Author : BEHARA VAISHNAVI

GRIP @ The Sparks Foundation Task 3 : Exploratory Data Analysis - Retail Here I have created the analysis model of a Super market datasheet given by the company and have deployed the parameters successfully.

AIM : Perform 'Exploratory Data Analysis' on dataset 'SampleSuperstore'

Technical Stack : Numpy, Pandas, Matplotlib, Seaborn

In [1]:
```
# Importing the Required Libraries
%matplotlib inline
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

Reading the Data from Source

In [2]:
```
# Reading data from Remote Link
url = r"https://raw.githubusercontent.com/vaishnavibehara/supermaretdataset/main/Sample
df = pd.read_csv(url)
df.head()
```

Out[2]:

	Ship Mode	Segment	Country	City	State	Postal Code	Region	Category	Sub-Category	Sales
0	Second Class	Consumer	United States	Henderson	Kentucky	42420	South	Furniture	Bookcases	261.9600
1	Second Class	Consumer	United States	Henderson	Kentucky	42420	South	Furniture	Chairs	731.9400
2	Second Class	Corporate	United States	Los Angeles	California	90036	West	Office Supplies	Labels	14.6200
3	Standard Class	Consumer	United States	Fort Lauderdale	Florida	33311	South	Furniture	Tables	957.5775
4	Standard Class	Consumer	United States	Fort Lauderdale	Florida	33311	South	Office Supplies	Storage	22.3680

In [3]:
```
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
```

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

Out[4]:
```
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 [5]:
```
df.columns
```

Out[5]:
```
Index(['Ship Mode', 'Segment', 'Country', 'City', 'State', 'Postal Code',
       'Region', 'Category', 'Sub-Category', 'Sales', 'Quantity', 'Discount',
       'Profit'],
      dtype='object')
```

In [6]:
```
df.shape
```

Out[6]:
```
(9994, 13)
```

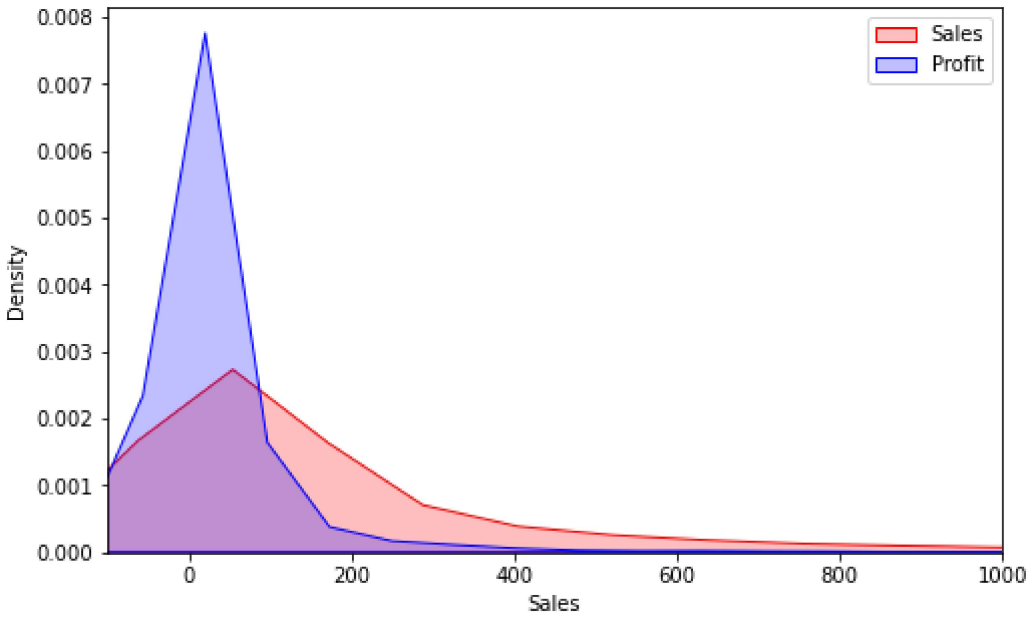
In [7]:
```
df.nunique()
```

Out[7]:
```
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
```

Exploratory Data Analysis

In [11]:
```
plt.figure(figsize=(8,5))
sns.kdeplot(df['Sales'],color='red',label='Sales',shade=True)
sns.kdeplot(df['Profit'],color='Blue',label='Profit',shade=True)
plt.xlim([-100,1000])
plt.legend()
```

Out[11]: <matplotlib.legend.Legend at 0x176d6dd9f10>



RESULT

Profit is more than that of sale but there are some areas where profit could be increased.

Analysis Using Pairplot of Each Column

[1] Based on the Category

In [13]:
```
sns.pairplot(df,hue='Category')
```

Out[13]: <seaborn.axisgrid.PairGrid at 0x176d6e054f0>

[2] Based on the Region

In [15]:
```
sns.pairplot(df,hue='Region')
```

Out[15]:
```
<seaborn.axisgrid.PairGrid at 0x176d9dd9970>
```

[3] Based on the Segment

In [16]:
```
sns.pairplot(df,hue='Segment')
```

Out[16]:
```
<seaborn.axisgrid.PairGrid at 0x176dc966970>
```

Finding Correlation

In [17]:
```
df.corr()
```

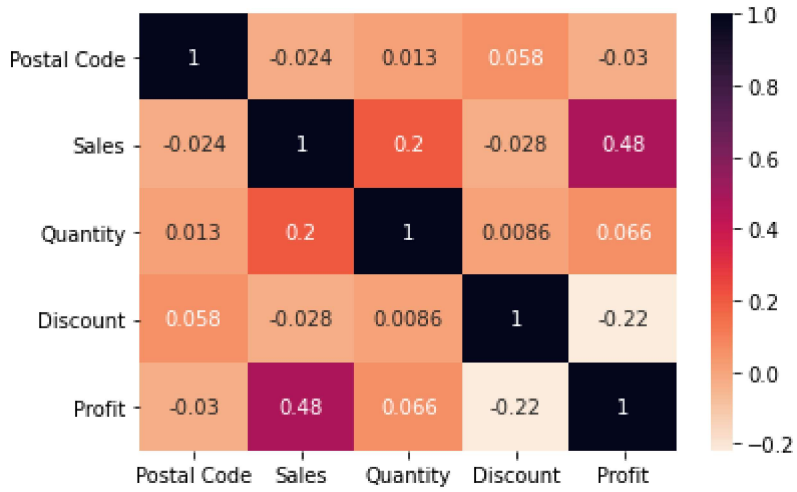
Out[17]:

	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

Heatmap for Correlation

In [18]:
```
sns.heatmap(df.corr(),cmap='rocket_r',annot=True)
```

Out[18]: <AxesSubplot:>



RESULT

From the above Heatmap, It's Concluded that

Sales and Profit are Moderately Correlated.

Discount and Profit are Negatively Correlated.

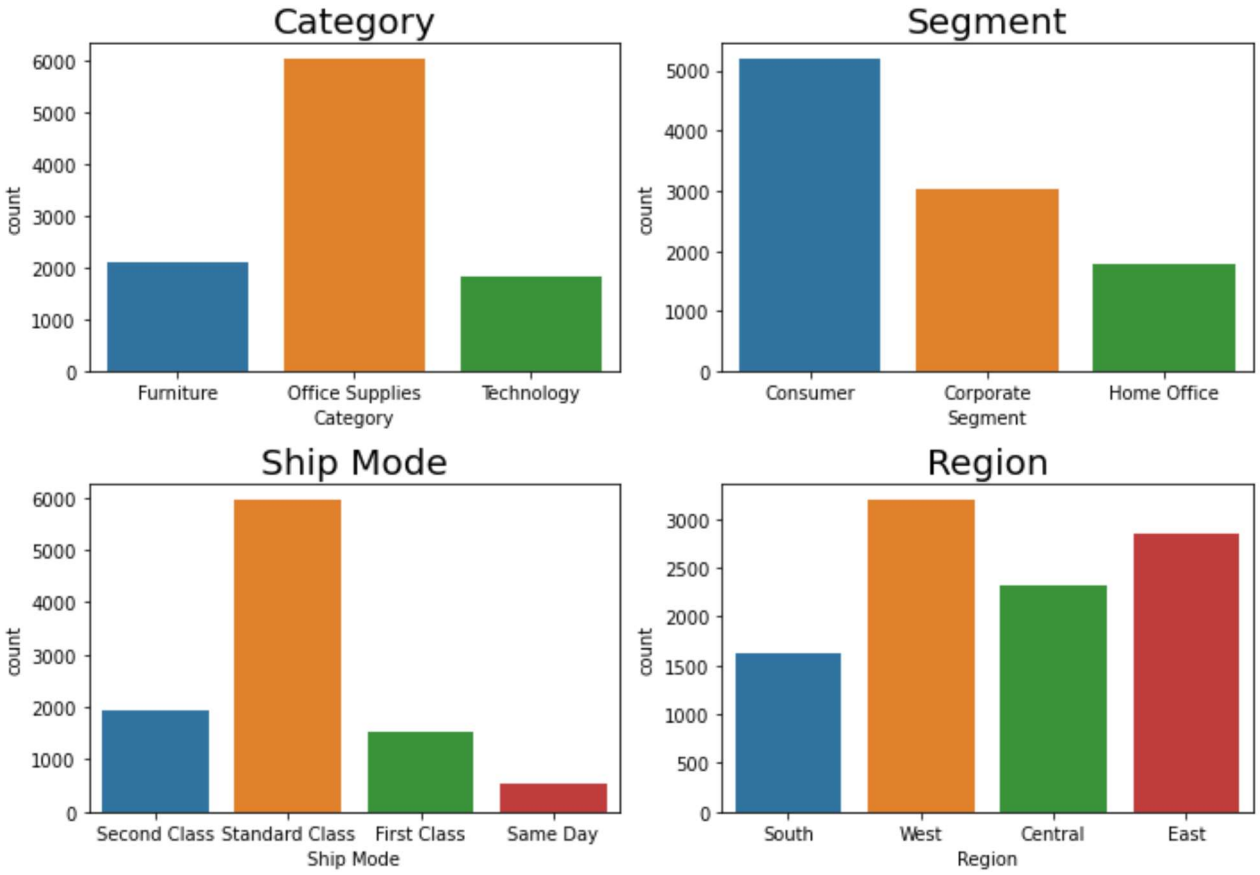
Quantity and Profit are less Moderately Correlated.

Analysis Using Countplot of Each Column

In [21]:
```
fig,axs=plt.subplots(nrows=2,ncols=2,figsize=(10,7));

sns.countplot(df['Category'],ax=axs[0][0])
sns.countplot(df['Segment'],ax=axs[0][1])
sns.countplot(df['Ship Mode'],ax=axs[1][0])
sns.countplot(df['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\VAISHNAVI\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 posit
ional argument will be `data`, and passing other arguments without an explicit keyword w
ill result in an error or misinterpretation.
  warnings.warn(
C:\Users\VAISHNAVI\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 posit
ional argument will be `data`, and passing other arguments without an explicit keyword w
ill result in an error or misinterpretation.
  warnings.warn(
C:\Users\VAISHNAVI\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 posit
```

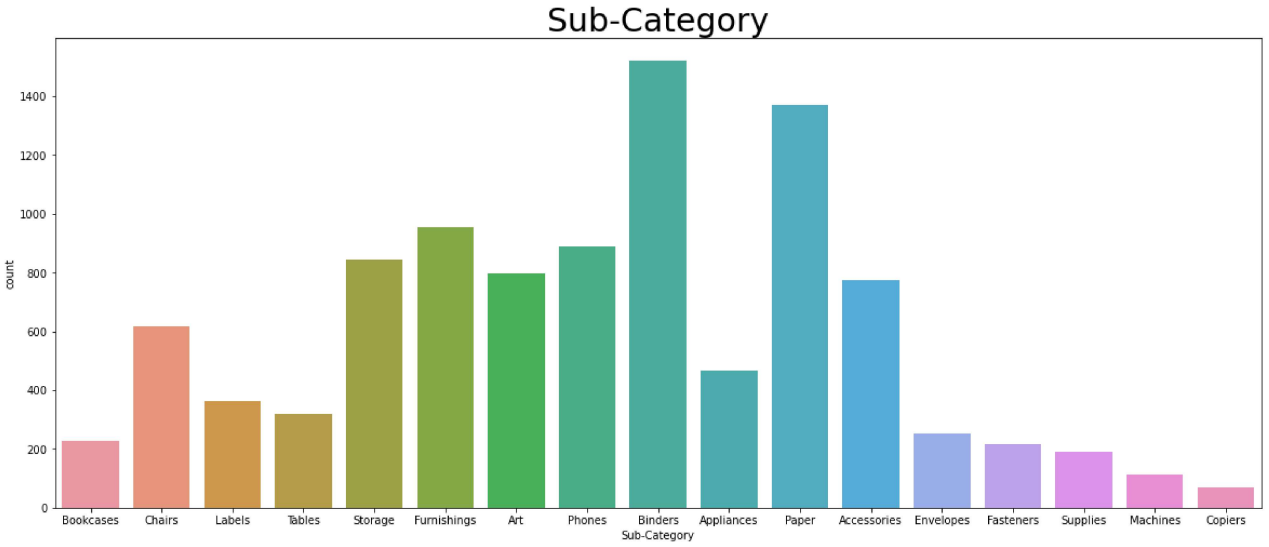
```
ional argument will be `data`, and passing other arguments without an explicit keyword w
ill result in an error or misinterpretation.
  warnings.warn(
C:\Users\VAISHNAVI\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning:
Pass the following variable as a keyword arg: x. From version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



In [22]:
```
plt.figure(figsize=(20,8))
sns.countplot(df['Sub-Category'])
plt.title('Sub-Category',fontsize=30)
```

```
C:\Users\VAISHNAVI\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 posit
ional argument will be `data`, and passing other arguments without an explicit keyword w
ill result in an error or misinterpretation.
  warnings.warn(
```

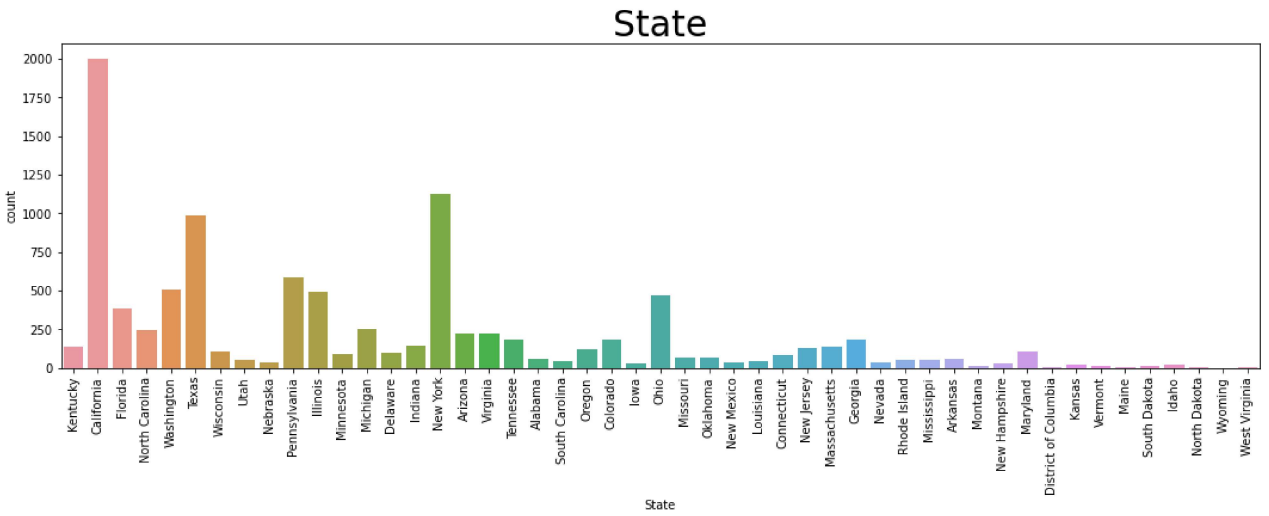
Out[22]:
```
Text(0.5, 1.0, 'Sub-Category')
```



In [23]:
```
plt.figure(figsize=(18,5))
sns.countplot(df['State'])
plt.xticks(rotation=90)
plt.title('State',fontsize=30)
```

```
C:\Users\VAISHNAVI\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 posit
ional argument will be `data`, and passing other arguments without an explicit keyword w
ill result in an error or misinterpretation.
  warnings.warn(
```

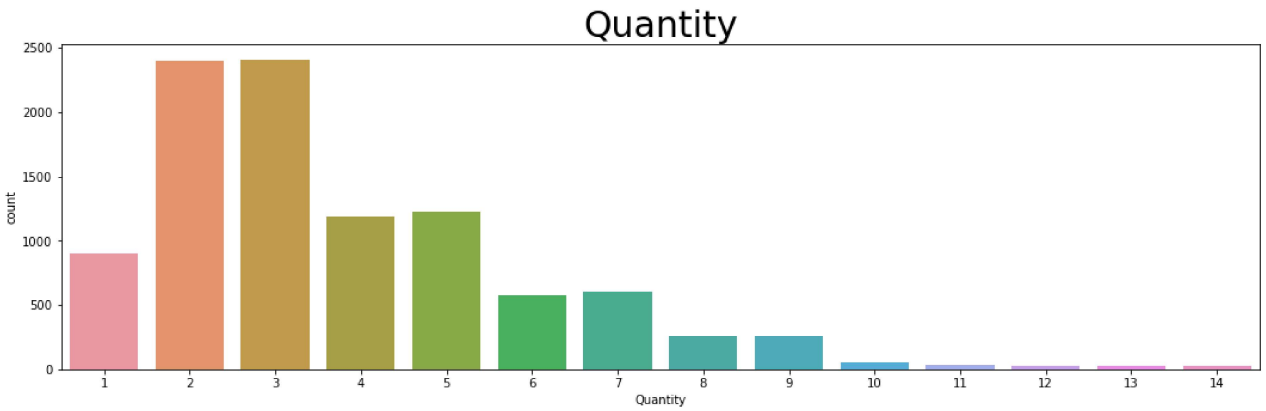
Out[23]: Text(0.5, 1.0, 'State')



In [24]:
```
plt.figure(figsize=(18,5))
sns.countplot(df['Quantity'])
plt.title('Quantity',fontsize=30)
```

```
C:\Users\VAISHNAVI\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 posit
ional argument will be `data`, and passing other arguments without an explicit keyword w
ill result in an error or misinterpretation.
  warnings.warn(
```

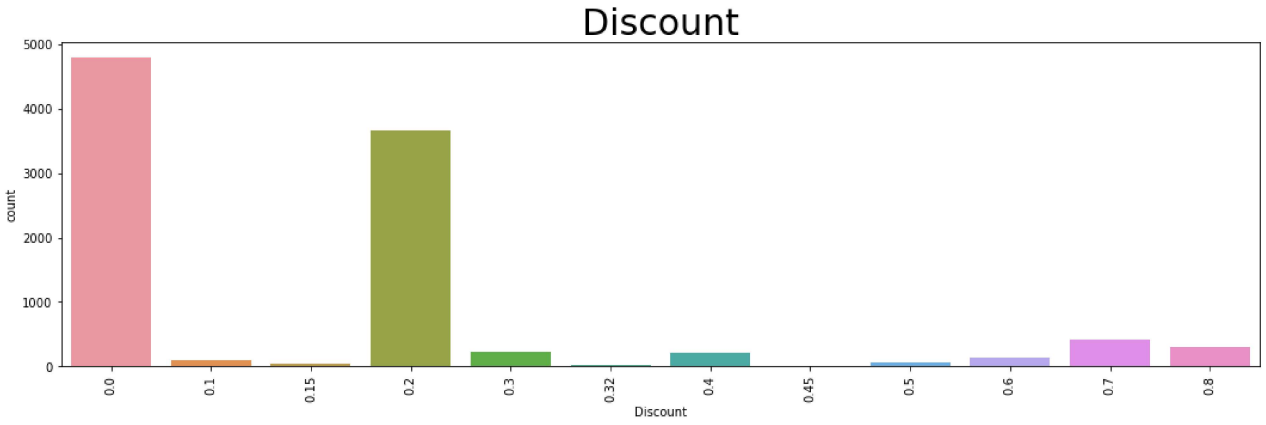
Out[24]: Text(0.5, 1.0, 'Quantity')



In [25]:
```
plt.figure(figsize=(18,5))
sns.countplot(df['Discount'])
plt.xticks(rotation=90)
plt.title('Discount',fontsize=30)
```

```
C:\Users\VAISHNAVI\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 posit
ional argument will be `data`, and passing other arguments without an explicit keyword w
ill result in an error or misinterpretation.
  warnings.warn(
```

Out[25]: Text(0.5, 1.0, 'Discount')

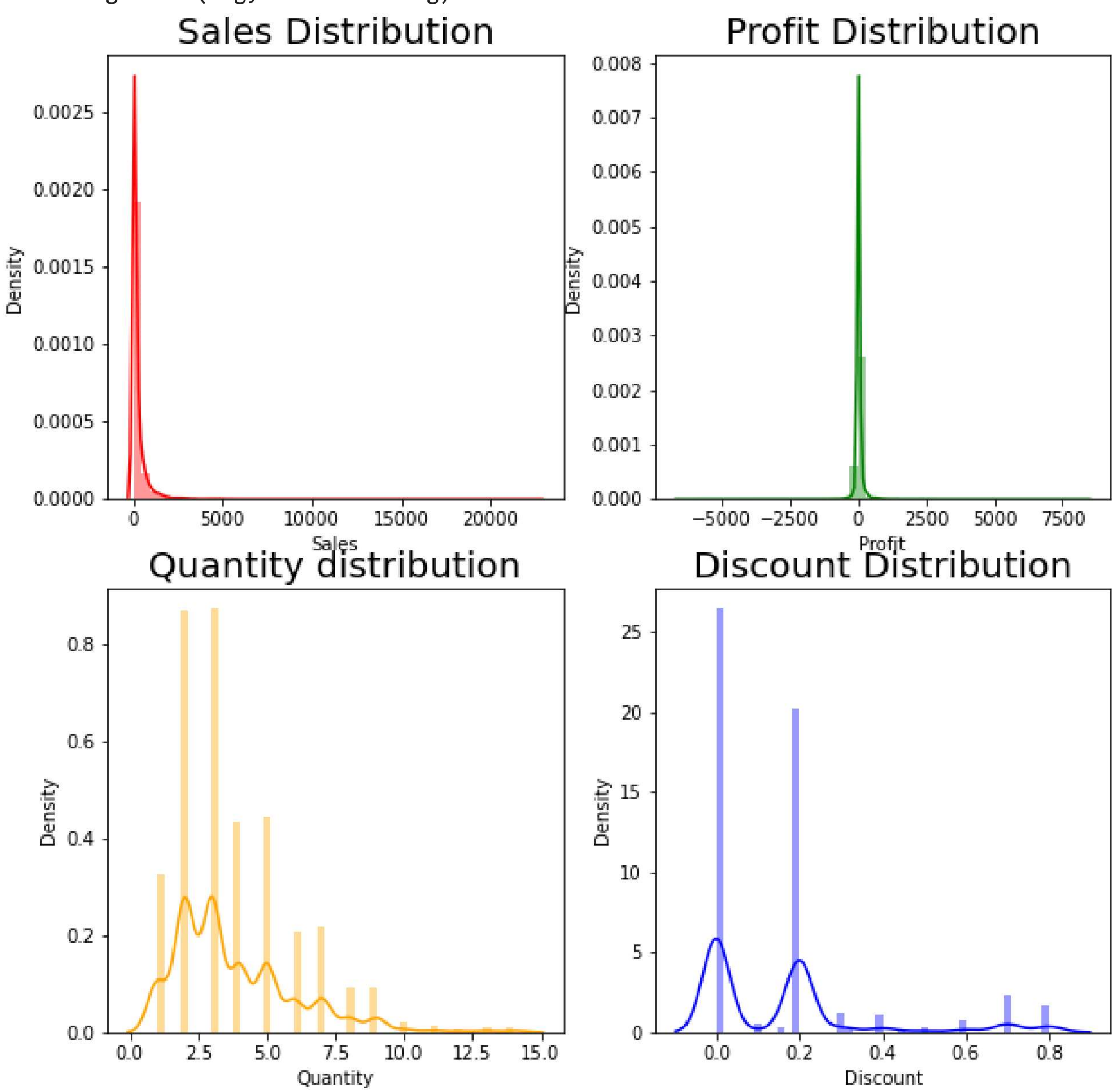


Distribution of the Data Using the Plot

In [26]:
```
fig, axs = plt.subplots(ncols=2, nrows = 2, figsize = (10,10))

sns.distplot(df['Sales'], color = 'red',  ax = axs[0][0])
sns.distplot(df['Profit'], color = 'green',  ax = axs[0][1])
sns.distplot(df['Quantity'], color = 'orange',  ax = axs[1][0])
sns.distplot(df['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\VAISHNAVI\anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarn
```

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



State-wise Deal Analysis

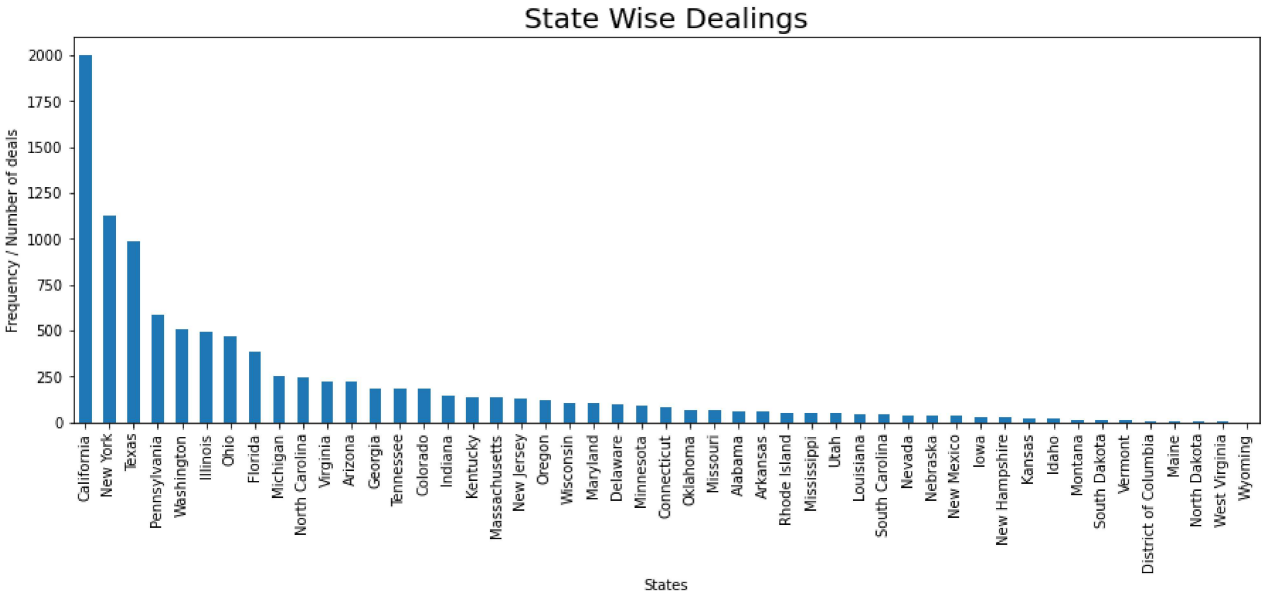
In [27]:
```
df['Country'].value_counts()
```

Out[27]:
```
United States    9994
Name: Country, dtype: int64
```

In [28]:
```
df1 = df['State'].value_counts()
df1.head(10)
```

Out[28]:
```
California        2001
New York          1128
Texas              985
Pennsylvania       587
Washington         506
Illinois           492
Ohio               469
Florida            383
Michigan           255
North Carolina     249
Name: State, dtype: int64
```

In [29]:
```
df1.plot(kind='bar',figsize=(15,5))

plt.ylabel('Frequency / Number of deals')
plt.xlabel('States')

plt.title('State Wise Dealings', fontsize = 20)

plt.show()
```



RESULT

Here is top 3 state where deals are Highest.

Califonia

New York

Texas

Wyoming : Lowest Number of Deal

In [30]:
```
df['State'].value_counts().mean()
```

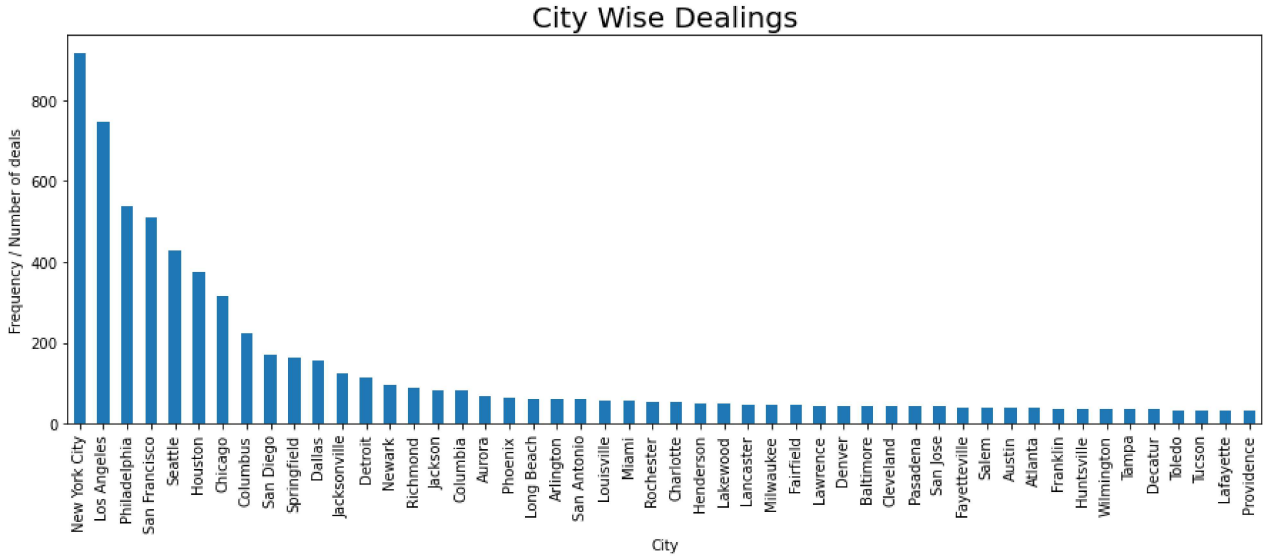
Out[30]: 203.9591836734694

Average number of deal per state is around 204

City-wise Deal Analysis

In [32]:
```
df2 = df['City'].value_counts()
df2=df2.head(50)
```

In [33]:
```
df2.plot(kind='bar',figsize=(15,5))

plt.ylabel('Frequency / Number of deals')
plt.xlabel('City')

plt.title('City Wise Dealings', fontsize = 20)

plt.show()
```



RESULT

Here is top 3 city where deals are Highest.

New York City

Los Angeles

Philadelphia

In [34]:
```
df['City'].value_counts().mean()
```

Out[34]:
```
18.821092278719398
```

Average number of deal per city is 19

Segment-wise Analysis of Sales, Discount & Profit

In [35]:
```
df['Segment'].value_counts()
```

Out[35]:
```
Consumer       5191
Corporate      3020
Home Office    1783
Name: Segment, dtype: int64
```

In [36]:
```
df_segment = df.groupby(['Segment'])[['Sales', 'Discount', 'Profit']].mean()
df_segment
```

Out[36]:

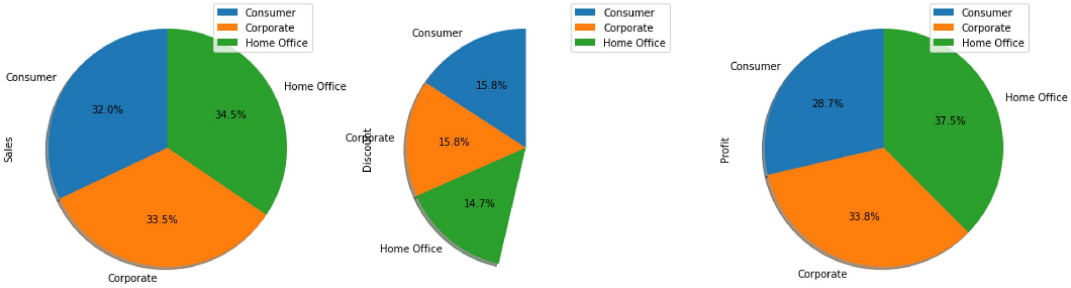
Segment	Sales	Discount	Profit
Consumer	223.733644	0.158141	25.836873
Corporate	233.823300	0.158228	30.456667
Home Office	240.972041	0.147128	33.818664

In [37]:
```
#1. sales 2. Discount 3. Profit

df_segment.plot.pie(subplots=True,
                    autopct='%1.1f%%',
                    figsize=(18, 20),
                    startangle=90,     # start angle 90° (Africa)
                    shadow=True,
                    labels = df_segment.index)

plt.title('Segment-wise Analysis of Sales, Discount & Profit', fontsize=30)
```

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

Out[37]:
```
Text(0.5, 1.0, 'Segment-wise Analysis of Sales, Discount & Profit')
```

Segment-wise Analysis of Sales, Discount & Profit



RESULT

Sales: Consumer : 32% Corporate - 33.5% Home Office : 34.5%

Discount : Consumer : 15.8% Corporate : 15.8% Home Office : 14.7%

Profit : Consumer : 15.8% Corporate : 15.8% Home Office : 14.7%

State-wise Analysis of Sales, Discount & Profit

In [39]:
```
df['State'].value_counts().head(10)
```

Out[39]:
```
California        2001
New York          1128
Texas              985
Pennsylvania       587
Washington         506
Illinois           492
Ohio               469
Florida            383
Michigan           255
North Carolina     249
Name: State, dtype: int64
```

In [40]:
```
df_state= df.groupby(['State'])[['Sales', 'Discount', 'Profit']].mean()
df_state.head(10)
```

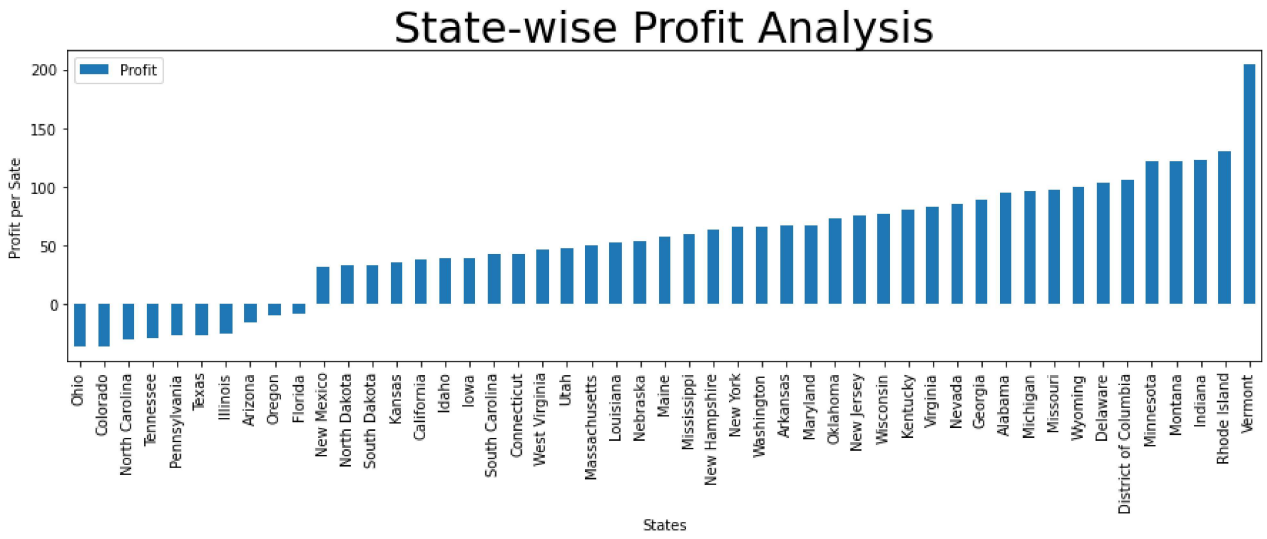
Out[40]:

State	Sales	Discount	Profit
Alabama	319.846557	0.000000	94.865989
Arizona	157.508933	0.303571	-15.303235
Arkansas	194.635500	0.000000	66.811452
California	228.729451	0.072764	38.171608
Colorado	176.418231	0.316484	-35.867351
Connecticut	163.223866	0.007317	42.823071
Delaware	285.948635	0.006250	103.930988

State	Sales	Discount	Profit
District of Columbia	286.502000	0.000000	105.958930
Florida	233.612815	0.299347	-8.875461
Georgia	266.825217	0.000000	88.315453

[1] State-wise Profit Analysis

In [41]:
```
df_state1=df_state.sort_values('Profit')

df_state1[['Profit']].plot(kind = 'bar', figsize = (15,4))
plt.title('State-wise Profit Analysis', fontsize = 30)

plt.ylabel('Profit per Sate')
plt.xlabel('States')

plt.show()
```



RESULT

Vermont :- Highest Profit

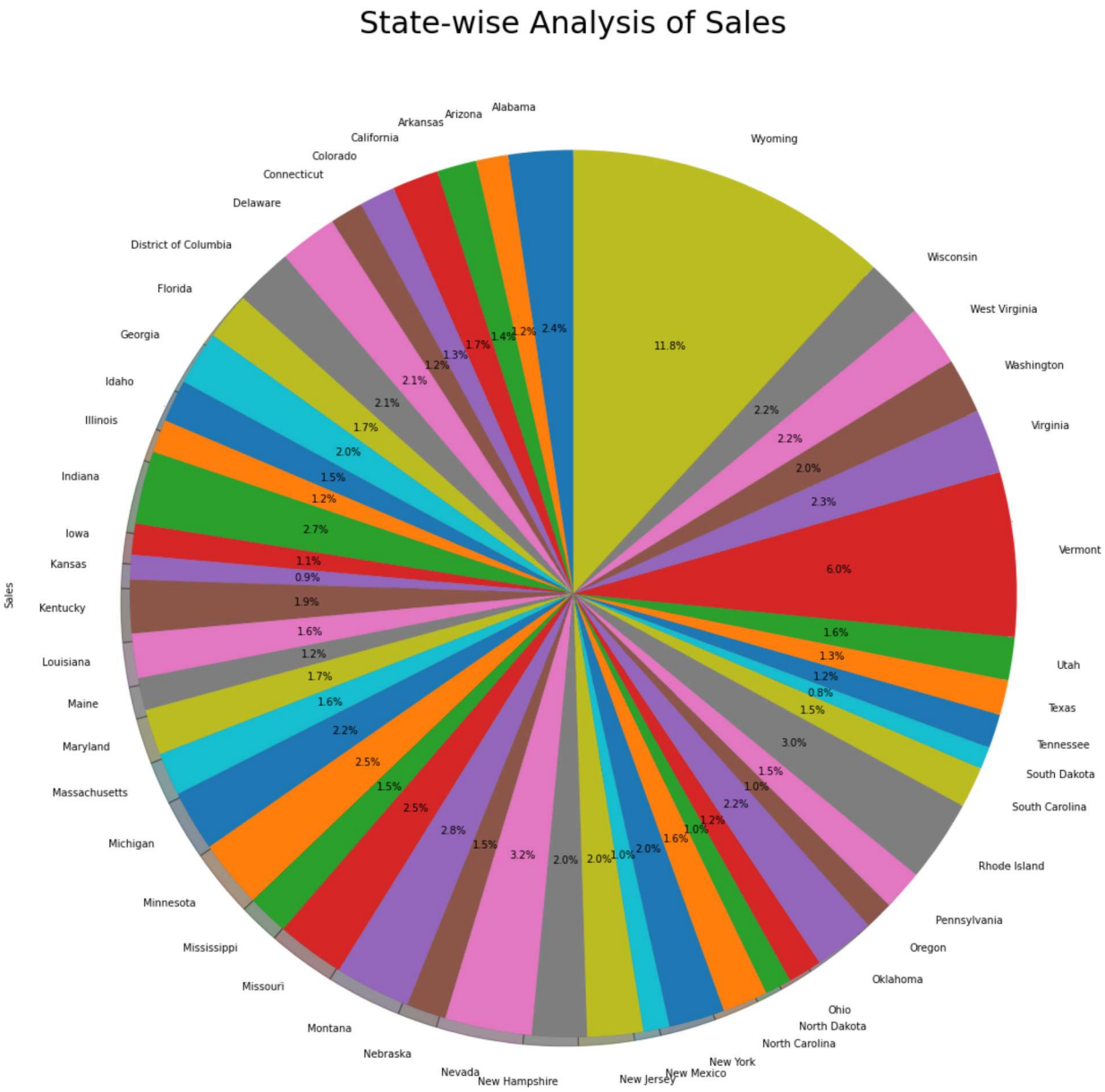
Ohio :- Lowest Profit

[2] State-wise Sales Analysis

In [42]:
```
df_state['Sales'].plot(kind='pie',
                       figsize = (20,20),
                       autopct='%1.1f%%',
                       startangle=90,     # start angle 90° (Africa)
                       shadow=True)
```

```
plt.title('State-wise Analysis of Sales',fontsize=30)
```

Out[42]: Text(0.5, 1.0, 'State-wise Analysis of Sales')



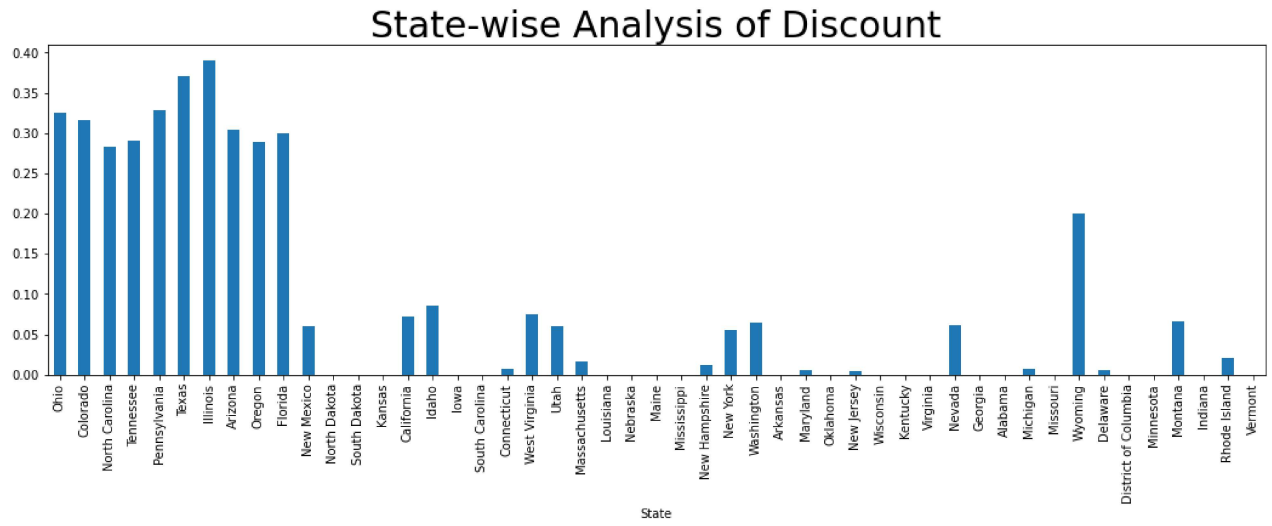
RESULT

Highest amount of sales = Wyoming(11.8%)

Lowest amount of sales = South Dakota(0.8%)

In [43]:
```
df_state1['Discount'].plot(kind='bar',figsize=(18,5))
plt.title('State-wise Analysis of Discount', fontsize=30)
```

Out[43]: Text(0.5, 1.0, 'State-wise Analysis of Discount')



RESULT

Illinois at the top

City-wise Analysis of Sales, Discount & Profit

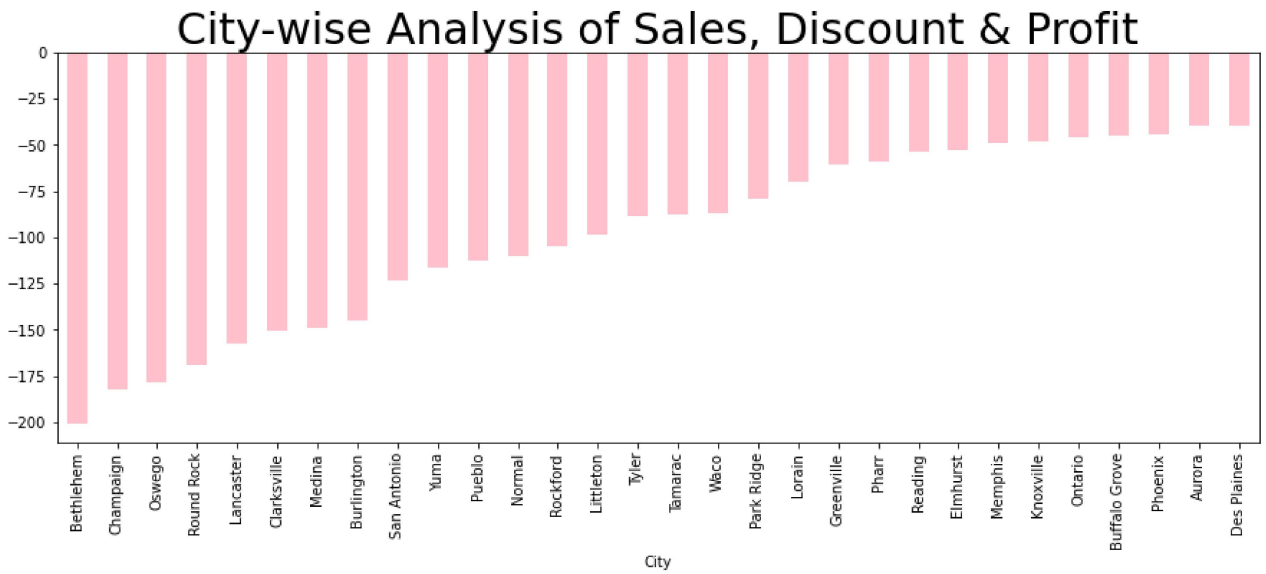
In [45]:
```
df_city= df.groupby(['City'])[['Sales', 'Discount', 'Profit']].mean()
df_city = df_city.sort_values('Profit')
df_city.head()
```

Out[45]:

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

In [46]:
```
#1.Low Profit

df_city['Profit'].head(30).plot(kind='bar',figsize=(15,5),color = 'Pink')
plt.title('City-wise Analysis of Sales, Discount & Profit', fontsize=30)
```

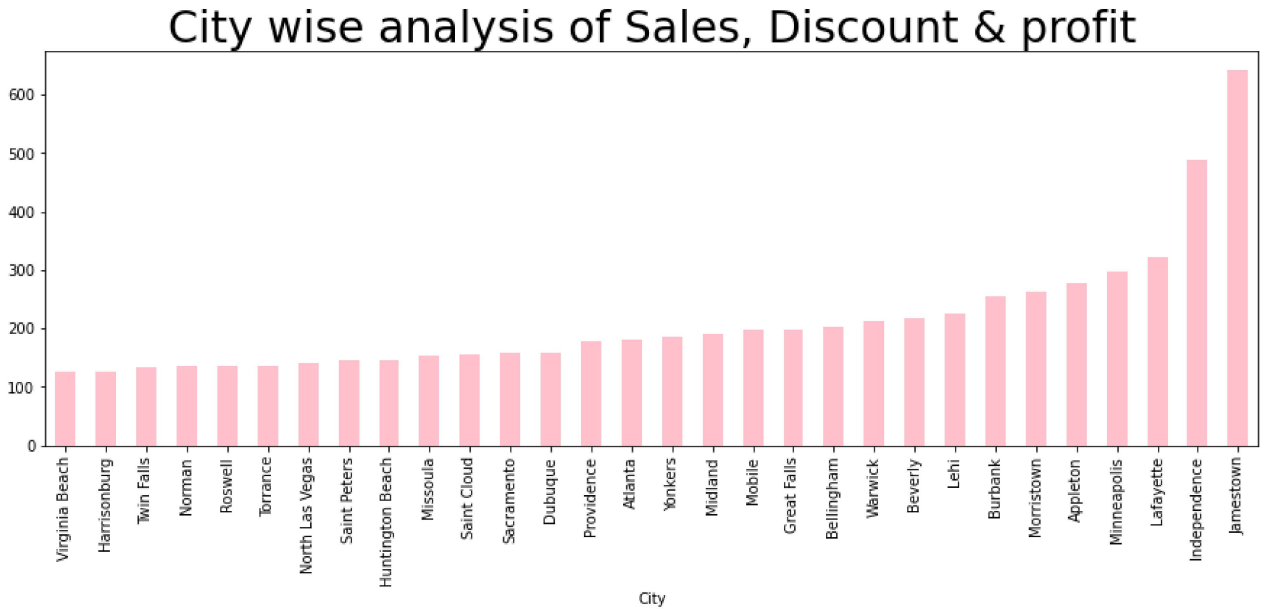
Out[46]:
```
Text(0.5, 1.0, 'City-wise Analysis of Sales, Discount & Profit')
```



In [47]:

```
#2. High Profit

df_city['Profit'].tail(30).plot(kind='bar',figsize=(15,5),color = 'Pink')
plt.title('City wise analysis of Sales, Discount & profit', fontsize=30)
```

Out[47]: Text(0.5, 1.0, 'City wise analysis of Sales, Discount & profit')



RESULT

30 Cities have Positive Profit.

30 Cities Have Negative Profit.

THE BALANCE IS PRETTY GOOD HERE!

Quantity-wise Analysis of Sales, Discount &

Profit

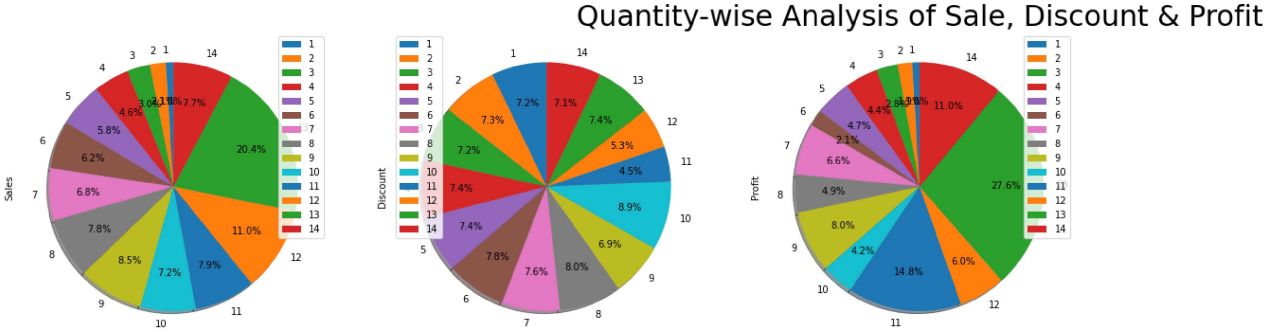
In [48]:
```
df_quantity = df.groupby(['Quantity'])[['Sales', 'Discount', 'Profit']].mean()
df_quantity.head(10)
```

Out[48]:

Quantity	Sales	Discount	Profit
1	59.234632	0.152959	8.276396
2	120.354488	0.154858	16.006831
3	175.201578	0.153329	23.667715
4	271.764059	0.157708	37.131310
5	337.936339	0.157146	40.257394
6	362.101960	0.166556	18.051517
7	395.888393	0.161980	56.579163
8	458.210802	0.171595	42.244342
9	498.083683	0.147946	68.557716
10	422.046737	0.190702	35.862404

In [49]:
```
#1. sales 2. Discount 3. Profit

df_quantity.plot.pie(subplots=True,
                     autopct='%1.1f%%',
                     figsize=(20, 20),
                     pctdistance=0.69,
                     startangle=90,     # start angle 90° (Africa)
                     shadow=True,
                     labels = df_quantity.index)
plt.title('Quantity-wise Analysis of Sale, Discount & Profit',fontsize=30)
```

Out[49]: Text(0.5, 1.0, 'Quantity-wise Analysis of Sale, Discount & Profit')



RESULT

13 Numbered Quantity is High for Sales & Profit

Category-wise Analysis of Sales, Discount & Profit

In [51]:
```
df_category = df.groupby(['Category'])[['Sales', 'Discount', 'Profit']].mean()
df_category
```

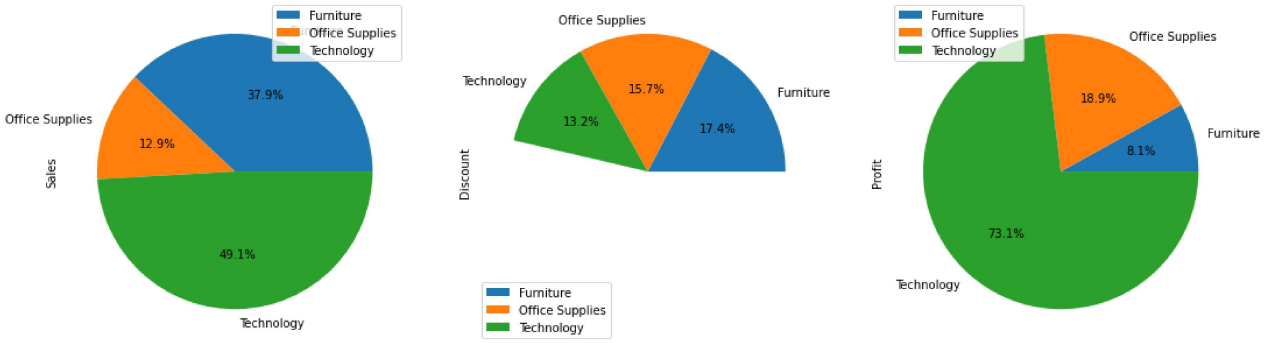
Out[51]:

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

In [52]:
```
df_category.plot.pie(subplots=True,
                     figsize=(18, 20),
                     autopct='%1.1f%%',
                     labels = df_category.index)
```

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

Out[52]:
```
array([<AxesSubplot:ylabel='Sales'>, <AxesSubplot:ylabel='Discount'>,
       <AxesSubplot:ylabel='Profit'>], dtype=object)
```



RESULT

Maximum Sales & Profit is Obtained in Technology

Minimum Profit is Obtained in Furniture

Sub-Category-wise Analysis of Sales, Discount & Profit

In [54]:
```
df_sub_category = df.groupby(['Sub-Category'])[['Sales', 'Discount', 'Profit']].mean()
```

```
df_sub_category.head(10)
```

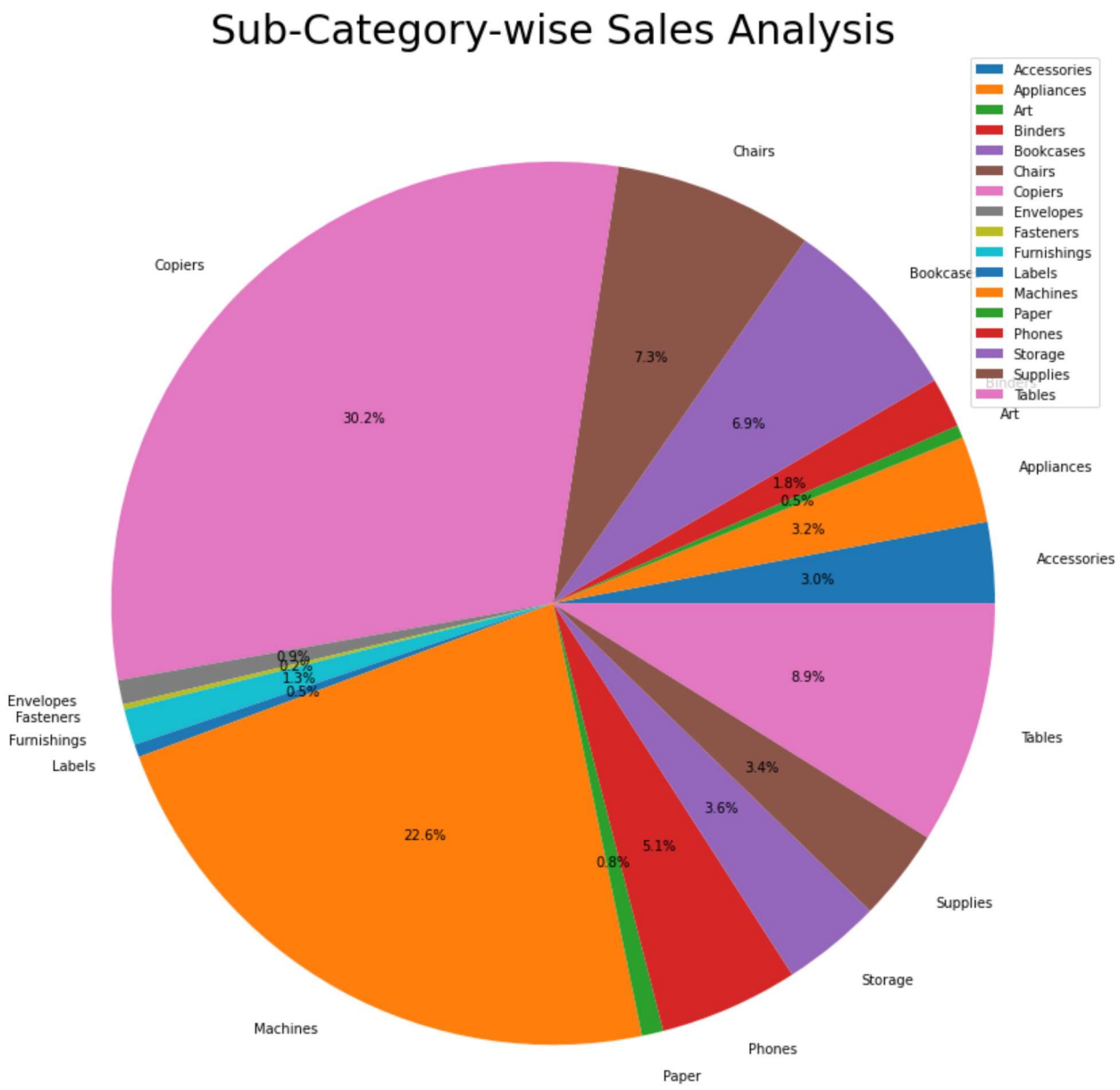
Out[54]:

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

[1] Based on the Sales

In [55]:

```
plt.figure(figsize = (15,15))
plt.pie(df_sub_category['Sales'], labels = df_sub_category.index, autopct = '%1.1f%%')
plt.title('Sub-Category-wise Sales Analysis', fontsize = 30)
plt.legend()
plt.xticks(rotation = 90)

plt.show()
```



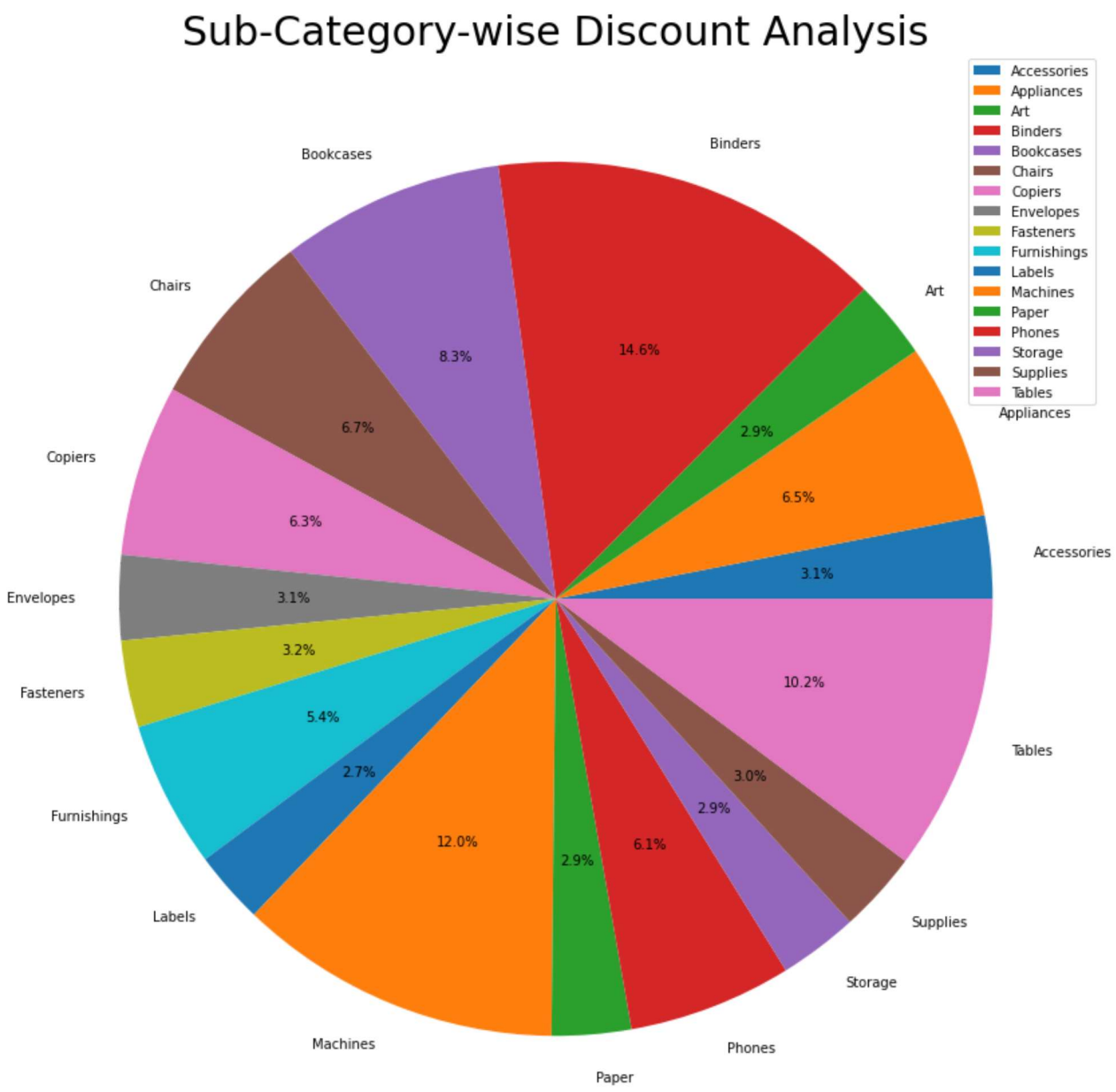
RESULT

Copier & Machines have High Sales

[2] Based on the Discount

In [56]:

```
plt.figure(figsize = (15,15))
plt.pie(df_sub_category['Discount'], labels = df_sub_category.index, autopct = '%1.1f%%
plt.title('Sub-Category-wise Discount Analysis', fontsize = 30)
plt.legend()
plt.xticks(rotation = 90)

plt.show()
```



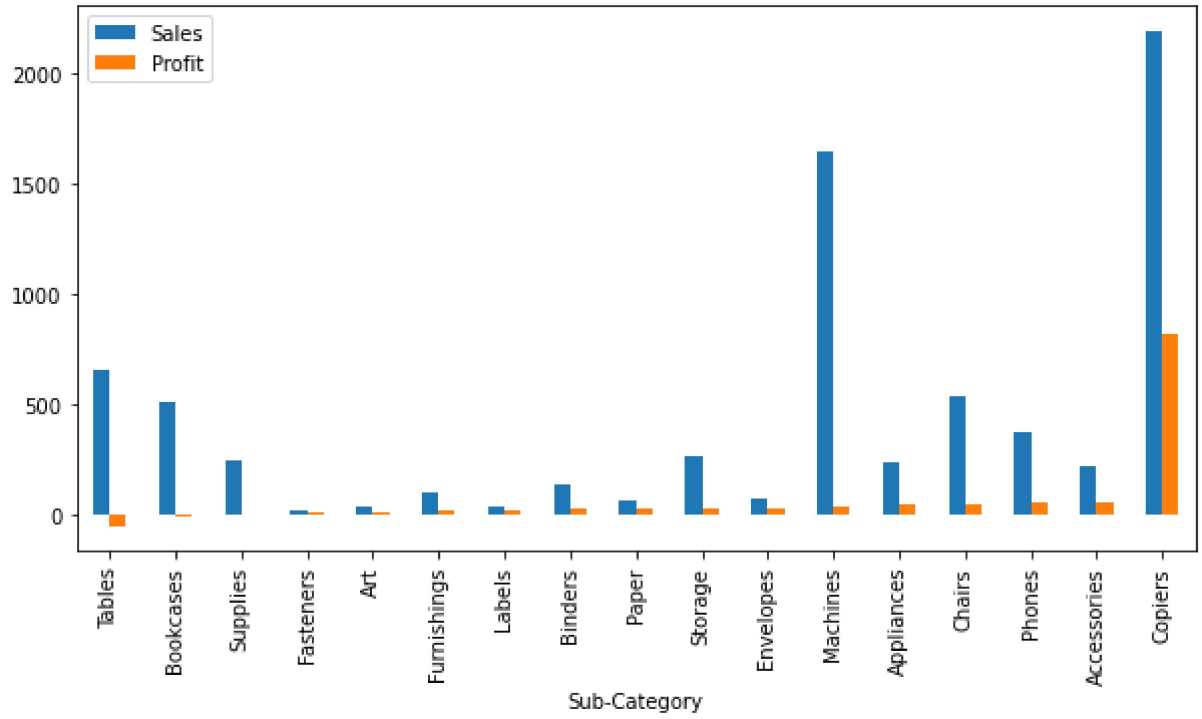
RESULT

Binders, Machines & Tables have High Discounts

[3] Based on the Profit

In [59]:
```
df_sub_category.sort_values('Profit')[['Sales','Profit']].plot(kind='bar',
                                                               figsize= (10,5),
                                                               label=['Avg Sales Price(
```

Out[59]: <AxesSubplot:xlabel='Sub-Category'>



RESULT

Copier has Highest Profit as well as Sale

Region-wise Analysis of Sales, Discount & Profit

In [61]:
```
df_region = df.groupby(['Region'])[['Sales', 'Discount', 'Profit']].mean()
df_region
```

Out[61]:

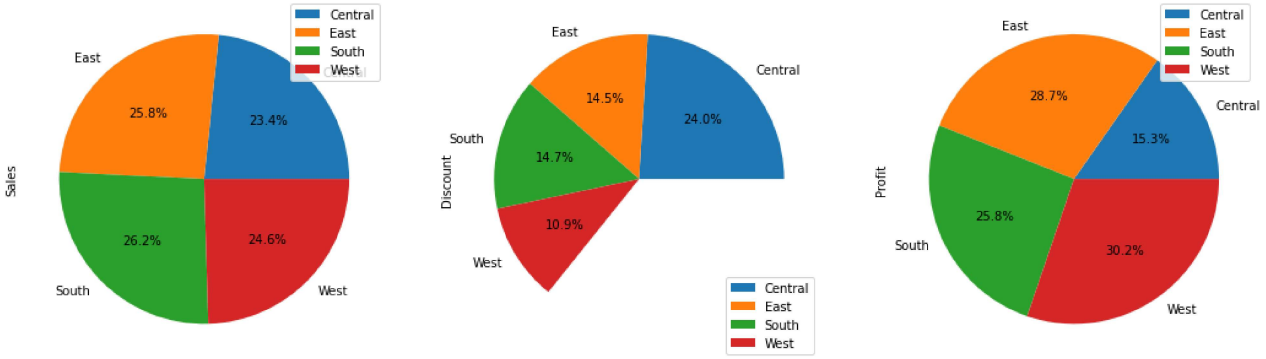
Region	Sales	Discount	Profit
Central	215.772661	0.240353	17.092709
East	238.336110	0.145365	32.135808
South	241.803645	0.147253	28.857673
West	226.493233	0.109335	33.849032

In [62]:
```
df_region.plot.pie(subplots=True,
                   figsize=(18, 20),
                   autopct='%1.1f%%',
                   labels = df_region.index)
```

```
C:\Users\VAISHNAVI\anaconda3\lib\site-packages\pandas\plotting\_matplotlib\core.py:1583:
MatplotlibDeprecationWarning: normalize=None does not normalize if the sum is less than
1 but this behavior is deprecated since 3.3 until two minor releases later. After the de
```

```
precation period the default value will be normalize=True. To prevent normalization pass
normalize=False
  results = ax.pie(y, labels=blabels, **kwds)
```

Out[62]:
```
array([<AxesSubplot:ylabel='Sales'>, <AxesSubplot:ylabel='Discount'>,
       <AxesSubplot:ylabel='Profit'>], dtype=object)
```



RESULT

West Region is the Most Profitable Region

Ship Mode-wise Analysis of Sales, Discount & Profit

In [64]:
```
df['Ship Mode'].value_counts()
```

Out[64]:
```
Standard Class    5968
Second Class      1945
First Class       1538
Same Day           543
Name: Ship Mode, dtype: int64
```

In [66]:
```
df_shipmode = df.groupby(['Ship Mode'])[['Sales', 'Discount', 'Profit']].mean()
df_shipmode
```

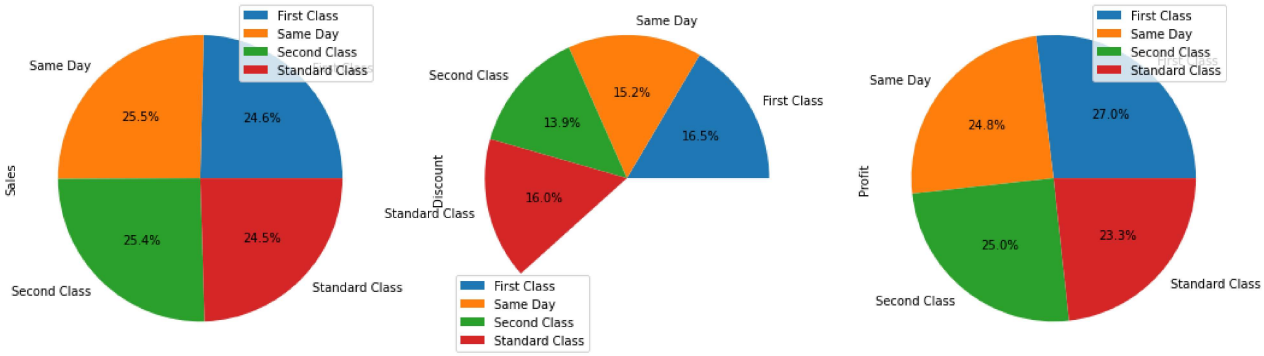
Out[66]:

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

In [67]:
```
df_shipmode.plot.pie(subplots=True,
                     figsize=(18, 20),
                     autopct='%1.1f%%',
                     labels = df_shipmode.index)
```

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

Out[67]:
```
array([<AxesSubplot:ylabel='Sales'>, <AxesSubplot:ylabel='Discount'>,
       <AxesSubplot:ylabel='Profit'>], dtype=object)
```



RESULT

Profit & Discount is High in First Class

Sales is High for Same day Ship

RESULT AND CONCLUSION

Profit is more than that of sale but there are some areas where profit could be increased.

Profit and Discount is high in First Class

Sales is high for Same day ship

Sub-category: Copier: High Profit & sales

Sub-category: Binders , Machines and then tables have high Discount.

Category: Maximun sales and Profit obtain in Technology.

Category: Minimun profit obtain in Furniture

State: Vermont: Highest Profit

State: Ohio: Lowest Profit

Segment: Home-office: High Profit & sales

Here is top 3 city where deals are Highest.

New York City

Los Angeles

Philadelphia

Sales and Profit are Moderately Correlated.

Quantity and Profit are less Moderately Correlated.

Discount and Profit are Negatively Correlated

Here is top 3 state where deals are Highest.

Califonia

New York

Texas

Wyoming : Lowest Number of Deal

Highest amount of sales = Wyoming(11.8%)

Lowest amount of sales = South Dakota(0.8%)

THANK YOU

In [ ]: