untitled2

January 23, 2024

[59]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
### Clear Warning
import warnings;
warnings.simplefilter('ignore')
```

[2]:
```
df=pd.read_csv('C:\\Users\\avili\\Downloads\\train.csv')
```

[3]:
```
df
```

[3]:

	Row ID	Order ID	Order Date	Ship Date	Ship Mode	\
0	1	CA-2017-152156	08/11/2017	11/11/2017	Second Class	
1	2	CA-2017-152156	08/11/2017	11/11/2017	Second Class	
2	3	CA-2017-138688	12/06/2017	16/06/2017	Second Class	
3	4	US-2016-108966	11/10/2016	18/10/2016	Standard Class	
4	5	US-2016-108966	11/10/2016	18/10/2016	Standard Class	
…	…	…	…	…	…	
9795	9796	CA-2017-125920	21/05/2017	28/05/2017	Standard Class	
9796	9797	CA-2016-128608	12/01/2016	17/01/2016	Standard Class	
9797	9798	CA-2016-128608	12/01/2016	17/01/2016	Standard Class	
9798	9799	CA-2016-128608	12/01/2016	17/01/2016	Standard Class	
9799	9800	CA-2016-128608	12/01/2016	17/01/2016	Standard Class	

	Customer ID	Customer Name	Segment	Country	City	\
0	CG-12520	Claire Gute	Consumer	United States	Henderson	
1	CG-12520	Claire Gute	Consumer	United States	Henderson	
2	DV-13045	Darrin Van Huff	Corporate	United States	Los Angeles	
3	SO-20335	Sean O'Donnell	Consumer	United States	Fort Lauderdale	
4	SO-20335	Sean O'Donnell	Consumer	United States	Fort Lauderdale	
…	…	…	…	…	…	
9795	SH-19975	Sally Hughsby	Corporate	United States	Chicago	
9796	CS-12490	Cindy Schnelling	Corporate	United States	Toledo	
9797	CS-12490	Cindy Schnelling	Corporate	United States	Toledo	
9798	CS-12490	Cindy Schnelling	Corporate	United States	Toledo	

```
9799    CS-12490  Cindy Schnelling  Corporate  United States           Toledo

           State  Postal Code   Region       Product ID         Category  \
0       Kentucky      42420.0    South  FUR-BO-10001798        Furniture
1       Kentucky      42420.0    South  FUR-CH-10000454        Furniture
2     California      90036.0     West  OFF-LA-10000240  Office Supplies
3        Florida      33311.0    South  FUR-TA-10000577        Furniture
4        Florida      33311.0    South  OFF-ST-10000760  Office Supplies
…          …          …      …              …              …
9795    Illinois      60610.0  Central  OFF-BI-10003429  Office Supplies
9796        Ohio      43615.0     East  OFF-AR-10001374  Office Supplies
9797        Ohio      43615.0     East  TEC-PH-10004977       Technology
9798        Ohio      43615.0     East  TEC-PH-10000912       Technology
9799        Ohio      43615.0     East  TEC-AC-10000487       Technology

     Sub-Category                                         Product Name     Sales
0       Bookcases                    Bush Somerset Collection Bookcase  261.9600
1          Chairs  Hon Deluxe Fabric Upholstered Stacking Chairs,…  731.9400
2          Labels  Self-Adhesive Address Labels for Typewriters b…   14.6200
3          Tables      Bretford CR4500 Series Slim Rectangular Table  957.5775
4         Storage                       Eldon Fold 'N Roll Cart System   22.3680
…           …                                                   …       …
9795      Binders  Cardinal HOLDit! Binder Insert Strips,Extra St…    3.7980
9796          Art            BIC Brite Liner Highlighters, Chisel Tip   10.3680
9797       Phones                                         GE 30524EE4  235.1880
9798       Phones            Anker 24W Portable Micro USB Car Charger   26.3760
9799  Accessories                 SanDisk Cruzer 4 GB USB Flash Drive   10.3840

[9800 rows x 18 columns]
```

[4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9800 entries, 0 to 9799
Data columns (total 18 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Row ID         9800 non-null   int64
 1   Order ID       9800 non-null   object
 2   Order Date     9800 non-null   object
 3   Ship Date      9800 non-null   object
 4   Ship Mode      9800 non-null   object
 5   Customer ID    9800 non-null   object
 6   Customer Name  9800 non-null   object
 7   Segment        9800 non-null   object
 8   Country        9800 non-null   object
 9   City           9800 non-null   object
```

```
 10  State          9800 non-null   object
 11  Postal Code    9789 non-null   float64
 12  Region         9800 non-null   object
 13  Product ID     9800 non-null   object
 14  Category       9800 non-null   object
 15  Sub-Category   9800 non-null   object
 16  Product Name   9800 non-null   object
 17  Sales          9800 non-null   float64
dtypes: float64(2), int64(1), object(15)
memory usage: 1.3+ MB
```

[5]:
```
df.shape
```

[5]:
```
(9800, 18)
```

[ ]:
```

```

[6]:
```
df.head()
```

[6]:

	Row ID	Order ID	Order Date	Ship Date	Ship Mode	Customer ID	Customer Name	Segment	Country	City	State	Postal Code	Region	Product ID	Category	Sub-Category	Product Name	Sales
0	1	CA-2017-152156	08/11/2017	11/11/2017	Second Class	CG-12520	Claire Gute	Consumer	United States	Henderson	Kentucky	42420.0	South	FUR-BO-10001798	Furniture	Bookcases	Bush Somerset Collection Bookcase	261.9600
1	2	CA-2017-152156	08/11/2017	11/11/2017	Second Class	CG-12520	Claire Gute	Consumer	United States	Henderson	Kentucky	42420.0	South	FUR-CH-10000454	Furniture	Chairs	Hon Deluxe Fabric Upholstered Stacking Chairs,…	731.9400
2	3	CA-2017-138688	12/06/2017	16/06/2017	Second Class	DV-13045	Darrin Van Huff	Corporate	United States	Los Angeles	California	90036.0	West	OFF-LA-10000240	Office Supplies	Labels	Self-Adhesive Address Labels for Typewriters b…	14.6200
3	4	US-2016-108966	11/10/2016	18/10/2016	Standard Class	SO-20335	Sean O'Donnell	Consumer	United States	Fort Lauderdale	Florida	33311.0	South	FUR-TA-10000577	Furniture	Tables	Bretford CR4500 Series Slim Rectangular Table	957.5775
4	5	US-2016-108966	11/10/2016	18/10/2016	Standard Class	SO-20335	Sean O'Donnell	Consumer	United States	Fort Lauderdale	Florida	33311.0	South	OFF-ST-10000760	Office Supplies	Storage	Eldon Fold 'N Roll Cart System	22.3680

[7]:
```
df.tail()
```

[7]:
```
      Row ID        Order ID  Order Date   Ship Date       Ship Mode  \
9795    9796  CA-2017-125920  21/05/2017  28/05/2017  Standard Class
9796    9797  CA-2016-128608  12/01/2016  17/01/2016  Standard Class
9797    9798  CA-2016-128608  12/01/2016  17/01/2016  Standard Class
9798    9799  CA-2016-128608  12/01/2016  17/01/2016  Standard Class
9799    9800  CA-2016-128608  12/01/2016  17/01/2016  Standard Class

     Customer ID     Customer Name    Segment        Country     City  \
9795    SH-19975     Sally Hughsby  Corporate  United States  Chicago
9796    CS-12490  Cindy Schnelling  Corporate  United States   Toledo
9797    CS-12490  Cindy Schnelling  Corporate  United States   Toledo
9798    CS-12490  Cindy Schnelling  Corporate  United States   Toledo
9799    CS-12490  Cindy Schnelling  Corporate  United States   Toledo

         State  Postal Code   Region       Product ID         Category  \
9795  Illinois      60610.0  Central  OFF-BI-10003429  Office Supplies
9796      Ohio      43615.0     East  OFF-AR-10001374  Office Supplies
9797      Ohio      43615.0     East  TEC-PH-10004977       Technology
9798      Ohio      43615.0     East  TEC-PH-10000912       Technology
9799      Ohio      43615.0     East  TEC-AC-10000487       Technology

     Sub-Category                                        Product Name    Sales
9795      Binders  Cardinal HOLDit! Binder Insert Strips,Extra St…    3.798
9796          Art           BIC Brite Liner Highlighters, Chisel Tip   10.368
9797       Phones                                     GE 30524EE4  235.188
9798       Phones           Anker 24W Portable Micro USB Car Charger   26.376
9799  Accessories                SanDisk Cruzer 4 GB USB Flash Drive   10.384
```

[8]:
```
df.isnull().sum()
```

[8]:
```
Row ID           0
Order ID         0
Order Date       0
Ship Date        0
Ship Mode        0
Customer ID      0
Customer Name    0
Segment          0
Country          0
City             0
State            0
Postal Code     11
Region           0
Product ID       0
Category         0
```

```
Sub-Category      0
Product Name      0
Sales             0
dtype: int64
```

[9]:
```
df.dtypes
```

[9]:
```
Row ID             int64
Order ID          object
Order Date        object
Ship Date         object
Ship Mode         object
Customer ID       object
Customer Name     object
Segment           object
Country           object
City              object
State             object
Postal Code      float64
Region            object
Product ID        object
Category          object
Sub-Category      object
Product Name      object
Sales            float64
dtype: object
```

[10]:
```
df.dropna(inplace=True)
```

[11]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 9789 entries, 0 to 9799
Data columns (total 18 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Row ID         9789 non-null   int64
 1   Order ID       9789 non-null   object
 2   Order Date     9789 non-null   object
 3   Ship Date      9789 non-null   object
 4   Ship Mode      9789 non-null   object
 5   Customer ID    9789 non-null   object
 6   Customer Name  9789 non-null   object
 7   Segment        9789 non-null   object
 8   Country        9789 non-null   object
 9   City           9789 non-null   object
 10  State          9789 non-null   object
```

```
 11  Postal Code    9789 non-null   float64
 12  Region         9789 non-null   object
 13  Product ID     9789 non-null   object
 14  Category       9789 non-null   object
 15  Sub-Category   9789 non-null   object
 16  Product Name   9789 non-null   object
 17  Sales          9789 non-null   float64
dtypes: float64(2), int64(1), object(15)
memory usage: 1.4+ MB
```

[12]:
```
df.describe()
```

[12]:

	Row ID	Postal Code	Sales
count	9789.000000	9789.000000	9789.000000
mean	4896.705588	55273.322403	230.116193
std	2827.486899	32041.223413	625.302079
min	1.000000	1040.000000	0.444000
25%	2449.000000	23223.000000	17.248000
50%	4896.000000	58103.000000	54.384000
75%	7344.000000	90008.000000	210.392000
max	9800.000000	99301.000000	22638.480000

[13]:
```
#indentify the duplicated values.
duplicate_rows = df.duplicated()
print(duplicate_rows)
```

```
0       False
1       False
2       False
3       False
4       False
        …
9795    False
9796    False
9797    False
9798    False
9799    False
Length: 9789, dtype: bool
```

1 Performing EDA(Exploratory Data Analysis)

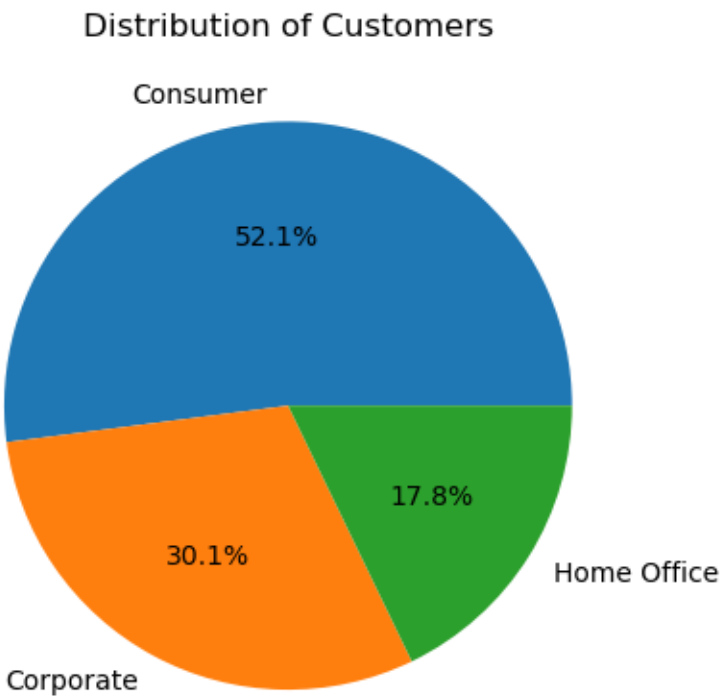
[14]:
```
types_of_customer = df['Segment'].unique()
print(types_of_customer)
```

```
['Consumer' 'Corporate' 'Home Office']
```

[15]:
```
number_of_customers = df['Segment'].value_counts().reset_index()
#number_of_customers= number_of_customers.rename(columns={'Segment':'Customer
 ↪Type','count':'Total Customers'})

print(number_of_customers)
```

```
       Segment  count
0     Consumer   5096
1    Corporate   2948
2  Home Office   1745
```

[16]:
```
plt.
 ↪pie(number_of_customers['count'],labels=number_of_customers['Segment'],autopct='%1.
 ↪1f%%')

plt.title('Distribution of Customers')

plt.show()
```

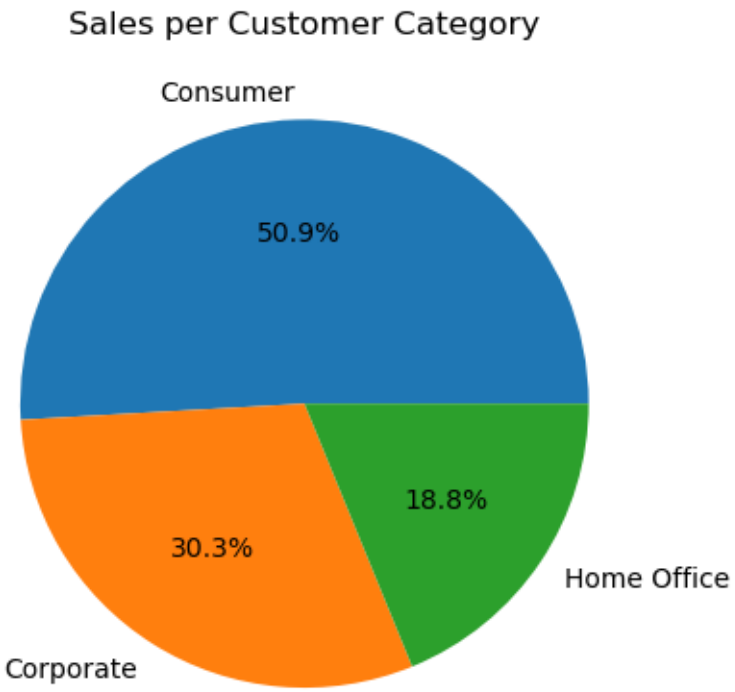


[17]:
```
sales_per_category = df.groupby('Segment')['Sales'].sum().reset_index()
```

```
sales_per_category = sales_per_category.rename(columns={'Segment':'Customer␣
↪type','Sales':'Total Sales'})
print(sales_per_category)
```

```
  Customer type   Total Sales
0      Consumer  1.146708e+06
1     Corporate  6.822118e+05
2   Home Office  4.236874e+05
```

[18]:
```
plt.pie(sales_per_category['Total Sales'],labels=sales_per_category['Customer␣
↪type'],autopct='%1.1f%%')

plt.title('Sales per Customer Category')

plt.show()
```



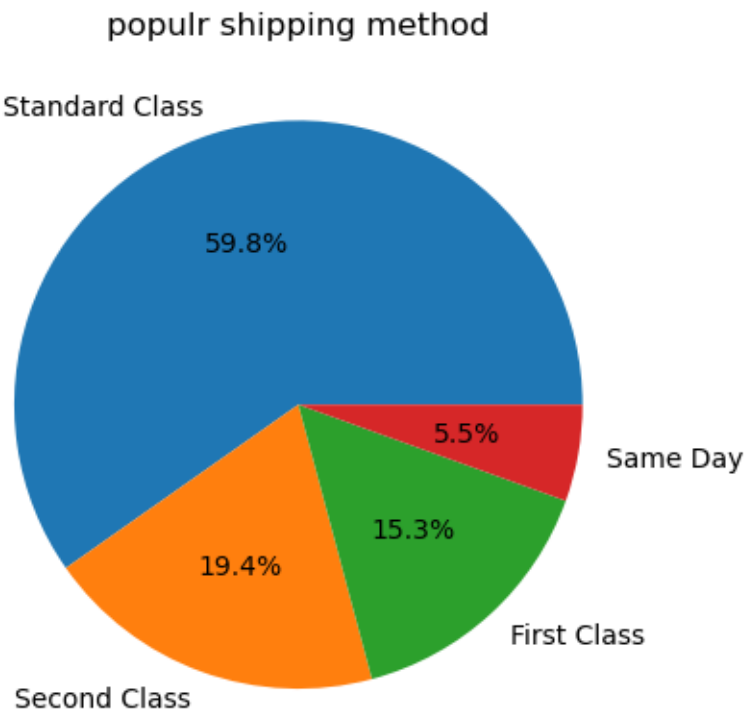
[19]:
```
#Analysis Of Shipping mode

types_of_shipping=df['Ship Mode'].unique()
print(types_of_shipping)
```

```
['Second Class' 'Standard Class' 'First Class' 'Same Day']
```

[20]:
```
#frequency use of shipping mode
shipp_mode = df['Ship Mode'].value_counts().reset_index()
shipp_mode = shipp_mode.rename(columns={'Ship Mode':'Mode of Shipping'})
print(shipp_mode)
```

```
  Mode of Shipping  count
0   Standard Class   5849
1     Second Class   1901
2      First Class   1501
3         Same Day    538
```

[21]:
```
plt.pie(shipp_mode['count'],labels = shipp_mode['Mode of Shipping'],␣
↪autopct='%1.1f%%')

plt.title('populr shipping method')

plt.show()
```



2 Graphical Analysis

[22]:
```
#Customer by State
state = df['State'].value_counts().reset_index()
print(state)
```

```
                State  count
0          California   1946
1            New York   1097
2               Texas    973
3        Pennsylvania    582
4          Washington    504
5            Illinois    483
6                Ohio    454
7             Florida    373
8            Michigan    253
9      North Carolina    247
10           Virginia    224
11            Arizona    223
12          Tennessee    183
13           Colorado    179
14            Georgia    177
15           Kentucky    137
16            Indiana    135
17      Massachusetts    135
18             Oregon    122
19         New Jersey    122
20           Maryland    105
21          Wisconsin    105
22           Delaware     93
23          Minnesota     89
24        Connecticut     82
25           Missouri     66
26           Oklahoma     66
27            Alabama     61
28           Arkansas     60
29       Rhode Island     55
30        Mississippi     53
31               Utah     53
32     South Carolina     42
33          Louisiana     41
34             Nevada     39
35           Nebraska     38
36         New Mexico     37
37      New Hampshire     27
38               Iowa     26
39             Kansas     24
```

```
40                 Idaho     21
41               Montana     15
42          South Dakota     12
43  District of Columbia     10
44                 Maine      8
45          North Dakota      7
46         West Virginia      4
47               Wyoming      1
```

[23]:
```
#customer by City
City = df['City'].value_counts().reset_index()
print(City)
```

```
              City  count
0    New York City    891
1      Los Angeles    728
2     Philadelphia    532
3    San Francisco    500
4          Seattle    426
..             …    …
524      San Mateo      1
525       Cheyenne      1
526         Conway      1
527      Melbourne      1
528     Springdale      1

[529 rows x 2 columns]
```

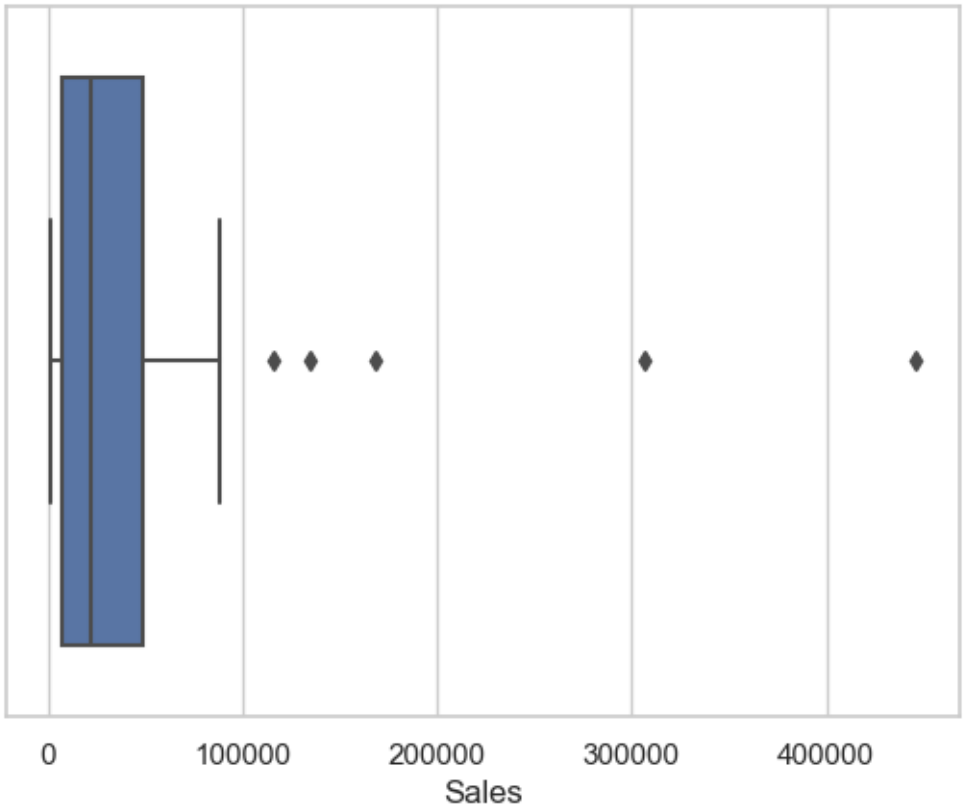
[24]:
```
#sales per state
sales_per_state = df.groupby('State')['Sales'].sum().reset_index()

print(sales_per_state)
```

```
                  State        Sales
0               Alabama   19510.6400
1               Arizona   35272.6570
2              Arkansas   11678.1300
3            California  446306.4635
4              Colorado   31841.5980
5           Connecticut   13384.3570
6              Delaware   27322.9990
7   District of Columbia    2865.0200
8               Florida   88436.5320
9               Georgia   48219.1100
10                Idaho    4382.4860
11             Illinois   79236.5170
12              Indiana   48718.4000
13                 Iowa    4443.5600
```

14	Kansas	2914.3100
15	Kentucky	36458.3900
16	Louisiana	9131.0500
17	Maine	1270.5300
18	Maryland	23705.5230
19	Massachusetts	28634.4340
20	Michigan	76136.0740
21	Minnesota	29863.1500
22	Mississippi	10771.3400
23	Missouri	22205.1500
24	Montana	5589.3520
25	Nebraska	7464.9300
26	Nevada	16729.1020
27	New Hampshire	7292.5240
28	New Jersey	34610.9720
29	New Mexico	4783.5220
30	New York	306361.1470
31	North Carolina	55165.9640
32	North Dakota	919.9100
33	Ohio	75130.3500
34	Oklahoma	19683.3900
35	Oregon	17284.4620
36	Pennsylvania	116276.6500
37	Rhode Island	22525.0260
38	South Carolina	8481.7100
39	South Dakota	1315.5600
40	Tennessee	30661.8730
41	Texas	168572.5322
42	Utah	11220.0560
43	Virginia	70636.7200
44	Washington	135206.8500
45	West Virginia	1209.8240
46	Wisconsin	31173.4300
47	Wyoming	1603.1360

[61]:
```
#creating boxplot
sns.boxplot(x=sales_per_state["Sales"])
```

[61]: <Axes: xlabel='Sales'>

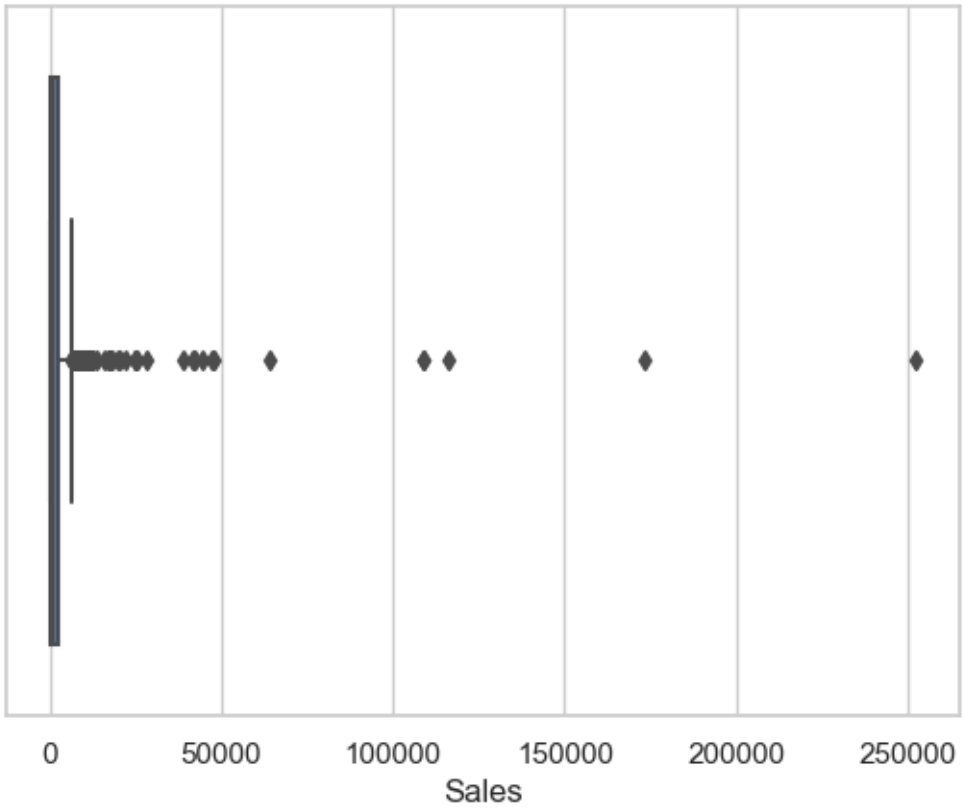


[26]:
```
#sales per City
sales_per_City = df.groupby('City')['Sales'].sum().reset_index()

print(sales_per_City)
```

```
            City     Sales
0       Aberdeen    25.500
1        Abilene     1.392
2          Akron  2724.244
3    Albuquerque  2220.160
4     Alexandria  5519.570
..           …       …
524   Woonsocket   195.550
525      Yonkers  7657.666
526         York   817.978
527      Yucaipa    50.800
528         Yuma   840.865

[529 rows x 2 columns]
```

[62]:
```
#creating box plot
sns.boxplot(x=sales_per_City["Sales"])
```

[62]: <Axes: xlabel='Sales'>

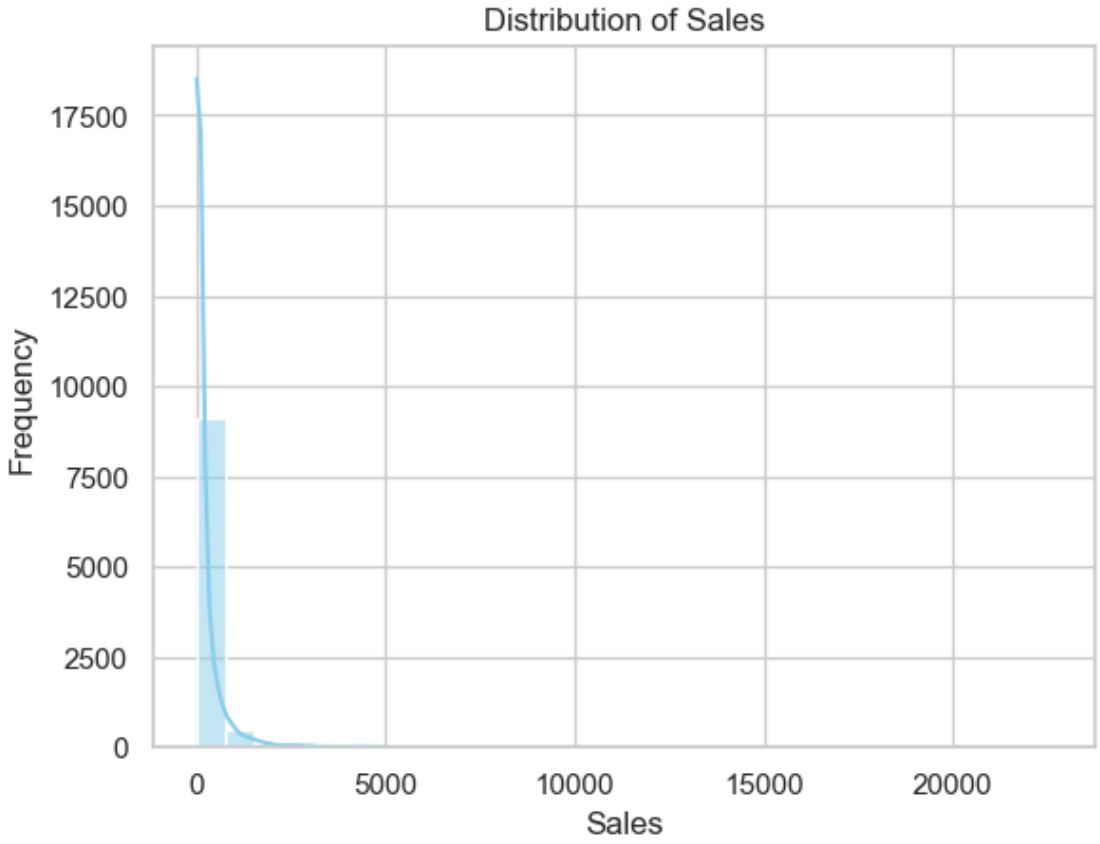


[28]:
```
#Descriptive Statistics of Sales
mean_sales = np.mean(df['Sales'])
median_sales = np.median(df['Sales'])
std_sales = np.std(df['Sales'])

# print
print(f"Mean Sales: {mean_sales}")
print(f"Median Sales: {median_sales}")
print(f"Standard Deviation Sales: {std_sales}")
```

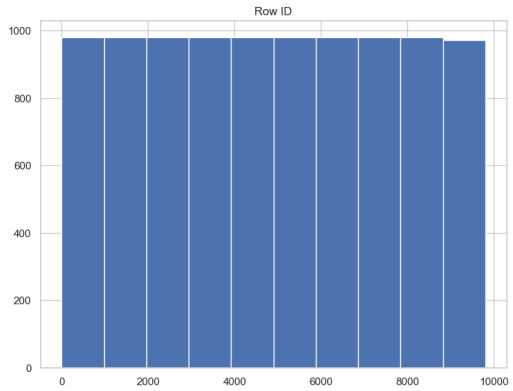
```
Mean Sales: 230.1161929410563
Median Sales: 54.384
Standard Deviation Sales: 625.2701391783485
```

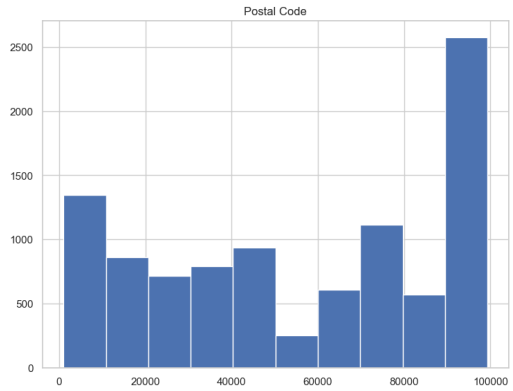
[29]:
```
#Creating histograms  Sales
```

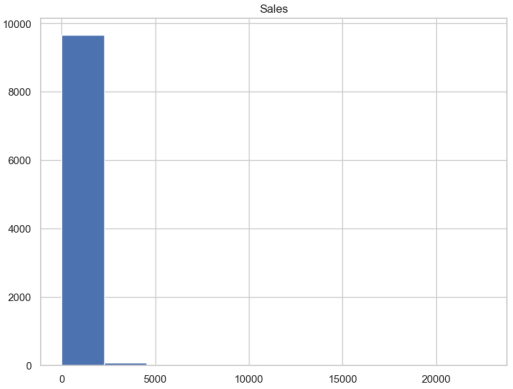
```
sns.set(style="whitegrid")

# Create a histogram for the 'sales' column
sns.histplot(df['Sales'], bins=30, kde=True, color='skyblue')

# Add labels and title
plt.xlabel('Sales')
plt.ylabel('Frequency')
plt.title('Distribution of Sales')

# Show the plot
plt.show()
```



[30]:
```
df.hist(figsize = (20,15))
plt.show()
```







[31]:
```
df1=df[['Row ID','Sales','Postal Code']]
print(df1)
```

```
      Row ID     Sales  Postal Code
0          1  261.9600      42420.0
1          2  731.9400      42420.0
2          3   14.6200      90036.0
3          4  957.5775      33311.0
4          5   22.3680      33311.0
...      ...       ...          ...
9795    9796    3.7980      60610.0
9796    9797   10.3680      43615.0
9797    9798  235.1880      43615.0
9798    9799   26.3760      43615.0
9799    9800   10.3840      43615.0

[9789 rows x 3 columns]
```

[32]:
```
correlation_matrix = df1.corr()

# Create an interactive heatmap using plotly
```

```
fig = px.imshow(correlation_matrix, x=correlation_matrix.columns,␣
 ↪y=correlation_matrix.index,
                labels=dict(color='Correlation'),␣
 ↪color_continuous_scale='Viridis')

fig.update_layout(title='Correlation Matrix')
fig.show()
```

[33]:
```
#types of product category
products_category=df['Category'].unique()
print(products_category)
```

```
['Furniture' 'Office Supplies' 'Technology']
```

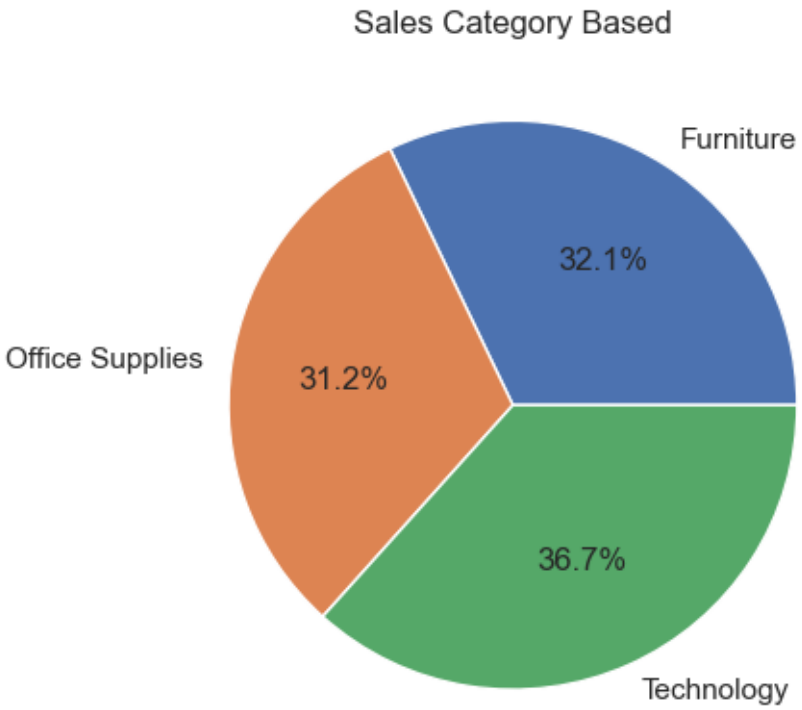
[34]:
```
# grouping the data  by procudt category
subcategory_count=df.groupby('Category')['Sub-Category'].nunique().reset_index()
subcategory_count = subcategory_count.
 ↪sort_values(by='Sub-Category',ascending=False)
print(subcategory_count)
```

```
          Category  Sub-Category
1  Office Supplies             9
0        Furniture             4
2       Technology             4
```

[35]:
```
#Sales per category
category_sales=df.groupby(['Category'])['Sales'].sum().reset_index()
print(category_sales)
```

```
          Category        Sales
0        Furniture  723538.4757
1  Office Supplies  703212.8240
2       Technology  825856.1130
```

[36]:
```
plt.pie(category_sales['Sales'],labels=category_sales['Category'],autopct='%1.
 ↪1f%%')

plt.title('Sales Category Based')

plt.show()
```

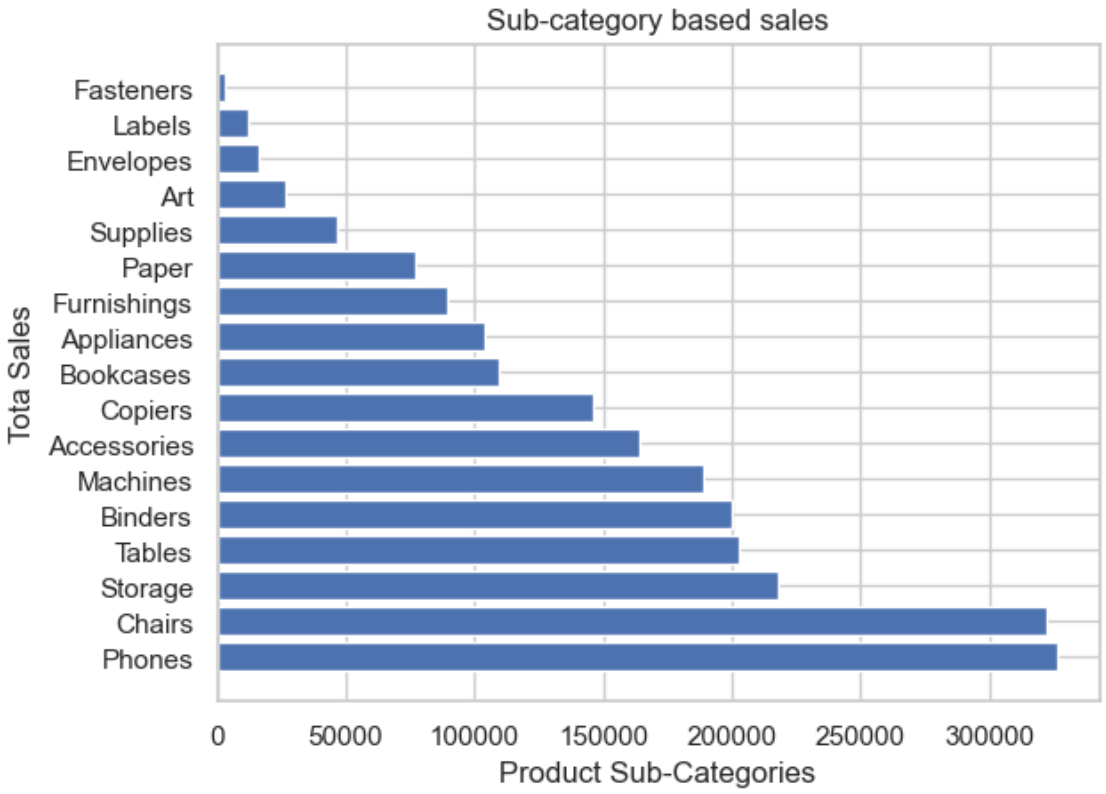


[37]:
```
#Group Data by product sub-category vs sales
sub_subcategory = df.groupby(['Sub-Category'])['Sales'].sum().reset_index()

#sorting  descending order
top_sub_subcategory = sub_subcategory.sort_values(by='Sales', ascending=False)
print(top_sub_subcategory)
```

```
   Sub-Category       Sales
13       Phones  326487.6980
5        Chairs  322107.5310
14      Storage  217779.1020
16       Tables  202810.6280
3       Binders  200028.7850
11     Machines  189238.6310
0   Accessories  163881.6900
6       Copiers  146248.0940
4     Bookcases  109408.2987
1    Appliances  104075.4630
9   Furnishings   89212.0180
12        Paper   76736.1040
15     Supplies   46420.3080
2           Art   26697.3700
7     Envelopes   16126.0060
```

```
10        Labels   12347.7260
8      Fasteners    3001.9600
```

[38]:
```
#ploting the graph
plt.barh(top_sub_subcategory['Sub-Category'],top_sub_subcategory['Sales'])
plt.title('Sub-category based sales')
plt.xlabel('Product Sub-Categories')
plt.ylabel('Tota Sales')
plt.show()
```



[63]:
```
# Showing Regions
region_sales=df['Region'].unique()
print(region_sales)
```

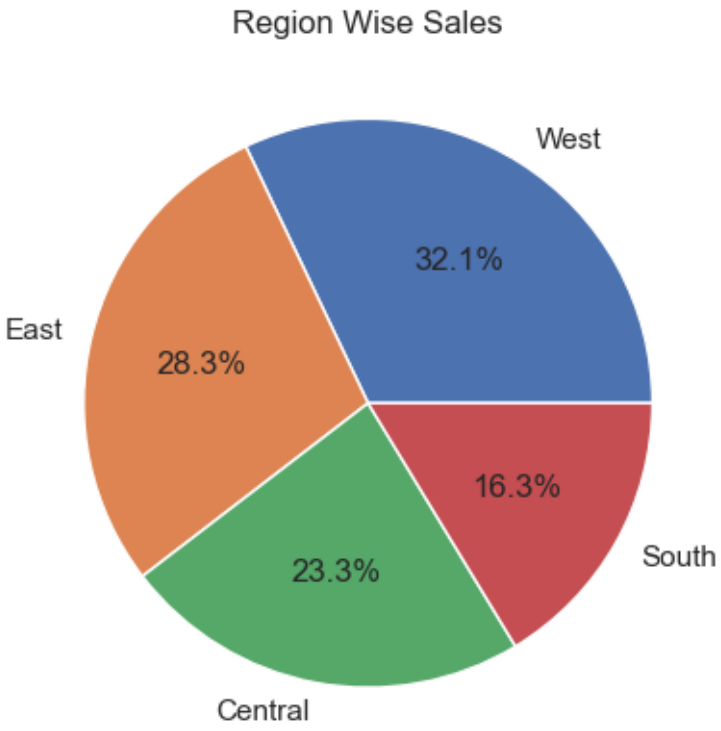
```
['South' 'West' 'Central' 'East']
```

[64]:
```
# Region wise sales
region_wise = df['Region'].value_counts().reset_index()

print(region_wise)
```

```
   Region  count
```

```
0     West   3140
1     East   2774
2  Central   2277
3    South   1598
```

[65]:
```
#plotting graph
plt.pie(region_wise['count'],labels=region_wise['Region'],autopct='%1.1f%%')

plt.title('Region Wise Sales')

plt.show()
```



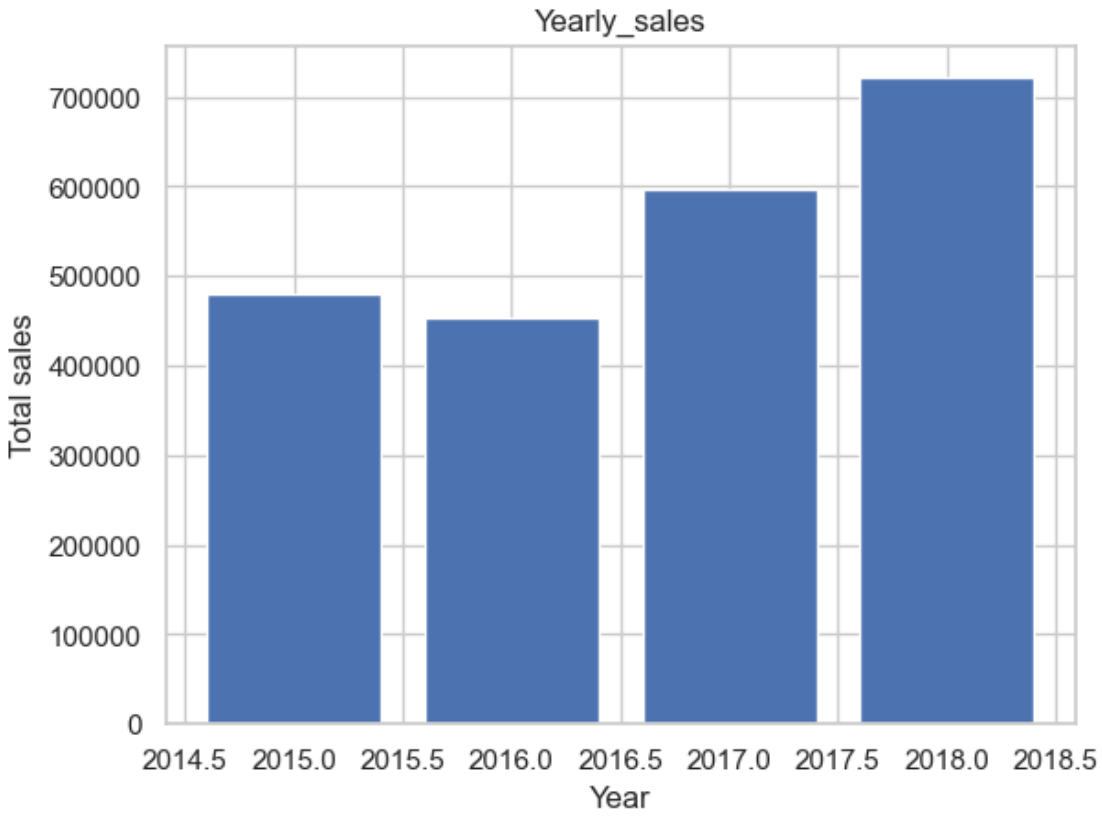
[39]:
```
import datetime

# Create a datetime object
my_datetime = datetime.datetime.now()

# Access the year using the 'year' attribute
year_value = my_datetime.year

# Print the year
print(year_value)
```

2024

[40]:
```
# converting the order date to date time format
df['Order Date'] = pd.to_datetime(df['Order Date'],dayfirst=True)

#goruping by year and summing the sales per year
yearly_sales = df.groupby(df['Order Date'].dt.year)['Sales'].sum()

yearly_sales = yearly_sales.reset_index()
yearly_sales = yearly_sales.rename(columns={'Order Date':'Year','Sales':'Total␣
 ↪Sales'})


print(yearly_sales)
```

```
   Year  Total Sales
0  2015  479856.2081
1  2016  454315.9054
2  2017  597225.4900
3  2018  721209.8092
```

[41]:
```
plt.bar(yearly_sales['Year'],yearly_sales['Total Sales'])

plt.title('Yearly_sales')
plt.xlabel('Year')
plt.ylabel('Total sales')
plt.show()
```

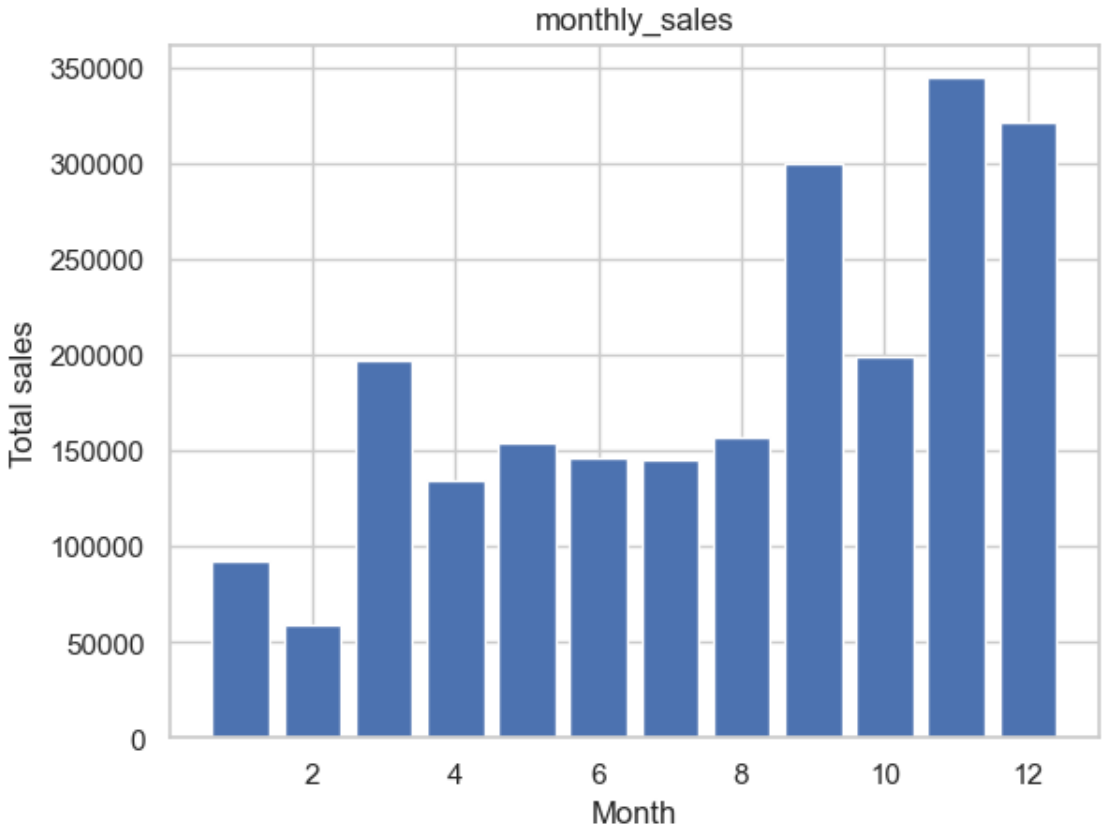


[42]:
```
df['Order Date'] = pd.to_datetime(df['Order Date'],dayfirst=True)

#goruping by year and summing the sales per year
monthly_sales = df.groupby(df['Order Date'].dt.month)['Sales'].sum()

monthly_sales = monthly_sales.reset_index()
monthly_sales = monthly_sales.rename(columns={'Order Date':'Month','Sales':
 ↪'Total Sales'})


print(monthly_sales)
```

```
   Month  Total Sales
0      1   91982.1396
1      2   59371.1154
2      3  197573.5872
3      4  134988.2506
4      5  154086.7237
5      6  145837.5233
6      7  145535.6890
7      8  157315.9270
```

```
8        9  300103.4117
9       10  199496.2947
10      11  345041.6110
11      12  321275.1395
```

[43]:
```
plt.bar(monthly_sales['Month'],monthly_sales['Total Sales'])

plt.title('monthly_sales')
plt.xlabel('Month')
plt.ylabel('Total sales')
plt.show()
```

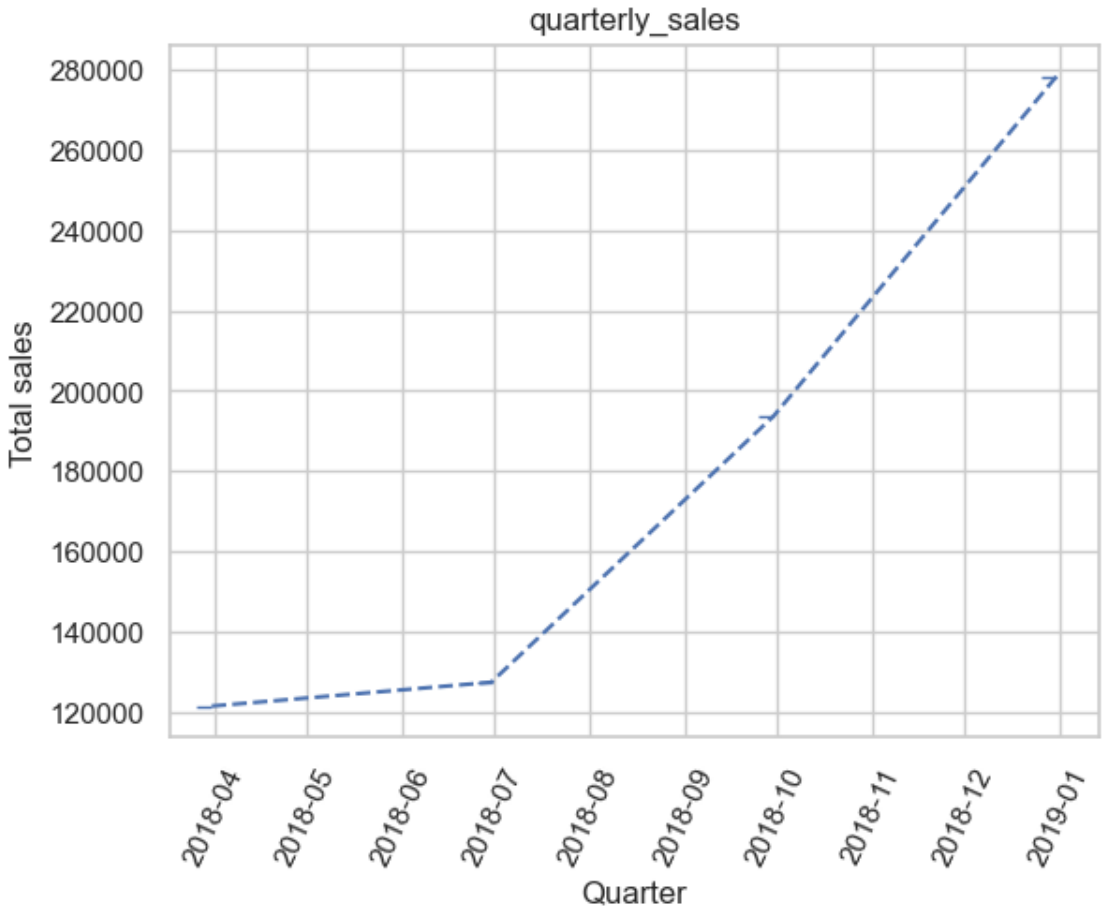


[44]:
```
df['Order Date'] = pd.to_datetime(df['Order Date'],dayfirst=True)

#fiter the data according to the year
yearly_sales = df[df['Order Date'].dt.year == 2018]

#calculate quarterly sales for the year 2018

quarterly_sales = yearly_sales.resample('Q',on='Order Date')['Sales'].sum()
quarterly_sales = quarterly_sales.reset_index()
```

```
quarterly_sales = quarterly_sales.rename(columns={'Order Date':
 ↪'Quarter','Sales':'Total Sales'})

print(quarterly_sales)
```

```
     Quarter  Total Sales
0 2018-03-31  121623.7042
1 2018-06-30  127558.6200
2 2018-09-30  193815.8400
3 2018-12-31  278211.6450
```

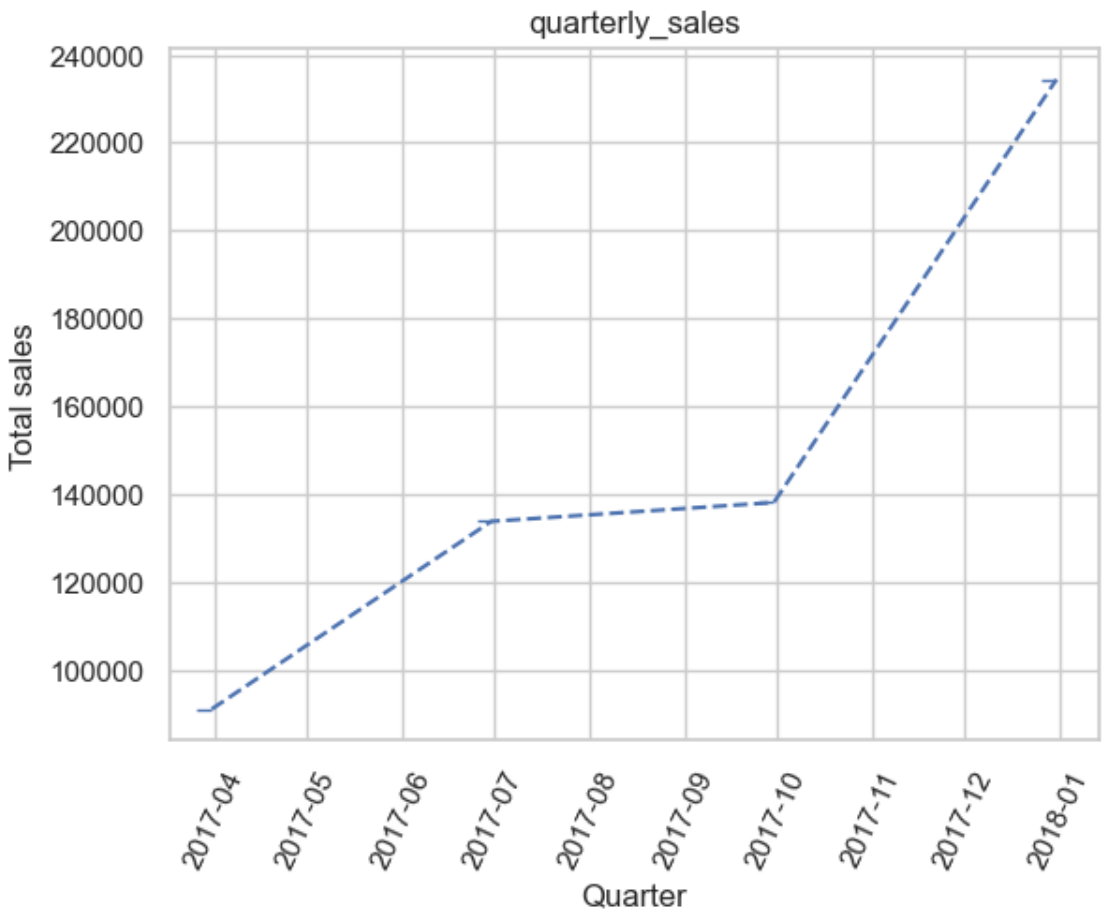
[45]:
```
plt.plot(quarterly_sales['Quarter'],quarterly_sales['Total Sales'], marker = 0,␣
 ↪linestyle='--')

plt.title('quarterly_sales')
plt.xlabel('Quarter')
plt.ylabel('Total sales')
plt.xticks(rotation=65)
plt.show()
```



[67]:
```
df['Order Date'] = pd.to_datetime(df['Order Date'],dayfirst=True)

#fiter the data according to the year
yearly_sales = df[df['Order Date'].dt.year == 2017]

#calculate quarterly sales for the year 2017

quarterly_sales = yearly_sales.resample('Q',on='Order Date')['Sales'].sum()
quarterly_sales = quarterly_sales.reset_index()
quarterly_sales = quarterly_sales.rename(columns={'Order Date':
 ↪'Quarter','Sales':'Total Sales'})

print(quarterly_sales)
```

```
     Quarter  Total Sales
0 2017-03-31   91014.0550
1 2017-06-30  133766.4110
2 2017-09-30  138056.3742
3 2017-12-31  234388.6498
```

[68]:
```
plt.plot(quarterly_sales['Quarter'],quarterly_sales['Total Sales'], marker = 0,␣
 ↪linestyle='--')

plt.title('quarterly_sales')
plt.xlabel('Quarter')
plt.ylabel('Total sales')
plt.xticks(rotation=65)
plt.show()
```

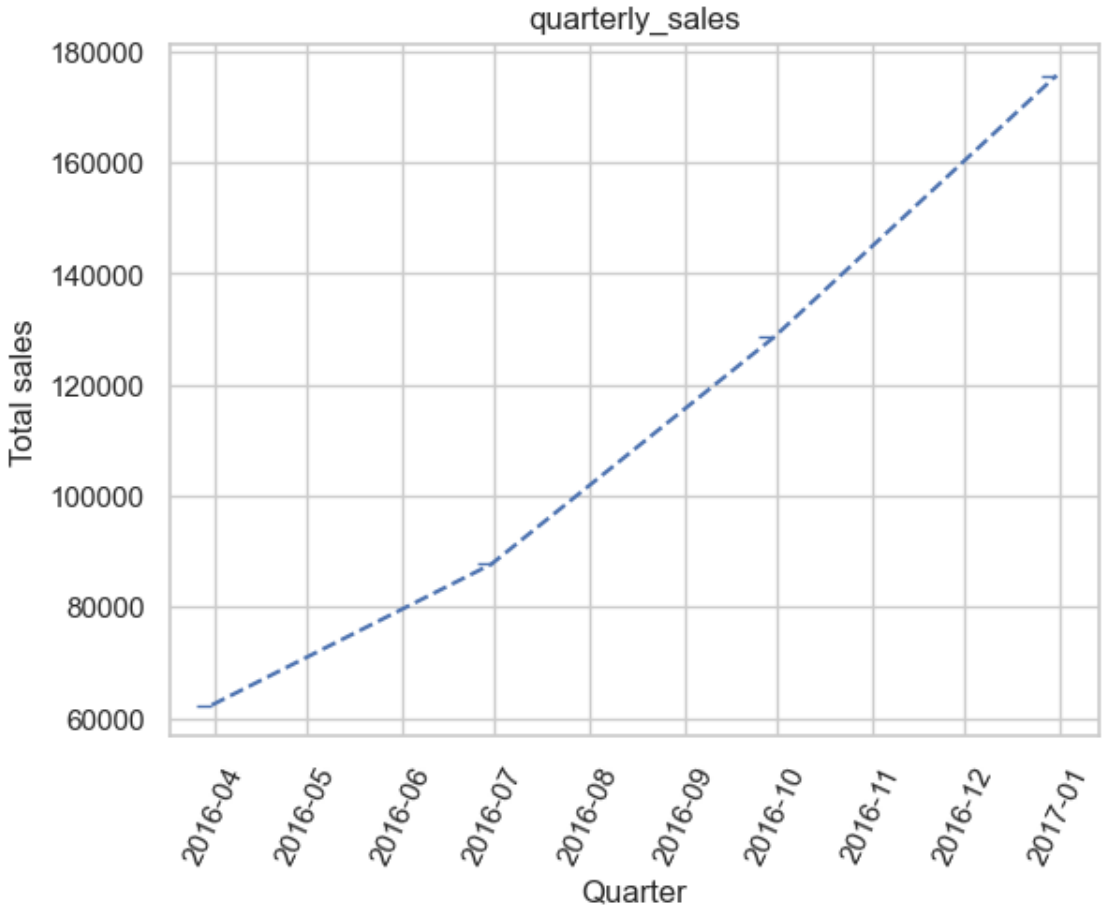


[69]:
```
df['Order Date'] = pd.to_datetime(df['Order Date'],dayfirst=True)

#fiter the data according to the year
yearly_sales = df[df['Order Date'].dt.year == 2016]

#calculate quarterly sales for the year 2016

quarterly_sales = yearly_sales.resample('Q',on='Order Date')['Sales'].sum()
quarterly_sales = quarterly_sales.reset_index()
quarterly_sales = quarterly_sales.rename(columns={'Order Date':
 ↪'Quarter','Sales':'Total Sales'})

print(quarterly_sales)
```

```
     Quarter  Total Sales
0 2016-03-31   62357.6870
1 2016-06-30   87713.3730
2 2016-09-30  128560.2072
```

```
3 2016-12-31  175684.6382
```

[70]:
```
plt.plot(quarterly_sales['Quarter'],quarterly_sales['Total Sales'], marker = 0,␣
 ↪linestyle='--')

plt.title('quarterly_sales')
plt.xlabel('Quarter')
plt.ylabel('Total sales')
plt.xticks(rotation=65)
plt.show()
```

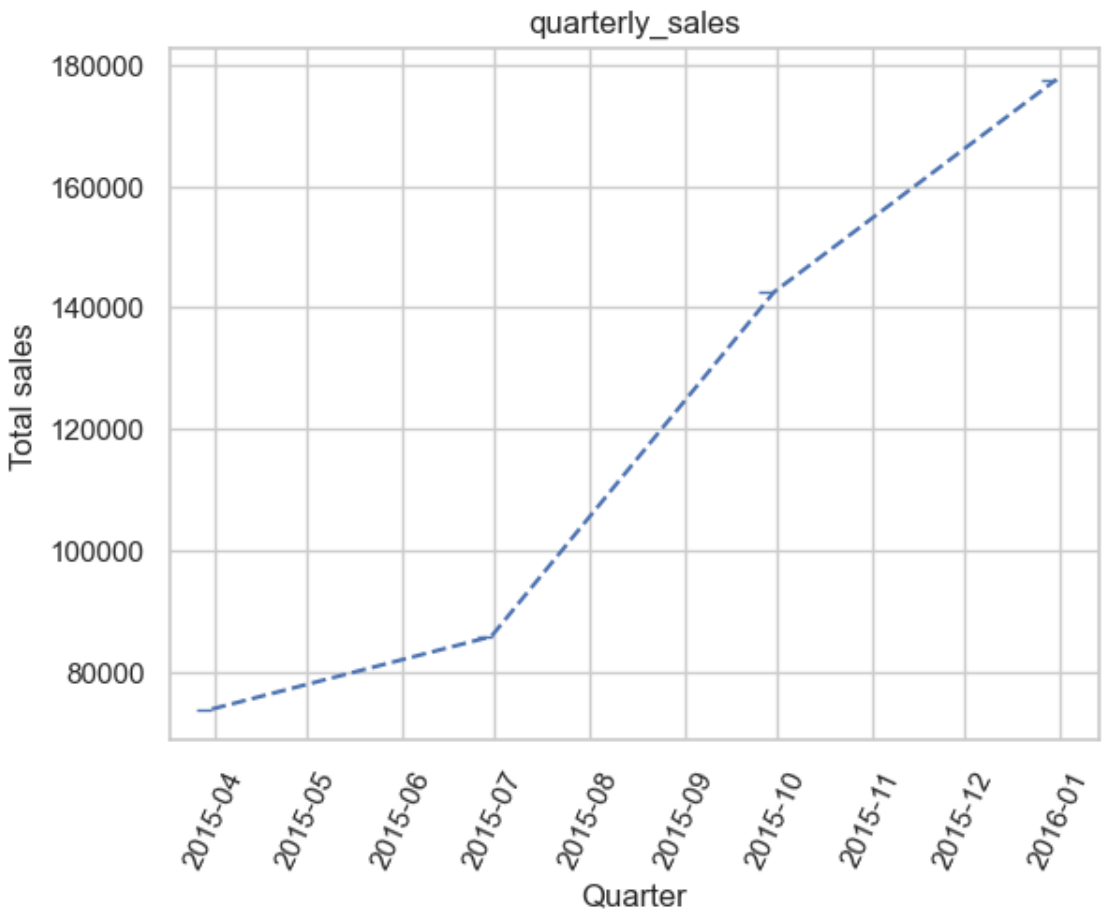


[71]:
```
df['Order Date'] = pd.to_datetime(df['Order Date'],dayfirst=True)

#fiter the data according to the year
yearly_sales = df[df['Order Date'].dt.year == 2015]

#calculate quarterly sales for the year 2015
```

```
quarterly_sales = yearly_sales.resample('Q',on='Order Date')['Sales'].sum()
quarterly_sales = quarterly_sales.reset_index()
quarterly_sales = quarterly_sales.rename(columns={'Order Date':
 ↪'Quarter','Sales':'Total Sales'})

print(quarterly_sales)
```

```
     Quarter  Total Sales
0 2015-03-31   73931.3960
1 2015-06-30   85874.0936
2 2015-09-30  142522.6063
3 2015-12-31  177528.1122
```

[72]:
```
plt.plot(quarterly_sales['Quarter'],quarterly_sales['Total Sales'], marker = 0,␣
 ↪linestyle='--')

plt.title('quarterly_sales')
plt.xlabel('Quarter')
plt.ylabel('Total sales')
plt.xticks(rotation=65)
plt.show()
```



[47]:
```
df
```

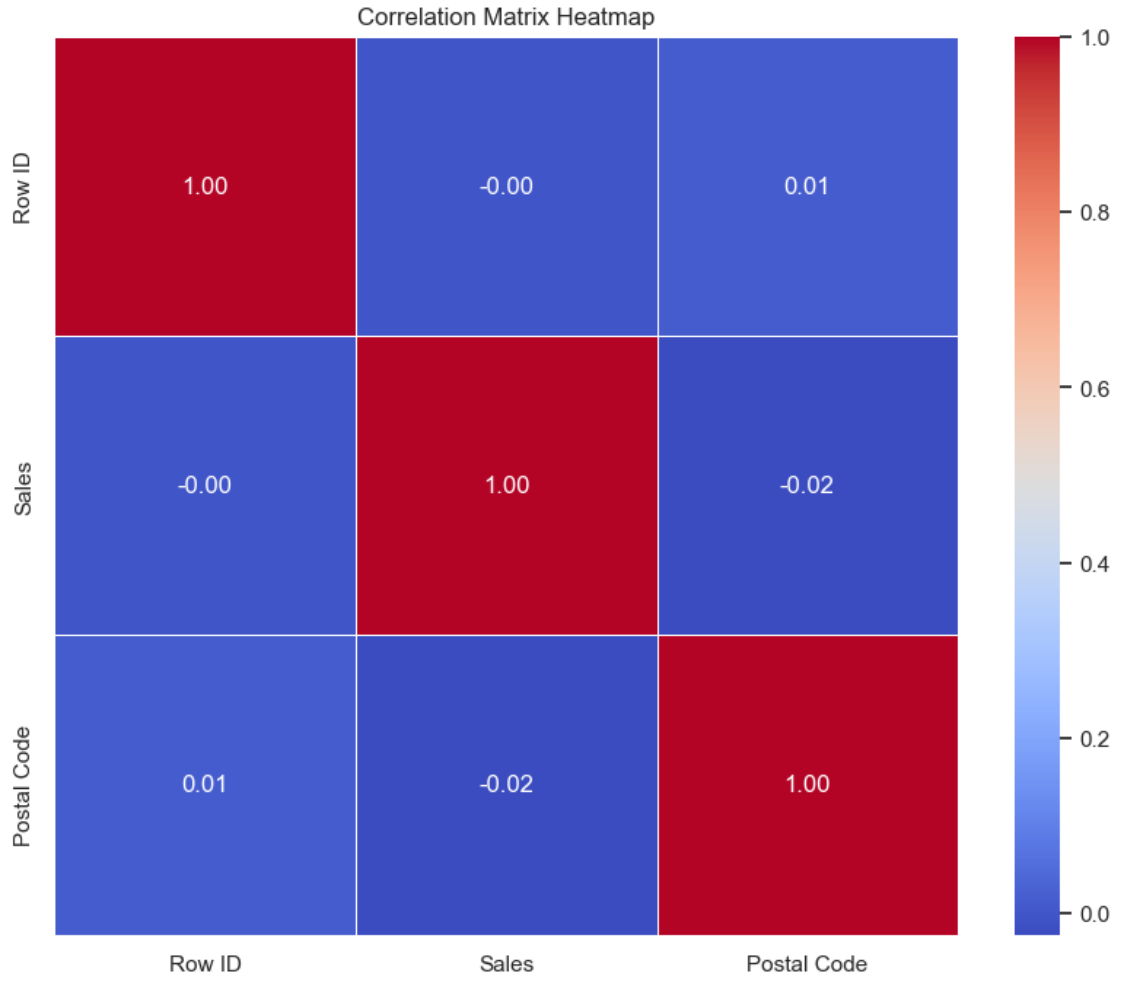
[47]:

	Row ID	Order ID	Order Date	Ship Date	Ship Mode	\
0	1	CA-2017-152156	2017-11-08	11/11/2017	Second Class	
1	2	CA-2017-152156	2017-11-08	11/11/2017	Second Class	
2	3	CA-2017-138688	2017-06-12	16/06/2017	Second Class	
3	4	US-2016-108966	2016-10-11	18/10/2016	Standard Class	
4	5	US-2016-108966	2016-10-11	18/10/2016	Standard Class	
…	…	…	…	…	…	
9795	9796	CA-2017-125920	2017-05-21	28/05/2017	Standard Class	
9796	9797	CA-2016-128608	2016-01-12	17/01/2016	Standard Class	
9797	9798	CA-2016-128608	2016-01-12	17/01/2016	Standard Class	
9798	9799	CA-2016-128608	2016-01-12	17/01/2016	Standard Class	
9799	9800	CA-2016-128608	2016-01-12	17/01/2016	Standard Class	

	Customer ID	Customer Name	Segment	Country	City	\
0	CG-12520	Claire Gute	Consumer	United States	Henderson	
1	CG-12520	Claire Gute	Consumer	United States	Henderson	

```
2       DV-13045   Darrin Van Huff  Corporate  United States      Los Angeles
3       SO-20335    Sean O'Donnell   Consumer  United States  Fort Lauderdale
4       SO-20335    Sean O'Donnell   Consumer  United States  Fort Lauderdale
…          …               …         …            …               …
9795    SH-19975     Sally Hughsby  Corporate  United States          Chicago
9796    CS-12490  Cindy Schnelling  Corporate  United States           Toledo
9797    CS-12490  Cindy Schnelling  Corporate  United States           Toledo
9798    CS-12490  Cindy Schnelling  Corporate  United States           Toledo
9799    CS-12490  Cindy Schnelling  Corporate  United States           Toledo

           State  Postal Code   Region       Product ID         Category  \
0       Kentucky      42420.0    South  FUR-BO-10001798        Furniture
1       Kentucky      42420.0    South  FUR-CH-10000454        Furniture
2     California      90036.0     West  OFF-LA-10000240  Office Supplies
3        Florida      33311.0    South  FUR-TA-10000577        Furniture
4        Florida      33311.0    South  OFF-ST-10000760  Office Supplies
…           …          …      …             …               …
9795    Illinois      60610.0  Central  OFF-BI-10003429  Office Supplies
9796        Ohio      43615.0     East  OFF-AR-10001374  Office Supplies
9797        Ohio      43615.0     East  TEC-PH-10004977       Technology
9798        Ohio      43615.0     East  TEC-PH-10000912       Technology
9799        Ohio      43615.0     East  TEC-AC-10000487       Technology

     Sub-Category                                         Product Name     Sales
0       Bookcases                     Bush Somerset Collection Bookcase  261.9600
1          Chairs  Hon Deluxe Fabric Upholstered Stacking Chairs,…  731.9400
2          Labels  Self-Adhesive Address Labels for Typewriters b…   14.6200
3          Tables       Bretford CR4500 Series Slim Rectangular Table  957.5775
4         Storage                         Eldon Fold 'N Roll Cart System   22.3680
…           …                                                    …       …
9795      Binders  Cardinal HOLDit! Binder Insert Strips,Extra St…    3.7980
9796          Art            BIC Brite Liner Highlighters, Chisel Tip   10.3680
9797       Phones                                          GE 30524EE4  235.1880
9798       Phones            Anker 24W Portable Micro USB Car Charger   26.3760
9799  Accessories                 SanDisk Cruzer 4 GB USB Flash Drive   10.3840

[9789 rows x 18 columns]
```

[48]:
```
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt='.2f',␣
 ↪linewidths=.5)
plt.title('Correlation Matrix Heatmap')
plt.show()
```



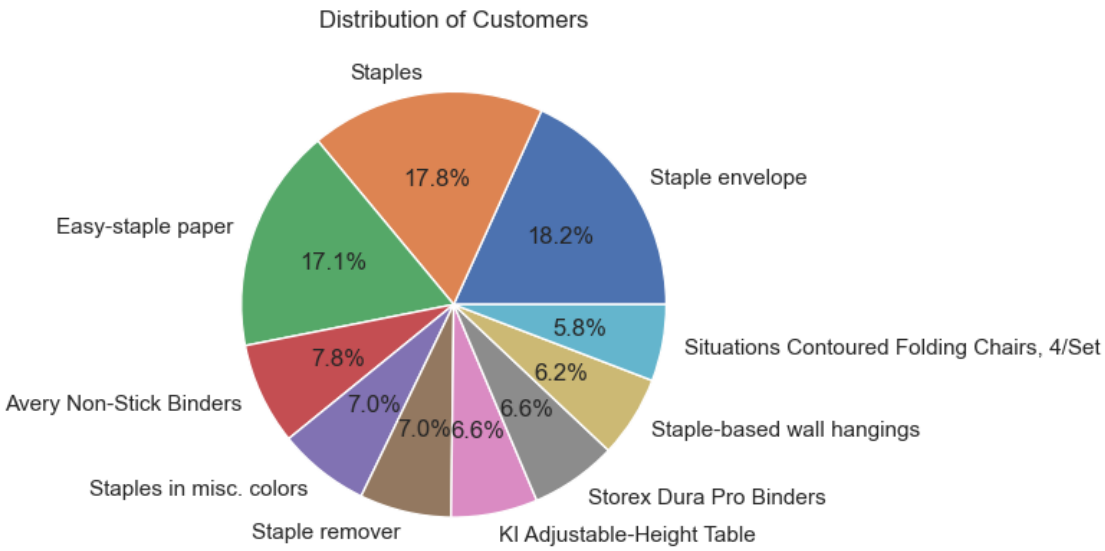
[49]:
```
#The 10 most Selling Products
most_product = df['Product Name'].value_counts().head(10)
most_product
```

[49]:
```
Product Name
Staple envelope                                47
Staples                                        46
Easy-staple paper                              44
Avery Non-Stick Binders                        20
Staples in misc. colors                        18
Staple remover                                 18
KI Adjustable-Height Table                     17
Storex Dura Pro Binders                        17
Staple-based wall hangings                     16
Situations Contoured Folding Chairs, 4/Set     15
Name: count, dtype: int64
```

[57]:
```
most_product = df['Product Name'].value_counts().reset_index().head(10)
#number_of_customers= number_of_customers.rename(columns={'Segment':'Customer
↪Type','count':'Total Customers'})

print(most_product)
```

```
                                Product Name  count
0                             Staple envelope     47
1                                     Staples     46
2                           Easy-staple paper     44
3                      Avery Non-Stick Binders     20
4                      Staples in misc. colors     18
5                               Staple remover     18
6                    KI Adjustable-Height Table     17
7                      Storex Dura Pro Binders     17
8                    Staple-based wall hangings     16
9  Situations Contoured Folding Chairs, 4/Set     15
```

[58]:
```
plt.pie(most_product['count'],labels=most_product['Product Name'],autopct='%1.
↪1f%%')

plt.title('Distribution of Customers')

plt.show()
```



[ ]:

[ ]:

[ ]:

[ ]: