Exploratory Data Analysis - Retail

Name-Pradyumna Rajendra Mangave

Data Science And Business Analytics Intern @TSF

Dataset : https://bit.ly/3i4rbWl

In [1]:
```
#Importing all libraries required in this notebook
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
```

In [2]:
```
Data=pd.read_csv('SampleSuperstore.csv')
Data.head()
```

Out[2]:

	Ship Mode	Segment	Country	City	State	Postal Code	Region	Category	Sub-Category	Sal
0	Second Class	Consumer	United States	Henderson	Kentucky	42420	South	Furniture	Bookcases	261.96(
1	Second Class	Consumer	United States	Henderson	Kentucky	42420	South	Furniture	Chairs	731.94(
2	Second Class	Corporate	United States	Los Angeles	California	90036	West	Office Supplies	Labels	14.62(
3	Standard Class	Consumer	United States	Fort Lauderdale	Florida	33311	South	Furniture	Tables	957.57
4	Standard Class	Consumer	United States	Fort Lauderdale	Florida	33311	South	Office Supplies	Storage	22.36(

In [3]:
```
Data.info()
```

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

In [4]:
```
Data.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]:
```
Data.isnull().sum()
```

Out[5]:
```
Ship Mode       0
Segment         0
Country         0
City            0
State           0
Postal Code     0
Region          0
Category        0
Sub-Category    0
Sales           0
Quantity        0
Discount        0
Profit          0
dtype: int64
```

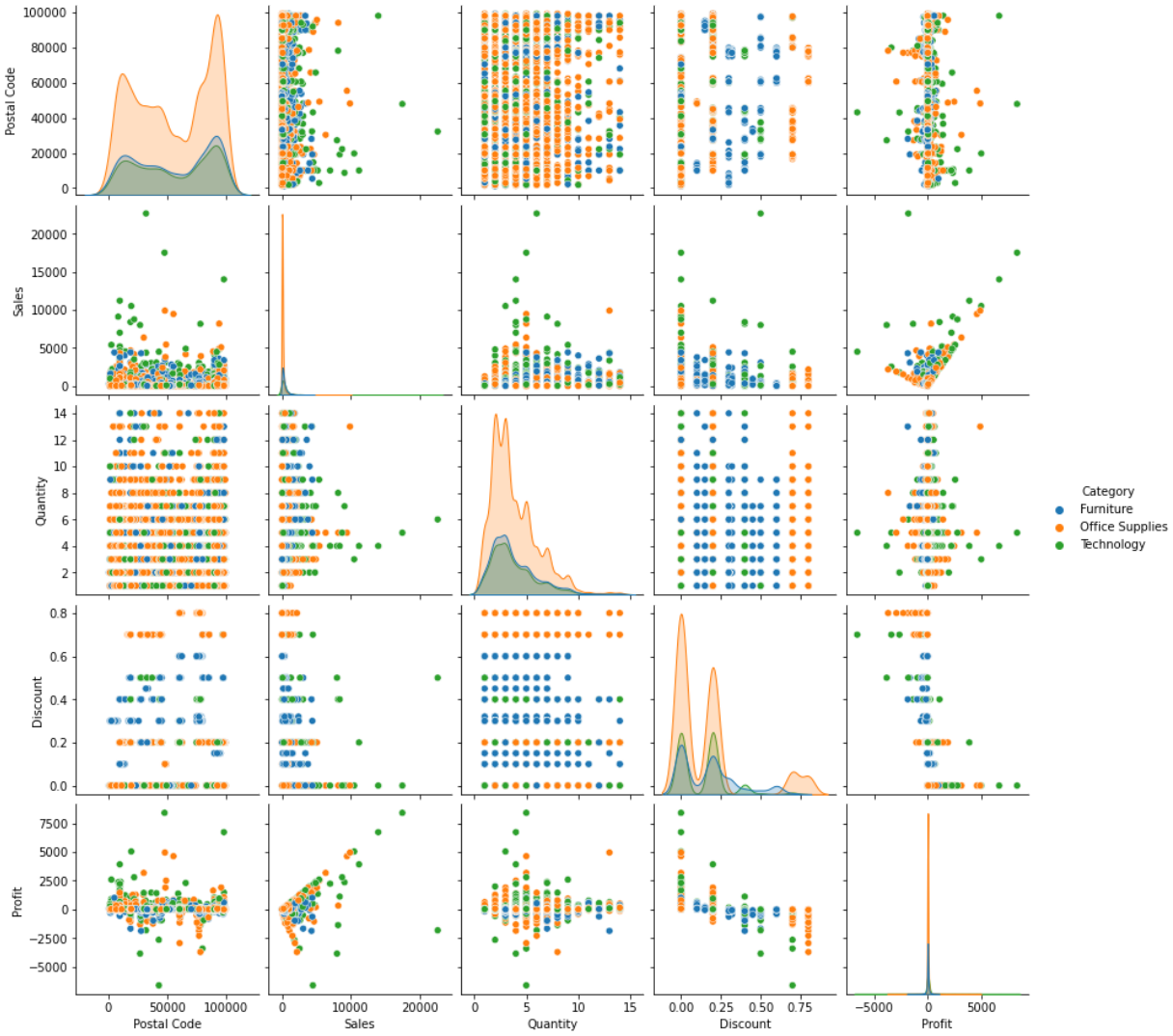
In [6]:
```
Data.corr()
```

Out[6]:

	Postal Code	Sales	Quantity	Discount	Profit
Postal Code	1.000000	-0.023854	0.012761	0.058443	-0.029961
Sales	-0.023854	1.000000	0.200795	-0.028190	0.479064
Quantity	0.012761	0.200795	1.000000	0.008623	0.066253
Discount	0.058443	-0.028190	0.008623	1.000000	-0.219487
Profit	-0.029961	0.479064	0.066253	-0.219487	1.000000

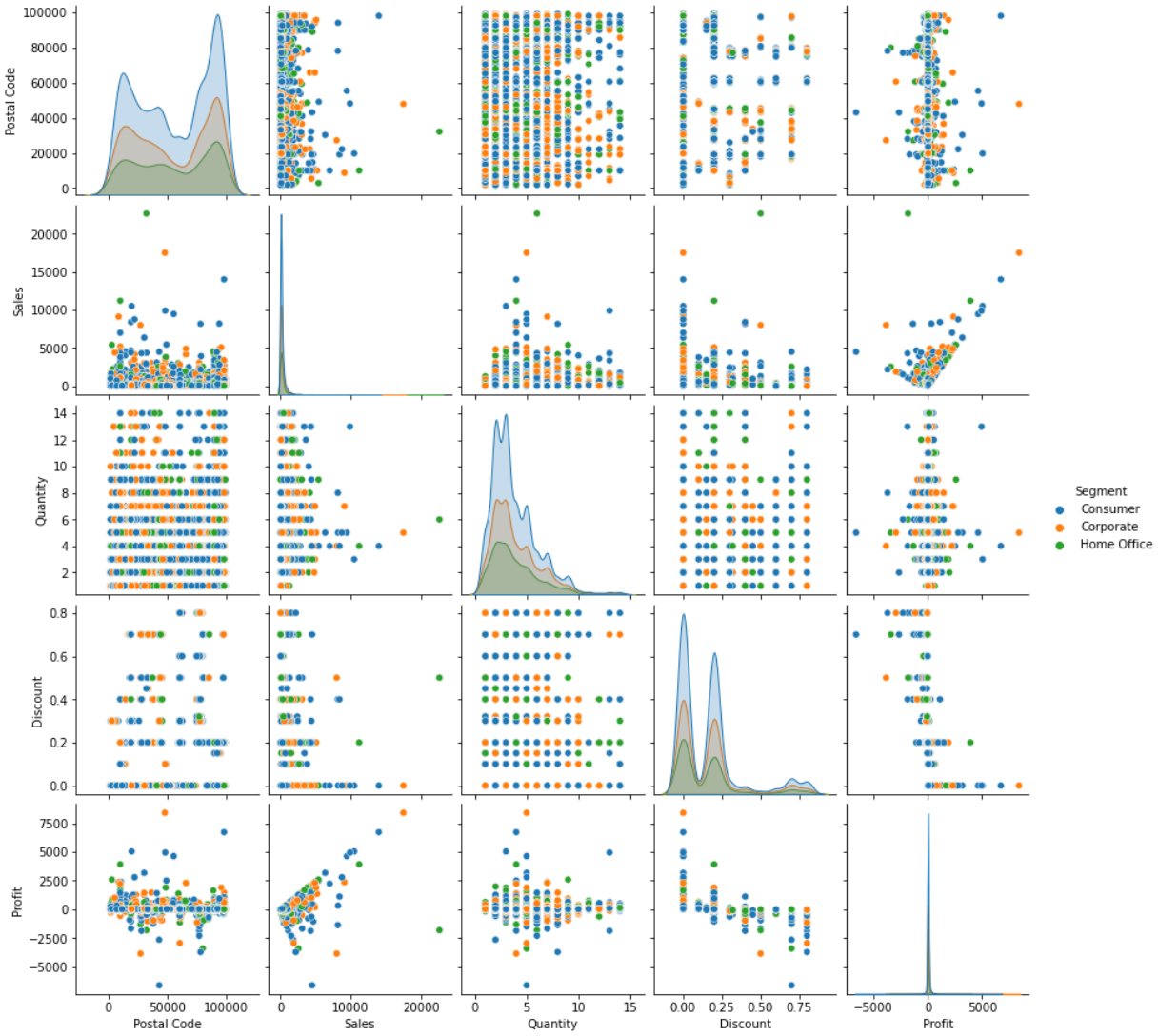
In [7]:
```
# Analysis Using pair Plot
# Pairplot based on 'Category'
sns.pairplot(Data,hue='Category')
```

Out[7]:
```
<seaborn.axisgrid.PairGrid at 0x231876abfa0>
```



In [8]:
```
# Pairplot using Segment
sns.pairplot(Data,hue='Segment')
```

Out[8]: <seaborn.axisgrid.PairGrid at 0x2318ae6f9d0>

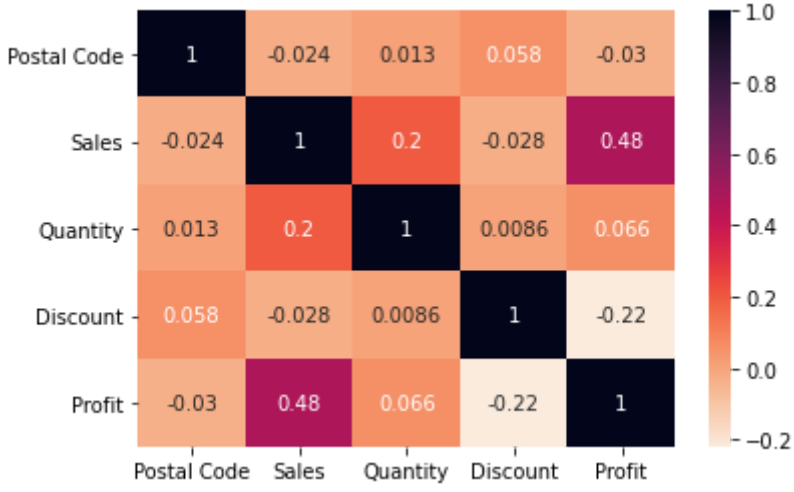


In [9]:
```
#Pairplot Using Region
sns.pairplot(Data,hue='Region')
```

Out[9]: <seaborn.axisgrid.PairGrid at 0x2318da7d700>

In [10]:
```
# Heatmap For Correlation
sns.heatmap(Data.corr(),annot=True,cmap='rocket_r')
```

Out[10]: <AxesSubplot:>



There is correlation in sales and profit

Also there is correlation in Sales, Quantity and Profit

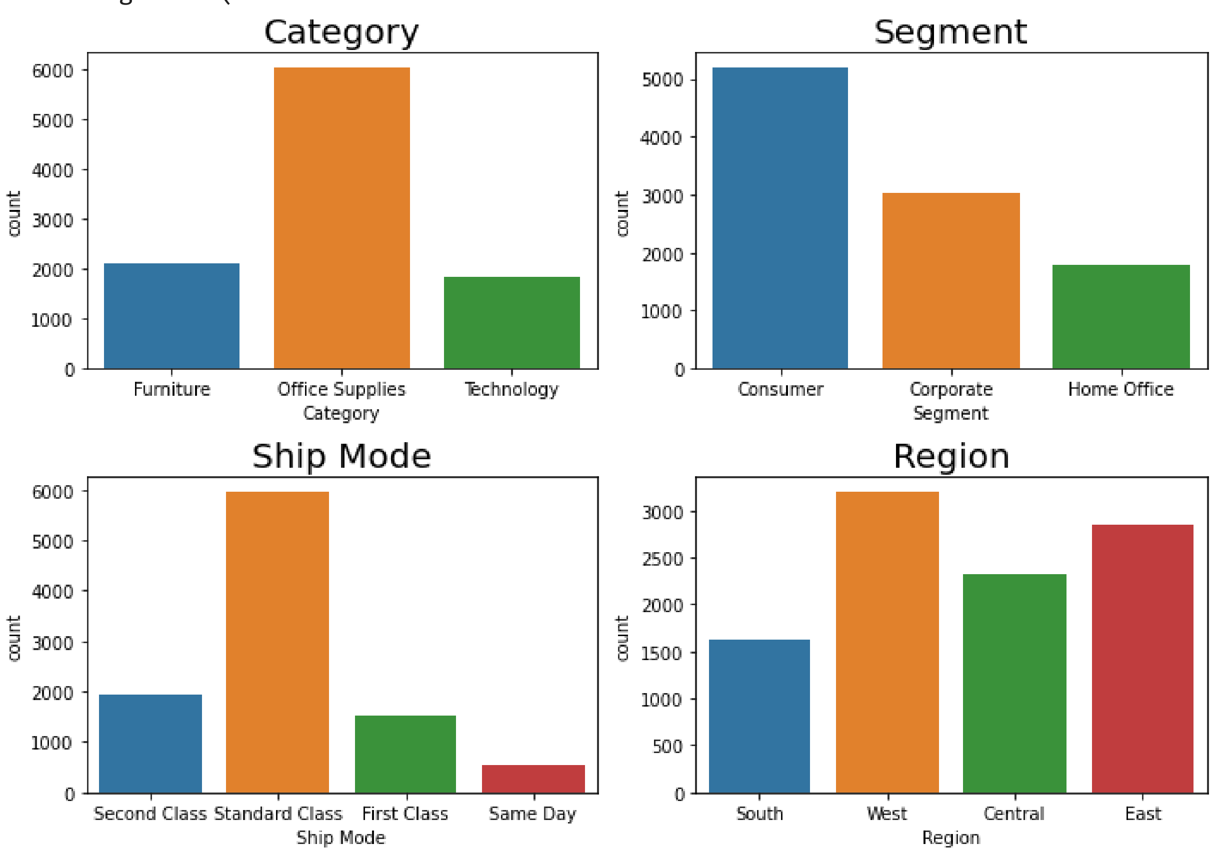
There is negative in postal code and Descount

In [11]:

```
# Countplot for each Columns
fig,axs=plt.subplots(nrows=2,ncols=2,figsize=(10,7))

sns.countplot(Data['Category'],ax=axs[0][0])
sns.countplot(Data['Segment'],ax=axs[0][1])
sns.countplot(Data['Ship Mode'],ax=axs[1][0])
sns.countplot(Data['Region'],ax=axs[1][1])
axs[0][0].set_title('Category',fontsize=20)
axs[0][1].set_title('Segment',fontsize=20)
axs[1][0].set_title('Ship Mode',fontsize=20)
axs[1][1].set_title('Region',fontsize=20)
plt.tight_layout()
```

```
C:\Users\admin\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning:
Pass the following variable as a keyword arg: x. From version 0.12, the only valid p
ositional argument will be `data`, and passing other arguments without an explicit k
eyword will result in an error or misinterpretation.
  warnings.warn(
C:\Users\admin\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning:
Pass the following variable as a keyword arg: x. From version 0.12, the only valid p
ositional argument will be `data`, and passing other arguments without an explicit k
eyword will result in an error or misinterpretation.
  warnings.warn(
C:\Users\admin\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning:
Pass the following variable as a keyword arg: x. From version 0.12, the only valid p
ositional argument will be `data`, and passing other arguments without an explicit k
eyword will result in an error or misinterpretation.
  warnings.warn(
C:\Users\admin\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning:
Pass the following variable as a keyword arg: x. From version 0.12, the only valid p
ositional argument will be `data`, and passing other arguments without an explicit k
eyword will result in an error or misinterpretation.
  warnings.warn(
```



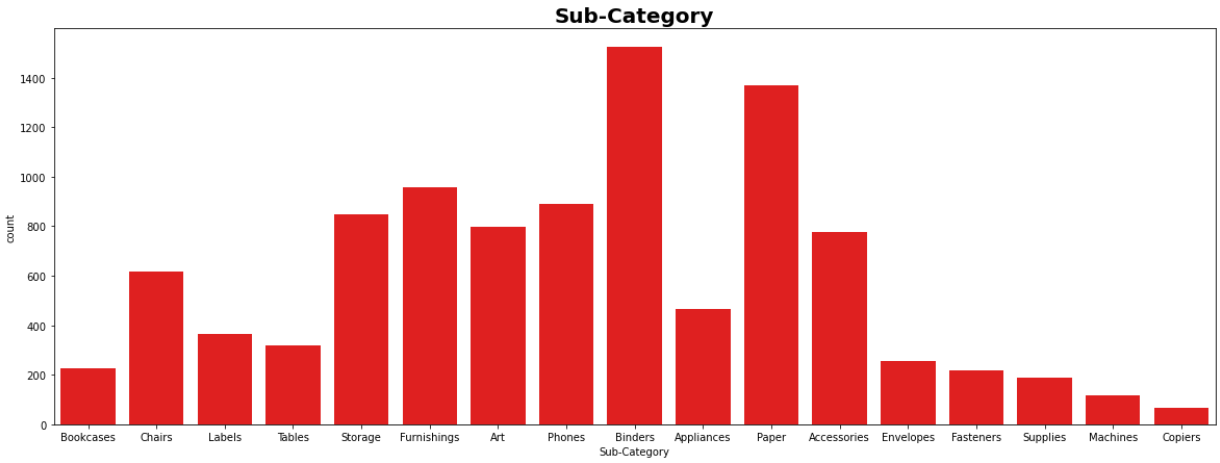
In [12]:

```
plt.figure(figsize=(20,7))
```

```
sns.countplot(Data['Sub-Category'],color='red')
plt.title('Sub-Category',fontsize=20,fontweight='bold')
```

```
C:\Users\admin\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning:
Pass the following variable as a keyword arg: x. From version 0.12, the only valid p
ositional argument will be `data`, and passing other arguments without an explicit k
eyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[12]:
```
Text(0.5, 1.0, 'Sub-Category')
```

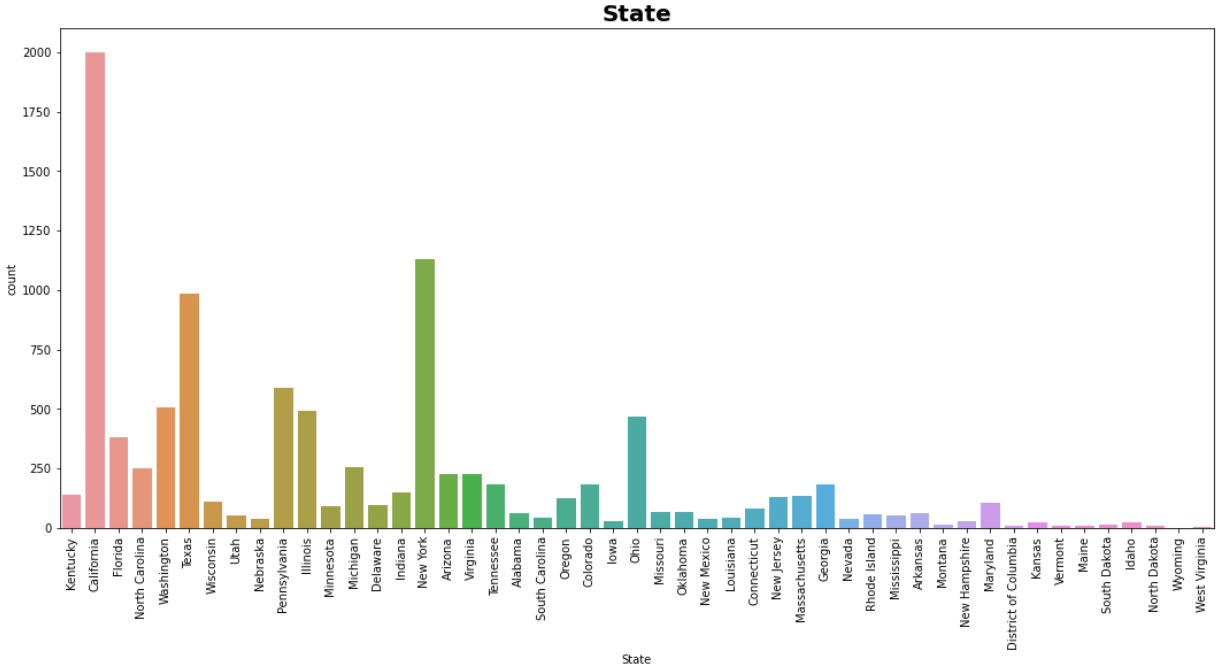


In [13]:
```
plt.figure(figsize=(18,8))
sns.countplot(Data['State'])
plt.title('State',fontsize=20,fontweight='bold')
plt.xticks(rotation=90)
```

```
C:\Users\admin\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning:
Pass the following variable as a keyword arg: x. From version 0.12, the only valid p
ositional argument will be `data`, and passing other arguments without an explicit k
eyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[13]:
```
(array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,
        17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33,
        34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48]),
 [Text(0, 0, 'Kentucky'),
  Text(1, 0, 'California'),
  Text(2, 0, 'Florida'),
  Text(3, 0, 'North Carolina'),
  Text(4, 0, 'Washington'),
  Text(5, 0, 'Texas'),
  Text(6, 0, 'Wisconsin'),
  Text(7, 0, 'Utah'),
  Text(8, 0, 'Nebraska'),
  Text(9, 0, 'Pennsylvania'),
  Text(10, 0, 'Illinois'),
  Text(11, 0, 'Minnesota'),
  Text(12, 0, 'Michigan'),
  Text(13, 0, 'Delaware'),
  Text(14, 0, 'Indiana'),
  Text(15, 0, 'New York'),
  Text(16, 0, 'Arizona'),
  Text(17, 0, 'Virginia'),
  Text(18, 0, 'Tennessee'),
  Text(19, 0, 'Alabama'),
  Text(20, 0, 'South Carolina'),
  Text(21, 0, 'Oregon'),
  Text(22, 0, 'Colorado'),
  Text(23, 0, 'Iowa'),
  Text(24, 0, 'Ohio'),
  Text(25, 0, 'Missouri'),
  Text(26, 0, 'Oklahoma'),
```

```
Text(27, 0, 'New Mexico'),
Text(28, 0, 'Louisiana'),
Text(29, 0, 'Connecticut'),
Text(30, 0, 'New Jersey'),
Text(31, 0, 'Massachusetts'),
Text(32, 0, 'Georgia'),
Text(33, 0, 'Nevada'),
Text(34, 0, 'Rhode Island'),
Text(35, 0, 'Mississippi'),
Text(36, 0, 'Arkansas'),
Text(37, 0, 'Montana'),
Text(38, 0, 'New Hampshire'),
Text(39, 0, 'Maryland'),
Text(40, 0, 'District of Columbia'),
Text(41, 0, 'Kansas'),
Text(42, 0, 'Vermont'),
Text(43, 0, 'Maine'),
Text(44, 0, 'South Dakota'),
Text(45, 0, 'Idaho'),
Text(46, 0, 'North Dakota'),
Text(47, 0, 'Wyoming'),
Text(48, 0, 'West Virginia')])
```



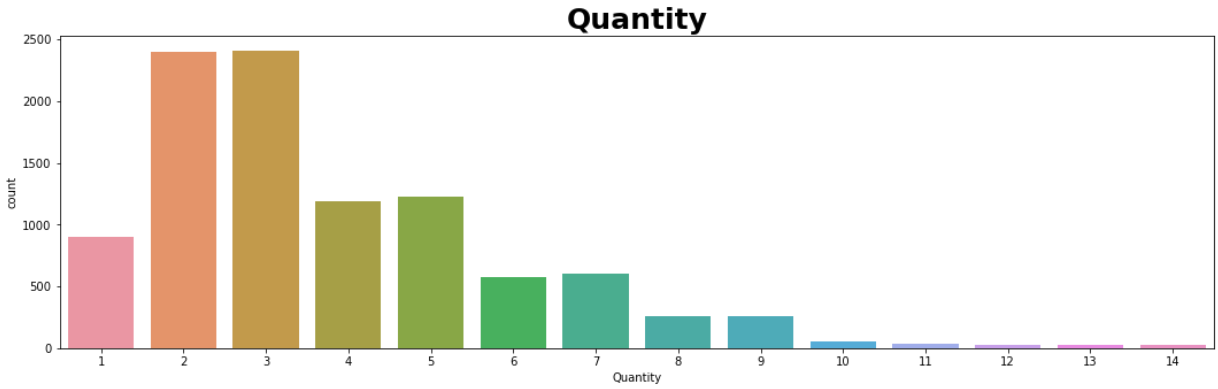
Highest numnber of buyers are from California and New York

In [14]:

```
plt.figure(figsize=(18,5))
sns.countplot(Data['Quantity'])
plt.title('Quantity',fontsize=25,fontweight='bold')
```

```
C:\Users\admin\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning:
Pass the following variable as a keyword arg: x. From version 0.12, the only valid p
ositional argument will be `data`, and passing other arguments without an explicit k
eyword will result in an error or misinterpretation.
  warnings.warn(
```

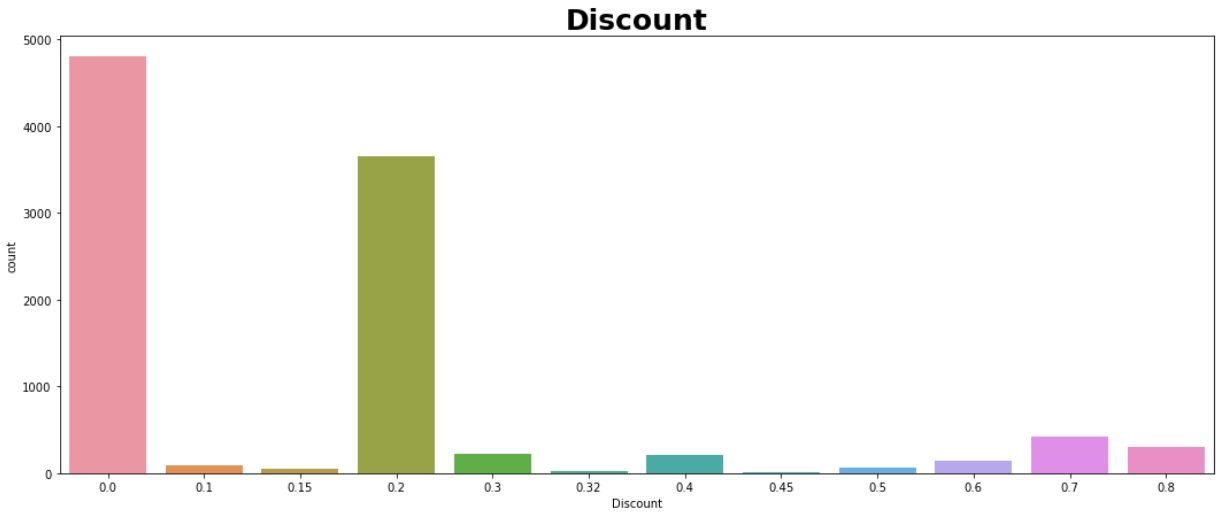
Out[14]: Text(0.5, 1.0, 'Quantity')



In [15]:
```
plt.figure(figsize=(18,7))
sns.countplot(Data['Discount'])
plt.title('Discount',fontsize=25,fontweight='bold')
```

```
C:\Users\admin\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning:
Pass the following variable as a keyword arg: x. From version 0.12, the only valid p
ositional argument will be `data`, and passing other arguments without an explicit k
eyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[15]:
```
Text(0.5, 1.0, 'Discount')
```



In [16]:
```
# Destribution Of Data Using Plot
fig, axs = plt.subplots(ncols=2, nrows = 2, figsize = (10,10))
sns.distplot(Data['Sales'], color = 'red',  ax = axs[0][0])
sns.distplot(Data['Profit'], color = 'green',  ax = axs[0][1])
sns.distplot(Data['Quantity'], color = 'orange',  ax = axs[1][0])
sns.distplot(Data['Discount'], color = 'blue',  ax = axs[1][1])
axs[0][0].set_title('Sales Distribution', fontsize = 20)
axs[0][1].set_title('Profit Distribution', fontsize = 20)
axs[1][0].set_title('Quantity distribution', fontsize = 20)
axs[1][1].set_title('Discount Distribution', fontsize = 20)
plt.show()
```

```
C:\Users\admin\anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarn
ing: `distplot` is a deprecated function and will be removed in a future version. Pl
ease adapt your code to use either `displot` (a figure-level function with similar f
lexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
C:\Users\admin\anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarn
ing: `distplot` is a deprecated function and will be removed in a future version. Pl
ease adapt your code to use either `displot` (a figure-level function with similar f
lexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

```
C:\Users\admin\anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarn
ing: `distplot` is a deprecated function and will be removed in a future version. Pl
ease adapt your code to use either `displot` (a figure-level function with similar f
lexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
C:\Users\admin\anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarn
ing: `distplot` is a deprecated function and will be removed in a future version. Pl
ease adapt your code to use either `displot` (a figure-level function with similar f
lexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

In [17]:

```
# State Wise Data Analysis
Data_1=Data['State'].value_counts()
Data_1
```

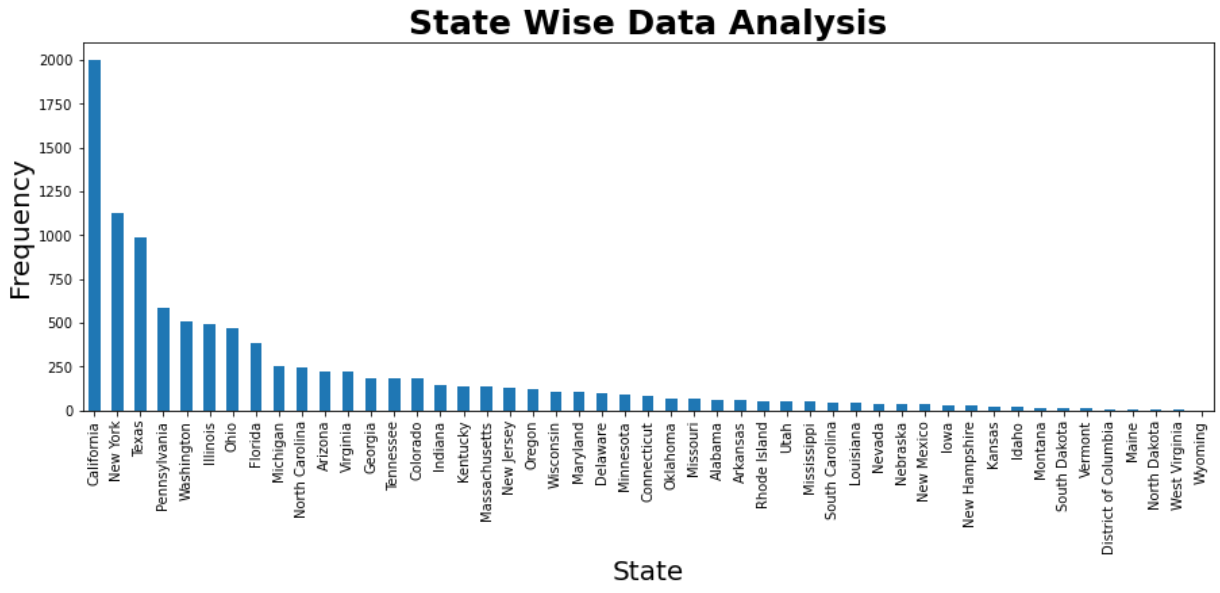
Out[17]:

```
California              2001
New York                1128
Texas                    985
Pennsylvania             587
Washington               506
Illinois                 492
Ohio                     469
Florida                  383
Michigan                 255
North Carolina           249
Arizona                  224
Virginia                 224
Georgia                  184
```

```
Tennessee                 183
Colorado                  182
Indiana                   149
Kentucky                  139
Massachusetts             135
New Jersey                130
Oregon                    124
Wisconsin                 110
Maryland                  105
Delaware                   96
Minnesota                  89
Connecticut                82
Oklahoma                   66
Missouri                   66
Alabama                    61
Arkansas                   60
Rhode Island               56
Utah                       53
Mississippi                53
South Carolina             42
Louisiana                  42
Nevada                     39
Nebraska                   38
New Mexico                 37
Iowa                       30
New Hampshire              27
Kansas                     24
Idaho                      21
Montana                    15
South Dakota               12
Vermont                    11
District of Columbia       10
Maine                       8
North Dakota                7
West Virginia               4
Wyoming                     1
Name: State, dtype: int64
```

In [18]:

```
Data_1.plot(kind='bar',figsize=(15,5))
plt.title('State Wise Data Analysis',fontsize=25,fontweight='bold')
plt.xlabel('State',fontsize=20)
plt.ylabel('Frequency',fontsize=20)
plt.show()
```



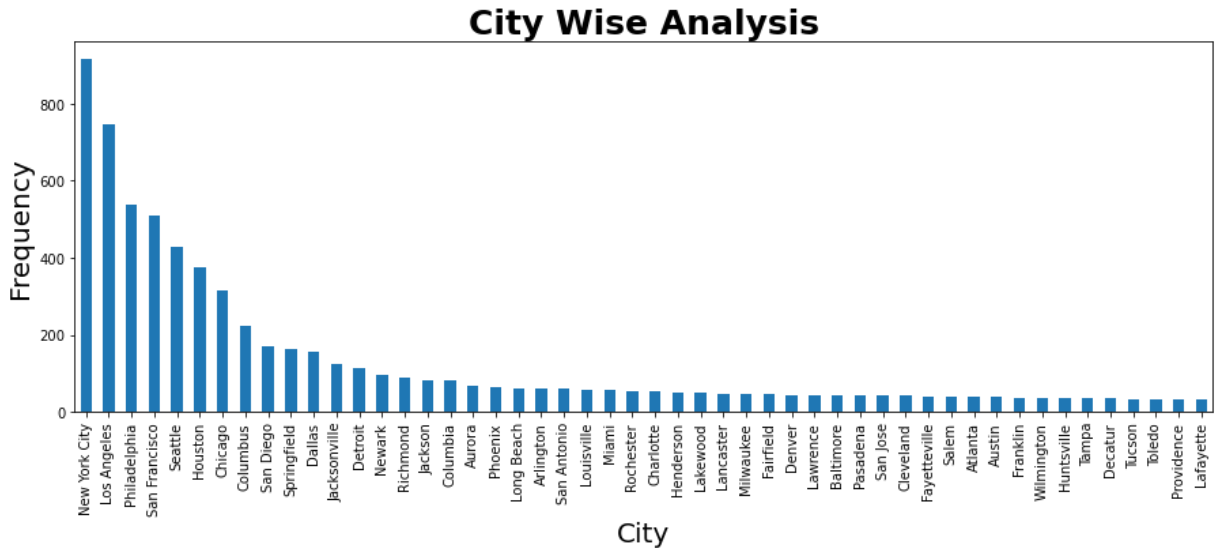
Here is top 3 states California,New

York,Texas where Deals are highest

In [19]:
```
# Top 50 city Wise analysis
Data_2=Data['City'].value_counts()
Data_2.head(50)
```

Out[19]:
```
New York City    915
Los Angeles      747
Philadelphia     537
San Francisco    510
Seattle          428
Houston          377
Chicago          314
Columbus         222
San Diego        170
Springfield      163
Dallas           157
Jacksonville     125
Detroit          115
Newark            95
Richmond          90
Jackson           82
Columbia          81
Aurora            68
Phoenix           63
Long Beach        61
Arlington         60
San Antonio       59
Louisville        57
Miami             57
Rochester         53
Charlotte         52
Henderson         51
Lakewood          49
Lancaster         46
Milwaukee         45
Fairfield         45
Denver            44
Lawrence          44
Baltimore         43
Pasadena          42
San Jose          42
Cleveland         42
Fayetteville      41
Salem             40
Atlanta           39
Austin            39
Franklin          37
Wilmington        36
Huntsville        36
Tampa             36
Decatur           35
Tucson            32
Toledo            32
Providence        31
Lafayette         31
Name: City, dtype: int64
```

In [20]:
```
Data_2.head(50).plot(kind='bar',figsize=(15,5))
plt.title('City Wise Analysis',fontsize=25,fontweight='bold')
plt.xlabel('City',fontsize=20)
plt.ylabel('Frequency',fontsize=20)
plt.show()
```



Here New York City, Los Angeles, Philadelphia are the cities where dealing is high

In [21]:
```
# Segment Wise Analysis of Sell, Profit and Discount
Data_segment=Data.groupby(['Segment'])[['Sales','Profit','Discount']].mean()
Data_segment
```

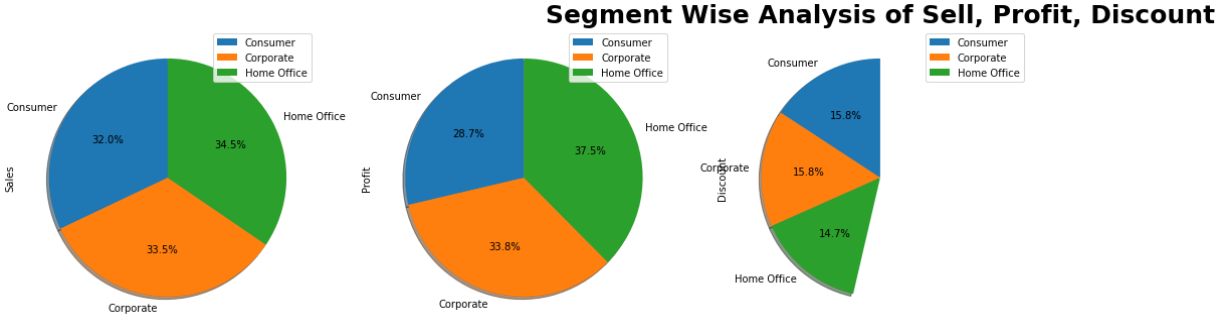
Out[21]:

Segment	Sales	Profit	Discount
Consumer	223.733644	25.836873	0.158141
Corporate	233.823300	30.456667	0.158228
Home Office	240.972041	33.818664	0.147128

In [22]:
```
Data_segment.plot.pie(subplots=True,autopct='%1.1f%%',shadow=True,figsize=(18,20),st
plt.title('Segment Wise Analysis of Sell, Profit, Discount',fontsize=25,fontweight='
```

```
C:\Users\admin\anaconda3\lib\site-packages\pandas\plotting\_matplotlib\core.py:1583:
MatplotlibDeprecationWarning: normalize=None does not normalize if the sum is less t
han 1 but this behavior is deprecated since 3.3 until two minor releases later. Afte
r the deprecation period the default value will be normalize=True. To prevent normal
ization pass normalize=False
  results = ax.pie(y, labels=blabels, **kwds)
```

Out[22]: Text(0.5, 1.0, 'Segment Wise Analysis of Sell, Profit, Discount')



Sales:

1) Consumer=32.0%

2) Corporate=33.5%

3) Home Office=34.5%

Profit:

1) Consumer=28.7%

2) Corporate=33.8%

3) Home Office=37.5%

Discount:

1)Consumer=15.8%

2)Corporate=15.8%

3)Home Office=14.7%

In [23]:
```
#State wise Analysis of Profit, Discount, Sale.
Data_state=Data.groupby(['State'])[['Sales','Profit','Discount']].mean()
Data_state
```

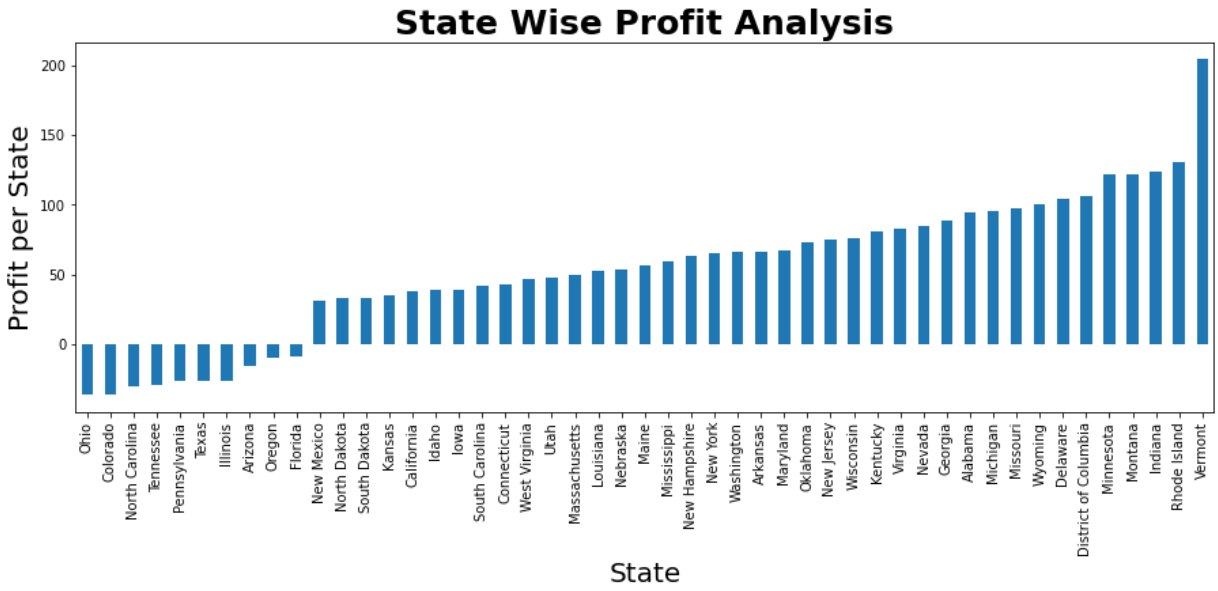
Out[23]:

State	Sales	Profit	Discount
Alabama	319.846557	94.865989	0.000000
Arizona	157.508933	-15.303235	0.303571
Arkansas	194.635500	66.811452	0.000000
California	228.729451	38.171608	0.072764
Colorado	176.418231	-35.867351	0.316484
Connecticut	163.223866	42.823071	0.007317
Delaware	285.948635	103.930988	0.006250
District of Columbia	286.502000	105.958930	0.000000
Florida	233.612815	-8.875461	0.299347
Georgia	266.825217	88.315453	0.000000
Idaho	208.689810	39.367767	0.085714
Illinois	162.939230	-25.625787	0.390041
Indiana	359.431946	123.375411	0.000000
Iowa	152.658667	39.460397	0.000000
Kansas	121.429583	34.851813	0.000000
Kentucky	263.250000	80.573357	0.000000

State	Sales	Profit	Discount
Louisiana	219.453095	52.288150	0.000000
Maine	158.816250	56.810775	0.000000
Maryland	225.766886	66.963608	0.005714
Massachusetts	212.106919	50.262975	0.015556
Michigan	299.096525	95.934069	0.007059
Minnesota	335.541011	121.608847	0.000000
Mississippi	203.232830	59.867475	0.000000
Missouri	336.441667	97.518341	0.000000
Montana	372.623467	122.221900	0.066667
Nebraska	196.445526	53.607742	0.000000
Nevada	428.951333	85.045279	0.061538
New Hampshire	270.093481	63.203807	0.011111
New Jersey	275.110092	75.176260	0.004615
New Mexico	129.284378	31.273408	0.059459
New York	275.599531	65.637011	0.055319
North Carolina	223.305880	-30.083985	0.283534
North Dakota	131.415714	32.878529	0.000000
Ohio	166.861697	-36.186304	0.324947
Oklahoma	298.233182	73.544788	0.000000
Oregon	140.573790	-9.600569	0.288710
Pennsylvania	198.487077	-26.507598	0.328620
Rhode Island	404.070643	130.100523	0.021429
South Carolina	201.945476	42.120395	0.000000
South Dakota	109.630000	32.902358	0.000000
Tennessee	167.551219	-29.189583	0.291257
Texas	172.779742	-26.121174	0.370193
Utah	211.699170	48.047802	0.060377
Vermont	811.760909	204.088936	0.000000
Virginia	315.342500	83.026564	0.000000
Washington	273.994605	66.013146	0.064032
West Virginia	302.456000	46.480400	0.075000
Wisconsin	291.951000	76.380004	0.000000
Wyoming	1603.136000	100.196000	0.200000

In [24]:
```
#1) State wise profit analysis
```

```
Data_state_1=Data_state.sort_values(['Profit'])
Data_state_1['Profit'].plot(kind='bar',figsize=(15,5))
plt.xlabel('State',fontsize=20)
plt.ylabel('Profit per State',fontsize=20)
plt.title('State Wise Profit Analysis',fontsize=25,fontweight='bold')
plt.show()
```



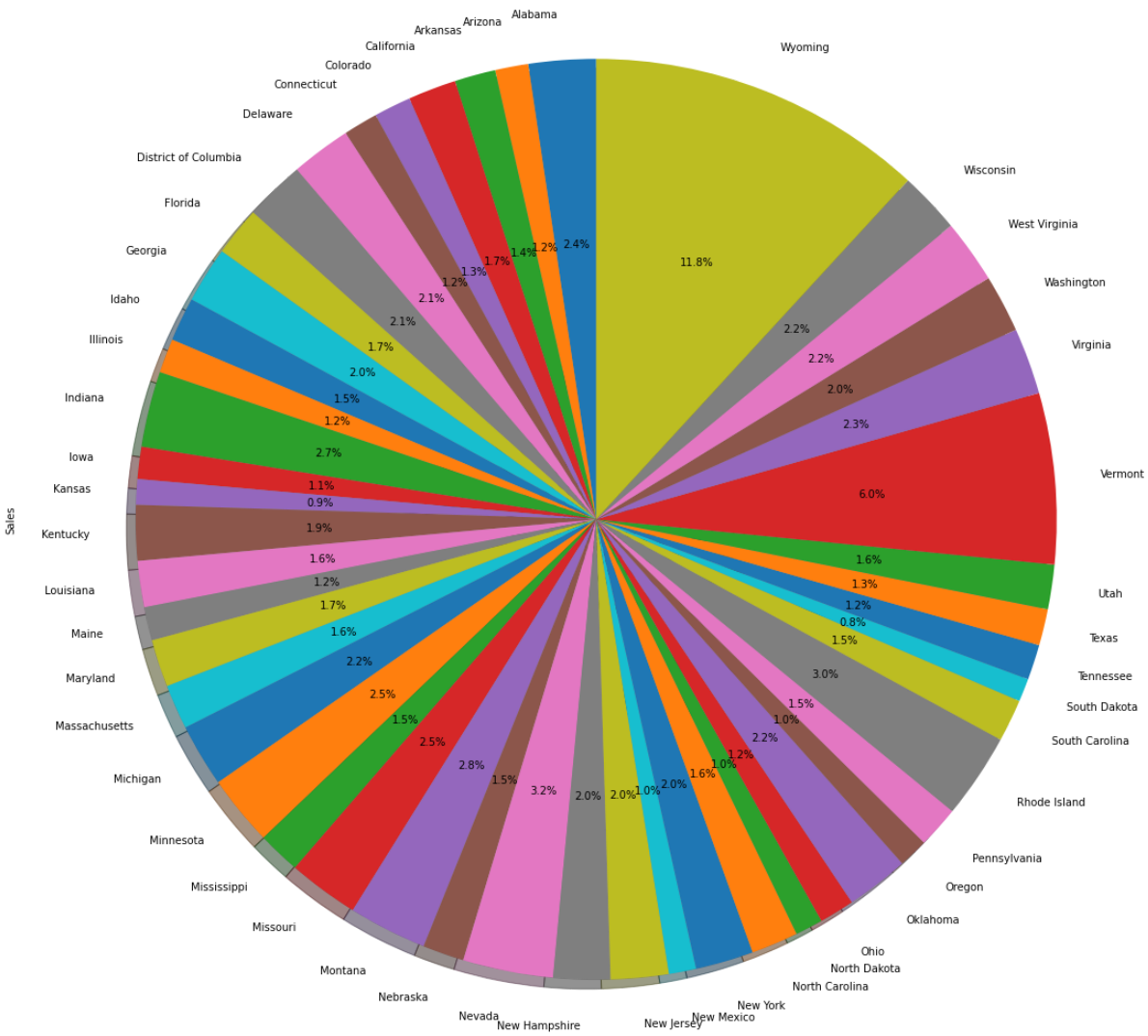
Result:

Highest Profit=vermont

Lowest Profit=Ohio

In [25]:

```
#2) State Wise Sales Analysis
Data_state['Sales'].plot(kind='pie',autopct='%1.1f%%',figsize=(20,20),startangle=90,
plt.title('State Wise Sales Analysis',fontsize=25,fontweight='bold')
plt.show()
```



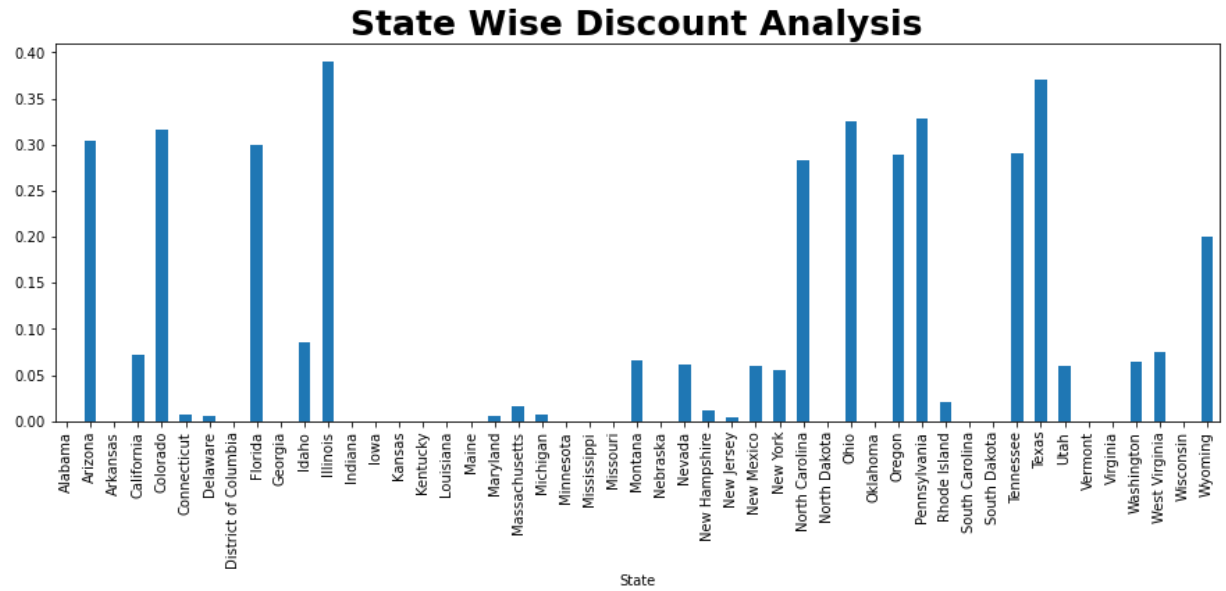
Result :

Highest Amount Of Sale= Wyoming(11.8%)

Lowest Amount Of Sale= South Dakota(0.8%)

In [26]:
```
# State Wise Discount Analysis
Data_state['Discount'].plot(kind='bar',figsize=(15,5))
plt.title('State Wise Discount Analysis',fontsize=25,fontweight='bold')
```

Out[26]:
```
Text(0.5, 1.0, 'State Wise Discount Analysis')
```



In [27]:

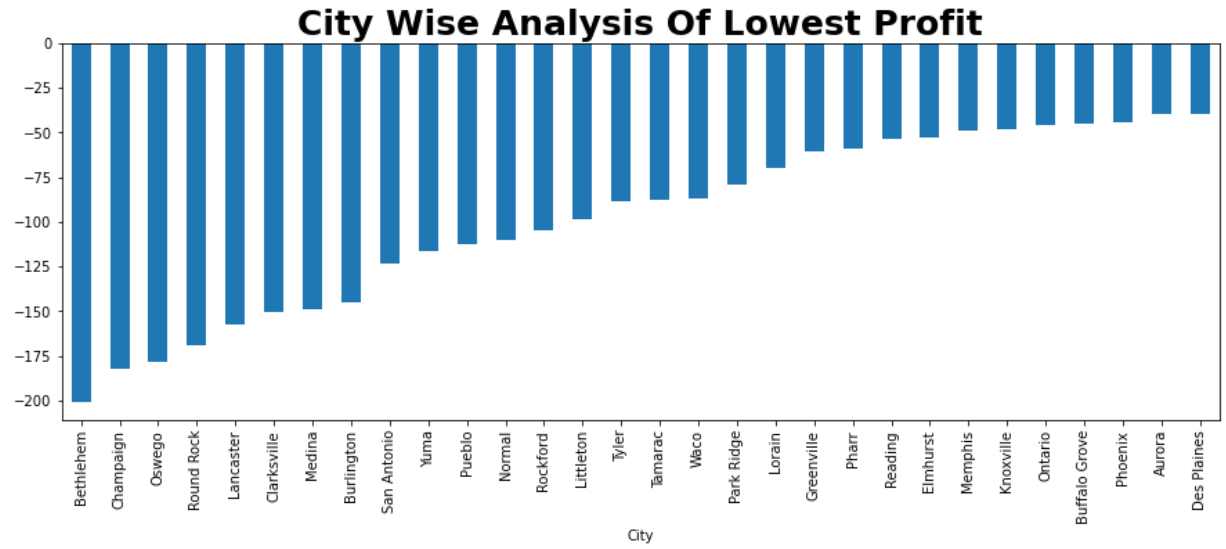
```
# City Wise Analysis Of Profit, Sales, Discount
Data_city=Data.groupby(['City'])[['Sales','Profit','Discount']].mean()
Data_city=Data_city.sort_values('Profit')
Data_city.head()
```

Out[27]:

City	Sales	Profit	Discount
Bethlehem	337.926800	-200.619160	0.380000
Champaign	151.960000	-182.352000	0.600000
Oswego	107.326000	-178.709200	0.600000
Round Rock	693.436114	-169.061614	0.274286
Lancaster	215.031826	-157.371052	0.315217

In [28]:

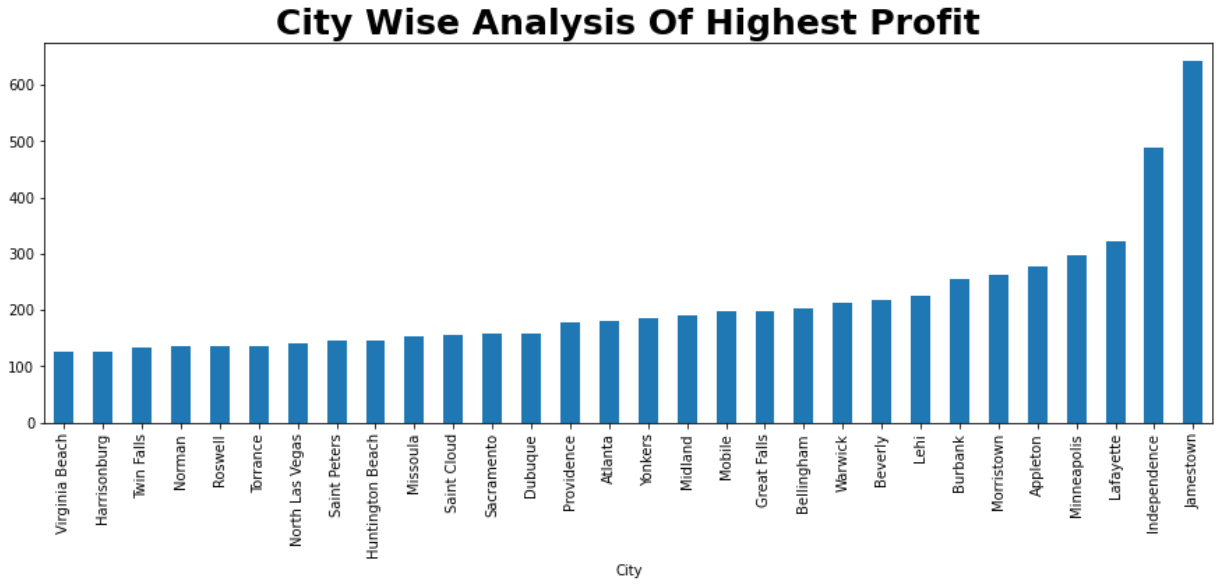
```
#1) City wise analysis of profit
# Low Profit
Data_city['Profit'].head(30).plot(kind='bar',figsize=(15,5))
plt.title('City Wise Analysis Of Lowest Profit',fontsize=25,fontweight='bold')
plt.show()
```



Result:

Bethlehem city has low profit

In [29]:
```
# High Profit
Data_city['Profit'].tail(30).plot(kind='bar',figsize=(15,5))
plt.title('City Wise Analysis Of Highest Profit',fontsize=25,fontweight='bold')
plt.show()
```



Result:

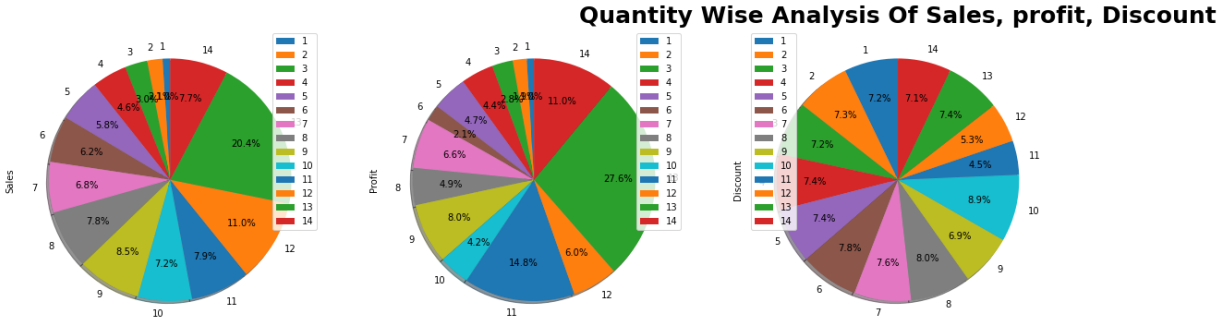
Jamestown city has highest profit

In [30]:
```
# Quantity Wise Analysis Of Sales, Profit, Discount
Data_Quantity=Data.groupby(['Quantity'])[['Sales','Profit','Discount']].mean()
Data_Quantity.head(5)
```

Out[30]:

Quantity	Sales	Profit	Discount
1	59.234632	8.276396	0.152959
2	120.354488	16.006831	0.154858
3	175.201578	23.667715	0.153329
4	271.764059	37.131310	0.157708
5	337.936339	40.257394	0.157146

In [31]:
```
Data_Quantity.plot.pie(subplots=True,shadow=True,autopct='%1.1f%%',startangle=90,pct
                       figsize=(20,20))
plt.title('Quantity Wise Analysis Of Sales, profit, Discount',fontsize=25,fontweight
plt.show()
```



Result:

Quantity 13 has Highest sales and Profit

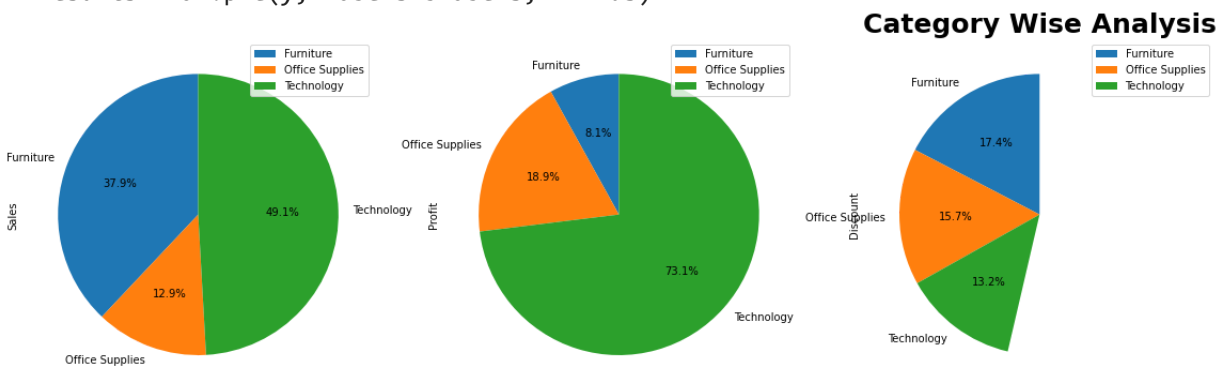
In [32]:
```
# Category Wise Profit, Sales and Discount Analysis
Data_Category=Data.groupby(['Category'])[['Sales','Profit','Discount']].mean()
Data_Category
```

Out[32]:

Category	Sales	Profit	Discount
Furniture	349.834887	8.699327	0.173923
Office Supplies	119.324101	20.327050	0.157285
Technology	452.709276	78.752002	0.132323

In [33]:
```
Data_Category.plot.pie(subplots=True,autopct='%1.1f%%',startangle=90,figsize=(20,20))
plt.title('Category Wise Analysis',fontsize=25,fontweight='bold')
plt.show()
```

```
C:\Users\admin\anaconda3\lib\site-packages\pandas\plotting\_matplotlib\core.py:1583:
MatplotlibDeprecationWarning: normalize=None does not normalize if the sum is less t
han 1 but this behavior is deprecated since 3.3 until two minor releases later. Afte
r the deprecation period the default value will be normalize=True. To prevent normal
ization pass normalize=False
  results = ax.pie(y, labels=blabels, **kwds)
```



Result:

Maximum profit and sales obtained in Technology.

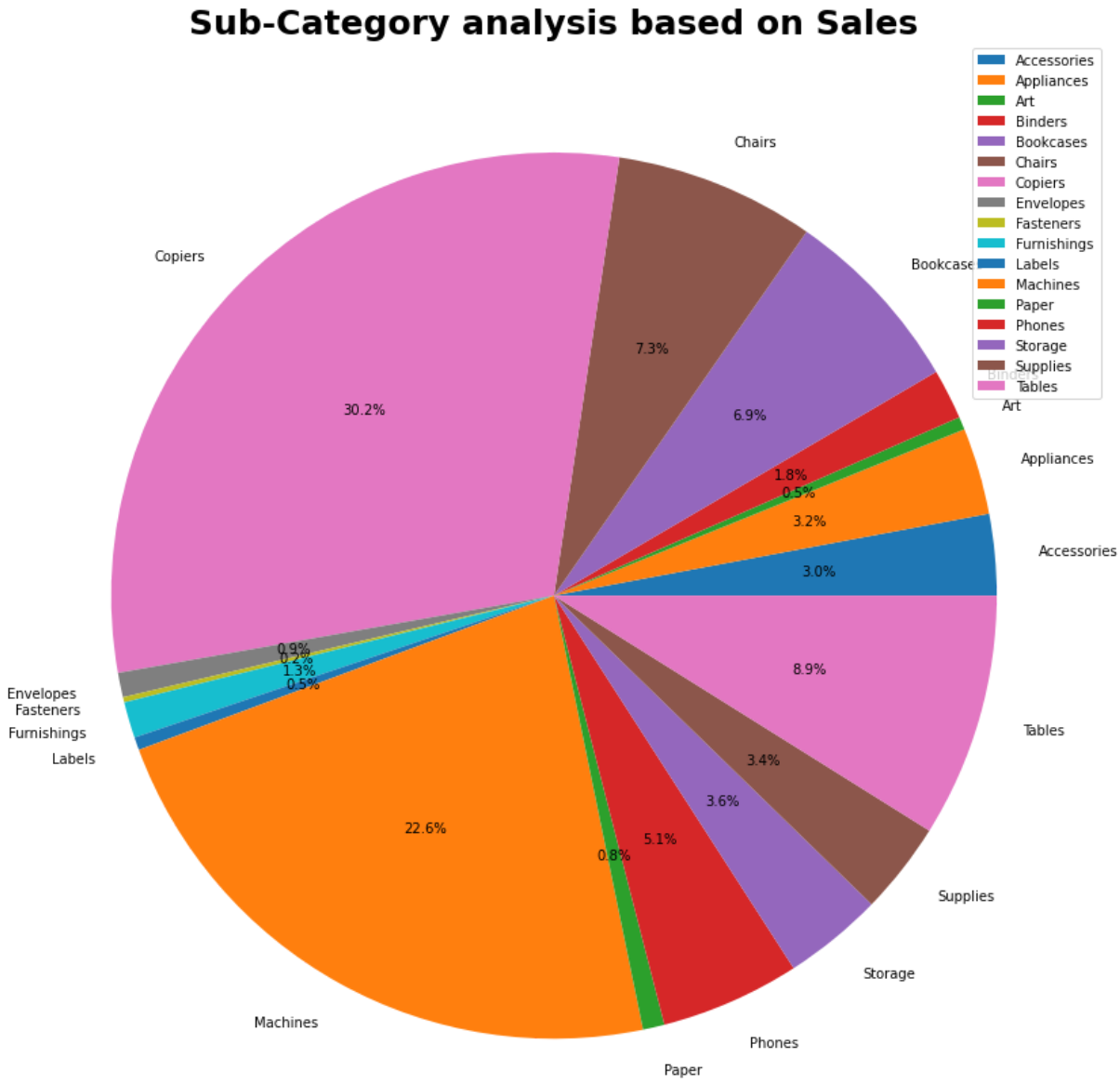
Minimum profit obtained in Furniture.

In [34]:
```
# Sub-Category Wise Profit, Sales, Discount Analysis
Data_sub_category=Data.groupby(['Sub-Category'])[['Sales','Profit','Discount']].mean
Data_sub_category.head(10)
```

Out[34]:

Sub-Category	Sales	Profit	Discount
Accessories	215.974604	54.111788	0.078452
Appliances	230.755710	38.922758	0.166524
Art	34.068834	8.200737	0.074874
Binders	133.560560	19.843574	0.372292
Bookcases	503.859633	-15.230509	0.211140
Chairs	532.332420	43.095894	0.170178
Copiers	2198.941618	817.909190	0.161765
Envelopes	64.867724	27.418019	0.080315
Fasteners	13.936774	4.375660	0.082028
Furnishings	95.825668	13.645918	0.138349

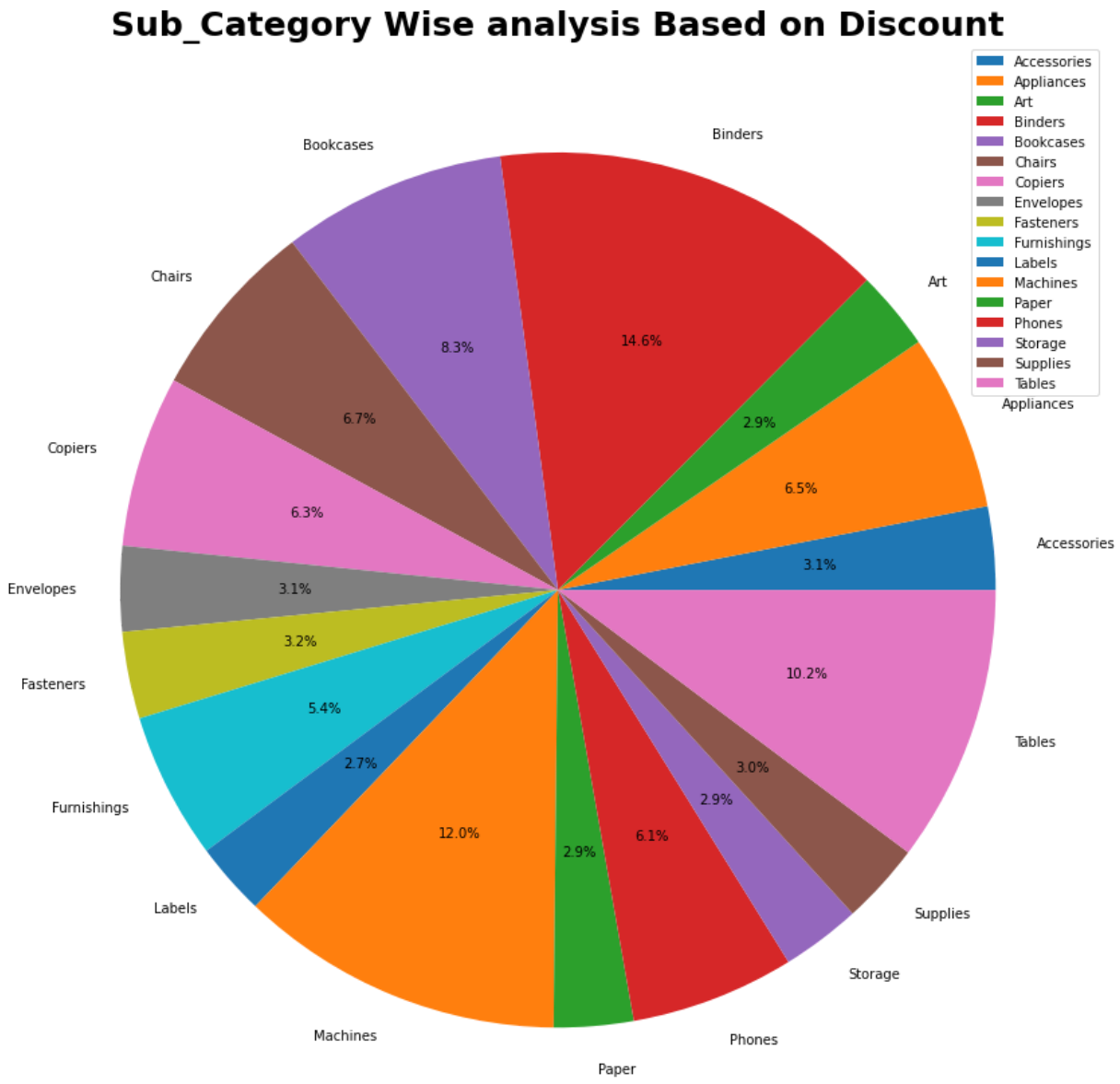
In [35]:
```
# 1) Bases on Sales
plt.figure(figsize=(15,15))
plt.pie(Data_sub_category['Sales'],autopct='%1.1f%%',labels=Data_sub_category.index)
plt.title('Sub-Category analysis based on Sales',fontsize=25,fontweight='bold')
plt.legend()
plt.xticks(rotation=90)
plt.show()
```



Result:

Copiers have highest sales

In [36]:
```
# 2) Basis on Discount
plt.figure(figsize=(15,15))
plt.pie(Data_sub_category['Discount'],autopct='%1.1f%%',labels=Data_sub_category.ind
plt.title('Sub_Category Wise analysis Based on Discount',fontsize=25,fontweight='bol
plt.legend()
plt.xticks(rotation=90)
plt.show()
```

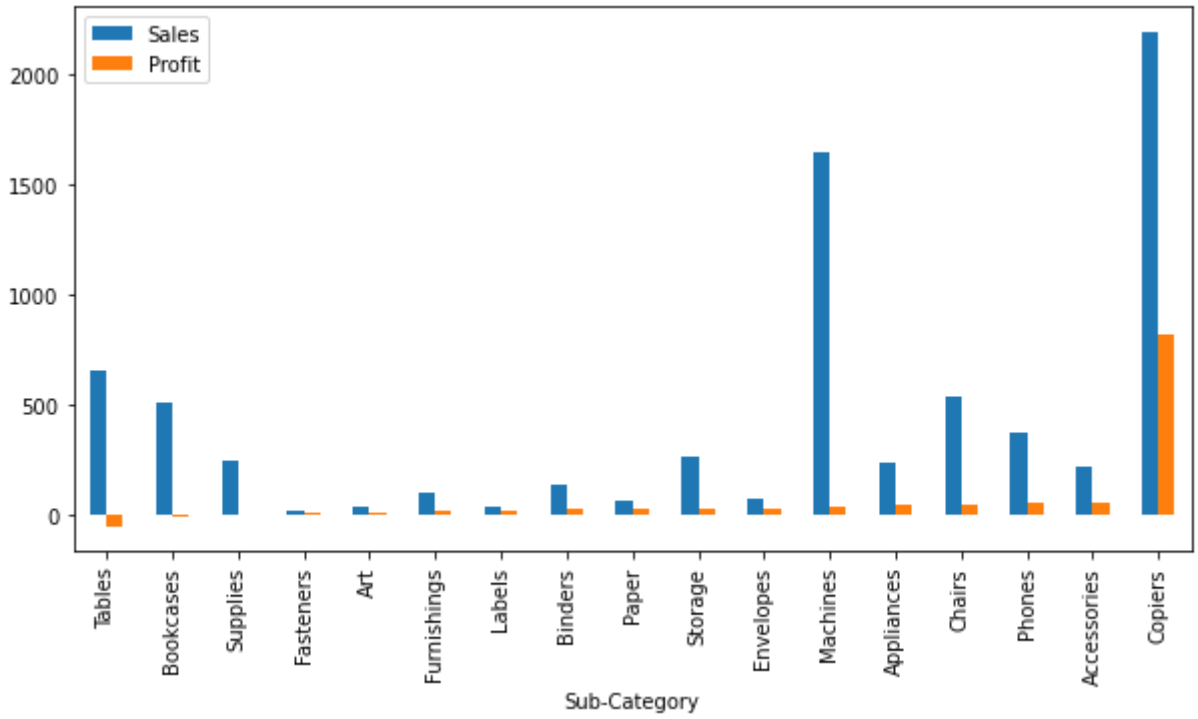


Result:

Machines Blinders and tables have highest Discount

In [37]:
```
# 3) Based on Profit
Data_sub_category.sort_values('Profit')[['Sales','Profit']].plot(kind='bar',figsize=
                                                                 label=['Avg Sales Pric
```

Out[37]: <AxesSubplot:xlabel='Sub-Category'>



Result:

Copier has highest Profit and Sales

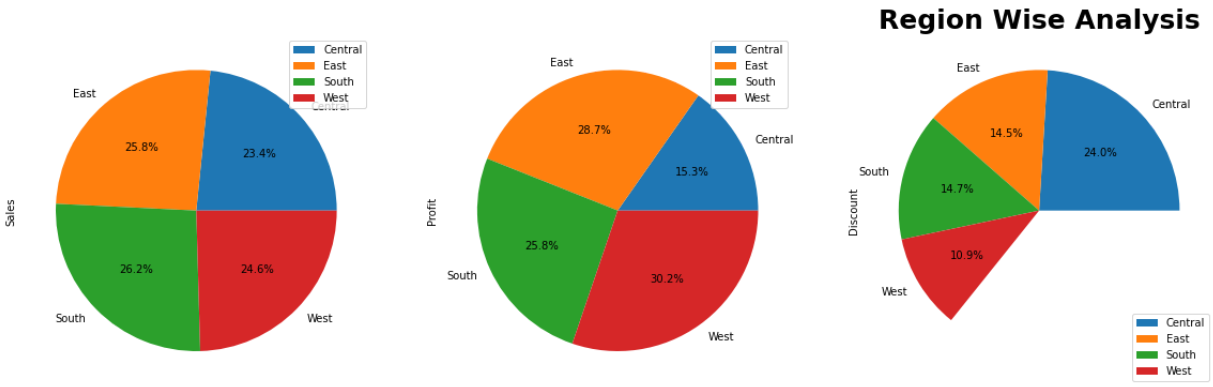
In [38]:
```
# Region Wise Analysis
Data_region=Data.groupby(['Region'])[['Sales','Profit','Discount']].mean()
Data_region
```

Out[38]:

Region	Sales	Profit	Discount
Central	215.772661	17.092709	0.240353
East	238.336110	32.135808	0.145365
South	241.803645	28.857673	0.147253
West	226.493233	33.849032	0.109335

In [39]:
```
Data_region.plot.pie(subplots=True,autopct='%1.1f%%',labels=Data_region.index,figsiz
plt.legend()
plt.title('Region Wise Analysis',fontsize=25,fontweight='bold')
plt.show()
```

```
C:\Users\admin\anaconda3\lib\site-packages\pandas\plotting\_matplotlib\core.py:1583:
MatplotlibDeprecationWarning: normalize=None does not normalize if the sum is less t
han 1 but this behavior is deprecated since 3.3 until two minor releases later. Afte
r the deprecation period the default value will be normalize=True. To prevent normal
ization pass normalize=False
  results = ax.pie(y, labels=blabels, **kwds)
```



Result:

In West region profit is high

In South region sales is high

In Central region discount is high

In [40]:
```
# Ship Mode Wise Analysis
Data['Ship Mode'].value_counts()
```

Out[40]:
```
Standard Class    5968
Second Class      1945
First Class       1538
Same Day           543
Name: Ship Mode, dtype: int64
```

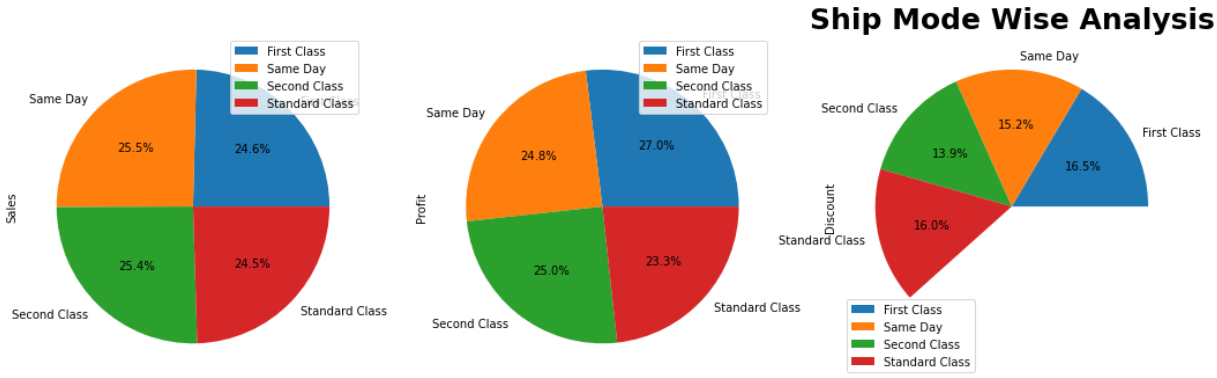
In [41]:
```
 Data_ship_mode=Data.groupby(['Ship Mode'])[['Sales','Profit','Discount']].mean()
Data_ship_mode
```

Out[41]:

Ship Mode	Sales	Profit	Discount
First Class	228.497024	31.839948	0.164610
Same Day	236.396179	29.266591	0.152394
Second Class	236.089239	29.535545	0.138895
Standard Class	227.583067	27.494770	0.160023

In [42]:
```
Data_ship_mode.plot.pie(subplots=True,autopct='%1.1f%%',figsize=(18,20),labels=Data_
plt.title('Ship Mode Wise Analysis',fontsize=25,fontweight='bold')
plt.legend()
plt.show()
```

```
C:\Users\admin\anaconda3\lib\site-packages\pandas\plotting\_matplotlib\core.py:1583:
MatplotlibDeprecationWarning: normalize=None does not normalize if the sum is less t
han 1 but this behavior is deprecated since 3.3 until two minor releases later. Afte
r the deprecation period the default value will be normalize=True. To prevent normal
ization pass normalize=False
  results = ax.pie(y, labels=blabels, **kwds)
```



Result:

Profit and Discount is high in First Class.

Sales is high for Same Day.

In [ ]: