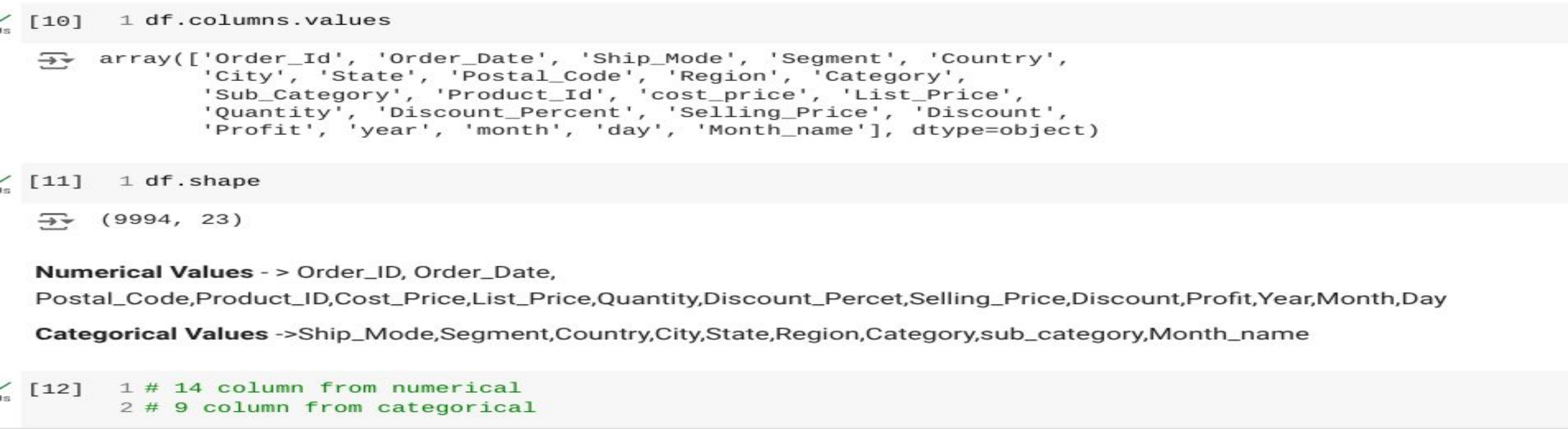
# Presentation  For Retail Order Data Analysis

Project - 1



```
[10]  df.columns.values
```

```
array(['Order_Id', 'Order_Date', 'Ship_Mode', 'Segment', 'Country',
       'City', 'State', 'Postal_Code', 'Region', 'Category',
       'Sub_Category', 'Product_Id', 'cost_price', 'List_Price',
       'Quantity', 'Discount_Percent', 'Selling_Price', 'Discount',
       'Profit', 'year', 'month', 'day', 'Month_name'], dtype=object)
```

```
[11]  df.shape
```

```
(9994, 23)
```

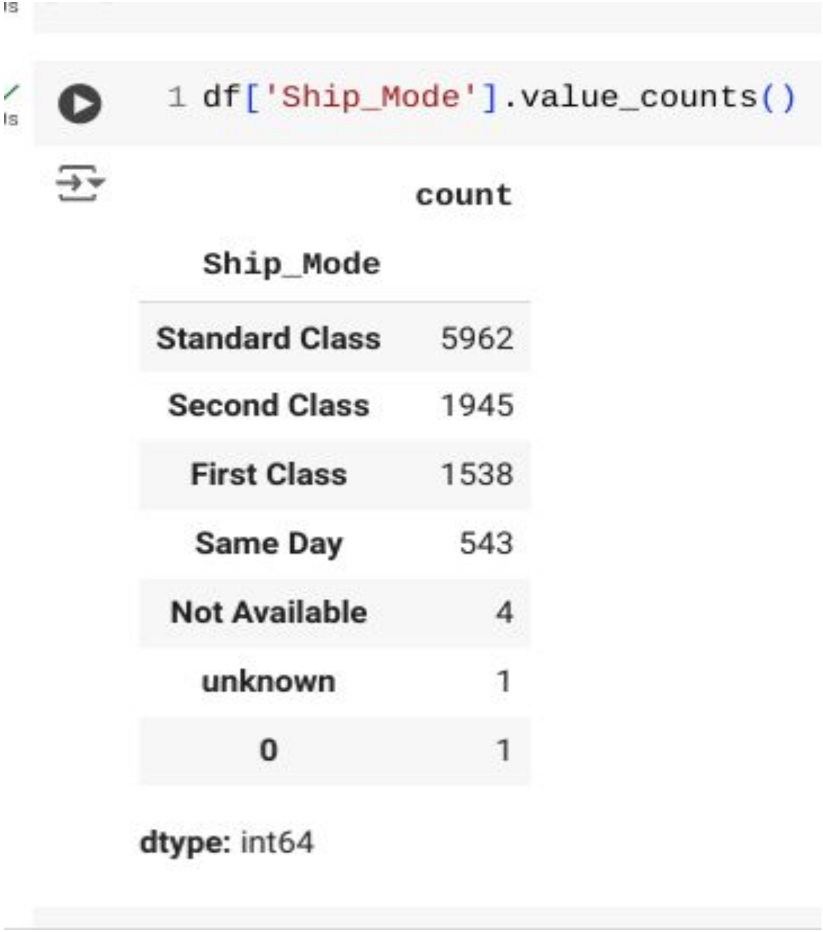
**Numerical Values** - > Order_ID, Order_Date, Postal_Code,Product_ID,Cost_Price,List_Price,Quantity,Discount_Percet,Selling_Price,Discount,Profit,Year,Month,Day

**Categorical Values** ->Ship_Mode,Segment,Country,City,State,Region,Category,sub_category,Month_name

```
[12]  # 14 column from numerical
      # 9 column from categorical
```

# Insights:

There are 14 numeric columns and 9 categorical columns

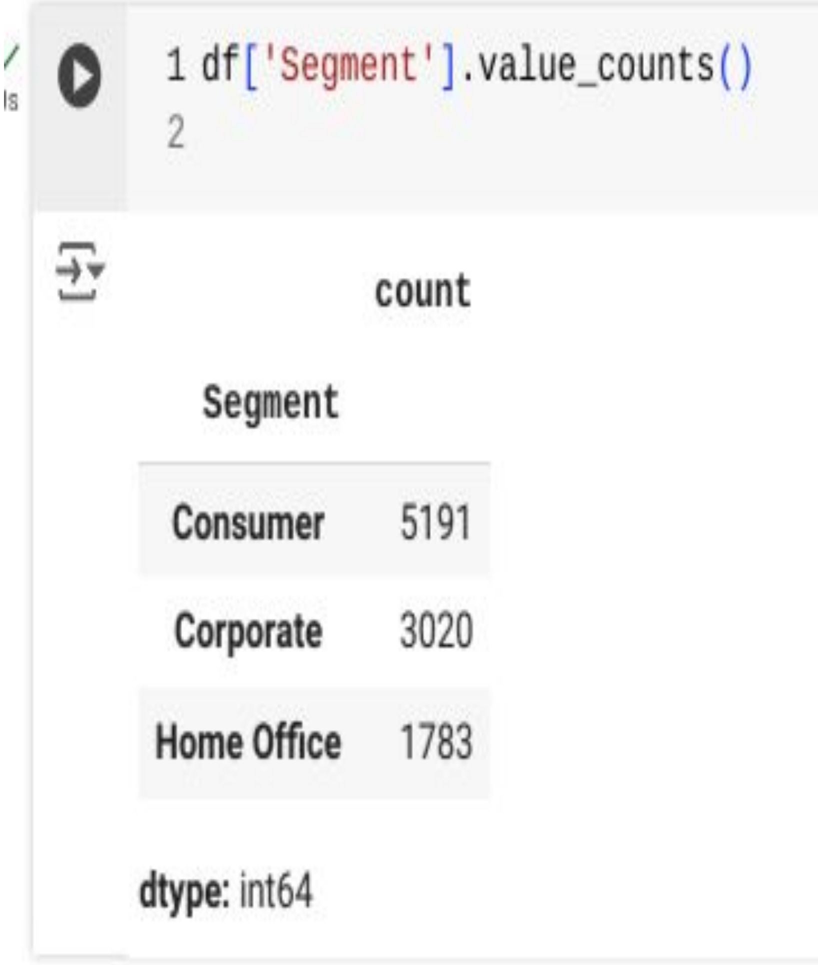


```python
df['Ship_Mode'].value_counts()
```

```
                  count
     Ship_Mode
Standard Class     5962
  Second Class     1945
   First Class     1538
      Same Day      543
 Not Available        4
       unknown        1
             0        1

dtype: int64
```

# Shiping Mode Analysing

Most of Peoples Preferred Ship Mode was Standard Class , second most preferred as Second Class head of First Class

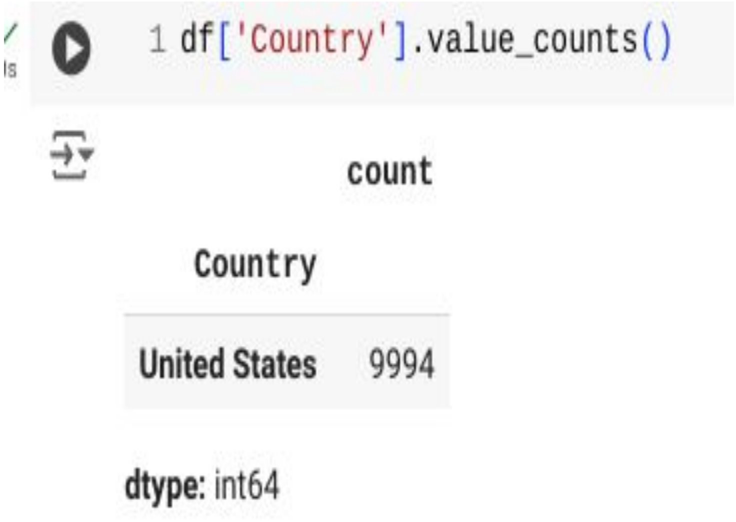


```python
df['Segment'].value_counts()
```

```
              count
   Segment

  Consumer     5191
  Corporate    3020
 Home Office   1783

dtype: int64
```

# Segment

Consumer Segment highest quantity was placed, 2nd Corporate Segment and Home Office was Ordered least Segment



```python
df['Country'].value_counts()
```

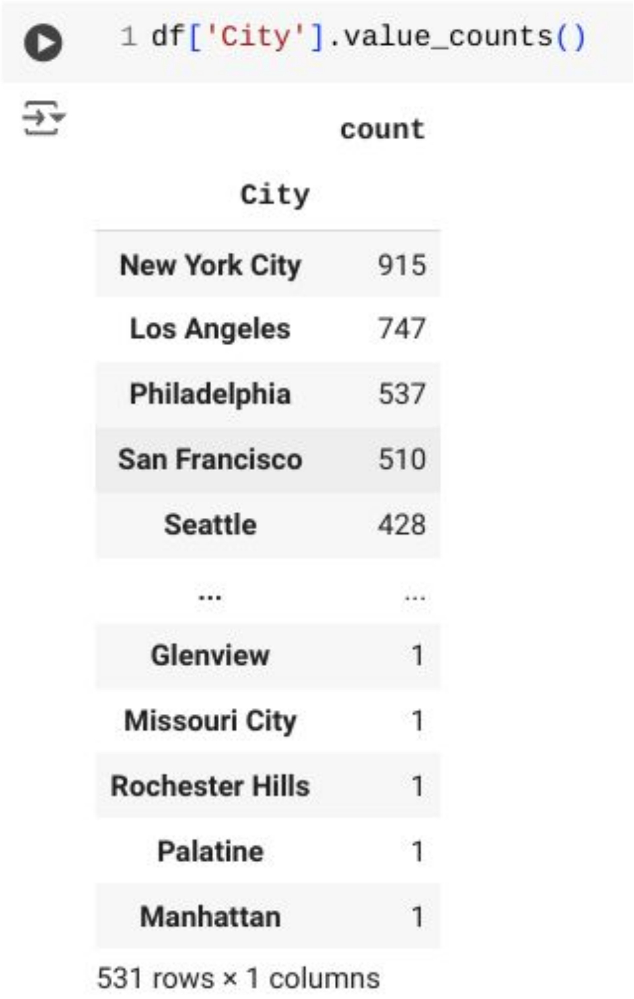
```
                   count
       Country
United States       9994

dtype: int64
```

# Countries

There are focus on only one Country - United States

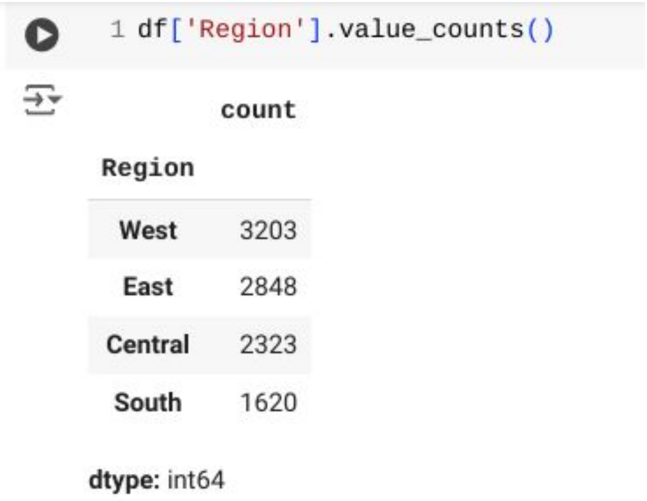
# Cities

There are focus of 531 Cities on United States



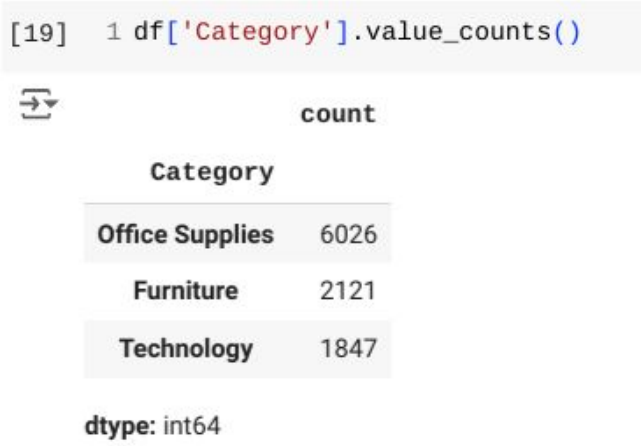
```python
df['City'].value_counts()
```

```
                  count
City
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

531 rows × 1 columns
```



```
df['Region'].value_counts()
```

```
          count
Region
West       3203
East       2848
Central    2323
South      1620

dtype: int64
```



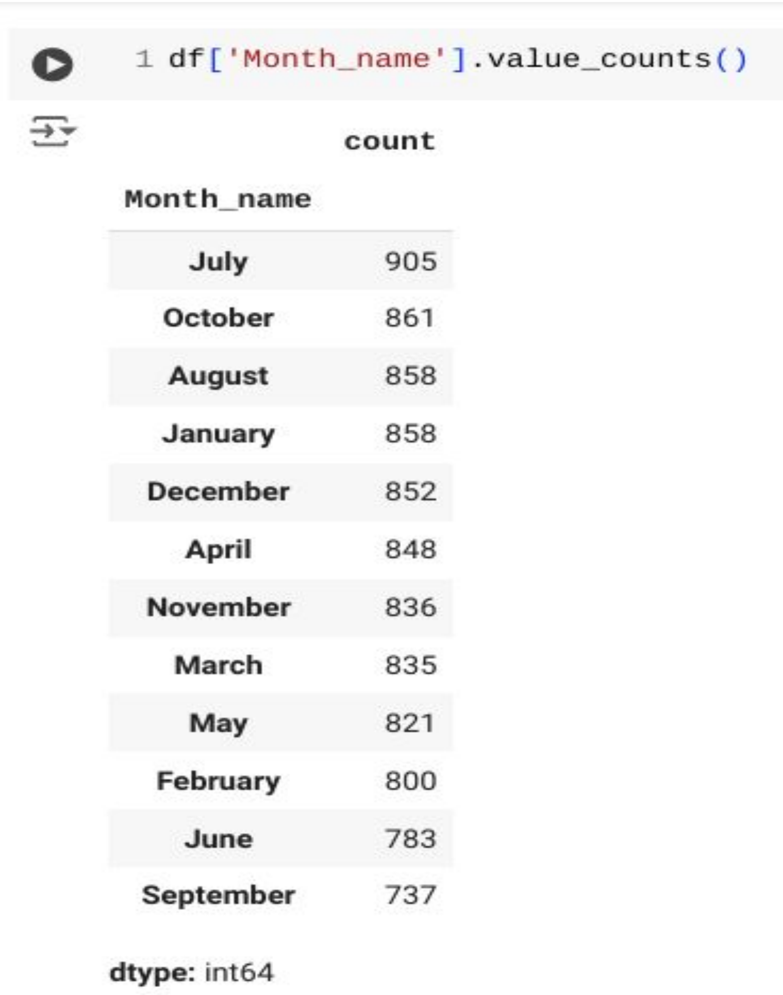
```
[19] df['Category'].value_counts()
```

```
                  count
     Category
Office Supplies    6026
Furniture          2121
Technology         1847

dtype: int64
```

# Region & Categories

There are Focus on 4 Region on 531 cities of United States

Based on 3 Categories - Office Supplies,Furniture, Technology



```
df['Month_name'].value_counts()
```

| Month_name | count |
|---|---|
| July | 905 |
| October | 861 |
| August | 858 |
| January | 858 |
| December | 852 |
| April | 848 |
| November | 836 |
| March | 835 |
| May | 821 |
| February | 800 |
| June | 783 |
| September | 737 |

**dtype:** int64

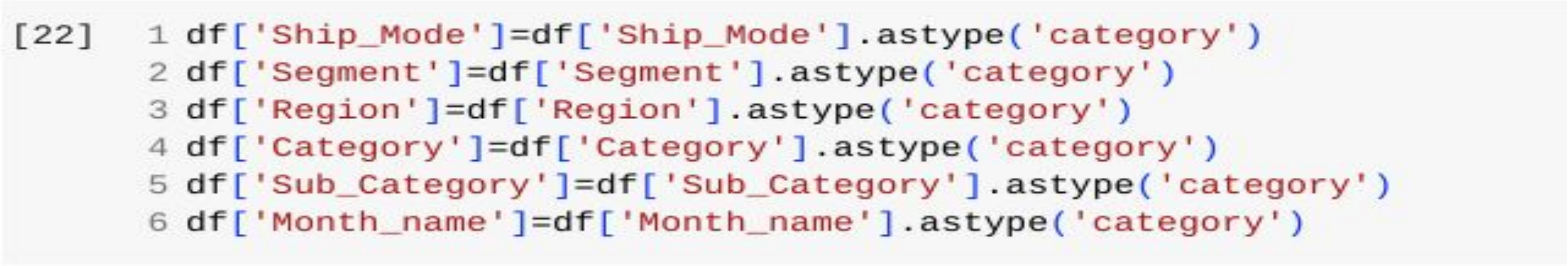
# Months

Month wise Quantity Shipped analysing

Maximum Quantity hold by July Month

Minimum Quantity hold by September Month

Average Quantity hold by November Month

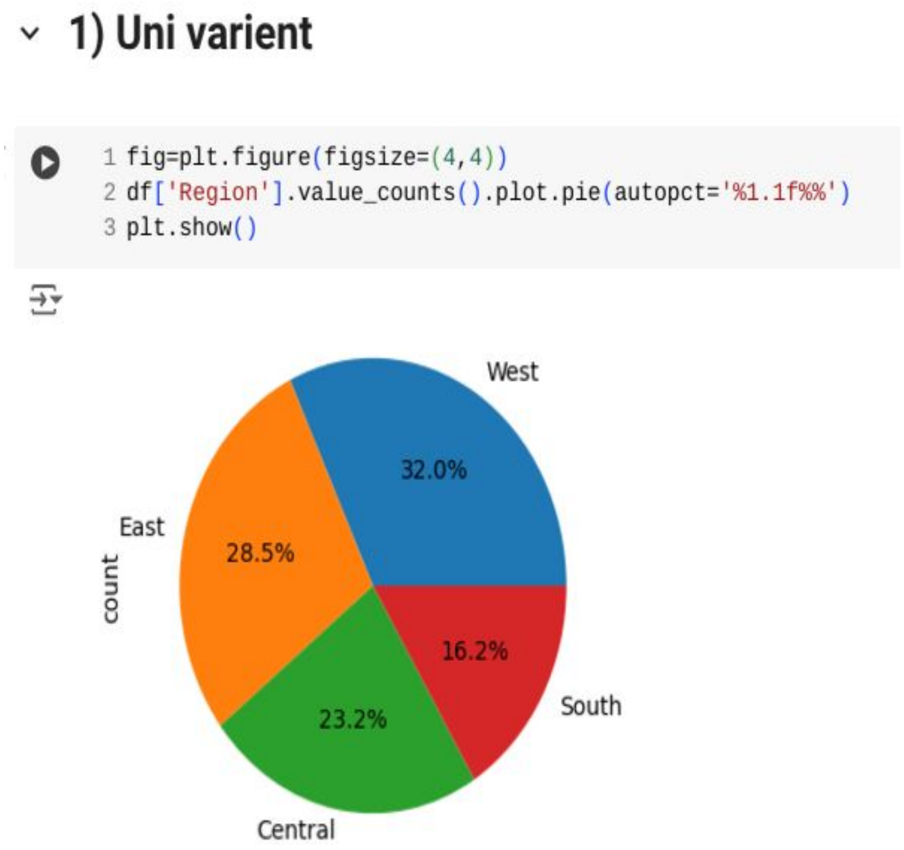


[22]

```
df['Ship_Mode']=df['Ship_Mode'].astype('category')
df['Segment']=df['Segment'].astype('category')
df['Region']=df['Region'].astype('category')
df['Category']=df['Category'].astype('category')
df['Sub_Category']=df['Sub_Category'].astype('category')
df['Month_name']=df['Month_name'].astype('category')
```

# Change object to category

Change object to category for particular columns for execute uni variant and bi variant effectively

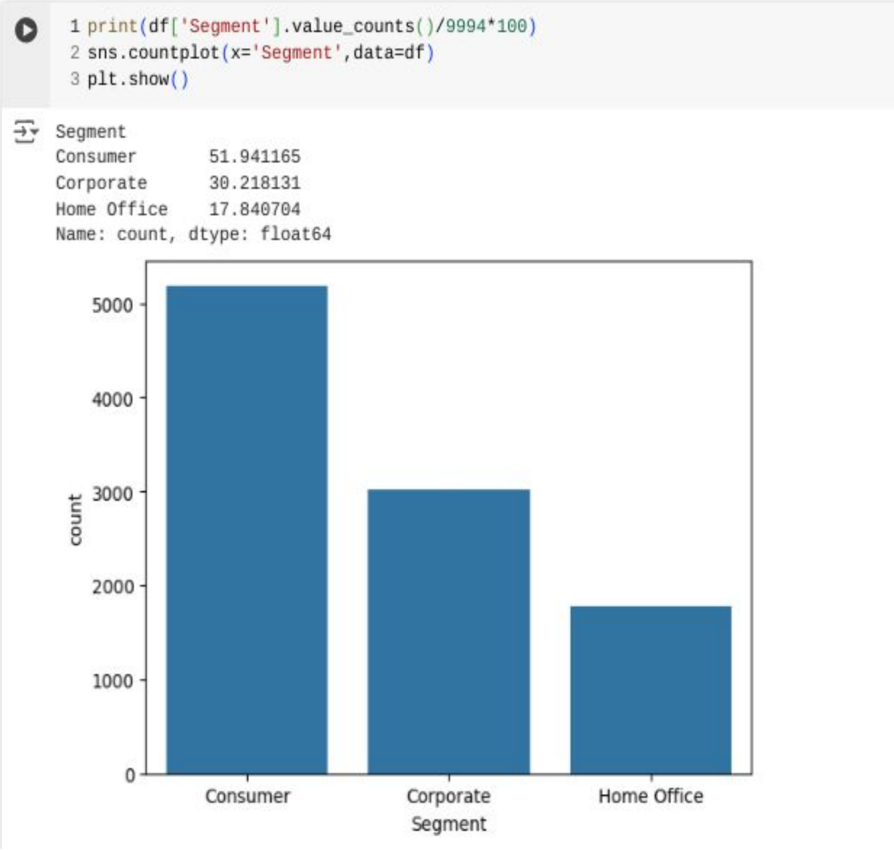


## 1) Uni varient

```
fig=plt.figure(figsize=(4,4))
df['Region'].value_counts().plot.pie(autopct='%1.1f%%')
plt.show()
```

# Insights:-

out of 9994 orders - 32% from West, 28% from East, 23% from Central, 16% from South

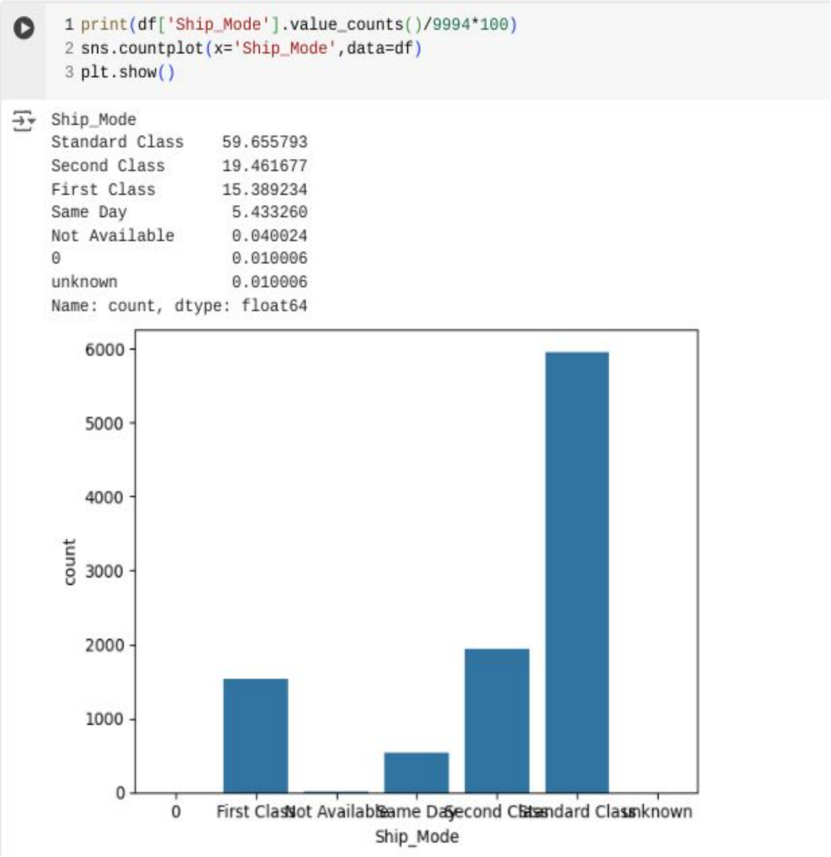


```python
print(df['Segment'].value_counts()/9994*100)
sns.countplot(x='Segment',data=df)
plt.show()
```

```
Segment
Consumer       51.941165
Corporate      30.218131
Home Office    17.840704
Name: count, dtype: float64
```

# Insights:-

Project Segment that shows count from consumer, corporate and home office out of 100 , 51 where is consumer, 30 where corporate segment, 17 were home office



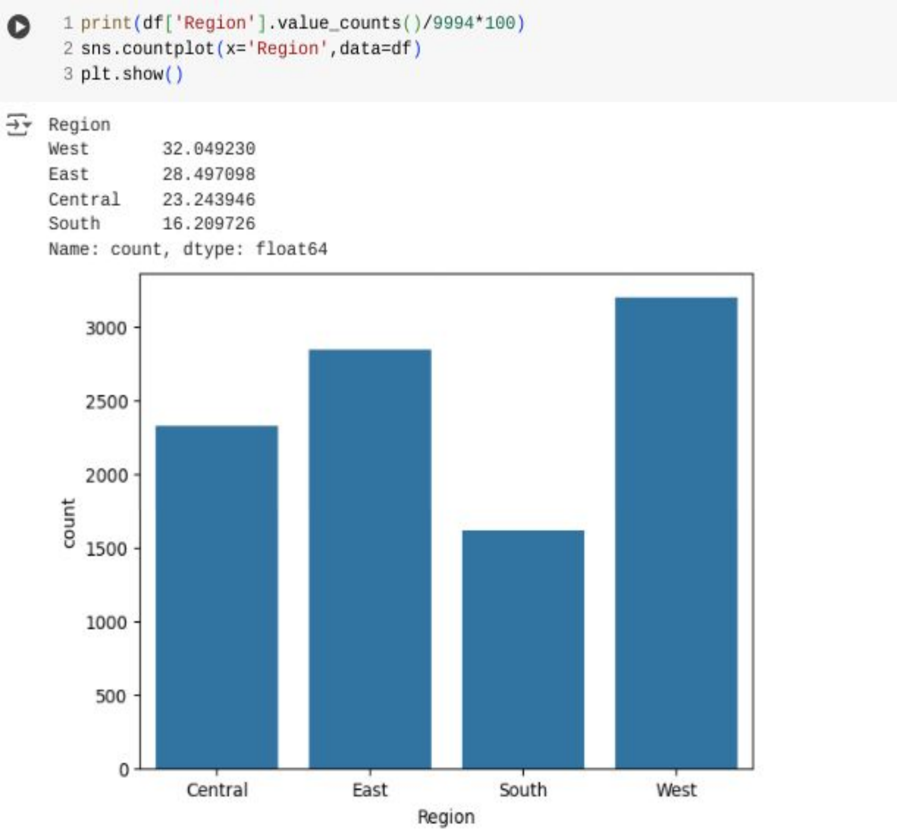
```
print(df['Ship_Mode'].value_counts()/9994*100)
sns.countplot(x='Ship_Mode',data=df)
plt.show()
```

```
Ship_Mode
Standard Class    59.655793
Second Class      19.461677
First Class       15.389234
Same Day           5.433260
Not Available      0.040024
0                  0.010006
unknown            0.010006
Name: count, dtype: float64
```

# Insights:-

its shows the count of ship_modes out of 100 - Standard Class 59, Second Class 19, First Class 15, Same Day 5,

**Maximum orders Prefers on Standard Class Shipmodes**



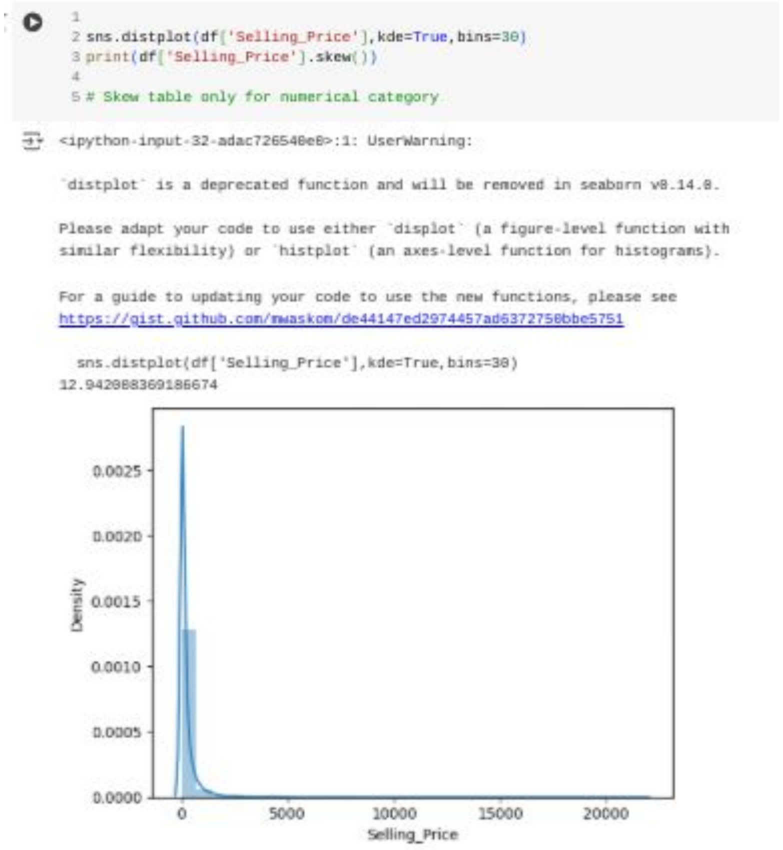
```python
print(df['Region'].value_counts()/9994*100)
sns.countplot(x='Region',data=df)
plt.show()
```

```
Region
West       32.049230
East       28.497098
Central    23.243946
South      16.209726
Name: count, dtype: float64
```

# Insights:-

Out of 100 customers of each - West 32, East 28, Central 23, South 16.

**Maximum Customer Headed by - West**

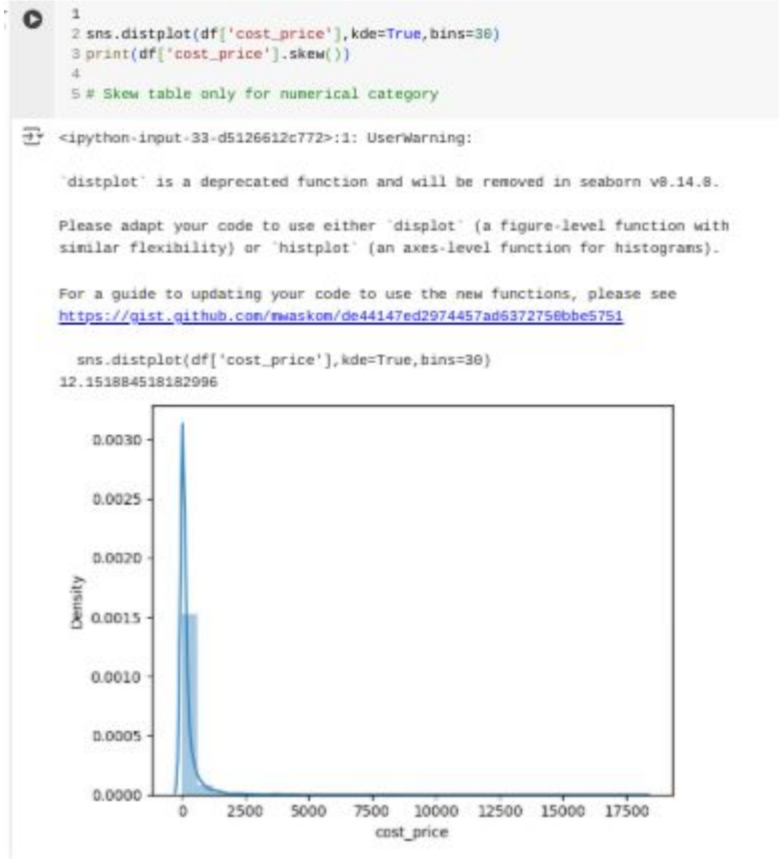


```python

sns.distplot(df['Selling_Price'],kde=True,bins=30)
print(df['Selling_Price'].skew())

# Skew table only for numerical category
```

```
<ipython-input-32-adac726540e0>:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df['Selling_Price'],kde=True,bins=30)
12.942088369186674
```

# Insights:-

the skew of selling price is around is 12.9 - its not normal distribution - its right tail or posative distribution - its not healthy chart

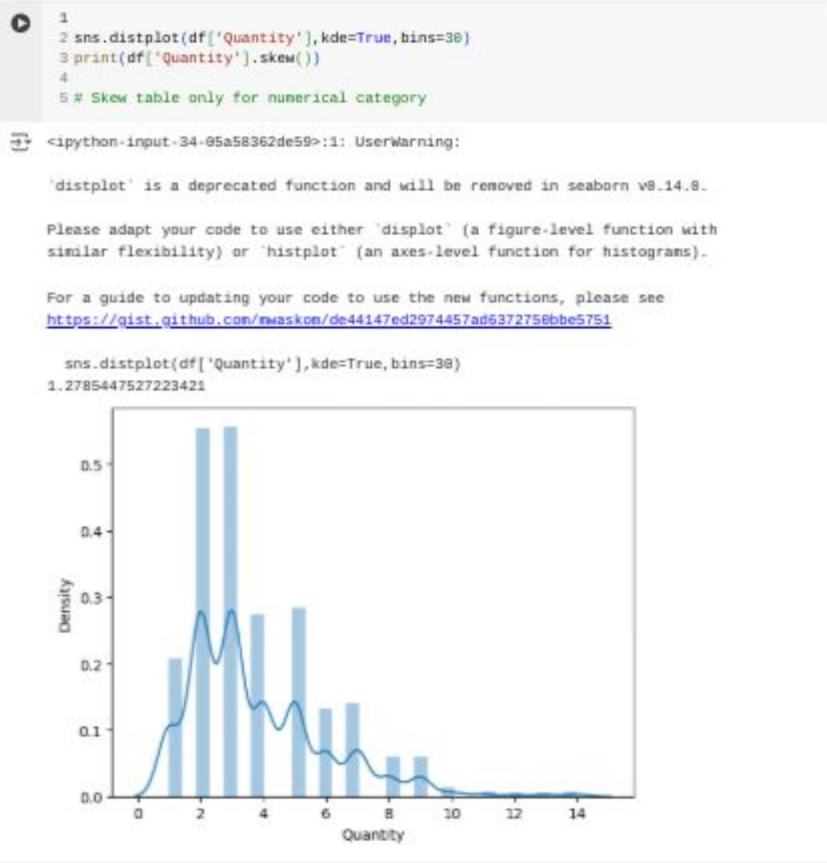


```python

sns.distplot(df['cost_price'],kde=True,bins=30)
print(df['cost_price'].skew())

# Skew table only for numerical category
```

```
<ipython-input-33-d5126612c772>:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df['cost_price'],kde=True,bins=30)
12.151884518182996
```

# Insights:-

the skew of cost_price is around is 12.15 - its not normal distribution - its right tail or posative distribution - its not healthy chart

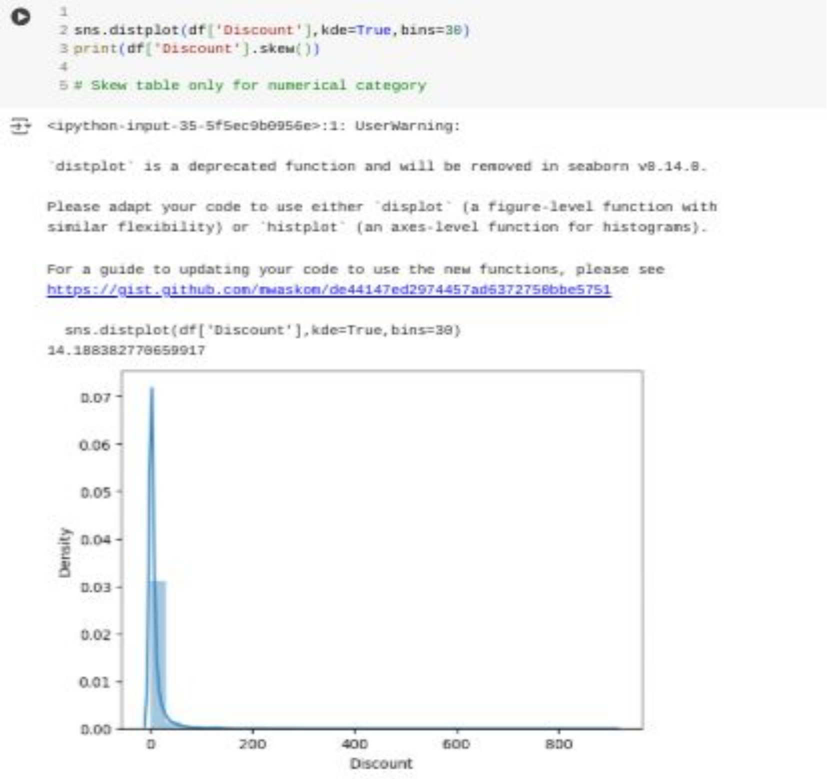


```

sns.distplot(df['Quantity'],kde=True,bins=30)
print(df['Quantity'].skew())

# Skew table only for numerical category
```

```
<ipython-input-34-05a58362de59>:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df['Quantity'],kde=True,bins=30)
1.2785447527223421
```

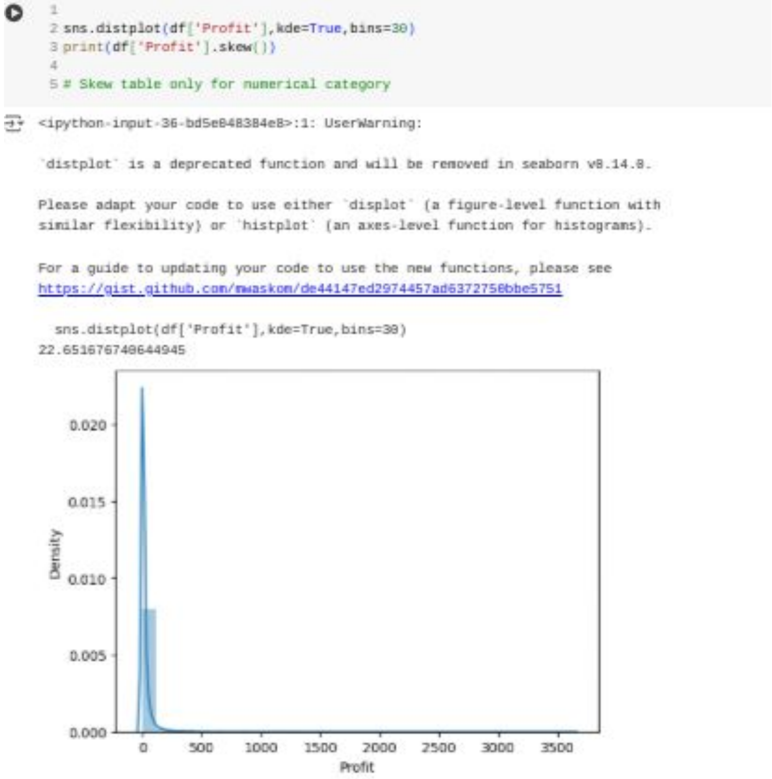
# Insights:

the skew of Quantity is around is 1.27 - its not normal distribution - its right tail or posative distribution - it is far better than selling price and cost price - **-0.5 to 0.5**



```
sns.distplot(df['Discount'],kde=True,bins=30)
print(df['Discount'].skew())

# Skew table only for numerical category
```

```
<ipython-input-35-5f5ec9b0956e>:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df['Discount'],kde=True,bins=30)
14.188382770659917
```



```
sns.distplot(df['Profit'],kde=True,bins=30)
print(df['Profit'].skew())

# Skew table only for numerical category
```

```
<ipython-input-36-bd5e048384e8>:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df['Profit'],kde=True,bins=30)
22.651676740644945
```

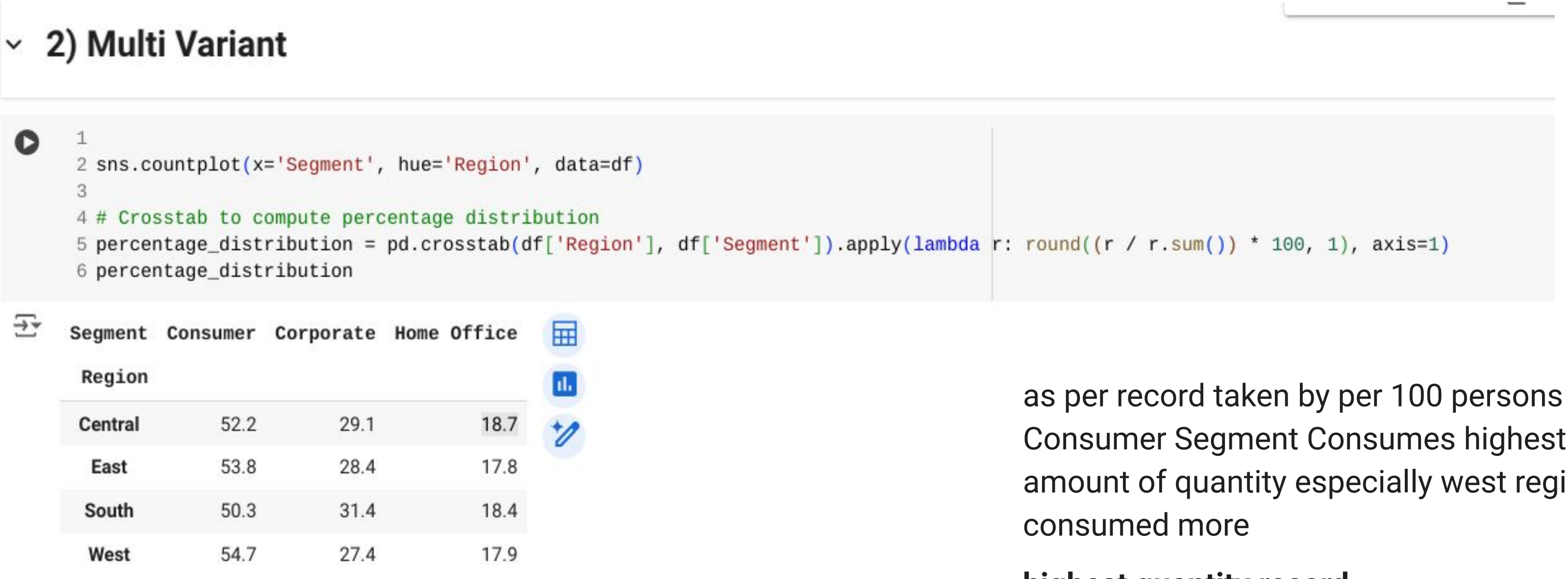
the skew of profit is 22.6 is very bad compare to selling price, cost price and quantity

# conclusion

this overall analysis look like cost price of product not focusing clearly of fields - they list low price product then they high cost products that they are overlapping

the skew point of discount is higher tha the selling price they give more discount to the product

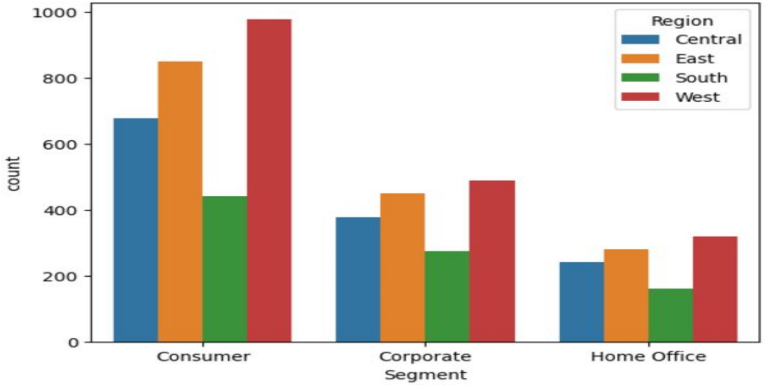
the skew point of profit is higher than the selling price,cost price, profit is very low not clearly manager



## 2) Multi Variant

```python

sns.countplot(x='Segment', hue='Region', data=df)

# Crosstab to compute percentage distribution
percentage_distribution = pd.crosstab(df['Region'], df['Segment']).apply(lambda r: round((r / r.sum()) * 100, 1), axis=1)
percentage_distribution
```

| Segment | Consumer | Corporate | Home Office |
|---|---|---|---|
| Region | | | |
| Central | 52.2 | 29.1 | 18.7 |
| East | 53.8 | 28.4 | 17.8 |
| South | 50.3 | 31.4 | 18.4 |
| West | 54.7 | 27.4 | 17.9 |



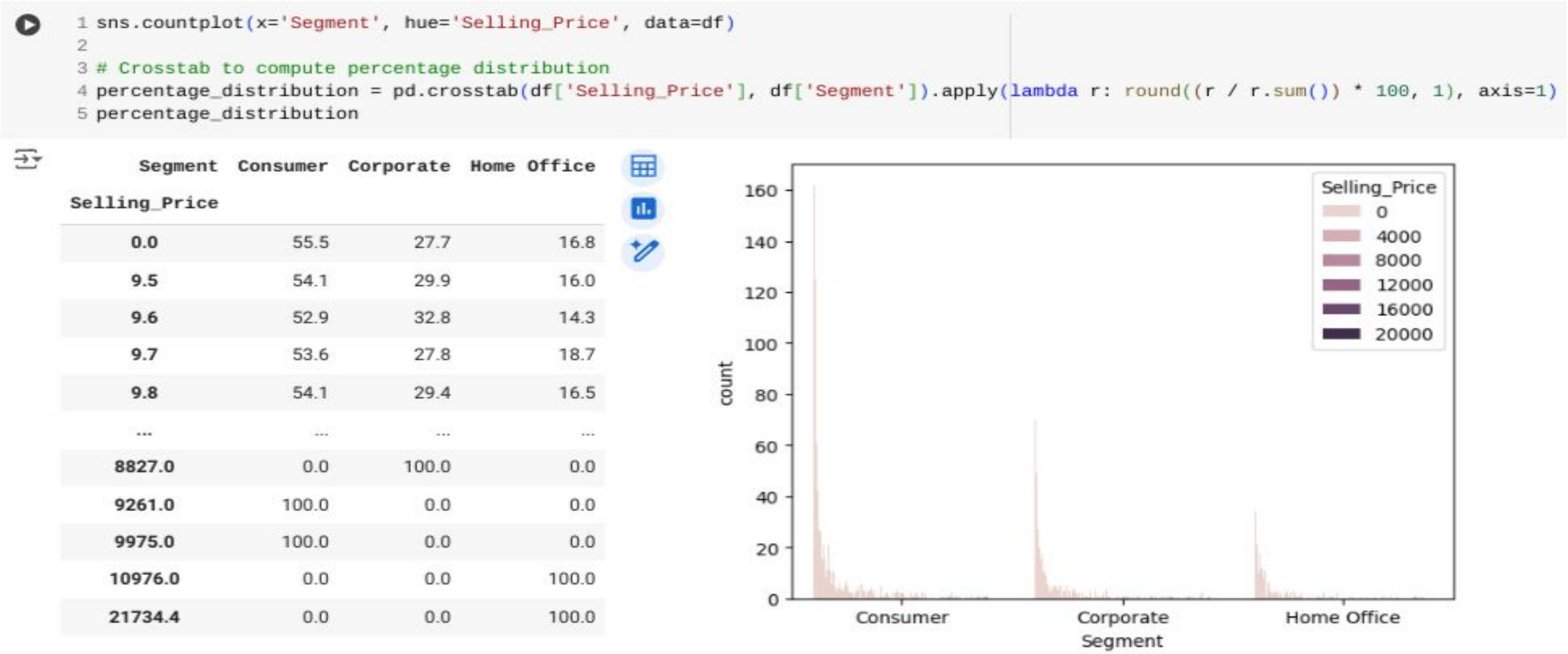
as per record taken by per 100 persons - Consumer Segment Consumes highest amount of quantity especially west region consumed more

**highest quantity record**

In Consumer - West Region consumed high 54% compare to higher than east, south and north

in corporate segment - south region consumed high 31.4%

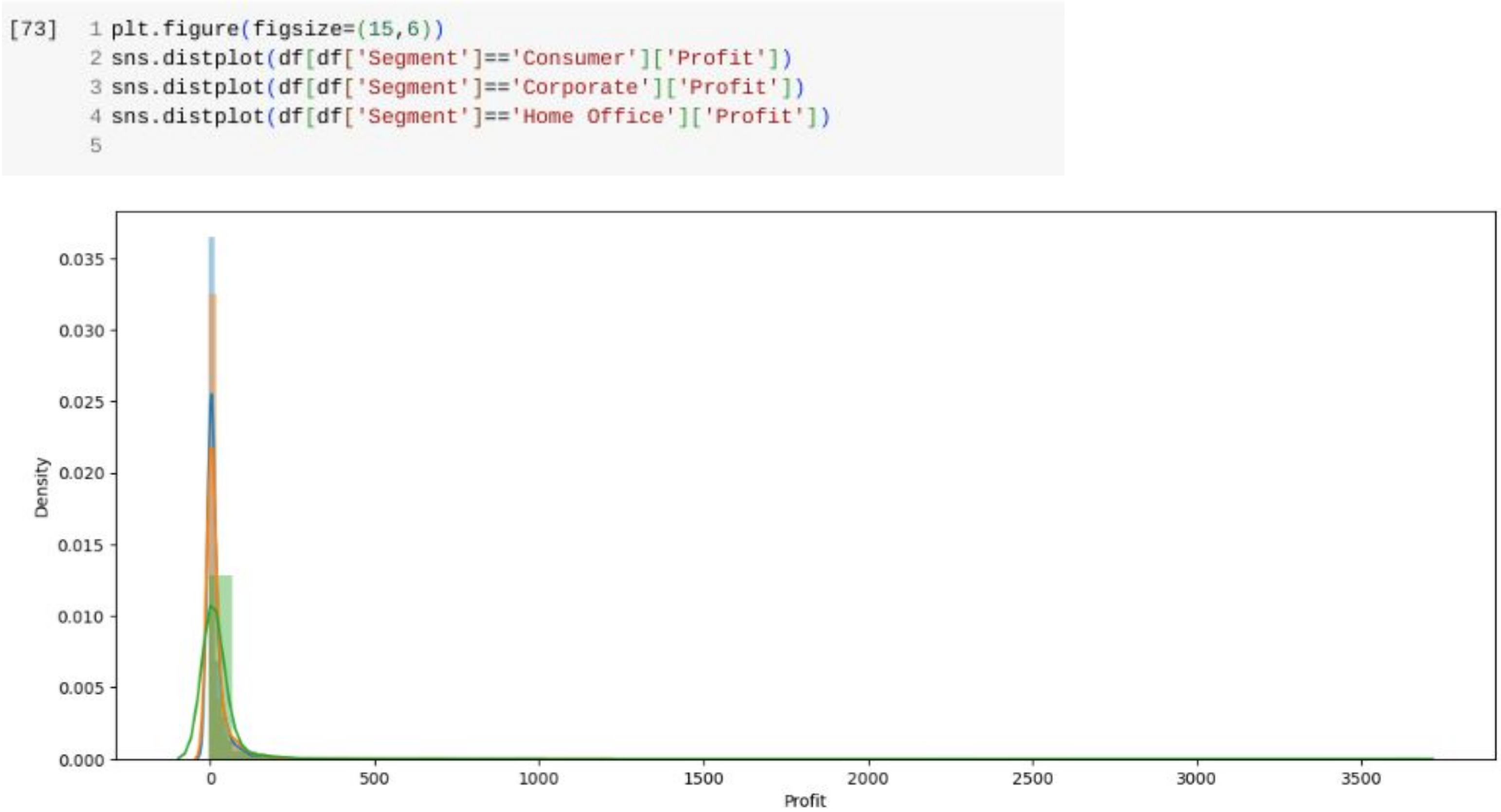
in home office segment - Central region consumed high 18.7%



```python
sns.countplot(x='Segment', hue='Selling_Price', data=df)

# Crosstab to compute percentage distribution
percentage_distribution = pd.crosstab(df['Selling_Price'], df['Segment']).apply(lambda r: round((r / r.sum()) * 100, 1), axis=1)
percentage_distribution
```

| Segment | Consumer | Corporate | Home Office |
|---|---|---|---|
| Selling_Price | | | |
| 0.0 | 55.5 | 27.7 | 16.8 |
| 9.5 | 54.1 | 29.9 | 16.0 |
| 9.6 | 52.9 | 32.8 | 14.3 |
| 9.7 | 53.6 | 27.8 | 18.7 |
| 9.8 | 54.1 | 29.4 | 16.5 |
| ... | ... | ... | ... |
| 8827.0 | 0.0 | 100.0 | 0.0 |
| 9261.0 | 100.0 | 0.0 | 0.0 |
| 9975.0 | 100.0 | 0.0 | 0.0 |
| 10976.0 | 0.0 | 0.0 | 100.0 |
| 21734.4 | 0.0 | 0.0 | 100.0 |

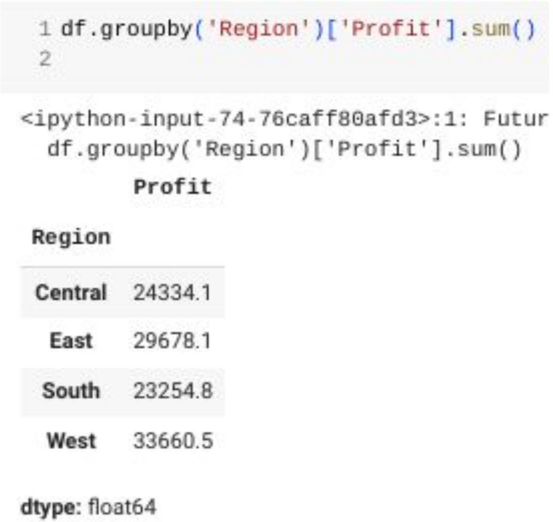
as per record taken by per 100 persons - Consumer Segment Consumed highest amount of Selling Price especially



[73]

```python
plt.figure(figsize=(15,6))
sns.distplot(df[df['Segment']=='Consumer']['Profit'])
sns.distplot(df[df['Segment']=='Corporate']['Profit'])
sns.distplot(df[df['Segment']=='Home Office']['Profit'])

```

# Insights:-



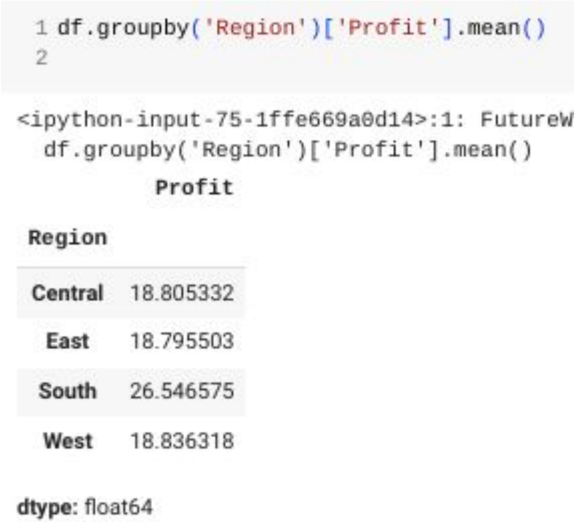
```python
df.groupby('Region')['Profit'].sum()

```

```
<ipython-input-74-76caff80afd3>:1: Futur
  df.groupby('Region')['Profit'].sum()
```

| | Profit |
|---|---|
| Region | |
| Central | 24334.1 |
| East | 29678.1 |
| South | 23254.8 |
| West | 33660.5 |

dtype: float64

Sum



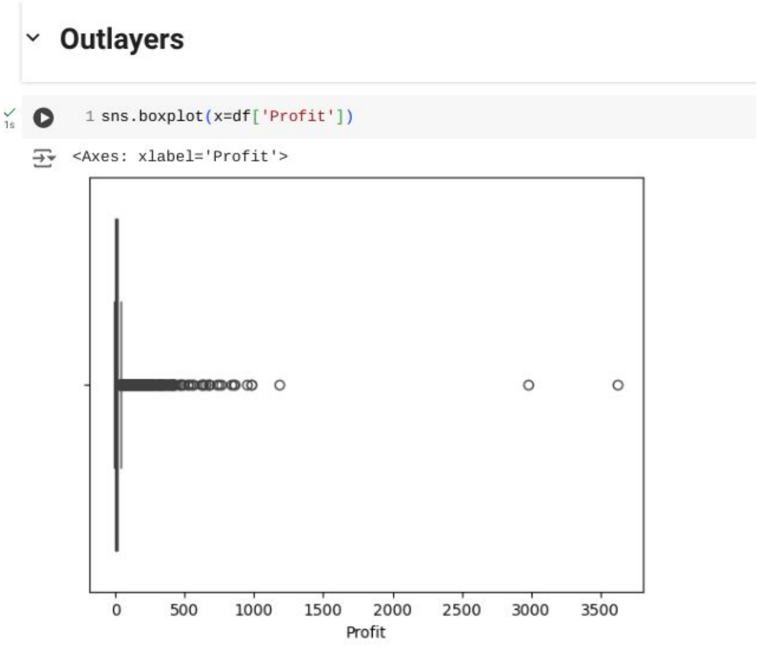
```python
df.groupby('Region')['Profit'].mean()

```

```
<ipython-input-75-1ffe669a0d14>:1: FutureW
  df.groupby('Region')['Profit'].mean()
```

| | Profit |
|---|---|
| Region | |
| Central | 18.805332 |
| East | 18.795503 |
| South | 26.546575 |
| West | 18.836318 |

dtype: float64

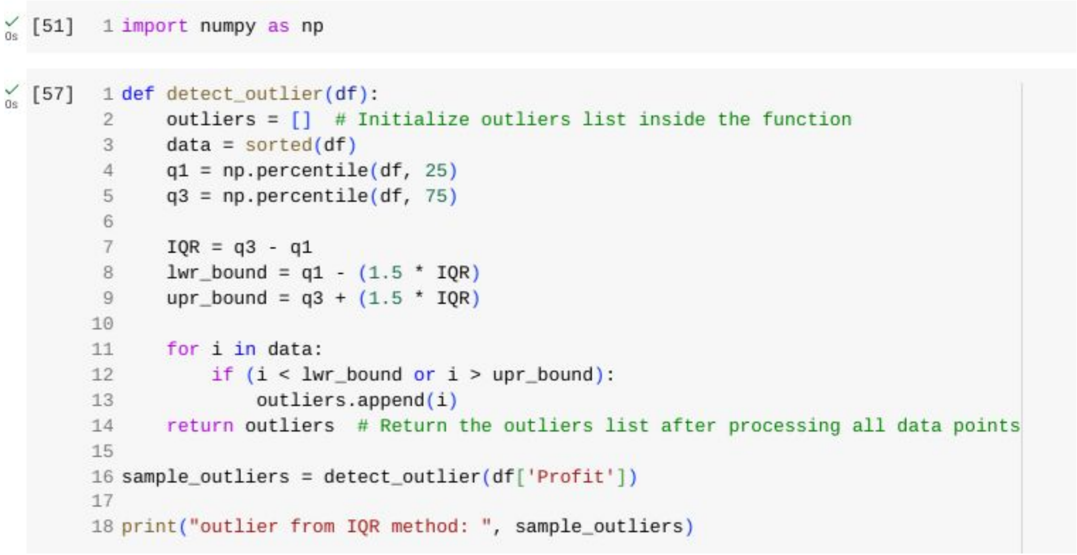
Mean



## Outlayers

```
sns.boxplot(x=df['Profit'])
```

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



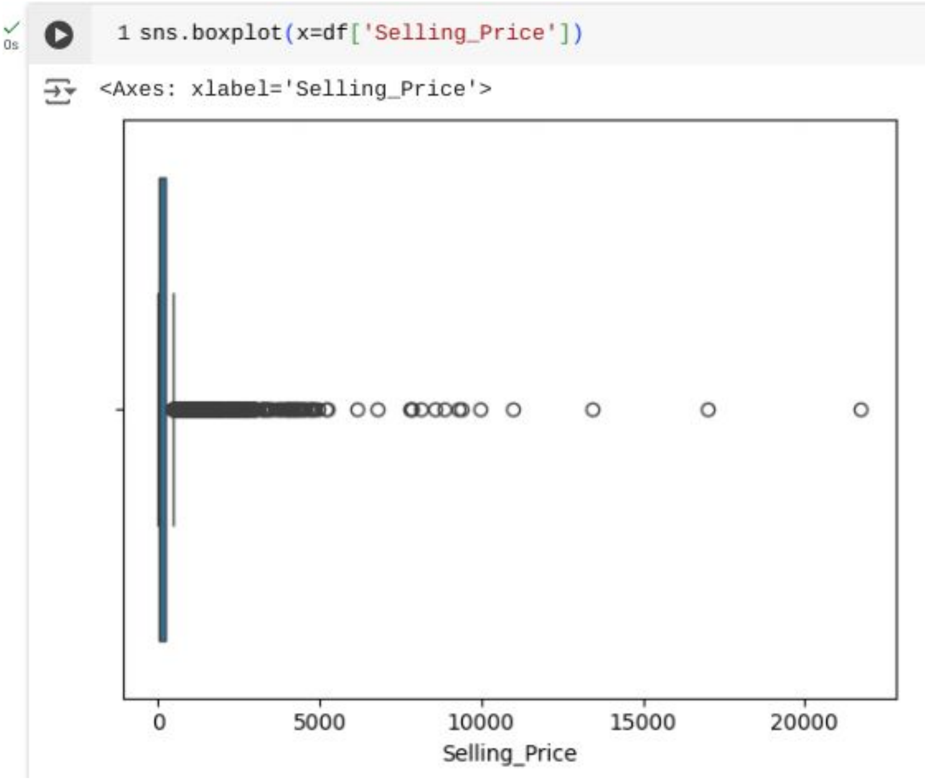
[51]

```
import numpy as np
```

[57]

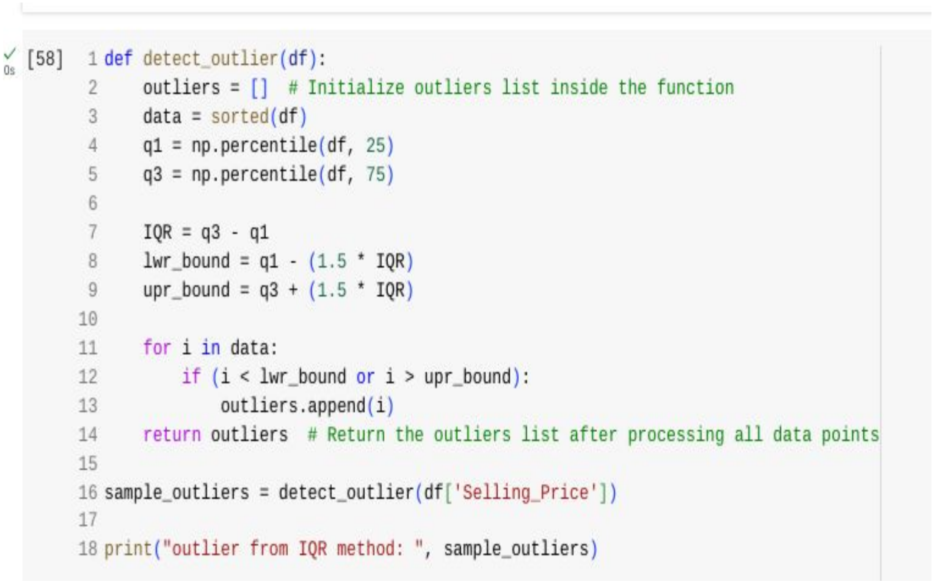
```
def detect_outlier(df):
    outliers = []  # Initialize outliers list inside the function
    data = sorted(df)
    q1 = np.percentile(df, 25)
    q3 = np.percentile(df, 75)

    IQR = q3 - q1
    lwr_bound = q1 - (1.5 * IQR)
    upr_bound = q3 + (1.5 * IQR)

    for i in data:
        if (i < lwr_bound or i > upr_bound):
            outliers.append(i)
    return outliers  # Return the outliers list after processing all data points

sample_outliers = detect_outlier(df['Profit'])

print("outlier from IQR method: ", sample_outliers)
```

```
outlier from IQR method: [42.39999999999998, 42.5, 42.5, 42.5, 42.5, 42.5, 42.5, 42.5, 42.5, 42.59999999999991, 42.60000000000002,
upto - 983.5999999999996, 985.5, 1187.0, 2975.0, 3624.399999999998]
```



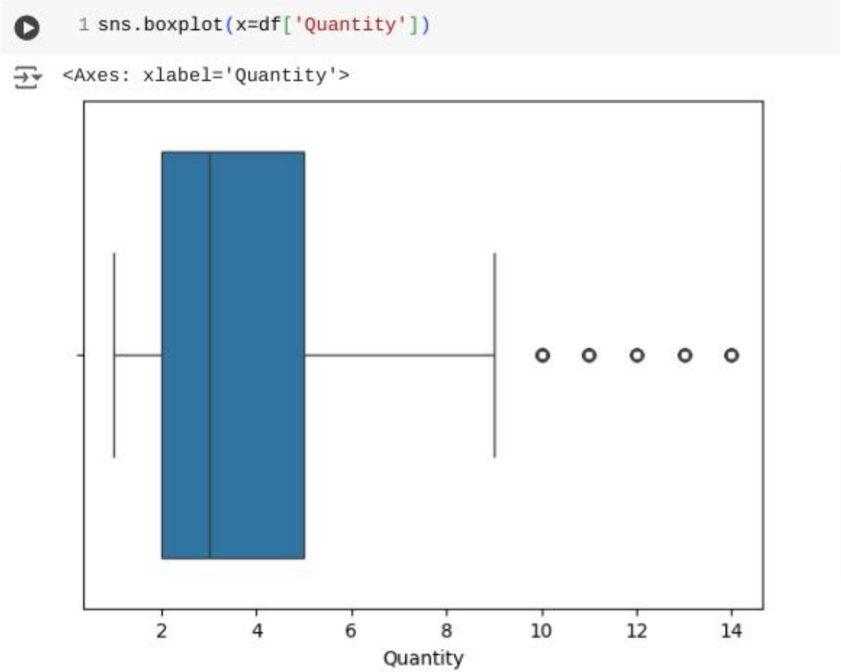
```
sns.boxplot(x=df['Selling_Price'])
```

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



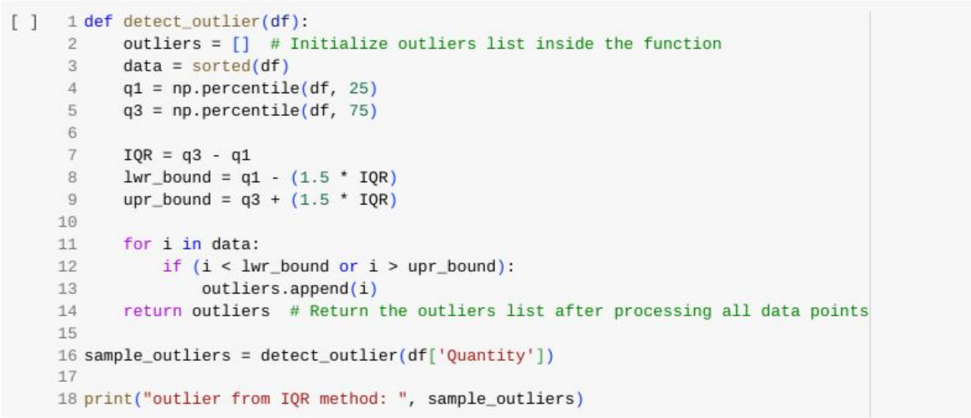
```
def detect_outlier(df):
    outliers = []  # Initialize outliers list inside the function
    data = sorted(df)
    q1 = np.percentile(df, 25)
    q3 = np.percentile(df, 75)

    IQR = q3 - q1
    lwr_bound = q1 - (1.5 * IQR)
    upr_bound = q3 + (1.5 * IQR)

    for i in data:
        if (i < lwr_bound or i > upr_bound):
            outliers.append(i)
    return outliers  # Return the outliers list after processing all data points

sample_outliers = detect_outlier(df['Selling_Price'])

print("outlier from IQR method: ", sample_outliers)
```

```
outlier from IQR method: [475.3, 475.3, 475.3, 475.3, 475.3, 475.3, upto - 9975.0, 10976.0, 13440.0,
16975.0, 21734.4
```



```python
sns.boxplot(x=df['Quantity'])
```

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



```python
def detect_outlier(df):
    outliers = []  # Initialize outliers list inside the function
    data = sorted(df)
    q1 = np.percentile(df, 25)
    q3 = np.percentile(df, 75)

    IQR = q3 - q1
    lwr_bound = q1 - (1.5 * IQR)
    upr_bound = q3 + (1.5 * IQR)

    for i in data:
        if (i < lwr_bound or i > upr_bound):
            outliers.append(i)
    return outliers  # Return the outliers list after processing all data points

sample_outliers = detect_outlier(df['Quantity'])

print("outlier from IQR method: ", sample_outliers)
```

# Insights:

```
outlier from IQR method quantity : start from [10, 10, 10, 10, 10, 10, 10, 10, up to 14, 14, 14, 14, 14,
14]
```