Bharat Intern [Task - 01]

Name : J.Harshit

Wrangling Dataset

In [1]:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

In [2]:

```
= pd.read_csv("C:/Users/harsh/OneDrive/Desktop/Bharat Intern(B A)/super_market_training_d
```

Out[2]:

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

9800 rows × 18 columns

In [3]:

```
df.head()
#It gets data's first 5 rows quickly
```

Out[3]:

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

In [4]:

```
df.tail()
#It gets data's last 5 rows quickly
```

Out[4]:

	Row ID	Order ID	Order Date	Ship Date	Ship Mode	Customer ID	Customer Name	Segment	Country	Cit
9795	9796	CA-2017-125920	21-05-2017	28-05-2017	Standard Class	SH-19975	Sally Hughsby	Corporate	United States	Chicag
9796	9797	CA-2016-128608	12-01-2016	17-01-2016	Standard Class	CS-12490	Cindy Schnelling	Corporate	United States	Toled
9797	9798	CA-2016-128608	12-01-2016	17-01-2016	Standard Class	CS-12490	Cindy Schnelling	Corporate	United States	Toled
9798	9799	CA-2016-128608	12-01-2016	17-01-2016	Standard Class	CS-12490	Cindy Schnelling	Corporate	United States	Toled
9799	9800	CA-2016-128608	12-01-2016	17-01-2016	Standard Class	CS-12490	Cindy Schnelling	Corporate	United States	Toled

In [5]:

```
sample = df.sample(10)
sample
#It gets random sample of the number you write between ()
```

115	116	CA-2015-115259	25-08-2015	27-08-2015	Second Class	RC-19960	Ryan Crowe	Consumer	United States	Columbus	Ohi
6754	6755	CA-2017-154767	28-06-2017	30-06-2017	Second Class	BP-11155	Becky Pak	Consumer	United States	Paterson	Nev Jerse
6124	6125	CA-2018-145772	03-06-2018	07-06-2018	Standard Class	SS-20140	Saphhira Shifley	Corporate	United States	Los Angeles	Californi
8243	8244	CA-2015-109897	12-08-2015	16-08-2015	Standard Class	BW-11200	Ben Wallace	Consumer	United States	San Francisco	Californi
873	874	CA-2015-	10-12-	15-12-	Standard Class	SM-20005	Sally Matthias	Consumer	United States	New York City	New Yor

Data Description

In [6]:

```
df.describe(include = "object")
```

Out[6]:

	Order ID	Order Date	Ship Date	Ship Mode	Customer ID	Customer Name	Segment	Country	City	S
count	9800	9800	9800	9800	9800	9800	9800	9800	9800	
unique	4922	1230	1326	4	793	793	3	1	529	
top	CA-2018-100111	05-09-2017	26-09-2018	Standard Class	WB-21850	William Brown	Consumer	United States	New York City	Calif
freq	14	38	34	5859	35	35	5101	9800	891	

In [7]:

```
df.isnull().sum()
#It shows the number of the missing values in data if exists
#In this data there is 11 missing values in the postal code
```

Out[7]:

```
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 [8]:

```
df[df['Postal Code'].isnull()]
#Here it shows the the missing values in the postal code colmn themselves
```

Out[8]:

	Row ID	Order ID	Order Date	Ship Date	Ship Mode	Customer ID	Customer Name	Segment	Country	C
2234	2235	CA-2018-104066	05-12-2018	10-12-2018	Standard Class	QJ-19255	Quincy Jones	Corporate	United States	Burling
5274	5275	CA-2016-162887	07-11-2016	09-11-2016	Second Class	SV-20785	Stewart Visinsky	Consumer	United States	Burling
8798	8799	US-2017-150140	06-04-2017	10-04-2017	Standard Class	VM-21685	Valerie Mitchum	Home Office	United States	Burling
9146	9147	US-2017-165505	23-01-2017	27-01-2017	Standard Class	CB-12535	Claudia Bergmann	Corporate	United States	Burling
9147	9148	US-2017-165505	23-01-2017	27-01-2017	Standard Class	CB-12535	Claudia Bergmann	Corporate	United States	Burling
9148	9149	US-2017-165505	23-01-2017	27-01-2017	Standard Class	CB-12535	Claudia Bergmann	Corporate	United States	Burling
9386	9387	US-2018-127292	19-01-2018	23-01-2018	Standard Class	RM-19375	Raymond Messe	Consumer	United States	Burling
9387	9388	US-2018-127292	19-01-2018	23-01-2018	Standard Class	RM-19375	Raymond Messe	Consumer	United States	Burling
9388	9389	US-2018-127292	19-01-2018	23-01-2018	Standard Class	RM-19375	Raymond Messe	Consumer	United States	Burling
9389	9390	US-2018-127292	19-01-2018	23-01-2018	Standard Class	RM-19375	Raymond Messe	Consumer	United States	Burling
9741	9742	CA-2016-117086	08-11-2016	12-11-2016	Standard Class	QJ-19255	Quincy Jones	Corporate	United States	Burling

In [9]:

```
df[(df.City == 'Burlington') & (df.State == 'Vermont')]
#as it shown the postal code is
```

Out[9]:

	Row ID	Order ID	Order Date	Ship Date	Ship Mode	Customer ID	Customer Name	Segment	Country	C
2234	2235	CA-2018-104066	05-12-2018	10-12-2018	Standard Class	QJ-19255	Quincy Jones	Corporate	United States	Burling
5274	5275	CA-2016-162887	07-11-2016	09-11-2016	Second Class	SV-20785	Stewart Visinsky	Consumer	United States	Burling
8798	8799	US-2017-150140	06-04-2017	10-04-2017	Standard Class	VM-21685	Valerie Mitchum	Home Office	United States	Burling
9146	9147	US-2017-165505	23-01-2017	27-01-2017	Standard Class	CB-12535	Claudia Bergmann	Corporate	United States	Burling
9147	9148	US-2017-165505	23-01-2017	27-01-2017	Standard Class	CB-12535	Claudia Bergmann	Corporate	United States	Burling
9148	9149	US-2017-165505	23-01-2017	27-01-2017	Standard Class	CB-12535	Claudia Bergmann	Corporate	United States	Burling
9386	9387	US-2018-127292	19-01-2018	23-01-2018	Standard Class	RM-19375	Raymond Messe	Consumer	United States	Burling
9387	9388	US-2018-127292	19-01-2018	23-01-2018	Standard Class	RM-19375	Raymond Messe	Consumer	United States	Burling
9388	9389	US-2018-127292	19-01-2018	23-01-2018	Standard Class	RM-19375	Raymond Messe	Consumer	United States	Burling
9389	9390	US-2018-127292	19-01-2018	23-01-2018	Standard Class	RM-19375	Raymond Messe	Consumer	United States	Burling
9741	9742	CA-2016-117086	08-11-2016	12-11-2016	Standard Class	QJ-19255	Quincy Jones	Corporate	United States	Burling

In [10]:

```
df['Postal Code'] = df['Postal Code'].fillna('05402')
df[(df.City == 'Burlington') & (df.State == 'Vermont')]
```

Out[10]:

	Row ID	Order ID	Order Date	Ship Date	Ship Mode	Customer ID	Customer Name	Segment	Country	City	State
2234	2235	CA-2018-104066	05-12-2018	10-12-2018	Standard Class	QJ-19255	Quincy Jones	Corporate	United States	Burlington	Vermont
5274	5275	CA-2016-162887	07-11-2016	09-11-2016	Second Class	SV-20785	Stewart Visinsky	Consumer	United States	Burlington	Vermont
8798	8799	US-2017-150140	06-04-2017	10-04-2017	Standard Class	VM-21685	Valerie Mitchum	Home Office	United States	Burlington	Vermont

In [11]:

```
df.values
```

Out[11]:

```
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 [12]:

```
df.columns
```

Out[12]:

```
Index(['Row ID', 'Order ID', 'Order Date', 'Ship Date', 'Ship Mode',
       'Customer ID', 'Customer Name', 'Segment', 'Country', 'City', 'Stat
e',
       'Postal Code', 'Region', 'Product ID', 'Category', 'Sub-Category',
       'Product Name', 'Sales'],
      dtype='object')
```

In [13]:

```
df.shape
```

Out[13]:

```
(9800, 18)
```

In [14]:

```
df.info()
#It is a method to get general information about dataset(column's name, feature's type)
#and check if there is any missing information in the dataset.
```

```
<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    9800 non-null   object 
 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(1), int64(1), object(16)
memory usage: 1.3+ MB
```

Data Cleaning

In [15]:

```
df.isnull()
```

Out[15]:

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

9800 rows × 18 columns

In [16]:

```
df.isnull().sum()
```

Out[16]:

```
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     0
Region          0
Product ID      0
Category        0
Sub-Category    0
Product Name    0
Sales           0
dtype: int64
```

In [17]:

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

In [18]:

```
df.isnull().sum()
```

Out[18]:

```
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      0
Region           0
Product ID       0
Category         0
Sub-Category     0
Product Name     0
Sales            0
dtype: int64
```

In [19]:

```
df['Sales'].value_counts()
```

Out[19]:

```
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 [20]:

```
df['Region'].value_counts()
```

Out[20]:

```
West       3140
East       2785
Central    2277
South      1598
Name: Region, dtype: int64
```

In [21]:

```
df['Country'].value_counts()
```

Out[21]:

```
United States    9800
Name: Country, dtype: int64
```

In [22]:

```
df.nunique()
```

Out[22]:

```
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       627
Region              4
Product ID       1861
Category            3
Sub-Category       17
Product Name     1849
Sales            5757
dtype: int64
```

Data Encoding

There are Two type of Encoding:

- LabelEncoding
- One-hot Encoding

In [23]:

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

Data Visualization

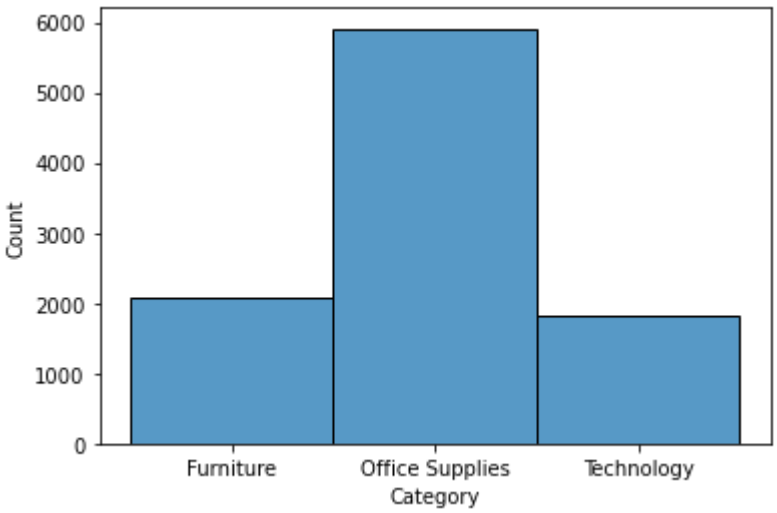
Histogram

It can used both Uni and bivariate analysis.

In [24]:

```
import matplotlib.pyplot as plt
import seaborn as sns

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



The Most Category across the sales is -

- office supplies
- Furniture
- Technology

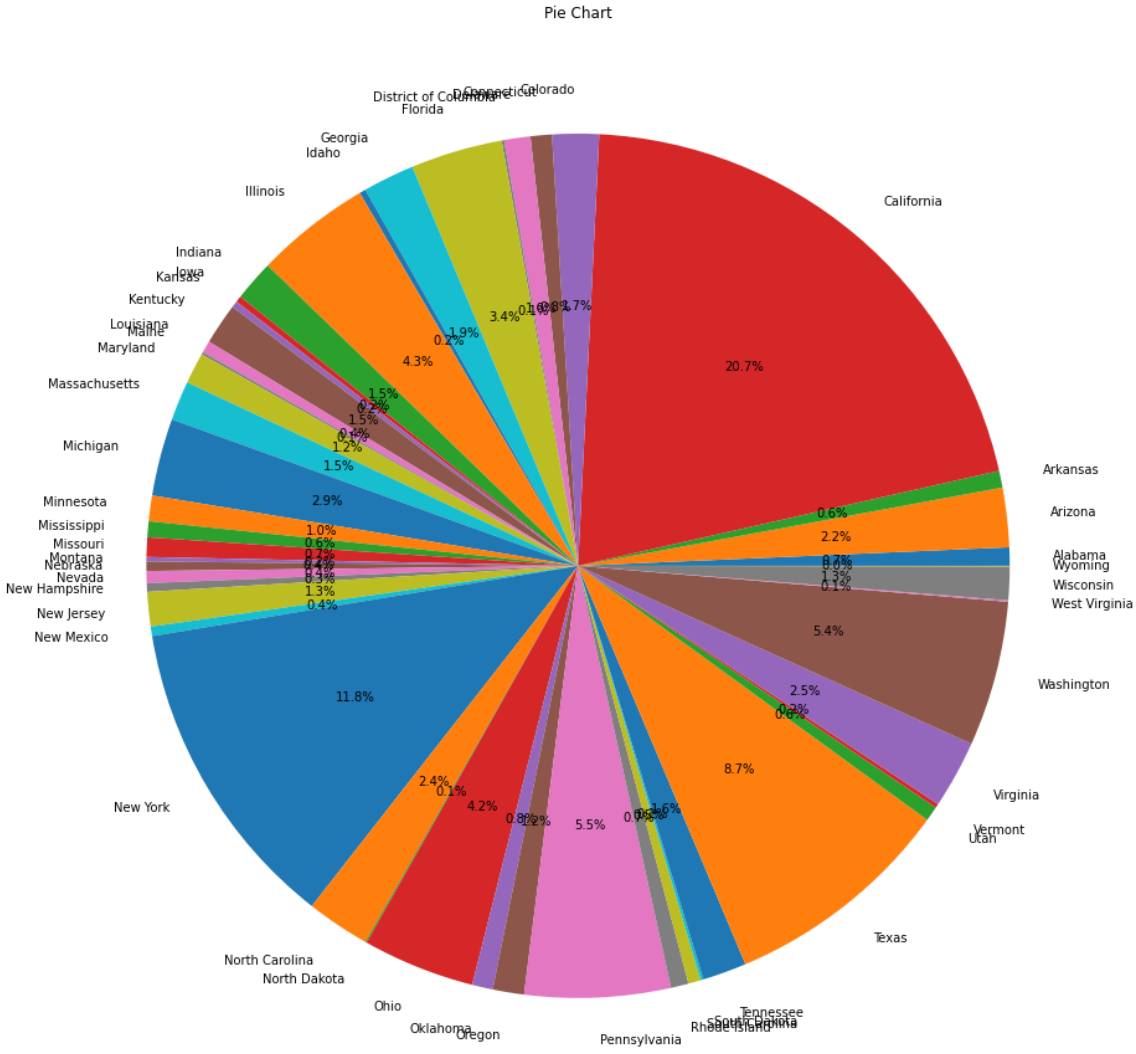
In [25]:

```
import pandas as pd
import matplotlib.pyplot as plt
```

In [26]:

```
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()
```



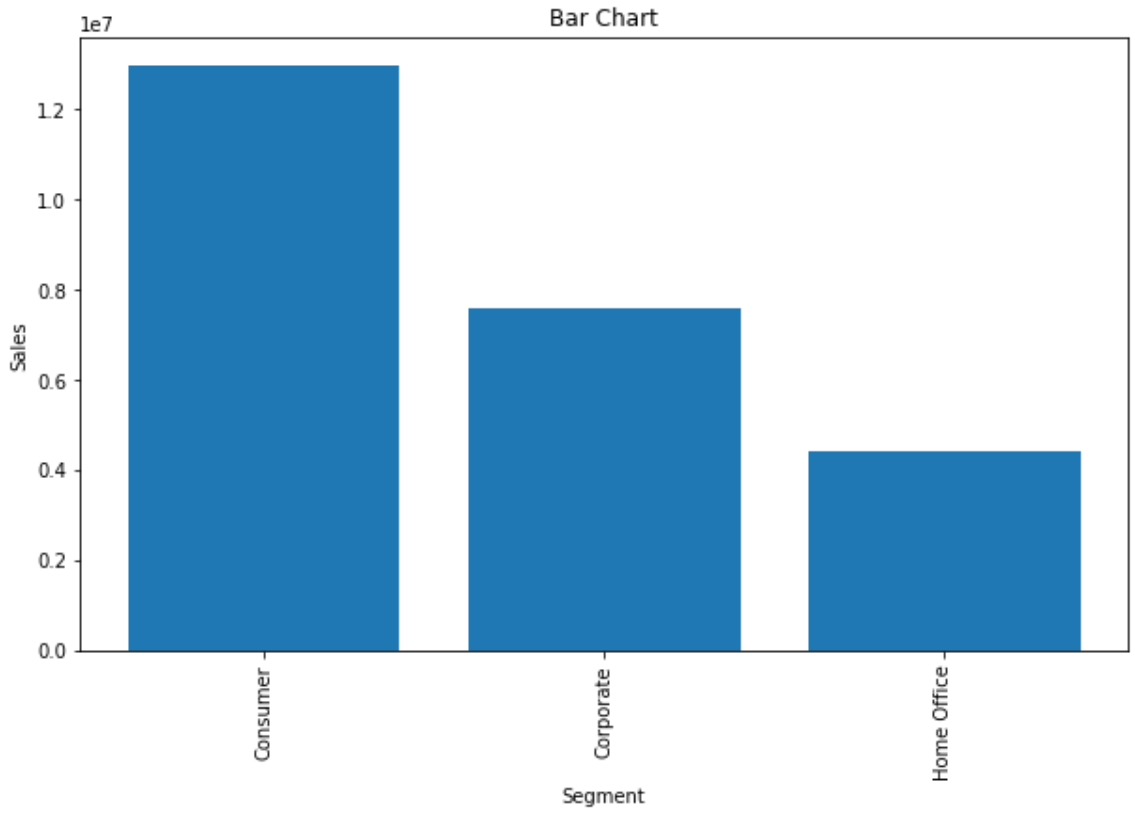
The Highest Sales is in Percentages

- California State - 19.7%
- New York State - 13.5%
- Texas - 7.5%

In [27]:

```
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 -

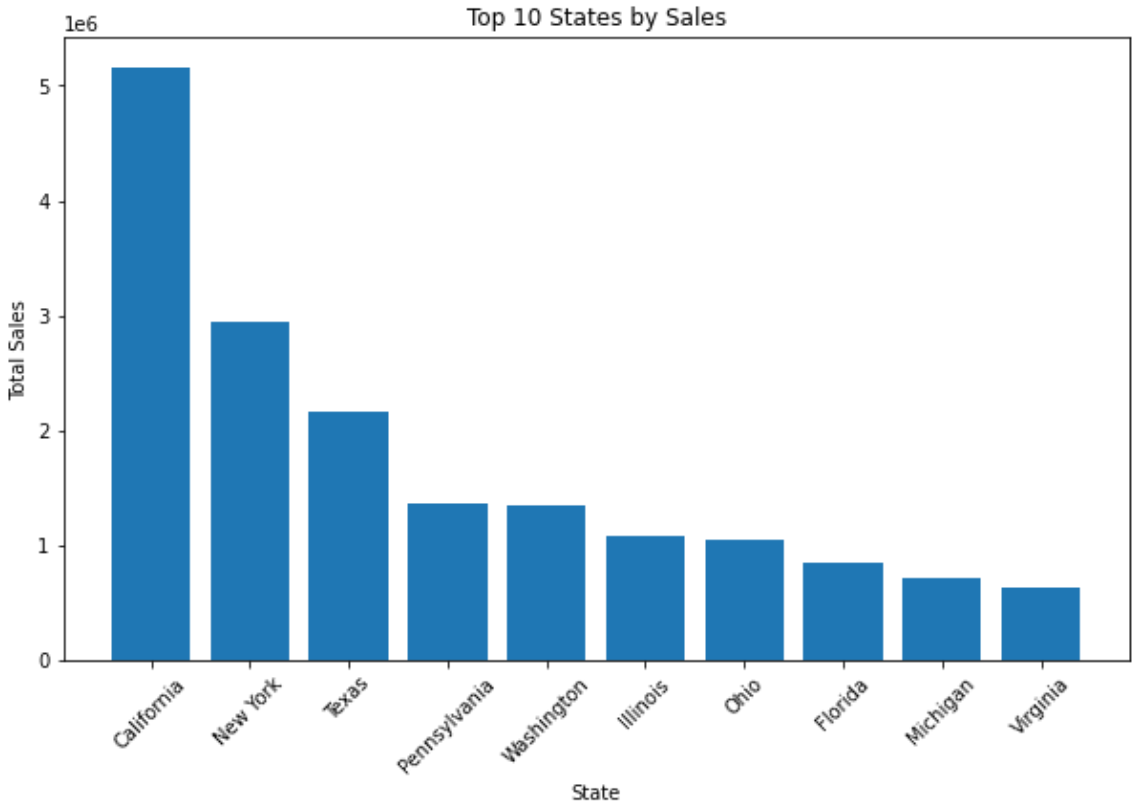
- Consumer
- Corporate
- Home Office

In [28]:

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

In [29]:

```
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()
```

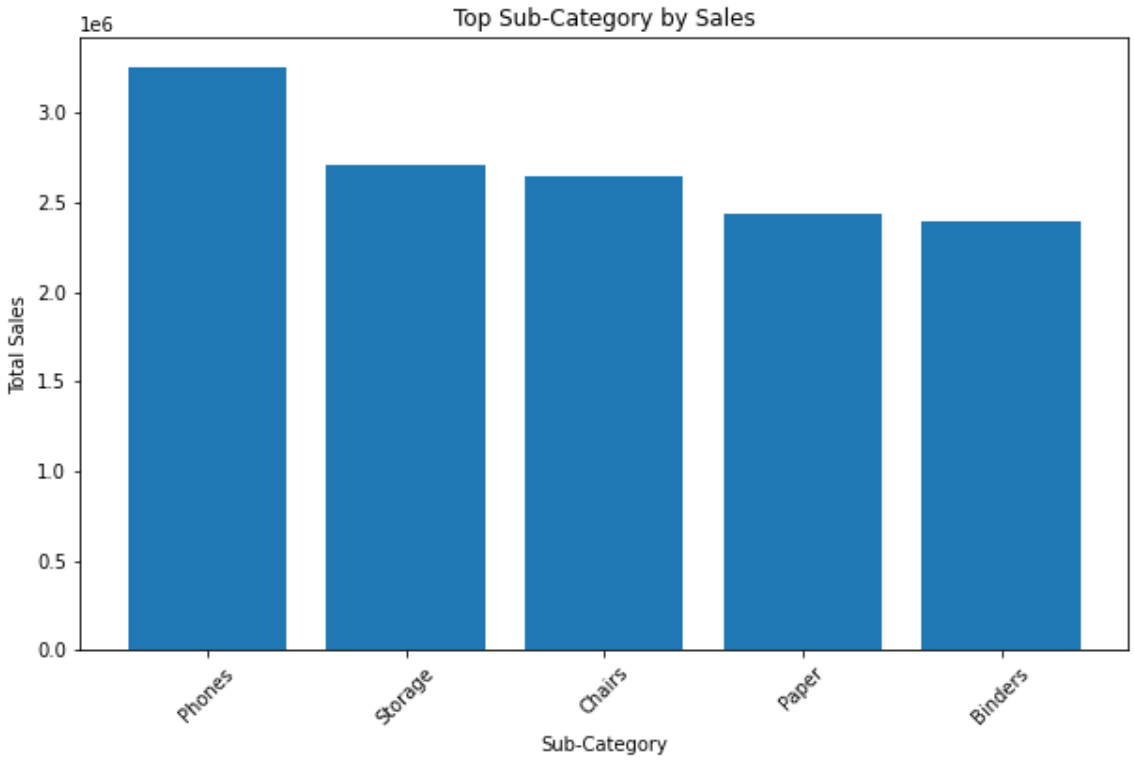


Top 5 State with Highest Sales

- California
- New York
- Texas
- Washington
- pennsylvania

In [30]:

```
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()
```

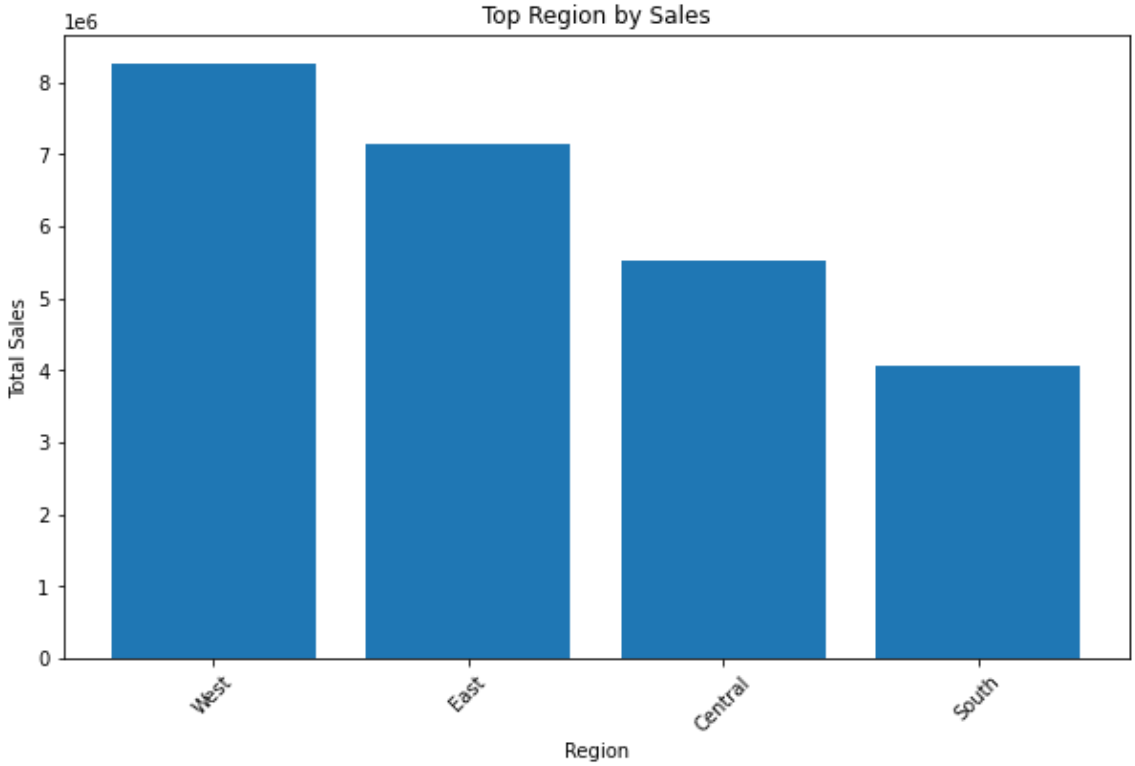


Top Sales according to Sub-Category

- Phones
- Chairs
- Storage
- Tables
- Binders

In [31]:

```
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()
```



Which products are most bought the cheapest or most expensive :

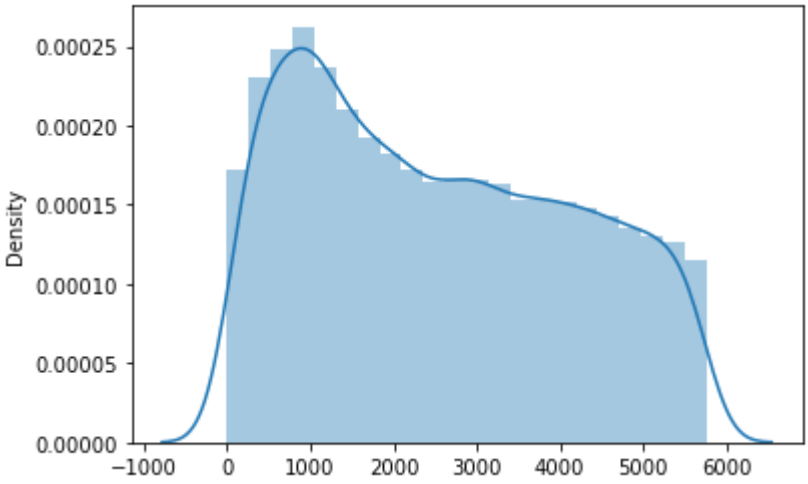
In [32]:

```
sns.distplot(x = df['Sales'])
```

```
C:\Users\harsh\anaconda3\lib\site-packages\seaborn\distributions.py:2619:
FutureWarning: `distplot` is a deprecated function and will be removed in
a future version. Please adapt your code to use either `displot` (a figure
-level function with similar flexibility) or `histplot` (an axes-level fun
ction for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[32]:

```
<AxesSubplot:ylabel='Density'>
```

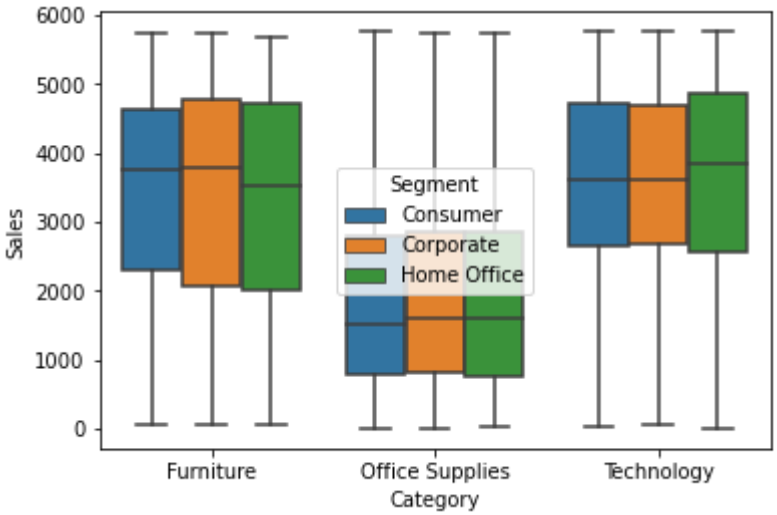


In [33]:

```
sns.boxplot(data = df, x = "Category", y = "Sales", hue = "Segment")
```

Out[33]:

```
<AxesSubplot:xlabel='Category', ylabel='Sales'>
```



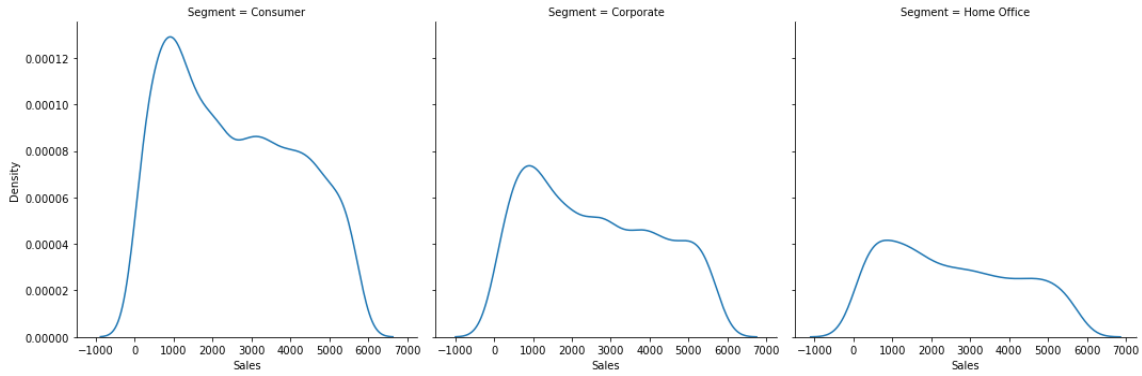
which segment has the most sales :

In [34]:

```
sns.displot(data = df, x = "Sales", col = "Segment", kind = "kde")
```

Out[34]:

```
<seaborn.axisgrid.FacetGrid at 0x1c6797370d0>
```



In [35]:

```
df['Segment'].value_counts()
```

Out[35]:

```
Consumer       5101
Corporate      2953
Home Office    1746
Name: Segment, dtype: int64
```

Which Shiping mode are most frequently used :

In [36]:

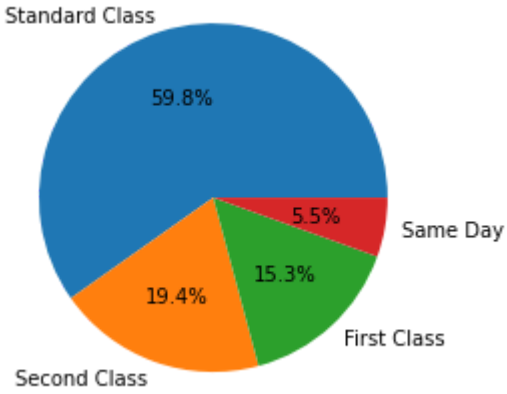
```
shipping=df['Ship Mode'].value_counts()
shipping
```

Out[36]:

```
Standard Class    5859
Second Class      1902
First Class       1501
Same Day           538
Name: Ship Mode, dtype: int64
```

In [37]:

```
plt.pie(shipping, labels = shipping.index, autopct = '%0.1f%%')
plt.show()
```



Which State and City have the most profits :

In [38]:

```
df['City'].value_counts()
```

Out[38]:

```
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: City, Length: 529, dtype: int64
```

In [39]:

```
df['State'].value_counts()
```

Out[39]:

```
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: State, dtype: int64
```

In [*]:

```
sns.barplot(data = df, x= 'State',y='Sales')
plt.xticks(rotation=90);
```

What are the daily sales :

In [*]:

```
data_daily_sales = data.groupby('Order Date').agg({'Sales':"sum"}).reset_index()
#I grouped  my dataset by "Order Date" and I want that according to "Sales" sum.
data_daily_sales
```

The Top Region by Sales is-

- West
- East
- Central
- South

To summarize the outcome:

- This comprehensive analysis not only sheds light on the disparities in sales performance among different countries but also offers a roadmap for strategic improvements.
- Armed with a deeper understanding of sales distribution, companies can tailor their marketing efforts to cater specifically to regions with high sales potential, while also devising targeted plans to revitalize areas where performance is lagging.
- The visualizations serve as a powerful tool to communicate findings across teams, fostering a shared comprehension that can drive collaborative efforts towards achieving both short-term targets and long-term expansion goals.
- Ultimately, this data-driven approach equips businesses with the competitive edge needed to adapt swiftly to market dynamics and fuel sustainable success in a dynamic global landscape.