Problem Defination

The goal is to analyze the sales performance and profitability of a company based on a dataset containing sales order information. The analysis includes examining overall sales and profit, identifying trends over time, determining top-performing product categories and sub-categories, investigating the impact of factors like ship mode and discounts on sales and profitability, and identifying any seasonality or trends in the data

Asking Questions

# 1- What is top selling product 2-top-profitable products ? 3-How does the sales and profit performance vary across different regions? 4-Categories and Regions for sales 5-What is the overall sales performance of the company? 6-Ship mode impact on sales or profitability 7-How does discounting impact sales and profitability? 8-Is there seasonality in the data 9-What is the relationship between quantity and profitability?

In [54]:
```python
# import pandas as pd
import matplotlib.pyplot as plt
```

In [3]:
```python
df=pd.read_csv('Sample-Superstore.csv', encoding='latin')
```

In [4]:
```python
df.head()
```

Out[4]:

	Row ID	Order ID	Order Date	Ship Date	Ship Mode	Customer ID	Customer Name	Segment	Country	City	...	Postal Code	Region	Prod
0	1	CA-2016-152156	11/8/2016	11/11/2016	Second Class	CG-12520	Claire Gute	Consumer	United States	Henderson	...	42420	South	FUR-I 10001
1	2	CA-2016-152156	11/8/2016	11/11/2016	Second Class	CG-12520	Claire Gute	Consumer	United States	Henderson	...	42420	South	FUR-(10000
2	3	CA-2016-138688	6/12/2016	6/16/2016	Second Class	DV-13045	Darrin Van Huff	Corporate	United States	Los Angeles	...	90036	West	OFF-10000
3	4	US-2015-108966	10/11/2015	10/18/2015	Standard Class	SO-20335	Sean O'Donnell	Consumer	United States	Fort Lauderdale	...	33311	South	FUR-10000
4	5	US-2015-108966	10/11/2015	10/18/2015	Standard Class	SO-20335	Sean O'Donnell	Consumer	United States	Fort Lauderdale	...	33311	South	OFF-10000

5 rows × 21 columns

In [5]:
```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9994 entries, 0 to 9993
Data columns (total 21 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Row ID         9994 non-null   int64  
 1   Order ID       9994 non-null   object 
 2   Order Date     9994 non-null   object 
 3   Ship Date      9994 non-null   object 
 4   Ship Mode      9994 non-null   object 
 5   Customer ID    9994 non-null   object 
 6   Customer Name  9994 non-null   object 
 7   Segment        9994 non-null   object 
 8   Country        9994 non-null   object 
 9   City           9994 non-null   object 
 10  State          9994 non-null   object 
 11  Postal Code    9994 non-null   int64  
 12  Region         9994 non-null   object 
 13  Product ID     9994 non-null   object 
 14  Category       9994 non-null   object 
 15  Sub-Category   9994 non-null   object 
 16  Product Name   9994 non-null   object 
 17  Sales          9994 non-null   float64
 18  Quantity       9994 non-null   int64  
 19  Discount       9994 non-null   float64
 20  Profit         9994 non-null   float64
dtypes: float64(3), int64(3), object(15)
memory usage: 1.6+ MB
```

In [6]:
```python
df['Order Date']=pd.to_datetime(df['Order Date'])
df['Ship Date']=pd.to_datetime(df['Ship Date'])
```

In [7]:
```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9994 entries, 0 to 9993
Data columns (total 21 columns):
 #   Column         Non-Null Count  Dtype         
---  ------         --------------  -----         
 0   Row ID         9994 non-null   int64         
 1   Order ID       9994 non-null   object        
 2   Order Date     9994 non-null   datetime64[ns]
 3   Ship Date      9994 non-null   datetime64[ns]
 4   Ship Mode      9994 non-null   object        
 5   Customer ID    9994 non-null   object        
 6   Customer Name  9994 non-null   object        
 7   Segment        9994 non-null   object        
 8   Country        9994 non-null   object        
 9   City           9994 non-null   object        
 10  State          9994 non-null   object        
 11  Postal Code    9994 non-null   int64         
 12  Region         9994 non-null   object        
 13  Product ID     9994 non-null   object        
 14  Category       9994 non-null   object        
 15  Sub-Category   9994 non-null   object        
 16  Product Name   9994 non-null   object        
 17  Sales          9994 non-null   float64       
 18  Quantity       9994 non-null   int64         
 19  Discount       9994 non-null   float64       
 20  Profit         9994 non-null   float64       
dtypes: datetime64[ns](2), float64(3), int64(3), object(13)
memory usage: 1.6+ MB
```

In [8]:
```python
df.describe()
```

Out[8]:

	Row ID	Postal Code	Sales	Quantity	Discount	Profit
count	9994.000000	9994.000000	9994.000000	9994.000000	9994.000000	9994.000000
mean	4997.500000	55190.379428	229.858001	3.789574	0.156203	28.656896
std	2885.163629	32063.693350	623.245101	2.225110	0.206452	234.260108
min	1.000000	1040.000000	0.444000	1.000000	0.000000	-6599.978000
25%	2499.250000	23223.000000	17.280000	2.000000	0.000000	1.728750
50%	4997.500000	56430.500000	54.490000	3.000000	0.200000	8.666500
75%	7495.750000	90008.000000	209.940000	5.000000	0.200000	29.364000
max	9994.000000	99301.000000	22638.480000	14.000000	0.800000	8399.976000

In [9]:
```python
df_cat=df[['Ship Mode','Customer ID','Customer Name','Segment','Country','City','State','Region',
           'Product ID','Category','Sub-Category','Product Name']]
```

In [10]:
```
df_cat.head()
```

Out[10]:

	Ship Mode	Customer ID	Customer Name	Segment	Country	City	State	Region	Product ID	Category	Sub-Category	Product Name
0	Second Class	CG-12520	Claire Gute	Consumer	United States	Henderson	Kentucky	South	FUR-BO-10001798	Furniture	Bookcases	Bush Somerset Collection Bookcase
1	Second Class	CG-12520	Claire Gute	Consumer	United States	Henderson	Kentucky	South	FUR-CH-10000454	Furniture	Chairs	Hon Deluxe Fabric Upholstered Stacking Chairs,...
2	Second Class	DV-13045	Darrin Van Huff	Corporate	United States	Los Angeles	California	West	OFF-LA-10000240	Office Supplies	Labels	Self-Adhesive Address Labels for Typewriters b...
3	Standard Class	SO-20335	Sean O'Donnell	Consumer	United States	Fort Lauderdale	Florida	South	FUR-TA-10000577	Furniture	Tables	Bretford CR4500 Series Slim Rectangular Table
4	Standard Class	SO-20335	Sean O'Donnell	Consumer	United States	Fort Lauderdale	Florida	South	OFF-ST-10000760	Office Supplies	Storage	Eldon Fold 'N Roll Cart System

In [11]:
```
for feature in df_cat.columns:
    print(feature ,": ",df[feature].nunique())
```

```
Ship Mode :  4
Customer ID :  793
Customer Name :  793
Segment :  3
Country :  1
City :  531
State :  49
Region :  4
Product ID :  1862
Category :  3
Sub-Category :  17
Product Name :  1850
```

In [12]:
```
df['Sub-Category'].value_counts()
```

Out[12]:
```
Binders        1523
Paper          1370
Furnishings     957
Phones          889
Storage         846
Art             796
Accessories     775
Chairs          617
Appliances      466
Labels          364
Tables          319
Envelopes       254
Bookcases       228
Fasteners       217
Supplies        190
Machines        115
Copiers          68
Name: Sub-Category, dtype: int64
```

Top selling product

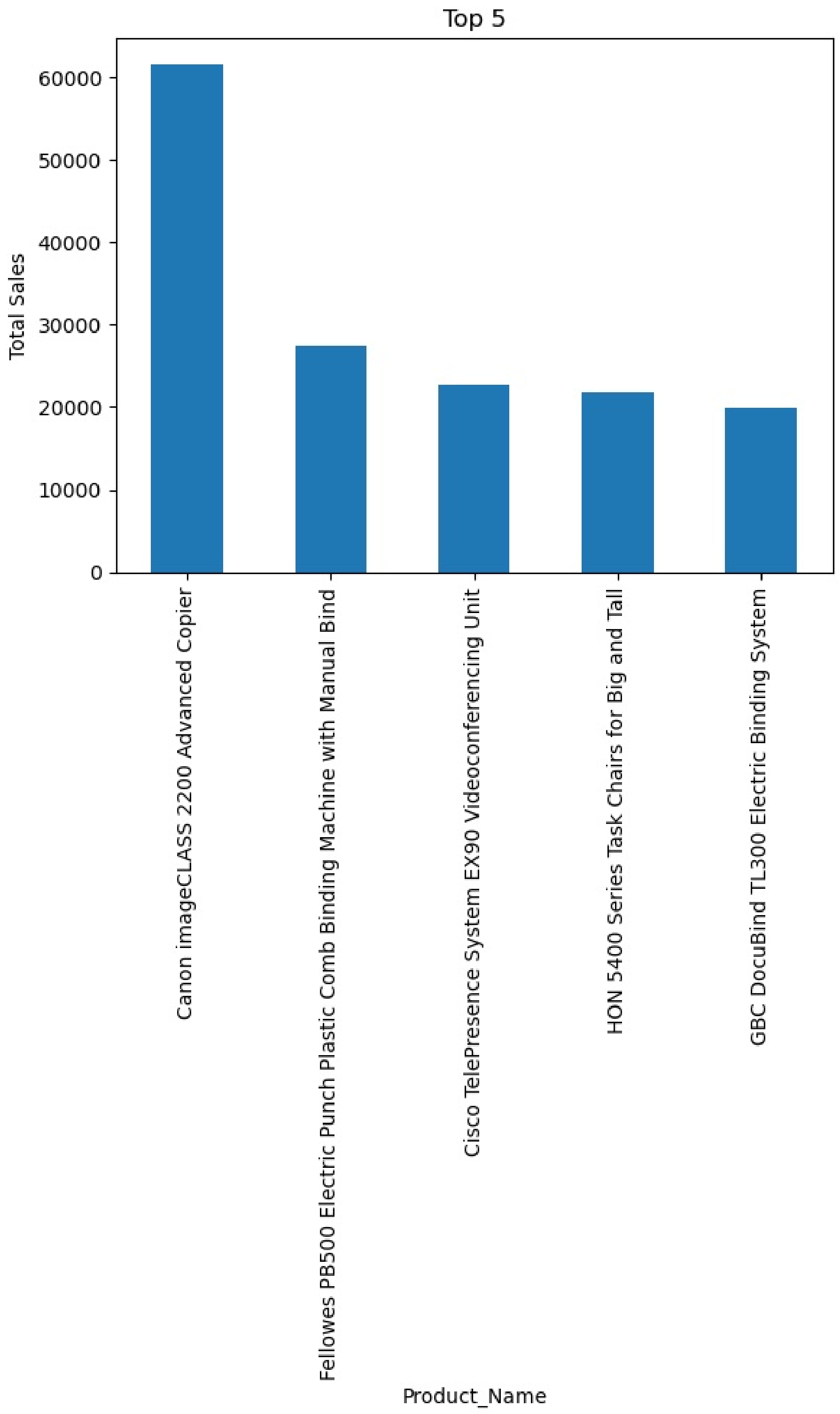
In [13]:
```
product_group = df.groupby(['Product Name'])['Sales'].sum()
```

In [14]:
```
product_group.head()
```

Out[14]:
```
Product Name
"While you Were Out" Message Book, One Form per Page     25.228
#10 Gummed Flap White Envelopes, 100/Box                 41.300
#10 Self-Seal White Envelopes                           108.682
#10 White Business Envelopes,4 1/8 x 9 1/2              488.904
#10- 4 1/8" x 9 1/2" Recycled Envelopes                 286.672
Name: Sales, dtype: float64
```

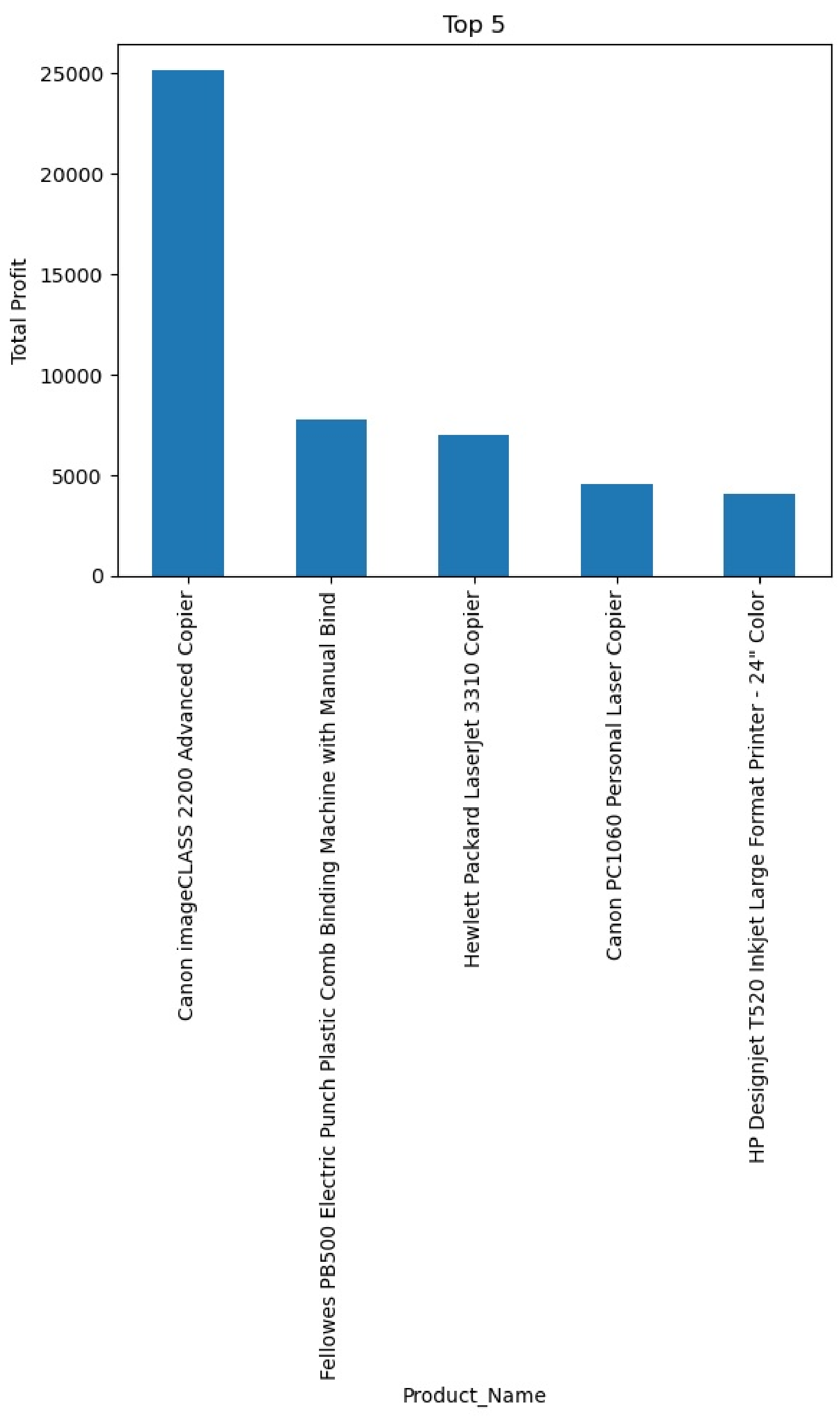
```
In [15]: top_products=product_group.sort_values(ascending=False)
```

```
In [16]: top_products[:5].plot(kind="bar")
         plt.title("Top 5")
         plt.xlabel("Product_Name")
         plt.ylabel("Total Sales")
         plt.show()
```



top-profitable products ?

```
In [50]: product_group = df.groupby(['Product Name'])['Profit'].sum()
         top_products=product_group.sort_values(ascending=False)
         top_products[:5].plot(kind="bar")
         plt.title("Top 5")
         plt.xlabel("Product_Name")
         plt.ylabel("Total Profit")
         plt.show()
```



In [18]:
```
product_group = df.groupby(['Product Name'])['Discount'].sum()
top_products=product_group.sort_values(ascending=False)
top_products[:5]
```

Out[18]:
```
Product Name
Storex Dura Pro Binders                                     7.2
Avery Non-Stick Binders                                     6.8
GBC Instant Report Kit                                      6.4
Avery Self-Adhesive Photo Pockets for Polaroid Photos       5.9
GBC Standard Recycled Report Covers, Clear Plastic Sheets   5.9
Name: Discount, dtype: float64
```

In [19]:
```
correlation = df['Sales'].corr(df['Profit'])
print("Correlation:", correlation)
```

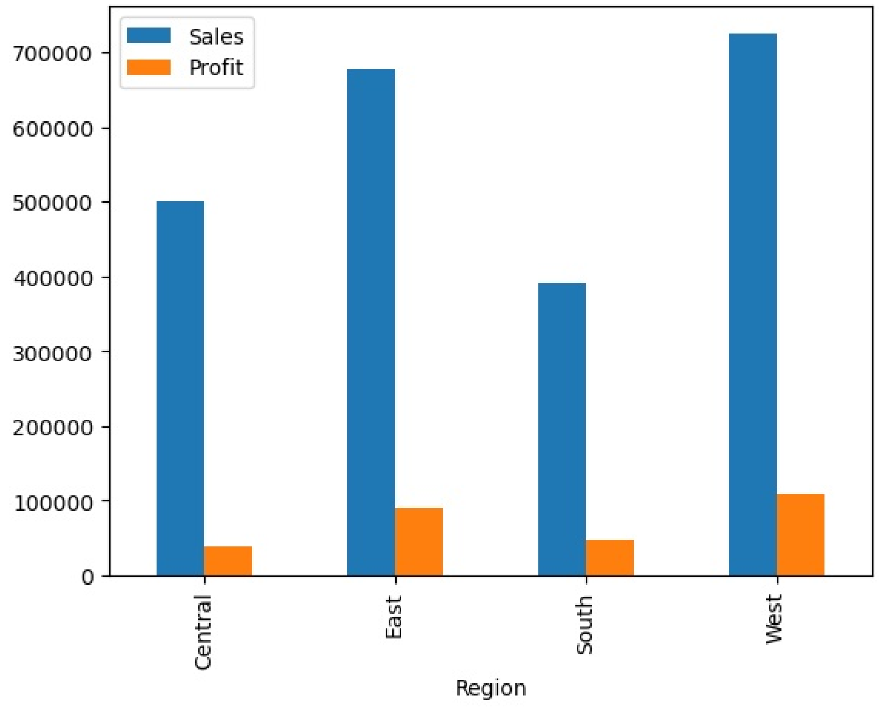
```
Correlation: 0.4790643497377058
```

How does the sales and profit performance vary across different regions?

In [51]:
```
region_group=df.groupby(['Region']).sum()[['Sales','Profit']]
region_group.plot(kind="bar")
```

```
plt.show()
```

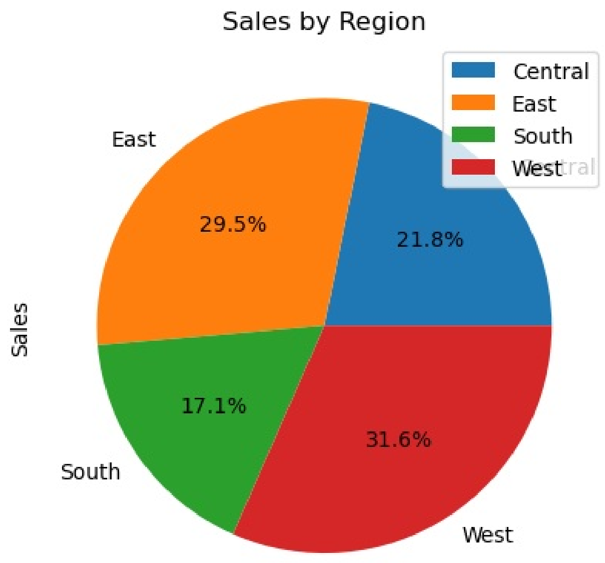
```
C:\Users\Yaseen\AppData\Local\Temp\ipykernel_11484\1005167294.py:1: FutureWarning: The default value of numeric_
only in DataFrameGroupBy.sum is deprecated. In a future version, numeric_only will default to False. Either spec
ify numeric_only or select only columns which should be valid for the function.
  region_group=df.groupby(['Region']).sum()[['Sales','Profit']]
```



In [28]:
```
# sales by Region
sales_Regions=df.groupby(['Region'])['Sales'].sum()
# Plotting the pie chart
plt.pie(sales_Regions, labels=sales_Regions.index, autopct='%1.1f%%')
plt.ylabel("Sales")
plt.title("Sales by Region")

# Display the values on the pie chart
plt.legend(sales_Regions.index, loc="best")

# Show the pie chart
plt.show()
```



Categories and Regions for sales

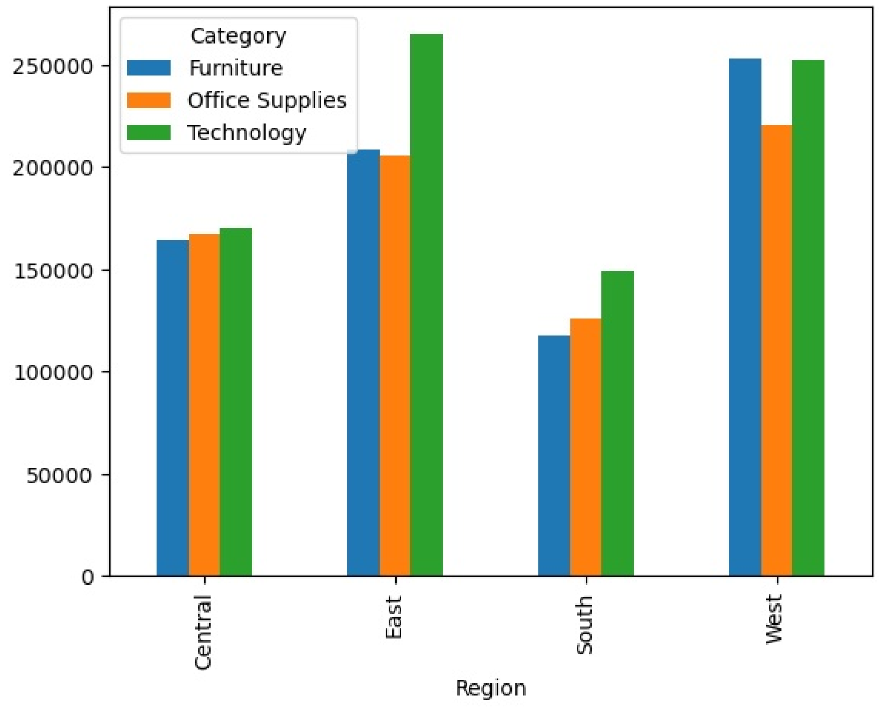
In [23]:
```
pivot_table=df.pivot_table(index='Region',columns='Category',values='Sales',aggfunc="sum")
pivot_table
```

Out[23]:

Category	Furniture	Office Supplies	Technology
Region			
Central	163797.1638	167026.415	170416.312
East	208291.2040	205516.055	264973.981
South	117298.6840	125651.313	148771.908
West	252612.7435	220853.249	251991.832

In [24]:

```python
pivot_table.plot(kind="bar",stacked=False)
plt.show()
```



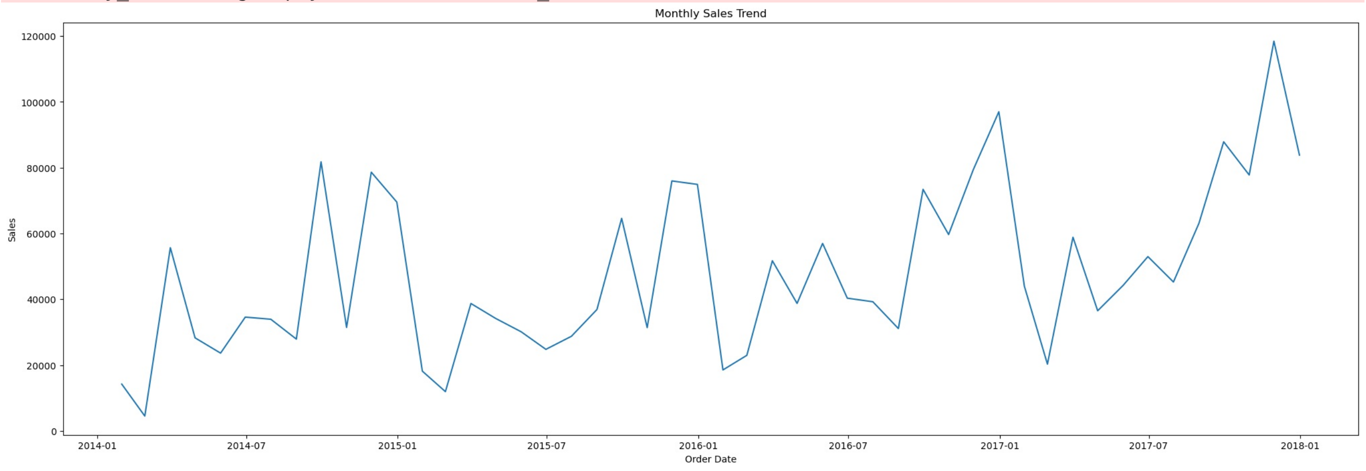
What is the overall sales performance of the company?

In [25]:

```python
#sales trend over time
monthly_sales=df.groupby(['Order Date'],as_index=False).sum()
monthly_sales.set_index('Order Date')
monthly_sales=monthly_sales.resample('M', on='Order Date').sum()

# Plot
plt.figure(figsize=(25,8))
plt.plot(monthly_sales['Sales'])
plt.xlabel("Order Date")
plt.ylabel("Sales")
plt.title("Monthly Sales Trend")
plt.show()
```

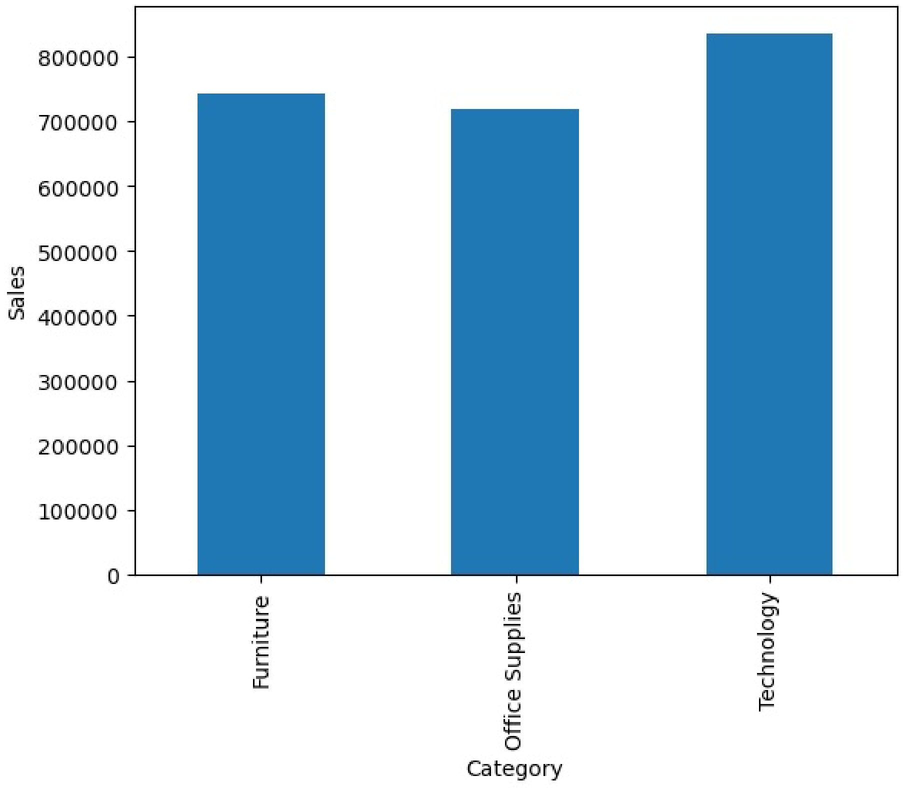
```
C:\Users\Yaseen\AppData\Local\Temp\ipykernel_11484\3593729363.py:2: FutureWarning: The default value of numeric_
only in DataFrameGroupBy.sum is deprecated. In a future version, numeric_only will default to False. Either spec
ify numeric_only or select only columns which should be valid for the function.
  monthly_sales=df.groupby(['Order Date'],as_index=False).sum()
```



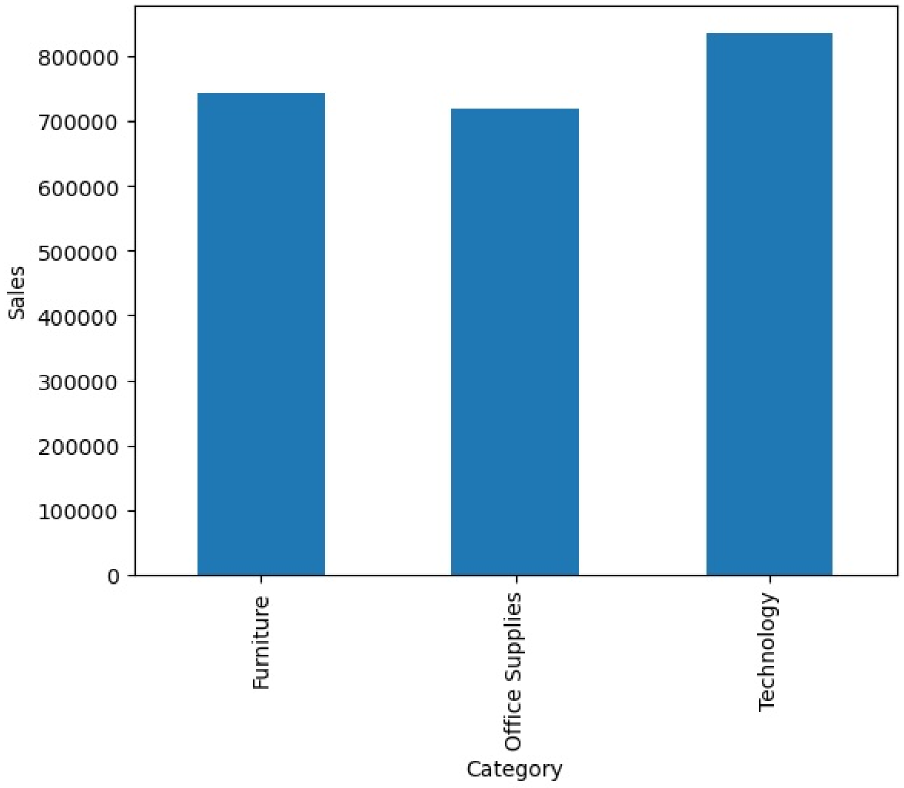
In [26]:

```python
# sales by category
```

```
category_by_sales=df.groupby(['Category'])['Sales'].sum()
category_by_sales
category_by_sales.plot(kind="bar")
plt.ylabel("Sales")
plt.show()
```

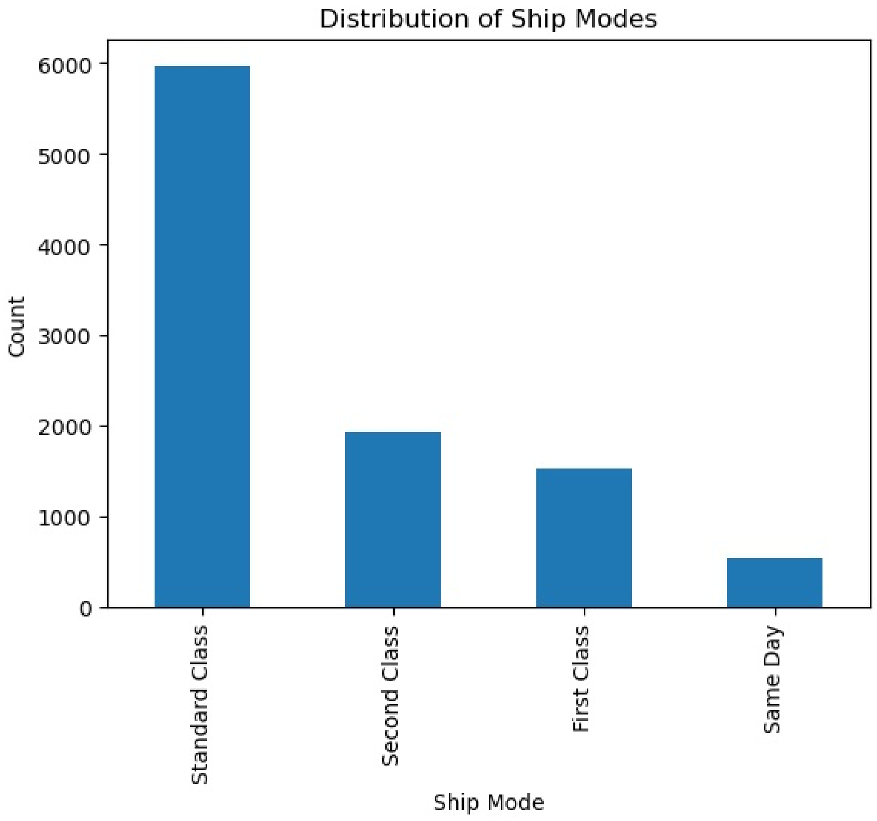


In [27]:
```
# sales by Segment
sales_by_Segment=df.groupby(['Segment'])['Sales'].sum()
sales_by_Segment
category_by_sales.plot(kind="bar")
plt.ylabel("Sales")
plt.show()
```



Ship mode impact on sales or profitability

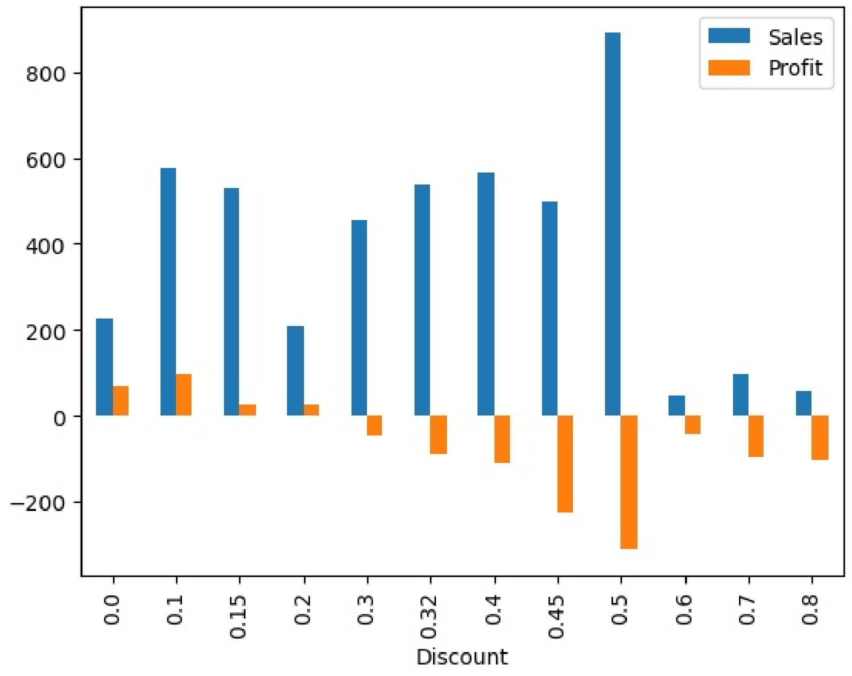
In [30]:
```
ship_mode_counts = df['Ship Mode'].value_counts()
ship_mode_counts.plot(kind='bar')
plt.xlabel('Ship Mode')
plt.ylabel('Count')
plt.title('Distribution of Ship Modes')
plt.show()
```



How does discounting impact sales and profitability?

In [31]:
```
discount_group=df.groupby(['Discount']).mean()[['Sales','Profit']]
disc=discount_group.plot(kind="bar")
plt.show()
```

```
C:\Users\Yaseen\AppData\Local\Temp\ipykernel_11484\2493961339.py:1: FutureWarning: The default value of numeric_
only in DataFrameGroupBy.mean is deprecated. In a future version, numeric_only will default to False. Either spe
cify numeric_only or select only columns which should be valid for the function.
  discount_group=df.groupby(['Discount']).mean()[['Sales','Profit']]
```



In [32]:
```
correlation = df['Discount'].corr(df['Profit'])
print("Correlation between Discount and Profit:", correlation)
```

```
Correlation between Discount and Profit: -0.21948745637176806
```

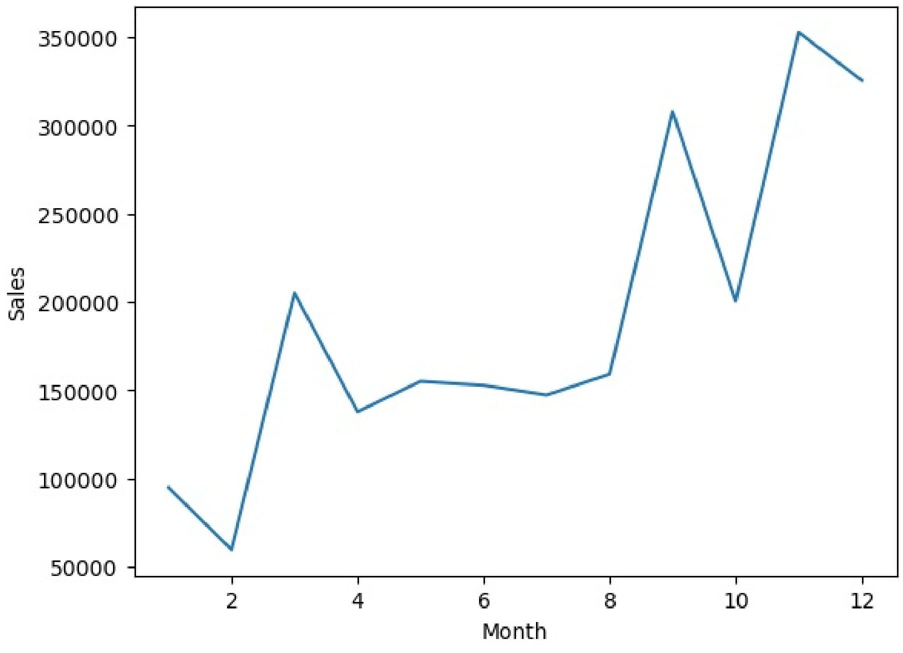
In [33]:
```
correlation = df['Discount'].corr(df['Sales'])
print("Correlation between Discount and Sales:", correlation)
```

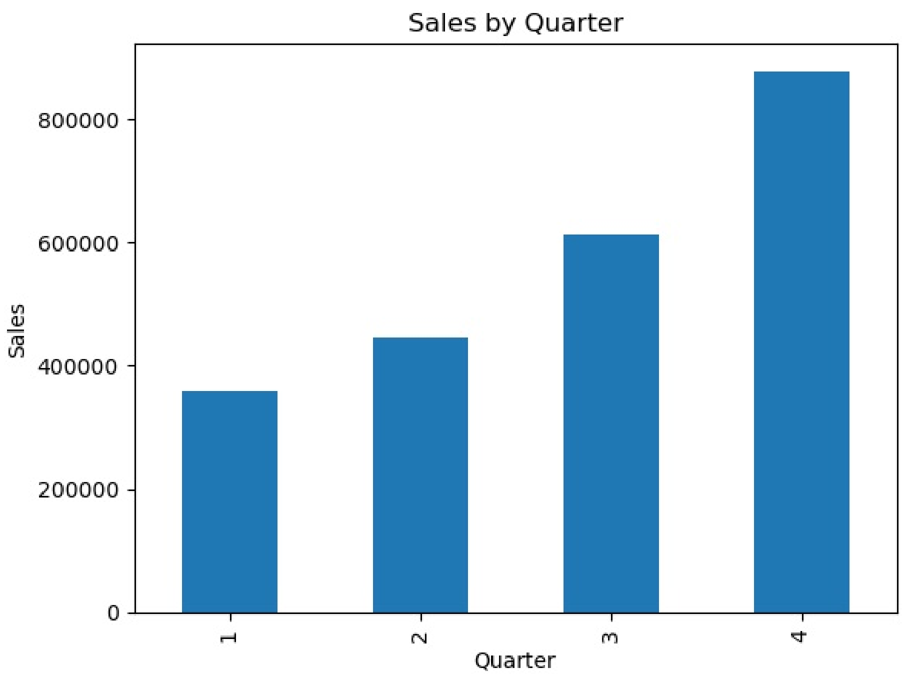
```
Correlation between Discount and Sales: -0.028190124157535553
```

Is there seasonality in the data

```
In [42]: df['Order Date'] = pd.to_datetime(df['Order Date'])
         df['Order Year']=df['Order Date'].dt.year
         df['Order Month']=df['Order Date'].dt.month
         df['Order Quarter']=df['Order Date'].dt.quarter
         #grouping
         sales_by_month=df.groupby('Order Month')['Sales'].sum()
         sales_by_quarter=df.groupby('Order Quarter')['Sales'].sum()
         # plotting
```

```
In [45]: sales_by_month.plot(kind="line")
         plt.xlabel("Month")
         plt.ylabel("Sales")
         plt.show()
         # Bar chart for sales by quarter
         sales_by_quarter.plot(kind='bar')
         plt.xlabel('Quarter')
         plt.ylabel('Sales')
         plt.title('Sales by Quarter')
         plt.show()
```

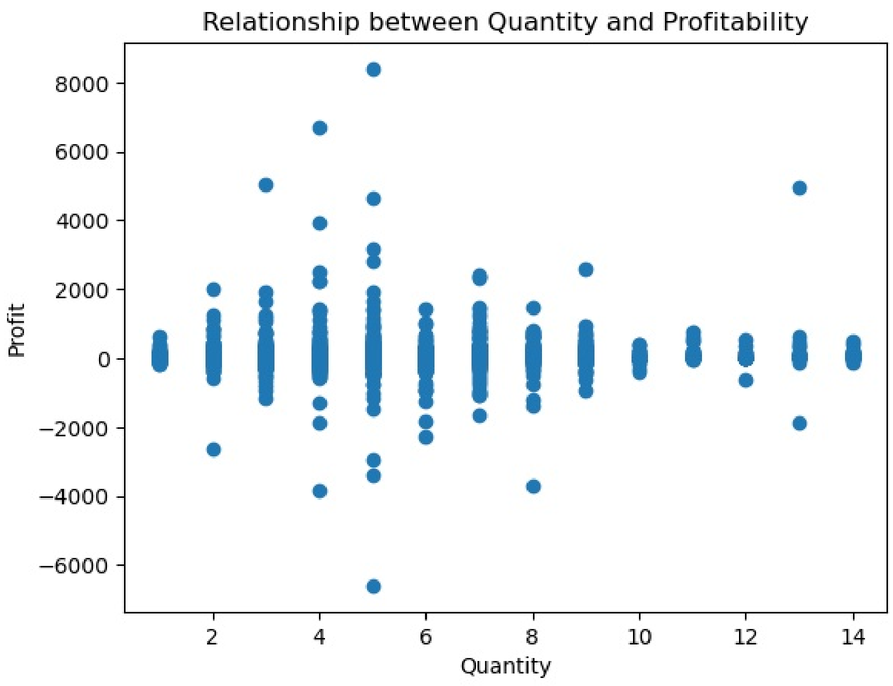




What is the relationship between quantity and profitability?

```
In [47]: avg_profit_by_quantity = df.groupby('Quantity')['Profit'].mean()
```

```
plt.scatter(df['Quantity'], df['Profit'])
plt.xlabel('Quantity')
plt.ylabel('Profit')
plt.title('Relationship between Quantity and Profitability')
plt.show()
```



In [48]:
```
correlation = df['Quantity'].corr(df['Profit'])
print(f"Correlation coefficient: {correlation}")
```

```
Correlation coefficient: 0.06625318912428485
```

In [49]:
```
# Almost no corrrelation between them
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js