tsf-mpa-1

April 6, 2023

Case study: US daily Covid 19 data

[ ]:
```

```

1 Time Series Forecasting Mini project

1.1 Project Member

1.1.1 Bharath K M

1.1.2 Marimuthu S

1.1.3 Santhosh Kumar G

[ ]:
```

```

[ ]:
```

```

1.2 Importing required libraries

[142]:
```
import pandas                                   as      pd
import numpy                                    as      np
import matplotlib.pyplot                        as      plt
import seaborn                                  as      sns
from   IPython.display                          import  display
from   pylab                                    import  rcParams
from   sklearn.metrics                          import  mean_squared_error
from   datetime                                 import  datetime, timedelta
from statsmodels.tsa.stattools                  import  adfuller
from statsmodels.tsa.stattools                  import  pacf
from statsmodels.tsa.stattools                  import  acf
from statsmodels.graphics.tsaplots              import  plot_pacf
from statsmodels.graphics.tsaplots              import  plot_acf
from statsmodels.graphics.gofplots              import  qqplot
from statsmodels.tsa.seasonal                   import  seasonal_decompose, STL
from statsmodels.tsa.arima_model                import  ARIMA
from statsmodels.tsa.statespace.sarimax         import  SARIMAX
from statsmodels.tsa.api                        import  ExponentialSmoothing,
 ↪SimpleExpSmoothing, Holt
```

```
import warnings
warnings.filterwarnings('ignore')
%matplotlib inline
```

1.3 Import the data

1.3.1 Data is collected for the period of 17th March 2020 to 06th December 2020. Data is collected on daily basis for Positive cases, Hospitalized and Death count

[51]:
```
df_covid = pd.read_csv('us_covid19_daily.csv')
```

[52]:
```
df_covid
```

[52]:
```
           Date  Positive  Hospitalized   Death
0    17-03-2020     10021           325     124
1    18-03-2020     13385           416     155
2    19-03-2020     18085           617     203
3    20-03-2020     24197          1042     273
4    21-03-2020     31013          1492     335
..          …         …             …       …
260  02-12-2020  13711156        100322  264522
261  03-12-2020  13921360        100755  267228
262  04-12-2020  14146191        101276  269791
263  05-12-2020  14357264        101190  272236
264  06-12-2020  14534035        101487  273374

[265 rows x 4 columns]
```

1.3.2 Convert the data into time series data

[53]:
```
df_covid['Date'] = pd.to_datetime(df_covid['Date'],infer_datetime_format=True)␣
 ↪#convert from string to datetime
df_covid = df_covid.set_index(['Date'])
df_covid
```

[53]:
```
            Positive  Hospitalized   Death
Date
2020-03-17     10021           325     124
2020-03-18     13385           416     155
2020-03-19     18085           617     203
2020-03-20     24197          1042     273
2020-03-21     31013          1492     335
…                …             …       …
2020-12-02  13711156        100322  264522
2020-12-03  13921360        100755  267228
2020-12-04  14146191        101276  269791
2020-12-05  14357264        101190  272236
```

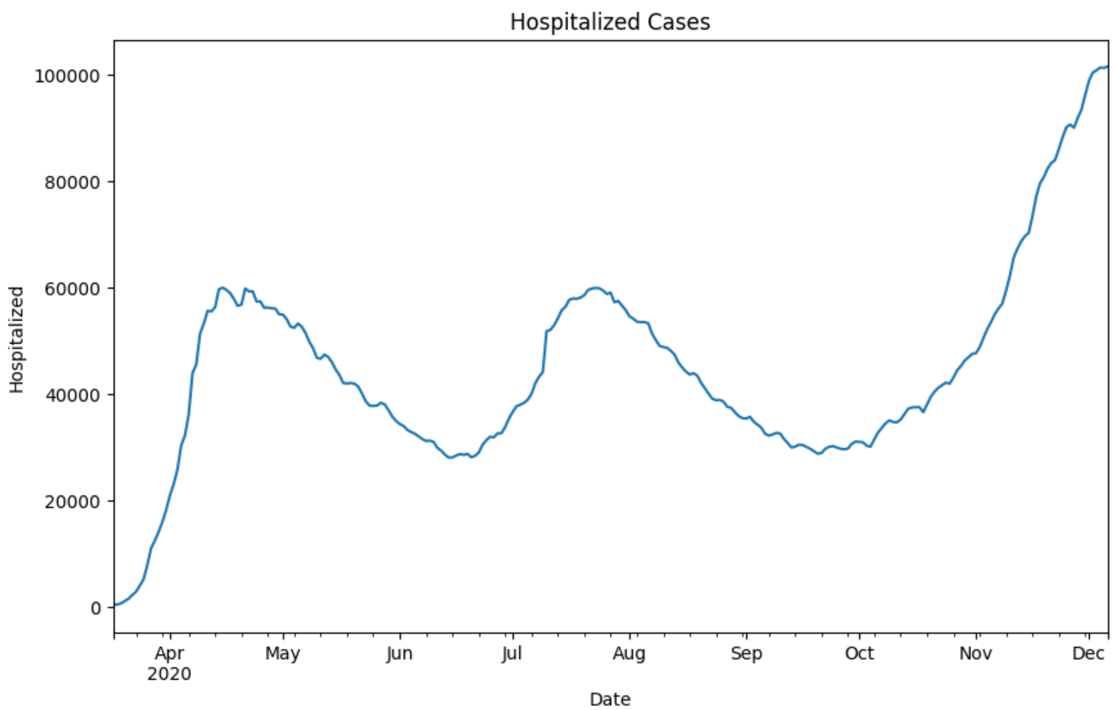
```
2020-12-06  14534035        101487  273374

[265 rows x 3 columns]
```

1.3.3 For this case study, we will build time series model to forecast the Hospitalized count

1.3.4 Plot the variable 'hospitalized'

[59]:
```
df_covid['Hospitalized'].plot(figsize=(10,6),title='Hospitalized Cases')
plt.ylabel('Hospitalized')
```

[59]:
```
Text(0, 0.5, 'Hospitalized')
```



From the above plot I would say there are outliers present in the Hospitalized column but which should not be removed from the dataset

1.4 Perform Exploratory Data Analysis

[ ]:
```
```

[55]:
```
df_covid.isnull().sum()
```

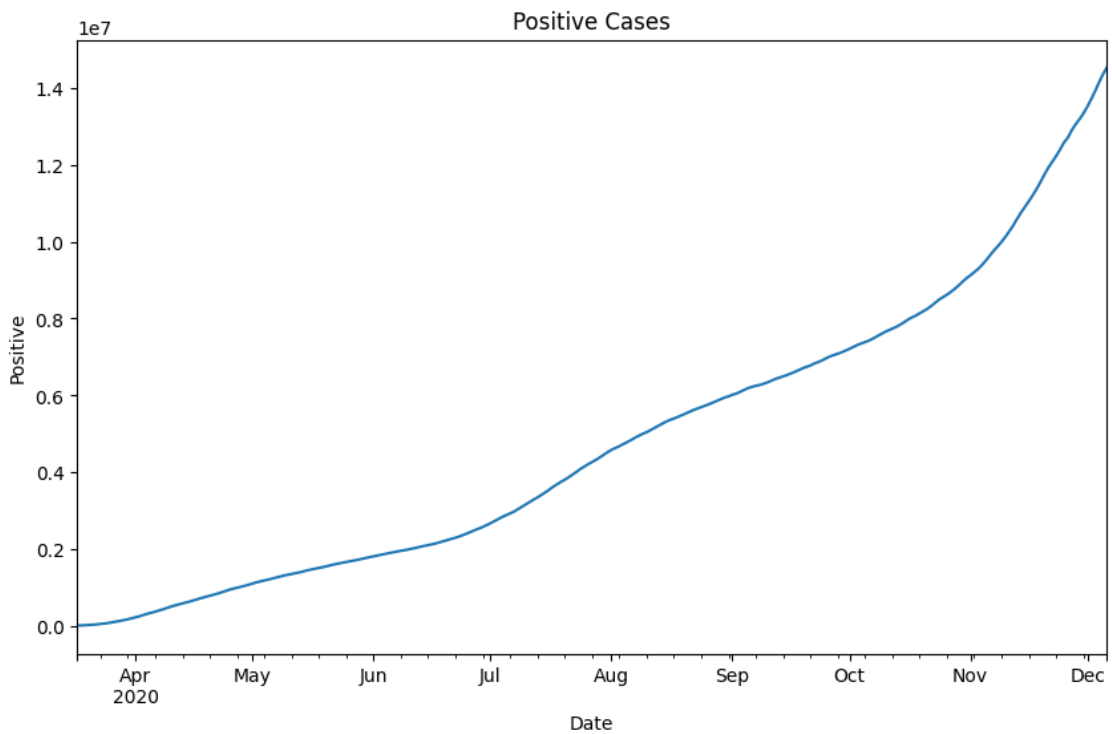
[55]: Positive   0
Hospitalized   0
Death   0
dtype: int64

1.4.1 Plot monthwise distribution

As per above output it was clear that we don't have any null values in our dataset

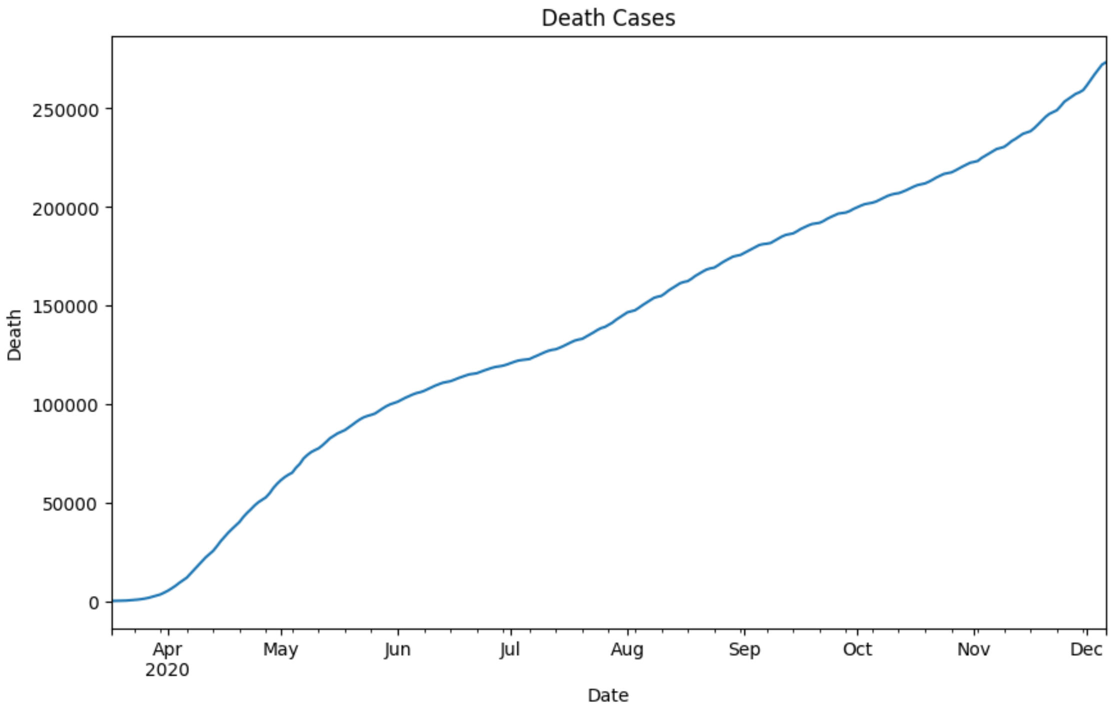
[60]:
```
### Plotting positive cases

df_covid['Positive'].plot(figsize=(10,6),title='Positive Cases')
plt.ylabel('Positive')
```

[60]: Text(0, 0.5, 'Positive')



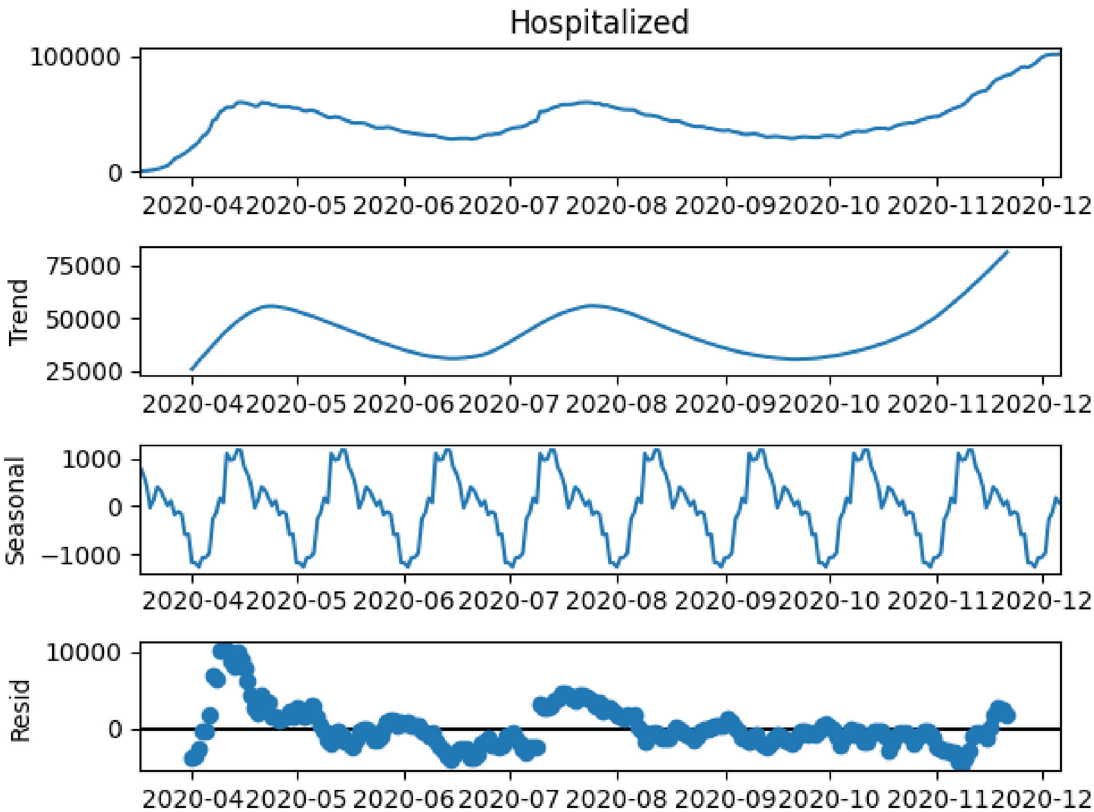
[61]:
```
### Plotting Death cases
df_covid['Death'].plot(figsize=(10,6),title='Death Cases')
plt.ylabel('Death')
```

[61]: Text(0, 0.5, 'Death')

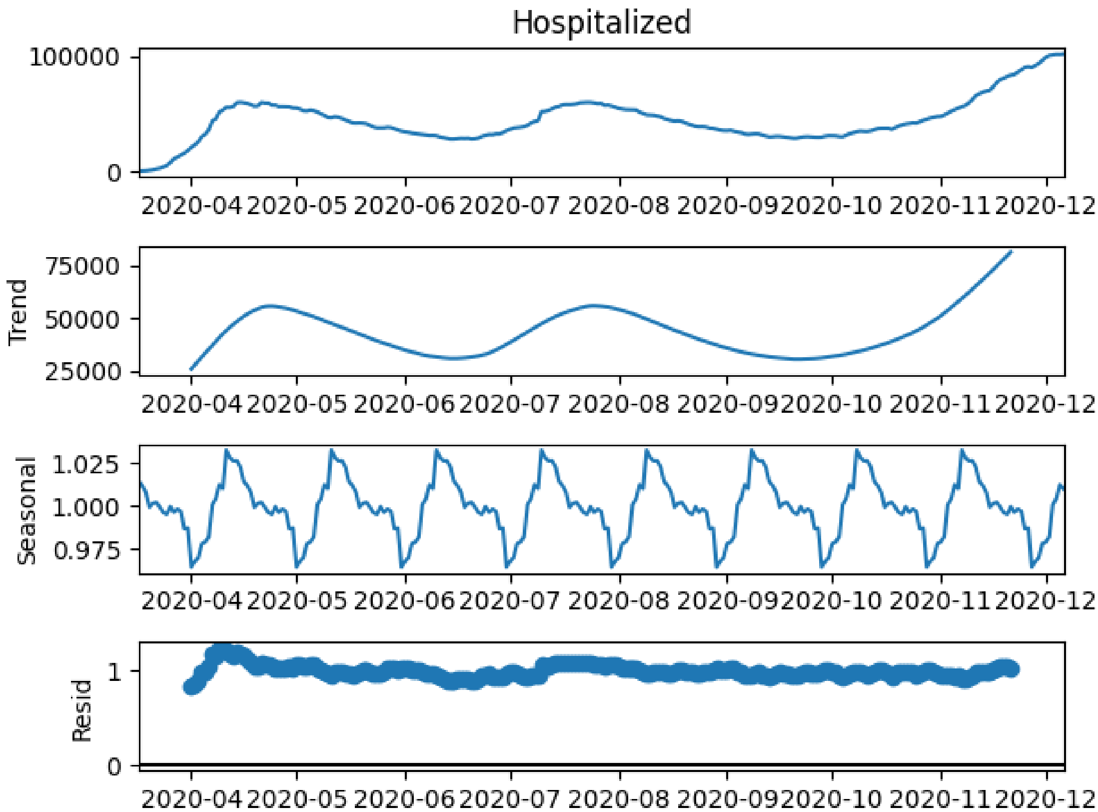


1.5 Decompose the time series

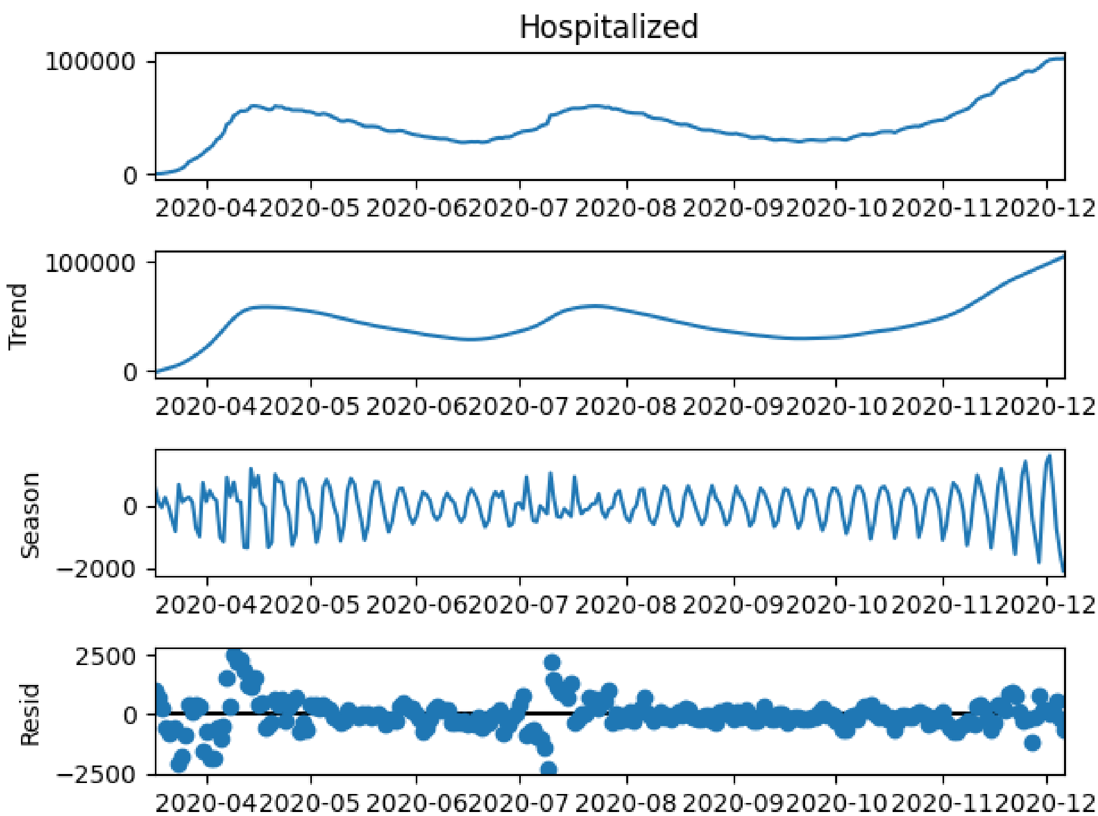
[66]:
```
decomposition = seasonal_decompose(df_covid["Hospitalized"],model='additive',
    period = 30)
decomposition.plot();
```



[65]:
```
decomposition =␣
 ↪seasonal_decompose(df_covid["Hospitalized"],model='multiplicative', period =␣
 ↪30)
decomposition.plot();
```

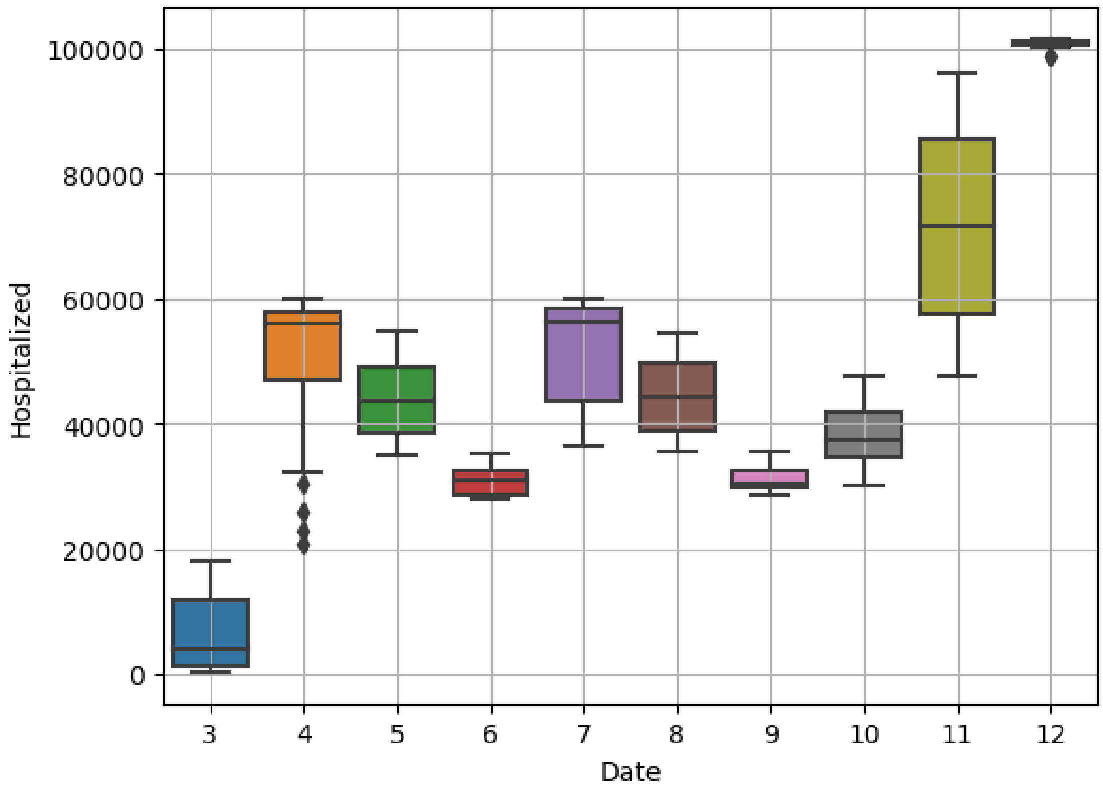


[70]:
```
decomposition = STL(df_covid["Hospitalized"]).fit()
decomposition.plot();
```



1.6 Check for stationarity in the data

[76]:
```
sns.boxplot(x=df_covid.index.month,y=df_covid['Hospitalized'])
plt.grid();
```




```
[79]: observations= df_covid["Hospitalized"].values
      test_result = adfuller(observations)
```

```
[80]: test_result
```

```
[80]: (-1.6387257227470495,
       0.46292671711887773,
       16,
       248,
       {'1%': -3.4569962781990573,
        '5%': -2.8732659015936024,
        '10%': -2.573018897632674},
       4101.584279516279)
```

```
[95]: print('ADF Statistic: %f' % test_result[0])
      print('p-value: %f' % test_result[1])
      print('Critical Values:')
      for key, value in test_result[4].items():
              print('\t%s: %.5f' % (key, value))
```

```
ADF Statistic: -3.155128
p-value: 0.022733
```

```
Critical Values:
	1%: -3.45711
	5%: -2.87331
	10%: -2.57304
```

1.7 applying differencing

[82]:
```
df_diff = df_covid["Hospitalized"].diff(periods=1).dropna()
observations= df_diff.values
test_result = adfuller(observations)
test_result
```

[82]:
```
(-3.1551279027958765,
 0.022733461203747,
 16,
 247,
 {'1%': -3.457105309726321,
  '5%': -2.873313676101283,
  '10%': -2.5730443824681606},
 4086.8057125045957)
```

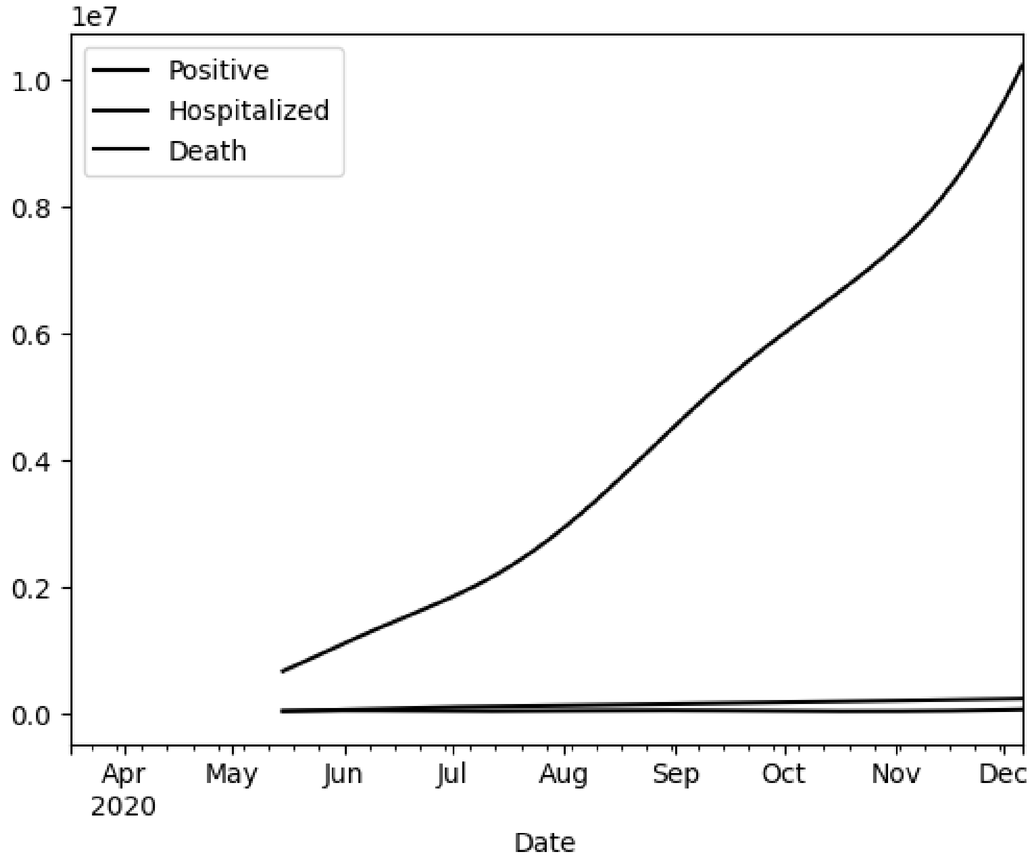
[87]:
```
print('ADF Statistic: %f' % test_result[0])
print('p-value: %f' % test_result[1])
print('Critical Values:')
for key, value in test_result[4].items():
        print('\t%s: %.5f' % (key, value))
```

```
ADF Statistic: -3.155128
p-value: 0.022733
Critical Values:
	1%: -3.45711
	5%: -2.87331
	10%: -2.57304
```

1.8 Let's plot rolling mean and std deviation

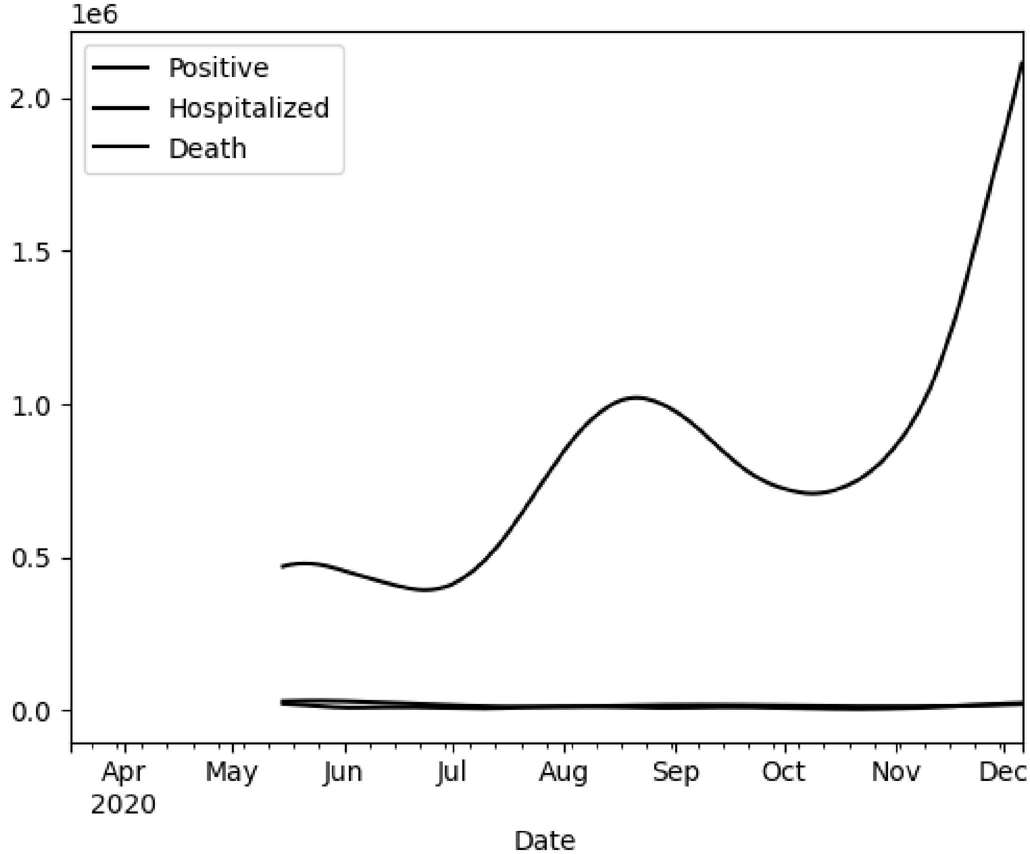
[90]:
```
# calculate a 60 day rolling mean and plot
df_covid.rolling(window=60).mean().plot(style='k')
```

[90]:
```
<AxesSubplot: xlabel='Date'>
```




```
[94]: # calculate a 60 day rolling std and plot
      df_covid.rolling(window=60).std().plot(style='k')
```

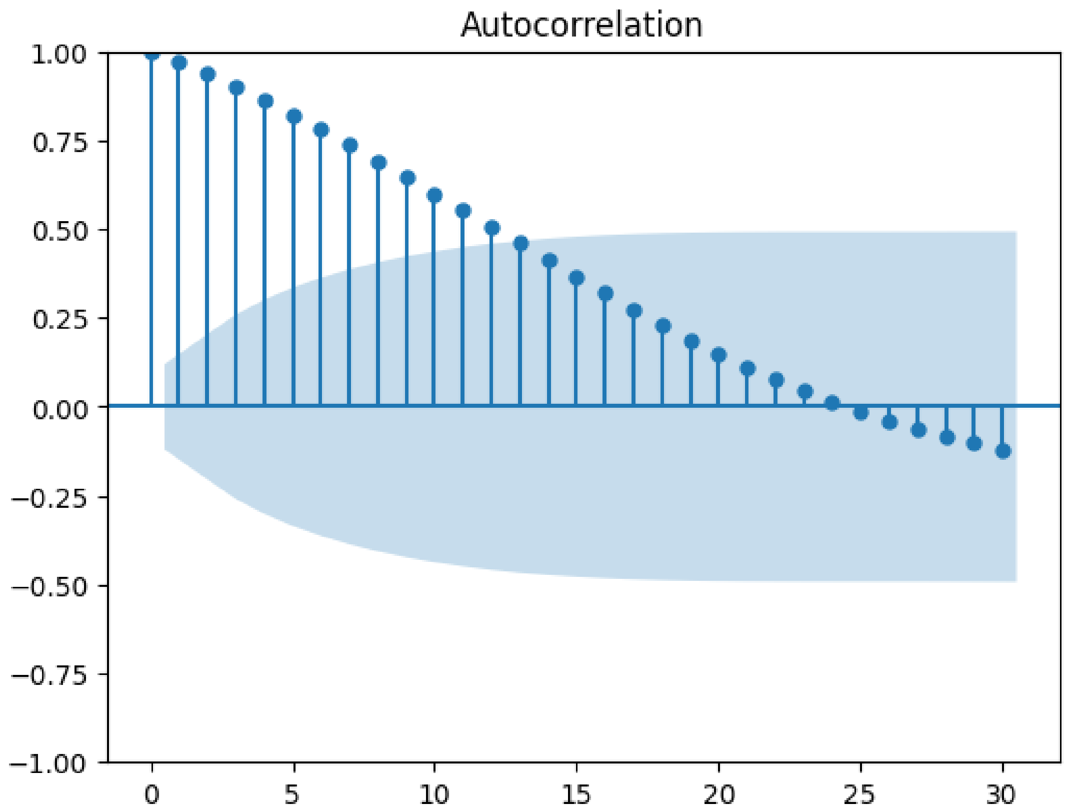
```
[94]: <AxesSubplot: xlabel='Date'>
```



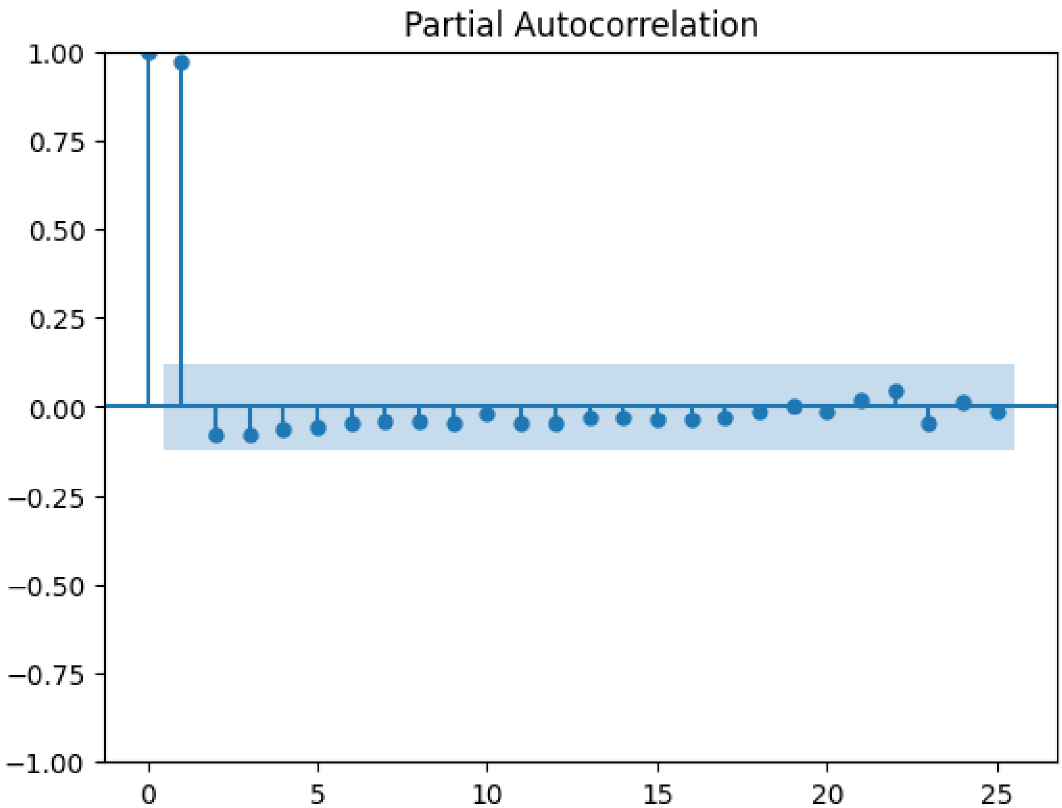
1.9 Perform statistical test to confirm the stationarity

1.10 Plot the ACF and PACF plots for the series

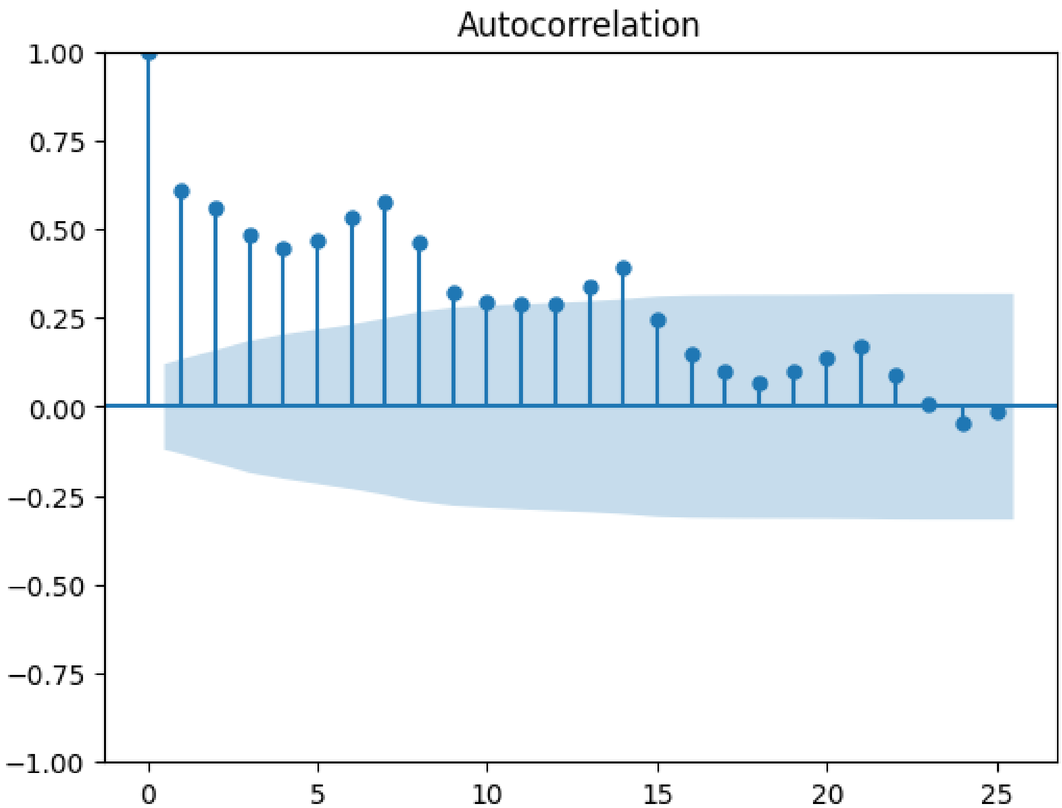
[83]:
```
plot_acf(df_covid["Hospitalized"],lags=30);
```



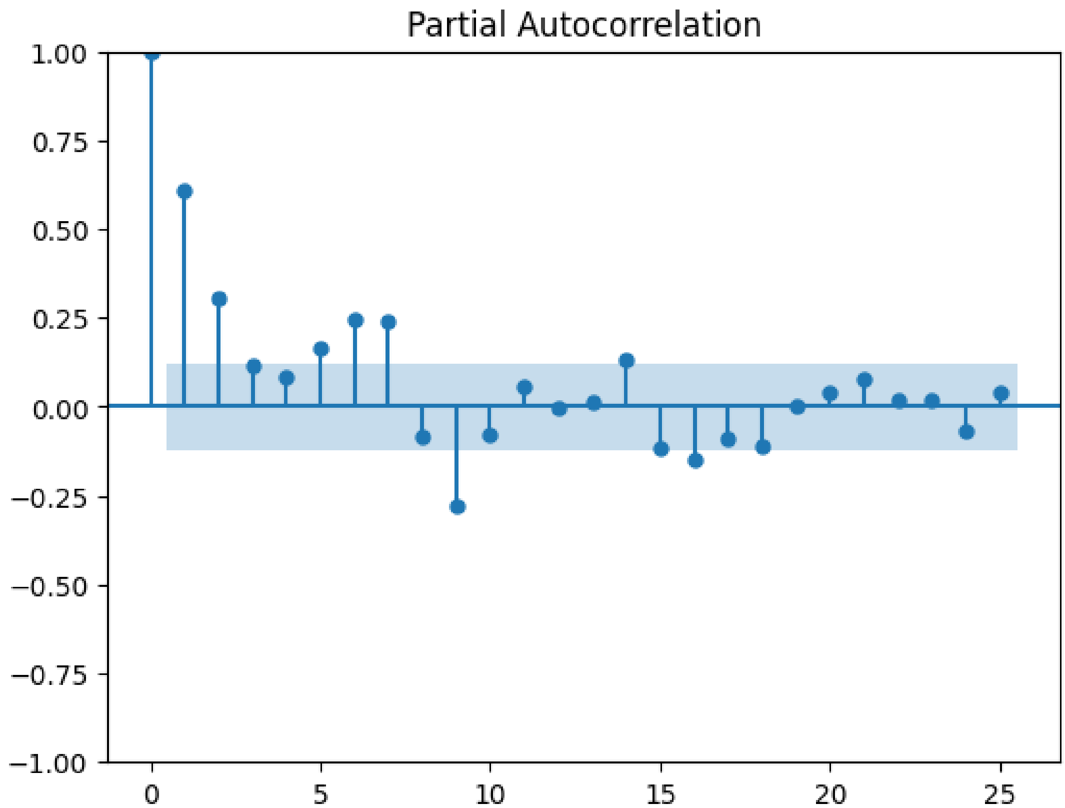
[84]:
```
plot_pacf(df_covid["Hospitalized"]);
```



[85]:
```
plot_acf(df_diff);
```



[86]:
```
plot_pacf(df_diff);
```



1.10.1 ACF plot is clearly showing, time series observations are heavily impacted by past values. While PACF is showing limited number of spikes before cut-off

1.11 Split the series into training and testing sets

[104]:
```
train = df_covid["Hospitalized"].loc[:'30-10-2020']
test = df_covid["Hospitalized"].loc['30-10-2020':]
```

[105]:
```
train_data = pd.DataFrame(train)
train_data.head(5)
```

[105]:

	Hospitalized
Date	
2020-03-17	325
2020-03-18	416
2020-03-19	617
2020-03-20	1042
2020-03-21	1492

[106]:
```
test_data = pd.DataFrame(test)
test_data.head(5)
```

[106]:
```
            Hospitalized
Date
2020-10-30         46856
2020-10-31         47486
2020-11-01         47615
2020-11-02         48773
2020-11-03         50512
```

1.11.1 We will build the Holt forecasting model and Holt-Winter forecasting model.

2 Double Exponential Smoothing / Holt's linear Method

[146]:
```
model_DES = Holt(train_data,exponential=True, initialization_method='estimated')
```

training the double exponential model

[147]:
```
model_DES_fit1 = model_DES.fit(optimized=True)
```

[148]:
```
model_DES_fit1.summary()
```

[148]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                                             Holt Model Results
================================================================================
============================
Dep. Variable:          Hospitalized   No. Observations:
228
Model:                          Holt   SSE
5647258012956255812842806650819730641571O208.000
Optimized:                      True   AIC
21739.360
Trend:                Multiplicative   BIC
21753.078
Seasonal:                       None   AICC
21739.740
Seasonal Periods:               None   Date:
Thu, 06 Apr 2023
Box-Cox:                       False   Time:
17:42:50
Box-Cox Coeff.:                 None
==============================================================================
                       coeff                 code              optimized
------------------------------------------------------------------------------
smoothing_level            0.9478571                alpha                True
smoothing_trend            0.4623693                 beta                True
initial_level              0.0100000                  l.0                True
initial_trend              0.5107490                  b.0                True
```

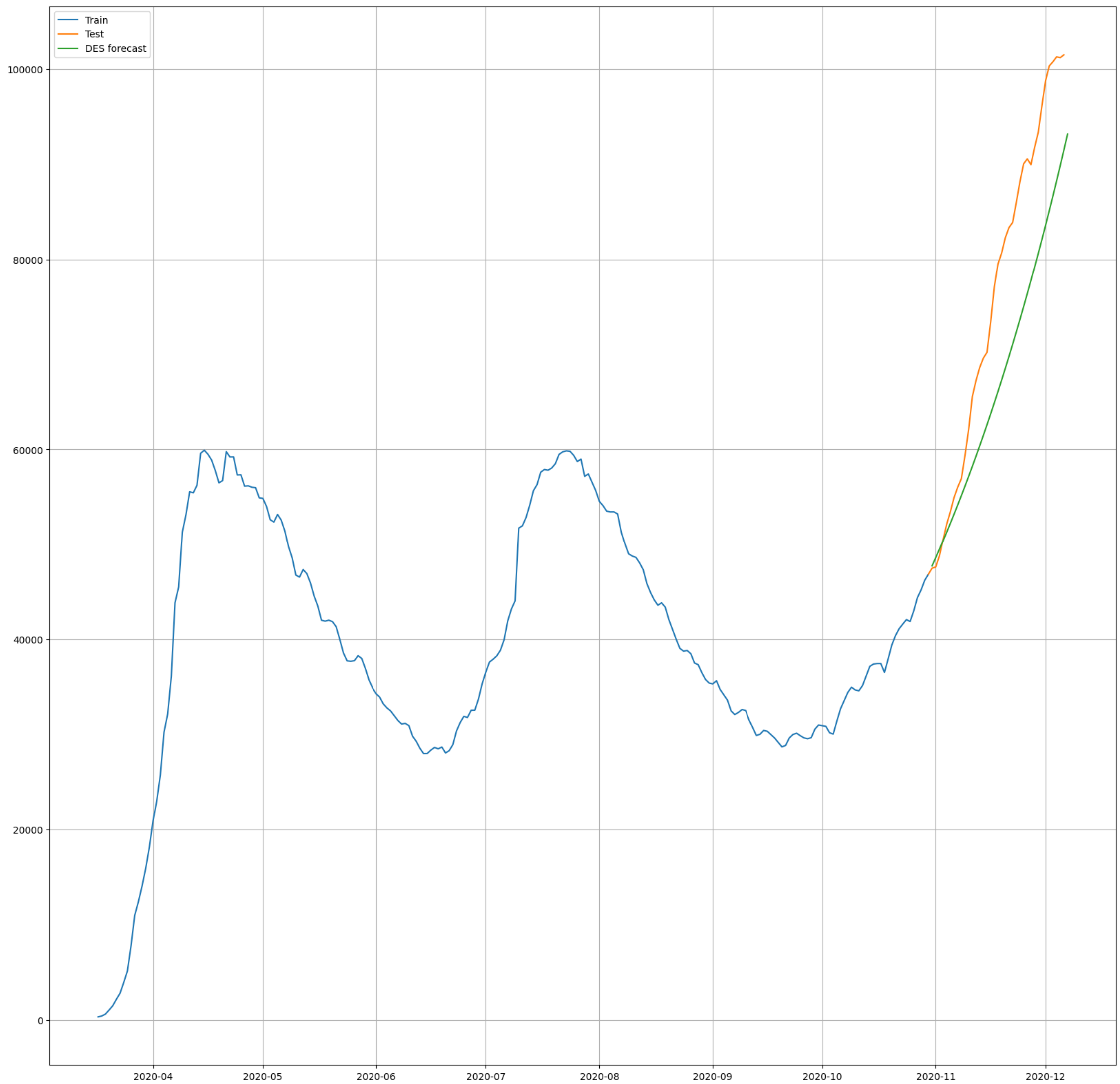
```
--------------------------------------------------------------------------------
"""
```

Predicting forecast

[174]:
```
DES_predict1 = model_DES_fit1.forecast(steps=len(test))
```

[174]:
```
array([47730.05240245, 48600.70510847, 49487.2395514 , 50389.94543292,
       51309.11773924, 52245.05683745, 53198.06857371, 54168.46437316,
       55156.56134171, 56162.68236966, 57187.1562372 , 58230.31772187,
       59292.50770795, 60374.07329786, 61475.36792555, 62596.75147206,
       63738.59038306, 64901.25778865, 66085.13362523, 67290.60475971,
       68518.06511593, 69767.91580332, 71040.56524807, 72336.42932651,
       73655.93150108, 74999.50295864, 76367.58275145, 77760.61794056,
       79179.06374199, 80623.38367539, 82094.0497156 , 83591.54244682,
       85116.35121967, 86668.97431112, 88249.91908729, 89859.70216928,
       91498.84960193, 93167.89702581])
```

[150]:
```
fig = plt.figure(figsize=(20, 20))
plt.plot(train_data, label='Train')
plt.plot(test_data, label='Test')

plt.plot(DES_predict1, label='DES forecast')
plt.legend(loc='best')
plt.grid()
plt.show()
```



3 Triple Exponential Smoothing / Holt-Winters Method

lets build model using 'additive' seasonality

[151]:
```
model_TES_add =␣
↪ExponentialSmoothing(train_data,trend='additive',seasonal='additive',initialization_method=
```

training the model

[152]:
```
model_TES_add = model_TES_add.fit(optimized=True)
```

[153]:
```
model_TES_add.summary()
```

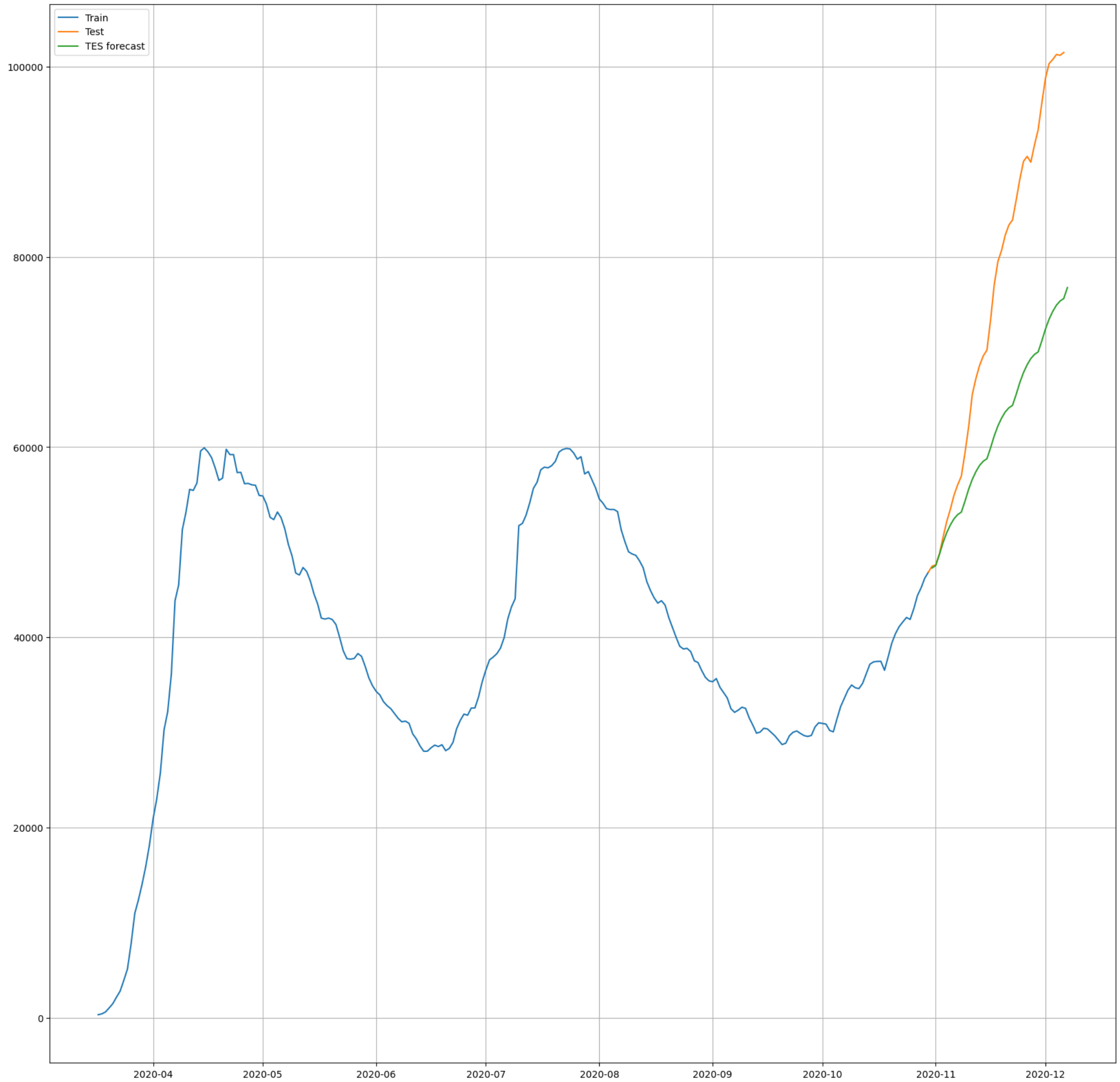
[153]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                       ExponentialSmoothing Model Results
================================================================================
Dep. Variable:              Hospitalized   No. Observations:                228
Model:              ExponentialSmoothing   SSE                    189761542.315
Optimized:                          True   AIC                         3130.081
Trend:                          Additive   BIC                         3167.804
Seasonal:                       Additive   AICC                        3131.782
Seasonal Periods:                      7   Date:               Thu, 06 Apr 2023
Box-Cox:                           False   Time:                       17:43:22
Box-Cox Coeff.:                     None
================================================================================
=
                           coeff                     code              optimized
--------------------------------------------------------------------------------
-
smoothing_level                0.8340960                alpha
True
smoothing_trend                0.3761838                 beta
True
smoothing_seasonal             0.0922137                gamma
True
initial_level                 -1580.7489                  l.0
True
initial_trend                  1383.1675                  b.0
True
initial_seasons.0              651.40550                  s.0
True
initial_seasons.1              369.12270                  s.1
True
initial_seasons.2              483.97959                  s.2
True
initial_seasons.3              103.29371                  s.3
True
initial_seasons.4              21.696583                  s.4
True
initial_seasons.5             -761.99596                  s.5
True
initial_seasons.6             -846.78481                  s.6
True
--------------------------------------------------------------------------------
-
"""
```

predicting forecast

[154]:
```
TES_add_predict =  model_TES_add.forecast(len(test_data))
```

lets plot foecast results

[155]:
```
fig = plt.figure(figsize=(20, 20))
plt.plot(train_data, label='Train')
plt.plot(test_data, label='Test')

plt.plot(TES_add_predict, label='TES forecast')
plt.legend(loc='best')
plt.grid()
```



3.1 lets build model uaing 'multiplicative' forecast

[158]:
```
model_TES_mul =␣
↪ExponentialSmoothing(train_data,trend='multiplicative',seasonal='multiplicative',initializa
```

[159]:
```
model_TES_mul = model_TES_mul.fit(optimized=True)
```

[160]:
```
model_TES_mul.summary()
```

[160]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                                                   ExponentialSmoothing Model
Results
================================================================================
=======================================================
Dep. Variable:              Hospitalized   No. Observations:
228
Model:              ExponentialSmoothing   SSE
1098696371805879230607281503342817423666386813913858548474553827328000.000
Optimized:                          True   AIC
34504.849
Trend:                    Multiplicative   BIC
34542.572
Seasonal:                 Multiplicative   AICC
34506.550
Seasonal Periods:                      7   Date:
Thu, 06 Apr 2023
Box-Cox:                           False   Time:
17:46:04
Box-Cox Coeff.:                     None
================================================================================
=
                          coeff                 code              optimized
--------------------------------------------------------------------------------
-
smoothing_level               0.8182143                alpha
True
smoothing_trend               0.4405769                 beta
True
smoothing_seasonal               0.0001                gamma
True
initial_level                 0.0100000                  l.0
True
initial_trend                 0.0807022                  b.0
True
initial_seasons.0             0.9992640                  s.0
True
```

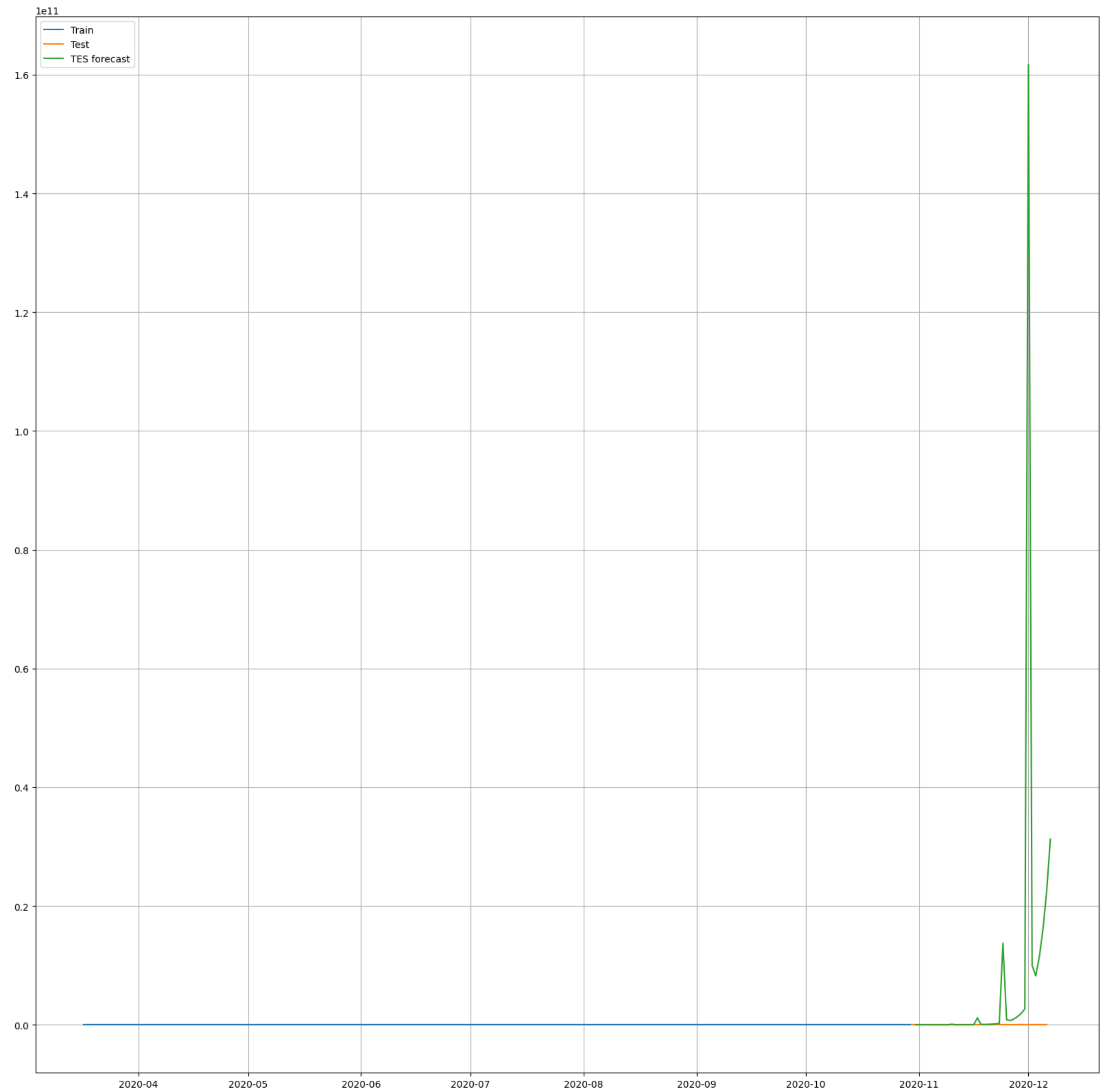
```
initial_seasons.1              0.9883090                 s.1
True
initial_seasons.2              1.0354751                 s.2
True
initial_seasons.3              1.0199128                 s.3
True
initial_seasons.4              1.0089510                 s.4
True
initial_seasons.5              0.9865219                 s.5
True
initial_seasons.6              0.9615662                 s.6
True
--------------------------------------------------------------------------------
-
"""
```

[161]:

```
#### predicting forecast
TES_mul_predict =  model_TES_mul.forecast(len(test))
```

[162]:

```
fig = plt.figure(figsize=(20, 20))
plt.plot(train, label='Train')
plt.plot(test, label='Test')
plt.plot(TES_mul_predict, label='TES forecast')
plt.legend(loc='best')
plt.grid()
```



[ ]:

3.1.1 Plot the model predictions and find the RMSE and MAPE value.

3.2 Evaluating Model Performance

3.2.1 Double Exponential Smoothing

[175]:
```
mean_squared_error(test_data["Hospitalized"].values,DES_predict1.
 ↪values,squared=False)
```

[175]: 9196.47083050614

[176]:
```
def MAPE(y_true, y_pred):
    return np.mean((np.abs(y_true-y_pred))/(y_true))*100
```

[177]:
```
MAPE(test_data["Hospitalized"],DES_predict1)
```

[177]: 10.76059656980832

3.3 Triple Exponential smoothing (Additive model evaluation)

[156]:
```
mean_squared_error(test_data["Hospitalized"].values,TES_add_predict.
 ↪values,squared=False)
```

[156]: 16262.10368882353

[144]:
```
def MAPE(y_true, y_pred):
    return np.mean((np.abs(y_true-y_pred))/(y_true))*100
```

[157]:
```
MAPE(test_data["Hospitalized"],TES_add_predict)
```

[157]: 16.80449617264712

3.4 Triple Exponential Smoothing (Multiplicative model evaluation)

[163]:
```
mean_squared_error(test_data["Hospitalized"].values,TES_mul_predict.
 ↪values,squared=False)
```

[163]: 27337865015.120583

[164]:
```
MAPE(test_data["Hospitalized"],TES_mul_predict)
```

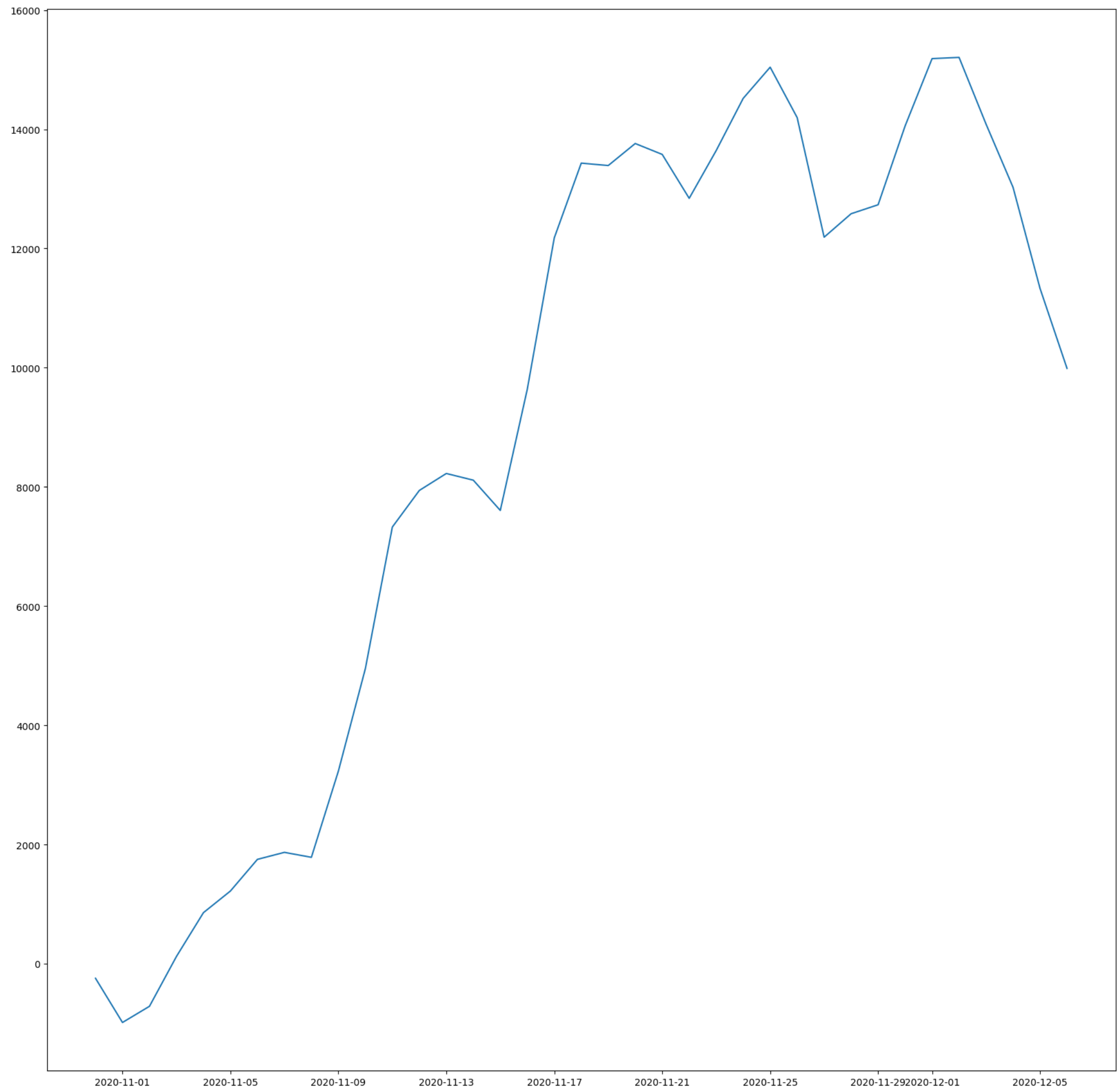
[164]: 6993646.599850103

3.4.1 Calculate and plot the residuals.

3.4.2 Calculating Double Exponential Smoothing Residuals

[182]:
```
DES_residuals = test_data.Hospitalized - DES_predict1
```

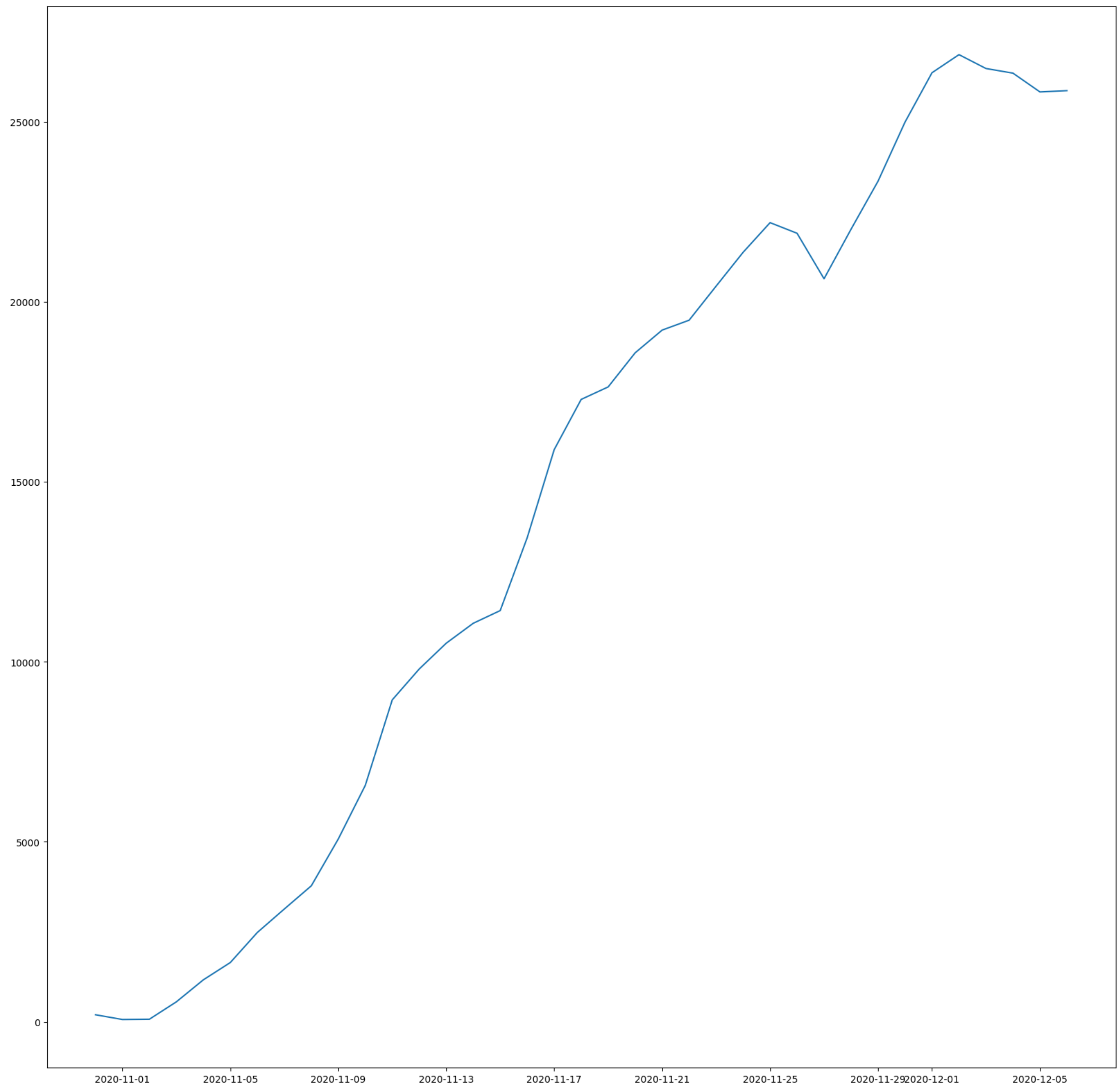
[179]:
```
fig = plt.figure(figsize=(20, 20))
plt.plot(DES_residuals)
plt.show()
```



3.4.3 Calculating Triple exponential Additive Residual

[166]:
```
TES_Additive_residuals = test_data.Hospitalized - TES_add_predict
```

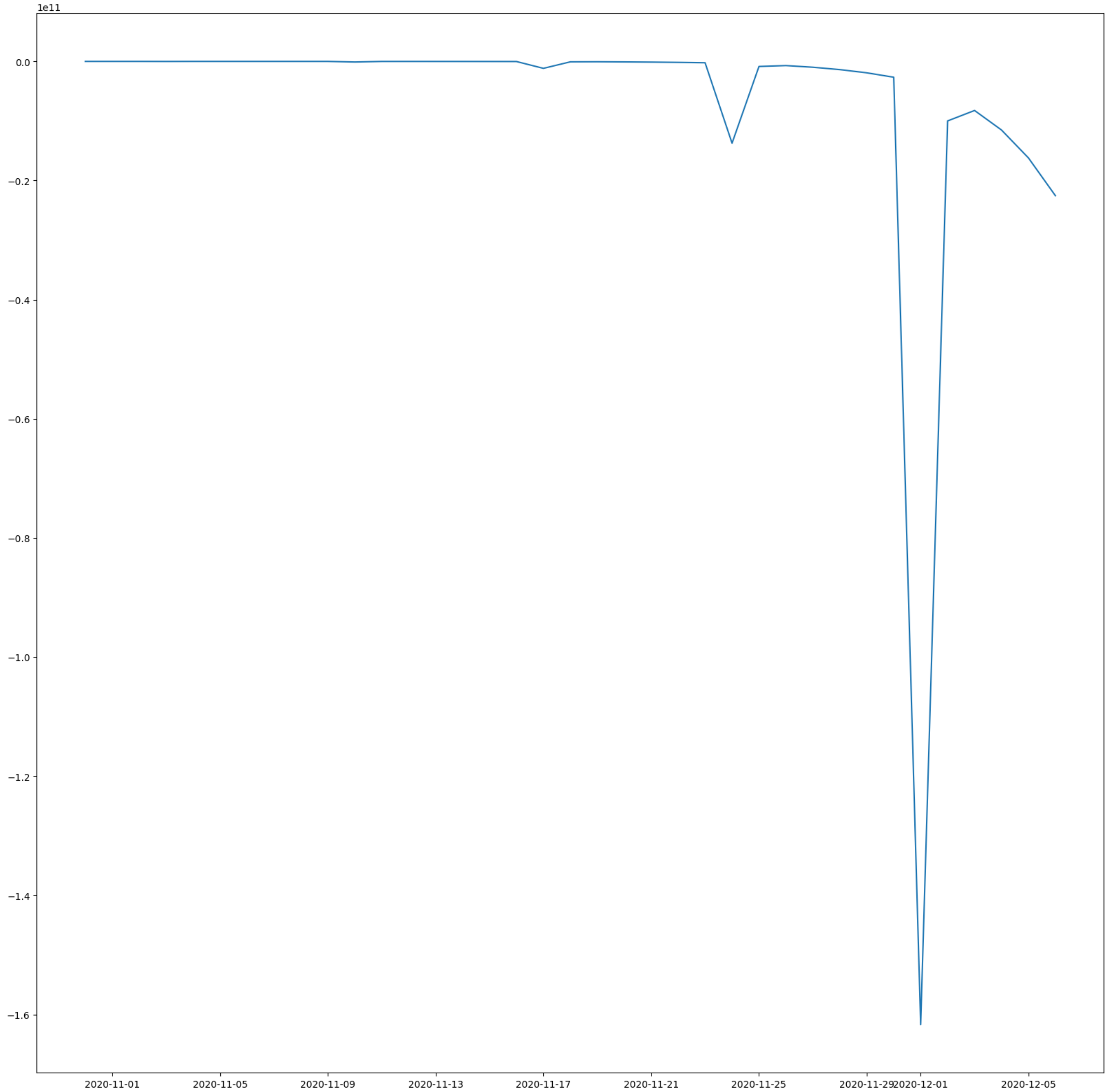
[168]:
```
fig = plt.figure(figsize=(20, 20))
plt.plot(TES_Additive_residuals)
plt.show()
```



3.4.4 Calculating Triple Exponential Multiplicative Residuals

[169]:
```
TES_Multiplicative_residuals = test_data.Hospitalized - TES_mul_predict
```

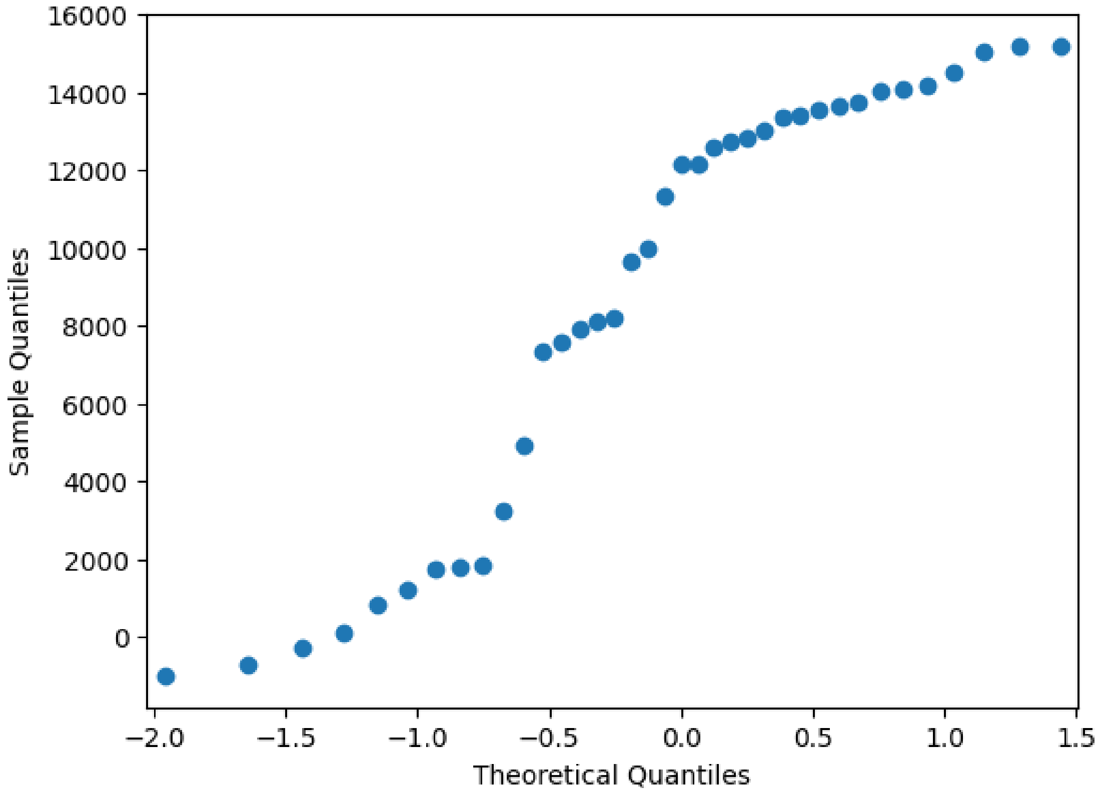
[170]:
```
fig = plt.figure(figsize=(20, 20))
plt.plot(TES_Multiplicative_residuals)
plt.show()
```



3.4.5 residual q-q plot for to check model performance

[189]:
```
# Double Exponential Smoothing

fig = plt.figure(figsize=(20, 20))
qqplot(DES_residuals,line="s");
```

```
<Figure size 2000x2000 with 0 Axes>
```



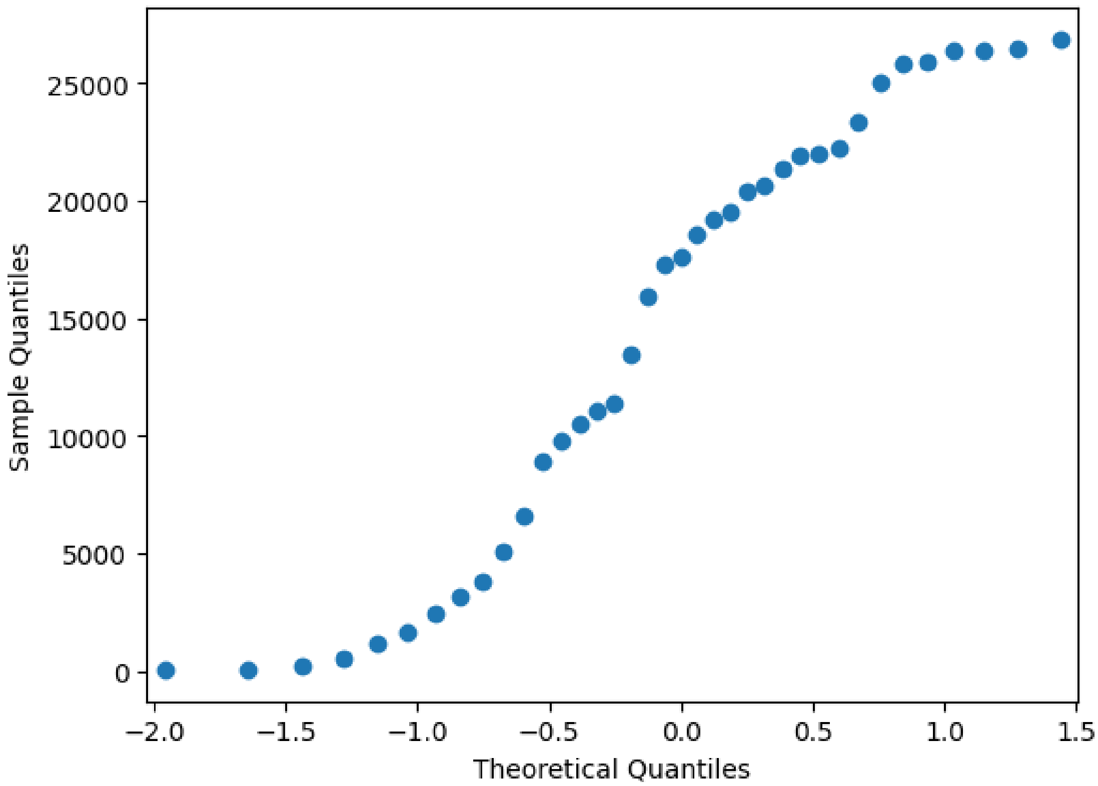
[ ]:

```

```

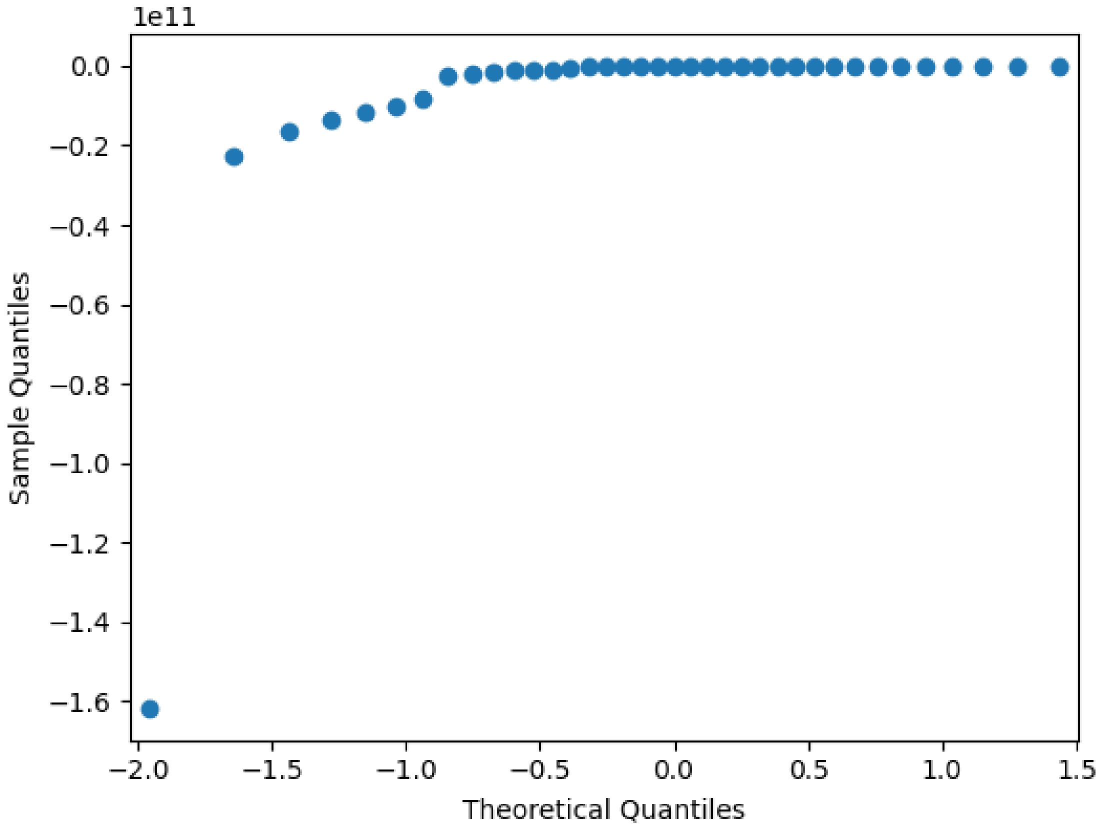
[187]:

```
# Triple Exponential Smoothing [Additive Model]

fig = plt.figure(figsize=(20, 20))
qqplot(TES_Additive_residuals,line="s");
```

```
<Figure size 2000x2000 with 0 Axes>
```



[172]:
```
# Triple Exponential Smoothing [Multiplicative Model]

qqplot(TES_Multiplicative_residuals,line="s");
```



3.4.6 By comparing MAPE score of Double exponential smoothing and Triple exponential Smoothing I would say Double exponential smoothing performs better and MAPE / Residual score are low when compare to Triple exponential smoothing

```
[ ]:
```