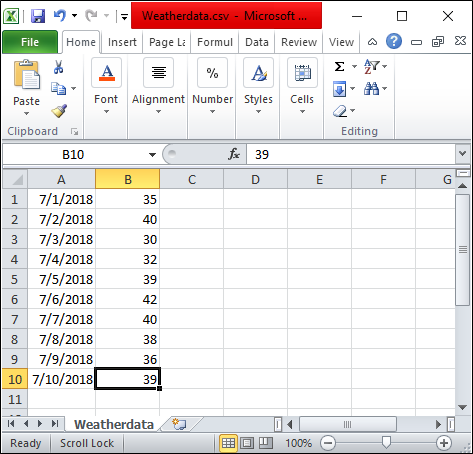
**DATA VISUALIZATION WITH PYTHON**

1. **Write a Python program to read data from a CSV file and create a line plot to visualize the trend over time. Customize the plot with appropriate labels, title, and color.**



Temperature (°C) on different dates is stored in a CSV file as ‘Weatherdata.csv’. These two rows ‘Dates’ and ‘Temperature (°C)’ are used as X and Y-axis for visualizing weather reports.

**Approach of the program:**

1. Import required libraries, matplotlib library for visualizing, and csv library for reading CSV data.
2. Open the file using open( )  function with ‘r’ mode (read-only) from CSV library and read the file using csv.reader( ) function.
3. Read each line in the file using for loop.
4. Append required columns of the CSV file into a list.
5. After reading the whole CSV file, plot the required data as X and Y axis.
6. In this Example, we are plotting Dates as X-axis and Temperature(°C ) as Y-axis.

import matplotlib.pyplot as plt

import csv

x = []

y = []

with open('Weatherdata.csv','r') as csvfile:

    lines = csv.reader(csvfile, delimiter=',')

    for row in lines:

        x.append(row[0])

        y.append(int(row[1]))

plt.plot(x, y, color = 'g', linestyle = 'dashed',

         marker = 'o',label = "Weather Data")

plt.xticks(rotation = 25)

plt.xlabel('Dates')

plt.ylabel('Temperature(°C)')

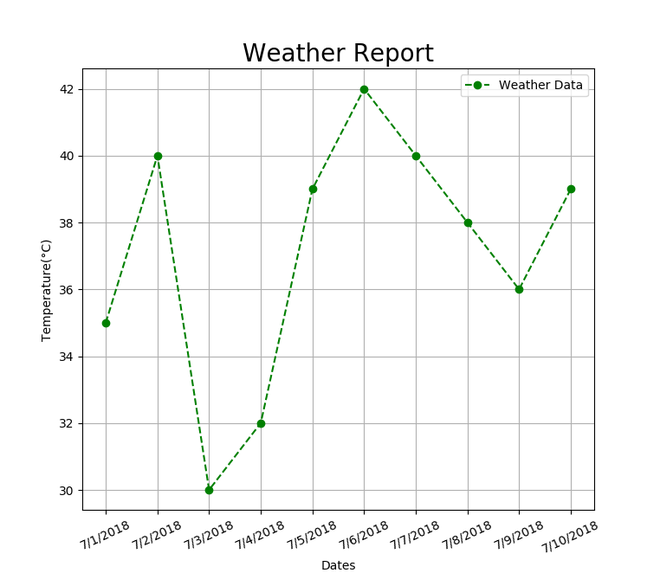
plt.title('Weather Report', fontsize = 20)

plt.grid()

plt.legend()

plt.show()

**Output:**

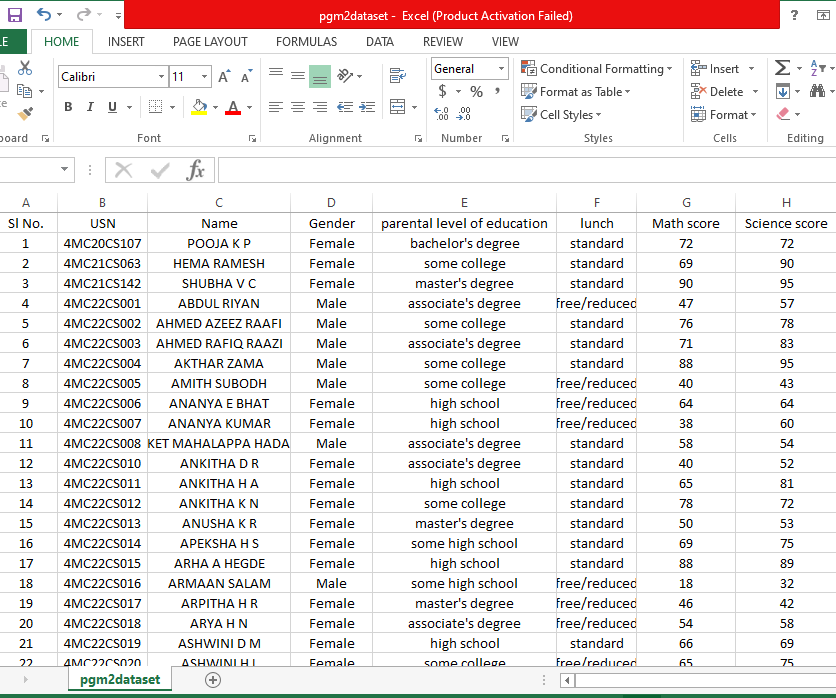


1. **Load a dataset containing information about students' scores in different subjects. Create a scatter plot to visualize the relationship between two variables (e.g., math score vs. science score). Use Seaborn to enhance the plot with appropriate styling and add labels.**

Visualizing Relationship between variables with scatter plots in Seaborn

To understand how variables in a dataset are related to one another and how that relationship is dependent on other variables, we perform statistical analysis. This Statistical analysis helps to visualize the trends and identify various patterns in the dataset. One of the functions which can be used to get the relationship between two variables in Seaborn is relplot().

Relplot() combines FacetGrid with either of the two axes-level functions [scatterplot()](https://www.geeksforgeeks.org/scatterplot-using-seaborn-in-python/) and [lineplot()](https://www.geeksforgeeks.org/lineplot-using-seaborn-in-python/" \t "_blank). Scatterplot is default kind of relplot(). Using this we can visualize joint distribution of two variables through a cloud of points. We can draw scatterplot in seaborn using various ways. The most common one is when both the variables are numeric.

****

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

# set grid style

sns.set(style ="darkgrid")

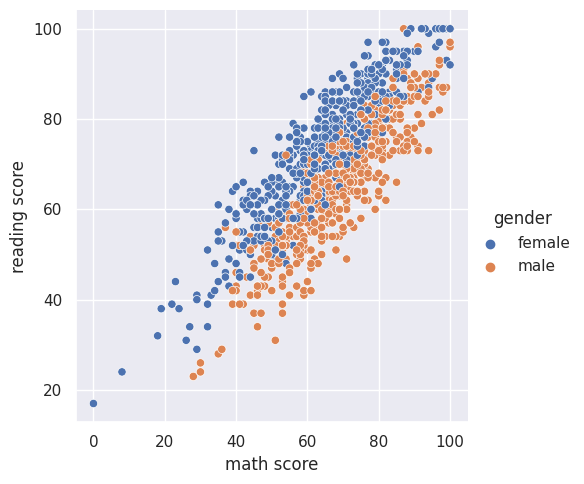
# import dataset

dataset = pd.read\_csv('pgm2dataset.csv')

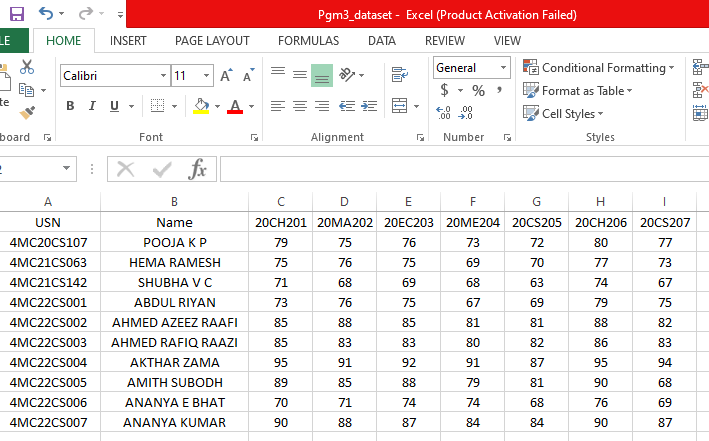
dataset.info()

sns.relplot(x ="math score", y ="reading score", hue ="gender", data = dataset);

Output:

****

1. **Given a dataset with multiple variables, create a figure with two subplots: one displaying a line plot and the other showing a bar chart. Customize the subplots with appropriate titles, legends, and colors.**

****

import pandas as pd

import seaborn as sns

from matplotlib import pyplot as plt

data = pd.read\_csv('Pgm3\_dataset.csv')

data.head()

data\_num = data[['USN', '20CH201', '20MA202', '20EC203', '20ME204', '20CS205', '20CH206', '20CS207']].set\_index('USN')

Student1 = data\_num.loc['4MC22CS001']

Student2 = data\_num.loc['4MC22CS001']

fig, axes = plt.subplots(1, 2, figsize=(15, 5))

fig.suptitle('Student Information', size=24, color = 'blue')

# 4MC22CS001

sns.barplot(ax=axes[0], x=Student1.index, y=Student1.values)

axes[0].set(xlabel='Courses', ylabel='Marks')

axes[0].set\_title('4MC22CS001', size=16, color = 'Red')

# 4MC22CS001

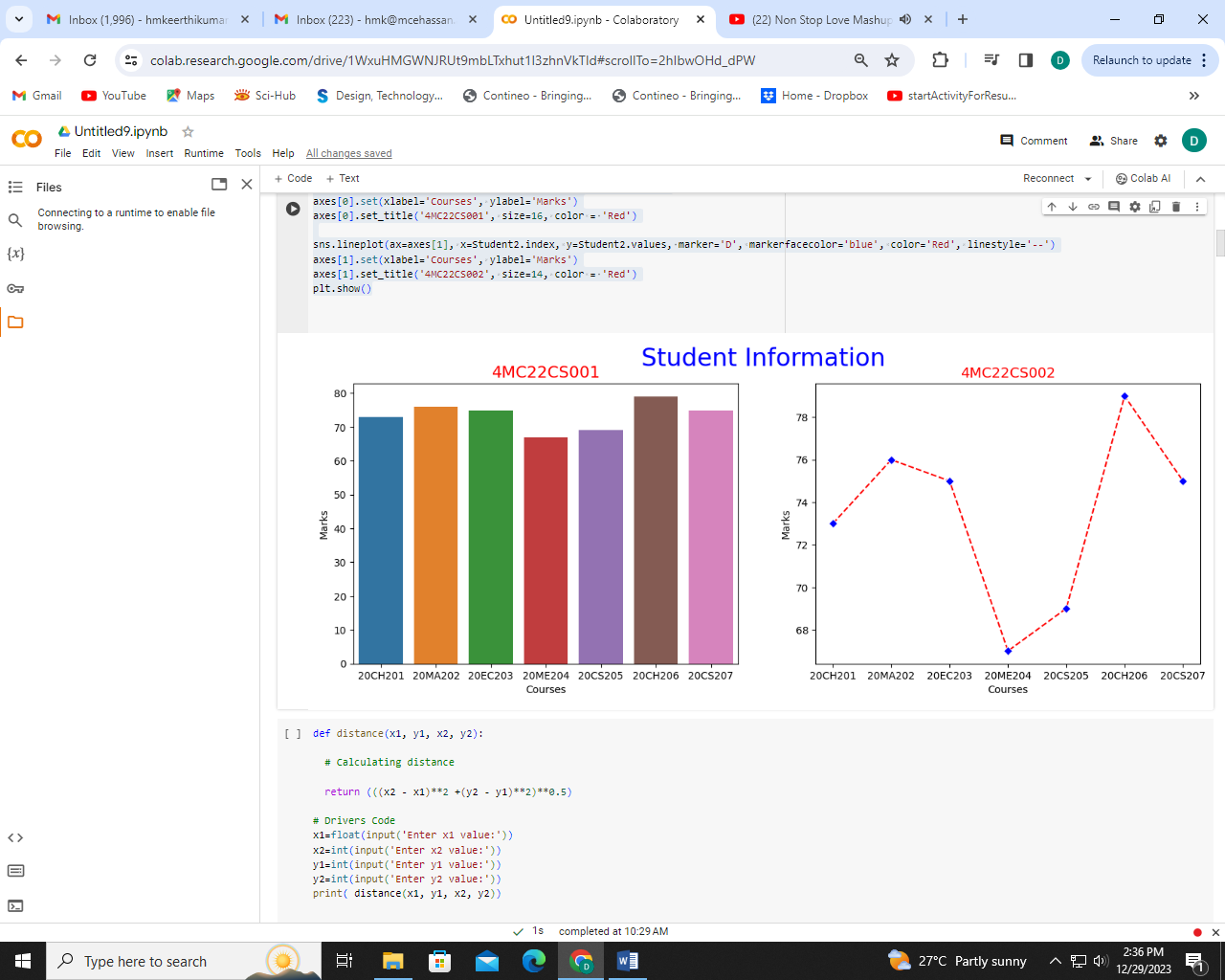
sns.lineplot(ax=axes[1], x=Student2.index, y=Student2.values, marker='D', markerfacecolor='blue', color='Red', linestyle='--')

axes[1].set(xlabel='Courses', ylabel='Marks')

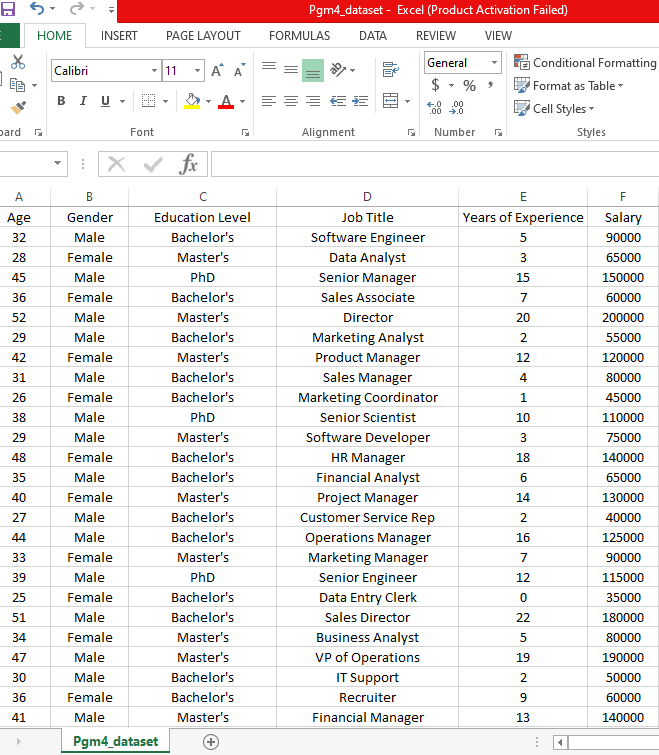
axes[1].set\_title('4MC22CS002', size=14, color = 'Red')

plt.show()

Output:



1. **Load a dataset containing information about employees' salaries across different departments. Create a box plot and a violin plot to visualize the distribution of salaries by department. Customize the plots and add appropriate labels and titles.**



import matplotlib.pyplot as plt

import csv

import seaborn as sns

import pandas as pd

df=pd.read\_csv('Pgm4\_dataset.csv')

df.info()

# Set the plot size

plt.figure(figsize=(20,8))

sns.set\_style("whitegrid")

sns.violinplot(data=df, x='Education Level', y='Salary')

# Set the plot title

plt.title('Salary Details', fontsize=16)

# Set the x-axis label

plt.xlabel('Education Level', fontsize=12)

# Set the y-axis label

plt.ylabel('Salary', fontsize=12)

# Display the plot

plt.show()

plt.title('Salary Details', fontsize=16)

sns.boxplot(x = 'Education Level', y = 'Salary', data = df)

plt.xlabel('Education Level', fontsize=12)

# Set the y-axis label

plt.ylabel('Salary', fontsize=12)

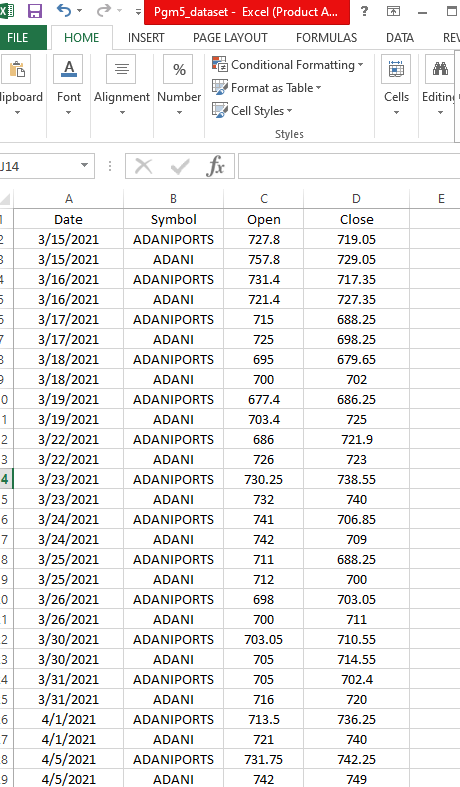
plt.show()

**Output:**





1. **Load a dataset containing stock prices over time. Create a line plot to visualize the stock prices and add appropriate labels and titles. Format the x-axis tick labels to display the dates properly.**

****

# importing packages

import seaborn as sns

import pandas as pd

import csv

import matplotlib.pyplot as plt

# loading dataset

data = pd.read\_csv("Pgm5\_dataset.csv")

sns.lineplot(x="Date", y="Close",

             hue="Symbol", linestyle = 'dashed',style="Symbol", marker = 'o', data=data)

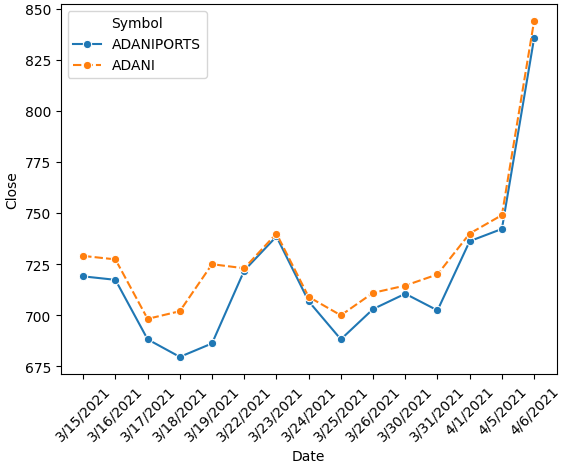
plt.xlabel("Time Line")

plt.ylabel("Closing Price")

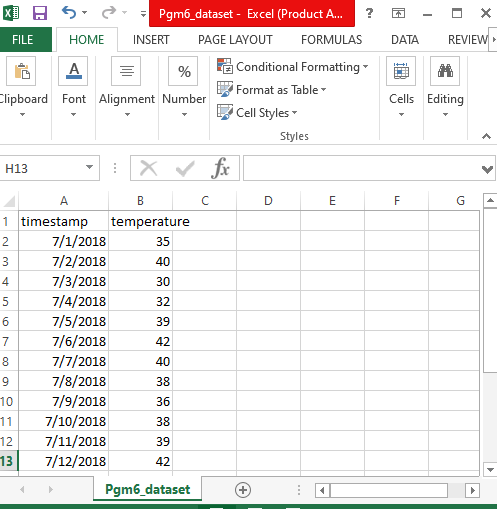
plt.xticks(rotation = 90, ha ='center', fontsize=20, fontfamily= "serif", fontweight='bold')

plt.show()

**Output:**

****

1. **Load a dataset containing temperature readings over time. Create an interactive line plot using Plotly, which displays the temperature when hovering over the data points. Add appropriate labels and customize the plot's appearance.**

****

import pandas as pd

import plotly.express as px

# Load the dataset

df = pd.read\_csv('Pgm6\_dataset.csv')

# Create an interactive line plot

fig = px.line(df, x='timestamp', y='temperature', title='Temperature Over Time',

              labels={'timestamp': 'Timestamp', 'temperature': 'Temperature'},

              hover\_data={'temperature': ':.2f'})

# Customize the appearance of the plot

fig.update\_traces(mode='lines+markers', line=dict(color='blue', width=2),

                  marker=dict(color='red', size=8, line=dict(color='black', width=2)))

fig.update\_layout(title\_text='Temperature Over Time',

                  xaxis\_title='Timestamp',

                  yaxis\_title='Temperature (°C)',

                  hovermode='y unified')

fig.update\_xaxes(tickangle = 45)

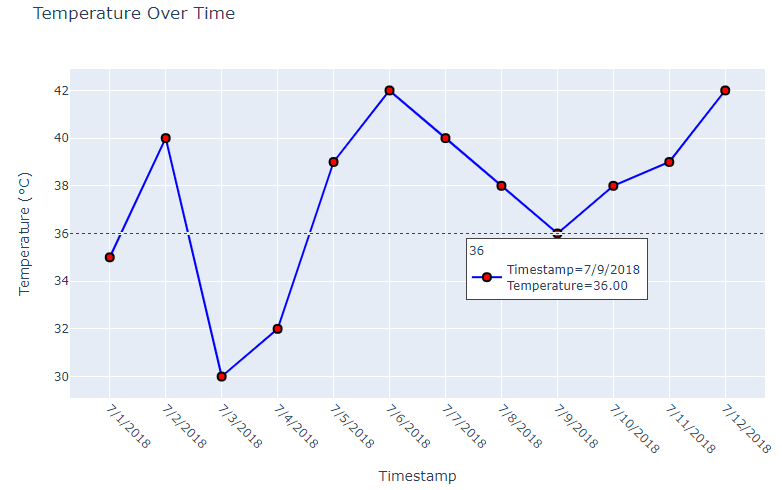
# Show the plot

fig.show()

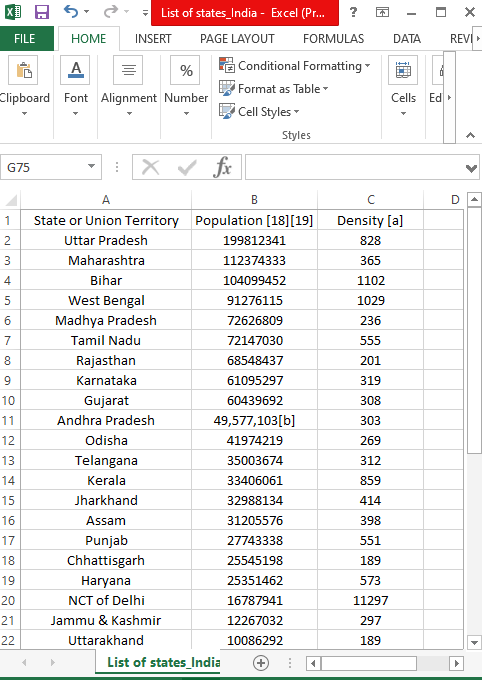
import plotly.io as pio

pio.renderers.default = "browser"

Output:

****

1. **Load a dataset with information about population density by country. Create a choropleth map using GeoPandas to visualize the population density. Customize the map's appearance and add a color legend.**

****

**Sample:**

import geopandas as gpd

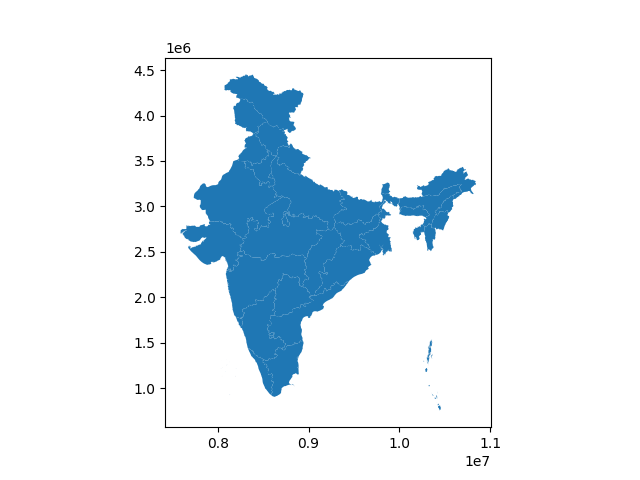
import matplotlib.pyplot as plt

import mapclassify

data = gpd.read\_file('India\_State\_Boundary.shp')

data.plot()

plt.show()



**Main program**

import plotly.express as px

import numpy as np

import pandas as pd

import json

import plotly.io as pio

pio.renderers.default = "browser"

india\_states = json.load(open('states\_india.geojson','r'))

df = pd.read\_csv('List of states\_India.csv')

state\_id\_map ={}

for feature in india\_states['features']:

feature['id'] = feature['properties']['state\_code']

state\_id\_map[feature['properties']['st\_nm']] =feature['id']

df['id']=df['State or Union Territory'].apply(lambda x: state\_id\_map[x])

print(df.head())

print(state\_id\_map)

df['DensityScale'] = np.log10(df['Density [a]'])

fig = px.choropleth(df,

locations = 'id',

geojson=india\_states,

color= 'DensityScale',

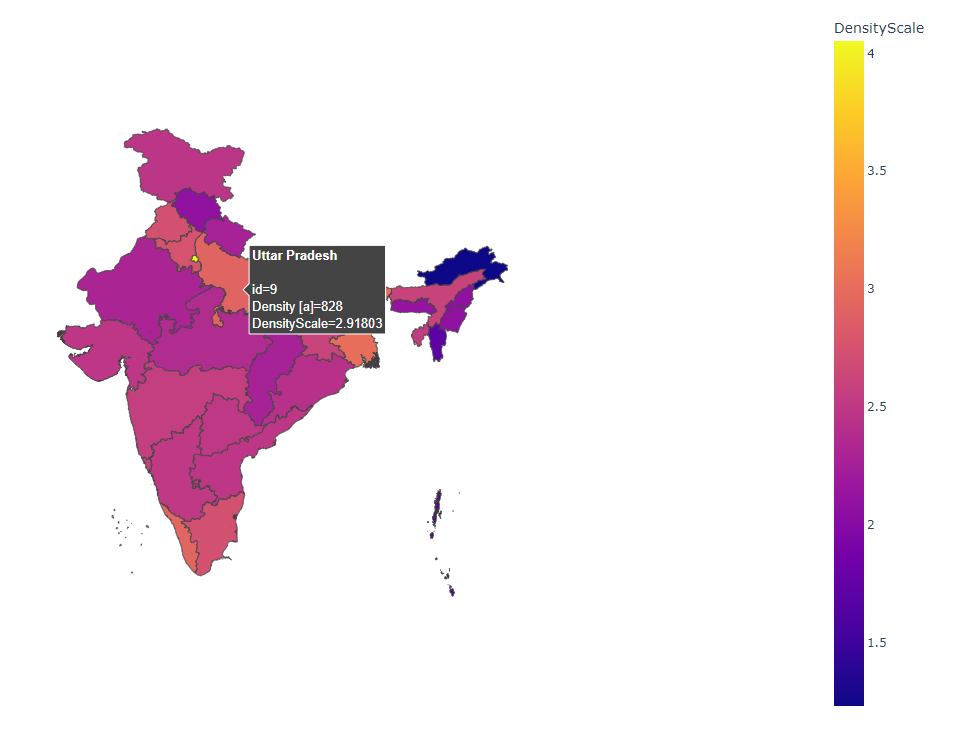
hover\_name='State or Union Territory',

hover\_data=['Density [a]'])

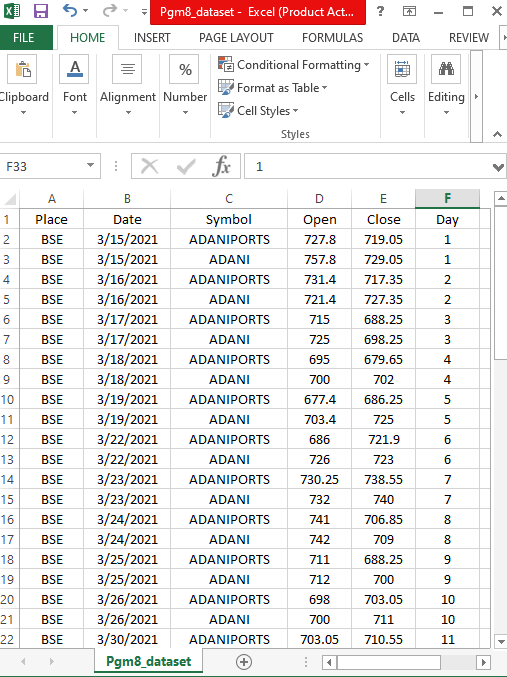
fig.update\_geos(fitbounds= 'locations', visible=False)

fig.show()

Output:



1. **Design and implement an interactive dashboard using Dash to display various visualizations. Include at least two interactive controls (e.g., dropdowns, sliders) to update the visualizations dynamically.**



import dash

from dash import dcc, html

from dash.dependencies import Input, Output

import pandas as pd

import plotly.express as px

# Load a sample dataset

df = pd.read\_csv('Stock\_details\_8.csv')  # Replace 'your\_data.csv' with the actual file name

# Create the Dash app

app = dash.Dash(\_\_name\_\_)

# Define the layout of the app

app.layout = html.Div([

    html.H1("Interactive Dashboard"),

    # Dropdown for selecting a categorical variable

    html.Label("Select a categorical variable:"),

    dcc.Dropdown(

        id='category-dropdown',

        options= df['Place'].unique(),

        value="BSE",  # Default selected column

        ),

    # Slider for selecting a range of numeric values

    html.Label("Select a range of numeric values:"),

    dcc.Slider(

        id='slider',

        min=df['Day'].min(),

        max=df['Day'].max(),

        step=1,

        value=df['Day'].max()  # Default selected range

    ),

    # Scatter plot for visualization

    dcc.Graph(id='line-plot'),

    # Bar chart for visualization

    dcc.Graph(id='bar-chart'),

])

# Define callback to update scatter plot and bar chart based on dropdown and slider values

@app.callback(

    [Output('line-plot', 'figure'),

     Output('bar-chart', 'figure')],

    [Input('category-dropdown', 'value'),

     Input('slider', 'value')]

)

def update\_visualizations(selected\_category, selected\_range):

    filtered\_df = df[df['Place'] == selected\_category]

    filtered\_df\_bar = df[df['Day']== selected\_range]

    # Scatter plot

    line\_fig = px.line(filtered\_df, x='Date', y='Open', color='Symbol',

                       title=f"line Plot - {selected\_category} ")

    # Bar chart

    bar\_fig = px.bar(filtered\_df\_bar, x='Date', y='Open', color='Symbol', title=f"Bar Chart - {selected\_range}")

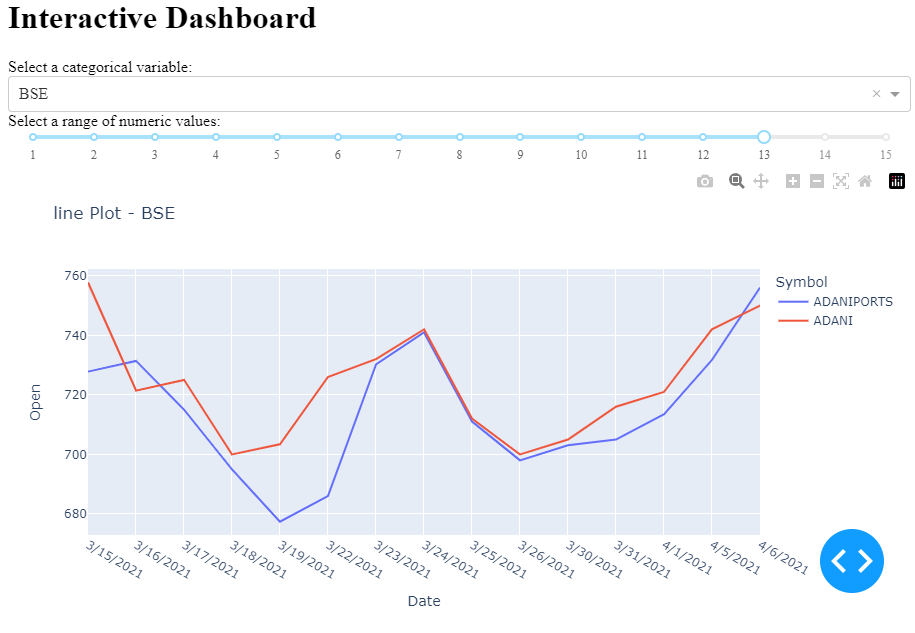
    return line\_fig, bar\_fig

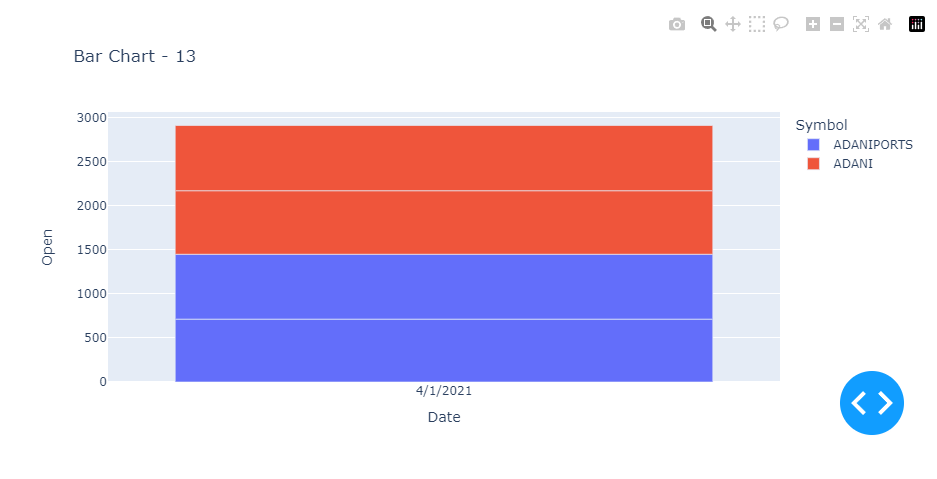
# Run the app

if \_\_name\_\_ == '\_\_main\_\_':

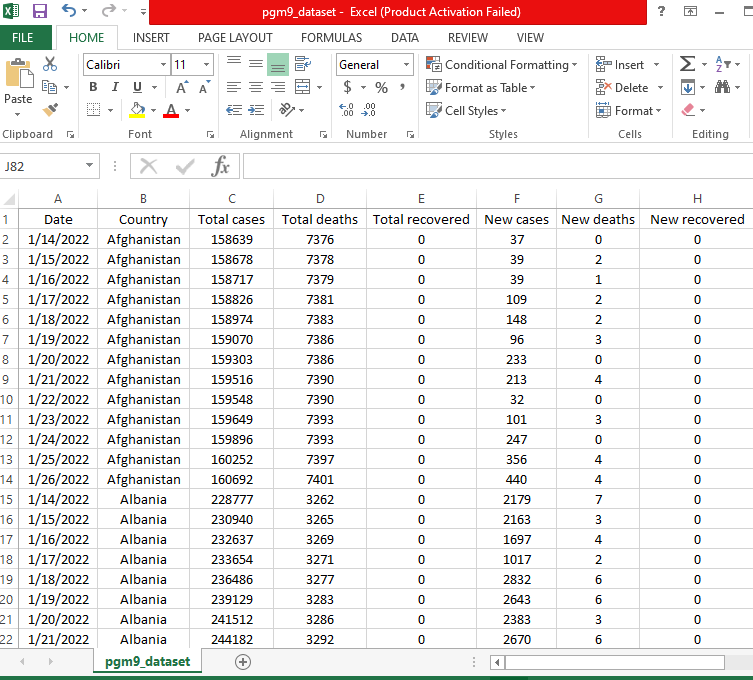
    app.run\_server(debug=True)

**Output:**





**9. Select a dataset related to a specific topic of interest (e.g., climate change, COVID-19). Design a series of visualizations that tell a compelling data story, highlighting key insights and trends. Present the visualizations with appropriate annotations and captions.**



import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

# Load COVID-19 dataset (replace 'your\_dataset.csv' with the actual dataset)

covid\_data = pd.read\_csv('pgm9\_dataset.csv')

plt.figure(figsize=(12, 6))

sns.lineplot(x='Date', y='Total cases', hue = 'Country', marker = 'o', data=covid\_data)

plt.title('Global COVID-19 Cases Over Time')

plt.xlabel('Date')

plt.ylabel('Total Cases')

plt.show()

top\_countries = covid\_data.groupby('Country')['Total cases'].max().sort\_values(ascending=False).head(5)

plt.figure(figsize=(12, 6))

sns.barplot(x=top\_countries.values, y=top\_countries.index, palette='viridis')

plt.title('Top 10 Countries with Highest COVID-19 Cases')

plt.xlabel('Total Cases')

plt.ylabel('Country')

plt.show()

covid\_data['MortalityRate'] = (covid\_data['Total deaths'] / covid\_data['Total cases']) \* 100

plt.figure(figsize=(12, 6))

sns.lineplot(x='Date', y='MortalityRate', hue = 'Country', data=covid\_data)

plt.title('COVID-19 Mortality Rate Over Time')

plt.xlabel('Date')

plt.ylabel('Mortality Rate (%)')

plt.show()

covid\_data['RecoveryRate'] = (covid\_data['Total recovered'] / covid\_data['Total cases']) \* 100

plt.figure(figsize=(12, 6))

sns.lineplot(x='Date', y='RecoveryRate', data=covid\_data)

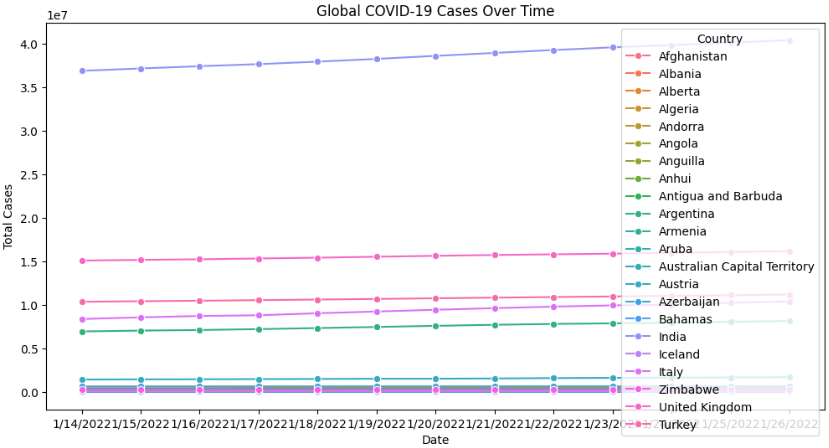
plt.title('COVID-19 Recovery Rate Over Time')

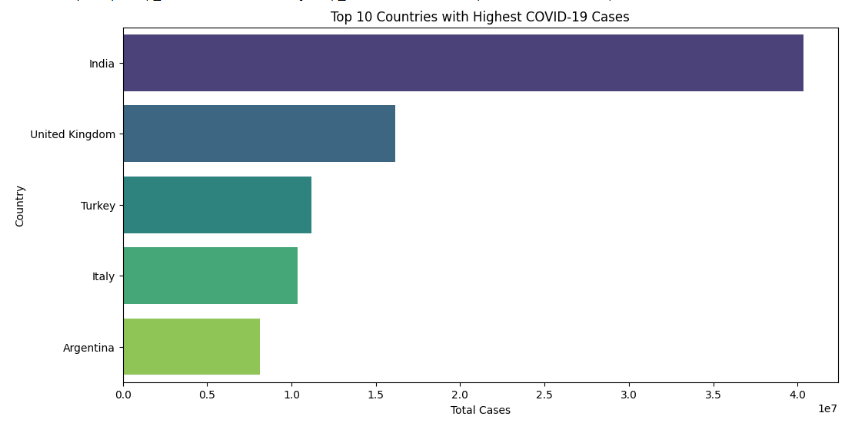
plt.xlabel('Date')

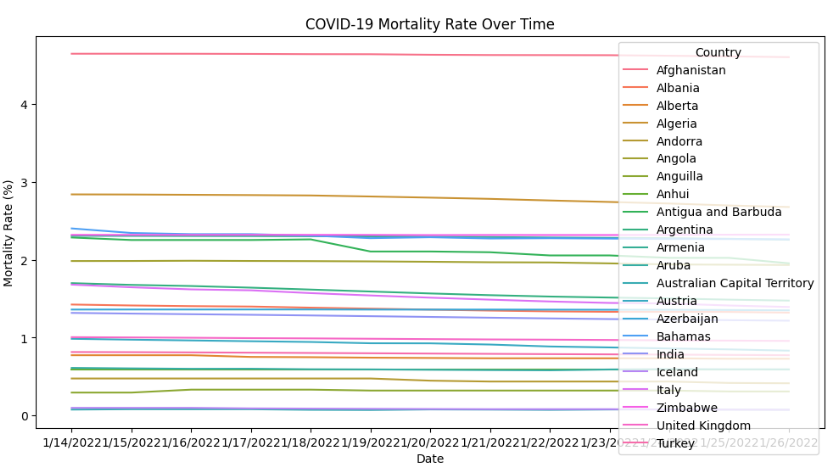
plt.ylabel('Recovery Rate (%)')

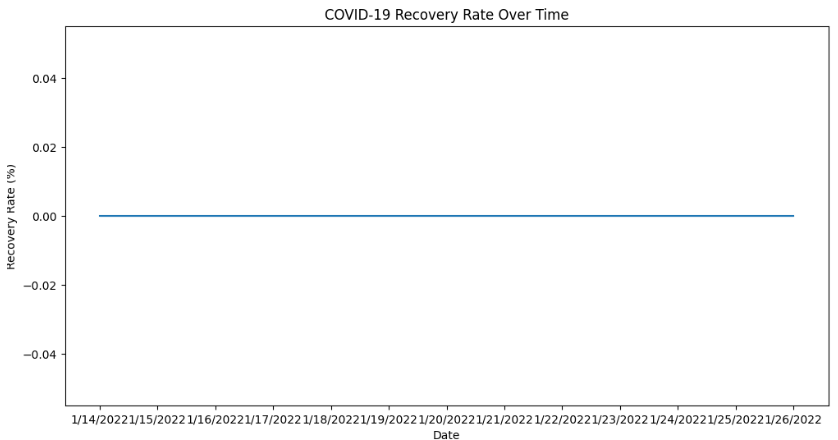
plt.show()

**Output:**

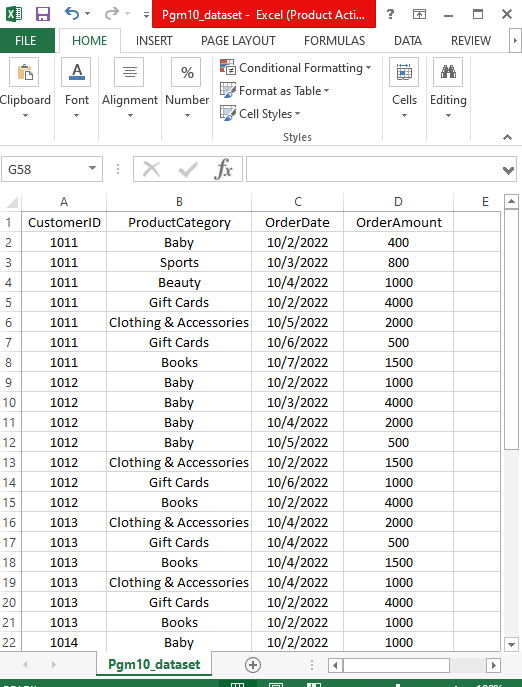








**10. Choose a dataset related to a real-world problem (e.g., retail sales, customer behavior). Explore the dataset, identify interesting patterns, and design a set of visualizations to present the findings effectively. Present the visualizations along with a brief explanation of the insights gained.**

****

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

retail\_data = pd.read\_csv('Pgm10\_dataset.csv')

# Display the first few rows of the dataset

print(retail\_data.head())

# Get summary statistics

print(retail\_data.describe())

# Check for missing values

print(retail\_data.isnull().sum())

retail\_data['OrderDate'] = pd.to\_datetime(retail\_data['OrderDate'])

plt.figure(figsize=(12, 6))

sns.lineplot(x='OrderDate', y='OrderAmount',marker = 'o', data=retail\_data)

plt.title('Sales Trend Over Time')

plt.xlabel('Order Date')

plt.ylabel('Order Amount')

plt.show()

customer\_segmentation = retail\_data.groupby('CustomerID').agg({'OrderAmount': 'sum', 'OrderDate': 'count'}).reset\_index()

customer\_segmentation.rename(columns={'OrderDate': 'Frequency', 'OrderAmount': 'TotalSpending'}, inplace=True)

# Plot

plt.figure(figsize=(10, 6))

sns.scatterplot(x='Frequency', y='TotalSpending', data=customer\_segmentation)

plt.title('Customer Segmentation')

plt.xlabel('Frequency of Purchases')

plt.ylabel('Total Spending')

plt.show()

plt.figure(figsize=(12, 6))

sns.barplot(x='ProductCategory', y='OrderAmount', data=retail\_data, estimator=sum)

plt.title('Sales by Product Category')

plt.xlabel('Product Category')

plt.ylabel('Total Sales')

plt.xticks(rotation=45)

plt.show()

**Output:**

