I.Libraries and Data Handling

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

```
# Read the CSV file
df = pd.read_csv("01_Adidas Sales Analysis.csv")

# Display the first few rows of the DataFrame
print(df.head())
```

```
       Retailer  Retailer ID                Invoice Date     Region  \
0  Foot Locker      1185732    Tuesday, October 26, 2021  Northeast
1  Foot Locker      1185732  Wednesday, October 27, 2021  Northeast
2  Foot Locker      1185732   Thursday, October 28, 2021  Northeast
3  Foot Locker      1185732     Friday, October 29, 2021  Northeast
4  Foot Locker      1185732   Saturday, October 30, 2021  Northeast

          State          City Gender Type   Product Category  Price per Unit  \
0  Pennsylvania  Philadelphia         Men            Apparel              55
1  Pennsylvania  Philadelphia       Women            Apparel              45
2  Pennsylvania  Philadelphia         Men    Street Footwear              45
3  Pennsylvania  Philadelphia         Men  Athletic Footwear              45
4  Pennsylvania  Philadelphia       Women    Street Footwear              35

   Units Sold  Total Sales  Operating Profit  Operating Margin Sales Method
0         125        68750           24062.5              0.35       Outlet
1         225       101250           30375.0              0.30       Outlet
2         475       213750          117562.5              0.55       Outlet
3         125        56250           19687.5              0.35       Outlet
4         175        61250           24500.0              0.40       Outlet
```

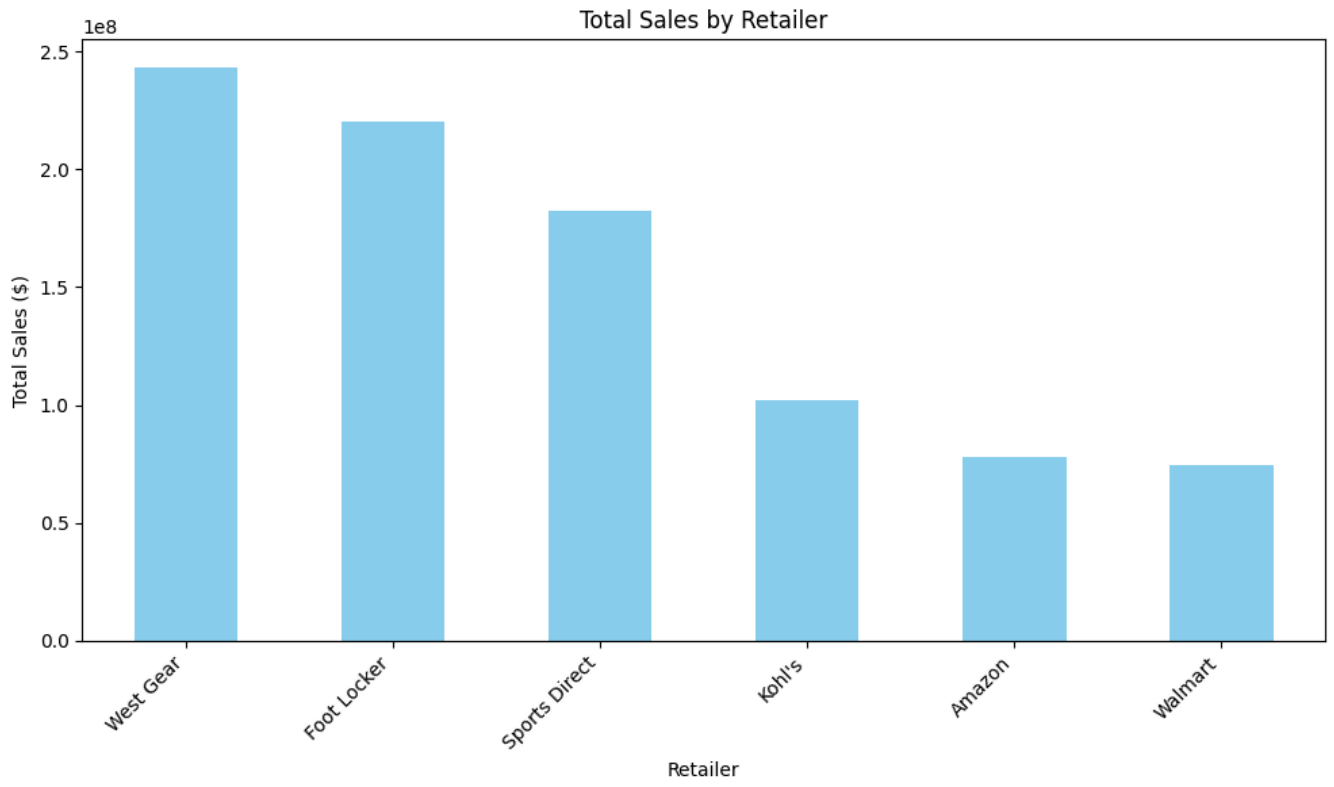
II. Data Analysis Techniques

```
# Compute descriptive statistics for numerical columns
descriptive_stats = df.describe()

# Display the descriptive statistics
print(descriptive_stats)
```

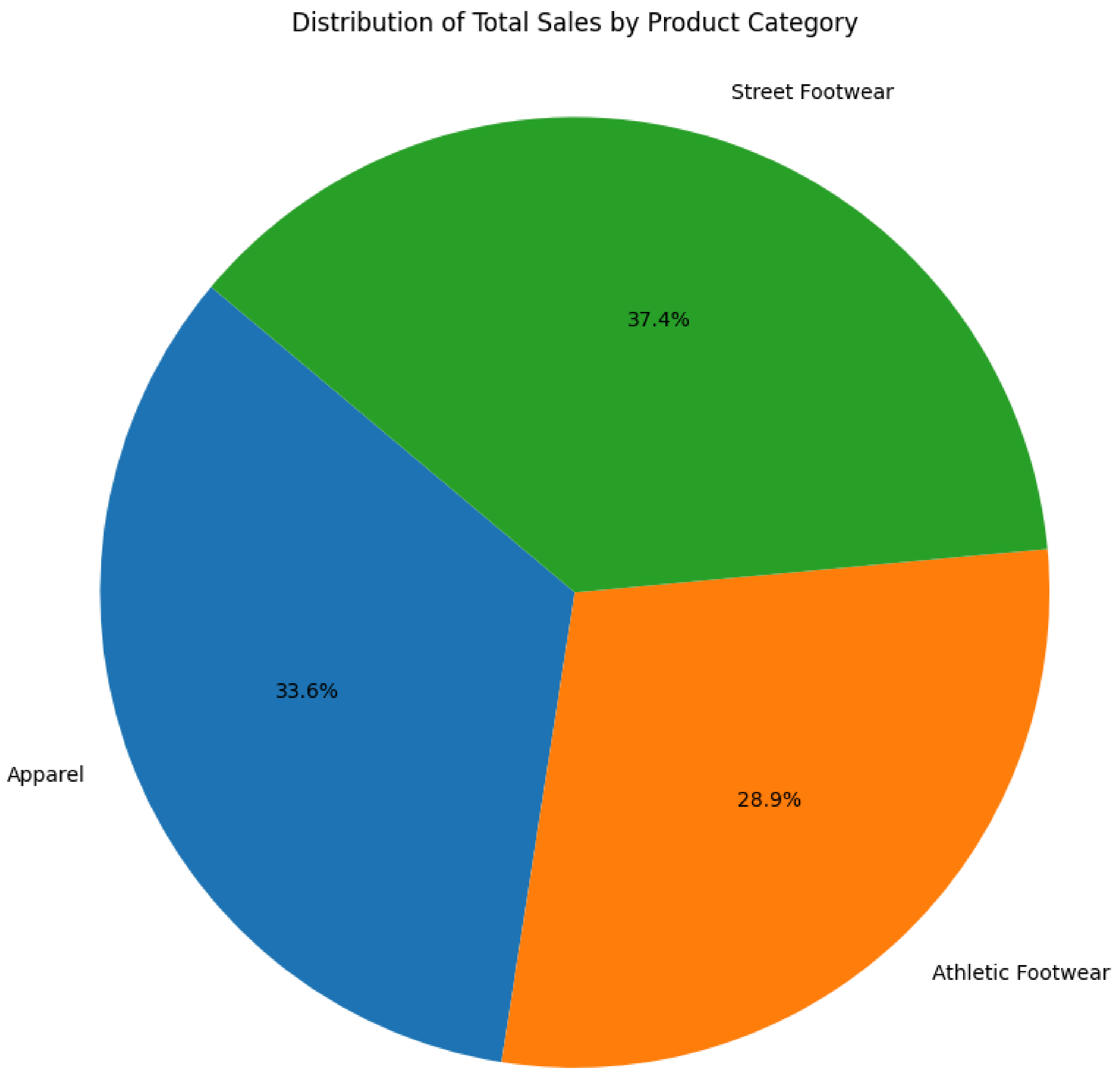
```
        Retailer ID  Price per Unit   Units Sold    Total Sales  \
count  9.648000e+03     9648.000000  9648.000000    9648.000000
mean   1.173850e+06       45.216625   256.930037   93273.437500
```

```
     std    2.636038e+04       14.705397   214.252030  141916.016727
     min    1.128299e+06        7.000000     0.000000       0.000000
     25%    1.185732e+06       35.000000   106.000000    4254.500000
     50%    1.185732e+06       45.000000   176.000000    9576.000000
     75%    1.185732e+06       55.000000   350.000000  150000.000000
     max    1.197831e+06      110.000000  1275.000000  825000.000000

            Operating Profit  Operating Margin
     count       9648.000000       9648.000000
     mean       34425.244761          0.422991
     std        54193.113713          0.097197
     min            0.000000          0.100000
     25%         1921.752500          0.350000
     50%         4371.420000          0.410000
     75%        52062.500000          0.490000
     max       390000.000000          0.800000
```

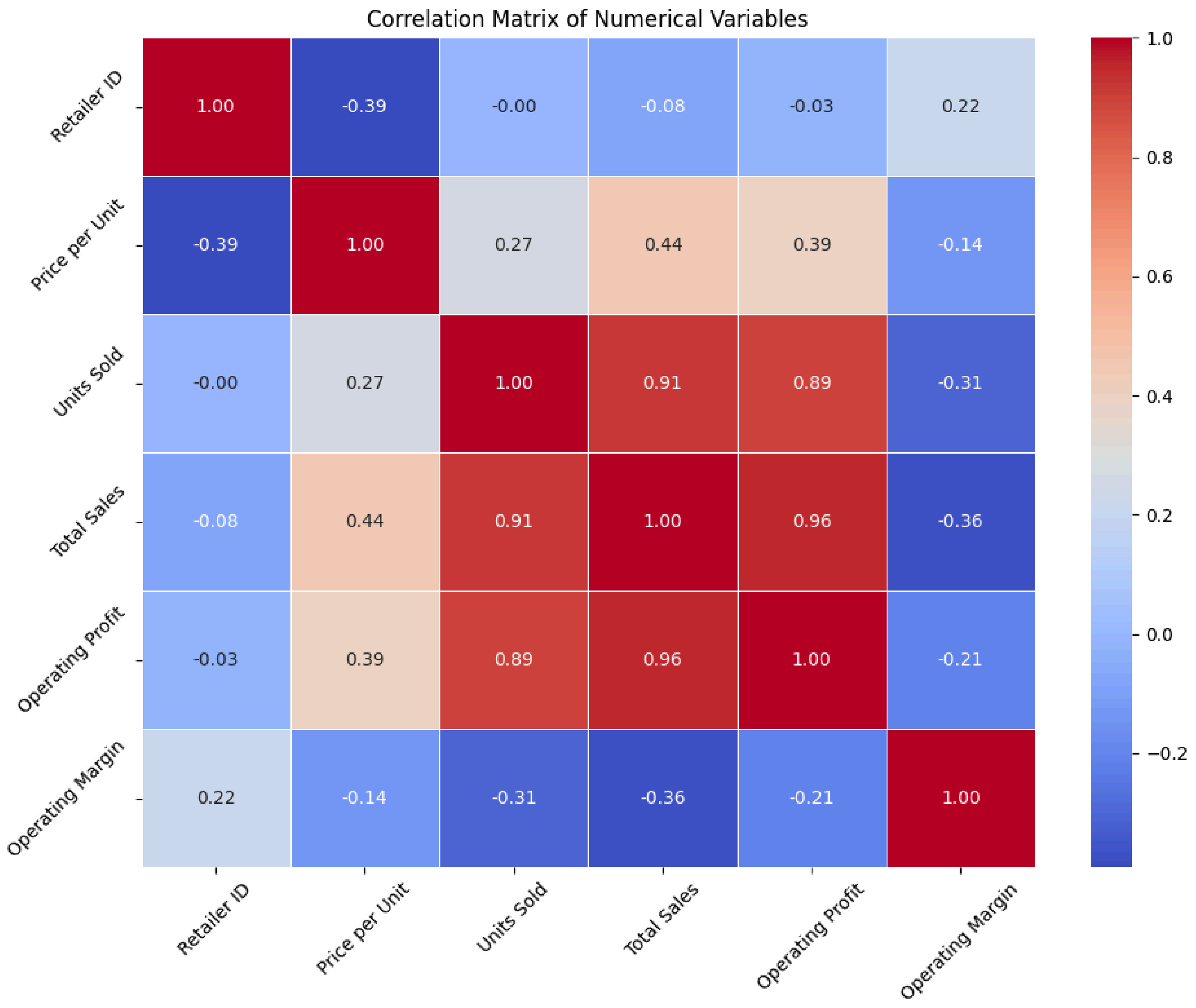
```
# Group data by retailer and sum total sales
total_sales_by_retailer = df.groupby('Retailer')['Total Sales'].sum()

# Sort the data by total sales
total_sales_by_retailer = total_sales_by_retailer.sort_values(ascending=False)

# Create a bar chart
plt.figure(figsize=(10, 6))
total_sales_by_retailer.plot(kind='bar', color='skyblue')
plt.title('Total Sales by Retailer')
plt.xlabel('Retailer')
plt.ylabel('Total Sales ($)')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()
```



```
# Group data by product category and sum total sales
total_sales_by_category = df.groupby('Product Category')['Total Sales'].sum()

# Create a pie chart
plt.figure(figsize=(8, 8))
plt.pie(total_sales_by_category, labels=total_sales_by_category.index, autopct='%1.1f%%', s
plt.title('Distribution of Total Sales by Product Category')
plt.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle.
plt.tight_layout()
plt.show()
```

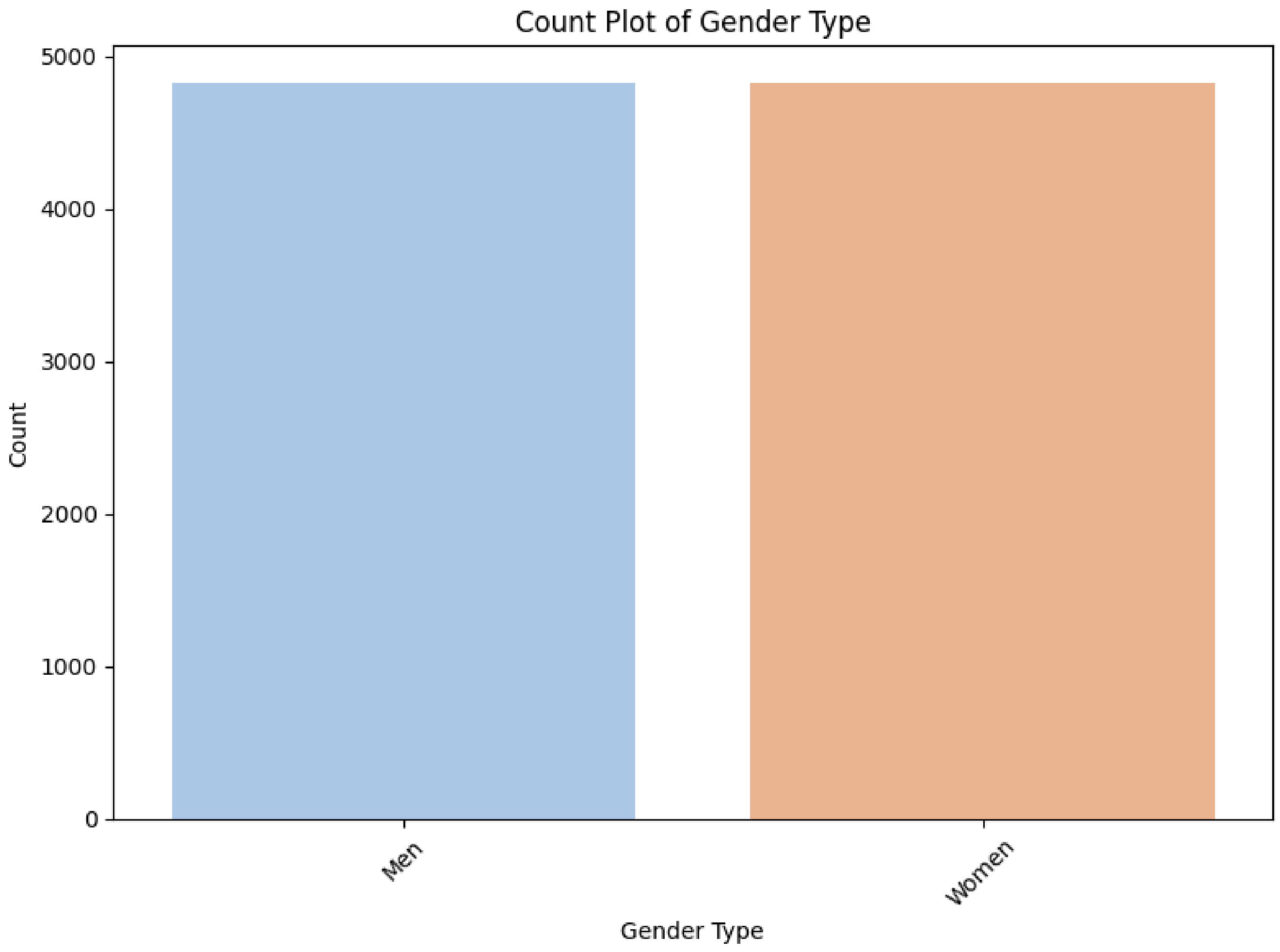

Distribution of Total Sales by Product Category
Street Footwear
37.4%
33.6%
Apparel
28.9%
Athletic Footwear

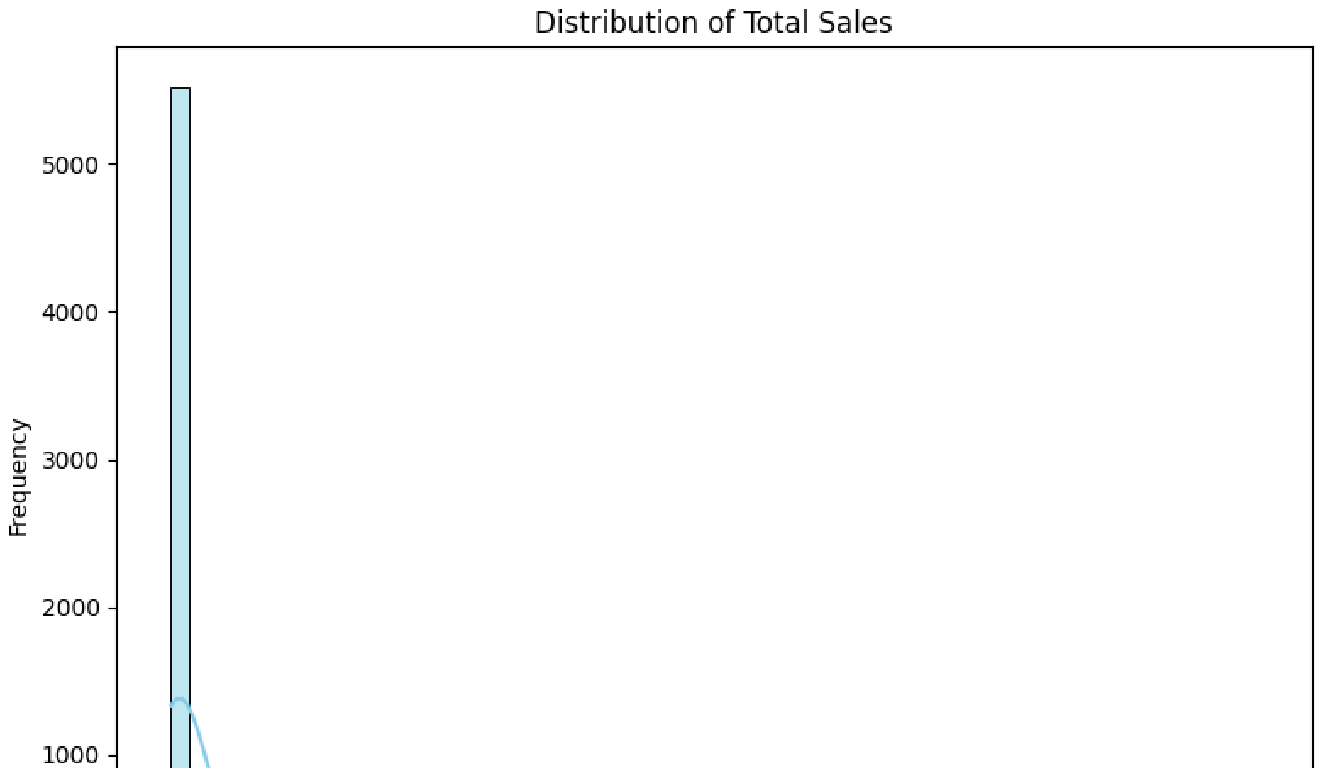
```
# Selecting only numerical columns
numerical_df = df.select_dtypes(include=['int64', 'float64'])

# Compute the correlation matrix
correlation_matrix = numerical_df.corr()

# Create a heatmap
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f", linewidths=0.5)
plt.title('Correlation Matrix of Numerical Variables')
plt.xticks(rotation=45)
plt.yticks(rotation=45)
plt.tight_layout()
plt.show()
```

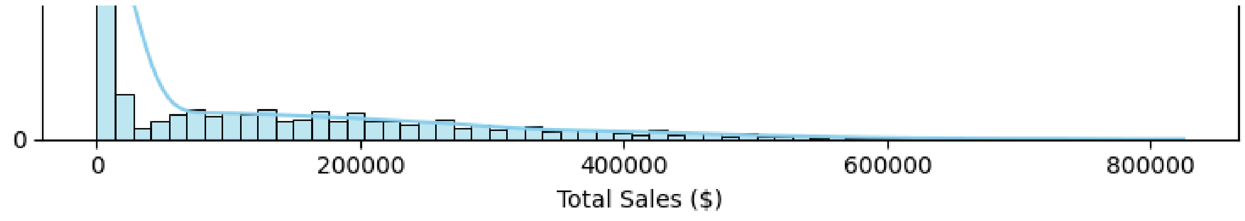

Correlation Matrix of Numerical Variables
Retailer ID
Price per Unit
Units Sold
Total Sales
Operating Profit
Operating Margin
1.00 -0.39 -0.00 -0.08 -0.03 0.22
-0.39 1.00 0.27 0.44 0.39 -0.14
-0.00 0.27 1.00 0.91 0.89 -0.31
-0.08 0.44 0.91 1.00 0.96 -0.36
-0.03 0.39 0.89 0.96 1.00 -0.21
0.22 -0.14 -0.31 -0.36 -0.21 1.00
1.0
0.8
0.6
0.4
0.2
0.0
−0.2

```
# Count Plot for Gender Type
plt.figure(figsize=(8, 6))
sns.countplot(x='Gender Type', data=df, palette='pastel')
plt.title('Count Plot of Gender Type')
plt.xlabel('Gender Type')
plt.ylabel('Count')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()

# Distribution Plot for Total Sales
plt.figure(figsize=(8, 6))
sns.histplot(df['Total Sales'], kde=True, color='skyblue')
plt.title('Distribution of Total Sales')
plt.xlabel('Total Sales ($)')
plt.ylabel('Frequency')
plt.tight_layout()
plt.show()
```

```
<ipython-input-24-5f3e527966f1>:3: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0.

  sns.countplot(x='Gender Type', data=df, palette='pastel')
```



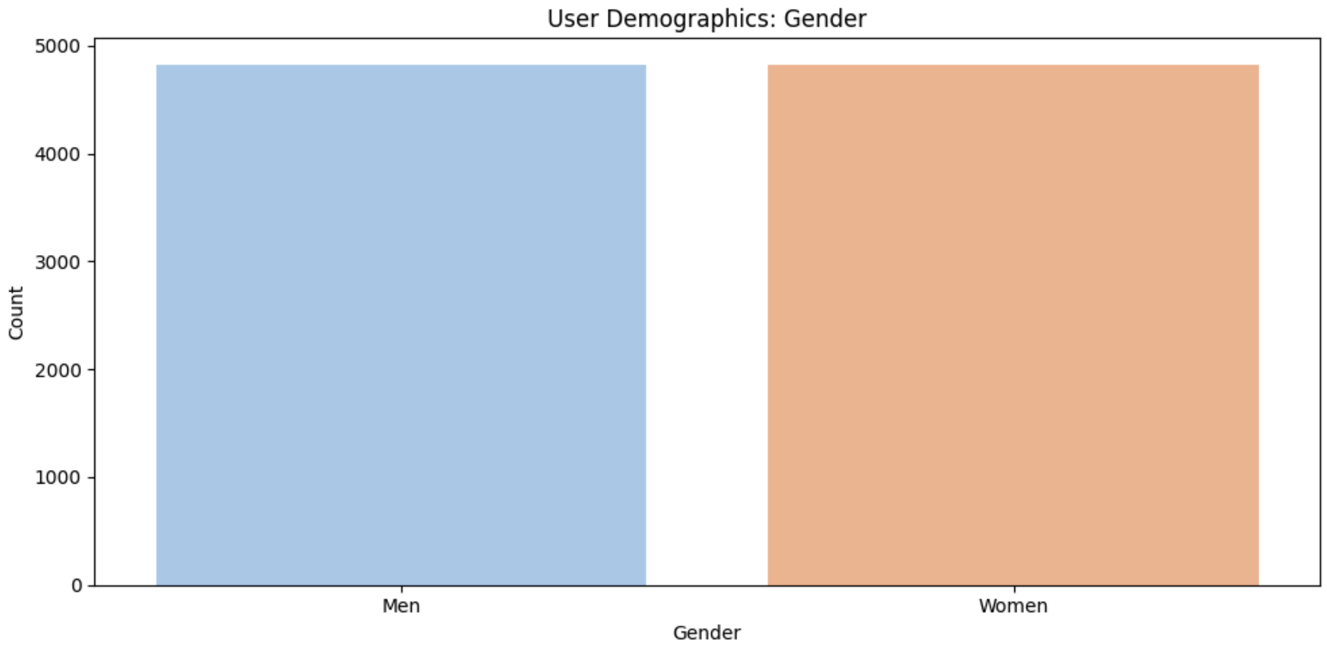




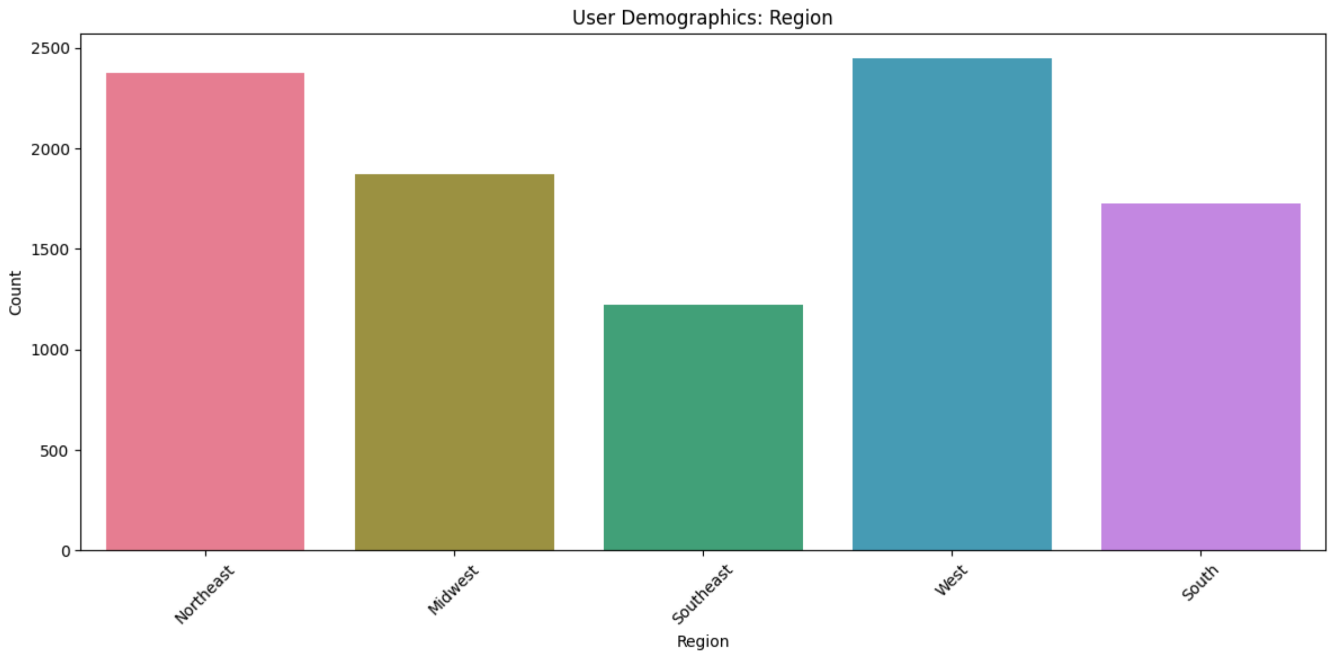
III. Key Findings

```
# User Demographics: Gender
plt.figure(figsize=(10, 5))
sns.countplot(x='Gender Type', data=df, palette='pastel')
plt.title('User Demographics: Gender')
plt.xlabel('Gender')
plt.ylabel('Count')
plt.tight_layout()
plt.show()

# User Demographics: Region
plt.figure(figsize=(12, 6))
sns.countplot(x='Region', data=df, palette='husl')
plt.title('User Demographics: Region')
plt.xlabel('Region')
plt.ylabel('Count')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()

# User Demographics: Product Category
plt.figure(figsize=(10, 6))
sns.countplot(x='Product Category', data=df, palette='Set2')
plt.title('User Demographics: Product Category')
plt.xlabel('Product Category')
plt.ylabel('Count')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

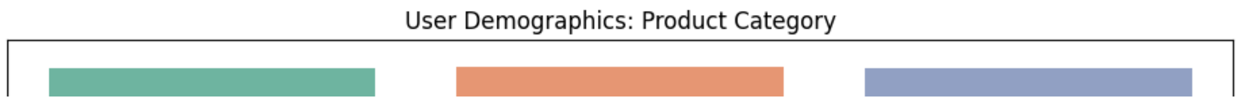
```
<ipython-input-25-263c0c2ebe0b>:3: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0.

  sns.countplot(x='Gender Type', data=df, palette='pastel')
```

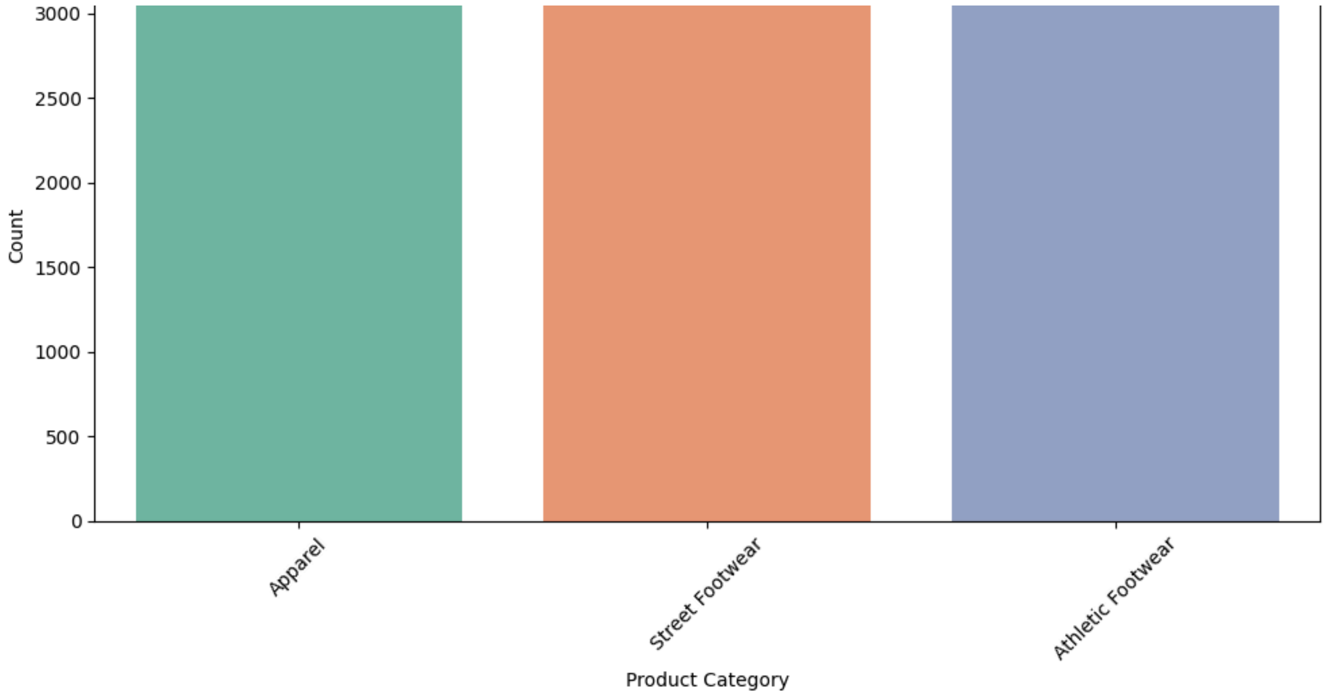


```
<ipython-input-25-263c0c2ebe0b>:12: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0.

  sns.countplot(x='Region', data=df, palette='husl')
```

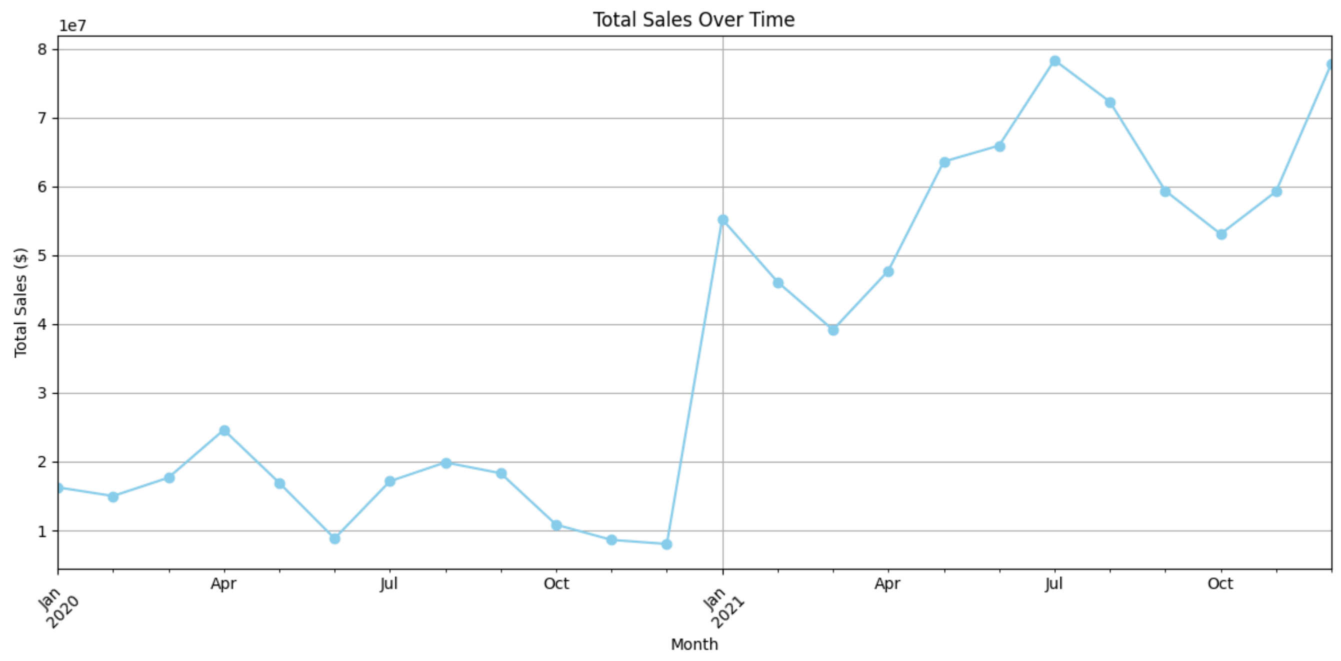


```
<ipython-input-25-263c0c2ebe0b>:22: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0.

  sns.countplot(x='Product Category', data=df, palette='Set2')
```

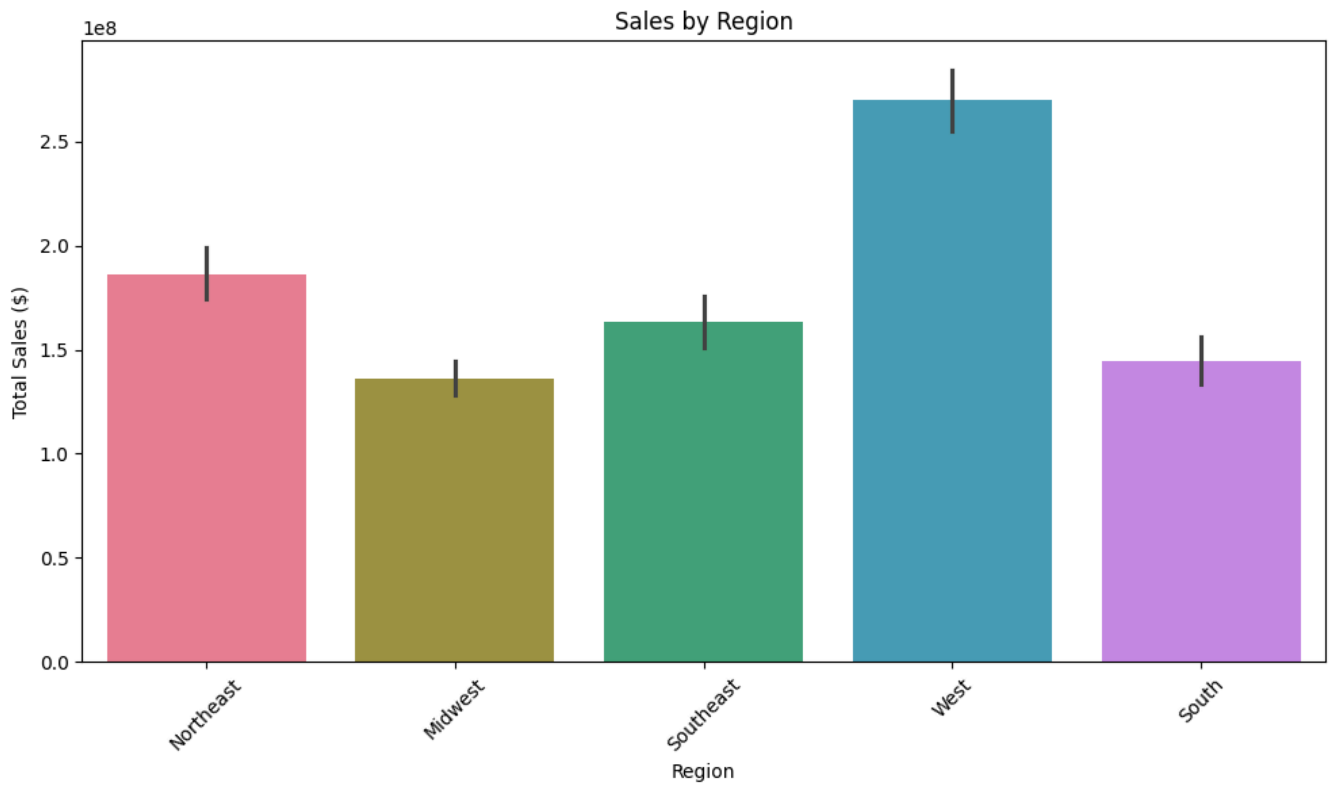



3000
2500
2000
1500
1000
500
0
Count
Apparel
Street Footwear
Athletic Footwear
Product Category

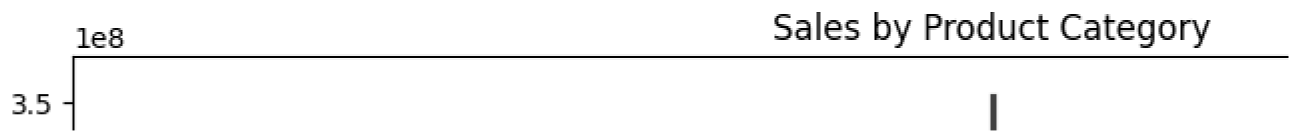
```
# Total Sales Over Time (e.g., by Invoice Date)
df['Invoice Date'] = pd.to_datetime(df['Invoice Date'])  # Convert to datetime format
df['Invoice Month'] = df['Invoice Date'].dt.to_period('M')  # Extract month and year
total_sales_over_time = df.groupby('Invoice Month')['Total Sales'].sum()

plt.figure(figsize=(12, 6))
total_sales_over_time.plot(marker='o', color='skyblue')
plt.title('Total Sales Over Time')
plt.xlabel('Month')
plt.ylabel('Total Sales ($)')
plt.xticks(rotation=45)
plt.grid(True)
plt.tight_layout()
plt.show()

# Sales by Region
plt.figure(figsize=(10, 6))
sns.barplot(x='Region', y='Total Sales', data=df, palette='husl', estimator=sum)
plt.title('Sales by Region')
plt.xlabel('Region')
plt.ylabel('Total Sales ($)')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()

# Sales by Product Category
plt.figure(figsize=(10, 6))
sns.barplot(x='Product Category', y='Total Sales', data=df, palette='Set2', estimator=sum)
plt.title('Sales by Product Category')
plt.xlabel('Product Category')
plt.ylabel('Total Sales ($)')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

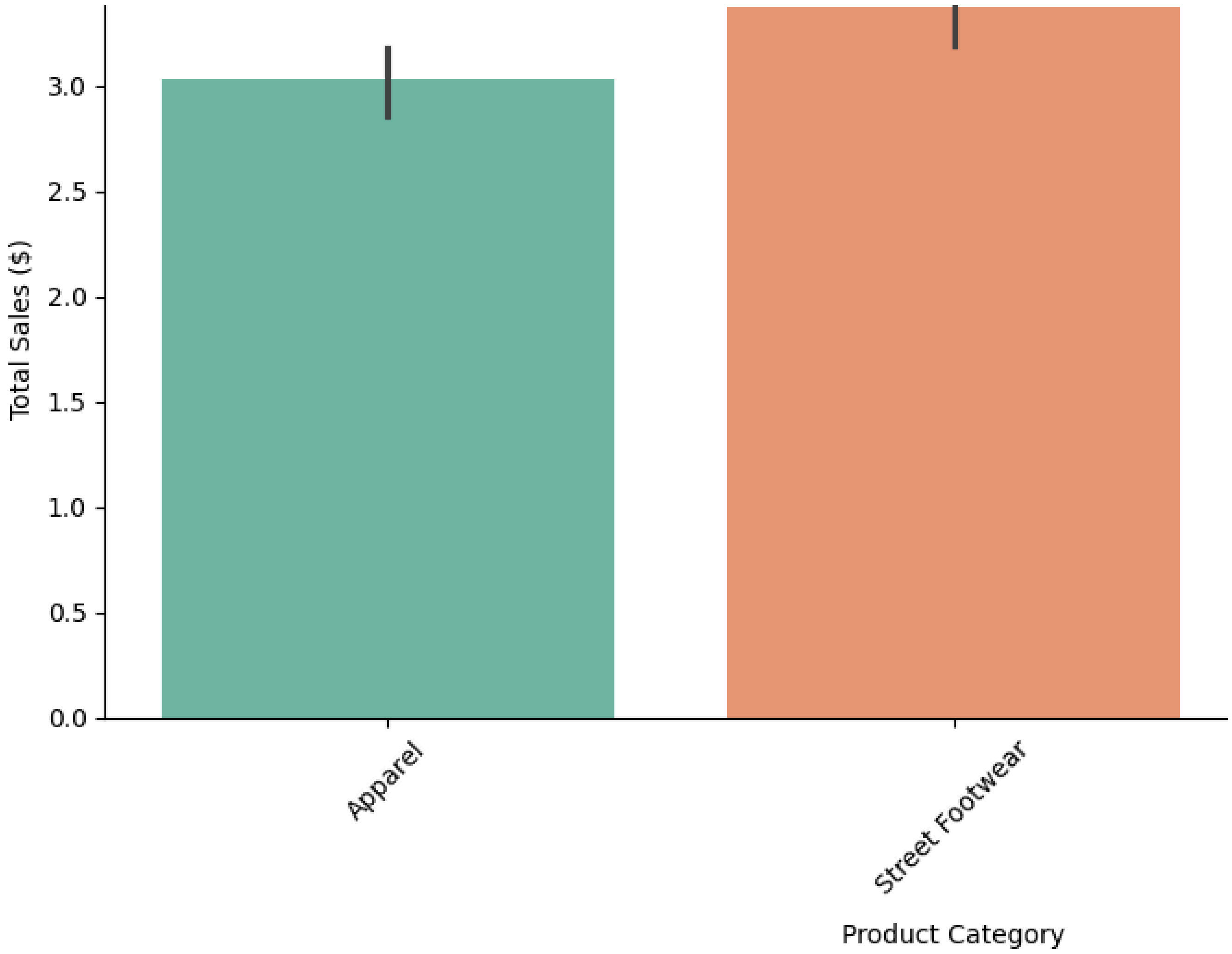


```
<ipython-input-30-f66ccb5611bc>:18: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0.

  sns.barplot(x='Region', y='Total Sales', data=df, palette='husl', estimator=sum)
```



```
<ipython-input-30-f66ccb5611bc>:28: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0.

  sns.barplot(x='Product Category', y='Total Sales', data=df, palette='Set2', estimator
```

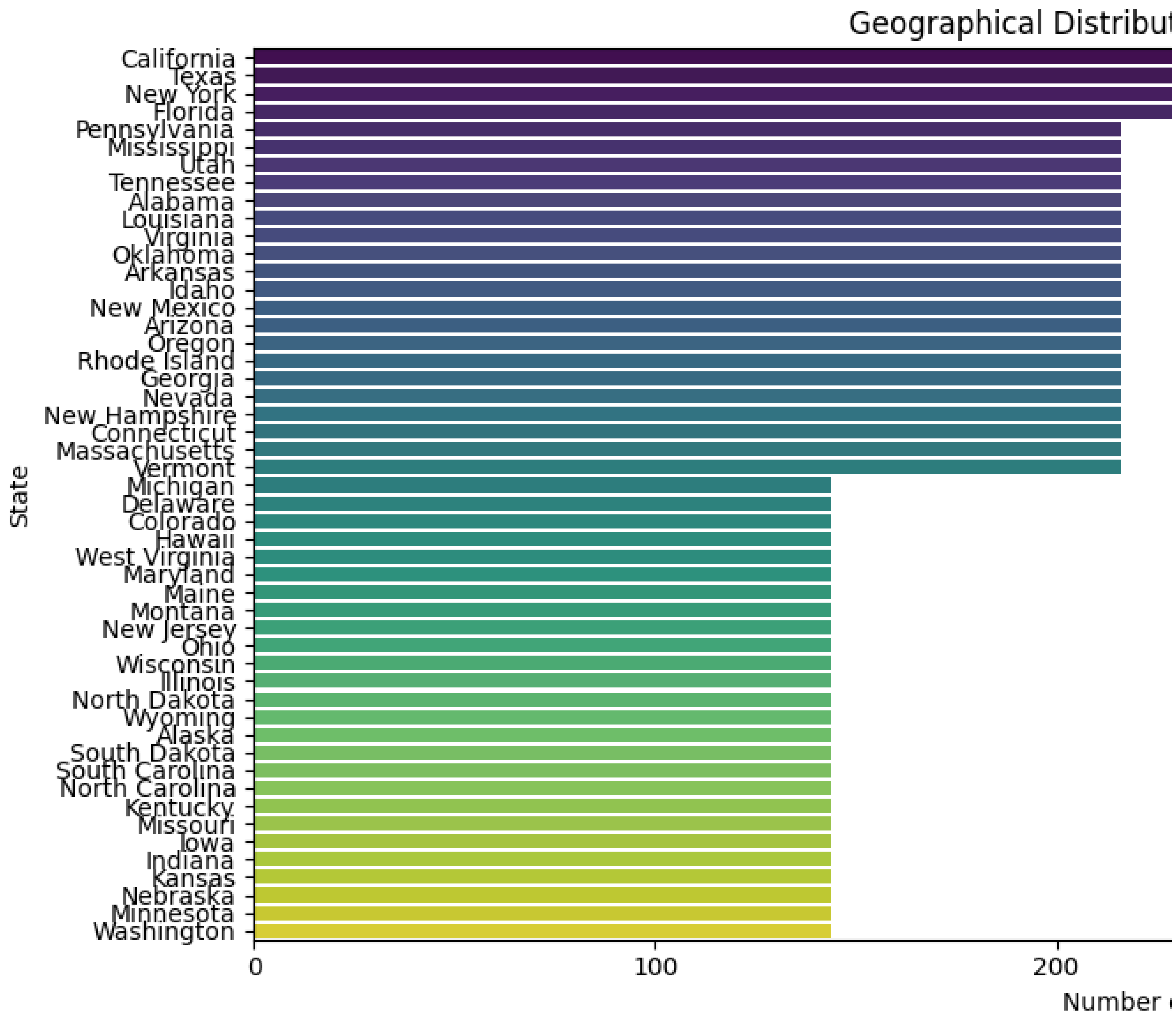




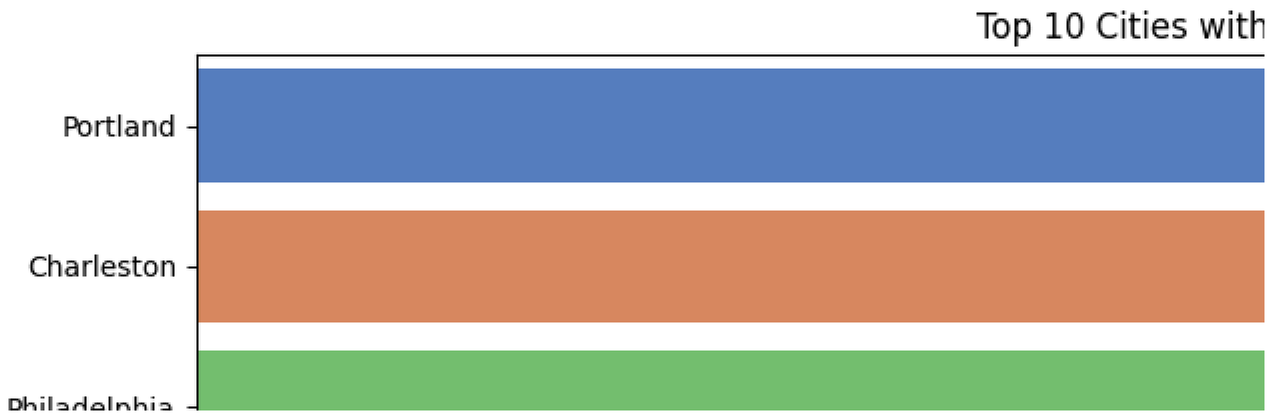
IV. Advanced Analysis

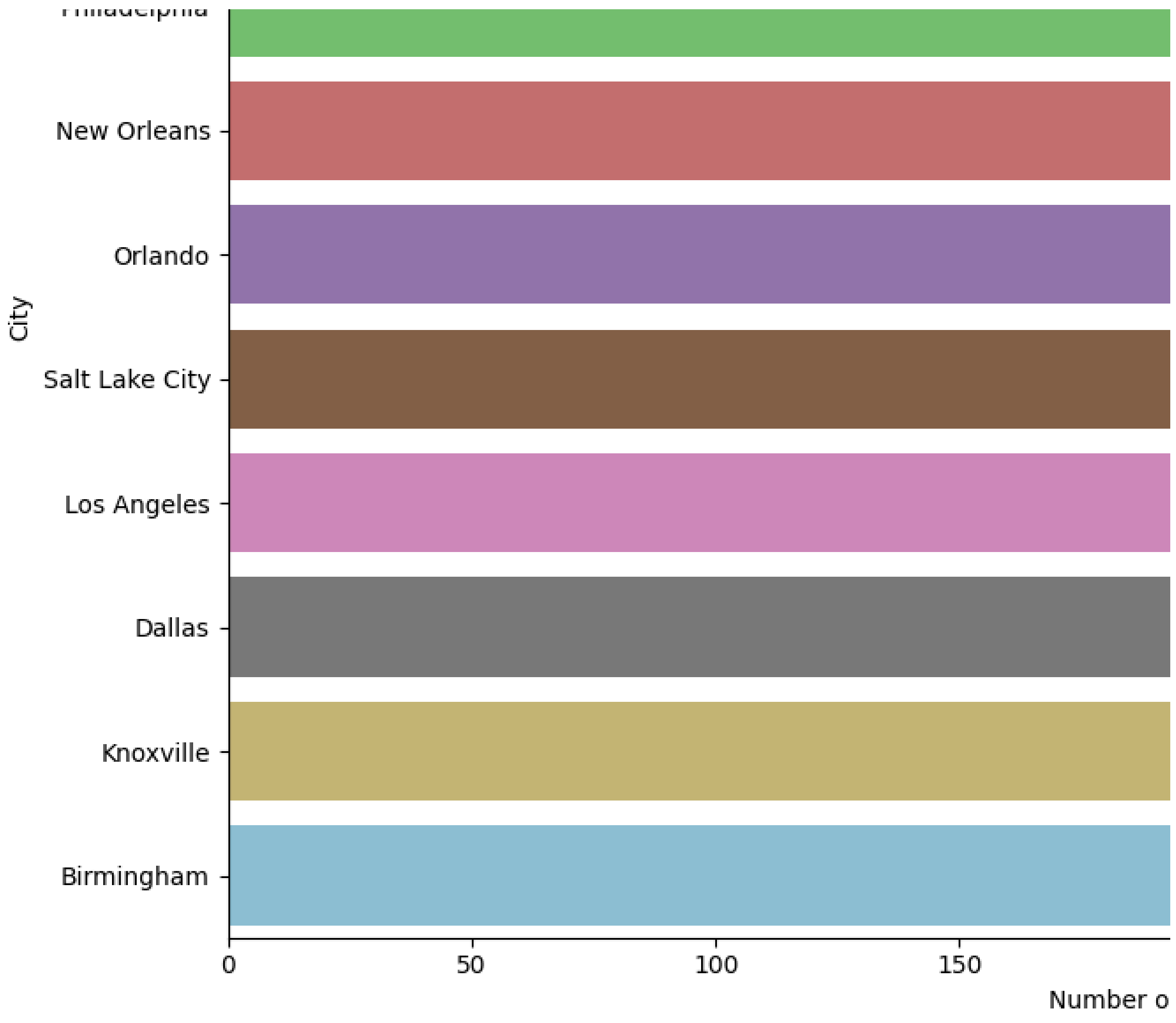
```
# Plot the geographical distribution of sales by State
plt.figure(figsize=(12, 6))
sns.countplot(y='State', data=df, palette='viridis', order=df['State'].value_counts().index
plt.title('Geographical Distribution of Sales by State')
plt.xlabel('Number of Sales')
plt.ylabel('State')
plt.tight_layout()
plt.show()

# Plot the geographical distribution of sales by City (you can adjust this based on your pr
plt.figure(figsize=(12, 8))
sns.countplot(y='City', data=df, palette='muted', order=df['City'].value_counts().iloc[:10]
plt.title('Top 10 Cities with Highest Sales')
plt.xlabel('Number of Sales')
plt.ylabel('City')
plt.tight_layout()
plt.show()
```

```
<ipython-input-31-76fc4f671de1>:3: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0.

  sns.countplot(y='State', data=df, palette='viridis', order=df['State'].value_counts()
```



```
<ipython-input-31-76fc4f671de1>:12: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0.

  sns.countplot(y='City', data=df, palette='muted', order=df['City'].value_counts().ilo
```





```
# Convert 'Invoice Date' column to datetime format
data['Invoice Date'] = pd.to_datetime(data['Invoice Date'])

# Group by 'Invoice Date' and sum total sales for each date
temporal_data = data.groupby('Invoice Date')['Total Sales'].sum().reset_index()

# Plot temporal trends
plt.figure(figsize=(10, 6))
plt.plot(temporal_data['Invoice Date'], temporal_data['Total Sales'], marker='o', linestyle
plt.title('Temporal Trends in Total Sales')
```