Actividad Módulo 30: Introducción a Series de Tiempo - Parte 1

1. Importación de datos
2. Ejercicios de limpieza de datos y visualización básica, inclusión de timestamps
3. Rangos de datos, shifting y lags, resampling
4. Visualización de % Cambios y retornos
5. Técnicas de comparación de series
6. Gráficos OHLC y candlestick
7. Descomposición de Series

In [ ]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
```

In [ ]:
```
# Importación de datos
df_ecopetrol = pd.read_csv('ECOPETROL.CL.csv', parse_dates=['Date'], index_col='Dat
df_trepel = pd.read_csv('TERPEL.CL.csv', parse_dates=['Date'], index_col='Date')

df_trepel
```

Out[ ]:

	Open	High	Low	Close	Adj Close	Volume
Date						
2015-01-01	17600.000000	17600.000000	17600.000000	17600.000000	9815.918945	0.0
2015-01-02	17600.000000	17600.000000	17600.000000	17600.000000	9815.918945	90901.0
2015-01-05	17400.000000	17400.000000	17400.000000	17400.000000	9704.375000	50000.0
2015-01-06	17400.000000	17400.000000	17400.000000	17400.000000	9704.375000	0.0
2015-01-07	17400.000000	17400.000000	17400.000000	17400.000000	9704.375000	38000.0
...	...	...	...	...	...	...
2022-12-26	7900.000000	7900.000000	7900.000000	7900.000000	7042.458984	6879.0
2022-12-27	7835.310059	7835.310059	7835.310059	7835.310059	6984.791504	2905.0
2022-12-28	8030.000000	8030.000000	8030.000000	8030.000000	7158.347656	6352.0
2022-12-29	8030.000000	8030.000000	8030.000000	8030.000000	7158.347656	6352.0
2022-12-30	8030.000000	8030.000000	8030.000000	8030.000000	7158.347656	0.0

2087 rows × 6 columns

In [ ]:
```
# Set option to display only 2 decimals for all pandas
pd.set_option('display.float_format', lambda x: '%.2f' % x)
```

EDA: Análisis Univariado

In [ ]:
```
df_ecopetrol.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 3914 entries, 2008-01-01 to 2022-12-30
Data columns (total 6 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   Open       3901 non-null   float64
 1   High       3901 non-null   float64
 2   Low        3901 non-null   float64
 3   Close      3901 non-null   float64
 4   Adj Close  3901 non-null   float64
 5   Volume     3901 non-null   float64
dtypes: float64(6)
memory usage: 214.0 KB
```

In [ ]:
```
df_ecopetrol.describe().T
```

Out[ ]:

	count	mean	std	min	25%	50%	75%	
Open	3901.00	2849.97	1115.89	881.00	2055.00	2690.00	3555.00	5
High	3901.00	2898.88	1447.54	931.00	2080.00	2715.00	3600.00	5(
Low	3901.00	2818.99	1108.08	880.00	2020.00	2660.00	3510.00	5
Close	3901.00	2850.15	1116.05	881.00	2055.00	2690.00	3550.00	5
Adj Close	3901.00	2000.46	735.56	643.21	1321.84	2126.33	2483.96	3
Volume	3901.00	20192058.45	651714136.69	0.00	4769304.00	7819581.00	12479255.00	4071026(

In [ ]:
```
# 13 null values in each column of the df, check those null rows
df_ecopetrol[df_ecopetrol.isnull().any(axis=1)]
```

Out[ ]:

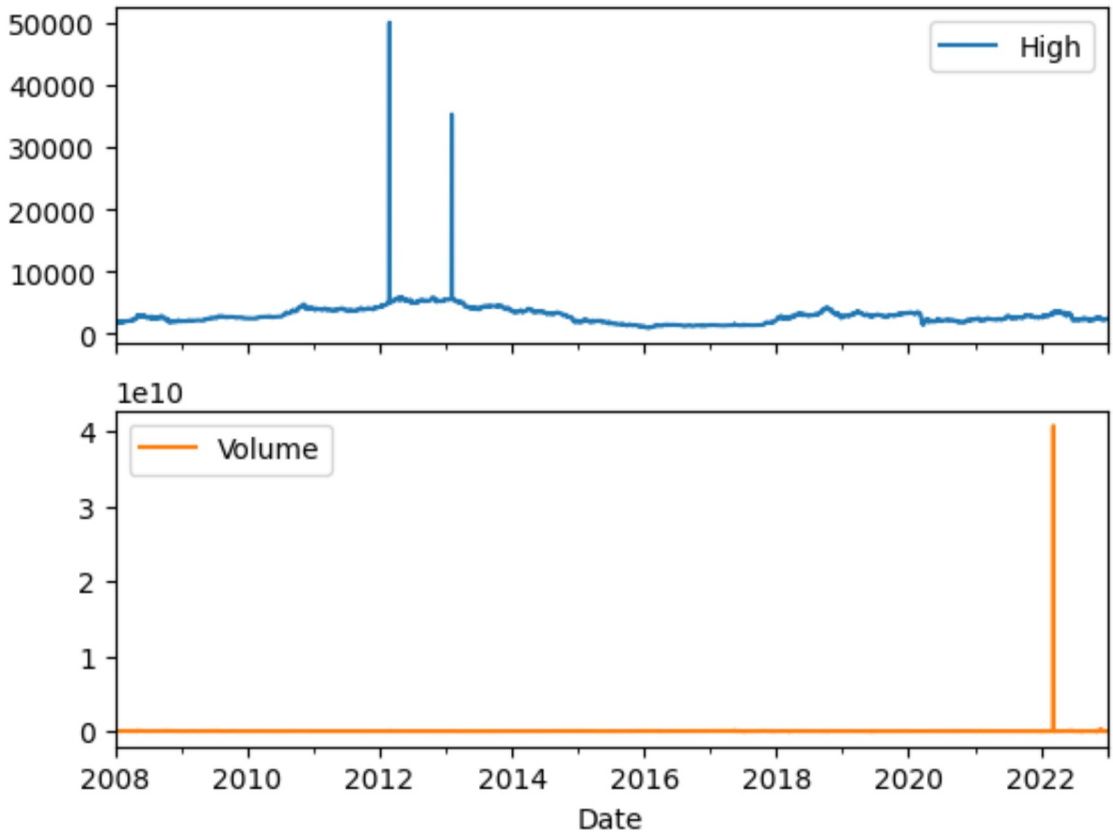
	Open	High	Low	Close	Adj Close	Volume
Date						
2017-05-15	NaN	NaN	NaN	NaN	NaN	NaN
2017-07-05	NaN	NaN	NaN	NaN	NaN	NaN
2017-12-11	NaN	NaN	NaN	NaN	NaN	NaN
2018-03-13	NaN	NaN	NaN	NaN	NaN	NaN
2018-10-08	NaN	NaN	NaN	NaN	NaN	NaN
2018-10-09	NaN	NaN	NaN	NaN	NaN	NaN
2019-06-10	NaN	NaN	NaN	NaN	NaN	NaN
2022-01-10	NaN	NaN	NaN	NaN	NaN	NaN
2022-03-21	NaN	NaN	NaN	NaN	NaN	NaN
2022-06-20	NaN	NaN	NaN	NaN	NaN	NaN
2022-08-15	NaN	NaN	NaN	NaN	NaN	NaN
2022-11-07	NaN	NaN	NaN	NaN	NaN	NaN
2022-11-14	NaN	NaN	NaN	NaN	NaN	NaN

In [ ]:

```
# A really high maximum was detected in the High and Volume Series
df_ecopetrol[['High', 'Volume']].plot(subplots=True)
```

Out[ ]:

```
array([<AxesSubplot: xlabel='Date'>, <AxesSubplot: xlabel='Date'>],
      dtype=object)
```



In [ ]:
```
# Borrar valores atípicos detectados
# Solamente se cambiarán los valores de los picos en ambas series con el método bac
df_ecopetrol.loc[df_ecopetrol['High'] > 20000, 'High'] = np.nan
df_ecopetrol['High'] = df_ecopetrol['High'].fillna(method='backfill')

df_ecopetrol.loc[df_ecopetrol['Volume'] > 100000000, 'Volume'] = np.nan
df_ecopetrol['Volume'] = df_ecopetrol['Volume'].fillna(method='backfill')
```

In [ ]:
```
df_trepel.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 2087 entries, 2015-01-01 to 2022-12-30
Data columns (total 6 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   Open       2077 non-null   float64
 1   High       2077 non-null   float64
 2   Low        2077 non-null   float64
 3   Close      2077 non-null   float64
 4   Adj Close  2077 non-null   float64
 5   Volume     2077 non-null   float64
dtypes: float64(6)
memory usage: 114.1 KB
```

In [ ]:
```
df_trepel.describe().T
```

Out[ ]:

	count	mean	std	min	25%	50%	75%	max
Open	2077.00	10422.91	2504.96	6300.00	8385.00	9940.00	12000.00	17600.00
High	2077.00	10447.80	2495.12	6300.00	8400.00	9990.00	12000.00	17600.00
Low	2077.00	10373.89	2498.33	6300.00	8330.00	9930.00	12000.00	17600.00
Close	2077.00	10422.98	2504.97	6300.00	8385.00	9940.00	12000.00	17600.00
Adj Close	2077.00	6983.46	1172.07	4555.85	6122.81	6892.46	7781.98	9823.79
Volume	2077.00	301626.86	12725978.37	0.00	0.00	2292.00	14748.00	579973730.00

In [ ]:

```
# 10 rows with null values for each of the columns, check those rows
df_trepel[df_trepel.isnull().any(axis=1)]
```

Out[ ]:

Date	Open	High	Low	Close	Adj Close	Volume
2017-05-15	NaN	NaN	NaN	NaN	NaN	NaN
2018-03-13	NaN	NaN	NaN	NaN	NaN	NaN
2018-10-08	NaN	NaN	NaN	NaN	NaN	NaN
2018-10-09	NaN	NaN	NaN	NaN	NaN	NaN
2022-01-10	NaN	NaN	NaN	NaN	NaN	NaN
2022-03-21	NaN	NaN	NaN	NaN	NaN	NaN
2022-06-20	NaN	NaN	NaN	NaN	NaN	NaN
2022-08-15	NaN	NaN	NaN	NaN	NaN	NaN
2022-11-07	NaN	NaN	NaN	NaN	NaN	NaN
2022-11-14	NaN	NaN	NaN	NaN	NaN	NaN

In [ ]:

```
# Eliminación del valor atípico encontrado en la columna de Volume
df_trepel.loc[df_trepel['Volume'] > 500000000, 'Volume'] = np.nan

df_trepel['Volume'].fillna(method='bfill')
```

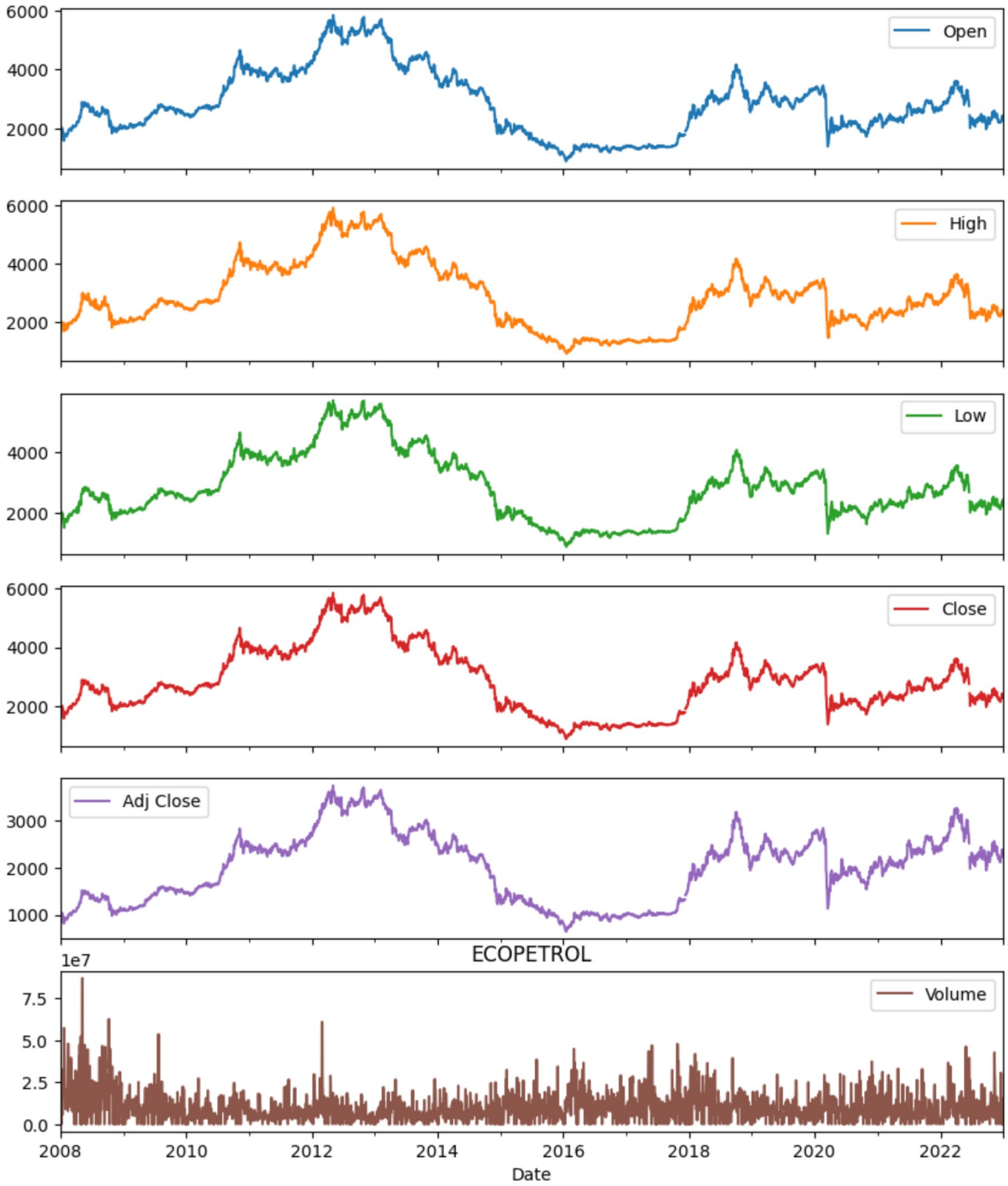
Out[ ]:

```
Date
2015-01-01       0.00
2015-01-02   90901.00
2015-01-05   50000.00
2015-01-06       0.00
2015-01-07   38000.00
                ...
2022-12-26    6879.00
2022-12-27    2905.00
2022-12-28    6352.00
2022-12-29    6352.00
2022-12-30       0.00
Name: Volume, Length: 2087, dtype: float64
```

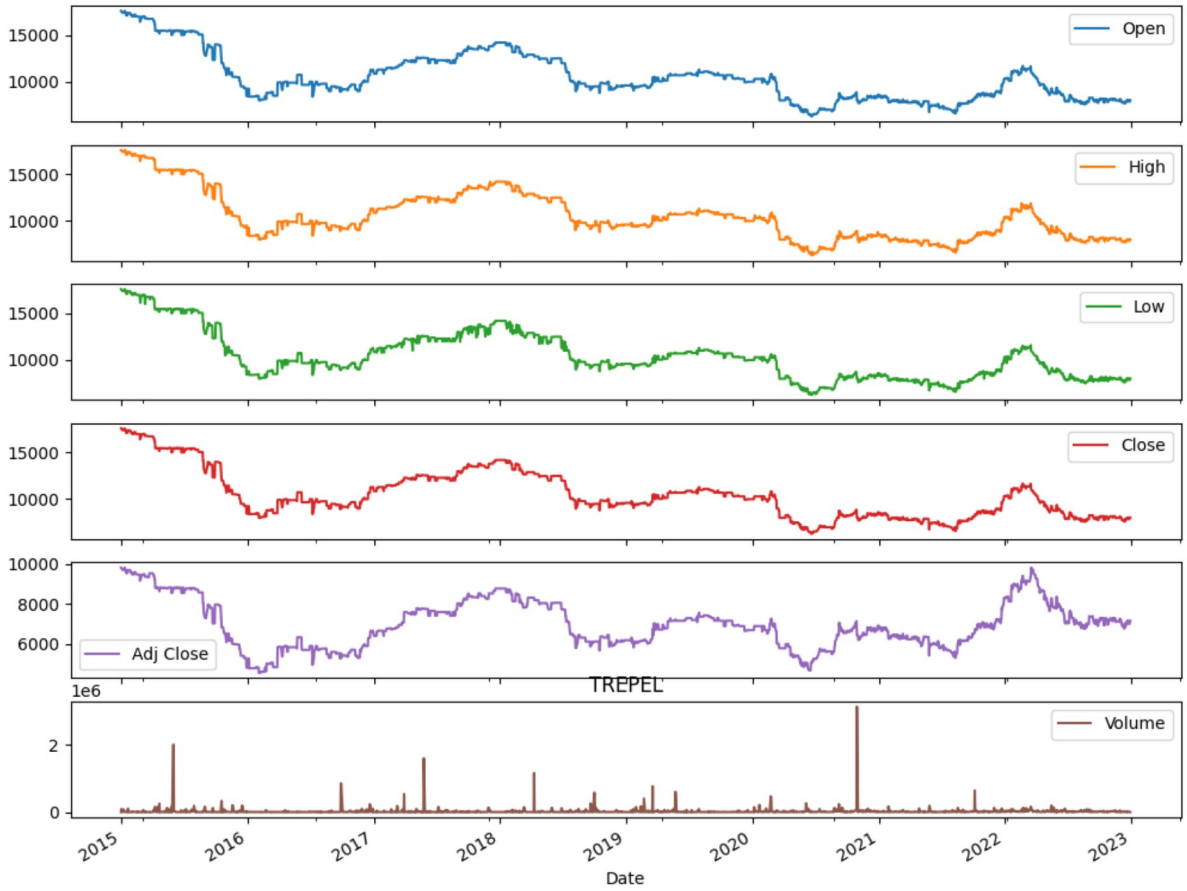
In [ ]:
```
# Eliminación de registros en blanco
df_ecopetrol = df_ecopetrol[df_ecopetrol.notnull().any(axis=1)]
df_trepel = df_trepel[df_trepel.notnull().any(axis=1)]
```

Visualización de Series de Tiempo

In [ ]:
```
df_ecopetrol.plot(subplots=True, figsize=(10,12))
plt.title('ECOPETROL')
plt.show()
```



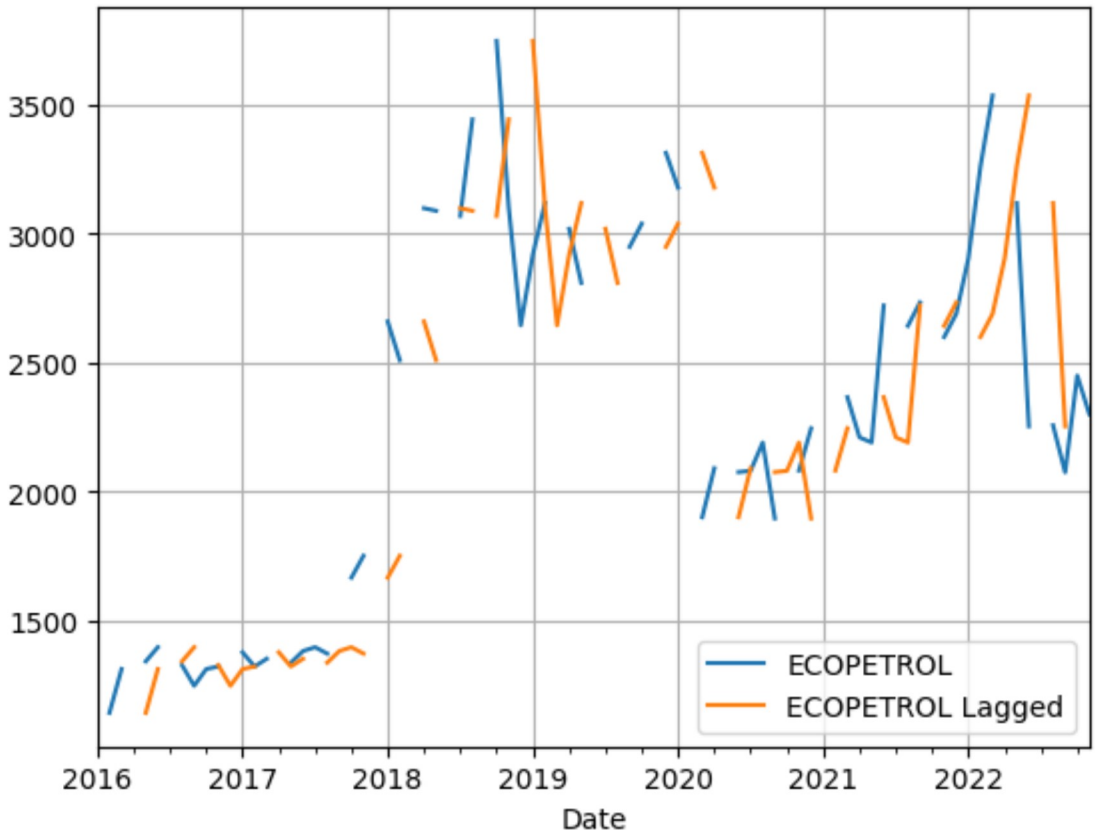
In [ ]:
```
df_trepel.plot(subplots=True, figsize=(12,10))
plt.title('TREPEL')
plt.show()
```



In [ ]:
```
# 3. Rangos de datos, shifting y lags, resampling
# Rangos - Igualar ventana de tiempo de ambas series de tiempo
df_ecopetrol_original = df_ecopetrol.copy()
df_trepel_original = df_trepel.copy()

df_ecopetrol = df_ecopetrol.loc['2016':]
df_trepel = df_trepel.loc['2016':]
```

In [ ]:
```
# Shifting
```

In [ ]:
```
# shifting and lag
lag = 3 #meses

df_ecopetrol['Close'].asfreq('M').plot(legend=True)
shifted = df_ecopetrol['Close'].asfreq('M').shift(lag).plot(legend=True)
shifted.legend(['ECOPETROL', 'ECOPETROL Lagged'])
plt.grid()
plt.show()
```



In [ ]:
```
# Resampling
# Cambiar la frecuencia de las observaciones (de días a meses, o a años)
# Downsampling (decreasing frequency - from days to months)
df_trepel.resample('Q').mean()      # Q for Quarter

# Puedes usar otros métodos como mínimo, máximo, etc.
```

Out[ ]:

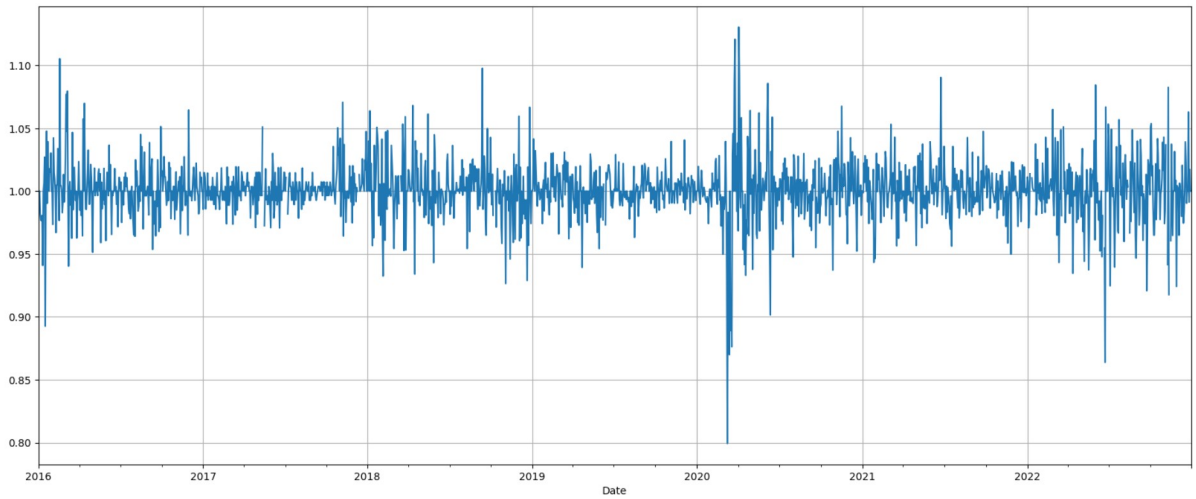
Date	Open	High	Low	Close	Adj Close	Volume
2016-03-31	8542.15	8542.15	8515.69	8542.15	4873.89	2775.17
2016-06-30	9924.31	9925.08	9887.38	9924.31	5852.79	3603.72
2016-09-30	9523.03	9528.79	9502.42	9523.03	5616.14	38412.95
2016-12-31	9763.54	9766.46	9736.77	9763.54	5757.97	16963.22
2017-03-31	11453.54	11461.54	11392.62	11453.54	6765.19	16848.80
2017-06-30	12345.00	12350.00	12282.81	12345.00	7624.48	31911.11
2017-09-30	12546.15	12550.77	12500.92	12546.15	7748.72	9128.86
2017-12-31	13705.54	13713.54	13553.85	13705.54	8464.78	6633.52
2018-03-31	13514.06	13517.50	13406.25	13514.06	8379.03	9451.98
2018-06-30	12488.00	12488.62	12394.15	12488.00	8048.59	30373.37
2018-09-30	10081.54	10130.77	10041.85	10081.54	6497.61	20829.28
2018-12-31	9531.72	9562.03	9518.28	9531.72	6143.25	26530.44
2019-03-31	9627.81	9635.94	9581.56	9627.81	6229.18	37571.20
2019-06-30	10535.69	10546.77	10514.46	10537.23	7051.21	23865.75
2019-09-30	10918.18	10924.24	10916.97	10918.18	7306.13	5817.24
2019-12-31	10362.88	10388.64	10357.27	10362.88	6934.53	4102.58
2020-03-31	9823.69	9856.62	9798.46	9823.69	6626.77	28380.05
2020-06-30	7254.92	7299.54	7226.46	7254.92	5309.81	17450.26
2020-09-30	7396.97	7431.52	7328.94	7396.97	5948.40	25145.35
2020-12-31	8266.82	8324.02	8196.74	8266.82	6647.91	70860.92
2021-03-31	8048.05	8074.53	8019.45	8048.05	6471.98	11477.08
2021-06-30	7607.85	7625.92	7576.92	7607.85	6117.98	10391.74
2021-09-30	7370.53	7414.24	7332.12	7370.53	5927.14	14415.12
2021-12-31	8774.17	8818.79	8708.03	8774.17	7055.90	27984.39
2022-03-31	10948.23	11063.06	10879.68	10948.23	8940.20	36314.66
2022-06-30	9058.44	9152.73	9010.86	9058.44	8075.15	23708.36
2022-09-30	7974.77	8010.46	7896.54	7974.77	7109.11	17612.95
2022-12-31	8008.50	8027.71	7946.44	8009.14	7139.75	12251.48

In [ ]:

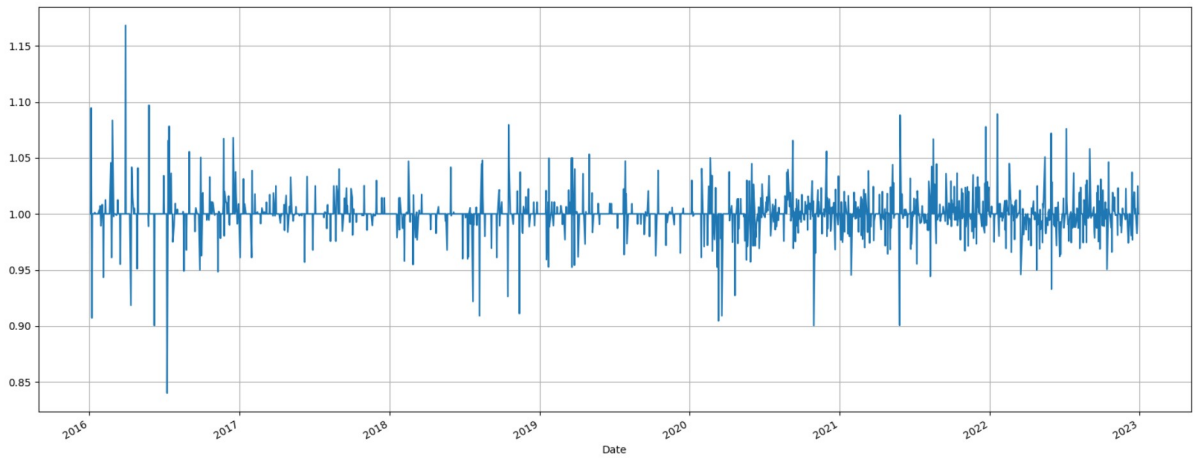
```
# Upsampling (increasing frequency - from days to hourly)
df_ecopetrol['Close'].resample('H').ffill()

# Hay otros métodos para llenar esas nuevas frecuencias como backfill e interpolaci
```

Out[ ]:
```
Date
2016-01-01 00:00:00   1110.00
2016-01-01 01:00:00   1110.00
2016-01-01 02:00:00   1110.00
2016-01-01 03:00:00   1110.00
2016-01-01 04:00:00   1110.00
                        ...
2022-12-29 20:00:00   2420.00
2022-12-29 21:00:00   2420.00
2022-12-29 22:00:00   2420.00
2022-12-29 23:00:00   2420.00
2022-12-30 00:00:00   2420.00
Freq: H, Name: Close, Length: 61321, dtype: float64
```

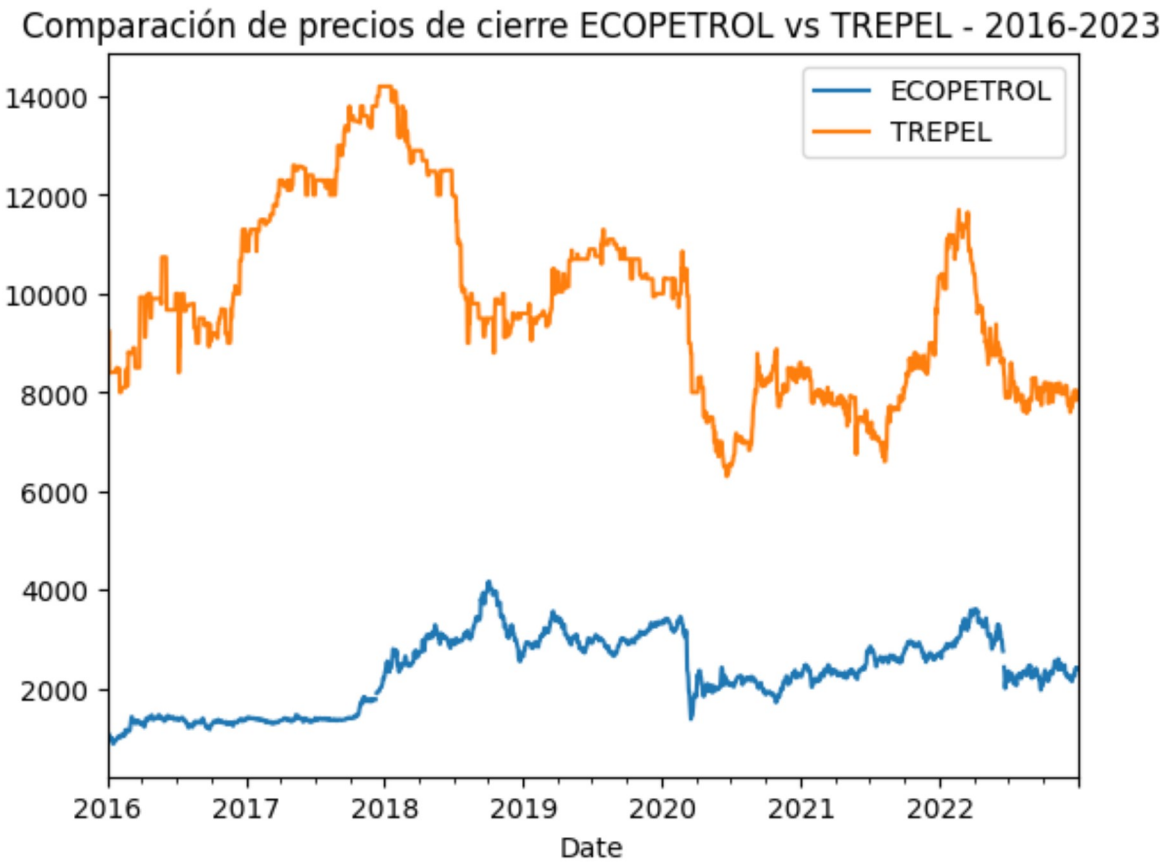
In [ ]:
```
# Visualización de % de cambios y retornos
# Gráfico de cambio porcentual
df_ecopetrol['Change'] = df_ecopetrol['Close'].div(df_ecopetrol['Close'].shift())
df_ecopetrol['Change'].plot(figsize=(20,8)).grid()
```



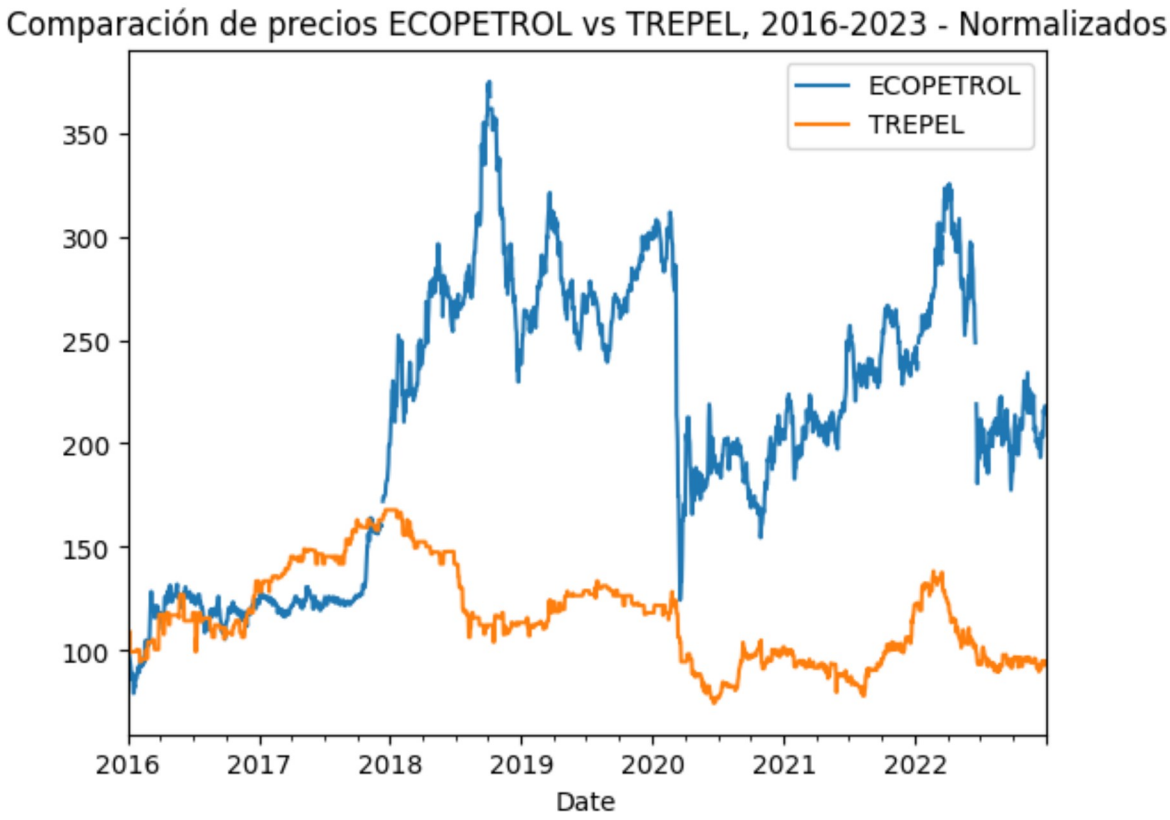
In [ ]:
```
df_trepel['Change'] = df_trepel['Close'].div(df_trepel['Close'].shift())
df_trepel['Change'].plot(figsize=(20,8)).grid()
```



In [ ]:
```
# Técnicas de comparación de series de tiempo
df_ecopetrol['Close'].plot()
df_trepel['Close'].plot()
plt.legend(['ECOPETROL', 'TREPEL'])
plt.title('Comparación de precios de cierre ECOPETROL vs TREPEL - 2016-2023')
plt.show()
```



In [ ]:
```
# Comparación de precios de cierre normalizados (como porcentaje de su valor inicia
normal_ecopetrol = df_ecopetrol['Close'].div(df_ecopetrol['Close'].iloc[0]).mul(100
normal_trepel = df_trepel['Close'].div(df_trepel['Close'].iloc[0]).mul(100)
normal_ecopetrol.plot()
normal_trepel.plot()
plt.legend(['ECOPETROL', 'TREPEL'])
plt.title('Comparación de precios ECOPETROL vs TREPEL, 2016-2023 - Normalizados')
plt.show()
```



In [ ]:
```
# Gráficos de OHLC y candlestick
%matplotlib inline

# Importar librerías y funciones necesarias
from plotly.offline import iplot
import plotly.graph_objects as go

# Gráfico OHLC (Open, High, Low, Close)
ohlc_range = go.Ohlc(x=df_ecopetrol.loc['2022':].index,
                     open=df_ecopetrol.loc['2022':]['Open'],
                     high=df_ecopetrol.loc['2022':]['High'],
                     low=df_ecopetrol.loc['2022':]['Low'],
                     close=df_ecopetrol.loc['2022':]['Close'])

data = [ohlc_range]
iplot(data)
```

In [ ]:
```
ohlc_range = go.Ohlc(x=df_trepel.loc['2022':].index,
                     open=df_trepel.loc['2022':]['Open'],
                     high=df_trepel.loc['2022':]['High'],
                     low=df_trepel.loc['2022':]['Low'],
                     close=df_trepel.loc['2022':]['Close'])

data = [ohlc_range]
iplot(data)
```

In [ ]:
```
# Gráfico de Candlesticks
ohlc_range = go.Candlestick(x=df_ecopetrol.loc['2022':].index,
                     open=df_ecopetrol.loc['2022':]['Open'],
                     high=df_ecopetrol.loc['2022':]['High'],
                     low=df_ecopetrol.loc['2022':]['Low'],
                     close=df_ecopetrol.loc['2022':]['Close'])

data = [ohlc_range]
iplot(data)
```

In [ ]:
```
ohlc_range = go.Candlestick(x=df_trepel.loc['2022':].index,
                     open=df_trepel.loc['2022':]['Open'],
                     high=df_trepel.loc['2022':]['High'],
                     low=df_trepel.loc['2022':]['Low'],
                     close=df_trepel.loc['2022':]['Close'])

data = [ohlc_range]
iplot(data)
```

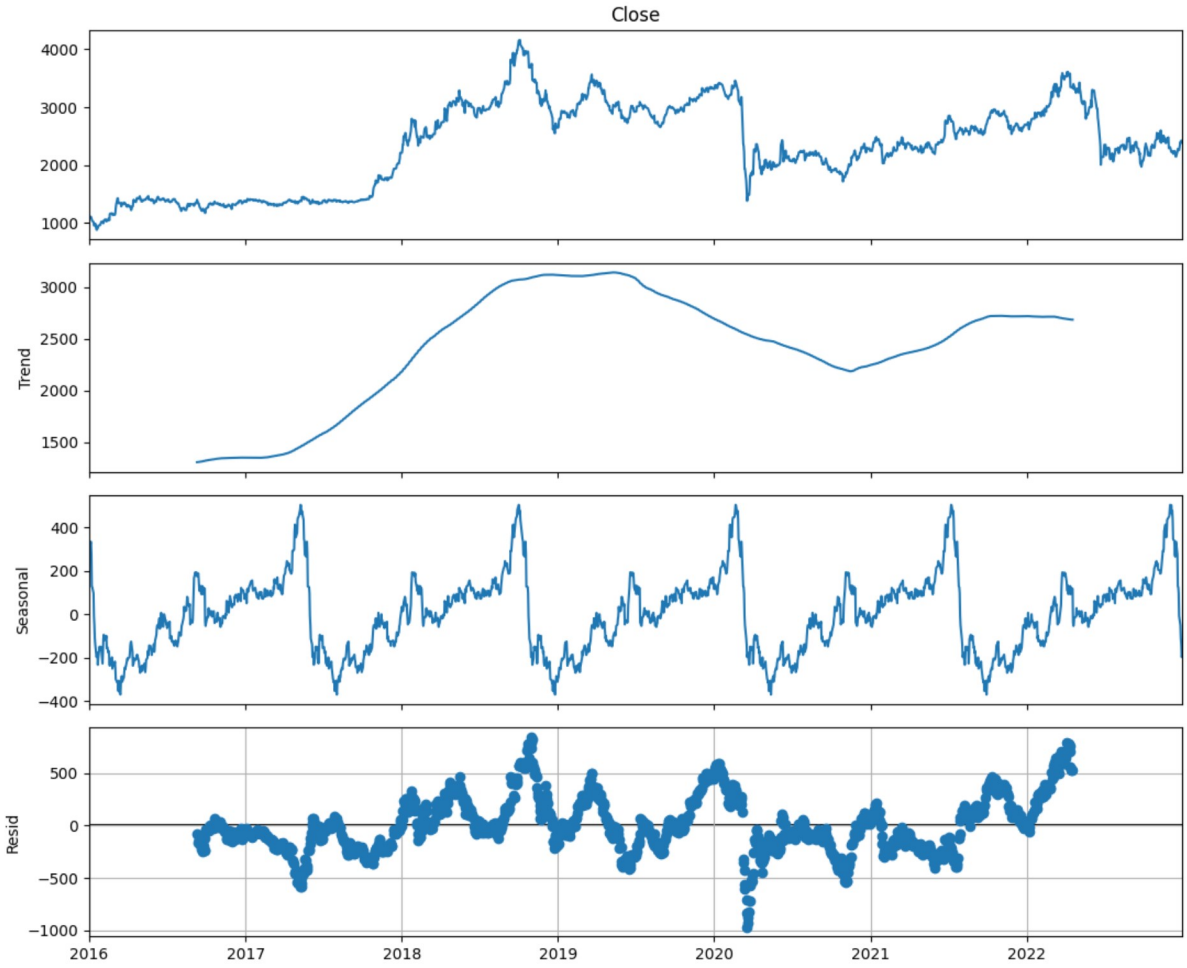
Insights:

- En este caso, los gráficos de OHLC y candlesticks se ven iguales dado que el precio de cierre en los dataframes casi siempre es igual al precio de apertura de la acción en el día

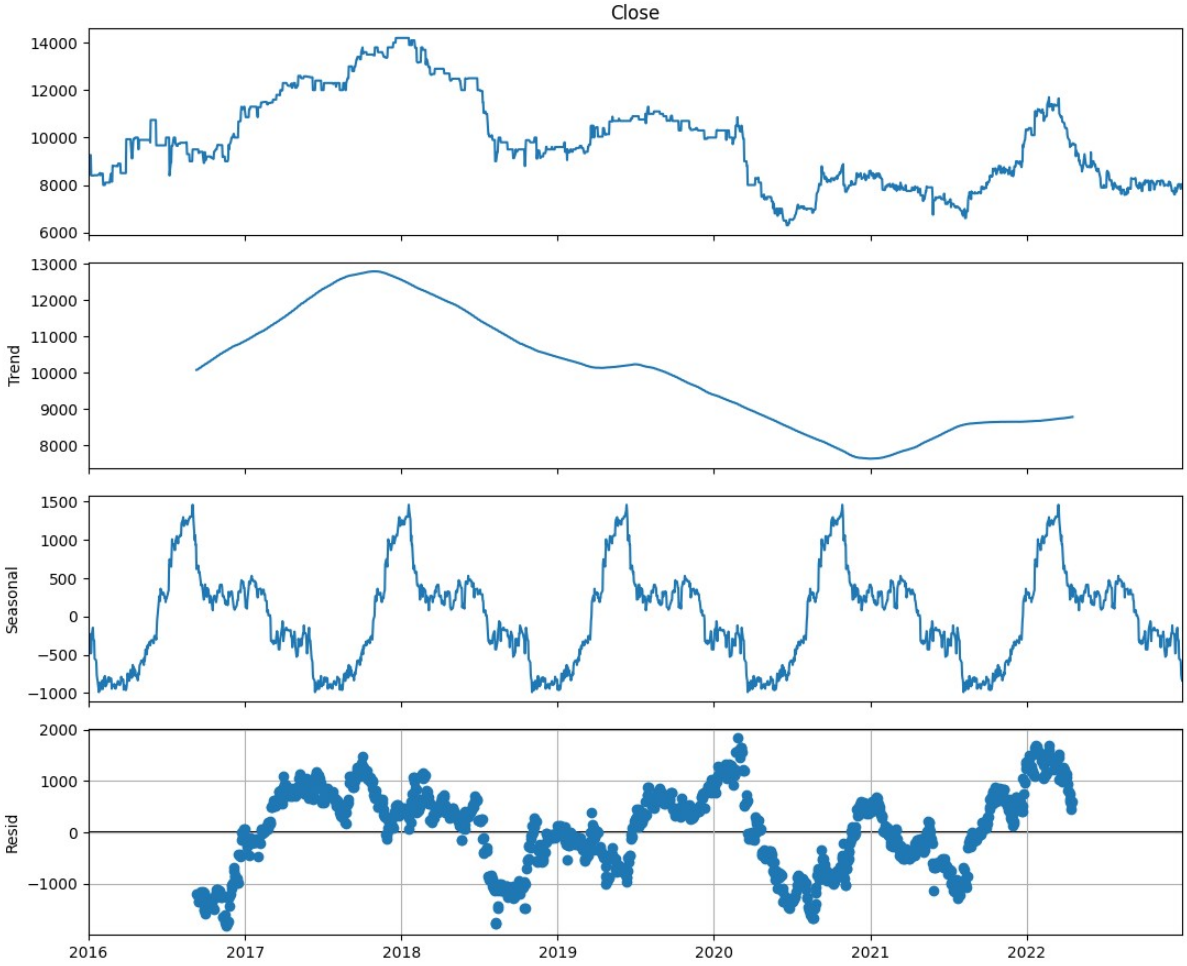
---

Descomposición de Series de Tiempo

In [ ]:
```
# Importación de librerías y funciones
import statsmodels.api as sm
from pylab import rcParams

aux_df = df_ecopetrol[df_ecopetrol['Close'].notnull()]

# Descomposición de la serie de tiempo
rcParams['figure.figsize'] = 11,9
decomposed_ecopetrol_volume = sm.tsa.seasonal_decompose(x=aux_df['Close'], period=3
figure = decomposed_ecopetrol_volume.plot()

plt.grid()
plt.show()
```



In [ ]:
```
# Importación de librerías y funciones
import statsmodels.api as sm
from pylab import rcParams

aux_df = df_trepel[df_trepel['Close'].notnull()]

# Descomposición de la serie de tiempo
rcParams['figure.figsize'] = 11,9
decomposed_trepel_volume = sm.tsa.seasonal_decompose(x=aux_df['Close'], period=360)
figure = decomposed_trepel_volume.plot()

plt.grid()
plt.show()
```


Close
14000
12000
10000
8000
6000
Trend
13000
12000
11000
10000
9000
8000
Seasonal
1500
1000
500
0
−500
−1000
Resid
2000
1000
0
−1000
2016
2017
2018
2019
2020
2021
2022