In [2]:

```
!pip install pmdarima
```

```
Requirement already satisfied: pmdarima in c:\users\kurno\anaconda3\lib\si
te-packages (1.8.5)
Requirement already satisfied: joblib>=0.11 in c:\users\kurno\anaconda3\li
b\site-packages (from pmdarima) (0.16.0)
Requirement already satisfied: urllib3 in c:\users\kurno\anaconda3\lib\sit
e-packages (from pmdarima) (1.25.9)
Requirement already satisfied: scipy>=1.3.2 in c:\users\kurno\anaconda3\li
b\site-packages (from pmdarima) (1.5.0)
Requirement already satisfied: numpy>=1.19.3 in c:\users\kurno\anaconda3\l
ib\site-packages (from pmdarima) (1.23.2)
Requirement already satisfied: setuptools!=50.0.0,>=38.6.0 in c:\users\kur
no\anaconda3\lib\site-packages (from pmdarima) (49.2.0.post20200714)
Requirement already satisfied: pandas>=0.19 in c:\users\kurno\anaconda3\li
b\site-packages (from pmdarima) (1.0.5)
Requirement already satisfied: Cython!=0.29.18,>=0.29 in c:\users\kurno\an
aconda3\lib\site-packages (from pmdarima) (0.29.21)
Requirement already satisfied: scikit-learn>=0.22 in c:\users\kurno\anacon
da3\lib\site-packages (from pmdarima) (0.23.1)
Requirement already satisfied: statsmodels!=0.12.0,>=0.11 in c:\users\kurn
o\anaconda3\lib\site-packages (from pmdarima) (0.11.1)
Requirement already satisfied: pytz>=2017.2 in c:\users\kurno\anaconda3\li
b\site-packages (from pandas>=0.19->pmdarima) (2020.1)
Requirement already satisfied: python-dateutil>=2.6.1 in c:\users\kurno\an
aconda3\lib\site-packages (from pandas>=0.19->pmdarima) (2.8.1)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\kurno\anac
onda3\lib\site-packages (from scikit-learn>=0.22->pmdarima) (2.1.0)
Requirement already satisfied: patsy>=0.5 in c:\users\kurno\anaconda3\lib
\site-packages (from statsmodels!=0.12.0,>=0.11->pmdarima) (0.5.1)
Requirement already satisfied: six>=1.5 in c:\users\kurno\anaconda3\lib\si
te-packages (from python-dateutil>=2.6.1->pandas>=0.19->pmdarima) (1.15.0)
```

In [3]:

```
import os
import warnings
warnings.filterwarnings('ignore')
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
plt.style.use('fivethirtyeight')
from pylab import rcParams
rcParams['figure.figsize'] = 10, 6
from statsmodels.tsa.stattools import adfuller
from statsmodels.tsa.seasonal import seasonal_decompose
from statsmodels.tsa.arima_model import ARIMA
from pmdarima.arima import auto_arima

from sklearn.metrics import mean_squared_error, mean_absolute_error
import math

for dirname, _, filenames in os.walk('https://www.kaggle.com/code/nageshsingh/stock-mar
ket-forecasting-arima/data'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
```

In [5]:

```
dateparse = lambda dates: pd.datetime.strptime(dates, '%Y-%m-%d')
stock_data = pd.read_csv('Downloads/acgl.us.txt',sep=',', index_col='Date', parse_dates
=['Date'], date_parser=dateparse).fillna(0)
stock_data
```

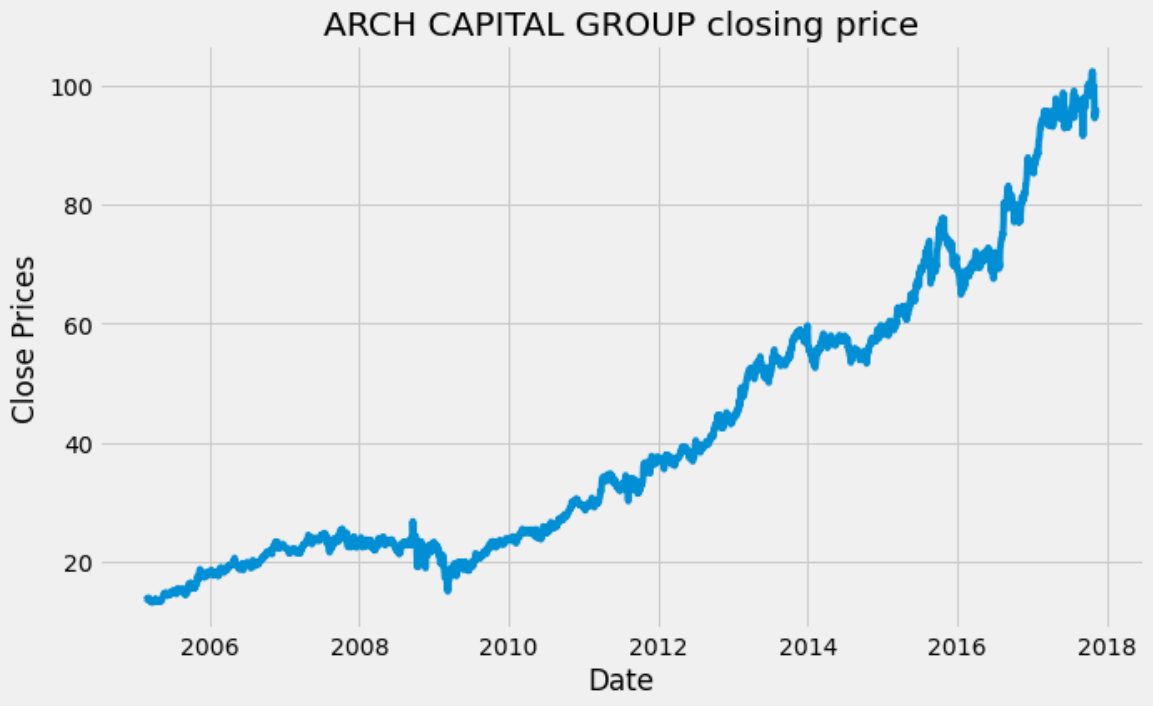
Out[5]:

Date	Open	High	Low	Close	Volume	OpenInt
2005-02-25	13.583	13.693	13.430	13.693	156240	0
2005-02-28	13.697	13.827	13.540	13.827	370509	0
2005-03-01	13.780	13.913	13.720	13.760	224484	0
2005-03-02	13.717	13.823	13.667	13.810	286431	0
2005-03-03	13.783	13.783	13.587	13.630	193824	0
...	...	...	...	...	...	...
2017-11-06	94.490	95.650	94.020	95.550	420192	0
2017-11-07	95.860	95.950	95.200	95.560	464011	0
2017-11-08	95.410	95.900	94.890	95.450	471756	0
2017-11-09	94.930	96.140	94.470	95.910	353498	0
2017-11-10	95.890	95.990	94.390	95.350	452833	0

3201 rows × 6 columns

In [11]:

```
plt.figure(figsize=(10,6))
plt.grid(True)
plt.xlabel('Date')
plt.ylabel('Close Prices')
plt.plot(stock_data['Close'])
plt.title('ARCH CAPITAL GROUP closing price')
plt.show()
```



In [17]:

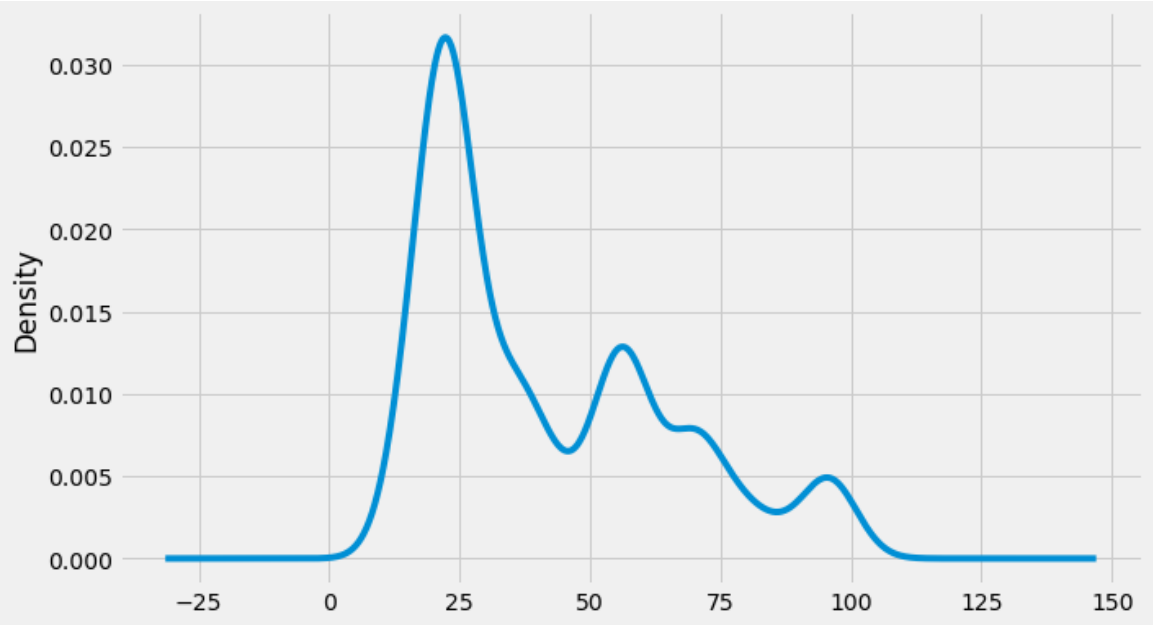
```
df_close=stock_data['Close']
```

In [18]:

```
df_close.plot(kind='kde')
```

Out[18]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1bd775f3c70>
```

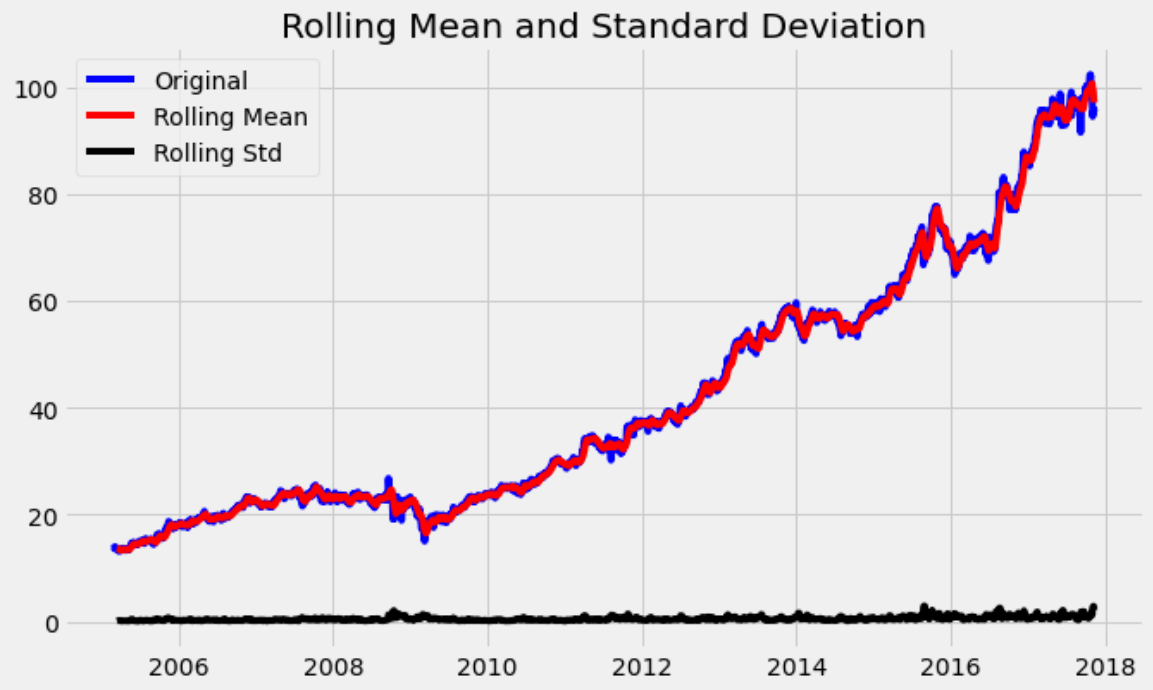


In [19]:

```
def test_stationarity(timeseries):
    #Determing rolling statistics
    rolmean = timeseries.rolling(12).mean()
    rolstd = timeseries.rolling(12).std()
    #Plot rolling statistics:
    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)

test_stationarity(df_close)
```

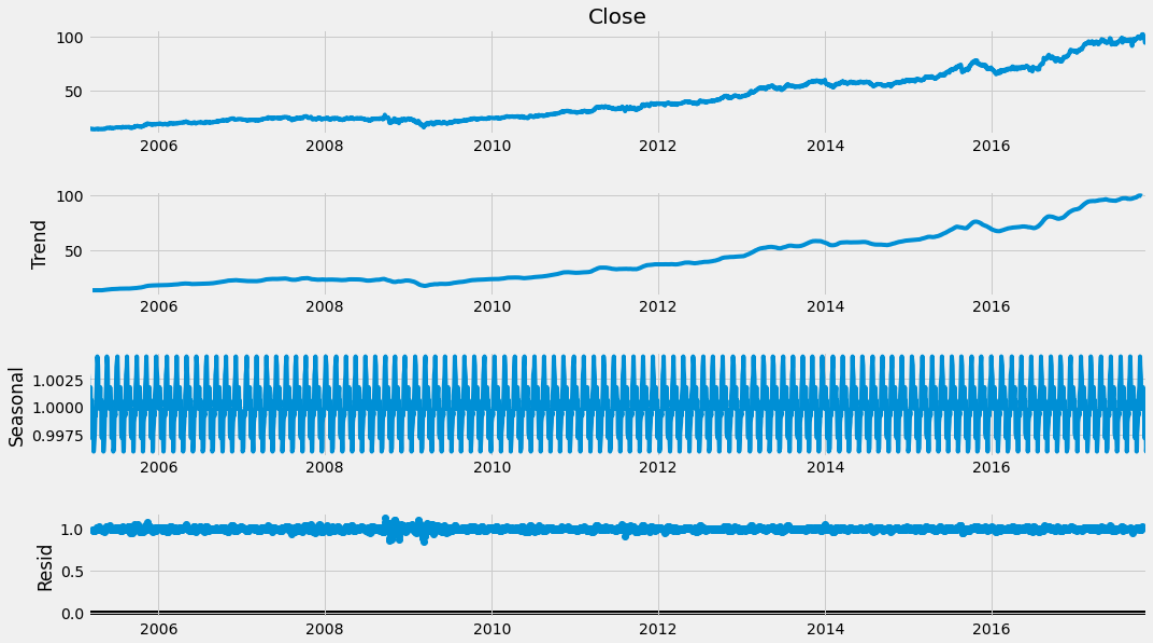



```
Results of dickey fuller test
Test Statistics                  1.374899
p-value                          0.996997
No. of lags used                 5.000000
Number of observations used    3195.000000
critical value (1%)              -3.432398
critical value (5%)              -2.862445
critical value (10%)             -2.567252
dtype: float64
```

In [20]:

```
result = seasonal_decompose(df_close, model='multiplicative', freq = 30)
fig = plt.figure()
fig = result.plot()
fig.set_size_inches(16, 9)
```

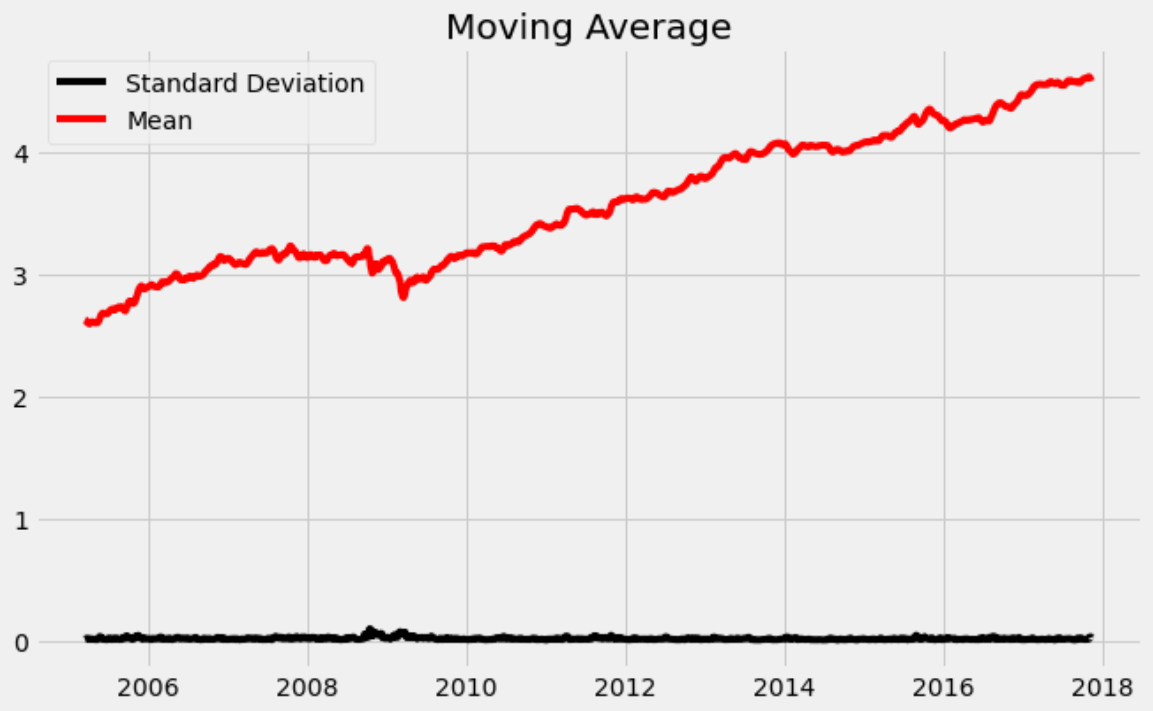
```
<Figure size 720x432 with 0 Axes>
```



In [21]:

```
from pylab import rcParams
rcParams['figure.figsize'] = 10, 6
df_log = np.log(df_close)
moving_avg = df_log.rolling(12).mean()
std_dev = df_log.rolling(12).std()
plt.legend(loc='best')
plt.title('Moving Average')
plt.plot(std_dev, color ="black", label = "Standard Deviation")
plt.plot(moving_avg, color="red", label = "Mean")
plt.legend()
plt.show()
```

```
No handles with labels found to put in legend.
```



In [22]:

```
train_data, test_data = df_log[3:int(len(df_log)*0.9)], df_log[int(len(df_log)*0.9):]
plt.figure(figsize=(10,6))
plt.grid(True)
plt.xlabel('Dates')
plt.ylabel('Closing Prices')
plt.plot(df_log, 'green', label='Train data')
plt.plot(test_data, 'blue', label='Test data')
plt.legend()
```

Out[22]:

```
<matplotlib.legend.Legend at 0x1bd7be0d670>
```

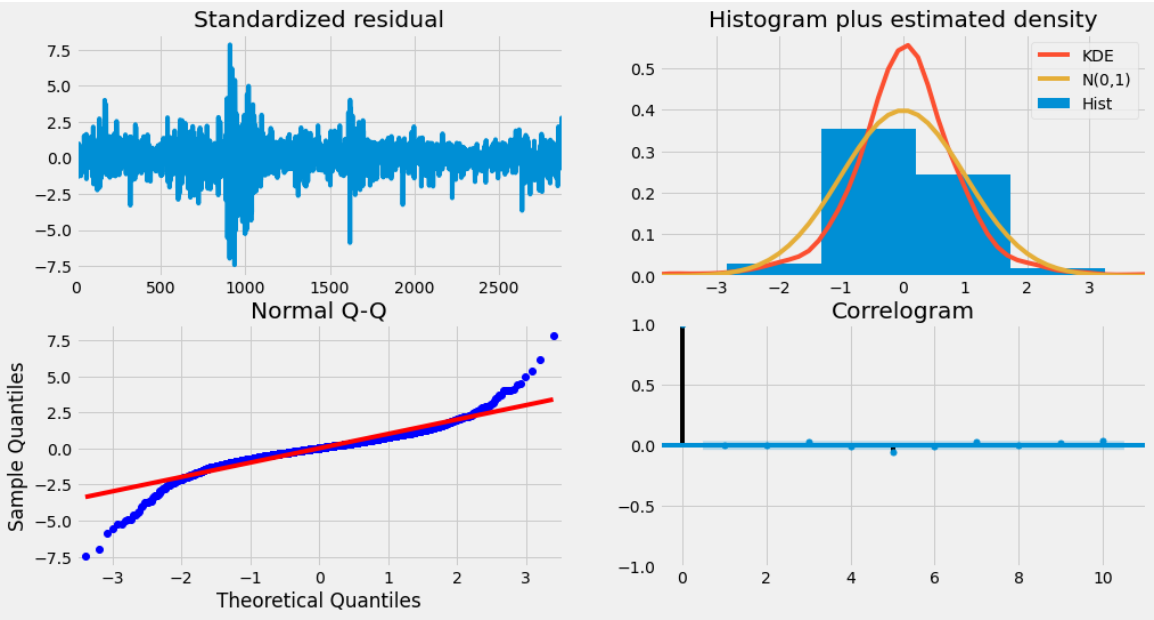
In [23]:

```
model_autoARIMA = auto_arima(train_data, start_p=0, start_q=0,
                      test='adf',       # use adftest to find optimal 'd'
                      max_p=3, max_q=3, # maximum p and q
                      m=1,              # frequency of series
                      d=None,           # let model determine 'd'
                      seasonal=False,   # No Seasonality
                      start_P=0,
                      D=0,
                      trace=True,
                      error_action='ignore',
                      suppress_warnings=True,
                      stepwise=True)
print(model_autoARIMA.summary())
model_autoARIMA.plot_diagnostics(figsize=(15,8))
plt.show()
```

```
Performing stepwise search to minimize aic
 ARIMA(0,1,0)(0,0,0)[0] intercept   : AIC=-16491.508, Time=0.77 sec
 ARIMA(1,1,0)(0,0,0)[0] intercept   : AIC=-16525.993, Time=0.46 sec
 ARIMA(0,1,1)(0,0,0)[0] intercept   : AIC=-16527.964, Time=0.70 sec
 ARIMA(0,1,0)(0,0,0)[0]             : AIC=-16488.323, Time=0.18 sec
 ARIMA(1,1,1)(0,0,0)[0] intercept   : AIC=-16527.157, Time=1.74 sec
 ARIMA(0,1,2)(0,0,0)[0] intercept   : AIC=-16527.120, Time=1.75 sec
 ARIMA(1,1,2)(0,0,0)[0] intercept   : AIC=-16528.152, Time=3.70 sec
 ARIMA(2,1,2)(0,0,0)[0] intercept   : AIC=inf, Time=4.12 sec
 ARIMA(1,1,3)(0,0,0)[0] intercept   : AIC=-16526.183, Time=4.94 sec
 ARIMA(0,1,3)(0,0,0)[0] intercept   : AIC=-16524.974, Time=2.05 sec
 ARIMA(2,1,1)(0,0,0)[0] intercept   : AIC=-16525.435, Time=1.41 sec
 ARIMA(2,1,3)(0,0,0)[0] intercept   : AIC=-16516.417, Time=0.88 sec
 ARIMA(1,1,2)(0,0,0)[0]             : AIC=-16527.597, Time=0.95 sec

Best model:  ARIMA(1,1,2)(0,0,0)[0] intercept
Total fit time: 23.685 seconds
                               SARIMAX Results
==============================================================================
====
Dep. Variable:                      y   No. Observations:
2877
Model:               SARIMAX(1, 1, 2)   Log Likelihood                826
9.076
Date:                Thu, 18 Aug 2022   AIC                          -1652
8.152
Time:                        23:10:51   BIC                          -1649
8.331
Sample:                             0   HQIC                         -1651
7.402
                                - 2877
Covariance Type:                  opg
==============================================================================
====
                 coef    std err          z      P>|z|      [0.025      0.
975]
------------------------------------------------------------------------------
----
intercept    2.27e-05   6.69e-06      3.393      0.001    9.59e-06    3.58
e-05
ar.L1          0.9538      0.009    104.122      0.000       0.936
0.972
ma.L1         -1.0708      0.015    -73.543      0.000      -1.099      -
1.042
ma.L2          0.0877      0.012      7.501      0.000       0.065
0.111
sigma2         0.0002   2.32e-06     80.812      0.000       0.000
0.000
==============================================================================
=========
Ljung-Box (Q):                      121.69   Jarque-Bera (JB):
7206.78
Prob(Q):                              0.00   Prob(JB):
0.00
Heteroskedasticity (H):               0.30   Skew:
-0.39
Prob(H) (two-sided):                  0.00   Kurtosis:
10.72
==============================================================================
=========
```

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



In [24]:

```
model = ARIMA(train_data, order=(1,1,2))
fitted = model.fit(disp=-1)
print(fitted.summary())
```

```
C:\Users\kurno\anaconda3\lib\site-packages\statsmodels\tsa\base\tsa_model.
py:216: ValueWarning: A date index has been provided, but it has no associ
ated frequency information and so will be ignored when e.g. forecasting.
  warnings.warn('A date index has been provided, but it has no'
C:\Users\kurno\anaconda3\lib\site-packages\statsmodels\tsa\base\tsa_model.
py:216: ValueWarning: A date index has been provided, but it has no associ
ated frequency information and so will be ignored when e.g. forecasting.
  warnings.warn('A date index has been provided, but it has no'
```

ARIMA Model Results

Dep. Variable:	D.Close	No. Observations:	2876
Model:	ARIMA(1, 1, 2)	Log Likelihood	8274.158
Method:	css-mle	S.D. of innovations	0.014
Date:	Thu, 18 Aug 2022	AIC	-16538.316
Time:	23:11:53	BIC	-16508.496
Sample:	1	HQIC	-16527.567

	coef	std err	z	P>|z|	[0.025	0.975]
const	0.0006	0.000	3.935	0.000	0.000	0.001
ar.L1.D.Close	0.9145	0.040	22.740	0.000	0.836	0.993
ma.L1.D.Close	-1.0351	0.045	-23.127	0.000	-1.123	-0.947
ma.L2.D.Close	0.0848	0.022	3.820	0.000	0.041	0.128

Roots

	Real	Imaginary	Modulus	Frequency
AR.1	1.0934	+0.0000j	1.0934	0.0000
MA.1	1.0578	+0.0000j	1.0578	0.0000
MA.2	11.1424	+0.0000j	11.1424	0.0000

In [28]:

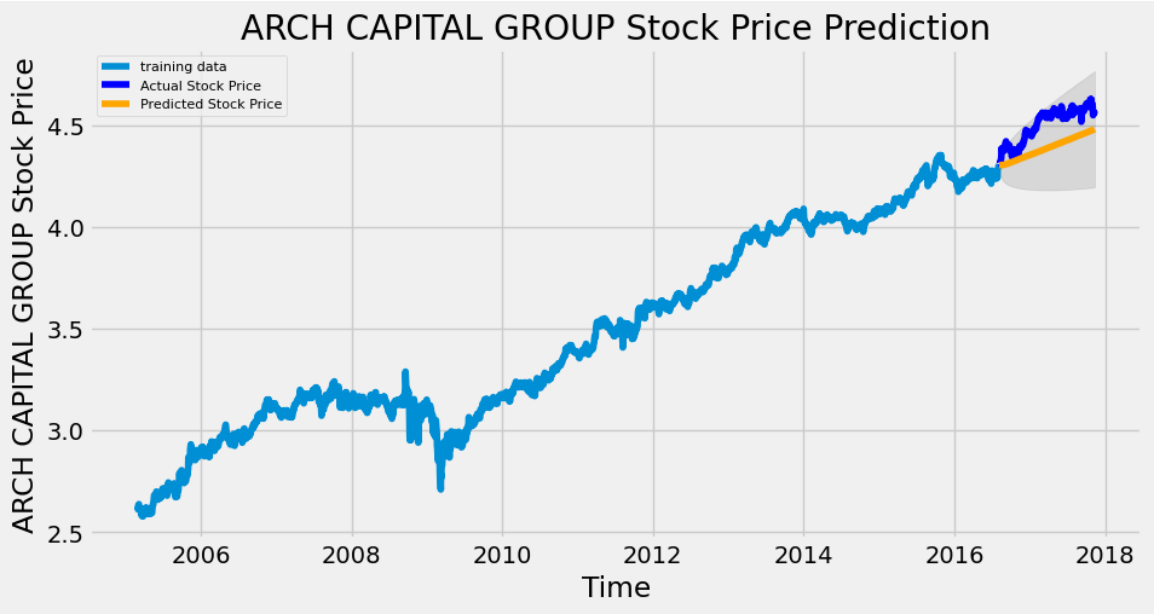
```
#Forecasting the test data
```

In [25]:

```
fc, se, conf = fitted.forecast(321, alpha=0.05)
```

In [26]:

```
fc_series = pd.Series(fc, index=test_data.index)
lower_series = pd.Series(conf[:, 0], index=test_data.index)
upper_series = pd.Series(conf[:, 1], index=test_data.index)
# Plot
plt.figure(figsize=(10,5), dpi=100)
plt.plot(train_data, label='training data')
plt.plot(test_data, color = 'blue', label='Actual Stock Price')
plt.plot(fc_series, color = 'orange',label='Predicted Stock Price')
plt.fill_between(lower_series.index, lower_series, upper_series,
                 color='k', alpha=.10)
plt.title('ARCH CAPITAL GROUP Stock Price Prediction')
plt.xlabel('Time')
plt.ylabel('ARCH CAPITAL GROUP Stock Price')
plt.legend(loc='upper left', fontsize=8)
plt.show()
```



In [27]:

```
mse = mean_squared_error(test_data, fc)
print('MSE: '+str(mse))
mae = mean_absolute_error(test_data, fc)
print('MAE: '+str(mae))
rmse = math.sqrt(mean_squared_error(test_data, fc))
print('RMSE: '+str(rmse))
mape = np.mean(np.abs(fc - test_data)/np.abs(test_data))
print('MAPE: '+str(mape))
```

```
MSE: 0.015076603462322346
MAE: 0.11500989703236117
RMSE: 0.12278682120782484
MAPE: 0.02539744321032314
```

In [29]:

```
#Around 2.5% MAPE implies the model is about 97.5% accurate in predicting the next 15 o
bservations.
```

In [ ]:

```

```