importing librariries and the data

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
data = pd.read_csv("/content/Global-Superstore - Global-Superstore.csv.csv")
```

Number of rows and columns

```
print(data.shape)
```

```
(35851, 24)
```

Column names in our dataset

```
print (data.columns)
```

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

```
data.head()
```

	Order Date	Ship Date	Ship Mode	Customer ID	Customer Name	Segment	City	State	...	Product ID	Category	Ca
	1/2012	7/31/2012	Same Day	RH-19495	Rick Hansen	Consumer	New York City	New York	...	TEC-AC-10003033	Technology	Acc
	5/2013	2/7/2013	Second Class	JR-16210	Justin Ritter	Corporate	Wollongong	New South Wales	...	FUR-CH-10003950	Furniture	
	7/2013	10/18/2013	First Class	CR-12730	Craig Reiter	Consumer	Brisbane	Queensland	...	TEC-PH-10004664	Technology	
	8/2013	1/30/2013	First Class	KM-16375	Katherine Murray	Home Office	Berlin	Berlin	...	TEC-PH-10004583	Technology	
	5/2013	11/6/2013	Same Day	RH-9495	Rick Hansen	Consumer	Dakar	Dakar	...	TEC-SHA-10000501	Technology	

we can also check weather their is a null value or not by looking at the type if there is null value if it is 'object' it has NULL value

```
Start coding or generate with AI.
```

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 35851 entries, 0 to 35850
Data columns (total 24 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Row ID          35851 non-null  int64
 1   Order ID        35851 non-null  object
 2   Order Date      35851 non-null  object
```

```
 3   Ship Date       35851 non-null  object
 4   Ship Mode       35851 non-null  object
 5   Customer ID     35851 non-null  object
 6   Customer Name   35851 non-null  object
 7   Segment         35851 non-null  object
 8   City            35851 non-null  object
 9   State           35851 non-null  object
 10  Country         35850 non-null  object
 11  Postal Code     5835 non-null   float64
 12  Market          35850 non-null  object
 13  Region          35850 non-null  object
 14  Product ID      35850 non-null  object
 15  Category        35850 non-null  object
 16  Sub-Category    35850 non-null  object
 17  Product Name    35850 non-null  object
 18  Sales           35850 non-null  float64
 19  Quantity        35850 non-null  float64
 20  Discount        35850 non-null  float64
 21  Profit          35850 non-null  float64
 22  Shipping Cost   35850 non-null  float64
 23  Order Priority  35850 non-null  object
dtypes: float64(6), int64(1), object(17)
memory usage: 6.6+ MB
```

check the number of Null values

```
np.sum(data.isna())
```

```
/usr/local/lib/python3.10/dist-packages/numpy/core/fromnumeric.py:86: FutureWarning: The behavior of DataFrame.sum with axis=None i
  return reduction(axis=axis, out=out, **passkwargs)
```

	0
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	1
Postal Code	30016
Market	1
Region	1
Product ID	1
Category	1
Sub-Category	1
Product Name	1
Sales	1
Quantity	1
Discount	1
Profit	1
Shipping Cost	1
Order Priority	1

dtype: int64

we are checking the details of the whole dataset

Double-click (or enter) to edit

```
data.describe()
```

	Row ID	Postal Code	Sales	Quantity	Discount	Profit	Shipping Cost
count	35851.000000	5835.000000	35850.000000	35850.000000	35850.000000	35850.000000	35850.000000
mean	24393.234024	54885.910026	340.291356	3.887755	0.120159	40.422117	37.054337
std	14394.906691	32448.849538	555.723623	2.378014	0.186597	206.506861	65.709325
min	2.000000	1040.000000	7.968000	1.000000	0.000000	-6599.978000	3.340000
25%	12397.500000	22204.000000	73.080000	2.000000	0.000000	1.680000	7.060000
50%	23589.000000	54703.000000	154.550000	3.000000	0.000000	19.440000	15.040000
75%	36330.000000	90008.000000	372.960000	5.000000	0.200000	60.115000	37.460000
max	51289.000000	99301.000000	22638.480000	14.000000	0.850000	8399.976000	933.570000

we are identifying unique,top,freq values in the dataset for each column

```
data.describe(include=["object", "bool"])
```

	Order ID	Order Date	Ship Date	Ship Mode	Customer ID	Customer Name	Segment	City	State	Country	Market	Region	Product ID	Cate
count	35851	35851	35851	35851	35851	35851	35851	35851	35851	35850	35850	35850	35850	3
unique	20445	1417	1464	4	1586	795	3	3443	1060	147	7	13	9205	
top	IN-2013-42311	11/18/2014	11/22/2014	Standard Class	BE-11335	Steven Ward	Consumer	New York City	California	United States	APAC	Central	FUR-CH-10003354	Sup
freq	13	94	97	20159	71	80	18548	572	1236	5835	8647	8048	28	1

checking for the duplicate values

```
data.loc[data.duplicated()]
```

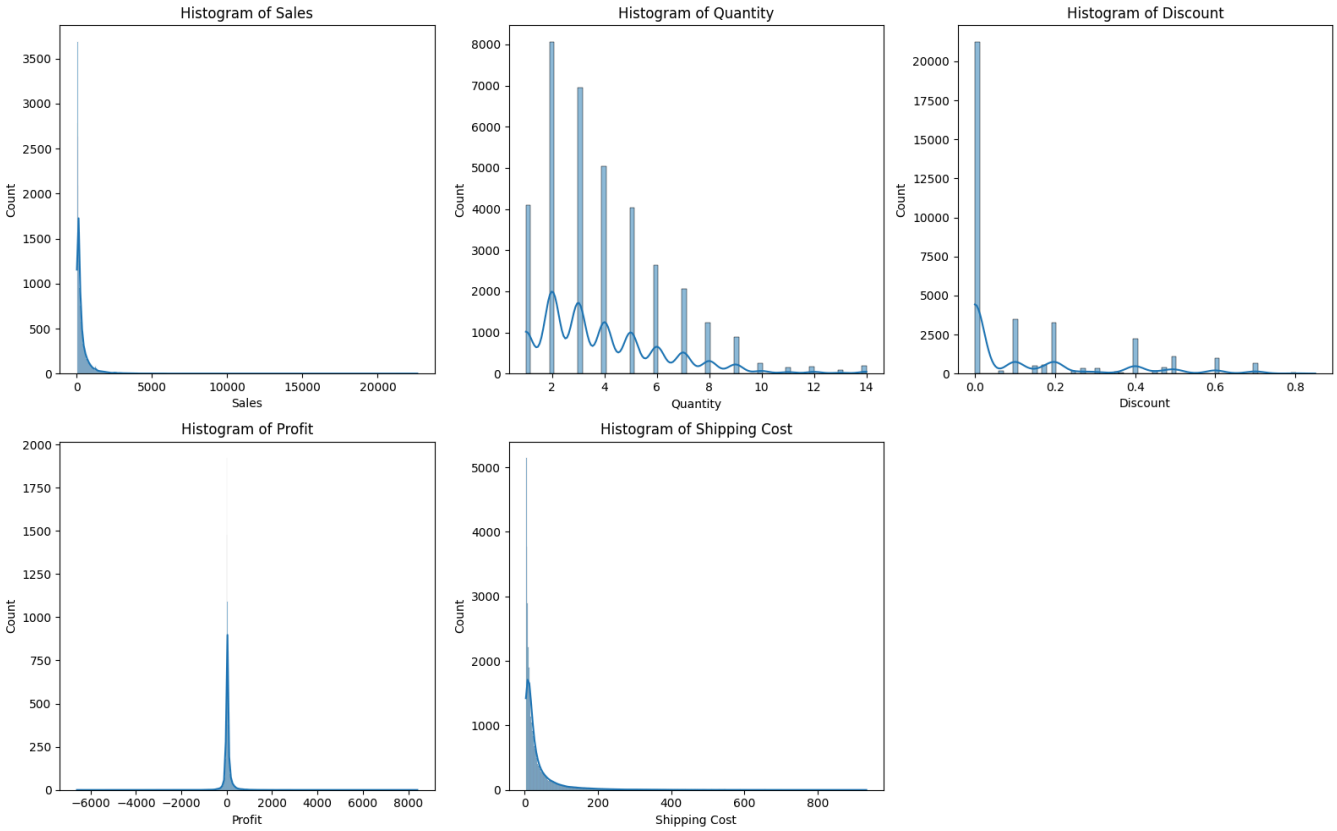
Row ID	Order ID	Order Date	Ship Date	Ship Mode	Customer ID	Customer Name	Segment	City	State	...	Product ID	Category	Sub-Category	Product Name	Sales	Quantity

0 rows × 24 columns

feature Relationships

Histogram for Numeric Features

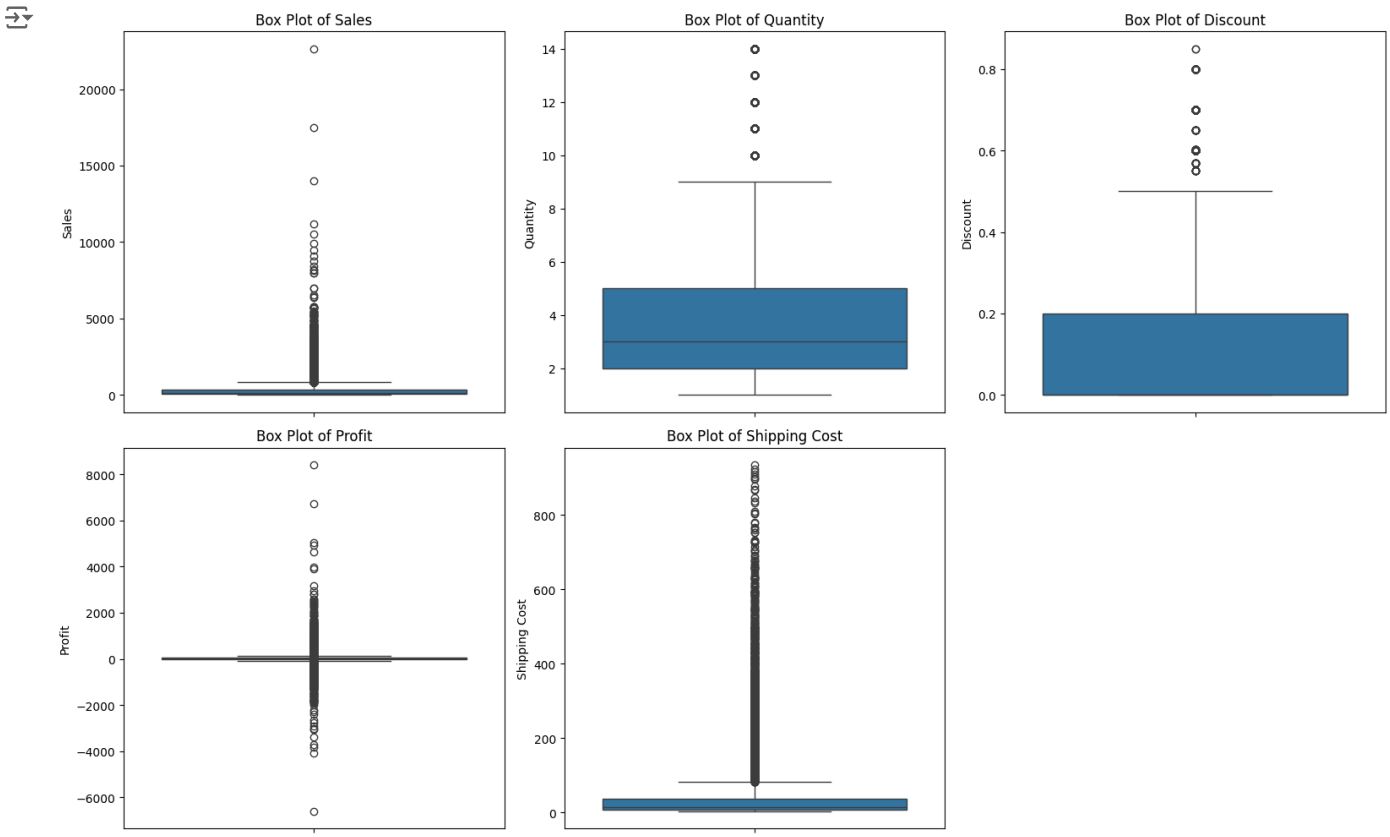
```
# Convert date columns to datetime format
data['Order Date'] = pd.to_datetime(data['Order Date'], format='%m/%d/%Y')
data['Ship Date'] = pd.to_datetime(data['Ship Date'], format='%m/%d/%Y')

# Set up matplotlib figure
plt.figure(figsize=(16, 10))

# Histograms for numeric features
numeric_features = ['Sales', 'Quantity', 'Discount', 'Profit', 'Shipping Cost']

for i, feature in enumerate(numeric_features, 1):
    plt.subplot(2, 3, i)
    sns.histplot(data[feature], kde=True
    plt.title(f'Histogram of {feature}')

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



This set of histograms displays the distribution of Sales, Quantity, profit,cost and Discount. The Histogram of Sales shows a highly skewed distribution with most sales values being low. The Quantity histogram shows a decreasing trend with most orders having low quantities. The Discount histogram indicates that most transactions have a low or zero discount, with a few instances of higher discounts.

Box Plots for Numeric Features

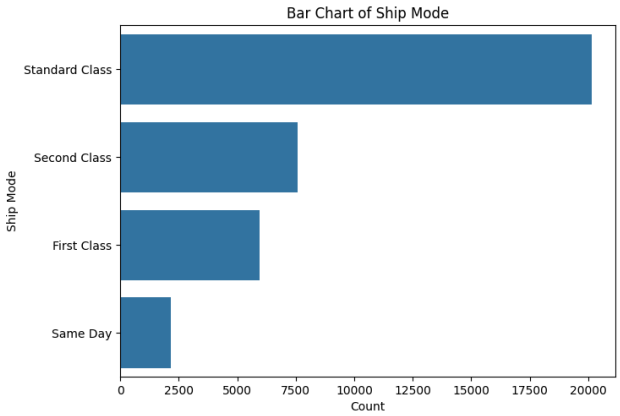
```
plt.figure(figsize=(16, 10))

# Box plots for numeric features
for i, feature in enumerate(numeric_features, 1):
    plt.subplot(2, 3, i)
    sns.boxplot(data[feature])
    plt.title(f'Box Plot of {feature}')

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

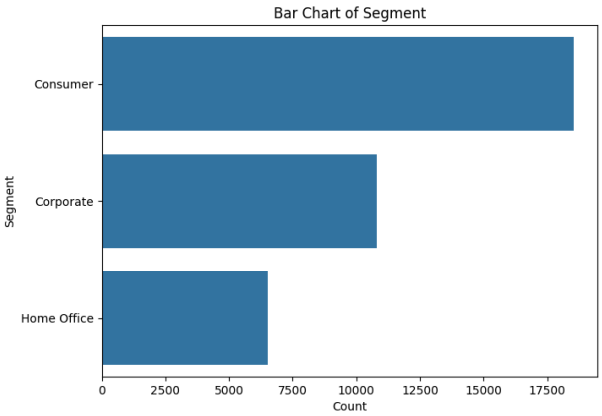


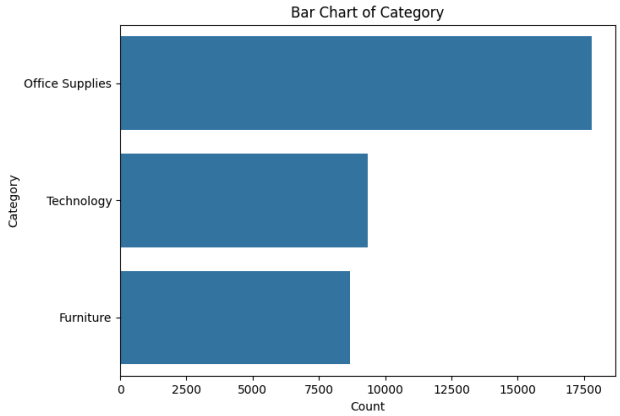
This collection of box plots represents the distribution of five metrics: Sales, Quantity, Discount, Profit, and Shipping Cost. The box plot for Sales indicates a highly skewed distribution with many outliers on the higher end. The Quantity box plot shows a more symmetric distribution with several outliers above the upper whisker. The Discount box plot highlights a concentration of values at the lower end with a few higher outliers. The Profit box plot reveals a skewed distribution with numerous outliers on the positive side. Lastly, the Shipping Cost box plot shows a significant number of high outliers, indicating variability in shipping costs.

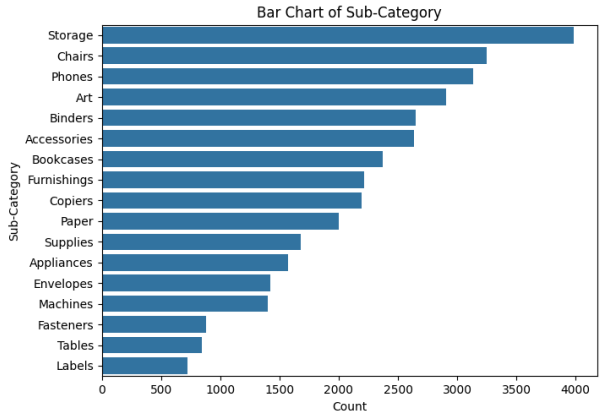
Bar Charts for categorical Features

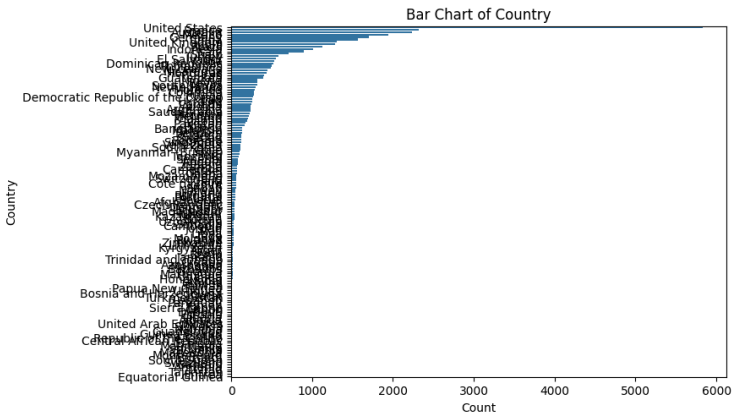
```
plt.figure(figsize=(16, 20))

# Categorical features
categorical_features = ['Ship Mode', 'Segment', 'Category', 'Sub-Category',
                        'Country', 'Market', 'Region', 'Order Priority']

for i, feature in enumerate(categorical_features, 1):
    plt.subplot(4, 2, i)
    sns.countplot(y=data[feature], order=data[feature].value_counts().index)
    plt.title(f'Bar Chart of {feature}')
    plt.xlabel('Count')

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

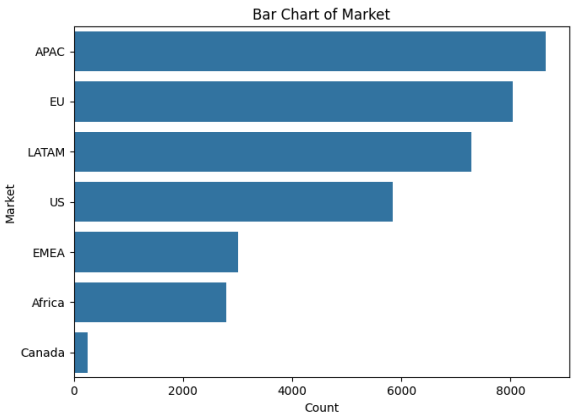


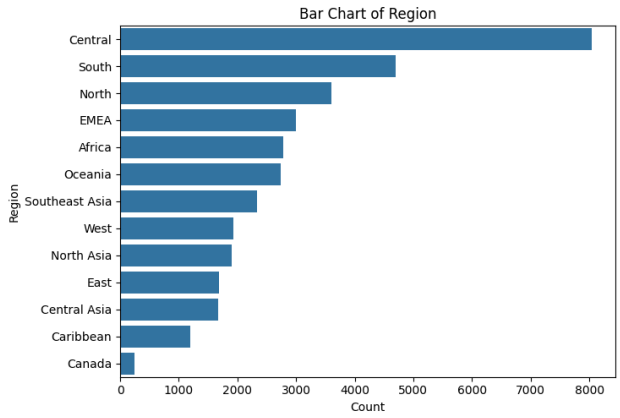


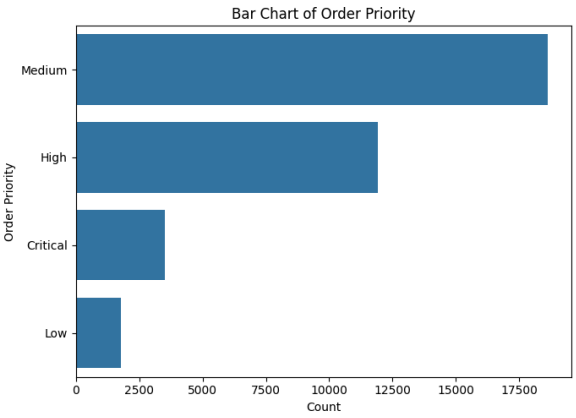






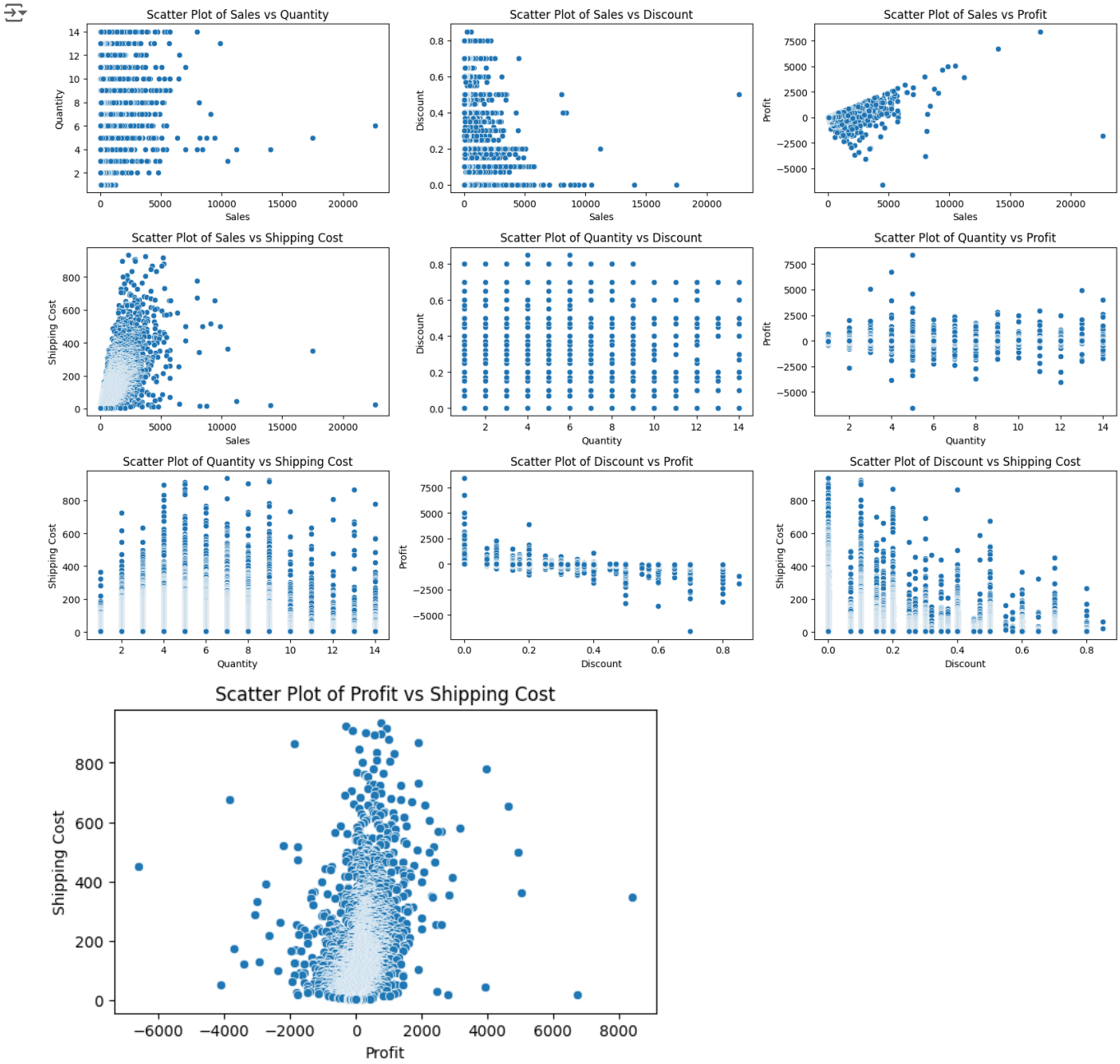






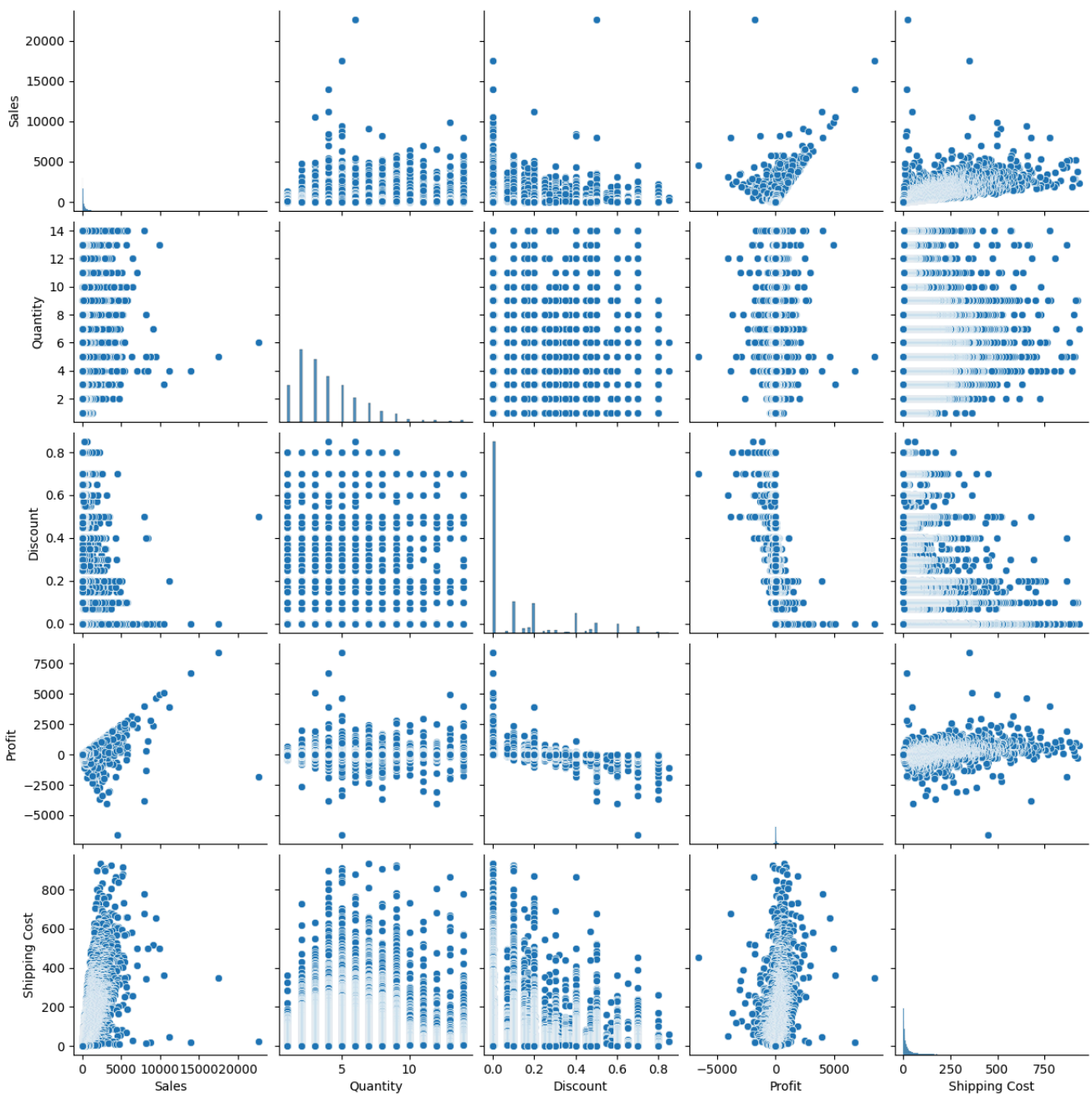
scatter plot

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

# Pairs of numeric features
pairs = [(numeric_features[i], numeric_features[j]) for i in range(len(numeric_features)) for j in range(i + 1, len(numeric_features))]

# Number of pairs
num_pairs = len(pairs)

# Define the number of plots per figure
plots_per_figure = 9

# Number of figures needed
num_figures = (num_pairs // plots_per_figure) + 1

for fig_num in range(num_figures):
    plt.figure(figsize=(16, 10))

    # Create subplots for each figure
    for i in range(plots_per_figure):
        index = fig_num * plots_per_figure + i
        if index >= num_pairs:
            break
        x, y = pairs[index]
        plt.subplot(3, 3, i + 1)
        sns.scatterplot(x=data[x], y=data[y])
        plt.title(f'Scatter Plot of {x} vs {y}')

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



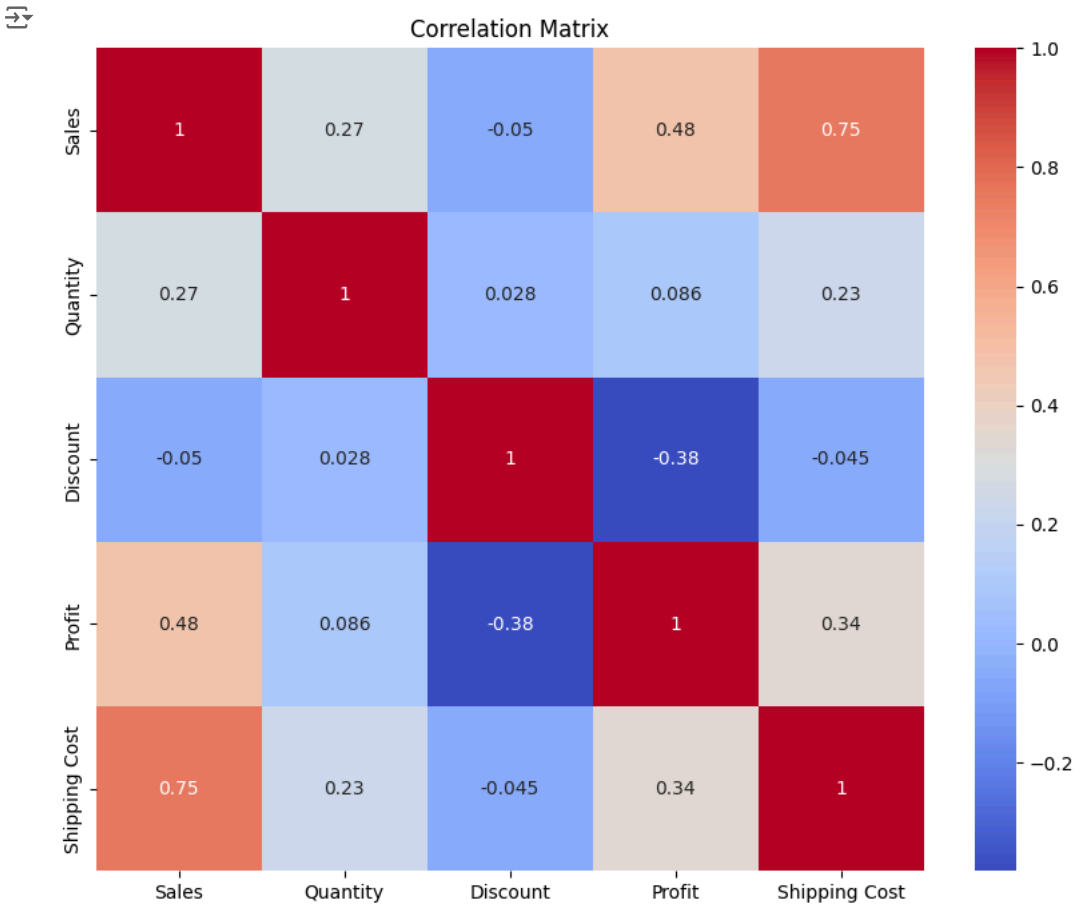
Pair Plots for Relationships Among Multiple Features

```
sns.pairplot(data[numeric_features])
plt.show()
```



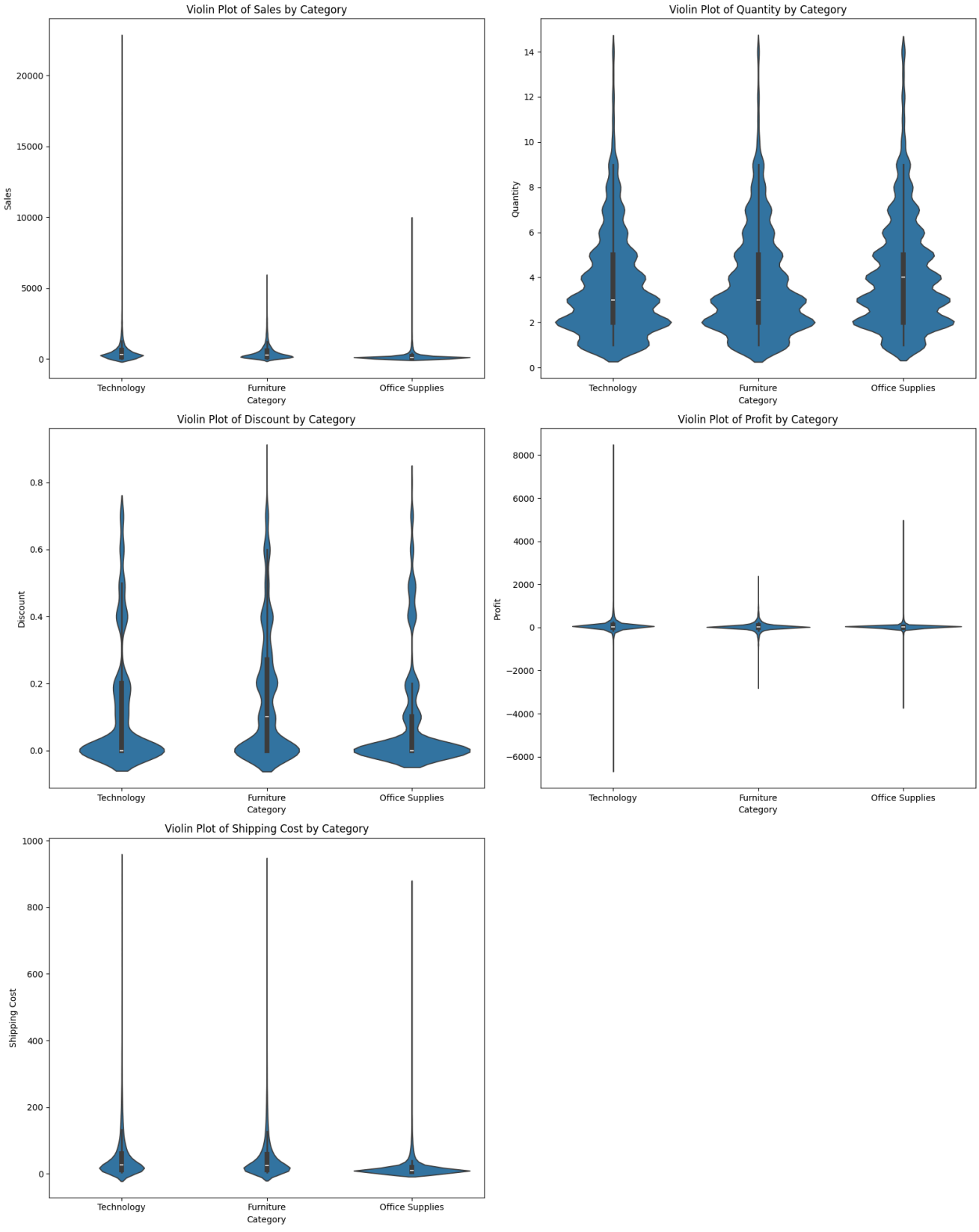
Heatmap for Correlation Matrix

```
plt.figure(figsize=(10, 8))
correlation_matrix = data[numeric_features].corr()
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')
plt.title('Correlation Matrix')
plt.show()
```



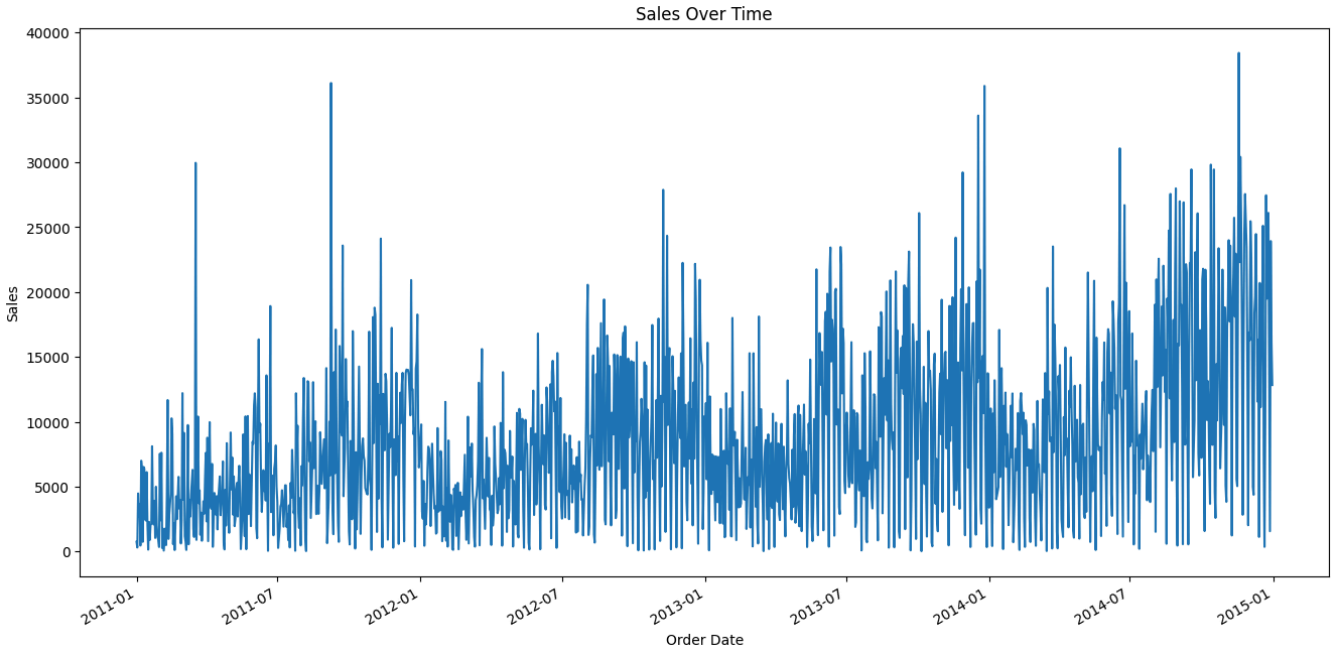
Violin Plots for Numeric Feature Distributions Across Categorical Variables

```
plt.figure(figsize=(16, 20))

# Violin plots for numeric features across 'Category'
for i, feature in enumerate(numeric_features, 1):
    plt.subplot(3, 2, i)
    sns.violinplot(x='Category', y=feature, data=data)
    plt.title(f'Violin Plot of {feature} by Category')

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



Line Charts for Time Series Data (Sales over Time)

```
plt.figure(figsize=(16, 8))
data.groupby('Order Date')['Sales'].sum().plot()
plt.title('Sales Over Time')
plt.xlabel('Order Date')
plt.ylabel('Sales')
plt.show()
```



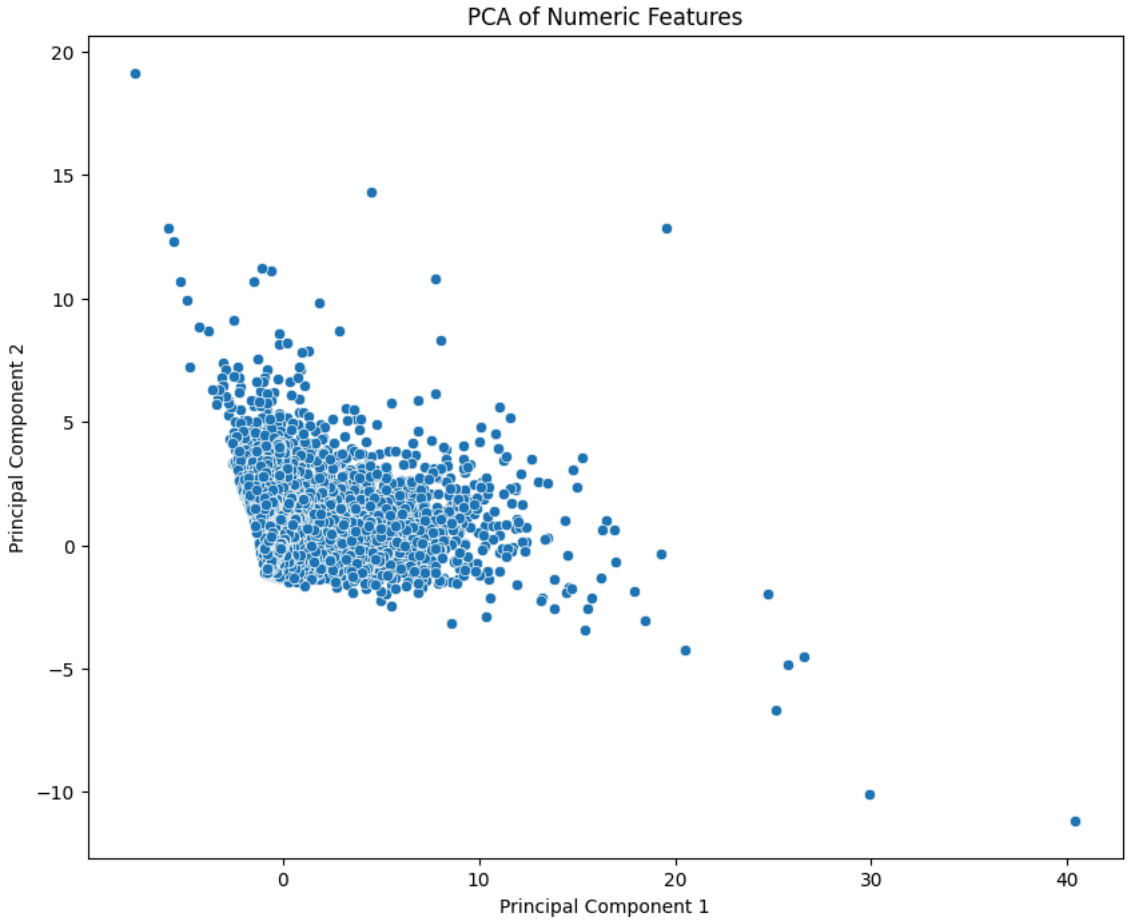
Pairwise Correlation Matrix

```
correlation_matrix = data[numeric_features].corr()
print(correlation_matrix)
```

```
                Sales  Quantity  Discount    Profit  Shipping Cost
Sales        1.000000  0.274477 -0.050004  0.477414       0.749432
Quantity     0.274477  1.000000  0.027579  0.085909       0.231797
Discount    -0.050004  0.027579  1.000000 -0.381730      -0.044952
Profit       0.477414  0.085909 -0.381730  1.000000       0.342382
Shipping Cost  0.749432  0.231797 -0.044952  0.342382       1.000000
```

PCA to Visualize Main Components

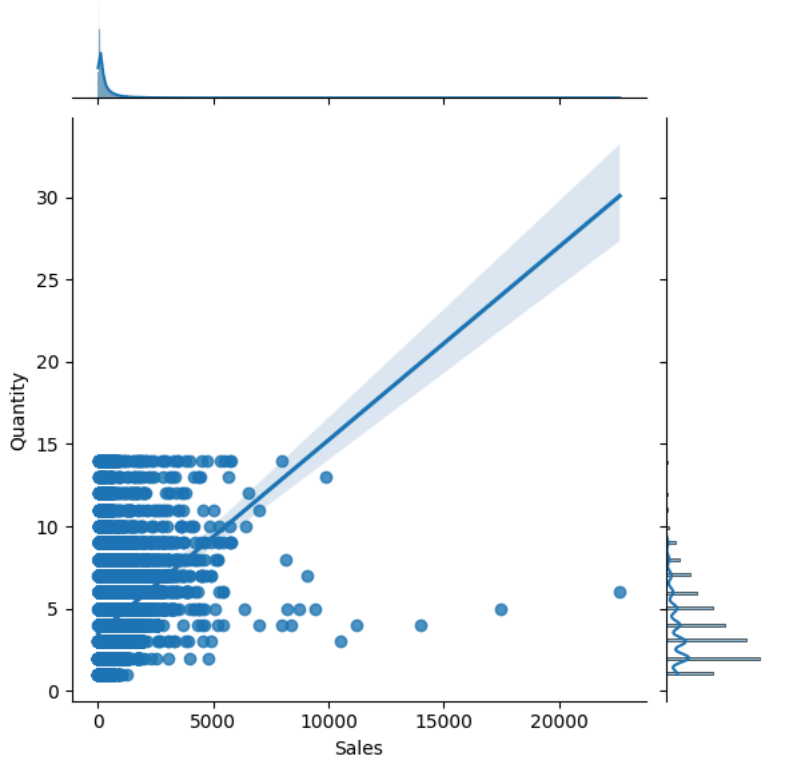
```
from sklearn.decomposition import PCA
import numpy as np

# Standardize the data
numeric_data = data[numeric_features].dropna()
numeric_data_standardized = (numeric_data - numeric_data.mean()) / numeric_data.std()

# Apply PCA
pca = PCA(n_components=2)
principal_components = pca.fit_transform(numeric_data_standardized)

# Create a DataFrame with principal components
pca_df = pd.DataFrame(data=principal_components, columns=['Principal Component 1', 'Principal Component 2'])

# Plot the PCA results
plt.figure(figsize=(10, 8))
sns.scatterplot(x='Principal Component 1', y='Principal Component 2', data=pca_df)
plt.title('PCA of Numeric Features')
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.show()
```

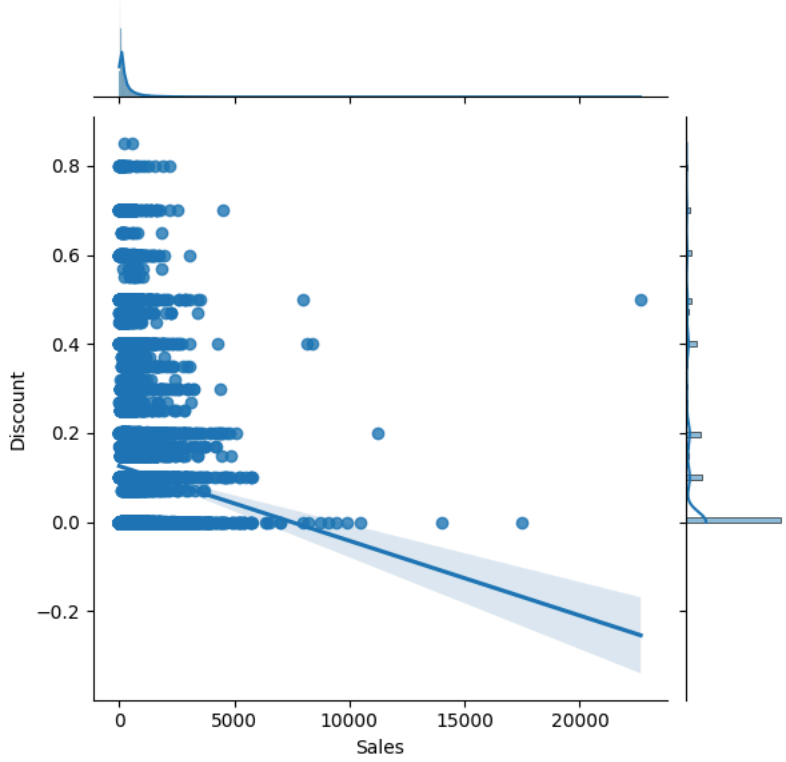


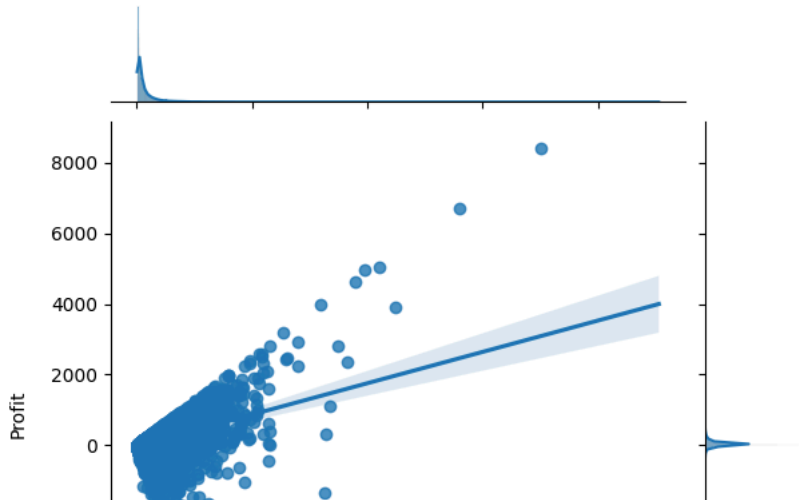
feature understanding

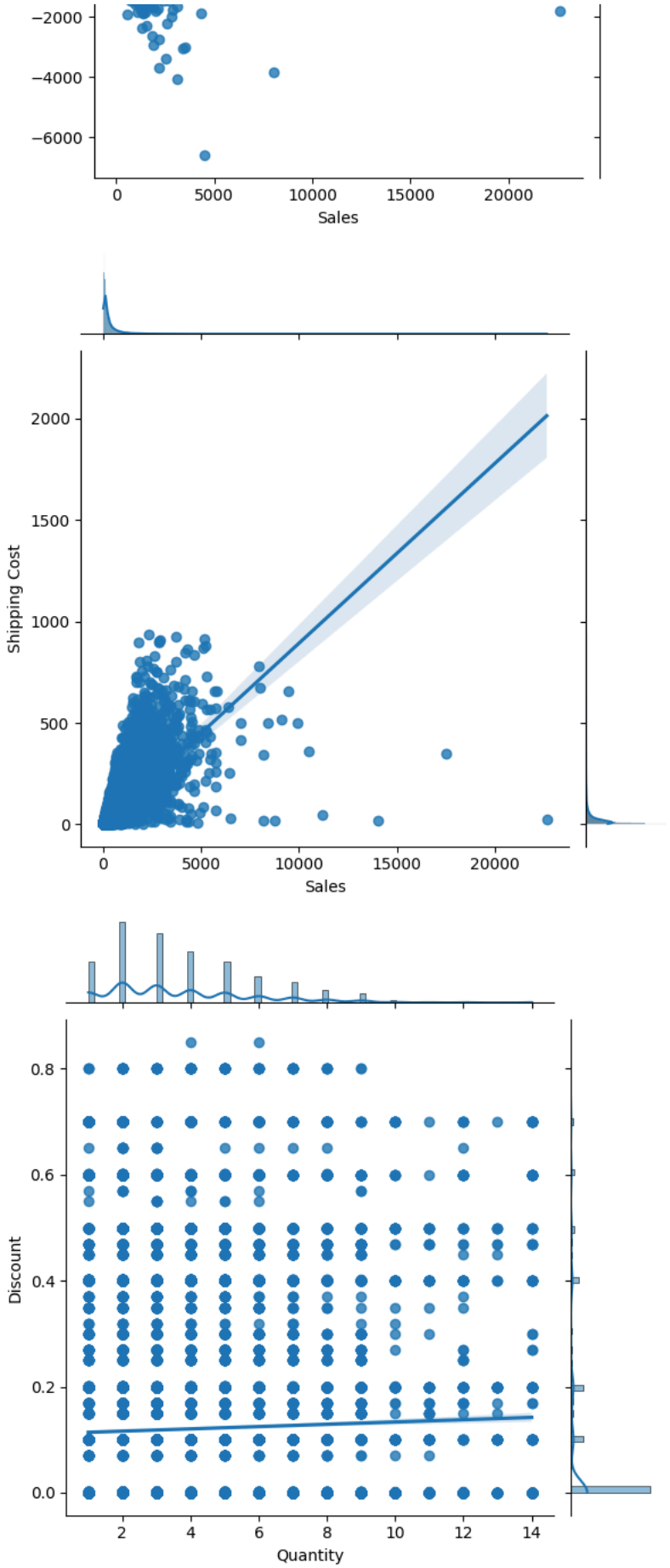
joint plots

```
import seaborn as sns

# Joint plots for pairs of numeric features
for x, y in pairs[:5]:  # Limiting to the first 5 pairs for demonstration
    sns.jointplot(x=x, y=y, data=data, kind='reg')
    plt.show()
```


Quantity
Sales
0
5
10
15
20
25
30
0
5000
10000
15000
20000

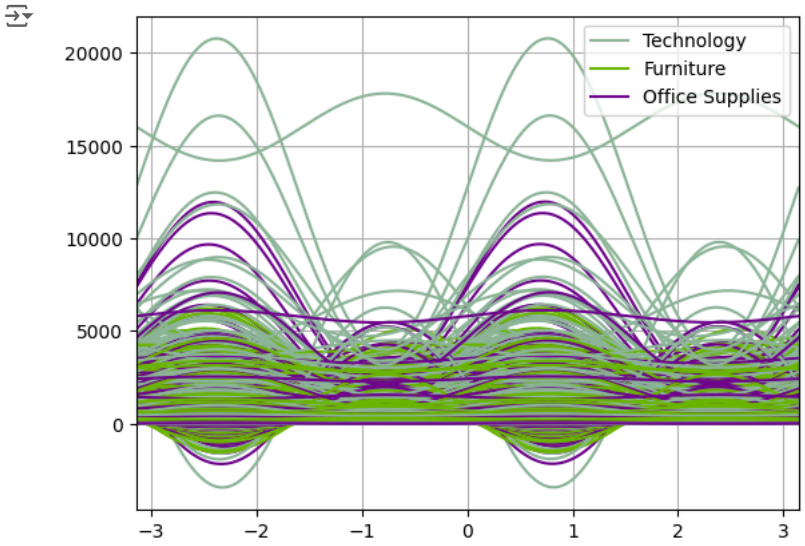

Discount
Sales
0.8
0.6
0.4
0.2
0.0
−0.2
0
5000
10000
15000
20000


Profit
8000
6000
4000
2000
0



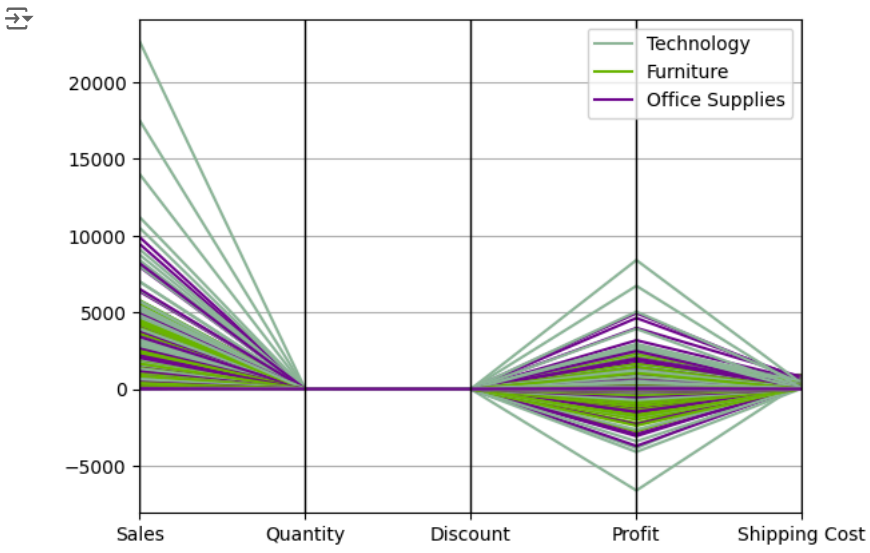
Andrews Curves

```
from pandas.plotting import andrews_curves

# Andrews curves for a subset of the data
andrews_curves(data[['Category'] + numeric_features].dropna(), 'Category')
plt.show()
```



Parallel Coordinate Plots

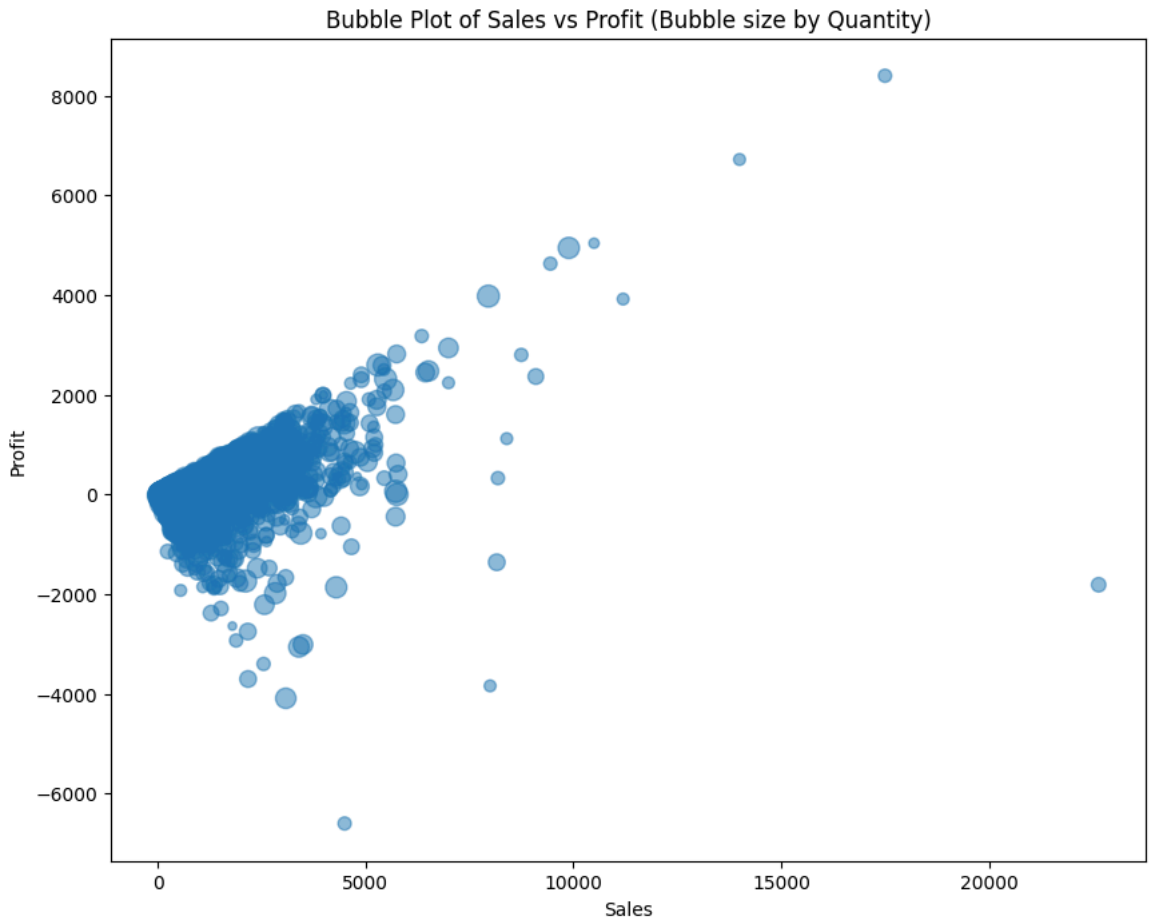
```
from pandas.plotting import parallel_coordinates

# Parallel coordinates for a subset of the data
parallel_coordinates(data[['Category'] + numeric_features].dropna(), 'Category')
plt.show()
```



This is a parallel coordinates plot visualizing the performance of three product categories: Technology, Furniture, and Office Supplies across five metrics: Sales, Quantity, Discount, Profit, and Shipping Cost. Each line represents a record, colored according to the product category. The plot shows that Technology generally has higher sales, while all categories cluster closely around similar values for other metrics, particularly Profit and Shipping Cost.

Bubble Plots

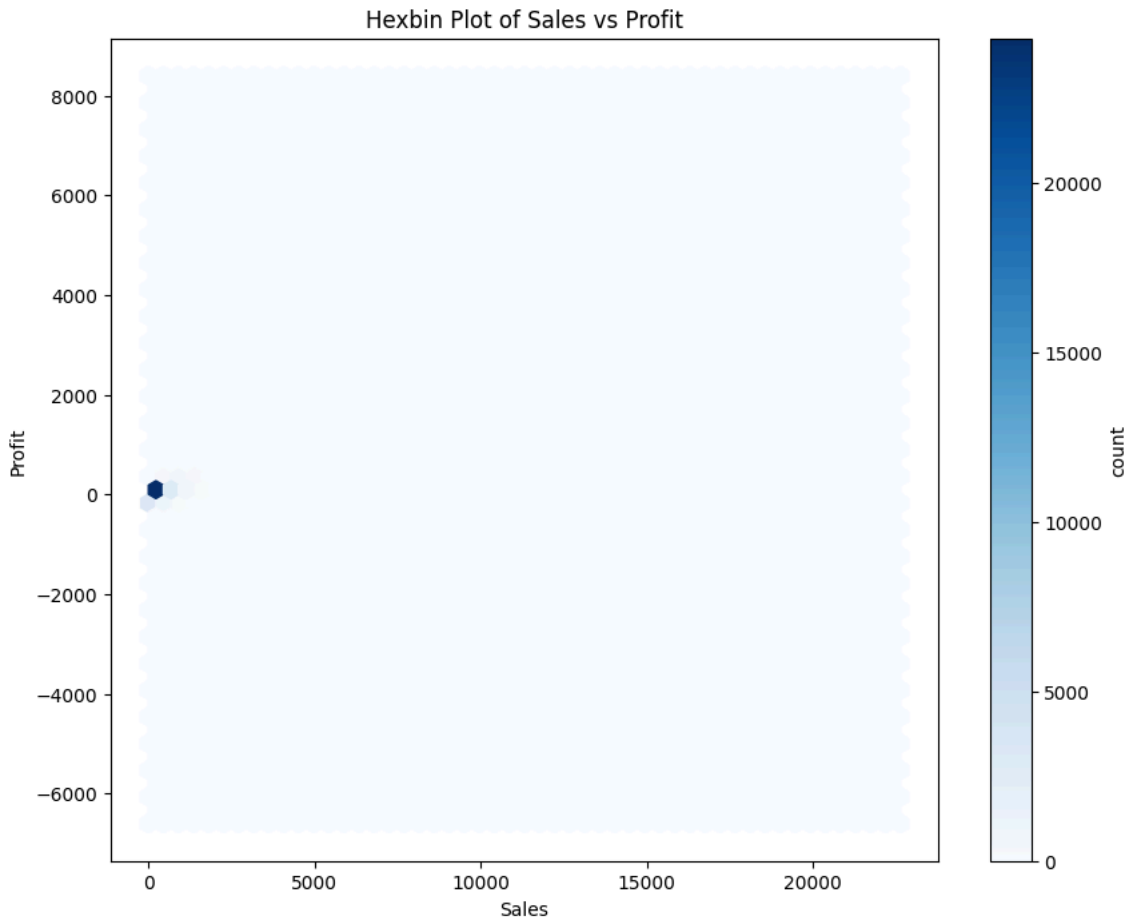
```
# Bubble plot (example with Sales vs. Profit, bubble size by Quantity)
plt.figure(figsize=(10, 8))
plt.scatter(data['Sales'], data['Profit'], s=data['Quantity']*10, alpha=0.5)
plt.title('Bubble Plot of Sales vs Profit (Bubble size by Quantity)')
plt.xlabel('Sales')
plt.ylabel('Profit')
plt.show()
```



This bubble plot shows the relationship between sales and profit, with bubble sizes representing the quantity sold. Most data points cluster around lower sales and profit values, indicating many smaller transactions. A few outliers show significantly higher sales and profits. The larger bubbles scattered across the plot suggest higher quantities sold in some instances, highlighting the variability in sales performance

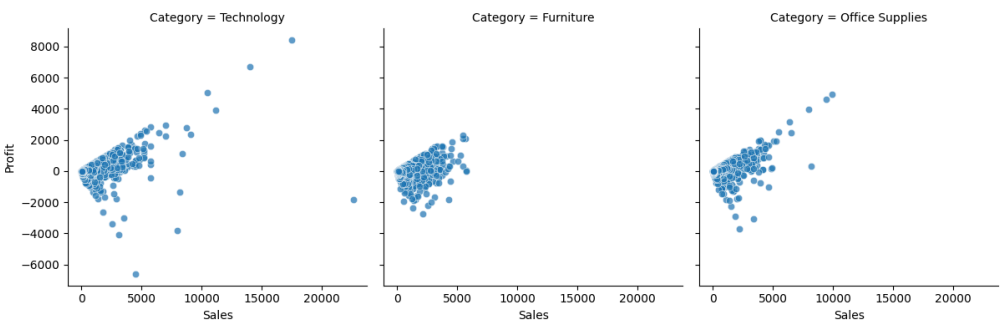
Hexbin Plots

```
# Hexbin plot for Sales vs. Profit
plt.figure(figsize=(10, 8))
plt.hexbin(data['Sales'], data['Profit'], gridsize=50, cmap='Blues')
cb = plt.colorbar(label='count')
plt.title('Hexbin Plot of Sales vs Profit')
plt.xlabel('Sales')
plt.ylabel('Profit')
plt.show()
```



Facet Grids

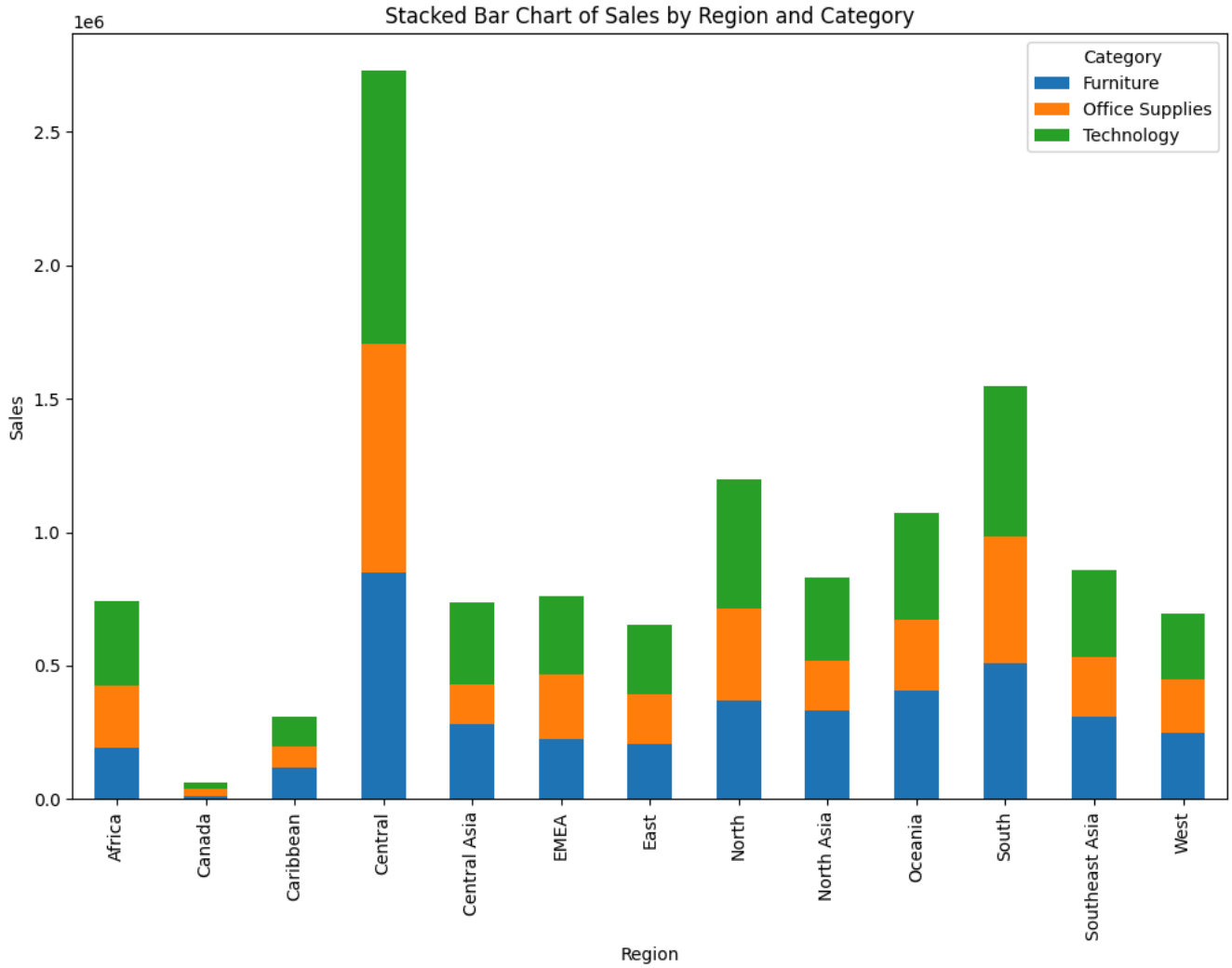
```
# Facet grid of Sales vs. Profit by Category
g = sns.FacetGrid(data, col="Category", col_wrap=4, height=4)
g.map(sns.scatterplot, "Sales", "Profit", alpha=.7)
g.add_legend()
plt.show()
```



In the category Technology, Furniture, and Office Supplies. In the Technology category, there's a broad range of sales with varying profits, including some high outliers. The Furniture category shows a tighter cluster of data points with a moderate range of sales and profits. Office Supplies also have a clustered distribution with a positive correlation between sales and profit, showing fewer outliers and a more consistent trend

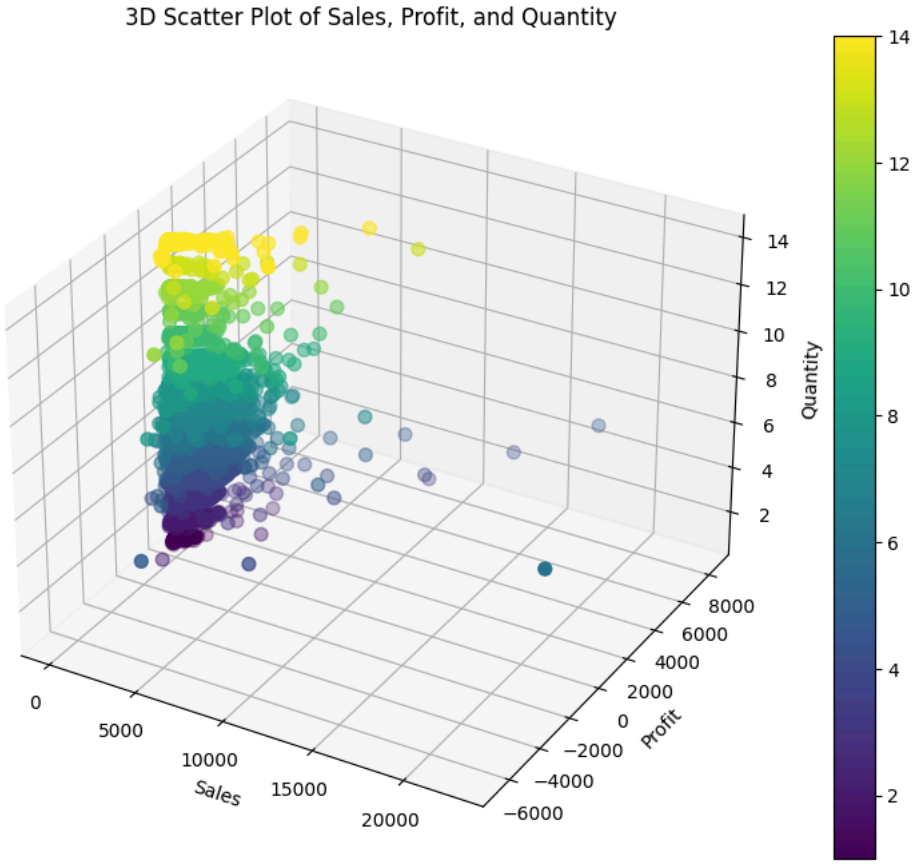
Stacked Bar Charts

```
# Stacked bar chart of Sales by Region and Category
stacked_data = data.groupby(['Region', 'Category'])['Sales'].sum().unstack()
stacked_data.plot(kind='bar', stacked=True, figsize=(12, 8))
plt.title('Stacked Bar Chart of Sales by Region and Category')
plt.xlabel('Region')
plt.ylabel('Sales')
plt.show()
```



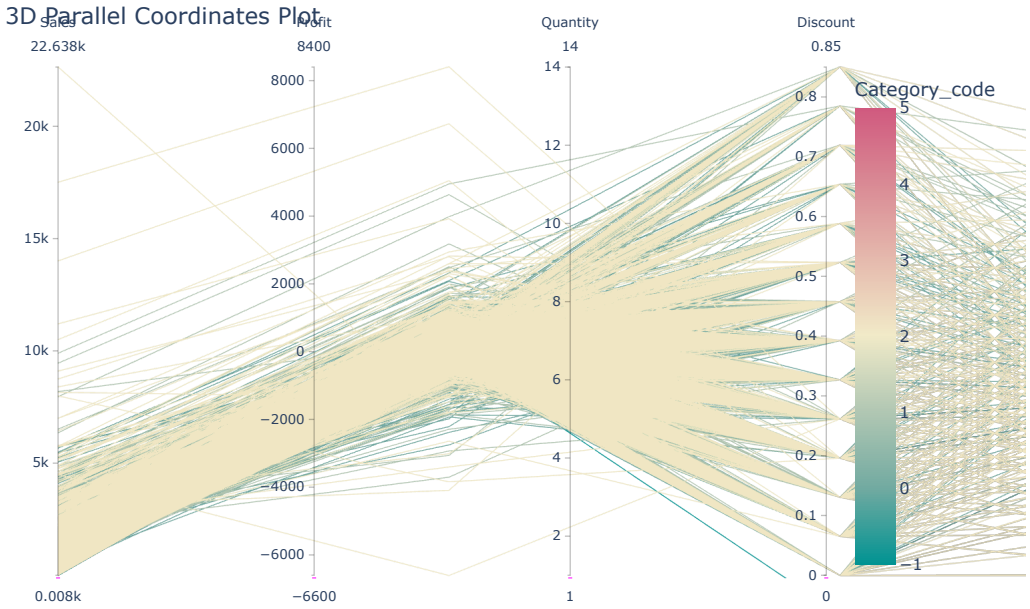
The stacked bar chart displays sales data across three categories: Furniture, Office Supplies, and Technology. Technology consistently has the largest share in most bars, with a notable spike indicating a significant surge in sales. Furniture and Office Supplies contribute steadily but less dominantly. Overall, Technology leads in sales, followed by Office Supplies and Furniture.

```
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

# 3D Scatter Plot (example with Sales, Profit, and Quantity)
fig = plt.figure(figsize=(10, 8))
ax = fig.add_subplot(111, projection='3d')
sc = ax.scatter(data['Sales'], data['Profit'], data['Quantity'], c=data['Quantity'], cmap='viridis', s=50)
ax.set_xlabel('Sales')
ax.set_ylabel('Profit')
ax.set_zlabel('Quantity')
plt.title('3D Scatter Plot of Sales, Profit, and Quantity')
plt.colorbar(sc)
plt.show()
```



The 3D scatter plot shows a positive correlation between sales and profit, with higher sales generally leading to higher profits. Quantities vary widely, with higher quantities clustered around higher sales. Most data points fall within low to medium sales and profit ranges, while extremely high sales and profits are less common. Losses occur across various sales and quantity levels.

```
import plotly.express as px

# Convert Category to numeric codes
data['Category_code'] = data['Category'].astype('category').cat.codes

# 3D Parallel Coordinates Plot
fig = px.parallel_coordinates(data, color='Category_code',
                              dimensions=['Sales', 'Profit', 'Quantity', 'Discount'],
                              color_continuous_scale=px.colors.diverging.Tealrose,
                              color_continuous_midpoint=2)
fig.update_layout(title='3D Parallel Coordinates Plot')
fig.show()
```



Axes and Variables:

Sales: The values range from approximately 0 to 22.638k. Profit: The values range from approximately -6600 to 8400. Quantity: The values range from 1 to 14. Discount: The values range from 0 to 0.85. Color Encoding:

The color of the lines represents the Category_code, which ranges from 0 to 5. The color gradient goes from teal (low values) to pink (high values). Line Representation:

Each line represents a data point and connects the values of the variables Sales, Profit, Quantity, and Discount. The color of the line indicates the Category_code of the data point. Insights:

Sales and Profit: There is a concentration of lines in the middle of the Profit axis, indicating that many data points have profits around the 0 to 2000 range. There are also lines extending to negative profits, showing losses. Quantity: The lines show a spread across all quantity values, with some clustering around specific quantities. Discount: The lines show that discounts are widely distributed across different data points, with some receiving higher discounts up to 0.85. Category Code: Different category codes are spread across the entire range of variables. The distribution of colors (teal to pink) is relatively even, suggesting that no single category code dominates any particular range of values. Trends and Patterns:

A noticeable trend is that higher sales generally align with higher profits, although there are exceptions. The data points with higher quantities tend to have a wide range of discounts, indicating variability in discount strategies. The color distribution suggests that the category codes are well-represented across different values of sales, profit, quantity, and discount.

3D SCATTER PLOT

```
# 3D Scatter Plot colored by density using Plotly
fig = px.scatter_3d(data, x='Sales', y='Profit', z='Quantity', color='Category')
fig.update_layout(title='3D Scatter Plot with Density Coloring')
fig.show()
```

Sales vs. Quantity:

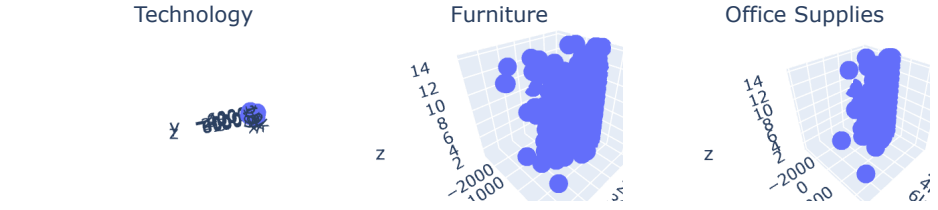
There is a general positive correlation between quantity and sales. Higher quantities are associated with higher sales values. Distribution of Categories:

All three categories (Technology, Furniture, and Office Supplies) are distributed across the scatter plot. Technology (blue) has a wider spread across all axes, indicating variability in sales, quantity, and the other measure. Furniture (red) and Office Supplies (green) show similar trends but are less spread out compared to Technology. Negative Values on X-Axis:

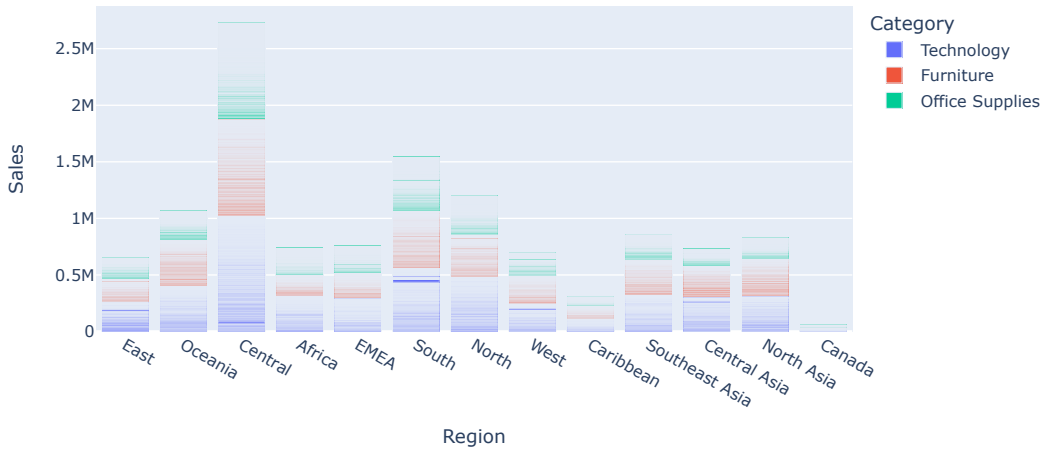
The presence of negative values on the x-axis suggests that this axis might represent profit or another measure that can have both positive and negative values. There are some points with high sales but negative values on the x-axis, indicating that some high-sales items might be unprofitable. Density and Clustering:

The majority of the data points are clustered around the lower range of quantity (4 to 10) and sales (0 to 10k). There are fewer data points as the sales and quantity increase, showing a decrease in density.

```
import plotly.express as px
from plotly.subplots import make_subplots
import math

# Get unique categories
categories = data['Category'].unique()

# Determine the number of rows and columns for subplots
num_categories = len(categories)
num_cols = 3
num_rows = math.ceil(num_categories / num_cols)

# Create subplots
fig = make_subplots(rows=num_rows, cols=num_cols,
                    specs=[[{'type': 'scatter3d'} for _ in range(num_cols)] for _ in range(num_rows)],
                    subplot_titles=categories)

# Add traces for each category
for idx, category in enumerate(categories):
    row = idx // num_cols + 1
    col = idx % num_cols + 1
    category_data = data[data['Category'] == category]
    trace = px.scatter_3d(category_data, x='Sales', y='Profit', z='Quantity').data[0]
    fig.add_trace(trace, row=row, col=col)

fig.update_layout(title='3D Facet Grids')
fig.show()
```




```
# 3D Stacked Bar Chart using Plotly
fig = px.bar(data, x='Region', y='Sales', color='Category', barmode='stack', height=400)
fig.update_layout(title='3D Stacked Bar Chart of Sales by Region and Category')
fig.show()
```



Sales Volume by Region:

Each region along the x-axis represents a different geographical area. The y-axis indicates the sales volume, ranging from 0 to 2.5 million. Product Categories:

Sales are divided into three product categories: Technology, Furniture, and Office Supplies, each represented by different colors. Regions with Highest Sales:

The North and South regions show the highest overall sales, with North slightly ahead. Canada also has significant sales figures. Regions with Moderate Sales:

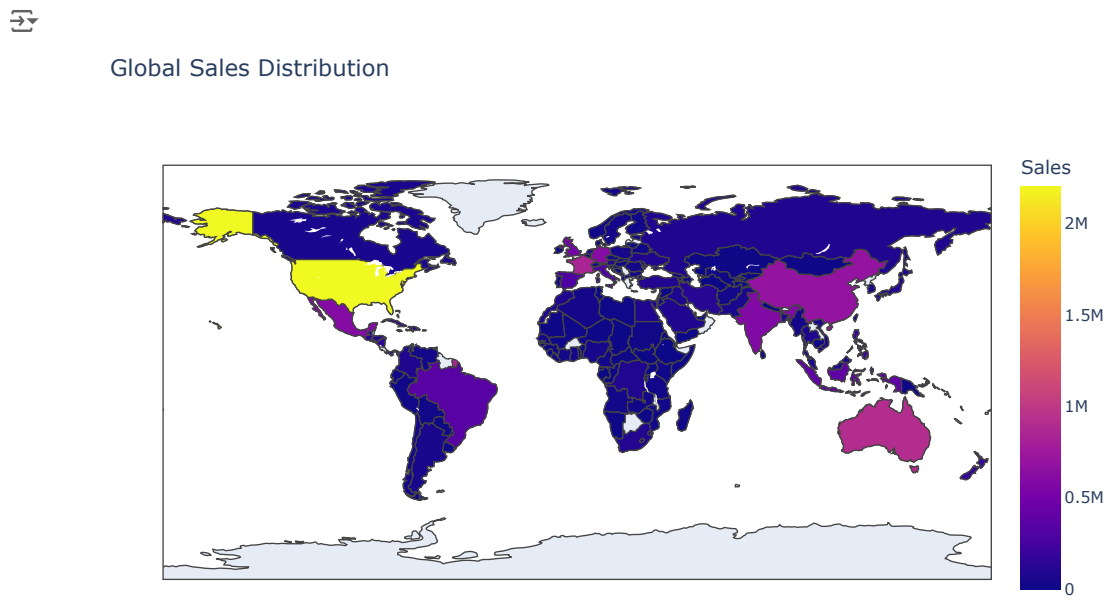
Oceania, Central, and Southeast Asia have moderate sales volumes. EMEA (Europe, Middle East, and Africa) shows a balanced distribution but lower overall sales compared to North and South. Regions with Lower Sales:

The Caribbean, Central Asia, and North Asia show lower sales volumes. Category Distribution:

In most regions, Technology seems to have the highest sales, followed by Furniture and then Office Supplies. North and South regions have higher sales in Technology and Furniture categories. Canada has a balanced distribution among all three categories.

sales by country

```
import pandas as pd
import plotly.express as px
country_sales = data.groupby('Country')['Sales'].sum().reset_index()

fig = px.choropleth(country_sales,
                    locations="Country",
                    locationmode='country names',
                    color="Sales",
                    hover_name="Country",
                    color_continuous_scale=px.colors.sequential.Plasma,
                    title="Global Sales Distribution")

fig.show()
```



Highest Sales:

The United States is highlighted in bright yellow, indicating it has the highest sales, over 2 million. Moderate Sales:

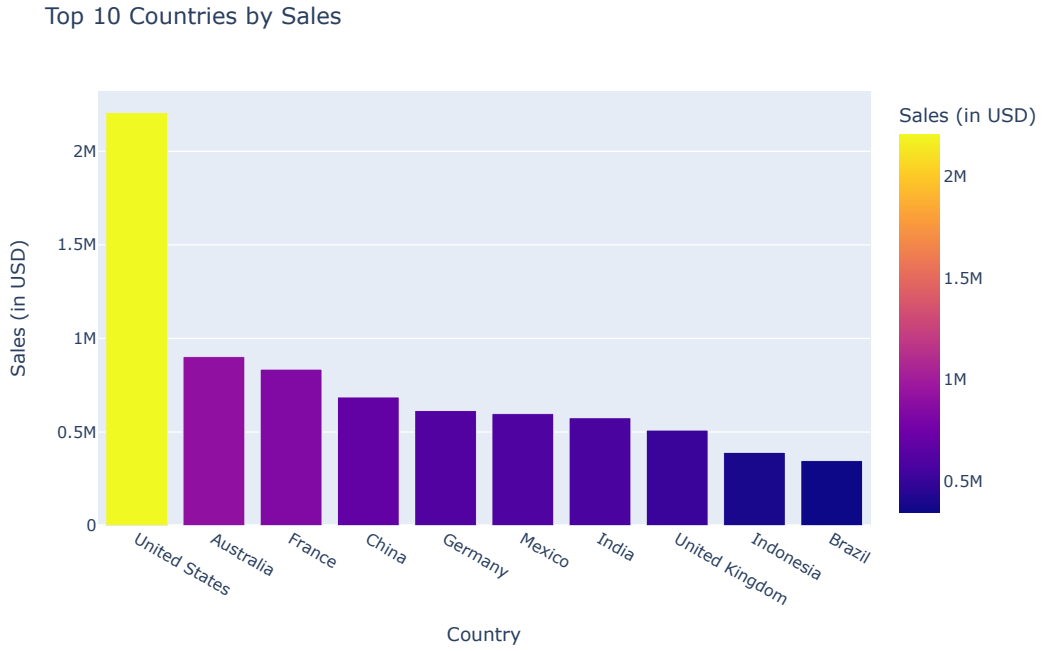
Countries like Canada, China, and Australia are shaded in lighter purple to pink, suggesting they have moderate sales figures. Lower Sales:

Many countries in Europe, Asia, and South America are shaded in dark purple, indicating lower sales values. No Data or Zero Sales:

Countries shaded in very dark purple or black likely have zero sales or no data available. This visualization effectively shows the geographical distribution of sales, highlighting regions with the highest and lowest sales volumes

top 10 states by sales

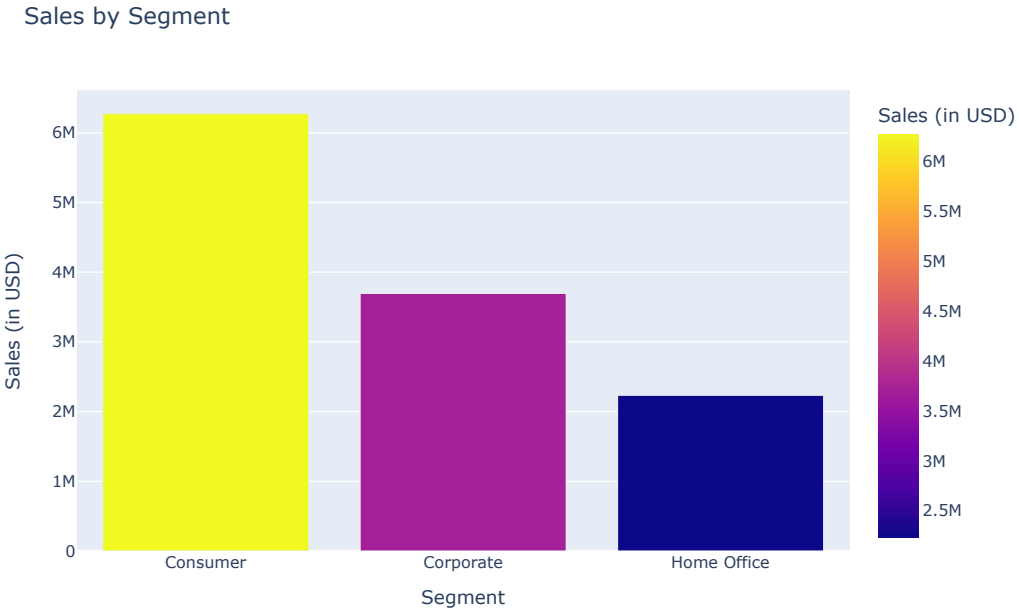
```
import pandas as pd
import plotly.express as px

# Aggregate sales by country
country_sales = data.groupby('Country')['Sales'].sum().reset_index()

# Get the top 10 countries by sales
top_10_sales = country_sales.nlargest(10, 'Sales')

# Create the bar plot for the top 10 sales
fig = px.bar(top_10_sales,
             x='Country',
             y='Sales',
             title='Top 10 Countries by Sales',
             labels={'Sales': 'Sales (in USD)', 'Country': 'Country'},
             color='Sales',
             color_continuous_scale=px.colors.sequential.Plasma)

fig.show()
```



United States leads by a significant margin with sales around $2million. Australia and France are the next highest, both hovering around$1 million in sales. China, Germany, Mexico, and India show moderate sales figures, each ranging between $600,000 to$800,000. United Kingdom, Indonesia, and Brazil display the lowest sales among the top ten, each just under $600,000. This visualization highlights the United States as the primary market in terms of sales volume, indicating a strong presence or demand in this region. The varied sales across other countries suggest diverse market penetration levels and potential areas for business growth or increased marketing efforts.

sales by segment

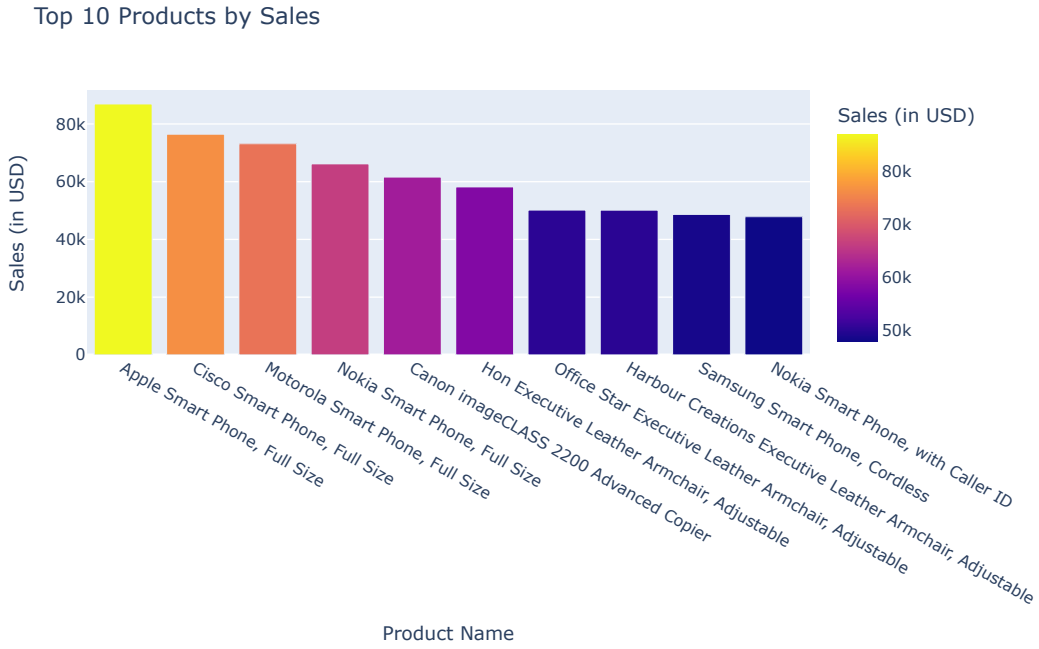
```
# Aggregate sales by segment
segment_sales = data.groupby('Segment')['Sales'].sum().reset_index()

# Create the bar plot for sales by segment
fig = px.bar(segment_sales,
             x='Segment',
             y='Sales',
             title='Sales by Segment',
             labels={'Sales': 'Sales (in USD)', 'Segment': 'Segment'},
             color='Sales',
             color_continuous_scale=px.colors.sequential.Plasma)

fig.show()
```



The Consumer segment has the highest sales, with a value close to $6 million. The Corporate segment shows significant sales as well, totaling about 4 million. The Home Office segment, while still substantial, has the lowest sales of the three, around $3.5 million. This visualization underscores that the consumer segment is the most robust in terms of sales, suggesting a strong market presence or demand within this group. Meanwhile, the corporate and home office segments, though trailing behind the consumer segment, still represent significant portions of the market, which might require different marketing strategies or product offerings to enhance performance in these areas.

top products by sales

```
# Aggregate sales by product
product_sales = data.groupby('Product Name')['Sales'].sum().reset_index()

# Get the top 10 products by sales
top_10_products = product_sales.nlargest(10, 'Sales')

# Create the bar plot for the top 10 products by sales
fig = px.bar(top_10_products,
             x='Product Name',
             y='Sales',
             title='Top 10 Products by Sales',
             labels={'Sales': 'Sales (in USD)', 'Product Name': 'Product Name'},
             color='Sales',
             color_continuous_scale=px.colors.sequential.Plasma)

fig.show()
```



Apple Smart Phone, Full Size leads with the highest sales, just under $80,000. Cisco Smart Phone, Full Size and Motorola Smart Phone, Full Size follow closely, each with sales between 70,000 and $75,000. Nokia Smart Phone, Full Size and Canon imageCLASS 2200 Advanced Copier are next, with sales in the range of 60,000 to $65,000. The remaining products, including various office furniture items like Hon Executive Leather Armchair – Adjustable and Harbour Creations Executive Leather Armchair, Adjustable, and other smartphone models like Nokia Smart Phone with Caller ID, show similar sales figures, each around 50,000 to $55,000. This visualization demonstrates the strong market performance of full-sized smart phones across several brands, as well as a substantial market for high-end office furniture and copiers, indicating robust sales in both consumer electronics and business equipment sectors.

dominating market by sales

```
# Aggregate sales by market
market_sales = data.groupby('Market')['Sales'].sum().reset_index()

# Create the bar plot for sales by market
fig = px.bar(market_sales,
             x='Market',
             y='Sales',
             title='Sales by Market',
             labels={'Sales': 'Sales (in USD)', 'Market': 'Market'},
             color='Sales',
             color_continuous_scale=px.colors.sequential.Plasma)

fig.show()
```