Times Series Forecasting and Visualization

In [1]:

```
import pandas as pd

# Loading the data from Excel file
data = pd.read_excel('ExampleOilMeasurementDaily.xlsx',header=None)
```

In [2]:

```
data.head()
```

Out[2]:

	0	1	2
0	Oil	2018-10-01	1329.6
1	Oil	2018-10-02	1435.2
2	Oil	2018-10-03	1372.1
3	Oil	2018-10-04	1371.7
4	Oil	2018-10-05	1200.0

In [3]:

```
data.describe()
```

Out[3]:

	2
count	1267.000000
mean	272.174191
std	213.450613
min	-8.100000
25%	125.900000
50%	206.600000
75%	364.900000
max	1435.200000

In [4]:

```
print(data.isna().sum())
```

```
0    0
1    0
2    0
dtype: int64
```

In [7]:

```
# renaming columns since they don't have any name to be more clear
data = data.rename(columns={0: 'Type', 1: 'Date', 2: 'Measurement'})

data.head()
```

Out[7]:

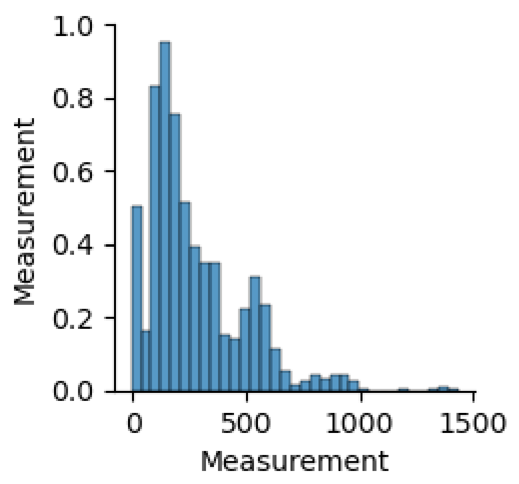
	Type	Date	Measurement
0	Oil	2018-10-01	1329.6
1	Oil	2018-10-02	1435.2
2	Oil	2018-10-03	1372.1
3	Oil	2018-10-04	1371.7
4	Oil	2018-10-05	1200.0

In [8]:

```
import seaborn as sns
# visualize data
sns.pairplot(data)
```

Out[8]:

```
<seaborn.axisgrid.PairGrid at 0x7fae91f51970>
```

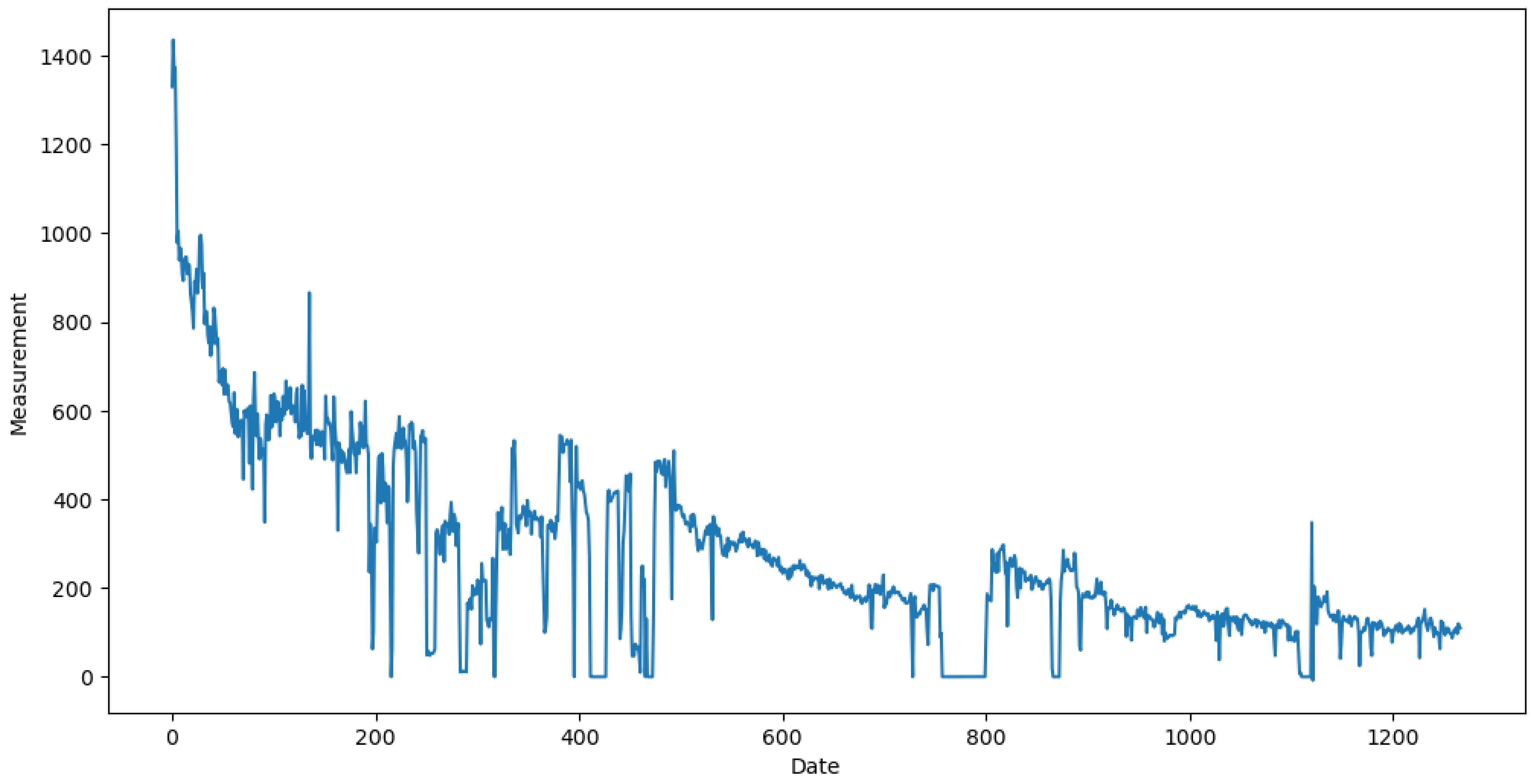


In [9]:

```
# Converting datetime column to datetime data type
data['Date'] = pd.to_datetime(data['Date'])
```

In [14]:

```
# Plotting the timeseries data
import matplotlib.pyplot as plt
plt.figure(figsize=(12, 6))
plt.plot(data['Measurement'].values.tolist())
plt.xlabel('Date')
plt.ylabel('Measurement')
plt.show()
```



In [15]:

```
from statsmodels.tsa.arima.model import ARIMA
# Fitting ARIMA model
model = ARIMA(data['Measurement'], order=(2, 2, 1))
model_fit = model.fit()
```

In [16]:

```
# Making a forecast
forecast = model_fit.forecast(steps=1)

print("ARIMA Forecast: ", forecast)
```

```
ARIMA Forecast:  1267    109.977433
dtype: float64
```

In [17]:

```
from statsmodels.tsa.holtwinters import ExponentialSmoothing

# Fitting Exponential Smoothing model
model = ExponentialSmoothing(data['Measurement'], trend='add', seasonal=None)
model_fit = model.fit()
```

In [18]:

```
# Making predictions
forecast = model_fit.forecast(steps=1)

# Printing the forecasted value
print("Forecasted Value:", forecast)
```

```
Forecasted Value: 1267    110.681439
dtype: float64
```

Forecasted Measurement using ARIMA model: 109.977433

Forecasted Measurement using Smoothing model: 110.681439