About Superstore Dataset:

The "Superstore Sales" dataset is a comprehensive and versatile collection of data that provides valuable insights into sales, customer behavior, and product performance. This dataset offers a rich resource for in-depth analysis.

Containing information from diverse regions and segments, the dataset enables exploration of trends, patterns, and correlations in sales and customer preferences. The dataset encompasses sales transactions, enabling researchers and analysts to understand buying patterns, identify high-demand products, and assess the effectiveness of different shipping modes.

Moreover, the dataset provides an opportunity to examine the impact of various factors such as discounts, geographical locations, and product categories on profitability. By analyzing this dataset, businesses and data enthusiasts can uncover actionable insights for optimizing pricing strategies, supply chain management, and customer engagement.

Whether used for educational purposes, business strategy formulation, or data analysis practice, the "Superstore Sales" dataset offers a comprehensive platform to delve into the dynamics of sales operations, customer interactions, and the factors that drive business success.

To get you started explanation of what the column names mean are provided below:

.Row ID= This is a unique id that is assigned to each Row.

.Order ID= This is an id that is assigned the unique number that gets assigned to an order placed by the customer.

.Order Date= This is an date of order at which the date when each order was shipped or sent out for delivery.

.Ship Date =The date that the order is shipped from the seller or warehouse to the customer.

.Customer ID= This is a unique id that is assigned to each customer.

.Customer Name= This is a name of customer which made by the order.

.Segment= Describe a categorization of customers based on certain criteria, such as "Corporate," "Consumer," or "Home Office."

.Country= This column specifies the country where each order was placed or shipped to.

.City= This column contains the name of the city where the order was delivered.

.State= This column likely stores the state or province information related to the delivery location.

.Postal Code= This column typically contains the postal or ZIP code associated with the delivery address.

.Region= This column may provide a broader geographical categorization, such as "West," "East," "North," or "South."

.Product ID= This column stores a unique identifier for each product in the dataset, allowing you to link products to specific orders.

.Category= This column categorizes products into broader groups, such as "Office Supplies," "Furniture," or "Technology."

.Sub-Category= This column provides a more detailed categorization of products within each category. For example, under "Binders","Paper","Furnishings","Phones","Storage","Art","Accessories","Chairs","Appliances"," Labels","Tables","Envelopes".

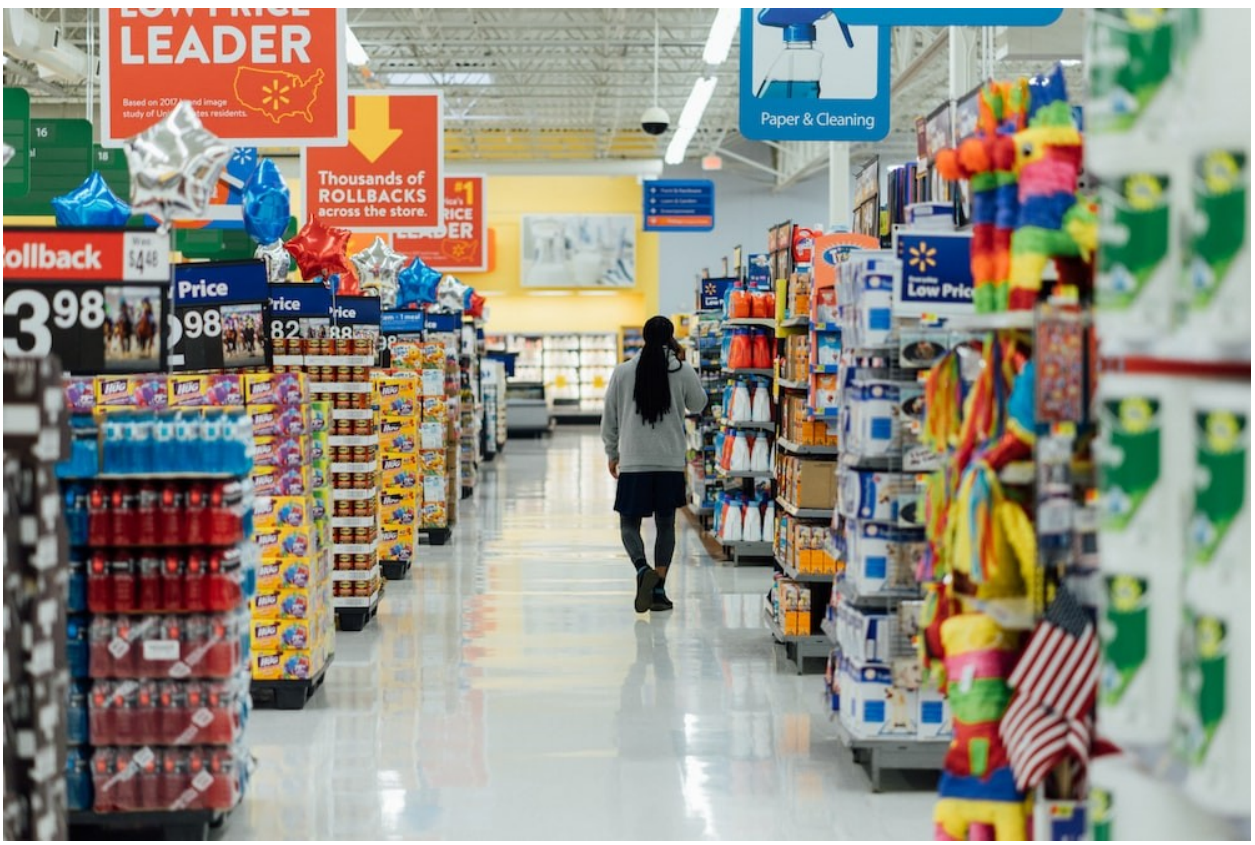
.Product Name= This column contains the name or description of each individual product.

.Sales= This column represents the total sales revenue generated by each order or product.It typically includes the price and quantity sold.

.Quantity: This column indicates the number of units or items sold for each product in an order.

.Discount: This column may contain information about any discounts applied to the products in an order.

.Profit: This column typically shows the profit earned from each order or product, taking into account factors such as cost, price, and discounts.



Step1:- Importing Necessary Libraries

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from warnings import filterwarnings
filterwarnings('ignore')
```

Step 2:- Loading the Dataset

```
store=pd.read_excel('Superstore.xlsx')  #Read the Excel_file
```

Top 5 Records

```
store.head()

   Row ID         Order ID Order Date  Ship Date       Ship Mode
Customer ID  \
0       1  CA-2013-152156 2013-11-09 2013-11-12    Second Class    CG-
12520
1       2  CA-2013-152156 2013-11-09 2013-11-12    Second Class    CG-
12520
2       3  CA-2013-138688 2013-06-13 2013-06-17    Second Class    DV-
13045
3       4  US-2012-108966 2012-10-11 2012-10-18  Standard Class    SO-
20335
4       5  US-2012-108966 2012-10-11 2012-10-18  Standard Class    SO-
20335

     Customer Name    Segment        Country             City  ...  \
0      Claire Gute   Consumer  United States        Henderson  ...
1      Claire Gute   Consumer  United States        Henderson  ...
2  Darrin Van Huff  Corporate  United States      Los Angeles  ...
3   Sean O'Donnell   Consumer  United States  Fort Lauderdale  ...
4   Sean O'Donnell   Consumer  United States  Fort Lauderdale  ...

  Postal Code  Region       Product ID        Category Sub-
Category  \
0       42420   South  FUR-BO-10001798        Furniture    Bookcases

1       42420   South  FUR-CH-10000454        Furniture       Chairs

2       90036    West  OFF-LA-10000240  Office Supplies       Labels

3       33311   South  FUR-TA-10000577        Furniture       Tables

4       33311   South  OFF-ST-10000760  Office Supplies      Storage


                                        Product Name     Sales
```

```
Quantity  \
0                    Bush Somerset Collection Bookcase  261.9600
2
1  Hon Deluxe Fabric Upholstered Stacking Chairs,...  731.9400
3
2  Self-Adhesive Address Labels for Typewriters b...   14.6200
2
3      Bretford CR4500 Series Slim Rectangular Table  957.5775
5
4                       Eldon Fold 'N Roll Cart System   22.3680
2

   Discount    Profit
0      0.00   41.9136
1      0.00  219.5820
2      0.00    6.8714
3      0.45 -383.0310
4      0.20    2.5164

[5 rows x 21 columns]
```

Bottom 5 Records

```
store.tail()

      Row ID        Order ID Order Date  Ship Date       Ship Mode  \
9989    9990  CA-2011-110422 2011-01-22 2011-01-24    Second Class
9990    9991  CA-2014-121258 2014-02-27 2014-03-04  Standard Class
9991    9992  CA-2014-121258 2014-02-27 2014-03-04  Standard Class
9992    9993  CA-2014-121258 2014-02-27 2014-03-04  Standard Class
9993    9994  CA-2014-119914 2014-05-05 2014-05-10    Second Class

     Customer ID     Customer Name   Segment        Country
City  ...  \
9989    TB-21400  Tom Boeckenhauer  Consumer  United States
Miami  ...
9990    DB-13060       Dave Brooks  Consumer  United States   Costa
Mesa  ...
9991    DB-13060       Dave Brooks  Consumer  United States   Costa
Mesa  ...
9992    DB-13060       Dave Brooks  Consumer  United States   Costa
Mesa  ...
9993    CC-12220      Chris Cortes  Consumer  United States
Westminster  ...

     Postal Code  Region       Product ID         Category Sub-
Category  \
9989       33180   South  FUR-FU-10001889        Furniture
Furnishings
9990       92627    West  FUR-FU-10000747        Furniture
```

```
Furnishings
9991       92627    West  TEC-PH-10003645       Technology
Phones
9992       92627    West  OFF-PA-10004041  Office Supplies
Paper
9993       92683    West  OFF-AP-10002684  Office Supplies
Appliances

                                                 Product Name    Sales
Quantity  \
9989                              Ultra Door Pull Handle   25.248
3
9990  Tenex B1-RE Series Chair Mats for Low Pile Car...   91.960
2
9991                               Aastra 57i VoIP phone  258.576
2
9992  It's Hot Message Books with Stickers, 2 3/4" x 5"   29.600
4
9993  Acco 7-Outlet Masterpiece Power Center, Wihtou...  243.160
2

      Discount   Profit
9989       0.2   4.1028
9990       0.0  15.6332
9991       0.2  19.3932
9992       0.0  13.3200
9993       0.0  72.9480

[5 rows x 21 columns]
```

Checking shape of an rows and columns

```
store.shape
```

```
(9994, 21)
```

Fetching the columns

```
store.columns
```

```
Index(['Row ID', 'Order ID', 'Order Date', 'Ship Date', 'Ship Mode',
       'Customer ID', 'Customer Name', 'Segment', 'Country', 'City',
'State',
       'Postal Code', 'Region', 'Product ID', 'Category', 'Sub-
Category',
       'Product Name', 'Sales', 'Quantity', 'Discount', 'Profit'],
      dtype='object')
```

Checking the datatypes of each features

```
store.dtypes
```

```
Row ID                     int64
Order ID                  object
Order Date        datetime64[ns]
Ship Date         datetime64[ns]
Ship Mode                 object
Customer ID               object
Customer Name             object
Segment                   object
Country                   object
City                      object
State                     object
Postal Code                int64
Region                    object
Product ID                object
Category                  object
Sub-Category              object
Product Name              object
Sales                    float64
Quantity                   int64
Discount                 float64
Profit                   float64
dtype: object
```

Featching the entire Unique Values

```
store['Ship Mode'].value_counts()
```

```
Standard Class    5968
Second Class      1945
First Class       1538
Same Day           543
Name: Ship Mode, dtype: int64
```

```
store.Segment.value_counts()
```

```
Consumer       5191
Corporate      3020
Home Office    1783
Name: Segment, dtype: int64
```

```
store.Country.value_counts()
```

```
United States    9994
Name: Country, dtype: int64
```

```
store['Customer Name'].value_counts()
```

```
William Brown          37
John Lee               34
Matt Abelman           34
Paul Prost             34
Chloris Kastensmidt    32
                       ..
Lela Donovan            1
Anthony O'Donnell       1
Carl Jackson            1
Ricardo Emerson         1
Jocasta Rupert          1
Name: Customer Name, Length: 793, dtype: int64
```

```
store.City.value_counts()
```

```
New York City      915
Los Angeles        747
Philadelphia       537
San Francisco      510
Seattle            428
                  ...
Glenview             1
Missouri City        1
Rochester Hills      1
Palatine             1
Manhattan            1
Name: City, Length: 531, dtype: int64
```

```
store.State.value_counts()
```

```
California              2001
New York                1128
Texas                    985
Pennsylvania             587
Washington               506
Illinois                 492
Ohio                     469
Florida                  383
Michigan                 255
North Carolina           249
Arizona                  224
Virginia                 224
Georgia                  184
Tennessee                183
Colorado                 182
Indiana                  149
Kentucky                 139
Massachusetts            135
New Jersey               130
Oregon                   124
```

```
Wisconsin                 110
Maryland                  105
Delaware                   96
Minnesota                  89
Connecticut                82
Oklahoma                   66
Missouri                   66
Alabama                    61
Arkansas                   60
Rhode Island               56
Utah                       53
Mississippi                53
Louisiana                  42
South Carolina             42
Nevada                     39
Nebraska                   38
New Mexico                 37
Iowa                       30
New Hampshire              27
Kansas                     24
Idaho                      21
Montana                    15
South Dakota               12
Vermont                    11
District of Columbia       10
Maine                       8
North Dakota                7
West Virginia               4
Wyoming                     1
Name: State, dtype: int64
```

```
store.Region.value_counts()
```

```
West       3203
East       2848
Central    2323
South      1620
Name: Region, dtype: int64
```

```
store.Category.value_counts()
```

```
Office Supplies    6026
Furniture          2121
Technology         1847
Name: Category, dtype: int64
```

```
store['Sub-Category'].value_counts()
```

```
Binders        1523
Paper          1370
Furnishings     957
```

```
Phones            889
Storage           846
Art               796
Accessories       775
Chairs            617
Appliances        466
Labels            364
Tables            319
Envelopes         254
Bookcases         228
Fasteners         217
Supplies          190
Machines          115
Copiers            68
Name: Sub-Category, dtype: int64
```

```
store.Quantity.value_counts()
```

```
3     2409
2     2402
5     1230
4     1191
1      899
7      606
6      572
9      258
8      257
10      57
11      34
14      29
13      27
12      23
Name: Quantity, dtype: int64
```

```
store.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9994 entries, 0 to 9993
Data columns (total 21 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Row ID         9994 non-null   int64
 1   Order ID       9994 non-null   object
 2   Order Date     9994 non-null   datetime64[ns]
 3   Ship Date      9994 non-null   datetime64[ns]
 4   Ship Mode      9994 non-null   object
 5   Customer ID    9994 non-null   object
 6   Customer Name  9994 non-null   object
 7   Segment        9994 non-null   object
 8   Country        9994 non-null   object
```

```
 9   City            9994 non-null   object
 10  State           9994 non-null   object
 11  Postal Code     9994 non-null   int64
 12  Region          9994 non-null   object
 13  Product ID      9994 non-null   object
 14  Category        9994 non-null   object
 15  Sub-Category    9994 non-null   object
 16  Product Name    9994 non-null   object
 17  Sales           9994 non-null   float64
 18  Quantity        9994 non-null   int64
 19  Discount        9994 non-null   float64
 20  Profit          9994 non-null   float64
dtypes: datetime64[ns](2), float64(3), int64(3), object(13)
memory usage: 1.6+ MB
```

```
store['Product Name'].value_counts()
```

```
Staples
227
Avery Non-Stick Binders
20
KI Adjustable-Height Table
18
Storex Dura Pro Binders
17
Logitech 910-002974 M325 Wireless Mouse for Web Scrolling
15

...
Boston 1900 Electric Pencil Sharpener
1
RCA ViSYS 25423RE1 Corded phone
1
Canon Color ImageCLASS MF8580Cdw Wireless Laser All-In-One Printer,
Copier, Scanner      1
Newell 342
1
Eldon Jumbo ProFile Portable File Boxes Graphite/Black
1
Name: Product Name, Length: 1841, dtype: int64
```

Featches the Unique values in an entire columns one-by-one

```
for i in store:
    print('\n',i,'\n')                # Return the unique of all
individual columns
    print(store[i].unique)
    print(store[i].nunique())
```

```
 Row ID

<bound method Series.unique of 0          1
1          2
2          3
3          4
4          5
        ...
9989    9990
9990    9991
9991    9992
9992    9993
9993    9994
Name: Row ID, Length: 9994, dtype: int64>
9994

 Order ID

<bound method Series.unique of 0       CA-2013-152156
1       CA-2013-152156
2       CA-2013-138688
3       US-2012-108966
4       US-2012-108966
             ...
9989    CA-2011-110422
9990    CA-2014-121258
9991    CA-2014-121258
9992    CA-2014-121258
9993    CA-2014-119914
Name: Order ID, Length: 9994, dtype: object>
5009

 Order Date

<bound method Series.unique of 0      2013-11-09
1      2013-11-09
2      2013-06-13
3      2012-10-11
4      2012-10-11
          ...
9989   2011-01-22
9990   2014-02-27
9991   2014-02-27
9992   2014-02-27
9993   2014-05-05
Name: Order Date, Length: 9994, dtype: datetime64[ns]>
1238

 Ship Date
```

```
<bound method Series.unique of 0      2013-11-12
1      2013-11-12
2      2013-06-17
3      2012-10-18
4      2012-10-18
          ...
9989   2011-01-24
9990   2014-03-04
9991   2014-03-04
9992   2014-03-04
9993   2014-05-10
Name: Ship Date, Length: 9994, dtype: datetime64[ns]>
1334

 Ship Mode

<bound method Series.unique of 0         Second Class
1         Second Class
2         Second Class
3       Standard Class
4       Standard Class
             ...
9989      Second Class
9990    Standard Class
9991    Standard Class
9992    Standard Class
9993      Second Class
Name: Ship Mode, Length: 9994, dtype: object>
4

 Customer ID

<bound method Series.unique of 0       CG-12520
1       CG-12520
2       DV-13045
3       SO-20335
4       SO-20335
          ...
9989    TB-21400
9990    DB-13060
9991    DB-13060
9992    DB-13060
9993    CC-12220
Name: Customer ID, Length: 9994, dtype: object>
793

 Customer Name

<bound method Series.unique of 0           Claire Gute
```

```
1             Claire Gute
2         Darrin Van Huff
3          Sean O'Donnell
4          Sean O'Donnell
              ...
9989    Tom Boeckenhauer
9990         Dave Brooks
9991         Dave Brooks
9992         Dave Brooks
9993        Chris Cortes
Name: Customer Name, Length: 9994, dtype: object>
793

 Segment

<bound method Series.unique of 0       Consumer
1        Consumer
2       Corporate
3        Consumer
4        Consumer
          ...
9989     Consumer
9990     Consumer
9991     Consumer
9992     Consumer
9993     Consumer
Name: Segment, Length: 9994, dtype: object>
3

 Country

<bound method Series.unique of 0      United States
1       United States
2       United States
3       United States
4       United States
            ...
9989    United States
9990    United States
9991    United States
9992    United States
9993    United States
Name: Country, Length: 9994, dtype: object>
1

 City

<bound method Series.unique of 0             Henderson
1              Henderson
2            Los Angeles
```

```
3       Fort Lauderdale
4       Fort Lauderdale
             ...
9989              Miami
9990         Costa Mesa
9991         Costa Mesa
9992         Costa Mesa
9993        Westminster
Name: City, Length: 9994, dtype: object>
531

 State

<bound method Series.unique of 0         Kentucky
1         Kentucky
2       California
3          Florida
4          Florida
           ...
9989       Florida
9990    California
9991    California
9992    California
9993    California
Name: State, Length: 9994, dtype: object>
49

 Postal Code

<bound method Series.unique of 0       42420
1       42420
2       90036
3       33311
4       33311
        ...
9989    33180
9990    92627
9991    92627
9992    92627
9993    92683
Name: Postal Code, Length: 9994, dtype: int64>
631

 Region

<bound method Series.unique of 0       South
1       South
2        West
3       South
4       South
```

```
        ...
9989    South
9990     West
9991     West
9992     West
9993     West
Name: Region, Length: 9994, dtype: object>
4

 Product ID

<bound method Series.unique of 0       FUR-BO-10001798
1       FUR-CH-10000454
2       OFF-LA-10000240
3       FUR-TA-10000577
4       OFF-ST-10000760
             ...
9989    FUR-FU-10001889
9990    FUR-FU-10000747
9991    TEC-PH-10003645
9992    OFF-PA-10004041
9993    OFF-AP-10002684
Name: Product ID, Length: 9994, dtype: object>
1862

 Category

<bound method Series.unique of 0             Furniture
1             Furniture
2       Office Supplies
3             Furniture
4       Office Supplies
             ...
9989          Furniture
9990          Furniture
9991         Technology
9992    Office Supplies
9993    Office Supplies
Name: Category, Length: 9994, dtype: object>
3

 Sub-Category

<bound method Series.unique of 0         Bookcases
1            Chairs
2            Labels
3            Tables
4           Storage
           ...
9989    Furnishings
```

```
9990    Furnishings
9991         Phones
9992          Paper
9993     Appliances
Name: Sub-Category, Length: 9994, dtype: object>
17

 Product Name

<bound method Series.unique of 0                       Bush Somerset Collection Bookcase
1       Hon Deluxe Fabric Upholstered Stacking Chairs,...
2       Self-Adhesive Address Labels for Typewriters b...
3           Bretford CR4500 Series Slim Rectangular Table
4                          Eldon Fold 'N Roll Cart System
                              ...
9989                               Ultra Door Pull Handle
9990    Tenex B1-RE Series Chair Mats for Low Pile Car...
9991                                Aastra 57i VoIP phone
9992    It's Hot Message Books with Stickers, 2 3/4" x 5"
9993    Acco 7-Outlet Masterpiece Power Center, Wihtou...
Name: Product Name, Length: 9994, dtype: object>
1841

 Sales

<bound method Series.unique of 0       261.9600
1       731.9400
2        14.6200
3       957.5775
4        22.3680
          ...
9989     25.2480
9990     91.9600
9991    258.5760
9992     29.6000
9993    243.1600
Name: Sales, Length: 9994, dtype: float64>
6144

 Quantity

<bound method Series.unique of 0       2
1       3
2       2
3       5
4       2
       ..
9989    3
9990    2
```

```
9991    2
9992    4
9993    2
Name: Quantity, Length: 9994, dtype: int64>
14

 Discount

<bound method Series.unique of 0       0.00
1       0.00
2       0.00
3       0.45
4       0.20
        ...
9989    0.20
9990    0.00
9991    0.20
9992    0.00
9993    0.00
Name: Discount, Length: 9994, dtype: float64>
12

 Profit

<bound method Series.unique of 0        41.9136
1       219.5820
2         6.8714
3      -383.0310
4         2.5164
          ...
9989      4.1028
9990     15.6332
9991     19.3932
9992     13.3200
9993     72.9480
Name: Profit, Length: 9994, dtype: float64>
7545
```

```
store.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9994 entries, 0 to 9993
Data columns (total 21 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Row ID         9994 non-null   int64
 1   Order ID       9994 non-null   object
 2   Order Date     9994 non-null   datetime64[ns]
 3   Ship Date      9994 non-null   datetime64[ns]
 4   Ship Mode      9994 non-null   object
```

```
 5   Customer ID    9994 non-null   object
 6   Customer Name  9994 non-null   object
 7   Segment        9994 non-null   object
 8   Country        9994 non-null   object
 9   City           9994 non-null   object
 10  State          9994 non-null   object
 11  Postal Code    9994 non-null   int64
 12  Region         9994 non-null   object
 13  Product ID     9994 non-null   object
 14  Category       9994 non-null   object
 15  Sub-Category   9994 non-null   object
 16  Product Name   9994 non-null   object
 17  Sales          9994 non-null   float64
 18  Quantity       9994 non-null   int64
 19  Discount       9994 non-null   float64
 20  Profit         9994 non-null   float64
dtypes: datetime64[ns](2), float64(3), int64(3), object(13)
memory usage: 1.6+ MB
```

3. Feature Engineering

```
store["Order_Date"] = pd.to_datetime(store['Order Date'],
format="%d/%m/%Y").dt.day
store["Order_Date"]

0        9
1        9
2       13
3       11
4       11
        ..
9989    22
9990    27
9991    27
9992    27
9993     5
Name: Order_Date, Length: 9994, dtype: int64

store["Order_Month"] = pd.to_datetime(store["Order Date"],format =
"%d/%m/%Y").dt.month
store["Order_Month"]

0       11
1       11
2        6
3       10
4       10
        ..
9989     1
9990     2
```

```
9991     2
9992     2
9993     5
Name: Order_Month, Length: 9994, dtype: int64
```

```
store["Order_Year"] = pd.to_datetime(store["Order Date"],format =
"%d/%m/%Y").dt.year
store["Order_Year"]
```

```
0       2013
1       2013
2       2013
3       2012
4       2012
        ... 
9989    2011
9990    2014
9991    2014
9992    2014
9993    2014
Name: Order_Year, Length: 9994, dtype: int64
```

```
store["Ship_Date"] = pd.to_datetime(store['Ship Date'],
format="%d/%m/%Y").dt.day
store["Ship_Date"]
```

```
0       12
1       12
2       17
3       18
4       18
        ..
9989    24
9990     4
9991     4
9992     4
9993    10
Name: Ship_Date, Length: 9994, dtype: int64
```

```
store["Ship_Month"] = pd.to_datetime(store['Ship Date'],
format="%d/%m/%Y").dt.month
store["Ship_Month"]
```

```
0       11
1       11
2        6
3       10
4       10
        ..
9989     1
9990     3
```

```
9991     3
9992     3
9993     5
Name: Ship_Month, Length: 9994, dtype: int64
```

```
store["Ship_Year"] = pd.to_datetime(store['Ship Date'],
format="%d/%m/%Y").dt.year
store["Ship_Year"]
```

```
0       2013
1       2013
2       2013
3       2012
4       2012
        ...
9989    2011
9990    2014
9991    2014
9992    2014
9993    2014
Name: Ship_Year, Length: 9994, dtype: int64
```

```
store.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9994 entries, 0 to 9993
Data columns (total 27 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Row ID         9994 non-null   int64
 1   Order ID       9994 non-null   object
 2   Order Date     9994 non-null   datetime64[ns]
 3   Ship Date      9994 non-null   datetime64[ns]
 4   Ship Mode      9994 non-null   object
 5   Customer ID    9994 non-null   object
 6   Customer Name  9994 non-null   object
 7   Segment        9994 non-null   object
 8   Country        9994 non-null   object
 9   City           9994 non-null   object
 10  State          9994 non-null   object
 11  Postal Code    9994 non-null   int64
 12  Region         9994 non-null   object
 13  Product ID     9994 non-null   object
 14  Category       9994 non-null   object
 15  Sub-Category   9994 non-null   object
 16  Product Name   9994 non-null   object
 17  Sales          9994 non-null   float64
 18  Quantity       9994 non-null   int64
 19  Discount       9994 non-null   float64
 20  Profit         9994 non-null   float64
```

```
 21  Order_Date      9994 non-null   int64
 22  Order_Month     9994 non-null   int64
 23  Order_Year      9994 non-null   int64
 24  Ship_Date       9994 non-null   int64
 25  Ship_Month      9994 non-null   int64
 26  Ship_Year       9994 non-null   int64
dtypes: datetime64[ns](2), float64(3), int64(9), object(13)
memory usage: 2.1+ MB
```

Droping the Ship Date & Order Date

As we have extracted Date, Month and Year from Order Date and Ship Date columns, So no need to keep both the column we can drop it

```
store.drop(['Ship Date','Order Date'],inplace=True,axis=1)
```

Checking the Null/Missing Values

```
store.isna().sum() # their is No Null / Missing Values in an dataset
```

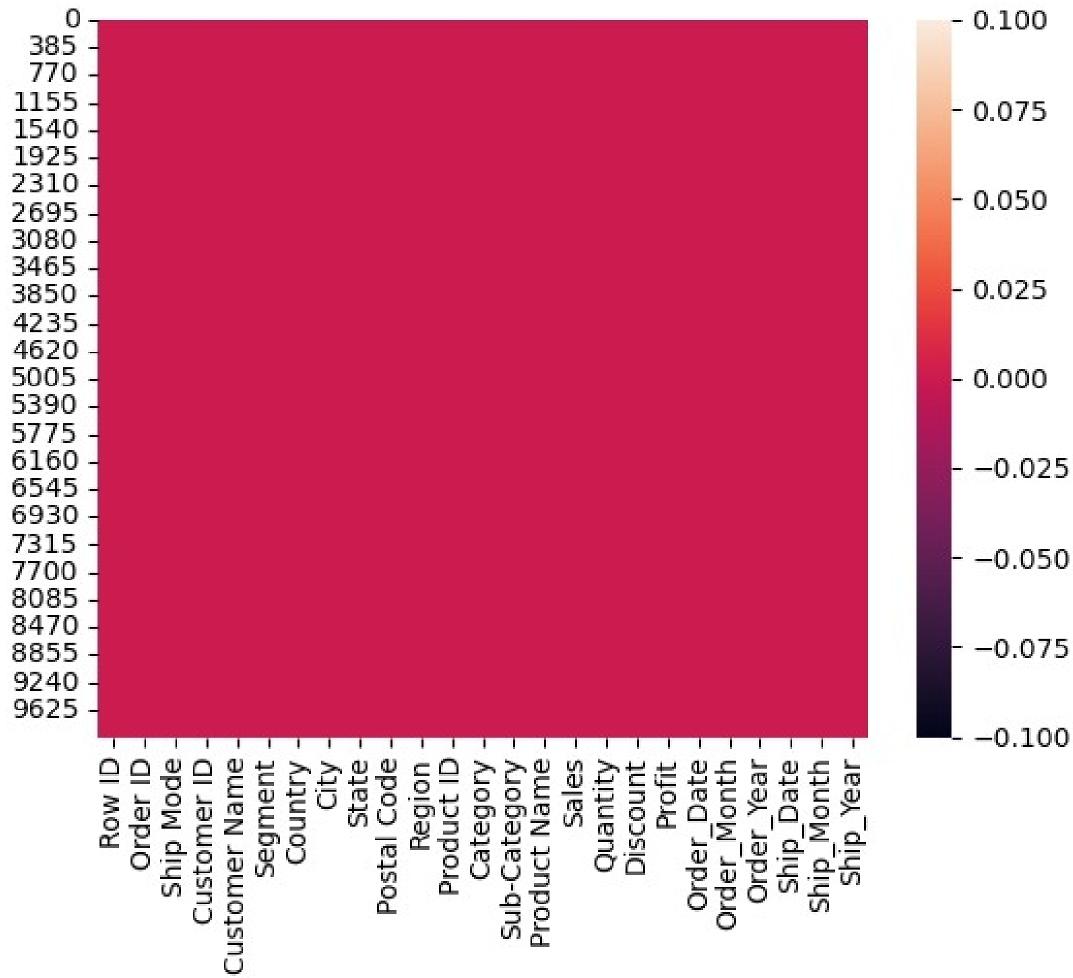
```
Row ID           0
Order ID         0
Ship Mode        0
Customer ID      0
Customer Name    0
Segment          0
Country          0
City             0
State            0
Postal Code      0
Region           0
Product ID       0
Category         0
Sub-Category     0
Product Name     0
Sales            0
Quantity         0
Discount         0
Profit           0
Order_Date       0
Order_Month      0
Order_Year       0
Ship_Date        0
Ship_Month       0
Ship_Year        0
dtype: int64
```

Their is No Null / Missing Values in an Superstore_dataset

Feteching the null/missing values with visualisation

```
sns.heatmap(store.isnull())  #here we have no null values are present
in our dataset
```

```
<Axes: >
```



Checking for duplicated values

```
store[store.duplicated]
```

```
Empty DataFrame
Columns: [Row ID, Order ID, Ship Mode, Customer ID, Customer Name,
Segment, Country, City, State, Postal Code, Region, Product ID,
Category, Sub-Category, Product Name, Sales, Quantity, Discount,
Profit, Order_Date, Order_Month, Order_Year, Ship_Date, Ship_Month,
Ship_Year]
Index: []
```

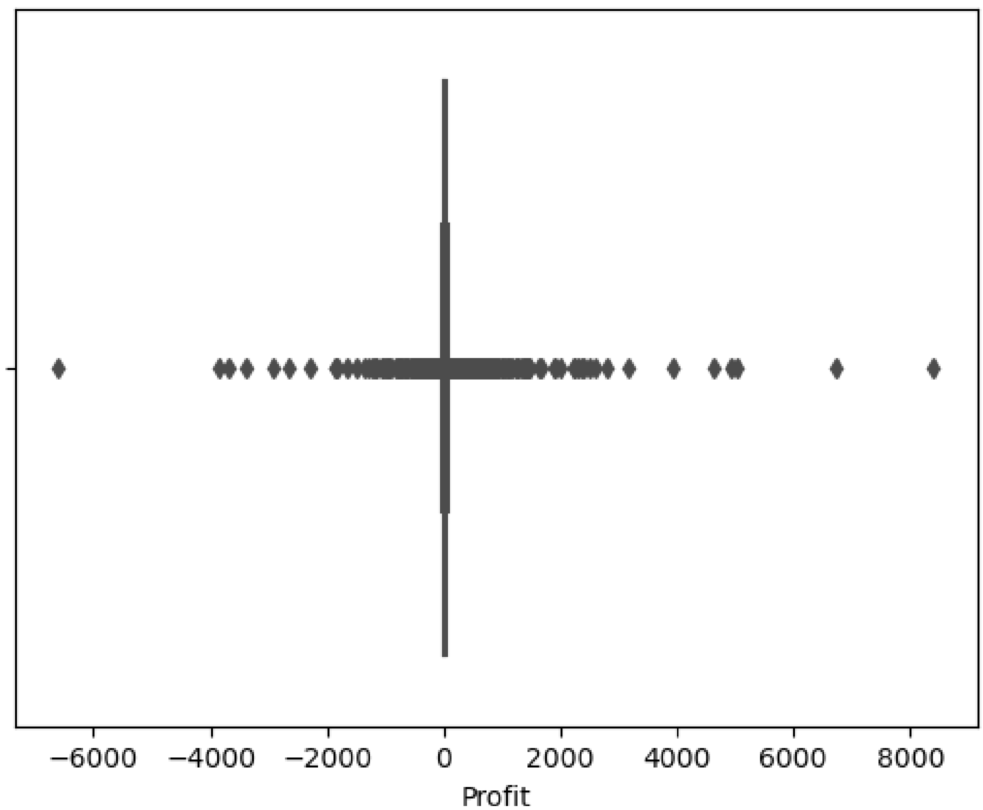
```
[0 rows x 25 columns]
```

We found there are no duplicate records in a datset

Featching Outliers

```
sns.boxplot(data=store,x='Profit',color='red')
```

```
<Axes: xlabel='Profit'>
```



There are outliers afre present in an Profit Attribute

Treating/Cleaning the Outiliers

```
Q1=store.Profit.quantile(0.25)
Q3=store.Profit.quantile(0.75)
print("First Quratile:",Q1)
print("Third Quartile:",Q3)
```

```
First Quratile: 1.72875
Third Quartile: 29.363999999999997
```

```
IQR=Q3-Q1
print("Inter Quartile Range:",IQR)
```

```
Inter Quartile Range: 27.635249999999996
```

```
HE=Q3+1.5*IQR
print("HIGHER END POINT:",HE)

LE=Q1-1.5*IQR
print("LOWER END POINT:",LE)
```

```
HIGHER END POINT: 70.81687499999998
LOWER END POINT: -39.724124999999994
```

```
store[(store.Profit>HE)|(store.Profit<LE)]
```

```
      Row ID        Order ID       Ship Mode Customer ID     Customer
Name  \
1          2  CA-2013-152156    Second Class    CG-12520        Claire
Gute
3          4  US-2012-108966  Standard Class    SO-20335    Sean
O'Donnell
7          8  CA-2011-115812  Standard Class    BH-11710   Brosina
Hoffman
10        11  CA-2011-115812  Standard Class    BH-11710   Brosina
Hoffman
13        14  CA-2013-161389  Standard Class    IM-15070      Irene
Maddox
...      ...             ...             ...         ...
...
9957    9958  US-2011-143287  Standard Class    KN-16705     Kristina
Nunn
9962    9963  CA-2012-168088     First Class    CM-12655  Corinna
Mitchell
9968    9969  CA-2014-153871  Standard Class    RB-19435   Richard
Bierner
9979    9980  US-2013-103674  Standard Class    AP-10720         Anne
Pryor
9993    9994  CA-2014-119914    Second Class    CC-12220      Chris
Cortes

          Segment        Country             City       State  Postal
Code  \
1        Consumer  United States        Henderson    Kentucky
42420
3        Consumer  United States  Fort Lauderdale     Florida
33311
7        Consumer  United States      Los Angeles  California
90032
10       Consumer  United States      Los Angeles  California
90032
13       Consumer  United States          Seattle  Washington
98103
```

```
...           ...            ...              ...         ...
...
9957  Home Office  United States     New Rochelle    New York
10801
9962  Home Office  United States          Houston       Texas
77041
9968     Consumer  United States       Plainfield  New Jersey
7060
9979  Home Office  United States      Los Angeles  California
90032
9993     Consumer  United States      Westminster  California
92683

      ...      Sales Quantity Discount    Profit Order_Date
Order_Month  \
1     ...   731.9400        3     0.00  219.5820          9
11
3     ...   957.5775        5     0.45 -383.0310         11
10
7     ...   907.1520        6     0.20   90.7152          9
6
10    ...  1706.1840        9     0.20   85.3092          9
6
13    ...   407.9760        3     0.20  132.5922          6
12
...   ...        ...      ...      ...       ...        ...          .
..
9957  ...   223.9200        4     0.00  109.7208         11
11
9962  ...   383.4656        4     0.32  -67.6704         19
3
9968  ...   735.9800        2     0.00  331.1910         12
12
9979  ...   437.4720       14     0.20  153.1152          7
12
9993  ...   243.1600        2     0.00   72.9480          5
5

      Order_Year  Ship_Date  Ship_Month  Ship_Year
1           2013         12          11       2013
3           2012         18          10       2012
7           2011         14           6       2011
10          2011         14           6       2011
13          2013         11          12       2013
...          ...        ...         ...        ...
9957        2011         17          11       2011
9962        2012         22           3       2012
9968        2014         18          12       2014
9979        2013         11          12       2013
```

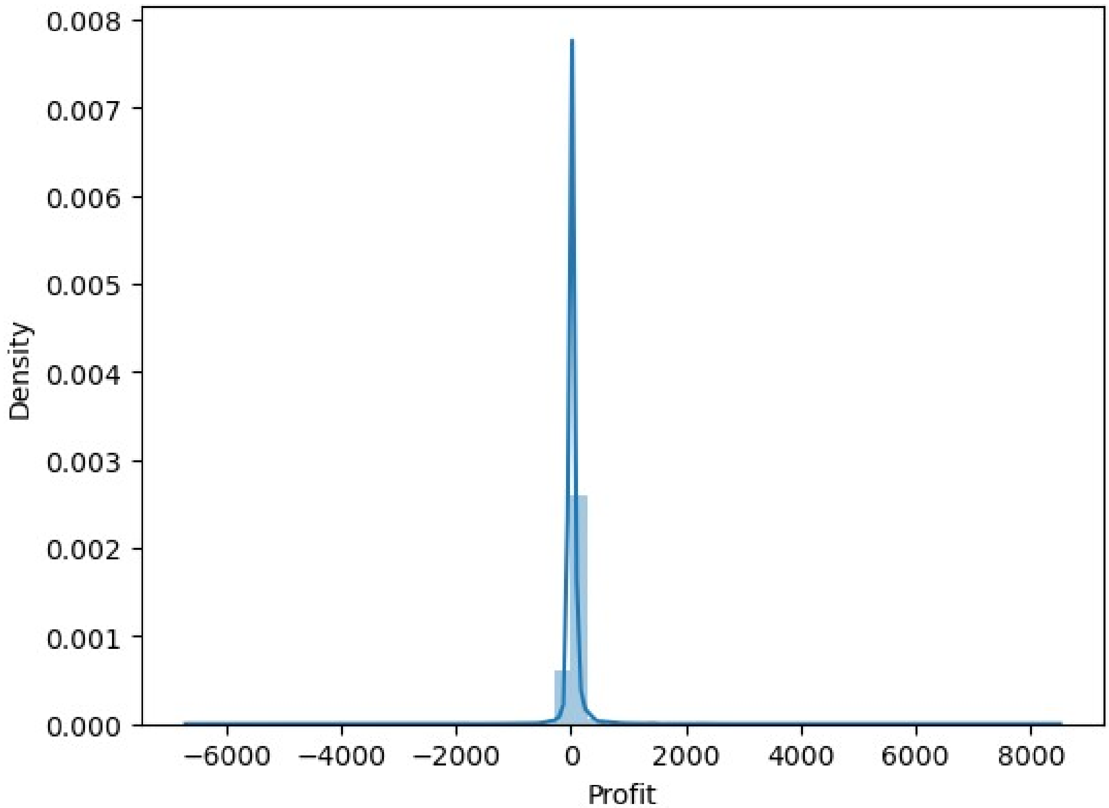
```
9993        2014         10           5       2014

[1881 rows x 25 columns]
```

The above are the Extreme Outliers contains in a Dataset shape is (1881,25)

```
store[(store['Profit'] >LE) & (store['Profit'] <HE)]

      Row ID        Order ID       Ship Mode Customer ID     Customer
Name  \
0          1  CA-2013-152156    Second Class    CG-12520       Claire
Gute
2          3  CA-2013-138688    Second Class    DV-13045   Darrin Van
Huff
4          5  US-2012-108966  Standard Class    SO-20335    Sean
O'Donnell
5          6  CA-2011-115812  Standard Class    BH-11710   Brosina
Hoffman
6          7  CA-2011-115812  Standard Class    BH-11710   Brosina
Hoffman
...      ...             ...             ...         ...
...
9988    9989  CA-2014-163629  Standard Class    RA-19885      Ruben
Ausman
9989    9990  CA-2011-110422    Second Class    TB-21400  Tom
Boeckenhauer
9990    9991  CA-2014-121258  Standard Class    DB-13060        Dave
Brooks
9991    9992  CA-2014-121258  Standard Class    DB-13060        Dave
Brooks
9992    9993  CA-2014-121258  Standard Class    DB-13060        Dave
Brooks

        Segment        Country             City       State  Postal
Code  ...  \
0      Consumer  United States        Henderson    Kentucky
42420  ...
2     Corporate  United States      Los Angeles  California
90036  ...
4      Consumer  United States  Fort Lauderdale     Florida
33311  ...
5      Consumer  United States      Los Angeles  California
90032  ...
6      Consumer  United States      Los Angeles  California
90032  ...
...         ...            ...              ...         ...           .
..  ...
9988  Corporate  United States           Athens     Georgia
30605  ...
9989   Consumer  United States            Miami     Florida
```

```
33180  ...
9990   Consumer  United States       Costa Mesa  California
92627  ...
9991   Consumer  United States       Costa Mesa  California
92627  ...
9992   Consumer  United States       Costa Mesa  California
92627  ...

        Sales Quantity Discount   Profit Order_Date  Order_Month
Order_Year  \
0     261.960        2      0.0  41.9136          9           11
2013
2      14.620        2      0.0   6.8714         13            6
2013
4      22.368        2      0.2   2.5164         11           10
2012
5      48.860        7      0.0  14.1694          9            6
2011
6       7.280        4      0.0   1.9656          9            6
2011
...       ...      ...      ...      ...        ...          ...
...
9988  206.100        5      0.0  55.6470         18           11
2014
9989   25.248        3      0.2   4.1028         22            1
2011
9990   91.960        2      0.0  15.6332         27            2
2014
9991  258.576        2      0.2  19.3932         27            2
2014
9992   29.600        4      0.0  13.3200         27            2
2014

      Ship_Date  Ship_Month  Ship_Year
0            12          11       2013
2            17           6       2013
4            18          10       2012
5            14           6       2011
6            14           6       2011
...         ...         ...        ...
9988         22          11       2014
9989         24           1       2011
9990          4           3       2014
9991          4           3       2014
9992          4           3       2014

[8113 rows x 25 columns]

sns.distplot(store['Profit'])
```

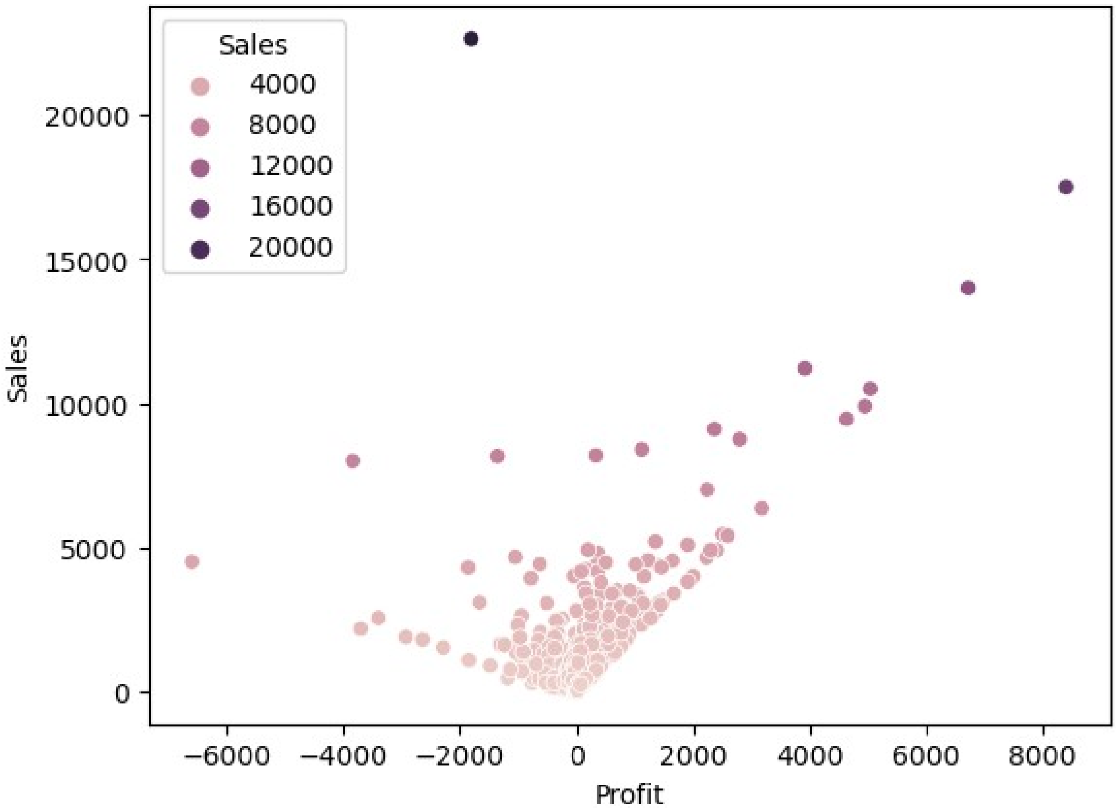
```
<Axes: xlabel='Profit', ylabel='Density'>
```



Without removed outliers disrtibution plot

```
sns.scatterplot(x="Profit",y="Sales" , data=store,hue="Sales")
```

```
<Axes: xlabel='Profit', ylabel='Sales'>
```

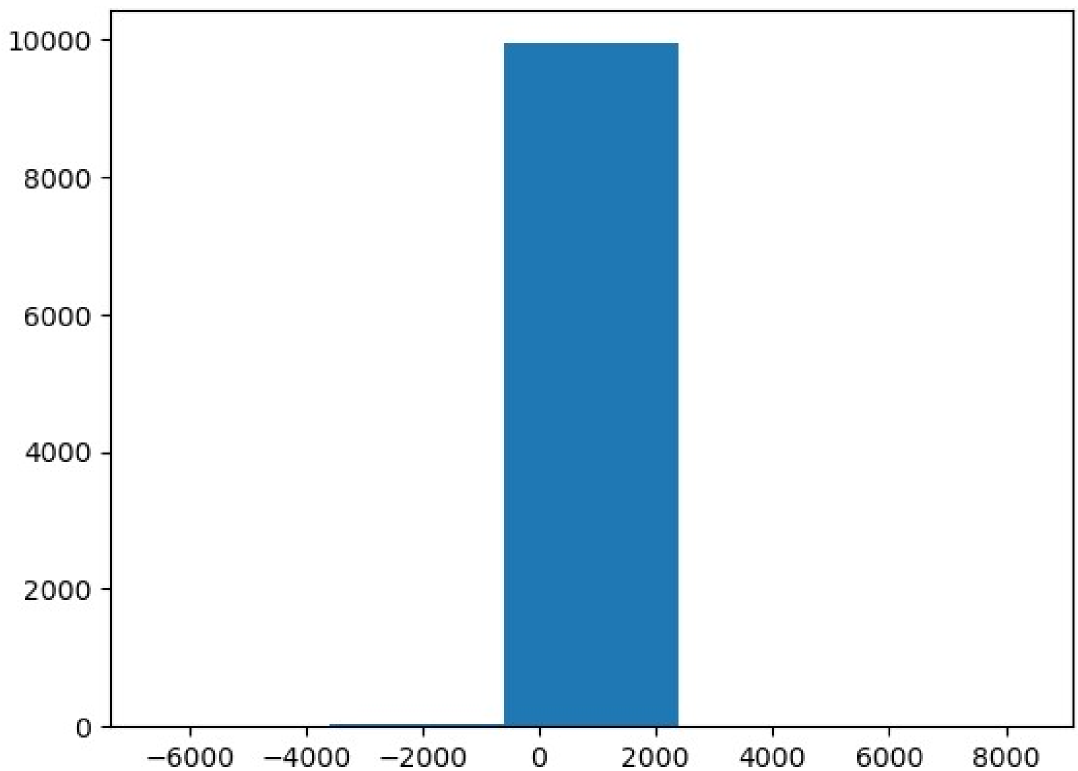


Without removed outliers Scatter Plot

These are the with in Inter Quartile Range shape is (8113,25)

```
plt.hist(store.Profit,bins=5)
```

```
(array([3.000e+00, 3.800e+01, 9.942e+03, 9.000e+00, 2.000e+00]),
 array([-6599.978 , -3599.9872,  -599.9964,  2399.9944,  5399.9852,
         8399.976 ]),
 <BarContainer object of 5 artists>)
```




```
store.Profit.describe()
```

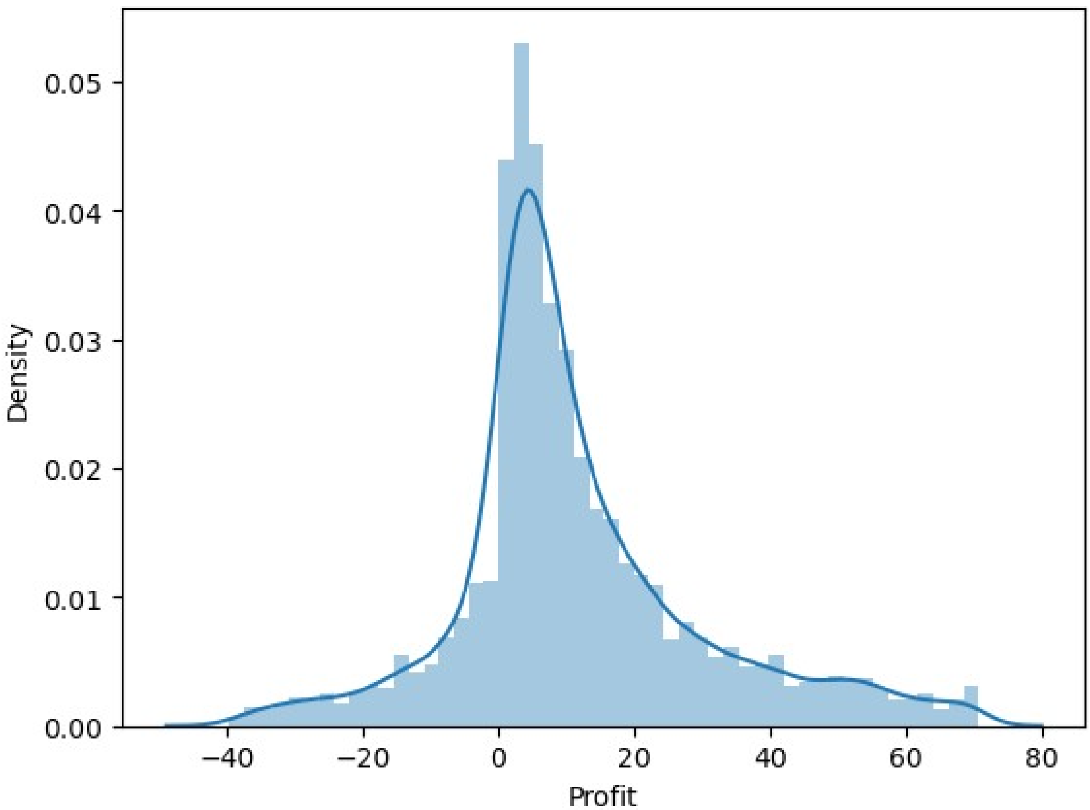
```
count    9994.000000
mean       28.656896
std       234.260108
min     -6599.978000
25%         1.728750
50%         8.666500
75%        29.364000
max      8399.976000
Name: Profit, dtype: float64
```

Triming - Treating the outliers

```
store = store[(store['Profit'] >LE) & (store['Profit'] <HE)]
```

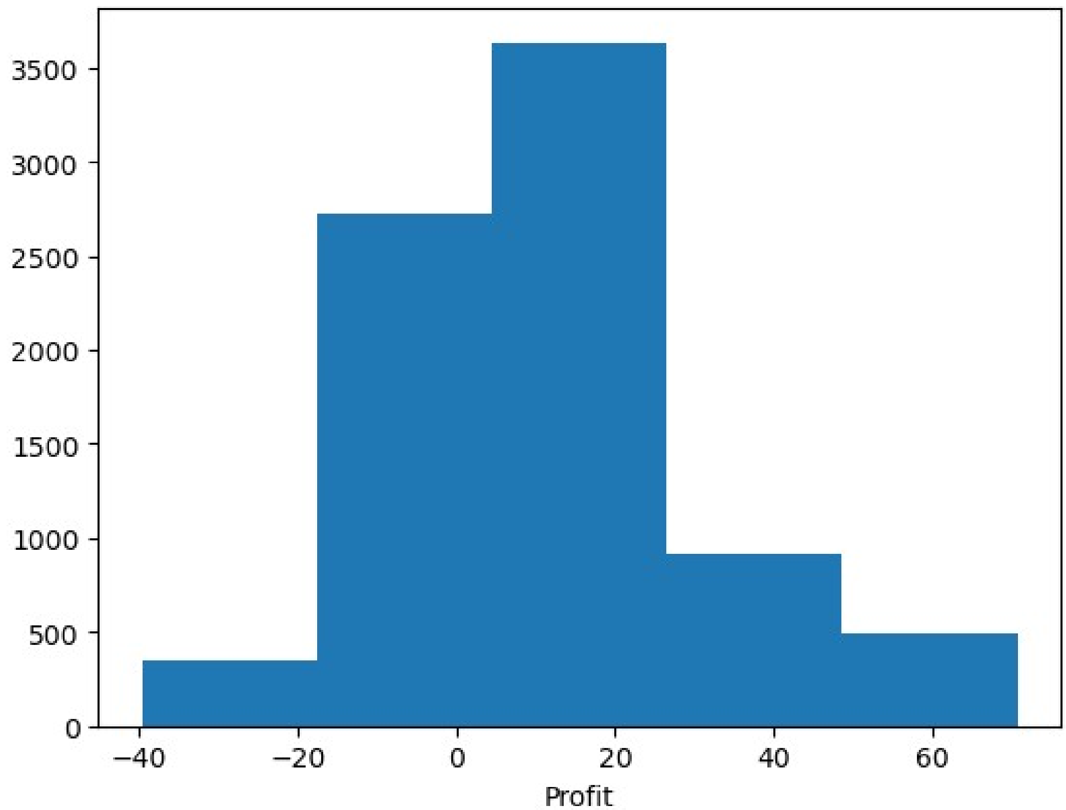
```
sns.distplot(store['Profit'])
```

```
<Axes: xlabel='Profit', ylabel='Density'>
```



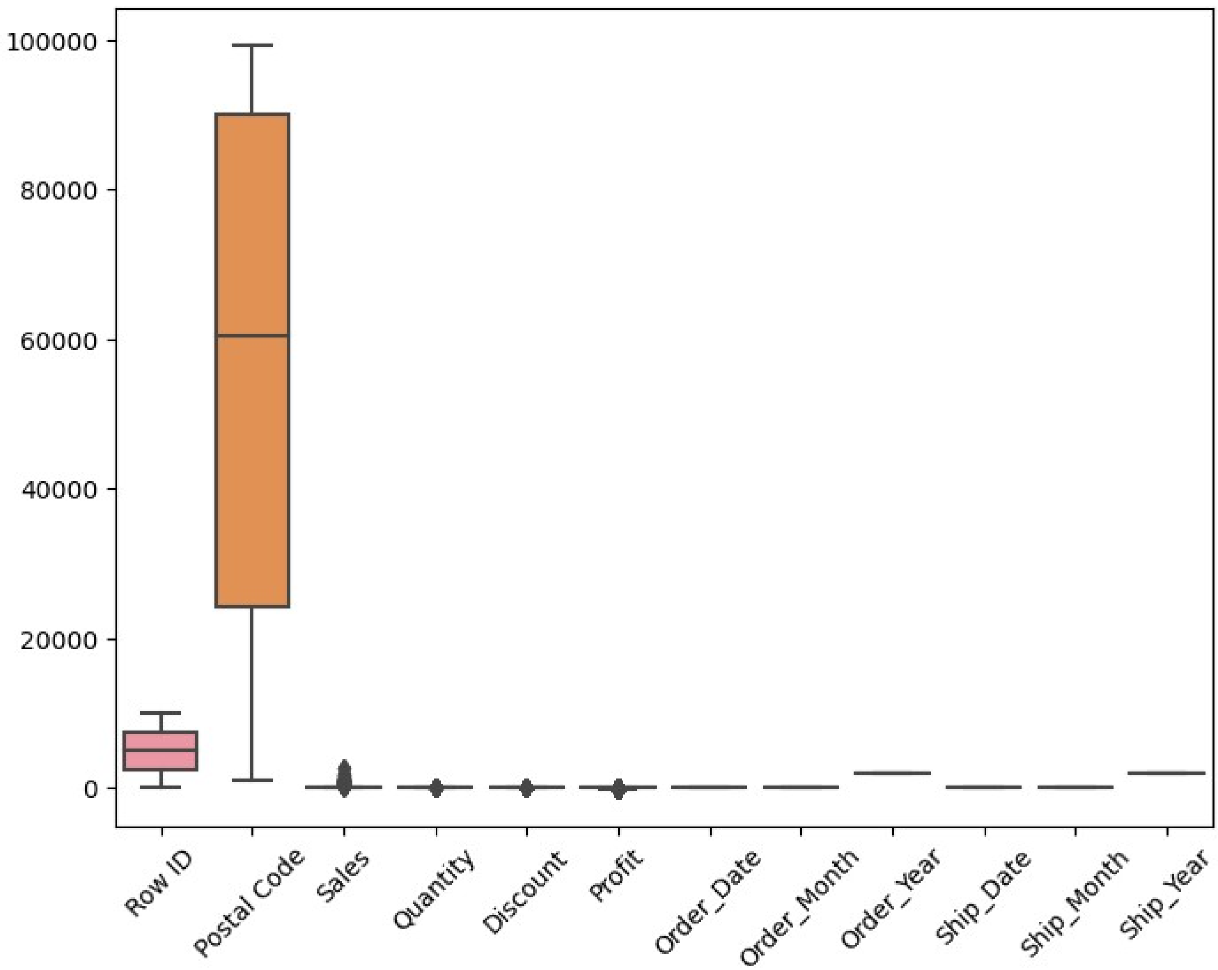
After removed outliers the disrtibution Plot looks like noramly distributed

```
plt.hist(store.Profit,bins=5)
plt.xlabel("Profit")

Text(0.5, 0, 'Profit')
```



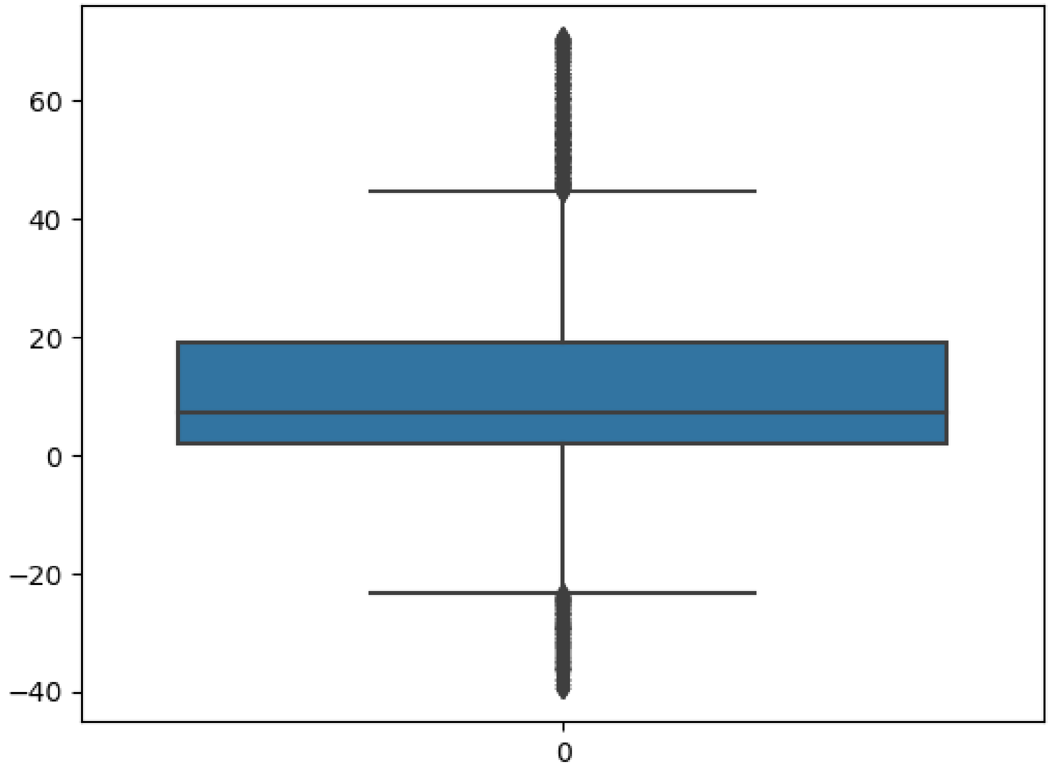

```
plt.figure(figsize=(8, 6))
sns.boxplot(store)
plt.xticks(rotation = 45)
```

```
(array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11]),
 [Text(0, 0, 'Row ID'),
  Text(1, 0, 'Postal Code'),
  Text(2, 0, 'Sales'),
  Text(3, 0, 'Quantity'),
  Text(4, 0, 'Discount'),
  Text(5, 0, 'Profit'),
  Text(6, 0, 'Order_Date'),
  Text(7, 0, 'Order_Month'),
  Text(8, 0, 'Order_Year'),
  Text(9, 0, 'Ship_Date'),
  Text(10, 0, 'Ship_Month'),
  Text(11, 0, 'Ship_Year')])
```




```
sns.boxplot(store.Profit)
```

```
<Axes: >
```

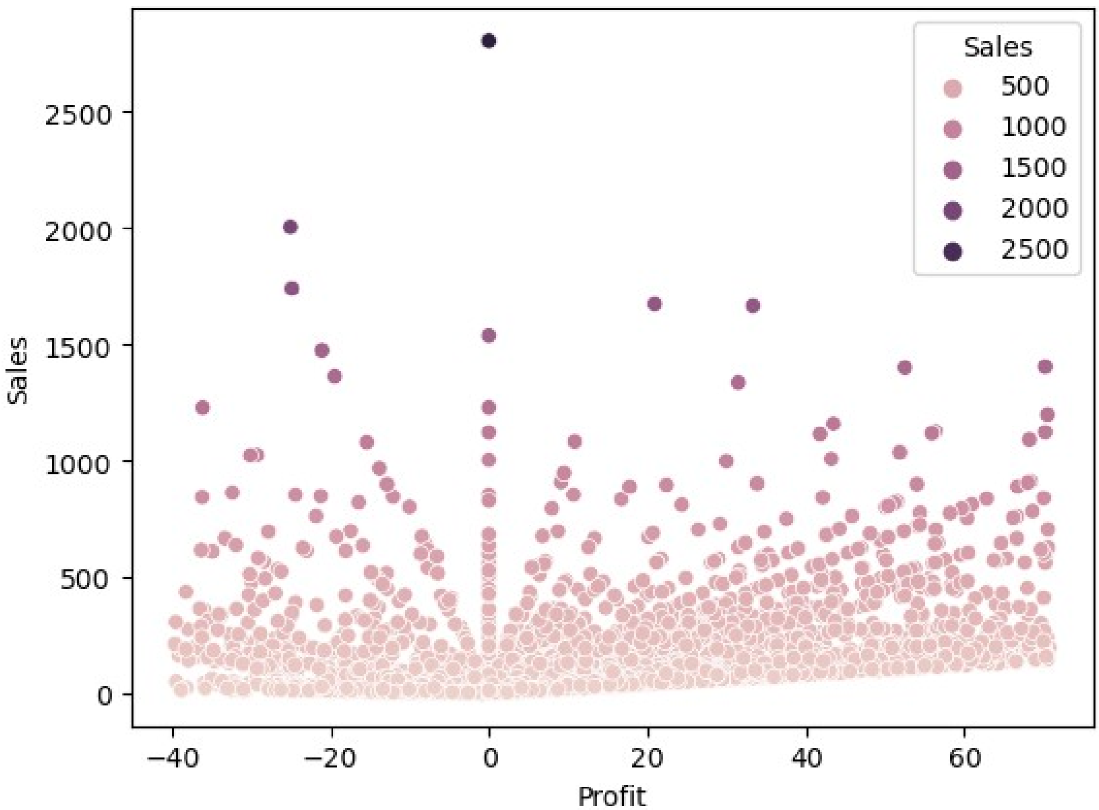



```
store.Profit.describe()
```

```
count    8113.000000
mean       11.604086
std        18.641425
min       -39.637000
25%         2.049200
50%         7.257600
75%        19.034400
max        70.722000
Name: Profit, dtype: float64
```

```
sns.scatterplot(x="Profit",y="Sales",data=store,hue="Sales")
```

```
<Axes: xlabel='Profit', ylabel='Sales'>
```



Featching entire information about dataset

```
store.info()  ##Return data types,null values,total_rows,total_column in a dataset
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 8113 entries, 0 to 9992
Data columns (total 25 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Row ID         8113 non-null   int64
 1   Order ID       8113 non-null   object
 2   Ship Mode      8113 non-null   object
 3   Customer ID    8113 non-null   object
 4   Customer Name  8113 non-null   object
 5   Segment        8113 non-null   object
 6   Country        8113 non-null   object
 7   City           8113 non-null   object
 8   State          8113 non-null   object
 9   Postal Code    8113 non-null   int64
 10  Region         8113 non-null   object
 11  Product ID     8113 non-null   object
 12  Category       8113 non-null   object
 13  Sub-Category   8113 non-null   object
 14  Product Name   8113 non-null   object
```

```
 15   Sales           8113 non-null   float64
 16   Quantity        8113 non-null   int64
 17   Discount        8113 non-null   float64
 18   Profit          8113 non-null   float64
 19   Order_Date      8113 non-null   int64
 20   Order_Month     8113 non-null   int64
 21   Order_Year      8113 non-null   int64
 22   Ship_Date       8113 non-null   int64
 23   Ship_Month      8113 non-null   int64
 24   Ship_Year       8113 non-null   int64
dtypes: float64(3), int64(9), object(13)
memory usage: 1.9+ MB
```

```
store.shape     #Checking Shape in an dataset
```

```
(8113, 25)
```

```
store.index     #Checking the index in an dataset
```

```
Int64Index([   0,    2,    4,    5,    6,    8,    9,   11,   12,
15,
            ...
            9983, 9984, 9985, 9986, 9987, 9988, 9989, 9990, 9991,
9992],
           dtype='int64', length=8113)
```

Checking the statistical_info

```
store.describe()   #Checking the statistical_info in an dataset
```

```
            Row ID   Postal Code        Sales     Quantity
Discount  \
count  8113.000000   8113.000000  8113.000000  8113.000000
8113.000000
mean   4997.032171  56065.988044    91.051345     3.550475
0.148920
std    2889.780636  32077.418326   156.189857     2.076080
0.197695
min       1.000000   1040.000000     0.444000     1.000000
0.000000
25%    2499.000000  24153.000000    13.970000     2.000000
0.000000
50%    4983.000000  60505.000000    35.440000     3.000000
0.200000
75%    7497.000000  90032.000000    99.870000     5.000000
0.200000
max    9993.000000  99301.000000  2803.920000    14.000000
0.800000

            Profit   Order_Date  Order_Month   Order_Year    Ship_Date
```

```
\
count   8113.000000  8113.000000  8113.000000  8113.000000  8113.000000

mean      11.604086    15.651917     7.798841  2012.727105    15.893258

std       18.641425     8.712501     3.282584     1.124620     8.787287

min      -39.637000     1.000000     1.000000  2011.000000     1.000000

25%        2.049200     8.000000     5.000000  2012.000000     8.000000

50%        7.257600    16.000000     9.000000  2013.000000    16.000000

75%       19.034400    23.000000    11.000000  2014.000000    24.000000

max       70.722000    31.000000    12.000000  2014.000000    31.000000


        Ship_Month    Ship_Year
count  8113.000000  8113.000000
mean      7.721188  2012.743745
std       3.340611     1.129456
min       1.000000  2011.000000
25%       5.000000  2012.000000
50%       8.000000  2013.000000
75%      11.000000  2014.000000
max      12.000000  2015.000000
```

Checking the Target Variable Statstical info

```
store.Profit.describe()   #Here i found there are outliers are present
in my target variable

count    8113.000000
mean       11.604086
std        18.641425
min       -39.637000
25%         2.049200
50%         7.257600
75%        19.034400
max        70.722000
Name: Profit, dtype: float64
```

Feteching only objects in a column

```
store.describe(include='object').columns

Index(['Order ID', 'Ship Mode', 'Customer ID', 'Customer Name',
'Segment',
       'Country', 'City', 'State', 'Region', 'Product ID', 'Category',
```

```
       'Sub-Category', 'Product Name'],
      dtype='object')
```

Feteching only integers and float values in a column

```
store.describe(include=['int','float']).columns
```

```
Index(['Row ID', 'Postal Code', 'Sales', 'Quantity', 'Discount',
'Profit',
       'Order_Date', 'Order_Month', 'Order_Year', 'Ship_Date',
'Ship_Month',
       'Ship_Year'],
      dtype='object')
```

Featching the repeated/duplicated values

```
store['Customer Name'].value_counts()   #Return the
Repeated/Duplicated customer names interms of values
```

```
William Brown          31
Chloris Kastensmidt    30
Matt Abelman           29
Arthur Prichep         28
Paul Prost             27
                       ..
Carl Jackson            1
Stefanie Holloman       1
Lela Donovan            1
Ricardo Emerson         1
Paul Knutson            1
Name: Customer Name, Length: 790, dtype: int64
```

```
store['Customer Name'].count()    #Return the total row records
```

```
8113
```

```
len(store)  #Return length of dataset
```

```
8113
```

```
store.sort_values(by=['Customer Name'],ascending=False,axis=0)  #
Descending the Customer Names based on rows.
```

```
      Row ID        Order ID       Ship Mode Customer ID
Customer Name  \
3814    3815  CA-2013-152471        Same Day    ZD-21925  Zuschuss
Donatelli
18        19  CA-2011-143336    Second Class    ZD-21925  Zuschuss
Donatelli
3040    3041  US-2013-147991  Standard Class    ZD-21925  Zuschuss
Donatelli
```

```
20        21  CA-2011-143336    Second Class    ZD-21925  Zuschuss
Donatelli
19        20  CA-2011-143336    Second Class    ZD-21925  Zuschuss
Donatelli
...      ...             ...             ...         ...
...
4962    4963  CA-2011-156587     First Class    AB-10015       Aaron
Bergman
4961    4962  CA-2011-156587     First Class    AB-10015       Aaron
Bergman
8801    8802  CA-2013-140935     First Class    AB-10015       Aaron
Bergman
8222    8223  CA-2011-152905  Standard Class    AB-10015       Aaron
Bergman
8802    8803  CA-2013-140935     First Class    AB-10015       Aaron
Bergman

       Segment        Country            City       State  Postal Code
...  \
3814  Consumer  United States    Jacksonville     Florida        32216
...
18    Consumer  United States   San Francisco  California        94109
...
3040  Consumer  United States     Chattanooga   Tennessee        37421
...
20    Consumer  United States   San Francisco  California        94109
...
19    Consumer  United States   San Francisco  California        94109
...
...        ...            ...             ...         ...          ...
...
4962  Consumer  United States         Seattle  Washington        98103
...
4961  Consumer  United States         Seattle  Washington        98103
...
8801  Consumer  United States   Oklahoma City    Oklahoma        73120
...
8222  Consumer  United States       Arlington       Texas        76017
...
8802  Consumer  United States   Oklahoma City    Oklahoma        73120
...

        Sales Quantity Discount   Profit Order_Date  Order_Month
Order_Year  \
3814  823.960        5      0.2  51.4975          9            7
2013
18      8.560        2      0.0   2.4824         27            8
2011
3040   16.720        5      0.2   3.3440          6            5
```

```
2013
20     22.720        4      0.2   7.3840         27            8
2011
19    213.480        3      0.2  16.0110         27            8
2011
...       ...      ...      ...      ...        ...          ...
...
4962   17.940        3      0.0   4.6644          7            3
2011
4961   48.712        1      0.2   5.4801          7            3
2011
8801  221.980        2      0.0  62.1544         11           11
2013
8222   12.624        2      0.2  -2.5248         19            2
2011
8802  341.960        2      0.0  54.7136         11           11
2013

      Ship_Date  Ship_Month  Ship_Year
3814          9           7       2013
18            1           9       2011
3040         10           5       2013
20            1           9       2011
19            1           9       2011
...         ...         ...        ...
4962          8           3       2011
4961          8           3       2011
8801         13          11       2013
8222         25           2       2011
8802         13          11       2013

[8113 rows x 25 columns]
```

1.Based on Average Profit and Average Sales by Product Category

```
Average_Sales_Profit=store.groupby(['Category'])
['Sales','Profit'].mean()
Average_Sales_Profit
```

```
                      Sales     Profit
Category
Furniture        174.215895   9.939868
Office Supplies   52.991658  10.307098
Technology       152.889336  18.969933
```

Furniture has the highest sales but the lowest profit to others.

Suggestions:-(High sales might be due to a wide variety of products. Consider focusing on higher-margin items.)

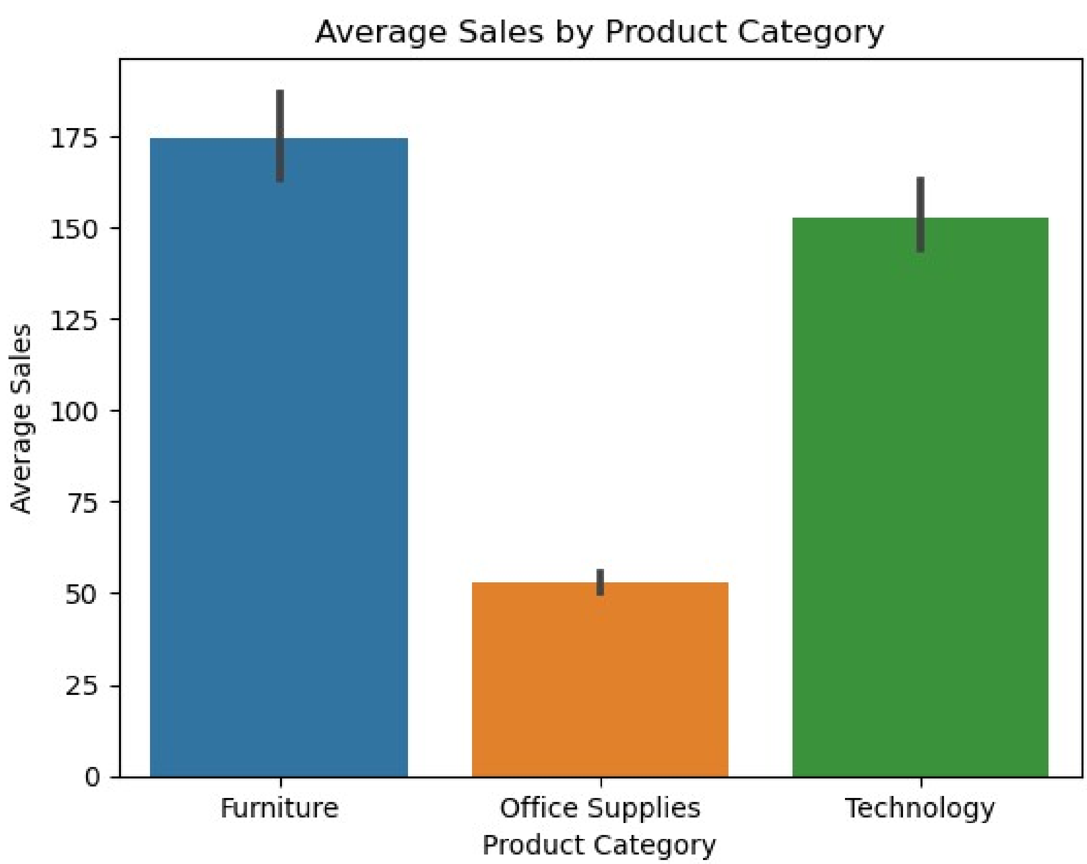
Technology comes second in sales and has the highest profit to others.

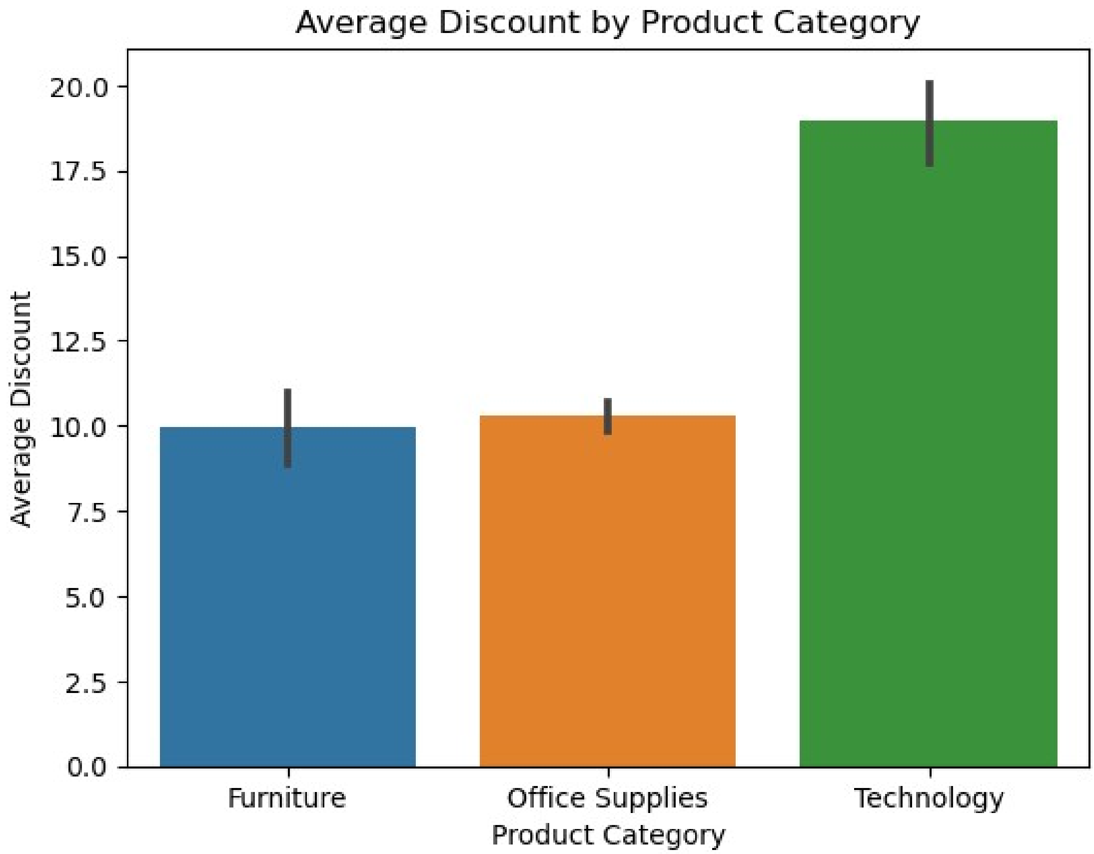
Suggestions:-(Maintain quality and customer satisfaction to sustain high-profit margins)

Office Supplies have both the lowest sales and the lowest profit.

Suggestions:-(Assess the demand for office supplies and potentially lookup for more product offerings.)

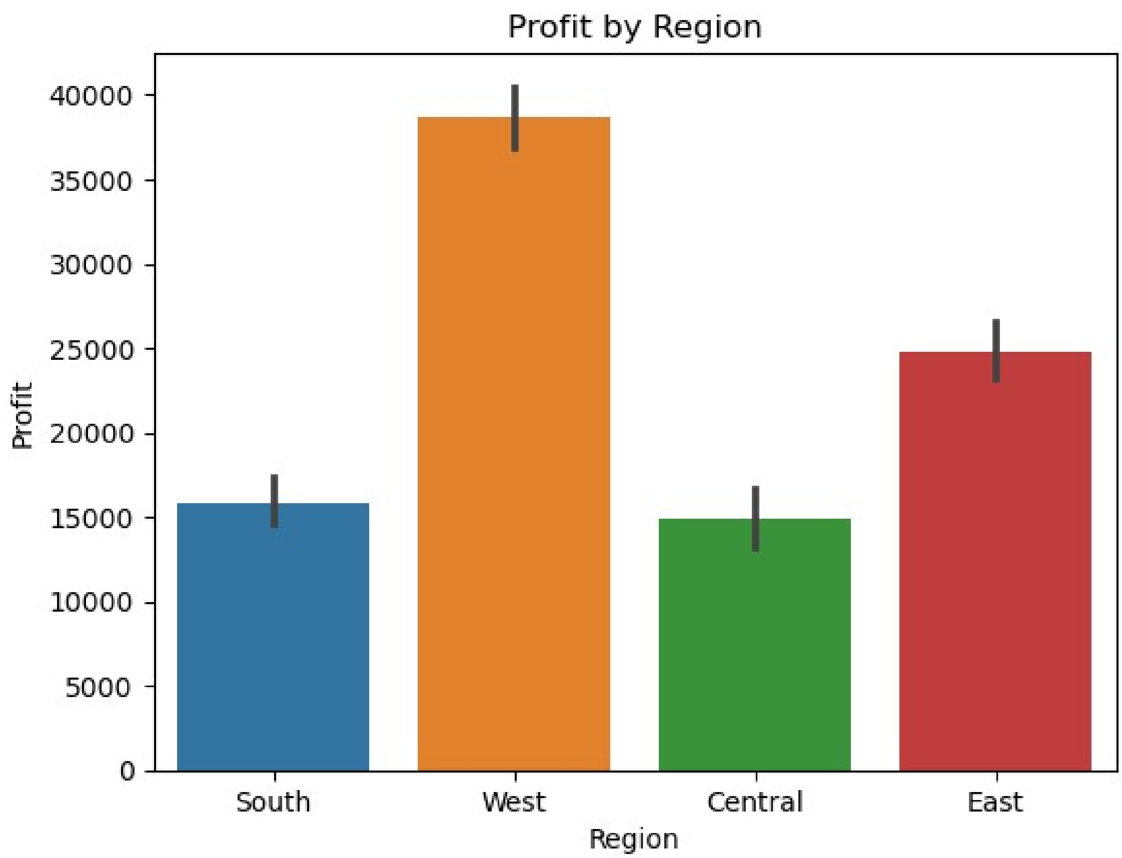
```
sns.barplot(data=store,x='Category',y='Sales',estimator='mean')
plt.xlabel('Product Category')
plt.ylabel('Average Sales')
plt.title('Average Sales by Product Category')
plt.show()

sns.barplot(data=store,x='Category',y='Profit',estimator='mean')
plt.xlabel('Product Category')
plt.ylabel('Average Discount')
plt.title('Average Discount by Product Category')
plt.show()
```

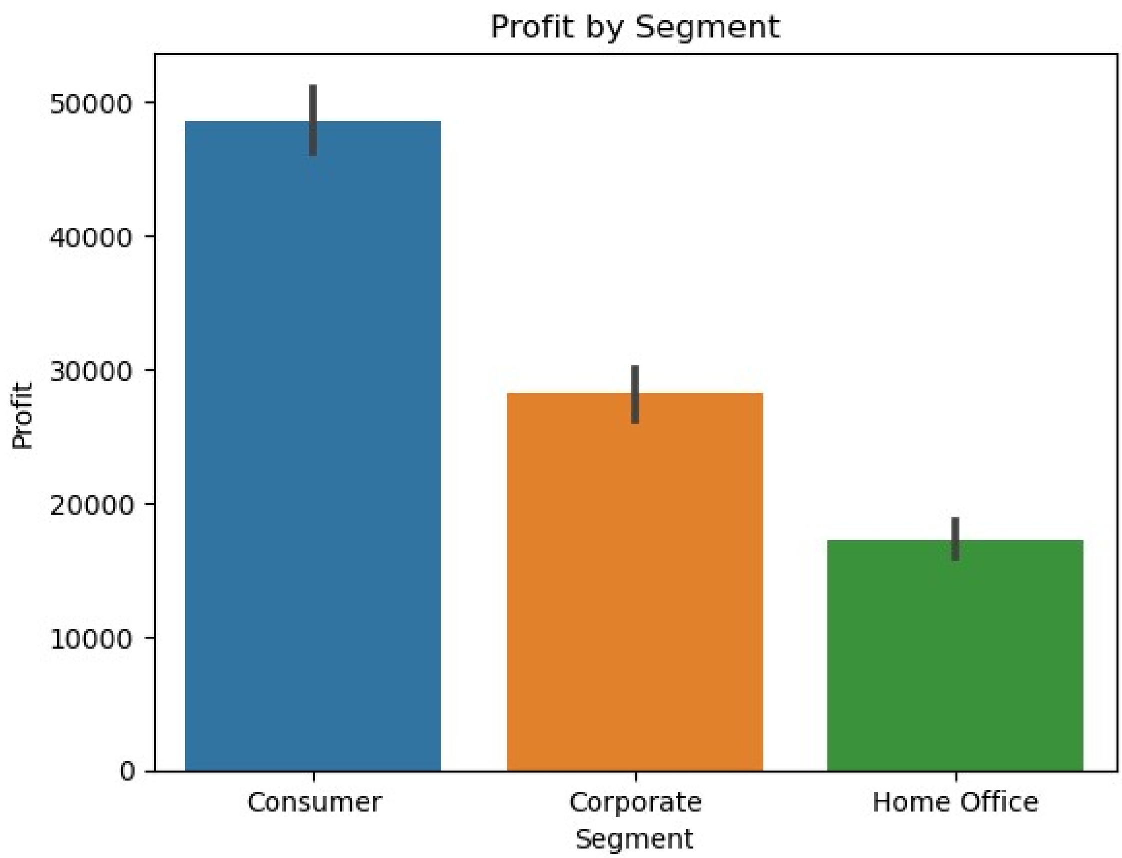




2.Profit by Region and Segment

```
profit_by_region_segment = store.groupby(['Region', 'Segment'])
['Profit'].sum().unstack()
profit_by_region_segment

Segment     Consumer    Corporate  Home Office
Region
Central    7344.6517    4542.3047    3053.8082
East      12517.7859    7677.8315    4557.8370
South      8348.9505    4593.4707    2883.5460
West      20422.4240   11407.4321    6793.9073

sns.barplot(data=store,x='Region',y='Profit',estimator='sum')
plt.xlabel('Region')
plt.ylabel('Profit')
plt.title('Profit by Region')
plt.show()

sns.barplot(data=store,x='Segment',y='Profit',estimator='sum')
plt.xlabel('Segment')
plt.ylabel('Profit')
plt.title('Profit by Segment')
plt.show()
```


Profit by Region
40000
35000
30000
25000
20000
15000
10000
5000
0
Profit
South
West
Central
East
Region


Profit by Segment
50000
40000
30000
20000
10000
0
Profit
Consumer
Corporate
Home Office
Segment

Consumer Segment:-This segment likely includes individual customers, representing people who make purchases for personal use. They may buy products for their homes, personal offices, or other individual needs.

Home Office Segment:-This segment is likely focused on customers who have a specific home office setup. These customers may purchase office supplies, equipment, or furniture for their home-based work or businesses.

Corporate Segment:-The corporate segment typically includes businesses and organizations. These customers may make bulk purchases of office supplies, furniture, and technology equipment for their employees or company use.

When considering the impact of region, it's notable that the West Region leads with the highest profit, followed closely by the East Region as the second-highest.

The other two regions have fairly similar profit levels. Additionally, within each region,

the Consumer segment consistently generates the highest profit, while the Corporate segment comes in second. On the other, the Home Office segment consistently records the lowest profit across all regions.

3.Based On Customer Name, Ship Date and Ship Mode by average of Profit, Quantity and Sales

```
store_1= pd.pivot_table(store, index=['Customer
Name','Ship_Date','Ship Mode'], values=['Quantity', 'Sales','Profit'],
aggfunc='mean')
store_1
```

```
                                               Profit  Quantity
Sales
Customer Name      Ship_Date Ship Mode

Aaron Bergman      8         First Class      5.00110  2.333333
103.197333
                   13        First Class     58.43400  2.000000
281.970000
                   25        Standard Class  -2.52480  2.000000
12.624000
Aaron Hawkins      1         First Class     10.08405  3.000000
26.835000
                   19        Standard Class  12.39960  4.000000
143.728000
...                                               ...       ...
...
Zuschuss Carroll   29        Standard Class -23.78220  3.000000
128.058000
```

```
Zuschuss Donatelli 1         Second Class     8.62580  3.000000
81.586667
                   9         Same Day        28.24625  3.500000
419.972000
                   10        Standard Class  12.69560  4.500000
43.920000
                   15        First Class     16.58880  3.000000
61.440000

[4344 rows x 3 columns]
```

```
store.groupby('Ship Mode')['Profit'].mean()
```

```
Ship Mode
First Class       11.339917
Same Day          10.691748
Second Class      12.158873
Standard Class    11.575178
Name: Profit, dtype: float64
```

1.Second class ship mode has highest Profit.

2.First class and Second class has the second highest Profit.

3.Same day ship mode has less profit

Suggestions:-Continuously monitor customer feedback and satisfaction to ensure that shipping options meet their expectations.

4. Based on Customer ID checking the average of Sales,Quantity,Profit

```
kiran=store.groupby("Customer ID")
[["Sales","Quantity","Profit"]].mean()
kiran.head()
```

```
                  Sales  Quantity     Profit
Customer ID
AA-10315      74.622500  2.750000   9.976113
AA-10375      39.743571  2.785714  11.599071
AA-10480      86.270200  2.900000  17.411280
AA-10645     147.464667  3.466667  11.285753
AB-10015     147.692667  2.166667  21.557750
```

5.What is the correlation between discounts and the quantity of products sold?

```
def histogram_intersection(discounts,quantity):
    k = np.minimum(discounts,quantity).sum().round(decimals=1)
```

```
    return k
store.corr(method=histogram_intersection)
```

```
                 Row ID  Postal Code      Sales  Quantity  Discount
Profit  \
Row ID              1.0   39952972.0   721858.6   28803.0    1208.2
94005.4
Postal Code  39952972.0          1.0   738699.6   28805.0    1208.2
94143.9
Sales          721858.6     738699.6        1.0   28701.1    1207.6
94143.9
Quantity        28803.0      28805.0    28701.1       1.0    1208.2
6941.7
Discount         1208.2       1208.2     1207.6    1208.2       1.0 -
15220.8
Profit          94005.4      94143.9    94143.9    6941.7  -15220.8
1.0
Order_Date     126925.0     126984.0   105587.0   27066.0    1208.2
44363.6
Order_Month     63254.0      63272.0    59446.0   26268.0    1208.2
24401.9
Order_Year   14675384.0   16317969.0   737117.7   28805.0    1208.2
94143.9
Ship_Date      128859.0     128942.0   106649.5   27042.0    1208.2
44105.1
Ship_Month      62624.0      62642.0    58893.0   26089.0    1208.2
24084.0
Ship_Year    14675499.0   16318102.0   737117.7   28805.0    1208.2
94143.9

             Order_Date  Order_Month  Order_Year  Ship_Date
Ship_Month  \
Row ID         126925.0      63254.0  14675384.0   128859.0
62624.0
Postal Code    126984.0      63272.0  16317969.0   128942.0
62642.0
Sales          105587.0      59446.0    737117.7   106649.5
58893.0
Quantity        27066.0      26268.0     28805.0    27042.0
26089.0
Discount         1208.2       1208.2      1208.2     1208.2
1208.2
Profit          44363.6      24401.9     94143.9    44105.1
24084.0
Order_Date          1.0      54621.0    126984.0   101383.0
53991.0
Order_Month     54621.0          1.0     63272.0    55045.0
61787.0
Order_Year     126984.0      63272.0         1.0   128942.0
62642.0
```

```
Ship_Date      101383.0      55045.0    128942.0        1.0
54907.0
Ship_Month      53991.0      61787.0     62642.0    54907.0
1.0
Ship_Year      126984.0      63272.0  16329255.0   128942.0
62642.0

               Ship_Year
Row ID       14675499.0
Postal Code  16318102.0
Sales          737117.7
Quantity        28805.0
Discount         1208.2
Profit          94143.9
Order_Date     126984.0
Order_Month     63272.0
Order_Year   16329255.0
Ship_Date      128942.0
Ship_Month      62642.0
Ship_Year           1.0
```

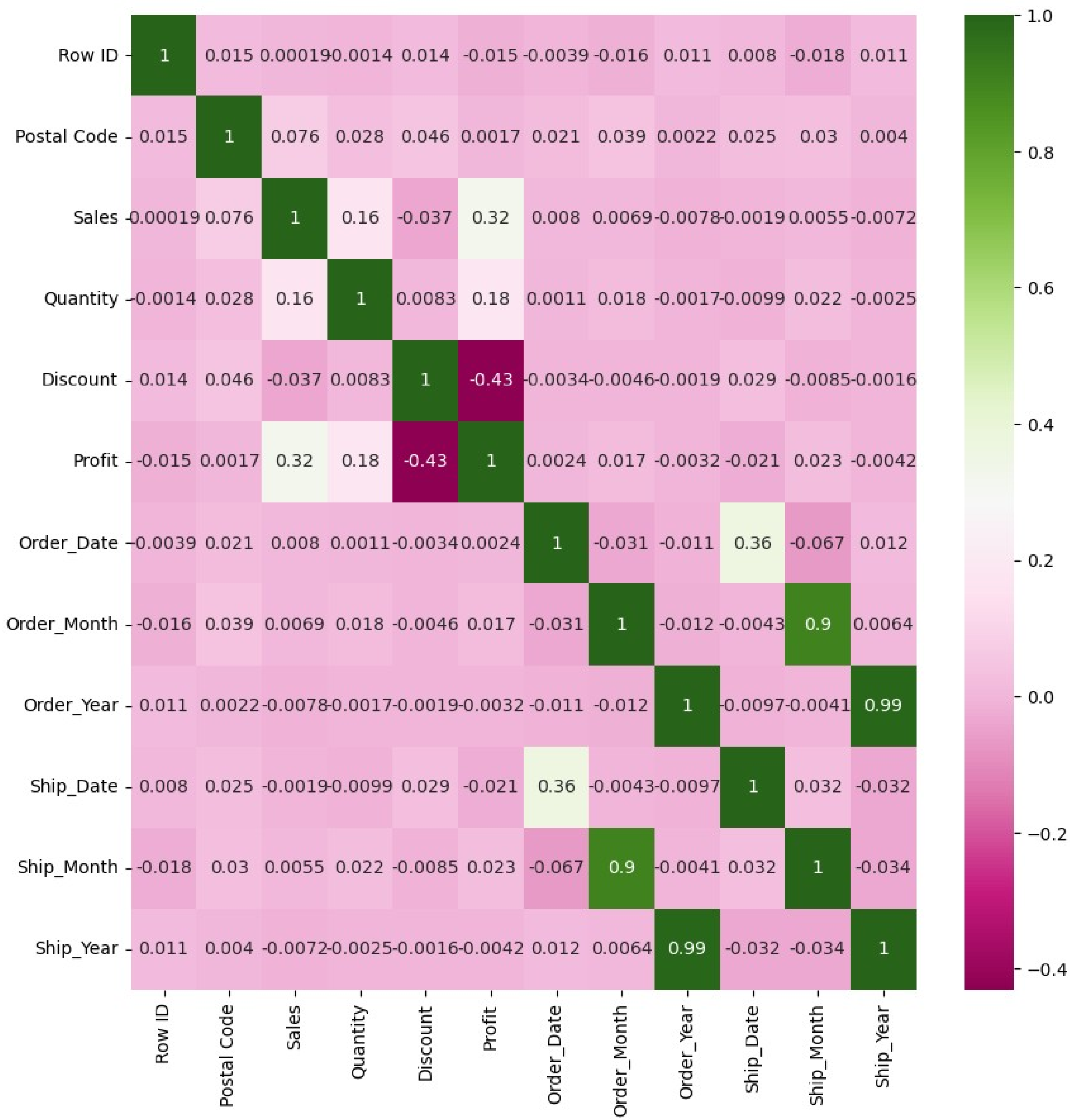
```
store.corr()
```

```
                Row ID  Postal Code     Sales  Quantity  Discount
Profit  \
Row ID        1.000000     0.014808  0.000187 -0.001408  0.014403 -
0.014604
Postal Code   0.014808     1.000000  0.075585  0.027722  0.046284
0.001696
Sales         0.000187     0.075585  1.000000  0.162896 -0.037185
0.322309
Quantity     -0.001408     0.027722  0.162896  1.000000  0.008275
0.182883
Discount      0.014403     0.046284 -0.037185  0.008275  1.000000 -
0.431420
Profit       -0.014604     0.001696  0.322309  0.182883 -0.431420
1.000000
Order_Date   -0.003916     0.020724  0.008042  0.001053 -0.003427
0.002350
Order_Month -0.015894     0.039160  0.006931  0.017698 -0.004615
0.016563
Order_Year    0.011027     0.002190 -0.007764 -0.001703 -0.001869 -
0.003222
Ship_Date     0.007969     0.025148 -0.001904 -0.009915  0.028899 -
0.021359
Ship_Month   -0.017702     0.030322  0.005456  0.022097 -0.008493
0.022715
Ship_Year     0.011222     0.004000 -0.007205 -0.002501 -0.001576 -
0.004243
```

```
              Order_Date  Order_Month  Order_Year  Ship_Date  
Ship_Month  \
Row ID         -0.003916    -0.015894    0.011027   0.007969    -
0.017702
Postal Code     0.020724     0.039160    0.002190   0.025148    
0.030322
Sales           0.008042     0.006931   -0.007764  -0.001904    
0.005456
Quantity        0.001053     0.017698   -0.001703  -0.009915    
0.022097
Discount       -0.003427    -0.004615   -0.001869   0.028899    -
0.008493
Profit          0.002350     0.016563   -0.003222  -0.021359    
0.022715
Order_Date      1.000000    -0.030656   -0.010929   0.361860    -
0.067075
Order_Month    -0.030656     1.000000   -0.012468  -0.004257    
0.903806
Order_Year     -0.010929    -0.012468    1.000000  -0.009684    -
0.004078
Ship_Date       0.361860    -0.004257   -0.009684   1.000000    
0.031557
Ship_Month     -0.067075     0.903806   -0.004078   0.031557    
1.000000
Ship_Year       0.011592     0.006443    0.993567  -0.031945    -
0.033706

             Ship_Year
Row ID        0.011222
Postal Code   0.004000
Sales        -0.007205
Quantity     -0.002501
Discount     -0.001576
Profit       -0.004243
Order_Date    0.011592
Order_Month   0.006443
Order_Year    0.993567
Ship_Date    -0.031945
Ship_Month   -0.033706
Ship_Year     1.000000
```

```
plt.figure(figsize=(10, 10))
sns.heatmap(store.corr(method='pearson'),cmap='PiYG',annot = True)
#Correlation techinques:-'pearson', 'spearman', 'kendall',
```

```
<Axes: >
```



1.Sales and Profit are Moderately Correlated

2.Discount and Profit are Negatively Correlated

3.Order_Month and Order_Year are 99% Correlated

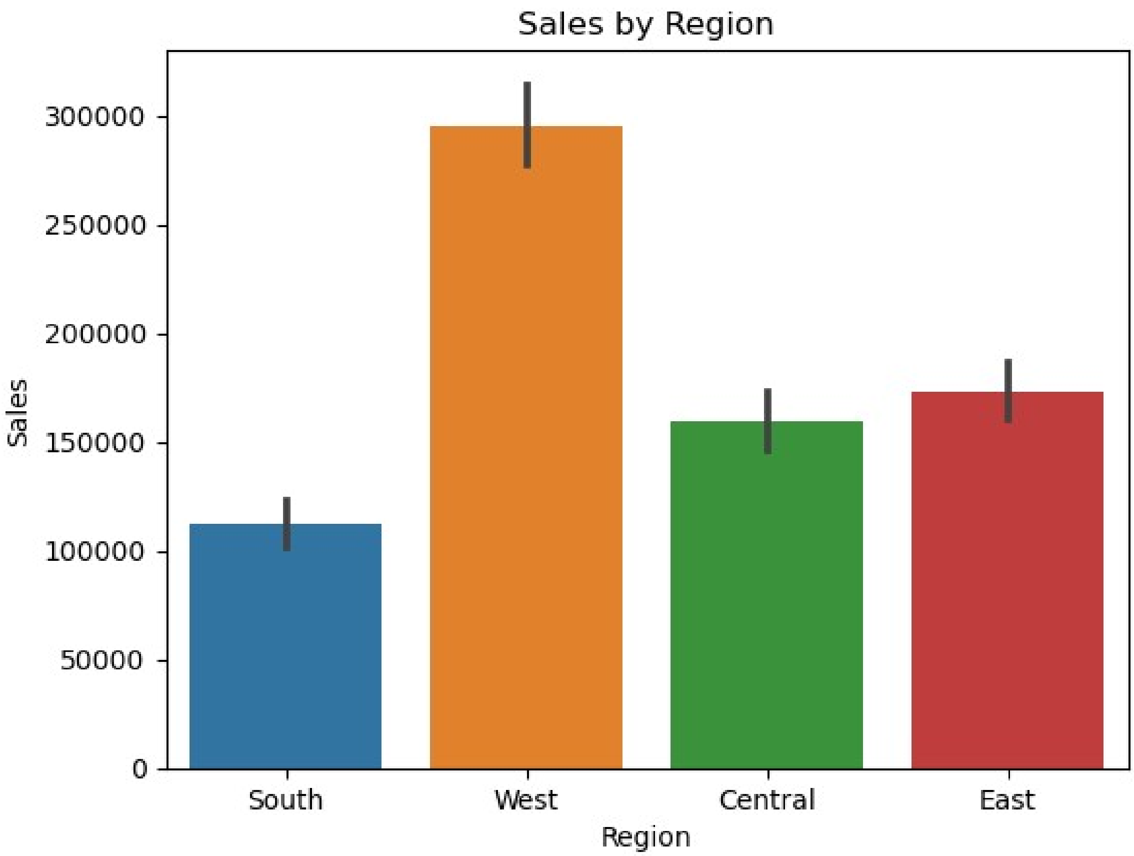
4.Ship_Year And Order_Year are 99% Correlated

6.What is the average Sales value for different geographical regions?

```
store.groupby('Region')[['Sales']].mean()
```

```
              Sales
Region
Central   86.369358
East      76.828996
South     85.027319
West     108.967761
```

```
sns.barplot(data=store,x='Region',y='Sales',estimator='sum')
plt.xlabel('Region')
plt.ylabel('Sales')
plt.title('Sales by Region')
plt.show()
```



When considering the impact of region, it's worth noting that the West Region leads with the highest sales, followed closely by the East Region, which has the second-highest sales levels.

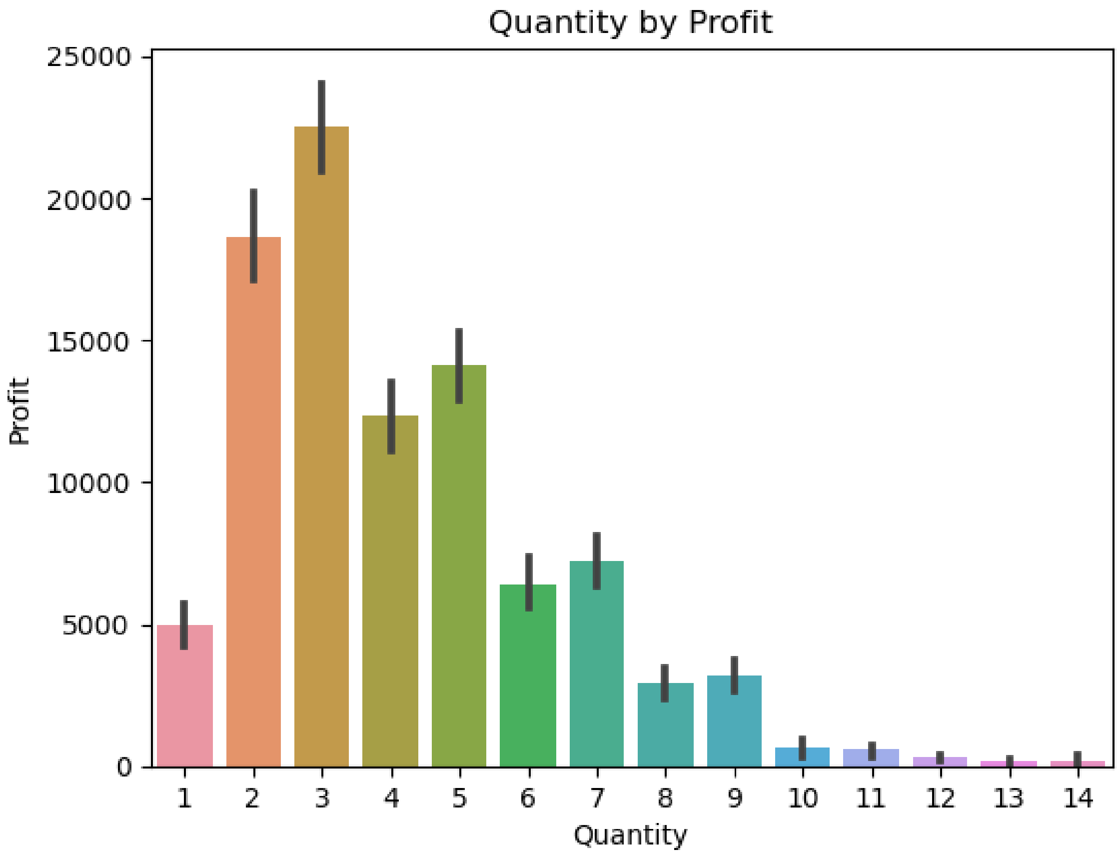
The Central Region comes next in terms of sales, and finally, the South Region has the lowest sales compared to the remaining regions..

7.What are the top-selling products in terms of quantity and profit?

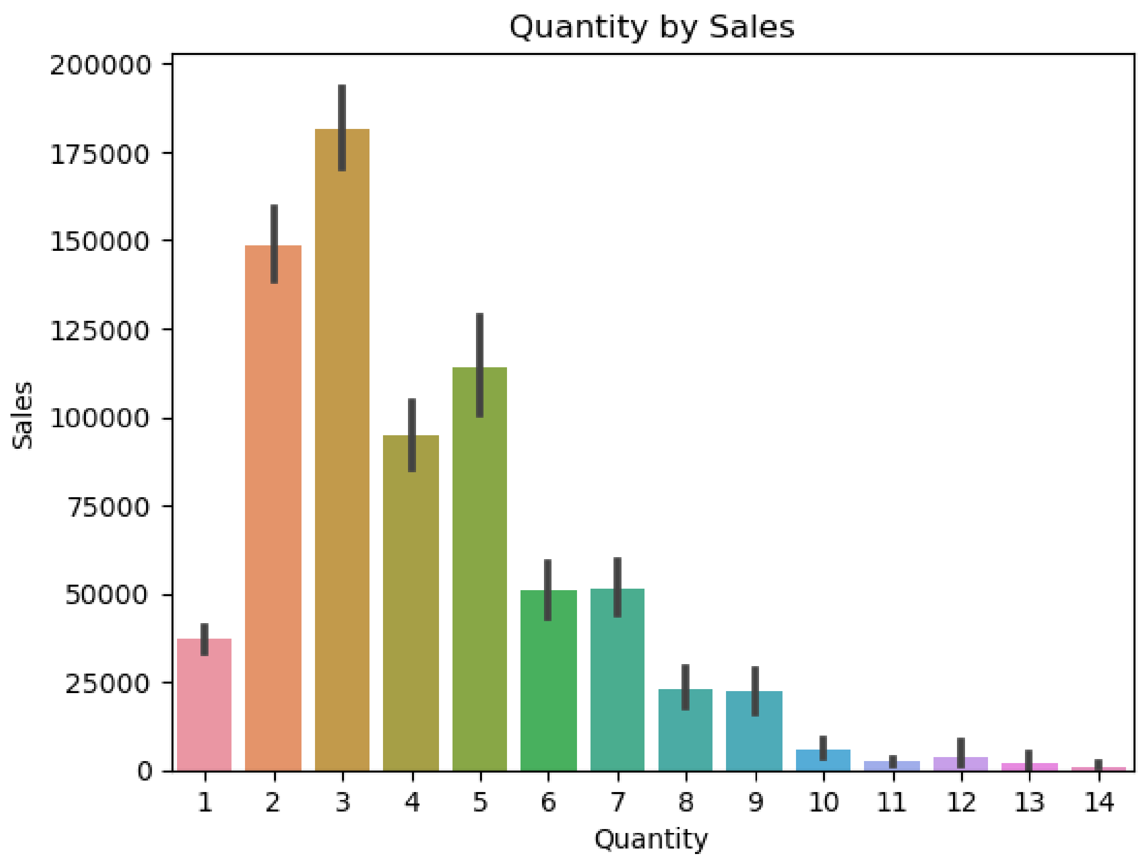
```
store.nlargest(n=5,columns=['Profit','Quantity']).groupby('Product Name')[['Product Name','Quantity','Profit']].head()
```

```
                              Product Name  Quantity   Profit
1866  Howard Miller 12" Round Wall Clock         5  70.7220
3131                           GE 30524EE4         4  70.5564
6635                           GE 30524EE4         4  70.5564
2554                          HTC One Mini         7  70.5544
3683      GBC Recycled VeloBinder Covers         9  70.5456
```

```
sns.barplot(data=store,x='Quantity',y='Profit',estimator='sum')
plt.xlabel('Quantity')
plt.ylabel('Profit')
plt.title('Quantity by Profit')
plt.show()


sns.barplot(data=store,x='Quantity',y='Sales',estimator='sum')
plt.xlabel('Quantity')
plt.ylabel('Sales')
plt.title('Quantity by Sales')
plt.show()
```



Quantity by Profit
25000
20000
15000
10000
5000
0
Profit
1
2
3
4
5
6
7
8
9
10
11
12
13
14
Quantity



The majority of customers opt to purchase three quantities of an item, resulting in higher sales and profit. In contrast, when 14 quantities of an item are selected, both sales and profit tend to be lower.

8.Which shipping mode is most commonly chosen by customers?

```
store.groupby('Ship Mode')[['Customer Name']].value_counts()   # Most
of the customers are choosen first class shipping mode

Ship Mode       Customer Name
First Class     Dean percer             11
                Dave Hallsten           11
                Clytie Kelty            11
                Matt Connell            10
                Sam Zeldin               9
                                        ..
Standard Class  Carol Adams              1
                Cathy Prescott           1
                Charlotte Melton         1
                David Philippe           1
                Denise Leinenbach        1
Length: 1962, dtype: int64
```

Most of the customers are choosen First Class Shipping Mode

```
store.columns

Index(['Row ID', 'Order ID', 'Ship Mode', 'Customer ID', 'Customer
Name',
       'Segment', 'Country', 'City', 'State', 'Postal Code', 'Region',
       'Product ID', 'Category', 'Sub-Category', 'Product Name',
'Sales',
       'Quantity', 'Discount', 'Profit', 'Order_Date', 'Order_Month',
       'Order_Year', 'Ship_Date', 'Ship_Month', 'Ship_Year'],
      dtype='object')
```

9.How do sales trends vary across different regions, country and categories?

```
Sales_by_Region_Categories = store.groupby(['Region',
'Country','Category'])['Sales',].mean().unstack()
Sales_by_Region_Categories

                                 Sales
Category                     Furniture Office Supplies  Technology
Region  Country
Central United States   166.736615       47.483814  161.627490
East    United States   154.699687       50.033972  101.706782
South   United States   137.930402       59.773431  137.101293
West    United States   208.674371       56.035114  189.750973
```

In the Furniture category, the majority of average sales occur in the West region, while in the Office Supplies category, the Central region experiences lowest average sales.

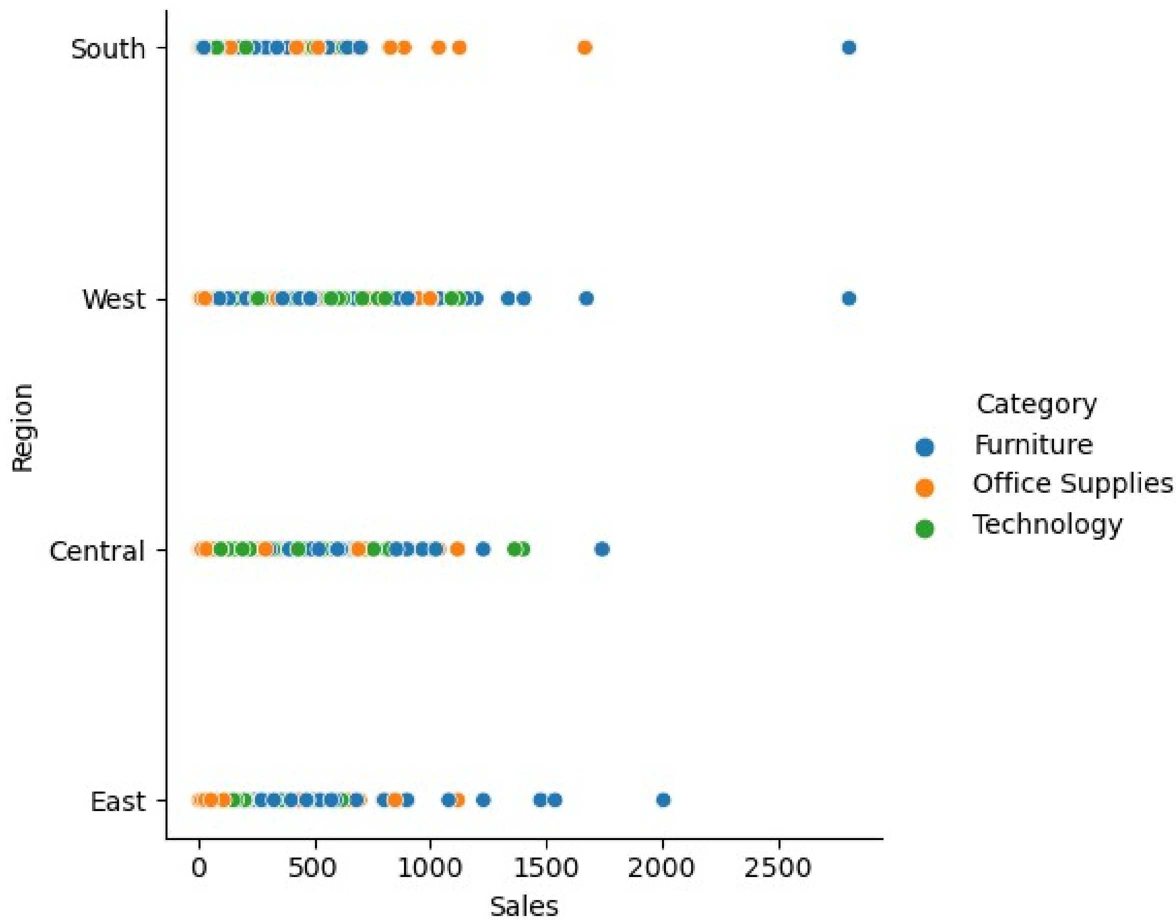
Suggestions:-

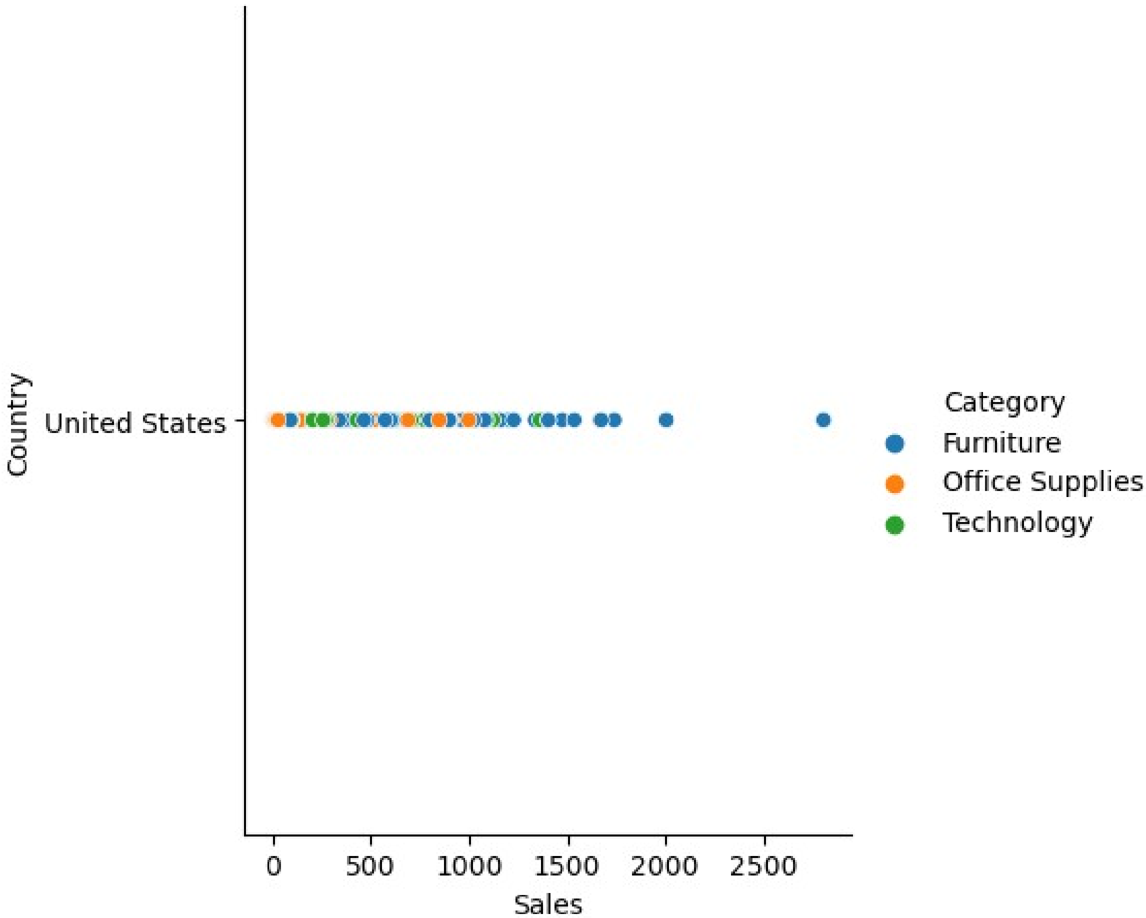
1.Marketing and Promotion:- Marketing and Advertising to increase brand visibility and attract more customers.

2.Customer Service:- Provide excellent customer service to build trust and loyalty.

3.Cost Reduction:- Identify and cut unnecessary costs without compromising quality.

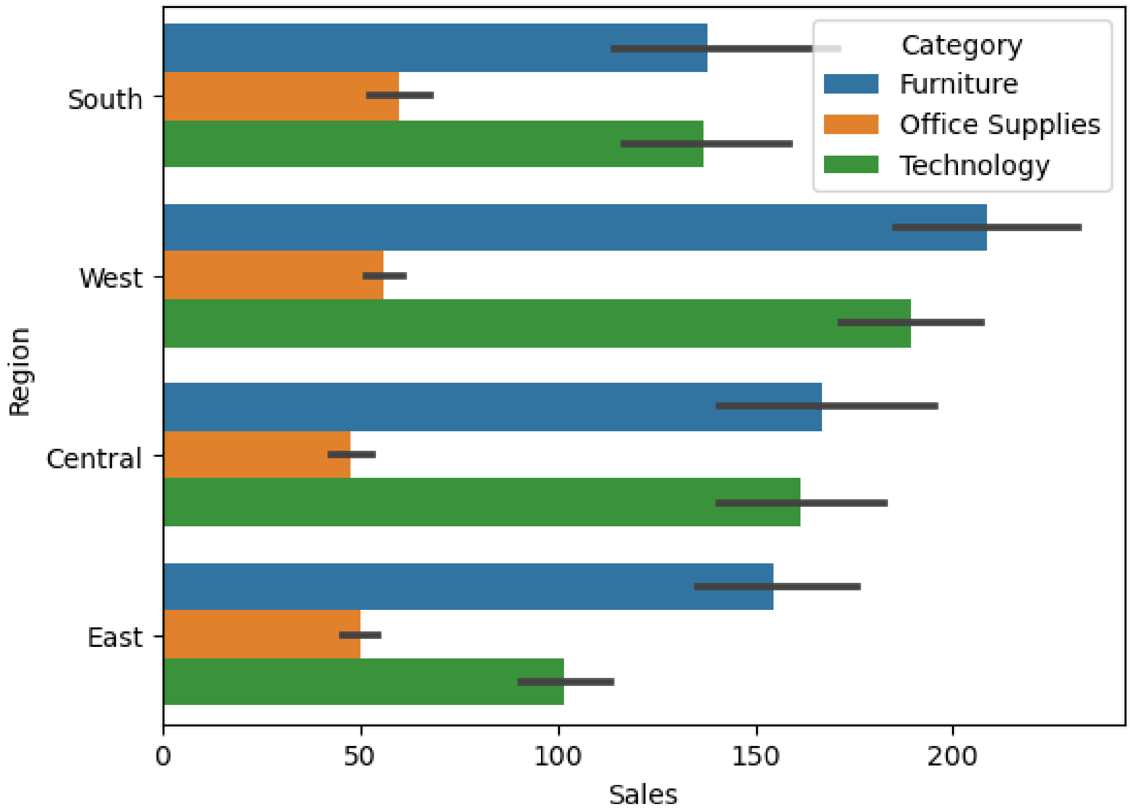
```
sns.relplot(data=store,x='Sales',y='Region',hue='Category')
sns.relplot(data=store,x='Sales',y='Country',hue='Category')

<seaborn.axisgrid.FacetGrid at 0x20316afd1e0>
```


South
West
Central
East
Region
0
500
1000
1500
2000
2500
Sales
Category
Furniture
Office Supplies
Technology



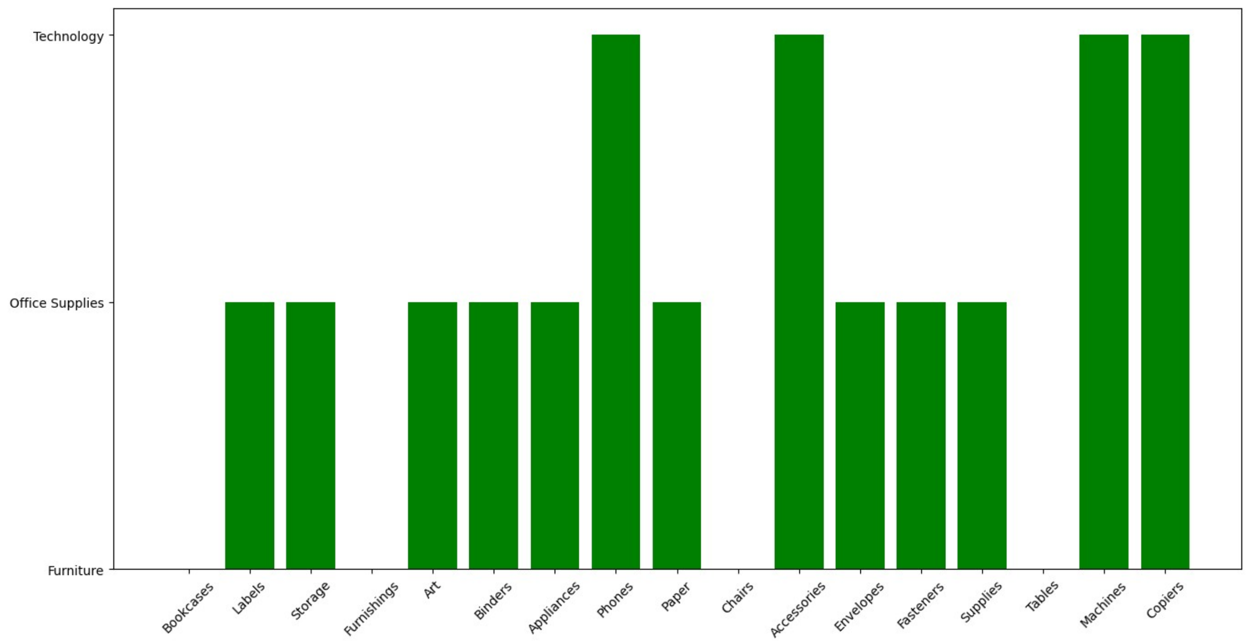

```
sns.barplot(data=store,x='Sales',y='Region',hue='Category',estimator='
mean')
```

```
<Axes: xlabel='Sales', ylabel='Region'>
```



10.How do Category trends vary across different Sub-Categories

```
#Lets see how sub-categories are distributed wrt to category
plt.figure(figsize=(16,8))
plt.bar('Sub-Category','Category',data=store,color='g')
plt.xticks(rotation = 45)
plt.show()
```



The above plot features contains:-

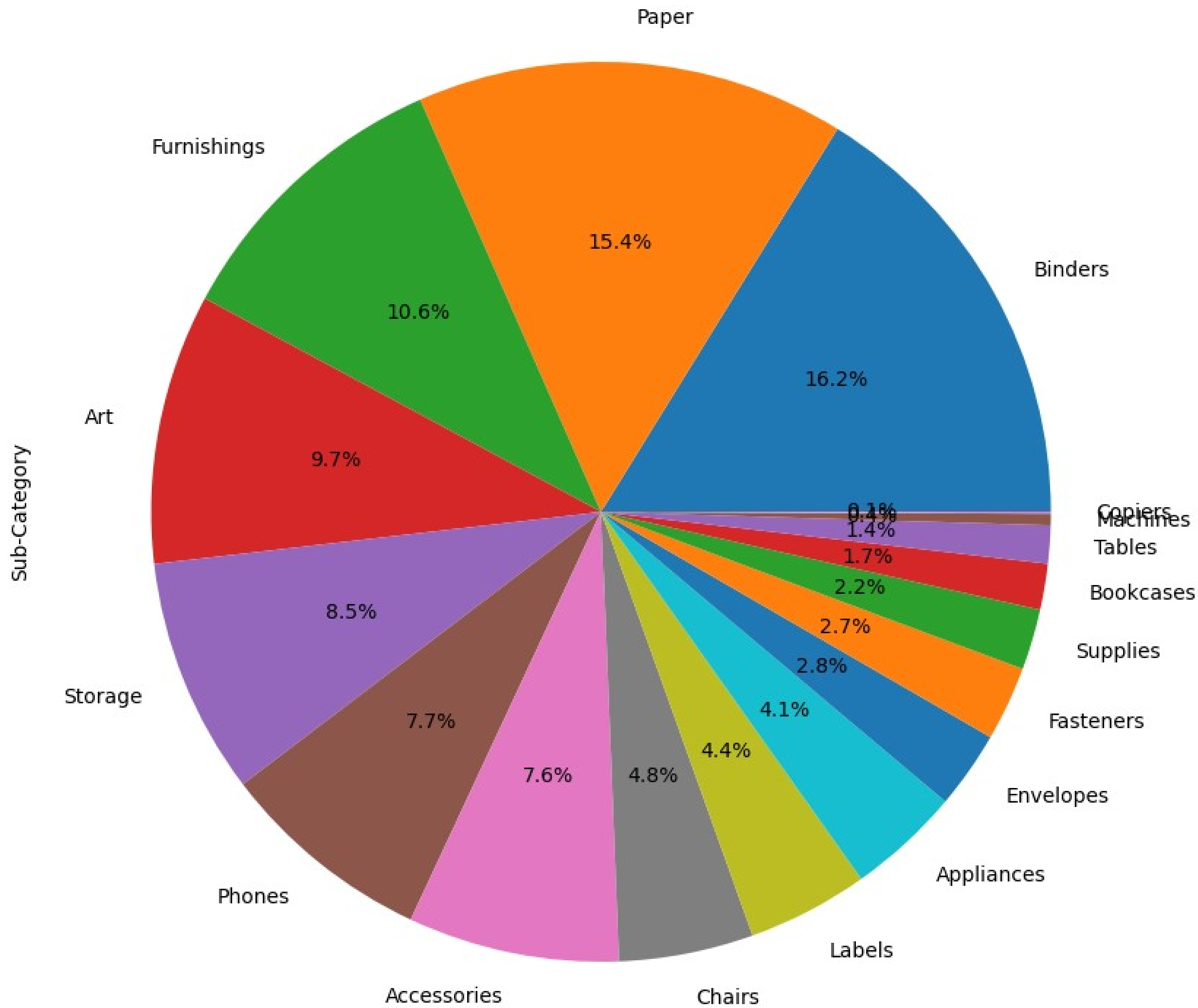
1.Technology - Phones,Accessories, Machines and Copiers

2.Office Supplies- Lables, Stronge, Art, Binders ,Appliances,Papers, Envelopes,Fasteners and Supplies

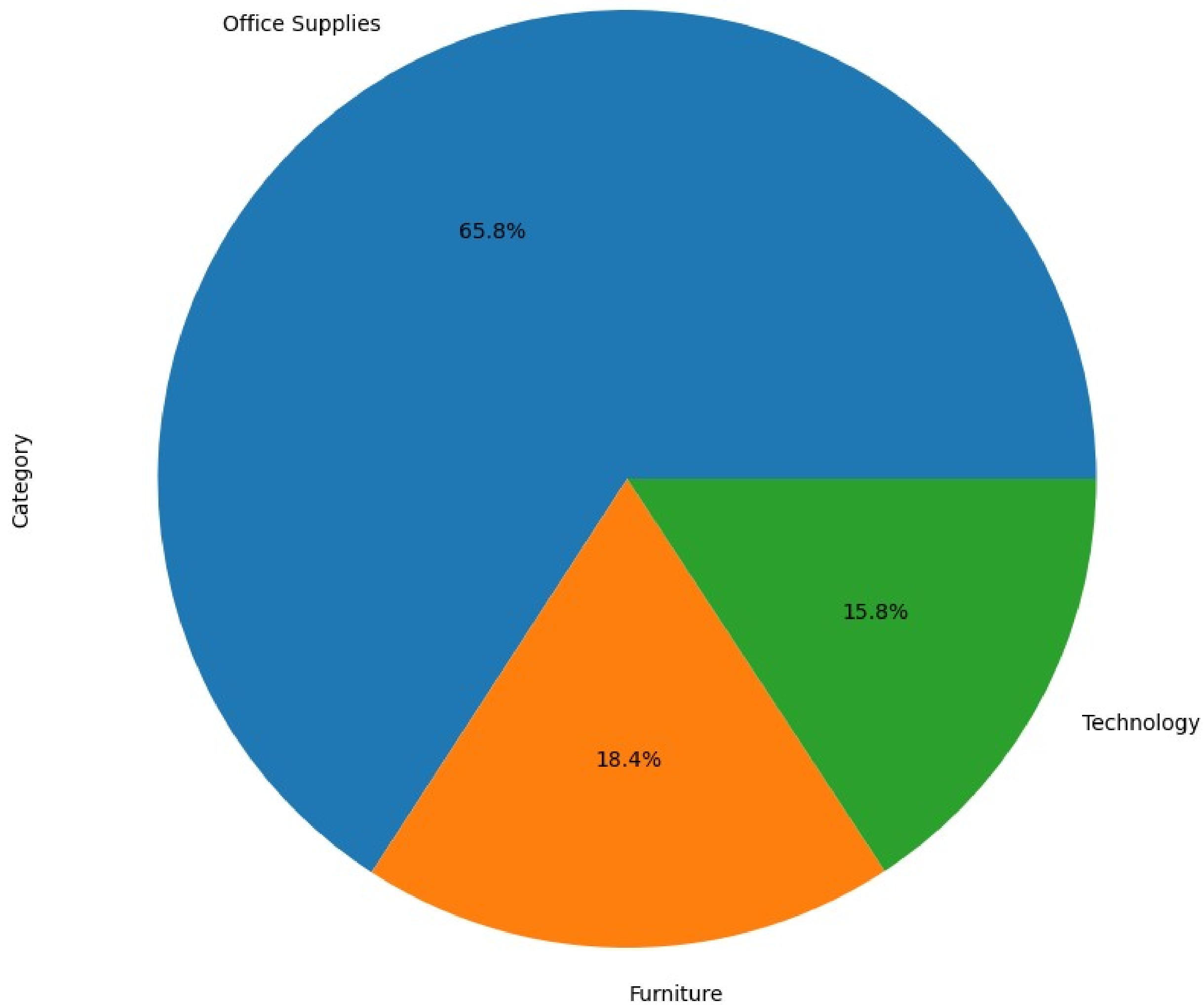
3.Furnitures- Bookcases, Furnishings, Chairs and Tables

11. How many unique categories and sub-categories? and can you provide insights into which categories or sub-categories are the most and least common?

```
plt.figure(figsize=(12,10))
store['Sub-Category'].value_counts().plot.pie(autopct="%1.1f%%")
plt.xticks(rotation = 45)
plt.show()

plt.figure(figsize=(12,10))
store['Category'].value_counts().plot.pie(autopct="%1.1f%%")
plt.xticks(rotation = 45)
plt.show()
```



Paper
Furnishings
Binders
Art
Sub-Category
Storage
Phones
Accessories
Chairs
Labels
Appliances
Envelopes
Fasteners
Supplies
Bookcases
Tables
Machines
Copiers
16.2%
15.4%
10.6%
9.7%
8.5%
7.7%
7.6%
4.8%
4.4%
4.1%
2.8%
2.7%
2.2%
1.7%
1.4%
0.4%
0.1%


Office Supplies
65.8%
Category
15.8%
Technology
18.4%
Furniture

In the pie chart mentioned above, the Sub-Category "Binders" accounts for the highest sales at 16.2%, while the Sub-Category "Copiers" represents the lowest sales.

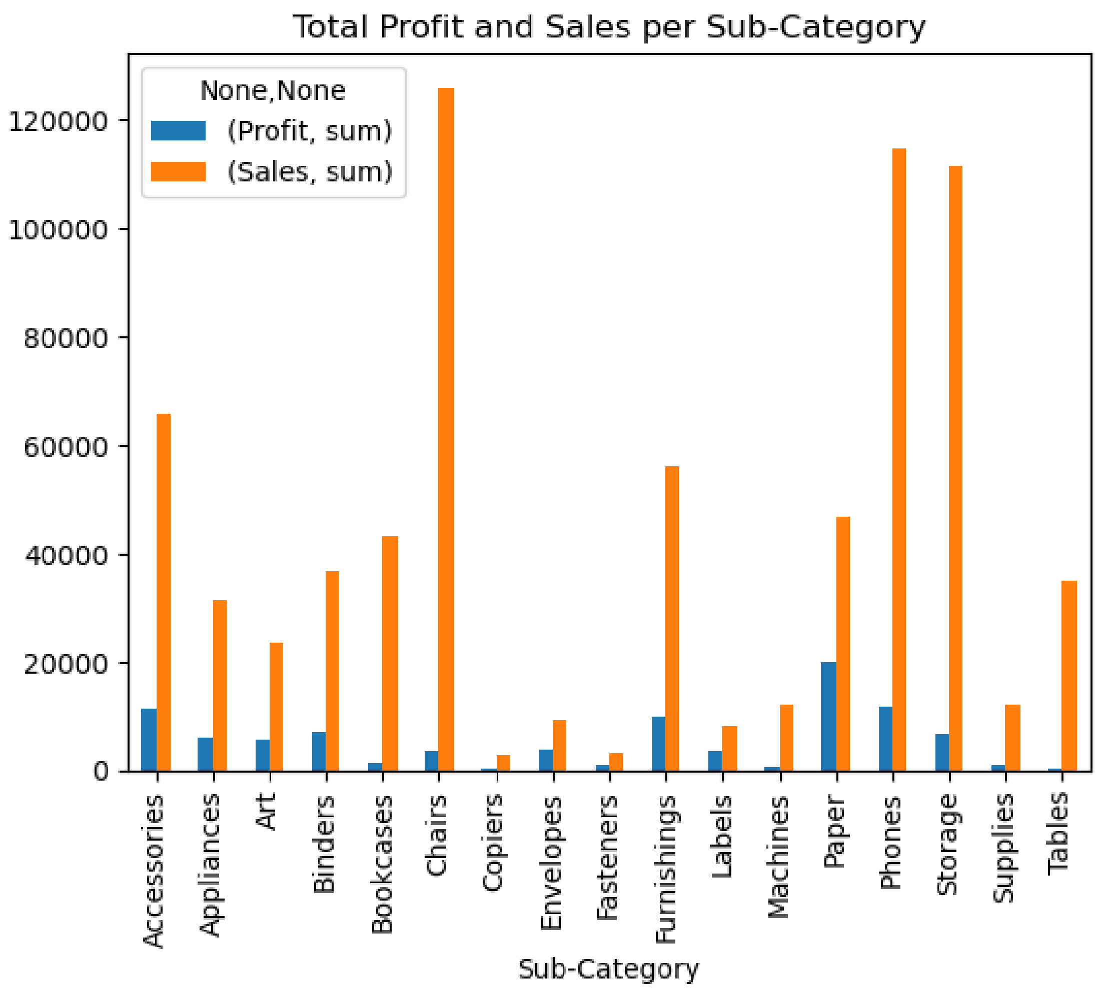
In Category "Office Suppilers" accounts for the highest sales at 65.8%, while the Category "Technology" represents the lowest sales.

Suggestions:-

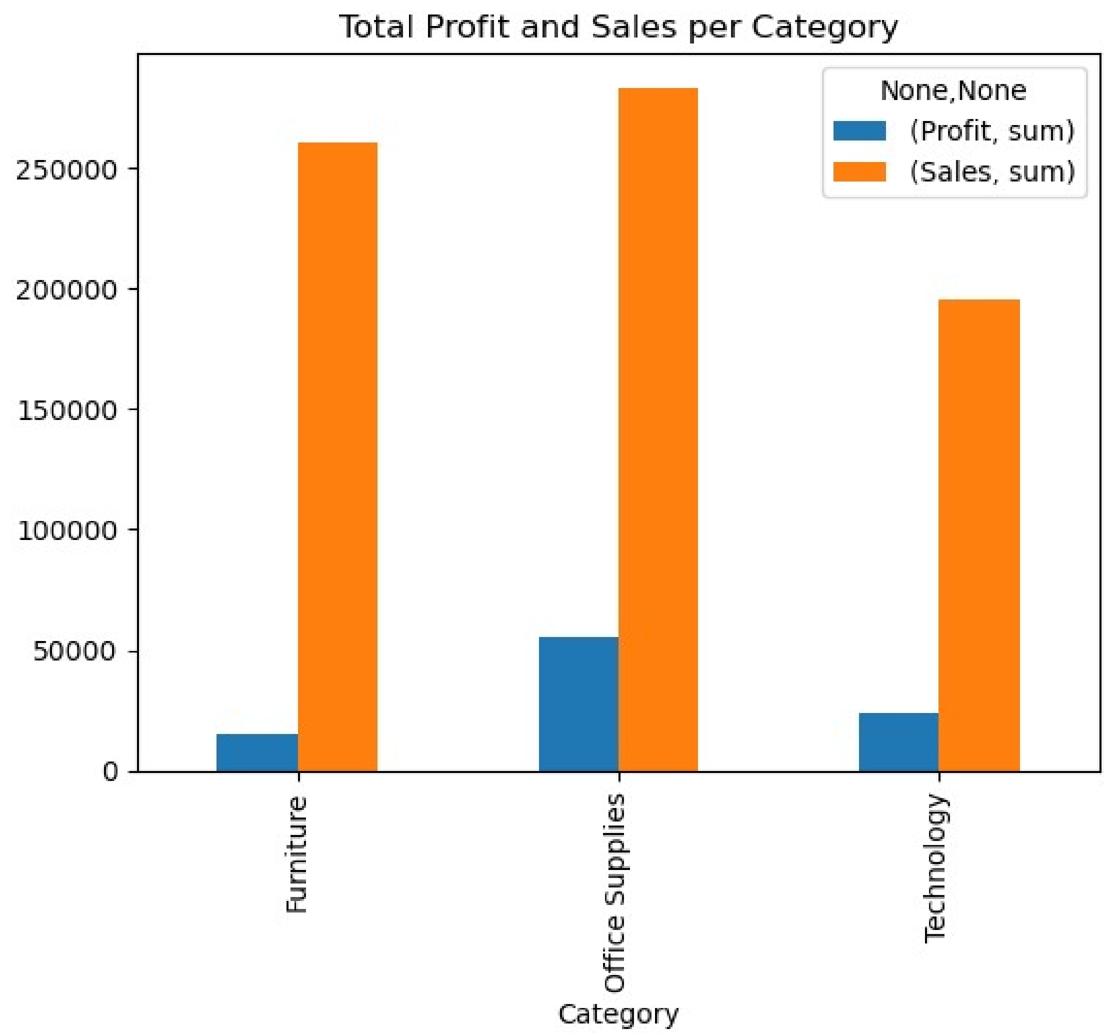
1.Market Expansion:Explore new markets and segments for potential growth, especially in the "Technology" category, which represents the lowest sales. Consider reaching out to different customer demographics.

12.Which sub-category generates the most sales and profit, and how does it outperform the others?

```
store.groupby('Sub-Category')
['Profit','Sales'].agg(['sum']).plot.bar()
plt.title('Total Profit and Sales per Sub-Category')
plt.show()

store.groupby('Category')['Profit','Sales'].agg(['sum']).plot.bar()
plt.title('Total Profit and Sales per Category')
plt.show()
```



Total Profit and Sales per Sub-Category
None,None
(Profit, sum)
(Sales, sum)
120000
100000
80000
60000
40000
20000
0
Accessories
Appliances
Art
Binders
Bookcases
Chairs
Copiers
Envelopes
Fasteners
Furnishings
Labels
Machines
Paper
Phones
Storage
Supplies
Tables
Sub-Category



In the Sub-Category, "Chairs" consistently leads with the highest sales, while "Copiers" and "Tables" generate comparatively lower profits.

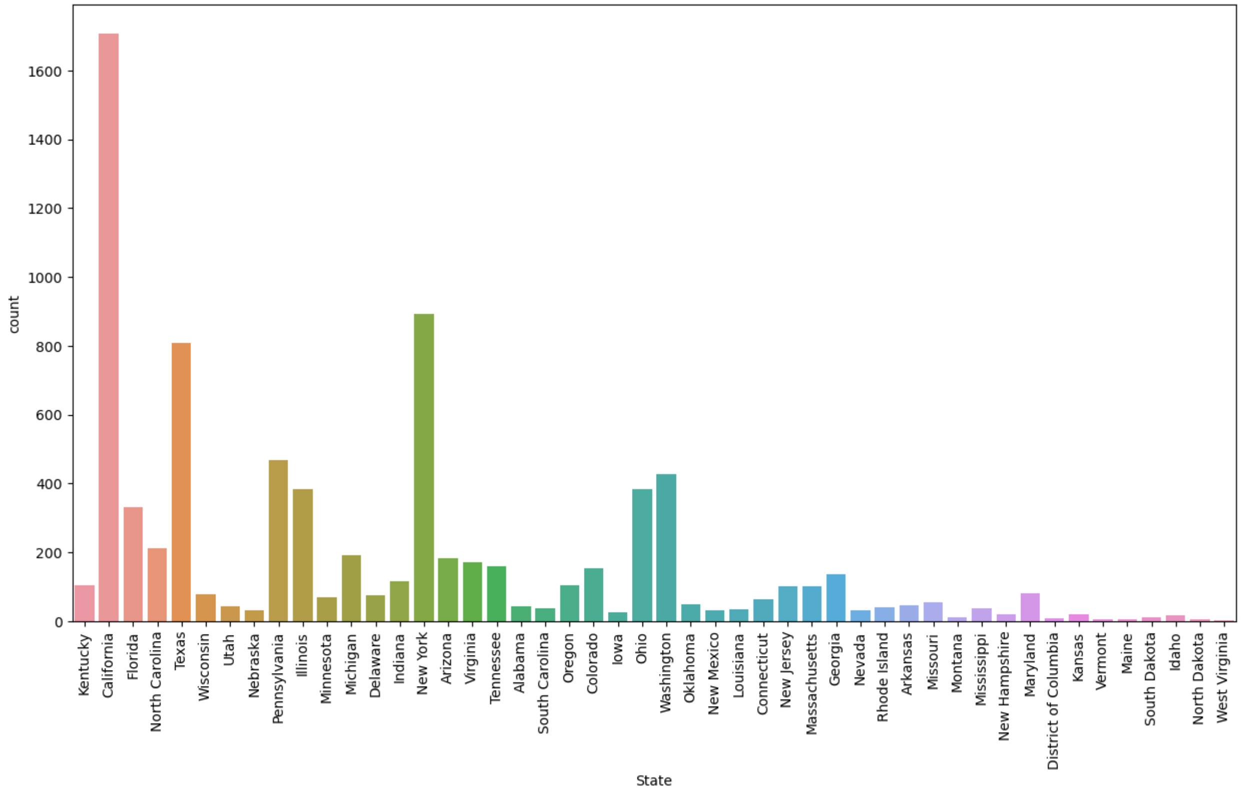
At the Category level, "Office Supplies" stands out as the top performer in terms of sales, while "Furniture" lags behind, resulting in lower profitability.

13.How many unique states are present in our dataset, and are there any states that appear more frequently than others?

Count Plot of States

```
plt.figure(figsize=(15,8))
sns.countplot(x=store['State'])
plt.xticks(rotation=90)
```

```
plt.show()
print(store.State.value_counts())
```




```
California              1707
New York                 892
Texas                    808
Pennsylvania             469
Washington               426
Ohio                     384
Illinois                 382
Florida                  332
North Carolina           213
Michigan                 192
Arizona                  182
Virginia                 170
Tennessee                160
Colorado                 153
Georgia                  137
Indiana                  116
Oregon                   105
Kentucky                 105
Massachusetts            101
New Jersey               100
Maryland                  82
```

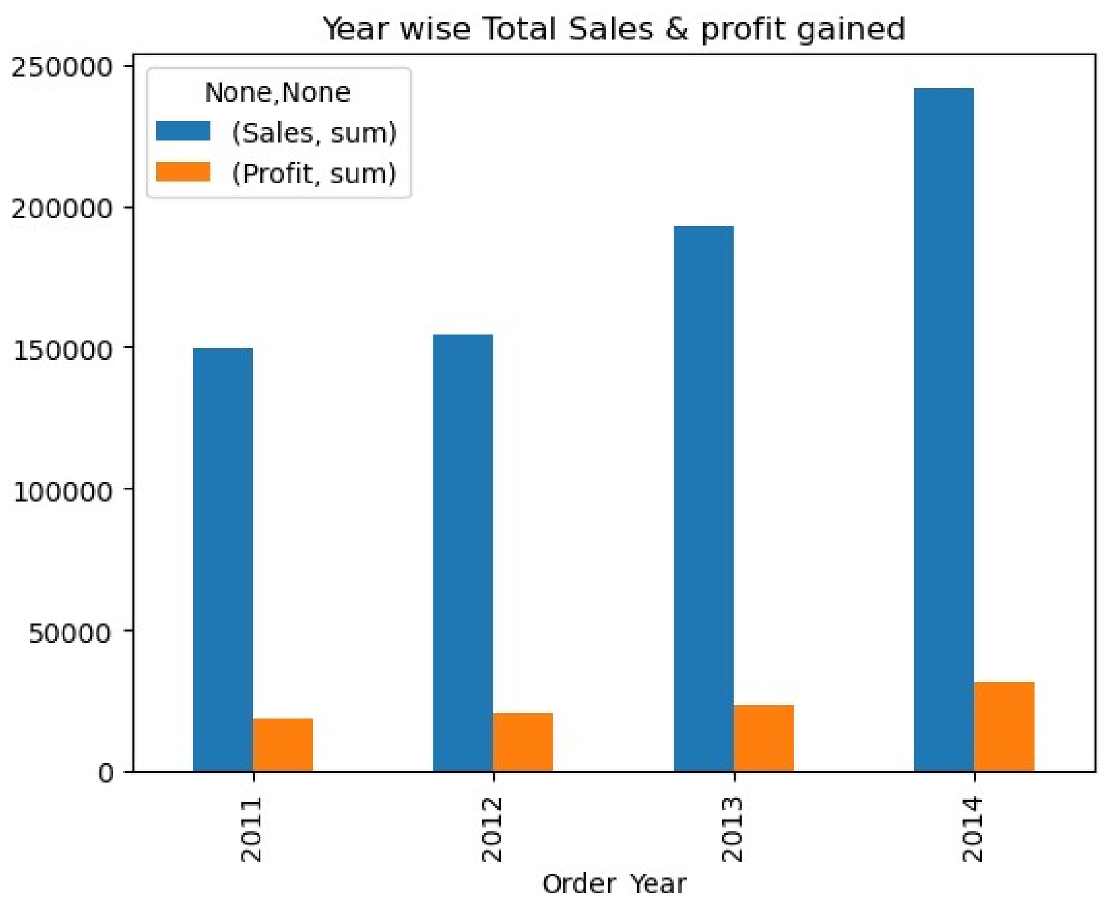
```
Wisconsin                  78
Delaware                   74
Minnesota                  70
Connecticut                64
Missouri                   55
Oklahoma                   48
Arkansas                   45
Alabama                    44
Utah                       43
Rhode Island               40
Mississippi                38
South Carolina             36
Louisiana                  35
Nebraska                   32
Nevada                     32
New Mexico                 30
Iowa                       25
New Hampshire              21
Kansas                     20
Idaho                      18
Montana                    12
South Dakota               10
District of Columbia        8
Vermont                     6
North Dakota                6
Maine                       5
West Virginia               2
Name: State, dtype: int64
```

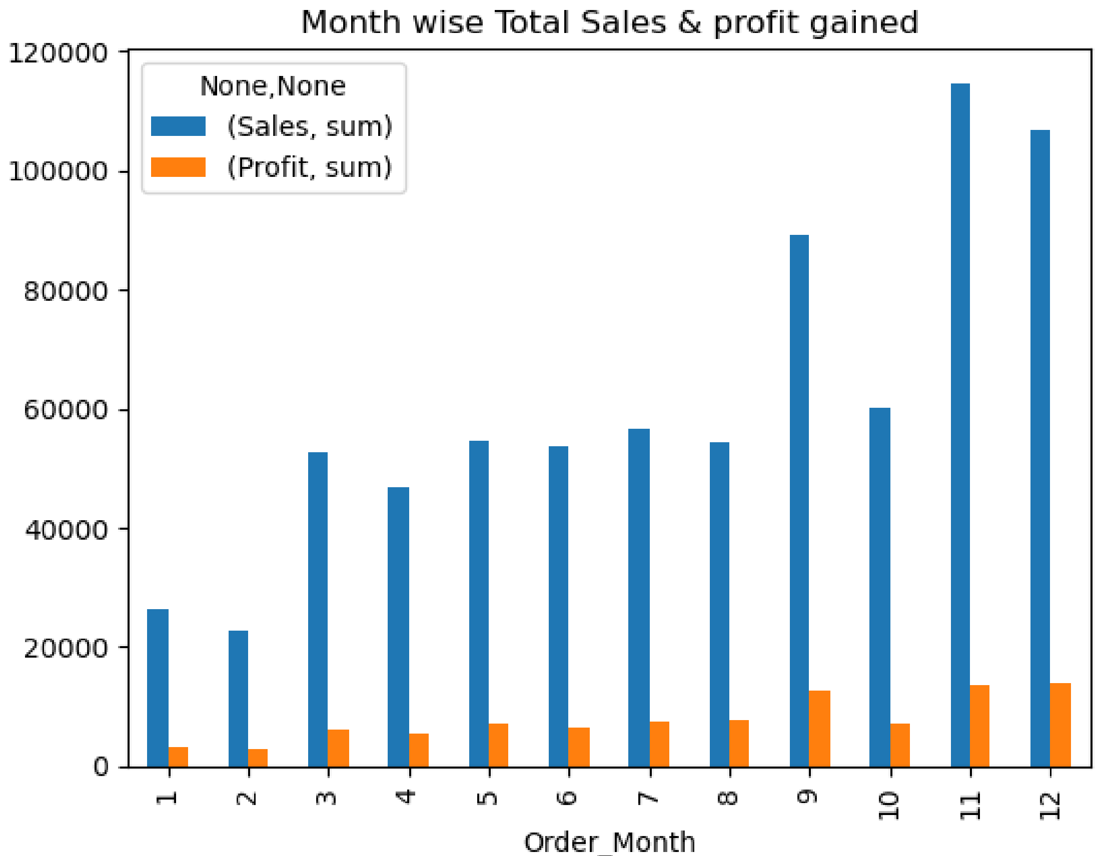
14.Top 10 customers who order frequently

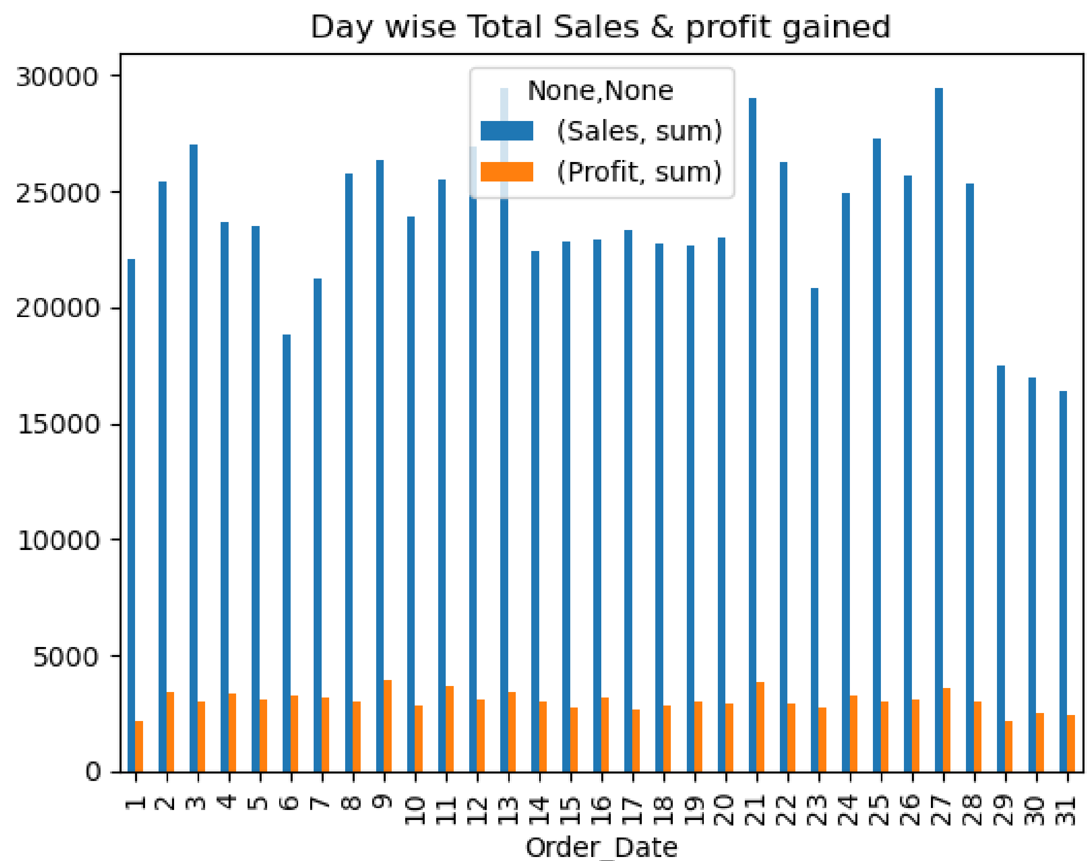
```
store_top10=store['Customer Name'].value_counts().head(10)
store_top10

William Brown           31
Chloris Kastensmidt     30
Matt Abelman            29
Arthur Prichep          28
Paul Prost              27
Sally Hughsby           27
Jonathan Doherty        26
Lena Cacioppo           26
Xylona Preis            26
Chris Selesnick         26
Name: Customer Name, dtype: int64

#Sales per year
store.groupby('Order_Year')['Sales','Profit'].agg(['sum']).plot.bar()
plt.title('Year wise Total Sales & profit gained')
```

```
#Sales per month
store.groupby('Order_Month')['Sales','Profit'].agg(['sum']).plot.bar()
plt.title('Month wise Total Sales & profit gained')

#Sales per date
store.groupby('Order_Date')['Sales','Profit'].agg(['sum']).plot.bar()
plt.title('Day wise Total Sales & profit gained')
```

```
Text(0.5, 1.0, 'Day wise Total Sales & profit gained')
```




Month wise Total Sales & profit gained
None,None
(Sales, sum)
(Profit, sum)
0
20000
40000
60000
80000
100000
120000
1
2
3
4
5
6
7
8
9
10
11
12
Order_Month



1.In terms of order_year, 2014 stands out with the highest sales and profitability, while 2011 records the lowest sales and profits. Year by year, both sales and profits show an increasing trend.

2.When considering order_month, the 11th month consistently achieves the highest sales and profit, with the 2nd month recording the lowest sales and profit. Some minor fluctuations can be observed on a monthly basis.

Analyzing sales and profit based on order_date, it's evident that the 27th and 13th of each month consistently yield the highest sales, while the 9th and 21st exhibit the lowest profit and sales. Furthermore, the 31st day of the month consistently has the lowest sales, with some minor fluctuations seen on a monthly basis.

Overall EDA Conclusion:

-> After a comprehensive analysis of the Super Store Profit Analysis Dataset, several key insights and trends have been identified:

-> The main reason which leads to loss is as if some areas lead to loss due to more discounts, and some areas lead to fewer sales due to fewer discounts, hence it needs to be improved.

-> It is better to give more discounts during festival seasons, additionally, that will result in more sales and profit.

-> Some cities have fewer sales, lack of awareness can be the reason for this, hence advertising in those cities might help in more sales.

1.The main reason which leads High sales might be due to a wide variety of products. Consider focusing on higher-margin items.)

2.Maintain quality and customer satisfaction to sustain high-profit margins) Office Supplies have both the lowest sales and the lowest profit. Assess the demand for office supplies and potentially lookup for more product offerings.)

```
pd.groupby(store.segment)['Profit'].mean()



# Statistical test

# t-test
from scipy.stats import ttest_ind
# Anova test
import statsmodels.api as sm
from statsmodels.formula.api import ols
# Tukey HSD
from statsmodels.stats.multicomp import pairwise_tukeyhsd
# chi-square
from scipy.stats import (chi2,chi2_contingency)

import statsmodels.formula.api as smf

store.columns
```

```
Index(['Row ID', 'Order ID', 'Ship Mode', 'Customer ID', 'Customer
Name',
       'Segment', 'Country', 'City', 'State', 'Postal Code', 'Region',
       'Product ID', 'Category', 'Sub-Category', 'Product Name',
'Sales',
       'Quantity', 'Discount', 'Profit', 'Order_Date', 'Order_Month',
       'Order_Year', 'Ship_Date', 'Ship_Month', 'Ship_Year'],
      dtype='object')
```

```
store.dtypes
```

```
Row ID                int64
Order ID             object
Ship Mode            object
Customer ID          object
Customer Name        object
Segment              object
Country              object
City                 object
```

```
State             object
Postal Code        int64
Region            object
Product ID        object
Category          object
Sub-Category      object
Product Name      object
Sales            float64
Quantity           int64
Discount         float64
Profit           float64
Order_Date         int64
Order_Month        int64
Order_Year         int64
Ship_Date          int64
Ship_Month         int64
Ship_Year          int64
dtype: object
```

```
np.corrcoef(store.Profit,store.Sales)
```

```
array([[1.        , 0.32230917],
       [0.32230917, 1.        ]])
```

Slightly 32% Correlated

```
np.corrcoef(store.Profit,store.Discount)
```

```
array([[ 1.        , -0.43141955],
       [-0.43141955,  1.        ]])
```

Negetively 43% Correlated

```
np.corrcoef(store.Profit,store.Quantity)
```

```
array([[1.        , 0.18288339],
       [0.18288339, 1.        ]])
```

Slightly 18% Correlated

```
store['Sub-Category'].value_counts()
```

```
Binders        1314
Paper          1248
Furnishings     857
Art             787
Storage         690
Phones          625
Accessories     615
```

```
Chairs            392
Labels            353
Appliances        329
Envelopes         226
Fasteners         217
Supplies          177
Bookcases         134
Tables            111
Machines           33
Copiers             5
Name: Sub-Category, dtype: int64
```

```
store.Category.value_counts()
```

```
Office Supplies    5341
Furniture          1494
Technology         1278
Name: Category, dtype: int64
```

Highest profit is earned in Copiers while Selling price for Chairs and Phones is extremely high compared to other products.

Another interesting fact- people dont prefer to buy Tables and Bookcases from Superstore. Hence these departments are in loss.

Now will perform some statistical test to check Company is a good predictor or not

Annova Test

```
store.columns
```

```
Index(['Row ID', 'Order ID', 'Ship Mode', 'Customer ID', 'Customer Name',
       'Segment', 'Country', 'City', 'State', 'Postal Code', 'Region',
       'Product ID', 'Category', 'Sub-Category', 'Product Name', 'Sales',
       'Quantity', 'Discount', 'Profit', 'Order_Date', 'Order_Month',
       'Order_Year', 'Ship_Date', 'Ship_Month', 'Ship_Year'],
      dtype='object')
```

```
store.dtypes
```

```
Row ID              int64
Order ID           object
Ship Mode          object
Customer ID        object
Customer Name      object
Segment            object
Country            object
```

```
City                 object
State                object
Postal Code           int64
Region               object
Product ID           object
Category             object
Sub-Category         object
Product Name         object
Sales               float64
Quantity              int64
Discount            float64
Profit              float64
Order_Date            int64
Order_Month           int64
Order_Year            int64
Ship_Date             int64
Ship_Month            int64
Ship_Year             int64
dtype: object

# Statistical test

# t-test
from scipy.stats import ttest_ind
# Anova test
import statsmodels.api as sm
from statsmodels.formula.api import ols
# Tukey HSD
from statsmodels.stats.multicomp import pairwise_tukeyhsd
# chi-square
from scipy.stats import (chi2,chi2_contingency)

import statsmodels.formula.api as smf

model_Segment = ols('Profit ~Segment',data = store).fit()
anova_Segment = sm.stats.anova_lm(model_Segment)
anova_Segment
```

	df	sum_sq	mean_sq	F	PR(>F)
Segment	2.0	3.217117e+02	160.855856	0.46283	0.629517
Residual	8110.0	2.818620e+06	347.548747	NaN	NaN

```
model_Segment = ols('Profit ~City',data = store).fit()
anova_Segment = sm.stats.anova_lm(model_Segment)
anova_Segment
```

	df	sum_sq	mean_sq	F	PR(>F)
City	516.0	4.615345e+05	894.446617	2.882071	3.999950e-83
Residual	7596.0	2.357408e+06	310.348551	NaN	NaN

City is good Predictor

```
model_State = ols('Profit ~ State',data = store).fit()
anova_State = sm.stats.anova_lm(model_State)
anova_State
```

	df	sum_sq	mean_sq	F	PR(>F)
State	47.0	3.843815e+05	8178.330813	27.092462	1.733605e-217
Residual	8065.0	2.434561e+06	301.867390	NaN	NaN

State is good Predictor

```
model_Region = ols('Profit ~ Region ',data = store).fit()
anova_Region = sm.stats.anova_lm(model_Region)
anova_Region
```

	df	sum_sq	mean_sq	F	PR(>F)
Region	3.0	4.265007e+04	14216.689254	41.524139	1.273337e-26
Residual	8109.0	2.776292e+06	342.371684	NaN	NaN

Region is good Predictor

```
model_Region = ols('Profit ~ Category ',data = store).fit()
anova_Region = sm.stats.anova_lm(model_Region)
anova_Region
```

	df	sum_sq	mean_sq	F	PR(>F)
Category	2.0	8.246112e+04	41230.558841	122.193372	5.199465e-53
Residual	8110.0	2.736481e+06	337.420584	NaN	NaN

```
model_Sales = ols('Profit ~ Sales ',data = store).fit()
anova_Sales = sm.stats.anova_lm(model_Sales)
anova_Sales
```

	df	sum_sq	mean_sq	F	PR(>F)
Sales	1.0	2.928407e+05	292840.731048	940.275495	1.792103e-195
Residual	8111.0	2.526101e+06	311.441415	NaN	NaN

Sales is good Predictor

```
model_Quantity = ols('Profit ~ Quantity',data = store).fit()
anova_Quantity = sm.stats.anova_lm(model_Quantity)
anova_Quantity
```

	df	sum_sq	mean_sq	F	PR(>F)
Quantity	1.0	9.428328e+04	94283.280258	280.670627	5.856133e-62
Residual	8111.0	2.724659e+06	335.921436	NaN	NaN

Quantity is good Predictor

```
model_Discount = ols('Profit ~ Discount',data = store).fit()
anova_Discount = sm.stats.anova_lm(model_Discount)
anova_Discount
```

```
              df        sum_sq        mean_sq            F  PR(>F)
Discount     1.0  5.246695e+05  524669.474604  1854.877294     0.0
Residual  8111.0  2.294273e+06     282.859398          NaN     NaN
```

Discount is very good Predictor

Category is not good Predictor

```
store.columns
```

```
Index(['Row ID', 'Order ID', 'Ship Mode', 'Customer ID', 'Customer
Name',
       'Segment', 'Country', 'City', 'State', 'Postal Code', 'Region',
       'Product ID', 'Category', 'Sub-Category', 'Product Name',
'Sales',
       'Quantity', 'Discount', 'Profit', 'Order_Date', 'Order_Month',
       'Order_Year', 'Ship_Date', 'Ship_Month', 'Ship_Year'],
      dtype='object')
```

```
store.Region.unique()
```

```
array(['South', 'West', 'Central', 'East'], dtype=object)
```

```
store.replace({'West':0, 'South':1, 'Central':2,
'East':3},inplace=True)
```

```
store.Region.unique()
```

```
array([1, 0, 2, 3], dtype=int64)
```

```
store.Segment.unique()
```

```
array(['Consumer', 'Corporate', 'Home Office'], dtype=object)
```

```
store.replace({'Corporate':0, 'Consumer':1, 'Home
Office':2},inplace=True)
```

```
store.head()
```

```
   Row ID        Order ID       Ship Mode Customer ID    Customer Name
\
0       1  CA-2013-152156    Second Class    CG-12520      Claire Gute

2       3  CA-2013-138688    Second Class    DV-13045  Darrin Van Huff

4       5  US-2012-108966  Standard Class    SO-20335   Sean O'Donnell

5       6  CA-2011-115812  Standard Class    BH-11710  Brosina Hoffman
```

```
6       7  CA-2011-115812  Standard Class    BH-11710  Brosina Hoffman


   Segment        Country             City       State  Postal
Code  ...  \
0        1  United States        Henderson    Kentucky
42420  ...
2        0  United States      Los Angeles  California
90036  ...
4        1  United States  Fort Lauderdale     Florida
33311  ...
5        1  United States      Los Angeles  California
90032  ...
6        1  United States      Los Angeles  California
90032  ...

     Sales Quantity Discount   Profit Order_Date  Order_Month
Order_Year  \
0  261.960        2      0.0  41.9136          9           11
2013
2   14.620        2      0.0   6.8714         13            6
2013
4   22.368        2      0.2   2.5164         11           10
2012
5   48.860        7      0.0  14.1694          9            6
2011
6    7.280        4      0.0   1.9656          9            6
2011

   Ship_Date  Ship_Month  Ship_Year
0         12          11       2013
2         17           6       2013
4         18          10       2012
5         14           6       2011
6         14           6       2011

[5 rows x 25 columns]
```

```
store['Ship Mode'].unique()
```

```
array(['Second Class', 'Standard Class', 'First Class', 'Same Day'],
      dtype=object)
```

```
store.columns
```

```
Index(['Row ID', 'Order ID', 'Ship Mode', 'Customer ID', 'Customer
Name',
       'Segment', 'Country', 'City', 'State', 'Postal Code', 'Region',
       'Product ID', 'Category', 'Sub-Category', 'Product Name',
'Sales',
```

```
       'Quantity', 'Discount', 'Profit', 'Order_Date', 'Order_Month',
       'Order_Year', 'Ship_Date', 'Ship_Month', 'Ship_Year'],
      dtype='object')
```

```
store.replace({'Second Class':1, 'Standard Class':2, 'First Class':0,
'Same Day':3},inplace=True)

store.drop(['Row ID','Order ID','Customer ID','Customer Name','Postal
Code','Product ID','Product Name'],axis=1,inplace=True)

print(len(store.City.value_counts()))
print(len(store.State.unique()))
```

```
517
48
```

```
from sklearn.preprocessing import LabelEncoder

# Initialize the LabelEncoder
le = LabelEncoder()

# Fit and transform the 'Category' column
store['City'] = le.fit_transform(store['City'])

store['State'] = le.fit_transform(store['State'])

store['Category'] = le.fit_transform(store['Category'])
store['Sub-Category'] = le.fit_transform(store['Sub-Category'])

store.drop('Country',axis=1,inplace=True)

store.head()
```

```
   Ship Mode  Segment  City  State  Region  Category  Sub-Category
Sales  \
0          1        1   190     15       1         0             4
261.960
2          1        0   259      3       0         1            10
14.620
4          2        1   149      8       1         1            14
22.368
5          2        1   259      3       0         0             9
48.860
6          2        1   259      3       0         1             2
7.280

   Quantity  Discount   Profit  Order_Date  Order_Month  Order_Year  \
0         2       0.0  41.9136           9           11        2013
2         2       0.0   6.8714          13            6        2013
4         2       0.2   2.5164          11           10        2012
5         7       0.0  14.1694           9            6        2011
6         4       0.0   1.9656           9            6        2011
```

```
   Ship_Date  Ship_Month  Ship_Year
0         12          11       2013
2         17           6       2013
4         18          10       2012
5         14           6       2011
6         14           6       2011
```

```
store.isna().sum()
```

```
Ship Mode       0
Segment         0
City            0
State           0
Region          0
Category        0
Sub-Category    0
Sales           0
Quantity        0
Discount        0
Profit          0
Order_Date      0
Order_Month     0
Order_Year      0
Ship_Date       0
Ship_Month      0
Ship_Year       0
dtype: int64
```

```
store.columns
```

```
Index(['Ship Mode', 'Segment', 'City', 'State', 'Region', 'Category',
       'Sub-Category', 'Sales', 'Quantity', 'Discount', 'Profit',
'Order_Date',
       'Order_Month', 'Order_Year', 'Ship_Date', 'Ship_Month',
'Ship_Year'],
      dtype='object')
```

```
store.describe()
```

```
         Ship Mode      Segment         City        State       Region
\
count  8113.000000  8113.000000  8113.000000  8113.000000  8113.000000

mean      1.549858     0.877481   273.591397    22.178232     1.447430

std       0.815445     0.679930   134.834637    15.593029     1.212089

min       0.000000     0.000000     0.000000     0.000000     0.000000

25%       1.000000     0.000000   164.000000     4.000000     0.000000
```

```
50%        2.000000     1.000000   288.000000    26.000000     2.000000

75%        2.000000     1.000000   390.000000    36.000000     3.000000

max        3.000000     2.000000   516.000000    47.000000     3.000000


          Category  Sub-Category        Sales     Quantity
Discount  \
count  8113.000000   8113.000000  8113.000000  8113.000000
8113.000000
mean      0.973376      7.455565    91.051345     3.550475
0.148920
std       0.583958      4.952340   156.189857     2.076080
0.197695
min       0.000000      0.000000     0.444000     1.000000
0.000000
25%       1.000000      3.000000    13.970000     2.000000
0.000000
50%       1.000000      9.000000    35.440000     3.000000
0.200000
75%       1.000000     12.000000    99.870000     5.000000
0.200000
max       2.000000     16.000000  2803.920000    14.000000
0.800000

            Profit   Order_Date  Order_Month   Order_Year    Ship_Date
\
count  8113.000000  8113.000000  8113.000000  8113.000000  8113.000000

mean     11.604086    15.651917     7.798841  2012.727105    15.893258

std      18.641425     8.712501     3.282584     1.124620     8.787287

min     -39.637000     1.000000     1.000000  2011.000000     1.000000

25%       2.049200     8.000000     5.000000  2012.000000     8.000000

50%       7.257600    16.000000     9.000000  2013.000000    16.000000

75%      19.034400    23.000000    11.000000  2014.000000    24.000000

max      70.722000    31.000000    12.000000  2014.000000    31.000000


        Ship_Month    Ship_Year
count  8113.000000  8113.000000
mean      7.721188  2012.743745
std       3.340611     1.129456
min       1.000000  2011.000000
```

```
25%       5.000000  2012.000000
50%       8.000000  2013.000000
75%      11.000000  2014.000000
max      12.000000  2015.000000
```

```
store.columns
```

```
Index(['Ship Mode', 'Segment', 'City', 'State', 'Region', 'Category',
       'Sub-Category', 'Sales', 'Quantity', 'Discount', 'Profit',
'Order_Date',
       'Order_Month', 'Order_Year', 'Ship_Date', 'Ship_Month',
'Ship_Year'],
      dtype='object')
```

```
X=store.drop('Profit',axis=1)
```

```
y=store['Profit']
```

```
X.head()
```

```
   Ship Mode  Segment  City  State  Region  Category  Sub-Category
Sales  \
0          1        1   190     15       1         0             4
261.960
2          1        0   259      3       0         1            10
14.620
4          2        1   149      8       1         1            14
22.368
5          2        1   259      3       0         0             9
48.860
6          2        1   259      3       0         1             2
7.280

   Quantity  Discount  Order_Date  Order_Month  Order_Year  Ship_Date
\
0         2       0.0           9           11        2013         12

2         2       0.0          13            6        2013         17

4         2       0.2          11           10        2012         18

5         7       0.0           9            6        2011         14

6         4       0.0           9            6        2011         14


   Ship_Month  Ship_Year
0          11       2013
2           6       2013
4          10       2012
```

```
5            6       2011
6            6       2011
```

```
y.head()
```

```
0    41.9136
2     6.8714
4     2.5164
5    14.1694
6     1.9656
Name: Profit, dtype: float64
```

```
X.describe()
```

```
         Ship Mode      Segment         City        State       Region
\
count  8113.000000  8113.000000  8113.000000  8113.000000  8113.000000

mean      1.549858     0.877481   273.591397    22.178232     1.447430

std       0.815445     0.679930   134.834637    15.593029     1.212089

min       0.000000     0.000000     0.000000     0.000000     0.000000

25%       1.000000     0.000000   164.000000     4.000000     0.000000

50%       2.000000     1.000000   288.000000    26.000000     2.000000

75%       2.000000     1.000000   390.000000    36.000000     3.000000

max       3.000000     2.000000   516.000000    47.000000     3.000000


          Category  Sub-Category        Sales     Quantity
Discount  \
count  8113.000000   8113.000000  8113.000000  8113.000000
8113.000000
mean      0.973376      7.455565    91.051345     3.550475
0.148920
std       0.583958      4.952340   156.189857     2.076080
0.197695
min       0.000000      0.000000     0.444000     1.000000
0.000000
25%       1.000000      3.000000    13.970000     2.000000
0.000000
50%       1.000000      9.000000    35.440000     3.000000
0.200000
75%       1.000000     12.000000    99.870000     5.000000
0.200000
max       2.000000     16.000000  2803.920000    14.000000
0.800000
```

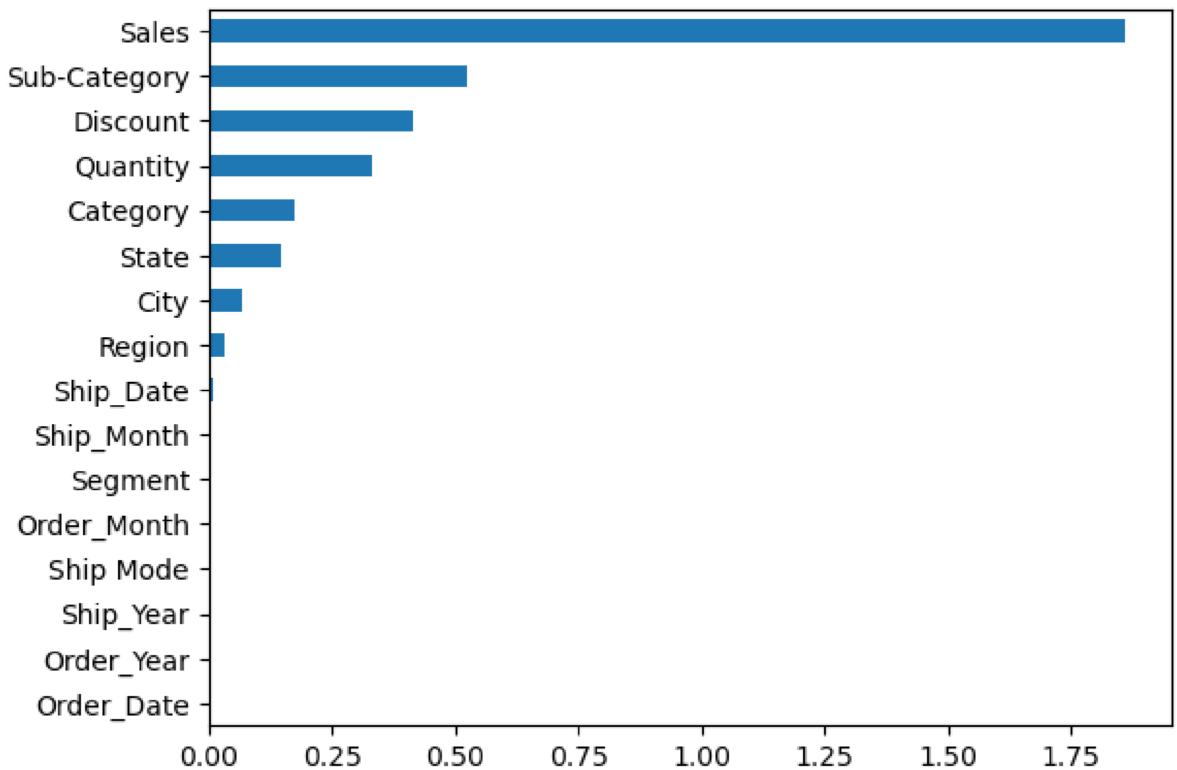
```
       Order_Date  Order_Month   Order_Year    Ship_Date   Ship_Month  \
count  8113.000000  8113.000000  8113.000000  8113.000000  8113.000000

mean     15.651917     7.798841  2012.727105    15.893258     7.721188

std       8.712501     3.282584     1.124620     8.787287     3.340611

min       1.000000     1.000000  2011.000000     1.000000     1.000000

25%       8.000000     5.000000  2012.000000     8.000000     5.000000

50%      16.000000     9.000000  2013.000000    16.000000     8.000000

75%      23.000000    11.000000  2014.000000    24.000000    11.000000

max      31.000000    12.000000  2014.000000    31.000000    12.000000


         Ship_Year
count  8113.000000
mean   2012.743745
std       1.129456
min    2011.000000
25%    2012.000000
50%    2013.000000
75%    2014.000000
max    2015.000000
```

```
# standardising the data
from sklearn.preprocessing import StandardScaler

sc=StandardScaler()
Scaled=sc.fit_transform(X)
X=pd.DataFrame(Scaled,columns=X.columns)

X.describe()
```

```
          Ship Mode       Segment          City         State
Region  \
count  8.113000e+03  8.113000e+03  8.113000e+03  8.113000e+03
8.113000e+03
mean   1.156066e-16 -7.707107e-17 -1.550180e-16  6.699928e-17
7.006461e-18
std    1.000062e+00  1.000062e+00  1.000062e+00  1.000062e+00
1.000062e+00
min   -1.900745e+00 -1.290625e+00 -2.029213e+00 -1.422405e+00 -
1.194235e+00
25%   -6.743459e-01 -1.290625e+00 -8.128338e-01 -1.165864e+00 -
1.194235e+00
```

```
50%    5.520536e-01  1.802053e-01  1.068679e-01  2.451097e-01
4.559103e-01
75%    5.520536e-01  1.802053e-01  8.633967e-01  8.864615e-01
1.280983e+00
max    1.778453e+00  1.651036e+00  1.797932e+00  1.591948e+00
1.280983e+00

           Category  Sub-Category         Sales      Quantity
Discount  \
count  8.113000e+03  8.113000e+03  8.113000e+03  8.113000e+03
8.113000e+03
mean  -8.057430e-17 -2.452261e-17 -6.174444e-17  8.101221e-18 -
6.305815e-17
std    1.000062e+00  1.000062e+00  1.000062e+00  1.000062e+00
1.000062e+00
min   -1.666962e+00 -1.505556e+00 -5.801461e-01 -1.228581e+00 -
7.533310e-01
25%    4.559501e-02 -8.997444e-01 -4.935410e-01 -7.468741e-01 -
7.533310e-01
50%    4.559501e-02  3.118789e-01 -3.560716e-01 -2.651673e-01
2.583931e-01
75%    4.559501e-02  9.176905e-01  5.646460e-02  6.982463e-01
2.583931e-01
max    1.758152e+00  1.725439e+00  1.737012e+01  5.033608e+00
3.293565e+00

         Order_Date   Order_Month    Order_Year     Ship_Date
Ship_Month  \
count  8.113000e+03  8.113000e+03  8.113000e+03  8.113000e+03
8.113000e+03
mean  -4.335248e-17 -6.305815e-17 -8.412658e-14 -4.685571e-17
3.152907e-17
std    1.000062e+00  1.000062e+00  1.000062e+00  1.000062e+00
1.000062e+00
min   -1.681816e+00 -2.071313e+00 -1.535818e+00 -1.694969e+00 -
2.012088e+00
25%   -8.783228e-01 -8.526861e-01 -6.465737e-01 -8.983141e-01 -
8.146283e-01
50%    3.995463e-02  3.659412e-01  2.426706e-01  1.214810e-02
8.346646e-02
75%    8.434474e-01  9.752548e-01  1.131915e+00  9.226103e-01
9.815612e-01
max    1.761725e+00  1.279912e+00  1.131915e+00  1.719265e+00
1.280926e+00

          Ship_Year
count  8.113000e+03
mean  -4.431061e-14
std    1.000062e+00
min   -1.543975e+00
```

```
25%    -6.585386e-01
50%     2.268979e-01
75%     1.112334e+00
max     1.997771e+00

# Information Gain
from sklearn.feature_selection import mutual_info_regression
import matplotlib.pyplot as plt

importances = mutual_info_regression(X, y)
feat_importances = pd.Series(importances,X.columns)
sorted_importances = feat_importances.sort_values()
sorted_importances.plot(kind='barh')
plt.show()
```



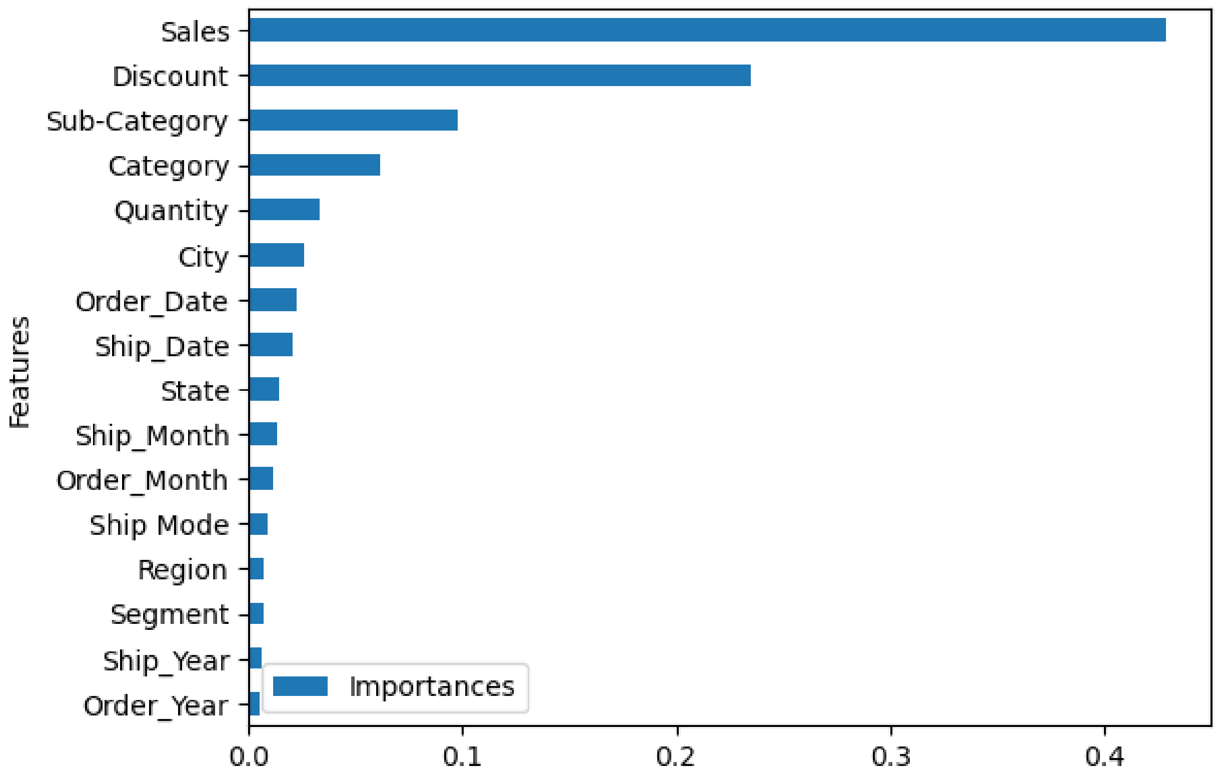

```
sorted_importances.sort_values(ascending = False)

Sales           1.860261
Sub-Category    0.526144
Discount        0.413933
Quantity        0.333369
Category        0.173386
State           0.145338
City            0.066758
Region          0.032112
```

```
Ship_Date       0.009801
Ship_Month      0.005424
Segment         0.005205
Order_Month     0.004082
Ship Mode       0.001661
Order_Date      0.000000
Order_Year      0.000000
Ship_Year       0.000000
dtype: float64
```

```
from sklearn.ensemble import RandomForestRegressor

# Create a Random Forest Regressor
rf = RandomForestRegressor(n_estimators=100, random_state=42)

# Fit the model to the data
rf.fit(X, y)

# Get the feature importances
importances = rf.feature_importances_
feature_names = X.columns

# Print the feature importances
print("Feature Importances:")
for feature, importance in zip(feature_names, importances):
    print(f"{feature}: {importance}")

# Creating a dataframe for visualization
final_df =
pd.DataFrame({'Features':feature_names,"Importances":importances})
final_df.set_index('Features',inplace=True)
sorted_importances = final_df.sort_values(by = 'Importances')
sorted_importances.plot(kind='barh')
plt.show()
```

```
Feature Importances:
Ship Mode: 0.008911285576783426
Segment: 0.0072927010410662576
City: 0.02621589767349239
State: 0.014200564090727098
Region: 0.007292996153653858
Category: 0.06181263337172867
Sub-Category: 0.09798699621456057
Sales: 0.42878472609289353
Quantity: 0.03346845420251476
Discount: 0.23458619256844299
Order_Date: 0.022754068050305446
Order_Month: 0.0118109692495668
Order_Year: 0.005420608809408856
Ship_Date: 0.020575675492120506
```

```
Ship_Month: 0.013132154433232248
Ship_Year: 0.005754076979502389
```




```
sorted_importances.sort_values(by ='Importances', ascending=False)
```

Features	Importances
Sales	0.428785
Discount	0.234586
Sub-Category	0.097987
Category	0.061813
Quantity	0.033468
City	0.026216
Order_Date	0.022754
Ship_Date	0.020576
State	0.014201
Ship_Month	0.013132
Order_Month	0.011811
Ship Mode	0.008911
Region	0.007293
Segment	0.007293
Ship_Year	0.005754
Order_Year	0.005421

```
sorted_importances[sorted_importances.values>=0.011811]
```

```
                  Importances
Features
Ship_Month           0.013132
State                0.014201
Ship_Date            0.020576
Order_Date           0.022754
City                 0.026216
Quantity             0.033468
Category             0.061813
Sub-Category         0.097987
Discount             0.234586
Sales                0.428785
```

Now will extract Top 10 features for building my Model

```
new_X =
X[sorted_importances[sorted_importances.values>=0.011811].index]
new_X
```

```
      Ship_Month     State  Ship_Date  Order_Date      City  Quantity  \
0       0.981561 -0.460377  -0.443083   -0.763538 -0.619993 -0.746874

1      -0.515263 -1.229999   0.125956   -0.304399 -0.108224 -0.746874

2       0.682196 -0.909323   0.239764   -0.533969 -0.924088 -0.746874

3      -0.515263 -1.229999  -0.215467   -0.763538 -0.108224  1.661660

4      -0.515263 -1.229999  -0.215467   -0.763538 -0.108224  0.216539

...          ...       ...        ...         ...       ...       ...

8108    0.981561 -0.845188   0.694995    0.269524 -1.880874  0.698246

8109   -2.012088 -0.909323   0.922610    0.728663  0.106868 -0.265167

8110   -1.413358 -1.229999  -1.353545    1.302586 -1.287519 -0.746874

8111   -1.413358 -1.229999  -1.353545    1.302586 -1.287519 -0.746874

8112   -1.413358 -1.229999  -1.353545    1.302586 -1.287519  0.216539


      Category  Sub-Category  Discount     Sales
0    -1.666962     -0.697807 -0.753331  1.094304
1     0.045595      0.513816 -0.753331 -0.489379
2     0.045595      1.321565  0.258393 -0.439770
3    -1.666962      0.311879 -0.753331 -0.270145
4     0.045595     -1.101682 -0.753331 -0.536376
...        ...           ...       ...       ...
```

```
8108  1.758152      1.119628 -0.753331  0.736640
8109 -1.666962      0.311879  0.258393 -0.421330
8110 -1.666962      0.311879 -0.753331  0.005818
8111  1.758152      1.119628  0.258393  1.072637
8112  0.045595      0.917690 -0.753331 -0.393464

[8113 rows x 10 columns]
```

5. Model Building

1.Splitting the data into train and test data set

```
# Algorithm used to build ML Model

from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor

X_train,X_test,y_train,y_test=train_test_split(new_X,y,test_size=0.20,
random_state=1)

X_train.shape,X_test.shape,y_train.shape,y_test.shape

((6490, 10), (1623, 10), (6490,), (1623,))
```

2.Scaling the data

```
sc=StandardScaler()

X_train_scaled=sc.fit_transform(X_train)
X_test_scaled=sc.transform(X_test)
```

3.Model Application

Now will try to build with various model like linear regression, Decision Tree, Randomforest etc

```
models=[LinearRegression(),DecisionTreeRegressor(),RandomForestRegress
or()]

# Model Evaluation
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_squared_error,r2_score

for i in range(3):
    models[i].fit(X_train_scaled,y_train)

    print(f'{models[i]}: ')
    y_pred_train=models[i].predict(X_train_scaled)
```

```
    y_pred_test=models[i].predict(X_test_scaled)

    print('MSE_train: ',mean_squared_error(y_train,y_pred_train))
    print('MSE_test: ',mean_squared_error(y_test,y_pred_test))

    print('RMSE_train: 
',np.sqrt(mean_squared_error(y_train,y_pred_train)))
    print('RMSE_test: 
',np.sqrt(mean_squared_error(y_test,y_pred_test)))

    print('R2_score_train: ',r2_score(y_train,y_pred_train))
    print('R2_score_test: ',r2_score(y_test,y_pred_test))

    print()
    print('--'*55)
```

```
LinearRegression():
MSE_train:  233.30233550112155
MSE_test:  247.73395770143446
RMSE_train:  15.274237640586897
RMSE_test:  15.739566630038912
R2_score_train:  0.33163831620438056
R2_score_test:  0.2734809905486105

----------------------------------------------------------------------
----------------------------------------
DecisionTreeRegressor():
MSE_train:  6.498139907550103e-05
MSE_test:  177.12520850863214
RMSE_train:  0.0080611040853906
RMSE_test:  13.30883948767255
R2_score_train:  0.9999998138420809
R2_score_test:  0.4805523141496354

----------------------------------------------------------------------
----------------------------------------
RandomForestRegressor():
MSE_train:  12.15171901122418
MSE_test:  97.74320253734766
RMSE_train:  3.4859315844153023
RMSE_test:  9.886516198203879
R2_score_train:  0.9651879036619675
R2_score_test:  0.7133526007216847

----------------------------------------------------------------------
----------------------------------------
```

Random Forest had the best accuracy based on training and test dataset but it is overfitted so we need to perfrom some hyper parameter tuning technique

Randomized Search CV

```
#Randomized Search CV

# Number of trees in random forest
n_estimators = list(range(100,1300,100))
# Number of features to consider at every split
max_features = ['auto', 'sqrt']
# Maximum number of levels in tree
max_depth = [int(x) for x in np.linspace(5, 30, num = 6)]
# Minimum number of samples required to split a node
min_samples_split = [2, 5, 10, 15, 100]
# Minimum number of samples required at each leaf node
min_samples_leaf = [1, 2, 5, 10]

# Create the random grid

random_grid = {'n_estimators': n_estimators,
               'max_features': max_features,
               'max_depth': max_depth,
               'min_samples_split': min_samples_split,
               'min_samples_leaf': min_samples_leaf}

from sklearn.model_selection import RandomizedSearchCV

rf_regressor=RandomForestRegressor()
rf_model=RandomizedSearchCV(estimator=rf_regressor,param_distributions
=random_grid,
                            cv=3,random_state=0)
rf_model.fit(X_train_scaled,y_train)
```

```
RandomizedSearchCV(cv=3, estimator=RandomForestRegressor(),
                   param_distributions={'max_depth': [5, 10, 15, 20,
25, 30],
                                        'max_features': ['auto',
'sqrt'],
                                        'min_samples_leaf': [1, 2, 5,
10],
                                        'min_samples_split': [2, 5,
10, 15,
                                                              100],
                                        'n_estimators': [100, 200,
300, 400,
                                                         500, 600,
700, 800,
                                                         900, 1000,
1100,
```

```
                                                                     1200]},
                   random_state=0)

# best parameter
rf_model.best_params_

{'n_estimators': 200,
 'min_samples_split': 5,
 'min_samples_leaf': 2,
 'max_features': 'auto',
 'max_depth': 15}

final_model = RandomForestRegressor(n_estimators =
900,min_samples_split = 10,min_samples_leaf = 2,max_features =
'sqrt',max_depth = 30)
final_model.fit(X_train_scaled,y_train)

RandomForestRegressor(max_depth=30, max_features='sqrt',
min_samples_leaf=2,
                      min_samples_split=10, n_estimators=900)

#predicting the values
trian_pred=final_model.predict(X_train_scaled)
test_pred=final_model.predict(X_test_scaled)

print('R2 score of training dataset',r2_score(y_train,trian_pred))
print('R2 score of testing dataset',r2_score(y_test,test_pred))

R2 score of training dataset 0.8673454913927121
R2 score of testing dataset 0.6868502834053685

from sklearn.metrics import mean_absolute_error
print('r2 score train', r2_score)
print('r2_score test:',r2_score(y_test,test_pred))
print('MAE:', mean_absolute_error(y_test, test_pred))
print('MSE:', mean_squared_error(y_test, test_pred))
print('RMSE:', np.sqrt(mean_squared_error(y_test, test_pred)))

r2 score train <function r2_score at 0x000002031C381EA0>
r2_score test: 0.6868502834053685
MAE: 5.461817594030465
MSE: 106.78016354128344
RMSE: 10.333448772858143
```

After Randomized Search CV(hypertuning),the accuracy of random forest decreases.

Now i will hyper tuning for Grid Serach CV

Grid Search CV

```
sc=StandardScaler()

X_train_scaled=sc.fit_transform(X_train)
X_test_scaled=sc.transform(X_test)

# Now will perform RandomForest Classifier

param_grid = {
    'n_estimators': list(range(500,1500,500)),
    'max_features': ['auto', 'sqrt', 'log2'],
    'max_depth' : [7,8,12,15,20,25],
}

from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestClassifier
rfr=RandomForestRegressor()
CV_rfr = GridSearchCV(estimator=rfr, param_grid=param_grid, cv= 5)
CV_rfr.fit(X_train_scaled,y_train)

GridSearchCV(cv=5, estimator=RandomForestRegressor(),
             param_grid={'max_depth': [7, 8, 12, 15, 20, 25],
                         'max_features': ['auto', 'sqrt', 'log2'],
                         'n_estimators': [500, 1000]})

CV_rfr.best_params_

{'max_depth': 12, 'max_features': 'auto', 'n_estimators': 500}

final_model_G = RandomForestRegressor(max_depth = 12,max_features =
'auto',n_estimators =500)
final_model_G.fit(X_train_scaled,y_train)

RandomForestRegressor(max_depth=12, max_features='auto',
n_estimators=500)

#predicting the values
trian_pred=final_model_G.predict(X_train_scaled)
test_pred=final_model_G.predict(X_test_scaled)

print('R2 score of training dataset',r2_score(y_train,trian_pred))
print('R2 score of testing dataset',r2_score(y_test,test_pred))

R2 score of training dataset 0.927689539918339
R2 score of testing dataset 0.7111925114133865
```

```
from sklearn.metrics import mean_absolute_error
print('r2_score test:',r2_score(y_test,test_pred))
print('MAE:', mean_absolute_error(y_test, test_pred))
print('MSE:', mean_squared_error(y_test, test_pred))
print('RMSE:', np.sqrt(mean_squared_error(y_test, test_pred)))
```

```
r2_score test: 0.7111925114133865
MAE: 4.650119552254509
MSE: 98.47976615973295
RMSE: 9.923697202138573
```

After Grid Search(hypertuning),the accuracy of random forest Increases.