Hussain_Motiwala_Assignment22

October 14, 2024

[1]:
```python
import numpy as np
import pandas as pd
import re
import chardet
from datetime import datetime
from sklearn.preprocessing import StandardScaler
from sklearn.metrics.pairwise import cosine_similarity
import requests
from IPython.display import display, HTML, Image
from io import BytesIO
from PIL import Image
import matplotlib.pyplot as plt
import base64
```

[2]:
```python
# Read the first few bytes of the file to detect encoding
with open('books-1.csv', 'rb') as file:
    raw_data = file.read(10000)  # Read first 10,000 bytes or so
    result = chardet.detect(raw_data)
    print(result)
```

```
{'encoding': 'ISO-8859-1', 'confidence': 0.73, 'language': ''}
```

[3]:
```python
# Regex pattern: match semicolons not preceded by '&amp'
pattern = r'\";\"'



# Custom function to handle bad lines
def log_bad_lines(bad_line):
    print(f"Bad line: {bad_line}")
    return None  # Return None to skip the line

df_books = pd.read_csv('books-1.csv', delimiter=pattern, engine='python',␣
 ↪encoding='ISO-8859-1', on_bad_lines=log_bad_lines, skipinitialspace=True)
```

[4]:
```python
len(df_books)
```

[4]:
```
271379
```

```
[5]: df_books.columns = df_books.columns.str.lstrip('"')
     df_books['ISBN'] = df_books['ISBN'].str.lstrip('"')
```

```
[6]: df_books.head()
```

```
[6]:          ISBN                                         Book-Title  \
     0  0195153448                                Classical Mythology
     1  0002005018                                       Clara Callan
     2  0060973129                               Decision in Normandy
     3  0374157065  Flu: The Story of the Great Influenza Pandemic...
     4  0393045218                             The Mummies of Urumchi

                 Book-Author  Year-Of-Publication                   Publisher  \
     0    Mark P. O. Morford                 2002     Oxford University Press
     1  Richard Bruce Wright                 2001       HarperFlamingo Canada
     2          Carlo D'Este                 1991             HarperPerennial
     3      Gina Bari Kolata                 1999        Farrar Straus Giroux
     4        E. J. W. Barber                1999  W. W. Norton &amp; Company

                                              Image-URL-S  \
     0  http://images.amazon.com/images/P/0195153448.0...
     1  http://images.amazon.com/images/P/0002005018.0...
     2  http://images.amazon.com/images/P/0060973129.0...
     3  http://images.amazon.com/images/P/0374157065.0...
     4  http://images.amazon.com/images/P/0393045218.0...

                                              Image-URL-M  \
     0  http://images.amazon.com/images/P/0195153448.0...
     1  http://images.amazon.com/images/P/0002005018.0...
     2  http://images.amazon.com/images/P/0060973129.0...
     3  http://images.amazon.com/images/P/0374157065.0...
     4  http://images.amazon.com/images/P/0393045218.0...

                                             Image-URL-L"
     0  http://images.amazon.com/images/P/0195153448.0...
     1  http://images.amazon.com/images/P/0002005018.0...
     2  http://images.amazon.com/images/P/0060973129.0...
     3  http://images.amazon.com/images/P/0374157065.0...
     4  http://images.amazon.com/images/P/0393045218.0...
```

```
[7]: df_books.tail()
```

```
[7]:               ISBN                                         Book-Title  \
     271374  0440400988                         There's a Bat in Bunk Five
     271375  0525447644                            From One to One Hundred
     271376  006008667X  Lily Dale : The True Story of the Town that Ta...
     271377  0192126040                          Republic (World's Classics)
```

```
271378  0767409752  A Guided Tour of Rene Descartes' Meditations o...

               Book-Author  Year-Of-Publication  \
271374      Paula Danziger                 1988
271375          Teri Sloat                 1991
271376    Christine Wicker                 2004
271377               Plato                 1996
271378  Christopher  Biffle                 2000

                                           Publisher  \
271374                  Random House Childrens Pub (Mm)
271375                                     Dutton Books
271376                                HarperSanFrancisco
271377                           Oxford University Press
271378  McGraw-Hill Humanities/Social Sciences/Languages

                                             Image-URL-S  \
271374  http://images.amazon.com/images/P/0440400988.0...
271375  http://images.amazon.com/images/P/0525447644.0...
271376  http://images.amazon.com/images/P/006008667X.0...
271377  http://images.amazon.com/images/P/0192126040.0...
271378  http://images.amazon.com/images/P/0767409752.0...

                                             Image-URL-M  \
271374  http://images.amazon.com/images/P/0440400988.0...
271375  http://images.amazon.com/images/P/0525447644.0...
271376  http://images.amazon.com/images/P/006008667X.0...
271377  http://images.amazon.com/images/P/0192126040.0...
271378  http://images.amazon.com/images/P/0767409752.0...

                                            Image-URL-L"
271374  http://images.amazon.com/images/P/0440400988.0...
271375  http://images.amazon.com/images/P/0525447644.0...
271376  http://images.amazon.com/images/P/006008667X.0...
271377  http://images.amazon.com/images/P/0192126040.0...
271378  http://images.amazon.com/images/P/0767409752.0...
```

[8]:
```
# ISBN and Image-URL-S and Image_URL-L are not signinficant for further analysis
df_books.drop(['Image-URL-S','Image-URL-L"'], axis=1, inplace=True)
```

[9]:
```
# Check if there are any duplicate values in dataset
print(sum(df_books.duplicated()))
```

```
0
```

[10]:
```
print(df_books.info())
```

```
<class 'pandas.core.frame.DataFrame'>
```

```
RangeIndex: 271379 entries, 0 to 271378
Data columns (total 6 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  271379 non-null  int64 
 4   Publisher            271377 non-null  object
 5   Image-URL-M          271379 non-null  object
dtypes: int64(1), object(5)
memory usage: 12.4+ MB
None
```

As the year of publication is already an int there is no chance "DK Publishing Inc." hence no need to check it and as year of publication is already an integer no need to convert it back to integer

[11]:
```
na_counts = pd.DataFrame(df_books.isna().sum(),columns=["NA Counts"]).
 ↪reset_index()
na_counts = na_counts.rename(columns={'index': 'Column Name'})
print(na_counts)
```

```
           Column Name  NA Counts
0                 ISBN          0
1           Book-Title          0
2          Book-Author          1
3  Year-Of-Publication          0
4            Publisher          2
5          Image-URL-M          0
```

[12]:
```
# Dropping the fields which have NA count
df_books.dropna(inplace = True)
```

[13]:
```
len(df_books)
```

[13]:
```
271376
```

[14]:
```
na_counts = pd.DataFrame(df_books.isna().sum(),columns=["NA Counts"]).
 ↪reset_index()
na_counts = na_counts.rename(columns={'index': 'Column Name'})
print(na_counts)
```

```
           Column Name  NA Counts
0                 ISBN          0
1           Book-Title          0
2          Book-Author          0
3  Year-Of-Publication          0
4            Publisher          0
5          Image-URL-M          0
```

[15]:
```
# Printing unique values in datasets columms
for column in ["Book-Author","Year-Of-Publication","Publisher"]:
    print(f"{column}:{df_books[column].unique()}\n")
```

```
Book-Author:['Mark P. O. Morford' 'Richard Bruce Wright' "Carlo D'Este" ...
 'David Biggs' 'Teri Sloat' 'Christopher  Biffle']

Year-Of-Publication:[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 2012 2006 1909 2008 1378 1919 1922
 1897 2024 1376 2037]

Publisher:['Oxford University Press' 'HarperFlamingo Canada' 'HarperPerennial'
...
 'Tempo' 'Life Works Books' 'Connaught']
```

[16]:
```
# Finding the books with publication year 0 and before 1900
df_books[df_books["Year-Of-Publication"] == 0]
```

[16]:
```
              ISBN                                         Book-Title  \
176     3150000335                                  Kabale Und Liebe
188     342311360X                            Die Liebe in Den Zelten
288     0571197639                         Poisonwood Bible Edition Uk
351     3596214629       Herr Der Fliegen (Fiction, Poetry and Drama)
542     8845229041  Biblioteca Universale Rizzoli: Sulla Sponda De...
...            ...                                                ...
270813  014029953X                          Foe (Essential.penguin S.)
270932  0340571187                                      Postmens House
271113  8427201079                           El Misterio De Sittaford
271201  0887781721                                           Tom Penny
271215  3150013763                                      Der Hofmeister

                  Book-Author  Year-Of-Publication  \
176                  Schiller                    0
188     Gabriel Garcia Marquez                    0
288         Barbara Kingsolver                    0
351                   Golding                    0
542                  P Coelho                    0
...                      ...                  ...
270813          J.M. Coetzee                    0
270932       Maggie Hemingway                    0
```

```
271113        Agatha Christie                   0
271201            Tony German                   0
271215             Jakob Lenz                   0

                                 Publisher  \
176            Philipp Reclam, Jun Verlag GmbH
188         Deutscher Taschenbuch Verlag (DTV)
288                            Faber Faber Inc
351            Fischer Taschenbuch Verlag GmbH
542                         Fabbri - RCS Libri
...                                        ...
270813                       Penguin Books Ltd
270932                        Trafalgar Square
271113                        Editorial Molino
271201                    P. Martin Associates
271215         Philipp Reclam, Jun Verlag GmbH

                                              Image-URL-M
176     http://images.amazon.com/images/P/3150000335.0...
188     http://images.amazon.com/images/P/342311360X.0...
288     http://images.amazon.com/images/P/0571197639.0...
351     http://images.amazon.com/images/P/3596214629.0...
542     http://images.amazon.com/images/P/8845229041.0...
...                                                   ...
270813  http://images.amazon.com/images/P/014029953X.0...
270932  http://images.amazon.com/images/P/0340571187.0...
271113  http://images.amazon.com/images/P/8427201079.0...
271201  http://images.amazon.com/images/P/0887781721.0...
271215  http://images.amazon.com/images/P/3150013763.0...

[4619 rows x 6 columns]
```

[17]:
```python
# Finding the books with publication year > current year
current_year = datetime.now().year
df_books[df_books["Year-Of-Publication"]>current_year]
```

[17]:
```
              ISBN                                         Book-Title  \
37488   0671746103  MY TEACHER FRIED MY BRAINS (RACK SIZE) (MY TEA...
55679   0671791990  MY TEACHER FLUNKED THE PLANET (RACK SIZE) (MY ...
78171   0870449842                                   Crossing America
80267   0140301690  Alice's Adventures in Wonderland and Through t...
97830   0140201092      Outline of European Architecture (Pelican S.)
116058  0394701658                       Three Plays of Eugene Oneill
118299  3442436893         Das groÃ?Â?e BÃ?Â¶se- MÃ?Â¤dchen- Lesebuch.
193004  0870446924  Field Guide to the Birds of North America, 3rd...
228187  0671266500       FOREST PEOPLE (Touchstone Books (Hardcover))
240184  0684718022            In Our Time: Stories (Scribner Classic)
```

```
255426  068471809X                               To Have and Have Not
260992  0671740989        FOOTBALL SUPER TEAMS : FOOTBALL SUPER TEAMS

                    Book-Author  Year-Of-Publication  \
37488                   Coville                 2030
55679             Bruce Coville                 2030
78171   National Geographic Society             2030
80267             Lewis Carroll                 2050
97830          Nikolaus Pevsner                 2050
116058           Eugene O'Neill                 2038
118299              Kathy Lette                 2026
193004  National Geographic Society             2030
228187         Colin M. Turnbull                2030
240184         Ernest Hemingway                 2030
255426         Ernest Hemingway                 2037
260992              Bill Gutman                 2030

                                      Publisher  \
37488                                   Aladdin
55679                                   Aladdin
78171                       National Geographic
80267                              Puffin Books
97830                               Penguin USA
116058                        Vintage Books USA
118299                                 Goldmann
193004                      National Geographic
228187                     Simon &amp; Schuster
240184                            Collier Books
255426                     Simon &amp; Schuster
260992  Simon &amp; Schuster Children's Publishing

                                             Image-URL-M
37488   http://images.amazon.com/images/P/0671746103.0...
55679   http://images.amazon.com/images/P/0671791990.0...
78171   http://images.amazon.com/images/P/0870449842.0...
80267   http://images.amazon.com/images/P/0140301690.0...
97830   http://images.amazon.com/images/P/0140201092.0...
116058  http://images.amazon.com/images/P/0394701658.0...
118299  http://images.amazon.com/images/P/3442436893.0...
193004  http://images.amazon.com/images/P/0870446924.0...
228187  http://images.amazon.com/images/P/0671266500.0...
240184  http://images.amazon.com/images/P/0684718022.0...
255426  http://images.amazon.com/images/P/068471809X.0...
260992  http://images.amazon.com/images/P/0671740989.0...
```

[18]:
```
# The dataset cannot have books published after current year hence making them 0
```

```
df_books.loc[df_books["Year-Of-Publication"]>current_year,␣
↪"Year-Of-Publication"] = 0
print(len(df_books[df_books["Year-Of-Publication"] == 0]))
```

```
4631
```

[19]:
```
#Printing unique values in dataset columns
for column in ["Book-Author","Year-Of-Publication","Publisher"]:
    print(f"{df_books[column].value_counts()}\n{df_books[column].
↪value_counts(normalize=True)}\n")
```

```
Agatha Christie        632
William Shakespeare    567
Stephen King           524
Ann M. Martin          423
Francine Pascal        373
                      ...
Jean Cassels             1
Bernice Meyers           1
Mark A. Taylor           1
Ellery R. Sheets         1
Christopher  Biffle      1
Name: Book-Author, Length: 102026, dtype: int64
Agatha Christie        0.002329
William Shakespeare    0.002089
Stephen King           0.001931
Ann M. Martin          0.001559
Francine Pascal        0.001374
                         ...
Jean Cassels           0.000004
Bernice Meyers         0.000004
Mark A. Taylor         0.000004
Ellery R. Sheets       0.000004
Christopher  Biffle    0.000004
Name: Book-Author, Length: 102026, dtype: float64

2002    17627
1999    17432
2001    17359
2000    17235
1998    15767
        ...
1917        1
1910        1
1914        1
1904        1
1376        1
Name: Year-Of-Publication, Length: 111, dtype: int64
```

```
2002    0.064954
1999    0.064236
2001    0.063967
2000    0.063510
1998    0.058100
          ...
1917    0.000004
1910    0.000004
1914    0.000004
1904    0.000004
1376    0.000004
Name: Year-Of-Publication, Length: 111, dtype: float64

Harlequin                  7536
Silhouette                 4220
Pocket                     3905
Ballantine Books           3783
Bantam Books               3647
                           ...
Polaris Books                 1
Hannover House                1
Amber Quill Press, LLC.       1
Lunchbox Press                1
Connaught                     1
Name: Publisher, Length: 16805, dtype: int64
Harlequin                  0.027770
Silhouette                 0.015550
Pocket                     0.014390
Ballantine Books           0.013940
Bantam Books               0.013439
                             ...
Polaris Books              0.000004
Hannover House             0.000004
Amber Quill Press, LLC.    0.000004
Lunchbox Press             0.000004
Connaught                  0.000004
Name: Publisher, Length: 16805, dtype: float64
```

Ratings

[20]:
```python
# Regex pattern: match semicolons not preceded by '&amp'
pattern = r'\";\"'


# Custom function to handle bad lines
def log_bad_lines(bad_line):
    print(f"Bad line: {bad_line}")
    return None  # Return None to skip the line

df_ratings = pd.read_csv('ratings-1.csv', delimiter=pattern, engine='python',␣
 ↪encoding='ISO-8859-1', on_bad_lines=log_bad_lines, skipinitialspace=True)
```

[21]:
```
len(df_ratings)
```

[21]:
```
1149780
```

[22]:
```
df_ratings.columns = df_ratings.columns.str.lstrip('"')
df_ratings.columns = df_ratings.columns.str.rstrip('"')
df_ratings['User-ID'] = df_ratings['User-ID'].str.lstrip('"')
df_ratings['Book-Rating'] = df_ratings['Book-Rating'].str.rstrip('"')
```

[23]:
```
df_ratings.head()
```

[23]:
```
  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
```

[24]:
```
df_ratings.tail()
```

[24]:
```
        User-ID         ISBN Book-Rating
1149775  276704   1563526298           9
1149776  276706   0679447156           0
1149777  276709   0515107662          10
1149778  276721   0590442449          10
1149779  276723  05162443314           8
```

[25]:
```
# Check if there are any duplicate values in dataset
print(sum(df_ratings.duplicated()))
```

```
0
```

[26]:
```
print(df_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  object
 1   ISBN         1149780 non-null  object
 2   Book-Rating  1149780 non-null  object
dtypes: object(3)
```

```
memory usage: 26.3+ MB
None
```

[27]:
```
# There are no nulls but additional check
na_counts = pd.DataFrame(df_ratings.isna().sum(),columns=["NA Counts"]).
 ↪reset_index()
na_counts = na_counts.rename(columns={'index': 'Column Name'})
print(na_counts)
```

```
   Column Name  NA Counts
0      User-ID          0
1         ISBN          0
2  Book-Rating          0
```

[28]:
```
# Printing unique values in datasets columns
for column in df_ratings.columns:
    print(f"{column}:{df_ratings[column].unique()}\n")
```

```
User-ID:['276725' '276726' '276727' ... '276709' '276721' '276723']

ISBN:['034545104X' '0155061224' '0446520802' ... '0679752714' '0806917695'
 '05162443314']

Book-Rating:['0' '5' '3' '6' '8' '7' '10' '9' '4' '1' '2']
```

[29]:
```
#Printing unique values in dataset columns
for column in df_ratings.columns:
    print(f"{df_ratings[column].value_counts()}\n{df_ratings[column].
 ↪value_counts(normalize=True)}\n\
mean is {df_ratings[column].value_counts().mean()} \n\
median is {df_ratings[column].value_counts().median()} \n")
```

```
11676     13602
198711     7550
153662     6109
98391      5891
35859      5850
          ...
116180        1
116166        1
116154        1
116137        1
276723        1
Name: User-ID, Length: 105283, dtype: int64
11676     1.183009e-02
198711    6.566474e-03
153662    5.313190e-03
98391     5.123589e-03
```

```
35859     5.087930e-03
              ...
116180    8.697316e-07
116166    8.697316e-07
116154    8.697316e-07
116137    8.697316e-07
276723    8.697316e-07
Name: User-ID, Length: 105283, dtype: float64
mean is 10.920851419507423
median is 1.0

0971880107     2502
0316666343     1295
0385504209      883
0060928336      732
0312195516      723
               ...
1568656386        1
1568656408        1
1569551553        1
1570081808        1
05162443314       1
Name: ISBN, Length: 340556, dtype: int64
0971880107     2.176068e-03
0316666343     1.126302e-03
0385504209     7.679730e-04
0060928336     6.366435e-04
0312195516     6.288159e-04
                   ...
1568656386     8.697316e-07
1568656408     8.697316e-07
1569551553     8.697316e-07
1570081808     8.697316e-07
05162443314    8.697316e-07
Name: ISBN, Length: 340556, dtype: float64
mean is 3.376184827164989
median is 1.0

0     716109
8     103736
10     78610
7      76457
9      67541
5      50974
6      36924
4       8904
3       5996
2       2759
```

```
1       1770
Name: Book-Rating, dtype: int64
0     0.622823
8     0.090222
10    0.068370
7     0.066497
9     0.058743
5     0.044334
6     0.032114
4     0.007744
3     0.005215
2     0.002400
1     0.001539
Name: Book-Rating, dtype: float64
mean is 104525.45454545454
median is 50974.0
```

[30]:
```
# Convert 'age' column to int type
df_ratings['Book-Rating'] = df_ratings['Book-Rating'].astype(int)
df_ratings.describe()
```

[30]:
```
        Book-Rating
count  1.149780e+06
mean   2.866950e+00
std    3.854184e+00
min    0.000000e+00
25%    0.000000e+00
50%    0.000000e+00
75%    7.000000e+00
max    1.000000e+01
```

Users

[31]:
```
# Regex pattern: match semicolons not preceded by '&amp'
pattern = r'\";\"|\";NULL'


# Custom function to handle bad lines
def log_bad_lines(bad_line):
    print(f"Bad line: {bad_line}")
    return None  # Return None to skip the line

df_users = pd.read_csv('users-1.csv', delimiter=pattern, engine='python',␣
 ↪encoding='ISO-8859-1', on_bad_lines=log_bad_lines, skipinitialspace=True)
```

[32]:
```
len(df_users)
```

[32]: 278859

[33]:
```
df_users.columns = df_users.columns.str.lstrip('"')
df_users.columns = df_users.columns.str.rstrip('"')
df_users['User-ID'] = df_users['User-ID'].str.lstrip('"')
df_users['Age'] = df_users['Age'].str.rstrip('"')
```

[34]:
```
df_users.head()
```

[34]:
```
  User-ID                            Location  Age
0       1                  nyc, new york, usa  NaN
1       2           stockton, california, usa   18
2       3     moscow, yukon territory, russia  NaN
3       4           porto, v.n.gaia, portugal   17
4       5  farnborough, hants, united kingdom  NaN
```

[35]:
```
df_users.tail()
```

[35]:
```
       User-ID                            Location  Age
278854  278854                portland, oregon, usa  NaN
278855  278855  tacoma, washington, united kingdom   50
278856  278856            brampton, ontario, canada  NaN
278857  278857            knoxville, tennessee, usa  NaN
278858  278858                  dublin, n/a, ireland  NaN
```

[36]:
```
# Check if there are any duplicate values in dataset
print(sum(df_users.duplicated()))
```

```
0
```

[37]:
```
print(df_users.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  object
dtypes: object(3)
memory usage: 6.4+ MB
None
```

[38]:
```
# Get unique values, excluding None and NaN, and sort them
unique_ages = df_users['Age'].unique()
filtered_ages = [int(age) for age in unique_ages if age is not None and pd.
 ↪notna(age)]
sorted_unique_ages = sorted(filtered_ages)
```

```
# Print the sorted unique age values
print(sorted_unique_ages)
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21,
22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41,
42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61,
62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81,
82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100,
101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 113, 114, 115, 116, 118,
119, 123, 124, 127, 128, 132, 133, 136, 137, 138, 140, 141, 143, 146, 147, 148,
151, 152, 156, 157, 159, 162, 168, 172, 175, 183, 186, 189, 199, 200, 201, 204,
207, 208, 209, 210, 212, 219, 220, 223, 226, 228, 229, 230, 231, 237, 239, 244]
```

[39]:
```
#Making all ages above 90 and below five to nan
# Replace values less than 5 or greater than 90 with NaN, keeping column as␣
↪object type
df_users['Age'] = df_users['Age'].apply(lambda x: np.nan if (x is None or pd.
↪isna(x) or (isinstance(x, str) and x.isnumeric() and (int(x) < 5 or int(x) >␣
↪90)))  else x)
```

[40]:
```
# Printing unique ages
print(df_users['Age'].unique())
```

```
[nan '18' '17' '61' '26' '14' '25' '19' '46' '55' '32' '24' '20' '34' '23'
 '51' '31' '21' '44' '30' '57' '43' '37' '41' '54' '42' '50' '39' '53'
 '47' '36' '28' '35' '13' '58' '49' '38' '45' '62' '63' '27' '33' '29'
 '66' '40' '15' '60' '79' '22' '16' '65' '59' '48' '72' '56' '67' '80'
 '52' '69' '71' '73' '78' '9' '64' '12' '74' '75' '76' '83' '68' '11' '77'
 '70' '8' '7' '81' '10' '5' '6' '84' '82' '90' '85' '86' '87' '89' '88']
```

[41]:
```
# Replacing all nan with mean age
# Replace values less than 5 or greater than 90 with NaN, keeping column as␣
↪object type
df_users['Age'] = df_users['Age'].apply(lambda x: np.nan if (x is None or x in␣
↪['NaN', ''] or (isinstance(x, str) and x.isnumeric() and (int(x) < 5 or int(x)␣
↪> 90))) else x)

# Convert Age to numeric to calculate mean, while coercing errors to NaN
df_users['Age'] = pd.to_numeric(df_users['Age'], errors='coerce')

# Calculate mean age, ignoring NaN values
mean_age = df_users['Age'].mean()

# Replace NaN values with the mean age
df_users['Age'].fillna(mean_age, inplace=True)

print(f"The mean age is {mean_age}")
```

```
The mean age is 34.72384041634689
```

[42]:
```
# Converting age to type int
df_users["Age"] = df_users["Age"].astype(int)
```

[43]:
```
# Printing unique ages
print(df_users['Age'].unique())
```

```
[34 18 17 61 26 14 25 19 46 55 32 24 20 23 51 31 21 44 30 57 43 37 41 54
 42 50 39 53 47 36 28 35 13 58 49 38 45 62 63 27 33 29 66 40 15 60 79 22
 16 65 59 48 72 56 67 80 52 69 71 73 78  9 64 12 74 75 76 83 68 11 77 70
  8  7 81 10  5  6 84 82 90 85 86 87 89 88]
```

Recommender System

We are making a colloborative filter specifically for The Queen of the Damned (Vampire Chronicles (Paperback)). We will only consider users who have rated the book The Queen of the Damned (Vampire Chronicles (Paperback))

[44]:
```
# Finding the ISBN number for The Queen of the Damned (Vampire Chronicles␣
↪(Paperback))
df_books[df_books["Book-Title"]=='The Queen of the Damned (Vampire Chronicles␣
↪(Paperback))']
```

[44]:
```
             ISBN                                         Book-Title  \
2527   0345351525  The Queen of the Damned (Vampire Chronicles (P...
72840  0833563505  The Queen of the Damned (Vampire Chronicles (P...

      Book-Author  Year-Of-Publication         Publisher  \
2527    Anne Rice                 1993  Ballantine Books
72840   Anne Rice                 1999    Sagebrush Bound

                                             Image-URL-M
2527   http://images.amazon.com/images/P/0345351525.0...
72840  http://images.amazon.com/images/P/0833563505.0...
```

[45]:
```
# We find we have two editions of The Queen of the Damned (Vampire Chronicles␣
↪(Paperback))
# we need to apply inner join between ratings df and books df to get the books␣
↪rating df
books_rating_df = pd.merge(df_books[["ISBN","Book-Title"]], df_ratings,␣
↪on='ISBN', how='inner')
```

[46]:
```
len(books_rating_df)
```

[46]:
```
1031172
```

[47]:
```
books_rating_df.head()
```

[47]:
```
         ISBN          Book-Title User-ID  Book-Rating
0  0195153448  Classical Mythology       2            0
1  0002005018         Clara Callan       8            5
2  0002005018         Clara Callan   11400            0
3  0002005018         Clara Callan   11676            8
4  0002005018         Clara Callan   41385            0
```

[48]:
```
books_rating_df.tail()
```

[48]:
```
               ISBN                                         Book-Title  \
1031167  0440400988                         There's a Bat in Bunk Five
1031168  0525447644                            From One to One Hundred
1031169  006008667X  Lily Dale : The True Story of the Town that Ta...
1031170  0192126040                        Republic (World's Classics)
1031171  0767409752  A Guided Tour of Rene Descartes' Meditations o...

        User-ID  Book-Rating
1031167  276463            7
1031168  276579            4
1031169  276680            0
1031170  276680            0
1031171  276680            0
```

[49]:
```
# Dropping the ISBN column
books_rating_df.drop("ISBN", axis=1, inplace=True)
```

[50]:
```
# Aggregrating the ratings
books_rating_df = books_rating_df.groupby(['Book-Title', 'User-ID']).agg({
    'Book-Rating': lambda x: np.floor(x.mean())
}).reset_index()
```

[51]:
```
len(books_rating_df)
```

[51]:
```
1026394
```

[52]:
```
# Step 1: Find all duplicates based on 'User-ID', 'Book_Title', and 'Book-Author'
duplicates = books_rating_df[books_rating_df.duplicated(subset=['User-ID',␣
 ↪'Book-Title'], keep=False)]

# Step 2: Sort by 'User-ID'
duplicates_sorted = duplicates.sort_values(by='User-ID')

# Step 3: Display the sorted duplicates
print(duplicates_sorted)
```

```
Empty DataFrame
Columns: [Book-Title, User-ID, Book-Rating]
Index: []
```

[53]:
```
# Step 1: Sort df_books by 'Book-Title', 'Book-Author', and␣
↪'Year-Of-Publication' in descending order
df_books_sorted = df_books.sort_values(['Book-Title', 'Year-Of-Publication'],␣
↪ascending=[True, False])

# Step 2: Drop duplicates, keeping only the entry with the latest␣
↪'Year-Of-Publication'
df_books_sorted = df_books_sorted.drop_duplicates(subset=['Book-Title'],␣
↪keep='first')

# Step 3: Merge the two DataFrames on 'Book-Title' and 'Book-Author'
books_rating_df = pd.merge(books_rating_df, df_books_sorted, on=['Book-Title'],␣
↪how='left')
```

[54]:
```
len(books_rating_df)
```

[54]:
```
1026394
```

[55]:
```
books_rating_df =␣
↪books_rating_df[["ISBN","Book-Title","Book-Author","Publisher","Year-Of-Publication","Image-U
```

[56]:
```
books_rating_df.head()
```

[56]:
```
         ISBN                                         Book-Title  \
0  0590567330  A Light in the Storm: The Civil War Diary of ...
1  0590567330  A Light in the Storm: The Civil War Diary of ...
2  0590567330  A Light in the Storm: The Civil War Diary of ...
3  0590567330  A Light in the Storm: The Civil War Diary of ...
4  0964147726                             Always Have Popsicles

     Book-Author                   Publisher  Year-Of-Publication  \
0    Karen Hesse  Hyperion Books for Children                 1999
1    Karen Hesse  Hyperion Books for Children                 1999
2    Karen Hesse  Hyperion Books for Children                 1999
3    Karen Hesse  Hyperion Books for Children                 1999
4  Rebecca Harvin            Rebecca L. Harvin                 1994

                                         Image-URL-M User-ID  Book-Rating
0  http://images.amazon.com/images/P/0590567330.0...   18995          0.0
1  http://images.amazon.com/images/P/0590567330.0...   35859          0.0
2  http://images.amazon.com/images/P/0590567330.0...   55927          0.0
3  http://images.amazon.com/images/P/0590567330.0...   96448          9.0
4  http://images.amazon.com/images/P/0964147726.0...  172742          0.0
```

[57]:
```
books_rating_df.tail()
```

[57]:
```
                   ISBN            Book-Title       Book-Author Publisher  \
1026389  3442725739  Ã?Â?stlich der Berge.    David Guterson       btb
1026390  3442725739  Ã?Â?stlich der Berge.    David Guterson       btb
1026391  3442725739  Ã?Â?stlich der Berge.    David Guterson       btb
1026392  2842192508      Ã?Â?thique en toc  Didier Daeninckx   Baleine
1026393  2842192508      Ã?Â?thique en toc  Didier Daeninckx   Baleine

         Year-Of-Publication  \
1026389                 2000
1026390                 2000
1026391                 2000
1026392                 1998
1026393                 1998

                                              Image-URL-M User-ID  \
1026389  http://images.amazon.com/images/P/3442725739.0...  243548
1026390  http://images.amazon.com/images/P/3442725739.0...  261105
1026391  http://images.amazon.com/images/P/3442725739.0...   90839
1026392  http://images.amazon.com/images/P/2842192508.0...   25436
1026393  http://images.amazon.com/images/P/2842192508.0...   53628

         Book-Rating
1026389          0.0
1026390          0.0
1026391          8.0
1026392          8.0
1026393          0.0
```

[58]:
```
# Now filtering only those ratings done by users who have rated "The Queen of␣
↪the Damned (Vampire Chronicles (Paperback))"

# Step 1: Get User-IDs of users who rated "The Queen of the Damned (Vampire␣
↪Chronicles (Paperback))"
queen_raters = books_rating_df[books_rating_df['Book-Title'] == 'The Queen of␣
↪the Damned (Vampire Chronicles (Paperback))']['User-ID'].unique()

# Step 2: Filter books_ratings_df to include only ratings from these users
filtered_df = books_rating_df[books_rating_df['User-ID'].isin(queen_raters)]
```

[59]:
```
len(filtered_df)
```

[59]:
```
94524
```

[60]:
```
len(filtered_df["User-ID"].unique())
```

[60]:
```
274
```

[61]:
```
# Now making user-item-rating matrix out of filtered df for recomender system
user_item_rating_matrix = filtered_df.pivot_table(index = 'Book-Title', columns␣
↪= 'User-ID', values = 'Book-Rating')

# Filling the NA values with '0'
user_item_rating_matrix.fillna(0, inplace = True)
```

[62]:
```
user_item_rating_matrix.shape
```

[62]:
```
(52821, 274)
```

[63]:
```
# Scaling the matrix
scaler = StandardScaler(with_mean=True, with_std=True)
user_item_rating_matrix_normalized = scaler.
↪fit_transform(user_item_rating_matrix)
```

[64]:
```
similarity_score = cosine_similarity(user_item_rating_matrix_normalized)
```

[65]:
```
similarity_score.shape
```

[65]:
```
(52821, 52821)
```

[66]:
```
# Convert to DataFrame and set indexes and columns to book names
similarity_df = pd.DataFrame(similarity_score,
                              index=user_item_rating_matrix.index,
                              columns=user_item_rating_matrix.index)
```

[67]:
```
# Defining the function to find the top_n recomendation for a particular book

def recommend_top_n_books(similarity_df, book_title, n):
    """
    Recommend top n books based on the highest similarity scores for a given␣
↪book title,
    including similarity scores and rankings.

    Parameters:
    - similarity_df: DataFrame containing similarity scores between books
    - book_title: Title of the book to base recommendations on
    - n: Number of top recommendations to return

    Returns:
    - DataFrame containing recommended book titles, similarity scores, and␣
↪rankings
    """
    # Check if the book title exists in the similarity DataFrame
    if book_title not in similarity_df.index:
        return f"The book '{book_title}' is not in the similarity DataFrame."
    # Step 1: Get the similarity scores for the given book title
    similarity_scores = similarity_df.loc[book_title]

    # Step 2: Sort the scores in descending order and get the top n
    top_n_books = similarity_scores.nlargest(n + 1)  # +1 to exclude the book␣
↪itself

    # Step 3: Create a DataFrame with book titles, similarity scores, and␣
↪rankings
    recommendations_df = pd.DataFrame({
        'Book-Title': top_n_books.index,
        'Similarity Score': top_n_books.values,
    })

    # Add a ranking column
    recommendations_df['Ranking'] = range(0, len(recommendations_df))

    # Return the DataFrame containing recommendations
    return recommendations_df
```

[68]:
```
recommendations_df = recommend_top_n_books(similarity_df, 'The Queen of the␣
↪Damned (Vampire Chronicles (Paperback))', 10)
```

[69]:
```
recommendations_df
```

[69]:
```
                                           Book-Title  Similarity Score  \
0   The Queen of the Damned (Vampire Chronicles (P...          1.000000
1    The Vampire Lestat (Vampire Chronicles, Book II)          0.252394
2   Pandora: New Tales of the Vampires (New Tales ...          0.239926
3   The Tale of the Body Thief (Vampire Chronicles...          0.237966
4                           Interview with the Vampire         0.234949
5                               The Celestine Prophecy         0.229347
6                           The Redemption of Althalus         0.217616
7                        The Stand: Complete and Uncut         0.193729
8        Memnoch the Devil (Vampire Chronicles, No 5)          0.181737
9                        Jennifer Government : A Novel         0.169625
10  Don't Sweat the Small Stuff and It's All Small...          0.166977

    Ranking
0         0
1         1
2         2
3         3
4         4
5         5
6         6
```

```
7         7
8         8
9         9
10       10
```

[70]:
```
# Getting additional details for recomendation_df
filtered_df_undupl = filtered_df.drop(["User-ID","Book-Rating"], axis=1).
 ↪drop_duplicates()
recommendations_detail_df = pd.merge(filtered_df_undupl, recommendations_df,␣
 ↪on='Book-Title', how='inner')
```

[71]:
```
recommendations_detail_df.sort_values("Ranking").reset_index()
```

[71]:
```
    index        ISBN                                         Book-Title  \
0       6  0833563505  The Queen of the Damned (Vampire Chronicles (P...
1      10  0345313860   The Vampire Lestat (Vampire Chronicles, Book II)
2       4  0345422384  Pandora: New Tales of the Vampires (New Tales ...
3       9  034538475X  The Tale of the Body Thief (Vampire Chronicles...
4       1  0345337662                          Interview with the Vampire
5       5  0446671002                              The Celestine Prophecy
6       7  0345440781                          The Redemption of Althalus
7       8  0451169530                        The Stand: Complete and Uncut
8       3  0345409671         Memnoch the Devil (Vampire Chronicles, No 5)
9       2  0385507593                        Jennifer Government : A Novel
10      0  0786881852  Don't Sweat the Small Stuff and It's All Small...

       Book-Author          Publisher  Year-Of-Publication  \
0        Anne Rice    Sagebrush Bound                 1999
1        ANNE RICE  Ballantine Books                  1986
2        Anne Rice  Ballantine Books                  1999
3        Anne Rice  Ballantine Books                  1993
4        Anne Rice  Ballantine Books                  1993
5   James Redfield      Warner Books                  1995
6     David Eddings     Del Rey Books                 2001
7     Stephen King       Signet Book                  1991
8        Anne Rice  Ballantine Books                  1997
9        Max Barry         Doubleday                  2003
10 Richard Carlson          Hyperion                  1997

                                         Image-URL-M  Similarity Score  \
0   http://images.amazon.com/images/P/0833563505.0...          1.000000
1   http://images.amazon.com/images/P/0345313860.0...          0.252394
2   http://images.amazon.com/images/P/0345422384.0...          0.239926
3   http://images.amazon.com/images/P/034538475X.0...          0.237966
4   http://images.amazon.com/images/P/0345337662.0...          0.234949
5   http://images.amazon.com/images/P/0446671002.0...          0.229347
6   http://images.amazon.com/images/P/0345440781.0...          0.217616
```

```
7   http://images.amazon.com/images/P/0451169530.0...          0.193729
8   http://images.amazon.com/images/P/0345409671.0...          0.181737
9   http://images.amazon.com/images/P/0385507593.0...          0.169625
10  http://images.amazon.com/images/P/0786881852.0...          0.166977

    Ranking
0         0
1         1
2         2
3         3
4         4
5         5
6         6
7         7
8         8
9         9
10       10
```

[72]:
```
# Create a new column for displaying images
recommendations_detail_df['Cover Image'] =␣
 ↪recommendations_detail_df['Image-URL-M'].apply(lambda url: f'<img src="{url}"␣
 ↪width="150" height="150"/>')
```

[73]:
```
# Display the DataFrame with images
display(HTML(recommendations_detail_df.drop("Image-URL-M",axis=1).
 ↪sort_values("Ranking").to_html(escape=False, index=False)))
```

```
<IPython.core.display.HTML object>
```