In [85]:
```
import numpy as np
import pandas as pd
df = pd.read_csv("Electric_Production.csv")
df
```

Out[85]:

	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 × 2 columns

In [86]:
```
df.isnull().sum()
```

Out[86]:
```
DATE          0
IPG2211A2N    0
dtype: int64
```

In [87]:
```
df["Value"] = df["IPG2211A2N"]
df = df.drop("IPG2211A2N",axis=1)
df.shape
```

Out[87]: (397, 2)

In [88]:
```
df["DATE"] = pd.to_datetime(df["DATE"])
df.set_index("DATE",inplace=True)
df
```

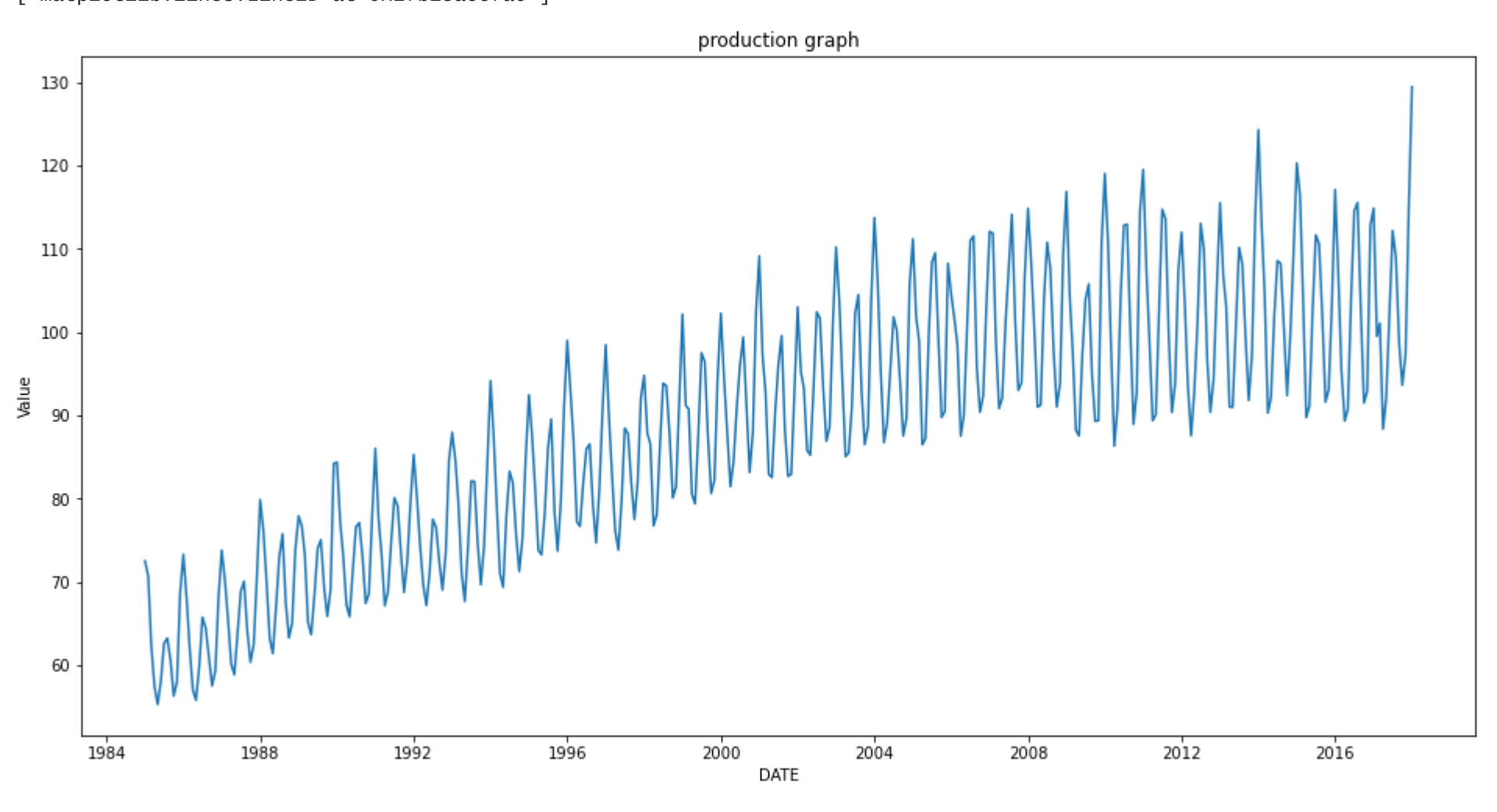
Out[88]:

DATE	Value
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

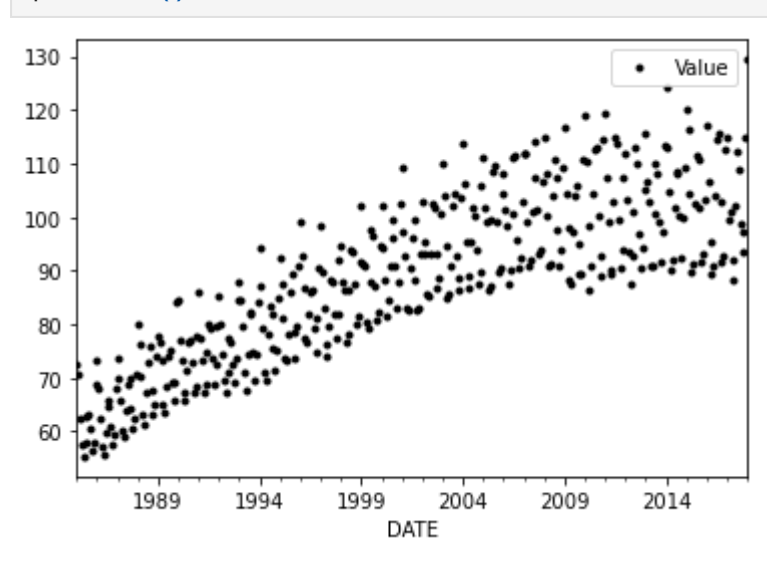
397 rows × 1 columns

In [9]:
```
import matplotlib.pyplot as plt
plt.figure(figsize=(16,8))
plt.xlabel("DATE")
plt.ylabel("Value")
plt.title("production graph")
plt.plot(df)
```

Out[9]: [<matplotlib.lines.Line2D at 0x1fb15acefa0>]

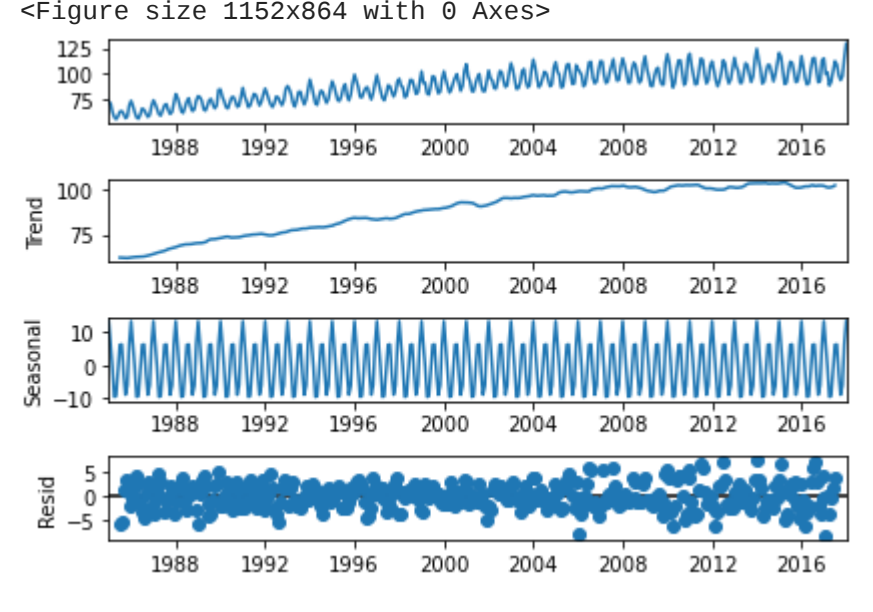


In [10]:
```
df.plot(style='k.')
plt.show()
```



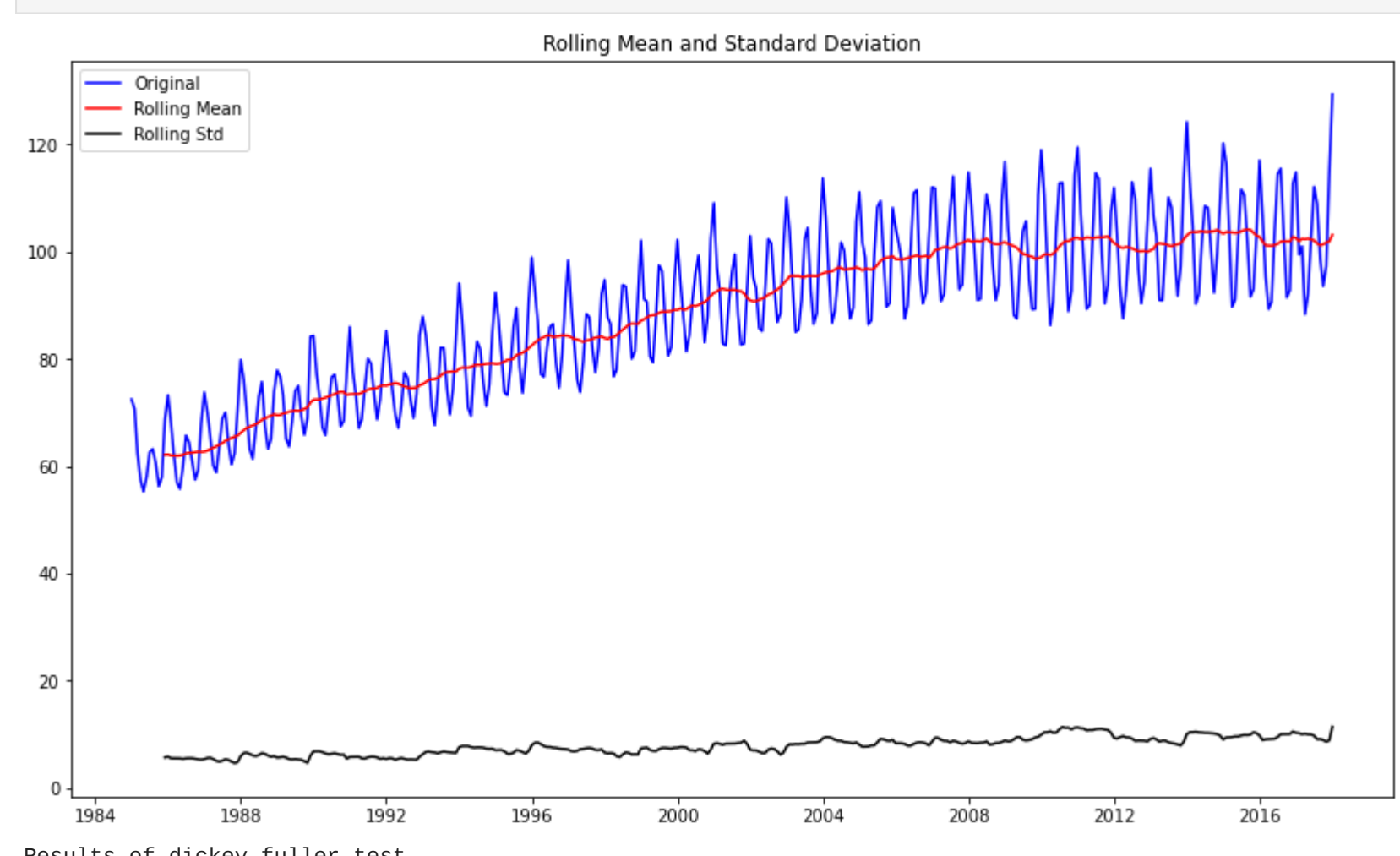
In [14]:
```
from statsmodels.tsa.seasonal import seasonal_decompose
result = seasonal_decompose(df)
result.plot()
plt.show()
```

<Figure size 1152x864 with 0 Axes>



In [28]:
```
 #perform dickey fuller test
from statsmodels.tsa.stattools import adfuller
def adfuller_test(timeseries):
    #Determing rolling statistics(mean and std)
    rolmean = timeseries.rolling(12).mean()
    rolstd = timeseries.rolling(12).std()
    #Plot rolling statistics:
    plt.figure(figsize=(14,8))
    plt.plot(timeseries, color='blue',label='Original')
    plt.plot(rolmean, color='red', label='Rolling Mean')
    plt.plot(rolstd, color='black', label = 'Rolling Std')
    plt.legend(loc='best')
    plt.title('Rolling Mean and Standard Deviation')
    plt.show(block=False)
    print("Results of dickey fuller test")
    adft = adfuller(timeseries,autolag='AIC')
    # output for dft will give us without defining what the values are.
    #hence we manually write what values does it explains using a for loop
    output = pd.Series(adft[0:4],index=['Test Statistics','p-value','No. of lags used','Number of observations used'])
    for key,values in adft[4].items():
        output['critical value (%s)'%key] =  values
    print(output)
```

In [29]:
```
adfuller_test(df["Value"])
```




```
Results of dickey fuller test
Test Statistics                 -2.256990
p-value                          0.186215
No. of lags used                15.000000
Number of observations used    381.000000
critical value (1%)             -3.447631
critical value (5%)             -2.869156
critical value (10%)            -2.570827
dtype: float64
```

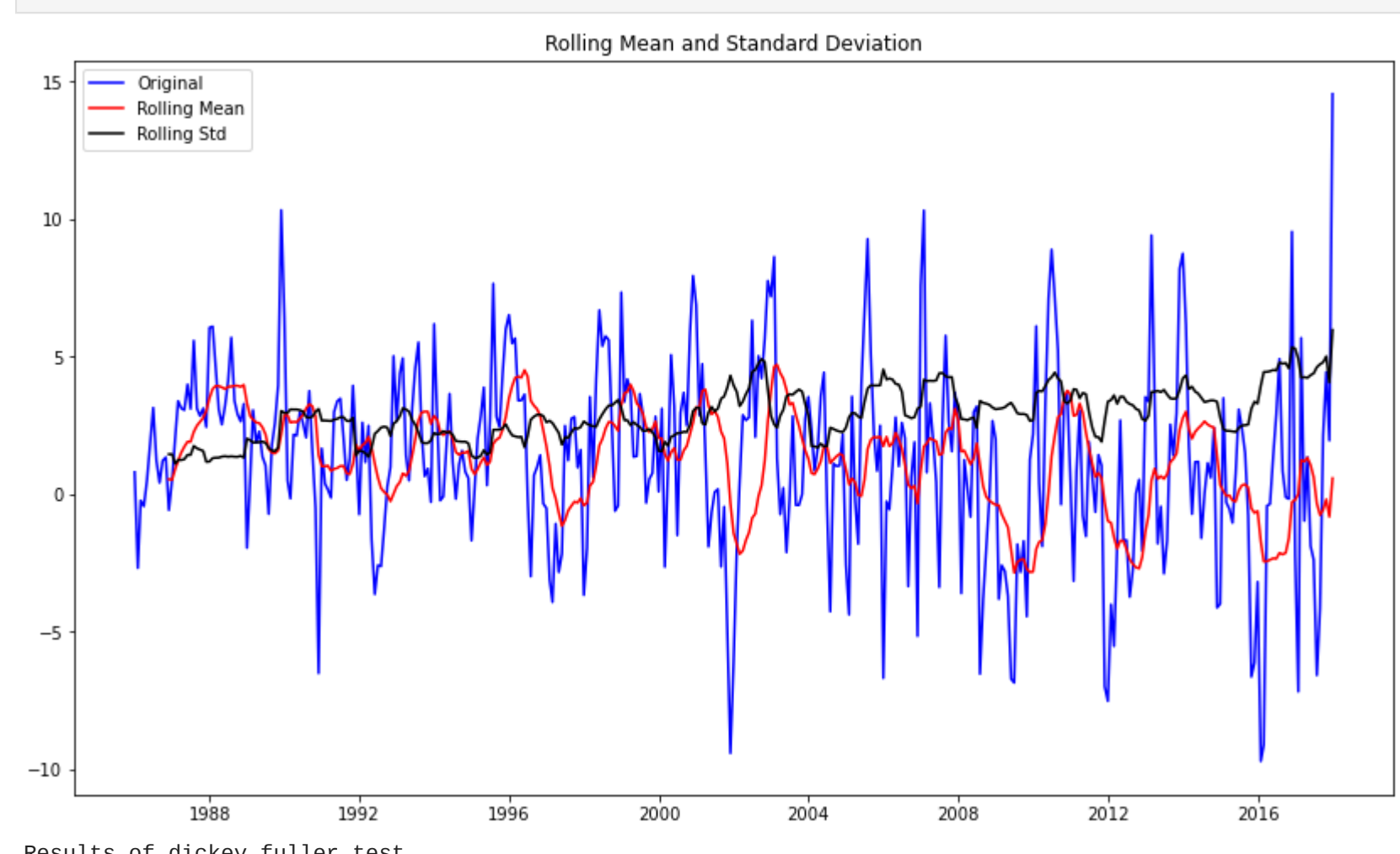
In [30]:
```
df["First_seasonal_diff"] = df["Value"]-df["Value"].shift(12)
df
```

Out[30]:

DATE	Value	First_seasonal_diff
1985-01-01	72.5052	NaN
1985-02-01	70.6720	NaN
1985-03-01	62.4502	NaN
1985-04-01	57.4714	NaN
1985-05-01	55.3151	NaN
...	...	...
2017-09-01	98.6154	-4.1483
2017-10-01	93.6137	2.1270
2017-11-01	97.3359	4.4459
2017-12-01	114.7212	1.9518
2018-01-01	129.4048	14.5543

397 rows × 2 columns

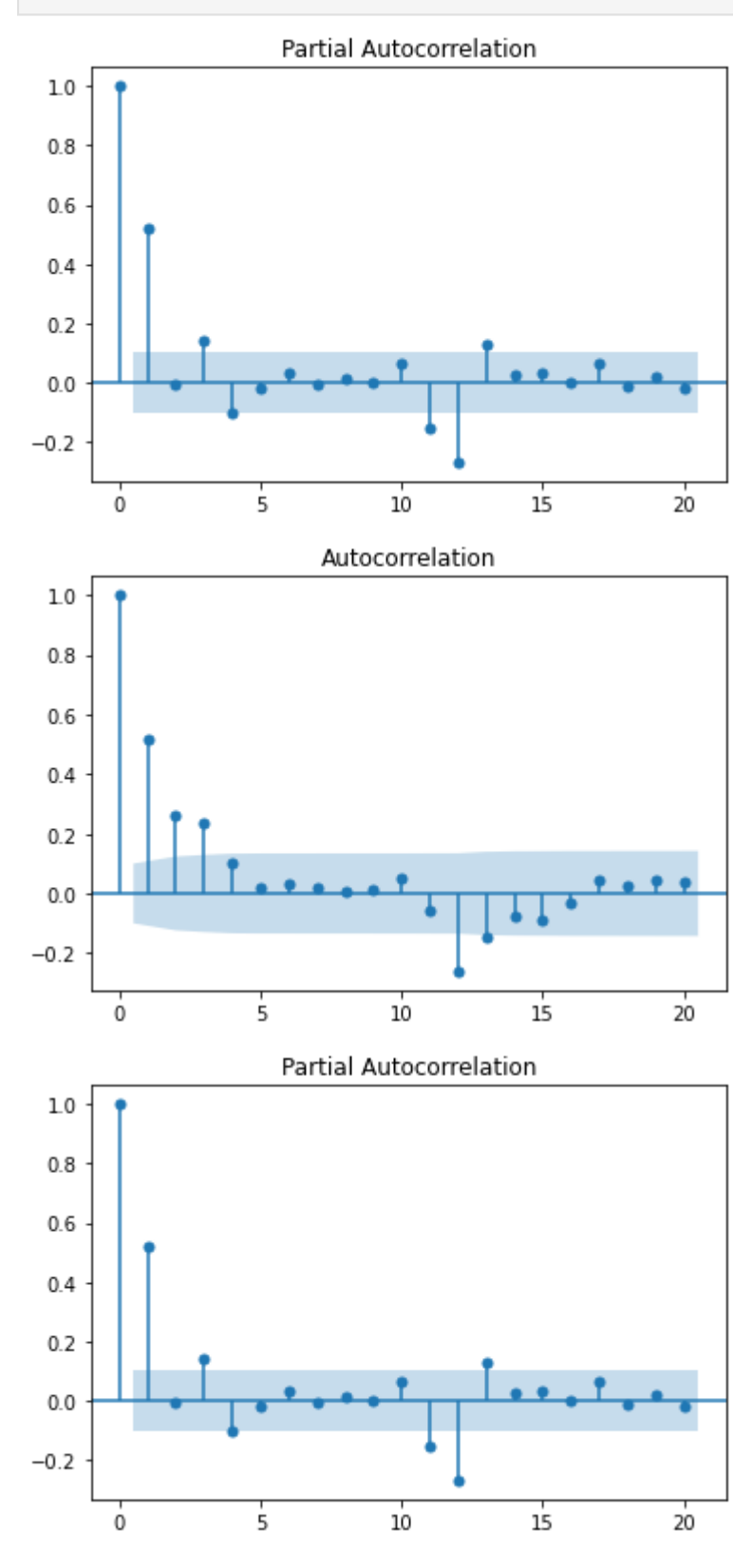
In [31]:
```
adfuller_test(df["First_seasonal_diff"].dropna())
```




```
Results of dickey fuller test
Test Statistics               -5.673482e+00
p-value                        8.812645e-07
No. of lags used               1.200000e+01
Number of observations used    3.720000e+02
critical value (1%)           -3.448052e+00
critical value (5%)           -2.869341e+00
critical value (10%)          -2.570926e+00
dtype: float64
```

In [32]:
```
from statsmodels.graphics.tsaplots import plot_acf,plot_pacf
plot_acf(df["First_seasonal_diff"].iloc[13:],lags=20)
plot_pacf(df["First_seasonal_diff"].iloc[13:],lags=20)
```

Out[32]:



In [62]:
```
df_stationay = df.drop(["Value"],axis=1)
df_stationay = df_stationay.dropna()
df_stationay
```

Out[62]:

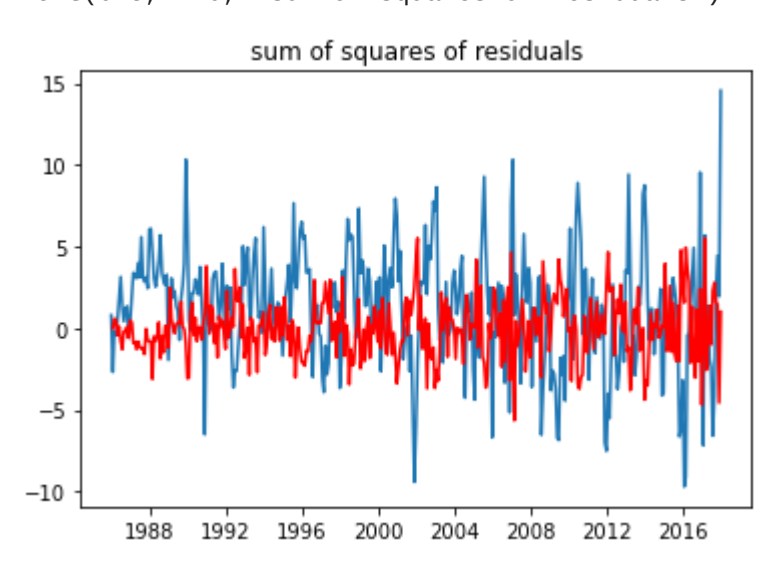
DATE	First_seasonal_diff
1986-01-01	0.8005
1986-02-01	-2.6851
1986-03-01	-0.2281
1986-04-01	-0.4385
1986-05-01	0.4986
...	...
2017-09-01	-4.1483
2017-10-01	2.1270
2017-11-01	4.4459
2017-12-01	1.9518
2018-01-01	14.5543

385 rows × 1 columns

In [64]:
```
from statsmodels.tsa.arima_model import ARIMA
model = ARIMA(df_stationay, order=(3,1,3))
result_AR = model.fit(disp = 0)
plt.plot(df_stationay)
plt.plot(result_AR.fittedvalues, color='red')
plt.title("sum of squares of residuals")
```

```
C:\Users\U.R Computer\anaconda\lib\site-packages\statsmodels\tsa\base\tsa_model.py:524: ValueWarning: No frequency information was provided, so inferred frequ
ency MS will be used.
  warnings.warn('No frequency information was'
C:\Users\U.R Computer\anaconda\lib\site-packages\statsmodels\tsa\base\tsa_model.py:524: ValueWarning: No frequency information was provided, so inferred frequ
ency MS will be used.
  warnings.warn('No frequency information was'
C:\Users\U.R Computer\anaconda\lib\site-packages\statsmodels\base\model.py:547: HessianInversionWarning: Inverting hessian failed, no bse or cov_params availa
ble
  warnings.warn('Inverting hessian failed, no bse or cov_params '
C:\Users\U.R Computer\anaconda\lib\site-packages\statsmodels\tsa\arima_model.py:472: FutureWarning:
statsmodels.tsa.arima_model.ARMA and statsmodels.tsa.arima_model.ARIMA have
been deprecated in favor of statsmodels.tsa.arima.model.ARIMA (note the .
between arima and model) and
statsmodels.tsa.SARIMAX. These will be removed after the 0.12 release.

statsmodels.tsa.arima.model.ARIMA makes use of the statespace framework and
is both well tested and maintained.

To silence this warning and continue using ARMA and ARIMA until they are
removed, use:

import warnings
warnings.filterwarnings('ignore', 'statsmodels.tsa.arima_model.ARMA',
                        FutureWarning)
warnings.filterwarnings('ignore', 'statsmodels.tsa.arima_model.ARIMA',
                        FutureWarning)

  warnings.warn(ARIMA_DEPRECATION_WARN, FutureWarning)
```

Out[64]: Text(0.5, 1.0, 'sum of squares of residuals')



In [65]:
```
result_AR.summary()
```

Out[65]:

ARIMA Model Results

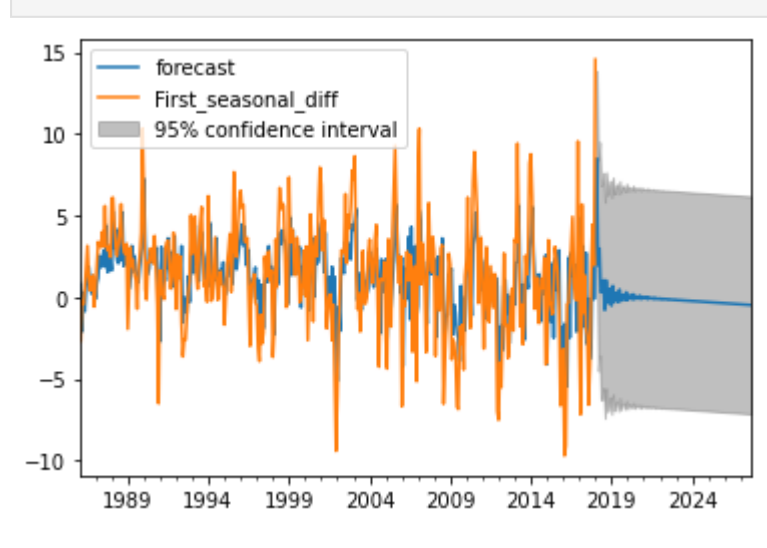
Dep. Variable:	D.First_seasonal_diff	No. Observations:	384
Model:	ARIMA(3, 1, 3)	Log Likelihood	-938.531
Method:	css-mle	S.D. of innovations	2.747
Date:	Fri, 07 May 2021	AIC	1893.063
Time:	17:30:18	BIC	1924.668
Sample:	02-01-1986	HQIC	1905.599
	- 01-01-2018		

	coef	std err	z	P>|z|	[0.025	0.975]
const	-0.0056	0.003	-2.127	0.033	-0.011	-0.000
ar.L1.D.First_seasonal_diff	-1.0432	0.048	-21.518	0.000	-1.138	-0.948
ar.L2.D.First_seasonal_diff	-0.1272	0.074	-1.725	0.085	-0.272	0.017
ar.L3.D.First_seasonal_diff	0.4004	0.053	7.611	0.000	0.297	0.504
ma.L1.D.First_seasonal_diff	0.7055	0.020	34.759	0.000	0.666	0.745
ma.L2.D.First_seasonal_diff	-0.7056	0.016	-44.604	0.000	-0.737	-0.675
ma.L3.D.First_seasonal_diff	-0.9999	0.020	-49.548	0.000	-1.039	-0.960

Roots

	Real	Imaginary	Modulus	Frequency
AR.1	-0.8970	-0.6150j	1.0875	-0.4043
AR.2	-0.8970	+0.6150j	1.0875	0.4043
AR.3	2.1116	-0.0000j	2.1116	-0.0000
MA.1	1.0000	-0.0000j	1.0000	-0.0000
MA.2	-0.8528	-0.5223j	1.0000	-0.4125
MA.3	-0.8528	+0.5223j	1.0000	0.4125

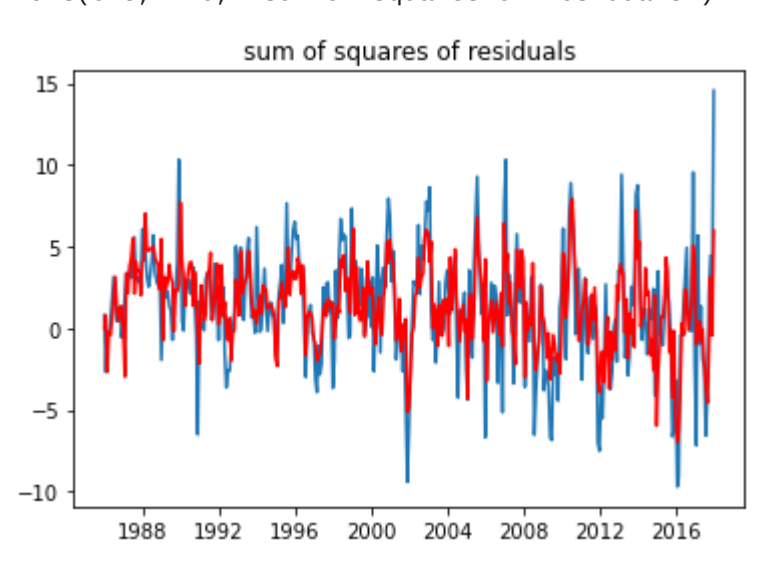
In [66]:
```
result_AR.plot_predict(1,500)
x=result_AR.forecast(steps=200)
```



In [46]:
```
import statsmodels.api as sm
model = sm.tsa.statespace.SARIMAX(df_stationay, order=(3,1,2),seasonal_order=(3,1,2,12))
result_SAR = model.fit(disp = 0)
plt.plot(df_stationay)
plt.plot(result_SAR.fittedvalues, color='red')
plt.title("sum of squares of residuals")
```

```
C:\Users\U.R Computer\anaconda\lib\site-packages\statsmodels\tsa\base\tsa_model.py:524: ValueWarning: No frequency information was provided, so inferred frequ
ency MS will be used.
  warnings.warn('No frequency information was'
C:\Users\U.R Computer\anaconda\lib\site-packages\statsmodels\tsa\base\tsa_model.py:524: ValueWarning: No frequency information was provided, so inferred frequ
ency MS will be used.
  warnings.warn('No frequency information was'
C:\Users\U.R Computer\anaconda\lib\site-packages\statsmodels\tsa\statespace\sarimax.py:965: UserWarning: Non-stationary starting autoregressive parameters fou
nd. Using zeros as starting parameters.
  warn('Non-stationary starting autoregressive parameters'
C:\Users\U.R Computer\anaconda\lib\site-packages\statsmodels\tsa\statespace\sarimax.py:977: UserWarning: Non-invertible starting MA parameters found. Using ze
ros as starting parameters.
  warn('Non-invertible starting MA parameters found.'
C:\Users\U.R Computer\anaconda\lib\site-packages\statsmodels\base\model.py:566: ConvergenceWarning: Maximum Likelihood optimization failed to converge. Check
mle_retvals
  warnings.warn("Maximum Likelihood optimization failed to "
```

Out[46]: Text(0.5, 1.0, 'sum of squares of residuals')



In [77]:
```
df_stationay["Forecast"] = result_SAR.predict(start=100,end=400)
df_stationay[["First_seasonal_diff","Forecast_1"]].plot(figsize=(16,8))
```

Out[77]: <AxesSubplot:xlabel='DATE'>