Cloud Report

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.impute import SimpleImputer
```

[401] Python

Importing libraries

```
df = pd.read_csv('books.csv')
df.head()
```

[402] Python

	book_id	goodreads_book_id	best_book_id	work_id	books_count	isbn	isbn13	authors	original_
0	1	2767052	2767052	2792775	272	439023483	9.780439e+12	Suzanne Collins	
1	2	3	3	4640799	491	439554934	9.780440e+12	J.K. Rowling, Mary GrandPré	
2	3	41865	41865	3212258	226	316015849	9.780316e+12	Stephenie Meyer	
3	6	11870085	11870085	16827462	226	525478817	9.780525e+12	John Green	
4	12	13335037	13335037	13155899	210	62024035	9.780062e+12	Veronica Roth	

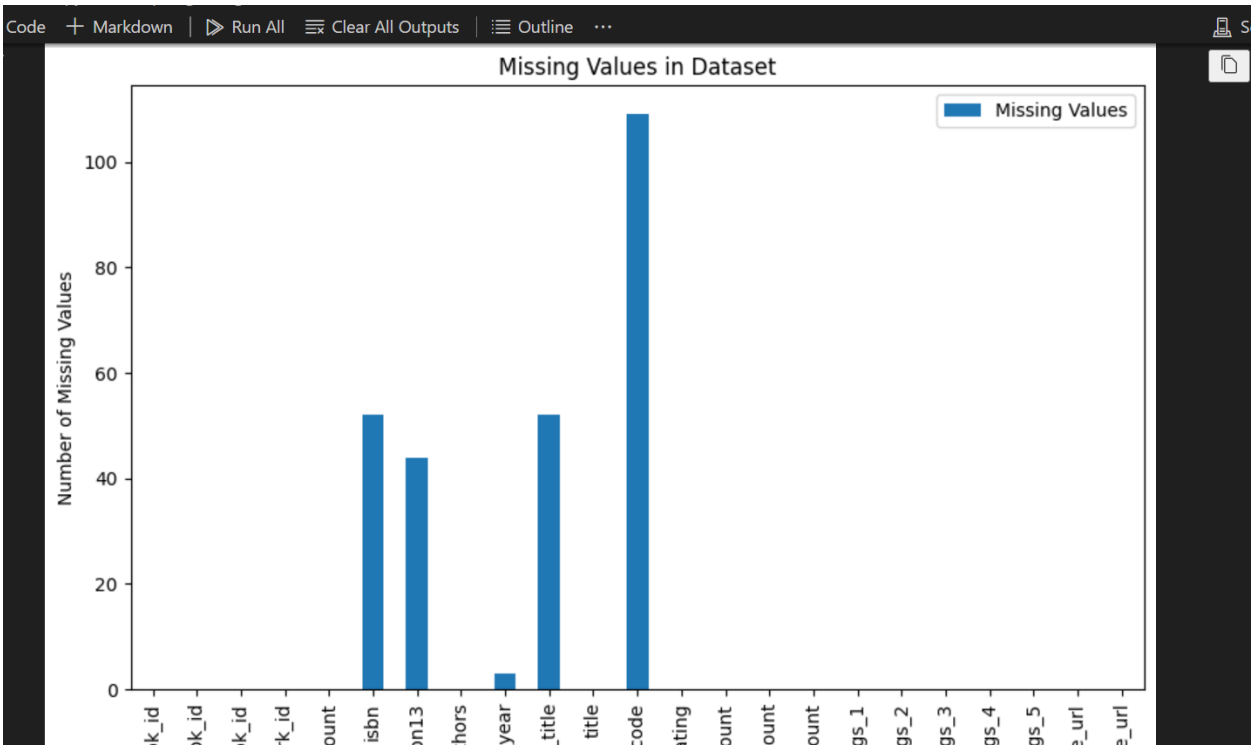
5 rows × 23 columns

Reading our dataset

```
missing_values = pd.DataFrame(df.isnull().sum(), columns=['Missing Values'])
missing_values
```

	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

Here we see the empty values in our dataset



Plotting missing values in our dataset

```python
num_cols = df.select_dtypes(include=['int64', 'float64']).columns
categorical_cols = df.select_dtypes(include=['object']).columns


imputer_numeric = SimpleImputer(strategy='mean')
df[num_cols] = imputer_numeric.fit_transform(df[num_cols])


imputer_categorical = SimpleImputer(strategy='most_frequent')
df[categorical_cols] = imputer_categorical.fit_transform(df[categorical_cols])


print("Missing Values After Imputation:")
print(df.isnull().sum().to_string())
```

Python

Here we impute our empty values in the numeric columns with strategy mean

And we impute our empty values in the categorical columns with strategy mean

Output:


Code
Markdown
Run All
Clear All Outputs
Outline
Missing Values After Imputation:
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

```
duplicates = df.duplicated().sum()
duplicates
```

Python

```
0
```

Check if there is duplicates in our dataset

```
descriptive_stats = df[num_cols].describe()
descriptive_stats
```

Python

	book_id	goodreads_book_id	best_book_id	work_id	books_count	isbn13	original_publication
count	1354.000000	1.354000e+03	1.354000e+03	1.354000e+03	1354.000000	1.354000e+03	1354.0(
mean	4453.584195	5.951852e+06	6.120589e+06	8.707028e+06	50.330871	9.766700e+12	2003.4
std	2894.277455	6.664595e+06	6.935008e+06	9.813696e+06	61.338867	3.513506e+11	16.7(
min	1.000000	1.000000e+00	1.000000e+00	1.150000e+02	1.000000	7.678361e+10	1868.0(
25%	1860.250000	1.537868e+05	1.537962e+05	1.375035e+06	22.000000	9.780142e+12	2003.0(
50%	4177.500000	3.305318e+06	3.422646e+06	4.005716e+06	37.000000	9.780440e+12	2008.0(
75%	6814.500000	9.917380e+06	1.019388e+07	1.435717e+07	58.000000	9.780804e+12	2011.0(
max	9955.000000	3.207567e+07	3.360215e+07	4.963819e+07	1314.000000	9.788424e+12	2017.0(

Show our descriptive statistics of our numeric columns

```
df_cleaned = pd.DataFrame()
df_cleaned = df[df[categorical_cols].apply(lambda x: x.str.contains('Harry Potter', na=False)).any
(axis=1)]
harry_potter_books = df_cleaned.iloc[:-1]
harry_potter_books.head()
```

Python

Here we filter our data to contain harry potter books only

Output:

	book_id	goodreads_book_id	best_book_id	work_id	books_count	isbn	isbn13	authors	origin
1	2.0	3.0	3.0	4640799.0	491.0	439554934	9.780440e+12	J.K. Rowling, Mary GrandPré	
6	18.0	5.0	5.0	2402163.0	376.0	043965548X	9.780440e+12	J.K. Rowling, Mary GrandPré, Rufus Beck	
8	21.0	2.0	2.0	2809203.0	307.0	439358078	9.780439e+12	J.K. Rowling, Mary GrandPré	
9	23.0	15881.0	15881.0	6231171.0	398.0	439064864	9.780439e+12	J.K. Rowling, Mary GrandPré	
10	24.0	6.0	6.0	3046572.0	332.0	439139600	9.780439e+12	J.K. Rowling, Mary	

```
409]  harry_potter_books['original_title']
```

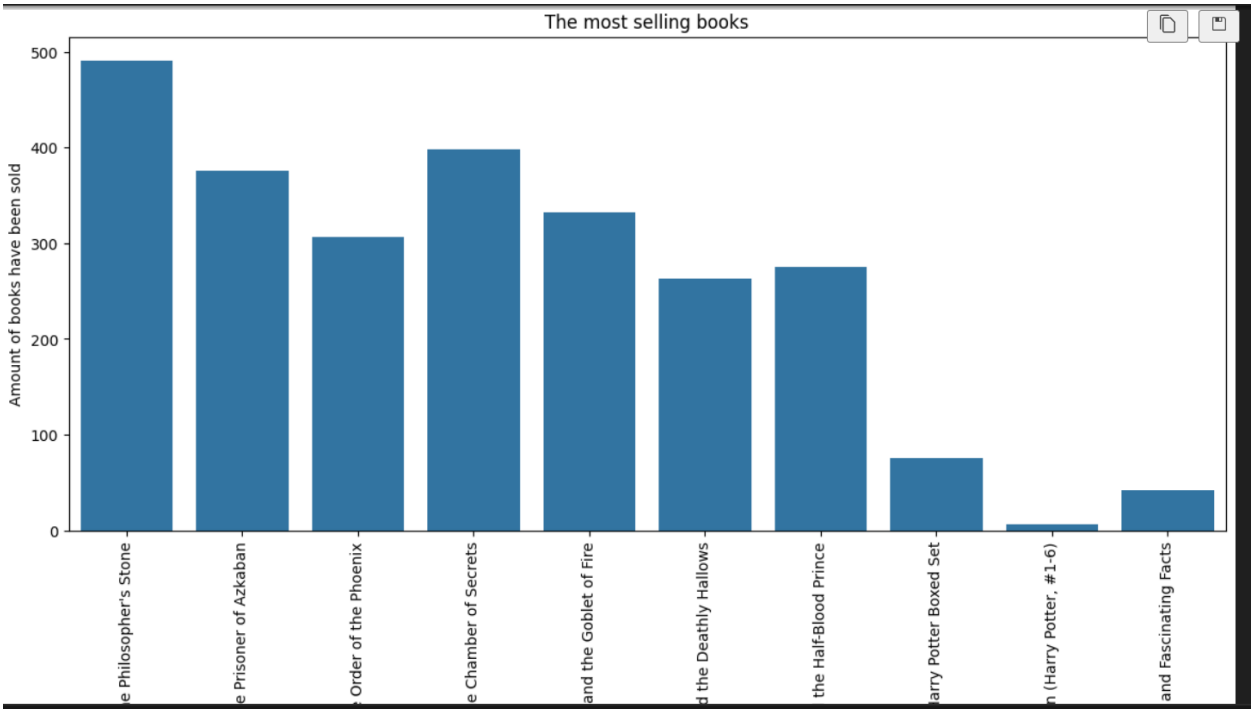
```
1                  Harry Potter and the Philosopher's Stone
6                  Harry Potter and the Prisoner of Azkaban
8                Harry Potter and the Order of the Phoenix
9                  Harry Potter and the Chamber of Secrets
10                     Harry Potter and the Goblet of Fire
11                   Harry Potter and the Deathly Hallows
12                 Harry Potter and the Half-Blood Prince
96                           Complete Harry Potter Boxed Set
613          Harry Potter Collection (Harry Potter, #1-6)
1036    The Magical Worlds of Harry Potter: A Treasury...
Name: original_title, dtype: object
```

Harry potter books title

```python
plt.figure(figsize=(14, 6))
sns.barplot(x=harry_potter_books['original_title'], y=harry_potter_books['books_count'])
plt.title("The most selling books")
plt.xlabel("Columns")
plt.ylabel("Amount of books have been sold")
plt.xticks(rotation=90)
plt.show()
```

Python

Here we plot the most harry potter selled books



```python
ratings_columns = ['ratings_1', 'ratings_2', 'ratings_3', 'ratings_4','ratings_5']


harry_potter_books['Average_Rating'] = harry_potter_books[ratings_columns].mean(axis=1)


harry_potter_books['Average_Rating'].head()
```

12] Python

```
C:\Users\hazem\AppData\Local\Temp\ipykernel_17140\1978471561.py:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.htm
  harry_potter_books['Average_Rating'] = harry_potter_books[ratings_columns].mean(axis=1)

1     960013.0
6     393875.0
8     368109.6
9     381239.8
10    373728.4
```

Create new column named Average rating to make analysis on it

```python
plt.figure(figsize=(14, 6))
sns.barplot(x=harry_potter_books['Average_Rating'], y=harry_potter_books['books_count'])
plt.title("Average rating of most selled books")
plt.xlabel("Average rating")
plt.ylabel("Sold books")
plt.xticks(rotation=90)
plt.show()
```

] Python

Ploting The Average rating of our books with the most sold books

```python
correlation_matrix = pd.DataFrame(harry_potter_books[ha_num_cols].corr())

correlation_with_books_count = pd.DataFrame(correlation_matrix['books_count'].sort_values
(ascending=False))

print("Correlation with books_count:")
print(correlation_with_books_count)
print(50*'*')
```

416] Python

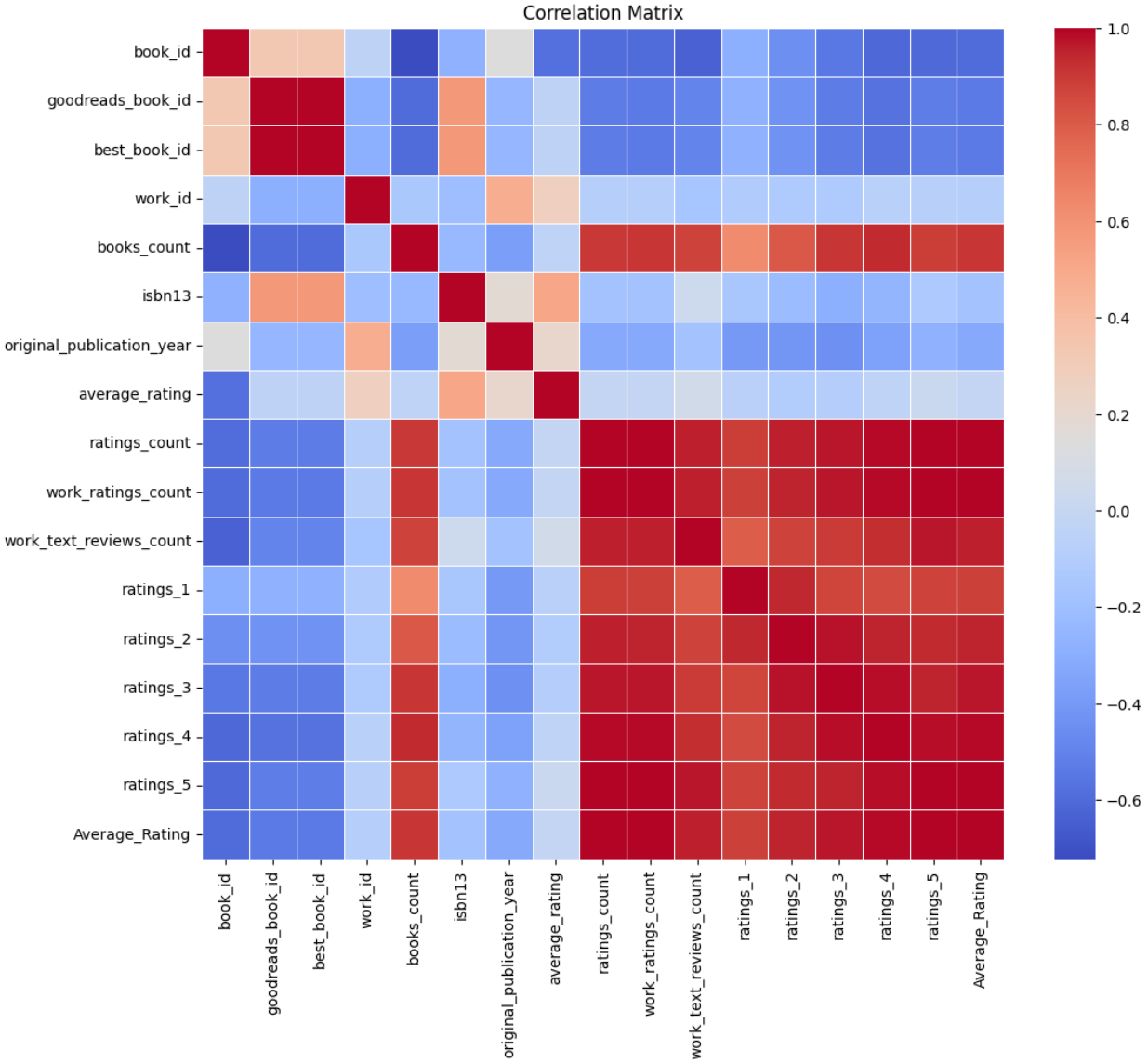
Create correlation matrix with most books sold if number bigger than 0.5 means strong relationship if less than 0.5 means weak relationship.

```
Correlation with books_count:
                            books_count
books_count                    1.000000
ratings_4                      0.936038
ratings_3                      0.908513
Average_Rating                 0.906421
work_ratings_count             0.906421
ratings_count                  0.900125
ratings_5                      0.891211
work_text_reviews_count        0.876589
ratings_2                      0.807796
ratings_1                      0.623421
average_rating                -0.037882
work_id                       -0.138068
isbn13                        -0.231655
original_publication_year     -0.374821
goodreads_book_id             -0.596628
best_book_id                  -0.596628
book_id                       -0.721804
**************************************************
```

```python
plt.figure(figsize=(12, 10))
sns.heatmap(correlation_matrix, cmap='coolwarm', linewidths=0.5)
plt.title('Correlation Matrix')
plt.show()
```

[417] Python

Plotting heat map to see our correlation matrix


Correlation Matrix
book_id
goodreads_book_id
best_book_id
work_id
books_count
isbn13
original_publication_year
average_rating
ratings_count
work_ratings_count
work_text_reviews_count
ratings_1
ratings_2
ratings_3
ratings_4
ratings_5
Average_Rating
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6