Data Cleaning - Flipkart

September 23, 2024

1 Filtering And Aggregating Raw Data

1.1 Importing Requisite Libraries

[1]:
```
import numpy as np
import pandas as pd
import re
from bs4 import BeautifulSoup
```

1.2 List Of The Features To Be Extracted From The Raw Data

- laptop_company = []
- processor_company = []
- processor = []
- operating_system = []
- RAM = []
- storage = []
- storage_type = [] # If not SSD Default will be HDD
- rating = []
- No_reviews = []
- screen_size = []
- price = [] # Target column

1.3 Filtering Features From Raw Data

[2]:
```
# Loading the raw CSV
df = pd.read_csv(r"data\raw.csv")
df.head()
```

[2]:

	Unnamed: 0	Raw Data
0	0	<!doctype html><html lang="en"><head><link hre…
1	1	<!doctype html><html lang="en"><head><link hre…
2	2	<!doctype html><html lang="en"><head><link hre…
3	3	<!doctype html><html lang="en"><head><link hre…
4	4	<!doctype html><html lang="en"><head><link hre…

[3]:
```
pages = []
for i,j in df.iterrows():
```

```
    pages.append(j["Raw Data"])
```

[4]:
```
# Iterating over all pages and for each page filtering out the laptop brand for␣
 ↪each product
laptop_brand = []
for page in pages:
    soup = BeautifulSoup(page)
    for i in soup.find_all("div",class_="tUxRFH"):
        regex = re.findall("Compare(\w+)",i.text)
        if regex:
            laptop_brand.append(regex[0])
        else:
            laptop_brand.append(np.nan)
```

[5]:
```
len(laptop_brand)
```

[5]: 1632

[6]:
```
# Filtering out laptop names
laptop_name = []
for page in pages:
    soup = BeautifulSoup(page)
    for i in soup.find_all("div",class_="KzDlHZ"):
        regex = re.findall("(^.+)\s(?:
 ↪Intel|intel|AMD|M1|M2|M3|Chromebook|Snapdragon)",i.text)
        if regex:
            laptop_name.append(regex[0])
        else:
            laptop_name.append(np.nan)
```

[7]:
```
len(laptop_name)
```

[7]: 1632

[8]:
```
processor = []
processor_company = []
for page in pages:
    soup = BeautifulSoup(page)
    for i in soup.find_all("div",class_="KzDlHZ"):

        # Regex to find the processor company of the laptop
        regex1 = re.
 ↪findall("Intel|intel|AMD|M1|M2|M3|Chromebook|Snapdragon",str(i.text))
        if regex1:
            processor_company.append(regex1[0])
        else:
            processor_company.append(np.nan)
```

```
        # Regex to find the exact processor in the laptop
        regex2 = re.findall("(?:
 ↪Intel|intel|AMD|M1|M2|M3|Chromebook|Snapdragon)\s(.+) - ",str(i.text))
        if regex2:
            processor.append(regex2[0])
        else:
            processor.append(np.nan)
```

[9]:
```
# Checking the number of datapoints after applying the regex
print(len(processor_company))
```

1632

[10]:
```
print(len(processor))
```

1632

[11]:
```
# Filtering out operating System of the laptops
operating_system = []
for page in pages:
    soup = BeautifulSoup(page)
    for i in soup.find_all("div",class_="KzDlHZ"):

        # Regex to find Operating Systems
        regex = re.findall(".+((?:Windows 10|Mac OS|DOS|Andorid|Chrome|Windows␣
 ↪11|Windows 11 Home))",str(i.text))

        if regex:
            operating_system.append(regex[0])
        else:
            operating_system.append(np.nan)
```

[12]:
```
# Checking the number of datapoints
len(operating_system)
```

[12]: 1632

[13]:
```
# Filtering out RAM of the laptops
RAM = []
for page in pages:
    soup = BeautifulSoup(page)
    for i in soup.find_all("div",class_="KzDlHZ"):

        # Regex to find RAM
        regex = re.findall("(\d+)\sGB\/",str(i.text))

        if regex:
```

```
            RAM.append(regex[0])
        else:
            RAM.append(np.nan)
```

[14]:
```
len(RAM)
```

[14]: 1632

[15]:
```
# Filtering out Storage of the laptops
storage = []
for page in pages:
    soup = BeautifulSoup(page)
    for i in soup.find_all("div",class_="KzDlHZ"):

        # Regex to find Storage Size
        regex = re.findall("\d+\sGB\/(\d+)\s(?:GB|TB)\s(?:SSD|HDD|EMMC)",str(i.
 ↪text))

        if regex:
            storage.append(regex[0])
        else:
            storage.append(np.nan)
```

[16]:
```
len(storage)
```

[16]: 1632

[17]:
```
# Filtering out Storage type of the laptops
storage_type = []
for page in pages:
    soup = BeautifulSoup(page)
    for i in soup.find_all("div",class_="KzDlHZ"):

        # Regex to find Storage type
        regex = re.findall("\d+\sGB\/\d+\s(?:GB|TB)\s((?:SSD|HDD|EMMC))",str(i.
 ↪text))

        if regex:
            storage_type.append(regex[0])
        else:
            storage_type.append(np.nan)
```

[18]:
```
len(storage_type)
```

[18]: 1632

[19]:
```
# Filtering out Rating of the laptops
rating = []
for page in pages:
    soup = BeautifulSoup(page)
    for i in soup.find_all("div",class_="tUxRFH"):

        r = i.find("div",class_="XQDdHH")

        if r:
            rating.append(r.text)
        else:
            rating.append(np.nan)
```

[20]:
```
len(rating)
```

[20]: 1632

[21]:
```
# Filtering out number of reviews for each laptops
No_reviews = []
for page in pages:
    soup = BeautifulSoup(page)
    for i in soup.find_all("div",class_="tUxRFH"):
        p =i.find("span",class_="Wphh3N")
        if p:
            regex = re.findall("\&\s(.+)\sReviews",p.text)
            if regex:
                No_reviews.append(regex[0])
            else:
                No_reviews.append(np.nan)
        else:
            No_reviews.append(np.nan)
```

[22]:
```
len(No_reviews)
```

[22]: 1632

[23]:
```
# Filtering out Screen Size of the laptops
screen_size = []
for page in pages:
    soup = BeautifulSoup(page)
    for i in soup.find_all("div",class_="_6NESgJ"):
        regex = re.findall("\d+?\.?\d+?",i.text)
        if regex:
            screen_size.append(regex[0])
        else:
            screen_size.append(np.nan)
```

[24]:
```
len(screen_size)
```

[24]: 1632

[25]:
```
# Filtering out the target column: price of the laptop
price = []
for page in pages:
    soup = BeautifulSoup(page)
    for i in soup.find_all("div",class_="tUxRFH"):

        p =i.find("div",class_="Nx9bqj _4b5DiR")

        if p:
            price.append(p.text)
        else:
            price.append(np.nan)
```

[26]:
```
len(price)
```

[26]: 1632

[27]:
```
# Creating a dataframe from all the extracted features present in list
feature_dict = {"Laptop_Brand":laptop_brand,
                "Laptop_Name":laptop_name,
                "Processor_Company":processor_company,
                "Processor":processor,
                "Operating_System":operating_system,
                "RAM":RAM,
                "Storage":storage,
                "Storage_Type":storage_type,
                "Screen_Size":screen_size,
                "Rating":rating,
                "Number_of_Reviews": No_reviews,
                "Price":price}
```

[28]:
```
# Creating a dataframe from the above dictionary
laptop_df = pd.DataFrame(feature_dict)
```

[29]:
```
# Saving the dataframe as a csv file for further analysis
laptop_df.to_csv(r"data\flipkart_laptop_data.csv",index=False)
```

2 Cleaning The Aggregated Data

2.1 Loading the Necessary Modules

[30]:
```
import numpy as np
import pandas as pd
import re
import warnings
warnings.filterwarnings("ignore")
```

2.2 Loading the CSV file

[31]:
```
laptop_df = pd.read_csv(r"data\flipkart_laptop_data.csv")
laptop_df.head()
```

[31]:

	Laptop_Brand	Laptop_Name	Processor_Company	Processor	Operating_System	RAM	Storage	Storage_Type	Screen_Size	Rating	Number_of_Reviews	Price
0	HP	HP Victus	Intel	Core i5 12th Gen	Windows 11	16	512	SSD	12	4.4	38.0	58,990
1	MSI	MSI Thin 15	Intel	Core i5 12th Gen 12450H	Windows 11	16	512	SSD	12	4.3	34.0	57,990
2	HP	HP Laptop	AMD	Ryzen 3 Quad Core 5300U	Windows 11	8	512	SSD	11	4.3	482.0	30,999
3	Acer	Acer One	Intel	Core i3 11th Gen 1115G4	Windows 11	8	512	SSD	11	4.2	571.0	26,990
4	HP	HP	AMD	Ryzen 5 Hexa Core 5500U	Windows 11	16	512	SSD	16	4.3	268.0	42,990

2.3 Cleaning Data

[32]:
```
laptop_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1632 entries, 0 to 1631
Data columns (total 12 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Laptop_Brand       1632 non-null   object
 1   Laptop_Name        1632 non-null   object
 2   Processor_Company  1632 non-null   object
```

```
 3   Processor          1632 non-null   object
 4   Operating_System   1632 non-null   object
 5   RAM                1632 non-null   int64
 6   Storage            1632 non-null   int64
 7   Storage_Type       1632 non-null   object
 8   Screen_Size        1632 non-null   int64
 9   Rating             1503 non-null   float64
 10  Number_of_Reviews  1503 non-null   float64
 11  Price              1623 non-null   object
dtypes: float64(2), int64(3), object(7)
memory usage: 153.1+ KB
```

2.3.1 Observation

- There are a total of `1632` datapoints but looking at the columns `Rating` and `Number_of_Reviews`, there are some null values which needs to be dealt with later.
- Need to check for wrong values or outliers in the data.
- The target column `Price` should be integer but is stored as an object so it must be converted to right datatype as well as the missing data needs to be replaced.

[33]:
```
# Inspecting Price Column
laptop_df["Price"].head(10)
```

[33]:
```
0    58,990
1    57,990
2    30,999
3    26,990
4    42,990
5    64,990
6    52,990
7    35,990
8    20,990
9    36,990
Name: Price, dtype: object
```

- Based on the above cell output, we see that Price is being treated as object column becuase of an extra symbol and ,.
- Therefore they need to be removed as they dont contribute in the analysis.

[34]:
```
# Removing extra characters from the price column
laptop_df["Price"] = laptop_df["Price"].str.replace(',','').str.replace(' ', '')
laptop_df.head()
```

[34]:

	Laptop_Brand	Laptop_Name	Processor_Company	Processor
0	HP	HP Victus	Intel	Core i5 12th Gen
1	MSI	MSI Thin 15	Intel	Core i5 12th Gen 12450H
2	HP	HP Laptop	AMD	Ryzen 3 Quad Core 5300U
3	Acer	Acer One	Intel	Core i3 11th Gen 1115G4
4	HP	HP	AMD	Ryzen 5 Hexa Core 5500U

```
  Operating_System  RAM  Storage Storage_Type  Screen_Size  Rating  \
0       Windows 11   16      512          SSD           12     4.4
1       Windows 11   16      512          SSD           12     4.3
2       Windows 11    8      512          SSD           11     4.3
3       Windows 11    8      512          SSD           11     4.2
4       Windows 11   16      512          SSD           16     4.3

   Number_of_Reviews  Price
0               38.0  58990
1               34.0  57990
2              482.0  30999
3              571.0  26990
4              268.0  42990
```

[35]:
```
# Analysing the complete description summary of the dataframe
laptop_df.describe(include="all").T
```

[35]:
```
                    count unique                    top  freq        mean  \
Laptop_Brand         1632      9                     HP   388         NaN
Laptop_Name          1632     49                     HP   136         NaN
Processor_Company    1632      4                  Intel  1040         NaN
Processor            1632     38  Core i3 12th Gen 1215U   182         NaN
Operating_System     1632      2              Windows 11  1541         NaN
RAM                1632.0    NaN                    NaN   NaN   12.252451
Storage            1632.0    NaN                    NaN   NaN  440.907475
Storage_Type         1632      2                    SSD  1541         NaN
Screen_Size        1632.0    NaN                    NaN   NaN   19.865809
Rating             1503.0    NaN                    NaN   NaN    4.203127
Number_of_Reviews  1503.0    NaN                    NaN   NaN  246.401863
Price                1623     53                  54990   100         NaN

                          std   min    25%    50%    75%    max
Laptop_Brand              NaN   NaN    NaN    NaN    NaN    NaN
Laptop_Name               NaN   NaN    NaN    NaN    NaN    NaN
Processor_Company         NaN   NaN    NaN    NaN    NaN    NaN
Processor                 NaN   NaN    NaN    NaN    NaN    NaN
Operating_System          NaN   NaN    NaN    NaN    NaN    NaN
RAM                  5.865025   4.0    8.0    8.0   16.0   32.0
Storage            171.443332   1.0  512.0  512.0  512.0  512.0
Storage_Type              NaN   NaN    NaN    NaN    NaN    NaN
Screen_Size         18.555163  11.0   11.0   12.0   16.0   81.0
Rating               0.238804   3.3    4.1    4.2    4.3    5.0
Number_of_Reviews  202.948019   0.0   25.0  214.0  467.0  597.0
Price                     NaN   NaN    NaN    NaN    NaN    NaN
```

2.3.2 Observation Based Tasks:

- Convert the price column to int after handling missing values.
- Convert the laptop names to proper product names
- In the storage column, there is a min vlaue of 1, which is measuring the data in TB, so all the values must be converted to a common GB measurement.
- Check the screen size for values and impute the outliers accordingly.
- In rating and no of reviews replace null values with 0.

[36]:
```
# Analyzing the datapoints which have null price
laptop_df[laptop_df["Price"].isnull()]
```

[36]:

	Laptop_Brand	Laptop_Name	Processor_Company	Processor	Operating_System	RAM	Storage	Storage_Type	Screen_Size	Rating	Number_of_Reviews	Price
25	Lenovo	Lenovo LOQ	Intel	Core i5 13th Gen 13450HX	Windows 11	16	512	SSD	13	4.2	44.0	NaN
145	Lenovo	Lenovo LOQ	Intel	Core i5 13th Gen 13450HX	Windows 11	16	512	SSD	13	4.2	44.0	NaN
217	Lenovo	Lenovo LOQ	Intel	Core i5 13th Gen 13450HX	Windows 11	16	512	SSD	13	4.2	44.0	NaN
241	Lenovo	Lenovo LOQ	Intel	Core i5 13th Gen 13450HX	Windows 11	16	512	SSD	13	4.2	44.0	NaN
265	Lenovo	Lenovo LOQ	Intel	Core i5 13th Gen 13450HX	Windows 11	16	512	SSD	13	4.2	44.0	NaN
641	Lenovo	Lenovo LOQ	Intel	Core i5 13th Gen 13450HX	Windows 11	16	512	SSD	13	4.2	44.0	NaN
1457	Lenovo	Lenovo LOQ	Intel	Core i5 13th Gen 13450HX	Windows 11	16	512	SSD	13	4.2	44.0	NaN
1553	Lenovo	Lenovo LOQ	Intel	Core i5 13th Gen 13450HX	Windows 11	16	512	SSD	13	4.2	44.0	NaN
1577	Lenovo	Lenovo LOQ	Intel	Core i5 13th Gen 13450HX	Windows 11	16	512	SSD	13	4.2	44.0	NaN

- All the above data points are duplicated, so will be dropped.
- The process of handling will be as follows:
  - Removing all the null values in Price column excpet for the 1 datapoint.
  - Replacing the null value in Price with the mean of 50th percentile and 75th percentile

of all Lenovo Laptop Prices.

[37]:
```
# Step 1: Drop 8 rows where Laptop_Name is 'Lenovo LOQ' and Price is null
lenovo_loq_null_price = laptop_df[(laptop_df['Laptop_Name'] == 'Lenovo LOQ') &␣
 ↪(laptop_df['Price'].isnull())]
laptop_df = laptop_df.drop(lenovo_loq_null_price.index[:8])

# Step 2: Calculate the mean of the 50th and 75th percentiles of the Price␣
 ↪column for Lenovo laptops
lenovo_prices = laptop_df[laptop_df['Laptop_Brand'] == 'Lenovo']['Price'].
 ↪dropna().astype(float)
percentile_50 = lenovo_prices.quantile(0.50)
percentile_75 = lenovo_prices.quantile(0.75)
mean_price = (percentile_50 + percentile_75) / 2

# Step 3: Replace the null value in the remaining 'Lenovo LOQ' row with the␣
 ↪calculated mean price
laptop_df.loc[(laptop_df['Laptop_Name'] == 'Lenovo LOQ') & (laptop_df['Price'].
 ↪isnull()), 'Price'] = mean_price

# Verifying the changes
laptop_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 1624 entries, 0 to 1631
Data columns (total 12 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Laptop_Brand       1624 non-null   object
 1   Laptop_Name        1624 non-null   object
 2   Processor_Company  1624 non-null   object
 3   Processor          1624 non-null   object
 4   Operating_System   1624 non-null   object
 5   RAM                1624 non-null   int64
 6   Storage            1624 non-null   int64
 7   Storage_Type       1624 non-null   object
 8   Screen_Size        1624 non-null   int64
 9   Rating             1495 non-null   float64
 10  Number_of_Reviews  1495 non-null   float64
 11  Price              1624 non-null   object
dtypes: float64(2), int64(3), object(7)
memory usage: 164.9+ KB
```

[38]:
```
# Converting Price column to numbers
laptop_df["Price"] = laptop_df["Price"].astype("float")
laptop_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
```

```
Index: 1624 entries, 0 to 1631
Data columns (total 12 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Laptop_Brand       1624 non-null   object
 1   Laptop_Name        1624 non-null   object
 2   Processor_Company  1624 non-null   object
 3   Processor          1624 non-null   object
 4   Operating_System   1624 non-null   object
 5   RAM                1624 non-null   int64
 6   Storage            1624 non-null   int64
 7   Storage_Type       1624 non-null   object
 8   Screen_Size        1624 non-null   int64
 9   Rating             1495 non-null   float64
 10  Number_of_Reviews  1495 non-null   float64
 11  Price              1624 non-null   float64
dtypes: float64(3), int64(3), object(6)
memory usage: 164.9+ KB
```

[39]:
```python
# Removing the brand name from Laptop Name to just have product name
def remove_company(name):
    words = name.split(' ',1)

    if len(words)>1:
        # Removing anything which comes after , or -
        trunc = words[1].split(',')[0].split('-')[0].strip()
        return trunc if trunc else name

    return name

laptop_df["Laptop_Name"] = laptop_df["Laptop_Name"].apply(remove_company)
```

[40]:
```python
laptop_df.head(10)
```

[40]:

	Laptop_Brand	Laptop_Name	Processor_Company	Processor	\
0	HP	Victus	Intel	Core i5 12th Gen	
1	MSI	Thin 15	Intel	Core i5 12th Gen 12450H	
2	HP	Laptop	AMD	Ryzen 3 Quad Core 5300U	
3	Acer	One	Intel	Core i3 11th Gen 1115G4	
4	HP	HP	AMD	Ryzen 5 Hexa Core 5500U	
5	Infinix	GT Book	Intel	Core i5 12th Gen 12450H	
6	Acer	Aspire 7	Intel	Core i5 12th Gen 12450H	
7	ASUS	Vivobook 15	Intel	Core i3 12th Gen 1215U	
8	Acer	Aspire 3	Intel	Celeron Dual Core N4500	
9	MSI	Modern 14	AMD	Ryzen 5 Hexa Core 7530U	

	Operating_System	RAM	Storage	Storage_Type	Screen_Size	Rating	\

```
0       Windows 11   16      512          SSD           12     4.4
1       Windows 11   16      512          SSD           12     4.3
2       Windows 11    8      512          SSD           11     4.3
3       Windows 11    8      512          SSD           11     4.2
4       Windows 11   16      512          SSD           16     4.3
5       Windows 11   16      512          SSD           12     4.4
6       Windows 11   16      512          SSD           12     4.1
7       Windows 11    8      512          SSD           12     4.2
8       Windows 11    8      512          SSD           11     3.8
9       Windows 11   16      512          SSD           16     4.3

   Number_of_Reviews    Price
0               38.0  58990.0
1               34.0  57990.0
2              482.0  30999.0
3              571.0  26990.0
4              268.0  42990.0
5               13.0  64990.0
6              214.0  52990.0
7              360.0  35990.0
8               25.0  20990.0
9              246.0  36990.0
```

[41]:
```
# Hnadling the values of Storage column
laptop_df[["Storage"]].describe()
```

[41]:
```
           Storage
count  1624.000000
mean    440.557266
std     171.792497
min       1.000000
25%     512.000000
50%     512.000000
75%     512.000000
max     512.000000
```

- The min value of a laptop storage can be 128 GB nothing less than that, if it is less than it means the value is TB and needs to be converted into GB or its a wrong data point.
- Lets filter out all the datapoints where the storage is less than 128 GB.

[42]:
```
# Filtering out the datapoints where the storage is less than 128
filtered_df = laptop_df[laptop_df['Storage'] < 128]
filtered_df
```

[42]:
```
     Laptop_Brand                 Laptop_Name Processor_Company  \
17           ASUS           ROG Strix Scar 16             Intel
18             HP  Chromebook MediaTek MT8183        Chromebook
20           Acer                 Predator Neo             Intel
```

```
29            MSI                    Claw AI PC             Intel
41          Acer  Acer Predator Helios Neo 16             Intel
…            …                           …                …
1601       Lenovo        Yoga Slim 7x Qualcomm        Snapdragon
1602           HP   Chromebook MediaTek MT8183        Chromebook
1613       Lenovo                    Yoga AI PC             Intel
1625       Lenovo        Yoga Slim 7x Qualcomm        Snapdragon
1626           HP   Chromebook MediaTek MT8183        Chromebook

                    Processor Operating_System  RAM  Storage Storage_Type  \
17    Core i9 14th Gen 14900HX       Windows 11   32        2          SSD
18             MediaTek MT8183           Chrome    4       32         EMMC
20    Core i7 13th Gen 13700HX       Windows 11   16        1          SSD
29           Core Ultra 7 155H       Windows 11   16        1          SSD
41    Core i9 13th Gen 13900HX       Windows 11   16        1          SSD
…                          …             …  …      …          …
1601                  X Elite       Windows 11   32        1          SSD
1602           MediaTek MT8183           Chrome    4       32         EMMC
1613         Core Ultra 7 155H       Windows 11   32        1          SSD
1625                  X Elite       Windows 11   32        1          SSD
1626           MediaTek MT8183           Chrome    4       32         EMMC

      Screen_Size  Rating  Number_of_Reviews     Price
17             14     NaN                NaN  339990.0
18             81     3.8              501.0   11990.0
20             13     4.4               89.0  104990.0
29             16     5.0                1.0   74990.0
41             13     4.2                8.0  134990.0
…           …     …                  …        …
1601           32     NaN                NaN  149990.0
1602           81     3.8              501.0   11990.0
1613           32     NaN                NaN  244890.0
1625           32     NaN                NaN  149990.0
1626           81     3.8              501.0   11990.0

[223 rows x 12 columns]
```

2.3.3 Observation

- EMMC Storage are exception to the traditional laptops as they aare made for extremely light weight load so whatever storage is provided need not to be changed.
- However the storage value in for SSDs/HDDs needs to be updated.

[43]:
```
# Converting the Storage values of Storage in TB to GB
condition = (laptop_df['Storage_Type'].isin(['SSD', 'HDD'])) &␣
 ↪(laptop_df['Storage'] < 128)
laptop_df.loc[condition, 'Storage'] *= 1024
```

[44]:
```
# Looking at Data Sumamry
laptop_df.describe(include="all").T
```

[44]:

	count	unique	top	freq	mean	\
Laptop_Brand	1624	9	HP	388	NaN	
Laptop_Name	1624	47	HP	136	NaN	
Processor_Company	1624	4	Intel	1032	NaN	
Processor	1624	38	Core i3 12th Gen 1215U	182	NaN	
Operating_System	1624	2	Windows 11	1533	NaN	
RAM	1624.0	NaN	NaN	NaN	12.23399	
Storage	1624.0	NaN	NaN	NaN	526.857143	
Storage_Type	1624	2	SSD	1533	NaN	
Screen_Size	1624.0	NaN	NaN	NaN	19.899631	
Rating	1495.0	NaN	NaN	NaN	4.203144	
Number_of_Reviews	1495.0	NaN	NaN	NaN	247.48495	
Price	1624.0	NaN	NaN	NaN	53425.598522	

	std	min	25%	50%	75%	max
Laptop_Brand	NaN	NaN	NaN	NaN	NaN	NaN
Laptop_Name	NaN	NaN	NaN	NaN	NaN	NaN
Processor_Company	NaN	NaN	NaN	NaN	NaN	NaN
Processor	NaN	NaN	NaN	NaN	NaN	NaN
Operating_System	NaN	NaN	NaN	NaN	NaN	NaN
RAM	5.873543	4.0	8.0	8.0	16.0	32.0
Storage	202.351364	32.0	512.0	512.0	512.0	2048.0
Storage_Type	NaN	NaN	NaN	NaN	NaN	NaN
Screen_Size	18.594559	11.0	11.0	12.0	16.0	81.0
Rating	0.239443	3.3	4.1	4.2	4.3	5.0
Number_of_Reviews	202.948047	0.0	25.0	214.0	467.0	597.0
Price	49743.83399	11990.0	30999.0	37999.0	54990.0	339990.0

2.3.4 Observation

- Based on the above describe(Mean, precentiles and median) of the storage column, we can be sure that all the values are now accurate.

[45]:
```
# Dealing with Screen Size outlier values
laptop_df[laptop_df["Screen_Size"] > 17].head()
```

[45]:

	Laptop_Brand	Laptop_Name	Processor_Company	\
12	HP	15s	AMD	
18	HP	Chromebook MediaTek MT8183	Chromebook	
36	HP	15s	AMD	
42	HP	Chromebook MediaTek MT8183	Chromebook	
60	HP	15s	AMD	

	Processor	Operating_System	RAM	Storage	Storage_Type	\
12	Ryzen 3 Quad Core 5300U	Windows 11	8	512	SSD	

```
18          MediaTek MT8183           Chrome    4       32         EMMC
36  Ryzen 3 Quad Core 5300U       Windows 11    8      512          SSD
42          MediaTek MT8183           Chrome    4       32         EMMC
60  Ryzen 3 Quad Core 5300U       Windows 11    8      512          SSD

    Screen_Size  Rating  Number_of_Reviews    Price
12           64     4.2              405.0  32490.0
18           81     3.8              501.0  11990.0
36           64     4.2              405.0  32490.0
42           81     3.8              501.0  11990.0
60           64     4.2              405.0  32490.0
```

[46]:
```
laptop_df[laptop_df["Screen_Size"] > 17].shape
```

[46]:
```
(265, 12)
```

2.3.5 Observation

- The Screen Size values have been mislabelled during the aggregation or were not present in the give data.
- In order to impute these outlier values, these values will be replaced by the mean value of screen size based on each laptop company

[47]:
```
# Filtering out what all values are there in Screen Size
screen_size_counts = laptop_df['Screen_Size'].value_counts().sort_index()
```

[48]:
```
screen_size_counts
```

[48]:
```
Screen_Size
11    485
12    403
13    218
14      5
16    248
25     22
32     62
64    113
81     68
Name: count, dtype: int64
```

2.3.6 Observation

- The above values confirm that some values are mis-represented, so we will be replacing it with the mean value based on each laptop.
- This will be done for screen size greater than 32, and for values between 17 and 32 will be converted to inches.

[49]:
```
# Imputing outliers in Screen_Size Column

# Function to replace screen sizes greater than 32 with the median screen size␣
 ↪for each brand
# and convert screen sizes between 17 and 32 from cm to inches
def update_screen_size(group):
    median_size = group['Screen_Size'].median()
    group.loc[group['Screen_Size'] > 32, 'Screen_Size'] = median_size
    group.loc[(group['Screen_Size'] > 17) & (group['Screen_Size'] <= 32),␣
 ↪'Screen_Size'] *= 0.393701
    return group

# Apply the function to each group of Laptop_Brand
laptop_df = laptop_df.groupby('Laptop_Brand').apply(update_screen_size).
 ↪reset_index(drop=True)
```

[50]:
```
# Filtering out what all values are there in Screen Size
screen_size_counts = laptop_df['Screen_Size'].value_counts().sort_index()
screen_size_counts
```

[50]:
```
Screen_Size
9.842525      22
11.000000    507
12.000000    403
12.598432     62
13.000000    218
14.000000      5
16.000000    407
Name: count, dtype: int64
```

2.3.7 Observation

- Need to replace all the screen size values less than 11 to 11 inches
- Make the the screen size to definitive 12.5 inches

[51]:
```
# Replace specific Screen_Size values
laptop_df['Screen_Size'] = laptop_df['Screen_Size'].replace({9.842525: 11.00,␣
 ↪12.598432: 12.5})

# Filtering out what all values are there in Screen Size
screen_size_counts = laptop_df['Screen_Size'].value_counts().sort_index()
screen_size_counts
```

[51]:
```
Screen_Size
11.0    529
12.0    403
12.5     62
```

```
13.0    218
14.0      5
16.0    407
Name: count, dtype: int64
```

2.3.8 Observation

- All the screen sizes are now valid.

[52]:
```
laptop_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1624 entries, 0 to 1623
Data columns (total 12 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Laptop_Brand       1624 non-null   object
 1   Laptop_Name        1624 non-null   object
 2   Processor_Company  1624 non-null   object
 3   Processor          1624 non-null   object
 4   Operating_System   1624 non-null   object
 5   RAM                1624 non-null   int64
 6   Storage            1624 non-null   int64
 7   Storage_Type       1624 non-null   object
 8   Screen_Size        1624 non-null   float64
 9   Rating             1495 non-null   float64
 10  Number_of_Reviews  1495 non-null   float64
 11  Price              1624 non-null   float64
dtypes: float64(4), int64(2), object(6)
memory usage: 152.4+ KB
```

[53]:
```
laptop_df.describe(include="all").T
```

[53]:

	count	unique	top	freq	mean	\
Laptop_Brand	1624	9	HP	388	NaN	
Laptop_Name	1624	47	Aspire 7	136	NaN	
Processor_Company	1624	4	Intel	1032	NaN	
Processor	1624	38	Core i3 12th Gen 1215U	182	NaN	
Operating_System	1624	2	Windows 11	1533	NaN	
RAM	1624.0	NaN	NaN	NaN	12.23399	
Storage	1624.0	NaN	NaN	NaN	526.857143	
Storage_Type	1624	2	SSD	1533	NaN	
Screen_Size	1624.0	NaN	NaN	NaN	12.836207	
Rating	1495.0	NaN	NaN	NaN	4.203144	
Number_of_Reviews	1495.0	NaN	NaN	NaN	247.48495	
Price	1624.0	NaN	NaN	NaN	53425.598522	

	std	min	25%	50%	75%	max

Laptop_Brand	NaN	NaN	NaN	NaN	NaN	NaN
Laptop_Name	NaN	NaN	NaN	NaN	NaN	NaN
Processor_Company	NaN	NaN	NaN	NaN	NaN	NaN
Processor	NaN	NaN	NaN	NaN	NaN	NaN
Operating_System	NaN	NaN	NaN	NaN	NaN	NaN
RAM	5.873543	4.0	8.0	8.0	16.0	32.0
Storage	202.351364	32.0	512.0	512.0	512.0	2048.0
Storage_Type	NaN	NaN	NaN	NaN	NaN	NaN
Screen_Size	1.94802	11.0	11.0	12.0	16.0	16.0
Rating	0.239443	3.3	4.1	4.2	4.3	5.0
Number_of_Reviews	202.948047	0.0	25.0	214.0	467.0	597.0
Price	49743.83399	11990.0	30999.0	37999.0	54990.0	339990.0

3 Observation

- Replacing all the missing values in `Rating` and `Number_of_Reviews` with zero, as they will be treated as the laptop not being sold or not that appealing to customers.

[54]:
```
# Replacing Nan values in the Rating and Number of Reviews column
laptop_df['Rating'] = laptop_df['Rating'].fillna(0)
laptop_df['Number_of_Reviews'] = laptop_df['Number_of_Reviews'].fillna(0)
```

[55]:
```
# Taking a final look at info and description of data
laptop_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1624 entries, 0 to 1623
Data columns (total 12 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Laptop_Brand       1624 non-null   object
 1   Laptop_Name        1624 non-null   object
 2   Processor_Company  1624 non-null   object
 3   Processor          1624 non-null   object
 4   Operating_System   1624 non-null   object
 5   RAM                1624 non-null   int64
 6   Storage            1624 non-null   int64
 7   Storage_Type       1624 non-null   object
 8   Screen_Size        1624 non-null   float64
 9   Rating             1624 non-null   float64
 10  Number_of_Reviews  1624 non-null   float64
 11  Price              1624 non-null   float64
dtypes: float64(4), int64(2), object(6)
memory usage: 152.4+ KB
```

[56]:
```
laptop_df.describe(include="all").T
```

[56]:

	count	unique	top	freq	mean	\
Laptop_Brand	1624	9	HP	388	NaN	
Laptop_Name	1624	47	Aspire 7	136	NaN	
Processor_Company	1624	4	Intel	1032	NaN	
Processor	1624	38	Core i3 12th Gen 1215U	182	NaN	
Operating_System	1624	2	Windows 11	1533	NaN	
RAM	1624.0	NaN	NaN	NaN	12.23399	
Storage	1624.0	NaN	NaN	NaN	526.857143	
Storage_Type	1624	2	SSD	1533	NaN	
Screen_Size	1624.0	NaN	NaN	NaN	12.836207	
Rating	1624.0	NaN	NaN	NaN	3.869273	
Number_of_Reviews	1624.0	NaN	NaN	NaN	227.826355	
Price	1624.0	NaN	NaN	NaN	53425.598522	

	std	min	25%	50%	75%	max
Laptop_Brand	NaN	NaN	NaN	NaN	NaN	NaN
Laptop_Name	NaN	NaN	NaN	NaN	NaN	NaN
Processor_Company	NaN	NaN	NaN	NaN	NaN	NaN
Processor	NaN	NaN	NaN	NaN	NaN	NaN
Operating_System	NaN	NaN	NaN	NaN	NaN	NaN
RAM	5.873543	4.0	8.0	8.0	16.0	32.0
Storage	202.351364	32.0	512.0	512.0	512.0	2048.0
Storage_Type	NaN	NaN	NaN	NaN	NaN	NaN
Screen_Size	1.94802	11.0	11.0	12.0	16.0	16.0
Rating	1.159917	0.0	4.075	4.2	4.3	5.0
Number_of_Reviews	205.902161	0.0	12.0	214.0	424.0	597.0
Price	49743.83399	11990.0	30999.0	37999.0	54990.0	339990.0

[57]:
```
# Since the data is now cleaned, it can be exported as a clean CSV for further␣
 ↪analysis
laptop_df.to_csv(r"data\flipkart_laptop_cleaned.csv",index=False)
```