Sunkara Venkata Krishna Vs2539 N16740683

In [2]:
```
import fbprophet as pro
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
import warnings
warnings.filterwarnings('ignore')
file= pd.read_csv('super_store.csv')
file.head()
```

Out[2]:

	Row ID	Order ID	Order Date	Ship Date	Ship Mode	Customer ID	Customer Name	Segment	Country	C
0	1	CA-2016-152156	08-11-2016	11-11-2016	Second Class	CG-12520	Claire Gute	Consumer	United States	Henders
1	2	CA-2016-152156	08-11-2016	11-11-2016	Second Class	CG-12520	Claire Gute	Consumer	United States	Henders
2	3	CA-2016-138688	12-06-2016	16-06-2016	Second Class	DV-13045	Darrin Van Huff	Corporate	United States	L Ange
3	4	US-2015-108966	11-10-2015	18-10-2015	Standard Class	SO-20335	Sean O'Donnell	Consumer	United States	F Lauderd
4	5	US-2015-108966	11-10-2015	18-10-2015	Standard Class	SO-20335	Sean O'Donnell	Consumer	United States	F Lauderd

5 rows × 21 columns

In [3]:
```
file.isna().sum()
```

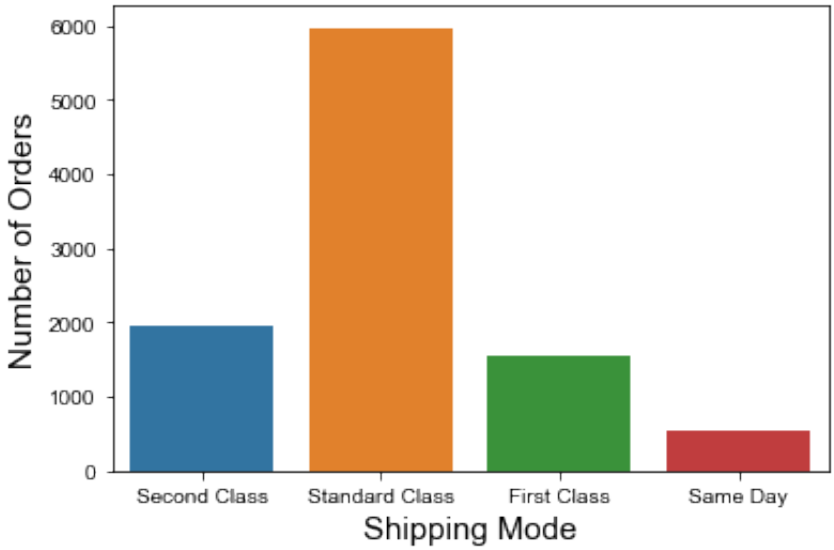
Out[3]:
```
Row ID           0
Order ID         0
Order Date       0
Ship Date        0
Ship Mode        0
Customer ID      0
Customer Name    0
Segment          0
Country          0
City             0
State            0
Postal Code      0
Region           0
Product ID       0
Category         0
Sub-Category     0
Product Name     0
Sales            0
Quantity         0
Discount         0
Profit           0
dtype: int64
```

In [4]:
```
file["Ship Mode"].value_counts()
```

Out[4]:
```
Standard Class    5968
Second Class      1945
First Class       1538
Same Day           543
Name: Ship Mode, dtype: int64
```

In [5]:
```
sns.countplot(x='Ship Mode', data = file)
sns.set(rc={'figure.figsize':(10,10)})
sns.set_palette("Paired")
plt.xlabel("Shipping Mode", fontsize = 15)
plt.ylabel("Number of Orders", fontsize = 15)
```

Out[5]:
```
Text(0, 0.5, 'Number of Orders')
```

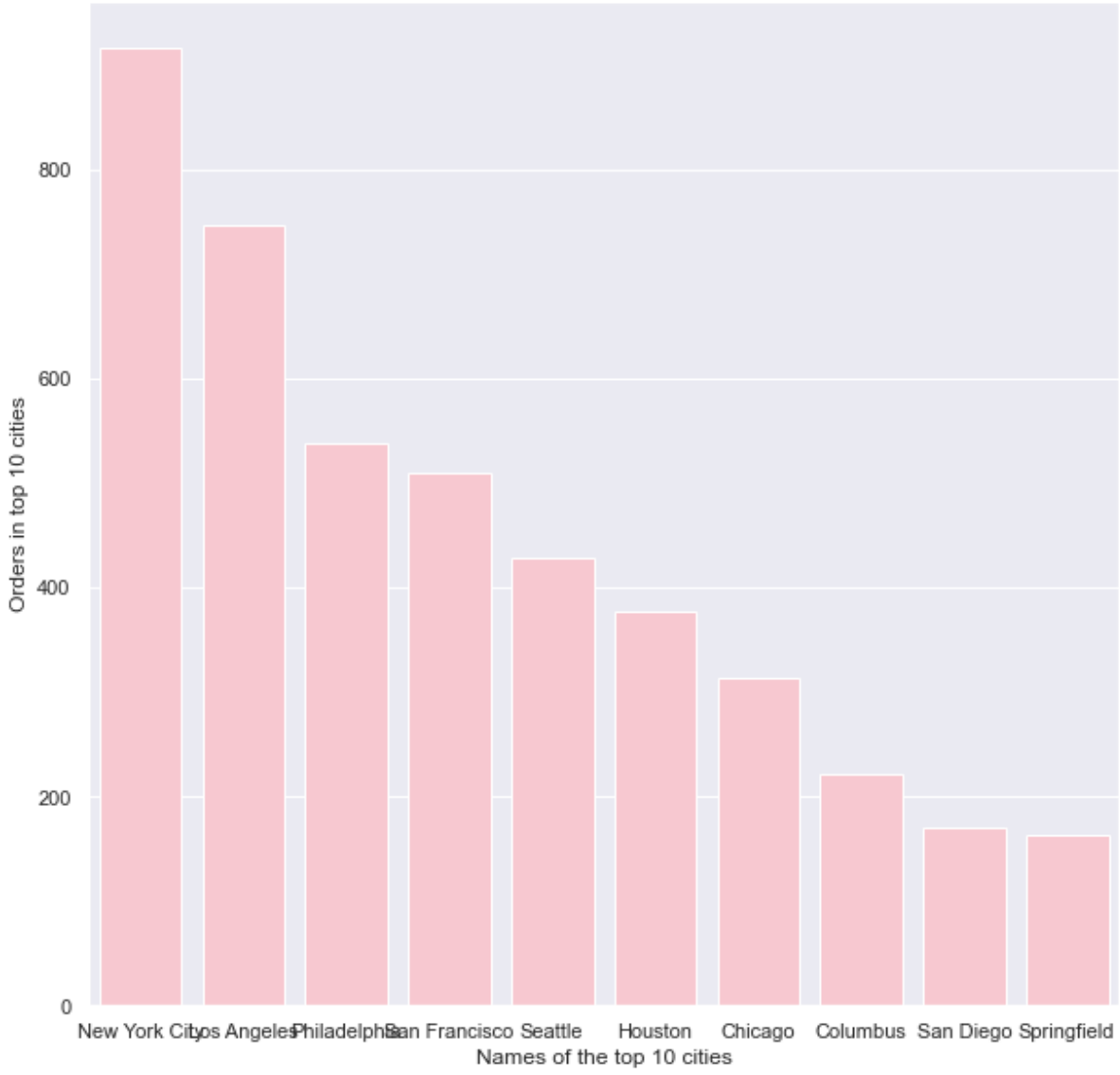


In [6]:
```
file["City"].value_counts().head(5)
```

Out[6]:
```
New York City    915
Los Angeles      747
Philadelphia     537
San Francisco    510
Seattle          428
Name: City, dtype: int64
```

In [7]:
```
sns.barplot(file["City"].value_counts().head(10).index, file["City"
sns.set(rc={'figure.figsize':(15,15)})
plt.xlabel('Names of the top 10 cities')
plt.ylabel('Orders in top 10 cities')
```

Out[7]:
```
Text(0, 0.5, 'Orders in top 10 cities')
```

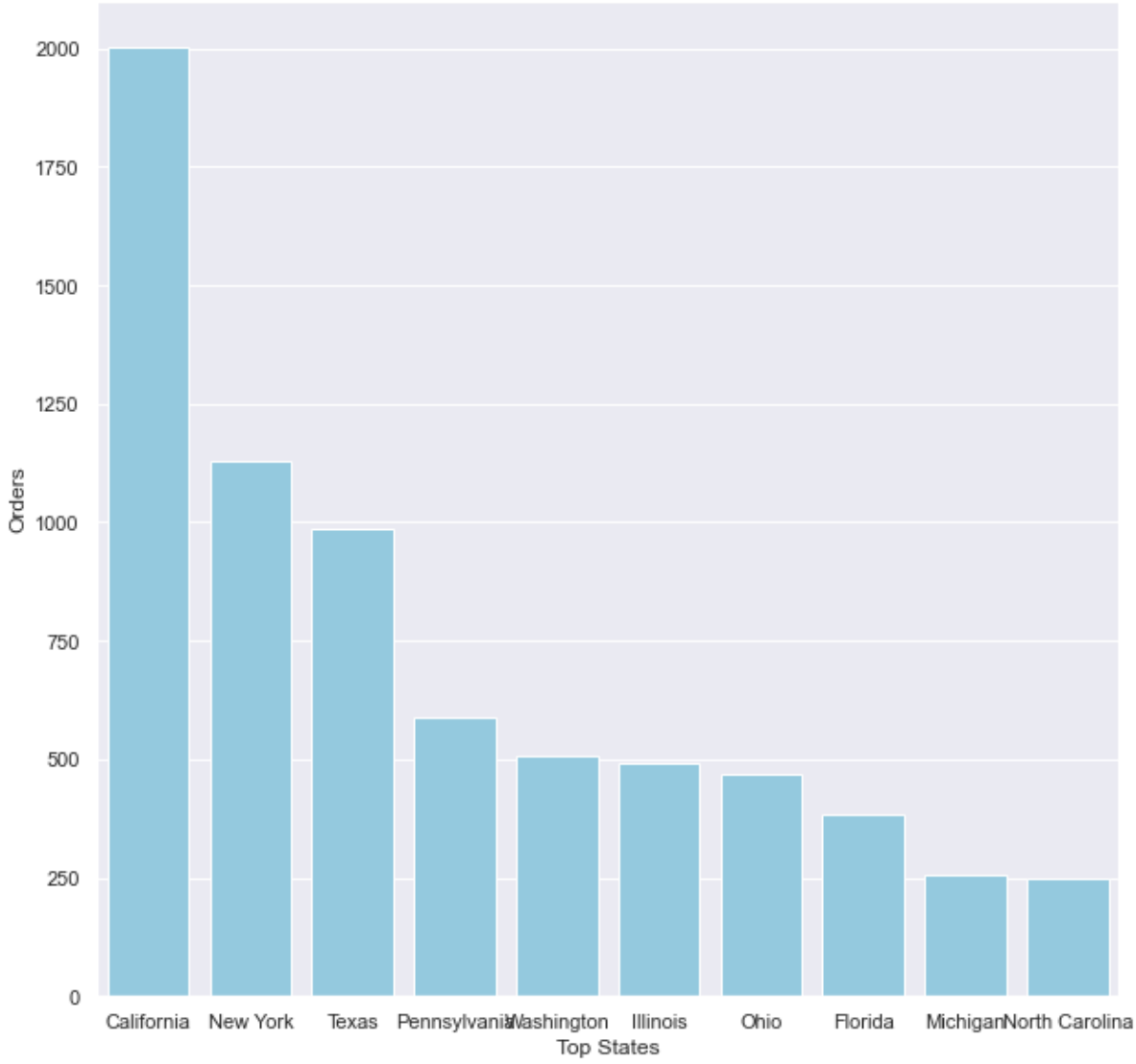


In [8]:
```
file['State'].value_counts().head(10)
```

Out[8]:
```
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 [107]:
```
sns.barplot(file['State'].value_counts().head(10).index, file['Stat
sns.set(rc={'figure.figsize':(15,15)})
plt.xlabel("Top States")
plt.ylabel("Orders")
```

Out[107]:
```
Text(0, 0.5, 'Orders')
```

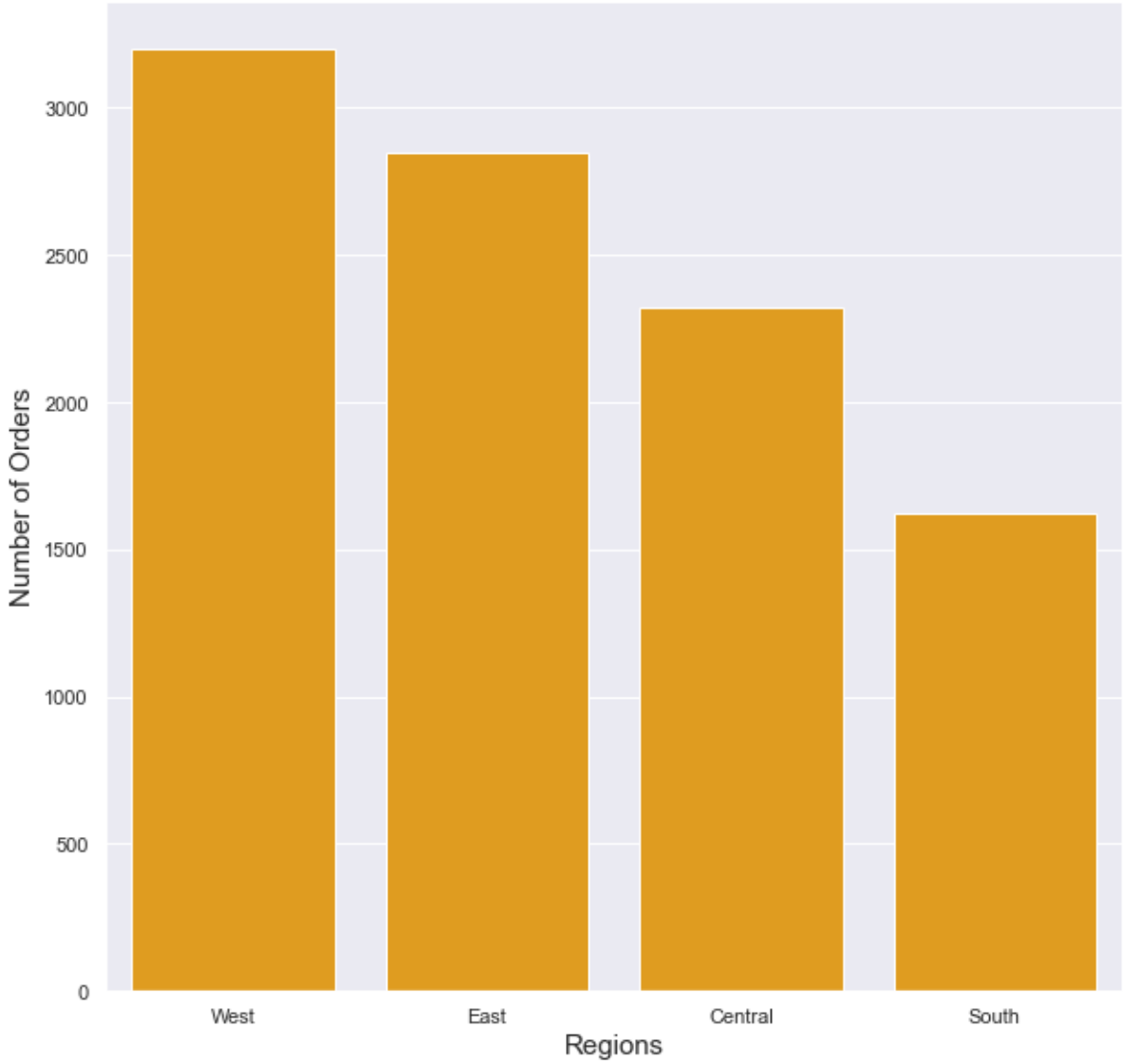


In [10]:
```
file['Region'].value_counts()
```

Out[10]:
```
West       3203
East       2848
Central    2323
South      1620
Name: Region, dtype: int64
```

In [109]:
```
sns.barplot(file['Region'].value_counts().index, file['Region'].val
sns.set(rc={'figure.figsize':(10,10)})
plt.xlabel("Regions", fontsize = 15)
plt.ylabel("Number of Orders", fontsize = 15)
```

Out[109]:
```
Text(0, 0.5, 'Number of Orders')
```

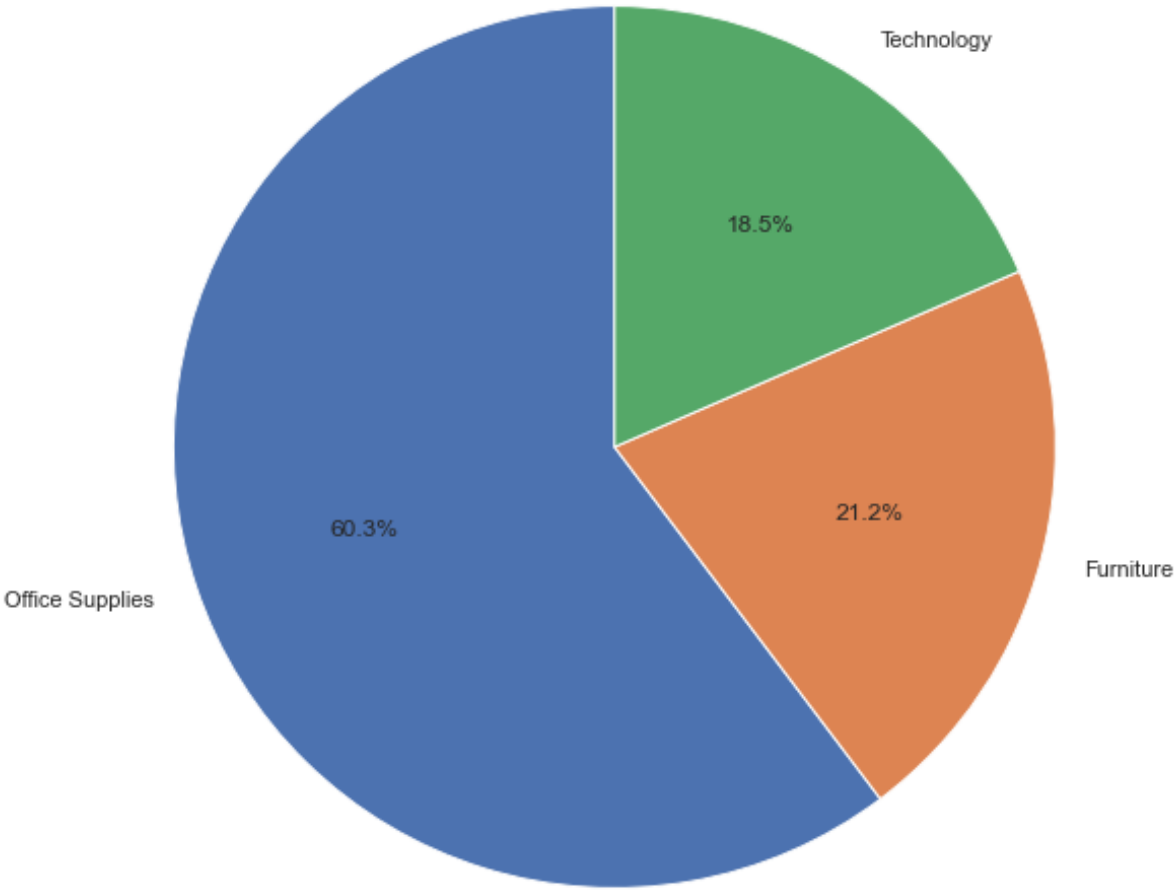


In [12]:
```
file['Category'].value_counts()
```

Out[12]:
```
Office Supplies    6026
Furniture          2121
Technology         1847
Name: Category, dtype: int64
```

In [13]:
```
fig,axis = plt.subplots()
x = list(file['Category'].value_counts().index)
y = list(file['Category'].value_counts().values)
axis.pie(y,labels = x, autopct = '%1.1f%%', startangle = 90 )
```

Out[13]:
```
([<matplotlib.patches.Wedge at 0x7fdb682d13a0>,
  <matplotlib.patches.Wedge at 0x7fdb682d1a60>,
  <matplotlib.patches.Wedge at 0x7fdb682d1fd0>],
 [Text(-1.0429540846910543, -0.34963806604293685, 'Office Supplies
'),
  Text(1.0638343615330323, -0.27974354545119656, 'Furniture'),
  Text(0.6033787837733775, 0.9197467277963863, 'Technology')],
 [Text(-0.5688840461951205, -0.19071167238705644, '60.3%'),
  Text(0.5802732881089266, -0.15258738842792538, '21.2%'),
  Text(0.32911570024002407, 0.5016800333434834, '18.5%')])
```

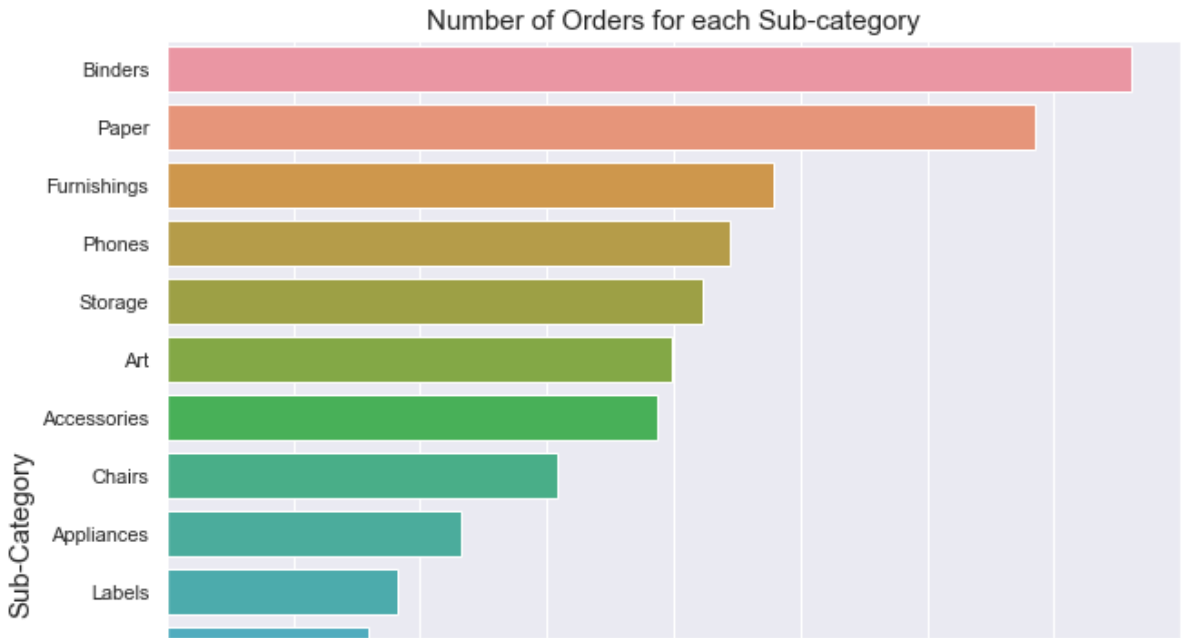


In [14]:
```
file['Sub-Category'].value_counts()
```

Out[14]:
```
Binders        1523
Paper          1370
Furnishings     957
Phones          889
Storage         846
Art             796
Accessories     775
Chairs          617
Appliances      466
Labels          364
Tables          319
Envelopes       254
Bookcases       228
Fasteners       217
Supplies        190
Machines        115
Copiers          68
Name: Sub-Category, dtype: int64
```

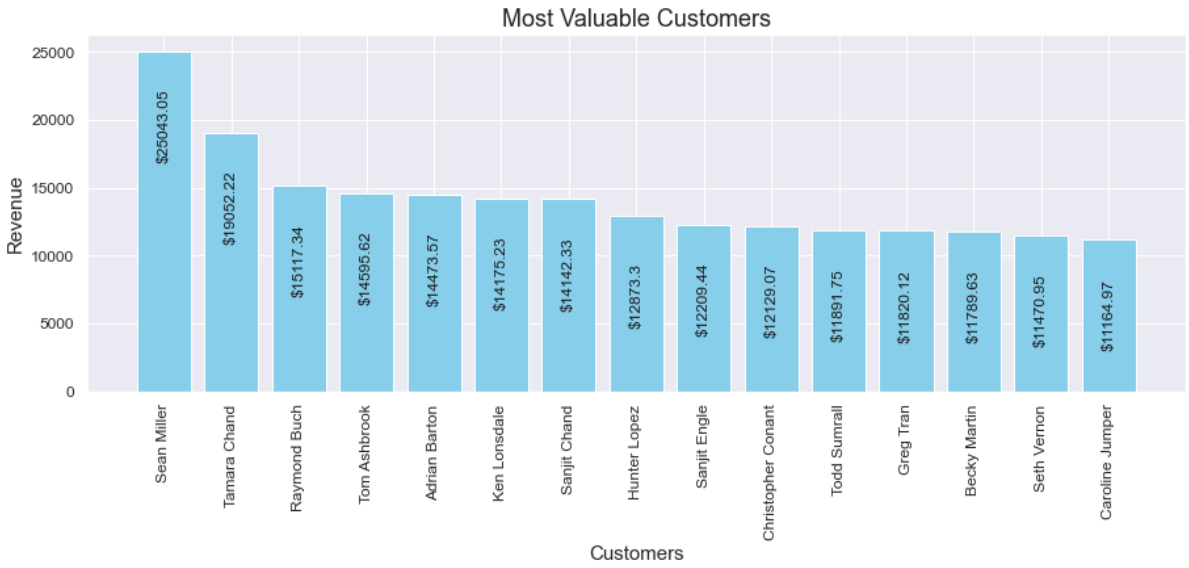
In [15]:
```
sns.set(rc={'figure.figsize':(10,10)})
sns.countplot(y="Sub-Category", data=file,order=file["Sub-Category"
plt.title("Number of Orders for each Sub-category", fontsize = 15)
plt.xlabel("Number of Orders", fontsize = 15)
plt.ylabel("Sub-Category", fontsize = 15)
```

Out[15]:
```
Text(0, 0.5, 'Sub-Category')
```



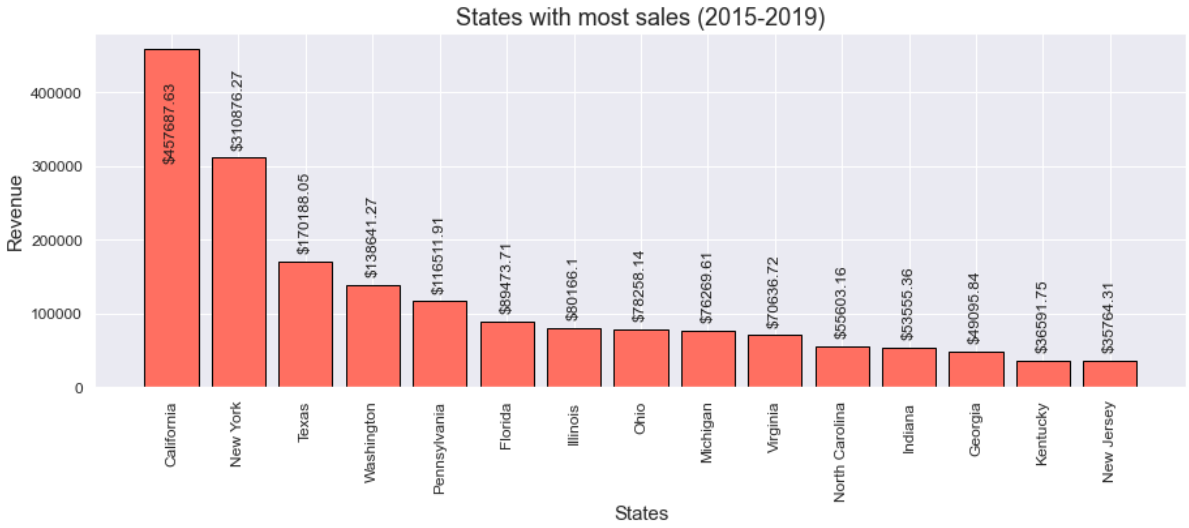
In [16]:
```
sales_sum = pd.pivot_table(file,index=["Customer Name"],values=["Sa
Top_customers = sales_sum.sort_values(by= [('sum', 'Sales')], ascen
Top_customers = Top_customers[[('sum', 'Sales')]].round(2)
```

In [143]:
```
plt.figure(figsize = (15,5))
plt.title("Most Valuable Customers", fontsize=18)
plt.bar(Top_customers.index,Top_customers[('sum', 'Sales')].values,
plt.xlabel("Customers",fontsize=15)
plt.ylabel("Revenue",fontsize=15)
plt.xticks(fontsize=12, rotation=90)
plt.yticks(fontsize=12)
for k,v in Top_customers[('sum', 'Sales')].items():
    plt.text(k,v-8000,'$'+ str(v), fontsize=12,rotation=90,color='k
```

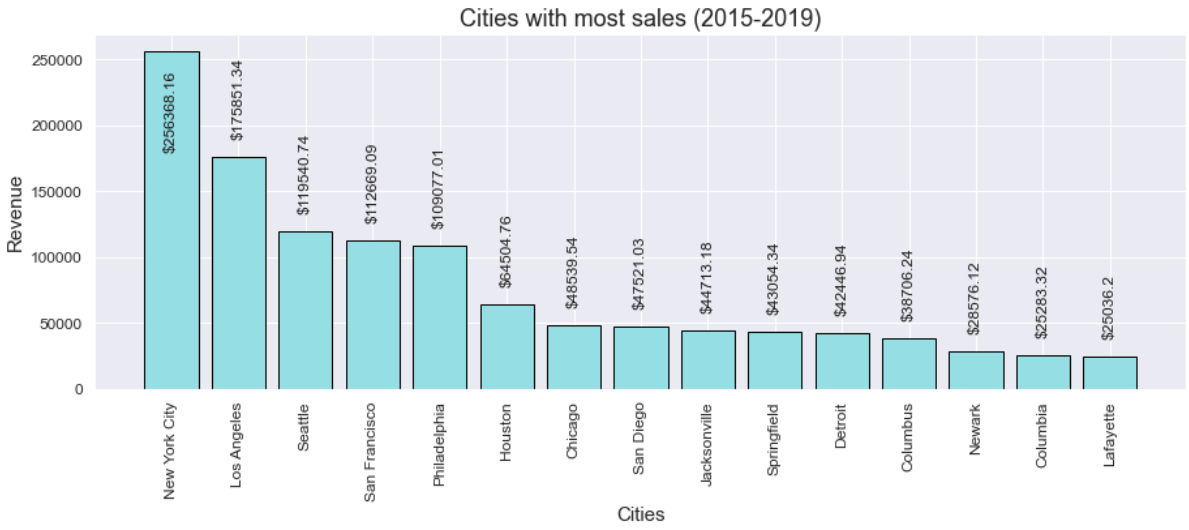


In [18]:
```
sales_sum2 = pd.pivot_table(file,index=["State"],values=["Sales"],a
Top_state = sales_sum2.sort_values(by= [('sum', 'Sales')], ascendin
Top_state = Top_state[[('sum', 'Sales')]].round(2)
```

In [19]:
```
plt.figure(figsize = (15,5))
plt.title("States with most sales (2015-2019)", fontsize=18)
plt.bar(Top_state.index,Top_state[('sum', 'Sales')].values,color= '
plt.xlabel("States",fontsize=15)
plt.ylabel("Revenue",fontsize=15)
plt.xticks(fontsize=12, rotation=90)
plt.yticks(fontsize=12)
for k,v in Top_state[('sum', 'Sales')].items():
    if v>400000:
        plt.text(k,v-150000,'$'+ str(v), fontsize=12,rotation=90,co
    else:
        plt.text(k,v+15000,'$'+ str(v), fontsize=12,rotation=90,col
```



In [20]:
```
sales_sum3 = pd.pivot_table(file,index=["City"],values=["Sales"],ag
Top_city = sales_sum3.sort_values(by= [('sum', 'Sales')], ascending
Top_city = Top_city[[('sum', 'Sales')]].round(2)
plt.figure(figsize = (15,5))
plt.title("Cities with most sales (2015-2019)", fontsize=18)
plt.bar(Top_city.index,Top_city[('sum', 'Sales')].values,color= '#9
plt.xlabel("Cities",fontsize=15)
plt.ylabel("Revenue",fontsize=15)
plt.xticks(fontsize=12, rotation=90)
plt.yticks(fontsize=12)
for k,v in Top_city[('sum', 'Sales')].items(): #To show the exact r
    if v>250000:
        plt.text(k,v-75000,'$'+ str(v), fontsize=12,rotation=90,col
    else:
        plt.text(k,v+15000,'$'+ str(v), fontsize=12,rotation=90,col
```



In [21]:
```
def conv_dates_series(df, col, old_date_format, new_date_format):

    df[col] = pd.to_datetime(df[col], format=old_date_format).dt.st

    return(df)
```

In [22]:
```
data2 = file.copy()
data2.drop('Row ID',axis = 1, inplace = True)

old_date_format='%d-%m-%Y'
new_date_format='%Y-%m-%d'

data2 = conv_dates_series(data2, "Order Date", old_date_format, new
```

In [23]:
```
df = data2.sort_values(by = "Order Date")
df.head(5)
```

Out[23]:

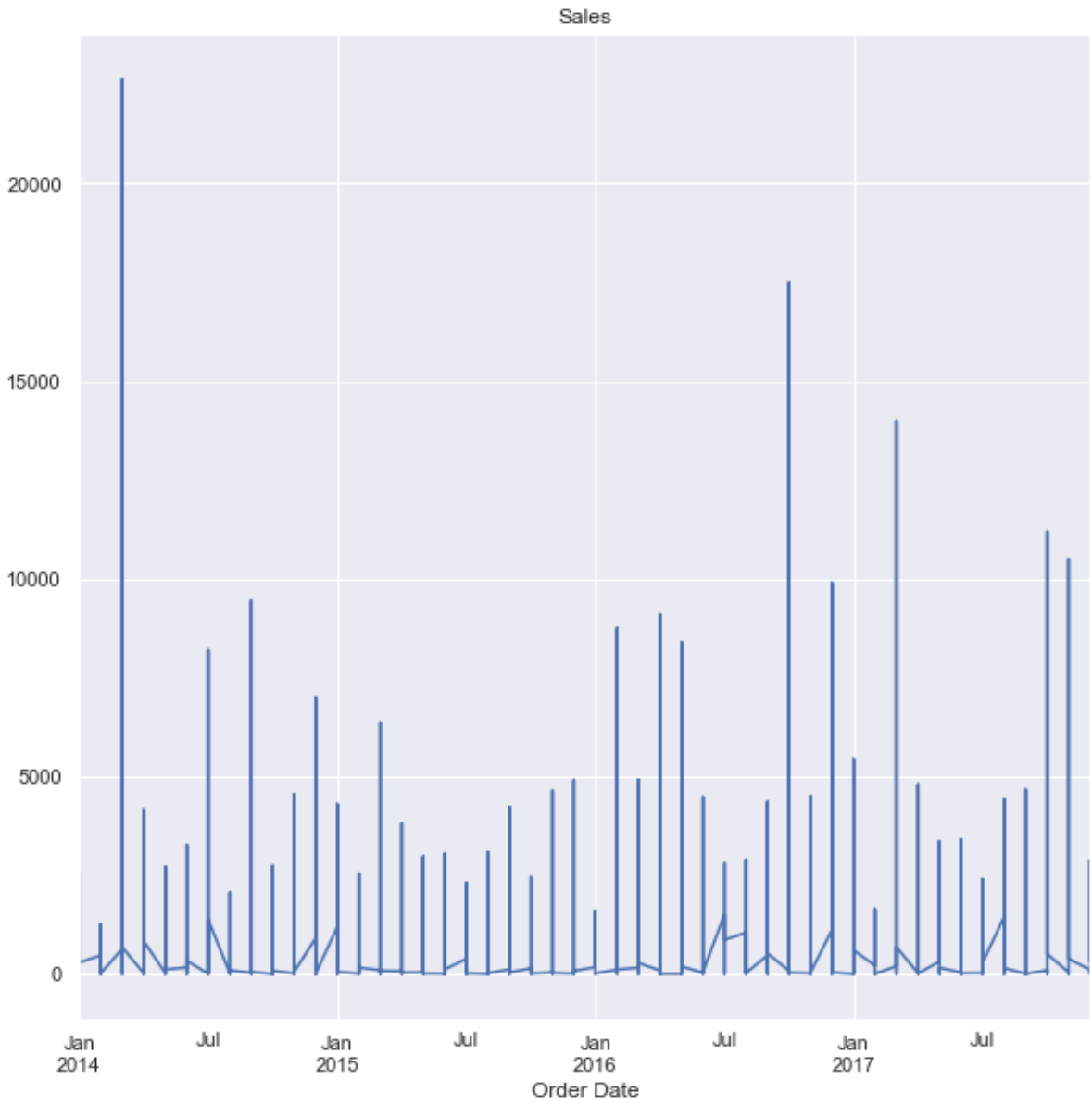
	Order ID	Order Date	Ship Date	Ship Mode	Customer ID	Customer Name	Segment	Country	City
7980	CA-2014-103800	2014-01-03	07-01-2014	Standard Class	DP-13000	Darren Powers	Consumer	United States	Houstor
739	CA-2014-112326	2014-01-04	08-01-2014	Standard Class	PO-19195	Phillina Ober	Home Office	United States	Naperville
740	CA-2014-112326	2014-01-04	08-01-2014	Standard Class	PO-19195	Phillina Ober	Home Office	United States	Naperville
741	CA-2014-112326	2014-01-04	08-01-2014	Standard Class	PO-19195	Phillina Ober	Home Office	United States	Naperville
1759	CA-2014-141817	2014-01-05	12-01-2014	Standard Class	MB-18085	Mick Brown	Consumer	United States	Philadelphia

In [24]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 9994 entries, 7980 to 906
Data columns (total 20 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Order ID       9994 non-null   object 
 1   Order Date     9994 non-null   object 
 2   Ship Date      9994 non-null   object 
 3   Ship Mode      9994 non-null   object 
 4   Customer ID    9994 non-null   object 
 5   Customer Name  9994 non-null   object 
 6   Segment        9994 non-null   object 
 7   Country        9994 non-null   object 
 8   City           9994 non-null   object 
 9   State          9994 non-null   object 
 10  Postal Code    9994 non-null   int64  
 11  Region         9994 non-null   object 
 12  Product ID     9994 non-null   object 
 13  Category       9994 non-null   object 
 14  Sub-Category   9994 non-null   object 
 15  Product Name   9994 non-null   object 
 16  Sales          9994 non-null   float64
 17  Quantity       9994 non-null   int64  
 18  Discount       9994 non-null   float64
 19  Profit         9994 non-null   float64
dtypes: float64(3), int64(2), object(15)
memory usage: 1.6+ MB
```

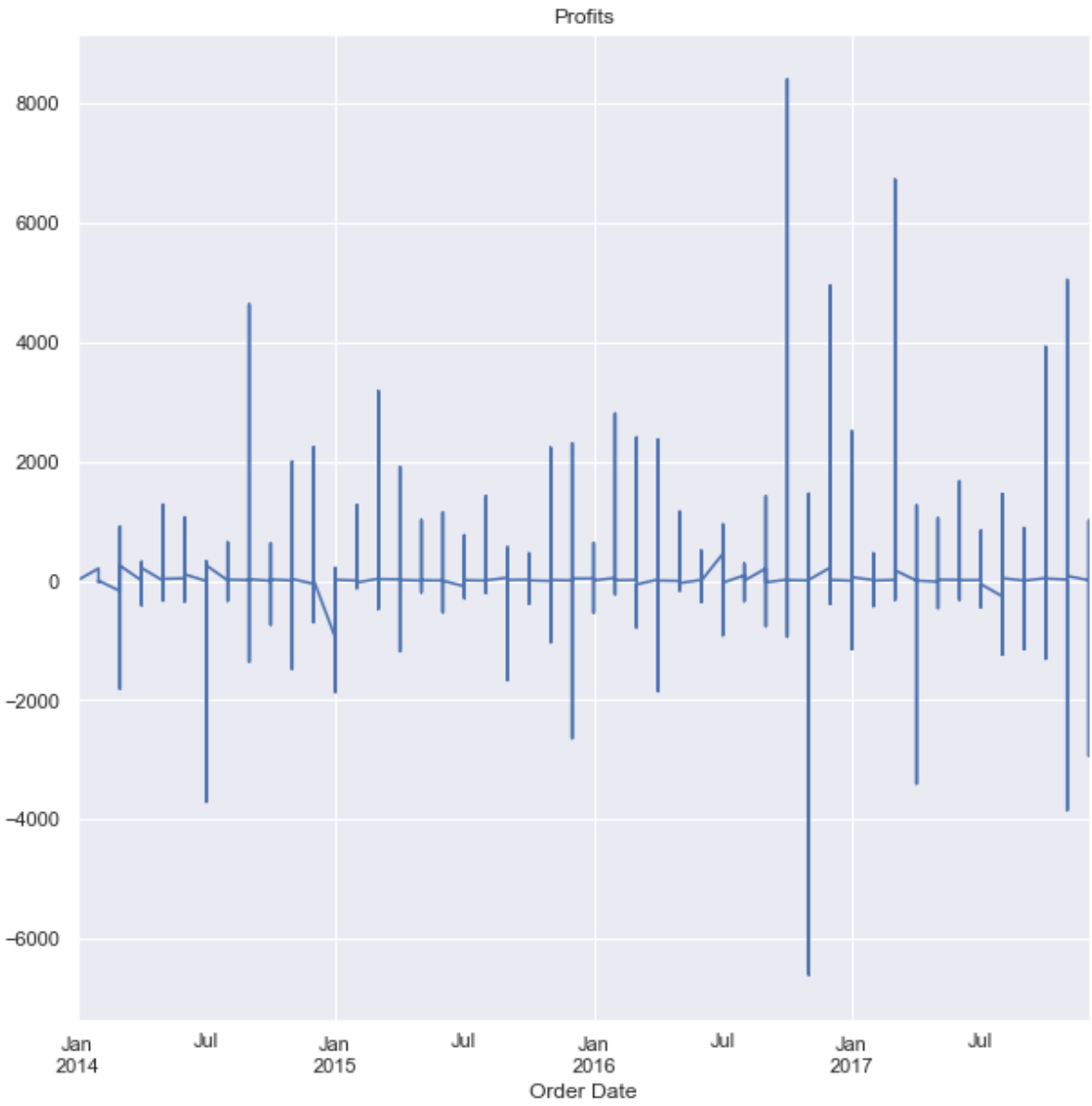
In [129]:
```
df['Order Date'] = pd.to_datetime(df['Order Date'])

df.index = pd.PeriodIndex(df["Order Date"], freq='M')

df["Sales"].plot()
plt.title('Sales')
```

Out[129]:
```
Text(0.5, 1.0, 'Sales')
```



In [130]:
```
df['Order Date'] = pd.to_datetime(df['Order Date'])

df.index = pd.PeriodIndex(df["Order Date"], freq='M')

df["Profit"].plot()
plt.title('Profits')
```

Out[130]: Text(0.5, 1.0, 'Profits')



In [26]:
```
df.head()
```

Out[26]:

Order Date	Order ID	Order Date	Ship Date	Ship Mode	Customer ID	Customer Name	Segment	Country	Ci
2014-01	CA-2014-103800	2014-01-03	07-01-2014	Standard Class	DP-13000	Darren Powers	Consumer	United States	Houstc
2014-01	CA-2014-112326	2014-01-04	08-01-2014	Standard Class	PO-19195	Phillina Ober	Home Office	United States	Napervil
2014-01	CA-2014-112326	2014-01-04	08-01-2014	Standard Class	PO-19195	Phillina Ober	Home Office	United States	Napervil
2014-01	CA-2014-112326	2014-01-04	08-01-2014	Standard Class	PO-19195	Phillina Ober	Home Office	United States	Napervil
2014-01	CA-2014-141817	2014-01-05	12-01-2014	Standard Class	MB-18085	Mick Brown	Consumer	United States	Philadelph

In [27]:
```
df['month'] = pd.DatetimeIndex(df['Order Date']).month

df['Year'] = pd.DatetimeIndex(df['Order Date']).year

df.head()
```

Out[27]:

Order Date	Order ID	Order Date	Ship Date	Ship Mode	Customer ID	Customer Name	Segment	Country	Ci
2014-01	CA-2014-103800	2014-01-03	07-01-2014	Standard Class	DP-13000	Darren Powers	Consumer	United States	Houstc
2014-01	CA-2014-112326	2014-01-04	08-01-2014	Standard Class	PO-19195	Phillina Ober	Home Office	United States	Napervil
2014-01	CA-2014-112326	2014-01-04	08-01-2014	Standard Class	PO-19195	Phillina Ober	Home Office	United States	Napervil
2014-01	CA-2014-112326	2014-01-04	08-01-2014	Standard Class	PO-19195	Phillina Ober	Home Office	United States	Napervil
2014-01	CA-2014-141817	2014-01-05	12-01-2014	Standard Class	MB-18085	Mick Brown	Consumer	United States	Philadelph

5 rows × 22 columns

In [28]:
```
dataPivot = pd.pivot_table(df, values = "Sales",aggfunc='mean', col
dataPivot
```

Out[28]:

Year	2014	2015	2016	2017
month				
1	180.213861	313.346131	208.342596	283.686284
2	98.258522	186.740797	276.853193	189.730219
3	354.719803	280.625014	317.275307	247.362827
4	209.595148	213.720053	227.941406	179.909045
5	193.838418	206.381414	253.278791	182.897150
6	256.260204	179.690522	202.736352	216.251942
7	237.387364	205.466607	195.333149	200.285027
8	182.414827	232.064982	176.791899	289.545358
9	305.139369	220.463884	202.231474	191.430614
10	197.820082	189.186286	304.529311	260.996387
11	247.260115	234.483221	214.626935	258.056264
12	250.164103	237.087092	275.565463	181.448742

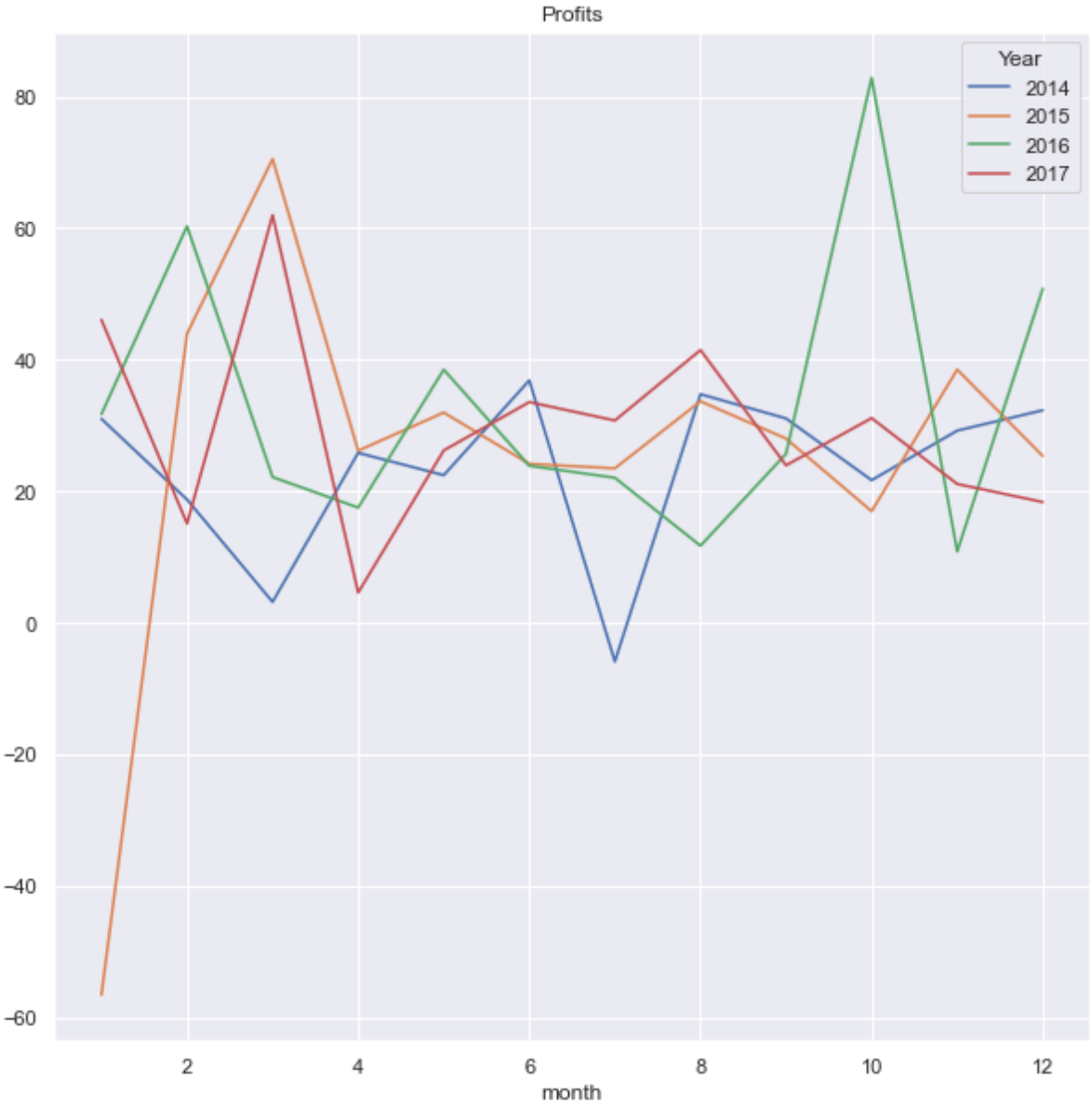
In [124]:
```
dataPivot1 = pd.pivot_table(df, values = "Profit",aggfunc='mean', c
dataPivot1
```

Out[124]:

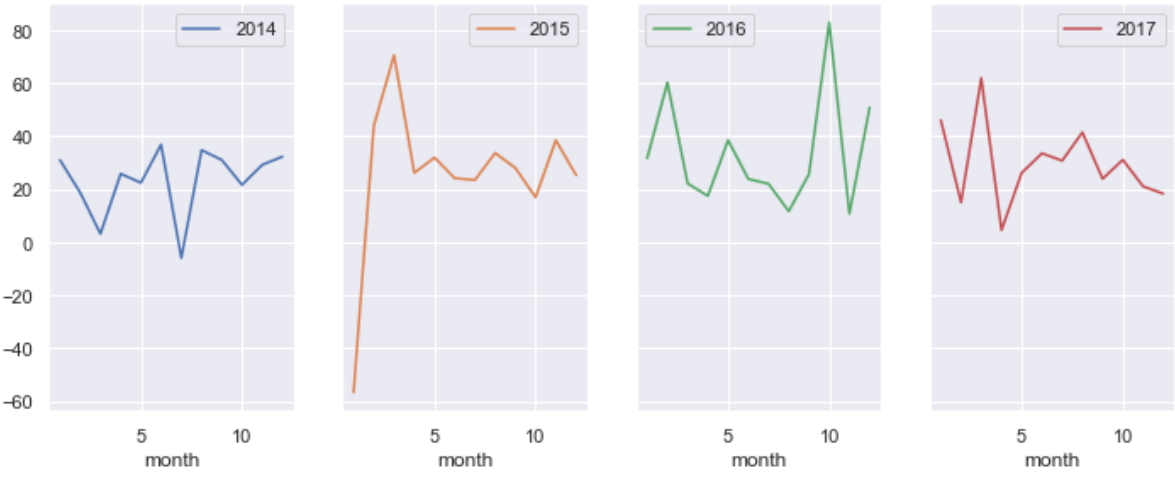
Year	2014	2015	2016	2017
month				
1	31.015072	-56.569086	31.739588	46.067349
2	18.745835	43.966419	60.296139	15.082916
3	3.176624	70.522448	22.159313	61.982737
4	25.843224	26.171851	17.516558	4.597488
5	22.448439	31.971705	38.498428	26.209020
6	36.863144	24.170704	23.871247	33.564636
7	-5.884494	23.490345	22.054119	30.763811
8	34.758856	33.684330	11.716303	41.472274
9	31.074998	28.017620	25.698781	23.946744
10	21.687153	16.972084	82.873176	31.125086
11	29.220525	38.502433	10.841642	21.111337
12	32.315000	25.370145	50.810538	18.362223

In [119]:
```
dataPivot1.plot()
plt.title("Profits")
```

Out[119]:
```
Text(0.5, 1.0, 'Profits')
```

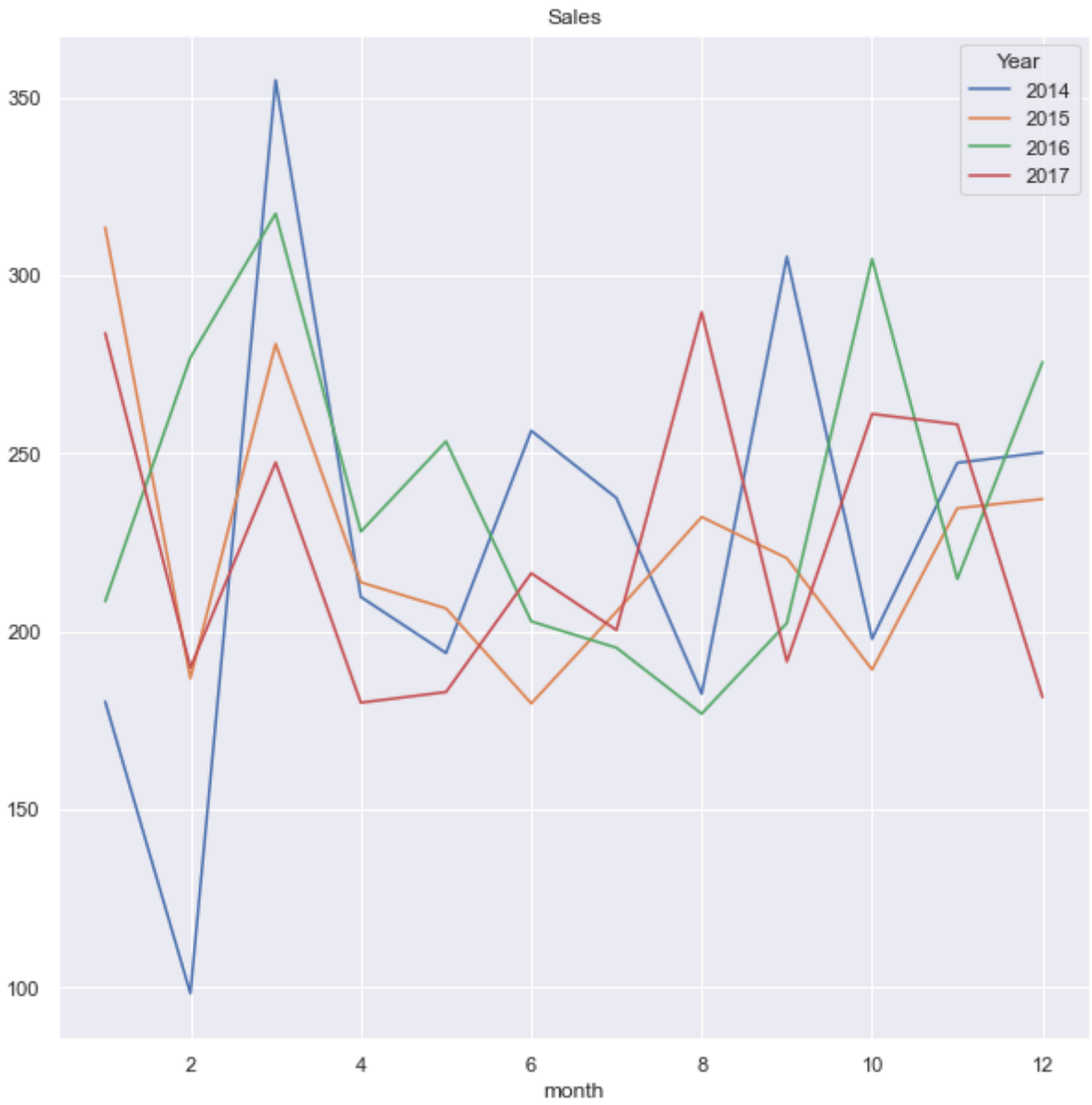


In [122]:
```
dataPivot1.plot(subplots = True, figsize=(15, 15), layout=(3, 5), s
plt.show()
```



In [123]:
```
dataPivot.plot()
plt.title('Sales')
```

Out[123]:
```
Text(0.5, 1.0, 'Sales')
```



In [125]:
```
dataPivot.plot(subplots = True, figsize=(15, 15), layout=(3, 5), sh
plt.show()
```

In [34]:
```
from sklearn.metrics import mean_squared_error
file.head()
```

Out[34]:

	Row ID	Order ID	Order Date	Ship Date	Ship Mode	Customer ID	Customer Name	Segment	Country	C
0	1	CA-2016-152156	08-11-2016	11-11-2016	Second Class	CG-12520	Claire Gute	Consumer	United States	Henders
1	2	CA-2016-152156	08-11-2016	11-11-2016	Second Class	CG-12520	Claire Gute	Consumer	United States	Henders
2	3	CA-2016-138688	12-06-2016	16-06-2016	Second Class	DV-13045	Darrin Van Huff	Corporate	United States	L Ange
3	4	US-2015-108966	11-10-2015	18-10-2015	Standard Class	SO-20335	Sean O'Donnell	Consumer	United States	F Lauderd
4	5	US-2015-108966	11-10-2015	18-10-2015	Standard Class	SO-20335	Sean O'Donnell	Consumer	United States	F Lauderd

5 rows × 21 columns

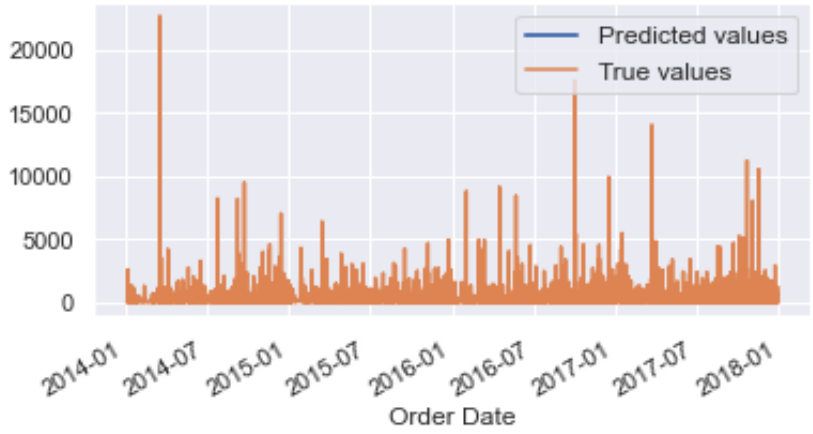
In [38]:
```
df_1 = file.copy()
df_1['Order Date'] = pd.to_datetime(df_1['Order Date'], format='%d-
df_1.sort_values(by=['Order Date'], inplace=True, ascending=True)
df_1.set_index("Order Date", inplace = True)
df_1['time'] = df_1.index.to_series().apply(lambda x: x.toordinal()

# Splitting the data into 70:30
X_train, X_test = df_1[['time']][:int(df_1.shape[0]*0.7)], df_1[['t
y_train, y_test = df_1['Sales'][:int(df_1.shape[0]*0.7)], df_1['Sal
X_train.shape, y_train.shape, X_test.shape, y_test.shape
```

Out[38]: ((6995, 1), (6995,), (2999, 1), (2999,))

In [39]:
```
from sklearn.linear_model import LinearRegression
lr = LinearRegression().fit(X_train, y_train)
lr_pred_train = lr.predict(X_train)

print('RMSE on train: ', mean_squared_error(y_train, lr_pred_train,
lr_pred = lr.predict(X_test)

print('RMSE on test: ', mean_squared_error(y_test, lr_pred, squared
```

```
RMSE on train:  637.3606526762836
RMSE on test:  588.8268668726282
```

In [41]:
```
def plot_predict(ts, x_test, y_pred):
    plt.figure(figsize=(6, 3))
    plt.plot(x_test.index, y_pred, label = 'Predicted values', line
    ts.plot(label = 'True values')
    plt.legend(loc = 'best', fontsize = 12)
    plt.xlabel('Order Date', fontsize = 12)
    plt.show()
plot_predict(df_1['Sales'], X_test, lr_pred)
```



In [42]:
```
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import TimeSeriesSplit

# check perfromance score
scores_lr = cross_val_score(LinearRegression(),
                            X_train, y_train,
                            cv=TimeSeriesSplit(n_splits=10),
                            scoring="neg_mean_squared_error")
np.mean(scores_lr)
```

Out[42]: -351125.3827629725

In [47]:
```
file['Order Date'] = pd.to_datetime(file['Order Date'], format='%d-
file['Ship Date'] = pd.to_datetime(file['Ship Date'], format='%d-%m
file.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9994 entries, 0 to 9993
Data columns (total 21 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Row ID         9994 non-null   int64
 1   Order ID       9994 non-null   object
 2   Order Date     9994 non-null   datetime64[ns]
 3   Ship Date      9994 non-null   datetime64[ns]
 4   Ship Mode      9994 non-null   object
 5   Customer ID    9994 non-null   object
 6   Customer Name  9994 non-null   object
 7   Segment        9994 non-null   object
 8   Country        9994 non-null   object
 9   City           9994 non-null   object
 10  State          9994 non-null   object
 11  Postal Code    9994 non-null   int64
 12  Region         9994 non-null   object
 13  Product ID     9994 non-null   object
 14  Category       9994 non-null   object
 15  Sub-Category   9994 non-null   object
 16  Product Name   9994 non-null   object
 17  Sales          9994 non-null   float64
 18  Quantity       9994 non-null   int64
 19  Discount       9994 non-null   float64
 20  Profit         9994 non-null   float64
dtypes: datetime64[ns](2), float64(3), int64(3), object(13)
memory usage: 1.6+ MB
```

In [56]:
```
file.drop('Row ID',axis = 1, inplace = True)
#sorting data by order date
file.sort_values(by=['Order Date'], inplace=True, ascending=True)
data_new = pd.DataFrame(file['Sales'])
print(data_new.head())
data_new.tail()
```

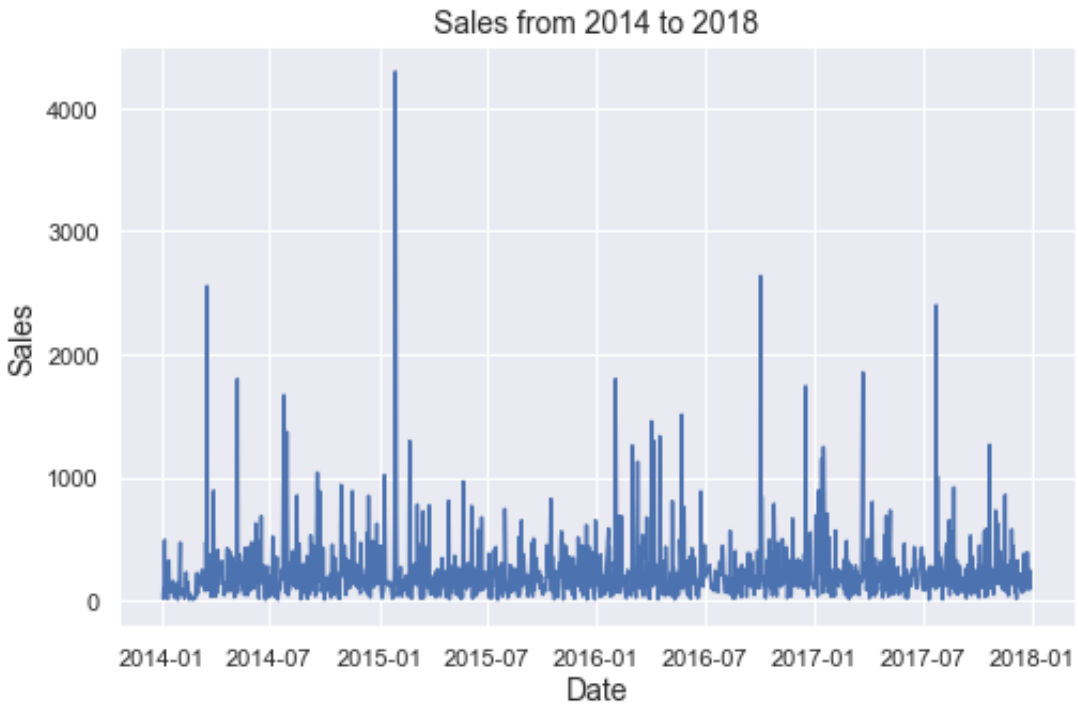
```
              Sales
Order Date
2014-01-03   16.448
2014-01-04   11.784
2014-01-04  272.736
2014-01-04    3.540
2014-01-05   19.536
```

Out[56]:

Order Date	Sales
2017-12-30	13.904
2017-12-30	20.720
2017-12-30	209.300
2017-12-30	3.024
2017-12-30	323.136

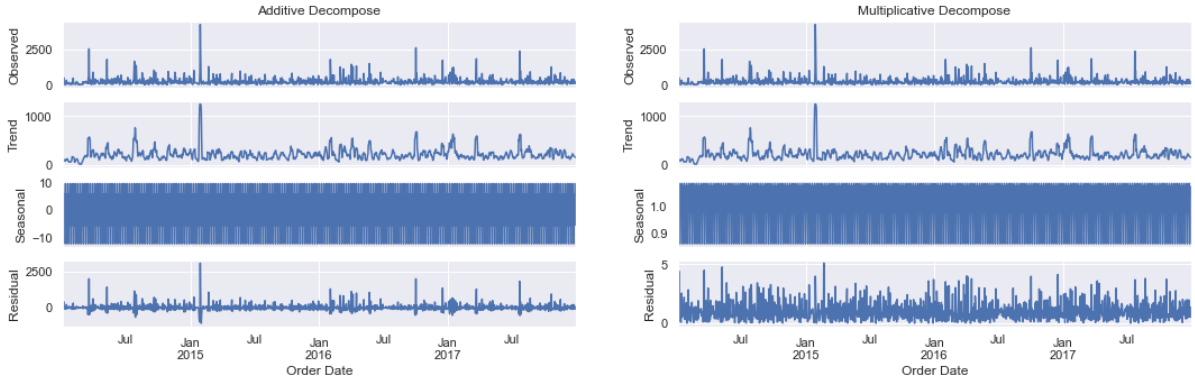
In [57]:
```
data_new =  pd.DataFrame(data_new['Sales'].resample('D').mean())
data_new = data_new.interpolate(method='linear')
#It ignore the index and treats the values as equally spaced.
```

In [59]:
```
# Visualize the time series
plt.figure(figsize=(8, 5))

plt.plot(data_new.index, data_new.Sales)
plt.xlabel('Date', fontsize = 14)
plt.ylabel('Sales', fontsize = 14)
plt.title('Sales from 2014 to 2018', fontsize = 14)
plt.show()
```



In [62]:
```
from statsmodels.tsa.seasonal import seasonal_decompose

# Additive Decomposition
result_add = seasonal_decompose(data_new['Sales'], model='additive'

# Multiplicative Decomposition
result_mul = seasonal_decompose(data_new['Sales'], model='multiplic
```

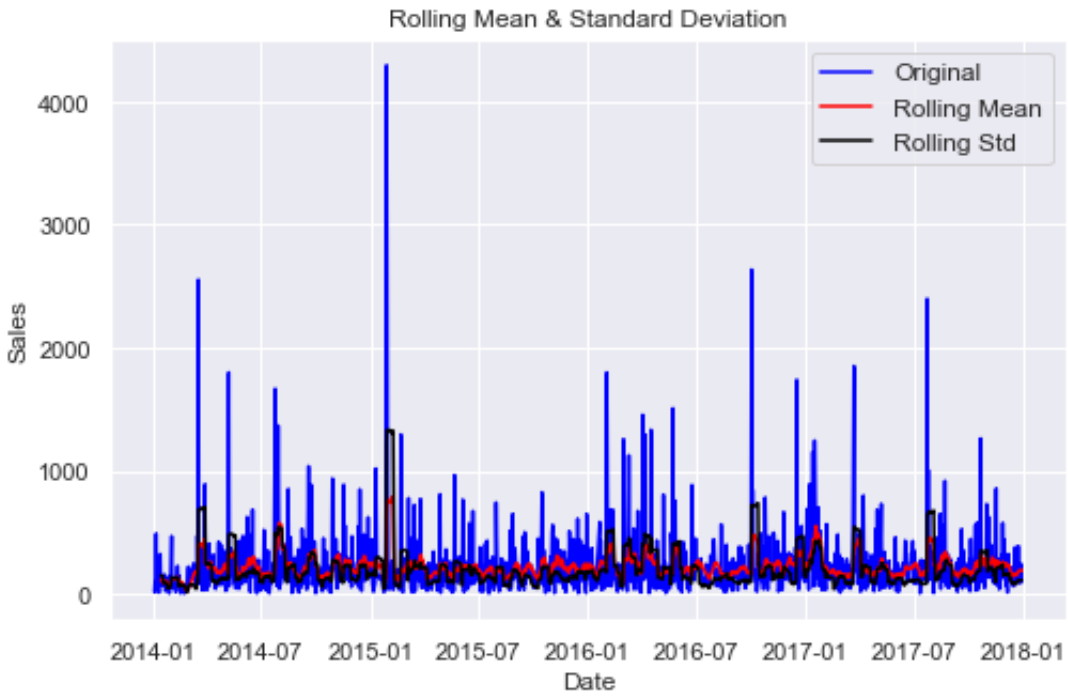
In [63]:
```
def plotseasonal(res, axes, suptitle):
    res.observed.plot(ax=axes[0])
    axes[0].set_ylabel('Observed', fontsize=12)
    axes[0].set_title(suptitle, fontsize=12)
    res.trend.plot(ax=axes[1])
    axes[1].set_ylabel('Trend', fontsize=12)
    res.seasonal.plot(ax=axes[2])
    axes[2].set_ylabel('Seasonal', fontsize=12)
    res.resid.plot(ax=axes[3])
    axes[3].set_ylabel('Residual', fontsize=12)


fig, axes = plt.subplots(ncols=2, nrows=4, sharex=True, figsize=(18

plotseasonal(result_add, axes[:,0], 'Additive Decompose')
plotseasonal(result_mul, axes[:,1], 'Multiplicative Decompose')
```



In [64]:

```
from statsmodels.tsa.stattools import adfuller, kpss

def test_stationarity(ts, do_adf=True, do_kpss = False):

    # Determing rolling statistics
    rolmean = ts.rolling(12).mean()
    rolstd = ts.rolling(12).std()

    #rolling statistics:
    plt.figure(figsize=(8, 5))
    orig = plt.plot(ts, color='blue',label='Original')
    mean = plt.plot(rolmean, color='red', label='Rolling Mean')
    std = plt.plot(rolstd, color='black', label = 'Rolling Std')
    plt.legend(loc='best', fontsize=12)
    plt.title('Rolling Mean & Standard Deviation', fontsize=12)
    plt.xlabel('Date', fontsize = 12)
    plt.ylabel('Sales', fontsize = 12)
    plt.show(block=False)

    if do_adf:
        # Perform Augmented Dickey-Fuller test:
        print('Results of ADF Test:')
        dftest = adfuller(ts, autolag='AIC')
        dfoutput = pd.Series(dftest[0:4], index=['Test Statistic','
        for key,value in dftest[4].items():
            dfoutput['Critical Value (%s)'%key] = value
        print(dfoutput)

    print()

    if do_kpss:
        # Perform Kwiatkowski-Phillips-Schmidt-Shin test:
        print('Results of KPSS Test:')
        dftest = kpss(ts, regression='c')
        dfoutput = pd.Series(dftest[0:3], index=['Test Statistic','
        for key,value in dftest[3].items():
            dfoutput['Critical Value (%s)'%key] = value
        print(dfoutput)
```

In [134]:
```
test_stationarity(data_new.Sales, do_kpss= True)
```




```
Results of ADF Test:
Test Statistic                  -20.876027
p-value                           0.000000
#Lags Used                        2.000000
Number of Observations Used    1455.000000
Critical Value (1%)              -3.434852
Critical Value (5%)              -2.863528
Critical Value (10%)             -2.567829
dtype: float64

Results of KPSS Test:
Test Statistic            0.116337
p-value                   0.100000
Truncated Lag #          24.000000
Critical Value (10%)      0.347000
Critical Value (5%)       0.463000
Critical Value (2.5%)     0.574000
Critical Value (1%)       0.739000
dtype: float64

/Users/venkatakrishnasunkara/opt/anaconda3/lib/python3.8/site-pack
ages/statsmodels/tsa/stattools.py:1910: InterpolationWarning: The
test statistic is outside of the range of p-values available in th
e
look-up table. The actual p-value is greater than the p-value retu
rned.

  warnings.warn(
```

In [66]:
```
sales = pd.DataFrame(data_new.Sales).reset_index()

# rename columns to fit Prophet
sales.columns = ['ds', 'y']
sales.shape[0]
```

Out[66]:
```
1458
```

In [67]:
```
from fbprophet import Prophet

X_tr = sales[:int(sales.shape[0]*0.7)]
X_tst = sales[int(sales.shape[0]*0.7):]

print("train shape",X_tr.shape)
print("test shape",X_tst.shape)
```

```
train shape (1020, 2)
test shape (438, 2)
```

In [68]:
```
pd.plotting.register_matplotlib_converters()
f, ax = plt.subplots(figsize=(14,5))
X_tr.plot(kind='line', x='ds', y='y', color='blue', label='Train',
X_tst.plot(kind='line', x='ds', y='y', color='red', label='Test', a
plt.title('Sales Amount Traning and Test data')
plt.show()
```

In [69]:
```
def mean_absolute_percentage_error(y_true, y_pred):
    y_true, y_pred = np.array(y_true), np.array(y_pred)
    return np.mean(np.abs((y_true - y_pred) / y_true)) * 100
```

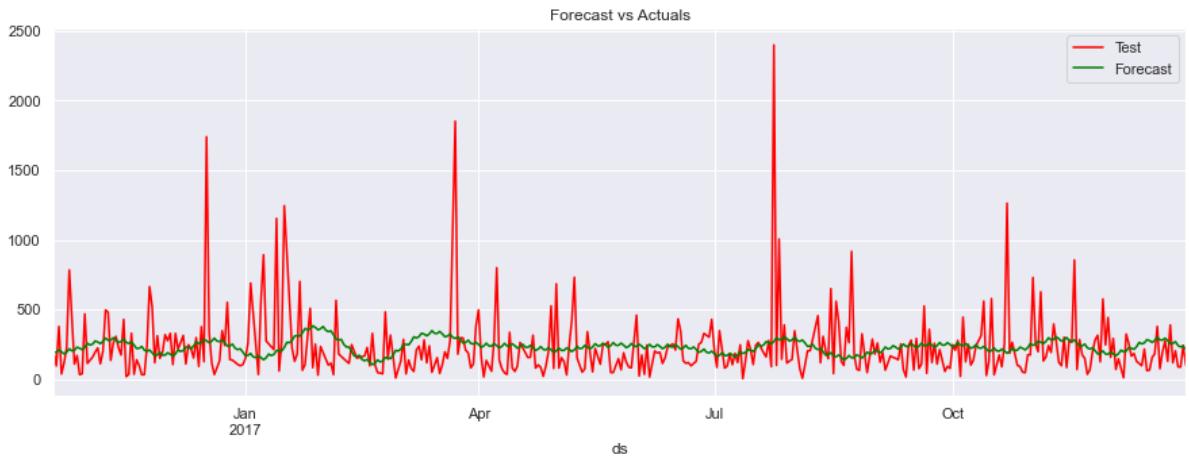
In [99]:
```
model_fb =Prophet(daily_seasonality=True )
model_fb.fit(X_tr)
X_tr_forecast = model_fb.predict(X_tr)
print('RMSE on train: ', mean_squared_error(X_tr['y'], X_tr_forecas
X_tst_forecast = model_fb.predict(X_tst)
print('RMSE on test: ', mean_squared_error(X_tst['y'], X_tst_foreca
X_tst_forecast[['ds', 'yhat', 'yhat_lower', 'yhat_upper']].tail(7)
```

```
RMSE on train:  274.1937992947828
RMSE on test:  242.5465766622908
```

Out[99]:

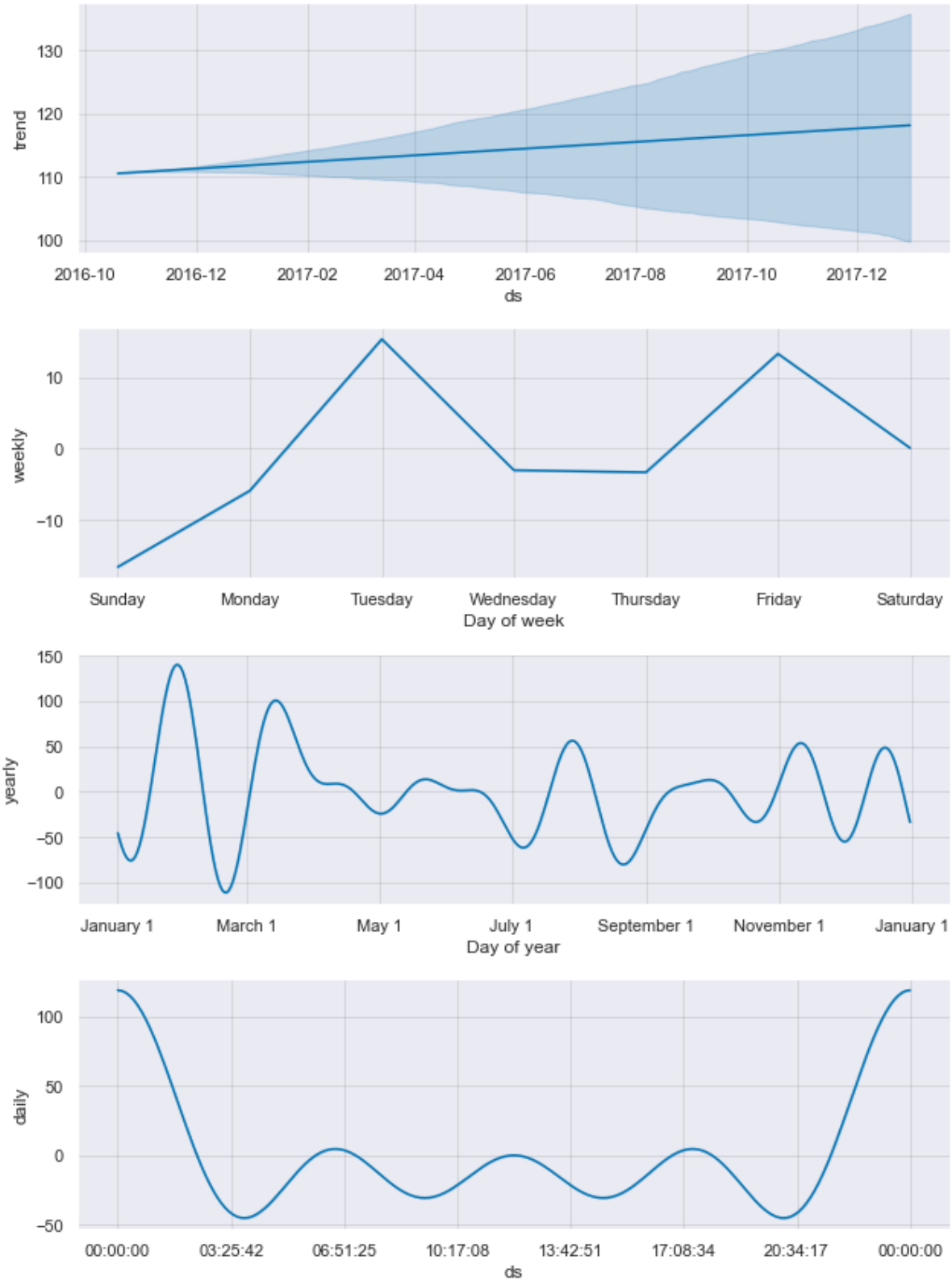
	ds	yhat	yhat_lower	yhat_upper
431	2017-12-24	253.672408	-96.253048	610.793274
432	2017-12-25	256.944111	-76.316834	608.097975
433	2017-12-26	269.657740	-82.565697	648.226513
434	2017-12-27	241.952096	-116.535499	620.022642
435	2017-12-28	231.649471	-148.345765	587.921729
436	2017-12-29	237.862969	-135.901774	593.087620
437	2017-12-30	214.233276	-123.252766	586.897031

In [135]:
```
f, ax = plt.subplots(figsize=(14,5))
f.set_figheight(5)
f.set_figwidth(15)
X_tst.plot(kind='line',x='ds', y='y', color='red', label='Test', ax
X_tst_forecast.plot(kind='line',x='ds',y='yhat', color='green',labe
plt.title('Forecast vs Actuals')
plt.show()
mape = mean_absolute_percentage_error(X_tst['y'],X_tst_forecast['yh
print("MAPE",round(mape,4))
```




```
MAPE 153.9406
```

In [102]:
```
fig=model_fb.plot_components(X_tst_forecast)
```



In [136]:
```
# parameter tuning
import holidays

holiday = pd.DataFrame([])
for date, name in sorted(holidays.UnitedStates(years=[2017,2018,201
    holiday = holiday.append(pd.DataFrame({'ds': date, 'holiday': "
holiday['ds'] = pd.to_datetime(holiday['ds'], format='%Y-%m-%d', er

holiday.head()
```

Out[136]:

	ds	holiday
0	2017-01-01	US-Holidays
1	2017-01-02	US-Holidays
2	2017-01-16	US-Holidays
3	2017-02-20	US-Holidays
4	2017-05-29	US-Holidays

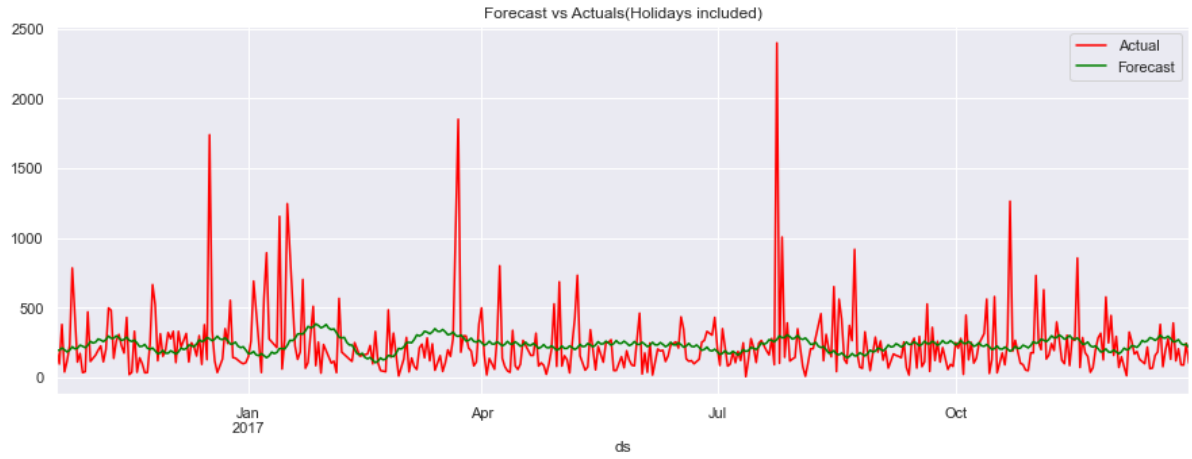
In [137]:
```
#train model with holidays
model_with_holidays = Prophet(holidays=holiday,daily_seasonality=Tr
model_with_holidays.fit(X_tr)

X_tr_forecast_holiday = model_with_holidays.predict(X_tr)
print('RMSE on train: ', mean_squared_error(X_tr['y'], X_tr_forecas
X_tst_forecast_holiday = model_with_holidays.predict(X_tst)
print('RMSE on test: ', mean_squared_error(X_tst['y'], X_tst_foreca
X_tst_forecast_holiday[['ds', 'yhat', 'yhat_lower', 'yhat_upper']].
```

```
RMSE on train:  274.1937992947828
RMSE on test:  242.5465766622908
```

Out[137]:

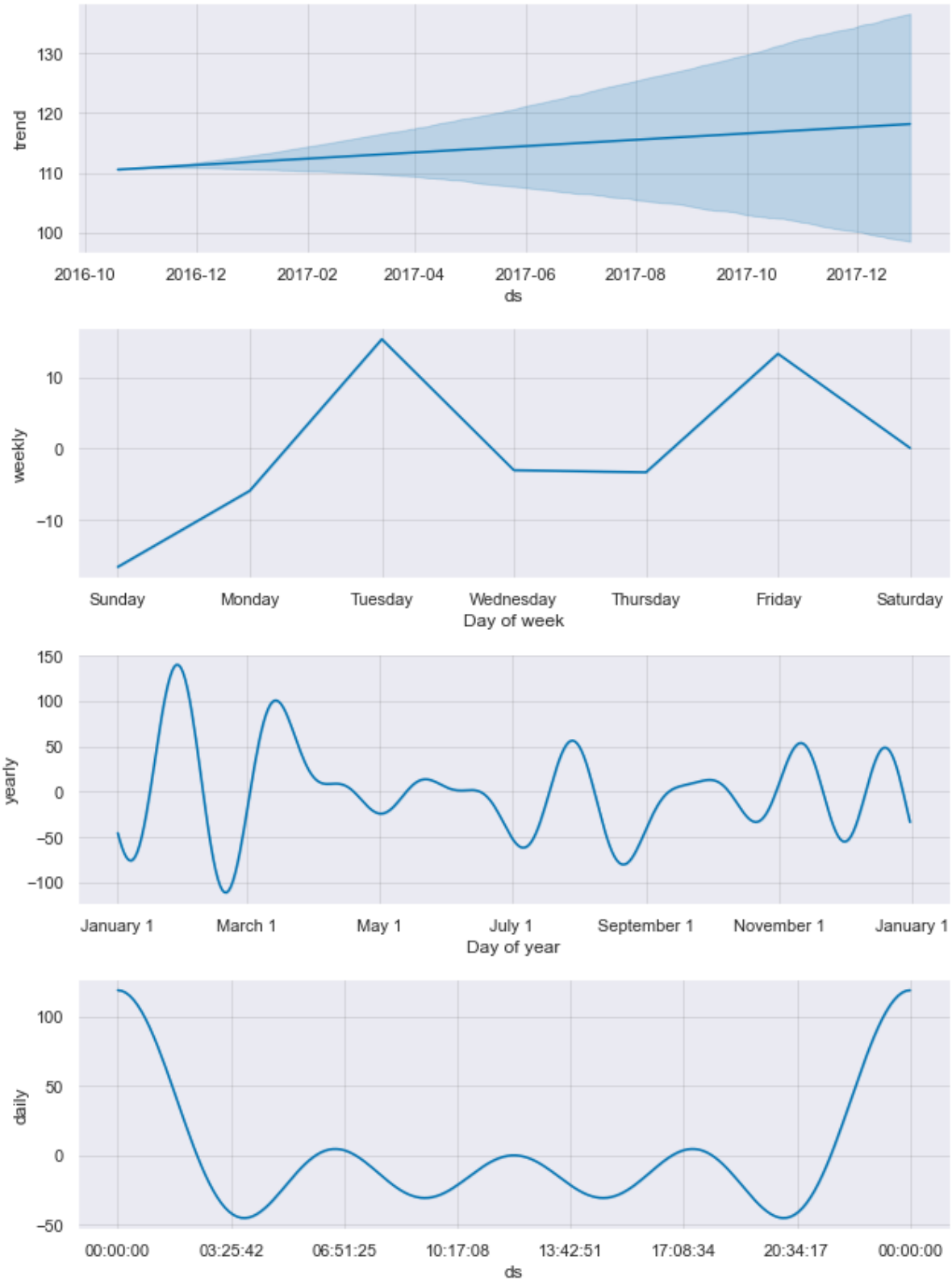
	ds	yhat	yhat_lower	yhat_upper
431	2017-12-24	253.672408	-106.533371	603.180682
432	2017-12-25	256.944111	-94.858397	620.869000
433	2017-12-26	269.657740	-82.238459	605.177007
434	2017-12-27	241.952096	-94.110133	585.838024
435	2017-12-28	231.649471	-153.591590	574.095983
436	2017-12-29	237.862969	-139.465329	556.149862
437	2017-12-30	214.233276	-152.100597	569.080807

In [142]:
```
f, ax = plt.subplots(figsize=(14,5))
f.set_figheight(5)
f.set_figwidth(15)
X_tst.plot(kind='line',x='ds', y='y', color='red', label='Actual',
X_tst_forecast_holiday.plot(kind='line',x='ds',y='yhat', color='gre
plt.title('Forecast vs Actuals(Holidays included)')
plt.show()
mape = mean_absolute_percentage_error(X_tst['y'],X_tst_forecast_hol
print("MAPE",round(mape,4))
```



MAPE 153.9406

In [141]:
```
fig=model_fb.plot_components(X_tst_forecast_holiday)
```



In [ ]: