Time Series Analysis and forecasting classical and Deep learning approach on LBMA gold price

In [1]:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
from datetime import date as dt
import warnings
warnings.filterwarnings('ignore')
import statsmodels.api as sm
from pylab import rcParams

from statsmodels.tsa.seasonal import seasonal_decompose
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
from statsmodels.tsa.stattools import adfuller
from sklearn.model_selection import train_test_split
from statsmodels.tsa.arima.model import ARIMA
from sklearn.metrics import mean_squared_error
from statsmodels.tsa.statespace.sarimax import SARIMAX
```

In [2]:

```
tsd=pd.read_csv("Daily.csv")
tsd.head()
```

Out[2]:

	Date	USD	EUR	JPY	GBP	CAD	CHF	INR	CNY	TRY	SAR	ID
0	12/29/1978	226.0	137.1	NaN	110.7	NaN	NaN	NaN	NaN	NaN	NaN	Na
1	1/1/1979	226.0	137.1	NaN	110.7	NaN	NaN	NaN	NaN	NaN	NaN	Na
2	1/2/1979	226.8	137.3	43,164.9	111.5	263.7	359.6	1,792.9	NaN	NaN	735.6	138,160
3	1/3/1979	218.6	134.0	43,717.9	108.0	264.4	365.9	1,802.2	NaN	NaN	739.4	138,877
4	1/4/1979	223.2	136.8	43,674.9	110.7	264.1	366.4	1,811.7	NaN	NaN	743.4	139,616

In [3]:
```
tsd.set_index(tsd['Date'],inplace=True)
tsd
```

Out[3]:

	Date	USD	EUR	JPY	GBP	CAD	CHF	INR	CN
Date									
12/29/1978	12/29/1978	226.0	137.1	NaN	110.7	NaN	NaN	NaN	Na
1/1/1979	1/1/1979	226.0	137.1	NaN	110.7	NaN	NaN	NaN	Na
1/2/1979	1/2/1979	226.8	137.3	43,164.9	111.5	263.7	359.6	1,792.9	Na
1/3/1979	1/3/1979	218.6	134.0	43,717.9	108.0	264.4	365.9	1,802.2	Na
1/4/1979	1/4/1979	223.2	136.8	43,674.9	110.7	264.1	366.4	1,811.7	Na
...	...	...	...	...	...	...	...	...	
7/17/2023	7/17/2023	1,949.6	1,735.6	270,624.0	1,491.1	2,568.8	1,677.6	159,940.3	13,982
7/18/2023	7/18/2023	1,975.0	1,761.0	274,742.3	1,515.2	2,604.6	1,697.5	162,067.1	14,192
7/19/2023	7/19/2023	1,975.4	1,765.0	275,818.1	1,530.2	2,602.1	1,698.0	162,141.7	14,273
7/20/2023	7/20/2023	1,976.1	1,776.4	277,563.0	1,537.5	2,604.5	1,714.5	162,159.4	14,193
7/21/2023	7/21/2023	1,960.6	1,762.5	277,758.2	1,524.7	2,590.3	1,697.3	160,788.8	14,093

11626 rows × 20 columns

Univariate Time Series Analysis

In [4]:
```
df=tsd[['INR']]
df
```

Out[4]:

	INR
Date	
12/29/1978	NaN
1/1/1979	NaN
1/2/1979	1,792.9
1/3/1979	1,802.2
1/4/1979	1,811.7
...	...
7/17/2023	159,940.3
7/18/2023	162,067.1
7/19/2023	162,141.7
7/20/2023	162,159.4
7/21/2023	160,788.8

11626 rows × 1 columns

In [5]:
```
df.dtypes
```

Out[5]:
```
INR    object
dtype: object
```

In [6]:
```
df.isnull().sum()
```

Out[6]:
```
INR    2
dtype: int64
```

In [7]:
```
df = df.dropna(axis=0)
```

In [8]:
```
df
```

Out[8]:

	INR
Date	
1/2/1979	1,792.9
1/3/1979	1,802.2
1/4/1979	1,811.7
1/5/1979	1,843.6
1/8/1979	1,841.3
...	...
7/17/2023	159,940.3
7/18/2023	162,067.1
7/19/2023	162,141.7
7/20/2023	162,159.4
7/21/2023	160,788.8

11624 rows × 1 columns

In [9]:
```
df['INR'] =df.INR.str.replace(',','')
df
```

Out[9]:

	INR
Date	
1/2/1979	1792.9
1/3/1979	1802.2
1/4/1979	1811.7
1/5/1979	1843.6
1/8/1979	1841.3
...	...
7/17/2023	159940.3
7/18/2023	162067.1
7/19/2023	162141.7
7/20/2023	162159.4
7/21/2023	160788.8

11624 rows × 1 columns

In [10]:
```
df['INR'] = df['INR'].astype('float64')
df
```

Out[10]:

	INR
Date	
1/2/1979	1792.9
1/3/1979	1802.2
1/4/1979	1811.7
1/5/1979	1843.6
1/8/1979	1841.3
...	...
7/17/2023	159940.3
7/18/2023	162067.1
7/19/2023	162141.7
7/20/2023	162159.4
7/21/2023	160788.8

11624 rows × 1 columns

In [11]:
```
df.index= pd.to_datetime(df.index)
```

In [12]:
```
df.index
```

Out[12]:
```
DatetimeIndex(['1979-01-02', '1979-01-03', '1979-01-04', '1979-01-05',
               '1979-01-08', '1979-01-09', '1979-01-10', '1979-01-11',
               '1979-01-12', '1979-01-15',
               ...
               '2023-07-10', '2023-07-11', '2023-07-12', '2023-07-13',
               '2023-07-14', '2023-07-17', '2023-07-18', '2023-07-19',
               '2023-07-20', '2023-07-21'],
              dtype='datetime64[ns]', name='Date', length=11624, freq=Non
e)
```

In [13]:
```
df['year']=df.index.year
df['month'] =df.index.month
df['day'] = df.index.day
df['week'] = df.index.week
df['quarter'] = df.index.quarter
```

In [14]:
```
df
```

Out[14]:

	INR	year	month	day	week	quarter
Date						
1979-01-02	1792.9	1979	1	2	1	1
1979-01-03	1802.2	1979	1	3	1	1
1979-01-04	1811.7	1979	1	4	1	1
1979-01-05	1843.6	1979	1	5	1	1
1979-01-08	1841.3	1979	1	8	2	1
...	...	...	...	...	...	...
2023-07-17	159940.3	2023	7	17	29	3
2023-07-18	162067.1	2023	7	18	29	3
2023-07-19	162141.7	2023	7	19	29	3
2023-07-20	162159.4	2023	7	20	29	3
2023-07-21	160788.8	2023	7	21	29	3

11624 rows × 6 columns

In [15]:
```
df['weekday']=df.index.dayofweek
```

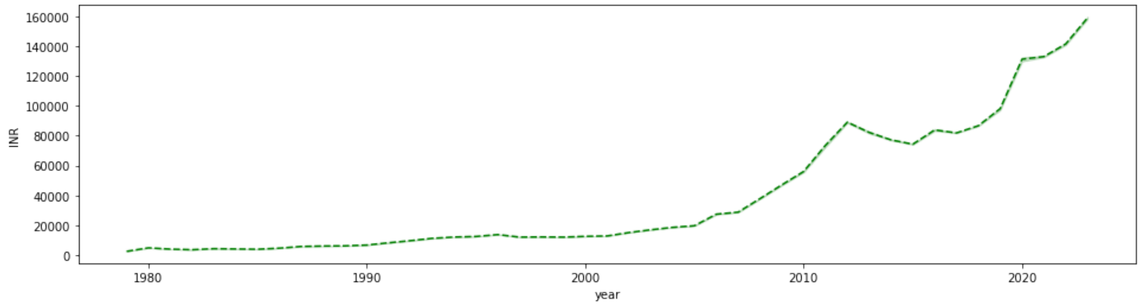
In [16]:
```
df
```

Out[16]:

	INR	year	month	day	week	quarter	weekday
Date							
1979-01-02	1792.9	1979	1	2	1	1	1
1979-01-03	1802.2	1979	1	3	1	1	2
1979-01-04	1811.7	1979	1	4	1	1	3
1979-01-05	1843.6	1979	1	5	1	1	4
1979-01-08	1841.3	1979	1	8	2	1	0
...	...	...	...	...	...	...	...
2023-07-17	159940.3	2023	7	17	29	3	0
2023-07-18	162067.1	2023	7	18	29	3	1
2023-07-19	162141.7	2023	7	19	29	3	2
2023-07-20	162159.4	2023	7	20	29	3	3
2023-07-21	160788.8	2023	7	21	29	3	4

11624 rows × 7 columns

In [17]:
```
plt.subplots(figsize=(16,4))
sns.lineplot(data=df,x='year',y='INR',ls="--",color='green')
plt.show()
```



In [18]:
```
data = df[df['year']>2008]
data
```

Out[18]:

	INR	year	month	day	week	quarter	weekday
Date							
2009-01-01	42152.4	2009	1	1	1	1	3
2009-01-02	42474.5	2009	1	2	1	1	4
2009-01-05	41488.6	2009	1	5	2	1	0
2009-01-06	41343.7	2009	1	6	2	1	1
2009-01-07	41419.5	2009	1	7	2	1	2
...	...	...	...	...	...	...	...
2023-07-17	159940.3	2023	7	17	29	3	0
2023-07-18	162067.1	2023	7	18	29	3	1
2023-07-19	162141.7	2023	7	19	29	3	2
2023-07-20	162159.4	2023	7	20	29	3	3
2023-07-21	160788.8	2023	7	21	29	3	4

3797 rows × 7 columns

In [19]:
```
data.isnull().sum()
```

Out[19]:
```
INR        0
year       0
month      0
day        0
week       0
quarter    0
weekday    0
dtype: int64
```

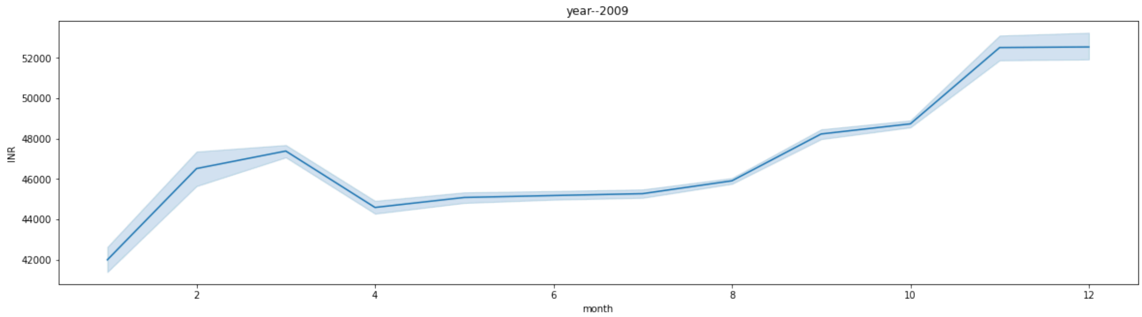
In [20]:
```
def unique(data):
    for i in data.columns:
        print('Unique values in ',i,"-->",data[i].nunique())

unique(data)
```

```
Unique values in  INR --> 3639
Unique values in  year --> 15
Unique values in  month --> 12
Unique values in  day --> 31
Unique values in  week --> 53
Unique values in  quarter --> 4
Unique values in  weekday --> 5
```

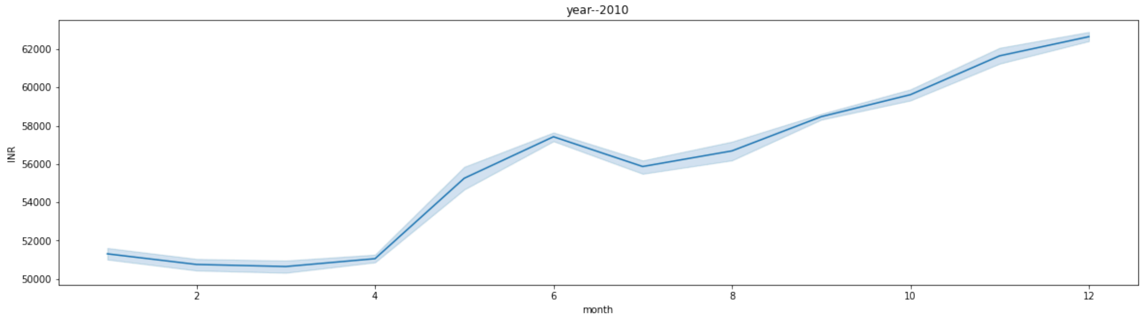
In [21]:

```
# function for each year

years = list(data['year'].value_counts().index.sort_values())
def yearly(x):
    plt.figure(figsize=(10, 6))  # Change size as needed

    newd=data[(data['year']==x)]
    plt.subplots(figsize = (20,5))

    sns.lineplot(data=newd,x=newd.month,ls='-',y=newd['INR'])
    plt.title(f'year--{x}')
    plt.show()





for i in years:
    yearly(i)
```

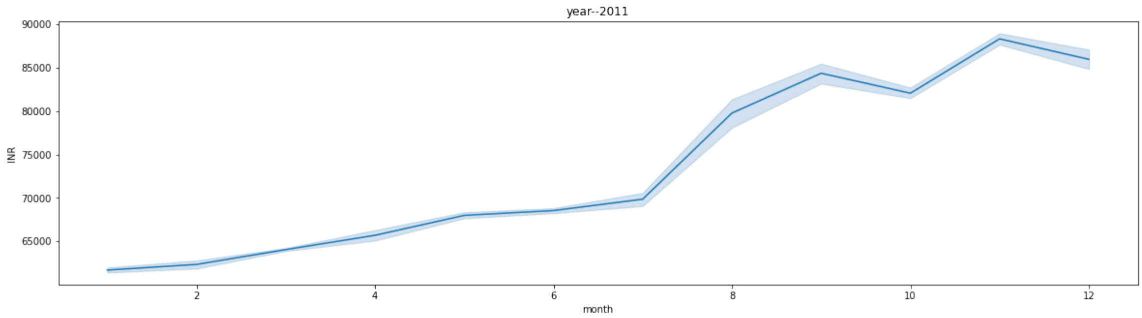
```
<Figure size 720x432 with 0 Axes>
```



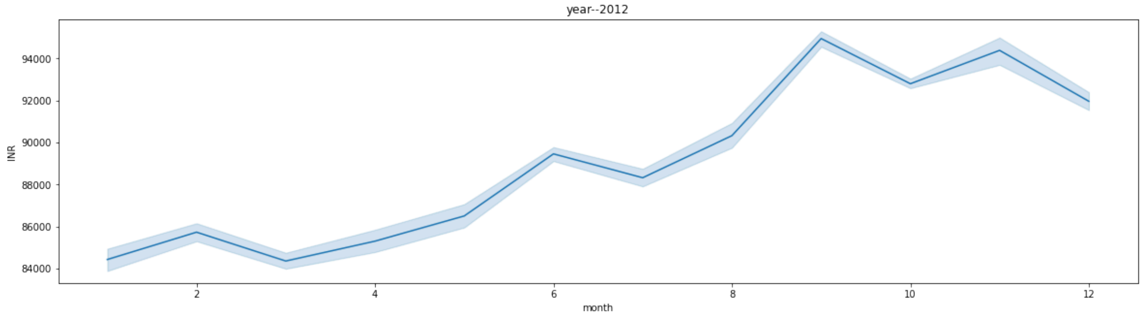

```
<Figure size 720x432 with 0 Axes>
```



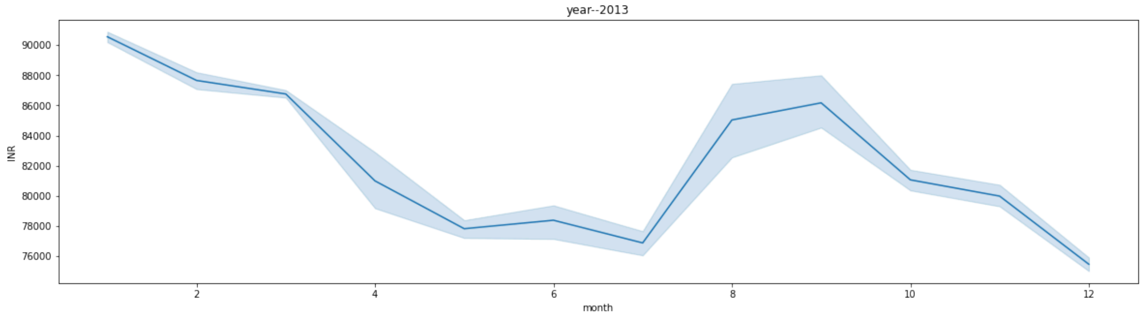

```
<Figure size 720x432 with 0 Axes>
```



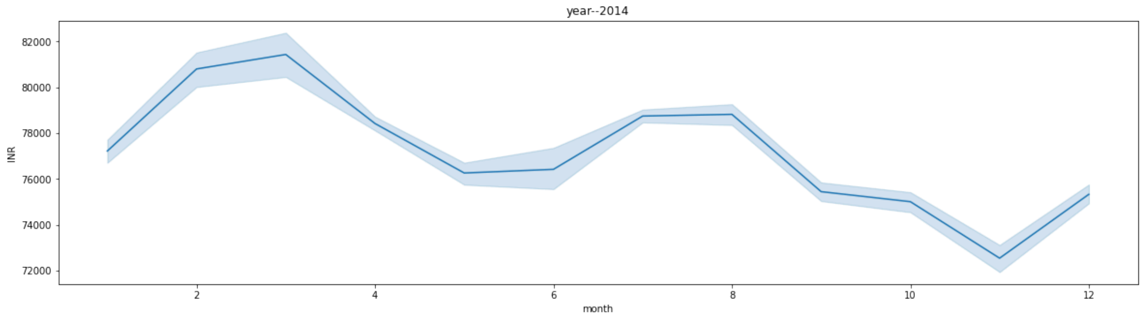

```
<Figure size 720x432 with 0 Axes>
```



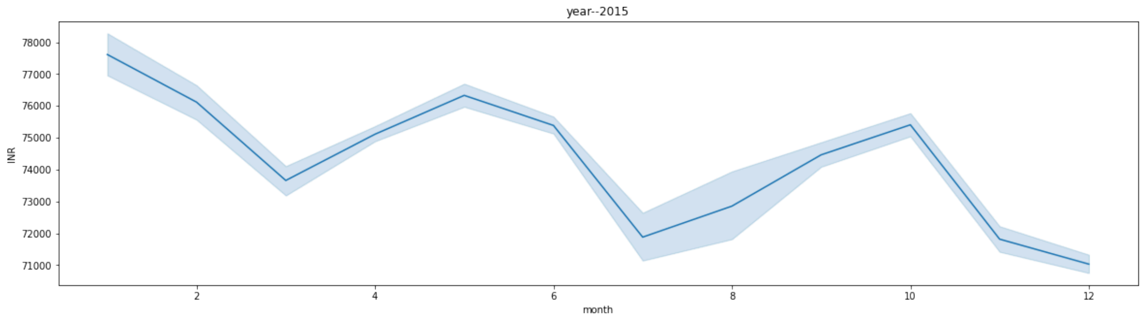
<Figure size 720x432 with 0 Axes>



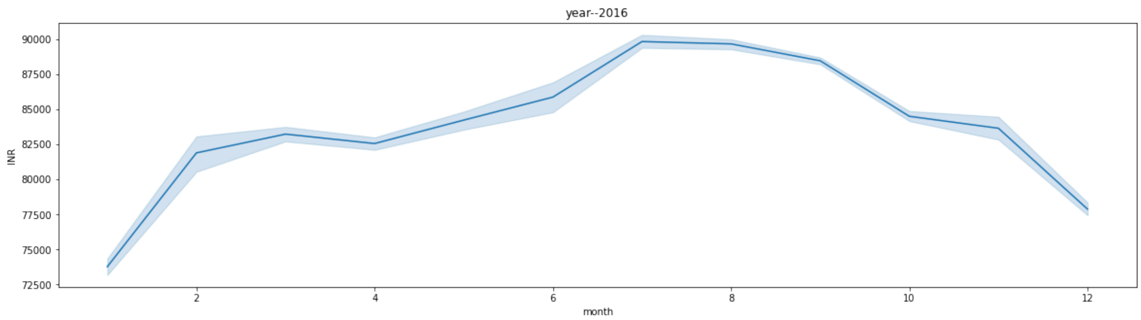
<Figure size 720x432 with 0 Axes>



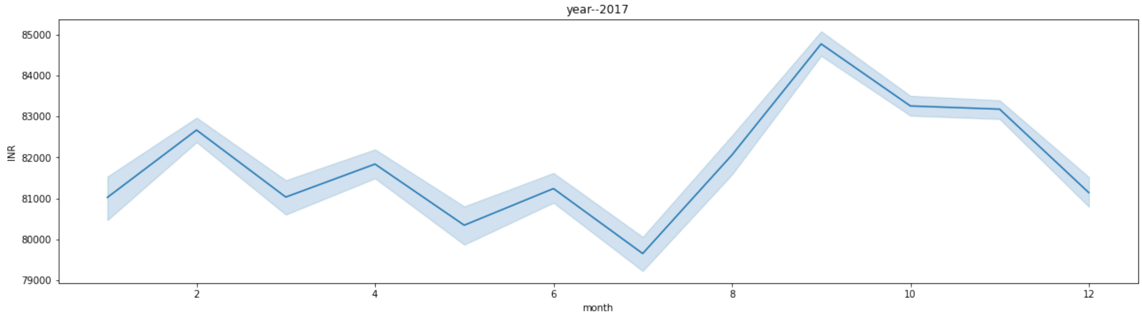
<Figure size 720x432 with 0 Axes>



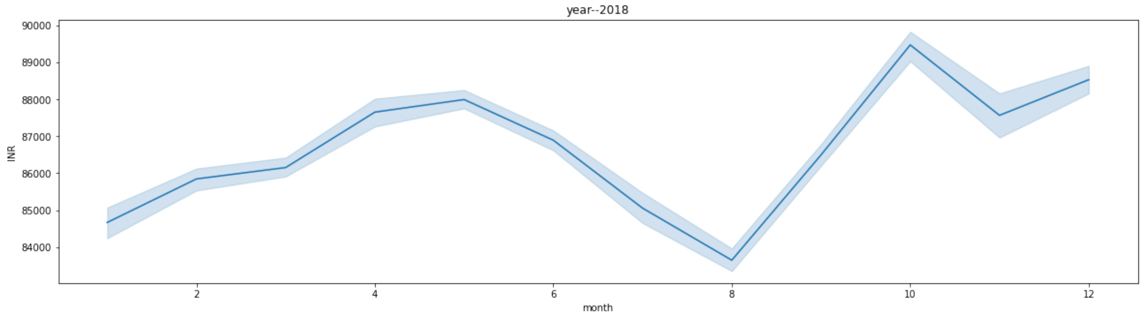
<Figure size 720x432 with 0 Axes>



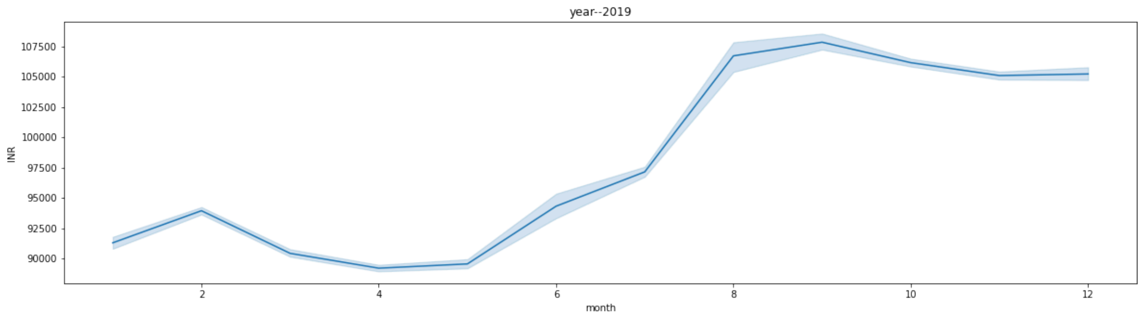
<Figure size 720x432 with 0 Axes>



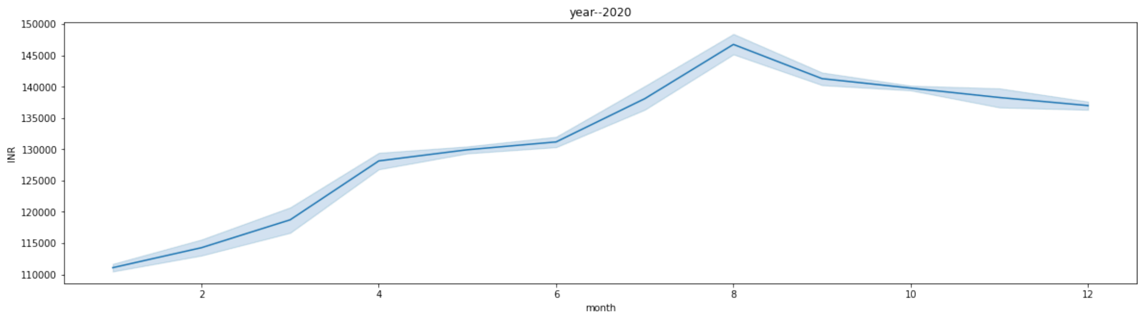
<Figure size 720x432 with 0 Axes>



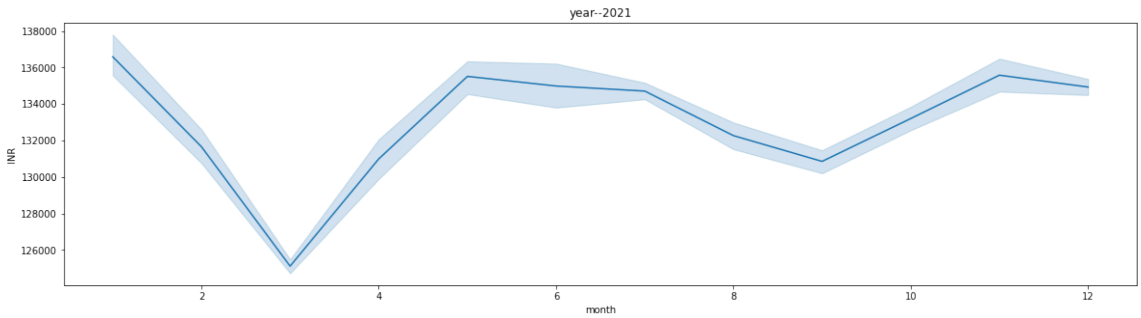
<Figure size 720x432 with 0 Axes>



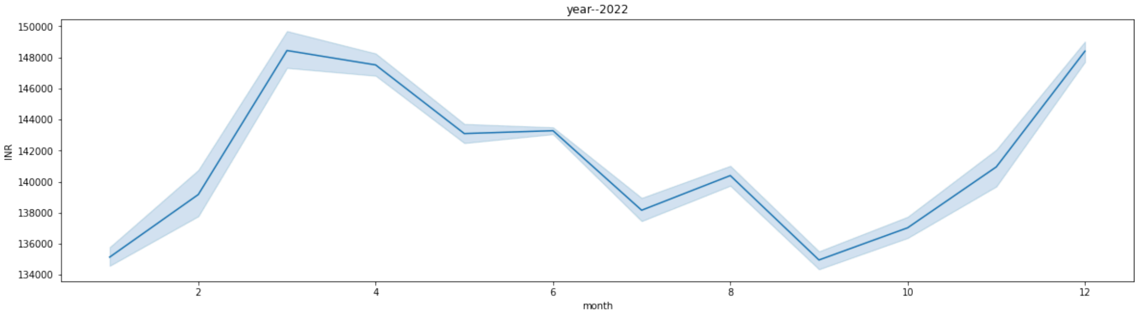
<Figure size 720x432 with 0 Axes>



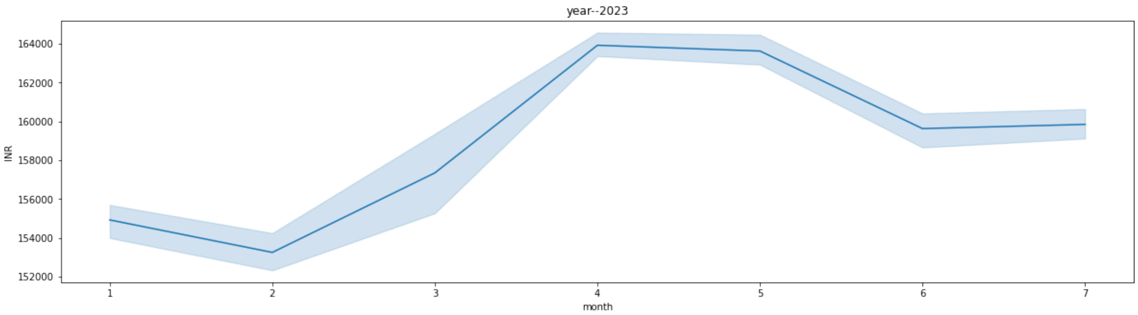
<Figure size 720x432 with 0 Axes>



<Figure size 720x432 with 0 Axes>

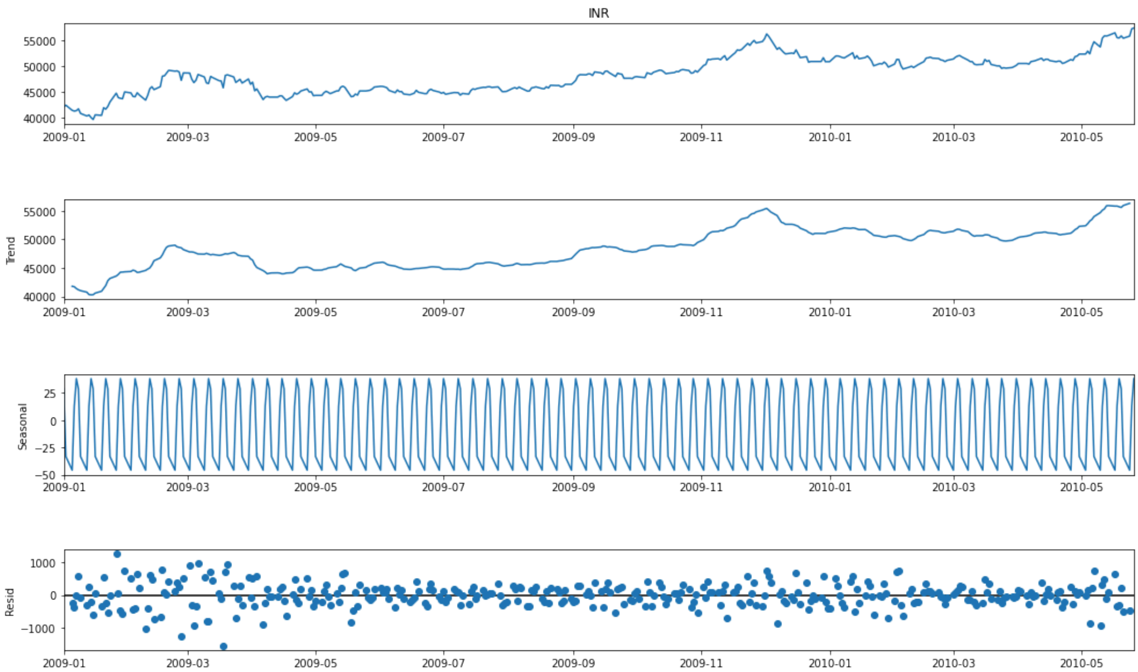


<Figure size 720x432 with 0 Axes>

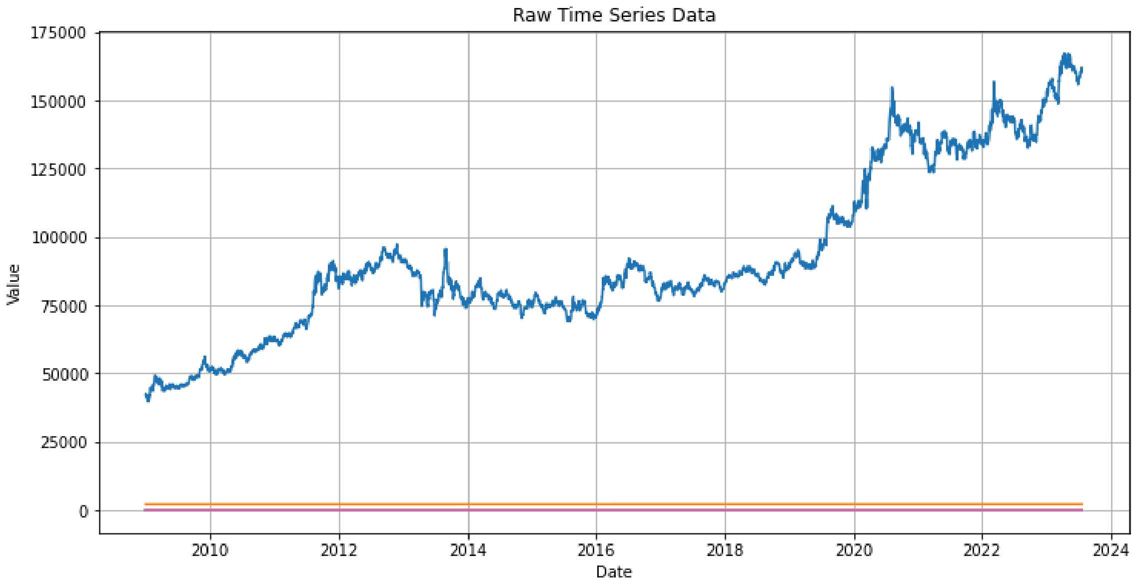


In [22]:
```
# decomposing a time series into trend, seasonal,residual
fig=plt.figure(figsize=((16,6)))
result=seasonal_decompose(data['INR'][:365])
fig=result.plot()
fig.set_size_inches(17,10)
```

<Figure size 1152x432 with 0 Axes>



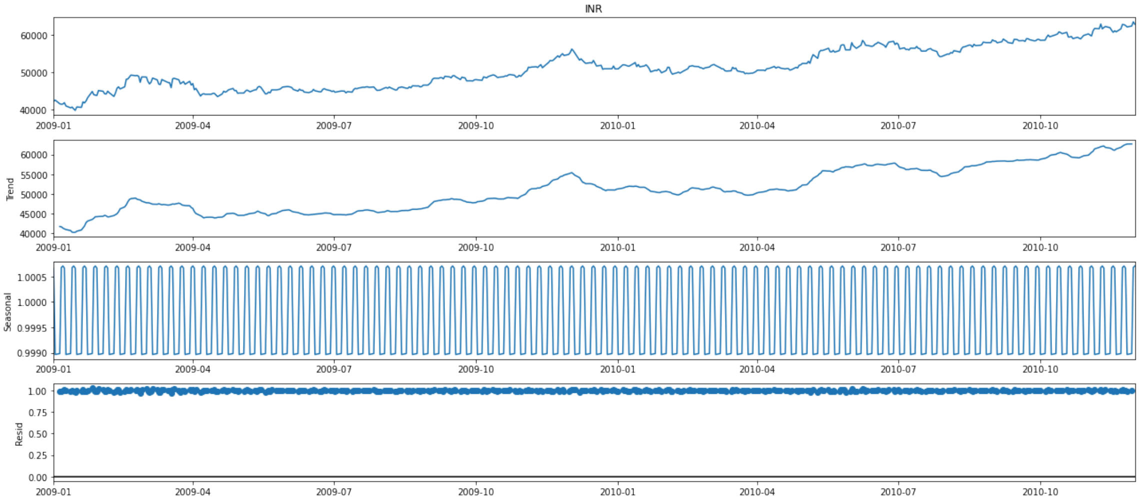
In [23]:
```
plt.figure(figsize=(12, 6))
plt.plot(data)
plt.title('Raw Time Series Data')
plt.xlabel('Date')
plt.ylabel('Value')
plt.grid(True)
plt.show()
```



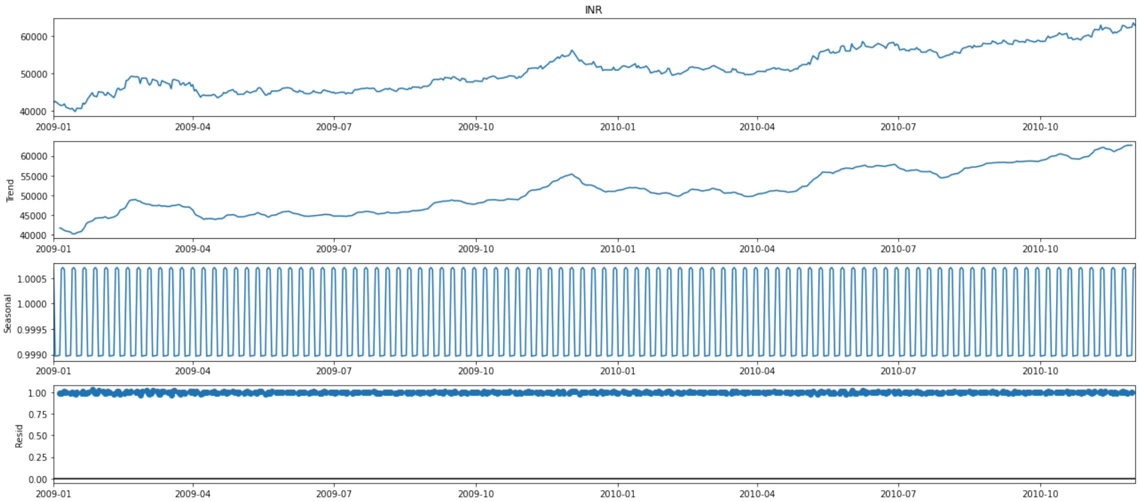
In [24]:

```
rcParams['figure.figsize'] = 18, 8
plt.figure(num=None, figsize=(50, 20), dpi=80, facecolor='w', edgecolor='k'
series = data.INR[:500]
result = seasonal_decompose(series, model='multiplicative')
result.plot()
```

Out[24]:



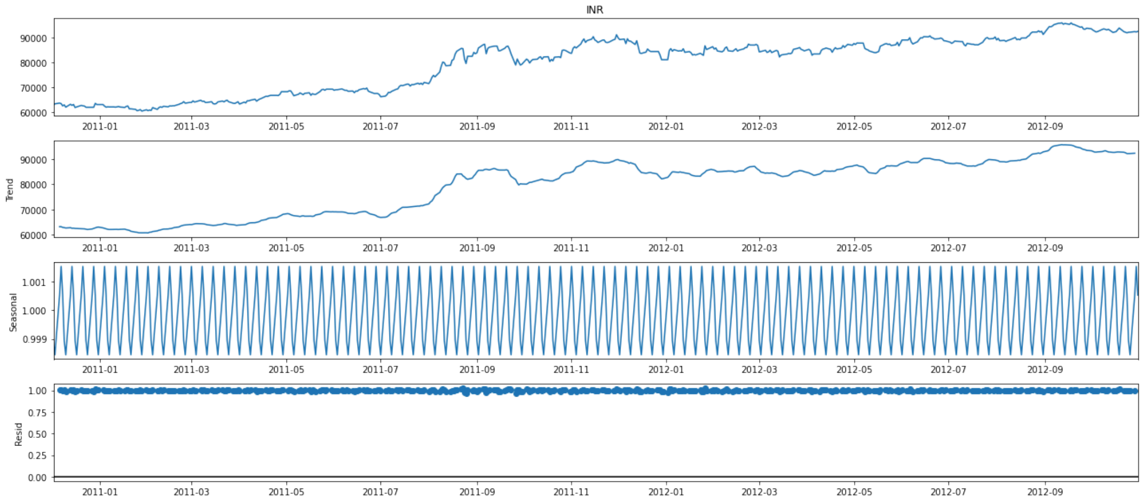

```
<Figure size 4000x1600 with 0 Axes>
```



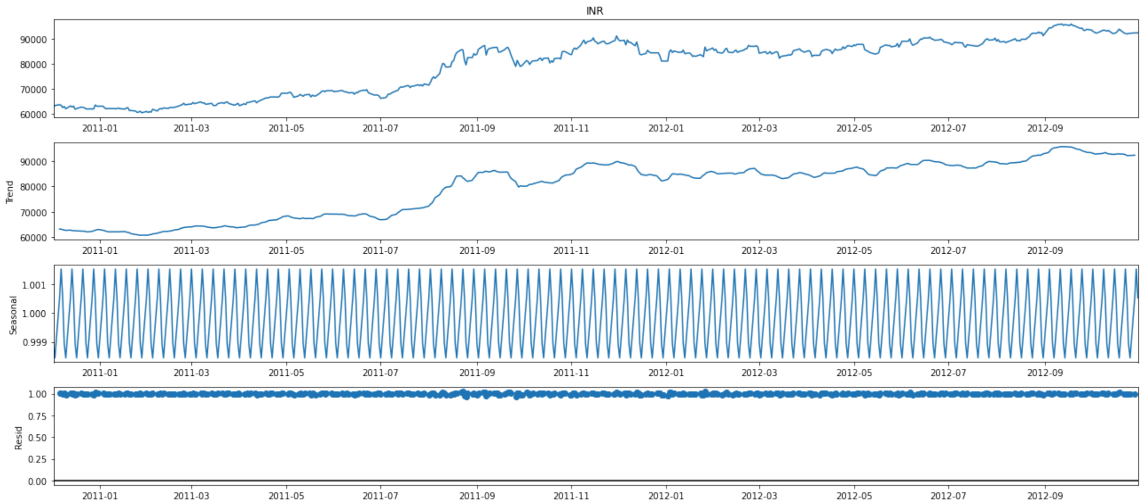
In [25]:

```
#rcParams['figure.figsize'] = 18, 8
plt.figure(num=None, figsize=(50, 20), dpi=80, facecolor='w', edgecolor='k'
series = data.INR[500:1000]
result = seasonal_decompose(series, model='multiplicative')
result.plot()
```

Out[25]:

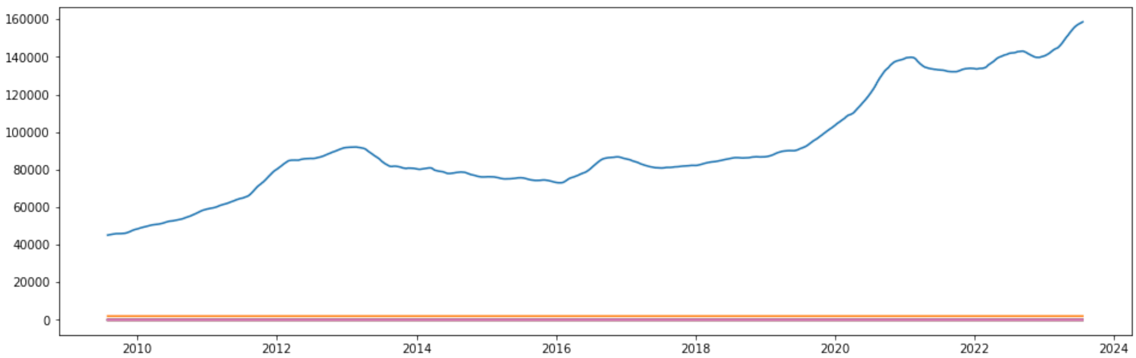



```
<Figure size 4000x1600 with 0 Axes>
```

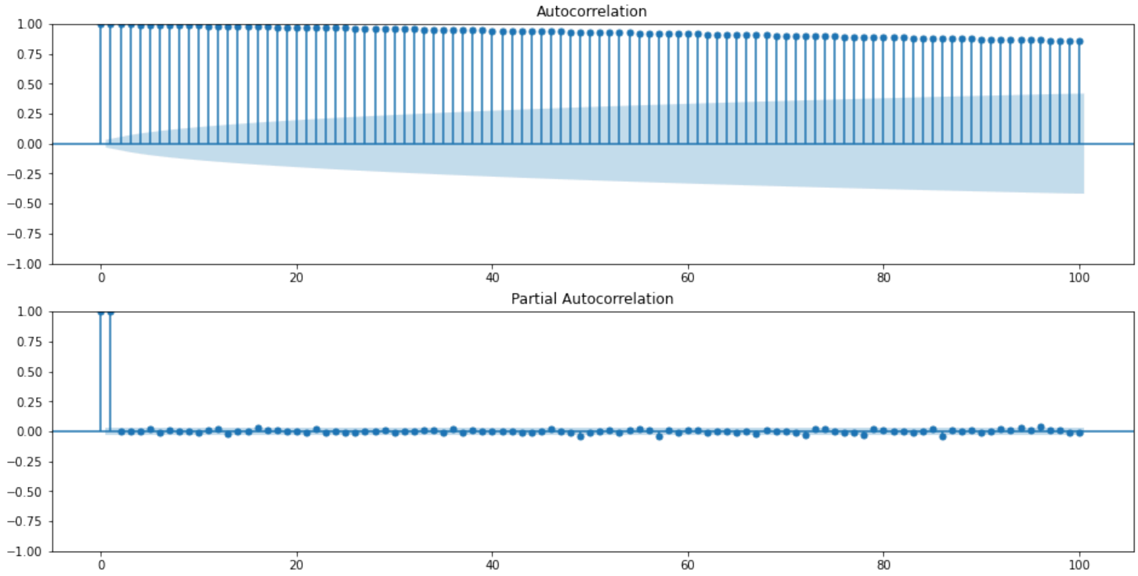


In [26]:

```
# Now we will check trend with automatic decompose, moving averages, std de
moving_avg = data.rolling(window=150).mean()

plt.figure(figsize=(16,5))
plt.plot(moving_avg,label='original data')
plt.show()
```



In [27]:
```
fig=plt.figure(figsize=(16,8))
ax1=fig.add_subplot(211)
fig=plot_acf(data['INR'],lags=100,ax=ax1)
ax2=fig.add_subplot(212)
fig=plot_pacf(data['INR'],lags=100,ax=ax2)
```

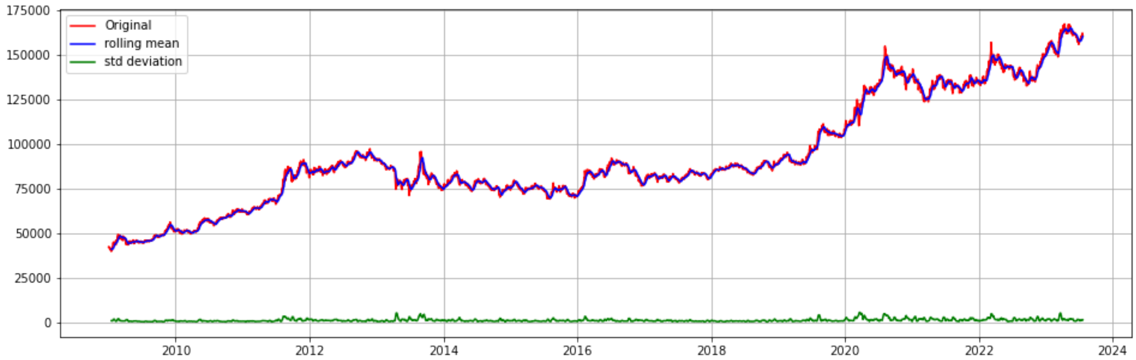


We are having very good correlation at all log values in ACF and in pacf it is showing 1 as perfect correlation. Still we will check with different test and will do stationarity conversion using differencing technique.

- This shows our time series is not stationary

Lets check whether our Rolling mean and Std deviation is constant over the time series data

In [28]:
```
std_deviation = data.INR.rolling(window=12).std()
rolling_mean = data.INR.rolling(window=12).mean()
plt.figure(figsize=(16,5))
plt.plot(data['INR'],label='Original',color='red')
plt.plot(rolling_mean,label='rolling mean',color='blue')
plt.plot(std_deviation,label='std deviation',color='green')
plt.legend()
plt.grid('both')
plt.show()
```



Rolling Mean: If the rolling mean with a window of 12 is almost identical to the original data, it suggests that your data does not have a significant trend over short periods. This is often an indication that your data is relatively stationary, at least in terms of its mean. However, it's essential to look at longer trends in the data to confirm stationarity.

Rolling Standard Deviation: If the rolling standard deviation with windows of 12 or 200 is nearly flat around 0, it suggests that the variability or dispersion of your data does not change significantly over time. This could mean that your data is relatively homoscedastic (constant variance) or that any variability is negligible compared to the scale of the data.

Lets check with ADF for more clearity on stationarity

In [29]:
```
adft = adfuller(data.INR.values)
print("stats value : ",adft[0])
print('P value :',adft[1])
print('critical value :')
for key, value in adft[4].items():
    print('\t%s: %.3f' % (key, value))
```

```
stats value :  -0.22886438049295016
P value : 0.9349924591130789
critical value :
	1%: -3.432
	5%: -2.862
	10%: -2.567
```

Since the p-value is not less than .05, we fail to reject the null hypothesis.

This means the time series is non-stationary. In other words, it has some time-dependent structure and does not have constant variance over time.

We can't reject the Null hypothesis because the p-value is bigger than 0.05. Furthermore, the test statistics exceed the critical values. As a result, the data is not stationary. Differencing is a method of transforming a non-stationary time series into a stationary one. This is an important step in preparing data to be used in an ARIMA model. So, to make the data stationary, we need to take the first-order difference of the data. Which is just another way of saying, subtract today’s close price from yesterday’s close price.

In [30]:
```
# will merge all above graphs here only
```

We will do differencing till our data can not con converted into stationarity.

In [31]:
```
diff_1 = data.diff(1)
diff_1
```

Out[31]:

Date	INR	year	month	day	week	quarter	weekday
2009-01-01	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2009-01-02	322.1	0.0	0.0	1.0	0.0	0.0	1.0
2009-01-05	-985.9	0.0	0.0	3.0	1.0	0.0	-4.0
2009-01-06	-144.9	0.0	0.0	1.0	0.0	0.0	1.0
2009-01-07	75.8	0.0	0.0	1.0	0.0	0.0	1.0
...	...	...	...	...	...	...	...
2023-07-17	-451.2	0.0	0.0	3.0	1.0	0.0	-4.0
2023-07-18	2126.8	0.0	0.0	1.0	0.0	0.0	1.0
2023-07-19	74.6	0.0	0.0	1.0	0.0	0.0	1.0
2023-07-20	17.7	0.0	0.0	1.0	0.0	0.0	1.0
2023-07-21	-1370.6	0.0	0.0	1.0	0.0	0.0	1.0

3797 rows × 7 columns

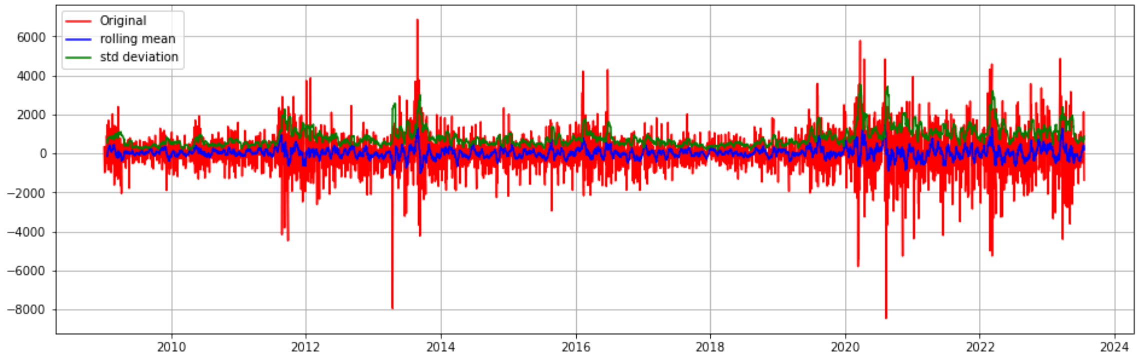
In [32]:
```
diff_1.dropna(inplace=True)
diff_1
```

Out[32]:

Date	INR	year	month	day	week	quarter	weekday
2009-01-02	322.1	0.0	0.0	1.0	0.0	0.0	1.0
2009-01-05	-985.9	0.0	0.0	3.0	1.0	0.0	-4.0
2009-01-06	-144.9	0.0	0.0	1.0	0.0	0.0	1.0
2009-01-07	75.8	0.0	0.0	1.0	0.0	0.0	1.0
2009-01-08	353.9	0.0	0.0	1.0	0.0	0.0	1.0
...	...	...	...	...	...	...	...
2023-07-17	-451.2	0.0	0.0	3.0	1.0	0.0	-4.0
2023-07-18	2126.8	0.0	0.0	1.0	0.0	0.0	1.0
2023-07-19	74.6	0.0	0.0	1.0	0.0	0.0	1.0
2023-07-20	17.7	0.0	0.0	1.0	0.0	0.0	1.0
2023-07-21	-1370.6	0.0	0.0	1.0	0.0	0.0	1.0

3796 rows × 7 columns

In [33]:
```
std_deviation = diff_1.INR.rolling(window=12).std()
rolling_mean = diff_1.INR.rolling(window=12).mean()
plt.figure(figsize=(16,5))
plt.plot(diff_1['INR'],label='Original',color='red')
plt.plot(rolling_mean,label='rolling mean',color='blue')
plt.plot(std_deviation,label='std deviation',color='green')
plt.legend()
plt.grid('both')
plt.show()
```

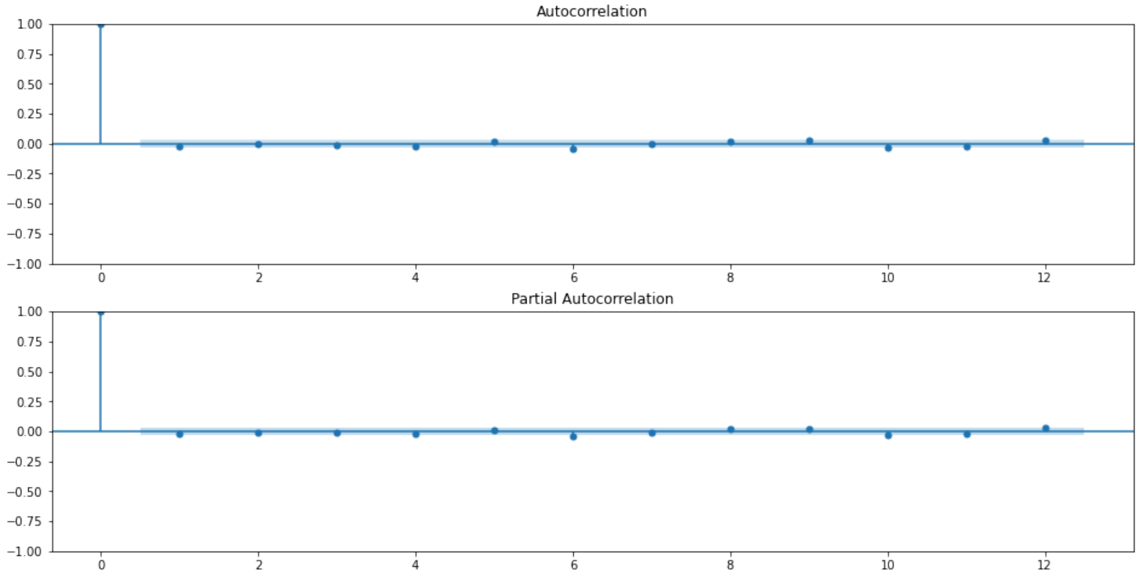


In [34]:
```
adft = adfuller(diff_1.INR.values)
print("stats value : ",adft[0])
print('P value :',adft[1])
print('critical value :')
for key, value in adft[4].items():
    print('\t%s: %.3f' % (key, value))
```

```
stats value :  -16.199689198613378
P value : 4.0895361389833677e-29
critical value :
	1%: -3.432
	5%: -2.862
	10%: -2.567
```

- Now we can reject the null hypothesis- our time series data is stationary now.
- We can go and check with ACF and PACF value now.

In [35]:
```
fig=plt.figure(figsize=(16,8))
ax1=fig.add_subplot(211)
fig=plot_acf(diff_1['INR'],lags=12,ax=ax1)
ax2=fig.add_subplot(212)
fig=plot_pacf(diff_1['INR'],lags=12,ax=ax2)
```



In summary, zero ACF and PACF values at lag 12 after differencing are common and generally indicate that the seasonal pattern has been effectively removed from the data. As long as the resulting differenced series is stationary and suitable for modeling, there is no need to be worried about this observation.

Now we will proceed with Model building and time series forecasting

In [36]:
```
diff_1.drop(['month','year','day','week','quarter','weekday'],axis=1,inplac
```

In [37]:
```
diff_1.isnull().sum()
```

Out[37]:
```
INR    0
dtype: int64
```

In [38]:
```
trainsize=0.7
train,test = train_test_split(diff_1, train_size=trainsize,shuffle=False)
print(train)
print(test)
```

```
                INR
Date
2009-01-02  322.1
2009-01-05 -985.9
2009-01-06 -144.9
2009-01-07   75.8
2009-01-08  353.9
...           ...
2019-03-05 -525.7
2019-03-06 -501.9
2019-03-07   -3.2
2019-03-08  634.3
2019-03-11 -545.6

[2657 rows x 1 columns]
                 INR
Date
2019-03-12    68.8
2019-03-13   633.0
2019-03-14 -1071.9
2019-03-15    38.9
2019-03-18  -250.6
...            ...
2023-07-17  -451.2
2023-07-18  2126.8
2023-07-19    74.6
2023-07-20    17.7
2023-07-21 -1370.6

[1139 rows x 1 columns]
```

In [39]:
```
train_data = train['INR'].to_numpy() # Select 'INR' column
print(train_data)

test_data = test['INR'].to_numpy() # Select 'INR' column
print(test_data)
```

```
[ 322.1 -985.9 -144.9 ...   -3.2  634.3 -545.6]
[   68.8   633.  -1071.9 ...    74.6    17.7 -1370.6]
```

In [ ]:

In [40]:

```
# Lets train it on arima model
def arima_model(train,test,arima_order):
    history=[x for x in train]
    predictions=list()
    for t in range(len(test_data)):
        model = ARIMA(history,order=arima_order)
        model_fit=model.fit()
        yhat=model_fit.forecast()[0]
        predictions.append(yhat)
        history.append(test_data[t])
    # calculating rmse
    print(predictions)
    rmse = np.sqrt(mean_squared_error(test_data,predictions))
    plt.plot(test_data, color='green',label='Original')
    plt.plot(predictions, color='black', label='AR predicted')
    plt.legend()
    print(rmse)
```

In [41]:

```
arima_order=(0,1,0)
arima_model(train_data,test_data,arima_order)
#print(f'RMSE : {rmse}')
```

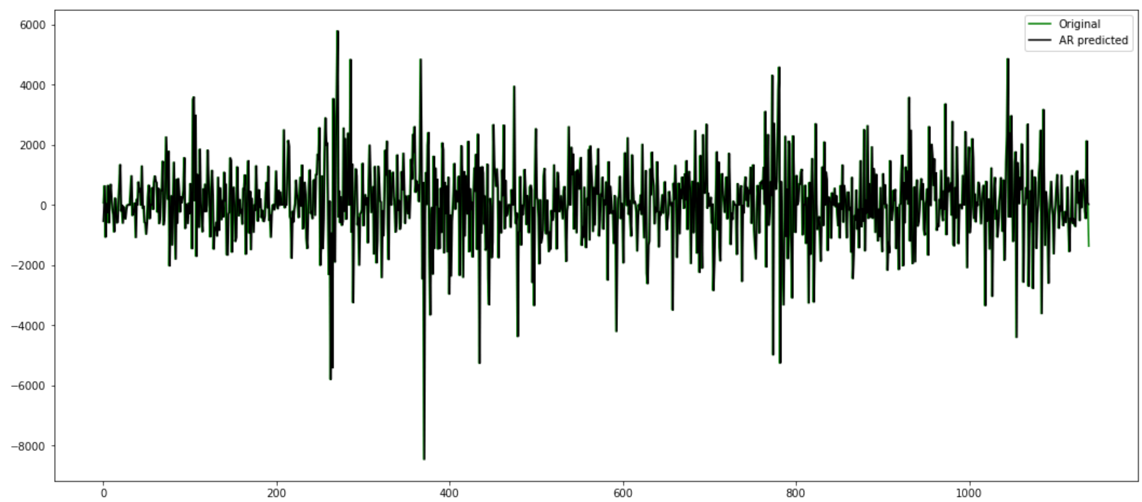
[-545.5999999999912, 68.80000000000291, 633.0, -1071.9000000000087, 38.900
00000000896, -250.60000000000582, 647.0, -596.5, 397.6000000000058, 681.5,
178.5, -139.40000000000876, -331.6999999999971, -897.1999999999971, 225.59
999999999127, -277.0, -597.5999999999913, -580.6999999999971, 134.1999999
99971, 653.3999999999942, 1336.6000000000058, -203.0, -11.10000000000582,
-606.5, -60.19999999999709, -380.1999999999972, -466.6000000000058, -167.3
9999999999418, -8.499999999999972, 0.0, 0.0, -8.999999999999998, 464.09999
99999913, 967.1000000000058, -349.8999999999942, -278.8000000000029, -121.
19999999999712, 63.99999999999997, -1084.7000000000116, 184.6000000000058
2, 2.842170943040401e-14, 760.1999999999972, 424.90000000000873, 537.89999
99999942, -8.39999999999418, 1290.899999999994, -113.30000000000291, -42.
5, -581.8000000000029, -596.6999999999971, -971.8000000000029, -568.799999
9999884, 259.79999999998836, 655.2000000000116, -458.5, 0.0, -11.5, 574.09
99999999913, -147.5, 750.6000000000058, 958.8999999999942, 788.30000000000
29, 740.6999999999971, -237.80000000000302, 553.4000000000087, -621.300000
0000029, -405.1000000000058, 497.600000000006, 482.0, 1450.0, -664.0, -41
7.0, 406.1999999999971, 2257.400000000009, 1251.0999999999913, 189.0, 177
8.800000000003, -2028.199999999997, -317.69999999999663, 299.899999999994
2, -1332.5000000000002, 39.80000000000291, 1415.699999999997, -235.4999999
9999977, -1802.3999999999944, 841.6999999999971, -608.4999999999999, 872.6
999999999971, 448.80000000000285, -235.8000000000029, 273.30000000000285,
72.30000000000288, 229.39999999999418, 497.99999999999994, 1566.5, -783.10
00000000058, 30.200000000011755, -19.49999999999999, -619.0, 53.5999999999
9127, -288.8000000000029, 706.8000000000029, 78.69999999999709, -1454.5000
000000002, 3504.4000000000087, 3585.5, 137.8000000000029, 2980.5, -1709.0,
1015.2999999999886, 829.5, -744.0999999999913, 1848.8999999999946, 85.4000
0000000896, -447.5, -894.8000000000029, 532.6999999999972, -225.1999999999
971, 688.6999999999971, -224.8000000000028, 2.842170943040401e-14, 1818.10
0000000006, 683.5999999999913, 39.100000000005934, -1061.8999999999942, 55
2.6999999999971, 1145.800000000003, 303.1999999999971, -1470.0, -758.10000
00000058, -840.3999999999942, -577.8000000000029, -1033.0, 916.8999999999
942, -844.9999999999999, 569.4000000000085, 160.30000000000297, -491.100000
0000056, 144.0, -401.3000000000029, 1084.5000000000002, 130.3000000000029,
689.3999999999942, -1653.7999999999881, -1668.9000000000087, -301.30000000
00027, -238.39999999999418, 1559.6000000000058, 1475.2999999999884, -1570.
399999999994, 479.1000000000056, 586.5999999999913, -86.5, -1214.800000000
0027, -1081.199999999997, 1262.699999999997, 241.10000000000582, -362.3999
999999943, 177.79999999998836, -272.0, -152.6999999999971, -638.3000000000
029, 541.7000000000116, 518.8999999999942, 994.0, -1639.1999999999969, -17
9.60000000000582, 544.8000000000029, 1467.199999999997, -934.199999999996
9, 450.19999999999686, -1486.6999999999969, 216.0, -320.19999999999703, -9
09.900000000009, 4.6000000000059345, -173.8999999999942, 1296.599999999991
3, 85.30000000000268, -539.0, 502.8000000000029, -81.19999999999703, 197.3
9999999999415, -423.3000000000028, -178.30000000000294, -530.699999999997
1, -552.8999999999942, -366.19999999999715, 413.29999999998836, 748.100000
0000058, -98.50000000000011, 1276.1000000000058, -514.7000000000116, -329.
6999999999971, -1074.0999999999913, -268.999999999999, -6.700000000011641
5, 11.60000000000582, -164.3000000000029, 82.0, 1125.6999999999973, -19.89
9999999993952, -125.60000000000582, 480.50000000000006, -18.899999999941
8, 426.99999999999994, 5.684341886080802e-14, 0.0, 0.0, 2493.499999999999
5, -94.80000000000337, -1.4210854715202004e-14, 0.0, 984.199999999997, 213
9.4000000000087, 1949.199999999997, -393.3000000000029, -476.399999999994
2, -1772.6000000000058, -210.39999999999463, -569.3000000000029, -150.3000
0000000297, 44.00000000000003, 818.6999999999971, 358.20000000001164, 136.
49999999999997, -421.6000000000057, 261.5, 914.1999999999971, -41.69999999
9996976, 1321.0, -795.3000000000031, 194.4999999999999, 749.6000000000058,
358.0, -1109.400000000009, -1447.2999999999884, -235.60000000000582, 707.9
999999999999, 1162.6000000000058, -277.70000000001164, -225.5, -456.699999
9999971, 796.0000000000002, 838.5, -346.3999999999407, 1011.19999999997
1, 1016.0, 1682.5, 1517.5, 2564.899999999994, -2009.199999999997, -1414.80
0000000003, 960.9999999999998, -1449.5, -445.5999999999914, 1759.699999999

997, 2891.300000000003, 1983.800000000003, 2061.399999999994, -312.1000000
0000537, -2311.199999999997, 122.19999999999709, -5801.699999999996, -937.
3000000000029, -5415.699999999996, 3532.9000000000096, -1334.500000000000
5, -1897.9000000000087, 2314.300000000003, 3555.600000000006, 5785.0, -39
6.90000000000873, 459.69999999999703, -590.7999999999884, 269.399999999994
2, -678.5, -549.0, 2552.699999999996, -216.1999999999971, 2203.39999999999
4, -475.3999999999937, 378.10000000000565, 2384.2999999999884, 0.0, 0.0, 4
836.600000000006, -902.5000000000009, 1354.8999999999944, -3250.199999999
97, -471.0, 451.80000000000297, 1195.199999999997, 968.4000000000087, -66
3.7000000000114, -430.2999999999884, -2004.2000000000114, -286.29999999998
836, -268.20000000001164, -151.0, 1380.300000000003, -689.9999999999998, 2
7.699999999997203, 335.90000000000873, 0.0, 231.80000000000288, -1259.0, 8
12.8999999999944, 1984.699999999997, 833.0999999999913, -253.199999999825
4, 153.5, 607.0, -1631.800000000003, 1273.0999999999913, -2.27373675443232
06e-13, -1927.1999999999973, -1189.3999999999942, 1279.6999999999973, 758.
6999999999971, 121.99999999999989, 103.69999999999709, -2415.799999999988
4, -190.2000000000112, -1025.3999999999942, 272.39999999999395, 1814.20000
00000116, 993.0000000000001, 2117.0, -686.0000000000009, -1817.10000000000
58, 1148.8999999999944, 299.39999999999407, -6.799999999988302, 915.299999
9999884, 1087.4000000000233, 286.0999999999767, -53.7999999998836, -904.1
000000000057, -612.3999999999942, 1660.0, -181.10000000000605, 143.2000000
0001164, -809.8999999999944, -588.8000000000175, 1104.8999999999942, 500.6
0000000000593, 1717.0, 527.8999999999942, -610.8999999999942, 296.0, 10.80
0000000017462, -291.6000000000059, 301.19999999998254, -449.0, 346.7000000
0001164, 1514.2000000000114, 856.1999999999825, 2343.0, 1746.399999999994
2, 2599.600000000006, 419.8999999999942, 803.1000000000058, 477.0, 671.800
0000000175, 111.39999999999418, 1136.8999999999944, 4841.0, 1567.899999999
9937, -2461.100000000006, 739.0, -8461.599999999977, -266.70000000001164,
1072.6000000000058, 13.100000000005593, 1988.0, 2405.0, -1651.100000000005
4, -3658.8000000000175, -325.59999999997626, -83.8000000001743, -2300.899
9999999937, 1614.5, -1447.0, 1223.7000000000116, 0.0, 848.3999999999943, -
1468.0, -115.80000000001746, -1448.6999999999825, 549.8999999999942, -915.
5000000000001, 2058.899999999994, 1823.5, -1593.7999999999888, 745.7999999
999884, -278.79999999998824, 623.0, -1559.0, 916.6000000000058, -2965.8000
00000017, -337.8999999999942, -2363.7000000000116, -547.5999999999767, -12
9.50000000000006, 474.0999999999767, 1374.8000000000175, -142.100000000005
82, 468.29999999998836, 210.10000000000582, 98.20000000001166, 951.2999999
999885, -2341.399999999994, 200.2999999999888, 1964.0, 885.8000000000175,
-2406.200000000012, 1107.3999999999946, -1029.7999999999881, 969.399999999
994, -114.89999999999418, -301.29999999998836, 2110.2999999999884, -1556.9
999999999995, 537.7000000000116, -153.30000000001723, 87.50000000000003, -
1731.6999999999825, 363.5, 1226.1999999999825, 450.20000000001176, 1681.10
00000000058, -1477.0000000000005, 2351.9999999999995, 191.299999999879, -
5266.899999999994, 1250.1000000000058, -938.2000000000119, 1267.2000000000
116, 1061.2999999999884, -634.3999999999942, 296.19999999998265, -1505.099
9999999765, -1422.6000000000058, 1350.7999999999884, -2435.5000000000005,
-3319.0, 213.800000000017, -59.70000000001161, -1758.100000000006, -1239.3
999999999942, 2665.899999999995, 1281.3999999999944, 945.7000000000116, 61
8.8999999999942, 1192.7000000000116, 546.6999999999825, -1759.0, 116.20000
000001187, -200.0, -929.2000000000116, 1158.9999999999998, 309.5, 2651.900
0000000233, -790.0, 811.6999999999822, -371.19999999998254, -315.400000000
0233, 0.0, 0.0, 0.0, -745.1999999999824, 505.19999999998254, 5.68434188608
0802e-14, 0.0, 3941.100000000006, -115.69999999998254, -622.5, -271.100000
0000058, -4376.399999999994, -919.7000000000116, -856.5, 939.200000000011
9, -1333.5000000000005, -8.700000000011642, -324.79999999998836, 53.5, 117
4.0999999999765, 492.0000000000001, -648.1999999999823, 306.6999999999824,
-18.299999999883, -924.5, 587.7000000000114, 650.8999999999942, 501.19999
999998254, -2581.2999999999884, -85.70000000001164, -3345.79999999989, 94
8.600000000058, 2529.100000000006, 242.69999999998254, 164.8999999999941
8, -419.7999999999883, -1958.1000000000058, 169.5, -1259.8999999999942, -1
110.800000000003, -834.1999999999971, 950.1000000000058, 1209.0, -589.1000

00000006, -964.1000000000058, 158.10000000000593, -779.8000000000029, -154
8.0, -897.0999999999913, -1469.6000000000058, -184.3999999999418, -514.5,
-420.40000000000873, 1327.4000000000087, -213.99999999999977, 541.29999999
99884, -1463.2999999999884, 1081.5, 765.3999999999943, -225.6999999999972,
-149.6999999999971, 300.0, -77.60000000000576, -461.8999999999942, 470.599
9999999913, 606.3000000000029, -494.8999999999407, -1877.1000000000058, -
202.99999999999977, 42.30000000000288, 2597.69999999997, 0.0, 0.0, 1914.3
00000000003, 1003.6999999999972, 1688.5, -799.1000000000056, -127.69999999
99972, 1111.300000000003, -925.8000000000029, 1163.5999999999913, 975.3000
000000175, 523.6999999999823, 1223.3000000000175, 1574.1000000000058, -134
2.800000000018, -729.8999999999943, -1073.2999999999884, 585.699999999982
8, -1084.8999999999944, -1419.300000000003, 224.5, 0.0, 1839.800000000003
1, -1166.2999999999881, 1956.599999999976, 1151.4000000000233, 531.2999999
999885, -881.4999999999998, 580.6000000000057, -666.3999999999941, 556.000
0000000001, 1299.8999999999942, 355.4999999999998, 1863.7999999999884, -10
32.3999999999942, -506.4999999999998, 364.2999999999884, 288.0, 916.600000
0000058, -803.7999999999885, 163.3999999999943, 0.0, 761.2999999999884, 48
3.6000000000059, -2499.899999999994, 1430.3999999999942, -289.299999999988
36, 615.6999999999825, 170.40000000002317, -285.5, -201.5, -1281.000000000
0002, 303.69999999998254, -411.19999999998254, -4207.8000000000175, -756.6
999999999821, 216.4999999999999, 294.5, 944.6999999999825, -458.7999999999
8836, 52.79999999998836, -238.39999999999415, -1927.5, 773.5, 1729.8999999
999942, 271.30000000001746, -22.5, 2229.699999999983, -313.1000000000058,
150.90000000002328, -531.4000000000233, -1036.0, 1660.3000000000175, 665.6
999999999827, 50.5, 122.20000000001164, 28.399999999994193, -229.199999999
98257, -1530.7000000000116, -376.0, 8.399999999994236, -161.199999999825
4, 327.3999999999942, -316.6000000000058, 2013.2999999999884, -97.59999999
997672, -1095.7000000000116, -81.20000000001153, 1138.0, -2271.69999999998
25, -2619.2000000000116, -1335.5, -1208.5, 1193.4000000000092, 333.6999999
999971, 1749.2000000000116, 927.0, 635.3999999999942, -627.5, -81.70000000
001164, -234.5, 1430.2000000000116, 477.6000000000058, -1396.600000000005
8, -227.80000000001746, -348.29999999998836, 0.0, 335.29999999998836, -27
5.09999999997666, 190.59999999997672, 767.7000000000118, 25.7999999999883
6, -827.4999999999999, -507.0999999999767, -295.30000000001746, 442.700000
00001164, 91.19999999998254, -24.599999999976703, 76.5999999999767, -3494.
799999999988, 719.1000000000058, 196.1999999999971, 1311.5, 87.69999999999
709, -1746.1999999999969, -262.5000000000002, 721.5999999999913, -959.8999
999999944, 508.5, 217.1999999999703, 923.9000000000087, 451.1999999999970
3, -113.80000000000291, 1029.100000000006, 163.5, 1467.1999999999825, -81
6.5999999999765, 891.3999999999941, 1110.7999999999884, 433.3000000000174
6, -1948.1000000000054, 148.60000000000582, 785.5, -925.7000000000116, 14
0.49999999999977, 2472.2000000000116, -163.60000000000582, -1602.200000000
0116, 732.5000000000005, 310.3999999999941, -2244.0, 1638.4000000000228, -
698.4000000000235, -2103.1999999999825, 2332.100000000006, 424.69999999998
32, 387.6000000000058, 821.8999999999942, 2678.7000000000116, -107.6000000
0000582, 389.50000000000006, -40.0, 0.8999999999941863, 251.5000000000000
6, -600.2999999999884, 25.8999999999418, -2845.899999999994, -1930.200000
0000114, -495.6000000000058, 446.9999999999999, 1761.9000000000233, -824.9
000000000233, 1423.1000000000058, -1406.6999999999825, -1859.200000000011
6, 678.2999999999884, 1036.9000000000233, 361.29999999998836, 126.20000000
001161, -246.00000000000003, 402.19999999998254, 926.3000000000175, -465.1
0000000000605, 64.10000000000582, 1709.599999999977, 469.5000000000002, -1
314.7999999999884, -275.5000000000002, -481.7000000000116, 121.20000000001
17, 0.0, 0.0, 0.0, -1645.3999999999942, 700.600000000006, 0.0, 0.0, 538.29
99999999884, 633.8999999999942, -2543.299999999989, -15.3999999999418, -2
68.8000000000174, 601.3000000000172, 926.0999999999767, 151.0, 578.8000000
000176, -59.30000000017576, 493.90000000002317, 382.0999999999767, 1248.1
000000000058, -608.399999999994, 236.29999999998824, 1328.9000000000233, -
847.600000000006, -1320.3999999999942, -1795.6000000000058, -223.100000000
00582, 644.6000000000058, 311.89999999999424, -932.7000000000119, 734.9000
000000235, 712.2999999999884, 796.1000000000058, 531.0, 1241.799999999988

4, 19.20000000001164, 3105.399999999994, -2063.600000000006, 717.700000000
0121, 2336.2999999999884, -674.2999999999884, -240.5, 771.5, 314.100000000
00576, 4306.399999999995, -4986.800000000018, 2710.4000000000233, 946.6999
999999825, 519.1000000000058, 795.1000000000058, 2074.5, 3800.799999999988
4, 4579.5, -5258.799999999989, 779.6999999999825, -829.2999999999882, -213
6.7000000000116, -3325.0, -418.8999999999937, 2279.3000000000175, -1051.0,
521.599999999977, -1786.5999999999765, 2114.5000000000005, 2034.0999999999
767, -861.8999999999942, -1632.2999999999884, -3093.899999999994, 2291.599
9999999767, 813.8000000000175, -898.6000000000057, -911.7000000000116, 97
5.5000000000002, 4.200000000116415, 113.29999999998836, 693.600000000005
8, 688.5, 1090.5, 1180.5, -996.2999999999886, 1.1368683772161603e-13, 0.0,
652.8999999999941, -1281.6000000000058, -416.7999999999885, 242.6999999999
8254, -3260.399999999994, 746.2000000000116, -1643.5, 351.2999999999886, 1
538.3999999999944, 0.0, -3231.0, -956.8999999999937, 2697.0, 45.3999999999
9418, -816.6999999999825, -490.3999999999942, -392.20000000001164, -1024.7
999999999884, -1868.8000000000175, -101.0, 1261.6000000000058, -684.600000
0000058, 2091.3000000000175, -4.300000000017462, 1091.3000000000175, 847.2
999999999885, -1512.9999999999995, 259.39999999999395, 245.10000000000582,
76.89999999999418, -1106.5000000000002, 399.10000000000605, 5.684341886080
802e-14, 0.0, 551.3999999999942, -60.39999999999429, 301.3999999999942, -3
19.3999999999942, -685.3999999999942, 46.79999999998836, -1098.89999999999
42, 639.8999999999944, -140.0, 1324.6000000000058, -571.2999999999886, 64
4.6999999999827, 330.00000000000006, -146.60000000000582, -1094.0999999999
767, 420.79999999998836, 422.70000000001164, -175.60000000000576, -82.3000
0000001748, -1568.5, 911.3000000000177, -2446.8000000000175, -1660.5, -35
2.69999999998254, -510.1000000000058, 491.7999999999882, -806.599999999976
5, -146.70000000001164, -1425.3999999999942, 177.10000000000582, 1466.5999
999999767, -724.2999999999881, -125.0, -538.0, 2499.100000000006, -1526.60
00000000054, 163.2000000000112, -268.9000000000233, 2632.9000000000233, -4
47.2000000000112, 706.8999999999942, -233.60000000000582, -414.5, 1930.300
0000000175, -446.60000000000537, 1208.7999999999884, 868.0, -913.0, 1016.7
000000000119, -285.29999999998836, -1197.4000000000233, -139.2999999999883
6, -1054.7000000000116, 575.3000000000175, -921.7000000000118, -1548.60000
00000058, 1035.3000000000175, 25.39999999999418, 667.3999999999942, -101.7
9999999998824, -1.4210854715202004e-14, -2171.399999999994, -1234.80000000
00175, -1589.1000000000058, 1466.9000000000233, 22.5, -510.6000000000058,
-189.89999999999418, 483.7999999999885, 145.70000000001158, 512.1999999999
825, -1451.1000000000058, -83.69999999998254, -597.0, -2148.100000000006,
0.0, 35.5, 1031.0, 1649.8999999999942, -2023.8999999999946, 739.1000000000
058, -595.7000000000116, 1257.3999999999942, 190.99999999999977, 1121.0, 2
6.400000000023283, 3572.5, -1199.6000000000067, 2480.299999999989, -815.0,
-1948.7999999999884, -1353.1000000000058, 593.2000000000114, -1890.6000000
00006, 346.7999999999888, 826.5, -703.2999999999885, -542.8000000000175, -
167.0, 486.60000000000576, 867.7000000000116, 173.19999999998242, 699.2000
000000116, -812.8999999999941, -1002.7000000000116, 55.39999999999429, 15
5.0, 534.3000000000175, -1667.8000000000175, 2601.400000000023, 54.5999999
9997672, -22.799999999988366, 2013.7000000000119, 1586.3999999999942, 648.
1000000000058, 1532.2999999999884, 149.60000000000582, 1062.3999999999942,
-842.8999999999942, -641.5, -714.1000000000058, 180.9999999999999, -124.39
999999999421, 1148.1999999999825, -306.5, -525.0, 466.1000000000058, -144.
79999999998842, 3357.5000000000005, -989.2000000000107, 128.3000000000173
5, 908.5, 443.39999999999424, 583.8999999999942, 728.2000000000116, -338.8
0000000001746, 2772.2000000000116, -1317.7000000000116, -1362.399999999994
2, 577.3999999999944, -440.3999999999407, 1926.7000000000116, 793.5999999
999767, -1288.099999999977, 0.0, -2.524354896707238e-29, 0.0, -128.6000000
000058, 976.9999999999999, 0.0, 0.0, 2430.699999999983, 900.8000000000175,
-2086.8999999999937, 931.7999999999884, 1919.6000000000058, -926.100000000
0058, -535.7999999999884, -63.70000000001164, 2198.600000000006, 1543.1999
999999825, -446.79999999998813, 360.2000000000115, -564.4000000000234, 88.
40000000002328, -18.70000000001164, 751.7000000000116, 603.3999999999942,
131.5, -628.3999999999942, 81.79999999998836, 387.70000000001164, 649.7999

```
999999884, -226.10000000000593, -3348.1000000000054, 781.2000000000116, -5
37.600000000006, 148.60000000000582, 197.0, -1464.8999999999942, -122.2000
0000001164, 1225.7000000000116, -3040.200000000012, -313.2999999998836, 5
30.6999999999825, 910.7000000000116, -472.5, -182.10000000000588, -1068.60
00000000058, -672.7999999999884, 114.70000000001176, 368.0, 902.5999999999
767, -399.5000000000001, -878.6999999999825, 826.1000000000057, -1836.5, -
564.0000000000002, 1422.0, 2293.399999999994, 4859.899999999994, -395.6000
000000058, 2393.0, -397.8999999999942, 2964.0, 589.8999999999942, -1215.79
99999999881, -260.30000000001735, 1413.8000000000172, 1763.0, -4406.100000
000006, 1408.5, 377.1000000000058, 40.799999999988415, 909.9999999999999,
510.8000000000175, 2026.0999999999765, 1296.2000000000116, -2569.299999999
989, -4.547473508864641e-13, 0.0, 574.0000000000001, 221.59999999997677, 2
683.100000000006, -2702.8999999999937, -932.4999999999998, 466.89999999999
42, -561.5, 1152.5, -2776.2999999999884, 130.0, 927.7999999999884, 755.100
0000000058, -1442.5000000000005, -295.70000000001164, 5.684341886080802e-1
4, 1067.7000000000116, 1455.2999999999884, 2482.100000000006, -3613.399999
9999937, 0.0, 3170.399999999994, 324.60000000000537, -1342.8999999999942,
439.60000000000605, 57.19999999998254, -890.7999999999885, -2603.300000000
0175, -361.29999999998836, 242.0, 779.1000000000057, -69.89999999999418, -
338.9000000000233, -1624.6999999999828, -328.3999999999942, -5.68434188608
0802e-14, 560.2999999999884, 1004.8999999999942, 80.8999999999943, -776.79
99999999884, -38.899999999994066, -198.39999999999415, 986.2999999999885,
-330.1000000000056, -688.3999999999942, -506.1000000000058, -220.699999999
9826, -582.7000000000116, -229.79999999998842, 588.5999999999767, -695.699
9999999825, -1550.1000000000058, -444.6000000000058, -527.1999999999825, 9
70.8999999999943, -616.2000000000116, -460.3999999999942, -654.0, -716.200
0000000116, 1050.0000000000002, 1128.9000000000233, 57.29999999998836, 41
2.29999999998836, -520.6999999999825, 833.299999999885, -91.7999999999883
6, 634.6000000000058, 844.7999999999884, 408.2000000000117, -210.700000000
01164, -451.20000000001164, 2126.800000000018, 74.60000000000537, 17.69999
9999982545]
1823.7831349811345
```



In [42]:

```
# Hence we have seen the work of ARIMA lets forecast for 60 days and compar
history = [x for x in train_data]
model=ARIMA(history,order=(0,1,0))
model_fit = model.fit()
yhat = model_fit.summary()
yhat
```

Out[42]:

SARIMAX Results

Dep. Variable:	y	**No. Observations:**	2657
Model:	ARIMA(0, 1, 0)	**Log Likelihood**	-22346.286
Date:	Sun, 17 Mar 2024	**AIC**	44694.573
Time:	17:32:49	**BIC**	44700.457
Sample:	0	**HQIC**	44696.703
	- 2657		
Covariance Type:	opg		

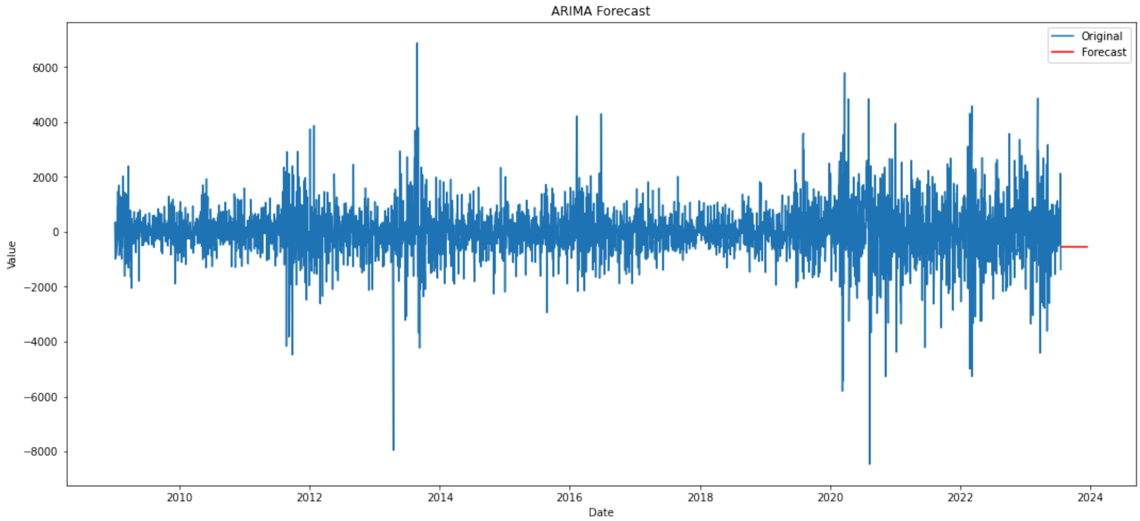
	coef	**std err**	**z**	**P>|z|**	**[0.025**	**0.975]**
sigma2	1.189e+06	1.7e+04	69.904	0.000	1.16e+06	1.22e+06

Ljung-Box (L1) (Q):	641.85	**Jarque-Bera (JB):**	3176.48
Prob(Q):	0.00	**Prob(JB):**	0.00
Heteroskedasticity (H):	0.71	**Skew:**	0.10
Prob(H) (two-sided):	0.00	**Kurtosis:**	8.35

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).

In [43]:
```python
# Forecast future values for 60 days
forecast_values = model_fit.forecast(steps=150)

# Create a time index for the forecasted values (e.g., next 60 days)
# Example: If the last date in your original data is '2022-12-31', create a
forecast_dates = pd.date_range(start='2023-7-22', periods=150)

# Plot original values
plt.plot(diff_1.index, diff_1['INR'], label='Original')

# Plot forecasted values
plt.plot(forecast_dates, forecast_values, label='Forecast', color='red')

# Add labels and legend
plt.xlabel('Date')
plt.ylabel('Value')
plt.title('ARIMA Forecast')
plt.legend()
```

Out[43]: <matplotlib.legend.Legend at 0x1a5a937d5c8>

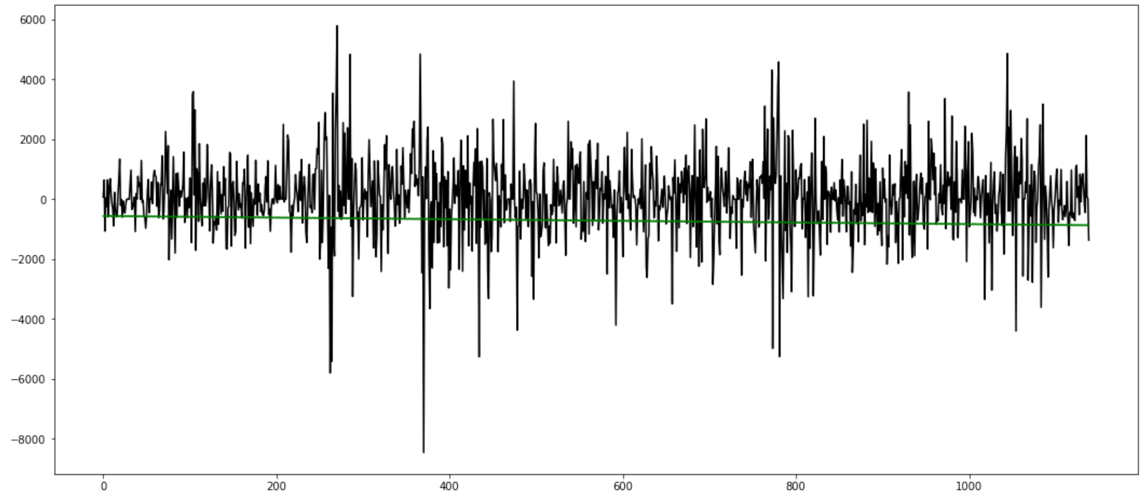


SARIMAX

In [46]:
```python
import numpy as np
import statsmodels.api as sm
from sklearn.metrics import mean_squared_error

# Fit SARIMA model to the training data
model = sm.tsa.statespace.SARIMAX(train_data, order=(0, 1, 0), seasonal_ord
model_fit = model.fit()

# Forecast future values for the entire test set
forecast_values = model_fit.forecast(steps=len(test_data))

# Calculate RMSE
rmse = np.sqrt(mean_squared_error(test_data, forecast_values))
print("RMSE:", rmse)
```

RMSE: 1497.082055537108

In [48]:
```
plt.plot(test_data,color='black',label='original')
plt.plot(forecast_values,color='green',label='forecasted values')
plt.show()
```



In [58]:
```
forecast_values.min()
```

Out[58]: -883.4631738481506

In [59]:
```
forecast_values.max()
```

Out[59]: -546.9506001278564

In [60]:
```
test_data.min()
```

Out[60]: -8461.599999999977

In [61]:
```
test_data.max()
```

Out[61]: 5785.0

If the visual representation of the forecasted values appears flat below zero when compared to the test dataset, it suggests that the model may not be accurately capturing the patterns or dynamics of the data. Here are some steps you can take to address this issue and improve the accuracy of your forecasts:

Model Selection: Revisit the choice of SARIMA model parameters (e.g., order and seasonal order) and consider whether they adequately capture the underlying patterns in the data. Experiment with different parameter configurations to find the best-fitting model.

Data Preprocessing: Ensure that the data preprocessing steps are appropriate for the modeling task. Consider techniques such as differencing, transformation, or outlier removal to make the data more amenable to modeling.

Model Evaluation: Evaluate the performance of the SARIMA model using additional diagnostic tools such as residual analysis, autocorrelation plots, and out-of-sample forecasting accuracy metrics. Identify any systematic errors or patterns in the model residuals that may indicate areas for improvement.

Feature Engineering: Explore the possibility of incorporating additional features or external variables that may improve the model's predictive performance. For example, economic indicators, weather data, or holiday information could provide valuable information for forecasting certain time series.

Ensemble Methods: Consider using ensemble methods such as model averaging or stacking to combine the predictions of multiple SARIMA models or different forecasting techniques. Ensemble methods can often lead to more robust and accurate forecasts by leveraging the strengths of individual models.

Hyperparameter Tuning: Fine-tune the hyperparameters of the SARIMA model, such as optimization algorithms, learning rates, and regularization parameters, to improve convergence and overall model performance.

Model Comparison: Compare the SARIMA model with alternative forecasting methods, such as machine learning algorithms (e.g., LSTM, Random Forests) or exponential smoothing methods, to determine whether a different approach may yield better results for your dataset.

Domain Expertise: Incorporate domain expertise and domain-specific knowledge into the modeling process to ensure that the forecasts align with the underlying dynamics and behavior of the time series data.

By carefully evaluating and adjusting the SARIMA model and considering alternative

In [ ]:

```

```