Extract and loading

```
from google.colab import files
import pandas as pd
import io

#Upload / read file
df = pd.read_csv("Books.csv")

# Display first 10 entries
df.head(10)
```

	index	Publishing Year	Book Name	Author	language_code	Author_Rating	Book_average
0	0	1975.0	Beowulf	Unknown, Seamus Heaney	en-US	Novice	
1	1	1987.0	Batman: Year One	Frank Miller, David Mazzucchelli, Richmond Lew...	eng	Intermediate	
2	2	2015.0	Go Set a Watchman	Harper Lee	eng	Novice	
3	3	2008.0	When You Are Engulfed in Flames	David Sedaris	en-US	Intermediate	
4	4	2011.0	Daughter of Smoke & Bone	Laini Taylor	eng	Intermediate	
5	5	2015.0	Red Queen	Victoria Aveyard	eng	Intermediate	
6	6	2011.0	The Power of Habit	Charles Duhigg	eng	Intermediate	
7	7	1994.0	Midnight in the Garden of Good and Evil	John Berendt	eng	Intermediate	

Next steps: View recommended plots

```
#identify missing data
print(df.isna().sum())
```

```
index                  0
Publishing Year        1
Book Name             23
Author                 0
language_code         53
Author_Rating          0
Book_average_rating    0
Book_ratings_count     0
genre                  0
gross sales            0
publisher revenue      0
sale price             0
sales rank             0
Publisher              0
units sold             0
dtype: int64
```

```
#Imputing missing data/ make missing values the mean of known values
#calculate mean
mean_py = df["Publishing Year"].mean()

#Replace missing values
df["Publishing Year"] = df["Publishing Year"].fillna(mean_py)

#Calculate mode
mode_lc = df["language_code"].mode()[0]

#Replace missing values categorically with most common value
df["language_code"] = df["language_code"].fillna(mode_lc)

#Replace gerne fiction to fiction in genre column
df["genre"] = df["genre"].replace("genre fiction", "fiction")

print(df.isna().sum())
```

```
index                   0
Publishing Year         0
Book Name              23
Author                  0
language_code           0
Author_Rating           0
Book_average_rating     0
Book_ratings_count      0
genre                   0
gross sales             0
publisher revenue       0
sale price              0
sales rank              0
Publisher               0
units sold              0
dtype: int64
```

```
#Indicator values/ new column that states if a value is missing

#creat new column
df["Missing Book Name"] = df["Book Name"].isna().astype(int)


df.head()
```

	index	Publishing Year	Book Name	Author	language_code	Author_Rating	Book_average
0	0	1975.0	Beowulf	Unknown, Seamus Heaney	en-US	Novice	
1	1	1987.0	Batman: Year One	Frank Miller, David Mazzucchelli, Richmond Lew...	eng	Intermediate	
2	2	2015.0	Go Set a Watchman	Harper Lee	eng	Novice	
3	3	2008.0	When You Are Engulfed in Flames	David Sedaris	en-US	Intermediate	
4	4	2011.0	Daughter of Smoke & Bone	Laini Taylor	eng	Intermediate	

Next steps: View recommended plots

Transformation

```
#Replace spaces with underscores
df.columns = df.columns.str.replace(" ", "_")

#Show updated
df.head()
```

	index	Publishing_Year	Book_Name	Author	language_code	Author_Rating	Book_a
0	0	1975.0	Beowulf	Unknown, Seamus Heaney	en-US	Novice	
1	1	1987.0	Batman: Year One	Frank Miller, David Mazzucchelli, Richmond Lew...	eng	Intermediate	
2	2	2015.0	Go Set a Watchman	Harper Lee	eng	Novice	
3	3	2008.0	When You Are Engulfed in Flames	David Sedaris	en-US	Intermediate	
4	4	2011.0	Daughter of Smoke & Bone	Laini Taylor	eng	Intermediate	

Next steps: View recommended plots

```
#Calclate revenue with error handling
def calculate(row):
  try:
    return row["publisher_revenue"] / row["units_sold"]
  except ZeroDivisionError:
    return 0

#Create new column
df["Revenue_per_Unit"] = df.apply(calculate, axis = 1)

#Show update
df.head()
```

shing_Year	Book_Name	Author	language_code	Author_Rating	Book_average_rating	Bo
1975.0	Beowulf	Unknown, Seamus Heaney	en-US	Novice	3.42	
1987.0	Batman: Year One	Frank Miller, David Mazzucchelli, Richmond Lew...	eng	Intermediate	4.23	
2015.0	Go Set a Watchman	Harper Lee	eng	Novice	3.31	
2008.0	When You Are Engulfed in Flames	David Sedaris	en-US	Intermediate	4.04	
2011.0	Daughter of Smoke & Bone	Laini Taylor	eng	Intermediate	4.04	

Next steps: View recommended plots

```
#Generate summary table
table_summary = df.groupby("genre").agg({"Book_average_rating": "mean", "gross_sales": "sum"})

table_summary.reset_index(inplace = True)
```

```
#Show table

print(table_summary)

#Fiction books have a much higher sales while having lowest average rating. The better the rating gets, the less sales it has
```

```
       genre  Book_average_rating  gross_sales
0   children             4.033333     13902.22
1    fiction             4.003529   1744525.46
2 nonfiction             4.022632    228158.87
```

```
#calculate ratio/ create new column
df["Sales_per_Rating"] = df["gross_sales"] / df["Book_ratings_count"]

#Show update
df.head()

#The higher the spr means that the rating contibutes more towards book sales.
#The lower the spr means that it could have been overpriced for the rating.
```

	index	Publishing_Year	Book_Name	Author	language_code	Author_Rating	Book_av
0	0	1975.0	Beowulf	Unknown, Seamus Heaney	en-US	Novice	
1	1	1987.0	Batman: Year One	Frank Miller, David Mazzucchelli, Richmond Lew...	eng	Intermediate	
2	2	2015.0	Go Set a Watchman	Harper Lee	eng	Novice	
3	3	2008.0	When You Are Engulfed in Flames	David Sedaris	en-US	Intermediate	
4	4	2011.0	Daughter of Smoke & Bone	Laini Taylor	eng	Intermediate	

```
#Find all categories of author rating
cats = df.groupby("Author_Rating").sum()

print(cats["units_sold"])
```

```
Author_Rating
Excellent       4828717
Famous           349796
Intermediate    4963160
Novice           212696
Name: units_sold, dtype: int64
```

```
#Calculate/create metric

#Change category to numerical
maprat = {
    "Novice": 1,
    "Intermediate": 2,
    "Famous": 3,
    "Excellent": 4
    }

#Calculate netric
def calculateae(row):
  effec = (
      .2 * maprat[row["Author_Rating"]] +
      .4 * row["Book_average_rating"] +
      .4 * row["units_sold"]
  )
  return effec

#Create new column
```

```
df["Author_Effectiveness"] = df.apply(calculateae, axis = 1)

df.head()

#I chose this formula to be less heavy of status of author, this gives more so result to actual numbers rather thatn enflated popularity. The
```

	index	Publishing_Year	Book_Name	Author	language_code	Author_Rating	Book_av
0	0	1975.0	Beowulf	Unknown, Seamus Heaney	en-US	Novice	
1	1	1987.0	Batman: Year One	Frank Miller, David Mazzucchelli, Richmond Lew...	eng	Intermediate	
2	2	2015.0	Go Set a Watchman	Harper Lee	eng	Novice	
3	3	2008.0	When You Are Engulfed in Flames	David Sedaris	en-US	Intermediate	
4	4	2011.0	Daughter of Smoke & Bone	Laini Taylor	eng	Intermediate	

```
#new initial column first letter from bookname
df["Initial"] = df["Book_Name"].str[0]

#Group sales/rating/initials
isum = df.groupby("Initial").agg({"Book_average_rating": "mean", "gross_sales": "sum", "units_sold": "sum"})

#sort
isumsort = isum.sort_values(by = "Book_average_rating", ascending = False)

#Display
print(isumsort)

#Uncommon real world letters seem to have better ratings
```

```
         Book_average_rating  gross_sales  units_sold
Initial
é                   4.397500      2377.92       55539
æ                   4.360000       666.65         335
'                   4.250000       770.24       30672
Q                   4.225000       464.67       64368
                    4.206667      3263.10        9396
Đ                   4.128000     10171.27        9008
O                   4.083684     30140.12       89097
W                   4.060571     81492.21      375549
Y                   4.060000       339.57       44712
1                   4.060000      7814.40         660
U                   4.047000      5261.72      251675
S                   4.040952    103883.30      641411
P                   4.040370     41952.90      150050
K                   4.040000     13112.83       77121
L                   4.036957     85969.64      403678
A                   4.028133    142548.70      862921
C                   4.026047     92292.82      450241
B                   4.024667     99199.34      370524
ã                   4.023333     13384.68      120199
I                   4.020909     42424.13      150605
N                   4.012778     24500.97      212112
2                   4.010000      1388.77        2862
F                   4.004688     38860.31      272624
G                   3.998696     68210.60      147813
H                   3.990500     85879.43      398463
T                   3.980029    603856.38     3513492
M                   3.972250     80285.41      461467
E                   3.968148     24204.21      178505
R                   3.964167     34873.63      267236
ä                   3.960000       967.68       43767
D                   3.942292    128329.52      426526
J                   3.941111     17310.21       53497
ç                   3.940000       960.29        4240
Z                   3.920000     16706.08        3373
```

```
9                   3.910000      1251.22        3942
V                   3.901429     21216.56       68036
è                   3.820000       594.51        4023
Ã                   3.810000       114.84       31752
Î                   3.785000      2848.01         699
X                   3.770000      1707.15         285
Ø                   3.760000      2328.97        7209
á                   3.600000       106.92         108
```

```
#Remove unknown names
df["Author"] = df["Author"].str.replace("Unknown", "")

#Get First name
def first(authors):
  if authors:
    return [author.split()[0] for author in authors.split(",") if author.strip()]
  else:
    return[]


#Get Last name
def last(authors):
  if authors:
    return [author.split()[-1] for author in authors.split(",") if author.strip()]
  else:
    return []

#Create Columns
df["Author_First_Name"] = df["Author"].apply(first)

df["Author_Last_Name"] = df["Author"].apply(last)

#Case for multiple authors

df["Author_First_Name"] = df["Author_First_Name"].apply(lambda x: ", ".join(x))

df["Author_Last_Name"] = df["Author_Last_Name"].apply(lambda x: ", ".join(x))

#Remove Author Field

df.drop(columns=["Author"], inplace = True)

#Display
df.head()

#There are many differen variables contributing to this, there are myltiple authores, there are spaces, there are commas, we have to take a
```

	index	Publishing_Year	Book_Name	language_code	Author_Rating	Book_average_rating
0	0	1975.0	Beowulf	en-US	Novice	3.42
1	1	1987.0	Batman: Year One	eng	Intermediate	4.23
2	2	2015.0	Go Set a Watchman	eng	Novice	3.31
3	3	2008.0	When You Are Engulfed in Flames	en-US	Intermediate	4.04
4	4	2011.0	Daughter of Smoke & Bone	eng	Intermediate	4.04

5 rows × 21 columns

```
#Calculate Book Age
df["Book_Age"] = 2024 - df["Publishing_Year"]

#Calculate correlation between book age and gross sales
corr = df["Book_Age"].corr(df["gross_sales"])
print("Correlation: ", corr)


#Weak negative correlation between book age and gross sales
```

```
Correlation:  -0.008907243647890048
```

```
#Calculate median uits sold
medus = df["units_sold"].median()

#Filter novice
filnov = df[df["Author_Rating"] == "Novice"]

#Better performing
bpb = filnov[filnov["units_sold"] > medus]

#Display
print(bpb[["Book_Name", "units_sold"]])

#Some factors that can contribute to the selling point of these books are their genre, price, rating, and storytelling
```

```
                                Book_Name  units_sold
0                                 Beowulf        7000
2                      Go Set a Watchman        5500
377                      The Tenth Circle        5940
393                     The Marriage Plot        5697
395                   Luckiest Girl Alive        5670
400                 Not that Kind of Girl        5616
470                             Het diner        4590
508         Pride and Prejudice and Zombies        4347
545                              The Nest        4077
554  Five Point Someone: What Not to Do at IIT        3996
580                              Mr Maybe       61128
643                             The Girls       44928
775                       The Silent Wife       28728
886                 Chasing Harry Winston        4440
890                       A Long Way Down        4440
899                 Everyone Worth Knowing        4400
950                   Her Fearful Symmetry        4280
```

```
#Filter out english books(assuming eng and en-us are seperate/different)
english = df[df["language_code"] == "eng"]

#Calculate english
aer = english["Book_average_rating"].mean()

aes = english["gross_sales"].mean()

#Filter and calculate non english
nenglish = df[df["language_code"] != "eng"]

aner = nenglish["Book_average_rating"].mean()

anes = nenglish["gross_sales"].mean()

#Compare average rating and sells, english us non
print("Average Rating of English Books:", aer)
print("\n")
print("Average Rating of Non-English Books:", aner)
print("\n")
print("Average Sales of English Books:", aes)
print("\n")
print("Average Sales of Non-English Books:", anes)

#Non-English books ourpreform english books

#I am just now seeing that i was only supposed to do 6 :(
```

```
Average Rating of English Books: 4.004177215189873


Average Rating of Non-English Books: 4.014964285714286
Average Sales of English Books: 1850.104683544304


Average Sales of Non-English Books: 1875.0137500000003
```

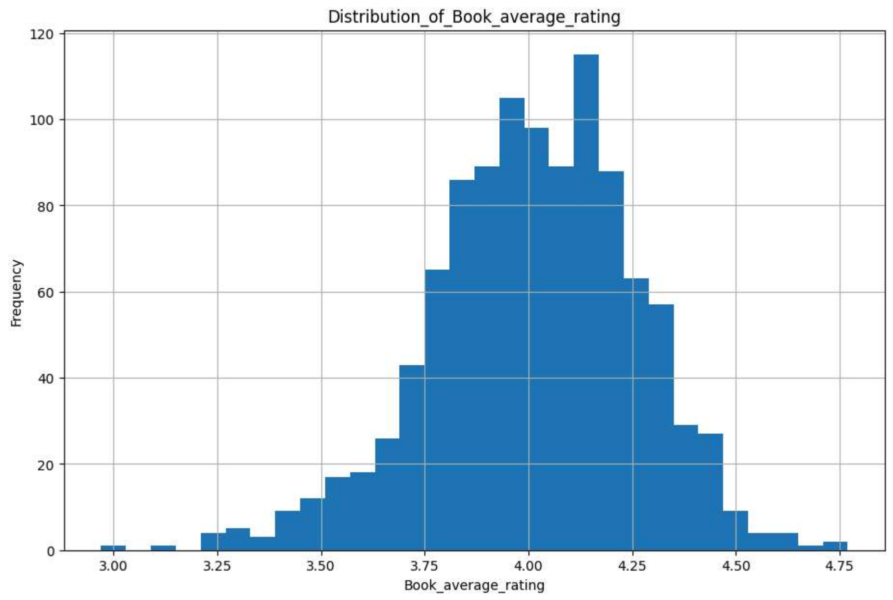
LOAD sqlite

```
import sqlite3

#connect to sqlite
connect = sqlite3.connect("books_database.db")
df.to_sql("books", connect, if_exists = "replace", index = False)
```

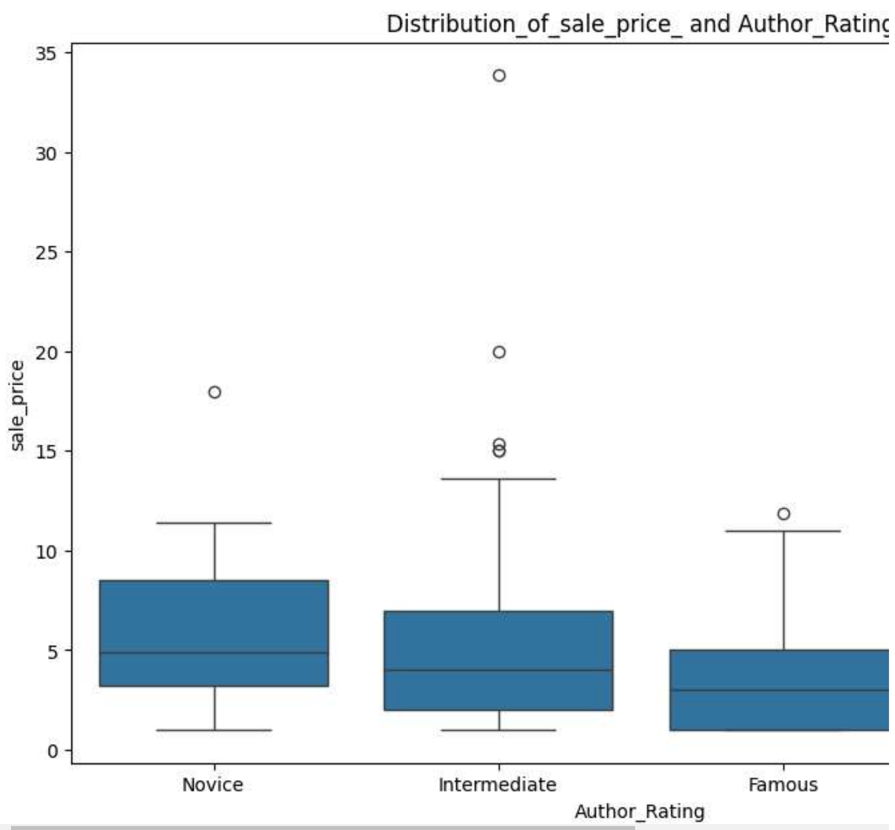
```
1070
```

Analysis and Plot

```
import matplotlib.pyplot as plt

#Read dataFrame
df =pd.read_sql_query("SELECT * FROM books", connect)

#Create Histogram
plt.figure(figsize = (11, 7))
plt.title("Distribution_of_Book_average_rating")
plt.hist(df["Book_average_rating"], bins = 30)
plt.xlabel("Book_average_rating")
plt.ylabel("Frequency")
plt.grid(True)
plt.show()

# There is a slight negative skew, there is above averate rating
```



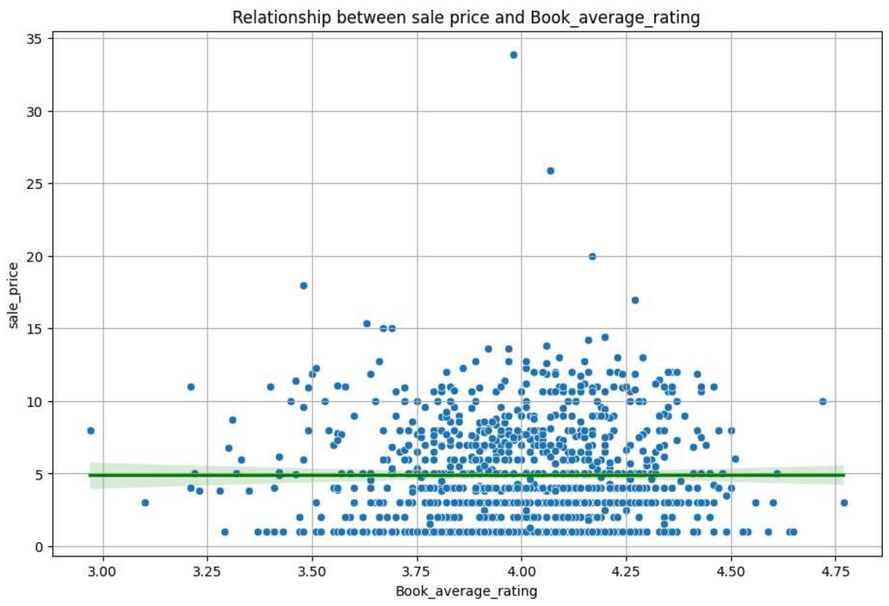
```
import seaborn as sns

#use seaborn to create a boxplot

plt.figure(figsize = (11, 7))
plt.title("Distribution_of_sale_price_ and Author_Rating")
sns.boxplot(x = "Author_Rating", y = "sale_price", data = df)
plt.xlabel("Author_Rating")
plt.ylabel("sale_price")
plt.show()

#Intermediate authors tend to price much higher than any other rating author
```



```
#Correlation Matrix
correm = df[["Book_average_rating", "gross_sales", "units_sold"]].corr()
print(correm)

#There is weak negative correlations between these values
```

```
                     Book_average_rating  gross_sales  units_sold
Book_average_rating             1.000000    -0.042240   -0.008516
gross_sales                    -0.042240     1.000000   -0.150592
units_sold                     -0.008516    -0.150592    1.000000
```

```
#Create scatterplot with seaborn

plt.figure(figsize = (11, 7))
plt.title("Relationship between sale price and Book_average_rating")
sns.scatterplot(data = df, x = "Book_average_rating", y = "sale_price")
sns.regplot(data = df, x = "Book_average_rating", y = "sale_price", scatter = False, color = "green")
plt.xlabel("Book_average_rating")
plt.ylabel("sale_price")
plt.grid(True)
plt.show()

#There does not seem to be a ture relationship, Price does not seem to have much effect on rating
```



```
#Calculate IQR

Q1 = df["Book_ratings_count"].quantile(0.25)
Q3 = df["Book_ratings_count"].quantile(0.75)

IQR = Q3 - Q1

#Define outliers
low = Q1 - 1.5 * IQR
upp = Q3 + 1.5 * IQR

outliers = df[(df["Book_ratings_count"] < low) | (df["Book_ratings_count"] > upp)]

#Analyze
print(outliers)
```

```
         index  Publishing_Year                       Book_Name language_code  \
    4        4           2011.0        Daughter of Smoke & Bone           eng
    8        8           2012.0                        Hopeless           eng
    9        9           1905.0                A Little Princess           eng
    10      10           2004.0          The Truth About Forever         en-US
    11      11           1954.0              The horse and his boy           eng
    12      12           2010.0                  Last Sacrifice            eng
    13      13           1935.0         Little House on the Prairie           eng
    27      27           2004.0                 Dead to the World           eng
    32      32           2003.0                        Club Dead         en-GB
    50      50           2013.0                          Scarlet           eng
    52      52           2011.0                         Silence            eng
    105    105           2011.0                             None           eng
    112    112           2008.0   Chosen: A House of Night Novel         en-US

        Author_Rating  Book_average_rating  Book_ratings_count    genre  \
    4    Intermediate                 4.04              198283  fiction
    8    Intermediate                 4.34              189938  fiction
    9    Intermediate                 4.20              199872  fiction
    10   Intermediate                 4.13              179415  fiction
    11   Intermediate                 3.90              189671  fiction
    12         Famous                 4.42              206792  fiction
    13   Intermediate                 4.18              195424  fiction
    27   Intermediate                 4.13              199572  fiction
    32   Intermediate                 4.03              181323  fiction
    50   Intermediate                 4.30              193766  fiction
    52   Intermediate                 4.16              190722  fiction
    105  Intermediate                 4.30              188136  fiction
    112  Intermediate                 3.90              180961  fiction
```

```
         gross_sales  publisher_revenue  ...                     Publisher_  \
    4       37952.50          22771.500  ...         Penguin Group (USA) LLC
    8       26093.67          15656.202  ...        HarperCollins Publishers
    9       23792.34          14275.404  ...                Random House LLC
    10      17964.00              0.000  ...  Amazon Digital Services,  Inc.
    11      21564.00          12938.400  ...         Penguin Group (USA) LLC
    12       3431.34              0.000  ...  Amazon Digital Services,  Inc.
    13       6897.34           4138.404  ...        HarperCollins Publishers
    27       2376.00              0.000  ...  Amazon Digital Services,  Inc.
    32      13178.00           7906.800  ...        HarperCollins Publishers
    50       1720.62           1032.372  ...  Amazon Digital Services,  Inc.
    52       8517.93              0.000  ...  Amazon Digital Services,  Inc.
    105      7670.40           4602.240  ...         Penguin Group (USA) LLC
    112      7759.20           4655.520  ...  Amazon Digital Services,  Inc.

         units_sold Missing_Book_Name  Revenue_per_Unit  Sales_per_Rating  \
    4          4750                 0             4.794          0.191406
    8          3733                 0             4.194          0.137380
    9          3666                 0             3.894          0.119038
    10         3600                 0             0.000          0.100125
    11         3600                 0             3.594          0.113692
    12         3466                 0             0.000          0.016593
    13         3466                 0             1.194          0.035294
    27         2400                 0             0.000          0.011905
    32         2200                 0             3.594          0.072677
    50         1738                 0             0.594          0.008880
    52         1707                 0             0.000          0.044661
    105         960                 1             4.794          0.040771
```

```
#outlier ext

outlierg = outliers.groupby("genre").size()

outliera = outliers.groupby("Author_First_Name").size()

print("Outliers by Genre:", outlierg)
print("\n")
print("Outliers by Author:", outliera)
#These anomolies may occur because of special occations such as populatirty or advertisement
```

```
Outliers by Genre: genre
fiction    13
dtype: int64


Outliers by Author: Author_First_Name
Becca             1
C.S.              1
Charlaine         2
Colleen           1
Frances, Nancy    1
Laini             1
Laura, Garth      1
Marissa           1
Mark              1
P.C., Kristin     1
Richelle          1
Sarah             1
dtype: int64
```

Backup

```
from datetime import datetime
import shutil
#Log message with date and time
def log(message):
  timestamp = datetime.now().strftime("%y-%m-%d %H:%M:%S")
  print(f"[{timestamp}] {message}")

def backup_database(database_file, backup_file):
  try:
    shutil.copy(database_file, backup_file)
```