sales prediction eda

In [1]:

```
import pandas as pd            #imporitng libraries
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:

```
sales = pd.read_csv('train.csv')        #import data
sales
```

Out[2]:

	Row ID	Order ID	Order Date	Ship Date	Ship Mode	Customer ID	Customer Name	Segment	Country	City	State	Po C
0	1	CA-2017-152156	08/11/2017	11/11/2017	Second Class	CG-12520	Claire Gute	Consumer	United States	Henderson	Kentucky	424
1	2	CA-2017-152156	08/11/2017	11/11/2017	Second Class	CG-12520	Claire Gute	Consumer	United States	Henderson	Kentucky	424
2	3	CA-2017-138688	12/06/2017	16/06/2017	Second Class	DV-13045	Darrin Van Huff	Corporate	United States	Los Angeles	California	900
3	4	US-2016-108966	11/10/2016	18/10/2016	Standard Class	SO-20335	Sean O'Donnell	Consumer	United States	Fort Lauderdale	Florida	333
4	5	US-2016-108966	11/10/2016	18/10/2016	Standard Class	SO-20335	Sean O'Donnell	Consumer	United States	Fort Lauderdale	Florida	333
...	...	...	...	...	...	...	...	...	...	...	...	
9795	9796	CA-2017-125920	21/05/2017	28/05/2017	Standard Class	SH-19975	Sally Hughsby	Corporate	United States	Chicago	Illinois	606
9796	9797	CA-2016-128608	12/01/2016	17/01/2016	Standard Class	CS-12490	Cindy Schnelling	Corporate	United States	Toledo	Ohio	436
9797	9798	CA-2016-128608	12/01/2016	17/01/2016	Standard Class	CS-12490	Cindy Schnelling	Corporate	United States	Toledo	Ohio	436
9798	9799	CA-2016-128608	12/01/2016	17/01/2016	Standard Class	CS-12490	Cindy Schnelling	Corporate	United States	Toledo	Ohio	436
9799	9800	CA-2016-128608	12/01/2016	17/01/2016	Standard Class	CS-12490	Cindy Schnelling	Corporate	United States	Toledo	Ohio	436

Row ID	Order ID	Order Date	Ship Date	Ship Mode	Customer ID	Customer Name	Segment	Country	City	State	Po C

9800 rows × 18 columns

In [ ]:

```

```

1. variabe identification

In [3]:

```
sales.dtypes
```

Out[3]:

```
Row ID             int64
Order ID          object
Order Date        object
Ship Date         object
Ship Mode         object
Customer ID       object
Customer Name     object
Segment           object
Country           object
City              object
State             object
Postal Code      float64
Region            object
Product ID        object
Category          object
Sub-Category      object
Product Name      object
Sales            float64
dtype: object
```

categorical data :

1.Segment 2.country 3.city
4.state
5.region
6.category
7.sub category 8.ship mode

continuous data :

1.sales 2.Order date 3.Ship date

rest all are discrete varaibles

the features with ID's are not required for eda so just drop the "ID" columns for sales data

In [4]:

```
sales.columns = sales.columns.str.replace(' ', '_')     #adding "_" instead of " " for ou
r convience
```

In [5]:

```
sales.drop(columns=['Order_ID','Row_ID','Customer_ID','Product_ID'],inplace=True)
```

In [6]:

```
sales
```

Out[6]:

	Order_Date	Ship_Date	Ship_Mode	Customer_Name	Segment	Country	City	State	Postal_Code	Region
0	08/11/2017	11/11/2017	Second Class	Claire Gute	Consumer	United States	Henderson	Kentucky	42420.0	South
1	08/11/2017	11/11/2017	Second Class	Claire Gute	Consumer	United States	Henderson	Kentucky	42420.0	South
2	12/06/2017	16/06/2017	Second Class	Darrin Van Huff	Corporate	United States	Los Angeles	California	90036.0	West
3	11/10/2016	18/10/2016	Standard Class	Sean O'Donnell	Consumer	United States	Fort Lauderdale	Florida	33311.0	South
4	11/10/2016	18/10/2016	Standard Class	Sean O'Donnell	Consumer	United States	Fort Lauderdale	Florida	33311.0	South
...	...	...	...	...	...	...	...	...	...	...
9795	21/05/2017	28/05/2017	Standard Class	Sally Hughsby	Corporate	United States	Chicago	Illinois	60610.0	Central
9796	12/01/2016	17/01/2016	Standard Class	Cindy Schnelling	Corporate	United States	Toledo	Ohio	43615.0	East
9797	12/01/2016	17/01/2016	Standard Class	Cindy Schnelling	Corporate	United States	Toledo	Ohio	43615.0	East
9798	12/01/2016	17/01/2016	Standard Class	Cindy Schnelling	Corporate	United States	Toledo	Ohio	43615.0	East
9799	12/01/2016	17/01/2016	Standard Class	Cindy Schnelling	Corporate	United States	Toledo	Ohio	43615.0	East

9800 rows × 14 columns

now we can perform eda on the repective features

2. Univariate Analysis

univariate analysis is usually done on categorical variables using countplot and continuous variable(sales) using distplot.

Before that we need to convert the date columns to yyyy-mm-dd format for easy plotting of date columns.

In [7]:

```
sales['Order_Date'] = pd.to_datetime(sales['Order_Date'])
```

In [8]:

```
sales['Ship_Date'] = pd.to_datetime(sales['Ship_Date'])
```

```
sales['Ship_Date'] = pd.to_datetime(sales['Ship_Date'])
```

In [9]:

```
sales
```

Out[9]:

	Order_Date	Ship_Date	Ship_Mode	Customer_Name	Segment	Country	City	State	Postal_Code	Region
0	2017-08-11	2017-11-11	Second Class	Claire Gute	Consumer	United States	Henderson	Kentucky	42420.0	South
1	2017-08-11	2017-11-11	Second Class	Claire Gute	Consumer	United States	Henderson	Kentucky	42420.0	South
2	2017-12-06	2017-06-16	Second Class	Darrin Van Huff	Corporate	United States	Los Angeles	California	90036.0	West
3	2016-11-10	2016-10-18	Standard Class	Sean O'Donnell	Consumer	United States	Fort Lauderdale	Florida	33311.0	South
4	2016-11-10	2016-10-18	Standard Class	Sean O'Donnell	Consumer	United States	Fort Lauderdale	Florida	33311.0	South
...	...	...	...	...	...	...	...	...	...	...
9795	2017-05-21	2017-05-28	Standard Class	Sally Hughsby	Corporate	United States	Chicago	Illinois	60610.0	Central
9796	2016-12-01	2016-01-17	Standard Class	Cindy Schnelling	Corporate	United States	Toledo	Ohio	43615.0	East
9797	2016-12-01	2016-01-17	Standard Class	Cindy Schnelling	Corporate	United States	Toledo	Ohio	43615.0	East
9798	2016-12-01	2016-01-17	Standard Class	Cindy Schnelling	Corporate	United States	Toledo	Ohio	43615.0	East
9799	2016-12-01	2016-01-17	Standard Class	Cindy Schnelling	Corporate	United States	Toledo	Ohio	43615.0	East

9800 rows × 14 columns

for categorical variables :

In [10]:

```
result1 = sales.groupby(["City"])['Sales'].aggregate(np.sum).reset_index().sort_values('
Sales',ascending = False).head(20)
result2 = sales.groupby(["State"])['Sales'].aggregate(np.sum).reset_index().sort_values(
'Sales',ascending = False).head(20)
```

In [11]:

```
result1
```

Out[11]:

	City	Sales
327	New York City	252462.5470
265	Los Angeles	173420.1810
450	Seattle	116106.3220
436	San Francisco	109041.1200
372	Philadelphia	108841.7490
207	Houston	63956.1428
80	Chicago	47820.1330
435	San Diego	47521.0290
216	Jacksonville	44713.1830
123	Detroit	42446.9440
462	Springfield	41827.8100
94	Columbus	38662.5630
328	Newark	28448.0490
93	Columbia	25283.3240
215	Jackson	24963.8580
233	Lafayette	24944.2800
432	San Antonio	21843.5280
60	Burlington	21668.0820
16	Arlington	20214.5320
109	Dallas	20127.9482

In [12]:

```
result2
```

Out[12]:

	State	Sales
3	California	446306.4635
30	New York	306361.1470
41	Texas	168572.5322
45	Washington	135206.8500
36	Pennsylvania	116276.6500
8	Florida	88436.5320
11	Illinois	79236.5170
20	Michigan	76136.0740
33	Ohio	75130.3500
44	Virginia	70636.7200
31	North Carolina	55165.9640
12	Indiana	48718.4000
9	Georgia	48219.1100
15	Kentucky	36458.3900
1	Arizona	35272.6570
28	New Jersey	34610.9720

	State	Sales
4	Colorado	31841.[illegible]
47	Wisconsin	31173.4300
40	Tennessee	30661.8730
21	Minnesota	29863.1500

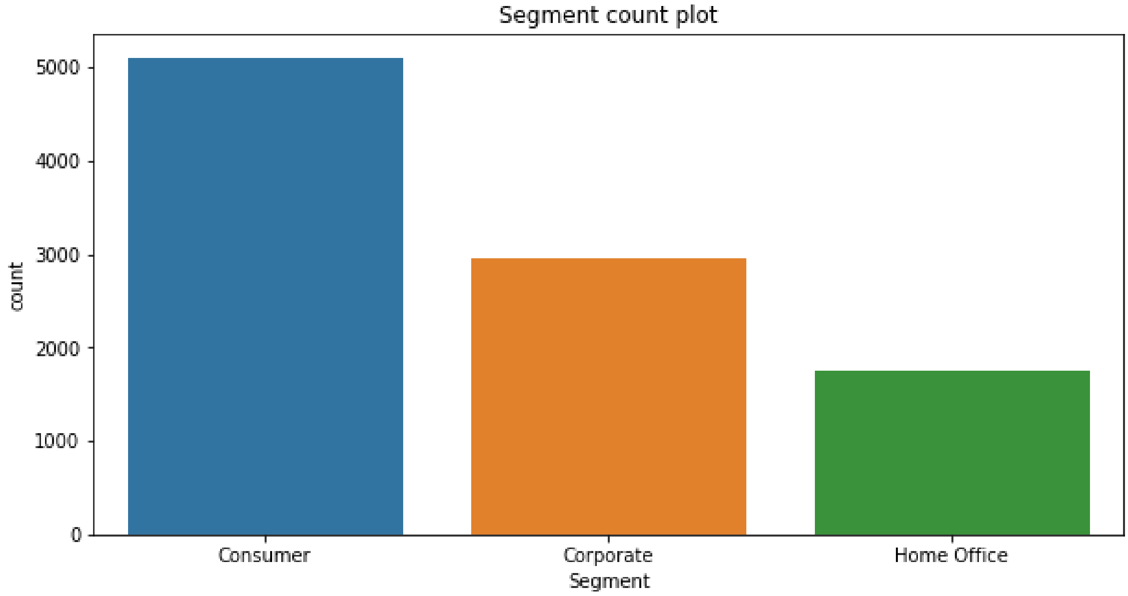
top 20 cities and states dataframes are made as the countplot would be very confusing for large data.

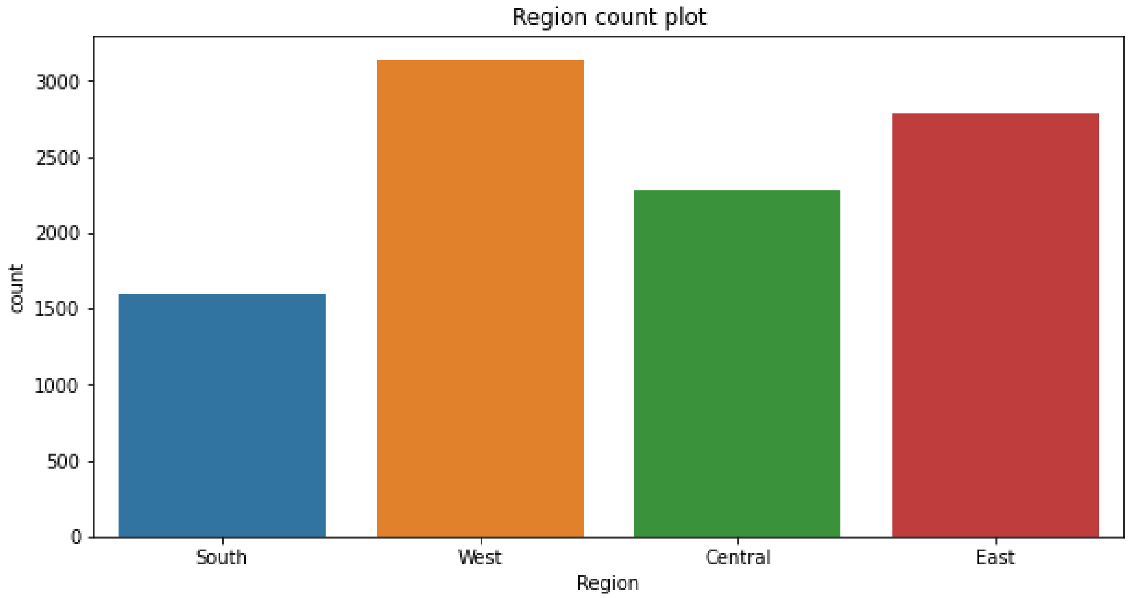
In [13]:

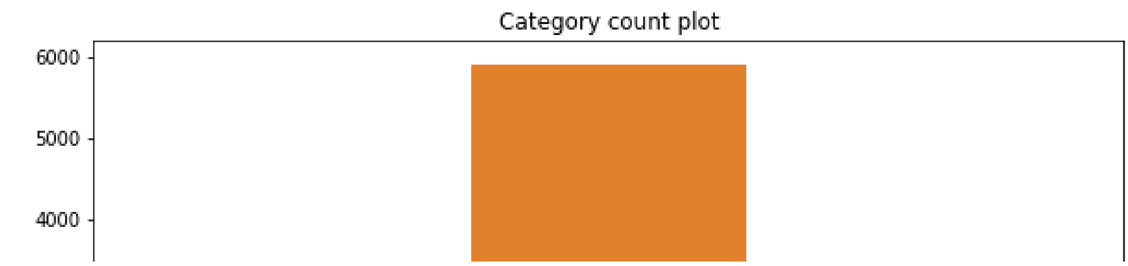
```
cols = ['Segment','Region','Category','Sub-Category','Ship_Mode']
```

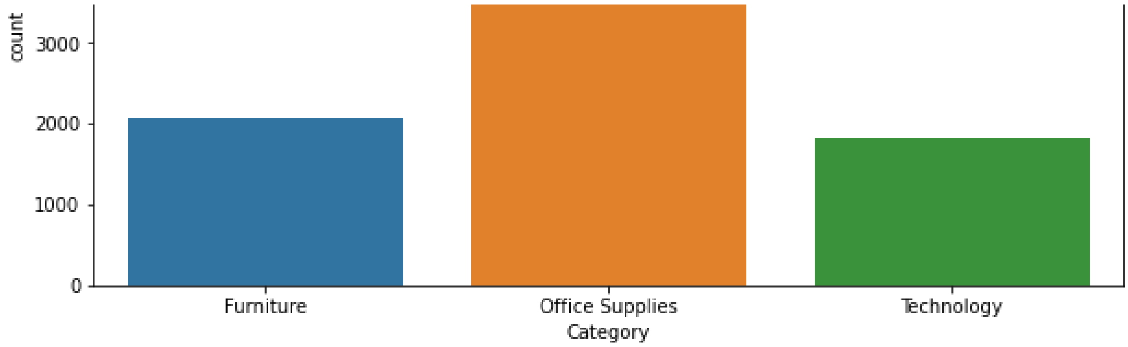
In [14]:

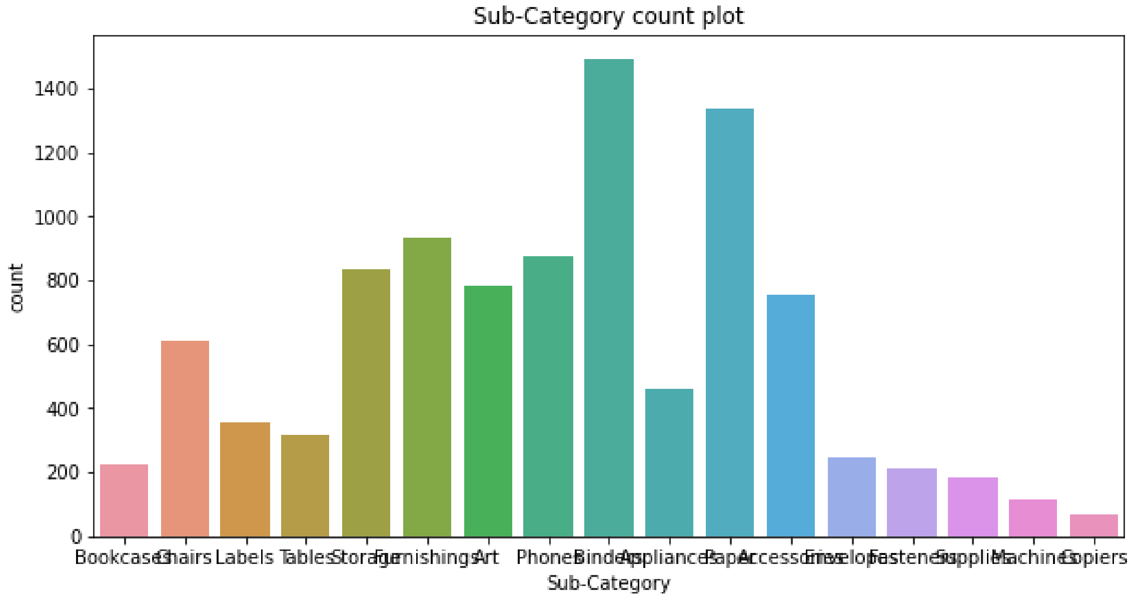
```
for i in cols:
    plt.figure(figsize=(10,5))
    plt.title(i+" count plot")
    sns.countplot(sales[i])
    plt.show()
```

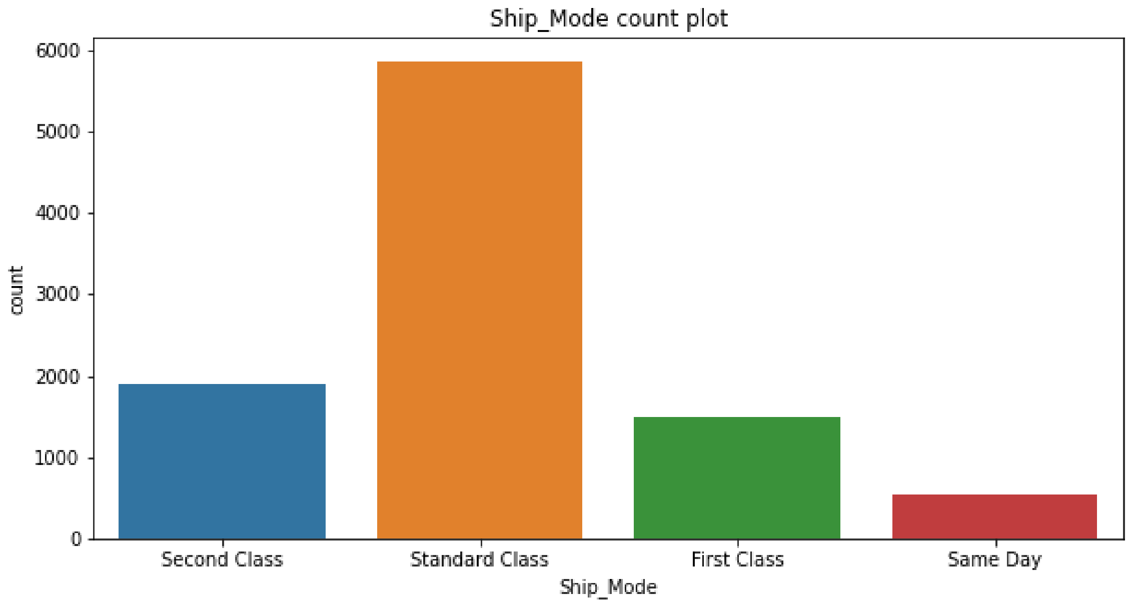












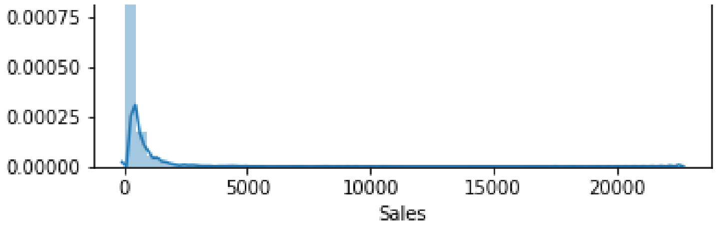
for continuous variable(sales) :

In [15]:

```
sns.distplot(sales["Sales"])
```

Out[15]:

<matplotlib.axes._subplots.AxesSubplot at 0x7f5212f162e0>

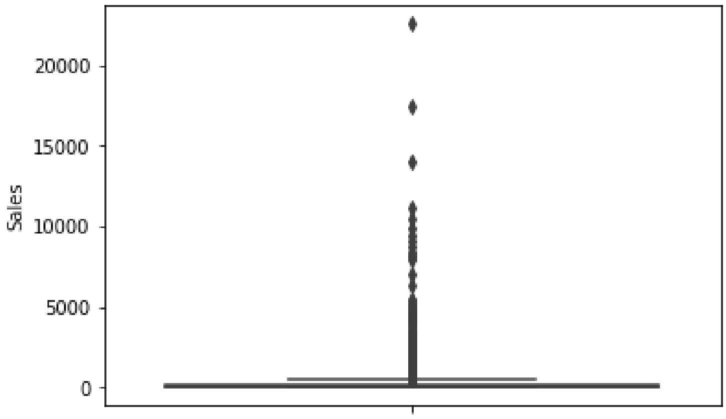


In [16]:

```
sns.boxplot(y='Sales',data=sales)
```

Out[16]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f5212e5e460>
```



from the above distplot we can observe that sales are *highly right skewed* with so many outliers

3. bivariate analysis

categorical vs categorical :

In [17]:

```
pd.crosstab(sales['Sub-Category'],sales['Category'])
```

Out[17]:

Category	**Furniture**	**Office Supplies**	**Technology**
Sub-Category			
Accessories	0	0	756
Appliances	0	459	0
Art	0	785	0
Binders	0	1492	0
Bookcases	226	0	0
Chairs	607	0	0
Copiers	0	0	66
Envelopes	0	248	0
Fasteners	0	214	0
Furnishings	931	0	0
Labels	0	357	0
Machines	0	0	115
Paper	0	1338	0
Phones	0	0	876

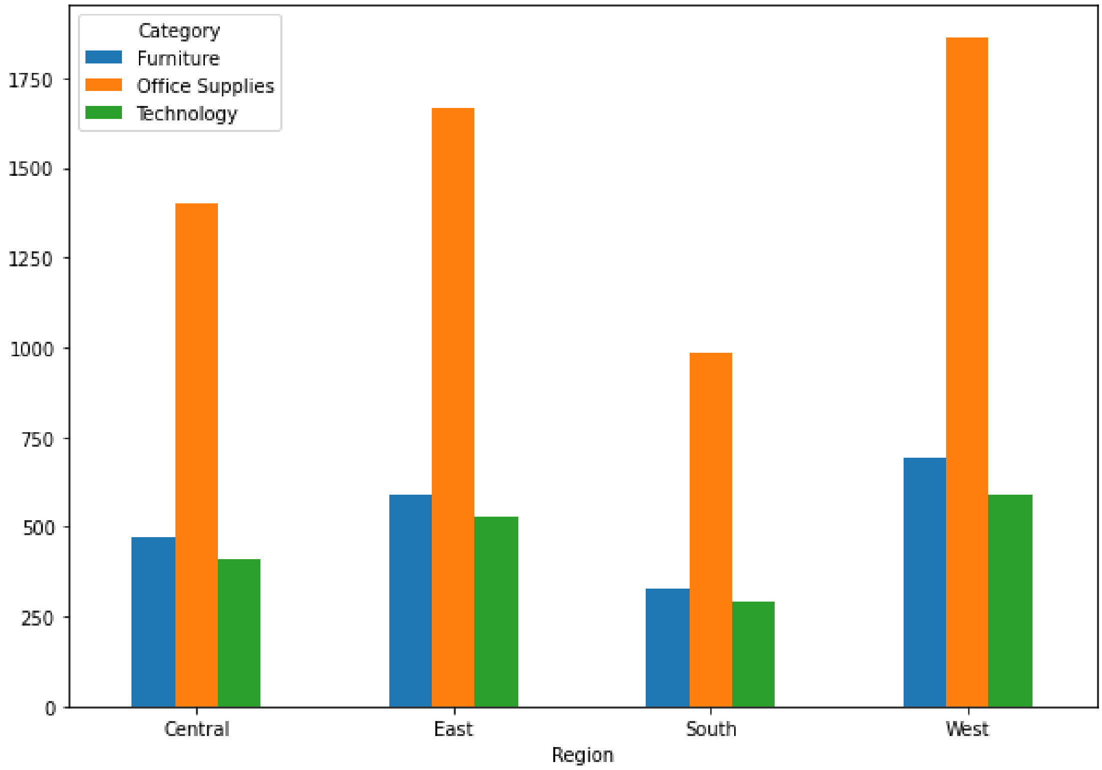
Category / Sub-Category	Furniture	Office Supplies	Technology
Storage	0	838	0
Supplies	0	184	0
Tables	314	0	0

In [18]:

```
pd.crosstab( sales['Region'],sales['Category']).plot(kind='bar',stacked=False, figsize=(
10,7))
plt.xticks(rotation=0)
plt.plot()
```

Out[18]:

```
[]
```



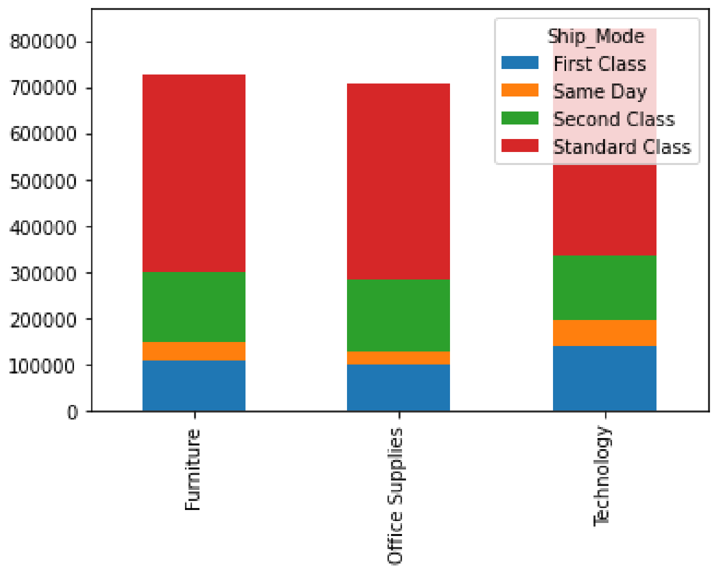
it is very clear that in all the regions office supplies are sold more

In [19]:

```
pd.crosstab(index=sales["Category"],columns=sales["Ship_Mode"],values=sales["Sales"],agg
func="sum").plot(kind="bar",stacked=True)
```

Out[19]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f5212d29e50>
```



Category

in all the segments customers prefer standard class shipment mode but significantly in technology sector same day shipment mode is high that other sectors.

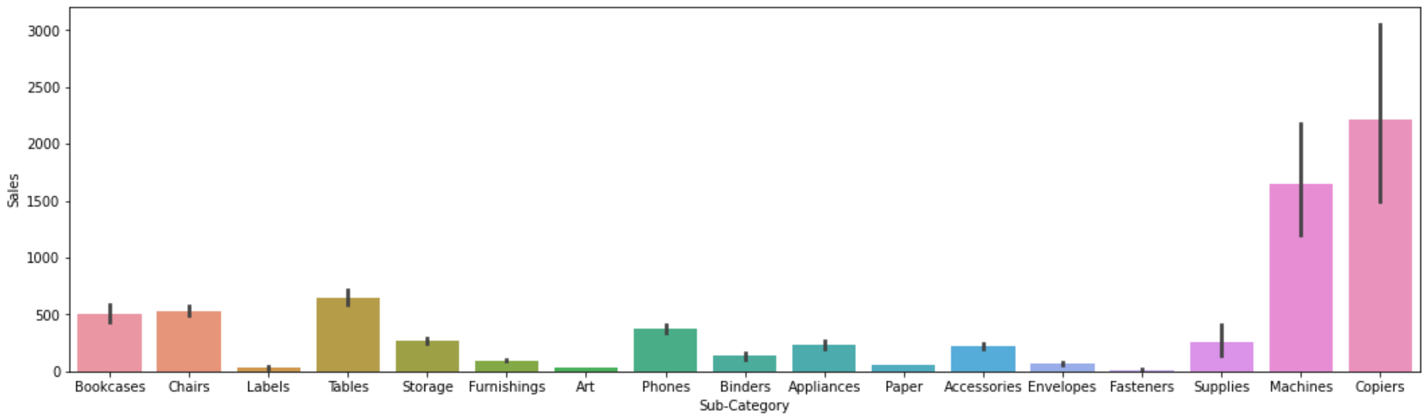
categorical vs continuous :

In [20]:

```
axes,fig=plt.subplots(0,1,figsize=(18,5))
sns.barplot("Sub-Category","Sales",data=sales)
```

Out[20]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f5212d02160>
```



1.From the above graph we can observe that sales are very high in copiers, machines sub-categories

2.And literally people are not interested to buy products from fasteners, paper, art,labels, furnishings, envelpoes categories, so better to stop these categry sales and concentrate on increasing categories.

top 20 products :

In [21]:

```
topprods=pd.DataFrame(sales.groupby('Product_Name').sum()['Sales'])
topprods.sort_values(by=['Sales'], inplace=True, ascending=False)
```

In [22]:

```
top20=topprods.head(20)
```

In [23]:

```
top20
```

Out[23]:

	Sales
Product_Name	
Canon imageCLASS 2200 Advanced Copier	61599.8240
Fellowes PB500 Electric Punch Plastic Comb Binding Machine with Manual Bind	27453.3840
Cisco TelePresence System EX90 Videoconferencing Unit	22638.4800
HON 5400 Series Task Chairs for Big and Tall	21870.5760
GBC DocuBind TL300 Electric Binding System	19823.4790
GBC Ibimaster 500 Manual ProClick Binding System	19024.5000
Hewlett Packard LaserJet 3310 Copier	18839.6860

Product_Name	Sales
HP Designjet T520 Inkjet Large Format Printer - 24" Color	18374.8950
GBC DocuBind P400 Electric Binding System	17965.0680
High Speed Automatic Electric Letter Opener	17030.3120
Lexmark MX611dhe Monochrome Laser Printer	16829.9010
Martin Yale Chadless Opener Electric Letter Opener	16656.2000
Ibico EPK-21 Electric Binding System	15875.9160
Riverside Palais Royal Lawyers Bookcase, Royale Cherry Finish	15610.9656
3D Systems Cube Printer, 2nd Generation, Magenta	14299.8900
Samsung Galaxy Mega 6.3	13943.6680
Apple iPhone 5	12996.6000
Bretford Rectangular Conference Table Tops	12995.2915
Global Troy Executive Leather Low-Back Tilter	12975.3820
SAFCO Arco Folding Chair	11572.7800

continuous vs continuous :

in this part of analysis we only deal with dates as per the data so different insights we could draw from the dates are :

1.overall sales trend

2.Month over Month (MoM) growth

3.Year over Year (YoY) growth

overall sales trend :

In [24]:

```
sales['Month_Year']=sales['Order_Date'].apply(lambda x:x.strftime('%Y-%m'))
sales
```

Out[24]:

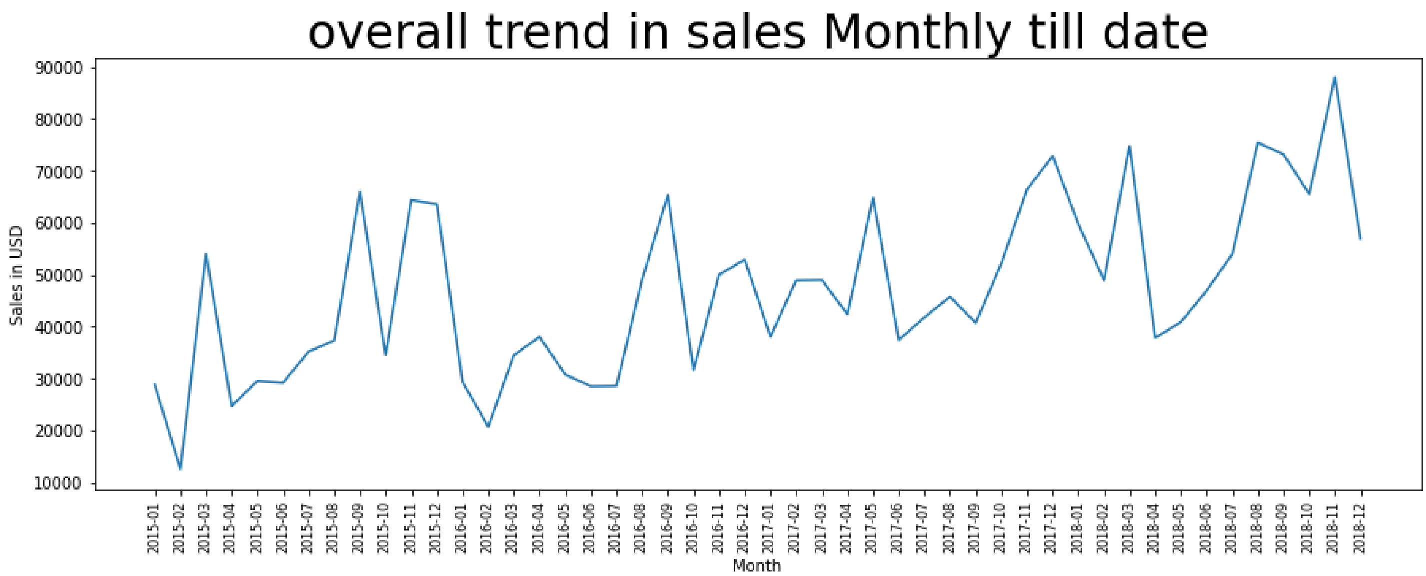
	Order_Date	Ship_Date	Ship_Mode	Customer_Name	Segment	Country	City	State	Postal_Code	Region
0	2017-08-11	2017-11-11	Second Class	Claire Gute	Consumer	United States	Henderson	Kentucky	42420.0	South
1	2017-08-11	2017-11-11	Second Class	Claire Gute	Consumer	United States	Henderson	Kentucky	42420.0	South
2	2017-12-06	2017-06-16	Second Class	Darrin Van Huff	Corporate	United States	Los Angeles	California	90036.0	West
3	2016-11-10	2016-10-18	Standard Class	Sean O'Donnell	Consumer	United States	Fort Lauderdale	Florida	33311.0	South
4	2016-11-10	2016-10-18	Standard Class	Sean O'Donnell	Consumer	United States	Fort Lauderdale	Florida	33311.0	South
...	...	...	...	...	...	...	...	...	...	...

	Order_Date	Ship_Date	Ship_Mode	Customer_Name	Segment	Country	City	State	Postal_Code	Region
9795	2017-05-21	2017-05-28	Standard Class	Sally Hughsby	Corporate	United States	Chicago	Illinois	60610.0	Central
9796	2016-12-01	2016-01-17	Standard Class	Cindy Schnelling	Corporate	United States	Toledo	Ohio	43615.0	East
9797	2016-12-01	2016-01-17	Standard Class	Cindy Schnelling	Corporate	United States	Toledo	Ohio	43615.0	East
9798	2016-12-01	2016-01-17	Standard Class	Cindy Schnelling	Corporate	United States	Toledo	Ohio	43615.0	East
9799	2016-12-01	2016-01-17	Standard Class	Cindy Schnelling	Corporate	United States	Toledo	Ohio	43615.0	East

9800 rows × 15 columns

In [25]:

```
ovrsls = sales.groupby('Month_Year').sum()
months = [month for month, sales in sales.groupby('Month_Year')]
plt.figure(figsize=(15,5))
plt.plot(months,ovrsls['Sales'])
plt.xticks(months, rotation='vertical', size = 8)
plt.ylabel('Sales in USD')
plt.xlabel('Month')
plt.title('overall trend in sales Monthly till date',fontsize=30)
plt.show()
```



Month over Month growth :

In [26]:

```
MoM=pd.DataFrame(ovrsls['Sales'])
```

In [27]:

```
MoM['Last_Month']=np.roll(MoM['Sales'],1)
MoM=MoM.drop(MoM.index[0])    #drop the first index as we dont know the previous sales
```

In [28]:

```
MoM['Growth']=(MoM['Sales']/MoM['Last_Month'])-1
MoM.head()          #calculating growth using current and previous month sales
```

Out[28]:

	Sales	Last_Month	Growth
Month_Year			
2015-02	12588.4840	28828.254	-0.563328
2015-03	54027.6920	12588.484	3.291835
2015-04	24710.0160	54027.692	-0.542642
2015-05	29520.4900	24710.016	0.194677
2015-06	29181.3346	29520.490	-0.011489

In [29]:

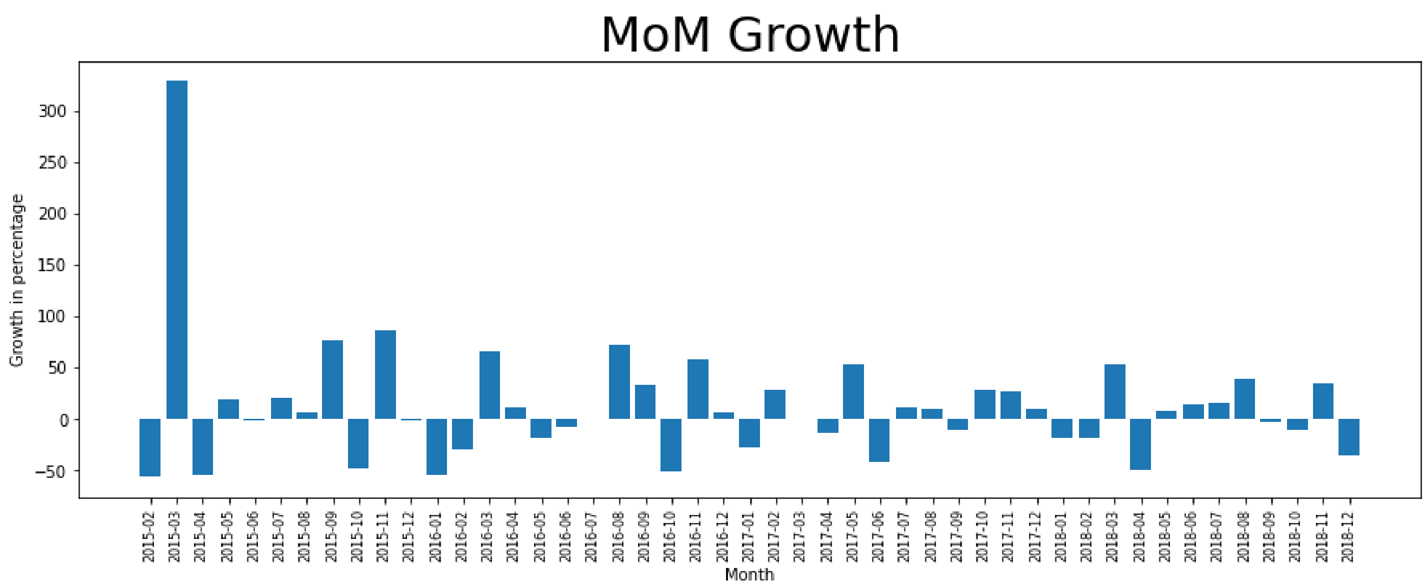
```
MoMsls = MoM.drop(columns = ["Sales", "Last_Month"])
MoMsls['Months'] =MoMsls.index
MoMsls.reset_index(drop=True, inplace=True)
MoMsls.head()                #dataframe of every month and its growth.
```

Out[29]:

	Growth	Months
0	-0.563328	2015-02
1	3.291835	2015-03
2	-0.542642	2015-04
3	0.194677	2015-05
4	-0.011489	2015-06

In [30]:

```
plt.figure(figsize=(15,5))
plt.bar(MoMsls['Months'],MoMsls['Growth']*100)   #percentage of growth.
plt.xticks(MoMsls['Months'], rotation='vertical', size = 8)
plt.ylabel('Growth in percentage')
plt.xlabel('Month')
plt.title("MoM Growth",fontsize=30)
plt.show()
```



Year over Year growth :

In [31]:

```
YoY = pd.DataFrame(sales.groupby('Month_Year').sum()['Sales'])
YoY['Last_Year'] = np.roll(YoY['Sales'],12)
```

In [32]:

```
YoY = YoY.drop(YoY.index[0:12])    #drop 2015 sales as we dont know pervious year sales
YoY['Growth'] = (YoY['Sales']/YoY['Last_Year'])-1
YoY.head()
```

Out[32]:

	Sales	Last_Year	Growth
Month_Year			
2016-01	29347.3864	28828.254	0.018008
2016-02	20728.3520	12588.484	0.646612
2016-03	34489.6776	54027.692	-0.361630
2016-04	38056.9685	24710.016	0.540143
2016-05	30761.5585	29520.490	0.042041

In [33]:

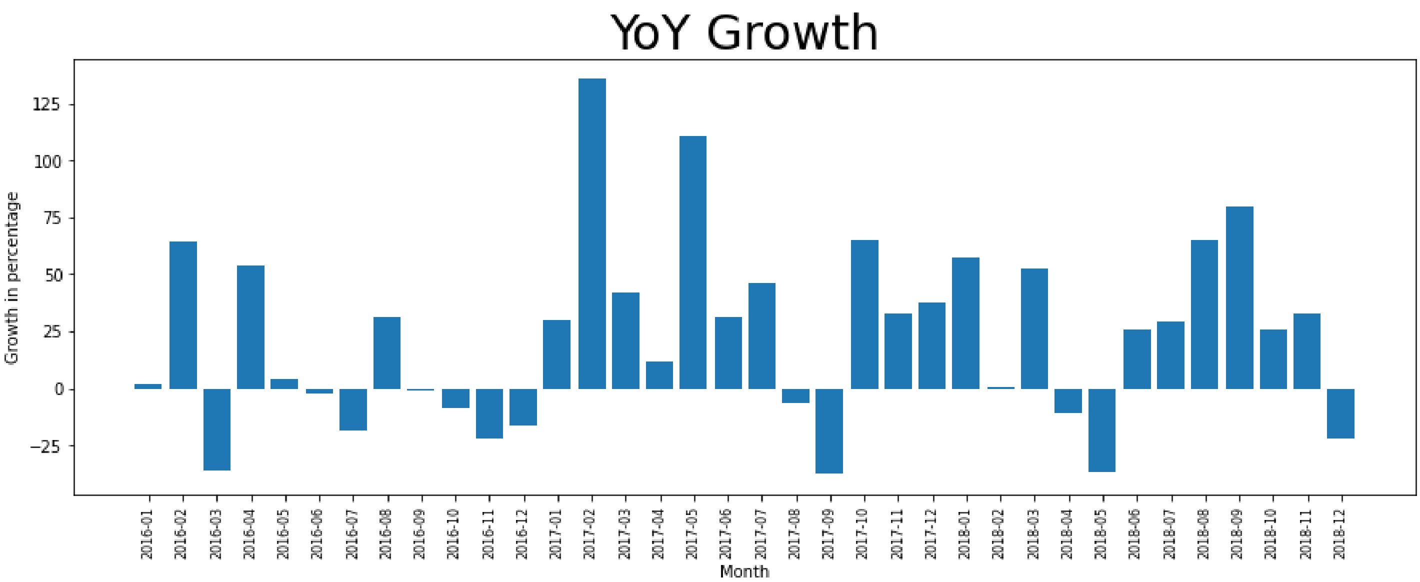
```
YoYsls = YoY.drop(columns = ["Sales", "Last_Year"])
YoYsls['Month_Year'] = YoYsls.index
YoYsls.reset_index(drop=True, inplace=True)
YoYsls.head()
```

Out[33]:

	Growth	Month_Year
0	0.018008	2016-01
1	0.646612	2016-02
2	-0.361630	2016-03
3	0.540143	2016-04
4	0.042041	2016-05

In [34]:

```
plt.figure(figsize=(15,5))
plt.bar(YoYsls['Month_Year'],YoYsls['Growth']*100)
plt.xticks(YoYsls['Month_Year'], rotation='vertical', size = 8)
plt.ylabel('Growth in percentage')
plt.xlabel('Month')
plt.title("YoY Growth", fontsize=30)
plt.show()
```



In [35]:

```
sales
```

Out[35]:

	Order_Date	Ship_Date	Ship_Mode	Customer_Name	Segment	Country	City	State	Postal_Code	Region
0	2017-08-11	2017-11-11	Second Class	Claire Gute	Consumer	United States	Henderson	Kentucky	42420.0	South
1	2017-08-11	2017-11-11	Second Class	Claire Gute	Consumer	United States	Henderson	Kentucky	42420.0	South
2	2017-12-06	2017-06-16	Second Class	Darrin Van Huff	Corporate	United States	Los Angeles	California	90036.0	West
3	2016-11-10	2016-10-18	Standard Class	Sean O'Donnell	Consumer	United States	Fort Lauderdale	Florida	33311.0	South
4	2016-11-10	2016-10-18	Standard Class	Sean O'Donnell	Consumer	United States	Fort Lauderdale	Florida	33311.0	South
...	...	...	...	...	...	...	...	...	...	...
9795	2017-05-21	2017-05-28	Standard Class	Sally Hughsby	Corporate	United States	Chicago	Illinois	60610.0	Central
9796	2016-12-01	2016-01-17	Standard Class	Cindy Schnelling	Corporate	United States	Toledo	Ohio	43615.0	East
9797	2016-12-01	2016-01-17	Standard Class	Cindy Schnelling	Corporate	United States	Toledo	Ohio	43615.0	East
9798	2016-12-01	2016-01-17	Standard Class	Cindy Schnelling	Corporate	United States	Toledo	Ohio	43615.0	East
9799	2016-12-01	2016-01-17	Standard Class	Cindy Schnelling	Corporate	United States	Toledo	Ohio	43615.0	East

9800 rows × 15 columns

4. missing values treatment and visualizing time series data

drop the missing values

In [36]:

```
sales.isnull().sum()     #only 11 missing values in postal code so just drop the rows with dropna()
```

Out[36]:

```
Order_Date        0
Ship_Date         0
Ship_Mode         0
Customer_Name     0
Segment           0
Country           0
City              0
State             0
Postal_Code      11
Region            0
Category          0
Sub-Category      0
Product_Name      0
Sales             0
Month_Year        0
dtype: int64
```

In [37]:

```
sales.dropna(inplace=True)
```

In [38]:

```
sales
```

Out[38]:

	Order_Date	Ship_Date	Ship_Mode	Customer_Name	Segment	Country	City	State	Postal_Code	Region
0	2017-08-11	2017-11-11	Second Class	Claire Gute	Consumer	United States	Henderson	Kentucky	42420.0	South
1	2017-08-11	2017-11-11	Second Class	Claire Gute	Consumer	United States	Henderson	Kentucky	42420.0	South
2	2017-12-06	2017-06-16	Second Class	Darrin Van Huff	Corporate	United States	Los Angeles	California	90036.0	West
3	2016-11-10	2016-10-18	Standard Class	Sean O'Donnell	Consumer	United States	Fort Lauderdale	Florida	33311.0	South
4	2016-11-10	2016-10-18	Standard Class	Sean O'Donnell	Consumer	United States	Fort Lauderdale	Florida	33311.0	South
...	...	...	...	...	...	...	...	...	...	...
9795	2017-05-21	2017-05-28	Standard Class	Sally Hughsby	Corporate	United States	Chicago	Illinois	60610.0	Central
9796	2016-12-01	2016-01-17	Standard Class	Cindy Schnelling	Corporate	United States	Toledo	Ohio	43615.0	East
9797	2016-12-01	2016-01-17	Standard Class	Cindy Schnelling	Corporate	United States	Toledo	Ohio	43615.0	East
9798	2016-12-01	2016-01-17	Standard Class	Cindy Schnelling	Corporate	United States	Toledo	Ohio	43615.0	East

	Order_Date	Ship_Date	Ship_Mode	Customer_Name	Segment	Country	City	State	Postal_Code	Region
9799	2016-12-01	2016-01-17	Standard Class	Cindy Schnelling	Corporate	United States	Toledo	Ohio	43615.0	East

9789 rows × 15 columns

visualize time series plots according to category feature

In [39]:

```
furniture = sales.loc[sales['Category'] == 'Furniture']
officesup = sales.loc[sales['Category'] == 'Office Supplies']
tech = sales.loc[sales['Category'] == 'Technology']
```

In [40]:

```
print(furniture['Order_Date'].min(), furniture['Order_Date'].max())
print(officesup['Order_Date'].min(), officesup['Order_Date'].max())
print(tech['Order_Date'].min(), tech['Order_Date'].max())
```

```
2015-01-03 00:00:00 2018-12-30 00:00:00
2015-01-03 00:00:00 2018-12-30 00:00:00
2015-01-02 00:00:00 2018-12-30 00:00:00
```

In [41]:

```
furniture = furniture.groupby('Order_Date')['Sales'].sum().reset_index()
officesup = officesup.groupby('Order_Date')['Sales'].sum().reset_index()
tech = tech.groupby('Order_Date')['Sales'].sum().reset_index()
```

In [42]:

```
furniture = furniture.set_index('Order_Date')
furniture.index
```

Out[42]:

```
DatetimeIndex(['2015-01-03', '2015-01-06', '2015-01-08', '2015-01-11',
               '2015-01-12', '2015-01-13', '2015-01-14', '2015-01-16',
               '2015-01-19', '2015-01-20',
               ...
               '2018-12-18', '2018-12-19', '2018-12-21', '2018-12-22',
               '2018-12-23', '2018-12-24', '2018-12-25', '2018-12-28',
               '2018-12-29', '2018-12-30'],
              dtype='datetime64[ns]', name='Order_Date', length=877, freq=None)
```

In [43]:

```
officesup = officesup.set_index('Order_Date')
officesup.index
```

Out[43]:

```
DatetimeIndex(['2015-01-03', '2015-01-04', '2015-01-06', '2015-01-07',
               '2015-01-08', '2015-01-09', '2015-01-10', '2015-01-11',
               '2015-01-12', '2015-01-13',
               ...
               '2018-12-21', '2018-12-22', '2018-12-23', '2018-12-24',
               '2018-12-25', '2018-12-26', '2018-12-27', '2018-12-28',
               '2018-12-29', '2018-12-30'],
              dtype='datetime64[ns]', name='Order_Date', length=1142, freq=None)
```

In [44]:

```
tech = tech.set_index('Order_Date')
tech.index
```

Out[44]:

```
DatetimeIndex(['2015-01-02', '2015-01-03', '2015-01-06', '2015-01-07',
```

```
'2015-01-09', '2015-01-11', '2015-01-12', '2015-01-13',
'2015-01-15', '2015-01-16',
...
'2018-12-18', '2018-12-21', '2018-12-22', '2018-12-23',
'2018-12-24', '2018-12-25', '2018-12-27', '2018-12-28',
'2018-12-29', '2018-12-30'],
dtype='datetime64[ns]', name='Order_Date', length=817, freq=None)
```

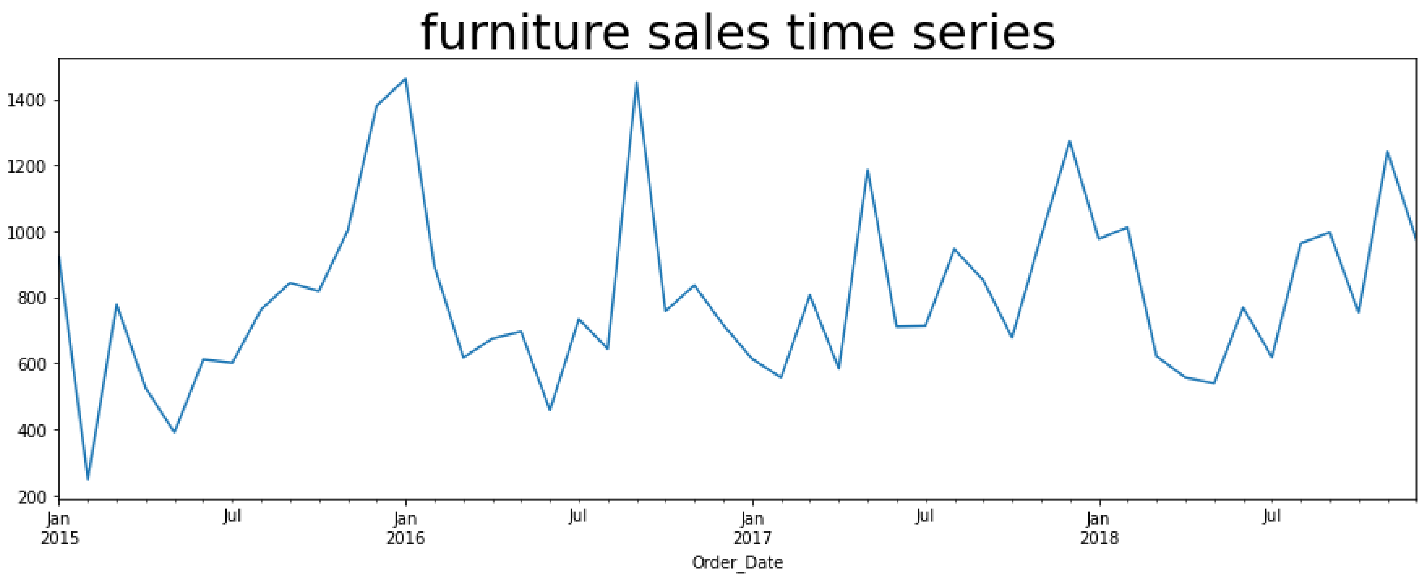
from all the 3 categories office supplies has more orders.

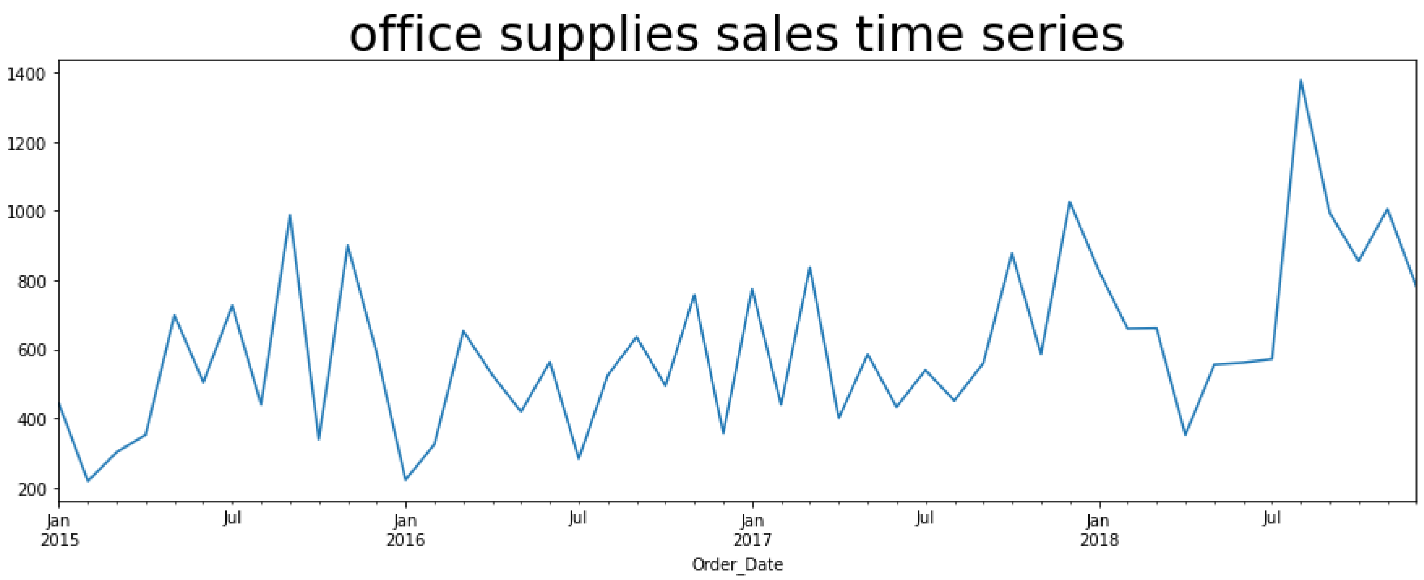
In [45]:

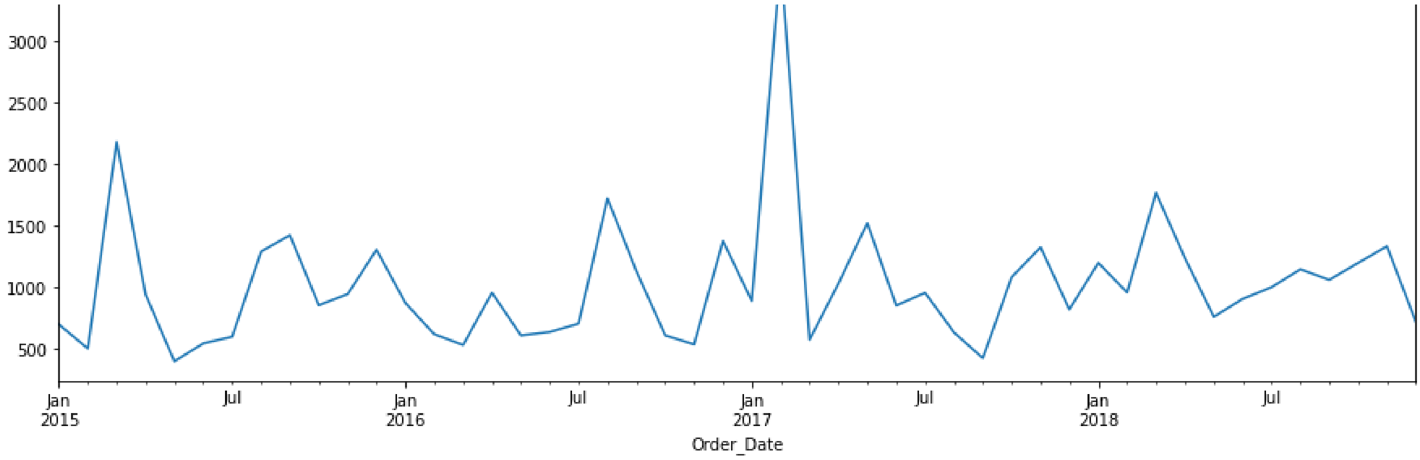
```
furn = furniture['Sales'].resample('MS').mean()
offi = officesup['Sales'].resample('MS').mean()
tec = tech['Sales'].resample('MS').mean()
```

In [46]:

```
furn.plot(figsize=(15, 5))
plt.title('furniture sales time series',fontsize=30)
plt.show()
offi.plot(figsize=(15, 5))
plt.title('office supplies sales time series',fontsize=30)
plt.show()
tec.plot(figsize=(15, 5))
plt.title('technology sales time series',fontsize=30)
plt.show()
```





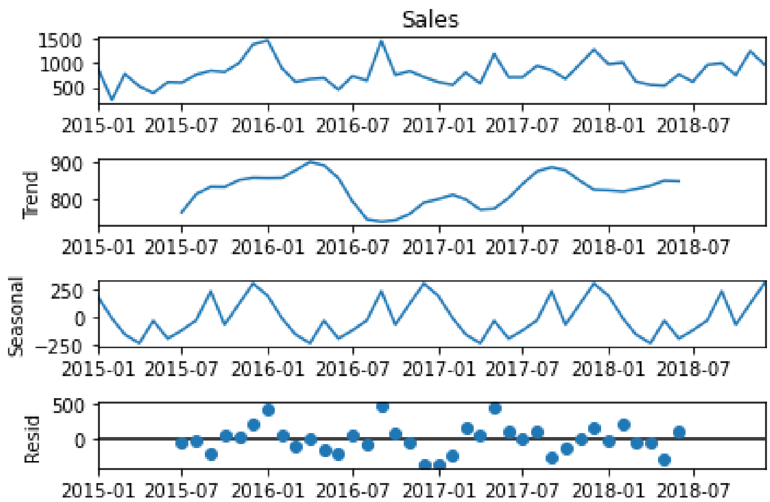


In [47]:

```
import statsmodels.api as sm
```

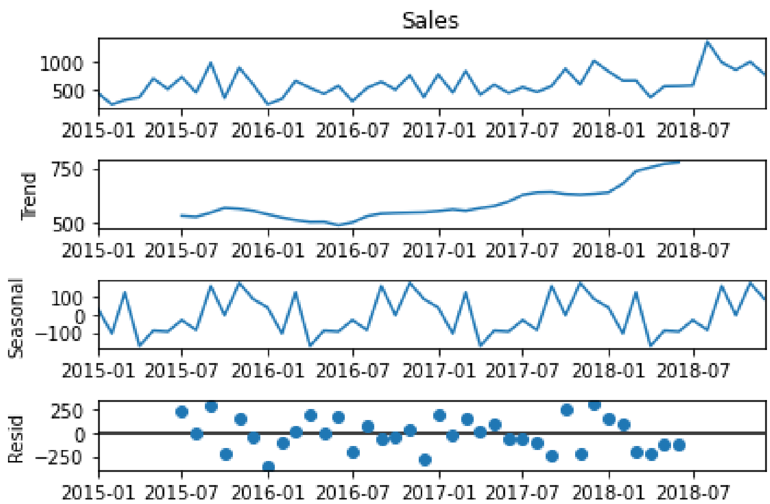
In [48]:

```
from pylab import rcParams

furndecomp = sm.tsa.seasonal_decompose(furn, model='additive')
fig = furndecomp.plot()

plt.show()
```



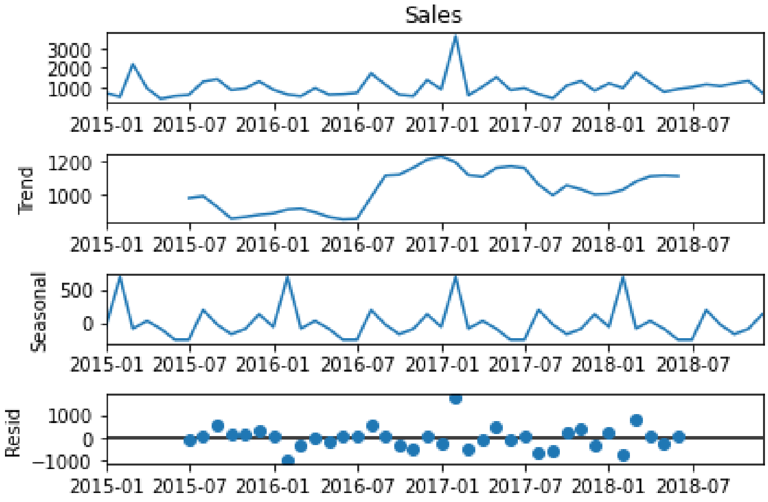
In [49]:

```
offidecomp = sm.tsa.seasonal_decompose(offi, model='additive')
fig = offidecomp.plot()

plt.show()
```



In [50]:

```
tecdecomp = sm.tsa.seasonal_decompose(tec, model='additive')
fig = tecdecomp.plot()

plt.show()
```



In [51]:

```
df = pd.DataFrame(columns=['date','sales'])
```

In [52]:

```
df.date = sales.Order_Date
df.sales = sales.Sales
```

In [53]:

```
df
```

Out[53]:

	date	sales
0	2017-08-11	261.9600
1	2017-08-11	731.9400
2	2017-12-06	14.6200
3	2016-11-10	957.5775
4	2016-11-10	22.3680
...	...	...
9795	2017-05-21	3.7980
9796	2016-12-01	10.3680
9797	2016-12-01	235.1880
9798	2016-12-01	26.3760
9799	2016-12-01	10.3840

9789 rows × 2 columns

In [55]:

```
df.to_csv("date_sales.csv",index=False)
```

In [ ]:

```

```