```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import plotly.express as px
import plotly.graph_objects as go
import plotly.io as pio
import plotly.colors as colors
pio.templates.default = "plotly_white"
```

```
import warnings
warnings.filterwarnings('ignore') #warning filters will get ignored
```

```
df=pd.read_csv('supp.csv')
```

```
df
```

	Ship Mode	Segment	City	State	Region	Category	Sub-Category	Sale
0	Second Class	Consumer	Henderson	Kentucky	South	Furniture	Bookcases	261.960
1	Second Class	Consumer	Henderson	Kentucky	South	Furniture	Chairs	731.940
2	Second Class	Corporate	Los Angeles	California	West	Office Supplies	Labels	14.620
3	Standard Class	Consumer	Fort Lauderdale	Florida	South	Furniture	Tables	957.577
4	Standard Class	Consumer	Fort Lauderdale	Florida	South	Office Supplies	Storage	22.368
...	...	...	...	...	...	...	...	.
9989	Second Class	Consumer	Miami	Florida	South	Furniture	Furnishings	25.248
9990	Standard Class	Consumer	Costa Mesa	California	West	Furniture	Furnishings	Na

```
df.head()
```

	Ship Mode	Segment	City	State	Region	Category	Sub-Category	Sales	Quantity	Discount	Profit
0	Second Class	Consumer	Henderson	Kentucky	South	Furniture	Bookcases	261.9600	2.0	0.00	41.9136
1	Second Class	Consumer	Henderson	Kentucky	South	Furniture	Chairs	731.9400	3.0	0.00	219.5820
2	Second Class	Corporate	Los Angeles	California	West	Office Supplies	Labels	14.6200	2.0	0.00	6.8714
3	Standard Class	Consumer	Fort Lauderdale	Florida	South	Furniture	Tables	957.5775	5.0	0.45	-383.0310
4	Standard Class	Consumer	Fort Lauderdale	Florida	South	Office Supplies	Storage	22.3680	2.0	0.20	2.5164

```
df.tail()
```

	Ship Mode	Segment	City	State	Region	Category	Sub-Category	Sales	Quantity	Discount	Profit
9989	Second Class	Consumer	Miami	Florida	South	Furniture	Furnishings	25.248	3.0	NaN	NaN
9990	Standard Class	Consumer	Costa Mesa	California	West	Furniture	Furnishings	NaN	2.0	0.0	15.6332
9991	Standard Class	Consumer	Costa Mesa	California	West	Technology	NaN	258.576	2.0	0.2	19.3932
9992	Standard Class	Consumer	Costa Mesa	California	West	Office Supplies	Paper	29.600	4.0	0.0	13.3200
9993	Second Class	Consumer	Westminster	California	West	Office Supplies	Appliances	243.160	2.0	0.0	72.9480

```
df.describe()
```

	Sales	Quantity	Discount	Profit
count	9959.000000	9963.000000	9962.000000	9958.000000
mean	229.908226	3.790826	0.156363	28.705040
std	623.747520	2.225585	0.206682	234.585082

```
#Displaying complete details of the dataset
print(df.to_string())
```

```
9936  Standard Class     Consumer         Cranston       Rhode Island     East  Office Supplies      Binders   102
9937    Second Class    Corporate      Los Angeles         California     West        Furniture       Tables    71
9938  Standard Class    Corporate    New York City           New York     East        Furniture  Furnishings    60
9939  Standard Class    Corporate    New York City           New York     East  Office Supplies     Supplies    35
9940  Standard Class    Corporate    New York City           New York     East  Office Supplies          Art    11
9941  Standard Class     Consumer    San Francisco         California     West       Technology  Accessories   223
9942  Standard Class     Consumer          Anaheim         California     West  Office Supplies      Storage   998
9943  Standard Class     Consumer          Anaheim         California     West  Office Supplies     Supplies    51
9944    Second Class  Home Office          Seattle         Washington     West  Office Supplies      Storage    40
9945  Standard Class    Corporate     Philadelphia       Pennsylvania     East  Office Supplies        Paper     3
9946  Standard Class    Corporate     Philadelphia       Pennsylvania     East       Technology  Accessories   151
9947    Second Class    Corporate     Indianapolis            Indiana  Central        Furniture       Chairs  1925
9948    Second Class    Corporate     Indianapolis            Indiana  Central  Office Supplies   Appliances  2405
9949    Second Class    Corporate     Indianapolis            Indiana  Central       Technology  Accessories    83
9950    Second Class    Corporate     Indianapolis            Indiana  Central       Technology  Accessories    39
9951    Second Class    Corporate     Indianapolis            Indiana  Central  Office Supplies      Binders    17
9952    Second Class    Corporate      Los Angeles         California     West  Office Supplies      Binders    55
9953    Second Class    Corporate      Los Angeles         California     West  Office Supplies        Paper     6
9954    Second Class    Corporate      Los Angeles         California     West  Office Supplies      Binders    34
9955    Second Class    Corporate      Los Angeles         California     West        Furniture       Tables   273
9956  Standard Class  Home Office     New Rochelle           New York     East  Office Supplies        Paper    46
9957  Standard Class  Home Office     New Rochelle           New York     East  Office Supplies        Paper   223
9958  Standard Class  Home Office     New Rochelle           New York     East  Office Supplies     Supplies     7
9959  Standard Class     Consumer         Chandler            Arizona     West  Office Supplies          Art     9
9960    Second Class  Home Office         Florence           Kentucky    South       Technology  Accessories    18
9961     First Class  Home Office          Houston              Texas  Central  Office Supplies        Paper    65
9962     First Class  Home Office          Houston              Texas  Central        Furniture    Bookcases   383
9963        Same Day     Consumer     Philadelphia       Pennsylvania     East  Office Supplies        Paper    10
9964    Second Class    Corporate           Newark           Delaware     East        Furniture  Furnishings    13
9965    Second Class    Corporate           Newark           Delaware     East  Office Supplies        Paper     4
9966    Second Class    Corporate           Newark           Delaware     East  Office Supplies    Envelopes   109
9967  Standard Class     Consumer       Plainfield         New Jersey     East  Office Supplies      Binders    40
9968  Standard Class     Consumer       Plainfield         New Jersey     East  Office Supplies      Binders   735
9969  Standard Class     Consumer       Plainfield         New Jersey     East  Office Supplies   Appliances    22
9970  Standard Class  Home Office           Smyrna            Georgia    South              NaN      Binders   119
9971  Standard Class  Home Office           Smyrna            Georgia    South  Office Supplies          Art   140
9972  Standard Class     Consumer          Houston              Texas  Central  Office Supplies    Envelopes    99
9973  Standard Class  Home Office      Los Angeles         California     West              NaN       Phones   271
9974  Standard Class  Home Office      Los Angeles         California      NaN  Office Supplies          Art    18
9975  Standard Class  Home Office      Los Angeles         California     West  Office Supplies        Paper    13
9976  Standard Class  Home Office      Los Angeles                NaN     West       Technology       Phones   249
9977  Standard Class  Home Office      Los Angeles         California     West  Office Supplies    Fasteners
9978  Standard Class  Home Office      Los Angeles         California     West  Office Supplies      Binders    13
9979  Standard Class  Home Office      Los Angeles         California     West  Office Supplies      Binders   437
9980    Second Class     Consumer        Lafayette                NaN    South        Furniture       Tables    85
9981     First Class     Consumer        Fairfield               Ohio     East  Office Supplies       Labels    16
9982  Standard Class     Consumer     Grand Rapids           Michigan  Central  Office Supplies        Paper
9983  Standard Class          NaN     Grand Rapids           Michigan  Central              NaN       Phones    97
9984  Standard Class     Consumer       Long Beach           New York     East  Office Supplies       Labels    31
9985  Standard Class     Consumer       Long Beach           New York     East  Office Supplies     Supplies    55
9986  Standard Class     Consumer              NaN         California     West       Technology          NaN    36
9987  Standard Class    Corporate           Athens            Georgia    South       Technology  Accessories    79
9988  Standard Class    Corporate           Athens            Georgia    South       Technology       Phones   206
9989    Second Class     Consumer            Miami            Florida    South        Furniture  Furnishings    25
9990  Standard Class     Consumer       Costa Mesa         California     West        Furniture  Furnishings
9991  Standard Class     Consumer       Costa Mesa         California     West       Technology          NaN   258
9992  Standard Class     Consumer       Costa Mesa         California     West  Office Supplies        Paper    29
9993    Second Class     Consumer      Westminster         California     West  Office Supplies   Appliances   243
```

```
#Displaying the name of columns
df.columns
```

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

```
#Displaying the no. of rows & columns
df.shape
```

```
(9994, 11)
```

```
#Displaying the datatypes of all the columns
df.dtypes
```

```
Ship Mode       object
Segment         object
```

```
City              object
State             object
Region            object
Category          object
Sub-Category      object
Sales            float64
Quantity         float64
Discount         float64
Profit           float64
dtype: object
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9994 entries, 0 to 9993
Data columns (total 11 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   Ship Mode     9994 non-null   object
 1   Segment       9989 non-null   object
 2   City          9975 non-null   object
 3   State         9964 non-null   object
 4   Region        9974 non-null   object
 5   Category      9960 non-null   object
 6   Sub-Category  9974 non-null   object
 7   Sales         9959 non-null   float64
 8   Quantity      9963 non-null   float64
 9   Discount      9962 non-null   float64
 10  Profit        9958 non-null   float64
dtypes: float64(4), object(7)
memory usage: 859.0+ KB
```

```
#checking null values
df.isna().sum()
```

```
Ship Mode        0
Segment          5
City            19
State           30
Region          20
Category        34
Sub-Category    20
Sales           35
Quantity        31
Discount        32
Profit          36
dtype: int64
```

```
df.duplicated().sum()
```

```
47
```

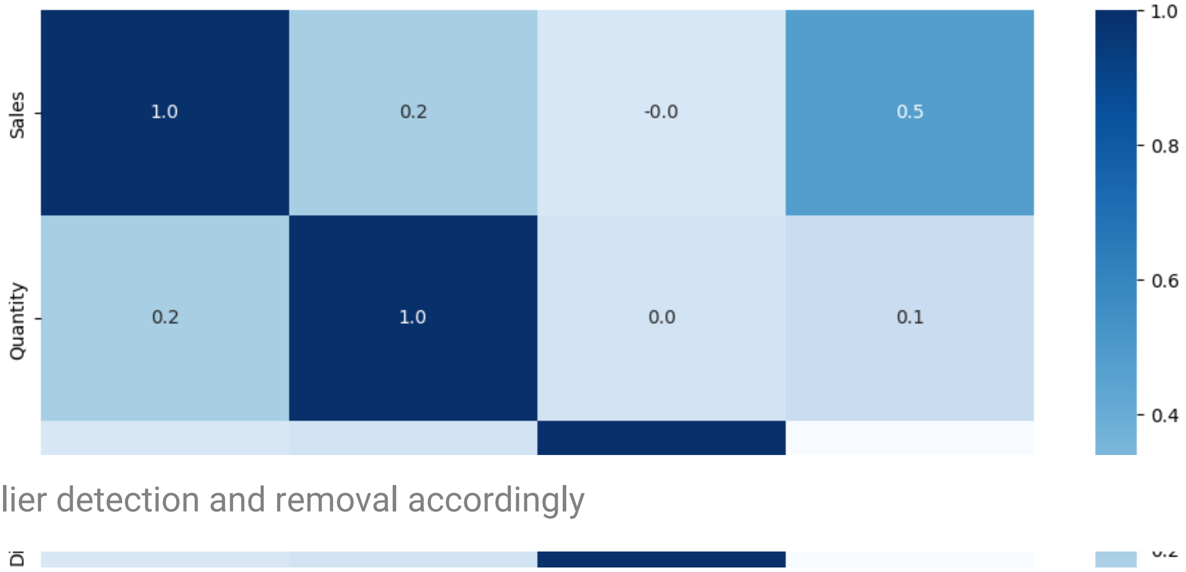
```
#Removing duplicated rows to avoid faults in further calculation
df.drop_duplicates(inplace=True)
```

```
#After removing duplicate entries
df.shape
```

```
(9947, 11)
```

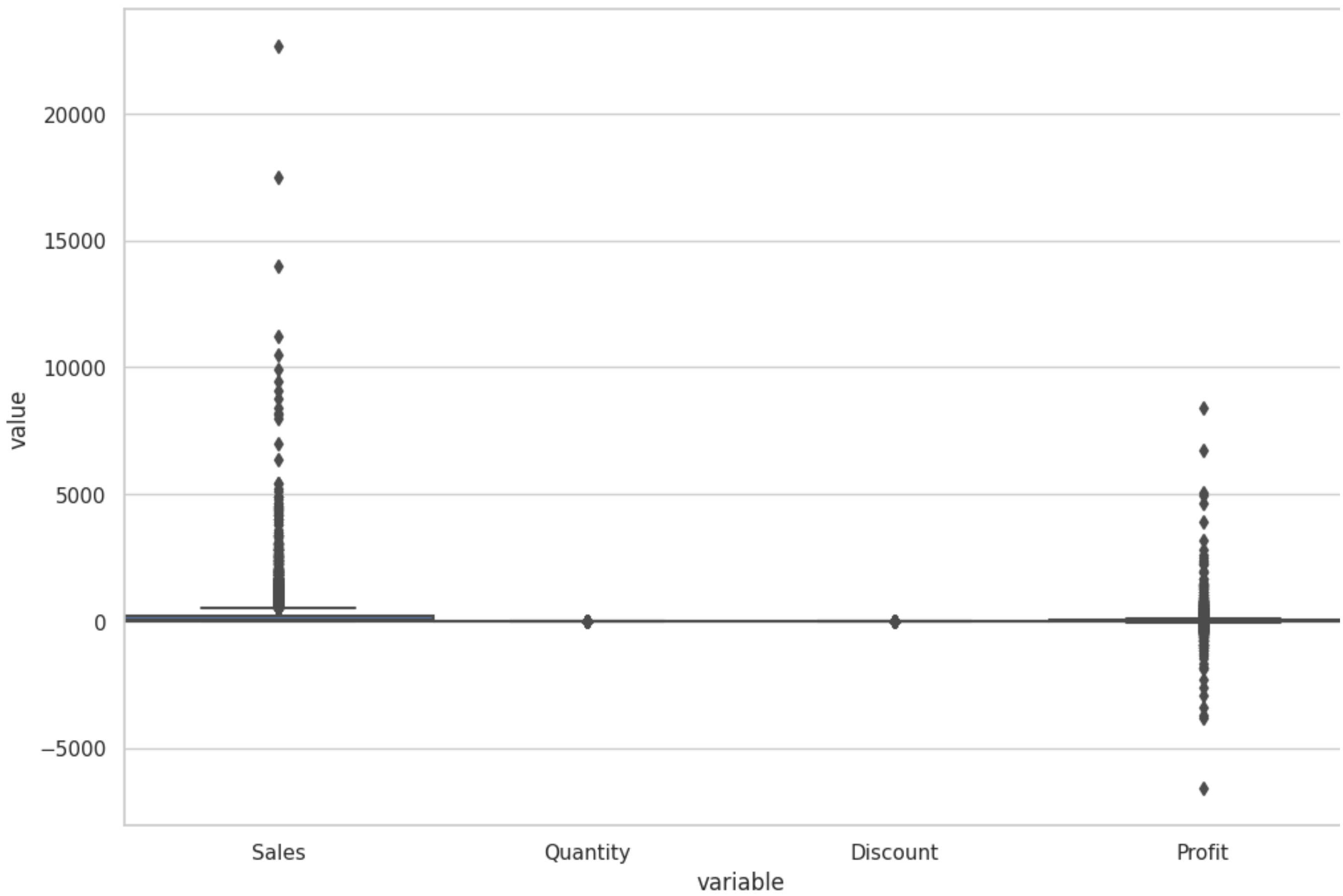
```
#Checking the corelations between numeric columns
df_con=df.select_dtypes(include=[np.number]) #getting the numerical features
f,ax = plt.subplots(figsize=(12, 8))
sns.heatmap(df_con.corr(method='pearson'), annot=True, fmt= '.1f',ax=ax, cmap="Blues") #plotting a heatmap
```

```
<Axes: >
```



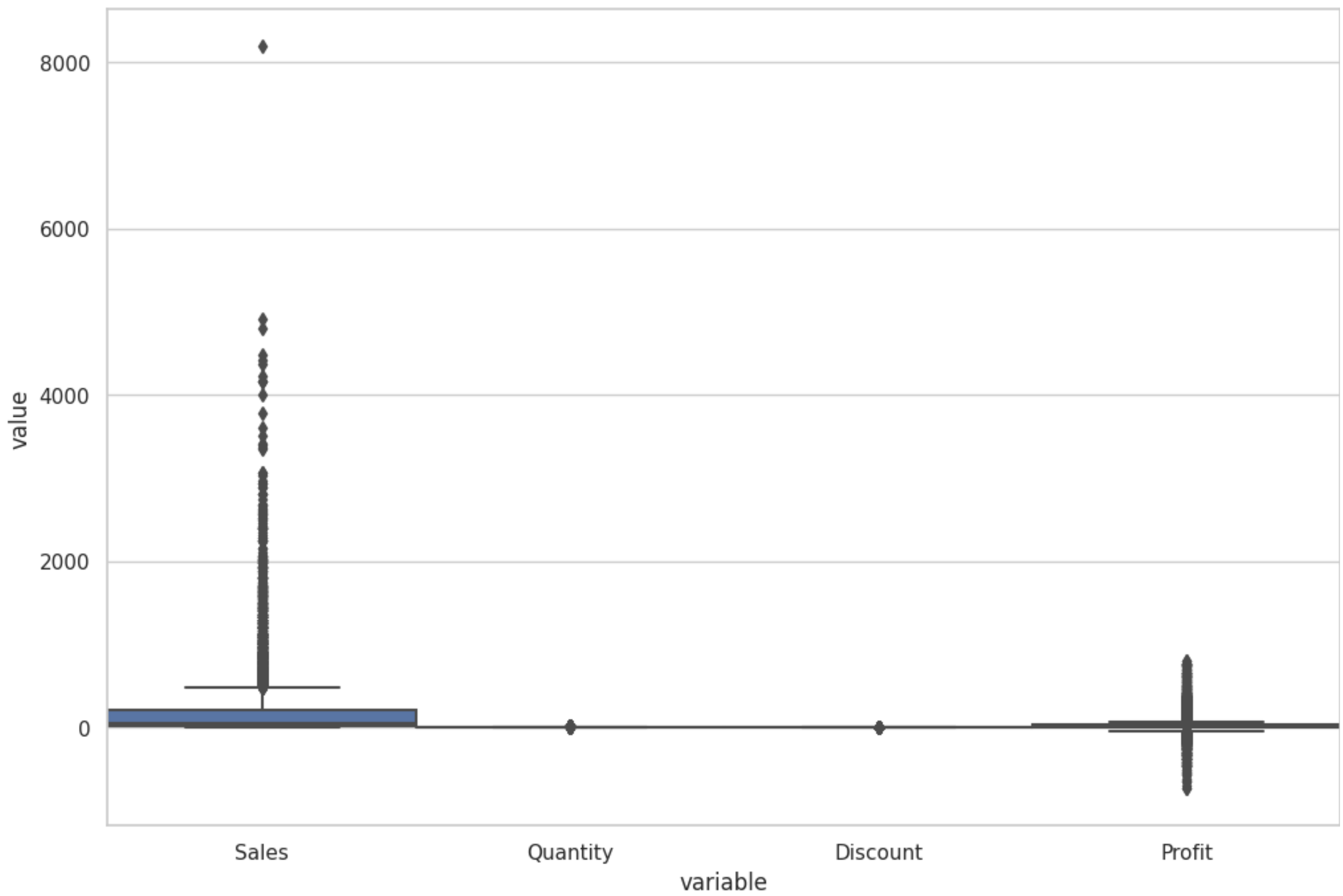
Outlier detection and removal accordingly

```
#BoxPlot to see the outliers clearly
plt.figure(figsize=[12,8])
sns.set(style="whitegrid")
sns.boxplot(x="variable", y="value", data=pd.melt(df_con), width=1)

plt.show()
```




```
#Removal of Outliers (Profit)
def remove_outlier(dataset,k=3.33):
    for col in dataset.columns:
            if (dataset[col].dtype=="int64" or dataset[col].dtype=="float64"):
                mean = dataset[col].mean()
                global ds
                std = dataset[col].std()
                outlier = [i for i in dataset[col] if (i > mean - k * std)]
                outlier = [i for i in outlier if (i < mean + k * std)]
                ds = dataset.loc[dataset[col].isin(outlier)]


remove_outlier(df,k=3.33)
```

```
#Let's see the Profit outliers are removed or not
ds_con=ds.select_dtypes(include=[np.number])
plt.figure(figsize=[12,8])
sns.set(style="whitegrid")
sns.boxplot(x="variable", y="value", data=pd.melt(ds_con), width=1)

plt.show()
```

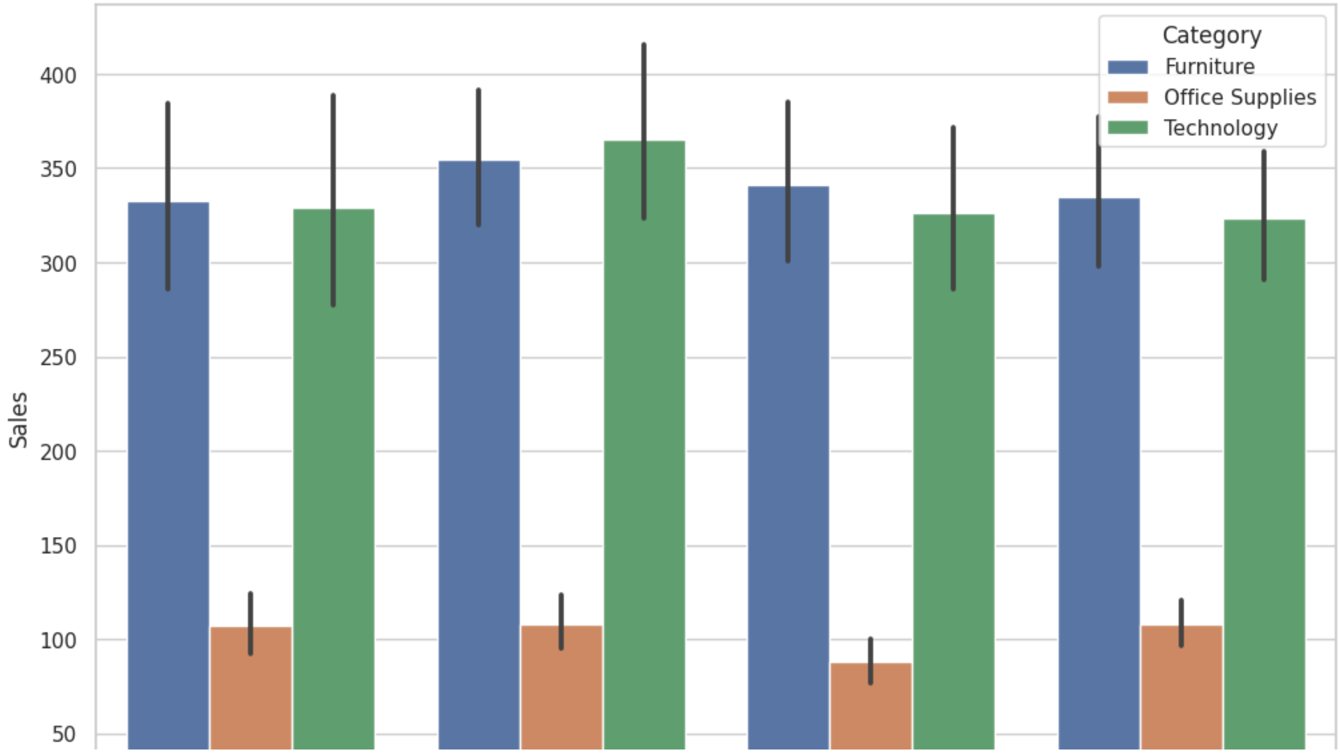


Exploratory Analysis and Visualization

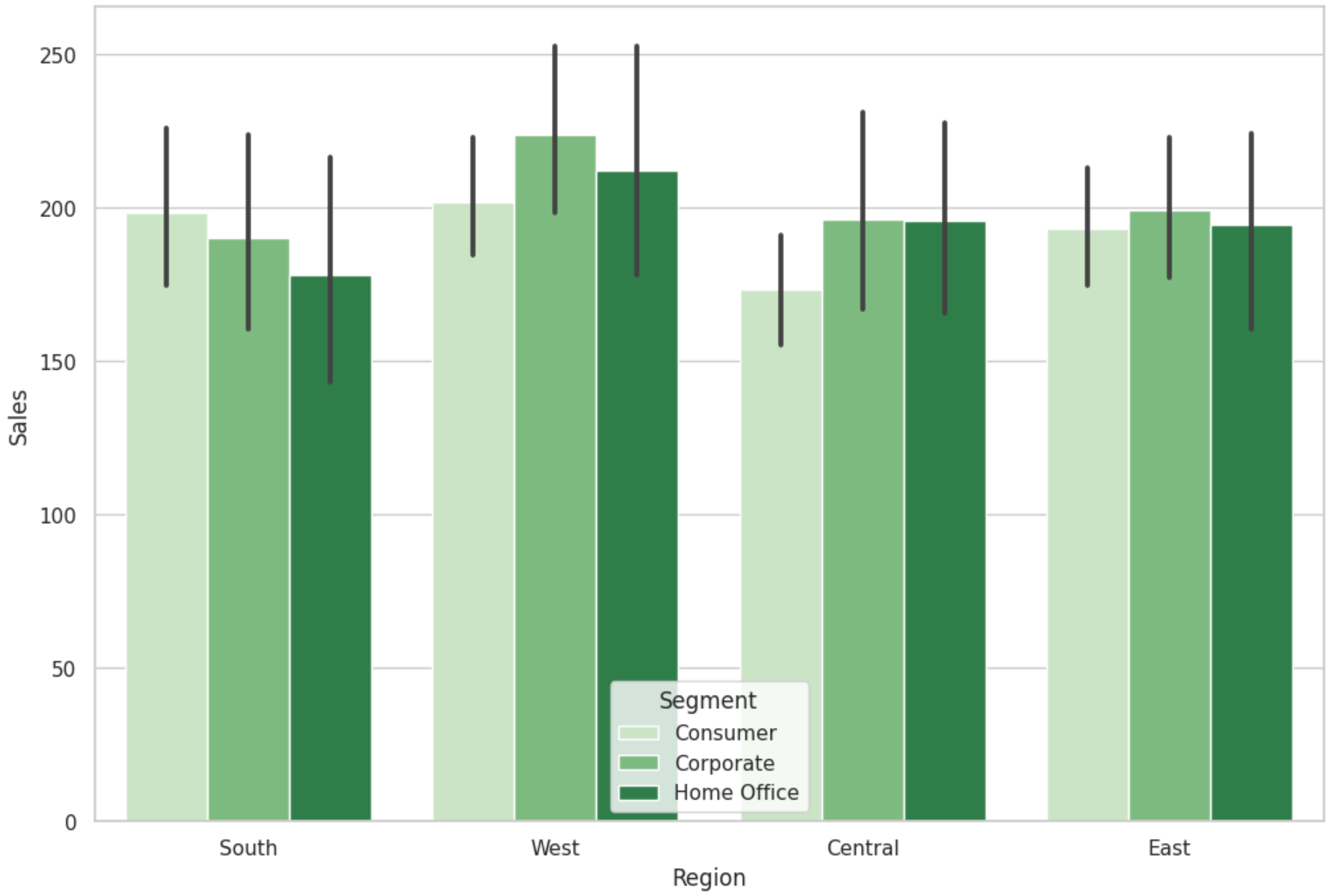
```
#numbers of unique entries in the Categorical columns
for col in ds.columns:
    if ds[col].dtype=='object':
        print("Number of unique entries in",col + " are",ds[col].nunique())
        print("================================================")
```

```
Number of unique entries in Ship Mode are 4
================================================
Number of unique entries in Segment are 3
================================================
Number of unique entries in City are 530
================================================
Number of unique entries in State are 49
================================================
Number of unique entries in Region are 4
================================================
Number of unique entries in Category are 3
================================================
Number of unique entries in Sub-Category are 17
================================================
```

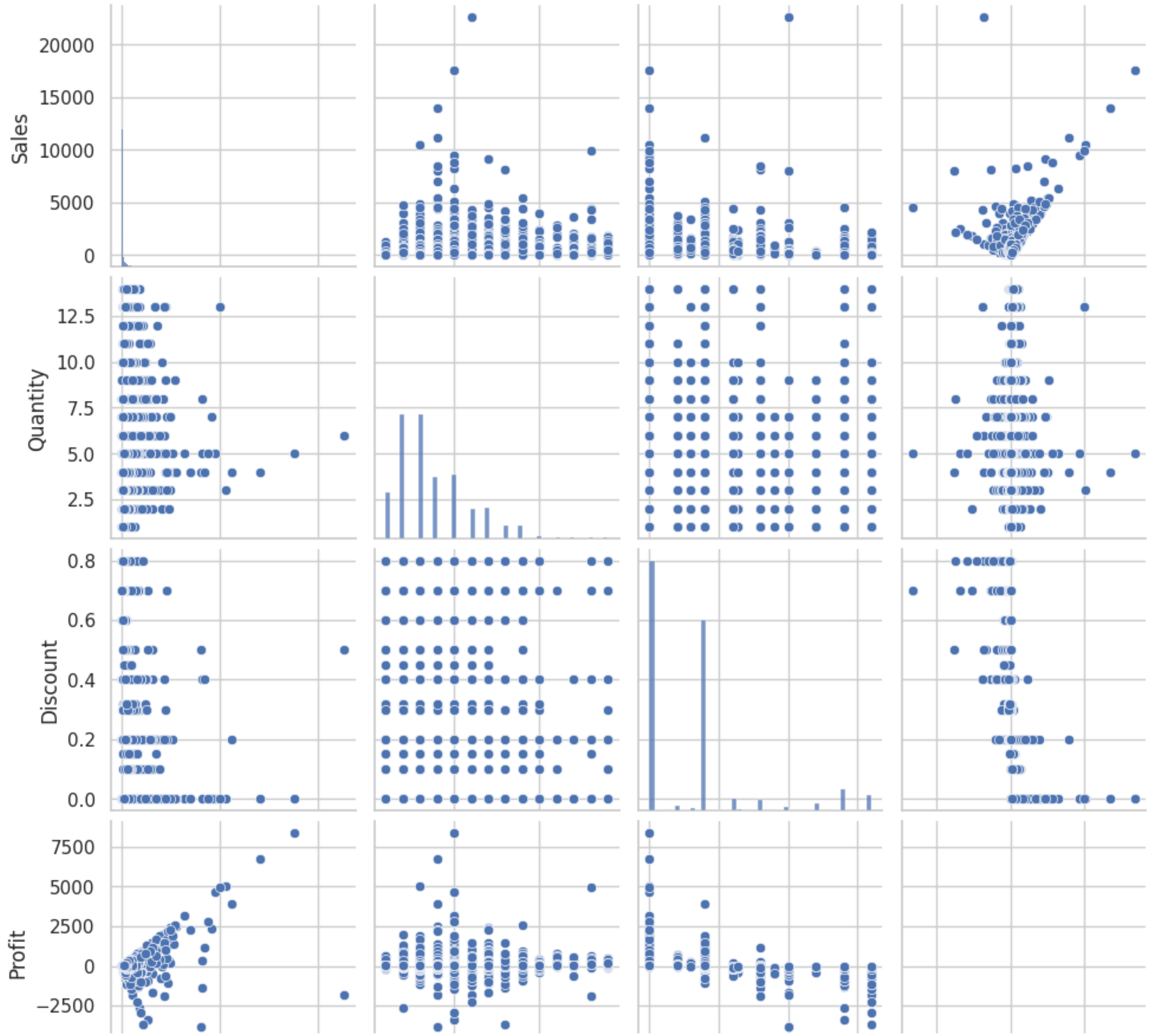
```
#Category wise sales in each region
plt.figure(figsize=[12,8])
ax = sns.barplot(x="Region", y="Sales", hue="Category", data=ds)
```




```
#Segment wise sales in each region
plt.figure(figsize=[12,8])
ax = sns.barplot(x="Region", y="Sales", hue="Segment", data=ds, palette="Greens")
```




```
#some aggregated views from pairplot
sns.pairplot(df)
plt.show()
```




```
#some insights based on Cities
grouped= ds.groupby("City")
#Aggregated Sales per city
agg_sales=grouped['Sales'].agg(np.sum).sort_values(ascending=False).reset_index()
#Cities with highest total sales
agg_sales.head()
```

	City	Sales
0	New York City	193174.057
1	Los Angeles	162836.213
2	San Francisco	107922.747
3	Seattle	95640.724
4	Philadelphia	91635.577

```
#Aggregated Profit per city
agg_profit=grouped['Profit'].agg(np.sum).sort_values(ascending=False).reset_index()
#Cities with Highest total Profit
agg_profit.head()
```

	City	Profit
0	New York City	39931.8712
1	Los Angeles	26344.2100
2	Seattle	18905.1617
3	San Francisco	16375.4373
4	Detroit	7953.2432

```
#Aggregates Discount per city
agg_dist=grouped['Discount'].agg(np.sum).sort_values(ascending=False).reset_index()

#Cities with highest aggregated Discount
agg_dist.head()
```

	City	Discount
0	Philadelphia	171.90
1	Houston	137.54
2	Chicago	115.30
3	Dallas	55.30
4	Los Angeles	53.50

```
#Average Sales per city
avg_sales=grouped['Sales'].agg(np.mean).sort_values(ascending=False).reset_index()
#Cities with highest Average sales
avg_sales.head()
```

	City	Sales
0	Cheyenne	1603.136000
1	Bellingham	1263.413333
2	Independence	1208.685000
3	Burbank	1082.386000
4	Buffalo	906.349600

```
#Cities with lowest Average sales
avg_sales.tail()
#Average Profit per city
avg_profit=grouped['Profit'].agg(np.mean).sort_values(ascending=False).reset_index()
#Cities with highest Average profit
avg_profit.head()
```

	City	Profit
0	Independence	487.831500
1	Appleton	277.383150
2	Burbank	254.844600
3	Lehi	225.831300
4	Beverly	218.306467

```
#Cities with lowest Average profit
avg_profit.tail()
```

	City	Profit
525	Rockford	-104.500709
526	Normal	-110.023200
527	Yuma	-116.497725
528	Oswego	-178.709200
529	Champaign	-182.352000

```
#Average Discount per city
avg_dist=grouped['Discount'].agg(np.mean).sort_values(ascending=False).reset_index()
#Cities with highest Average discount
avg_dist.head()
```

```
#Cities with lowest Average Discount
avg_dist.tail()
```

	City	Discount
525	Cedar Rapids	0.0
526	Paterson	0.0
527	Belleville	0.0
528	Perth Amboy	0.0
529	Aberdeen	0.0

```
#Cities having High Average Discounts
high_dist=avg_dist[avg_dist['Discount'] >=0.7]
high_dist
```

	City	Discount
0	Deer Park	0.8
1	Romeoville	0.8
2	Missouri City	0.8
3	Abilene	0.8
4	Littleton	0.7
5	Elyria	0.7
6	Ormond Beach	0.7

```
#Cities having low/no Average Discounts
low_dist=avg_dist[avg_dist['Discount']==0]
low_dist
```

	City	Discount
345	Waynesboro	0.0
346	Wichita	0.0
347	Fargo	0.0
348	Elkhart	0.0
349	Cottage Grove	0.0
...	...	...
525	Cedar Rapids	0.0
526	Paterson	0.0
527	Belleville	0.0
528	Perth Amboy	0.0
529	Aberdeen	0.0

185 rows × 2 columns

```
#Cities having High Average Sales
high_sales=avg_sales[avg_sales['Sales']>500]
high_sales
```

	City	Sales
0	Cheyenne	1603.136000
1	Bellingham	1263.413333
2	Independence	1208.685000
3	Burbank	1082.386000
4	Buffalo	906.349600
5	Beverly	861.063333
6	Sparks	853.986667
7	Appleton	835.655000
8	Torrance	783.067000
9	Noblesville	772.795000
10	Lehi	758.363000
11	Kissimmee	751.984000
12	Saint Peters	697.160000

```
#Cities having low Average Sales
low_sales=avg_sales[avg_sales['Sales']<50]
low_sales
```

	City	Sales
435	Urbandale	49.706667
436	Woonsocket	48.887500
437	Frankfort	48.816000
438	Thousand Oaks	47.760800
439	The Colony	47.386667
...	...	...
525	San Luis Obispo	3.620000
526	Ormond Beach	2.808000
527	Jupiter	2.064000
528	Elyria	1.824000
529	Abilene	1.392000

95 rows × 2 columns

```
#Cities having High Average Profit
high_profit=avg_profit[avg_profit['Profit']>100]
high_profit
```

	City	Profit
0	Independence	487.831500
1	Appleton	277.383150
2	Burbank	254.844600
3	Lehi	225.831300
4	Beverly	218.306467
5	Bellingham	203.530267
6	Morristown	165.842750
7	Dubuque	159.224800
8	Saint Cloud	156.538000
9	Missoula	152.495000

```
#Cities having low Average profit
low_profit=avg_profit[avg_profit['Profit']<0]
low_profit
```

	City	Profit
421	Austin	-0.522918
422	Hickory	-0.547800
423	Altoona	-0.591750
424	Bolingbrook	-0.776833
425	Elyria	-1.398400
...	...	...
525	Rockford	-104.500709
526	Normal	-110.023200
527	Yuma	-116.497725
528	Oswego	-178.709200
529	Champaign	-182.352000

109 rows × 2 columns

```
#Cities with High-Average-Discounts but Low-Average-Sales
merged= pd.merge(high_dist,low_sales, on=['City'],how='inner')
merged
```

	City	Discount	Sales
0	Deer Park	0.8	6.924
1	Romeoville	0.8	8.952
2	Missouri City	0.8	6.370
3	Abilene	0.8	1.392
4	Elyria	0.7	1.824
5	Ormond Beach	0.7	2.808

```
#Cities with high Average Sales as well as Average Profit
merged2= pd.merge(high_sales,high_profit, on=['City'], how='inner')
merged2
```

	City	Sales	Profit
0	Cheyenne	1603.136000	100.196000
1	Bellingham	1263.413333	203.530267
2	Independence	1208.685000	487.831500
3	Burbank	1082.386000	254.844600
4	Beverly	861.063333	218.306467
5	Appleton	835.655000	277.383150

```
#Cities where Average Discount is less but Average Sales is High
merged3= pd.merge(low_dist,high_sales, on='City', how='inner')
merged3
```

	City	Discount	Sales
0	Saint Peters	0.0	697.160000
1	Dubuque	0.0	562.433333
2	Appleton	0.0	835.655000
3	Morristown	0.0	539.853333
4	Madison	0.0	534.679000
5	Harrisonburg	0.0	626.958571
6	Independence	0.0	1208.685000
7	Noblesville	0.0	772.795000
8	Norman	0.0	675.665000
9	Beverly	0.0	861.063333

```
#Cities with high Average sales but low Average profit
merged4= pd.merge(high_sales,low_profit, on='City', how='inner')
merged4
```

	City	Sales	Profit
0	Richardson	644.232	-12.24465

```
#Cities with high Average discount but low Average profit
merged5= pd.merge(high_dist,low_profit, on='City', how='inner')
merged5
```

	City	Discount	Profit
0	Deer Park	0.8	-10.3860
1	Romeoville	0.8	-14.7708
2	Missouri City	0.8	-9.5550
3	Abilene	0.8	-3.7584
4	Littleton	0.7	-98.8018
5	Elyria	0.7	-1.3984
6	Ormond Beach	0.7	-1.9656

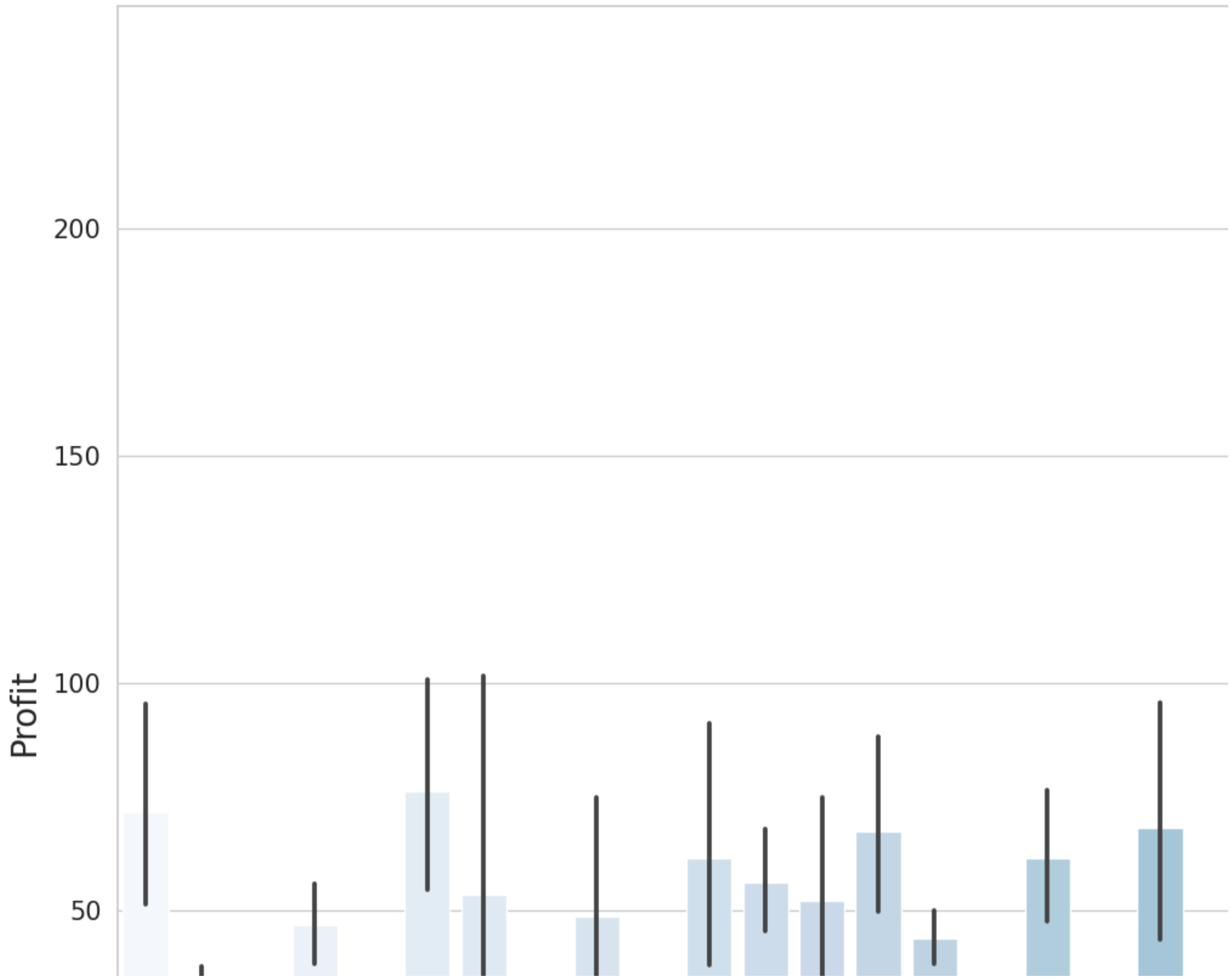
```
#Cities with low Average discount but High Average profit
merged6= pd.merge(low_dist, high_profit, on='City', how='inner')
merged6
```

	City	Discount	Profit
0	Saint Cloud	0.0	156.538000
1	Saint Peters	0.0	146.403600
2	Dubuque	0.0	159.224800
3	Washington	0.0	105.958930
4	Vacaville	0.0	110.052800
5	Edmond	0.0	121.551950
6	Appleton	0.0	277.383150
7	Kenosha	0.0	114.230311
8	Morristown	0.0	165.842750
9	Muskogee	0.0	110.649150

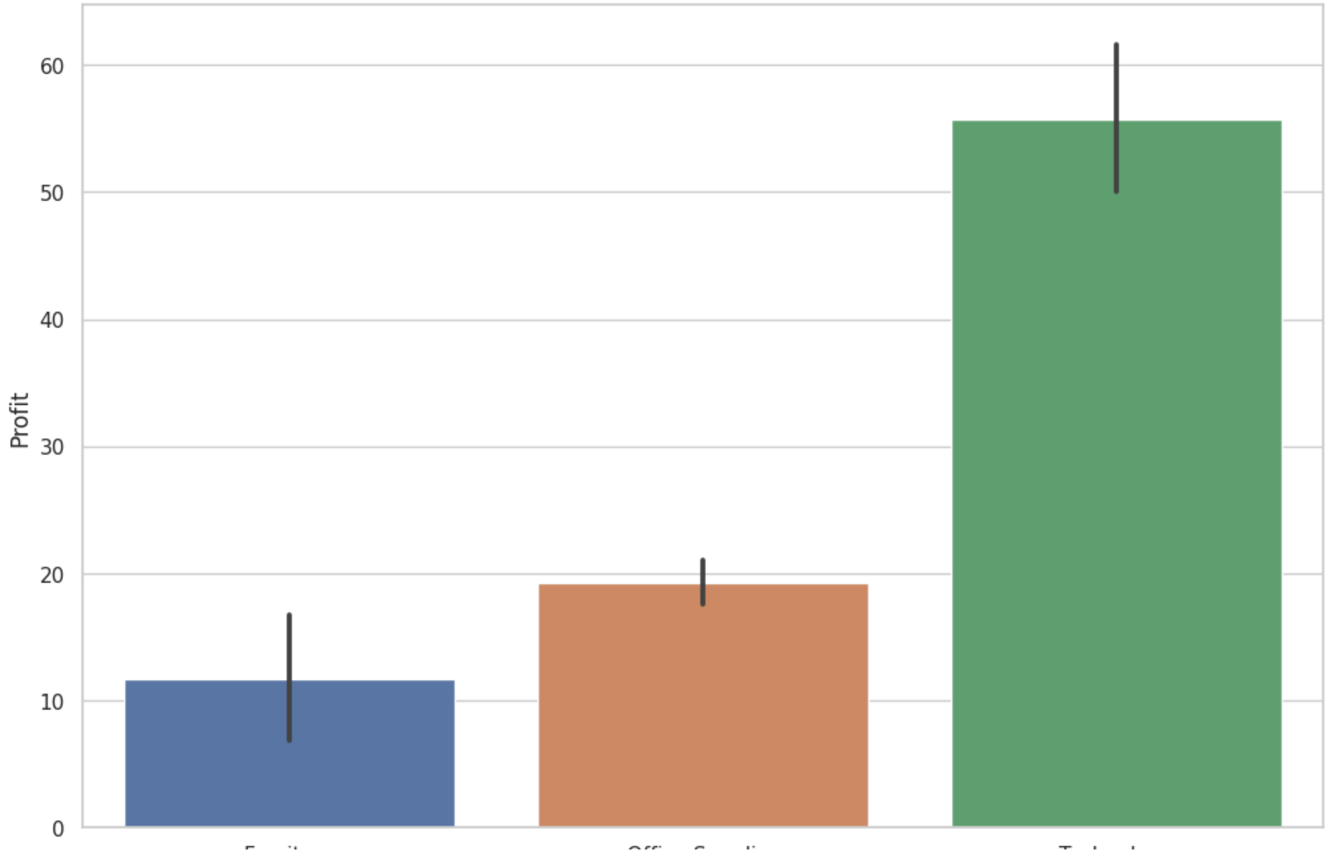
Visuals with profit

11	Broken Arrow	0.0	115.104520

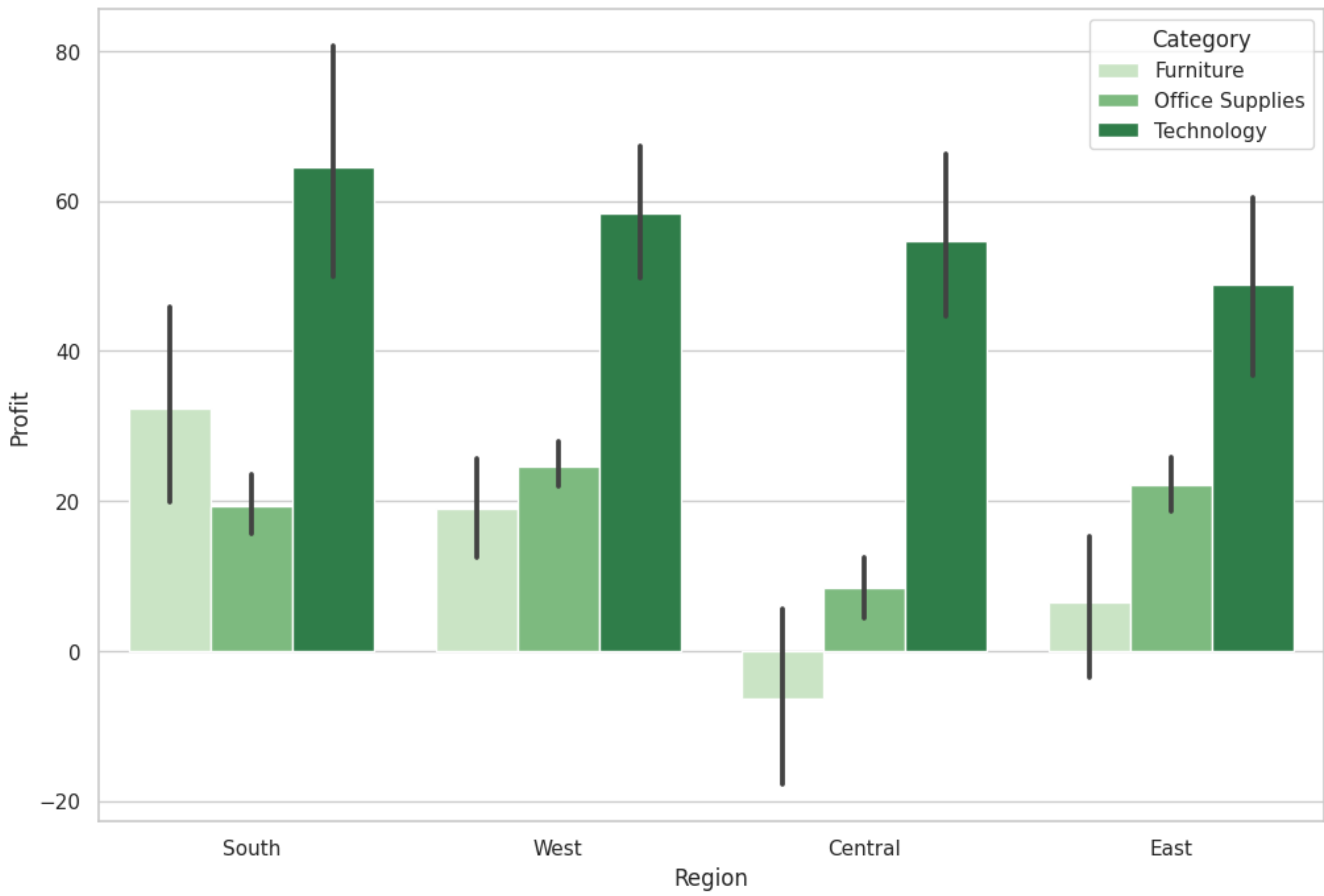
```
plt.figure(figsize=[24,15])
ax = sns.barplot(x="State", y="Profit", data=ds, palette="Blues")
plt.xticks(rotation=90, fontsize=16)
plt.yticks(fontsize=15)
plt.title("States VS Profit",fontsize=24)
plt.xlabel("States",fontsize=20)
plt.ylabel("Profit",fontsize=20)
plt.tight_layout()
```




```
#Category wise profit in the whole country
plt.figure(figsize=[12,8])
ax = sns.barplot(x="Category", y="Profit", data=ds)
```

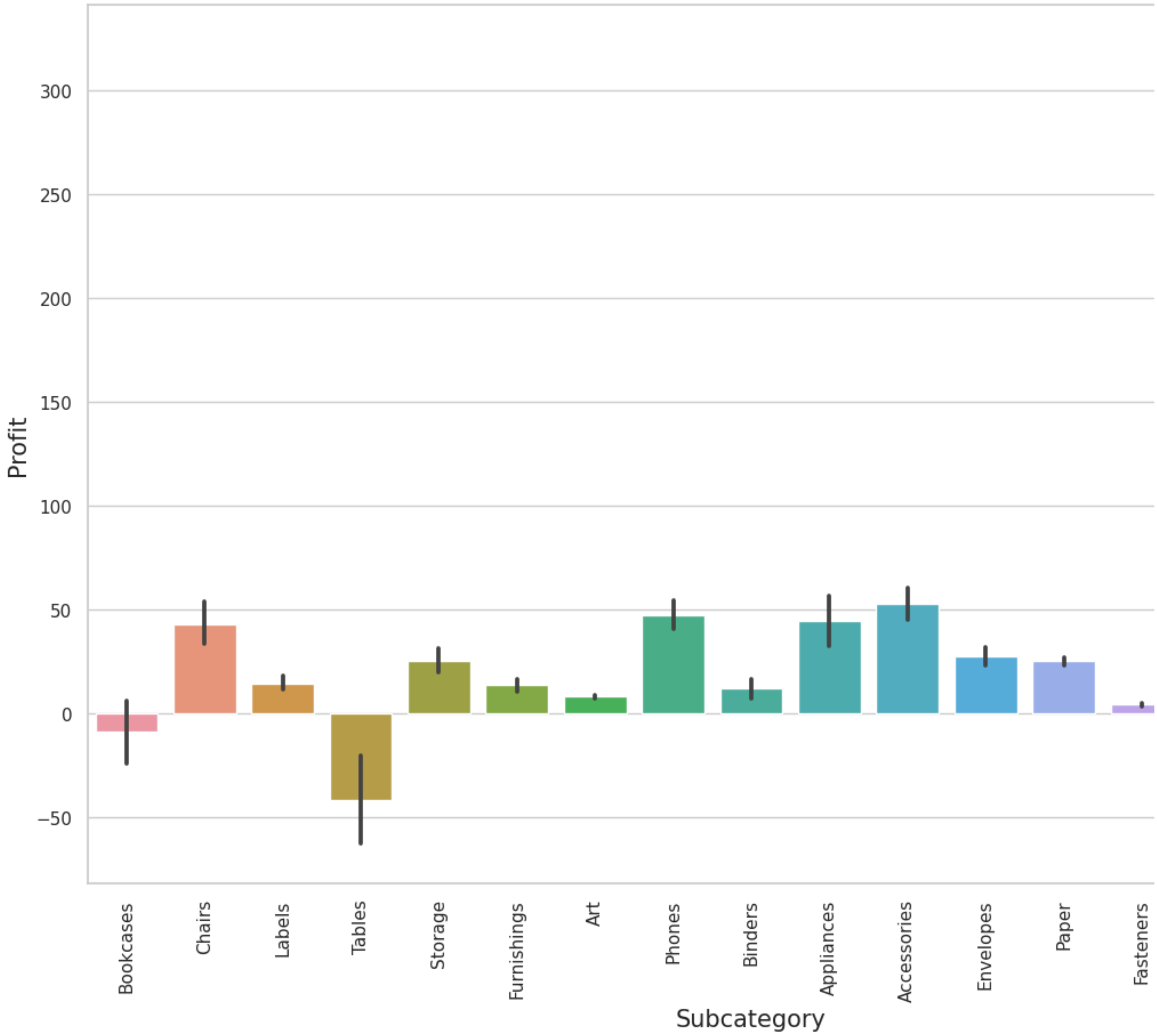



```
#Category wise Profit in Each Region
plt.figure(figsize=[12,8])
ax = sns.barplot(x="Region", y="Profit", hue="Category", data=ds, palette="Greens")
```




```
#Subcategory wise profit
plt.figure(figsize=[15,10])
ax = sns.barplot(x="Sub-Category", y="Profit", data=ds)
plt.xlabel("Subcategory", fontsize=15)
plt.ylabel("Profit",fontsize=15)
```

```
plt.xticks(rotation=90)
plt.show()
```

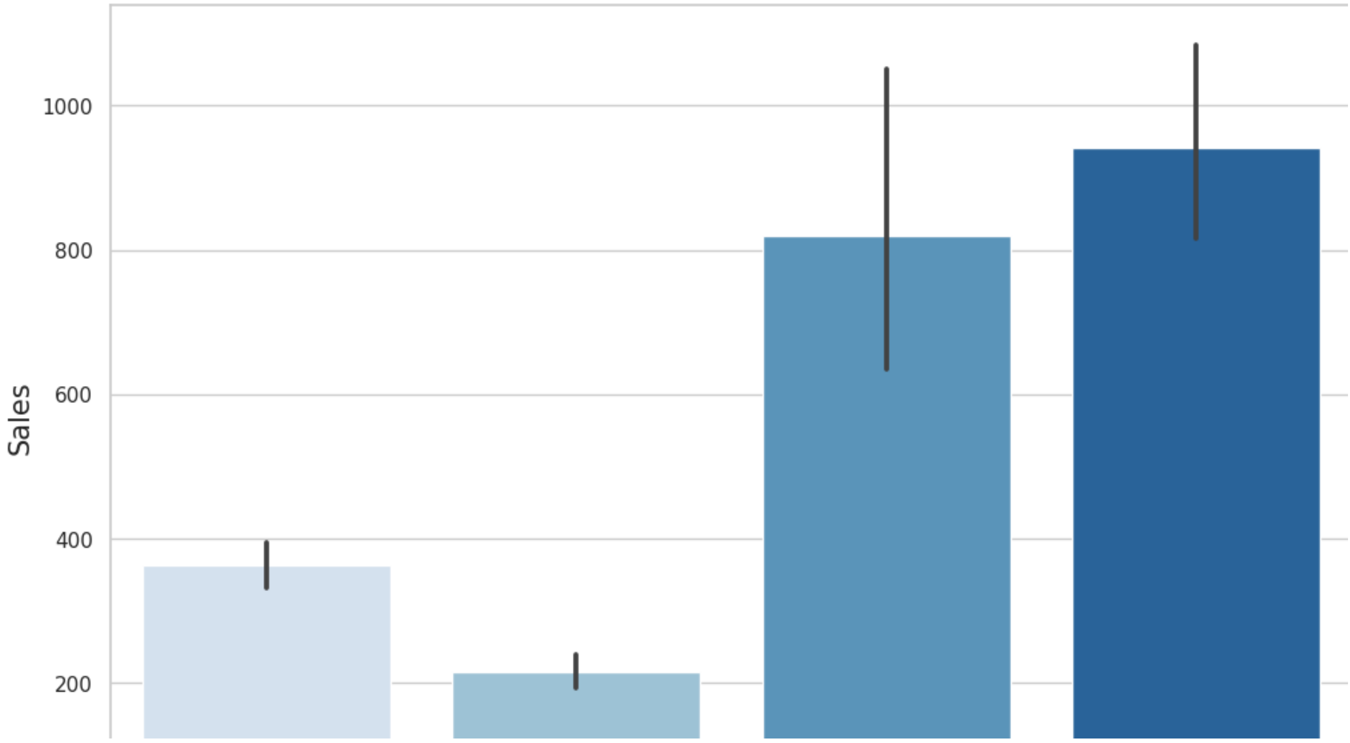



```
#Entries with Category=Technology
ds_tech=ds[(ds['Category']=="Technology")]
ds_tech.head()
```

	Ship Mode	Segment	City	State	Region	Category	Sub-Category	Sales	Quantity	Discount	Profit
7	Standard Class	Consumer	Los Angeles	California	West	Technology	Phones	907.152	6.0	0.2	90.7152
11	Standard Class	Consumer	NaN	California	West	Technology	Phones	NaN	4.0	0.2	68.3568
19	Second Class	Consumer	San Francisco	California	West	Technology	Phones	NaN	3.0	0.2	16.0110
26	Second Class	Consumer	Los Angeles	California	West	Technology	Accessories	90.570	3.0	0.0	11.7741
35	First Class	Corporate	Richardson	Texas	Central	Technology	Phones	1097.544	7.0	0.2	123.4737

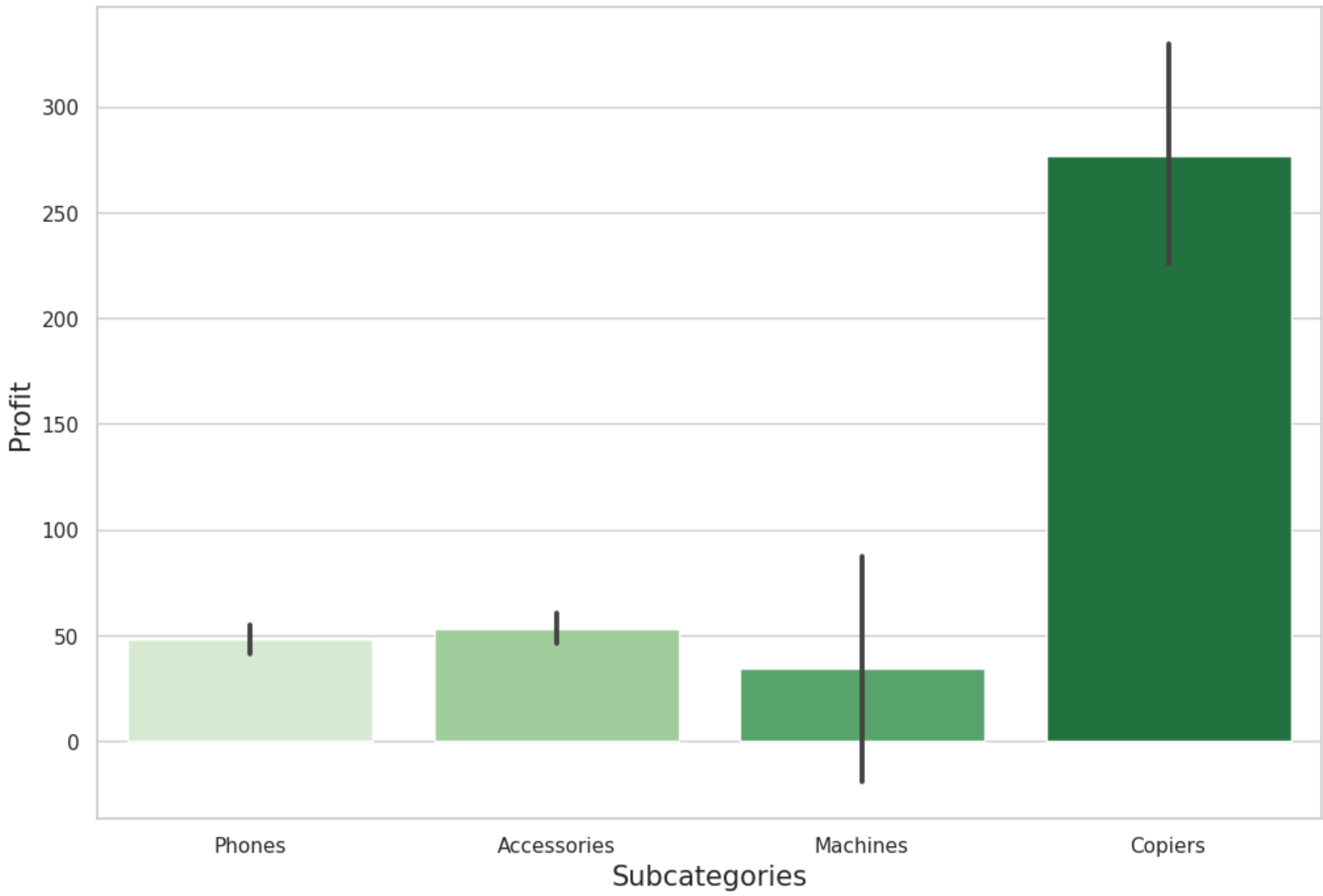
```
#Sales of each Subcategory under Technology
plt.figure(figsize=[12,8])
ax = sns.barplot(x="Sub-Category", y="Sales", data=ds_tech, palette="Blues")
plt.xlabel("Subcategories",fontsize=15)
plt.ylabel("Sales",fontsize=15)
```

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



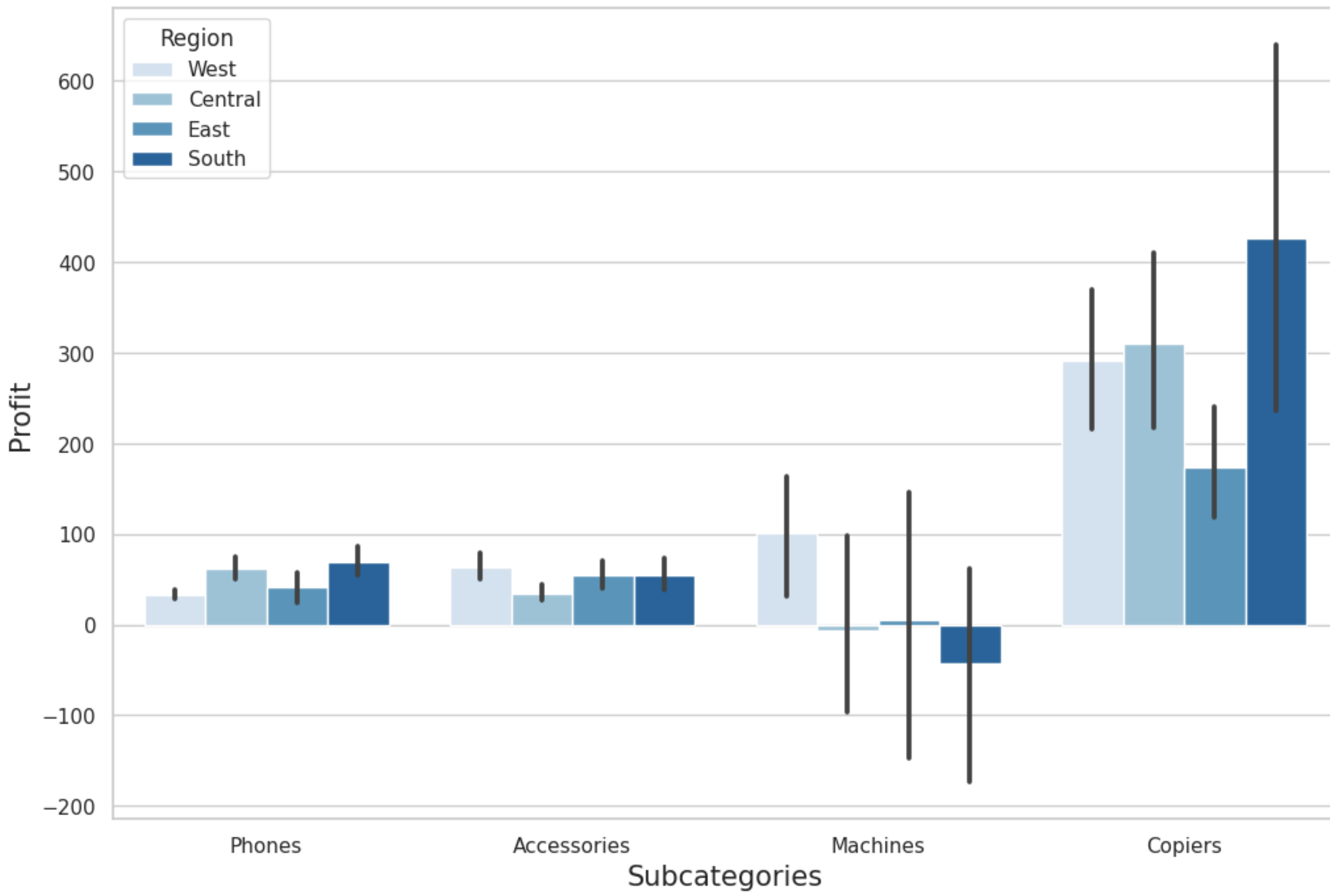

```
#Profit of each Subcategory under Technology
plt.figure(figsize=[12,8])
ax = sns.barplot(x="Sub-Category", y="Profit", data=ds_tech, palette="Greens")
plt.xlabel("Subcategories",fontsize=15)
plt.ylabel("Profit",fontsize=15)
```

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




```
#Profit of each Subcategory under Technology for each Region
plt.figure(figsize=[12,8])
ax = sns.barplot(x="Sub-Category", y="Profit",hue="Region", data=ds_tech, palette="Blues")
plt.xlabel("Subcategories",fontsize=15)
plt.ylabel("Profit",fontsize=15)
```

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




```
#checking skewness of Profit
df['Profit'].skew()
```

```
7.538715203527991
```

```
# Checking Skewness from 'Ship Mode' to 'Profit'

df_filtered_univar=df.loc[:,'Ship Mode':'Profit']
df_filtered_univar=df_filtered_univar.select_dtypes([np.int, np.float])
for i, col in enumerate(df_filtered_univar.columns):
    print("\nSkewness of "+col +" is", df_filtered_univar[col].skew()) #measures skewness
```

```
Skewness of Sales is 12.956466331241966

Skewness of Quantity is 1.2752018630542497

Skewness of Discount is 1.679624000034756

Skewness of Profit is 7.538715203527991
```

```
# Replace the null values with median

df['Sales'].fillna(df['Sales'].median(),inplace=True)
df['Quantity'].fillna(df['Quantity'].median(),inplace=True)
df['Discount'].fillna(df['Discount'].median(),inplace=True)
df['Profit'].fillna(df['Profit'].median(),inplace=True)
```

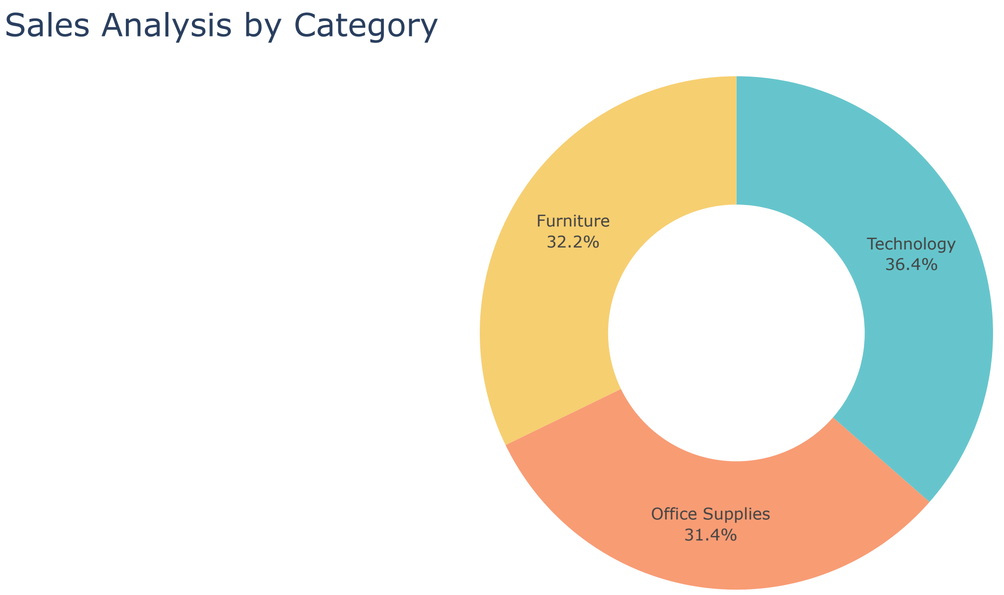
```
# Check unique values
df["Sub-Category"].unique()
```

```
array(['Bookcases', 'Chairs', 'Labels', 'Tables', 'Storage',
       'Furnishings', 'Art', 'Phones', 'Binders', 'Appliances', 'Paper',
       'Accessories', 'Envelopes', 'Fasteners', 'Supplies', 'Machines',
       nan, 'Copiers'], dtype=object)
```

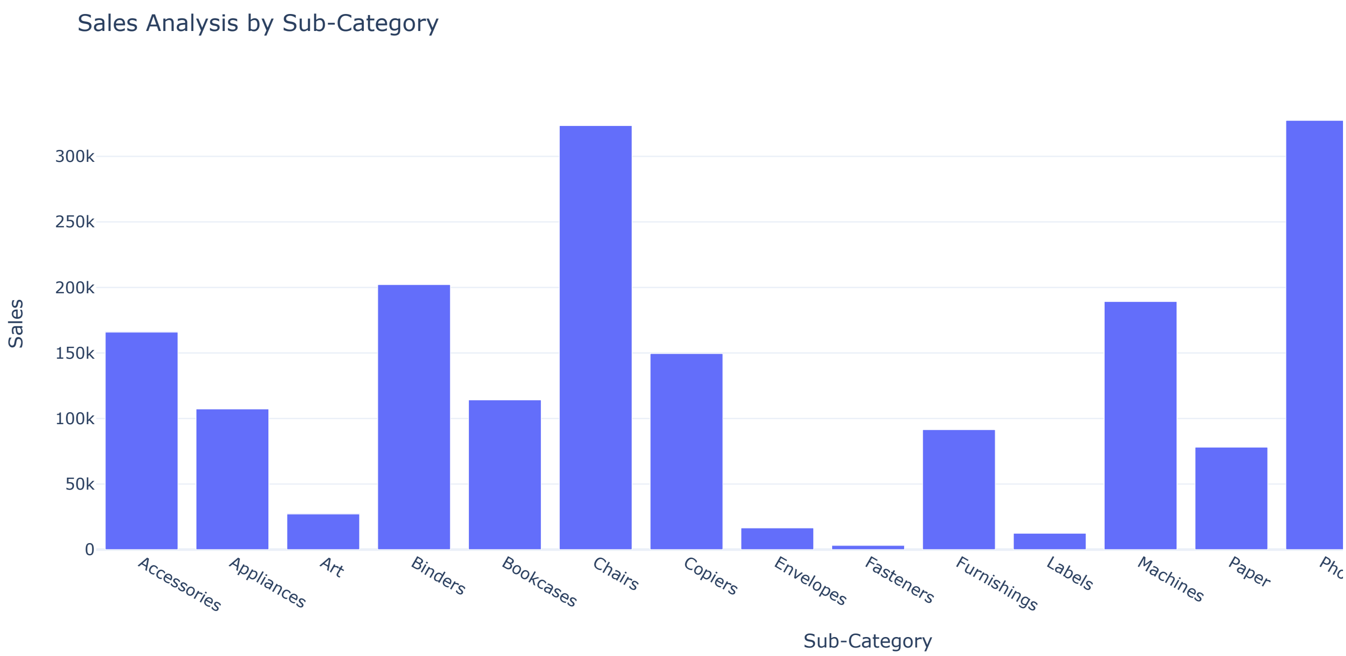
Sub-category sales analysis

```
sales_by_category = df.groupby('Category')['Sales'].sum().reset_index()
```

```
fig = px.pie(sales_by_category,
             values='Sales',
             names='Category',
             hole=0.5,
             color_discrete_sequence=px.colors.qualitative.Pastel)

fig.update_traces(textposition='inside', textinfo='percent+label')
fig.update_layout(title_text='Sales Analysis by Category', title_font=dict(size=24))

fig.show()
```

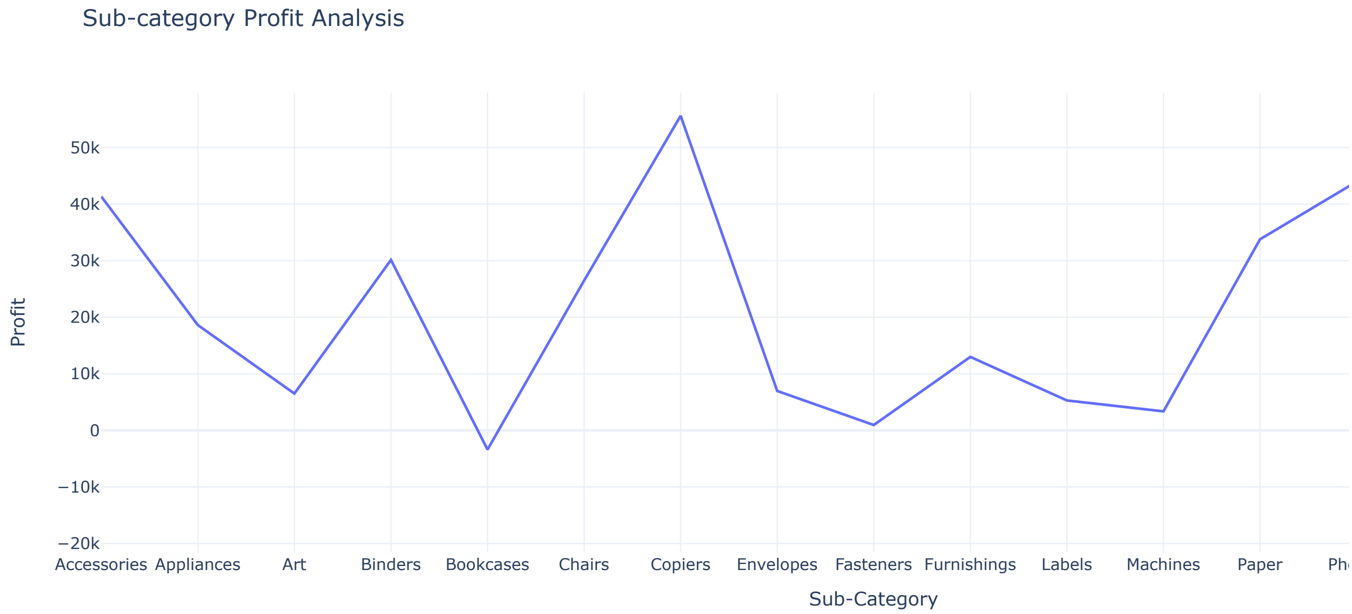



```
sales_by_subcategory = df.groupby('Sub-Category')['Sales'].sum().reset_index()
fig = px.bar(sales_by_subcategory,
             x='Sub-Category',
             y='Sales',
             title='Sales Analysis by Sub-Category')
fig.show()
```

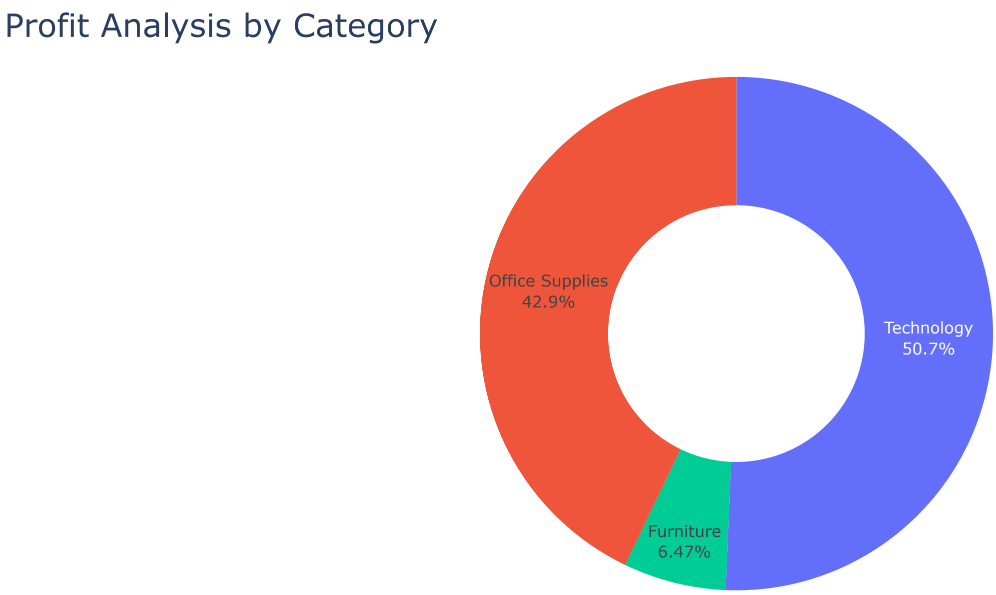



```
profit_by_subcategory = df.groupby('Sub-Category')['Profit'].sum().reset_index()
fig = px.line(profit_by_subcategory,
              x='Sub-Category',
              y='Profit',
```

```
              title='Sub-category Profit Analysis')
fig.show()
```

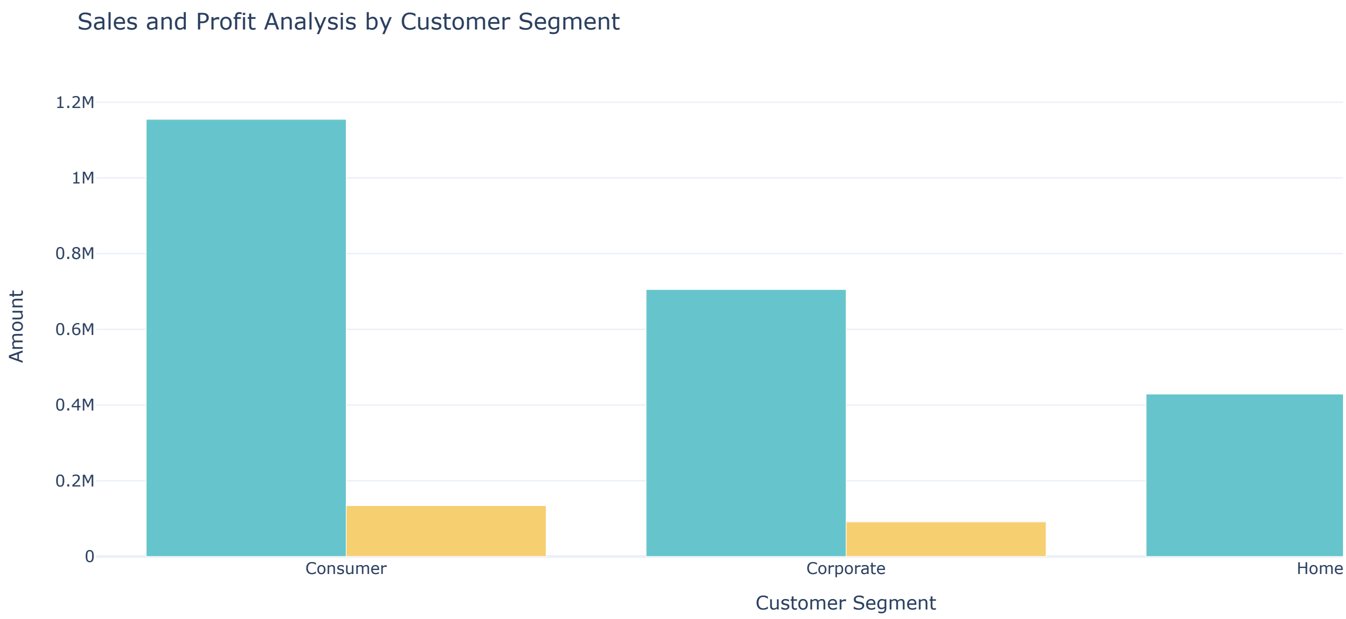



```
profit_by_category = df.groupby('Category')['Profit'].sum().reset_index()

fig = px.pie(profit_by_category,
             values='Profit',
             names='Category',
             hole=0.5,
            )

fig.update_traces(textposition='inside', textinfo='percent+label')
fig.update_layout(title_text='Profit Analysis by Category', title_font=dict(size=24))

fig.show()
```




```
sales_profit_by_segment = df.groupby('Segment').agg({'Sales': 'sum', 'Profit': 'sum'}).reset_index()

color_palette = colors.qualitative.Pastel

fig = go.Figure()
fig.add_trace(go.Bar(x=sales_profit_by_segment['Segment'],
                     y=sales_profit_by_segment['Sales'],
```

```
                     name='Sales',
                     marker_color=color_palette[0]))
fig.add_trace(go.Bar(x=sales_profit_by_segment['Segment'],
                     y=sales_profit_by_segment['Profit'],
                     name='Profit',
                     marker_color=color_palette[1]))

fig.update_layout(title='Sales and Profit Analysis by Customer Segment',
                  xaxis_title='Customer Segment', yaxis_title='Amount')

fig.show()
```




```
sales_profit_by_segment = df.groupby('Segment').agg({'Sales': 'sum', 'Profit': 'sum'}).reset_index()
sales_profit_by_segment['Sales_to_Profit_Ratio'] = sales_profit_by_segment['Sales'] / sales_profit_by_segment['Profit']
print(sales_profit_by_segment[['Segment', 'Sales_to_Profit_Ratio']])
```

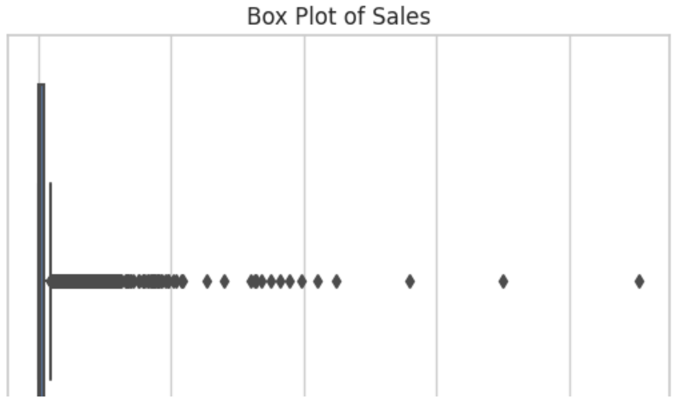
```
        Segment  Sales_to_Profit_Ratio
0      Consumer               8.596148
1     Corporate               7.731070
2   Home Office               7.117491
```

```
# Replace the null values in "segment","region","city"
df["Segment"].fillna("unknown", inplace = True)
df["Region"].fillna("can't say", inplace = True)
df["City"].fillna("no idea", inplace = True)
df["State"].fillna("didn't mention", inplace = True)
df["Category"].fillna("NA", inplace = True)
df["Sub-Category"].fillna("doesn't exist", inplace = True)
```

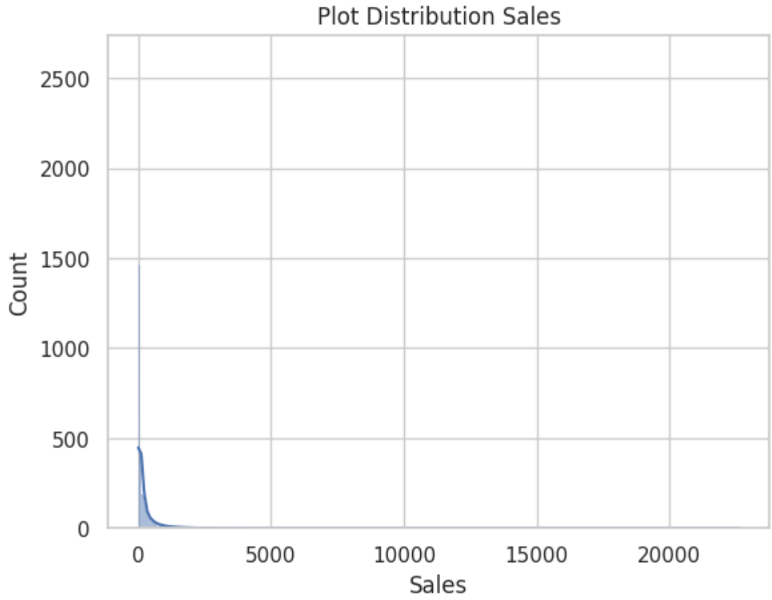
```
#All null values are removed
df.isna().sum()
```

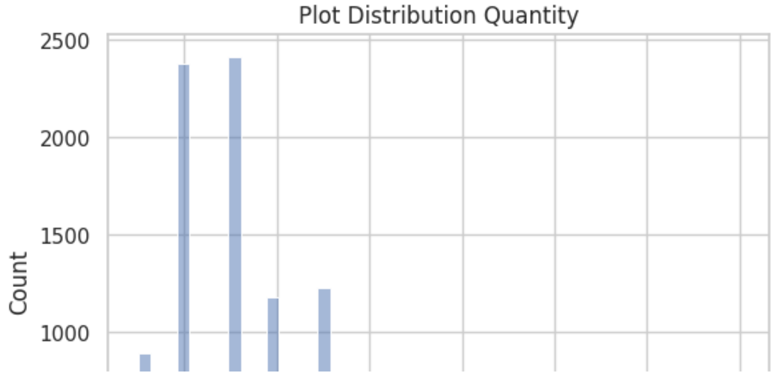
```
Ship Mode       0
Segment         0
City            0
State           0
Region          0
Category        0
Sub-Category    0
Sales           0
Quantity        0
Discount        0
Profit          0
dtype: int64
```

```
df_filtered_univar_1=df.loc[:,'Ship Mode':'Profit']
df_filtered_univar_1=df_filtered_univar_1.select_dtypes([np.int, np.float])

for i, col in enumerate(df_filtered_univar_1.columns):
  plt.figure(i)
  sns.boxplot(x=col, data=df_filtered_univar_1).set_title("Box Plot of "+col)
```




```
df_filtered_univar_1=df.loc[:,'Ship Mode':'Profit']
df_filtered_univar_1=df_filtered_univar_1.select_dtypes([np.int, np.float])

for i, col in enumerate(df_filtered_univar_1.columns):
  plt.figure(i)
  sns.histplot(x=col, data=df_filtered_univar_1, kde=True).set_title("Plot Distribution "+col)
```






```
#Correlation
df.corr()
```

	Sales	Quantity	Discount	Profit
Sales	1.000000	0.199429	-0.028421	0.478635
Quantity	0.199429	1.000000	0.009518	0.064951
Discount	-0.028421	0.009518	1.000000	-0.218722
Profit	0.478635	0.064951	-0.218722	1.000000

```
#plotting the heatmap for correlation
plt.figure(figsize=(18,10))
Corr=df.corr()
correlation_heatMap = sns.heatmap(Corr, annot=True)
```

Linear Regression

```
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsRegressor
```

```
X = df[['Sales', 'Quantity', 'Discount']]
y = df['Profit']
```

```
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.2,random_state=11)
```

```
#creating a model
model = KNeighborsRegressor()
```

```
#fitting the training data in the model
model.fit(X_train, y_train)
```

```
▾ KNeighborsRegressor
KNeighborsRegressor()
```

```
y_pred = model.predict(X_test)
y_pred
```

```
array([ 21.0128 ,   9.53092,  12.80922, ..., -41.81418,  15.9884 ,
         0.98454])
```

```
model.score(X_test, y_test)
```

```
0.7116594330180604
```

```
model.predict(X_test)
```

```
array([  21.0128 ,   19.13478,   12.80922, ..., -100.11046,   12.3044 ,
          0.98376])
```

```
from sklearn.metrics import mean_absolute_error, mean_squared_error
```

```
mae = mean_absolute_error(y_test, y_pred)
rmse = np.sqrt(mean_squared_error(y_test, y_pred))

print(f'Mean Absolute Error: {mae}')
print(f'Root Mean Squared Error: {rmse}')
```

```
Mean Absolute Error: 51.85685897487438
Root Mean Squared Error: 189.4931559784699
```