DATA EXPLORATION

1. Read the dataset into a dataframe and print its shape
2. Check for invalid values in the dataset
3. Know the data types of variables
4. Describe the data
5. Make Histograms and Box-Plots and look for outliers

In [1]:

```
# importing libraries for the project
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
from scipy.stats import norm
from sklearn import preprocessing
```

Import Data

In [2]:

```
#Load the csv file as Pandas dataframe and check its shape
#Note the warning below: The data contains few erroneous rows that have extra values
#read_csv function skips these erroneous cases from our dataframe df.
#The original csv file

file_path = ("C:/Users/steph/OneDrive/Desktop/data Analyst DSTI/Python_Machine_Labs/
```

In [3]:

```
df = pd.read_csv(file_path,sep = ",", error_bad_lines=False)
print("The data contains {0} Rows and {1} Columns".format(df.shape[0],df.shape[1]))
```

```
The data contains 11123 Rows and 12 Columns
```

```
C:\Users\steph\AppData\Local\Temp/ipykernel_1044/2489409688.py:1: FutureWarning: The
error_bad_lines argument has been deprecated and will be removed in a future versio
n. Use on_bad_lines in the future.


  df = pd.read_csv(file_path,sep = ",", error_bad_lines=False)
b'Skipping line 3350: expected 12 fields, saw 13\nSkipping line 4704: expected 12 fi
elds, saw 13\nSkipping line 5879: expected 12 fields, saw 13\nSkipping line 8981: ex
pected 12 fields, saw 13\n'
```

A. Check the firsts rows and look at the columns names

In [4]:

```
#Let's look at the first 5 rows of the data
#We do see the 12 column names and clearly J.K. Rowling's Harry Potter books!
df.head()
```

Out[4]:

bookID	title	authors	average_rating	isbn	isbn13	language_code	num

	bookID	title	authors	average_rating	isbn	isbn13	language_code	num
0	1	Harry Potter and the Half-Blood Prince (Harry ...	J.K. Rowling/Mary GrandPré	4.57	0439785960	9.780440e+12	eng	
1	2	Harry Potter and the Order of the Phoenix (Har...	J.K. Rowling/Mary GrandPré	4.49	0439358078	9.780439e+12	eng	
2	4	Harry Potter and the Chamber of Secrets (Harry...	J.K. Rowling	4.42	0439554896	9.780440e+12	eng	
3	5	Harry Potter and the Prisoner of Azkaban (Harr...	J.K. Rowling/Mary GrandPré	4.56	043965548X	9.780440e+12	eng	
4	8	Harry Potter Boxed Set Books 1-5 (Harry Potte...	J.K. Rowling/Mary GrandPré	4.78	0439682584	9.780440e+12	eng	

In [5]:

```
df.describe()
```

Out[5]:

	average_rating	isbn13	num_pages	ratings_count	text_reviews_count
count	11094.000000	1.109400e+04	11094.000000	1.109400e+04	11094.000000
mean	3.935026	9.759826e+12	336.543537	1.798750e+04	543.304309
std	0.346458	4.435532e+11	241.313733	1.126427e+05	2579.856004
min	0.000000	8.987060e+09	0.000000	0.000000e+00	0.000000
25%	3.770000	9.780345e+12	192.000000	1.050000e+02	9.000000
50%	3.960000	9.780582e+12	299.000000	7.490000e+02	47.000000
75%	4.140000	9.780872e+12	416.000000	5.018750e+03	238.000000
max	5.000000	9.790008e+12	6576.000000	4.597666e+06	94265.000000

In [6]:

```
# print column names
print("Column names: {0}".format(list(df.columns)))
```

```
Column names: ['bookID', 'title', 'authors', 'average_rating', 'isbn', 'isbn13', 'la
nguage_code', '  num_pages', 'ratings_count', 'text_reviews_count', 'publication_dat
e', 'publisher;;;']
```

In [7]:

```
# Rename the columns "  num_pages", "publisher;;;"
df.rename(columns={"  num_pages":"num_pages", "publisher;;;":"publisher"}, inplace =
```

In [8]:

```
df["publisher"] = df["publisher"].str.replace(";;;", "")
```

In [9]:

```
df.head()
```

Out[9]:

	bookID	title	authors	average_rating	isbn	isbn13	language_code	num
0	1	Harry Potter and the Half-Blood Prince (Harry ...	J.K. Rowling/Mary GrandPré	4.57	0439785960	9.780440e+12	eng	
1	2	Harry Potter and the Order of the Phoenix (Har...	J.K. Rowling/Mary GrandPré	4.49	0439358078	9.780439e+12	eng	
2	4	Harry Potter and the Chamber of Secrets (Harry...	J.K. Rowling	4.42	0439554896	9.780440e+12	eng	
3	5	Harry Potter and the Prisoner of Azkaban (Harr...	J.K. Rowling/Mary GrandPré	4.56	043965548X	9.780440e+12	eng	
4	8	Harry Potter Boxed Set Books 1-5 (Harry Potte...	J.K. Rowling/Mary GrandPré	4.78	0439682584	9.780440e+12	eng	

B. Exploring the data for missing values

In [10]:

```
# check if the data has missing values
df.isna().sum()
```

Out[10]:

```
bookID                 0
title                 29
authors               29
average_rating        29
isbn                  29
isbn13                29
language_code         29
num_pages             29
ratings_count         29
text_reviews_count    29
publication_date      29
publisher             29
dtype: int64
```

In [11]:

```
freq=(df.isna().sum()/len(df))*100
print(freq)
```

```
bookID                0.000000
title                 0.260721
authors               0.260721
average_rating        0.260721
isbn                  0.260721
isbn13                0.260721
language_code         0.260721
num_pages             0.260721
ratings_count         0.260721
text_reviews_count    0.260721
publication_date      0.260721
publisher             0.260721
dtype: float64
```

In [12]:

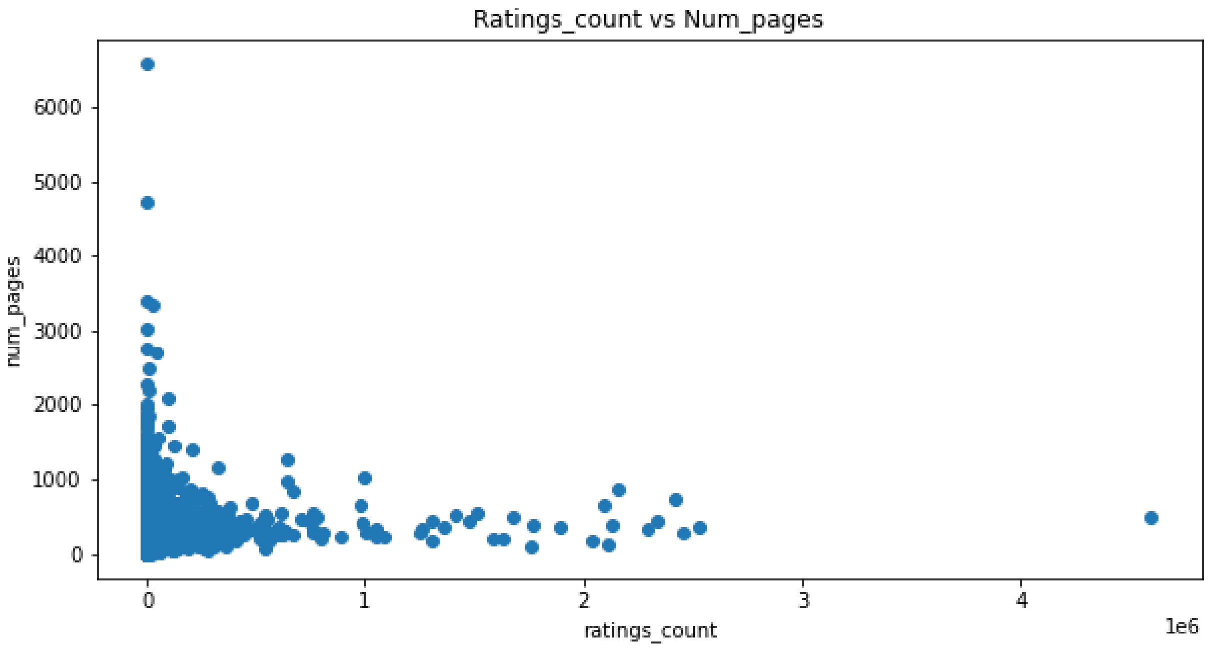
```
# Let's see the data type of columns in the dataframe
df.dtypes
```

Out[12]:

```
bookID                 object
title                  object
authors                object
average_rating        float64
isbn                   object
isbn13                float64
language_code          object
num_pages             float64
ratings_count         float64
text_reviews_count    float64
publication_date       object
publisher              object
dtype: object
```

In [13]:

```
plt.figure(figsize=(10,5))
plt.scatter(df.ratings_count,df.num_pages)
plt.title("Ratings_count vs Num_pages")
plt.xlabel("ratings_count")
plt.ylabel("num_pages")
plt.show()
```



In [14]:
```
df.describe()
```

Out[14]:

	average_rating	isbn13	num_pages	ratings_count	text_reviews_count
count	11094.000000	1.109400e+04	11094.000000	1.109400e+04	11094.000000
mean	3.935026	9.759826e+12	336.543537	1.798750e+04	543.304309
std	0.346458	4.435532e+11	241.313733	1.126427e+05	2579.856004
min	0.000000	8.987060e+09	0.000000	0.000000e+00	0.000000
25%	3.770000	9.780345e+12	192.000000	1.050000e+02	9.000000
50%	3.960000	9.780582e+12	299.000000	7.490000e+02	47.000000
75%	4.140000	9.780872e+12	416.000000	5.018750e+03	238.000000
max	5.000000	9.790008e+12	6576.000000	4.597666e+06	94265.000000

In [15]:
```
# regarding the proportion of na values in the database we can remove them
df = df.dropna()
```

In [16]:
```
df.isna().sum()
```

Out[16]:
```
bookID                0
title                 0
authors               0
average_rating        0
isbn                  0
isbn13                0
language_code         0
num_pages             0
ratings_count         0
text_reviews_count    0
publication_date      0
publisher             0
dtype: int64
```

In [17]:
```
print("The data contains {0} Rows and {1} Columns".format(df.shape[0],df.shape[1]))
```

```
The data contains 11094 Rows and 12 Columns
```

C. Exploring variables to understand the data better.

In [18]:

```
df.language_code.value_counts()
```

Out[18]:

```
eng      8885
en-US    1403
spa       218
en-GB     213
fre       144
ger        99
jpn        46
mul        19
zho        14
grc        11
por        10
en-CA       7
ita         5
enm         3
lat         3
swe         2
rus         2
srp         1
nl          1
msa         1
glg         1
wel         1
ara         1
nor         1
tur         1
gla         1
ale         1
Name: language_code, dtype: int64
```

In [19]:

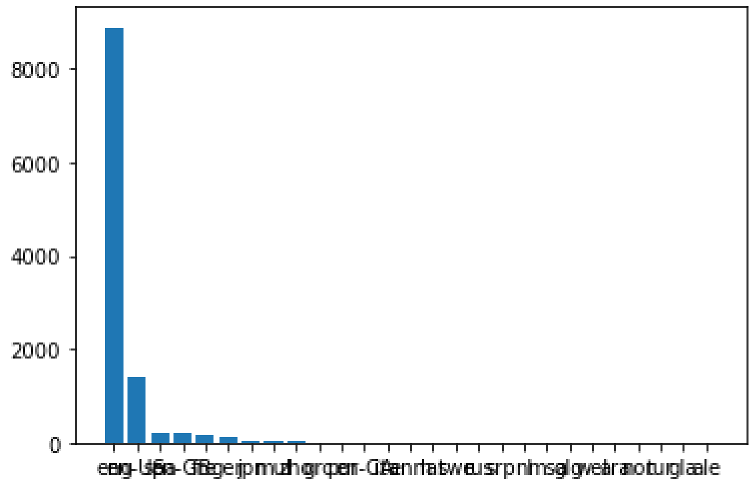
```
df.groupby(["num_pages","language_code"]).mean().ratings_count
```

Out[19]:

```
num_pages  language_code
0.0        en-GB             1308.666667
           en-US               46.181818
           eng                419.500000
           fre                 17.000000
           ger                  2.000000
                                 ...
3020.0     eng               2734.000000
3342.0     eng              28242.000000
3400.0     eng                  6.000000
4736.0     eng               1493.000000
6576.0     eng               1338.000000
Name: ratings_count, Length: 2030, dtype: float64
```

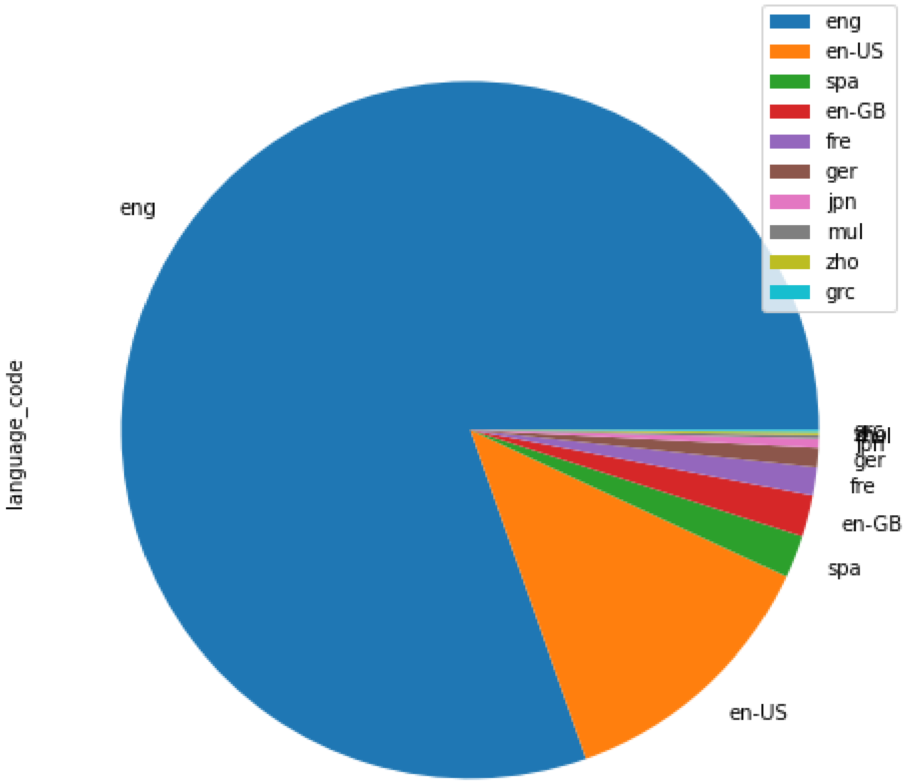
In [20]:

```
plt.bar(x=df.language_code.value_counts().index,height=df.language_code.value_counts
plt.show()
```



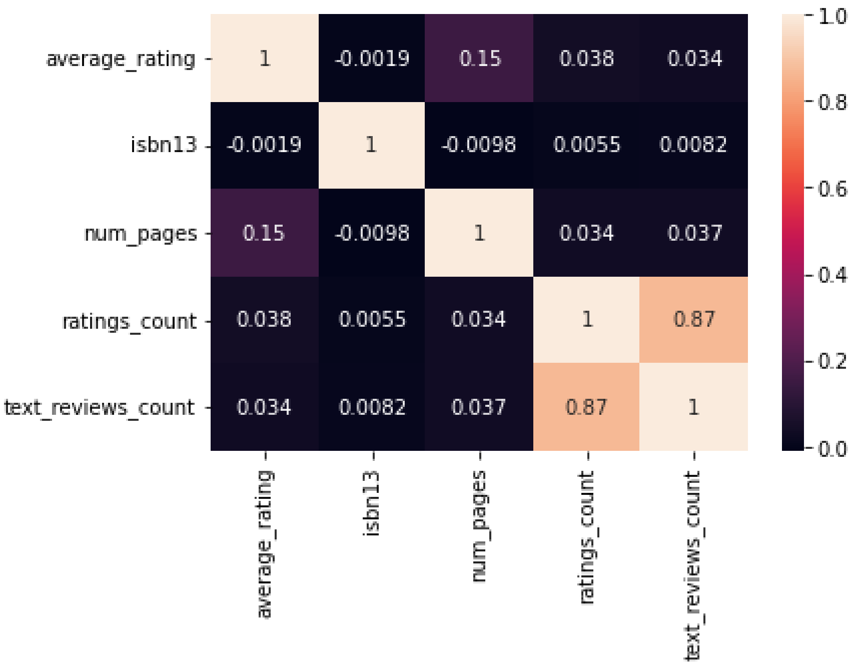
In [21]:

```
df['language_code'].value_counts().head(10).plot(kind = 'pie', figsize=(8, 8)).legen
plt.show()
```



In [22]:

```
corrMatrix = df.corr()
sns.heatmap(corrMatrix, annot=True)
plt.show()
```



D. Exploring the continuous variables in data

1. Build histograms on numerical variables

In [24]:

```
# check the continuous varibles description
continuousVars = ['average_rating', 'isbn13','num_pages','ratings_count', 'text_revi
df[continuousVars].describe()
```

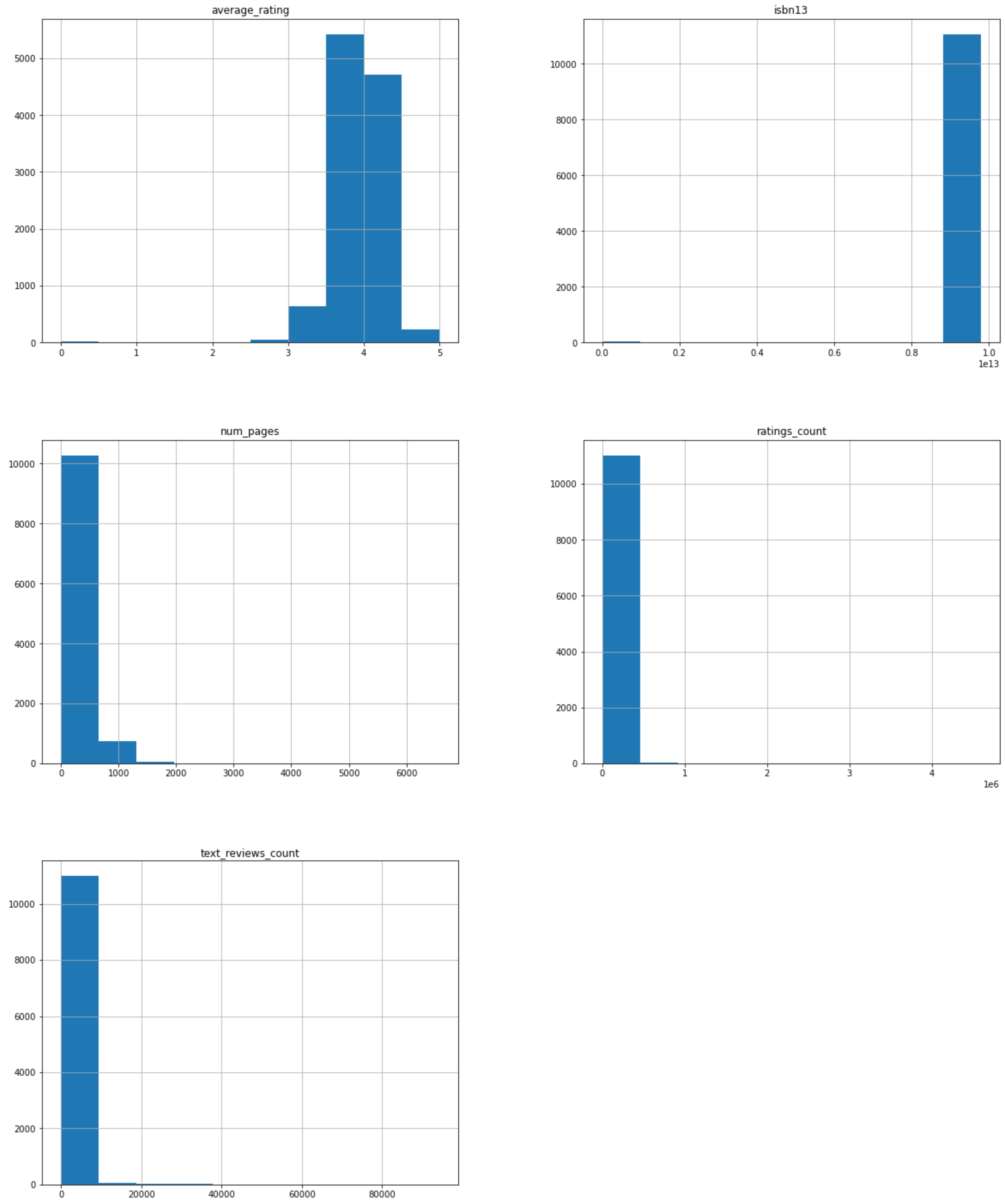
Out[24]:

	average_rating	isbn13	num_pages	ratings_count	text_reviews_count
count	11094.000000	1.109400e+04	11094.000000	1.109400e+04	11094.000000
mean	3.935026	9.759826e+12	336.543537	1.798750e+04	543.304309
std	0.346458	4.435532e+11	241.313733	1.126427e+05	2579.856004
min	0.000000	8.987060e+09	0.000000	0.000000e+00	0.000000
25%	3.770000	9.780345e+12	192.000000	1.050000e+02	9.000000
50%	3.960000	9.780582e+12	299.000000	7.490000e+02	47.000000
75%	4.140000	9.780872e+12	416.000000	5.018750e+03	238.000000
max	5.000000	9.790008e+12	6576.000000	4.597666e+06	94265.000000

In [25]:

```
fig = plt.figure(figsize = (20,25))
ax = fig.gca()
df[continuousVars].hist(ax = ax)
plt.show()
```

```
C:\Users\steph\AppData\Local\Temp/ipykernel_1044/133798998.py:3: UserWarning: To out
put multiple subplots, the figure containing the passed axes is being cleared.
  df[continuousVars].hist(ax = ax)
```



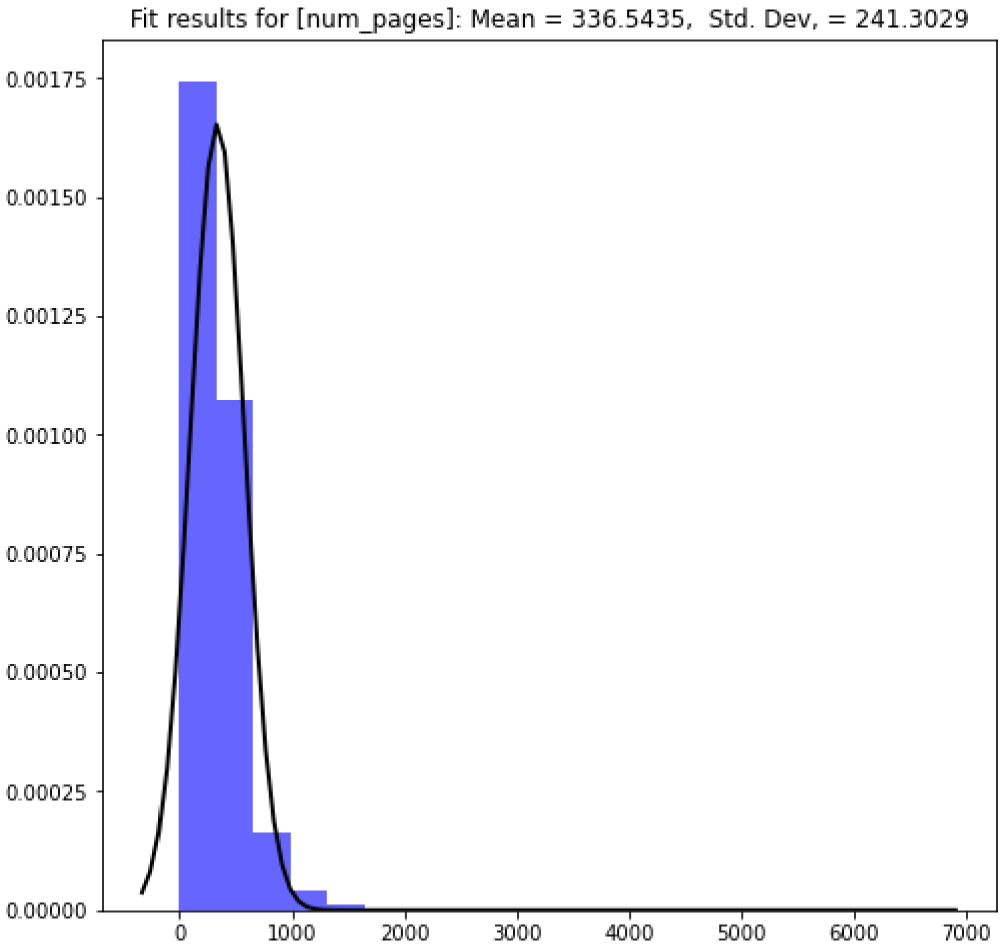
1. Drawing a Normal curve on the histograms could help understand the distribution type

In [26]:

```
#Think it would be nice to draw a normal curve on the histograms to check out the sk
# The function below fits a normal distribution to the data
# It is named PlotHistogramsWithNormalCurve and the following are the parameters,
#  booksCol   - The univariate column for which to plot the histogram (pandas vector
#  varName - Column name to print on titles (string)
#  bins    - Preferred bins in the histogram (default = 20)
#  color   - Preferred color of histogram (default is blue)
def PlotHistogramsWithNormalCurve(dfCol, varName, bins=20, color='b'):
    dMean, dStd = norm.fit(dfCol)
    plt.figure(figsize = (8, 8))
    # Plot hist
    plt.hist(dfCol, bins, density=True, alpha=0.6, color=color)
    # Plot PDF.
    xmin, xmax = plt.xlim()
    xlin = np.linspace(xmin, xmax, 100)
    pdf = norm.pdf(xlin, dMean, dStd)
    plt.plot(xlin, pdf, 'k', linewidth=2)
    title = "Fit results for [" + varName + "]: Mean = %.4f,  Std. Dev, = %.4f" % (d
    plt.title(title)
    plt.show()
```

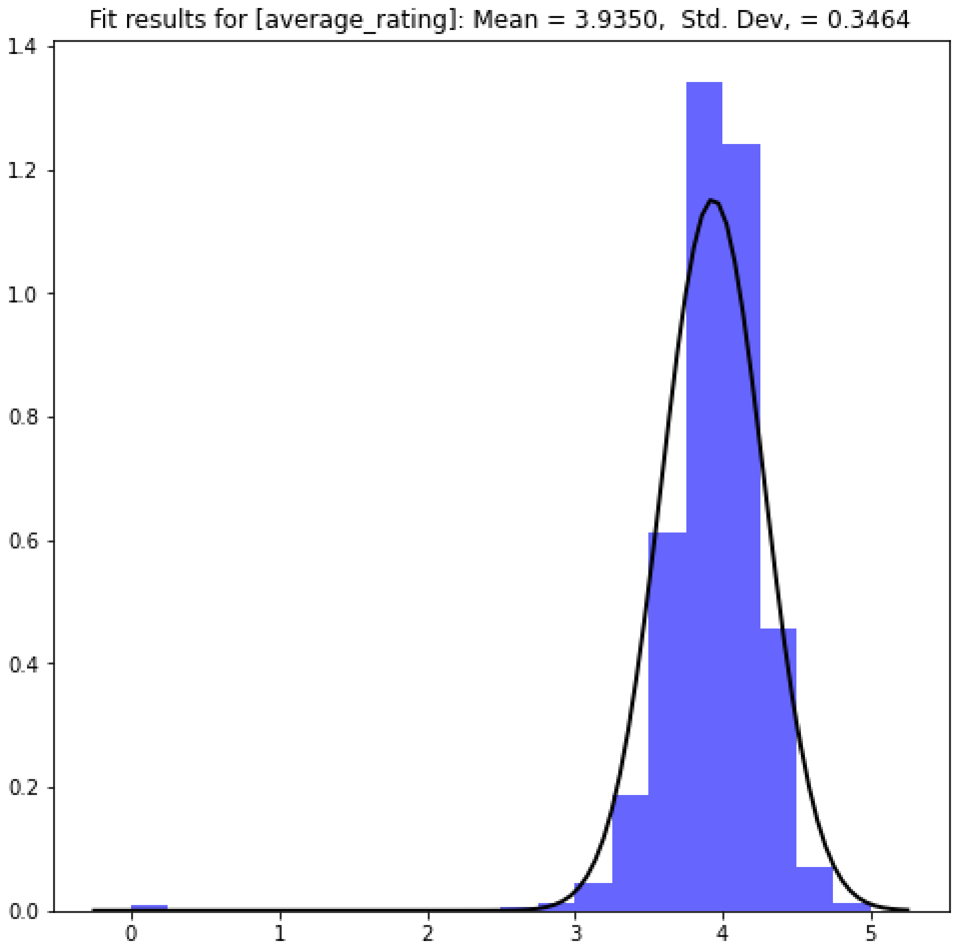
' num_pages' has a left skewed distribution

In [27]:
```
PlotHistogramsWithNormalCurve(df['num_pages'], "num_pages")
```



'average_rating' is Normally distributed

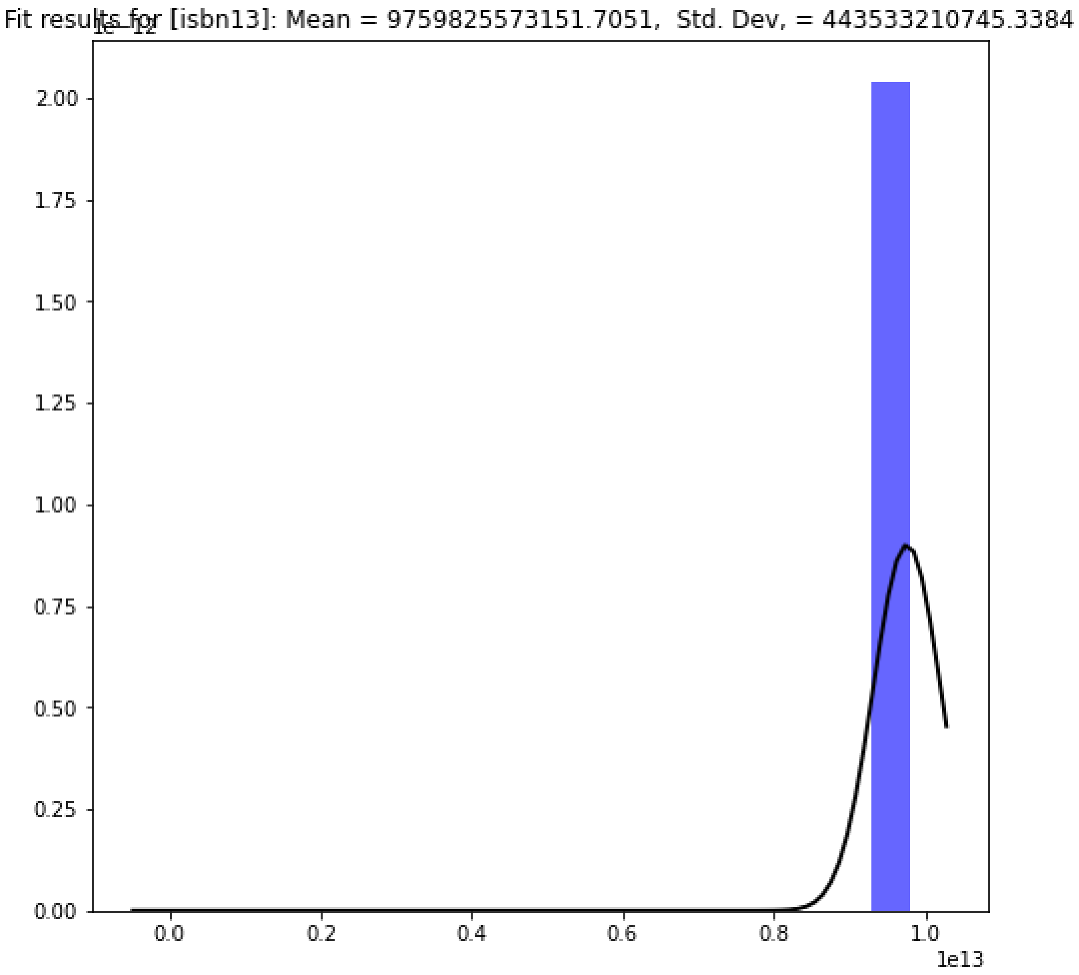
In [28]:
```
PlotHistogramsWithNormalCurve(df['average_rating'], "average_rating")
```



'isbn13' shows not much of a distribution, mostly falls into one bin

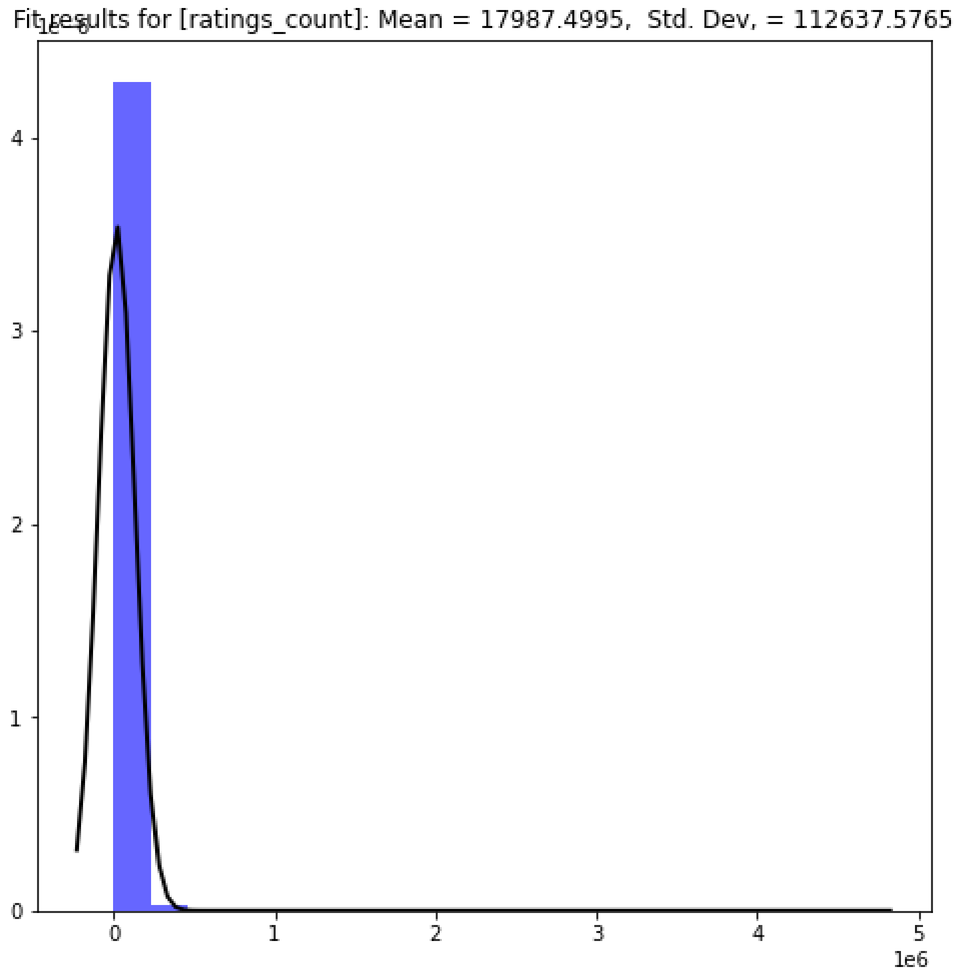
In [29]:

```
PlotHistogramsWithNormalCurve(df['isbn13'], "isbn13")
```



'ratings_count' is possibly left skewed but could there be extreme values in the distribution causing it to be skewed?

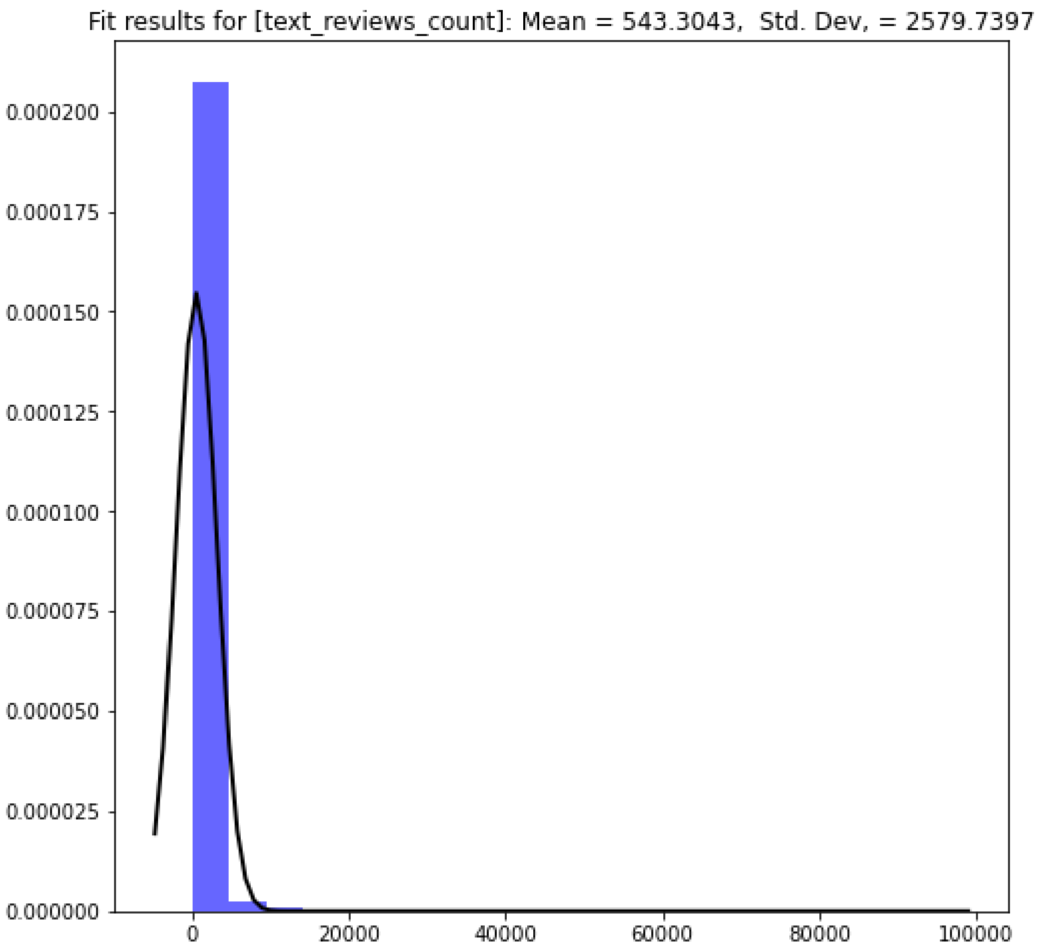
```
In [30]:  PlotHistogramsWithNormalCurve(df['ratings_count'], "ratings_count")
```



'text_reviews_count' is possibly left skewed but could there be extreme values in the distribution causing it to be skewed?

In [31]:

```
#Possibly left skewed but looks like there are some extreme values in the distributi
PlotHistogramsWithNormalCurve(df['text_reviews_count'], "text_reviews_count")
```

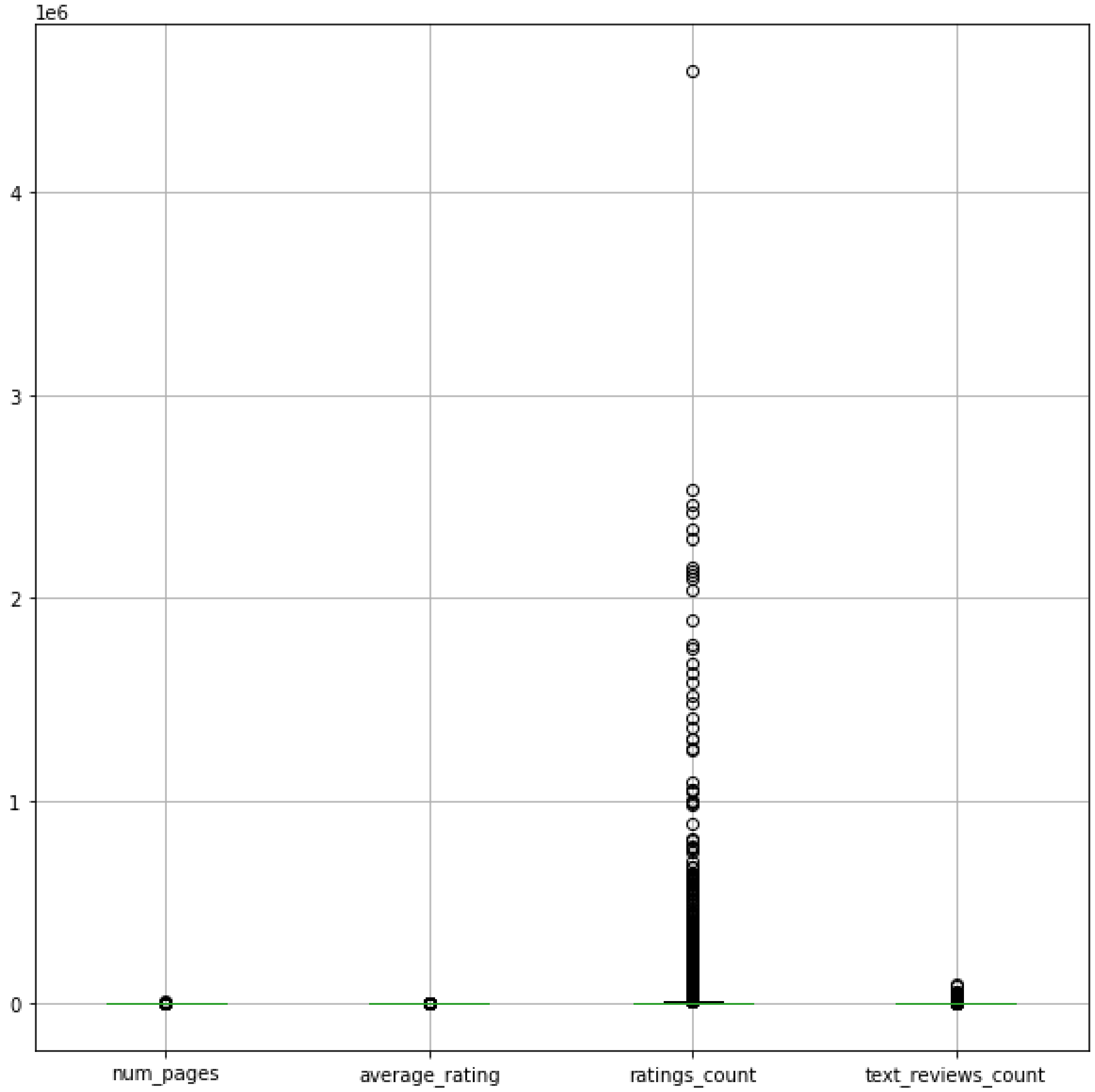


1. Box-plots could help detect variables with outliers

Looks like there are couple of outliers values that are 4 and 5 millons rating_counts that stretch the y-axis scale of the box plots

In [32]:

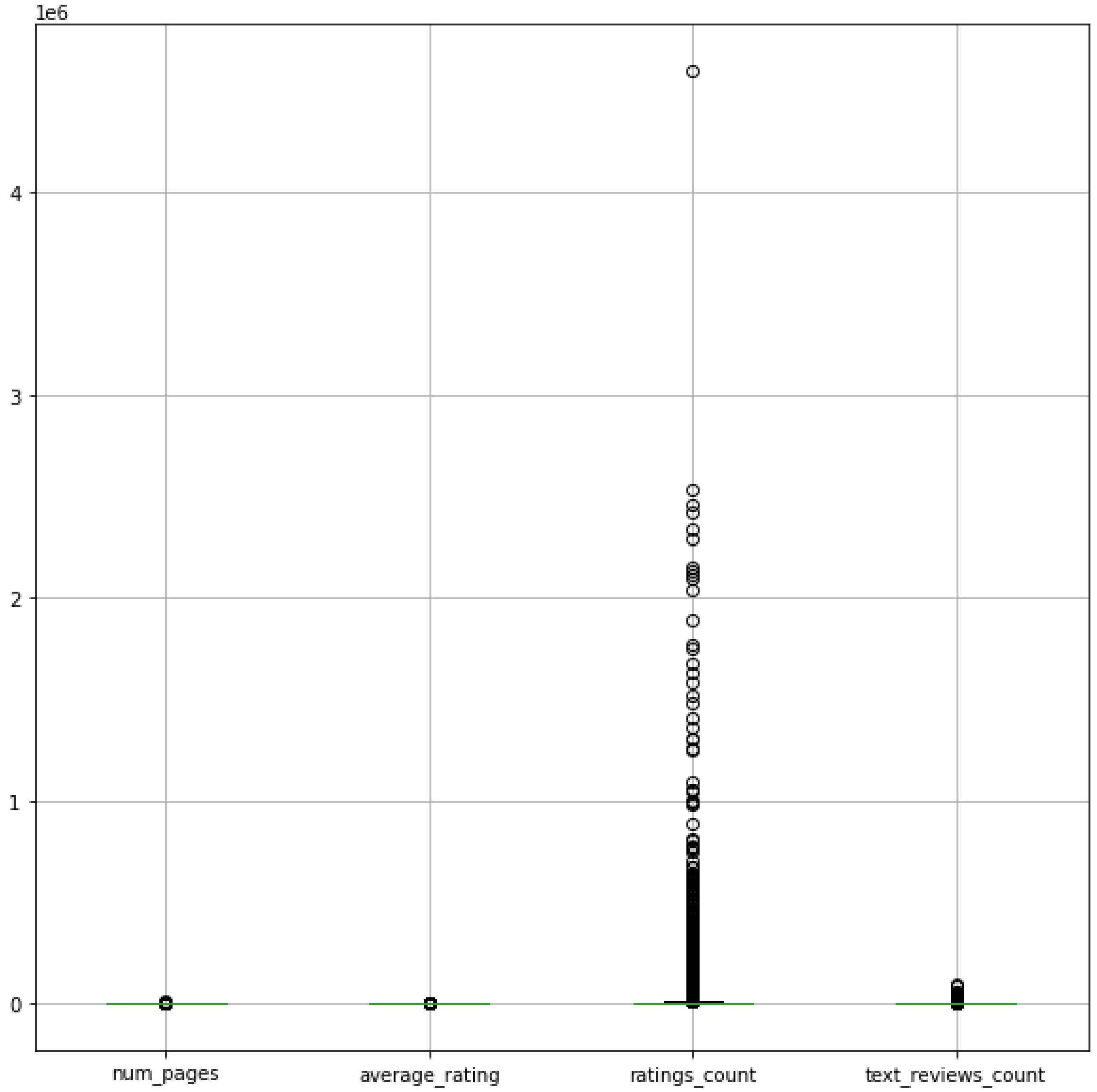
```
plt.figure(figsize = (10, 10))
df.boxplot(column= ['num_pages', 'average_rating', 'ratings_count', 'text_reviews_co
plt.show()
```



Clearly we need to normalize this data to see all variables on the same scale

In [33]:

```
df1 = df[(df['ratings_count'] < 1000)]
plt.figure(figsize = (10, 10))
df.boxplot(column= ['num_pages', 'average_rating', 'ratings_count', 'text_reviews_co
plt.show()
```



The normalized box-plot clearly fits our variables on the same scale and also shows many values outside of the Inter Quartile Range (IQR), min and max values

In [34]:

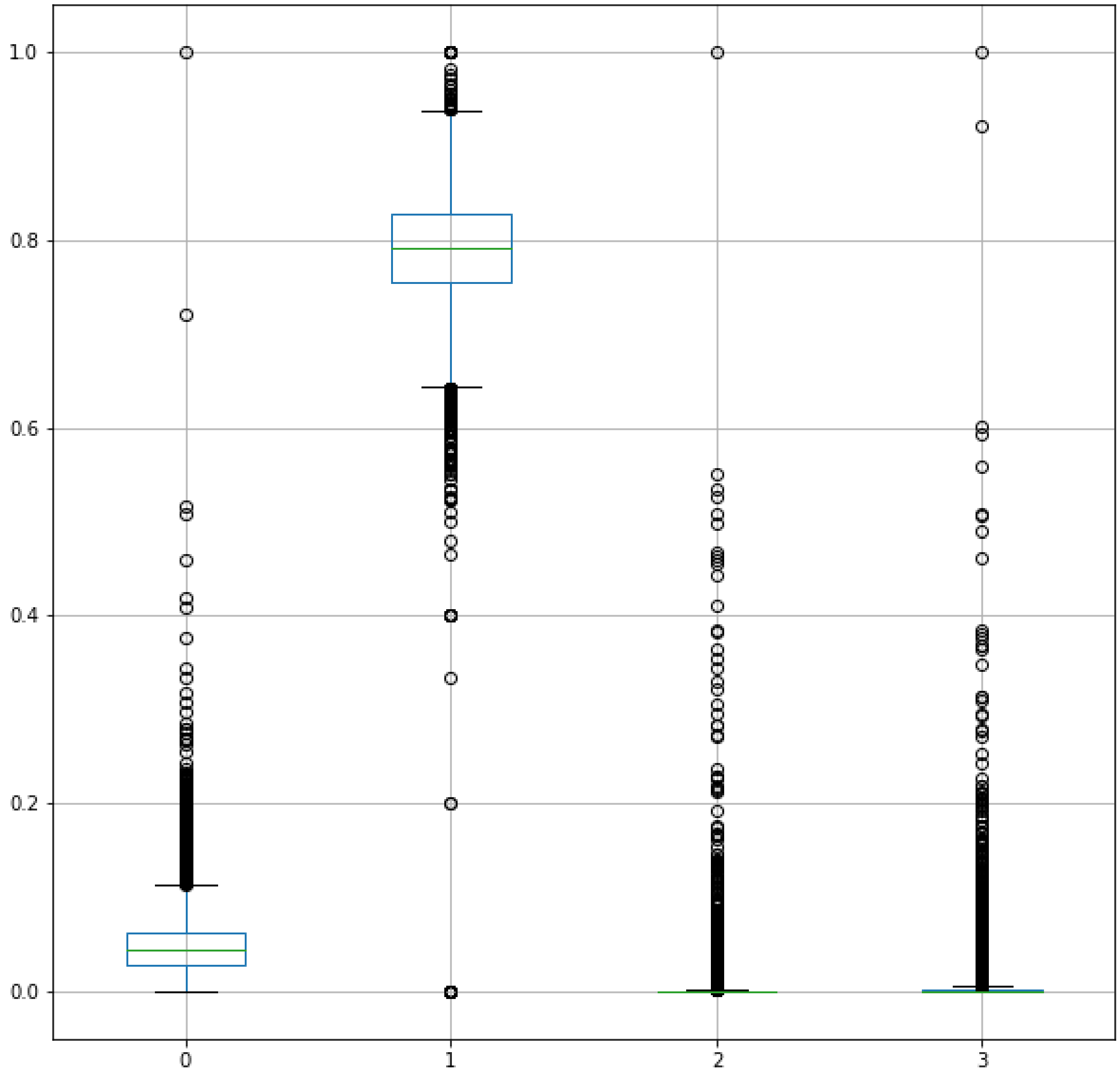
```
# Create varsToNormalize, where all the varsToNormalize values are treated as floats
varsToNormalize = df[['num_pages', 'average_rating', 'ratings_count', 'text_reviews_

# Create a minimum and maximum preprocessing object
range_Scaler = preprocessing.MinMaxScaler()

# Create an object to transform the data to fit minmax processor
vars_Scaled = range_Scaler.fit_transform(varsToNormalize)

# Run the normalizer on the dataframe
df_normalized = pd.DataFrame(vars_Scaled)

plt.figure(figsize = (10, 10))
df_normalized.boxplot()
plt.show()
```



E. Let's check out the categorical variables in data

In [35]:

```
categoricalVars = ['bookID', 'title', 'authors', 'isbn', 'language_code', 'publicati
df[categoricalVars].describe()
```

Out[35]:

	bookID	title	authors	isbn	language_code	publication_date	publisher
count	11094	11094	11094	11094	11094	11094	11094
unique	11094	10319	6618	11094	27	3669	2314
top	1	The Iliad	P.G. Wodehouse	0439785960	eng	10/1/2005	Vintage
freq	1	9	40	1	8885	56	318

Regarding the result on the categorical variables we found:

1. the top recurring book title is "The liad" (count of 9)
2. P.G. Wodehouse is the author with the most books (count of 40)
3. Most of the books are in English (8885/11094)
4. Most publisher is Vintage

5. The top publication date is "10/01/2005"

DATA ANALYSIS

Since we have explored the data we are going to make some analysis.

To perform a Machine Learning model on supervised learning we need to fill these requirements:

- No missing values
- Data in numeric format
- Data stores in Pandas DataFrame

In [36]:

```
import os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn import metrics
from sklearn import preprocessing
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
```

In [95]:

```
file_path = ("C:/Users/steph/OneDrive/Desktop/data Analyst DSTI/Python_Machine_Labs/
df = pd.read_csv(file_path,sep = ",", error_bad_lines=False)
print("The data contains {0} Rows and {1} Columns".format(df.shape[0],df.shape[1]))
```

```
The data contains 11123 Rows and 12 Columns
C:\Users\steph\AppData\Local\Temp/ipykernel_1044/2146545102.py:2: FutureWarning: The
error_bad_lines argument has been deprecated and will be removed in a future versio
n. Use on_bad_lines in the future.


  df = pd.read_csv(file_path,sep = ",", error_bad_lines=False)
b'Skipping line 3350: expected 12 fields, saw 13\nSkipping line 4704: expected 12 fi
elds, saw 13\nSkipping line 5879: expected 12 fields, saw 13\nSkipping line 8981: ex
pected 12 fields, saw 13\n'
```

In [96]:

```
df.head(5)
```

Out[96]:

	bookID	title	authors	average_rating	isbn	isbn13	language_code	num
0	1	Harry Potter and the Half-Blood Prince (Harry ...	J.K. Rowling/Mary GrandPré	4.57	0439785960	9.780440e+12	eng	
1	2	Harry Potter and the Order of the Phoenix (Har...	J.K. Rowling/Mary GrandPré	4.49	0439358078	9.780439e+12	eng	

	bookID	title	authors	average_rating	isbn	isbn13	language_code	num
2	4	Harry Potter and the Chamber of Secrets (Harry...	J.K. Rowling	4.42	0439554896	9.780440e+12	eng	
3	5	Harry Potter and the Prisoner of Azkaban (Harr...	J.K. Rowling/Mary GrandPré	4.56	043965548X	9.780440e+12	eng	
4	8	Harry Potter Boxed Set Books 1-5 (Harry Potte...	J.K. Rowling/Mary GrandPré	4.78	0439682584	9.780440e+12	eng	

In [100…

```
df=df.drop(['bookID', 'publication_date', 'publisher;;;'], axis=1)
df
```

Out[100…

	title	authors	average_rating	isbn	isbn13	language_code	num
0	Harry Potter and the Half-Blood Prince (Harry ...	J.K. Rowling/Mary GrandPré	4.57	0439785960	9.780440e+12	eng	
1	Harry Potter and the Order of the Phoenix (Har...	J.K. Rowling/Mary GrandPré	4.49	0439358078	9.780439e+12	eng	
2	Harry Potter and the Chamber of Secrets (Harry...	J.K. Rowling	4.42	0439554896	9.780440e+12	eng	
3	Harry Potter and the Prisoner of Azkaban (Harr...	J.K. Rowling/Mary GrandPré	4.56	043965548X	9.780440e+12	eng	

	title	authors	average_rating	isbn	isbn13	language_code	num
4	Harry Potter Boxed Set Books 1-5 (Harry Potte...	J.K. Rowling/Mary GrandPré	4.78	0439682584	9.780440e+12	eng	
...	...	...	...	...	...	...	
11118	Expelled from Eden: A William T. Vollmann Reader	William T. Vollmann/Larry McCaffery/Michael He...	4.06	1560254416	9.781560e+12	eng	
11119	You Bright and Risen Angels	William T. Vollmann	4.08	0140110879	9.780140e+12	eng	
11120	The Ice-Shirt (Seven Dreams #1)	William T. Vollmann	3.96	0140131965	9.780140e+12	eng	
11121	Poor People	William T. Vollmann	3.72	0060878827	9.780061e+12	eng	
11122	Las aventuras de Tom Sawyer	Mark Twain	3.91	8497646983	9.788498e+12	spa	

11123 rows × 9 columns

In [101…

```
df_index_ChosenLangs = df.loc[df['language_code'].isin(['eng','en-US', 'spa', 'fre']
df_index_ChosenLangs.shape
df.head()
```

Out[101…

	title	authors	average_rating	isbn	isbn13	language_code	num_pages	r
0	Harry Potter and the Half-Blood Prince (Harry ...	J.K. Rowling/Mary GrandPré	4.57	0439785960	9.780440e+12	eng	652.0	
1	Harry Potter and the Order of the Phoenix (Har...	J.K. Rowling/Mary GrandPré	4.49	0439358078	9.780439e+12	eng	870.0	

	title	authors	average_rating	isbn	isbn13	language_code	num_pages	r
2	Harry Potter and the Chamber of Secrets (Harry...	J.K. Rowling	4.42	0439554896	9.780440e+12	eng	352.0	
3	Harry Potter and the Prisoner of Azkaban (Harr...	J.K. Rowling/Mary GrandPré	4.56	043965548X	9.780440e+12	eng	435.0	
4	Harry Potter Boxed Set Books 1-5 (Harry Potte...	J.K. Rowling/Mary GrandPré	4.78	0439682584	9.780440e+12	eng	2690.0	

3. Split data into two sets based on ratings count and chosen languages ['eng','en-US', 'spa', 'fre'].

1. Assume high ratings as ratings >= 100000
2. 'ratings_count' - Drop ratings below 100000

In [102...

```
High_Rating = 100000
df_HighRatedBooks = df_index_ChosenLangs.drop(df_index_ChosenLangs.index[df_index_Ch
```

In [103...

```
df_HighRatedBooks = df_index_ChosenLangs.drop(df_index_ChosenLangs.index[df_index_Ch
```

In [104...

```
df_HighRatedBooks.shape
```

Out[104...

```
(351, 9)
```

In [106...

```
df_HighRatedBooks.head()
```

Out[106...

	title	authors	average_rating	isbn	isbn13	language_code	num_page
0	Harry Potter and the Half-Blood Prince (Harry ...	J.K. Rowling/Mary GrandPré	4.57	0439785960	9.780440e+12	eng	652.(

	title	authors	average_rating	isbn	isbn13	language_code	num_page
1	Harry Potter and the Order of the Phoenix (Har...	J.K. Rowling/Mary GrandPré	4.49	0439358078	9.780439e+12	eng	870.
3	Harry Potter and the Prisoner of Azkaban (Harr...	J.K. Rowling/Mary GrandPré	4.56	043965548X	9.780440e+12	eng	435.
8	The Ultimate Hitchhiker's Guide to the Galaxy ...	Douglas Adams	4.38	0345453743	9.780345e+12	eng	815.
12	A Short History of Nearly Everything	Bill Bryson	4.21	076790818X	9.780768e+12	eng	544.

In [107…

```
df_OtherBooks = df_index_ChosenLangs.drop(df_index_ChosenLangs.index[df_index_Chosen
```

In [108…

```
df_OtherBooks.shape
```

Out[108…

```
(10299, 9)
```

In [109…

```
df_OtherBooks.head()
```

Out[109…

	title	authors	average_rating	isbn	isbn13	language_code	num_pages
2	Harry Potter and the Chamber of Secrets (Harry...	J.K. Rowling	4.42	0439554896	9.780440e+12	eng	352.0
4	Harry Potter Boxed Set Books 1-5 (Harry Potte...	J.K. Rowling/Mary GrandPré	4.78	0439682584	9.780440e+12	eng	2690.0
6	Harry Potter Collection (Harry Potter #1-6)	J.K. Rowling	4.73	0439827604	9.780440e+12	eng	3342.0

	title	authors	average_rating	isbn	isbn13	language_code	num_pages
7	The Ultimate Hitchhiker's Guide: Five Complete...	Douglas Adams	4.38	0517226952	9.780517e+12	eng	815.0
9	The Hitchhiker's Guide to the Galaxy (Hitchhik...	Douglas Adams	4.22	1400052920	9.781400e+12	eng	215.0

4. Encoding Categorical variables of the two samples

A. Encoding 'title'

Encoding samples in books_HighRatedBooks

In [110…

```
# encode title column
labelEncode = preprocessing.LabelEncoder()
df_HighRatedBooks['title'] = labelEncode.fit_transform(df_HighRatedBooks['title'])
df_OtherBooks['title'] = labelEncode.fit_transform(df_OtherBooks['title'])
```

In [111…

```
df_HighRatedBooks.head()
```

Out[111…

	title	authors	average_rating	isbn	isbn13	language_code	num_pages	ratir
0	97	J.K. Rowling/Mary GrandPré	4.57	0439785960	9.780440e+12	eng	652.0	
1	98	J.K. Rowling/Mary GrandPré	4.49	0439358078	9.780439e+12	eng	870.0	
3	99	J.K. Rowling/Mary GrandPré	4.56	043965548X	9.780440e+12	eng	435.0	
8	312	Douglas Adams	4.38	0345453743	9.780345e+12	eng	815.0	
12	19	Bill Bryson	4.21	076790818X	9.780768e+12	eng	544.0	

Encoding samples in books_OtherBooks

In [112…

```
df_OtherBooks.head()
```

Out[112…

	title	authors	average_rating	isbn	isbn13	language_code	num_pages	ratin
2	2854	J.K. Rowling	4.42	0439554896	9.780440e+12	eng	352.0	
4	2849	J.K. Rowling/Mary GrandPré	4.78	0439682584	9.780440e+12	eng	2690.0	

	title	authors	average_rating	isbn	isbn13	language_code	num_pages	ratin
6	2850	J.K. Rowling	4.73	0439827604	9.780440e+12	eng	3342.0	
7	8571	Douglas Adams	4.38	0517226952	9.780517e+12	eng	815.0	
9	7321	Douglas Adams	4.22	1400052920	9.781400e+12	eng	215.0	

B. Encode 'authors'

Encoding samples in books_HighRatedBooks

In [113...

```
# encode authors column
df_HighRatedBooks['authors'] = labelEncode.fit_transform(df_HighRatedBooks['authors'
df_HighRatedBooks.head()
```

Out[113...

	title	authors	average_rating	isbn	isbn13	language_code	num_pages	ratings_co
0	97	89	4.57	0439785960	9.780440e+12	eng	652.0	20956
1	98	89	4.49	0439358078	9.780439e+12	eng	870.0	21531
3	99	89	4.56	043965548X	9.780440e+12	eng	435.0	23395
8	312	54	4.38	0345453743	9.780345e+12	eng	815.0	2495
12	19	26	4.21	076790818X	9.780768e+12	eng	544.0	2485

Encoding samples in books_OtherBooks

In [114...

```
df_OtherBooks['authors'] = labelEncode.fit_transform(df_OtherBooks['authors'])
df_OtherBooks.head()
```

Out[114...

	title	authors	average_rating	isbn	isbn13	language_code	num_pages	ratings_co
2	2854	2493	4.42	0439554896	9.780440e+12	eng	352.0	633
4	2849	2495	4.78	0439682584	9.780440e+12	eng	2690.0	4142
6	2850	2493	4.73	0439827604	9.780440e+12	eng	3342.0	2824
7	8571	1396	4.38	0517226952	9.780517e+12	eng	815.0	362
9	7321	1396	4.22	1400052920	9.781400e+12	eng	215.0	493

In [115...

```
#df_OtherBooks = df_OtherBooks.loc[df['language_code'].isin(['eng','en-US', 'spa', '
```

C. Dummy encode 'language_code'

Encoding samples in books_HighRatedBooks

In [116...

```
encoded_lang_high = pd.get_dummies(df_HighRatedBooks['language_code'])
colsExist2 = df_HighRatedBooks.columns.isin(['en-US', 'eng', 'fre', 'spa']).any()
if  colsExist2 == False:
    df_HighRatedBooks = pd.concat([df_HighRatedBooks, encoded_lang_high], axis = 1)
print(df_HighRatedBooks.shape)
df_HighRatedBooks.head()
```

```
(351, 13)
```

Out[116...

	title	authors	average_rating	isbn	isbn13	language_code	num_pages	ratings_c
0	97	89	4.57	0439785960	9.780440e+12	eng	652.0	20956
1	98	89	4.49	0439358078	9.780439e+12	eng	870.0	21531
3	99	89	4.56	043965548X	9.780440e+12	eng	435.0	23395
8	312	54	4.38	0345453743	9.780345e+12	eng	815.0	2495
12	19	26	4.21	076790818X	9.780768e+12	eng	544.0	2485

Encoding samples in books_Otherbooks

In [117...

```
encoded_lang_other = pd.get_dummies(df_OtherBooks['language_code'])
encoded_lang_other.head()
colsExist = df_OtherBooks.columns.isin(['en-US', 'eng', 'fre', 'spa']).any()
if  colsExist == False:
    df_OtherBooks = pd.concat([df_OtherBooks, encoded_lang_other], axis = 1)
print(df_OtherBooks.shape)
df_OtherBooks.head()
```

```
(10299, 13)
```

Out[117...

	title	authors	average_rating	isbn	isbn13	language_code	num_pages	ratings_co
2	2854	2493	4.42	0439554896	9.780440e+12	eng	352.0	633
4	2849	2495	4.78	0439682584	9.780440e+12	eng	2690.0	4142
6	2850	2493	4.73	0439827604	9.780440e+12	eng	3342.0	2824
7	8571	1396	4.38	0517226952	9.780517e+12	eng	815.0	362
9	7321	1396	4.22	1400052920	9.781400e+12	eng	215.0	493

5. Building a Linear Regression Model

In [119...

```
def ModelBuilding_LinearRegression(df_Current, testSize=0.2):
    # divide the data into attributes and labels
    X = df_Current.drop(['average_rating', 'language_code', 'isbn'], axis = 1)
    y = df_Current['average_rating']
    print("Shape of Inputs = {0}".format(X.shape))
    print("Shape of Target = {0}".format(y.shape))
    # split 80% of the data to the training set and 20% of the data to test set
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = testSize,
    linReg = LinearRegression()
    linReg.fit(X_train, y_train)
    print("Intercept = {0}".format(linReg.intercept_))
    print("Coefficients = {0}".format(linReg.coef_.tolist()))
    predictions = linReg.predict(X_test)
    return (predictions, X_test, y_test, linReg)
```

a. Model Intercept, coefficients and "average_rating" predictions for Highly rated books

In [120…

```
Pedicted_Scores, X_test, y_test, linReg = ModelBuilding_LinearRegression(df_HighRate
```

```
Shape of Inputs = (351, 10)
Shape of Target = (351,)
Intercept = 169.14234175494153
Coefficients = [-0.00013462986403421518, -0.0003737002129126569, -1.6891973385686258
e-11, 0.00020845172853968324, -1.4742504764332833e-08, 1.7784589531589977e-06, 0.147
84474460454192, 0.08049285029530812, -0.22745584968127244, -0.0008817452185777917]
```

In [121…

```
Actual_Predicted = pd.DataFrame({'Observed': y_test.tolist(), 'Predicted': Pedicted_
Actual_Predicted['diff'] = Actual_Predicted['Observed'] - Actual_Predicted['Predicte
Actual_Predicted.head(10)
```

Out[121…

	Observed	Predicted	diff
0	3.89	3.997108	-0.107108
1	3.82	4.008990	-0.188990
2	3.97	3.943684	0.026316
3	4.27	4.036814	0.233186
4	4.22	3.941376	0.278624
5	4.18	3.991822	0.188178
6	4.24	4.102334	0.137666
7	3.70	4.020919	-0.320919
8	3.82	4.018871	-0.198871
9	3.74	3.938545	-0.198545

b. The model looks reasonable with Root Mean Square Error (RMSE) at around 0.22 and Mean Absolute Error (MAE) around 0.2

In [122…

```
# evaluate the performance of the algorithm
print('Mean Absolute Error (MAE):', metrics.mean_absolute_error(y_test, Pedicted_Sco
print('Mean Squared Error (MSE):', metrics.mean_squared_error(y_test, Pedicted_Score
print('Root Mean Squared Error (RMSE):', np.sqrt(metrics.mean_squared_error(y_test,
print('Mean Absolute Error (MAE):', np.sum(abs(Actual_Predicted['diff']))/Actual_Pre
```

```
Mean Absolute Error (MAE): 0.20163286723913348
Mean Squared Error (MSE): 0.06489348732774741
Root Mean Squared Error (RMSE): 0.2547420014990606
Mean Absolute Error (MAE): 0.20163286723913348
```

In [123…

```
X_test.head()
```

Out[123…

	title	authors	isbn13	num_pages	ratings_count	text_reviews_count	en-US	eng	fre	s
2126	213	168	9.780061e+12	327.0	202043.0	2850.0	0	1	0	
3557	87	152	9.780386e+12	240.0	132584.0	9341.0	0	1	0	
2831	215	224	9.781590e+12	273.0	235924.0	5869.0	0	1	0	

	title	authors	isbn13	num_pages	ratings_count	text_reviews_count	en-US	eng	fre	s
1697	261	92	9.780618e+12	366.0	2530894.0	32871.0	0	1	0	
868	41	233	9.781591e+12	200.0	140403.0	1063.0	0	1	0	

c. Testing a mocked up case for deployment (prediction accuracy seems very close for this case, around 0.99)

In [127…

```
Mocked_Case = np.array([888, 27, 9780812474947, 232, 117003, 5141, 0, 1, 0, 0])
```

In [128…

```
Score_Mocked_Case = linReg.predict(Mocked_Case.reshape(1, -1))
```

```
C:\Users\steph\anaconda3\envs\ClassProject\lib\site-packages\sklearn\base.py:450: Us
erWarning: X does not have valid feature names, but LinearRegression was fitted with
feature names
  warnings.warn(
```

In [129…

```
Predictions = pd.DataFrame({'Observed': y_test.iloc[0], 'Predicted': Score_Mocked_Ca
Predictions
```

Out[129…

	Observed	Predicted
0	3.89	3.931748

In [130…

```
X2 = df_OtherBooks.drop(['average_rating', 'language_code', 'isbn'], axis = 1)
y2 = df_OtherBooks['average_rating']
print(X2.shape)
print(y2.shape)
```

```
(10299, 10)
(10299,)
```

In [131…

```
Scores2 = linReg.predict(X2)
```

In [132…

```
DeployedModelPredictions = pd.DataFrame({'Observed': y2.tolist(), 'Predicted': Score
DeployedModelPredictions['diff'] = DeployedModelPredictions['Observed'] - DeployedMo
DeployedModelPredictions.head(10)
```

Out[132…

	Observed	Predicted	diff
0	4.42	2.769757	1.650243
1	4.78	3.256381	1.523619
2	4.73	3.394242	1.335758
3	4.38	2.505286	1.874714
4	4.22	2.533937	1.686063
5	4.22	2.499349	1.720651
6	4.38	2.505328	1.874672

	Observed	Predicted	diff
7	3.44	3.683715	-0.243715
8	3.87	3.694340	0.175660
9	4.07	3.434095	0.635905

d. The model built for df_HighRatedBooks performs worse on df_OtherBooks with a Mean Absolute Error (MAE) 2.06

In [133…

```
print('Mean Absolute Error (MAE):', np.sum(abs(DeployedModelPredictions['diff']))/De
```

```
Mean Absolute Error (MAE): 2.0640258938254616
```

e. Model Intercept, coefficients and "average_rating" predictions for Other books not highly rated (i.e. with ratings < 100,000)

In [134…

```
Pedicted_Scores, X_test, y_test, linReg = ModelBuilding_LinearRegression(df_OtherBoo
```

```
Shape of Inputs = (10299, 10)
Shape of Target = (10299,)
Intercept = 3.819179685994803
Coefficients = [1.5170610663429518e-06, 3.300343791981658e-06, 3.0595756690579383e-1
5, 0.00020803057965679484, 3.617109650654179e-06, -6.402159857807258e-05, -0.0234840
5635272422, -0.010198447631590656, 0.04272885979107744, -0.00904635580676259]
```

In [135…

```
Actual_Predicted = pd.DataFrame({'Observed': y_test.tolist(), 'Predicted': Pedicted_
Actual_Predicted['diff'] = Actual_Predicted['Observed'] - Actual_Predicted['Predicte
Actual_Predicted.head(10)
```

Out[135…

	Observed	Predicted	diff
0	3.82	3.933810	-0.113810
1	3.79	3.896995	-0.106995
2	4.04	3.999121	0.040879
3	3.21	3.934726	-0.724726
4	4.08	3.932726	0.147274
5	4.00	3.927792	0.072208
6	4.12	3.871268	0.248732
7	4.07	3.859860	0.210140
8	3.75	3.939775	-0.189775
9	4.04	4.038734	0.001266

f. The second model performs way better with Root Mean Square Error (RMSE) at around 0.35 and Mean Absolute Error (MAE) around 0.23 for other books not rated high

In [136…

```
# evaluate the performance of the algorithm
print('Mean Absolute Error (MAE):', metrics.mean_absolute_error(y_test, Pedicted_Sco
print('Mean Squared Error (MSE):', metrics.mean_squared_error(y_test, Pedicted_Score
print('Root Mean Squared Error (RMSE):', np.sqrt(metrics.mean_squared_error(y_test,
Mean Absolute Error (MAE): 0.2285414359993815
Mean Squared Error (MSE): 0.12196648465344245
Root Mean Squared Error (RMSE): 0.34923700355695764
```

In [ ]:

```
```