Exploratory Data Analysis

June 20, 2023

1 Exploratory Data Analys (EDA)

1.1 Adidas sales dataset in the United States

This Python project showcases Exploratory Data Analysis (EDA) process applied to an Adidas sales dataset in the United States. It highlights key steps in data cleaning, preprocessing, and exploration, utilizing various Python libraries including Pandas, NumPy, Matplotlib, and Seaborn. The analysis encompasses visualization techniques such as bar charts, lollipop plots, stack area charts, polar bar plots, treemaps, donut charts, and more to uncover insights about sales trends, profitability, and regional performance. As a testament to the power of data analytics, this project provides valuable business insights which can help Adidas optimize their operations and strategies in the U.S. market."

[2]:
```
# Importing the libraries
import pandas as pd              #for data manipulation and analysis
import numpy as np               # for numerical computing that provides␣
 ↪support for handling arrays and mathematical operations
import matplotlib.pyplot as plt  #to create various types of visualizations
import seaborn as sns            #provides additional aesthetic and statistical␣
 ↪plotting capabilities
```

[3]:
```
# To load the dataset
df = pd.read_excel(r'C:\Users\tasne\OneDrive\Desktop\My fiels\Data␣
 ↪analysis\Python\Adidas.xlsx')
```

1.2 Data cleaning and preprocessing

[5]:
```
df.head()
```

[5]:
```
  Unnamed: 0   Unnamed: 1             Unnamed: 2           Unnamed: 3  \
0        NaN          NaN  Adidas Sales Database                  NaN
1        NaN          NaN                    NaN                  NaN
2        NaN          NaN                    NaN                  NaN
3        NaN     Retailer            Retailer ID         Invoice Date
4        NaN  Foot Locker                1185732  2020-01-01 00:00:00

 Unnamed: 4 Unnamed: 5 Unnamed: 6             Unnamed: 7      Unnamed: 8  \
0        NaN        NaN        NaN                    NaN             NaN
```

```
1        NaN        NaN        NaN                    NaN             NaN
2        NaN        NaN        NaN                    NaN             NaN
3     Region      State       City                Product  Price per Unit
4  Northeast   New York   New York  Men's Street Footwear              50

   Unnamed: 9  Unnamed: 10       Unnamed: 11       Unnamed: 12   Unnamed: 13
0         NaN          NaN               NaN               NaN           NaN
1         NaN          NaN               NaN               NaN           NaN
2         NaN          NaN               NaN               NaN           NaN
3  Units Sold  Total Sales  Operating Profit  Operating Margin  Sales Method
4        1200       600000            300000               0.5      In-store
```

[6]:

```
# to delete the first three rows
df = df.iloc[3:]

# to delete the first column
df = df.iloc[:, 1:]
```

[7]:

```
df.head()
```

[7]:

```
    Unnamed: 1   Unnamed: 2           Unnamed: 3 Unnamed: 4 Unnamed: 5  \
3     Retailer  Retailer ID         Invoice Date     Region      State
4  Foot Locker      1185732  2020-01-01 00:00:00  Northeast   New York
5  Foot Locker      1185732  2020-01-02 00:00:00  Northeast   New York
6  Foot Locker      1185732  2020-01-03 00:00:00  Northeast   New York
7  Foot Locker      1185732  2020-01-04 00:00:00  Northeast   New York

  Unnamed: 6                 Unnamed: 7      Unnamed: 8  Unnamed: 9  \
3       City                    Product  Price per Unit  Units Sold
4   New York      Men's Street Footwear              50        1200
5   New York    Men's Athletic Footwear              50        1000
6   New York    Women's Street Footwear              40        1000
7   New York  Women's Athletic Footwear              45         850

   Unnamed: 10       Unnamed: 11       Unnamed: 12   Unnamed: 13
3  Total Sales  Operating Profit  Operating Margin  Sales Method
4       600000            300000               0.5      In-store
5       500000            150000               0.3      In-store
6       400000            140000              0.35      In-store
7       382500            133875              0.35      In-store
```

[8]:

```
# to set the index to start from 0
df = df.reset_index(drop=True)

# to use the first row as column headers
df.columns = df.iloc[0]
```

```
# to delete the duplicate row containing column headers
df = df[1:].reset_index(drop=True)
```

[9]:

```
df.head()
```

[9]:

```
0     Retailer Retailer ID         Invoice Date     Region     State  \
0  Foot Locker     1185732  2020-01-01 00:00:00  Northeast  New York
1  Foot Locker     1185732  2020-01-02 00:00:00  Northeast  New York
2  Foot Locker     1185732  2020-01-03 00:00:00  Northeast  New York
3  Foot Locker     1185732  2020-01-04 00:00:00  Northeast  New York
4  Foot Locker     1185732  2020-01-05 00:00:00  Northeast  New York

0      City                    Product Price per Unit Units Sold Total Sales  \
0  New York      Men's Street Footwear             50       1200      600000
1  New York    Men's Athletic Footwear             50       1000      500000
2  New York    Women's Street Footwear             40       1000      400000
3  New York  Women's Athletic Footwear             45        850      382500
4  New York              Men's Apparel             60        900      540000

0 Operating Profit Operating Margin Sales Method
0           300000              0.5     In-store
1           150000              0.3     In-store
2           140000             0.35     In-store
3           133875             0.35     In-store
4           162000              0.3     In-store
```

[10]:

```
# to display the shape of the dataset (rows, columns)
df.shape
```

[10]:

```
(9648, 13)
```

[11]:

```
# to display the columns on the dataset
df.columns
```

[11]:

```
Index(['Retailer', 'Retailer ID', 'Invoice Date', 'Region', 'State', 'City',
       'Product', 'Price per Unit', 'Units Sold', 'Total Sales',
       'Operating Profit', 'Operating Margin', 'Sales Method'],
      dtype='object', name=0)
```

[12]:

```
# to check the type of each column
df.dtypes
```

[12]:

```
0
Retailer            object
Retailer ID         object
Invoice Date        object
Region              object
State               object
```

```
City                object
Product             object
Price per Unit      object
Units Sold          object
Total Sales         object
Operating Profit    object
Operating Margin    object
Sales Method        object
dtype: object
```

[13]:

```python
# to convert "Invoice Date" to datetime
df['Invoice Date'] = pd.to_datetime(df['Invoice Date'])
```

[14]:

```python
# to check the data type of the "Invoice Date" column
print(df['Invoice Date'].dtypes)
```

```
datetime64[ns]
```

[15]:

```python
# to convert columns to float data type
df['Price per Unit'] = df['Price per Unit'].astype(float)
df['Units Sold'] = df['Units Sold'].astype(float)
df['Total Sales'] = df['Total Sales'].astype(float)
df['Operating Profit'] = df['Operating Profit'].astype(float)
```

[16]:

```python
df['Operating Margin'] = df['Operating Margin']
```

[19]:

```python
df['Operating Margin'] = df['Operating Margin'].astype(float)

# to check the type of each column
df.dtypes
```

[19]:

```
0
Retailer                    object
Retailer ID                 object
Invoice Date        datetime64[ns]
Region                      object
State                       object
City                        object
Product                     object
Price per Unit             float64
Units Sold                 float64
Total Sales                float64
Operating Profit           float64
Operating Margin           float64
Sales Method                object
dtype: object
```

[20]:

```python
df.head()
```

```
[20]: 0     Retailer Retailer ID Invoice Date     Region     State      City  \
      0  Foot Locker     1185732   2020-01-01  Northeast  New York  New York
      1  Foot Locker     1185732   2020-01-02  Northeast  New York  New York
      2  Foot Locker     1185732   2020-01-03  Northeast  New York  New York
      3  Foot Locker     1185732   2020-01-04  Northeast  New York  New York
      4  Foot Locker     1185732   2020-01-05  Northeast  New York  New York

      0                    Product  Price per Unit  Units Sold  Total Sales  \
      0      Men's Street Footwear            50.0      1200.0     600000.0
      1    Men's Athletic Footwear            50.0      1000.0     500000.0
      2    Women's Street Footwear            40.0      1000.0     400000.0
      3  Women's Athletic Footwear            45.0       850.0     382500.0
      4              Men's Apparel            60.0       900.0     540000.0

      0  Operating Profit  Operating Margin Sales Method
      0          300000.0              0.50     In-store
      1          150000.0              0.30     In-store
      2          140000.0              0.35     In-store
      3          133875.0              0.35     In-store
      4          162000.0              0.30     In-store
```

```
[21]: # to get an overview of the dataset
      print(df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9648 entries, 0 to 9647
Data columns (total 13 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   Retailer          9648 non-null   object
 1   Retailer ID       9648 non-null   object
 2   Invoice Date      9648 non-null   datetime64[ns]
 3   Region            9648 non-null   object
 4   State             9648 non-null   object
 5   City              9648 non-null   object
 6   Product           9648 non-null   object
 7   Price per Unit    9648 non-null   float64
 8   Units Sold        9648 non-null   float64
 9   Total Sales       9648 non-null   float64
 10  Operating Profit  9648 non-null   float64
 11  Operating Margin  9648 non-null   float64
 12  Sales Method      9648 non-null   object
dtypes: datetime64[ns](1), float64(5), object(7)
memory usage: 980.0+ KB
None
```

2 EDA

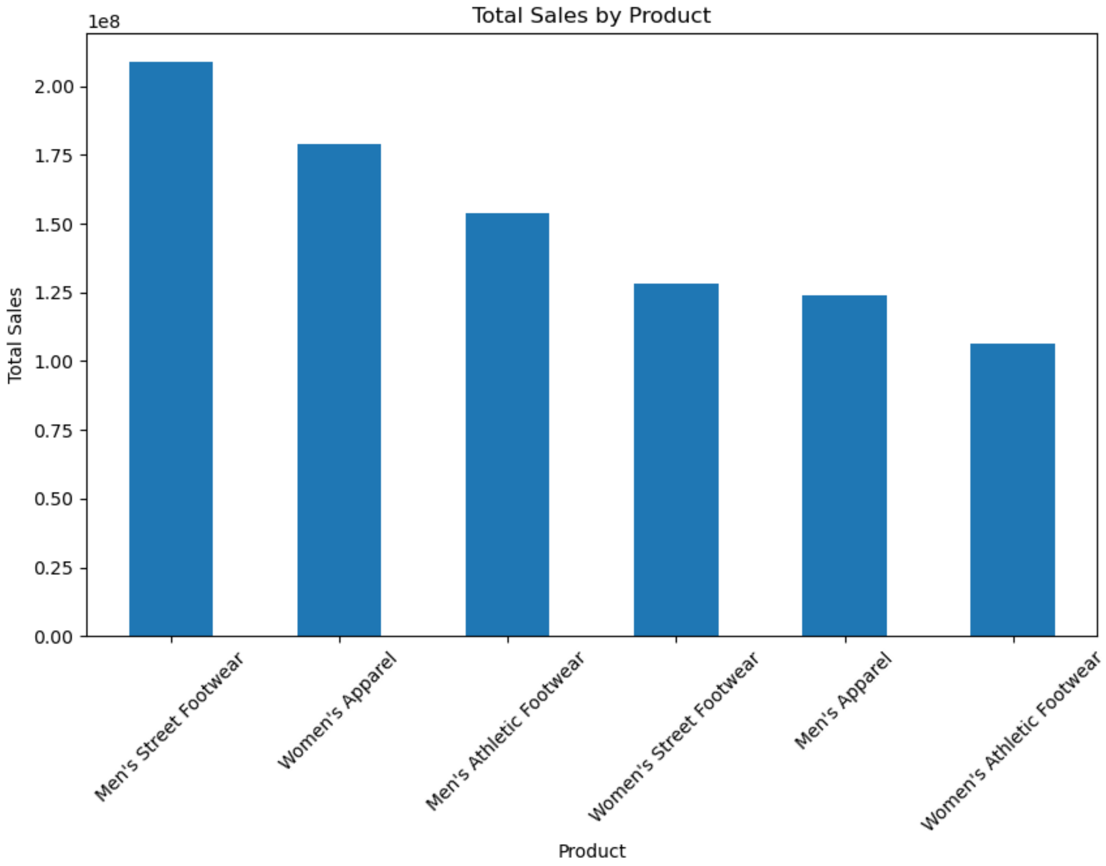
[22]:

```
# to display Summary statistics
print(df.describe())
```

```
0      Price per Unit   Units Sold    Total Sales  Operating Profit  \
count     9648.000000  9648.000000    9648.000000       9648.000000
mean        45.216625   256.930037   93273.437500      34425.244761
std         14.705397   214.252030  141916.016727      54193.113713
min          7.000000     0.000000       0.000000          0.000000
25%         35.000000   106.000000    4254.500000       1921.752500
50%         45.000000   176.000000    9576.000000       4371.420000
75%         55.000000   350.000000  150000.000000      52062.500000
max        110.000000  1275.000000  825000.000000     390000.000000

0      Operating Margin
count       9648.000000
mean           0.422991
std            0.097197
min            0.100000
25%            0.350000
50%            0.410000
75%            0.490000
max            0.800000
```

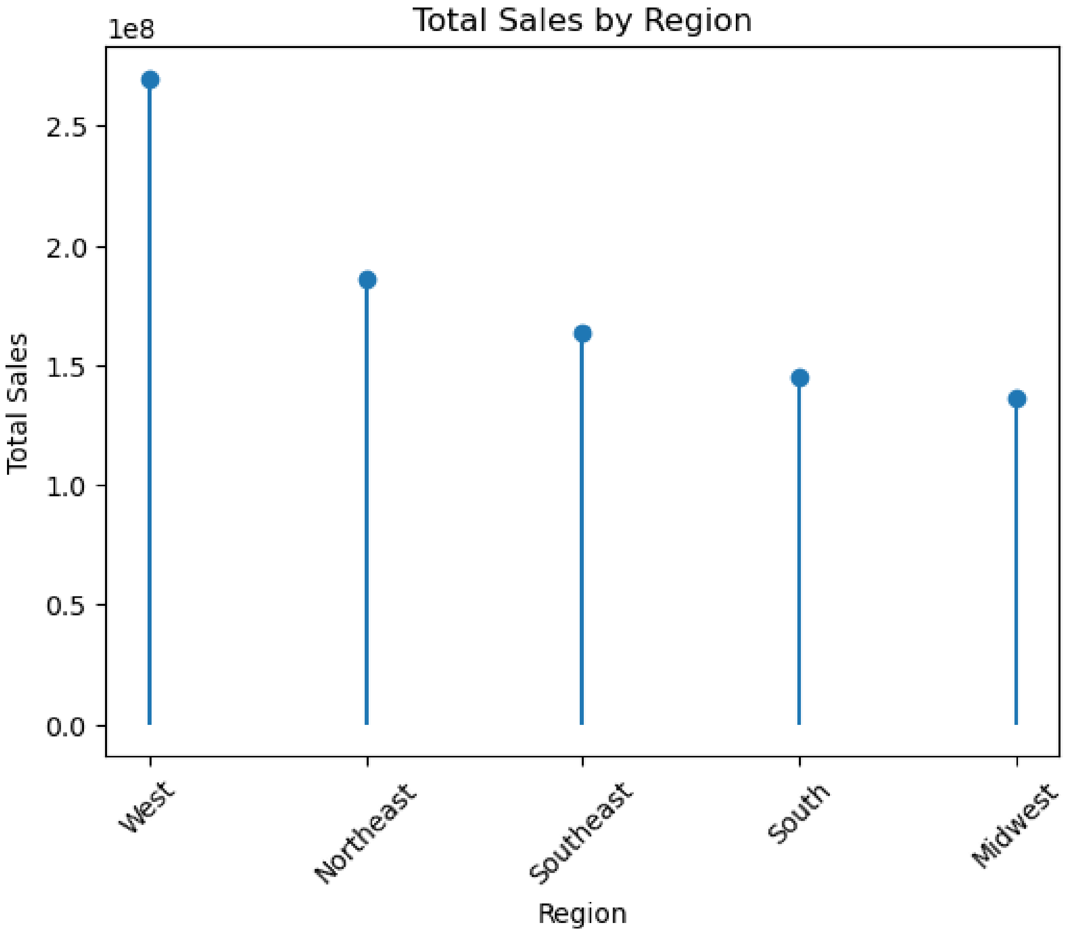
[24]:

```
import matplotlib.pyplot as plt

# Groupping the data by product and calculate the total sales for each product
product_sales = df.groupby('Product')['Total Sales'].sum().
  ↪sort_values(ascending=False)

# Plotting the bar chart
plt.figure(figsize=(10, 6))
product_sales.plot(kind='bar')
plt.title('Total Sales by Product')
plt.xlabel('Product')
plt.ylabel('Total Sales')
plt.xticks(rotation=45)
plt.show()
```



[40]:

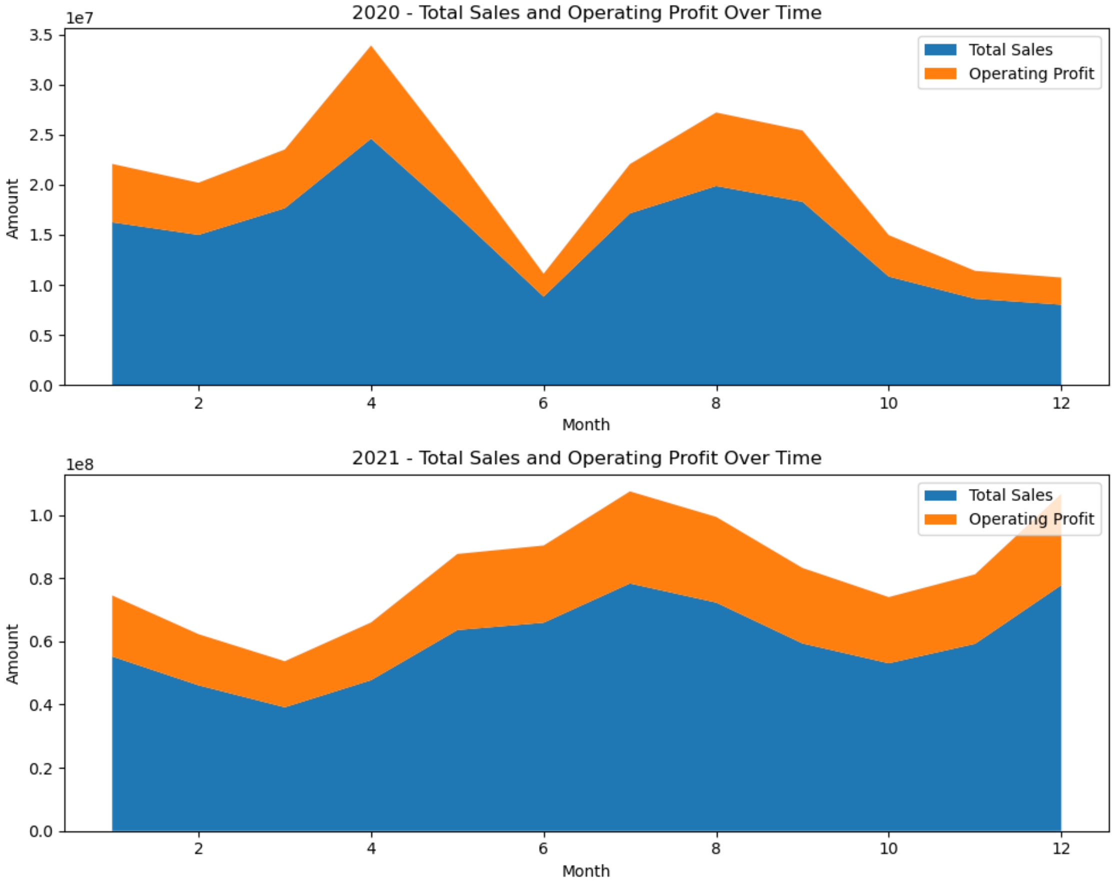
```python
import pandas as pd
import matplotlib.pyplot as plt

# Calculating total sales by region
region_sales = df.groupby('Region')['Total Sales'].sum().reset_index()

# Sorting the data by total sales in descending order
region_sales = region_sales.sort_values(by='Total Sales', ascending=False)

# Creating a lollipop plot
plt.stem(region_sales['Region'], region_sales['Total Sales'], basefmt=' ')
plt.xlabel('Region')
plt.ylabel('Total Sales')
plt.title('Total Sales by Region')
plt.xticks(rotation=45)

plt.show()
```



[42]:

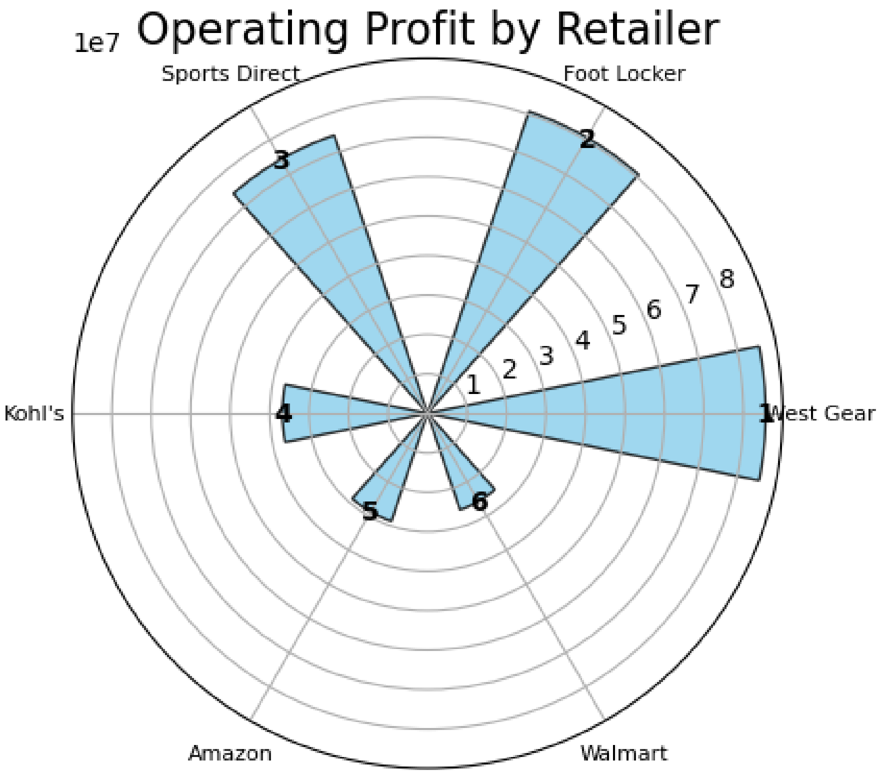
```
import pandas as pd
import matplotlib.pyplot as plt


# Filtering the data for the years 2020 and 2021
df_2020 = df[df['Invoice Date'].dt.year == 2020]
df_2021 = df[df['Invoice Date'].dt.year == 2021]

# Calculating the total sales and operating profit by month for each year
sales_2020 = df_2020.groupby(df_2020['Invoice Date'].dt.month)['Total Sales'].
  ↪sum()
profit_2020 = df_2020.groupby(df_2020['Invoice Date'].dt.month)['Operating␣
  ↪Profit'].sum()

sales_2021 = df_2021.groupby(df_2021['Invoice Date'].dt.month)['Total Sales'].
  ↪sum()
```

```python
profit_2021 = df_2021.groupby(df_2021['Invoice Date'].dt.month)['Operating Profit'].sum()

# Creating two separate stacked area charts for 2020 and 2021
fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(10, 8))

ax1.stackplot(sales_2020.index, [sales_2020, profit_2020], labels=['Total Sales', 'Operating Profit'])
ax1.set_xlabel('Month')
ax1.set_ylabel('Amount')
ax1.set_title('2020 - Total Sales and Operating Profit Over Time')
ax1.legend()

ax2.stackplot(sales_2021.index, [sales_2021, profit_2021], labels=['Total Sales', 'Operating Profit'])
ax2.set_xlabel('Month')
ax2.set_ylabel('Amount')
ax2.set_title('2021 - Total Sales and Operating Profit Over Time')
ax2.legend()

plt.tight_layout()
plt.show()
```



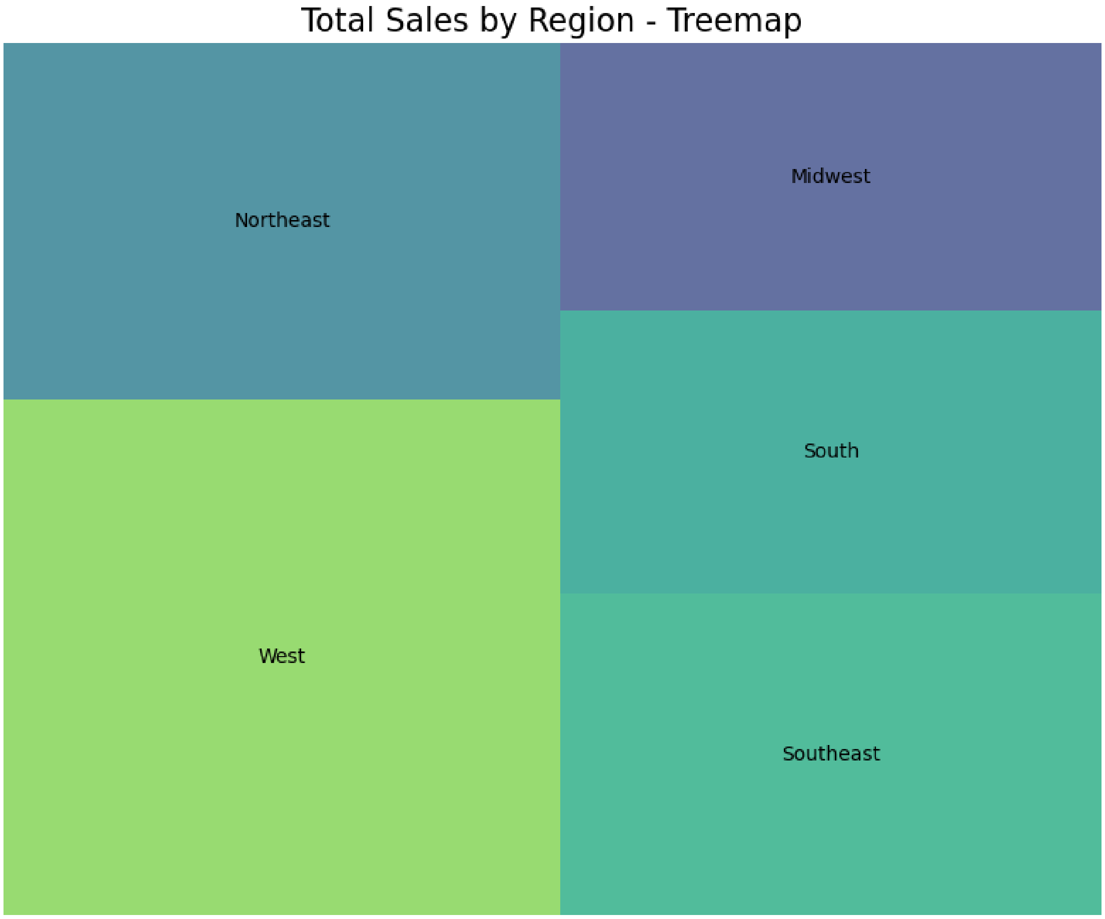
[62]:

```
import numpy as np
import matplotlib.pyplot as plt

# Grouping the data by retailer and calculate the total operating profit
profit_by_retailer = df.groupby('Retailer')['Operating Profit'].sum()

# Sorting the data by operating profit in descending order
profit_by_retailer = profit_by_retailer.sort_values(ascending=False)

# to create a polar bar plot
fig, ax = plt.subplots(subplot_kw={'projection': 'polar'})
theta = np.linspace(0, 2 * np.pi, len(profit_by_retailer), endpoint=False)
bars = ax.bar(theta, profit_by_retailer, width=0.4)

# to Customize the bars
for i, bar in enumerate(bars):
    bar.set_alpha(0.8)
    bar.set_color('skyblue')
    bar.set_edgecolor('black')
    # Add the ranking number as text with red color and bold style
    ax.text(theta[i], profit_by_retailer[i] + 5000, str(i + 1), ha='center',␣
 ↪va='center', fontsize=10, color='black', weight='bold')

# to set the ticks and labels
ticks = np.arange(0, 2 * np.pi, 2 * np.pi / len(profit_by_retailer))
labels = profit_by_retailer.index
ax.set_xticks(ticks)
ax.set_xticklabels(labels, fontsize=8)

# to Set the title
ax.set_title('Operating Profit by Retailer', fontsize=16)

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



[66]:
```
import matplotlib.pyplot as plt
import squarify

# Grouping the data by region and calculate the total sales
sales_by_region = df.groupby('Region')['Total Sales'].sum()
```

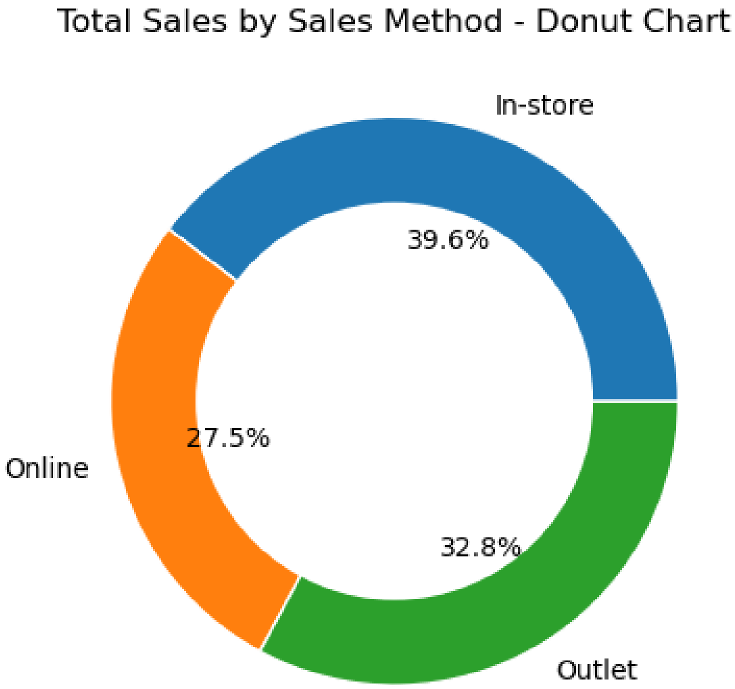
```python

# Sorting the data by total sales in descending order
sales_by_region = sales_by_region.sort_values(ascending=False)

# to Generate the treemap
plt.figure(figsize=(10, 8))
squarify.plot(sizes=sales_by_region, label=sales_by_region.index, alpha=0.8)

# to Add labels and title
plt.title('Total Sales by Region - Treemap', fontsize=16)
plt.axis('off')

# to Show the treemap
plt.show()
```



[68]:

```python
import matplotlib.pyplot as plt

# to Group the data by sales method and calculate the total sales
```

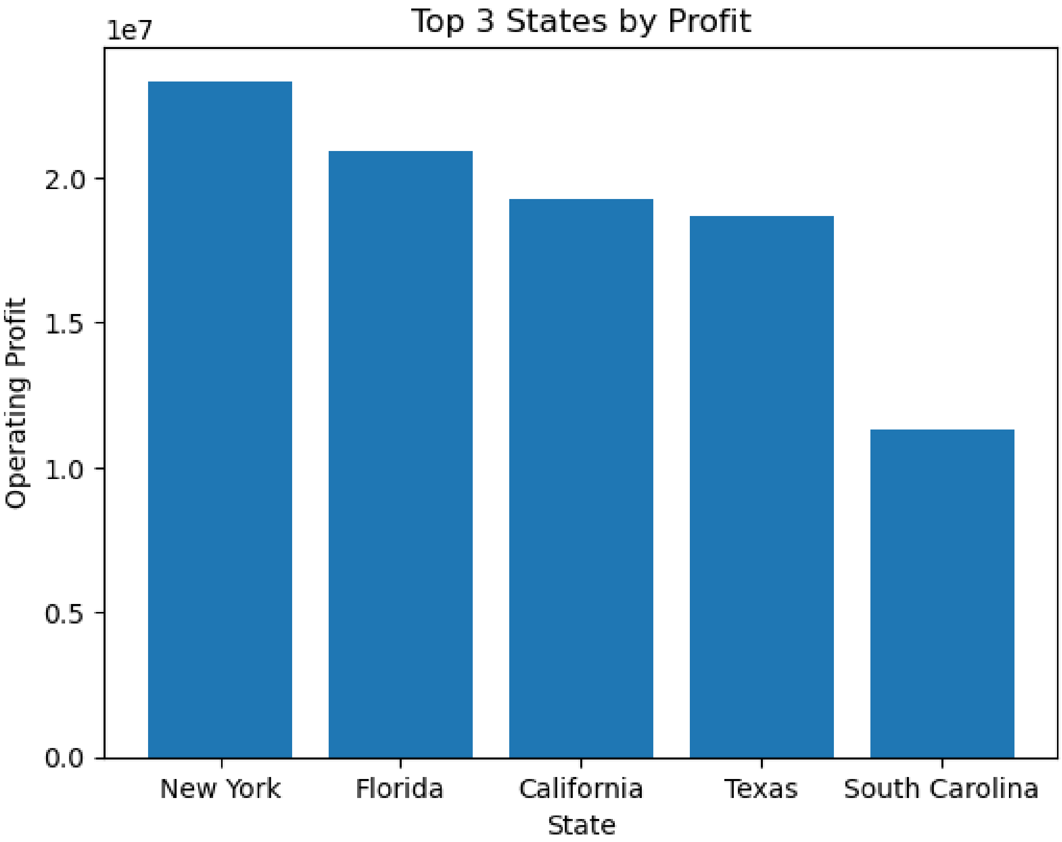
```
sales_by_method = df.groupby('Sales Method')['Total Sales'].sum()

# to Create the donut chart
plt.pie(sales_by_method, labels=sales_by_method.index, autopct='%1.1f%%',␣
 ↪wedgeprops={'edgecolor': 'white'})

# to Draw a white circle at the center to create the donut shape
center_circle = plt.Circle((0, 0), 0.70, fc='white')
fig = plt.gcf()
fig.gca().add_artist(center_circle)

# to Set title
plt.title('Total Sales by Sales Method - Donut Chart')

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



[70]:

```
import matplotlib.pyplot as plt

# Grouping the data by state and calculate the total profit
profit_by_state = df.groupby('State')['Operating Profit'].sum()
```

```python
# to Sort the data by profit in descending order and select the top 3 states
top_3_states = profit_by_state.sort_values(ascending=False).head(5)

# to Create the bar plot
plt.bar(top_3_states.index, top_3_states)

# to Set labels and title
plt.xlabel('State')
plt.ylabel('Operating Profit')
plt.title('Top 3 States by Profit')

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



```
[ ]:
```