In [71]:

```
import pandas as pd
df = pd.read_csv(r"C:\\Users\\jeeva\\Downloads\\archive (12)\\superstore_final_dataset (1).csv",encoding='latin
```

In [72]:

```
df.head()
```

Out[72]:

	Row_ID	Order_ID	Order_Date	Ship_Date	Ship_Mode	Customer_ID	Customer_Name	Segment	Country	City	State
0	1	CA-2017-152156	8/11/2017	11/11/2017	Second Class	CG-12520	Claire Gute	Consumer	United States	Henderson	Kentucky
1	2	CA-2017-152156	8/11/2017	11/11/2017	Second Class	CG-12520	Claire Gute	Consumer	United States	Henderson	Kentucky
2	3	CA-2017-138688	12/6/2017	16/06/2017	Second Class	DV-13045	Darrin Van Huff	Corporate	United States	Los Angeles	Californi
3	4	US-2016-108966	11/10/2016	18/10/2016	Standard Class	SO-20335	Sean O Donnel	Consumer	United States	Fort Lauderdale	Florida
4	5	US-2016-108966	11/10/2016	18/10/2016	Standard Class	SO-20335	Sean O Donnel	Consumer	United States	Fort Lauderdale	Florida

In [73]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9800 entries, 0 to 9799
Data columns (total 18 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Row_ID         9800 non-null   int64  
 1   Order_ID       9800 non-null   object 
 2   Order_Date     9800 non-null   object 
 3   Ship_Date      9800 non-null   object 
 4   Ship_Mode      9800 non-null   object 
 5   Customer_ID    9800 non-null   object 
 6   Customer_Name  9800 non-null   object 
 7   Segment        9800 non-null   object 
 8   Country        9800 non-null   object 
 9   City           9800 non-null   object 
 10  State          9800 non-null   object 
 11  Postal_Code    9789 non-null   float64
 12  Region         9800 non-null   object 
 13  Product_ID     9800 non-null   object 
 14  Category       9800 non-null   object 
 15  Sub_Category   9800 non-null   object 
 16  Product_Name   9800 non-null   object 
 17  Sales          9800 non-null   float64
dtypes: float64(2), int64(1), object(15)
memory usage: 1.3+ MB
```

In [74]:

```
df.columns = df.columns.str.strip().str.replace(" ", "_").str.replace("-", "_")
```

In [75]:

```
df.shape
```

Out[75]:

```
(9800, 18)
```

In [76]:

```
df.columns.tolist()
```

Out[76]:

```
['Row_ID',
 'Order_ID',
 'Order_Date',
 'Ship_Date',
 'Ship_Mode',
 'Customer_ID',
 'Customer_Name',
 'Segment',
 'Country',
 'City',
 'State',
 'Postal_Code',
 'Region',
 'Product_ID',
 'Category',
 'Sub_Category',
 'Product_Name',
 'Sales']
```

In [77]:

```
df['Order_Date'] = pd.to_datetime(df['Order_Date'], errors='coerce')
df['Ship_Date'] = pd.to_datetime(df['Ship_Date'], errors='coerce')
```

In [78]:

```
df['Year'] = df['Order_Date'].dt.year
df['Month'] = df['Order_Date'].dt.month
df['Month_Name'] = df['Order_Date'].dt.strftime('%B')
```

In [79]:

```
print("\nMissing Values:\n", df.isnull().sum())
```

```

Missing Values:
 Row_ID              0
Order_ID            0
Order_Date       5841
Ship_Date        5985
Ship_Mode           0
Customer_ID         0
Customer_Name       0
Segment             0
Country             0
City                0
State               0
Postal_Code        11
Region              0
Product_ID          0
Category            0
Sub_Category        0
Product_Name        0
Sales               0
Year             5841
Month            5841
Month_Name       5841
dtype: int64
```

In [80]:

```
df.drop_duplicates(inplace=True)
```

In [81]:

```
if 'Postal_Code' in df.columns:
    df['Postal_Code'].fillna(0, inplace=True)
```

```
C:\Users\jeeva\AppData\Local\Temp\ipykernel_11280\252986599.py:2: FutureWarning: A value is trying to be set on
a copy of a DataFrame or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on w
hich we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)'
or df[col] = df[col].method(value) instead, to perform the operation inplace on the original object.


  df['Postal_Code'].fillna(0, inplace=True)
```

In [82]:

```
print("\nData Types:\n", df.dtypes)
```

```
Data Types:
 Row_ID                    int64
Order_ID                 object
Order_Date       datetime64[ns]
Ship_Date        datetime64[ns]
Ship_Mode                object
Customer_ID              object
Customer_Name            object
Segment                  object
Country                  object
City                     object
State                    object
Postal_Code             float64
Region                   object
Product_ID               object
Category                 object
Sub_Category             object
Product_Name             object
Sales                   float64
Year                    float64
Month                   float64
Month_Name               object
dtype: object
```

In [83]:

```
print("\nDescriptive Statistics:")
print(df.describe(include='all'))
```

```
Descriptive Statistics:
             Row_ID        Order_ID                     Order_Date  \
count   9800.000000            9800                           3959   
unique          NaN            4922                            NaN   
top             NaN  CA-2018-100111                            NaN   
freq            NaN              14                            NaN   
mean    4900.500000             NaN  2017-03-14 18:19:11.199798016   
min        1.000000             NaN            2015-01-02 00:00:00   
25%     2450.750000             NaN            2016-04-05 00:00:00   
50%     4900.500000             NaN            2017-05-02 00:00:00   
75%     7350.250000             NaN            2018-03-07 00:00:00   
max     9800.000000             NaN            2018-12-11 00:00:00   
std     2829.160653             NaN                            NaN   

                            Ship_Date       Ship_Mode Customer_ID  \
count                            3815            9800        9800   
unique                            NaN               4         793   
top                               NaN  Standard Class    WB-21850   
freq                              NaN            5859          35   
mean    2017-04-09 17:04:02.516382720             NaN         NaN   
min               2015-01-04 00:00:00             NaN         NaN   
25%               2016-04-12 00:00:00             NaN         NaN   
50%               2017-06-06 00:00:00             NaN         NaN   
75%               2018-05-01 00:00:00             NaN         NaN   
max               2019-05-01 00:00:00             NaN         NaN   
std                               NaN             NaN         NaN   

        Customer_Name    Segment        Country           City  ...  \
count            9800       9800           9800           9800  ...   
unique            793          3              1            529  ...   
top     William Brown   Consumer  United States  New York City  ...   
freq               35       5101           9800            891  ...   
mean              NaN        NaN            NaN            NaN  ...   
min               NaN        NaN            NaN            NaN  ...   
25%               NaN        NaN            NaN            NaN  ...   
50%               NaN        NaN            NaN            NaN  ...   
75%               NaN        NaN            NaN            NaN  ...   
max               NaN        NaN            NaN            NaN  ...   
std               NaN        NaN            NaN            NaN  ...   

         Postal_Code  Region       Product_ID         Category Sub_Category  \
count    9800.000000    9800             9800             9800         9800   
unique           NaN       4             1861                3           17   
top              NaN    West  OFF-PA-10001970  Office Supplies      Binders   
freq             NaN    3140               19             5909         1492   
mean    55211.280918     NaN              NaN              NaN          NaN   
min         0.000000     NaN              NaN              NaN          NaN   
25%     23223.000000     NaN              NaN              NaN          NaN   
50%     57551.000000     NaN              NaN              NaN          NaN   
75%     90008.000000     NaN              NaN              NaN          NaN   
max     99301.000000     NaN              NaN              NaN          NaN   
std     32076.677954     NaN              NaN              NaN          NaN   

           Product_Name         Sales         Year        Month  Month_Name  
count              9800   9800.000000  3959.000000  3959.000000        3959  
unique             1849           NaN          NaN          NaN          12  
top     Staple envelope           NaN          NaN          NaN    February  
freq                 47           NaN          NaN          NaN         370  
mean                NaN    230.769059  2016.728467     6.452892         NaN  
min                 NaN      0.444000  2015.000000     1.000000         NaN  
25%                 NaN     17.248000  2016.000000     3.000000         NaN  
50%                 NaN     54.490000  2017.000000     6.000000         NaN  
75%                 NaN    210.605000  2018.000000     9.000000         NaN  
max                 NaN  22638.480000  2018.000000    12.000000         NaN  
std                 NaN    626.651875     1.119118     3.497959         NaN  

[11 rows x 21 columns]
```

In [84]:

```
for col in df.columns:
    print(f"{col}: {df[col].nunique()} unique values")
```

```
Row_ID: 9800 unique values
Order_ID: 4922 unique values
Order_Date: 479 unique values
Ship_Date: 526 unique values
Ship_Mode: 4 unique values
Customer_ID: 793 unique values
Customer_Name: 793 unique values
Segment: 3 unique values
Country: 1 unique values
City: 529 unique values
State: 49 unique values
Postal_Code: 627 unique values
Region: 4 unique values
Product_ID: 1861 unique values
Category: 3 unique values
Sub_Category: 17 unique values
Product_Name: 1849 unique values
Sales: 5757 unique values
Year: 4 unique values
Month: 12 unique values
Month_Name: 12 unique values
```

In [85]:

```
categorical_cols = df.select_dtypes(include=['object']).columns

for col in categorical_cols:
    print(f"\nValue counts for '{col}':")
    print(df[col].value_counts())
```

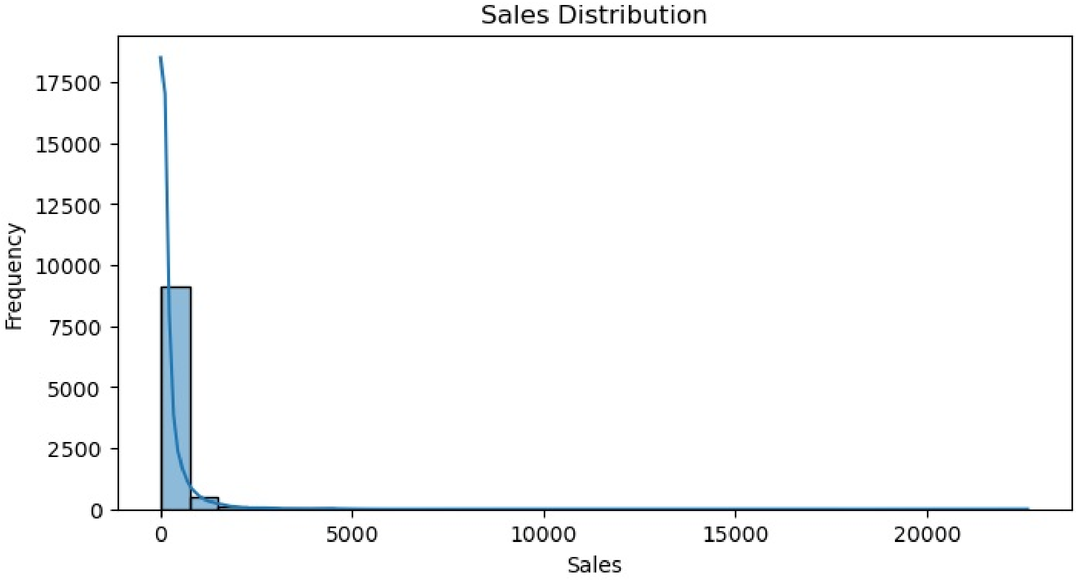
```
Value counts for 'Order_ID':
Order_ID
CA-2018-100111    14
CA-2018-157987    12
CA-2017-165330    11
US-2017-108504    11
CA-2017-105732    10
                  ..
US-2016-110261     1
CA-2016-125710     1
US-2016-137960     1
CA-2016-124975     1
CA-2016-142202     1
Name: count, Length: 4922, dtype: int64

Value counts for 'Ship_Mode':
Ship_Mode
Standard Class    5859
Second Class      1902
First Class       1501
Same Day           538
Name: count, dtype: int64

Value counts for 'Customer_ID':
Customer_ID
WB-21850    35
MA-17560    34
PP-18955    34
JL-15835    33
CK-12205    32
            ..
JR-15700     1
CJ-11875     1
SC-20845     1
RE-19405     1
AO-10810     1
Name: count, Length: 793, dtype: int64

Value counts for 'Customer_Name':
Customer_Name
William Brown          35
Matt Abelman           34
Paul Prost             34
John Lee               33
Chloris Kastensmidt    32
                       ..
Jocasta Rupert          1
Carl Jackson            1
Sung Chung              1
Ricardo Emerson         1
Anthony O Donnel        1
Name: count, Length: 793, dtype: int64

Value counts for 'Segment':
Segment
Consumer       5101
```

```
Corporate      2953
Home Office    1746
Name: count, dtype: int64

Value counts for 'Country':
Country
United States    9800
Name: count, dtype: int64

Value counts for 'City':
City
New York City    891
Los Angeles      728
Philadelphia     532
San Francisco    500
Seattle          426
                ...
San Mateo          1
Cheyenne           1
Conway             1
Melbourne          1
Springdale         1
Name: count, Length: 529, dtype: int64

Value counts for 'State':
State
California              1946
New York                1097
Texas                    973
Pennsylvania             582
Washington               504
Illinois                 483
Ohio                     454
Florida                  373
Michigan                 253
North Carolina           247
Virginia                 224
Arizona                  223
Tennessee                183
Colorado                 179
Georgia                  177
Kentucky                 137
Indiana                  135
Massachusetts            135
Oregon                   122
New Jersey               122
Maryland                 105
Wisconsin                105
Delaware                  93
Minnesota                 89
Connecticut               82
Missouri                  66
Oklahoma                  66
Alabama                   61
Arkansas                  60
Rhode Island              55
Mississippi               53
Utah                      53
South Carolina            42
Louisiana                 41
Nevada                    39
Nebraska                  38
New Mexico                37
New Hampshire             27
Iowa                      26
Kansas                    24
Idaho                     21
Montana                   15
South Dakota              12
Vermont                   11
District of Columbia      10
Maine                      8
North Dakota               7
West Virginia              4
Wyoming                    1
Name: count, dtype: int64

Value counts for 'Region':
Region
West       3140
East       2785
Central    2277
South      1598
```

```
Name: count, dtype: int64

Value counts for 'Product_ID':
Product_ID
OFF-PA-10001970    19
TEC-AC-10003832    18
FUR-FU-10004270    16
TEC-AC-10002049    15
TEC-AC-10003628    15
                   ..
OFF-PA-10000919     1
TEC-MA-10003353     1
OFF-LA-10003388     1
OFF-EN-10004206     1
TEC-PH-10002645     1
Name: count, Length: 1861, dtype: int64

Value counts for 'Category':
Category
Office Supplies    5909
Furniture          2078
Technology         1813
Name: count, dtype: int64

Value counts for 'Sub_Category':
Sub_Category
Binders        1492
Paper          1338
Furnishings     931
Phones          876
Storage         832
Art             785
Accessories     756
Chairs          607
Appliances      459
Labels          357
Tables          314
Envelopes       248
Bookcases       226
Fasteners       214
Supplies        184
Machines        115
Copiers          66
Name: count, dtype: int64

Value counts for 'Product_Name':
Product_Name
Staple envelope                                                              47
Staples                                                                      46
Easy-staple paper                                                            44
Avery Non-Stick Binders                                                      20
Staple remover                                                               18
                                                                             ..
Park Ridge Embossed Executive Business Envelopes                              1
Canon imageCLASS MF7460 Monochrome Digital Laser Multifunction Copier         1
Belkin 8 Outlet SurgeMaster II Gold Surge Protector with Phone Protection     1
Boston 1900 Electric Pencil Sharpener                                         1
LG G2                                                                         1
Name: count, Length: 1849, dtype: int64

Value counts for 'Month_Name':
Month_Name
February     370
May          362
March        357
November     351
August       348
September    344
December     334
January      329
April        305
October      295
July         294
June         270
Name: count, dtype: int64
```

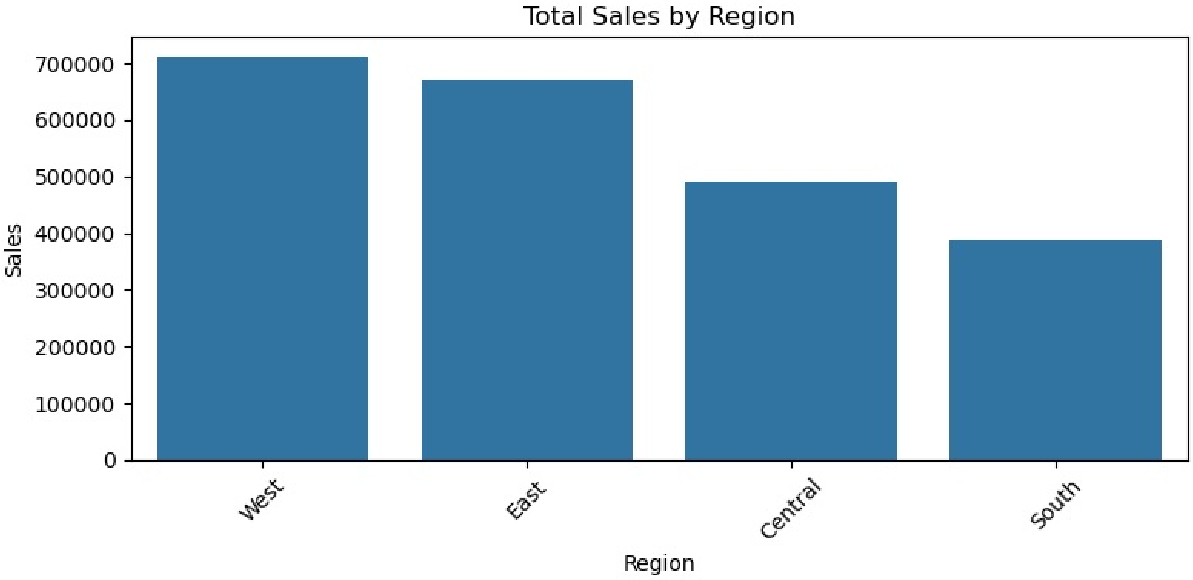
In [86]:

```python
import matplotlib.pyplot as plt
import seaborn as sns
# Sales Distribution
plt.figure(figsize=(8, 4))
sns.histplot(df['Sales'], bins=30, kde=True)
plt.title('Sales Distribution')
plt.xlabel('Sales')
plt.ylabel('Frequency')
plt.show()
```



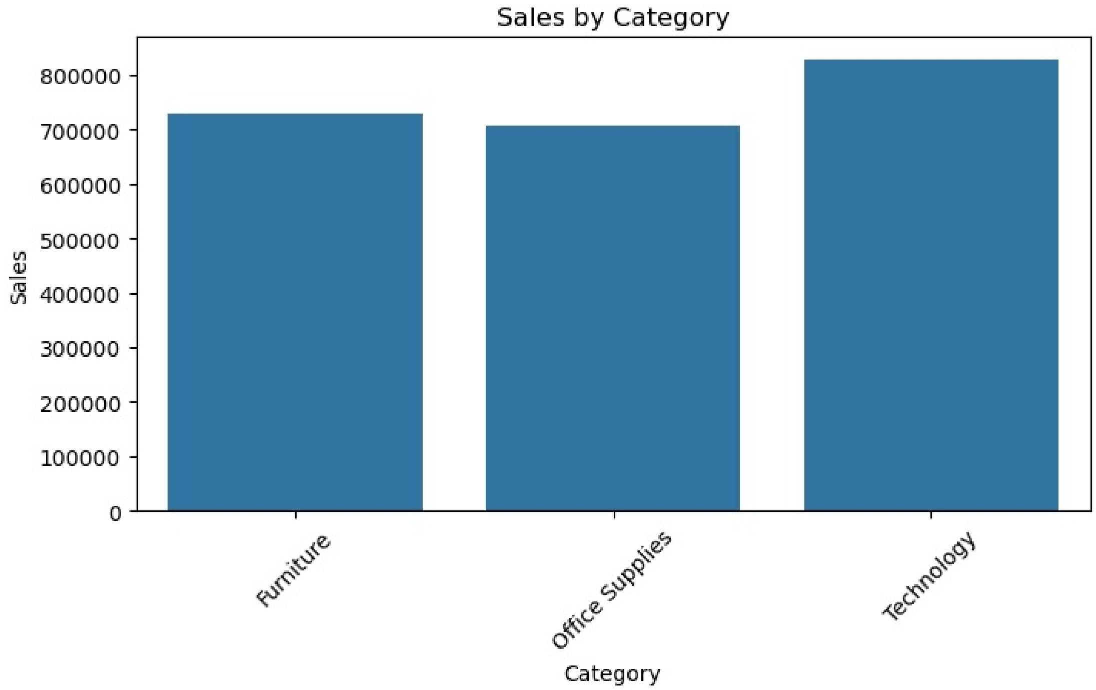
In [87]:

```
# Sales by Region
plt.figure(figsize=(8, 4))
region_sales = df.groupby('Region')['Sales'].sum().sort_values(ascending=False)
sns.barplot(x=region_sales.index, y=region_sales.values)
plt.title('Total Sales by Region')
plt.ylabel('Sales')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



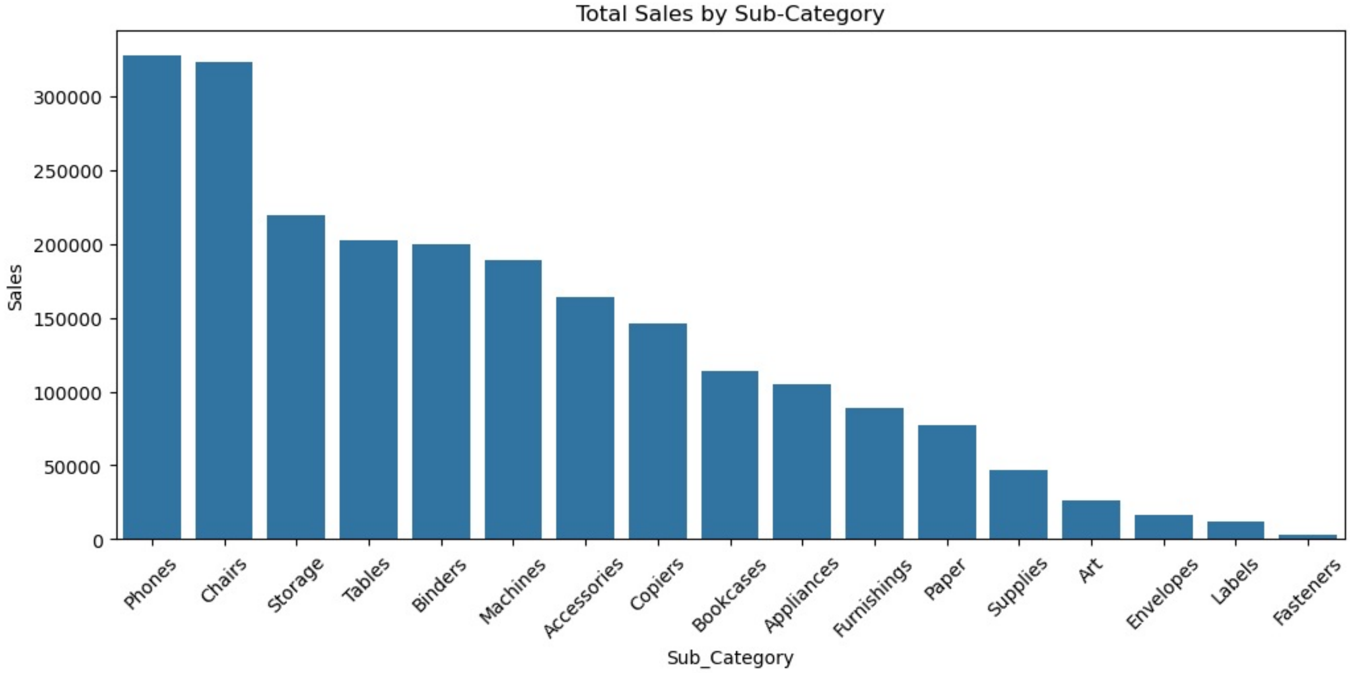
In [88]:

```
# Sales by Category
plt.figure(figsize=(8, 4))
cat_sales = df.groupby('Category')['Sales'].sum()
sns.barplot(x=cat_sales.index, y=cat_sales.values)
plt.title('Sales by Category')
plt.ylabel('Sales')
plt.xticks(rotation=45)
plt.show()
```



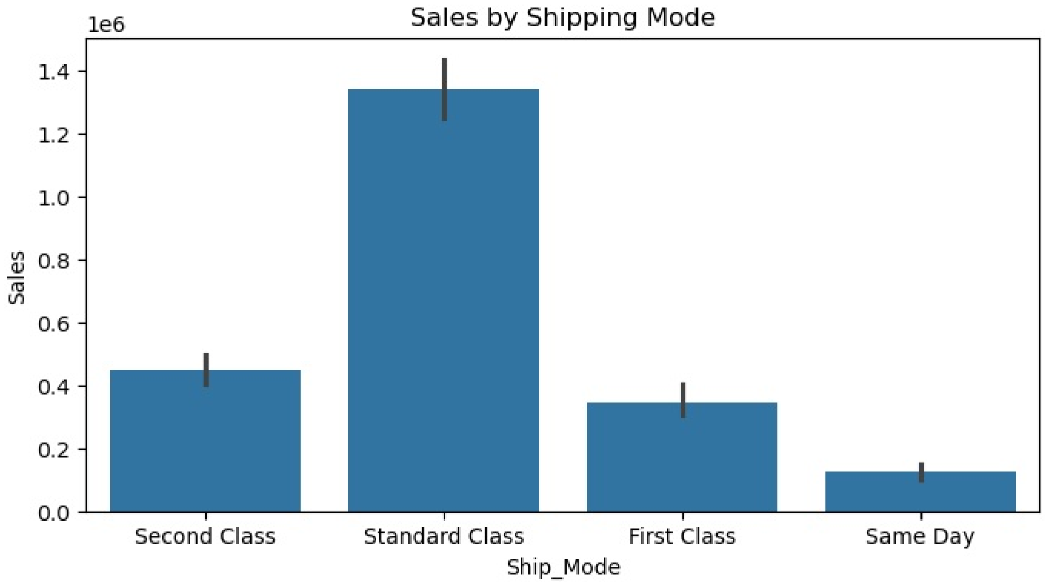
In [120…

```
# Sales by Sub-Category
plt.figure(figsize=(12, 5))
subcat_sales = df.groupby('Sub_Category')['Sales'].sum().sort_values(ascending=False)
sns.barplot(x=subcat_sales.index, y=subcat_sales.values)
plt.title('Total Sales by Sub-Category')
plt.xticks(rotation=45)
plt.ylabel('Sales')
plt.show()
```



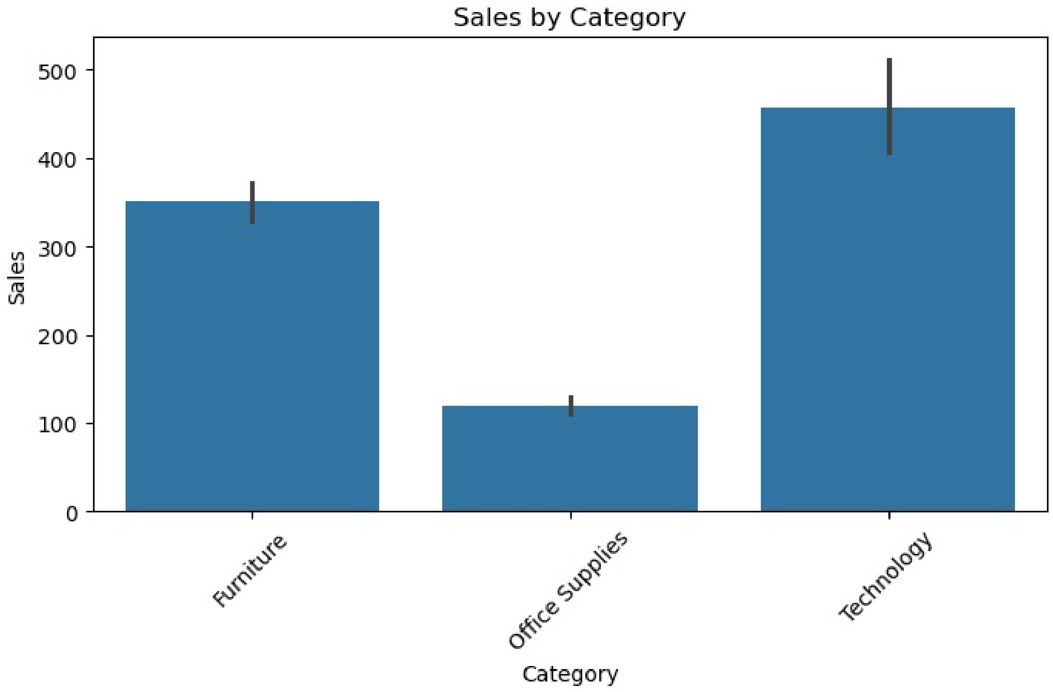
In [122…

```
# Sales by Ship Mode
plt.figure(figsize=(8, 4))
sns.barplot(data=df, x='Ship_Mode', y='Sales', estimator=sum)
plt.title('Sales by Shipping Mode')
plt.ylabel('Sales')
plt.show()
```



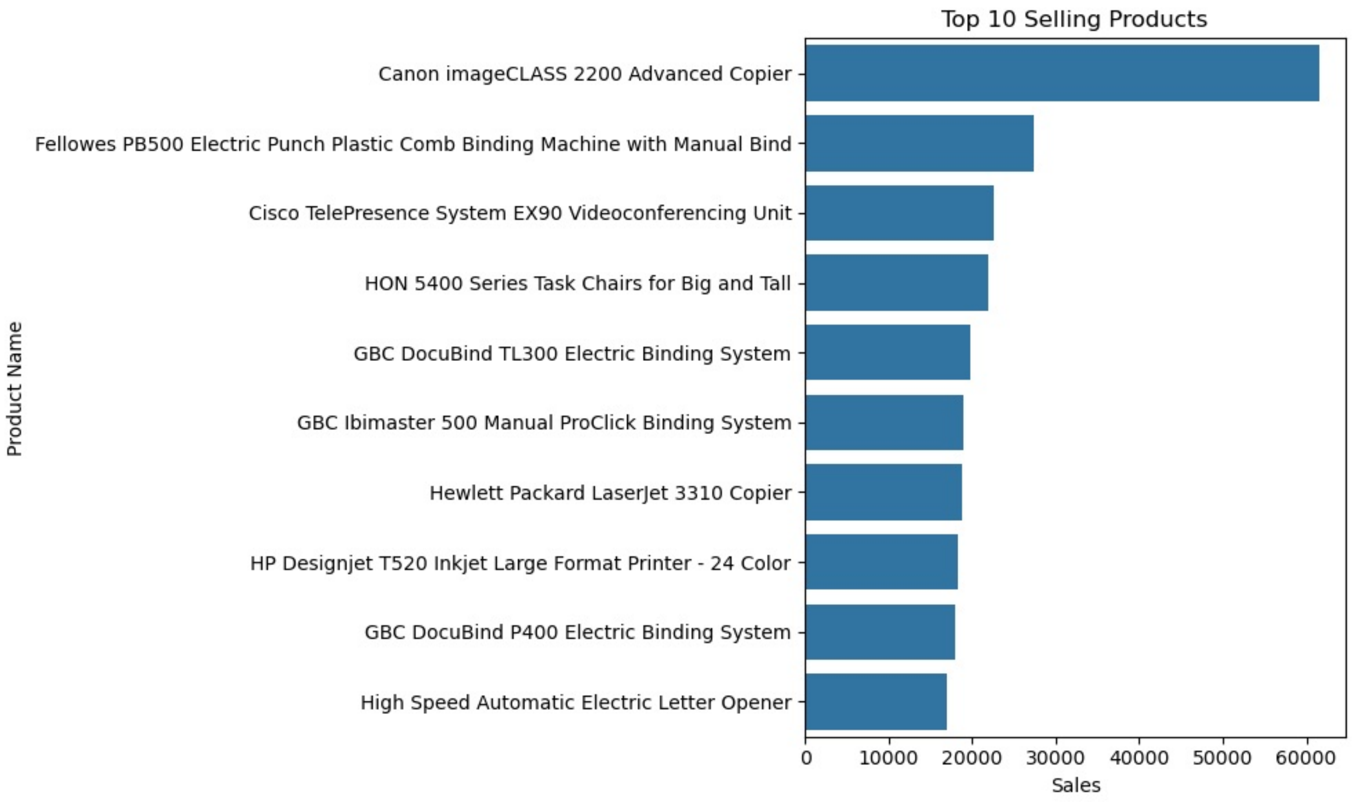
In [90]:

```
if 'Category' in df.columns and 'Sales' in df.columns:
    plt.figure(figsize=(8, 4))
    sns.barplot(x='Category', y='Sales', data=df)
    plt.title("Sales by Category")
    plt.xticks(rotation=45)
    plt.show()
```



In [92]:

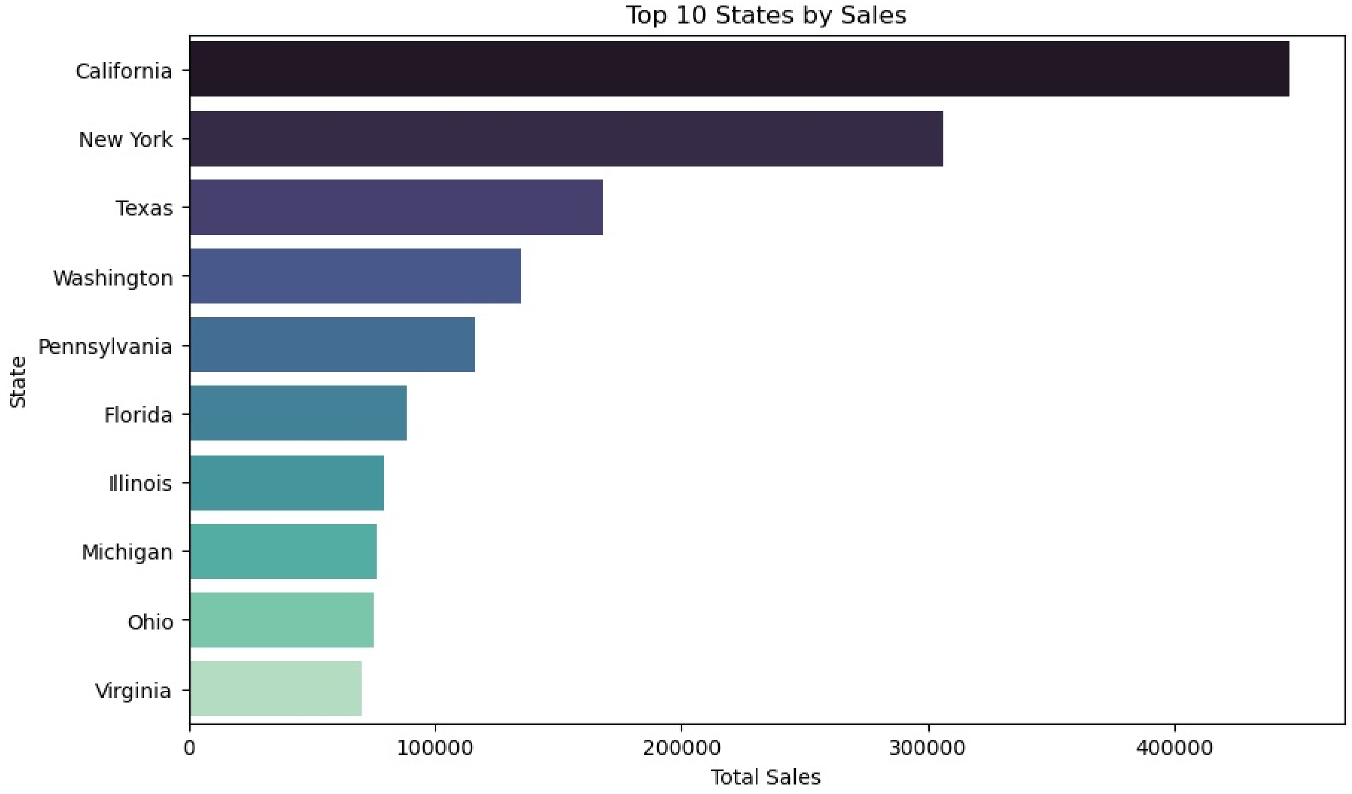
```
# Top 10 Selling Products
top_products = df.groupby('Product_Name')['Sales'].sum().sort_values(ascending=False).head(10)
plt.figure(figsize=(10, 6))
sns.barplot(x=top_products.values, y=top_products.index)
plt.title("Top 10 Selling Products")
plt.xlabel("Sales")
plt.ylabel("Product Name")
plt.tight_layout()
plt.show()
```



In [128…

```
top_states = df.groupby('State')['Sales'].sum().sort_values(ascending=False).head(10)

plt.figure(figsize=(10, 6))
sns.barplot(x=top_states.values, y=top_states.index, palette='mako')
plt.title('Top 10 States by Sales')
plt.xlabel('Total Sales')
plt.ylabel('State')
plt.show()
```

```
C:\Users\jeeva\AppData\Local\Temp\ipykernel_11280\3804100394.py:4: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable
to `hue` and set `legend=False` for the same effect.

  sns.barplot(x=top_states.values, y=top_states.index, palette='mako')
```

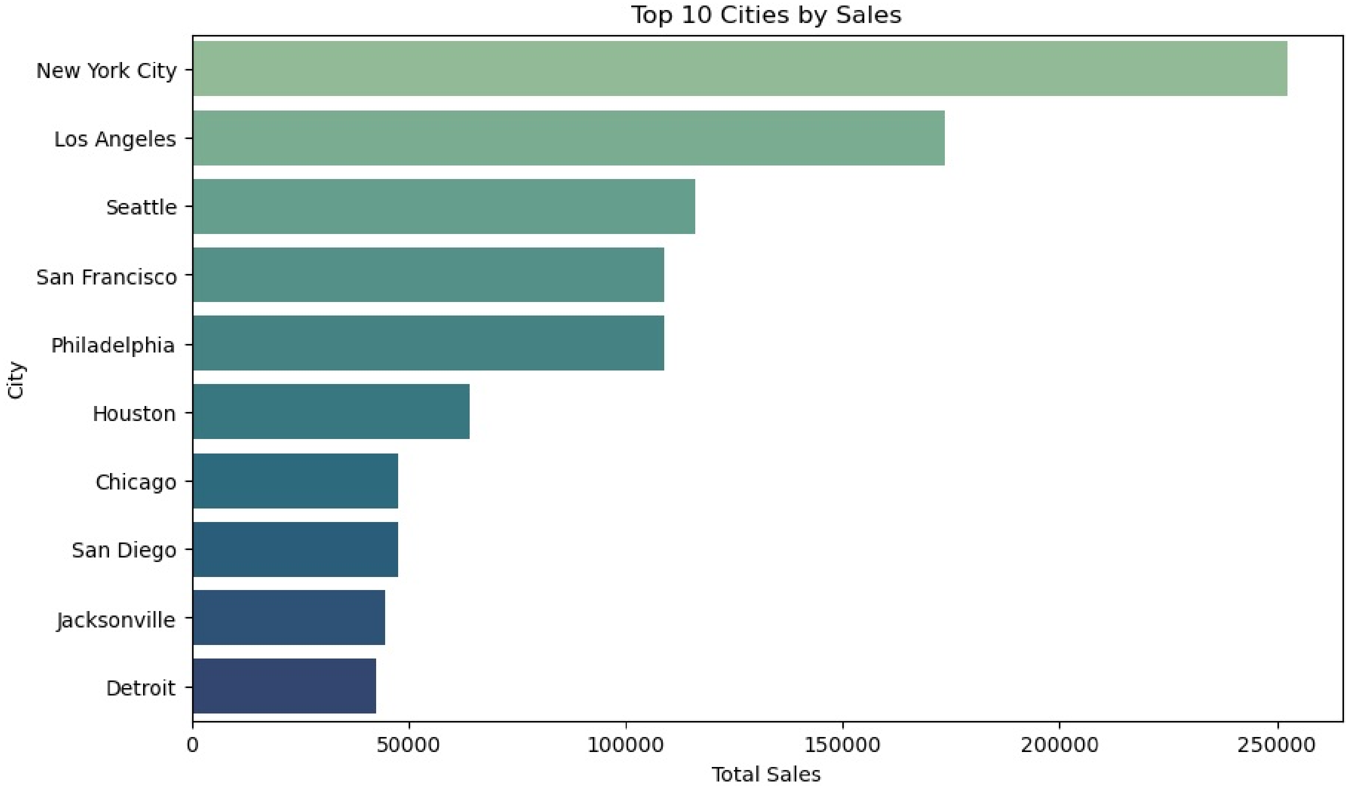


In [130…

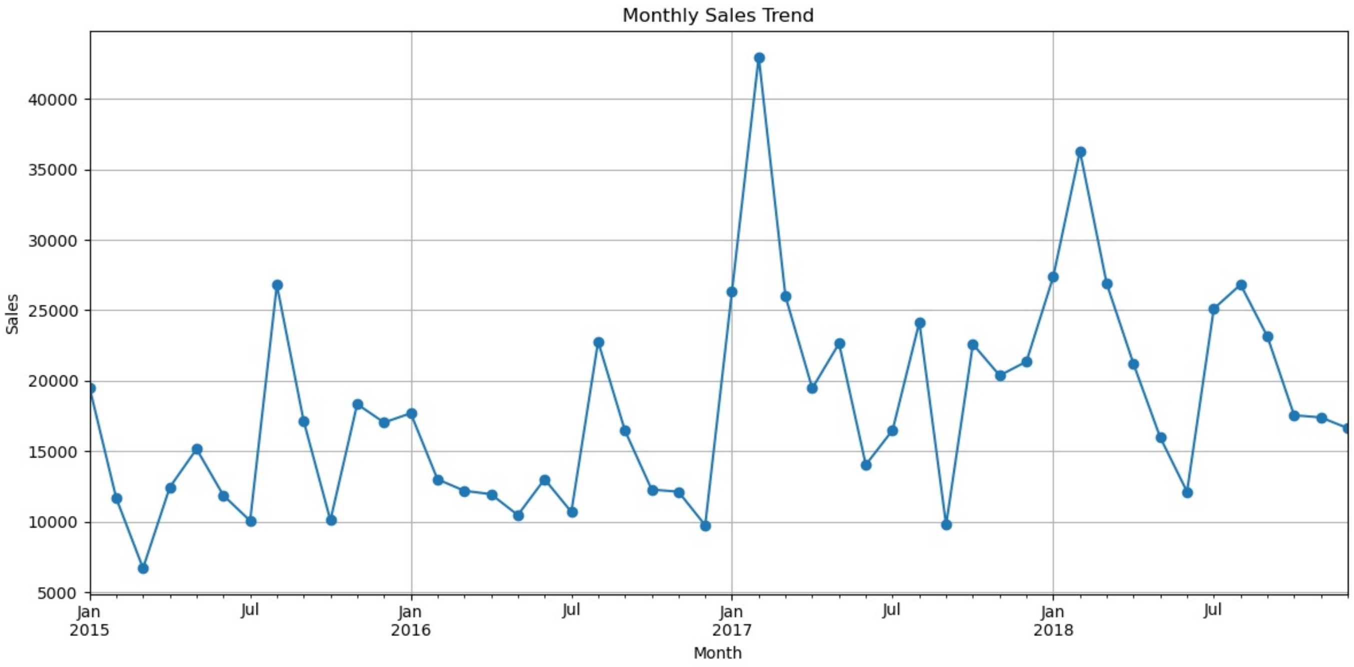
```
top_cities = df.groupby('City')['Sales'].sum().sort_values(ascending=False).head(10)

plt.figure(figsize=(10, 6))
sns.barplot(x=top_cities.values, y=top_cities.index, palette='crest')
```

```
plt.title('Top 10 Cities by Sales')
plt.xlabel('Total Sales')
plt.ylabel('City')
plt.show()
```

```
C:\Users\jeeva\AppData\Local\Temp\ipykernel_11280\2636483304.py:4: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable
to `hue` and set `legend=False` for the same effect.

  sns.barplot(x=top_cities.values, y=top_cities.index, palette='crest')
```

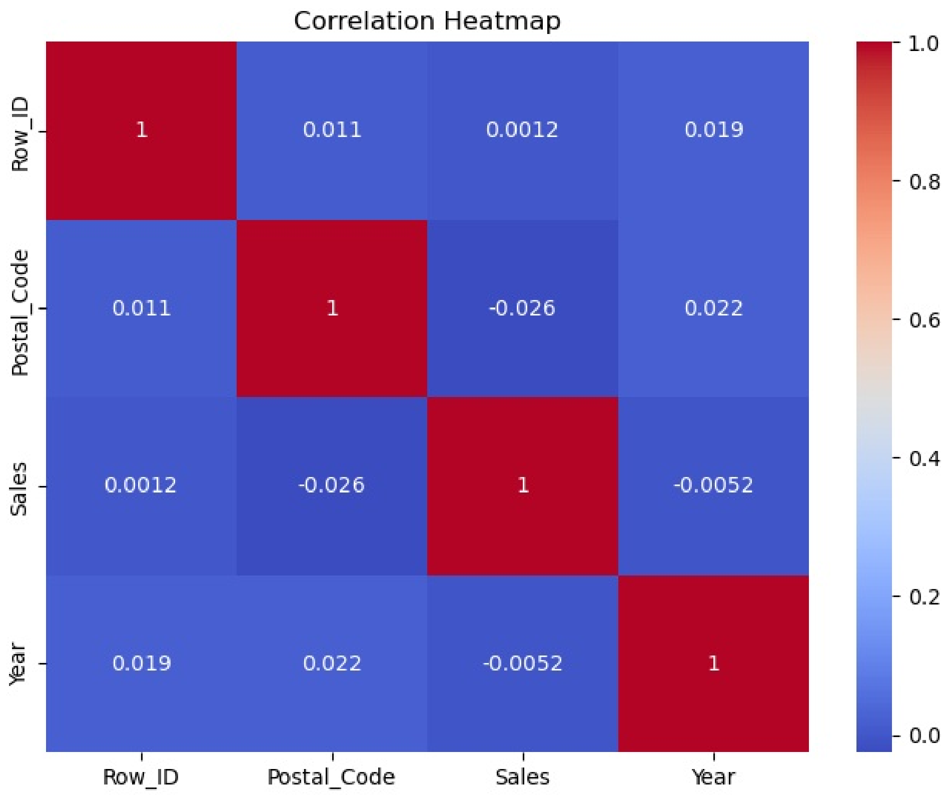


In [93]:
```
# Monthly Sales Trend
df['Month'] = df['Order_Date'].dt.to_period('M')
monthly_sales = df.groupby('Month')['Sales'].sum()

plt.figure(figsize=(12, 6))
monthly_sales.plot(marker='o')
plt.title("Monthly Sales Trend")
plt.xlabel("Month")
plt.ylabel("Sales")
plt.grid(True)
plt.tight_layout()
plt.show()
```



In [94]:
```
#CORRELATION HEATMAP
```

```
plt.figure(figsize=(8, 6))
sns.heatmap(df.corr(numeric_only=True), annot=True, cmap='coolwarm')
plt.title("Correlation Heatmap")
plt.show()
```

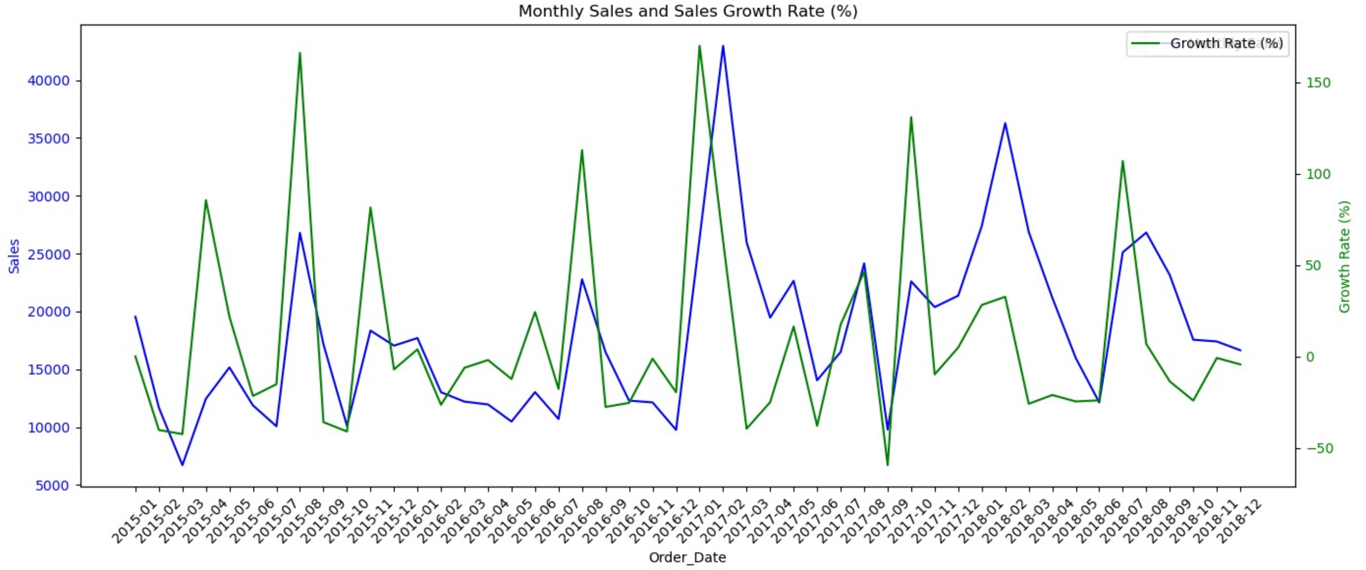


In [132…

```
# Monthly Sales Growth Rate
# Grouping by Year and Month
monthly_sales = df.groupby([df['Order_Date'].dt.to_period('M')])['Sales'].sum().reset_index()
monthly_sales['Order_Date'] = monthly_sales['Order_Date'].astype(str)  # convert period to string
monthly_sales['Growth_Rate_%'] = monthly_sales['Sales'].pct_change() * 100

monthly_sales.fillna(0, inplace=True)

print("\nMonthly Sales and Growth Rate:\n")
print(monthly_sales)

# Plot Sales and Growth Rate
fig, ax1 = plt.subplots(figsize=(14, 6))

sns.lineplot(data=monthly_sales, x='Order_Date', y='Sales', label='Monthly Sales', ax=ax1, color='blue')
ax1.set_ylabel('Sales', color='blue')
ax1.tick_params(axis='y', labelcolor='blue')
ax1.set_xticklabels(monthly_sales['Order_Date'], rotation=45)

ax2 = ax1.twinx()
sns.lineplot(data=monthly_sales, x='Order_Date', y='Growth_Rate_%', label='Growth Rate (%)', ax=ax2, color='gre
ax2.set_ylabel('Growth Rate (%)', color='green')
ax2.tick_params(axis='y', labelcolor='green')

plt.title('Monthly Sales and Sales Growth Rate (%)')
fig.tight_layout()
plt.show()
```

```
Monthly Sales and Growth Rate:

   Order_Date        Sales  Growth_Rate_%
0     2015-01  19546.1630       0.000000
1     2015-02  11678.9940     -40.249173
2     2015-03   6716.0440     -42.494670
3     2015-04  12455.4820      85.458612
4     2015-05  15165.0510      21.754028
5     2015-06  11884.1690     -21.634494
6     2015-07  10075.7400     -15.217126
7     2015-08  26797.7630     165.963225
8     2015-09  17158.9320     -35.968790
9     2015-10  10112.6410     -41.064858
10    2015-11  18349.7640      81.453727
11    2015-12  17045.8427      -7.105930
12    2016-01  17701.6864       3.847529
13    2016-02  13018.3150     -26.457205
14    2016-03  12207.4066      -6.228981
15    2016-04  11963.6960      -1.996416
16    2016-05  10483.4820     -12.372548
17    2016-06  13026.6682      24.258984
18    2016-07  10706.7200     -17.809222
19    2016-08  22782.5770     112.787642
20    2016-09  16484.9010     -27.642509
21    2016-10  12293.0380     -25.428500
22    2016-11  12139.0395      -1.252729
23    2016-12   9761.3330     -19.587270
24    2017-01  26342.5410     169.866226
25    2017-02  42967.9150      63.112264
26    2017-03  25982.2870     -39.530957
27    2017-04  19472.1640     -25.056005
28    2017-05  22649.3888      16.316752
29    2017-06  14050.1430     -37.966790
30    2017-07  16501.0070      17.443694
31    2017-08  24156.3226      46.393021
32    2017-09   9789.6620     -59.473707
33    2017-10  22599.7820     130.853547
34    2017-11  20378.2400      -9.829927
35    2017-12  21365.1485       4.842953
36    2018-01  27367.5920      28.094555
37    2018-02  36285.9360      32.587244
38    2018-03  26882.9530     -25.913574
39    2018-04  21203.6070     -21.126198
40    2018-05  15979.1570     -24.639440
41    2018-06  12138.1558     -24.037571
42    2018-07  25110.4795     106.872279
43    2018-08  26823.6900       6.822691
44    2018-09  23148.8700     -13.699905
45    2018-10  17558.3220     -24.150414
46    2018-11  17407.2700      -0.860287
47    2018-12  16647.0420      -4.367302
```

```
C:\Users\jeeva\AppData\Local\Temp\ipykernel_11280\4285511468.py:24: UserWarning: set_ticklabels() should only be
used with a fixed number of ticks, i.e. after set_ticks() or using a FixedLocator.
  ax1.set_xticklabels(monthly_sales['Order_Date'], rotation=45)
```



In [ ]:

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