1.

MRINMAYI SANIA

22BAU030

IOT ASSIGNMENT 1

Coding:

import matplotlib.pyplot as plt

years=[2016,2017,2018,2019,2020]

sales=[250,300,350,400,450]

plt.plot(years, sales,marker='o')

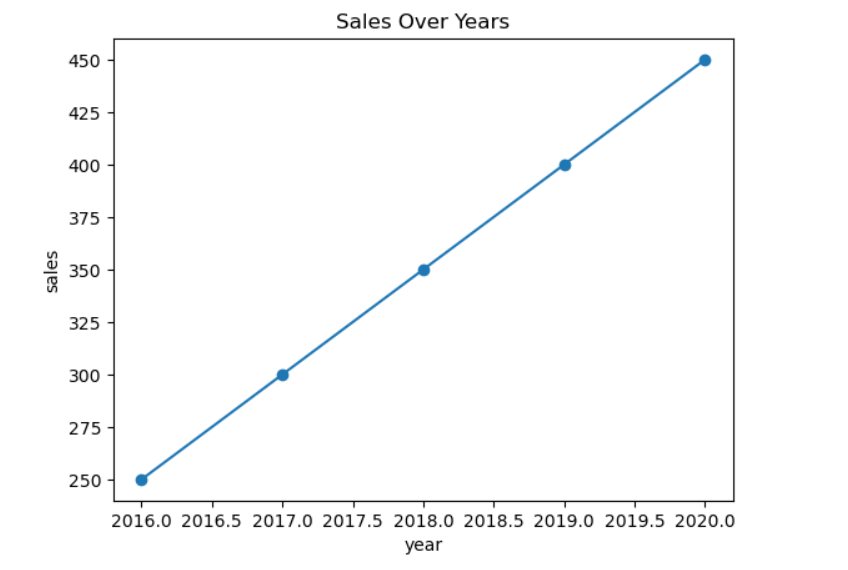
plt.title('Sales Over Years')

plt.xlabel('year')

plt.ylabel('sales')

plt.show()

Output:



2.

Coding:

import matplotlib.pyplot as plt

categories = ['A', 'B', 'C', 'D']

values = [10, 24, 36, 40]

plt.bar(categories, values, color='skyblue')

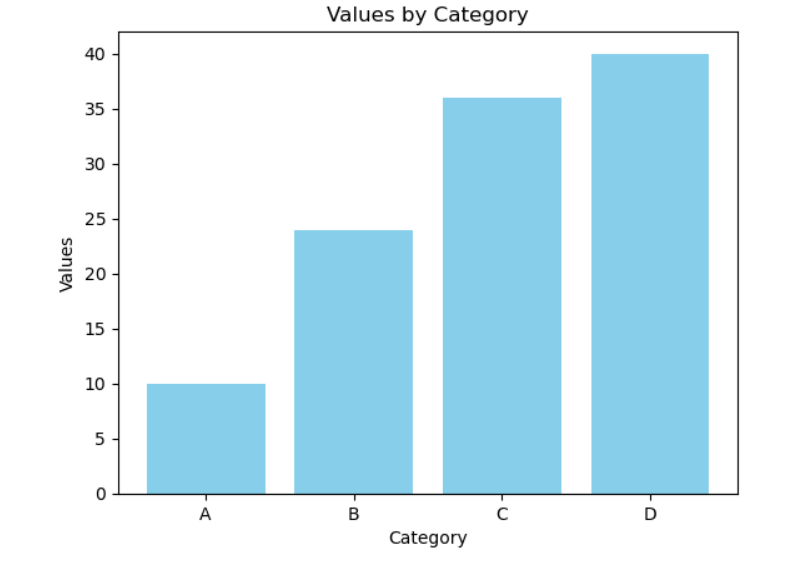
plt.title('Values by Category')

plt.xlabel('Category')

plt.ylabel('Values')

plt.show()

Output:



3.

Coding:

import matplotlib.pyplot as plt

import numpy as np

data = np.random.randn(1000)

plt.hist(data, bins=30, color='purple')

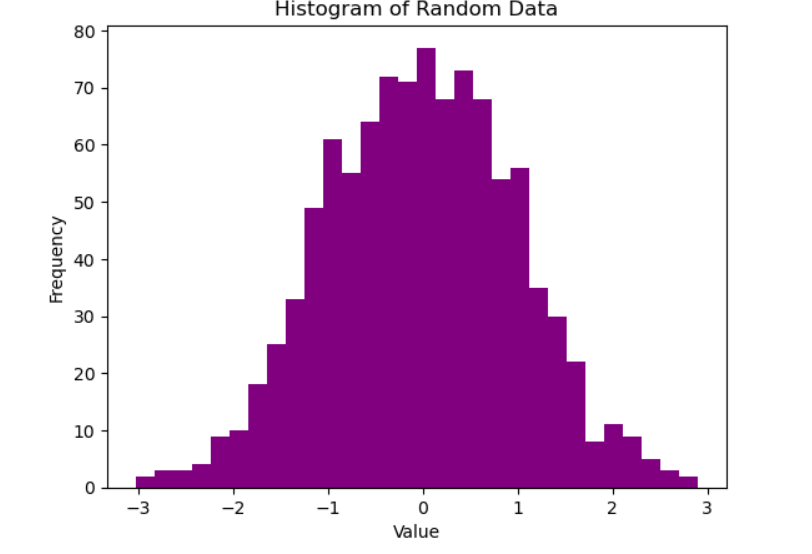
plt.title('Histogram of Random Data')

plt.xlabel('Value')

plt.ylabel('Frequency')

plt.show()

Output:



4.

Coding:

import matplotlib.pyplot as plt

x = [5, 7, 8, 7, 2, 17, 2, 9, 4, 11]

y = [99, 86, 87, 88, 100, 86, 103, 87, 94, 78]

plt.scatter(x, y, color='red')

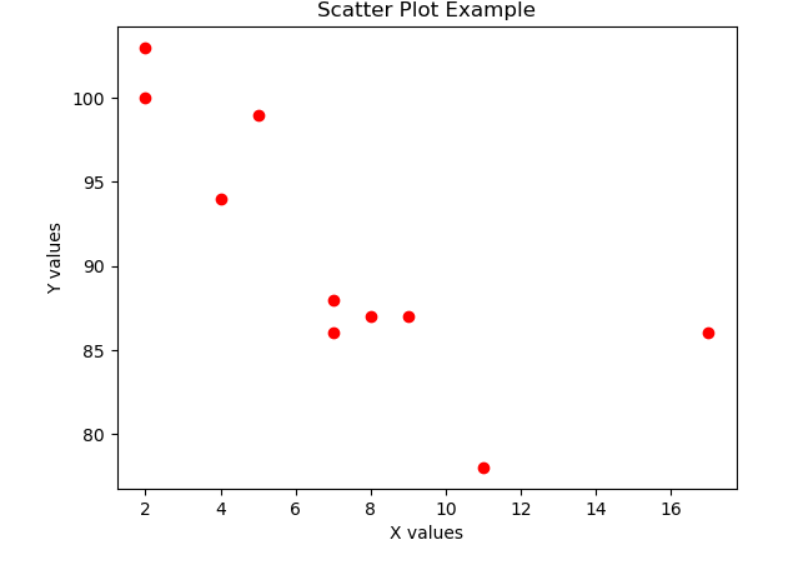
plt.title('Scatter Plot Example')

plt.xlabel('X values')

plt.ylabel('Y values')

plt.show()

Output:



5.

Coding:

import seaborn as sns

import matplotlib.pyplot as plt

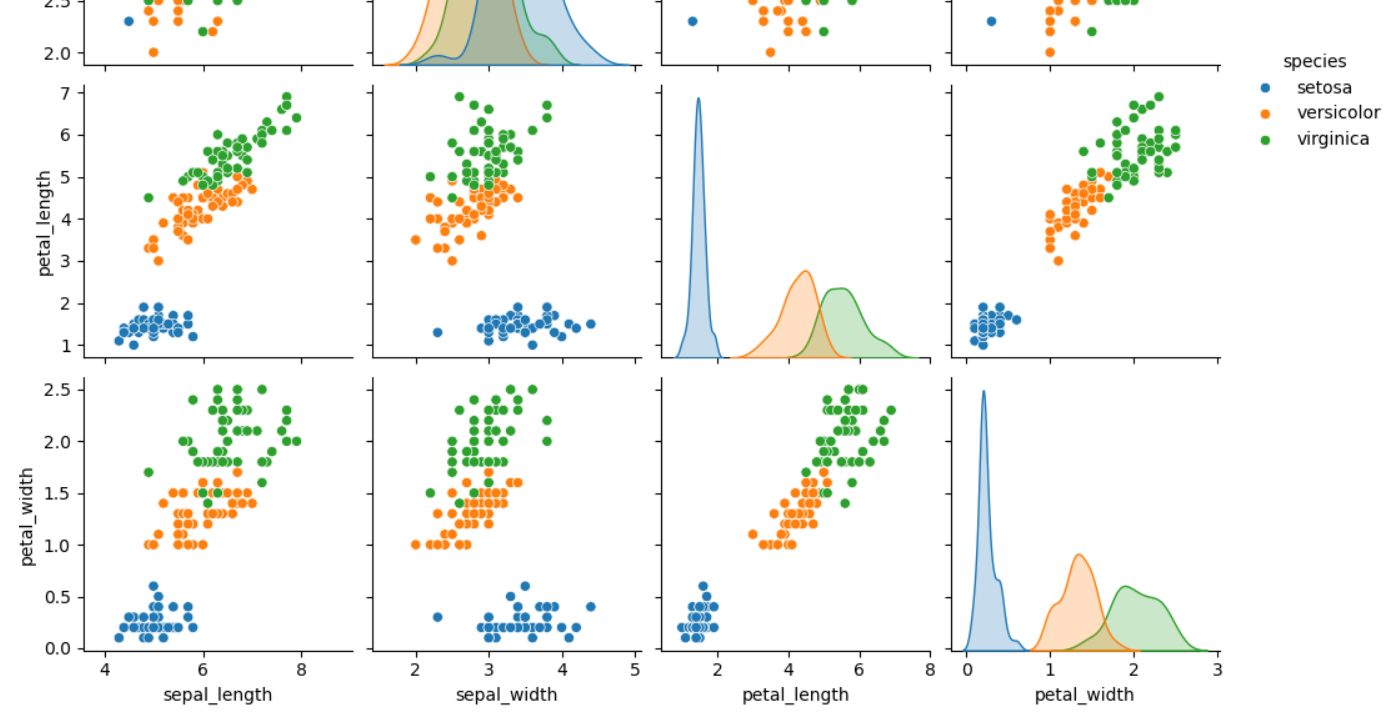
import pandas as pd

data = sns.load\_dataset('iris')

sns.pairplot(data, hue='species')

plt.show()

Output:



6.

Coding:

import seaborn as sns

import matplotlib.pyplot as plt

import numpy as np

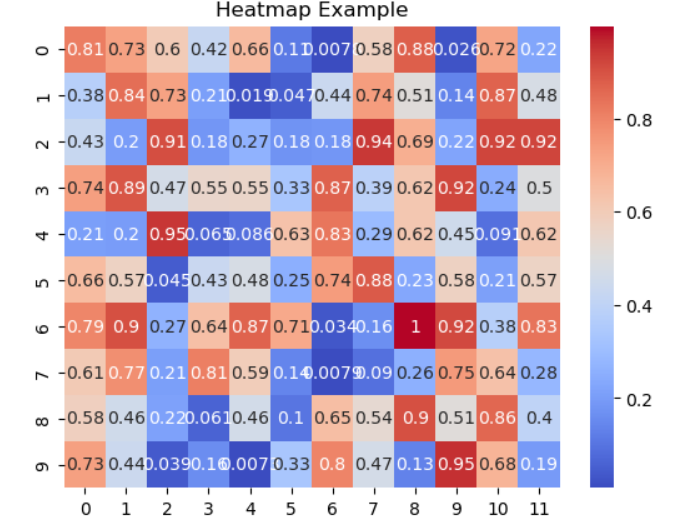
data = np.random.rand(10, 12)

sns.heatmap(data, annot=True, cmap='coolwarm')

plt.title('Heatmap Example')

plt.show()

Output:



1. Time Series

Coding:

dates = pd.date\_range(start='2023-01-01', periods=100, freq='D')

data = pd.DataFrame({

'Date': dates,

'Sales': np.random.randint(100, 500, size=(100,))

})

data.set\_index('Date', inplace=True)

print(data.head())

Output:

Date

2023-01-01 395

2023-01-02 435

2023-01-03 387

2023-01-04 264

2023-01-05 470

8.

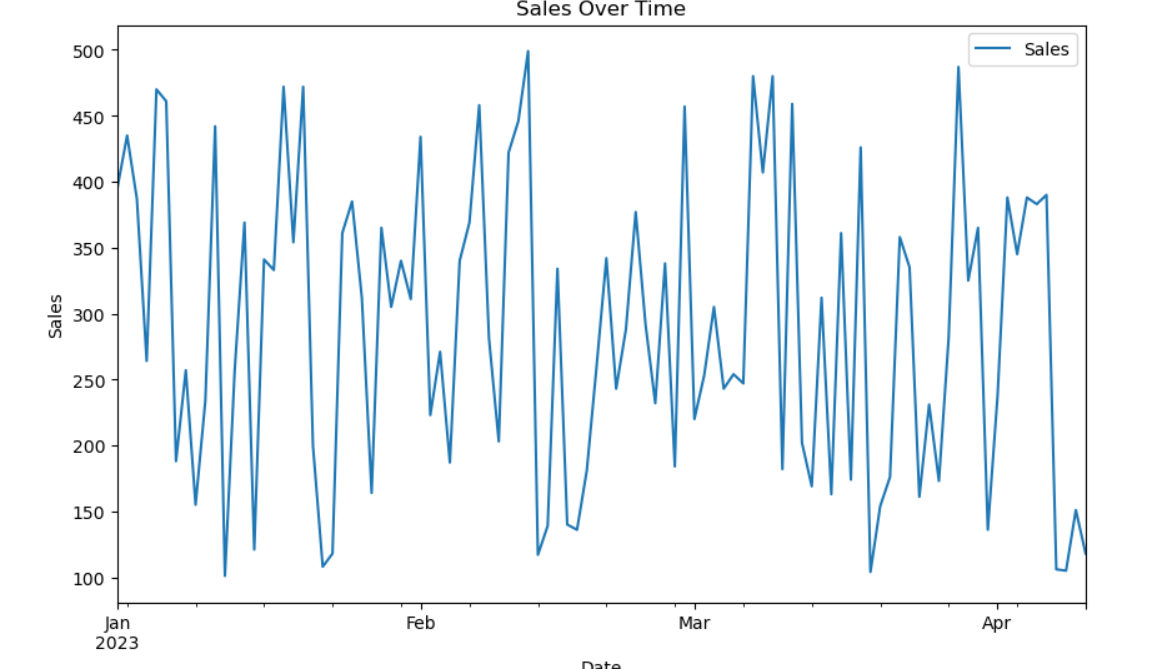
Coding:

data.plot(y='Sales', title='Sales Over Time', figsize=(10, 6))

plt.ylabel('Sales')

plt.show()

Output:



9.

Coding:

monthly\_sales = data.resample('M').sum()

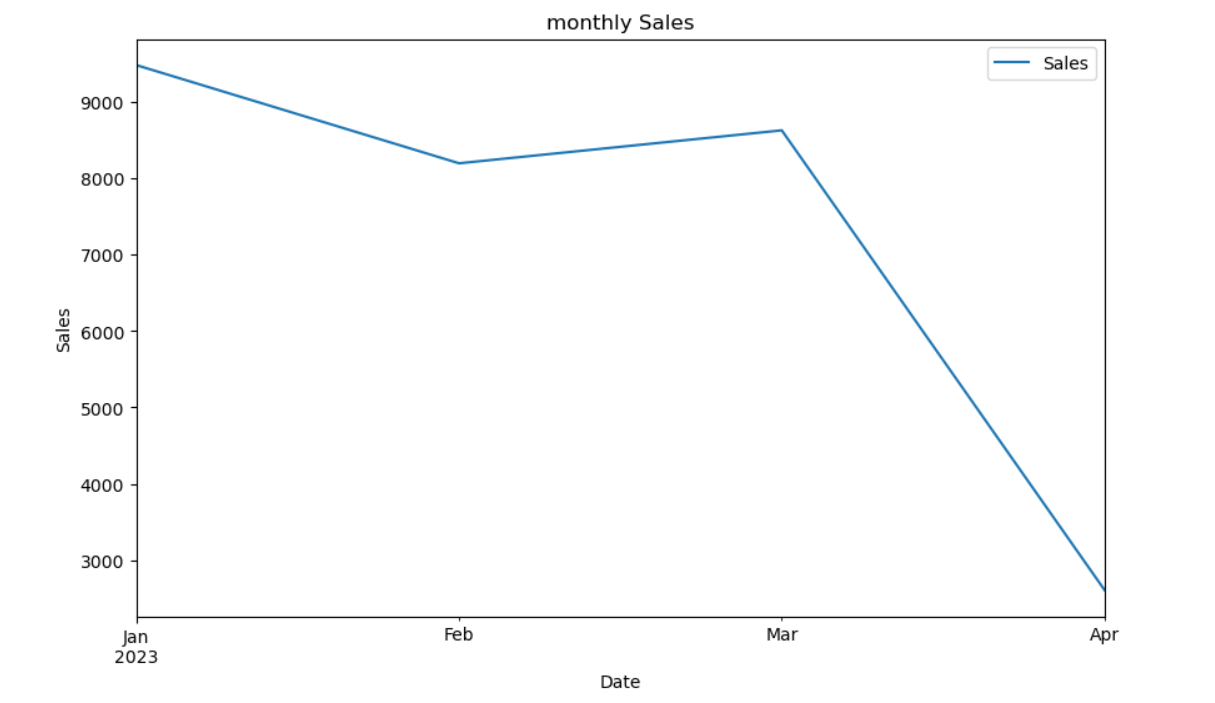
print(monthly\_sales)

monthly\_sales.plot(y='Sales', title='Monthly Sales', figsize=(10, 6))

plt.ylabel('Sales')

plt.show()

Output:



10.

Coding:

data['7-Day Rolling Mean'] = data['Sales'].rolling(window=7).mean()

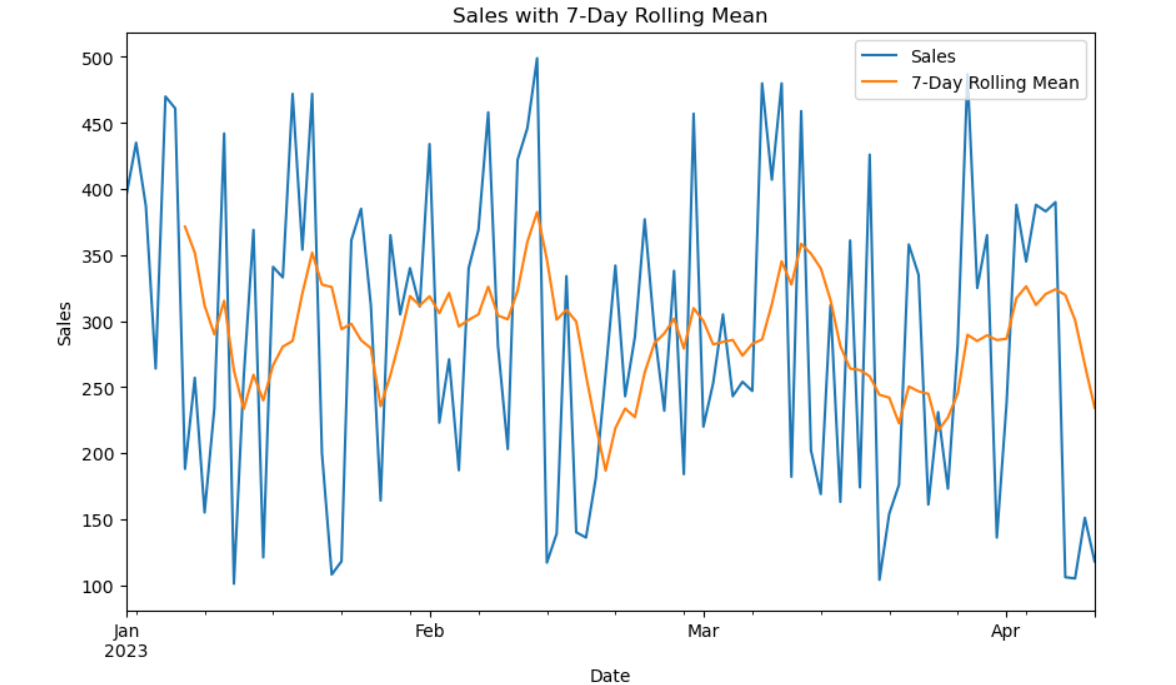
data.plot(y=['Sales', '7-Day Rolling Mean'], title='Sales with 7-Day Rolling Mean',

figsize=(10, 6))

plt.ylabel('Sales')

plt.show()

Output:



11.

Coding:

data\_with\_nans = data.copy()

data\_with\_nans.iloc[20:25] = np.nan

data\_filled = data\_with\_nans.fillna(method='ffill')

print("Original data with NaNs:\n", data\_with\_nans.head(30))

print("\nData after filling NaNs:\n", data\_filled.head(30))

Output;

Original data with NaNs:

Sales 7-Day Rolling Mean

Date

2023-01-01 395.0 NaN

2023-01-02 435.0 NaN

2023-01-03 387.0 NaN

2023-01-04 264.0 NaN

2023-01-05 470.0 NaN

2023-01-06 461.0 NaN

2023-01-07 188.0 371.428571

2023-01-08 257.0 351.714286

2023-01-09 155.0 311.714286

2023-01-10 234.0 289.857143

2023-01-11 442.0 315.285714

2023-01-12 101.0 262.571429

2023-01-13 257.0 233.428571

2023-01-14 369.0 259.285714

2023-01-15 121.0 239.857143

2023-01-16 341.0 266.428571

2023-01-17 333.0 280.571429

2023-01-18 472.0 284.857143

2023-01-19 354.0 321.000000

2023-01-20 472.0 351.714286

2023-01-21 NaN NaN

2023-01-22 NaN NaN

2023-01-23 NaN NaN

2023-01-24 NaN NaN

2023-01-25 NaN NaN

2023-01-26 312.0 279.428571

2023-01-27 164.0 235.428571

2023-01-28 365.0 259.000000

2023-01-29 305.0 287.142857

2023-01-30 340.0 318.857143

Data after filling NaNs:

Sales 7-Day Rolling Mean

Date

2023-01-01 395.0 NaN

2023-01-02 435.0 NaN

2023-01-03 387.0 NaN

2023-01-04 264.0 NaN

2023-01-05 470.0 NaN

2023-01-06 461.0 NaN

2023-01-07 188.0 371.428571

2023-01-08 257.0 351.714286

2023-01-09 155.0 311.714286

2023-01-10 234.0 289.857143

2023-01-11 442.0 315.285714

2023-01-12 101.0 262.571429

2023-01-13 257.0 233.428571

2023-01-14 369.0 259.285714

2023-01-15 121.0 239.857143

2023-01-16 341.0 266.428571

2023-01-17 333.0 280.571429

2023-01-18 472.0 284.857143

2023-01-19 354.0 321.000000

2023-01-20 472.0 351.714286

2023-01-21 472.0 351.714286

2023-01-22 472.0 351.714286

2023-01-23 472.0 351.714286

2023-01-24 472.0 351.714286

2023-01-25 472.0 351.714286

2023-01-26 312.0 279.428571

2023-01-27 164.0 235.428571

2023-01-28 365.0 259.000000

2023-01-29 305.0 287.142857

2023-01-30 340.0 318.857143

12.

Coding:

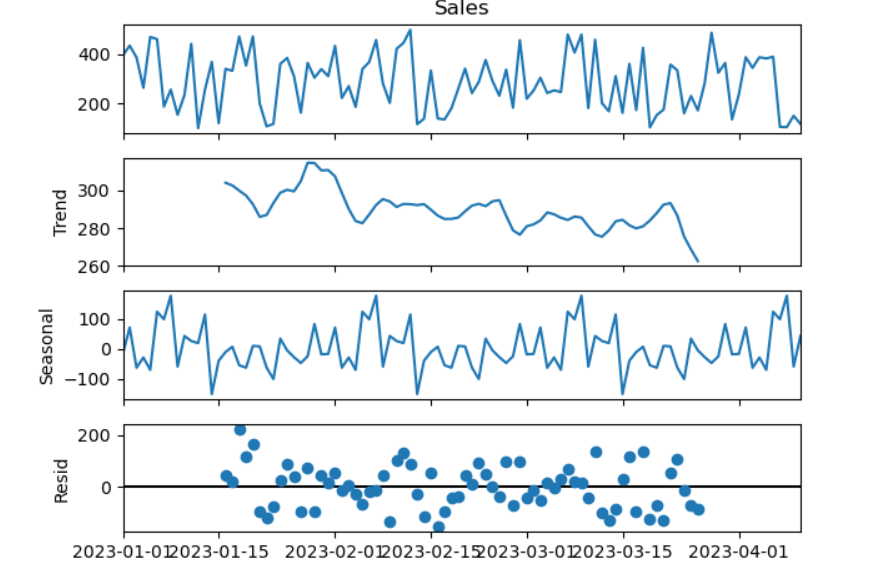
from statsmodels.tsa.seasonal import seasonal\_decompose

decomposition = seasonal\_decompose(data['Sales'], model='additive', period=30)

decomposition.plot()

plt.show()

Output:



1. Handling Missing Data

Coding:

import pandas as pd

import numpy as np

data = {

'timestamp': pd.date\_range(start='2023-01-01', periods=10, freq='D'),

'temperature': [22, 23, np.nan, 25, np.nan, 26, 27, 28, np.nan, 30]

}

df = pd.DataFrame(data)

df\_dropped = df.dropna()

df\_filled = df.fillna(method='ffill')

print("Original Data:\n", df)

print("\nData after Dropping Missing Values:\n", df\_dropped)

print("\nData after Forward-Fill Imputation:\n", df\_filled)

Output:

Original Data:

timestamp temperature

0 2023-01-01 22.0

1 2023-01-02 23.0

2 2023-01-03 NaN

3 2023-01-04 25.0

4 2023-01-05 NaN

5 2023-01-06 26.0

6 2023-01-07 27.0

7 2023-01-08 28.0

8 2023-01-09 NaN

9 2023-01-10 30.0

Data after Dropping Missing Values:

timestamp temperature

0 2023-01-01 22.0

1 2023-01-02 23.0

3 2023-01-04 25.0

5 2023-01-06 26.0

6 2023-01-07 27.0

7 2023-01-08 28.0

9 2023-01-10 30.0

Data after Forward-Fill Imputation:

timestamp temperature

0 2023-01-01 22.0

1 2023-01-02 23.0

2 2023-01-03 23.0

3 2023-01-04 25.0

4 2023-01-05 25.0

5 2023-01-06 26.0

6 2023-01-07 27.0

7 2023-01-08 28.0

8 2023-01-09 28.0

9 2023-01-10 30.0

1. Remove Duplicates

Coding:

data = {

'timestamp': pd.date\_range(start='2023-01-01', periods=5, freq='D').tolist() \* 2,

'temperature': [22, 23, 24, 25, 26] \* 2

}

df = pd.DataFrame(data)

df\_no\_duplicates = df.drop\_duplicates()

print("Original Data:\n", df)

print("\nData after Removing Duplicates:\n", df\_no\_duplicates)

Output:

Original Data:

timestamp temperature

0 2023-01-01 22

1 2023-01-02 23

2 2023-01-03 24

3 2023-01-04 25

4 2023-01-05 26

5 2023-01-01 22

6 2023-01-02 23

7 2023-01-03 24

8 2023-01-04 25

9 2023-01-05 26

Data after Removing Duplicates:

timestamp temperature

0 2023-01-01 22

1 2023-01-02 23

2 2023-01-03 24

3 2023-01-04 25

4 2023-01-05 26

1. Handle Outliers

Coding:

import numpy as np

data = {

'timestamp': pd.date\_range(start='2023-01-01', periods=10, freq='D'),

'temperature': [22, 23, 24, 250, 25, 26, 27, 28, 29, 30]

}

df = pd.DataFrame(data)

df['z\_score'] = (df['temperature'] - df['temperature'].mean()) / df['temperature'].std()

df\_outliers\_removed = df[df['z\_score'].abs() < 3]

print("Original Data with Outlier:\n", df)

print("\nData after Removing Outlier:\n", df\_outliers\_removed)

Output:

Original Data with Outlier:

timestamp temperature z\_score

0 2023-01-01 22 -0.372450

1 2023-01-02 23 -0.358342

2 2023-01-03 24 -0.344234

3 2023-01-04 250 2.844161

4 2023-01-05 25 -0.330126

5 2023-01-06 26 -0.316018

6 2023-01-07 27 -0.301910

7 2023-01-08 28 -0.287802

8 2023-01-09 29 -0.273694

9 2023-01-10 30 -0.259586

Data after Removing Outlier:

timestamp temperature z\_score

0 2023-01-01 22 -0.372450

1 2023-01-02 23 -0.358342

2 2023-01-03 24 -0.344234

3 2023-01-04 250 2.844161

4 2023-01-05 25 -0.330126

5 2023-01-06 26 -0.316018

6 2023-01-07 27 -0.301910

7 2023-01-08 28 -0.287802

8 2023-01-09 29 -0.273694

9 2023-01-10 30 -0.259586

1. Standardize Data

Coding:

from sklearn.preprocessing import MinMaxScaler, StandardScaler

data = {

'timestamp': pd.date\_range(start='2023-01-01', periods=5, freq='D'),

'temperature': [22, 23, 24, 25, 26],

'humidity': [0.55, 0.60, 0.58, 0.62, 0.64]

}

df = pd.DataFrame(data)

scaler = MinMaxScaler()

df\_normalized = df.copy()

df\_normalized[['temperature', 'humidity']] = scaler.fit\_transform(df[['temperature',

'humidity']])

scaler = StandardScaler()

df\_standardized = df.copy()

df\_standardized[['temperature', 'humidity']] = scaler.fit\_transform(df[['temperature',

'humidity']])

print("Original Data:\n", df)

print("\nNormalized Data:\n", df\_normalized)

print("\nStandardized Data:\n", df\_standardized)

Output:

Original Data:

timestamp temperature humidity

0 2023-01-01 22 0.55

1 2023-01-02 23 0.60

2 2023-01-03 24 0.58

3 2023-01-04 25 0.62

4 2023-01-05 26 0.64

Normalized Data:

timestamp temperature humidity

0 2023-01-01 0.00 0.000000

1 2023-01-02 0.25 0.555556

2 2023-01-03 0.50 0.333333

3 2023-01-04 0.75 0.777778

4 2023-01-05 1.00 1.000000

Standardized Data:

timestamp temperature humidity

0 2023-01-01 -1.414214 -1.536443

1 2023-01-02 -0.707107 0.064018

2 2023-01-03 0.000000 -0.576166

3 2023-01-04 0.707107 0.704203

4 2023-01-05 1.414214 1.344387

1. Convert Datatypes

Coding:

data = {

'timestamp': ['2023-01-01', '2023-01-02', '2023-01-03'],

'temperature': ['22.5', '23.1', '21.9'] # Temperature as string

}

df = pd.DataFrame(data)

df['timestamp'] = pd.to\_datetime(df['timestamp'])

df['temperature'] = df['temperature'].astype(float)

print("Data with Corrected Types:\n", df)

Output:

Data with Corrected Types:

timestamp temperature

0 2023-01-01 22.5

1 2023-01-02 23.1

2 2023-01-03 21.9

1. Handle Time zone and Time Formatting

Coding:

data = {

'timestamp': ['2023-01-01 12:00:00', '2023-01-02 12:00:00'],

'temperature': [22.5, 23.1]

}

df = pd.DataFrame(data)

df['timestamp'] = pd.to\_datetime(df['timestamp']).dt.tz\_localize('UTC')

df['timestamp\_est'] = df['timestamp'].dt.tz\_convert('US/Eastern')

print("Data with Timezone Handling:\n", df)

Output:

Data with Timezone Handling:

timestamp temperature timestamp\_est

0 2023-01-01 12:00:00+00:00 22.5 2023-01-01 07:00:00-05:00

1 2023-01-02 12:00:00+00:00 23.1 2023-01-02 07:00:00-05:00