In [129]:

```
# Import relevant libraries
import pandas as pd
import numpy as np
import statsmodels.api as sm
# Time Series Analysis Library: AR,VAR,ARMA; ACF&PACF
import statsmodels.tsa as tsa
# Import acf & pacf plotting functions
from statsmodels.graphics.tsaplots import plot_acf
from statsmodels.graphics.tsaplots import plot_pacf
# Plotting Library
import matplotlib
import matplotlib.pyplot as plt
# Use the lzip to pretty-print short descriptions for the test returns
from statsmodels.compat import lzip
# Ljung-Box test function
from statsmodels.stats.diagnostic import acorr_ljungbox
```

In [130]:

```
pip install arch
```

```
Requirement already satisfied: arch in c:\users\35387\anaconda3\lib\site-pac
kages (4.15)
Requirement already satisfied: pandas>=0.23 in c:\users\35387\anaconda3\lib
\site-packages (from arch) (1.0.5)
Note: you may need to restart the kernel to use updated packages.
Requirement already satisfied: statsmodels>=0.9 in c:\users\35387\anaconda3
\lib\site-packages (from arch) (0.11.1)
Requirement already satisfied: numpy>=1.14 in c:\users\35387\anaconda3\lib\s
ite-packages (from arch) (1.18.5)
Requirement already satisfied: cython>=0.29.14 in c:\users\35387\anaconda3\l
ib\site-packages (from arch) (0.29.21)
Requirement already satisfied: scipy>=1.0.1 in c:\users\35387\anaconda3\lib
\site-packages (from arch) (1.5.0)
Requirement already satisfied: property-cached>=1.6.3 in c:\users\35387\anac
onda3\lib\site-packages (from arch) (1.6.4)
Requirement already satisfied: pytz>=2017.2 in c:\users\35387\anaconda3\lib
\site-packages (from pandas>=0.23->arch) (2020.1)
Requirement already satisfied: python-dateutil>=2.6.1 in c:\users\35387\anac
onda3\lib\site-packages (from pandas>=0.23->arch) (2.8.1)
Requirement already satisfied: patsy>=0.5 in c:\users\35387\anaconda3\lib\si
te-packages (from statsmodels>=0.9->arch) (0.5.1)
Requirement already satisfied: six>=1.5 in c:\users\35387\anaconda3\lib\site
-packages (from python-dateutil>=2.6.1->pandas>=0.23->arch) (1.15.0)
```

In [131]:

```
# Check auxiliary regression for DF test
from arch.unitroot import ADF
```

In [132]:

```
data = pd.read_csv('GDPC1.csv')
```

In [133]:

```
# 1. Rename the columns
data.rename(columns={'DATE': 'Date','GDPC1': 'Real GDP'},inplace=True)
```

In [134]:

```
# 2. Setting the Date column of strings to the datetime format in Pandas:
data['Date'] = pd.to_datetime(data['Date'],format='%Y/%m/%d')
```

In [135]:

```
# 3. Set the Date column as Index of the dataframe
data.set_index('Date',inplace=True)
```

In [136]:

```
data.info()
```

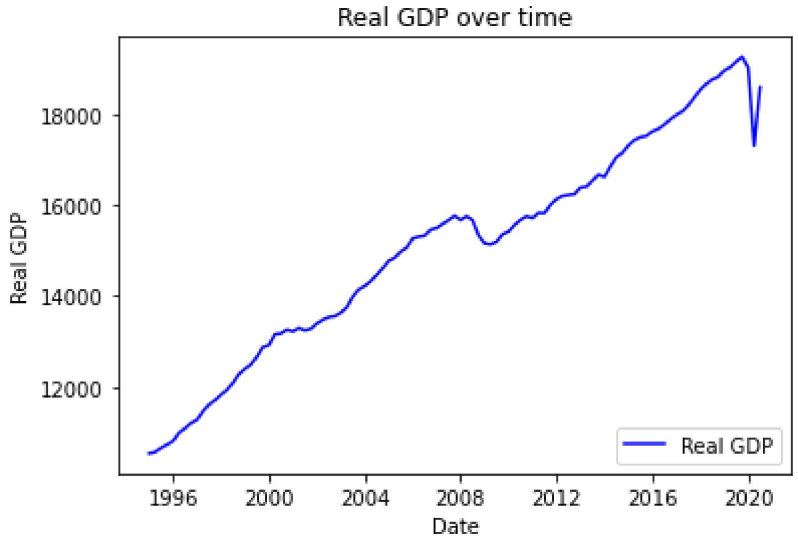
```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 103 entries, 1995-01-01 to 2020-07-01
Data columns (total 1 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   Real GDP  103 non-null    float64
dtypes: float64(1)
memory usage: 1.6 KB
```

In [137]:

```
#1.1 Graphical Detection of Stationarity
```

In [138]:

```
#Plotting the original series against the time to detect stationarity


# Plotting the CPI series against the time index of dataframe
plt.plot(data,'b-',label='Real GDP')
# Set the labels for axies
plt.xlabel('Date')
plt.ylabel('Real GDP')
# Set the title for graph
plt.title('Real GDP over time')
# Set the legend position
plt.legend(loc='lower right')
# Save the figure
plt.savefig('Real GDP Plot.jpg',bbox_inches='tight')
# Disply the figure
plt.show()
```



In [139]:

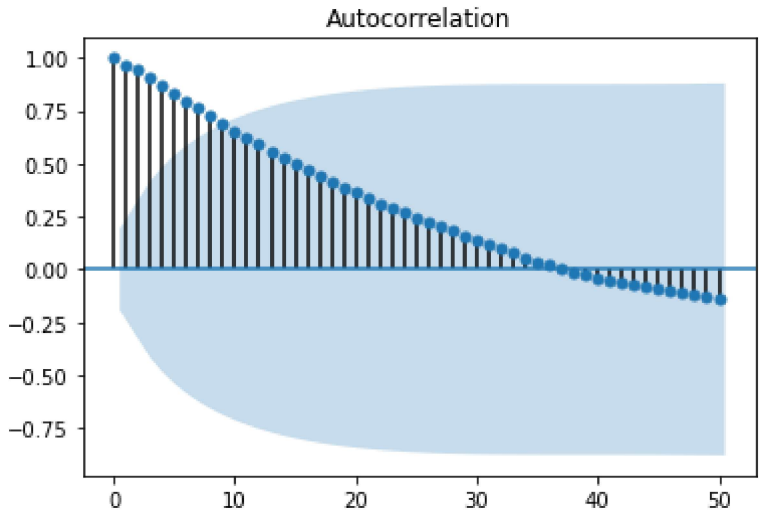
```
#Correlogram Plotting
```

In [140]:

```
#Applying the plot_acf() and plot_pacf()functions

#Check whether the series is stationary or non-stationary:
    #Stationary: Absence of correlation
    #Non-stationary: Gradually declining correlogram (spurious correlation due to common tr
```

In [227]:

```
plot1 = plot_acf(data, lags=50)
plt.show()
```

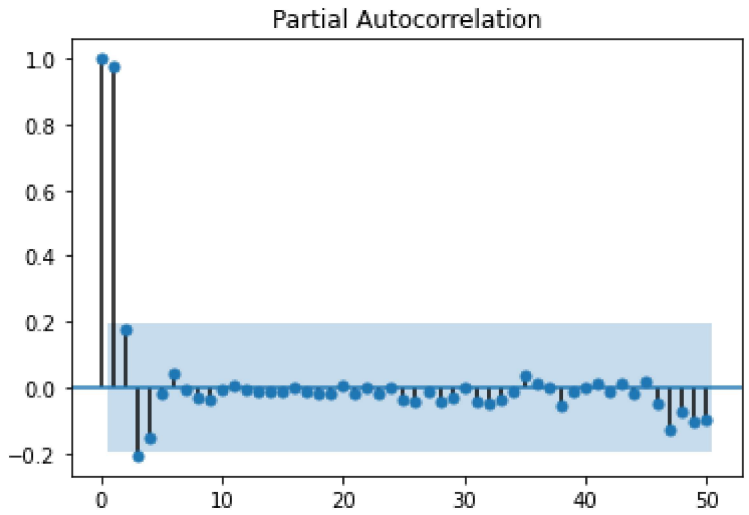


In [142]:

```
#We know this series non stationary as the AFC is slowly decreasing, implying a spurious co
#the empirical correlogram is useless in identifying the know stationary process.
```

In [226]:

```
# Plot the Pacf (CPI)
plot2 = plot_pacf(data, lags=50)
```



In [144]:

```
#1.2 Unit Roots Testing: Augmented Dickey-Fuller Test

#There are 3 verisons, however, all have the same null hypothesis that the series has a uni
```

In [145]:

```
#1st Dickey-Fuller test for a random walk against a stationary autoregressive process of or

#H0 = Series has a unit root
#H1 = Series is stationary AR(1)

#Test stats not following the usual t-distribution of under the null: should compare it wit
```

In [146]:

```
# 1. Dickey-Fuller Test without a constant and trend
adf_1_name = ['Dickey-Fuller Test Statistic', 'p-value','Number of lags used','Number of ob
                'Critical values','maximized information criterion']
# Set regression parameter as 'nc': no constant, no trend
# Number of lags is chosen to minimize the corresponding information criterion(AIC by defau
adf_1 = sm.tsa.stattools.adfuller(data['Real GDP'],regression='nc')
lzip(adf_1_name, adf_1)

#nc = alternative hypothesis
```

Out[146]:

```
[('Dickey-Fuller Test Statistic', 3.9359168647015883),
 ('p-value', 0.9999998246268814),
 ('Number of lags used', 1),
 ('Number of observations', 101),
 ('Critical values',
  {'1%': -2.5882321870404863,
   '5%': -1.9439589708250309,
   '10%': -1.614431731329}),
 ('maximized information criterion', 1230.296276131842)]
```

In [147]:

```
#p value really high so we cannot reject the null hypothesis
#thus we can say that this time series has a unit root and non stationary
```

In [148]:

```
#2. Dickey-Fuller test for a random walk against a stationary AR(1) with drift

# c = H1:

# 2. Dickey-Fuller Test with a constant
adf_2_name = ['Dickey-Fuller Test Statistic', 'p-value','Number of lags used','Number of ob
              'Critical values','maximized information criterion']
# Set regression parameter as 'c': constant only (default)
# Number of lags is chosen to minimize the corresponding information criterion(AIC by defau
adf_2 = sm.tsa.stattools.adfuller(data['Real GDP'],regression='c')
lzip(adf_2_name, adf_2)
```

Out[148]:

```
[('Dickey-Fuller Test Statistic', -1.7828088500834927),
 ('p-value', 0.38902940220094234),
 ('Number of lags used', 1),
 ('Number of observations', 101),
 ('Critical values',
  {'1%': -3.4968181663902103,
   '5%': -2.8906107514600103,
   '10%': -2.5822770483285953}),
 ('maximized information criterion', 1228.1991276243475)]
```

In [149]:

```
#Again the results show that we can reject the Null Hypthothesis but not with as much certa
#Series has a unit root.
```

In [150]:

```
# 3. Dickey-Fuller Test with a constant and trend

adf_3_name = ['Dickey-Fuller Test Statistic', 'p-value','Number of lags used','Number of ob
              'Critical values','maximized information criterion']
# Set regression parameter as 'ct': constant and trend
adf_3 = sm.tsa.stattools.adfuller(data['Real GDP'],regression='ct')
lzip(adf_3_name, adf_3)
```

Out[150]:

```
[('Dickey-Fuller Test Statistic', -1.9395885077225477),
 ('p-value', 0.6339163353741536),
 ('Number of lags used', 1),
 ('Number of observations', 101),
 ('Critical values',
  {'1%': -4.051321648595896,
   '5%': -3.454889141998309,
   '10%': -3.1530564880069027}),
 ('maximized information criterion', 1226.0854437315197)]
```

In [151]:

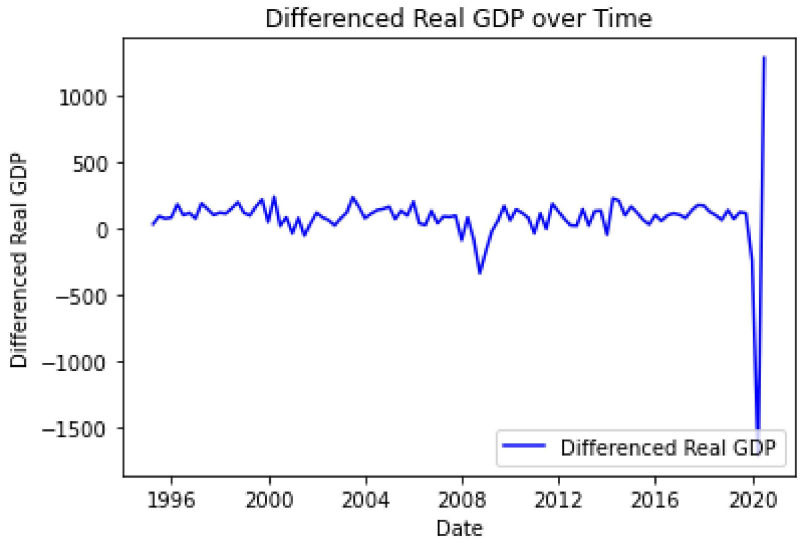
```
#Again the results show that we can reject the Null Hypthothesis but not with as much certa
#Series has a unit root.

#Conclusion: We have a non-stationary time series.

#Now we convert it to a stationary time sereies in order to fit an ARMA model to it.
```

In [152]:

```
# Take the first difference
diff_data = data.diff()
# Drop NA
diff_data.dropna(inplace=True)
```

In [153]:

```
# Plotting the CPI series against the time index of dataframe
plt.plot(diff_data,'b-',label='Differenced Real GDP')
# Set the labels for axies
plt.xlabel('Date')
plt.ylabel('Differenced Real GDP')
# Set the title for graph
plt.title('Differenced Real GDP over Time')
# Set the legend position
plt.legend(loc='lower right')
# Save the figure
plt.savefig('Differenced Real GDP Plot.jpg',bbox_inches='tight')
# Display the figure
plt.show()
```



In [154]:

```
adf_dt_name = ['Dickey-Fuller Test Statistic', 'p-value','Number of lags used','Number of o
               'Critical values','maximized information criterion']
# DF test for diff_data: differenced CPI
adf_dt = sm.tsa.stattools.adfuller(diff_data['Real GDP'])
lzip(adf_dt_name, adf_dt)
```

Out[154]:

```
[('Dickey-Fuller Test Statistic', -11.878973406830053),
 ('p-value', 6.242922235256828e-22),
 ('Number of lags used', 0),
 ('Number of observations', 101),
 ('Critical values',
  {'1%': -3.4968181663902103,
   '5%': -2.8906107514600103,
   '10%': -2.5822770483285953}),
 ('maximized information criterion', 1215.7726620333765)]
```

In [155]:

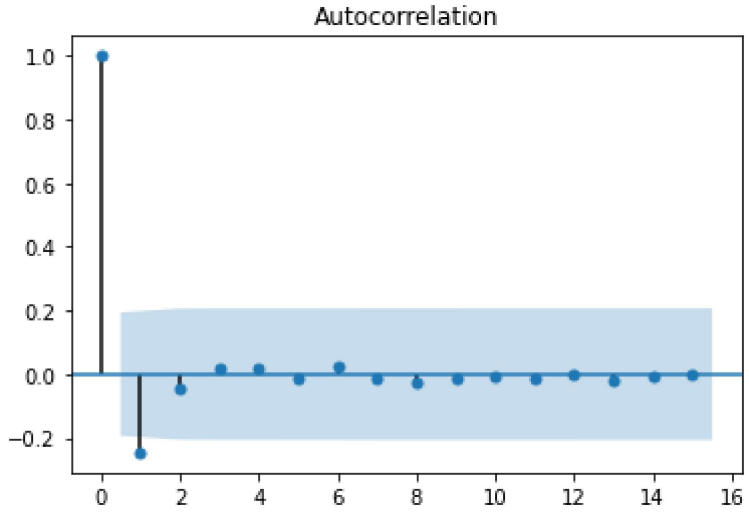
```
# DF Test statistic is negative and high and p value is basically zero so we reject the H0
# Time series is does not have a unit rate and therefore is stationary.
```

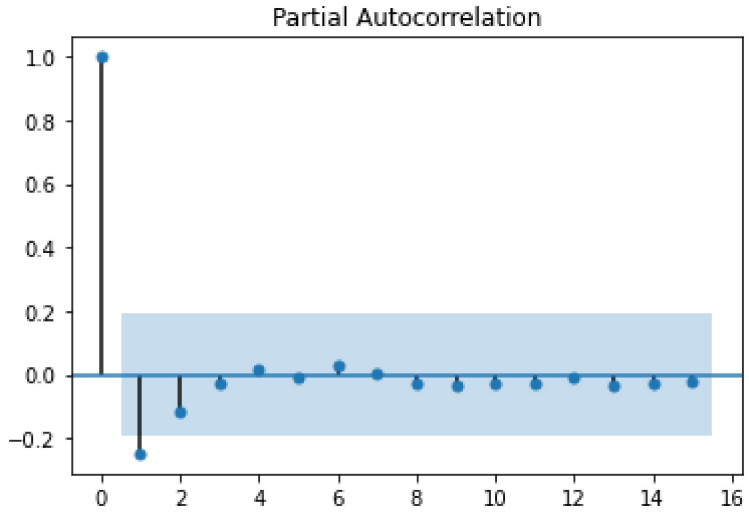
In [156]:

```
#ARMA MODEL ESTIMATION
```

In [193]:

```
plot_acf(diff_data, lags=15)
plot_pacf(diff_data, lags=15)
plt.show()
```





In [194]:

```
#There is one lag that is significantly different from zero. So, this would define the orde
#in this case if would be of order 1.

#There is 1 relevent lag at which the PACF is significantly different from zero. The 1st la
#for the purpose of completing the process of comaring ARMA models.

#Thus we have 3 possible models to compare

#ARMA(1,1)
#ARMA(2,1)
#ARMA(7,1)
```

In [160]:

```
#ARMA (1,1)
```

In [189]:

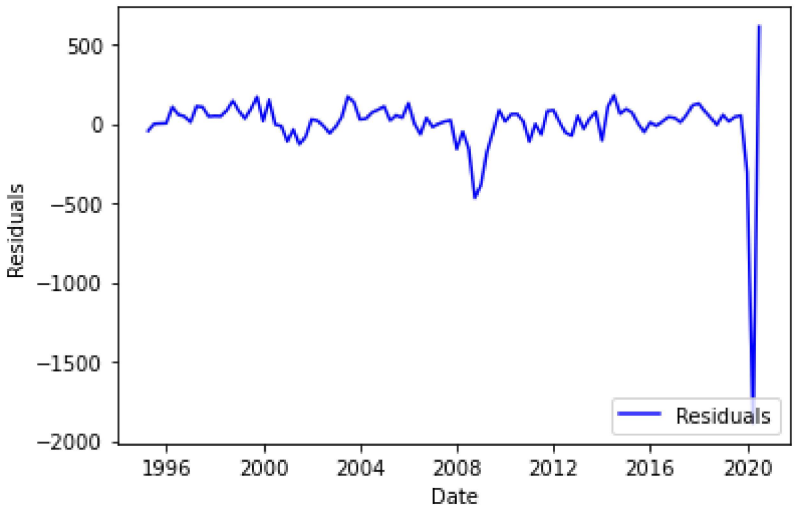
```
# 1. Check model estimates
arma1_1 = tsa.arima_model.ARMA(diff_data,order=(1,1)).fit()
print(arma1_1.summary())
```

```
                              ARMA Model Results
==============================================================================
==
Dep. Variable:                Real GDP   No. Observations:                   1
02
Model:                     ARMA(1, 1)   Log Likelihood                -695.4
34
Method:                        css-mle   S.D. of innovations            221.0
65
Date:                Sun, 08 Nov 2020   AIC                           1398.8
68
Time:                        11:23:17   BIC                           1409.3
68
Sample:                    04-01-1995   HQIC                          1403.1
19
                         - 07-01-2020
==============================================================================
======
                     coef    std err          z      P>|z|      [0.025
0.975]
------------------------------------------------------------------------------
------
const              75.9431     16.595      4.576      0.000      43.418     1
08.468
ar.L1.Real GDP     -0.3505      0.574     -0.610      0.542      -1.476
0.775
ma.L1.Real GDP      0.0171      0.530      0.032      0.974      -1.022
1.057
                                    Roots
==============================================================================
=
                  Real          Imaginary           Modulus         Frequenc
y
------------------------------------------------------------------------------
-
AR.1           -2.8532           +0.0000j            2.8532            0.500
0
MA.1          -58.6001           +0.0000j           58.6001            0.500
0
------------------------------------------------------------------------------
-
```

```
C:\Users\35387\anaconda3\lib\site-packages\statsmodels\tsa\base\tsa_model.p
y:159: ValueWarning: No frequency information was provided, so inferred freq
uency QS-OCT will be used.
  warnings.warn('No frequency information was'
```

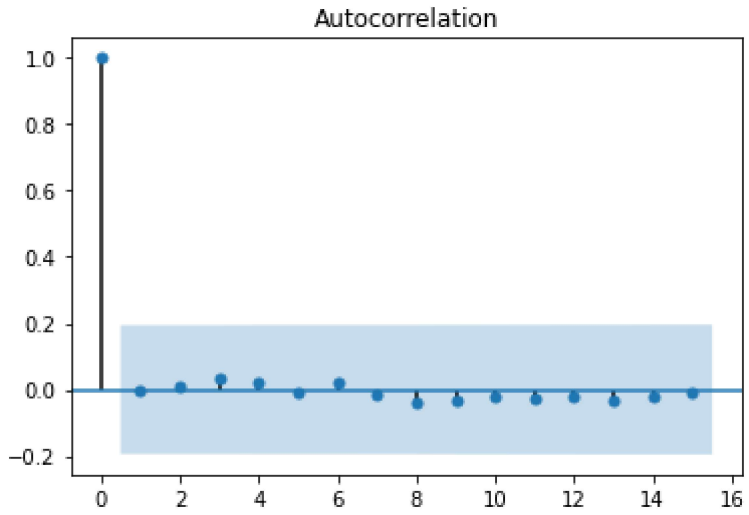
In [192]:

```
# plot residual errors
res1_1 = pd.DataFrame(arma1_1.resid)
plt.plot(res1_1,'b-',label='Residuals')
# Set the labels for axies
plt.xlabel('Date')
plt.ylabel('Residuals')
# Set the legend position
plt.legend(loc='lower right')
plt.show()
```



In [196]:

```
# # Plot the acf of residuals
plot_acf(res1_1, lags=15)
plt.show()
```

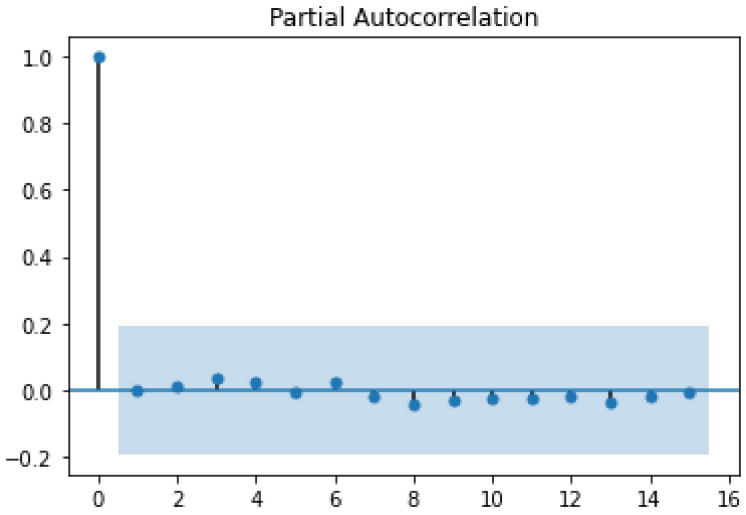


In [164]:

```
#Seems to be no autocorrelation of the residuals
```

In [197]:

```
# # Plot the pacf of residuals
plot_pacf(res1_1, lags=15)
plt.show()
```



In [198]:

```
#Seems to be no autocorrelation of the residuals

#Check non-zero autocorrelation lags with with Ljung-Box test:

# Check possible non-zero lags: i.e. lag 2 in ARMA(1,1)
lag2_name = ['Ljung-Box Test Statistic', 'p-value']
lag2_lb = acorr_ljungbox(res1_1,lags=2)
# Indexing the stats and p-value for the 2th lag
lag2_q = lag2_lb[0][-1]
lag2_p = lag2_lb[1][-1]
# Q-stats
q_stats_lag2 = {'Ljung-Box Test Statistic':lag2_q,'p-value':lag2_p}
q_stats_lag2
```

```
C:\Users\35387\anaconda3\lib\site-packages\statsmodels\stats\diagnostic.py:5
24: FutureWarning: The value returned will change to a single DataFrame afte
r 0.12 is released.  Set return_df to True to use to return a DataFrame now.
Set return_df to False to silence this warning.
  warnings.warn(msg, FutureWarning)
```

Out[198]:

```
{'Ljung-Box Test Statistic': 0.014813203091533458,
 'p-value': 0.9926207597344222}
```

In [176]:

```
#ARMA(2,1)
```

In [199]:

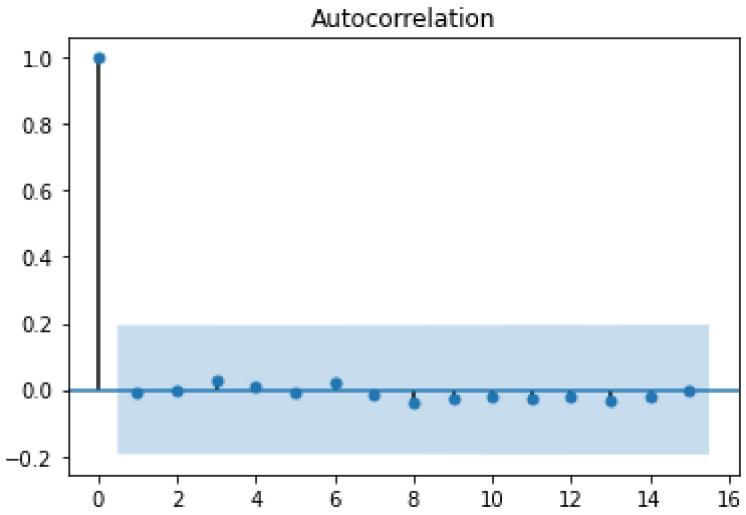
```
# 1. Check model estimates
arma2_1 = tsa.arima_model.ARMA(diff_data,order=(2,1)).fit()
print(arma2_1.summary())
```

```
                              ARMA Model Results
==============================================================================
==
Dep. Variable:                Real GDP   No. Observations:                  1
02
Model:                     ARMA(2, 1)   Log Likelihood                -695.3
49
Method:                       css-mle   S.D. of innovations            220.8
41
Date:                Sun, 08 Nov 2020   AIC                           1400.6
99
Time:                        13:16:53   BIC                           1413.8
24
Sample:                    04-01-1995   HQIC                          1406.0
13
                         - 07-01-2020
==============================================================================
======
                     coef    std err          z      P>|z|      [0.025
0.975]
------------------------------------------------------------------------------
------
const              74.4998     18.880      3.946      0.000      37.496     1
11.504
ar.L1.Real GDP      0.2266      0.674      0.336      0.737      -1.094
1.548
ar.L2.Real GDP      0.2686      0.252      1.066      0.287      -0.226
0.763
ma.L1.Real GDP     -0.5741      0.652     -0.880      0.379      -1.852
0.704
                                    Roots
==============================================================================
=
                  Real          Imaginary           Modulus         Frequenc
y
------------------------------------------------------------------------------
-
AR.1            1.5532           +0.0000j            1.5532            0.000
0
AR.2           -2.3966           +0.0000j            2.3966            0.500
0
MA.1            1.7418           +0.0000j            1.7418            0.000
0
------------------------------------------------------------------------------
-
```

```
C:\Users\35387\anaconda3\lib\site-packages\statsmodels\tsa\base\tsa_model.p
y:159: ValueWarning: No frequency information was provided, so inferred freq
uency QS-OCT will be used.
  warnings.warn('No frequency information was'
```

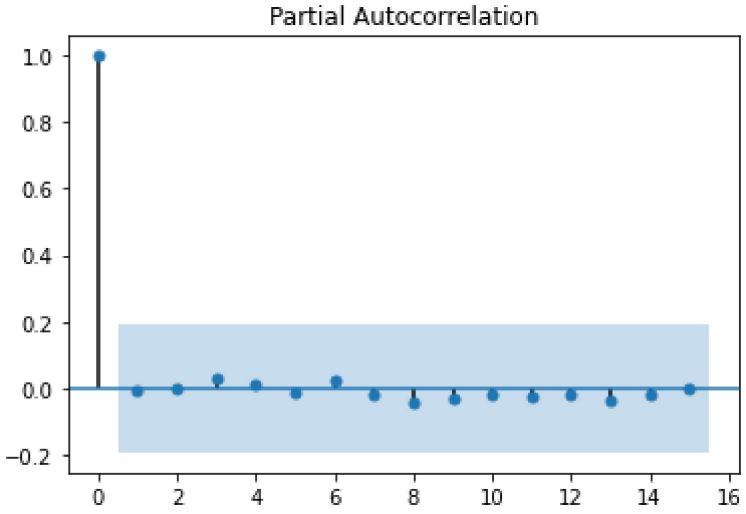
In [200]:

```
# # 2. Check correlogram of residuals: ARMA(2,1)
res2_1 = pd.DataFrame(arma2_1.resid)
plot_acf(res2_1, lags=15)
plt.show()
```



In [201]:

```
plot_pacf(res2_1, lags=15)
plt.show()
```



In [202]:

```
#Seems to be no autocorrelation of the residuals

#Check non-zero autocorrelation lags with with Ljung-Box test:

# Check possible non-zero lags: i.e. lag 8 in ARMA(2,1)
lag8_name = ['Ljung-Box Test Statistic', 'p-value']
lag8_lb = acorr_ljungbox(res2_1,lags=8)
# Indexing the stats and p-value for the 8th lag
lag8_q = lag8_lb[0][-1]
lag8_p = lag8_lb[1][-1]
# Q-stats
q_stats_lag8 = {'Ljung-Box Test Statistic':lag8_q,'p-value':lag8_p}
q_stats_lag8
```

```
C:\Users\35387\anaconda3\lib\site-packages\statsmodels\stats\diagnostic.py:5
24: FutureWarning: The value returned will change to a single DataFrame afte
r 0.12 is released.  Set return_df to True to use to return a DataFrame now.
Set return_df to False to silence this warning.
  warnings.warn(msg, FutureWarning)
```

Out[202]:

```
{'Ljung-Box Test Statistic': 0.3432142272253769, 'p-value': 0.99996848772963
24}
```

In [203]:

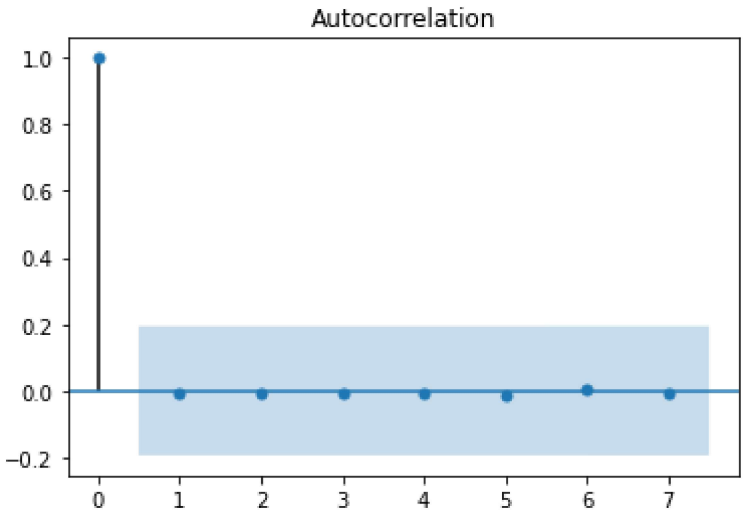
```
# 1. Check model estimates
arma7_1 = tsa.arima_model.ARMA(diff_data,order=(7,1)).fit()
print(arma7_1.summary())
```

```
C:\Users\35387\anaconda3\lib\site-packages\statsmodels\tsa\base\tsa_model.p
y:159: ValueWarning: No frequency information was provided, so inferred freq
uency QS-OCT will be used.
  warnings.warn('No frequency information was'

                              ARMA Model Results
==============================================================================
==
Dep. Variable:                Real GDP   No. Observations:                   1
02
Model:                     ARMA(7, 1)   Log Likelihood                -693.5
21
Method:                       css-mle   S.D. of innovations            214.1
04
Date:                Sun, 08 Nov 2020   AIC                           1407.0
43
Time:                        13:26:26   BIC                           1433.2
93
Sample:                    04-01-1995   HQIC                          1417.6
72
                         - 07-01-2020
==============================================================================
======
                     coef    std err          z      P>|z|      [0.025
0.975]
------------------------------------------------------------------------------
------
const              75.2593      5.469     13.762      0.000      64.541
85.978
ar.L1.Real GDP      0.6320      0.120      5.284      0.000       0.398
0.866
ar.L2.Real GDP      0.2625      0.312      0.842      0.400      -0.349
0.874
ar.L3.Real GDP      0.1951      0.324      0.603      0.547      -0.439
0.829
ar.L4.Real GDP     -0.0209      0.336     -0.062      0.950      -0.680
0.638
ar.L5.Real GDP     -0.2250      0.331     -0.679      0.497      -0.874
0.424
ar.L6.Real GDP      0.2165      0.305      0.709      0.478      -0.382
0.815
ar.L7.Real GDP     -0.1816      0.220     -0.827      0.409      -0.612
0.249
ma.L1.Real GDP     -1.0000      0.036    -27.736      0.000      -1.071
-0.929
                                    Roots
==============================================================================
=
                  Real          Imaginary           Modulus         Frequenc
y
------------------------------------------------------------------------------
-
AR.1           -1.1882           -0.0000j            1.1882           -0.500
0
AR.2           -0.5734           -1.0578j            1.2032           -0.329
```

```
1
AR.3           -0.5734           +1.0578j            1.2032            0.329
1
AR.4            1.1204           -0.1544j            1.1310           -0.021
8
AR.5            1.1204           +0.1544j            1.1310            0.021
8
AR.6            0.6431           -1.4454j            1.5820           -0.183
4
AR.7            0.6431           +1.4454j            1.5820            0.183
4
MA.1            1.0000           +0.0000j            1.0000            0.000
0
-----------------------------------------------------------------------------
-
```

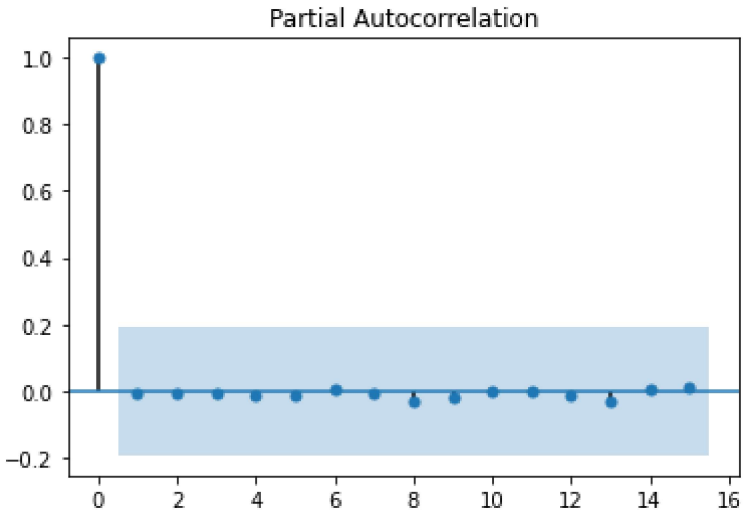
In [204]:

```
# 2. Check correlogram of residuals: ARMA(7,1)
res7_1 = pd.DataFrame(arma7_1.resid)
plot_acf(res7_1, lags=7)
plt.show()
```



In [205]:

```
plot_pacf(res7_1, lags=15)
plt.show()
```



In [206]:

```
# Summarizing all IC
IC = {'AIC':[arma1_1.aic,arma2_1.aic,arma7_1.aic],'BIC':[arma1_1.bic,arma2_1.bic,arma7_1.bi
      'HQIC':[arma1_1.hqic,arma2_1.hqic,arma7_1.hqic]}
IC_df = pd.DataFrame(IC,index=['ARMA(1,1)','ARMA(2,1)','ARMA(7,1)'])
print(IC_df)
```

```
                AIC          BIC         HQIC
ARMA(1,1)  1398.867629  1409.367520  1403.119393
ARMA(2,1)  1400.698783  1413.823647  1406.013488
ARMA(7,1)  1407.042999  1433.292727  1417.672409
```