In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import statsmodels.api as sm
import seaborn as sns
from statsmodels.graphics.tsaplots import plot_acf,plot_pacf
from statsmodels.tsa.arima.model import ARIMA
from statsmodels.tsa.stattools import adfuller
from sklearn import metrics
```

In [2]:

```
from ucimlrepo import fetch_ucirepo

# fetch dataset
air_quality = fetch_ucirepo(id=360)

# data (as pandas dataframes)
df = air_quality.data.features
```

In [3]:

```
# Convert 'Date' and 'Time' columns to datetime format
df['DateTime'] = pd.to_datetime(df['Date']+ ' ' + df['Time'])
df['Date'] = pd.to_datetime(df['Date'])
df.index = df['Date']
df['Year'] = df['Date'].dt.year
df['Month'] = df['Date'].dt.month
df['Day'] = df['Date'].dt.day

# Drop Date and Time columns
df.drop(columns=['Time','Date'], inplace=True)
```

In [4]:

```
df.head()
```

Out[4]:

	CO(GT)	PT08.S1(CO)	NMHC(GT)	C6H6(GT)	PT08.S2(NMHC)	NOx(GT)	PT08.S3(NOx)	I
Date								
2004-03-10	2.6	1360	150	11.9	1046	166	1056	
2004-03-10	2.0	1292	112	9.4	955	103	1174	
2004-03-10	2.2	1402	88	9.0	939	131	1140	
2004-03-10	2.2	1376	80	9.2	948	172	1092	
2004-03-10	1.6	1272	51	6.5	836	131	1205	

In [5]:

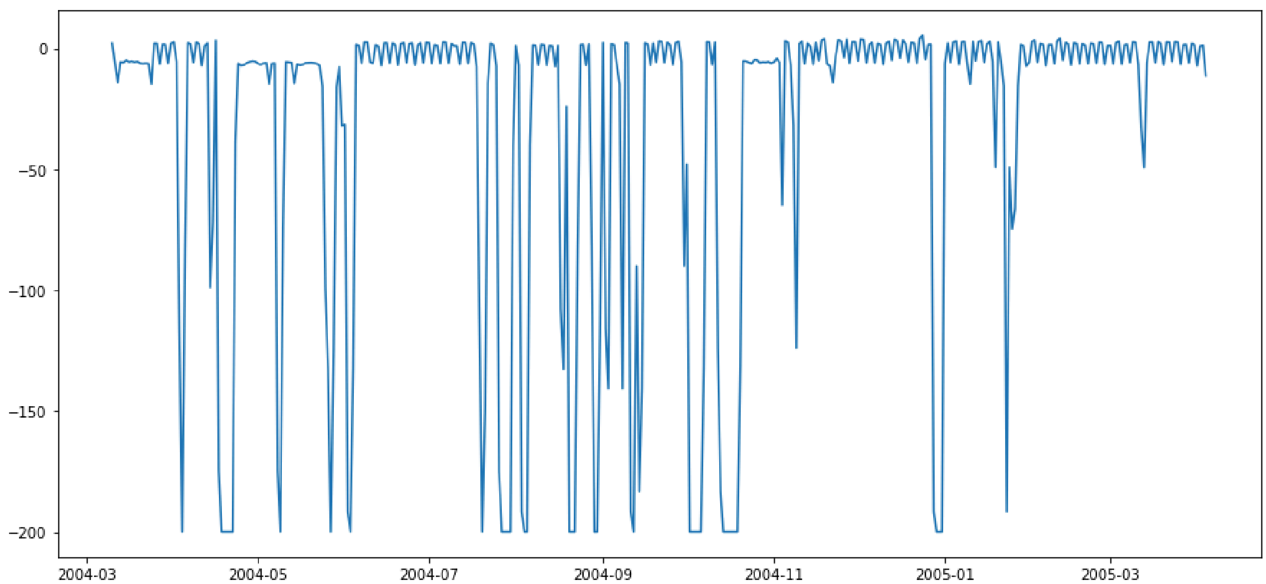
```
year = pd.DataFrame(df.groupby(['Date'])['CO(GT)'].mean())
year.index = pd.DatetimeIndex(year.index, freq=year.index.inferred_freq)
year.head()
```

Out[5]:

Date	CO(GT)
2004-03-10	1.966667
2004-03-11	-6.187500
2004-03-12	-14.095833
2004-03-13	-5.750000
2004-03-14	-5.966667

In [6]:

```
plt.figure(figsize = (15,7))
plt.plot(year, label = 'Average CO2 by Day')
plt.show()
```



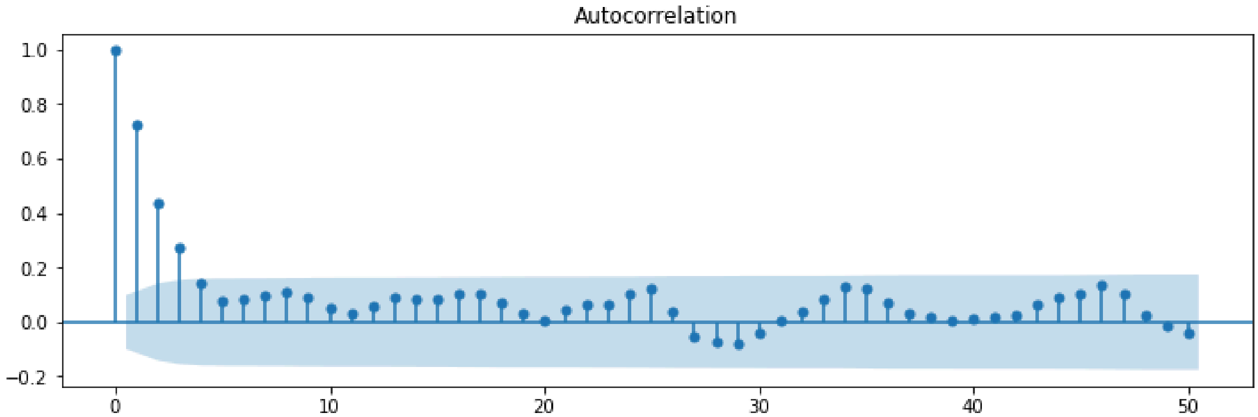
In [7]:

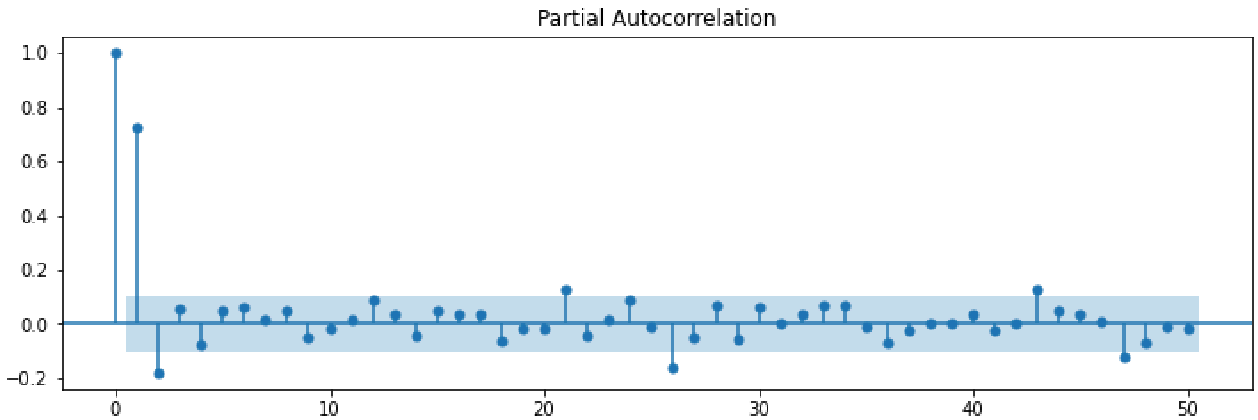
```
adf_test = adfuller(year['CO(GT)'])
# Output the results
print('ADF Statistic: %f' % adf_test[0])
print('p-value: %f' % adf_test[1])
```

```
ADF Statistic: -8.748258
p-value: 0.000000
```

In [8]:

```
fig = plt.figure(figsize=(12,8))
ax1 = fig.add_subplot(211)
sm.graphics.tsa.plot_acf(year['CO(GT)'], lags = 50, ax=ax1)

ax2 = fig.add_subplot(212)
fig = sm.graphics.tsa.plot_pacf(year['CO(GT)'],lags=50,ax=ax2)
```





In [9]:

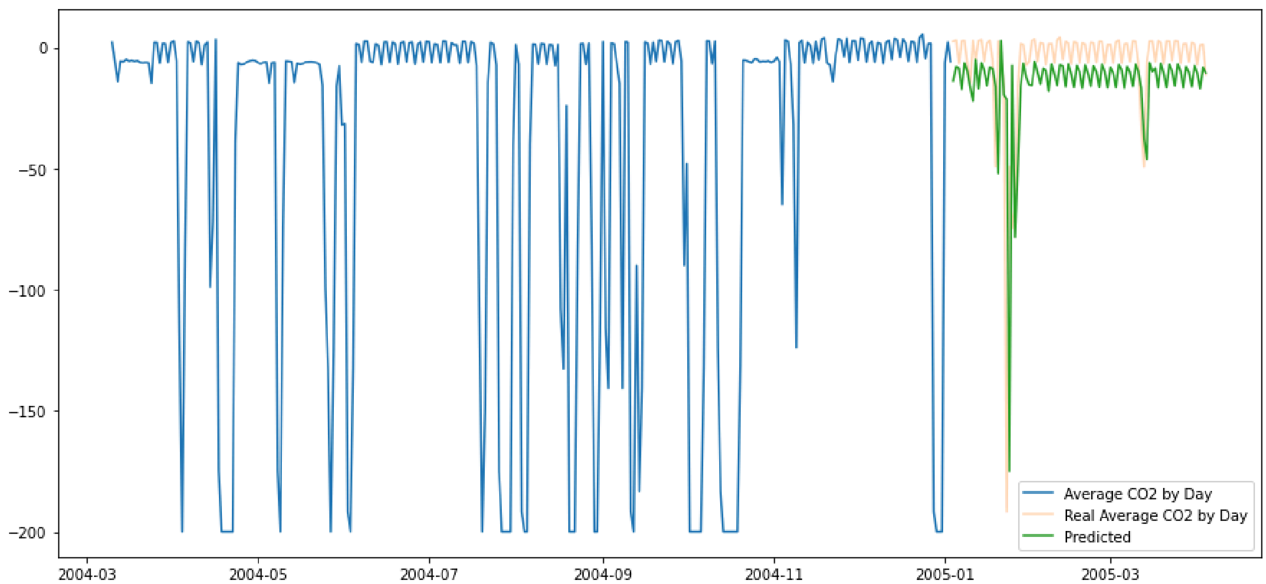
```
mod = ARIMA(year['CO(GT)'], order = (2,0,2))
fit = mod.fit()
print(fit.summary())
```

```
                               SARIMAX Results                                
==============================================================================
Dep. Variable:                 CO(GT)   No. Observations:                  391
Model:                 ARIMA(2, 0, 2)   Log Likelihood               -2042.572
Date:                Wed, 03 Jul 2024   AIC                           4097.144
Time:                        20:06:23   BIC                           4120.957
Sample:                    03-10-2004   HQIC                          4106.583
                         - 04-04-2005                                         
Covariance Type:                  opg                                         
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const        -34.0107     14.572     -2.334      0.020     -62.571      -5.450
ar.L1         -0.3085      0.548     -0.563      0.573      -1.382       0.765
ar.L2          0.5399      0.305      1.768      0.077      -0.058       1.138
ma.L1          1.1786      0.551      2.138      0.033       0.098       2.259
ma.L2          0.2423      0.186      1.303      0.193      -0.122       0.607
sigma2      2014.1686    150.742     13.362      0.000    1718.720    2309.618
================================================================================
===
Ljung-Box (L1) (Q):                   0.00   Jarque-Bera (JB):                58
0.66
Prob(Q):                              0.95   Prob(JB):                          
0.00
Heteroskedasticity (H):               0.68   Skew:                             -
1.45
Prob(H) (two-sided):                  0.03   Kurtosis:                          
8.22
```

```
================================================================================
===

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

In [10]:

```python
year['forecast']=fit.predict(start=300)

plt.figure(figsize = (15,7))
plt.plot(year.iloc[:300]['CO(GT)'], label = 'Average CO2 by Day')
plt.plot(year.iloc[300:]['CO(GT)'], label = 'Real Average CO2 by Day', alpha = .
plt.plot(year['forecast'], label = 'Predicted')
plt.legend()
plt.show()
print('MAE:',metrics.mean_absolute_error(year.iloc[300:]['CO(GT)'],year.iloc[300
```

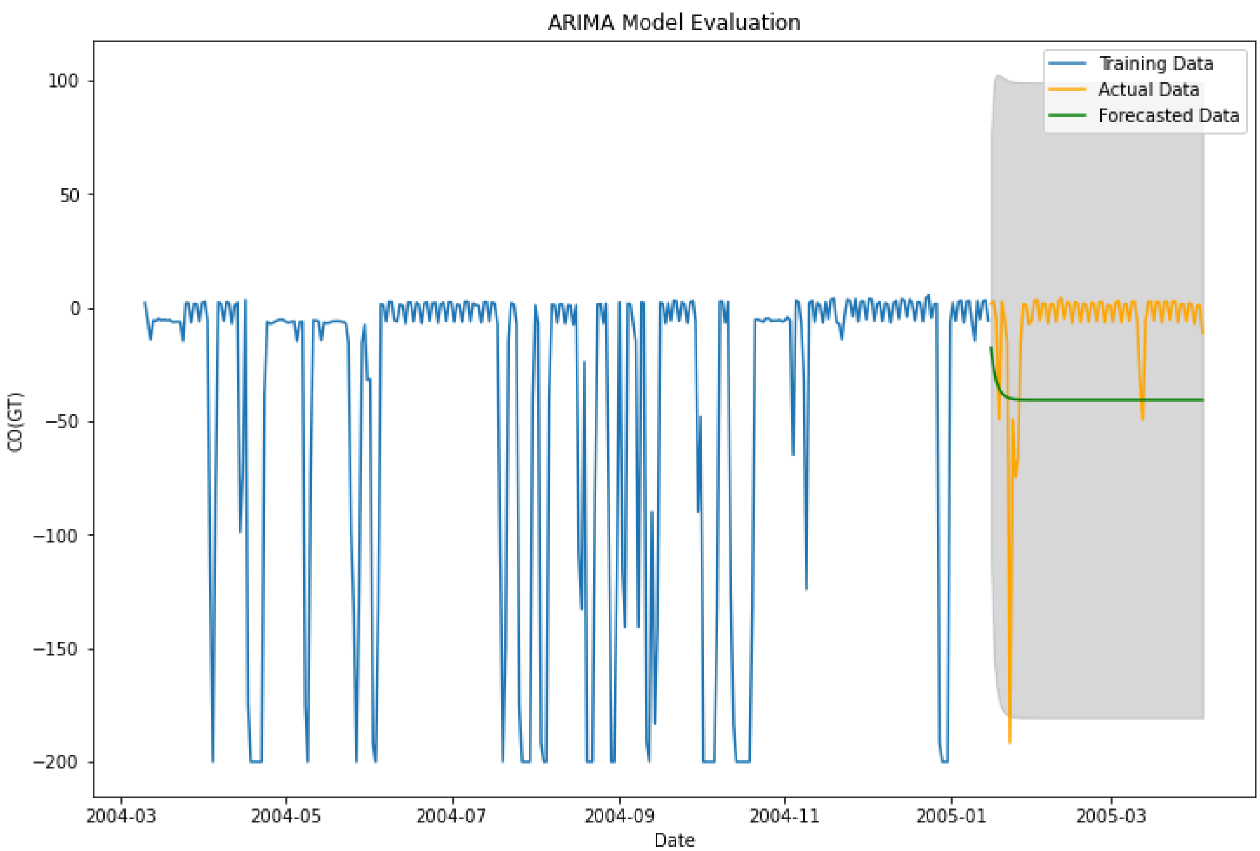



```
MAE: 15.271274917815404
```

In [11]:

```python
#https://medium.com/datainc/time-series-analysis-and-forecasting-with-arima-in-p

# Split the data into train and test
train_size = int(len(year) * 0.8)
train, test = year[0:train_size], year[train_size:len(year)]

# Fit the ARIMA model on the training dataset
model_train = ARIMA(train['CO(GT)'], order=(2, 0, 2))
model_train_fit = model_train.fit()

# Forecast on the test dataset
test_forecast = model_train_fit.get_forecast(steps=len(test))
test_forecast_series = pd.Series(test_forecast.predicted_mean, index=test.index)

# Calculate the mean squared error
mse = metrics.mean_squared_error(test['CO(GT)'], test_forecast_series)
rmse = mse**0.5

# Create a plot to compare the forecast with the actual test data
plt.figure(figsize=(12,8))
plt.plot(train['CO(GT)'], label='Training Data')
```

```python
plt.plot(test['CO(GT)'], label='Actual Data', color='orange')
plt.plot(test_forecast_series, label='Forecasted Data', color='green')
plt.fill_between(test.index,
                 test_forecast.conf_int().iloc[:, 0],
                 test_forecast.conf_int().iloc[:, 1],
                 color='k', alpha=.15)
plt.title('ARIMA Model Evaluation')
plt.xlabel('Date')
plt.ylabel('CO(GT)')
plt.legend()
plt.show()

print('RMSE:', rmse)
```




```
RMSE: 41.310347660980455
```

In [12]:

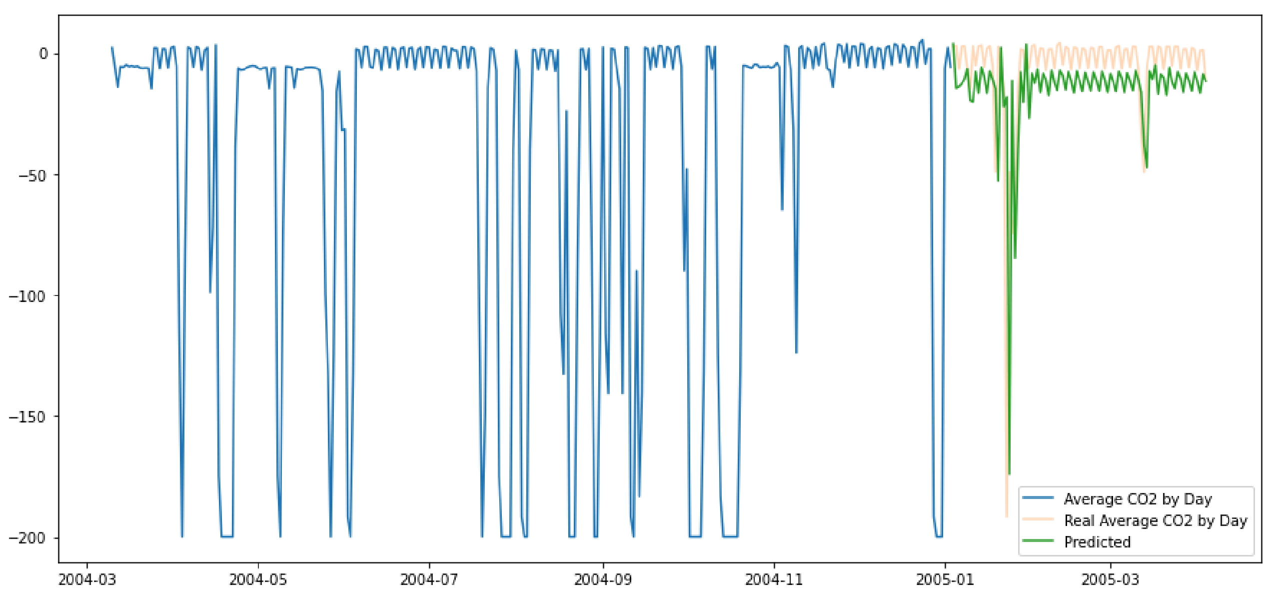
```python
mod = ARIMA(year['CO(GT)'], order = (2,0,4))
fit = mod.fit()
print(fit.summary())
```

```
                               SARIMAX Results
==============================================================================
Dep. Variable:                 CO(GT)   No. Observations:                  391
Model:                 ARIMA(2, 0, 4)   Log Likelihood               -2040.848
Date:                Wed, 03 Jul 2024   AIC                           4097.695
Time:                        20:06:24   BIC                           4129.445
Sample:                    03-10-2004   HQIC                          4110.280
                         - 04-04-2005
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const        -34.0910     13.885     -2.455      0.014     -61.305      -6.877
```

```
ar.L1         -0.1952      0.305     -0.640      0.522     -0.793      0.403
ar.L2          0.2474      0.253      0.976      0.329     -0.249      0.744
ma.L1          1.0642      0.305      3.490      0.000      0.466      1.662
ma.L2          0.4311      0.203      2.120      0.034      0.033      0.830
ma.L3          0.2445      0.150      1.635      0.102     -0.049      0.538
ma.L4          0.1480      0.068      2.170      0.030      0.014      0.282
sigma2      1996.2661    144.407     13.824      0.000   1713.234   2279.298
================================================================================
===
Ljung-Box (L1) (Q):                   0.00   Jarque-Bera (JB):              53
4.87
Prob(Q):                              0.97   Prob(JB):
0.00
Heteroskedasticity (H):               0.68   Skew:                          -
1.37
Prob(H) (two-sided):                  0.03   Kurtosis:
8.04
================================================================================
===

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

In [13]:

```python
year['forecast']=fit.predict(start=300)

plt.figure(figsize = (15,7))
plt.plot(year.iloc[:300]['CO(GT)'], label = 'Average CO2 by Day')
plt.plot(year.iloc[300:]['CO(GT)'], label = 'Real Average CO2 by Day', alpha = .
plt.plot(year['forecast'], label = 'Predicted')
plt.legend()
plt.show()
print('MAE:',metrics.mean_absolute_error(year.iloc[300:]['CO(GT)'],year.iloc[300
```

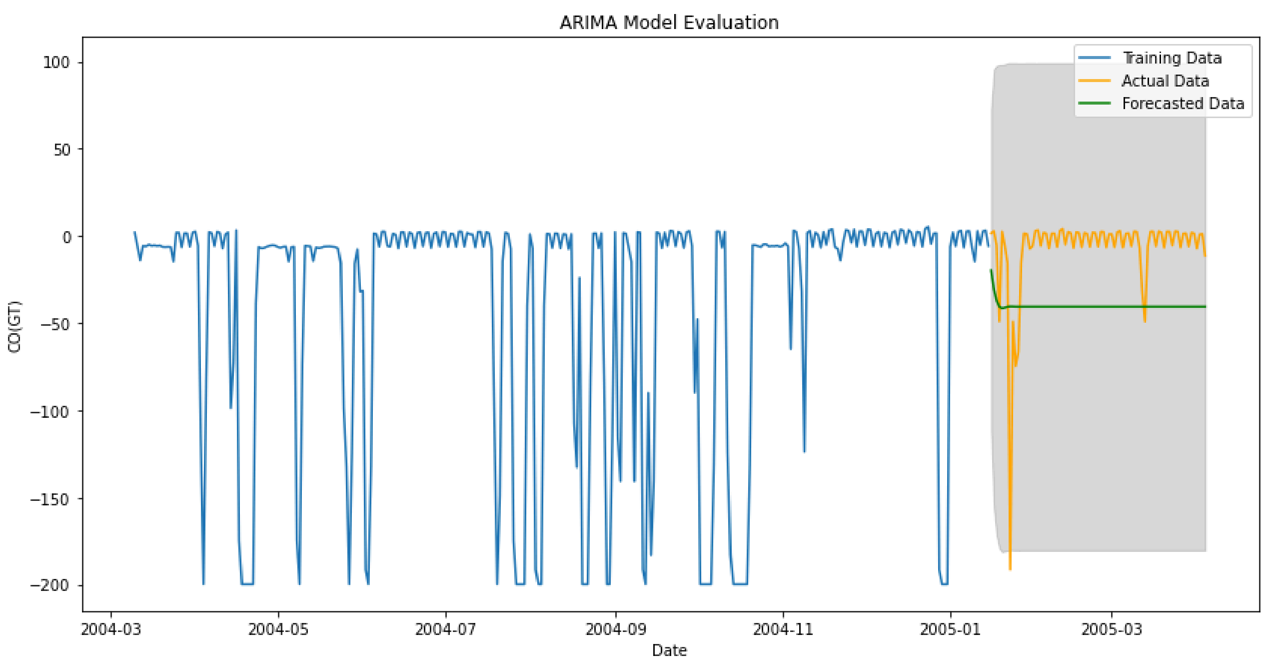



```
MAE: 15.750089920820542
```

In [14]:

```python
#https://medium.com/datainc/time-series-analysis-and-forecasting-with-arima-in-p

# Split the data into train and test
train_size = int(len(year) * 0.8)
```

```python
train, test = year[0:train_size], year[train_size:len(year)]

# Fit the ARIMA model on the training dataset
model_train = ARIMA(train['CO(GT)'], order=(2, 0, 4))
model_train_fit = model_train.fit()

# Forecast on the test dataset
test_forecast = model_train_fit.get_forecast(steps=len(test))
test_forecast_series = pd.Series(test_forecast.predicted_mean, index=test.index)

# Calculate the mean squared error
mse = metrics.mean_squared_error(test['CO(GT)'], test_forecast_series)
rmse = mse**0.5

# Create a plot to compare the forecast with the actual test data
plt.figure(figsize=(14,7))
plt.plot(train['CO(GT)'], label='Training Data')
plt.plot(test['CO(GT)'], label='Actual Data', color='orange')
plt.plot(test_forecast_series, label='Forecasted Data', color='green')
plt.fill_between(test.index,
                 test_forecast.conf_int().iloc[:, 0],
                 test_forecast.conf_int().iloc[:, 1],
                 color='k', alpha=.15)
plt.title('ARIMA Model Evaluation')
plt.xlabel('Date')
plt.ylabel('CO(GT)')
plt.legend()
plt.show()

print('RMSE:', rmse)
```




```
RMSE: 41.4507992420051
```

In [15]:

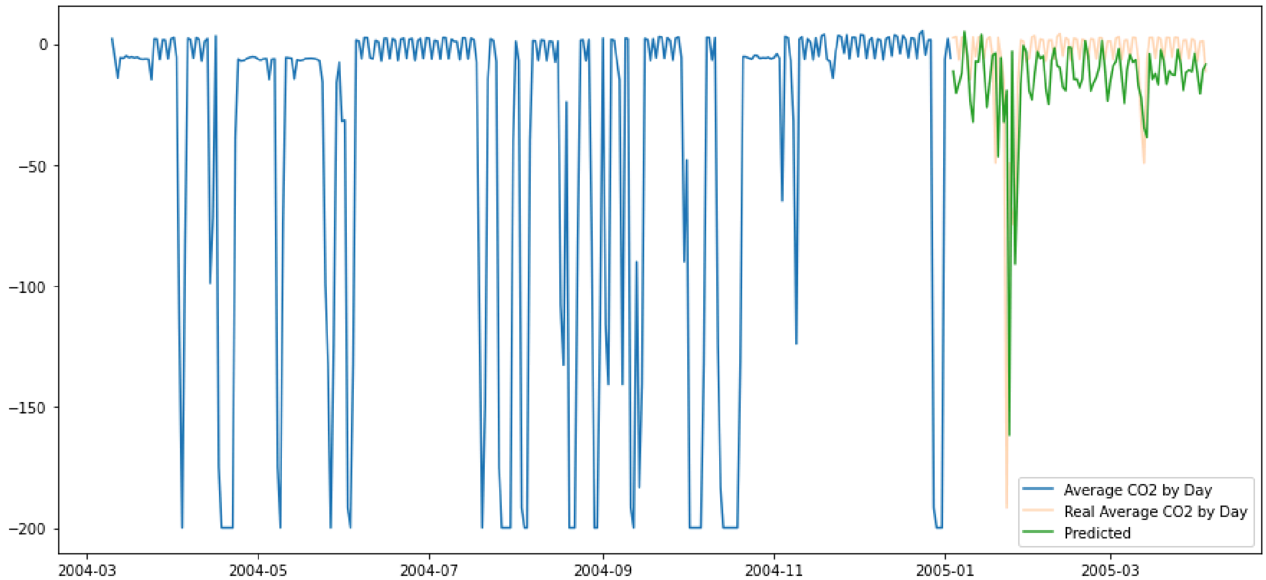
```python
mod = ARIMA(year['CO(GT)'], order = (3,0,4))
fit = mod.fit()
print(fit.summary())
```

```
                               SARIMAX Results
==============================================================================
```

```
Dep. Variable:                 CO(GT)   No. Observations:                  391
Model:                 ARIMA(3, 0, 4)   Log Likelihood               -2037.768
Date:                Wed, 03 Jul 2024   AIC                           4093.535
Time:                        20:06:25   BIC                           4129.253
Sample:                    03-10-2004   HQIC                          4107.693
                         - 04-04-2005                                         
Covariance Type:                  opg                                         
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const        -34.0837     13.919     -2.449      0.014     -61.364      -6.804
ar.L1          1.4409      0.105     13.754      0.000       1.236       1.646
ar.L2         -1.4472      0.096    -15.121      0.000      -1.635      -1.260
ar.L3          0.5867      0.094      6.259      0.000       0.403       0.770
ma.L1         -0.5846      0.101     -5.765      0.000      -0.783      -0.386
ma.L2          0.7267      0.053     13.780      0.000       0.623       0.830
ma.L3          0.2917      0.055      5.347      0.000       0.185       0.399
ma.L4         -0.0760      0.072     -1.054      0.292      -0.217       0.065
sigma2       1893.4882    132.486     14.292      0.000    1633.820    2153.156
================================================================================
===
Ljung-Box (L1) (Q):                   0.00   Jarque-Bera (JB):                52
7.77
Prob(Q):                              0.99   Prob(JB):                        
0.00
Heteroskedasticity (H):               0.71   Skew:                            -
1.35
Prob(H) (two-sided):                  0.05   Kurtosis:                        
8.01
================================================================================
===

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-s
tep).
/Users/zanderbonnet/opt/anaconda3/lib/python3.9/site-packages/statsmodels/base/m
odel.py:566: ConvergenceWarning: Maximum Likelihood optimization failed to conve
rge. Check mle_retvals
  warnings.warn("Maximum Likelihood optimization failed to "
```

In [16]:

```
year['forecast']=fit.predict(start=300)

plt.figure(figsize = (15,7))
plt.plot(year.iloc[:300]['CO(GT)'], label = 'Average CO2 by Day')
plt.plot(year.iloc[300:]['CO(GT)'], label = 'Real Average CO2 by Day', alpha = .
plt.plot(year['forecast'], label = 'Predicted')
plt.legend()
plt.show()
print('MAE:',metrics.mean_absolute_error(year.iloc[300:]['CO(GT)'],year.iloc[300
```

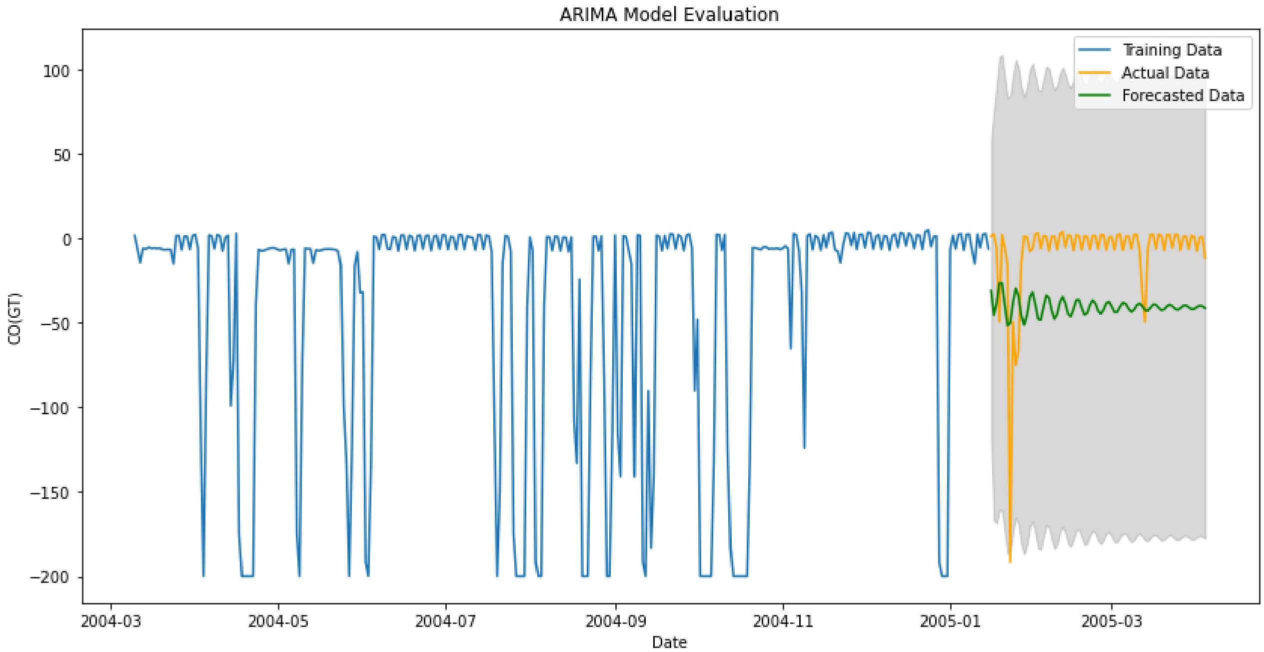



```
MAE: 15.932133664626555
```

In [17]:

```python
#https://medium.com/datainc/time-series-analysis-and-forecasting-with-arima-in-p

# Split the data into train and test
train_size = int(len(year) * 0.8)
train, test = year[0:train_size], year[train_size:len(year)]

# Fit the ARIMA model on the training dataset
model_train = ARIMA(train['CO(GT)'], order=(3, 0, 4))
model_train_fit = model_train.fit()

# Forecast on the test dataset
test_forecast = model_train_fit.get_forecast(steps=len(test))
test_forecast_series = pd.Series(test_forecast.predicted_mean, index=test.index)

# Calculate the mean squared error
mse = metrics.mean_squared_error(test['CO(GT)'], test_forecast_series)
rmse = mse**0.5

# Create a plot to compare the forecast with the actual test data
plt.figure(figsize=(14,7))
plt.plot(train['CO(GT)'], label='Training Data')
plt.plot(test['CO(GT)'], label='Actual Data', color='orange')
plt.plot(test_forecast_series, label='Forecasted Data', color='green')
plt.fill_between(test.index,
                 test_forecast.conf_int().iloc[:, 0],
                 test_forecast.conf_int().iloc[:, 1],
                 color='k', alpha=.15)
plt.title('ARIMA Model Evaluation')
plt.xlabel('Date')
plt.ylabel('CO(GT)')
plt.legend()
plt.show()

print('RMSE:', rmse)
```

```
/Users/zanderbonnet/opt/anaconda3/lib/python3.9/site-packages/statsmodels/base/m
odel.py:566: ConvergenceWarning: Maximum Likelihood optimization failed to conve
rge. Check mle_retvals
  warnings.warn("Maximum Likelihood optimization failed to "
```



RMSE: 41.70866896770278