# nap-queen-assessment-20mia1105

August 2, 2024

**Submitted by: Parvathy Menon - 20MIA1105**

[297]:

```
# Import necessary libraries
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score, classification_report,␣
 ↪confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns
```

[299]:

```
!pip install missingno
import missingno
```

```
Requirement already satisfied: missingno in c:\users\parvathy
menon\anaconda3\lib\site-packages (0.5.2)
Requirement already satisfied: numpy in c:\users\parvathy
menon\anaconda3\lib\site-packages (from missingno) (1.26.4)
Requirement already satisfied: matplotlib in c:\users\parvathy
menon\anaconda3\lib\site-packages (from missingno) (3.8.4)
Requirement already satisfied: scipy in c:\users\parvathy
menon\anaconda3\lib\site-packages (from missingno) (1.13.1)
Requirement already satisfied: seaborn in c:\users\parvathy
menon\anaconda3\lib\site-packages (from missingno) (0.13.2)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\parvathy
menon\anaconda3\lib\site-packages (from matplotlib->missingno) (1.2.0)
Requirement already satisfied: cycler>=0.10 in c:\users\parvathy
menon\anaconda3\lib\site-packages (from matplotlib->missingno) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\parvathy
menon\anaconda3\lib\site-packages (from matplotlib->missingno) (4.51.0)
Requirement already satisfied: kiwisolver>=1.3.1 in c:\users\parvathy
menon\anaconda3\lib\site-packages (from matplotlib->missingno) (1.4.4)
Requirement already satisfied: packaging>=20.0 in c:\users\parvathy
menon\anaconda3\lib\site-packages (from matplotlib->missingno) (23.2)
Requirement already satisfied: pillow>=8 in c:\users\parvathy
menon\anaconda3\lib\site-packages (from matplotlib->missingno) (10.3.0)
Requirement already satisfied: pyparsing>=2.3.1 in c:\users\parvathy
menon\anaconda3\lib\site-packages (from matplotlib->missingno) (3.0.9)
```

```
Requirement already satisfied: python-dateutil>=2.7 in c:\users\parvathy
menon\anaconda3\lib\site-packages (from matplotlib->missingno) (2.9.0.post0)
Requirement already satisfied: pandas>=1.2 in c:\users\parvathy
menon\anaconda3\lib\site-packages (from seaborn->missingno) (2.2.2)
Requirement already satisfied: pytz>=2020.1 in c:\users\parvathy
menon\anaconda3\lib\site-packages (from pandas>=1.2->seaborn->missingno)
(2024.1)
Requirement already satisfied: tzdata>=2022.7 in c:\users\parvathy
menon\anaconda3\lib\site-packages (from pandas>=1.2->seaborn->missingno)
(2023.3)
Requirement already satisfied: six>=1.5 in c:\users\parvathy
menon\anaconda3\lib\site-packages (from python-
dateutil>=2.7->matplotlib->missingno) (1.16.0)
```

[300]:

```
df = pd.read_csv('Global-Superstore.csv')
df
```

[300]:

```
       Row ID         Order ID  Order Date   Ship Date       Ship Mode  \
0       32298   CA-2012-124891   7/31/2012   7/31/2012        Same Day
1       26341    IN-2013-77878    2/5/2013    2/7/2013    Second Class
2       25330    IN-2013-71249  10/17/2013  10/18/2013     First Class
3       13524  ES-2013-1579342   1/28/2013   1/30/2013     First Class
4       47221     SG-2013-4320   11/5/2013   11/6/2013        Same Day
…         …              …           …          …             …
51285   29002    IN-2014-62366   6/19/2014   6/19/2014        Same Day
51286   35398   US-2014-102288   6/20/2014   6/24/2014  Standard Class
51287   40470   US-2013-155768   12/2/2013   12/2/2013        Same Day
51288    9596   MX-2012-140767   2/18/2012   2/22/2012  Standard Class
51289    6147   MX-2012-134460   5/22/2012   5/26/2012    Second Class

      Customer ID     Customer Name      Segment           City  \
0        RH-19495       Rick Hansen     Consumer  New York City
1        JR-16210     Justin Ritter    Corporate     Wollongong
2        CR-12730      Craig Reiter     Consumer       Brisbane
3        KM-16375  Katherine Murray  Home Office         Berlin
4         RH-9495       Rick Hansen     Consumer          Dakar
…           …               …             …            …
51285    KE-16420  Katrina Edelman    Corporate           Kure
51286    ZC-21910  Zuschuss Carroll     Consumer        Houston
51287    LB-16795    Laurel Beltran  Home Office         Oxnard
51288    RB-19795        Ross Baird  Home Office       Valinhos
51289    MC-18100     Mick Crebagga     Consumer       Tipitapa

                 State  …       Product ID        Category Sub-Category  \
0             New York  …  TEC-AC-10003033      Technology  Accessories
1      New South Wales  …  FUR-CH-10003950       Furniture       Chairs
2           Queensland  …  TEC-PH-10004664      Technology       Phones
```

```
3               Berlin  …   TEC-PH-10004583       Technology       Phones
4                Dakar  …  TEC-SHA-10000501       Technology      Copiers
…                …  …               …              …          …
51285        Hiroshima  …   OFF-FA-10000746  Office Supplies    Fasteners
51286            Texas  …   OFF-AP-10002906  Office Supplies   Appliances
51287       California  …   OFF-EN-10001219  Office Supplies    Envelopes
51288        S<o Paulo  …   OFF-BI-10000806  Office Supplies      Binders
51289          Managua  …   OFF-PA-10004155  Office Supplies        Paper

                                            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
…                                                  …       …      …
51285                      Advantus Thumb Tacks, 12 Pack    65.100        5
51286  Hoover Replacement Belt for Commercial Guardsm…     0.444        1
51287       #10- 4 1/8" x 9 1/2" Security-Tint Envelopes    22.920        3
51288                            Acco Index Tab, Economy    13.440        2
51289            Eaton Computer Printout Paper, 8.5 x 11    61.380        3

      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
…        …       …            …             …
51285      0.0    4.5000           0.01          Medium
51286      0.8   -1.1100           0.01          Medium
51287      0.0   11.2308           0.01            High
51288      0.0    2.4000           0.00          Medium
51289      0.0    1.8000           0.00            High

[51290 rows x 24 columns]
```

[301]:

```
# To display the summary statistics of the data
df.describe(include='all')
```

[301]:

```
             Row ID        Order ID Order Date   Ship Date       Ship Mode  \
count   51290.00000           51290      51290       51290           51290
unique          NaN           25035       1430        1464               4
top             NaN  CA-2014-100111  6/18/2014  11/22/2014  Standard Class
freq            NaN              14        135         130           30775
mean    25645.50000             NaN        NaN         NaN             NaN
std     14806.29199             NaN        NaN         NaN             NaN
```

```
min         1.00000             NaN        NaN         NaN             NaN
25%     12823.25000             NaN        NaN         NaN             NaN
50%     25645.50000             NaN        NaN         NaN             NaN
75%     38467.75000             NaN        NaN         NaN             NaN
max     51290.00000             NaN        NaN         NaN             NaN

       Customer ID    Customer Name  Segment           City       State  ...  \
count        51290            51290    51290          51290       51290  ...
unique        1590              795        3           3636        1094  ...
top       PO-18850  Muhammed Yedwab  Consumer  New York City  California  ...
freq            97              108    26518            915        2001  ...
mean           NaN              NaN      NaN            NaN         NaN  ...
std            NaN              NaN      NaN            NaN         NaN  ...
min            NaN              NaN      NaN            NaN         NaN  ...
25%            NaN              NaN      NaN            NaN         NaN  ...
50%            NaN              NaN      NaN            NaN         NaN  ...
75%            NaN              NaN      NaN            NaN         NaN  ...
max            NaN              NaN      NaN            NaN         NaN  ...

             Product ID         Category Sub-Category Product Name  \
count             51290            51290        51290        51290
unique            10292                3           17         3788
top     OFF-AR-10003651  Office Supplies      Binders      Staples
freq                 35            31273         6152          227
mean                NaN              NaN          NaN          NaN
std                 NaN              NaN          NaN          NaN
min                 NaN              NaN          NaN          NaN
25%                 NaN              NaN          NaN          NaN
50%                 NaN              NaN          NaN          NaN
75%                 NaN              NaN          NaN          NaN
max                 NaN              NaN          NaN          NaN

               Sales      Quantity      Discount        Profit  Shipping Cost  \
count   51290.000000  51290.000000  51290.000000  51290.000000   51290.000000
unique           NaN           NaN           NaN           NaN            NaN
top              NaN           NaN           NaN           NaN            NaN
freq             NaN           NaN           NaN           NaN            NaN
mean      246.490581      3.476545      0.142908     28.610982      26.375915
std       487.565361      2.278766      0.212280    174.340972      57.296804
min         0.444000      1.000000      0.000000  -6599.978000       0.000000
25%        30.758625      2.000000      0.000000      0.000000       2.610000
50%        85.053000      3.000000      0.000000      9.240000       7.790000
75%       251.053200      5.000000      0.200000     36.810000      24.450000
max     22638.480000     14.000000      0.850000   8399.976000     933.570000

        Order Priority
count            51290
```

```
unique               4
top             Medium
freq             29433
mean               NaN
std                NaN
min                NaN
25%                NaN
50%                NaN
75%                NaN
max                NaN

[11 rows x 24 columns]
```

[302]:

```
# To display the first few rows of the data
df.head()
```

[302]:

```
   Row ID         Order ID  Order Date   Ship Date     Ship Mode Customer ID  \
0   32298   CA-2012-124891   7/31/2012   7/31/2012      Same Day    RH-19495
1   26341    IN-2013-77878    2/5/2013    2/7/2013  Second Class    JR-16210
2   25330    IN-2013-71249  10/17/2013  10/18/2013   First Class    CR-12730
3   13524  ES-2013-1579342   1/28/2013   1/30/2013   First Class    KM-16375
4   47221     SG-2013-4320   11/5/2013   11/6/2013      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]
```

[303]:
```
# To check if there is null values
df.isnull().sum()
```

[303]:
```
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
```

[304]:
```
# To display the dimensions of a data
df.shape
```

[304]: 
```
(51290, 24)
```

[305]:
```
# To check for basic infomation about the data
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 51290 entries, 0 to 51289
Data columns (total 24 columns):
 #   Column          Non-Null Count  Dtype
```

```
---  ------          --------------  -----
 0   Row ID          51290 non-null  int64
 1   Order ID        51290 non-null  object
 2   Order Date      51290 non-null  object
 3   Ship Date       51290 non-null  object
 4   Ship Mode       51290 non-null  object
 5   Customer ID     51290 non-null  object
 6   Customer Name   51290 non-null  object
 7   Segment         51290 non-null  object
 8   City            51290 non-null  object
 9   State           51290 non-null  object
 10  Country         51290 non-null  object
 11  Postal Code     9994 non-null   float64
 12  Market          51290 non-null  object
 13  Region          51290 non-null  object
 14  Product ID      51290 non-null  object
 15  Category        51290 non-null  object
 16  Sub-Category    51290 non-null  object
 17  Product Name    51290 non-null  object
 18  Sales           51290 non-null  float64
 19  Quantity        51290 non-null  int64
 20  Discount        51290 non-null  float64
 21  Profit          51290 non-null  float64
 22  Shipping Cost   51290 non-null  float64
 23  Order Priority  51290 non-null  object
dtypes: float64(5), int64(2), object(17)
memory usage: 9.4+ MB
```

[306]:
```python
# To count the number of unique values in each column of the data
df.nunique()
```

[306]:
```
Row ID            51290
Order ID          25035
Order Date         1430
Ship Date          1464
Ship Mode             4
Customer ID        1590
Customer Name       795
Segment               3
City               3636
State              1094
Country             147
Postal Code         631
Market                7
Region               13
Product ID        10292
Category              3
```

```
Sub-Category          17
Product Name        3788
Sales              22995
Quantity              14
Discount              27
Profit             24575
Shipping Cost      10037
Order Priority         4
dtype: int64
```

[307]:

```python
# Check for duplicates in the data
duplicates = df.duplicated()
print(f"Number of duplicate rows: {duplicates.sum()}")
```

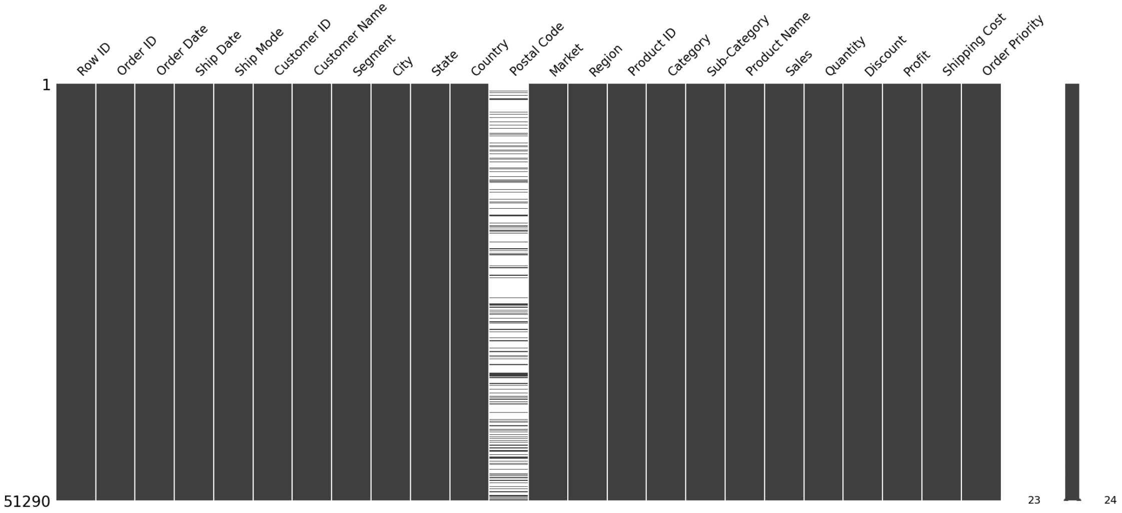
```
Number of duplicate rows: 0
```

[308]:

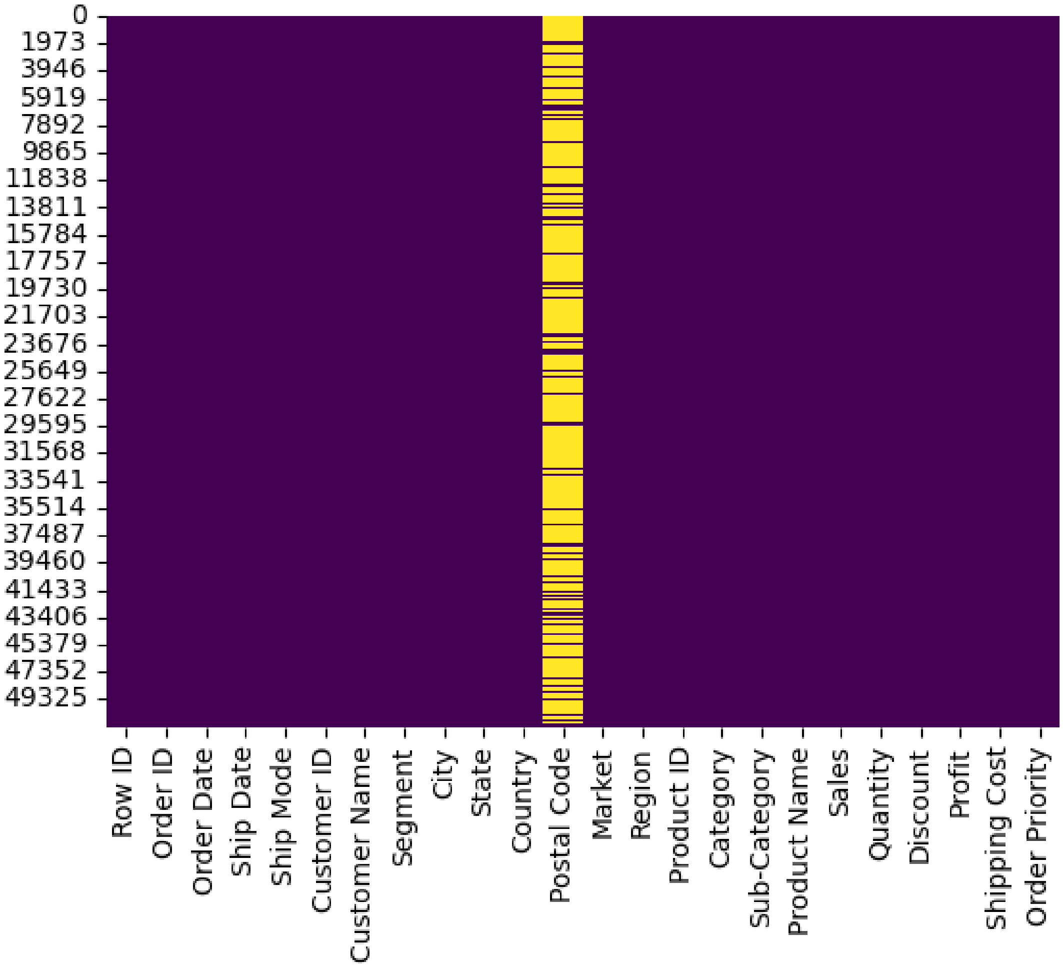
```python
# To check if there exists any null values in the data and if there exists any,␣
↪it will print preview of rows that contains it and also visualizes the␣
↪missing data.
if df.isnull().any(axis=None):
    print("\nPreview of data with null values:\nxxxxxxxxxxxxx")
    print(df[df.isnull().any(axis=1)].head(3))
    missingno.matrix(df)
    plt.show()
```

```
Preview of data with null values:
xxxxxxxxxxxxx
   Row ID         Order ID  Order Date   Ship Date     Ship Mode Customer ID  \
1   26341    IN-2013-77878    2/5/2013    2/7/2013  Second Class    JR-16210
2   25330    IN-2013-71249  10/17/2013  10/18/2013   First Class    CR-12730
3   13524  ES-2013-1579342   1/28/2013   1/30/2013   First Class    KM-16375

      Customer Name      Segment        City           State  ...  \
1     Justin Ritter    Corporate  Wollongong  New South Wales  ...
2      Craig Reiter     Consumer    Brisbane       Queensland  ...
3  Katherine Murray  Home Office      Berlin           Berlin  ...

        Product ID    Category Sub-Category  \
1  FUR-CH-10003950   Furniture       Chairs
2  TEC-PH-10004664  Technology       Phones
3  TEC-PH-10004583  Technology       Phones

                               Product Name     Sales Quantity Discount  \
1  Novimex Executive Leather Armchair, Black  3709.395        9      0.1
2          Nokia Smart Phone, with Caller ID  5175.171        9      0.1
3             Motorola Smart Phone, Cordless  2892.510        5      0.1
```

```
    Profit  Shipping Cost  Order Priority
1 -288.765         923.63        Critical
2  919.971         915.49          Medium
3  -96.540         910.16          Medium

[3 rows x 24 columns]
```



[310]:
```python
# Visualize missing data
sns.heatmap(df.isnull(), cbar=False, cmap='viridis')
plt.show()
```



[311]:

```
# Fill missing values for numerical columns with their median
numeric_cols = ['Sales', 'Quantity', 'Discount','Profit','Shipping Cost']
#numeric_cols = df.select_dtypes(include=['number']).columns
#df[numeric_cols] = df[numeric_cols].fillna(df[numeric_cols].median())

# Fill missing values for categorical columns with their mode
categorical_cols = df.select_dtypes(include=['object']).columns
#df[categorical_cols] = df[categorical_cols].fillna(df[categorical_cols].mode().
 ↪iloc[0])
```

**Since we have no missing values in attributes other than postal code, we are not going to use missing value handling here. In case of a data which has missing values present, use codeblock provided above.**

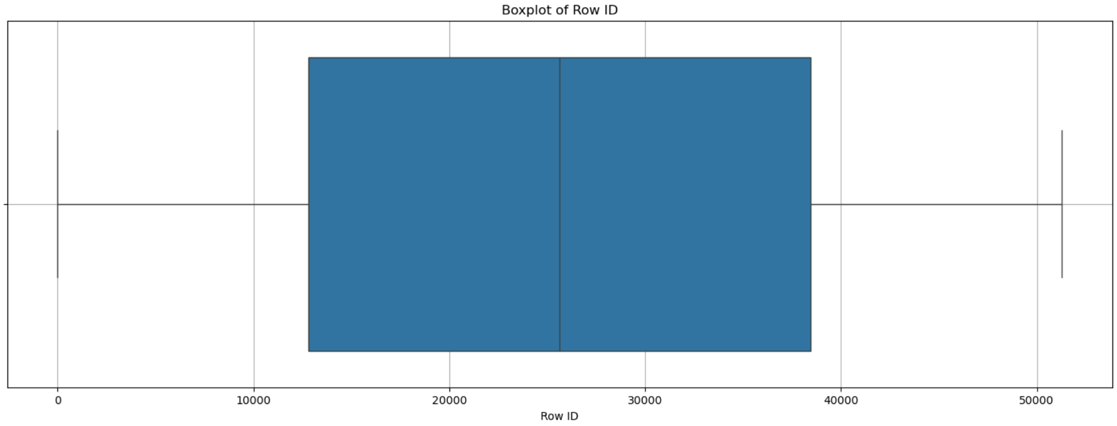
[315]:

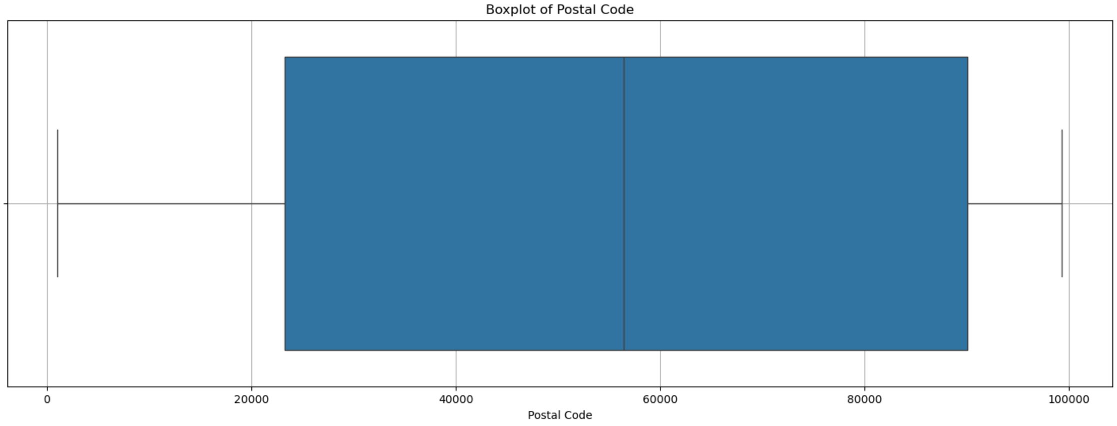
```
#To get the summary statistics of data and to display the statistics more␣
 ↪clearly
display(df.describe().T)
```

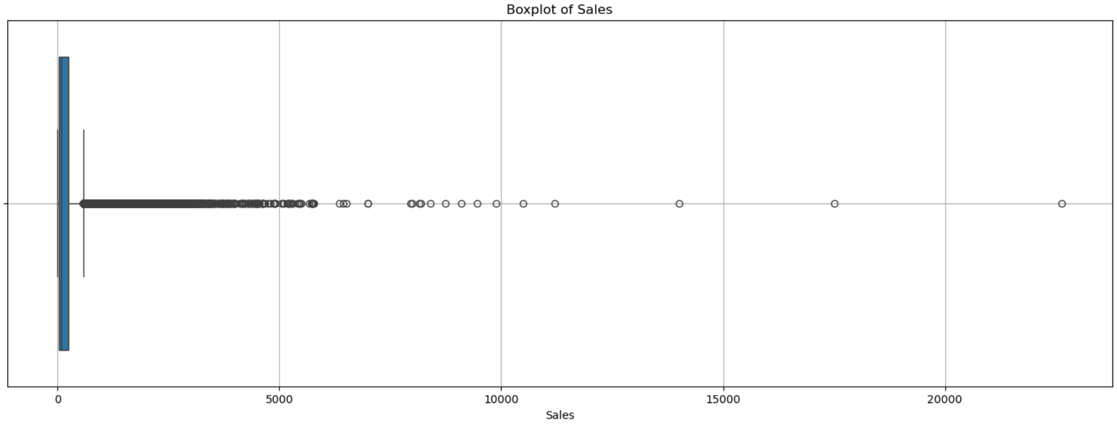
```
                count          mean           std       min           25%  \
Row ID        51290.0  25645.500000  14806.291990     1.000  12823.250000
Postal Code    9994.0  55190.379428  32063.693350  1040.000  23223.000000
Sales         51290.0    246.490581    487.565361     0.444     30.758625
Quantity      51290.0      3.476545      2.278766     1.000      2.000000
Discount      51290.0      0.142908      0.212280     0.000      0.000000
Profit        51290.0     28.610982    174.340972 -6599.978      0.000000
Shipping Cost 51290.0     26.375915     57.296804     0.000      2.610000

                    50%         75%        max
Row ID        25645.500  38467.7500  51290.000
Postal Code   56430.500  90008.0000  99301.000
Sales            85.053    251.0532  22638.480
Quantity          3.000      5.0000     14.000
Discount          0.000      0.2000      0.850
Profit            9.240     36.8100   8399.976
Shipping Cost     7.790     24.4500    933.570
```

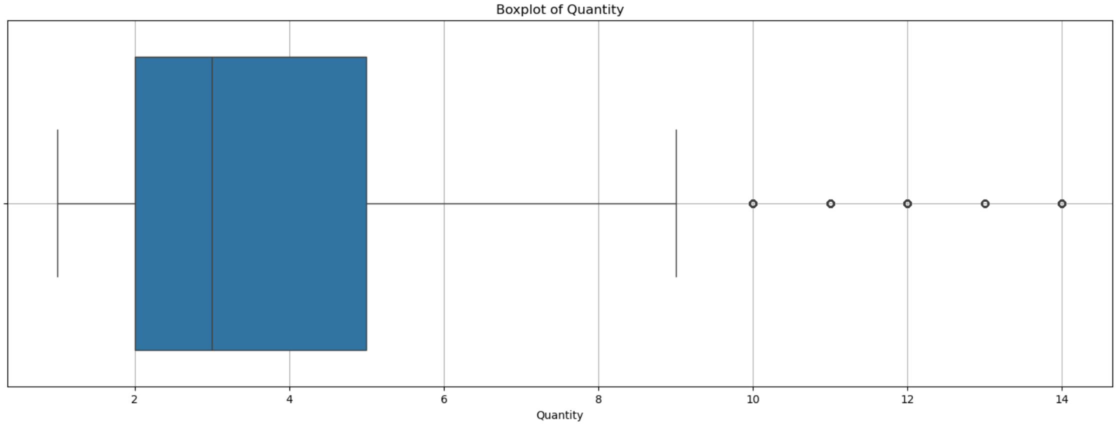
[320]:

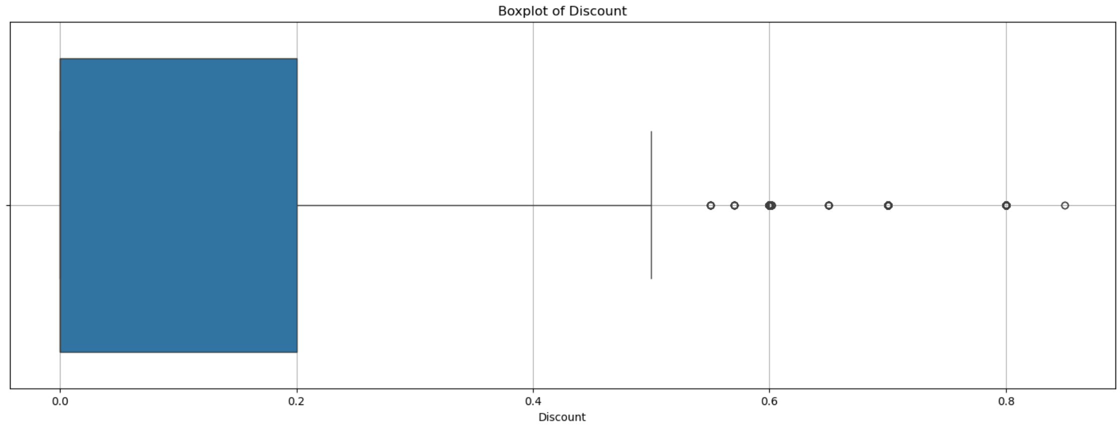
```python
# Boxplot to detect outliers in numerical columns
for col in df.select_dtypes(include=['float64', 'int64']).columns:
    plt.figure(figsize=(18, 6))
    sns.boxplot(x=df[col])
    plt.title(f'Boxplot of {col}')
    plt.grid(True)
    plt.show()
```

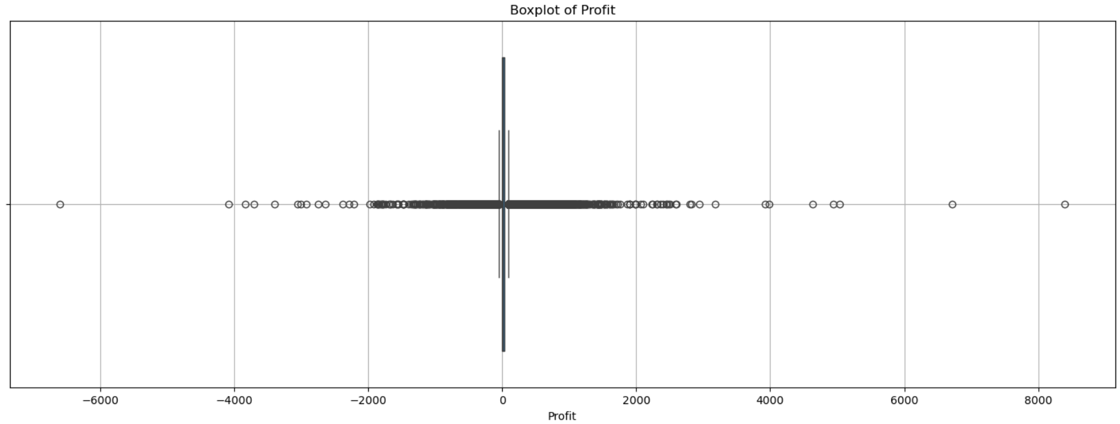


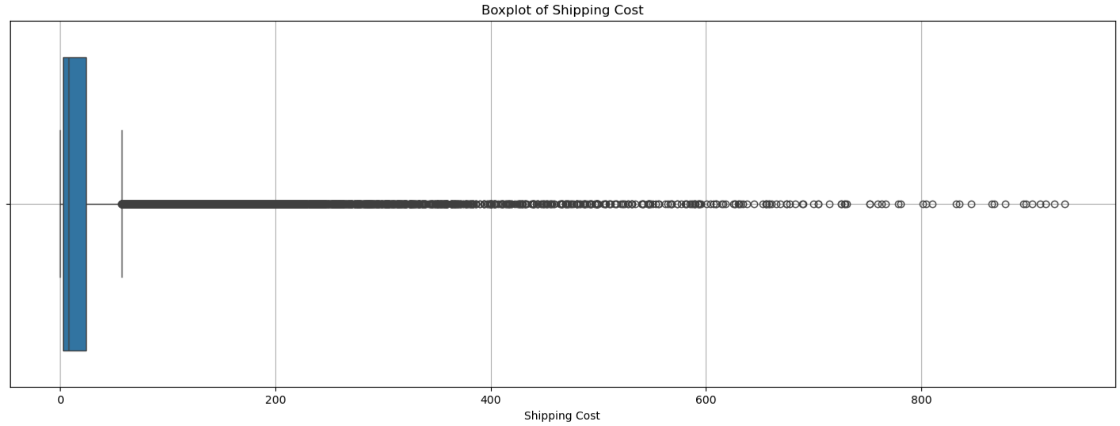

Boxplot of Postal Code
0
20000
40000
60000
80000
100000
Postal Code


Boxplot of Sales
0
5000
10000
15000
20000
Sales


Boxplot of Quantity
2
4
6
8
10
12
14
Quantity


Boxplot of Discount
0.0
0.2
0.4
0.6
0.8
Discount


Boxplot of Profit
−6000
−4000
−2000
0
2000
4000
6000
8000
Profit


Boxplot of Shipping Cost
0
200
400
600
800
Shipping Cost

```
[323]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 51290 entries, 0 to 51289
Data columns (total 24 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Row ID          51290 non-null  int64
 1   Order ID        51290 non-null  object
 2   Order Date      51290 non-null  object
 3   Ship Date       51290 non-null  object
 4   Ship Mode       51290 non-null  object
 5   Customer ID     51290 non-null  object
 6   Customer Name   51290 non-null  object
 7   Segment         51290 non-null  object
 8   City            51290 non-null  object
 9   State           51290 non-null  object
 10  Country         51290 non-null  object
 11  Postal Code     9994 non-null   float64
 12  Market          51290 non-null  object
 13  Region          51290 non-null  object
 14  Product ID      51290 non-null  object
 15  Category        51290 non-null  object
 16  Sub-Category    51290 non-null  object
 17  Product Name    51290 non-null  object
 18  Sales           51290 non-null  float64
 19  Quantity        51290 non-null  int64
 20  Discount        51290 non-null  float64
 21  Profit          51290 non-null  float64
 22  Shipping Cost   51290 non-null  float64
 23  Order Priority  51290 non-null  object
dtypes: float64(5), int64(2), object(17)
memory usage: 9.4+ MB
```

[324]:

```
# Check for null values
df.isnull().sum()
```

[324]:

```
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
```

[325]:

```python
# To remove outliers from our data by using Z-score and IQR methods to identify␣
 ↪and filter out anomalous data points.
from scipy import stats

def remove_outliers(df, numeric_cols, z_score_threshold=3, iqr_factor=1.5):
    """Removes outliers from a DataFrame using Z-score and IQR.

    Args:
        df: The DataFrame containing the data.
        numeric_cols: A list of column names to apply the outlier detection to.
        z_score_threshold: The Z-score threshold for identifying outliers.␣
 ↪Default is 3.
        iqr_factor: The IQR factor for identifying outliers. Default is 1.5.

    Returns:
        A DataFrame with outliers removed.
    """
    # Z-score method
    z_scores = np.abs(stats.zscore(df[numeric_cols]))
    is_outlier_zscore = (z_scores >= z_score_threshold).any(axis=1)

    # IQR method
    Q1 = df[numeric_cols].quantile(0.25)
    Q3 = df[numeric_cols].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - iqr_factor * IQR
    upper_bound = Q3 + iqr_factor * IQR
    is_outlier_iqr = ((df[numeric_cols] < lower_bound) | (df[numeric_cols] >␣
 ↪upper_bound)).any(axis=1)

    # Combine both methods using OR logic
    is_outlier = is_outlier_zscore | is_outlier_iqr
    # Remove rows with outliers
    df_clean = df[~is_outlier]

    return df_clean
```

[326]:

```
# To remove outliers from numeric columns our data
df[numeric_cols] = remove_outliers(df[numeric_cols], numeric_cols)
```

[327]:

```
# To Check null values after remove outliers
df.isnull().sum()
```

[327]:

```
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             14979
Quantity          14979
Discount          14979
Profit            14979
Shipping Cost     14979
Order Priority        0
dtype: int64
```

[329]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 51290 entries, 0 to 51289
Data columns (total 24 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Row ID          51290 non-null  int64
 1   Order ID        51290 non-null  object
 2   Order Date      51290 non-null  object
 3   Ship Date       51290 non-null  object
 4   Ship Mode       51290 non-null  object
 5   Customer ID     51290 non-null  object
 6   Customer Name   51290 non-null  object
 7   Segment         51290 non-null  object
 8   City            51290 non-null  object
 9   State           51290 non-null  object
 10  Country         51290 non-null  object
 11  Postal Code     9994 non-null   float64
 12  Market          51290 non-null  object
 13  Region          51290 non-null  object
 14  Product ID      51290 non-null  object
 15  Category        51290 non-null  object
 16  Sub-Category    51290 non-null  object
 17  Product Name    51290 non-null  object
 18  Sales           36311 non-null  float64
 19  Quantity        36311 non-null  float64
 20  Discount        36311 non-null  float64
 21  Profit          36311 non-null  float64
 22  Shipping Cost   36311 non-null  float64
 23  Order Priority  51290 non-null  object
dtypes: float64(6), int64(1), object(17)
memory usage: 9.4+ MB
```

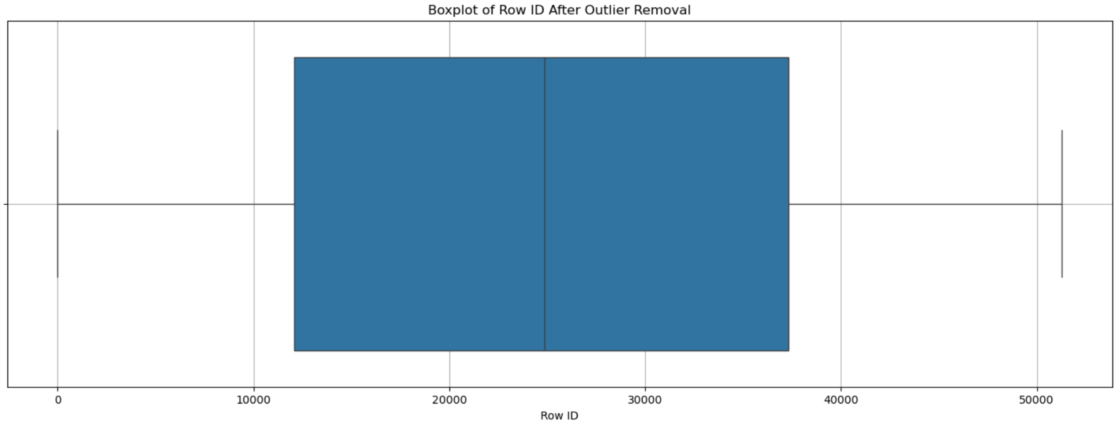
[330]:
```
# To drop the 'Postal Code' column and to drop any rows with missing values␣
 ↪(NaNs).
df =df[df.columns.drop('Postal Code')].dropna()
```

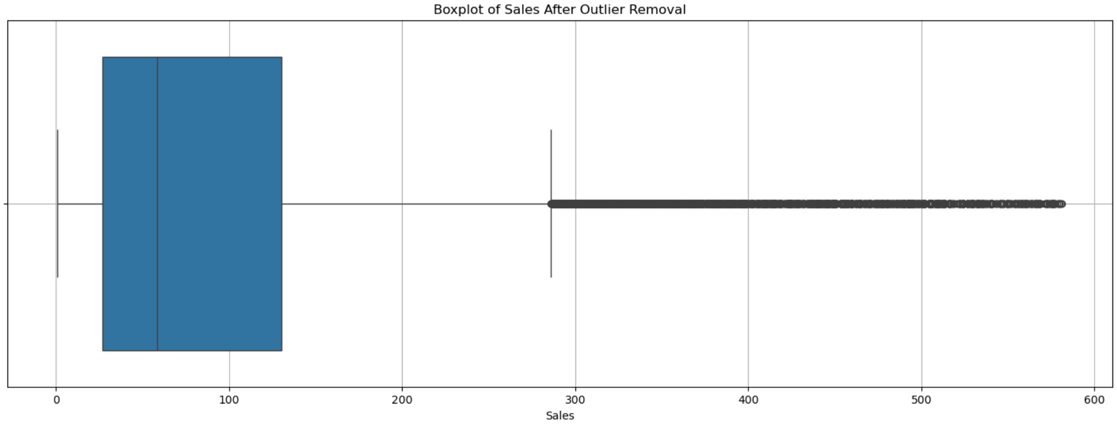
[331]:
```
df.info()
```

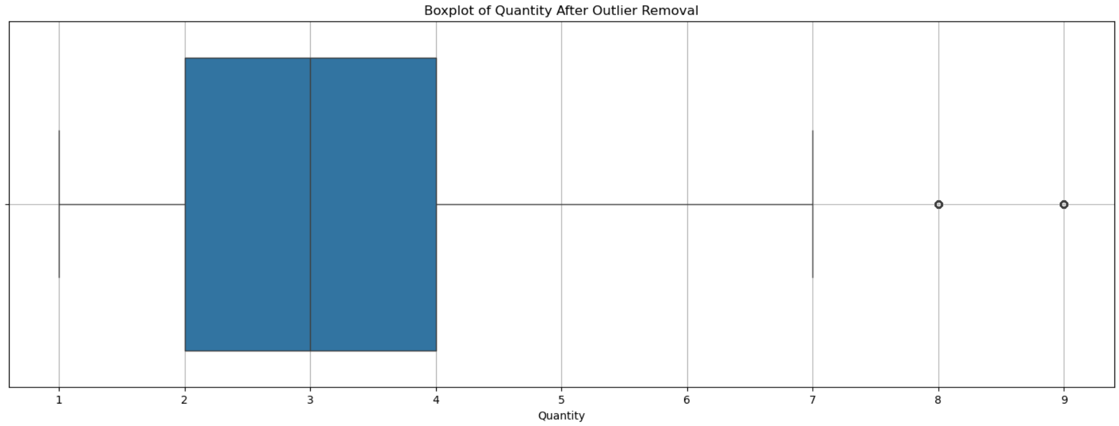
```
<class 'pandas.core.frame.DataFrame'>
Index: 36311 entries, 5909 to 51289
Data columns (total 23 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Row ID          36311 non-null  int64
 1   Order ID        36311 non-null  object
 2   Order Date      36311 non-null  object
 3   Ship Date       36311 non-null  object
 4   Ship Mode       36311 non-null  object
 5   Customer ID     36311 non-null  object
 6   Customer Name   36311 non-null  object
 7   Segment         36311 non-null  object
 8   City            36311 non-null  object
 9   State           36311 non-null  object
 10  Country         36311 non-null  object
 11  Market          36311 non-null  object
 12  Region          36311 non-null  object
 13  Product ID      36311 non-null  object
 14  Category        36311 non-null  object
 15  Sub-Category    36311 non-null  object
 16  Product Name    36311 non-null  object
 17  Sales           36311 non-null  float64
 18  Quantity        36311 non-null  float64
 19  Discount        36311 non-null  float64
 20  Profit          36311 non-null  float64
 21  Shipping Cost   36311 non-null  float64
 22  Order Priority  36311 non-null  object
dtypes: float64(5), int64(1), object(17)
memory usage: 6.6+ MB
```

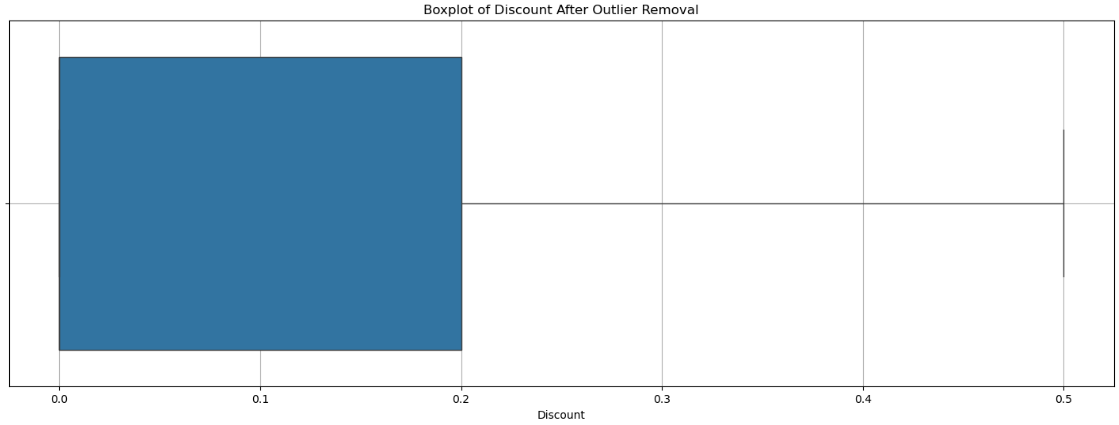
[332]:

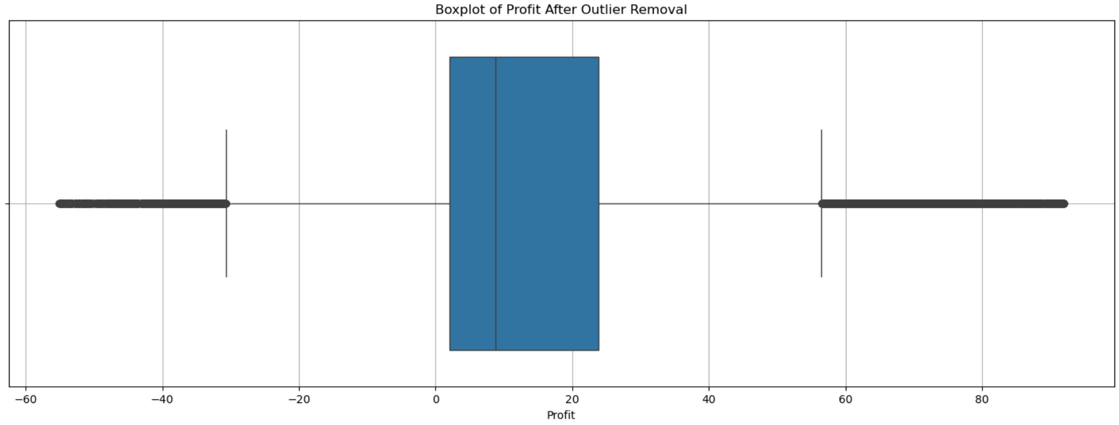
```
#Boxplots after outlier removal
for col in df.select_dtypes(include=['float64', 'int64']).columns:
    plt.figure(figsize=(18, 6))
    sns.boxplot(x=df[col])
    plt.title(f'Boxplot of {col} After Outlier Removal')
    plt.grid(True)
    plt.show()
```

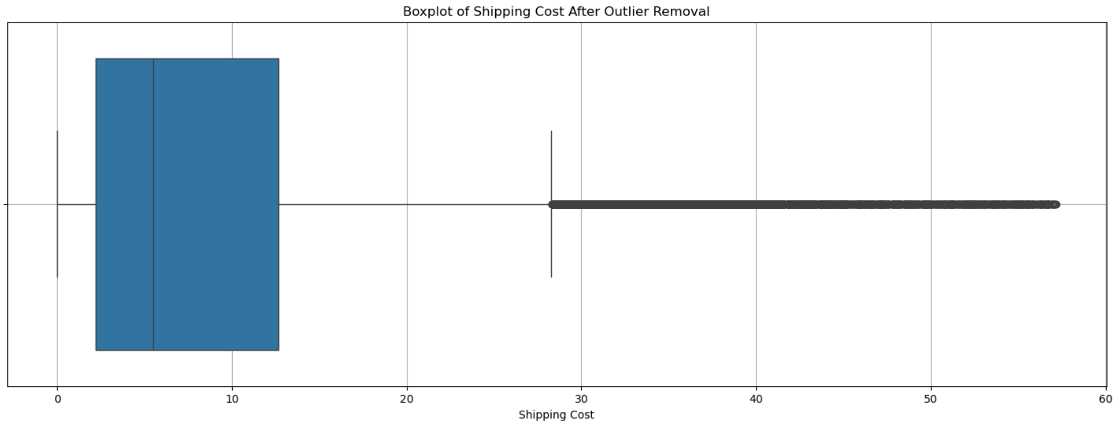





Boxplot of Quantity After Outlier Removal
1
2
3
4
5
6
7
8
9
Quantity


Boxplot of Discount After Outlier Removal
0.0
0.1
0.2
0.3
0.4
0.5
Discount

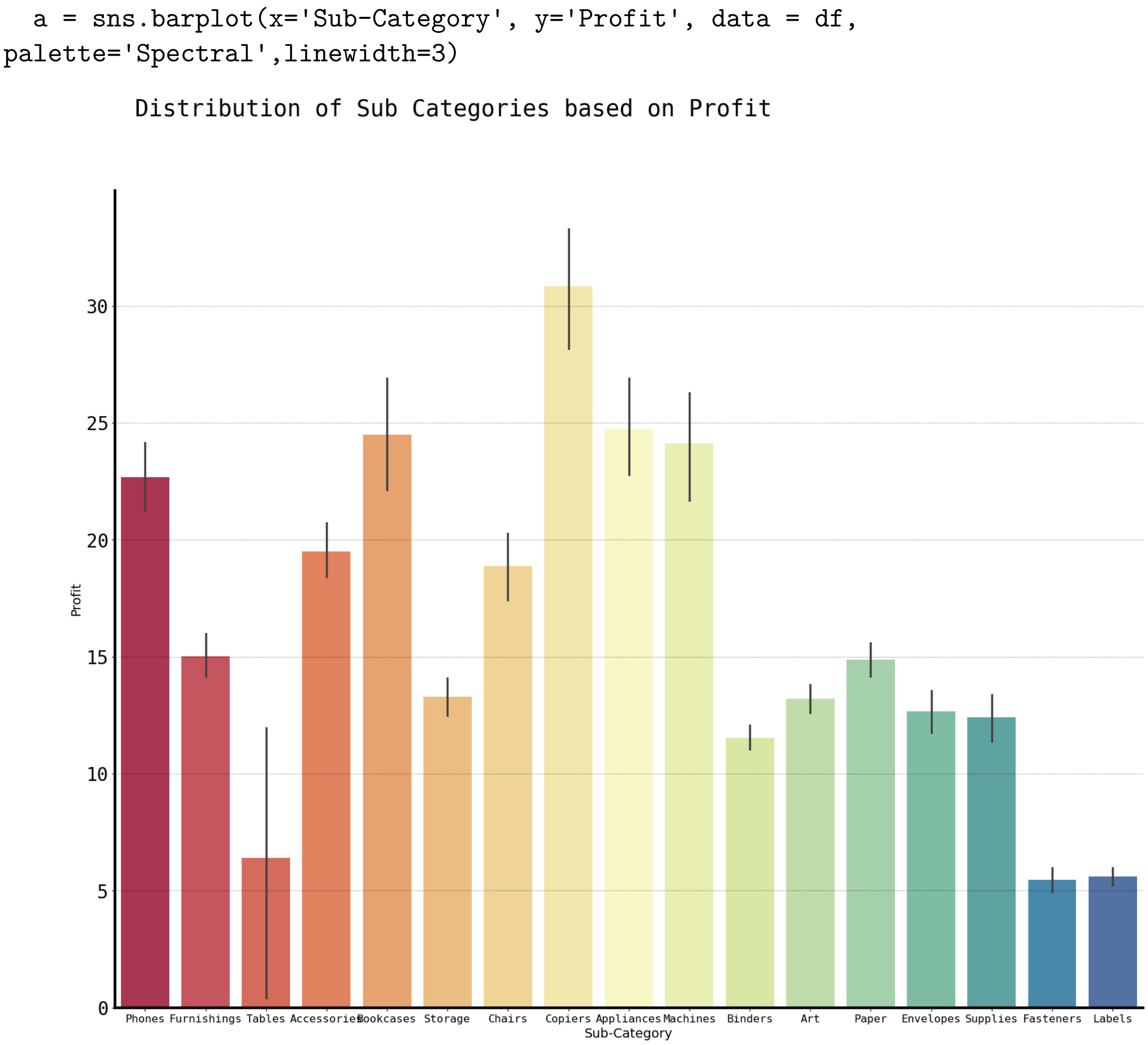

Boxplot of Profit After Outlier Removal
−60
−40
−20
0
20
40
60
80
Profit



[333]:

```
# To give a bar plot to show distribution of profits with different␣
↪sub-categories in our data.
plt.figure(figsize = (20,16))

a = sns.barplot(x='Sub-Category', y='Profit', data = df,␣
↪palette='Spectral',linewidth=3)

plt.figtext(x=0.14, y=0.95,
            s='Distribution of Sub Categories based on Profit',
            fontsize=25, fontname='monospace')

plt.xticks(fontsize=12, fontname='monospace')
plt.yticks(fontsize=20, fontname='monospace')
plt.xlabel('Sub-Category', fontsize=14)
plt.ylabel('Profit', fontsize=14)

plt.grid(axis='y', color='black', linestyle = ':', alpha=0.5)

for q in [a]:
    for w in ['bottom', 'left']:
        q.spines[w].set_linewidth(3)
    for w in ['right', 'top']:
        q.spines[w].set_visible(False)

plt.show()
```

```
C:\Users\PARVATHY MENON\AppData\Local\Temp\ipykernel_11080\2958157332.py:4:
FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same
effect.
```

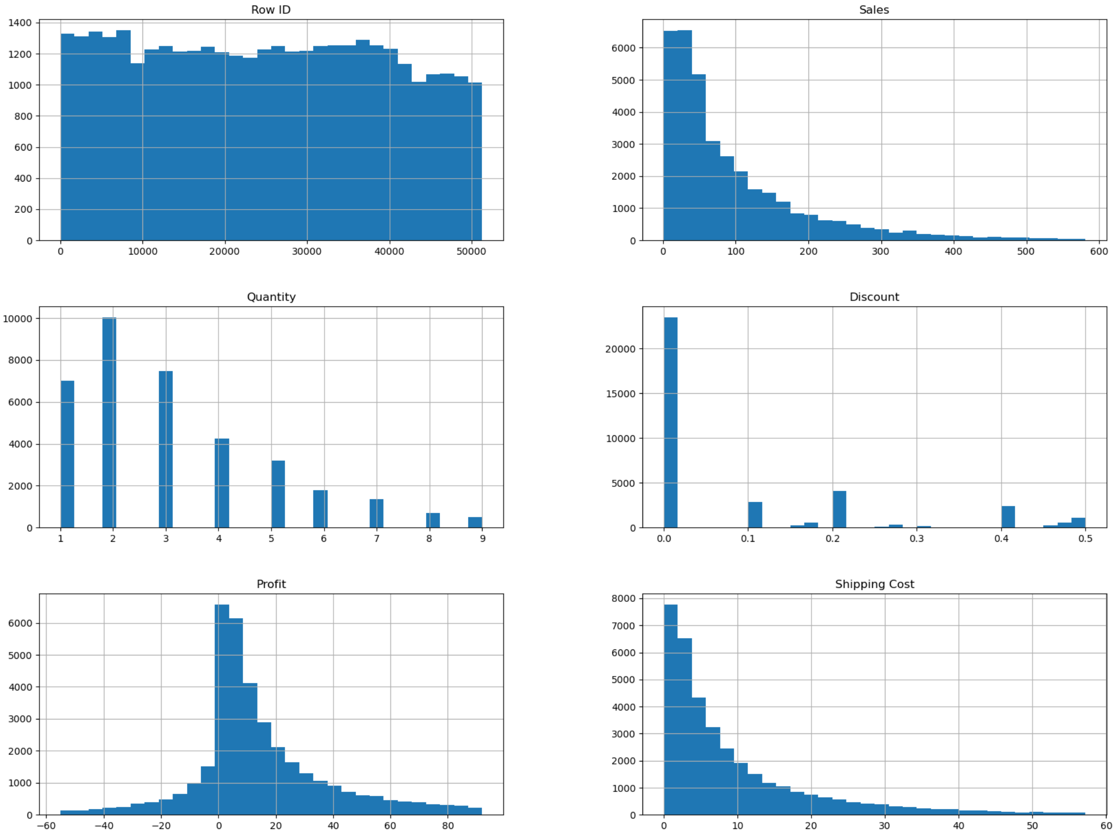


```
  a = sns.barplot(x='Sub-Category', y='Profit', data = df,
palette='Spectral',linewidth=3)
```

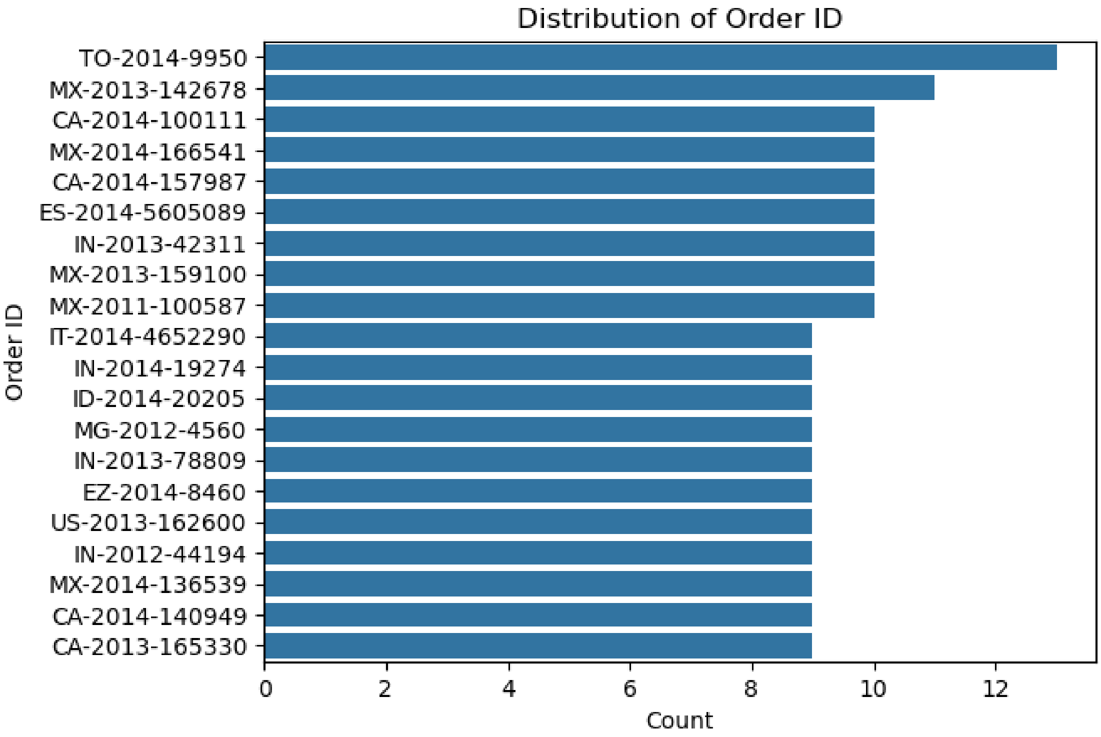
Distribution of Sub Categories based on Profit

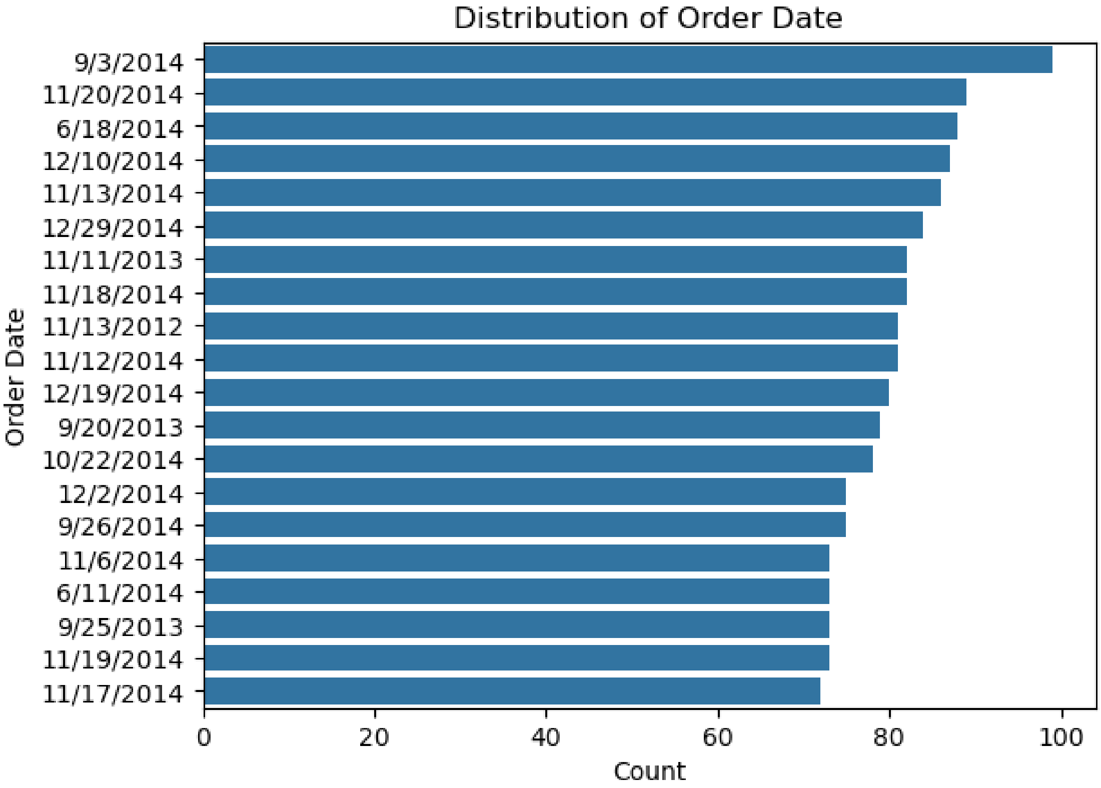
[334]:

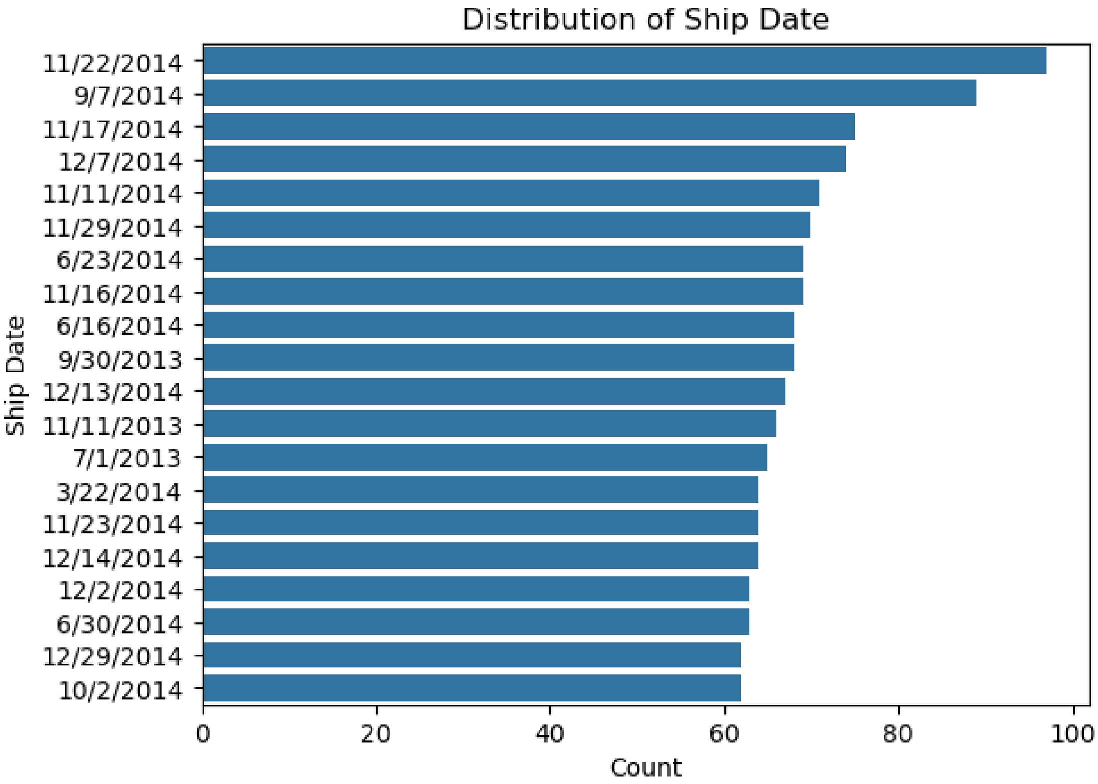
```
#To draw histograms for numerical columns
df.hist(bins=30, figsize=(20, 15))
plt.show()
```

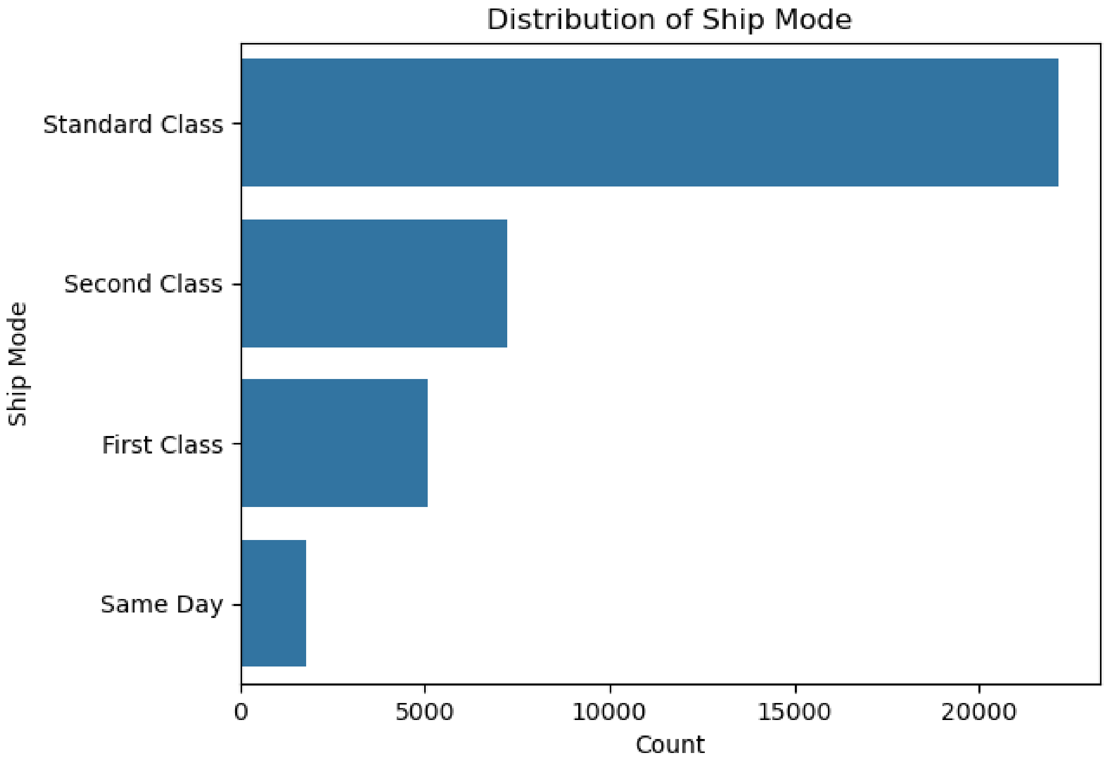


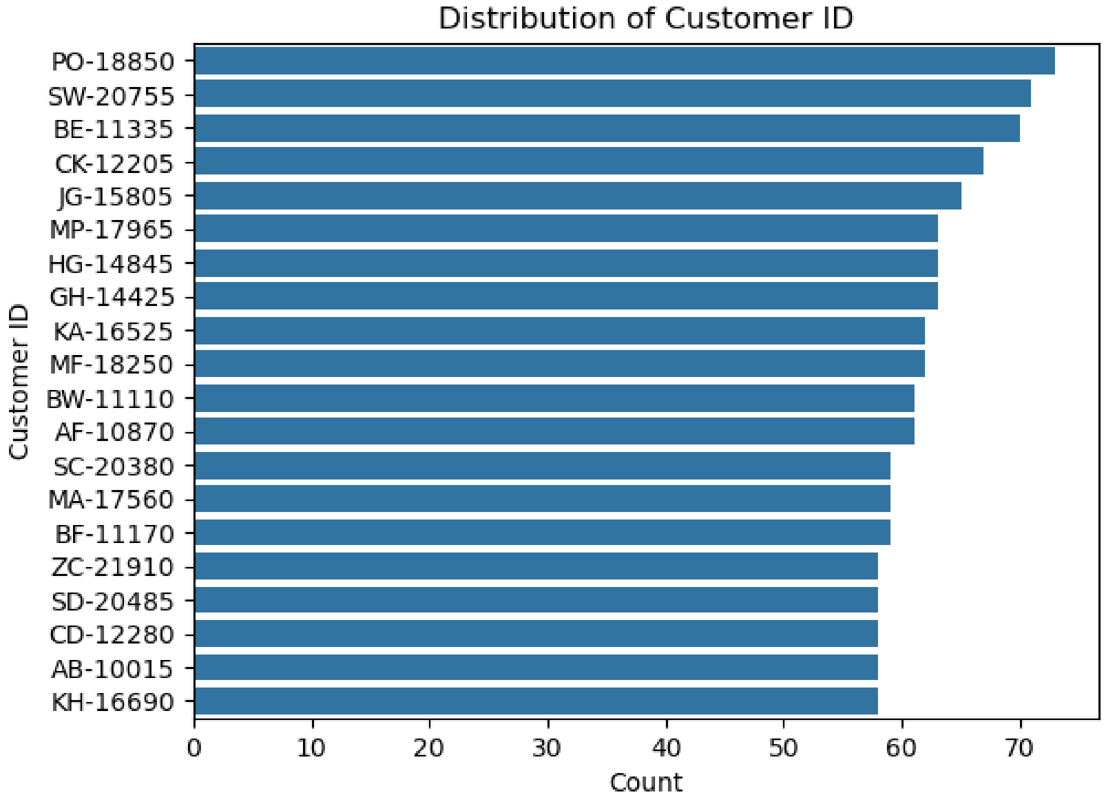
```
[335]: # To draw count plots for all categorical columns in your DataFrame
categorical_cols = df.select_dtypes(include=['object']).columns
for col in categorical_cols:
    sns.countplot(y=col, data=df, order=df[col].value_counts().index[:20])  #␣
 ↪Limit to top 20 categories
    plt.title(f'Distribution of {col}')
    plt.xlabel('Count')
    plt.ylabel(col)
    plt.show()
```

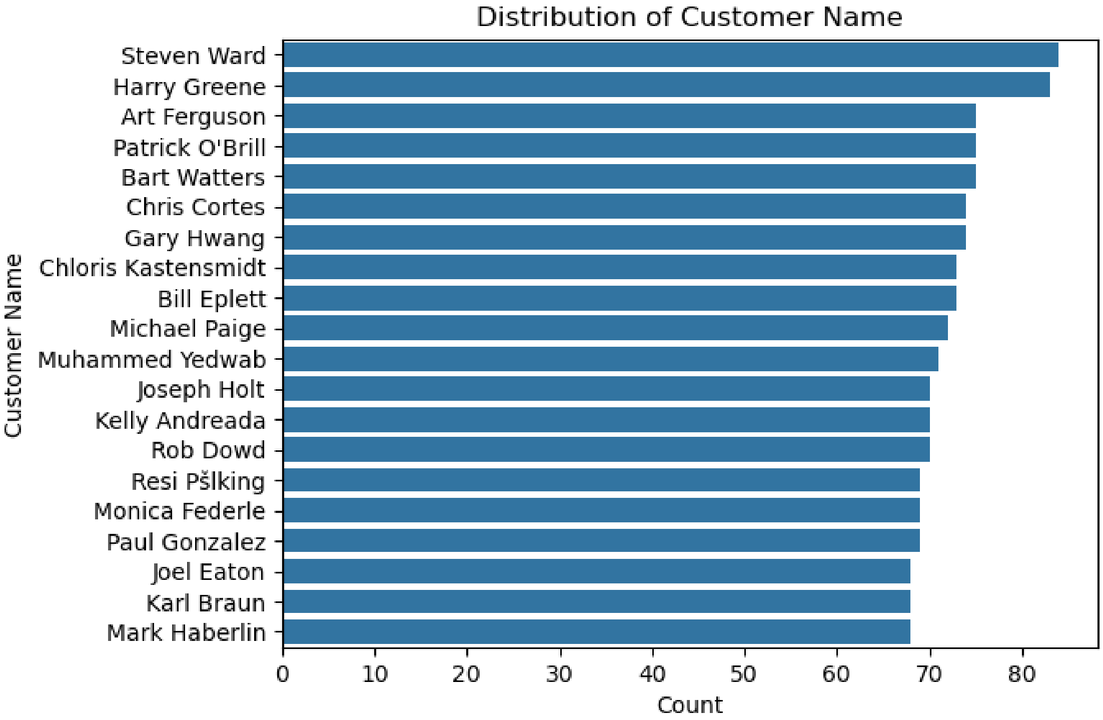

Distribution of Order ID
Order ID
TO-2014-9950
MX-2013-142678
CA-2014-100111
MX-2014-166541
CA-2014-157987
ES-2014-5605089
IN-2013-42311
MX-2013-159100
MX-2011-100587
IT-2014-4652290
IN-2014-19274
ID-2014-20205
MG-2012-4560
IN-2013-78809
EZ-2014-8460
US-2013-162600
IN-2012-44194
MX-2014-136539
CA-2014-140949
CA-2013-165330
0
2
4
6
8
10
12
Count

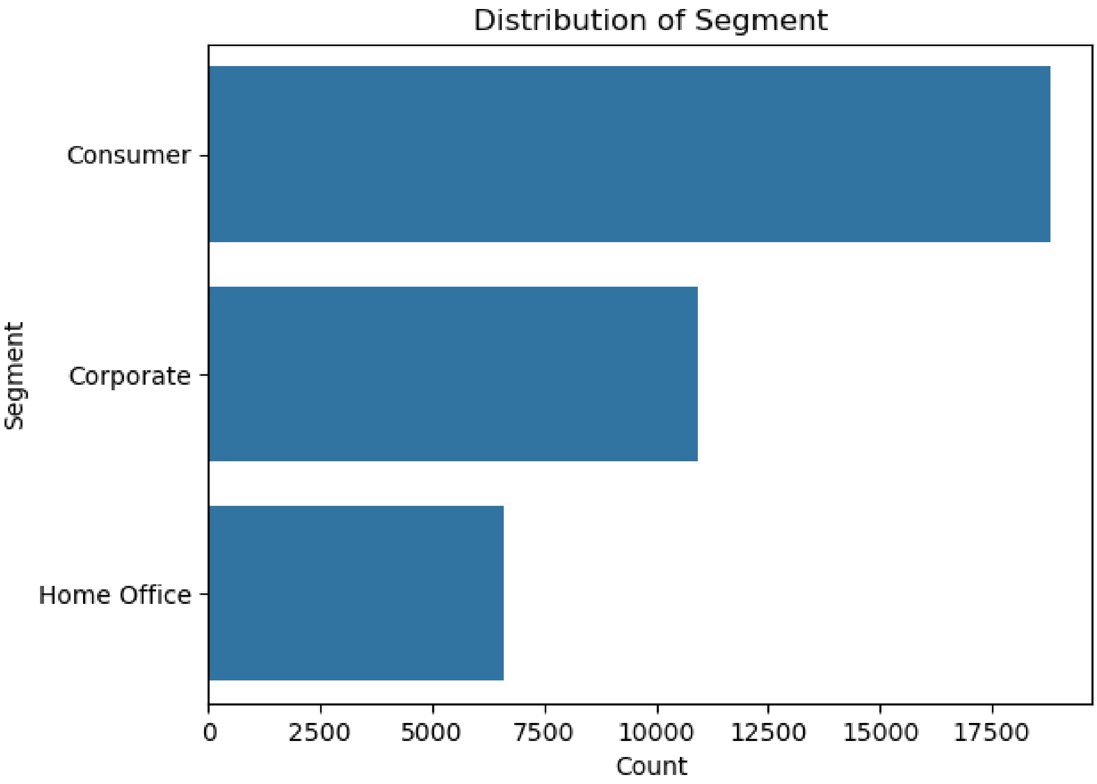

Distribution of Order Date
Order Date
9/3/2014
11/20/2014
6/18/2014
12/10/2014
11/13/2014
12/29/2014
11/11/2013
11/18/2014
11/13/2012
11/12/2014
12/19/2014
9/20/2013
10/22/2014
12/2/2014
9/26/2014
11/6/2014
6/11/2014
9/25/2013
11/19/2014
11/17/2014
0
20
40
60
80
100
Count

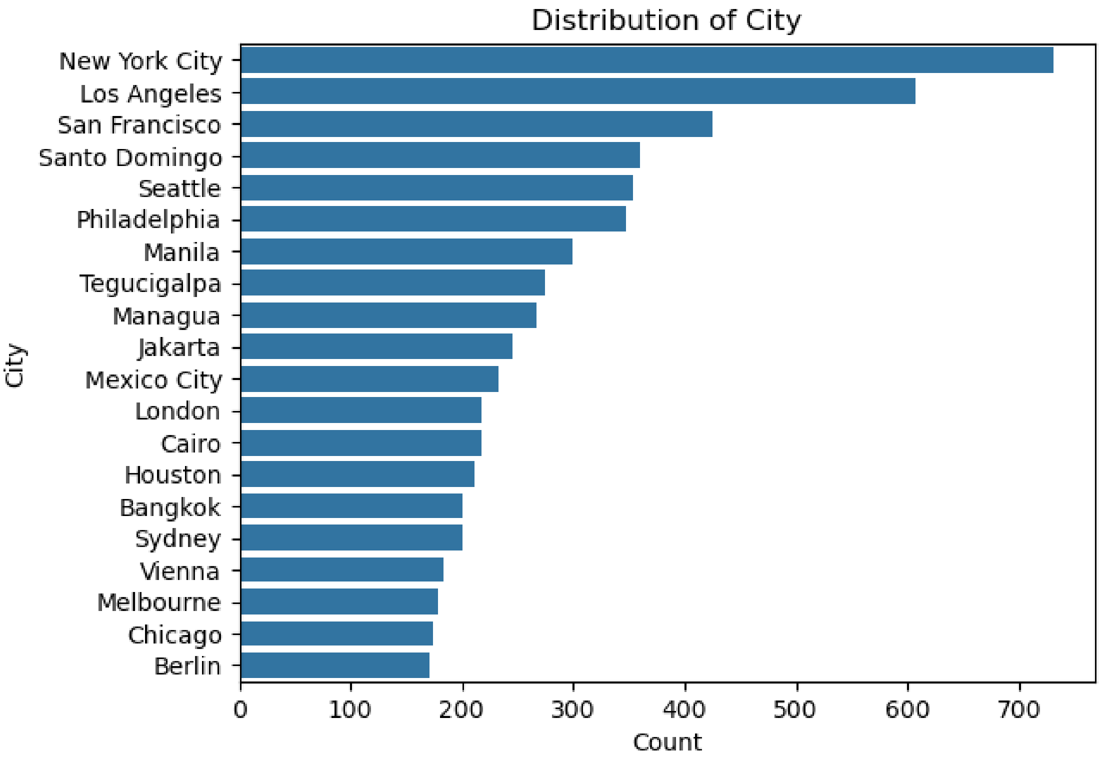

Distribution of Ship Date
Ship Date
11/22/2014
9/7/2014
11/17/2014
12/7/2014
11/11/2014
11/29/2014
6/23/2014
11/16/2014
6/16/2014
9/30/2013
12/13/2014
11/11/2013
7/1/2013
3/22/2014
11/23/2014
12/14/2014
12/2/2014
6/30/2014
12/29/2014
10/2/2014
0
20
40
60
80
100
Count

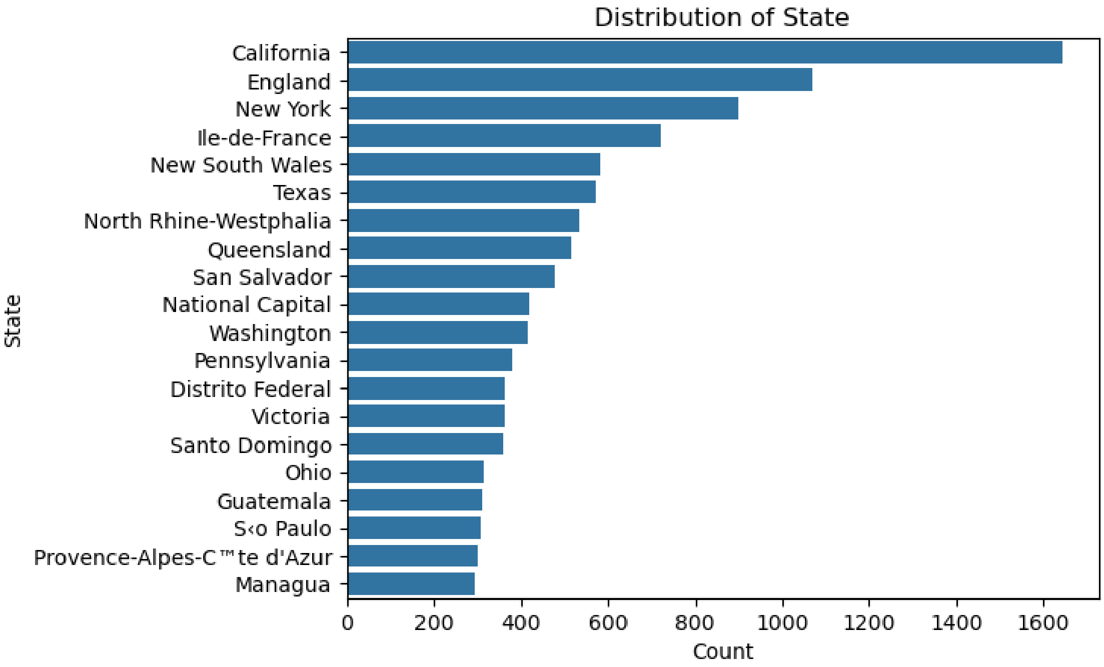

Distribution of Ship Mode
Ship Mode
Standard Class
Second Class
First Class
Same Day
0
5000
10000
15000
20000
Count

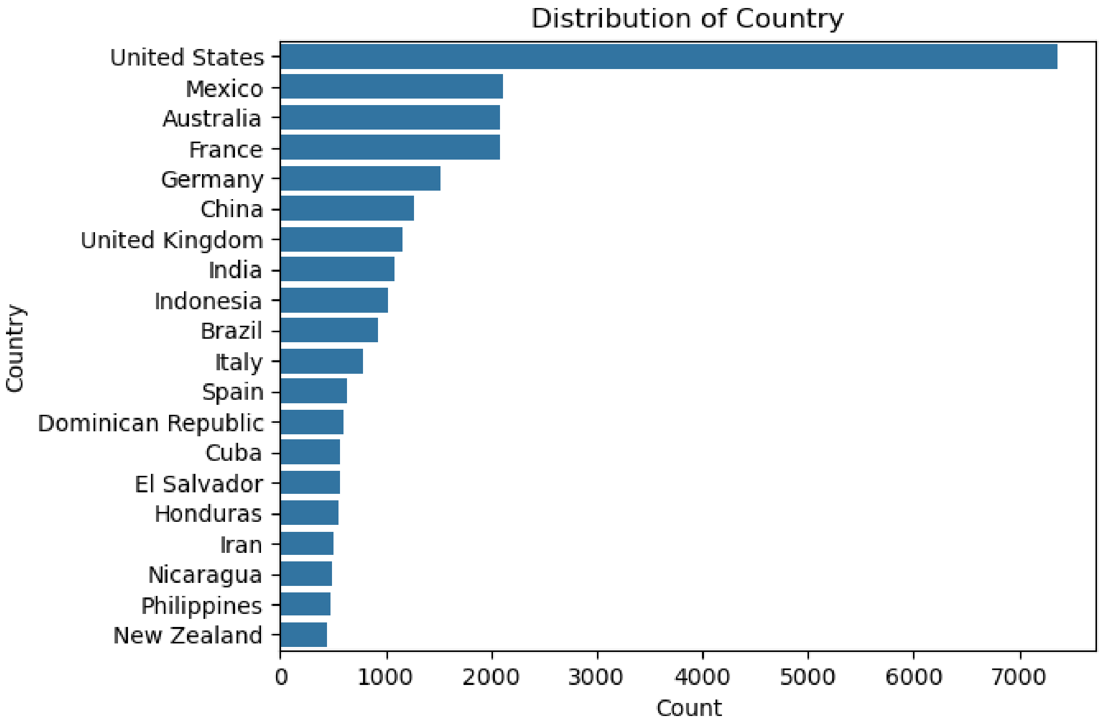

Distribution of Customer ID
Customer ID
PO-18850
SW-20755
BE-11335
CK-12205
JG-15805
MP-17965
HG-14845
GH-14425
KA-16525
MF-18250
BW-11110
AF-10870
SC-20380
MA-17560
BF-11170
ZC-21910
SD-20485
CD-12280
AB-10015
KH-16690
0
10
20
30
40
50
60
70
Count

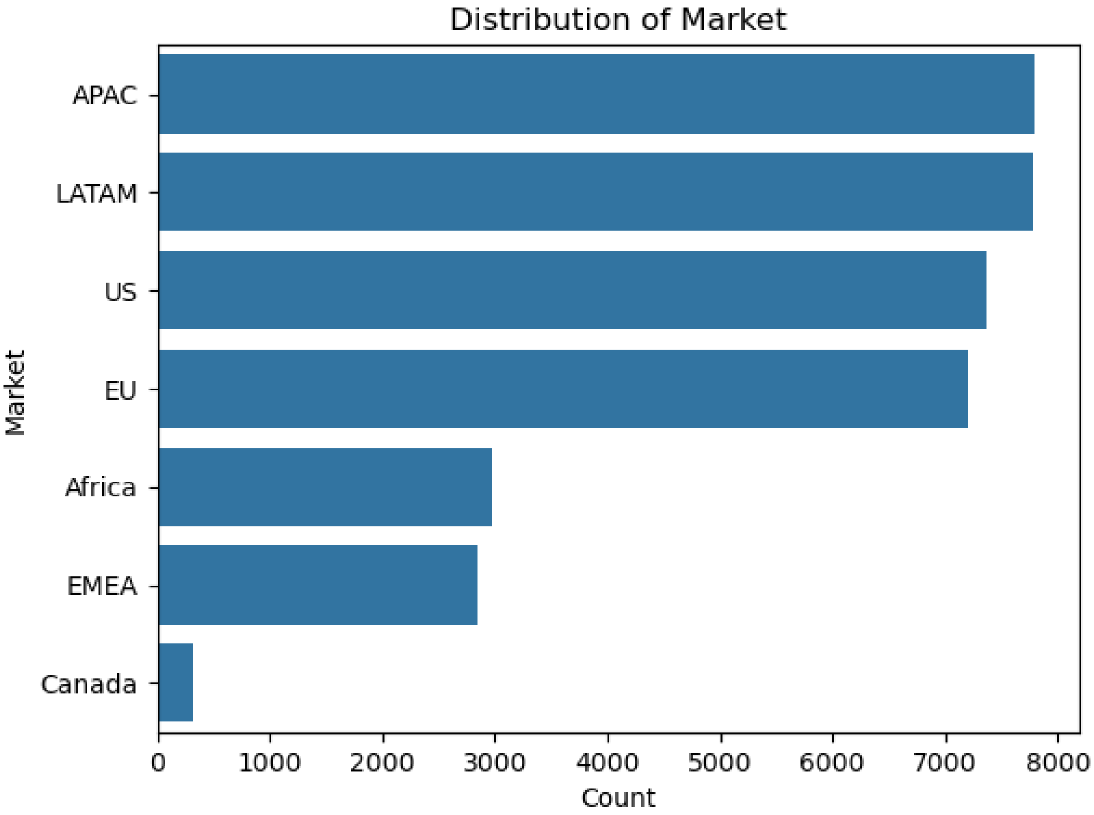

Distribution of Customer Name
Customer Name
Steven Ward
Harry Greene
Art Ferguson
Patrick O'Brill
Bart Watters
Chris Cortes
Gary Hwang
Chloris Kastensmidt
Bill Eplett
Michael Paige
Muhammed Yedwab
Joseph Holt
Kelly Andreada
Rob Dowd
Resi Pšlking
Monica Federle
Paul Gonzalez
Joel Eaton
Karl Braun
Mark Haberlin
0
10
20
30
40
50
60
70
80
Count

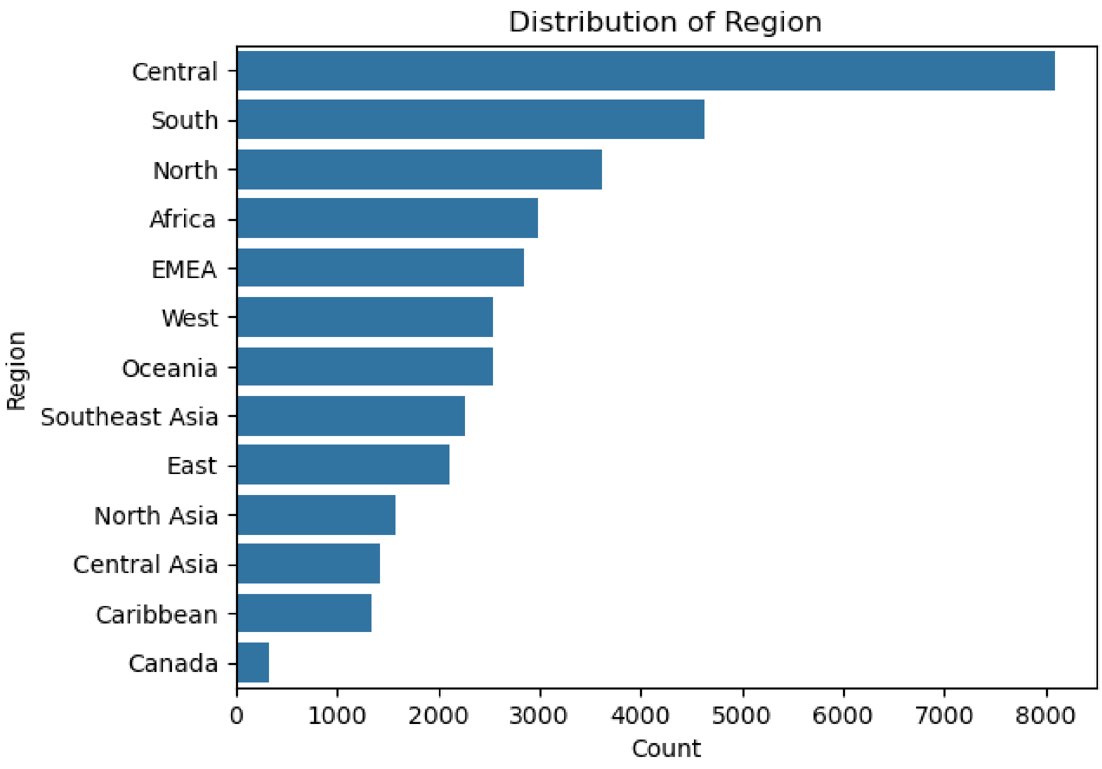

Distribution of Segment
Consumer
Corporate
Home Office
Segment
0
2500
5000
7500
10000
12500
15000
17500
Count

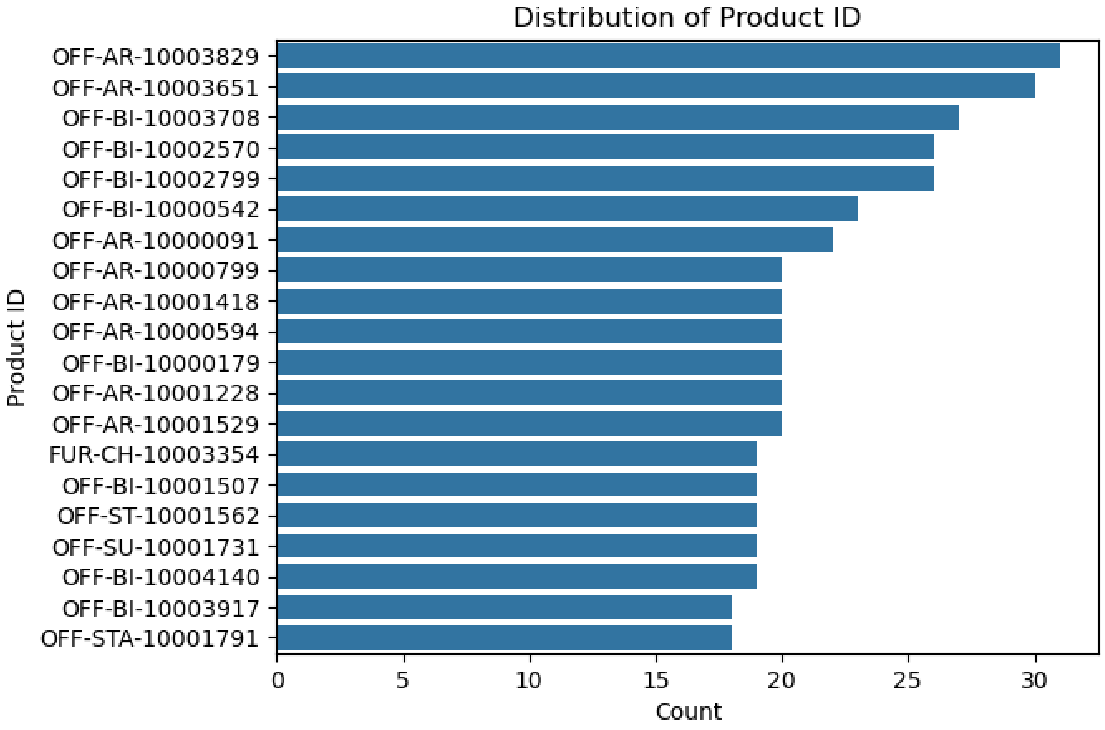

Distribution of City
New York City
Los Angeles
San Francisco
Santo Domingo
Seattle
Philadelphia
Manila
Tegucigalpa
Managua
Jakarta
Mexico City
London
Cairo
Houston
Bangkok
Sydney
Vienna
Melbourne
Chicago
Berlin
City
0
100
200
300
400
500
600
700
Count

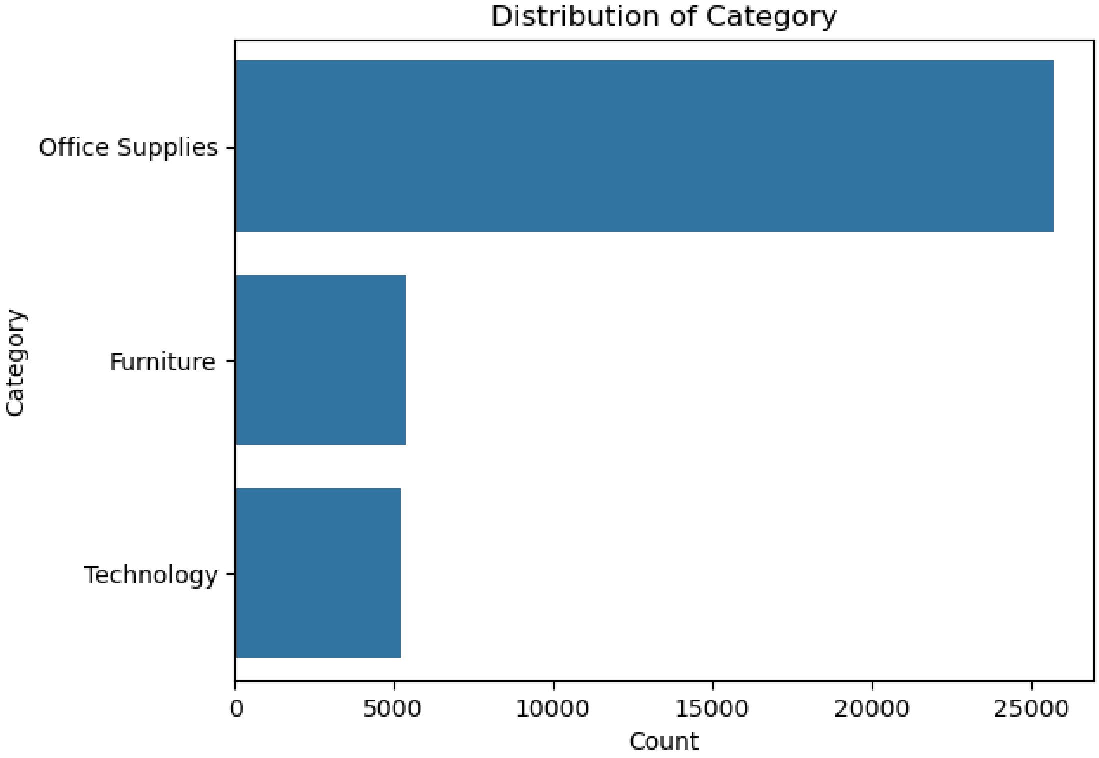

Distribution of State
State
California
England
New York
Ile-de-France
New South Wales
Texas
North Rhine-Westphalia
Queensland
San Salvador
National Capital
Washington
Pennsylvania
Distrito Federal
Victoria
Santo Domingo
Ohio
Guatemala
S‹o Paulo
Provence-Alpes-C™te d'Azur
Managua
0
200
400
600
800
1000
1200
1400
1600
Count

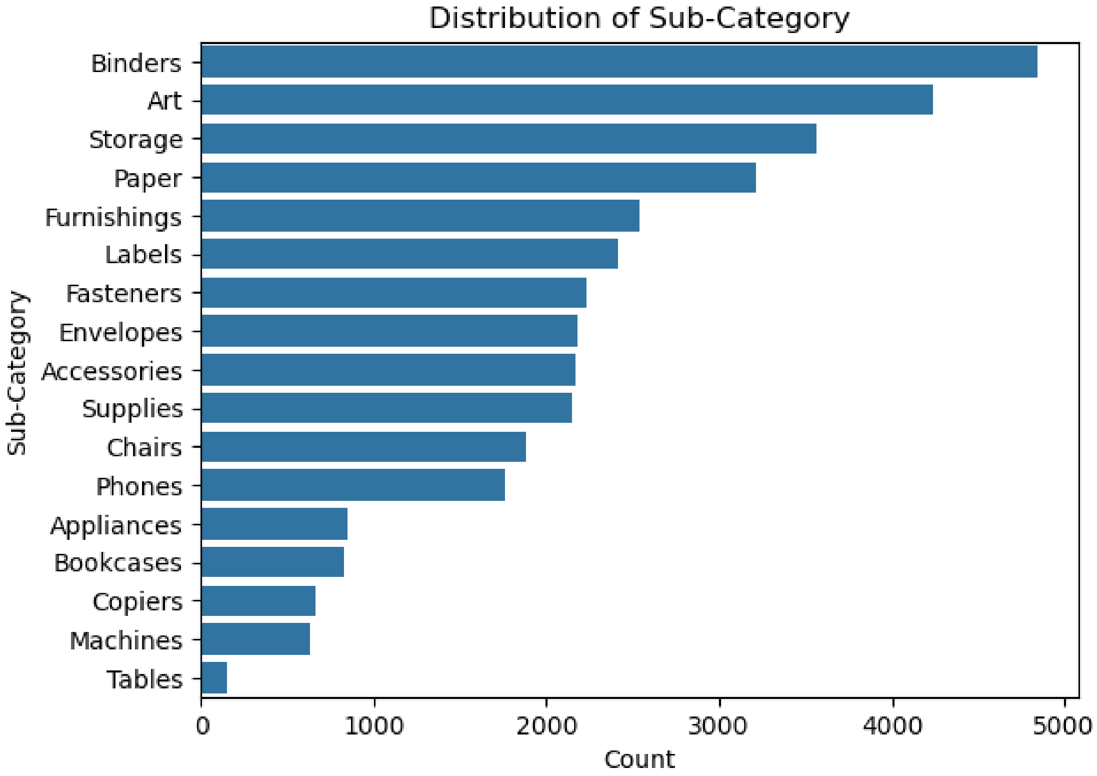

Distribution of Country
Country
United States
Mexico
Australia
France
Germany
China
United Kingdom
India
Indonesia
Brazil
Italy
Spain
Dominican Republic
Cuba
El Salvador
Honduras
Iran
Nicaragua
Philippines
New Zealand
0
1000
2000
3000
4000
5000
6000
7000
Count

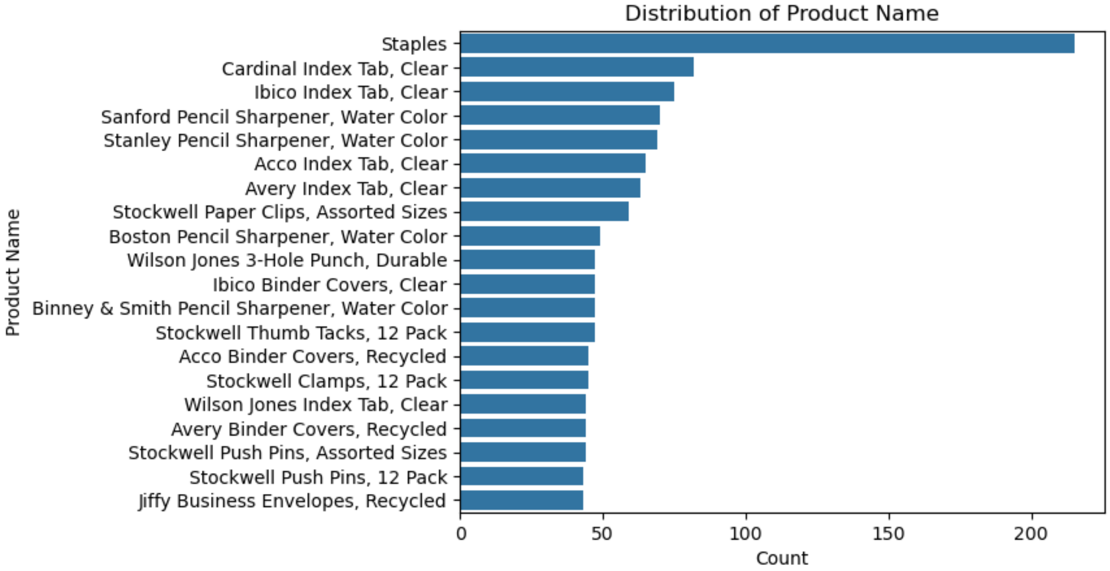

Distribution of Market
Market
APAC
LATAM
US
EU
Africa
EMEA
Canada
0
1000
2000
3000
4000
5000
6000
7000
8000
Count

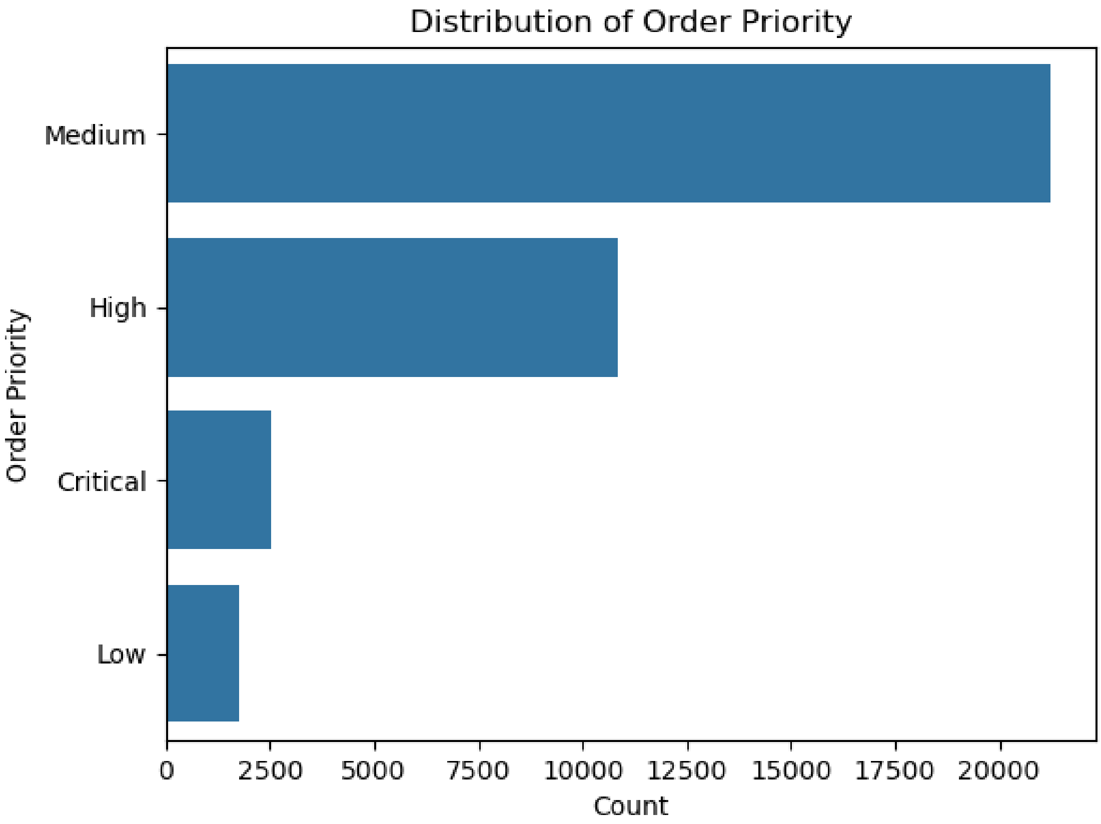

Distribution of Region
Region
Central
South
North
Africa
EMEA
West
Oceania
Southeast Asia
East
North Asia
Central Asia
Caribbean
Canada
0
1000
2000
3000
4000
5000
6000
7000
8000
Count


Distribution of Product ID
Product ID
OFF-AR-10003829
OFF-AR-10003651
OFF-BI-10003708
OFF-BI-10002570
OFF-BI-10002799
OFF-BI-10000542
OFF-AR-10000091
OFF-AR-10000799
OFF-AR-10001418
OFF-AR-10000594
OFF-BI-10000179
OFF-AR-10001228
OFF-AR-10001529
FUR-CH-10003354
OFF-BI-10001507
OFF-ST-10001562
OFF-SU-10001731
OFF-BI-10004140
OFF-BI-10003917
OFF-STA-10001791
0
5
10
15
20
25
30
Count

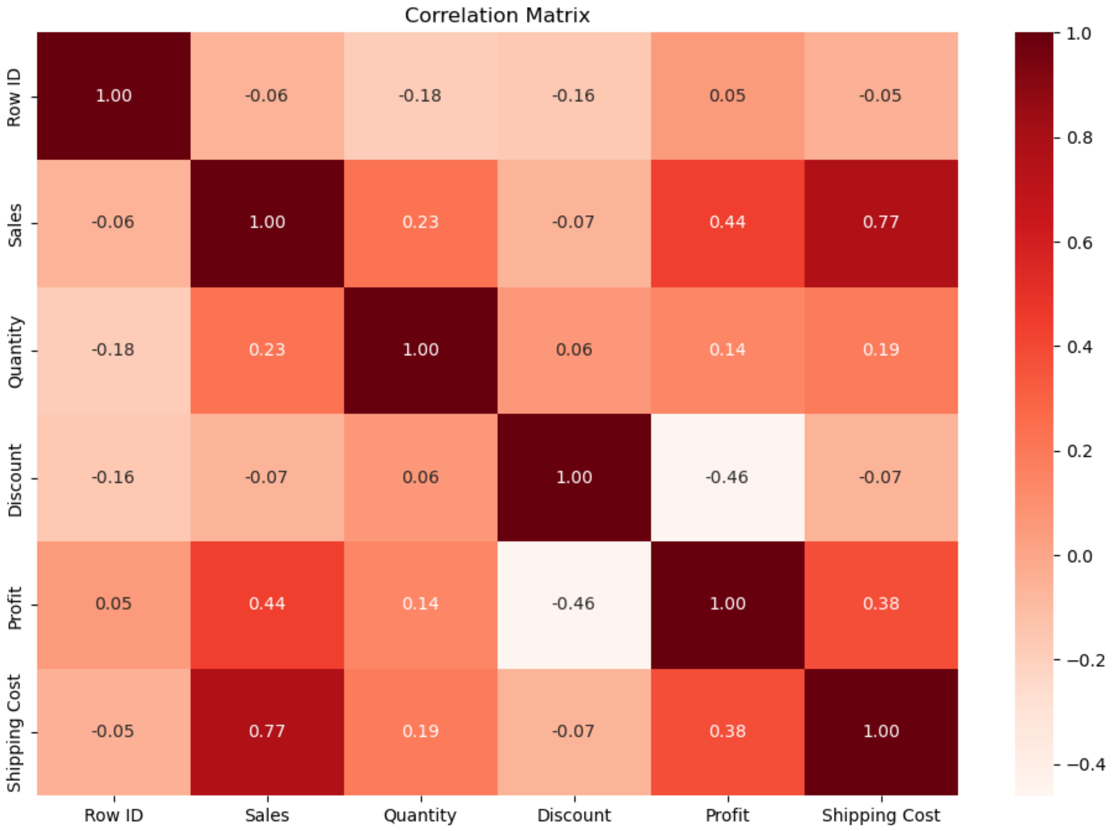

Distribution of Category
Office Supplies
Furniture
Technology
Category
0
5000
10000
15000
20000
25000
Count


Distribution of Sub-Category
Sub-Category
Binders
Art
Storage
Paper
Furnishings
Labels
Fasteners
Envelopes
Accessories
Supplies
Chairs
Phones
Appliances
Bookcases
Copiers
Machines
Tables
0
1000
2000
3000
4000
5000
Count


Distribution of Product Name
Product Name
Staples
Cardinal Index Tab, Clear
Ibico Index Tab, Clear
Sanford Pencil Sharpener, Water Color
Stanley Pencil Sharpener, Water Color
Acco Index Tab, Clear
Avery Index Tab, Clear
Stockwell Paper Clips, Assorted Sizes
Boston Pencil Sharpener, Water Color
Wilson Jones 3-Hole Punch, Durable
Ibico Binder Covers, Clear
Binney & Smith Pencil Sharpener, Water Color
Stockwell Thumb Tacks, 12 Pack
Acco Binder Covers, Recycled
Stockwell Clamps, 12 Pack
Wilson Jones Index Tab, Clear
Avery Binder Covers, Recycled
Stockwell Push Pins, Assorted Sizes
Stockwell Push Pins, 12 Pack
Jiffy Business Envelopes, Recycled
0
50
100
150
200
Count



[336]:
```
# Computes and visualizes correlation matrix for all numeric columns in our␣
↪data.
numeric_df = df.select_dtypes(include=['number'])
# Compute the correlation matrix
corr_matrix = numeric_df.corr()
# Plot the correlation matrix
plt.figure(figsize=(12, 8))
sns.heatmap(corr_matrix, annot=True, cmap='Reds', fmt=".2f")
plt.title("Correlation Matrix")
plt.show()
```



Strong Positive Correlations is seen for:

```
Sales and Shipping Cost (0.77)
Sales and Profit (0.44)
Shipping Cost and Profit (0.38)
```

Moderate Positive Correlations is for:

```
Quantity and Sales (0.23)
```

And Negative Correlationsis seen for:

```
Discount and Profit (-0.46)
```

[341]:
```python
# To find total profit for each category
profit = df.groupby('Category')['Profit'].sum()
profit
```

[341]:
```
Category
Furniture           94863.92310
Office Supplies    308072.31270
Technology         117791.25908
Name: Profit, dtype: float64
```

[342]:

```
#Create a new Dataframe loss which contains only rows from df where the␣
↪'Profit' column has negative values(indicating a loss).
loss = df[df['Profit'] < 0]
loss
```

[342]:

```
       Row ID         Order ID  Order Date   Ship Date       Ship Mode  \
5922    30952    ID-2014-85529    5/8/2014   5/12/2014  Standard Class
5940    12345  IT-2013-1143457    4/9/2013   4/15/2013  Standard Class
5968    18804  IT-2014-5800736   6/11/2014   6/17/2014  Standard Class
6005    17157  ES-2013-5266365    9/4/2013    9/7/2013    Second Class
6023    39226   CA-2014-167549   7/26/2014   7/28/2014     First Class
…       …              …         …         …              …
51232    3569   US-2013-148705  12/16/2013  12/18/2013    Second Class
51234   22652    IN-2014-46686  11/12/2014  11/16/2014  Standard Class
51235   24004    IN-2011-43298   11/3/2011   11/5/2011    Second Class
51253   29192    ID-2014-66174    7/8/2014   7/13/2014  Standard Class
51267   27081    ID-2012-36725   8/10/2012   8/11/2012     First Class

      Customer ID        Customer Name      Segment      City  \
5922     AG-10300  Aleksandra Gannaway    Corporate  Auckland
5940     HM-14980    Henry MacAllister     Consumer     Paris
5968     HL-15040         Hunter Lopez     Consumer    Oviedo
6005     HG-14965        Henry Goldwyn    Corporate     Niort
6023     EM-14200        Evan Minnotte  Home Office    Dallas
…           …                  …          …       …
51232    GK-14620          Grace Kelly    Corporate   La Vega
51234    EJ-13720            Ed Jacobs     Consumer     Busan
51235    MV-17485        Mark Van Huff     Consumer  Peshawar
51253    JE-15610              Jim Epp    Corporate    Lahore
51267    EK-13795        Eileen Kiefer  Home Office     Pasig

                     State  …       Product ID         Category Sub-Category  \
5922              Auckland  …  FUR-BO-10001872        Furniture    Bookcases
5940         Ile-de-France  …  TEC-PH-10003492       Technology       Phones
5968              Asturias  …  OFF-ST-10001758  Office Supplies      Storage
6005      Poitou-Charentes  …  OFF-AP-10000005  Office Supplies   Appliances
6023                 Texas  …  FUR-TA-10004767        Furniture       Tables
…                   …  …              …               …          …
51232              La Vega  …  FUR-FU-10002698        Furniture  Furnishings
51234                Busan  …  OFF-SU-10001877  Office Supplies     Supplies
51235   Khyber Pakhtunkhwa  …  OFF-LA-10000436  Office Supplies       Labels
51253               Punjab  …  OFF-PA-10001653  Office Supplies        Paper
51267     National Capital  …  OFF-PA-10004968  Office Supplies        Paper

                                      Product Name    Sales Quantity Discount  \
5922                 Bush Stackable Bookrack, Pine  449.496      6.0     0.40
5940                   Cisco Audio Dock, Full Size  467.568      3.0     0.15
```

```
5968                          Rogers Lockers, Blue  380.916      2.0     0.10
6005                     KitchenAid Microwave, Red  558.576      2.0     0.10
6023                          Safco Drafting Table  298.116      6.0     0.30
…                                                …       …      …      …
51232                   Deflect-O Clock, Erganomic   48.930      3.0     0.50
51234                      Elite Ruler, High Speed    6.900      1.0     0.50
51235   Novimex File Folder Labels, 5000 Label Set   17.280      4.0     0.50
51253             Green Bar Memo Slips, Multicolor   18.360      2.0     0.50
51267  Enermax Computer Printout Paper, Multicolor   49.302      3.0     0.45

        Profit  Shipping Cost  Order Priority
5922  -30.0240          57.05            High
5940   -0.0720          56.90             Low
5968  -12.7440          56.63             Low
6005  -18.6240          56.26          Medium
6023   -4.2588          56.12            High
…          …            …             …
51232  -2.9700           0.05          Medium
51234  -0.8400           0.05          Medium
51235 -13.9200           0.05          Medium
51253 -13.2600           0.04          Medium
51267 -18.8280           0.03          Medium

[5775 rows x 23 columns]
```

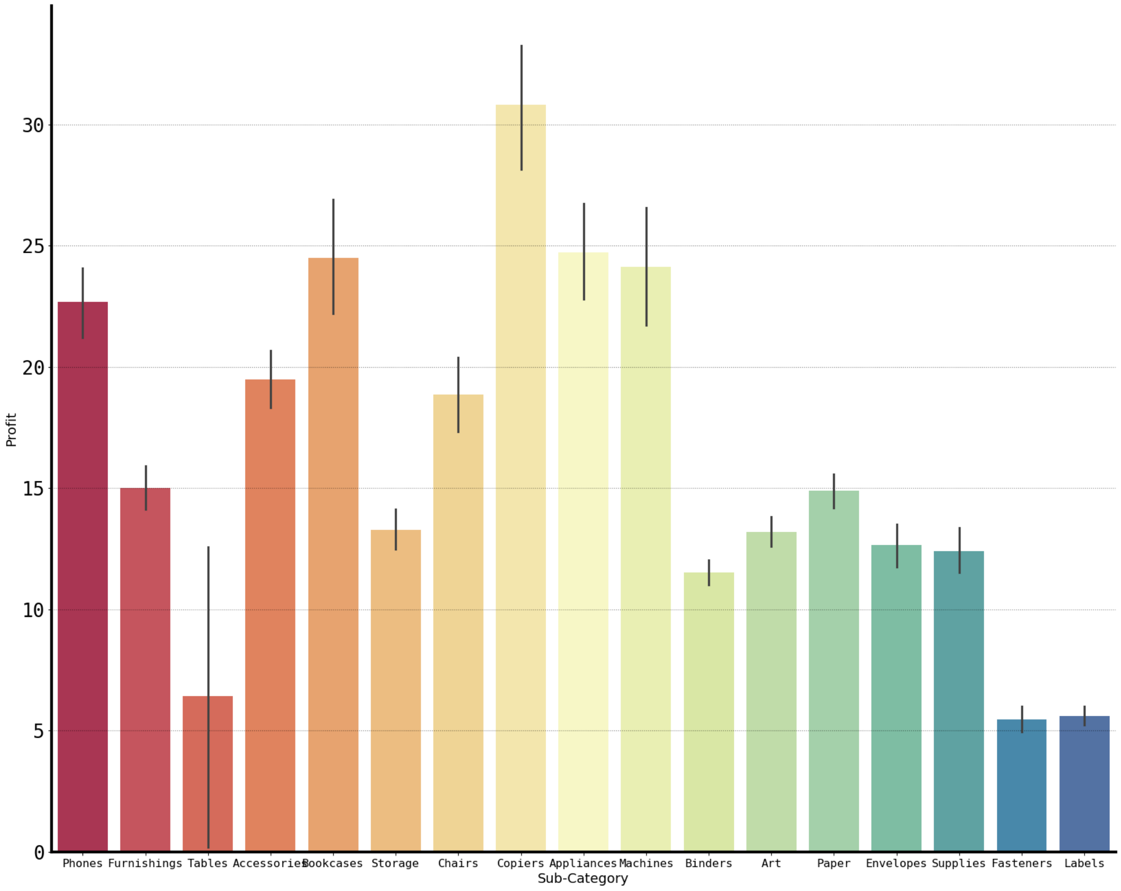
[343]:

```
# To creates bar plot of distribution of profit with different sub-categories
plt.figure(figsize = (20,16))

a = sns.barplot(x='Sub-Category', y='Profit', data = df,␣
 ↪palette='Spectral',linewidth=3)

plt.figtext(x=0.14, y=0.95,
            s='Distribution of Sub Categories based on Profit',
            fontsize=25, fontname='monospace')

plt.xticks(fontsize=12, fontname='monospace')
plt.yticks(fontsize=20, fontname='monospace')
plt.xlabel('Sub-Category', fontsize=14)
plt.ylabel('Profit', fontsize=14)

plt.grid(axis='y', color='black', linestyle = ':', alpha=0.5)

for q in [a]:
    for w in ['bottom', 'left']:
        q.spines[w].set_linewidth(3)
    for w in ['right', 'top']:
        q.spines[w].set_visible(False)
```

```
plt.show()
```

```
C:\Users\PARVATHY MENON\AppData\Local\Temp\ipykernel_11080\123768846.py:4:
FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same
effect.

  a = sns.barplot(x='Sub-Category', y='Profit', data = df,
palette='Spectral',linewidth=3)
```

Distribution of Sub Categories based on Profit



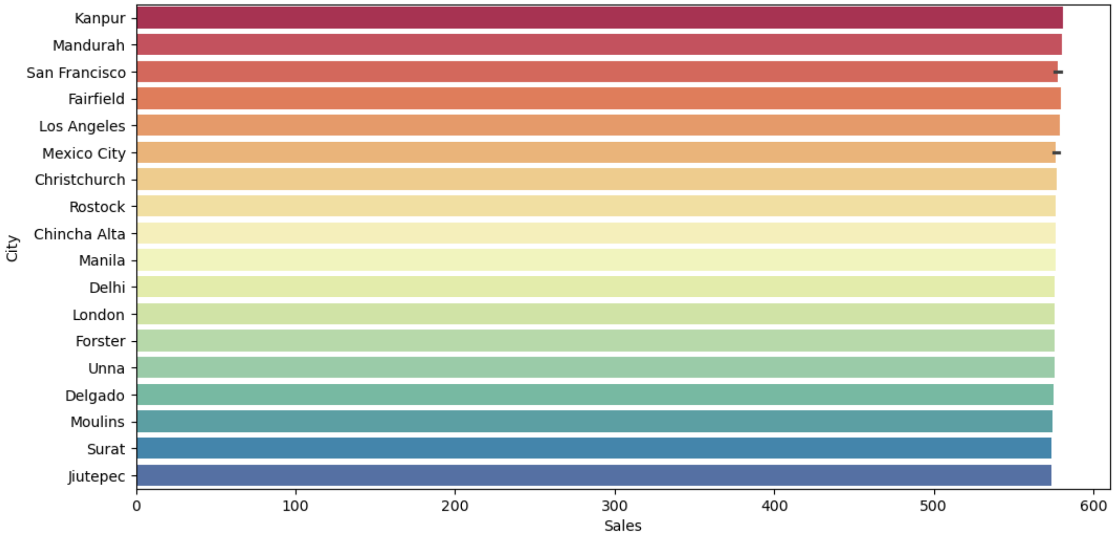
```
[344]: # To print cities with highest sales and sorting in descending order and␣
         ↪display top 20
       print("Cities with highest Sales are= ")
       print((df.sort_values("Sales",ascending=False).head(20))['City'])
```

```
Cities with highest Sales are=
8673            Kanpur
12659         Mandurah
8702     San Francisco
8316         Fairfield
6309       Los Angeles
9429       Mexico City
6275      Christchurch
7488           Rostock
7693      Chincha Alta
10133           Manila
16033    San Francisco
12421            Delhi
7619            London
7568           Forster
9393              Unna
6107           Delgado
9785       Mexico City
7827           Moulins
13037            Surat
8866          Jiutepec
Name: City, dtype: object
```

[345]:
```
# To create bar plot showing cities with highest sales
plt.subplots(figsize=(12,6))
sns.barplot(x="Sales", y="City" , data= df.sort_values("Sales",ascending=False).
  ↪head(20),palette='Spectral',linewidth=3)
```

```
C:\Users\PARVATHY MENON\AppData\Local\Temp\ipykernel_11080\3236778697.py:3:
FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same
effect.

  sns.barplot(x="Sales", y="City" , data=
df.sort_values("Sales",ascending=False).head(20),palette='Spectral',linewidth=3)
```

[345]:
```
<Axes: xlabel='Sales', ylabel='City'>
```



[346]:

```
# To print cities with lowest sales
print("Cities with lowest Sales are= ")
print((df.sort_values("Sales",ascending=True).head(20))['City'])
```

```
Cities with lowest Sales are=
50827    San Francisco
51110           Auburn
50744          Houston
51147            Tampa
51270          Seattle
51033          Phoenix
50437         San Jose
51080           Toledo
51057     Philadelphia
51143          Memphis
50355           Dallas
50791        Baltimore
49629             Troy
50186    New York City
50739      Brownsville
51067       Burlington
51061          Roswell
48897          Madison
50665    Santa Barbara
51085          Houston
Name: City, dtype: object
```

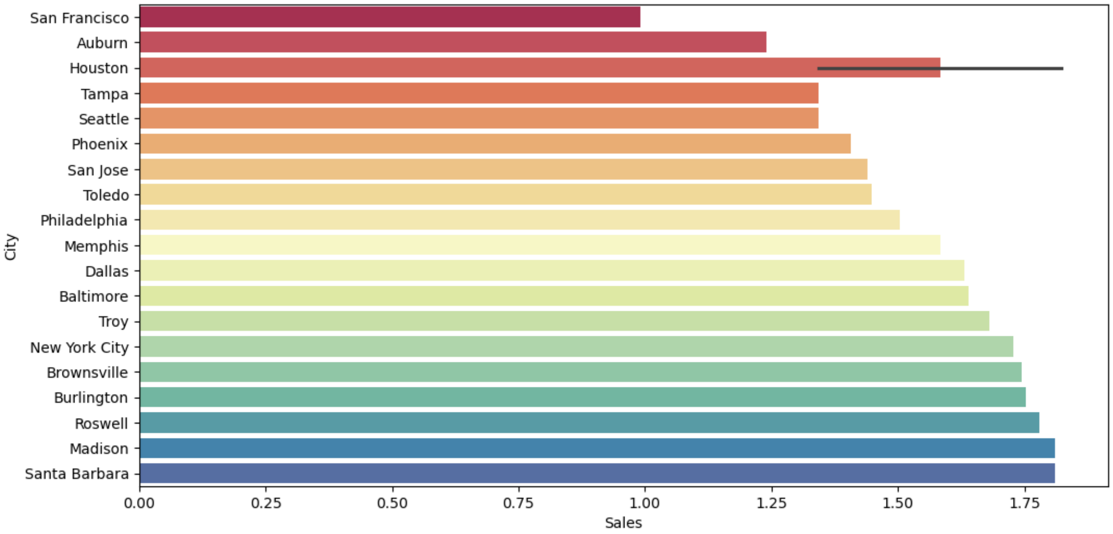
[352]:

```
#To create bar plot showing cities with lowest sales
plt.subplots(figsize=(12,6))
```

```
sns.barplot(x="Sales", y="City" , data= df.sort_values("Sales",ascending=True).
↪head(20),palette='Spectral',linewidth=3)
```

```
C:\Users\PARVATHY MENON\AppData\Local\Temp\ipykernel_11080\386077520.py:3:
FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same
effect.

  sns.barplot(x="Sales", y="City" , data=
df.sort_values("Sales",ascending=True).head(20),palette='Spectral',linewidth=3)
```

[352]:
```
<Axes: xlabel='Sales', ylabel='City'>
```



[357]:
```
# To print products generating highest profit
print("Products generating highest profit = ")
print((df.sort_values("Profit",ascending=False).head(20))['Product Name'])
```

```
Products generating highest profit =
9056     Razer Tiamat Over Ear 7.1 Surround Sound PC Ga…
7927     Computer Printout Paper with Letter-Trim Fine …
6520                                         Xerox 1911
15542                                        Xerox 1917
17531                    Tops White Computer Printout Paper
11435                                        Xerox 1917
7915                        Hewlett Fax and Copier, Digital
12973                       Hewlett Fax and Copier, Digital
6060                        Hewlett Fax and Copier, Digital
```

```
17136                          Enermax Mouse, Programmable
13368                          Enermax Mouse, Programmable
16962                                 Panasonic Phone, Red
9019                      LogitechÊGaming G510s - Keyboard
9923                      LogitechÊGaming G510s - Keyboard
6763                 Dana Halogen Swing-Arm Architect Lamp
20494                Dana Halogen Swing-Arm Architect Lamp
16987                      Logitech Flash Drive, Bluetooth
11837                   Hewlett Fax and Copier, High-Speed
22360                   Hewlett Fax and Copier, High-Speed
6551                                BIC Canvas, Easy-Erase
Name: Product Name, dtype: object
```

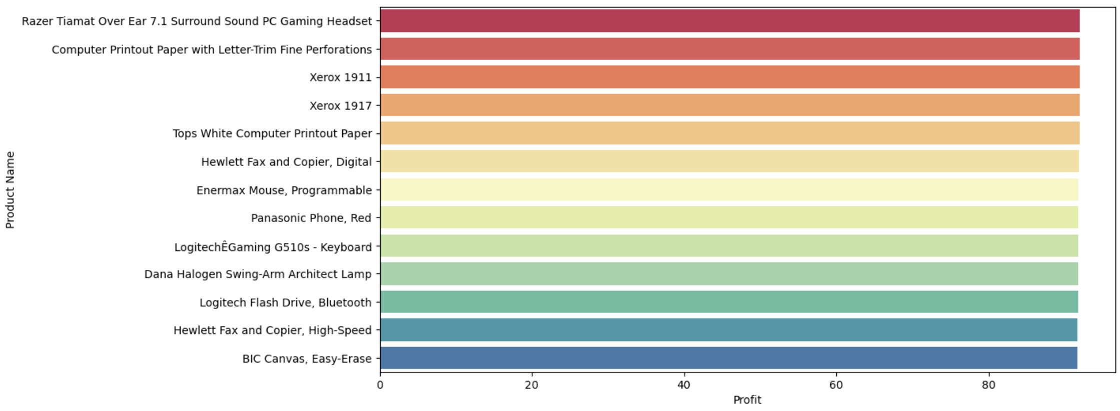
[358]:

```
plt.subplots(figsize=(12,6))
sns.barplot(x="Profit", y="Product Name" , data= df.
  ↪sort_values("Profit",ascending=False).
  ↪head(20),palette='Spectral',linewidth=3)
```

```
C:\Users\PARVATHY MENON\AppData\Local\Temp\ipykernel_11080\3533477376.py:2:
FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same
effect.

  sns.barplot(x="Profit", y="Product Name" , data= df.sort_values("Profit",ascen
ding=False).head(20),palette='Spectral',linewidth=3)
```

[358]:

```
<Axes: xlabel='Profit', ylabel='Product Name'>
```



[359]:

```
print("Products generating lowest profit = ")
print((df.sort_values("Profit",ascending=True).head(20))['Product Name'])
```

```
Products generating lowest profit =
20693             Stanley Pencil Sharpener, Water Color
15429                       Novimex Swivel Stool, Black
19904                       Novimex Swivel Stool, Black
35723                Stiletto Letter Opener, High Speed
13937                             Tenex Frame, Duo Pack
22751                             Tenex Frame, Duo Pack
24990                                Tenex Frame, Black
7985                   Fellowes File Cart, Single Width
20726                            Rogers File Cart, Blue
19258    Carina Media Storage Towers in Natural & Black
34494                                  Rogers Box, Blue
10360                     Logitech Router, Programmable
6685                            Nokia Headset, Cordless
20191               Hon Steel Folding Chair, Set of Two
35136                          Accos Paper Clips, Metal
7212                  High-Back Leather Manager's Chair
13644                 High-Back Leather Manager's Chair
12643             Safco Industrial Wire Shelving System
20067              Ikea Floating Shelf Set, Traditional
21263                        Xerox Memo Slips, 8.5 x 11
Name: Product Name, dtype: object
```

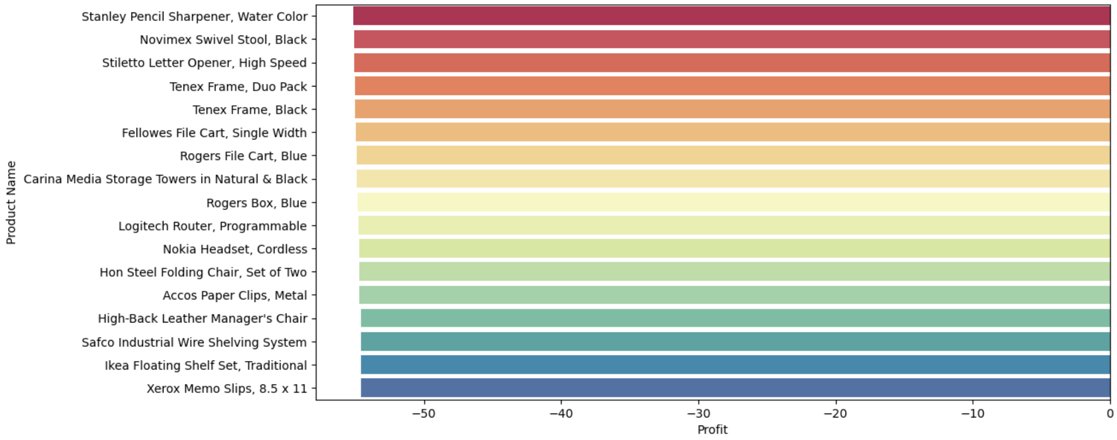
[360]:

```python
plt.subplots(figsize=(12,6))
sns.barplot(x="Profit", y="Product Name" , data= df.sort_values("Profit",ascending=True).head(20),palette='Spectral',linewidth=3)
```

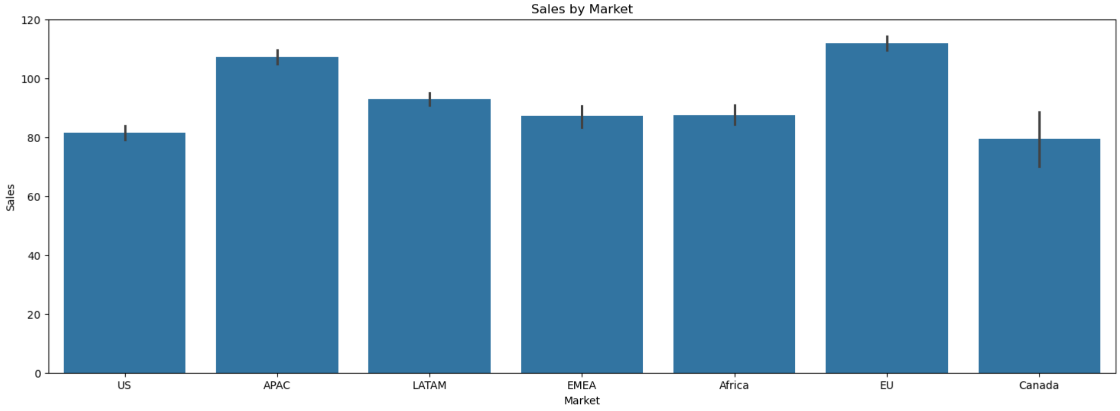
```
C:\Users\PARVATHY MENON\AppData\Local\Temp\ipykernel_11080\112663604.py:2:
FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same
effect.

  sns.barplot(x="Profit", y="Product Name" , data=
df.sort_values("Profit",ascending=True).head(20),palette='Spectral',linewidth=3)
```

[360]:

```
<Axes: xlabel='Profit', ylabel='Product Name'>
```

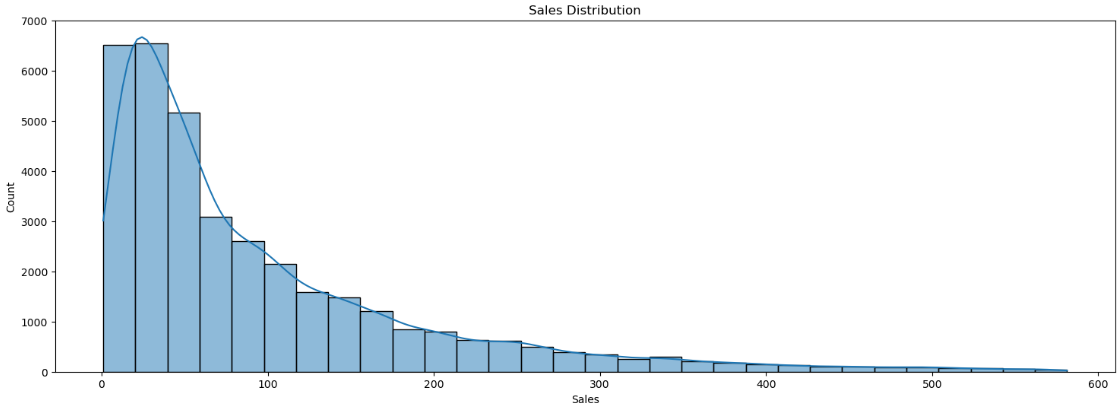


[361]:
```
# Bar Plot of sales by different markets
plt.figure(figsize=(18, 6))
sns.barplot(x='Market', y='Sales', data=df)
plt.title('Sales by Market')
plt.show()
```



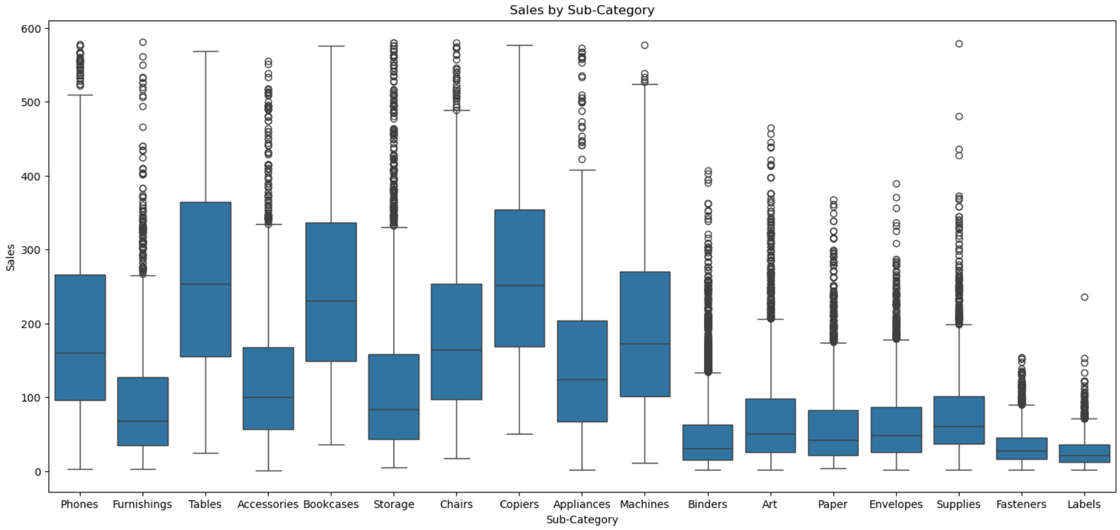
The company's performance is strongest in the EU and APAC market. There is potential for sales growth in the US and Canadian markets. The company has a relatively stable sales performance across the Africa, LATAM, EMEA.

[365]:
```
# Histogram of sales distribution
plt.figure(figsize=(18, 6))
sns.histplot(df['Sales'], bins=30, kde=True)
plt.title('Sales Distribution')
plt.show()
```



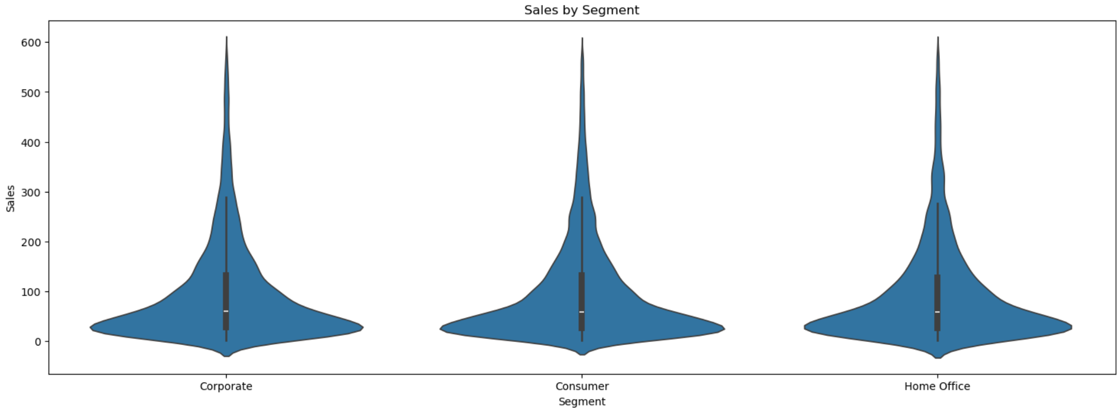
It can be observed that majority of the sales transactions are low and very few of them have high selling values. It means this right-skewed distribution shows that even though low amounts of sales are frequent, larger ones are not common. This type of distribution is very common for sales data, where small transactions prevail over large ones.

[367]:
```python
# Box Plot
plt.figure(figsize=(18, 8))
sns.boxplot(x='Sub-Category', y='Sales', data=df)
plt.title('Sales by Sub-Category')
plt.show()
```



Phones: The distribution of sale is quite high, with a median of about 200 and a lot of outliers going beyond the 500 mark. Furnishings: sales are lower, about 100, with a wide spread of data but many outliers. Tables: The sales are far more concentrated, have a medium close to 200, and contain many high outliers. Accessories, Bookcases, Storage, Chairs, Copiers, Appliances: With the exception of two or three categories, most of these categories yield a pattern much like the scatter plot above. They all have medians that vary between 100 to 200, are very volatile in the sale's values, and show an abundance of outliers. Binders, Art, Paper, Envelopes, Supplies, Fasteners, Labels: On average, the sales are in lower numbers at under 100. Also, the spread of the sales is narrower with many outliers.

[370]:
```
# Violin Plot for sales by segment
plt.figure(figsize=(18, 6))
sns.violinplot(x='Segment', y='Sales', data=df)
plt.title('Sales by Segment')
plt.show()
```



[371]:
```
df
```

[371]:
```
       Row ID         Order ID  Order Date   Ship Date       Ship Mode  \
5909    31907  CA-2013-161816   4/29/2013    5/2/2013     First Class
5914    24261   ID-2013-79432  12/11/2013  12/14/2013    Second Class
5915    33355  CA-2014-120376  12/23/2014  12/26/2014     First Class
5921    24245   IN-2014-22606   8/11/2014   8/11/2014        Same Day
5922    30952   ID-2014-85529    5/8/2014   5/12/2014  Standard Class
…       …             …            …          …         …
51284   24175   IN-2014-57662    8/5/2014   8/10/2014  Standard Class
51285   29002   IN-2014-62366   6/19/2014   6/19/2014        Same Day
51287   40470  US-2013-155768   12/2/2013   12/2/2013        Same Day
51288    9596  MX-2012-140767   2/18/2012   2/22/2012  Standard Class
51289    6147  MX-2012-134460   5/22/2012   5/26/2012    Second Class

      Customer ID        Customer Name     Segment       City  \
5909     NB-18655            Nona Balk   Corporate     Dallas
5914     JK-15325    Jason Klamczynski   Corporate   Adelaide
5915     TP-21130     Theone Pippenger    Consumer    Detroit
5921     RC-19960           Ryan Crowe    Consumer    Wenzhou
5922     AG-10300  Aleksandra Gannaway   Corporate   Auckland
```

```
…           …                  …          …         …
51284    DB-13270    Deborah Brumfield  Home Office  Townsville
51285    KE-16420      Katrina Edelman    Corporate        Kure
51287    LB-16795       Laurel Beltran  Home Office      Oxnard
51288    RB-19795           Ross Baird  Home Office    Valinhos
51289    MC-18100        Mick Crebagga     Consumer    Tipitapa

                 State  …      Product ID         Category Sub-Category  \
5909             Texas  …  TEC-PH-10003012       Technology       Phones
5914   South Australia  …  FUR-FU-10001477        Furniture  Furnishings
5915          Michigan  …  FUR-TA-10004534        Furniture       Tables
5921          Zhejiang  …  TEC-AC-10000861       Technology  Accessories
5922          Auckland  …  FUR-BO-10001872        Furniture    Bookcases
…                …  …              …              …          …
51284       Queensland  …  OFF-BI-10002424  Office Supplies      Binders
51285        Hiroshima  …  OFF-FA-10000746  Office Supplies    Fasteners
51287       California  …  OFF-EN-10001219  Office Supplies    Envelopes
51288        S<o Paulo  …  OFF-BI-10000806  Office Supplies      Binders
51289          Managua  …  OFF-PA-10004155  Office Supplies        Paper

                                           Product Name    Sales Quantity  \
5909   Nortel Meridian M3904 Professional Digital phone  369.576      3.0
5914                    Rubbermaid Stacking Tray, Black  158.760      7.0
5915                    Bevis 44 x 96 Conference Tables  411.800      2.0
5921                          SanDisk Router, Erganomic  254.160      1.0
5922                      Bush Stackable Bookrack, Pine  449.496      6.0
…                                                 …      …      …
51284                             Avery Binder, Economy   58.050      5.0
51285                     Advantus Thumb Tacks, 12 Pack   65.100      5.0
51287      #10- 4 1/8" x 9 1/2" Security-Tint Envelopes   22.920      3.0
51288                           Acco Index Tab, Economy   13.440      2.0
51289          Eaton Computer Printout Paper, 8.5 x 11   61.380      3.0

      Discount   Profit  Shipping Cost  Order Priority
5909       0.2  41.5773          57.20          Medium
5914       0.1  21.0000          57.12        Critical
5915       0.0  70.0060          57.12            High
5921       0.0  40.6500          57.07          Medium
5922       0.4 -30.0240          57.05            High
…        …      …            …             …
51284      0.1  19.9500           0.01          Medium
51285      0.0   4.5000           0.01          Medium
51287      0.0  11.2308           0.01            High
51288      0.0   2.4000           0.00          Medium
51289      0.0   1.8000           0.00            High

[36311 rows x 23 columns]
```

[372]:

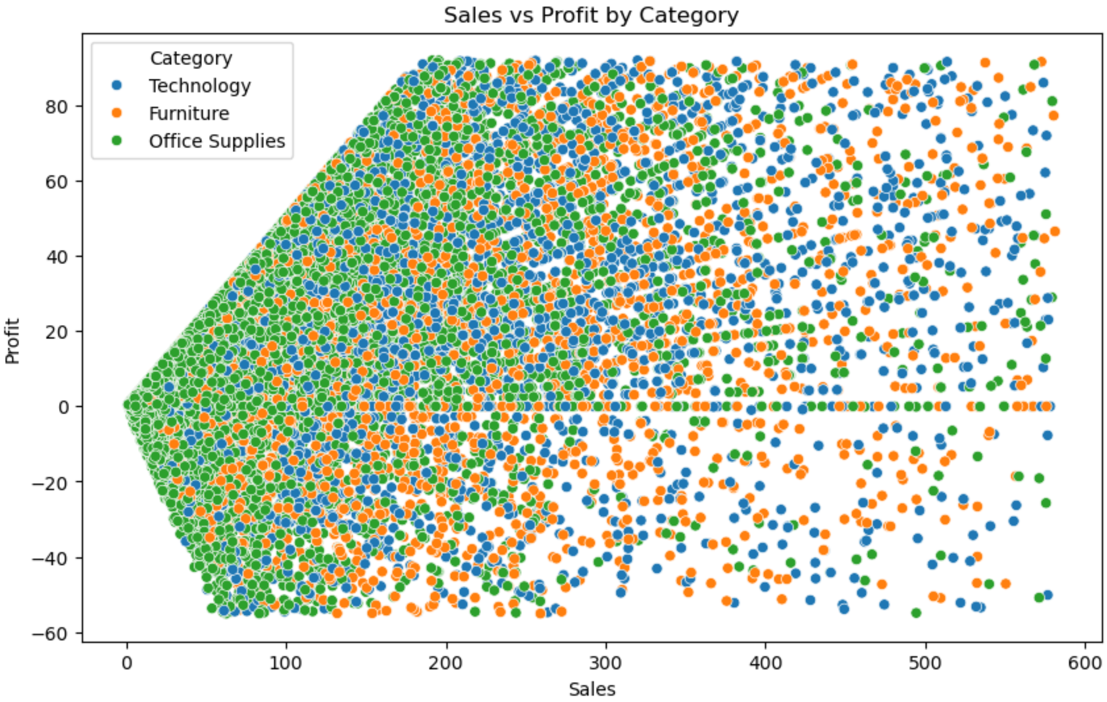
```python
# Check for null values
df.isnull().sum()
```

[372]:

```
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
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
```

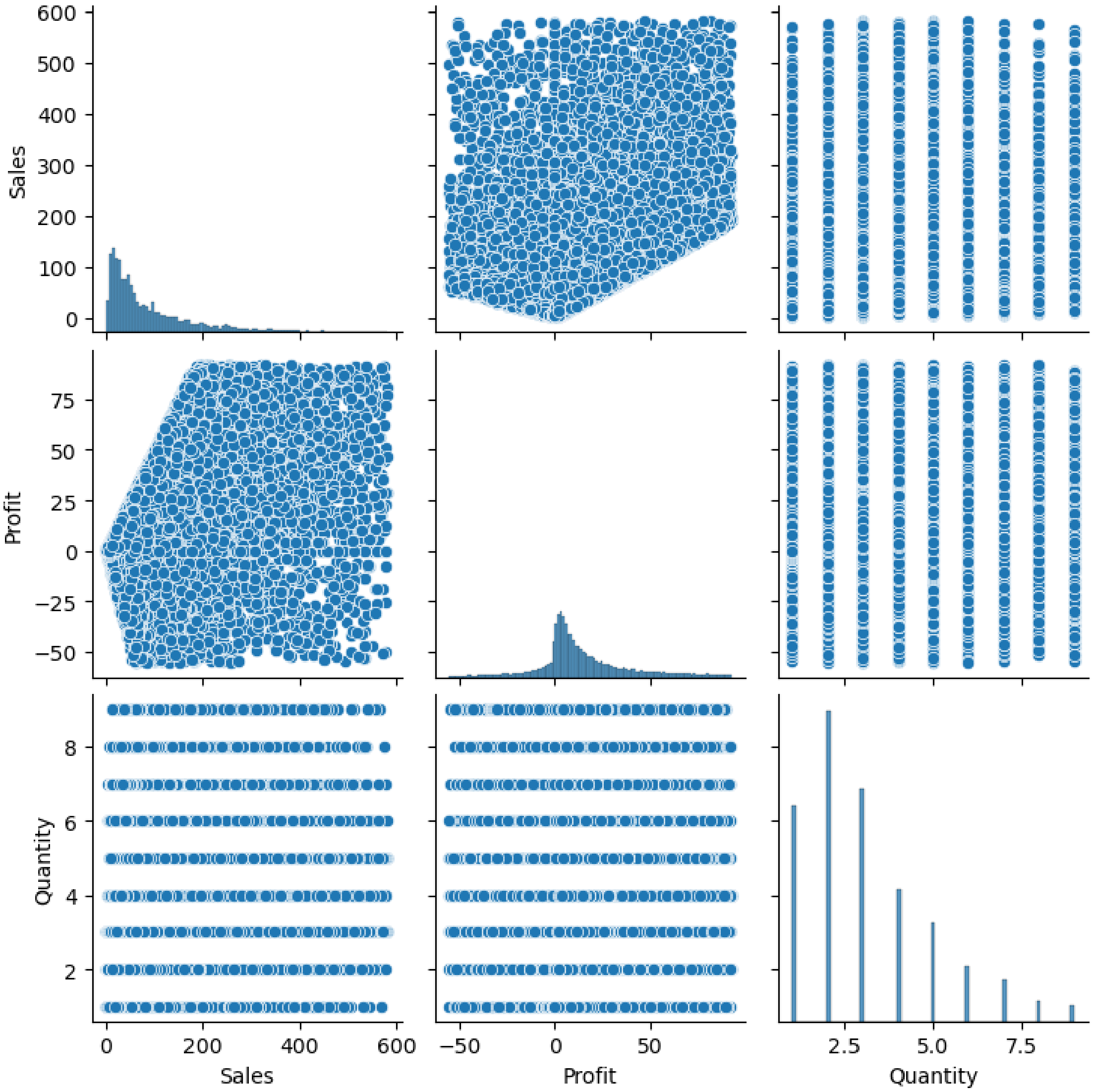
[373]:

```python
# Scatter Plot of sales vs profit by category
plt.figure(figsize=(10, 6))
sns.scatterplot(x='Sales', y='Profit', data=df, hue='Category')
plt.title('Sales vs Profit by Category')
plt.show()
```

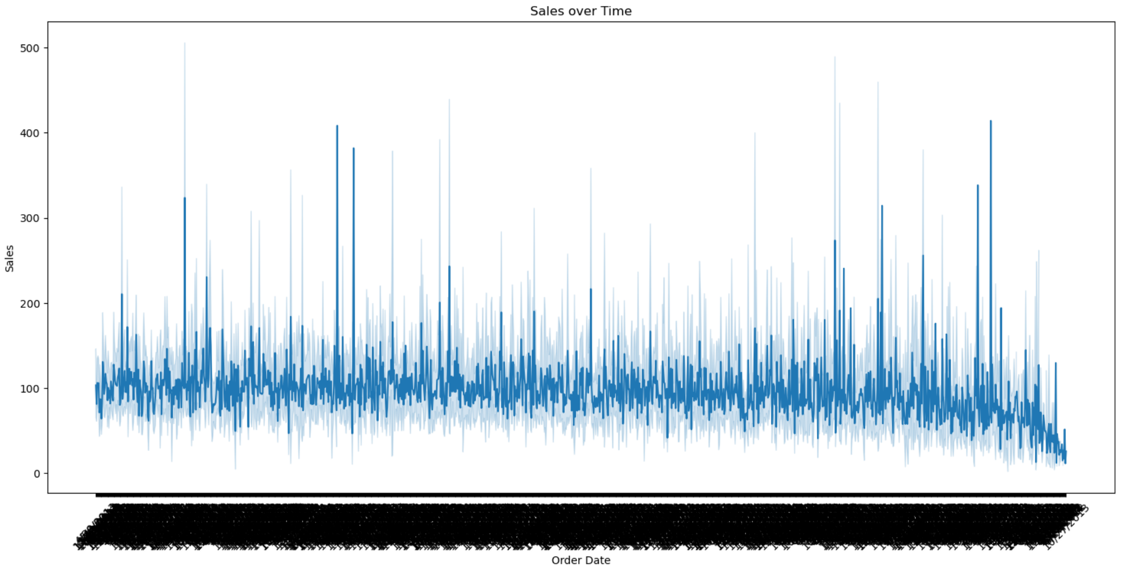


Technology(blue): The sales and profit both are very spread out with some of those being high profit and negative profit. Furnitur(orange)s:Ssimilar to Technology, but seems to havh more significant presence in the negative profit area, particularly between -20 to -40. Office Supplies (Green Dots)H have the largest spread along the sales axis but also a wide dispersion in profit, including a large part of it in the negative range. The graph sh that is a lot of variation in profit over all categories at all levels of sales, and no consistent trend that higher sales result in higher profit. A lot of negative profits are shown, mainly in the Furniture and Office Supplies categories.

[375]:
```
# Pair Plot
sns.pairplot(df[['Sales', 'Profit', 'Market', 'Quantity']])
plt.show()
```

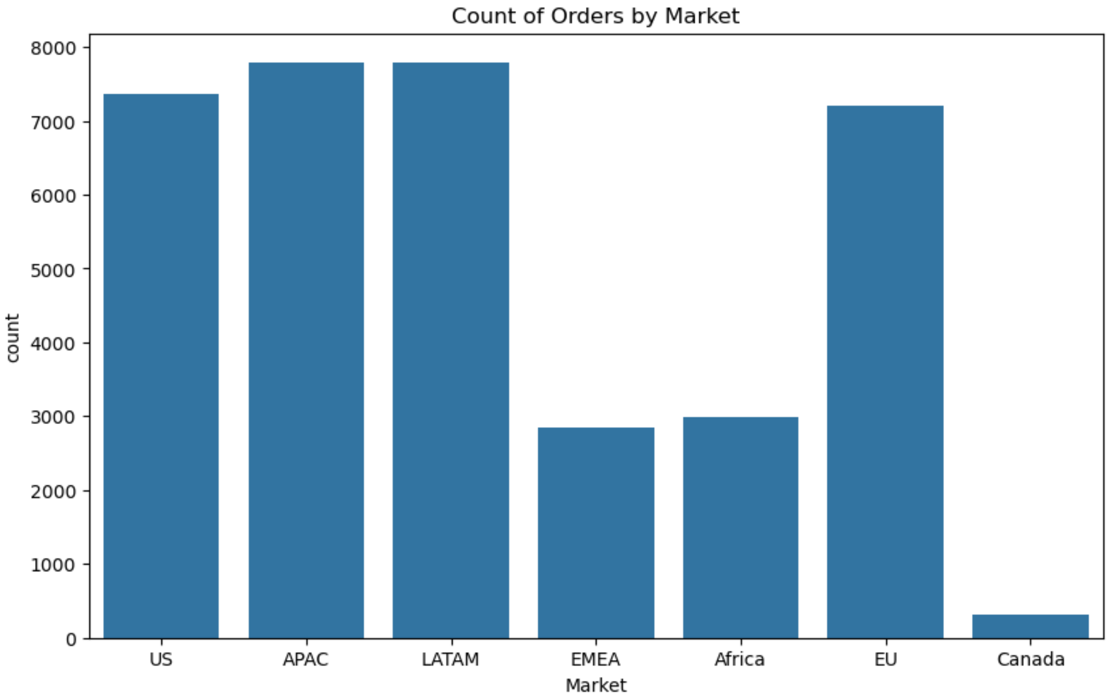


[376]:
```
# Line Plot of sales over time
plt.figure(figsize=(18, 8))
sns.lineplot(x='Order Date', y='Sales', data=df)
plt.title('Sales over Time')
plt.xticks(rotation=45)
plt.show()
```



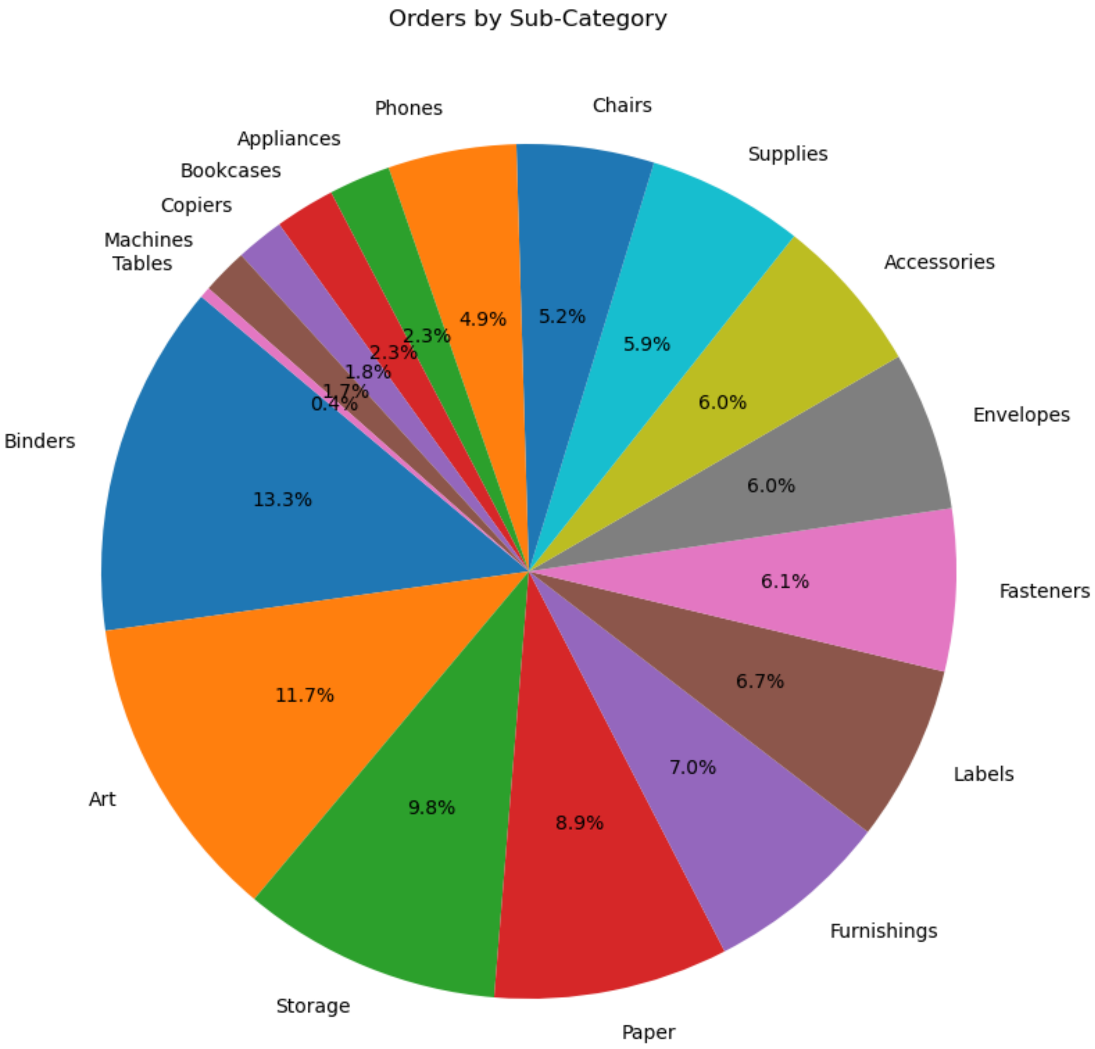
[377]:

```
# Count Plot of orders by market
plt.figure(figsize=(10, 6))
sns.countplot(x='Market', data=df)
plt.title('Count of Orders by Market')
plt.show()
```



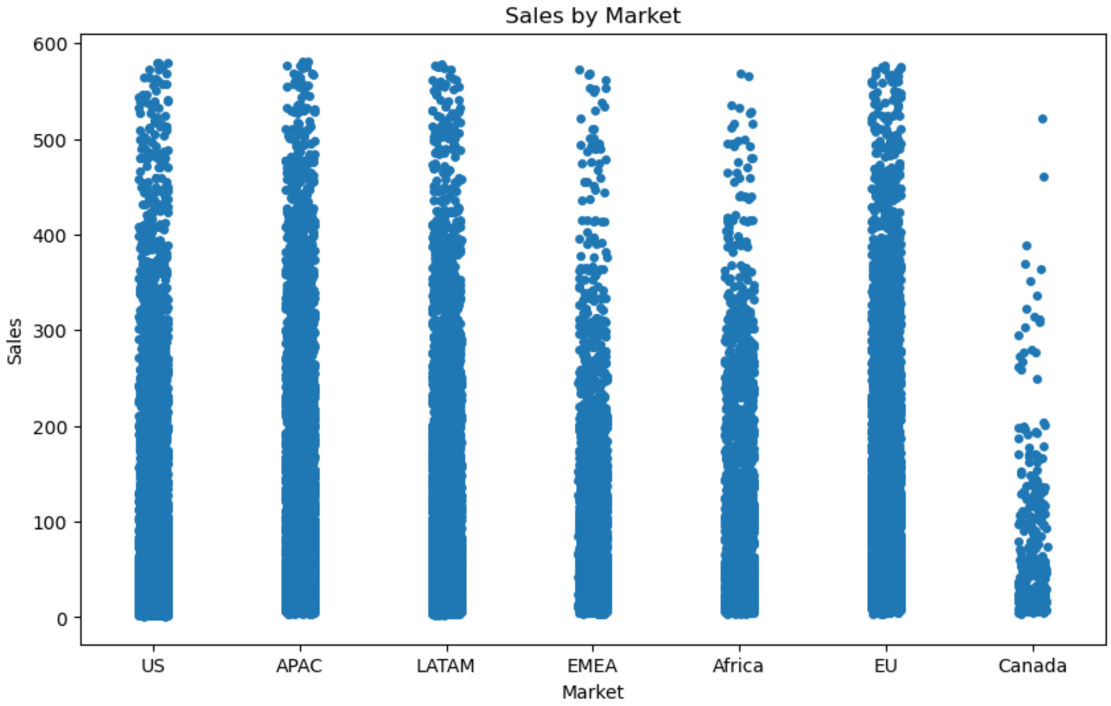
[378]:

```
# Pie Chart of orders by sub category
category_counts = df['Sub-Category'].value_counts()
plt.figure(figsize=(10, 10))
plt.pie(category_counts, labels=category_counts.index, autopct='%1.1f%%',␣
 ↪startangle=140)
plt.title('Orders by Sub-Category')
plt.show()
```



[380]:

```
# Strip Plot of sales over time
plt.figure(figsize=(10, 6))
sns.stripplot(x='Market', y='Sales', data=df, jitter=True)
plt.title('Sales by Market')
plt.show()
```



[ ]:

```
# Time Series Plot of Sales (if 'Order Date' column exists and is in datetime format)
if 'Order Date' in df.columns:
    df['Order Date'] = pd.to_datetime(df['Order Date'], errors='coerce')
    plt.figure(figsize=(10, 6))
    sns.lineplot(x='Order Date', y='Sales', data=df)
    plt.title('Sales over Time')
    plt.xticks(rotation=45)
    plt.show()
```

[ ]:

[ ]:

[ ]:

[ ]:

[ ]: