Data Science and Business Analytics Task 3 :-

INTERN NAME : TWINKLE PANDEY

To conduct Exploratory Data Analysis (EDA) on the 'SampleSuperstore' dataset, we will start by loading the dataset and examining its structure. We will clean the data to handle any missing values, duplicates, and inconsistencies. Using various visualizations, we will identify trends and patterns to uncover weak areas and potential business problems. Our goal is to provide insights and suggest strategies to improve profitability

Exploratory Data Analysis

Problem Statement:

1. Perform 'Exploratory Data Analysis' on dataset 'SampleSuperstore'.
2. As manager,try to find weak areas where you can work to make more profit.
3. What all business problems you can derive by exploring the data?

Importing Required Libraries

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

Importing Dataset

```
path = "/content/SampleSuperstore.csv"
dataset = pd.read_csv(path)
```

Reading the dataset

```
dataset.head() #loads the first 5 rows
```

	Ship Mode	Segment	Country	City	State	Postal Code	Region	Category	Su Catego
0	Second Class	Consumer	United States	Henderson	Kentucky	42420	South	Furniture	Bookcas
1	Second Class	Consumer	United States	Henderson	Kentucky	42420	South	Furniture	Cha
2	Second Class	Corporate	United States	Los Angeles	California	90036	West	Office Supplies	Lab

Next steps: Generate code with `dataset` View recommended plots

```
dataset.tail() #loads the last 5 rows
```

	Ship Mode	Segment	Country	City	State	Postal Code	Region	Category	
9989	Second Class	Consumer	United States	Miami	Florida	33180	South	Furniture	F
9990	Standard Class	Consumer	United States	Costa Mesa	California	92627	West	Furniture	F
9991	Standard Class	Consumer	United States	Costa Mesa	California	92627	West	Technology	

Checking the number of elements in each dimension in an array

```
dataset.shape
```

```
(9994, 13)
```

Checking the information of Data

```
dataset.info()  #Returns the concise summary of the dataset
```

```
<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
```

```
dataset.describe() #returns the statistical data
```

	Postal Code	Sales	Quantity	Discount	Profit
count	9994.000000	9994.000000	9994.000000	9994.000000	9994.000000
mean	55190.379428	229.858001	3.789574	0.156203	28.656896
std	32063.693350	623.245101	2.225110	0.206452	234.260108
min	1040.000000	0.444000	1.000000	0.000000	-6599.978000
25%	23223.000000	17.280000	2.000000	0.000000	1.728750
50%	56430.500000	54.490000	3.000000	0.200000	8.666500
75%	90008.000000	209.940000	5.000000	0.200000	29.364000
max	99301.000000	22638.480000	14.000000	0.800000	8399.976000

Checking the missing values

```
dataset.isnull().sum()
```

```
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
```

Checking for the duplicate data

```
dataset.duplicated().sum()
```

```
17
```

Dropping the duplicated data

```
dataset.drop_duplicates()
```

	Ship Mode	Segment	Country	City	State	Postal Code	Region	Category	
0	Second Class	Consumer	United States	Henderson	Kentucky	42420	South	Furniture	E
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	F
9990	Standard	Consumer	United	Costa Mesa	California	92627	West	Furniture	F

```
dataset.nunique() # Displays the unique data now
```

```
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
```

Dropping irrelevant columns

```
col = ["Postal Code"]
```

```
dataset1 = dataset.drop(columns = col,axis = 1)
```

```
dataset1
```

	Ship Mode	Segment	Country	City	State	Region	Category	Sub-Category
0	Second Class	Consumer	United States	Henderson	Kentucky	South	Furniture	Bookcases
1	Second Class	Consumer	United States	Henderson	Kentucky	South	Furniture	Chairs
2	Second Class	Corporate	United States	Los Angeles	California	West	Office Supplies	Labels
3	Standard Class	Consumer	United States	Fort Lauderdale	Florida	South	Furniture	Tables
4	Standard Class	Consumer	United States	Fort Lauderdale	Florida	South	Office Supplies	Storage
...	...	...	...	...	...	...	...	..
9989	Second Class	Consumer	United States	Miami	Florida	South	Furniture	Furnishings
9990	Standard	Consumer	United	Costa Mesa	California	West	Furniture	Furnishings

Next steps: Generate code with dataset1 View recommended plots

Checking statistical relation between the various rows & columns

```
# Select only numeric columns
numeric_data = dataset1[['Sales', 'Quantity', 'Discount', 'Profit']]

# Calculate the correlation matrix
correlation_matrix = numeric_data.corr()

# Print the correlation matrix
correlation_matrix
```

	Sales	Quantity	Discount	Profit
Sales	1.000000	0.200795	-0.028190	0.479064
Quantity	0.200795	1.000000	0.008623	0.066253
Discount	-0.028190	0.008623	1.000000	-0.219487
Profit	0.479064	0.066253	-0.219487	1.000000

Next steps: Generate code with correlation_matrix View recommended plots

```
# Calculate the covariance matrix
covariance_matrix = numeric_data.cov()

# Print the covariance matrix
covariance_matrix
```

	Sales	Quantity	Discount	Profit
Sales	388434.455308	278.459923	-3.627228	69944.096586
Quantity	278.459923	4.951113	0.003961	34.534769
Discount	-3.627228	0.003961	0.042622	-10.615173
Profit	69944.096586	34.534769	-10.615173	54877.798055

Next steps: Generate code with covariance_matrix View recommended plots

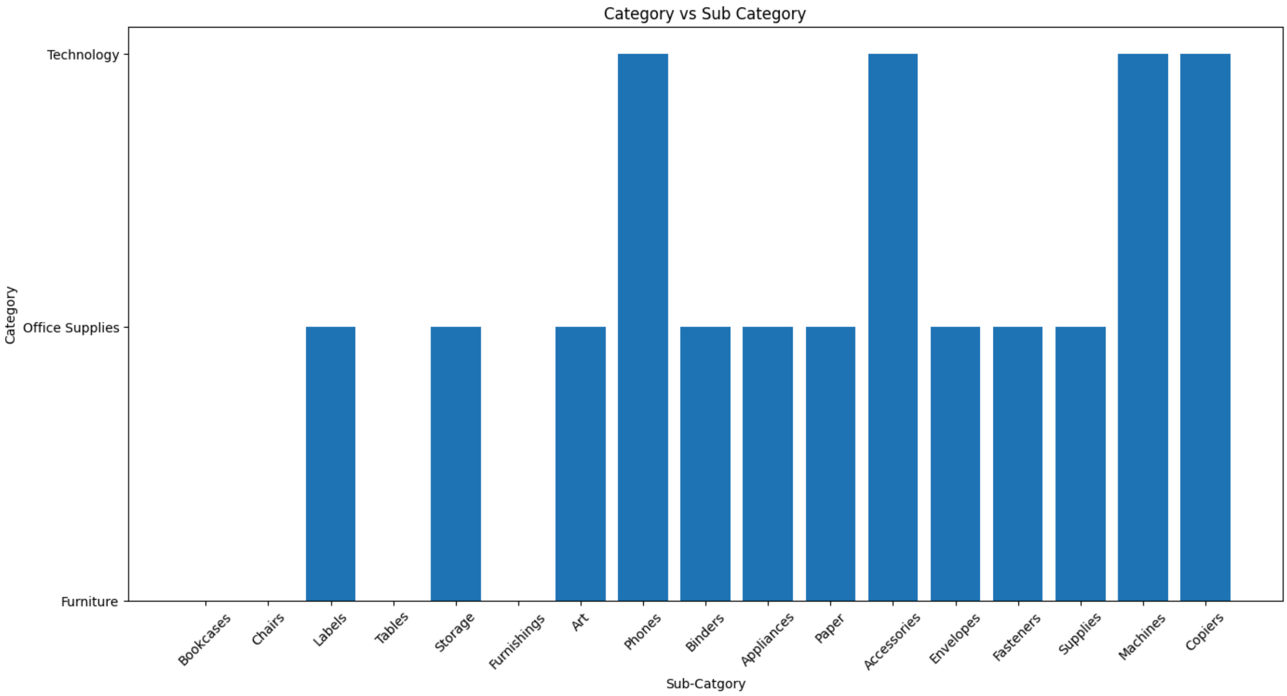
```
dataset1.head() #loads first five rows
```

	Ship Mode	Segment	Country	City	State	Region	Category	Sub-Category	S
0	Second Class	Consumer	United States	Henderson	Kentucky	South	Furniture	Bookcases	261.
1	Second Class	Consumer	United States	Henderson	Kentucky	South	Furniture	Chairs	731.
2	Second	Corporate	United	Los	California	West	Office	Labels	14.

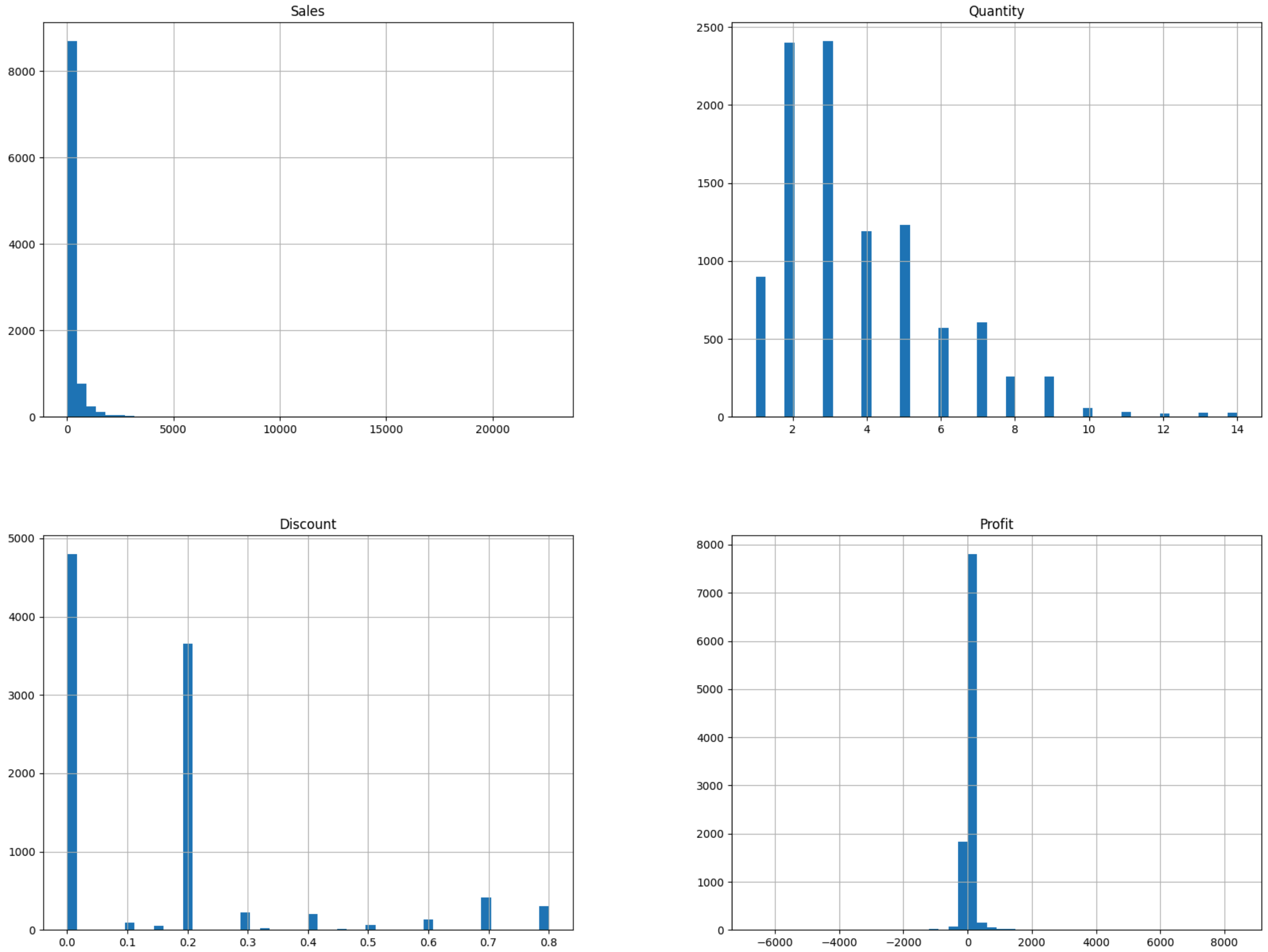
Next steps: Generate code with dataset1 View recommended plots

Data Visualisation

```
plt.figure(figsize=(16,8))
plt.bar('Sub-Category','Category', data=dataset1)
plt.title('Category vs Sub Category')
plt.xlabel('Sub-Catgory')
plt.ylabel('Category')
plt.xticks(rotation=45)
plt.show()
```



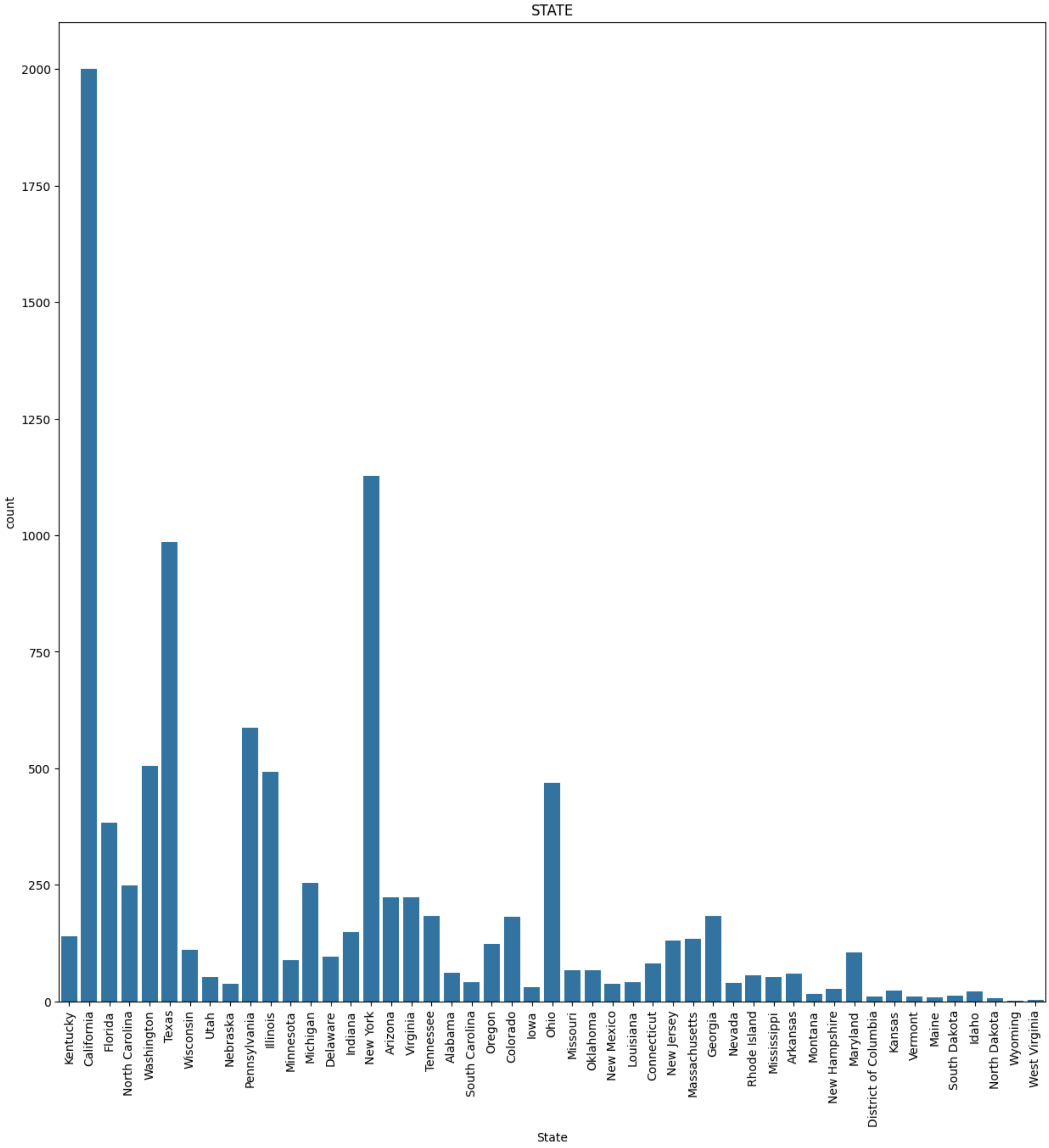
```
dataset1.hist(bins=50,figsize=(20,15))
plt.show();
```


Sales
0
2000
4000
6000
8000
0
5000
10000
15000
20000
Quantity
0
500
1000
1500
2000
2500
2
4
6
8
10
12
14
Discount
0
1000
2000
3000
4000
5000
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
Profit
0
1000
2000
3000
4000
5000
6000
7000
8000
−6000
−4000
−2000
0
2000
4000
6000
8000

```
# Count the total repeatable states
dataset1['State'].value_counts()
```

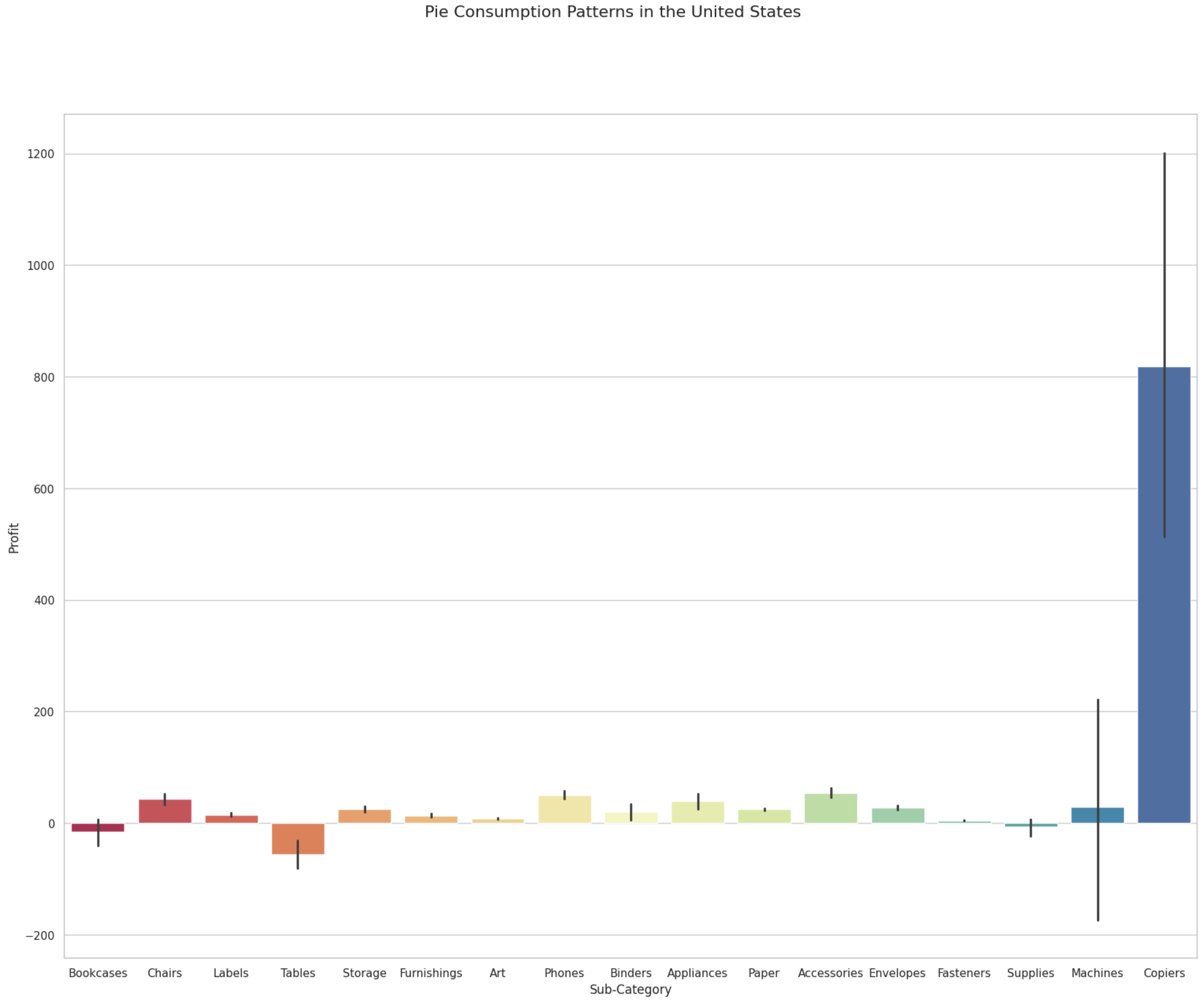
```
State
California              2001
New York                1128
Texas                    985
Pennsylvania             587
Washington               506
Illinois                 492
Ohio                     469
Florida                  383
Michigan                 255
North Carolina           249
Arizona                  224
Virginia                 224
Georgia                  184
Tennessee                183
Colorado                 182
Indiana                  149
Kentucky                 139
Massachusetts            135
New Jersey               130
Oregon                   124
Wisconsin                110
Maryland                 105
Delaware                  96
Minnesota                 89
Connecticut               82
Oklahoma                  66
Missouri                  66
Alabama                   61
Arkansas                  60
Rhode Island              56
Utah                      53
Mississippi               53
Louisiana                 42
South Carolina            42
Nevada                    39
Nebraska                  38
New Mexico                37
Iowa                      30
New Hampshire             27
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
```

```
plt.figure(figsize=(15,15))
sns.countplot(x=dataset1['State'])
plt.xticks(rotation=90)
plt.title("STATE")
plt.show()
```


STATE
count
0
250
500
750
1000
1250
1500
1750
2000
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
State

```
sns.set(style="whitegrid")
plt.figure(2, figsize=(20,15))
sns.barplot(x='Sub-Category',y='Profit', data=dataset, palette='Spectral')
plt.suptitle('Pie Consumption Patterns in the United States', fontsize=16)
plt.show()
```

```
<ipython-input-47-8a12df664a8c>:3: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14

  sns.barplot(x='Sub-Category',y='Profit', data=dataset, palette='Spectral')
```

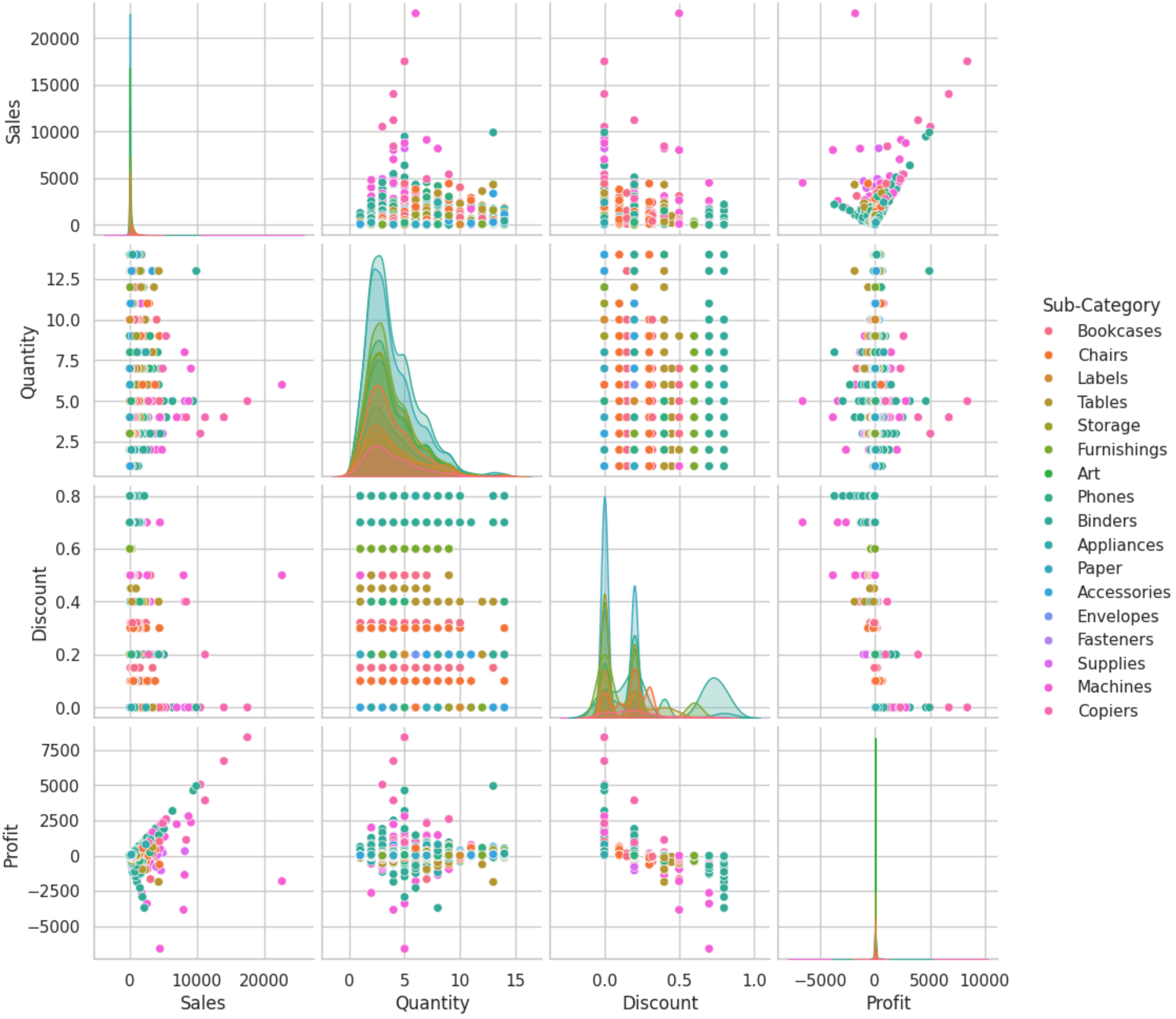


Pie Consumption Patterns in the United States

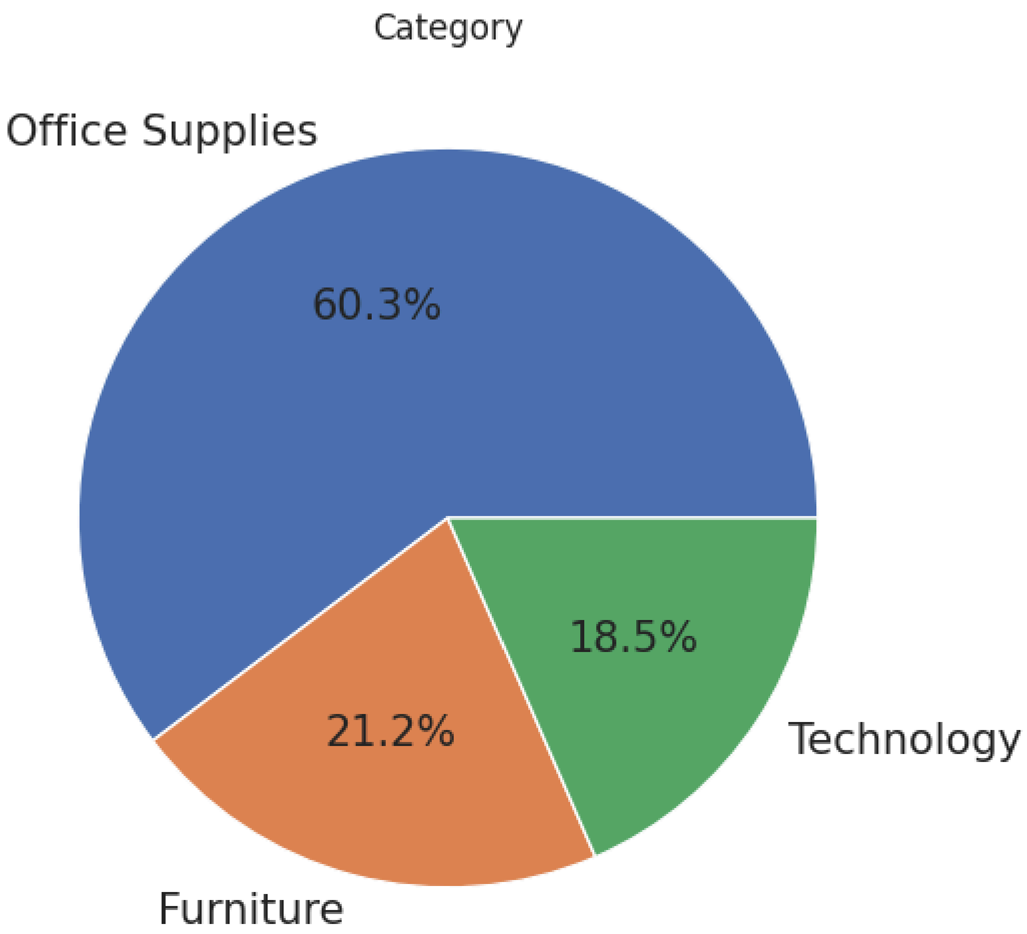
```
figsize=(15,10)
sns.pairplot(dataset1,hue='Sub-Category')
plt.show
```

```
matplotlib.pyplot.show
def show(*args, **kwargs)
```

```
/usr/local/lib/python3.10/dist-packages/matplotlib/pyplot.py
Display all open figures.

Parameters
----------
block : bool, optional
```

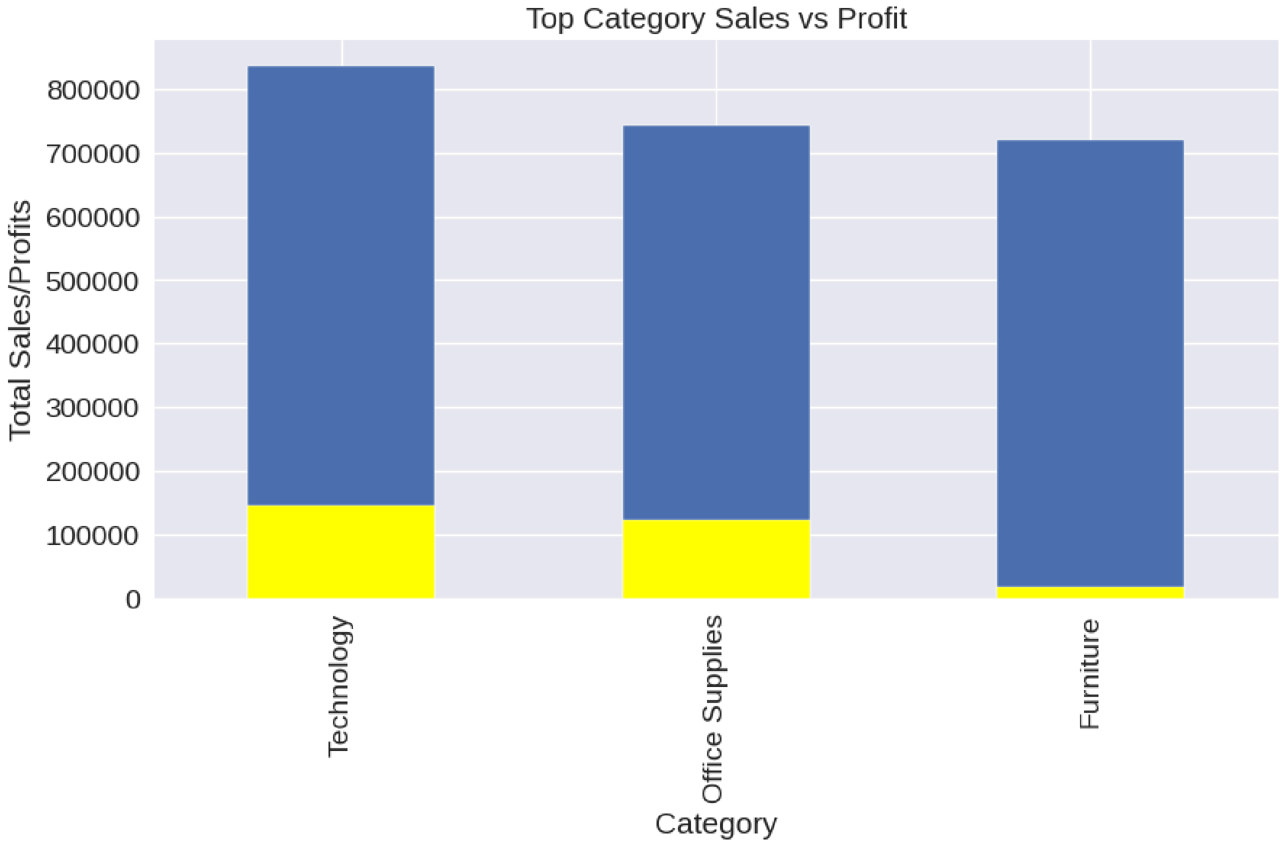


```
plt.figure(figsize = (6,6))
textprops = {"fontsize":15}
plt.title('Category')
plt.pie(dataset['Category'].value_counts(), labels=dataset['Category'].value_counts().in
plt.show()
```



```
# computing top categories in terms of sales from first 100 observations

top_category_s = dataset.groupby("Category").Sales.sum().nlargest(n=100)

# computing top categories in terms of profit from first 100 observations

top_category_p = dataset.groupby("Category").Profit.sum().nlargest(n=100)

# plotting to see it visually

plt.style.use('seaborn')
top_category_s.plot(kind = 'bar',figsize = (10,5),fontsize = 14)
top_category_p.plot(kind = 'bar',figsize = (10,5),fontsize = 14,color='yellow')
plt.xlabel('Category',fontsize = 15)
plt.ylabel('Total Sales/Profits',fontsize = 15)
plt.title("Top Category Sales vs Profit",fontsize = 15)
plt.show()
```

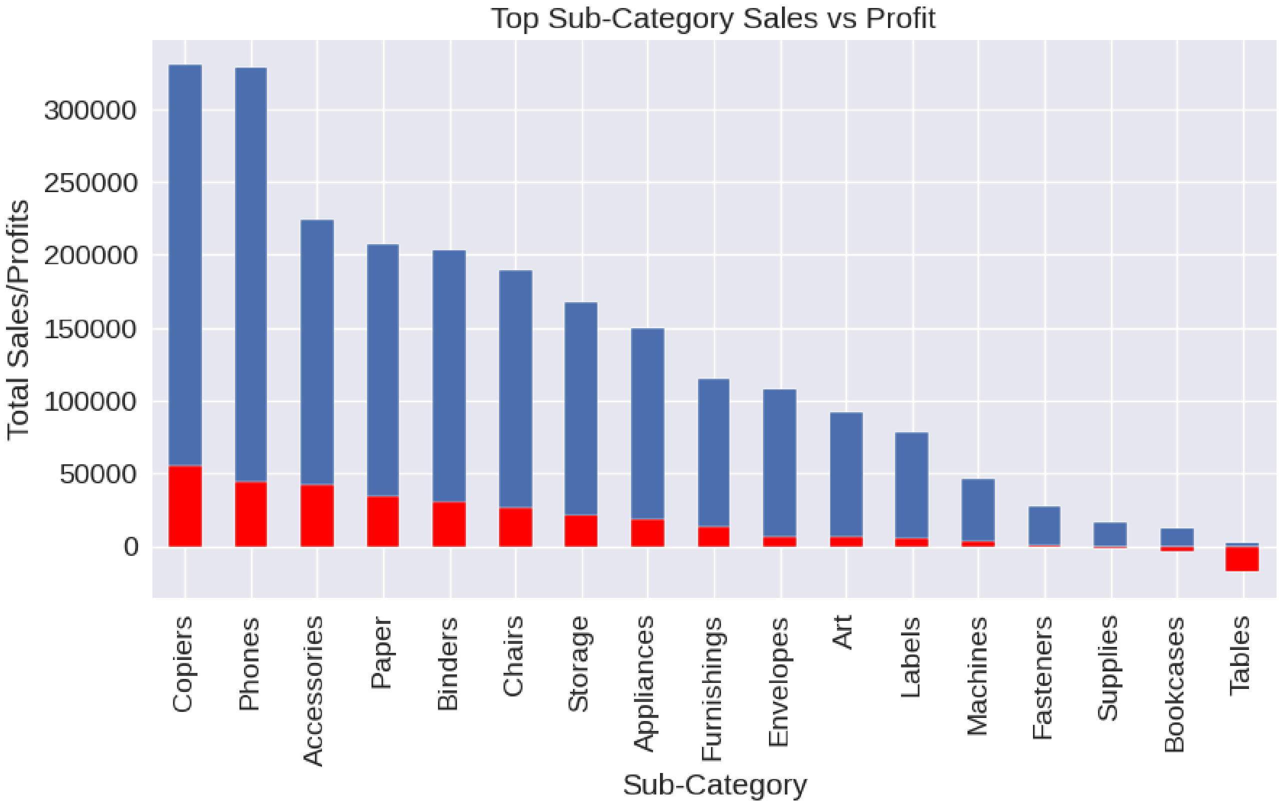
```
<ipython-input-57-1747aad01761>:11: MatplotlibDeprecationWarning: The seaborn styles
  plt.style.use('seaborn')
```



Visualising the Sub Categories

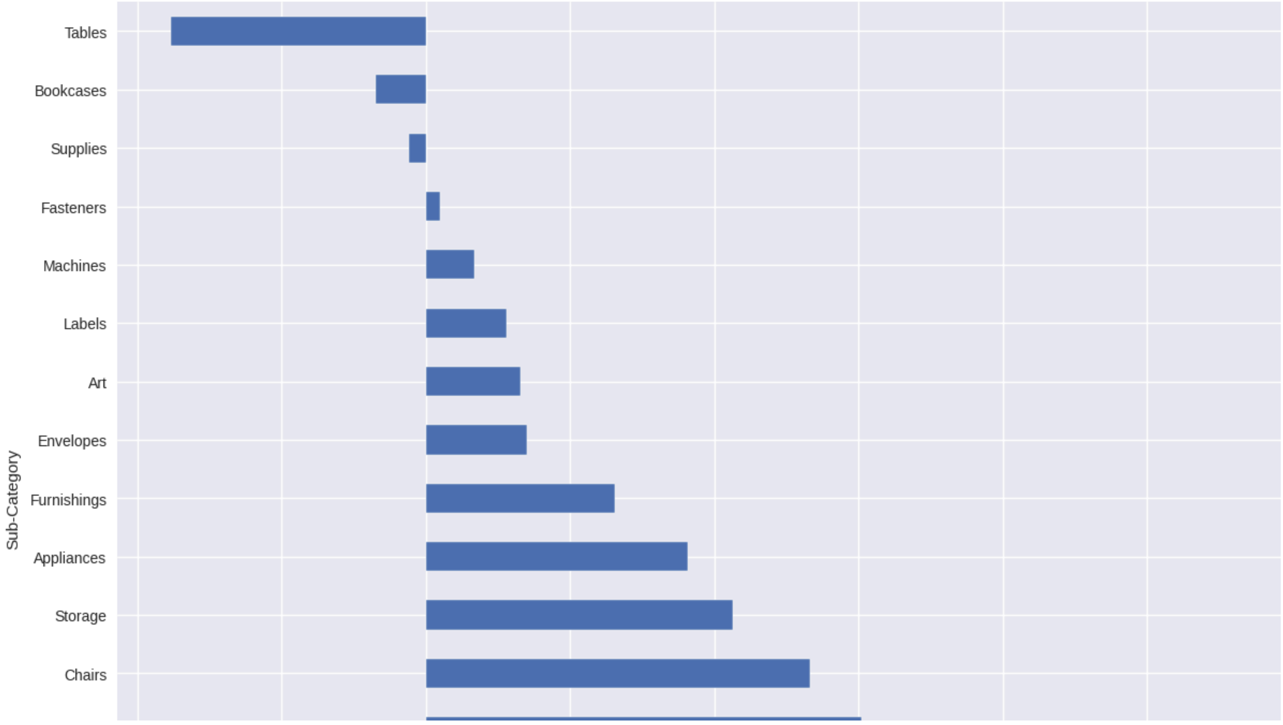
```
# computing top sub-categories in terms of sales from first 100 observations

top_subcategory_s = dataset.groupby("Sub-Category").Sales.sum().nlargest(n = 100)

# computing top sub-categories in terms of profit from first 100 observations

top_subcategory_p = dataset.groupby("Sub-Category").Profit.sum().nlargest(n = 100)

# plotting to see it visually

plt.style.use('seaborn')
top_subcategory_s.plot(kind = 'bar',figsize = (10,5),fontsize = 14)
top_subcategory_p.plot(kind = 'bar',figsize = (10,5),fontsize = 14, color = 'red')
plt.xlabel('Sub-Category',fontsize = 15)
plt.ylabel('Total Sales/Profits',fontsize = 15)
plt.title("Top Sub-Category Sales vs Profit",fontsize = 15)
plt.show()
```

```
<ipython-input-58-f6bf79d7e4f8>:11: MatplotlibDeprecationWarning: The seaborn styles
  plt.style.use('seaborn')
```



```
# A more detailed view
plt.figure(figsize=(14,12))
statewise = dataset.groupby(['Sub-Category'])['Profit'].sum().nlargest(50)
statewise.plot.barh() # h for horizontal
```

```
<Axes: ylabel='Sub-Category'>
```


Tables
Bookcases
Supplies
Fasteners
Machines
Labels
Art
Envelopes
Furnishings
Appliances
Storage
Chairs
Sub-Category