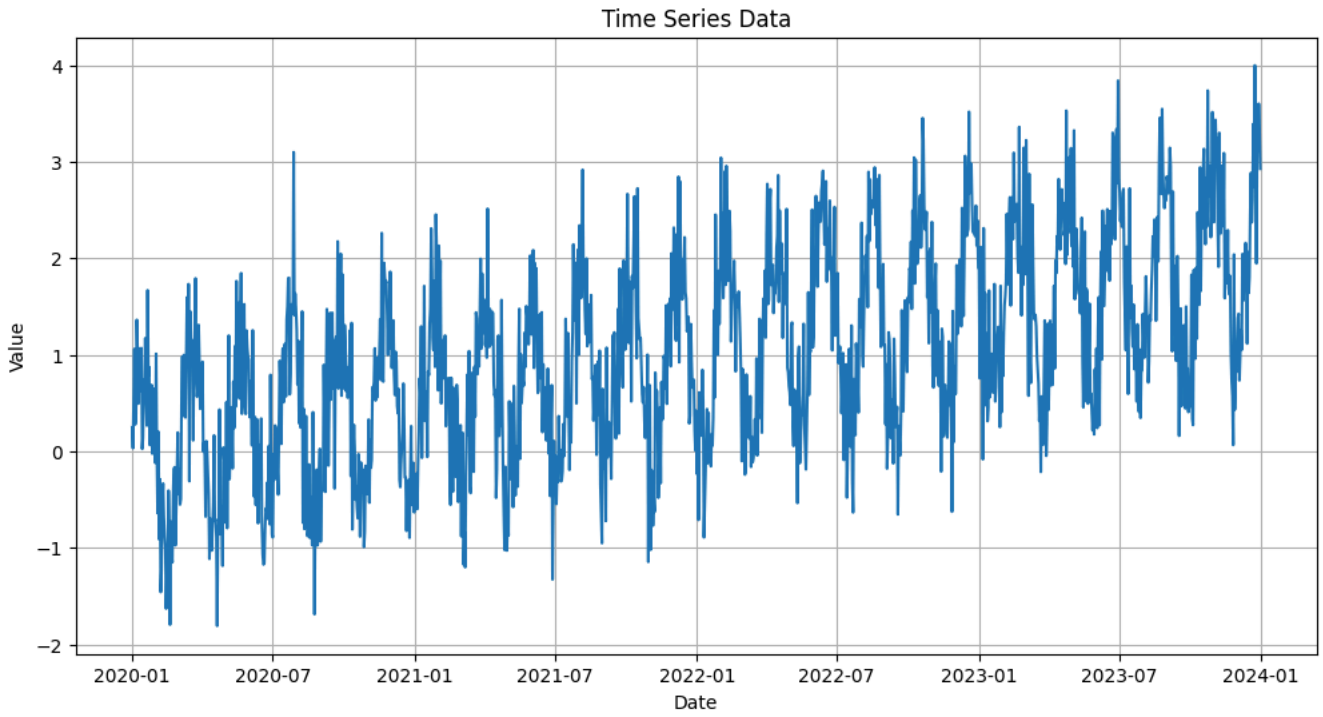
ASSSIGNMENT NO 3 WITH DATASET

```
# Import necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
from statsmodels.tsa.arima.model import ARIMA
from scipy import stats


# Generate a sample time series dataset
np.random.seed(42)
date_rng = pd.date_range(start='2020-01-01', end='2023-12-31', freq='D')
data = pd.DataFrame(date_rng, columns=['date'])
data['value'] = np.sin(data.index / 10) + np.random.normal(0, 0.5, size=(len(date_rng))) + np.linspace(0, 2, len(date_rng))
data.set_index('date', inplace=True)

# Load the dataset into a Pandas DataFrame
print("Sample Data:")
print(data.head())
```

```
Sample Data:
                value
date
2020-01-01  0.248357
2020-01-02  0.032071
2020-01-03  0.525253
2020-01-04  1.061145
2020-01-05  0.277821
```

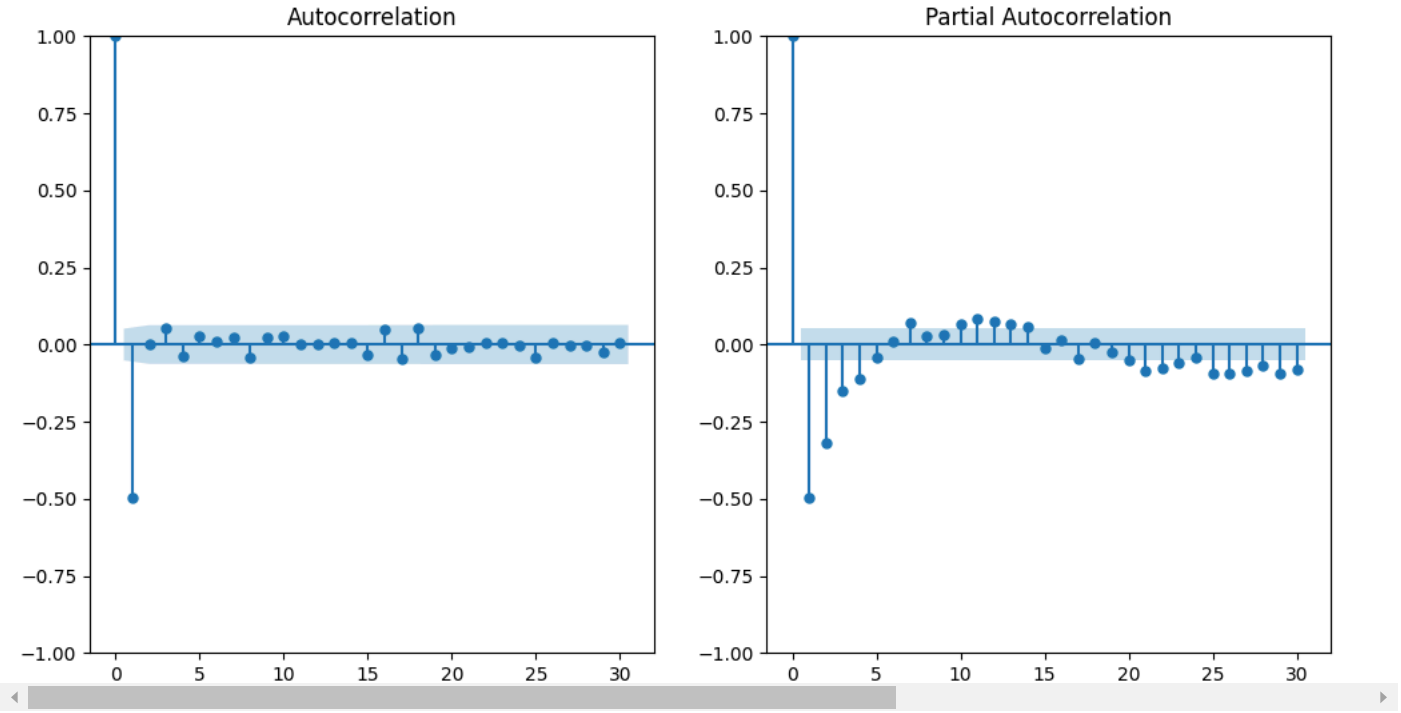
```
# Visualize the data to identify trends, seasonality, and outliers
plt.figure(figsize=(12, 6))
plt.plot(data)
plt.title('Time Series Data')
plt.xlabel('Date')
plt.ylabel('Value')
plt.grid()
plt.show()

# Check for missing values
missing_values = data.isnull().sum()
print("Missing Values:\n", missing_values)

# Handle missing values (if any)
data.fillna(method='ffill', inplace=True)
```




```
Missing Values:
 value    0
dtype: int64
<ipython-input-6-c431c336e73d>:15: FutureWarning: DataFrame.fillna with 'method' is deprecated and will raise in a future version. Use c
```

```
# Check for outliers using Z-score
z_scores = stats.zscore(data)
outliers = (abs(z_scores) > 3).sum()
print(f'Number of outliers: {outliers}')

# Descriptive statistics
mean = data.mean()
std_dev = data.std()
variance = data.var()
print(f'Mean: {mean}, Standard Deviation: {std_dev}, Variance: {variance}')

# Differencing to remove trend (if necessary)
data_diff = data.diff().dropna()

# Analyze ACF and PACF plots
plt.figure(figsize=(12, 6))
plot_acf(data_diff, lags=30, ax=plt.subplot(121))
plot_pacf(data_diff, lags=30, ax=plt.subplot(122))
plt.show()
```

```
Number of outliers: value    0
dtype: int64
Mean: value    1.029117
dtype: float64, Standard Deviation: value    1.041028
dtype: float64, Variance: value    1.083739
dtype: float64
```



Generate a slider using jupyter widgets Close

```
# Select ARIMA model (replace p, d, q with your values based on ACF and PACF)
p = 1  # AR term (from PACF)
d = 1  # Differencing
q = 1  # MA term (from ACF)

model = ARIMA(data, order=(p, d, q))

# Fit the model
model_fit = model.fit()
print(model_fit.summary())

# Evaluate residuals
residuals = model_fit.resid

# Residuals plot
plt.figure(figsize=(12, 6))
plt.subplot(121)
plt.plot(residuals)
plt.title('Residuals')
plt.xlabel('Date')
plt.ylabel('Residual Value')
plt.subplot(122)
plt.hist(residuals, bins=30)
plt.title('Residuals Distribution')
plt.xlabel('Residual Value')
plt.ylabel('Frequency')
plt.show()

# Forecasting
forecast_steps = 10
forecast = model_fit.forecast(steps=forecast_steps)

# Prepare forecast index
forecast_index = pd.date_range(start=data.index[-1] + pd.Timedelta(days=1), periods=forecast_steps, freq='D')

# Create a DataFrame for the forecast
forecast_df = pd.DataFrame(forecast, index=forecast_index, columns=['Forecast'])

# Visualize the forecast
```

```
plt.figure(figsize=(12, 6))
plt.plot(data.index, data['value'], label='Historical Data')
plt.plot(forecast_df.index, forecast_df['Forecast'], label='Forecast', color='red')
plt.title('Forecast vs Historical Data')
plt.xlabel('Date')
plt.ylabel('Value')
plt.legend()
plt.show()
```