Name- Ishika Bhardwaj

Roll no.- 8209

Global Superstore- Statistical Analysis and Visualization

Global Super Store is a data set which has around 50000 values. Its a customer centric data set, which has the data of all the orders that have been placed through different vendors and markets.

Dataset Download Link: https://www.kaggle.com/shekpaul/global-superstore

Importing Required Libraries

In [3]:
```
#importing all the required libraries for data manipulation and analysis
!pip install xlrd

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

```
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: xlrd in /home/ishika/.local/lib/python3.8/site-packages (2.0.1)
WARNING: You are using pip version 21.2.4; however, version 21.3.1 is available.
You should consider upgrading via the '/usr/bin/python3 -m pip install --upgrade pip' command.
```

Converting .xls to CSV

In [4]:
```
sales_data = pd.read_excel("Global Superstore.xls")
```

In [5]:
```
sales_data.to_csv("Global_Superstore.csv", index = None, header = True)

#converting excel to csv
df = pd.DataFrame(pd.read_csv("Global_Superstore.csv"))
```

Data Exploring

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

Out[6]:

	Row ID	Order ID	Order Date	Ship Date	Ship Mode	Customer ID	Customer Name	Segment	City	State	...	Product ID	Category	Sub-Category	Product Name	Sales	Quantity	Discount	Profit	Shipping Cost	Order Priority
0	32298	CA-2012-124891	2012-07-31	2012-07-31	Same Day	RH-19495	Rick Hansen	Consumer	New York City	New York	...	TEC-AC-10003033	Technology	Accessories	Plantronics CS510 - Over-the-Head monaural Wir...	2309.650	7	0.0	762.1845	933.57	Critical
1	26341	IN-2013-77878	2013-02-05	2013-02-07	Second Class	JR-16210	Justin Ritter	Corporate	Wollongong	New South Wales	...	FUR-CH-10003950	Furniture	Chairs	Novimex Executive Leather Armchair, Black	3709.395	9	0.1	-288.7650	923.63	Critical
2	25330	IN-2013-71249	2013-10-17	2013-10-18	First Class	CR-12730	Craig Reiter	Consumer	Brisbane	Queensland	...	TEC-PH-10004664	Technology	Phones	Nokia Smart Phone, with Caller ID	5175.171	9	0.1	919.9710	915.49	Medium
3	13524	ES-2013-1579342	2013-01-28	2013-01-30	First Class	KM-16375	Katherine Murray	Home Office	Berlin	Berlin	...	TEC-PH-10004583	Technology	Phones	Motorola Smart Phone, Cordless	2892.510	5	0.1	-96.5400	910.16	Medium

	Row ID	Order ID	Order Date	Ship Date	Ship Mode	Customer ID	Customer Name	Segment	City	State	...	Product ID	Category	Sub-Category	Product Name	Sales	Quantity	Discount	Profit	Shipping Cost	Order Priority
4	47221	SG-2013-4320	2013-11-05	2013-11-06	Same Day	RH-9495	Rick Hansen	Consumer	Dakar	Dakar	...	TEC-SHA-10000501	Technology	Copiers	Sharp Wireless Fax, High-Speed	2832.960	8	0.0	311.5200	903.04	Critical

5 rows × 24 columns

In [7]:
```
# df.shape
print("This dataset has\nColumns: {}\nRows: {}".format(df.shape[0],df.shape[1]))
```

```
This dataset has
Columns: 51290
Rows: 24
```

In [8]:
```
df.columns
```

Out[8]:
```
Index(['Row ID', 'Order ID', 'Order Date', 'Ship Date', 'Ship Mode',
       'Customer ID', 'Customer Name', 'Segment', 'City', 'State', 'Country',
       'Postal Code', 'Market', 'Region', 'Product ID', 'Category',
       'Sub-Category', 'Product Name', 'Sales', 'Quantity', 'Discount',
       'Profit', 'Shipping Cost', 'Order Priority'],
      dtype='object')
```

In [9]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 51290 entries, 0 to 51289
Data columns (total 24 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   Row ID          51290 non-null  int64  
 1   Order ID        51290 non-null  object 
 2   Order Date      51290 non-null  object 
 3   Ship Date       51290 non-null  object 
 4   Ship Mode       51290 non-null  object 
 5   Customer ID     51290 non-null  object 
 6   Customer Name   51290 non-null  object 
 7   Segment         51290 non-null  object 
 8   City            51290 non-null  object 
 9   State           51290 non-null  object 
 10  Country         51290 non-null  object 
 11  Postal Code     9994 non-null   float64
 12  Market          51290 non-null  object 
 13  Region          51290 non-null  object 
 14  Product ID      51290 non-null  object 
 15  Category        51290 non-null  object 
 16  Sub-Category    51290 non-null  object 
 17  Product Name    51290 non-null  object 
 18  Sales           51290 non-null  float64
 19  Quantity        51290 non-null  int64  
 20  Discount        51290 non-null  float64
 21  Profit          51290 non-null  float64
 22  Shipping Cost   51290 non-null  float64
 23  Order Priority  51290 non-null  object 
dtypes: float64(5), int64(2), object(17)
memory usage: 9.4+ MB
```

In [10]:
```
df.describe()
```

Out[10]:

	Row ID	Postal Code	Sales	Quantity	Discount	Profit	Shipping Cost
count	51290.00000	9994.000000	51290.000000	51290.000000	51290.000000	51290.000000	51290.000000
mean	25645.50000	55190.379428	246.490581	3.476545	0.142908	28.610982	26.375818
std	14806.29199	32063.693350	487.565361	2.278766	0.212280	174.340972	57.296810
min	1.00000	1040.000000	0.444000	1.000000	0.000000	-6599.978000	0.002000
25%	12823.25000	23223.000000	30.758625	2.000000	0.000000	0.000000	2.610000
50%	25645.50000	56430.500000	85.053000	3.000000	0.000000	9.240000	7.790000
75%	38467.75000	90008.000000	251.053200	5.000000	0.200000	36.810000	24.450000
max	51290.00000	99301.000000	22638.480000	14.000000	0.850000	8399.976000	933.570000

Data Cleaning

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

Out[11]:

	Row ID	Order ID	Order Date	Ship Date	Ship Mode	Customer ID	Customer Name	Segment	City	State	...	Product ID	Category	Sub-Category	Product Name	Sales	Quantity	Discount	Profit	Shipping Cost	Order Priority
0	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
51285	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False	False	False
51286	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False	False	False
51287	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False	False	False
51288	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False	False	False
51289	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False	False	False

51290 rows × 24 columns

In [12]:
```
print(df.isnull().sum())
print("-----------------------")
print("Total             {}".format(df.isnull().sum().sum()))
```

```
Row ID                0
Order ID              0
Order Date            0
Ship Date             0
Ship Mode             0
Customer ID           0
Customer Name         0
Segment               0
City                  0
State                 0
Country               0
Postal Code       41296
Market                0
```

```
Region                0
Product ID            0
Category              0
Sub-Category          0
Product Name          0
Sales                 0
Quantity              0
Discount              0
Profit                0
Shipping Cost         0
Order Priority        0
dtype: int64
------------------------
Total             41296
```

In [13]:
```
sales_data = df.dropna(axis=1)
```

In [14]:
```
print(sales_data.isnull().sum())
print("------------------------")
print("Total             {}".format(sales_data.isnull().sum().sum()))
```

```
Row ID            0
Order ID          0
Order Date        0
Ship Date         0
Ship Mode         0
Customer ID       0
Customer Name     0
Segment           0
City              0
State             0
Country           0
Market            0
Region            0
Product ID        0
Category          0
Sub-Category      0
Product Name      0
Sales             0
Quantity          0
Discount          0
Profit            0
Shipping Cost     0
Order Priority    0
dtype: int64
------------------------
Total             0
```

In [15]:
```
#checking if dataset has any duplicate rows
sales_data.duplicated().sum()
```

Out[15]:
```
0
```

In [16]:
```
#converting the Dates in proper datetime format
sales_data['Order Date'] = pd.to_datetime(sales_data['Order Date'], errors = 'coerce')
sales_data['Ship Date'] = pd.to_datetime(sales_data['Ship Date'], errors = 'coerce')
```

```
/tmp/ipykernel_34406/253882909.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead
```

```
See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  sales_data['Order Date'] = pd.to_datetime(sales_data['Order Date'], errors = 'coerce')
/tmp/ipykernel_34406/253882909.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  sales_data['Ship Date'] = pd.to_datetime(sales_data['Ship Date'], errors = 'coerce')
```

In [17]:

```
sales_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 51290 entries, 0 to 51289
Data columns (total 23 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Row ID          51290 non-null  int64
 1   Order ID        51290 non-null  object
 2   Order Date      51290 non-null  datetime64[ns]
 3   Ship Date       51290 non-null  datetime64[ns]
 4   Ship Mode       51290 non-null  object
 5   Customer ID     51290 non-null  object
 6   Customer Name   51290 non-null  object
 7   Segment         51290 non-null  object
 8   City            51290 non-null  object
 9   State           51290 non-null  object
 10  Country         51290 non-null  object
 11  Market          51290 non-null  object
 12  Region          51290 non-null  object
 13  Product ID      51290 non-null  object
 14  Category        51290 non-null  object
 15  Sub-Category    51290 non-null  object
 16  Product Name    51290 non-null  object
 17  Sales           51290 non-null  float64
 18  Quantity        51290 non-null  int64
 19  Discount        51290 non-null  float64
 20  Profit          51290 non-null  float64
 21  Shipping Cost   51290 non-null  float64
 22  Order Priority  51290 non-null  object
dtypes: datetime64[ns](2), float64(4), int64(2), object(15)
memory usage: 9.0+ MB
```

In [18]:

```
sales_data['Year'] = sales_data['Order Date'].dt.year #extracting the order year from orderdate column
sales_data['Month'] = sales_data['Order Date'].dt.month  #extracting the order month from orderdate column
sales_data['Day'] = sales_data['Order Date'].dt.day   #extracting the order day from orderdate column
sales_data["Month_year"] = sales_data['Order Date'].apply(lambda x: x.strftime('%Y-%m'))
```

In [19]:

```
sales_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 51290 entries, 0 to 51289
Data columns (total 27 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Row ID          51290 non-null  int64
 1   Order ID        51290 non-null  object
 2   Order Date      51290 non-null  datetime64[ns]
 3   Ship Date       51290 non-null  datetime64[ns]
 4   Ship Mode       51290 non-null  object
 5   Customer ID     51290 non-null  object
 6   Customer Name   51290 non-null  object
 7   Segment         51290 non-null  object
 8   City            51290 non-null  object
 9   State           51290 non-null  object
 10  Country         51290 non-null  object
 11  Market          51290 non-null  object
 12  Region          51290 non-null  object
 13  Product ID      51290 non-null  object
 14  Category        51290 non-null  object
 15  Sub-Category    51290 non-null  object
 16  Product Name    51290 non-null  object
 17  Sales           51290 non-null  float64
 18  Quantity        51290 non-null  int64
 19  Discount        51290 non-null  float64
 20  Profit          51290 non-null  float64
 21  Shipping Cost   51290 non-null  float64
 22  Order Priority  51290 non-null  object
 23  Year            51290 non-null  int64
 24  Month           51290 non-null  int64
 25  Day             51290 non-null  int64
 26  Month_year      51290 non-null  object
dtypes: datetime64[ns](2), float64(4), int64(5), object(16)
memory usage: 10.6+ MB
```

In [20]:
```
#exporting the cleaned data into a csv file so we can perform operations on that
sales_data.to_csv("GlobalSuperstore_Cleaned.csv")
```

In [21]:
```
sales_data.columns
```

Out[21]:
```
Index(['Row ID', 'Order ID', 'Order Date', 'Ship Date', 'Ship Mode',
       'Customer ID', 'Customer Name', 'Segment', 'City', 'State', 'Country',
       'Market', 'Region', 'Product ID', 'Category', 'Sub-Category',
       'Product Name', 'Sales', 'Quantity', 'Discount', 'Profit',
       'Shipping Cost', 'Order Priority', 'Year', 'Month', 'Day',
       'Month_year'],
      dtype='object')
```

Exploring the Dataset

In [22]:
```
sales_data["Country"].unique()
```

Out[22]:
```
array(['United States', 'Australia', 'Germany', 'Senegal', 'New Zealand',
       'Afghanistan', 'Saudi Arabia', 'Brazil', 'China', 'France',
       'Italy', 'Tanzania', 'Poland', 'United Kingdom', 'Mexico',
       'El Salvador', 'Taiwan', 'India', 'Dominican Republic',
       'Democratic Republic of the Congo', 'Indonesia', 'Uruguay', 'Iran',
       'Mozambique', 'Bangladesh', 'Spain', 'Ukraine', 'Nicaragua',
       'Morocco', 'Canada', 'Philippines', 'Austria', 'Colombia',
       'Netherlands', 'Malaysia', 'Ecuador', 'Thailand', 'Somalia',
       'Guatemala', 'Belarus', 'Cambodia', 'South Africa', 'Japan',
       'Russia', 'Egypt', 'Azerbaijan', 'Lithuania', 'Argentina',
       'Lesotho', 'Vietnam', 'Cuba', 'Romania', 'Turkey', 'Cameroon',
       'Hungary', 'Singapore', 'Angola', 'Belgium', 'Pakistan', 'Finland',
       'Ghana', 'Zambia', 'Iraq', 'Liberia', 'Georgia', 'Switzerland',
       'Albania', 'Chad', 'Montenegro', 'Namibia', 'Portugal',
       'Madagascar', 'Sweden', 'Myanmar (Burma)', 'Jamaica', 'Qatar',
       'Republic of the Congo', 'Norway', 'Algeria', 'South Korea',
       'Nigeria', 'Estonia', "Cote d'Ivoire", 'Honduras', 'Paraguay',
       'Czech Republic', 'Central African Republic', 'Benin', 'Bolivia',
       'Chile', 'Martinique', 'Syria', 'Lebanon', 'Kenya', 'Mali',
       'Libya', 'Venezuela', 'Trinidad and Tobago', 'Ireland', 'Bulgaria',
       'Panama', 'Israel', 'Haiti', 'Barbados', 'Slovenia', 'Togo',
       'Mauritania', 'Guinea', 'Rwanda', 'Denmark', 'Niger',
       'Papua New Guinea', 'Mongolia', 'Sudan', 'Peru', 'Sierra Leone',
       'Bosnia and Herzegovina', 'Guinea-Bissau', 'Djibouti', 'Tunisia',
       'Croatia', 'Hong Kong', 'Nepal', 'Guadeloupe', 'Kyrgyzstan',
       'Zimbabwe', 'Uzbekistan', 'South Sudan', 'Gabon', 'Bahrain',
       'Yemen', 'Jordan', 'United Arab Emirates', 'Moldova', 'Swaziland',
       'Turkmenistan', 'Kazakhstan', 'Ethiopia', 'Uganda', 'Slovakia',
       'Sri Lanka', 'Tajikistan', 'Burundi', 'Macedonia', 'Eritrea',
       'Equatorial Guinea', 'Armenia'], dtype=object)
```

In [23]:
```
print("Global Supermarket has its reach over {} countries".format(sales_data["Country"].nunique()))
```

```
Global Supermarket has its reach over 147 countries
```

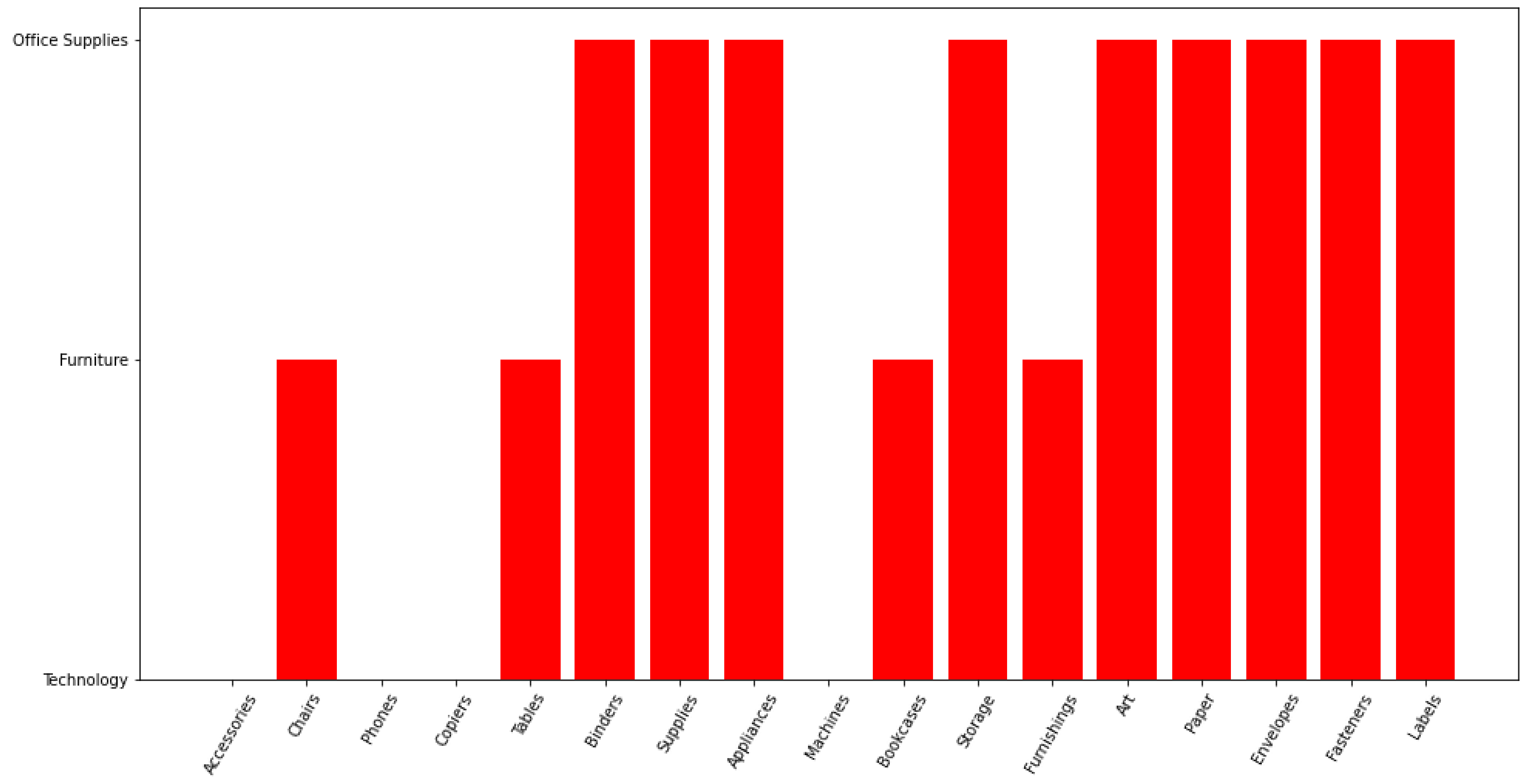
In [24]:
```
sales_data["Category"].unique()
```

Out[24]:
```
array(['Technology', 'Furniture', 'Office Supplies'], dtype=object)
```

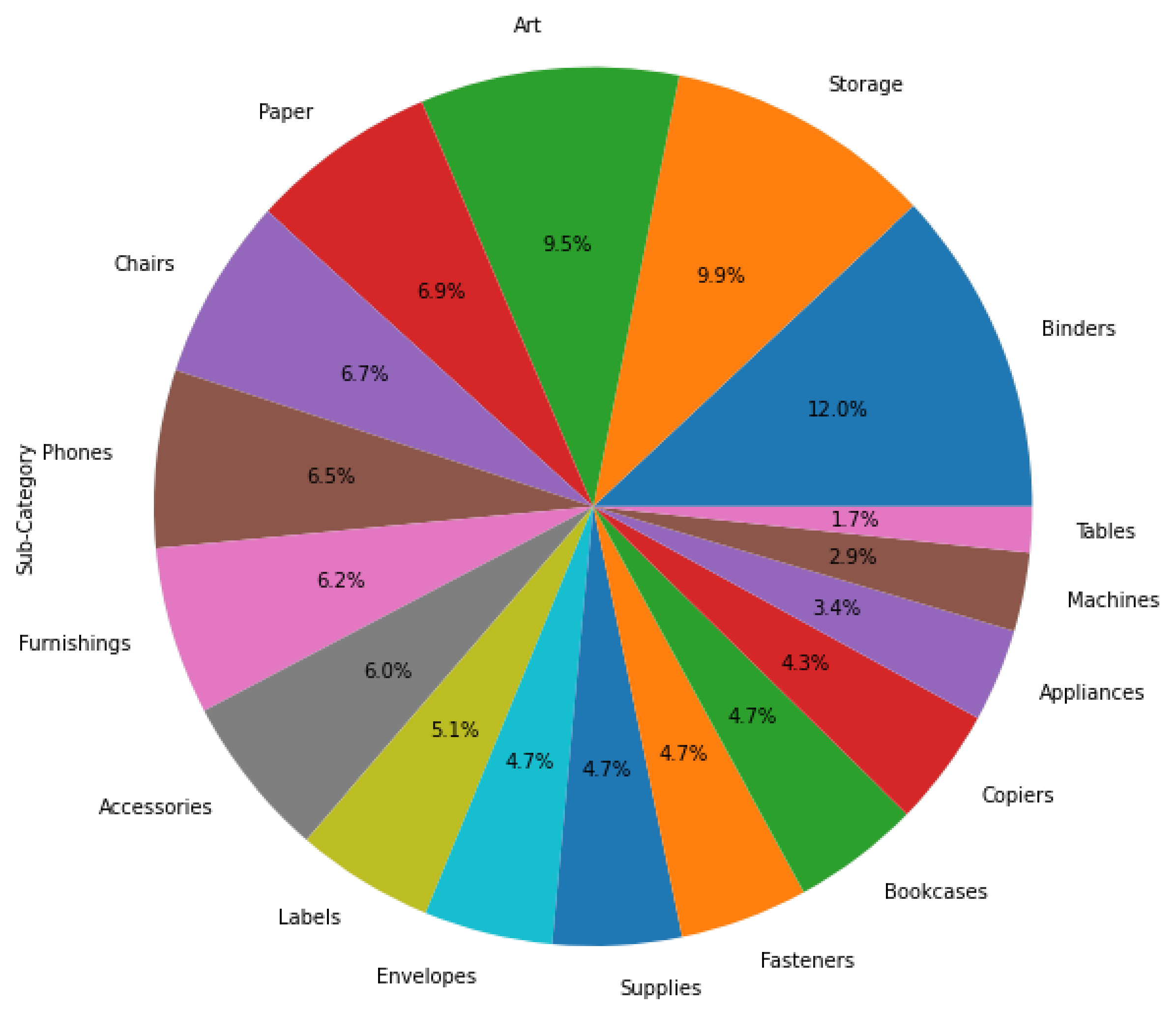
In [25]:
```
sales_data["Sub-Category"].unique()
```

Out[25]:
```
array(['Accessories', 'Chairs', 'Phones', 'Copiers', 'Tables', 'Binders',
       'Supplies', 'Appliances', 'Machines', 'Bookcases', 'Storage',
       'Furnishings', 'Art', 'Paper', 'Envelopes', 'Fasteners', 'Labels'],
      dtype=object)
```

In [28]:
```
plt.figure(figsize=(16,8))
plt.bar('Sub-Category','Category',data=sales_data,color='r')
plt.xticks(rotation=60)
plt.show()
```



```
In [30]: plt.figure(figsize=(10,15))
         sales_data['Sub-Category'].value_counts().plot.pie(autopct = "%1.1f%%")
         plt.show()
```



From the pie-chart we can infer that, Sales of Binders is the most and Sales of Tables is the least in th Global Superstore.

In [31]:
```
group_by_sub_category = sales_data.groupby("Sub-Category")
group_by_sub_category.first()
```

Out[31]:

	Row ID	Order ID	Order Date	Ship Date	Ship Mode	Customer ID	Customer Name	Segment	City	State	...	Sales	Quantity	Discount	Profit	Shipping Cost	Order Priority	Year	Month	Day	Month_year
Sub-Category																					
Accessories	32298	CA-2012-124891	2012-07-31	2012-07-31	Same Day	RH-19495	Rick Hansen	Consumer	New York City	New York	...	2309.650	7	0.00	762.1845	933.57	Critical	2012	7	31	2012-07
Appliances	27704	IN-2013-73951	2013-06-06	2013-06-08	Second Class	PF-19120	Peter Fuller	Consumer	Mudanjiang	Heilongjiang	...	3701.520	12	0.00	1036.0800	804.54	Critical	2013	6	6	2013-06
Art	14572	ES-2014-2591706	2014-09-22	2014-09-22	Same Day	SJ-20215	Sarah Jordon	Consumer	Vienna	Vienna	...	439.680	8	0.00	153.8400	179.84	Critical	2014	9	22	2014-09
Binders	40155	CA-2014-135909	2014-10-14	2014-10-21	Standard Class	JW-15220	Jane Waco	Corporate	Sacramento	California	...	5083.960	5	0.20	1906.4850	867.69	Low	2014	10	14	2014-10
Bookcases	25795	IN-2014-76016	2014-09-26	2014-09-28	Second Class	VG-21805	Vivek Grady	Corporate	Thiruvananthapuram	Kerala	...	5667.870	13	0.00	2097.0300	658.35	Medium	2014	9	26	2014-09
Chairs	26341	IN-2013-77878	2013-02-05	2013-02-07	Second Class	JR-16210	Justin Ritter	Corporate	Wollongong	New South Wales	...	3709.395	9	0.10	-288.7650	923.63	Critical	2013	2	5	2013-02
Copiers	47221	SG-2013-4320	2013-11-05	2013-11-06	Same Day	RH-9495	Rick Hansen	Consumer	Dakar	Dakar	...	2832.960	8	0.00	311.5200	903.04	Critical	2013	11	5	2013-11

	Row ID	Order ID	Order Date	Ship Date	Ship Mode	Customer ID	Customer Name	Segment	City	State	...	Sales	Quantity	Discount	Profit	Shipping Cost	Order Priority	Year	Month	Day	Month_year
Sub-Category																					
Envelopes	28658	IN-2013-37929	2013-09-20	2013-09-22	Second Class	BW-11110	Bart Watters	Corporate	Newcastle	New South Wales	...	361.584	8	0.10	-16.1760	104.12	Critical	2013	9	20	2013-09
Fasteners	50601	MZ-2014-140	2014-06-28	2014-06-28	Same Day	JW-5220	Jane Waco	Corporate	Maputo	Cidade De Maputo	...	199.080	12	0.00	65.5200	87.09	Critical	2014	6	28	2014-06
Furnishings	38499	CA-2013-120369	2013-10-29	2013-10-29	Same Day	VB-21745	Victoria Brennan	Corporate	Rochester	New York	...	756.800	5	0.00	75.6800	206.61	High	2013	10	29	2013-10
Labels	38219	CA-2012-134257	2012-03-16	2012-03-19	Second Class	MS-17710	Maurice Satty	Consumer	Auburn	Alabama	...	491.550	5	0.00	240.8595	77.93	High	2012	3	16	2012-03
Machines	10648	ES-2012-5870268	2012-07-17	2012-07-19	First Class	BS-11365	Bill Shonely	Corporate	Saint-Brieuc	Brittany	...	2402.865	9	0.15	763.1550	699.55	Critical	2012	7	17	2012-07
Paper	36220	CA-2014-114055	2014-12-26	2014-12-30	Second Class	MH-18115	Mick Hernandez	Home Office	Huntsville	Alabama	...	629.100	6	0.00	301.9680	141.52	High	2014	12	26	2014-12
Phones	25330	IN-2013-71249	2013-10-17	2013-10-18	First Class	CR-12730	Craig Reiter	Consumer	Brisbane	Queensland	...	5175.171	9	0.10	919.9710	915.49	Medium	2013	10	17	2013-10
Storage	38362	CA-2011-106726	2011-12-06	2011-12-08	First Class	RS-19765	Roland Schwarz	Corporate	Los Angeles	California	...	1261.330	7	0.00	327.9458	506.49	Critical	2011	12	6	2011-12
Supplies	34577	CA-2011-102988	2011-04-05	2011-04-09	Second Class	GM-14695	Greg Maxwell	Corporate	Alexandria	Virginia	...	4164.050	5	0.00	83.2810	846.54	High	2011	4	5	2011-04
Tables	31192	IN-2012-86369	2012-04-14	2012-04-18	Standard Class	MB-18085	Mick Brown	Consumer	Hamilton	Waikato	...	5244.840	6	0.00	996.4800	878.38	High	2012	4	14	2012-04

17 rows × 26 columns

In [32]:
```
profit = sales_data.groupby("Sub-Category")['Profit']
profit.first()
```

Out[32]:
```
Sub-Category
Accessories     762.1845
Appliances     1036.0800
Art             153.8400
Binders        1906.4850
Bookcases      2097.0300
Chairs         -288.7650
Copiers         311.5200
Envelopes       -16.1760
Fasteners        65.5200
Furnishings      75.6800
Labels          240.8595
Machines        763.1550
Paper           301.9680
Phones          919.9710
Storage         327.9458
Supplies         83.2810
Tables          996.4800
Name: Profit, dtype: float64
```

In [33]:
```
sales = sales_data.groupby("Sub-Category")['Sales']
sales.first()
```

Out[33]:
```
Sub-Category
Accessories    2309.650
Appliances     3701.520
Art             439.680
```

```
Binders        5083.960
Bookcases      5667.870
Chairs         3709.395
Copiers        2832.960
Envelopes       361.584
Fasteners       199.080
Furnishings     756.800
Labels          491.550
Machines       2402.865
Paper           629.100
Phones         5175.171
Storage        1261.330
Supplies       4164.050
Tables         5244.840
Name: Sales, dtype: float64
```

In [34]:

```
profit_sales = sales_data.groupby("Sub-Category")['Profit','Sales']
profit_sales.first()
```

```
/tmp/ipykernel_34406/4112585796.py:1: FutureWarning: Indexing with multiple keys (implicitly converted to a tuple of keys) will be deprecated, use a list instead.
  profit_sales = sales_data.groupby("Sub-Category")['Profit','Sales']
```

Out[34]:

	Profit	Sales
Sub-Category		
Accessories	762.1845	2309.650
Appliances	1036.0800	3701.520
Art	153.8400	439.680
Binders	1906.4850	5083.960
Bookcases	2097.0300	5667.870
Chairs	-288.7650	3709.395
Copiers	311.5200	2832.960
Envelopes	-16.1760	361.584
Fasteners	65.5200	199.080
Furnishings	75.6800	756.800
Labels	240.8595	491.550
Machines	763.1550	2402.865
Paper	301.9680	629.100
Phones	919.9710	5175.171
Storage	327.9458	1261.330
Supplies	83.2810	4164.050
Tables	996.4800	5244.840

In [35]:

```
profit_sales = sales_data.groupby("Sub-Category")['Profit','Sales']
profit_sales.sum()
```

```
/tmp/ipykernel_34406/2529840627.py:1: FutureWarning: Indexing with multiple keys (implicitly converted to a tuple of keys) will be deprecated, use a list instead.
  profit_sales = sales_data.groupby("Sub-Category")['Profit','Sales']
```

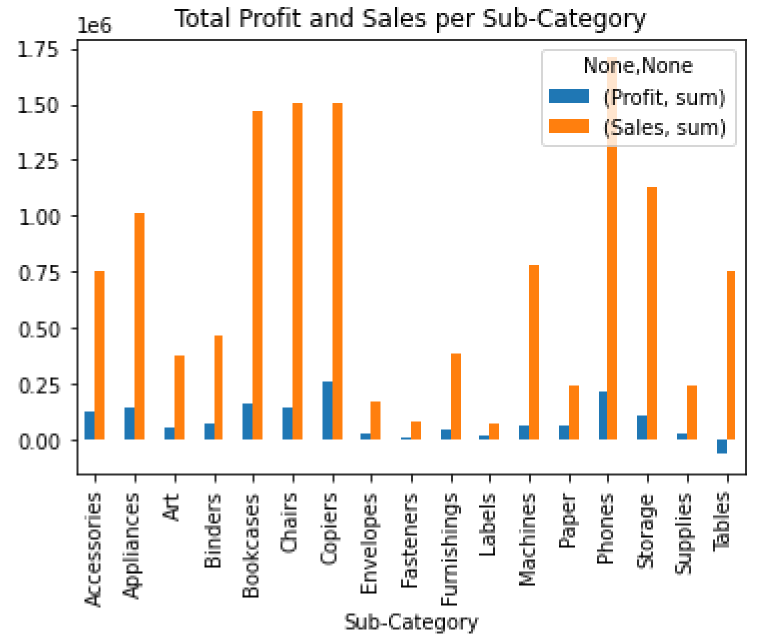
Out[35]:

	Profit	Sales
Sub-Category		

Sub-Category	Profit	Sales
Accessories	129626.30620	7.492370e+05
Appliances	141680.58940	1.011064e+06
Art	57953.91090	3.720920e+05
Binders	72449.84600	4.619115e+05
Bookcases	161924.41950	1.466572e+06
Chairs	140396.26750	1.501682e+06
Copiers	258567.54818	1.509436e+06
Envelopes	29601.11630	1.709043e+05
Fasteners	11525.42410	8.324232e+04
Furnishings	46967.42550	3.855783e+05
Labels	15010.51200	7.340403e+04
Machines	58867.87300	7.790601e+05
Paper	59207.68270	2.442917e+05
Phones	216717.00580	1.706824e+06
Storage	108461.48980	1.127086e+06
Supplies	22583.26310	2.430742e+05
Tables	-64083.38870	7.570419e+05

In [36]:
```
plt.figure(figsize=(10,15))
profit_sales.agg(['sum']).plot.bar()
plt.title('Total Profit and Sales per Sub-Category')
plt.show()
```

```
<Figure size 720x1080 with 0 Axes>
```



In [37]:
```
sales_data.columns
```

Out[37]:
```
Index(['Row ID', 'Order ID', 'Order Date', 'Ship Date', 'Ship Mode',
       'Customer ID', 'Customer Name', 'Segment', 'City', 'State', 'Country',
       'Market', 'Region', 'Product ID', 'Category', 'Sub-Category',
       'Product Name', 'Sales', 'Quantity', 'Discount', 'Profit',
       'Shipping Cost', 'Order Priority', 'Year', 'Month', 'Day',
       'Month_year'],
      dtype='object')
```

In [38]:

```
yearly_sale = sales_data.groupby('Year')['Sales']
yearly_sale.sum()
```

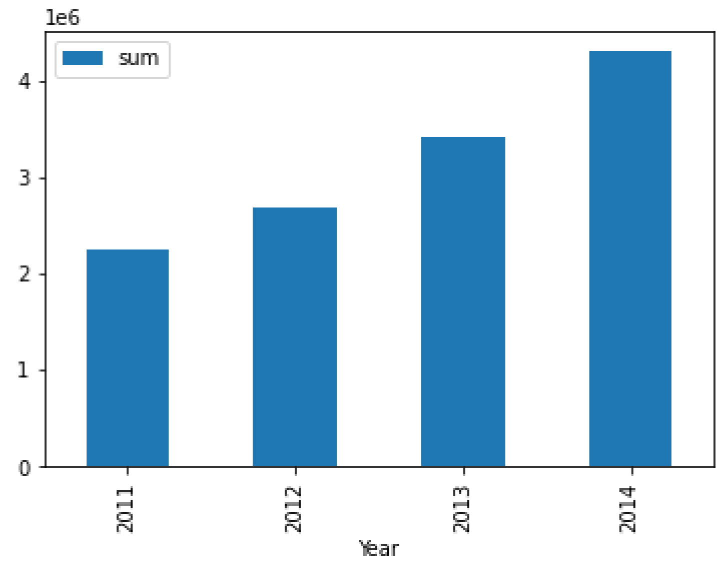
Out[38]:

```
Year
2011    2.259451e+06
2012    2.677439e+06
2013    3.405746e+06
2014    4.299866e+06
Name: Sales, dtype: float64
```

In [39]:

```
plt.figure(figsize=(12, 8))
yearly_sale.agg(['sum']).plot.bar()
plt.show()
```

```
<Figure size 864x576 with 0 Axes>
```



In [40]:

```
sales_data['Product Name'].nunique()
```

Out[40]:

```
3788
```

Analysis Based on Category

In [41]:

```
data = sales_data.copy()
```

Plotting sales vs profit graph for each subcategory

In [42]:

```
group_by_sub_category = data.groupby("Sub-Category")
```

In [43]:

```
profit = data.groupby("Sub-Category")['Profit']
profit.first()
```

```
Sub-Category
```

Out[43]:
```
Accessories     762.1845
Appliances     1036.0800
Art             153.8400
Binders        1906.4850
Bookcases      2097.0300
Chairs         -288.7650
Copiers         311.5200
Envelopes       -16.1760
Fasteners        65.5200
Furnishings      75.6800
Labels          240.8595
Machines        763.1550
Paper           301.9680
Phones          919.9710
Storage         327.9458
Supplies         83.2810
Tables          996.4800
Name: Profit, dtype: float64
```

In [44]:
```
sales = data.groupby("Sub-Category")['Sales']
sales.first()
```

Out[44]:
```
Sub-Category
Accessories    2309.650
Appliances     3701.520
Art             439.680
Binders        5083.960
Bookcases      5667.870
Chairs         3709.395
Copiers        2832.960
Envelopes       361.584
Fasteners       199.080
Furnishings     756.800
Labels          491.550
Machines       2402.865
Paper           629.100
Phones         5175.171
Storage        1261.330
Supplies       4164.050
Tables         5244.840
Name: Sales, dtype: float64
```

In [45]:
```
profit_sales = data.groupby("Sub-Category")['Profit','Sales']
profit_sales.first()
```

```
/tmp/ipykernel_34406/2932204674.py:1: FutureWarning: Indexing with multiple keys (implicitly converted to a tuple of keys) will be deprecated, use a list instead.
  profit_sales = data.groupby("Sub-Category")['Profit','Sales']
```

Out[45]:

	Profit	Sales
Sub-Category		
Accessories	762.1845	2309.650
Appliances	1036.0800	3701.520
Art	153.8400	439.680
Binders	1906.4850	5083.960
Bookcases	2097.0300	5667.870
Chairs	-288.7650	3709.395
Copiers	311.5200	2832.960
Envelopes	-16.1760	361.584

	Profit	Sales
Sub-Category		
Fasteners	65.5200	199.080
Furnishings	75.6800	756.800
Labels	240.8595	491.550
Machines	763.1550	2402.865
Paper	301.9680	629.100
Phones	919.9710	5175.171
Storage	327.9458	1261.330
Supplies	83.2810	4164.050
Tables	996.4800	5244.840

In [46]:

```
profit_sales = data.groupby("Sub-Category")['Profit','Sales']
profit_sales.sum()
```

```
/tmp/ipykernel_34406/806587871.py:1: FutureWarning: Indexing with multiple keys (implicitly converted to a tuple of keys) will be deprecated, use a list instead.
  profit_sales = data.groupby("Sub-Category")['Profit','Sales']
```

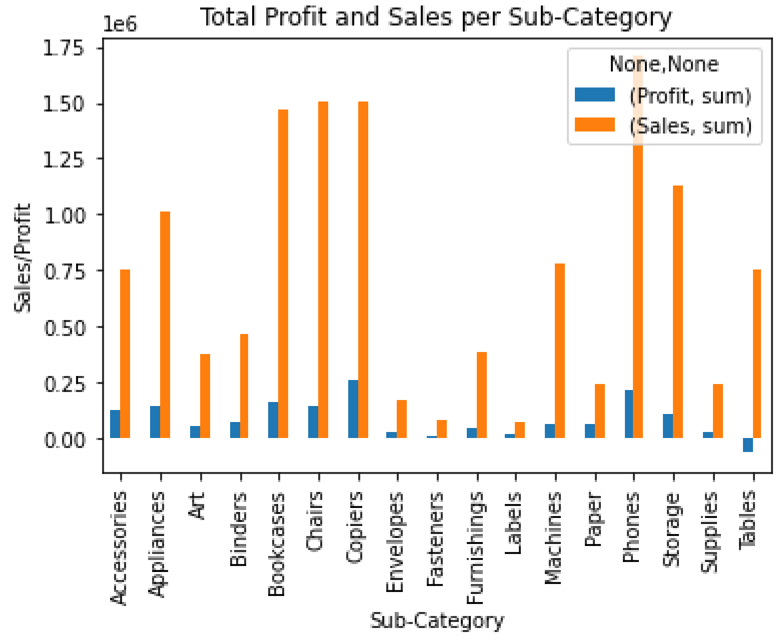
Out[46]:

	Profit	Sales
Sub-Category		
Accessories	129626.30620	7.492370e+05
Appliances	141680.58940	1.011064e+06
Art	57953.91090	3.720920e+05
Binders	72449.84600	4.619115e+05
Bookcases	161924.41950	1.466572e+06
Chairs	140396.26750	1.501682e+06
Copiers	258567.54818	1.509436e+06
Envelopes	29601.11630	1.709043e+05
Fasteners	11525.42410	8.324232e+04
Furnishings	46967.42550	3.855783e+05
Labels	15010.51200	7.340403e+04
Machines	58867.87300	7.790601e+05
Paper	59207.68270	2.442917e+05
Phones	216717.00580	1.706824e+06
Storage	108461.48980	1.127086e+06
Supplies	22583.26310	2.430742e+05
Tables	-64083.38870	7.570419e+05

In [47]:

```
plt.figure(figsize=(15,8))
profit_sales.agg(['sum']).plot.bar()
plt.title('Total Profit and Sales per Sub-Category')
plt.ylabel('Sales/Profit')
plt.show()
```

```
<Figure size 1080x576 with 0 Axes>
```



In [48]:
```python
profit_sales_year = data.groupby(["Sub-Category",'Year'])['Profit','Sales']
profit_sales_year.sum()
```

```
/tmp/ipykernel_34406/669520987.py:1: FutureWarning: Indexing with multiple keys (implicitly converted to a tuple of keys) will be deprecated, use a list instead.
  profit_sales_year = data.groupby(["Sub-Category",'Year'])['Profit','Sales']
```

Out[48]:

		Profit	Sales
Sub-Category	**Year**		
Accessories	**2011**	15719.8606	113456.0076
	2012	33507.1002	172397.6850
	2013	38805.4168	209895.1623
	2014	41593.9286	253488.1636
Appliances	**2011**	22838.4413	173383.4264
...	**...**	...	...
Supplies	**2014**	7365.4090	86283.0918
Tables	**2011**	-11075.2945	147131.4937
	2012	-8421.6986	164086.4968
	2013	-14040.4872	202363.5351
	2014	-30545.9084	243460.3988

68 rows × 2 columns

Which sub-category is most popular in which city?

In [49]:
```python
popular_subcategory = data[['Sub-Category', 'City', 'Quantity']].groupby(['Sub-Category', 'City', 'Quantity']).sum()
popular_subcategory
```

Out[49]:

Sub-Category	City	Quantity
Accessories	**Abadan**	**2**

Sub-Category	City	Quantity
	Abidjan	1
		2
	Abu Kabir	1
	Accra	2
...	...	...
Tables	Yuci	6
	Yulin	1
	Yunyang	2
	Zapopan	3
	Zaria	8

36481 rows × 0 columns

In [50]:
```
data.groupby(["Sub-Category","City"])['City'].count().sort_values().groupby(level=0).tail(1)
```

Out[50]:
```
Sub-Category  City
Machines      Manila             20
Envelopes     Tegucigalpa        26
Copiers       Manila             27
Bookcases     Managua            31
Tables        Los Angeles        32
Supplies      Santo Domingo      32
Fasteners     Santo Domingo      35
Labels        New York City      36
Appliances    Los Angeles        37
Chairs        New York City      62
Accessories   New York City      64
Art           New York City      70
Furnishings   New York City      78
Storage       New York City      82
Phones        New York City      89
Paper         New York City     124
Binders       New York City     145
Name: City, dtype: int64
```

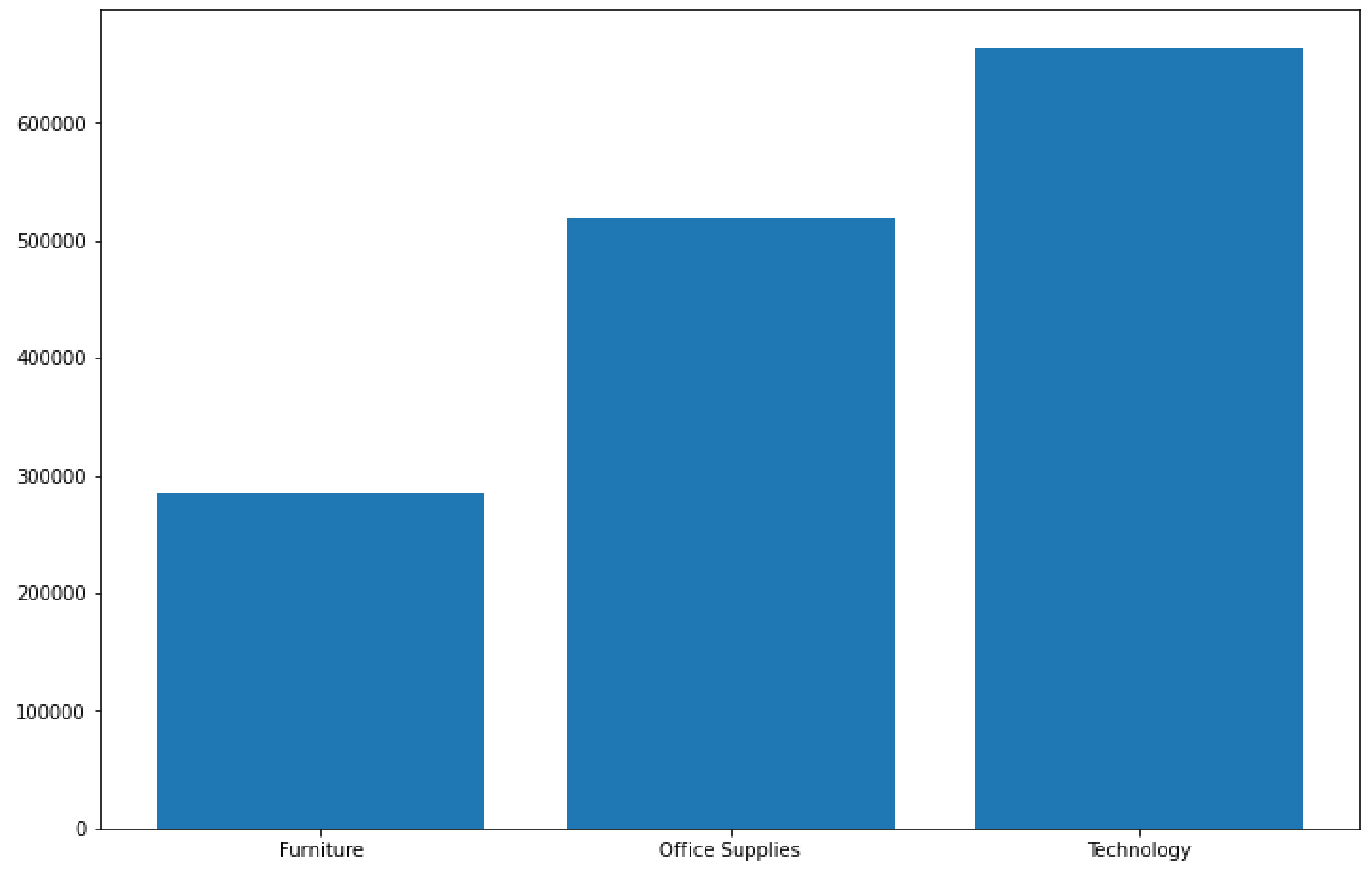
Which Category and Sub-Category gives us the highest profit?

In [51]:
```
highest_category_profit = data.groupby(by=['Category']).sum().sort_values(by=['Profit']).reset_index()
highest_category_profit
```

Out[51]:

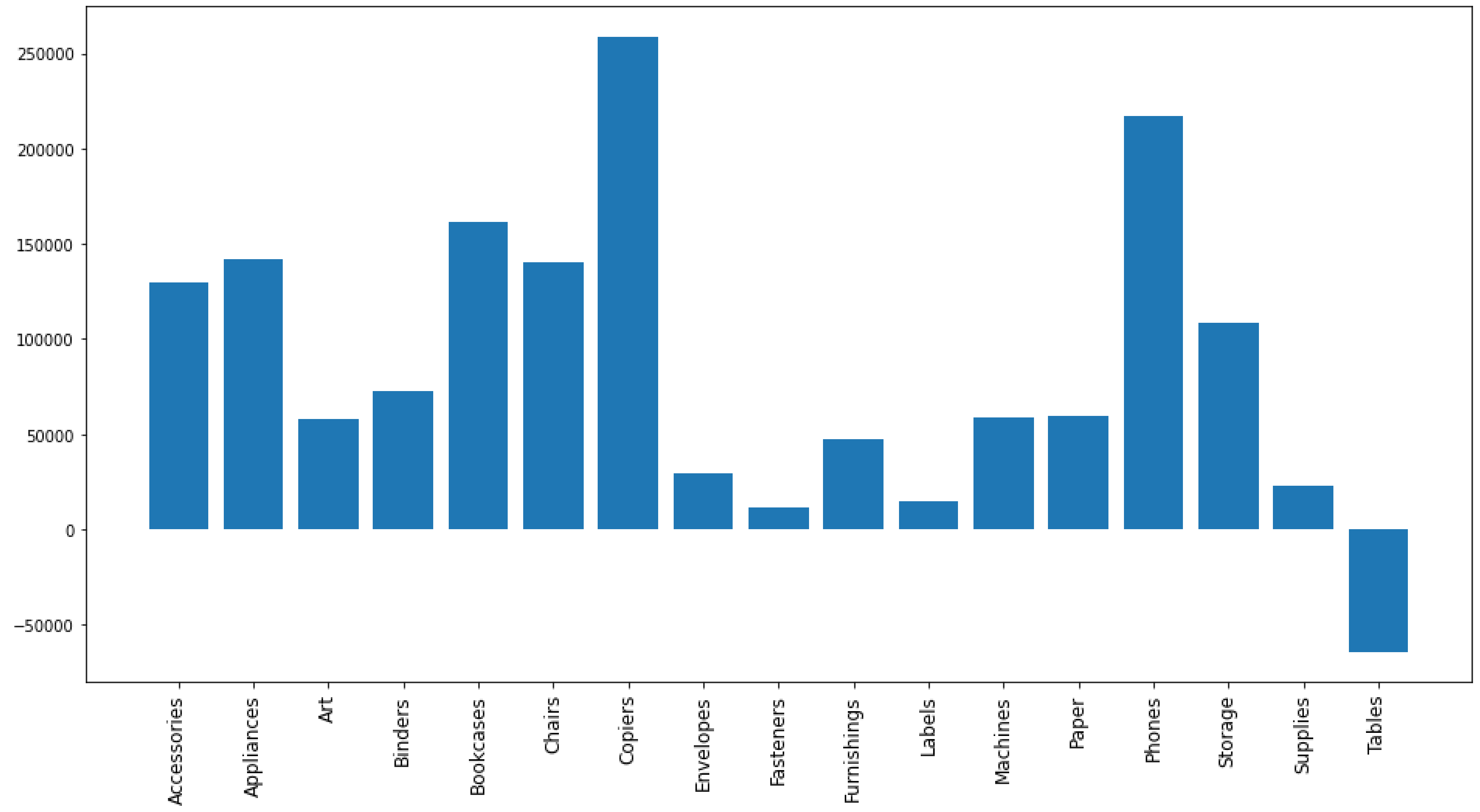
	Category	Row ID	Sales	Quantity	Discount	Profit	Shipping Cost	Year	Month	Day
0	Furniture	241487920	4.110874e+06	34954	1660.030	285204.72380	440319.4790	19878166	74968	156524
1	Office Supplies	815386724	3.787070e+06	108182	4297.190	518473.83430	405448.3450	62945679	234497	494357
2	Technology	258483051	4.744557e+06	35176	1372.508	663778.73318	507047.8794	20411498	76371	158627

In [52]:
```
plt.figure(figsize=(12,8))
plt.bar("Category",'Profit',data=highest_category_profit)
plt.show()
```



In [53]:

```
highest_subcategory_profit = data[['Sub-Category', 'Profit']].groupby(['Sub-Category']).sum().reset_index()
plt.figure(figsize=(16, 8))
plt.bar('Sub-Category', 'Profit', data = highest_subcategory_profit)
plt.xticks(rotation='vertical', size=12)
plt.show()
```



From the above bar-graph, we can conclude that Sub-Category 'Copiers' gives us the highest profit.

Category and sub-category distribution

In [54]:
```
sub_category_sale_distribution_accross_year = pd.pivot_table(data = data, values = "Sales", index = ["Sub-Category", "Year"], columns = "Month")
sub_category_sale_distribution_accross_year.style.background_gradient(cmap = 'Reds')
```

Out[54]:

	Month	1	2	3	4	5	6	7	8	9	10	11	12
Sub-Category	**Year**												
Accessories	**2011**	193.204753	184.668425	259.785500	190.718438	272.315538	230.358972	205.468708	185.208953	200.038965	197.100274	192.933962	224.182354
	2012	206.094553	263.553937	241.443861	177.938481	212.942012	288.582870	387.442078	247.253904	264.174436	243.101303	225.011672	339.982078
	2013	205.369447	261.758703	260.716292	187.400128	204.739868	258.522027	214.797553	254.615994	292.475843	332.882788	262.809181	212.799050
	2014	207.862269	310.371459	166.594337	258.008322	231.190433	261.452029	282.469485	303.821281	227.957437	232.332495	256.827383	207.059123
Appliances	**2011**	480.869667	313.689133	560.470580	639.233495	551.023577	545.715661	488.965979	435.388852	617.115492	457.041367	721.841381	363.314021
	2012	812.205015	276.204750	419.907087	344.182769	507.948645	584.952000	617.819947	799.164305	543.569412	375.692021	610.323600	766.148624
	2013	995.744700	910.422812	532.612260	553.667476	429.291005	700.962859	609.616370	415.982026	391.304868	684.963280	499.495713	478.588202
	2014	819.292765	649.152370	496.124014	632.230906	565.063477	465.459384	644.059667	803.506354	484.471231	601.407338	620.911439	616.292748
Art	**2011**	91.400067	70.123554	49.517339	79.150576	88.353839	71.585414	68.732483	80.115164	80.128810	85.053897	52.610273	90.266856
	2012	75.458462	70.324450	68.000762	69.790413	91.832771	68.732451	81.240061	82.252630	72.993751	81.075196	75.694164	84.217237
	2013	79.580353	82.377509	64.456728	60.164407	81.706466	83.442342	84.582159	82.698167	76.108095	68.781738	75.579182	72.134862

	Month	1	2	3	4	5	6	7	8	9	10	11	12
Sub-Category	**Year**												
	2014	75.499031	77.299203	72.215439	77.162580	66.018439	80.270931	71.783343	86.630860	82.032034	70.028774	76.740237	66.384153
Binders	**2011**	50.459980	48.600402	68.528081	50.502712	111.116860	79.913273	79.698169	79.272894	146.798407	40.117105	88.754483	82.001898
	2012	51.920270	94.536638	160.960930	98.559518	46.939390	79.683394	58.995306	69.552163	58.307623	59.527571	59.300316	79.652740
	2013	65.076641	62.620640	64.944823	67.544357	59.336796	55.584989	66.456205	51.552408	80.064668	86.911196	60.672947	130.865479
	2014	118.250040	54.942455	60.026190	80.507359	62.873727	56.993254	38.697364	120.198341	80.681449	91.029999	68.768907	59.650113
Bookcases	**2011**	582.695783	919.737385	603.725947	400.331750	554.322748	614.556430	677.190000	651.844355	708.108073	654.013000	756.838145	566.835921
	2012	755.355459	858.686643	687.131777	622.117904	665.406384	541.294509	490.035615	580.340382	603.437198	610.982402	625.004626	512.099352
	2013	768.013357	605.698011	572.386313	592.677161	493.477814	492.604482	564.763429	752.725356	551.032019	551.046048	598.248520	573.673894
	2014	558.964907	703.925984	611.048913	525.011408	467.332244	685.664239	696.529728	592.124821	782.552175	560.777862	537.879323	605.900846
Chairs	**2011**	470.414146	497.369643	539.356220	353.065851	431.651179	425.579645	446.254923	545.568018	483.090485	538.841064	410.731742	514.224071
	2012	378.148110	465.932248	334.670874	355.838973	498.116968	343.390647	309.548348	552.884958	432.211311	479.727973	403.991824	492.753506
	2013	482.485779	431.643188	503.769379	486.820393	463.697804	483.278929	462.174522	425.124027	386.958574	379.735613	417.762550	428.054816
	2014	512.453200	430.499611	457.303682	288.536393	422.458309	402.523478	360.506347	464.356257	420.877480	389.093149	514.716649	421.223492
Copiers	**2011**	527.390221	550.568838	504.043366	404.819506	595.390608	563.283343	533.923436	665.062740	669.549906	687.635138	660.834724	678.410704
	2012	592.773514	503.408927	696.823941	671.108180	883.506090	682.933353	522.790585	707.805085	650.816531	637.934237	502.868333	655.011607
	2013	616.086099	817.084931	548.308729	528.992657	855.557287	681.312486	690.066402	626.145342	708.254585	1109.095475	665.253854	678.013681
	2014	842.735882	658.444193	1073.877360	642.708505	657.135982	566.008383	561.147534	584.176858	658.022130	890.645485	727.664805	649.328810
Envelopes	**2011**	75.540836	63.085773	67.369568	78.429720	72.851114	71.454378	84.621645	77.066848	85.476053	62.996296	72.291382	60.891579
	2012	80.720947	68.779708	44.237674	93.153975	70.954854	67.380688	71.619479	58.199039	70.242403	74.692831	62.512778	78.031535
	2013	64.395408	69.464261	70.801545	53.460544	72.900228	68.529595	87.015088	82.446260	86.255847	65.852550	65.177112	74.946873
	2014	68.239478	52.667788	76.026787	62.365167	67.542031	64.888324	67.799806	66.474225	57.283029	81.704486	68.439321	64.378593
Fasteners	**2011**	29.339195	51.700000	45.048480	31.159100	41.047571	38.984030	33.689525	31.482895	32.335027	34.670108	24.513607	30.667147
	2012	31.960354	43.205095	40.295200	30.884265	36.795907	37.980935	30.688387	42.541653	34.598260	38.849472	34.446979	33.017974
	2013	27.457468	28.323704	28.100006	36.611369	29.463333	29.108369	25.603100	38.678570	32.322258	36.229553	32.714101	39.746662
	2014	28.958691	30.423445	33.263393	35.032300	31.919819	43.960995	28.919671	35.208342	29.679789	38.751212	34.590793	34.658745
Furnishings	**2011**	96.603440	145.049037	109.722559	160.382149	93.584621	89.684561	83.422735	92.572504	128.933558	134.234763	117.153742	85.330184
	2012	132.886477	144.956632	126.626181	142.225111	112.250041	153.596373	138.488770	109.047047	120.772610	88.452212	123.783958	123.877865
	2013	131.239108	122.791870	123.892604	99.874787	110.216831	131.476930	124.069612	128.666209	135.470215	118.716988	143.753863	141.308088
	2014	146.930219	134.962337	123.779592	110.807696	99.043051	109.406025	125.331582	131.437707	116.839046	124.471819	129.731080	112.831990
Labels	**2011**	25.837060	20.854133	23.924483	23.096829	25.535871	37.378976	36.113374	22.756308	30.670052	26.348131	29.303960	34.225272
	2012	25.682391	28.074913	43.028250	25.724662	28.734622	19.459779	29.626442	32.433194	23.661961	34.806047	30.256979	24.475593
	2013	27.116961	26.030192	25.159162	23.450731	26.589898	27.569892	26.236170	27.292599	25.475494	25.436147	33.812773	25.242397
	2014	28.854772	25.301519	24.498992	24.471533	26.619791	30.234091	39.470961	33.920387	28.895642	25.615108	27.388036	26.622750
Machines	**2011**	507.334464	331.129314	1443.688236	417.032800	335.917950	298.783290	543.944550	349.649339	1097.390497	469.094537	453.639755	543.068988
	2012	627.318789	332.025771	416.250660	400.381921	565.412308	399.876828	678.450900	489.643603	427.029326	479.668173	553.764091	577.333878
	2013	300.626233	981.691857	694.664955	1079.764469	809.214985	566.683493	346.564915	400.475909	393.328510	531.654011	529.045571	371.392887
	2014	545.184448	366.913333	305.035986	620.414735	414.804759	536.998296	604.541865	509.323025	431.608935	595.932073	636.343392	368.651160
Paper	**2011**	62.901141	63.962773	60.724929	70.138514	54.714140	82.358990	61.983426	91.114067	49.869576	64.621273	79.924252	63.010615

	Month	1	2	3	4	5	6	7	8	9	10	11	12
Sub-Category	Year												
	2012	77.002735	52.343774	72.759803	46.222194	85.010800	53.957408	64.854506	72.356151	66.126882	67.902416	79.058092	71.381866
	2013	86.380002	94.838614	68.617442	59.272443	84.381440	77.136751	75.689371	67.172887	58.093245	67.027590	66.036844	67.404326
	2014	67.460573	61.421774	74.605052	67.331426	69.170333	70.217085	57.671591	79.991252	64.664053	62.047768	65.704736	70.308045
Phones	2011	608.169016	570.213913	428.302819	516.451310	496.494813	463.949278	414.053495	686.716927	493.413316	629.144683	669.228163	504.477045
	2012	332.238933	547.026517	328.612904	418.937206	534.703974	469.725114	505.512333	781.081741	536.823014	624.133151	372.195243	525.198187
	2013	507.417782	362.318229	518.399470	400.310417	551.545563	636.602677	512.030196	524.388851	468.245878	531.161703	451.909511	519.376952
	2014	476.668674	421.636575	408.733715	454.141369	442.191111	467.549257	529.468393	627.767362	434.979449	454.701667	509.629645	553.047743
Storage	2011	263.020719	227.963118	204.818966	219.620593	292.633757	166.568600	214.628142	175.812694	242.284392	200.311677	255.617697	252.850665
	2012	270.270695	246.800082	232.917579	219.988328	187.745682	186.200850	223.697984	258.135054	194.238758	177.488202	201.497215	193.088945
	2013	267.911900	269.448734	231.843764	206.229372	227.771961	216.586905	242.763144	213.065100	219.701180	222.593735	248.459984	252.057859
	2014	227.648621	211.974627	208.846285	254.232201	253.757091	222.720786	217.226461	212.097316	209.813523	217.109998	242.278730	175.122500
Supplies	2011	80.867500	85.777526	73.784139	324.182529	70.932152	94.372189	468.720324	63.982916	100.705352	83.754877	84.981887	86.436788
	2012	78.916133	68.214122	79.874863	80.422811	79.032792	88.825205	59.440621	87.941067	86.845461	85.883222	81.115124	106.245905
	2013	86.041848	106.508444	308.932151	95.227109	84.951233	95.902639	98.633386	88.546895	68.887394	93.028236	82.380728	104.369464
	2014	176.869587	83.233005	128.272328	101.537545	84.402063	75.550379	84.911455	90.785081	127.524979	91.527421	76.963445	122.769396
Tables	2011	919.590563	617.647286	889.392222	477.948250	785.439893	890.627650	655.933091	771.841500	891.193833	818.273611	840.323125	1231.317120
	2012	1241.370429	1574.104800	1347.544917	1183.319187	770.160333	1003.779061	588.146833	846.856857	905.748063	1117.802238	720.555777	1077.400594
	2013	1295.021241	1562.756583	476.241400	511.319967	908.143450	808.133261	767.078400	848.691673	733.567750	536.534750	1027.539842	876.381584
	2014	806.547719	464.065050	954.842353	1038.213285	1096.898532	954.055932	548.114738	1136.487930	796.677009	618.439659	816.051472	881.534700