Bharat Intern

Name : Shiva D. Mehenge

In [4]:
```
#import the packages
#pasndas is a Data Wrangling and manipulation library
import pandas as pd
import warnings
```

Wrangling Dataset

In [5]:
```
df = pd.read_csv("E:\\CSV Data\\train.csv")
```

In [6]:
```
df.head(3)
```

Out[6]:

omer ID	Customer Name	Segment	Country	City	State	Postal Code	Region	Product ID	Category
l2520	Claire Gute	Consumer	United States	Henderson	Kentucky	42420.0	South	FUR-BO-10001798	Furniture
l2520	Claire Gute	Consumer	United States	Henderson	Kentucky	42420.0	South	FUR-CH-10000454	Furniture
l3045	Darrin Van Huff	Corporate	United States	Los Angeles	California	90036.0	West	OFF-LA-10000240	Office Supplies

In [7]:
```
df.tail(3)
```

Out[7]:

Order Date	Ship Date	Ship Mode	Customer ID	Customer Name	Segment	Country	City	State	Postal Code
12/01/2016	17/01/2016	Standard Class	CS-12490	Cindy Schnelling	Corporate	United States	Toledo	Ohio	43615.0
12/01/2016	17/01/2016	Standard Class	CS-12490	Cindy Schnelling	Corporate	United States	Toledo	Ohio	43615.0
12/01/2016	17/01/2016	Standard Class	CS-12490	Cindy Schnelling	Corporate	United States	Toledo	Ohio	43615.0

Data Description

Describe - use for measure the Central Tendency (Mean, Madian , Standard Daviation , Max , 25%-50%-75%)

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

Out[8]:

	Row ID	Postal Code	Sales
count	9800.000000	9789.000000	9800.000000
mean	4900.500000	55273.322403	230.769059
std	2829.160653	32041.223413	626.651875
min	1.000000	1040.000000	0.444000
25%	2450.750000	23223.000000	17.248000
50%	4900.500000	58103.000000	54.490000
75%	7350.250000	90008.000000	210.605000
max	9800.000000	99301.000000	22638.480000

In [9]:
```
df.values
```

Out[9]:
```
array([[1, 'CA-2017-152156', '08/11/2017', ..., 'Bookcases',
        'Bush Somerset Collection Bookcase', 261.96],
       [2, 'CA-2017-152156', '08/11/2017', ..., 'Chairs',
        'Hon Deluxe Fabric Upholstered Stacking Chairs, Rounded Back',
        731.94],
       [3, 'CA-2017-138688', '12/06/2017', ..., 'Labels',
        'Self-Adhesive Address Labels for Typewriters by Universal',
        14.62],
       ...,
       [9798, 'CA-2016-128608', '12/01/2016', ..., 'Phones',
        'GE 30524EE4', 235.188],
       [9799, 'CA-2016-128608', '12/01/2016', ..., 'Phones',
        'Anker 24W Portable Micro USB Car Charger', 26.376],
       [9800, 'CA-2016-128608', '12/01/2016', ..., 'Accessories',
        'SanDisk Cruzer 4 GB USB Flash Drive', 10.384]], dtype=object)
```

In [10]:
```
df.columns
```

Out[10]:
```
Index(['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'],
      dtype='object')
```

In [11]:
```
df.nunique()
```

Out[11]:
```
Row ID           9800
Order ID         4922
Order Date       1230
Ship Date        1326
Ship Mode           4
Customer ID       793
Customer Name     793
Segment             3
Country             1
City              529
State              49
Postal Code       626
Region              4
Product ID       1861
Category            3
Sub-Category       17
Product Name     1849
Sales            5757
dtype: int64
```

In [12]:
```
df.shape
```

Out[12]:
```
(9800, 18)
```

In [13]:
```
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
```

Data Cleaning

In [12]:
```
df.isnull()
```

Out[12]:

	Row ID	Order ID	Order Date	Ship Date	Ship Mode	Customer ID	Customer Name	Segment	Country	City	State
0	False	False	False	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...	...	...
9795	False	False	False	False	False	False	False	False	False	False	False
9796	False	False	False	False	False	False	False	False	False	False	False
9797	False	False	False	False	False	False	False	False	False	False	False
9798	False	False	False	False	False	False	False	False	False	False	False
9799	False	False	False	False	False	False	False	False	False	False	False

9800 rows × 18 columns

In [13]:
```
df.isnull().sum()
```

Out[13]:
```
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
```

There are 11 null values are present in "Postal Code" column.

.isna() use for predicting the null value in the form of True & False, where True = null value & False = Fill value) .isna()sum() use for calculate (Addition) the null value.

In [14]:
```
# df = df.drop('Postal Code' , axis= 1)
value = df['Postal Code']
value = df.fillna(value.mode(), inplace = True)
```

In [15]:
```
df.isnull().sum()
```

Out[15]:
```
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 [16]:
```
df['Sales'].value_counts()
```

Out[16]:
```
12.960     55
15.552     39
19.440     39
10.368     35
25.920     34
           ..
339.136     1
60.048      1
5.022       1
7.857       1
10.384      1
Name: Sales, Length: 5757, dtype: int64
```

In [25]:
```
df['Region'].value_counts()
```

Out[25]:
```
West       3140
East       2785
Central    2277
South      1598
Name: Region, dtype: int64
```

In [29]:
```
df['Country'].value_counts()
```

Out[29]:
```
United States    9800
Name: Country, dtype: int64
```

df.nunique()

Data Encoding

There are Two type of Encoding 1) LabelEncoding and 2) One-hot Encoding

In [19]:
```
from sklearn.preprocessing import LabelEncoder
#creat an instead of LabelEncoder
le = LabelEncoder()
#column with LabelEncoder
df['Sales'] = le.fit_transform\
(df['Sales'])
```

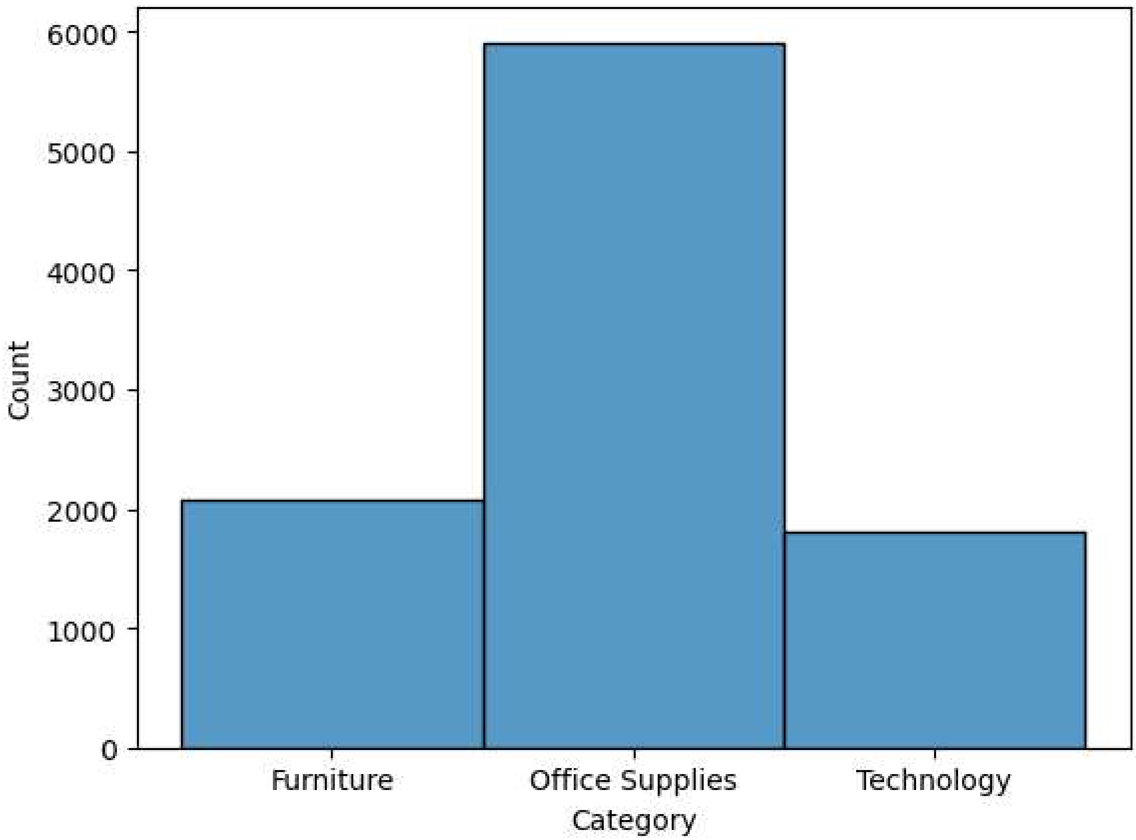
Data Visualization

Histogram

It can used both Uni and bivariate analysis.

In [21]:
```
#import packages
import matplotlib.pyplot as plt
import seaborn as sns

sns.histplot(x = 'Category' , data = df,)
plt.show()
```



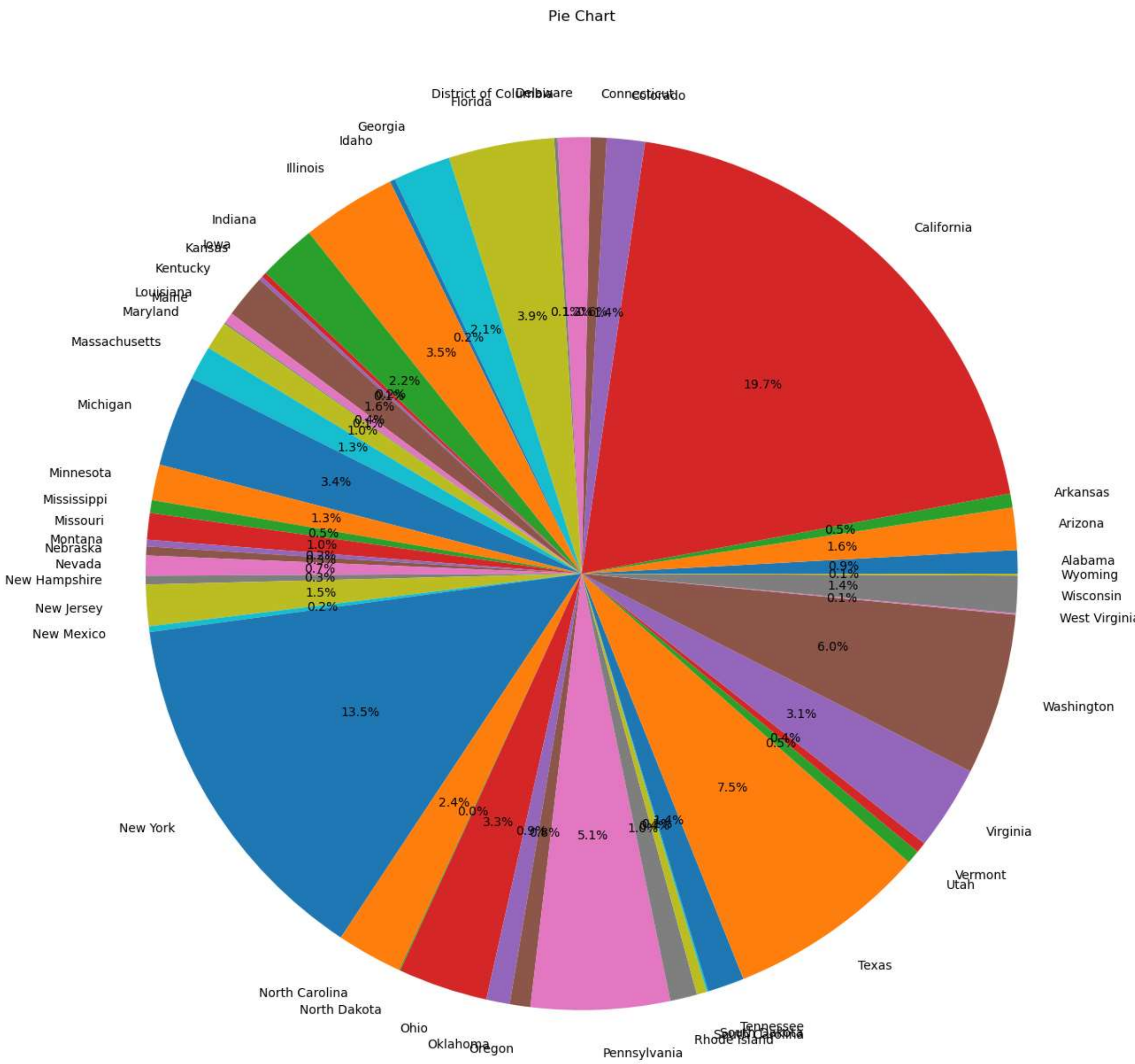
The Most Category across the sales is -

1. Office Supplies
2. Furniture
3. Technology

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

In [64]:
```
grouped_data = df.groupby('State')["Sales"].sum()

# Plot the pie chart
plt.figure(figsize=(16, 16))
plt.pie(grouped_data.values, labels=grouped_data.index, autopct='%1.1f%%')
plt.title('Pie Chart')
plt.show()
```



The Highest Sales is in Percentages

1. California State - 19.7%
2. New York State - 13.5%
3. Texas - 7.5%

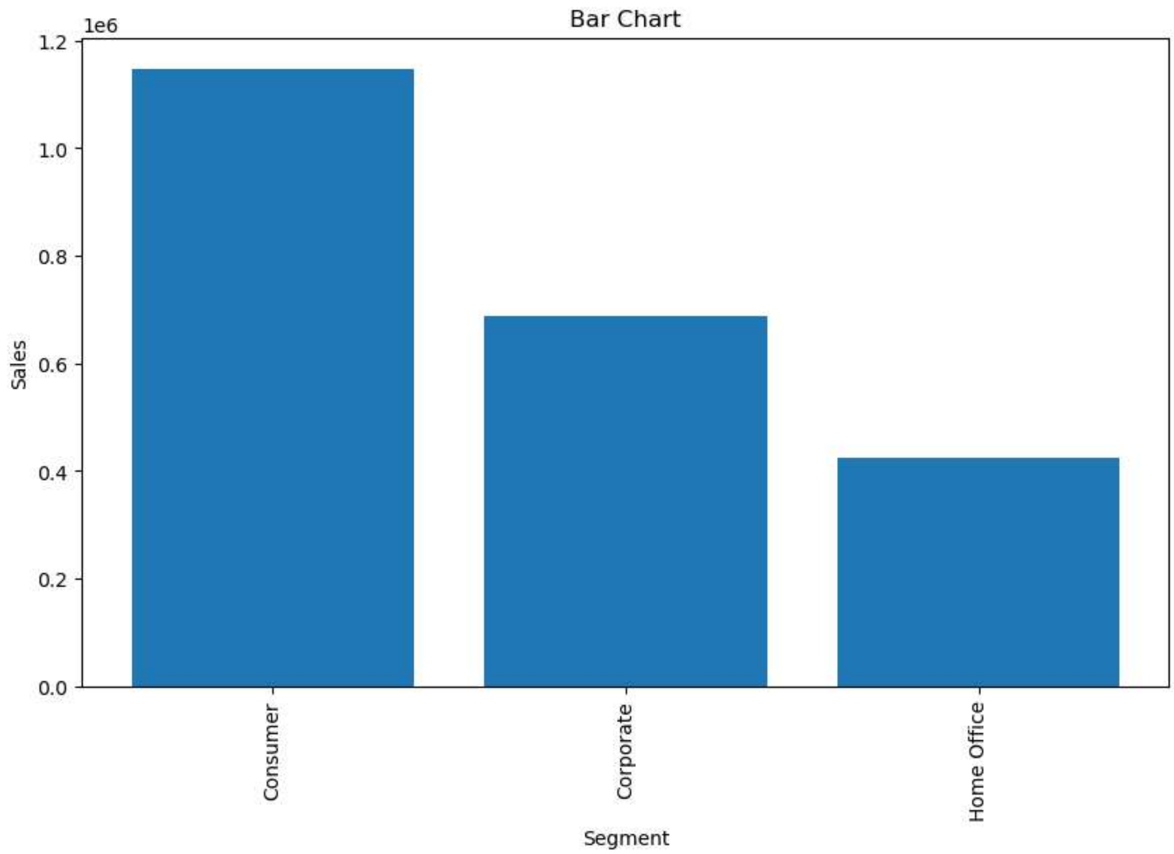
In [15]:

```
```

In [36]:

```
grouped_data = df.groupby('Segment')['Sales'].sum()

# Plot the bar chart
plt.figure(figsize=(10, 6))
plt.bar(grouped_data.index, grouped_data.values)
plt.xlabel('Segment')
plt.ylabel('Sales')
plt.title('Bar Chart')
plt.xticks(rotation=90)  # Rotate x-axis labels if needed
plt.show()
```

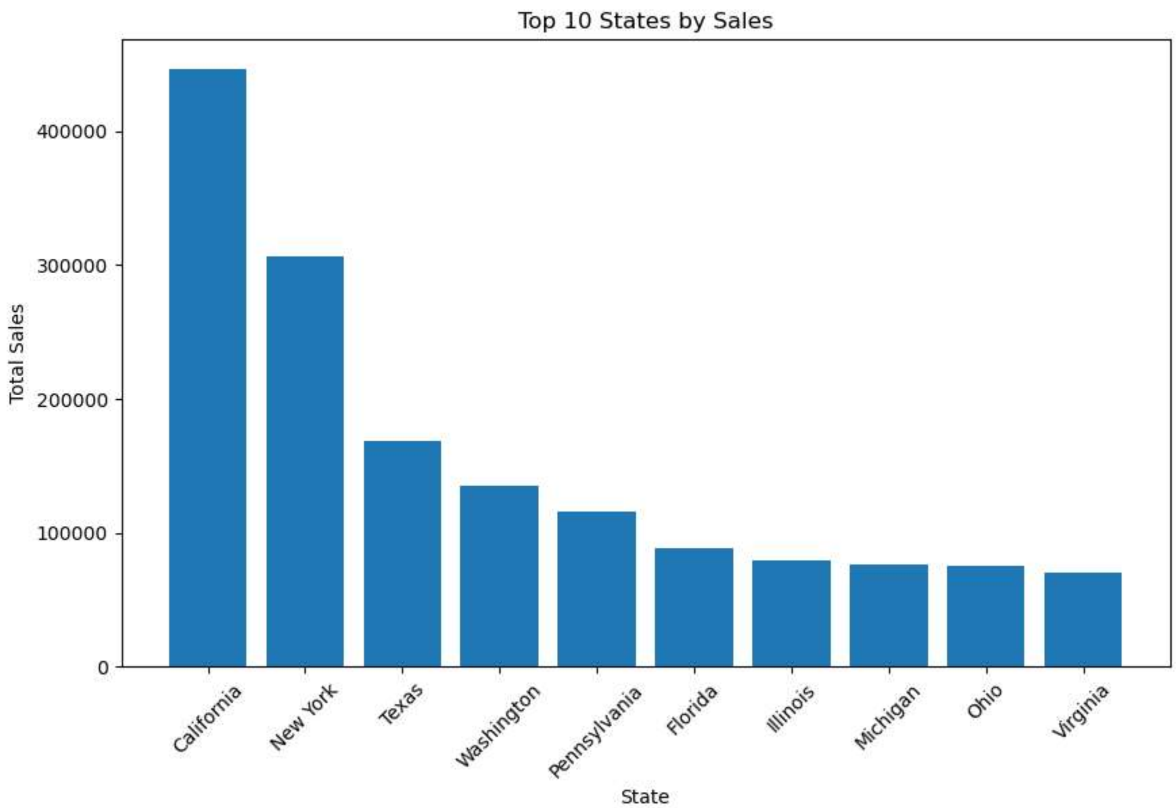


The highest sales is in the Segment is -

1. Consumer
2. Corporate
3. Home Office

In [52]:
```
grouped_data = df.groupby('State')['Sales'].sum()
top_10_states = grouped_data.nlargest(10)
```

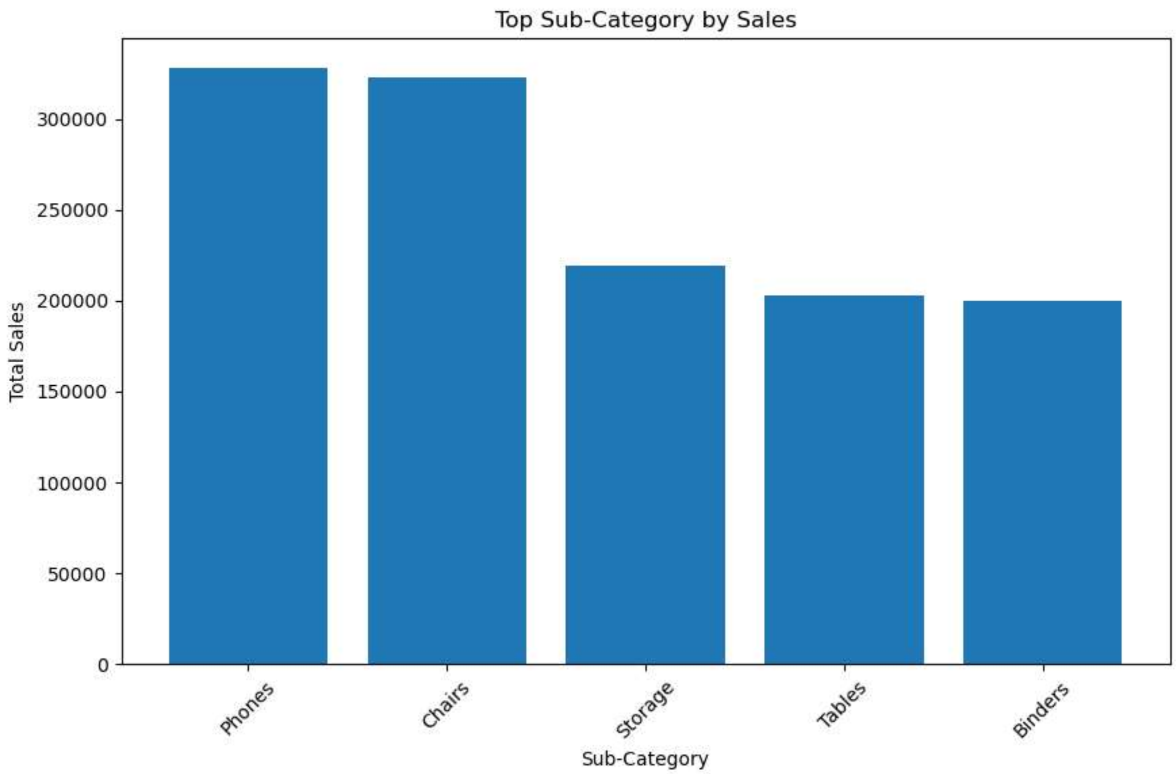
In [53]:
```
# Plot the bar chart
plt.figure(figsize=(10, 6))
plt.bar(top_10_states.index, top_10_states.values)
plt.xlabel('State')
plt.ylabel('Total Sales')
plt.title('Top 10 States by Sales')
plt.xticks(rotation=45)  # Rotate x-axis labels for better readability
plt.show()
```



Top 5 State with Highest Sales

1. California
2. New York
3. Texas
4. Washington
5. pennsylvania

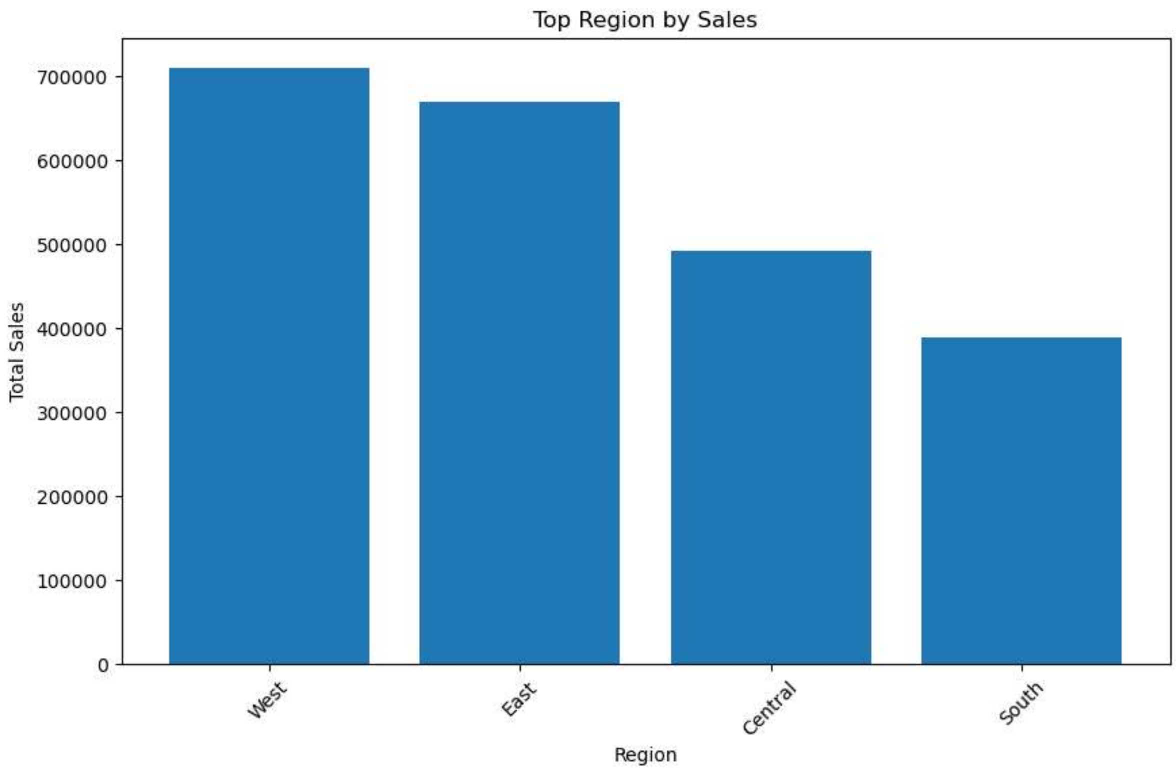
In [62]:
```
grouped_data = df.groupby('Sub-Category')['Sales'].sum()
top_10_states = grouped_data.nlargest(5)
    # Plot the bar chart
plt.figure(figsize=(10, 6))
plt.bar(top_10_states.index, top_10_states.values)
plt.xlabel('Sub-Category')
plt.ylabel('Total Sales')
plt.title('Top Sub-Category by Sales')
plt.xticks(rotation=45)  # Rotate x-axis labels for better readability
plt.show()
```



Top Sales according to Sub-Category

1. Phones
2. Chairs
3. Storage
4. Tables
5. Binders

In [60]:
```
grouped_data = df.groupby('Region')['Sales'].sum()
top_10_states = grouped_data.nlargest(5)
    # Plot the bar chart
plt.figure(figsize=(10, 6))
plt.bar(top_10_states.index, top_10_states.values)
plt.xlabel('Region')
plt.ylabel('Total Sales')
plt.title('Top Region by Sales')
plt.xticks(rotation=45)  # Rotate x-axis labels for better readability
plt.show()
```



The Top Region by Sales is-

1. West
2. East
3. Central
4. South

To summarize the outcome:

1. The analysis of a supermarket based on sales reveals significant disparities in sales performance among different countries, with some countries contributing more to the total sales than others.
2. The bar chart and pie chart visualizations effectively present the sales data, offering clear comparisons and insights into the distribution of sales among the top-performing countries and states.

3. Decision-makers can leverage these insights to make data-driven decisions, optimize business strategies, and allocate resources efficiently to maximize growth and profitability.

The outcome of this analysis provides valuable information for businesses to identify potential areas for growth, target high-performing regions, and address challenges in underperforming areas. By leveraging this data, businesses can position themselves for success in competitive markets and enhance overall performance and profitability.

Thank You!

In [ ]:

```

```