In [210]:
```
import numpy as np
import pandas as pd
import matplotlib
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import boxcox
```

In [44]:
```
data=pd.read_csv('C:/Users/User/Downloads/G17MVSFTTRUCKS.csv',header=0,index_col=
```

In [45]:
```
data.head()
```

Out[45]:

DATE	G17MVSFTTRUCKS
1996-01-01	99.23
1996-02-01	105.93
1996-03-01	109.04
1996-04-01	106.96
1996-05-01	113.06

In [51]:
```
len(data)
```

Out[51]:
```
330
```

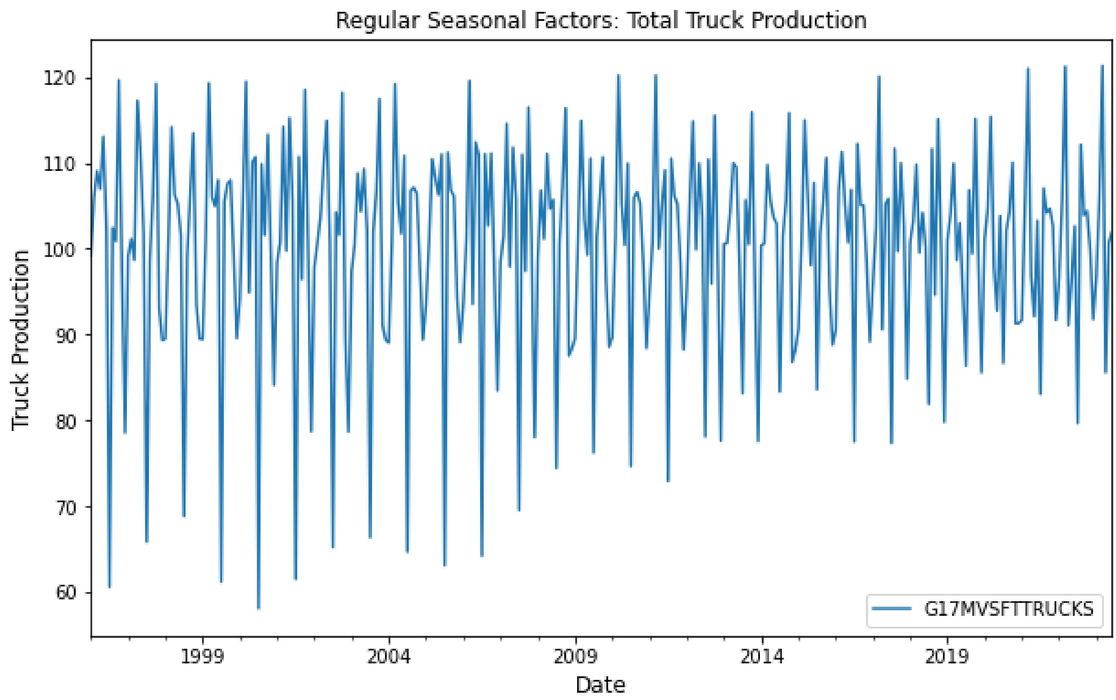
In [46]:
```
data.describe()
```

Out[46]:

	G17MVSFTTRUCKS
count	330.000000
mean	100.122758
std	12.062724
min	58.050000
25%	94.955000
50%	101.595000
75%	107.027500
max	121.320000

In [ ]:
```
Time series plot
```

In [85]:
```
data.plot()
plt.title('Regular Seasonal Factors: Total Truck Production')
plt.xlabel('Date', size = 12)
plt.ylabel('Truck Production', size  = 12)
plt.rc("figure", figsize=(10,8))
```



In [ ]:
```
We can observe from the above plot that the data is stationary.

Next we shall perform the Augmented Dickey Fuller test to ensure our data is stat
```

In [86]:
```
from statsmodels.tsa.stattools import adfuller
```

In [87]:
```
adfuller(data)
```

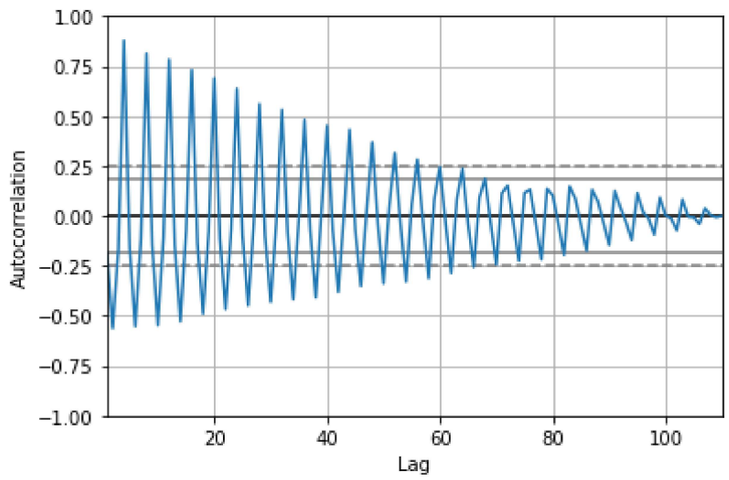
Out[87]:
```
(-3.338469786748859,
 0.013244039555686926,
 17,
 312,
 {'1%': -3.4514843502727306,
  '5%': -2.8708485956333556,
  '10%': -2.571729625657462},
 1735.0980479552418)
```

In [ ]:
```
The Augmented Dickey fuller test has a P value of greater less than 0.05 which se
stationary.
so we will move ahead to next step to plot the acf and pacf and find the models
```

In [89]:
```
from statsmodels.graphics.tsaplots import plot_acf
from statsmodels.graphics.tsaplots import plot_pacf
```

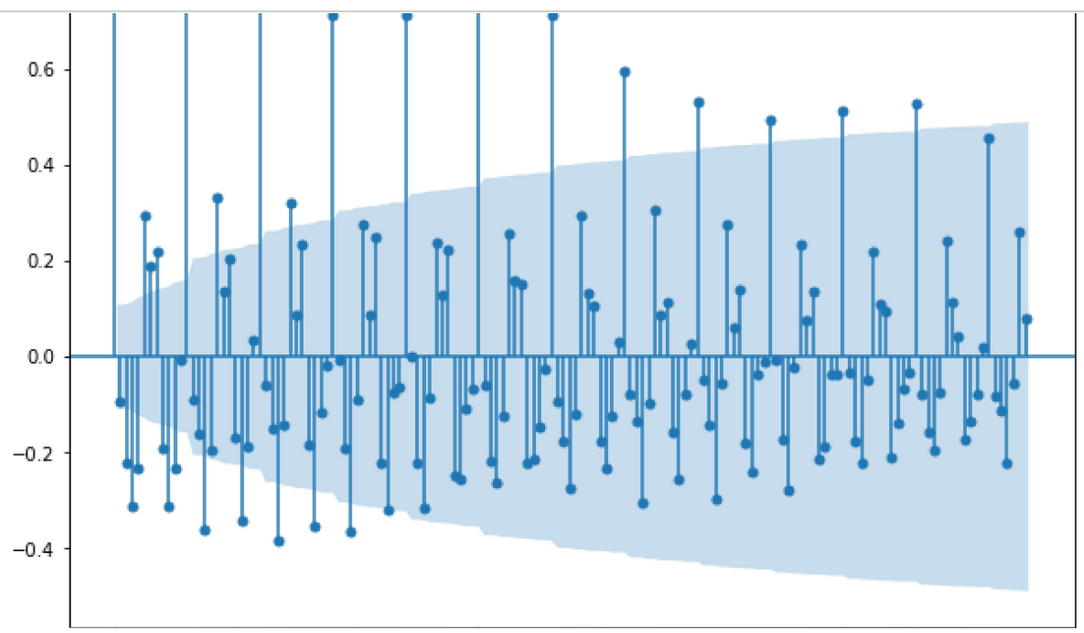
In [22]:
```
import pandas
from pandas.plotting import autocorrelation_plot
from matplotlib import pyplot
%matplotlib inline
```

In [23]:
```
#Test
autocorrelation_plot(data_q)
pyplot.show()
```



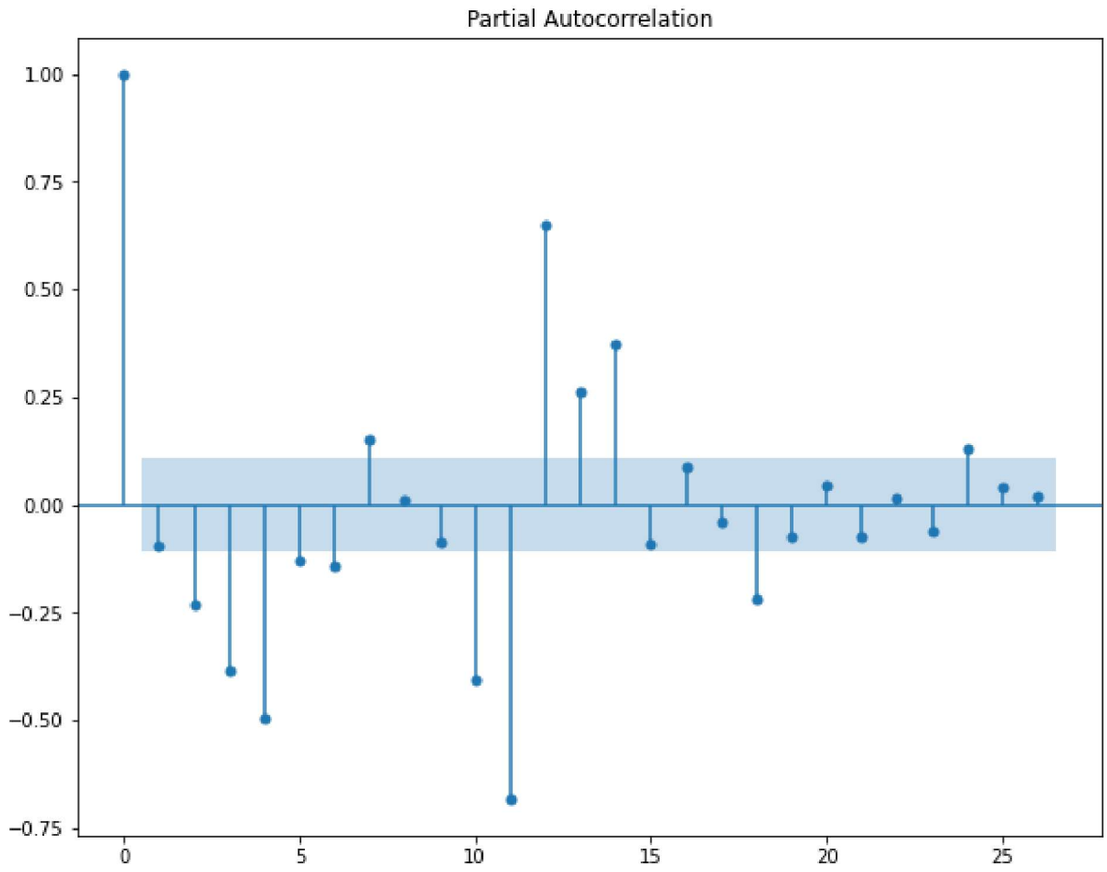
In [ ]:
```
Plotting the ACF and Pacf
```

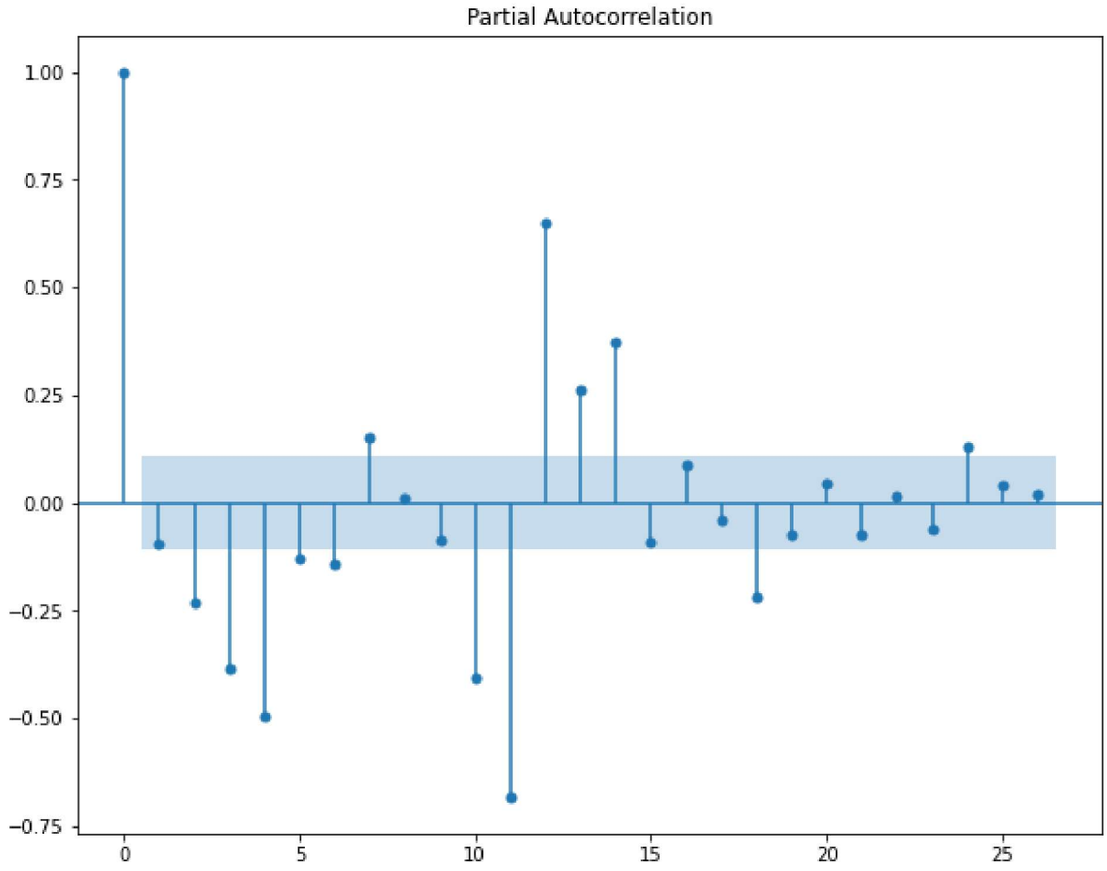
In [331]:
```
plot_acf(data,lags=150)
```



In [94]:
```
plot_pacf(data)
```

Out[94]:





In [ ]:
```
Model Prediction by checking p,d,q values from ACF and PACF graph
```

In [96]:
```
from pmdarima import auto_arima
import warnings
warnings.filterwarnings("ignore")
stepwise_fit=auto_arima(data['G17MVSFTTRUCKS'],trace=True,suppress_warnings=True)
```

```
Performing stepwise search to minimize aic
 ARIMA(2,0,2)(0,0,0)[0] intercept   : AIC=2537.704, Time=0.57 sec
 ARIMA(0,0,0)(0,0,0)[0] intercept   : AIC=2582.977, Time=0.01 sec
 ARIMA(1,0,0)(0,0,0)[0] intercept   : AIC=2582.045, Time=0.03 sec
 ARIMA(0,0,1)(0,0,0)[0] intercept   : AIC=2546.157, Time=0.07 sec
 ARIMA(0,0,0)(0,0,0)[0]             : AIC=3983.463, Time=0.01 sec
 ARIMA(1,0,2)(0,0,0)[0] intercept   : AIC=2549.563, Time=0.80 sec
 ARIMA(2,0,1)(0,0,0)[0] intercept   : AIC=2579.678, Time=0.26 sec
 ARIMA(3,0,2)(0,0,0)[0] intercept   : AIC=2567.545, Time=0.62 sec
 ARIMA(2,0,3)(0,0,0)[0] intercept   : AIC=2465.026, Time=0.70 sec
 ARIMA(1,0,3)(0,0,0)[0] intercept   : AIC=2489.930, Time=0.47 sec
 ARIMA(3,0,3)(0,0,0)[0] intercept   : AIC=2455.023, Time=0.72 sec
 ARIMA(4,0,3)(0,0,0)[0] intercept   : AIC=2409.031, Time=0.95 sec
 ARIMA(4,0,2)(0,0,0)[0] intercept   : AIC=2487.515, Time=0.73 sec
 ARIMA(5,0,3)(0,0,0)[0] intercept   : AIC=2361.373, Time=1.11 sec
 ARIMA(5,0,2)(0,0,0)[0] intercept   : AIC=2424.988, Time=1.02 sec
 ARIMA(5,0,4)(0,0,0)[0] intercept   : AIC=2439.187, Time=1.08 sec
 ARIMA(4,0,4)(0,0,0)[0] intercept   : AIC=2366.167, Time=0.97 sec
 ARIMA(5,0,3)(0,0,0)[0]             : AIC=inf, Time=0.88 sec

Best model:  ARIMA(5,0,3)(0,0,0)[0] intercept
Total fit time: 11.001 seconds
```

In [98]:
```
stepwise_fit.summary()
```

Out[98]:

SARIMAX Results

Dep. Variable:	y	**No. Observations:**	330
Model:	SARIMAX(5, 0, 3)	**Log Likelihood**	-1170.687
Date:	Mon, 16 May 2022	**AIC**	2361.373
Time:	16:38:50	**BIC**	2399.364
Sample:	0	**HQIC**	2376.527
	- 330		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
intercept	317.9807	33.258	9.561	0.000	252.796	383.165
ar.L1	-0.4832	0.108	-4.477	0.000	-0.695	-0.272
ar.L2	-0.1495	0.114	-1.306	0.192	-0.374	0.075
ar.L3	-0.9453	0.039	-23.946	0.000	-1.023	-0.868
ar.L4	-0.4472	0.091	-4.931	0.000	-0.625	-0.269
ar.L5	-0.1477	0.096	-1.544	0.123	-0.335	0.040
ma.L1	-0.0872	0.110	-0.796	0.426	-0.302	0.128
ma.L2	-0.1732	0.106	-1.632	0.103	-0.381	0.035
ma.L3	0.8566	0.094	9.108	0.000	0.672	1.041
sigma2	84.6304	9.433	8.971	0.000	66.141	103.119

Ljung-Box (L1) (Q):	0.01	**Jarque-Bera (JB):**	4.85
Prob(Q):	0.90	**Prob(JB):**	0.09
Heteroskedasticity (H):	0.43	**Skew:**	0.24
Prob(H) (two-sided):	0.00	**Kurtosis:**	3.35

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

In [412]:
```
model=ARIMA(data['G17MVSFTTRUCKS'],order=(4,0,4))
model=model.fit()
model.summary()
```

Out[412]:

SARIMAX Results

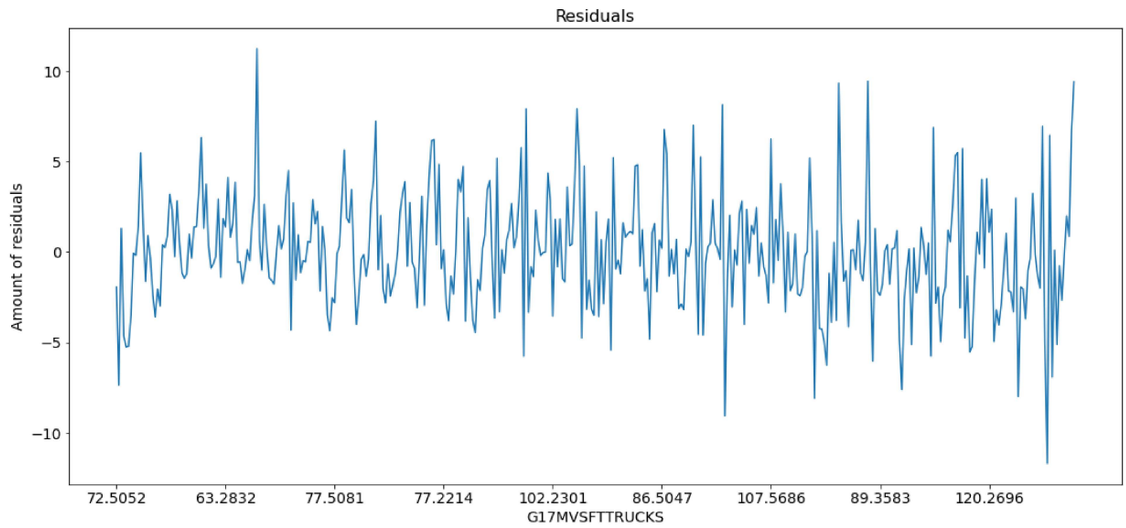
Dep. Variable:	G17MVSFTTRUCKS	**No. Observations:**	330
Model:	ARIMA(4, 0, 4)	**Log Likelihood**	-1135.360
Date:	Mon, 16 May 2022	**AIC**	2290.720
Time:	23:12:20	**BIC**	2328.711
Sample:	01-01-1996	**HQIC**	2305.874
	- 06-01-2023		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
const	100.0955	0.333	300.652	0.000	99.443	100.748
ar.L1	-0.7377	0.013	-55.287	0.000	-0.764	-0.712
ar.L2	-0.2784	0.022	-12.560	0.000	-0.322	-0.235
ar.L3	-0.7402	0.013	-57.224	0.000	-0.766	-0.715
ar.L4	-0.9966	0.003	-332.261	0.000	-1.002	-0.991
ma.L1	0.5217	48.691	0.011	0.991	-94.911	95.955
ma.L2	-0.1375	5.165	-0.027	0.979	-10.261	9.986
ma.L3	0.5314	90.666	0.006	0.995	-177.170	178.233
ma.L4	0.9921	79.088	0.013	0.990	-154.018	156.002
sigma2	52.2048	4161.318	0.013	0.990	-8103.829	8208.238

Ljung-Box (L1) (Q):	9.11	**Jarque-Bera (JB):**	4.27
Prob(Q):	0.00	**Prob(JB):**	0.12
Heteroskedasticity (H):	0.46	**Skew:**	-0.28
Prob(H) (two-sided):	0.00	**Kurtosis:**	2.98

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

In [425]:
```
# Plotting residual
residuals = pd.DataFrame(model2_fit.resid)
plt.figure(figsize=(18, 8))
plt.plot(residuals)
plt.title('Residuals', fontsize=16)
plt.xlabel("G17MVSFTTRUCKS", fontsize=14)
plt.ylabel("Amount of residuals", fontsize=14)
plt.xticks(np.arange(0, len(data2.IPG2211A2N)+1, 45), labels=[data2.IPG2211A2N[i]

plt.yticks(fontsize=14)
plt.show()
```



In [99]:
```
print(data.shape)
train= data.iloc[:-30]
test= data.iloc[-30:]
print(train.shape,test.shape)
```

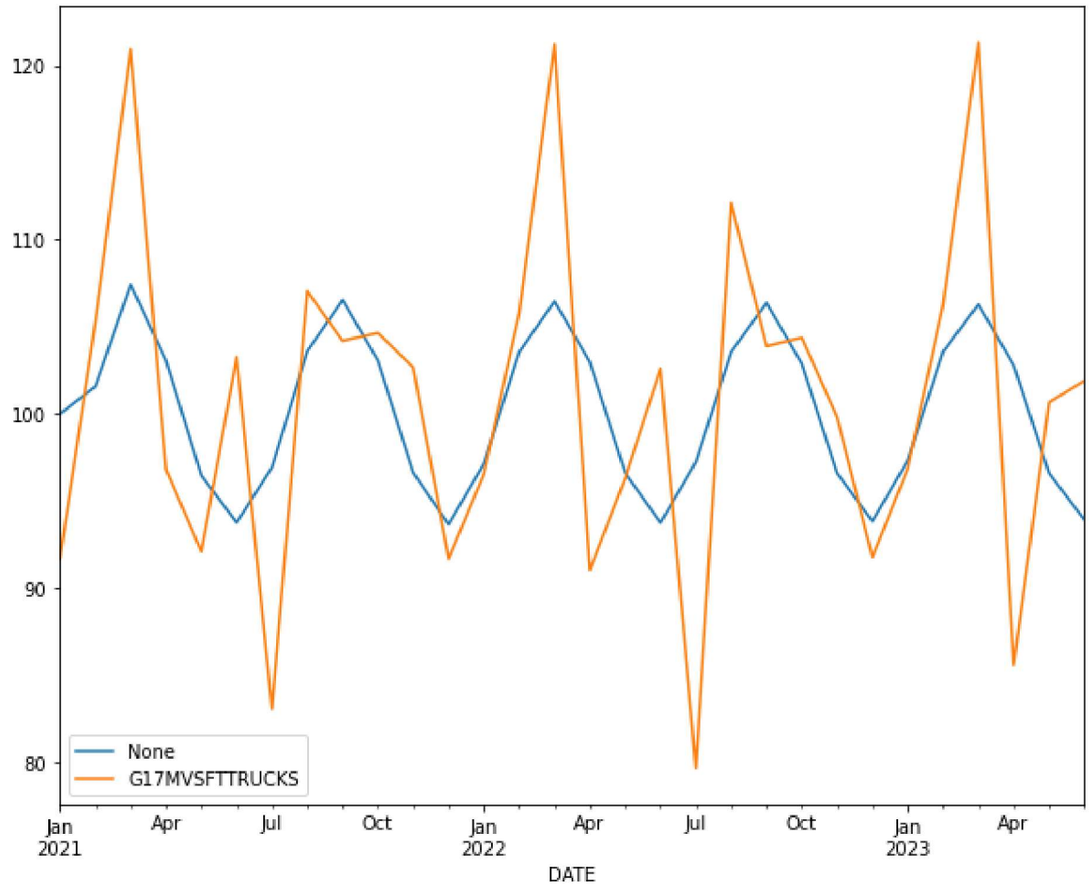
```
(330, 1)
(300, 1) (30, 1)
```

In [ ]:
```
model=ARIMA(train['G17MVSFTTRUCKS'],order=(5,0,3))
model=model.fit()
model.summary()
```

In [370]:
```
start=len(train)
end=len(train)+len(test)
predictions = model.predict(start=start, end=end, dynamic=False, typ='levels')
```

In [102]:
```
pred.plot(legend=True)
test['G17MVSFTTRUCKS'].plot(legend=True)
```

Out[102]: <AxesSubplot:xlabel='DATE'>



In [ ]:
```
From the above graph we can say that the actual and predicted values looks almost
Hence we can say it is the best model
```

In [39]:
```
#Non-Seasonal Dataset
```

In [227]:
```
data2=pd.read_csv('C:/Users/User/Downloads/Electric_Production.csv',header=0,pars
print(data2)
```

```
          DATE  IPG2211A2N
0     1/1/1985     72.5052
1     2/1/1985     70.6720
2     3/1/1985     62.4502
3     4/1/1985     57.4714
4     5/1/1985     55.3151
..         ...         ...
392   9/1/2017     98.6154
393  10/1/2017     93.6137
394  11/1/2017     97.3359
395  12/1/2017    114.7212
396   1/1/2018    129.4048

[397 rows x 2 columns]
```

In [423]:
```
data2=pd.read_csv('C:/Users/User/Downloads/Electric_Production.csv',header=0,inde
data2
```

Out[423]:

DATE	IPG2211A2N
1985-01-01	72.5052
1985-02-01	70.6720
1985-03-01	62.4502
1985-04-01	57.4714
1985-05-01	55.3151
...	...
2017-09-01	98.6154
2017-10-01	93.6137
2017-11-01	97.3359
2017-12-01	114.7212
2018-01-01	129.4048

397 rows × 1 columns

In [128]:
```
data2.dtypes=='object'
num_vars=data2.columns[data2.dtypes!='object']
data2[num_vars]
```

Out[128]:

DATE	IPG2211A2N
1985-01-01	72.5052
1985-02-01	70.6720
1985-03-01	62.4502
1985-04-01	57.4714
1985-05-01	55.3151
...	...
2017-09-01	98.6154
2017-10-01	93.6137
2017-11-01	97.3359
2017-12-01	114.7212
2018-01-01	129.4048

397 rows × 1 columns

In [129]:
```
#Checking the null values in our dataset
data2[num_vars].isnull()
```

Out[129]:

DATE	IPG2211A2N
1985-01-01	False
1985-02-01	False
1985-03-01	False
1985-04-01	False
1985-05-01	False
...	...
2017-09-01	False
2017-10-01	False
2017-11-01	False
2017-12-01	False
2018-01-01	False

397 rows × 1 columns

In [130]:
```
data2.isnull().sum()
```

Out[130]:
```
IPG2211A2N    0
dtype: int64
```

In [131]:
```
data2.head()
```

Out[131]:

DATE	IPG2211A2N
1985-01-01	72.5052
1985-02-01	70.6720
1985-03-01	62.4502
1985-04-01	57.4714
1985-05-01	55.3151

In [132]:
```
data2.describe()
```

Out[132]:

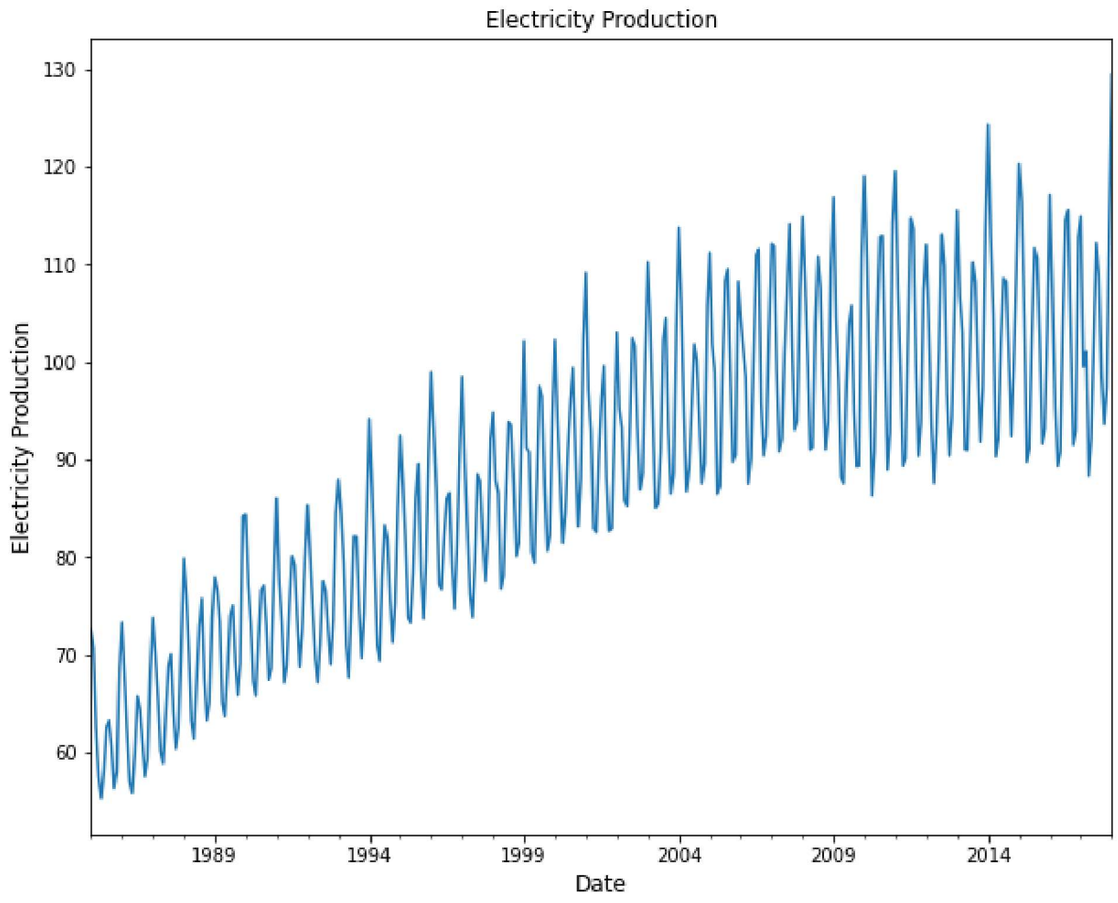
	IPG2211A2N
count	397.000000
mean	88.847218
std	15.387834
min	55.315100
25%	77.105200
50%	89.779500
75%	100.524400
max	129.404800

In [133]:
```
data2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 397 entries, 1985-01-01 to 2018-01-01
Data columns (total 1 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   IPG2211A2N  397 non-null    float64
dtypes: float64(1)
memory usage: 6.2 KB
```

In [ ]:
```
Time series plot
```

In [424]:
```
y=data2.IPG2211A2N
y.plot()
plt.rc("figure", figsize=(10,8))
plt.title('Electricity Production')
plt.xlabel('Date', size = 12)
plt.ylabel('Electricity Production', size  = 12)
plt.rc("figure", figsize=(10,8))
```



In [ ]:
```
Looking at the plot we can observe there is an **upward trend** over the period c


Stationarity Test

We can observe from the above plot that the Electric production is fairly seasona
Next we shall perform the Augmented Dickey Fuller test to see if the trend of the
is stationary or non stationary.
```

In [ ]:
```
The augmented Dickey-Fuller (ADF) test statistic is the t-statistic of the estima
least squares regression. However, the ADF test statistic
is not approximately t-distributed under the null hypothesis; instead,
it has a certain nonstandard large-sample distribution under the null hypothesis
```

In [229]:
```
from statsmodels.tsa.stattools import adfuller
```

In [293]:
```
data2=pd.read_csv('C:/Users/User/Downloads/Electric_Production.csv',header=0,inde

adfuller(data2)
```

Out[293]:
```
(-2.2569903500472455,
 0.18621469116586759,
 15,
 381,
 {'1%': -3.4476305904172904,
  '5%': -2.869155980820355,
  '10%': -2.570827146203181},
 1840.8474501627156)
```

In [ ]:
```
The Augmented Dickey fuller test has a P value of greater than 0.05 which seems t
is non-stationary. The p-value is obtained is greater than significance level of
any of the critical values.

Clearly, there is no reason to reject the null hypothesis. So, the time series is

We can clearly see a trend in data so let us perform some more formal test of sta
```

In [291]:
```
#first differencing
first_difference=adfuller(data2.IPG2211A2N.diff().dropna())
print(f'ADF Statistics:{result[0]}')
print(f'p-value:{result[1]}')
```

```
ADF Statistics:-6.748333370019157
p-value:2.9951614981156204e-09
```

In [ ]:
```
The p-value:2.9951614981156204e-09 is less than 0.05. Now the series looks static
so q will be 1 for our model. I am also testing another method two make data stat
Converting Non-stationary data to Stationary
```

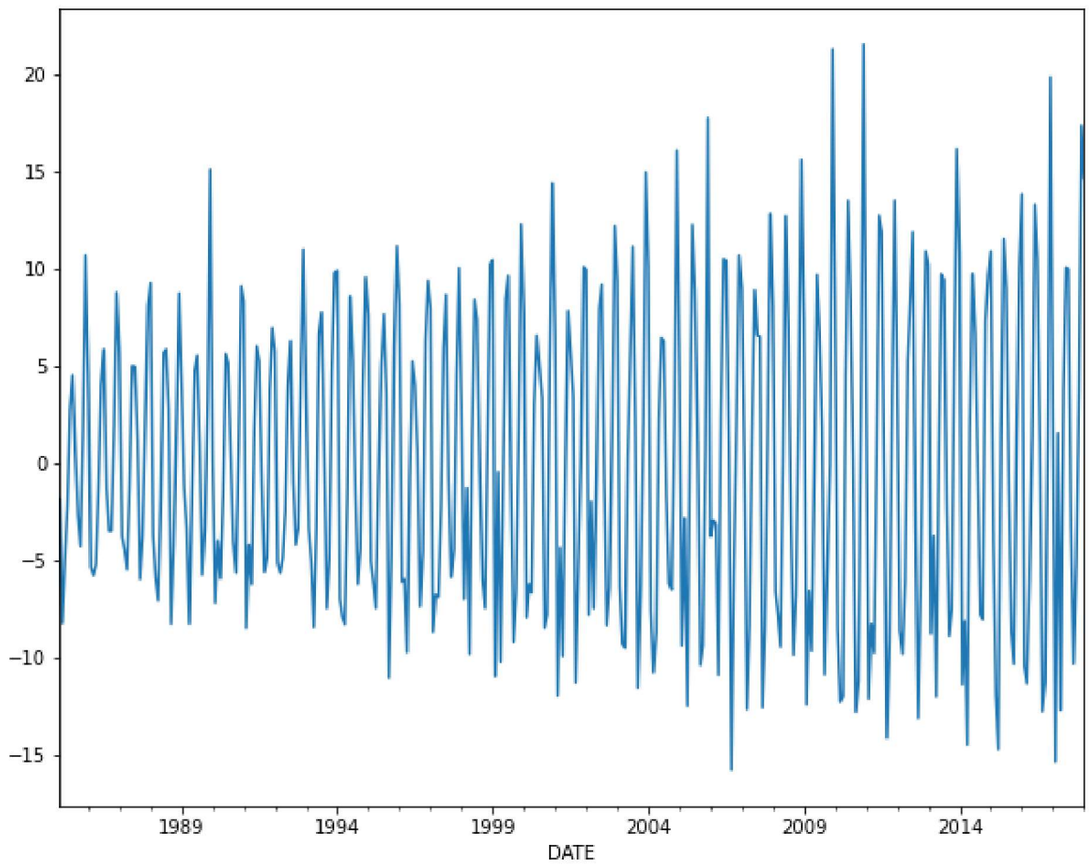
In [236]:
```
#Method2 only for Test
#Differencing and then log transformation
#Making data Stationary using Differencing
from statsmodels.tsa.stattools import adfuller
y1=y-y.shift(1)
y1=y1[1:]
y2=np.log(y1-np.min(y1)+1)
```

In [237]:
```
result=adfuller(y1)
print('ADF statistic: %f' %result[0])
print('p-value: %f' %result[1])
print('critical values:')

for key,value in result[4].items():
    print('\t%s: %.3f' % (key,value))
```

```
ADF statistic: -7.104891
p-value: 0.000000
critical values:
        1%: -3.448
        5%: -2.869
        10%: -2.571
```

In [238]:
```
y1.plot()
```

Out[238]: <AxesSubplot:xlabel='DATE'>



In [ ]:
```
The data looks stationary now but we can also try First Log Transformation
and then differencing (only for test purpose and I am taking d as 1 as data chang
in first differencing)
```

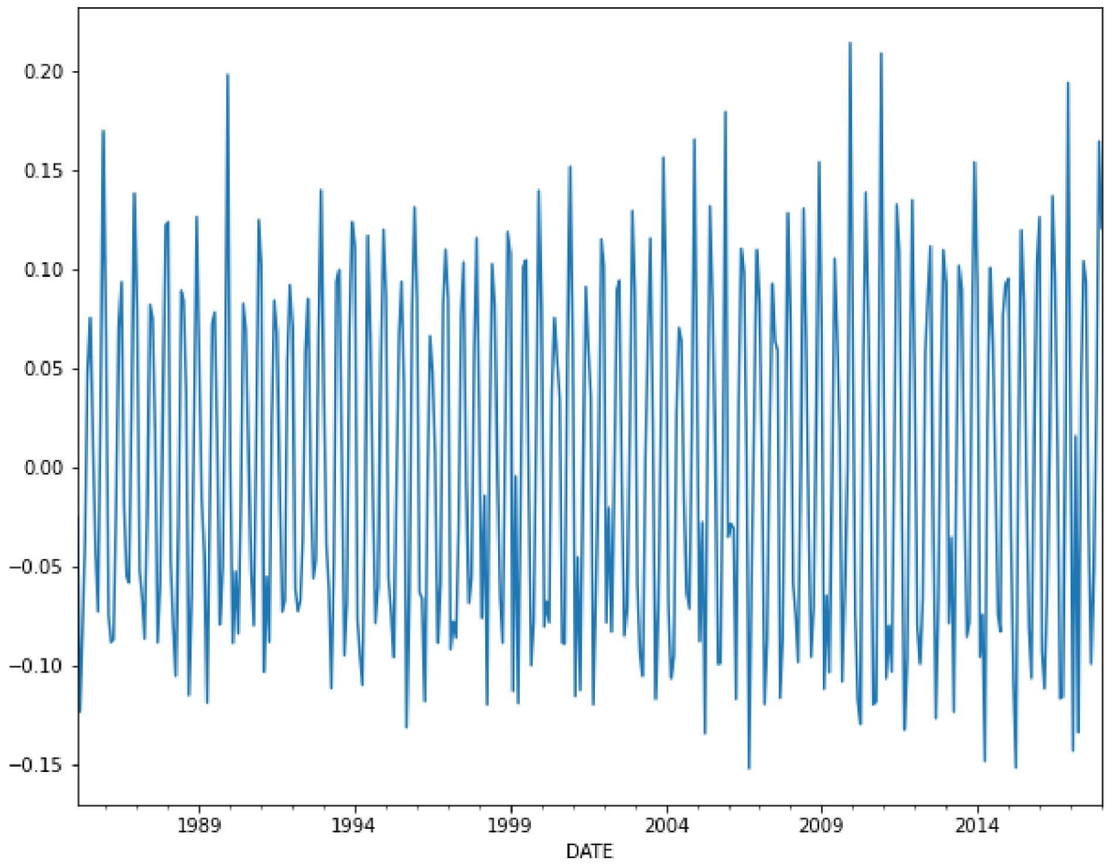
In [239]:
```
from statsmodels.tsa.stattools import adfuller
y1=np.log(y)
y2=y1-y1.shift(1)
y2.dropna(inplace=True)
```

In [240]:
```
result=adfuller(y2)

print('ADF statistic: %f' %result[0])
print('p-value: %f' %result[1])
print('critical values:')

for key,value in result[4].items():
    print('\t%s: %.3f' % (key,value))
```

```
ADF statistic: -6.748333
p-value: 0.000000
critical values:
        1%: -3.448
        5%: -2.869
        10%: -2.571
```

In [241]:
```
y2.plot()
```

Out[241]: <AxesSubplot:xlabel='DATE'>



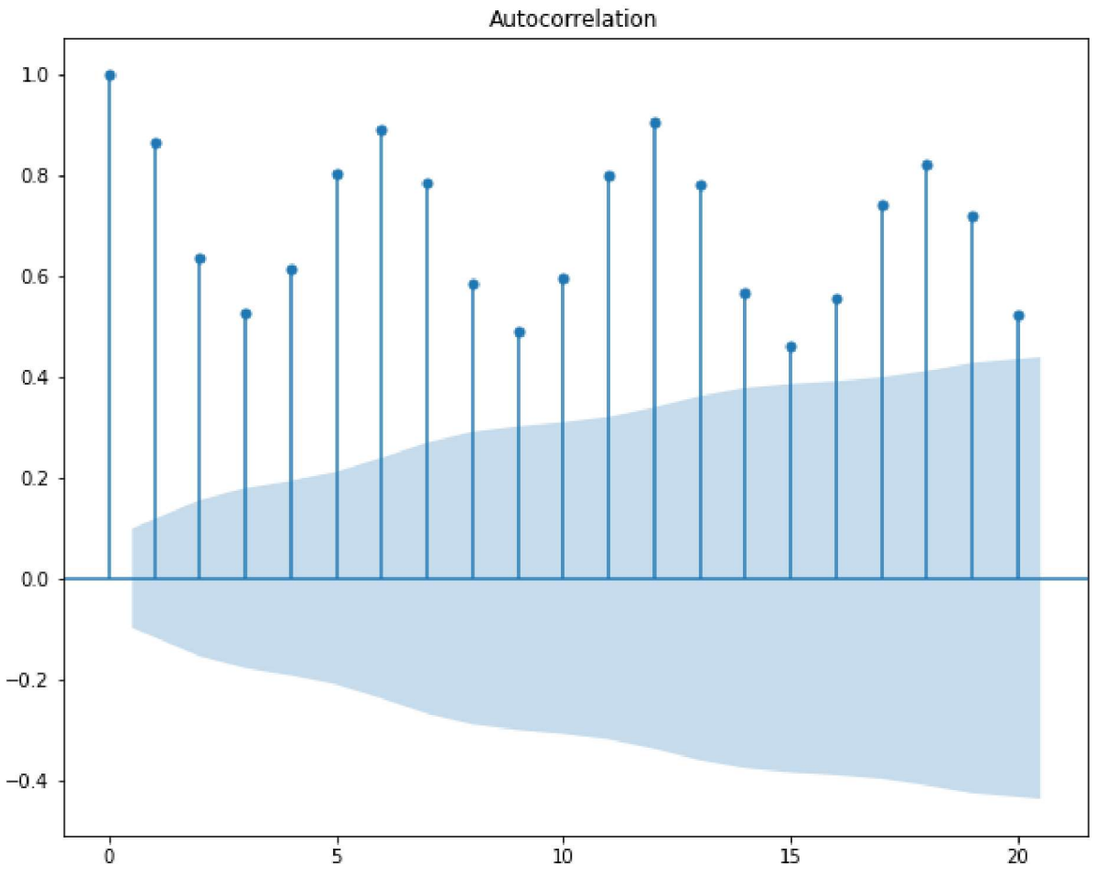
In [142]:
```
print("Data Shape: {}".format(data2.shape))
value_1 = data2[0:198]
value_2 = data2[199:398]
```

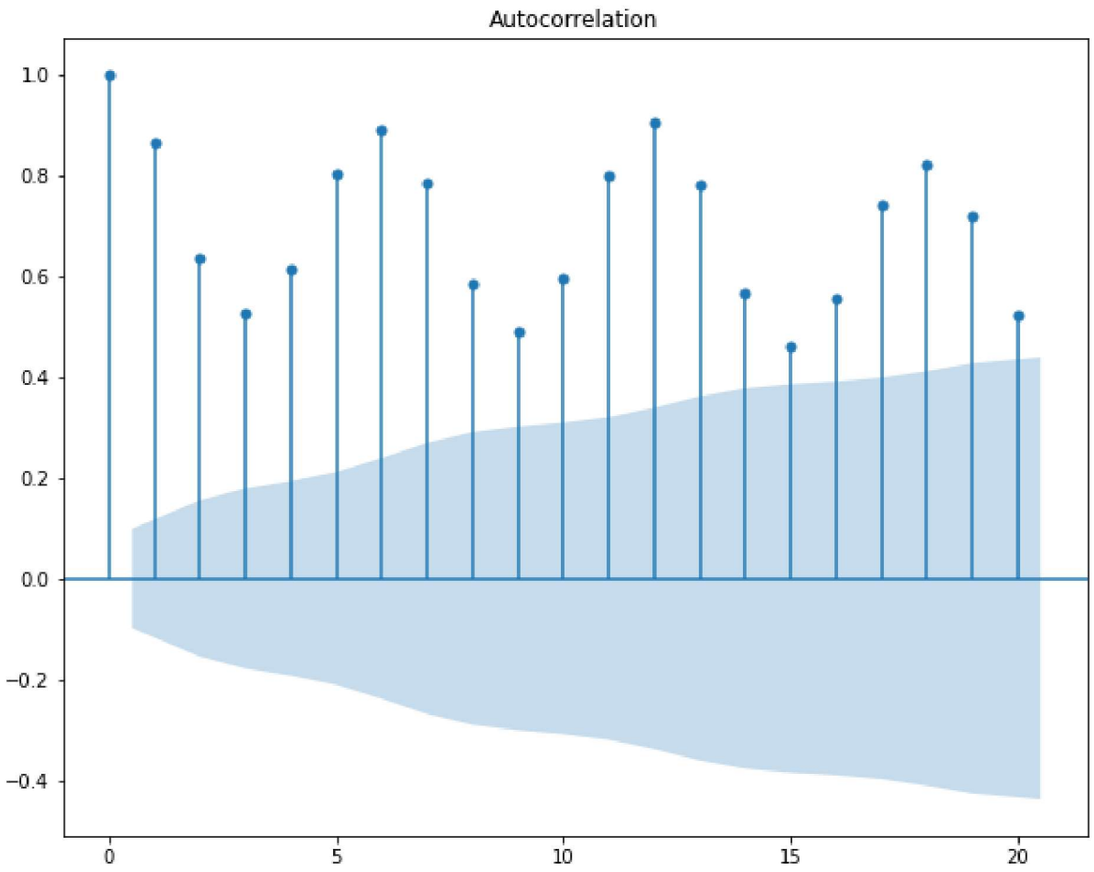
```
Data Shape: (397, 1)
```

In [242]:
```
from statsmodels.graphics.tsaplots import plot_acf
from statsmodels.graphics.tsaplots import plot_pacf
```

In [247]:
```
plot_acf(data2,lags=20)
```

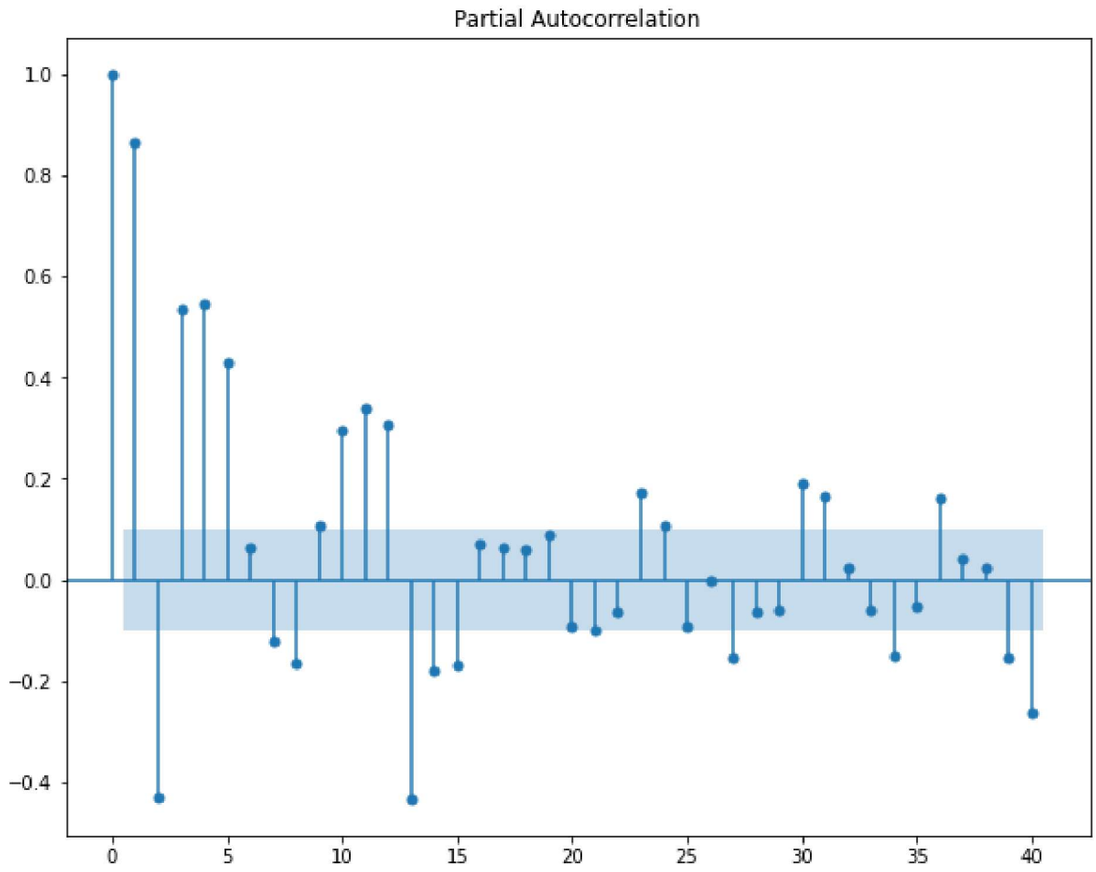
Out[247]:

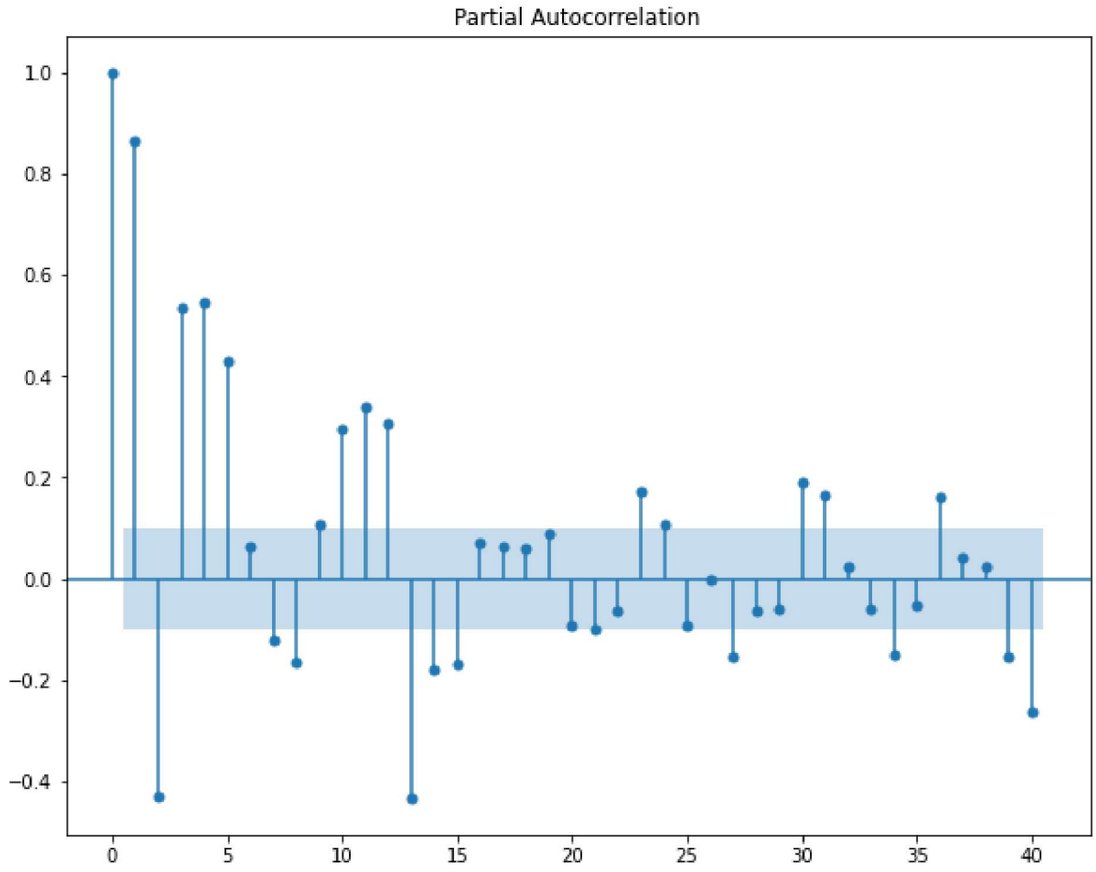




In [246]:
```
plot_pacf(data2,lags=40)
```

Out[246]:





In [ ]:
```
Forecasting models

"Pmdarima" is a useful package to help us find the best parameters for SARIMA mod
```

In [157]:
```
!pip install pmdarima
```

```
Requirement already satisfied: pmdarima in c:\users\user\anaconda3\lib\site-pac
kages (1.8.5)
Requirement already satisfied: scipy>=1.3.2 in c:\users\user\anaconda3\lib\site
-packages (from pmdarima) (1.6.2)
Requirement already satisfied: Cython!=0.29.18,>=0.29 in c:\users\user\anaconda
3\lib\site-packages (from pmdarima) (0.29.23)
Requirement already satisfied: urllib3 in c:\users\user\anaconda3\lib\site-pack
ages (from pmdarima) (1.26.4)
Requirement already satisfied: numpy>=1.19.3 in c:\users\user\anaconda3\lib\sit
e-packages (from pmdarima) (1.20.1)
Requirement already satisfied: pandas>=0.19 in c:\users\user\anaconda3\lib\site
-packages (from pmdarima) (1.2.4)
Requirement already satisfied: setuptools!=50.0.0,>=38.6.0 in c:\users\user\ana
conda3\lib\site-packages (from pmdarima) (52.0.0.post20210125)
Requirement already satisfied: scikit-learn>=0.22 in c:\users\user\anaconda3\li
b\site-packages (from pmdarima) (0.24.1)
Requirement already satisfied: joblib>=0.11 in c:\users\user\anaconda3\lib\site
-packages (from pmdarima) (1.0.1)
Requirement already satisfied: statsmodels!=0.12.0,>=0.11 in c:\users\user\anac
onda3\lib\site-packages (from pmdarima) (0.12.2)
Requirement already satisfied: pytz>=2017.3 in c:\users\user\anaconda3\lib\site
-packages (from pandas>=0.19->pmdarima) (2021.1)
Requirement already satisfied: python-dateutil>=2.7.3 in c:\users\user\anaconda
3\lib\site-packages (from pandas>=0.19->pmdarima) (2.8.1)
Requirement already satisfied: six>=1.5 in c:\users\user\anaconda3\lib\site-pac
kages (from python-dateutil>=2.7.3->pandas>=0.19->pmdarima) (1.15.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\user\anaconda3
\lib\site-packages (from scikit-learn>=0.22->pmdarima) (2.1.0)
Requirement already satisfied: patsy>=0.5 in c:\users\user\anaconda3\lib\site-p
ackages (from statsmodels!=0.12.0,>=0.11->pmdarima) (0.5.1)
```

In [158]:
```
from pmdarima import auto_arima
import warnings
warnings.filterwarnings("ignore")
```

In [ ]:
```
ARIMA Forecast to get best p,d,q,P,D,Q values
```

In [253]:
```
from statsmodels.tsa.arima_model import ARIMA
from statsmodels.tsa.statespace.sarimax import SARIMAX
```

In [283]:
```
from statsmodels.tsa.arima_model import ARIMA

model1=ARIMA(data2['IPG2211A2N'],order=(2,1,4))
model1_fit=model1.fit()
print(model1_fit.summary())
```

```
                             ARIMA Model Results
==============================================================================
Dep. Variable:           D.IPG2211A2N   No. Observations:                  396
Model:                 ARIMA(2, 1, 4)   Log Likelihood               -1032.303
Method:                       css-mle   S.D. of innovations       11550678.802
Date:                Mon, 16 May 2022   AIC                           2080.605
Time:                        21:08:31   BIC                           2112.457
Sample:                    02-01-1985   HQIC                          2093.224
                         - 01-01-2018
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.1075   4.35e-05   2472.383      0.000       0.107       0.108
ar.L1.D.IPG2211A2N     0.9997      0.000   2472.306      0.000       0.999       1.000
ar.L2.D.IPG2211A2N    -1.0000        nan        nan        nan         nan         nan
ma.L1.D.IPG2211A2N    -1.3643      0.009   -151.276      0.000      -1.382      -1.347
ma.L2.D.IPG2211A2N     0.9130        nan        nan        nan         nan         nan
ma.L3.D.IPG2211A2N     0.0126      0.002      5.872      0.000       0.008       0.017
ma.L4.D.IPG2211A2N    -0.4326      0.008    -51.455      0.000      -0.449      -0.416
                                    Roots
=============================================================================
                  Real          Imaginary           Modulus         Frequency
-----------------------------------------------------------------------------
AR.1            0.4998           -0.8661j            1.0000           -0.1667
AR.2            0.4998           +0.8661j            1.0000            0.1667
MA.1            1.0920           -0.0000j            1.0920           -0.0000
MA.2            0.4877           -0.8948j            1.0191           -0.1706
MA.3            0.4877           +0.8948j            1.0191            0.1706
MA.4           -2.0383           -0.0000j            2.0383           -0.5000
-----------------------------------------------------------------------------
```

In [ ]:
```
When we look at the plot we can see there is a seasonality in data. That is why w
instead of ARIMA.

There are four seasonal elements that are not part of ARIMA that must be configur
P: Seasonal autoregressive order.
D: Seasonal difference order.
Q: Seasonal moving average order.
m: The number of time steps for a single seasonal period.

Now we will try to get best p,d,q,P,D,Q values based on the acf and pacf plots
```

In [284]:
```
#ARIMA(1,1,1)(3,1,1)[6]
model2=SARIMAX(data2,order=(1,1,1), seasonal_order=(3,1,1,6))
results=model.fit()
results.summary()
```

Out[284]:

SARIMAX Results

Dep. Variable:	IPG2211A2N	**No. Observations:**	397
Model:	SARIMAX(1, 1, 1)x(3, 1, 1, 6)	**Log Likelihood**	-925.755
Date:	Mon, 16 May 2022	**AIC**	1865.510
Time:	21:09:04	**BIC**	1893.273
Sample:	01-01-1985	**HQIC**	1876.515
	- 01-01-2018		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	0.5433	0.041	13.167	0.000	0.462	0.624
ma.L1	-0.9645	0.015	-62.294	0.000	-0.995	-0.934
ar.S.L6	-0.2686	0.063	-4.252	0.000	-0.392	-0.145
ar.S.L12	0.2281	0.064	3.568	0.000	0.103	0.353
ar.S.L18	-0.2357	0.051	-4.618	0.000	-0.336	-0.136
ma.S.L6	-0.8015	0.050	-16.004	0.000	-0.900	-0.703
sigma2	6.4779	0.376	17.211	0.000	5.740	7.216

Ljung-Box (L1) (Q):	0.27	**Jarque-Bera (JB):**	36.75
Prob(Q):	0.60	**Prob(JB):**	0.00
Heteroskedasticity (H):	2.75	**Skew:**	0.24
Prob(H) (two-sided):	0.00	**Kurtosis:**	4.43

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

In [ ]:
```
The model summary provides lot of information. The table in the middle is the coe
under 'coef' are the weights of the respective terms.

The P-Value in 'P>|z|' column is highly significant as it should be less than 0.0

So, we will rebuild the model.
```

In [287]:
```
#ARIMAX(4, 1, 4)x(4, 0, [1], 12)
model2=SARIMAX(data2,order=(4,1,4), seasonal_order=(4,0,1,12))
results=model.fit()
results.summary()
```

Out[287]:

SARIMAX Results

Dep. Variable:	IPG2211A2N	**No. Observations:**	397
Model:	SARIMAX(1, 1, 1)x(3, 1, 1, 6)	**Log Likelihood**	-925.755
Date:	Mon, 16 May 2022	**AIC**	1865.510
Time:	21:16:28	**BIC**	1893.273
Sample:	01-01-1985	**HQIC**	1876.515
	- 01-01-2018		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	0.5433	0.041	13.167	0.000	0.462	0.624
ma.L1	-0.9645	0.015	-62.294	0.000	-0.995	-0.934
ar.S.L6	-0.2686	0.063	-4.252	0.000	-0.392	-0.145
ar.S.L12	0.2281	0.064	3.568	0.000	0.103	0.353
ar.S.L18	-0.2357	0.051	-4.618	0.000	-0.336	-0.136
ma.S.L6	-0.8015	0.050	-16.004	0.000	-0.900	-0.703
sigma2	6.4779	0.376	17.211	0.000	5.740	7.216

Ljung-Box (L1) (Q):	0.27	**Jarque-Bera (JB):**	36.75
Prob(Q):	0.60	**Prob(JB):**	0.00
Heteroskedasticity (H):	2.75	**Skew:**	0.24
Prob(H) (two-sided):	0.00	**Kurtosis:**	4.43

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

In [326]:
```
model=SARIMAX(data2,order=(2,1,2),  seasonal_order=(1, 1, 2, 12))
results=model.fit()
```

In [327]:
```
results.summary()
```

Out[327]:

SARIMAX Results

Dep. Variable:	IPG2211A2N	**No. Observations:**	397
Model:	SARIMAX(2, 1, 2)x(1, 1, 2, 12)	**Log Likelihood**	-884.922
Date:	Mon, 16 May 2022	**AIC**	1785.844
Time:	22:14:35	**BIC**	1817.449
Sample:	01-01-1985	**HQIC**	1798.380
	- 01-01-2018		
Covariance Type:	opg		

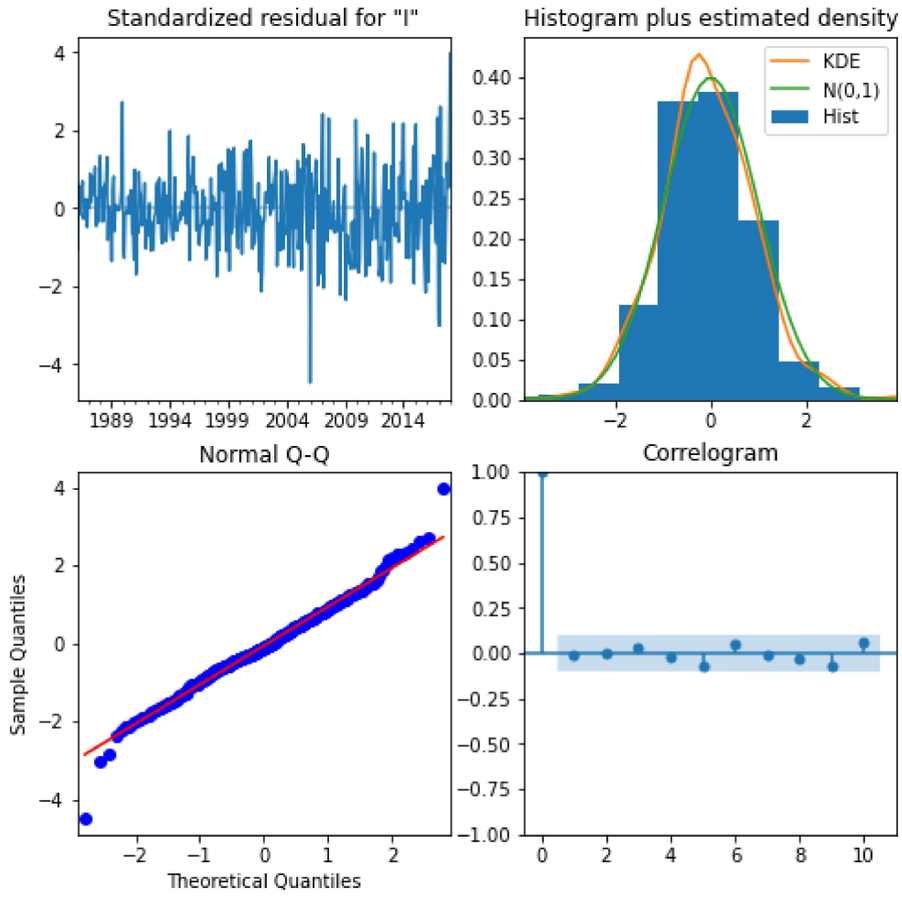
	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	0.0383	0.407	0.094	0.925	-0.759	0.836
ar.L2	0.1850	0.241	0.768	0.443	-0.287	0.657
ma.L1	-0.4249	0.398	-1.069	0.285	-1.204	0.354
ma.L2	-0.4995	0.385	-1.299	0.194	-1.253	0.254
ar.S.L12	-0.4947	0.334	-1.482	0.138	-1.149	0.160
ma.S.L12	-0.1772	0.331	-0.536	0.592	-0.826	0.471
ma.S.L24	-0.4653	0.238	-1.959	0.050	-0.931	0.000
sigma2	5.6905	0.334	17.057	0.000	5.037	6.344

Ljung-Box (L1) (Q):	0.06	**Jarque-Bera (JB):**	28.47
Prob(Q):	0.81	**Prob(JB):**	0.00
Heteroskedasticity (H):	3.01	**Skew:**	0.00
Prob(H) (two-sided):	0.00	**Kurtosis:**	4.33

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

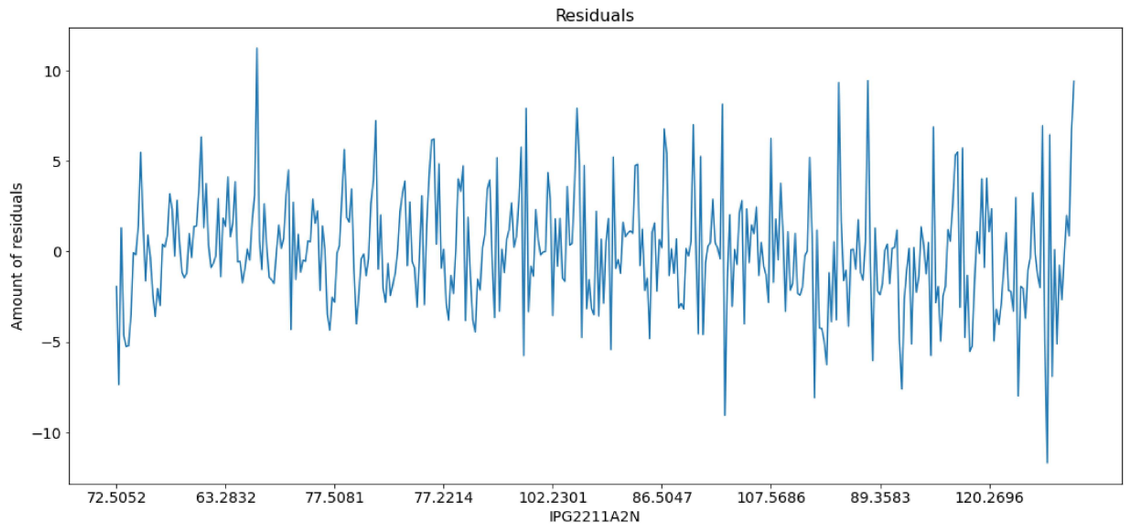
In [ ]:
```
As we can see from above models, Model: SARIMAX(2, 1, 2)x(1, 1, 2, 12) fit the be
AIC value of 1785 and ma4 has less p-value.
```

In [295]:
```
results.plot_diagnostics(figsize=(8,8))
plt.show()
```

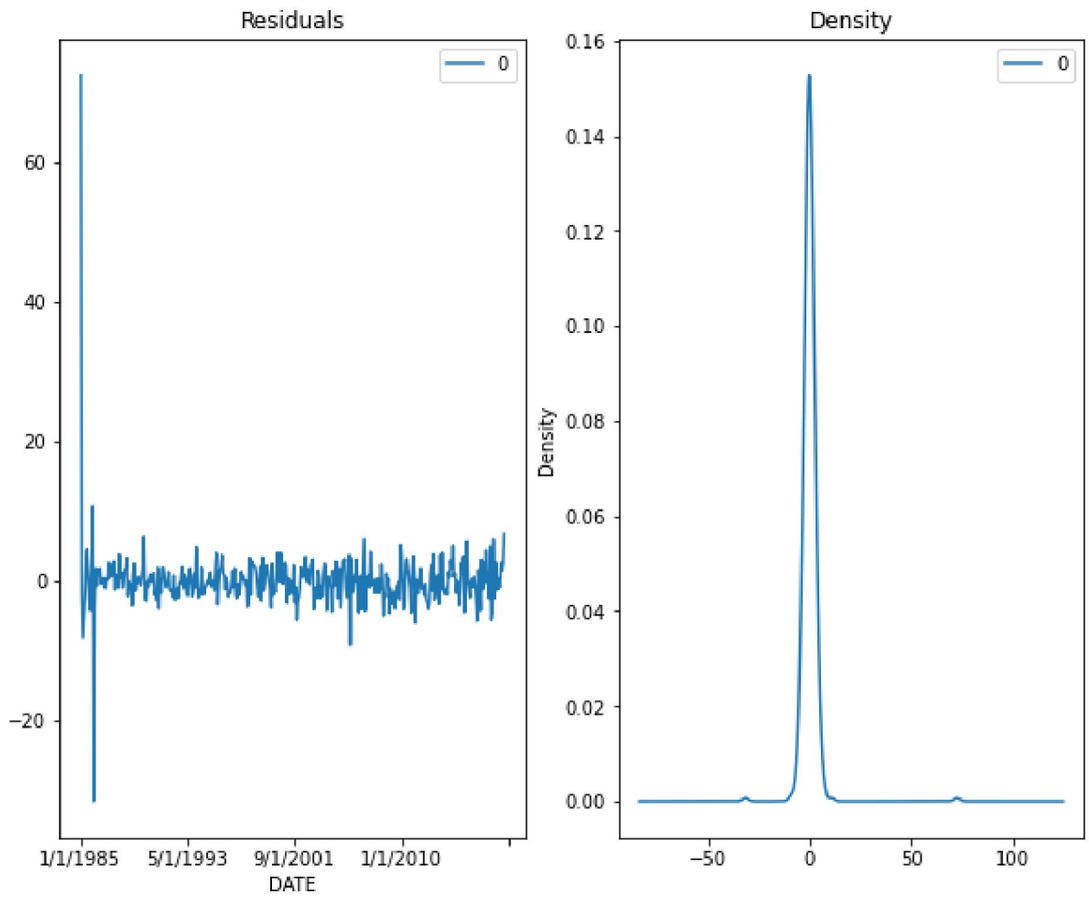


In [ ]:
```
All the 4 plots indicates a good fit of the SARIMA model on the given time serie.
slightly reduced, which is good. The p-values of the AR1 and MA1 terms have impro

next I will plot the residuals to ensure there are no patterns (that is, look for
```

In [379]:
```
# Plotting residual
residuals = pd.DataFrame(model2_fit.resid)
plt.figure(figsize=(18, 8))
plt.plot(residuals)
plt.title('Residuals', fontsize=16)
plt.xlabel("IPG2211A2N", fontsize=14)
plt.ylabel("Amount of residuals", fontsize=14)
plt.xticks(np.arange(0, len(data2.IPG2211A2N)+1, 45), labels=[data2.IPG2211A2N[i]
plt.yticks(fontsize=14)
plt.show()
```



In [381]:
```
residuals = pd.DataFrame(model_fit.resid)
fig, ax = plt.subplots(1,2)
residuals.plot(title="Residuals", ax=ax[0])
residuals.plot(kind='kde', title='Density', ax=ax[1])
plt.show()
```



In [ ]:
```
The residual errors seem fine with near zero mean and uniform variance
```

In [391]:
```
import statsmodels.api as sm
model=sm.tsa.statespace.SARIMAX(train['IPG2211A2N'],order=(4,0,2),seasonal_order=
results=model.fit()
results.summary()
```

Out[391]:

SARIMAX Results

Dep. Variable:	IPG2211A2N	**No. Observations:**	367
Model:	SARIMAX(4, 0, 2)x(1, 1, [1], 12)	**Log Likelihood**	-795.452
Date:	Mon, 16 May 2022	**AIC**	1608.904
Time:	22:58:09	**BIC**	1643.753
Sample:	01-01-1985	**HQIC**	1622.768
	- 07-01-2015		
Covariance Type:	opg		

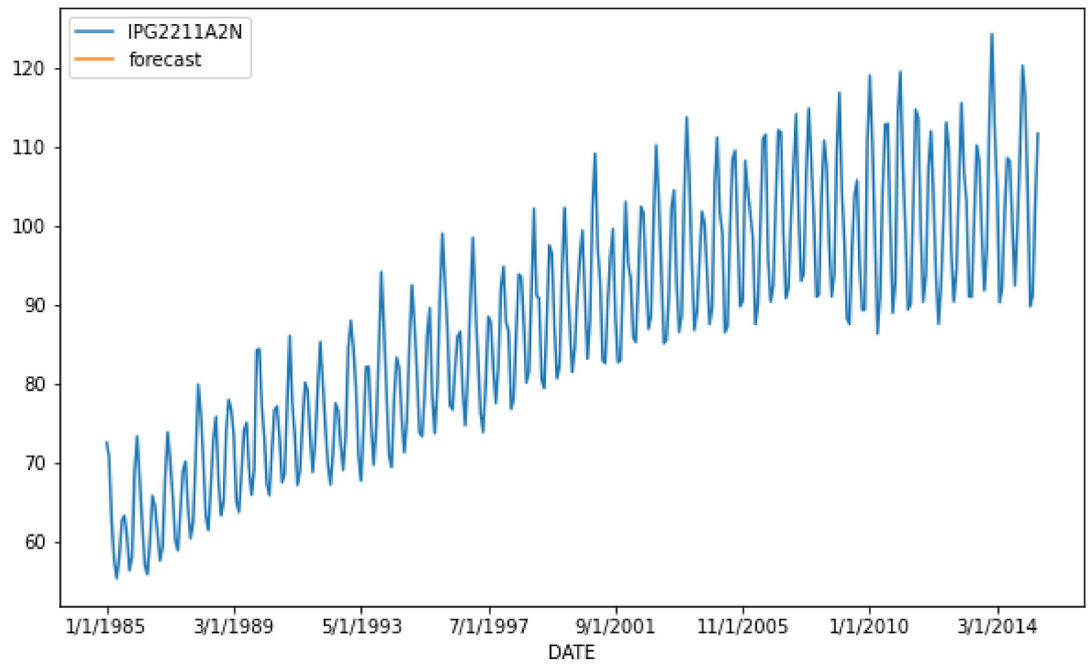
	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	0.5565	0.066	8.442	0.000	0.427	0.686
ar.L2	0.9080	0.084	10.768	0.000	0.743	1.073
ar.L3	-0.5455	0.068	-7.978	0.000	-0.680	-0.412
ar.L4	0.0802	0.057	1.417	0.156	-0.031	0.191
ma.L1	0.0525	0.144	0.364	0.716	-0.230	0.335
ma.L2	-0.9458	0.134	-7.045	0.000	-1.209	-0.683
ar.S.L12	0.0898	0.073	1.235	0.217	-0.053	0.232
ma.S.L12	-0.7655	0.058	-13.118	0.000	-0.880	-0.651
sigma2	5.0095	0.705	7.104	0.000	3.627	6.392

Ljung-Box (L1) (Q):	0.00	**Jarque-Bera (JB):**	20.28
Prob(Q):	0.96	**Prob(JB):**	0.00
Heteroskedasticity (H):	2.67	**Skew:**	-0.15
Prob(H) (two-sided):	0.00	**Kurtosis:**	4.13

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

In [392]:
```
train['forecast']=results.predict(start=300,end=384,dynamic=True)
train[['IPG2211A2N','forecast']].plot(figsize=(10,6))
```

Out[392]: <AxesSubplot:xlabel='DATE'>



In [ ]:
```
Prediction
```

In [344]:
```
train= data2.iloc[:-30]
test= data2.iloc[-30:]
```

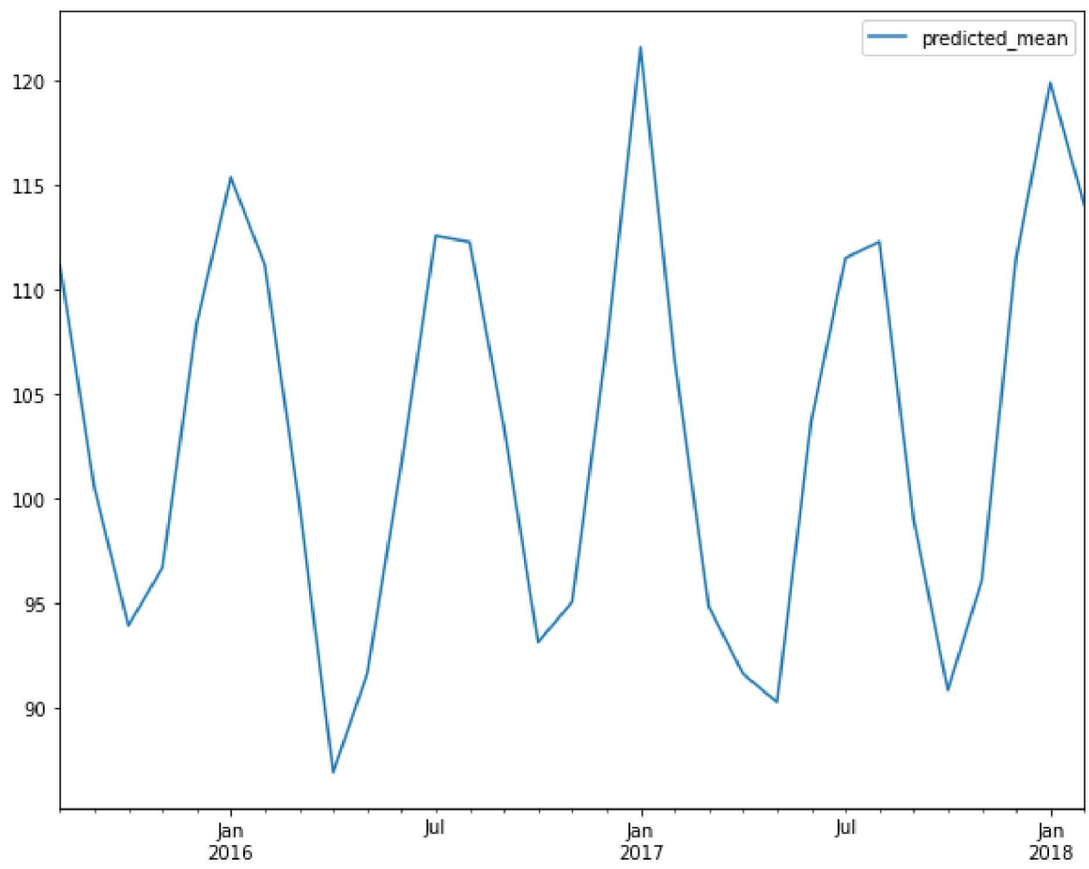
In [360]:
```
#I train the model

model = SARIMAX(data2['IPG2211A2N'],order=(2,1,2),  seasonal_order=(1, 1, 2, 12))
results = model.fit()
```

In [357]:
```
start=len(train)
end=len(train)+len(test)
predictions = results.predict(start=start, end=end, dynamic=False, typ='levels')
print(predictions)
```

```
2015-08-01    111.158086
2015-09-01    100.557282
2015-10-01     93.913472
2015-11-01     96.688830
2015-12-01    108.370986
2016-01-01    115.382063
2016-02-01    111.180910
2016-03-01     99.835402
2016-04-01     86.886892
2016-05-01     91.621486
2016-06-01    101.650420
2016-07-01    112.576346
2016-08-01    112.281792
2016-09-01    103.445344
2016-10-01     93.117555
2016-11-01     95.053003
2016-12-01    107.220903
2017-01-01    121.601176
2017-02-01    106.679798
2017-03-01     94.822281
2017-04-01     91.614827
2017-05-01     90.246322
2017-06-01    103.711778
2017-07-01    111.509670
2017-08-01    112.300009
2017-09-01     98.990602
2017-10-01     90.816413
2017-11-01     96.092912
2017-12-01    111.491693
2018-01-01    119.919912
2018-02-01    114.031149
Freq: MS, Name: predicted_mean, dtype: float64
```

In [358]:
```
predictions.plot(legend=True)
```

Out[358]: <AxesSubplot:>



In [ ]: