**EX:No.2 221501019**

**31/01/25**

**Program to implement visualising time series**

# import libraries

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

# load in data, specifying the encoding

sales\_data = pd.read\_csv("/content/Sample - Superstore.csv", encoding='latin-1') # or 'cp1252', try different encodings if this doesn't work

# peek at first few rows of data

sales\_data.head()

# Check the column names of the DataFrame

print(sales\_data.columns)

# convert string to datetime64

# Assuming 'Order Date' column exists and should be used as index:

sales\_data.index = pd.to\_datetime(sales\_data["Order Date"]) # Replace 'Date' with actual date column name if different

# create line plot of sales data

# Replace 'Sales' with the actual sales column name if different

plt.plot(sales\_data.index, sales\_data["Sales"])

plt.xlabel("Date")

plt.ylabel("Sales (USD)")

plt.show()

# extract year from 'Order Date' column

sales\_data["year"] = pd.to\_datetime(sales\_data["Order Date"]).dt.year

# box plot grouped by year

sns.boxplot(data=sales\_data, x="year", y="Sales") # Also changed 'sales' to 'Sales' assuming it's the sales column

plt.show()

# extract year and month from 'Order Date' column

sales\_data["year"] = pd.to\_datetime(sales\_data["Order Date"]).dt.year

sales\_data["month"] = pd.to\_datetime(sales\_data["Order Date"]).dt.month # Extract month

# calculate total sales for each month

sales = sales\_data.groupby(["year", "month"])['Sales'].sum() # Specify 'Sales' column for sum

# re-format the data for the heat-map

sales\_month\_year = sales.reset\_index().pivot(index="year", columns="month", values="Sales") # Use 'Sales' for values

# create heatmap

sns.heatmap(sales\_month\_year, cbar\_kws={"label": "Total Sales"})

plt.title("Sales Over Time")

plt.xlabel("Month")

plt.ylabel("Year")

plt.show()

# import lag\_plot function

from pandas.plotting import lag\_plot

# lag scatter plot using 'Sales' column

lag\_plot(sales\_data['Sales']) # Specify the numerical column 'Sales' for the lag plot

plt.show()

# import autocorrelation function

from pandas.plotting import autocorrelation\_plot

# autocorrelation plot using the 'Sales' column

autocorrelation\_plot(sales\_data['Sales']) # Replace 'Sales' with your desired numeric column if needed

plt.show()