In [42]:
```
# Import libraries and dataset
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.mlab as mlab
import statsmodels.api as sm
import math
import statsmodels.tsa.stattools as ts
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
from statsmodels.tsa.arima.model import ARIMA
from statsmodels.tsa.stattools import adfuller
from statsmodels.tsa.seasonal import seasonal_decompose

df = pd.read_csv("C:/Users/hkeim/OneDrive/Documents/School/D213/Task One/teleco_time_series.c
print(df.shape)
print(df.info())
print(df.head(5))
print(df.index)
```

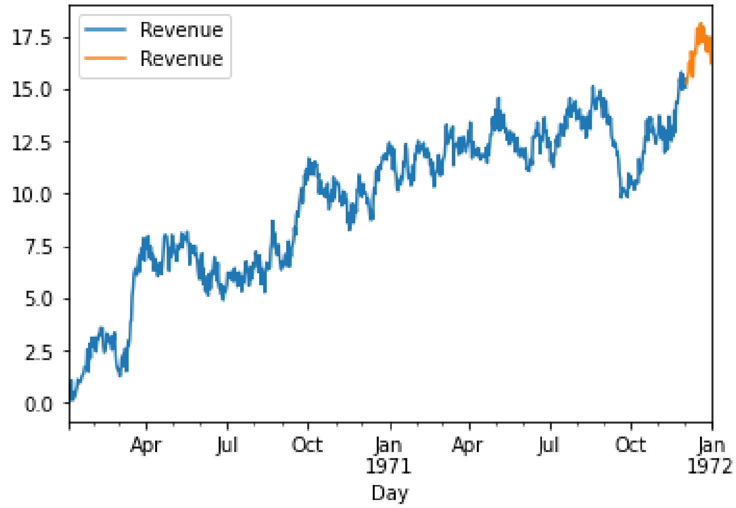
```
(731, 2)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 731 entries, 0 to 730
Data columns (total 2 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   Day      731 non-null    int64
 1   Revenue  731 non-null    float64
dtypes: float64(1), int64(1)
memory usage: 11.5 KB
None
   Day   Revenue
0    1  0.000000
1    2  0.000793
2    3  0.825542
3    4  0.320332
4    5  1.082554
RangeIndex(start=0, stop=731, step=1)
```

In [43]:
```
# Change 'Day' to datetime, set frequency
df['Day'] = pd.to_datetime(df['Day'], unit = 'D')
df['Day'] = pd.date_range('1970-01-02', periods = 731, freq = 'D')

# Set 'Day' as index
df.set_index('Day', inplace = True)
revenue = df.resample('D').last()
revenue.index
```

Out[43]:
```
DatetimeIndex(['1970-01-02', '1970-01-03', '1970-01-04', '1970-01-05',
               '1970-01-06', '1970-01-07', '1970-01-08', '1970-01-09',
               '1970-01-10', '1970-01-11',
               ...
               '1971-12-24', '1971-12-25', '1971-12-26', '1971-12-27',
               '1971-12-28', '1971-12-29', '1971-12-30', '1971-12-31',
               '1972-01-01', '1972-01-02'],
              dtype='datetime64[ns]', name='Day', length=731, freq='D')
```

In [44]:
```
# Split into train and test (30%) sets
revenue_train, revenue_test= np.split(revenue, [int(0.959 *len(revenue))])

# Create an axis
fig, ax = plt.subplots()
```

```
# Plot the train and test sets on the axis ax
revenue_train.plot(ax=ax)
revenue_test.plot(ax=ax)
plt.show()
```

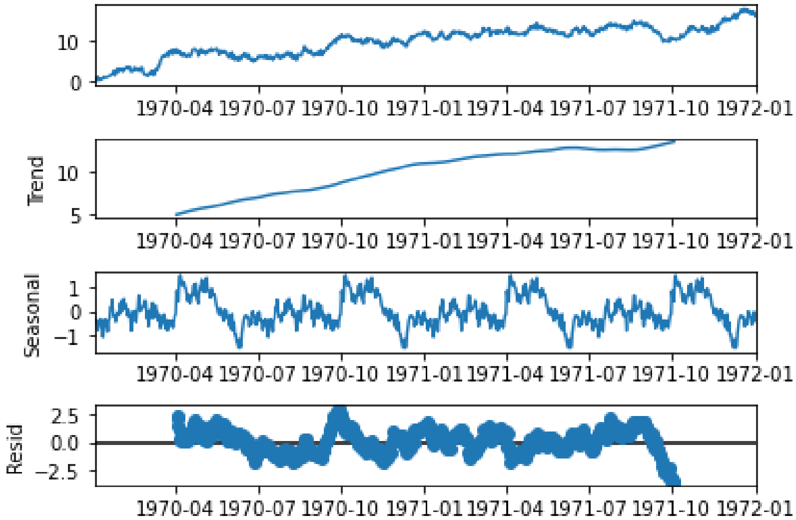


In [45]:
```
# Export clean data for report purposes
revenue.to_csv("C:/Users/hkeim/OneDrive/Documents/School/D213/Task One/Keim D213 Task One Cle
```

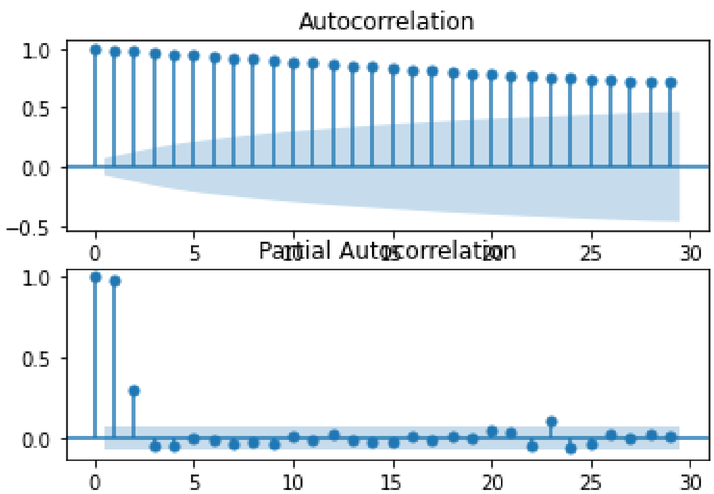
In [46]:
```
# Run the ADF test on the time series
result = adfuller(df)

# Print the test statistic and the p-value
print('ADF Statistic:', result[0])
print('p-value:', result[1])
print('Critical Values:')
for key, value in result[4].items():
        print('\t%s: %.3f' % (key, value))
```

```
ADF Statistic: -1.924612157310184
p-value: 0.3205728150793963
Critical Values:
        1%: -3.439
        5%: -2.866
        10%: -2.569
```

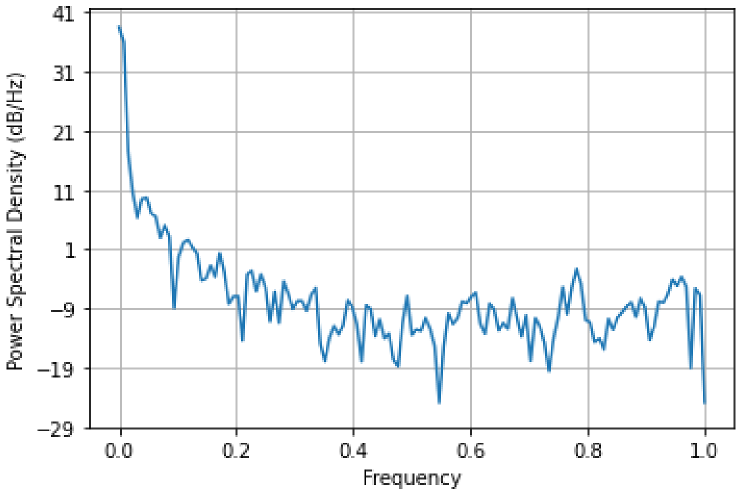
In [47]:
```
# Decompose time series
result=seasonal_decompose(df, model='additive', period=182).plot()
```



In [48]:
```
# Plot ACF and PACF of time series
series = revenue
plt.figure()
plt.subplot(211)
plot_acf(series, ax=plt.gca())
plt.subplot(212)
plot_pacf(series, ax=plt.gca())
plt.show()
```



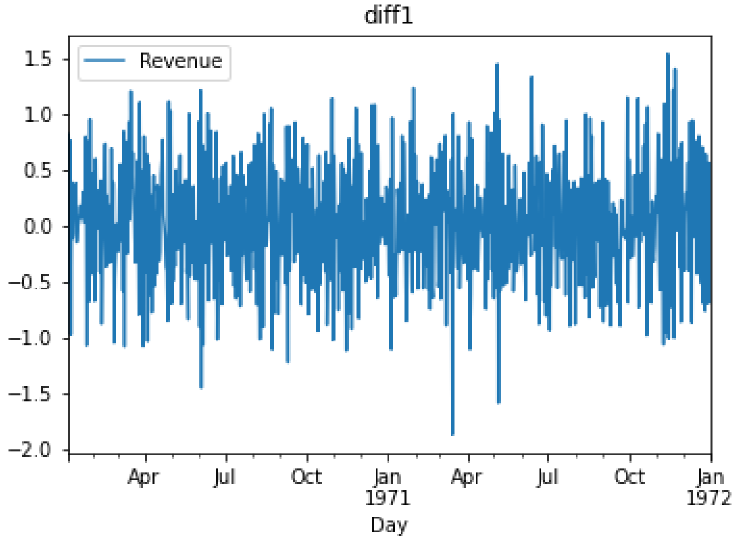
In [49]:
```
# Plot the spectral density of the time series
plt.psd(revenue['Revenue'])
plt.show()
```

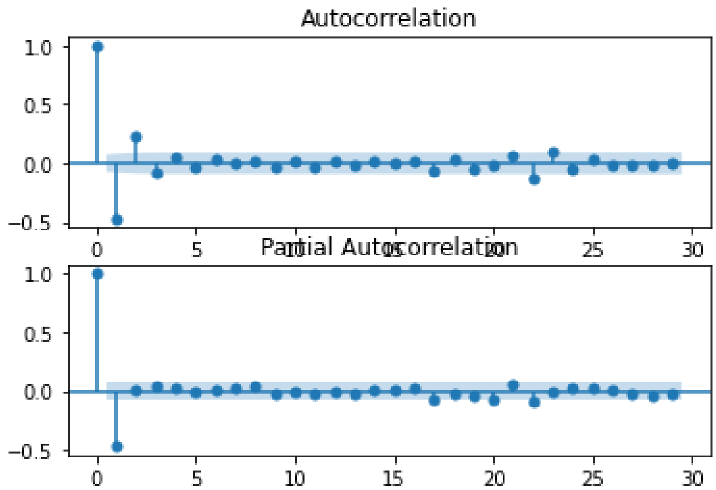


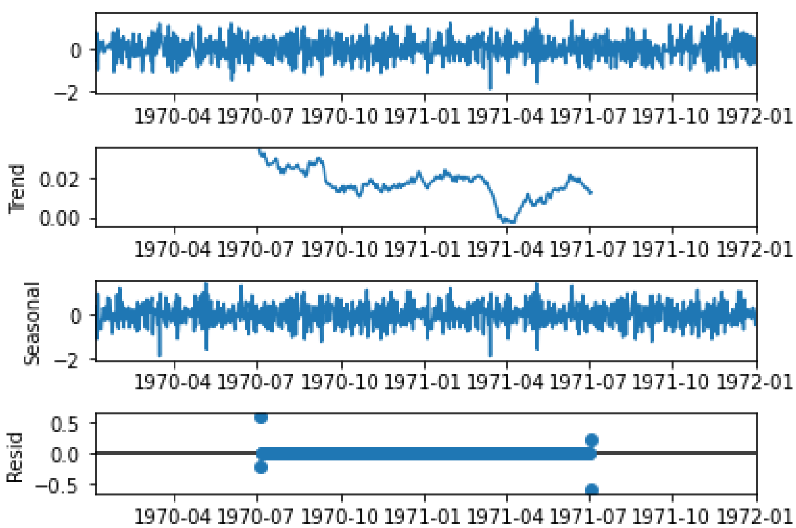
In [50]:

```
# Calculate the difference of the time series 1
revenue_diff1 = df.diff().dropna()

# Run ADF test on the differenced time series 1
result = adfuller(revenue_diff1)

# Print the test statistic and the p-value 1
print('ADF Statistic 1:', result[0])
print('p-value 1:', result[1])

# Plot the differenced time series 1
fig, ax = plt.subplots()
revenue_diff1.plot(ax=ax)
plt.title('diff1')
plt.show()

# Plot the ACF and PACF of the differenced time series 1
series = revenue_diff1
plt.figure()
plt.subplot(211)
plot_acf(series, ax=plt.gca())
plt.subplot(212)
plot_pacf(series, ax=plt.gca())
plt.show()

# decompose time series 1
result=seasonal_decompose(revenue_diff1, model='additive', period=364).plot()
```

```
ADF Statistic 1: -44.87452719387599
p-value 1: 0.0
```





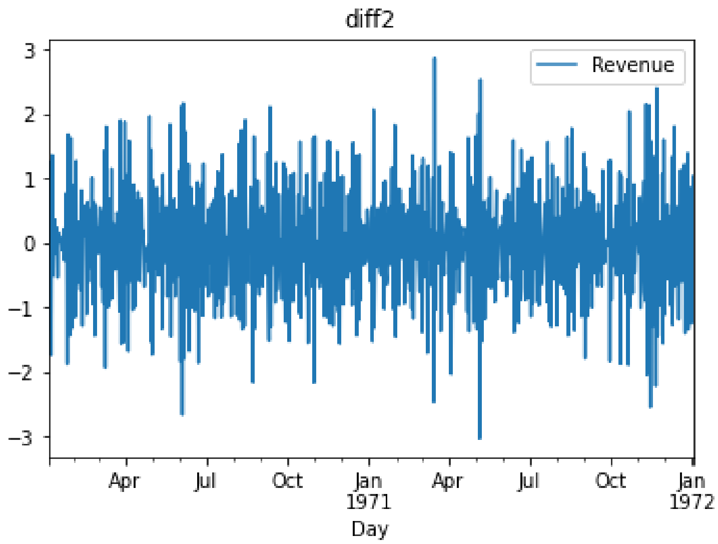


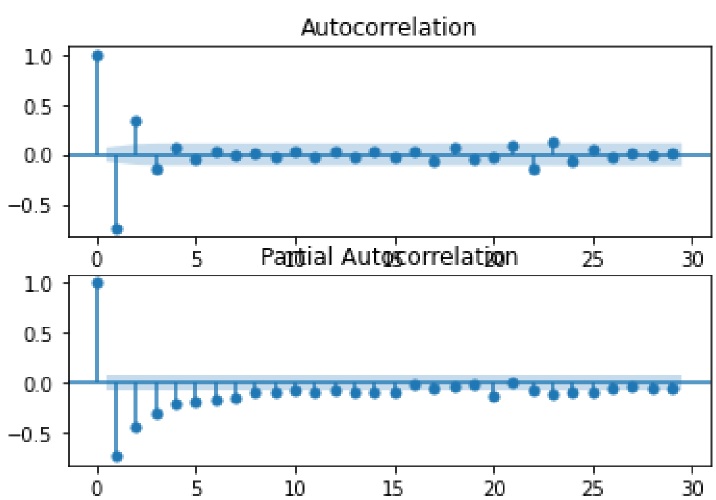
In [51]:

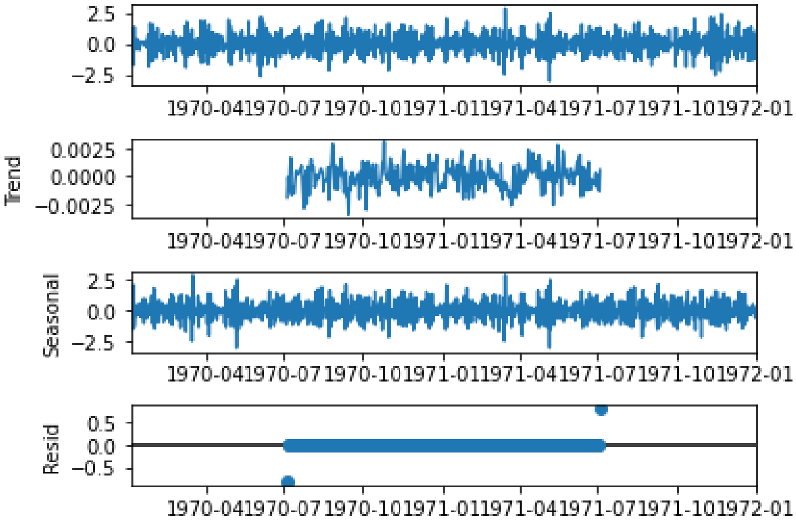
```
# Calculate the difference of the time series 2
revenue_diff2 = revenue_diff1.diff().dropna()

# Run ADF test on the differenced time series 2
result = adfuller(revenue_diff2)

# Print the test statistic and the p-value 2
print('ADF Statistic 2:', result[0])
print('p-value 2:', result[1])
```

```
# Plot the differenced time series 2
fig, ax = plt.subplots()
revenue_diff2.plot(ax=ax)
plt.title('diff2')
plt.show()

# Plot the ACF and PACF of the differenced time series 2
series = revenue_diff2
plt.figure()
plt.subplot(211)
plot_acf(series, ax=plt.gca())
plt.subplot(212)
plot_pacf(series, ax=plt.gca())
plt.show()

# decompose time series 2
result=seasonal_decompose(revenue_diff2, model='additive', period=364).plot()
```

```
ADF Statistic 2: -9.727189289782931
p-value 2: 9.207133401372169e-17
```







In [52]:

```
# Create ARIMA() model
arima = ARIMA(revenue_train, order=(2,2,2))

# Fit ARIMA model
arima_results = arima.fit()

# Print summary
print(arima_results.summary())
```

```
                               SARIMAX Results
==============================================================================
Dep. Variable:                Revenue   No. Observations:                  701
Model:                 ARIMA(2, 2, 2)   Log Likelihood                -468.534
Date:                Mon, 03 Jan 2022   AIC                            947.069
Time:                        12:54:23   BIC                            969.817
Sample:                    01-02-1970   HQIC                           955.863
                         - 12-03-1971
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1         -1.4459      0.105    -13.781      0.000      -1.651      -1.240
ar.L2         -0.4558      0.064     -7.119      0.000      -0.581      -0.330
ma.L1         -0.0238      0.112     -0.213      0.831      -0.243       0.195
ma.L2         -0.9758      0.117     -8.321      0.000      -1.206      -0.746
sigma2         0.2214      0.020     11.238      0.000       0.183       0.260
===================================================================================
Ljung-Box (L1) (Q):                   0.02   Jarque-Bera (JB):                 1.27
Prob(Q):                              0.89   Prob(JB):                         0.53
Heteroskedasticity (H):               0.97   Skew:                            -0.00
Prob(H) (two-sided):                  0.79   Kurtosis:                         2.79
===================================================================================

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

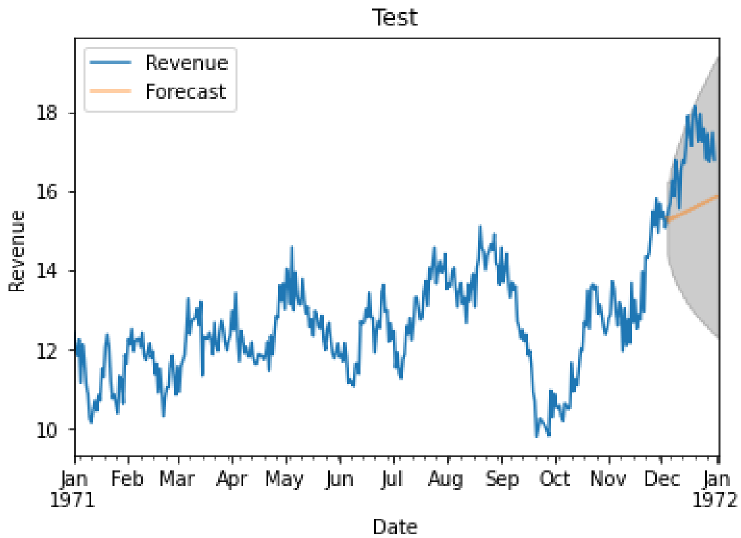
In [53]:

```
# Make ARIMA forecast of next 30 values
forecast = arima_results.get_forecast(steps = 30, dynamic = False)

# Summarize forecast and confidence intervals
ci = forecast.conf_int()
print(forecast.predicted_mean)
print(ci)

# Plot the forecast and confidence interval
```

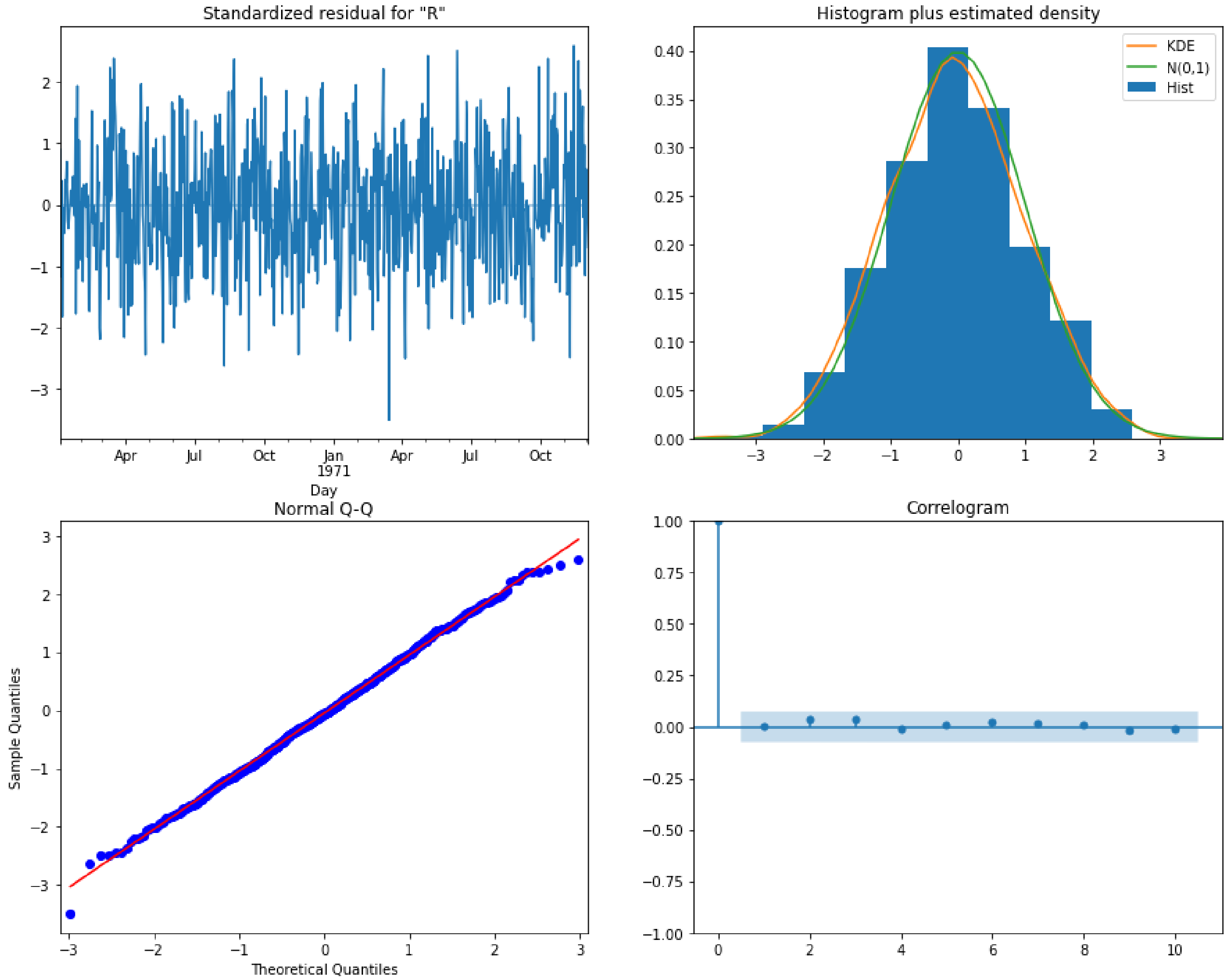
```
ax = revenue.loc['1971'].plot(label='observed')
forecast.predicted_mean.plot(ax=ax, label='Forecast', alpha=.5)

ax.fill_between(ci.index,
                ci.iloc[:, 0],
                ci.iloc[:, 1], color='k', alpha=.2)

ax.set_xlabel('Date')
ax.set_ylabel('Revenue')
plt.legend()
plt.title('Test')
plt.show()
```

```
1971-12-04    15.332352
1971-12-05    15.211518
1971-12-06    15.333336
1971-12-07    15.275360
1971-12-08    15.366737
1971-12-09    15.324122
1971-12-10    15.407165
1971-12-11    15.369599
1971-12-12    15.449140
1971-12-13    15.414335
1971-12-14    15.491480
1971-12-15    15.458882
1971-12-16    15.533929
1971-12-17    15.503358
1971-12-18    15.576430
1971-12-19    15.547790
1971-12-20    15.618970
1971-12-21    15.592186
1971-12-22    15.661545
1971-12-23    15.636548
1971-12-24    15.704153
1971-12-25    15.680877
1971-12-26    15.746794
1971-12-27    15.725174
1971-12-28    15.789465
1971-12-29    15.769442
1971-12-30    15.832166
1971-12-31    15.813680
1972-01-01    15.874896
1972-01-02    15.857891
Freq: D, Name: predicted_mean, dtype: float64
            lower Revenue  upper Revenue
1971-12-04      14.409652      16.255052
1971-12-05      14.166558      16.256478
1971-12-06      14.076999      16.589673
1971-12-07      13.884039      16.666680
1971-12-08      13.831009      16.902465
1971-12-09      13.667020      16.981224
1971-12-10      13.630329      17.184001
1971-12-11      13.484591      17.254607
1971-12-12      13.458264      17.440016
1971-12-13      13.325441      17.503230
1971-12-14      13.306239      17.676721
1971-12-15      13.183381      17.734383
1971-12-16      13.169258      17.898599
1971-12-17      13.054522      17.952194
1971-12-18      13.044112      18.108748
1971-12-19      12.936236      18.159344
1971-12-20      12.928587      18.309352
1971-12-21      12.826653      18.357719
1971-12-22      12.821082      18.502007
1971-12-23      12.724389      18.548707
1971-12-24      12.720389      18.687917
```

```
1971-12-25      12.628385      18.733369
1971-12-26      12.625574      18.868014
1971-12-27      12.537810      18.912539
1971-12-28      12.535893      19.043037
1971-12-29      12.452000      19.086884
1971-12-30      12.450747      19.213585
1971-12-31      12.370413      19.256948
1972-01-01      12.369641      19.380150
1972-01-02      12.292599      19.423182
```



In [54]:
```
# Diagnostic plots
residuals = arima_results.resid
mae = np.mean(np.abs(residuals))
arima_results.plot_diagnostics(figsize=(15, 12))
plt.show()
print(mae)
```



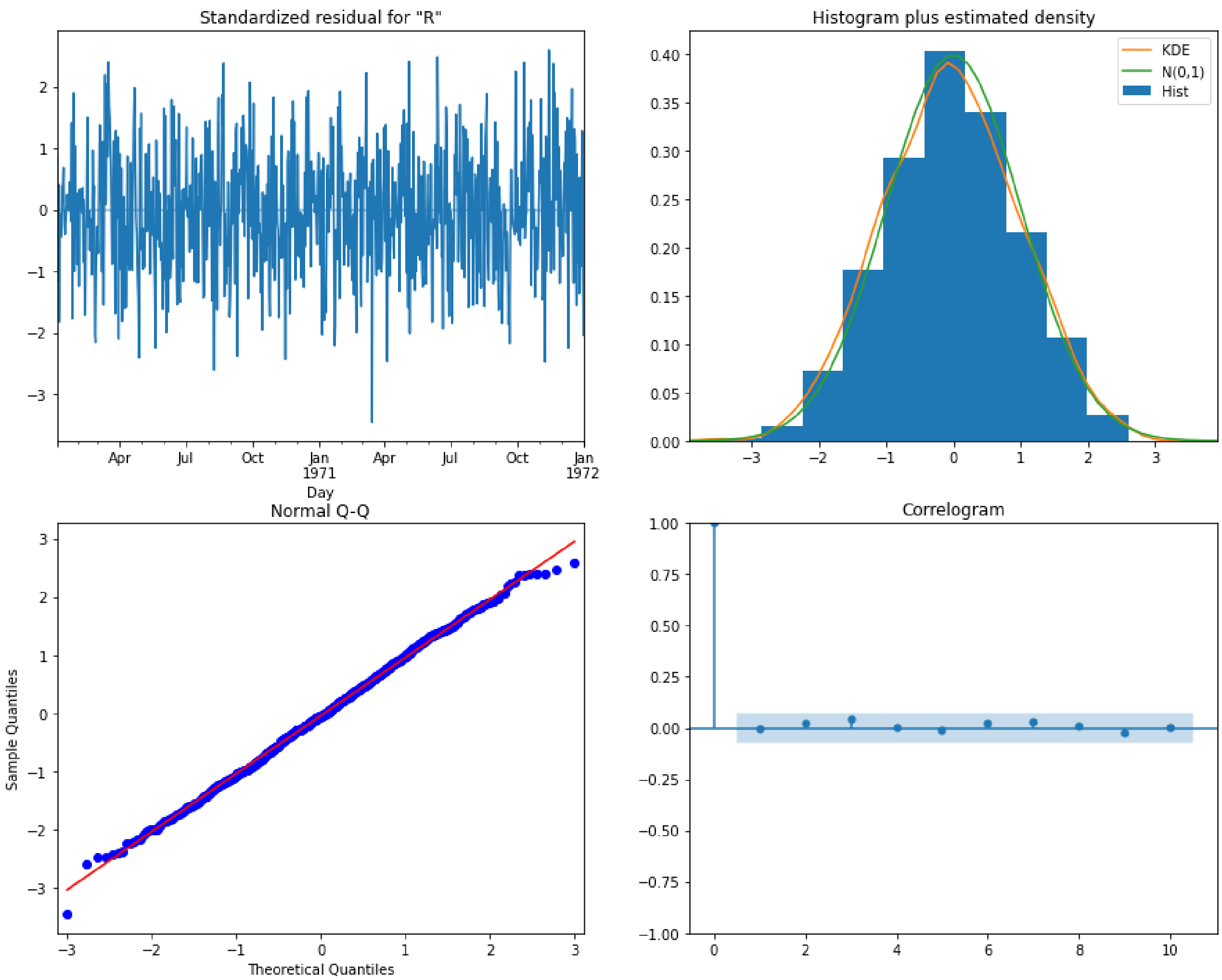

```
0.3782221089472484
```

In [55]:

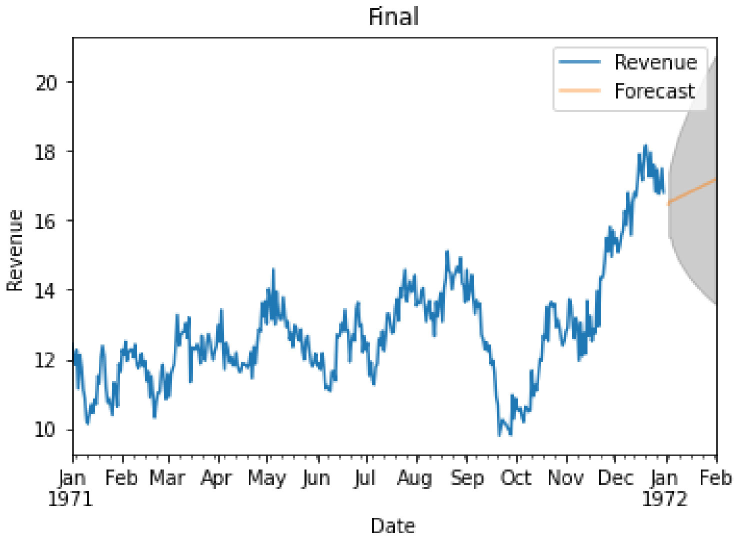
```python
# Final ARIMA model
final = ARIMA(revenue, order=(2,2,2))
results = final.fit()
fin_forecast = results.get_forecast(steps = 30, dynamic = False)
fin_ci = fin_forecast.conf_int()

# Print summary
print(arima_results.summary())

# Diagnostic plots
residuals_fin = results.resid
mae_fin = np.mean(np.abs(residuals_fin))
results.plot_diagnostics(figsize=(15, 12))
plt.show()
print(mae_fin)

# Plot the forecast and confidence interval
ax = revenue.loc['1971'].plot(label='observed')
fin_forecast.predicted_mean.plot(ax=ax, label='Forecast', alpha=.5)

ax.fill_between(fin_ci.index,
                fin_ci.iloc[:, 0],
                fin_ci.iloc[:, 1], color='k', alpha=.2)

ax.set_xlabel('Date')
ax.set_ylabel('Revenue')
plt.legend()
```

```
plt.title('Final')
plt.show()
```

```
                               SARIMAX Results
==============================================================================
Dep. Variable:                Revenue   No. Observations:                  701
Model:                 ARIMA(2, 2, 2)   Log Likelihood                -468.534
Date:                Mon, 03 Jan 2022   AIC                            947.069
Time:                        12:54:25   BIC                            969.817
Sample:                    01-02-1970   HQIC                           955.863
                         - 12-03-1971
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1         -1.4459      0.105    -13.781      0.000      -1.651      -1.240
ar.L2         -0.4558      0.064     -7.119      0.000      -0.581      -0.330
ma.L1         -0.0238      0.112     -0.213      0.831      -0.243       0.195
ma.L2         -0.9758      0.117     -8.321      0.000      -1.206      -0.746
sigma2         0.2214      0.020     11.238      0.000       0.183       0.260
===================================================================================
Ljung-Box (L1) (Q):                   0.02   Jarque-Bera (JB):                 1.27
Prob(Q):                              0.89   Prob(JB):                         0.53
Heteroskedasticity (H):               0.97   Skew:                            -0.00
Prob(H) (two-sided):                  0.79   Kurtosis:                         2.79
===================================================================================

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




```
0.38097653837413714
```


Final
Revenue
Forecast
20
18
16
14
12
10
Revenue
Jan
1971
Feb
Mar
Apr
May
Jun
Jul
Aug
Sep
Oct
Nov
Dec
Jan
1972
Feb
Date

In [ ]: