In [ ]:
```
# The Spark Foundation
# Ayush Singh
```

In [ ]:
```
# Function - Data Science and Business Analytics
# Grip Task - 3
# Exploratory Data Analysis - Retail.
# Dataset used - SampleSuperstore.csv
```

In [1]:
```
# Importing the required libraries.
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
# Importinting the data.
df=pd.read_csv("D:/Data/SampleSuperstore.csv")
```

In [3]:
```
# Understanding the nature of data
df.head()
```

Out[3]:

	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	261.96
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 [4]:
```
df.tail()
```

Out[4]:

	Ship Mode	Segment	Country	City	State	Postal Code	Region	Category	Sub-Category
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

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

Out[5]:
```
(9994, 13)
```

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

Out[6]:

	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 [7]:
```
df.describe(include='object')
```

Out[7]:

	Ship Mode	Segment	Country	City	State	Region	Category	Sub-Category
count	9994	9994	9994	9994	9994	9994	9994	9994
unique	4	3	1	531	49	4	3	17
top	Standard Class	Consumer	United States	New York City	California	West	Office Supplies	Binders
freq	5968	5191	9994	915	2001	3203	6026	1523

In [8]:
```
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 [9]:
```
columns=df.columns
columns
```

Out[9]:
```
Index(['Ship Mode', 'Segment', 'Country', 'City', 'State', 'Postal Code',
       'Region', 'Category', 'Sub-Category', 'Sales', 'Quantity', 'Discount',
       'Profit'],
      dtype='object')
```

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

Out[10]:
```
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
```

In [11]:
```
for a in columns:
    print(a,":","unique values are:",df[a].unique())
```

```
Ship Mode : unique values are: ['Second Class' 'Standard Class' 'First Clas
s' 'Same Day']
Segment : unique values are: ['Consumer' 'Corporate' 'Home Office']
Country : unique values are: ['United States']
City : unique values are: ['Henderson' 'Los Angeles' 'Fort Lauderdale' 'Conc
ord' 'Seattle'
 'Fort Worth' 'Madison' 'West Jordan' 'San Francisco' 'Fremont'
 'Philadelphia' 'Orem' 'Houston' 'Richardson' 'Naperville' 'Melbourne'
 'Eagan' 'Westland' 'Dover' 'New Albany' 'New York City' 'Troy' 'Chicago'
 'Gilbert' 'Springfield' 'Jackson' 'Memphis' 'Decatur' 'Durham' 'Columbia'
 'Rochester' 'Minneapolis' 'Portland' 'Saint Paul' 'Aurora' 'Charlotte'
 'Orland Park' 'Urbandale' 'Columbus' 'Bristol' 'Wilmington' 'Bloomington'
 'Phoenix' 'Roseville' 'Independence' 'Pasadena' 'Newark' 'Franklin'
 'Scottsdale' 'San Jose' 'Edmond' 'Carlsbad' 'San Antonio' 'Monroe'
 'Fairfield' 'Grand Prairie' 'Redlands' 'Hamilton' 'Westfield' 'Akron'
 'Denver' 'Dallas' 'Whittier' 'Saginaw' 'Medina' 'Dublin' 'Detroit'
 'Tampa' 'Santa Clara' 'Lakeville' 'San Diego' 'Brentwood' 'Chapel Hill'
 'Morristown' 'Cincinnati' 'Inglewood' 'Tamarac' 'Colorado Springs'
 'Belleville' 'Taylor' 'Lakewood' 'Arlington' 'Arvada' 'Hackensack'
```

In [43]:
```
def getunique(unq):
    print(unq,": unique values are :",df[unq].unique())
```

In [46]:
```
unq=input("Enter the name of column to get the unique values")
getunique(unq)
```

```
Enter the name of column to get the unique valuesSub-Category
Sub-Category : unique values are : ['Bookcases' 'Chairs' 'Labels' 'Tables' 'Sto
rage' 'Furnishings' 'Art'
 'Phones' 'Binders' 'Appliances' 'Paper' 'Accessories' 'Envelopes'
 'Fasteners' 'Supplies' 'Machines' 'Copiers']
```

In [47]:
```
# Checking the null value in data.
df.isnull().sum()
```

Out[47]:
```
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 [48]:
```
# Univariant Analysis:
```

In [49]:
```
df.head(2)
```

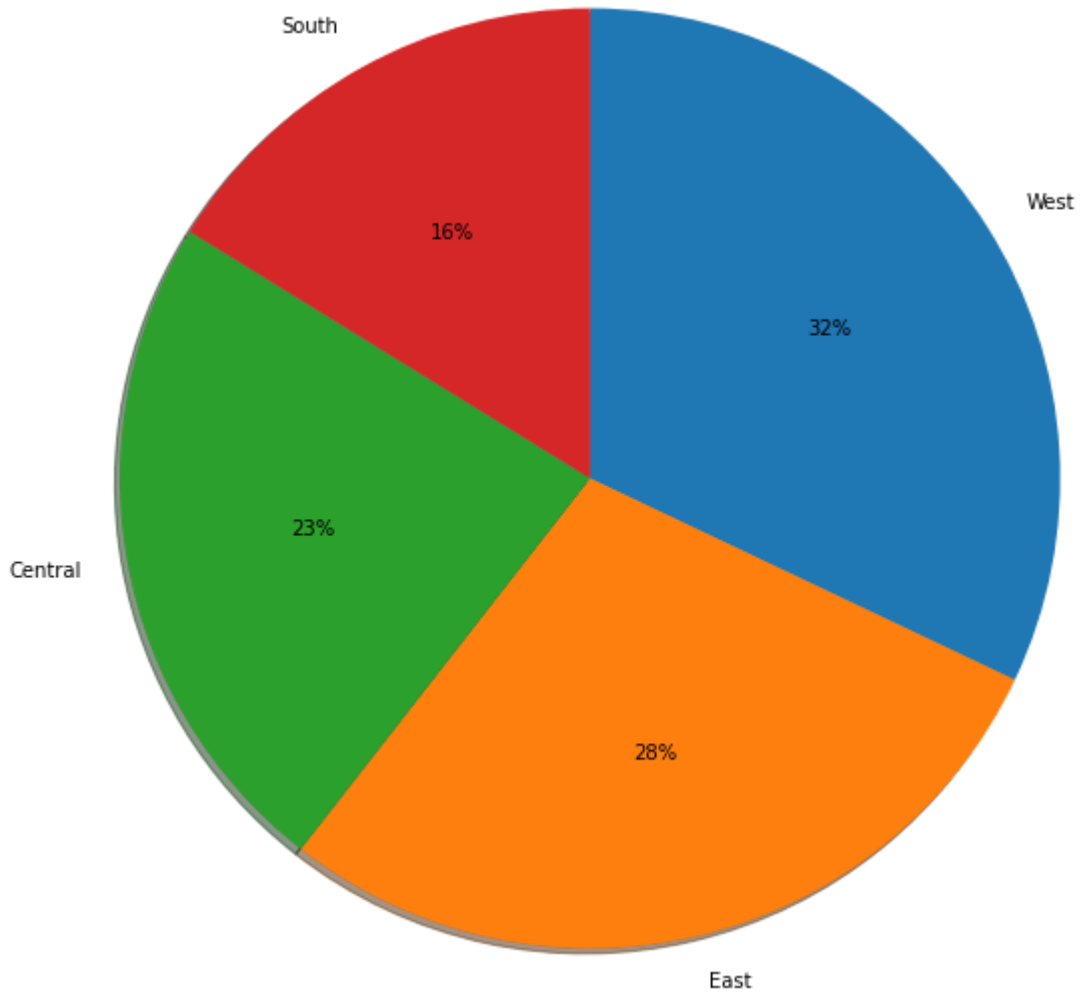
Out[49]:

	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	261.96
1	Second Class	Consumer	United States	Henderson	Kentucky	42420	South	Furniture	Chairs	731.94

In [50]:
```
def univaranalitics(clmname):
    print(clmname)
    print("Count of its unique values")
    print(df[clmname].value_counts())
    print("==============")
    uva=df[clmname].value_counts()/len(df['Ship Mode'])*100
    print("Percentage sharing of its unique values")
    print(uva)
    print("==============")
    x=uva.index.tolist()
    y=uva.values.tolist()
    print("Pie Chart of percentage Sharing")
    plt.pie(y,labels=x,autopct='%1.f%%',startangle=90,shadow=True,counterclock=Fa
    plt.show()
```

In [52]:
```
clmname=input("Enter the column name to get its univariant analysis (only Categor
univaranalitics(clmname)
#Here we can see the unique values of Column and their percentage sharing in data
```

```
Enter the column name to get its univariant analysis (only Categorical type Col
umn names are applicable )Region
Region
Count of its unique values
West       3203
East       2848
Central    2323
South      1620
Name: Region, dtype: int64
==============
Percentage sharing of its unique values
West       32.049230
East       28.497098
Central    23.243946
South      16.209726
Name: Region, dtype: float64
==============
Pie Chart of percentage Sharing
```



In [53]:
```
# Bivarient Analysis:
```

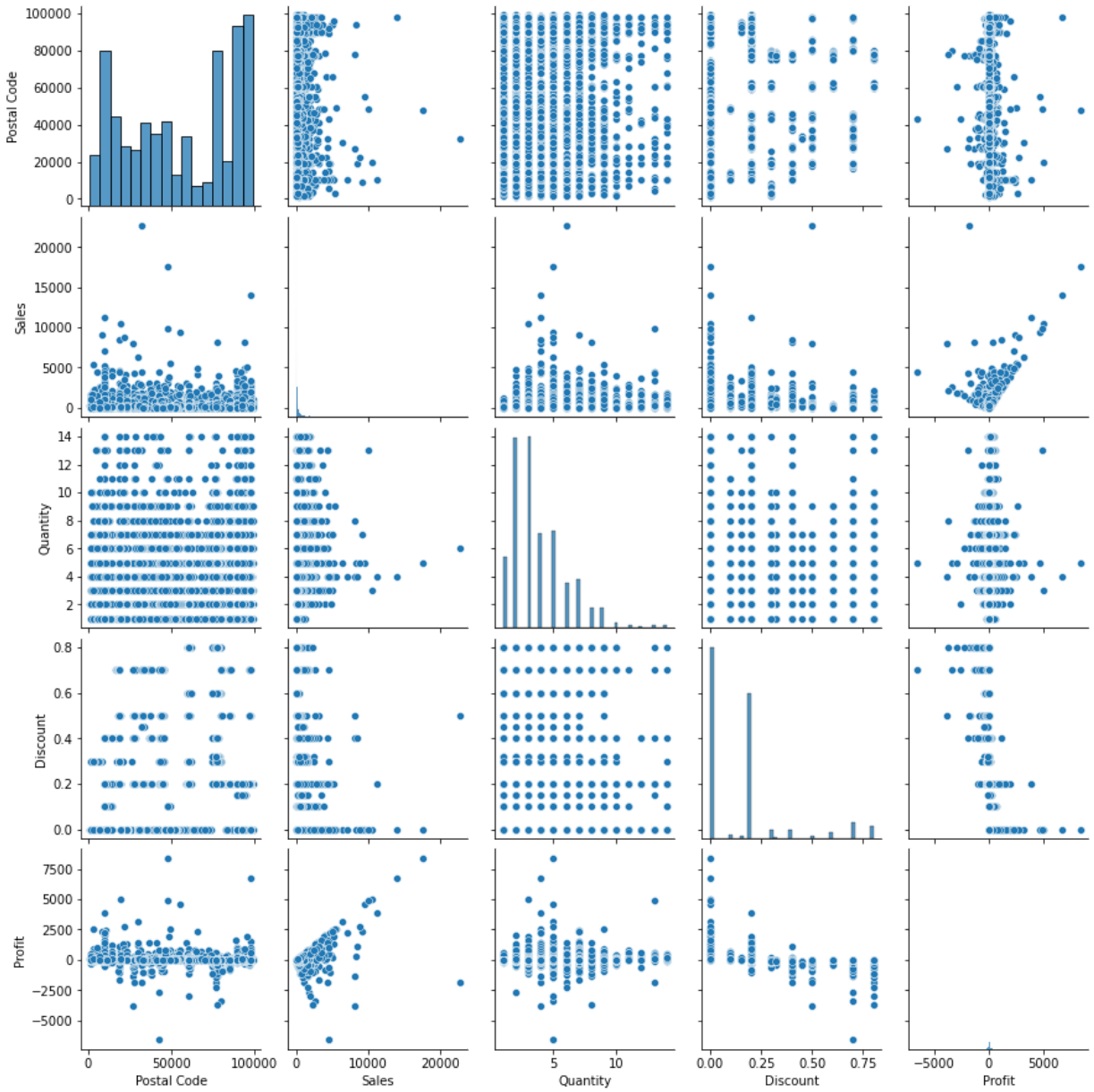
In [54]:
```
df.head(2)
```

Out[54]:

	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	261.96
1	Second Class	Consumer	United States	Henderson	Kentucky	42420	South	Furniture	Chairs	731.94

In [23]:
```
sns.pairplot(df) # It will give plot between each and evry column with the help o
#we can get a breif knowledge of data trends.
```

Out[23]:
```
<seaborn.axisgrid.PairGrid at 0x2601b5e93a0>
```



In [24]:
```
cr=df.corr() # It will show the relationship(Correlation) between the each numeri
cr
```

Out[24]:

	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 [25]:
```
sns.heatmap(cr,xticklabels=cr.columns,yticklabels=cr.columns,annot=True)
```

Out[25]: <AxesSubplot:>

In [26]:
```
df.columns
```

Out[26]:
```
Index(['Ship Mode', 'Segment', 'Country', 'City', 'State', 'Postal Code',
       'Region', 'Category', 'Sub-Category', 'Sales', 'Quantity', 'Discount',
       'Profit'],
      dtype='object')
```

In [27]:
```
from ipywidgets import interact
```

In [28]:
```
def getplot(clmn):
    sns.relplot(x=df['Sales'],y=df['Profit'],hue=df[clmn],data='df')
# This function gives us the scatter plot between 'Sales' and 'Profit' with resec
```

In [29]:
```
ls=['Ship Mode', 'Segment', 'Country', 'City', 'State', 'Postal Code',
       'Region', 'Category', 'Sub-Category','Quantity', 'Discount',]
interact(getplot,clmn=ls)
```

A Jupyter widget could not be displayed because the widget state could not be found. This could happen if the kernel storing the widget is no longer available, or if the widget state was not saved in the notebook. You may be able to create the widget by running the appropriate cells.

Out[29]: <function __main__.getplot(clmn)>

In [55]:
```
def getgroupby(atr):
    prw=df.groupby(atr)['Profit'].sum().round().sort_values(ascending=False)
    print(prw)
    x=prw.index.tolist()
    y=prw.values.tolist()
    #plt.pie(y,labels=x,startangle=90,autopct='%1.f%%',radius=2,counterclock=Fals
    plt.rcParams['figure.figsize'][0]=12
    plt.rcParams['figure.figsize'][1]=6
    plt.bar(x,y,label='Profit Values')
    plt.legend()
    plt.xlabel(atr)
    plt.ylabel('Profit Value in USD')
    plt.xticks(rotation=90)
    for i in range(len(x)):
        plt.annotate(xy=[x[i],y[i]+.5], text=y[i],rotation=45)
    plt.show()
# This function gives us the sum of profit as per our selection in dropdown , whi
# represented in a bar chart so we can easily get the insights of heghest and low
```

In [56]:
```
ls1=['Ship Mode', 'Segment', 'State', 'Region', 'Category', 'Sub-Category', 'Quar
interact(getgroupby,atr=ls1)
# While Selecting the 'Sub-Category' and 'State' from the dropdown we can see tha
# -ve value of Profit that is loss so we need work on those item and those state.
```

A Jupyter widget could not be displayed because the widget state could not be found. This could happen if the kernel storing the widget is no longer available, or if the widget state was not saved in the notebook. You may be able to create the widget by running the appropriate cells.

Out[56]:
```
<function __main__.getgroupby(atr)>
```

In [57]:
```
def getgroupby1(atr1):
    slw=df.groupby(atr1)['Sales'].sum().round().sort_values(ascending=False)
    print(slw)
    x1=slw.index.tolist()
    y1=slw.values.tolist()
    #plt.pie(y,labels=x,autopct='%1.f%%',startangle=True,counterclock=False,radiu
    plt.bar(x1,y1,label='Sales Value')
    plt.legend()
    plt.xlabel(atr1)
    plt.ylabel('Sales Value in USD')
    plt.xticks(rotation=90)
    for i in range(len(x1)):
        plt.annotate(xy=[x1[i],y1[i]+.5], text=y1[i],rotation=45)
    plt.show()
# This function gives us the sum of profit as per our selection in dropdown , whi
# represented in a bar chart so we can easily get the insights of heghest and low
```

In [58]:
```
ls1=['Ship Mode', 'Segment', 'State', 'Region', 'Category', 'Sub-Category', 'Quar
interact(getgroupby1,atr1=ls1)
```

A Jupyter widget could not be displayed because the widget state could not be found. This could happen if the kernel storing the widget is no longer available, or if the widget state was not saved in the notebook. You may be able to create the widget by running the appropriate cells.

Out[58]:
```
<function __main__.getgroupby1(atr1)>
```

In [59]:
```
def gettable(abc):
    M=df.groupby(abc)['Sales','Profit','Quantity'].sum().round()
    print(M)
# Using this function we can receive a table showing the sum of 'Sales','Profit'
```

In [60]:
```
lsm=['Ship Mode', 'Segment', 'City', 'State', 'Postal Code',
       'Region', 'Category', 'Sub-Category',]
interact(gettable,abc=lsm)
```

A Jupyter widget could not be displayed because the widget state could not be found. This could happen if the kernel storing the widget is no longer available, or if the widget state was not saved in the notebook. You may be able to create the widget by running the appropriate cells.

Out[60]:
```
<function __main__.gettable(abc)>
```

In [36]:
```
df.groupby(['Category','Sub-Category'])['Sales','Profit','Quantity'].sum().round(
# Below is the table showing the brancing of 'Category' and 'Sub-Category' and th
# and 'Sold Quantity', so we can easily say that
```

```
<ipython-input-36-f003727708ee>:1: FutureWarning: Indexing with multiple keys
(implicitly converted to a tuple of keys) will be deprecated, use a list instea
d.
  df.groupby(['Category','Sub-Category'])['Sales','Profit','Quantity'].sum().ro
und()
```

Out[36]:

Category	Sub-Category	Sales	Profit	Quantity
Furniture	**Bookcases**	114880.0	-3473.0	868
	Chairs	328449.0	26590.0	2356
	Furnishings	91705.0	13059.0	3563
	Tables	206966.0	-17725.0	1241
Office Supplies	**Appliances**	107532.0	18138.0	1729
	Art	27119.0	6528.0	3000
	Binders	203413.0	30222.0	5974
	Envelopes	16476.0	6964.0	906
	Fasteners	3024.0	950.0	914
	Labels	12486.0	5546.0	1400
	Paper	78479.0	34054.0	5178
	Storage	223844.0	21279.0	3158
	Supplies	46674.0	-1189.0	647
Technology	**Accessories**	167380.0	41937.0	2976
	Copiers	149528.0	55618.0	234
	Machines	189239.0	3385.0	440
	Phones	330007.0	44516.0	3289

In [37]:
```
df.groupby(['Region','State'])['Sales','Profit','Quantity'].sum().round()
# Below is table showing the branching of 'Region' and 'State' and representation
# in each state
```

```
<ipython-input-37-46c79382ebcb>:1: FutureWarning: Indexing with multiple keys
(implicitly converted to a tuple of keys) will be deprecated, use a list instea
d.
  df.groupby(['Region','State'])['Sales','Profit','Quantity'].sum().round()
```

Out[37]:

Region	State	Sales	Profit	Quantity
Central	Illinois	80166.0	-12608.0	1845
	Indiana	53555.0	18383.0	578
	Iowa	4580.0	1184.0	112
	Kansas	2914.0	836.0	74
	Michigan	76270.0	24463.0	946
	Minnesota	29863.0	10823.0	331
	Missouri	22205.0	6436.0	252
	Nebraska	7465.0	2037.0	136
	North Dakota	920.0	230.0	30
	Oklahoma	19683.0	4854.0	247
	South Dakota	1316.0	395.0	42
	Texas	170188.0	-25729.0	3724
	Wisconsin	32115.0	8402.0	463
East	Connecticut	13384.0	3511.0	281
	Delaware	27451.0	9977.0	367
	District of Columbia	2865.0	1060.0	40
	Maine	1271.0	454.0	35
	Maryland	23706.0	7031.0	420
	Massachusetts	28634.0	6786.0	491
	New Hampshire	7293.0	1707.0	127
	New Jersey	35764.0	9773.0	454
	New York	310876.0	74039.0	4224
	Ohio	78258.0	-16971.0	1759
	Pennsylvania	116512.0	-15560.0	2153
	Rhode Island	22628.0	7286.0	199
	Vermont	8929.0	2245.0	50
	West Virginia	1210.0	186.0	18
South	Alabama	19511.0	5787.0	256

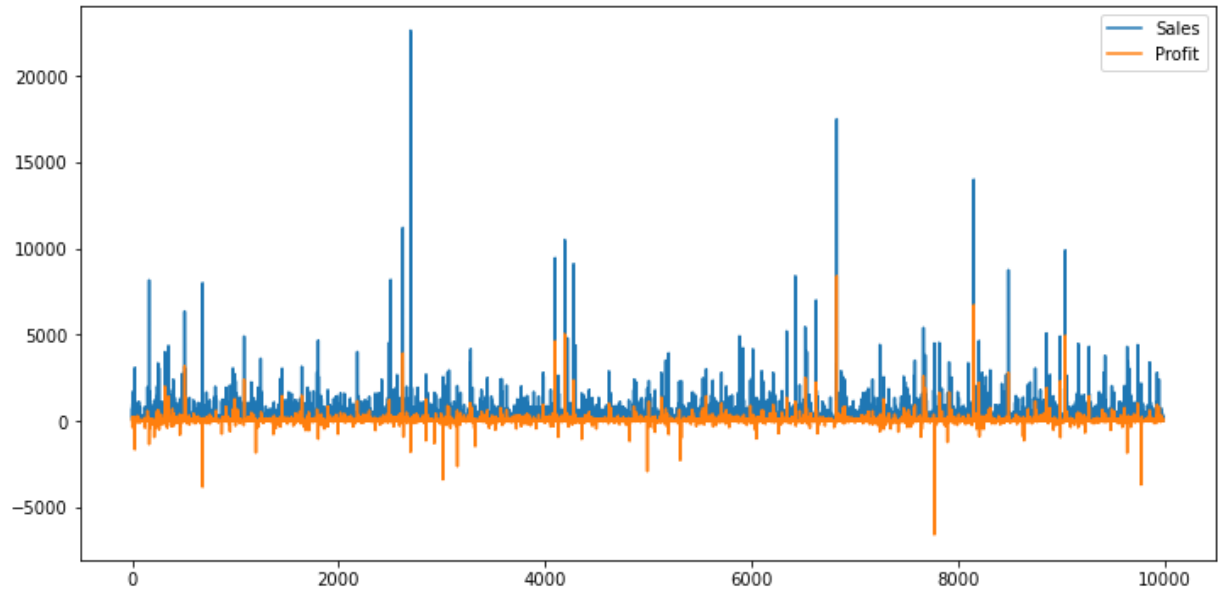
Region	State	Sales	Profit	Quantity
	Arkansas	11678.0	4009.0	240
	Florida	89474.0	-3399.0	1379
	Georgia	49096.0	16250.0	705
	Kentucky	36592.0	11200.0	523
	Louisiana	9217.0	2196.0	156
	Mississippi	10771.0	3173.0	221
	North Carolina	55603.0	-7491.0	983
	South Carolina	8482.0	1769.0	172
	Tennessee	30662.0	-5342.0	681
	Virginia	70637.0	18598.0	893
West	**Arizona**	35282.0	-3428.0	862
	California	457688.0	76381.0	7667
	Colorado	32108.0	-6528.0	693
	Idaho	4382.0	827.0	64
	Montana	5589.0	1833.0	56
	Nevada	16729.0	3317.0	168
	New Mexico	4784.0	1157.0	151
	Oregon	17431.0	-1190.0	499
	Utah	11220.0	2547.0	219
	Washington	138641.0	33403.0	1883
	Wyoming	1603.0	100.0	4

In [38]:
```
df.head(2)
```

Out[38]:

	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	261.96
1	Second Class	Consumer	United States	Henderson	Kentucky	42420	South	Furniture	Chairs	731.94

In [39]:
```
plt.rcParams['figure.figsize'][0]=12
plt.rcParams['figure.figsize'][1]=6
plt.plot(range(len(df)),df['Sales'],label='Sales')
plt.plot(range(len(df)),df['Profit'],label='Profit')
plt.legend()
plt.show()
# This will provide a line chart of 'Sales' and 'Profit' through out the data.
# And by seeing it we can say that the profit line is going below the 0 many time
# explore that and resolve it.
```



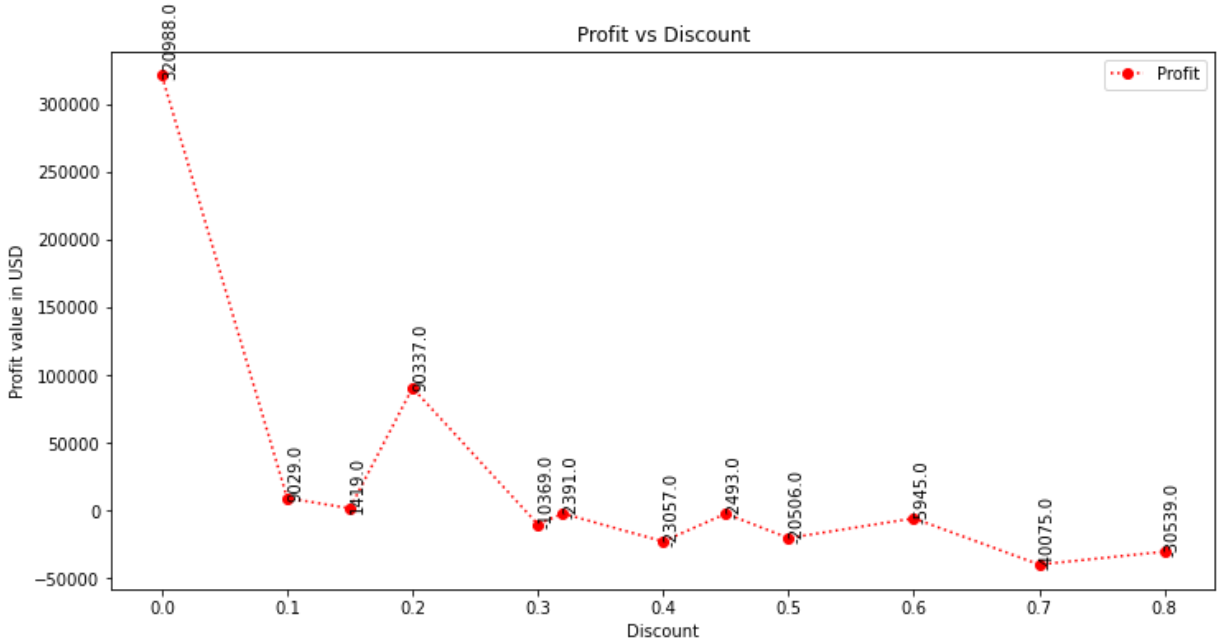
In [40]:
```
df.groupby('Discount')['Sales','Profit'].sum().round()
# Below is table showing the sum of 'Sales' and 'Profit' as per discount amount.
# As we are incresing the discount amount our profit is decreasing and highest pr
# As it clearly visible the when we have increases the discount amount at 30% and
```

```
<ipython-input-40-4f26cd1ffe0b>:1: FutureWarning: Indexing with multiple keys
(implicitly converted to a tuple of keys) will be deprecated, use a list instea
d.
  df.groupby('Discount')['Sales','Profit'].sum().round()
```

Out[40]:

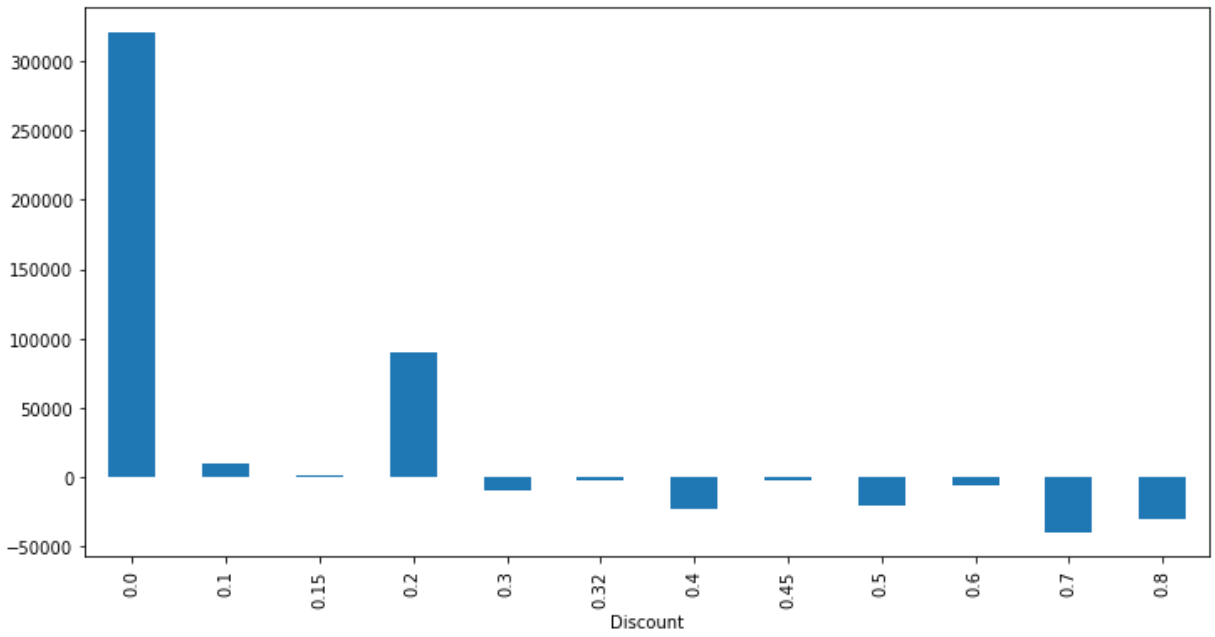
Discount	Sales	Profit
0.00	1087908.0	320988.0
0.10	54369.0	9029.0
0.15	27559.0	1419.0
0.20	764594.0	90337.0
0.30	103227.0	-10369.0
0.32	14493.0	-2391.0
0.40	116418.0	-23057.0
0.45	5485.0	-2493.0
0.50	58919.0	-20506.0
0.60	6645.0	-5945.0
0.70	40620.0	-40075.0
0.80	16964.0	-30539.0

In [41]:
```
k=df.groupby('Discount')['Profit'].sum().round()
x=k.index.tolist()
y=k.values.tolist()
plt.plot(x,y,":go",color='red',label='Profit')
plt.legend()
for i in range(len(x)):
    plt.annotate(xy=[x[i],y[i]+.5], text=y[i], rotation=90)
plt.xlabel("Discount")
plt.ylabel('Profit value in USD')
plt.title("Profit vs Discount")
plt.show()
```



In [42]:
```
k.plot(kind='bar',stacked=True)
```

Out[42]: <AxesSubplot:xlabel='Discount'>



In [ ]:
```
# Thank You.
```