best-sellers-review

December 17, 2023

[1]:
```python
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
import plotly.express as px
from plotly.subplots import make_subplots
from datetime import datetime
```

```
C:\Users\Soubhik\anaconda3\Anaconda\lib\site-
packages\pandas\core\computation\expressions.py:21: UserWarning: Pandas requires
version '2.8.0' or newer of 'numexpr' (version '2.7.3' currently installed).
  from pandas.core.computation.check import NUMEXPR_INSTALLED
C:\Users\Soubhik\anaconda3\Anaconda\lib\site-
packages\pandas\core\arrays\masked.py:62: UserWarning: Pandas requires version
'1.3.4' or newer of 'bottleneck' (version '1.3.2' currently installed).
  from pandas.core import (
```

[2]:
```python
best_seller_data = pd.read_csv("bestsellers with categories.csv")
best_seller_data.head(5)
```

[2]:
```
                                               Name  \
0                      10-Day Green Smoothie Cleanse
1                                  11/22/63: A Novel
2            12 Rules for Life: An Antidote to Chaos
3                             1984 (Signet Classics)
4  5,000 Awesome Facts (About Everything!) (Natio…

                     Author  User Rating  Reviews  Price  Year        Genre
0                  JJ Smith          4.7    17350      8  2016  Non Fiction
1              Stephen King          4.6     2052     22  2011      Fiction
2        Jordan B. Peterson          4.7    18979     15  2018  Non Fiction
3             George Orwell          4.7    21424      6  2017      Fiction
4  National Geographic Kids          4.8     7665     12  2019  Non Fiction
```

[3]:
```python
best_seller_data
```

[3]:
```
                                                  Name  \
0                          10-Day Green Smoothie Cleanse
1                                    11/22/63: A Novel
2              12 Rules for Life: An Antidote to Chaos
3                               1984 (Signet Classics)
4    5,000 Awesome Facts (About Everything!) (Natio…
..                                                 …
545        Wrecking Ball (Diary of a Wimpy Kid Book 14)
546  You Are a Badass: How to Stop Doubting Your Gr…
547  You Are a Badass: How to Stop Doubting Your Gr…
548  You Are a Badass: How to Stop Doubting Your Gr…
549  You Are a Badass: How to Stop Doubting Your Gr…

                       Author  User Rating  Reviews  Price  Year        Genre
0                    JJ Smith          4.7    17350      8  2016  Non Fiction
1                Stephen King          4.6     2052     22  2011      Fiction
2          Jordan B. Peterson          4.7    18979     15  2018  Non Fiction
3               George Orwell          4.7    21424      6  2017      Fiction
4    National Geographic Kids          4.8     7665     12  2019  Non Fiction
..                          …          …        …      …     …            …
545               Jeff Kinney          4.9     9413      8  2019      Fiction
546               Jen Sincero          4.7    14331      8  2016  Non Fiction
547               Jen Sincero          4.7    14331      8  2017  Non Fiction
548               Jen Sincero          4.7    14331      8  2018  Non Fiction
549               Jen Sincero          4.7    14331      8  2019  Non Fiction

[550 rows x 7 columns]
```

[4]:
```
best_seller_data.sample(5)
```

[4]:
```
                                                  Name            Author  \
298  Shred: The Revolutionary Diet: 6 Weeks 4 Inche…  Ian K. Smith M.D.
472  The Total Money Makeover: Classic Edition: A P…        Dave Ramsey
391                               The Going-To-Bed Book     Sandra Boynton
327  The 5 Love Languages: The Secret to Love that …       Gary Chapman
103                         Fear: Trump in the White House       Bob Woodward

     User Rating  Reviews  Price  Year        Genre
298          4.1     2272      6  2013  Non Fiction
472          4.7    11550     10  2019  Non Fiction
391          4.8     5249      5  2017      Fiction
327          4.8    25554      8  2017  Non Fiction
103          4.4     6042      2  2018  Non Fiction
```

[5]:
```
best_seller_data.shape
```

[5]:
```
(550, 7)
```

0.1 Give an info of the data

[6]:
```
best_seller_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 550 entries, 0 to 549
Data columns (total 7 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   Name         550 non-null    object
 1   Author       550 non-null    object
 2   User Rating  550 non-null    float64
 3   Reviews      550 non-null    int64
 4   Price        550 non-null    int64
 5   Year         550 non-null    int64
 6   Genre        550 non-null    object
dtypes: float64(1), int64(3), object(3)
memory usage: 30.2+ KB
```

Understanding Univariate Analysis

Univariate analysis involves the examination of a single variable in isolation. It allows us to gain insights into the distribution, central tendency, and variability of that variable. Non-graphical univariate analysis techniques are particularly useful when dealing with categorical or discrete data, or when a simplified summary is needed.

Value Counts

Value counts are a straightforward and informative way to analyze the distribution of categorical data. This technique involves counting the occurrences of each unique category or value within a variable. Python's Pandas library provides a convenient method for this: value_counts().

[7]:
```
best_seller_data.isna().sum()
```

[7]:
```
Name           0
Author         0
User Rating    0
Reviews        0
Price          0
Year           0
Genre          0
dtype: int64
```

[8]:
```
best_seller_data.duplicated().sum()
```

[8]:
```
0
```

[9]:
```
best_seller_data.nunique()
```

[9]:
```
Name           351
Author         248
User Rating     14
Reviews        346
Price           40
Year            11
Genre            2
dtype: int64
```

[10]:
```python
best_seller_data["Year"].value_counts()
```

[10]:
```
Year
2016    50
2011    50
2018    50
2017    50
2019    50
2014    50
2010    50
2009    50
2015    50
2013    50
2012    50
Name: count, dtype: int64
```

[11]:
```python
best_seller_data["Year"].value_counts(normalize=True)
```

[11]:
```
Year
2016    0.090909
2011    0.090909
2018    0.090909
2017    0.090909
2019    0.090909
2014    0.090909
2010    0.090909
2009    0.090909
2015    0.090909
2013    0.090909
2012    0.090909
Name: proportion, dtype: float64
```

[12]:
```python
best_seller_data["Reviews"].value_counts(normalize=True)
```

[12]:
```
Reviews
8580     0.018182
5069     0.016364
21834    0.014545
```

```
19546    0.012727
19576    0.010909
           …
5272     0.001818
3776     0.001818
1930     0.001818
13471    0.001818
5680     0.001818
Name: proportion, Length: 346, dtype: float64
```

Binning

Binning is a technique used to convert continuous or numerical data into categorical or discrete intervals (bins or buckets). This simplification allows for easier analysis and interpretation of data. Binning can help reveal patterns or trends in the data distribution.

Example: Let's say you have a dataset containing the ages of customers. To perform a binning analysis, you can group these ages into age ranges (bins):

[13]:
```
bins = (0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100)
best_seller_data["Price"].value_counts(bins = bins)
```

[13]:
```
(-0.001, 10.0]    266
(10.0, 20.0]      216
(20.0, 30.0]       42
(40.0, 50.0]       11
(30.0, 40.0]        9
(50.0, 60.0]        3
(80.0, 90.0]        1
(60.0, 70.0]        0
(70.0, 80.0]        0
(90.0, 100.0]       0
Name: count, dtype: int64
```

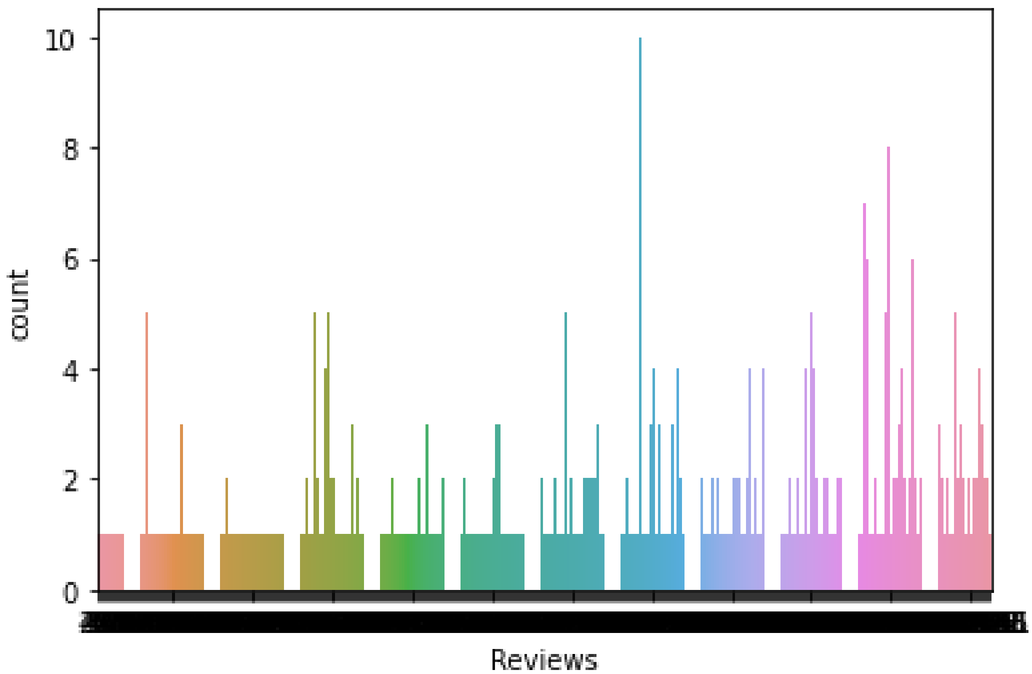
[14]:
```
max_price = best_seller_data["Price"].max()
print("Max price : ", max_price)
min_price = best_seller_data["Price"].min()
print("Min price : ", min_price)
mean_price = best_seller_data["Price"].mean()
print("Mean price : ", mean_price)
dev_price = best_seller_data["Price"].std()
print("Deviation in price : ", dev_price)
kurt_price = best_seller_data["Price"].kurt()
print("Kurtosis : ", kurt_price)
```

```
Max price :  105
Min price :  0
Mean price :  13.1
```

```
Deviation in price :  10.84226197842236
Kurtosis :  22.43352032785043
```

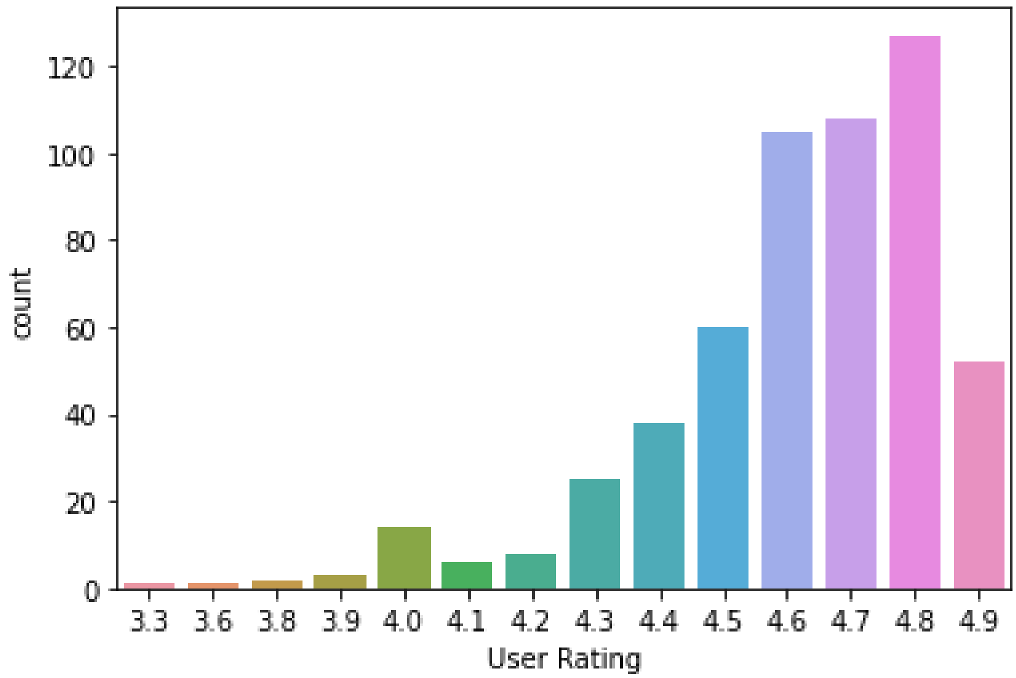
[15]:
```python
import seaborn as sns

sns.countplot(data = best_seller_data, x ="Reviews")
```

[15]: <AxesSubplot:xlabel='Reviews', ylabel='count'>



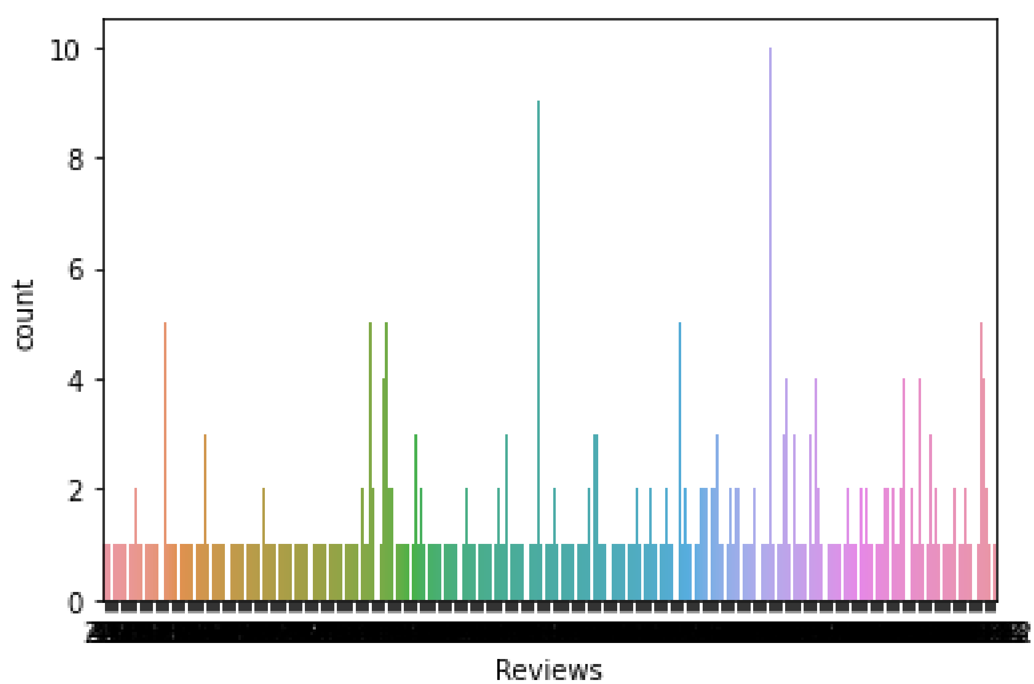
[16]:
```python
sns.countplot(data = best_seller_data, x ="User Rating")
```

[16]: <AxesSubplot:xlabel='User Rating', ylabel='count'>



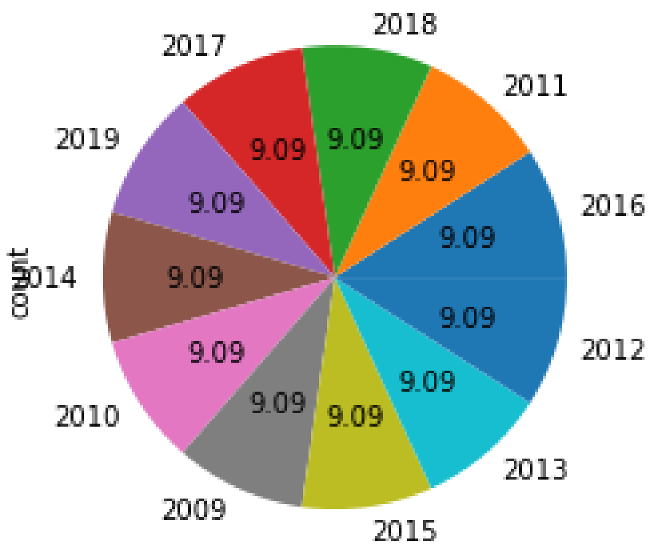
[17]:
```
sns.countplot(data = best_seller_data[best_seller_data["Reviews"]<15000], x =␣
↪"Reviews")
```

[17]: <AxesSubplot:xlabel='Reviews', ylabel='count'>



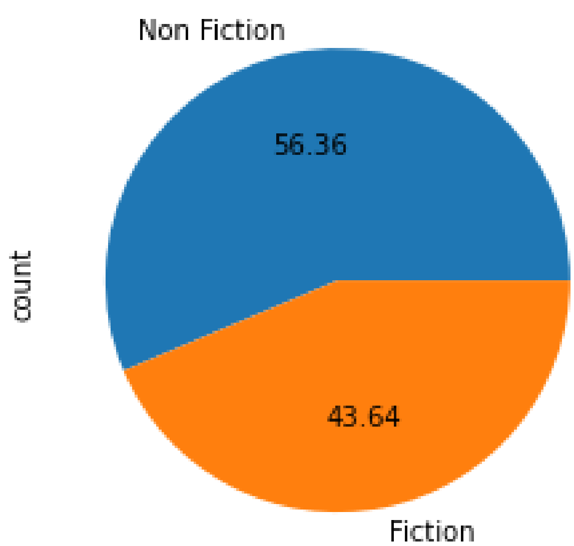
[18]:
```
best_seller_data["Year"].value_counts().plot(kind = "pie", autopct = "%.2f")
```

[18]: <AxesSubplot:ylabel='count'>



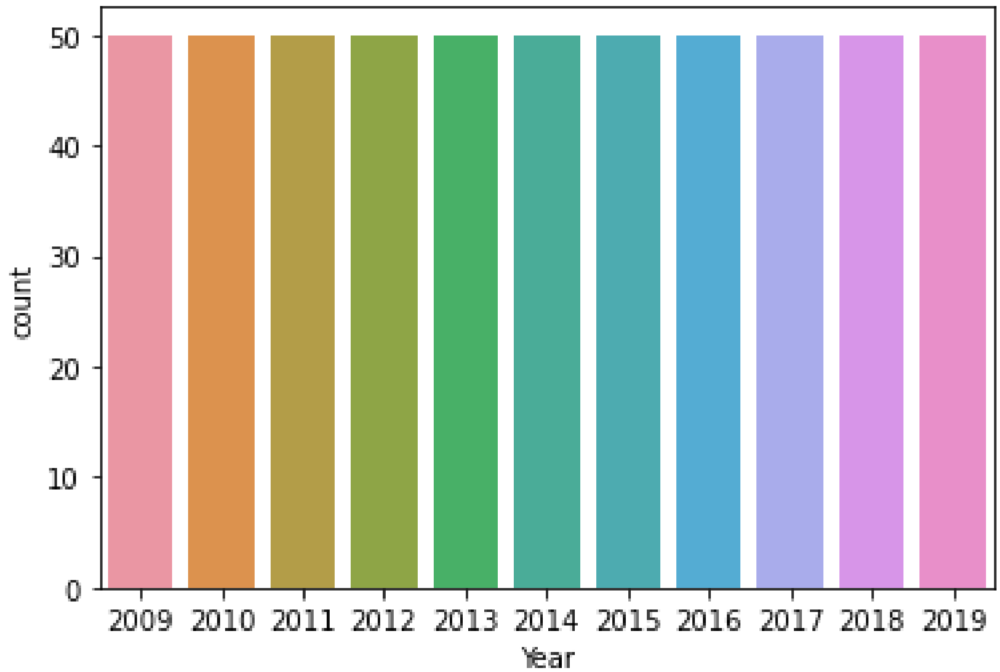
[19]:
```
best_seller_data["Genre"].value_counts().plot(kind = "pie", autopct = "%.2f")
```

[19]: <AxesSubplot:ylabel='count'>



[20]:
```
sns.countplot(data = best_seller_data, x ="Year")
```

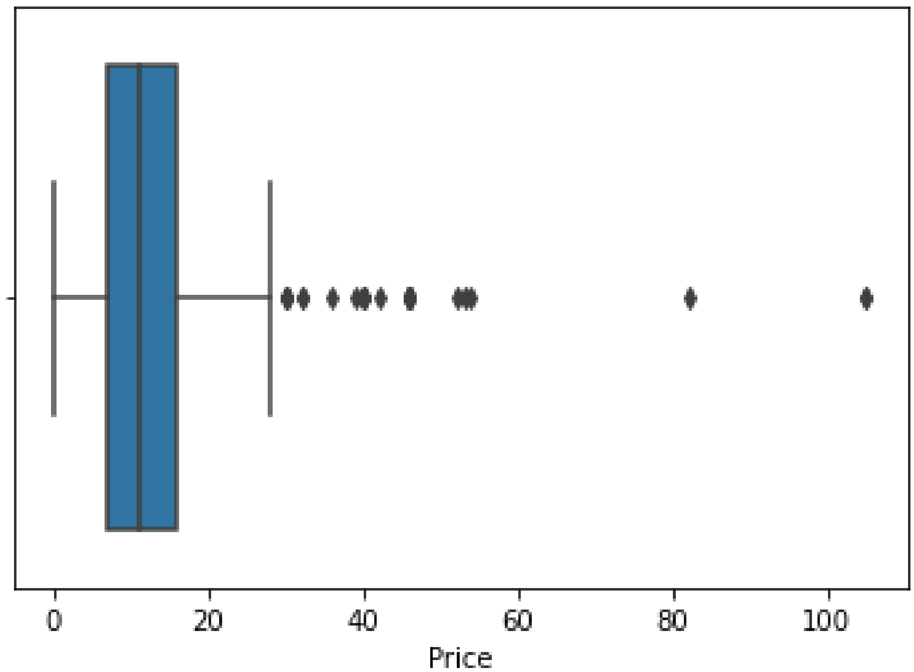
[20]: <AxesSubplot:xlabel='Year', ylabel='count'>



[21]:
```
sns.boxplot(best_seller_data["Price"])
```

```
C:\Users\Soubhik\anaconda3\Anaconda\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variable as a keyword arg: x. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
```

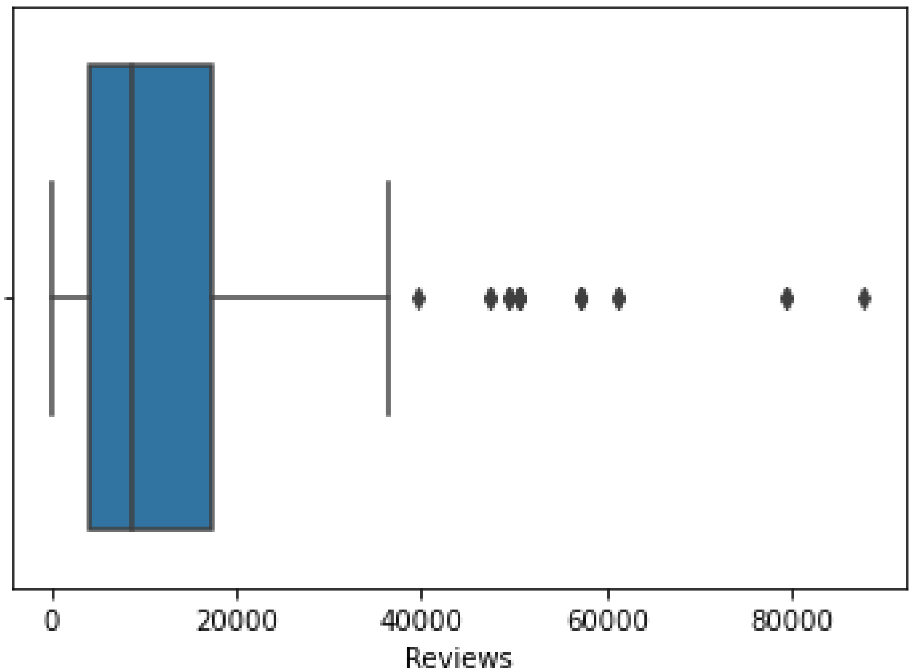
[21]: <AxesSubplot:xlabel='Price'>



[22]:
```
sns.boxplot(best_seller_data["Reviews"])
```

```
C:\Users\Soubhik\anaconda3\Anaconda\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variable as a keyword arg: x. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
```

[22]:
```
<AxesSubplot:xlabel='Reviews'>
```



[23]:
```
print(best_seller_data["Year"].dtype)
```

```
int64
```

[24]:
```
best_seller_data.describe()
```

[24]:

	User Rating	Reviews	Price	Year
count	550.000000	550.000000	550.000000	550.000000
mean	4.618364	11953.281818	13.100000	2014.000000
std	0.226980	11731.132017	10.842262	3.165156
min	3.300000	37.000000	0.000000	2009.000000
25%	4.500000	4058.000000	7.000000	2011.000000
50%	4.700000	8580.000000	11.000000	2014.000000
75%	4.800000	17253.250000	16.000000	2017.000000
max	4.900000	87841.000000	105.000000	2019.000000

[25]:
```
best_seller_data.drop(["Name","Reviews","User Rating","Price"], inplace = True,␣
↪axis = 1)
```

[26]:
```
best_seller_data.head(10)
```

[26]:

	Author	Year	Genre
0	JJ Smith	2016	Non Fiction
1	Stephen King	2011	Fiction
2	Jordan B. Peterson	2018	Non Fiction

```
3             George Orwell  2017      Fiction
4  National Geographic Kids  2019  Non Fiction
5        George R. R. Martin  2011      Fiction
6        George R. R. Martin  2014      Fiction
7                Amor Towles  2017      Fiction
8                James Comey  2018  Non Fiction
9           Fredrik Backman  2016      Fiction
```

[27]:
```python
yearwise_bestseller = best_seller_data.groupby(by = "Year")
yearwise_bestseller
```

[27]:
```
<pandas.core.groupby.generic.DataFrameGroupBy object at 0x000002733220C940>
```

[28]:
```python
print(yearwise_bestseller)
```

```
<pandas.core.groupby.generic.DataFrameGroupBy object at 0x000002733220C940>
```

GroupBy

Groupby is a pretty simple concept. We can create a grouping of categories and apply a function to the categories. It's a simple concept but it's an extremely valuable technique that's widely used in data science. In real data science projects, you'll be dealing with large amounts of data and trying things over and over, so for efficiency, we use Groupby concept. Groupby concept is really important because it's ability to aggregate data efficiently, both in performance and the amount code is magnificent. Groupby mainly refers to a process involving one or more of the following steps they are:

Splitting : It is a process in which we split data into group by applying some conditions on datasets. Applying : It is a process in which we apply a function to each group independently Combining : It is a process in which we combine different datasets after applying groupby and results into a data structure

[29]:
```python
from sklearn.impute import SimpleImputer
impute = SimpleImputer(missing_values = np.nan , strategy = 'mean')
df = pd.read_csv("bestsellers with categories.csv")
impute.fit(df.iloc[ : , 2:3 ].values)
df.iloc[ : , 2:3 ] = impute.transform(df.iloc[ : , 2:3 ].values)

df = df.dropna()
```

[30]:
```python
print(df.head())
```

```
                                              Name  \
0                        10-Day Green Smoothie Cleanse
1                                 11/22/63: A Novel
2           12 Rules for Life: An Antidote to Chaos
3                            1984 (Signet Classics)
4  5,000 Awesome Facts (About Everything!) (Natio…
```

```
                     Author  User Rating  Reviews  Price  Year        Genre
0                  JJ Smith          4.7    17350      8  2016  Non Fiction
1              Stephen King          4.6     2052     22  2011      Fiction
2        Jordan B. Peterson          4.7    18979     15  2018  Non Fiction
3             George Orwell          4.7    21424      6  2017      Fiction
4  National Geographic Kids          4.8     7665     12  2019  Non Fiction
```

0.2 Q. How many Non-Fiction Books are avilable with user rating above 4.6

[31]:
```python
df_pr = df[df['Genre'] == 'Non Fiction']
print(len(df_pr[df_pr['User Rating'] > 4.6]))
```

```
136
```

0.3 Q. How many Non-Fiction Books are avilable with user rating above 4.6 and Review less than 15k

[32]:
```python
df_pr = df[df['Genre'] == 'Non Fiction']
df_pr = df_pr[df_pr['User Rating'] > 4.5]
df_pr = df_pr[df_pr['Reviews'] > 15000]

print(len(df_pr))
```

```
54
```

0.4 Q. How many Non-Fiction Books are avilable with user rating above 4.6 and Review less than 20k

[33]:
```python
df_pr = df[df['Genre'] == 'Non Fiction']
df_pr = df_pr[df_pr['User Rating'] > 4.6]
df_pr = df_pr[df_pr['Reviews'] > 20000]

print(len(df_pr))
```

```
29
```

0.5 Q. Which Non-Fiction Books are avilable with user rating above 4.6 and Review less than 20k

[34]:
```python
df_pr = df[df['Genre'] == 'Non Fiction']
df_pr = df_pr[df_pr['User Rating'] > 4.6]
df_pr = df_pr[df_pr['Reviews'] > 20000]

print(df_pr['Name'])
```

```
32                                               Becoming
33                                               Becoming
97                                     Educated: A Memoir
```

```
98                                  Educated: A Memoir
166                 How to Win Friends & Influence People
167                 How to Win Friends & Influence People
168                 How to Win Friends & Influence People
169                 How to Win Friends & Influence People
170                 How to Win Friends & Influence People
293    School Zone - Big Preschool Workbook - Ages 4 …
294    School Zone - Big Preschool Workbook - Ages 4 …
325    The 5 Love Languages: The Secret to Love that …
326    The 5 Love Languages: The Secret to Love that …
327    The 5 Love Languages: The Secret to Love that …
328    The 5 Love Languages: The Secret to Love that …
329    The 5 Love Languages: The Secret to Love that …
351    The Boys in the Boat: Nine Americans and Their…
352    The Boys in the Boat: Nine Americans and Their…
375    The Four Agreements: A Practical Guide to Pers…
376    The Four Agreements: A Practical Guide to Pers…
377    The Four Agreements: A Practical Guide to Pers…
378    The Four Agreements: A Practical Guide to Pers…
379    The Four Agreements: A Practical Guide to Pers…
380    The Four Agreements: A Practical Guide to Pers…
515    Unbroken: A World War II Story of Survival, Re…
516    Unbroken: A World War II Story of Survival, Re…
517    Unbroken: A World War II Story of Survival, Re…
518    Unbroken: A World War II Story of Survival, Re…
519    Unbroken: A World War II Story of Survival, Re…
Name: Name, dtype: object
```

0.6 Q. Give the statistical interpretation of the data's major features

[35]:
```
df_pr.describe()
```

[35]:

	User Rating	Reviews	Price	Year
count	29.000000	29.000000	29.000000	29.000000
mean	4.755172	28052.482759	10.206897	2016.000000
std	0.050612	9467.432035	3.609307	2.521338
min	4.700000	23047.000000	6.000000	2010.000000
25%	4.700000	23308.000000	6.000000	2014.000000
50%	4.800000	25001.000000	11.000000	2016.000000
75%	4.800000	28729.000000	12.000000	2018.000000
max	4.800000	61133.000000	16.000000	2019.000000