```
#Time Series Prediction. https://drive.google.com/file/d/1JJB4SO5XCq_B3TVbYe57c4RDQ6-xV074


import pandas as pd
import numpy as np
import statsmodels.api as sm
import matplotlib.pyplot as plt


from pandas import read_csv
from statsmodels.tsa.stattools import adfuller
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

import statsmodels.api as sm
from numpy.random import normal, seed
from scipy.stats import norm
from statsmodels.tsa.arima_model import ARMA
from statsmodels.tsa.stattools import adfuller
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
from statsmodels.tsa.arima_process import ArmaProcess
from statsmodels.tsa.arima_model import ARIMA
import math
from sklearn.metrics import mean_squared_error

from plotly import tools

from plotly.offline import init_notebook_mode, iplot
init_notebook_mode(connected=True)
import plotly.graph_objs as go
import plotly.figure_factory as ff



from pydrive.auth import GoogleAuth
from pydrive.drive import GoogleDrive
from google.colab import auth
from oauth2client.client import GoogleCredentials

auth.authenticate_user()
gauth = GoogleAuth()
gauth.credentials = GoogleCredentials.get_application_default()
drive = GoogleDrive(gauth)
```

```
WARNING:root:pydrive is deprecated and no longer maintained. We recommend that you migrate your projects to pydrive2, the
```

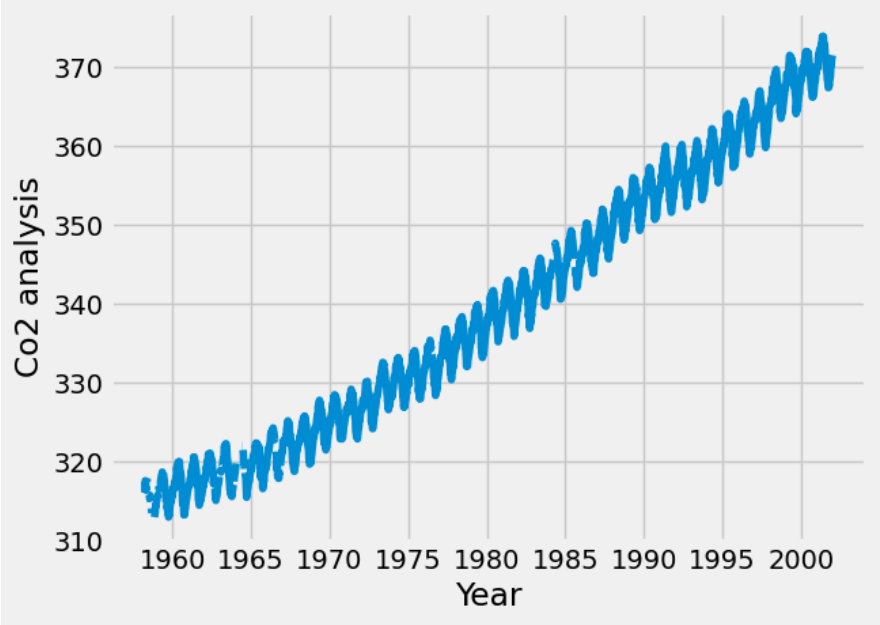
```
Lingyi_Co2_df = pd.read_csv('/content/co2.csv',parse_dates = ['date'],index_col = 'date')
Lingyi_Co2_df.head()
```

date	co2
1958-03-29	316.1
1958-04-05	317.3
1958-04-12	317.6
1958-04-19	317.5
1958-04-26	316.4

Next steps: View recommended plots

```
plt.xlabel('Year')
plt.ylabel('Co2 analysis')
plt.plot(Lingyi_Co2_df)
print('missing data:',Lingyi_Co2_df.isnull().sum())
```

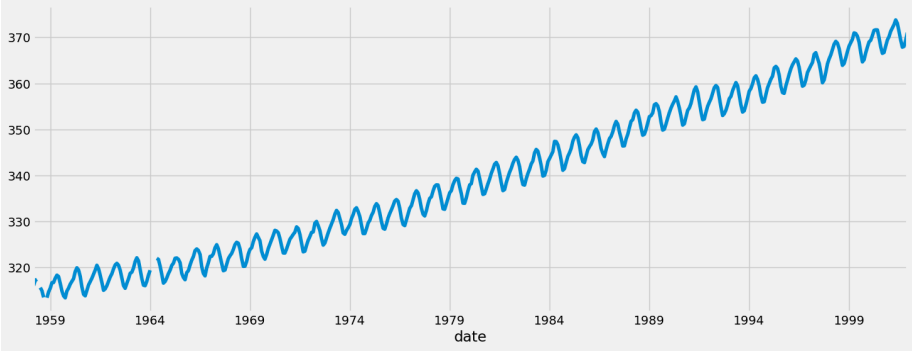
```
missing data: co2    59
dtype: int64
```




```
Lingyi_Co2_df
y=Lingyi_Co2_df['co2'].resample('MS').mean()
y.isnull().sum()
```

```
5
```

```
y.plot(figsize = (16,6))
plt.show()
```




```
y.head()
```

```
date
1958-03-01    316.100000
1958-04-01    317.200000
1958-05-01    317.433333
1958-06-01           NaN
1958-07-01    315.625000
Freq: MS, Name: co2, dtype: float64
```

```
type(y)
```

```
pandas.core.series.Series
def __init__(data=None, index=None, dtype: Dtype | None=None, name=None,
copy: bool=False, fastpath: bool=False) -> None

/usr/local/lib/python3.10/dist-packages/pandas/core/series.py
One-dimensional ndarray with axis labels (including time series).

Labels need not be unique but must be a hashable type. The object
supports both integer- and label-based indexing and provides a host of
```

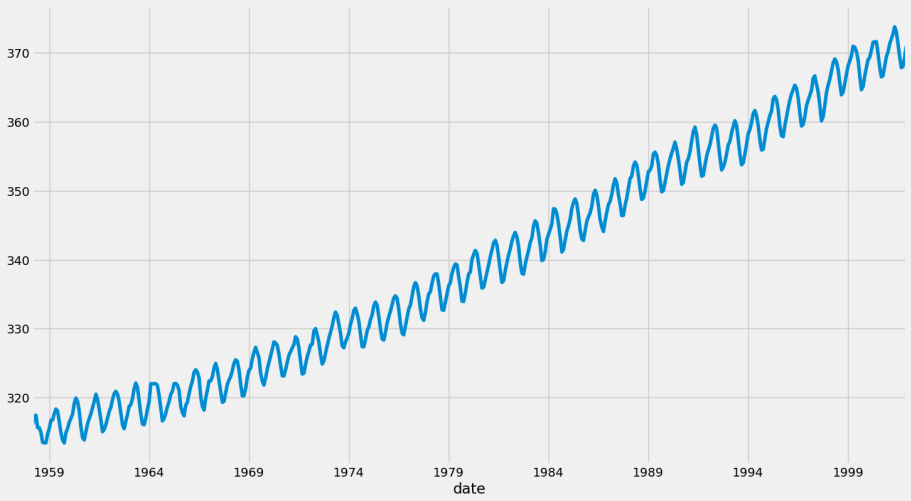
```
y.describe()
```

```
count    521.000000
mean     339.822665
std       17.068711
min      313.400000
25%      324.125000
50%      337.950000
75%      354.675000
max      373.800000
Name: co2, dtype: float64
```

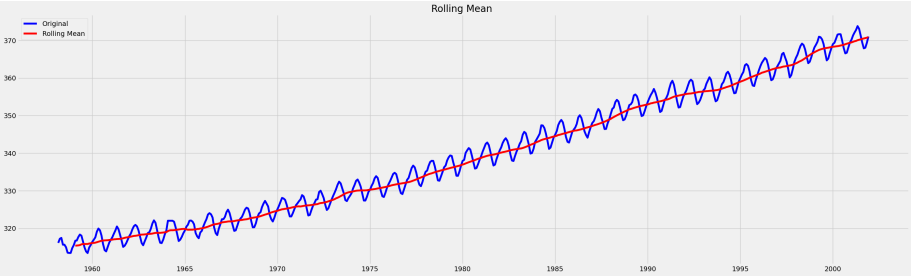
```
y.index
```

```
DatetimeIndex(['1958-03-01', '1958-04-01', '1958-05-01', '1958-06-01',
               '1958-07-01', '1958-08-01', '1958-09-01', '1958-10-01',
               '1958-11-01', '1958-12-01',
               ...
               '2001-03-01', '2001-04-01', '2001-05-01', '2001-06-01',
               '2001-07-01', '2001-08-01', '2001-09-01', '2001-10-01',
               '2001-11-01', '2001-12-01'],
              dtype='datetime64[ns]', name='date', length=526, freq='MS')
```

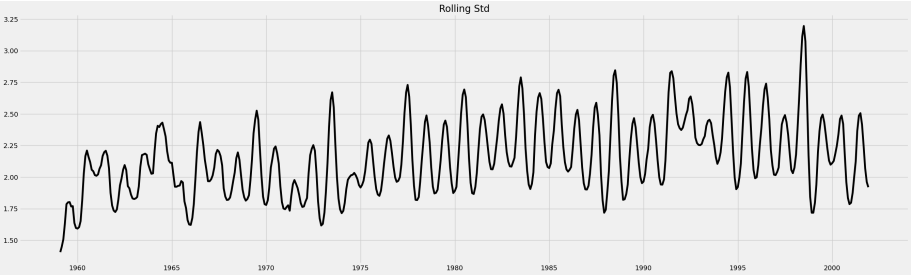
```
y=y.fillna(y.bfill())
y.plot(figsize = (16,9))
plt.show()
```



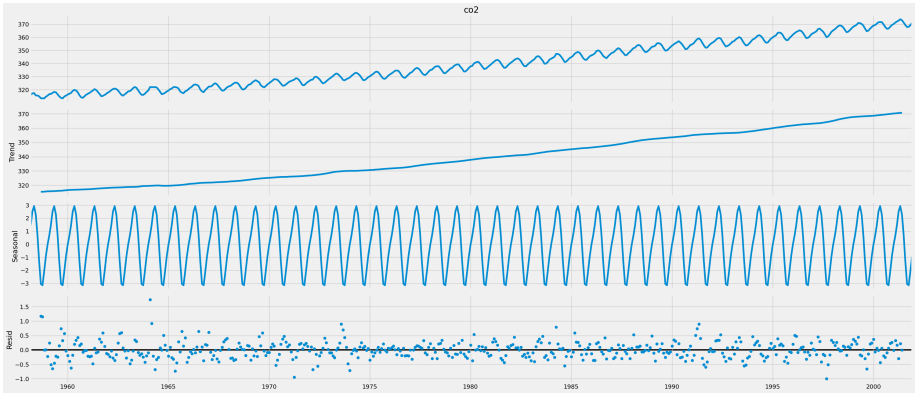
```
rollingmean = y.rolling(window=12).mean()
rollingstd = y.rolling(window=12).std()

orig = plt.plot(y, color='blue', label='Original')
mean = plt.plot(rollingmean , color='red', label='Rolling Mean')
#std = plt.plot(rollingstd, color='black', label='Rolling Std')
plt.rcParams["figure.figsize"] = (30,9)

plt.legend(loc='best')
plt.title('Rolling Mean ')
plt.show(block=False)
```




```
std = plt.plot(rollingstd,color = 'black',label = 'Rolling Std')
plt.title('Rolling Std')
plt.show(block = False)
```




```
deco = sm.tsa.seasonal_decompose(y,model = 'additive')
fig = deco.plot()
plt.rcParams['figure.figsize'] = (30,15)
plt.show()
```




```
deco.resid.describe()
```

```
count    514.000000
mean       0.002790
std        0.298683
min       -0.997063
25%       -0.195065
50%       -0.008551
75%        0.166299
max        1.745334
Name: resid, dtype: float64
```

```
deco.resid.isnull().sum()
```

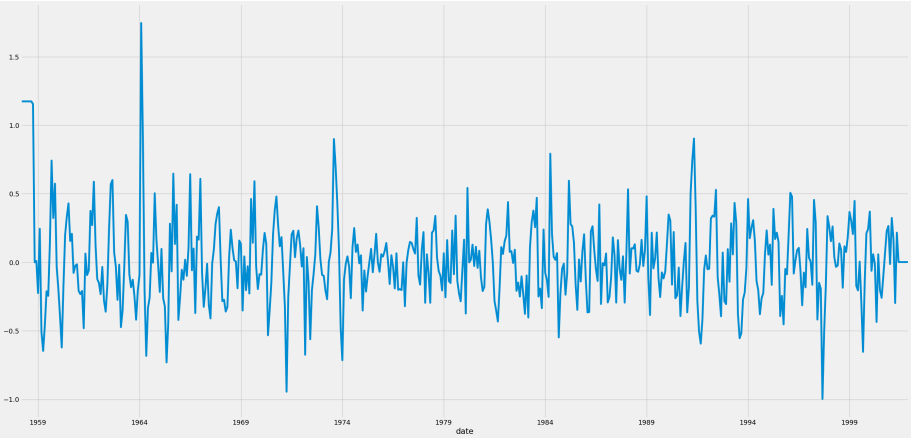
```
12
```

```
CO2Residual = deco.resid
CO2Residual = CO2Residual.fillna(CO2Residual.bfill())
CO2Residual = CO2Residual.fillna(CO2Residual.ffill())
```

```
CO2Residual.describe()
```

```
count    526.000000
mean       0.016137
std        0.320423
min       -0.997063
25%       -0.192372
50%       -0.000486
75%        0.169395
max        1.745334
Name: resid, dtype: float64
```

```
fig = CO2Residual.plot()
plt.show()
```



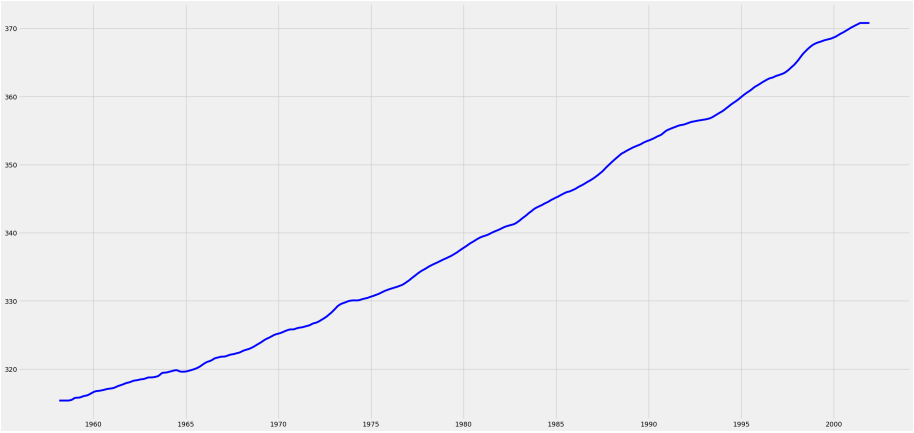

```
LingyiADFresult = adfuller(CO2Residual)


print('ADF Statistic:%f'% LingyiADFresult[0])
print('p_value:%f'% LingyiADFresult[1])
print('Critical Values:')
for key,value in LingyiADFresult[4].items():
  print('\t%s:%.3f'%(key,value))
```

```
    ADF Statistic:-12.181866
    p_value:0.000000
    Critical Values:
            1%:-3.443
            5%:-2.867
            10%:-2.570
```

```
CO2trend = deco.trend
CO2trend = CO2trend.fillna(CO2trend.bfill())
CO2trend = CO2trend.fillna(CO2trend.ffill())
CO2trend
```

```
    date
    1958-03-01    315.375000
    1958-04-01    315.375000
    1958-05-01    315.375000
    1958-06-01    315.375000
    1958-07-01    315.375000
                      ...
    2001-08-01    370.787917
    2001-09-01    370.787917
    2001-10-01    370.787917
    2001-11-01    370.787917
    2001-12-01    370.787917
    Freq: MS, Name: trend, Length: 526, dtype: float64
```

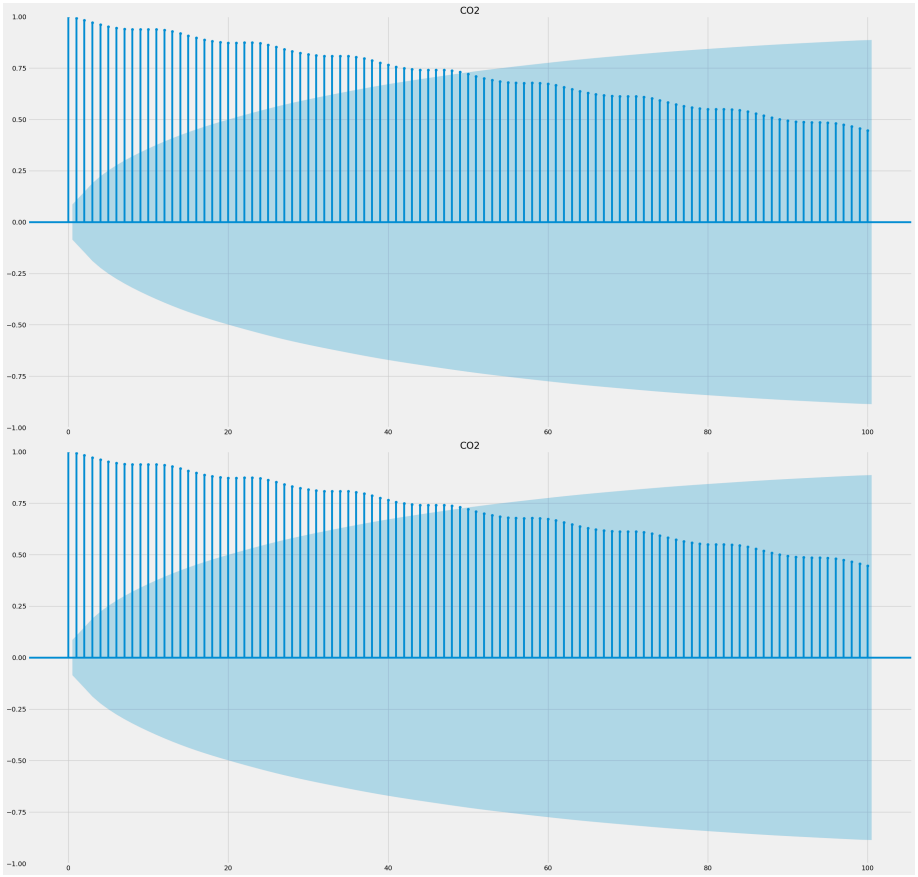
```
mean = plt.plot(CO2trend , color = 'blue',label = 'CO2trend')
```



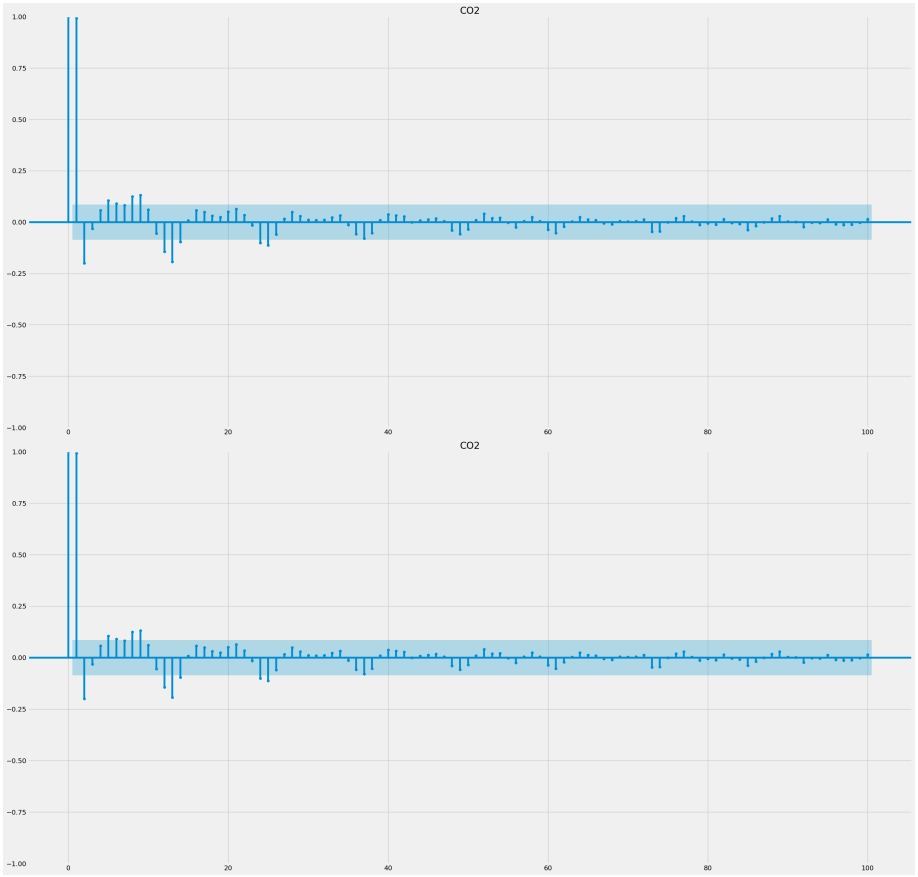

```
LingyiADFresult = adfuller(CO2trend)


print('ADF Statistic:%f'% LingyiADFresult[0])
print('p_value:%f'% LingyiADFresult[1])
print('Critical Values:')
for key,value in LingyiADFresult[4].items():
  print('\t%s:%.3f'%(key,value))
```

```
    ADF Statistic:1.761347
    p_value:0.998269
    Critical Values:
            1%:-3.443
            5%:-2.867
            10%:-2.570
```

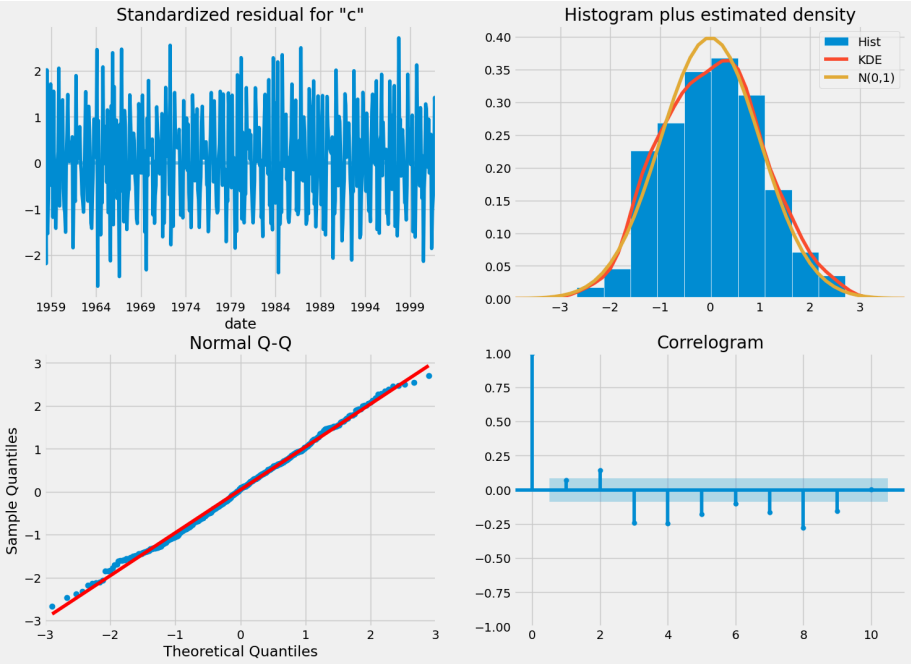
```
from statsmodels.graphics.tsaplots import plot_acf,plot_pacf
plot_acf(y,lags = 100,title = 'CO2')
```




```
plot_pacf(y,lags = 100,title = 'CO2')
```



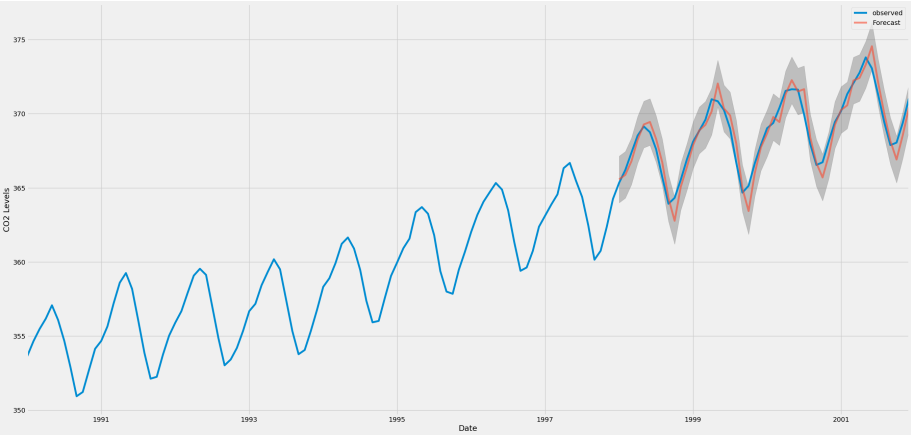

```
mod = sm.tsa.statespace.SARIMAX(y,order = (1,1,1),)
TSresults = mod.fit()
TSresults.plot_diagnostics(figsize = (16,12))
plt.show()
```




```
y
```

```
            date
1958-03-01    316.100000
1958-04-01    317.200000
1958-05-01    317.433333
1958-06-01    315.625000
1958-07-01    315.625000
                 ...
2001-08-01    369.425000
2001-09-01    367.880000
2001-10-01    368.050000
2001-11-01    369.375000
2001-12-01    371.020000
Freq: MS, Name: co2, Length: 526, dtype: float64
```

```
pred = TSresults.get_prediction(start = pd.to_datetime('1998-01-01'),dynamic = False)
pred_ci = pred.conf_int()
ax = y['1990':].plot(label = 'observed')
pred.predicted_mean.plot(ax=ax, label = 'Forecast',alpha = 0.6)

ax.fill_between(pred_ci.index, pred_ci.iloc[:, 0], pred_ci.iloc[:, 1], color='k', alpha=.2)
ax.set_xlabel('Date')
ax.set_ylabel('CO2 Levels')
plt.legend()

plt.show()
```




```
pred_ci#confident interval
```