Online Retail Time Series Analysis & Forecasting

Modeling and Forecasting

We see tons of different stores here and there through the web. Internet made it possible to trade with anyone and everywhere. We can buy goods without leaving our house, we can compare prices in different stores within seconds, we can find what we really want and do not accept just the first more or less suitable offer. And I believe it would be really interesting to look at this world through the data it produces.

This dataset which contains all the transactions occurring between 01/12/2010 and 09/12/2011 for a UK-based and registered non-store online retail.The company mainly sells unique all-occasion gifts. Most sales comes from UK and about 90% customers also are from UK, outside UK, most of the sales are from Europe. In my research I analyzed and studied this dataset and used time series analysis to build a model using ARIMA for the top products and total sales.

Libraries and Packeges

As always, we start our analysis by setting up our environment and by importing necessary libraries.

In [1]:
```
import pandas as pd
import numpy as np
import math
import matplotlib.pyplot as plt
import random
import seaborn as sns
import math
import re
import plotly.express as px
plt.style.use('fivethirtyeight')
%matplotlib inline
from sklearn import metrics
import time, warnings
import datetime as dt
import geopandas
from sklearn.metrics import average_precision_score
from scipy import stats
from matplotlib import pylab
from matplotlib import pyplot
```

```
from collections import defaultdict
from sklearn.model_selection import KFold
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import confusion_matrix,accuracy_score,roc_curve,
classification_report
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
import sklearn.cluster as cluster
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_squared_error
from math import sqrt
from sklearn.metrics import r2_score
from sklearn.cluster import KMeans
from pandas.plotting import scatter_matrix
from sklearn.decomposition import PCA
from sklearn import mixture
from sklearn.metrics import f1_score
from sklearn.feature_selection import SelectKBest, f_regression, mutua
l_info_regression
import statsmodels.api as sm
from statsmodels.formula.api import ols
from statsmodels.tsa.stattools import adfuller
from statsmodels.graphics.tsaplots import plot_pacf,plot_acf
from statsmodels.tsa.arima_model import ARMA
from statsmodels.tsa.arima_model import ARIMA
from statsmodels.tsa.statespace.sarimax import SARIMAX
from sklearn.model_selection import TimeSeriesSplit
import pmdarima as pm

from math import ceil

import plotly.express as px
import plotly.graph_objects as go
from plotly.subplots import make_subplots
from sklearn.metrics import silhouette_samples, silhouette_score
from sklearn.manifold import TSNE
import plotly
plotly.offline.init_notebook_mode(connected=True)

from ipywidgets import interact, interactive, fixed, interact_manual,V
Box,HBox,Layout
import ipywidgets as widgets


sns.set()
```

```
warnings.filterwarnings("ignore")
```

```
/Users/AMINO/anaconda3/envs/learn-env/lib/python3.6/site-packages/st
atsmodels/tools/_testing.py:19: FutureWarning: pandas.util.testing i
s deprecated. Use the functions in the public API at pandas.testing
instead.
  import pandas.util.testing as tm
/Users/AMINO/anaconda3/envs/learn-env/lib/python3.6/site-packages/sk
learn/externals/six.py:31: DeprecationWarning:

The module is deprecated in version 0.21 and will be removed in vers
ion 0.23 since we've dropped support for Python 2.7. Please rely on
the official version of six (https://pypi.org/project/six/).

/Users/AMINO/anaconda3/envs/learn-env/lib/python3.6/site-packages/sk
learn/externals/joblib/__init__.py:15: DeprecationWarning:

sklearn.externals.joblib is deprecated in 0.21 and will be removed i
n 0.23. Please import this functionality directly from joblib, which
can be installed with: pip install joblib. If this warning is raised
when loading pickled models, you may need to re-serialize those mode
ls with scikit-learn 0.21+.
```

In [327]:
```
pyplot.style.use('fivethirtyeight')
```

Cleaning the Data & EDA

In [3]:
```
# Reading the dataset

Original_df = pd.read_excel (r'Online Retail.xlsx')
Original_df.head()
```

Out[3]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2010-12-01 08:26:00	2.55	17850.0	United Kingdom
1	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	2010-12-01 08:26:00	2.75	17850.0	United Kingdom
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom

In [4]:
```
# Checking the shape

Original_df.shape
```

Out[4]: `(541909, 8)`

In [5]:
```
# Checking my column IDs

Original_df.columns
```

Out[5]:
```
Index(['InvoiceNo', 'StockCode', 'Description', 'Quantity', 'Invoice
Date',
       'UnitPrice', 'CustomerID', 'Country'],
      dtype='object')
```

In [6]:
```
# How many unique values I have

Original_df.nunique()
```

Out[6]:
```
InvoiceNo      25900
StockCode       4070
Description     4223
Quantity         722
InvoiceDate    23260
UnitPrice       1630
CustomerID      4372
Country           38
dtype: int64
```

In [7]:
```
# Checking on the data type

Original_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 541909 entries, 0 to 541908
Data columns (total 8 columns):
 #   Column       Non-Null Count   Dtype
---  ------       --------------   -----
 0   InvoiceNo    541909 non-null  object
 1   StockCode    541909 non-null  object
 2   Description  540455 non-null  object
 3   Quantity     541909 non-null  int64
 4   InvoiceDate  541909 non-null  datetime64[ns]
 5   UnitPrice    541909 non-null  float64
 6   CustomerID   406829 non-null  float64
 7   Country      541909 non-null  object
dtypes: datetime64[ns](1), float64(2), int64(1), object(4)
memory usage: 33.1+ MB
```

In [8]:
```
# Checking the data-types of the data

Original_df.dtypes
```

Out[8]:
```
InvoiceNo              object
StockCode              object
Description            object
Quantity                int64
InvoiceDate    datetime64[ns]
UnitPrice             float64
CustomerID            float64
Country                object
dtype: object
```

In [9]:
```
# What are my countries

Original_df['Country'].unique()
```

Out[9]:
```
array(['United Kingdom', 'France', 'Australia', 'Netherlands', 'Germ
any',
       'Norway', 'EIRE', 'Switzerland', 'Spain', 'Poland', 'Portugal
',
       'Italy', 'Belgium', 'Lithuania', 'Japan', 'Iceland',
       'Channel Islands', 'Denmark', 'Cyprus', 'Sweden', 'Austria',
       'Israel', 'Finland', 'Bahrain', 'Greece', 'Hong Kong', 'Singa
pore',
       'Lebanon', 'United Arab Emirates', 'Saudi Arabia',
       'Czech Republic', 'Canada', 'Unspecified', 'Brazil', 'USA',
       'European Community', 'Malta', 'RSA'], dtype=object)
```

In [10]:
```
# Checking the different values for country in the dataset

Original_df['Country'].value_counts().head(20)
```

Out[10]:
```
United Kingdom     495478
Germany              9495
France               8557
EIRE                 8196
Spain                2533
Netherlands          2371
Belgium              2069
Switzerland          2002
Portugal             1519
Australia            1259
Norway               1086
Italy                 803
Channel Islands       758
Finland               695
Cyprus                622
Sweden                462
Unspecified           446
Austria               401
Denmark               389
Japan                 358
Name: Country, dtype: int64
```

In [11]:
```
# Checking on the values

Original_df.describe()
```

Out[11]:

	Quantity	UnitPrice	CustomerID
count	541909.000000	541909.000000	406829.000000
mean	9.552250	4.611114	15287.690570
std	218.081158	96.759853	1713.600303
min	-80995.000000	-11062.060000	12346.000000
25%	1.000000	1.250000	13953.000000
50%	3.000000	2.080000	15152.000000
75%	10.000000	4.130000	16791.000000
max	80995.000000	38970.000000	18287.000000

In [12]:
```
# Getting some more general information about the data

print("Number of transactions: ", Original_df['InvoiceNo'].nunique())
print("Number of products bought: ",Original_df['StockCode'].nunique()
)
print("Number of customers:", Original_df['CustomerID'].nunique() )
print("Percentage of customers NA: ", round(Original_df['CustomerID'].
isnull().sum() * 100 / len(Original_df),2),"%" )
print('Number of countries: ',Original_df['Country'].nunique())
```

```
Number of transactions:  25900
Number of products bought:  4070
Number of customers: 4372
Percentage of customers NA:  24.93 %
Number of countries:  38
```

In [13]:
```
# Checking how many quantity of products have been sold online from ea
ch country

Products_Sold = Original_df['Quantity'].groupby(Original_df['Country']
).agg('sum').sort_values(ascending = False)
print(Products_Sold)
```

```
Country
United Kingdom          4263829
Netherlands              200128
EIRE                     142637
Germany                  117448
France                   110480
Australia                 83653
Sweden                    35637
Switzerland               30325
Spain                     26824
Japan                     25218
Belgium                   23152
Norway                    19247
Portugal                  16180
Finland                   10666
Channel Islands            9479
Denmark                    8188
Italy                      7999
Cyprus                     6317
Singapore                  5234
Austria                    4827
Hong Kong                  4769
Israel                     4353
Poland                     3653
Unspecified                3300
Canada                     2763
Iceland                    2458
Greece                     1556
USA                        1034
United Arab Emirates        982
Malta                       944
Lithuania                   652
Czech Republic              592
European Community          497
Lebanon                     386
Brazil                      356
RSA                         352
Bahrain                     260
Saudi Arabia                 75
Name: Quantity, dtype: int64
```

In [ ]:

Checking and dealing with null/missing values

In [14]:
```
# Checking for Missing Values

Original_df.isnull().sum()
```

Out[14]:
```
InvoiceNo           0
StockCode           0
Description      1454
Quantity            0
InvoiceDate         0
UnitPrice           0
CustomerID     135080
Country             0
dtype: int64
```

As you can see CustomerID has a lot of null values and since I am doing sells prediction this feature cannot help our prediction at this moment so I am just going to drop the CustomerID column

In [15]:
```
df = Original_df.drop(columns=['CustomerID'])
```

In [16]:
```
# Checking to see if NaN values were filtered out

df.isnull().sum()
```

Out[16]:
```
InvoiceNo         0
StockCode         0
Description    1454
Quantity          0
InvoiceDate       0
UnitPrice         0
Country           0
dtype: int64
```

As you can see I still have null values left in the description so in our case instead deleting those rows I am going to impute them with 'UNKNOWN ITEM' at the moment

In [17]:
```
df['Description'] = df['Description'].fillna('UNKNOWN ITEM')
```

In [18]:
```
# Checking to see if NaN values were filtered out

df.isnull().sum()
```

Out[18]:
```
InvoiceNo      0
StockCode      0
Description    0
Quantity       0
InvoiceDate    0
UnitPrice      0
Country        0
dtype: int64
```

In [19]:
```
# I still have 1 missing value and I need to clear that out

df = df[df['Description'].notnull()]
```

In [20]:
```
df.isnull().sum()
```

Out[20]:
```
InvoiceNo      0
StockCode      0
Description    0
Quantity       0
InvoiceDate    0
UnitPrice      0
Country        0
dtype: int64
```

In [21]:
```
# Lets make all the describtion upper case so it looks more clean

df['Description'] = df['Description'].str.upper()
```

In [22]:
```
# Lets check to make sure if it worked and see what customers bought o
ften

df['Description'].value_counts().head()
```

Out[22]:
```
WHITE HANGING HEART T-LIGHT HOLDER    2369
REGENCY CAKESTAND 3 TIER              2200
JUMBO BAG RED RETROSPOT               2159
PARTY BUNTING                         1727
LUNCH BAG RED RETROSPOT               1638
Name: Description, dtype: int64
```

In [23]:
```
df.describe()
```

Out[23]:

	Quantity	UnitPrice
count	541909.000000	541909.000000
mean	9.552250	4.611114
std	218.081158	96.759853
min	-80995.000000	-11062.060000
25%	1.000000	1.250000
50%	3.000000	2.080000
75%	10.000000	4.130000
max	80995.000000	38970.000000

Exploring the negative Quantity and UnitePrice

In [24]:
```
df[df['Quantity'] < 0].head(10)
```

Out[24]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	Country
141	C536379	D	DISCOUNT	-1	2010-12-01 09:41:00	27.50	United Kingdom
154	C536383	35004C	SET OF 3 COLOURED FLYING DUCKS	-1	2010-12-01 09:49:00	4.65	United Kingdom
235	C536391	22556	PLASTERS IN TIN CIRCUS PARADE	-12	2010-12-01 10:24:00	1.65	United Kingdom
236	C536391	21984	PACK OF 12 PINK PAISLEY TISSUES	-24	2010-12-01 10:24:00	0.29	United Kingdom
237	C536391	21983	PACK OF 12 BLUE PAISLEY TISSUES	-24	2010-12-01 10:24:00	0.29	United Kingdom
238	C536391	21980	PACK OF 12 RED RETROSPOT TISSUES	-24	2010-12-01 10:24:00	0.29	United Kingdom
239	C536391	21484	CHICK GREY HOT WATER BOTTLE	-12	2010-12-01 10:24:00	3.45	United Kingdom
240	C536391	22557	PLASTERS IN TIN VINTAGE PAISLEY	-12	2010-12-01 10:24:00	1.65	United Kingdom
241	C536391	22553	PLASTERS IN TIN SKULLS	-24	2010-12-01 10:24:00	1.65	United Kingdom
939	C536506	22960	JAM MAKING SET WITH JARS	-6	2010-12-01 12:38:00	4.25	United Kingdom

As you can see all the negative quantity starts with 'C' in InvoiceNo. The negative quantities appears to be return/canceled/discount, and maybe unknown items.

In [25]:

```
# For our analysis lets remove them for now

df = df[df['Quantity'] > 0]
```

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

Out[26]:

	Quantity	UnitPrice
count	531285.000000	531285.000000
mean	10.655262	3.857296
std	156.830323	41.810047
min	1.000000	-11062.060000
25%	1.000000	1.250000
50%	3.000000	2.080000
75%	10.000000	4.130000
max	80995.000000	13541.330000

As we can see we still have negative UnitPrice so let's filter out those as well

In [27]:
```
df = df[df['UnitPrice'] > 0]
```

In [28]:
```
df.describe()
```

Out[28]:

	Quantity	UnitPrice
count	530104.000000	530104.000000
mean	10.542037	3.907625
std	155.524124	35.915681
min	1.000000	0.001000
25%	1.000000	1.250000
50%	3.000000	2.080000
75%	10.000000	4.130000
max	80995.000000	13541.330000

In [29]:
```
df.head()
```

Out[29]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	Country
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2010-12-01 08:26:00	2.55	United Kingdom
1	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00	3.39	United Kingdom
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	2010-12-01 08:26:00	2.75	United Kingdom
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	2010-12-01 08:26:00	3.39	United Kingdom
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	2010-12-01 08:26:00	3.39	United Kingdom

In [ ]:

Checking for duplicates

In [30]:
```
# Checking for duplicates

print("Number of duplicated transactions:", len(df[df.duplicated()]))
```

```
Number of duplicated transactions: 5226
```

In [31]:
```
# Lets remove the duplicates

df.drop_duplicates(inplace = True)
```

In [32]:
```
# Checking again for duplicates

print("Number of duplicated transactions:", len(df[df.duplicated()]))
```

```
Number of duplicated transactions: 0
```

In [33]:
```
df.shape
```

Out[33]: (524878, 7)

Feature Engineering

In [34]:
```
df.isnull().values.any()
```

Out[34]:
```
False
```

In [35]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 524878 entries, 0 to 541908
Data columns (total 7 columns):
 #   Column       Non-Null Count   Dtype
---  ------       --------------   -----
 0   InvoiceNo    524878 non-null  object
 1   StockCode    524878 non-null  object
 2   Description  524878 non-null  object
 3   Quantity     524878 non-null  int64
 4   InvoiceDate  524878 non-null  datetime64[ns]
 5   UnitPrice    524878 non-null  float64
 6   Country      524878 non-null  object
dtypes: datetime64[ns](1), float64(1), int64(1), object(4)
memory usage: 32.0+ MB
```

Creating total sales column and Removing outliers

In [36]:
```
# Creating a another new column with the total value of each order

df['Total_Sales_Amount'] = df['Quantity'] * df['UnitPrice']
```

In [37]:
```
# sorting the dataset by sales amount

df.sort_values(by = 'Total_Sales_Amount')
```

Out[37]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	Country	To
279045	561226	PADS	PADS TO MATCH ALL CUSHIONS	1	2011-07-26 10:13:00	0.001	United Kingdom	
359871	568200	PADS	PADS TO MATCH ALL CUSHIONS	1	2011-09-25 14:58:00	0.001	United Kingdom	
157195	550193	PADS	PADS TO MATCH ALL CUSHIONS	1	2011-04-15 09:27:00	0.001	United Kingdom	
361741	568375	BANK CHARGES	BANK CHARGES	1	2011-09-26 17:01:00	0.001	United Kingdom	
423991	573174	16218	CARTOON PENCIL SHARPENERS	1	2011-10-28 10:25:00	0.060	United Kingdom	
...	...	...	...	...	...	...	...	
299982	A563185	B	ADJUST BAD DEBT	1	2011-08-12 14:50:00	11062.060	United Kingdom	
15017	537632	AMAZONFEE	AMAZON FEE	1	2010-12-07 15:08:00	13541.330	United Kingdom	
222680	556444	22502	PICNIC BASKET WICKER 60 PIECES	60	2011-06-10 15:28:00	649.500	United Kingdom	
61619	541431	23166	MEDIUM CERAMIC TOP STORAGE JAR	74215	2011-01-18 10:01:00	1.040	United Kingdom	
540421	581483	23843	PAPER CRAFT , LITTLE BIRDIE	80995	2011-12-09 09:15:00	2.080	United Kingdom	

524878 rows × 8 columns

Let's break down our InvoiceDate to year, month, hour, and other categories

In [38]:
```
df['InvoiceDate']=pd.to_datetime(df['InvoiceDate'])
df['Year']=df.InvoiceDate.dt.year
df['Month']=df.InvoiceDate.dt.month
df['Week']=df['InvoiceDate'].dt.week
df['Year_Month']=df.InvoiceDate.dt.to_period('M')
df['Hour']=df.InvoiceDate.dt.hour
df['Day']=df.InvoiceDate.dt.day
df['WeekDay'] = df.InvoiceDate.dt.day_name()
df['Quarter'] = df.Month.apply(lambda m:'Q'+str(ceil(m/4)))
df['Date']=pd.to_datetime(df[['Year','Month','Day']])
```

In [39]:
```
df.head(30)
```

Out[39]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	Country	Total_Sale
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2010-12-01 08:26:00	2.55	United Kingdom	
1	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00	3.39	United Kingdom	
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	2010-12-01 08:26:00	2.75	United Kingdom	
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	2010-12-01 08:26:00	3.39	United Kingdom	
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	2010-12-01 08:26:00	3.39	United Kingdom	
5	536365	22752	SET 7 BABUSHKA NESTING BOXES	2	2010-12-01 08:26:00	7.65	United Kingdom	
6	536365	21730	GLASS STAR FROSTED T-LIGHT HOLDER	6	2010-12-01 08:26:00	4.25	United Kingdom	
7	536366	22633	HAND WARMER UNION JACK	6	2010-12-01 08:28:00	1.85	United Kingdom	
			HAND					

8	536366	22632	WARMER RED POLKA DOT	6	2010-12-01 08:28:00	1.85	United Kingdom
9	536367	84879	ASSORTED COLOUR BIRD ORNAMENT	32	2010-12-01 08:34:00	1.69	United Kingdom
10	536367	22745	POPPY'S PLAYHOUSE BEDROOM	6	2010-12-01 08:34:00	2.10	United Kingdom
11	536367	22748	POPPY'S PLAYHOUSE KITCHEN	6	2010-12-01 08:34:00	2.10	United Kingdom
12	536367	22749	FELTCRAFT PRINCESS CHARLOTTE DOLL	8	2010-12-01 08:34:00	3.75	United Kingdom
13	536367	22310	IVORY KNITTED MUG COSY	6	2010-12-01 08:34:00	1.65	United Kingdom
14	536367	84969	BOX OF 6 ASSORTED COLOUR TEASPOONS	6	2010-12-01 08:34:00	4.25	United Kingdom
15	536367	22623	BOX OF VINTAGE JIGSAW BLOCKS	3	2010-12-01 08:34:00	4.95	United Kingdom
16	536367	22622	BOX OF VINTAGE ALPHABET BLOCKS	2	2010-12-01 08:34:00	9.95	United Kingdom
17	536367	21754	HOME BUILDING BLOCK WORD	3	2010-12-01 08:34:00	5.95	United Kingdom
18	536367	21755	LOVE BUILDING BLOCK WORD	3	2010-12-01 08:34:00	5.95	United Kingdom
19	536367	21777	RECIPE BOX WITH METAL HEART	4	2010-12-01 08:34:00	7.95	United Kingdom
20	536367	48187	DOORMAT NEW ENGLAND	4	2010-12-01 08:34:00	7.95	United Kingdom
21	536368	22960	JAM MAKING SET WITH JARS	6	2010-12-01 08:34:00	4.25	United Kingdom

22	536368	22913	RED COAT RACK PARIS FASHION	3	2010-12-01 08:34:00	4.95	United Kingdom
23	536368	22912	YELLOW COAT RACK PARIS FASHION	3	2010-12-01 08:34:00	4.95	United Kingdom
24	536368	22914	BLUE COAT RACK PARIS FASHION	3	2010-12-01 08:34:00	4.95	United Kingdom
25	536369	21756	BATH BUILDING BLOCK WORD	3	2010-12-01 08:35:00	5.95	United Kingdom
26	536370	22728	ALARM CLOCK BAKELIKE PINK	24	2010-12-01 08:45:00	3.75	France
27	536370	22727	ALARM CLOCK BAKELIKE RED	24	2010-12-01 08:45:00	3.75	France
28	536370	22726	ALARM CLOCK BAKELIKE GREEN	12	2010-12-01 08:45:00	3.75	France
29	536370	21724	PANDA AND BUNNIES STICKER SHEET	12	2010-12-01 08:45:00	0.85	France

In [40]:
```
df.dtypes
```

Out[40]:
```
InvoiceNo                       object
StockCode                       object
Description                     object
Quantity                         int64
InvoiceDate             datetime64[ns]
UnitPrice                      float64
Country                         object
Total_Sales_Amount             float64
Year                             int64
Month                            int64
Week                             int64
Year_Month                   period[M]
Hour                             int64
Day                              int64
WeekDay                         object
Quarter                         object
Date                    datetime64[ns]
dtype: object
```

In [41]:
```
top_products = df['Description'].value_counts()[:15]
```

Looking for top products

In [42]:
```
top_products
```

Out[42]:
```
WHITE HANGING HEART T-LIGHT HOLDER    2311
JUMBO BAG RED RETROSPOT               2109
REGENCY CAKESTAND 3 TIER              2007
PARTY BUNTING                         1699
LUNCH BAG RED RETROSPOT               1581
ASSORTED COLOUR BIRD ORNAMENT         1476
SET OF 3 CAKE TINS PANTRY DESIGN      1392
PACK OF 72 RETROSPOT CAKE CASES       1352
LUNCH BAG  BLACK SKULL.               1301
NATURAL SLATE HEART CHALKBOARD        1255
JUMBO BAG PINK POLKADOT               1232
HEART OF WICKER SMALL                 1219
JUMBO STORAGE BAG SUKI                1194
PAPER CHAIN KIT 50'S CHRISTMAS        1184
JUMBO SHOPPER VINTAGE RED PAISLEY     1180
Name: Description, dtype: int64
```

Analyzing and saving top products

In [43]:
```
dfp1 = df[df['Description'] == 'WHITE HANGING HEART T-LIGHT HOLDER']
```

In [44]:
```
dfp1.head()
```

Out[44]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	Country	Total_Sale
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2010-12-01 08:26:00	2.55	United Kingdom	
49	536373	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2010-12-01 09:02:00	2.55	United Kingdom	
66	536375	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2010-12-01 09:32:00	2.55	United Kingdom	
220	536390	85123A	WHITE HANGING HEART T-LIGHT HOLDER	64	2010-12-01 10:19:00	2.55	United Kingdom	
262	536394	85123A	WHITE HANGING HEART T-LIGHT HOLDER	32	2010-12-01 10:39:00	2.55	United Kingdom	

In [45]:
```
dfp2 = df[df['Description'] == 'JUMBO BAG RED RETROSPOT']
dfp2.head()
```

Out[45]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	Country	Total_Sa
177	536386	85099B	JUMBO BAG RED RETROSPOT	100	2010-12-01 09:57:00	1.65	United Kingdom	
234	536390	85099B	JUMBO BAG RED RETROSPOT	100	2010-12-01 10:19:00	1.65	United Kingdom	
533	536409	85099B	JUMBO BAG RED RETROSPOT	2	2010-12-01 11:45:00	1.95	United Kingdom	
790	536464	85099B	JUMBO BAG RED RETROSPOT	1	2010-12-01 12:23:00	1.95	United Kingdom	
1069	536522	85099B	JUMBO BAG RED RETROSPOT	1	2010-12-01 12:49:00	1.95	United Kingdom	

In [46]:
```
dfp3 = df[df['Description'] == 'REGENCY CAKESTAND 3 TIER']
dfp3.head()
```

Out[46]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	Country	Total_Sa
880	536477	22423	REGENCY CAKESTAND 3 TIER	16	2010-12-01 12:27:00	10.95	United Kingdom	
936	536502	22423	REGENCY CAKESTAND 3 TIER	2	2010-12-01 12:36:00	12.75	United Kingdom	
1092	536525	22423	REGENCY CAKESTAND 3 TIER	2	2010-12-01 12:54:00	12.75	United Kingdom	
1155	536528	22423	REGENCY CAKESTAND 3 TIER	1	2010-12-01 13:17:00	12.75	United Kingdom	
1197	536530	22423	REGENCY CAKESTAND 3 TIER	1	2010-12-01 13:21:00	12.75	United Kingdom	

In [47]:
```
dfp4 = df[df['Description'] == 'PARTY BUNTING']
dfp4.head()
```

Out[47]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	Country	Total_Sal
5535	536864	47566	PARTY BUNTING	1	2010-12-03 11:27:00	9.32	United Kingdom	
5656	536865	47566	PARTY BUNTING	3	2010-12-03 11:28:00	9.32	United Kingdom	
6022	536876	47566	PARTY BUNTING	2	2010-12-03 11:36:00	8.47	United Kingdom	
6572	536956	47566	PARTY BUNTING	5	2010-12-03 12:43:00	4.65	United Kingdom	
7904	537065	47566	PARTY BUNTING	5	2010-12-05 11:57:00	4.65	France	

In [48]:
```
dfp5 = df[df['Description'] == 'LUNCH BAG RED RETROSPOT']
dfp5.head(0)
```

Out[48]:

InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	Country	Total_Sales_Ar

In [49]:
```
dfp6 = df[df['Description'] == 'ASSORTED COLOUR BIRD ORNAMENT']
dfp6.head()
```

Out[49]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	Country	Total_Sale
9	536367	84879	ASSORTED COLOUR BIRD ORNAMENT	32	2010-12-01 08:34:00	1.69	United Kingdom	
250	536392	84879	ASSORTED COLOUR BIRD ORNAMENT	16	2010-12-01 10:29:00	1.69	United Kingdom	
265	536395	84879	ASSORTED COLOUR BIRD ORNAMENT	32	2010-12-01 10:47:00	1.69	United Kingdom	
458	536408	84879	ASSORTED COLOUR BIRD ORNAMENT	8	2010-12-01 11:41:00	1.69	United Kingdom	
769	536460	84879	ASSORTED COLOUR BIRD ORNAMENT	24	2010-12-01 12:22:00	1.69	United Kingdom	

In [50]:
```
dfp7 = df[df['Description'] == 'PACK OF 72 RETROSPOT CAKE CASES']
dfp7.head()
```

Out[50]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	Country	Total_Sal
96	536378	21212	PACK OF 72 RETROSPOT CAKE CASES	120	2010-12-01 09:37:00	0.42	United Kingdom	
268	536395	21212	PACK OF 72 RETROSPOT CAKE CASES	24	2010-12-01 10:47:00	0.55	United Kingdom	
408	536404	21212	PACK OF 72 RETROSPOT CAKE CASES	24	2010-12-01 11:29:00	0.55	United Kingdom	
469	536408	21212	PACK OF 72 RETROSPOT CAKE CASES	24	2010-12-01 11:41:00	0.55	United Kingdom	
657	536415	21212	PACK OF 72 RETROSPOT CAKE CASES	2	2010-12-01 11:57:00	0.55	United Kingdom	

In [51]:
```
dfp8 = df[df['Description'] == 'LUNCH BAG  BLACK SKULL.']
dfp8.head()
```

Out[51]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	Country	Total_Sal
413	536404	20727	LUNCH BAG BLACK SKULL.	10	2010-12-01 11:29:00	1.65	United Kingdom	
546	536412	20727	LUNCH BAG BLACK SKULL.	3	2010-12-01 11:49:00	1.65	United Kingdom	
1426	536542	20727	LUNCH BAG BLACK SKULL.	10	2010-12-01 14:11:00	1.65	United Kingdom	
2315	536576	20727	LUNCH BAG BLACK SKULL.	60	2010-12-01 16:11:00	1.45	United Kingdom	
2338	536579	20727	LUNCH BAG BLACK SKULL.	60	2010-12-01 16:16:00	1.45	United Kingdom	

In [52]:
```
dfp9 = df[df['Description'] == 'HEART OF WICKER SMALL']
dfp9.head()
```

Out[52]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	Country	Total_Sale
160	536384	22469	HEART OF WICKER SMALL	40	2010-12-01 09:53:00	1.45	United Kingdom	
195	536388	22469	HEART OF WICKER SMALL	12	2010-12-01 09:59:00	1.65	United Kingdom	
392	536404	22469	HEART OF WICKER SMALL	12	2010-12-01 11:29:00	1.65	United Kingdom	
634	536415	22469	HEART OF WICKER SMALL	5	2010-12-01 11:57:00	1.65	United Kingdom	
995	536520	22469	HEART OF WICKER SMALL	1	2010-12-01 12:43:00	1.65	United Kingdom	

In [53]:
```
dfp10 = df[df['Description'] == 'JUMBO SHOPPER VINTAGE RED PAISLEY']
dfp10.head()
```

Out[53]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	Country	Total_Sal
108	536381	22411	JUMBO SHOPPER VINTAGE RED PAISLEY	10	2010-12-01 09:41:00	1.95	United Kingdom	
145	536382	22411	JUMBO SHOPPER VINTAGE RED PAISLEY	10	2010-12-01 09:45:00	1.95	United Kingdom	
881	536477	22411	JUMBO SHOPPER VINTAGE RED PAISLEY	10	2010-12-01 12:27:00	1.95	United Kingdom	
1107	536526	22411	JUMBO SHOPPER VINTAGE RED PAISLEY	10	2010-12-01 12:58:00	1.95	United Kingdom	
1136	536528	22411	JUMBO SHOPPER VINTAGE RED PAISLEY	1	2010-12-01 13:17:00	1.95	United Kingdom	

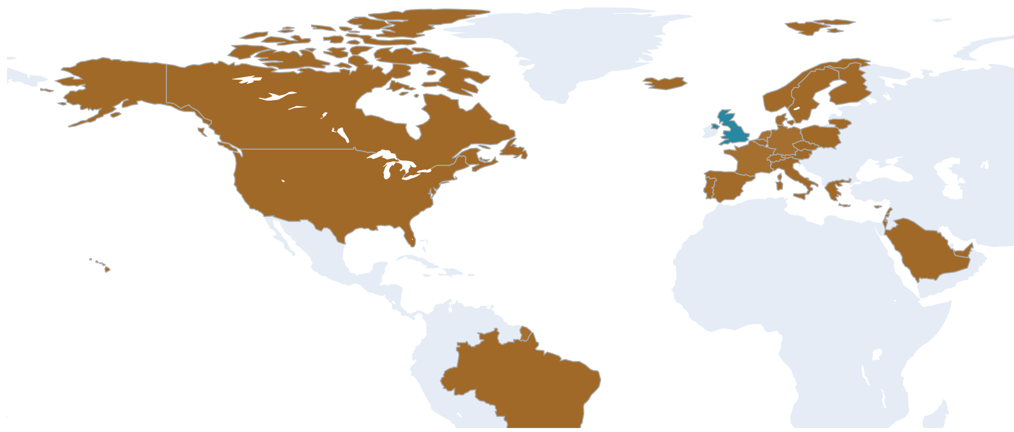
In [ ]:

Visualizations

For the first visualization let's see which countries the company doing business with

In [54]:
```
# I want to create a map that shows countries and their total sales
grp_data = df.groupby(by='Country')['Total_Sales_Amount'].sum().sort_v
alues(ascending=False).reset_index()

# below function is going help me to build a Choropleth Map using go.C
horopleth graph object

fig = go.Figure(data=go.Choropleth(
    locations = grp_data['Country'],
    z = grp_data['Total_Sales_Amount'],
    text = grp_data['Country'],
    colorscale = 'earth',
    locationmode = 'country names',
    autocolorscale=False,
    reversescale=False,
    marker_line_color='darkgray',
    marker_line_width=0.5,
    colorbar_title = 'Total_Sales_Amount',
))


fig.update_layout(
    title_text='Sales by country',
    geo=dict(showframe=False,showcoastlines=False,projection_type='equ
irectangular'),
    annotations = [dict(x=0.55,y=0.1,xref='paper',yref='paper',showarr
ow = False)])

fig.show()

del grp_data
```

Sales by country

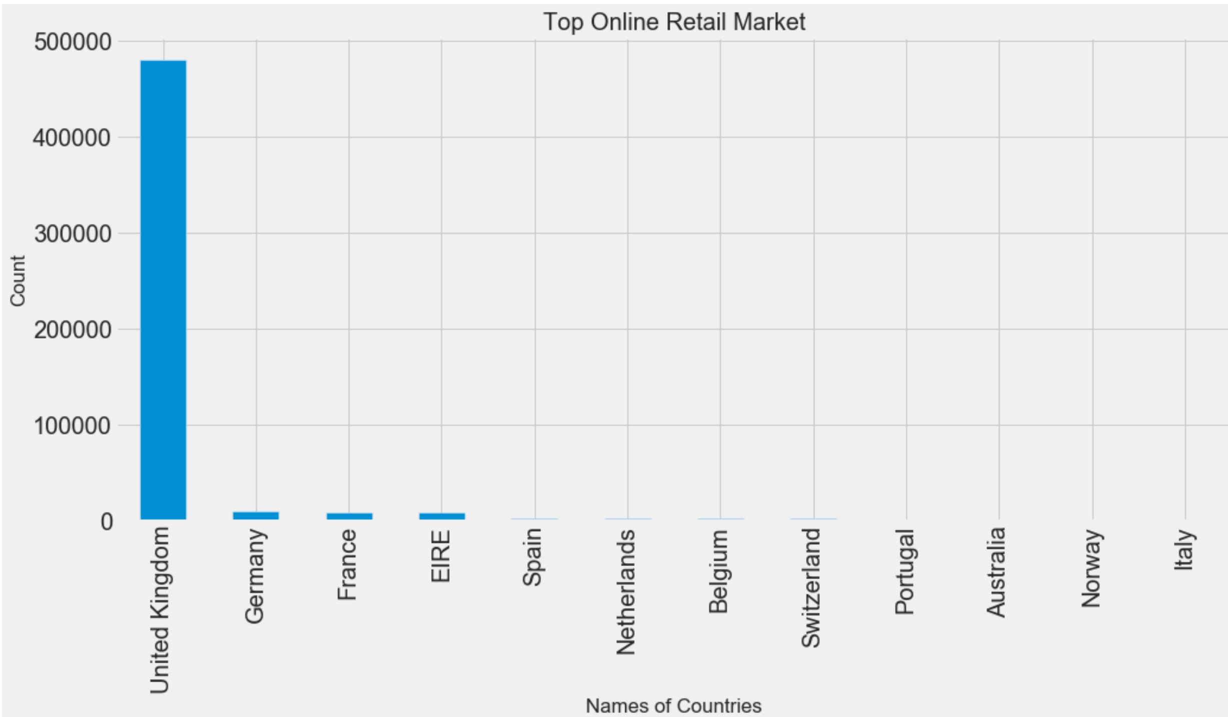


If you hover over the map you can see each country total sales. As we can see UK has the highest total sales

```
In [ ]:
```

Which country ordered the most

In [55]:
```
df['Country'].value_counts().head(12).plot.bar(figsize = (15, 7))
plt.title('Top Online Retail Market', fontsize = 20)
plt.xlabel('Names of Countries')
plt.ylabel('Count')
plt.xticks(fontsize = 20)
plt.yticks(fontsize = 20)
plt.show()
```

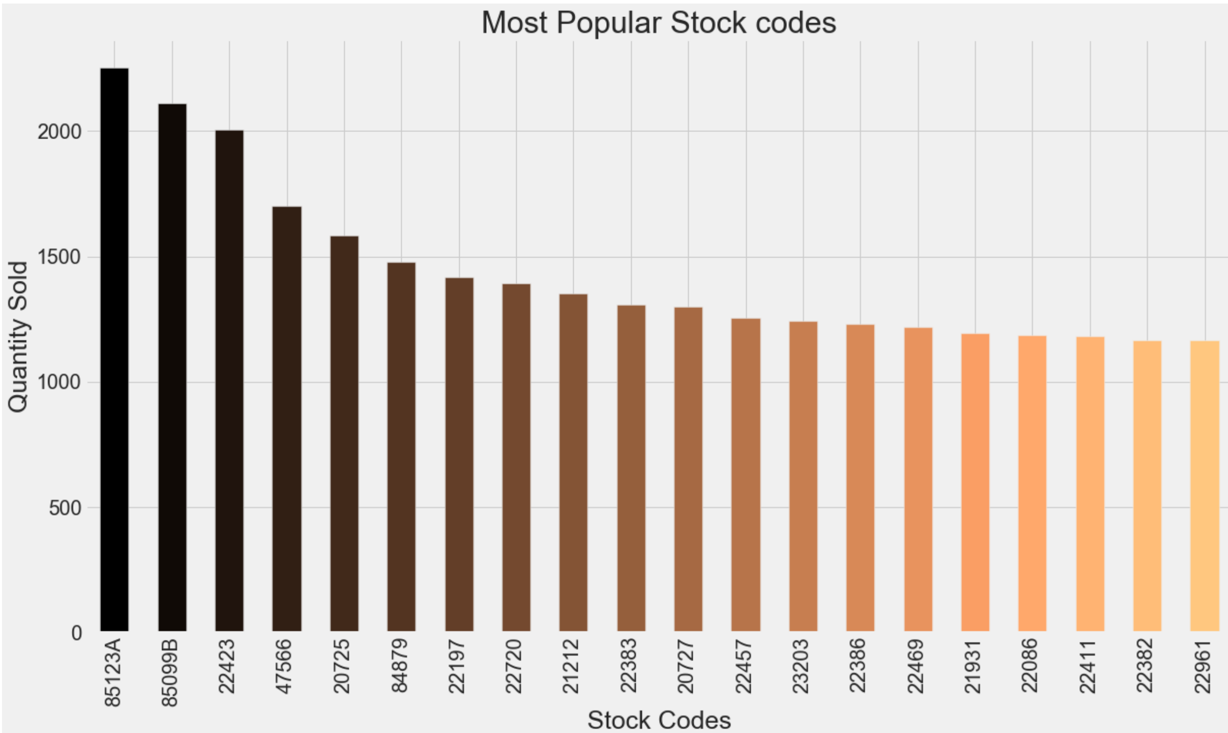


The biggest number of orders are made in United Kingdom. Based on this graph I can say that outside UK, Germany, France, and Ireland (EIRE) are the top business customers.

In [ ]:

Let's look at the top stockcodes on the dataset

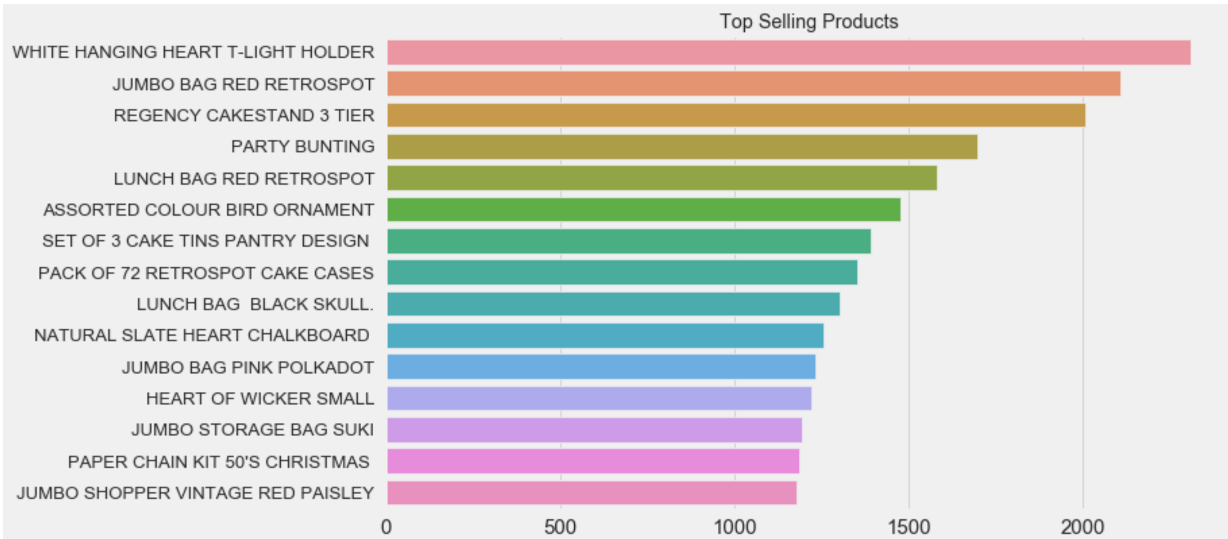
In [56]:
```
color = plt.cm.copper(np.linspace(0, 1, 20))
df['StockCode'].value_counts().head(20).plot.bar(color = color, figsiz
e = (18, 10))
plt.title('Most Popular Stock codes', fontsize = 30)
plt.xticks(fontsize = 20)
plt.yticks(fontsize = 20)
plt.xlabel('Stock Codes', fontsize=25)
plt.ylabel('Quantity Sold', fontsize=25)
plt.show()
```



Each of these stock codes is associate with a specific product. If we look at these stock codes it will show us which product is it. As I can see stock: '85123A' is the most popular product. If you look up that stock code, that code will show the 'White Hanging Heart T-Light Holder' product.

Now lets check what is our top selling products

In [57]:
```
top_products = df['Description'].value_counts()[:15]
plt.figure(figsize=(10,6))
sns.set_context("paper", font_scale=1.5)
sns.barplot(y = top_products.index,
            x = top_products.values)
plt.title("Top Selling Products")
plt.xticks(fontsize = 15)
plt.show();
```



In my modeling and forecasting I am going to select the top 10 product to do my analysis. I can choose more if needed or if there is a specific product the manager wants to know, but for my analysis I am going to choose only 10 from this list. I can see that the 'White Hanging Heart T-Light Holder', 'Jumbo Bad Red Retrospot', and Regency Cakestand 3 Tier are most sold products from this company.

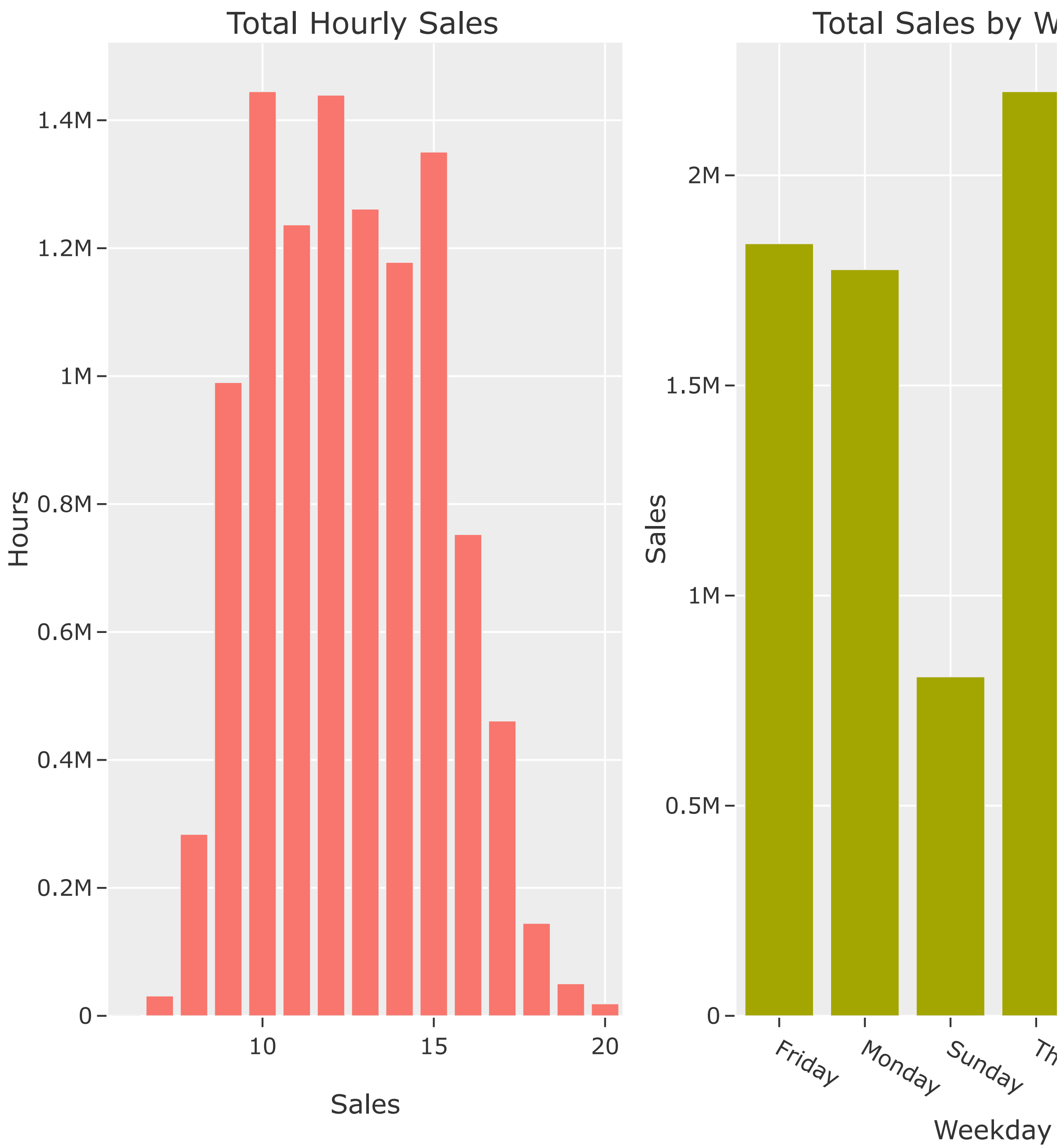
It is important to have those items in stock and have enough inventory for the customers. When I do the forecast and modeling, my model can help the business with inventory management to know which day, week, or month they are going to have the most sales for that Particular product.

In [ ]:

Let's see the total hourly and weekday sales

In [ ]:

In [264]:
```
sales_by_hour = df.groupby(by='Hour')['Total_Sales_Amount'].sum().rese
t_index()
sales_by_weekday = df.groupby(by='WeekDay')['Total_Sales_Amount'].sum(
).reset_index()

fig = make_subplots(rows=1, cols=2,subplot_titles=("Total Hourly Sales
", "Total Sales by Weekday"))
fig.add_trace(go.Bar(y=sales_by_hour.Total_Sales_Amount, x=sales_by_ho
ur.Hour,orientation='v'),row=1, col=1)
fig.add_trace(go.Bar(x=sales_by_weekday.WeekDay, y=sales_by_weekday.To
tal_Sales_Amount),row=1, col=2)
fig.update_layout(height=700, width=800,template='ggplot2')
fig.update_xaxes(title_text="Sales", row=1, col=1)
fig.update_xaxes(title_text="Weekday", row=1, col=2)
fig.update_yaxes(title_text="Hours", row=1, col=1)
fig.update_yaxes(title_text="Sales", row=1, col=2)
fig.show()
```



Based on the graph most of the sales happens around 10 AM and 12 PM, and also most sales transactions happen on Tuesday and Thursday. It is interesting to see no sales happen on Saturdays

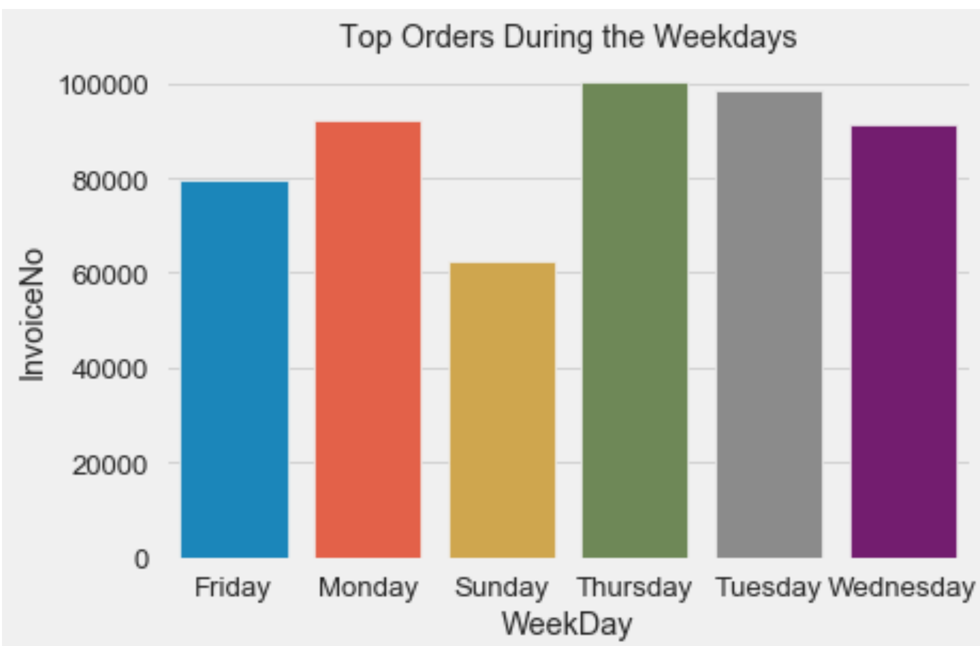
Let's how many orders we are getting during the weekdays

In [276]:
```
Dy = pd.DataFrame(df.groupby(['WeekDay'])['InvoiceNo'].count())
Dy = Dy.reindex(['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday
','Saturday','Sunday']).reset_index
Dy
```

Out[276]:
```
<bound method DataFrame.reset_index of            InvoiceNo
WeekDay
Monday        92466.0
Tuesday       98726.0
Wednesday     91467.0
Thursday     100213.0
Friday        79667.0
Saturday          NaN
Sunday        62339.0>
```

Interestingly we have got no orders on Saturday's. Maybe because they are closed or don't process invoices that day?

In [279]:
```
P1 = pd.DataFrame(df.groupby(['WeekDay'])['InvoiceNo'].count()).reset_
index()
ax = sns.barplot(x="WeekDay", y="InvoiceNo", data = P1)
ax.set_title('Top Orders During the Weekdays')
plt.show()
```

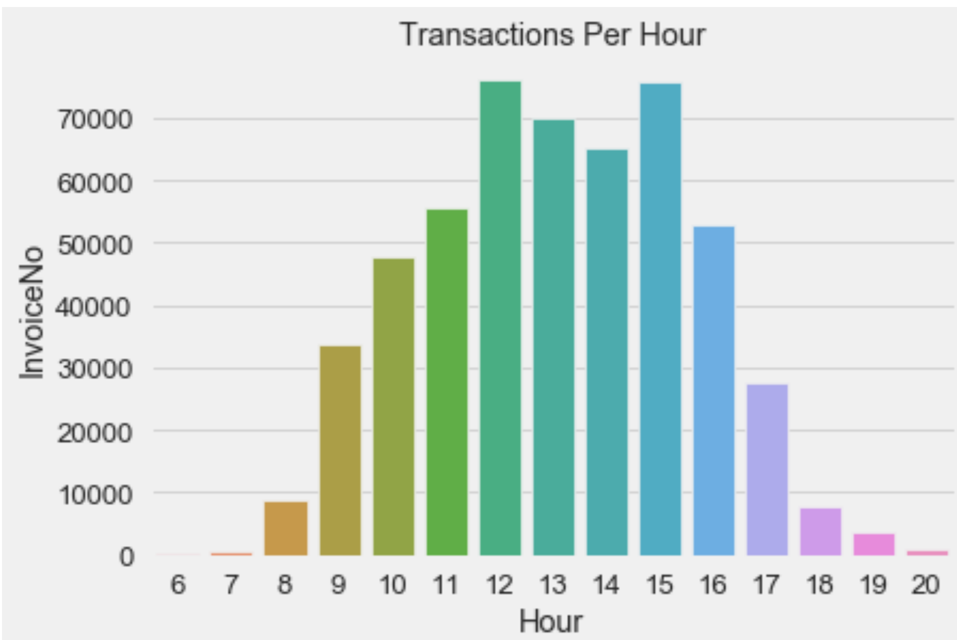


Based on this graph I can that the most orders are placed on Thursday and Tuesday. This can change over the years as I collect more data. There is also an interesting factor that there is no Saturday orders? Why? Are they closed or they don't process orders on Saturdays? This is something I have to investigate in my future analysis.

In [ ]:

Let's check our transactions per hour

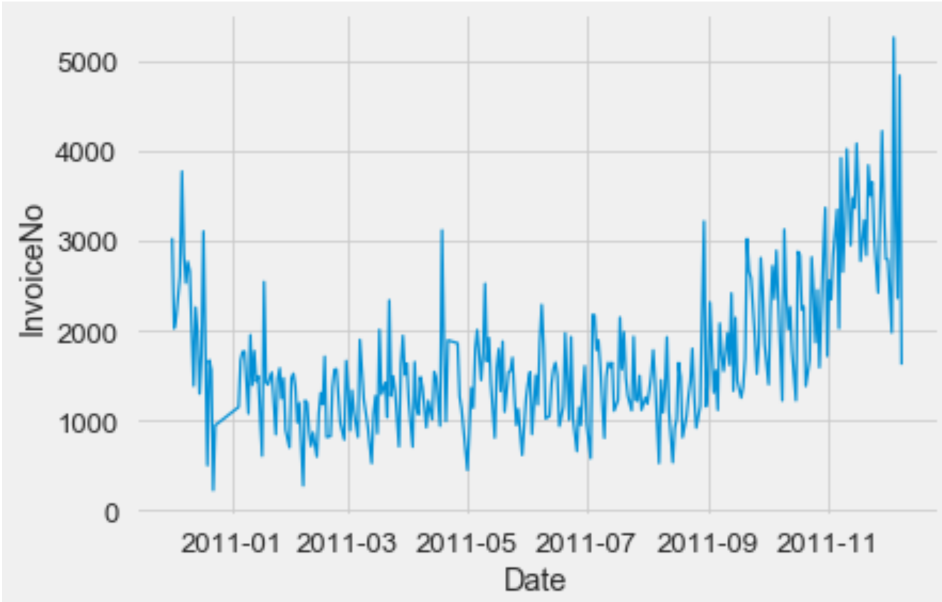
In [62]:
```
P2 = pd.DataFrame(df.groupby(['Hour'])['InvoiceNo'].count()).reset_ind
ex()
ax = sns.barplot(x= 'Hour', y= 'InvoiceNo', data = P2)
ax.set_title('Transactions Per Hour')
plt.show()
```



The most number of transactions is done between 12 p.m. and 2 p.m., people tend to make there purchase during the lunch time. There aren't any transactions after 8 p.m. till 6 a.m.

In [ ]:

In [65]:
```
P3 = pd.DataFrame(df.groupby(['Date'])['InvoiceNo'].count()).reset_ind
ex()
ax = sns.lineplot(x="Date", y="InvoiceNo", data = P3)
```



There are more purchasing made at the end of the year

As shown in the graph there is upward trend during the year especially close to holidays. This is also a good indication that the company is healthy and doing well. With the model I well be making it can forecast the trend so it can prepare the company for feature Inventory management and sales.

In [ ]:

Modeling

I was recently tasked with creating a weekly, and daily sales forecast because it can help business inventory management and sales. For my research, I will be doing time series analysis using ARIMA model to forecast

An autoregressive integrated moving average, or ARIMA, is a statistical analysis model that uses time series data to either better understand the data set or to predict future trends.

The first step in the model-building process is to plot the series and look for any evidence that the mean or variance is not stationary. (The ARIMA procedure assumes that the original series is stationary.)

Weekly Sales

In [280]:
```
ds_weekly = df.groupby(by=['Year','Week'])['Total_Sales_Amount'].sum()
.reset_index()
```

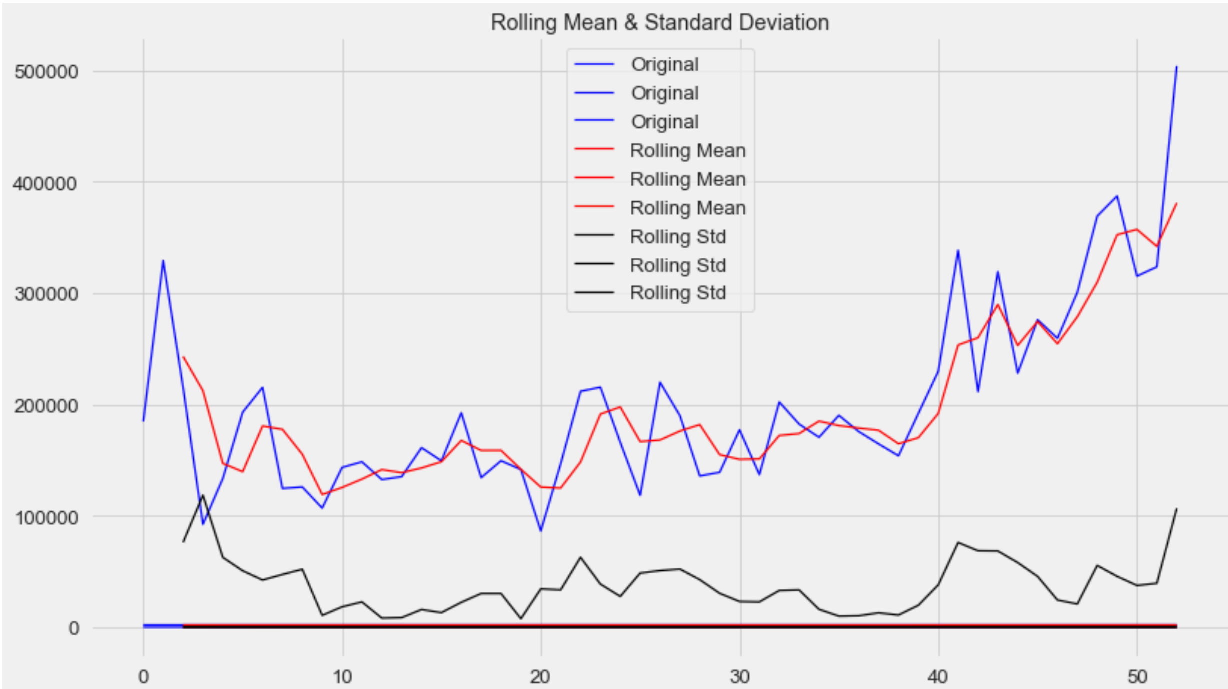
In [281]:
```
ds_weekly.head()
```

Out[281]:

	Year	Week	Total_Sales_Amount
0	2010	48	184669.47
1	2010	49	329108.22
2	2010	50	215357.04
3	2010	51	92318.00
4	2011	1	133429.72

In [282]:
```
roll_mean = ds_weekly.rolling(window=3, center=False).mean()
roll_std = ds_weekly.rolling(window=3, center=False).std()
```

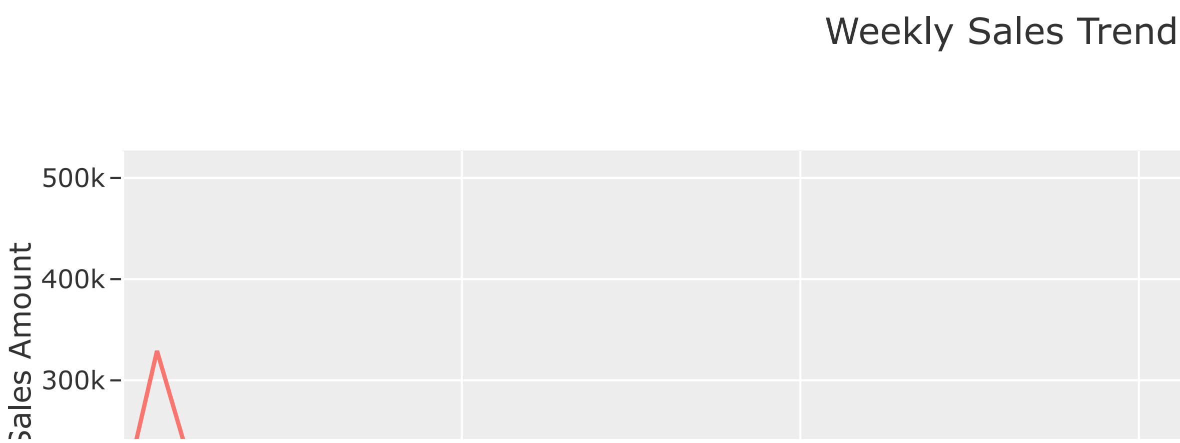
In [283]:
```
fig = plt.figure(figsize=(12,7))
plt.plot(ds_weekly, color='blue', label='Original')
plt.plot(roll_mean, color='red', label='Rolling Mean')
plt.plot(roll_std, color='black', label = 'Rolling Std')
plt.legend(loc='best')
plt.title('Rolling Mean & Standard Deviation')
plt.show(block=False)
```



In [ ]:

Weekly Sales: Weekly Trend

In [284]:
```
fig = go.Figure(data=[go.Scatter(x=ds_weekly.index,y=ds_weekly.Total_S
ales_Amount)])
fig.update_layout(xaxis_title="Week",yaxis_title="Total Sales Amount",
title='Weekly Sales Trend',height=400,template='ggplot2')
fig.show()
```



As we can see in the graph above, the weekly trend looks like a Exponential trend.

A typical example could be a company's sales. Initially, when small companies start to grow, there sales could be slower; but when their product catches people's attention, the sales can start to grow exponentially.

Weekly Sales: Test of Stationarity of Actual Series

It is clear that the level of the series is not stationary. Some degree of differencing will be necessary to stabilize the series level

In [285]:
```
output = adfuller(ds_weekly.Total_Sales_Amount)
print('**************************Week*********************************')
print('ADF Statistic: {0:.2f} and P value:{1:.5f}'.format(*output))
print("As we can see the p value is extreemly high which indicates that we are fail to reject null hypothesis " \
      "and can conclude that series is not stationary ")
```

```
**************************Week********************************
ADF Statistic: 1.59 and P value:0.99784
As we can see the p value is extreemly high which indicates that we
are fail to reject null hypothesis and can conclude that series is n
ot stationary
```

Weekly Sales: Test of Stationarity with 1 differencing of series

Test of Stationarity with 1 differencing of series.

Differencing is a method of transforming a non-stationary time series into a stationary one. This is an important step in preparing data to be used in an ARIMA model.

In [286]:
```
d=1
print('**************************Week*********************************')
series = ds_weekly.Total_Sales_Amount.diff(d)
series = series.dropna()
output = adfuller(series)

print('ADF Statistic: {0:.2f} and P value:{1:.5f}'.format(*output))
```

```
**************************Week********************************
ADF Statistic: -7.39 and P value:0.00000
```

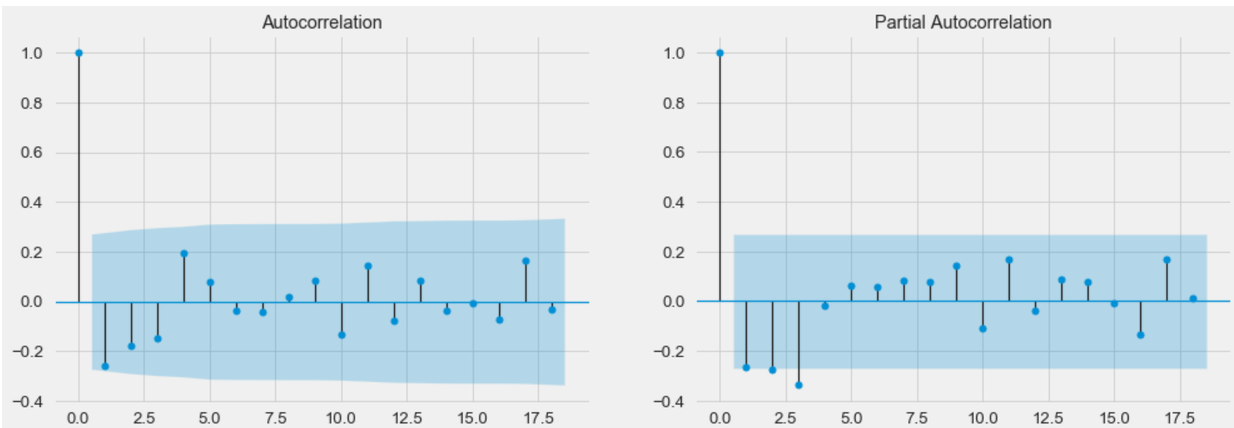
As we can see the p value is close to zero which is less than .05 hence we reject null hypothesis

Weekly Sales: ACF & PACF

Autocorrelation refers to how correlated a time series is with its past values whereas the ACF is the plot used to see the correlation between the points, up to and including the lag unit. In ACF, the correlation coefficient is in the x-axis whereas the number of lags is shown in the y-axis.

After plotting the ACF plot we move to Partial Autocorrelation Function plots (PACF). A partial autocorrelation is a summary of the relationship between an observation in a time series with observations at prior time steps with the relationships of intervening observations removed.

In [287]:
```
fig, ax = plt.subplots(1,2,figsize=(15,5))
plot_acf(series, ax=ax[0])
plot_pacf(series, ax=ax[1])
plt.show()
```



One lag can be found above the significance level and thus q = 1. The first lag is the only one vastly above the signicance level and so p = 1.

The autocorrelation function can tell the order of MA terms, q, needed to remove autocorrelation in the stationary series.

Weekly Sales: Train & Test Split

Let's validate how accurate our model is. I am going to use the test train validation split to achieve this

In [288]:
```
series=ds_weekly.Total_Sales_Amount
split_time = 45
time=np.arange(len(ds_weekly))
xtrain=series[:split_time]
xtest=series[split_time:]
timeTrain = time[:split_time]
timeTest = time[split_time:]
```

Weekly Sales: ARIMA Model

Fitting the ARIMA model using above optimal combination of p, d, q

In [289]:
```
model = ARIMA(xtrain, order=(1,1,0))
model_fit = model.fit()
print(model_fit.summary())
```

```
                              ARIMA Model Results
======================================================================
============
Dep. Variable:     D.Total_Sales_Amount   No. Observations:
44
Model:                   ARIMA(1, 1, 0)   Log Likelihood
-543.940
Method:                         css-mle   S.D. of innovations
56476.313
Date:                  Wed, 06 Jan 2021   AIC
1093.881
Time:                          02:22:39   BIC
1099.233
Sample:                               1   HQIC
1095.866

======================================================================
==========================
                                 coef    std err          z      P>|
z|      [0.025      0.975]
----------------------------------------------------------------------
--------------------------
const                        653.5832   6212.436      0.105      0.9
17   -1.15e+04    1.28e+04
ar.L1.D.Total_Sales_Amount    -0.3793      0.153     -2.478      0.0
17      -0.679      -0.079
                                    Roots
======================================================================
=========
                  Real          Imaginary           Modulus
Frequency
----------------------------------------------------------------------
---------
AR.1           -2.6364           +0.0000j            2.6364
0.5000
----------------------------------------------------------------------
---------
```

Weekly Sales: Weekly Sales Trend and Forecast

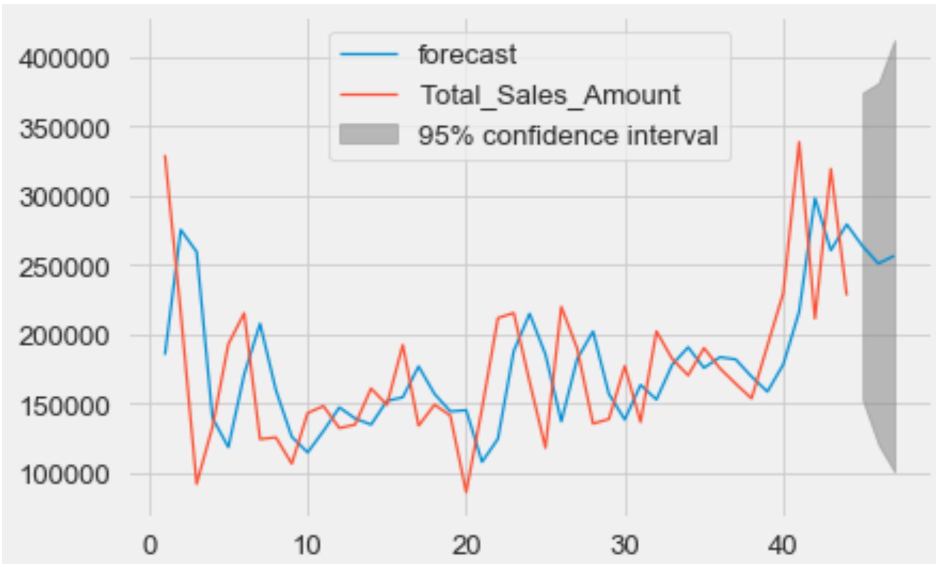
Now we have all we need to fit and plot the model

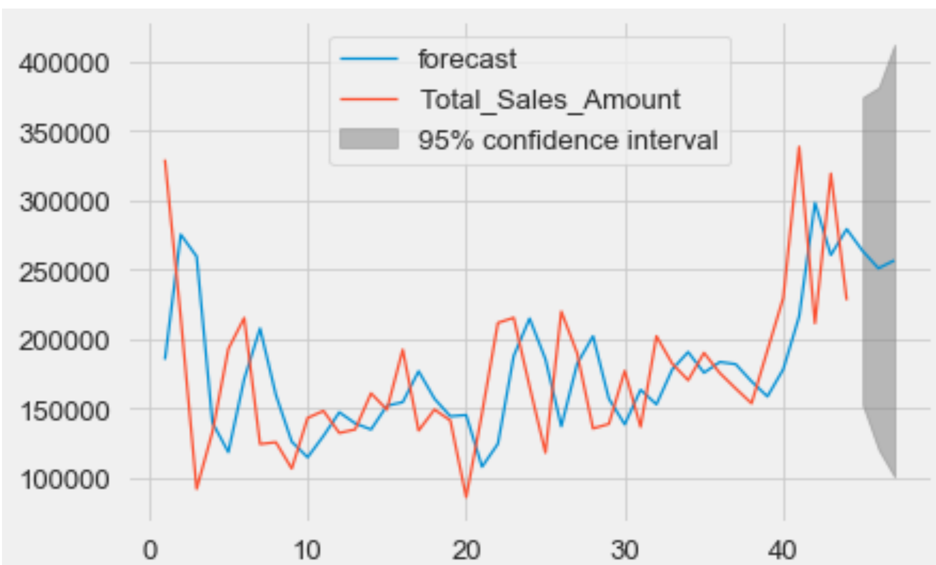
In [294]:

```
ytrain_pred = model_fit.predict()
ytest_pred = model_fit.predict(start=min(timeTest),end=max(timeTest),d
ynamic=True)
print('RMSE Train :',np.sqrt(np.mean((ytrain_pred - xtrain)**2)))
print('RMSE Test :',np.sqrt(np.mean((ytest_pred - xtest)**2)))
forecast = model_fit.forecast(20, alpha=0.05)
print('Weekly Sales Trend and Forecast')
model_fit.plot_predict(1,47)
```

```
RMSE Train : 182578.3533437104
RMSE Test : 346578.0272511122
Weekly Sales Trend and Forecast
```

Out[294]:





In [77]:
```
model_fit.forecast(30)[0]
```

Out[77]:
```
array([263473.39251312, 250933.71479928, 256591.56553247, 255347.005
89351,
       256720.56382389, 257101.05712361, 257858.22463086, 258472.517
79314,
       259141.00388407, 259788.93433559, 260444.66164035, 261097.431
56105,
       261751.32323179, 262404.78941733, 263058.41699146, 263711.983
35012,
       264365.5729281 , 265019.15369888, 265672.73781028, 266326.320
65456,
       266979.90397947, 267633.48712207, 268287.07033382, 268940.653
51935,
       269594.23671482, 270247.81990652, 270901.40309965, 271554.986
29223,
       272208.56948503, 272862.15267774])
```

By the graph you can see an upward trend in sales and high peaks sales before holidays. My forecast is closely align with the sales trend which is good, showing an overall increase trend after a small dip, but the model can still can be improved to align more closely with the sales trend. After all we can see it upward trend forecast for upcoming weeks which can be a good indication that the company is healthy and doing well and shouldn't be worried about the small dip.

Daily Sales

Now let's build a ARIMA model and forecast our Daily Sales. I will be doing the same steps as I did for the Weekly Sales model, lets see if our results would be any different.

As usual he first step in the model-building process is to plot the series and look for any evidence that the mean or variance is not stationary.

In [295]:
```
ds_daily = df.groupby(by=['Date'])['Total_Sales_Amount'].sum().reset_i
ndex()
```

In [296]:
```
ds_daily.head()
```

Out[296]:

	Date	Total_Sales_Amount
0	2010-12-01	58776.79
1	2010-12-02	47629.42
2	2010-12-03	46898.63
3	2010-12-05	31364.63
4	2010-12-06	54624.15

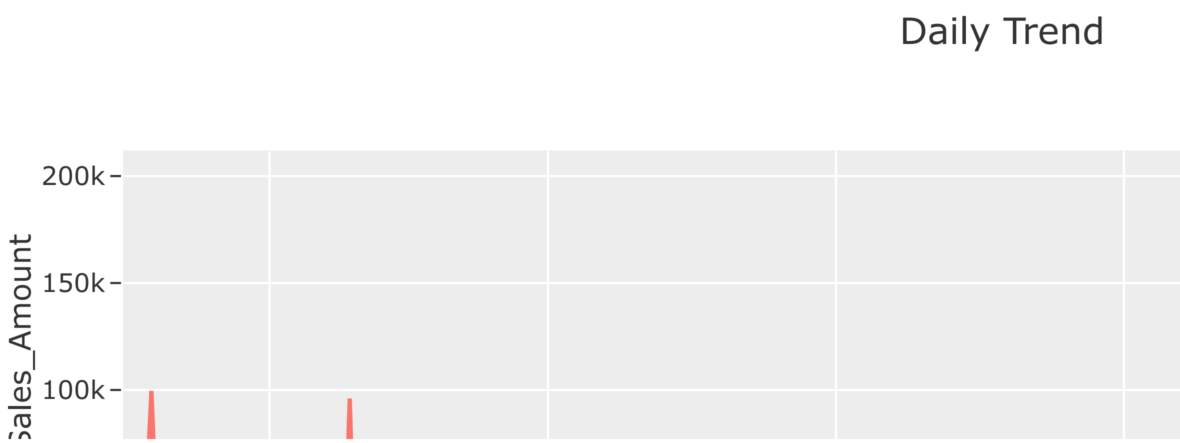
In [297]:
```
roll_mean = ds_daily.rolling(window=8, center=False).mean()
roll_std = ds_daily.rolling(window=8, center=False).std()
```

In [298]:
```
#fig = plt.figure(figsize=(12,7))
#plt.plot(ds_daily, color='blue', label='Original')
#plt.plot(roll_mean, color='red', label='Rolling Mean')
#plt.plot(roll_std, color='black', label = 'Rolling Std')
#plt.legend(loc='best')
#plt.title('Rolling Mean & Standard Deviation')
#plt.show
```

In [ ]:
```
```

Daily Sales: Daily Trend

In [299]:
```
fig = go.Figure(data=[go.Scatter(x=ds_daily.Date,y=ds_daily.Total_Sale
s_Amount)])
fig.update_layout(xaxis_title="Date",yaxis_title="Total_Sales_Amount",
title='Daily Trend',height=400,template='ggplot2')
fig.show()
```



Based on the graph we can see a slightly upward trend in our daily sales and because it is upward trend it is not stationarity.

Daily Sales: Test of Stationarity of Actual Series

To make sure let's do the stationarity test

In [300]:
```
print('\n*************************Daily*****************************
****')
output = adfuller(ds_daily.Total_Sales_Amount)
print('ADF Statistic: {0:.2f} and P value:{1:.5f}'.format(*output))
print("As we can see the p value is extreemly high which indicates tha
t we are fail to reject null hypothesis " \
      "and can conclude that series is not stationary ")
```

```
*************************Daily********************************
ADF Statistic: 0.03 and P value:0.96093
As we can see the p value is extreemly high which indicates that we
are fail to reject null hypothesis and can conclude that series is n
ot stationary
```

Daily Sales: Test of Stationarity with 1 differencing of series

As I mentioned before to proceed with our time series analysis, we need to stationarize the dataset. There are many approaches to stationarize data, but we'll use differencing.

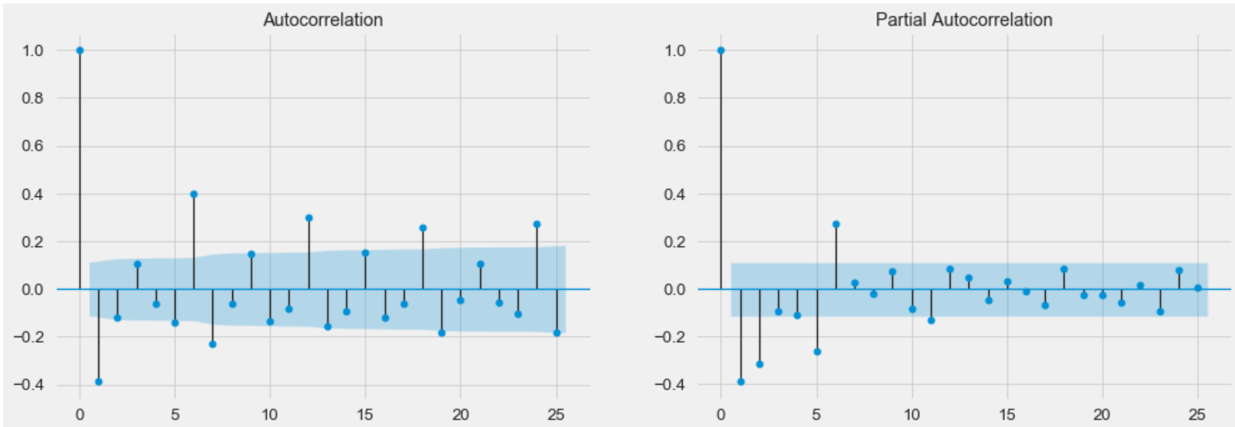
In [301]:
```
print('\n*************************Daily*****************************
****')
series_date = ds_daily.Total_Sales_Amount.diff(d)
series_date = series_date.dropna()
output = adfuller(series_date)
print('ADF Statistic: {0:.2f} and P value:{1:.5f}'.format(*output))
print("As we can see the p value is close to zero which is less than .
05 hence we reject null hypothesis")
```

```
*************************Daily********************************
ADF Statistic: -6.38 and P value:0.00000
As we can see the p value is close to zero which is less than .05 he
nce we reject null hypothesis
```

Daily Sales: ACF & PACF

Let's do a visual inspection of ACF and PACF

In [302]:
```
fig, ax = plt.subplots(1,2,figsize=(15,5))
plot_acf(series_date, ax=ax[0])
plot_pacf(series_date, ax=ax[1])
plt.show()
```



Daily Sales: Train & Test Split

Again let's validate how accurate our model is, we are going to use the test train validation split to achieve this

In [303]:
```
series_date=ds_daily.Total_Sales_Amount
split_time = 244
time_d=np.arange(len(ds_daily))
xtrain_d=series_date[:split_time]
xtest_d=series_date[split_time:]
timeTrain_d = time_d[:split_time]
timeTest_d = time_d[split_time:]
```

Daily Sales: ARIMA Model

Fitting the ARIMA model using above optimal combination of p, d, q

In [304]:
```
s_model = ARIMA(endog=xtrain_d , order=(1, 1, 0))
s_model_fit=s_model.fit()
print(s_model_fit.summary())
```

```
                              ARIMA Model Results
==============================================================================
============
Dep. Variable:     D.Total_Sales_Amount   No. Observations:
243
Model:                   ARIMA(1, 1, 0)   Log Likelihood
-2734.702
Method:                         css-mle   S.D. of innovations
18669.617
Date:                  Wed, 06 Jan 2021   AIC
5475.405
Time:                          02:25:11   BIC
5485.884
Sample:                               1   HQIC
5479.626

==============================================================================
==========================
                                 coef    std err          z      P>|
z|      [0.025      0.975]
------------------------------------------------------------------------------
--------------------------
const                        -51.1664    866.371     -0.059      0.9
53   -1749.223    1646.890
ar.L1.D.Total_Sales_Amount    -0.3840      0.059     -6.500      0.0
00      -0.500      -0.268
                                    Roots
==============================================================================
=========
                  Real          Imaginary           Modulus
Frequency
------------------------------------------------------------------------------
---------
AR.1            -2.6044           +0.0000j            2.6044
0.5000
------------------------------------------------------------------------------
---------
```

Daily Sales: Daily Sales Trend & Forecast

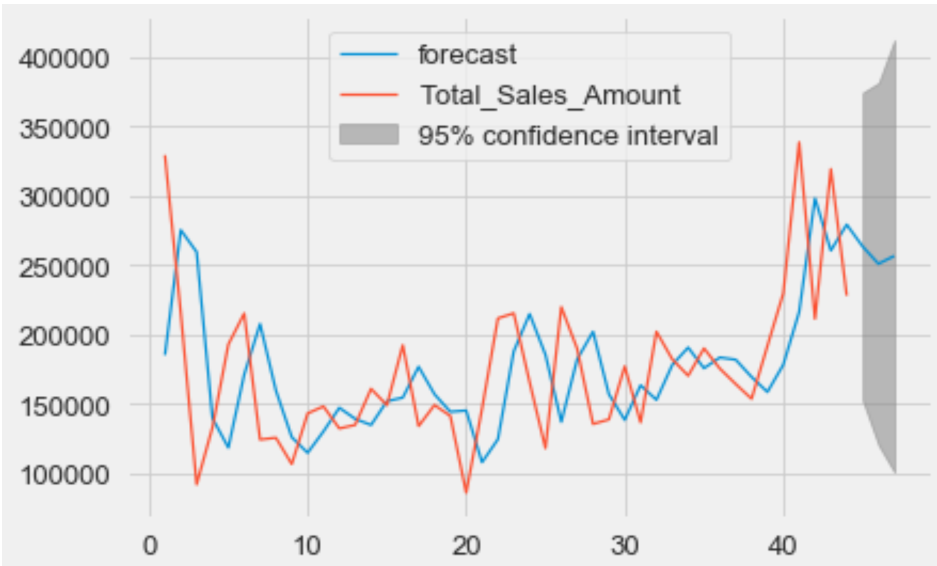
Lastly let's fit and plot the model

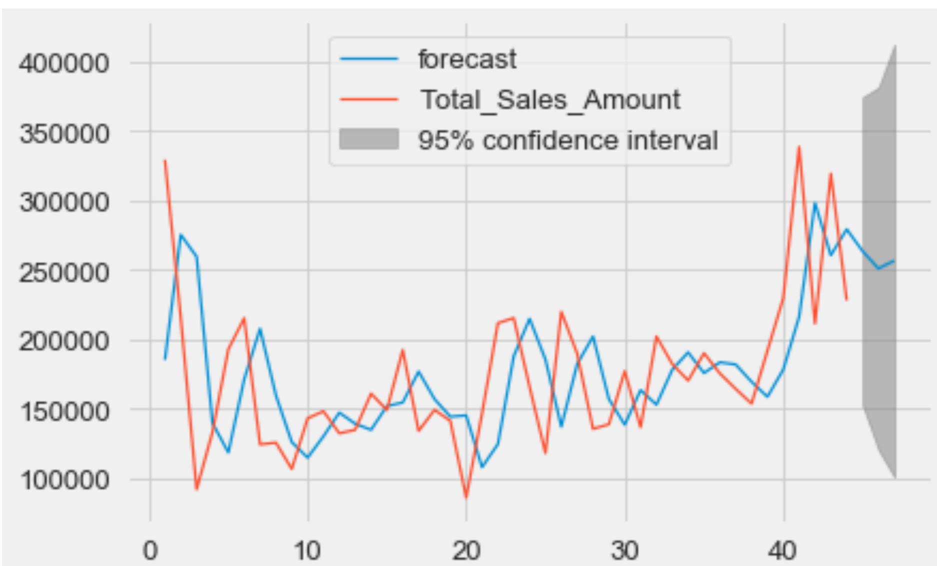
In [307]:

```
ytrain_pred = s_model_fit.predict()
ytest_pred = s_model_fit.predict(start=min(timeTest_d),end=max(timeTes
t_d),dynamic=True)

print('RMSE Train :',np.sqrt(np.mean((ytrain_pred - xtrain_d)**2)))
print('RMSE Test :',np.sqrt(np.mean((ytest_pred - xtest_d)**2)))
forecast = s_model_fit.forecast(20, alpha=0.05)
model_fit.plot_predict(1,47)
```

```
RMSE Train : 35106.516413391924
RMSE Test : 60999.69388072438
```

Out[307]:





In [89]:
```
model_fit.forecast(30)[0]
```

Out[89]:
```
array([263473.39251312, 250933.71479928, 256591.56553247, 255347.005
89351,
       256720.56382389, 257101.05712361, 257858.22463086, 258472.517
79314,
       259141.00388407, 259788.93433559, 260444.66164035, 261097.431
56105,
       261751.32323179, 262404.78941733, 263058.41699146, 263711.983
35012,
       264365.5729281 , 265019.15369888, 265672.73781028, 266326.320
65456,
       266979.90397947, 267633.48712207, 268287.07033382, 268940.653
51935,
       269594.23671482, 270247.81990652, 270901.40309965, 271554.986
29223,
       272208.56948503, 272862.15267774])
```

Based on the graph, the forecast is not completely align with the sales trend which can be improve in our feature analysis. Also, my RSME training is lower than the test RMSE indicating that our model is possibly under fitting and more data is needed to makel model better. There is an upward trend in the daily sales which indicates the company is healthy.

In [ ]:

Products Models

Now we are going to do a time series analysis and forecast weekly and daily for the top 10 products in our dataset.

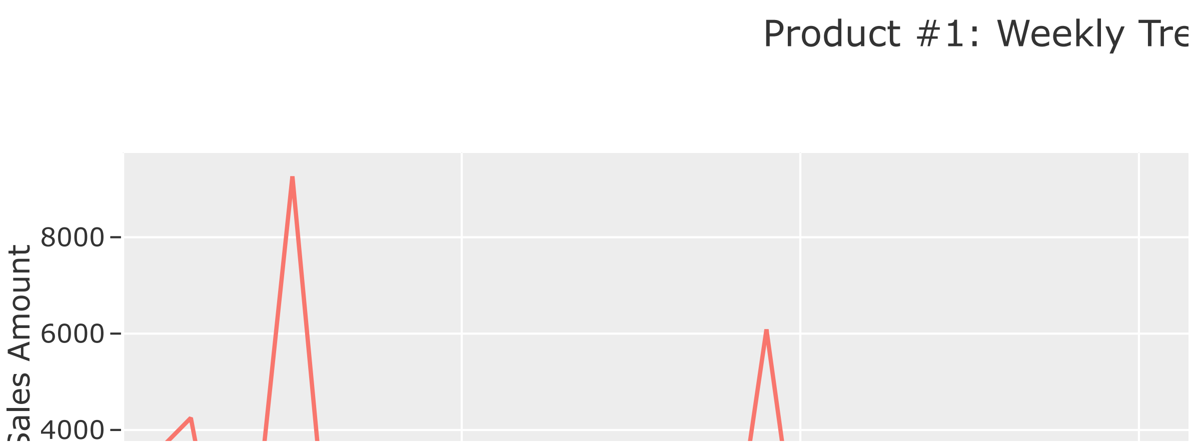
Again, I will be using ARIMA statistical model for my time series analysis, the steps are going to be similar to my previous models. Please check the previous models description for any clarifications or questions. Also, I will be removing the 'Test of Stationarity' step because usually product sales are not stationary due to the fact of inconstancy in sales. It has a variable variance and a mean that does not remain near, or returns to a long-run mean over time.

Product #1: 'White Hanging Heart T-Light Holder'

In [308]:
```
ds_weekly_P1 = dfp1.groupby(by=['Year','Week'])['Total_Sales_Amount'].
sum().reset_index()
```

Product #1: Total Sales Weekly Trend

In [309]:
```
fig = go.Figure(data=[go.Scatter(x=ds_weekly_P1.index,y=ds_weekly_P1.T
otal_Sales_Amount)])
fig.update_layout(xaxis_title="Week",yaxis_title="Total Sales Amount",
title='Product #1: Weekly Trend',height=400,template='ggplot2')
fig.show()
```



It is clear that the level of the series is not stationary. Some degree of differencing will be necessary to stabilize the series level

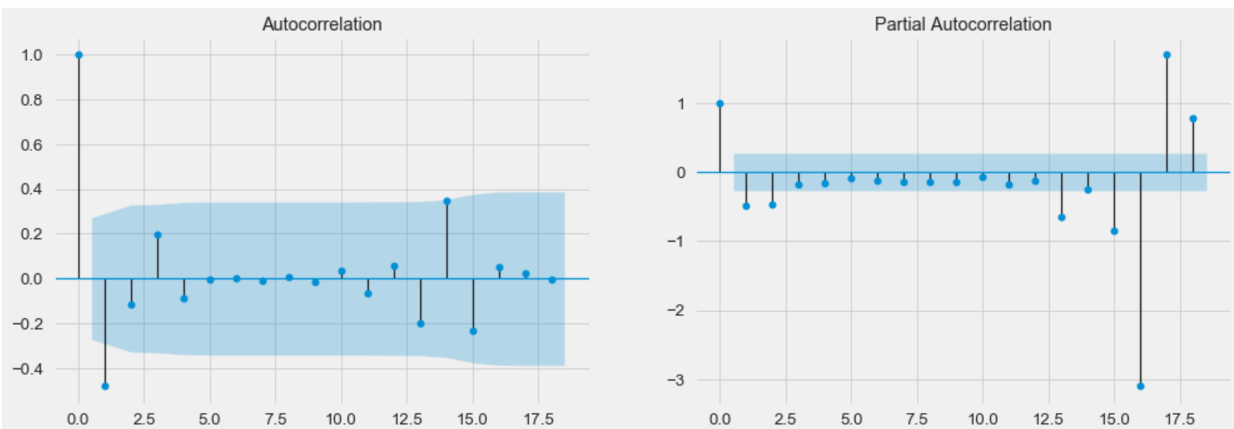
Product #1: Test of Stationarity with 1 differencing of series

In [310]:
```
d=1
print('**************************Week**********************************')
series_1 = ds_weekly_P1.Total_Sales_Amount.diff(d)
series_1 = series_1.dropna()
output = adfuller(series_1)

print('ADF Statistic: {0:.2f} and P value:{1:.5f}'.format(*output))
```

```
**************************Week*********************************
ADF Statistic: -9.67 and P value:0.00000
```

Product #1: PACF & ACF

In [311]:
```
fig, ax = plt.subplots(1,2,figsize=(15,5))
plot_acf(series_1, ax=ax[0])
plot_pacf(series_1, ax=ax[1])
plt.show()
```



Product #1: Train & Test Split

In [312]:
```
series_1=ds_weekly_P1.Total_Sales_Amount
split_time = 42
time=np.arange(len(ds_weekly_P1))
xtrain=series_1[:split_time]
xtest=series_1[split_time:]
timeTrain = time[:split_time]
timeTest = time[split_time:]
```

Product #1: ARIMA Model

In [313]:
```
model = ARIMA(xtrain, order=(1,1,0))
model_fit = model.fit()
print(model_fit.summary())
```

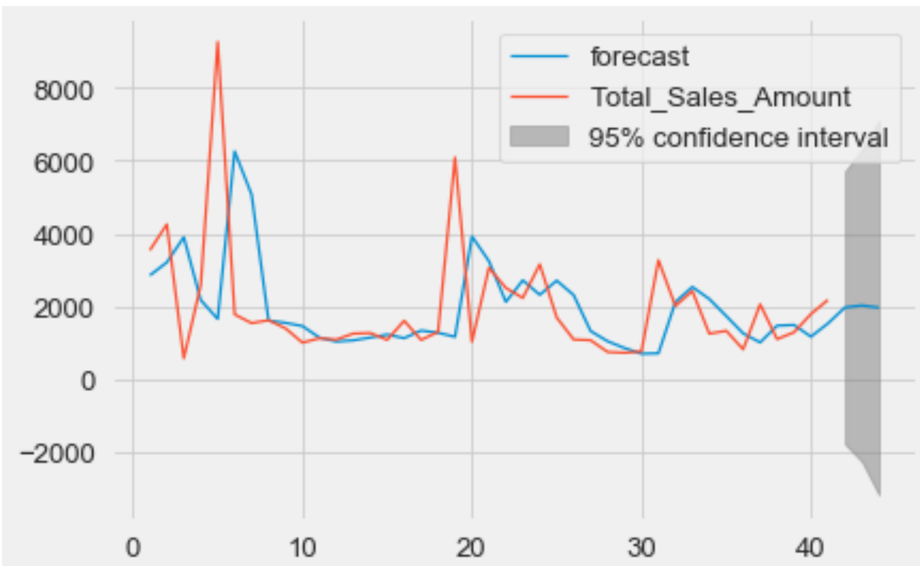
```
                              ARIMA Model Results
==============================================================================
============
Dep. Variable:     D.Total_Sales_Amount   No. Observations:
41
Model:                   ARIMA(1, 1, 0)   Log Likelihood
-368.122
Method:                         css-mle   S.D. of innovations
1914.027
Date:                  Wed, 06 Jan 2021   AIC
742.244
Time:                          02:25:57   BIC
747.385
Sample:                               1   HQIC
744.116

==============================================================================
==========================
                                coef    std err          z      P>|
z|      [0.025      0.975]
------------------------------------------------------------------------------
--------------------------
const                       -25.2606    208.499     -0.121      0.9
04    -433.911     383.389
ar.L1.D.Total_Sales_Amount   -0.4446      0.137     -3.247      0.0
02      -0.713      -0.176
                                    Roots
==============================================================================
=========
                  Real          Imaginary           Modulus
Frequency
------------------------------------------------------------------------------
---------
AR.1           -2.2491           +0.0000j            2.2491
0.5000
------------------------------------------------------------------------------
---------
```

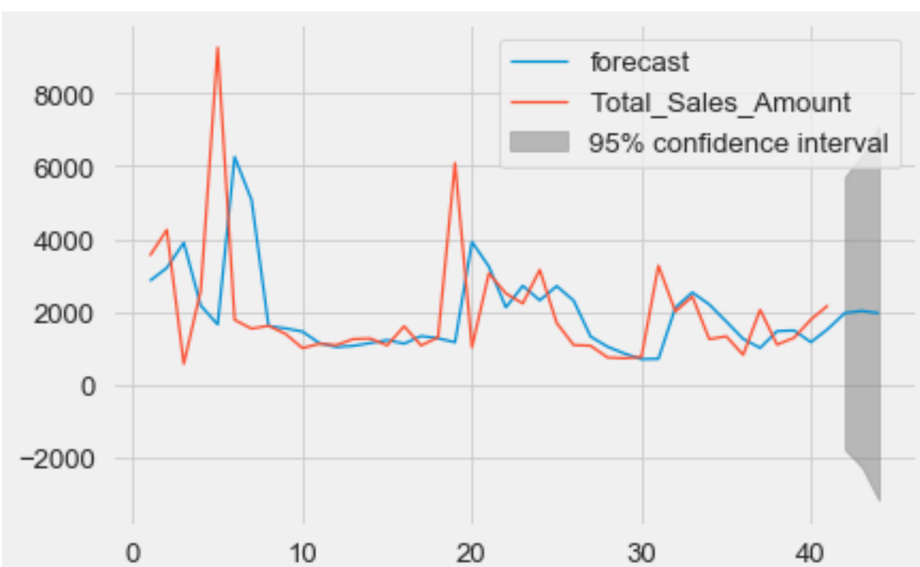
Product #1: Weekly Sales Trend and Forecast

In [317]:
```
ytrain_pred = model_fit.predict()
ytest_pred = model_fit.predict(start=min(timeTest),end=max(timeTest),d
ynamic=True)
print('RMSE Train :',np.sqrt(np.mean((ytrain_pred - xtrain)**2)))
print('RMSE Test :',np.sqrt(np.mean((ytest_pred - xtest)**2)))

forecast = model_fit.forecast(20, alpha=0.05)
model_fit.plot_predict(1,44)
```

```
RMSE Train : 2727.898344042451
RMSE Test : 2468.4484704964975
```

Out[317]:





In [97]:
```
model_fit.forecast(30)[0]
```

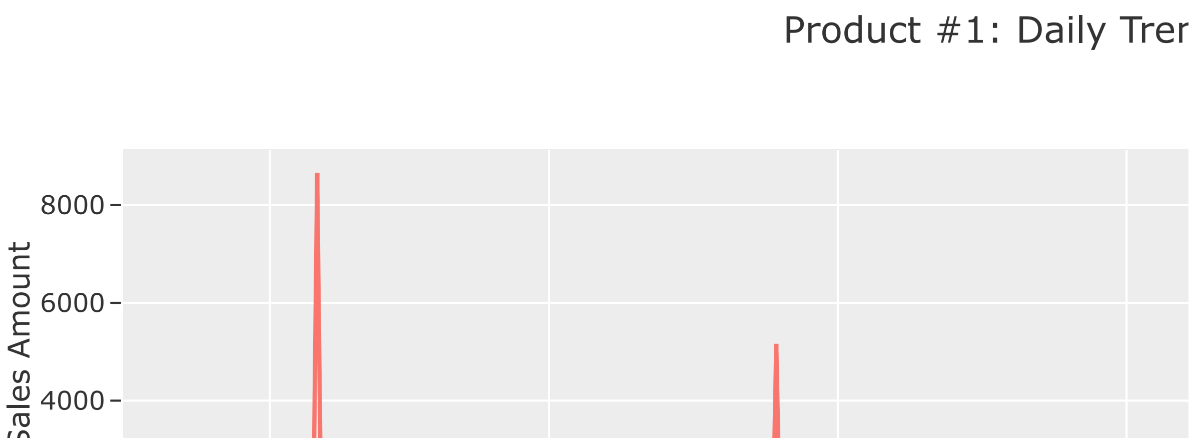
Out[97]:
```
array([1966.55314582, 2022.85643603, 1961.33080989, 1952.19432365,
       1919.76455708, 1897.69145812, 1871.01357347, 1846.38307048,
       1820.84225983, 1795.70619054, 1770.39016498, 1745.15415165,
       1719.88256325, 1694.62679225, 1669.36398852, 1644.10431168,
       1618.84324456, 1593.58279559, 1568.32207177, 1543.06147016,
       1517.80081422, 1492.54018243, 1467.2795399 , 1442.01890214,
       1416.75826227, 1391.49762333, 1366.23698398, 1340.97634481,
       1315.71570557, 1290.45506635])
```

As we can see the product has high peak sales close to holidays seasons, but as we get closer to in end of the month there is a downward trend forecast close to October signaling demand is low around those times

Product #1 Daily: Daily Trend

In [98]:
```
ds_daily_P1 = dfp1.groupby(by=['Date'])['Total_Sales_Amount'].sum().re
set_index()
```

In [99]:
```
fig = go.Figure(data=[go.Scatter(x=ds_daily_P1.Date,y=ds_daily_P1.Tota
l_Sales_Amount)])
fig.update_layout(xaxis_title="Date",yaxis_title="Total Sales Amount",
title='Product #1: Daily Trend',height=400,template='ggplot2')
fig.show()
```



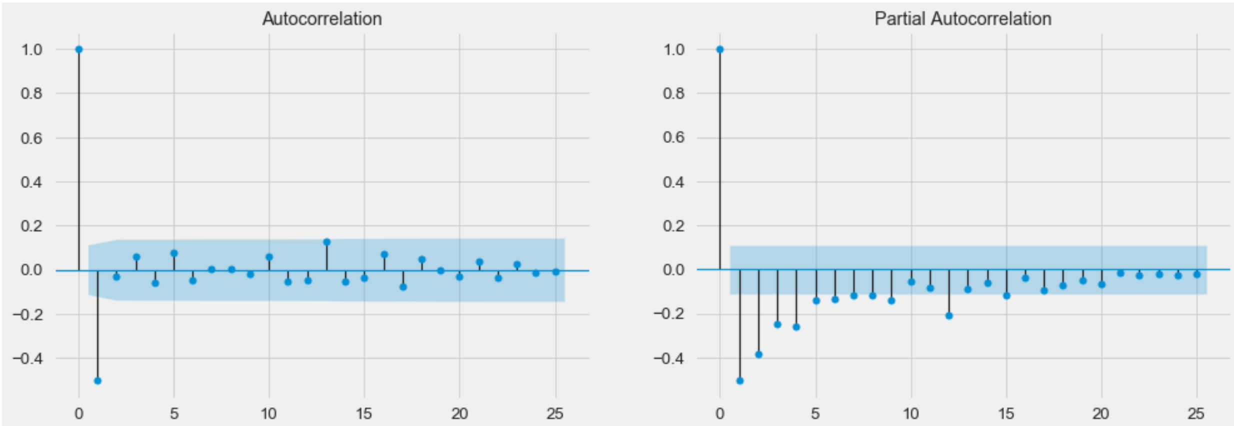
Product #1 Daily: Test of Stationarity with 1 differencing of series

In [100]:
```
print('\n***************************Daily*****************************
****')
series_date_1 = ds_daily_P1.Total_Sales_Amount.diff(d)
series_date_1 = series_date_1.dropna()
output = adfuller(series_date_1)
print('ADF Statistic: {0:.2f} and P value:{1:.5f}'.format(*output))
```

```
***************************Daily*********************************
ADF Statistic: -9.22 and P value:0.00000
```

Product #1 Daily: PACF & ACF

In [101]:
```
fig, ax = plt.subplots(1,2,figsize=(15,5))
plot_acf(series_date_1, ax=ax[0])
plot_pacf(series_date_1, ax=ax[1])
plt.show()
```



Product #1 Daily: Train & Test Split

In [102]:
```
series_date_1=ds_daily_P1.Total_Sales_Amount
split_time = 250
time_d=np.arange(len(ds_daily_P1))
xtrain_d=series_date_1[:split_time]
xtest_d=series_date_1[split_time:]
timeTrain_d = time_d[:split_time]
timeTest_d = time_d[split_time:]
```

Product #1 Daily: ARIMA Model

In [103]:
```
s_model = ARIMA(endog=xtrain_d , order=(1, 1, 0))
s_model_fit=s_model.fit()
print(s_model_fit.summary())
```

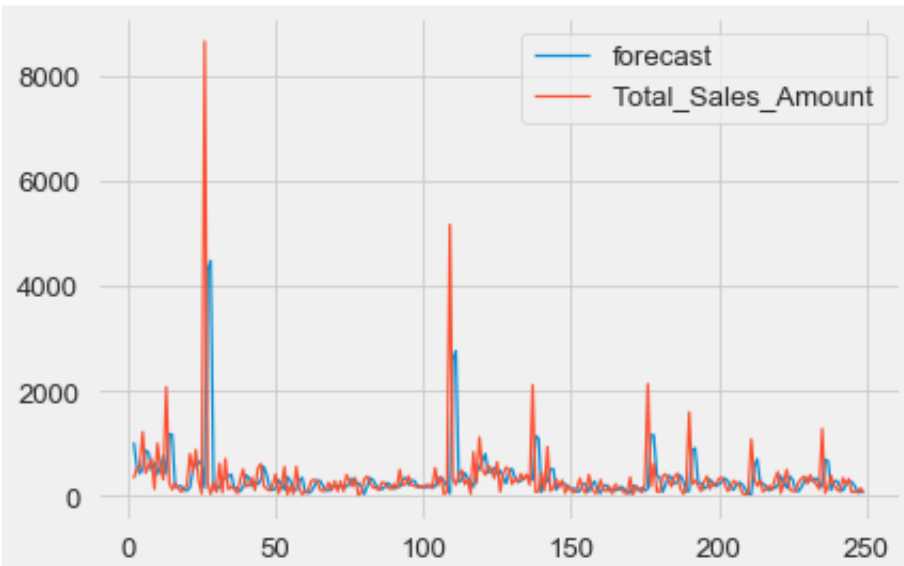
```
                              ARIMA Model Results
==============================================================================
============
Dep. Variable:     D.Total_Sales_Amount   No. Observations:
249
Model:                   ARIMA(1, 1, 0)   Log Likelihood
-2032.436
Method:                         css-mle   S.D. of innovations
847.989
Date:                  Wed, 06 Jan 2021   AIC
4070.872
Time:                          00:02:38   BIC
4081.425
Sample:                               1   HQIC
4075.120

==============================================================================
==========================
                                 coef    std err          z      P>|
z|      [0.025      0.975]
------------------------------------------------------------------------------
--------------------------
const                         -4.0456     35.778     -0.113      0.9
10     -74.170      66.079
ar.L1.D.Total_Sales_Amount    -0.5040      0.055     -9.247      0.0
00      -0.611      -0.397
                                    Roots
==============================================================================
=========
                  Real          Imaginary           Modulus
Frequency
------------------------------------------------------------------------------
---------
AR.1           -1.9840           +0.0000j            1.9840
0.5000
------------------------------------------------------------------------------
---------
```

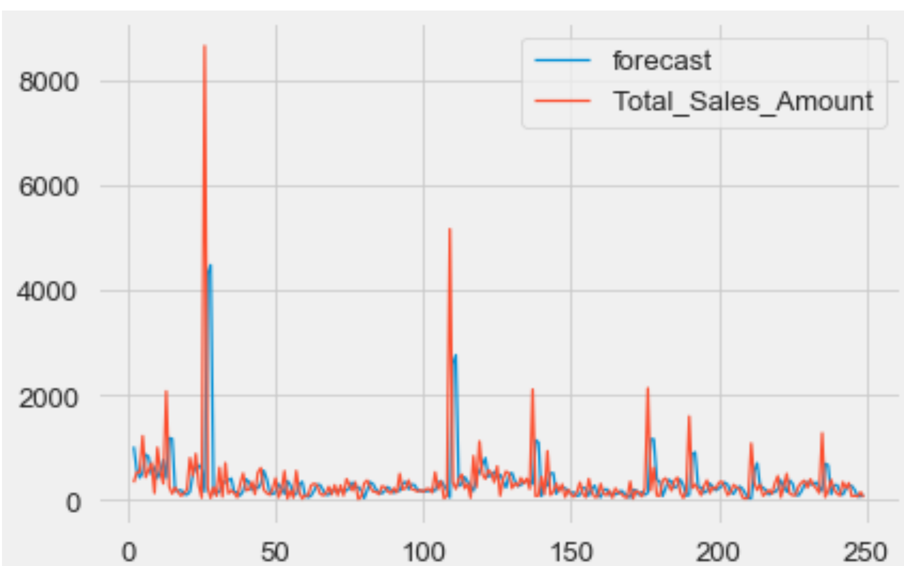
Product #1 Daily: Daily Sales Trend and Forecast

In [104]:
```
ytrain_pred = s_model_fit.predict()
ytest_pred = s_model_fit.predict(start=min(timeTest_d),end=max(timeTes
t_d),dynamic=True)

print('RMSE Train :',np.sqrt(np.mean((ytrain_pred - xtrain_d)**2)))
print('RMSE Test :',np.sqrt(np.mean((ytest_pred - xtest_d)**2)))
forecast = s_model_fit.forecast(20, alpha=0.05)
print('White Hanging Heart T-Light Holder Daily Trend and Forecast')
s_model_fit.plot_predict()
```

```
RMSE Train : 906.9327276856891
RMSE Test : 689.0375819990196
White Hanging Heart T-Light Holder Daily Trend and Forecast
```

Out[104]:





In [105]:
```
s_model_fit.forecast(30)[0]
```

Out[105]:
```
array([ 85.71465972,  57.18843141,  65.48177581,  55.21708524,
        54.30613337,  48.68065069,  45.43141578,  40.98448937,
        37.14123104,  32.99370809,  28.99954252,  24.9280808 ,
        20.89557837,  16.84343943,  12.80119781,   8.75396768,
         4.7092519 ,   0.66326881,  -3.38207552,  -7.4277418 ,
       -11.47324581, -15.51883161, -19.56437618, -23.60994153,
       -27.65549641, -31.70105657, -35.74661407, -39.79217291,
       -43.83773107, -47.88328957])
```

Again as you can see my forecast aligns with the product trend very well, but there is downward trend towards October. The top 2 peaks sales can be related to valentines day and eastern or Ascension day in the beginning in May. This easily can tell the managers or owners to have enough inventory on those peak days but as not as much towards the end of the year since there downward forecast trend.

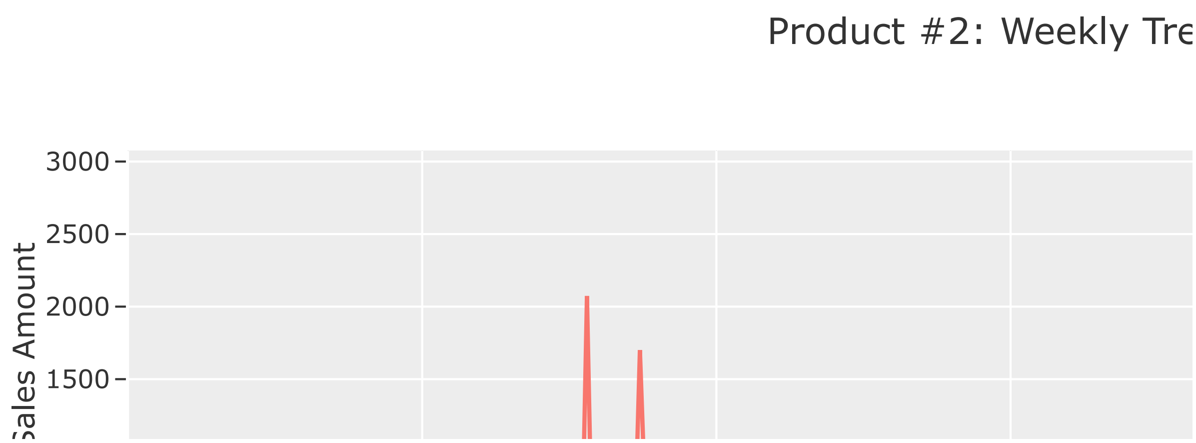
In [ ]:
```

```

Product #2: 'Jumbo Bag Red Retrospot'

In [318]:
```
ds_weekly_P2 = dfp2.groupby(by=['Date'])['Total_Sales_Amount'].sum().r
eset_index()
```

Product #2 Weekly: Weekly Trend

In [319]:
```
fig = go.Figure(data=[go.Scatter(x=ds_weekly_P2.index,y=ds_weekly_P2.Total_Sales_Amount)])
fig.update_layout(xaxis_title="Week",yaxis_title="Total Sales Amount",
title='Product #2: Weekly Trend',height=400,template='ggplot2')
fig.show()
```



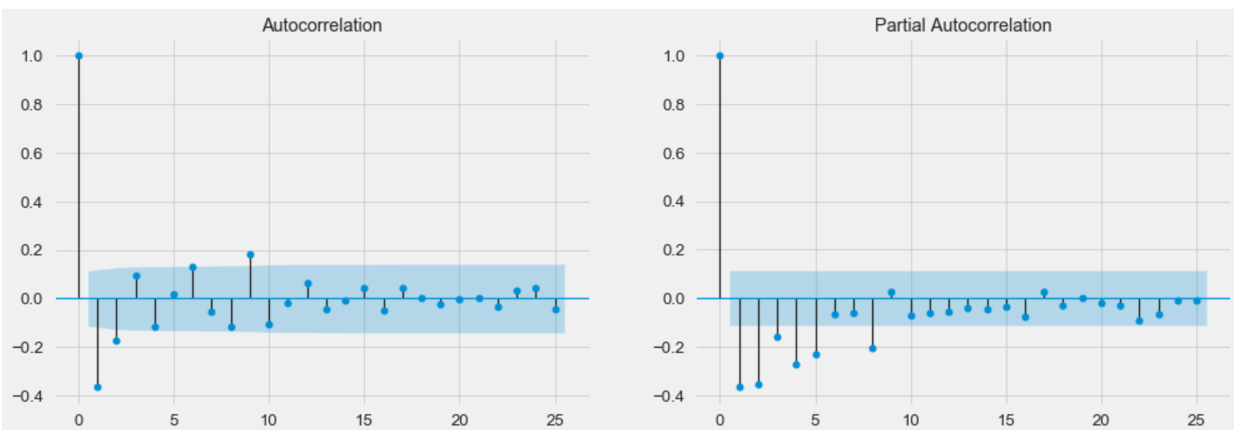
Product #2 Weekly: Test of Stationarity with 1 differencing of series

In [320]:
```
d=1
print('**************************Week*********************************')
series_2 = ds_weekly_P2.Total_Sales_Amount.diff(d)
series_2 = series_2.dropna()
output = adfuller(series_2)

print('ADF Statistic: {0:.2f} and P value:{1:.5f}'.format(*output))
print("As we can see the p value is close to zero which is less than .05 hence we reject null hypothesis")
```

```
**************************Week*********************************
ADF Statistic: -10.87 and P value:0.00000
As we can see the p value is close to zero which is less than .05 he
nce we reject null hypothesis
```

Product #2 Weekly: PACF & ACF

In [321]:
```
fig, ax = plt.subplots(1,2,figsize=(15,5))
plot_acf(series_2, ax=ax[0])
plot_pacf(series_2, ax=ax[1])
plt.show()
```



Product #2 Weekly: Train & Test Split

In [322]:
```
series_2=ds_weekly_P2.Total_Sales_Amount
split_time = 50
time=np.arange(len(ds_weekly))
xtrain=series_2[:split_time]
xtest=series_2[split_time:]
timeTrain = time[:split_time]
timeTest = time[split_time:]
```

Product #2 Weekly: ARIMA Model

In [323]:
```
model = ARIMA(xtrain, order=(1,1,0))
model_fit = model.fit()
print(model_fit.summary())
```

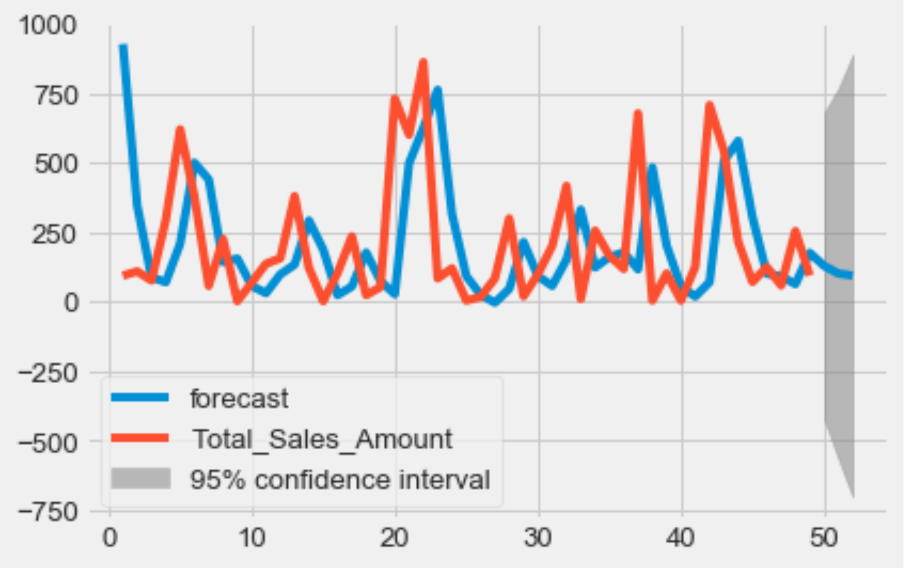
```
                              ARIMA Model Results
==============================================================================
============
Dep. Variable:     D.Total_Sales_Amount   No. Observations:
49
Model:                   ARIMA(1, 1, 0)   Log Likelihood
-345.965
Method:                         css-mle   S.D. of innovations
281.596
Date:                  Wed, 06 Jan 2021   AIC
697.929
Time:                          02:43:56   BIC
703.605
Sample:                               1   HQIC
700.083

==============================================================================
==========================
                                 coef    std err          z      P>|
z|      [0.025      0.975]
------------------------------------------------------------------------------
--------------------------
const                        -12.4033     30.709     -0.404      0.6
88     -72.593      47.786
ar.L1.D.Total_Sales_Amount    -0.3185      0.148     -2.159      0.0
36      -0.608      -0.029
                                    Roots
==============================================================================
=========
                  Real          Imaginary           Modulus
Frequency
------------------------------------------------------------------------------
---------
AR.1           -3.1396           +0.0000j            3.1396
0.5000
------------------------------------------------------------------------------
---------
```

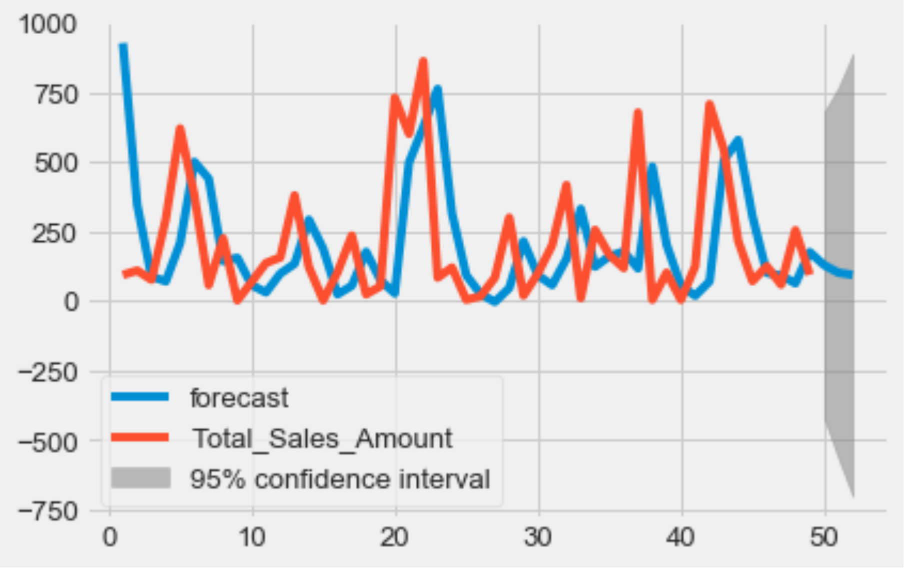
Product #2 Weekly: Weekly Trend and Forecast

In [328]:
```
ytrain_pred = model_fit.predict()
ytest_pred = model_fit.predict(start=min(timeTest),end=max(timeTest),d
ynamic=True)
print('RMSE Train :',np.sqrt(np.mean((ytrain_pred - xtrain)**2)))
print('RMSE Test :',np.sqrt(np.mean((ytest_pred - xtest)**2)))
forecast = model_fit.forecast(20, alpha=0.05)
model_fit.plot_predict(1,52)
```

```
RMSE Train : 341.3040033477939
RMSE Test : 420.1330541319082
```

Out[328]:





In [329]:
```
model_fit.forecast(30)[0]
```

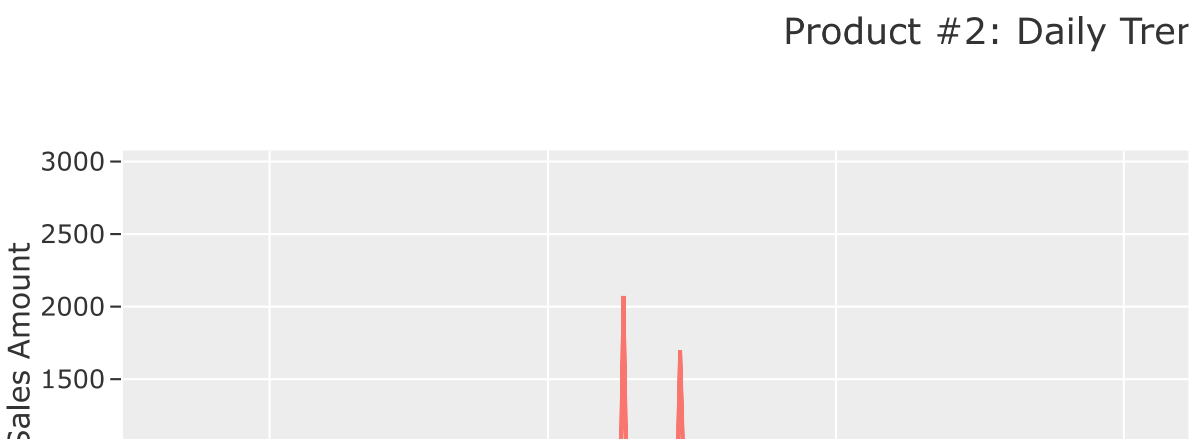
Out[329]:
```
array([ 128.7728193 ,  101.40046532,   93.76506546,   79.84311241,
         67.92353271,   55.36616328,   43.01194066,   30.59301234,
         18.19469389,    5.78981084,   -6.61298127,  -19.01643938,
        -31.41968536,  -43.82299891,  -56.22629094,  -68.62958982,
        -81.03288651,  -93.43618391, -105.83948108, -118.24277832,
       -130.64607554, -143.04937276, -155.45266999, -167.85596721,
       -180.25926444, -192.66256166, -205.06585889, -217.46915611,
       -229.87245333, -242.27575056])
```

The forecast shows a downward trend for this product close to October which not much inventory is required. My model forecast is closely aligned with sales trend which is good. During valentines and Eastern the product has an upward spikes when the demand is high.

Product #2 Daily: Daily Trend

In [330]:
```
ds_daily_P2 = dfp2.groupby(by=['Date'])['Total_Sales_Amount'].sum().re
set_index()
```

In [331]:
```
fig = go.Figure(data=[go.Scatter(x=ds_daily_P2.Date,y=ds_daily_P2.Tota
l_Sales_Amount)])
fig.update_layout(xaxis_title="Date",yaxis_title="Total Sales Amount",
title='Product #2: Daily Trend',height=400,template='ggplot2')
fig.show()
```



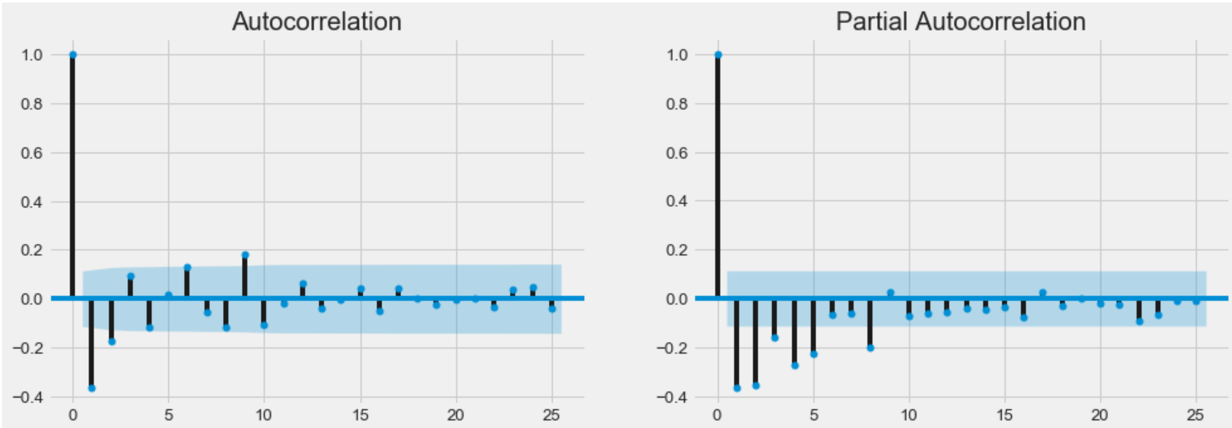
Product #2 Daily: Test of Stationarity with 1 differencing of series

In [332]:
```
print('\n**************************Daily******************************
****')
series_date_2 = ds_daily_P2.Total_Sales_Amount.diff(d)
series_date_2 = series_date_2.dropna()
output = adfuller(series_date_2)
print('ADF Statistic: {0:.2f} and P value:{1:.5f}'.format(*output))
```

```
**************************Daily*********************************
ADF Statistic: -10.87 and P value:0.00000
```

Product #2 Daily: PACF & ACF

In [333]:
```
fig, ax = plt.subplots(1,2,figsize=(15,5))
plot_acf(series_date_2, ax=ax[0])
plot_pacf(series_date_2, ax=ax[1])
plt.show()
```



Product #2 Daily: Train & Test Split

In [334]:
```
series_date_2=ds_daily_P2.Total_Sales_Amount
split_time = 245
time_d=np.arange(len(ds_daily_P2))
xtrain_d=series_date_2[:split_time]
xtest_d=series_date_2[split_time:]
timeTrain_d = time_d[:split_time]
timeTest_d = time_d[split_time:]
```

Product #2 Daily: ARIMA Model

In [335]:
```
s_model = ARIMA(endog=xtrain_d , order=(1, 1, 0))
s_model_fit=s_model.fit()
print(s_model_fit.summary())
```

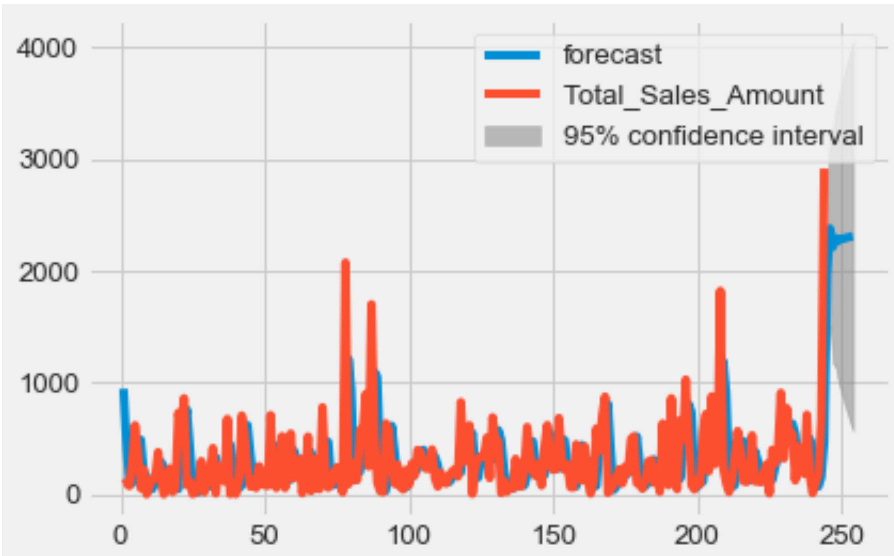
```
                                ARIMA Model Results
==============================================================================
============
Dep. Variable:     D.Total_Sales_Amount   No. Observations:
244
Model:                   ARIMA(1, 1, 0)   Log Likelihood
-1798.230
Method:                         css-mle   S.D. of innovations
383.926
Date:                  Wed, 06 Jan 2021   AIC
3602.459
Time:                          03:04:13   BIC
3612.951
Sample:                               1   HQIC
3606.685

==============================================================================
==========================
                                coef    std err          z      P>|
z|      [0.025      0.975]
------------------------------------------------------------------------------
--------------------------
const                         6.3895     17.287      0.370      0.7
12     -27.493      40.272
ar.L1.D.Total_Sales_Amount   -0.4236      0.063     -6.723      0.0
00      -0.547      -0.300
                                    Roots
==============================================================================
=========
                  Real          Imaginary           Modulus
Frequency
------------------------------------------------------------------------------
---------
AR.1           -2.3609           +0.0000j            2.3609
0.5000
------------------------------------------------------------------------------
---------
```

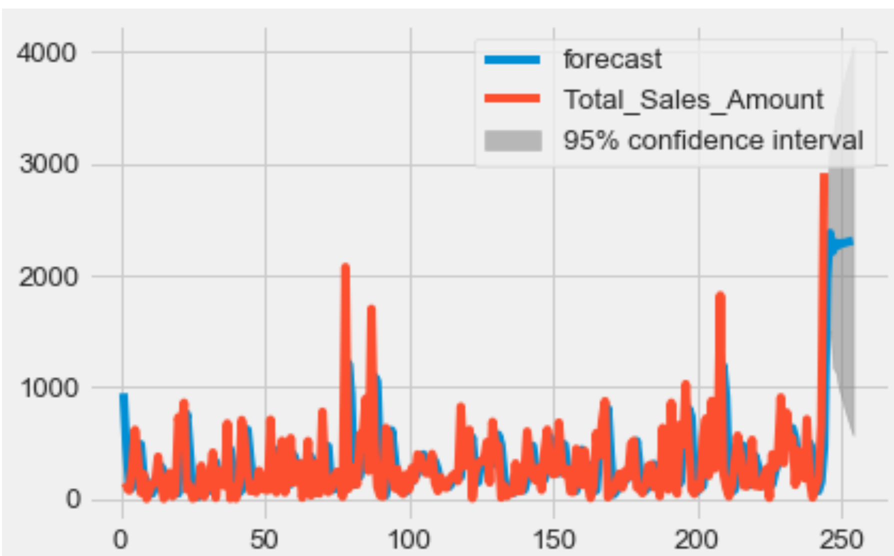
Product #2 Daily: Daily Trend and Forecast

In [340]:
```
ytrain_pred = s_model_fit.predict()
ytest_pred = s_model_fit.predict(start=min(timeTest_d),end=max(timeTes
t_d),dynamic=True)

print('RMSE Train :',np.sqrt(np.mean((ytrain_pred - xtrain_d)**2)))
print('RMSE Test :',np.sqrt(np.mean((ytest_pred - xtest_d)**2)))
forecast = s_model_fit.forecast(20, alpha=0.05)
s_model_fit.plot_predict(1,254)
```

```
RMSE Train : 482.696861284742
RMSE Test : 572.2031613279908
```

Out[340]:





In [337]:
```
s_model_fit.forecast(30)[0]
```

Out[337]:
```
array([1968.30306627, 2377.62185276, 2213.34387605, 2292.02252096,
       2267.79277485, 2287.15162029, 2288.0477797 , 2296.7641224 ,
       2302.16809405, 2308.97507654, 2315.18779004, 2321.65221623,
       2328.01002525, 2334.41299378, 2340.79683423, 2347.18877671,
       2353.57728743, 2359.96725173, 2366.35660035, 2372.74620975,
       2379.13570869, 2385.52525441, 2391.91478032, 2398.30431462,
       2404.69384537, 2411.08337762, 2417.47290924, 2423.86244112,
       2430.25197289, 2436.64150471])
```

My model forecast fits well with daily sales trend. Based on the my model forecast we can see an upward trend in December for this product so more inventory is required to keep up with demand and not losing the customers. This product demand spikes high close to March time and Christmas time.

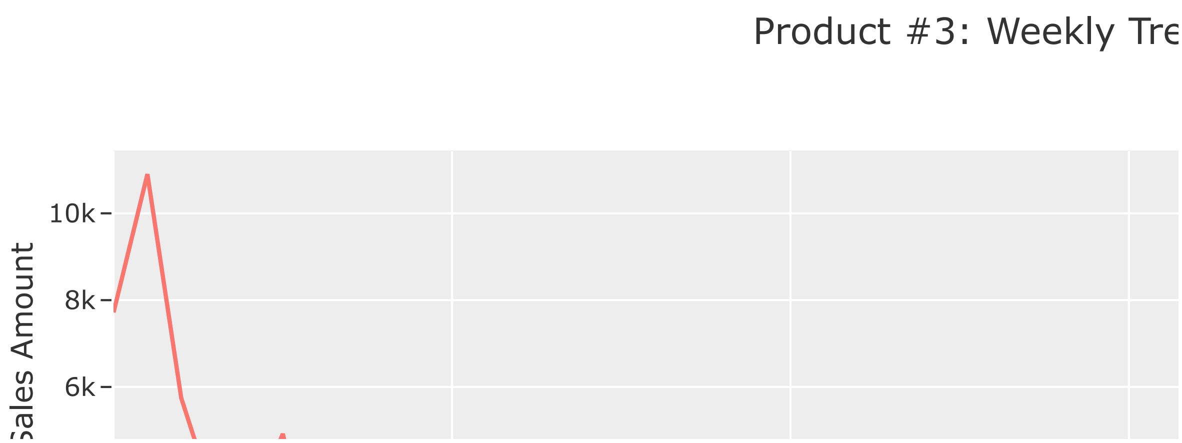
In [ ]:
```

```

Product #3: 'Regency Cakestand 3 Tier'

In [341]:
```
ds_weekly_P3 = dfp3.groupby(by=['Year','Week'])['Total_Sales_Amount'].
sum().reset_index()
```

Product #3 Weekly: Total Sales Weekly Trend

In [342]:
```
fig = go.Figure(data=[go.Scatter(x=ds_weekly_P3.index,y=ds_weekly_P3.T
otal_Sales_Amount)])
fig.update_layout(xaxis_title="Week",yaxis_title="Total Sales Amount",
title='Product #3: Weekly Trend',height=400,template='ggplot2')
fig.show()
```



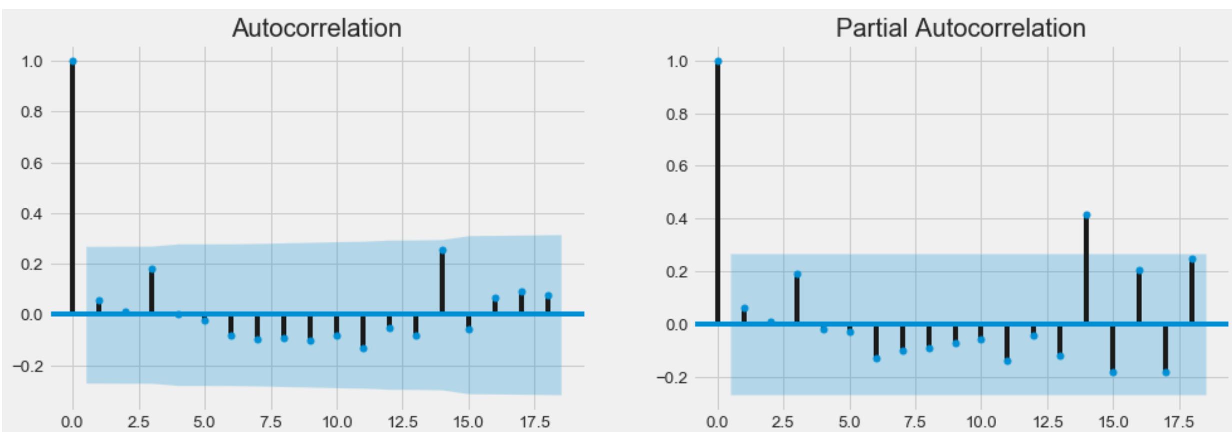
Product #3 Weekly: Test of Stationarity with 1 differencing of series

In [343]:
```
d=1
print('**************************Week*********************************
*')
series_3 = ds_weekly_P3.Total_Sales_Amount.diff(d)
series_3 = series_3.dropna()
output = adfuller(series_3)

print('ADF Statistic: {0:.2f} and P value:{1:.5f}'.format(*output))
print("As we can see the p value is close to zero which is less than .
05 hence we reject null hypothesis ")
```

```
**************************Week********************************
ADF Statistic: -11.38 and P value:0.00000
As we can see the p value is close to zero which is less than .05 he
nce we reject null hypothesis
```

Product #3 Weekly: PACF & ACF

In [344]:
```
fig, ax = plt.subplots(1,2,figsize=(15,5))
plot_acf(series_1, ax=ax[0])
plot_pacf(series_1, ax=ax[1])
plt.show()
```



Product #3 Weekly: Train & Test Split

In [345]:
```
series_3=ds_weekly_P3.Total_Sales_Amount
split_time = 45
time=np.arange(len(ds_weekly_P1))
xtrain=series_3[:split_time]
xtest=series_3[split_time:]
timeTrain = time[:split_time]
timeTest = time[split_time:]
```

Product #3 Weekly: ARIMA Model

In [346]:
```
model = ARIMA(xtrain, order=(1,1,0))
model_fit = model.fit()
print(model_fit.summary())
```

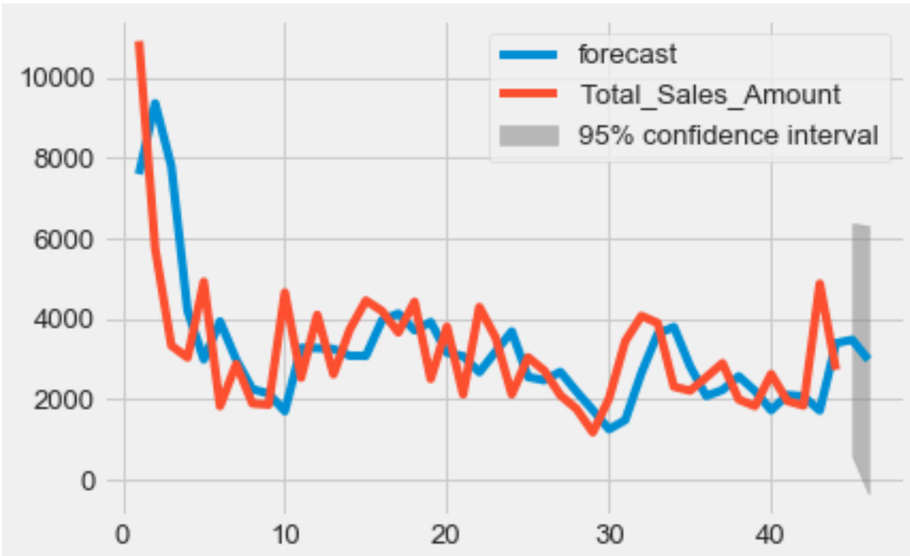
```
                              ARIMA Model Results
==============================================================================
============
Dep. Variable:     D.Total_Sales_Amount   No. Observations:
44
Model:                   ARIMA(1, 1, 0)   Log Likelihood
-383.705
Method:                         css-mle   S.D. of innovations
1479.269
Date:                  Wed, 06 Jan 2021   AIC
773.410
Time:                          03:27:02   BIC
778.763
Sample:                               1   HQIC
775.395

==============================================================================
==========================
                                 coef    std err          z      P>|
z|      [0.025      0.975]
------------------------------------------------------------------------------
--------------------------
const                       -122.0914    157.018     -0.778      0.4
41    -429.841     185.658
ar.L1.D.Total_Sales_Amount    -0.4302      0.144     -2.977      0.0
05      -0.713      -0.147
                                    Roots
==============================================================================
=========
                  Real          Imaginary           Modulus
Frequency
------------------------------------------------------------------------------
---------
AR.1           -2.3244           +0.0000j            2.3244
0.5000
------------------------------------------------------------------------------
---------
```

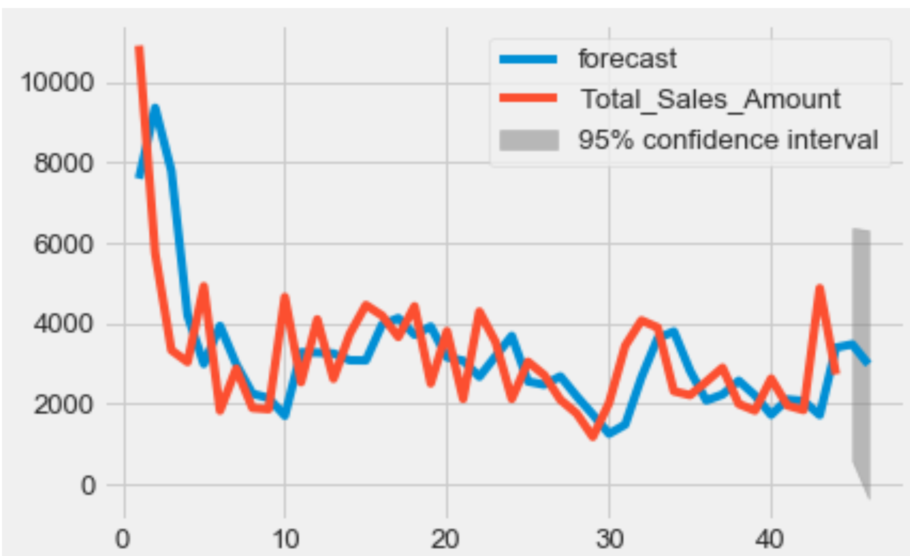
Product #3 Weekly: Weekly Trend and Forecast

In [348]:
```
ytrain_pred = model_fit.predict()
ytest_pred = model_fit.predict(start=min(timeTest),end=max(timeTest),d
ynamic=True)
print('RMSE Train :',np.sqrt(np.mean((ytrain_pred - xtrain)**2)))
print('RMSE Test :',np.sqrt(np.mean((ytest_pred - xtest)**2)))
forecast = model_fit.forecast(20, alpha=0.05)
model_fit.plot_predict(1,46)
```

```
RMSE Train : 3427.851159725249
RMSE Test : 3324.0270936553843
```

Out[348]:





In [129]:
```
model_fit.forecast(30)[0]
```

Out[129]:
```
array([3475.94970187, 2980.08109092, 3018.7931479 , 2827.52220267,
       2735.19269786, 2600.29740106, 2483.71441749, 2359.25325757,
       2238.18138341, 2115.65139795, 1993.74870942, 1871.57615025,
       1749.51969266, 1627.41328678, 1505.32836925, 1383.23420717,
       1261.14402221, 1139.05212624, 1016.96096637,  894.86948983,
        772.77814952,  650.6867506 ,  528.59537689,  406.50399234,
        284.41261245,  162.32123056,   40.22984953,  -81.86153187,
       -203.95291311, -326.04429442])
```

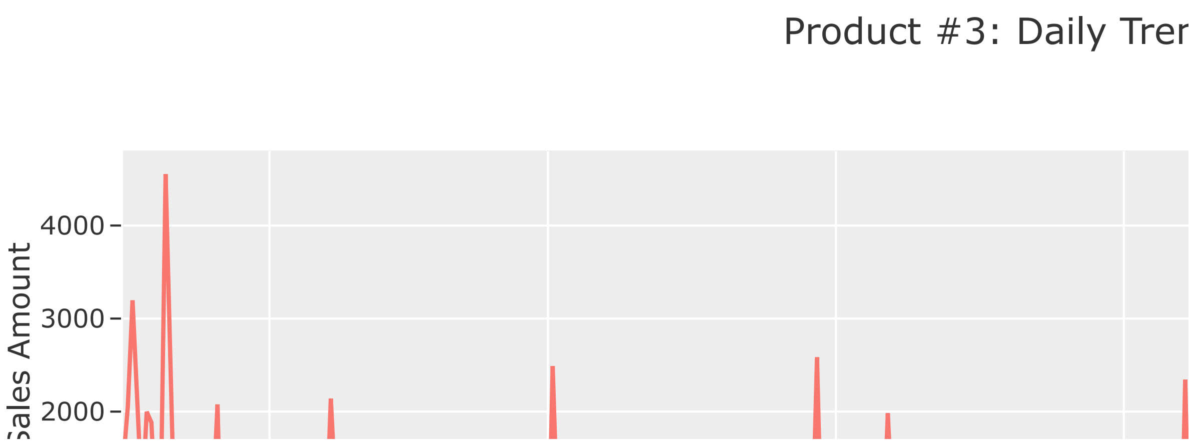
Based on our forecast model the product has downward weekly trend. This product highest spike is in December and July time. The forecast line seems struggling little bit to capture the weekly trend for this product. This can be due the fact is more observation is required.

In [ ]:
```

```

Product #3 Daily: Daily Trend

In [349]:
```
ds_daily_P3 = dfp3.groupby(by=['Date'])['Total_Sales_Amount'].sum().re
set_index()
```

In [350]:
```
fig = go.Figure(data=[go.Scatter(x=ds_daily_P3.Date,y=ds_daily_P3.Tota
l_Sales_Amount)])
fig.update_layout(xaxis_title="Date",yaxis_title="Total Sales Amount",
title='Product #3: Daily Trend',height=400,template='ggplot2')
fig.show()
```



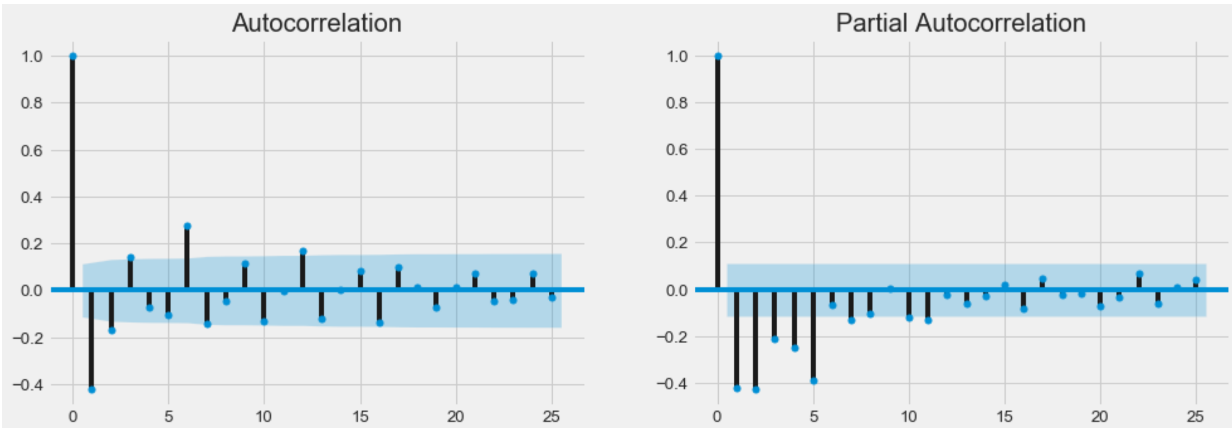
Product #3 Daily: Test of Stationarity with 1 differencing of series

In [351]:
```
print('\n*************************Daily*****************************
****')
series_date_3 = ds_daily_P3.Total_Sales_Amount.diff(d)
series_date_3 = series_date_3.dropna()
output = adfuller(series_date_3)
print('ADF Statistic: {0:.2f} and P value:{1:.5f}'.format(*output))
print("As we can see the p value is close to zero which is less than .
05 hence we reject null hypothesis")
```

```
*************************Daily*********************************
ADF Statistic: -8.68 and P value:0.00000
As we can see the p value is close to zero which is less than .05 he
nce we reject null hypothesis
```

Product #3 Daily: PACF & ACF

In [352]:
```
fig, ax = plt.subplots(1,2,figsize=(15,5))
plot_acf(series_date_3, ax=ax[0])
plot_pacf(series_date_3, ax=ax[1])
plt.show()
```



Product #3 Daily: Train & Test Split

In [353]:
```
series_date_3=ds_daily_P3.Total_Sales_Amount
split_time = 250
time_d=np.arange(len(ds_daily_P3))
xtrain_d=series_date_3[:split_time]
xtest_d=series_date_3[split_time:]
timeTrain_d = time_d[:split_time]
timeTest_d = time_d[split_time:]
```

Product #3 Daily: ARIMA Model

In [354]:
```
s_model = ARIMA(endog=xtrain_d , order=(1, 1, 0))
s_model_fit=s_model.fit()
print(s_model_fit.summary())
```

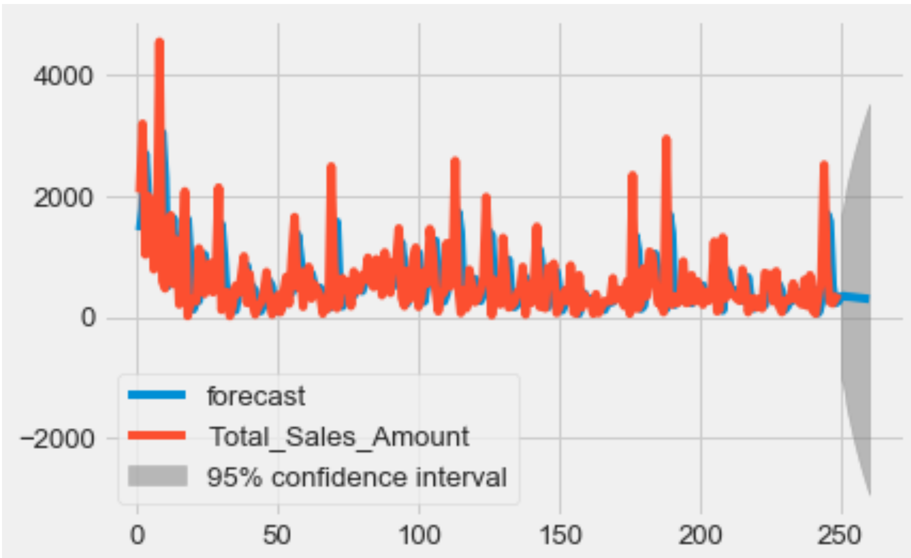
```
                              ARIMA Model Results
==============================================================================
============
Dep. Variable:     D.Total_Sales_Amount   No. Observations:
249
Model:                   ARIMA(1, 1, 0)   Log Likelihood
-1979.917
Method:                         css-mle   S.D. of innovations
686.850
Date:                  Wed, 06 Jan 2021   AIC
3965.833
Time:                          03:38:52   BIC
3976.385
Sample:                               1   HQIC
3970.081

==============================================================================
==========================
                                 coef    std err          z      P>|
z|      [0.025      0.975]
------------------------------------------------------------------------------
--------------------------
const                         -5.1094     30.410     -0.168      0.8
67     -64.712      54.493
ar.L1.D.Total_Sales_Amount    -0.4331      0.057     -7.598      0.0
00      -0.545      -0.321
                                    Roots
==============================================================================
=========
                  Real          Imaginary           Modulus
Frequency
------------------------------------------------------------------------------
---------
AR.1           -2.3089          +0.0000j            2.3089
0.5000
------------------------------------------------------------------------------
---------
```

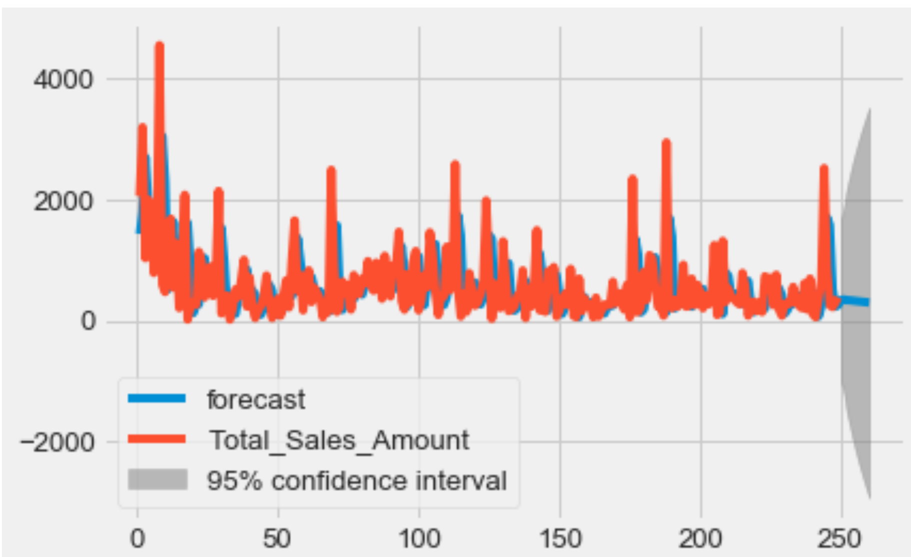
Product #3 Daily: Daily Trend and Forecast

In [358]:
```
ytrain_pred = s_model_fit.predict()
ytest_pred = s_model_fit.predict(start=min(timeTest_d),end=max(timeTes
t_d),dynamic=True)

print('RMSE Train :',np.sqrt(np.mean((ytrain_pred - xtrain_d)**2)))
print('RMSE Test :',np.sqrt(np.mean((ytest_pred - xtest_d)**2)))
forecast = s_model_fit.forecast(20, alpha=0.05)
s_model_fit.plot_predict(1,260)
```

```
RMSE Train : 906.3767753359934
RMSE Test : 667.7277857345033
```

Out[358]:





In [137]:
```
s_model_fit.forecast(30)[0]
```

Out[137]:
```
array([335.06480942, 347.3512854 , 334.707618  , 332.86127609,
       326.33857769, 321.84122436, 316.46668755, 311.47206174,
       306.31289524, 301.22499185, 296.10622418, 291.00082393,
       285.8896342 , 280.78095192, 275.67118365, 270.56188573,
       265.45238409, 260.34297069, 255.23351907, 250.12408401,
       245.01464177, 239.90520264, 234.79576217, 229.68632228,
       224.57688213, 219.4674421 , 214.35800201, 209.24856195,
       204.13912188, 199.02968182])
```

Based on our daily forecast we can see we have a downward trend for this product like we mentioned in our weekly trend for this product. Since this product is a cake stand I can see why it is very popular during December holiday time and not as popular in other holidays like other products.

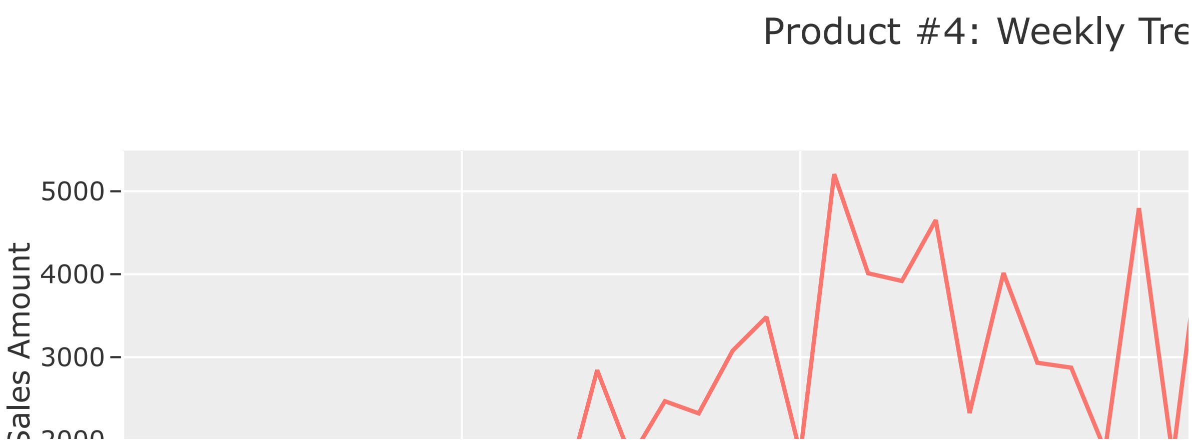
In [ ]:
```

```

Product #4: 'Party Bunting'

In [359]:
```
ds_weekly_P4 = dfp4.groupby(by=['Year','Week'])['Total_Sales_Amount'].
sum().reset_index()
ds_daily_P4 = dfp4.groupby(by=['Date'])['Total_Sales_Amount'].sum().re
set_index()
```

Product #4 Weekly: Weekly Trend

In [360]:
```
fig = go.Figure(data=[go.Scatter(x=ds_weekly_P4.index,y=ds_weekly_P4.Total_Sales_Amount)])
fig.update_layout(xaxis_title="Week",yaxis_title="Total Sales Amount",
title='Product #4: Weekly Trend',height=400,template='ggplot2')
fig.show()
```



Product #4 Weekly: Test of Stationarity of Actual Series

In [361]:
```
output = adfuller(ds_weekly_P4.Total_Sales_Amount)
print('**************************Week*********************************')
print('ADF Statistic: {0:.2f} and P value:{1:.5f}'.format(*output))
print("As we can see the p value is extreemly high which indicates that we are fail to reject null hypothesis " \
      "and can conclude that series is not stationary ")
```

```
**************************Week*********************************
ADF Statistic: -1.55 and P value:0.51080
As we can see the p value is extreemly high which indicates that we are fail to reject null hypothesis and can conclude that series is not stationary
```

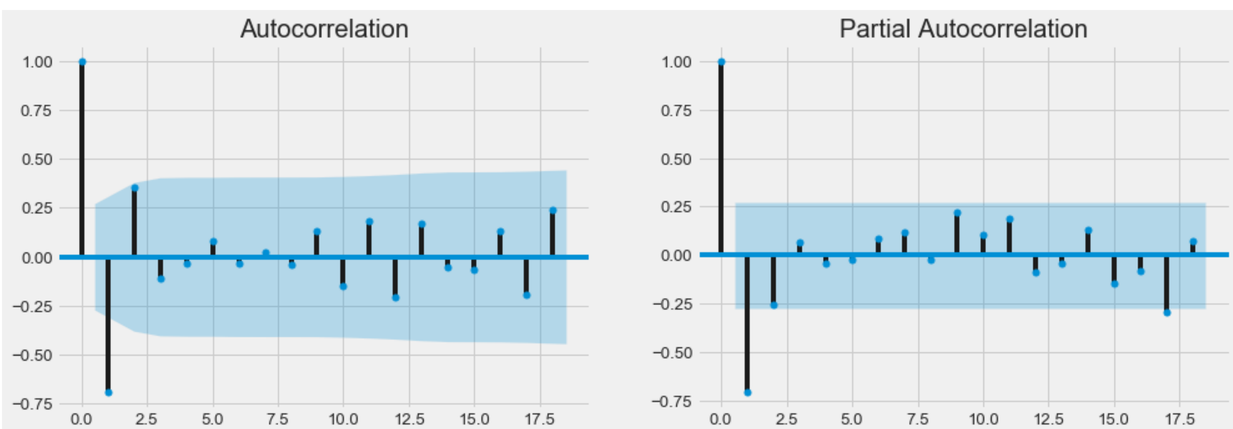
Product #4 Weekly: Test of Stationarity with 1 differencing of series

In [362]:
```
d=1
print('*************************Week********************************
*')
series_4 = ds_weekly_P4.Total_Sales_Amount.diff(d)
series_4 = series_4.dropna()
output = adfuller(series_4)

print('ADF Statistic: {0:.2f} and P value:{1:.5f}'.format(*output))
print("As we can see the p value is close to zero which is less than .
05 hence we reject null hypothesis")
```

```
**************************Week********************************
ADF Statistic: -7.96 and P value:0.00000
As we can see the p value is close to zero which is less than .05 he
nce we reject null hypothesis
```

Product #4 Weekly: PACF & ACF

In [363]:
```
fig, ax = plt.subplots(1,2,figsize=(15,5))
plot_acf(series_4, ax=ax[0])
plot_pacf(series_4, ax=ax[1])
plt.show()
```



Product #4 Weekly: Train & Test Split

In [364]:
```
series_4=ds_weekly_P4.Total_Sales_Amount
split_time = 42
time=np.arange(len(ds_weekly_P4))
xtrain=series_4[:split_time]
xtest=series_4[split_time:]
timeTrain = time[:split_time]
timeTest = time[split_time:]
```

Product #4 Weekly: ARIMA Model

In [365]:
```
model = ARIMA(xtrain, order=(1,1,0))
model_fit = model.fit()
print(model_fit.summary())
```

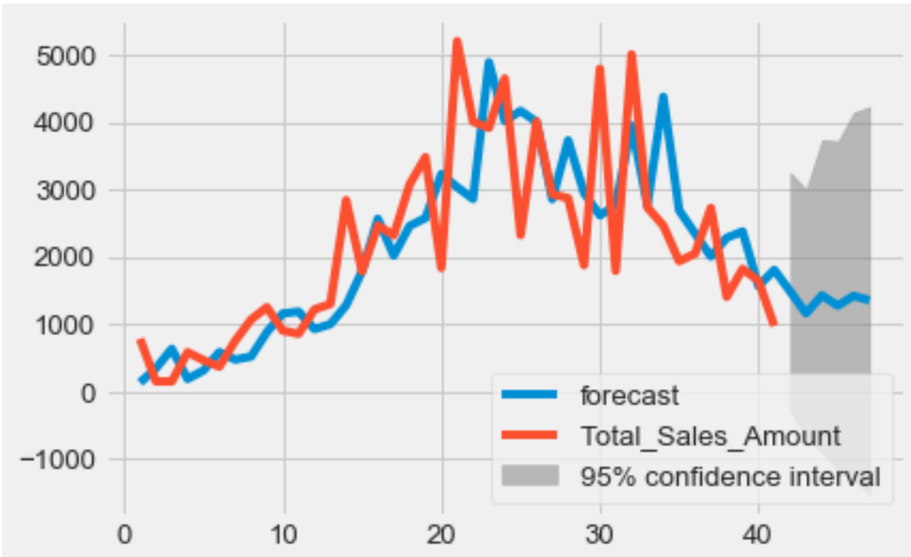
```
                                 ARIMA Model Results
====================================================================
============
Dep. Variable:     D.Total_Sales_Amount   No. Observations:
41
Model:                   ARIMA(1, 1, 0)   Log Likelihood
-337.494
Method:                         css-mle   S.D. of innovations
901.606
Date:                  Wed, 06 Jan 2021   AIC
680.988
Time:                          03:47:42   BIC
686.129
Sample:                               1   HQIC
682.860

====================================================================
==========================
                                 coef    std err          z      P>|
z|      [0.025      0.975]
--------------------------------------------------------------------
--------------------------
const                         21.5046     83.340      0.258      0.7
98    -141.838     184.847
ar.L1.D.Total_Sales_Amount    -0.7069      0.106     -6.692      0.0
00      -0.914      -0.500
                                    Roots
====================================================================
=========
                  Real          Imaginary           Modulus
Frequency
--------------------------------------------------------------------
---------
AR.1           -1.4146           +0.0000j            1.4146
0.5000
--------------------------------------------------------------------
---------
```

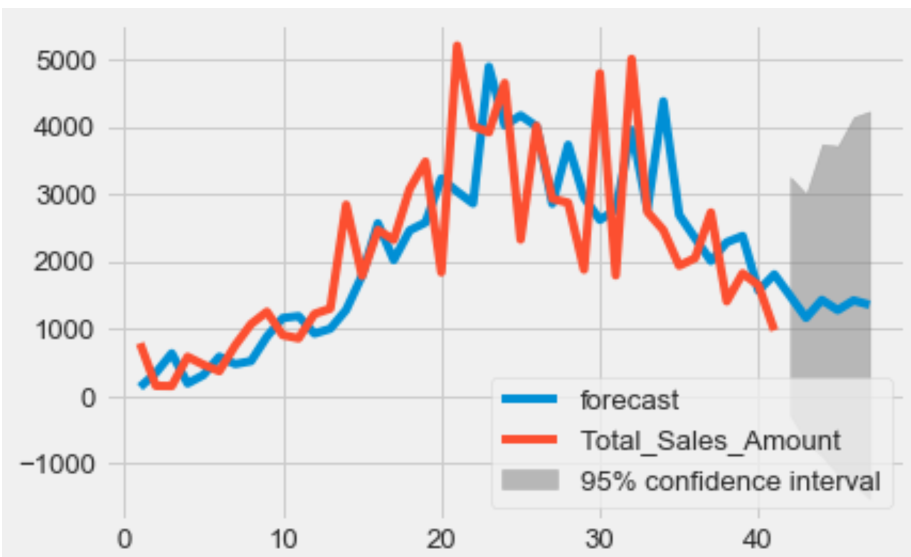
Product #4 Weekly: Weekly Trend and Forecast

In [367]:
```
ytrain_pred = model_fit.predict()
ytest_pred = model_fit.predict(start=min(timeTest),end=max(timeTest),d
ynamic=True)
print('RMSE Train :',np.sqrt(np.mean((ytrain_pred - xtrain)**2)))
print('RMSE Test :',np.sqrt(np.mean((ytest_pred - xtest)**2)))
forecast = model_fit.forecast(20, alpha=0.05)
model_fit.plot_predict(1,47)
```

```
RMSE Train : 2629.2003150370124
RMSE Test : 1042.6618421295823
```

Out[367]:





In [146]:
```
model_fit.forecast(30)[0]
```

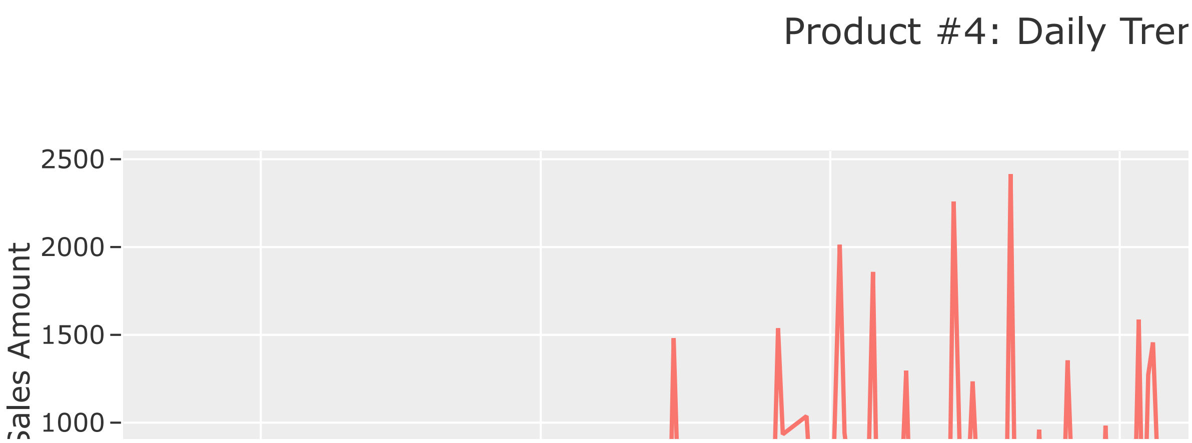
Out[146]:
```
array([1488.96341566, 1164.82820539, 1430.66203893, 1279.45292292,
       1423.04710819, 1358.24792165, 1440.75983111, 1419.13906679,
       1471.12860097, 1471.08378466, 1507.82150674, 1518.55803695,
       1547.67453788, 1563.79842966, 1589.10666102, 1607.92257874,
       1631.32784511, 1651.48894889, 1673.94331729, 1694.77660091,
       1716.75581241, 1737.92497947, 1759.66675848, 1781.00376408,
       1802.62689978, 1824.0477731 , 1845.61162358, 1867.07440501,
       1888.60863109, 1910.09235369])
```

Our weekly forecast model trend is upward for this product as we can see and it is also struggling little to align with the weekly trend. Based on RSME my model may be underfitting. The graph indicates the demand for this product is mostly in May and June time. The forecast indicates an upward trend in December time but it is not going to crazy as May or June time.

Product #4 Daily: Daily Trend

In [368]:
```
ds_daily_P4 = dfp4.groupby(by=['Date'])['Total_Sales_Amount'].sum().re
set_index()
```

In [369]:
```
fig = go.Figure(data=[go.Scatter(x=ds_daily_P4.Date,y=ds_daily_P4.Tota
l_Sales_Amount)])
fig.update_layout(xaxis_title="Date",yaxis_title="Total Sales Amount",
title='Product #4: Daily Trend',height=400,template='ggplot2')
fig.show()
```



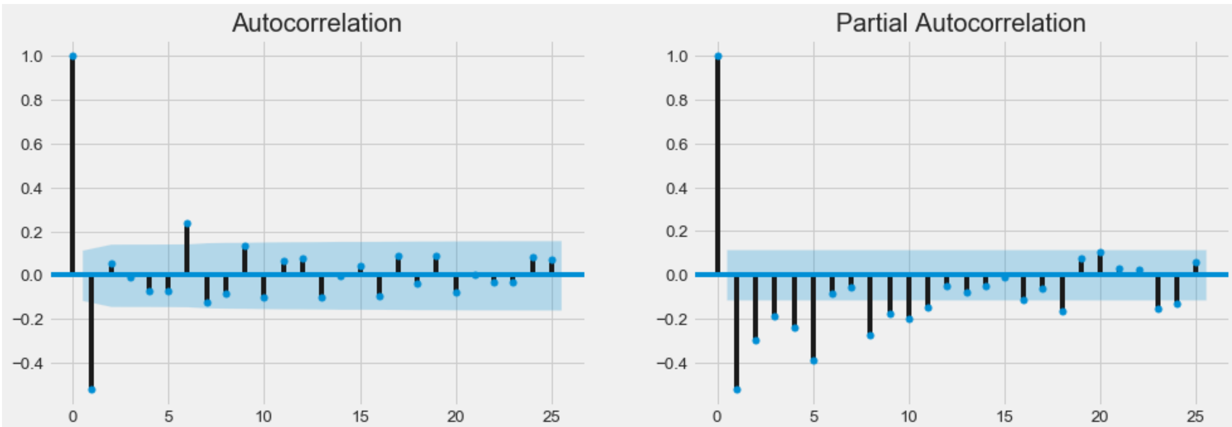
Product #4 Daily: Test of Stationarity with 1 differencing of series

In [370]:
```
print('\n*************************Daily*****************************
****')
series_date_4 = ds_daily_P4.Total_Sales_Amount.diff(d)
series_date_4 = series_date_4.dropna()
output = adfuller(series_date_4)
print('ADF Statistic: {0:.2f} and P value:{1:.5f}'.format(*output))
print("As we can see the p value is close to zero which is less than .
05 hence we reject null hypothesis")
```

```
*************************Daily********************************
ADF Statistic: -10.11 and P value:0.00000
As we can see the p value is close to zero which is less than .05 he
nce we reject null hypothesis
```

Product #4 Daily: PACF & ACF

In [371]:
```
fig, ax = plt.subplots(1,2,figsize=(15,5))
plot_acf(series_date_4, ax=ax[0])
plot_pacf(series_date_4, ax=ax[1])
plt.show()
```



Product #4 Daily: Train & Test Split

In [372]:
```
series_date_4=ds_daily_P4.Total_Sales_Amount
split_time_4 = 236
time_d=np.arange(len(ds_daily_P4))
xtrain_d_4=series_date_4[:split_time_4]
xtest_d_4=series_date_4[split_time_4:]
timeTrain_d = time_d[:split_time_4]
timeTest_d = time_d[split_time_4:]
```

Product #4 Daily: ARIMA Model

In [373]:
```
s_model_4 = ARIMA(endog=xtrain_d_4 , order=(1, 1, 0))
s_model_fit_4=s_model_4.fit()
print(s_model_fit_4.summary())
```

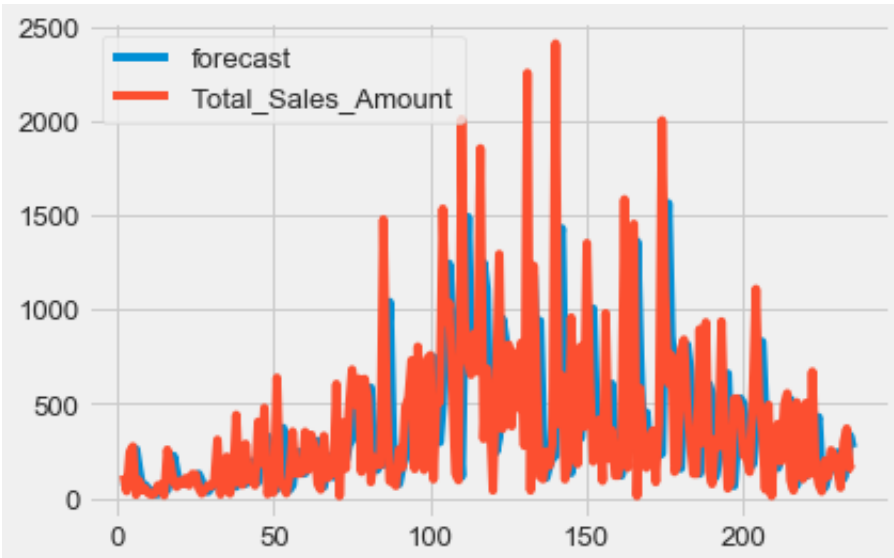
```
                              ARIMA Model Results
==============================================================================
=============
Dep. Variable:     D.Total_Sales_Amount   No. Observations:
235
Model:                   ARIMA(1, 1, 0)   Log Likelihood
-1771.549
Method:                         css-mle   S.D. of innovations
454.371
Date:                  Wed, 06 Jan 2021   AIC
3549.097
Time:                          04:01:10   BIC
3559.476
Sample:                               1   HQIC
3553.281

==============================================================================
==========================
                                 coef    std err          z      P>|
z|      [0.025      0.975]
------------------------------------------------------------------------------
--------------------------
const                          0.3745     19.606      0.019      0.9
85     -38.052      38.801
ar.L1.D.Total_Sales_Amount    -0.5140      0.056     -9.233      0.0
00      -0.623      -0.405
                                    Roots
==============================================================================
=========
                  Real          Imaginary           Modulus
Frequency
------------------------------------------------------------------------------
---------
AR.1           -1.9456           +0.0000j            1.9456
0.5000
------------------------------------------------------------------------------
---------
```

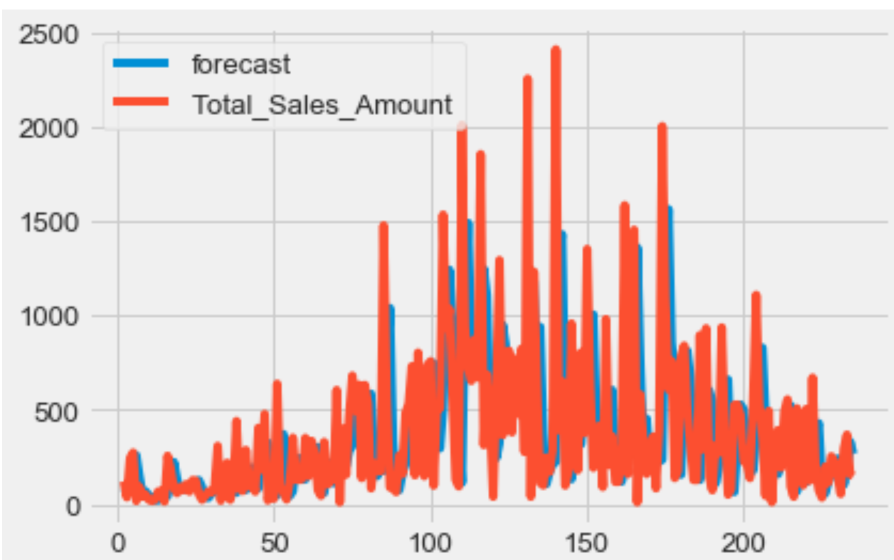
Product #4 Daily: Daily Trend and Forecast

In [374]:
```
ytrain_pred_4 = s_model_fit_4.predict()
ytest_pred_4 = s_model_fit_4.predict(start=min(timeTest_d),end=max(tim
eTest_d),dynamic=True)

print('RMSE Train :',np.sqrt(np.mean((ytrain_pred_4 - xtrain_d_4)**2))
)
print('RMSE Test :',np.sqrt(np.mean((ytest_pred_4 - xtest_d_4)**2)))
forecast = s_model_fit_4.forecast(20, alpha=0.05)
s_model_fit_4.plot_predict()
```

```
RMSE Train : 625.0065246027209
RMSE Test : 257.0304067579681
```

Out[374]:





In [154]:
```
s_model_fit_4.forecast(30)[0]
```

Out[154]:
```
array([154.15534469, 151.81555076, 153.58509949, 153.24252761,
       153.98554383, 154.1705893 , 154.64242067, 154.96684977,
       155.36704071, 155.72829157, 156.10955686, 156.48053513,
       156.85680072, 157.23034874, 157.60529353, 157.97952041,
       158.35411629, 158.7285225 , 159.1030262 , 159.4774798 ,
       159.85195914, 160.22642525, 160.60089816, 160.97536758,
       161.3498388 , 161.72430909, 162.09877985, 162.47325037,
       162.84772102, 163.2221916 ])
```

Even my model forecast have a little hard time capturing the spikes it well Capturing the mid points. It does look a lot better than the weekly trend model. The spikes mainly happen in middle from end of march to end of August

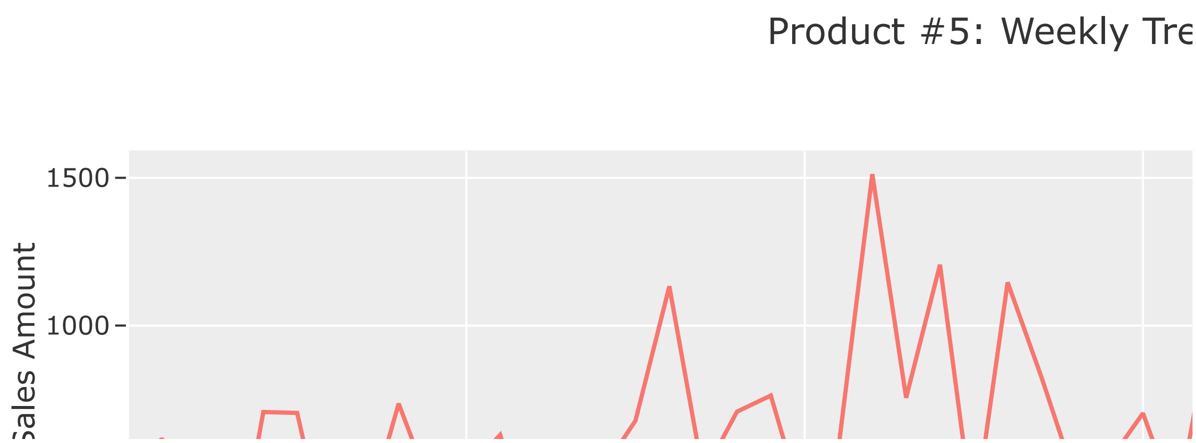
In [ ]:

Product #5: 'Lunch Bag Red Retrospot'

In [375]:
```
ds_weekly_P5 = dfp5.groupby(by=['Year','Week'])['Total_Sales_Amount'].
sum().reset_index()
```

Product #5 Weekly: Weekly Trend

In [376]:
```
fig = go.Figure(data=[go.Scatter(x=ds_weekly_P5.index,y=ds_weekly_P5.Total_Sales_Amount)])
fig.update_layout(xaxis_title="Week",yaxis_title="Total Sales Amount",
title='Product #5: Weekly Trend',height=400,template='ggplot2')
fig.show()
```



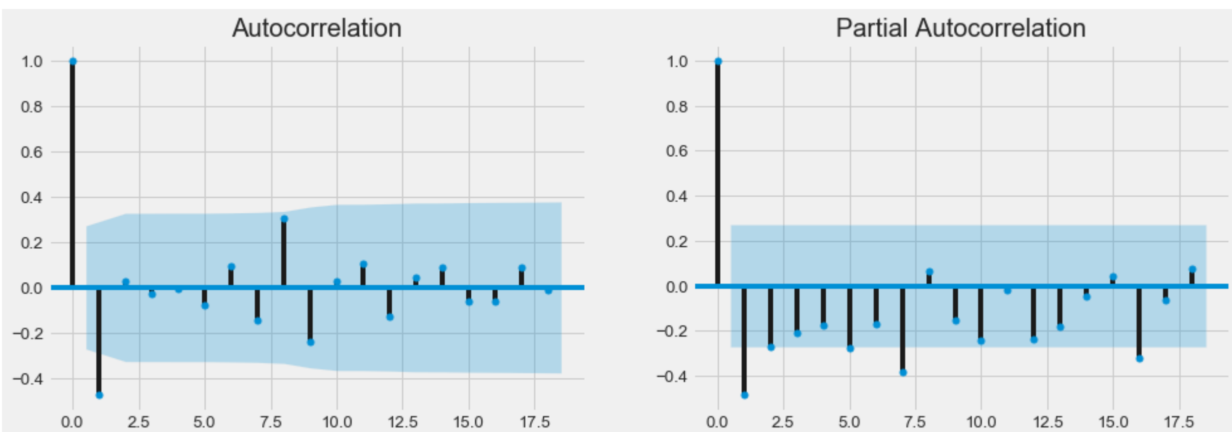
Product #5 Weekly: Test of Stationarity with 1 differencing of series

In [377]:
```
d=1
print('**************************Week*********************************
*')
series_5 = ds_weekly_P5.Total_Sales_Amount.diff(d)
series_5 = series_5.dropna()
output = adfuller(series_5)

print('ADF Statistic: {0:.2f} and P value:{1:.5f}'.format(*output))
print("As we can see the p value is close to zero which is less than .
05 hence we reject null hypothesis")
```

```
**************************Week********************************
ADF Statistic: -5.14 and P value:0.00001
As we can see the p value is close to zero which is less than .05 he
nce we reject null hypothesis
```

Product #5 Weekly: PACF & ACF

In [378]:
```
fig, ax = plt.subplots(1,2,figsize=(15,5))
plot_acf(series_5, ax=ax[0])
plot_pacf(series_5, ax=ax[1])
plt.show()
```



Product #5 Weekly: Train & Test Split

In [379]:
```
series_5=ds_weekly_P5.Total_Sales_Amount
split_time = 42
time=np.arange(len(ds_weekly_P1))
xtrain=series_5[:split_time]
xtest=series_5[split_time:]
timeTrain = time[:split_time]
timeTest = time[split_time:]
```

Product #5 Weekly: ARIMA Model

In [380]:
```
model = ARIMA(xtrain, order=(1,1,0))
model_fit = model.fit()
print(model_fit.summary())
```

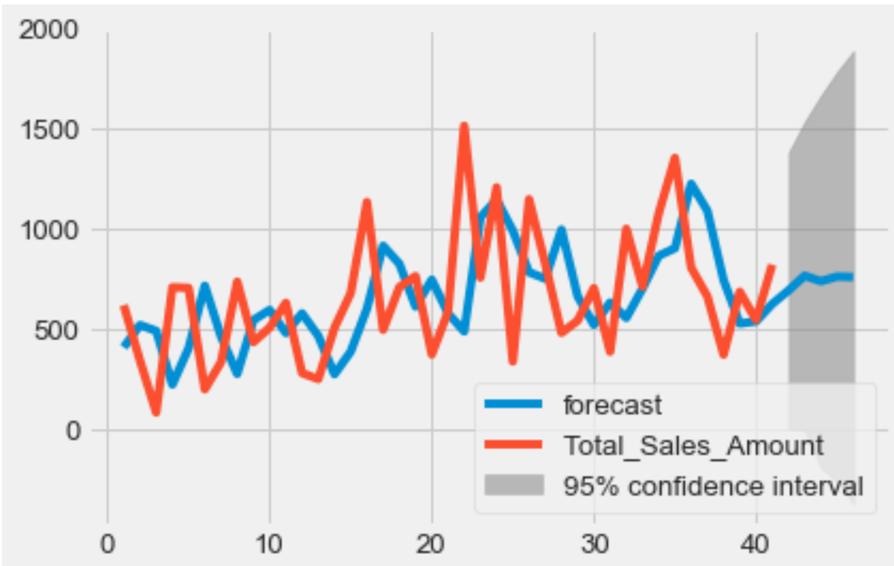
```
                              ARIMA Model Results
==============================================================================
============
Dep. Variable:     D.Total_Sales_Amount   No. Observations:
41
Model:                   ARIMA(1, 1, 0)   Log Likelihood
-298.758
Method:                         css-mle   S.D. of innovations
352.243
Date:                  Wed, 06 Jan 2021   AIC
603.517
Time:                          04:09:19   BIC
608.657
Sample:                               1   HQIC
605.389

==============================================================================
==========================
                                 coef    std err          z      P>|
z|      [0.025      0.975]
------------------------------------------------------------------------------
--------------------------
const                          6.2894     36.948      0.170      0.8
66     -66.128      78.707
ar.L1.D.Total_Sales_Amount    -0.5014      0.133     -3.772      0.0
01      -0.762      -0.241
                                    Roots
==============================================================================
=========
                  Real          Imaginary           Modulus
Frequency
------------------------------------------------------------------------------
---------
AR.1           -1.9944           +0.0000j            1.9944
0.5000
------------------------------------------------------------------------------
---------
```

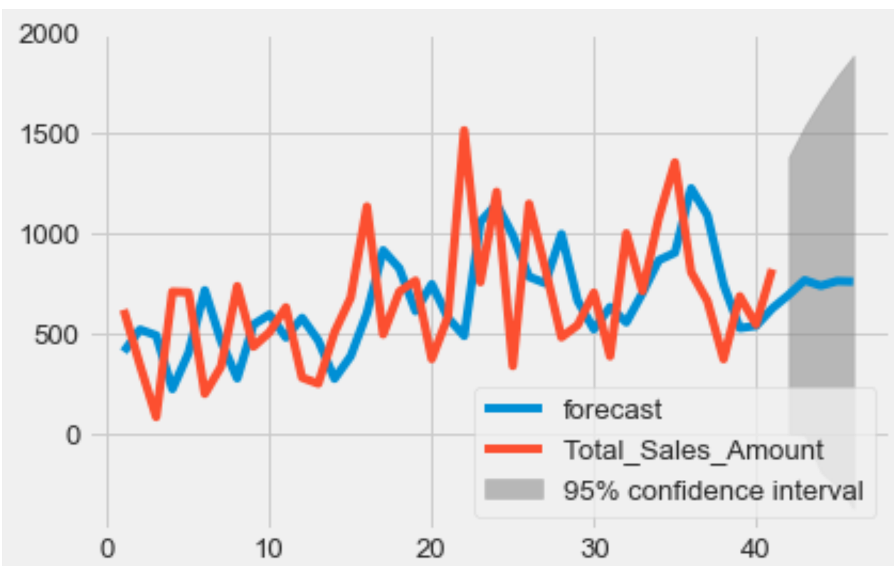
Product #5 Weekly: Weekly Trend and Forecast

In [383]:
```
ytrain_pred = model_fit.predict()
ytest_pred = model_fit.predict(start=min(timeTest),end=max(timeTest),d
ynamic=True)
print('RMSE Train :',np.sqrt(np.mean((ytrain_pred - xtrain)**2)))
print('RMSE Test :',np.sqrt(np.mean((ytest_pred - xtest)**2)))
forecast = model_fit.forecast(20, alpha=0.05)
model_fit.plot_predict(1,46)
```

```
RMSE Train : 750.9393711026631
RMSE Test : 777.364053672424
```

Out[383]:





In [382]:
```
model_fit.forecast(30)[0]
```

Out[382]:
```
array([691.18911775, 765.17924614, 737.52277924, 760.83305871,
       758.58803819, 769.15673258, 773.30048616, 780.66577734,
       786.41575332, 792.97566652, 799.12946809, 805.4868986 ,
       811.74222726, 818.04875094, 824.32960486, 830.62332989,
       836.9106012 , 843.20110847, 849.4899932 , 855.77969149,
       862.06898185, 868.35847676, 874.6478691 , 880.93731286,
       887.22673085, 893.51616176, 899.80558619, 906.09501386,
       912.38443991, 918.67386678])
```

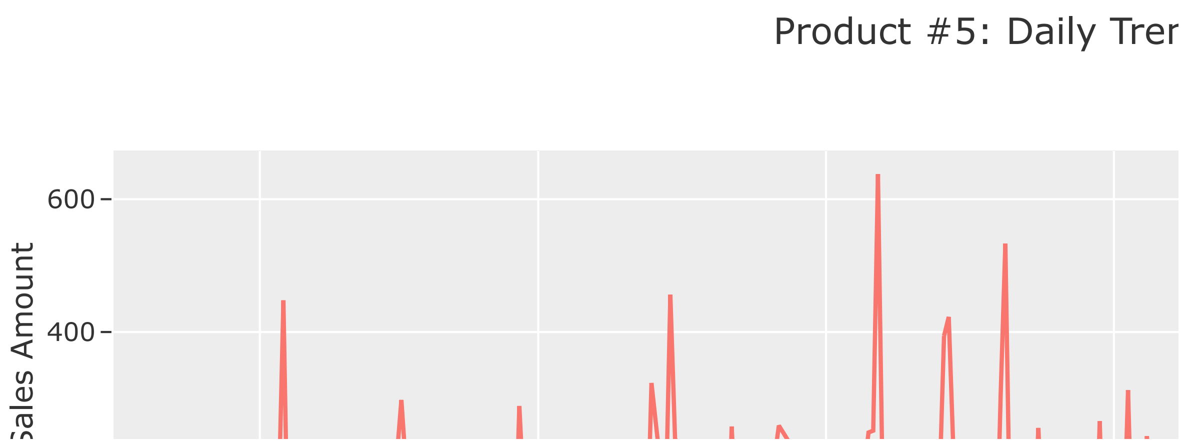
Based on the weekly forecast this product has a upward trend. The graph and RMSE score indicates my model might be underfitting. I believe with more observations this issue can be solved. As I can see in graph this product has the highest spike during mid May time

In [ ]:

Product #5 Daily: Daily Trend

In [384]:
```
ds_daily_P5 = dfp5.groupby(by=['Date'])['Total_Sales_Amount'].sum().re
set_index()
```

In [385]:
```
fig = go.Figure(data=[go.Scatter(x=ds_daily_P5.Date,y=ds_daily_P5.Tota
l_Sales_Amount)])
fig.update_layout(xaxis_title="Date",yaxis_title="Total Sales Amount",
title='Product #5: Daily Trend',height=400,template='ggplot2')
fig.show()
```



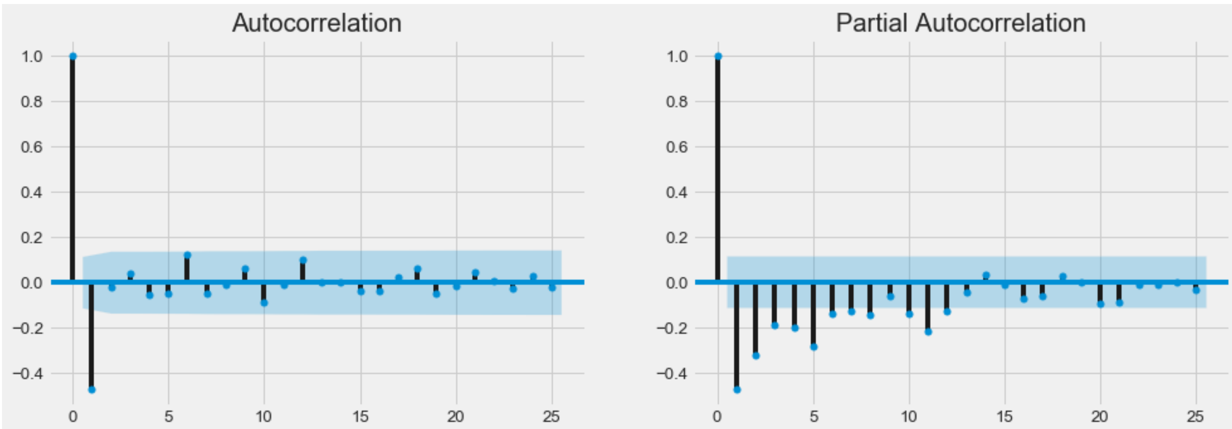
Product #5 Daily: Test of Stationarity with 1 differencing of series

In [392]:
```
print('\n*************************Daily*****************************
****')
series_date_5 = ds_daily_P5.Total_Sales_Amount.diff(d)
series_date_5 = series_date_5.dropna()
output = adfuller(series_date_5)
print('ADF Statistic: {0:.2f} and P value:{1:.5f}'.format(*output))
print("As we can see the p value is close to zero which is less than .
05 hence we reject null hypothesis")
```

```
*************************Daily********************************
ADF Statistic: -9.46 and P value:0.00000
As we can see the p value is close to zero which is less than .05 he
nce we reject null hypothesis
```

Product #5 Daily: PACF & ACF

In [393]:
```
fig, ax = plt.subplots(1,2,figsize=(15,5))
plot_acf(series_date_5, ax=ax[0])
plot_pacf(series_date_5, ax=ax[1])
plt.show()
```



Product #5 Daily: Train & Test Split

In [394]:
```
series_date_5=ds_daily_P5.Total_Sales_Amount
split_time = 238
time_d=np.arange(len(ds_daily_P5))
xtrain_d=series_date_5[:split_time]
xtest_d=series_date_5[split_time:]
timeTrain_d = time_d[:split_time]
timeTest_d = time_d[split_time:]
```

Product #5 Daily: ARIMA Model

In [395]:
```
s_model = ARIMA(endog=xtrain_d , order=(1, 1, 0))
s_model_fit=s_model.fit()
print(s_model_fit.summary())
```

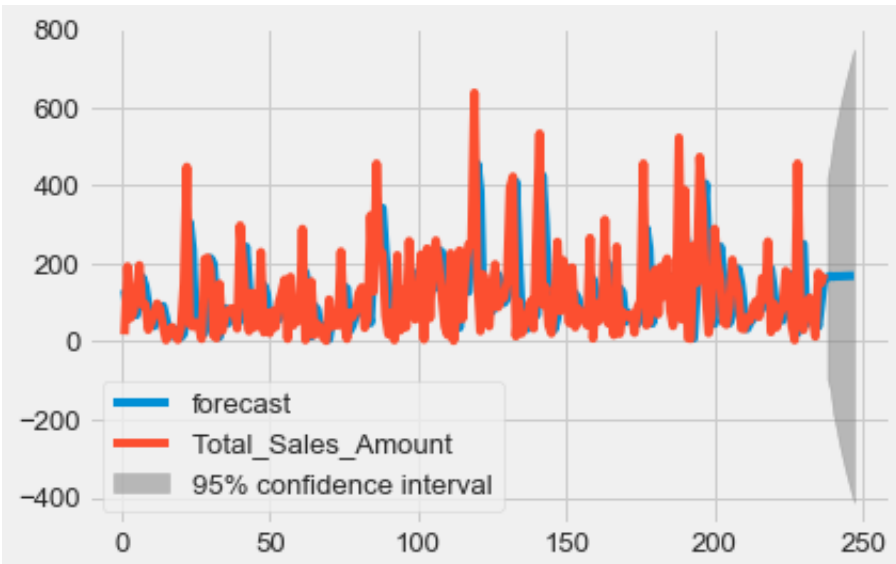
```
                              ARIMA Model Results
==============================================================================
============
Dep. Variable:     D.Total_Sales_Amount   No. Observations:
237
Model:                   ARIMA(1, 1, 0)   Log Likelihood
-1493.624
Method:                         css-mle   S.D. of innovations
131.992
Date:                  Wed, 06 Jan 2021   AIC
2993.249
Time:                          04:22:08   BIC
3003.653
Sample:                               1   HQIC
2997.443

==============================================================================
==========================
                              coef    std err          z      P>|
z|      [0.025      0.975]
------------------------------------------------------------------------------
--------------------------
const                       0.2966      5.826      0.051      0.9
59     -11.121      11.715
ar.L1.D.Total_Sales_Amount -0.4737      0.057     -8.305      0.0
00      -0.586      -0.362
                                    Roots
==============================================================================
=========
                 Real          Imaginary           Modulus
Frequency
------------------------------------------------------------------------------
---------
AR.1           -2.1109           +0.0000j            2.1109
0.5000
------------------------------------------------------------------------------
---------
```

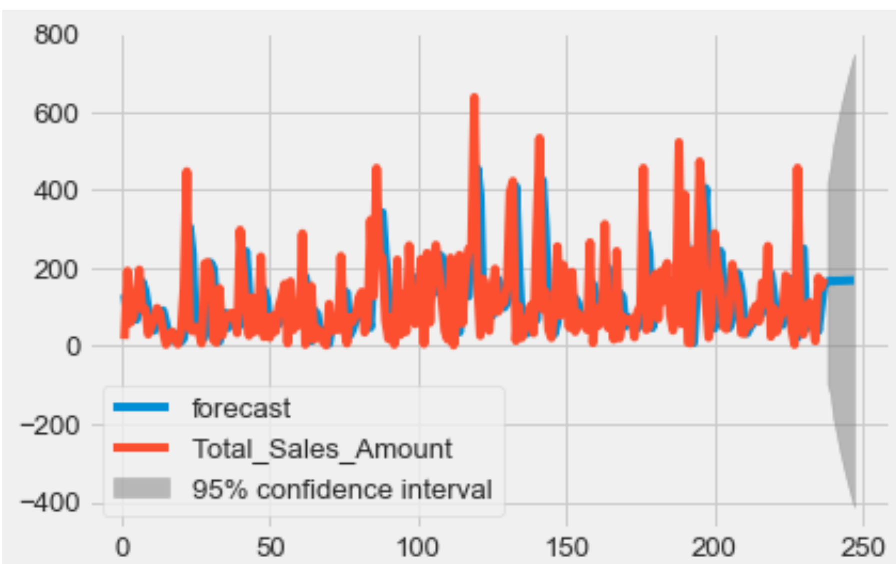
Product #5 Daily: Daily Trend and Forecast

In [396]:
```
ytrain_pred = s_model_fit.predict()
ytest_pred = s_model_fit.predict(start=min(timeTest_d),end=max(timeTes
t_d),dynamic=True)

print('RMSE Train :',np.sqrt(np.mean((ytrain_pred - xtrain_d)**2)))
print('RMSE Test :',np.sqrt(np.mean((ytest_pred - xtest_d)**2)))
forecast = s_model_fit.forecast(20, alpha=0.05)
print('Lunch Bag Red Retrospot Daily Trend and Forecast')
s_model_fit.plot_predict(1,247)
```

```
RMSE Train : 177.90099590826765
RMSE Test : 153.11877435188427
Lunch Bag Red Retrospot Daily Trend and Forecast
```

Out[396]:





In [397]:
```
s_model_fit.forecast(30)[0]
```

Out[397]:
```
array([162.35537431, 167.95358961, 165.73860002, 167.2249843 ,
       166.95790521, 167.52149978, 167.69157673, 168.04807601,
       168.31626086, 168.62628318, 168.91648571, 169.21607753,
       169.51122133, 169.8084723 , 170.10472503, 170.40145067,
       170.69795227, 170.99456001, 171.29111747, 171.58769874,
       171.88426874, 172.18084407, 172.47741688, 172.77399089,
       173.07056432, 173.36713803, 173.66371161, 173.96028525,
       174.25685886, 174.55343248])
```

Based on my daily forecast model this product has a upward trend. This model fits better to the daily trend in compare to the weekly trend. Most of spikes in the product happen between mid May and August. The biggest spike happens in mid May close to spring break. My suggestion based on the model would to have some inventory for end of December however I don't believe to see a major spike like in May for this product.

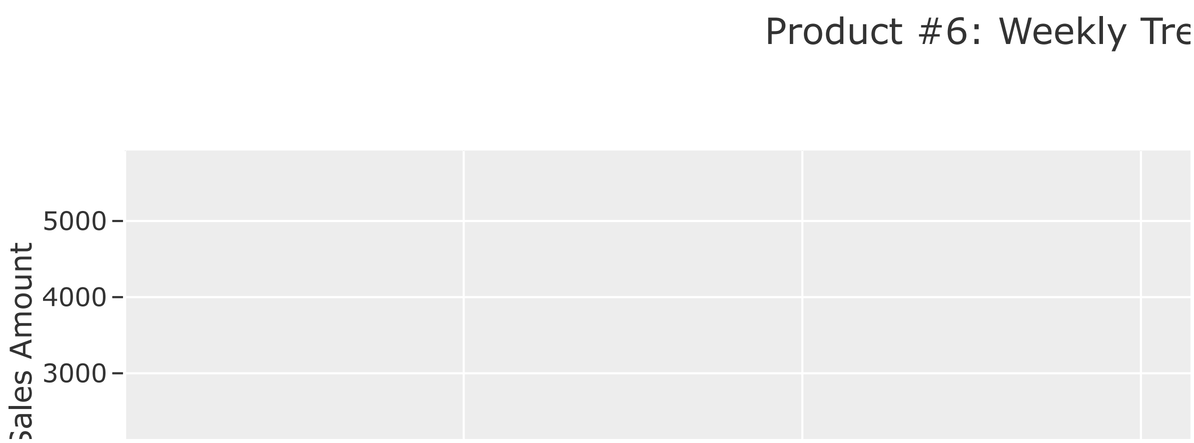
In [ ]:
```

```

Product #6: 'Assorted Colour Bird Ornament'

In [398]:
```
ds_weekly_P6 = dfp6.groupby(by=['Year','Week'])['Total_Sales_Amount'].
sum().reset_index()
```

Product #6 Weekly: Weekly Trend

In [399]:
```
fig = go.Figure(data=[go.Scatter(x=ds_weekly_P6.index,y=ds_weekly_P6.T
otal_Sales_Amount)])
fig.update_layout(xaxis_title="Week",yaxis_title="Total Sales Amount",
title='Product #6: Weekly Trend',height=400,template='ggplot2')
fig.show()
```



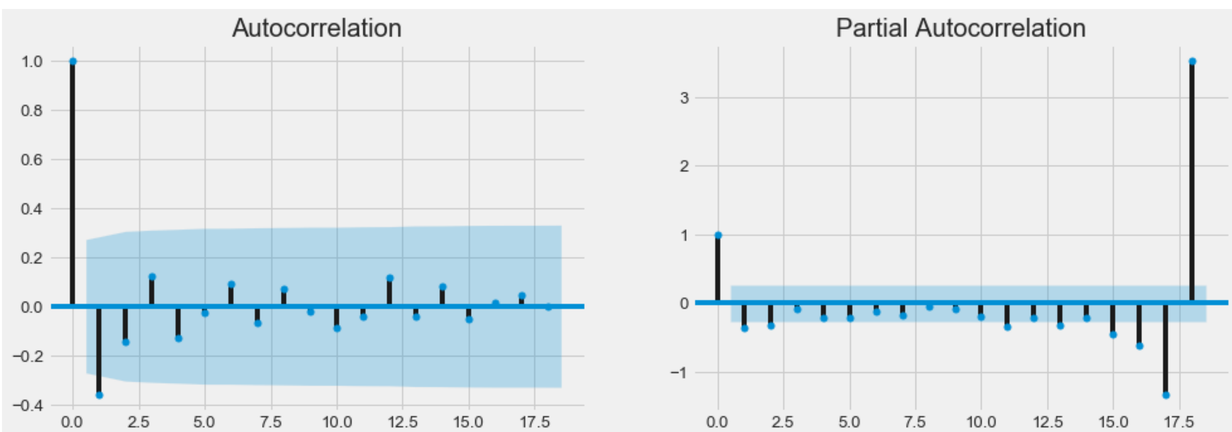
Product #6 Weekly: Test of Stationarity with 1 differencing of series

In [400]:
```
d=1
print('**************************Week********************************
*')
series_6 = ds_weekly_P6.Total_Sales_Amount.diff(d)
series_6 = series_6.dropna()
output = adfuller(series_6)

print('ADF Statistic: {0:.2f} and P value:{1:.5f}'.format(*output))
print("As we can see the p value is close to zero which is less than .
05 hence we reject null hypothesis")
```

```
**************************Week********************************
ADF Statistic: -7.87 and P value:0.00000
As we can see the p value is close to zero which is less than .05 he
nce we reject null hypothesis
```

Product #6 Weekly: PACF & ACF

In [401]:
```
fig, ax = plt.subplots(1,2,figsize=(15,5))
plot_acf(series_6, ax=ax[0])
plot_pacf(series_6, ax=ax[1])
plt.show()
```



Product #6 Weekly: Train & Test Split

In [402]:
```
series_6=ds_weekly_P6.Total_Sales_Amount
split_time = 42
time=np.arange(len(ds_weekly_P6))
xtrain=series_6[:split_time]
xtest=series_6[split_time:]
timeTrain = time[:split_time]
timeTest = time[split_time:]
```

Product #6 Weekly: ARIMA Model

In [403]:
```
model = ARIMA(xtrain, order=(1,1,0))
model_fit = model.fit()
print(model_fit.summary())
```

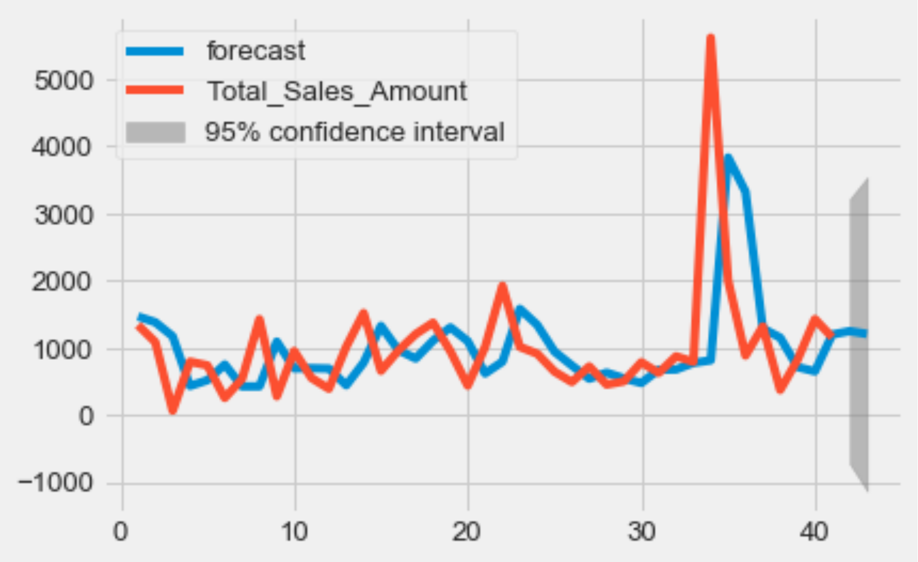
```
                              ARIMA Model Results
==============================================================================
Dep. Variable:     D.Total_Sales_Amount   No. Observations:                   41
Model:                   ARIMA(1, 1, 0)   Log Likelihood                -341.727
Method:                         css-mle   S.D. of innovations           1006.345
Date:                  Wed, 06 Jan 2021   AIC                            689.453
Time:                          04:33:55   BIC                            694.594
Sample:                               1   HQIC                           691.325

==============================================================================
                                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const                         -5.1447    115.646     -0.044      0.965    -231.807     221.517
ar.L1.D.Total_Sales_Amount    -0.3681      0.143     -2.581      0.014      -0.647      -0.089
                                    Roots
=============================================================================
                  Real          Imaginary           Modulus         Frequency
-----------------------------------------------------------------------------
AR.1           -2.7170           +0.0000j            2.7170            0.5000
-----------------------------------------------------------------------------
```

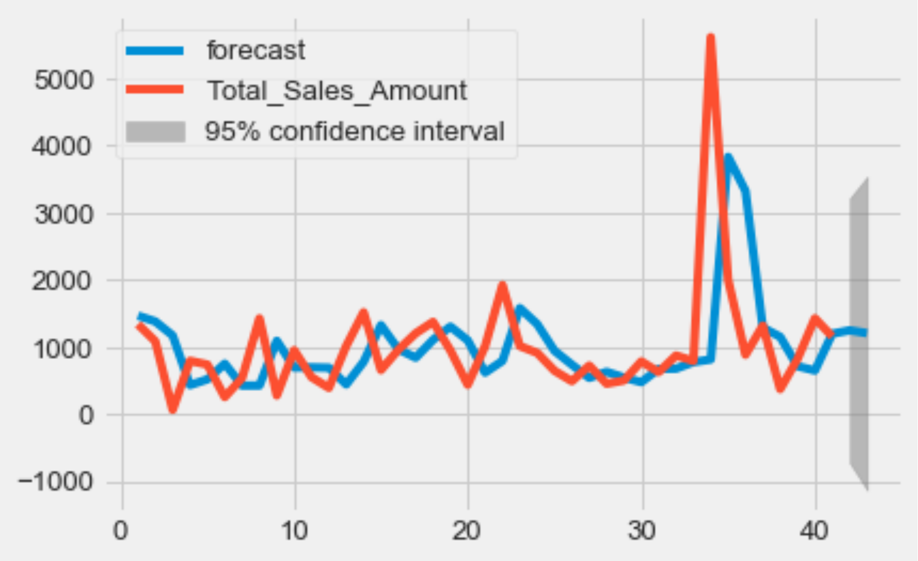
Product #6 Weekly: Weekly Trend and Forecast

In [407]:
```
ytrain_pred = model_fit.predict()
ytest_pred = model_fit.predict(start=min(timeTest),end=max(timeTest),d
ynamic=True)
print('RMSE Train :',np.sqrt(np.mean((ytrain_pred - xtrain)**2)))
print('RMSE Test :',np.sqrt(np.mean((ytest_pred - xtest)**2)))
forecast = model_fit.forecast(20, alpha=0.05)
model_fit.plot_predict(1,43)
```

```
RMSE Train : 1412.630042593545
RMSE Test : 1580.884087123212
```

Out[407]:





In [405]:
```
model_fit.forecast(30)[0]
```

Out[405]:
```
array([1251.18065929, 1209.74787688, 1217.9590317 , 1207.89870357,
       1204.56320647, 1198.75263099, 1193.8530097 , 1188.61811113,
       1183.50661158, 1178.34969495, 1173.2094941 , 1168.06314099,
       1162.91905223, 1157.77413008, 1152.62951465, 1147.48478633,
       1142.34009957, 1137.19539751, 1132.05070108, 1126.90600257,
       1121.76130483, 1116.61660681, 1111.47190889, 1106.32721094,
       1101.182513  , 1096.03781505, 1090.8931171 , 1085.74841916,
       1080.60372121, 1075.45902327])
```

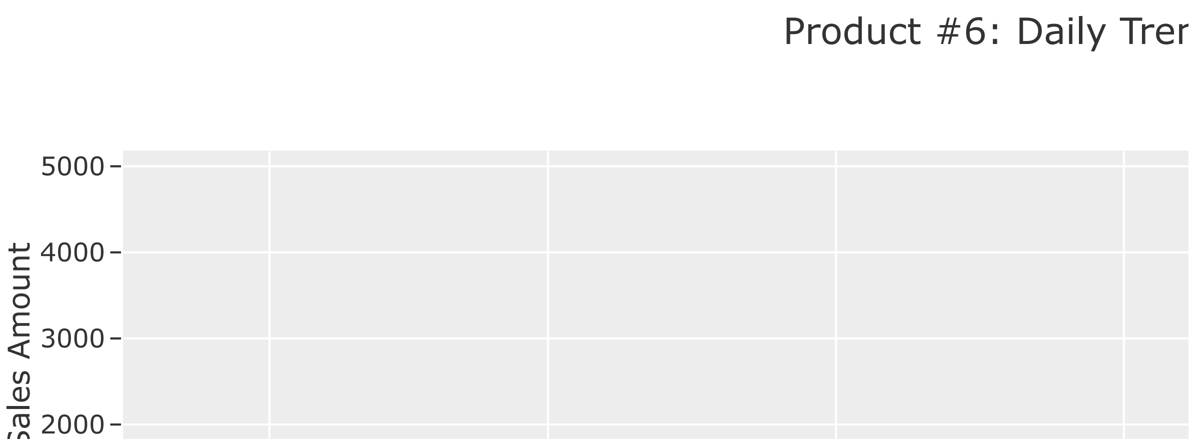
Based on our weekly forecast model this product has a downward trend. The major spike occurs in the beginning of August. In compare to the other products, this product doesn't seem to have much big spike movements.

In [ ]:
```

```

Product #6 Daily: Daily Trend

In [408]:
```
ds_daily_P6 = dfp6.groupby(by=['Date'])['Total_Sales_Amount'].sum().re
set_index()
```

In [409]:
```
fig = go.Figure(data=[go.Scatter(x=ds_daily_P6.Date,y=ds_daily_P6.Tota
l_Sales_Amount)])
fig.update_layout(xaxis_title="Date",yaxis_title="Total Sales Amount",
title='Product #6: Daily Trend',height=400,template='ggplot2')
fig.show()
```



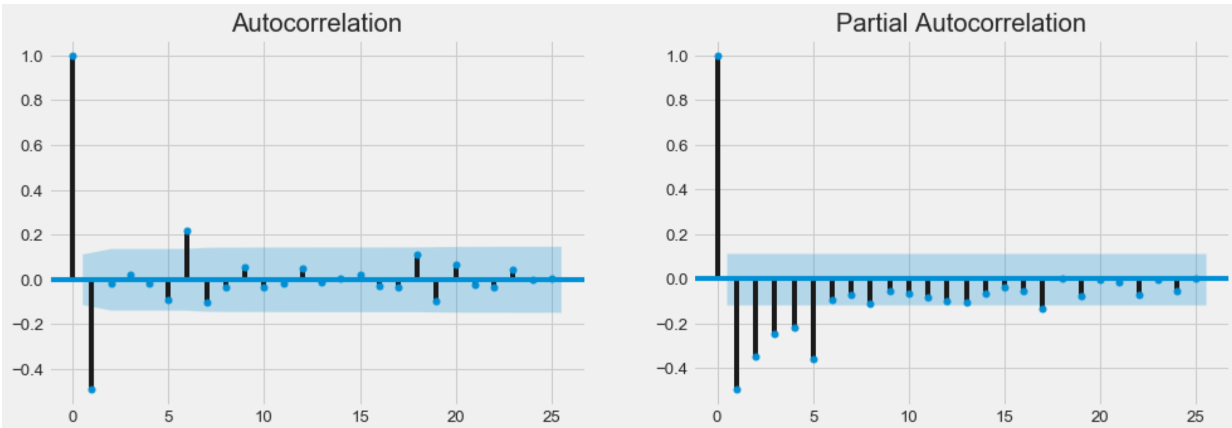
Product #6 Daily: Test of Stationarity with 1 differencing of series

In [410]:
```
print('\n**************************Daily******************************
****')
series_date_6 = ds_daily_P6.Total_Sales_Amount.diff(d)
series_date_6 = series_date_6.dropna()
output = adfuller(series_date_6)
print('ADF Statistic: {0:.2f} and P value:{1:.5f}'.format(*output))
print("As we can see the p value is close to zero which is less than .
05 hence we reject null hypothesis")
```

```
**************************Daily********************************
ADF Statistic: -9.99 and P value:0.00000
As we can see the p value is close to zero which is less than .05 he
nce we reject null hypothesis
```

Product #6 Daily: PACF & ACF

In [411]:
```
fig, ax = plt.subplots(1,2,figsize=(15,5))
plot_acf(series_date_6, ax=ax[0])
plot_pacf(series_date_6, ax=ax[1])
plt.show()
```



Product #6 Daily: Train & Test Split

In [412]:
```
series_date_6=ds_daily_P6.Total_Sales_Amount
split_time = 240
time_d=np.arange(len(ds_daily_P6))
xtrain_d=series_date_6[:split_time]
xtest_d=series_date_6[split_time:]
timeTrain_d = time_d[:split_time]
timeTest_d = time_d[split_time:]
```

Product #6 Daily: ARIMA Model

In [413]:
```
s_model = ARIMA(endog=xtrain_d , order=(1, 1, 0))
s_model_fit=s_model.fit()
print(s_model_fit.summary())
```

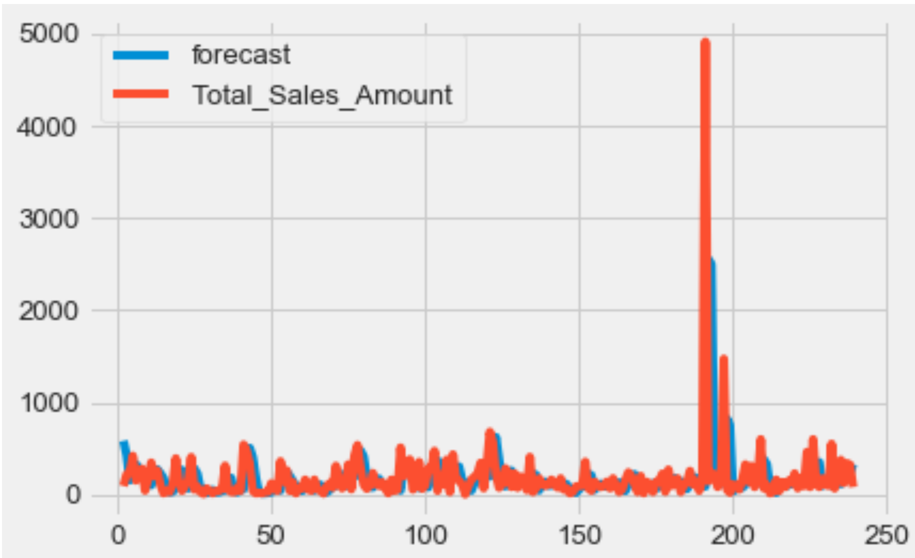
```
                              ARIMA Model Results
=====================================================================
============
Dep. Variable:     D.Total_Sales_Amount   No. Observations:
239
Model:                   ARIMA(1, 1, 0)   Log Likelihood
-1784.992
Method:                         css-mle   S.D. of innovations
423.715
Date:                  Wed, 06 Jan 2021   AIC
3575.984
Time:                          04:42:03   BIC
3586.413
Sample:                               1   HQIC
3580.187

=====================================================================
==========================
                                 coef    std err          z      P>|
z|      [0.025      0.975]
---------------------------------------------------------------------
--------------------------
const                         -1.4950     18.373     -0.081      0.9
35     -37.506      34.516
ar.L1.D.Total_Sales_Amount    -0.4938      0.056     -8.800      0.0
00      -0.604      -0.384
                                    Roots
=====================================================================
=========
                  Real          Imaginary           Modulus
Frequency
---------------------------------------------------------------------
---------
AR.1           -2.0251           +0.0000j            2.0251
0.5000
---------------------------------------------------------------------
---------
```

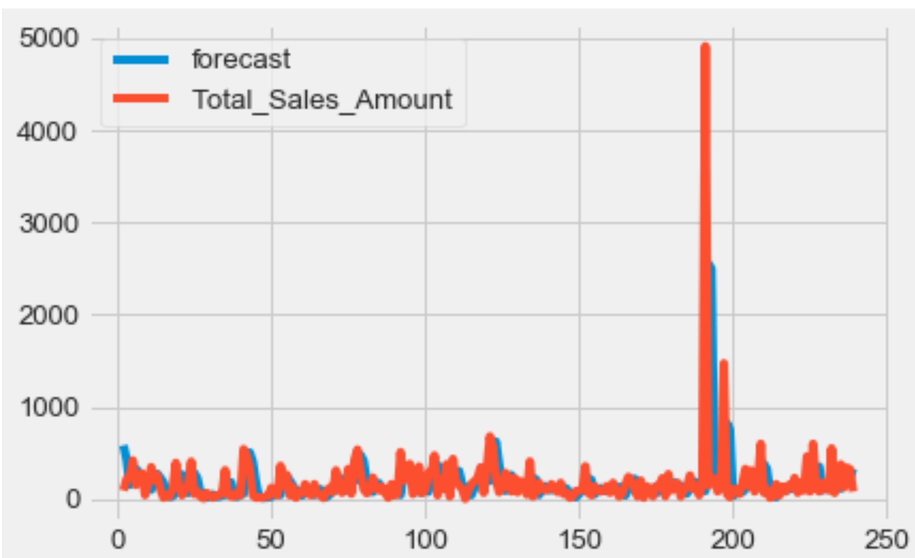
Product #6 Daily: Daily Trend and Forecast

In [414]:
```
ytrain_pred = s_model_fit.predict()
ytest_pred = s_model_fit.predict(start=min(timeTest_d),end=max(timeTes
t_d),dynamic=True)

print('RMSE Train :',np.sqrt(np.mean((ytrain_pred - xtrain_d)**2)))
print('RMSE Test :',np.sqrt(np.mean((ytest_pred - xtest_d)**2)))
forecast = s_model_fit.forecast(20, alpha=0.05)
s_model_fit.plot_predict()
```

```
RMSE Train : 459.64873798363493
RMSE Test : 375.0939642290524
```

Out[414]:





In [415]:
```
s_model_fit.forecast(30)[0]
```

Out[415]:
```
array([198.03332417, 138.06711131, 165.44582788, 149.6927225 ,
       155.23855315, 150.26675075, 150.48866342, 148.14586632,
       147.06954921, 145.3678313 , 143.97494271, 142.42955103,
       140.95946695, 139.4521952 , 137.9632871 , 136.46531085,
       134.97181254, 133.47610297, 131.98148535, 130.48632852,
       128.99143795, 127.4964159 , 126.00145877, 124.50646959,
       123.01149624, 121.51651506, 120.02153775, 118.52655854,
       117.03158026, 115.53660152])
```

Based on my daily forecasts, this product has slight downward trend. Again there is not much movement in this product and even the forecast doesn't suggest a big movement. The big spikes only occurs in August and beginning in November

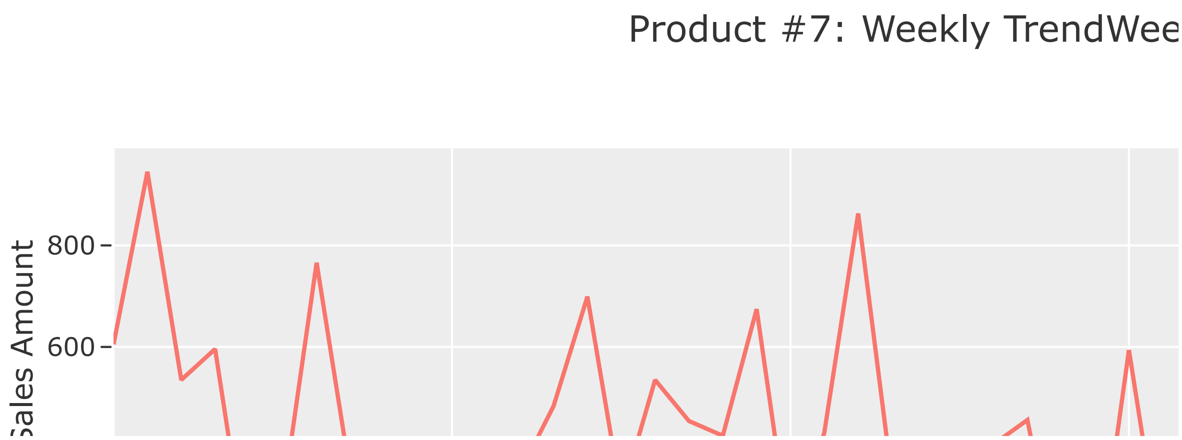
In [ ]:
```

```

Product #7: 'Set of 3 Cake Tins Pantry Design'

In [416]:
```
ds_weekly_P7 = dfp7.groupby(by=['Year','Week'])['Total_Sales_Amount'].
sum().reset_index()
```

Product #7 Weekly: Weekly Trend

In [417]:
```
fig = go.Figure(data=[go.Scatter(x=ds_weekly_P7.index,y=ds_weekly_P7.T
otal_Sales_Amount)])
fig.update_layout(xaxis_title="Week",yaxis_title="Total Sales Amount",
title='Product #7: Weekly TrendWeekly Trend',height=400,template='ggpl
ot2')
fig.show()
```



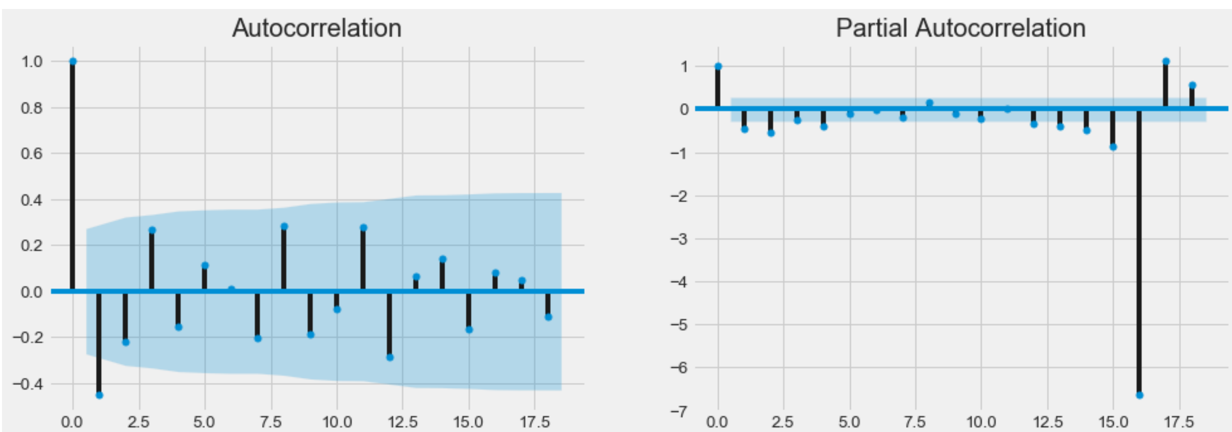
Product #7 Weekly: Test of Stationarity with 1 differencing of series

In [418]:
```
d=1
print('**************************Week********************************
*')
series_7 = ds_weekly_P7.Total_Sales_Amount.diff(d)
series_7 = series_7.dropna()
output = adfuller(series_7)

print('ADF Statistic: {0:.2f} and P value:{1:.5f}'.format(*output))
print("As we can see the p value is close to zero which is less than .
05 hence we reject null hypothesis")
```

```
**************************Week*********************************
ADF Statistic: -7.59 and P value:0.00000
As we can see the p value is close to zero which is less than .05 he
nce we reject null hypothesis
```

Product #7 Weekly: PACF & ACF

In [419]:
```
fig, ax = plt.subplots(1,2,figsize=(15,5))
plot_acf(series_7, ax=ax[0])
plot_pacf(series_7, ax=ax[1])
plt.show()
```



Product #7 Weekly: Train & Test Split

In [420]:
```
series_7=ds_weekly_P7.Total_Sales_Amount
split_time = 42
time=np.arange(len(ds_weekly_P7))
xtrain=series_7[:split_time]
xtest=series_7[split_time:]
timeTrain = time[:split_time]
timeTest = time[split_time:]
```

Product #7 Weekly: ARIMA Model

In [421]:
```
model = ARIMA(xtrain, order=(1,1,0))
model_fit = model.fit()
print(model_fit.summary())
```

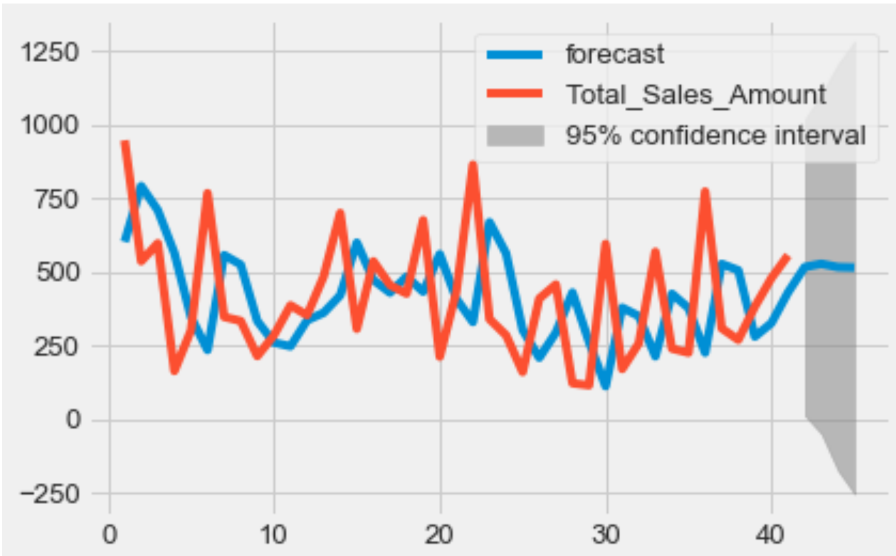
```
                              ARIMA Model Results
======================================================================
============
Dep. Variable:     D.Total_Sales_Amount   No. Observations:
41
Model:                   ARIMA(1, 1, 0)   Log Likelihood
-285.710
Method:                         css-mle   S.D. of innovations
256.466
Date:                  Wed, 06 Jan 2021   AIC
577.419
Time:                          04:48:10   BIC
582.560
Sample:                               1   HQIC
579.291

======================================================================
==========================
                                 coef    std err          z      P>|
z|      [0.025      0.975]
----------------------------------------------------------------------
--------------------------
const                         -4.3827     28.079     -0.156      0.8
77     -59.416      50.651
ar.L1.D.Total_Sales_Amount    -0.4376      0.140     -3.117      0.0
03      -0.713      -0.162
                                    Roots
======================================================================
=========
                  Real          Imaginary           Modulus
Frequency
----------------------------------------------------------------------
---------
AR.1           -2.2850           +0.0000j            2.2850
0.5000
----------------------------------------------------------------------
---------
```

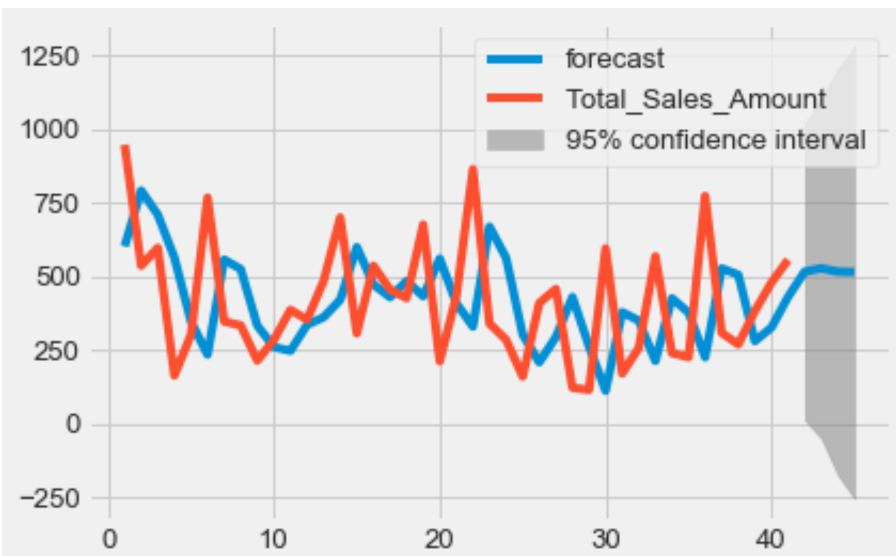
Product #7 Weekly: Weekly Trend and Forecast

In [423]:
```
ytrain_pred = model_fit.predict()
ytest_pred = model_fit.predict(start=min(timeTest),end=max(timeTest),d
ynamic=True)
print('RMSE Train :',np.sqrt(np.mean((ytrain_pred - xtrain)**2)))
print('RMSE Test :',np.sqrt(np.mean((ytest_pred - xtest)**2)))
forecast = model_fit.forecast(20, alpha=0.05)
model_fit.plot_predict(1,45)
```

```
RMSE Train : 469.5124619442146
RMSE Test : 347.8962531827525
```

Out[423]:





In [424]:
```
model_fit.forecast(30)[0]
```

Out[424]:
```
array([515.19054104, 526.42110018, 515.20560044, 513.81317306,
       508.12187294, 504.31188919, 499.67858466, 495.40559013,
       490.97491308, 486.61324266, 482.22137285, 477.84271921,
       473.45828177, 469.0763755 , 464.69336151, 460.31083229,
       455.92809092, 451.5454424 , 447.16275324, 442.78008187,
       438.39740271, 434.01472696, 429.63204972, 425.24937313,
       420.86669626, 416.48401951, 412.1013427 , 407.71866592,
       403.33598913, 398.95331235])
```

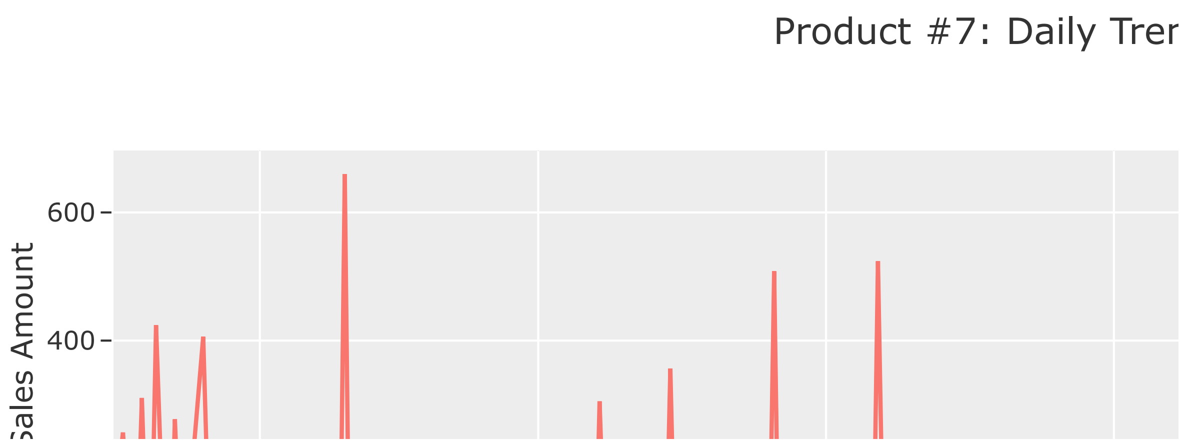
Based on my weekly forecast model we can slight downward trend for this product. Again my forecast seems having a hard time fitting with weekly trend line. Mostly this happens to my weekly models since I have lower observations. Based on the graph looks like most of sale activity happens mid March to April. The reason can be related to several holidays that happens during that time.

In [ ]:

Product #7 Daily: Daily Trend

In [425]:
```
ds_daily_P7 = dfp7.groupby(by=['Date'])['Total_Sales_Amount'].sum().re
set_index()
```

In [426]:
```
fig = go.Figure(data=[go.Scatter(x=ds_daily_P7.Date,y=ds_daily_P7.Tota
l_Sales_Amount)])
fig.update_layout(xaxis_title="Date",yaxis_title="Total Sales Amount",
title='Product #7: Daily Trend',height=400,template='ggplot2')
fig.show()
```



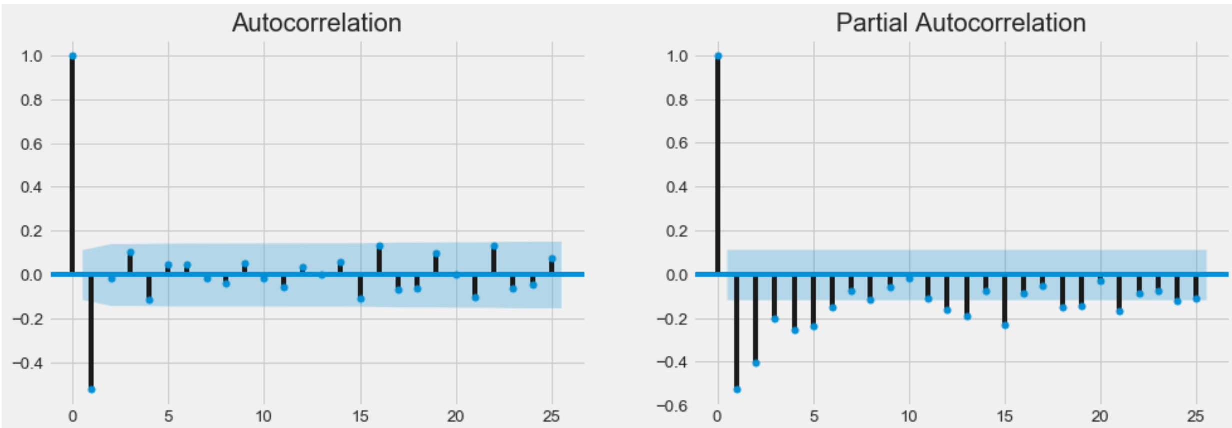
Product #7 Daily: Test of Stationarity with 1 differencing of series

In [427]:
```
print('\n*************************Daily*****************************
****')
series_date_7 = ds_daily_P7.Total_Sales_Amount.diff(d)
series_date_7 = series_date_7.dropna()
output = adfuller(series_date_7)
print('ADF Statistic: {0:.2f} and P value:{1:.5f}'.format(*output))
print("As we can see the p value is close to zero which is less than .
05 hence we reject null hypothesis")
```

```
*************************Daily*********************************
ADF Statistic: -8.97 and P value:0.00000
As we can see the p value is close to zero which is less than .05 he
nce we reject null hypothesis
```

Product #7 Daily: PACF & ACF

In [428]:
```
fig, ax = plt.subplots(1,2,figsize=(15,5))
plot_acf(series_date_7, ax=ax[0])
plot_pacf(series_date_7, ax=ax[1])
plt.show()
```



Product #7 Daily: Train & Test Split

In [429]:
```
series_date_7=ds_daily_P7.Total_Sales_Amount
split_time = 238
time_d=np.arange(len(ds_daily_P7))
xtrain_d=series_date_7[:split_time]
xtest_d=series_date_7[split_time:]
timeTrain_d = time_d[:split_time]
timeTest_d = time_d[split_time:]
```

Product #7 Daily: ARIMA Model

In [430]:
```
s_model = ARIMA(endog=xtrain_d , order=(1, 1, 0))
s_model_fit=s_model.fit()
print(s_model_fit.summary())
```

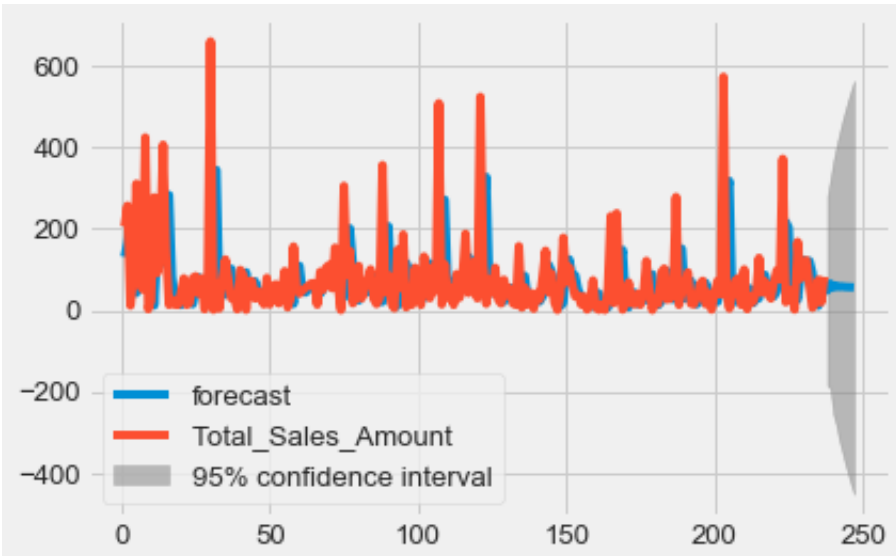
```
                             ARIMA Model Results
==============================================================================
============
Dep. Variable:     D.Total_Sales_Amount   No. Observations:
237
Model:                   ARIMA(1, 1, 0)   Log Likelihood
-1468.066
Method:                         css-mle   S.D. of innovations
118.482
Date:                  Wed, 06 Jan 2021   AIC
2942.133
Time:                          04:59:44   BIC
2952.537
Sample:                               1   HQIC
2946.326

==============================================================================
==========================
                                 coef    std err          z      P>|
z|      [0.025      0.975]
------------------------------------------------------------------------------
--------------------------
const                         -0.4078      5.065     -0.081      0.9
36     -10.334       9.519
ar.L1.D.Total_Sales_Amount    -0.5218      0.055     -9.453      0.0
00      -0.630      -0.414
                                    Roots
==============================================================================
=========
                  Real          Imaginary           Modulus
Frequency
------------------------------------------------------------------------------
---------
AR.1           -1.9164           +0.0000j            1.9164
0.5000
------------------------------------------------------------------------------
---------
```

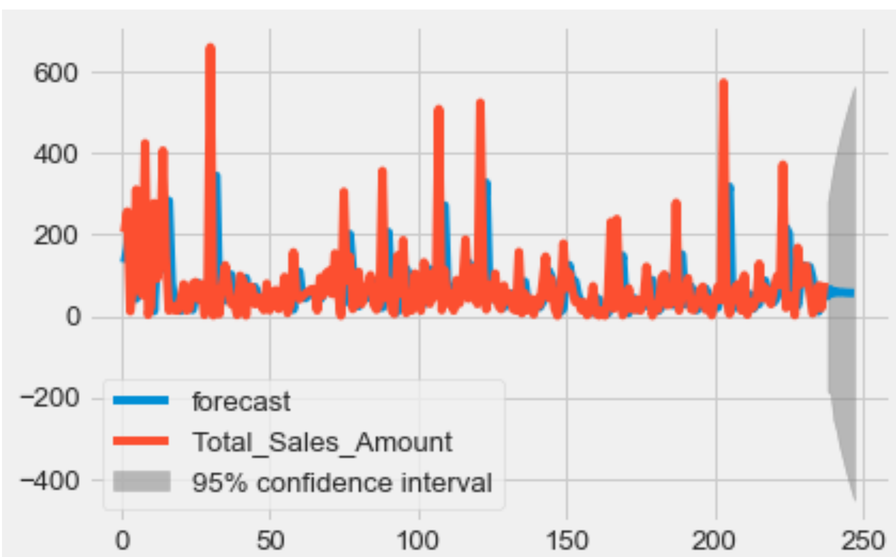
Product #7 Daily: Daily Trend and Forecast

In [437]:
```
ytrain_pred = s_model_fit.predict()
ytest_pred = s_model_fit.predict(start=min(timeTest_d),end=max(timeTes
t_d),dynamic=True)

print('RMSE Train :',np.sqrt(np.mean((ytrain_pred - xtrain_d)**2)))
print('RMSE Test :',np.sqrt(np.mean((ytest_pred - xtest_d)**2)))
forecast = s_model_fit.forecast(20, alpha=0.05)
s_model_fit.plot_predict(1,247)
```

```
RMSE Train : 139.3416354337633
RMSE Test : 71.53195291193856
```

Out[437]:





In [202]:
```
s_model_fit.forecast(30)[0]
```

Out[202]:
```
array([48.76227375, 65.5375642 , 56.16360539, 60.43446946, 57.585389
48,
       58.45153569, 57.37905633, 57.31816251, 56.72941776, 56.416108
54,
       55.95907554, 55.5770384 , 55.15586801, 54.75511757, 54.343711
88,
       53.93786616, 53.52911922, 53.12188616, 52.71386314, 52.306252
32,
       51.89842642, 51.49071275, 51.08294052, 50.67519884, 50.267441
22,
       49.85969192, 49.45193828, 49.0441869 , 48.63643434, 48.228682
4 ])
```

Based on my daily forecast model we can see slight downward trend for this product. I would suggest a small inventory sale during beginning of December. The big spike is happening in Jan and mid August.

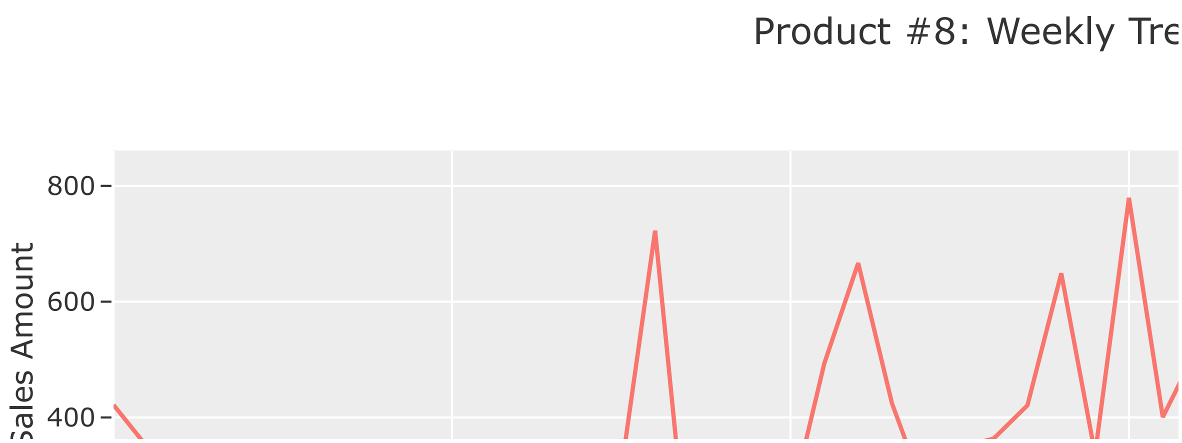
In [ ]:
```

```

Product #8: 'Pack of 72 Retrospot Cake Cases'

In [446]:
```
ds_weekly_P8 = dfp8.groupby(by=['Year','Week'])['Total_Sales_Amount'].
sum().reset_index()
```

Product #8 Weekly: Weekly Trend

In [447]:
```
fig = go.Figure(data=[go.Scatter(x=ds_weekly_P8.index,y=ds_weekly_P8.T
otal_Sales_Amount)])
fig.update_layout(xaxis_title="Week",yaxis_title="Total Sales Amount",
title='Product #8: Weekly Trend',height=400,template='ggplot2')
fig.show()
```



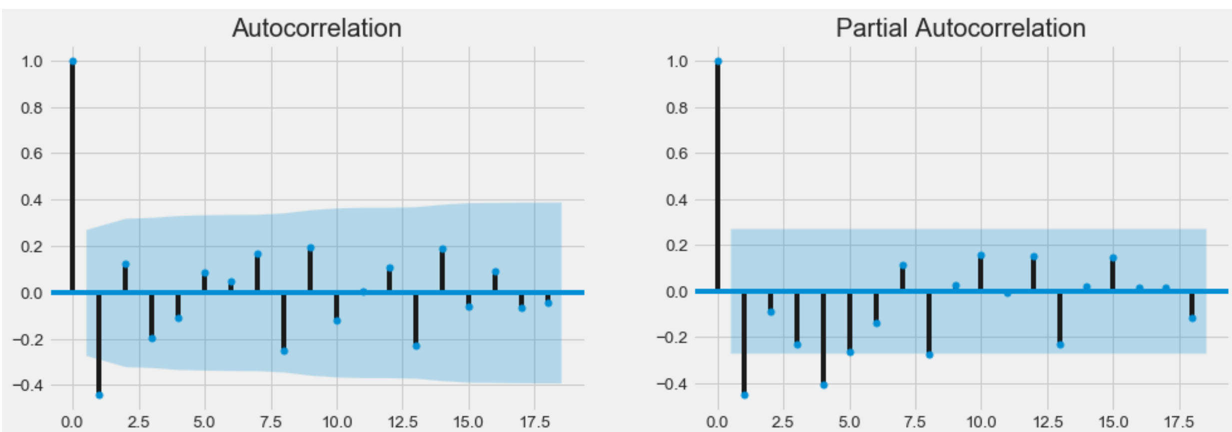
Product #8 Weekly: Test of Stationarity with 1 differencing of series

In [448]:
```
d=1
print('**************************Week*********************************')
series_8 = ds_weekly_P8.Total_Sales_Amount.diff(d)
series_8 = series_8.dropna()
output = adfuller(series_8)

print('ADF Statistic: {0:.2f} and P value:{1:.5f}'.format(*output))
print("As we can see the p value is close to zero which is less than .05 hence we reject null hypothesis")
```

```
**************************Week********************************
ADF Statistic: -5.90 and P value:0.00000
As we can see the p value is close to zero which is less than .05 he
nce we reject null hypothesis
```

Product #8 Weekly: PACF & ACF

In [449]:
```
fig, ax = plt.subplots(1,2,figsize=(15,5))
plot_acf(series_8, ax=ax[0])
plot_pacf(series_8, ax=ax[1])
plt.show()
```



Product #8 Weekly: Train & Test Split

In [450]:
```
series_8=ds_weekly_P8.Total_Sales_Amount
split_time = 42
time=np.arange(len(ds_weekly_P8))
xtrain=series_8[:split_time]
xtest=series_8[split_time:]
timeTrain = time[:split_time]
timeTest = time[split_time:]
```

Product #8 Weekly: ARIMA Model

In [451]:
```
model = ARIMA(xtrain, order=(1,1,0))
model_fit = model.fit()
print(model_fit.summary())
```

```
                              ARIMA Model Results
==============================================================================
============
Dep. Variable:     D.Total_Sales_Amount   No. Observations:
41
Model:                   ARIMA(1, 1, 0)   Log Likelihood
-269.114
Method:                         css-mle   S.D. of innovations
170.854
Date:                  Wed, 06 Jan 2021   AIC
544.229
Time:                          05:09:57   BIC
549.369
Sample:                               1   HQIC
546.101

==============================================================================
==========================
                                 coef    std err          z      P>|
z|      [0.025      0.975]
------------------------------------------------------------------------------
--------------------------
const                          6.2318     17.590      0.354      0.7
25     -28.245      40.708
ar.L1.D.Total_Sales_Amount    -0.5299      0.129     -4.098      0.0
00      -0.783      -0.276
                                    Roots
==============================================================================
=========
                  Real          Imaginary           Modulus
Frequency
------------------------------------------------------------------------------
---------
AR.1           -1.8872           +0.0000j            1.8872
0.5000
------------------------------------------------------------------------------
---------
```

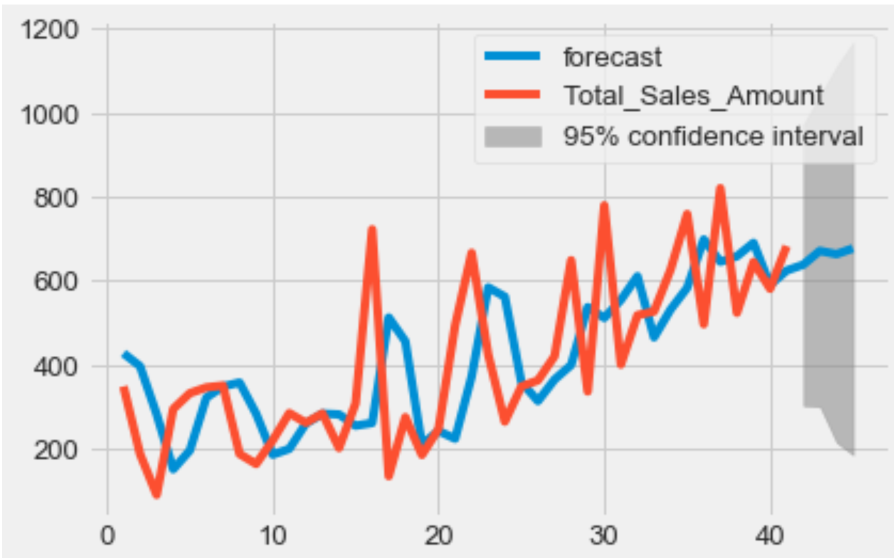
Product #8 Weekly: Weekly Trend and Forecast

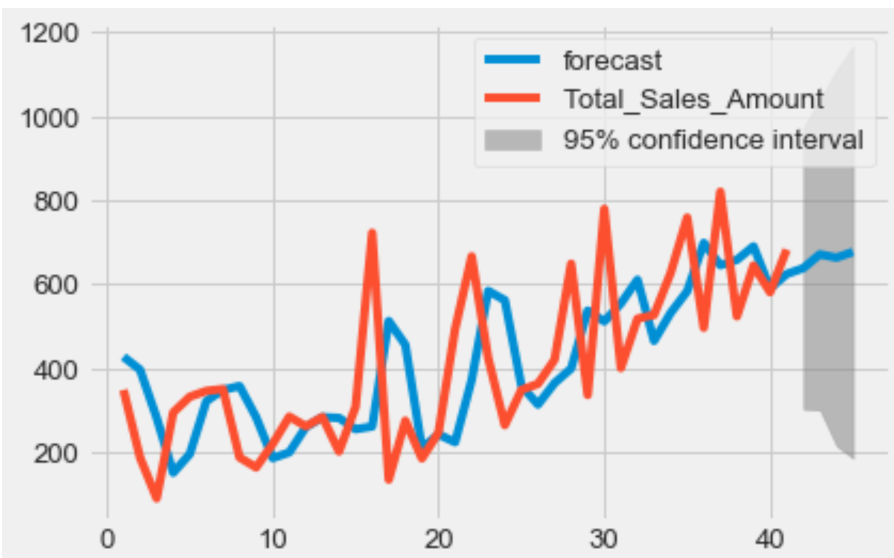
In [452]:

```
ytrain_pred = model_fit.predict()
ytest_pred = model_fit.predict(start=min(timeTest),end=max(timeTest),d
ynamic=True)
print('RMSE Train :',np.sqrt(np.mean((ytrain_pred - xtrain)**2)))
print('RMSE Test :',np.sqrt(np.mean((ytest_pred - xtest)**2)))
forecast = model_fit.forecast(20, alpha=0.05)
model_fit.plot_predict(1,45)
```

```
RMSE Train : 459.59555356394054
RMSE Test : 494.4852514241298
```

Out[452]:





In [453]:
```
model_fit.forecast(30)[0]
```

Out[453]:
```
array([638.31091107, 671.29661663, 663.35218386, 677.09563614,
       679.34720549, 687.68804417, 692.8023271 , 699.62628346,
       705.54432545, 711.94238943, 718.08610143, 724.36458836,
       730.57166134, 736.81657483, 743.04143755, 749.27692469,
       755.5067822 , 761.73962272, 767.97088262, 774.20298005,
       780.43463369, 786.66652248, 792.89828668, 799.13011689,
       805.36191212, 811.59372589, 817.82552984, 824.05733899,
       830.28914538, 836.52095324])
```

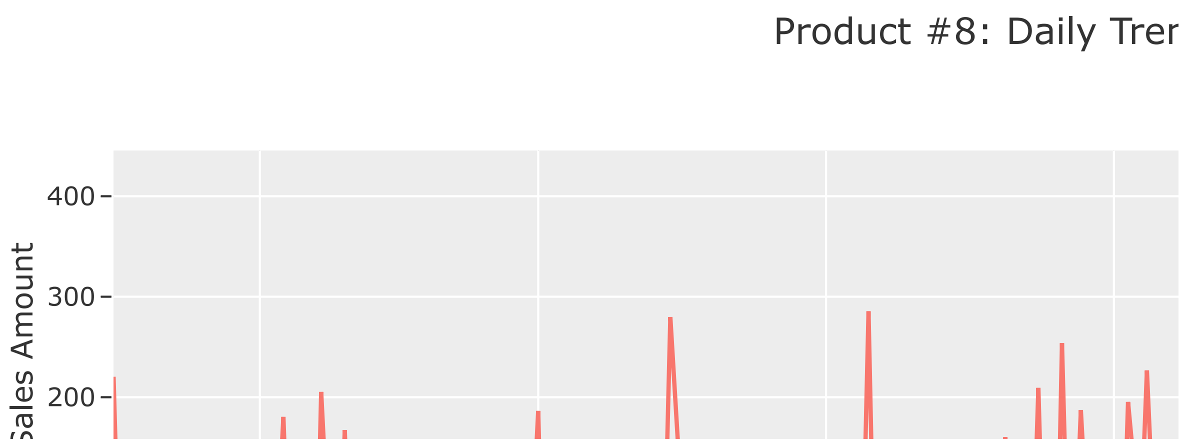
Based on my weekly forecast model we can see a upward trend after a small peak for this product. It seems the demand for this product is increasing over time. Based on forecasting I would suggest to have some inventory in stock due to forecasting and sales.

In [ ]:
```

```

Product #8 Daily: Daily Trend

In [454]:
```
ds_daily_P8 = dfp8.groupby(by=['Date'])['Total_Sales_Amount'].sum().re
set_index()
```

In [455]:
```
fig = go.Figure(data=[go.Scatter(x=ds_daily_P8.Date,y=ds_daily_P8.Tota
l_Sales_Amount)])
fig.update_layout(xaxis_title="Date",yaxis_title="Total Sales Amount",
title='Product #8: Daily Trend',height=400,template='ggplot2')
fig.show()
```



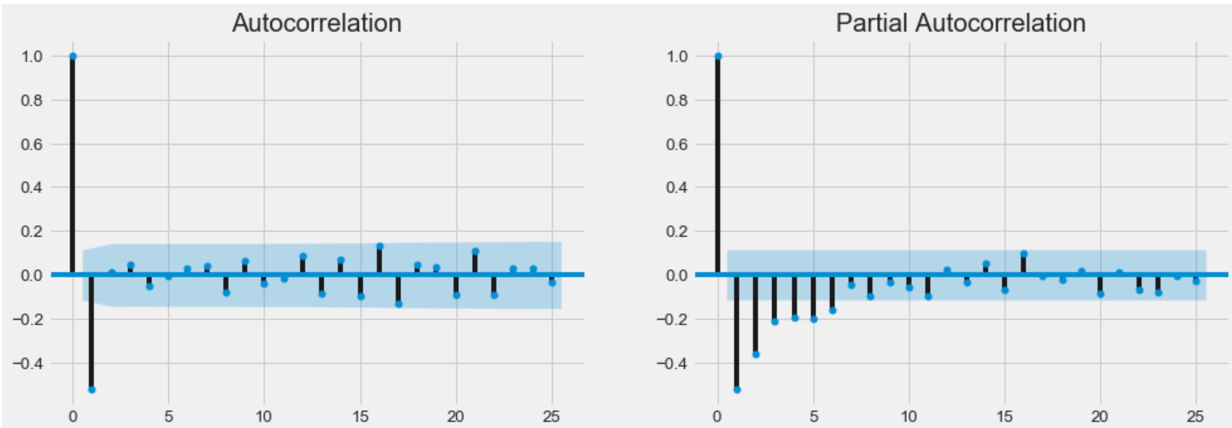
Product #8 Weekly: Test of Stationarity with 1 differencing of series

In [456]:
```
print('\n**************************Daily*****************************
****')
series_date_8 = ds_daily_P8.Total_Sales_Amount.diff(d)
series_date_8 = series_date_8.dropna()
output = adfuller(series_date_8)
print('ADF Statistic: {0:.2f} and P value:{1:.5f}'.format(*output))
print("As we can see the p value is close to zero which is less than .
05 hence we reject null hypothesis")
```

```
**************************Daily*********************************
ADF Statistic: -8.73 and P value:0.00000
As we can see the p value is close to zero which is less than .05 he
nce we reject null hypothesis
```

Product #8 Weekly: PACF & ACF

In [457]:
```
fig, ax = plt.subplots(1,2,figsize=(15,5))
plot_acf(series_date_8, ax=ax[0])
plot_pacf(series_date_8, ax=ax[1])
plt.show()
```



Product #8 Weekly: Train & Test Split

In [458]:
```
series_date_8=ds_daily_P8.Total_Sales_Amount
split_time = 245
time_d=np.arange(len(ds_daily_P8))
xtrain_d=series_date_8[:split_time]
xtest_d=series_date_8[split_time:]
timeTrain_d = time_d[:split_time]
timeTest_d = time_d[split_time:]
```

Product #8 Weekly: ARIMA Model

In [459]:
```
s_model = ARIMA(endog=xtrain_d , order=(1, 1, 0))
s_model_fit=s_model.fit()
print(s_model_fit.summary())
```

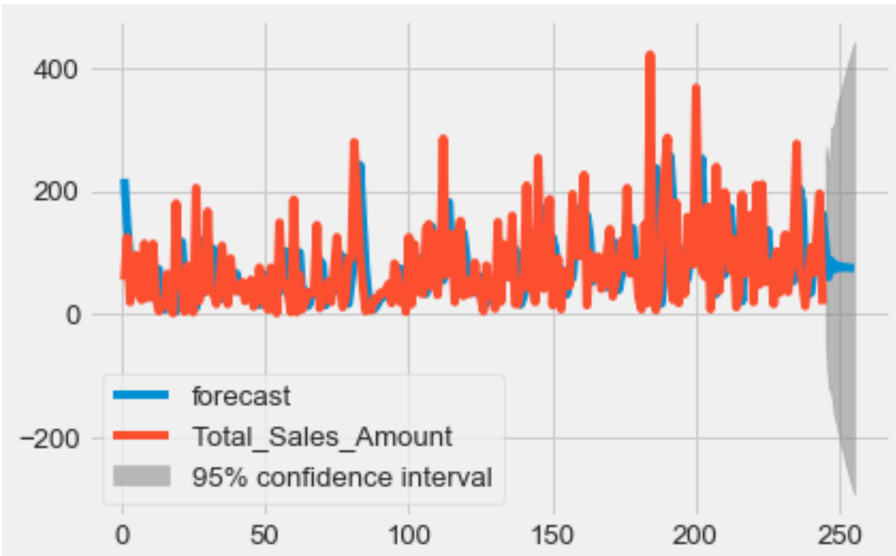
```
                              ARIMA Model Results
==============================================================================
============
Dep. Variable:     D.Total_Sales_Amount   No. Observations:
244
Model:                   ARIMA(1, 1, 0)   Log Likelihood
-1423.869
Method:                         css-mle   S.D. of innovations
82.756
Date:                  Wed, 06 Jan 2021   AIC
2853.738
Time:                          05:12:24   BIC
2864.230
Sample:                               1   HQIC
2857.964

==============================================================================
==========================
                                coef    std err          z      P>|
z|      [0.025      0.975]
------------------------------------------------------------------------------
--------------------------
const                        -0.3455      3.458     -0.100      0.9
20      -7.122       6.431
ar.L1.D.Total_Sales_Amount   -0.5346      0.055     -9.757      0.0
00      -0.642      -0.427
                                    Roots
==============================================================================
=========
                 Real          Imaginary           Modulus
Frequency
------------------------------------------------------------------------------
---------
AR.1            -1.8707           +0.0000j            1.8707
0.5000
------------------------------------------------------------------------------
---------
```

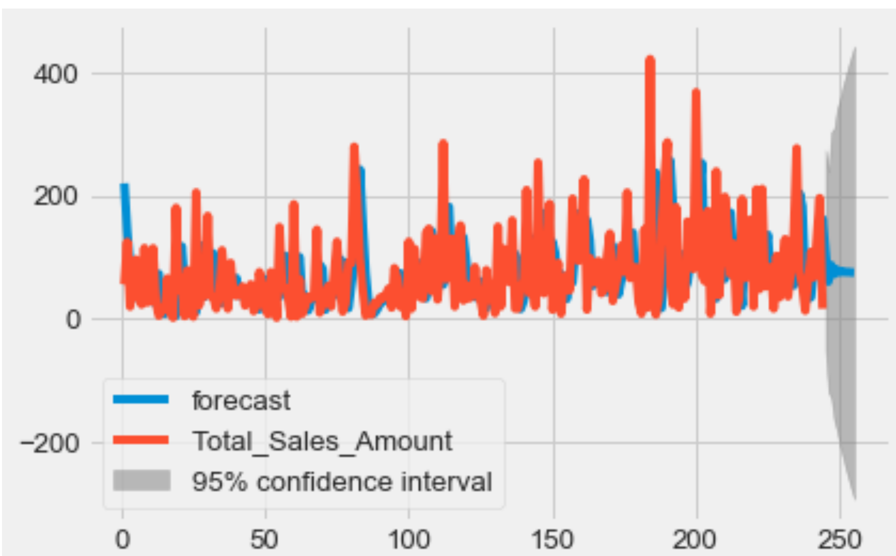
Product #8 Daily: Daily Trend and Forecast

In [463]:
```
ytrain_pred = s_model_fit.predict()
ytest_pred = s_model_fit.predict(start=min(timeTest_d),end=max(timeTes
t_d),dynamic=True)

print('RMSE Train :',np.sqrt(np.mean((ytrain_pred - xtrain_d)**2)))
print('RMSE Test :',np.sqrt(np.mean((ytest_pred - xtest_d)**2)))
forecast = s_model_fit.forecast(20, alpha=0.05)
s_model_fit.plot_predict(1,255)
```

```
RMSE Train : 113.98033250887431
RMSE Test : 91.1920147126132
```

Out[463]:





In [218]:
```
s_model_fit.forecast(30)[0]
```

Out[218]:
```
array([112.03545661,  60.43596998,  87.48879532,  72.49731737,
        79.98100124,  75.4503845 ,  77.34212059,  75.80073001,
        76.09454828,  75.40733909,  75.24454698,  74.80142299,
        74.50815294,  74.13477711,  73.80422255,  73.45077749,
        73.10956876,  72.76181899,  72.41756578,  72.07144346,
        71.72632029,  71.38066301,  71.03529125,  70.68976686,
        70.34432406,  69.99883765,  69.65337455,  69.30789899,
        68.96243008,  68.61695762])
```

Based on my daily forecast model we can see a up and downward trend on this product. In my opinion and based on the model and forecast, the company should buy or keep inventory for this product since there is a lot activity on product. The highest peak for this product happened in the August.

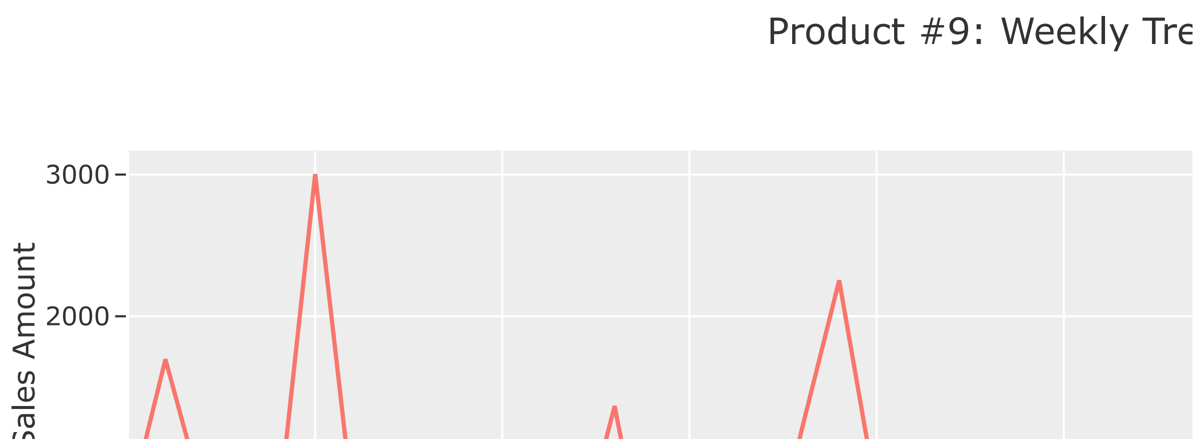
In [ ]:

Product #9: 'Lunch Bag Black Skull.'

In [464]:
```
ds_weekly_P9 = dfp9.groupby(by=['Year','Week'])['Total_Sales_Amount'].
sum().reset_index()
```

Product #9 Weekly: Weekly Trend

In [465]:
```
fig = go.Figure(data=[go.Scatter(x=ds_weekly_P9.index,y=ds_weekly_P9.T
otal_Sales_Amount)])
fig.update_layout(xaxis_title="Week",yaxis_title="Total Sales Amount",
title='Product #9: Weekly Trend',height=400,template='ggplot2')
fig.show()
```



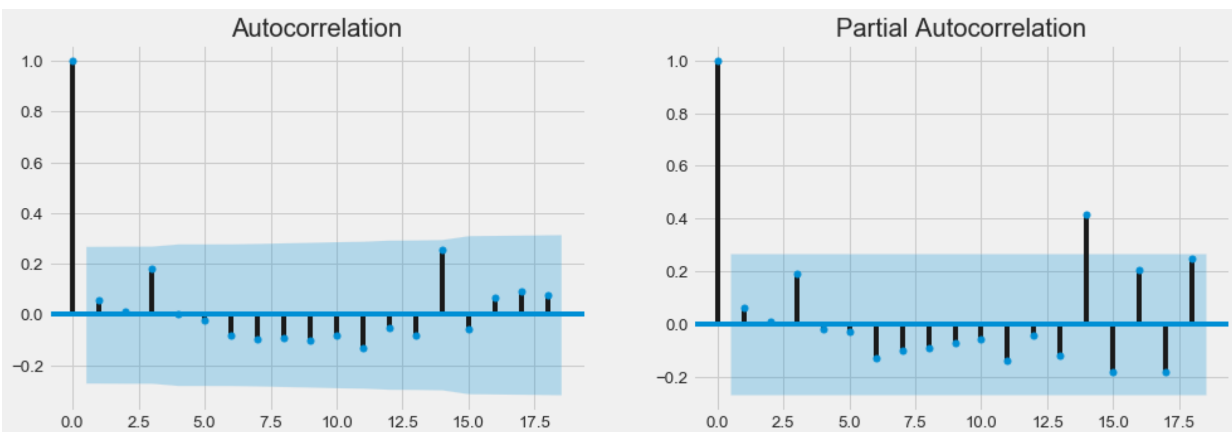
Product #9 Weekly: Test of Stationarity with 1 differencing of series

In [466]:
```
d=1
print('**************************Week*********************************
*')
series_9 = ds_weekly_P9.Total_Sales_Amount.diff(d)
series_9 = series_9.dropna()
output = adfuller(series_9)

print('ADF Statistic: {0:.2f} and P value:{1:.5f}'.format(*output))
print("As we can see the p value is close to zero which is less than .
05 hence we reject null hypothesis")
```

```
**************************Week********************************
ADF Statistic: -7.09 and P value:0.00000
As we can see the p value is close to zero which is less than .05 he
nce we reject null hypothesis
```

Product #9 Weekly: PACF & ACF

In [467]:
```
fig, ax = plt.subplots(1,2,figsize=(15,5))
plot_acf(series_1, ax=ax[0])
plot_pacf(series_1, ax=ax[1])
plt.show()
```



Product #9 Weekly: Train & Test Split

In [468]:
```
series_9=ds_weekly_P9.Total_Sales_Amount
split_time = 38
time=np.arange(len(ds_weekly_P9))
xtrain=series_9[:split_time]
xtest=series_9[split_time:]
timeTrain = time[:split_time]
timeTest = time[split_time:]
```

Product #9 Weekly: ARIMA Model

In [469]:
```
model = ARIMA(xtrain, order=(1,1,0))
model_fit = model.fit()
print(model_fit.summary())
```

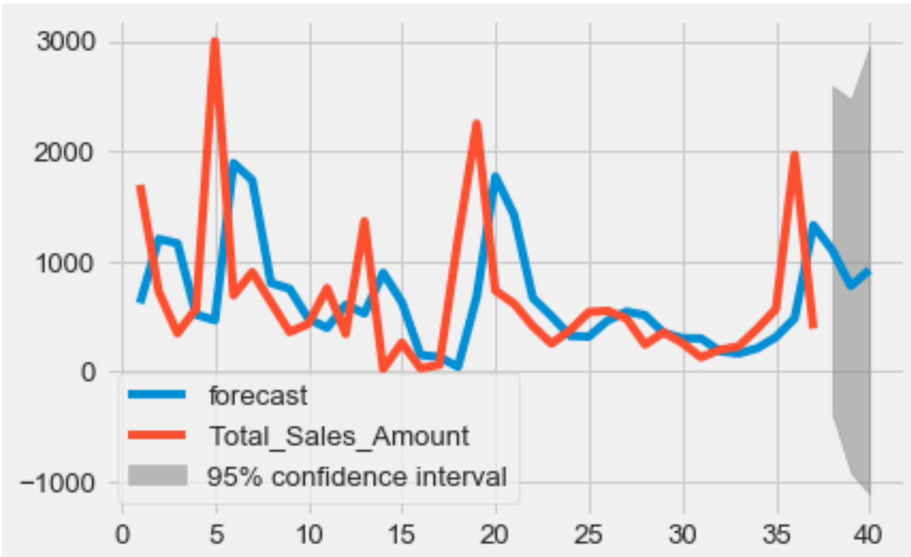
```
                              ARIMA Model Results
==============================================================================
============
Dep. Variable:     D.Total_Sales_Amount   No. Observations:
37
Model:                   ARIMA(1, 1, 0)   Log Likelihood
-298.107
Method:                         css-mle   S.D. of innovations
761.192
Date:                  Wed, 06 Jan 2021   AIC
602.214
Time:                          05:19:49   BIC
607.047
Sample:                               1   HQIC
603.918

==============================================================================
==========================
                                coef    std err          z      P>|
z|      [0.025      0.975]
------------------------------------------------------------------------------
--------------------------
const                         -2.0172     86.781     -0.023      0.9
82    -172.106     168.071
ar.L1.D.Total_Sales_Amount    -0.4544      0.156     -2.919      0.0
06      -0.760      -0.149
                                    Roots
==============================================================================
=========
                  Real          Imaginary           Modulus
Frequency
------------------------------------------------------------------------------
---------
AR.1           -2.2005           +0.0000j            2.2005
0.5000
------------------------------------------------------------------------------
---------
```

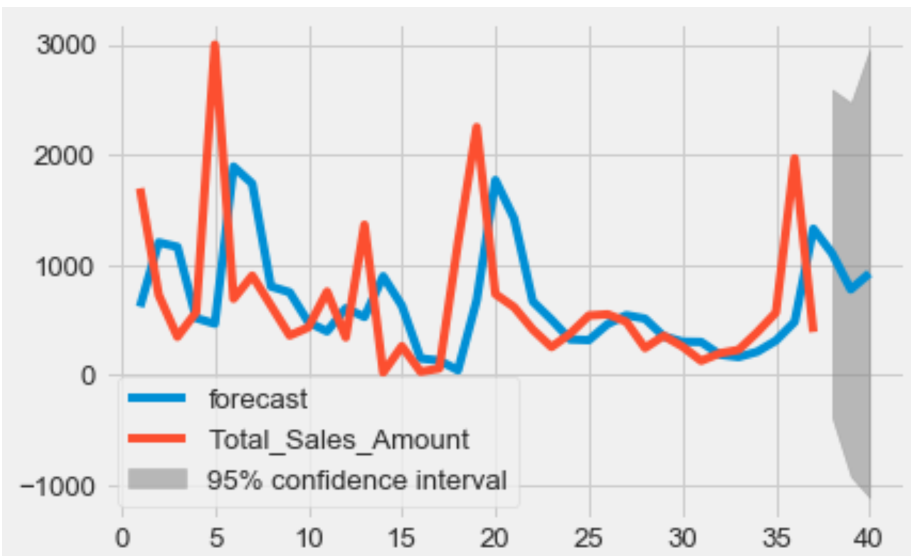
Product #9 Weekly: Weekly Trend and Forecast

In [472]:
```
ytrain_pred = model_fit.predict()
ytest_pred = model_fit.predict(start=min(timeTest),end=max(timeTest),d
ynamic=True)
print('RMSE Train :',np.sqrt(np.mean((ytrain_pred - xtrain)**2)))
print('RMSE Test :',np.sqrt(np.mean((ytest_pred - xtest)**2)))
forecast = model_fit.forecast(20, alpha=0.05)
model_fit.plot_predict(1,40)
```

```
RMSE Train : 999.925532981403
RMSE Test : 889.623574713618
```

Out[472]:





In [473]:
```
model_fit.forecast(30)[0]
```

Out[473]:
```
array([1105.76028591,  777.99419793,  924.00949184,  854.72070205,
        883.27418769,  867.36446252,  871.66049306,  866.77426806,
        866.06080541,  863.45108257,  861.7030917 ,  859.56349741,
        857.60186244,  855.55935606,  853.55360069,  851.53114428,
        849.51627745,  847.49796163,  845.48121317,  843.46375243,
        841.44661538,  839.42933124,  837.41211394,  835.39486626,
        833.37763239,  831.36039224,  829.34315495,  827.32591636,
        825.30867835,  823.29144009])
```

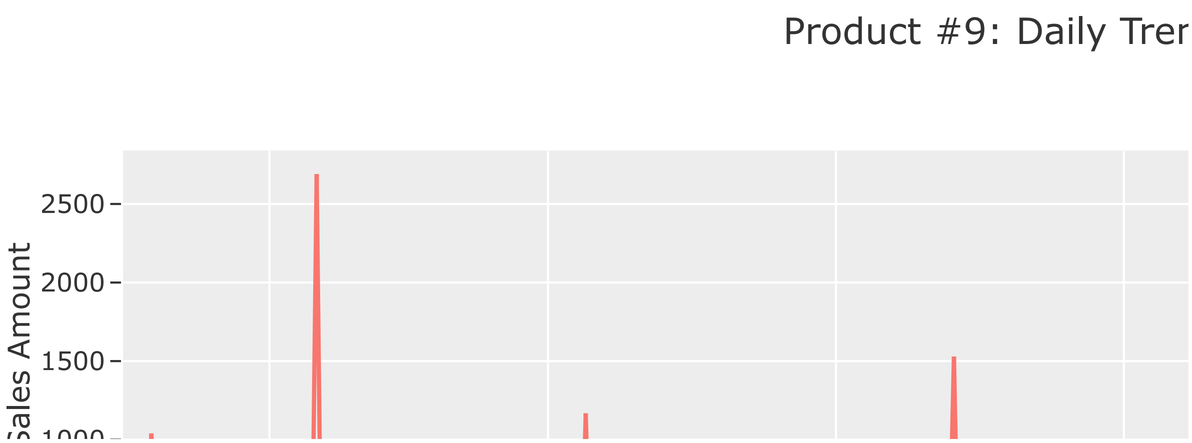
Based on my weekly forecast model we can see a slight downward trend for this product but it seems like is going to pick up by end of December. The forecast model fit looks close to the actual weekly sales which is good. Based on my forecast I would suggest to have some inventory by end of December.

In [ ]:
```

```

Product #9 Daily: Daily Trend

In [474]:
```
ds_daily_P9 = dfp9.groupby(by=['Date'])['Total_Sales_Amount'].sum().reset_index()
```

In [475]:
```
fig = go.Figure(data=[go.Scatter(x=ds_daily_P9.Date,y=ds_daily_P9.Tota
l_Sales_Amount)])
fig.update_layout(xaxis_title="Date",yaxis_title="Total Sales Amount",
title='Product #9: Daily Trend',height=400,template='ggplot2')
fig.show()
```



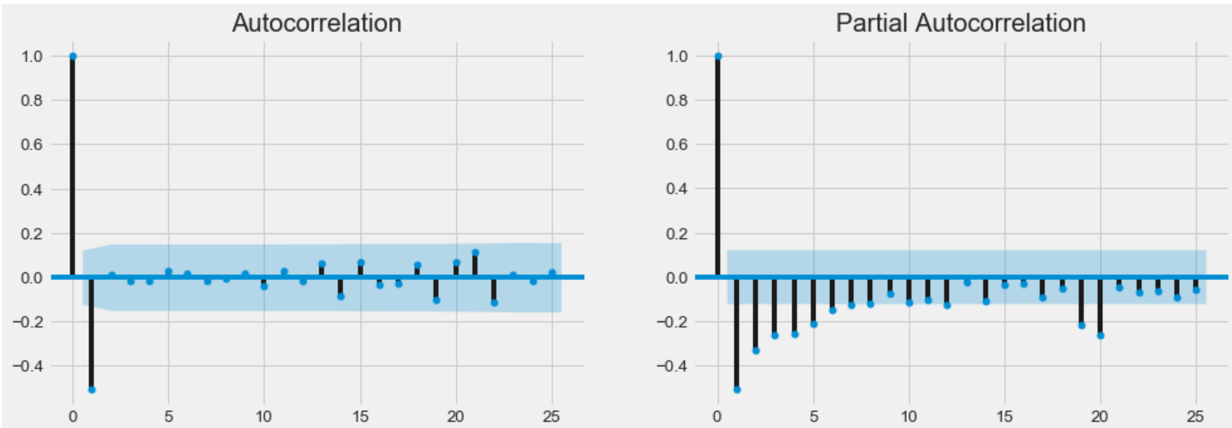
Product #9 Weekly: Test of Stationarity with 1 differencing of series

In [476]:
```
print('\n*************************Daily**********************************')
series_date_9 = ds_daily_P9.Total_Sales_Amount.diff(d)
series_date_9 = series_date_9.dropna()
output = adfuller(series_date_9)
print('ADF Statistic: {0:.2f} and P value:{1:.5f}'.format(*output))
print("As we can see the p value is close to zero which is less than .
05 hence we reject null hypothesis")
```

```
*************************Daily**********************************
ADF Statistic: -7.76 and P value:0.00000
As we can see the p value is close to zero which is less than .05 he
nce we reject null hypothesis
```

Product #9 Weekly: PACF & ACF

In [477]:
```
fig, ax = plt.subplots(1,2,figsize=(15,5))
plot_acf(series_date_9, ax=ax[0])
plot_pacf(series_date_9, ax=ax[1])
plt.show()
```



Product #9 Weekly: Train & Test Split

In [478]:
```
series_date_9=ds_daily_P9.Total_Sales_Amount
split_time = 206
time_d=np.arange(len(ds_daily_P9))
xtrain_d=series_date_9[:split_time]
xtest_d=series_date_9[split_time:]
timeTrain_d = time_d[:split_time]
timeTest_d = time_d[split_time:]
```

Product #9 Weekly: ARIMA Model

In [479]:
```
s_model = ARIMA(endog=xtrain_d , order=(1, 1, 0))
s_model_fit=s_model.fit()
print(s_model_fit.summary())
```

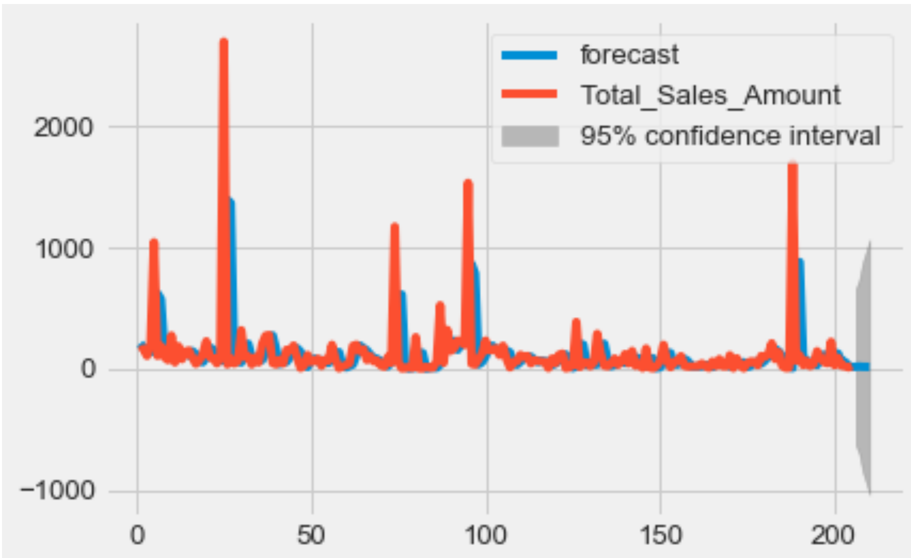
```
                             ARIMA Model Results
==============================================================================
============
Dep. Variable:     D.Total_Sales_Amount   No. Observations:
205
Model:                   ARIMA(1, 1, 0)   Log Likelihood
-1478.011
Method:                         css-mle   S.D. of innovations
327.069
Date:                  Wed, 06 Jan 2021   AIC
2962.023
Time:                          05:27:58   BIC
2971.992
Sample:                               1   HQIC
2966.055

==============================================================================
==========================
                              coef    std err          z      P>|
z|      [0.025      0.975]
------------------------------------------------------------------------------
--------------------------
const                      -0.8202     15.253     -0.054      0.9
57     -30.715      29.074
ar.L1.D.Total_Sales_Amount    -0.5001      0.060     -8.315      0.0
00      -0.618      -0.382
                                    Roots
==============================================================================
=========
                  Real          Imaginary           Modulus
Frequency
------------------------------------------------------------------------------
---------
AR.1           -1.9995           +0.0000j            1.9995
0.5000
------------------------------------------------------------------------------
---------
```

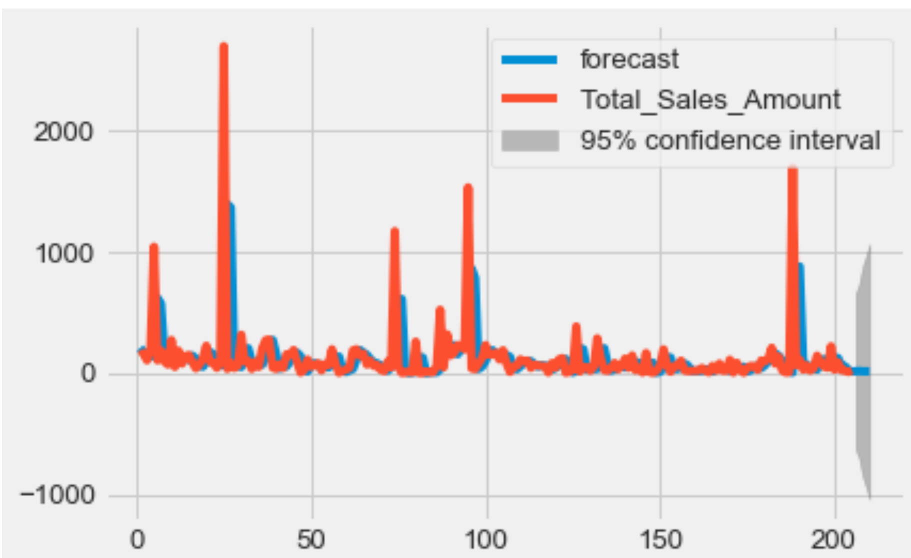
Product #9 Daily: Daily Trend and Forecast

In [483]:
```
ytrain_pred = s_model_fit.predict()
ytest_pred = s_model_fit.predict(start=min(timeTest_d),end=max(timeTes
t_d),dynamic=True)

print('RMSE Train :',np.sqrt(np.mean((ytrain_pred - xtrain_d)**2)))
print('RMSE Test :',np.sqrt(np.mean((ytest_pred - xtest_d)**2)))
forecast = s_model_fit.forecast(20, alpha=0.05)
s_model_fit.plot_predict(1,210)
```

```
RMSE Train : 349.11737276846293
RMSE Test : 276.9159975138687
```

Out[483]:





In [234]:
```
s_model_fit.forecast(30)[0]
```

Out[234]:
```
array([15.26292623, 17.94705236, 15.37424669, 15.43057301, 14.172000
21,
       13.57104358, 12.64119554, 11.87583487, 11.02820968, 10.221727
17,
        9.39466811,  8.57789993,  7.75598501,  6.93664411,  6.116015
87,
        5.29603146,  4.47572506,  3.6555797 ,  2.83535379,  2.015168
17,
        1.1949624 ,  0.37476671, -0.44543403, -1.26563224, -2.085831
71,
       -2.90603055, -3.72622971, -4.54642871, -5.36662779, -6.186826
83])
```

Based on my daily forecast model we see a slight downward trend for this product. We can see the big peak is happening in Jan, Sep, and May.

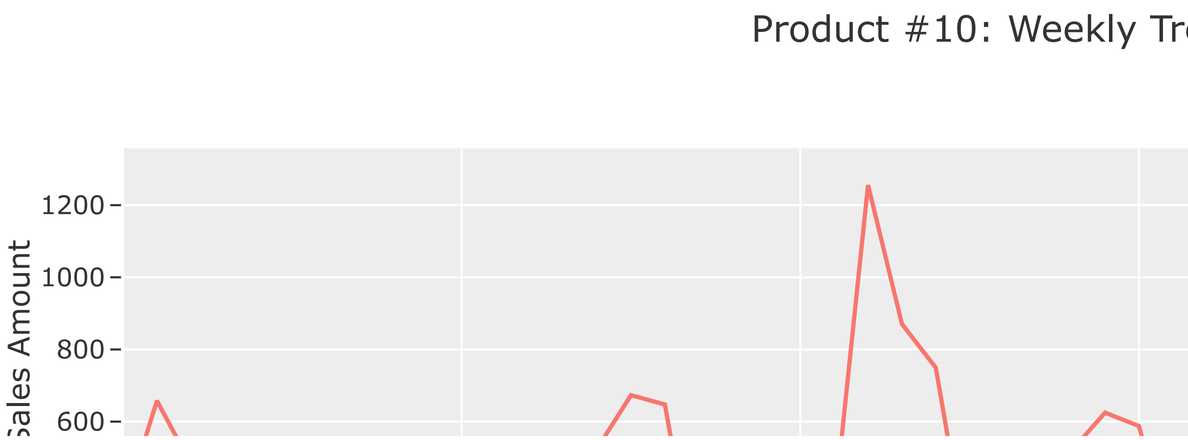
In [ ]:

Product #10: 'Heart of Wicker Small'

In [492]:
```
ds_weekly_P10 = dfp10.groupby(by=['Year','Week'])['Total_Sales_Amount'
].sum().reset_index()
```

Product #10 Weekly: Weekly Trend

In [493]:
```
fig = go.Figure(data=[go.Scatter(x=ds_weekly_P10.index,y=ds_weekly_P10
.Total_Sales_Amount)])
fig.update_layout(xaxis_title="Weeks in a Year",yaxis_title="Total Sal
es Amount",title='Product #10: Weekly Trend',height=400,template='ggpl
ot2')
fig.show()
```



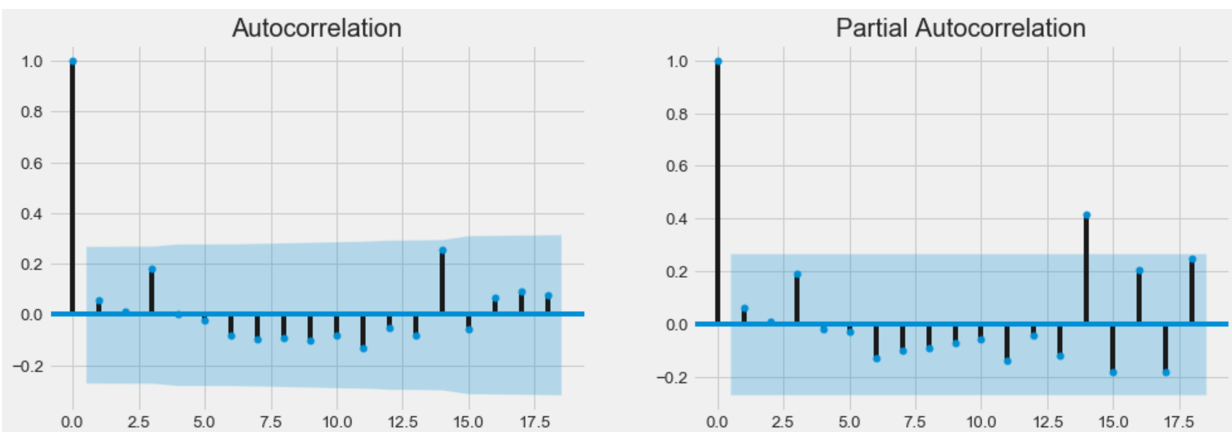
Product #10 Weekly: Test of Stationarity with 1 Differencing of Series

In [494]:
```
d=1
print('**************************Week*******************************
*')
series_10 = ds_weekly_P10.Total_Sales_Amount.diff(d)
series_10 = series_1.dropna()
output = adfuller(series_10)

print('ADF Statistic: {0:.2f} and P value:{1:.5f}'.format(*output))
print("As we can see the p value is close to zero which is less than .
05 hence we reject null hypothesis ")
```

```
**************************Week*******************************
ADF Statistic: -6.69 and P value:0.00000
As we can see the p value is close to zero which is less than .05 he
nce we reject null hypothesis
```

Product #10 Weekly: PACF & ACF

In [495]:
```
fig, ax = plt.subplots(1,2,figsize=(15,5))
plot_acf(series_1, ax=ax[0])
plot_pacf(series_1, ax=ax[1])
plt.show()
```



Product #10 Weekly: Train & Test Split

In [496]:
```
series_10=ds_weekly_P10.Total_Sales_Amount
split_time = 45
time=np.arange(len(ds_weekly_P10))
xtrain=series_10[:split_time]
xtest=series_10[split_time:]
timeTrain = time[:split_time]
timeTest = time[split_time:]
```

Product #10 Weekly: ARIMA Model

In [497]:
```
model = ARIMA(xtrain, order=(1,1,0))
model_fit = model.fit()
print(model_fit.summary())
```

```
                              ARIMA Model Results
==============================================================================
============
Dep. Variable:     D.Total_Sales_Amount   No. Observations:
44
Model:                   ARIMA(1, 1, 0)   Log Likelihood
-309.566
Method:                         css-mle   S.D. of innovations
274.471
Date:                  Wed, 06 Jan 2021   AIC
625.131
Time:                          05:34:05   BIC
630.484
Sample:                               1   HQIC
627.116

==============================================================================
==========================
                                 coef    std err          z      P>|
z|      [0.025      0.975]
------------------------------------------------------------------------------
--------------------------
const                          1.8034     30.132      0.060      0.9
53     -57.255      60.862
ar.L1.D.Total_Sales_Amount    -0.3823      0.139     -2.752      0.0
09      -0.655      -0.110
                                    Roots
==============================================================================
=========
                  Real          Imaginary           Modulus
Frequency
------------------------------------------------------------------------------
---------
AR.1           -2.6155           +0.0000j            2.6155
0.5000
------------------------------------------------------------------------------
---------
```

Product #10 Weekly: Weekly Trend and Forecast

In [498]:

```
ytrain_pred = model_fit.predict()
ytest_pred = model_fit.predict(start=min(timeTest),end=max(timeTest),d
ynamic=True)
print('RMSE Train :',np.sqrt(np.mean((ytrain_pred - xtrain)**2)))
print('RMSE Test :',np.sqrt(np.mean((ytest_pred - xtest)**2)))
forecast = model_fit.forecast(20, alpha=0.05)
model_fit.plot_predict(1,46)
```

```
RMSE Train : 597.4041969744953
RMSE Test : 704.3164520695769
```

Out[498]:

In [499]:
```
model_fit.forecast(30)[0]
```

Out[499]:
```
array([508.01441498, 529.86686661, 524.00498378, 528.7391161 ,
       529.42207636, 531.65392031, 533.29357992, 535.15964923,
       536.93915538, 538.75175719, 540.55170555, 542.3564917 ,
       544.15942822, 545.96307191, 547.76644522, 549.56992191,
       551.37335908, 553.17681135, 554.98025785, 556.78370656,
       558.58715442, 560.39060261, 562.19405067, 563.99749878,
       565.80094687, 567.60439497, 569.40784307, 571.21129116,
       573.01473926, 574.81818736])
```

Based on my weekly forecast we see a slight upward trend for this product. I would suggest the company to keep some inventory for this product since the forecast and trend is upward.

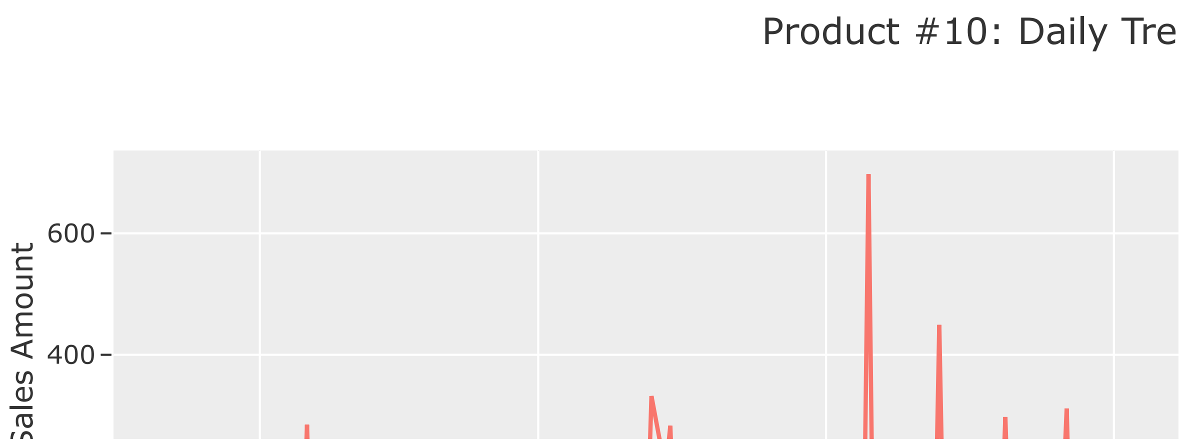
In [ ]:
```

```

Product #10: 'Heart of Wicker Small' Daily Time Series Forecasting

Product #10 Daily: Daily Trend

In [500]:
```
ds_daily_P10 = dfp10.groupby(by=['Date'])['Total_Sales_Amount'].sum().
reset_index()
```

In [501]:
```
fig = go.Figure(data=[go.Scatter(x=ds_daily_P10.Date,y=ds_daily_P10.To
tal_Sales_Amount)])
fig.update_layout(xaxis_title="Date",yaxis_title="Total Sales Amount",
title='Product #10: Daily Trend',height=400,template='ggplot2')
fig.show()
```



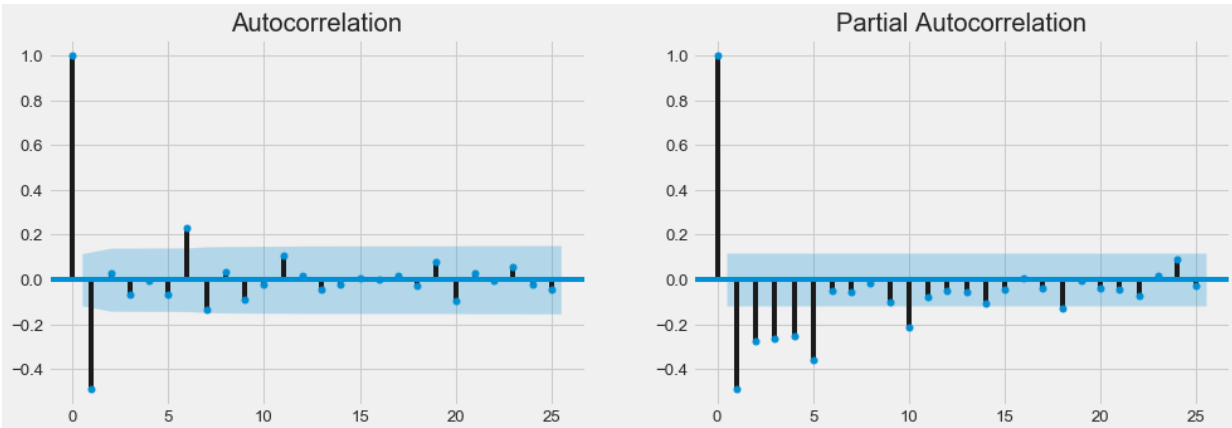
Product #10 Daily: Test of Stationarity with 1 differencing of series

In [502]:
```
print('\n**************************Daily******************************
****')
series_date_10 = ds_daily_P10.Total_Sales_Amount.diff(d)
series_date_10 = series_date_10.dropna()
output = adfuller(series_date_10)
print('ADF Statistic: {0:.2f} and P value:{1:.5f}'.format(*output))
print("As we can see the p value is close to zero which is less than .
05 hence we reject null hypothesis")
```

```
**************************Daily**********************************
ADF Statistic: -9.28 and P value:0.00000
As we can see the p value is close to zero which is less than .05 he
nce we reject null hypothesis
```

Product #10 Daily: PACF & ACF

In [503]:
```
fig, ax = plt.subplots(1,2,figsize=(15,5))
plot_acf(series_date_10, ax=ax[0])
plot_pacf(series_date_10, ax=ax[1])
plt.show()
```



Product #10 Daily: Train & Test Split

In [504]:
```
series_date_10=ds_daily_P10.Total_Sales_Amount
split_time = 228
time_d=np.arange(len(ds_daily_P10))
xtrain_d=series_date_10[:split_time]
xtest_d=series_date_10[split_time:]
timeTrain_d = time_d[:split_time]
timeTest_d = time_d[split_time:]
```

Product #10 Daily: ARIMA Model

In [505]:
```
s_model = ARIMA(endog=xtrain_d , order=(1, 1, 0))
s_model_fit=s_model.fit()
print(s_model_fit.summary())
```

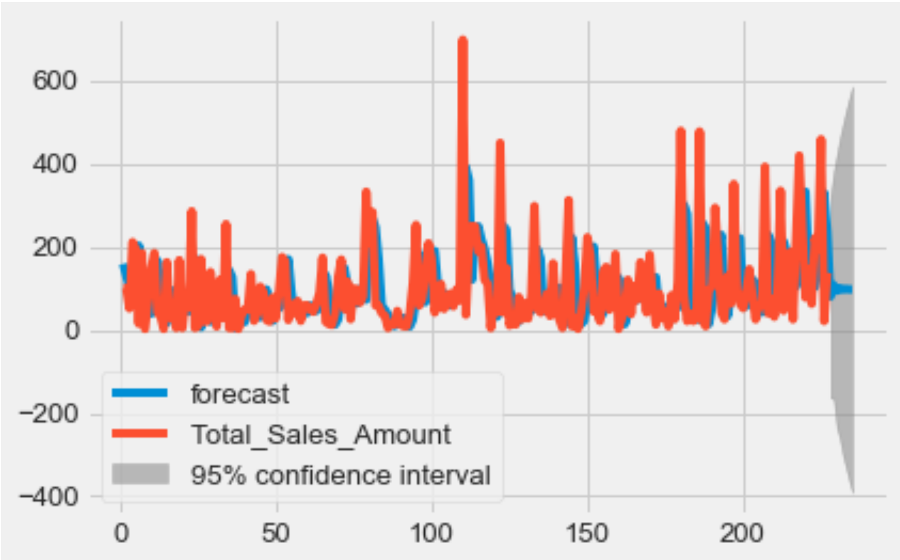
```
                              ARIMA Model Results
==============================================================================
Dep. Variable:     D.Total_Sales_Amount   No. Observations:
227
Model:                   ARIMA(1, 1, 0)   Log Likelihood
-1416.166
Method:                         css-mle   S.D. of innovations
123.850
Date:                  Wed, 06 Jan 2021   AIC
2838.332
Time:                          05:38:26   BIC
2848.607
Sample:                               1   HQIC
2842.478

==============================================================================
=========================
                              coef    std err          z      P>|
z|      [0.025      0.975]
------------------------------------------------------------------------------
-------------------------
const                      -0.1811      5.528     -0.033      0.9
74     -11.015      10.653
ar.L1.D.Total_Sales_Amount -0.4892      0.058     -8.473      0.0
00      -0.602      -0.376
                                    Roots
==============================================================================
=========
                  Real          Imaginary           Modulus
Frequency
------------------------------------------------------------------------------
---------
AR.1           -2.0441           +0.0000j            2.0441
0.5000
------------------------------------------------------------------------------
---------
```

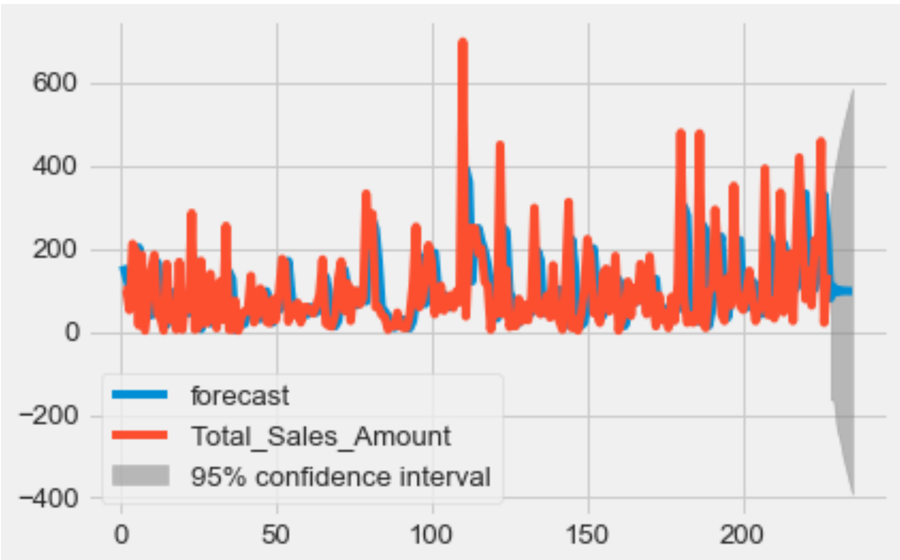
Product #10 Daily: Daily Trend and Forecast

In [510]:
```
ytrain_pred = s_model_fit.predict()
ytest_pred = s_model_fit.predict(start=min(timeTest_d),end=max(timeTest_d),dynamic=True)

print('RMSE Train :',np.sqrt(np.mean((ytrain_pred - xtrain_d)**2)))
print('RMSE Test :',np.sqrt(np.mean((ytest_pred - xtest_d)**2)))
forecast = s_model_fit.forecast(20, alpha=0.05)
s_model_fit.plot_predict(1,235)
```

```
RMSE Train : 159.1113618973634
RMSE Test : 151.29898111648194
```

Out[510]:





In [511]:
```
s_model_fit.forecast(20)[0]
```

Out[511]:
```
array([ 79.79613028, 107.42861641,  93.64060371, 100.11633929,
        96.67863815,  98.09079135,  97.13029743,  97.33055061,
        96.96294276,  96.87314405,  96.64743547,  96.48821675,
        96.29646984,  96.12063639,  95.93701775,  95.75720778,
        95.57553453,  95.39477284,  95.2135652 ,  95.03257572])
```

Based on my daily forecast model we can see a slight up and downward trend on this product. I still believe the company should have some inventory on this product. The highest peak for this product was on May probably due to Eastern holiday.

Conclusion

There is also a lot of scope in EDA, which can be tried in future analysis. After doing some EDA most of company sales mostly happened between Tuesday and Thursday, and the most number of transactions is done between 12 p.m. and 2 p.m. Most of the sales happens in UK, but Outside UK, Germany, France, and Ireland (EIRE) are the top business customers. Models which I have tried are not perfect and certainly there are room to improve the performance. Between weekly and daily models, I think my daily models perform better since I had more observation to train and test the models. As I analyze each product, most of sales happened close to the holidays. As mentioned earlier forecasting can very helpful in inventory management determining for determining which products trend is upward or downward. For example, products number 2, 4, 5, 8, and 10 had upward trend so I suggested that to purchase some inventory for those products to make we don't lose any customers.

In [ ]:
```

```