Hello, welcome to another lesson where you'll learn more about the vast range of tools available in Python, Numpy Pandas, Matplotlib e.t.c for use in Data Science Toolbox.

This is me sharing my journey from scratch . And hoping this will encourage you to keep doing it, even if it means doing it poorly, till you get better and gain mastery.

I'm just so excited about my discovery and the vast possibilities on this career path using Python and its libraries.

I'm sure if you follow the lessons and the previous ones, the vibes will definitely spread to you.

Let's Get Started !!!

The first step is to import the required libraries that you'll be working with which includes Numpy, Pandas and Matplotlib. Remember to include "%matplotlib inline" so that your plots can be displayed immediately.

The dataset you'll be dealing with is a Sample Sales Data which can be found online. import the data and read it

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
```

In [2]:
```
dataset = pd.read_excel("Sample-sales-data-excel.xlsx")
dataset.tail(10)
```

Out[2]:

	Row ID	Order ID	Order Date	Ship Date	Ship Mode	Customer ID	Customer Name	Segment	Country	
9984	9985	CA-2015-100251	2015-05-17	2015-05-23	Standard Class	DV-13465	Dianna Vittorini	Consumer	United States	Long B
9985	9986	CA-2015-100251	2015-05-17	2015-05-23	Standard Class	DV-13465	Dianna Vittorini	Consumer	United States	Long B
9986	9987	CA-2016-125794	2016-09-29	2016-10-03	Standard Class	ML-17410	Maris LaWare	Consumer	United States	Los An
9987	9988	CA-2017-163629	2017-11-17	2017-11-21	Standard Class	RA-19885	Ruben Ausman	Corporate	United States	At
9988	9989	CA-2017-163629	2017-11-17	2017-11-21	Standard Class	RA-19885	Ruben Ausman	Corporate	United States	At
9989	9990	CA-2014-110422	2014-01-21	2014-01-23	Second Class	TB-21400	Tom Boeckenhauer	Consumer	United States	M
9990	9991	CA-2017-121258	2017-02-26	2017-03-03	Standard Class	DB-13060	Dave Brooks	Consumer	United States	Costa I
9991	9992	CA-2017-121258	2017-02-26	2017-03-03	Standard Class	DB-13060	Dave Brooks	Consumer	United States	Costa I
9992	9993	CA-2017-121258	2017-02-26	2017-03-03	Standard Class	DB-13060	Dave Brooks	Consumer	United States	Costa I
9993	9994	CA-2017-119914	2017-05-04	2017-05-09	Second Class	CC-12220	Chris Cortes	Consumer	United States	Westmi

10 rows × 21 columns

By default, Pandas assigns a numerical index to the dataframe. And the dataset has a column named "ROW ID".

Intuitively, that should be the row index so i made that the row index and deleted the row.

Note that the same task can be achieved using "set_index()"

In [3]:
```
dataset.index = dataset["Row ID"]
dataset.drop(['Row ID'], axis = "columns", inplace = True)
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 9994 entries, 1 to 9994
Data columns (total 20 columns):
Order ID         9994 non-null object
Order Date       9994 non-null datetime64[ns]
Ship Date        9994 non-null datetime64[ns]
Ship Mode        9994 non-null object
Customer ID      9994 non-null object
Customer Name    9994 non-null object
Segment          9994 non-null object
Country          9994 non-null object
City             9994 non-null object
State            9994 non-null object
Postal Code      9994 non-null int64
Region           9994 non-null object
Product ID       9994 non-null object
Category         9994 non-null object
Sub-Category     9994 non-null object
Product Name     9994 non-null object
Sales            9994 non-null float64
Quantity         9994 non-null int64
Discount         9994 non-null float64
Profit           9994 non-null float64
dtypes: datetime64[ns](2), float64(3), int64(2), object(13)
memory usage: 1.1+ MB
```

After the data has been loaded and read, you should familiarize yourself with various attributes of the dataset

In [4]:
```
dataset.describe()
```

Out[4]:

	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

In [5]:
```
dataset.shape
```

Out[5]:
```
(9994, 20)
```

In [6]:
```
dataset.size
```

Out[6]:
```
199880
```

In [7]:
```
dataset.ndim
```

Out[7]:
```
2
```

In [8]:
```
dataset.dtypes
```

Out[8]:
```
Order ID                  object
Order Date        datetime64[ns]
Ship Date         datetime64[ns]
Ship Mode                 object
Customer ID               object
Customer Name             object
Segment                   object
Country                   object
City                      object
State                     object
Postal Code                int64
Region                    object
Product ID                object
Category                  object
Sub-Category              object
Product Name              object
Sales                    float64
Quantity                   int64
Discount                 float64
Profit                   float64
dtype: object
```

To Check If there's duplicate rows or information contained in the dataframe

In [9]:
```
dataset.duplicated().sum()
```

Out[9]: 1

Now that it's confirmed that there's duplicate rows in the dataframe and the sum() aggregate function reveals its count to be 1, there's need to call out the exact duplicate(s)

In [10]:
```
dataset[dataset.duplicated()]
```

Out[10]:

Row ID	Order ID	Order Date	Ship Date	Ship Mode	Customer ID	Customer Name	Segment	Country	City	State	I
3407	US-2014-150119	2014-04-23	2014-04-27	Standard Class	LB-16795	Laurel Beltran	Home Office	United States	Columbus	Ohio	

To further narrow the search down, make the work easier and simplify the task, I used dot index and dot info respectively on the code that revealed the duplicates. This will reveal the info as was revealed by the code. It might not be much but it's to be double sure of the results. And the index of the duplicate row will be known

In [11]:
```
dataset[dataset.duplicated()].index
```

Out[11]:
```
Int64Index([3407], dtype='int64', name='Row ID')
```

In [12]:
```
dataset[dataset.duplicated()]. info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1 entries, 3407 to 3407
Data columns (total 20 columns):
Order ID         1 non-null object
Order Date       1 non-null datetime64[ns]
Ship Date        1 non-null datetime64[ns]
Ship Mode        1 non-null object
Customer ID      1 non-null object
Customer Name    1 non-null object
Segment          1 non-null object
Country          1 non-null object
City             1 non-null object
State            1 non-null object
Postal Code      1 non-null int64
Region           1 non-null object
Product ID       1 non-null object
Category         1 non-null object
Sub-Category     1 non-null object
Product Name     1 non-null object
Sales            1 non-null float64
Quantity         1 non-null int64
Discount         1 non-null float64
Profit           1 non-null float64
dtypes: datetime64[ns](2), float64(3), int64(2), object(13)
memory usage: 116.0+ bytes
```

Then, i thought that if the number revealed as a duplicate row index was correct, the row index from which it was duplicated from will not be far away from that index number. So, i called the index number directly above it alongside the duplicate row index, to compare the information and confirm the duplicate value. The result proved to be true as it revealed my exact assumption

In [13]:
```
dataset.loc[[3406,3407]]
```

Out[13]:

Row ID	Order ID	Order Date	Ship Date	Ship Mode	Customer ID	Customer Name	Segment	Country	City	State	I
3406	US-2014-150119	2014-04-23	2014-04-27	Standard Class	LB-16795	Laurel Beltran	Home Office	United States	Columbus	Ohio	4
3407	US-2014-150119	2014-04-23	2014-04-27	Standard Class	LB-16795	Laurel Beltran	Home Office	United States	Columbus	Ohio	4

Now, i can delete whichever of the duplicated values as i deem fit using the drop_duplicates function.

Then, I'll call the duplicated function to confirm if there's still duplicate values in the dataframe

In [14]:
```
dataset.drop_duplicates(inplace = True)
dataset.duplicated().sum()
```

Out[14]:
```
0
```

Note that the procedure can also be achieved using the drop function and setting inplace to be True. I run the code below and it worked effectively

To check for NA values, i use the isna()

In [15]:
```
dataset.isna().sum()
```

Out[15]:
```
Order ID         0
Order Date       0
Ship Date        0
Ship Mode        0
Customer ID      0
Customer Name    0
Segment          0
Country          0
City             0
State            0
Postal Code      0
Region           0
Product ID       0
Category         0
Sub-Category     0
Product Name     0
Sales            0
Quantity         0
Discount         0
Profit           0
dtype: int64
```

In [16]:
```
dataset.notna().sum()
```

Out[16]:
```
Order ID         9993
Order Date       9993
Ship Date        9993
Ship Mode        9993
Customer ID      9993
Customer Name    9993
Segment          9993
Country          9993
City             9993
State            9993
Postal Code      9993
Region           9993
Product ID       9993
Category         9993
Sub-Category     9993
Product Name     9993
Sales            9993
Quantity         9993
Discount         9993
Profit           9993
dtype: int64
```

There are no null values in the dataset.

With my level present level of the dataset, I'll check for the relationship between some of the columns of this dataset, do some grouping on it and start the analysis and visualization steps

In [17]:
```
dataset.index
```

Out[17]:
```
Int64Index([   1,    2,    3,    4,    5,    6,    7,    8,    9,   10,
                ...
            9985, 9986, 9987, 9988, 9989, 9990, 9991, 9992, 9993, 9994],
           dtype='int64', name='Row ID', length=9993)
```

On Checking the index values, i observed it's been altered as a result of the duplicate value(s) dropped.

I'll rename the columns to be in an orderly manner and I'll check again

In [18]:
```
dataset.index = range(1, 9994)
dataset.index
```

Out[18]:
```
RangeIndex(start=1, stop=9994, step=1)
```

The dataset is now ready for further analysis and exploration.

I rewrite the the completed dataset back to an excel file. Note that this is optional, in case you need the file for use somewhere else

dataset.to_excel("SSDE.xlsx", index = False)

In [19]:
```
dataset.head()
```

Out[19]:

	Order ID	Order Date	Ship Date	Ship Mode	Customer ID	Customer Name	Segment	Country	City	State
1	CA-2016-152156	2016-11-08	2016-11-11	Second Class	CG-12520	Claire Gute	Consumer	United States	Henderson	Kentucky
2	CA-2016-152156	2016-11-08	2016-11-11	Second Class	CG-12520	Claire Gute	Consumer	United States	Henderson	Kentucky
3	CA-2016-138688	2016-06-12	2016-06-16	Second Class	DV-13045	Darrin Van Huff	Corporate	United States	Los Angeles	California
4	US-2015-108966	2015-10-11	2015-10-18	Standard Class	SO-20335	Sean O'Donnell	Consumer	United States	Fort Lauderdale	Florida
5	US-2015-108966	2015-10-11	2015-10-18	Standard Class	SO-20335	Sean O'Donnell	Consumer	United States	Fort Lauderdale	Florida

It's time to start exploring the dataset, deriving insights and plotting visualization.

What you derive from the datqset depends on your wnd goal, but for learning purpose, let's obtain basic information and plot the visualization where necessary

Using the unique() on Segment Column revealed that there 3 distinct segment of buyers

In [20]:
```
dataset["Segment"].unique()
```

Out[20]:
```
array(['Consumer', 'Corporate', 'Home Office'], dtype=object)
```

Using the unique() on Country Column revealed that there's only one distinct country where sales was made.

There are 4 distinct Regions where sales was made

There are 49 distinct states where sales was made

There are 531 distinct cities where sales was made

In [21]:
```
dataset["Country"].unique()
```

Out[21]:
```
array(['United States'], dtype=object)
```

In [22]:
```
dataset["Region"].unique()
```

Out[22]:
```
array(['South', 'West', 'Central', 'East'], dtype=object)
```

In [23]:
```
pd.value_counts(dataset["State"].unique()).sum()
```

Out[23]:
```
49
```

In [24]:
```
pd.value_counts(dataset["City"].unique()).sum()
```

Out[24]:
```
531
```

The Company has 3 distinct categories of products and

17 distinct sub categories of them

In [25]:
```
dataset["Category"].unique()
```

Out[25]:
```
array(['Furniture', 'Office Supplies', 'Technology'], dtype=object)
```

In [26]:
```
pd.value_counts(dataset["Sub-Category"].unique()).sum()
```

Out[26]:
```
17
```

The company uses 4 distinct means of shipping her products to her customers

In [27]:
```
pd.value_counts(dataset["Ship Mode"].unique()).sum()
```

Out[27]:
```
4
```

And there are 793 unique customers who made purchase

In [28]:
```
pd.value_counts(dataset["Customer Name"].unique()).count()
```

Out[28]:
```
793
```

First, let me show all the data we've derived so far in visualization

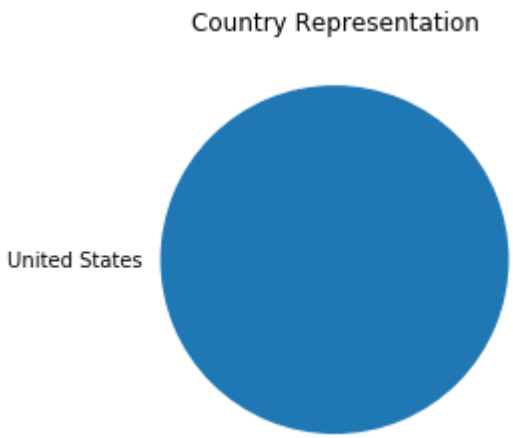
To show how many countries the company covered in sales

In [29]:
```
cov_country = dataset["Sales"].groupby(dataset["Country"])
cov_country1 = cov_country.count()
cov_country1
```

Out[29]:
```
Country
United States    9993
Name: Sales, dtype: int64
```

In [30]:
```
label = cov_country1.index
plt.pie(cov_country.unique().count(), labels = label)
plt.ylabel("")
plt.title("Country Representation")
plt.savefig("S-Dset.png")
```

```
/data/user/0/ru.iiec.pydroid3/files/arm-linux-androideabi/lib/python3.
7/site-packages/ipykernel_launcher.py:2: MatplotlibDeprecationWarning:
Non-1D inputs to pie() are currently squeeze()d, but this behavior is
deprecated since 3.1 and will be removed in 3.3; pass a 1D array inste
ad.
```

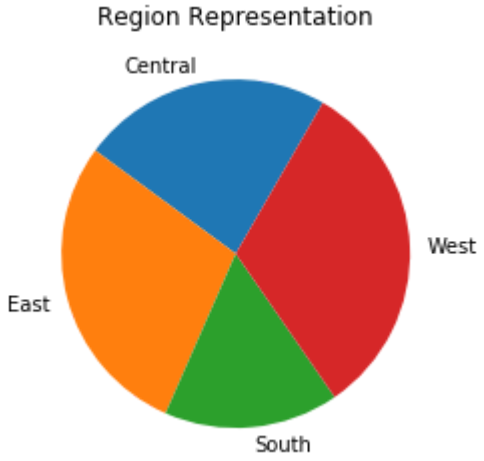


Now, let me show how many regions are covered in the sales activities

In [31]:
```
cov_region = dataset["Sales"].groupby(dataset["Region"])
cov_region1 = cov_region.count()
cov_region1
```

Out[31]:
```
Region
Central    2323
East       2847
South      1620
West       3203
Name: Sales, dtype: int64
```

In [32]:
```
plt.pie(cov_region1, labels = cov_region1.index,startangle = 60)
plt.title("Region Representation")
plt.savefig("S-Dset1.png")
```



Now, let me show the states that are covered in the company's sales activities

In [33]:
```
cov_states = pd.value_counts(dataset.State.unique()).sum()
cov_states
```

Out[33]:
```
49
```

In [34]:
```
print(" The number of states covered in the sales data is",cov_states)
```

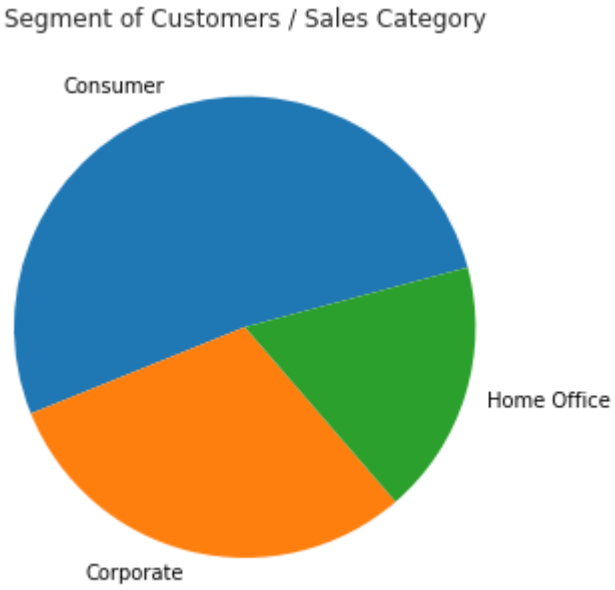
```
 The number of states covered in the sales data is 49
```

Now, let me create a representation of the segment of customers who patronize the company or where sales are made to

In [35]:
```
cov_segment = dataset["Sales"].groupby(dataset["Segment"])
cov_segment1 = cov_segment.count()
cov_segment1
```

Out[35]:
```
Segment
Consumer        5191
Corporate       3020
Home Office     1782
Name: Sales, dtype: int64
```

In [36]:
```
fig = plt.figure()
axes = fig.add_axes([0,0,1,1])
axes.pie(cov_segment1, labels = cov_segment1.index,startangle = 15)
axes.set_title("Segment of Customers / Sales Category", alpha = 0.85)
fig.savefig("S-Dset3.png")
```



Now, let me represent the cities covered in sales by this company

In [37]:
```
cov_cities = pd.value_counts(dataset.City.unique()).sum()
cov_cities
```

Out[37]:
```
531
```

In [38]:
```
print("The number of Cities covered in the sales activities is", cov_c
ities)
```

```
The number of Cities covered in the sales activities is 531
```

Now, let me create a visualization of the shipping modes used by the company to deliver products to her customers

In [39]:
```
ship_mode = pd.value_counts(dataset["Ship Mode"].unique()).sum()
ship_mode
```

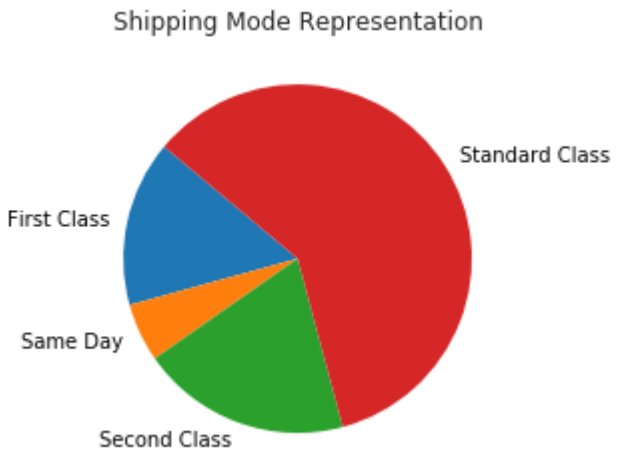
Out[39]:
```
4
```

In [40]:
```
shipping_mode = dataset.Sales.groupby(dataset["Ship Mode"])
shipping_mode1 = shipping_mode.count()
shipping_mode1
```

Out[40]:
```
Ship Mode
First Class       1538
Same Day           543
Second Class      1945
Standard Class    5967
Name: Sales, dtype: int64
```

In [41]:
```
plt.pie(shipping_mode1, startangle = 140, labels = shipping_mode1.inde
x)
plt.title("Shipping Mode Representation", alpha = 0.85)
```

Out[41]:
```
Text(0.5, 1.0, 'Shipping Mode Representation')
```



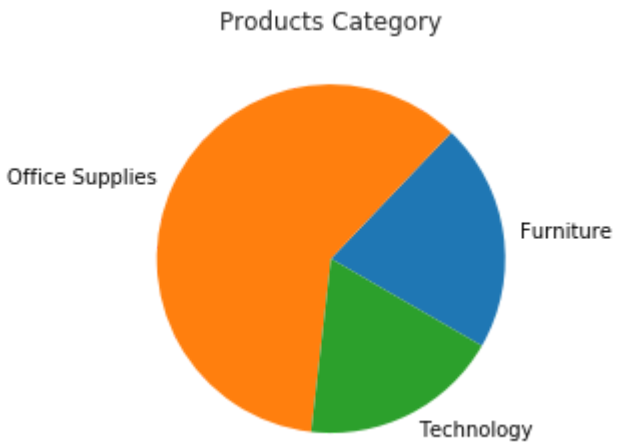
Now, I'm going to create a visualization of the company's product categories as well as product sub-categories

In [42]:
```
products_cat = dataset["Sales"].groupby(dataset["Category"])
products_cat1 = products_cat.count()
products_cat1
```

Out[42]:
```
Category
Furniture          2120
Office Supplies    6026
Technology         1847
Name: Sales, dtype: int64
```

In [43]:
```
plt.pie(products_cat1, labels = products_cat1.index, startangle = 330)
plt.title("Products Category", alpha = 0.85)
```

Out[43]:
```
Text(0.5, 1.0, 'Products Category')
```

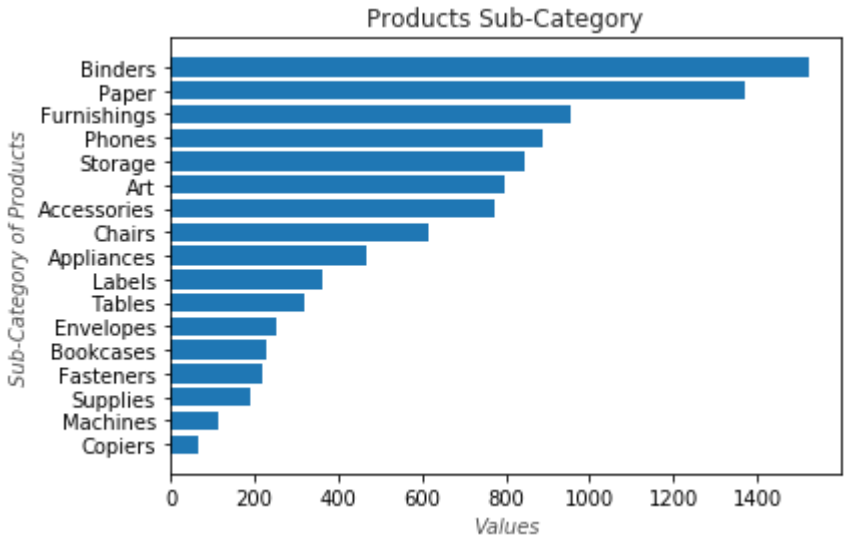


In [44]:
```
products_subcat = dataset["Sales"].groupby(dataset["Sub-Category"])
products_subcat1 = products_subcat.count()
products_subcat11 = products_subcat1.sort_values()
products_subcat11
```

Out[44]:
```
Sub-Category
Copiers          68
Machines        115
Supplies        190
Fasteners       217
Bookcases       228
Envelopes       254
Tables          319
Labels          364
Appliances      466
Chairs          616
Accessories     775
Art             796
Storage         846
Phones          889
Furnishings     957
Paper          1370
Binders        1523
Name: Sales, dtype: int64
```

In [45]:
```
x2 = products_subcat11.index
y2 = products_subcat11.values
plt.barh(x2, y2)
plt.xlabel("Values", alpha = 0.7, fontstyle = 'italic')
plt.ylabel('Sub-Category of Products', alpha = 0.7, fontstyle = 'italic')
plt.title("Products Sub-Category", alpha = 0.85)
```

Out[45]:
```
Text(0.5, 1.0, 'Products Sub-Category')
```



In [46]:
```
dataset.head()
```

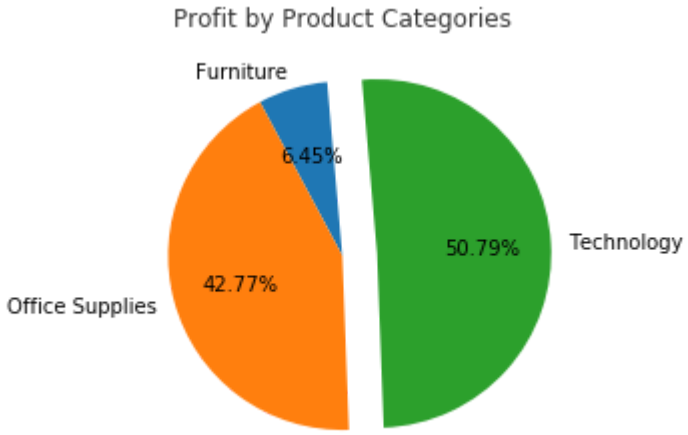
Out[46]:

	Order ID	Order Date	Ship Date	Ship Mode	Customer ID	Customer Name	Segment	Country	City	State
1	CA-2016-152156	2016-11-08	2016-11-11	Second Class	CG-12520	Claire Gute	Consumer	United States	Henderson	Kentucky
2	CA-2016-152156	2016-11-08	2016-11-11	Second Class	CG-12520	Claire Gute	Consumer	United States	Henderson	Kentucky
3	CA-2016-138688	2016-06-12	2016-06-16	Second Class	DV-13045	Darrin Van Huff	Corporate	United States	Los Angeles	California
4	US-2015-108966	2015-10-11	2015-10-18	Standard Class	SO-20335	Sean O'Donnell	Consumer	United States	Fort Lauderdale	Florida
5	US-2015-108966	2015-10-11	2015-10-18	Standard Class	SO-20335	Sean O'Donnell	Consumer	United States	Fort Lauderdale	Florida

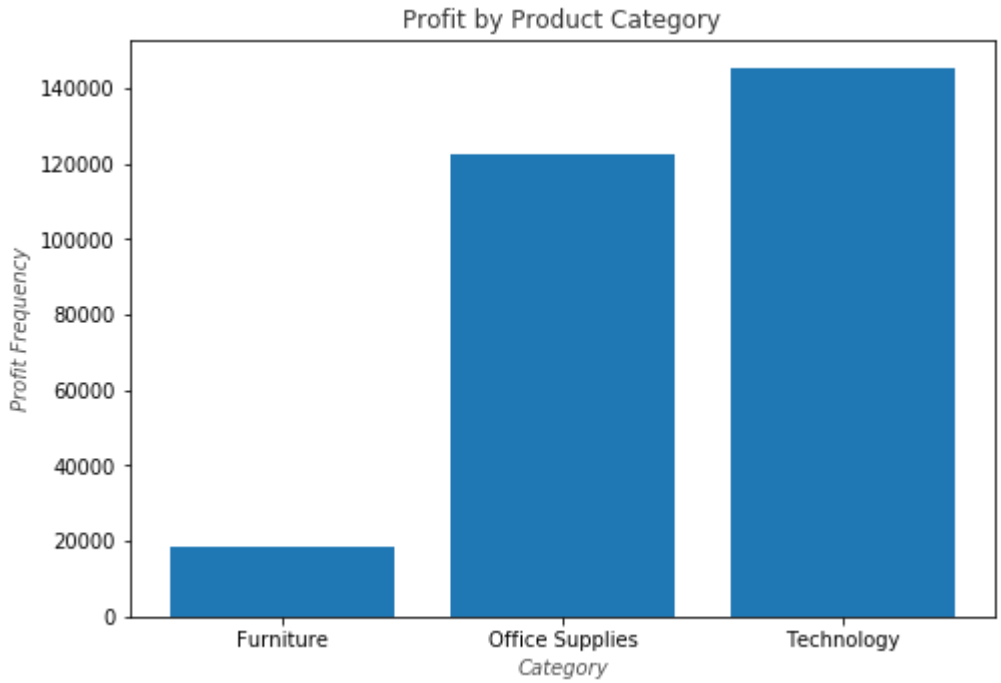
To know which category and sub-category of products brought in the highest profit

In [47]:
```
profit_dist = dataset["Profit"].groupby(dataset["Category"])
profit_dist1 = profit_dist.sum()
profit_dist1
label = profit_dist1.index
explodes = [0,0,0.2]
```

In [48]:
```
plt.pie(profit_dist1, autopct = "%1.2f%%", startangle = 95, labels = l
abel, explode = explodes)
plt.title("Profit by Product Categories", alpha = 0.8)
plt.savefig("S-Dset10.png")
```



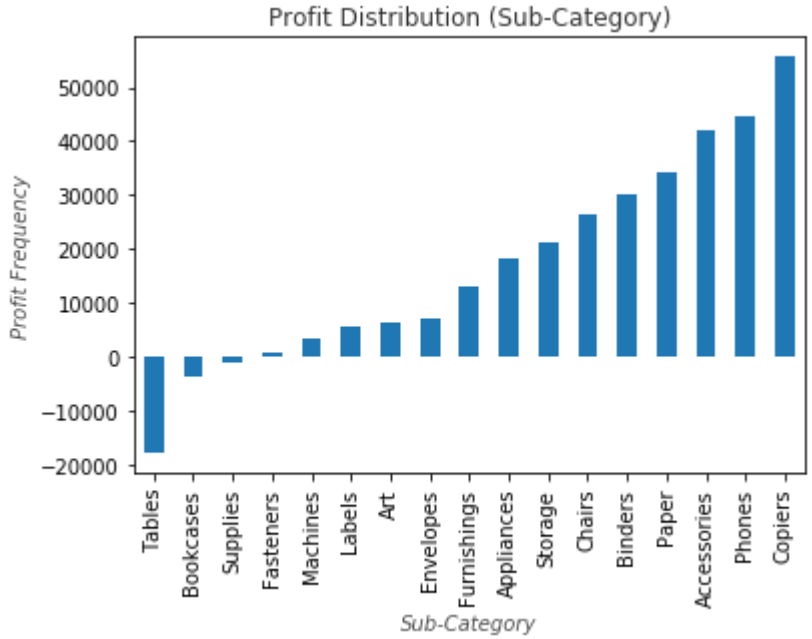
In [49]:
```
fig = plt.figure()
axes = fig.add_axes([0,0,1,1])
axes.bar(profit_dist1.index, profit_dist1.values)
axes.set_xlabel("Category", alpha = 0.7, fontstyle = "italic")
axes.set_ylabel("Profit Frequency", alpha = 0.7, fontstyle = 'italic')
axes.set_title("Profit by Product Category", alpha = 0.8)
fig.savefig("S-Dset11.png")
```



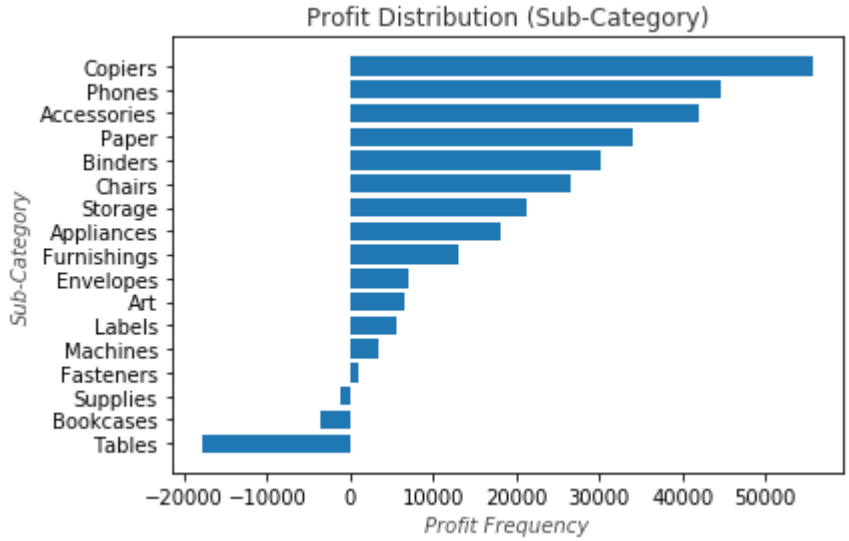
In [50]:
```
profits_subcat = dataset['Profit'].groupby(dataset['Sub-Category'])
profits_subcat1 = profits_subcat.sum()
profits_subcat11 = profits_subcat1.sort_values()
profits_subcat11
```

Out[50]:
```
Sub-Category
Tables        -17725.4811
Bookcases      -3472.5560
Supplies       -1189.0995
Fasteners        949.5182
Machines        3384.7569
Labels          5546.2540
Art             6527.7870
Envelopes       6964.1767
Furnishings    13059.1436
Appliances     18138.0054
Storage        21278.8264
Chairs         26602.2251
Binders        30221.7633
Paper          34053.5693
Accessories    41936.6357
Phones         44515.7306
Copiers        55617.8249
Name: Profit, dtype: float64
```

In [51]:
```
profits_subcat11.plot(kind = 'bar')
plt.title('Profit Distribution (Sub-Category)', alpha = 0.8)
plt.xlabel('Sub-Category', alpha = 0.7, fontstyle = 'italic')
plt.ylabel("Profit Frequency", alpha = 0.7, fontstyle = 'italic')
plt.savefig('S-Dset12.png')
```



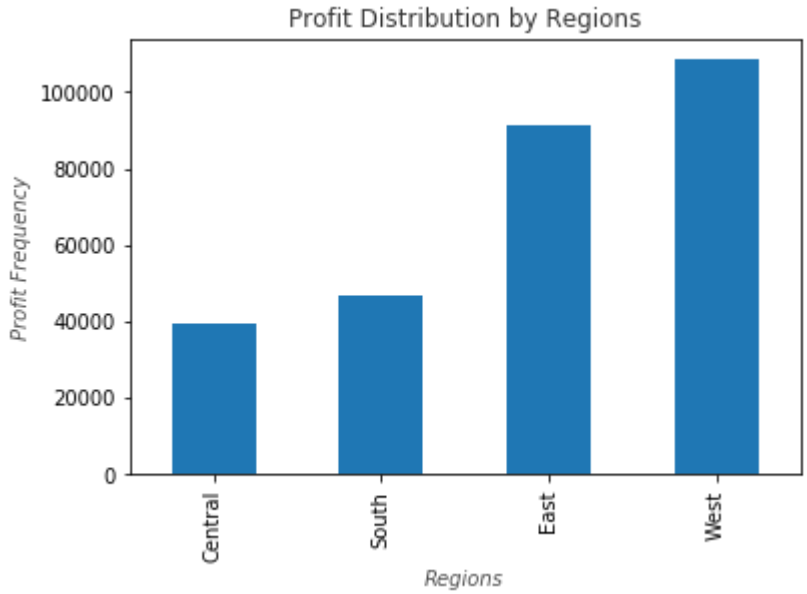
In [52]:
```
plt.barh(profits_subcat11.index, profits_subcat11.values)
plt.title('Profit Distribution (Sub-Category)', alpha = 0.8)
plt.ylabel('Sub-Category', alpha = 0.7, fontstyle = 'italic')
plt.xlabel("Profit Frequency", alpha = 0.7, fontstyle = 'italic')
plt.savefig('S-Dset13.png')
```



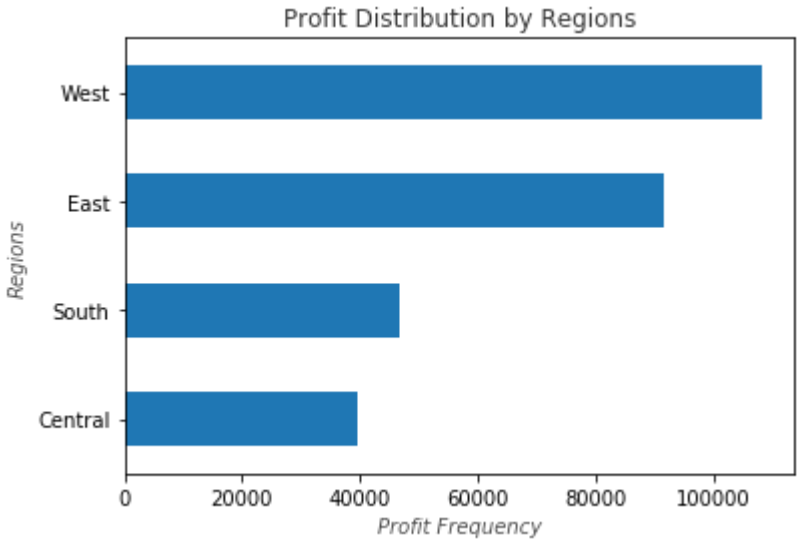
In [53]:
```
profit_by_region = dataset['Profit'].groupby(dataset["Region"])
profit_by_region1 = profit_by_region.sum()
profit_by_region11 = profit_by_region1.sort_values()
profit_by_region11
```

Out[53]:
```
Region
Central     39706.3625
South       46749.4303
East        91534.8388
West       108418.4489
Name: Profit, dtype: float64
```

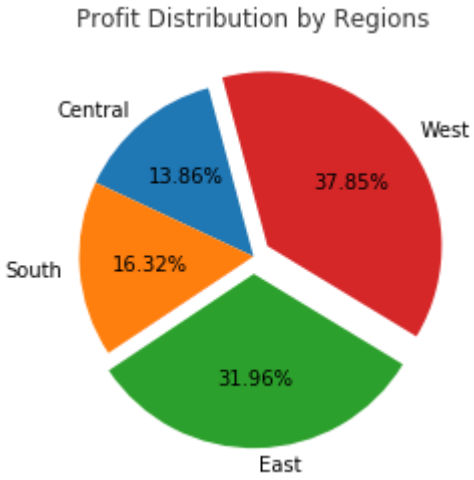
In [54]:
```
profit_by_region11.plot(kind = 'bar')
plt.title('Profit Distribution by Regions', alpha = 0.8)
plt.xlabel('Regions', alpha = 0.7, fontstyle = 'italic')
plt.ylabel("Profit Frequency", alpha = 0.7, fontstyle = 'italic')
plt.savefig('S-Dset14.png')
```



In [55]:
```
profit_by_region11.plot(kind = 'barh')
plt.title('Profit Distribution by Regions', alpha = 0.8)
plt.ylabel('Regions', alpha = 0.7, fontstyle = 'italic')
plt.xlabel("Profit Frequency", alpha = 0.7, fontstyle = 'italic')
plt.savefig('S-Dset15.png')
```



In [56]:
```
expo = [0,0,0.1,0.1]
labelz = profit_by_region11.index
plt.pie(profit_by_region11, startangle = 105, labels = labelz, explode
= expo, autopct = "%1.2f%%")
plt.title("Profit Distribution by Regions", alpha = 0.8)
plt.savefig("S-Dset16.png")
```



In [57]:
```
dataset.tail()
```

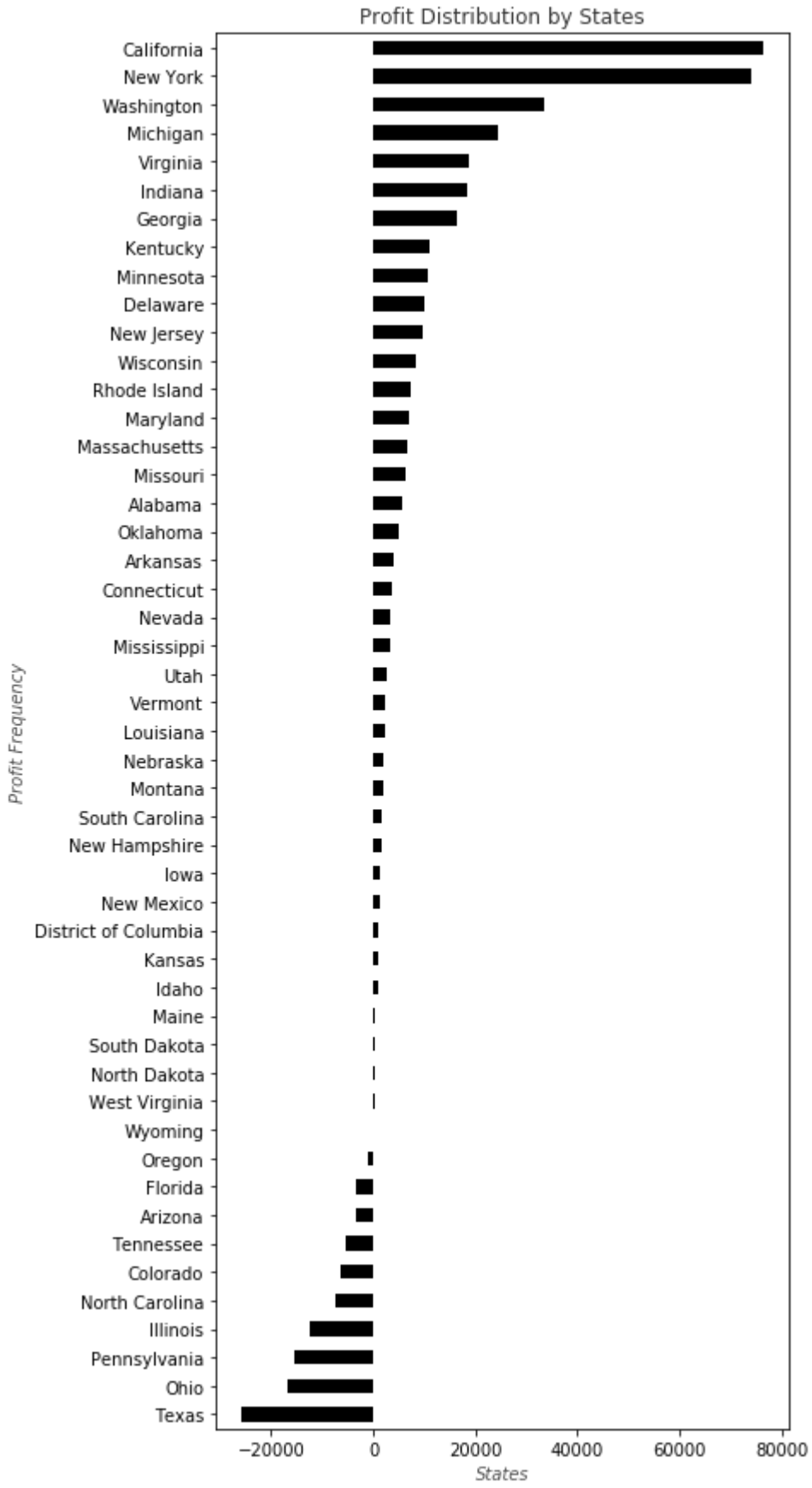
Out[57]:

	Order ID	Order Date	Ship Date	Ship Mode	Customer ID	Customer Name	Segment	Country	City
9989	CA-2014-110422	2014-01-21	2014-01-23	Second Class	TB-21400	Tom Boeckenhauer	Consumer	United States	Miami
9990	CA-2017-121258	2017-02-26	2017-03-03	Standard Class	DB-13060	Dave Brooks	Consumer	United States	Costa Mesa
9991	CA-2017-121258	2017-02-26	2017-03-03	Standard Class	DB-13060	Dave Brooks	Consumer	United States	Costa Mesa
9992	CA-2017-121258	2017-02-26	2017-03-03	Standard Class	DB-13060	Dave Brooks	Consumer	United States	Costa Mesa
9993	CA-2017-119914	2017-05-04	2017-05-09	Second Class	CC-12220	Chris Cortes	Consumer	United States	Westminster

In [58]:
```
profits_by_state = dataset["Profit"].groupby(dataset["State"])
profits_by_state1 = profits_by_state.sum()
profits_by_state11 = profits_by_state1.sort_values()
profits_by_state11
```

Out[58]:
```
State
Texas                 -25729.3563
Ohio                  -16959.3178
Pennsylvania          -15559.9603
Illinois              -12607.8870
North Carolina         -7490.9122
Colorado               -6527.8579
Tennessee              -5341.6936
Arizona                -3427.9246
Florida                -3399.3017
Oregon                 -1190.4705
Wyoming                  100.1960
West Virginia            185.9216
North Dakota             230.1497
South Dakota             394.8283
Maine                    454.4862
Idaho                    826.7231
Kansas                   836.4435
District of Columbia    1059.5893
New Mexico              1157.1161
Iowa                    1183.8119
New Hampshire           1706.5028
South Carolina          1769.0566
Montana                 1833.3285
Nebraska                2037.0942
Louisiana               2196.1023
Vermont                 2244.9783
Utah                    2546.5335
Mississippi             3172.9762
Nevada                  3316.7659
Connecticut             3511.4918
Arkansas                4008.6871
Oklahoma                4853.9560
Alabama                 5786.8253
Missouri                6436.2105
Massachusetts           6785.5016
Maryland                7031.1788
Rhode Island            7285.6293
Wisconsin               8401.8004
New Jersey              9772.9138
Delaware                9977.3748
Minnesota              10823.1874
Kentucky               11199.6966
Georgia                16250.0433
Indiana                18382.9363
Virginia               18597.9504
Michigan               24463.1876
Washington             33402.6517
New York               74038.5486
California             76381.3871
Name: Profit, dtype: float64
```

In [59]:
```
profits_by_state11.plot(kind = 'barh', color = 'k', figsize = (6,15))
plt.title('Profit Distribution by States', alpha = 0.8)
plt.xlabel('States', alpha = 0.7, fontstyle = 'italic')
plt.ylabel("Profit Frequency", alpha = 0.7, fontstyle = 'italic')
plt.savefig('S-Dset17.png')
```


Profit Distribution by States
California
New York
Washington
Michigan
Virginia
Indiana
Georgia
Kentucky
Minnesota
Delaware
New Jersey
Wisconsin
Rhode Island
Maryland
Massachusetts
Missouri
Alabama
Oklahoma
Arkansas
Connecticut
Nevada
Mississippi
Utah
Vermont
Louisiana
Nebraska
Montana
South Carolina
New Hampshire
Iowa
New Mexico
District of Columbia
Kansas
Idaho
Maine
South Dakota
North Dakota
West Virginia
Wyoming
Oregon
Florida
Arizona
Tennessee
Colorado
North Carolina
Illinois
Pennsylvania
Ohio
Texas
Profit Frequency
−20000
0
20000
40000
60000
80000
States

In [60]:
```
dataset.head()
```

Out[60]:

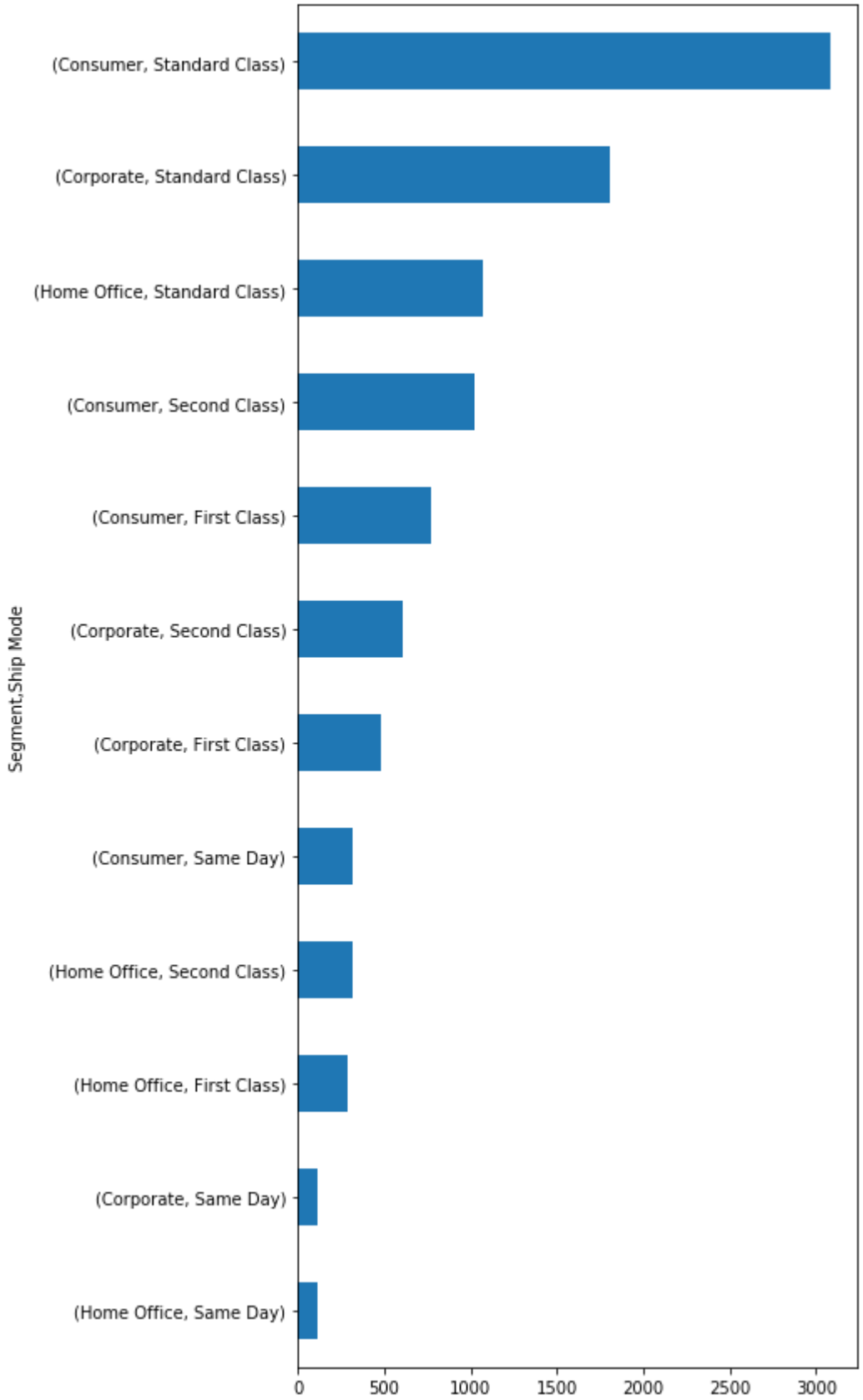
	Order ID	Order Date	Ship Date	Ship Mode	Customer ID	Customer Name	Segment	Country	City	State
1	CA-2016-152156	2016-11-08	2016-11-11	Second Class	CG-12520	Claire Gute	Consumer	United States	Henderson	Kentucky
2	CA-2016-152156	2016-11-08	2016-11-11	Second Class	CG-12520	Claire Gute	Consumer	United States	Henderson	Kentucky
3	CA-2016-138688	2016-06-12	2016-06-16	Second Class	DV-13045	Darrin Van Huff	Corporate	United States	Los Angeles	California
4	US-2015-108966	2015-10-11	2015-10-18	Standard Class	SO-20335	Sean O'Donnell	Consumer	United States	Fort Lauderdale	Florida
5	US-2015-108966	2015-10-11	2015-10-18	Standard Class	SO-20335	Sean O'Donnell	Consumer	United States	Fort Lauderdale	Florida

In [61]:
```
seg_ship = dataset.Sales.groupby([dataset.Segment, dataset["Ship Mode"
]])
seg_ship1 = seg_ship.count().sort_values()
seg_ship1
```

Out[61]:
```
Segment      Ship Mode
Home Office  Same Day          112
Corporate    Same Day          114
Home Office  First Class       284
             Second Class      316
Consumer     Same Day          317
Corporate    First Class       485
             Second Class      609
Consumer     First Class       769
             Second Class     1020
Home Office  Standard Class   1070
Corporate    Standard Class   1812
Consumer     Standard Class   3085
Name: Sales, dtype: int64
```

In [62]:
```
seg_ship1.plot(kind = "barh", figsize = (6,15))
```

Out[62]:
```
<matplotlib.axes._subplots.AxesSubplot at 0xa62b0350>
```



In [63]:
```
subcat_ship = dataset["Sales"].groupby([dataset["Sub-Category"], datas
et["Ship Mode"]])
subcat_ship1 = subcat_ship.count()
subcat_ship1
```

Out[63]:
```
Sub-Category  Ship Mode
Accessories   First Class       128
              Same Day           41
              Second Class      162
              Standard Class    444
Appliances    First Class        76
                                ...
Supplies      Standard Class    111
Tables        First Class        47
              Same Day           21
              Second Class       61
              Standard Class    190
Name: Sales, Length: 68, dtype: int64
```

Now, let's try some other visualization.

You'll extract Sales across various Regions and States of the United States

Note that these exercise are just for your practice sake. The sales is made over a period of years. There result obtained is the sales over time. To obtain that of specific periods will require different approach.

The info extracted below is the overall sales over a period of time

In [64]:
```
newset = dataset.Sales.groupby([dataset.Region, dataset.State])
newset1 = newset.sum()
newset1.head(15)
```

Out[64]:
```
Region   State
Central  Illinois         80166.1010
         Indiana          53555.3600
         Iowa              4579.7600
         Kansas            2914.3100
         Michigan         76269.6140
         Minnesota        29863.1500
         Missouri         22205.1500
         Nebraska          7464.9300
         North Dakota       919.9100
         Oklahoma         19683.3900
         South Dakota      1315.5600
         Texas           170188.0458
         Wisconsin        32114.6100
East     Connecticut      13384.3570
         Delaware         27451.0690
Name: Sales, dtype: float64
```

In [65]:
```
newset11 = newset1.unstack(0)
newset11.head(20)
```

Out[65]:

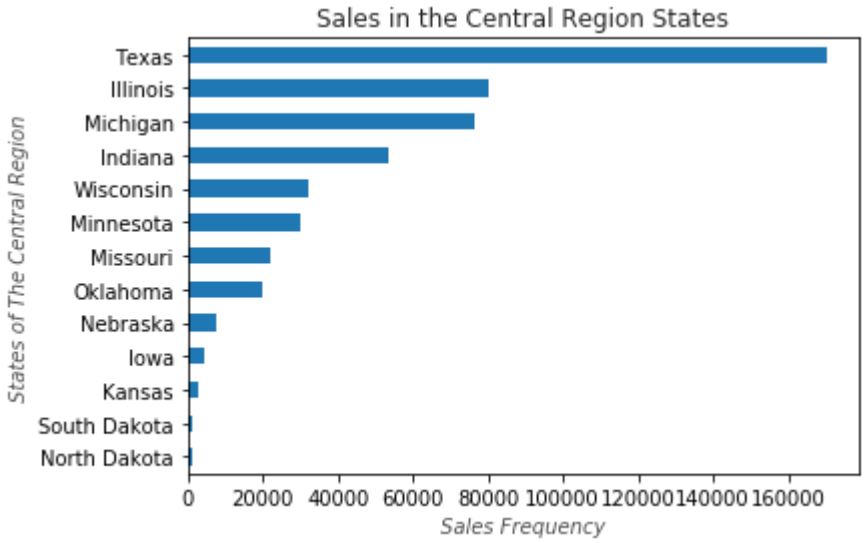
Region	Central	East	South	West
State				
Alabama	NaN	NaN	19510.640	NaN
Arizona	NaN	NaN	NaN	35282.0010
Arkansas	NaN	NaN	11678.130	NaN
California	NaN	NaN	NaN	457687.6315
Colorado	NaN	NaN	NaN	32108.1180
Connecticut	NaN	13384.357	NaN	NaN
Delaware	NaN	27451.069	NaN	NaN
District of Columbia	NaN	2865.020	NaN	NaN
Florida	NaN	NaN	89473.708	NaN
Georgia	NaN	NaN	49095.840	NaN
Idaho	NaN	NaN	NaN	4382.4860
Illinois	80166.101	NaN	NaN	NaN
Indiana	53555.360	NaN	NaN	NaN
Iowa	4579.760	NaN	NaN	NaN
Kansas	2914.310	NaN	NaN	NaN
Kentucky	NaN	NaN	36591.750	NaN
Louisiana	NaN	NaN	9217.030	NaN
Maine	NaN	1270.530	NaN	NaN
Maryland	NaN	23705.523	NaN	NaN
Massachusetts	NaN	28634.434	NaN	NaN

The Sales made in the states under the Central Region is extracted below

In [66]:
```
cent_states = newset11.Central[newset11.Central > 0]
cent_states
```

Out[66]:
```
State
Illinois         80166.1010
Indiana          53555.3600
Iowa              4579.7600
Kansas            2914.3100
Michigan         76269.6140
Minnesota        29863.1500
Missouri         22205.1500
Nebraska          7464.9300
North Dakota       919.9100
Oklahoma         19683.3900
South Dakota      1315.5600
Texas           170188.0458
Wisconsin        32114.6100
Name: Central, dtype: float64
```

In [67]:
```
sorted = cent_states.sort_values()
sorted.plot.barh()
plt.title("Sales in the Central Region States", alpha = 0.85)
plt.xlabel("Sales Frequency", alpha = 0.7, fontstyle = 'italic')
plt.ylabel("States of The Central Region", alpha = 0.7, fontstyle = "i
talic")
plt.savefig("S-Dset39.png")
```

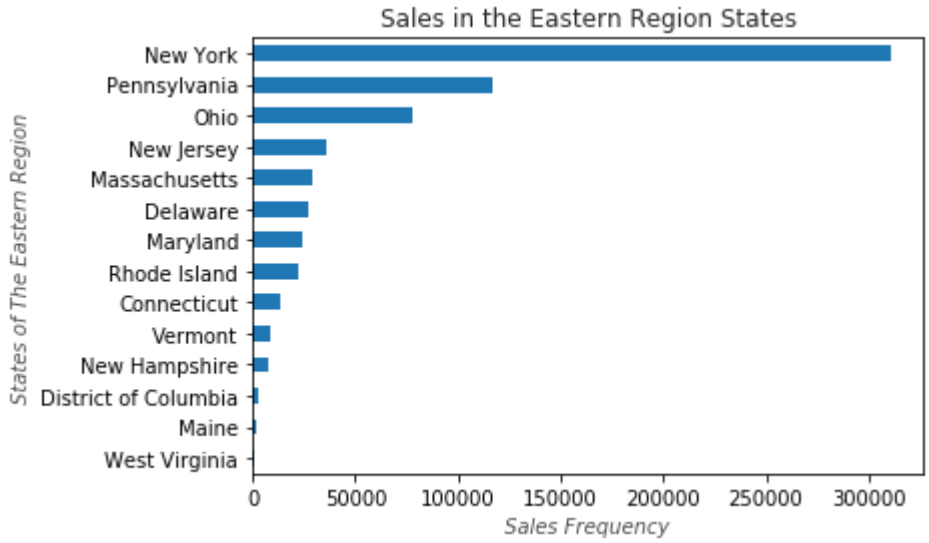


The Sales made in the states under the Eastern Region is extracted below

In [68]:
```
east_states = newset11.East[newset11.East > 0]
east_states
```

Out[68]:
```
State
Connecticut              13384.357
Delaware                 27451.069
District of Columbia      2865.020
Maine                     1270.530
Maryland                 23705.523
Massachusetts            28634.434
New Hampshire             7292.524
New Jersey               35764.312
New York                310876.271
Ohio                     77976.764
Pennsylvania            116511.914
Rhode Island             22627.956
Vermont                   8929.370
West Virginia             1209.824
Name: East, dtype: float64
```

In [69]:
```
east_sorted = east_states.sort_values()
east_sorted.plot.barh()
plt.title("Sales in the Eastern Region States", alpha = 0.85)
plt.xlabel("Sales Frequency", alpha = 0.7, fontstyle = 'italic')
plt.ylabel("States of The Eastern Region", alpha = 0.7, fontstyle = "i
talic")
plt.savefig("S-Dset40.png")
```

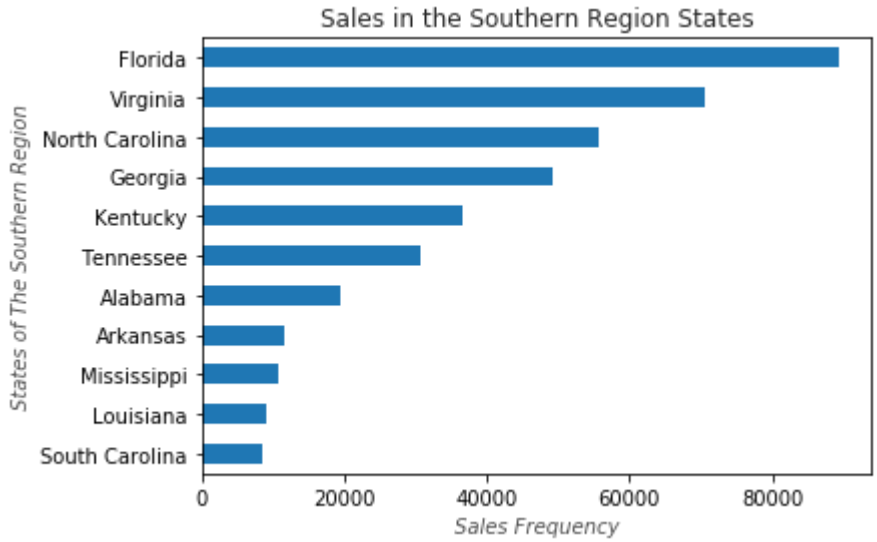


The Sales made in the states under the Southern Region is extracted below

In [70]:
```
south_states = newset11.South[newset11.South > 0]
south_states
```

Out[70]:
```
State
Alabama           19510.640
Arkansas          11678.130
Florida           89473.708
Georgia           49095.840
Kentucky          36591.750
Louisiana          9217.030
Mississippi       10771.340
North Carolina    55603.164
South Carolina     8481.710
Tennessee         30661.873
Virginia          70636.720
Name: South, dtype: float64
```

In [71]:
```
sorted_south = south_states.sort_values()
sorted_south.plot.barh()
plt.title("Sales in the Southern Region States", alpha = 0.85)
plt.xlabel("Sales Frequency", alpha = 0.7, fontstyle = 'italic')
plt.ylabel("States of The Southern Region", alpha = 0.7, fontstyle =
"italic")
plt.savefig("S-Dset41.png")
```

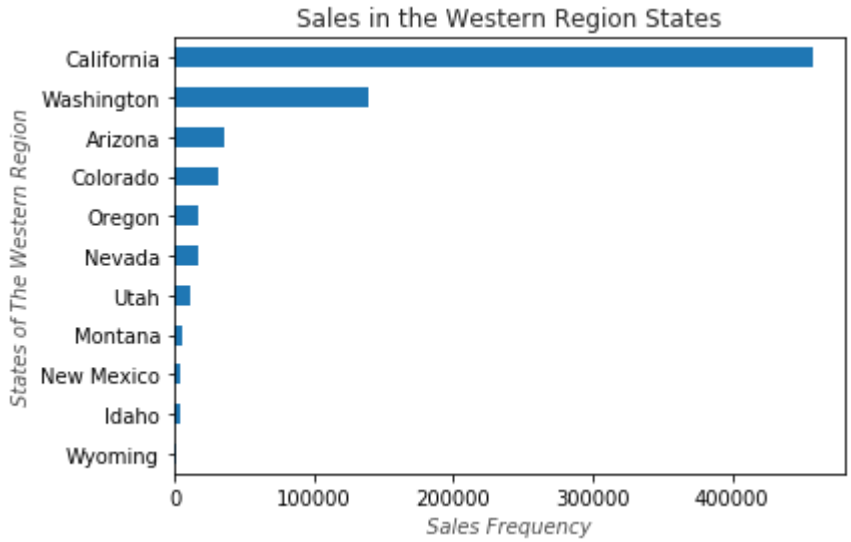


The Sales made in the states under the Western Region is extracted below

In [72]:
```
west_states = newset11.West[newset11.West > 0]
west_states
```

Out[72]:
```
State
Arizona        35282.0010
California    457687.6315
Colorado       32108.1180
Idaho           4382.4860
Montana         5589.3520
Nevada         16729.1020
New Mexico      4783.5220
Oregon         17431.1500
Utah           11220.0560
Washington    138641.2700
Wyoming         1603.1360
Name: West, dtype: float64
```

In [73]:
```
sorted_west = west_states.sort_values()
sorted_west.plot.barh()
plt.title("Sales in the Western Region States", alpha = 0.85)
plt.xlabel("Sales Frequency", alpha = 0.7, fontstyle = 'italic')
plt.ylabel("States of The Western Region", alpha = 0.7, fontstyle = "i
talic")
plt.savefig("S-Dset42.png")
```



Here comes the end of another lesson using all the amazing libraries. And i must say it again that I'm excited at the possibilities one can achieve using the Python Data Science libraries.

The best way to learn is by doing, so, get your hands into action.

Practice! Practice!! Practice!!!

And I urge you, once again, to feel encouraged by what you're doing. It'll get better and come naturally as you get more committed to the process

Happy Learning !

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: