In [1]:
```
#import necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
#Load the dataset
df=pd.read_csv("global Superstore Sales Data.csv")
```

In [3]:
```
df.head(5)
```

Out[3]:

	Row ID	Order ID	Order Date	Ship Date	Ship Mode	Customer ID	Customer Name	Segment	Country	
0	1	CA-2017-152156	08-11-2017	11-11-2017	Second Class	CG-12520	Claire Gute	Consumer	United States	Hen
1	2	CA-2017-152156	08-11-2017	11-11-2017	Second Class	CG-12520	Claire Gute	Consumer	United States	Hen
2	3	CA-2017-138688	12-06-2017	16-06-2017	Second Class	DV-13045	Darrin Van Huff	Corporate	United States	A
3	4	US-2016-108966	11-10-2016	18-10-2016	Standard Class	SO-20335	Sean O'Donnell	Consumer	United States	Lauc
4	5	US-2016-108966	11-10-2016	18-10-2016	Standard Class	SO-20335	Sean O'Donnell	Consumer	United States	Lauc

In [4]:
```
#Identify missing values and duplicates.
df.isnull().sum()
```

Out[4]:
```
Row ID           0
Order ID         0
Order Date       0
Ship Date        0
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
dtype: int64
```

In [5]:
```
df.duplicated().sum()
```

Out[5]:
```
np.int64(0)
```

In [6]:
```
# drop missing values
df = df.dropna()
```

In [7]:
```
df.dtypes
```

Out[7]:
```
Row ID           int64
Order ID        object
Order Date      object
Ship Date       object
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
dtype: object
```

In [8]:
```
# change data types to date & time
df['Order Date'] = pd.to_datetime(df['Order Date'], format='%d-%m-%Y', errors='c
df['Ship Date'] = pd.to_datetime(df['Ship Date'], format='%d-%m-%Y', errors='coe
```

In [9]:
```
df = df.dropna(subset=['Ship Date'])
```

In [10]:
```
df['Ship Date'].fillna(df['Ship Date'].mean(), inplace=True)
```

In [11]:
```
df = df.dropna(subset=['Order Date'])
```

In [12]:
```
df['Order Date'].fillna(df['Order Date'].mean(), inplace=True)
```

In [13]:
```
# change data types to date & time
df['Order Date'] = pd.to_datetime(df['Order Date'], format='%d-%m-%Y', errors='c
df['Ship Date'] = pd.to_datetime(df['Ship Date'], format='%d-%m-%Y', errors='coe
```

In [14]:
```
# change postal code to str
df['Postal Code'] = df['Postal Code'].astype(str)
```

In [15]:
```
# change to categorical column
cat_cols = ['Ship Mode', 'Segment', 'Country', 'City', 'State',
            'Region', 'Category', 'Sub-Category']

for col in cat_cols:
    df[col] = df[col].astype('category')
```

In [16]:
```
id_cols = ['Order ID', 'Customer ID', 'Product ID']
for col in id_cols:
    df[col] = df[col].astype(str)
```

In [17]:
```
df.dtypes
```

Out[17]:
```
Row ID                 int64
Order ID              object
Order Date    datetime64[ns]
Ship Date     datetime64[ns]
Ship Mode           category
Customer ID           object
Customer Name         object
Segment             category
Country             category
City                category
State               category
Postal Code           object
Region              category
Product ID            object
Category            category
Sub-Category        category
Product Name          object
Sales                float64
dtype: object
```

In [18]:
```
# Extract Year and Month from 'Order Date'
df['Year'] = df['Order Date'].dt.year
df['Month_Name'] = df['Order Date'].dt.strftime('%B')
```

In [19]:
```
#grouped Sales into levels – Low, Medium, High and very high
df['Sales_Category'] = pd.cut(
    df['Sales'],
    bins=[0, 100, 500, 1000, df['Sales'].max()],
    labels=['Low', 'Medium', 'High', 'Very High']
)
```

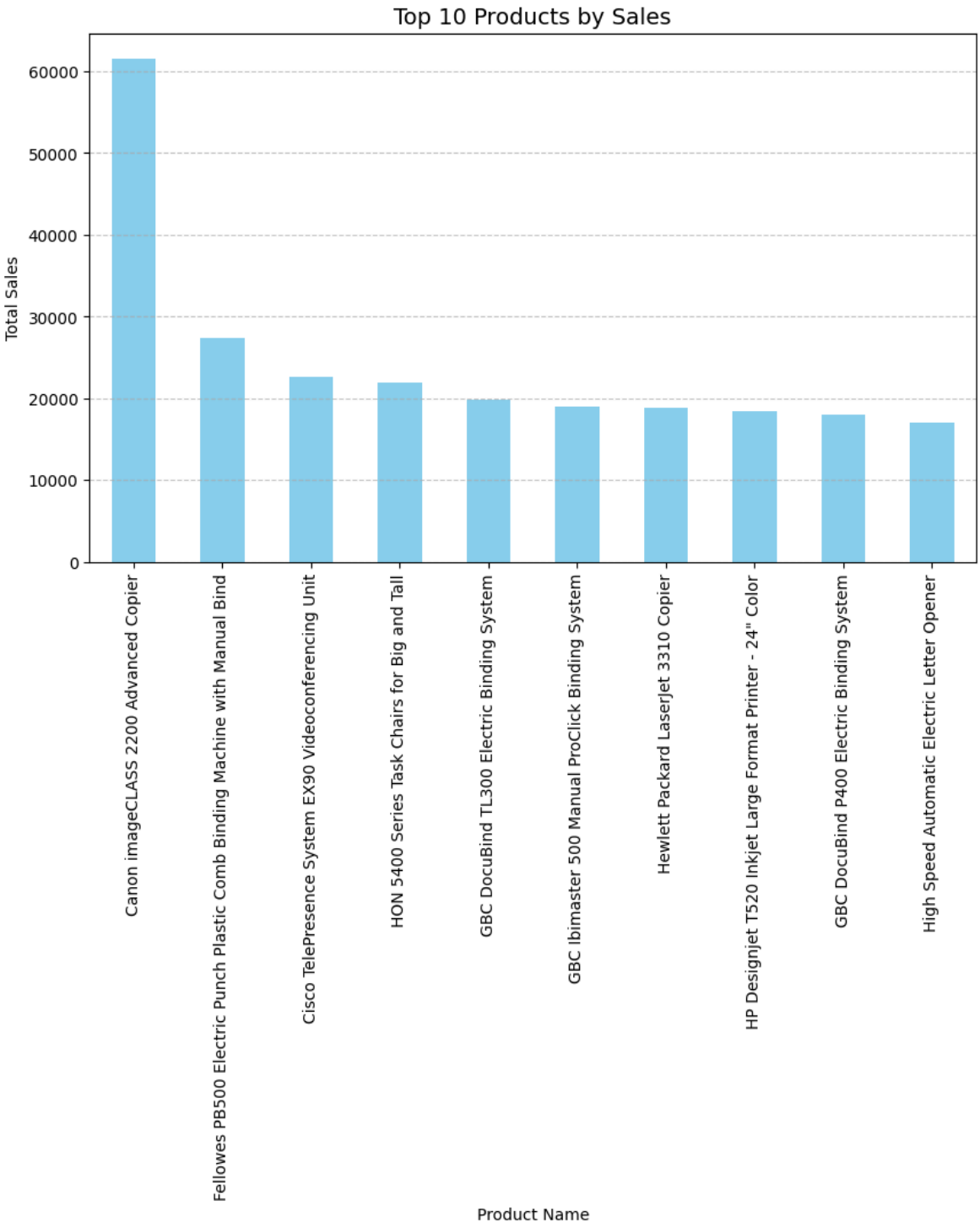
In [20]:
```
# top 10 product by sales
top_products = (
    df.groupby('Product Name')['Sales'].sum().sort_values(ascending=False).head(
```

```
print(top_products)
```

```
Product Name
Canon imageCLASS 2200 Advanced Copier                                    61
599.824
Fellowes PB500 Electric Punch Plastic Comb Binding Machine with Manual Bind    27
453.384
Cisco TelePresence System EX90 Videoconferencing Unit                    22
638.480
HON 5400 Series Task Chairs for Big and Tall                             21
870.576
GBC DocuBind TL300 Electric Binding System                               19
823.479
GBC Ibimaster 500 Manual ProClick Binding System                         19
024.500
Hewlett Packard LaserJet 3310 Copier                                     18
839.686
HP Designjet T520 Inkjet Large Format Printer - 24" Color                18
374.895
GBC DocuBind P400 Electric Binding System                                17
965.068
High Speed Automatic Electric Letter Opener                              17
030.312
Name: Sales, dtype: float64
```

In [21]:
```
# bar chart
plt.figure(figsize=(10,6))

top_products.plot(kind='bar', color='skyblue')
plt.title('Top 10 Products by Sales', fontsize=14)
plt.xlabel('Product Name')
plt.ylabel('Total Sales')
plt.xticks(rotation=90)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()
```

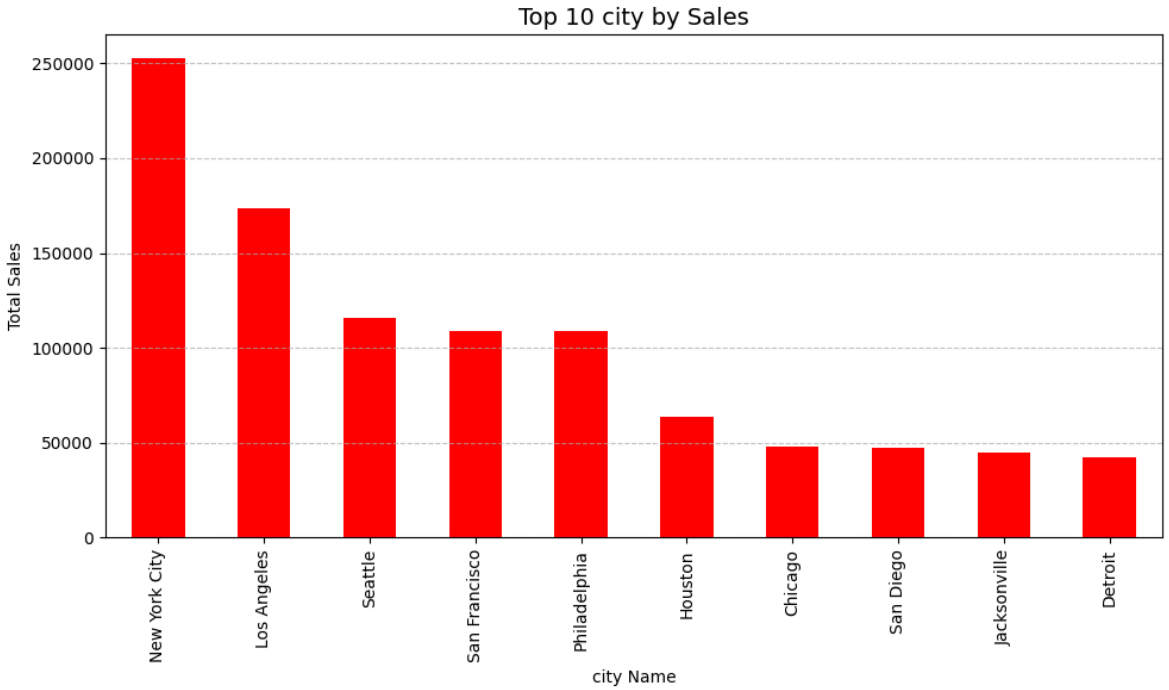


In [22]:
```
# top 10 city by sales
sales_by_city = (df.groupby('City')['Sales'].sum().sort_values(ascending=False).
print(sales_by_city)
```

```
City
New York City    252462.5470
Los Angeles      173420.1810
Seattle          116106.3220
San Francisco    109041.1200
Philadelphia     108841.7490
Houston           63956.1428
Chicago           47820.1330
San Diego         47521.0290
Jacksonville      44713.1830
Detroit           42446.9440
Name: Sales, dtype: float64
```

In [23]:
```
plt.figure(figsize=(10,6))

sales_by_city.plot(kind='bar', color = "Red")
plt.title('Top 10 city by Sales', fontsize=14)
plt.xlabel('city Name')
plt.ylabel('Total Sales')
plt.xticks(rotation=90)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()
```



In [24]:
```
#insights
# Newyork city has the highest no of sales with the sales amount of  252462.5470
```

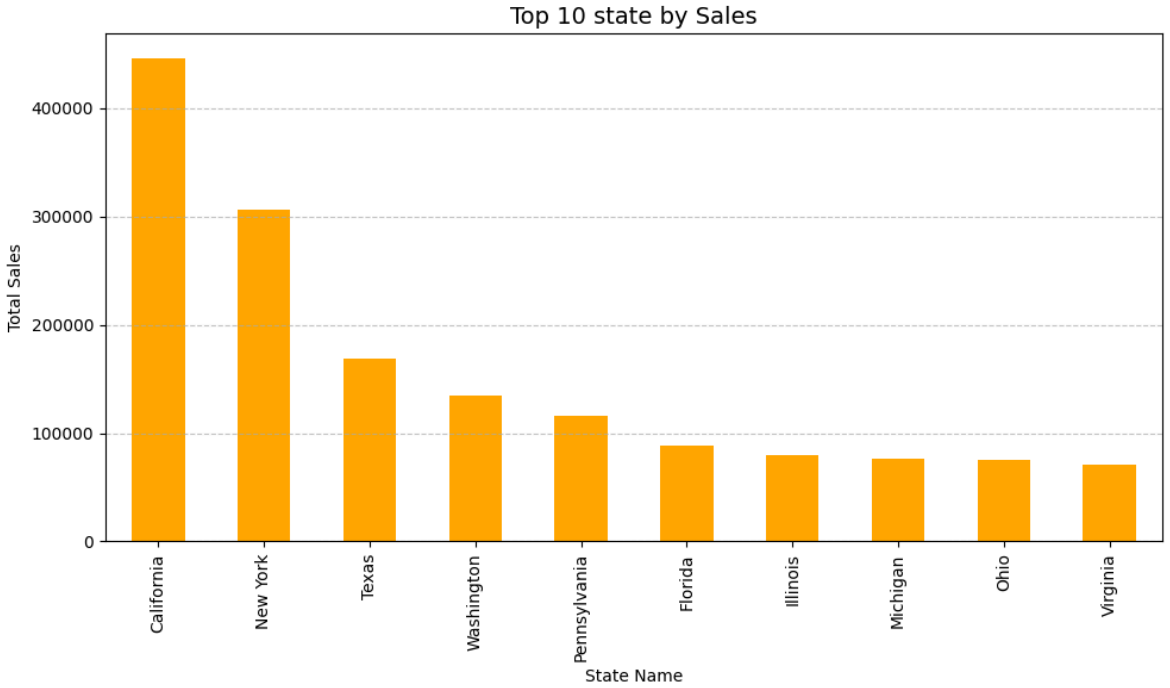
In [25]:
```
# top 10 states by sales
sales_by_state = (df.groupby('State')['Sales'].sum().sort_values(ascending=False
print(sales_by_state)
```

```
State
California      446306.4635
New York        306361.1470
Texas           168572.5322
Washington      135206.8500
Pennsylvania    116276.6500
Florida          88436.5320
Illinois         79236.5170
Michigan         76136.0740
Ohio             75130.3500
Virginia         70636.7200
Name: Sales, dtype: float64
```

In [26]:
```
plt.figure(figsize=(10,6))

sales_by_state.plot(kind='bar', color = "orange")
plt.title('Top 10 state by Sales', fontsize=14)
plt.xlabel('State Name')
plt.ylabel('Total Sales')
plt.xticks(rotation=90)
```

```
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()
```

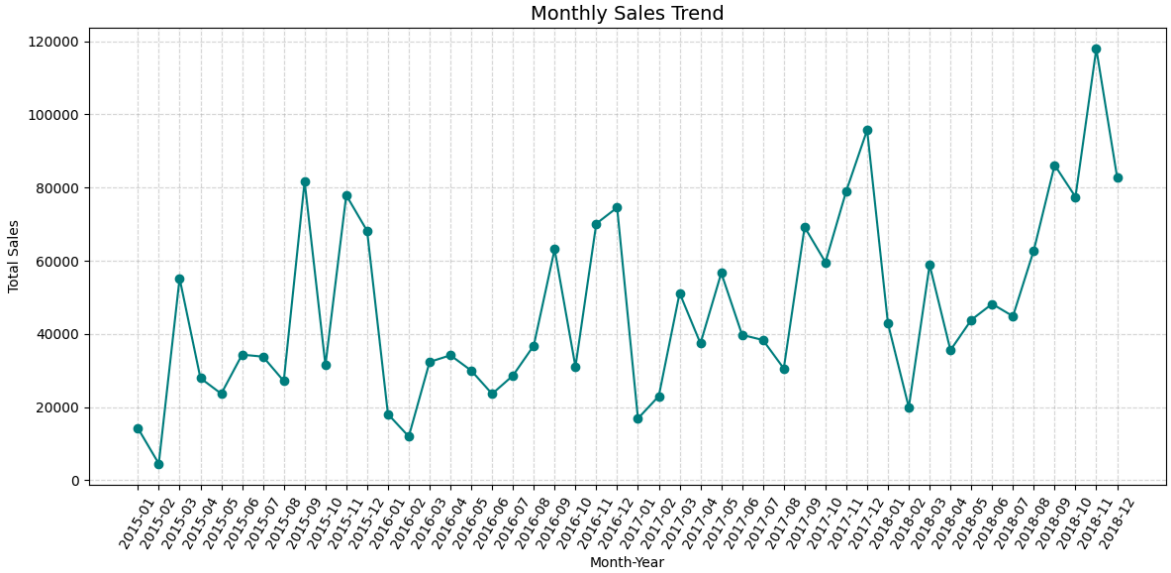


In [27]:
```
#insights
# California has the highest no of sales with the sales amount of 446306.4635 .
```

In [28]:
```
df['Month_Year'] = df['Order Date'].dt.to_period('M')
```

In [29]:
```
monthly_sales = (
    df.groupby('Month_Year')['Sales']
      .sum()
      .reset_index()
)
monthly_sales['Month_Year'] = monthly_sales['Month_Year'].astype(str)
```

In [30]:
```
# Plot Monthly Sales Trend (Line Chart)
plt.figure(figsize=(12,6))
plt.plot(monthly_sales['Month_Year'], monthly_sales['Sales'], marker='o', color=
plt.title('Monthly Sales Trend', fontsize=14)
plt.xlabel('Month-Year')
plt.ylabel('Total Sales')
plt.xticks(rotation=60)
plt.grid(True, linestyle='--', alpha=0.5)
plt.tight_layout()
plt.show()
```

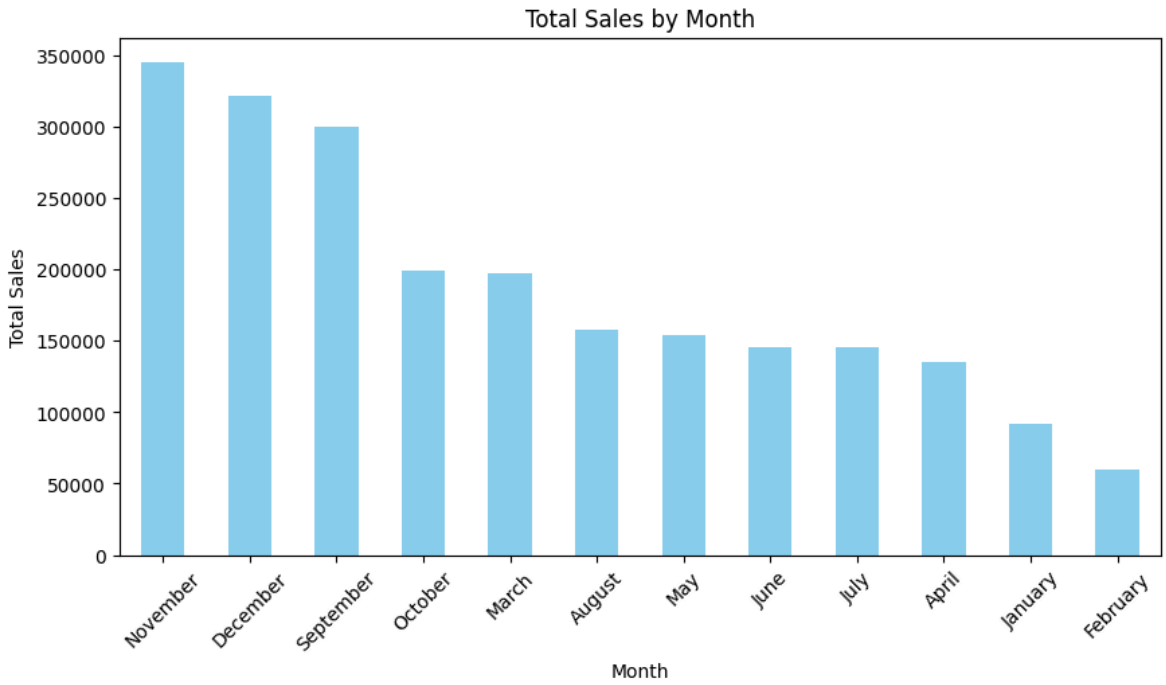


In [31]:
```
# Seasonal Peaks
df['Month'] = df['Order Date'].dt.month_name()
monthly_avg = (
    df.groupby('Month')['Sales']
      .sum()
      .sort_values(ascending=False)
)

print(monthly_avg)
```

```
Month
November     345041.6110
December     321275.1395
September    300103.4117
October      199496.2947
March        197573.5872
August       157315.9270
May          154086.7237
June         145837.5233
July         145535.6890
April        134988.2506
January       91982.1396
February      59371.1154
Name: Sales, dtype: float64
```

In [32]:
```
monthly_avg.plot(kind='bar', figsize=(10,5), color='skyblue')
plt.title('Total Sales by Month')
plt.ylabel('Total Sales')
plt.xticks(rotation=45)
plt.show()
```


Total Sales by Month
Total Sales
350000
300000
250000
200000
150000
100000
50000
0
November
December
September
October
March
August
May
June
July
April
January
February
Month

In [33]:
```
df
```

Out[33]:

	Row ID	Order ID	Order Date	Ship Date	Ship Mode	Customer ID	Customer Name	Segment	Country
0	1	CA-2017-152156	2017-11-08	2017-11-11	Second Class	CG-12520	Claire Gute	Consumer	United States
1	2	CA-2017-152156	2017-11-08	2017-11-11	Second Class	CG-12520	Claire Gute	Consumer	United States
2	3	CA-2017-138688	2017-06-12	2017-06-16	Second Class	DV-13045	Darrin Van Huff	Corporate	United States
3	4	US-2016-108966	2016-10-11	2016-10-18	Standard Class	SO-20335	Sean O'Donnell	Consumer	United States
4	5	US-2016-108966	2016-10-11	2016-10-18	Standard Class	SO-20335	Sean O'Donnell	Consumer	United States
...	...	...	...	...	...	...	...	...	...
9795	9796	CA-2017-125920	2017-05-21	2017-05-28	Standard Class	SH-19975	Sally Hughsby	Corporate	United States
9796	9797	CA-2016-128608	2016-01-12	2016-01-17	Standard Class	CS-12490	Cindy Schnelling	Corporate	United States
9797	9798	CA-2016-128608	2016-01-12	2016-01-17	Standard Class	CS-12490	Cindy Schnelling	Corporate	United States
9798	9799	CA-2016-128608	2016-01-12	2016-01-17	Standard Class	CS-12490	Cindy Schnelling	Corporate	United States
9799	9800	CA-2016-128608	2016-01-12	2016-01-17	Standard Class	CS-12490	Cindy Schnelling	Corporate	United States

9789 rows × 23 columns

In [34]:
```
df['Shipping_Days'] = (df['Ship Date'] - df['Order Date']).dt.days
```

In [35]:
```
shipping_by_ship_mode = (
    df.groupby('Ship Mode')['Shipping_Days']
      .mean()
      .reset_index()
      .sort_values('Shipping_Days')
)

print(shipping_by_ship_mode)
```

```
       Ship Mode  Shipping_Days
1       Same Day       0.044610
0    First Class       2.179214
2   Second Class       3.249868
3  Standard Class       5.009916
```

In [36]:
```
shipping_by_Region = (
    df.groupby('Region')['Shipping_Days']
      .mean()
      .reset_index()
      .sort_values('Shipping_Days')
)
```

In [37]:
```
shipping_by_Region
```

Out[37]:

	Region	Shipping_Days
1	East	3.910238
3	West	3.930255
2	South	3.961202
0	Central	4.065876

In [38]:
```
sales_by_Region = (
    df.groupby('Region')['Sales']
      .mean()
      .reset_index()
      .sort_values('Sales')
)
```

In [39]:
```
sales_by_Region
```

Out[39]:

	Region	Sales
0	Central	216.357889
3	West	226.184613
1	East	238.136033
2	South	243.524067

In [40]:
```
sales_by_Category = (
    df.groupby('Category')['Sales']
      .mean()
```

```
      .reset_index()
      .sort_values('Sales')
)
sales_by_Category
```

Out[40]:

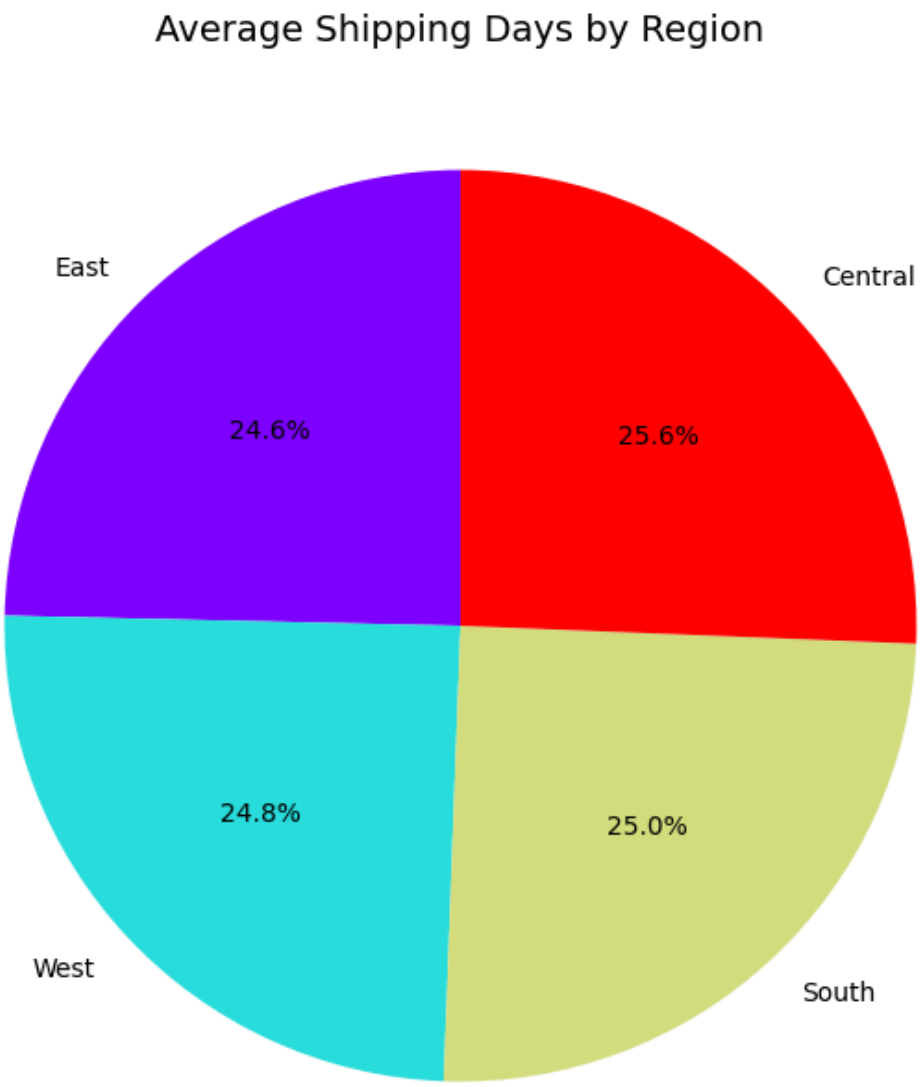
	Category	Sales
1	Office Supplies	119.128041
0	Furniture	348.525277
2	Technology	456.274096

In [41]:
```
df['Shipping_Days'] = (df['Ship Date'] - df['Order Date']).dt.days
```

In [42]:
```
df['Shipping_Days']
```

Out[42]:
```
0       3
1       3
2       4
3       7
4       7
       ..
9795    7
9796    5
9797    5
9798    5
9799    5
Name: Shipping_Days, Length: 9789, dtype: int64
```

In [43]:
```
# Visualize a pie chat Average shipping by region
plt.figure(figsize=(8,8))
plt.pie(
    shipping_by_Region['Shipping_Days'],       # numeric values
    labels=shipping_by_Region['Region'],       # category names
    autopct='%1.1f%%',                         # show % values
    startangle=90,                             # rotate start
    colors=plt.cm.rainbow(np.linspace(0, 1, len(shipping_by_Region)))  # color m
)
plt.title('Average Shipping Days by Region', fontsize=14)
plt.show()
```



In [44]:
```
# Sort by date
df = df.sort_values('Month_Year')
# group by month
monthly_sales = df.groupby(pd.Grouper(key='Month_Year'))['Sales'].sum().reset_in
```

In [45]:
```
monthly_sales
```

Out[45]:

	Month_Year	Sales
0	2015-01	14205.7070
1	2015-02	4519.8920
2	2015-03	55205.7970
3	2015-04	27906.8550
4	2015-05	23644.3030
5	2015-06	34322.9356
6	2015-07	33781.5430
7	2015-08	27117.5365
8	2015-09	81623.5268
9	2015-10	31453.3930
10	2015-11	77907.6607
11	2015-12	68167.0585
12	2016-01	18066.9576
13	2016-02	11951.4110
14	2016-03	32339.3184
15	2016-04	34154.4685
16	2016-05	29959.5305
17	2016-06	23599.3740
18	2016-07	28608.2590
19	2016-08	36818.3422
20	2016-09	63133.6060
21	2016-10	31011.7375
22	2016-11	70129.2995
23	2016-12	74543.6012
24	2017-01	16870.1810
25	2017-02	22978.8150
26	2017-03	51165.0590
27	2017-04	37385.0170
28	2017-05	56656.9080
29	2017-06	39724.4860
30	2017-07	38320.7830
31	2017-08	30542.2003
32	2017-09	69193.3909

	Month_Year	Sales
33	2017-10	59583.0330
34	2017-11	79066.4958
35	2017-12	95739.1210
36	2018-01	42839.2940
37	2018-02	19920.9974
38	2018-03	58863.4128
39	2018-04	35541.9101
40	2018-05	43825.9822
41	2018-06	48190.7277
42	2018-07	44825.1040
43	2018-08	62837.8480
44	2018-09	86152.8880
45	2018-10	77448.1312
46	2018-11	117938.1550
47	2018-12	82825.3588

In [46]:
```
# Convert Date → Numeric (for regression)
monthly_sales['Month_Num'] = range(1, len(monthly_sales) + 1)
```

In [47]:
```
# Train the Linear Regression Model
from sklearn.linear_model import LinearRegression

X = monthly_sales[['Month_Num']]   # independent variable
y = monthly_sales['Sales']         # dependent variable

model = LinearRegression()
model.fit(X, y)
```

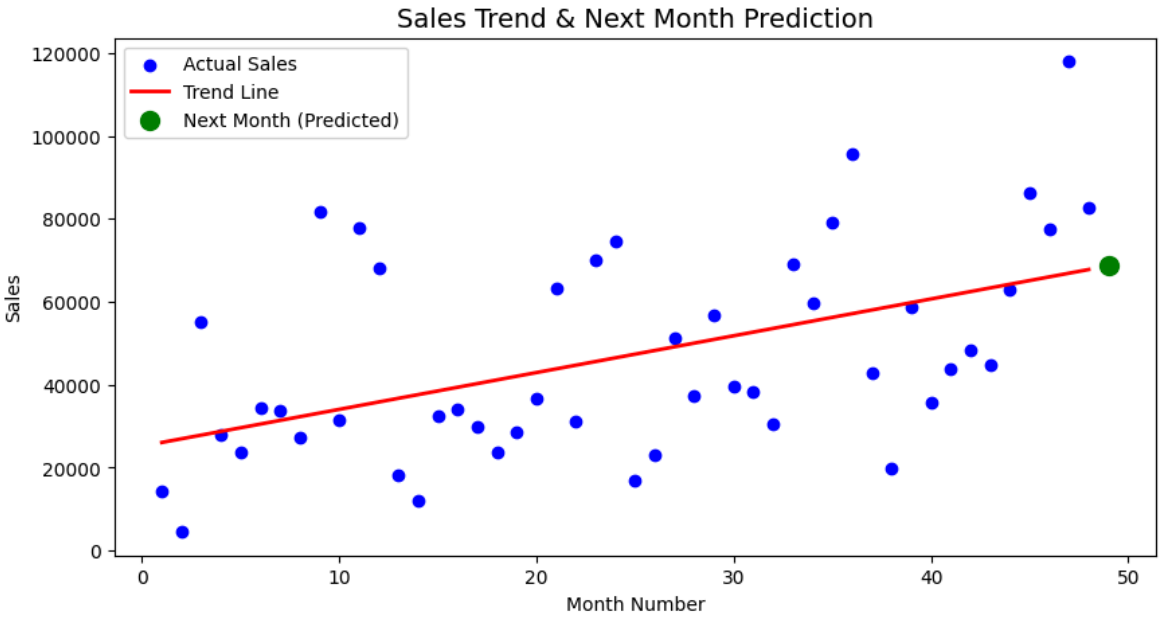
Out[47]:
```
▾ LinearRegression
LinearRegression()
```

In [48]:
```
# Predict Next Month's Sales
next_month = np.array([[len(monthly_sales) + 1]])
predicted_sales = model.predict(next_month)
print(f"Predicted sales for next month: ₹{predicted_sales[0]:.2f}")
```

```
Predicted sales for next month: ₹68665.12
```

In [49]:
```
# Visualize the trend
plt.figure(figsize=(10,5))
plt.scatter(X, y, color='blue', label='Actual Sales')
plt.plot(X, model.predict(X), color='red', linewidth=2, label='Trend Line')
plt.scatter(len(monthly_sales)+1, predicted_sales, color='green', s=100, label='
plt.title('Sales Trend & Next Month Prediction', fontsize=14)
```

```
plt.xlabel('Month Number')
plt.ylabel('Sales')
plt.legend()
plt.show()
```



In [50]:
```
# Insights:-
# South Region drive the most sales .
# No.1 product by sales is "Canon imageCLASS 2200 Advanced Copier" with sales am
# in categories Technology perform best.
# November drive most sales and February the lowest .
#
```

In [ ]:
```

```