# Innobyte Services - Data Analyst Internship

> ## Data Science & Business Analytics

This Notebook performs EDA (Exploratory Data Analysis) on SampleSuperstore Data.

## Task (Level - Beginner)

- Analyze retail sales data to derive insights into customer behavior, popular products, and sales trends ***.

> ## About The Data
>
> ## EDA on SampleSuperstore Data
>
> ## Objective

- This 'SampleSuperstore' data contains 13 columns and 9,994 rows
- Data source: Link
- Data Format: .csv


### Intern - Aryan Gautam

### Inter Id - IS/A1/A6297

### Contact-no: 9024309427


**Exploratory Data Analysis (EDA)**

- It is used by data scientists to analyze and investigate data sets and summarize their main characteristics, often employing data visualization methods.
- It helps determine how best to manipulate data sources to get the answers you need, making it easier for data scientists to discover patterns, spot anomalies, test a hypothesis, or check assumptions.

## Table of Content

In [287…
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

# 1.Data Exploration:

## 1.1 Load the dataset into your preferred data analysis tool.

In [288…
```
df=pd.read_csv("SampleSuperstore.csv")
```

In [289…
```
df.sample(5)
```

Out[289…

| | Ship Mode | Segment | Country | City | State | Postal Code | Region | Catego |
|---|---|---|---|---|---|---|---|---|
| **163** | Standard Class | Consumer | United States | Seattle | Washington | 98115 | West | Offi Suppli |
| **6153** | First Class | Corporate | United States | Philadelphia | Pennsylvania | 19120 | East | Furnitu |
| **2792** | First Class | Corporate | United States | Omaha | Nebraska | 68104 | Central | Offi Suppli |
| **3495** | Standard Class | Corporate | United States | Saint Cloud | Minnesota | 56301 | Central | Furnitu |
| **5510** | Standard Class | Consumer | United States | Chicago | Illinois | 60653 | Central | Technolo |

## 1.2 Explore the structure of the dataset.

In [290…
```
print(f"dimension:{df.shape}")
```

```
dimension:(9994, 13)
```

In [291…
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9994 entries, 0 to 9993
Data columns (total 13 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   Ship Mode     9994 non-null   object 
 1   Segment       9994 non-null   object 
 2   Country       9994 non-null   object 
 3   City          9994 non-null   object 
 4   State         9994 non-null   object 
 5   Postal Code   9994 non-null   int64  
 6   Region        9994 non-null   object 
 7   Category      9994 non-null   object 
 8   Sub-Category  9994 non-null   object 
 9   Sales         9994 non-null   float64
 10  Quantity      9994 non-null   int64  
 11  Discount      9994 non-null   float64
 12  Profit        9994 non-null   float64
dtypes: float64(3), int64(2), object(8)
memory usage: 1015.1+ KB
```

In [292…
```
print(f"coloumns names:", df.columns)
```

```
coloumns names: Index(['Ship Mode', 'Segment', 'Country', 'City', 'State', 'Posta
l Code',
       'Region', 'Category', 'Sub-Category', 'Sales', 'Quantity', 'Discount',
       'Profit'],
      dtype='object')
```

In [293…
```
df.describe().T
```

Out[293…

| | count | mean | std | min | 25% | 50% | |
|---|---|---|---|---|---|---|---|
| **Postal Code** | 9994.0 | 55190.379428 | 32063.693350 | 1040.000 | 23223.00000 | 56430.5000 | 90008 |
| **Sales** | 9994.0 | 229.858001 | 623.245101 | 0.444 | 17.28000 | 54.4900 | 209 |
| **Quantity** | 9994.0 | 3.789574 | 2.225110 | 1.000 | 2.00000 | 3.0000 | 5 |
| **Discount** | 9994.0 | 0.156203 | 0.206452 | 0.000 | 0.00000 | 0.2000 | 0 |
| **Profit** | 9994.0 | 28.656896 | 234.260108 | -6599.978 | 1.72875 | 8.6665 | 29 |

In [294…

```
print("No Null values in any column:")
print(df.isnull().sum())
```

```
No Null values in any column:
Ship Mode       0
Segment         0
Country         0
City            0
State           0
Postal Code     0
Region          0
Category        0
Sub-Category    0
Sales           0
Quantity        0
Discount        0
Profit          0
dtype: int64
```

In [295…

```
df.nunique()
```

Out[295…

```
Ship Mode          4
Segment            3
Country            1
City             531
State             49
Postal Code      631
Region             4
Category           3
Sub-Category      17
Sales           5825
Quantity          14
Discount          12
Profit          7287
dtype: int64
```

In [296…

```
print(f"{df.duplicated().sum()} duplicates rows found")
```

```
17 duplicates rows found
```

In [297…

```
df.drop_duplicates()
```

Out[297...

| | Ship Mode | Segment | Country | City | State | Postal Code | Region | Category |
|---|---|---|---|---|---|---|---|---|
| **0** | Second Class | Consumer | United States | Henderson | Kentucky | 42420 | South | Furniture |
| **1** | Second Class | Consumer | United States | Henderson | Kentucky | 42420 | South | Furniture |
| **2** | Second Class | Corporate | United States | Los Angeles | California | 90036 | West | Office Supplies |
| **3** | Standard Class | Consumer | United States | Fort Lauderdale | Florida | 33311 | South | Furniture |
| **4** | Standard Class | Consumer | United States | Fort Lauderdale | Florida | 33311 | South | Office Supplies |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... |
| **9989** | Second Class | Consumer | United States | Miami | Florida | 33180 | South | Furniture |
| **9990** | Standard Class | Consumer | United States | Costa Mesa | California | 92627 | West | Furniture |
| **9991** | Standard Class | Consumer | United States | Costa Mesa | California | 92627 | West | Technology |
| **9992** | Standard Class | Consumer | United States | Costa Mesa | California | 92627 | West | Office Supplies |
| **9993** | Second Class | Consumer | United States | Westminster | California | 92683 | West | Office Supplies |

9977 rows × 13 columns

In [298...

```
# seperating numerical & catogerical columns
num_col=[]
cat_col=[]
for x in df.columns:
    if (df[x].dtype == 'object'):
        cat_col.append(x)
    else:
        num_col.append(x)
```

In [299...

```
# Unique Values
print("Unique Values:")
for column in cat_col:
    print(column + ":", df[column].nunique())
```

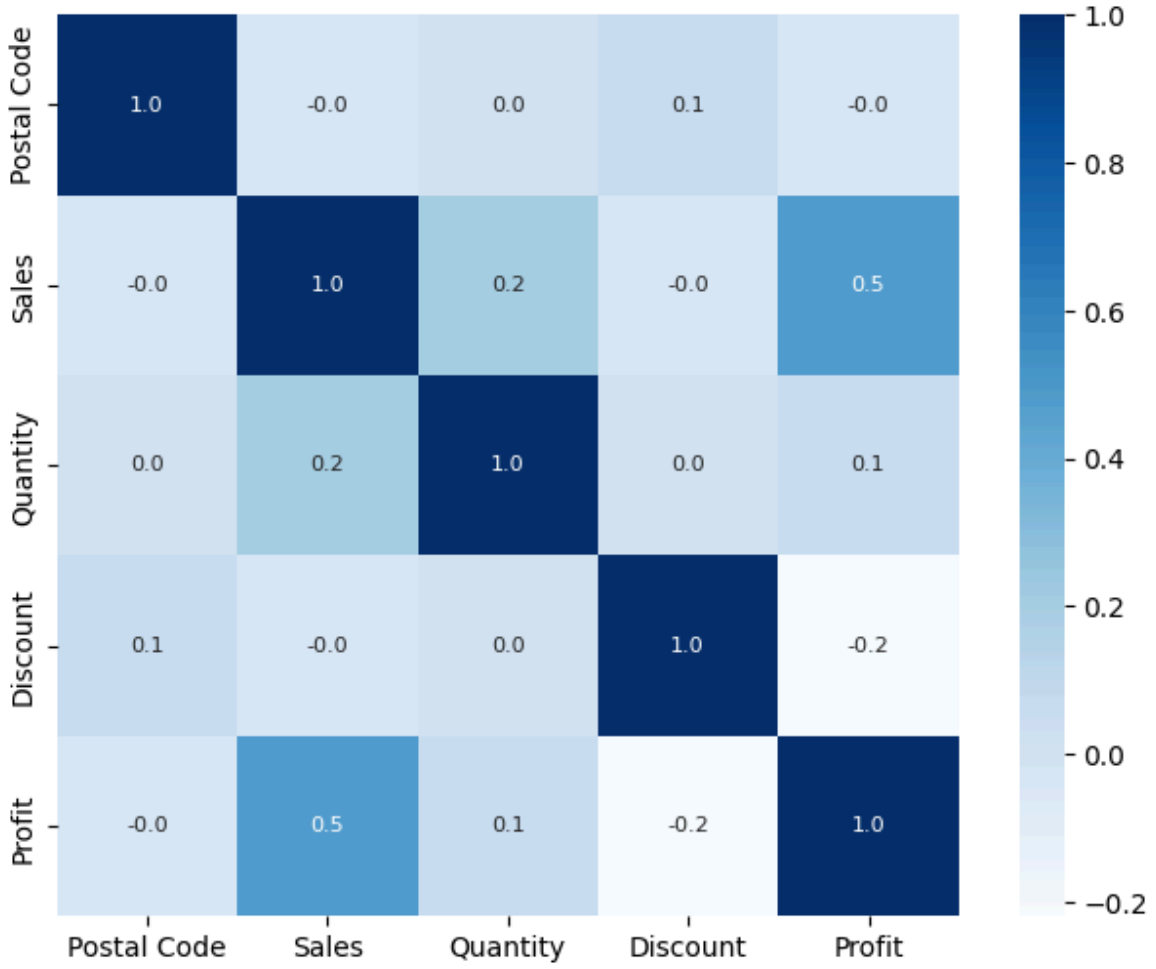
```
Unique Values:
Ship Mode: 4
Segment: 3
Country: 1
City: 531
State: 49
Region: 4
Category: 3
Sub-Category: 17
```

In [300…
```
df_nm=df.copy()
df_nm.drop(columns=['Ship Mode','Segment', 'Country', 'City', 'State', 'Region',
```

In [301…
```
correlation=df_nm.corr()
```

In [302…
```
plt.figure(figsize=(8, 6))
sns.heatmap(correlation, fmt='.1f', cbar=True, square=True, annot=True,  cmap='B
```

Out[302… <Axes: >




Conclusion:

1. Postal Code are not at all correlated to other columns.
2. Sales is positively correlated to Quantity & Profit -> 0.2 & 0.5 resp.
3. Discount has only effect on and Profit are highly negatively correlated to each other.

- We can clearly see that "Sales" and "Discount" have a direct impact on our "Profit," with Sales showing a positive correlation and Discount showing a negative correlation, respectively. This is expected because as sales rise, profits rise as well, and as discounts are offered on sales, profit margins fall.
- Additionally, we can see that there is a positive correlation between "Sales" and "Quantity," which also makes sense because the more quantity, the greater the amount of sales, but this correlation appears to be weak.

- We can also observe that there is little to no difference in sales and quantity due to discounts, proving that the discount strategy is ineffective for certain items. As a result, Superstore needs to rethink its strategies for luring customers.

# 2. Data Cleaning

## 2.1 Handle missing values, duplicates, and any inconsistencies in the data.

In [303…

```
for x in cat_col:
    print(f"{x} has {df[x].nunique()} unique values:")
    # print(f"{df[x].nunique()} unique values ")
    print(df[x].unique())
    print("-------------")
```

```
Ship Mode has 4 unique values:
['Second Class' 'Standard Class' 'First Class' 'Same Day']
-------------
Segment has 3 unique values:
['Consumer' 'Corporate' 'Home Office']
-------------
Country has 1 unique values:
['United States']
-------------
City has 531 unique values:
['Henderson' 'Los Angeles' 'Fort Lauderdale' 'Concord' 'Seattle'
 'Fort Worth' 'Madison' 'West Jordan' 'San Francisco' 'Fremont'
 'Philadelphia' 'Orem' 'Houston' 'Richardson' 'Naperville' 'Melbourne'
 'Eagan' 'Westland' 'Dover' 'New Albany' 'New York City' 'Troy' 'Chicago'
 'Gilbert' 'Springfield' 'Jackson' 'Memphis' 'Decatur' 'Durham' 'Columbia'
 'Rochester' 'Minneapolis' 'Portland' 'Saint Paul' 'Aurora' 'Charlotte'
 'Orland Park' 'Urbandale' 'Columbus' 'Bristol' 'Wilmington' 'Bloomington'
 'Phoenix' 'Roseville' 'Independence' 'Pasadena' 'Newark' 'Franklin'
 'Scottsdale' 'San Jose' 'Edmond' 'Carlsbad' 'San Antonio' 'Monroe'
 'Fairfield' 'Grand Prairie' 'Redlands' 'Hamilton' 'Westfield' 'Akron'
 'Denver' 'Dallas' 'Whittier' 'Saginaw' 'Medina' 'Dublin' 'Detroit'
 'Tampa' 'Santa Clara' 'Lakeville' 'San Diego' 'Brentwood' 'Chapel Hill'
 'Morristown' 'Cincinnati' 'Inglewood' 'Tamarac' 'Colorado Springs'
 'Belleville' 'Taylor' 'Lakewood' 'Arlington' 'Arvada' 'Hackensack'
 'Saint Petersburg' 'Long Beach' 'Hesperia' 'Murfreesboro' 'Layton'
 'Austin' 'Lowell' 'Manchester' 'Harlingen' 'Tucson' 'Quincy'
 'Pembroke Pines' 'Des Moines' 'Peoria' 'Las Vegas' 'Warwick' 'Miami'
 'Huntington Beach' 'Richmond' 'Louisville' 'Lawrence' 'Canton'
 'New Rochelle' 'Gastonia' 'Jacksonville' 'Auburn' 'Norman' 'Park Ridge'
 'Amarillo' 'Lindenhurst' 'Huntsville' 'Fayetteville' 'Costa Mesa'
 'Parker' 'Atlanta' 'Gladstone' 'Great Falls' 'Lakeland' 'Montgomery'
 'Mesa' 'Green Bay' 'Anaheim' 'Marysville' 'Salem' 'Laredo' 'Grove City'
 'Dearborn' 'Warner Robins' 'Vallejo' 'Mission Viejo' 'Rochester Hills'
 'Plainfield' 'Sierra Vista' 'Vancouver' 'Cleveland' 'Tyler' 'Burlington'
 'Waynesboro' 'Chester' 'Cary' 'Palm Coast' 'Mount Vernon' 'Hialeah'
 'Oceanside' 'Evanston' 'Trenton' 'Cottage Grove' 'Bossier City'
 'Lancaster' 'Asheville' 'Lake Elsinore' 'Omaha' 'Edmonds' 'Santa Ana'
 'Milwaukee' 'Florence' 'Lorain' 'Linden' 'Salinas' 'New Brunswick'
 'Garland' 'Norwich' 'Alexandria' 'Toledo' 'Farmington' 'Riverside'
 'Torrance' 'Round Rock' 'Boca Raton' 'Virginia Beach' 'Murrieta'
 'Olympia' 'Washington' 'Jefferson City' 'Saint Peters' 'Rockford'
 'Brownsville' 'Yonkers' 'Oakland' 'Clinton' 'Encinitas' 'Roswell'
 'Jonesboro' 'Antioch' 'Homestead' 'La Porte' 'Lansing' 'Cuyahoga Falls'
 'Reno' 'Harrisonburg' 'Escondido' 'Royal Oak' 'Rockville' 'Coral Springs'
 'Buffalo' 'Boynton Beach' 'Gulfport' 'Fresno' 'Greenville' 'Macon'
 'Cedar Rapids' 'Providence' 'Pueblo' 'Deltona' 'Murray' 'Middletown'
 'Freeport' 'Pico Rivera' 'Provo' 'Pleasant Grove' 'Smyrna' 'Parma'
 'Mobile' 'New Bedford' 'Irving' 'Vineland' 'Glendale' 'Niagara Falls'
 'Thomasville' 'Westminster' 'Coppell' 'Pomona' 'North Las Vegas'
 'Allentown' 'Tempe' 'Laguna Niguel' 'Bridgeton' 'Everett' 'Watertown'
 'Appleton' 'Bellevue' 'Allen' 'El Paso' 'Grapevine' 'Carrollton' 'Kent'
 'Lafayette' 'Tigard' 'Skokie' 'Plano' 'Suffolk' 'Indianapolis' 'Bayonne'
 'Greensboro' 'Baltimore' 'Kenosha' 'Olathe' 'Tulsa' 'Redmond' 'Raleigh'
 'Muskogee' 'Meriden' 'Bowling Green' 'South Bend' 'Spokane' 'Keller'
 'Port Orange' 'Medford' 'Charlottesville' 'Missoula' 'Apopka' 'Reading'
 'Broomfield' 'Paterson' 'Oklahoma City' 'Chesapeake' 'Lubbock'
 'Johnson City' 'San Bernardino' 'Leominster' 'Bozeman' 'Perth Amboy'
 'Ontario' 'Rancho Cucamonga' 'Moorhead' 'Mesquite' 'Stockton'
 'Ormond Beach' 'Sunnyvale' 'York' 'College Station' 'Saint Louis'
 'Manteca' 'San Angelo' 'Salt Lake City' 'Knoxville' 'Little Rock'
```

```
 'Lincoln Park' 'Marion' 'Littleton' 'Bangor' 'Southaven' 'New Castle'
 'Midland' 'Sioux Falls' 'Fort Collins' 'Clarksville' 'Sacramento'
 'Thousand Oaks' 'Malden' 'Holyoke' 'Albuquerque' 'Sparks' 'Coachella'
 'Elmhurst' 'Passaic' 'North Charleston' 'Newport News' 'Jamestown'
 'Mishawaka' 'La Quinta' 'Tallahassee' 'Nashville' 'Bellingham'
 'Woodstock' 'Haltom City' 'Wheeling' 'Summerville' 'Hot Springs'
 'Englewood' 'Las Cruces' 'Hoover' 'Frisco' 'Vacaville' 'Waukesha'
 'Bakersfield' 'Pompano Beach' 'Corpus Christi' 'Redondo Beach' 'Orlando'
 'Orange' 'Lake Charles' 'Highland Park' 'Hempstead' 'Noblesville'
 'Apple Valley' 'Mount Pleasant' 'Sterling Heights' 'Eau Claire' 'Pharr'
 'Billings' 'Gresham' 'Chattanooga' 'Meridian' 'Bolingbrook' 'Maple Grove'
 'Woodland' 'Missouri City' 'Pearland' 'San Mateo' 'Grand Rapids'
 'Visalia' 'Overland Park' 'Temecula' 'Yucaipa' 'Revere' 'Conroe'
 'Tinley Park' 'Dubuque' 'Dearborn Heights' 'Santa Fe' 'Hickory'
 'Carol Stream' 'Saint Cloud' 'North Miami' 'Plantation'
 'Port Saint Lucie' 'Rock Hill' 'Odessa' 'West Allis' 'Chula Vista'
 'Manhattan' 'Altoona' 'Thornton' 'Champaign' 'Texarkana' 'Edinburg'
 'Baytown' 'Greenwood' 'Woonsocket' 'Superior' 'Bedford' 'Covington'
 'Broken Arrow' 'Miramar' 'Hollywood' 'Deer Park' 'Wichita' 'Mcallen'
 'Iowa City' 'Boise' 'Cranston' 'Port Arthur' 'Citrus Heights'
 'The Colony' 'Daytona Beach' 'Bullhead City' 'Portage' 'Fargo' 'Elkhart'
 'San Gabriel' 'Margate' 'Sandy Springs' 'Mentor' 'Lawton' 'Hampton'
 'Rome' 'La Crosse' 'Lewiston' 'Hattiesburg' 'Danville' 'Logan'
 'Waterbury' 'Athens' 'Avondale' 'Marietta' 'Yuma' 'Wausau' 'Pasco'
 'Oak Park' 'Pensacola' 'League City' 'Gaithersburg' 'Lehi' 'Tuscaloosa'
 'Moreno Valley' 'Georgetown' 'Loveland' 'Chandler' 'Helena' 'Kirkwood'
 'Waco' 'Frankfort' 'Bethlehem' 'Grand Island' 'Woodbury' 'Rogers'
 'Clovis' 'Jupiter' 'Santa Barbara' 'Cedar Hill' 'Norfolk' 'Draper'
 'Ann Arbor' 'La Mesa' 'Pocatello' 'Holland' 'Milford' 'Buffalo Grove'
 'Lake Forest' 'Redding' 'Chico' 'Utica' 'Conway' 'Cheyenne' 'Owensboro'
 'Caldwell' 'Kenner' 'Nashua' 'Bartlett' 'Redwood City' 'Lebanon'
 'Santa Maria' 'Des Plaines' 'Longview' 'Hendersonville' 'Waterloo'
 'Cambridge' 'Palatine' 'Beverly' 'Eugene' 'Oxnard' 'Renton' 'Glenview'
 'Delray Beach' 'Commerce City' 'Texas City' 'Wilson' 'Rio Rancho'
 'Goldsboro' 'Montebello' 'El Cajon' 'Beaumont' 'West Palm Beach'
 'Abilene' 'Normal' 'Saint Charles' 'Camarillo' 'Hillsboro' 'Burbank'
 'Modesto' 'Garden City' 'Atlantic City' 'Longmont' 'Davis' 'Morgan Hill'
 'Clifton' 'Sheboygan' 'East Point' 'Rapid City' 'Andover' 'Kissimmee'
 'Shelton' 'Danbury' 'Sanford' 'San Marcos' 'Greeley' 'Mansfield' 'Elyria'
 'Twin Falls' 'Coral Gables' 'Romeoville' 'Marlborough' 'Laurel' 'Bryan'
 'Pine Bluff' 'Aberdeen' 'Hagerstown' 'East Orange' 'Arlington Heights'
 'Oswego' 'Coon Rapids' 'San Clemente' 'San Luis Obispo' 'Springdale'
 'Lodi' 'Mason']
-------------
State has 49 unique values:
['Kentucky' 'California' 'Florida' 'North Carolina' 'Washington' 'Texas'
 'Wisconsin' 'Utah' 'Nebraska' 'Pennsylvania' 'Illinois' 'Minnesota'
 'Michigan' 'Delaware' 'Indiana' 'New York' 'Arizona' 'Virginia'
 'Tennessee' 'Alabama' 'South Carolina' 'Oregon' 'Colorado' 'Iowa' 'Ohio'
 'Missouri' 'Oklahoma' 'New Mexico' 'Louisiana' 'Connecticut' 'New Jersey'
 'Massachusetts' 'Georgia' 'Nevada' 'Rhode Island' 'Mississippi'
 'Arkansas' 'Montana' 'New Hampshire' 'Maryland' 'District of Columbia'
 'Kansas' 'Vermont' 'Maine' 'South Dakota' 'Idaho' 'North Dakota'
 'Wyoming' 'West Virginia']
-------------
Region has 4 unique values:
['South' 'West' 'Central' 'East']
-------------
Category has 3 unique values:
['Furniture' 'Office Supplies' 'Technology']
```

```
-------------
Sub-Category has 17 unique values:
['Bookcases' 'Chairs' 'Labels' 'Tables' 'Storage' 'Furnishings' 'Art'
 'Phones' 'Binders' 'Appliances' 'Paper' 'Accessories' 'Envelopes'
 'Fasteners' 'Supplies' 'Machines' 'Copiers']
-------------
```

## 2.2 Converting data types.

In [304…
```
# Convert Postal Code column to string b/z it is not used for calculation.
df['Postal Code'] = df['Postal Code'].astype(str)
```

# 3. Descriptive Statistics

## 3.1 Calculate basic descriptive statistics.

In [305…
```
total_sales = df['Sales'].sum()
total_sales
```

Out[305… 2297200.8603000003

In [306…
```
total_quantity_sold = df['Quantity'].sum()
total_quantity_sold
```

Out[306… 37873

In [307…
```
total_profit = df['Profit'].sum()
total_profit
```

Out[307… 286397.0217

In [308…
```
average_order_value = df['Sales'].mean()
average_order_value
```

Out[308… 229.85800083049833

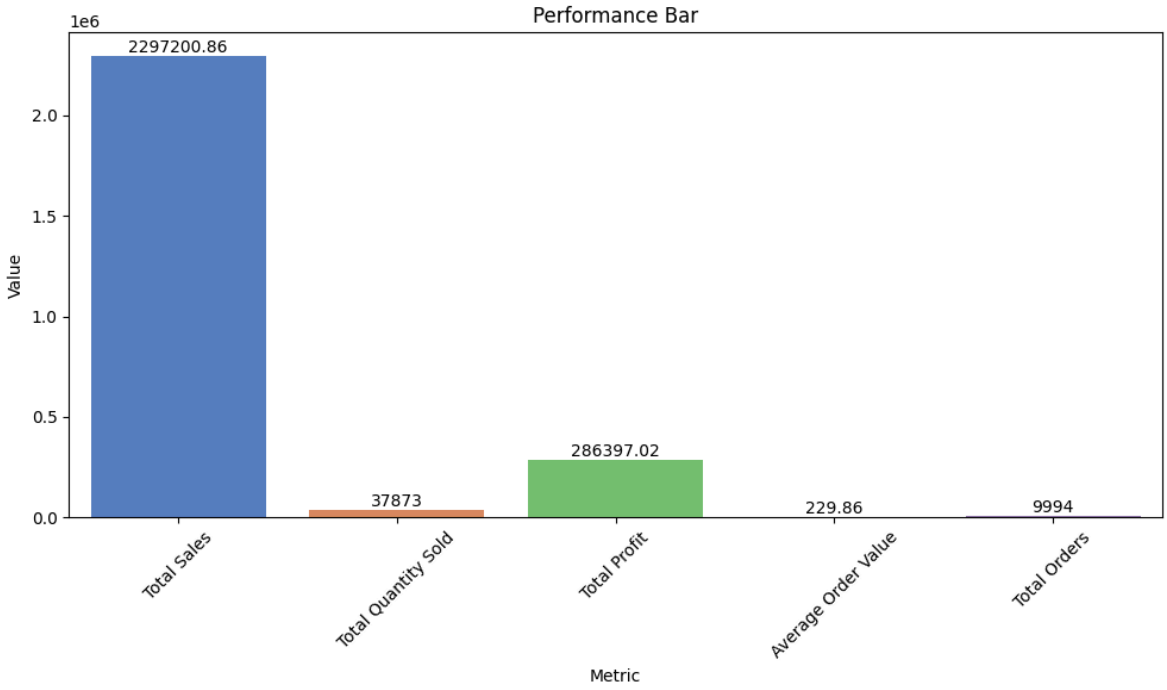
In [309…
```
total_orders = len(df)
total_orders
```

Out[309… 9994

In [310…
```
# Define the metrics and their corresponding values
metrics = ['Total Sales', 'Total Quantity Sold', 'Total Profit', 'Average Order
values = [total_sales, total_quantity_sold, total_profit, average_order_value, t

# Create the bar plot using Seaborn
plt.figure(figsize=(10, 6))
sns.barplot(x=metrics, y=values, palette='muted')

# Annotate each bar with its corresponding value
for i in range(len(metrics)):
    plt.text(i, values[i], str(round(values[i], 2)), ha='center', va='bottom')

plt.xlabel('Metric')
```

```
plt.ylabel('Value')
plt.title('Performance Bar')
plt.xticks(rotation=45)  # Rotate x-axis labels for better readability
plt.tight_layout()  # Adjust layout to prevent clipping of labels
plt.show()
```

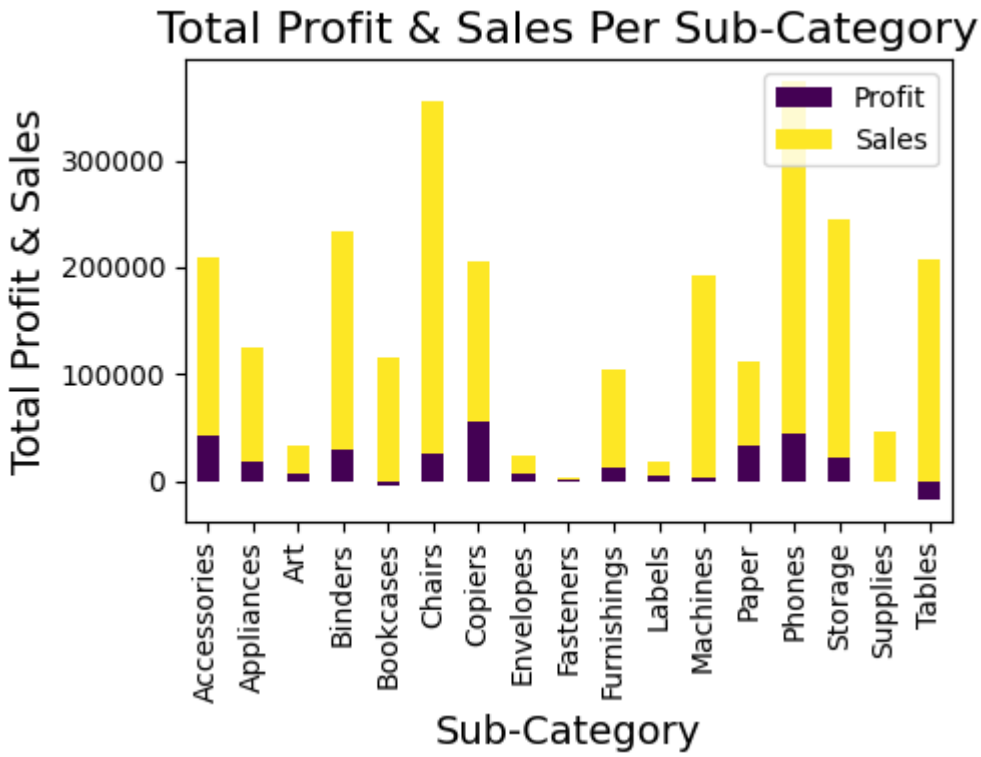
```
C:\Users\user\AppData\Local\Temp\ipykernel_16144\1734670603.py:7: FutureWarning:


Passing `palette` without assigning `hue` is deprecated and will be removed in v
0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effe
ct.
```




## 3.2 Visualize the distribution of sales, order quantity, and other relevant metrics.

In [311…

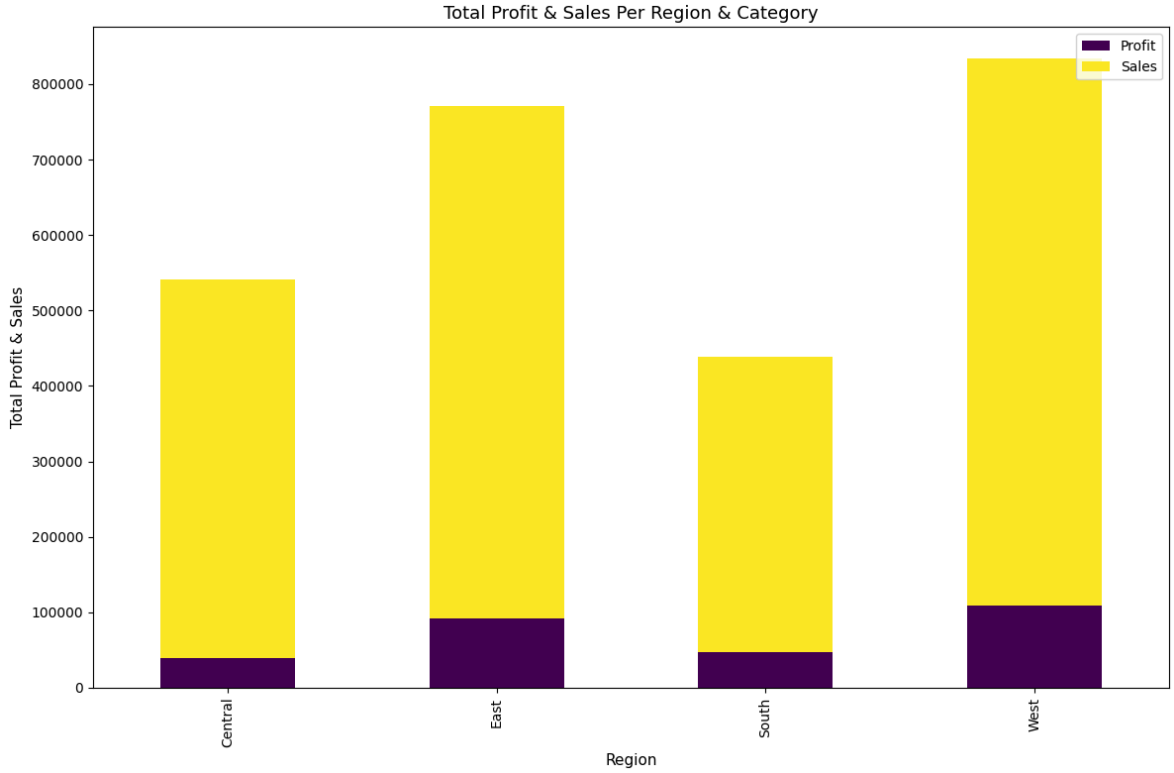
```
grouped_data = df.groupby(['Sub-Category'])[['Profit', 'Sales']].agg('sum')
ax = grouped_data.plot(kind='bar', stacked=True, colormap='viridis')

plt.title('Total Profit & Sales Per Sub-Category', fontsize=16)
plt.xlabel('Sub-Category', fontsize=14)
plt.ylabel('Total Profit & Sales', fontsize=14)

plt.xticks(rotation=90)
plt.legend(['Profit', 'Sales'], loc='upper right')
plt.rcParams['figure.figsize'] = [12, 8]

plt.tight_layout()
plt.show()
```




- Copiers" Sub-category has gain highest amount of profit with no loss.There are other sub-categories too who are not faced any kind of losses but their profit margins are also low.

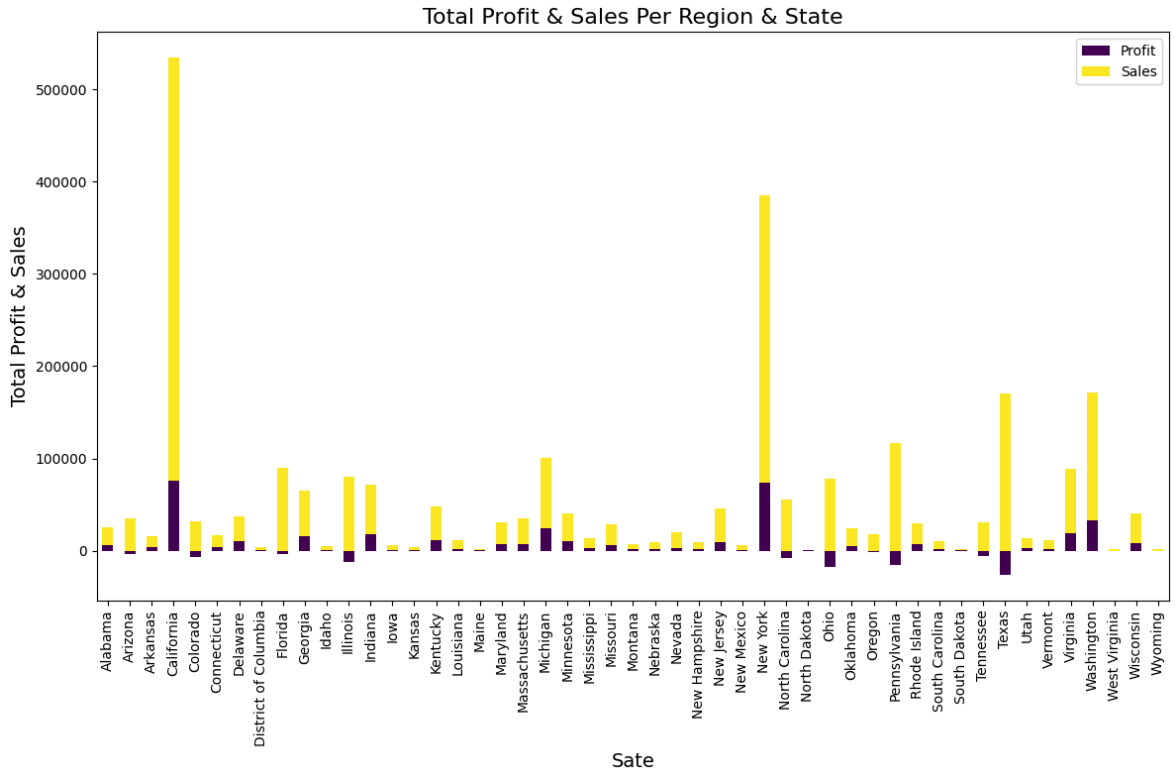
In [312…

```
grouped_data = df.groupby(['Region'])[['Profit', 'Sales']].agg('sum')
ax = grouped_data.plot(kind='bar', stacked=True, colormap='viridis')

plt.title('Total Profit & Sales Per Region & Category', fontsize=13)
plt.xlabel('Region', fontsize=11)
plt.ylabel('Total Profit & Sales', fontsize=11)

plt.xticks(rotation=90)
plt.legend(['Profit', 'Sales'], loc='upper right')
plt.rcParams['figure.figsize'] = [6,4]

plt.tight_layout()
plt.show()
```



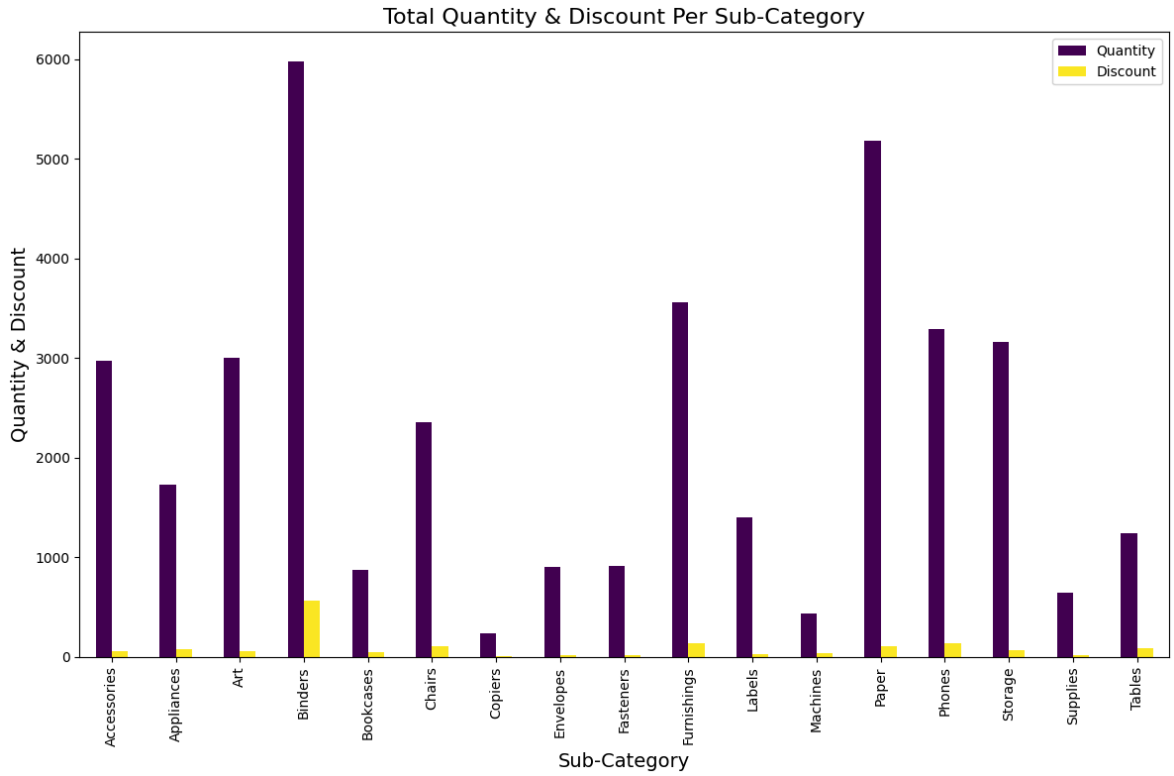

In [314…

```
grouped_data = df.groupby(['State'])[['Profit', 'Sales']].agg('sum')
ax = grouped_data.plot(kind='bar', stacked=True, colormap='viridis')

plt.title('Total Profit & Sales Per Region & State', fontsize=16)
plt.xlabel('Sate', fontsize=14)
plt.ylabel('Total Profit & Sales', fontsize=14)

plt.xticks(rotation=90)
plt.legend(['Profit', 'Sales'], loc='upper right')
plt.rcParams['figure.figsize'] = [12, 8]

plt.tight_layout()
plt.show()
```



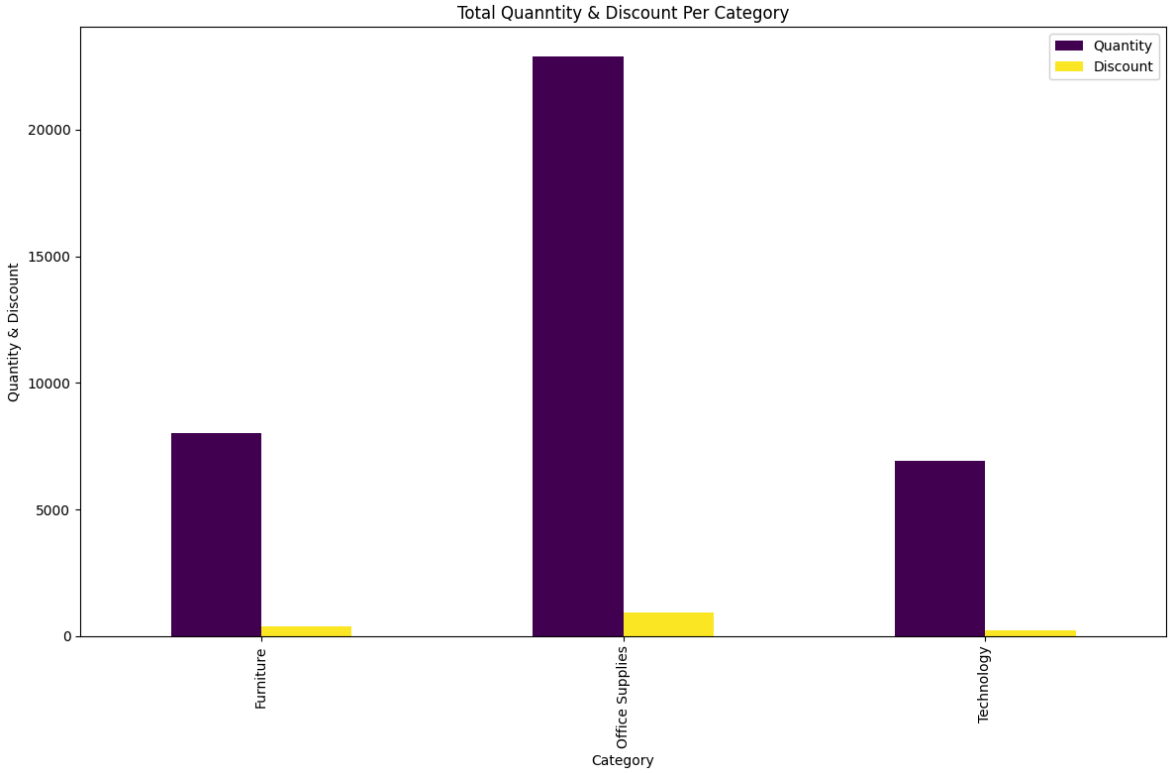

In [315…

```
grouped_data = df.groupby(['Sub-Category'])[['Quantity', 'Discount']].agg('sum')
ax = grouped_data.plot(kind='bar', stacked=False, colormap='viridis')

plt.title('Total Quantity & Discount Per Sub-Category', fontsize=16)
plt.xlabel('Sub-Category', fontsize=14)
plt.ylabel('Quantity & Discount', fontsize=14)

plt.xticks(rotation=90)
plt.legend(['Quantity', 'Discount'], loc='upper right')
plt.rcParams['figure.figsize'] = [12, 8]

plt.tight_layout()
plt.show()
```



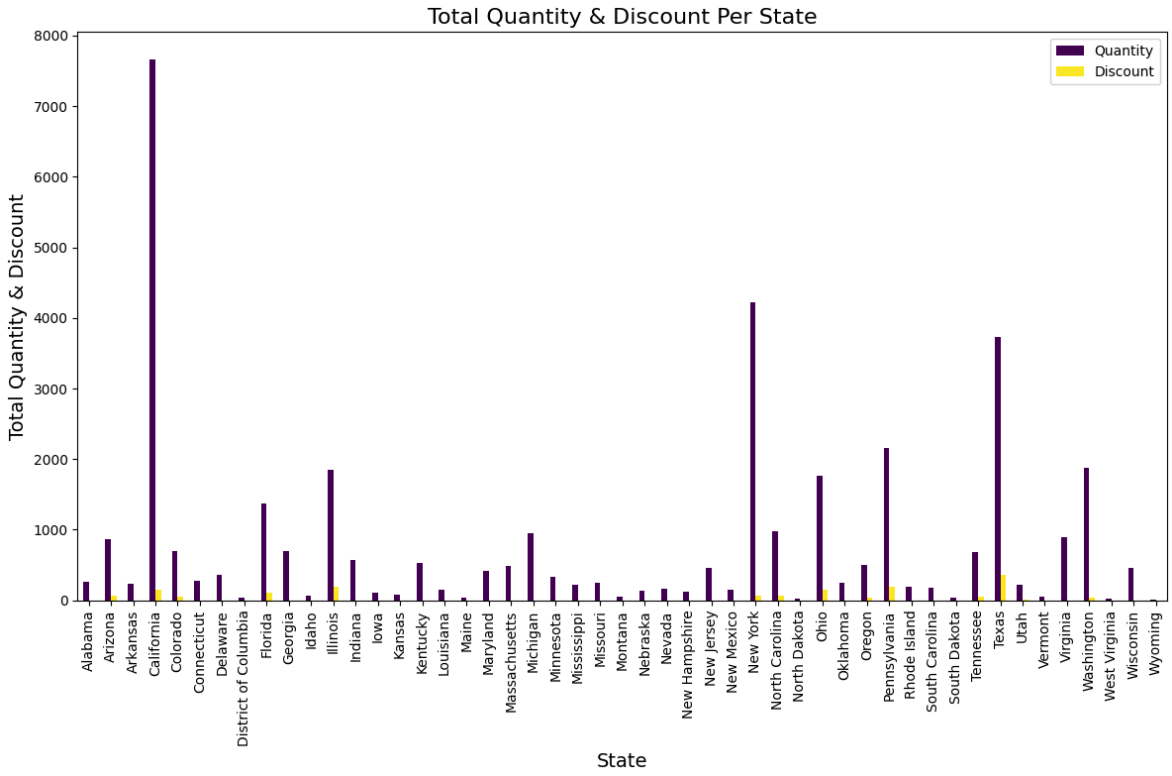

In [316…

```
grouped_data = df.groupby(['Category'])[['Quantity', 'Discount']].agg('sum')
ax = grouped_data.plot(kind='bar', stacked=False, colormap='viridis')

plt.title('Total Quanntity & Discount Per Category', fontsize=12)
plt.xlabel('Category', fontsize=10)
plt.ylabel('Quantity & Discount', fontsize=10)

plt.xticks(rotation=90)
plt.legend(['Quantity', 'Discount'], loc='upper right')
plt.rcParams['figure.figsize'] = [6,4]

plt.tight_layout()
plt.show()
```




In [318…

```
grouped_data = df.groupby(['State'])[['Quantity', 'Discount']].agg('sum')
ax = grouped_data.plot(kind='bar', stacked=False, colormap='viridis')

plt.title('Total Quantity & Discount Per State', fontsize=16)
plt.xlabel('State', fontsize=14)
plt.ylabel('Total Quantity & Discount', fontsize=14)

plt.xticks(rotation=90)
plt.legend(['Quantity', 'Discount'], loc='upper right')
plt.rcParams['figure.figsize'] = [12, 8]

plt.tight_layout()
plt.show()
```




# 4. Customer Segmentation.

## 4.1 Segment customers based on their purchasing behavior.

In [319…

```
# Monetary segmentation
monetary_segment = pd.qcut(df['Sales'], q=3, labels=['Low-purchase', 'Medium-pur

# Adding segmentation columns to the DataFrame
df['Customer_Segment'] = monetary_segment

# Displaying segmentation results
print("\nCustomer_Segment:")
print(df['Customer_Segment'].value_counts())
```

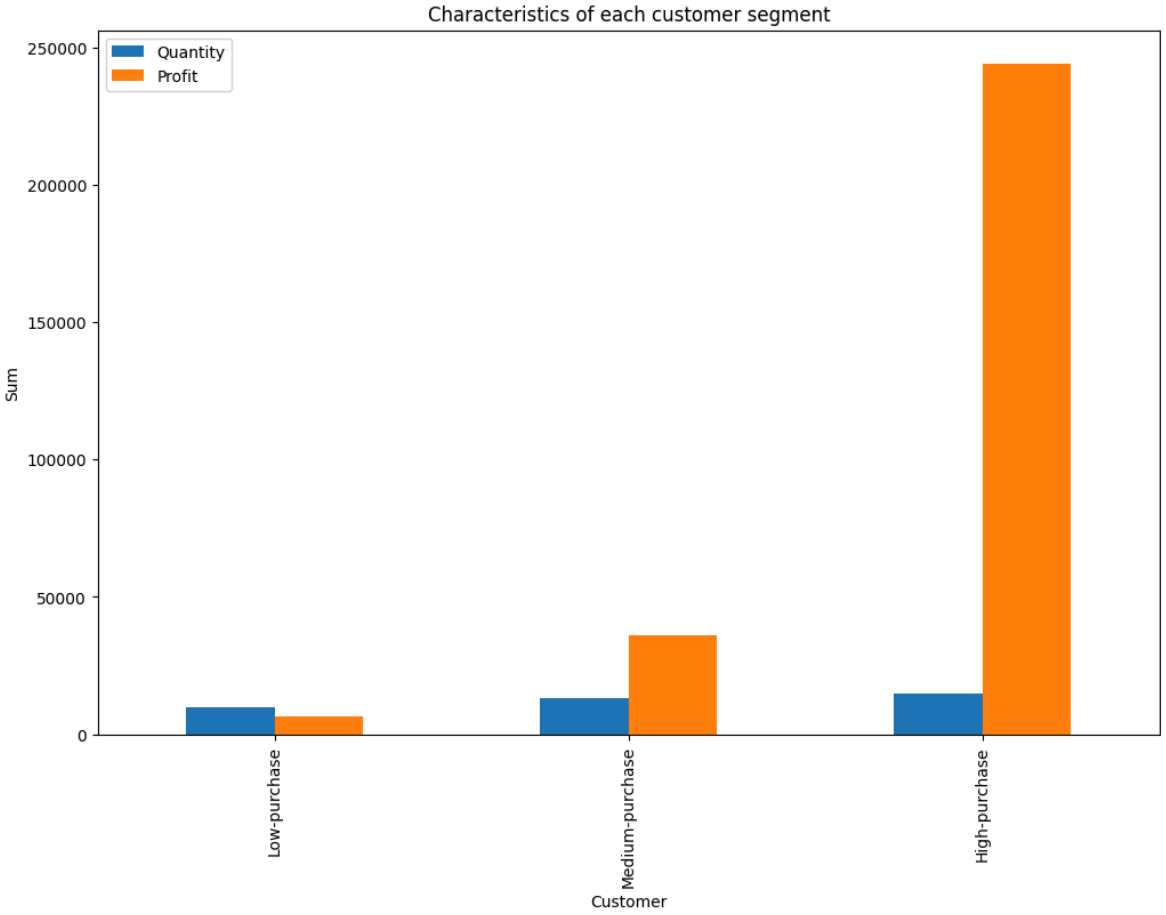
```
Customer_Segment:
Customer_Segment
Medium-purchase    3333
Low-purchase       3332
High-purchase      3329
Name: count, dtype: int64
```

## 4.2 Analyze the characteristics of each customer segment.

In [320…

```
df.groupby('Customer_Segment')[['Quantity', 'Profit']].sum().plot.bar()
plt.title('Characteristics of each customer segment')
plt.xlabel('Customer')
plt.ylabel("Sum")
plt.xticks(rotation=90)
plt.rcParams['figure.figsize'] = [5,4]
plt.show()
```

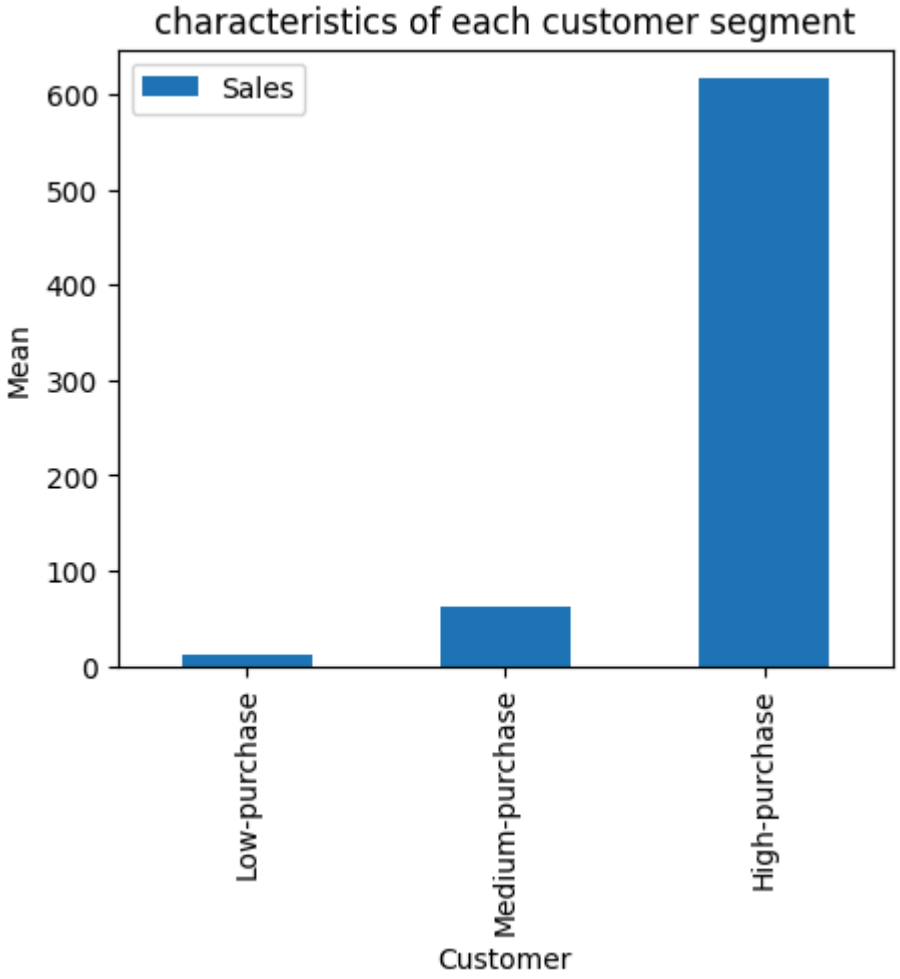
```
C:\Users\user\AppData\Local\Temp\ipykernel_16144\268246056.py:1: FutureWarning:

The default of observed=False is deprecated and will be changed to True in a futu
re version of pandas. Pass observed=False to retain current behavior or observed=
True to adopt the future default and silence this warning.
```




In [321…

```
df.groupby('Customer_Segment')[["Sales"]].mean().plot.bar()
plt.title('characteristics of each customer segment')
plt.xlabel('Customer')
plt.ylabel("Mean")
plt.xticks(rotation=90)
plt.rcParams['figure.figsize'] = [5,4]
plt.show()
```

```
C:\Users\user\AppData\Local\Temp\ipykernel_16144\2570836708.py:1: FutureWarning:

The default of observed=False is deprecated and will be changed to True in a futu
re version of pandas. Pass observed=False to retain current behavior or observed=
True to adopt the future default and silence this warning.
```




# 5. Product Analysis:

## 5.1 Identify the top-selling products and categories.

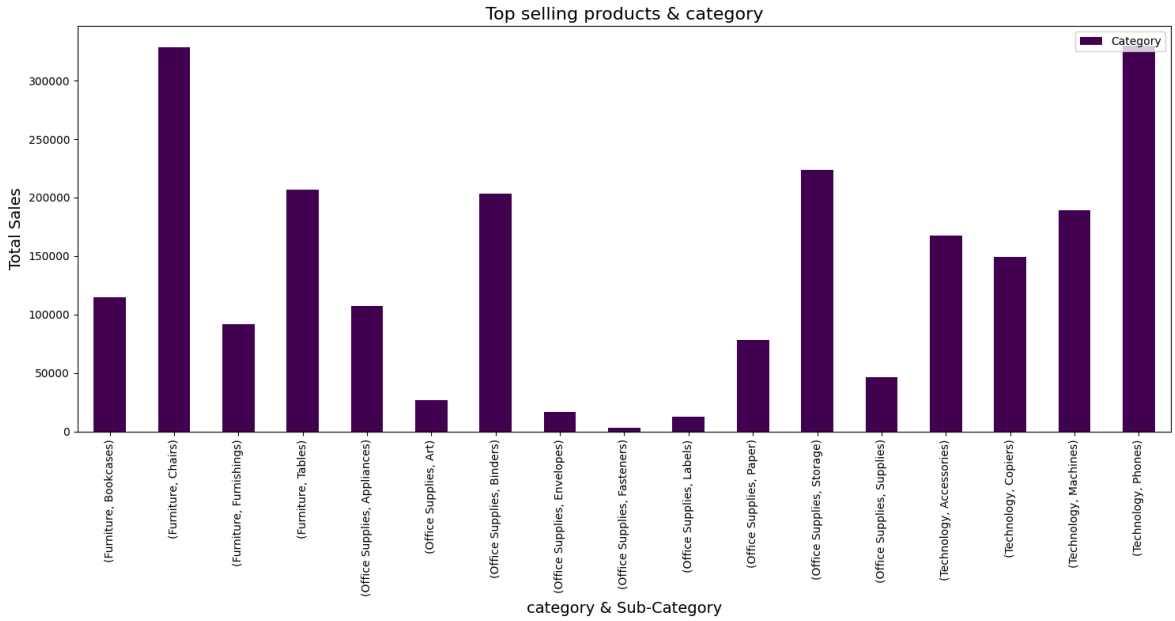
In [325…
```
top_selling = df.groupby(['Category', 'Sub-Category'])['Sales'].sum()
```

In [326…
```
print("Top Selling Products and Categories:")
print(top_selling.head())
```

```
Top Selling Products and Categories:
Category         Sub-Category
Furniture        Bookcases       114879.9963
                 Chairs          328449.1030
                 Furnishings      91705.1640
                 Tables          206965.5320
Office Supplies  Appliances      107532.1610
Name: Sales, dtype: float64
```

In [328…
```
grouped_data = df.groupby(['Category','Sub-Category'])['Sales'].sum()
ax = grouped_data.plot(kind='bar', stacked=False, colormap='viridis')

plt.title('Top selling products & category', fontsize=16)
plt.xlabel('category & Sub-Category', fontsize=14)
plt.ylabel('Total Sales', fontsize=14)

plt.xticks(rotation=90)
```

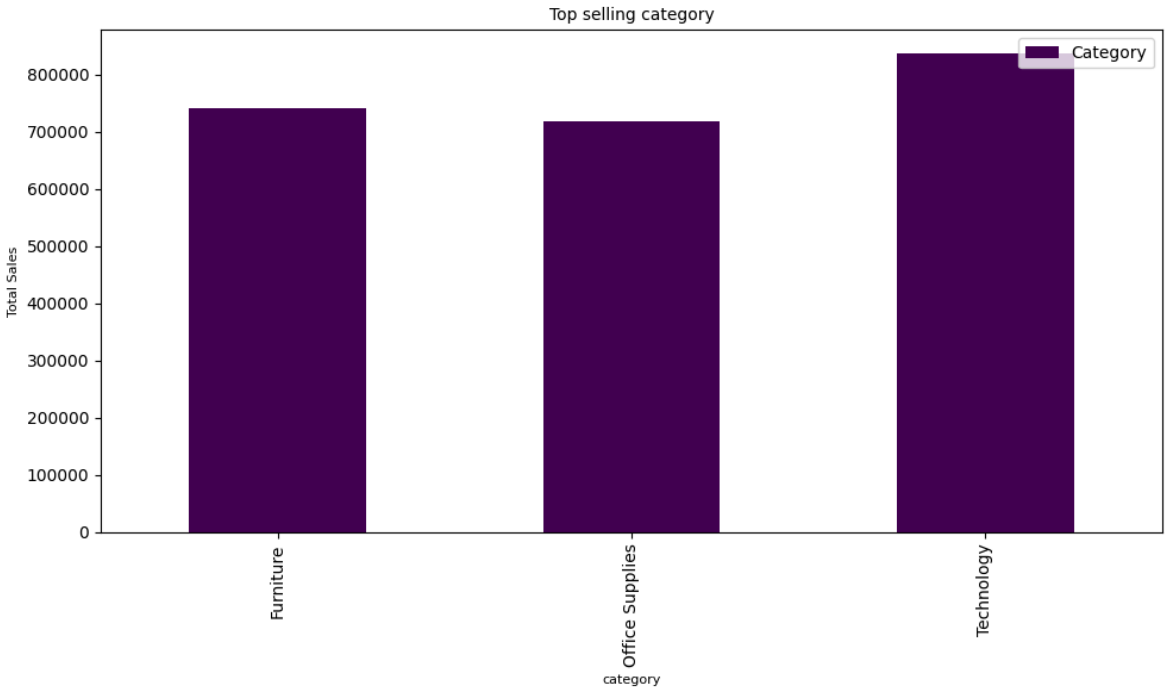
```
plt.legend(['Category', 'Sub-Category'], loc='upper right')
plt.rcParams['figure.figsize'] = [15, 8]

plt.tight_layout()
plt.show()
```




- Here, top selling products are 'Furniture'->'Chairs' & 'Technology'->'Phones' which are equal. min selling products are 'Fasteners'.

In [330…

```
grouped_data = df.groupby(['Category'])['Sales'].sum()
ax = grouped_data.plot(kind='bar', stacked=False, colormap='viridis')

plt.title('Top selling category', fontsize=10)
plt.xlabel('category', fontsize=8)
plt.ylabel('Total Sales', fontsize=8)

plt.xticks(rotation=90)
plt.legend(['Category'], loc='upper right')
plt.rcParams['figure.figsize'] = [10, 6]

plt.tight_layout()
plt.show()
```




- Top selling category is Technology

In [331…
```
df['Segment'].value_counts()
```

Out[331…
```
Segment
Consumer       5191
Corporate      3020
Home Office    1783
Name: count, dtype: int64
```

### 5.2 Analyze the performance of products over time.

Since the dataset you provided doesn't contain a time-related column, such as 'Order Date', we won't be able to analyze trends over time. However, we can still analyze the performance of products based on other variables present in the dataset.

# 6.Time Series Analysis.

### 6.1 Examine sales trends over different time periods.

Since the dataset doesn't contain a time-related column, such as 'Order Date', we can't directly examine sales trends over different time periods like daily, monthly, or yearly.

### 6.2 Identify any seasonality or patterns in the sales data.

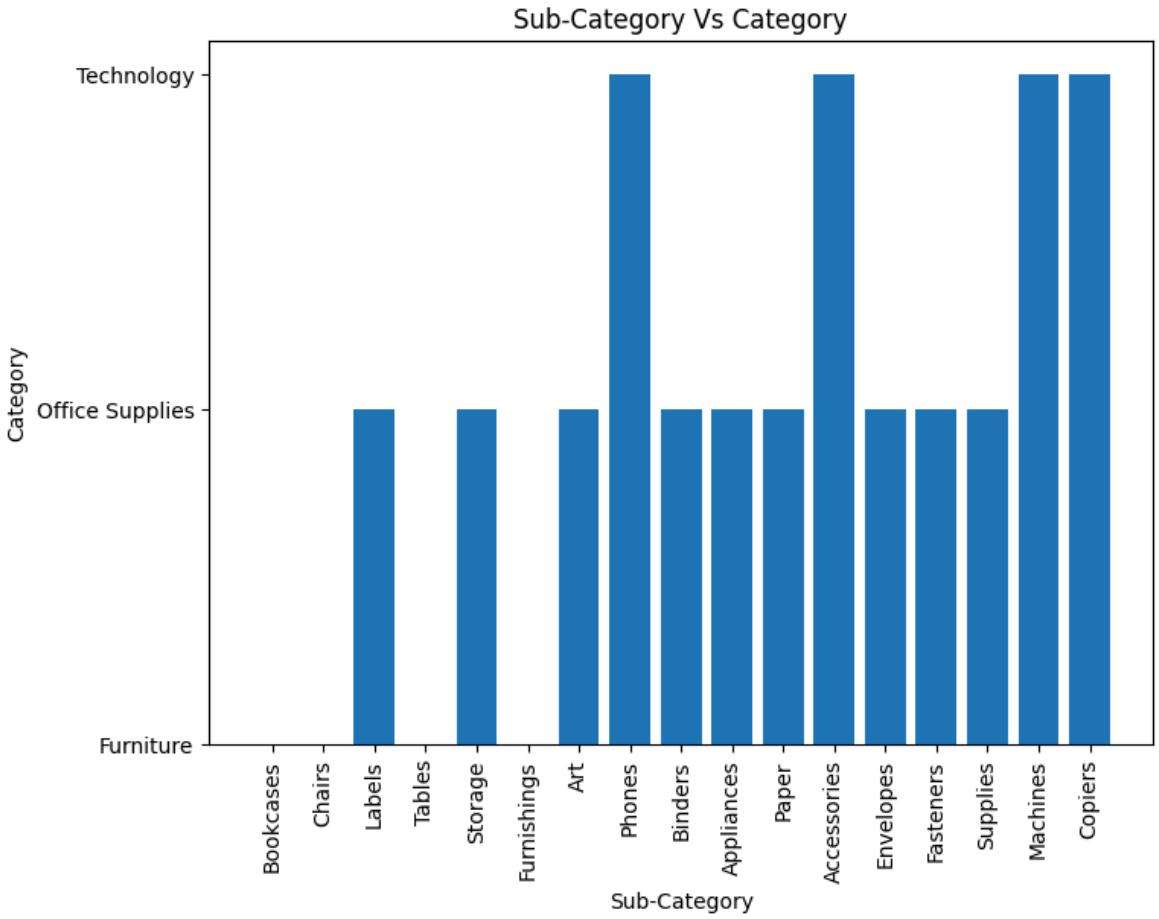
Without a time-related column in the dataset, it's not possible to directly identify seasonality or patterns in the sales data

# 7. Visualizations.

## 7.1 Create visualizations (charts, graphs, dashboards) to present key findings. .

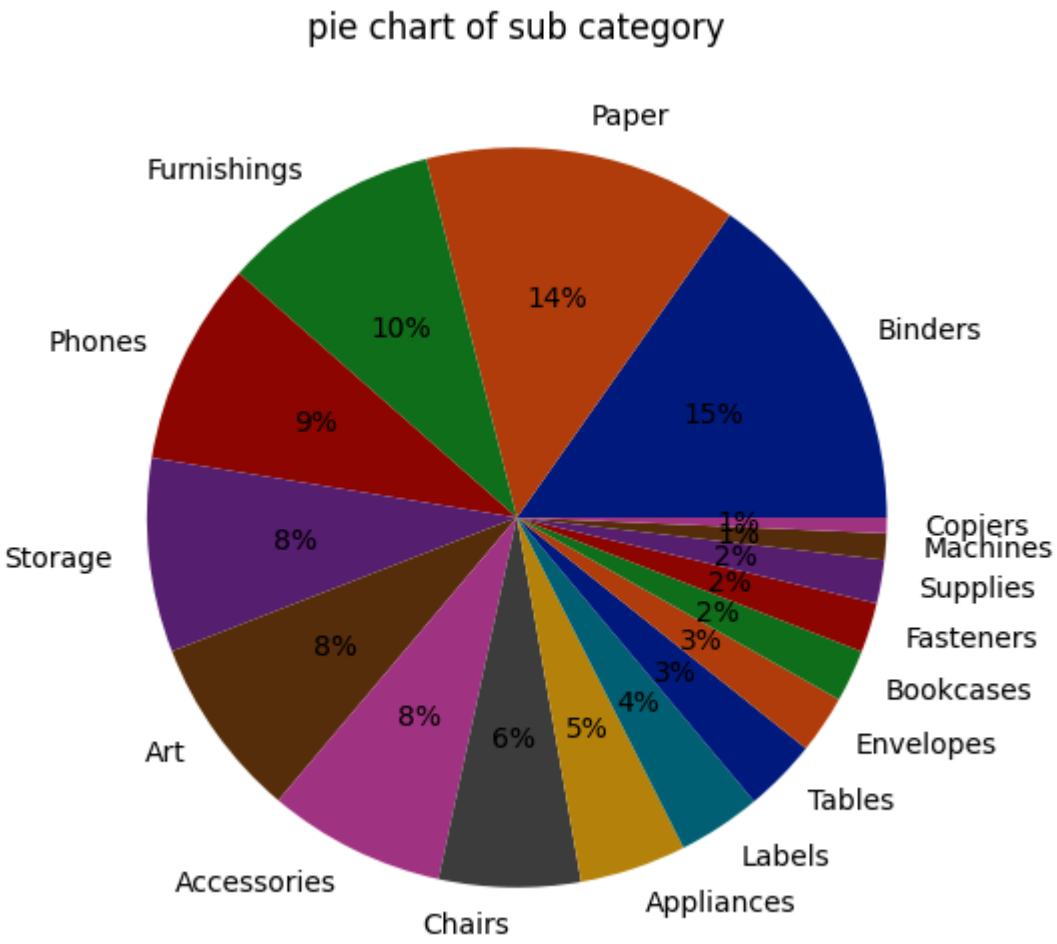
In [332…

```
# Sub-Category Vs Category

plt.figure(figsize=(8,6))
plt.bar('Sub-Category','Category', data=df)
plt.xlabel("Sub-Category")
plt.ylabel("Category")
plt.title("Sub-Category Vs Category")
plt.xticks(rotation=90)
plt.show()
```



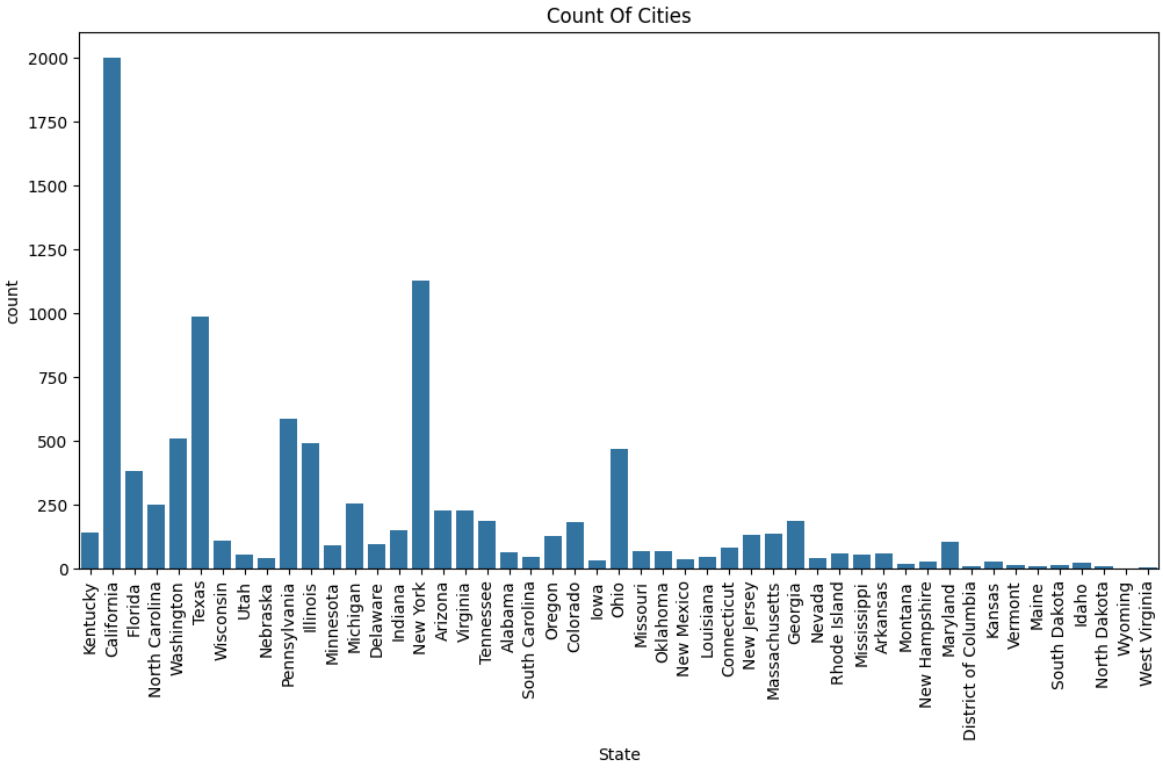

In [333…

```
# pie chart of sub category
palette_color = sns.color_palette('dark')

ship_mode_counts = df['Sub-Category'].value_counts()

plt.pie(ship_mode_counts, labels=ship_mode_counts.index, colors=palette_color, a
plt.title('pie chart of sub category')
plt.show()
```




In [334…

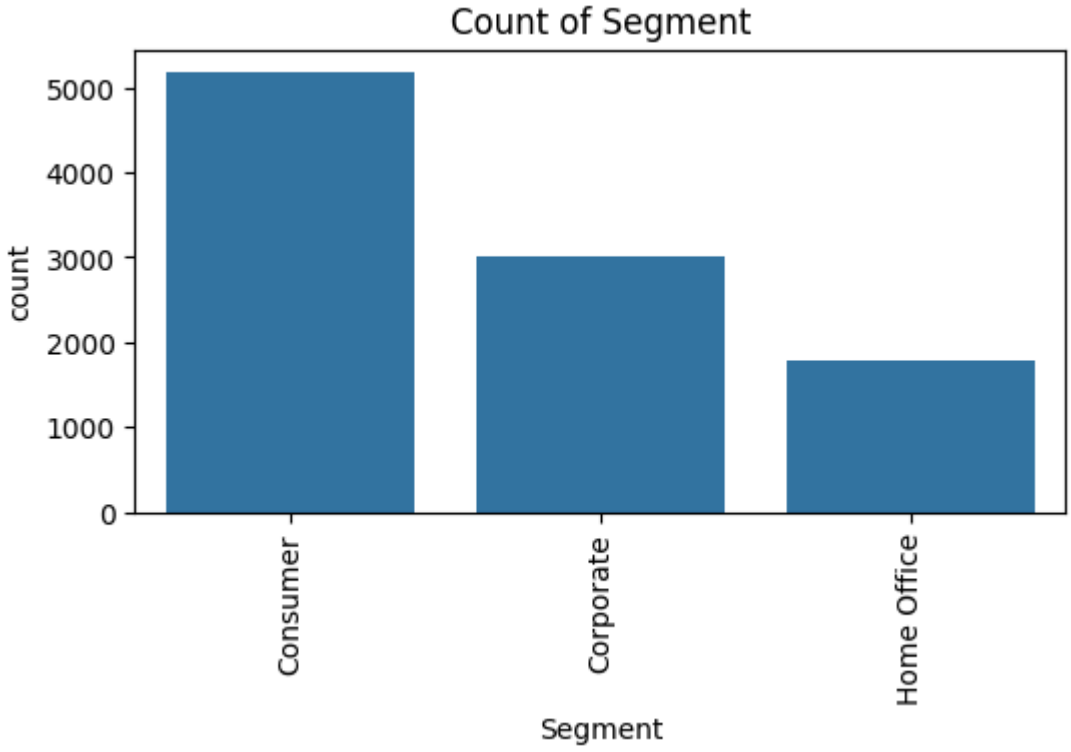
```
# count plot of States
plt.figure(figsize=(12,6))
plt.title('Count Of Cities')
plt.xlabel('State')
sns.countplot(x=df['State'])
plt.xticks(rotation=90)
plt.show()
```




- California in the United States records the highest number of orders placed.

In [335…

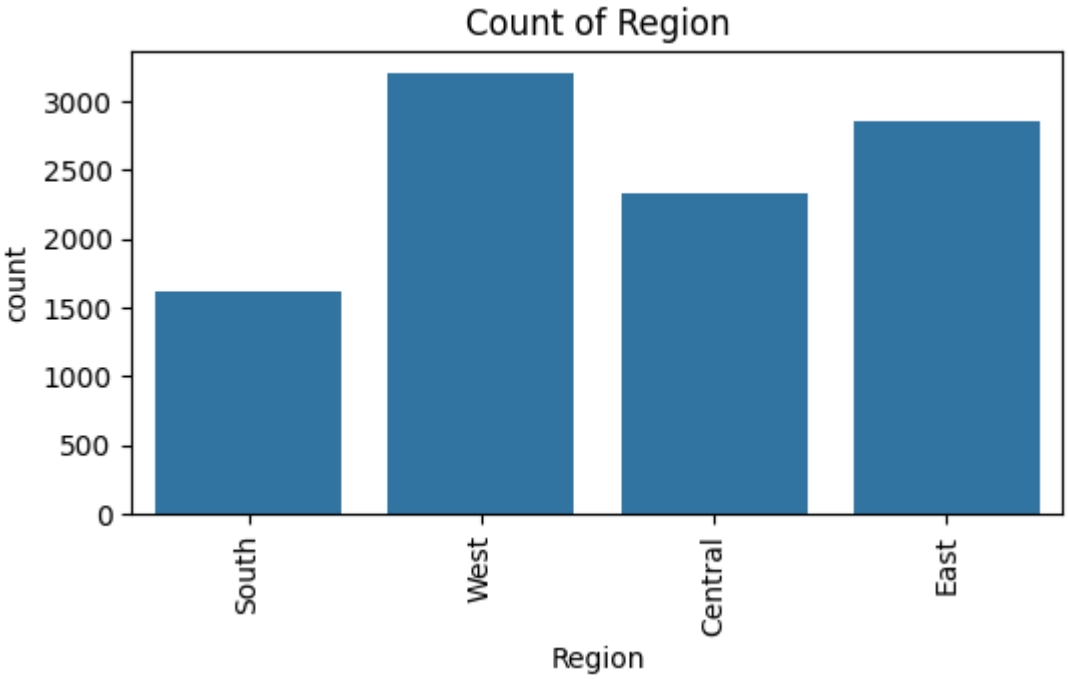
```
# countplot of sub-category
plt.figure(figsize=(6,3))
plt.title("Count of Segment")
sns.countplot(x=df['Segment'])
plt.xticks(rotation=90)
plt.show()
```




In [336…

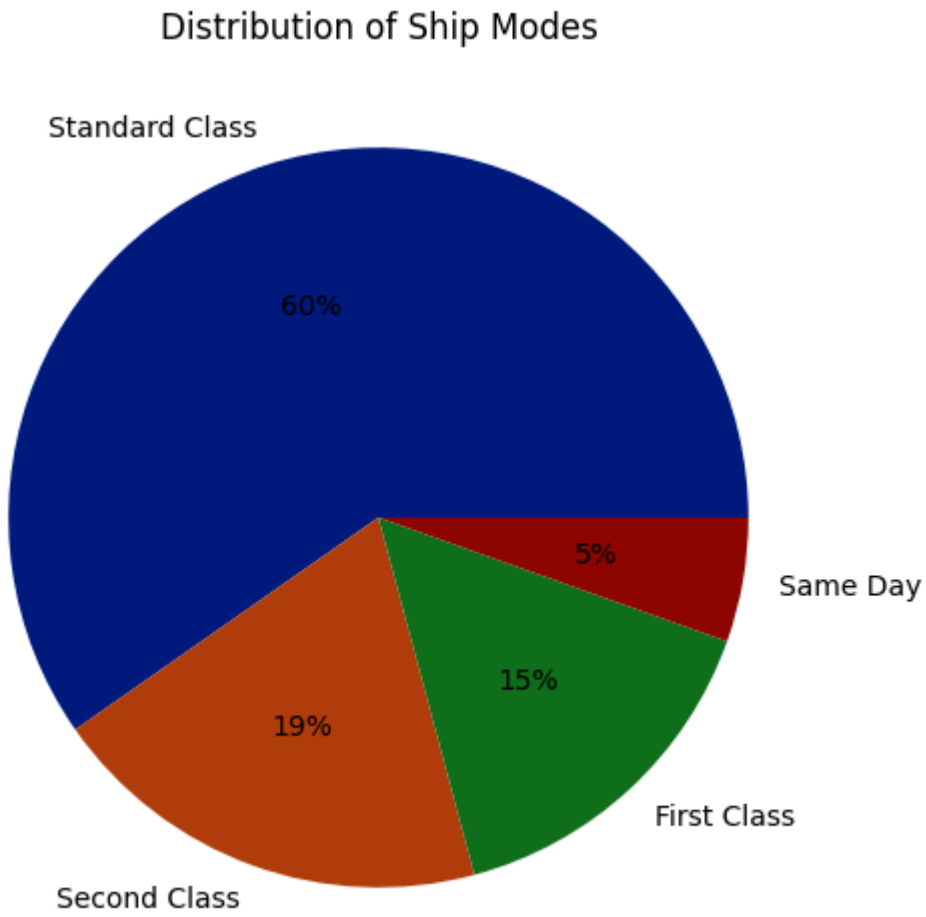
```
# countplot of Region
plt.figure(figsize=(6,3))
```

```
plt.title("Count of Region")
sns.countplot(x=df['Region'])
plt.xticks(rotation=90)
plt.show()
```




In [337…

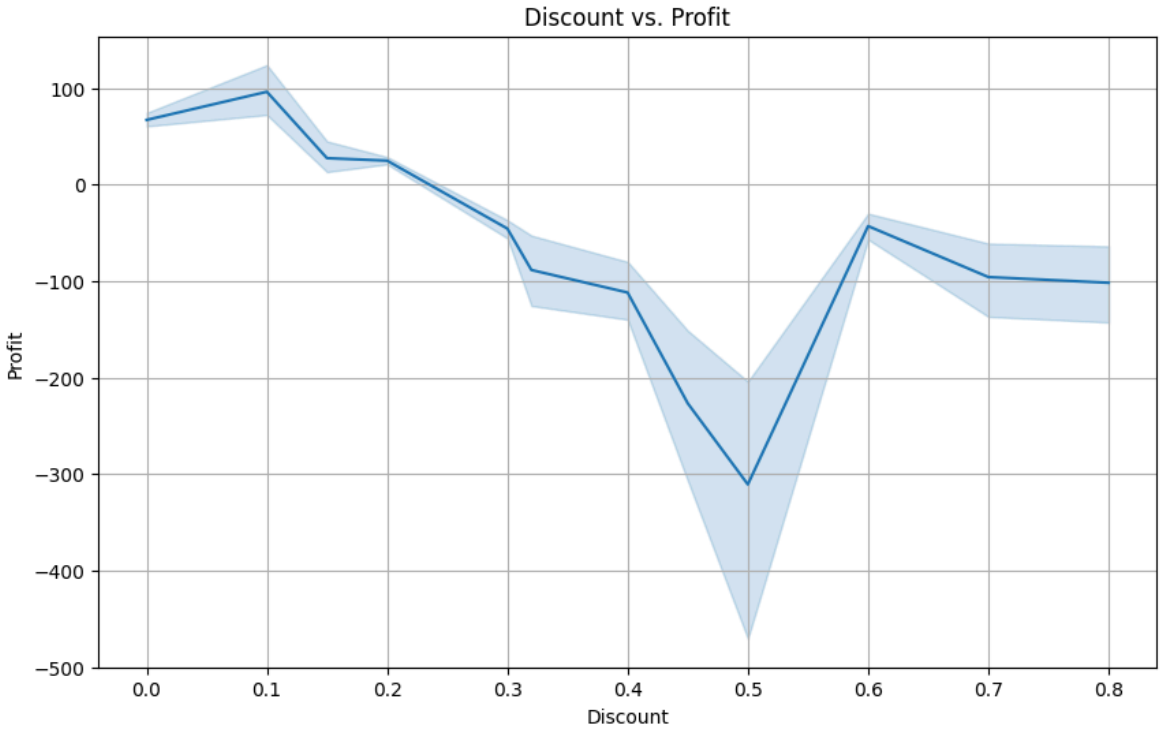
```
palette_color = sns.color_palette('dark')

ship_mode_counts = df['Ship Mode'].value_counts()

plt.pie(ship_mode_counts, labels=ship_mode_counts.index, colors=palette_color, a
plt.title('Distribution of Ship Modes')
plt.show()
```




In [338…

```
# Plotting discount vs. profit
plt.figure(figsize=(10, 6))
sns.lineplot(x='Discount', y='Profit', data=df, palette='viridis')
plt.title('Discount vs. Profit')
plt.xlabel('Discount')
plt.ylabel('Profit')
plt.grid(True)
plt.show()
```

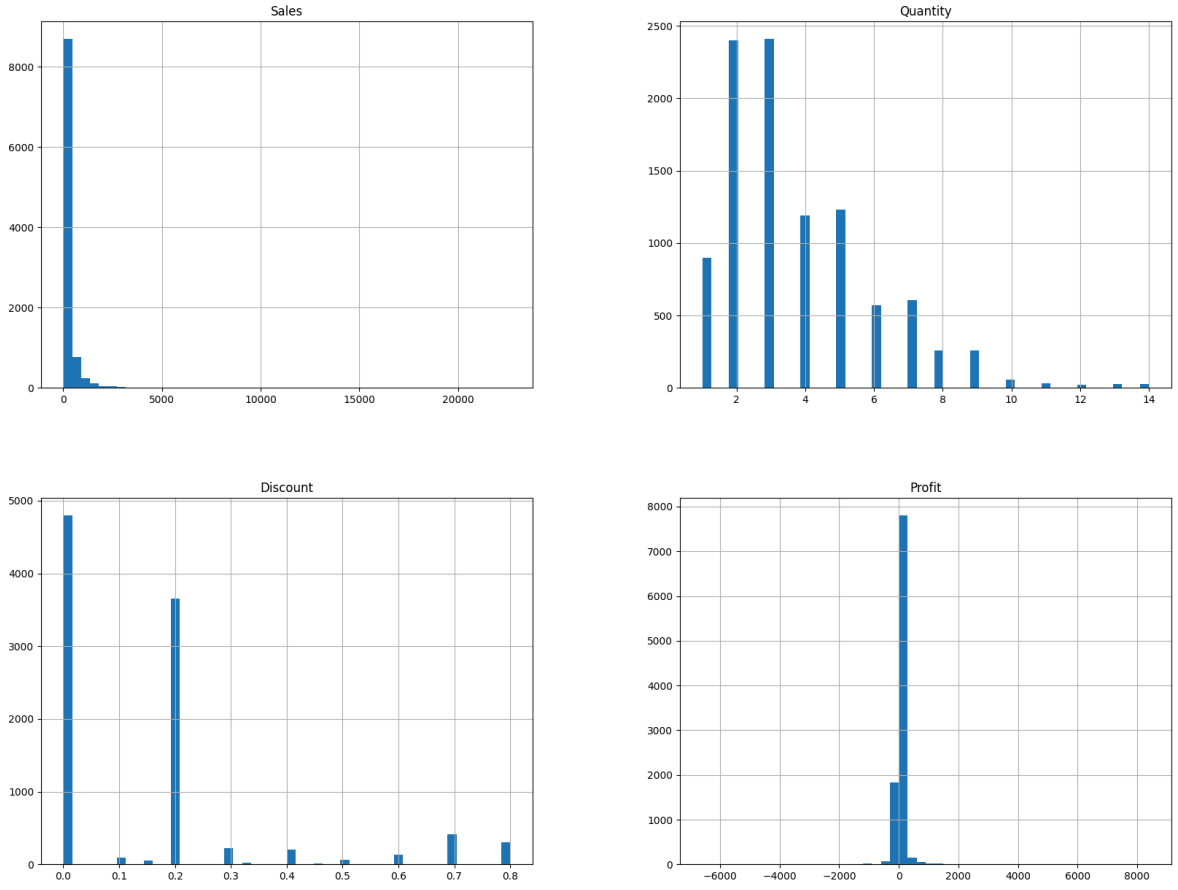
```
C:\Users\user\AppData\Local\Temp\ipykernel_16144\759940174.py:3: UserWarning:

Ignoring `palette` because no `hue` variable has been assigned.
```




The above graph showing that company is bariely making anny profit but when company offer discounts <20 % then only company makes a profit .

In [339…

```
df.hist(bins=50, figsize=(20,15))
plt.show()
```




**Skewness**

- Skewness is a statistical measure that quantifies the asymmetry of a probability distribution. It provides insight into the shape of the distribution by describing the degree to which the data is skewed to the left or right relative to the mean.
- In a perfectly symmetrical distribution, the mean, median, and mode are all equal, and the skewness value is close to zero. Positive skewness indicates that the right tail of the distribution is longer or fatter, while negative skewness indicates a longer or fatter left tail.
- Skewness can have implications for data analysis and modeling. It can impact the assumptions made by statistical tests and regression models that assume normality. Skewed data may require appropriate transformations to achieve normality and meet the assumptions of certain statistical methods.
- There are different methods to measure skewness, but one common measure is the skewness coefficient or skewness index. Some widely used skewness measures include Pearson's first skewness coefficient, which is based on the third standardized moment, and the Fisher-Pearson standardized moment coefficient.

In [340…
```
df_nm.skew()
```

Out[340…
```
Postal Code    -0.128526
Sales          12.972752
Quantity        1.278545
Discount        1.684295
Profit          7.561432
dtype: float64
```

- Sales is only has skew data.

## Representing Total Sales by state on map of US.

In [341…
```
import plotly.express as px
import plotly.graph_objects as go
from plotly.subplots import make_subplots
```

In [342…
```
state_code = {'Alabama': 'AL','Alaska': 'AK','Arizona': 'AZ','Arkansas': 'AR','C
df['state_code'] = df.State.apply(lambda x: state_code[x])
```

In [343…
```
state_data = df[['Sales', 'Profit', 'state_code']].groupby(['state_code']).sum()


fig = go.Figure(data=go.Choropleth(
    locations=state_data.index,
    z = state_data.Sales,
    locationmode = 'USA-states',
    colorscale = 'Reds',
    colorbar_title = 'Sales in USD',
))

fig.update_layout(
    title_text = 'Total State-Wise Sales',
```

```
    geo_scope='usa',
    height=800,
)

fig.show()
```

# 8. Conclusion and Recommendations.

## 8.1 Summarize the main insights derived from the analysis.

- Profit in south & central is less.
- Profit in east & west regions Is better than south and central.
- Highest profit is earned in Copiers while Selling price for Chairs and Phones is extremely high compared to other products.
- Another interesting fact - people dont prefer to buy Tables and Bookcases from Superstore. Hence these departments are in loss.
- The store has wide variety of Office Supplies especially in Binders and Paper department.
- Negative correlation between profit and discount.
- Total sum of profit in sale of tables is negative.
- Profit is more in sale of copiers.
- No or very less profit in sale of supplies.
- Technology segment is more profitable.

### 8.2 Provide actionable recommendations for improving sales or addressing

identified challenges .

- The sale of tables should be stopped as it is producing high loss.
- Bookcasses and suppliers are producing negligible loss so their price should be increased or sale should be stopped.
- For the central region we should definitely look into the furniture category as we are suffering most losses there.
- The supply of technology should be increased as it appears to be promising way to increase profits exponentially as all the regions are gaining high profit from it.
- Texas, Ohio, Illinois and Pennsylvania are incurring huge loss even when the sale is good and the main reason is huge discounts. So, we shuld consider reducing these discounts if we want aprofit.
- From the bar graph we could see that from more than 20% discount, loss is happening so we should not give anymore than 20% discount in all the states especially the states incurring loss at the moment

# 9. Documentation.

### 9.1 Document your analysis process, including the tools and libraries used.

1. Pandas (pd): Pandas is a powerful data manipulation library in Python. It provides data structures and functions for efficiently handling and analyzing structured data, making it a popular choice for data analysis and preprocessing tasks.

2. NumPy (np): NumPy is a fundamental library for scientific computing in Python. It provides support for large, multi-dimensional arrays and matrices, along with a vast collection of mathematical functions to operate on these arrays efficiently.

3. Warnings: The warnings library is used to manage warning messages that may arise during the code execution. In this code, the `warnings.filterwarnings("ignore")` statement suppresses the display of warning messages.

4. Matplotlib.pyplot (plt): Matplotlib is a widely-used plotting library in Python. The pyplot module provides a simple interface for creating various types of plots, charts, and visualizations.

5. Seaborn (sns): Seaborn is a Python data visualization library built on top of Matplotlib. It provides a high-level interface for creating aesthetically pleasing statistical graphics. Seaborn enhances Matplotlib's functionalities and offers additional plot types, color palettes, and themes.

These visualization libraries are essential tools for analyzing and presenting data in a meaningful and visually appealing way. They provide a wide range of functions and methods to create various types of plots, charts, and graphs, allowing for effective data exploration and communication of insights.

----------END----------

In [ ]:

In [ ]: