In [7]:

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

In [8]:

```
data = pd.read_excel(r"C:\Users\Shri Ganesha\Documents\akash doc\msc\project\data.xlsx")
data.head()
```

Out[8]:

	Month	Export	Import
0	2011-01-01	1030.06	1514.03
1	2011-02-01	1056.09	1498.17
2	2011-03-01	1368.57	1541.72
3	2011-04-01	1041.34	1623.76
4	2011-05-01	1190.95	2032.12

In [9]:

```
len(data)
```

Out[9]:

```
132
```

In [10]:

```
data = data.set_index(['Month'])
```

In [11]:

```
data.head()
```

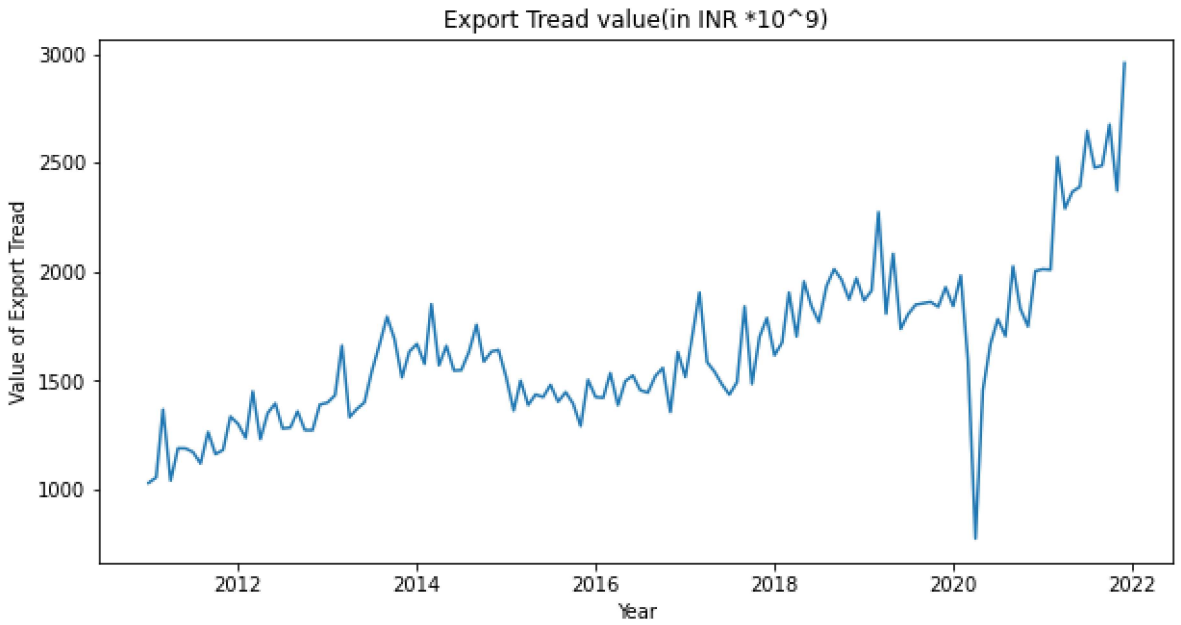
Out[11]:

Month	Export	Import
2011-01-01	1030.06	1514.03
2011-02-01	1056.09	1498.17
2011-03-01	1368.57	1541.72
2011-04-01	1041.34	1623.76
2011-05-01	1190.95	2032.12

Forecasting for Export

In [12]:

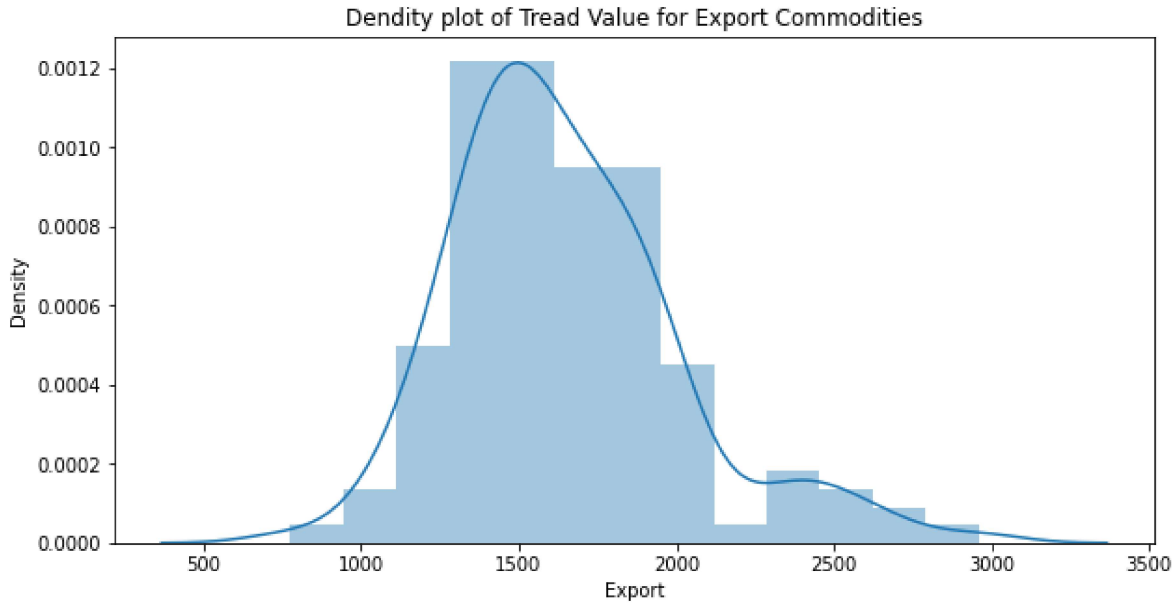
```
plt.figure(figsize=(10,5))
plt.title("Export Tread value(in INR *10^9)")
plt.plot(data['Export'])
plt.xlabel('Year')
plt.ylabel('Value of Export Tread')
plt.show()
```



In [13]:

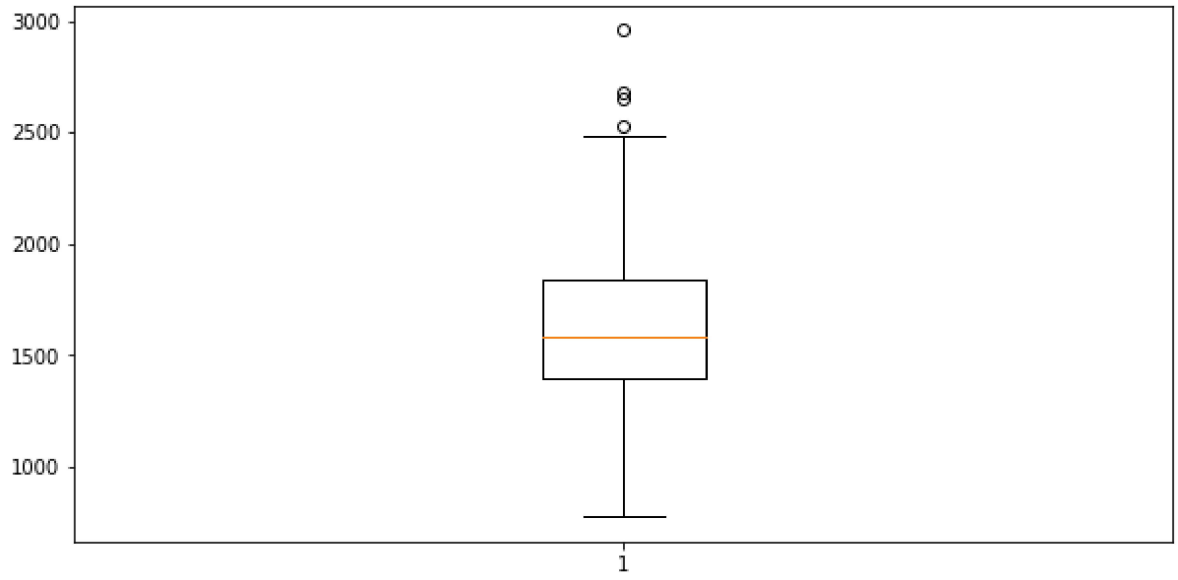
```
plt.figure(figsize=(10,5))
sns.distplot(data['Export'])
plt.title('Dendity plot of Tread Value for Export Commodities')
plt.show()
```

```
C:\Users\Shri Ganesha\AppData\Local\Programs\Python\Python39\lib\site-packag
es\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated
function and will be removed in a future version. Please adapt your code to
use either `displot` (a figure-level function with similar flexibility) or `
histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```



In [14]:

```
plt.figure(figsize=(10,5))
plt.boxplot(data['Export'])
plt.show()
```



In [15]:

```
import matplotlib.pyplot as plt
import seaborn as sns
from statsmodels.tsa.seasonal import seasonal_decompose
from statsmodels.tsa.holtwinters import SimpleExpSmoothing # SES
from statsmodels.tsa.holtwinters import Holt # Holts Exponential Smoothing
from statsmodels.tsa.holtwinters import ExponentialSmoothing
from statsmodels.tsa.stattools import adfuller
from sklearn.metrics import mean_squared_error
```

Testing For Stationarity of Data using Statsmodels adfuller

In [16]:

```
print('Observations of Dickey-fuller test')
dftest = adfuller(data['Export'],autolag = 'AIC')
dfoutput = pd.Series(dftest[0:4],index = ['Test Statistic','p-value','Lag Used','Number of
for key,value in dftest[4].items():
    dfoutput['Critical Value (%s)'%key] = value
print(dfoutput)
```

```
Observations of Dickey-fuller test
Test Statistic                  -0.202247
p-value                          0.938253
Lag Used                        13.000000
Number of observations Used    118.000000
Critical Value (1%)             -3.487022
Critical Value (5%)             -2.886363
Critical Value (10%)            -2.580009
dtype: float64
```

In [17]:

```
diff_export = data['Export'].diff()
print('Observations of Dickey-fuller test')
dftest = adfuller(diff_export.dropna(),autolag = 'AIC')
dfoutput = pd.Series(dftest[0:4],index = ['Test Statistic','p-value','Lag Used','Number of
for key,value in dftest[4].items():
    dfoutput['Critical Value (%s)'%key] = value
print(dfoutput)
```

```
Observations of Dickey-fuller test
Test Statistic                  -2.347443
p-value                          0.157123
Lag Used                        12.000000
Number of observations Used    118.000000
Critical Value (1%)             -3.487022
Critical Value (5%)             -2.886363
Critical Value (10%)            -2.580009
dtype: float64
```

In [18]:

```
diff2_export = data['Export'].diff().diff()
print('Observations of Dickey-fuller test')
dftest = adfuller(diff2_export.dropna(),autolag = 'AIC')
dfoutput = pd.Series(dftest[0:4],index = ['Test Statistic','p-value','Lag Used','Number of
for key,value in dftest[4].items():
    dfoutput['Critical Value (%s)'%key] = value
print(dfoutput)
```

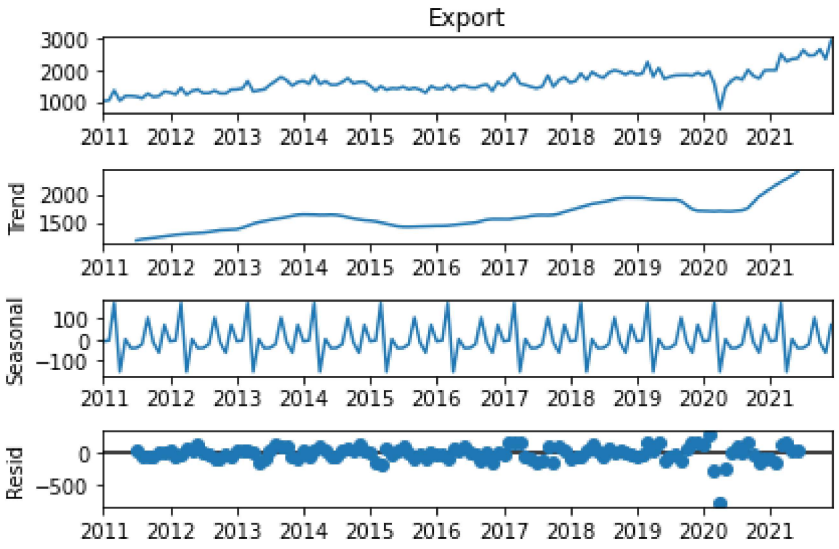
```
Observations of Dickey-fuller test
Test Statistic                 -7.332339e+00
p-value                         1.119351e-10
Lag Used                        1.000000e+01
Number of observations Used     1.190000e+02
Critical Value (1%)            -3.486535e+00
Critical Value (5%)            -2.886151e+00
Critical Value (10%)           -2.579896e+00
dtype: float64
```

ACF and PACF Plot

Decomposition

In [19]:

```
decompose_ts_add = seasonal_decompose(data['Export'],period=12)
decompose_ts_add.plot()
plt.show()
```



In [20]:

```
train = data[:110]
test = data[110:]
```

In [21]:

```
print(train.shape,test.shape)
```

```
(110, 2) (22, 2)
```

ARIMA Model

In [22]:

```
from pmdarima import auto_arima
model = auto_arima(train['Export'], start_p=1, start_q=1,
                   max_p=7, max_q=7,
                   m=12,  ###################################
                   seasonal=True,
                   start_P=0,
                   D=None,
                   trace=True,
                   error_action='ignore',
                   suppress_warnings=True,
                   stepwise=True)
```

```
Performing stepwise search to minimize aic
 ARIMA(1,1,1)(0,0,1)[12] intercept   : AIC=1355.989, Time=0.92 sec
 ARIMA(0,1,0)(0,0,0)[12] intercept   : AIC=1412.779, Time=0.01 sec
 ARIMA(1,1,0)(1,0,0)[12] intercept   : AIC=1349.918, Time=0.42 sec
 ARIMA(0,1,1)(0,0,1)[12] intercept   : AIC=1354.277, Time=0.42 sec
 ARIMA(0,1,0)(0,0,0)[12]             : AIC=1411.125, Time=0.01 sec
 ARIMA(1,1,0)(0,0,0)[12] intercept   : AIC=1376.643, Time=0.03 sec
 ARIMA(1,1,0)(2,0,0)[12] intercept   : AIC=1345.321, Time=0.82 sec
 ARIMA(1,1,0)(2,0,1)[12] intercept   : AIC=inf, Time=1.81 sec
 ARIMA(1,1,0)(1,0,1)[12] intercept   : AIC=inf, Time=0.79 sec
 ARIMA(0,1,0)(2,0,0)[12] intercept   : AIC=1365.595, Time=0.45 sec
 ARIMA(2,1,0)(2,0,0)[12] intercept   : AIC=1347.014, Time=1.00 sec
 ARIMA(1,1,1)(2,0,0)[12] intercept   : AIC=1346.335, Time=1.09 sec
 ARIMA(0,1,1)(2,0,0)[12] intercept   : AIC=1344.340, Time=0.76 sec
 ARIMA(0,1,1)(1,0,0)[12] intercept   : AIC=1347.721, Time=0.41 sec
 ARIMA(0,1,1)(2,0,1)[12] intercept   : AIC=inf, Time=1.42 sec
 ARIMA(0,1,1)(1,0,1)[12] intercept   : AIC=inf, Time=1.02 sec
 ARIMA(0,1,2)(2,0,0)[12] intercept   : AIC=1346.338, Time=0.88 sec
 ARIMA(1,1,2)(2,0,0)[12] intercept   : AIC=inf, Time=1.80 sec
 ARIMA(0,1,1)(2,0,0)[12]             : AIC=1343.004, Time=0.39 sec
 ARIMA(0,1,1)(1,0,0)[12]             : AIC=1346.601, Time=0.23 sec
 ARIMA(0,1,1)(2,0,1)[12]             : AIC=inf, Time=1.84 sec
 ARIMA(0,1,1)(1,0,1)[12]             : AIC=inf, Time=0.46 sec
 ARIMA(0,1,0)(2,0,0)[12]             : AIC=1363.696, Time=0.20 sec
 ARIMA(1,1,1)(2,0,0)[12]             : AIC=1345.004, Time=0.53 sec
 ARIMA(0,1,2)(2,0,0)[12]             : AIC=1345.004, Time=0.45 sec
 ARIMA(1,1,0)(2,0,0)[12]             : AIC=1343.663, Time=0.32 sec
 ARIMA(1,1,2)(2,0,0)[12]             : AIC=1346.199, Time=1.29 sec

Best model:  ARIMA(0,1,1)(2,0,0)[12]
Total fit time: 19.922 seconds
```

In [23]:

```
pred_arima = model.predict(len(test))
```

In [24]:

```
RMSE1 = np.sqrt(mean_squared_error(test['Export'],pred_arima))
RMSE1
```

Out[24]:

```
480.5271034036899
```

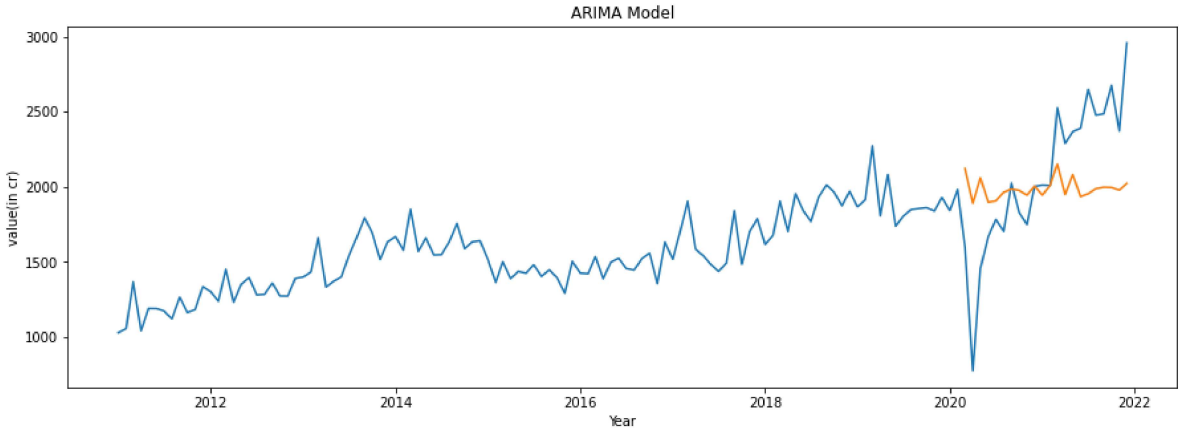
In [25]:

```
prediction_series = pd.Series(pred_arima,index = test.index)

fig, ax = plt.subplots(1, 1, figsize=(15, 5))
plt.title("ARIMA Model")
plt.plot(data['Export'])
plt.plot(prediction_series)
plt.ylabel("value(in cr)")
plt.xlabel('Year')
```

Out[25]:

```
Text(0.5, 0, 'Year')
```



Holts winter exponential smoothing with additive seasonality and additive trend

In [26]:

```
hwe_model_add_add = ExponentialSmoothing(train['Export'],seasonal="add",trend="add",seasona
pred_hwe_add_add = hwe_model_add_add.predict(start = test.index[0],end = test.index[-1])
RMSE2 =np.sqrt(mean_squared_error(test['Export'],pred_hwe_add_add))
```

```
C:\Users\Shri Ganesha\AppData\Local\Programs\Python\Python39\lib\site-packag
es\statsmodels\tsa\base\tsa_model.py:536: ValueWarning: No frequency informa
tion was provided, so inferred frequency MS will be used.
  warnings.warn('No frequency information was'
```

In [27]:

```
RMSE2
```

Out[27]:

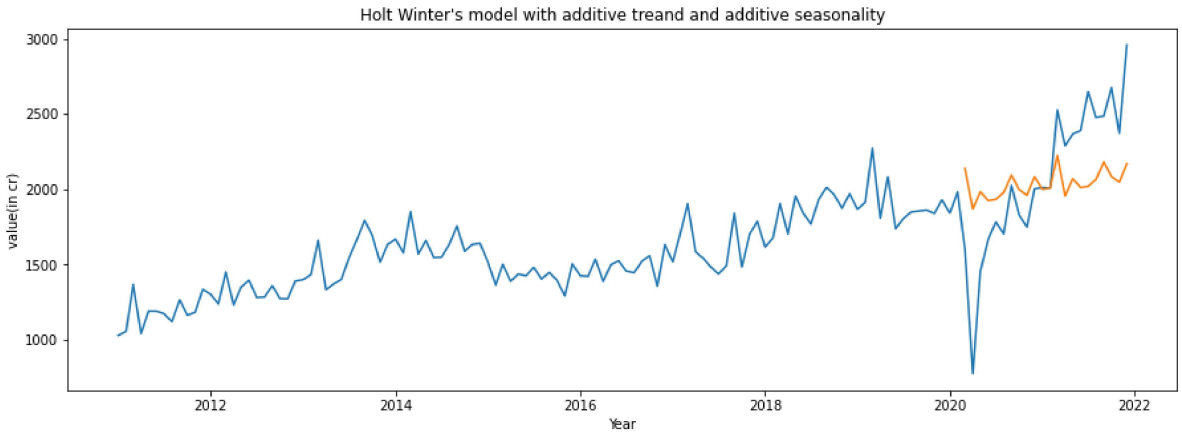
```
435.9089439742816
```

In [28]:

```
prediction_series = pd.Series(pred_hwe_add_add,index = test.index)
fig, ax = plt.subplots(1, 1, figsize=(15, 5))
plt.title("Holt Winter's model with additive treand and additive seasonality")
plt.plot(data['Export'])
plt.plot(prediction_series)
plt.ylabel("value(in cr)")
plt.xlabel('Year')
```

Out[28]:

```
Text(0.5, 0, 'Year')
```



Holts winter exponential smoothing with multiplicative seasonality and additive trend

In [29]:

```
hwe_model_mul_add = ExponentialSmoothing(train["Export"],seasonal="mul",trend="add",seasona
pred_hwe_mul_add = hwe_model_mul_add.predict(start = test.index[0],end = test.index[-1])
RMSE3 =np.sqrt(mean_squared_error(test['Export'],pred_hwe_mul_add))
```

```
C:\Users\Shri Ganesha\AppData\Local\Programs\Python\Python39\lib\site-packag
es\statsmodels\tsa\base\tsa_model.py:536: ValueWarning: No frequency informa
tion was provided, so inferred frequency MS will be used.
  warnings.warn('No frequency information was'
```

In [30]:

```
RMSE3
```

Out[30]:

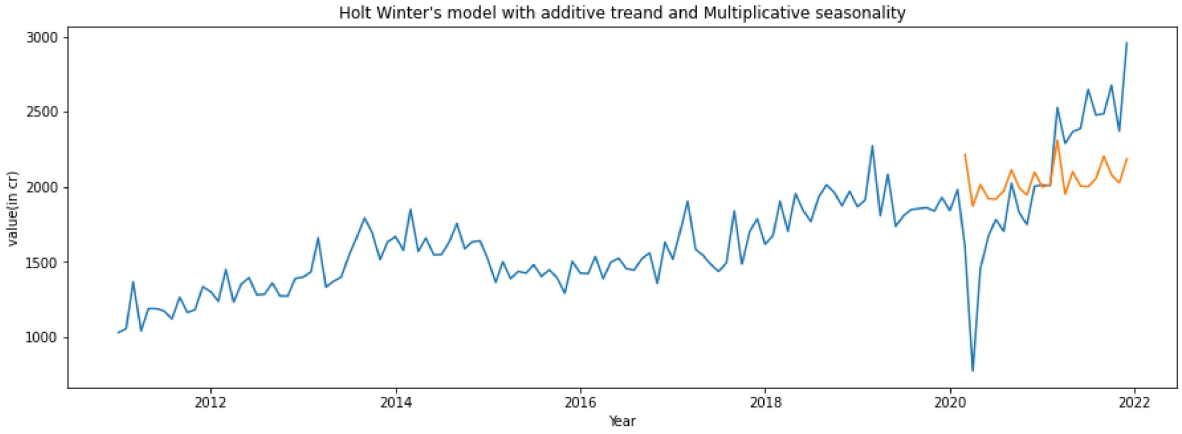
```
439.93609062230223
```

In [31]:

```
prediction_series = pd.Series(pred_hwe_mul_add,index = test.index)
fig, ax = plt.subplots(1, 1, figsize=(15, 5))
plt.title("Holt Winter's model with additive treand and Multiplicative seasonality")
plt.plot(data['Export'])
plt.plot(prediction_series)
plt.ylabel("value(in cr)")
plt.xlabel('Year')
```

Out[31]:

```
Text(0.5, 0, 'Year')
```



Final Model by comparing RMSE values

In [32]:

```
rmse_df = pd.DataFrame({'Model':['ARIMA','ExponentialSmoothing_add','ExponentialSmoothing_m
```

In [33]:

```
rmse_df
```

Out[33]:

	Model	RMSE
0	ARIMA	480.527103
1	ExponentialSmoothing_add	435.908944
2	ExponentialSmoothing_mul	439.936091

ExponentialSmoothing_add have a smallest RMSE from above three model

In [34]:

```
hwe_model_add_add = ExponentialSmoothing(data['Export'],seasonal="add",trend="add",seasonal
```

```
C:\Users\Shri Ganesha\AppData\Local\Programs\Python\Python39\lib\site-packag
es\statsmodels\tsa\base\tsa_model.py:536: ValueWarning: No frequency informa
tion was provided, so inferred frequency MS will be used.
  warnings.warn('No frequency information was'
```

In [35]:

```
hwe_model_add_add.forecast(24)
```

Out[35]:

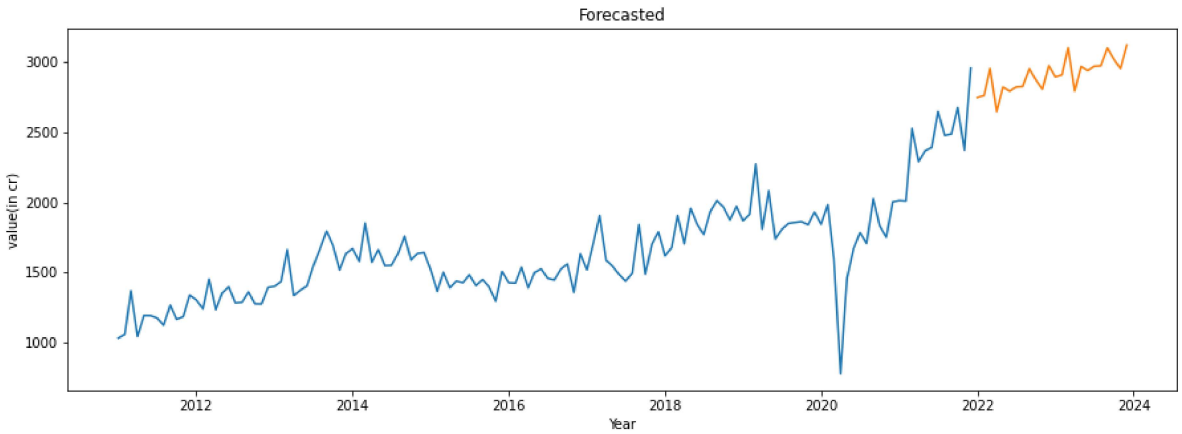
```
2022-01-01    2747.288998
2022-02-01    2762.445429
2022-03-01    2954.910465
2022-04-01    2646.032225
2022-05-01    2821.683138
2022-06-01    2792.503134
2022-07-01    2823.158505
2022-08-01    2826.429413
2022-09-01    2953.116689
2022-10-01    2873.551392
2022-11-01    2806.587931
2022-12-01    2974.701568
2023-01-01    2894.713372
2023-02-01    2909.869803
2023-03-01    3102.334839
2023-04-01    2793.456600
2023-05-01    2969.107512
2023-06-01    2939.927508
2023-07-01    2970.582879
2023-08-01    2973.853787
2023-09-01    3100.541063
2023-10-01    3020.975766
2023-11-01    2954.012305
2023-12-01    3122.125942
Freq: MS, dtype: float64
```

In [36]:

```
forecast= pd.DataFrame(hwe_model_add_add.forecast(24))
```

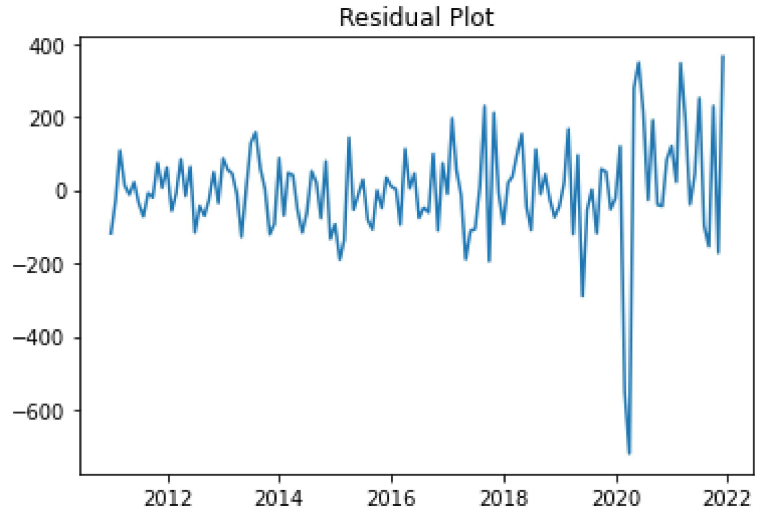
In [37]:

```
fig, ax = plt.subplots(1, 1, figsize=(15, 5))
plt.title("Forecasted")
plt.plot(data['Export'])
plt.plot(forecast)
plt.ylabel("value(in cr)")
plt.xlabel('Year')
plt.show()
```



In [38]:

```
residual = hwe_model_add_add.resid
plt.title("Residual Plot")
plt.plot(residual)
plt.show()
```



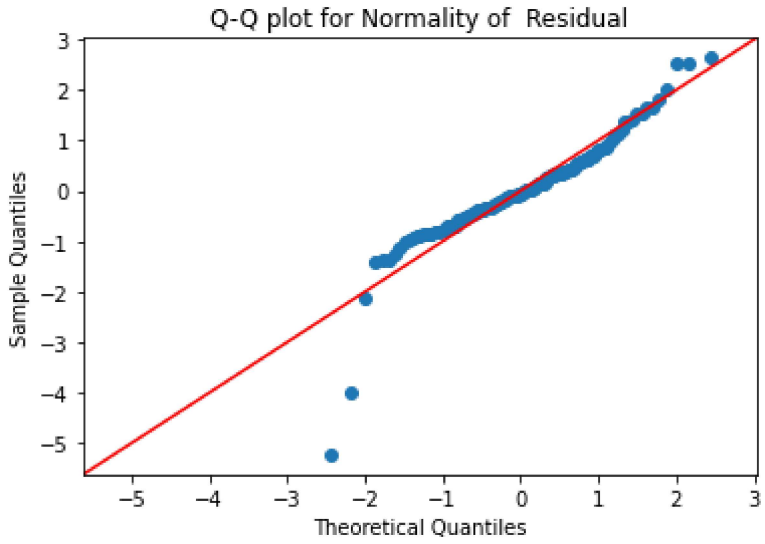
In [39]:

```
from statsmodels.api import qqplot,stats
import scipy.stats as stats

qqplot(residual,line='45',fit = True,dist = stats.norm)
plt.title("Q-Q plot for Normality of  Residual")
plt.plot()
```

Out[39]:

```
[]
```



In [40]:

```
stats.shapiro(residual)
```

Out[40]:

```
ShapiroResult(statistic=0.8961471319198608, pvalue=4.035624812104288e-08)
```

In [41]:

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

In [42]:

```
sm.stats.acorr_ljungbox(residual,return_df =True)
```

Out[42]:

	lb_stat	lb_pvalue
1	0.517833	0.471767
2	2.365774	0.306393
3	3.283248	0.349981
4	4.199752	0.379647
5	4.860211	0.433178
6	5.687676	0.459070
7	9.374379	0.226883
8	9.573710	0.296229
9	13.712874	0.132912
10	14.090268	0.168914

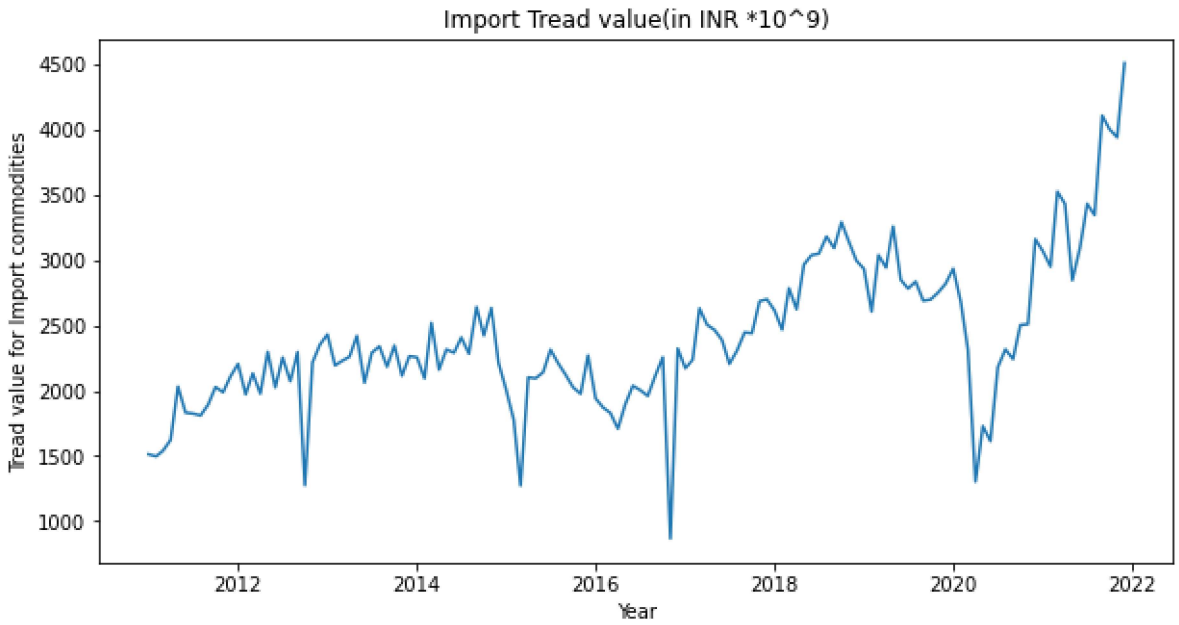
H0 : The residuals are independently distributrd
H1 : The residual are not indendently distributed :the exabit serial correlation

p-value for all 10 lags are grater than 0.05, Hence we may accept H0 i.e Residuals are independently distributed

Forecasting import

In [43]:

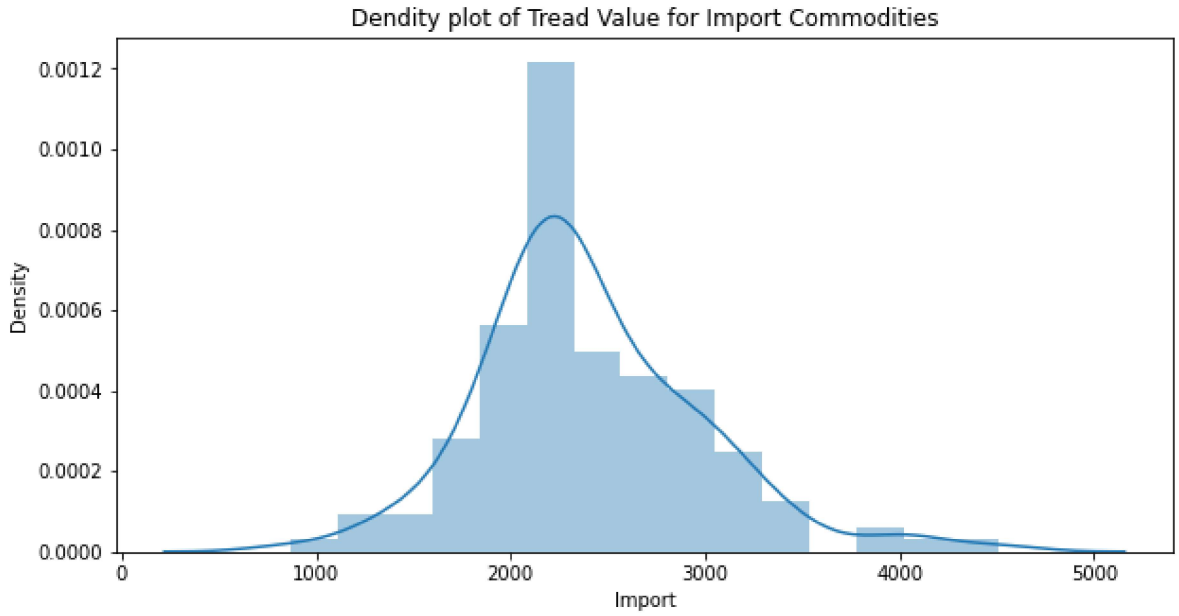
```
plt.figure(figsize=(10,5))
plt.title("Import Tread value(in INR *10^9)")
plt.plot(data['Import'])
plt.xlabel('Year')
plt.ylabel('Tread value for Import commodities')
plt.show()
```



In [44]:

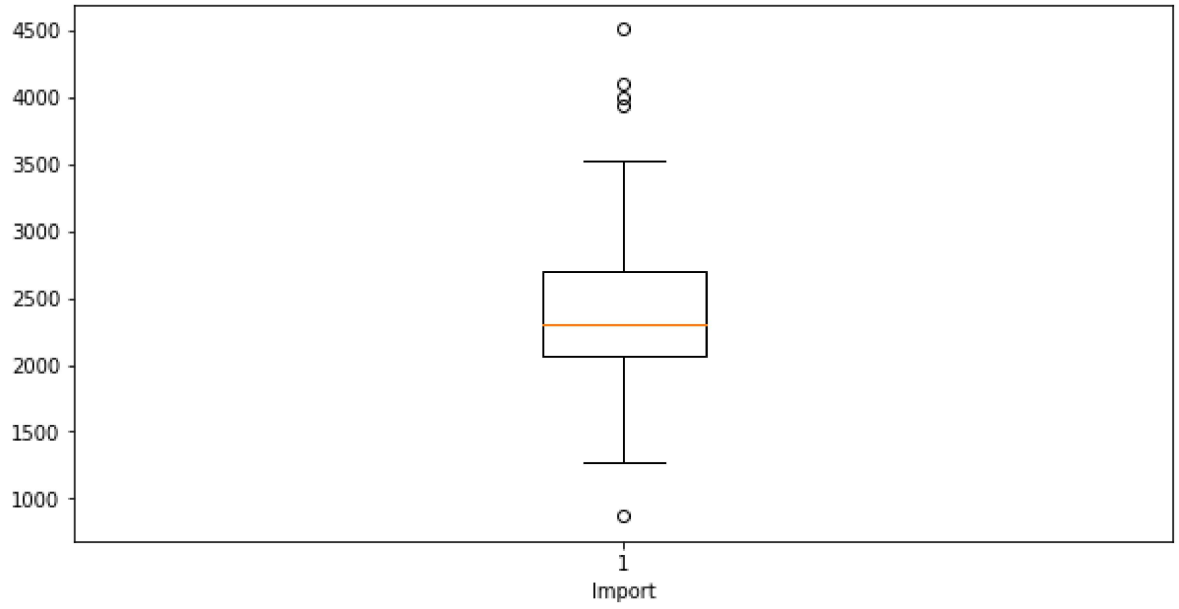
```
plt.figure(figsize=(10,5))
plt.title('Dendity plot of Tread Value for Import Commodities')
sns.distplot(data['Import'])
plt.show()
```

```
C:\Users\Shri Ganesha\AppData\Local\Programs\Python\Python39\lib\site-packag
es\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated
function and will be removed in a future version. Please adapt your code to
use either `displot` (a figure-level function with similar flexibility) or `
histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```



In [45]:

```
plt.figure(figsize=(10,5))
plt.boxplot(data['Import'])
plt.xlabel("Import")
plt.show()
```



In [46]:

```
import matplotlib.pyplot as plt
import seaborn as sns
from statsmodels.tsa.seasonal import seasonal_decompose
from statsmodels.tsa.holtwinters import SimpleExpSmoothing # SES
from statsmodels.tsa.holtwinters import Holt # Holts Exponential Smoothing
from statsmodels.tsa.holtwinters import ExponentialSmoothing
from statsmodels.tsa.stattools import adfuller
from sklearn.metrics import mean_squared_error
```

Testing For Stationarity of Data using Statsmodels adfuller

In [47]:

```
print('Observations of Dickey-fuller test')
dftest = adfuller(data['Import'],autolag = 'AIC')
dfoutput = pd.Series(dftest[0:4],index = ['Test Statistic','p-value','Lag Used','Number of
for key,value in dftest[4].items():
    dfoutput['Critical Value (%s)'%key] = value
print(dfoutput)
```

```
Observations of Dickey-fuller test
Test Statistic                  -1.257123
p-value                          0.648553
Lag Used                         1.000000
Number of observations Used    130.000000
Critical Value (1%)             -3.481682
Critical Value (5%)             -2.884042
Critical Value (10%)            -2.578770
dtype: float64
```

In [48]:

```
diff_Import = data['Import'].diff()
print('Observations of Dickey-fuller test')
dftest = adfuller(diff_Import.dropna(),autolag = 'AIC')
dfoutput = pd.Series(dftest[0:4],index = ['Test Statistic','p-value','Lag Used','Number of
for key,value in dftest[4].items():
    dfoutput['Critical Value (%s)'%key] = value
print(dfoutput)
```

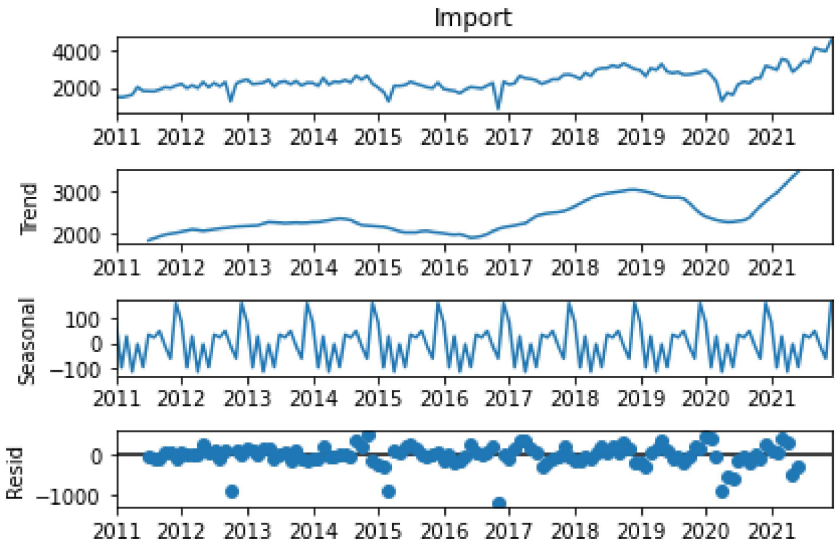
```
Observations of Dickey-fuller test
Test Statistic                -1.727543e+01
p-value                        5.825761e-30
Lag Used                       0.000000e+00
Number of observations Used    1.300000e+02
Critical Value (1%)           -3.481682e+00
Critical Value (5%)           -2.884042e+00
Critical Value (10%)          -2.578770e+00
dtype: float64
```

ACF and PACF Plot

Decomposition

In [49]:

```
decompose_ts_add = seasonal_decompose(data['Import'],period=12)
decompose_ts_add.plot()
plt.show()
```



In [50]:

```
train = data[:110]
test = data[110:]
```

In [51]:

```
print(train.shape,test.shape)
```

```
(110, 2) (22, 2)
```

ARIMA Model

In [52]:

```
from pmdarima import auto_arima
model = auto_arima(train['Import'], start_p=1, start_q=1,
                   max_p=7, max_q=7,
                   m=12,  ###################################
                   seasonal=True,
                   start_P=0,
                   D=None,
                   trace=True,
                   error_action='ignore',
                   suppress_warnings=True,
                   stepwise=True)
```

```
Performing stepwise search to minimize aic
 ARIMA(1,1,1)(0,0,1)[12] intercept   : AIC=1533.584, Time=0.56 sec
 ARIMA(0,1,0)(0,0,0)[12] intercept   : AIC=1565.951, Time=0.01 sec
 ARIMA(1,1,0)(1,0,0)[12] intercept   : AIC=1537.427, Time=0.19 sec
 ARIMA(0,1,1)(0,0,1)[12] intercept   : AIC=1531.593, Time=0.37 sec
 ARIMA(0,1,0)(0,0,0)[12]             : AIC=1564.079, Time=0.01 sec
 ARIMA(0,1,1)(0,0,0)[12] intercept   : AIC=1529.634, Time=0.13 sec
 ARIMA(0,1,1)(1,0,0)[12] intercept   : AIC=1531.603, Time=0.39 sec
 ARIMA(0,1,1)(1,0,1)[12] intercept   : AIC=inf, Time=0.59 sec
 ARIMA(1,1,1)(0,0,0)[12] intercept   : AIC=1531.629, Time=0.17 sec
 ARIMA(0,1,2)(0,0,0)[12] intercept   : AIC=1531.631, Time=0.24 sec
 ARIMA(1,1,0)(0,0,0)[12] intercept   : AIC=1535.549, Time=0.02 sec
 ARIMA(1,1,2)(0,0,0)[12] intercept   : AIC=1532.946, Time=0.31 sec
 ARIMA(0,1,1)(0,0,0)[12]             : AIC=1529.150, Time=0.06 sec
 ARIMA(0,1,1)(1,0,0)[12]             : AIC=1531.143, Time=0.16 sec
 ARIMA(0,1,1)(0,0,1)[12]             : AIC=1531.141, Time=0.18 sec
 ARIMA(0,1,1)(1,0,1)[12]             : AIC=1532.995, Time=0.45 sec
 ARIMA(1,1,1)(0,0,0)[12]             : AIC=1531.067, Time=0.10 sec
 ARIMA(0,1,2)(0,0,0)[12]             : AIC=1531.103, Time=0.11 sec
 ARIMA(1,1,0)(0,0,0)[12]             : AIC=1534.020, Time=0.02 sec
 ARIMA(1,1,2)(0,0,0)[12]             : AIC=1532.399, Time=0.28 sec

Best model:  ARIMA(0,1,1)(0,0,0)[12]
Total fit time: 4.412 seconds
```

In [53]:

```
pred_arima = model.predict(len(test))
RMSE1 = np.sqrt(mean_squared_error(test['Import'],pred_arima))
RMSE1
```

Out[53]:

```
840.6929338674336
```

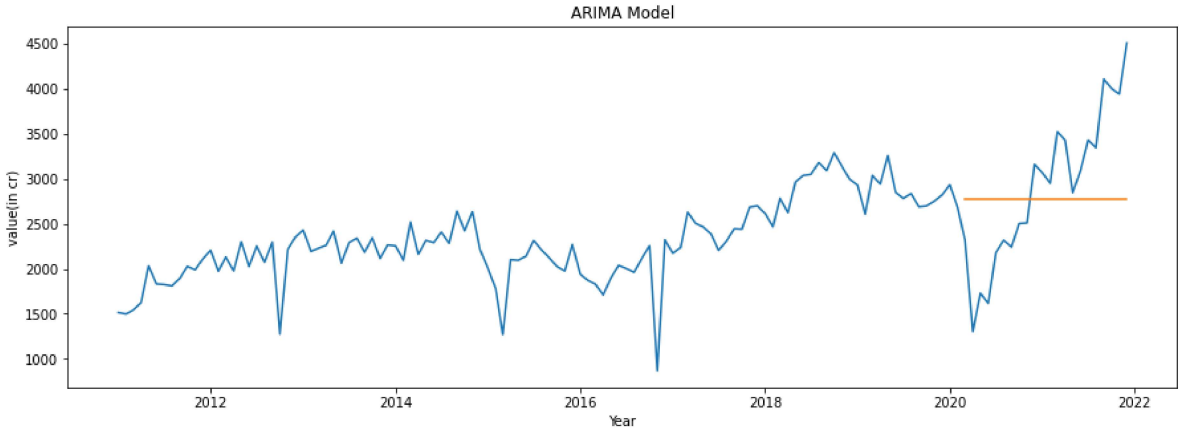
In [54]:

```
prediction_series = pd.Series(pred_arima,index = test.index)

fig, ax = plt.subplots(1, 1, figsize=(15, 5))
plt.title("ARIMA Model")
plt.plot(data['Import'])
plt.plot(prediction_series)
plt.ylabel("value(in cr)")
plt.xlabel('Year')
```

Out[54]:

```
Text(0.5, 0, 'Year')
```



Holts winter exponential smoothing with additive seasonality and additive trend

In [55]:

```
hwe_model_add_add = ExponentialSmoothing(train['Import'],seasonal="add",trend="add",seasona
pred_hwe_add_add = hwe_model_add_add.predict(start = test.index[0],end = test.index[-1])
RMSE2 =np.sqrt(mean_squared_error(test['Import'],pred_hwe_add_add))
```

```
C:\Users\Shri Ganesha\AppData\Local\Programs\Python\Python39\lib\site-packag
es\statsmodels\tsa\base\tsa_model.py:536: ValueWarning: No frequency informa
tion was provided, so inferred frequency MS will be used.
  warnings.warn('No frequency information was'
```

In [56]:

```
RMSE2
```

Out[56]:

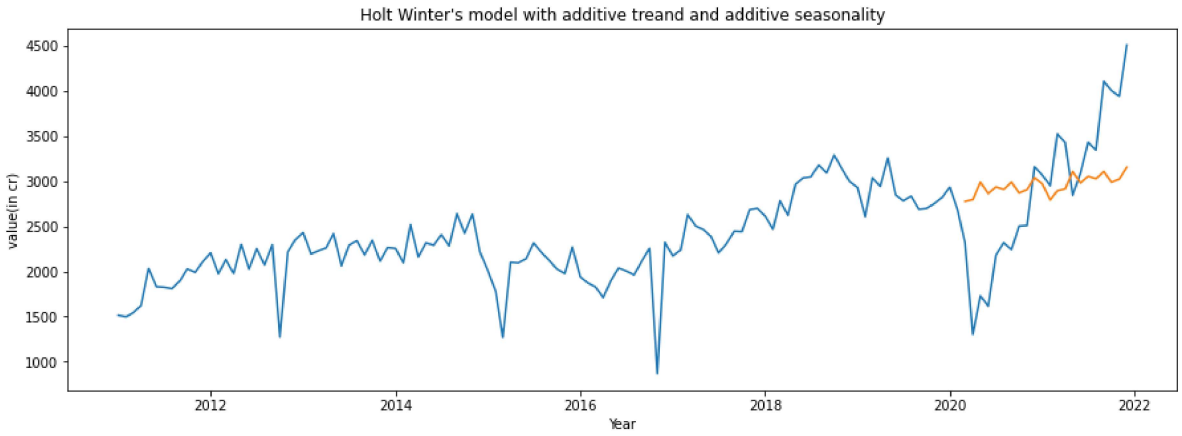
```
771.7282306265721
```

In [57]:

```
prediction_series = pd.Series(pred_hwe_add_add,index = test.index)
fig, ax = plt.subplots(1, 1, figsize=(15, 5))
plt.title("Holt Winter's model with additive treand and additive seasonality")
plt.plot(data['Import'])
plt.plot(prediction_series)
plt.ylabel("value(in cr)")
plt.xlabel('Year')
```

Out[57]:

```
Text(0.5, 0, 'Year')
```



Holts winter exponential smoothing with multiplicative seasonality and additive trend

In [58]:

```
hwe_model_mul_add = ExponentialSmoothing(train["Import"],seasonal="mul",trend="add",seasona
pred_hwe_mul_add = hwe_model_mul_add.predict(start = test.index[0],end = test.index[-1])
RMSE3 =np.sqrt(mean_squared_error(test['Import'],pred_hwe_mul_add))
```

```
C:\Users\Shri Ganesha\AppData\Local\Programs\Python\Python39\lib\site-packag
es\statsmodels\tsa\base\tsa_model.py:536: ValueWarning: No frequency informa
tion was provided, so inferred frequency MS will be used.
  warnings.warn('No frequency information was'
```

In [59]:

```
RMSE3
```

Out[59]:

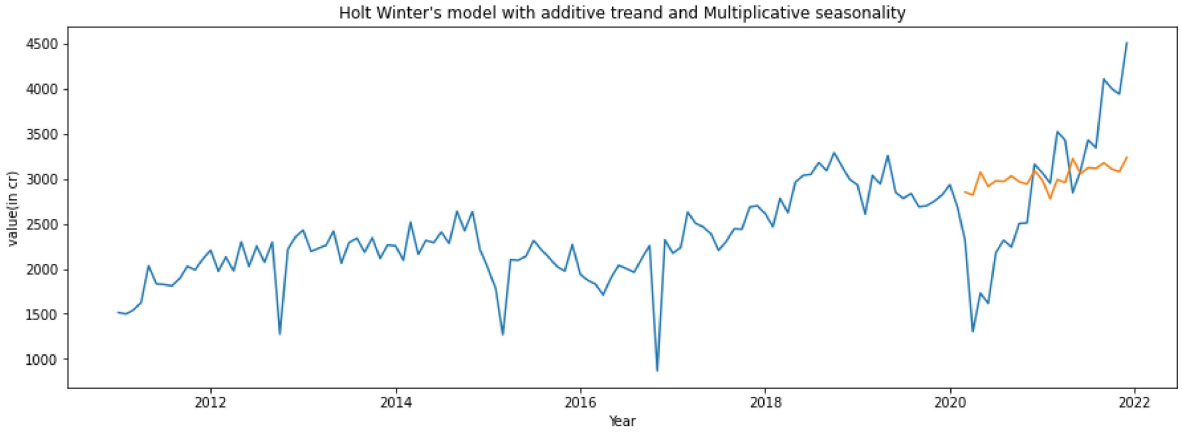
```
769.8621735680146
```

In [60]:

```
prediction_series = pd.Series(pred_hwe_mul_add,index = test.index)
fig, ax = plt.subplots(1, 1, figsize=(15, 5))
plt.title("Holt Winter's model with additive treand and Multiplicative seasonality")
plt.plot(data['Import'])
plt.plot(prediction_series)
plt.ylabel("value(in cr)")
plt.xlabel('Year')
```

Out[60]:

```
Text(0.5, 0, 'Year')
```



Final Model by combining train and test

In [61]:

```
rmse_df = pd.DataFrame({'Model':['ARIMA','ExponentialSmoothing_add','ExponentialSmoothing_m
```

In [62]:

```
rmse_df
```

Out[62]:

	Model	RMSE
0	ARIMA	840.692934
1	ExponentialSmoothing_add	771.728231
2	ExponentialSmoothing_mul	769.862174

Here we have less RMSE value for Holt winter's model with additive trend and multiplicative seasonality

So now use Holt winter's model with additive trend and multiplicative seasonality for forecasting

In [63]:

```
hwe_model_mul_add = ExponentialSmoothing(data["Import"],seasonal="mul",trend="add",seasonal
```

```
C:\Users\Shri Ganesha\AppData\Local\Programs\Python\Python39\lib\site-packag
es\statsmodels\tsa\base\tsa_model.py:536: ValueWarning: No frequency informa
tion was provided, so inferred frequency MS will be used.
  warnings.warn('No frequency information was'
```

In [64]:

```
forecast= pd.DataFrame(hwe_model_mul_add.forecast(24))
```

In [65]:

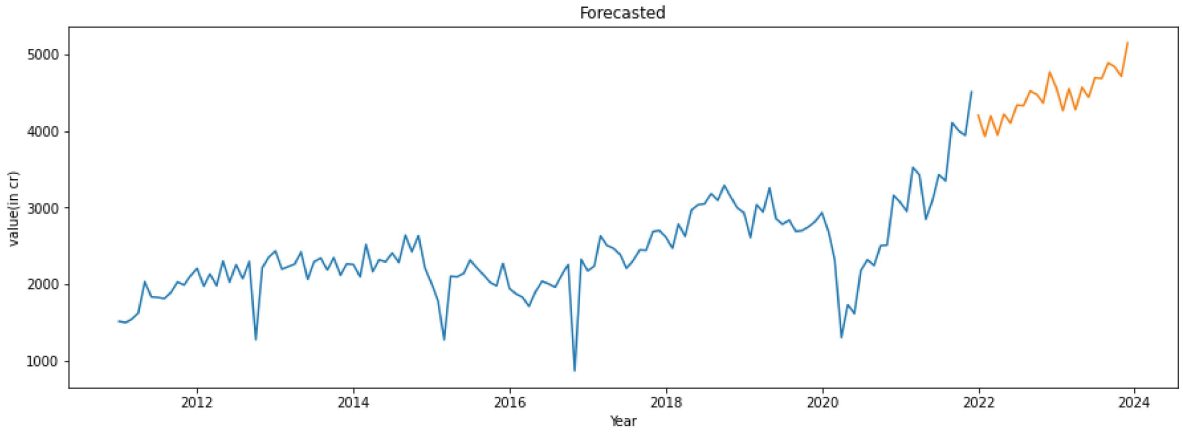
```
forecast
```

Out[65]:

	0
2022-01-01	4204.833164
2022-02-01	3928.642566
2022-03-01	4196.489215
2022-04-01	3944.819531
2022-05-01	4217.916990
2022-06-01	4102.753575
2022-07-01	4338.294537
2022-08-01	4330.374507
2022-09-01	4522.772050
2022-10-01	4477.873820
2022-11-01	4364.448988
2022-12-01	4770.564872
2023-01-01	4564.857555
2023-02-01	4262.636000
2023-03-01	4550.743878
2023-04-01	4275.502745
2023-05-01	4569.040394
2023-06-01	4441.937158
2023-07-01	4694.496803
2023-08-01	4683.510257
2023-09-01	4889.108026
2023-10-01	4838.141376
2023-11-01	4713.252361
2023-12-01	5149.302344

In [66]:

```
fig, ax = plt.subplots(1, 1, figsize=(15, 5))
plt.title("Forecasted")
plt.plot(data['Import'])
plt.plot(forecast)
plt.ylabel("value(in cr)")
plt.xlabel('Year')
plt.show()
```



#Residual Normality of ARIMA model

#Resdual normality of Howlt winter's model with additive seasonality and additive treand

In [67]:

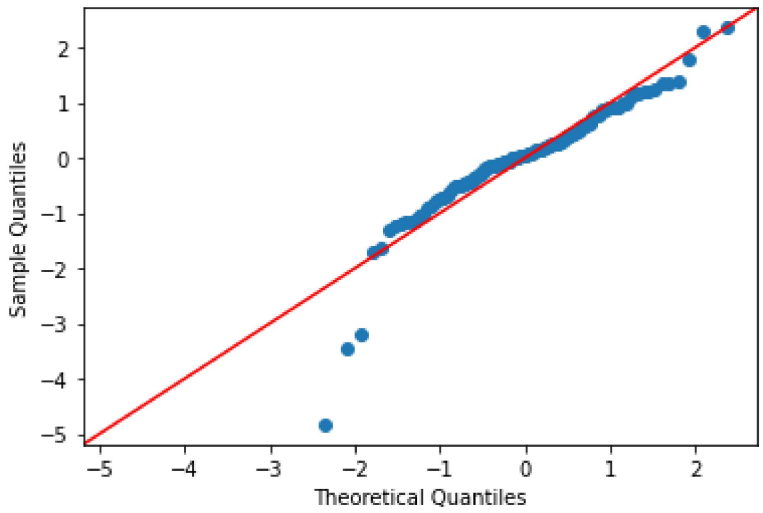
```
residual_import = hwe_model_add_add.resid
```

In [68]:

```
from statsmodels.api import qqplot
import scipy.stats as stats
qqplot(residual_import,line='45',fit = True,dist = stats.norm)
plt.plot()
```

Out[68]:

```
[]
```



In [69]:

```
stats.shapiro(residual_import)
```

Out[69]:

```
ShapiroResult(statistic=0.8989105224609375, pvalue=4.565797269151517e-07)
```

Here p-value is less than 0.05 ,We reject null hypothesis i.e we can say that data does not came from normal distribution.

In [70]:

```
sm.stats.acorr_ljungbox(residual_import,return_df =True)
```

Out[70]:

	lb_stat	lb_pvalue
1	0.003762	0.951091
2	2.264908	0.322242
3	2.686582	0.442512
4	7.048423	0.133351
5	8.433541	0.133906
6	8.462912	0.206112
7	8.784852	0.268474
8	11.921674	0.154733
9	12.083326	0.208652
10	12.177764	0.273336

H0 : The residuals are independently distributrd
H1 : The residual are not indendently distributed :the exabit serial correlation

p-value for all 10 lags are grater than 0.05, Hence we may accept H0 i.e Residuals are independently distributed

In [ ]:

```

```