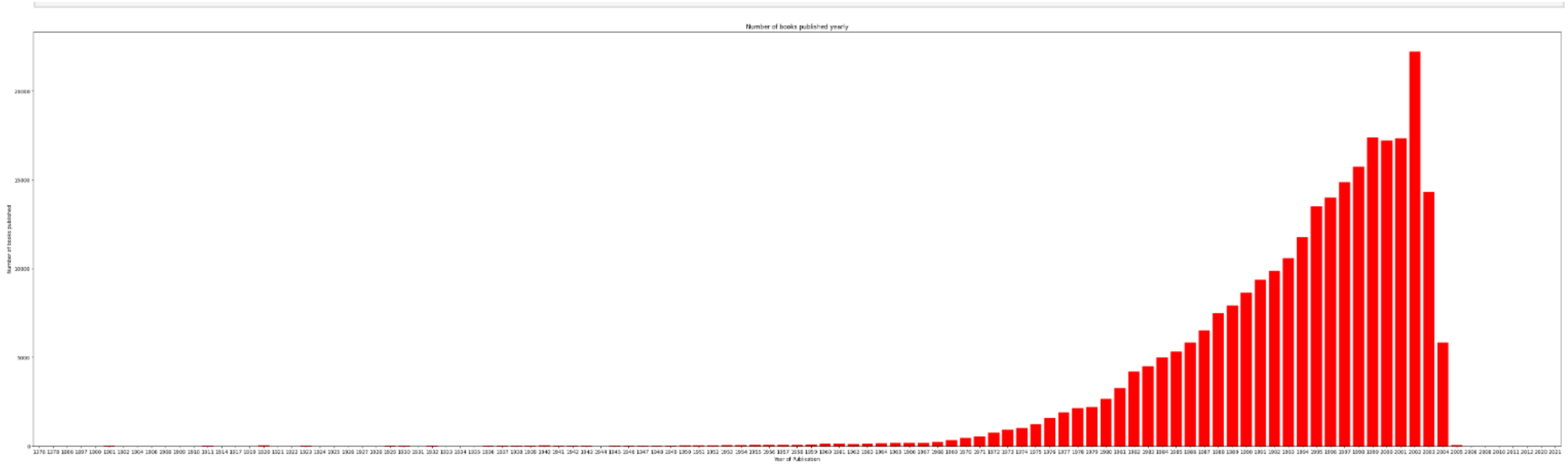
	ISBN	Book-Title	Book-Author	Year-Of-Publication	Publisher	User-ID	Book-Rating	Age	C
0	0195153448	Classical Mythology	Mark P. O. Morford	2002	Oxford University Press	2	0	18	stockt
1	0060973129	Decision in Normandy	Carlo D'Este	1991	HarperPerennial	8	0	35	timm
2	0374157065	Flu: The Story of the Great Influenza Pandemic of 1918 and the Search for the Virus That Caused It	Gina Bari Kolata	1999	Farrar Straus Giroux	8	0	35	timm
3	0393045218	The Mummies of Urumchi	E. J. W. Barber	1999	W. W. Norton & Company	8	0	35	timm
4	0399135782	The Kitchen God's	Amy Tan	1991	Putnam Pub Group	8	0	35	timm

Descriptive Analysis
Published books per year

1. Creating a bar chart for the number of books that are published yearly

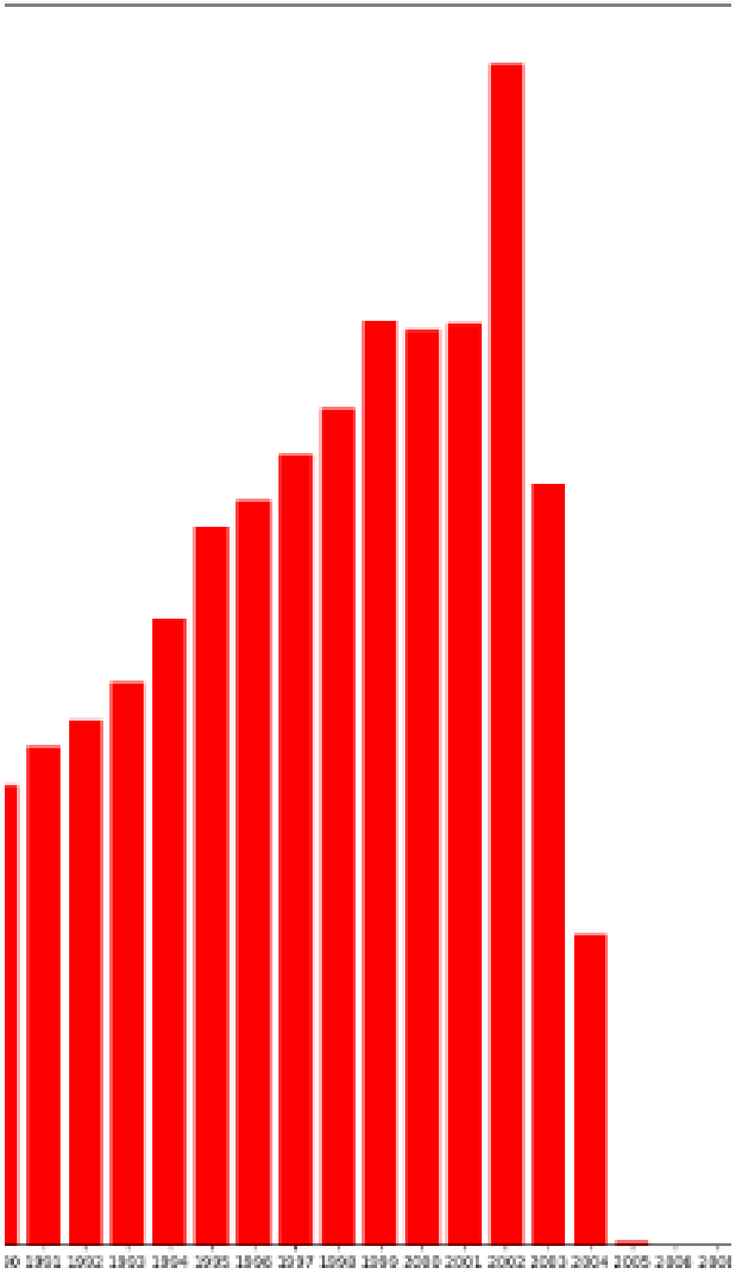
```
publications = {}
for year in books['Year-Of-Publication']:
    if str(year) not in publications:
        publications[str(year)] = 0
    publications[str(year)] +=1

publications = {k:v for k, v in sorted(publications.items())}

fig = plt.figure(figsize =(55, 15))
plt.bar(list(publications.keys()),list(publications.values()), color = 'red')
plt.ylabel("Number of books published")
plt.xlabel("Year of Publication")
plt.title("Number of books published yearly")
plt.margins(x = 0)
plt.show()
```

Descriptive Analysis
Published books per year



Descriptive Analysis
Published books per year

- The year where there are the most number of books published is 2002.
- The distribution is also skewed to the left.



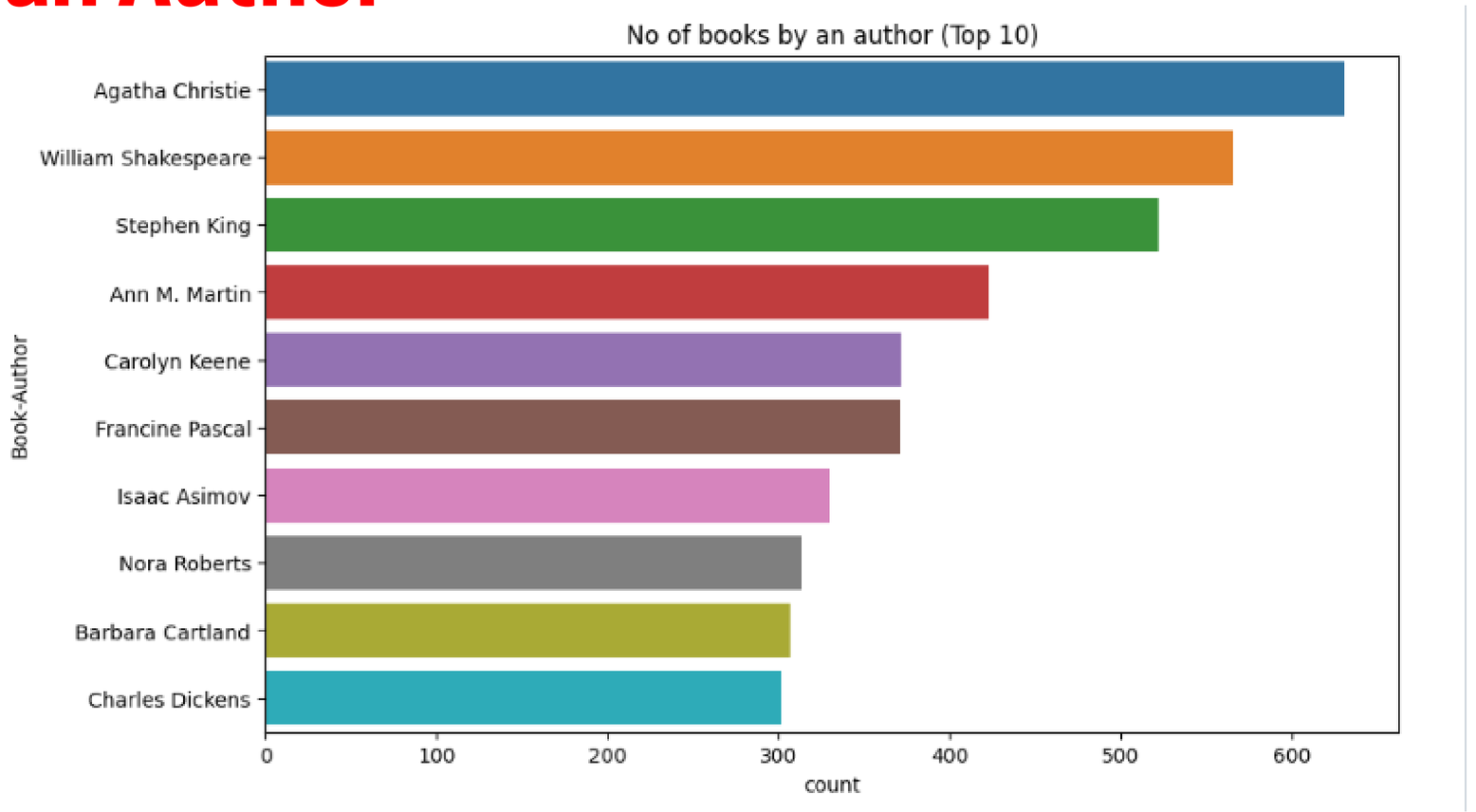
Descriptive Analysis
Number of books by an Author

```
plt.figure(figsize=(10,6))
sns.countplot(y="Book-Author", data=books,order=books['Book-Author'].value_counts(
plt.title("No of books by an author (Top 10)")
```

Descriptive Analysis

Number of books by an Author

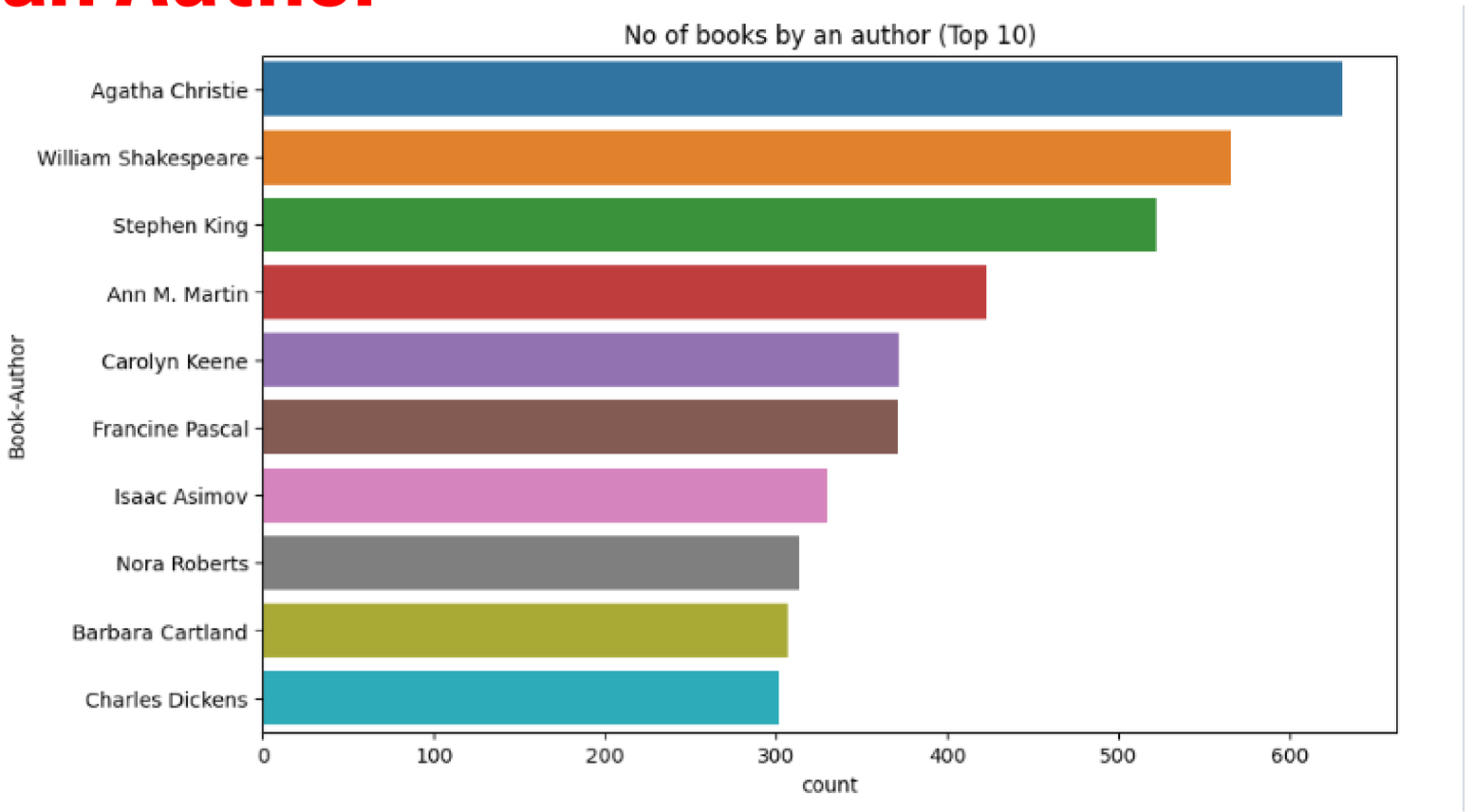
- As there are a lot of authors, we got the top 10 of those who had written the most number of books within the data set



Descriptive Analysis

Number of books by an Author

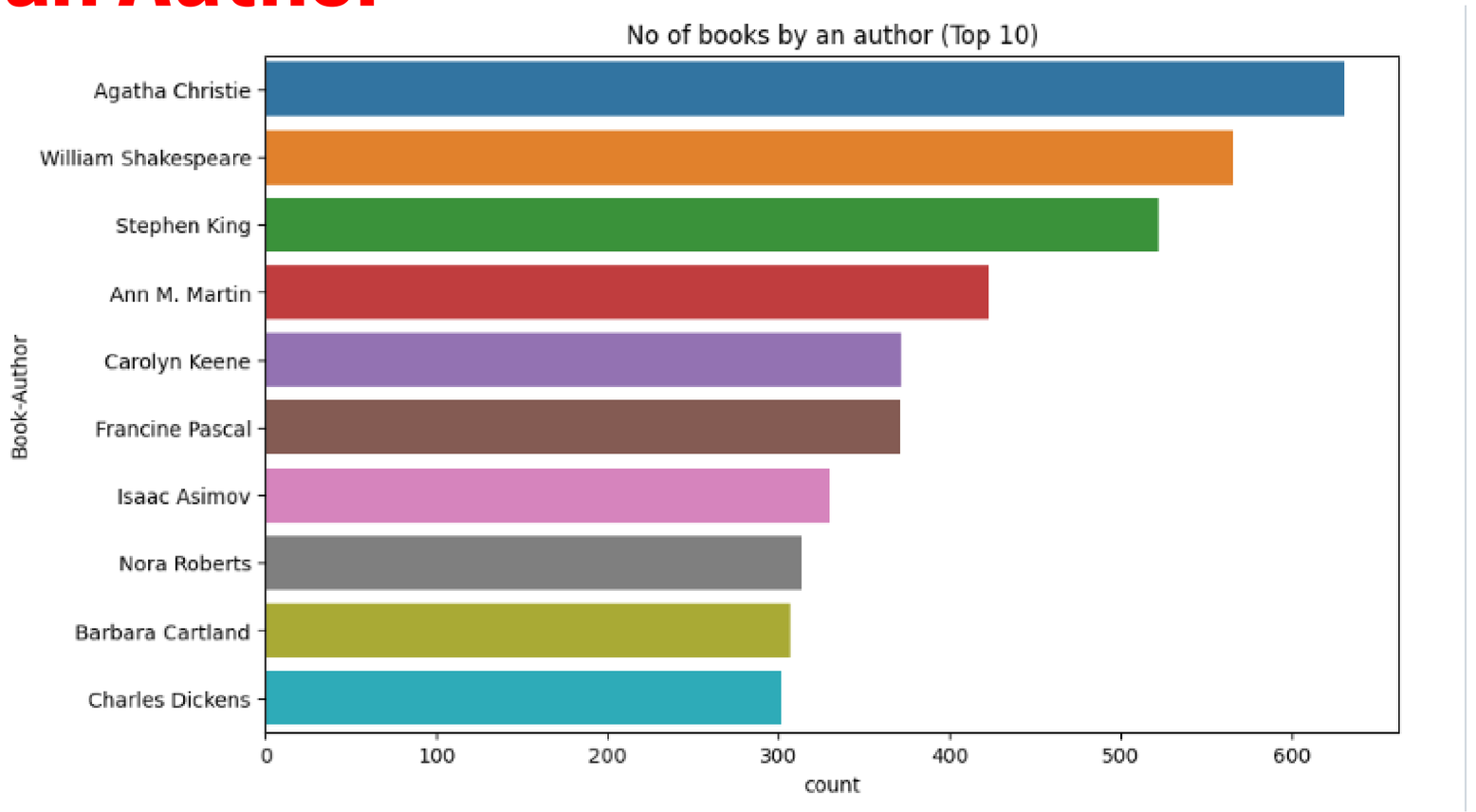
- Agatha Christie comes first which has more than 600 books in the data set. This includes the different versions as well.



Descriptive Analysis

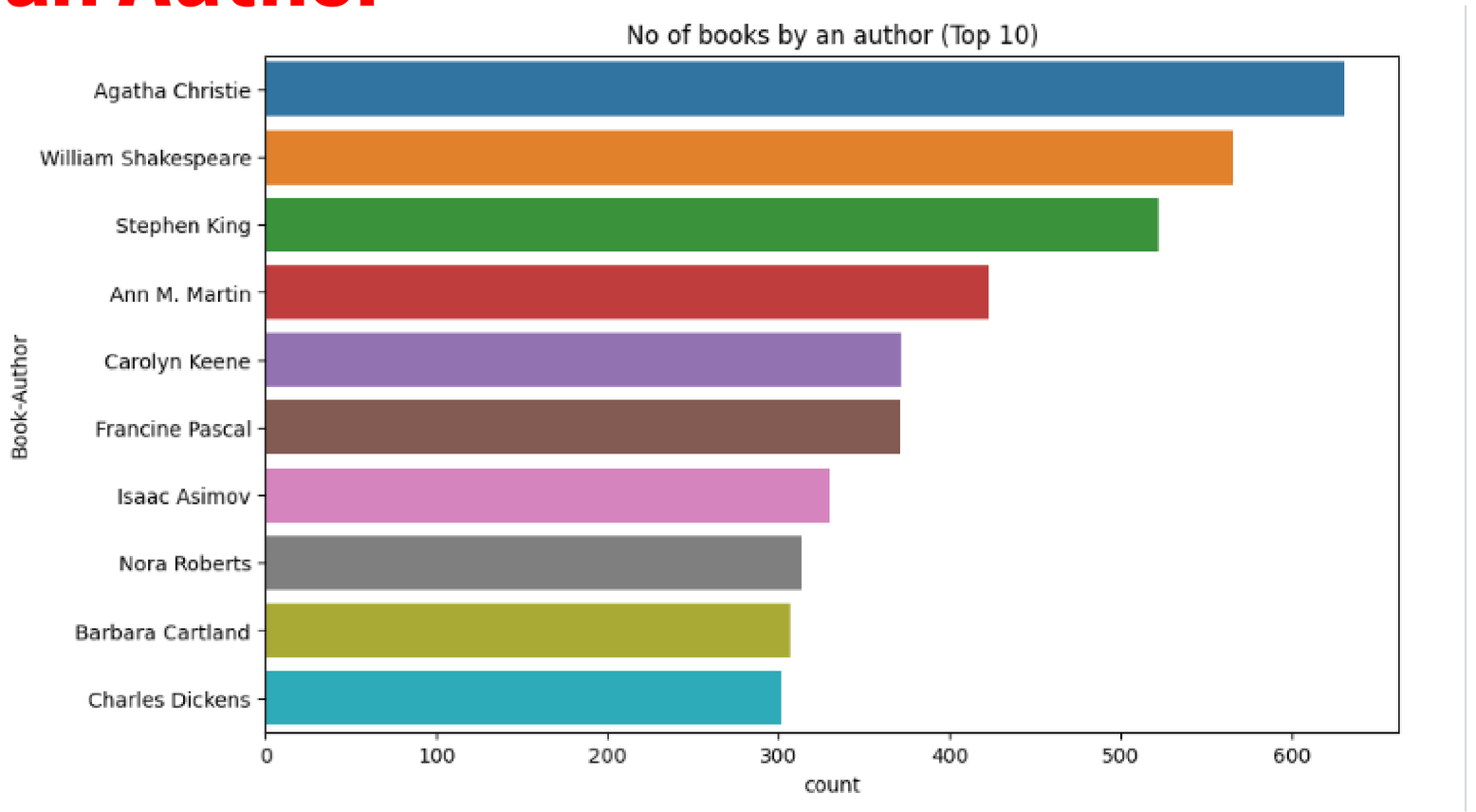
Number of books by an Author

- She is known for her detective and mystery novels and among her works was The Murder in the Orient Express.



Descriptive Analysis
Number of books by an Author

- The following authors were also known for their works which we also use for educational references as well.

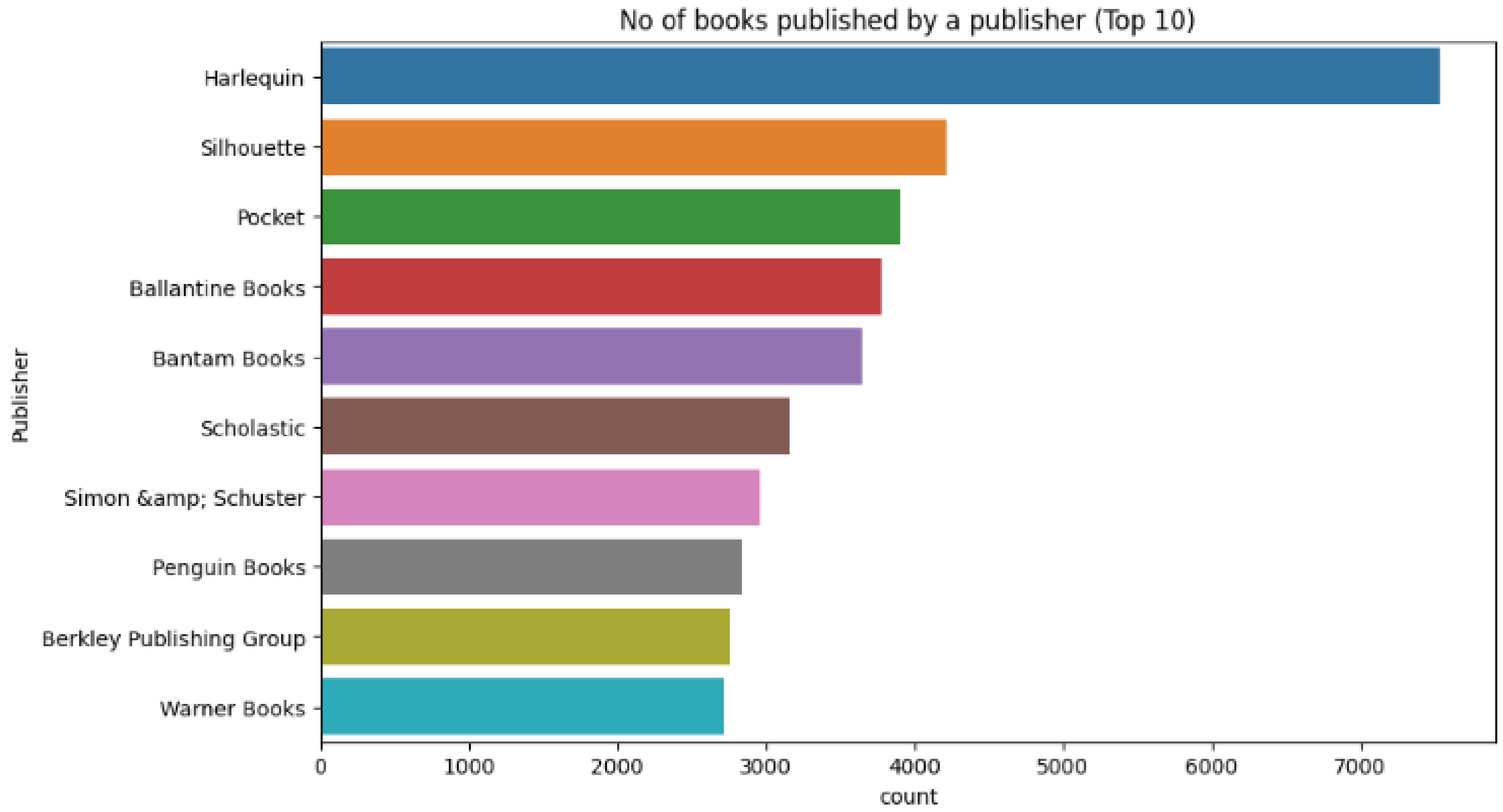


Descriptive Analysis
Number of books by a Publisher

```
plt.figure(figsize=(10,6))
sns.countplot(y="Publisher", data=books,order=books['Publisher'].value_counts().inc
plt.title("No of books published by a publisher (Top 10)")
```

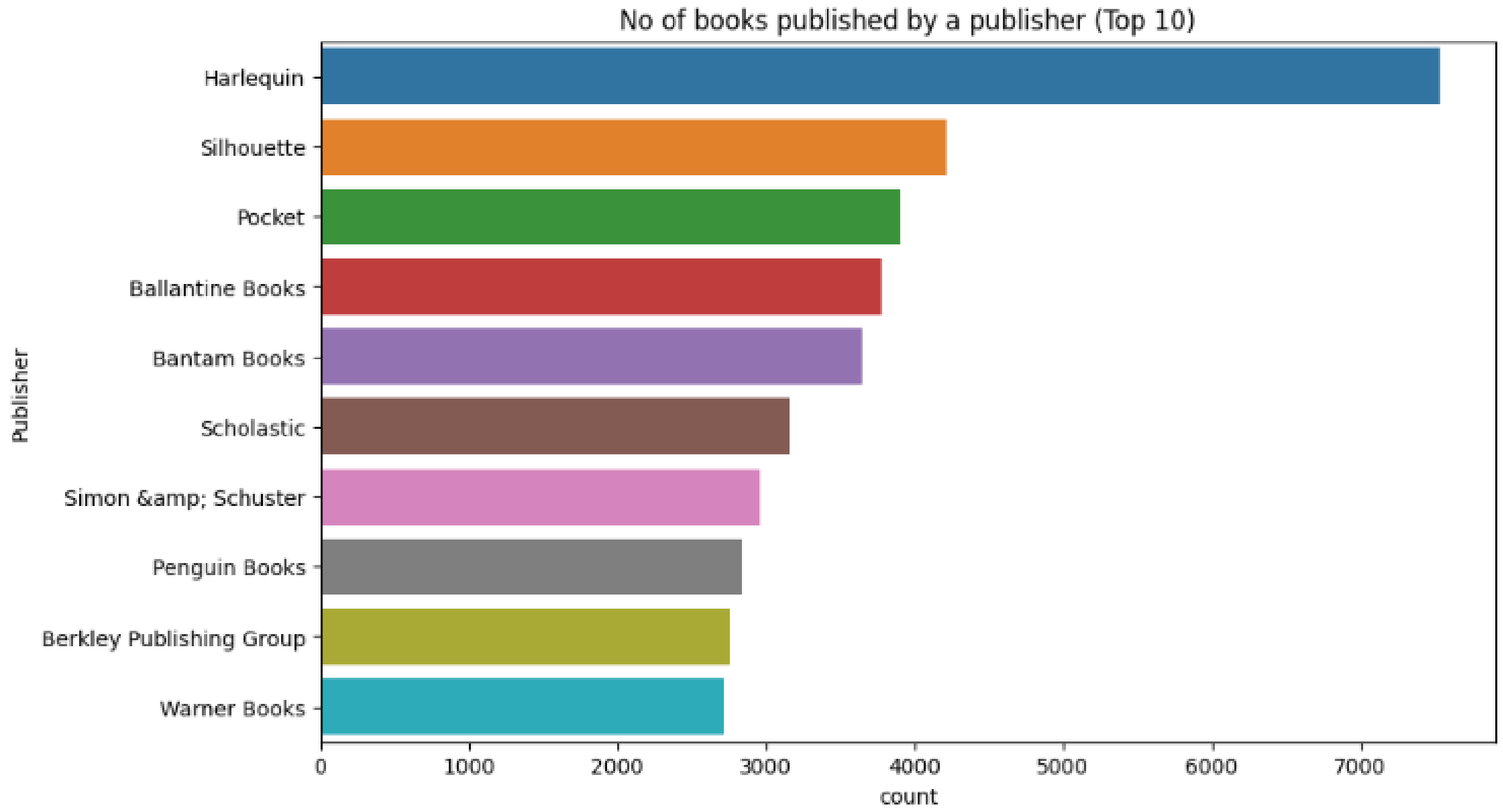
Descriptive Analysis
Number of books by a Publisher

- Harlequin Publishing comes to the top with over 7000 published books under their belt.



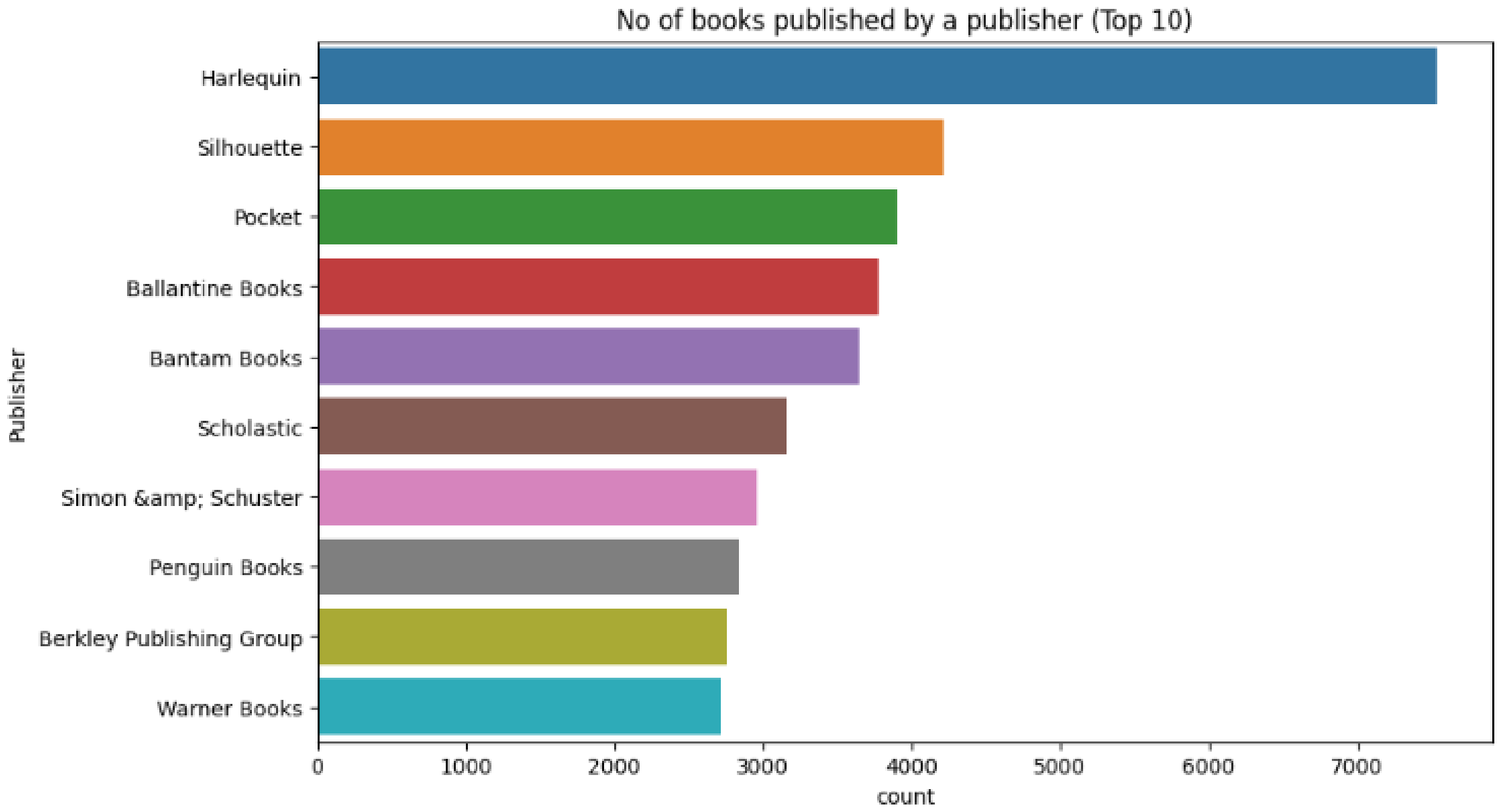
Descriptive Analysis
Number of books by a Publisher

- It is known to publish romance and women's fiction since 1949.



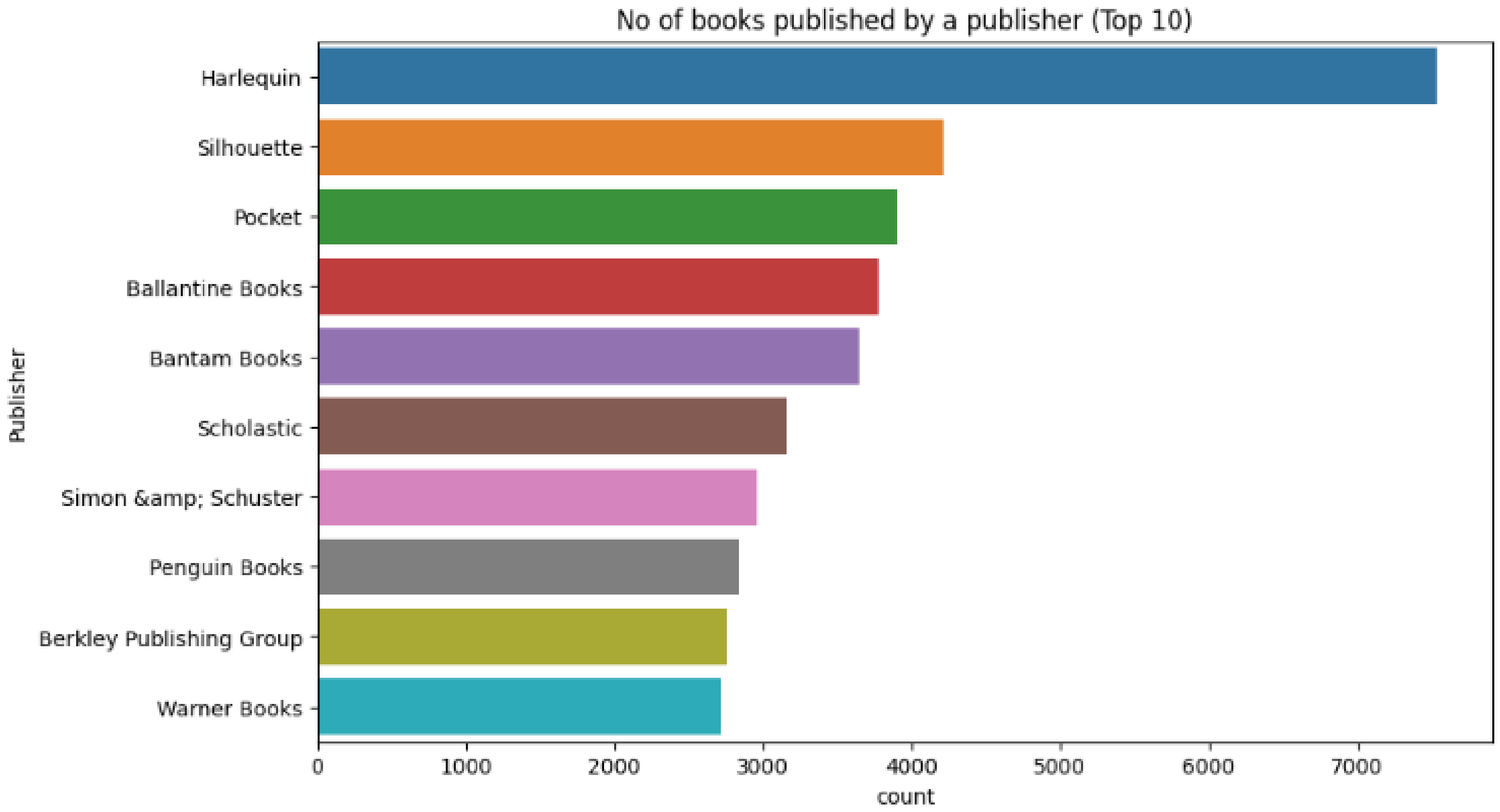
Descriptive Analysis
Number of books by a Publisher

- The following authors have also set their record within the top ten such as Sillhoute known as the publishing house of Nora Roberts



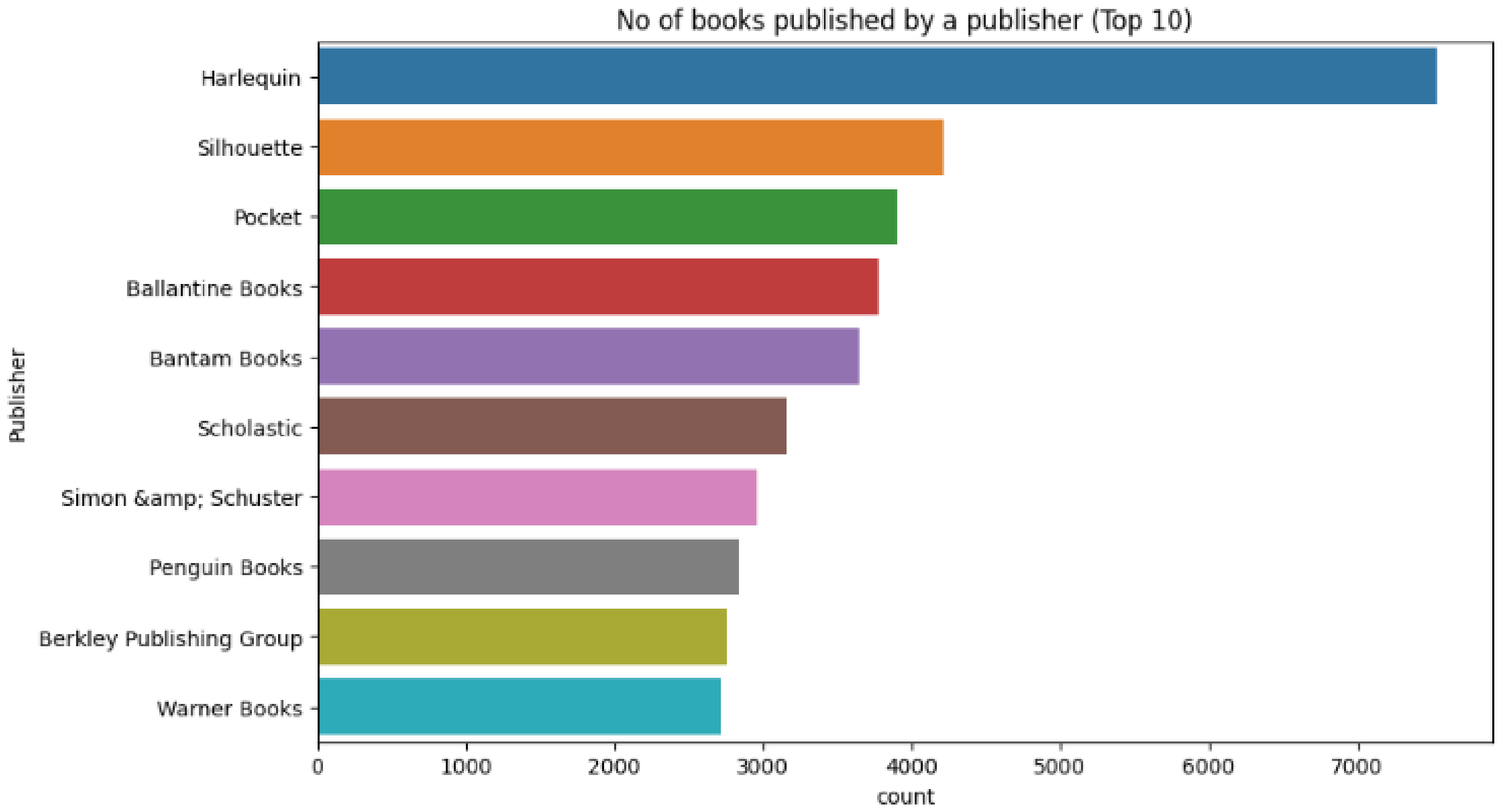
Descriptive Analysis
Number of books by a Publisher

- Scholastic which had been the publishing house of J.K. Rowling's Harry Potter Series.



Descriptive Analysis
Number of books by a Publisher

- Scholastic which had been the publishing house of J.K. Rowling's Harry Potter Series.

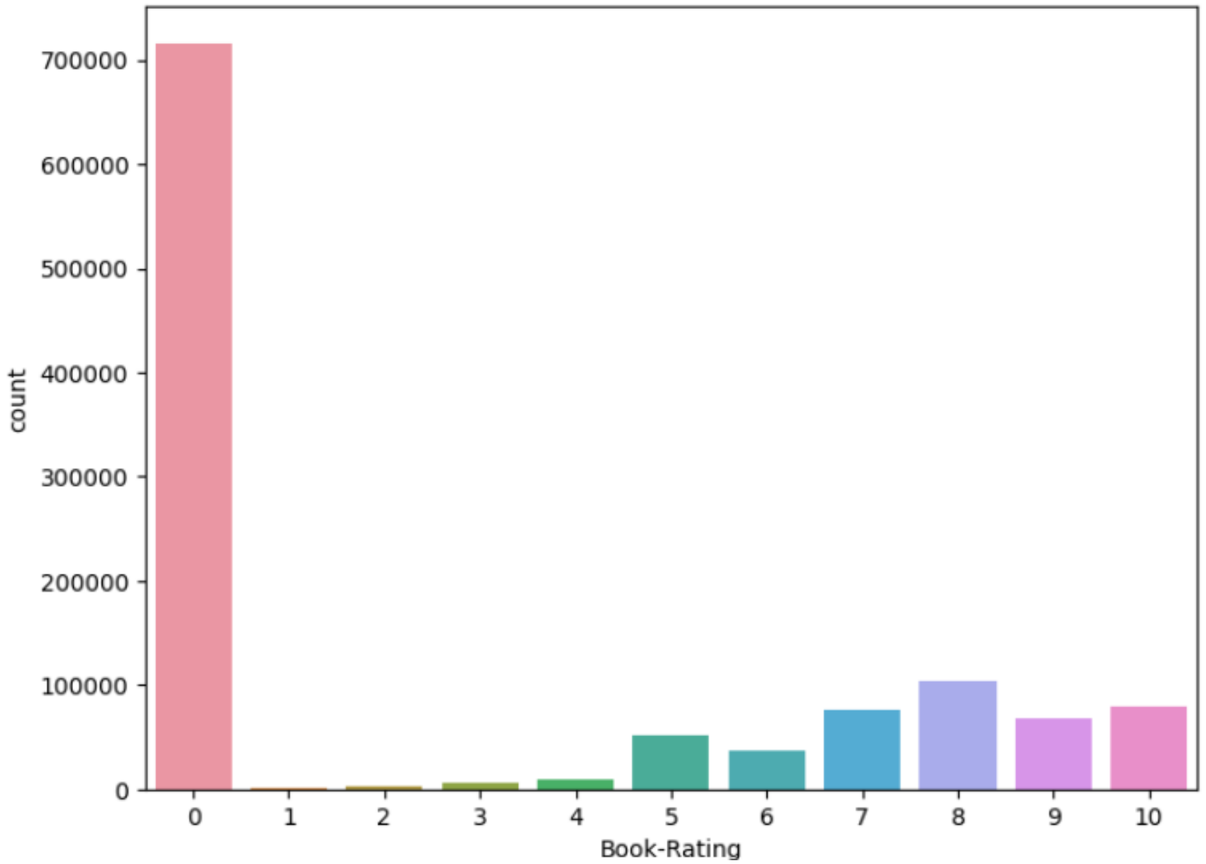


Descriptive Analysis
Frequency of Rating: 0-10

```
plt.figure(figsize=(8,6))
sns.countplot(x="Book-Rating", data=ratings)
```

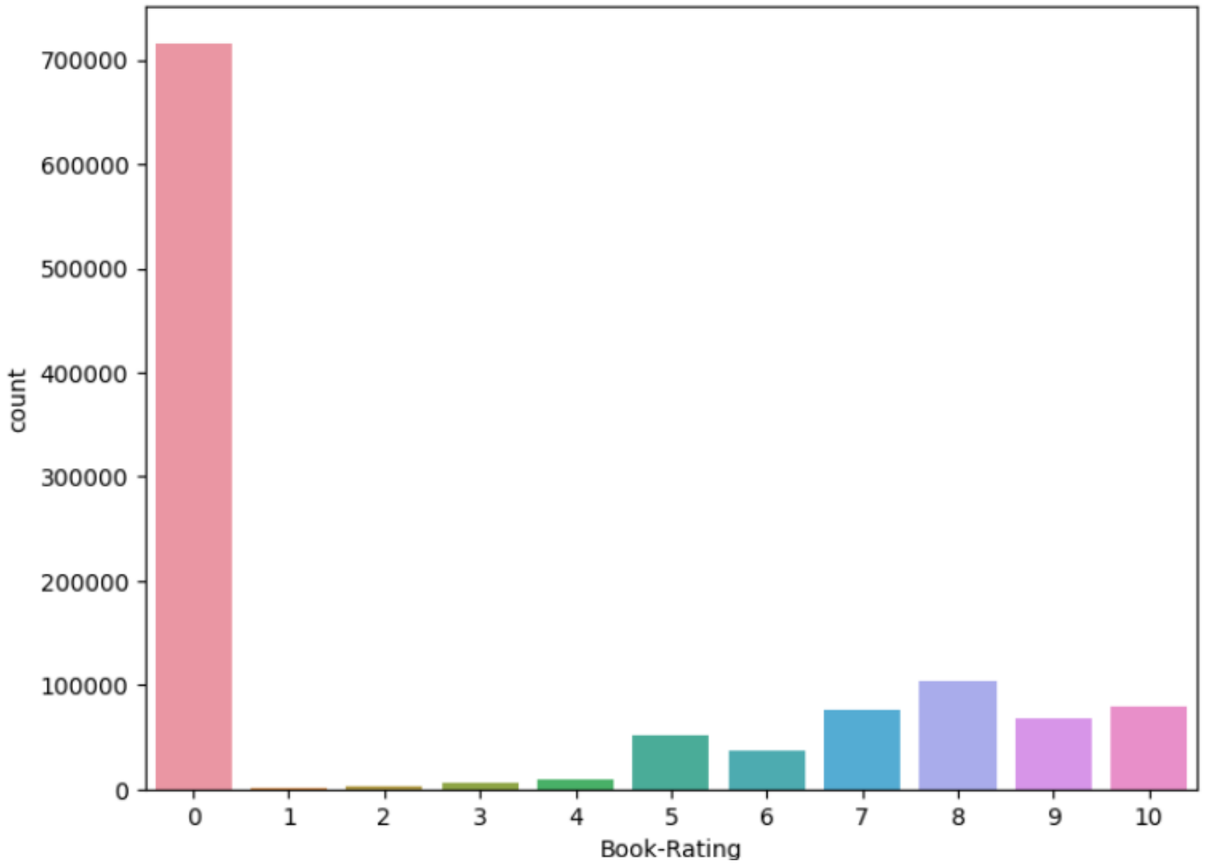
Descriptive Analysis
Frequency of Rating: 0-10

- The rating of zero had been prominent within the data set.
- Factors are still to be assessed what influences readers to give a particular rating.



Descriptive Analysis
Frequency of Rating: 0-10

- This makes the data skewed to the right where ratings are more prominent in lower rates among the readers included in the data set.

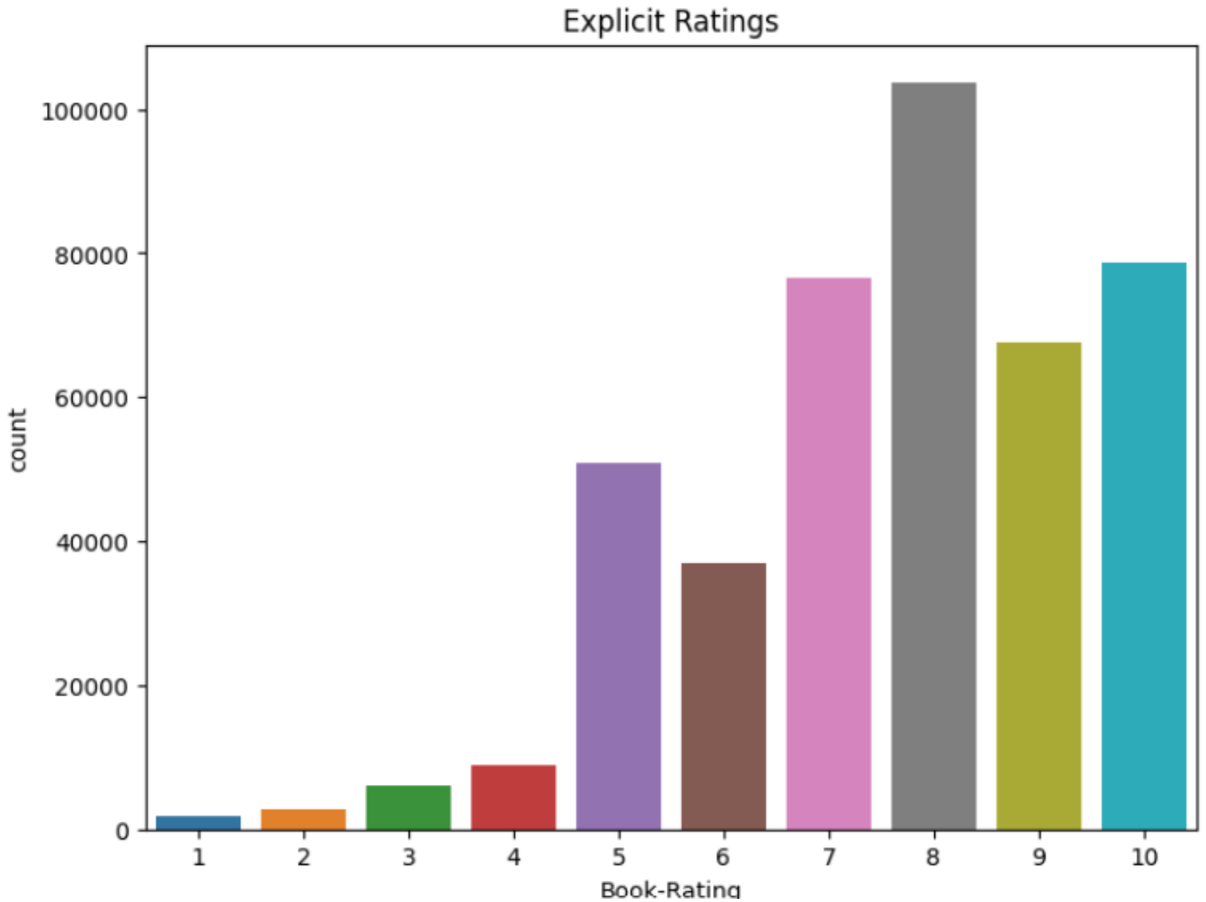


Descriptive Analysis
Frequency of Rating:>0

```
plt.figure(figsize=(8,6))
data = ratings[ratings['Book-Rating'] != 0]
sns.countplot(x="Book-Rating", data=data)
plt.title("Explicit Ratings")
```

Descriptive Analysis
Frequency of Rating: >0

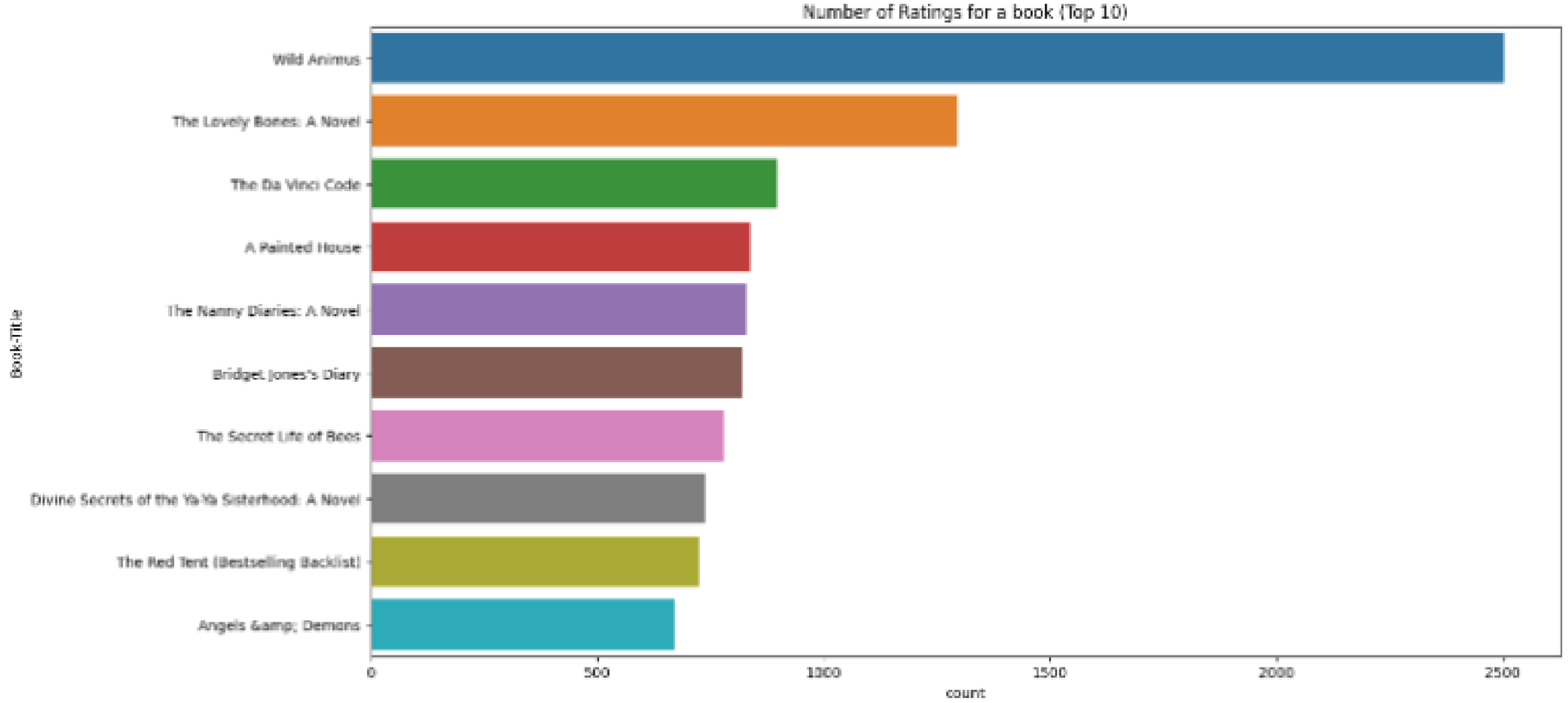
- Among the ratings greater than 0, most readers had given and 8 for the ratings o the books they have read.



Descriptive Analysis
Ratings for a book (Top 10)

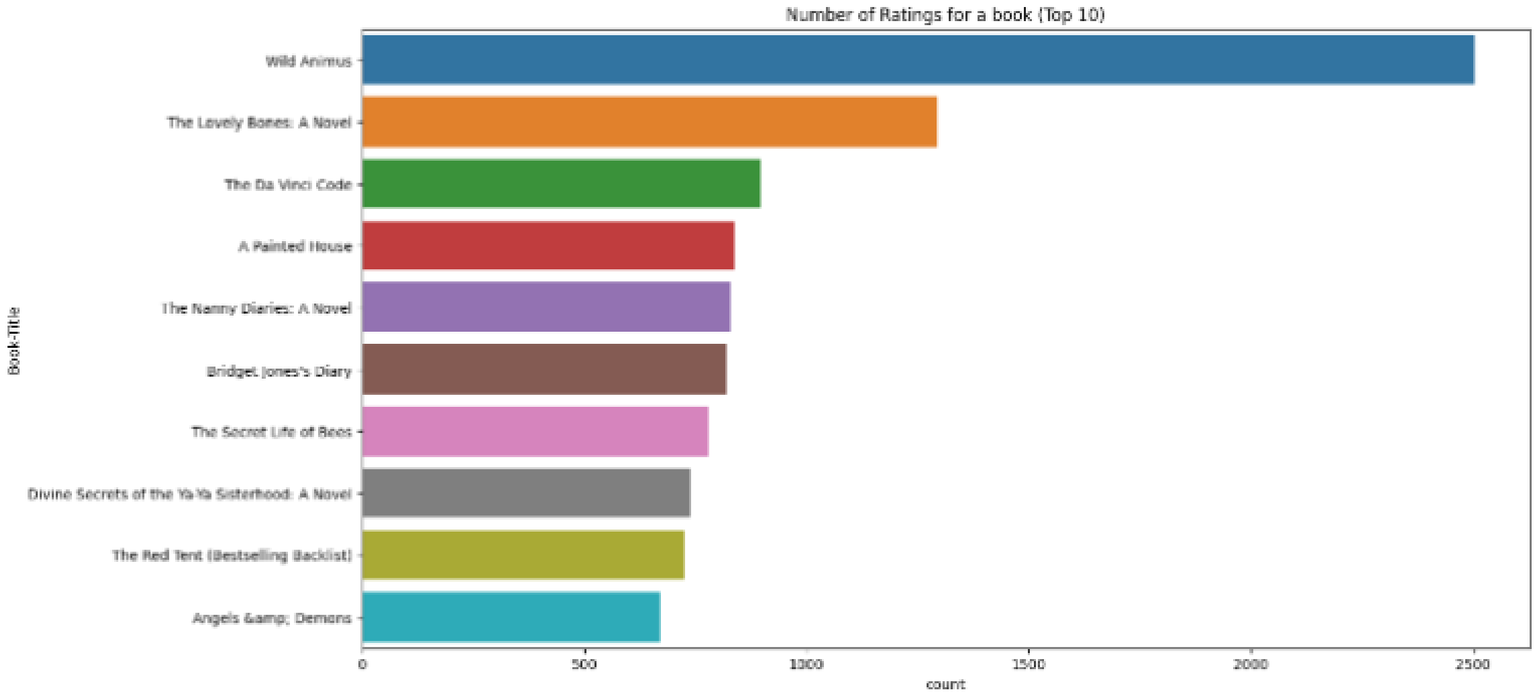
```
plt.figure(figsize=(15,8))
sns.countplot(y="Book-Title", data=dataset, order=dataset['Book-Title'].value_count
plt.title("Number of Ratings for a book (Top 10)")
```

Descriptive Analysis
Ratings for a book (Top 10)



Descriptive Analysis
Ratings for a book (Top 10)

- Among the books Wild Animus had recieved the mos ratings, this inclu ratings from 0-10



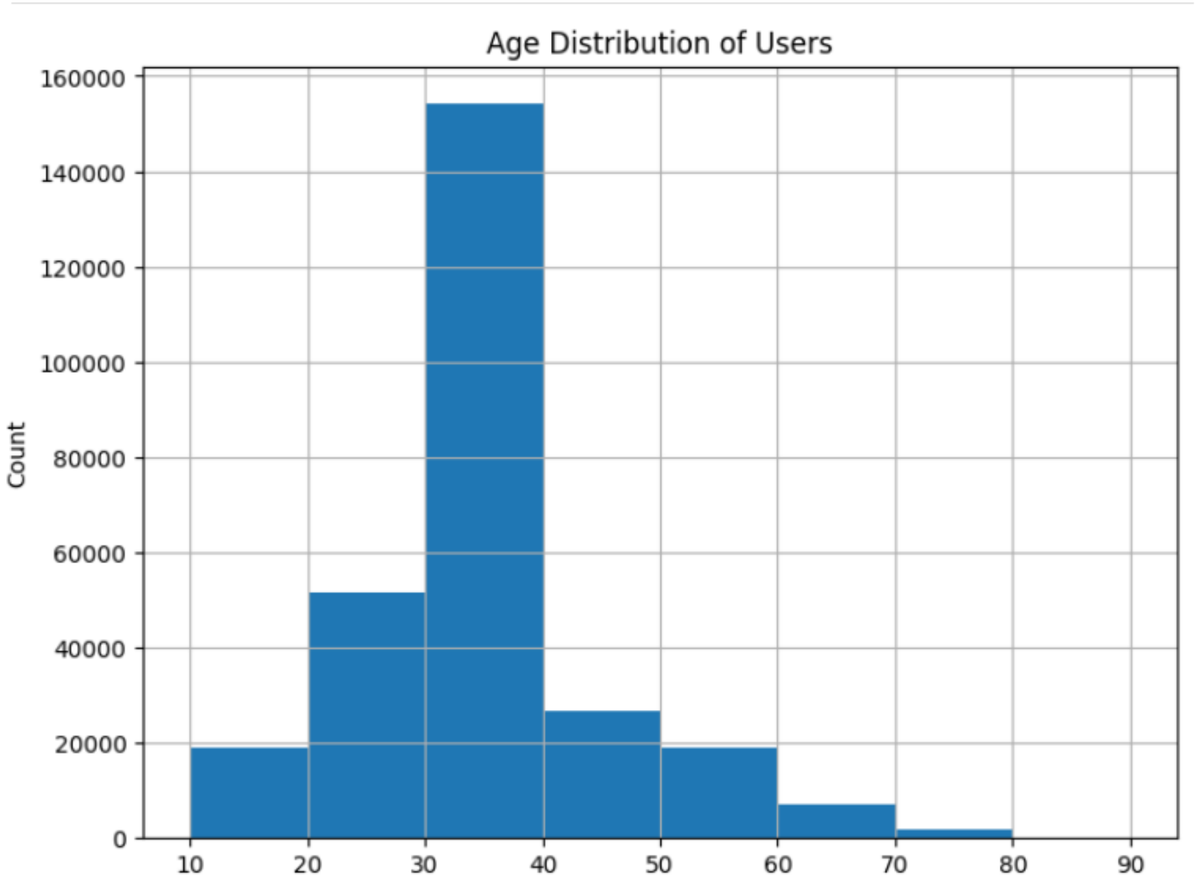
Descriptive Analysis
Age Distribution of Users

```
plt.figure(figsize=(8,6))
users.Age.hist(bins=[10*i for i in range(1, 10)])
plt.title('Age Distribution of Users')
plt.xlabel('Age')
plt.ylabel('Count')
plt.show()
```

Descriptive Analysis

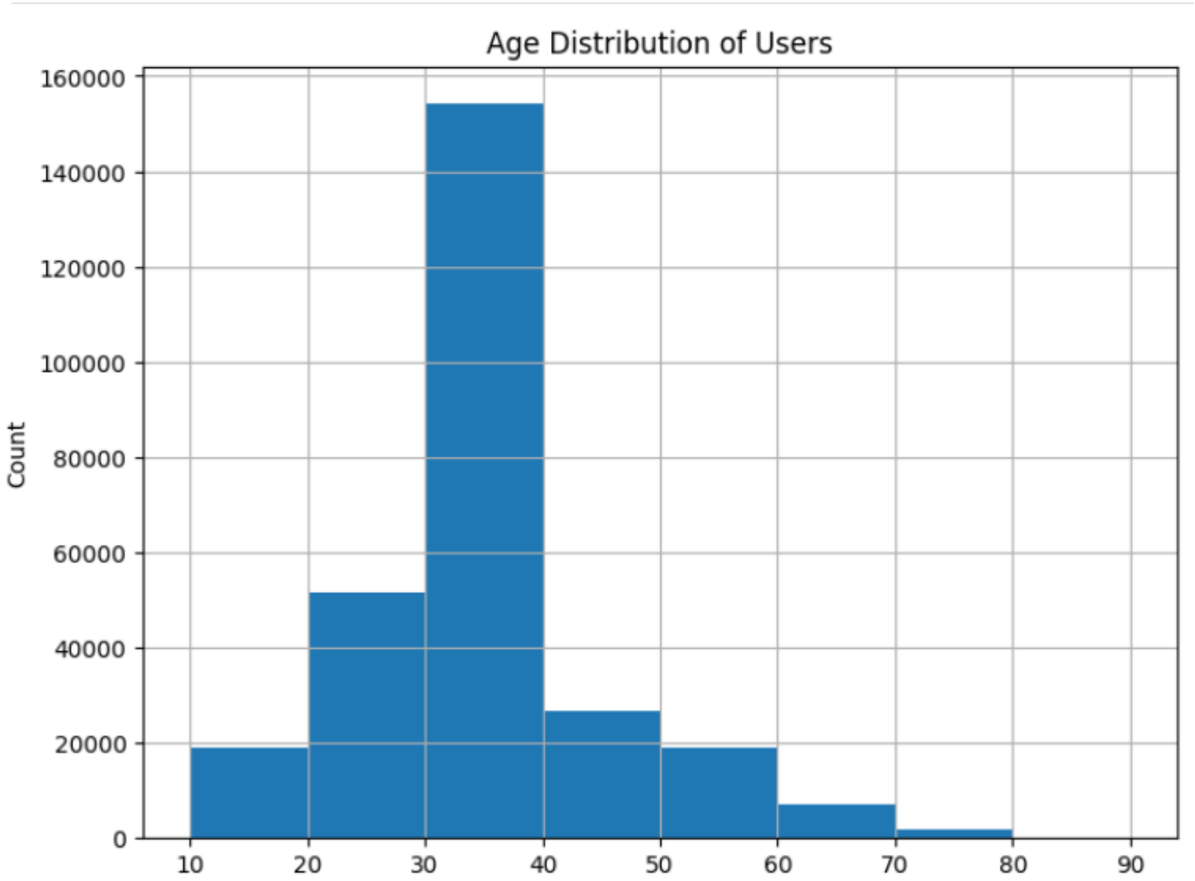
Age Distribution od Users

- Within the data set, users age 30-40 are the ones who frequently use the books.



Descriptive Analysis
Age Distribution of Users

- The distribution of ages of users are skewed to the right.

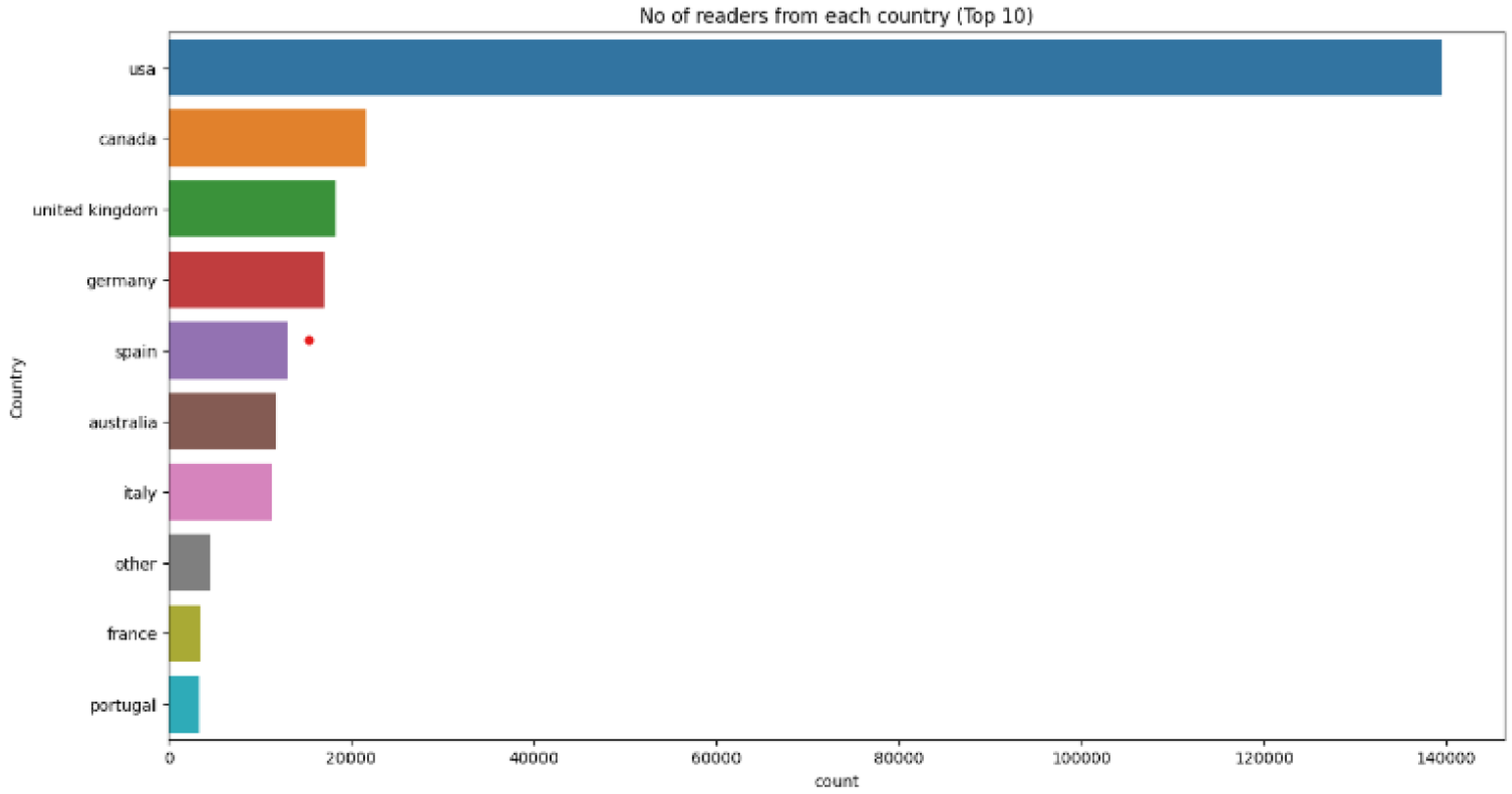


Descriptive Analysis
Number of reader for each country (Top 10)

```
plt.figure(figsize=(15,8))
sns.countplot(y="Country", data=users, order=users['Country'].value_counts().index
plt.title("No of readers from each country (Top 10)")
```

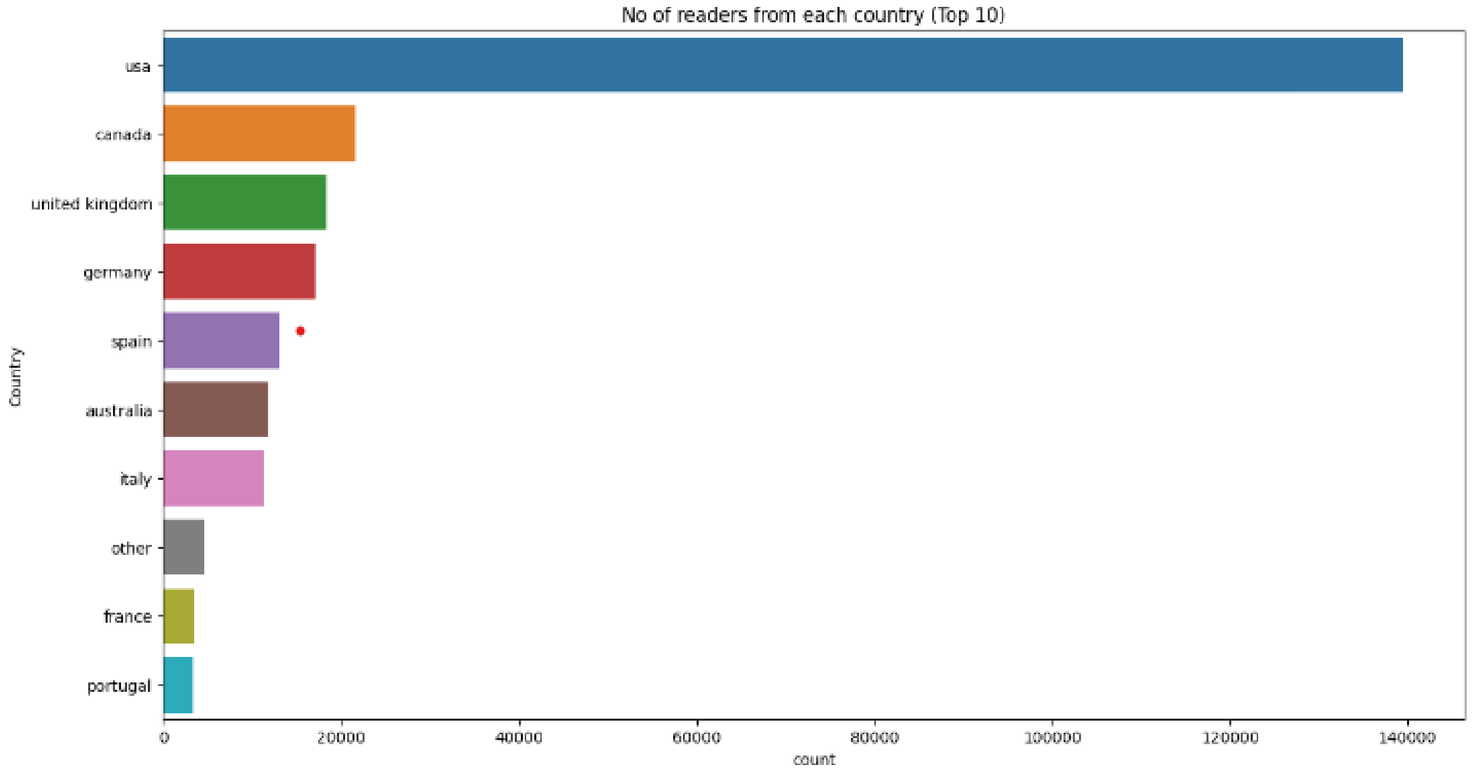
Descriptive Analysis

Number of reader for each country (Top 10)



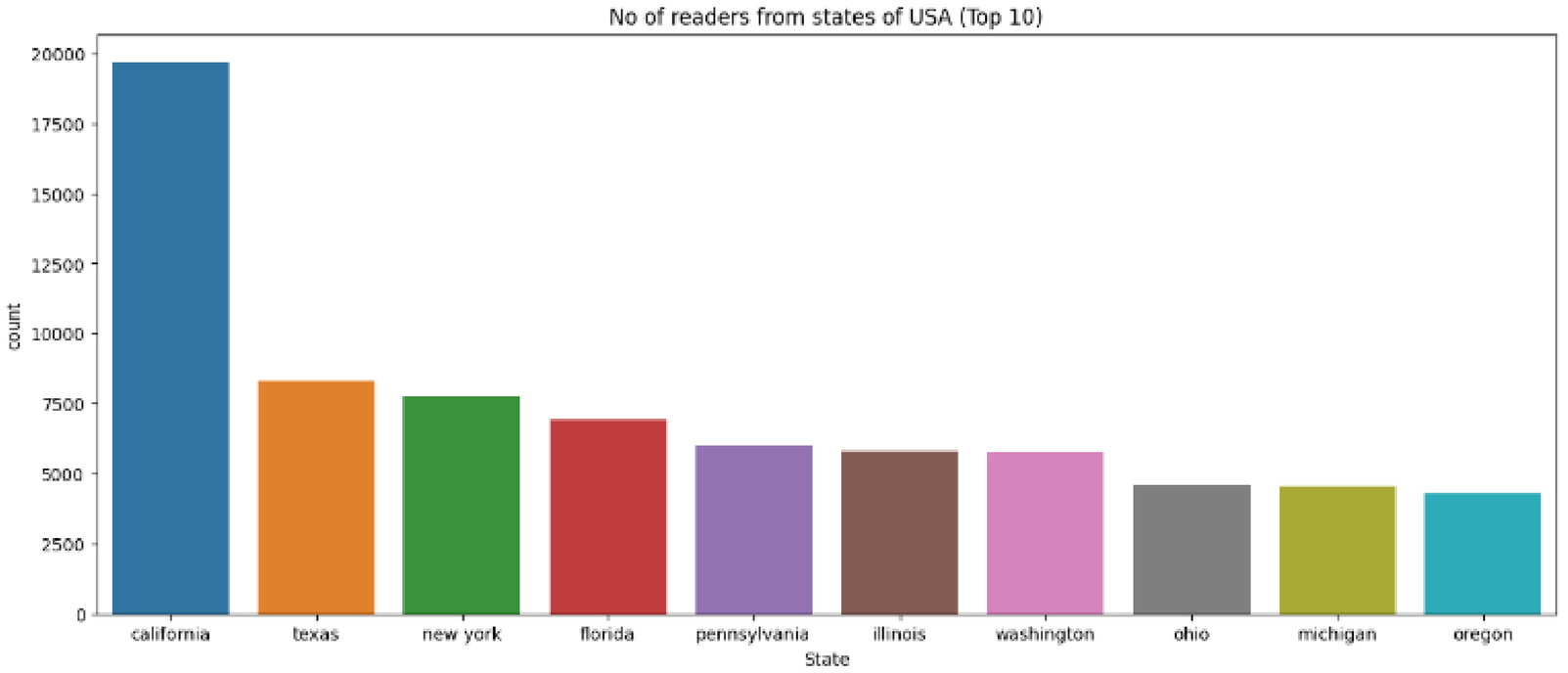
Descriptive Analysis
Number of reader for each city (Top 10)

- Most readers recorded came from the USA



Descriptive Analysis
Number of reader for each city (Top 10)

- Specifically in California



Recommendations

- KNN (Nearest Neighbors) algorithm
  - cluster of similar users based on common book ratingd
  - uses the average rating of top-k nearest neighbors. (Li, 2017).

Recommendations

What will you recommend for your self?

```
bookName = input("Enter a book name: ")
number = int(input("Enter number of books to recommend: "))
```

```
Enter a book name: Harry Potter and the Sorcerer's Stone (Harry Potter (Paperback))
Enter number of books to recommend: 5
```

```
data = (dataset1.groupby(by = ['Book-Title'])['Book-Rating'].count().reset_index()
        rename(columns = {'Book-Rating': 'Total-Rating'})[['Book-Title', 'Total-Rat

result = pd.merge(data, dataset1, on='Book-Title')
result = result[result['Total-Rating'] >= popularity_threshold]
result = result.reset_index(drop = True)

matrix = result.pivot_table(index = 'Book-Title', columns = 'User-ID', values = 'Bo
up_matrix = csr_matrix(matrix)
```

```
model = NearestNeighbors(metric = 'cosine', algorithm = 'brute')
model.fit(up_matrix)

distances, indices = model.kneighbors(matrix.loc[bookName].values.reshape(1, -1), n
print("\nRecommended books:\n")
for i in range(0, len(distances.flatten())):
    if i > 0:
        print(matrix.index[indices.flatten()[i]])
```

Recommendations

What will you recommend for your self?

As an avid fan of the Harry Potter Series, both in books and movies, I've used this as the book to base the recommendation for myself.

Recommendations

What will you recommend for your self?

As a result, it had shown the one in the image which also included the Lord of the Rings which almost have the same genre, fiction and fantasy.

```
Recommended books:

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 Order of the Phoenix (Book 5)
The Fellowship of the Ring (The Lord of the Rings, Part 1)
```

Recommendations

What would you recommend to readers who enjoyed The Lord of the Rings?

```
bookName = input("Enter a book name: ")
number = int(input("Enter number of books to recommend: "))
```

```
Enter a book name: The Two Towers (The Lord of the Rings, Part 2)
Enter number of books to recommend: 5
```

```
data = (dataset1.groupby(by = ['Book-Title'])['Book-Rating'].count().reset_index()
        rename(columns = {'Book-Rating': 'Total-Rating'})[['Book-Title', 'Total-Ra

result = pd.merge(data, dataset1, on='Book-Title')
result = result[result['Total-Rating'] >= popularity_threshold]
result = result.reset_index(drop = True)

matrix = result.pivot_table(index = 'Book-Title', columns = 'User-ID', values = 'B
up_matrix = csr_matrix(matrix)
```

```
model = NearestNeighbors(metric = 'cosine', algorithm = 'brute')
model.fit(up_matrix)

distances, indices = model.kneighbors(matrix.loc[bookName].values.reshape(1, -1),
print("\nRecommended books:\n")
for i in range(0, len(distances.flatten())):
    if i > 0:
        print(matrix.index[indices.flatten()[i]])
```

Recommendations

What would you recommend to readers who enjoyed The Lord of the Rings?

As a result, it had also shown the different books of The Lord of the Rings and Harry Potter which as mentioned almost have the same genre as well.

```
Recommended books:

The Return of the King (The Lord of the Rings, Part 3)
The Fellowship of the Ring (The Lord of the Rings, Part 1)
The Hobbit : The Enchanting Prelude to The Lord of the Rings
Harry Potter and the Prisoner of Azkaban (Book 3)
Harry Potter and the Sorcerer's Stone (Book 1)
```

Recommendations

What would you recommend to readers who enjoyed The Lord of the Rings?

The genre may have allowed the same users to have almost the same choice in books but will still need to be proven.

```
Recommended books:

The Return of the King (The Lord of the Rings, Part 3)
The Fellowship of the Ring (The Lord of the Rings, Part 1)
The Hobbit : The Enchanting Prelude to The Lord of the Rings
Harry Potter and the Prisoner of Azkaban (Book 3)
Harry Potter and the Sorcerer's Stone (Book 1)
```

Recommendations

What would you recommend to readers who enjoyed The Lord of the Rings and Cloud Atlas?

```
bookName = input("Enter a book name: ")
number = int(input("Enter number of books to recommend: "))
```

```
Enter a book name: The Cloud Atlas
Enter number of books to recommend: 5
```

```
data = (dataset1.groupby(by = ['Book-Title'])['Book-Rating'].count().reset_index()
        rename(columns = {'Book-Rating': 'Total-Rating'})[['Book-Title', 'Total-Rat

result = pd.merge(data, dataset1, on='Book-Title')
result = result[result['Total-Rating'] >= popularity_threshold]
result = result.reset_index(drop = True)

matrix = result.pivot_table(index = 'Book-Title', columns = 'User-ID', values = 'Bo
up_matrix = csr_matrix(matrix)
```

```
model = NearestNeighbors(metric = 'cosine', algorithm = 'brute')
model.fit(up_matrix)

distances, indices = model.kneighbors(matrix.loc[bookName].values.reshape(1, -1), n
print("\nRecommended books:\n")
for i in range(0, len(distances.flatten())):
    if i > 0:
        print(matrix.index[indices.flatten()[i]])
```

Recommendations

What would you recommend to readers who enjoyed The Lord of the Rings and Cloud Atlas?

The book was originally searched together with The Lord of the Rings but unfortunately was encountered by a key error. It was then separately but with as a lone book, still encountered the same key error.

Recommendations

What would you recommend to readers who enjoyed The Lord of the Rings and Cloud Atlas?

```
File ~\AppData\Local\Programs\Python\Python311\Lib\site-packages\pandas\core\indexes
\base.py:3654, in Index.get_loc(self, key)
   3652     return self._engine.get_loc(casted_key)
   3653 except KeyError as err:
-> 3654     raise KeyError(key) from err
   3655 except TypeError:
   3656     # If we have a listlike key, _check_indexing_error will raise
   3657     #  InvalidIndexError. Otherwise we fall through and re-raise
   3658     #  the TypeError.
   3659     self._check_indexing_error(key)

KeyError: 'The Cloud Atlas'
```

Recommendations

What would you recommend to readers who enjoyed The Lord of the Rings and Cloud Atlas?

NOTE:

- KNN heavily relies on ratings that goes the same with other recommendation system.
- Data set used to run KNN are the data set of books with ratings.
- If the book isn't rated then no recommendations will be given.

Recommendations

What would you recommend to readers who enjoyed The Lord of the Rings and Cloud Atlas?

The book is then run under a different recommendation system, The once with same author, same publisher but yeild the same result.

Recommendations

```
bookName = input("Enter a book name: ")
number = int(input("Enter number of books to recommend: "))
```

```
Enter a book name: The Cloud Atlas
Enter number of books to recommend: 5
```

```
def printBook(k, n):
    z = k['Book-Title'].unique()
    for x in range(len(z)):
        print(z[x])
        if x >= n-1:
            break
```

```
def get_books(dataframe, name, n):
    print("\nBooks by same Author:\n")
    au = dataframe['Book-Author'].unique()

    data = dataset1[dataset1['Book-Title'] != name]

    if au[0] in list(data['Book-Author'].unique()):
        k2 = data[data['Book-Author'] == au[0]]
    k2 = k2.sort_values(by=['Book-Rating'])
    printBook(k2, n)

    print("\n\nBooks by same Publisher:\n")
    au = dataframe['Publisher'].unique()

    if au[0] in list(data['Publisher'].unique()):
        k2 = pd.DataFrame(data[data['Publisher'] == au[0]])
    k2=k2.sort_values(by=['Book-Rating'])
    printBook(k2, n)
```

```
if bookName in list(dataset1['Book-Title'].unique()):
    d = dataset1[dataset1['Book-Title'] == bookName]
    get_books(d, bookName, number)
else:
    print("Invalid Book Name!!")
```

Recommendations

What would you recommend to readers who enjoyed The Lord of the Rings and Cloud Atlas?

As seen in this image, k2 was said to be not associated with any value which is the Book Rating.

```
---------------------------------------------------------------------------
UnboundLocalError                         Traceback (most recent call last)
Cell In[92], line 3
      1 if bookName in list(dataset1['Book-Title'].unique()):
      2     d = dataset1[dataset1['Book-Title'] == bookName]
----> 3     get_books(d, bookName, number)
      4 else:
      5     print("Invalid Book Name!!")

Cell In[91], line 9, in get_books(dataframe, name, n)
      7 if au[0] in list(data['Book-Author'].unique()):
      8     k2 = data[data['Book-Author'] == au[0]]
----> 9 k2 = k2.sort_values(by=['Book-Rating'])
     10 printBook(k2, n)
     12 print("\n\nBooks by same Publisher:\n")

UnboundLocalError: cannot access local variable 'k2' where it is not associated with
a value
```

Recommendations

What would you recommend to readers who enjoyed The Lord of the Rings and Cloud Atlas?

This shows that the book may not have been rated or was rated zero.

```
---------------------------------------------------------------------------
UnboundLocalError                         Traceback (most recent call last)
Cell In[92], line 3
      1 if bookName in list(dataset1['Book-Title'].unique()):
      2     d = dataset1[dataset1['Book-Title'] == bookName]
----> 3     get_books(d, bookName, number)
      4 else:
      5     print("Invalid Book Name!!")

Cell In[91], line 9, in get_books(dataframe, name, n)
      7 if au[0] in list(data['Book-Author'].unique()):
      8     k2 = data[data['Book-Author'] == au[0]]
----> 9 k2 = k2.sort_values(by=['Book-Rating'])
     10 printBook(k2, n)
     12 print("\n\nBooks by same Publisher:\n")

UnboundLocalError: cannot access local variable 'k2' where it is not associated with
a value
```

Recommendations

What would you recommend to readers who enjoyed The Lord of the Rings and Cloud Atlas?

To verify this, we have run a query under Microsoft SQL server. wherein we have looked for the book's ISBN from the Book-ratings data set and had return no data.

Conclusion

- The top 10 most rated books were mostly novels with The Bones and other fiction books in the list.
- Most users are that of 30-40 years of age and most have come from the USA. Still, city-wise, they most readers came from London.
- Most ratings are concentrated on '8'
- Those at the top of the list with the most books published and with ratings are thos eof Agatha Christie, William Shakespear, and Stephen King.

Conclusion

- A recommender system is vital for search engine and website users.
- This allows them to connect and make them feel included in terms of using these websites.
- It makes websites be user-friendly as well due to the personal touch that its algorithm provides.

Challenges encountered

- Insufficient knowledge in relation to programming software.
- MS Excel
  - Book-Ratings data set
- MS SQL server
  - creating syntax
  - changing data
  - Disconnect form the data base
  - Unfamiliarity

Challenges encountered

- Phyton
  - Unfamiliarity
  - Outsourced source code also has errors

References

Garg A., Tyagi A., Chowhan, A. 2021. Building a Book Recommendation System. https://github.com/ashima96/Book-Recommendation-System

Li, S., 2017. How did we build book recommender systems in an hour part 2--k Nearest Beighbors and Matrix Factorization. https://towardsdatascience.com/how-did-we-build-book-recommender-systems-in-an-hour-part-2-k-nearest-neighbors-and-matrix-c04b3c2ef55c#:~:text=kNN%20is%20a%20machine%20learning,of%20top%2Dk%20nearest%20neighbors.