eda-for-super-store-using-python

April 9, 2024

1 EDA for Super Store using Python

[17]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

[18]:
```
df = pd.read_csv("Downloads/Global_SuperStore.csv", encoding = "latin1")
df.head()
```

[18]:
```
   Row ID         Order ID  Order Date   Ship Date     Ship Mode Customer ID  \
0   32298   CA-2012-124891  31-07-2012  31-07-2012      Same Day    RH-19495
1   26341    IN-2013-77878  05-02-2013  07-02-2013  Second Class    JR-16210
2   25330    IN-2013-71249  17-10-2013  18-10-2013   First Class    CR-12730
3   13524  ES-2013-1579342  28-01-2013  30-01-2013   First Class    KM-16375
4   47221     SG-2013-4320  05-11-2013  06-11-2013      Same Day     RH-9495

      Customer Name      Segment           City            State  …  \
0       Rick Hansen     Consumer  New York City         New York  …
1     Justin Ritter    Corporate     Wollongong  New South Wales  …
2      Craig Reiter     Consumer       Brisbane       Queensland  …
3  Katherine Murray  Home Office         Berlin           Berlin  …
4       Rick Hansen     Consumer          Dakar            Dakar  …

         Product ID    Category Sub-Category  \
0   TEC-AC-10003033  Technology  Accessories
1   FUR-CH-10003950   Furniture       Chairs
2   TEC-PH-10004664  Technology       Phones
3   TEC-PH-10004583  Technology       Phones
4  TEC-SHA-10000501  Technology      Copiers

                                        Product Name     Sales Quantity  \
0  Plantronics CS510 - Over-the-Head monaural Wir…  2309.650        7
1          Novimex Executive Leather Armchair, Black  3709.395        9
2                  Nokia Smart Phone, with Caller ID  5175.171        9
3                     Motorola Smart Phone, Cordless  2892.510        5
4                     Sharp Wireless Fax, High-Speed  2832.960        8
```

```
   Discount     Profit  Shipping Cost  Order Priority
0       0.0   762.1845         933.57        Critical
1       0.1  -288.7650         923.63        Critical
2       0.1   919.9710         915.49          Medium
3       0.1   -96.5400         910.16          Medium
4       0.0   311.5200         903.04        Critical

[5 rows x 24 columns]
```

[19]:
```
df.shape
```

[19]:
```
(51290, 24)
```

1.1 A) Data Preprcessing

1.1.1 1) Check Null Values

[20]:
```
df.isnull().sum()
```

[20]:
```
Row ID                0
Order ID              0
Order Date            0
Ship Date             0
Ship Mode             0
Customer ID           0
Customer Name         0
Segment               0
City                  0
State                 0
Country               0
Postal Code       41296
Market                0
Region                0
Product ID            0
Category              0
Sub-Category          0
Product Name          0
Sales                 0
Quantity              0
Discount              0
Profit                0
Shipping Cost         0
Order Priority        0
dtype: int64
```

[21]:

```
# df['Postal code']
print(f'Postal Code constains {41296*100/51290}% of null values')
```

```
Postal Code constains 80.51472021836615% of null values
```

[22]:

```
df.drop('Postal Code',axis=1,inplace=True)
df.columns
```

[22]:

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

1.1.2 2) Check Data Types

[24]:

```
df.dtypes
```

[24]:

```
Row ID              int64
Order ID           object
Order Date         object
Ship Date          object
Ship Mode          object
Customer ID        object
Customer Name      object
Segment            object
City               object
State              object
Country            object
Market             object
Region             object
Product ID         object
Category           object
Sub-Category       object
Product Name       object
Sales             float64
Quantity            int64
Discount          float64
Profit            float64
Shipping Cost     float64
Order Priority     object
dtype: object
```

1.1.3 3) Check Duplicates

[25]:
```
df.duplicated().sum()
```

[25]:
```
0
```

1.1.4 4) Extract all categorial columns and numerical columns

[26]:
```
cat_cols = df.select_dtypes(include="object").columns
print(cat_cols)
num_cols = df.select_dtypes(exclude="object").columns
print(num_cols)
```

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

2 B) Uni-Variate EDA

Statistical or visual analysis of a single column (one variable)

2.0.1 1. Find value counts of categorial columns namely Category, Segment, Sub Category, Region,Ship Model and Market .

2.0.2 Depict the following

a) Category count on a bar chart(matplotlib)
b) Sub-Category on a horizontal bar chart (matplotlib)
c) Segment on a Pie Chart(matplotlib)
d) Region on a bar chart(seaborn)
e) Ship Mode on a line chart(matplotlib)
f) market on a area chart(matplotlib)

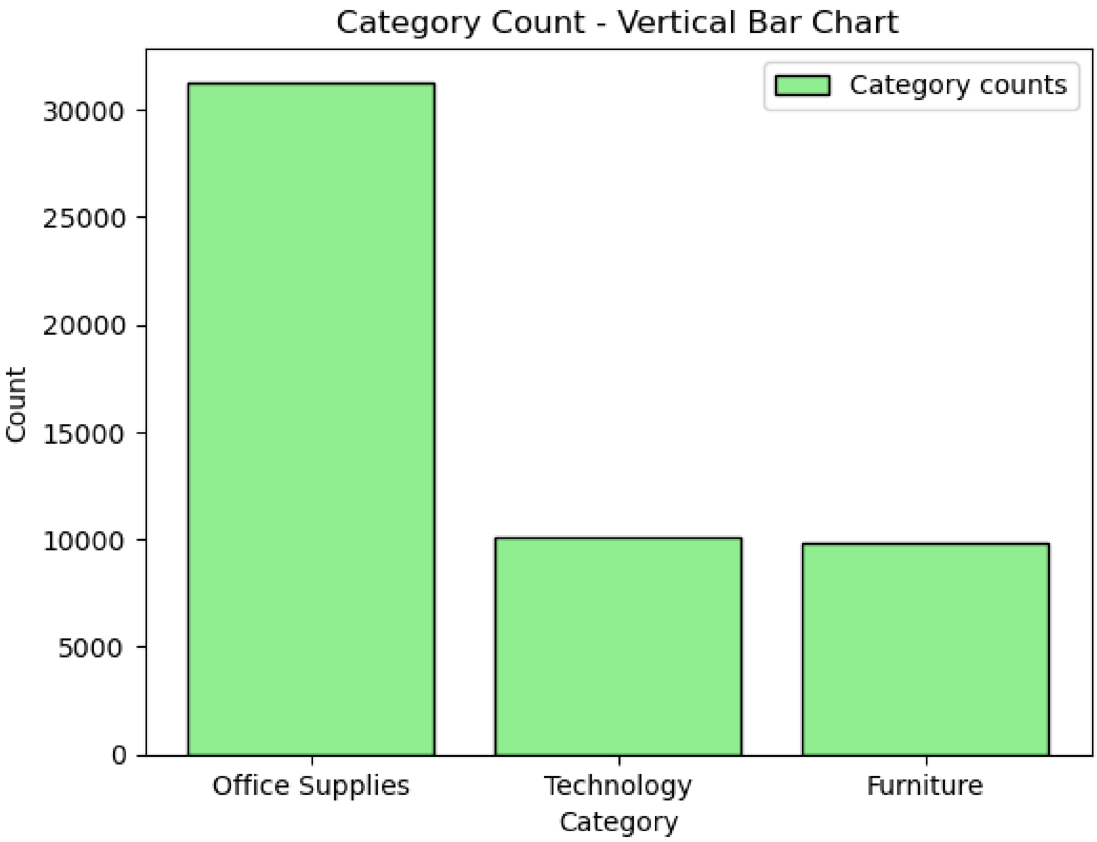
[27]:
```
cat_cols
```

[27]:
```
Index(['Order ID', 'Order Date', 'Ship Date', 'Ship Mode', 'Customer ID',
       'Customer Name', 'Segment', 'City', 'State', 'Country', 'Market',
       'Region', 'Product ID', 'Category', 'Sub-Category', 'Product Name',
       'Order Priority'],
      dtype='object')
```

[28]:
```
a1 = df["Category"].value_counts()
a1
```

[28]:
```
Office Supplies    31273
Technology         10141
Furniture           9876
Name: Category, dtype: int64
```

[31]:
```
plt.bar(a1.index,a1.values,color="lightgreen",edgecolor="black",label="Category␣
  ↪counts")
plt.xlabel("Category")
plt.ylabel("Count")
plt.title("Category Count - Vertical Bar Chart")
plt.legend()
plt.show()
```

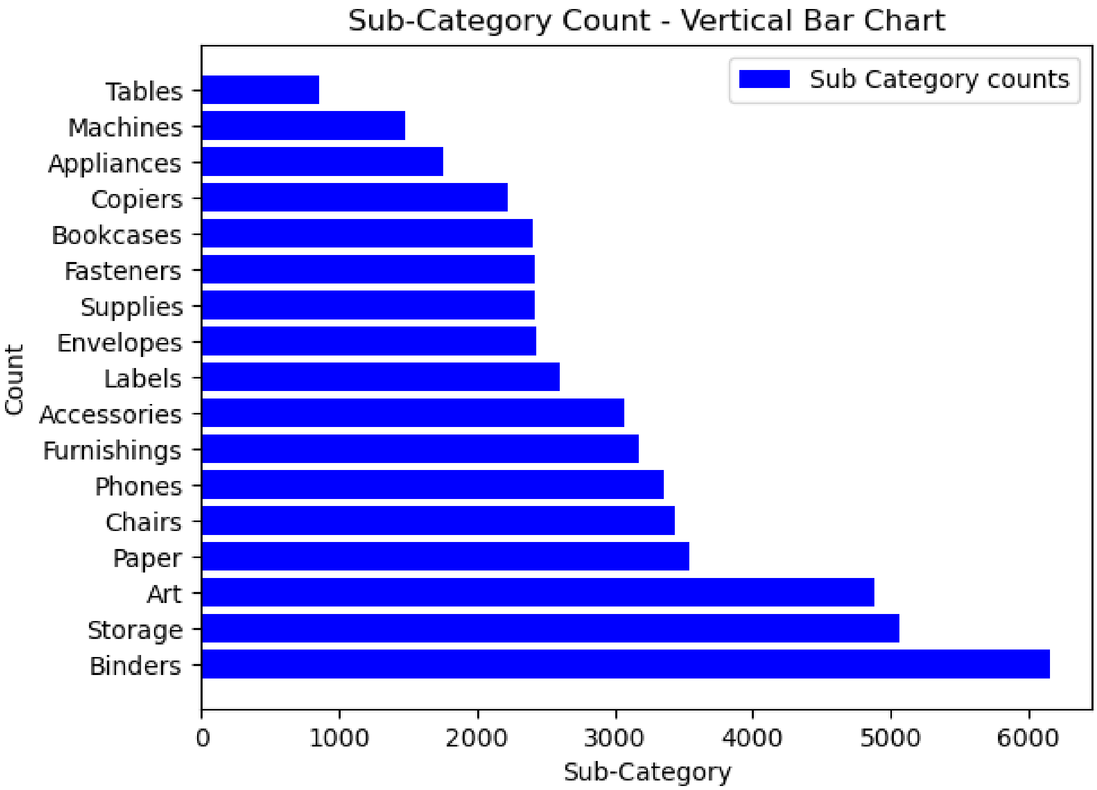


[32]:
```
a2 = df["Sub-Category"].value_counts()
a2
```

[32]:
```
Binders        6152
Storage        5059
Art            4883
Paper          3538
```

```
Chairs         3434
Phones         3357
Furnishings    3170
Accessories    3075
Labels         2606
Envelopes      2435
Supplies       2425
Fasteners      2420
Bookcases      2411
Copiers        2223
Appliances     1755
Machines       1486
Tables          861
Name: Sub-Category, dtype: int64
```

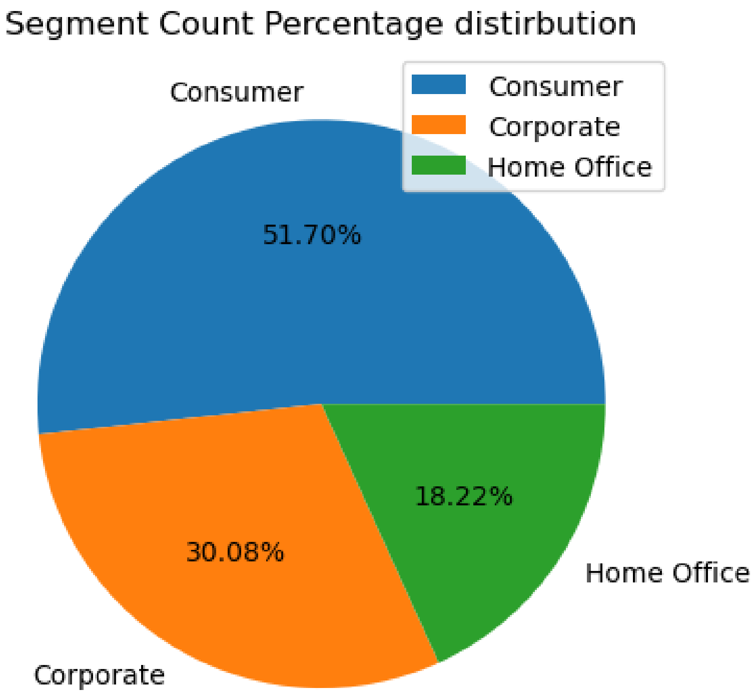
[34]:
```
plt.barh(a2.index,a2.values,color="blue",label="Sub Category counts")
plt.xlabel("Sub-Category")
plt.ylabel("Count")
plt.title("Sub-Category Count - Vertical Bar Chart")
plt.legend()
plt.show()
```



```
[35]: a3 = df["Segment"].value_counts()
      a3
```

```
[35]: Consumer       26518
      Corporate      15429
      Home Office     9343
      Name: Segment, dtype: int64
```

```
[44]: plt.pie(a3.values,labels=a3.index,autopct="%.2f%%")
      plt.title("Segment Count Percentage distirbution")
      plt.legend(loc=1)
      plt.show()
```



```
[45]: a4 = df["Region"].value_counts()
      print(type(a4)) #series
      a4
```

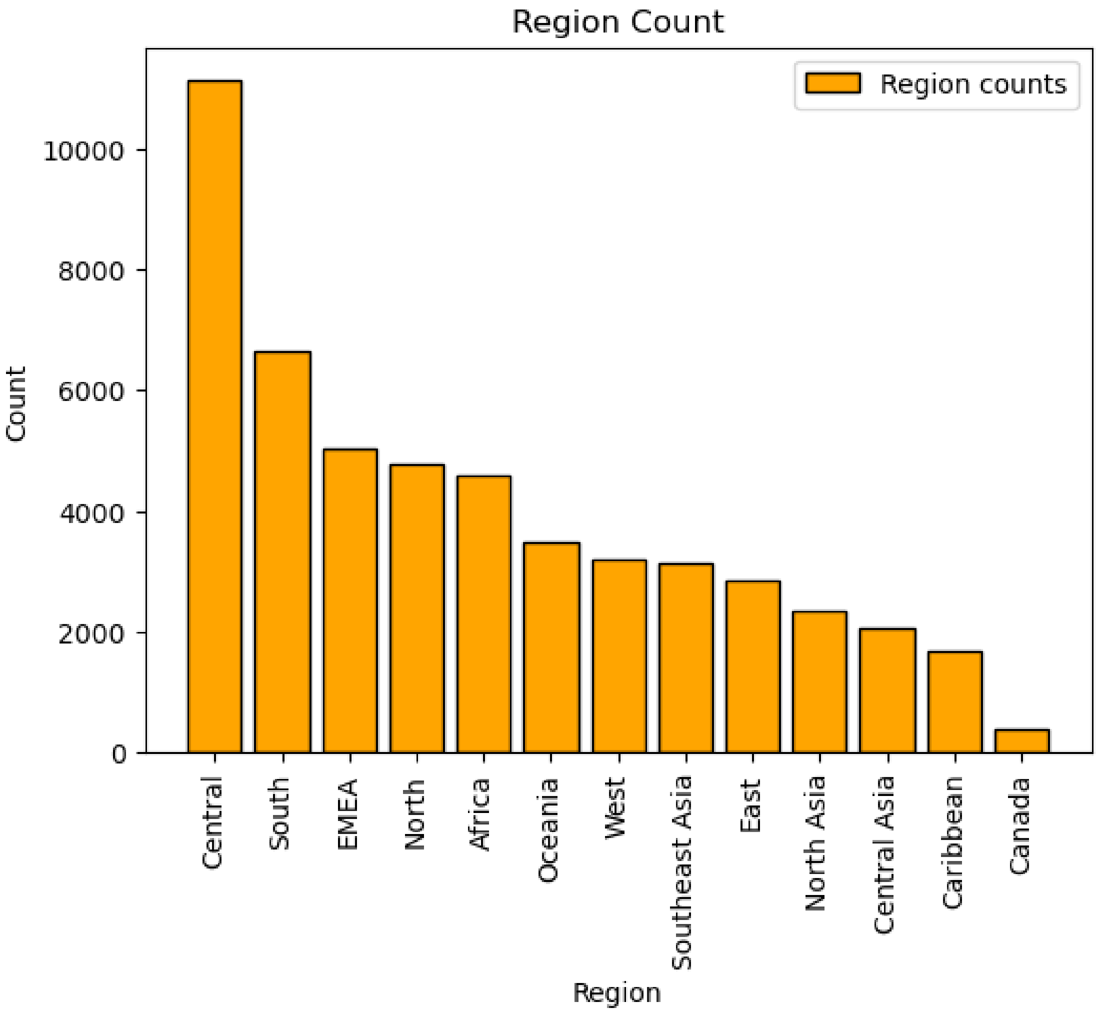
```
<class 'pandas.core.series.Series'>
```

```
[45]: Central           11117
      South              6645
      EMEA               5029
```

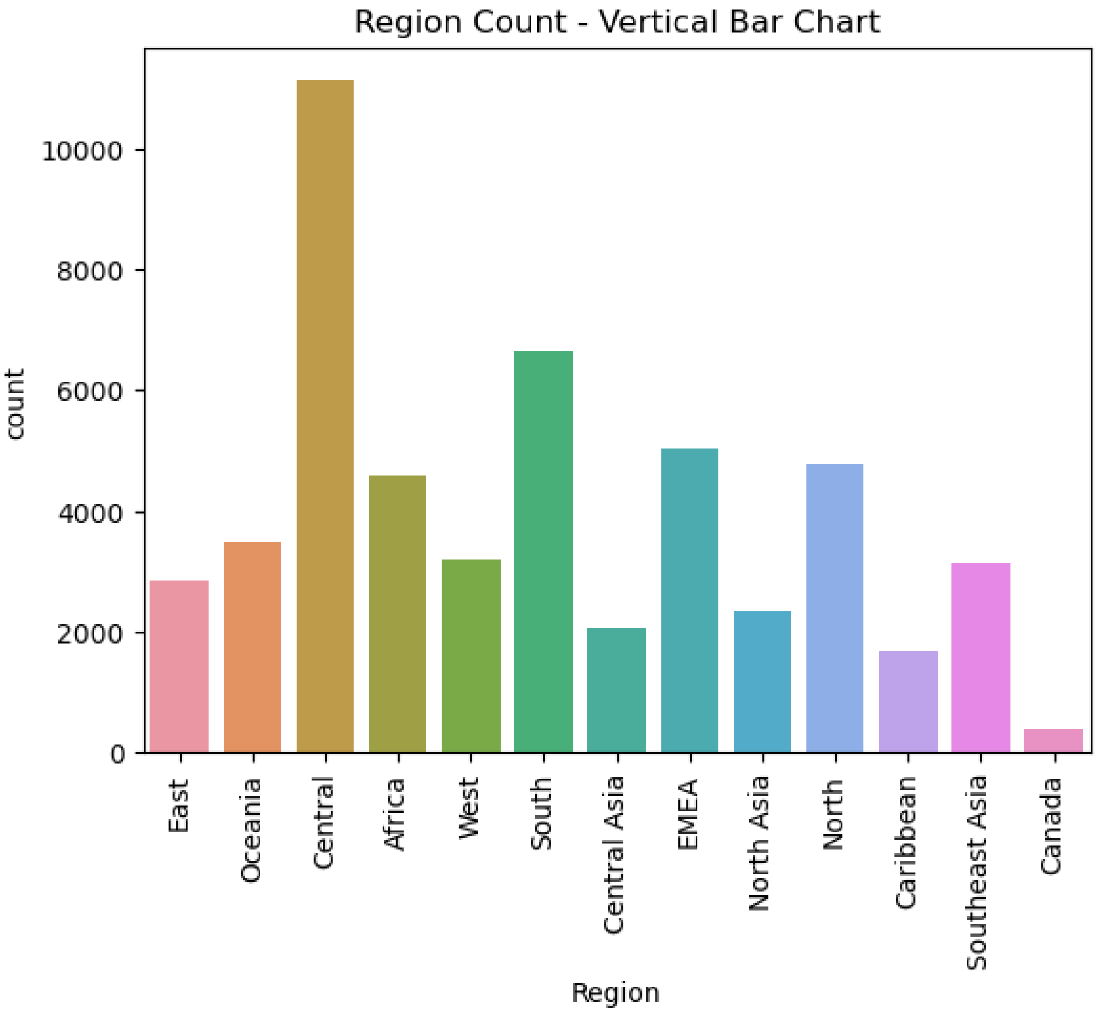
```
North               4785
Africa              4587
Oceania             3487
West                3203
Southeast Asia      3129
East                2848
North Asia          2338
Central Asia        2048
Caribbean           1690
Canada               384
Name: Region, dtype: int64
```

[46]:

```
plt.bar(a4.index,a4.values,color="orange",edgecolor="black",label="Region␣
 ↪counts")
plt.xlabel("Region")
plt.ylabel("Count")
plt.title("Region Count")
plt.legend()
plt.xticks(rotation=90)
plt.show()
```



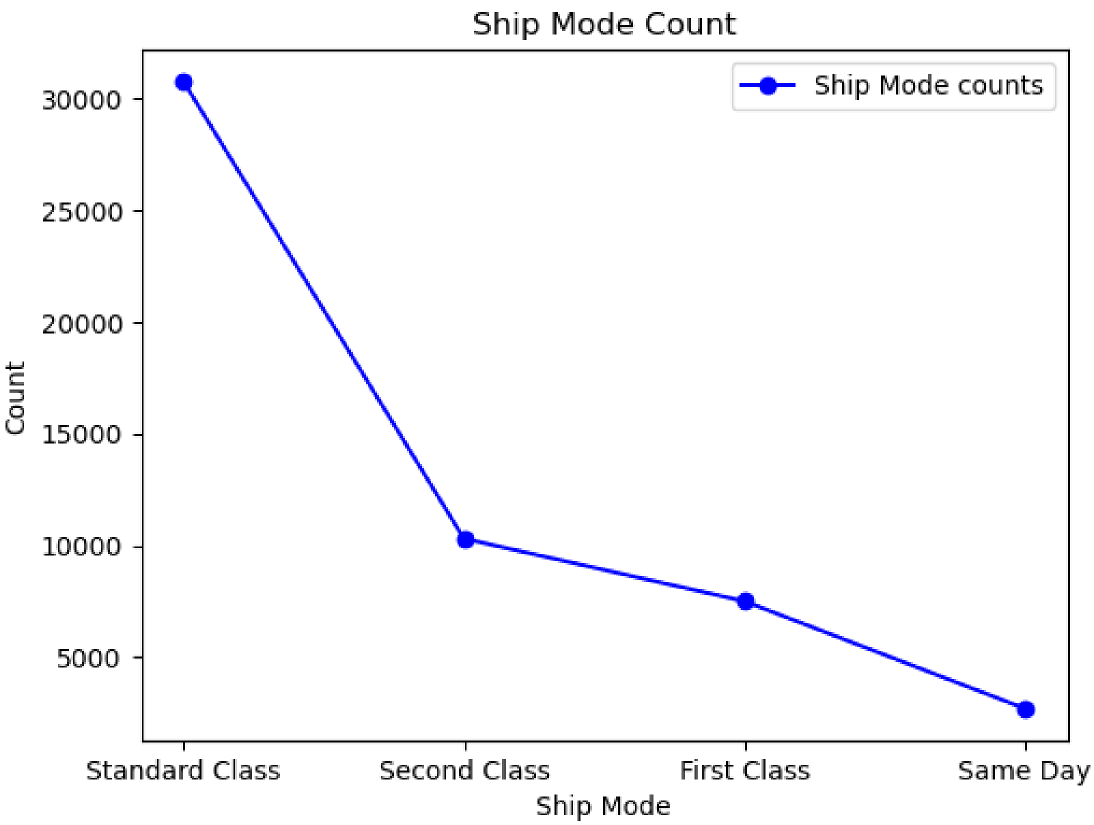
[47]:
```
sns.countplot(x=df["Region"])
plt.title("Region Count - Vertical Bar Chart")
plt.xticks(rotation=90)
plt.show()
```



```
[48]: a5 = df["Ship Mode"].value_counts()
      a5
```

```
[48]: Standard Class    30775
      Second Class      10309
      First Class        7505
      Same Day           2701
      Name: Ship Mode, dtype: int64
```

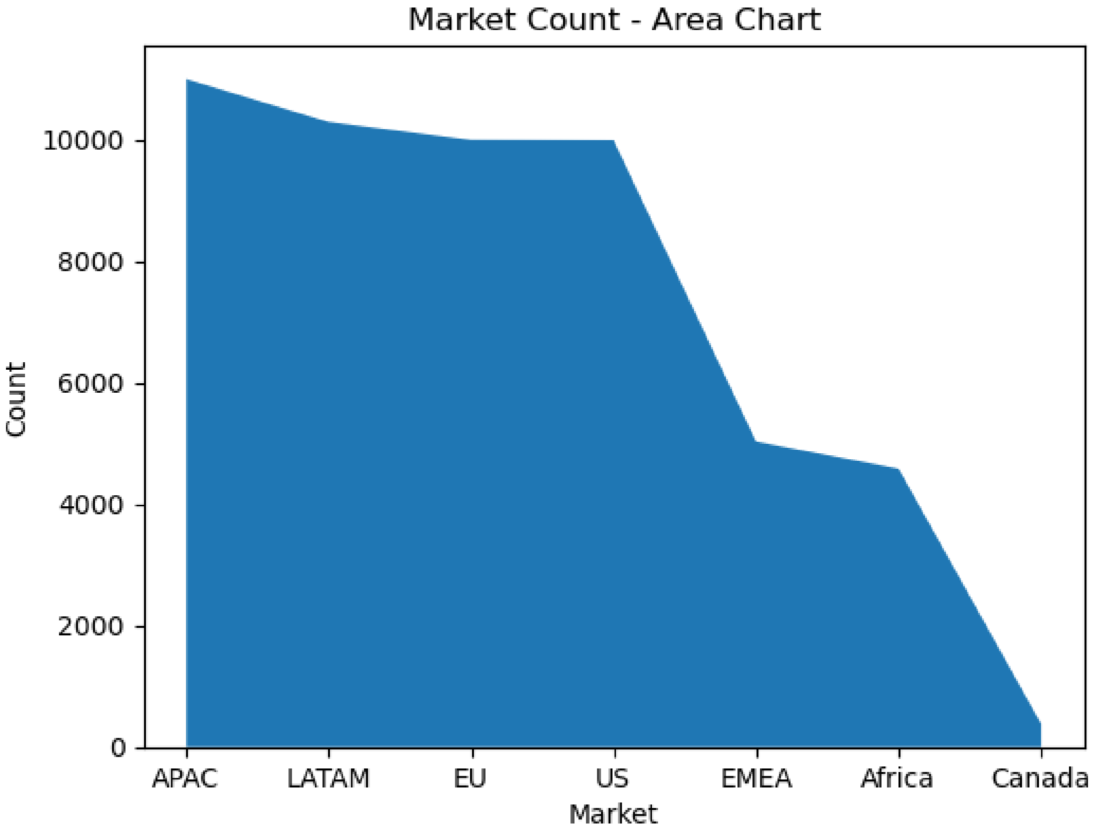
```
[49]: plt.plot(a5.index,a5.values,color="blue",marker="o",label="Ship Mode counts")
      plt.xlabel("Ship Mode")
      plt.ylabel("Count")
      plt.title("Ship Mode Count")
      plt.legend()
      plt.show()
```



[50]:
```
a6 = df["Market"].value_counts()
a6
```

[50]:
```
APAC      11002
LATAM     10294
EU        10000
US         9994
EMEA       5029
Africa     4587
Canada      384
Name: Market, dtype: int64
```

[51]:
```
plt.stackplot(a6.index,a6.values)
plt.xlabel("Market")
plt.ylabel("Count")
plt.title("Market Count - Area Chart")
plt.show()
```



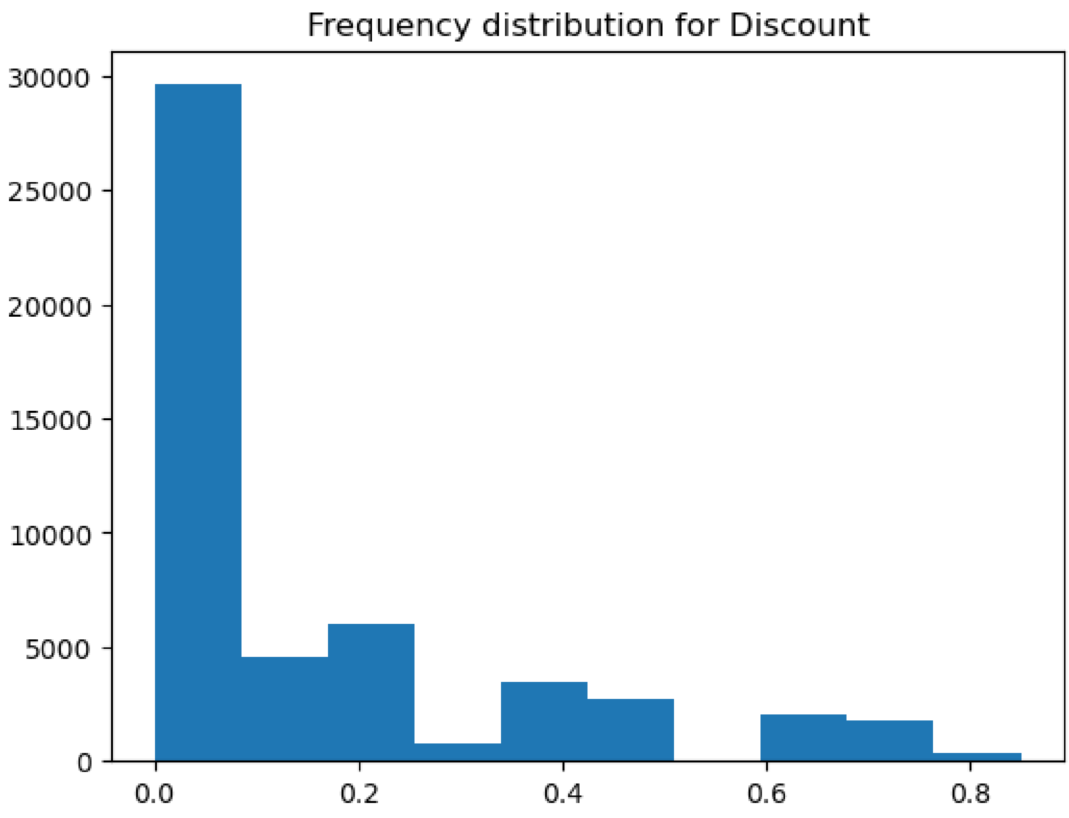
2.1 2) Plot the following

2.1.1 a) Histogram for Discount

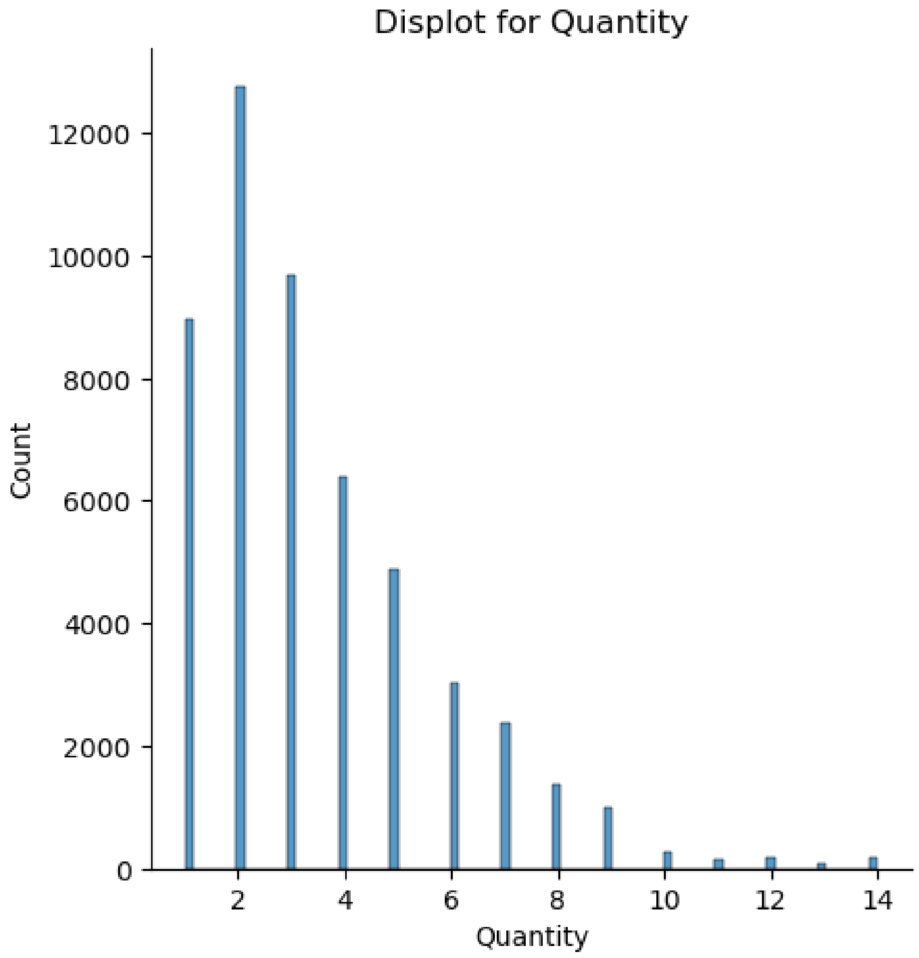
2.1.2 b) kdeplot/displot for Quantity

[52]:

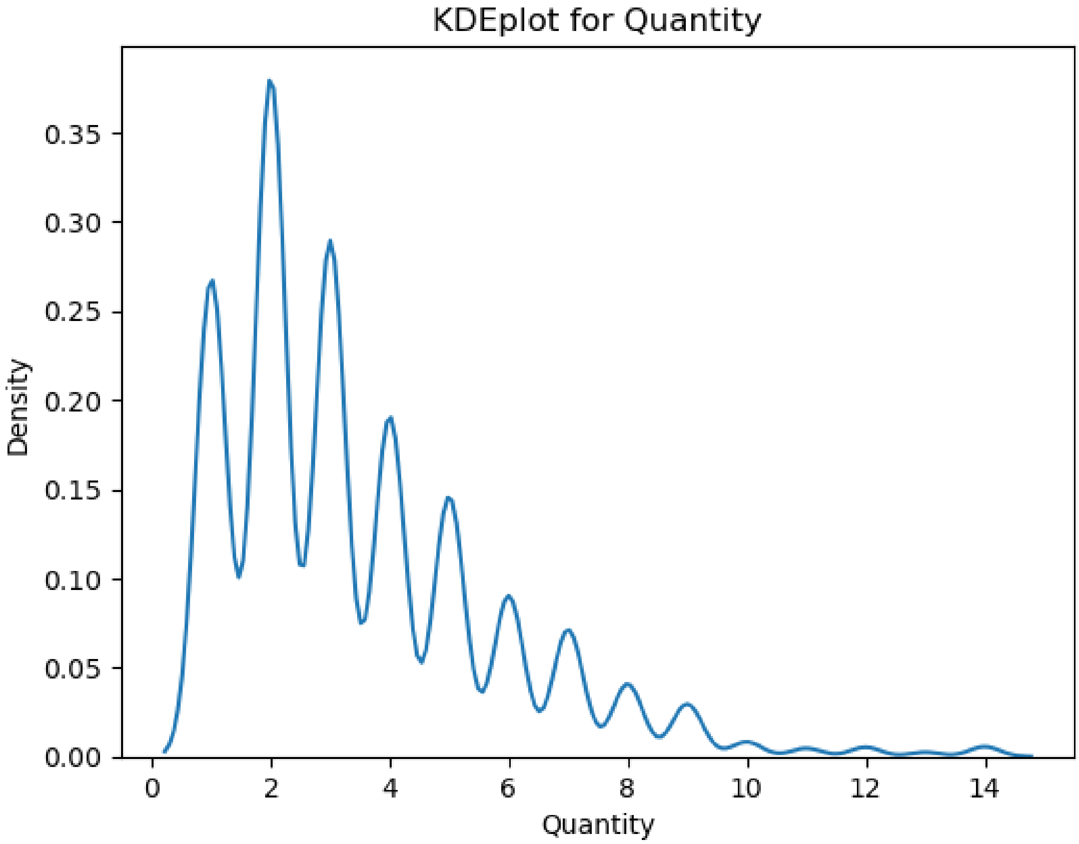
```
plt.hist(df["Discount"])
plt.title("Frequency distribution for Discount")
plt.show()
```



[53]:
```
sns.displot(x=df["Quantity"])
plt.title("Displot for Quantity")
plt.show()
```



```
[54]: sns.kdeplot(x=df["Quantity"])
      plt.title("KDEplot for Quantity")
      plt.show()
```



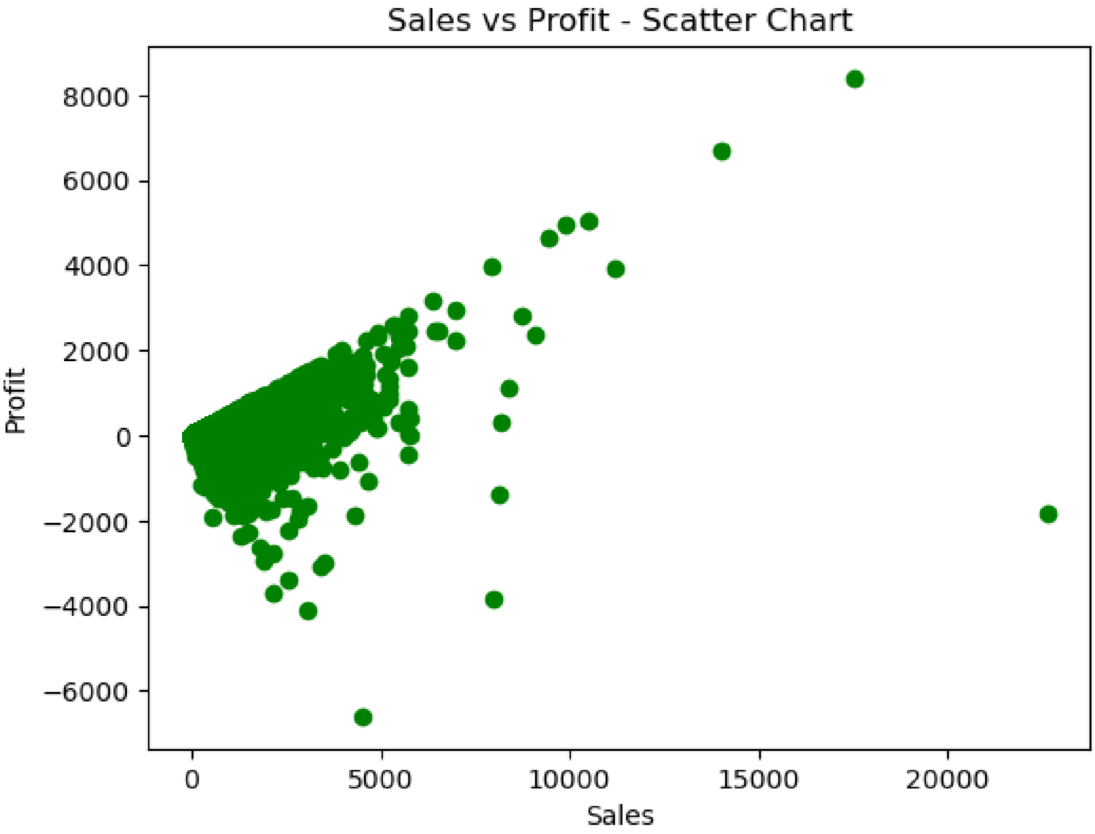
2.2 c) Bivariate Analysis

Statistical or Visual Analysis of 2 variables

2.3 1) Depict the following on a Scater plot

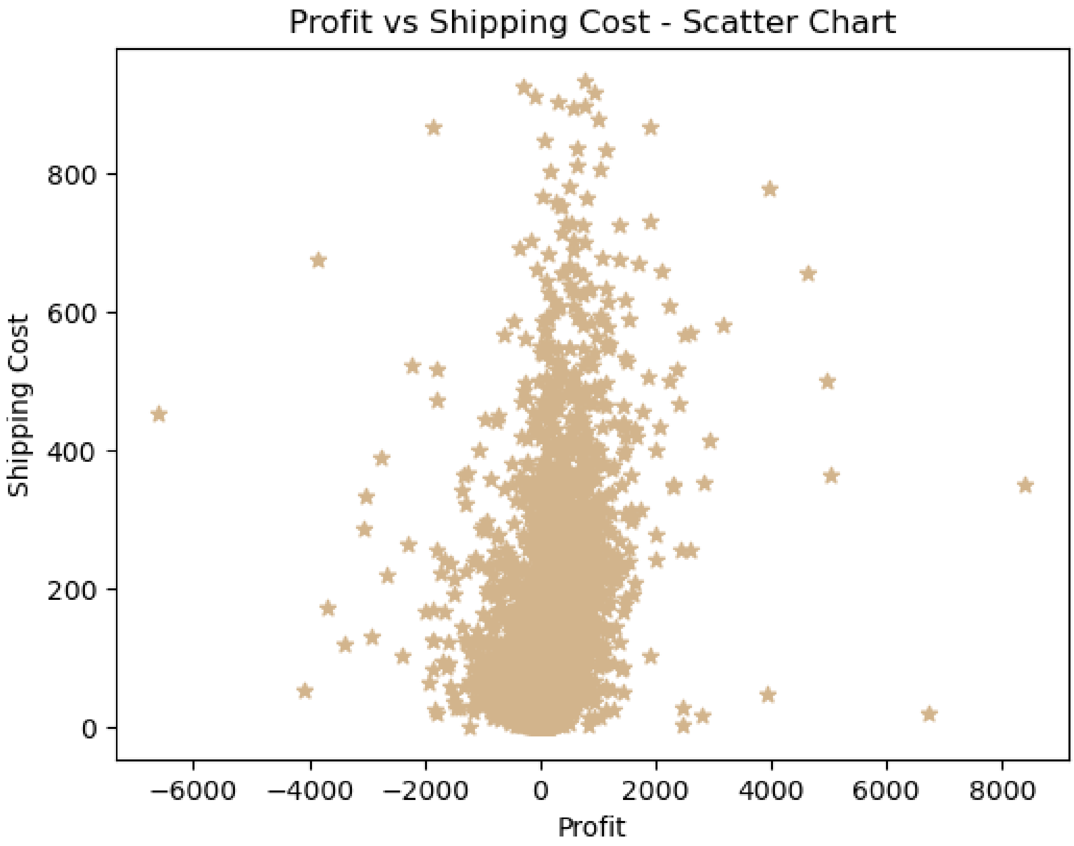
a) Sales vs Profit
b) Profit vs Shopping Cost

[58]:
```
plt.scatter(df["Sales"],df["Profit"],color = "green")
plt.title("Sales vs Profit - Scatter Chart")
plt.xlabel("Sales")
plt.ylabel("Profit")
plt.show()
```



[59]:

```
plt.scatter(df["Profit"],df["Shipping Cost"],color = "tan",marker="*")
plt.title("Profit vs Shipping Cost - Scatter Chart")
plt.xlabel("Profit")
plt.ylabel("Shipping Cost")
plt.show()
```

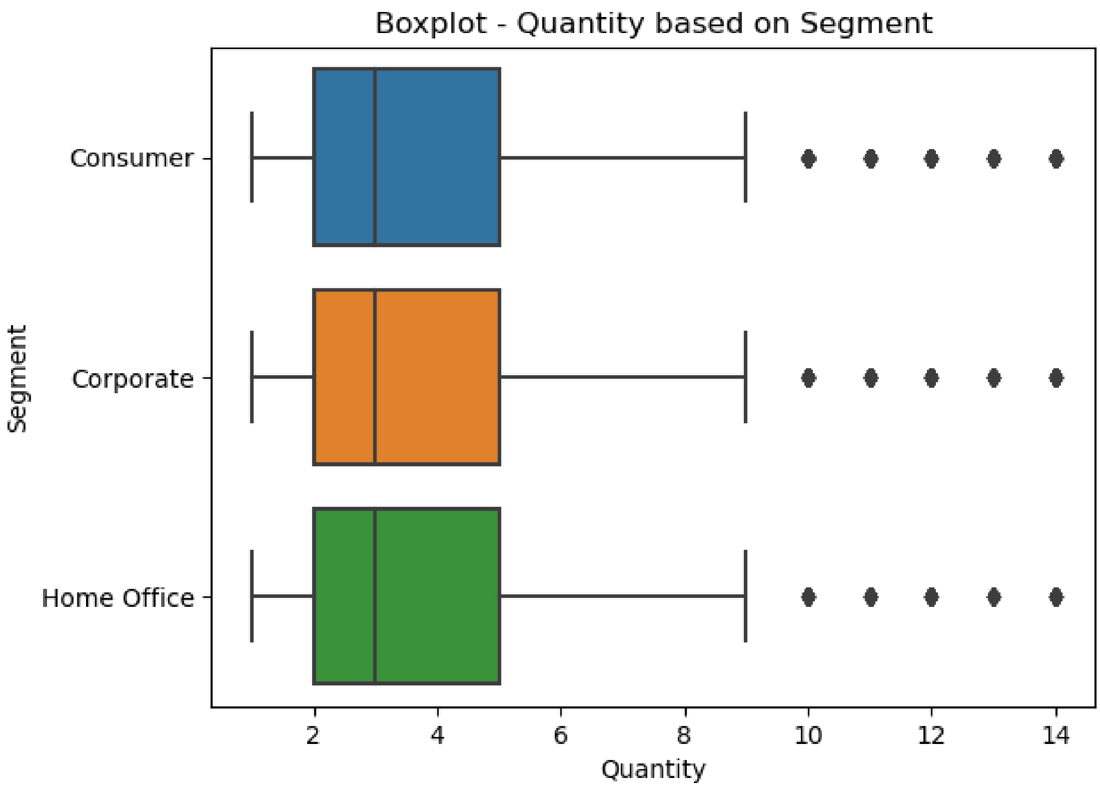


2.4 2) Depict Boxplot for the following

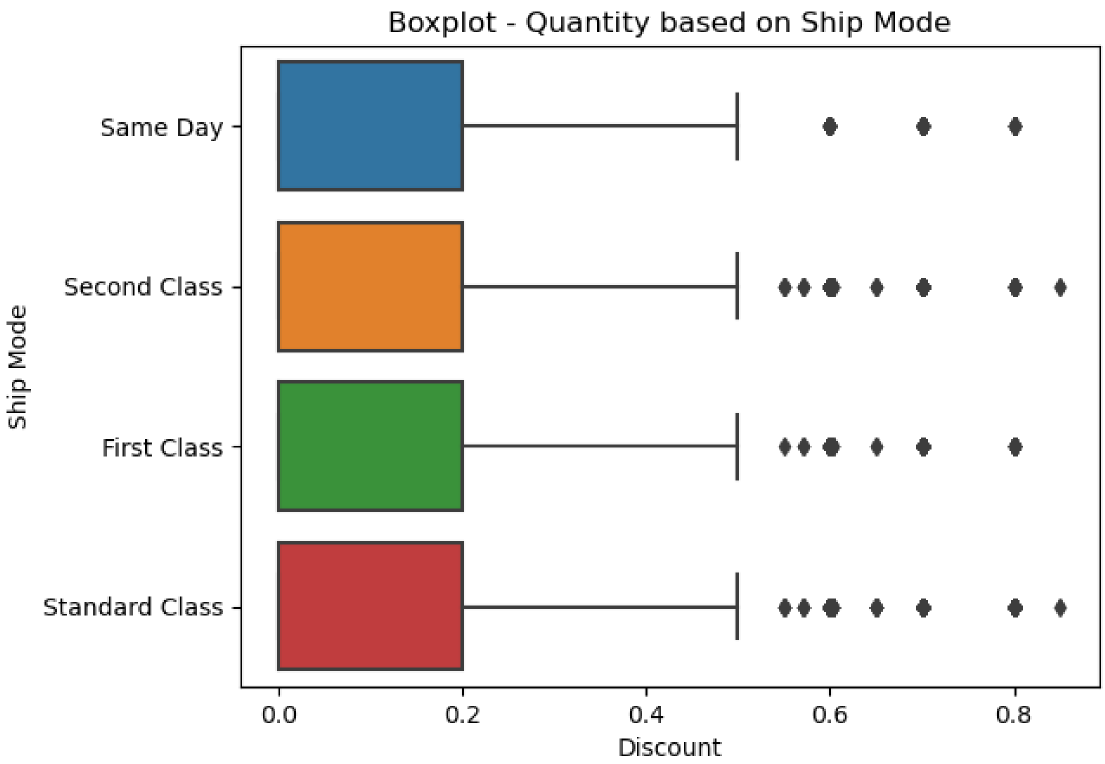
2.4.1 a) Quantity based on Segment

2.4.2 b) Discount based on Ship Mode

```
[60]: sns.boxplot(x=df["Quantity"],y=df["Segment"])
plt.title("Boxplot - Quantity based on Segment")
plt.show()
```



[61]:
```
sns.boxplot(x=df["Discount"],y=df["Ship Mode"])
plt.title("Boxplot - Quantity based on Ship Mode")
plt.show()
```

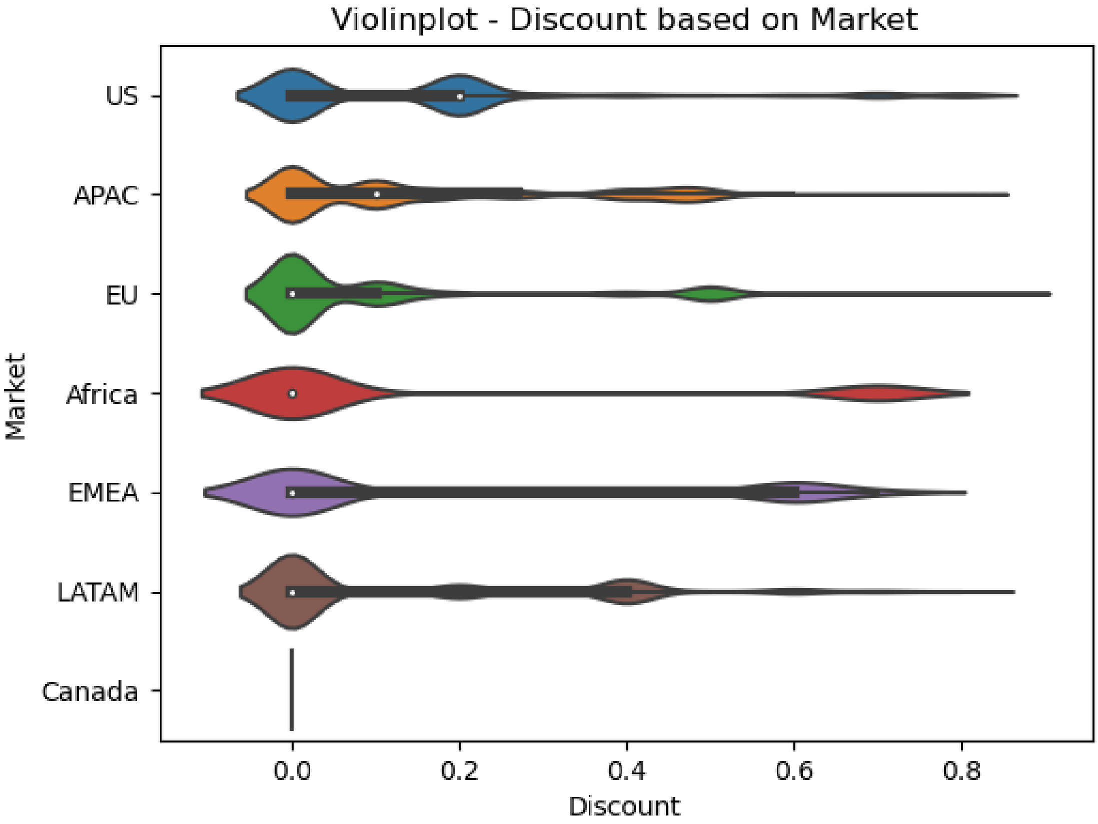


2.5 Depict the following

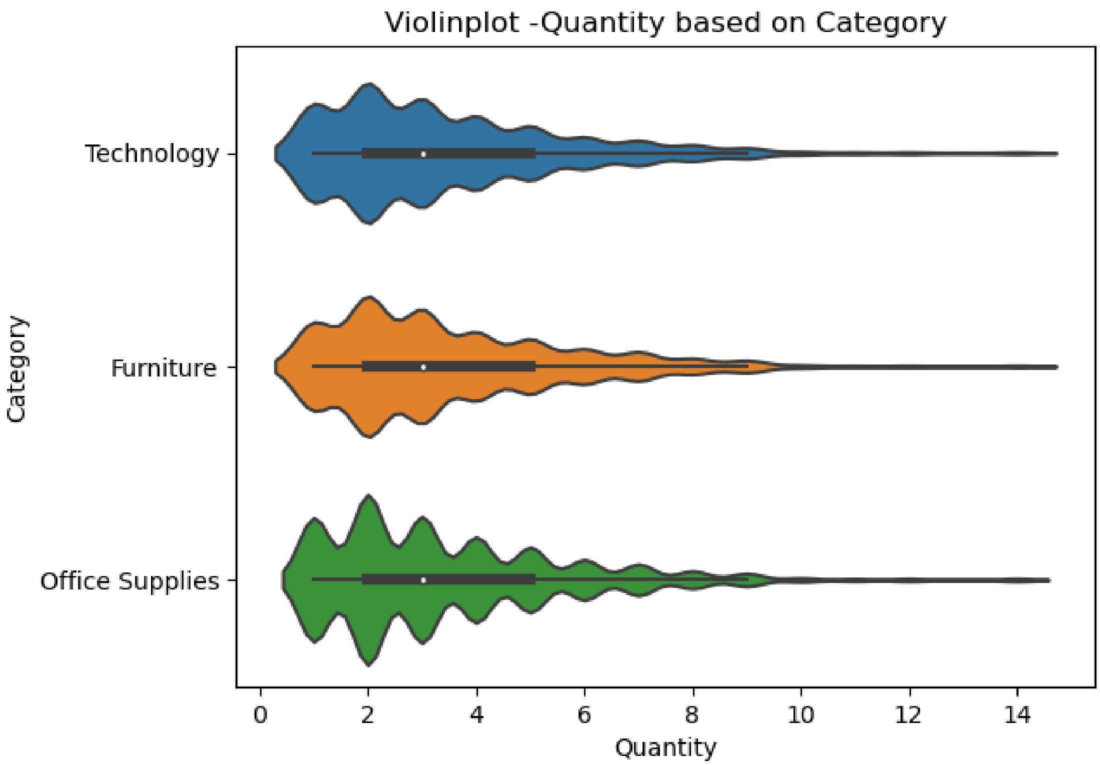
2.5.1 a) Discount vs Market on a violinplot

2.5.2 b) Quantity vs Category on a violinplot

[64]:
```
sns.violinplot(x=df["Discount"],y=df["Market"])
plt.title("Violinplot - Discount based on Market")
plt.show()
```



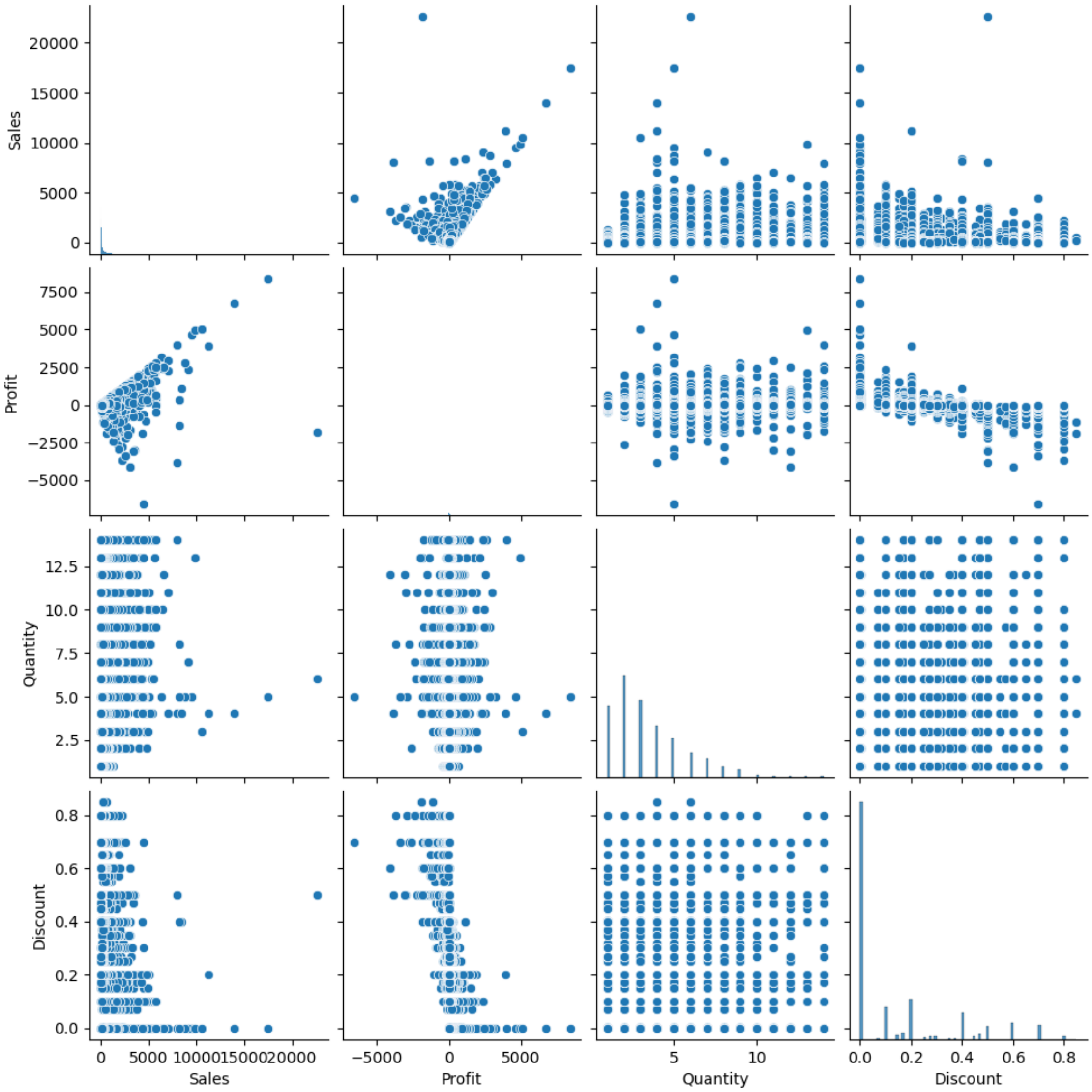
[66]:
```
sns.violinplot(x=df["Quantity"],y=df["Category"])
plt.title("Violinplot -Quantity based on Category")
plt.show()
```



2.5.3 4) Plot pairplot for the categorial variables including [“Sales”,“Profit”,“Quantity”,“Discount”]

[69]:

```
sns.pairplot(df,vars=["Sales","Profit","Quantity","Discount"])
plt.show()
```



[70]:
```
print(num_cols)
print(cat_cols)
```

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

2.5.4 5) Depict Correlation on a heatmap

[71]:

```
corr = df.corr()
corr
```

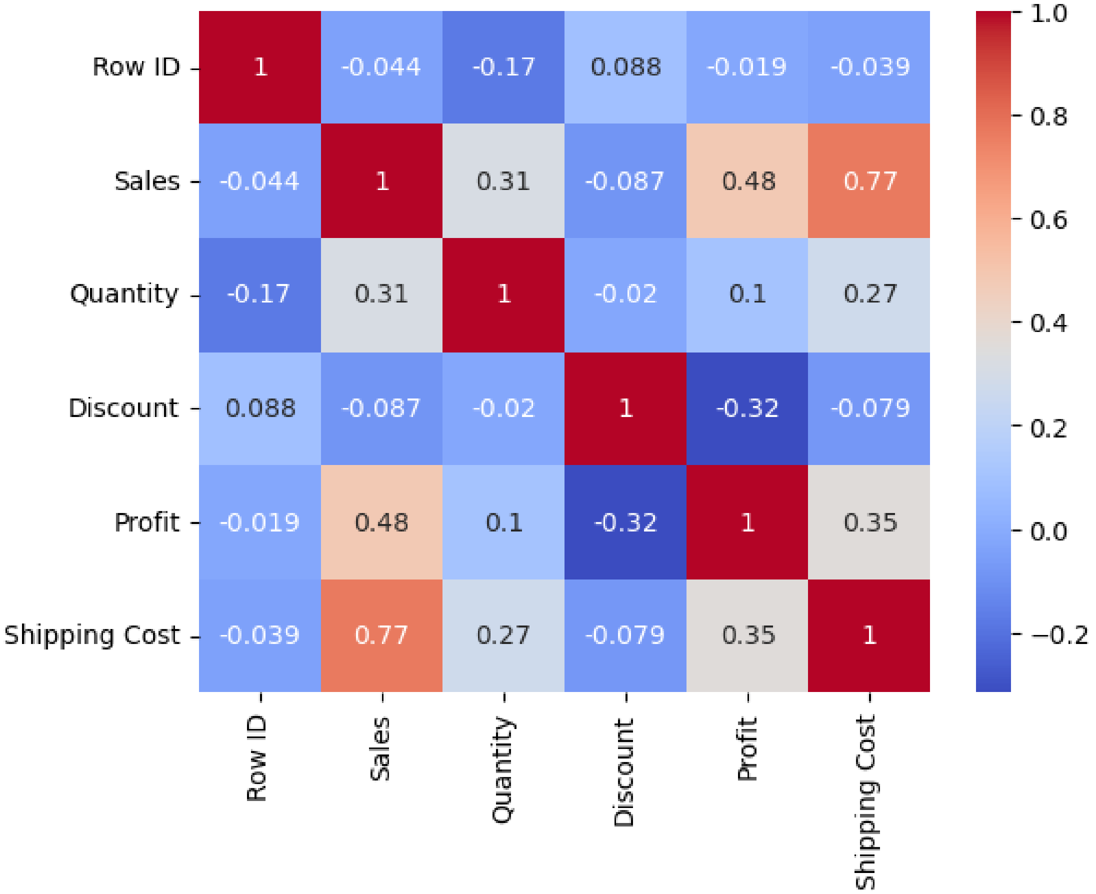
```
C:\Users\Dell\AppData\Local\Temp\ipykernel_4488\2438084875.py:1: FutureWarning:
The default value of numeric_only in DataFrame.corr is deprecated. In a future
version, it will default to False. Select only valid columns or specify the
value of numeric_only to silence this warning.
  corr = df.corr()
```

[71]:

```
                 Row ID     Sales  Quantity  Discount    Profit  Shipping Cost
Row ID         1.000000 -0.043889 -0.173483  0.087594 -0.019037      -0.039076
Sales         -0.043889  1.000000  0.313577 -0.086722  0.484918       0.768073
Quantity      -0.173483  0.313577  1.000000 -0.019875  0.104365       0.272649
Discount       0.087594 -0.086722 -0.019875  1.000000 -0.316490      -0.079055
Profit        -0.019037  0.484918  0.104365 -0.316490  1.000000       0.354441
Shipping Cost -0.039076  0.768073  0.272649 -0.079055  0.354441       1.000000
```

[72]:

```
sns.heatmap(corr,annot=True,cmap="coolwarm")
plt.show()
```


Row ID
Sales
Quantity
Discount
Profit
Shipping Cost
1 | -0.044 | -0.17 | 0.088 | -0.019 | -0.039
-0.044 | 1 | 0.31 | -0.087 | 0.48 | 0.77
-0.17 | 0.31 | 1 | -0.02 | 0.1 | 0.27
0.088 | -0.087 | -0.02 | 1 | -0.32 | -0.079
-0.019 | 0.48 | 0.1 | -0.32 | 1 | 0.35
-0.039 | 0.77 | 0.27 | -0.079 | 0.35 | 1
Row ID
Sales
Quantity
Discount
Profit
Shipping Cost
1.0
0.8
0.6
0.4
0.2
0.0
−0.2