Name: Nada Mohamed Mahfouz Zakria Henedy

ID:20221377458

Department: AI

Time: Tuesday at 8:30AM

Data cleaning

Cell1:

importing important libraires that we will use to analysis

```
[4]: import pandas as pd
     import numpy as np
     import matplotlib.pyplot as plt
     import seaborn as sns
```

Cell2:

Reading our data by using read_csv and it will put it in Data Frame

Then start to show the head of the data by default it shows first 5 rows

```
[5]: # reading csv file
     data = pd.read_csv('books.csv')
     # show first 5 rows
     data.head()
```

[5]:

	book_id	goodreads_book_id	best_book_id	work_id	books_count	isbn	isbn13	authors	original_publication_year	original_title	...	ratings_count	wo
0	1	2767052	2767052	2792775	272	439023483	9.780439e+12	Suzanne Collins	2008.0	The Hunger Games	...	4780653	
1	2	3	3	4640799	491	439554934	9.780440e+12	J.K. Rowling, Mary GrandPré	1997.0	Harry Potter and the Philosopher's Stone	...	4602479	
2	3	41865	41865	3212258	226	316015849	9.780316e+12	Stephenie Meyer	2005.0	Twilight	...	3866839	
3	6	11870085	11870085	16827462	226	525478817	9.780525e+12	John Green	2012.0	The Fault in Our Stars	...	2346404	
4	12	13335037	13335037	13155899	210	62024035	9.780062e+12	Veronica Roth	2011.0	Divergent	...	1903563	

5 rows × 23 columns

Cell3: List the names of columns

```
[6]: # to see all the column's name
     data.columns.tolist()
```

```
[6]: ['book_id',
      'goodreads_book_id',
      'best_book_id',
      'work_id',
      'books_count',
      'isbn',
      'isbn13',
      'authors',
      'original_publication_year',
      'original_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',
      'image_url',
      'small_image_url']
```

Cell 4:

To know the shape of the data

```
[7]: # to see the shape of the image
     data.shape
```

```
[7]: (1354, 23)
```

Cell 5:

Because there are a lot of columns that will not affect our analysis so we will remove them to make our analysis simpler and focus on the most important columns that affect our analysis so here we write the columns names we want to be available

code

```
[8]: '''
     we will remove the columns that will not affect our analysis like image url , small image url , best book id
     goodreads book id , work id ,isbn ,rating count , title

     '''
     data = data.loc[:,[
                        'book_id' ,
                        'books_count',
                        'isbn13',
                        'authors',
                        'original_publication_year',
                        'language_code',
                        'average_rating',
                        'title',
                        'ratings_count',
                        'work_ratings_count',
                        'work_text_reviews_count',
                        'ratings_1',
                        'ratings_2',
                        'ratings_3',
                        'ratings_4',
                        'ratings_5',
                        'original_title']]
     data.head()
```

output:

[8]:

	book_id	books_count	isbn13	authors	original_publication_year	language_code	average_rating	title	ratings_count	work_ratings_count	work_text_rev
0	1	272	9.780439e+12	Suzanne Collins	2008.0	eng	4.34	The Hunger Games (The Hunger Games, #1)	4780653	4942365	
1	2	491	9.780440e+12	J.K. Rowling, Mary GrandPré	1997.0	eng	4.44	Harry Potter and the Sorcerer's Stone (Harry P...	4602479	4800065	
2	3	226	9.780316e+12	Stephenie Meyer	2005.0	en-US	3.57	Twilight (Twilight, #1)	3866839	3916824	
3	6	226	9.780525e+12	John Green	2012.0	eng	4.26	The Fault in Our Stars	2346404	2478609	
4	12	210	9.780062e+12	Veronica Roth	2011.0	eng	4.24	Divergent (Divergent, #1)	1903563	2216814	

Cell 6:

our target is to know the title of the book, so we pop it and concatenate it to the data to be the last column

code:

```
[9]: # making the original title as our dependent column
     col = data.pop('original_title')
     data = pd.concat([data, col], axis=1)
```

output:

	book_id	books_count	isbn13	authors	original_publication_year	language_code	average_rating	title	ratings_count	work_ratings_count	work_t
0	1	272	9.780439e+12	Suzanne Collins	2008.0	eng	4.34	The Hunger Games (The Hunger Games, #1)	4780653	4942365	
1	2	491	9.780440e+12	J.K. Rowling, Mary GrandPré	1997.0	eng	4.44	Harry Potter and the Sorcerer's Stone (Harry P...	4602479	4800065	
2	3	226	9.780316e+12	Stephenie Meyer	2005.0	en-US	3.57	Twilight (Twilight, #1)	3866839	3916824	
3	6	226	9.780525e+12	John Green	2012.0	eng	4.26	The Fault in Our Stars	2346404	2478609	
4	12	210	9.780062e+12	Veronica Roth	2011.0	eng	4.24	Divergent (Divergent, #1)	1903563	2216814	
...	...	...	...	...	...	...	...	...	...	...	
1349	9925	52	9.781582e+12	Mary Hoffman	2002.0	eng	3.90	City of Masks (Stravaganza, #1)	12048	13385	
1350	9937	22	9.781596e+12	Caragh M. O'Brien	2012.0	en-US	3.77	Promised (Birthmarked, #3)	11766	12884	

Cell 7:

See the shape of the data after removing unnecessary columns

```
[11]: # to see the shape of the data after removing unnecessary
      data.shape
```

```
[11]: (1354, 17)
```

Cell 8:

To check null values in each column by summing the number of null values in each column

```
[12]: # how many null values we have in each column
      data.isna().sum()
```

```
[12]: book_id                     0
      books_count                 0
      isbn13                     44
      authors                     0
      original_publication_year   3
      language_code             109
      average_rating              0
      title                       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
      original_title             52
      dtype: int64
```

Cell 9:

As we see , there are a few null values in isbn13, original publication year and original title

So, we will drop the null rows in each column

```
[13]: # we can drop the rows of orginal publication year and isbn13 because it have few null values
data = data.dropna(subset = ['original_publication_year'])
data = data.dropna(subset = ['isbn13'])
data = data.dropna(subset = ['original_title'])
```

Cell 10:

In language code , we can replace null rows by the most common string has been used in this column, so we take the mode of the column

```
[24]: # we will replace the value of language code by most frequent use
data['language_code'] = data['language_code'].mode()[0]
```

Cell 11:

Checking the null values in the columns after removing all nulls in the columns

```
[25]: # see the data after removing all the null values
data.isna().sum()
```

```
[25]: book_id                    0
books_count                0
isbn13                     0
authors                    0
original_publication_year  0
language_code              0
average_rating             0
title                      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
original_title             0
dtype: int64
```

Cell 12:

To check about the duplicates in rows

```
[16]: # to check for duplicates
data.duplicated().sum()
```

```
[16]: 0
```

Cell 13:

to check about the inconsistency in the columns we will see the types of the data inside the columns by dtypes

```
[17]: # to check the type of data columns
      data.dtypes
```

```
[17]: book_id                        int64
      books_count                    int64
      isbn13                       float64
      authors                       object
      original_publication_year    float64
      language_code                 object
      average_rating               float64
      title                         object
      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
      original_title                object
      dtype: object
```

Cell 14:

To see more information about the data in the columns like the types , null values

```
[18]: # to see information about columns
      data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 1261 entries, 0 to 1353
Data columns (total 17 columns):
 #   Column                     Non-Null Count  Dtype
---  ------                     --------------  -----
 0   book_id                    1261 non-null   int64
 1   books_count                1261 non-null   int64
 2   isbn13                     1261 non-null   float64
 3   authors                    1261 non-null   object
 4   original_publication_year  1261 non-null   float64
 5   language_code              1261 non-null   object
 6   average_rating             1261 non-null   float64
 7   title                      1261 non-null   object
 8   ratings_count              1261 non-null   int64
 9   work_ratings_count         1261 non-null   int64
 10  work_text_reviews_count    1261 non-null   int64
 11  ratings_1                  1261 non-null   int64
 12  ratings_2                  1261 non-null   int64
 13  ratings_3                  1261 non-null   int64
 14  ratings_4                  1261 non-null   int64
 15  ratings_5                  1261 non-null   int64
 16  original_title             1261 non-null   object
dtypes: float64(3), int64(10), object(4)
memory usage: 177.3+ KB
```

Cell 15:

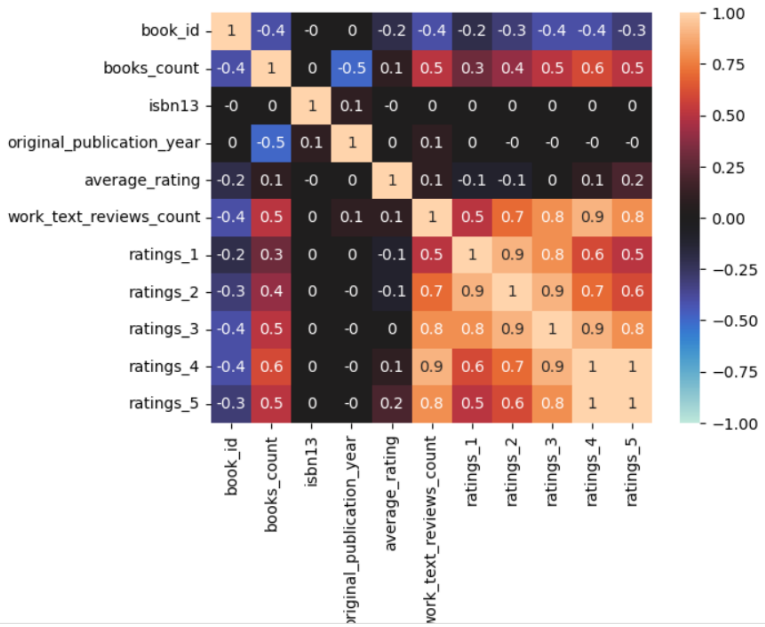
To see the correlation between columns by heatmap that visualize this relation

We customize the bar that max is 1 , min -1 and center 0

When the correlation is closed to 1 or -1 then this considered as high correlated columns

```
[38]: # to check about correlation between columns
      sns.heatmap(data.corr(numeric_only = True).round(1), annot=True, vmax= 1 , vmin=-1 , center = 0 )
```

```
[38]: <Axes: >
```



Cell 16:

here we select from the heatmap the highest correlated columns that exceeds certain threshold

```
[28]: # to select the highest correlation
      highest = data.corr(numeric_only = True)
      highest = highest.unstack()
      highest = highest[abs(highest) >= 0.97]
      highest = highest[abs(highest)<1]
      print(highest)
```

```
ratings_count       work_ratings_count    0.998814
                    ratings_4             0.986152
                    ratings_5             0.974711
work_ratings_count  ratings_count         0.998814
                    ratings_4             0.989610
                    ratings_5             0.976314
ratings_4           ratings_count         0.986152
                    work_ratings_count    0.989610
ratings_5           ratings_count         0.974711
                    work_ratings_count    0.976314
dtype: float64
```

Cell 17:

As we see , work ratings count, and rating count are the most correlated columns so we will drop them because there are other columns do the same function as them

```
[29]: # we will drop the highest corrlated columns
      data.drop(['work_ratings_count'] , axis = 1 ,inplace = True)
      data.drop(['ratings_count'] , axis = 1 ,inplace = True)
```

Cell 18:

After removing high correlated columns, we will check if there are other columns have high correlation or not

```
[30]: # to see the result after dropping highest correlated columns
      highest = data.corr(numeric_only = True)
      highest = highest.unstack()
      highest = highest[abs(highest) >= 0.97]
      highest = highest[abs(highest)<1]
      print(highest)
```

```
Series([], dtype: float64)
```

Data analysis for Harry Potter series

Cell 19:

Here after we finish our cleaning for data

We will start choosing harry potter books by first selecting the author then get the rows that author is inside them

Then accessing the names of the books from title column and get the whole row that have the author's name and his written about harry potter

```
[31]: # to get the author name for harry potter series
      author = [i for i in data.authors.unique() if i.find('J.K. Rowling') != -1]
      # to get the books that have been written by the author
      author_books = data.loc[data.authors.isin(author)]
      # get harry poter books that have been written by the author
      harry_potter = [i for i in author_books.title.unique() if i.find('Harry Potter and ') != -1]
      # to get the rows that have harry potter books
      harry_potter_books = author_books.loc[author_books.title.isin(harry_potter)]
      harry_potter_books
```

[31]:

	book_id	books_count	isbn13	authors	original_publication_year	language_code	average_rating	title	work_text_reviews_count	ratings_1	ratings_2	r
1	2	491	9.780440e+12	J.K. Rowling, Mary GrandPré	1997.0	eng	4.44	Harry Potter and the Sorcerer's Stone (Harry P...	75867	75504	101676	
6	18	376	9.780440e+12	J.K. Rowling, Mary GrandPré, Rufus Beck	1999.0	eng	4.53	Harry Potter and the Prisoner of Azkaban (Harr...	36099	6716	20413	
8	21	307	9.780439e+12	J.K. Rowling, Mary GrandPré	2003.0	eng	4.46	Harry Potter and the Order of the Phoenix (Har...	28685	9528	31577	

Cell 20:

Get the shape of the data after selecting harry potter books

```
[32]: # to know the shape of the data
      harry_potter_books.shape
```

```
[32]: (7, 15)
```

Cell 21:

we want to see the most selling harry potter books

so, we will know by books count that have the total editions of each book

we will group book title with books count and ordering it in descending order to get the highest one at the top

```
[34]: # to show the most selling harry potter books in descending order
      grouped = harry_potter_books['books_count'].groupby(harry_potter_books['original_title']).sum().reset_index()
      grouped.sort_values(['books_count'],ascending = False)
```

[34]:

	original_title	books_count
5	Harry Potter and the Philosopher's Stone	491
0	Harry Potter and the Chamber of Secrets	398
6	Harry Potter and the Prisoner of Azkaban	376
2	Harry Potter and the Goblet of Fire	332
4	Harry Potter and the Order of the Phoenix	307
3	Harry Potter and the Half-Blood Prince	275
1	Harry Potter and the Deathly Hallows	263

Cell 22:

Here we extract that name of the book that have the most selling

```
[35]: #the most selling book within the Harry Potter series.
      harry_potter_books['original_title'].loc[harry_potter_books['books_count'].idxmax()]
```

```
[35]: "Harry Potter and the Philosopher's Stone"
```

Cell 23:

We will calculate the weighted average of the average rating by taking the mean of the column

```
[36]: #calculate the average rating of all Harry Potter books.
      harry_potter_books.loc[:, 'average_rating'].mean()
```

```
[36]: 4.497142857142857
```

Cell 24:

To visualize the books count in bar plot graph

```
[37]: # barplot to see different number of editions in different type of harry potter books
      plt.barh(harry_potter_books['original_title'],harry_potter_books['books_count'])
```

```
[37]: <BarContainer object of 7 artists>
```