```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

from sklearn.metrics import mean_squared_error

from statsmodels.graphics.tsaplots import plot_acf
from statsmodels.tsa.seasonal import seasonal_decompose
```

```
df_5y = pd.read_csv(r'/content/drive/MyDrive/PRN23039142546/df_bax_cleaned_to_view_outliers_5y.csv', index_col=0,parse_dates=True)
df_5y.head()
```

	Price	Open	High	Low	Vol.	Change %
Date						
2020-08-05	1292.44	1291.74	1292.44	1289.27	5030000.0	0.15
2020-08-06	1288.83	1292.44	1292.51	1287.92	6130000.0	-0.28
2020-08-09	1292.91	1288.83	1292.91	1288.83	2920000.0	0.32
2020-08-10	1299.94	1293.34	1300.22	1292.97	4880000.0	0.54
2020-08-11	1307.47	1300.66	1307.47	1300.65	5890000.0	0.58

```
# For Naive - get only 5 years data

from datetime import timedelta

df = df_5y.copy()

# Get latest date in the index
latest_date = df.index.max()

# Define how many days of data each model needs
model_durations = {
    'Naive': 60,                  # last ~2 months
    'SES': 180,                   # ~6 months
    'Holt': 365,                  # ~1 year
    'Holt-Winters': 3 * 365,      # ~3 years
    'ARIMA': 2 * 365,             # ~2 years
    'SARIMA': 3 * 365,            # ~3 years
    'Prophet': 5 * 365            # ~5 years (full data)
}

# Create dictionary with sliced data for each model
model_data = {
    model: df[df.index >= (latest_date - timedelta(days=days))].copy()
    for model, days in model_durations.items()
}

# Optional: Save to CSV files
for model, data in model_data.items():
    data.to_csv(f"/content/drive/MyDrive/PRN23039142546/{model}_data.csv")
```

```
df_Naive_60 = pd.read_csv(r'/content/drive/MyDrive/PRN23039142546/Naive_data.csv', index_col=0,parse_dates=True)
df_Naive_60.head()
```

	Price	Open	High	Low	Vol.	Change %
Date						
2025-01-26	1900.20	1896.43	1900.20	1896.43	332820.0	0.20
2025-01-27	1890.11	1902.07	1904.06	1889.40	1250000.0	-0.53
2025-01-28	1891.85	1890.11	1893.78	1890.11	451970.0	0.09
2025-01-29	1873.73	1891.85	1893.54	1873.73	909230.0	-0.96
2025-01-30	1879.48	1873.73	1879.48	1872.98	554100.0	0.31

```
df_Naive_60.info() #check data types
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 33 entries, 2025-01-26 to 2025-03-27
Data columns (total 6 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   Price     33 non-null     float64
 1   Open      33 non-null     float64
 2   High      33 non-null     float64
 3   Low       33 non-null     float64
 4   Vol.      33 non-null     float64
 5   Change %  33 non-null     float64
dtypes: float64(6)
memory usage: 1.8 KB
```

```
df_naive = df_Naive_60.copy()
```

```
price = df_naive['Price']
```

```
split_point = int(len(price)*0.8)
#price[split_point:]
train,test = price[:split_point],price[split_point:]
```

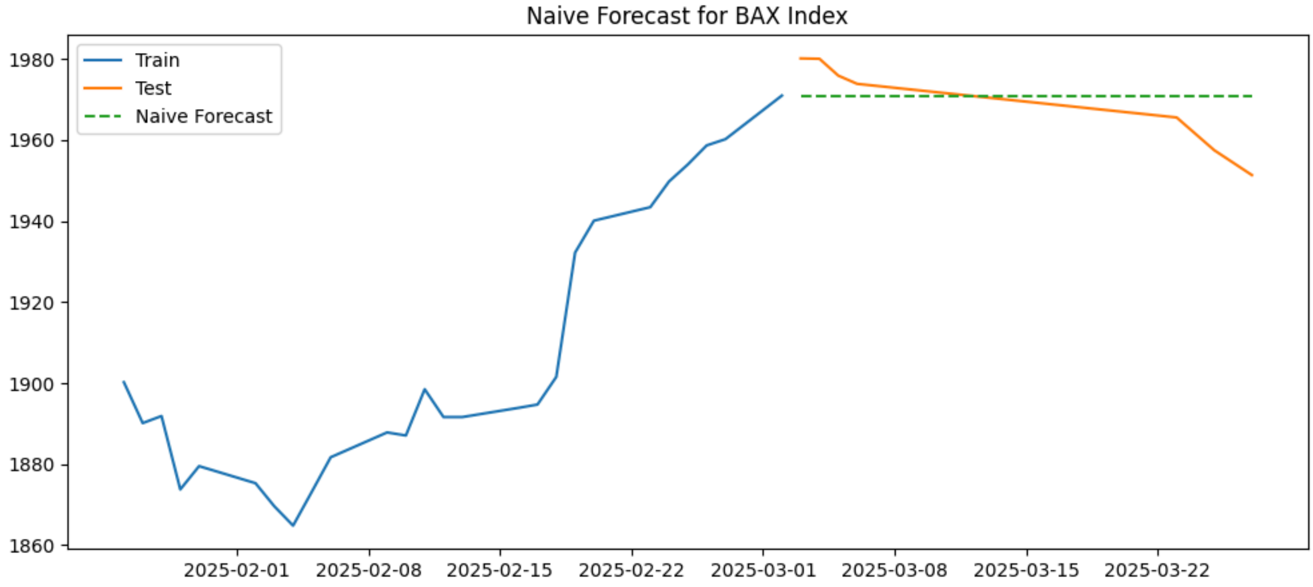
```
naive_forecast = pd.Series(train.iloc[-1], index=test.index)
```

```
rmse = np.sqrt(mean_squared_error(test, naive_forecast))
```

```
print(f"Naive Forecast - RMSE: {rmse:.2f}")
```

```
Naive Forecast - RMSE: 10.66
```

```
plt.figure(figsize=(12,5))
plt.plot(train, label='Train')
plt.plot(test, label='Test')
plt.plot(naive_forecast,label='Naive Forecast', linestyle='--')
plt.legend()
plt.title("Naive Forecast for BAX Index")
plt.show()
```



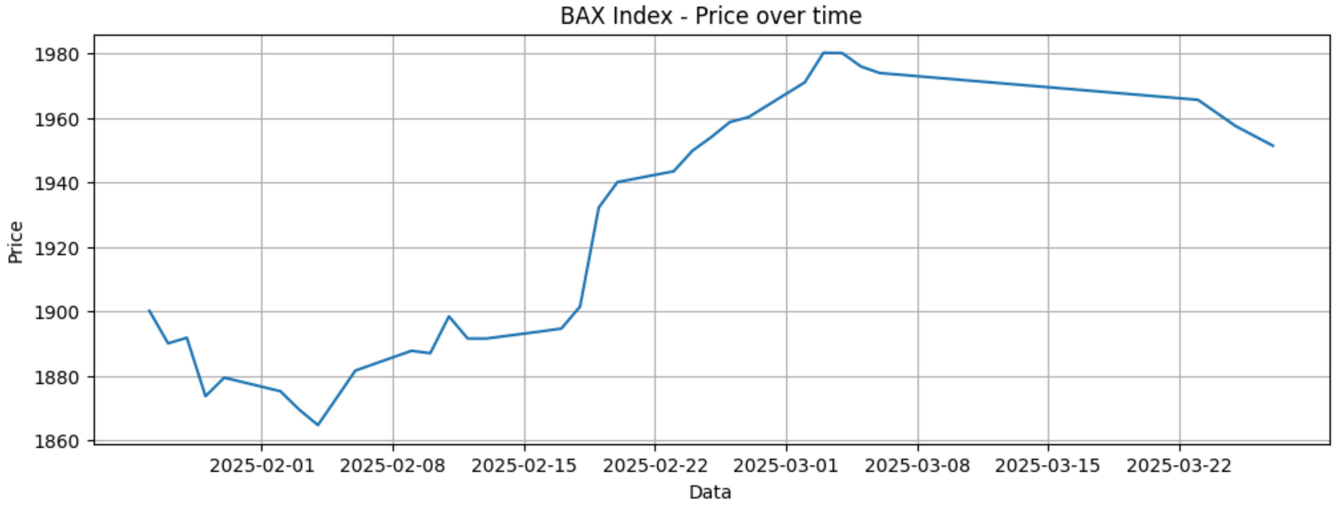
Check seasonality in BAX dataset

```
df = df_Naive_60.copy()
```

```
series = df['Price']
```

```
# line plot

plt.figure(figsize=(12,4))
plt.plot(series)
plt.title("BAX Index - Price over time")
plt.xlabel("Data")
plt.ylabel("Price")
plt.grid(True)
plt.show()
```

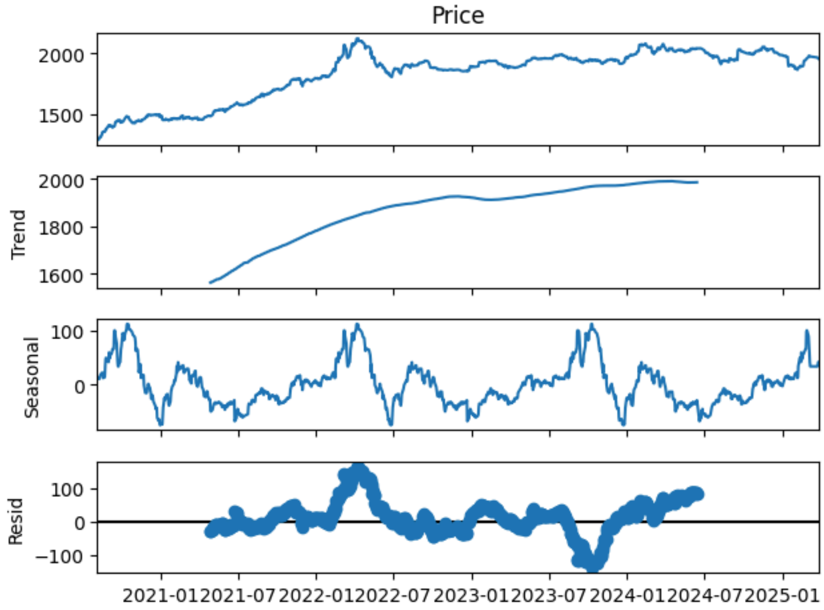


```
df_PROP_5_365 = pd.read_csv(r'/content/drive/MyDrive/PRN23039142546/Prophet_data.csv', index_col=0,parse_dates=True)
df_PROP_5_365.head()
```

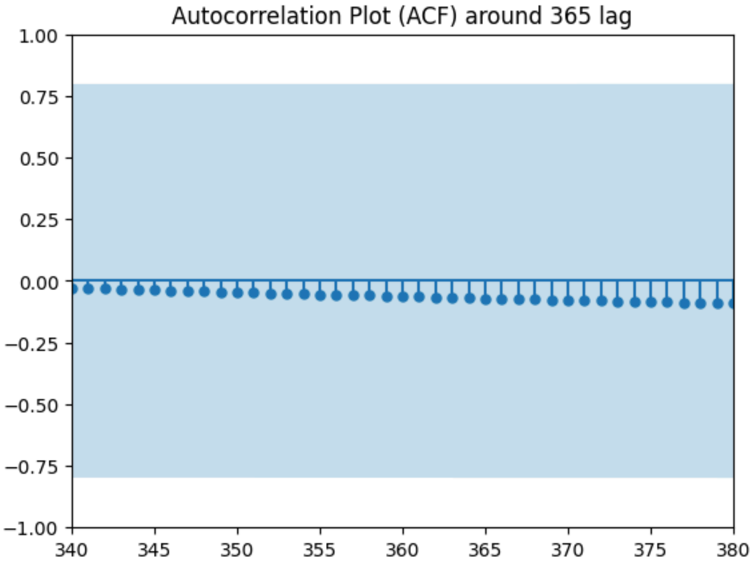
	Price	Open	High	Low	Vol.	Change %
Date						
2020-08-05	1292.44	1291.74	1292.44	1289.27	5030000.0	0.15
2020-08-06	1288.83	1292.44	1292.51	1287.92	6130000.0	-0.28
2020-08-09	1292.91	1288.83	1292.91	1288.83	2920000.0	0.32
2020-08-10	1299.94	1293.34	1300.22	1292.97	4880000.0	0.54
2020-08-11	1307.47	1300.66	1307.47	1300.65	5890000.0	0.58

```
df = df_PROP_5_365.copy()
series = df['Price']
```

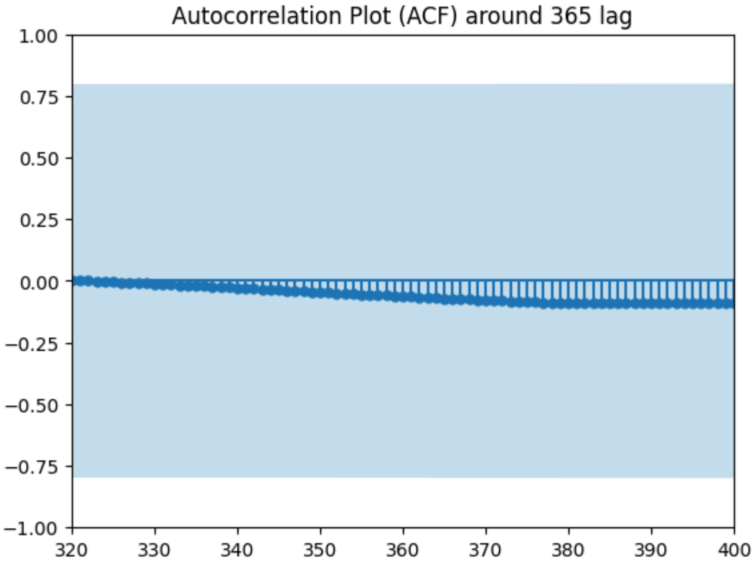
```
# seasonal decomposition (using additive model for now)
decomposition = seasonal_decompose(series, model='additive', period=365) # Assuming daily data with yearly seasonality
decomposition.plot()
plt.tight_layout()
plt.show()
```



```
# autocorrelation plot
plot_acf(series, lags=400)
plt.xlim(340, 380) # Focus on 365 lag
#plt.title("Autocorrelation Plot (ACF)")
plt.title("Autocorrelation Plot (ACF) around 365 lag")
plt.show()
```



```
# autocorrelation plot
plot_acf(series, lags=700)
plt.xlim(320, 400) # Focus on 365 lag
#plt.title("Autocorrelation Plot (ACF)")
plt.title("Autocorrelation Plot (ACF) around 365 lag")
plt.show()
```



```
Start coding or generate with AI.
```