In [2]:

```
%matplotlib inline
import pandas as pd
import numpy as np
```

In [2]:

```
data = pd.read_excel("EDS_Branch.xlsx")
data.head(5)
```

Out[2]:

| | BillDate | Days | BranchName | BillNo | Createddate | Hours | HallNam |
|---|---|---|---|---|---|---|---|
| 0 | 2018-07-01 | Sunday | EDS Branch Hopes | 1 | 2018-07-01 07:49:37.000 | 7 | Swee Ha |
| 1 | 2018-07-01 | Sunday | EDS Branch Hopes | 1 | 2018-07-02 07:49:36.995 | 7 | Swee Ha |
| 2 | 2018-07-01 | Sunday | EDS Branch Hopes | 1 | 2018-07-03 07:49:36.995 | 7 | Swee Ha |
| 3 | 2018-07-01 | Sunday | EDS Branch Hopes | 2 | 2018-07-01 08:02:28.813 | 8 | Swee Ha |
| 4 | 2018-07-01 | Sunday | EDS Branch Hopes | 2 | 2018-07-01 08:02:28.813 | 8 | Swee Ha |

# Data Cleaning

In [3]:

```
data1 = data[["BillDate","Value"]]
data1.head(10)
```

Out[3]:

| | BillDate | Value |
|---|---|---|
| 0 | 2018-07-01 | 50.000 |
| 1 | 2018-07-01 | 249.375 |
| 2 | 2018-07-01 | 210.000 |
| 3 | 2018-07-01 | 192.500 |
| 4 | 2018-07-01 | 140.000 |
| 5 | 2018-07-01 | 115.000 |
| 6 | 2018-07-01 | 30.000 |
| 7 | 2018-07-01 | 30.000 |
| 8 | 2018-07-01 | 32.800 |
| 9 | 2018-07-01 | 118.750 |

In [4]:

```
data1['BillDate'] = pd.to_datetime(data1['BillDate'], format='%Y-
data1['BillDate'] = data1['BillDate'].dt.date
data1['BillDate'] = pd.to_datetime(data1['BillDate'], errors='coe
data1.head()
```

```
/usr/local/lib/python3.7/site-packages/ipykernel_lau
ncher.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice fro
m a DataFrame.
Try using .loc[row_indexer,col_indexer] = value inst
ead

See the caveats in the documentation: http://pandas.
pydata.org/pandas-docs/stable/user_guide/indexing.ht
ml#returning-a-view-versus-a-copy
(http://pandas.pydata.org/pandas-docs/stable/user_gu
ide/indexing.html#returning-a-view-versus-a-copy)
  """Entry point for launching an IPython kernel.
/usr/local/lib/python3.7/site-packages/ipykernel_lau
ncher.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice fro
```

```
m a DataFrame.
Try using .loc[row_indexer,col_indexer] = value inst
ead

See the caveats in the documentation: http://pandas.
pydata.org/pandas-docs/stable/user_guide/indexing.ht
ml#returning-a-view-versus-a-copy
(http://pandas.pydata.org/pandas-docs/stable/user_gu
ide/indexing.html#returning-a-view-versus-a-copy)

/usr/local/lib/python3.7/site-packages/ipykernel_lau
ncher.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice fro
m a DataFrame.
Try using .loc[row_indexer,col_indexer] = value inst
ead

See the caveats in the documentation: http://pandas.
pydata.org/pandas-docs/stable/user_guide/indexing.ht
ml#returning-a-view-versus-a-copy
(http://pandas.pydata.org/pandas-docs/stable/user_gu
ide/indexing.html#returning-a-view-versus-a-copy)
  This is separate from the ipykernel package so we
can avoid doing imports until
```

Out[4]:

| | BillDate | Value |
|---|---|---|
| 0 | 2018-07-01 | 50.000 |
| 1 | 2018-07-01 | 249.375 |
| 2 | 2018-07-01 | 210.000 |
| 3 | 2018-07-01 | 192.500 |
| 4 | 2018-07-01 | 140.000 |

In [7]:

```
data1.head(10)
```

Out[7]:

| | BillDate | Value |
|---|---|---|
| 0 | 2018-07-01 | 50.000 |
| 1 | 2018-07-01 | 249.375 |
| 2 | 2018-07-01 | 210.000 |
| 3 | 2018-07-01 | 192.500 |
| 4 | 2018-07-01 | 140.000 |
| 5 | 2018-07-01 | 115.000 |
| 6 | 2018-07-01 | 30.000 |
| 7 | 2018-07-01 | 30.000 |
| 8 | 2018-07-01 | 32.800 |
| 9 | 2018-07-01 | 118.750 |

In [8]:

```
data1.isna().sum()
```

Out[8]:

```
BillDate    0
Value       0
dtype: int64
```

In [9]:

```
data1.describe()
```

Out[9]:

| | Value |
|---|---|
| count | 442420.000000 |
| mean | 102.204138 |
| std | 275.697064 |
| min | 0.200000 |
| 25% | 41.000000 |
| 50% | 75.000000 |
| 75% | 116.250000 |
| max | 136500.000000 |

In [5]:

```
data2 = data1.groupby('BillDate')['Value'].sum().reset_index()
data2.head()
```

Out[5]:

| | BillDate | Value |
|---|---|---|
| 0 | 2018-07-01 | 124060.591 |
| 1 | 2018-07-02 | 64358.945 |
| 2 | 2018-07-03 | 70038.430 |
| 3 | 2018-07-04 | 68084.010 |
| 4 | 2018-07-05 | 80714.220 |

In [29]:

```
data2.count()
```

Out[29]:

```
BillDate    396
Value       396
dtype: int64
```

In [6]:

```
data2.nlargest(10, ['Value'])
```

Out[6]:

| | BillDate | Value |
|---|---|---|
| 124 | 2018-11-03 | 531305.015 |
| 123 | 2018-11-02 | 424899.835 |
| 126 | 2018-11-05 | 420702.985 |
| 125 | 2018-11-04 | 364015.735 |
| 62 | 2018-09-02 | 222505.255 |
| 57 | 2018-08-28 | 218958.670 |
| 391 | 2019-07-28 | 204500.986 |
| 384 | 2019-07-21 | 201396.977 |
| 61 | 2018-09-01 | 198219.040 |
| 377 | 2019-07-14 | 195715.580 |

In [7]:

```
from datetime import datetime
d1 = "2018-07-01"
d2 = "2019-08-01"
d1_1 = datetime.strptime(d1, "%Y-%m-%d")
d2_1 = datetime.strptime(d2, "%Y-%m-%d")
abs((d1_1-d2_1).days)
```

Out[7]:

396

In [8]:

```
data2 = data2.resample('1D', on='BillDate').mean().reset_index()
data2
```

Out[8]:

| | BillDate | Value |
|---|---|---|
| 0 | 2018-07-01 | 124060.591 |
| 1 | 2018-07-02 | 64358.945 |
| 2 | 2018-07-03 | 70038.430 |
| 3 | 2018-07-04 | 68084.010 |
| 4 | 2018-07-05 | 80714.220 |
| ... | ... | ... |
| 391 | 2019-07-27 | 177738.250 |
| 392 | 2019-07-28 | 204500.986 |
| 393 | 2019-07-29 | 107830.368 |
| 394 | 2019-07-30 | 126576.676 |
| 395 | 2019-07-31 | 131196.725 |

396 rows × 2 columns

In [9]:

```
data2['day_week'] = data2['BillDate'].dt.dayofweek
data2
```

Out[9]:

| | BillDate | Value | day_week |
|---|---|---|---|
| 0 | 2018-07-01 | 124060.591 | 6 |
| 1 | 2018-07-02 | 64358.945 | 0 |
| 2 | 2018-07-03 | 70038.430 | 1 |
| 3 | 2018-07-04 | 68084.010 | 2 |
| 4 | 2018-07-05 | 80714.220 | 3 |
| ... | ... | ... | ... |
| 391 | 2019-07-27 | 177738.250 | 5 |
| 392 | 2019-07-28 | 204500.986 | 6 |
| 393 | 2019-07-29 | 107830.368 | 0 |
| 394 | 2019-07-30 | 126576.676 | 1 |
| 395 | 2019-07-31 | 131196.725 | 2 |

396 rows × 3 columns

In [10]:

```
data2[data2.isna().any(axis=1)]
```

Out[10]:

| | BillDate | Value | day_week |
|---|---|---|---|
| 38 | 2018-08-08 | NaN | 2 |

In [10]:

```
data2[data2["day_week"]==2]
```

Out[10]:

| | BillDate | Value | day_week |
|---|---|---|---|
| 3 | 2018-07-04 | 68084.010 | 2 |
| 10 | 2018-07-11 | 74548.825 | 2 |
| 17 | 2018-07-18 | 76797.780 | 2 |
| 24 | 2018-07-25 | 70109.120 | 2 |
| 31 | 2018-08-01 | 74221.134 | 2 |
| 38 | 2018-08-08 | NaN | 2 |
| 45 | 2018-08-15 | 123470.605 | 2 |
| 52 | 2018-08-22 | 104626.485 | 2 |
| 59 | 2018-08-29 | 82487.775 | 2 |
| 66 | 2018-09-05 | 75402.590 | 2 |
| 73 | 2018-09-12 | 103570.726 | 2 |

In [11]:

```
data2.groupby('day_week')['Value'].mean()
```

Out[11]:

```
day_week
0     98358.634298
1    103612.778509
2     98216.076929
3    103373.065473
4    113821.672750
5    138893.000357
6    144978.426684
Name: Value, dtype: float64
```

In [11]:

```
data2["Value"] = data2.groupby('day_week')['Value'].transform(lam
```

In [11]:

```
data2[data2["day_week"]==2].head(7)
```

Out[11]:

| | BillDate | Value | day_week |
|---|---|---|---|
| 3 | 2018-07-04 | 68084.010000 | 2 |
| 10 | 2018-07-11 | 74548.825000 | 2 |
| 17 | 2018-07-18 | 76797.780000 | 2 |
| 24 | 2018-07-25 | 70109.120000 | 2 |
| 31 | 2018-08-01 | 74221.134000 | 2 |
| 38 | 2018-08-08 | 98216.076929 | 2 |
| 45 | 2018-08-15 | 123470.605000 | 2 |

In [12]:

```
data2.nlargest(10, ['Value'])
```

Out[12]:

| | BillDate | Value | day_week |
|---|---|---|---|
| 125 | 2018-11-03 | 531305.015 | 5 |
| 124 | 2018-11-02 | 424899.835 | 4 |
| 127 | 2018-11-05 | 420702.985 | 0 |
| 126 | 2018-11-04 | 364015.735 | 6 |
| 63 | 2018-09-02 | 222505.255 | 6 |
| 58 | 2018-08-28 | 218958.670 | 1 |
| 392 | 2019-07-28 | 204500.986 | 6 |
| 385 | 2019-07-21 | 201396.977 | 6 |
| 62 | 2018-09-01 | 198219.040 | 5 |
| 378 | 2019-07-14 | 195715.580 | 6 |

In [16]:

```
data2['Value'] = data2['Value'].mask(data2['Value'] > (300000) )
```

In [17]:

```
data2.nlargest(10, ['Value'])
```

Out[17]:

| | BillDate | Value | day_week |
|---|---|---|---|
| 63 | 2018-09-02 | 222505.255 | 6 |
| 58 | 2018-08-28 | 218958.670 | 1 |
| 392 | 2019-07-28 | 204500.986 | 6 |
| 385 | 2019-07-21 | 201396.977 | 6 |
| 62 | 2018-09-01 | 198219.040 | 5 |
| 378 | 2019-07-14 | 195715.580 | 6 |
| 384 | 2019-07-20 | 191041.914 | 5 |
| 123 | 2018-11-01 | 185600.850 | 3 |
| 363 | 2019-06-29 | 185372.150 | 5 |
| 357 | 2019-06-23 | 185323.505 | 6 |

In [19]:

```
data2.isnull().sum()
```

Out[19]:

```
BillDate    0
Value       4
day_week    0
dtype: int64
```

In [20]:

```
data2["Value"] = data2.groupby('day_week')['Value'].transform(lam
```

In [21]:

```
data2.isnull().sum()
```

Out[21]:

```
BillDate    0
Value       0
day_week    0
dtype: int64
```

In [22]:

```
data2.nlargest(10, ['Value'])
```

Out[22]:

| | BillDate | Value | day_week |
|---|---|---|---|
| 63 | 2018-09-02 | 222505.255 | 6 |
| 58 | 2018-08-28 | 218958.670 | 1 |
| 392 | 2019-07-28 | 204500.986 | 6 |
| 385 | 2019-07-21 | 201396.977 | 6 |
| 62 | 2018-09-01 | 198219.040 | 5 |
| 378 | 2019-07-14 | 195715.580 | 6 |
| 384 | 2019-07-20 | 191041.914 | 5 |
| 123 | 2018-11-01 | 185600.850 | 3 |
| 363 | 2019-06-29 | 185372.150 | 5 |
| 357 | 2019-06-23 | 185323.505 | 6 |

In [24]:

```
data2[data2["BillDate"]== ('2018-11-02')]
```

Out[24]:

| | BillDate | Value | day_week |
|---|---|---|---|
| 124 | 2018-11-02 | 108165.706164 | 4 |

In [34]:

```
data2.groupby('day_week')['Value'].mean()
```

Out[34]:

```
day_week
0     92602.485179
1    103612.778509
2     98216.076929
3    103373.065473
4    108165.706164
5    131758.236455
6    141067.046179
Name: Value, dtype: float64
```

In [35]:

```
data2[data2["day_week"]==1].head(20)
```

Out[35]:

| | BillDate | Value | day_week |
|---|---|---|---|
| 2 | 2018-07-03 | 70038.430 | 1 |
| 9 | 2018-07-10 | 78692.480 | 1 |
| 16 | 2018-07-17 | 84405.934 | 1 |
| 23 | 2018-07-24 | 110241.930 | 1 |
| 30 | 2018-07-31 | 76438.990 | 1 |
| 37 | 2018-08-07 | 98080.701 | 1 |
| 44 | 2018-08-14 | 80329.940 | 1 |
| 51 | 2018-08-21 | 96127.680 | 1 |
| 58 | 2018-08-28 | 218958.670 | 1 |
| 65 | 2018-09-04 | 88757.860 | 1 |
| 72 | 2018-09-11 | 72812.010 | 1 |
| 79 | 2018-09-18 | 72271.715 | 1 |
| 86 | 2018-09-25 | 60982.520 | 1 |
| 93 | 2018-10-02 | 116042.144 | 1 |
| 100 | 2018-10-09 | 69669.185 | 1 |
| 107 | 2018-10-16 | 91879.850 | 1 |
| 114 | 2018-10-23 | 69535.400 | 1 |
| 121 | 2018-10-30 | 121823.610 | 1 |
| 128 | 2018-11-06 | 133161.210 | 1 |
| 135 | 2018-11-13 | 63883.575 | 1 |

In [26]:

```
#data2.to_csv("Daily_Sales2.csv")
```

In [3]:

```
data2 = pd.read_csv("Daily_Sales2.csv")
```

#################################################################

# Data Analysis

In [4]:

```
import warnings
import matplotlib.pyplot as plt
warnings.filterwarnings("ignore")
plt.style.use('fivethirtyeight')
import statsmodels.api as sm
import matplotlib
```

In [5]:

```
from chart_studio.plotly import plot_mpl
from statsmodels.tsa.seasonal import seasonal_decompose
from plotly.offline import plot_mpl
```

In [6]:

```
matplotlib.rcParams['axes.labelsize'] = 14
matplotlib.rcParams['xtick.labelsize'] = 12
matplotlib.rcParams['ytick.labelsize'] = 12
matplotlib.rcParams['text.color'] = 'k'
```

In [7]:

```
data2
```

Out[7]:

| | Unnamed: 0 | BillDate | Value | day_week |
|---|---|---|---|---|
| 0 | 0 | 2018-07-01 | 124060.591 | 6 |
| 1 | 1 | 2018-07-02 | 64358.945 | 0 |
| 2 | 2 | 2018-07-03 | 70038.430 | 1 |
| 3 | 3 | 2018-07-04 | 68084.010 | 2 |
| 4 | 4 | 2018-07-05 | 80714.220 | 3 |
| ... | ... | ... | ... | ... |
| 391 | 391 | 2019-07-27 | 177738.250 | 5 |
| 392 | 392 | 2019-07-28 | 204500.986 | 6 |
| 393 | 393 | 2019-07-29 | 107830.368 | 0 |
| 394 | 394 | 2019-07-30 | 126576.676 | 1 |
| 395 | 395 | 2019-07-31 | 131196.725 | 2 |

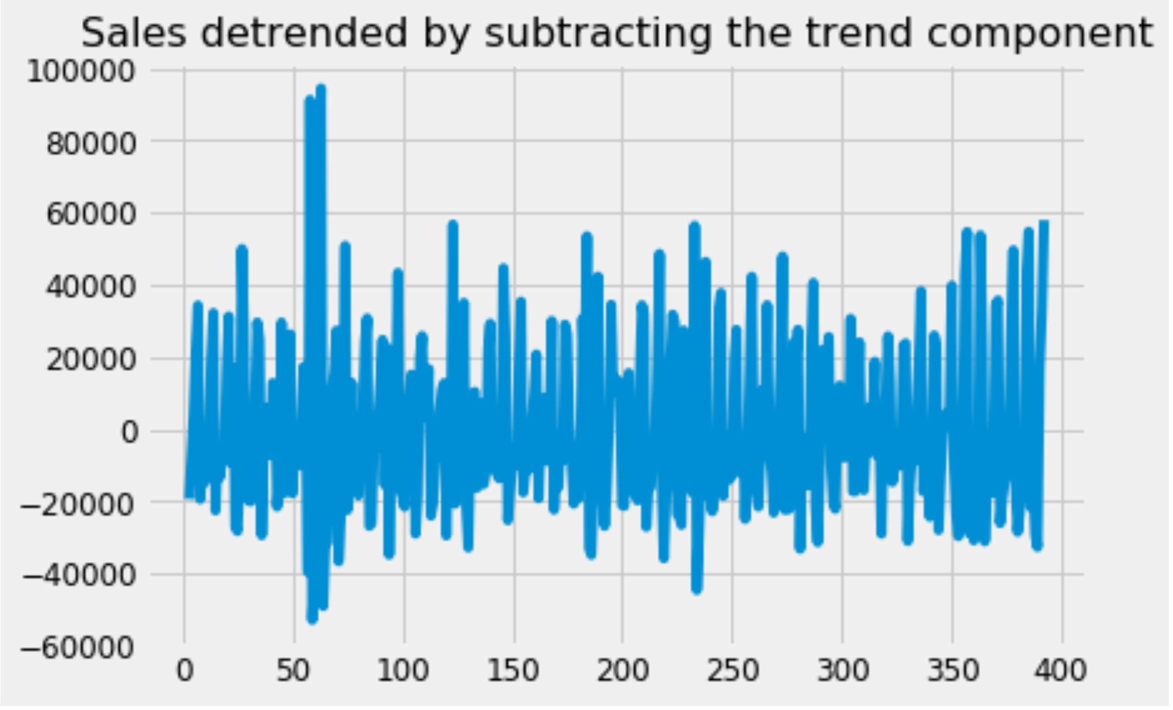
396 rows × 4 columns

In [15]:

```
result_mul = seasonal_decompose(data2['Value'], model='multiplica
detrended = data2.Value.values - result_mul.trend
plt.plot(detrended)
plt.title('Sales detrended by subtracting the trend component', f
```

Out[15]:

```
Text(0.5, 1.0, 'Sales detrended by subtracting the t
rend component')
```




In [29]:

```
data3 = data2[["BillDate","Value"]]
```

In [30]:

```
data3 = data3.set_index('BillDate')
data3.index
```

Out[30]:

```
DatetimeIndex(['2018-07-01', '2018-07-02', '2018-07-
03', '2018-07-04',
               '2018-07-05', '2018-07-06', '2018-07-
07', '2018-07-08',
               '2018-07-09', '2018-07-10',
               ...
               '2019-07-22', '2019-07-23', '2019-07-
24', '2019-07-25',
               '2019-07-26', '2019-07-27', '2019-07-
28', '2019-07-29',
               '2019-07-30', '2019-07-31'],
              dtype='datetime64[ns]', name='BillDate
', length=396, freq=None)
```

In [31]:

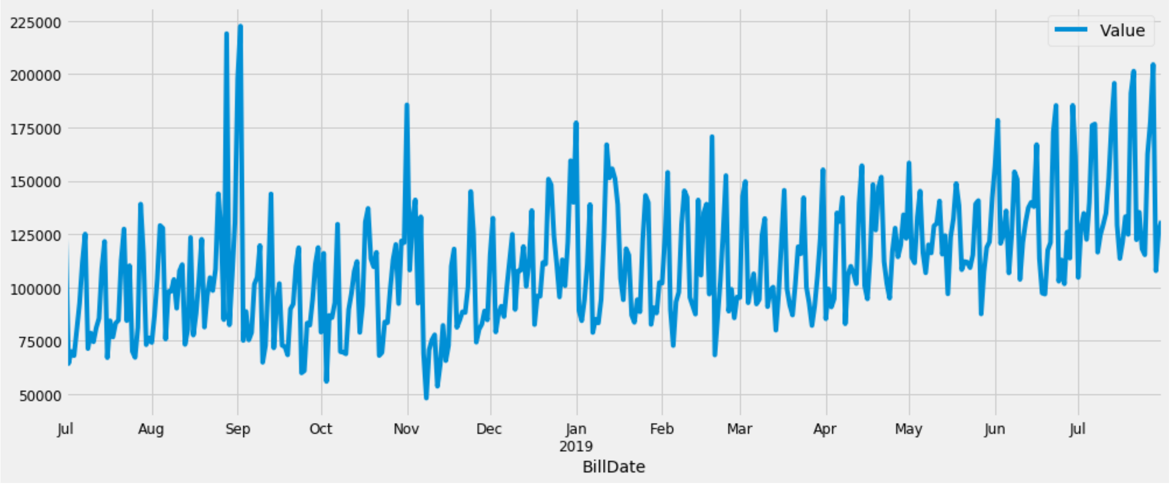
```
year_sales = data3['Value'].resample('MS').mean()
print(year_sales['2018':])
```

```
BillDate
2018-07-01     90753.347661
2018-08-01    105103.745772
2018-09-01     98366.006667
2018-10-01     97200.652774
2018-11-01     94177.810983
2018-12-01    111739.175726
2019-01-01    112206.810065
2019-02-01    111379.844714
2019-03-01    108786.581774
2019-04-01    117044.663467
2019-05-01    123073.983710
2019-06-01    133312.133400
2019-07-01    142825.586806
Freq: MS, Name: Value, dtype: float64
```

### Time Series Plot

In [32]:

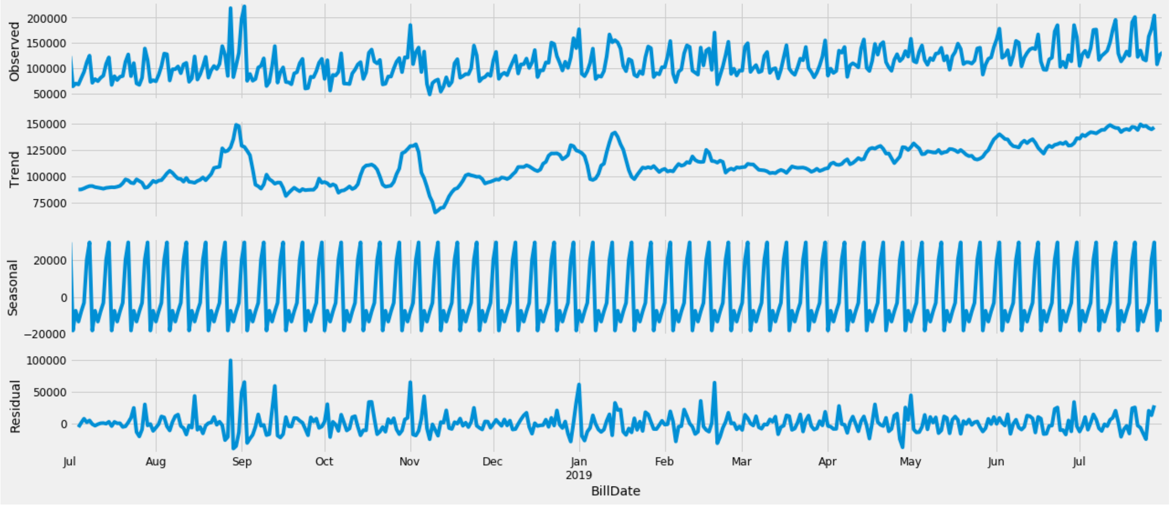
```
data3.plot(figsize=(15, 6))
plt.show()
```




## Observing Trend and Seasonality

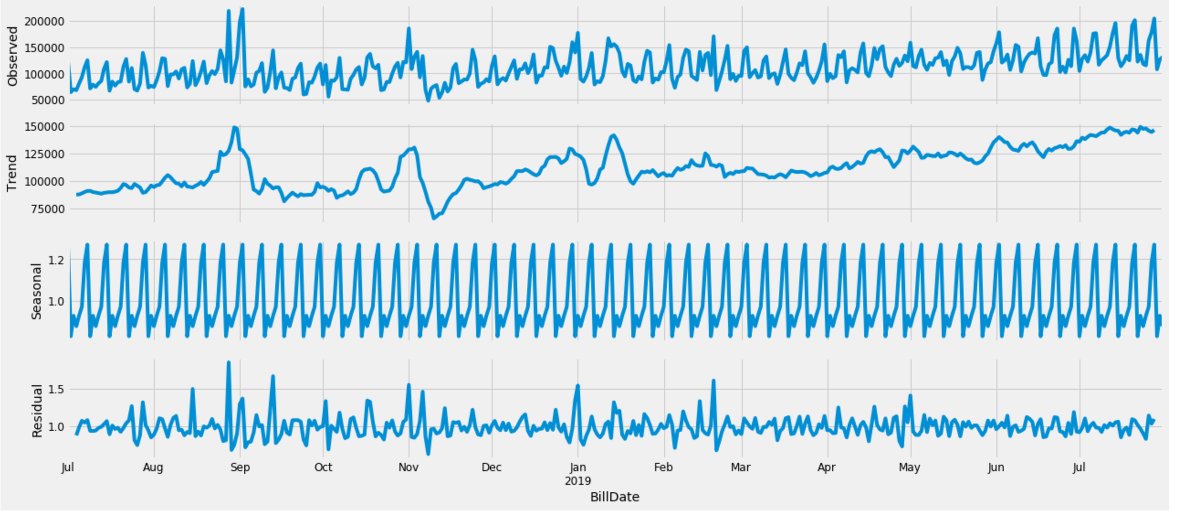
In [33]:

```
# Additive Seasonal_Decoposition
from pylab import rcParams
rcParams['figure.figsize'] = 18, 8
decomposition = sm.tsa.seasonal_decompose(data3, model='additive'
fig = decomposition.plot()
plt.show()
```



In [36]:

```
# Multiplicative Seasonal_Decoposition
from pylab import rcParams
rcParams['figure.figsize'] = 18, 8
decomposition = sm.tsa.seasonal_decompose(data3, model='multiplic
fig = decomposition.plot()
plt.show()
```




In [22]:

```
data2.head()
```

Out[22]:

| | BillDate | Value | day_week |
|---|---|---|---|
| 0 | 2018-07-01 | 124060.591 | 6 |
| 1 | 2018-07-02 | 64358.945 | 0 |
| 2 | 2018-07-03 | 70038.430 | 1 |
| 3 | 2018-07-04 | 68084.010 | 2 |
| 4 | 2018-07-05 | 80714.220 | 3 |

In [37]:

```
data2_1 = data2.set_index("BillDate")
data2_1.head()
```

Out[37]:

| | Value | day_week |
|---|---|---|
| **BillDate** | | |
| 2018-07-01 | 124060.591 | 6 |
| 2018-07-02 | 64358.945 | 0 |
| 2018-07-03 | 70038.430 | 1 |
| 2018-07-04 | 68084.010 | 2 |
| 2018-07-05 | 80714.220 | 3 |

In [38]:

```
# Required libraries for data visulization, normality test, and s
pd.set_option('display.float_format', lambda x: '%.4f' % x)
import seaborn as sns
sns.set_context("paper", font_scale=1.3)
sns.set_style('white')
import warnings
warnings.filterwarnings('ignore')
from time import time
import matplotlib.ticker as tkr
from scipy import stats
from statsmodels.tsa.stattools import adfuller
from sklearn import preprocessing
from statsmodels.tsa.stattools import pacf
```

## Normality Test

In [39]:

```
stat, p = stats.normaltest(data2.Value)
print('Statistics=%.3f, p=%.3f' % (stat, p))
alpha = 0.05
if p > alpha:
    print('Data looks Normally Distributed (Accept H0)')
else:
    print('Data does not look Normally Distributed (Reject H0)')
```

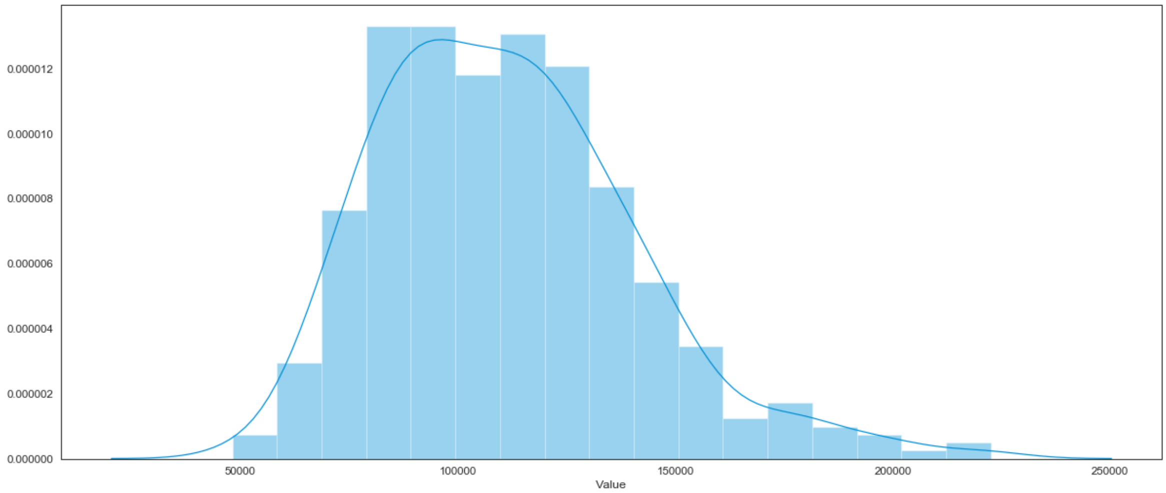
```
Statistics=39.268, p=0.000
Data does not look Normally Distributed (Reject H0)
```

### Kurtosis and Skewness

In [40]:

```
sns.distplot(data2.Value);
print( 'Kurtosis of normal distribution: {}'.format(stats.kurtosi
print( 'Skewness of normal distribution: {}'.format(stats.skew(da
```

```
Kurtosis of normal distribution: 0.8047300298703224
Skewness of normal distribution: 0.7665756612800276
```

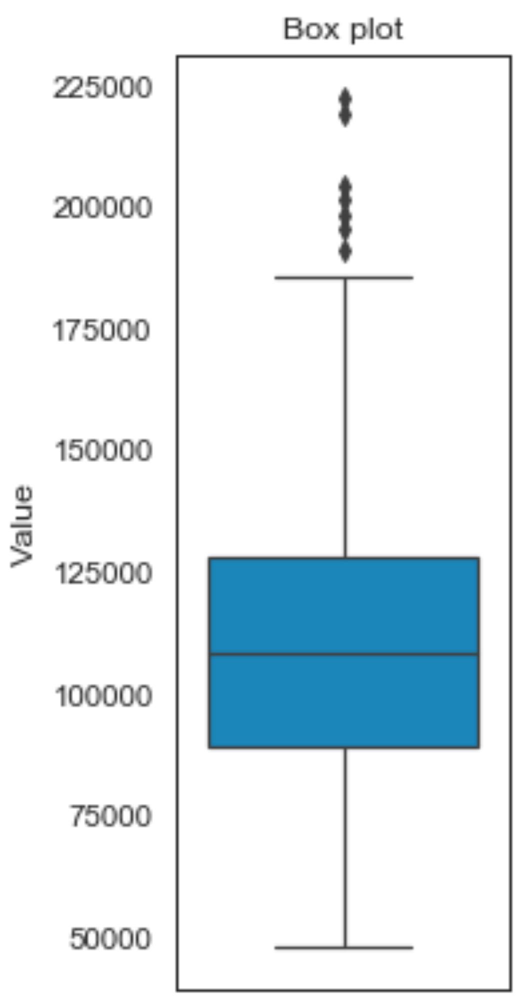


In [41]:

```
plt.figure(figsize=(2,6))
plt.subplot(1,1,1)
plt.subplots_adjust(wspace=0.2)
sns.boxplot(y="Value", data=data2)
plt.title('Box plot')
```

Out[41]:

```
Text(0.5, 1.0, 'Box plot')
```



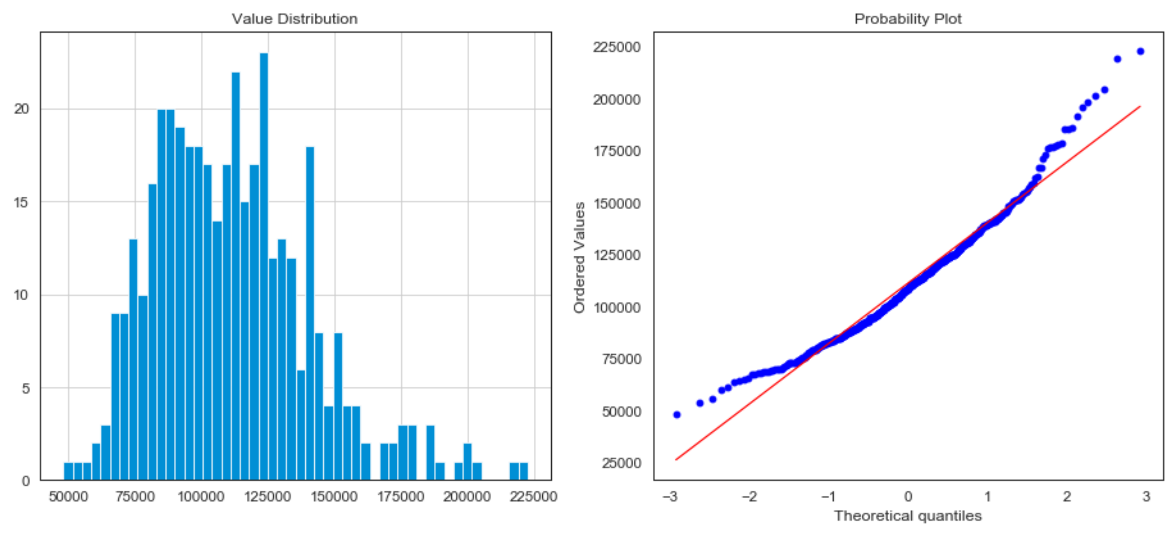

## Distribution and Normal probability plots

In [42]:

```
plt.figure(figsize=(14,6))
plt.subplot(1,2,1)
data3['Value'].hist(bins=50)
plt.title('Value Distribution')
plt.subplot(1,2,2)
stats.probplot(data3['Value'], plot=plt);
data3.describe().T
```

Out[42]:

| | count | mean | std | min | 25% | |
|---|---|---|---|---|---|---|
| Value | 396.0000 | 111232.4255 | 29425.4967 | 48189.8500 | 89019.5175 | 1084 |




## Average Value Over Day, Week, and Month

In [43]:

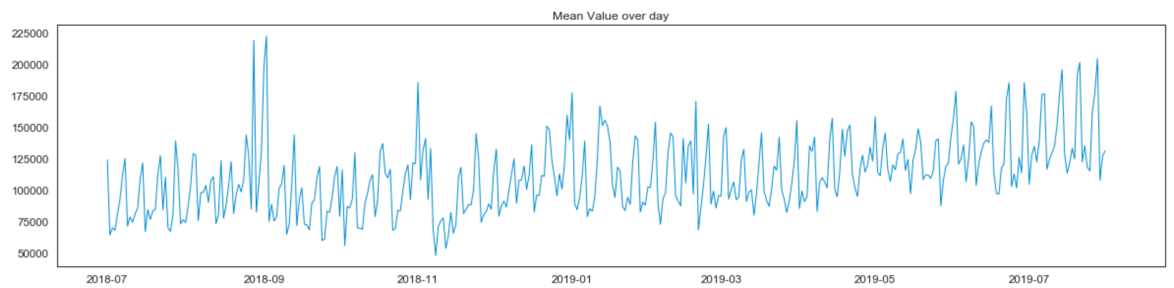
```
data4=data2.loc[:,['BillDate','Value']]
```

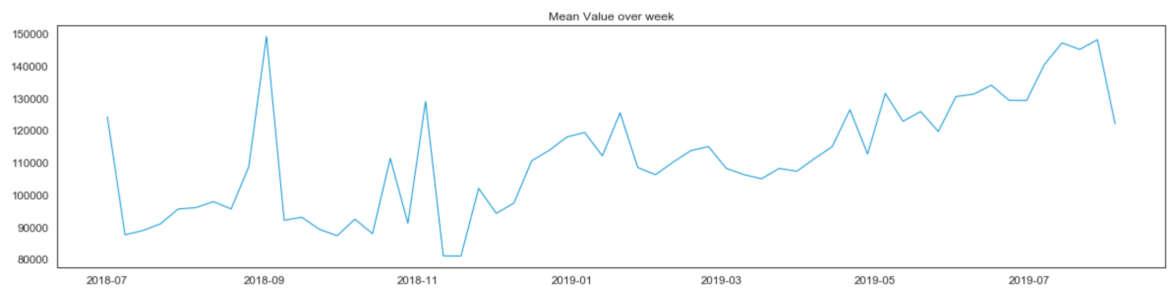
In [44]:

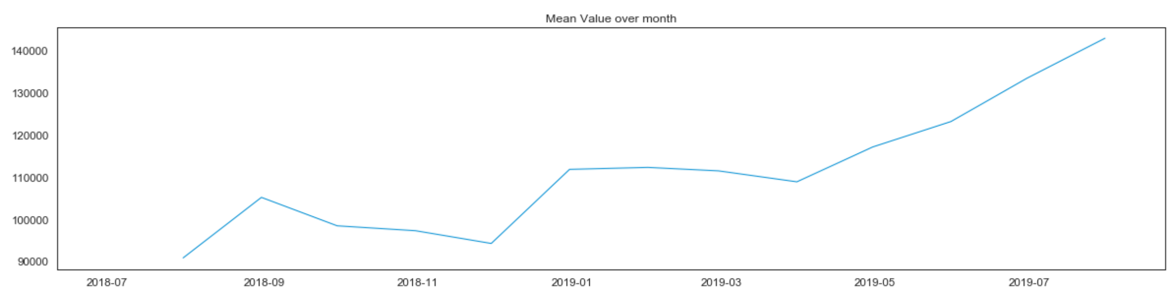
```
data4.set_index('BillDate',inplace=True)
```

In [45]:

```
fig = plt.figure(figsize=(18,16))
fig.subplots_adjust(hspace=.4)
ax1 = fig.add_subplot(3,1,1)
ax1.plot(data4['Value'].resample('D').mean(),linewidth=1)
ax1.set_title('Mean Value over day')
ax1.tick_params(axis='both', which='major')

ax2 = fig.add_subplot(3,1,2, sharex=ax1)
ax2.plot(data4['Value'].resample('W').mean(),linewidth=1)
ax2.set_title('Mean Value over week')
ax2.tick_params(axis='both', which='major')

ax3 = fig.add_subplot(3,1,3, sharex=ax1)
ax3.plot(data4['Value'].resample('M').mean(),linewidth=1)
ax3.set_title('Mean Value over month')
ax3.tick_params(axis='both', which='major')
```




## Checking whether Data is Stationary or not

## Dickey-Fuller test

In [46]:

```
data4.head()
```

Out[46]:

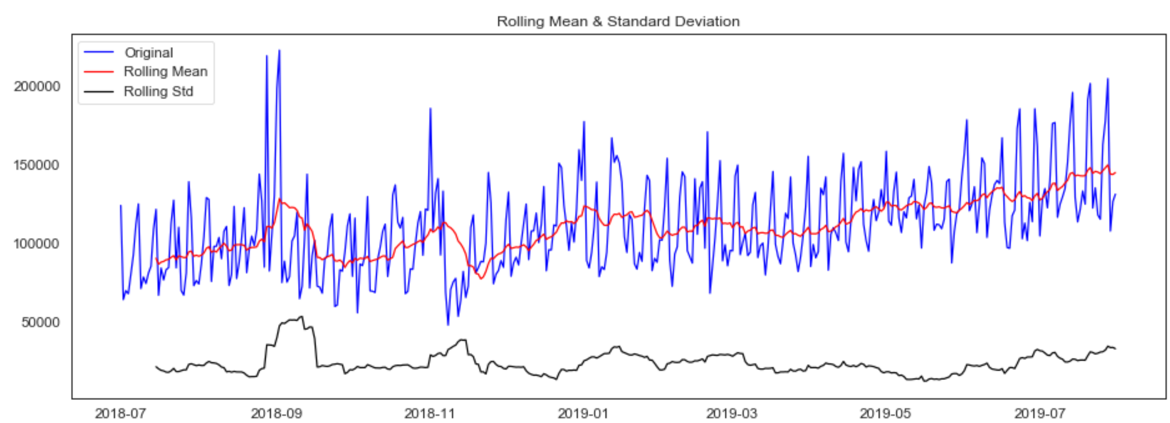
| | Value |
|---|---|
| **BillDate** | |
| 2018-07-01 | 124060.5910 |
| 2018-07-02 | 64358.9450 |
| 2018-07-03 | 70038.4300 |
| 2018-07-04 | 68084.0100 |
| 2018-07-05 | 80714.2200 |

In [47]:

```
import matplotlib.pyplot as plt
```

In [48]:

```
data5 =data4.resample('D', how=np.mean)

def test_stationarity(timeseries):
    rolmean = timeseries.rolling(window=15).mean()
    rolstd = timeseries.rolling(window=15).std()

    plt.figure(figsize=(14,5))
    sns.despine(left=True)
    orig = plt.plot(timeseries, color='blue',label='Original')
    mean = plt.plot(rolmean, color='red', label='Rolling Mean')
    std = plt.plot(rolstd, color='black', label = 'Rolling Std')

    plt.legend(loc='best'); plt.title('Rolling Mean & Standard De
    plt.show()

    print ('<Results of Dickey-Fuller Test>')
    dftest = adfuller(timeseries, autolag='AIC')
    dfoutput = pd.Series(dftest[0:4],
                         index=['Test Statistic','p-value','#Lags
    for key,value in dftest[4].items():
        dfoutput['Critical Value (%s)'%key] = value
    print(dfoutput)
test_stationarity(data5.Value.dropna())
```




```
<Results of Dickey-Fuller Test>
Test Statistic                  -1.6867
p-value                          0.4380
#Lags Used                      13.0000
Number of Observations Used    382.0000
Critical Value (1%)             -3.4476
Critical Value (5%)             -2.8691
Critical Value (10%)            -2.5708
dtype: float64
```

Note:- Null Hypothesis for Dickey-Fuller Test: Data is Not Stationary

Here, Test Statistic value is less than Critical Value (1%, 5% and 10%), So we can reject Null Hypothesis.

Dickey-Fuller test telling that the data is Stationary

# KPSS Test

In [49]:

```
from statsmodels.tsa.stattools import kpss
def kpss_test(timeseries):
    print ('Results of KPSS Test:')
    kpsstest = kpss(timeseries, regression='ct')
    kpss_output = pd.Series(kpsstest[0:3], index=['Test Statistic
    for key,value in kpsstest[3].items():
        kpss_output['Critical Value (%s)'%key] = value
    print (kpss_output)
```

In [50]:

```
kpss_test(data3)
```

```
Results of KPSS Test:
Test Statistic            0.1403
p-value                   0.0605
Lags Used                17.0000
Critical Value (10%)      0.1190
Critical Value (5%)       0.1460
Critical Value (2.5%)     0.1760
Critical Value (1%)       0.2160
dtype: float64
```

Note:- Null Hypothesis for KPSS test: Data is Stationary

Test for stationarity: If the test statistic is greater than the critical value, we reject the null hypothesis (series is not stationary).

Here, test statistic is smaller than the critical value, "We Accept Null Hypothesis" and the data has "Stationarity".

Types of Stationarity:

```
1. Strict Stationary

2. Trend Stationary

3. Difference Stationary
```

Case 1: Both tests conclude that the series is not stationary -> series is not stationary

Case 2: Both tests conclude that the series is stationary -> series is stationary

Case 3: KPSS = stationary and ADF = not stationary -> trend stationary, remove the trend to make series strict stationary

Case 4: KPSS = not stationary and ADF = stationary -> difference stationary, use differencing to make series strict stationary

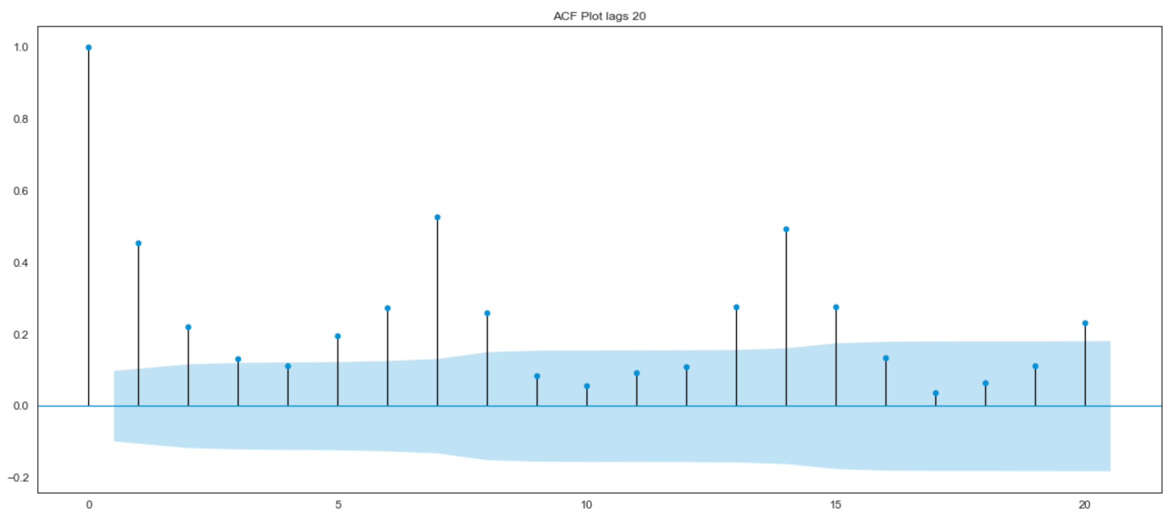
## ACF and PACF Plots till lag 20

In [51]:

```
from statsmodels.tsa.stattools import acf, pacf
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
```

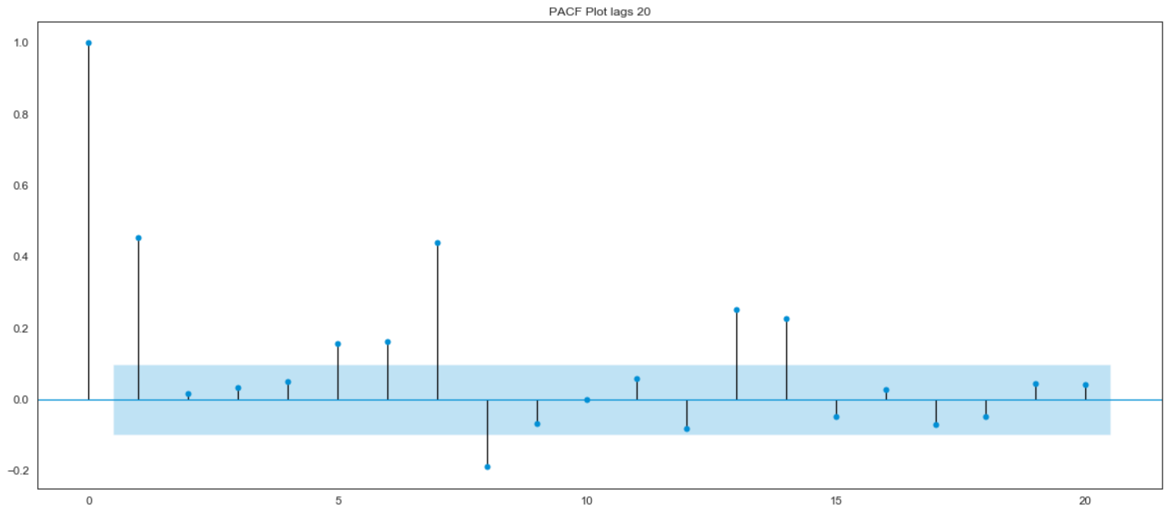
In [52]:

```
acf = plot_acf(data3, lags = 20)
plt.title("ACF Plot lags 20")
acf.show()
```



In [53]:

```
pacf = plot_pacf(data3, lags = 20)
plt.title("PACF Plot lags 20")
pacf.show()
```




In [49]:

```
data3.head(10)
```

Out[49]:

| BillDate | Value |
|---|---|
| 2018-07-01 | 124060.5910 |
| 2018-07-02 | 64358.9450 |
| 2018-07-03 | 70038.4300 |
| 2018-07-04 | 68084.0100 |
| 2018-07-05 | 80714.2200 |
| 2018-07-06 | 92766.6620 |
| 2018-07-07 | 111833.9990 |
| 2018-07-08 | 125058.1600 |
| 2018-07-09 | 71373.8750 |
| 2018-07-10 | 78692.4800 |

#################################################################################

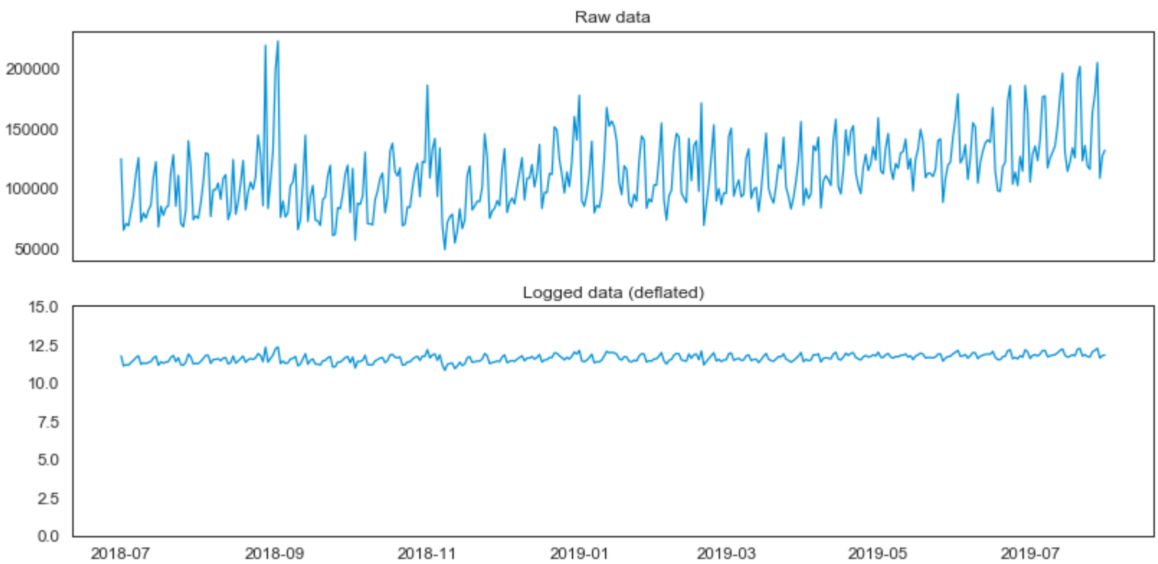
# ARIMA

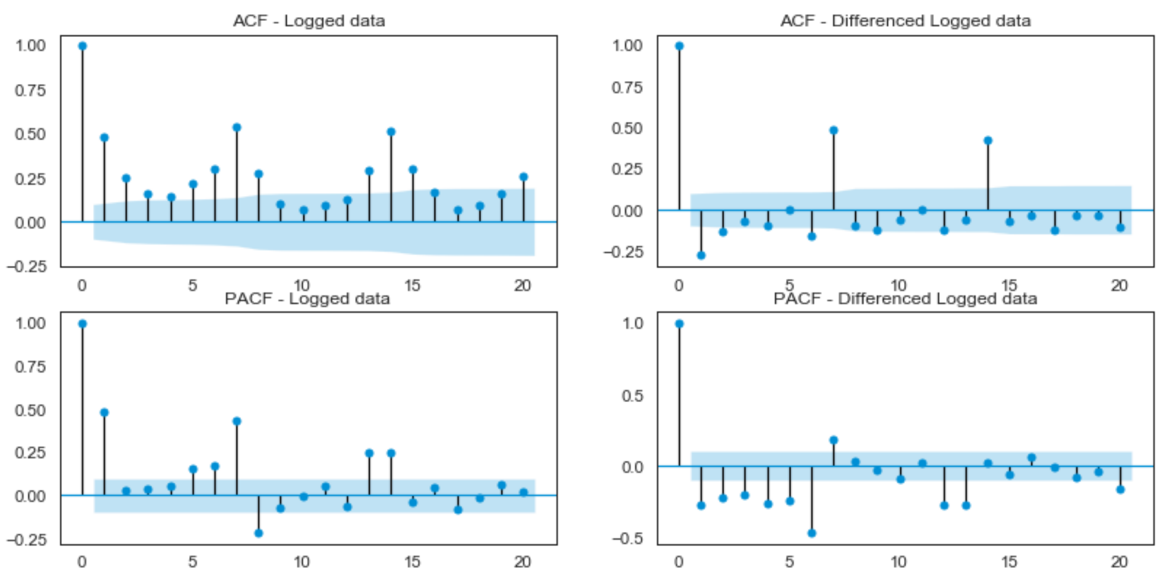
In [71]:

```
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
from statsmodels.tsa.arima_model import ARIMA
```

In [72]:

```
fig, ax = plt.subplots(2, sharex=True, figsize=(12,6))
ax[0].plot(data3.Value);
ax[0].set_title("Raw data");
ax[1].plot(np.log(data3.Value));
ax[1].set_title("Logged data (deflated)");
ax[1].set_ylim(0, 15);

from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
fig, ax = plt.subplots(2, 2, figsize=(12,6))
first_diff = (np.log(data3)- np.log(data3).shift()).dropna()
ax[0, 0] = plot_acf(np.log(data3), ax=ax[0, 0], lags=20, title="A
ax[1, 0] = plot_pacf(np.log(data3), ax=ax[1, 0], lags=20, title='
ax[0, 1] = plot_acf(first_diff , ax=ax[0, 1], lags=20, title="ACF
ax[1, 1] = plot_pacf(first_diff, ax=ax[1, 1], lags=20, title="PAC
```




## Choosing the differencing order

In [73]:

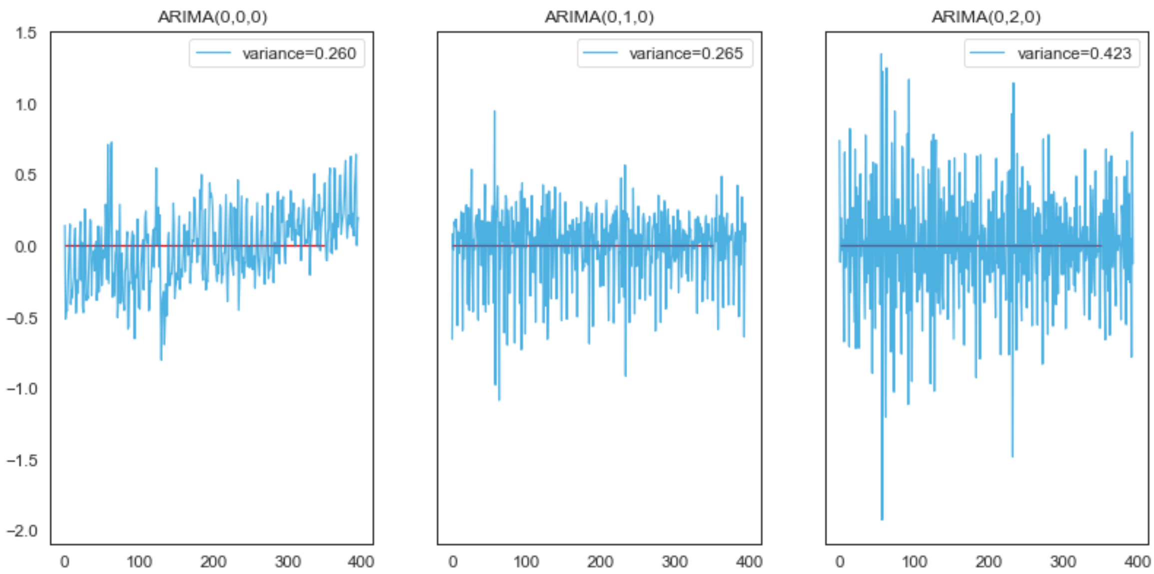
```python
from statsmodels.tsa.arima_model import ARIMA

model = ARIMA(np.log(data3).dropna(), (0, 0, 0))
res_000 = model.fit()
print(res_000.summary())

model = ARIMA(np.log(data3).dropna(), (0, 1, 0))
res_010 = model.fit()
print(res_010.summary())

model = ARIMA(np.log(data3).dropna(), (0, 2, 0))
res_020 = model.fit()
print(res_020.summary())
```

```
                              ARMA Model Results
====================================================
==========================
Dep. Variable:                  Value   No. Observat
ions:                        396
Model:                     ARMA(0, 0)   Log Likeliho
od                       -28.231
Method:                           css   S.D. of inno
vations                    0.260
Date:                Tue, 05 Nov 2019   AIC
60.462
Time:                        15:37:13   BIC
68.425
Sample:                    07-01-2018   HQIC
63.617
                         - 07-31-2019
====================================================
==========================
                 coef    std err          z      P>|
z|      [0.025      0.975]
----------------------------------------------------
--------------------------
const         11.5856      0.013    887.241      0.0
00      11.560      11.611
====================================================
==========================
                             ARIMA Model Results
====================================================
==========================
Dep. Variable:                D.Value   No. Observat
ions:                        395
Model:                 ARIMA(0, 1, 0)   Log Likeliho
od                       -35.851
```

```
Method:                           css   S.D. of inno
vations                     0.265
Date:                Tue, 05 Nov 2019   AIC
75.702
Time:                        15:37:13   BIC
83.660
Sample:                    07-02-2018   HQIC
78.855
                         - 07-31-2019
====================================================
==========================
                 coef    std err          z      P>|
z|      [0.025      0.975]
----------------------------------------------------
--------------------------
const          0.0001      0.013      0.011      0.9
92      -0.026       0.026
====================================================
==========================
                             ARIMA Model Results
====================================================
==========================
Dep. Variable:                D2.Value   No. Observat
ions:                         394
Model:                 ARIMA(0, 2, 0)   Log Likeliho
od                     -219.890
Method:                           css   S.D. of inno
vations                     0.423
Date:                Tue, 05 Nov 2019   AIC
443.780
Time:                        15:37:13   BIC
451.733
Sample:                    07-03-2018   HQIC
446.932
                         - 07-31-2019
====================================================
==========================
                 coef    std err          z      P>|
z|      [0.025      0.975]
----------------------------------------------------
--------------------------
const          0.0018      0.021      0.082      0.9
34      -0.040       0.044
====================================================
==========================
```

In [74]:

```
fig, ax = plt.subplots(1, 3, sharey=True, figsize=(12, 6))
ax[0].plot(res_000.resid.values, alpha=0.7, label='variance={:.3f
ax[0].hlines(0, xmin=0, xmax=350, color='r');
ax[0].set_title("ARIMA(0,0,0)");
ax[0].legend();
ax[1].plot(res_010.resid.values, alpha=0.7, label='variance={:.3f
ax[1].hlines(0, xmin=0, xmax=350, color='r');
ax[1].set_title("ARIMA(0,1,0)");
ax[1].legend();
ax[2].plot(res_020.resid.values, alpha=0.7, label='variance={:.3f
ax[2].hlines(0, xmin=0, xmax=350, color='r');
ax[2].set_title("ARIMA(0,2,0)");
ax[2].legend();
```




Note: From the above results, we can see that the AIC value is low for "Diffrencing order 1".

## Choosing the MA order

In [76]:

```
model = ARIMA(np.log(data3).dropna(), (0, 0, 0))
res_010 = model.fit()
print(res_010.summary())

model = ARIMA(np.log(data3).dropna(), (1, 0, 0))
res_110 = model.fit()
print(res_110.summary())

model = ARIMA(np.log(data3).dropna(), (2, 0, 0))
```

```
res_210 = model.fit()
print(res_210.summary())
```

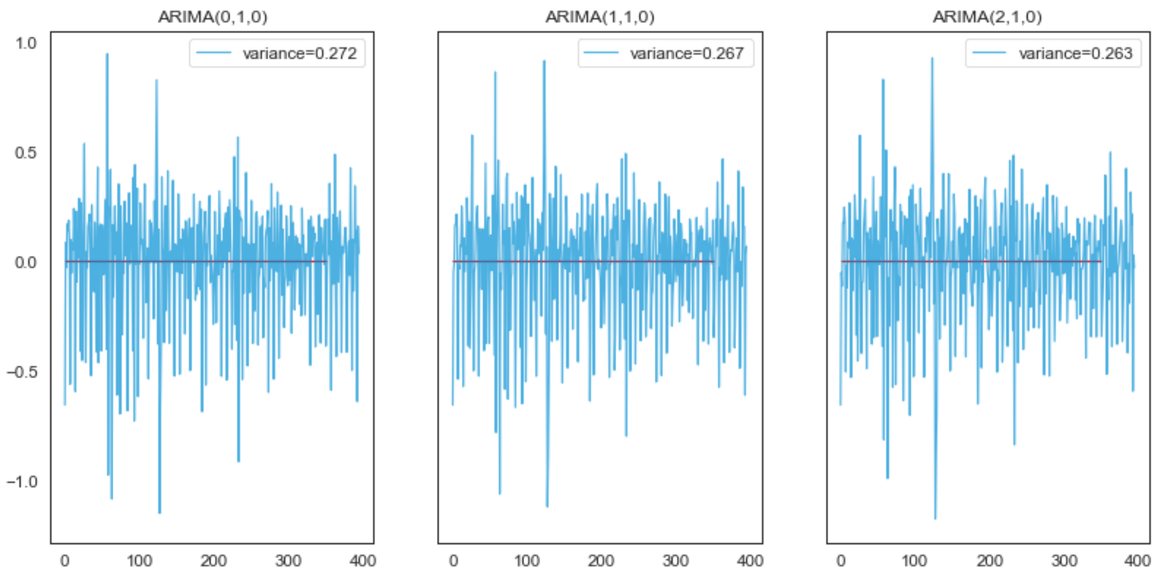
```
                              ARMA Model Results
==============================================================================
Dep. Variable:                  Value   No. Observations:                  396
Model:                     ARMA(0, 0)   Log Likelihood                 -28.231
Method:                           css   S.D. of innovations              0.260
Date:                Tue, 05 Nov 2019   AIC                             60.462
Time:                        15:38:18   BIC                             68.425
Sample:                    07-01-2018   HQIC                            63.617
                         - 07-31-2019
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const         11.5856      0.013    887.241      0.000      11.560      11.611
==============================================================================
                              ARMA Model Results
==============================================================================
Dep. Variable:                  Value   No. Observations:                  396
Model:                     ARMA(1, 0)   Log Likelihood                  23.708
Method:                       css-mle   S.D. of innovations              0.228
Date:                Tue, 05 Nov 2019   AIC                            -41.416
Time:                        15:38:18   BIC                            -29.471
Sample:                    07-01-2018   HQIC                           -36.684
                         - 07-31-2019
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
```

```
----------------------------------------------------------------------
---------------------------
const          11.5864      0.022    527.242      0.
000      11.543      11.629
ar.L1.Value     0.4802      0.044     10.909      0.
000       0.394       0.566
                                     Roots
======================================================================
=========================
                  Real          Imaginary
Modulus         Frequency
----------------------------------------------------------------------
-------------------------
AR.1            2.0824          +0.0000j
2.0824            0.0000
----------------------------------------------------------------------
-------------------------
                               ARMA Model Results
======================================================================
==========================
Dep. Variable:                  Value   No. Observat
ions:                     396
Model:                     ARMA(2, 0)   Log Likeliho
od                     23.851
Method:                       css-mle   S.D. of inno
vations                 0.228
Date:                Tue, 05 Nov 2019   AIC
-39.702
Time:                        15:38:18   BIC
-23.777
Sample:                    07-01-2018   HQIC
-33.393
                         - 07-31-2019
======================================================================
===========================
                  coef    std err          z      P>
|z|      [0.025      0.975]
----------------------------------------------------------------------
---------------------------
const          11.5864      0.023    513.386      0.
000      11.542      11.631
ar.L1.Value     0.4671      0.050      9.275      0.
000       0.368       0.566
ar.L2.Value     0.0271      0.051      0.536      0.
592      -0.072       0.126
                                     Roots
======================================================================
=========================
```

```
                 Real          Imaginary
Modulus         Frequency
-----------------------------------------------------
-------------------------
AR.1             1.9258          +0.0000j
1.9258             0.0000
AR.2           -19.1607          +0.0000j
19.1607             0.5000
-----------------------------------------------------
-------------------------
```

In [58]:

```
fig, ax = plt.subplots(1, 3, sharey=True, figsize=(12, 6))
ax[0].plot(res_010.resid.values, alpha=0.7, label='variance={:.3f
ax[0].hlines(0, xmin=0, xmax=350, color='r');
ax[0].set_title("ARIMA(0,1,0)");
ax[0].legend();
ax[1].plot(res_110.resid.values, alpha=0.7, label='variance={:.3f
ax[1].hlines(0, xmin=0, xmax=350, color='r');
ax[1].set_title("ARIMA(1,1,0)");
ax[1].legend();
ax[2].plot(res_210.resid.values, alpha=0.7, label='variance={:.3f
ax[2].hlines(0, xmin=0, xmax=350, color='r');
ax[2].set_title("ARIMA(2,1,0)");
ax[2].legend();
```




## Choosing the AR order

In [78]:

```
model = ARIMA(np.log(data3).dropna(), (1, 0, 0))
res_210 = model.fit()
print(res_210.summary())
```

```
print(res_210.summary())

model = ARIMA(np.log(data3).dropna(), (1, 0, 1))
res_211 = model.fit()
print(res_211.summary())

model = ARIMA(np.log(data3).dropna(), (0, 1, 2))
res_212 = model.fit()
print(res_212.summary())
```

```
                              ARMA Model Results
====================================================
==========================
Dep. Variable:                  Value   No. Observat
ions:                       396
Model:                     ARMA(1, 0)   Log Likeliho
od                       23.708
Method:                       css-mle   S.D. of inno
vations                   0.228
Date:                Tue, 05 Nov 2019   AIC
-41.416
Time:                        15:40:02   BIC
-29.471
Sample:                    07-01-2018   HQIC
-36.684
                         - 07-31-2019
====================================================
===========================
                  coef    std err          z      P>
|z|      [0.025      0.975]
----------------------------------------------------
---------------------------
const          11.5864      0.022    527.242      0.
000      11.543      11.629
ar.L1.Value     0.4802      0.044     10.909      0.
000       0.394       0.566
                                    Roots
====================================================
=========================
                  Real          Imaginary
Modulus         Frequency
----------------------------------------------------
-------------------------
AR.1            2.0824           +0.0000j
2.0824            0.0000
----------------------------------------------------
-------------------------
                              ARMA Model Results
====================================================
```

```
==========================
Dep. Variable:                  Value   No. Observat
ions:                     396
Model:                     ARMA(1, 1)   Log Likeliho
od                     23.882
Method:                       css-mle   S.D. of inno
vations                 0.228
Date:                Tue, 05 Nov 2019   AIC
-39.763
Time:                        15:40:03   BIC
-23.838
Sample:                    07-01-2018   HQIC
-33.454
                         - 07-31-2019
====================================================
===========================
                  coef    std err          z      P>
|z|      [0.025      0.975]
----------------------------------------------------
---------------------------
const          11.5864      0.023    509.222      0.
000      11.542      11.631
ar.L1.Value     0.5331      0.098      5.453      0.
000       0.341       0.725
ma.L1.Value    -0.0692      0.119     -0.582      0.
561      -0.302       0.164
                                    Roots
====================================================
=========================
                  Real          Imaginary
Modulus         Frequency
----------------------------------------------------
-------------------------
AR.1            1.8760           +0.0000j
1.8760            0.0000
MA.1           14.4608           +0.0000j
14.4608            0.0000
----------------------------------------------------
-------------------------
                              ARIMA Model Results
====================================================
==========================
Dep. Variable:                D.Value   No. Observat
ions:                     395
Model:                 ARIMA(0, 1, 2)   Log Likeliho
od                     42.073
Method:                       css-mle   S.D. of inno
vations                 0.216
```

```
Date:                    Tue, 05 Nov 2019   AIC
-76.146
Time:                            15:40:03   BIC
-60.231
Sample:                        07-02-2018   HQIC
-69.840
                             - 07-31-2019
=====================================================
==============================
                     coef    std err          z
P>|z|      [0.025      0.975]
-----------------------------------------------------
------------------------------
const             0.0011      0.000      8.547
0.000       0.001       0.001
ma.L1.D.Value    -0.6687      0.048    -13.912
0.000      -0.763      -0.575
ma.L2.D.Value    -0.3313      0.046     -7.180
0.000      -0.422      -0.241
                                    Roots
=====================================================
==========================
                   Real          Imaginary
Modulus         Frequency
-----------------------------------------------------
--------------------------
MA.1             1.0000           +0.0000j
1.0000             0.0000
MA.2            -3.0188           +0.0000j
3.0188             0.5000
-----------------------------------------------------
--------------------------
```

## Model

In [55]:

```
data3.head()
```

Out[55]:

| | Value |
|---|---|
| **BillDate** | |
| 2018-07-01 | 124060.5910 |
| 2018-07-02 | 64358.9450 |
| 2018-07-03 | 70038.4300 |
| 2018-07-04 | 68084.0100 |
| 2018-07-05 | 80714.2200 |

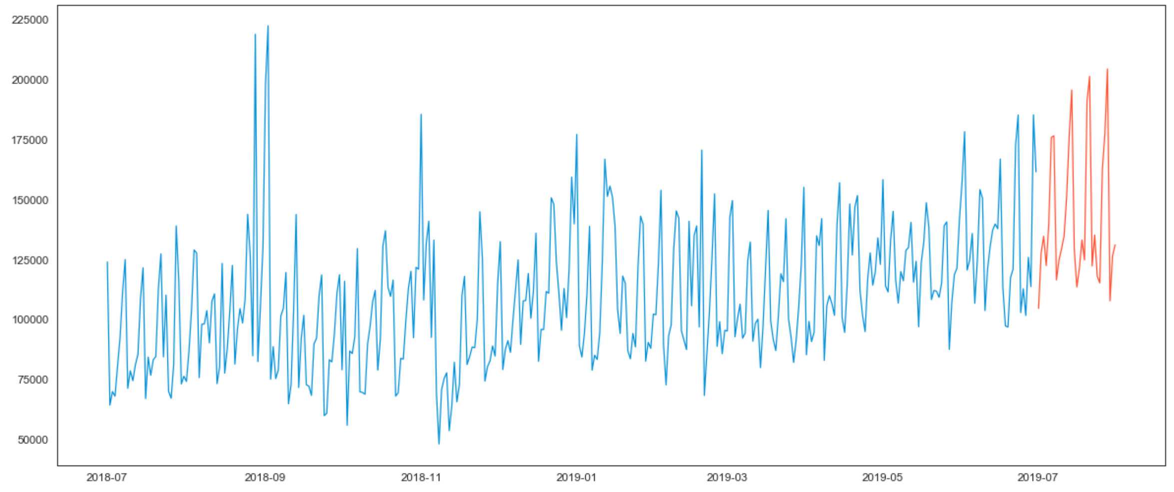
In [56]:

```
train, test = data3[:365], data3[365:]
print(train.shape)
print(test.shape)
```

```
(365, 1)
(31, 1)
```

In [57]:

```
plt.plot(train)
plt.plot(test)
plt.show();
```




# Auto_Arima

In [58]:

```
from pmdarima.arima import auto_arima
```

In [59]:

```
model = auto_arima(train, start_p=0, start_q=0, max_p=6, max_q=6,
                    seasonal=True, trace=True, d=1, D=1, error_act
                   random_state=20, n_fits=30)
```

```
Fit ARIMA: order=(0, 1, 0) seasonal_order=(0, 1, 0,
7); AIC=8416.003, BIC=8423.759, Fit time=0.031 secon
ds
Fit ARIMA: order=(1, 1, 0) seasonal_order=(1, 1, 0,
7); AIC=8309.917, BIC=8325.427, Fit time=0.156 secon
ds
Fit ARIMA: order=(0, 1, 1) seasonal_order=(0, 1, 1,
7); AIC=8162.646, BIC=8178.157, Fit time=1.135 secon
ds
Fit ARIMA: order=(0, 1, 1) seasonal_order=(1, 1, 1,
7); AIC=8201.163, BIC=8220.552, Fit time=0.738 secon
ds
Fit ARIMA: order=(0, 1, 1) seasonal_order=(0, 1, 0,
7); AIC=8356.116, BIC=8367.749, Fit time=0.062 secon
ds
Fit ARIMA: order=(0, 1, 1) seasonal_order=(0, 1, 2,
7); AIC=8201.156, BIC=8220.545, Fit time=0.981 secon
ds
Fit ARIMA: order=(0, 1, 1) seasonal_order=(1, 1, 2,
7); AIC=8201.847, BIC=8225.113, Fit time=1.194 secon
ds
Fit ARIMA: order=(1, 1, 1) seasonal_order=(0, 1, 1,
7); AIC=8191.512, BIC=8210.901, Fit time=0.820 secon
ds
Fit ARIMA: order=(0, 1, 0) seasonal_order=(0, 1, 1,
7); AIC=8204.934, BIC=8216.568, Fit time=0.657 secon
ds
Fit ARIMA: order=(0, 1, 2) seasonal_order=(0, 1, 1,
7); AIC=8117.996, BIC=8137.385, Fit time=1.214 secon
ds
Fit ARIMA: order=(1, 1, 3) seasonal_order=(0, 1, 1,
7); AIC=8194.886, BIC=8222.030, Fit time=2.026 secon
ds
Fit ARIMA: order=(0, 1, 2) seasonal_order=(1, 1, 1,
7); AIC=8200.916, BIC=8224.182, Fit time=0.481 secon
ds
Fit ARIMA: order=(0, 1, 2) seasonal_order=(0, 1, 0,
7); AIC=8324.783, BIC=8340.294, Fit time=0.488 secon
```

```
ds
Fit ARIMA: order=(0, 1, 2) seasonal_order=(0, 1, 2,
7); AIC=8200.920, BIC=8224.186, Fit time=0.808 secon
ds
Fit ARIMA: order=(0, 1, 2) seasonal_order=(1, 1, 2,
7); AIC=8200.641, BIC=8227.785, Fit time=1.502 secon
ds
Fit ARIMA: order=(1, 1, 2) seasonal_order=(0, 1, 1,
7); AIC=8163.005, BIC=8186.272, Fit time=1.633 secon
ds
Fit ARIMA: order=(0, 1, 3) seasonal_order=(0, 1, 1,
7); AIC=8209.466, BIC=8232.732, Fit time=1.005 secon
ds
Total fit time: 14.936 seconds
```

In [60]:

```
model.summary()
```

Out[60]:

Statespace Model Results

| | | | |
|---|---|---|---|
| **Dep. Variable:** | y | **No. Observations:** | 365 |
| **Model:** | SARIMAX(0, 1, 2)x(0, 1, 1, 7) | **Log Likelihood** | -4053.998 |
| **Date:** | Tue, 05 Nov 2019 | **AIC** | 8117.996 |
| **Time:** | 10:36:26 | **BIC** | 8137.385 |
| **Sample:** | 0 | **HQIC** | 8125.708 |
| | - 365 | | |
| **Covariance Type:** | opg | | |

| | coef | std err | z | P>\|z\| | [0.025 | 0.975] |
|---|---|---|---|---|---|---|
| **intercept** | -23.1829 | 24.431 | -0.949 | 0.343 | -71.068 | 24.702 |
| **ma.L1** | -0.5272 | 0.034 | -15.390 | 0.000 | -0.594 | -0.460 |
| **ma.L2** | -0.1901 | 0.043 | -4.416 | 0.000 | -0.274 | -0.106 |
| **ma.S.L7** | -0.9914 | 0.050 | -19.690 | 0.000 | -1.090 | -0.893 |
| **sigma2** | 4.04e+08 | 3.57e-07 | 1.13e+15 | 0.000 | 4.04e+08 | 4.04e+08 |

| | | | |
|---|---|---|---|
| **Ljung-Box (Q):** | 59.32 | **Jarque-Bera (JB):** | 210.40 |
| **Prob(Q):** | 0.03 | **Prob(JB):** | 0.00 |
| **Heteroskedasticity (H):** | 0.36 | **Skew:** | 0.53 |
| **Prob(H) (two-sided):** | 0.00 | **Kurtosis:** | 6.61 |

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
[2] Covariance matrix is singular or near-singular, with condition number 2.73e+31. Standard errors may be unstable.

In [61]:

```
pred = pd.DataFrame(model.predict(n_periods=31), index=test.inde
```

In [62]:

```
pred.columns = ['Forecasted']
```

In [63]:

```
pred
```

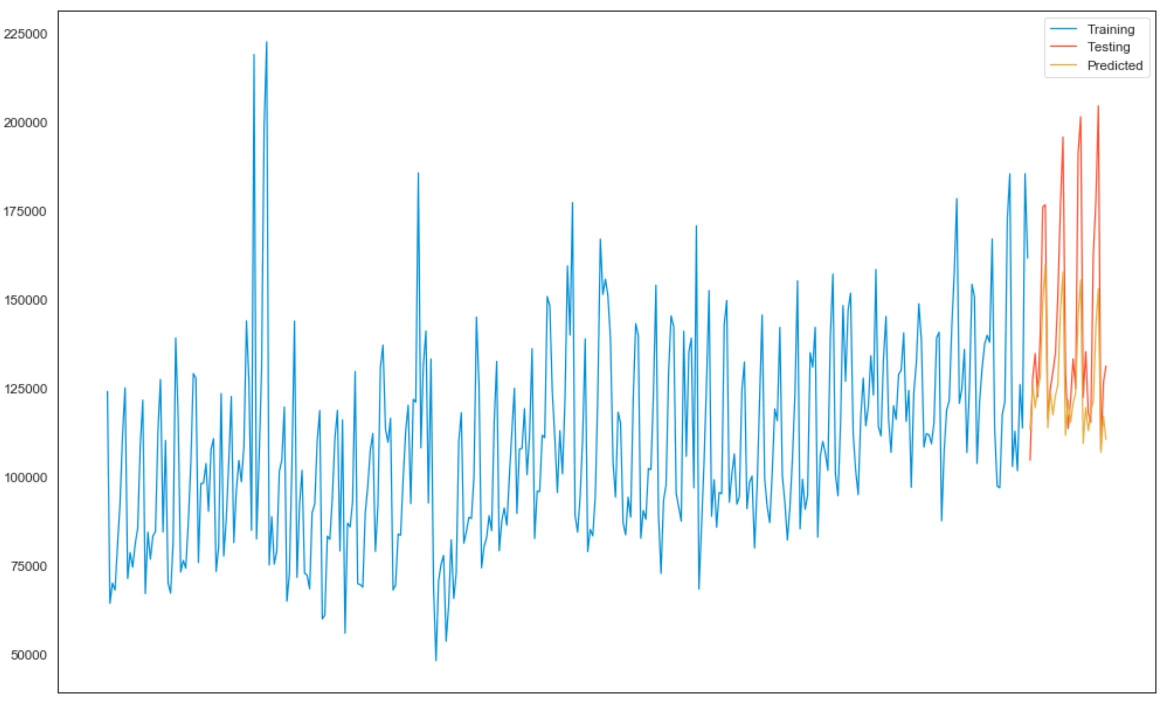
Out[63]:

| | Forecasted |
|---|---|
| **BillDate** | |
| 2019-07-01 | 113339.0241 |
| 2019-07-02 | 125883.7944 |
| 2019-07-03 | 119444.8986 |
| 2019-07-04 | 124833.1830 |
| 2019-07-05 | 127928.8809 |
| 2019-07-06 | 149259.4597 |
| 2019-07-07 | 159766.6176 |
| 2019-07-08 | 113736.6994 |
| 2019-07-09 | 123918.8396 |
| 2019-07-10 | 117456.7609 |
| 2019-07-11 | 122821.8623 |
| 2019-07-12 | 125894.3772 |
| 2019-07-13 | 147201.7731 |
| 2019-07-14 | 157685.7481 |
| 2019-07-15 | 111632.6469 |
| 2019-07-16 | 121791.6042 |
| 2019-07-17 | 115306.3425 |
| 2019-07-18 | 120648.2609 |
| 2019-07-19 | 123697.5929 |
| 2019-07-20 | 144981.8059 |

| | |
|---|---|
| 2019-07-21 | 155442.5979 |
| 2019-07-22 | 109366.3138 |
| 2019-07-23 | 119502.0881 |
| 2019-07-24 | 112993.6435 |
| 2019-07-25 | 118312.3790 |
| 2019-07-26 | 121338.5281 |
| 2019-07-27 | 142599.5580 |
| 2019-07-28 | 153037.1672 |
| 2019-07-29 | 106937.7001 |
| 2019-07-30 | 117050.2915 |
| 2019-07-31 | 110518.6639 |

In [64]:

```
plt.figure(figsize=(15,10))
plt.plot(train, label='Training')
plt.plot(test, label='Testing')
plt.plot(pred, label='Predicted')
plt.legend()
plt.show();
```




In [65]:

```
test['forecasted'] = pred
test['error'] = test['Value'] - test['forecasted']
test
```

Out[65]:

| BillDate | Value | forecasted | error |
|---|---|---|---|
| 2019-07-01 | 104695.7250 | 113339.0241 | -8643.2991 |
| 2019-07-02 | 127803.5900 | 125883.7944 | 1919.7956 |
| 2019-07-03 | 134738.3800 | 119444.8986 | 15293.4814 |
| 2019-07-04 | 122540.8750 | 124833.1830 | -2292.3080 |
| 2019-07-05 | 139906.2360 | 127928.8809 | 11977.3551 |
| 2019-07-06 | 175973.6300 | 149259.4597 | 26714.1703 |
| 2019-07-07 | 176618.0850 | 159766.6176 | 16851.4674 |
| 2019-07-08 | 116535.6400 | 113736.6994 | 2798.9406 |
| 2019-07-09 | 124766.9700 | 123918.8396 | 848.1304 |
| 2019-07-10 | 129517.0500 | 117456.7609 | 12060.2891 |
| 2019-07-11 | 134813.2500 | 122821.8623 | 11991.3877 |
| 2019-07-12 | 151176.9640 | 125894.3772 | 25282.5868 |
| 2019-07-13 | 176453.4200 | 147201.7731 | 29251.6469 |
| 2019-07-14 | 195715.5800 | 157685.7481 | 38029.8319 |
| 2019-07-15 | 129262.1900 | 111632.6469 | 17629.5431 |
| 2019-07-16 | 113593.5720 | 121791.6042 | -8198.0322 |
| 2019-07-17 | 121408.7200 | 115306.3425 | 6102.3775 |
| 2019-07-18 | 133213.1000 | 120648.2609 | 12564.8391 |
| 2019-07-19 | 124875.5150 | 123697.5929 | 1177.9221 |
| 2019-07-20 | 191041.9140 | 144981.8059 | 46060.1081 |
| 2019-07-21 | 201396.9770 | 155442.5979 | 45954.3791 |
| 2019-07-22 | 122373.8600 | 109366.3138 | 13007.5462 |
| 2019-07-23 | 135259.0700 | 119502.0881 | 15756.9819 |
| 2019-07-24 | 118110.3900 | 112993.6435 | 5116.7465 |
| 2019-07-25 | 115316.2280 | 118312.3790 | -2996.1510 |
| 2019-07-26 | 162643.2550 | 121338.5281 | 41304.7269 |
| 2019-07-27 | 177738.2500 | 142599.5580 | 35138.6920 |
| 2019-07-28 | 204500.9860 | 153037.1672 | 51463.8188 |

| | | | |
|---|---|---|---|
| 2019-07-29 | 107830.3680 | 106937.7001 | 892.6679 |
| 2019-07-30 | 126576.6760 | 117050.2915 | 9526.3845 |
| 2019-07-31 | 131196.7250 | 110518.6639 | 20678.0611 |

In [66]:

```
from sklearn import metrics
```

In [67]:

```
metrics.mean_absolute_error(test.Value, test.forecasted)
```

Out[67]:

```
17339.473179332323
```

In [68]:

```
metrics.median_absolute_error(test.Value, test.forecasted)
```

Out[68]:

```
12564.839062976258
```

In [69]:

```
import math
```

In [70]:

```
math.sqrt(metrics.mean_squared_error(test.Value, test.forecasted)
```

Out[70]:

```
22765.63356590128
```

```
#################################################################
```

In [ ]:

```

```

In [ ]:

```

```

In [ ]:

```

```

# LSTM Model

In [10]:

```
import math
import keras
from keras.layers import *
from sklearn.preprocessing import MinMaxScaler
from keras.models import Sequential
from keras.layers import Dense, Dropout, LSTM
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
from keras.callbacks import EarlyStopping
```

```
Using TensorFlow backend.
```

In [ ]:

```

```

In [69]:

```
data2.head()
```

Out[69]:

| | Createddate | Value | year | quarter | month | day | weekday |
|---|---|---|---|---|---|---|---|
| 0 | 2018-07-01 | 123601.2160 | 2018 | 3 | 7 | 1 | 0 |
| 1 | 2018-07-02 | 64608.3200 | 2018 | 3 | 7 | 2 | 1 |
| 2 | 2018-07-03 | 70248.4300 | 2018 | 3 | 7 | 3 | 1 |
| 3 | 2018-07-04 | 68084.0100 | 2018 | 3 | 7 | 4 | 1 |
| 4 | 2018-07-05 | 80714.2200 | 2018 | 3 | 7 | 5 | 1 |

In [30]:

```
data = data2
new_data = pd.DataFrame(index=range(0,len(data2)),columns=['BillD
for i in range(0,len(data)):
    new_data['BillDate'][i] = data['BillDate'][i]
    new_data['Value'][i] = data['Value'][i]
```

In [31]:

```
new_data.index = new_data.BillDate
new_data.drop('BillDate', axis=1, inplace=True)
```

In [32]:

```
dataset = new_data.values
```

In [33]:

```
train = dataset[0:316,:]
valid = dataset[316:,:]
```

In [34]:

```
scaler = MinMaxScaler(feature_range=(0, 1))
scaled_data = scaler.fit_transform(dataset)
```

In [35]:

```
x_train, y_train = [], []
for i in range(20,len(train)):
    x_train.append(scaled_data[i-20:i,0])
    y_train.append(scaled_data[i,0])
x_train, y_train = np.array(x_train), np.array(y_train)
```

In [36]:

```
x_train = np.reshape(x_train, (x_train.shape[0],x_train.shape[1],
```

In [37]:

```
model = Sequential()
model.add(LSTM(units=50, return_sequences=True, input_shape=(x_tr
model.add(LSTM(units=50))
model.add(Dense(1))
```

In [38]:

```
model.compile(loss='mean_squared_error', optimizer='adam')
model.fit(x_train, y_train, epochs=20, batch_size=1, verbose=2)
```

```
Epoch 1/20
11s - loss: 0.0100
Epoch 2/20
10s - loss: 0.0084
Epoch 3/20
11s - loss: 0.0084
Epoch 4/20
10s - loss: 0.0073
Epoch 5/20
10s - loss: 0.0076
Epoch 6/20
10s - loss: 0.0066
Epoch 7/20
10s - loss: 0.0062
Epoch 8/20
11s - loss: 0.0069
Epoch 9/20
10s - loss: 0.0061
Epoch 10/20
```

In [39]:

```
inputs = new_data[len(new_data) - len(valid) - 20:].values
inputs = inputs.reshape(-1,1)
inputs  = scaler.transform(inputs)
```

In [40]:

```
X_test = []
for i in range(20,inputs.shape[0]):
    X_test.append(inputs[i-20:i,0])
X_test = np.array(X_test)
```

In [210]:

```
X_test = np.reshape(X_test, (X_test.shape[0],X_test.shape[1],1))
closing_price = model.predict(X_test)
closing_price = scaler.inverse_transform(closing_price)
```

In [211]:

```
mae=np.absolute(np.mean((valid-closing_price)))
mae
```

Out[211]:

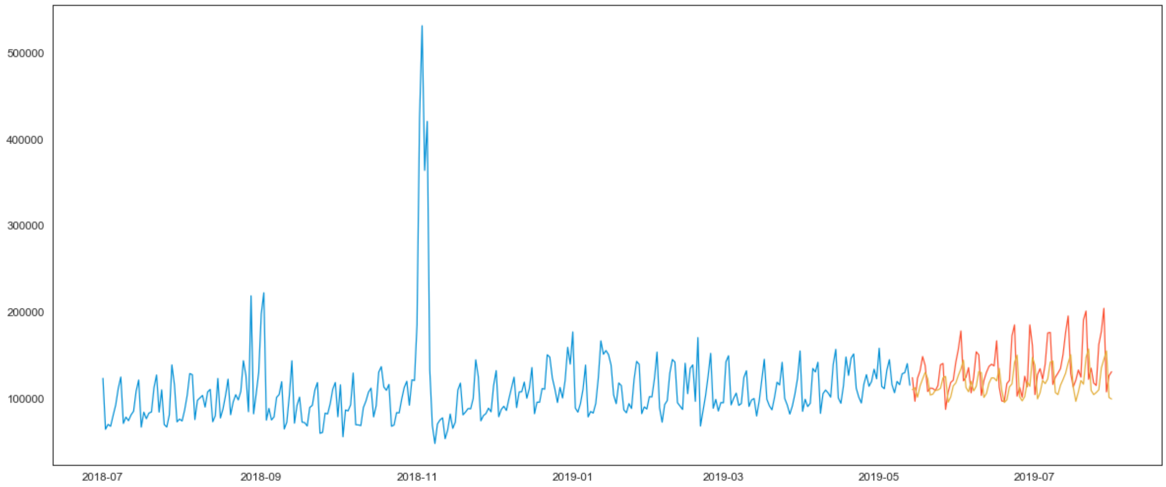
```
15210.18796202546
```

In [214]:

```
train = new_data[:316]
valid = new_data[316:]
valid['Predictions'] = closing_price
plt.plot(train['Value'])
plt.plot(valid[['Value','Predictions']])
```

Out[214]:

```
[<matplotlib.lines.Line2D at 0x14109b080>,
 <matplotlib.lines.Line2D at 0x14109b208>]
```




##################################################################

In [4]:

```
sweet = pd.read_csv("Daily_Sales1.csv")
sweet.head()
```

Out[4]:

| | Unnamed: 0 | BillDate | Value | day_week |
|---|---|---|---|---|
| 0 | 0 | 2018-07-01 | 124060.591 | 6 |
| 1 | 1 | 2018-07-02 | 64358.945 | 0 |
| 2 | 2 | 2018-07-03 | 70038.430 | 1 |
| 3 | 3 | 2018-07-04 | 68084.010 | 2 |
| 4 | 4 | 2018-07-05 | 80714.220 | 3 |

In [5]:

```
data = sweet
new_data = pd.DataFrame(index=range(0,len(sweet)),columns=['BillD
for i in range(0,len(data)):
    new_data['BillDate'][i] = data['BillDate'][i]
    new_data['Value'][i] = data['Value'][i]
```

In [6]:

```
new_data.index = new_data.BillDate
new_data.drop('BillDate', axis=1, inplace=True)
```

In [7]:

```
dataset = new_data.values
```

In [8]:

```
train = dataset[0:316,:]
valid = dataset[316:,:]
```

In [11]:

```
scaler = MinMaxScaler(feature_range=(0, 1))
scaled_data = scaler.fit_transform(dataset)
```

In [12]:

```
x_train, y_train = [], []
for i in range(20,len(train)):
    x_train.append(scaled_data[i-20:i,0])
    y_train.append(scaled_data[i,0])
x_train, y_train = np.array(x_train), np.array(y_train)
```

In [13]:

```
x_train = np.reshape(x_train, (x_train.shape[0],x_train.shape[1],
```

In [14]:

```
model = Sequential()
model.add(LSTM(units=50, return_sequences=True, input_shape=(x_tı
model.add(LSTM(units=50))
model.add(Dense(1))
```

In [15]:

```
model.compile(loss='mean_squared_error', optimizer='adam')
model.fit(x_train, y_train, epochs=100, batch_size=1, verbose=1)
```

```
Epoch 1/100
296/296 [==============================] - 11s - los
s: 0.0102
Epoch 2/100
296/296 [==============================] - 11s - los
s: 0.0092
Epoch 3/100
296/296 [==============================] - 12s - los
s: 0.0082
Epoch 4/100
296/296 [==============================] - 14s - los
s: 0.0070
Epoch 5/100
296/296 [==============================] - 11s - los
s: 0.0071
Epoch 6/100
296/296 [==============================] - 10s - los
s: 0.0064
Epoch 7/100
```

In [16]:

```
inputs = new_data[len(new_data) - len(valid) - 20:].values
inputs = inputs.reshape(-1,1)
inputs  = scaler.transform(inputs)
```

In [17]:

```
X_test = []
for i in range(20,inputs.shape[0]):
    X_test.append(inputs[i-20:i,0])
X_test = np.array(X_test)
```

In [18]:

```
X_test = np.reshape(X_test, (X_test.shape[0],X_test.shape[1],1))
closing_price = model.predict(X_test)
closing_price = scaler.inverse_transform(closing_price)
```

In [19]:

```
mae=np.absolute(np.mean((valid-closing_price)))
mae
```

Out[19]:

```
19698.17791718764
```

In [23]:

```
import matplotlib.pyplot as plt
```

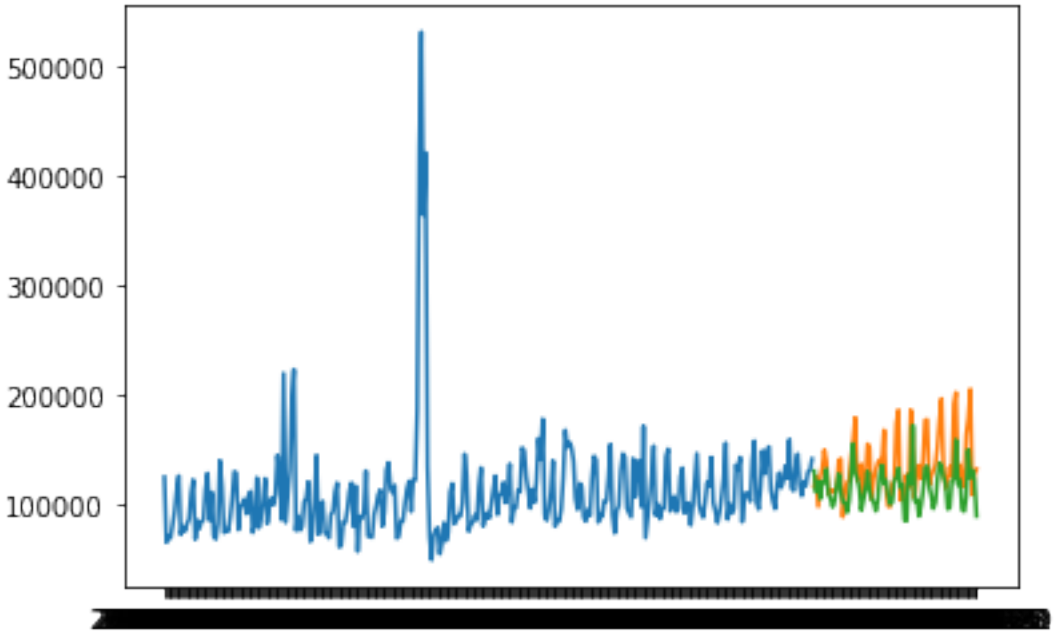
In [24]:

```
train = new_data[:316]
valid = new_data[316:]
valid['Predictions'] = closing_price
plt.plot(train['Value'])
plt.plot(valid[['Value','Predictions']])
```

```
/anaconda3/lib/python3.6/site-packages/ipykernel_lau
ncher.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice fro
m a DataFrame.
Try using .loc[row_indexer,col_indexer] = value inst
ead

See the caveats in the documentation: http://pandas.
pydata.org/pandas-docs/stable/user_guide/indexing.ht
ml#returning-a-view-versus-a-copy
(http://pandas.pydata.org/pandas-docs/stable/user_gu
ide/indexing.html#returning-a-view-versus-a-copy)
  This is separate from the ipykernel package so we
can avoid doing imports until
```

Out[24]:

```
[<matplotlib.lines.Line2D at 0x117567dd8>,
 <matplotlib.lines.Line2D at 0x117567f28>]
```




In [ ]:

# Stock Market Data

In [180]:

```
stock = pd.read_csv("Stock.csv")
```

In [181]:

```
stock.head()
```

Out[181]:

| | Date | Open | High | Low | Last | Close | Total Trac Quanti |
|---|---|---|---|---|---|---|---|
| 0 | 2013-10-08 | 157.0000 | 157.8000 | 155.2000 | 155.8000 | 155.8000 | 1720413.00( |
| 1 | 2013-10-09 | 155.7000 | 158.2000 | 154.1500 | 155.3000 | 155.5500 | 2049580.00( |
| 2 | 2013-10-10 | 156.0000 | 160.8000 | 155.8500 | 160.3000 | 160.1500 | 3124853.00( |
| 3 | 2013-10-11 | 161.1500 | 163.4500 | 159.0000 | 159.8000 | 160.0500 | 1880046.00( |
| 4 | 2013-10-14 | 160.8500 | 161.4500 | 157.7000 | 159.3000 | 159.4500 | 1281419.00( |

In [260]:

```
stock.describe()
```

Out[260]:

| | Open | High | Low | Last | Close | Total Qu |
|---|---|---|---|---|---|---|
| count | 1235.0000 | 1235.0000 | 1235.0000 | 1235.0000 | 1235.0000 | 1235 |
| mean | 168.9549 | 171.4291 | 166.4023 | 168.7364 | 168.7311 | 2604151 |
| std | 51.4991 | 52.4368 | 50.5429 | 51.5874 | 51.5449 | 2277027 |
| min | 103.0000 | 104.6000 | 100.0000 | 102.6000 | 102.6500 | 100180 |
| 25% | 137.5500 | 138.9250 | 135.2500 | 137.1750 | 137.2250 | 1284481 |
| 50% | 151.5000 | 153.2500 | 149.5000 | 151.2000 | 151.1000 | 1964885 |
| 75% | 169.0000 | 172.3250 | 166.7000 | 169.1000 | 169.5000 | 3095788 |
| max | 327.7000 | 328.7500 | 321.6500 | 325.9500 | 325.7500 | 29191015 |

In [182]:

```
data = stock.sort_index(ascending=True, axis=0)
new_data = pd.DataFrame(index=range(0,len(stock)),columns=['Date'
for i in range(0,len(data)):
    new_data['Date'][i] = data['Date'][i]
    new_data['Close'][i] = data['Close'][i]
```

In [183]:

```
new_data.index = new_data.Date
new_data.drop('Date', axis=1, inplace=True)
```

In [184]:

```
dataset = new_data.values
```

In [185]:

```
train = dataset[0:987,:]
valid = dataset[987:,:]
```

In [186]:

```
scaler = MinMaxScaler(feature_range=(0, 1))
scaled_data = scaler.fit_transform(dataset)
```

In [187]:

```
x_train, y_train = [], []
for i in range(60,len(train)):
    x_train.append(scaled_data[i-60:i,0])
    y_train.append(scaled_data[i,0])
x_train, y_train = np.array(x_train), np.array(y_train)
```

In [188]:

```
x_train = np.reshape(x_train, (x_train.shape[0],x_train.shape[1],
```

In [189]:

```
model = Sequential()
model.add(LSTM(units=50, return_sequences=True, input_shape=(x_tr
model.add(LSTM(units=50))
model.add(Dense(1))
```

In [190]:

```
model.compile(loss='mean_squared_error', optimizer='adam')
model.fit(x_train, y_train, epochs=20, batch_size=1, verbose=2)
```

```
Epoch 1/20
96s - loss: 0.0011
Epoch 2/20
94s - loss: 5.3325e-04
Epoch 3/20
94s - loss: 3.1337e-04
Epoch 4/20
94s - loss: 2.8767e-04
Epoch 5/20
94s - loss: 2.7148e-04
Epoch 6/20
94s - loss: 2.4441e-04
Epoch 7/20
95s - loss: 2.3399e-04
Epoch 8/20
94s - loss: 2.4268e-04
Epoch 9/20
95s - loss: 2.3833e-04
```

```
Epoch 10/20
95s - loss: 2.1853e-04
Epoch 11/20
94s - loss: 2.3461e-04
Epoch 12/20
95s - loss: 2.0833e-04
Epoch 13/20
94s - loss: 2.2464e-04
Epoch 14/20
96s - loss: 2.1272e-04
Epoch 15/20
95s - loss: 2.1288e-04
Epoch 16/20
94s - loss: 2.2758e-04
Epoch 17/20
94s - loss: 2.2130e-04
Epoch 18/20
96s - loss: 2.1155e-04
Epoch 19/20
93s - loss: 2.0730e-04
Epoch 20/20
95s - loss: 2.0916e-04
```

Out[190]:

```
<keras.callbacks.History at 0x13e6e9b00>
```

In [191]:

```
inputs = new_data[len(new_data) - len(valid) - 60:].values
inputs = inputs.reshape(-1,1)
inputs  = scaler.transform(inputs)
```

In [192]:

```
X_test = []
for i in range(60,inputs.shape[0]):
    X_test.append(inputs[i-60:i,0])
X_test = np.array(X_test)
```

In [193]:

```
X_test = np.reshape(X_test, (X_test.shape[0],X_test.shape[1],1))
closing_price = model.predict(X_test)
closing_price = scaler.inverse_transform(closing_price)
```

In [194]:

```
rms=np.sqrt(np.mean(np.power((valid-closing_price),2)))
rms
```

Out[194]:

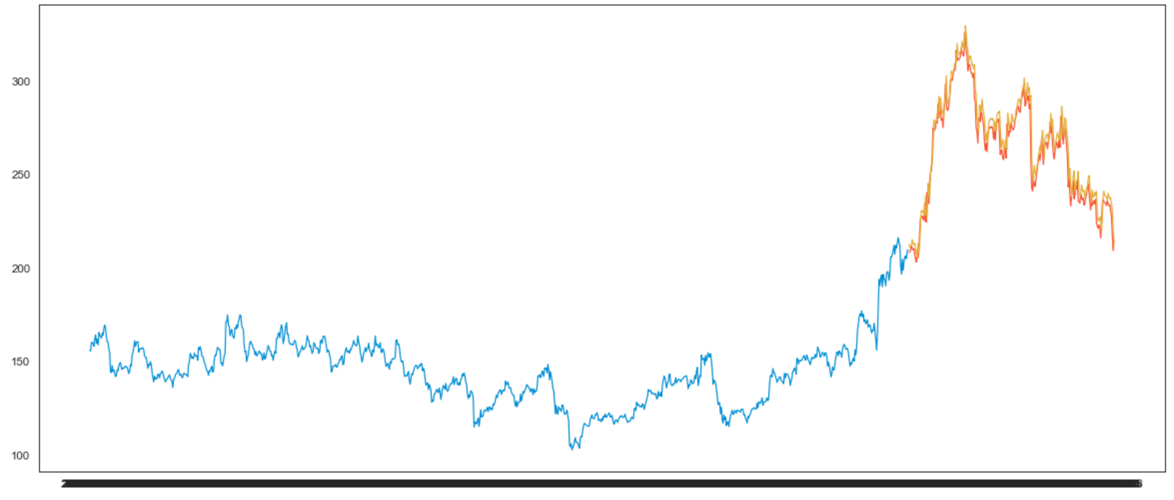
```
7.211366722400344
```

In [195]:

```
train = new_data[:987]
valid = new_data[987:]
valid['Predictions'] = closing_price
plt.plot(train['Close'])
plt.plot(valid[['Close','Predictions']])
```

Out[195]:

```
[<matplotlib.lines.Line2D at 0x140f4d668>,
 <matplotlib.lines.Line2D at 0x140f4d828>]
```




In [ ]:

```
```

In [2]:

```
```

In [2]:

Out[2]:

| | BillDate | Days | BranchName | BillNo | Createddate | Hours | HallNam |
|---|---|---|---|---|---|---|---|
| 0 | 2018-07-01 | Sunday | EDS Branch Hopes | 1 | 2018-07-01 07:49:37.000 | 7 | Swee Ha |
| 1 | 2018-07-01 | Sunday | EDS Branch Hopes | 1 | 2018-07-02 07:49:36.995 | 7 | Swee Ha |
| 2 | 2018-07-01 | Sunday | EDS Branch Hopes | 1 | 2018-07-03 07:49:36.995 | 7 | Swee Ha |
| 3 | 2018-07-01 | Sunday | EDS Branch Hopes | 2 | 2018-07-01 08:02:28.813 | 8 | Swee Ha |
| 4 | 2018-07-01 | Sunday | EDS Branch Hopes | 2 | 2018-07-01 08:02:28.813 | 8 | Swee Ha |

# Data Cleaning

In [3]:

Out[3]:

| | BillDate | Value |
|---|---|---|
| 0 | 2018-07-01 | 50.000 |
| 1 | 2018-07-01 | 249.375 |
| 2 | 2018-07-01 | 210.000 |
| 3 | 2018-07-01 | 192.500 |
| 4 | 2018-07-01 | 140.000 |
| 5 | 2018-07-01 | 115.000 |
| 6 | 2018-07-01 | 30.000 |
| 7 | 2018-07-01 | 30.000 |
| 8 | 2018-07-01 | 32.800 |
| 9 | 2018-07-01 | 118.750 |

In [4]:

```
/usr/local/lib/python3.7/site-packages/ipykernel_lau
ncher.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice fro
m a DataFrame.
Try using .loc[row_indexer,col_indexer] = value inst
ead

See the caveats in the documentation: http://pandas.
pydata.org/pandas-docs/stable/user_guide/indexing.ht
ml#returning-a-view-versus-a-copy
(http://pandas.pydata.org/pandas-docs/stable/user_gu
ide/indexing.html#returning-a-view-versus-a-copy)
  """Entry point for launching an IPython kernel.
/usr/local/lib/python3.7/site-packages/ipykernel_lau
ncher.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice fro
m a DataFrame.
Try using .loc[row_indexer,col_indexer] = value inst
ead
```

In [7]:

Out[7]:

| | BillDate | Value |
|---|---|---|
| 0 | 2018-07-01 | 50.000 |
| 1 | 2018-07-01 | 249.375 |
| 2 | 2018-07-01 | 210.000 |
| 3 | 2018-07-01 | 192.500 |
| 4 | 2018-07-01 | 140.000 |
| 5 | 2018-07-01 | 115.000 |
| 6 | 2018-07-01 | 30.000 |
| 7 | 2018-07-01 | 30.000 |
| 8 | 2018-07-01 | 32.800 |
| 9 | 2018-07-01 | 118.750 |

In [8]:

```

```

Out[8]:

```
BillDate    0
Value       0
dtype: int64
```

In [9]:

```

```

Out[9]:

| | Value |
|---|---|
| count | 442420.000000 |
| mean | 102.204138 |
| std | 275.697064 |
| min | 0.200000 |
| 25% | 41.000000 |
| 50% | 75.000000 |
| 75% | 116.250000 |
| max | 136500.000000 |

In [5]:

```

```

Out[5]:

| | BillDate | Value |
|---|---|---|
| 0 | 2018-07-01 | 124060.591 |
| 1 | 2018-07-02 | 64358.945 |
| 2 | 2018-07-03 | 70038.430 |
| 3 | 2018-07-04 | 68084.010 |
| 4 | 2018-07-05 | 80714.220 |

In [29]:

Out[29]:

```
BillDate    396
Value       396
dtype: int64
```

In [6]:

Out[6]:

| | BillDate | Value |
|---|---|---|
| 124 | 2018-11-03 | 531305.015 |
| 123 | 2018-11-02 | 424899.835 |
| 126 | 2018-11-05 | 420702.985 |
| 125 | 2018-11-04 | 364015.735 |
| 62 | 2018-09-02 | 222505.255 |
| 57 | 2018-08-28 | 218958.670 |
| 391 | 2019-07-28 | 204500.986 |
| 384 | 2019-07-21 | 201396.977 |
| 61 | 2018-09-01 | 198219.040 |
| 377 | 2019-07-14 | 195715.580 |

In [7]:

Out[7]:

```
396
```

In [8]:

```

```

Out[8]:

| | BillDate | Value |
|---|---|---|
| 0 | 2018-07-01 | 124060.591 |
| 1 | 2018-07-02 | 64358.945 |
| 2 | 2018-07-03 | 70038.430 |
| 3 | 2018-07-04 | 68084.010 |
| 4 | 2018-07-05 | 80714.220 |
| ... | ... | ... |
| 391 | 2019-07-27 | 177738.250 |
| 392 | 2019-07-28 | 204500.986 |
| 393 | 2019-07-29 | 107830.368 |
| 394 | 2019-07-30 | 126576.676 |
| 395 | 2019-07-31 | 131196.725 |

396 rows × 2 columns

In [9]:

```

```

Out[9]:

| | BillDate | Value | day_week |
|---|---|---|---|
| 0 | 2018-07-01 | 124060.591 | 6 |
| 1 | 2018-07-02 | 64358.945 | 0 |
| 2 | 2018-07-03 | 70038.430 | 1 |
| 3 | 2018-07-04 | 68084.010 | 2 |
| 4 | 2018-07-05 | 80714.220 | 3 |
| ... | ... | ... | ... |
| 391 | 2019-07-27 | 177738.250 | 5 |
| 392 | 2019-07-28 | 204500.986 | 6 |
| 393 | 2019-07-29 | 107830.368 | 0 |
| 394 | 2019-07-30 | 126576.676 | 1 |
| 395 | 2019-07-31 | 131196.725 | 2 |

396 rows × 3 columns

In [10]:

```

```

Out[10]:

| | BillDate | Value | day_week |
|---|---|---|---|
| 38 | 2018-08-08 | NaN | 2 |

In [10]:

```

```

Out[10]:

| | BillDate | Value | day_week |
|---|---|---|---|
| 3 | 2018-07-04 | 68084.010 | 2 |
| 10 | 2018-07-11 | 74548.825 | 2 |
| 17 | 2018-07-18 | 76797.780 | 2 |
| 24 | 2018-07-25 | 70109.120 | 2 |
| 31 | 2018-08-01 | 74221.134 | 2 |
| 38 | 2018-08-08 | NaN | 2 |
| 45 | 2018-08-15 | 123470.605 | 2 |
| 52 | 2018-08-22 | 104626.485 | 2 |
| 59 | 2018-08-29 | 82487.775 | 2 |
| 66 | 2018-09-05 | 75402.590 | 2 |
| 73 | 2018-09-12 | 103570.726 | 2 |

In [11]:

```

```

Out[11]:

```
day_week
0     98358.634298
1    103612.778509
2     98216.076929
3    103373.065473
4    113821.672750
5    138893.000357
6    144978.426684
Name: Value, dtype: float64
```

In [11]:

```

```

In [11]:

```

```

Out[11]:

| | BillDate | Value | day_week |
|---|---|---|---|
| 3 | 2018-07-04 | 68084.010000 | 2 |
| 10 | 2018-07-11 | 74548.825000 | 2 |
| 17 | 2018-07-18 | 76797.780000 | 2 |
| 24 | 2018-07-25 | 70109.120000 | 2 |
| 31 | 2018-08-01 | 74221.134000 | 2 |
| 38 | 2018-08-08 | 98216.076929 | 2 |
| 45 | 2018-08-15 | 123470.605000 | 2 |

In [12]:

```

```

Out[12]:

| | BillDate | Value | day_week |
|---|---|---|---|
| 125 | 2018-11-03 | 531305.015 | 5 |
| 124 | 2018-11-02 | 424899.835 | 4 |
| 127 | 2018-11-05 | 420702.985 | 0 |
| 126 | 2018-11-04 | 364015.735 | 6 |
| 63 | 2018-09-02 | 222505.255 | 6 |
| 58 | 2018-08-28 | 218958.670 | 1 |
| 392 | 2019-07-28 | 204500.986 | 6 |
| 385 | 2019-07-21 | 201396.977 | 6 |
| 62 | 2018-09-01 | 198219.040 | 5 |
| 378 | 2019-07-14 | 195715.580 | 6 |

In [16]:

```

```

In [17]:

```
```

Out[17]:

| | BillDate | Value | day_week |
|---|---|---|---|
| 63 | 2018-09-02 | 222505.255 | 6 |
| 58 | 2018-08-28 | 218958.670 | 1 |
| 392 | 2019-07-28 | 204500.986 | 6 |
| 385 | 2019-07-21 | 201396.977 | 6 |
| 62 | 2018-09-01 | 198219.040 | 5 |
| 378 | 2019-07-14 | 195715.580 | 6 |
| 384 | 2019-07-20 | 191041.914 | 5 |
| 123 | 2018-11-01 | 185600.850 | 3 |
| 363 | 2019-06-29 | 185372.150 | 5 |
| 357 | 2019-06-23 | 185323.505 | 6 |

In [19]:

```
```

Out[19]:

```
BillDate    0
Value       4
day_week    0
dtype: int64
```

In [20]:

```
```

In [21]:

```
```

Out[21]:

```
BillDate    0
Value       0
day_week    0
dtype: int64
```

In [22]:

```

```

Out[22]:

| | BillDate | Value | day_week |
|---|---|---|---|
| 63 | 2018-09-02 | 222505.255 | 6 |
| 58 | 2018-08-28 | 218958.670 | 1 |
| 392 | 2019-07-28 | 204500.986 | 6 |
| 385 | 2019-07-21 | 201396.977 | 6 |
| 62 | 2018-09-01 | 198219.040 | 5 |
| 378 | 2019-07-14 | 195715.580 | 6 |
| 384 | 2019-07-20 | 191041.914 | 5 |
| 123 | 2018-11-01 | 185600.850 | 3 |
| 363 | 2019-06-29 | 185372.150 | 5 |
| 357 | 2019-06-23 | 185323.505 | 6 |

In [24]:

```

```

Out[24]:

| | BillDate | Value | day_week |
|---|---|---|---|
| 124 | 2018-11-02 | 108165.706164 | 4 |

In [34]:

```

```

Out[34]:

```
day_week
0     92602.485179
1    103612.778509
2     98216.076929
3    103373.065473
4    108165.706164
5    131758.236455
6    141067.046179
Name: Value, dtype: float64
```

In [35]:

Out[35]:

| | BillDate | Value | day_week |
|---|---|---|---|
| 2 | 2018-07-03 | 70038.430 | 1 |
| 9 | 2018-07-10 | 78692.480 | 1 |
| 16 | 2018-07-17 | 84405.934 | 1 |
| 23 | 2018-07-24 | 110241.930 | 1 |
| 30 | 2018-07-31 | 76438.990 | 1 |
| 37 | 2018-08-07 | 98080.701 | 1 |
| 44 | 2018-08-14 | 80329.940 | 1 |
| 51 | 2018-08-21 | 96127.680 | 1 |
| 58 | 2018-08-28 | 218958.670 | 1 |
| 65 | 2018-09-04 | 88757.860 | 1 |
| 72 | 2018-09-11 | 72812.010 | 1 |
| 79 | 2018-09-18 | 72271.715 | 1 |
| 86 | 2018-09-25 | 60982.520 | 1 |
| 93 | 2018-10-02 | 116042.144 | 1 |
| 100 | 2018-10-09 | 69669.185 | 1 |
| 107 | 2018-10-16 | 91879.850 | 1 |
| 114 | 2018-10-23 | 69535.400 | 1 |
| 121 | 2018-10-30 | 121823.610 | 1 |
| 128 | 2018-11-06 | 133161.210 | 1 |
| 135 | 2018-11-13 | 63883.575 | 1 |

In [26]:

In [3]:

```
################################################################################
```

# Data Analysis

In [4]:

```
```

In [5]:

```
```

In [6]:

```
```

In [7]:

```
```

Out[7]:

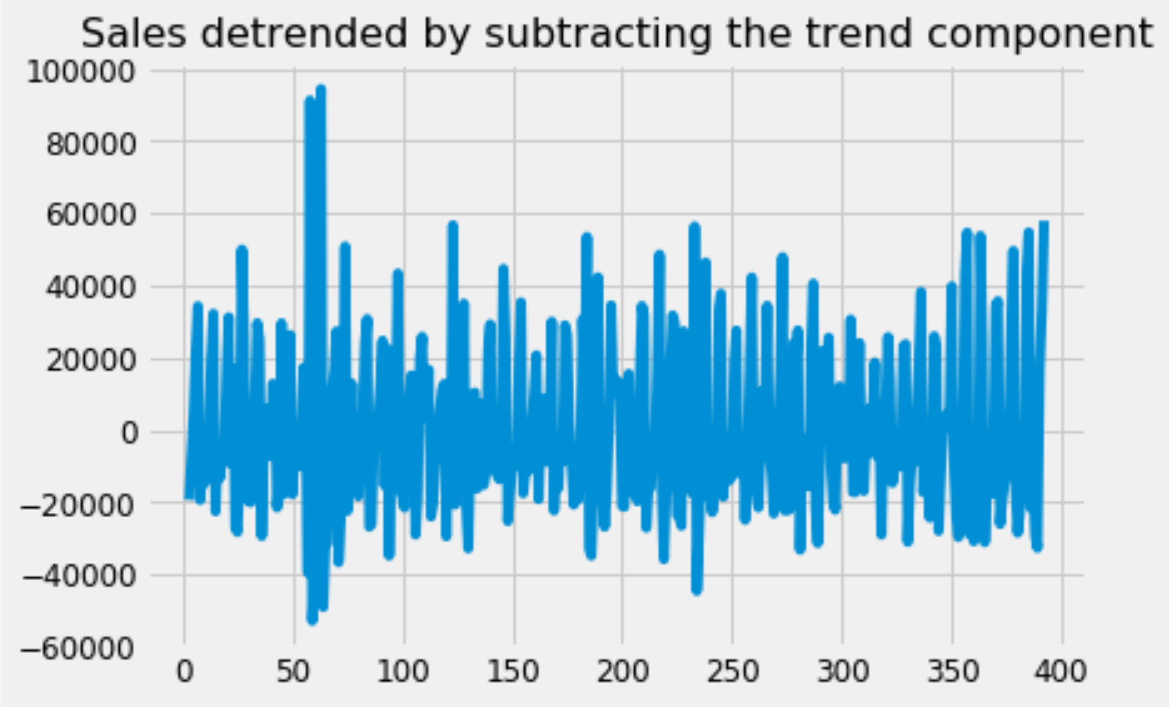
| | Unnamed: 0 | BillDate | Value | day_week |
|---|---|---|---|---|
| 0 | 0 | 2018-07-01 | 124060.591 | 6 |
| 1 | 1 | 2018-07-02 | 64358.945 | 0 |
| 2 | 2 | 2018-07-03 | 70038.430 | 1 |
| 3 | 3 | 2018-07-04 | 68084.010 | 2 |
| 4 | 4 | 2018-07-05 | 80714.220 | 3 |
| ... | ... | ... | ... | ... |
| 391 | 391 | 2019-07-27 | 177738.250 | 5 |
| 392 | 392 | 2019-07-28 | 204500.986 | 6 |
| 393 | 393 | 2019-07-29 | 107830.368 | 0 |
| 394 | 394 | 2019-07-30 | 126576.676 | 1 |
| 395 | 395 | 2019-07-31 | 131196.725 | 2 |

396 rows × 4 columns

In [15]:

Out[15]:

```
Text(0.5, 1.0, 'Sales detrended by subtracting the t
rend component')
```




In [29]:

In [30]:

Out[30]:

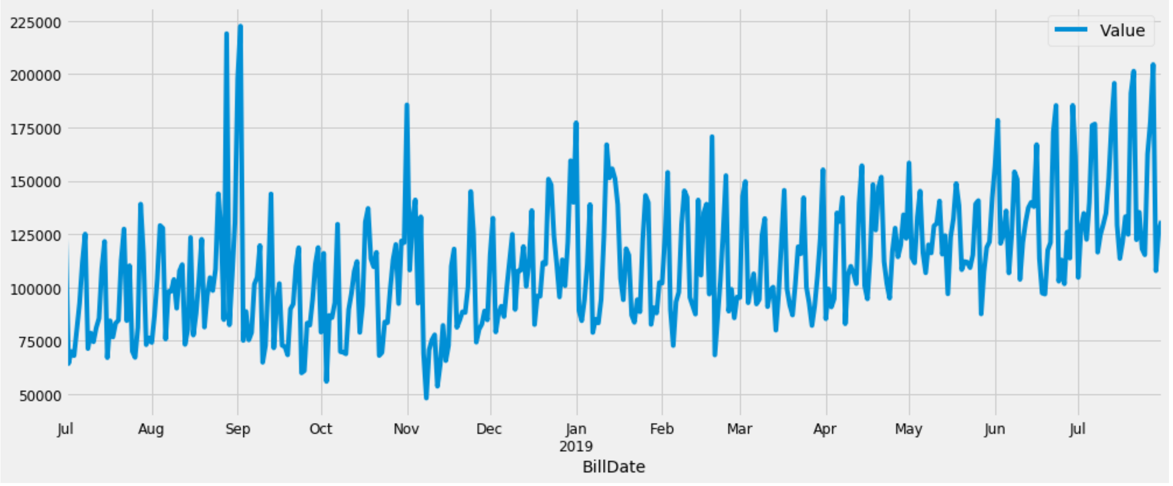
```
DatetimeIndex(['2018-07-01', '2018-07-02', '2018-07-
03', '2018-07-04',
               '2018-07-05', '2018-07-06', '2018-07-
07', '2018-07-08',
               '2018-07-09', '2018-07-10',
               ...
               '2019-07-22', '2019-07-23', '2019-07-
24', '2019-07-25',
               '2019-07-26', '2019-07-27', '2019-07-
28', '2019-07-29',
               '2019-07-30', '2019-07-31'],
              dtype='datetime64[ns]', name='BillDate
', length=396, freq=None)
```

In [31]:

```
BillDate
2018-07-01     90753.347661
2018-08-01    105103.745772
2018-09-01     98366.006667
2018-10-01     97200.652774
2018-11-01     94177.810983
2018-12-01    111739.175726
2019-01-01    112206.810065
2019-02-01    111379.844714
2019-03-01    108786.581774
2019-04-01    117044.663467
2019-05-01    123073.983710
2019-06-01    133312.133400
2019-07-01    142825.586806
Freq: MS, Name: Value, dtype: float64
```

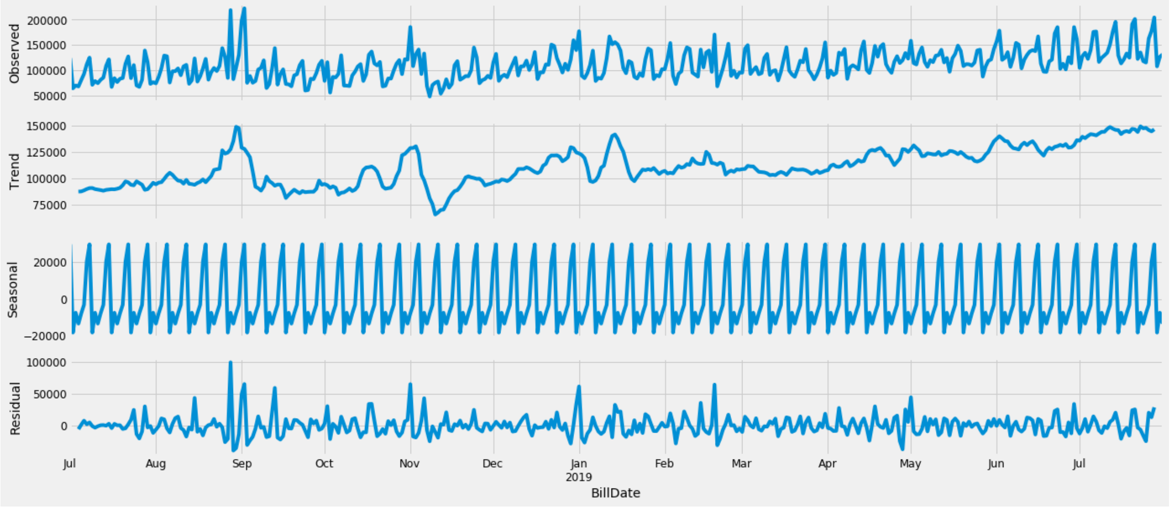
# Time Series Plot

In [32]:

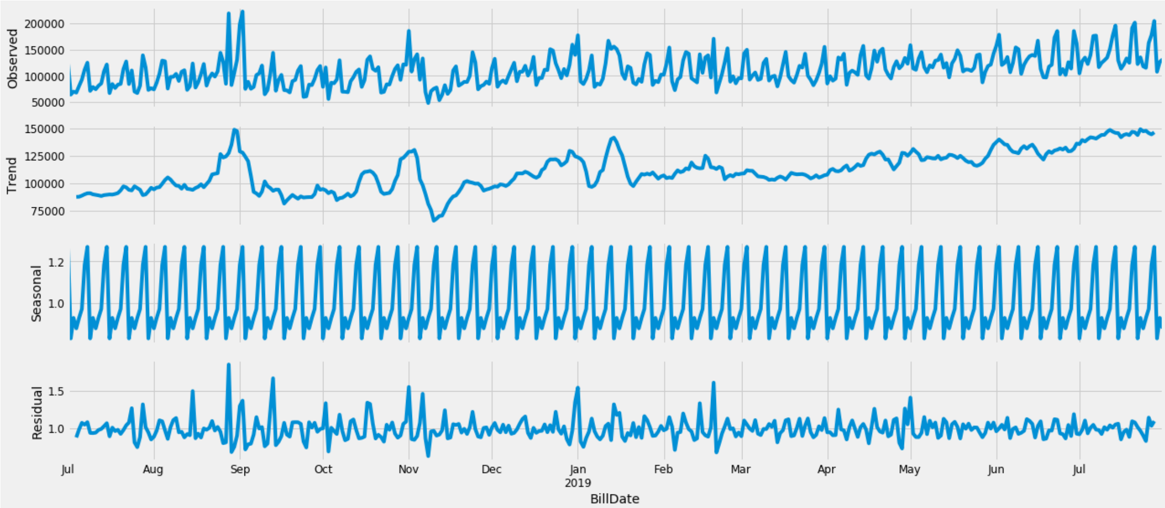



# Observing Trend and Seasonality

In [33]:



In [36]:




In [22]:

Out[22]:

| | BillDate | Value | day_week |
|---|---|---|---|
| 0 | 2018-07-01 | 124060.591 | 6 |
| 1 | 2018-07-02 | 64358.945 | 0 |
| 2 | 2018-07-03 | 70038.430 | 1 |
| 3 | 2018-07-04 | 68084.010 | 2 |
| 4 | 2018-07-05 | 80714.220 | 3 |

In [37]:

Out[37]:

| | Value | day_week |
|---|---|---|
| **BillDate** | | |
| 2018-07-01 | 124060.591 | 6 |
| 2018-07-02 | 64358.945 | 0 |
| 2018-07-03 | 70038.430 | 1 |
| 2018-07-04 | 68084.010 | 2 |
| 2018-07-05 | 80714.220 | 3 |

In [38]:

## Normality Test

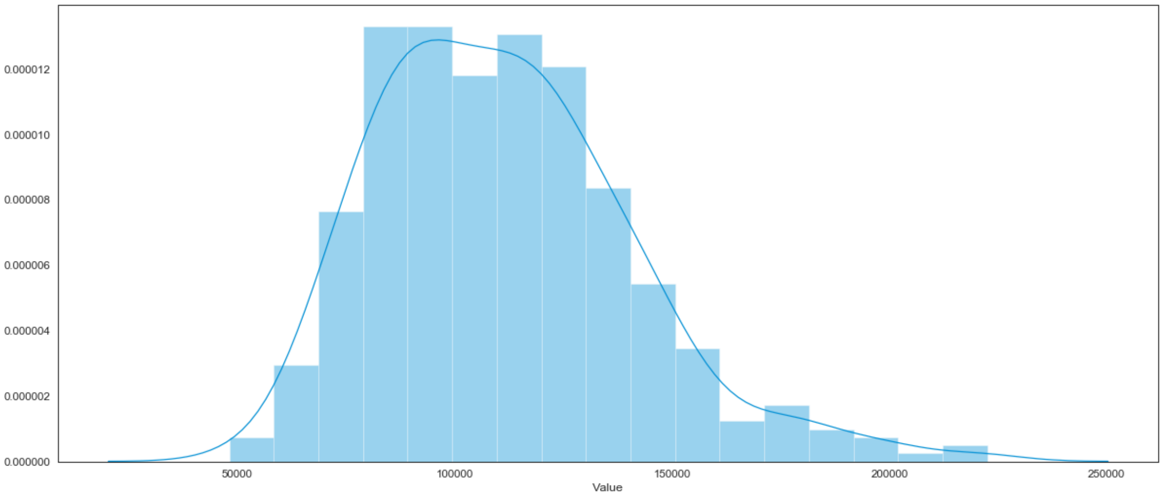
In [39]:

```
Statistics=39.268, p=0.000
Data does not look Normally Distributed (Reject H0)
```

## Kurtosis and Skewness

In [40]:

```
Kurtosis of normal distribution: 0.8047300298703224
Skewness of normal distribution: 0.7665756612800276
```

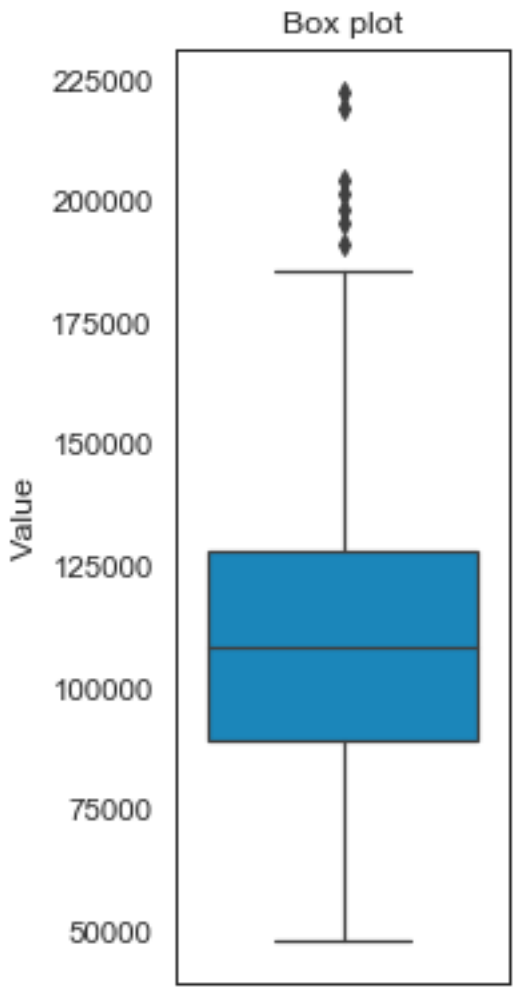


In [41]:

```
```

Out[41]:

```
Text(0.5, 1.0, 'Box plot')
```

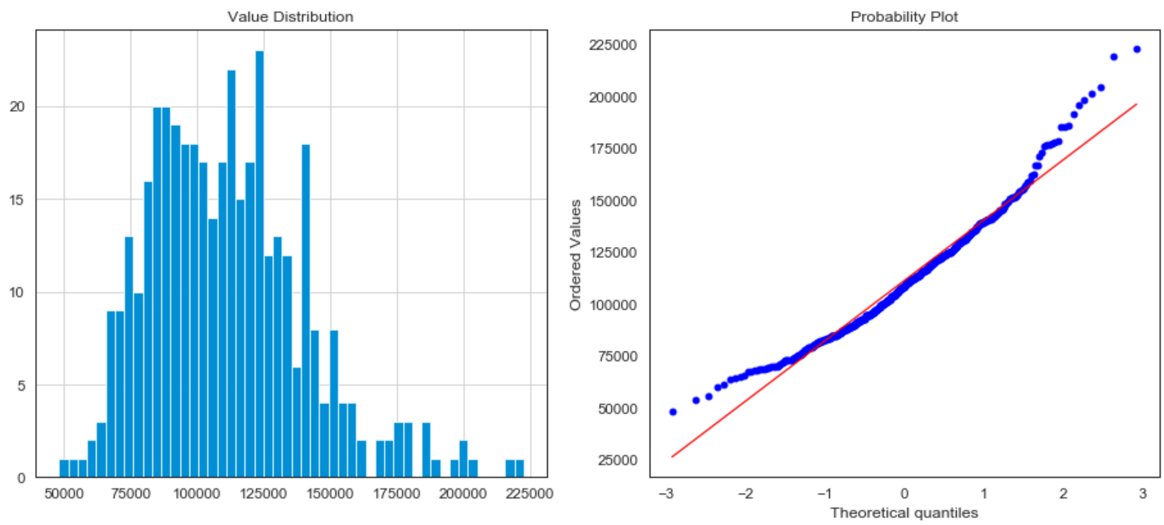



### Distribution and Normal probability plots

In [42]:

Out[42]:

| | count | mean | std | min | 25% | |
|---|---|---|---|---|---|---|
| Value | 396.0000 | 111232.4255 | 29425.4967 | 48189.8500 | 89019.5175 | 1084 |

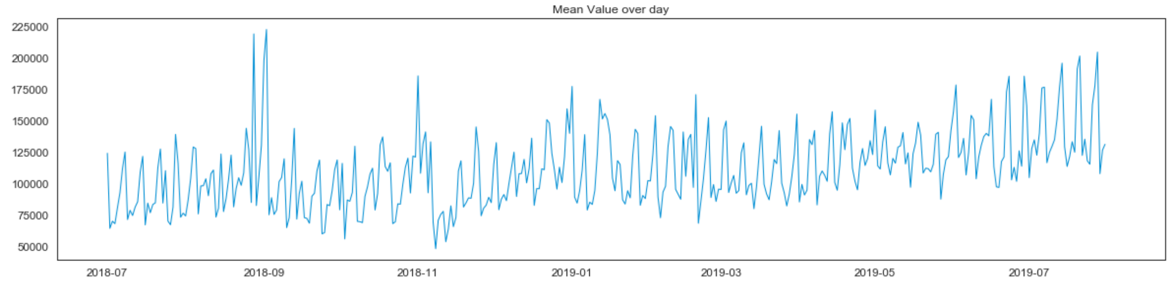


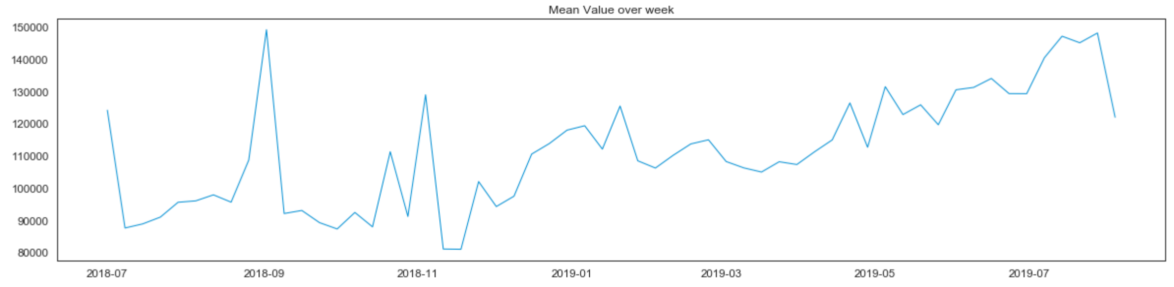

## Average Value Over Day, Week, and Month

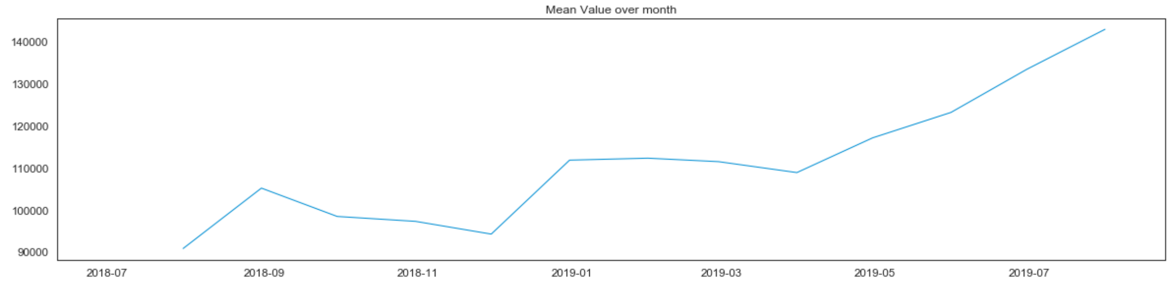
In [43]:

In [44]:

In [45]:




# Checking whether Data is Stationary or not

# Dickey-Fuller test

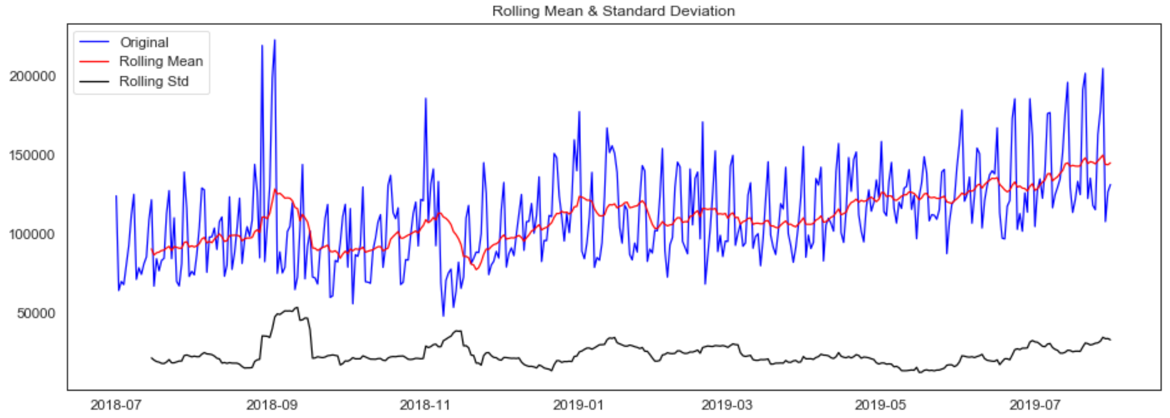
In [46]:

Out[46]:

| | Value |
|---|---|
| **BillDate** | |
| 2018-07-01 | 124060.5910 |
| 2018-07-02 | 64358.9450 |
| 2018-07-03 | 70038.4300 |
| 2018-07-04 | 68084.0100 |
| 2018-07-05 | 80714.2200 |

In [47]:

In [48]:




```
<Results of Dickey-Fuller Test>
Test Statistic                  -1.6867
p-value                          0.4380
#Lags Used                      13.0000
Number of Observations Used    382.0000
Critical Value (1%)             -3.4476
Critical Value (5%)             -2.8691
Critical Value (10%)            -2.5708
dtype: float64
```

Note:- Null Hypothesis for Dickey-Fuller Test: Data is Not Stationary

Here, Test Statistic value is less than Critical Value (1%, 5% and 10%), So we can reject Null Hypothesis.

Dickey-Fuller test telling that the data is Stationary

# KPSS Test

In [49]:

In [50]:

```
Results of KPSS Test:
Test Statistic            0.1403
p-value                   0.0605
Lags Used                17.0000
Critical Value (10%)      0.1190
Critical Value (5%)       0.1460
Critical Value (2.5%)     0.1760
Critical Value (1%)       0.2160
dtype: float64
```

Note:- Null Hypothesis for KPSS test: Data is Stationary

Test for stationarity: If the test statistic is greater than the critical value, we reject the null hypothesis (series is not stationary).

Here, test statistic is smaller than the critical value, "We Accept Null Hypothesis" and the data has "Stationarity".

Types of Stationarity:

```
1. Strict Stationary

2. Trend Stationary

3. Difference Stationary
```

Case 1: Both tests conclude that the series is not stationary -> series is not stationary

Case 2: Both tests conclude that the series is stationary -> series is stationary

Case 3: KPSS = stationary and ADF = not stationary -> trend stationary, remove the trend to make series strict stationary

Case 4: KPSS = not stationary and ADF = stationary -> difference stationary, use differencing to make series strict stationary

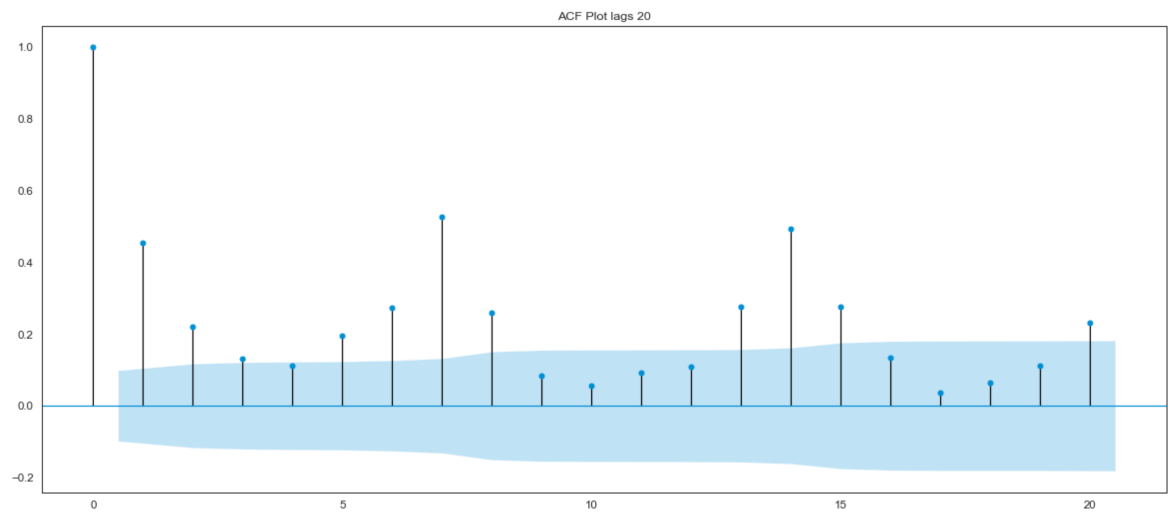
## ACF and PACF Plots till lag 20

In [51]:

```

```

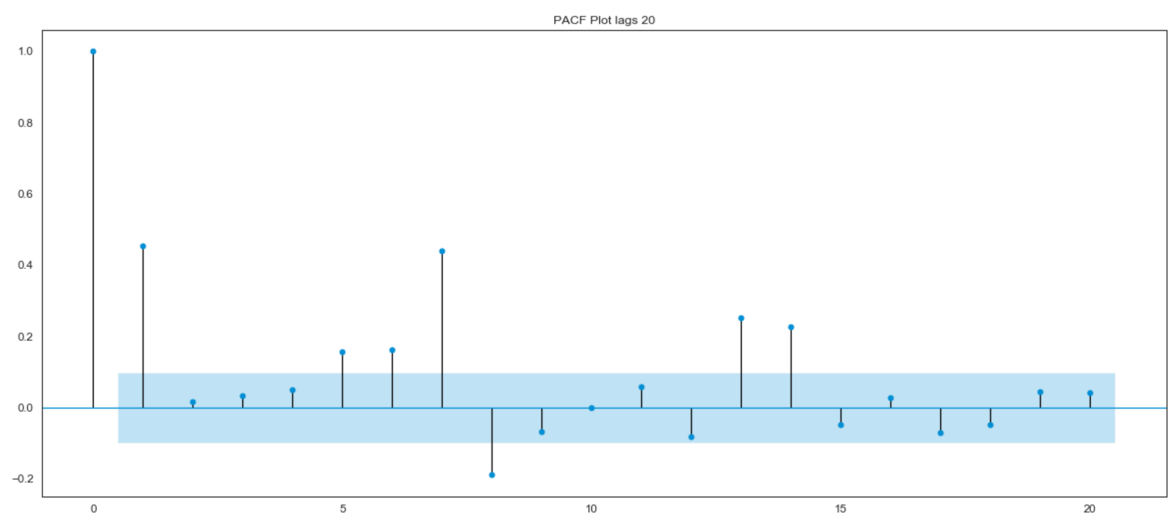
In [52]:

```

```




In [53]:

```

```



In [49]:

```

```

Out[49]:

| BillDate | Value |
|---|---|
| 2018-07-01 | 124060.5910 |
| 2018-07-02 | 64358.9450 |
| 2018-07-03 | 70038.4300 |
| 2018-07-04 | 68084.0100 |
| 2018-07-05 | 80714.2200 |
| 2018-07-06 | 92766.6620 |
| 2018-07-07 | 111833.9990 |
| 2018-07-08 | 125058.1600 |
| 2018-07-09 | 71373.8750 |
| 2018-07-10 | 78692.4800 |

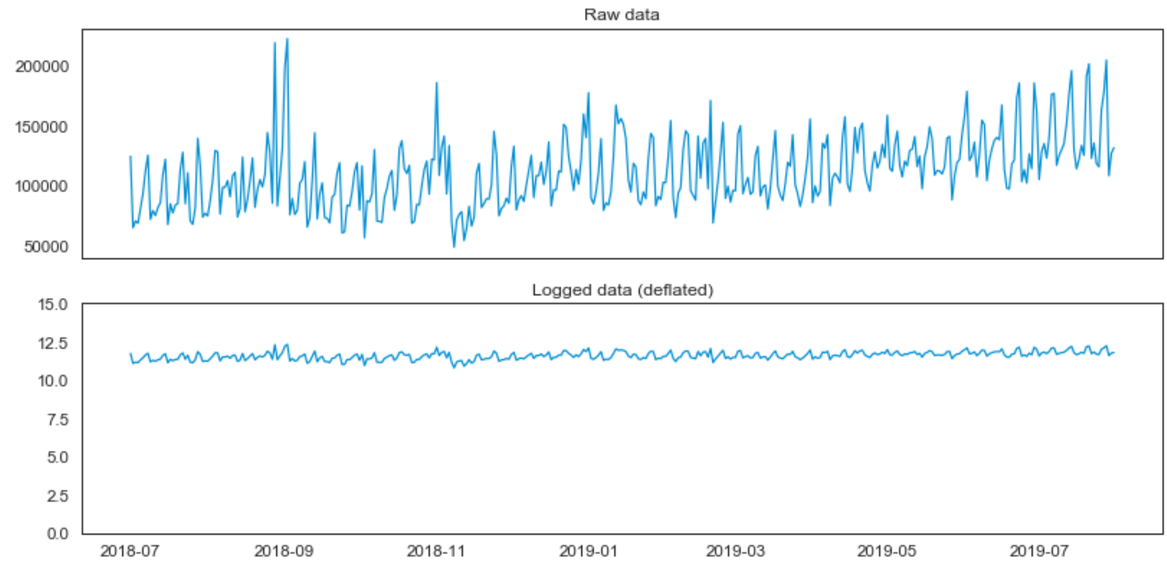
```
##########################################################################
```

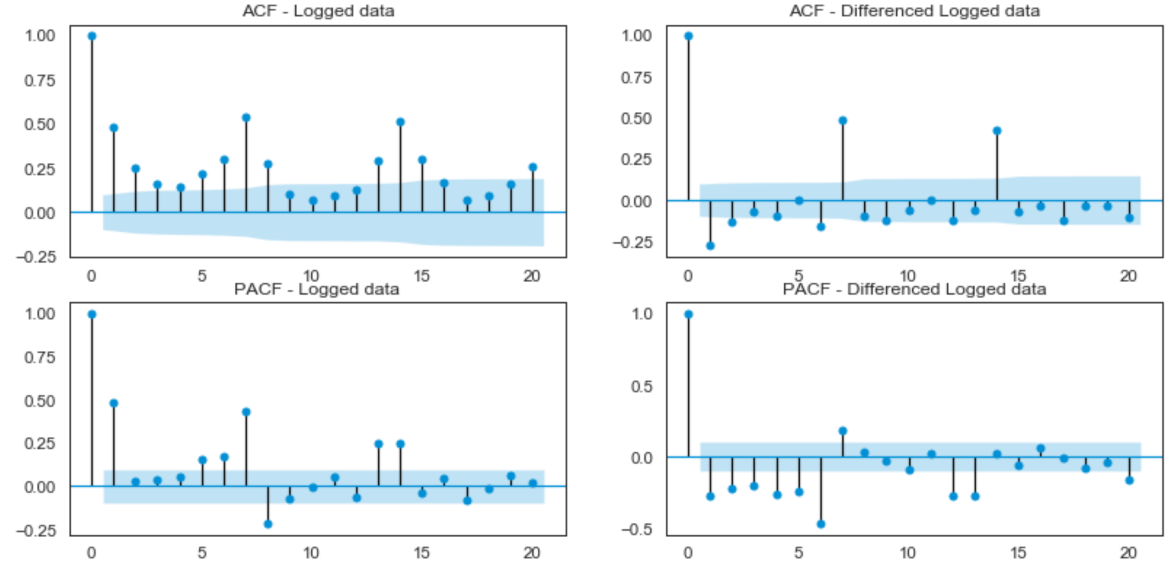
# ARIMA

In [71]:

```

```

In [72]:



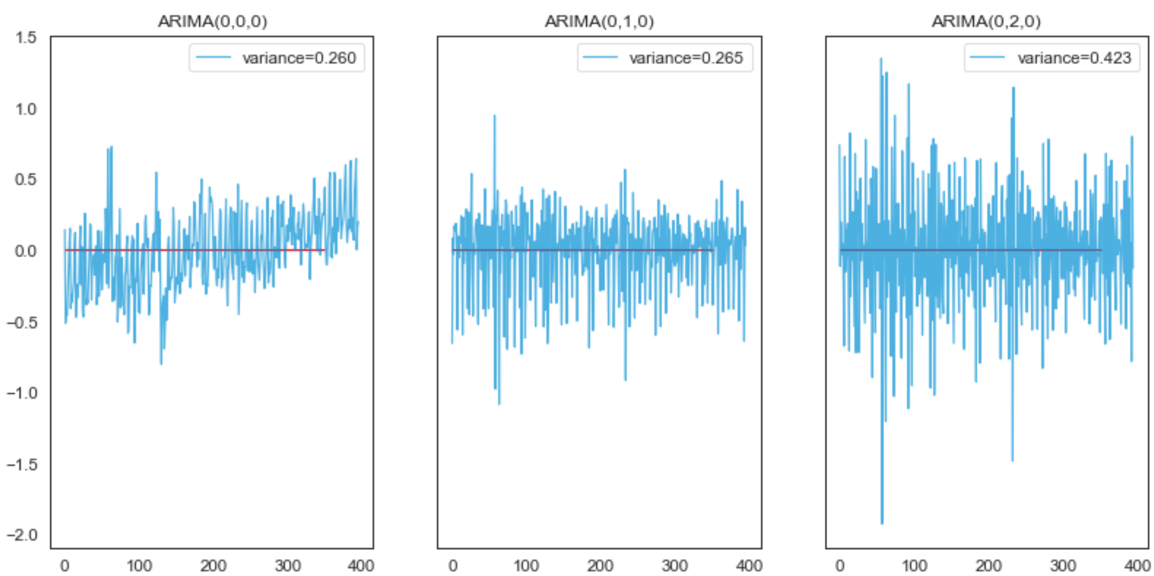




## Choosing the differencing order

In [73]:

```
                              ARMA Model Results
====================================================
==========================
Dep. Variable:                    Value   No. Observat
ions:                      396
Model:                       ARMA(0, 0)   Log Likeliho
od                     -28.231
Method:                             css   S.D. of inno
vations                  0.260
Date:                  Tue, 05 Nov 2019   AIC
60.462
Time:                          15:37:13   BIC
68.425
Sample:                      07-01-2018   HQIC
63.617
                           - 07-31-2019
====================================================
==========================
                 coef    std err          z      P>|
z|        [0.025      0.975]
```

In [74]:



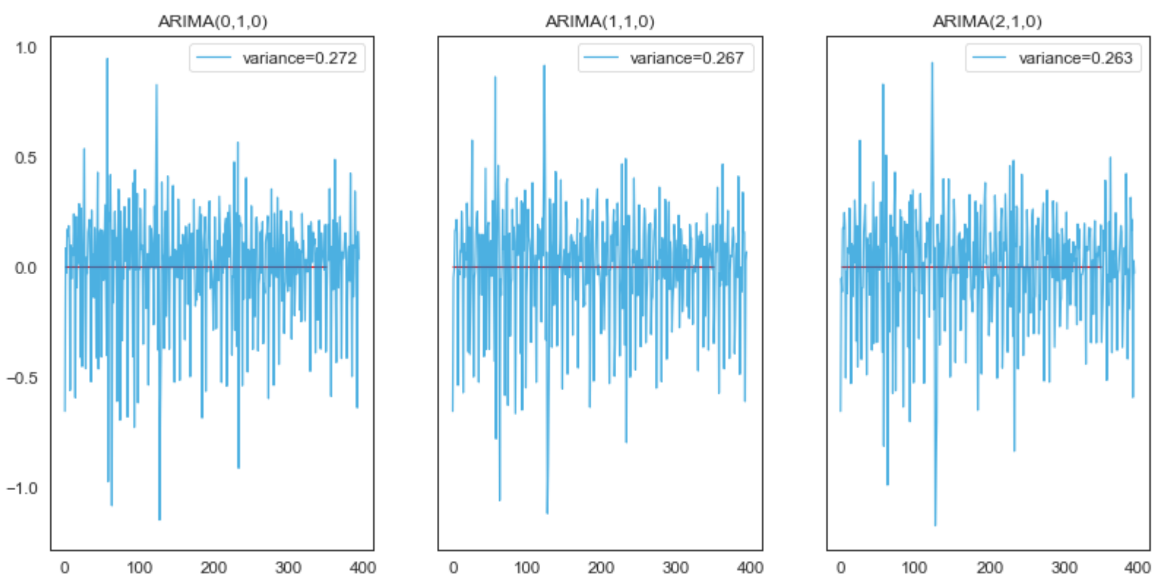

Note: From the above results, we can see that the AIC value is low for "Diffrencing order 1".

## Choosing the MA order

In [76]:

```
                              ARMA Model Results
==============================================================================
Dep. Variable:                  Value   No. Observations:                  396
Model:                     ARMA(0, 0)   Log Likelihood                 -28.231
Method:                           css   S.D. of innovations              0.260
Date:                Tue, 05 Nov 2019   AIC                             60.462
Time:                        15:38:18   BIC                             68.425
Sample:                    07-01-2018   HQIC                            63.617
                         - 07-31-2019
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
```

In [58]:




## Choosing the AR order

In [78]:

```
                              ARMA Model Results
=====================================================
==========================
Dep. Variable:                    Value   No. Observat
ions:                      396
Model:                       ARMA(1, 0)   Log Likeliho
od                      23.708
Method:                         css-mle   S.D. of inno
vations                  0.228
Date:                  Tue, 05 Nov 2019   AIC
-41.416
Time:                          15:40:02   BIC
-29.471
Sample:                      07-01-2018   HQIC
-36.684
                           - 07-31-2019
=====================================================
===========================
                  coef    std err          z      P>
```

## Model

In [55]:

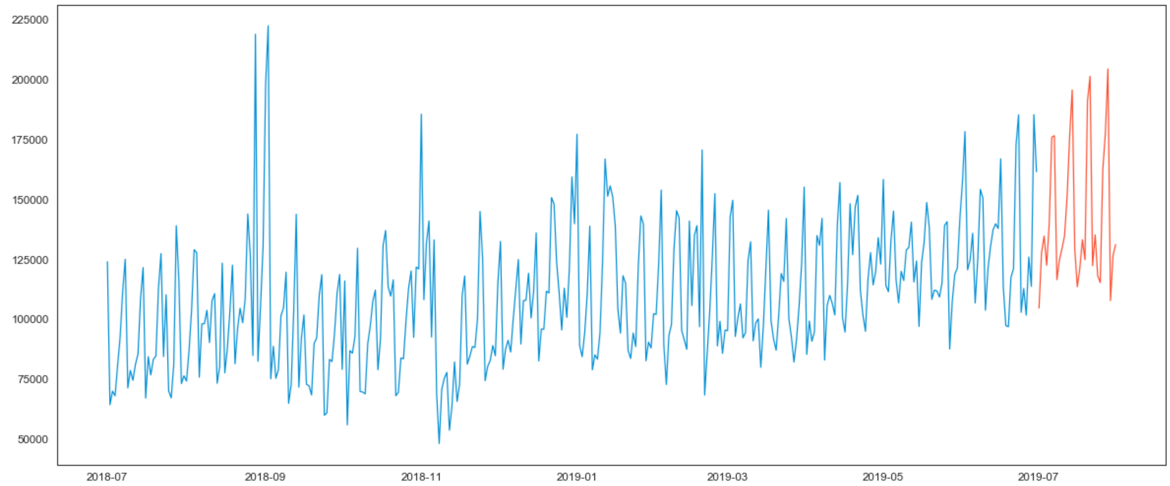
Out[55]:

| | Value |
|---|---|
| **BillDate** | |
| 2018-07-01 | 124060.5910 |
| 2018-07-02 | 64358.9450 |
| 2018-07-03 | 70038.4300 |
| 2018-07-04 | 68084.0100 |
| 2018-07-05 | 80714.2200 |

In [56]:

```
(365, 1)
(31, 1)
```

In [57]:




# Auto_Arima

In [58]:

In [59]:

```
Fit ARIMA: order=(0, 1, 0) seasonal_order=(0, 1, 0, 
7); AIC=8416.003, BIC=8423.759, Fit time=0.031 secon
ds
Fit ARIMA: order=(1, 1, 0) seasonal_order=(1, 1, 0, 
7); AIC=8309.917, BIC=8325.427, Fit time=0.156 secon
ds
Fit ARIMA: order=(0, 1, 1) seasonal_order=(0, 1, 1, 
7); AIC=8162.646, BIC=8178.157, Fit time=1.135 secon
ds
Fit ARIMA: order=(0, 1, 1) seasonal_order=(1, 1, 1, 
7); AIC=8201.163, BIC=8220.552, Fit time=0.738 secon
ds
Fit ARIMA: order=(0, 1, 1) seasonal_order=(0, 1, 0, 
7); AIC=8356.116, BIC=8367.749, Fit time=0.062 secon
ds
Fit ARIMA: order=(0, 1, 1) seasonal_order=(0, 1, 2, 
7); AIC=8201.156, BIC=8220.545, Fit time=0.981 secon
ds
Fit ARIMA: order=(0, 1, 1) seasonal_order=(1, 1, 2, 
7); AIC=8201.847, BIC=8225.113, Fit time=1.104 secon
```

In [60]:

Out[60]:

Statespace Model Results

| | | | |
|---|---|---|---|
| **Dep. Variable:** | y | **No. Observations:** | 365 |
| **Model:** | SARIMAX(0, 1, 2)x(0, 1, 1, 7) | **Log Likelihood** | -4053.998 |
| **Date:** | Tue, 05 Nov 2019 | **AIC** | 8117.996 |
| **Time:** | 10:36:26 | **BIC** | 8137.385 |
| **Sample:** | 0 | **HQIC** | 8125.708 |
| | - 365 | | |
| **Covariance Type:** | opg | | |

| | coef | std err | z | P>\|z\| | [0.025 | 0.975] |
|---|---|---|---|---|---|---|
| **intercept** | -23.1829 | 24.431 | -0.949 | 0.343 | -71.068 | 24.702 |
| **ma.L1** | -0.5272 | 0.034 | -15.390 | 0.000 | -0.594 | -0.460 |
| **ma.L2** | -0.1901 | 0.043 | -4.416 | 0.000 | -0.274 | -0.106 |
| **ma.S.L7** | -0.9914 | 0.050 | -19.690 | 0.000 | -1.090 | -0.893 |
| **sigma2** | 4.04e+08 | 3.57e-07 | 1.13e+15 | 0.000 | 4.04e+08 | 4.04e+08 |

| | | | |
|---|---|---|---|
| **Ljung-Box (Q):** | 59.32 | **Jarque-Bera (JB):** | 210.40 |
| **Prob(Q):** | 0.03 | **Prob(JB):** | 0.00 |
| **Heteroskedasticity (H):** | 0.36 | **Skew:** | 0.53 |
| **Prob(H) (two-sided):** | 0.00 | **Kurtosis:** | 6.61 |

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
[2] Covariance matrix is singular or near-singular, with condition number 2.73e+31. Standard errors may be unstable.

In [61]:

In [62]:

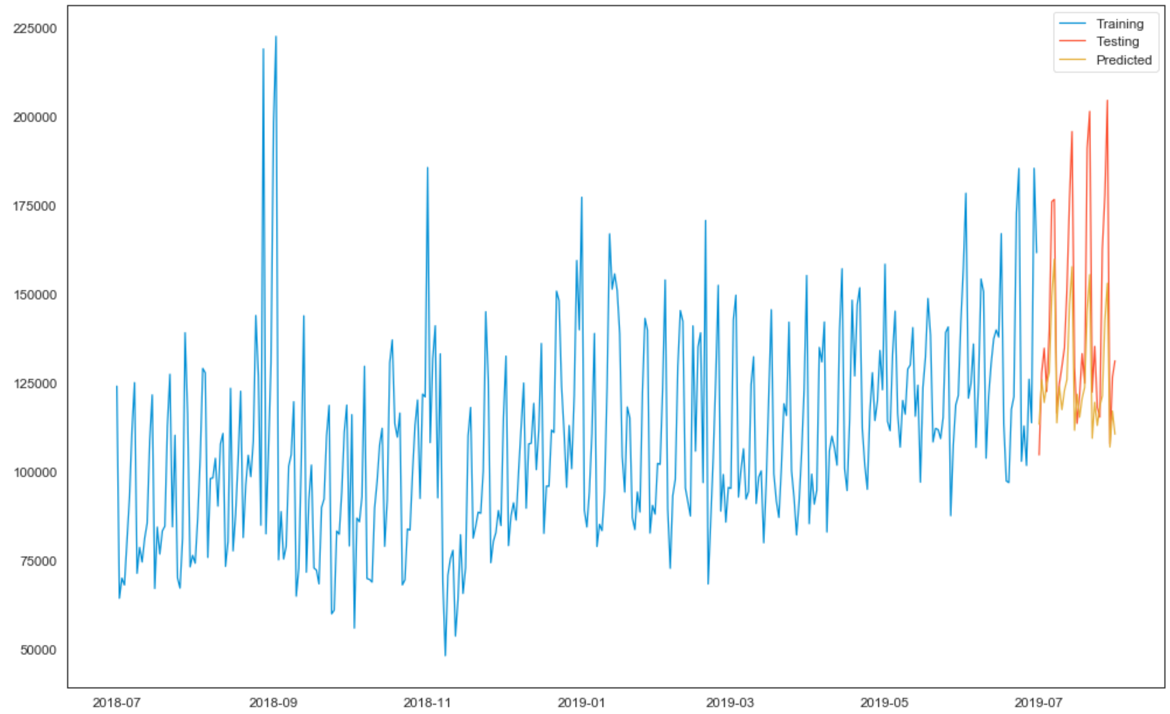
In [63]:

Out[63]:

| | Forecasted |
|---|---|
| **BillDate** | |
| 2019-07-01 | 113339.0241 |
| 2019-07-02 | 125883.7944 |
| 2019-07-03 | 119444.8986 |
| 2019-07-04 | 124833.1830 |
| 2019-07-05 | 127928.8809 |
| 2019-07-06 | 149259.4597 |
| 2019-07-07 | 159766.6176 |
| 2019-07-08 | 113736.6994 |
| 2019-07-09 | 123918.8396 |
| 2019-07-10 | 117456.7609 |
| 2019-07-11 | 122821.8623 |
| 2019-07-12 | 125894.3772 |
| 2019-07-13 | 147201.7731 |
| 2019-07-14 | 157685.7481 |
| 2019-07-15 | 111632.6469 |
| 2019-07-16 | 121791.6042 |
| 2019-07-17 | 115306.3425 |
| 2019-07-18 | 120648.2609 |
| 2019-07-19 | 123697.5929 |
| 2019-07-20 | 144981.8059 |
| 2019-07-21 | 155442.5979 |
| 2019-07-22 | 109366.3138 |

| | |
|---|---|
| 2019-07-23 | 119502.0881 |
| 2019-07-24 | 112993.6435 |
| 2019-07-25 | 118312.3790 |
| 2019-07-26 | 121338.5281 |
| 2019-07-27 | 142599.5580 |
| 2019-07-28 | 153037.1672 |
| 2019-07-29 | 106937.7001 |
| 2019-07-30 | 117050.2915 |
| 2019-07-31 | 110518.6639 |

In [64]:




In [65]:

Out[65]:

| | Value | forecasted | error |
|---|---|---|---|
| **BillDate** | | | |
| 2019-07-01 | 104695.7250 | 113339.0241 | -8643.2991 |
| 2019-07-02 | 127803.5900 | 125883.7944 | 1919.7956 |
| 2019-07-03 | 134738.3800 | 119444.8986 | 15293.4814 |
| 2019-07-04 | 122540.8750 | 124833.1830 | -2292.3080 |

| | | | |
|---|---|---|---|
| 2019-07-05 | 139906.2360 | 127928.8809 | 11977.3551 |
| 2019-07-06 | 175973.6300 | 149259.4597 | 26714.1703 |
| 2019-07-07 | 176618.0850 | 159766.6176 | 16851.4674 |
| 2019-07-08 | 116535.6400 | 113736.6994 | 2798.9406 |
| 2019-07-09 | 124766.9700 | 123918.8396 | 848.1304 |
| 2019-07-10 | 129517.0500 | 117456.7609 | 12060.2891 |
| 2019-07-11 | 134813.2500 | 122821.8623 | 11991.3877 |
| 2019-07-12 | 151176.9640 | 125894.3772 | 25282.5868 |
| 2019-07-13 | 176453.4200 | 147201.7731 | 29251.6469 |
| 2019-07-14 | 195715.5800 | 157685.7481 | 38029.8319 |
| 2019-07-15 | 129262.1900 | 111632.6469 | 17629.5431 |
| 2019-07-16 | 113593.5720 | 121791.6042 | -8198.0322 |
| 2019-07-17 | 121408.7200 | 115306.3425 | 6102.3775 |
| 2019-07-18 | 133213.1000 | 120648.2609 | 12564.8391 |
| 2019-07-19 | 124875.5150 | 123697.5929 | 1177.9221 |
| 2019-07-20 | 191041.9140 | 144981.8059 | 46060.1081 |
| 2019-07-21 | 201396.9770 | 155442.5979 | 45954.3791 |
| 2019-07-22 | 122373.8600 | 109366.3138 | 13007.5462 |
| 2019-07-23 | 135259.0700 | 119502.0881 | 15756.9819 |
| 2019-07-24 | 118110.3900 | 112993.6435 | 5116.7465 |
| 2019-07-25 | 115316.2280 | 118312.3790 | -2996.1510 |
| 2019-07-26 | 162643.2550 | 121338.5281 | 41304.7269 |
| 2019-07-27 | 177738.2500 | 142599.5580 | 35138.6920 |
| 2019-07-28 | 204500.9860 | 153037.1672 | 51463.8188 |
| 2019-07-29 | 107830.3680 | 106937.7001 | 892.6679 |
| 2019-07-30 | 126576.6760 | 117050.2915 | 9526.3845 |
| 2019-07-31 | 131196.7250 | 110518.6639 | 20678.0611 |

In [66]:

```
```

In [67]:

```

```

Out[67]:

```
17339.473179332323
```

In [68]:

```

```

Out[68]:

```
12564.839062976258
```

In [69]:

```

```

In [70]:

```

```

Out[70]:

```
22765.63356590128
```

```
################################################################################
```

In [ ]:

```

```

In [ ]:

```

```

In [ ]:

```

```

# LSTM Model

In [10]:

```

```

```
Using TensorFlow backend.
```

In [ ]:

```

```

In [69]:

Out[69]:

| | Createddate | Value | year | quarter | month | day | weekday |
|---|---|---|---|---|---|---|---|
| 0 | 2018-07-01 | 123601.2160 | 2018 | 3 | 7 | 1 | 0 |
| 1 | 2018-07-02 | 64608.3200 | 2018 | 3 | 7 | 2 | 1 |
| 2 | 2018-07-03 | 70248.4300 | 2018 | 3 | 7 | 3 | 1 |
| 3 | 2018-07-04 | 68084.0100 | 2018 | 3 | 7 | 4 | 1 |
| 4 | 2018-07-05 | 80714.2200 | 2018 | 3 | 7 | 5 | 1 |

In [30]:

In [31]:

In [32]:

In [33]:

In [34]:

In [35]:

In [36]:

In [37]:

In [38]:

```
Epoch 1/20
11s - loss: 0.0100
Epoch 2/20
10s - loss: 0.0084
Epoch 3/20
11s - loss: 0.0084
Epoch 4/20
10s - loss: 0.0073
Epoch 5/20
10s - loss: 0.0076
Epoch 6/20
10s - loss: 0.0066
Epoch 7/20
10s - loss: 0.0062
Epoch 8/20
11s - loss: 0.0069
Epoch 9/20
10s - loss: 0.0061
Epoch 10/20
```

In [39]:

In [40]:

In [210]:

In [211]:

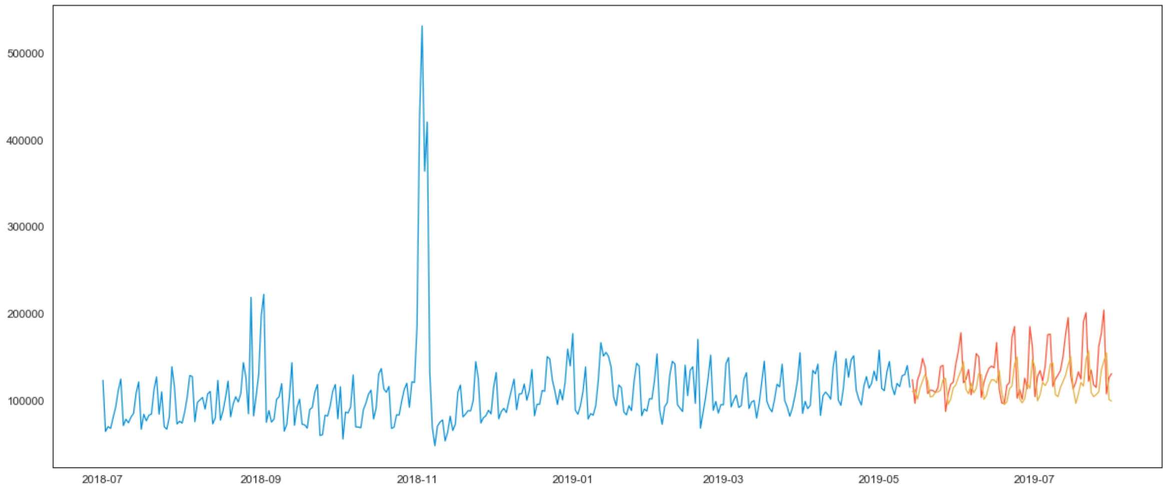
Out[211]:

```
15210.18796202546
```

In [214]:

Out[214]:

```
[<matplotlib.lines.Line2D at 0x14109b080>,
 <matplotlib.lines.Line2D at 0x14109b208>]
```




##############################################################################

In [4]:

Out[4]:

| | Unnamed: 0 | BillDate | Value | day_week |
|---|---|---|---|---|
| 0 | 0 | 2018-07-01 | 124060.591 | 6 |
| 1 | 1 | 2018-07-02 | 64358.945 | 0 |
| 2 | 2 | 2018-07-03 | 70038.430 | 1 |
| 3 | 3 | 2018-07-04 | 68084.010 | 2 |
| 4 | 4 | 2018-07-05 | 80714.220 | 3 |

In [5]:

In [6]:

In [7]:

In [8]:

In [11]:

In [12]:

In [13]:

In [14]:

In [15]:

```
Epoch 1/100
296/296 [==============================] - 11s - los
s: 0.0102
Epoch 2/100
296/296 [==============================] - 11s - los
s: 0.0092
Epoch 3/100
296/296 [==============================] - 12s - los
s: 0.0082
Epoch 4/100
296/296 [==============================] - 14s - los
s: 0.0070
Epoch 5/100
296/296 [==============================] - 11s - los
s: 0.0071
Epoch 6/100
296/296 [==============================] - 10s - los
s: 0.0064
Epoch 7/100
```

In [16]:

In [17]:

In [18]:

In [19]:

Out[19]:

```
19698.17791718764
```

In [23]:

In [24]:

```
/anaconda3/lib/python3.6/site-packages/ipykernel_lau
ncher.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice fro
m a DataFrame.
Try using .loc[row_indexer,col_indexer] = value inst
ead

See the caveats in the documentation: http://pandas.
pydata.org/pandas-docs/stable/user_guide/indexing.ht
ml#returning-a-view-versus-a-copy
(http://pandas.pydata.org/pandas-docs/stable/user_gu
ide/indexing.html#returning-a-view-versus-a-copy)
  This is separate from the ipykernel package so we
can avoid doing imports until
```

Out[24]:

```
[<matplotlib.lines.Line2D at 0x117567dd8>,
 <matplotlib.lines.Line2D at 0x117567f28>]
```

In [ ]:

# Stock Market Data

In [180]:

In [181]:

```

```

Out[181]:

| | Date | Open | High | Low | Last | Close | Total Trac Quanti |
|---|---|---|---|---|---|---|---|
| 0 | 2013-10-08 | 157.0000 | 157.8000 | 155.2000 | 155.8000 | 155.8000 | 1720413.00( |
| 1 | 2013-10-09 | 155.7000 | 158.2000 | 154.1500 | 155.3000 | 155.5500 | 2049580.00( |
| 2 | 2013-10-10 | 156.0000 | 160.8000 | 155.8500 | 160.3000 | 160.1500 | 3124853.00( |
| 3 | 2013-10-11 | 161.1500 | 163.4500 | 159.0000 | 159.8000 | 160.0500 | 1880046.00( |
| 4 | 2013-10-14 | 160.8500 | 161.4500 | 157.7000 | 159.3000 | 159.4500 | 1281419.00( |

In [260]:

```

```

Out[260]:

| | Open | High | Low | Last | Close | Total Qu |
|---|---|---|---|---|---|---|
| count | 1235.0000 | 1235.0000 | 1235.0000 | 1235.0000 | 1235.0000 | 1235 |
| mean | 168.9549 | 171.4291 | 166.4023 | 168.7364 | 168.7311 | 2604151 |
| std | 51.4991 | 52.4368 | 50.5429 | 51.5874 | 51.5449 | 2277027 |
| min | 103.0000 | 104.6000 | 100.0000 | 102.6000 | 102.6500 | 100180 |
| 25% | 137.5500 | 138.9250 | 135.2500 | 137.1750 | 137.2250 | 1284481 |
| 50% | 151.5000 | 153.2500 | 149.5000 | 151.2000 | 151.1000 | 1964885 |
| 75% | 169.0000 | 172.3250 | 166.7000 | 169.1000 | 169.5000 | 3095788 |
| max | 327.7000 | 328.7500 | 321.6500 | 325.9500 | 325.7500 | 29191015 |

In [182]:

```

```

In [183]:

```

```

In [184]:

In [185]:

In [186]:

In [187]:

In [188]:

In [189]:

In [190]:

```
Epoch 1/20
96s - loss: 0.0011
Epoch 2/20
94s - loss: 5.3325e-04
Epoch 3/20
94s - loss: 3.1337e-04
Epoch 4/20
94s - loss: 2.8767e-04
Epoch 5/20
94s - loss: 2.7148e-04
Epoch 6/20
94s - loss: 2.4441e-04
Epoch 7/20
95s - loss: 2.3399e-04
Epoch 8/20
94s - loss: 2.4268e-04
Epoch 9/20
95s - loss: 2.3833e-04
Epoch 10/20
95s - loss: 2.1853e-04
```

In [191]:

In [192]:

In [193]:

```
```

In [194]:

```
```

Out[194]:

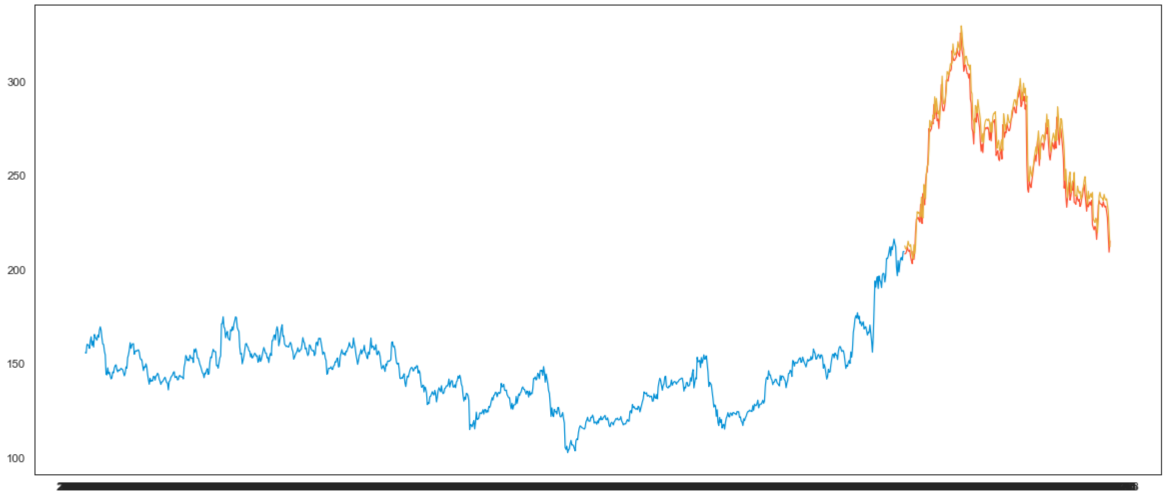
```
7.211366722400344
```

In [195]:

```
```

Out[195]:

```
[<matplotlib.lines.Line2D at 0x140f4d668>,
 <matplotlib.lines.Line2D at 0x140f4d828>]
```




In [ ]:

```
```