Data Analysis on Sales dataset

Project By: Firdaush Alam

Date:23-07-2025

1. Importing Pandas Library

In [1]:
```
import pandas as pd
```

2.Reading the csv file data using panda and storing into variable data.

In [2]:
```
data=pd.read_csv('sales_data.csv')
```

3. Displaying the data

3.1 Length of the data

In [3]:
```
len(data)
```

Out[3]:
```
1000
```

3.2 Shape of the data

In [4]:
```
data.shape
```

Out[4]:
```
(1000, 14)
```

3.3 Data of Top 5 Rows

In [5]:
```
data.head()
```

Out[5]:

	Product_ID	Sale_Date	Sales_Rep	Region	Sales_Amount	Quantity_Sold	Product_Cat
0	1052	2023-02-03	Bob	North	5053.97	18	Fu
1	1093	2023-04-21	Bob	West	4384.02	17	Fu
2	1015	2023-09-21	David	South	4631.23	30	
3	1072	2023-08-24	Bob	South	2167.94	39	Cl
4	1061	2023-03-24	Charlie	East	3750.20	13	Elec

3.4 Data of 5 Rows from bottom

In [6]:
```
data.tail()
```

Out[6]:

	Product_ID	Sale_Date	Sales_Rep	Region	Sales_Amount	Quantity_Sold	Product_(
995	1010	2023-04-15	Charlie	North	4733.88	4	
996	1067	2023-09-07	Bob	North	4716.36	37	
997	1018	2023-04-27	David	South	7629.70	17	
998	1100	2023-12-20	David	West	1629.47	39	E
999	1086	2023-08-16	Alice	East	4923.93	48	

3.5 Displaying the first n rows

In [8]:
```
data.head(10)
```

Out[8]:

	Product_ID	Sale_Date	Sales_Rep	Region	Sales_Amount	Quantity_Sold	Product_Cat
0	1052	2023-02-03	Bob	North	5053.97	18	Fu
1	1093	2023-04-21	Bob	West	4384.02	17	Fu
2	1015	2023-09-21	David	South	4631.23	30	
3	1072	2023-08-24	Bob	South	2167.94	39	Cl
4	1061	2023-03-24	Charlie	East	3750.20	13	Elec
5	1021	2023-02-11	Charlie	West	3761.15	32	
6	1083	2023-04-11	Bob	West	618.31	29	Fu
7	1087	2023-01-06	Eve	South	7698.92	46	Fu
8	1075	2023-06-29	David	South	4223.39	30	Fu
9	1075	2023-10-09	Charlie	West	8239.58	18	Cl

3.6 Information of the dataset for numerical data and categorical data

In [9]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 14 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   Product_ID            1000 non-null   int64  
 1   Sale_Date             1000 non-null   object 
 2   Sales_Rep             1000 non-null   object 
 3   Region                1000 non-null   object 
 4   Sales_Amount          1000 non-null   float64
 5   Quantity_Sold         1000 non-null   int64  
 6   Product_Category      1000 non-null   object 
 7   Unit_Cost             1000 non-null   float64
 8   Unit_Price            1000 non-null   float64
 9   Customer_Type         1000 non-null   object 
 10  Discount              1000 non-null   float64
 11  Payment_Method        1000 non-null   object 
 12  Sales_Channel         1000 non-null   object 
 13  Region_and_Sales_Rep  1000 non-null   object 
dtypes: float64(4), int64(2), object(8)
memory usage: 109.5+ KB
```

3.7 Stastical summary of the dataset

In [11]:
```
data.describe()
```

Out[11]:

	Product_ID	Sales_Amount	Quantity_Sold	Unit_Cost	Unit_Price	Discount
count	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000	1000.00000
mean	1050.128000	5019.265230	25.355000	2475.304550	2728.440120	0.15239
std	29.573505	2846.790126	14.159006	1417.872546	1419.399839	0.08720
min	1001.000000	100.120000	1.000000	60.280000	167.120000	0.00000
25%	1024.000000	2550.297500	13.000000	1238.380000	1509.085000	0.08000
50%	1051.000000	5019.300000	25.000000	2467.235000	2696.400000	0.15000
75%	1075.000000	7507.445000	38.000000	3702.865000	3957.970000	0.23000
max	1100.000000	9989.040000	49.000000	4995.300000	5442.150000	0.30000

3.8 Accessing the data using the three methods of pandas:

(i) By indexing
(ii)By slicing
(iii)By Filtering

3.8.1.1 Accessing the data of column_name "Sales_Rep" using index.

In [12]:
```
data["Sales_Rep"]
```

Out[12]:
```
0          Bob
1          Bob
2        David
3          Bob
4      Charlie
        ...
995    Charlie
996        Bob
997      David
998      David
999      Alice
Name: Sales_Rep, Length: 1000, dtype: object
```

3.8.1.2 Accessing the data of Multiple columns at once using index.

In [17]:
```
data[["Sales_Rep", "Region", "Customer_Type", "Sales_Amount"]].head()
```

Out[17]:

	Sales_Rep	Region	Customer_Type	Sales_Amount
0	Bob	North	Returning	5053.97
1	Bob	West	Returning	4384.02
2	David	South	Returning	4631.23
3	Bob	South	New	2167.94
4	Charlie	East	New	3750.20

3.8.1.3 Data of all the column values in the row with index label 10

In [18]:
```
data.loc[10]
```

Out[18]:
```
Product_ID                 1088
Sale_Date            2023-11-16
Sales_Rep                   Eve
Region                    North
Sales_Amount             8518.45
Quantity_Sold                 13
Product_Category       Furniture
Unit_Cost                2440.11
Unit_Price                2517.6
Customer_Type                New
Discount                    0.23
Payment_Method     Bank Transfer
Sales_Channel             Retail
Region_and_Sales_Rep   North-Eve
Name: 10, dtype: object
```

3.8.1.4 Data of all column values in the Multiple rows with index label 5,10,15,20

In [19]:
```
data.loc[[5, 10, 15, 20]]
```

Out[19]:

	Product_ID	Sale_Date	Sales_Rep	Region	Sales_Amount	Quantity_Sold	Product_C
5	1021	2023-02-11	Charlie	West	3761.15	32	
10	1088	2023-11-16	Eve	North	8518.45	13	F
15	1053	2023-10-16	Bob	North	2235.83	48	F
20	1002	2023-04-22	Eve	North	6551.23	9	Ele

3.8.1.5 Data of the rows with index labels from 0 to 29 using range

In [21]:
```
data.loc[[x for x in range(30)]]
```

Out[21]:

	Product_ID	Sale_Date	Sales_Rep	Region	Sales_Amount	Quantity_Sold	Product_C
0	1052	2023-02-03	Bob	North	5053.97	18	F
1	1093	2023-04-21	Bob	West	4384.02	17	F
2	1015	2023-09-21	David	South	4631.23	30	
3	1072	2023-08-24	Bob	South	2167.94	39	
4	1061	2023-03-24	Charlie	East	3750.20	13	Ele
5	1021	2023-02-11	Charlie	West	3761.15	32	
6	1083	2023-04-11	Bob	West	618.31	29	F
7	1087	2023-01-06	Eve	South	7698.92	46	F
8	1075	2023-06-29	David	South	4223.39	30	F
9	1075	2023-10-09	Charlie	West	8239.58	18	
10	1088	2023-11-16	Eve	North	8518.45	13	F
11	1100	2023-08-14	Bob	West	2198.74	43	
12	1024	2023-11-11	Eve	West	6607.80	21	
13	1003	2023-12-31	Alice	South	4775.59	30	F
14	1022	2023-08-17	Charlie	South	8813.55	21	
15	1053	2023-10-16	Bob	North	2235.83	48	F
16	1002	2023-05-30	David	North	6810.35	17	F
17	1088	2023-10-04	Bob	East	6116.75	40	Ele
18	1030	2023-07-17	David	West	3023.48	19	
19	1038	2023-03-11	Bob	South	1452.35	15	
20	1002	2023-04-22	Eve	North	6551.23	9	Ele

	Product_ID	Sale_Date	Sales_Rep	Region	Sales_Amount	Quantity_Sold	Product_C
21	1064	2023-01-04	Eve	East	7412.11	10	Ele
22	1060	2023-12-16	Eve	East	3224.71	44	(
23	1021	2023-11-27	Alice	South	6483.84	31	F
24	1033	2023-11-14	David	South	4011.80	23	F
25	1076	2023-12-16	Eve	East	7160.75	30	(
26	1058	2023-04-05	Alice	North	2072.23	33	F
27	1022	2023-06-01	David	East	8913.13	9	Ele
28	1089	2023-11-07	Bob	West	2945.36	47	F
29	1049	2023-05-17	Alice	West	3741.08	1	F

In [ ]:

3.8.2.1 Accessing the data using "slice method" with "iloc Function"

[:5,:5]-->>Data of 5 rows and 5 columns

In [22]:
```
data.iloc[:5,:5]
```

Out[22]:

	Product_ID	Sale_Date	Sales_Rep	Region	Sales_Amount
0	1052	2023-02-03	Bob	North	5053.97
1	1093	2023-04-21	Bob	West	4384.02
2	1015	2023-09-21	David	South	4631.23
3	1072	2023-08-24	Bob	South	2167.94
4	1061	2023-03-24	Charlie	East	3750.20

3.8.2.2 Accessing the data using "slice method" with "iloc Function" and providing the "range"

In [24]:
```
data.iloc[8:12,:5]
```

Out[24]:

	Product_ID	Sale_Date	Sales_Rep	Region	Sales_Amount
8	1075	2023-06-29	David	South	4223.39
9	1075	2023-10-09	Charlie	West	8239.58
10	1088	2023-11-16	Eve	North	8518.45
11	1100	2023-08-14	Bob	West	2198.74

In [25]:
```
data.iloc[8:13,3:7]
```

Out[25]:

	Region	Sales_Amount	Quantity_Sold	Product_Category
8	South	4223.39	30	Furniture
9	West	8239.58	18	Clothing
10	North	8518.45	13	Furniture
11	West	2198.74	43	Food
12	West	6607.80	21	Food

3.8.3.1 Accessing the data using "Filter Method"

In [27]:
```
data[data["Region"]=="West"]
```

Out[27]:

	Product_ID	Sale_Date	Sales_Rep	Region	Sales_Amount	Quantity_Sold	Product_
1	1093	2023-04-21	Bob	West	4384.02	17	
5	1021	2023-02-11	Charlie	West	3761.15	32	
6	1083	2023-04-11	Bob	West	618.31	29	
9	1075	2023-10-09	Charlie	West	8239.58	18	
11	1100	2023-08-14	Bob	West	2198.74	43	
...	...	...	...	...	...	...	
980	1089	2023-11-28	David	West	8719.62	8	
983	1039	2023-08-01	Eve	West	7527.63	36	
989	1025	2023-12-30	David	West	9215.32	28	E
992	1084	2023-02-19	David	West	2154.66	35	
998	1100	2023-12-20	David	West	1629.47	39	E

244 rows × 14 columns

In [28]:
```
data[data["Quantity_Sold"]>30]
```

Out[28]:

	Product_ID	Sale_Date	Sales_Rep	Region	Sales_Amount	Quantity_Sold	Product_
3	1072	2023-08-24	Bob	South	2167.94	39	
5	1021	2023-02-11	Charlie	West	3761.15	32	
7	1087	2023-01-06	Eve	South	7698.92	46	
11	1100	2023-08-14	Bob	West	2198.74	43	
15	1053	2023-10-16	Bob	North	2235.83	48	
...	...	...	...	...	...	...	
993	1025	2023-06-26	David	North	2457.65	47	
994	1068	2023-04-06	Alice	South	9093.50	31	
996	1067	2023-09-07	Bob	North	4716.36	37	
998	1100	2023-12-20	David	West	1629.47	39	E
999	1086	2023-08-16	Alice	East	4923.93	48	

393 rows × 14 columns

4. Cleaning the data

4.1Checking missing values

In [32]:
```
print(data.isnull().sum())
```

```
Product_ID             0
Sale_Date              0
Sales_Rep              0
Region                 0
Sales_Amount           0
Quantity_Sold          0
Product_Category       0
Unit_Cost              0
Unit_Price             0
Customer_Type          0
Discount               0
Payment_Method         0
Sales_Channel          0
Region_and_Sales_Rep   0
dtype: int64
```

4.2 Check Missing Values Row-by-Row

In [35]:
```
missing=data.isnull().sum(axis=1)
print(missing)
print("Rows with missing values are: ",(missing>0).sum())
```

```
0      0
1      0
2      0
3      0
4      0
      ..
995    0
996    0
997    0
998    0
999    0
Length: 1000, dtype: int64
Rows with missing values are:  0
```

In [37]:
```
data[missing>0]
```

Out[37]:

Product_ID	Sale_Date	Sales_Rep	Region	Sales_Amount	Quantity_Sold	Product_Cate

4.3 Checking the percentage of Missing Values

In [40]:
```
data.isnull().sum(axis=1).sort_values(ascending=False)/data.shape[1]*100
```

Out[40]:
```
0      0.0
671    0.0
658    0.0
659    0.0
660    0.0
      ...
338    0.0
339    0.0
340    0.0
341    0.0
999    0.0
Length: 1000, dtype: float64
```

4.4 Checking Missing Values column wise

In [43]:
```
data.isnull().sum(axis=0)
```

Out[43]:
```
Product_ID              0
Sale_Date               0
Sales_Rep               0
Region                  0
Sales_Amount            0
Quantity_Sold           0
Product_Category        0
Unit_Cost               0
Unit_Price              0
Customer_Type           0
Discount                0
Payment_Method          0
Sales_Channel           0
Region_and_Sales_Rep    0
dtype: int64
```

4.5 Checking Missing Values percentage

In [44]:
```
data.isnull().sum(axis=0)/data.shape[0]*100
```

Out[44]:
```
Product_ID              0.0
Sale_Date               0.0
Sales_Rep               0.0
Region                  0.0
Sales_Amount            0.0
Quantity_Sold           0.0
Product_Category        0.0
Unit_Cost               0.0
Unit_Price              0.0
Customer_Type           0.0
Discount                0.0
Payment_Method          0.0
Sales_Channel           0.0
Region_and_Sales_Rep    0.0
dtype: float64
```

5. Data Visualization

Using Matplot Library which includes:

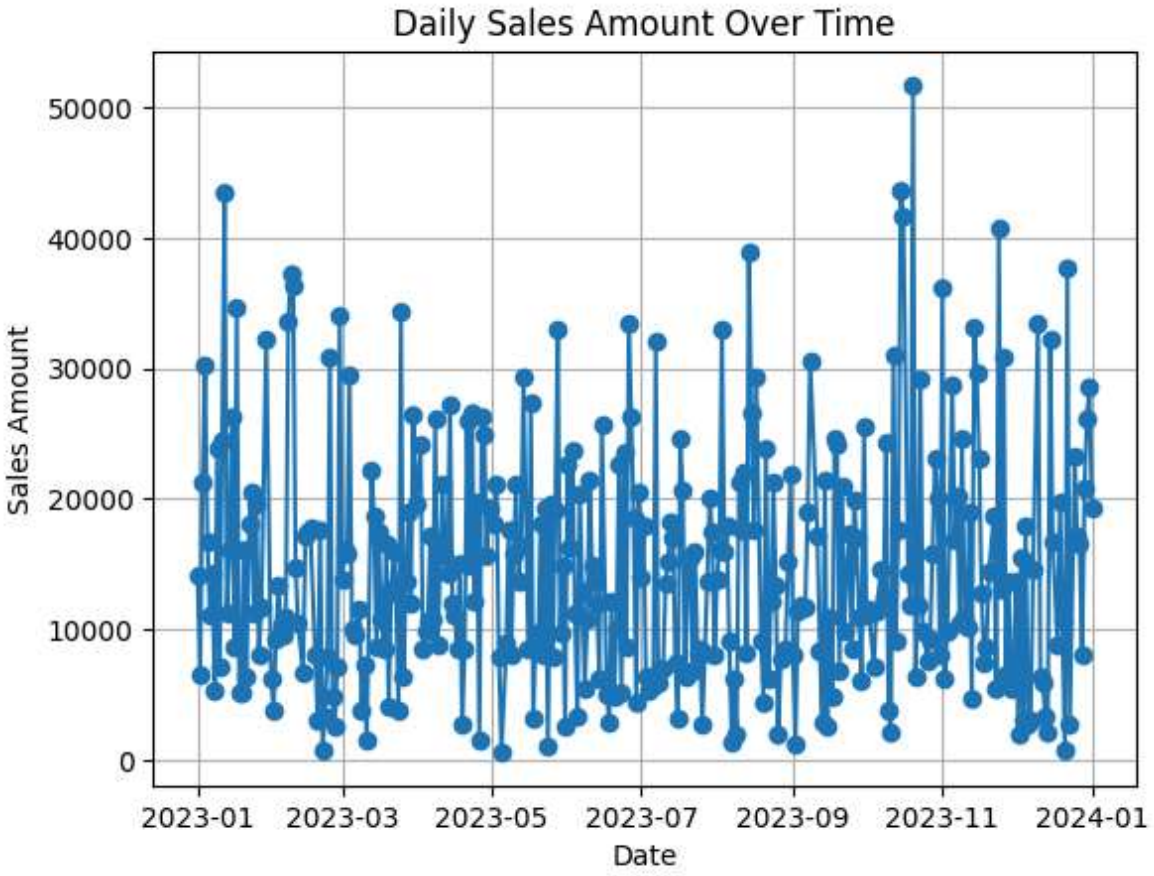
(i) Line plot
(ii) Scatter Plot
(iii) Bar Graph
(iv) Pie Plot
(v) Area Plot
(vi) Histogram

In [46]:
```
import matplotlib.pyplot as plt
```

5.1 Line Plot — Trends over Time

Use Case: Total sales amount over time.

In [50]:
```
data['Sale_Date'] = pd.to_datetime(data['Sale_Date'])
daily_sales = data.groupby('Sale_Date')['Sales_Amount'].sum()

plt.plot(daily_sales.index, daily_sales.values, marker='o')
plt.title('Daily Sales Amount Over Time')
plt.xlabel('Date')
plt.ylabel('Sales Amount')
plt.grid(True)
plt.show()
```



5.2 Scatter Plot — Relationship Between Variables

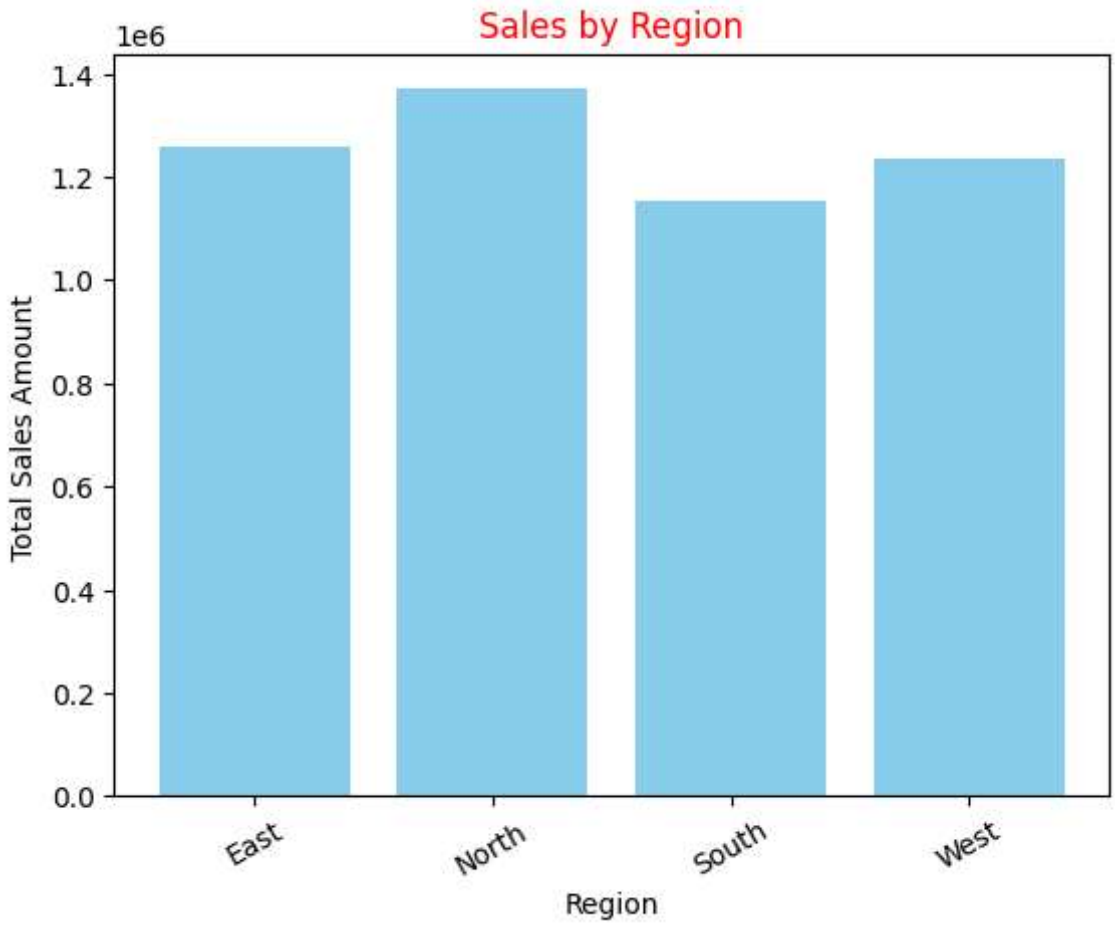
Use Case: Correlation between Unit Price and Quantity Sold.

In [66]:
```
plt.scatter(data['Unit_Price'], data['Quantity_Sold'], alpha=1)
plt.title('Unit Price vs Quantity Sold',color='red')
plt.xlabel('Unit Price')
plt.ylabel('Quantity Sold')
plt.show()
```

5.3 Bar Graph — Category-wise Comparisons

Use Case: Total sales across regions or product categories.

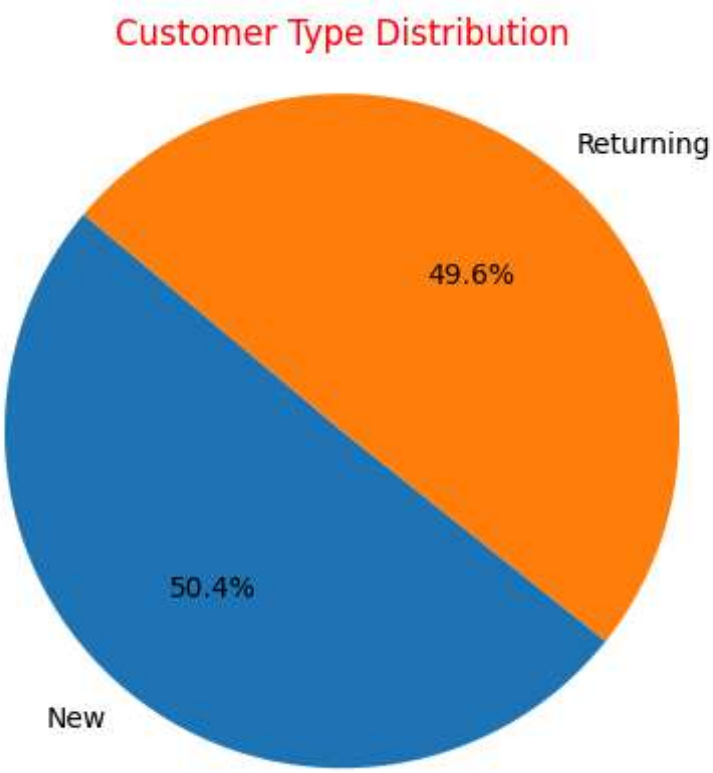
In [71]:
```
region_sales = data.groupby('Region')['Sales_Amount'].sum()

plt.bar(region_sales.index, region_sales.values, color='skyblue')
plt.title('Sales by Region',color='red')
plt.xlabel('Region')
plt.ylabel('Total Sales Amount')
plt.xticks(rotation=30)
plt.show()
```



5.4 Pie Plot — Proportional Breakdown

Use Case: Proportion of customer types in total sales.

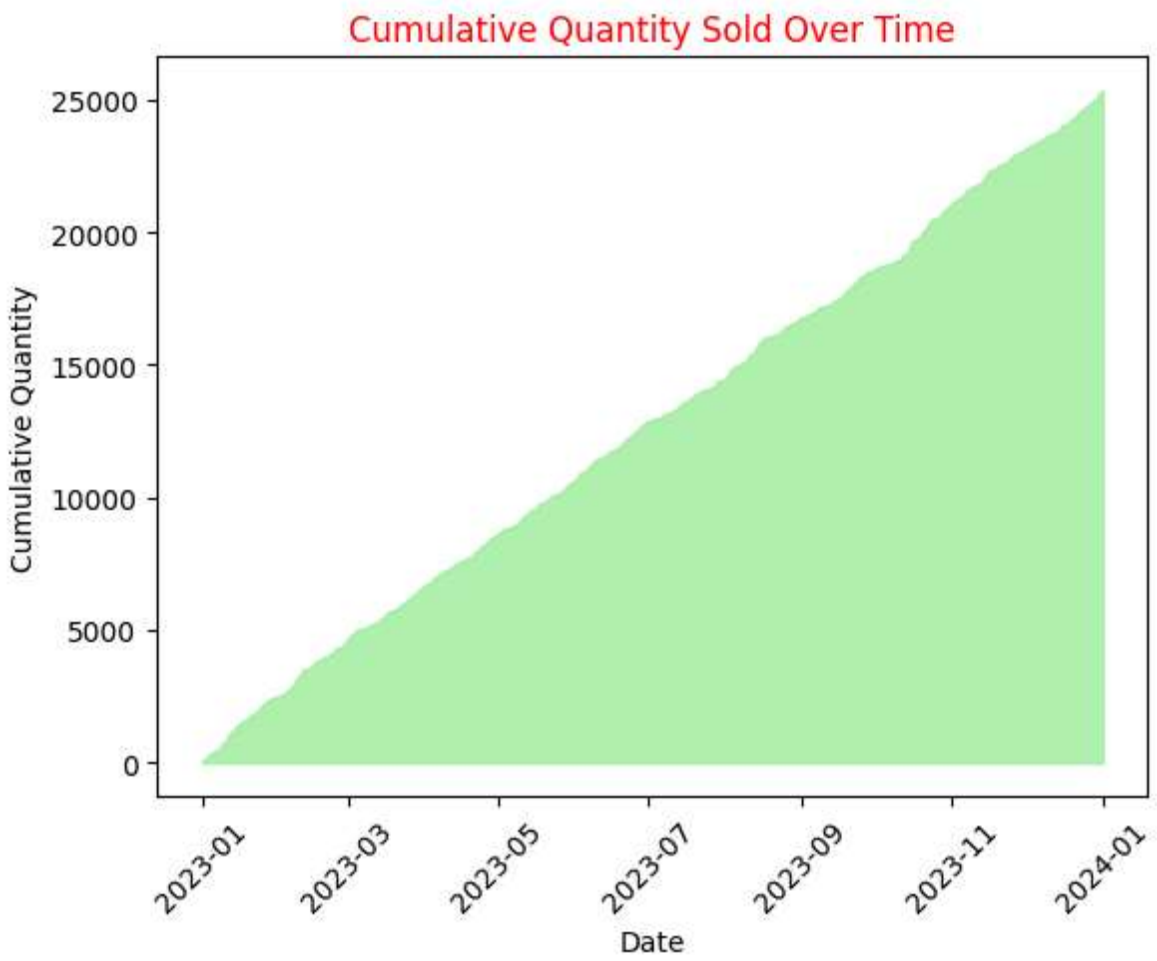
In [74]:
```
customer_counts = data['Customer_Type'].value_counts()

plt.pie(customer_counts.values, labels=customer_counts.index, autopct='%1.1f%%',
plt.title('Customer Type Distribution',color="red")
plt.axis('equal')
plt.show()
```



5.5 Area Plot — Cumulative Distribution Over Time

Use Case: Growth of cumulative quantity sold.

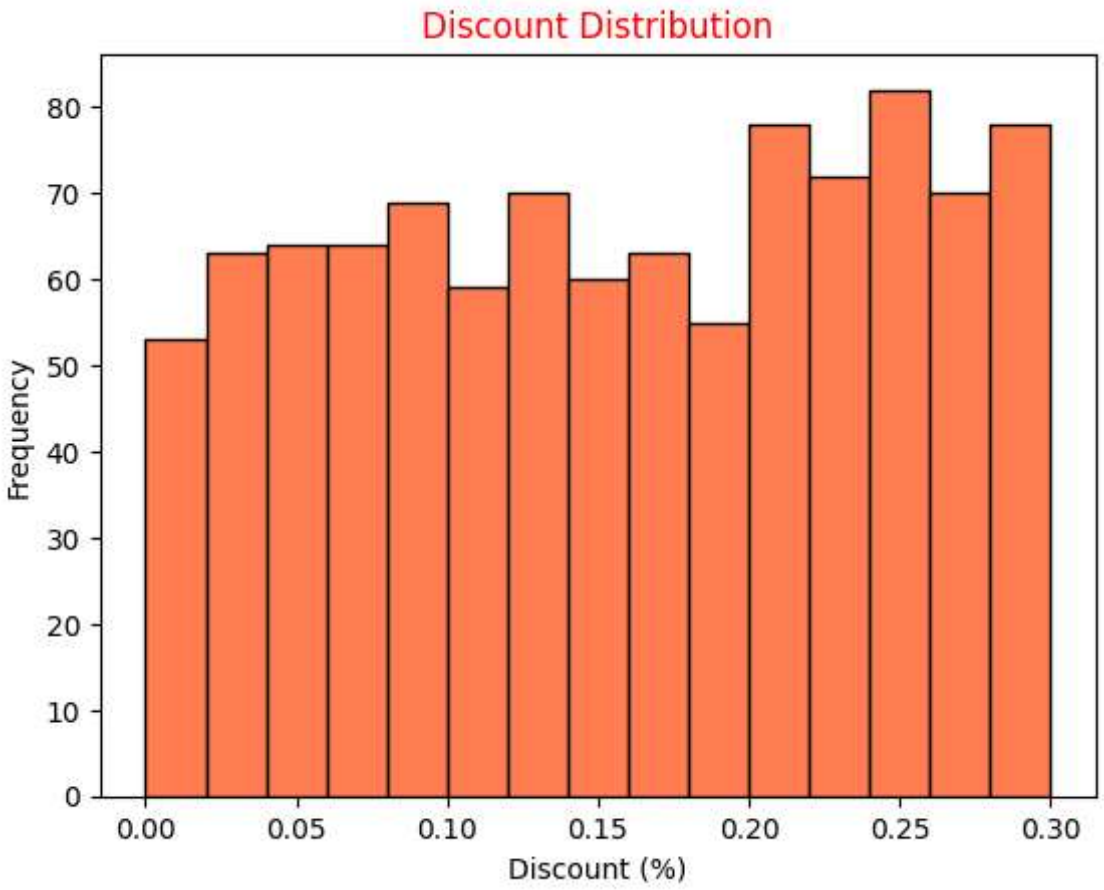
In [77]:
```
data['Sale_Date'] = pd.to_datetime(data['Sale_Date'])
daily_quantity = data.groupby('Sale_Date')['Quantity_Sold'].sum().cumsum()

plt.fill_between(daily_quantity.index, daily_quantity.values, color='lightgreen'
plt.title('Cumulative Quantity Sold Over Time',color="red")
plt.xlabel('Date')
plt.ylabel('Cumulative Quantity')
plt.xticks(rotation=45)
plt.show()
```



5.6 Histogram — Distribution of Numerical Values

Use Case: Distribution of discounts offered

In [82]:
```
plt.hist(data['Discount'], bins=15, color='coral', edgecolor='black')
plt.title('Discount Distribution',color='red')
plt.xlabel('Discount (%)')
plt.ylabel('Frequency')
plt.show()
```



6. Data Analysis

Types of Data Analysis:

1. Univariate Analysis:- Analyzing one variable at a time
2. Bivariate Analysis:- Exploring relationship between two variables
3. Multivariate Analysis:- Studying interactions between three or more variables

6.1 Univariate Analysis:

6.1.1 Analyzing Numerical Data

In [154…

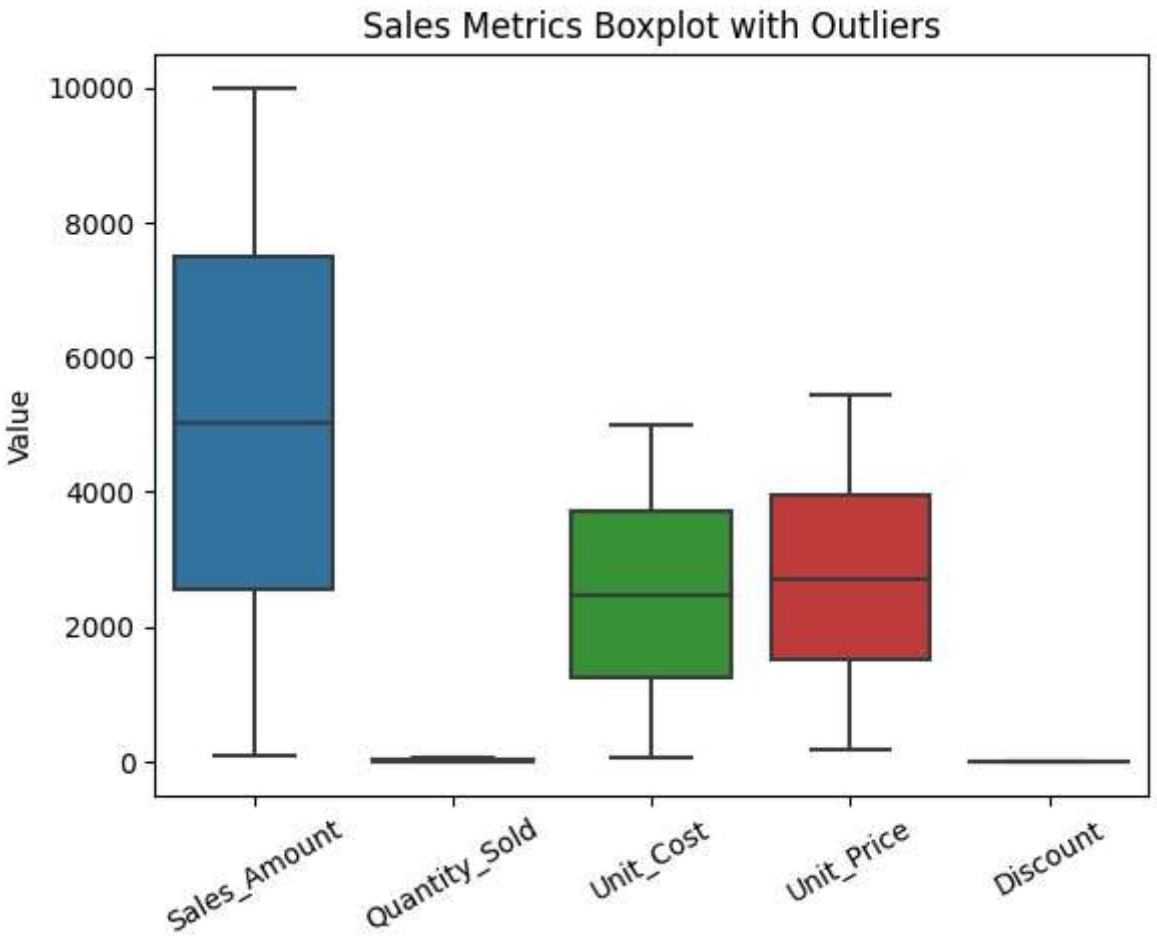
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 14 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   Product_ID            1000 non-null   int64
 1   Sale_Date             1000 non-null   datetime64[ns]
 2   Sales_Rep             1000 non-null   object
 3   Region                1000 non-null   object
 4   Sales_Amount          1000 non-null   float64
 5   Quantity_Sold         1000 non-null   int64
 6   Product_Category      1000 non-null   object
 7   Unit_Cost             1000 non-null   float64
 8   Unit_Price            1000 non-null   float64
 9   Customer_Type         1000 non-null   object
 10  Discount              1000 non-null   float64
 11  Payment_Method        1000 non-null   object
 12  Sales_Channel         1000 non-null   object
 13  Region_and_Sales_Rep  1000 non-null   object
dtypes: datetime64[ns](1), float64(4), int64(2), object(7)
memory usage: 109.5+ KB
```

6.1.2 Checking for any available Outliers

In [155…

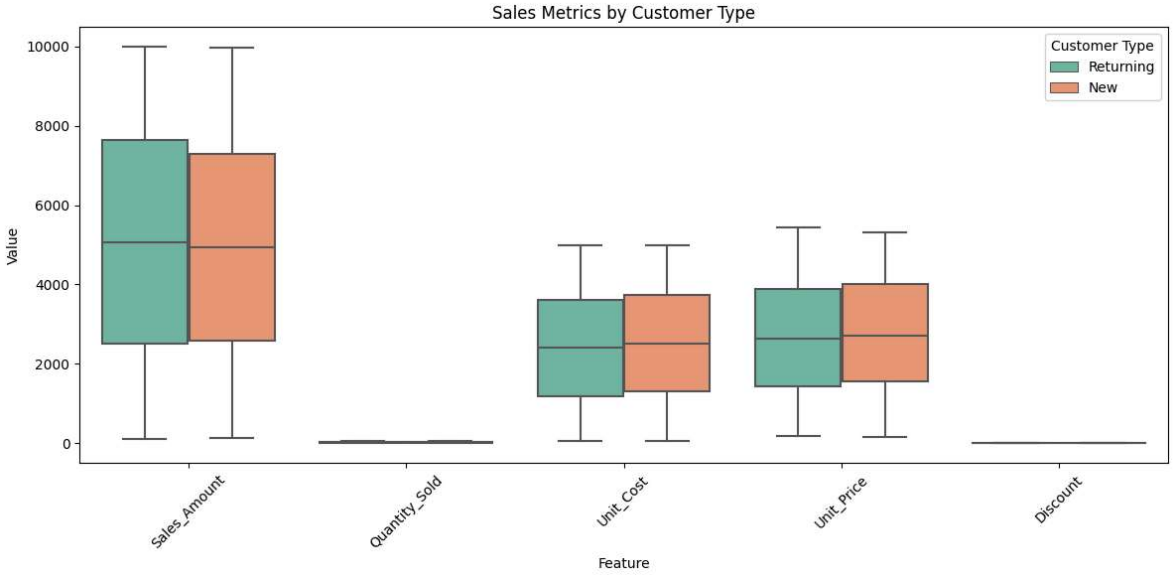
```python
cols = ["Sales_Amount", "Quantity_Sold", "Unit_Cost", "Unit_Price", "Discount"]

# Creating boxplot
sns.boxplot(data=data[cols])
plt.title("Sales Metrics Boxplot with Outliers")
plt.xticks(rotation=30)
plt.ylabel("Value")
plt.show()
```



Sales Metrics By Customer Type

In [156…

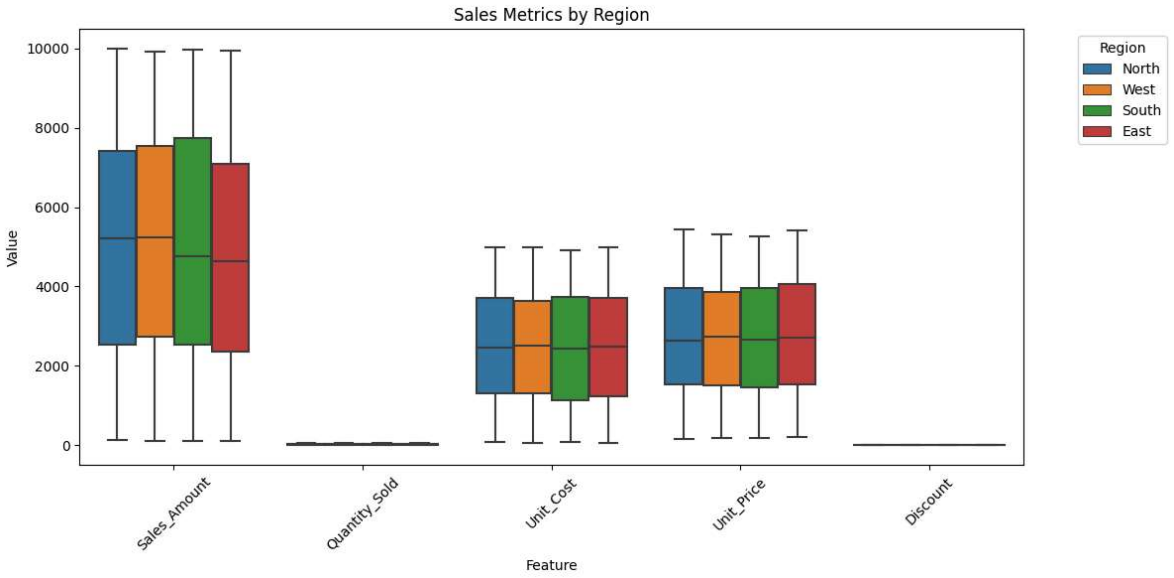
```
# Melting the dataset to a long format
melted = data.melt(id_vars='Customer_Type', value_vars=["Sales_Amount", "Quantit
                   var_name='Feature', value_name='Value')

# Creating boxplot colored by Customer_Type
plt.figure(figsize=(12,6))
sns.boxplot(x='Feature', y='Value', hue='Customer_Type', data=melted, palette='S
plt.title("Sales Metrics by Customer Type")
plt.xticks(rotation=45)
plt.ylabel("Value")
plt.legend(title='Customer Type')
plt.tight_layout()
plt.show()
```



Sales Metrics By Region

In [157…

```
# Melt for region grouping
melted_region = data.melt(id_vars='Region', value_vars=["Sales_Amount", "Quantit
                          var_name='Feature', value_name='Value')

# Boxplot by Region
plt.figure(figsize=(12,6))
sns.boxplot(x='Feature', y='Value', hue='Region', data=melted_region, palette='t
plt.title("Sales Metrics by Region")
plt.xticks(rotation=45)
plt.ylabel("Value")
plt.legend(title='Region', bbox_to_anchor=(1.05, 1), loc='upper left')
plt.tight_layout()
plt.show()
```



6.1.3 Finding Mean() And Median() of Numerical Data

In [158…

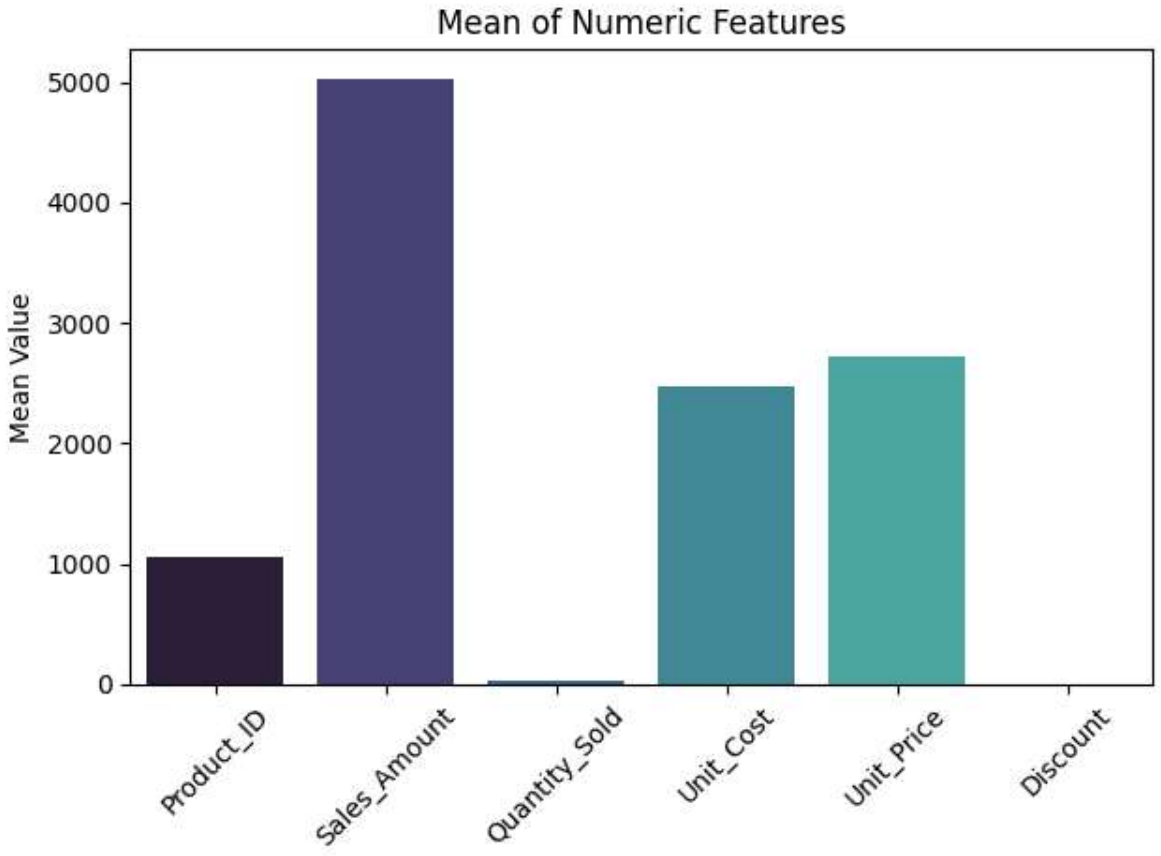
```
numeric_means = data.mean(numeric_only=True)
print("Mean of Numerical data:\n")
print(numeric_means)
```

```
Mean of Numerical data:

Product_ID       1050.12800
Sales_Amount     5019.26523
Quantity_Sold      25.35500
Unit_Cost        2475.30455
Unit_Price       2728.44012
Discount            0.15239
dtype: float64
```

In [159…
```
numeric_medians = data.median(numeric_only=True)
print("\nMedian of Numeric Features:\n")
print(numeric_medians)
```

```
Median of Numeric Features:

Product_ID       1051.000
Sales_Amount     5019.300
Quantity_Sold      25.000
Unit_Cost        2467.235
Unit_Price       2696.400
Discount            0.150
dtype: float64
```

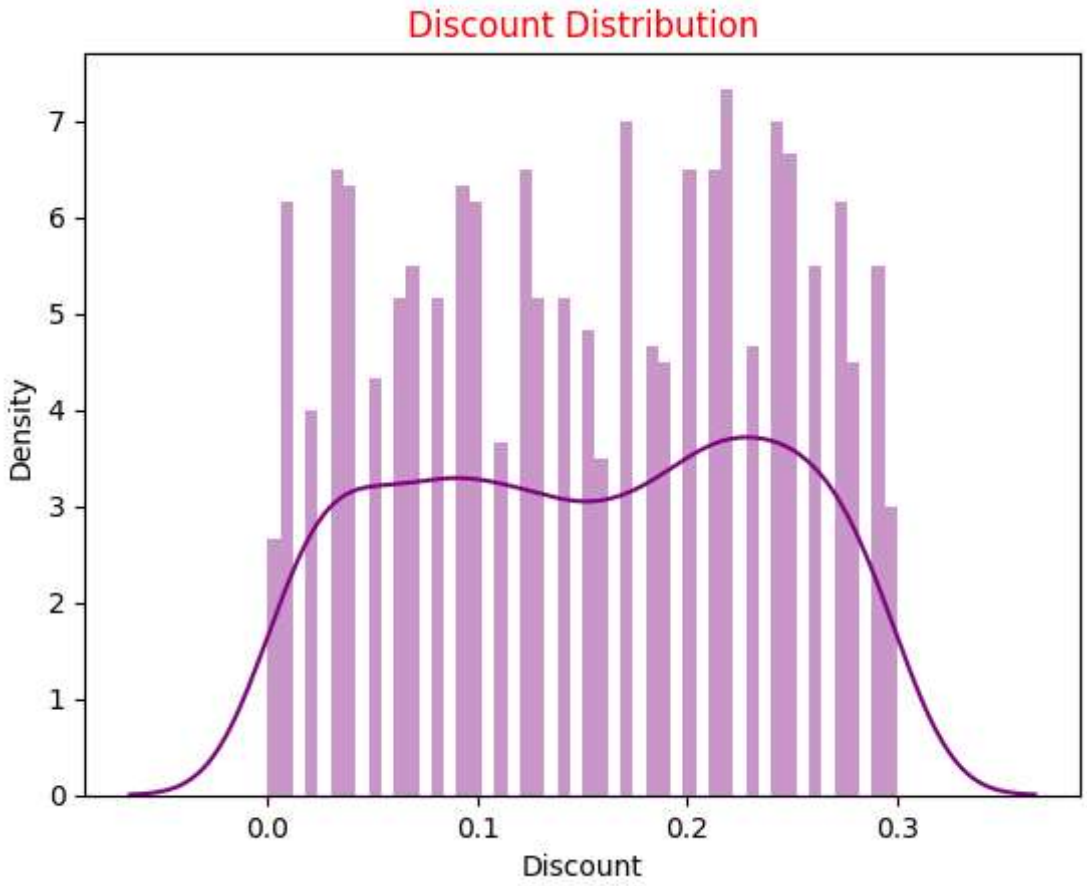
In [160…
```
sns.barplot(x=numeric_means.index, y=numeric_means.values, palette="mako")
plt.title("Mean of Numeric Features")
plt.ylabel("Mean Value")
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



In [161…
```
sns.distplot(data["Discount"], bins=50, color='purple')
plt.title('Discount Distribution', color="red")
```

```
C:\Users\Firdaush Alam\AppData\Local\Temp\ipykernel_15084\2938313584.py:1: UserWa
rning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(data["Discount"], bins=50, color='purple')
```

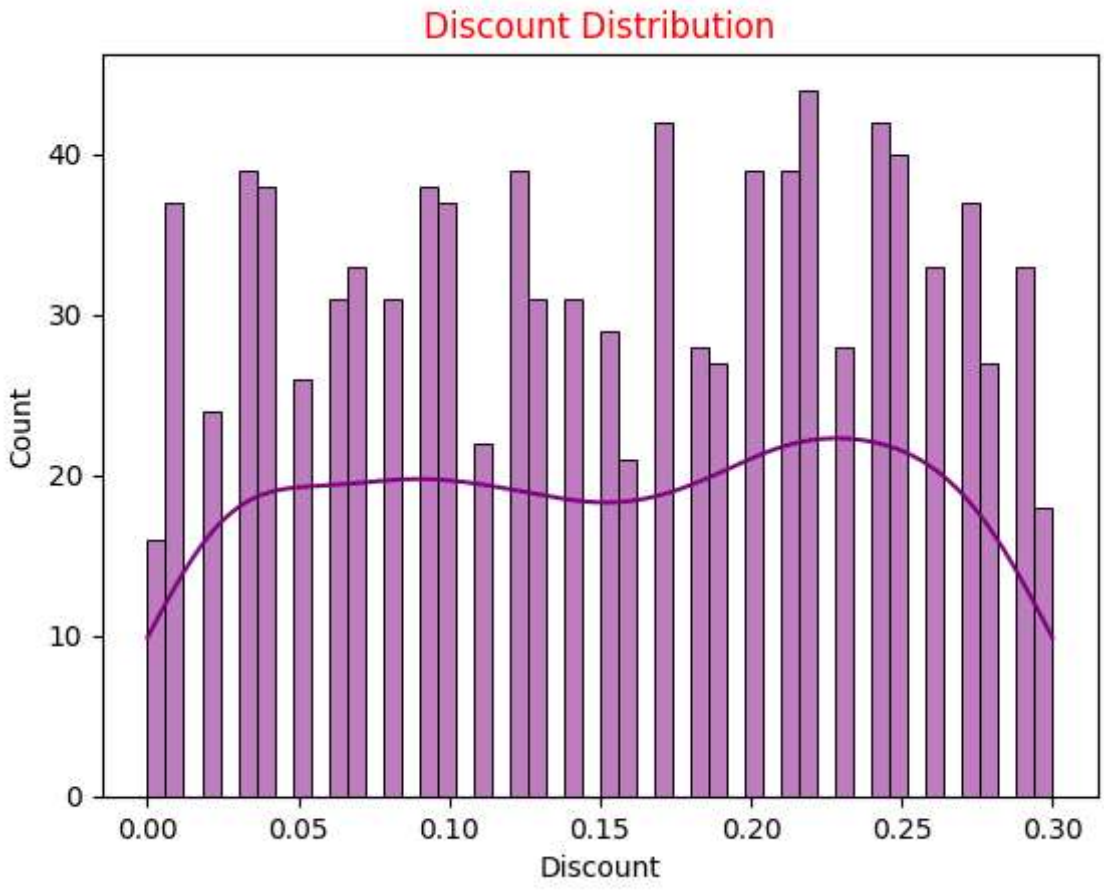
Out[161]: Text(0.5, 1.0, 'Discount Distribution')



In [162…

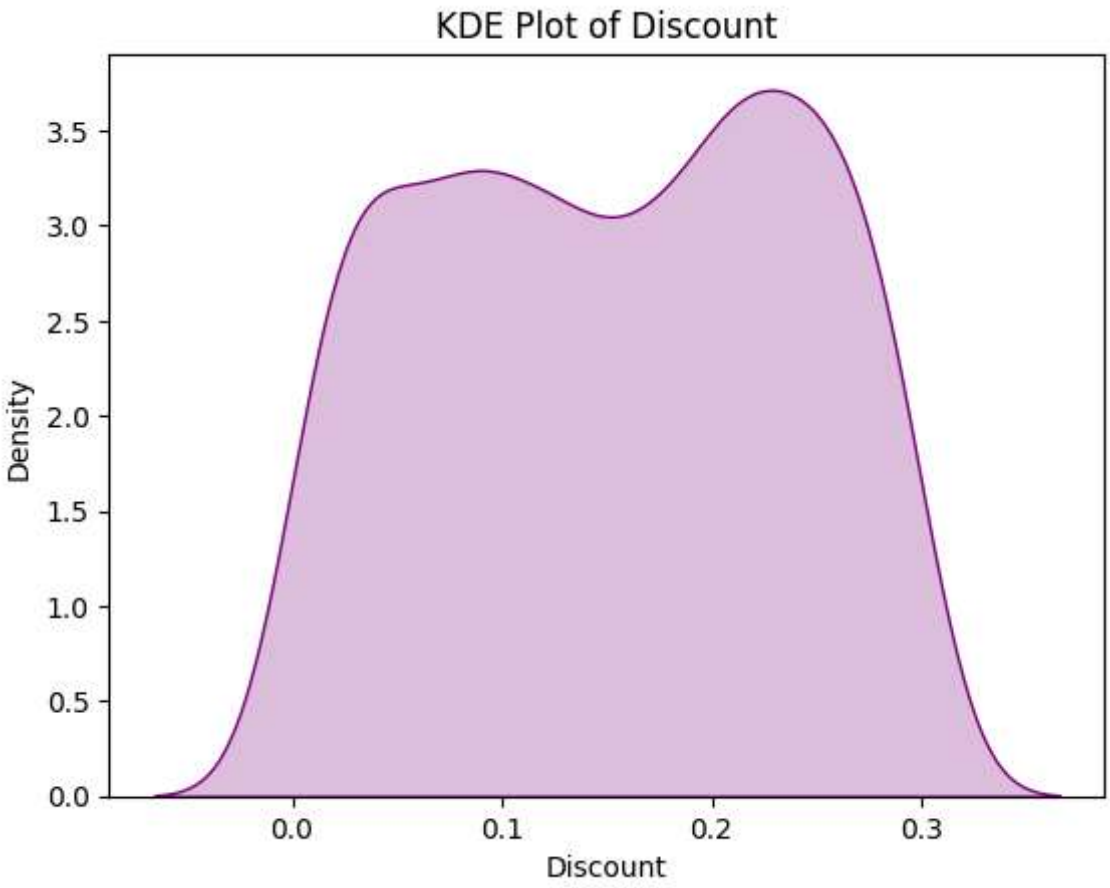
```
sns.histplot(data['Discount'], bins=50,kde=True, color='purple')
plt.title('Discount Distribution', color="red")
```

Out[162]: Text(0.5, 1.0, 'Discount Distribution')



In [163…

```
sns.kdeplot(data['Discount'], color='purple', fill=True)
plt.title('KDE Plot of Discount')
plt.xlabel('Discount')
plt.ylabel('Density')
plt.show()
```



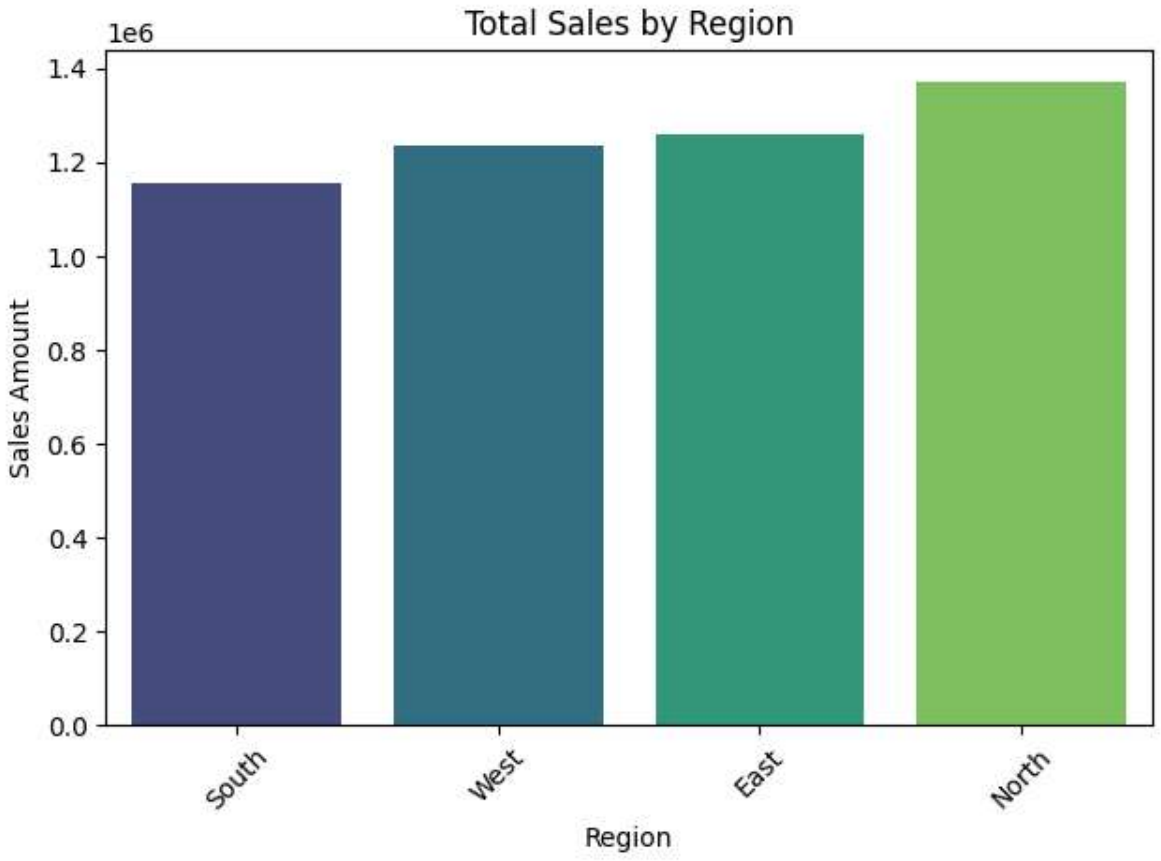
6.1.4 Analyzing Categorical Data:

In [164…
```
pd.crosstab(data['Region'], data['Sales_Channel'])
```

Out[164]:

Sales_Channel	Online	Retail
Region		
East	121	142
North	130	137
South	112	114
West	125	119

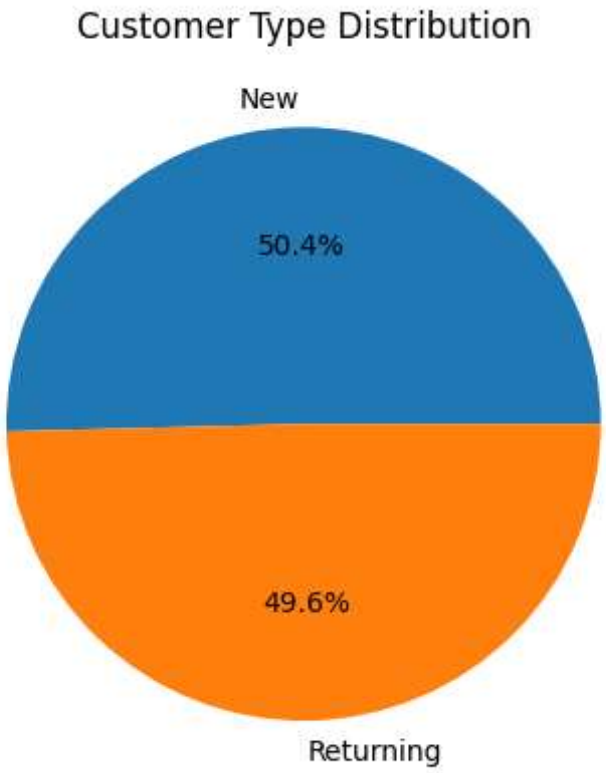
In [165…
```
# Example: Total Sales by Region
region_sales = data.groupby('Region')['Sales_Amount'].sum().sort_values()
sns.barplot(x=region_sales.index, y=region_sales.values, palette='viridis')
plt.title("Total Sales by Region")
plt.ylabel("Sales Amount")
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



In [166…
```
data.groupby('Customer_Type')['Unit_Price'].mean()
data.groupby('Region')['Discount'].median()
```

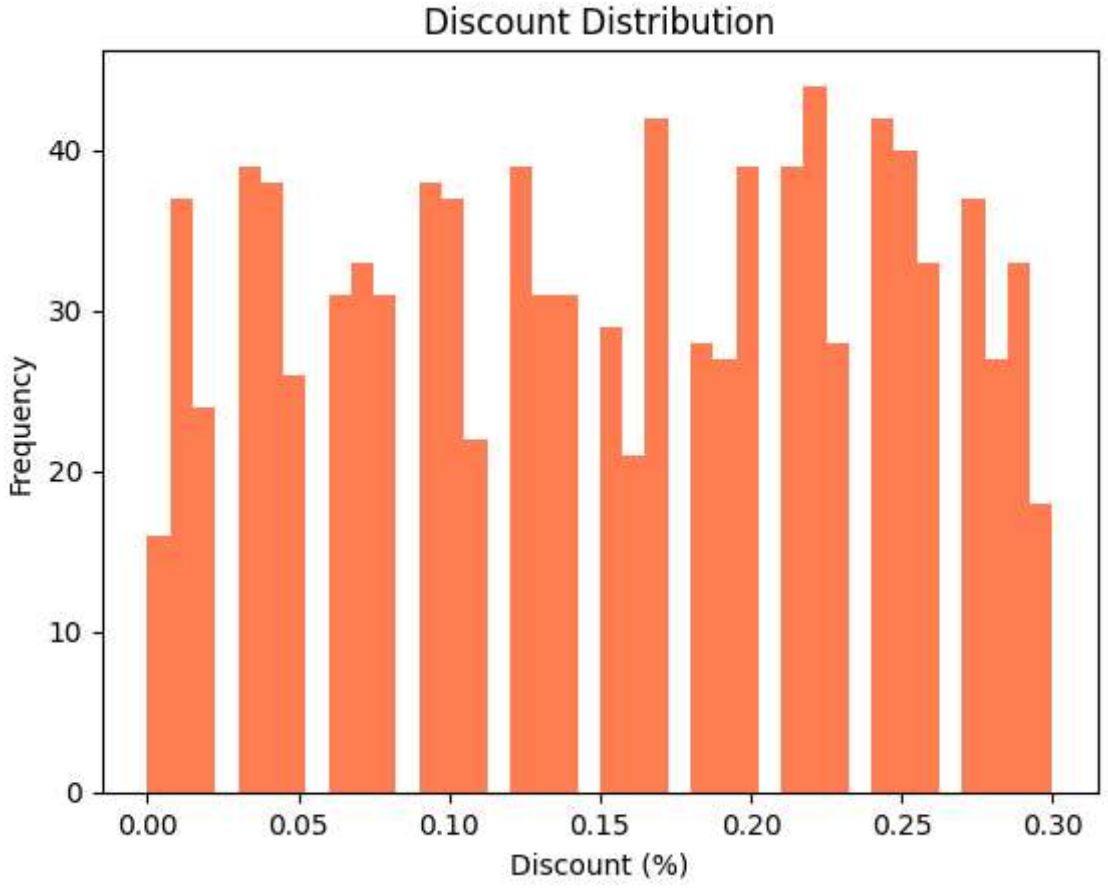
Out[166]:
```
Region
East     0.17
North    0.15
South    0.15
West     0.15
Name: Discount, dtype: float64
```

In [167…
```
# Pie chart for Customer Type
customer_counts = data['Customer_Type'].value_counts()
plt.pie(customer_counts, labels=customer_counts.index, autopct='%1.1f%%')
plt.title('Customer Type Distribution')
plt.show()
```



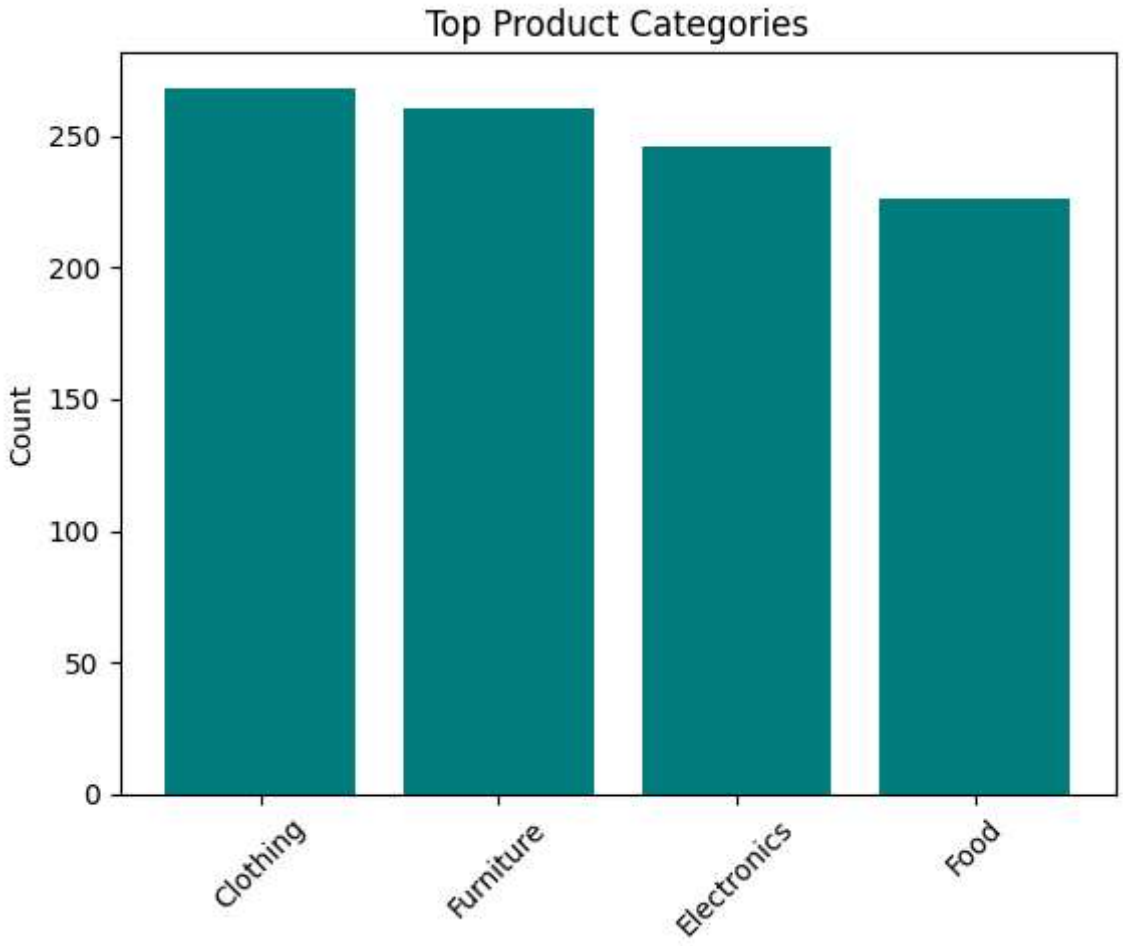
In [168…

```
# Histogram of Discount
plt.hist(data['Discount'], bins=40, color='coral')
plt.title('Discount Distribution')
plt.xlabel('Discount (%)')
plt.ylabel('Frequency')
plt.show()
```



In [169…

```
# Bar chart for Product Category counts
product_counts = data['Product_Category'].value_counts()
plt.bar(product_counts.index, product_counts.values, color='teal')
plt.title('Top Product Categories')
plt.ylabel('Count')
plt.xticks(rotation=45)
plt.show()
```

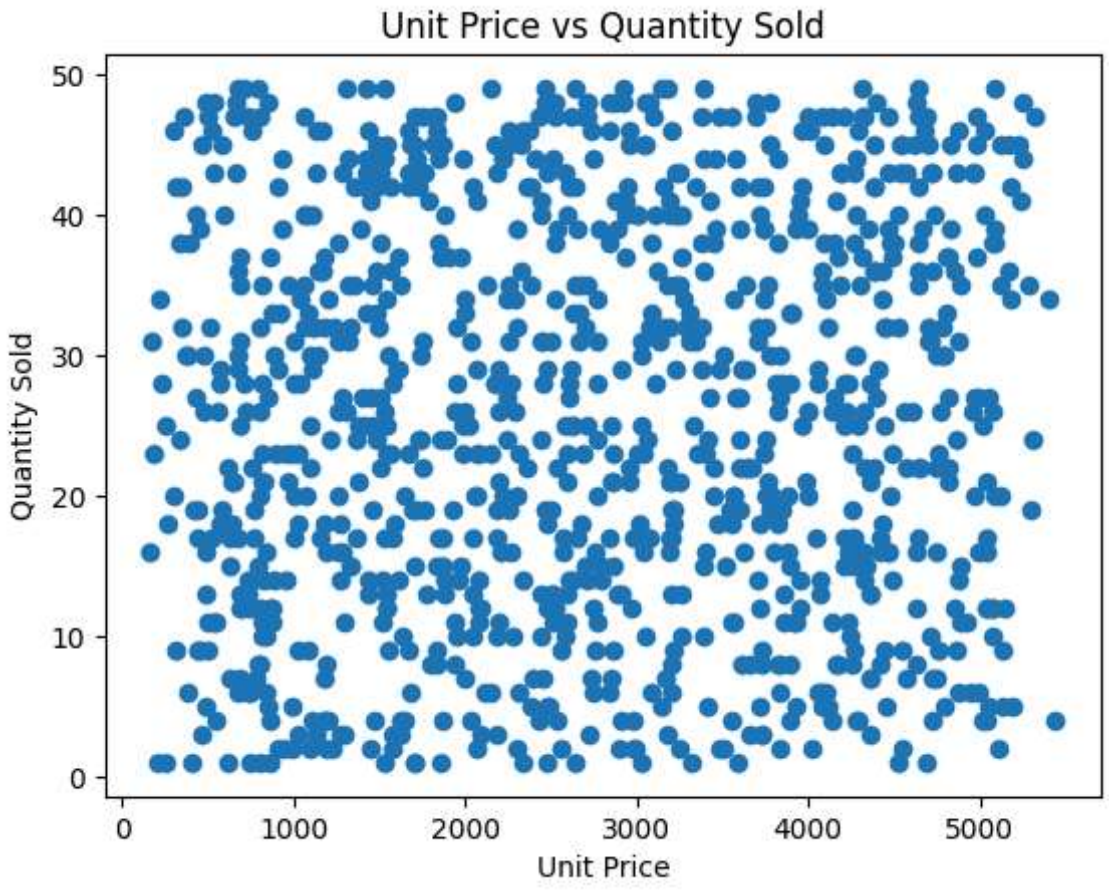


6.2 Bivariate Analysis:

6.2.1 Scatter Plot: "Quantity vs Unit Price"

In [170…

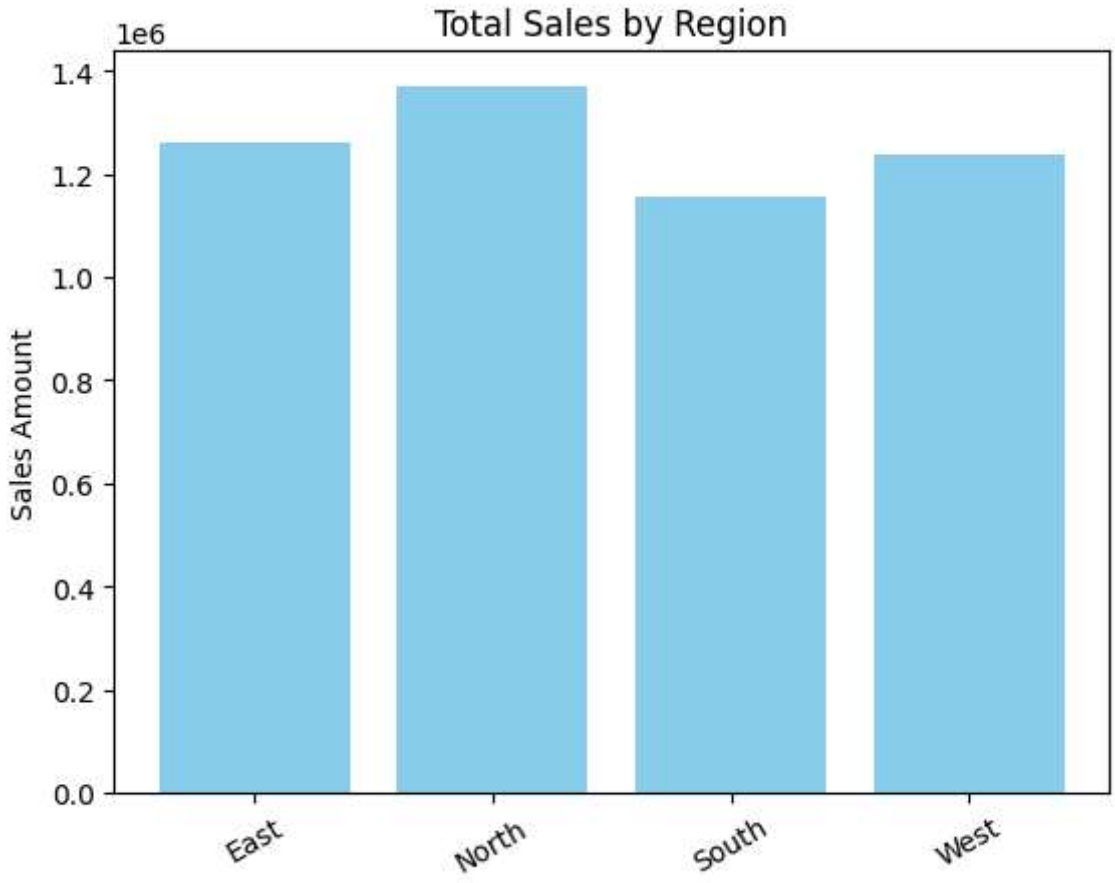
```
plt.scatter(data['Unit_Price'], data['Quantity_Sold'], alpha=1)
plt.title('Unit Price vs Quantity Sold')
plt.xlabel('Unit Price')
plt.ylabel('Quantity Sold')
plt.show()
```



6.2.2 Bar Graph: Sales Amount by "Region"

In [171…

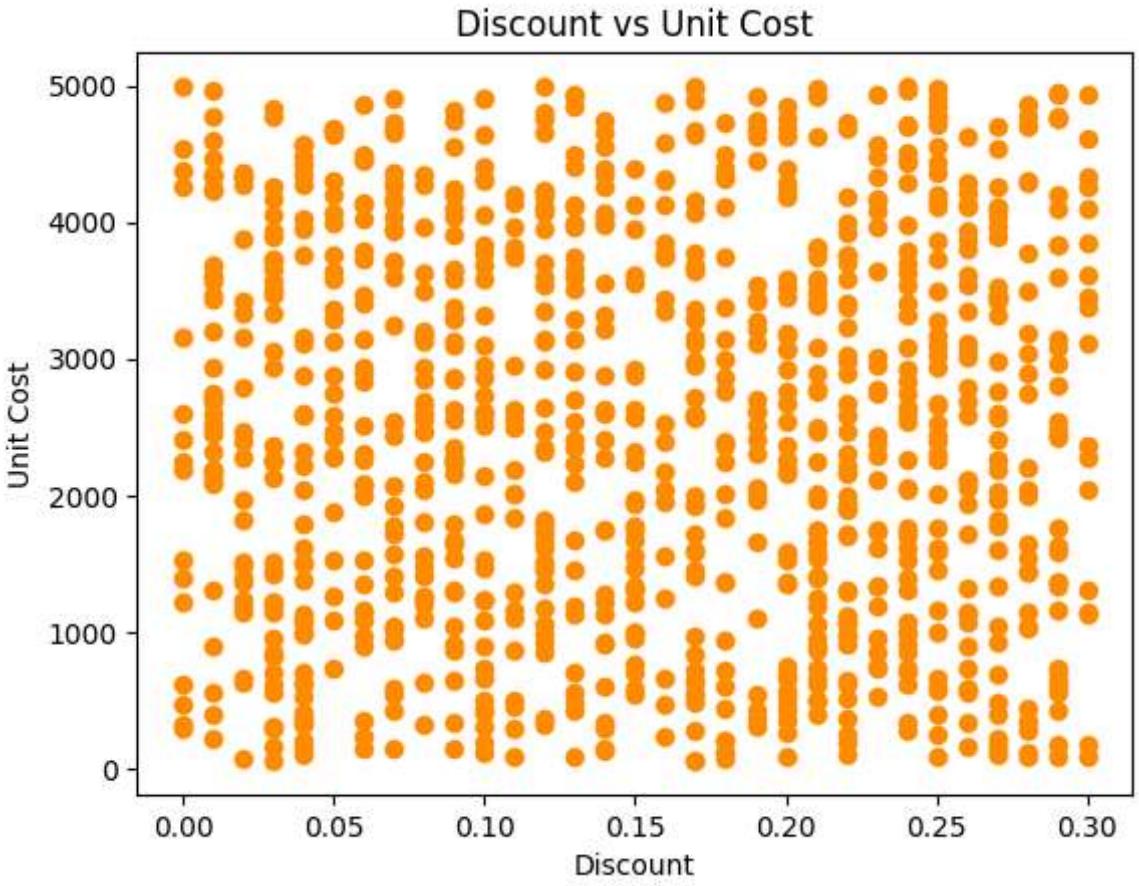
```
region_sales = data.groupby('Region')['Sales_Amount'].sum()
plt.bar(region_sales.index, region_sales.values, color='skyblue')
plt.title('Total Sales by Region')
plt.xticks(rotation=30)
plt.ylabel('Sales Amount')
plt.show()
```



6.2.3 Scatter Plot: Discount Vs Unit Cost

In [172…

```
plt.scatter(data['Discount'], data['Unit_Cost'], c='darkorange')
plt.title('Discount vs Unit Cost')
plt.xlabel('Discount')
plt.ylabel('Unit Cost')
plt.show()
```



6.3 Multivariate Analysis:

Visualize the correlation between different features

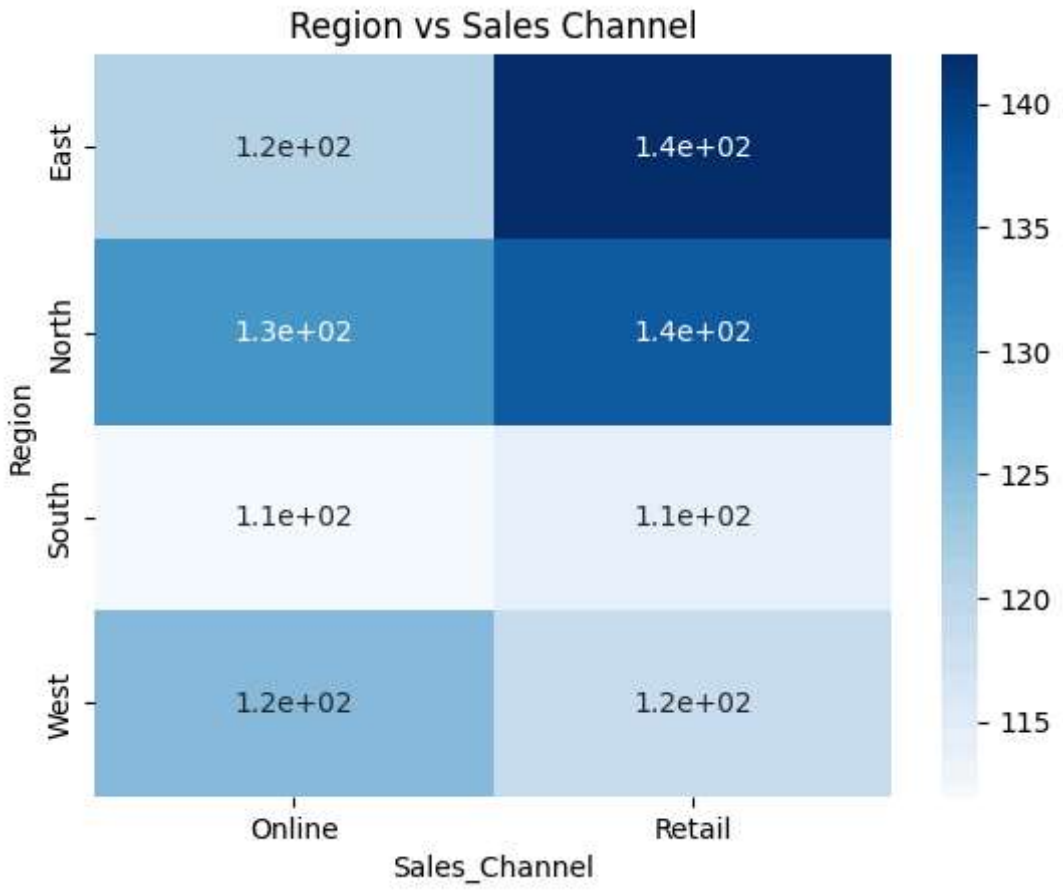
In [173…
```
import seaborn as sns
```

In [174…
```
pd.crosstab(data['Region'], data['Sales_Channel'])
```

Out[174]:

Sales_Channel	Online	Retail
Region		
East	121	142
North	130	137
South	112	114
West	125	119

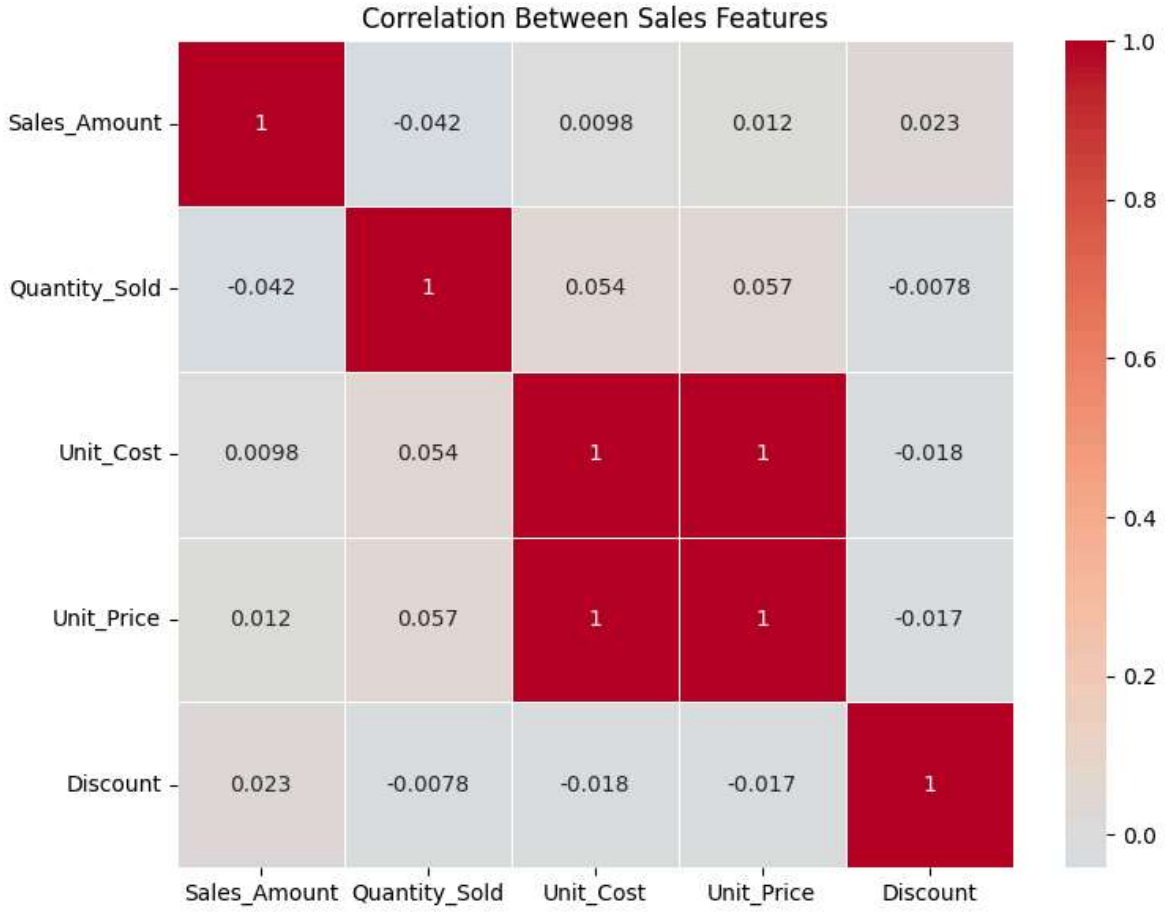
In [175…
```
cross = pd.crosstab(data['Region'], data['Sales_Channel'])
sns.heatmap(cross, annot=True, cmap='Blues')
plt.title('Region vs Sales Channel')
plt.show()
```



Heatmap()

In [176…

```
# Select relevant numerical columns
numeric_cols = ["Sales_Amount", "Quantity_Sold", "Unit_Cost", "Unit_Price", "Dis

# Compute the correlation matrix
corr_matrix = data[numeric_cols].corr()

# Create heatmap
plt.figure(figsize=(8, 6))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', center=0, linewidths=0.5,
plt.title("Correlation Between Sales Features")
plt.tight_layout()
plt.show()
```



----------The End----------

Project By: Firdaush Alam