‘

Artificial Intelligence

Project 1: Book Recommendation System

Problem Statement: Recommendation of similar books to the user based on collaboration.

Objectives:

- To filter the useful data from the given data set
- To apply popularity-based filtering
- To recommend the top 50 books on page from the previous dataset
- To apply collaborative filtering and cosine similarity
- To recommend similar books the user is searching
- To make an easy user interface
- To create a user interface for a recommendation system

Languages Used: Python

Python has a standard library in development, and a few for AI. It has an intuitive syntax, basic control flow, and data structures. It also supports interpretive run-time, without standard compiler languages. This makes Python especially useful for prototyping algorithms for AI.

Dependencies:

Jupyter Notebook

Jupyter notebook is an open-source IDE that is used to create Jupyter documents that can be created and shared with live codes. Also, it is a web-based interactive computational environment. The Jupyter notebook can support various languages that are popular in data science such as Python, Scala, R, etc. Jupyter Notebook is basically a web application. Unlike IDEs (Integrated Development Environment), it uses the internet to run. And even after not being able to perform offline, it is highly preferred by most of the beginners because of its rich formatting and user-friendly interface.

PyCharm

PyCharm is a dedicated Python Integrated Development Environment (IDE) providing a wide range of essential tools for Python developers, tightly integrated to create a convenient environment for productive Python, web, and data science development. PyCharm is an Integrated Development Environment (IDE) used for programming in Python. It provides code analysis, a graphical debugger, an integrated unit tester, integration with version control systems, and supports web development with Django.

NumPy

NumPy is a library for Python that allows it to work with multidimensional arrays and matrices. It's perfect for scientific or mathematical calculations because it's fast and efficient. In addition, NumPy includes support for signal processing and linear algebra operations.to do any mathematical operations on data.

Pandas

Pandas is an open-source Python package that is most widely used for data science/data analysis and machine learning tasks. It is built on top of another package named NumPy, which provides support for multi-dimensional arrays. The Pandas module mainly works with the tabular data, whereas the NumPy module works with the numerical data. The Pandas provides some sets of powerful tools like DataFrame and Series that are mainly used for analyzing the data, whereas the NumPy module offers a powerful object called Array.

Pickle

Pickle in Python is primarily used in serializing and deserializing a Python object structure. In other words, it's the process of converting a Python object into a byte stream to store it in a file/database, maintain program state across sessions, or transport data over the network.

Sk-learn

Scikit-learn (sk-learn) is the most useful and robust library for machine learning in Python. It provides a selection of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction via a consistent interface in Python. Scikit-learn is a machine learning library for Python. It features several regression, classification and clustering algorithms including SVMs, gradient boosting, k-means. It is designed to work with Python NumPy.

Framework Used: Flask

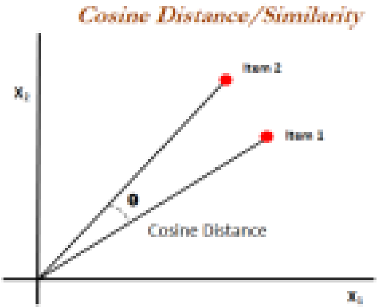
Flask is a small and lightweight Python web framework that provides useful tools and features that make creating web applications in Python easier. It gives developers flexibility and is a more accessible framework for new developers since you can build a web application quickly using only a single Python file. This means flask provides us with tools, libraries and technologies that allow us to build a web application. This web application can be some web pages, a blog, a wiki or go as big as a web-based calendar application or a commercial website.

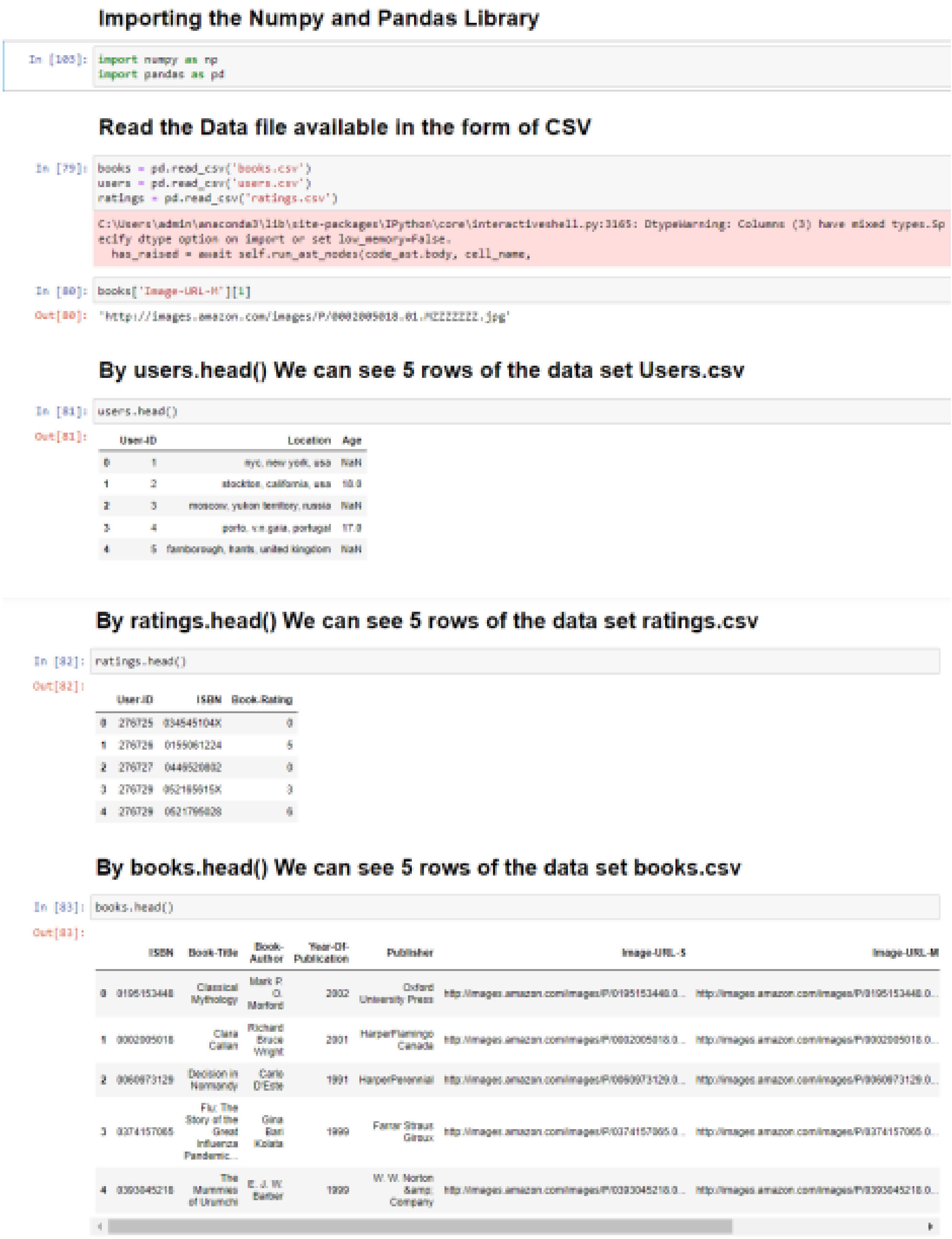
Collaborative filtering

Collaborative filtering is a technique that can filter out items that a user might like on the basis of reactions by similar users. It works by searching a large group of people and finding a smaller set of users with tastes similar to a particular user. Amazon is known for its use of collaborative filtering, matching products to users based on past purchases. For example, the system can identify all of the products a customer and users with similar behaviors have purchased and/or positively rated. To address some of the limitations of content-based filtering, collaborative filtering uses similarities between users and items simultaneously to provide recommendations.

Algorithm: Cosine similarity

In data analysis, cosine similarity is a measure of similarity between two sequences of numbers. For defining it, the sequences are viewed as vectors in an inner product space, and the cosine similarity is defined as the cosine of the angle between them, that is, the dot product of the vectors divided by the product of their lengths. It follows that the cosine similarity does not depend on the magnitudes of the vectors, but only on their angle.





Importing the Numpy and Pandas Library

In [103]:
```
import numpy as np
import pandas as pd
```

Read the Data file available in the form of CSV

In [79]:
```
books = pd.read_csv('books.csv')
users = pd.read_csv('users.csv')
ratings = pd.read_csv('ratings.csv')
```

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

In [80]:
```
books['Image-URL-M'][1]
```

Out[80]:
```
'http://images.amazon.com/images/P/0002005018.01.MZZZZZZZ.jpg'
```

By users.head() We can see 5 rows of the data set Users.csv

In [81]:
```
users.head()
```

Out[81]:

	User-ID	Location	Age
0	1	nyc, new york, usa	NaN
1	2	stockton, california, usa	18.0
2	3	moscow, yukon territory, russia	NaN
3	4	porto, v.n.gaia, portugal	17.0
4	5	farnborough, hants, united kingdom	NaN

By ratings.head() We can see 5 rows of the data set ratings.csv

In [82]:
```
ratings.head()
```

Out[82]:

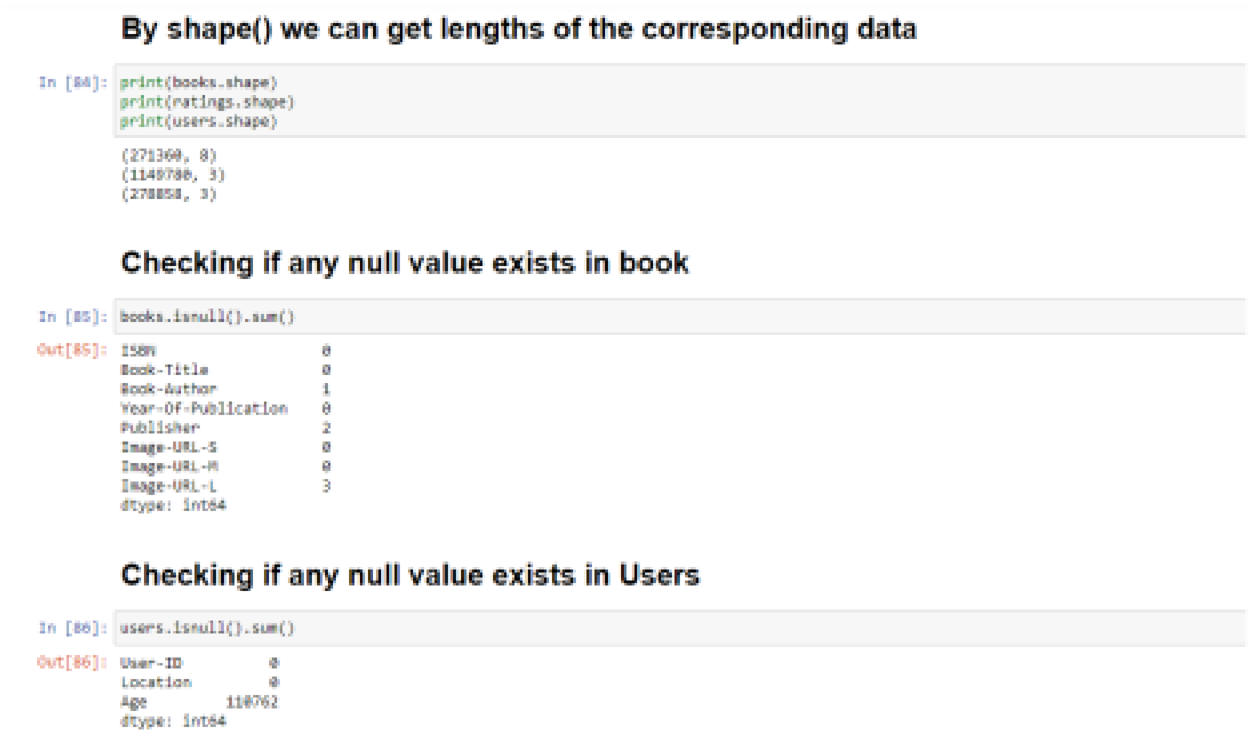
	User-ID	ISBN	Book-Rating
0	276725	034545104X	0
1	276726	0155061224	5
2	276727	0446520802	0
3	276729	052165615X	3
4	276729	0521795028	6

By books.head() We can see 5 rows of the data set books.csv

In [83]:
```
books.head()
```

Out[83]:

	ISBN	Book-Title	Book-Author	Year-Of-Publication	Publisher	Image-URL-S	Image-URL-M
0	0195153448	Classical Mythology	Mark P. O. Morford	2002	Oxford University Press	http://images.amazon.com/images/P/0195153448.0...	http://images.amazon.com/images/P/0195153448.0...
1	0002005018	Clara Callan	Richard Bruce Wright	2001	HarperFlamingo Canada	http://images.amazon.com/images/P/0002005018.0...	http://images.amazon.com/images/P/0002005018.0...
2	0060973129	Decision in Normandy	Carlo D'Este	1991	HarperPerennial	http://images.amazon.com/images/P/0060973129.0...	http://images.amazon.com/images/P/0060973129.0...
3	0374157065	Flu: The Story of the Great Influenza Pandemic...	Gina Bari Kolata	1999	Farrar Straus Giroux	http://images.amazon.com/images/P/0374157065.0...	http://images.amazon.com/images/P/0374157065.0...
4	0393045218	The Mummies of Urumchi	E. J. W. Barber	1999	W. W. Norton &amp; Company	http://images.amazon.com/images/P/0393045218.0...	http://images.amazon.com/images/P/0393045218.0...



By shape() we can get lengths of the corresponding data

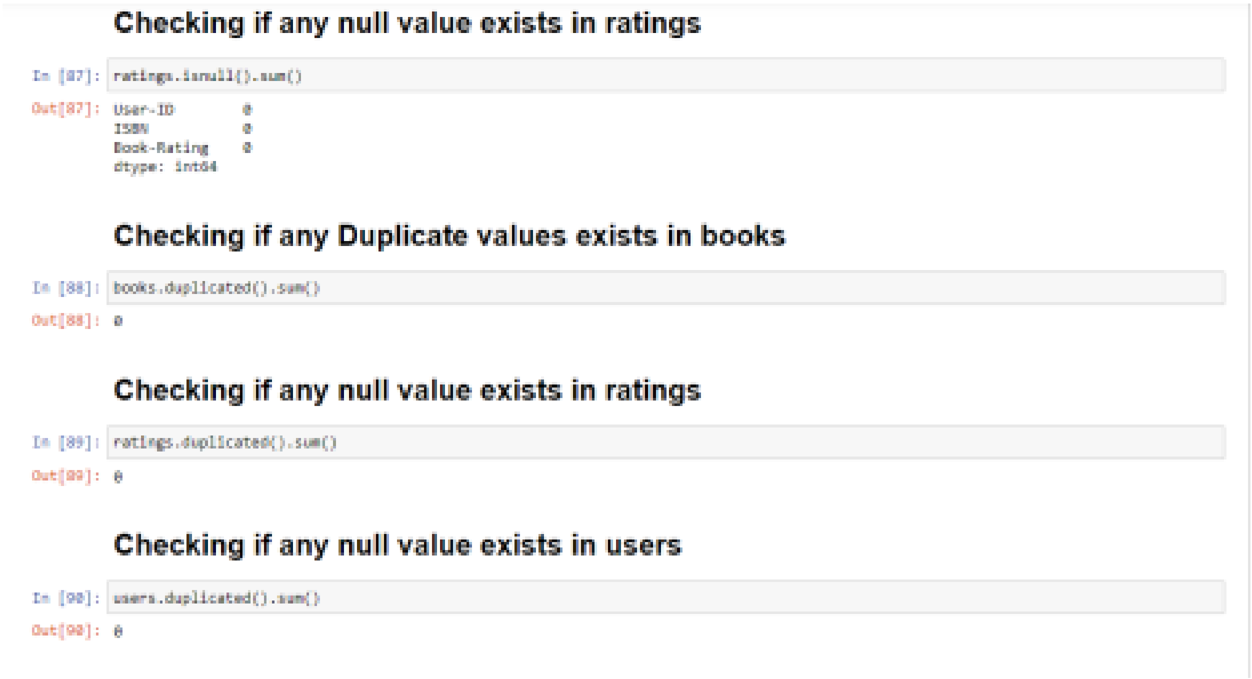
```
In [84]: print(books.shape)
         print(ratings.shape)
         print(users.shape)

         (271360, 8)
         (1149780, 3)
         (278858, 3)
```

Checking if any null value exists in book

```
In [85]: books.isnull().sum()

Out[85]: ISBN                   0
         Book-Title             0
         Book-Author            1
         Year-Of-Publication    0
         Publisher              2
         Image-URL-S            0
         Image-URL-M            0
         Image-URL-L            3
         dtype: int64
```

Checking if any null value exists in Users

```
In [86]: users.isnull().sum()

Out[86]: User-ID          0
         Location         0
         Age         110762
         dtype: int64
```



Checking if any null value exists in ratings

```
In [87]: ratings.isnull().sum()

Out[87]: User-ID        0
         ISBN           0
         Book-Rating    0
         dtype: int64
```

Checking if any Duplicate values exists in books

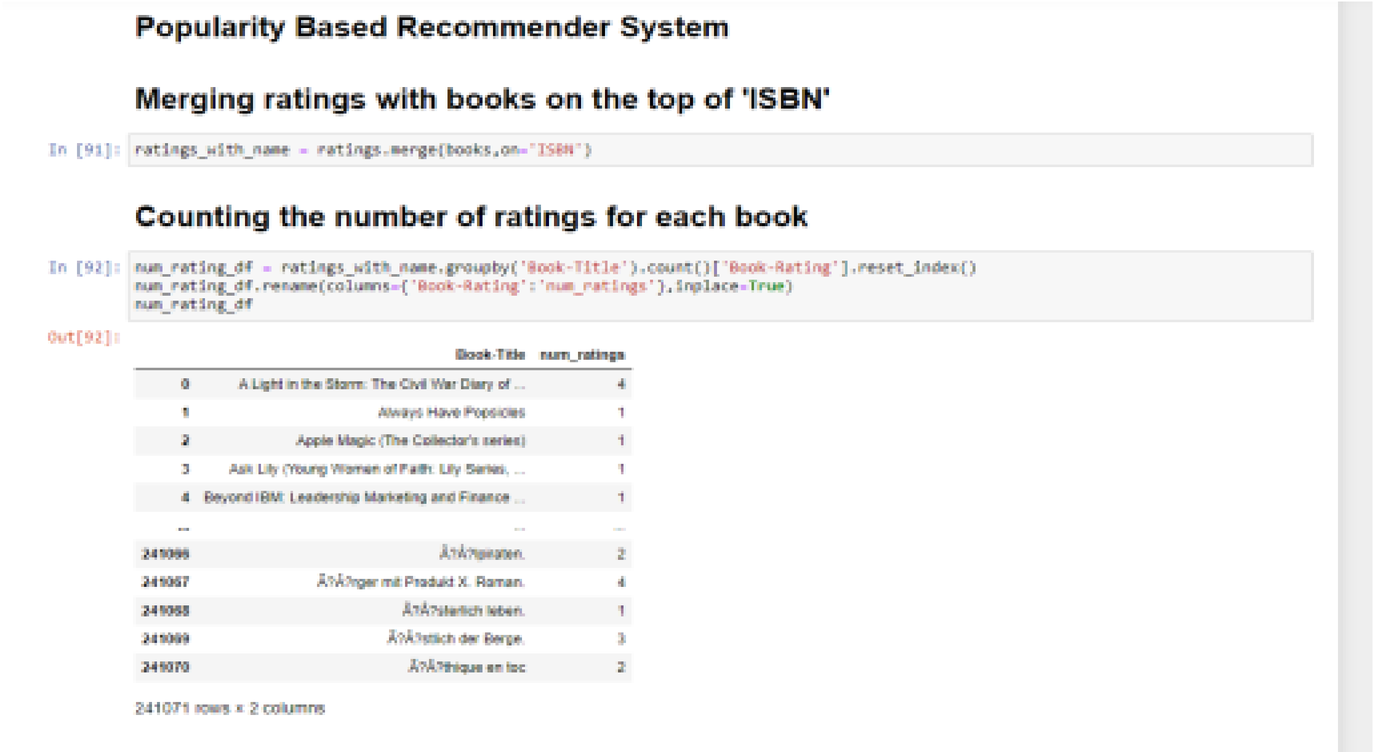
```
In [88]: books.duplicated().sum()

Out[88]: 0
```

Checking if any null value exists in ratings

```
In [89]: ratings.duplicated().sum()

Out[89]: 0
```

Checking if any null value exists in users

```
In [90]: users.duplicated().sum()

Out[90]: 0
```



Popularity Based Recommender System

Merging ratings with books on the top of 'ISBN'

```
In [91]: ratings_with_name = ratings.merge(books,on='ISBN')
```

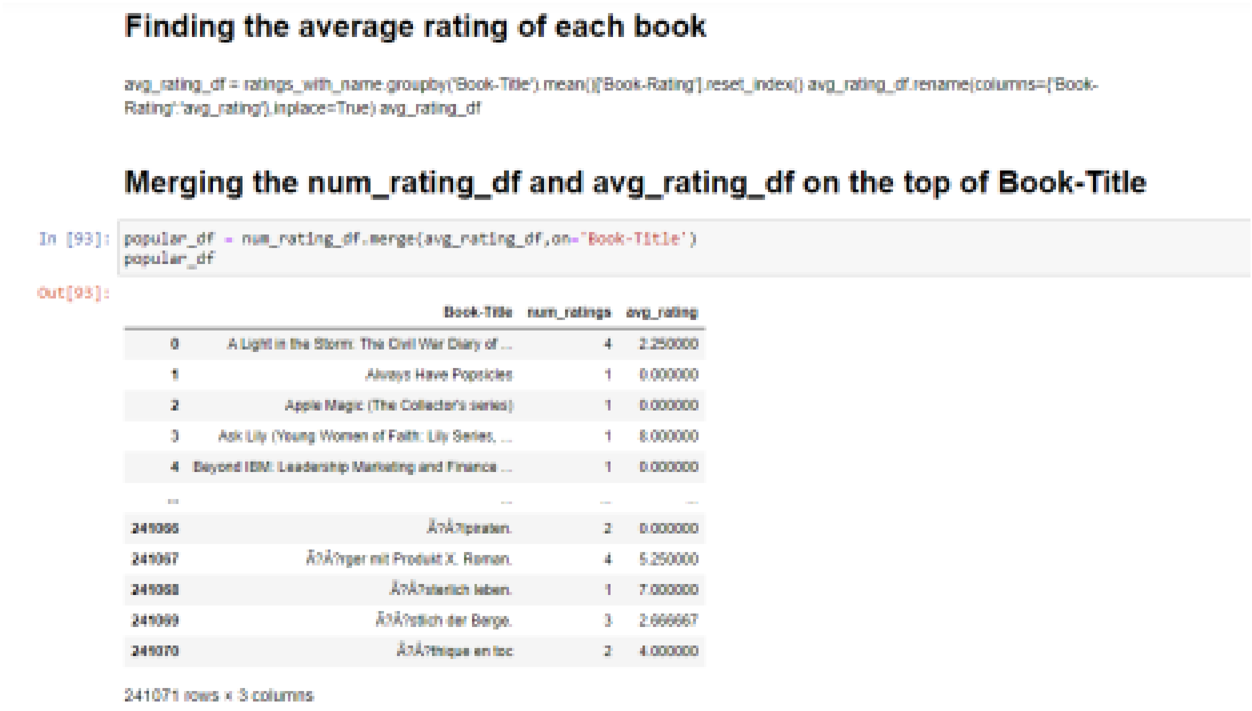
Counting the number of ratings for each book

```
In [92]: num_rating_df = ratings_with_name.groupby('Book-Title').count()['Book-Rating'].reset_index()
         num_rating_df.rename(columns={'Book-Rating':'num_ratings'},inplace=True)
         num_rating_df
```

Out[92]:

	Book-Title	num_ratings
0	A Light in the Storm: The Civil War Diary of ...	4
1	Always Have Popsicles	1
2	Apple Magic (The Collector's series)	1
3	Ask Lily (Young Women of Faith: Lily Series, ...	1
4	Beyond IBM: Leadership Marketing and Finance ...	1
...	...	...
241066	Ã?Â?lpiraten.	2
241067	Ã?Â?rger mit Produkt X. Roman.	4
241068	Ã?Â?sterlich leben.	1
241069	Ã?Â?stlich der Berge.	3
241070	Ã?Â?thique en toc	2

241071 rows × 2 columns



Finding the average rating of each book

avg_rating_df = ratings_with_name.groupby('Book-Title').mean()['Book-Rating'].reset_index() avg_rating_df.rename(columns={'Book-Rating':'avg_rating'},inplace=True) avg_rating_df

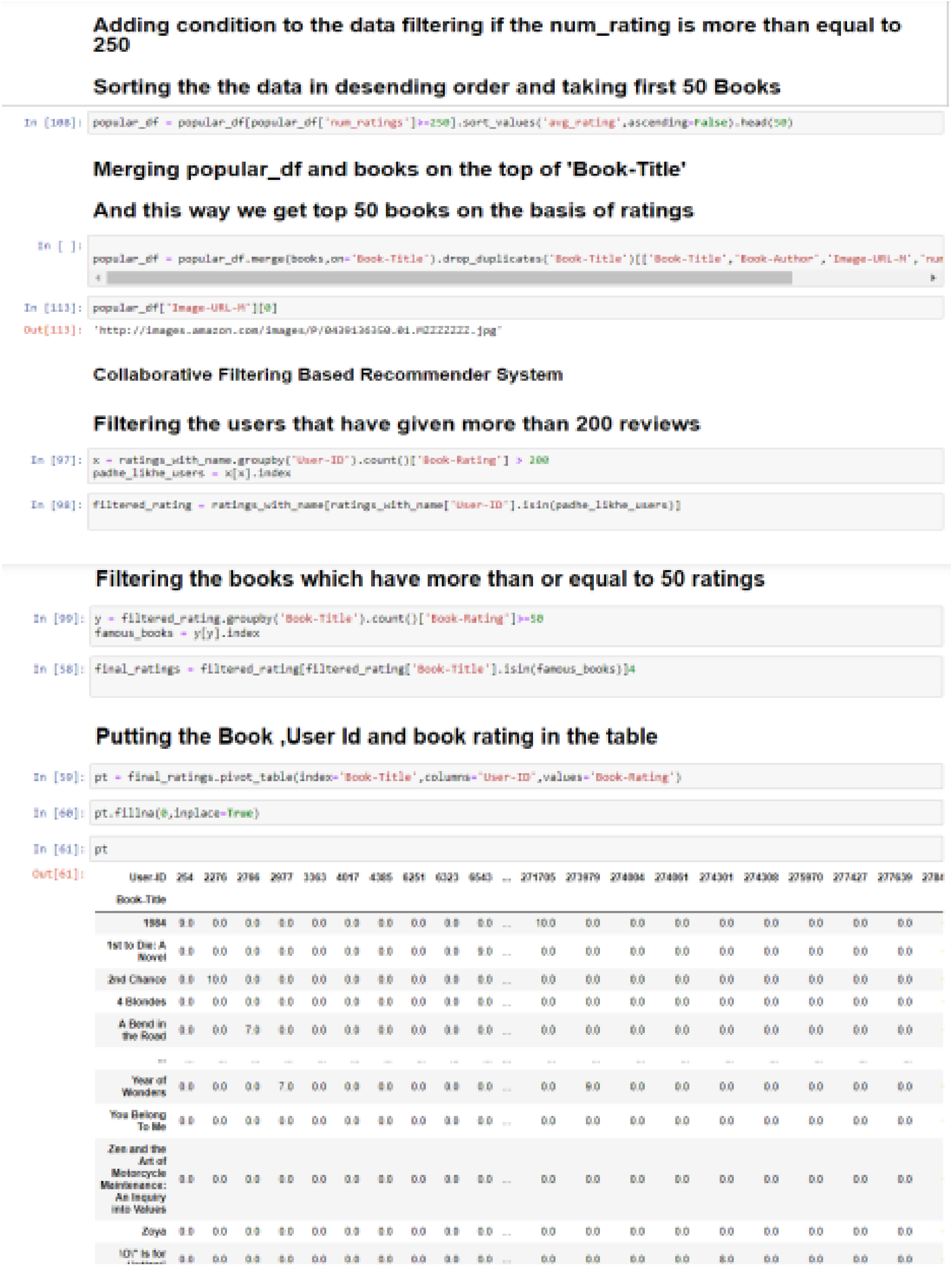
Merging the num_rating_df and avg_rating_df on the top of Book-Title

```
In [93]: popular_df = num_rating_df.merge(avg_rating_df,on='Book-Title')
         popular_df
```

Out[93]:

	Book-Title	num_ratings	avg_rating
0	A Light in the Storm: The Civil War Diary of ...	4	2.250000
1	Always Have Popsicles	1	0.000000
2	Apple Magic (The Collector's series)	1	0.000000
3	Ask Lily (Young Women of Faith: Lily Series, ...	1	8.000000
4	Beyond IBM: Leadership Marketing and Finance ...	1	0.000000
...	...	...	...
241066	Ã?Â?lpiraten.	2	0.000000
241067	Ã?Â?rger mit Produkt X. Roman.	4	5.250000
241068	Ã?Â?sterlich leben.	1	7.000000
241069	Ã?Â?stlich der Berge.	3	2.666667
241070	Ã?Â?thique en toc	2	4.000000

241071 rows × 3 columns



Adding condition to the data filtering if the num_rating is more than equal to 250

Sorting the the data in desending order and taking first 50 Books

```
In [108]: popular_df = popular_df[popular_df['num_ratings']>=250].sort_values('avg_rating',ascending=False).head(50)
```

Merging popular_df and books on the top of 'Book-Title'

And this way we get top 50 books on the basis of ratings

```
In [ ]:
popular_df = popular_df.merge(books,on='Book-Title').drop_duplicates('Book-Title')[['Book-Title','Book-Author','Image-URL-M','num
```

```
In [113]: popular_df['Image-URL-M'][0]
Out[113]: 'http://images.amazon.com/images/P/0439136350.01.MZZZZZZZ.jpg'
```

Collaborative Filtering Based Recommender System

Filtering the users that have given more than 200 reviews

```
In [97]: x = ratings_with_name.groupby('User-ID').count()['Book-Rating'] > 200
         padhe_likhe_users = x[x].index
```

```
In [98]: filtered_rating = ratings_with_name[ratings_with_name['User-ID'].isin(padhe_likhe_users)]
```

Filtering the books which have more than or equal to 50 ratings

```
In [99]: y = filtered_rating.groupby('Book-Title').count()['Book-Rating']>=50
         famous_books = y[y].index
```

```
In [58]: final_ratings = filtered_rating[filtered_rating['Book-Title'].isin(famous_books)]4
```

Putting the Book ,User Id and book rating in the table

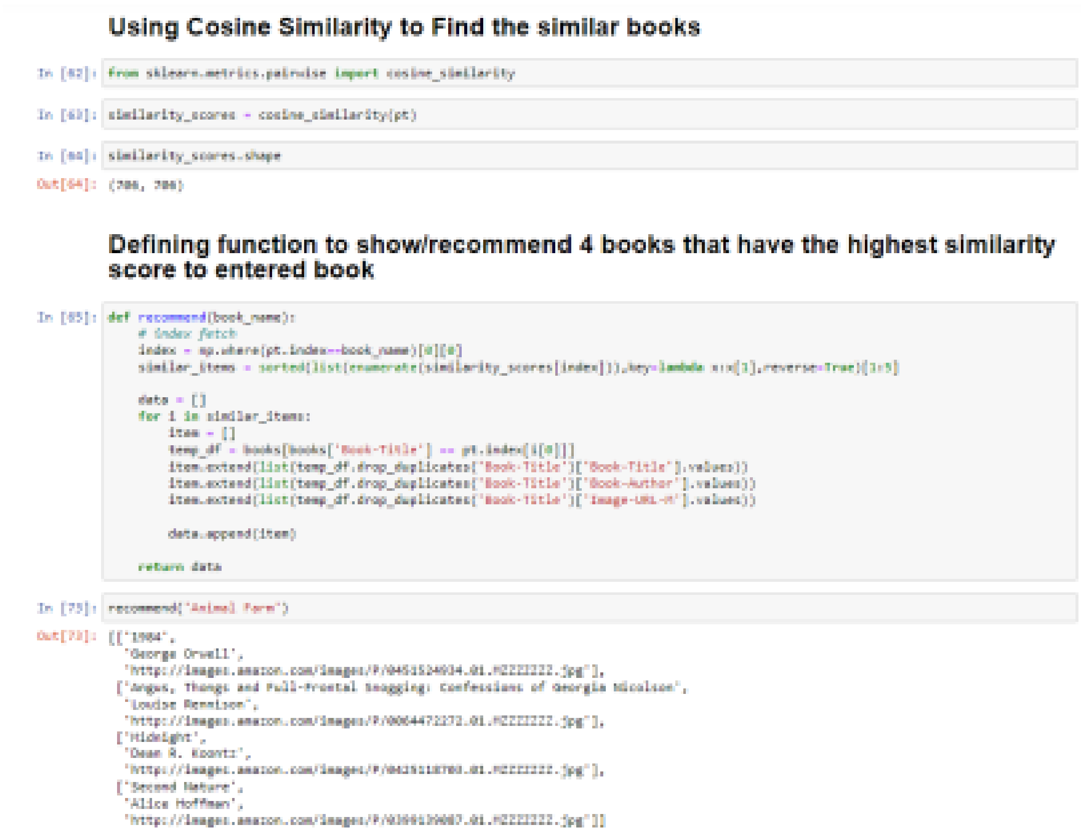
```
In [59]: pt = final_ratings.pivot_table(index='Book-Title',columns='User-ID',values='Book-Rating')
```

```
In [60]: pt.fillna(0,inplace=True)
```

```
In [61]: pt
```

Out[61]:

User-ID	254	2276	2766	2977	3363	4017	4385	6251	6323	6543	...	271705	273979	274004	274061	274301	274308	275970	277427	277639	278
Book-Title																					
1984	9.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	10.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
1st to Die: A Novel	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	9.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
2nd Chance	0.0	10.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
4 Blondes	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
A Bend in the Road	0.0	0.0	7.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
Year of Wonders	0.0	0.0	0.0	7.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	9.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
You Belong To Me	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
Zen and the Art of Motorcycle Maintenance: An Inquiry into Values	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
Zoya	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
\"O\" Is for	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	8.0	0.0	0.0	0.0	0.0	



Using Cosine Similarity to Find the similar books

```
In [62]: from sklearn.metrics.pairwise import cosine_similarity
```

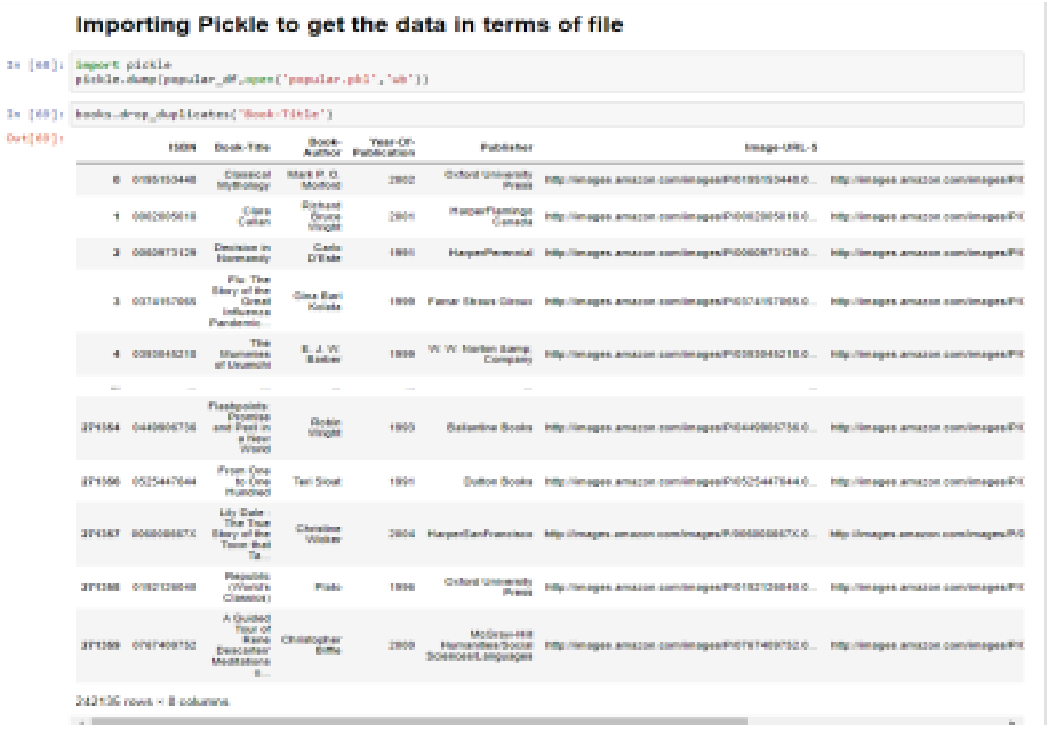
```
In [63]: similarity_scores = cosine_similarity(pt)
```

```
In [64]: similarity_scores.shape
Out[64]: (706, 706)
```

Defining function to show/recommend 4 books that have the highest similarity score to entered book

```
In [65]: def recommend(book_name):
             # index fetch
             index = np.where(pt.index==book_name)[0][0]
             similar_items = sorted(list(enumerate(similarity_scores[index])),key=lambda x:x[1],reverse=True)[1:5]

             data = []
             for i in similar_items:
                 item = []
                 temp_df = books[books['Book-Title'] == pt.index[i[0]]]
                 item.extend(list(temp_df.drop_duplicates('Book-Title')['Book-Title'].values))
                 item.extend(list(temp_df.drop_duplicates('Book-Title')['Book-Author'].values))
                 item.extend(list(temp_df.drop_duplicates('Book-Title')['Image-URL-M'].values))

                 data.append(item)

             return data
```

```
In [73]: recommend('Animal Farm')
Out[73]: [['1984',
           'George Orwell',
           'http://images.amazon.com/images/P/0451524934.01.MZZZZZZZ.jpg'],
          ['Angus, Thongs and Full-Frontal Snogging: Confessions of Georgia Nicolson',
           'Louise Rennison',
           'http://images.amazon.com/images/P/0064472272.01.MZZZZZZZ.jpg'],
          ['Midnight',
           'Dean R. Koontz',
           'http://images.amazon.com/images/P/0425118703.01.MZZZZZZZ.jpg'],
          ['Second Nature',
           'Alice Hoffman',
           'http://images.amazon.com/images/P/0399139087.01.MZZZZZZZ.jpg']]
```



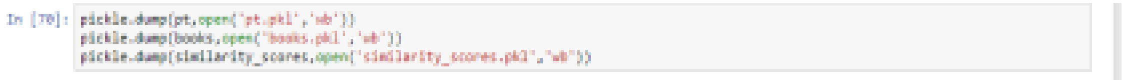
Importing Pickle to get the data in terms of file

```
In [68]: import pickle
         pickle.dump(popular_df,open('popular.pkl','wb'))
```

```
In [69]: books.drop_duplicates('Book-Title')
Out[69]:
```

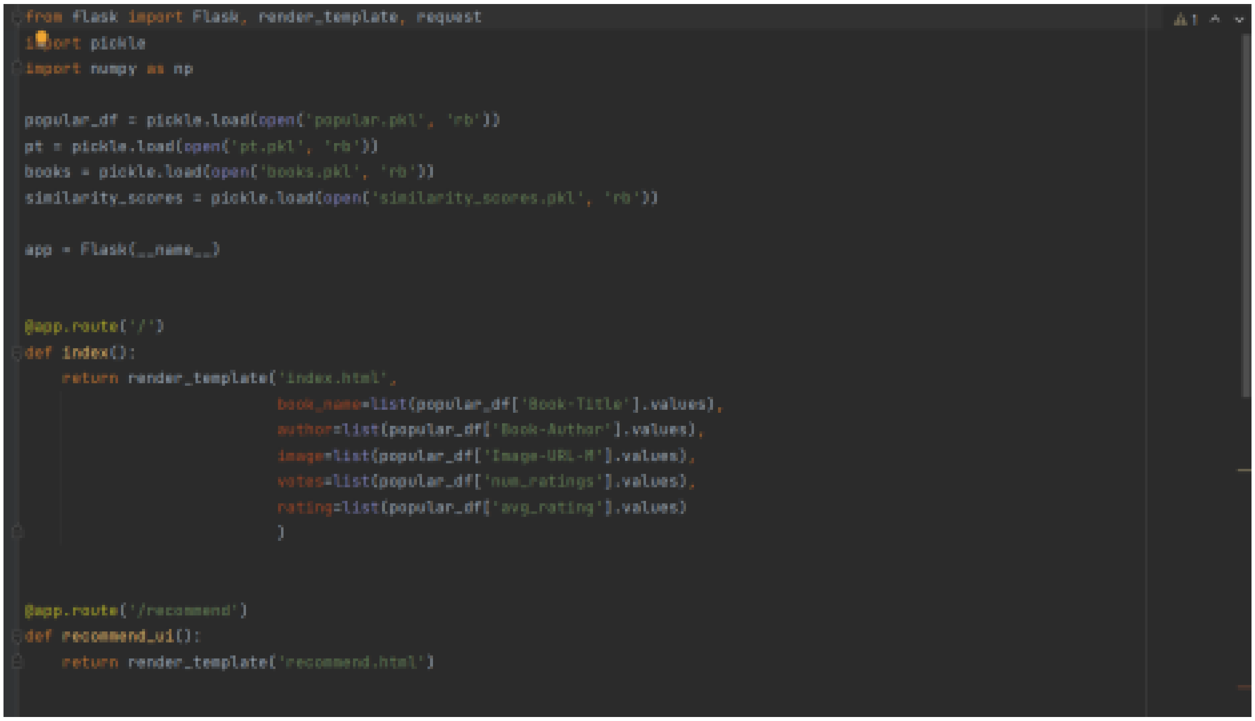
	ISBN	Book-Title	Book-Author	Year-Of-Publication	Publisher	Image-URL-S	
0	0195153448	Classical Mythology	Mark P. O. Morford	2002	Oxford University Press	http://images.amazon.com/images/P/0195153448.0...	http://images.amazon.com/images/P/0
1	0002005018	Clara Callan	Richard Bruce Wright	2001	HarperFlamingo Canada	http://images.amazon.com/images/P/0002005018.0...	http://images.amazon.com/images/P/0
2	0060973129	Decision in Normandy	Carlo D'Este	1991	HarperPerennial	http://images.amazon.com/images/P/0060973129.0...	http://images.amazon.com/images/P/0
3	0374157065	Flu: The Story of the Great Influenza Pandemic...	Gina Bari Kolata	1999	Farrar Straus Giroux	http://images.amazon.com/images/P/0374157065.0...	http://images.amazon.com/images/P/0
4	0393045218	The Mummies of Urumchi	E. J. W. Barber	1999	W. W. Norton &amp; Company	http://images.amazon.com/images/P/0393045218.0...	http://images.amazon.com/images/P/0
...	...	...	...	...	...	...	...
271354	0449906736	Flashpoints: Promise and Peril in a New World	Robin Wright	1993	Ballantine Books	http://images.amazon.com/images/P/0449906736.0...	http://images.amazon.com/images/P/0
271356	0525447644	From One to One Hundred	Teri Sloat	1991	Dutton Books	http://images.amazon.com/images/P/0525447644.0...	http://images.amazon.com/images/P/0
271357	006008667X	Lily Dale : The True Story of the Town that Ta...	Christine Wicker	2004	HarperSanFrancisco	http://images.amazon.com/images/P/006008667X.0...	http://images.amazon.com/images/P/0
271358	0192126040	Republic (World's Classics)	Plato	1996	Oxford University Press	http://images.amazon.com/images/P/0192126040.0...	http://images.amazon.com/images/P/0
271359	0767409752	A Guided Tour of Rene Descartes' Meditations o...	Christopher Biffle	2000	McGraw-Hill Humanities/Social Sciences/Languages	http://images.amazon.com/images/P/0767409752.0...	http://images.amazon.com/images/P/0

242135 rows × 8 columns

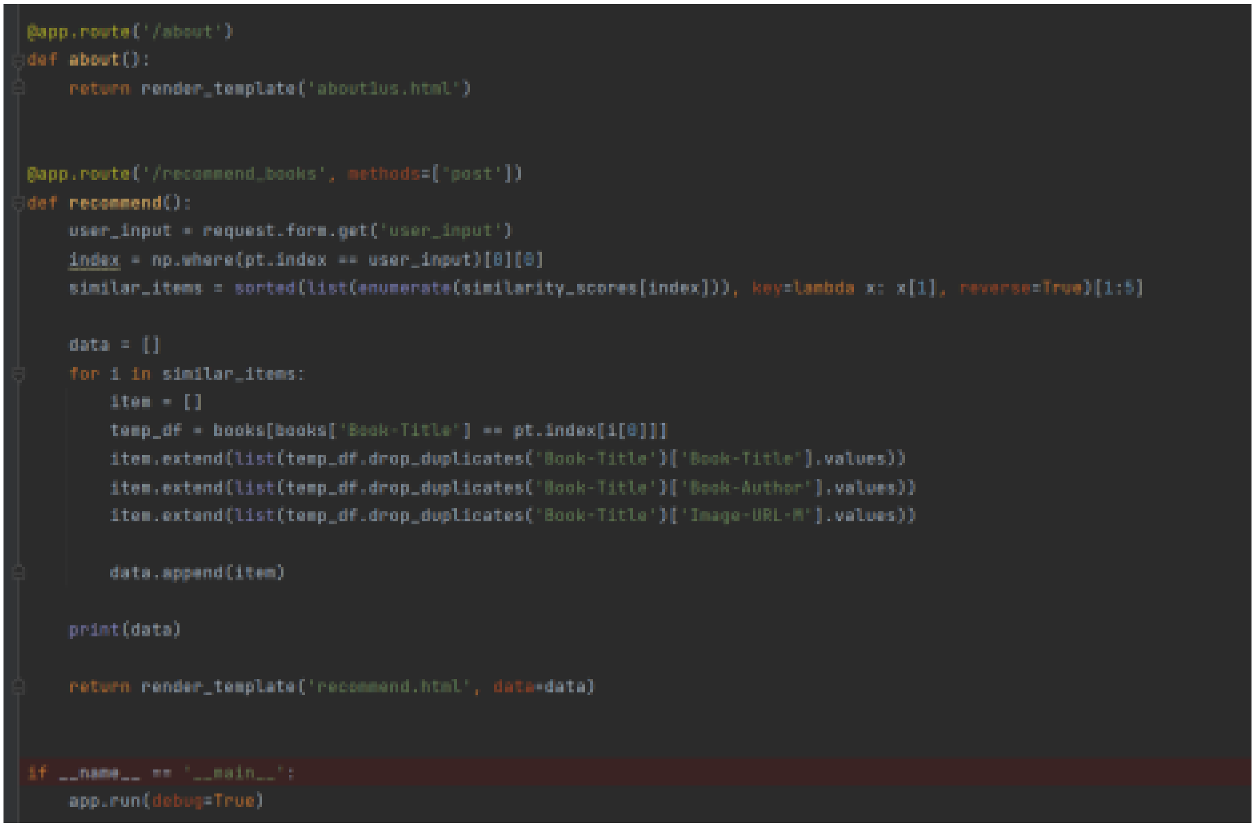


```
In [70]: pickle.dump(pt,open('pt.pkl','wb'))
         pickle.dump(books,open('books.pkl','wb'))
         pickle.dump(similarity_scores,open('similarity_scores.pkl','wb'))
```

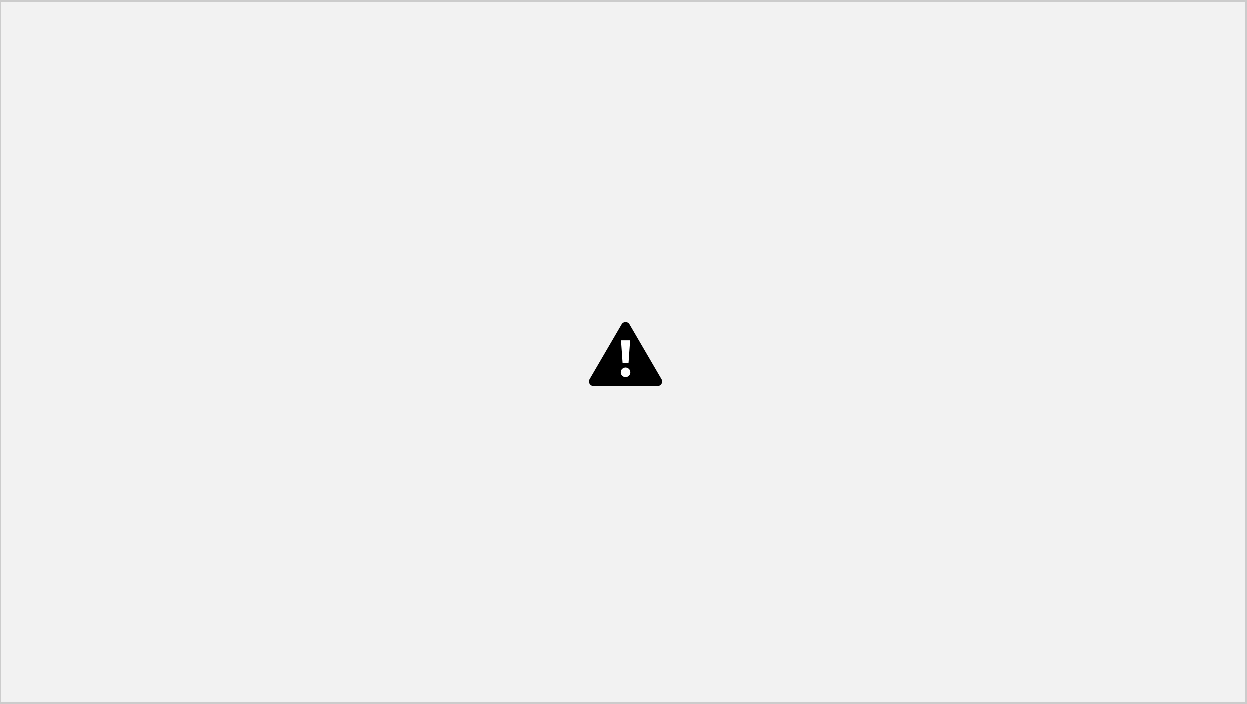
User Interface

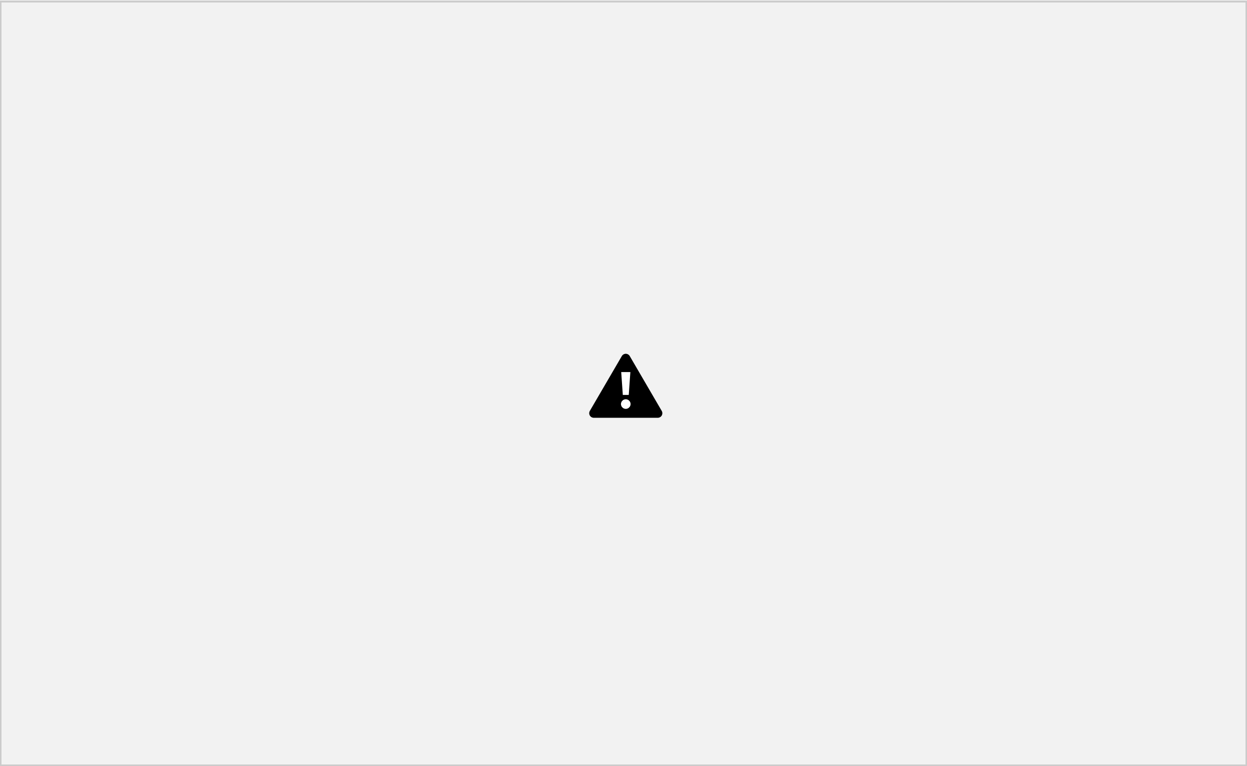


```python
from flask import Flask, render_template, request
import pickle
import numpy as np

popular_df = pickle.load(open('popular.pkl', 'rb'))
pt = pickle.load(open('pt.pkl', 'rb'))
books = pickle.load(open('books.pkl', 'rb'))
similarity_scores = pickle.load(open('similarity_scores.pkl', 'rb'))

app = Flask(__name__)


@app.route('/')
def index():
    return render_template('index.html',
                           book_name=list(popular_df['Book-Title'].values),
                           author=list(popular_df['Book-Author'].values),
                           image=list(popular_df['Image-URL-M'].values),
                           votes=list(popular_df['num_ratings'].values),
                           rating=list(popular_df['avg_rating'].values)
                           )


@app.route('/recommend')
def recommend_ui():
    return render_template('recommend.html')
```



```python
@app.route('/about')
def about():
    return render_template('aboutUs.html')


@app.route('/recommend_books', methods=['post'])
def recommend():
    user_input = request.form.get('user_input')
    index = np.where(pt.index == user_input)[0][0]
    similar_items = sorted(list(enumerate(similarity_scores[index])), key=lambda x: x[1], reverse=True)[1:5]

    data = []
    for i in similar_items:
        item = []
        temp_df = books[books['Book-Title'] == pt.index[i[0]]]
        item.extend(list(temp_df.drop_duplicates('Book-Title')['Book-Title'].values))
        item.extend(list(temp_df.drop_duplicates('Book-Title')['Book-Author'].values))
        item.extend(list(temp_df.drop_duplicates('Book-Title')['Image-URL-M'].values))

        data.append(item)

    print(data)

    return render_template('recommend.html', data=data)


if __name__ == '__main__':
    app.run(debug=True)
```





Conclusion -

We learned about filtering the useful data from the given data set to get the relevant recommendation. We also learned to nullify the blank values and remove the duplicate values out of the data sets. We got to know the different types of recommendation systems. We used the popularity recommendation and collaborative filtering system. According to the rating given by the users on each book the books were arranged in the highest rating to the lowest rating and amongst them the top 50 books were recommended. And were shown on the web page.

We also created a webpage where the user could enter a book name and the 4 similar books would be recommended to the user. The similarity in books was obtained by the collaborative filtering by applying **cosine similarity** method. Collaborative filtering uses similarities between users and items simultaneously to provide recommendations. This allows for serendipitous recommendations; that is, collaborative filtering models can recommend an item to user A based on the interests of a similar user. For these purposes we used different python libraries such as NumPy, pandas, flask and sk-learn library.

So, this way we created a book recommendation system which can recommend similar books.