Importing necessary libraries

In [1]:

```python
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
```

Reading Data Set

In [3]:

```python
df = pd.read_csv('SampleSuperstore.csv')
df
```

Out[3]:

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

9994 rows × 13 columns

View the first 5 rows of our dataset / The last five rows of the dataset, use the tail() method.

In [4]:

```python
df.head(5)
```

Out[4]:

	Ship Mode	Segment	Country	City	State	Postal Code	Region	Category	Sub-Category	Sa
0	Second Class	Consumer	United States	Henderson	Kentucky	42420	South	Furniture	Bookcases	261.9
1	Second Class	Consumer	United States	Henderson	Kentucky	42420	South	Furniture	Chairs	731.9
2	Second Class	Corporate	United States	Los Angeles	California	90036	West	Office Supplies	Labels	14.6
3	Standard Class	Consumer	United States	Fort Lauderdale	Florida	33311	South	Furniture	Tables	957.5
4	Standard Class	Consumer	United States	Fort Lauderdale	Florida	33311	South	Office Supplies	Storage	22.3

View all the columns in the Dataframe.

In [5]:

```
df.columns
```

Out[5]:

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

View the shape of the Dataframe that contains the number of rows and the number of columns.

In [6]:

```
df.shape
```

Out[6]:

```
(9994, 13)
```

View the information like Range index, datatypes, number of non-null entries for each column by using the info() method.

In [7]:

```
df.info()
```

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

Take a sample from the data set

In [8]:

```
df.sample( )
```

Out[8]:

	Ship Mode	Segment	Country	City	State	Postal Code	Region	Category	Sub-Category	Sales
1564	Same Day	Consumer	United States	Columbia	South Carolina	29203	South	Office Supplies	Storage	354.9

Find the unique values in the data set.

In [9]:

```
df.nunique( )
```

Out[9]:

```
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
```

Returns how much memory each column uses in bytes. It is useful especially when we work with large data frames.

In [ ]:

```
df.memory_usage( )
```

Out[ ]:

```
Index             128
Ship Mode       79952
Segment         79952
Country         79952
City            79952
State           79952
Postal Code     79952
Region          79952
Category        79952
Sub-Category    79952
Sales           79952
Quantity        79952
Discount        79952
Profit          79952
dtype: int64
```

Returns the first n rows ordered by columns in descending order. (only Numeric columns)

In [ ]:

```
df.nlargest(10, 'Quantity')
```

Out[ ]:

	Ship Mode	Segment	Country	City	State	Postal Code	Region	Category	Sub-Category
113	Second Class	Consumer	United States	Columbus	Ohio	43229	East	Office Supplies	Fasteners
139	Standard Class	Consumer	United States	Roseville	California	95661	West	Furniture	Furnishings
575	Second Class	Consumer	United States	Long Beach	California	90805	West	Office Supplies	Paper
660	Standard Class	Consumer	United States	Arlington	Texas	76017	Central	Office Supplies	Storage
1045	Standard Class	Home Office	United States	Rockford	Illinois	61107	Central	Furniture	Chairs
1363	First Class	Corporate	United States	Tucson	Arizona	85705	West	Office Supplies	Binders
1429	Second Class	Corporate	United States	Salinas	California	93905	West	Office Supplies	Labels
1433	Second Class	Consumer	United States	Florence	Alabama	35630	South	Furniture	Chairs
1711	Standard Class	Consumer	United States	San Francisco	California	94122	West	Office Supplies	Appliances
2793	Standard Class	Corporate	United States	Redondo Beach	California	90278	West	Technology	Phones

Returns a boolean Series denoting duplicate rows.

In [ ]:

```
duplicates = df.duplicated().sum()
duplicates
```

Out[ ]:

```
17
```

To check if there are null values in the df, use isnull() method.

In [ ]:

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

Out[ ]:

```
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
```

Observations:

1. There are no null values over the entire data.
2. Thus no necessities of imputations.

View the unique categories in the data frame.

```
In [ ]: df['Category'].unique()
```

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

View the states in the dataset.

```
In [ ]: df['State'].unique()
```

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

Observation: There are 49 states in this df.

Let's find uniques categories and sub categories in the data frame:

```
In [ ]: df['Sub-Category'].unique()
```

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

Find the value count of the segment column

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

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

View the statistical description of the Dataframe.

- Description contains the count of features, mean of them, Standard deviation, minimum and maximum values in that particular
- attribute, 25%, 50%, 75% of the values in the dataset.

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

Out[ ]:

	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

Creating Profit Dataframe

```
In [ ]: profit_df = df[df['Profit'] > 0]
        profit_df
```

Out[ ]:

	Ship Mode	Segment	Country	City	State	Postal Code	Region	Category	Sub-Category
0	Second Class	Consumer	United States	Henderson	Kentucky	42420	South	Furniture	Bookcases
1	Second Class	Consumer	United States	Henderson	Kentucky	42420	South	Furniture	Chairs
2	Second Class	Corporate	United States	Los Angeles	California	90036	West	Office Supplies	Labels
4	Standard Class	Consumer	United States	Fort Lauderdale	Florida	33311	South	Office Supplies	Storage
5	Standard Class	Consumer	United States	Los Angeles	California	90032	West	Furniture	Furnishings
...	...	...	...	...	...	...	...	...	...
9989	Second Class	Consumer	United States	Miami	Florida	33180	South	Furniture	Furnishings
9990	Standard Class	Consumer	United States	Costa Mesa	California	92627	West	Furniture	Furnishings
9991	Standard Class	Consumer	United States	Costa Mesa	California	92627	West	Technology	Phones
9992	Standard Class	Consumer	United States	Costa Mesa	California	92627	West	Office Supplies	Paper
9993	Second Class	Consumer	United States	Westminster	California	92683	West	Office Supplies	Appliances

8058 rows × 13 columns

Viewing its shape, Size , info and describe this frame.

```
In [ ]: df.groupby(by='Segment').sum()
```

Out[ ]:

	Postal Code	Sales	Quantity	Discount	Profit
Segment					
Consumer	288878609	1.161401e+06	19521	820.91	134119.2092
Corporate	164536330	7.061464e+05	11608	477.85	91979.1340
Home Office	98157713	4.296531e+05	6744	262.33	60298.6785

In [ ]:

```
loss_df= df[df['Profit'] < 0]
loss_df
```

Out[ ]:

	Ship Mode	Segment	Country	City	State	Postal Code	Region	Category	Sub Categor
3	Standard Class	Consumer	United States	Fort Lauderdale	Florida	33311	South	Furniture	Table
14	Standard Class	Home Office	United States	Fort Worth	Texas	76106	Central	Office Supplies	Appliance
15	Standard Class	Home Office	United States	Fort Worth	Texas	76106	Central	Office Supplies	Binder
23	Second Class	Consumer	United States	Philadelphia	Pennsylvania	19140	East	Furniture	Chair
27	Standard Class	Consumer	United States	Philadelphia	Pennsylvania	19140	East	Furniture	Bookcase
...	...	...	...	...	...	...	...	...	.
9920	Standard Class	Corporate	United States	Bryan	Texas	77803	Central	Office Supplies	Binder
9921	Standard Class	Home Office	United States	Akron	Ohio	44312	East	Office Supplies	Binder
9931	Standard Class	Consumer	United States	San Bernardino	California	92404	West	Furniture	Bookcase
9937	Second Class	Corporate	United States	Los Angeles	California	90049	West	Furniture	Table
9962	First Class	Home Office	United States	Houston	Texas	77041	Central	Furniture	Bookcase

1871 rows × 13 columns

View the shape of loss df

In [ ]:

```
loss_df.shape
```

Out[ ]:

```
(1871, 13)
```

only 1871 rows that are related to loss.

In [ ]:

```
loss_df.describe()
```

Out[ ]:

	Postal Code	Sales	Quantity	Discount	Profit
count	1871.000000	1871.000000	1871.000000	1871.000000	1871.000000
mean	55991.122929	250.511574	3.762694	0.480887	-83.448042
std	26041.501999	715.067296	2.141347	0.235080	284.423422
min	1841.000000	0.444000	1.000000	0.100000	-6599.978000
25%	33024.000000	12.503000	2.000000	0.200000	-58.660950
50%	60623.000000	71.088000	3.000000	0.400000	-18.088200
75%	77095.000000	284.922000	5.000000	0.700000	-6.261500
max	98198.000000	22638.480000	14.000000	0.800000	-0.089500

In [ ]:

```
Total_loss=np.negative(loss_df['Profit'].sum())
print("Total loss = %.2f" %Total_loss)
```

```
Total loss = 156131.29
```

In [ ]:

```
loss_df.groupby(by='Segment').sum()
```

Out[ ]:

	Postal Code	Sales	Quantity	Discount	Profit
Segment					
Consumer	57202260	247196.2460	3651	476.76	-84945.7112
Corporate	30034273	131860.5383	2191	272.00	-44787.2076
Home Office	17522858	89650.3705	1198	150.98	-26398.3669

More discount leads to more loss, so, to make more profit provide fewer discounts.

In [ ]:

```
loss_df.groupby(by='Sub-Category').sum()
```

Out[ ]:

	Postal Code	Sales	Quantity	Discount	Profit
Sub-Category					
Accessories	5286382	10958.8000	330	18.20	-930.6265
Appliances	4825871	3382.5340	235	53.60	-8629.6412
Binders	32609300	36140.6130	2456	452.40	-38510.4964
Bookcases	6423506	48072.7408	422	37.99	-12152.2060
Chairs	15008025	91988.4560	876	61.40	-9880.8413
Fasteners	701930	149.2800	55	2.40	-33.1952
Furnishings	10970913	12845.8440	597	88.60	-6490.9134
Machines	2236261	72456.2530	157	25.60	-30118.6682
Phones	6105294	35797.8400	476	46.60	-7530.6235
Storage	8606475	37869.0720	569	32.20	-6426.3038
Supplies	1761430	14067.1760	110	6.60	-3015.6219
Tables	10224004	104978.5460	757	74.15	-32412.1483

1. We can observe more loss in the Binders category, machines category, and tables category when compared to other categories.

2. Binders are sold more. So even giving less discount may lead to vast loss.

3. So better to give discounts on which are getting less sold so that even they will start getting sold more.

In [ ]:

```
loss_df.groupby(by='City').sum().sort_values('Profit',ascending=True).head(10)
```

Out[ ]:

	Postal Code	Sales	Quantity	Discount	Profit
City					
Philadelphia	4783713	70460.5510	892	115.30	-19590.7411
Houston	14256474	37640.7304	683	104.14	-14785.3668
Chicago	9397492	19910.0120	541	88.20	-11120.6271
San Antonio	2580831	17395.1450	139	17.10	-7831.0254
Lancaster	683904	7699.2420	71	9.40	-7632.4946
Burlington	108868	12044.8740	19	2.00	-5999.3318
Dallas	5487794	9994.0562	280	39.30	-4208.5218
Jacksonville	1237176	31146.2710	154	18.85	-4059.9857
New York City	400828	19533.8020	132	12.20	-3966.0226
Louisville	640216	2884.7840	35	4.90	-3694.1045

In [ ]:

```
loss_df.sort_values(['Sales'],ascending=True).groupby(by='Category').mean()
```

Out[ ]:

	Postal Code	Sales	Quantity	Discount	Profit
Category					
Furniture	59700.907563	361.184295	3.714286	0.367143	-85.344690
Office Supplies	54746.056433	103.395796	3.865688	0.617607	-63.899840
Technology	50287.590406	439.899974	3.553506	0.333579	-142.361322

In [ ]:

```
df.groupby(['State']).sum()['Sales'].nsmallest(10)
```

Out[ ]:

```
State
North Dakota             919.910
West Virginia           1209.824
Maine                   1270.530
South Dakota            1315.560
Wyoming                 1603.136
District of Columbia    2865.020
Kansas                  2914.310
Idaho                   4382.486
Iowa                    4579.760
New Mexico              4783.522
Name: Sales, dtype: float64
```

In [ ]:

```
df.sort_values(['Segment'],ascending=True).groupby('Segment').sum()
```

Out[ ]:

Segment	Postal Code	Sales	Quantity	Discount	Profit
Consumer	288878609	1.161401e+06	19521	820.91	134119.2092
Corporate	164536330	7.061464e+05	11608	477.85	91979.1340
Home Office	98157713	4.296531e+05	6744	262.33	60298.6785

Here Consumer segment sales might be less when compared to other segments, but this is the only segment that provides the highest profits. So, if we increase sales in this Segment by advertisements or something else then, for sure, we can gain more profits.

In [ ]:

```
df.groupby(by='Region').sum()
```

Out[ ]:

Region	Postal Code	Sales	Quantity	Discount	Profit
Central	151786150	501239.8908	8780	558.34	39706.3625
East	50171698	678781.2400	10618	414.00	91522.7800
South	55875052	391721.9050	6209	238.55	46749.4303
West	293739752	725457.8245	12266	350.20	108418.4489