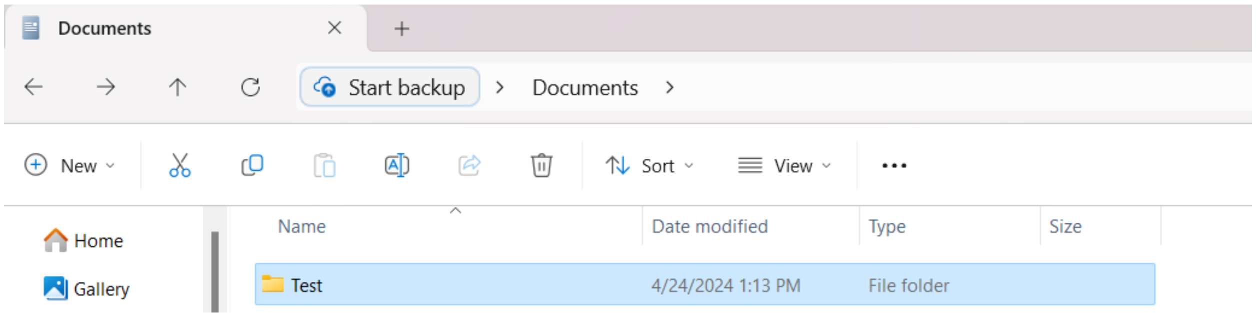
Assignment #2

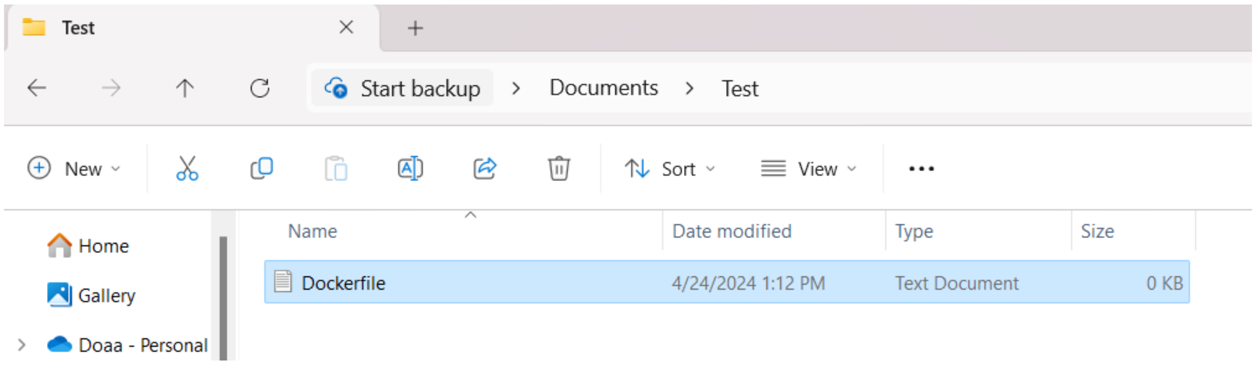
Name: Doaa Samir **ID:** 2103114 **Department:** AI

First I created a folder in the documents and named it as Test



Name	Date modified	Type	Size
Test	4/24/2024 1:13 PM	File folder	

Then I created a text file inside this folder and named it as Dockerfile



Name	Date modified	Type	Size
Dockerfile	4/24/2024 1:12 PM	Text Document	0 KB

And then I will open this Dockerfile in the visual studio and I will write these commands as its shown below:

1. FROM jupyter/datascience-notebook: This line specifies the base image to use for the Docker image.

2. WORKDIR /Test: This line sets the working directory inside the Docker container to /Test.

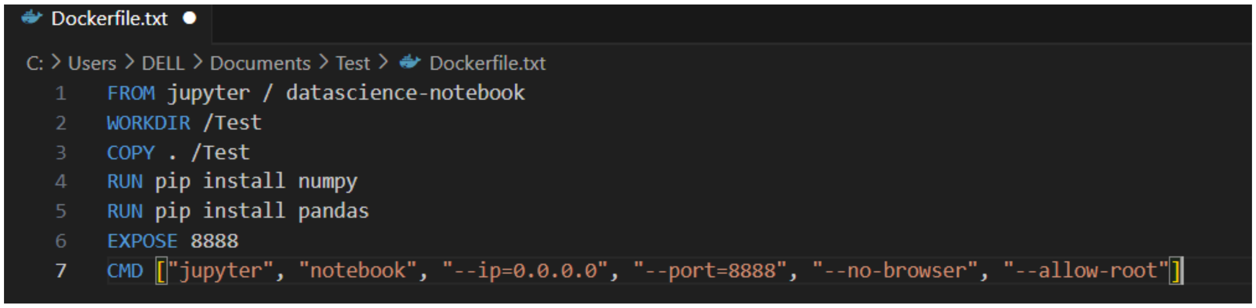
3. COPY . /Test: This line copies all files and directories from the current directory on the host machine (where the Dockerfile is located) to the /Test directory inside the Docker container.

4. RUN pip install numpy: This line installs the NumPy library inside the Docker container using pip.

5. RUN pip install pandas: This line installs the pandas library inside the Docker container using pip.

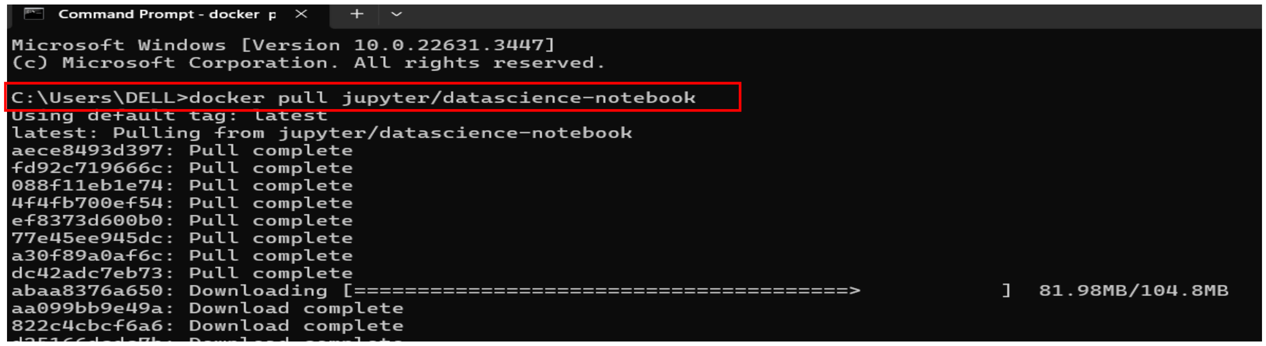
6. EXPOSE 8888: This line exposes port 8888, which is the default port used by Jupyter Notebook, to allow incoming connections.

7. CMD ["jupyter", "notebook", "--ip=0.0.0.0", "--port=8888", "--no-browser", "--allow-root"]: This line specifies the default command to run when the Docker container starts. It starts a Jupyter Notebook server accessible on all network interfaces (**"--ip=0.0.0.0)** on port 8888 **(--port=8888)**. The **--no-browser",** option prevents Jupyter Notebook from opening a browser window automatically, and the **--allow-root** option allows running the notebook server with root permissions inside the Docker container.



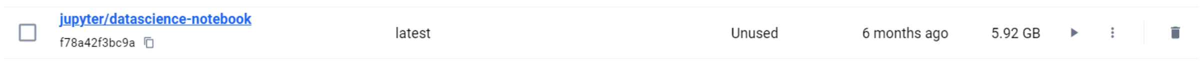
```
FROM jupyter / datascience-notebook
WORKDIR /Test
COPY . /Test
RUN pip install numpy
RUN pip install pandas
EXPOSE 8888
CMD ["jupyter", "notebook", "--ip=0.0.0.0", "--port=8888", "--no-browser", "--allow-root"]
```

I will use this command **“docker pull jupyter/datascience-notebook”** and run it in cmd, it is used to download a specific package or software from the internet. it's pulling a package called "jupyter/datascience-notebook" from a service called Docker.



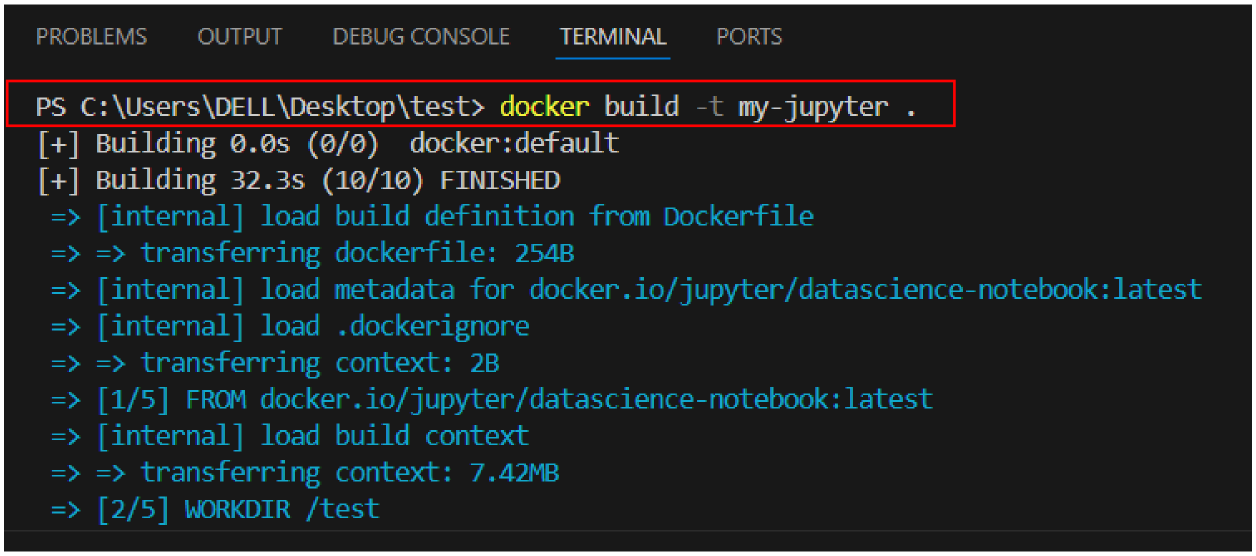
```
Microsoft Windows [Version 10.0.22631.3447]
(c) Microsoft Corporation. All rights reserved.

C:\Users\DELL>docker pull jupyter/datascience-notebook
Using default tag: latest
latest: Pulling from jupyter/datascience-notebook
aece8493d397: Pull complete
fd92c719666c: Pull complete
088f11eb1e74: Pull complete
4f4fb700ef54: Pull complete
ef8373d600b0: Pull complete
77e45ee945dc: Pull complete
a30f89a0af6c: Pull complete
dc42adc7eb73: Pull complete
abaa8376a650: Downloading [======================================>            ]  81.98MB/104.8MB
aa099bb9e49a: Download complete
822c4cbcf6a6: Download complete
```

Now I have this jupyter/datascience-notebook image in Docker

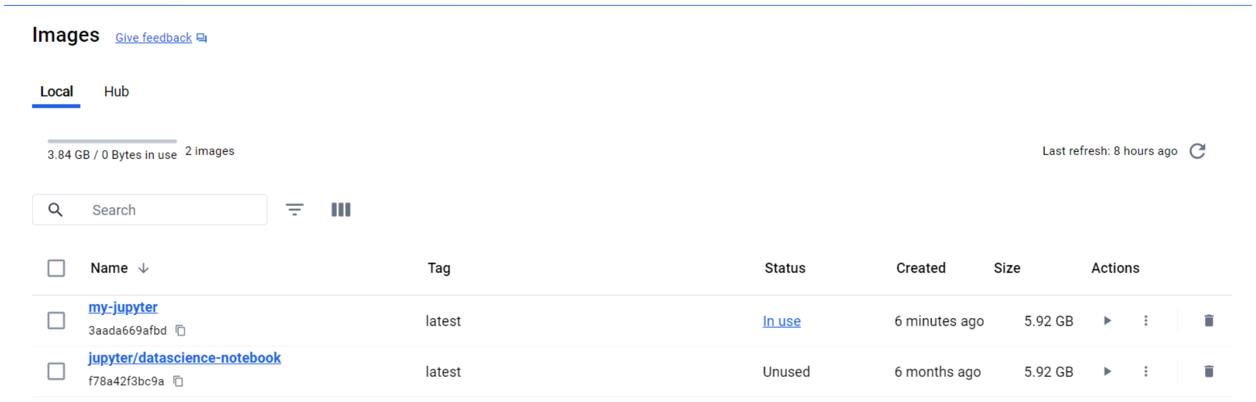


Name	Tag	Status	Created	Size
jupyter/datascience-notebook f78a42f3bc9a	latest	Unused	6 months ago	5.92 GB

After this I did the docker-build as the command is written below

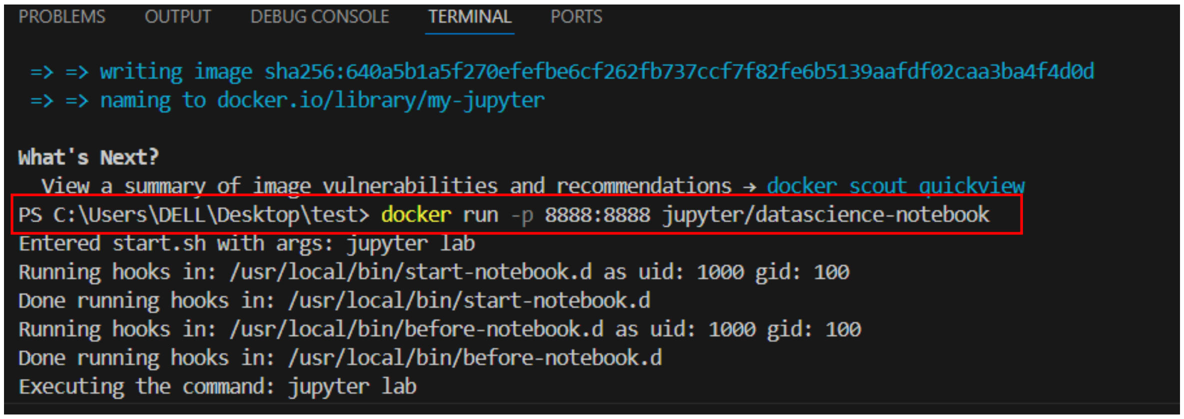


```
PS C:\Users\DELL\Desktop\test> docker build -t my-jupyter .
[+] Building 0.0s (0/0)  docker:default
[+] Building 32.3s (10/10) FINISHED
 => [internal] load build definition from Dockerfile
 => => transferring dockerfile: 254B
 => [internal] load metadata for docker.io/jupyter/datascience-notebook:latest
 => [internal] load .dockerignore
 => => transferring context: 2B
 => [1/5] FROM docker.io/jupyter/datascience-notebook:latest
 => [internal] load build context
 => => transferring context: 7.42MB
 => [2/5] WORKDIR /test
```



Name	Tag	Status	Created	Size
my-jupyter 3aada669afbd	latest	In use	6 minutes ago	5.92 GB
jupyter/datascience-notebook f78a42f3bc9a	latest	Unused	6 months ago	5.92 GB

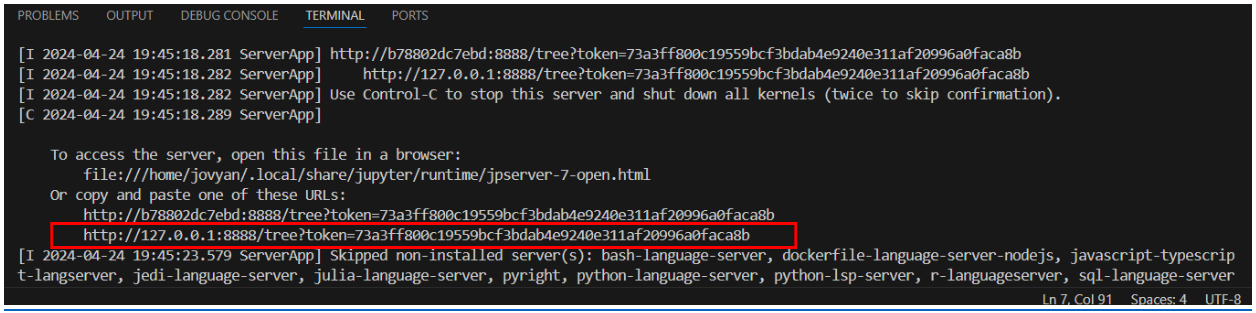
Next thing is that I run the docker as the command is written below



```
 => => writing image sha256:640a5b1a5f270efefbe6cf262fb737ccf7f82fe6b5139aafdf02caa3ba4f4d0d
 => => naming to docker.io/library/my-jupyter

What's Next?
  View a summary of image vulnerabilities and recommendations → docker scout quickview
PS C:\Users\DELL\Desktop\test> docker run -p 8888:8888 jupyter/datascience-notebook
Entered start.sh with args: jupyter lab
Running hooks in: /usr/local/bin/start-notebook.d as uid: 1000 gid: 100
Done running hooks in: /usr/local/bin/start-notebook.d
Running hooks in: /usr/local/bin/before-notebook.d as uid: 1000 gid: 100
Done running hooks in: /usr/local/bin/before-notebook.d
Executing the command: jupyter lab
```

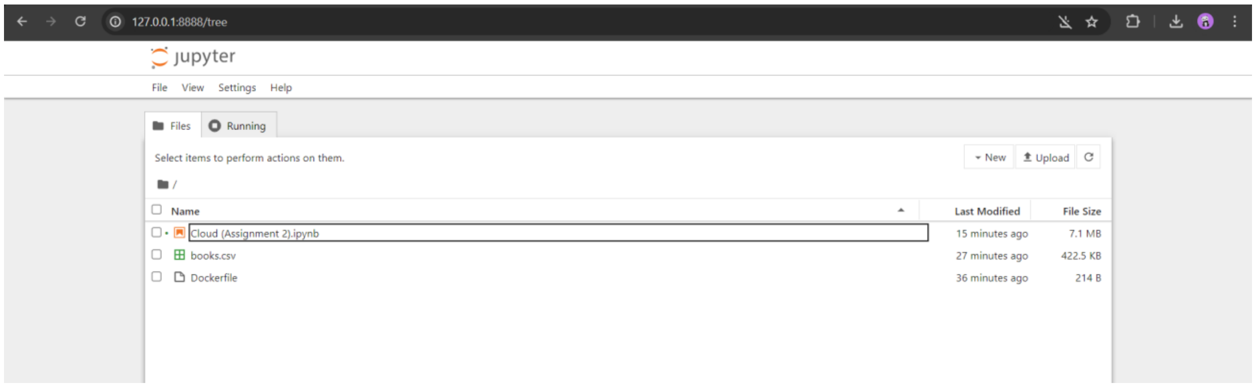
It gave me these two links so I will be clicking on the second one:

http://127.0.0.1:8888/tree?token=73a3ff800c19559bcf3bdab4e9240e311af20996a0faca8b



```
[I 2024-04-24 19:45:18.281 ServerApp] http://b78802dc7ebd:8888/tree?token=73a3ff800c19559bcf3bdab4e9240e311af20996a0faca8b
[I 2024-04-24 19:45:18.282 ServerApp]     http://127.0.0.1:8888/tree?token=73a3ff800c19559bcf3bdab4e9240e311af20996a0faca8b
[I 2024-04-24 19:45:18.282 ServerApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[C 2024-04-24 19:45:18.289 ServerApp]

    To access the server, open this file in a browser:
        file:///home/jovyan/.local/share/jupyter/runtime/jpserver-7-open.html
    Or copy and paste one of these URLs:
        http://b78802dc7ebd:8888/tree?token=73a3ff800c19559bcf3bdab4e9240e311af20996a0faca8b
        http://127.0.0.1:8888/tree?token=73a3ff800c19559bcf3bdab4e9240e311af20996a0faca8b
[I 2024-04-24 19:45:23.579 ServerApp] Skipped non-installed server(s): bash-language-server, dockerfile-language-server-nodejs, javascript-typescrip
t-langserver, jedi-language-server, julia-language-server, pyright, python-language-server, python-lsp-server, r-languageserver, sql-language-server
```

This is how the page will look like once I click on the link:



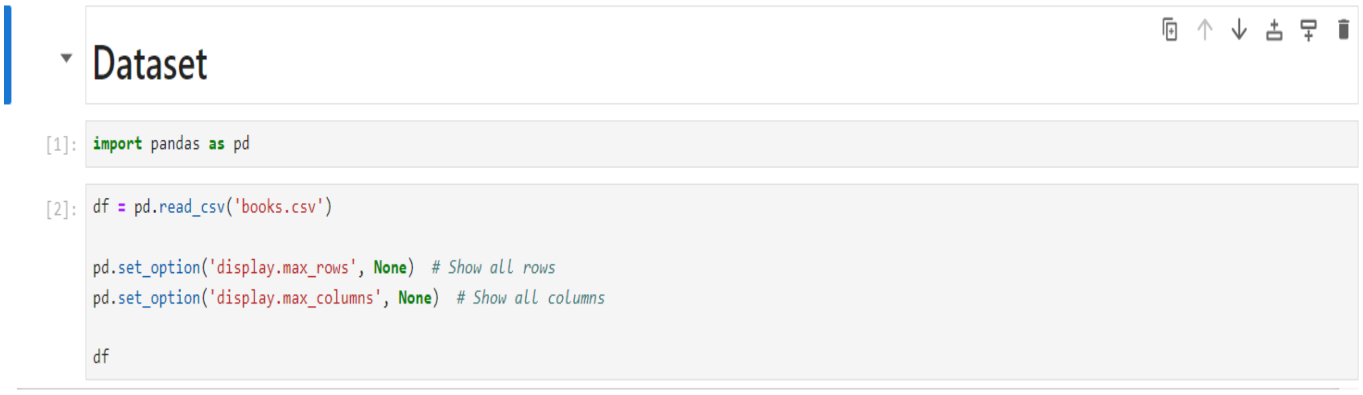
Select items to perform actions on them.

/

Name	Last Modified	File Size
Cloud (Assignment 2).ipynb	15 minutes ago	7.1 MB
books.csv	27 minutes ago	422.5 KB
Dockerfile	36 minutes ago	214 B

Code & Explanation

This is the displaying dataset

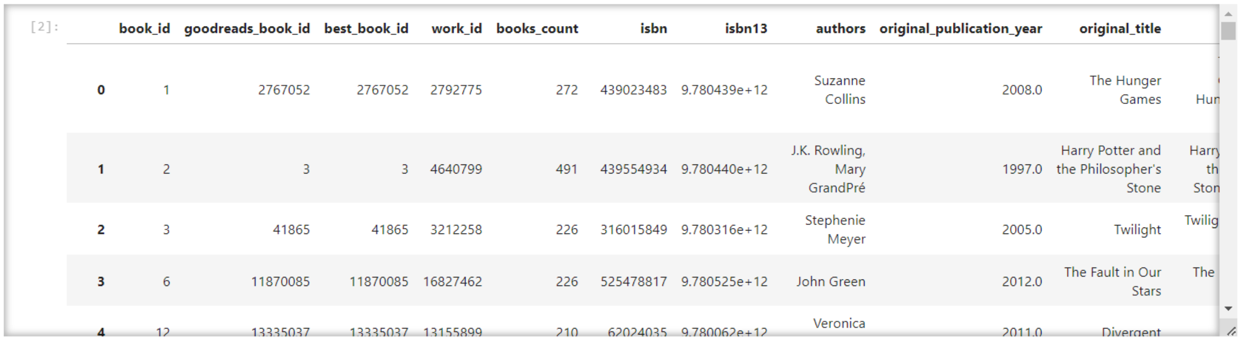


Dataset

```
[1]: import pandas as pd
```

```
[2]: df = pd.read_csv('books.csv')

     pd.set_option('display.max_rows', None)  # Show all rows
     pd.set_option('display.max_columns', None)  # Show all columns

     df
```

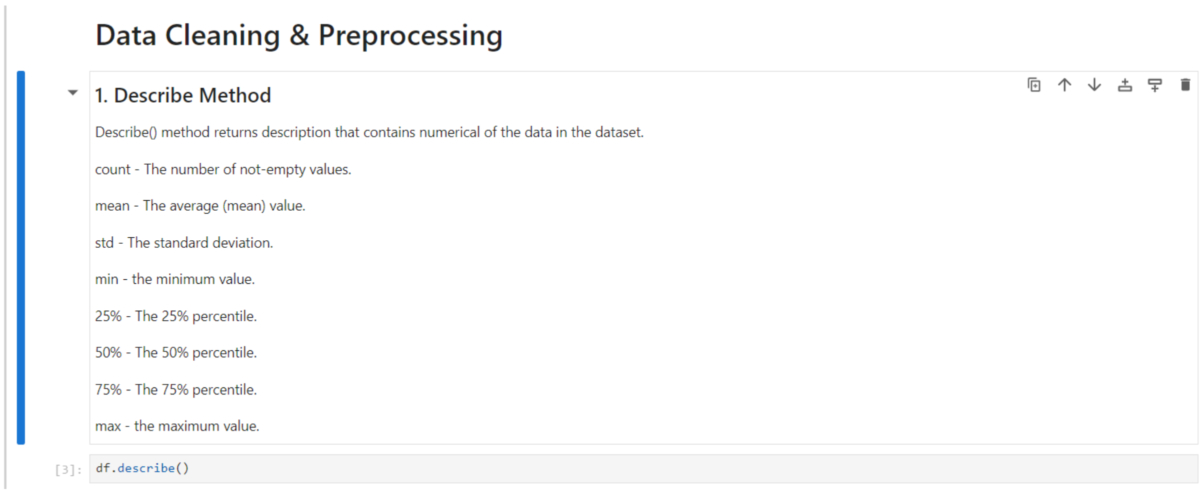
Output:



[2]:

	book_id	goodreads_book_id	best_book_id	work_id	books_count	isbn	isbn13	authors	original_publication_year	original_title	
0	1	2767052	2767052	2792775	272	439023483	9.780439e+12	Suzanne Collins	2008.0	The Hunger Games	Hun
1	2	3	3	4640799	491	439554934	9.780440e+12	J.K. Rowling, Mary GrandPré	1997.0	Harry Potter and the Philosopher's Stone	Harry th Ston
2	3	41865	41865	3212258	226	316015849	9.780316e+12	Stephenie Meyer	2005.0	Twilight	Twilig
3	6	11870085	11870085	16827462	226	525478817	9.780525e+12	John Green	2012.0	The Fault in Our Stars	The
4	12	13335037	13335037	13155899	210	62024035	9.780062e+12	Veronica	2011.0	Divergent	

Data Cleaning & Preprocessing:



Data Cleaning & Preprocessing

1. Describe Method

Describe() method returns description that contains numerical of the data in the dataset.

count - The number of not-empty values.

mean - The average (mean) value.

std - The standard deviation.

min - the minimum value.

25% - The 25% percentile.

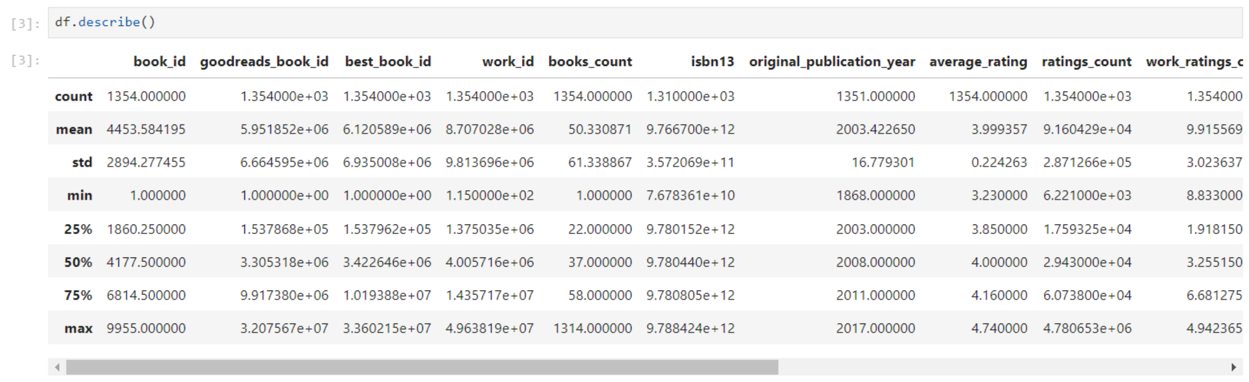
50% - The 50% percentile.

75% - The 75% percentile.

max - the maximum value.

```
[3]: df.describe()
```

Output:

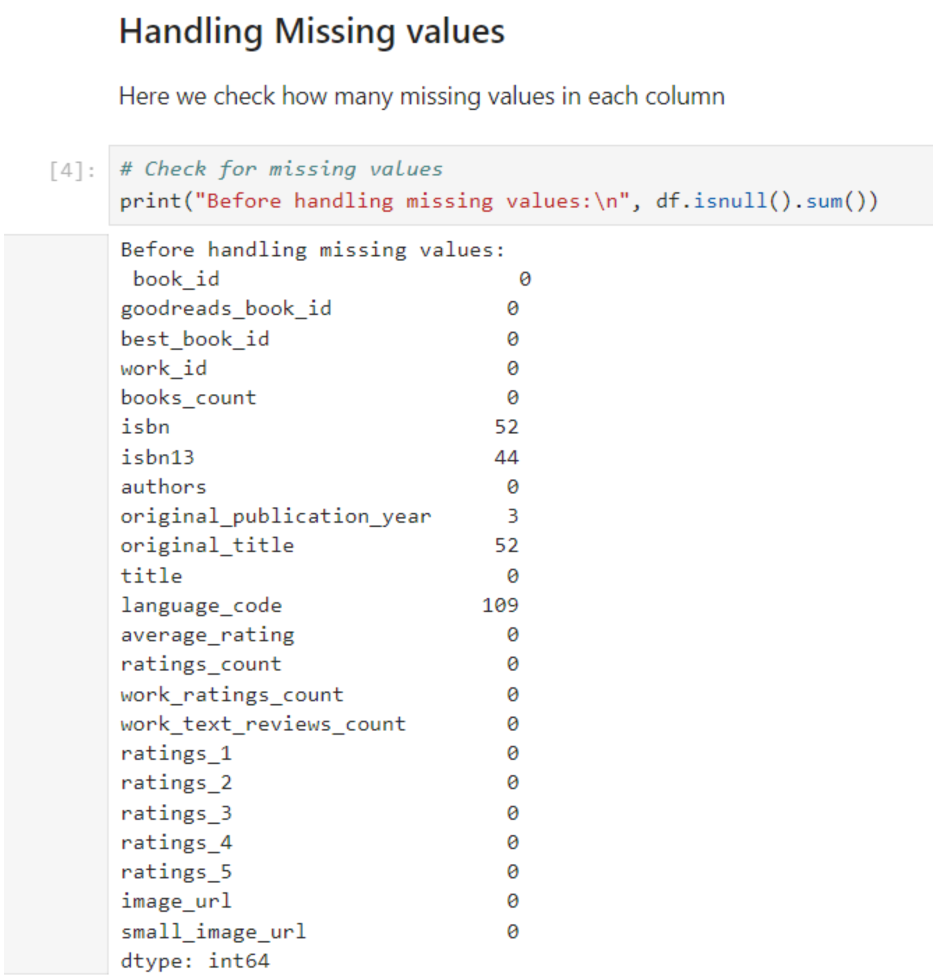


```
[3]: df.describe()
```

[3]:

	book_id	goodreads_book_id	best_book_id	work_id	books_count	isbn13	original_publication_year	average_rating	ratings_count	work_ratings_c
count	1354.000000	1.354000e+03	1.354000e+03	1.354000e+03	1354.000000	1.310000e+03	1351.000000	1354.000000	1.354000e+03	1.354000
mean	4453.584195	5.951852e+06	6.120589e+06	8.707028e+06	50.330871	9.766700e+12	2003.422650	3.999357	9.160429e+04	9.915569
std	2894.277455	6.664595e+06	6.935008e+06	9.813696e+06	61.338867	3.572069e+11	16.779301	0.224263	2.871266e+05	3.023637
min	1.000000	1.000000e+00	1.000000e+00	1.150000e+02	1.000000	7.678361e+10	1868.000000	3.230000	6.221000e+03	8.833000
25%	1860.250000	1.537868e+05	1.537962e+05	1.375035e+06	22.000000	9.780152e+12	2003.000000	3.850000	1.759325e+04	1.918150
50%	4177.500000	3.305318e+06	3.422646e+06	4.005716e+06	37.000000	9.780440e+12	2008.000000	4.000000	2.943000e+04	3.255150
75%	6814.500000	9.917380e+06	1.019388e+07	1.435717e+07	58.000000	9.780805e+12	2011.000000	4.160000	6.073800e+04	6.681275
max	9955.000000	3.207567e+07	3.360215e+07	4.963819e+07	1314.000000	9.788424e+12	2017.000000	4.740000	4.780653e+06	4.942365

This is handling missing values. First we check if there are missing values and as we can see down below the numbers that are in column isbn, isbn13, original_publication_year, original_title, language_code indicates how many missing values in each column.



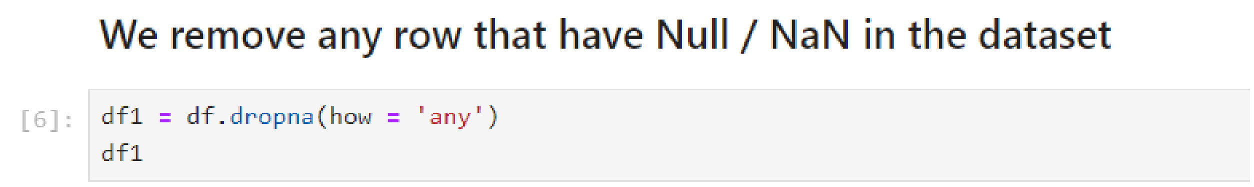
Handling Missing values

Here we check how many missing values in each column

```
[4]: # Check for missing values
     print("Before handling missing values:\n", df.isnull().sum())
```

```
Before handling missing values:
 book_id                      0
goodreads_book_id            0
best_book_id                 0
work_id                      0
books_count                  0
isbn                        52
isbn13                      44
authors                      0
original_publication_year    3
original_title              52
title                        0
language_code              109
average_rating               0
ratings_count                0
work_ratings_count           0
work_text_reviews_count      0
ratings_1                    0
ratings_2                    0
ratings_3                    0
ratings_4                    0
ratings_5                    0
image_url                    0
small_image_url              0
dtype: int64
```

Below we remove any row that have NaN / Null in the dataset

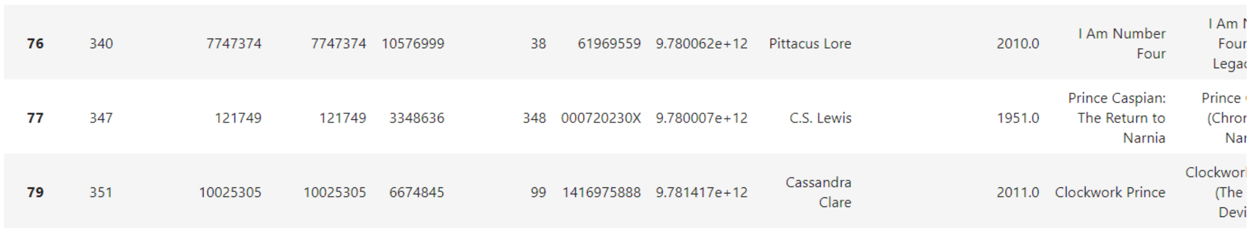


We remove any row that have Null / NaN in the dataset

```
[6]: df1 = df.dropna(how = 'any')
     df1
```

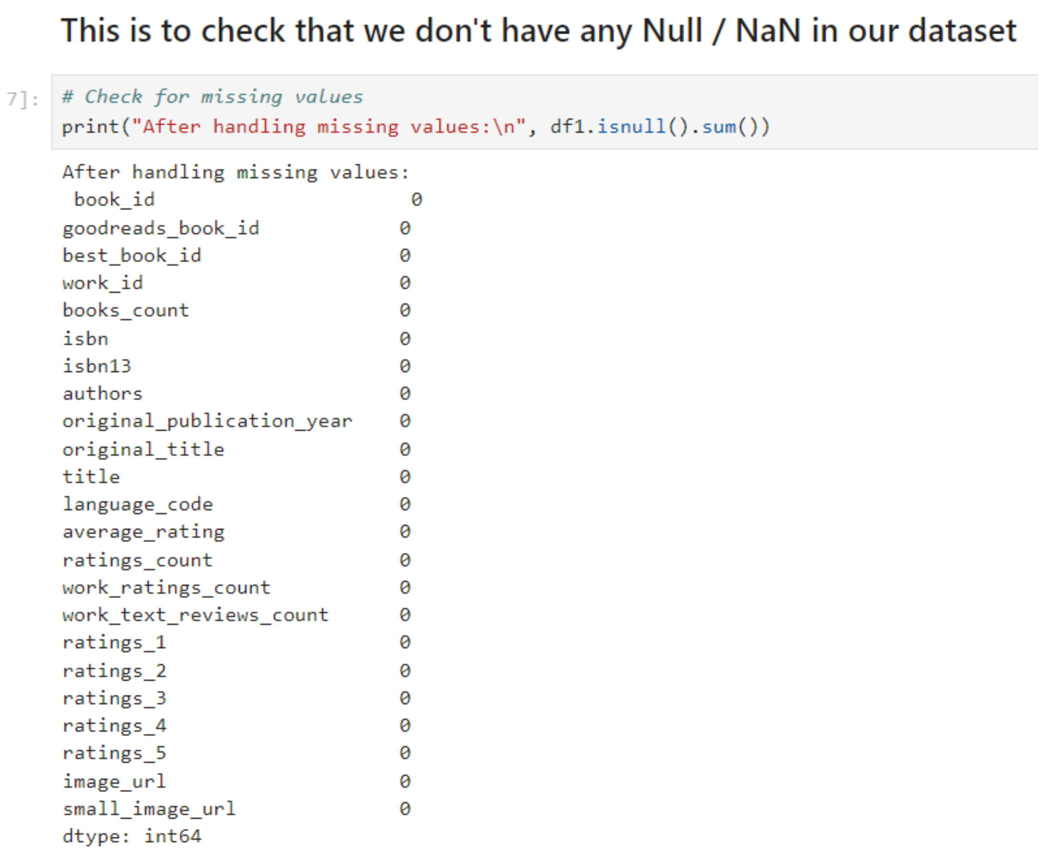
Output:

In the original data when we look at the column isbn, isbn13 of row 78 we will see that they had NaN, but after doing the function dropna() we can see in the output below that it had been removed



76	340	7747374	7747374	10576999	38	61969559	9.780062e+12	Pittacus Lore	2010.0	I Am Number Four	I Am I Four Legac
77	347	121749	121749	3348636	348	000720230X	9.780007e+12	C.S. Lewis	1951.0	Prince Caspian: The Return to Narnia	Prince (Chror Nar
79	351	10025305	10025305	6674845	99	1416975888	9.781417e+12	Cassandra Clare	2011.0	Clockwork Prince	Clockwor (The Devi

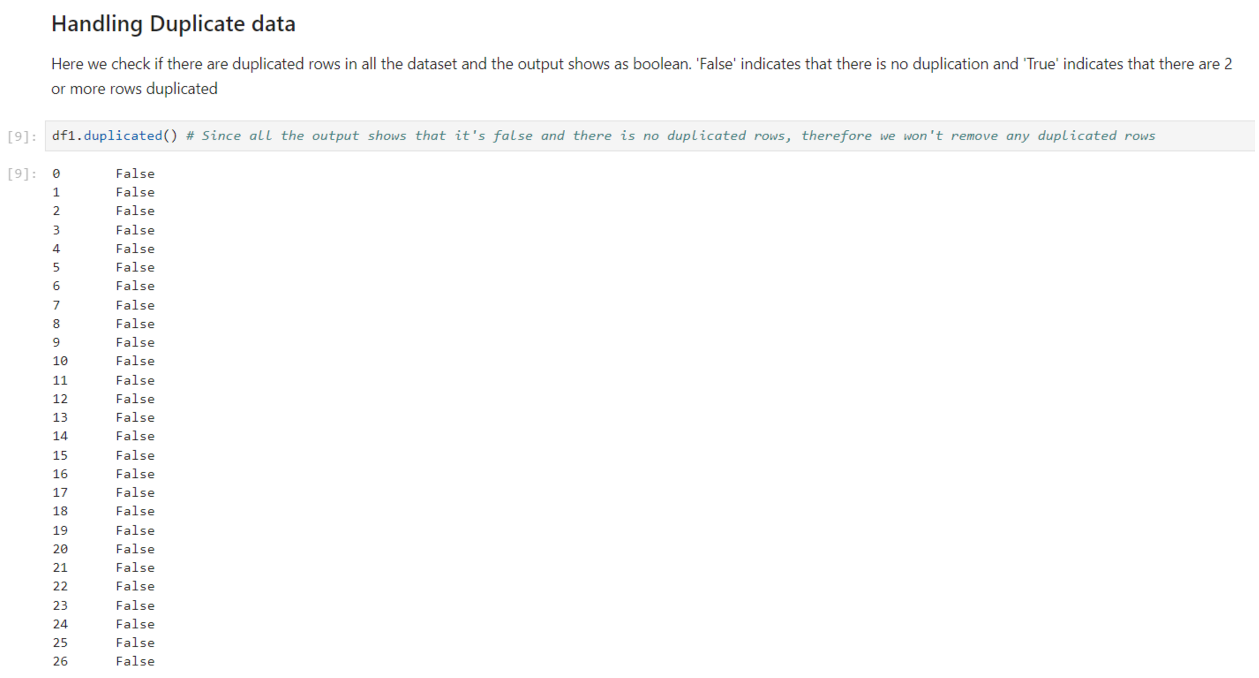
This is how we check that our dataset does not have any NaN / Null values



This is to check that we don't have any Null / NaN in our dataset

```
7]: # Check for missing values
    print("After handling missing values:\n", df1.isnull().sum())

    After handling missing values:
     book_id                     0
    goodreads_book_id           0
    best_book_id                0
    work_id                     0
    books_count                 0
    isbn                        0
    isbn13                      0
    authors                     0
    original_publication_year   0
    original_title              0
    title                       0
    language_code               0
    average_rating              0
    ratings_count               0
    work_ratings_count          0
    work_text_reviews_count     0
    ratings_1                   0
    ratings_2                   0
    ratings_3                   0
    ratings_4                   0
    ratings_5                   0
    image_url                   0
    small_image_url             0
    dtype: int64
```

Here we check to see if our data have duplicate rows, and since the output shows all False then that means that there is no any duplicated rows, otherwise it would have shown True.



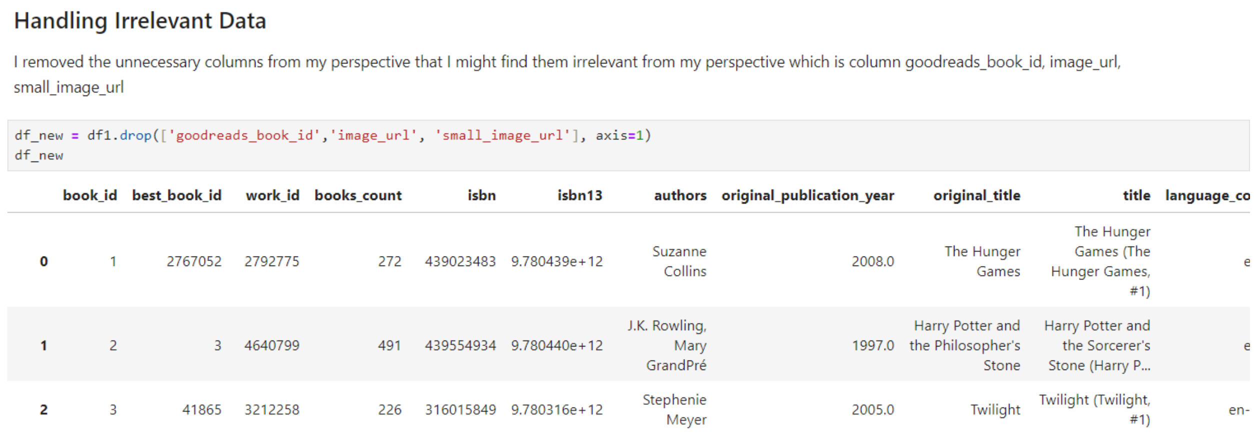
Handling Duplicate data

Here we check if there are duplicated rows in all the dataset and the output shows as boolean. 'False' indicates that there is no duplication and 'True' indicates that there are 2 or more rows duplicated

```
[9]: df1.duplicated() # Since all the output shows that it's false and there is no duplicated rows, therefore we won't remove any duplicated rows
```

```
[9]: 0       False
     1       False
     2       False
     3       False
     4       False
     5       False
     6       False
     7       False
     8       False
     9       False
     10      False
     11      False
     12      False
     13      False
     14      False
     15      False
     16      False
     17      False
     18      False
     19      False
     20      False
     21      False
     22      False
     23      False
     24      False
     25      False
     26      False
```

Here we remove the unnecessary / irrelevant column and so from my perspective I see that ‘goodreads_book_id’, ‘image_url’, ‘small_image_url’ are irrelevant, so I remove them for better results in the analysis and as we see down below it has been removed from the dataset.

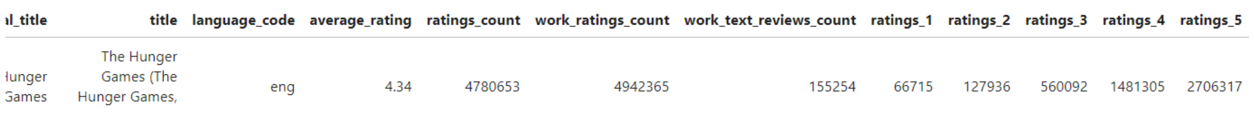


Handling Irrelevant Data

I removed the unnecessary columns from my perspective that I might find them irrelevant from my perspective which is column goodreads_book_id, image_url, small_image_url

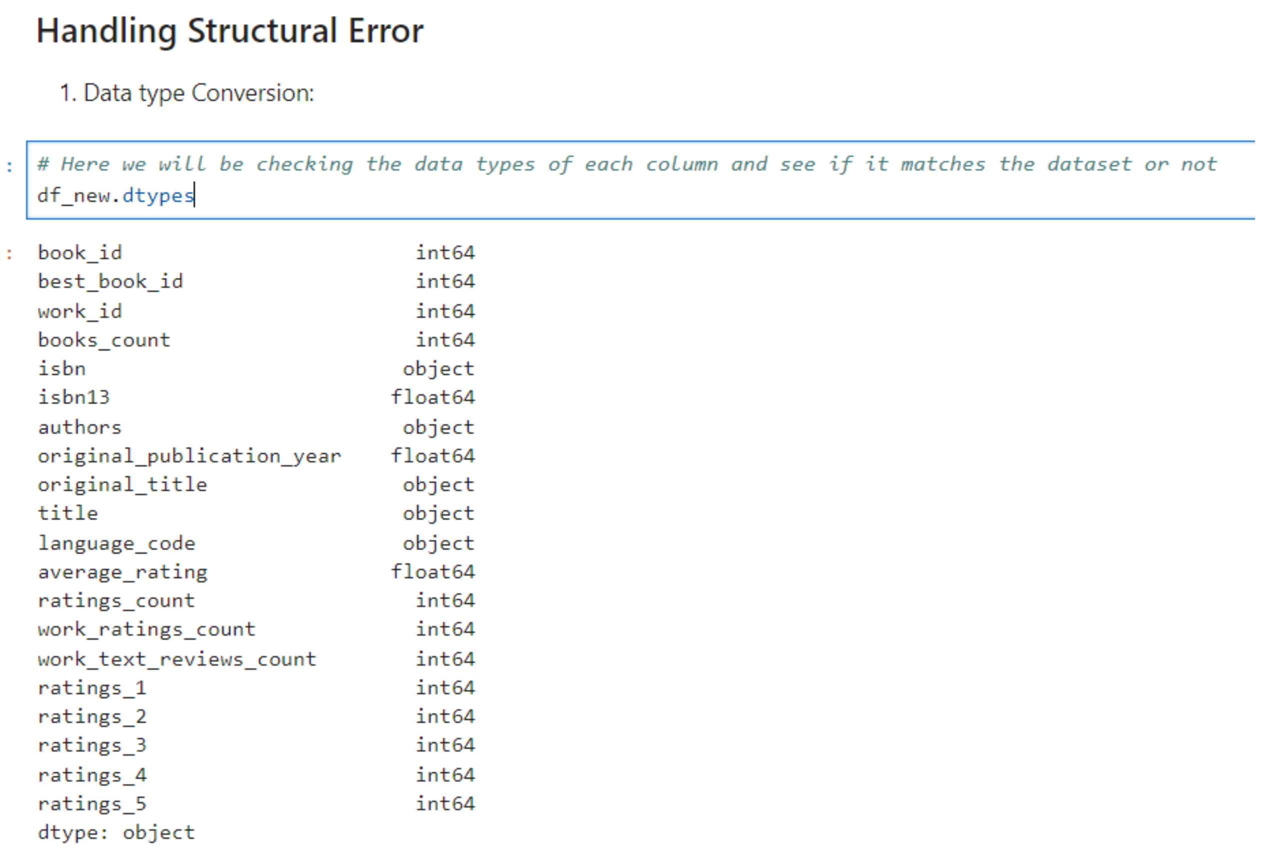
```
df_new = df1.drop(['goodreads_book_id','image_url', 'small_image_url'], axis=1)
df_new
```

	book_id	best_book_id	work_id	books_count	isbn	isbn13	authors	original_publication_year	original_title	title	language_co
0	1	2767052	2792775	272	439023483	9.780439e+12	Suzanne Collins	2008.0	The Hunger Games	The Hunger Games (The Hunger Games, #1)	e
1	2	3	4640799	491	439554934	9.780440e+12	J.K. Rowling, Mary GrandPré	1997.0	Harry Potter and the Philosopher's Stone	Harry Potter and the Sorcerer's Stone (Harry P...	e
2	3	41865	3212258	226	316015849	9.780316e+12	Stephenie Meyer	2005.0	Twilight	Twilight (Twilight, #1)	en-



l_title	title	language_code	average_rating	ratings_count	work_ratings_count	work_text_reviews_count	ratings_1	ratings_2	ratings_3	ratings_4	ratings_5
lunger Games	The Hunger Games (The Hunger Games,	eng	4.34	4780653	4942365	155254	66715	127936	560092	1481305	2706317

Here I was handling the data types of the data, so here we will be checking the data types of each column and see if it matches the dataset or not



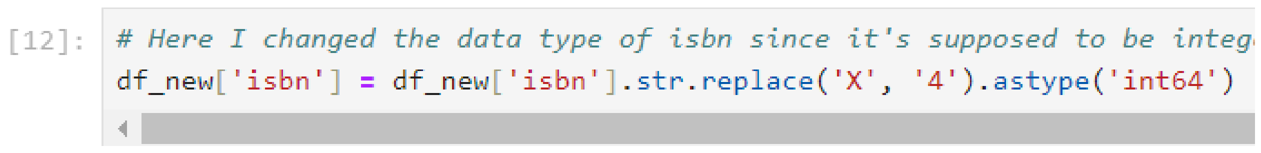
Handling Structural Error

1. Data type Conversion:

```
# Here we will be checking the data types of each column and see if it matches the dataset or not
df_new.dtypes
```

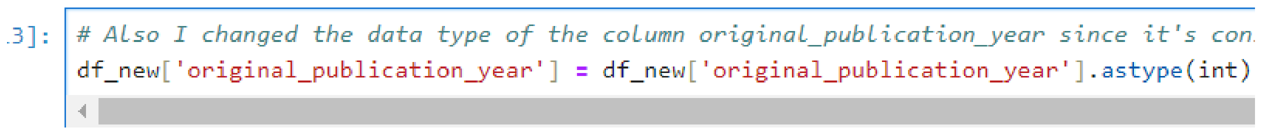
```
book_id                        int64
best_book_id                   int64
work_id                        int64
books_count                    int64
isbn                          object
isbn13                       float64
authors                       object
original_publication_year    float64
original_title                object
title                         object
language_code                 object
average_rating               float64
ratings_count                  int64
work_ratings_count             int64
work_text_reviews_count        int64
ratings_1                      int64
ratings_2                      int64
ratings_3                      int64
ratings_4                      int64
ratings_5                      int64
dtype: object
```

Here I changed the data type of isbn since it's supposed to be integer but previously it was giving it to me as string and I replaced all the X in the column of isbn with 4



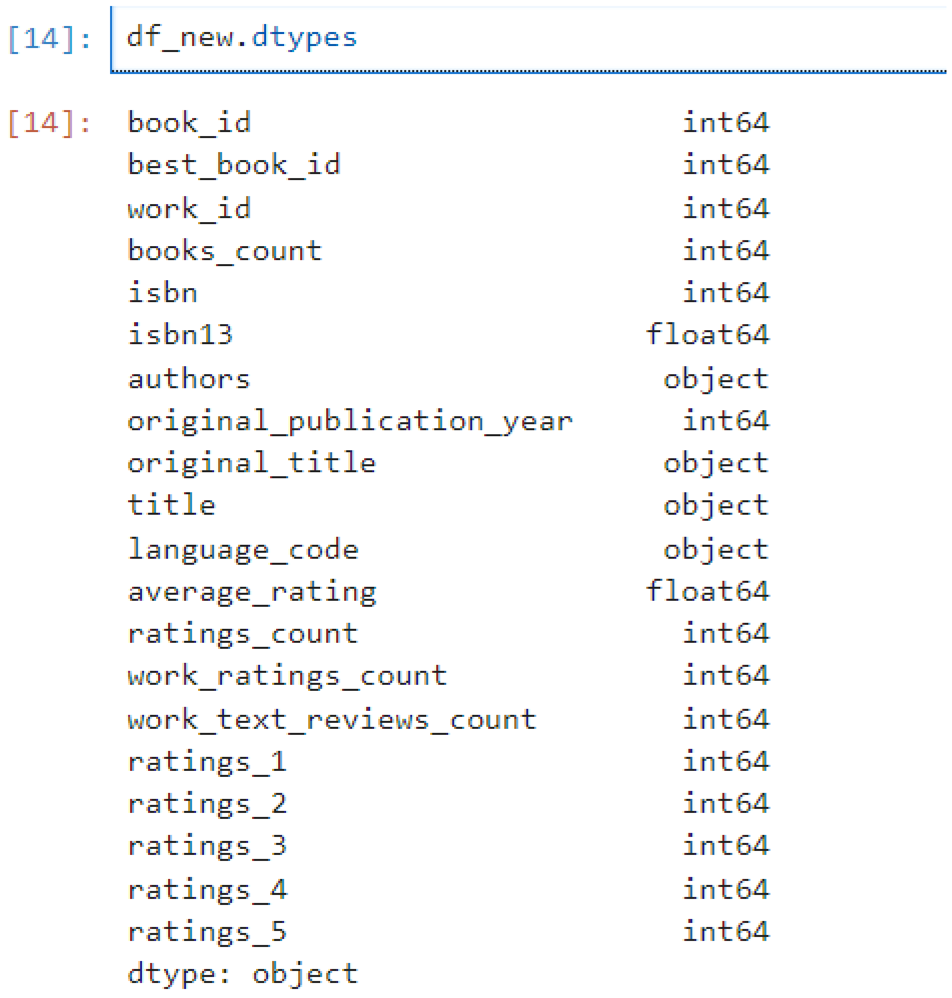
```
[12]: # Here I changed the data type of isbn since it's supposed to be integ
df_new['isbn'] = df_new['isbn'].str.replace('X', '4').astype('int64')
```

And here I also changed the data type of the column original_publication_year since it's considered as float and it should be converted into an integer for better results



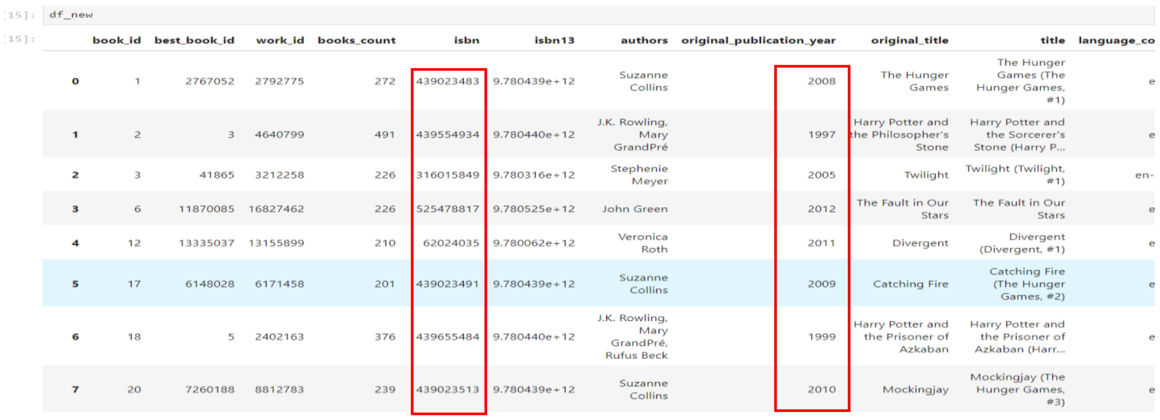
```
3]: # Also I changed the data type of the column original_publication_year since it's con
df_new['original_publication_year'] = df_new['original_publication_year'].astype(int)
```

This is the output to check in our dataset that after we changed the data type of isbn and original_publication_year and as we see down below that now isbn data type is now int64 and in original_publication_year is now int64



```
[14]: df_new.dtypes

[14]: book_id                        int64
      best_book_id                   int64
      work_id                        int64
      books_count                    int64
      isbn                           int64
      isbn13                       float64
      authors                       object
      original_publication_year      int64
      original_title                object
      title                         object
      language_code                 object
      average_rating               float64
      ratings_count                  int64
      work_ratings_count             int64
      work_text_reviews_count        int64
      ratings_1                      int64
      ratings_2                      int64
      ratings_3                      int64
      ratings_4                      int64
      ratings_5                      int64
      dtype: object
```

This is how our dataset looks like after we changed the datatype of isbn and original_publication_year



```
[15]: df_new
```

[15]:

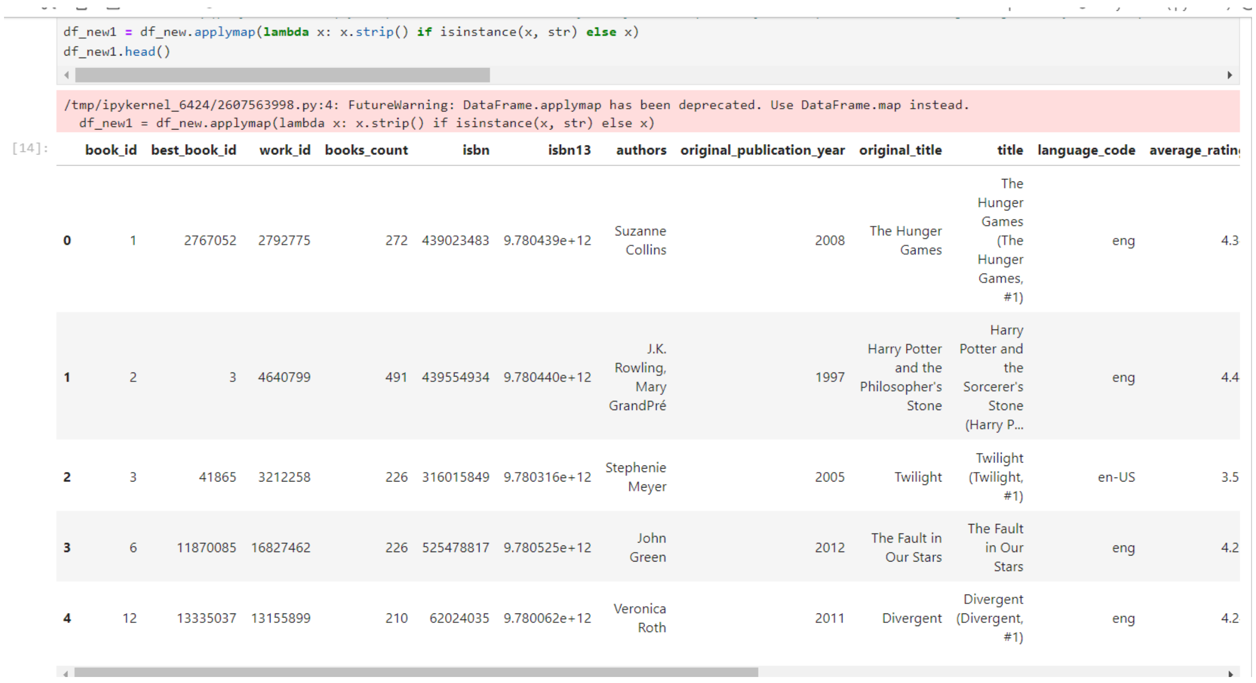
	book_id	best_book_id	work_id	books_count	isbn	isbn13	authors	original_publication_year	original_title	title	language_co
0	1	2767052	2792775	272	439023483	9.780439e+12	Suzanne Collins	2008	The Hunger Games	The Hunger Games (The Hunger Games, #1)	e
1	2	3	4640799	491	439554934	9.780440e+12	J.K. Rowling, Mary GrandPré	1997	Harry Potter and the Philosopher's Stone	Harry Potter and the Sorcerer's Stone (Harry P...	e
2	3	41865	3212258	226	316015849	9.780316e+12	Stephenie Meyer	2005	Twilight	Twilight (Twilight, #1)	en-
3	6	11870085	16827462	226	525478817	9.780525e+12	John Green	2012	The Fault in Our Stars	The Fault in Our Stars	e
4	12	13335037	13155899	210	62024035	9.780062e+12	Veronica Roth	2011	Divergent	Divergent (Divergent, #1)	e
5	17	6148028	6171458	201	439023491	9.780439e+12	Suzanne Collins	2009	Catching Fire	Catching Fire (The Hunger Games, #2)	e
6	18	5	2402163	376	439655484	9.780440e+12	J.K. Rowling, Mary GrandPré, Rufus Beck	1999	Harry Potter and the Prisoner of Azkaban	Harry Potter and the Prisoner of Azkaban (Harr...	e
7	20	7260188	8812783	239	439023513	9.780439e+12	Suzanne Collins	2010	Mockingjay	Mockingjay (The Hunger Games, #3)	e

Here we remove any whitespace.

strip() function is to remove leading and trailing whitespaces from all string columns.

applymap(): This is a pandas DataFrame method that applies a function to every element of the DataFrame.

lambda x: x.strip() if isinstance(x, str) else x: This is an anonymous function (lambda function) that takes a single argument x, which represents each individual element of the DataFrame as it's being processed.. It checks if x is a string using isinstance(x, str). If x is a string, it removes leading and trailing whitespaces using the strip() method (x.strip()). If x is not a string, the function returns x unchanged.



```
df_new1 = df_new.applymap(lambda x: x.strip() if isinstance(x, str) else x)
df_new1.head()
```

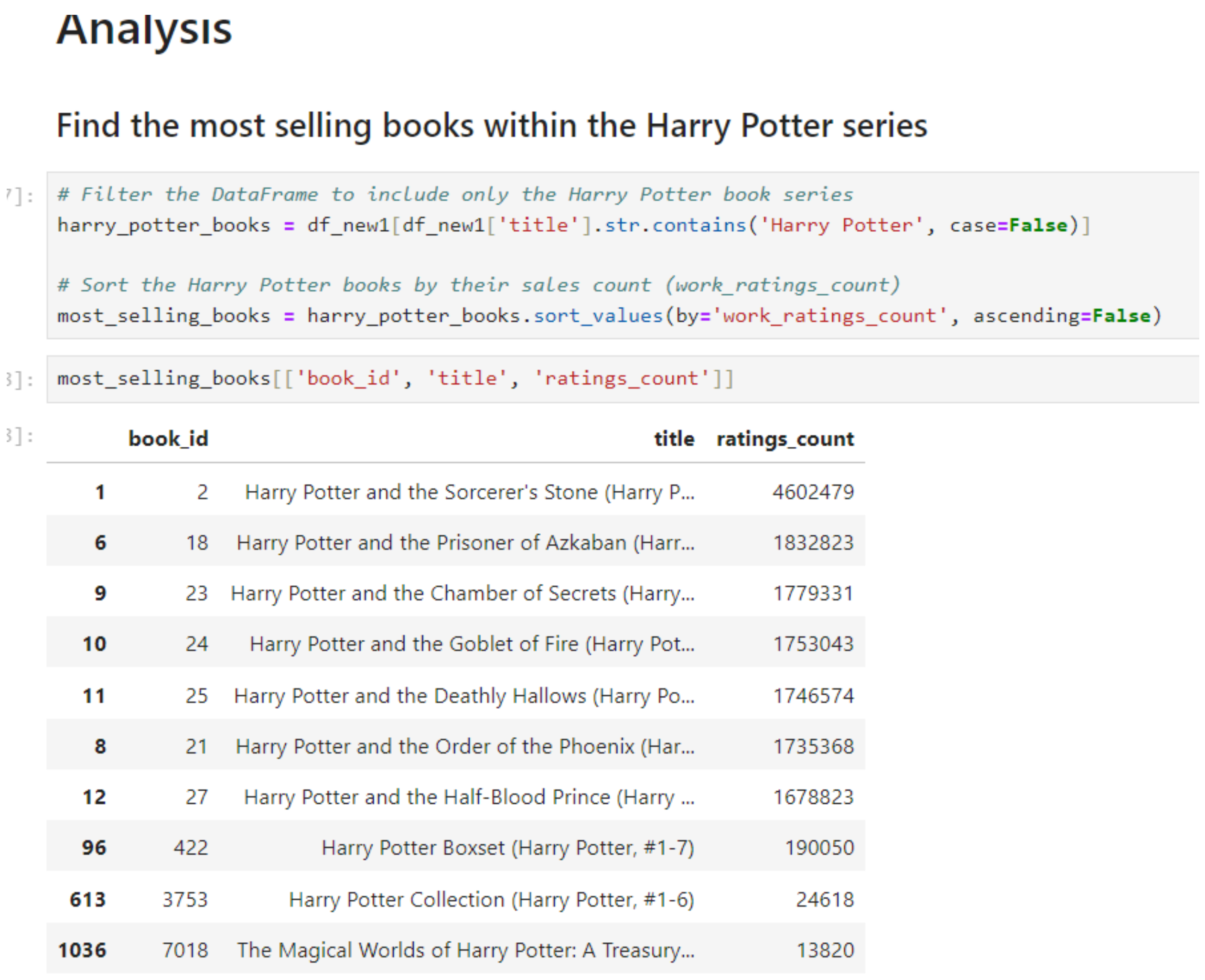
```
/tmp/ipykernel_6424/2607563998.py:4: FutureWarning: DataFrame.applymap has been deprecated. Use DataFrame.map instead.
  df_new1 = df_new.applymap(lambda x: x.strip() if isinstance(x, str) else x)
```

[14]:

	book_id	best_book_id	work_id	books_count	isbn	isbn13	authors	original_publication_year	original_title	title	language_code	average_ratin
0	1	2767052	2792775	272	439023483	9.780439e+12	Suzanne Collins	2008	The Hunger Games	The Hunger Games (The Hunger Games, #1)	eng	4.3
1	2	3	4640799	491	439554934	9.780440e+12	J.K. Rowling, Mary GrandPré	1997	Harry Potter and the Philosopher's Stone	Harry Potter and the Sorcerer's Stone (Harry P...	eng	4.4
2	3	41865	3212258	226	316015849	9.780316e+12	Stephenie Meyer	2005	Twilight	Twilight (Twilight, #1)	en-US	3.5
3	6	11870085	16827462	226	525478817	9.780525e+12	John Green	2012	The Fault in Our Stars	The Fault in Our Stars	eng	4.2
4	12	13335037	13155899	210	62024035	9.780062e+12	Veronica Roth	2011	Divergent	Divergent (Divergent, #1)	eng	4.2

Analysis:

As we see here below the output shows the most selling books of Harry Potter Series in a descending order



Analysis

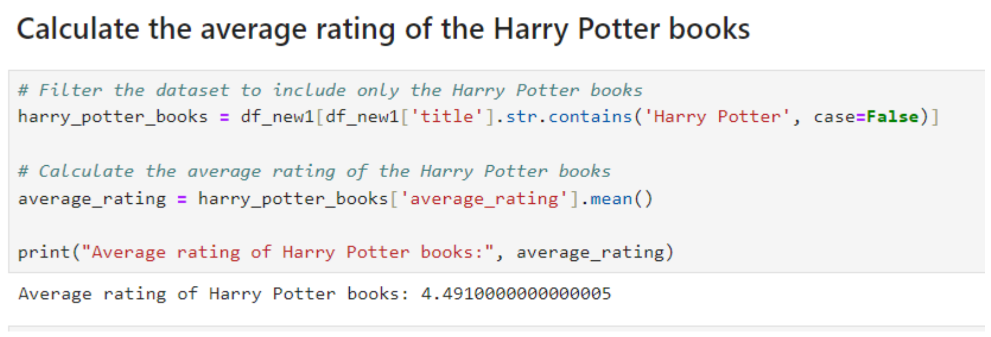
Find the most selling books within the Harry Potter series

```python
# Filter the DataFrame to include only the Harry Potter book series
harry_potter_books = df_new1[df_new1['title'].str.contains('Harry Potter', case=False)]

# Sort the Harry Potter books by their sales count (work_ratings_count)
most_selling_books = harry_potter_books.sort_values(by='work_ratings_count', ascending=False)
```

```python
most_selling_books[['book_id', 'title', 'ratings_count']]
```

	book_id	title	ratings_count
1	2	Harry Potter and the Sorcerer's Stone (Harry P...	4602479
6	18	Harry Potter and the Prisoner of Azkaban (Harr...	1832823
9	23	Harry Potter and the Chamber of Secrets (Harry...	1779331
10	24	Harry Potter and the Goblet of Fire (Harry Pot...	1753043
11	25	Harry Potter and the Deathly Hallows (Harry Po...	1746574
8	21	Harry Potter and the Order of the Phoenix (Har...	1735368
12	27	Harry Potter and the Half-Blood Prince (Harry ...	1678823
96	422	Harry Potter Boxset (Harry Potter, #1-7)	190050
613	3753	Harry Potter Collection (Harry Potter, #1-6)	24618
1036	7018	The Magical Worlds of Harry Potter: A Treasury...	13820

Below shows the output which shows the average ratings of all Harry Potter books



Calculate the average rating of the Harry Potter books

```python
# Filter the dataset to include only the Harry Potter books
harry_potter_books = df_new1[df_new1['title'].str.contains('Harry Potter', case=False)]

# Calculate the average rating of the Harry Potter books
average_rating = harry_potter_books['average_rating'].mean()

print("Average rating of Harry Potter books:", average_rating)
```

```
Average rating of Harry Potter books: 4.4910000000000005
```