



RAJALAKSHMI ENGINEERING COLLEGE

Approved by AICTE | Affiliated to Anna University | Accredited by NAAC

**Department of Computer Science and
Engineering**

**CS23334 Fundamentals of Data Science Lab
III semester II Year (2023R)**

Name of the Student : Grace Jeyarani P

Register Number : 2116240701158

1. Create a DataFrame from a dictionary Q: Create a pandas DataFrame using the following dictionary: python CopyEdit data = { 'Name': ['Alice', 'Bob', 'Charlie'], 'Age': [25, 30, 35], 'City': ['New York', 'Los Angeles', 'Chicago'] }

In [1]:

```
import pandas as pd

data = {
    'Name': ['Alice', 'Bob', 'Charlie'],
    'Age': [25, 30, 35],
    'City': ['New York', 'Los Angeles', 'Chicago']
}

df = pd.DataFrame(data)
print(df)
```

	Name	Age	City
0	Alice	25	New York
1	Bob	30	Los Angeles
2	Charlie	35	Chicago

1. Read a CSV file Q: Read a CSV file named data.csv into a DataFrame and print the first 5 rows.

In [13]:

```
import pandas as pd

data = {
    'Name': ['Alice', 'Bob', 'Charlie'],
    'Age': [25, 30, 35],
    'City': ['New York', 'Los Angeles', 'Chicago']
}

df = pd.DataFrame(data)
df.to_csv('data.csv', index=False)
df = pd.read_csv('data.csv')
print(df.head())
```

	Name	Age	City
0	Alice	25	New York
1	Bob	30	Los Angeles
2	Charlie	35	Chicago

1. Filter rows based on a condition Q: From a DataFrame df, print only the rows where Age is greater than 28.

In [2]:

```
filtered_df = df[df['Age'] > 28]
print(filtered_df)
```

	Name	Age	City
1	Bob	30	Los Angeles
2	Charlie	35	Chicago

1. Add a new column Q: Add a new column Salary with values [50000, 60000, 70000].

```
In [3]: df['Salary'] = [50000, 60000, 70000]
print(df)
```

	Name	Age	City	Salary
0	Alice	25	New York	50000
1	Bob	30	Los Angeles	60000
2	Charlie	35	Chicago	70000

1. Group by a column and find the mean Q: Given a DataFrame with columns Department and Salary, find the average salary per department.

```
In [4]: import pandas as pd

data = {
'Department': ['HR', 'IT', 'HR', 'IT', 'Finance'],
'Salary': [40000, 60000, 42000, 62000, 50000]
}

df = pd.DataFrame(data)
mean_salary = df.groupby('Department')['Salary'].mean()
print(mean_salary)
```

Department	Salary
Finance	50000.0
HR	41000.0
IT	61000.0

Name: Salary, dtype: float64

1. Replace values in a column Q: Replace all occurrences of 'New York' in City with 'NYC'.

```
In [6]: import pandas as pd

data = {
'Name': ['Alice', 'Bob', 'Charlie'],
'Age': [25, 30, 35],
'City': ['New York', 'Los Angeles', 'Chicago']
}

df = pd.DataFrame(data)
df['City'] = df['City'].replace('New York', 'NYC')
print(df)
```

	Name	Age	City
0	Alice	25	NYC
1	Bob	30	Los Angeles
2	Charlie	35	Chicago

1. Drop a column Q: Drop the Age column from the DataFrame.

```
In [7]: df = df.drop(columns=['Age'])
print(df)
```

	Name	City
0	Alice	NYC
1	Bob	Los Angeles
2	Charlie	Chicago

1. Sort the DataFrame by a column Q: Sort the DataFrame by Salary in descending order.

```
In [9]: import pandas as pd

data = {
'Department': ['HR', 'IT', 'HR', 'IT', 'Finance'],
'Salary': [40000, 60000, 42000, 62000, 50000]
}

df = pd.DataFrame(data)
df_sorted = df.sort_values(by='Salary', ascending=False)
print(df_sorted)
```

```
Department    Salary
3            IT    62000
1            IT    60000
4        Finance   50000
2            HR    42000
0            HR    40000
```

1. Find missing values Q: Check for missing values in the DataFrame.

```
In [10]: print(df.isnull())
print(df.isnull().sum())
```

```
Department    Salary
0      False    False
1      False    False
2      False    False
3      False    False
4      False    False
Department      0
Salary         0
dtype: int64
```

1. Save DataFrame to a CSV file Q: Save the DataFrame to a CSV file named output.csv.

```
In [11]: import pandas as pd

data = {
'Name': ['Alice', 'Bob', 'Charlie'],
'Age': [25, 30, 35],
'City': ['New York', 'Los Angeles', 'Chicago']
}

df = pd.DataFrame(data)
df.to_csv('output.csv', index=False)
```

```
In [ ]:
```

1. Create a DataFrame from the following data: `data = { 'Name': ['Alice', 'Bob', 'Charlie', 'David'], 'Age': [24, 27, 22, 32], 'City': ['New York', 'Los Angeles', 'Chicago', 'Houston'] }` Write code to:
- a) Display the first two rows
 - b) Print the column names
 - c) Show the shape of the DataFrame
 - d) Display the summary info of the DataFrame

```
In [2]: import pandas as pd

data = {
    'Name': ['Alice', 'Bob', 'Charlie', 'David'],
    'Age': [24, 27, 22, 32],
    'City': ['New York', 'Los Angeles', 'Chicago', 'Houston']
}

df = pd.DataFrame(data)

print(df.head(2))
print(df.columns.tolist())
print(df.shape)
print(df.info())
```

	Name	Age	City
0	Alice	24	New York
1	Bob	27	Los Angeles
			['Name', 'Age', 'City']
			(4, 3)

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4 entries, 0 to 3
Data columns (total 3 columns):
 # Column Non-Null Count Dtype
--- ---
 0 Name 4 non-null object
 1 Age 4 non-null int64
 2 City 4 non-null object
dtypes: int64(1), object(2)
memory usage: 224.0+ bytes
None

1. Using the DataFrame above, write code to:
- a) Select only the Name column
 - b) Select both Name and City columns
 - c) Select the second row using `.iloc[1]`
 - d) Select the row where Name is 'Charlie' using `.loc`

```
In [3]: print(df['Name'])
print(df[['Name', 'City']])
print(df.iloc[1])
print(df.loc[df['Name'] == 'Charlie'])
```

```

0      Alice
1      Bob
2    Charlie
3    David
Name: Name, dtype: object
      Name      City
0    Alice   New York
1    Bob   Los Angeles
2  Charlie   Chicago
3  David   Houston
Name          Bob
Age         27
City   Los Angeles
Name: 1, dtype: object
      Name  Age      City
2  Charlie  22   Chicago

```

1. Filter the DataFrame to show:
- a) People older than 25
 - b) People living in 'Chicago' or 'Houston'
 - c) People whose age is between 23 and 30

```
In [4]: print(df[df['Age'] > 25])
print(df[df['City'].isin(['Chicago', 'Houston'])])
print(df[(df['Age'] >= 23) & (df['Age'] <= 30)])
```

```

      Name  Age      City
1    Bob   27   Los Angeles
3  David   32   Houston
      Name  Age      City
2  Charlie   22   Chicago
3  David   32   Houston
      Name  Age      City
0  Alice   24   New York
1  Bob    27   Los Angeles

```

1. Modify the DataFrame:
- a) Add a new column Score with values [85, 90, 88, 95]
 - b) Change Bob's age to 28
 - c) Remove the City column
 - d) Drop the row of David

```
In [5]: df['Score'] = [85, 90, 88, 95]
df.loc[df['Name'] == 'Bob', 'Age'] = 28
df = df.drop(columns=['City'])
df = df[df['Name'] != 'David']
print(df)
```

```

      Name  Age  Score
0  Alice   24    85
1    Bob   28    90
2  Charlie   22    88

```

1. Create a new DataFrame: `data = { 'Department': ['HR', 'IT', 'HR', 'IT'], 'Salary': [30000, 50000, 35000, 55000], 'Experience': [2, 5, 3, 6] }`
- Write code to:
- a) Group by Department and find average Salary
 - b) Find maximum Experience in each Department
 - c) Calculate total Salary paid

```
In [6]: import pandas as pd
```

```

data = {
    'Department': ['HR', 'IT', 'HR', 'IT'],
    'Salary': [30000, 50000, 35000, 55000],
    'Experience': [2, 5, 3, 6]
}

df2 = pd.DataFrame(data)

print(df2.groupby('Department')['Salary'].mean())
print(df2.groupby('Department')['Experience'].max())
print(df2['Salary'].sum())

```

```

Department
HR      32500.0
IT      52500.0
Name: Salary, dtype: float64
Department
HR      3
IT      6
Name: Experience, dtype: int64
170000

```

- Given a DataFrame with missing values: data = { 'Student': ['John', 'Emma', 'Sam', 'Olivia'], 'Marks': [80, None, 75, 90] } Write code to:
 - Fill missing marks with 0
 - Drop rows with missing values
 - Sort the DataFrame by Marks in descending order

```

In [7]: import pandas as pd
data = {
    'Student': ['John', 'Emma', 'Sam', 'Olivia'],
    'Marks': [80, None, 75, 90]
}
df3 = pd.DataFrame(data)
print(df3.fillna(0))
print(df3.dropna())
print(df3.sort_values(by='Marks', ascending=False))

```

```

Student  Marks
0   John    80.0
1   Emma     0.0
2   Sam     75.0
3  Olivia   90.0
      Student  Marks
0   John    80.0
2   Sam     75.0
3  Olivia   90.0
      Student  Marks
3  Olivia   90.0
0   John    80.0
2   Sam     75.0
1  Emma     NaN

```

- For any DataFrame:
 - Save it as a CSV file named students.csv
 - Load a CSV file named employees.csv
 - Set Name as the index
 - Reset the index

```

In [9]: import pandas as pd
data = {
    'Name': ['Alice', 'Bob', 'Charlie', 'David'],

```

```

'Age': [24, 27, 22, 32],
'City': ['New York', 'Los Angeles', 'Chicago', 'Houston']
}
df = pd.DataFrame(data)
df.to_csv('students.csv', index=False)
dummy_employees = pd.DataFrame({
    'Name': ['Eve', 'Frank'],
    'Age': [29, 31],
    'Department': ['IT', 'HR']
})
dummy_employees.to_csv('employees.csv', index=False)
employees_df = pd.read_csv('employees.csv')
df.set_index('Name', inplace=True)
df.reset_index(inplace=True)
print(employees_df)
print(df)

```

	Name	Age	Department
0	Eve	29	IT
1	Frank	31	HR

	Name	Age	City
0	Alice	24	New York
1	Bob	27	Los Angeles
2	Charlie	22	Chicago
3	David	32	Houston

8) You are given the following DataFrame: import pandas as pd data = { 'Product': ['Laptop', 'Tablet', 'Smartphone', 'Monitor', 'Keyboard'], 'Price': [70000, 30000, 25000, 15000, 2000], 'Stock': [10, 25, 50, 15, 100] } df = pd.DataFrame(data) Task: Write a line of code using .loc[] to display the details of the first and third products in the DataFrame.

In [10]:

```

import pandas as pd
data = {
    'Product': ['Laptop', 'Tablet', 'Smartphone', 'Monitor', 'Keyboard'],
    'Price': [70000, 30000, 25000, 15000, 2000],
    'Stock': [10, 25, 50, 15, 100]
}
df = pd.DataFrame(data)
print(df.loc[[0, 2]])

```

	Product	Price	Stock
0	Laptop	70000	10
2	Smartphone	25000	50

9) You are given the following data: import pandas as pd data = { 'Subject': ['Math', 'Science', 'English'], 'Marks': [88, 92, 85] }

Task: Create a DataFrame from the above data and assign custom row labels: 'Student1', 'Student2', and 'Student3' using the index argument. Then, using .loc[], print the marks obtained by 'Student2'.

In [11]:

```

import pandas as pd

data = {
    'Subject': ['Math', 'Science', 'English'],
    'Marks': [88, 92, 85]
}

```

```
df = pd.DataFrame(data, index=['Student1', 'Student2', 'Student3'])

print(df.loc['Student2'])

Subject      Science
Marks          92
Name: Student2, dtype: object
```

1. You are assigned a data creation task. Based on the field/topic given to you, use Pandas to create a CSV file containing at least 25 entries. Each record should have relevant columns (fields) suitable for your dataset. Steps:
 2. Identify 6–10 suitable columns for your assigned topic.
 3. Generate realistic sample data (manually or using random generation).
 4. Create a Pandas DataFrame with the data.
 5. Save the data as a CSV file with a suitable filename (e.g., cricketers.csv).

```
In [3]: import pandas as pd
cricketers = [
    {"Name": "Virat Kohli", "Age": 34, "Country": "India", "Matches_Played": 254, "F
    {"Name": "Joe Root", "Age": 33, "Country": "England", "Matches_Played": 215, "F
    {"Name": "Steve Smith", "Age": 35, "Country": "Australia", "Matches_Played": 19
    {"Name": "Kane Williamson", "Age": 34, "Country": "New Zealand", "Matches_Playe
    {"Name": "Babar Azam", "Age": 30, "Country": "Pakistan", "Matches_Played": 150,
    {"Name": "Ben Stokes", "Age": 33, "Country": "England", "Matches_Played": 180,
    {"Name": "Rohit Sharma", "Age": 36, "Country": "India", "Matches_Played": 230,
    {"Name": "David Warner", "Age": 37, "Country": "Australia", "Matches_Played": 2
    {"Name": "Shakib Al Hasan", "Age": 36, "Country": "Bangladesh", "Matches_Played
    {"Name": "Quinton de Kock", "Age": 32, "Country": "South Africa", "Matches_Play
    {"Name": "Jasprit Bumrah", "Age": 31, "Country": "India", "Matches_Played": 136
    {"Name": "Pat Cummins", "Age": 32, "Country": "Australia", "Matches_Played": 14
    {"Name": "Rashid Khan", "Age": 26, "Country": "Afghanistan", "Matches_Played": 1
    {"Name": "Trent Boult", "Age": 35, "Country": "New Zealand", "Matches_Played": 1
    {"Name": "Mohammed Shami", "Age": 34, "Country": "India", "Matches_Played": 156
    {"Name": "KL Rahul", "Age": 32, "Country": "India", "Matches_Played": 140, "Rur
    {"Name": "Faf du Plessis", "Age": 39, "Country": "South Africa", "Matches_Playe
    {"Name": "Marnus Labuschagne", "Age": 30, "Country": "Australia", "Matches_Play
    {"Name": "Ravindra Jadeja", "Age": 35, "Country": "India", "Matches_Played": 19
    {"Name": "Tim Southee", "Age": 36, "Country": "New Zealand", "Matches_Played": 1
    {"Name": "Shaheen Afridi", "Age": 25, "Country": "Pakistan", "Matches_Played": 1
    {"Name": "Dewald Brevis", "Age": 22, "Country": "South Africa", "Matches_Played
    {"Name": "Jos Buttler", "Age": 34, "Country": "England", "Matches_Played": 160,
    {"Name": "Rishabh Pant", "Age": 27, "Country": "India", "Matches_Played": 100,
    {"Name": "Mitchell Starc", "Age": 35, "Country": "Australia", "Matches_Played": 1
]
df=pd.DataFrame(cricketers)
df.to_csv('Cricketers.csv',index=False)
print("CSV file 'Cricketers.csv' created successfully.")
```

CSV file 'Cricketers.csv' created successfully.

```
In [4]: import pandas as pd
restaurant_menu = [
    {
        "Dish_ID": "DISH001",
        "Dish_Name": "Tomato Basil Soup",
        "Category": "Starter",
        "Price": 210,
        "Ingredients": ["Tomato", "Basil", "Garlic", "Cream"],
        "Vegetarian": "Yes",
        "Preparation_Time": 20
    },
    {
        "Dish_ID": "DISH002",
        "Dish_Name": "Grilled Chicken",
        "Category": "Main Course",
        "Price": 390,
        "Ingredients": ["Chicken", "Garlic", "Butter", "Lemon"]
    }
]
```

```
        "Vegetarian": "No",
        "Preparation_Time": 35
    },
    {
        "Dish_ID": "DISH003",
        "Dish_Name": "Paneer Tikka",
        "Category": "Starter",
        "Price": 280,
        "Ingredients": ["Paneer", "Yogurt", "Spices", "Onion", "Capsicum"],
        "Vegetarian": "Yes",
        "Preparation_Time": 25
    },
    {
        "Dish_ID": "DISH004",
        "Dish_Name": "Beef Steak",
        "Category": "Main Course",
        "Price": 470,
        "Ingredients": ["Beef", "Butter", "Garlic", "Pepper"],
        "Vegetarian": "No",
        "Preparation_Time": 40
    },
    {
        "Dish_ID": "DISH005",
        "Dish_Name": "Veggie Burger",
        "Category": "Main Course",
        "Price": 320,
        "Ingredients": ["Bread", "Lettuce", "Tomato", "Cheese", "Onion"],
        "Vegetarian": "Yes",
        "Preparation_Time": 30
    },
    {
        "Dish_ID": "DISH006",
        "Dish_Name": "Chicken Biryani",
        "Category": "Main Course",
        "Price": 450,
        "Ingredients": ["Chicken", "Rice", "Spices", "Yogurt", "Onion"],
        "Vegetarian": "No",
        "Preparation_Time": 40
    },
    {
        "Dish_ID": "DISH007",
        "Dish_Name": "Mushroom Risotto",
        "Category": "Main Course",
        "Price": 370,
        "Ingredients": ["Mushroom", "Rice", "Cream", "Cheese", "Garlic"],
        "Vegetarian": "Yes",
        "Preparation_Time": 35
    },
    {
        "Dish_ID": "DISH008",
        "Dish_Name": "Chocolate Lava Cake",
        "Category": "Dessert",
        "Price": 250,
        "Ingredients": ["Chocolate", "Flour", "Butter", "Egg"],
        "Vegetarian": "No",
        "Preparation_Time": 20
    },
    {
        "Dish_ID": "DISH009",
        "Dish_Name": "Fruit Salad",
        "Category": "Dessert",
        "Price": 180,
        "Ingredients": ["Apple", "Banana", "Grapes", "Orange", "Mint"],
        "Vegetarian": "Yes",
        "Preparation_Time": 15
    }
]
```

```
        "Preparation_Time": 15
    },
    {
        "Dish_ID": "DISH010",
        "Dish_Name": "Cold Coffee",
        "Category": "Beverage",
        "Price": 150,
        "Ingredients": ["Coffee", "Milk", "Sugar", "Ice"],
        "Vegetarian": "Yes",
        "Preparation_Time": 10
    },
    {
        "Dish_ID": "DISH011",
        "Dish_Name": "Masala Chai",
        "Category": "Beverage",
        "Price": 120,
        "Ingredients": ["Tea", "Milk", "Spices", "Sugar"],
        "Vegetarian": "Yes",
        "Preparation_Time": 10
    },
    {
        "Dish_ID": "DISH012",
        "Dish_Name": "Butter Naan",
        "Category": "Main Course",
        "Price": 100,
        "Ingredients": ["Flour", "Butter", "Salt", "Yeast"],
        "Vegetarian": "Yes",
        "Preparation_Time": 15
    },
    {
        "Dish_ID": "DISH013",
        "Dish_Name": "Egg Fried Rice",
        "Category": "Main Course",
        "Price": 300,
        "Ingredients": ["Egg", "Rice", "Onion", "Soy Sauce"],
        "Vegetarian": "No",
        "Preparation_Time": 25
    },
    {
        "Dish_ID": "DISH014",
        "Dish_Name": "Mint Lemonade",
        "Category": "Beverage",
        "Price": 130,
        "Ingredients": ["Mint", "Lemon", "Sugar", "Water"],
        "Vegetarian": "Yes",
        "Preparation_Time": 10
    },
    {
        "Dish_ID": "DISH015",
        "Dish_Name": "Garlic Bread",
        "Category": "Starter",
        "Price": 160,
        "Ingredients": ["Bread", "Garlic", "Butter", "Cheese"],
        "Vegetarian": "Yes",
        "Preparation_Time": 15
    },
    {
        "Dish_ID": "DISH016",
        "Dish_Name": "Peas Pulao",
        "Category": "Main Course",
        "Price": 270,
        "Ingredients": ["Rice", "Peas", "Spices", "Onion"],
        "Vegetarian": "Yes",
        "Preparation_Time": 30
    }
]
```

```
},
{
  "Dish_ID": "DISH017",
  "Dish_Name": "Carrot Halwa",
  "Category": "Dessert",
  "Price": 200,
  "Ingredients": ["Carrot", "Milk", "Sugar", "Ghee", "Cardamom"],
  "Vegetarian": "Yes",
  "Preparation_Time": 30
},
{
  "Dish_ID": "DISH018",
  "Dish_Name": "Yogurt Parfait",
  "Category": "Dessert",
  "Price": 220,
  "Ingredients": ["Yogurt", "Granola", "Honey", "Berries"],
  "Vegetarian": "Yes",
  "Preparation_Time": 15
},
{
  "Dish_ID": "DISH019",
  "Dish_Name": "Spinach Pasta",
  "Category": "Main Course",
  "Price": 350,
  "Ingredients": ["Spinach", "Pasta", "Cream", "Cheese", "Garlic"],
  "Vegetarian": "Yes",
  "Preparation_Time": 30
},
{
  "Dish_ID": "DISH020",
  "Dish_Name": "Cucumber Raita",
  "Category": "Starter",
  "Price": 140,
  "Ingredients": ["Cucumber", "Yogurt", "Salt", "Mint"],
  "Vegetarian": "Yes",
  "Preparation_Time": 10
},
{
  "Dish_ID": "DISH021",
  "Dish_Name": "Chicken Tikka",
  "Category": "Starter",
  "Price": 320,
  "Ingredients": ["Chicken", "Yogurt", "Spices", "Lemon"],
  "Vegetarian": "No",
  "Preparation_Time": 30
},
{
  "Dish_ID": "DISH022",
  "Dish_Name": "Cheese Sandwich",
  "Category": "Starter",
  "Price": 180,
  "Ingredients": ["Bread", "Cheese", "Butter", "Tomato"],
  "Vegetarian": "Yes",
  "Preparation_Time": 15
},
{
  "Dish_ID": "DISH023",
  "Dish_Name": "Lemon Tart",
  "Category": "Dessert",
  "Price": 240,
  "Ingredients": ["Lemon", "Flour", "Butter", "Sugar", "Egg"],
  "Vegetarian": "No",
  "Preparation_Time": 25
},
```

```
{  
    "Dish_ID": "DISH024",  
    "Dish_Name": "Vanilla Ice Cream",  
    "Category": "Dessert",  
    "Price": 160,  
    "Ingredients": ["Milk", "Cream", "Sugar", "Vanilla"],  
    "Vegetarian": "Yes",  
    "Preparation_Time": 20  
},  
{  
    "Dish_ID": "DISH025",  
    "Dish_Name": "Mango Smoothie",  
    "Category": "Beverage",  

```

```

        "Dish_ID": "DISH031",
        "Dish_Name": "Pani Puri",
        "Category": "Starter",
        "Price": 120,
        "Ingredients": ["Semolina", "Potato", "Tamarind", "Mint", "Spices"],
        "Vegetarian": "Yes",
        "Preparation_Time": 15
    },
    {
        "Dish_ID": "DISH032",
        "Dish_Name": "Watermelon Juice",
        "Category": "Beverage",
        "Price": 110,
        "Ingredients": ["Watermelon", "Mint", "Sugar", "Lemon"],
        "Vegetarian": "Yes",
        "Preparation_Time": 10
    },
    {
        "Dish_ID": "DISH033",
        "Dish_Name": "Chana Samosa",
        "Category": "Starter",
        "Price": 150,
        "Ingredients": ["Chickpeas", "Potato", "Flour", "Spices", "Tamarind"],
        "Vegetarian": "Yes",
        "Preparation_Time": 20
    },
    {
        "Dish_ID": "DISH034",
        "Dish_Name": "Kari Dosa",
        "Category": "Main Course",
        "Price": 300,
        "Ingredients": ["Rice flour", "Meat curry", "Onion", "Spices"],
        "Vegetarian": "No",
        "Preparation_Time": 30
    },
    {
        "Dish_ID": "DISH035",
        "Dish_Name": "Fish Fingers",
        "Category": "Starter",
        "Price": 280,
        "Ingredients": ["Fish", "Breadcrumbs", "Egg", "Spices", "Oil"],
        "Vegetarian": "No",
        "Preparation_Time": 25
    }
]
df=pd.DataFrame(restaurant_menu)
df.to_csv('Restaurant_menu.csv',index=False)
print("CSV file'Restaurant_menu.csv'created successfully.")

```

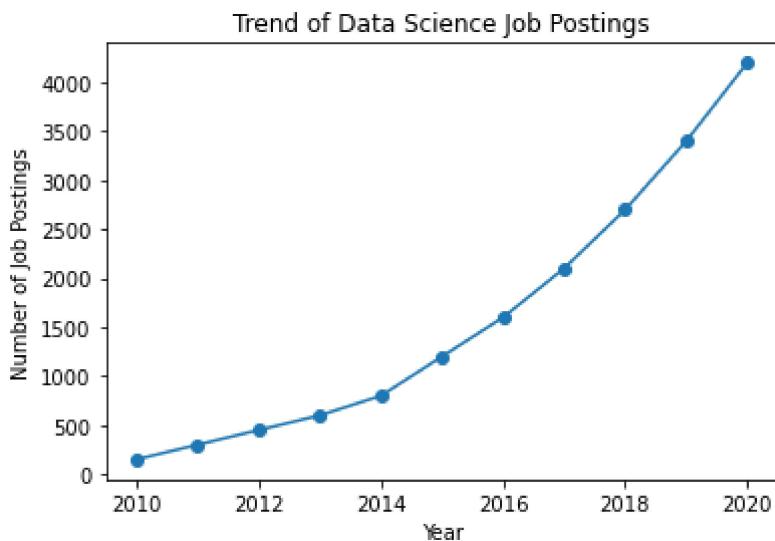
CSV file'Restaurant_menu.csv'created successfully.

In []:

1. The following table shows the number of Data Science job postings recorded over the years 2010 to 2020. Year Job Postings
2010 150
2011 300
2012 450
2013 600
2014 800
2015 1200
2016 1600
2017 2100
2018 2700
2019 3400
2020 4200
Using Python (Pandas and Matplotlib):
2. Create a DataFrame for the given data.
3. Plot a line graph showing the trend of Data Science job postings over the years with markers on data points.
4. Add suitable title and axis labels to the graph.

In [1]:

```
import pandas as pd
import matplotlib.pyplot as plt
data = {'Year': list(range(2010, 2021)),
        'Job Postings': [150, 300, 450, 600, 800, 1200, 1600, 2100, 2700, 3400, 4200]}
df = pd.DataFrame(data)
plt.plot(df['Year'], df['Job Postings'], marker='o')
plt.title('Trend of Data Science Job Postings')
plt.xlabel('Year')
plt.ylabel('Number of Job Postings')
plt.show()
```



1. "Write a Python program using Matplotlib to create a bar chart that shows the distribution of different Data Science roles (Data Analyst, Data Engineer, Data Scientist, ML Engineer, and

Business Analyst) with their respective counts. Add appropriate axis labels and a title to the chart."

In [2]:

```
import matplotlib.pyplot as plt

roles = ['Data Analyst', 'Data Engineer', 'Data Scientist', 'ML Engineer', 'Business Analyst']
counts = [120, 95, 150, 80, 100]

plt.figure(figsize=(10, 6))
plt.bar(roles, counts, color='skyblue', edgecolor='black')

plt.xlabel('Data Science Roles', fontsize=12)
plt.ylabel('Number of Professionals', fontsize=12)
plt.title('Distribution of Data Science Roles', fontsize=14)
```

```
plt.grid(axis='y', linestyle='--', alpha=0.7)

plt.tight_layout()
plt.show()
```



3 "Write a Python program to demonstrate the three main types of data: Structured, Unstructured, and Semi-structured. Use a Pandas DataFrame for structured data, a text string for unstructured data, and a dictionary for semi-structured data. Print each type of data with clear labels."

```
In [3]: import pandas as pd

structured_data = pd.DataFrame({
    'Name': ['Alice', 'Bob', 'Charlie'],
    'Age': [25, 30, 35],
    'Department': ['Data Science', 'Engineering', 'Marketing']
})

unstructured_data = """
Alice is a data scientist with 5 years of experience.
Bob works in the engineering department.
Charlie recently joined the marketing team.
"""

semi_structured_data = {
    'employee1': {
        'name': 'Alice',
        'skills': ['Python', 'Machine Learning'],
        'experience': 5
    },
    'employee2': {
        'name': 'Bob',
        'skills': ['C++', 'Cloud Computing']
    },
    'employee3': {
        'name': 'Charlie',
        'department': 'Marketing'
    }
}
```

```

print("== Structured Data (DataFrame) ==")
print(structured_data)
print("\n== Unstructured Data (Text) ==")
print(unstructured_data.strip())
print("\n== Semi-structured Data (Dictionary) ==")
print(semi_structured_data)

== Structured Data (DataFrame) ==
   Name  Age  Department
0  Alice  25  Data Science
1    Bob  30  Engineering
2 Charlie  35  Marketing

== Unstructured Data (Text) ==
Alice is a data scientist with 5 years of experience.
Bob works in the engineering department.
Charlie recently joined the marketing team.

== Semi-structured Data (Dictionary) ==
{'employee1': {'name': 'Alice', 'skills': ['Python', 'Machine Learning'], 'experience': 5}, 'employee2': {'name': 'Bob', 'skills': ['C++', 'Cloud Computing']}, 'employee3': {'name': 'Charlie', 'department': 'Marketing'}}

```

4 "Write a Python program using the cryptography.fernet module to demonstrate symmetric key encryption and decryption. Encrypt the text 'Rajalakshmi Engineering College' using a generated key, display the encrypted ciphertext, and then decrypt it back to the original text. Print the original, encrypted, and decrypted data."

```

In [4]: from cryptography.fernet import Fernet
key = Fernet.generate_key() # 1
f = Fernet(key) # 2
token = f.encrypt(b"Rajalakshmi Engineering College") # 3
token # 4 (REPL would display the ciphertext bytes)
f.decrypt(token) # 5 (REPL would display the original bytes)
key = Fernet.generate_key() # 6
cipher_suite = Fernet(key) # 7
plain_text = b"Rajalakshmi Engineering College." # 8
cipher_text = cipher_suite.encrypt(plain_text) # 9
decrypted_text = cipher_suite.decrypt(cipher_text) # 10
print("Original Data:", plain_text) # 11
print("Encrypted Data:", cipher_text) # 12
print("Decrypted Data:", decrypted_text) # 13

```

Original Data: b'Rajalakshmi Engineering College.'
 Encrypted Data: b'gAAAAABouQSXx_uTN50LDS0_1qERPosR6Ua2VJBFLKtADCHpngDAxo8osarRd3diMhd7lnSgq4J0DNPvt5LG0mlMbme5fmsFn08vxMqcIw2l3kLJHU0-IMRhdbUqC3WEDJRdsR_oHjW1'
 Decrypted Data: b'Rajalakshmi Engineering College.'

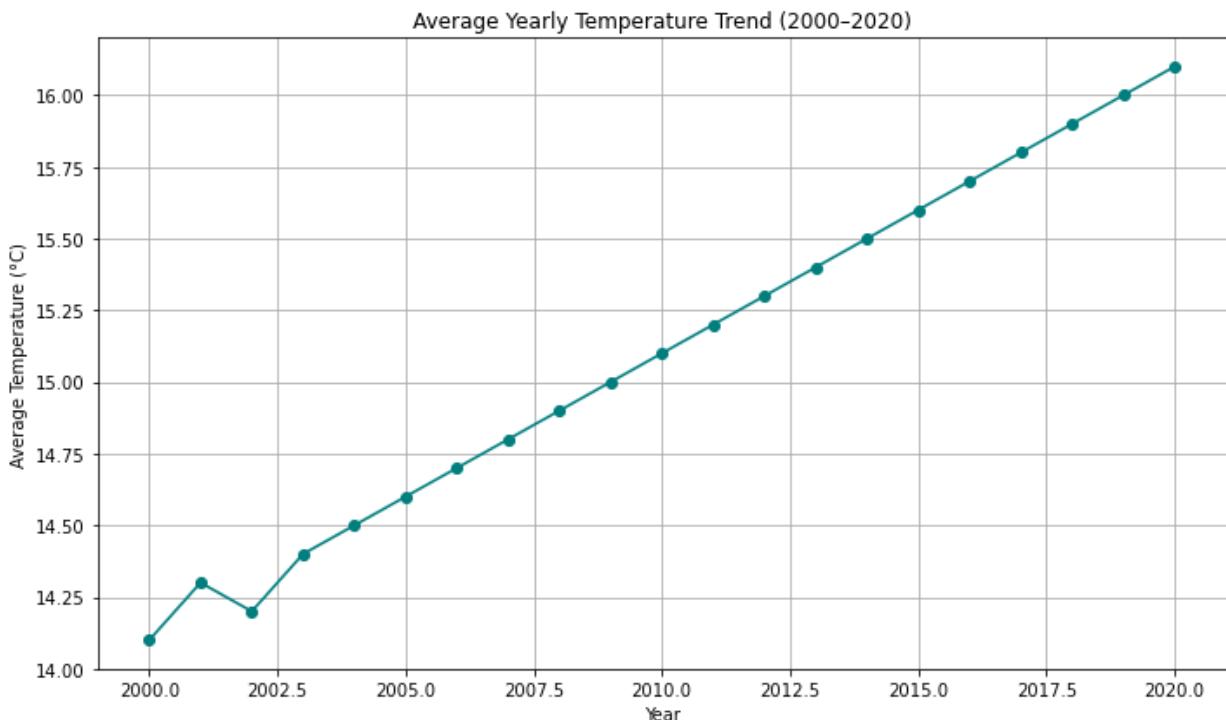
Q1. Line Plot from CSV (Temperature Trends) Download or create a CSV file weather.csv with the following columns: Year, AvgTemperature. Write a Python program using Pandas and Matplotlib to plot a line chart showing the trend of average yearly temperature from 2000 to 2020. • Add markers on the line. • Label the axes (Year, Average Temperature (°C)) and add a title.

```
In [1]: import pandas as pd
import matplotlib.pyplot as plt

data = {
    'Year': list(range(2000, 2021)),
    'AvgTemperature': [
        14.1, 14.3, 14.2, 14.4, 14.5, 14.6, 14.7, 14.8, 14.9, 15.0,
        15.1, 15.2, 15.3, 15.4, 15.5, 15.6, 15.7, 15.8, 15.9, 16.0, 16.1
    ]
}

df = pd.DataFrame(data)
df.to_csv('weather.csv', index=False)

df = pd.read_csv('weather.csv')
plt.figure(figsize=(10, 6))
plt.plot(df['Year'], df['AvgTemperature'], marker='o', linestyle='-', color='teal')
plt.xlabel('Year')
plt.ylabel('Average Temperature (°C)')
plt.title('Average Yearly Temperature Trend (2000-2020)')
plt.grid(True)
plt.tight_layout()
plt.show()
```



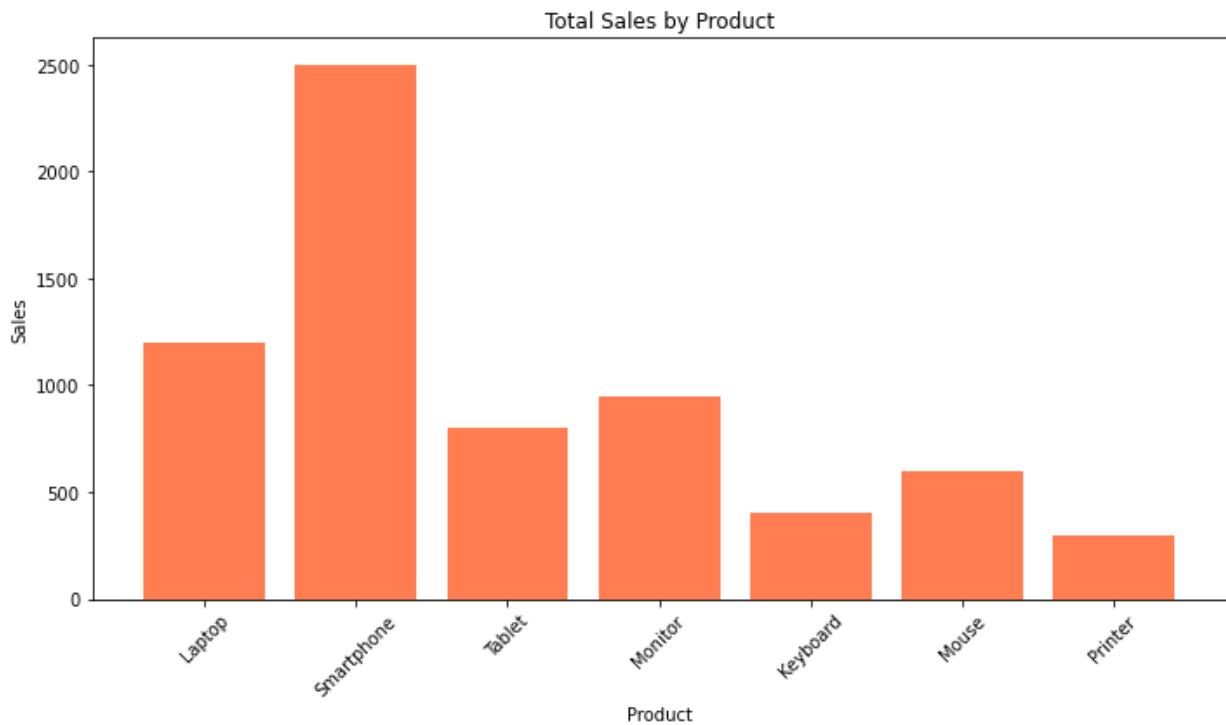
Q2. Bar Chart from CSV (Sales Data) Given a CSV file sales.csv with columns: Product, Sales.
Write a Python program to plot a bar chart showing the total sales of each product. • Add axis labels and a title. • Rotate product names if necessary for better visibility.

```
In [2]: import pandas as pd
import matplotlib.pyplot as plt

data = {
    'Product': ['Laptop', 'Smartphone', 'Tablet', 'Monitor', 'Keyboard', 'Mouse', 'Printer'],
    'Sales': [1200, 2500, 800, 950, 400, 600, 300]
}

df = pd.DataFrame(data)
df.to_csv('sales.csv', index=False)

df = pd.read_csv('sales.csv')
plt.figure(figsize=(10, 6))
plt.bar(df['Product'], df['Sales'], color='coral')
plt.xlabel('Product')
plt.ylabel('Sales')
plt.title('Total Sales by Product')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



Q3. Pie Chart (Movie Genres Distribution) Create a CSV file movies.csv with columns: Genre, Count. Write a Python program to plot a pie chart showing the percentage distribution of movie genres. • Add labels and percentage values to each slice. • Add a title: "Distribution of Movie Genres".

```
In [3]: import pandas as pd
import matplotlib.pyplot as plt
```

```

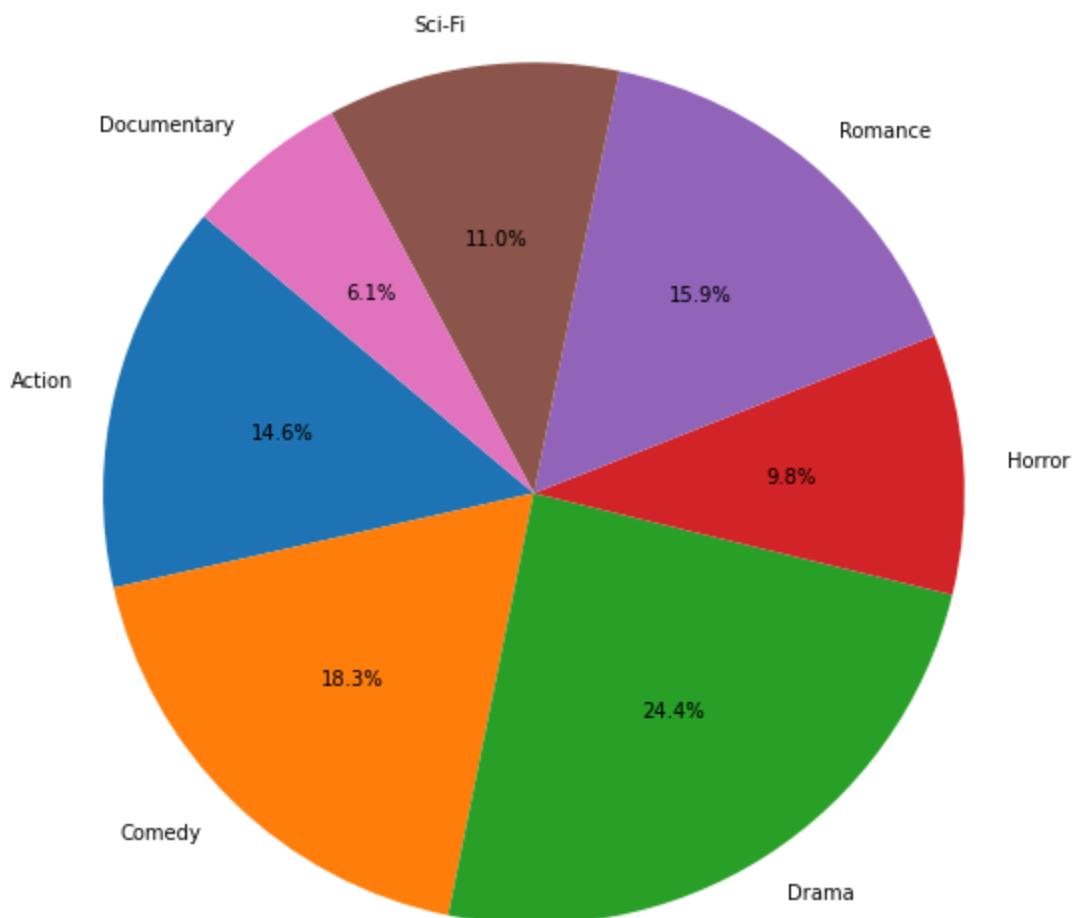
data = {
    'Genre': ['Action', 'Comedy', 'Drama', 'Horror', 'Romance', 'Sci-Fi', 'Documentary'],
    'Count': [120, 150, 200, 80, 130, 90, 50]
}

df = pd.DataFrame(data)
df.to_csv('movies.csv', index=False)

df = pd.read_csv('movies.csv')
plt.figure(figsize=(8, 8))
plt.pie(df['Count'], labels=df['Genre'], autopct='%1.1f%%', startangle=140)
plt.title('Distribution of Movie Genres')
plt.tight_layout()
plt.show()

```

Distribution of Movie Genres



Q4. Scatter Plot (Students' Marks) Given a CSV file students.csv with columns: MathMarks, ScienceMarks. Write a Python program to plot a scatter plot showing the relationship between marks in Mathematics and Science.

- Add axis labels and a title.
- Use different colors/markers if possible.

In [4]:

```

import pandas as pd
import matplotlib.pyplot as plt

```

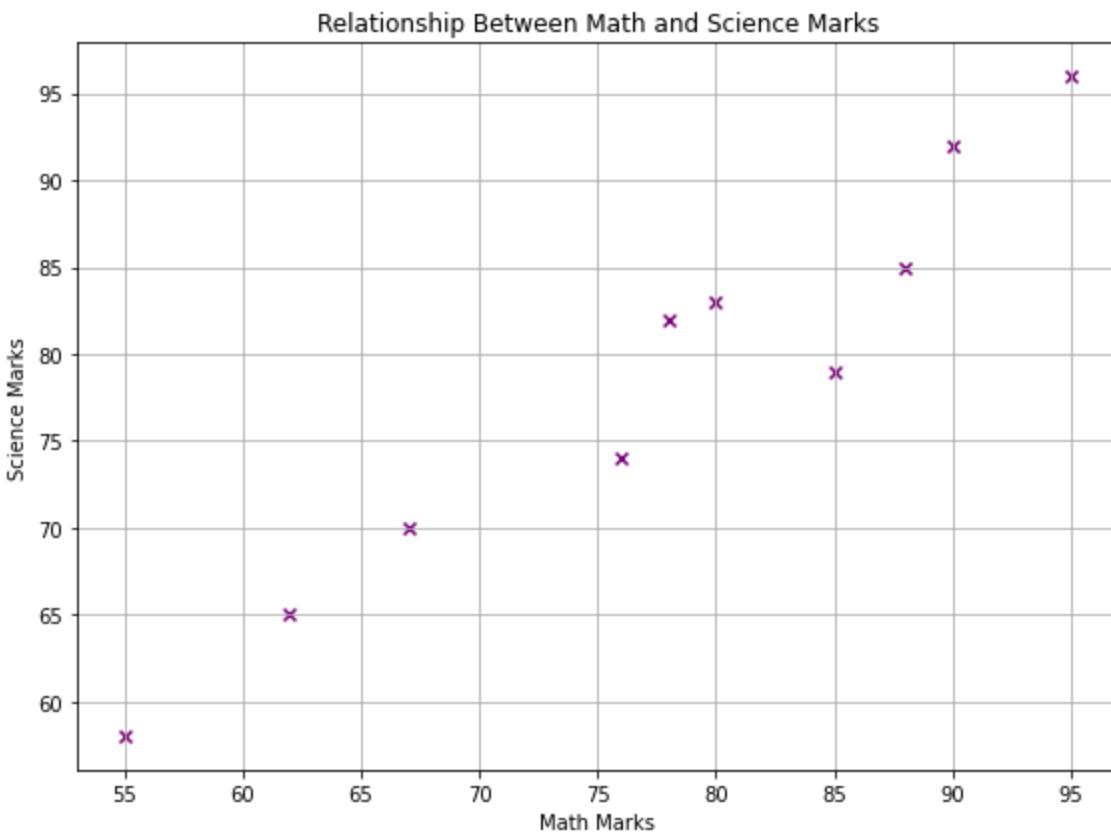
```

data = {
    'MathMarks': [78, 85, 62, 90, 55, 88, 76, 95, 67, 80],
    'ScienceMarks': [82, 79, 65, 92, 58, 85, 74, 96, 70, 83]
}

df = pd.DataFrame(data)
df.to_csv('students.csv', index=False)

df = pd.read_csv('students.csv')
plt.figure(figsize=(8, 6))
plt.scatter(df['MathMarks'], df['ScienceMarks'], color='purple', marker='x')
plt.xlabel('Math Marks')
plt.ylabel('Science Marks')
plt.title('Relationship Between Math and Science Marks')
plt.grid(True)
plt.tight_layout()
plt.show()

```



Q5. Histogram (Customer Ages) Given a CSV file `customers.csv` with a column: Age. Write a Python program to plot a histogram showing the age distribution of customers. • Use 10 bins. • Add axis labels and a title.

In [5]:

```

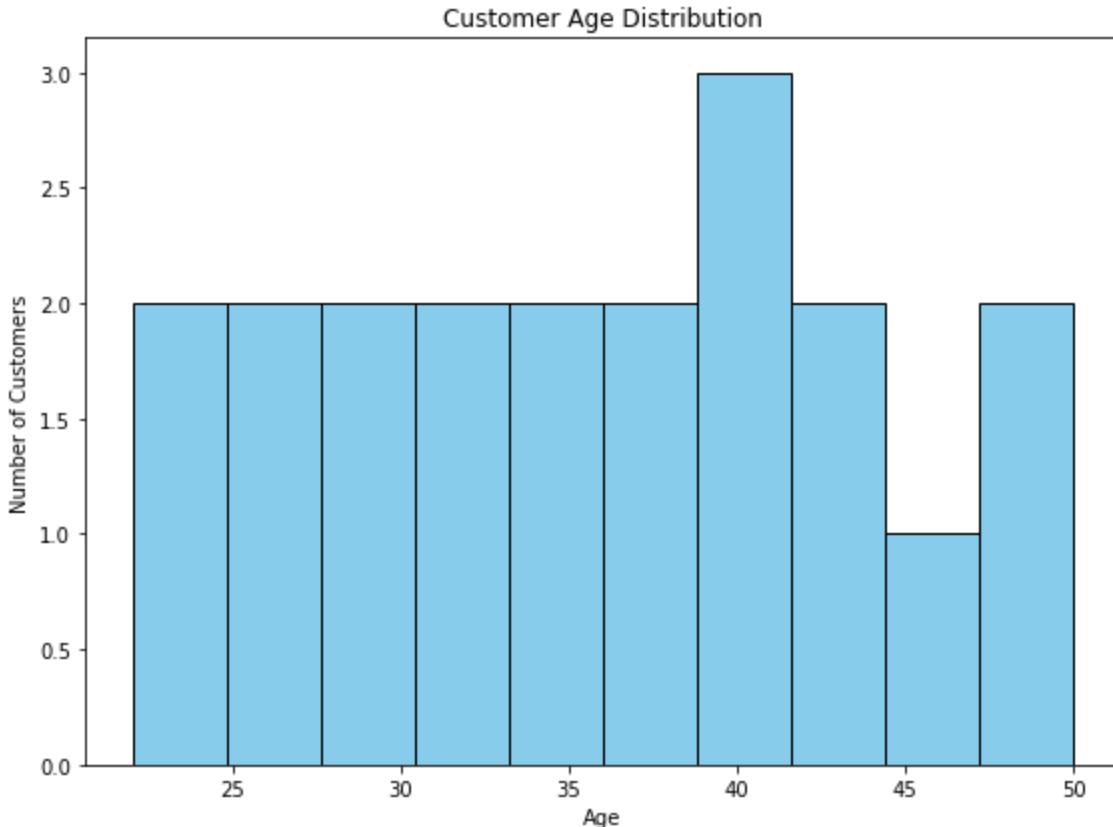
import pandas as pd
import matplotlib.pyplot as plt

data = {
    'Age': [22, 35, 45, 23, 34, 42, 29, 31, 38, 40, 27, 36, 50, 48, 33, 26, 39, 41, 36]
}

df = pd.DataFrame(data)
df.to_csv('customers.csv', index=False)

```

```
df = pd.read_csv('customers.csv')
plt.figure(figsize=(8, 6))
plt.hist(df['Age'], bins=10, color='skyblue', edgecolor='black')
plt.xlabel('Age')
plt.ylabel('Number of Customers')
plt.title('Customer Age Distribution')
plt.tight_layout()
plt.show()
```



Q6. Boxplot (Exam Scores Comparison) Create a CSV file exam_scores.csv with columns: Math, English, Science. Write a Python program to plot a boxplot comparing score distributions across subjects. • Add axis labels and a title.

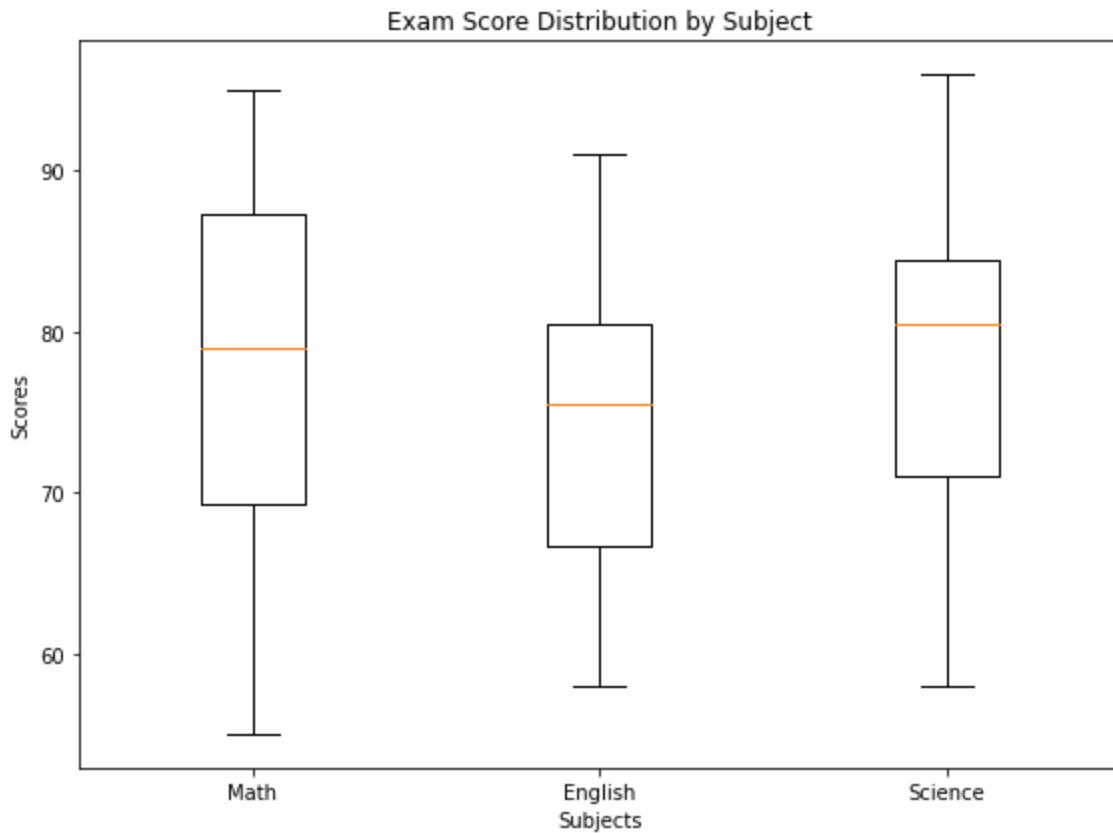
```
In [6]: import pandas as pd
import matplotlib.pyplot as plt

data = {
    'Math': [78, 85, 62, 90, 55, 88, 76, 95, 67, 80],
    'English': [72, 81, 60, 85, 58, 79, 74, 91, 65, 77],
    'Science': [82, 79, 65, 92, 58, 85, 74, 96, 70, 83]
}

df = pd.DataFrame(data)
df.to_csv('exam_scores.csv', index=False)

df = pd.read_csv('exam_scores.csv')
plt.figure(figsize=(8, 6))
plt.boxplot([df['Math'], df['English'], df['Science']], labels=['Math', 'English', 'Science'])
plt.xlabel('Subjects')
plt.ylabel('Scores')
plt.title('Exam Score Distribution by Subject')
```

```
plt.tight_layout()
plt.show()
```



Q7. Area Chart (Website Traffic) Create a CSV file traffic.csv with columns: Month, Visitors. Write a Python program to plot an area chart showing monthly website visitors for a year.

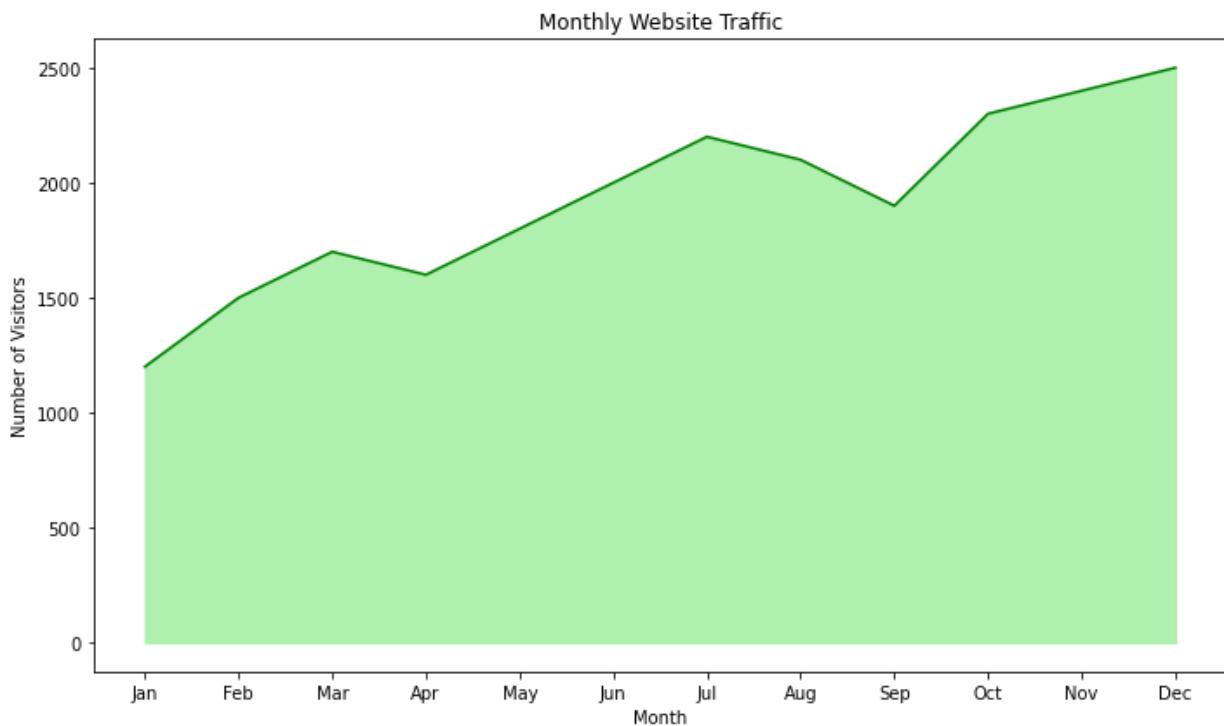
- Add axis labels and a title.
- Use a different color for the filled area.

```
In [7]: import pandas as pd
import matplotlib.pyplot as plt

data = {
    'Month': ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec'],
    'Visitors': [1200, 1500, 1700, 1600, 1800, 2000, 2200, 2100, 1900, 2300, 2400, 2500]
}

df = pd.DataFrame(data)
df.to_csv('traffic.csv', index=False)

df = pd.read_csv('traffic.csv')
plt.figure(figsize=(10, 6))
plt.fill_between(df['Month'], df['Visitors'], color='lightgreen', alpha=0.7)
plt.plot(df['Month'], df['Visitors'], color='green')
plt.xlabel('Month')
plt.ylabel('Number of Visitors')
plt.title('Monthly Website Traffic')
plt.tight_layout()
plt.show()
```



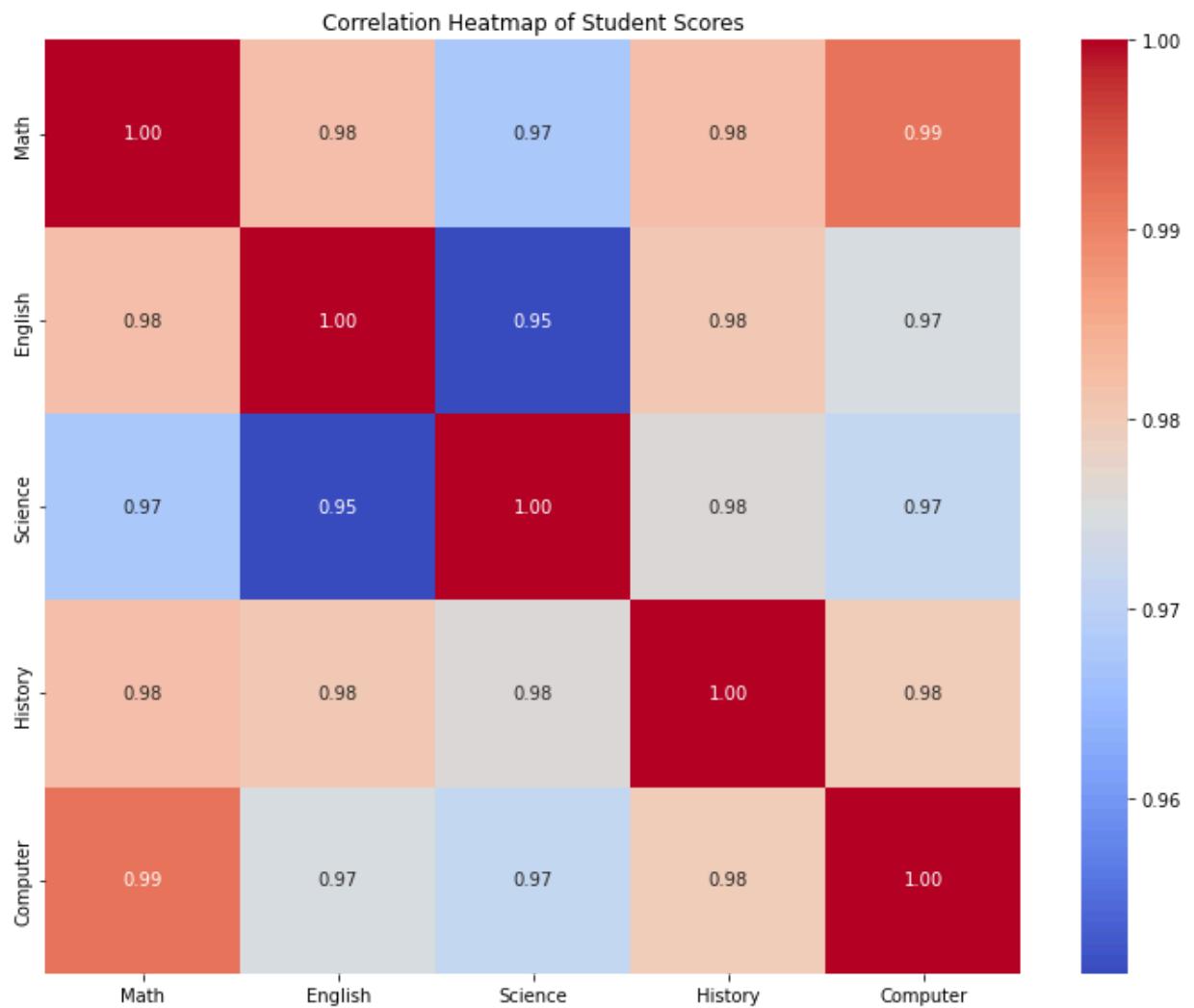
Q8. Heatmap (Correlation Matrix) Given a CSV file `students_scores.csv` with columns: Math, English, Science, History, Computer. Write a Python program using Seaborn + Matplotlib to create a heatmap of the correlation between subjects. • Add a title "Correlation Heatmap of Student Scores".

```
In [8]: import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

data = {
    'Math': [78, 85, 62, 90, 55, 88, 76, 95, 67, 80],
    'English': [72, 81, 60, 85, 58, 79, 74, 91, 65, 77],
    'Science': [82, 79, 65, 92, 58, 85, 74, 96, 70, 83],
    'History': [70, 75, 55, 88, 52, 80, 68, 90, 60, 73],
    'Computer': [85, 90, 70, 95, 65, 92, 80, 98, 72, 88]
}

df = pd.DataFrame(data)
df.to_csv('students_scores.csv', index=False)

df = pd.read_csv('students_scores.csv')
plt.figure(figsize=(10, 8))
sns.heatmap(df.corr(), annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Correlation Heatmap of Student Scores')
plt.tight_layout()
plt.show()
```



1) A retail company has provided a CSV file (sales_data.csv) containing sales transaction details with the following columns: Date, Product, Quantity, Sales, and Region. As a Data Analyst, perform an Exploratory Data Analysis (EDA) using Python (Pandas, NumPy, Matplotlib, and Seaborn) with the following tasks:

1. Load the dataset into a Pandas DataFrame and display the first few records.
2. Identify and handle missing values (replace missing Sales values with the mean, and drop rows with missing Product, Quantity, or Region).
3. Generate summary statistics of the numerical columns.
4. Group the data by Product and compute the total Sales and Quantity sold.
5. Create a bar chart showing the total sales for each product.
6. Convert the Date column to datetime and plot a line chart of total sales over time.
7. Construct a pivot table to analyze sales by Region and Product.
8. Compute the correlation matrix of numerical variables and visualize it using a heatmap.

In [2]:

```
# Import necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from datetime import datetime, timedelta
import random

sns.set(style="whitegrid")

# 1 Create a sample dataset and save it as CSV
np.random.seed(42)
random.seed(42)

# Generate random sample data
num_records = 100

dates = [datetime(2024, 1, 1) + timedelta(days=i) for i in range(num_records)]
products = ['Laptop', 'Smartphone', 'Tablet', 'Headphones', 'Smartwatch']
regions = ['North', 'South', 'East', 'West']

data = {
    'Date': np.random.choice(dates, num_records),
    'Product': np.random.choice(products, num_records),
    'Quantity': np.random.randint(1, 10, num_records),
    'Sales': np.random.uniform(100, 2000, num_records),
    'Region': np.random.choice(regions, num_records)
}

df = pd.DataFrame(data)

# Introduce some missing values for testing
for _ in range(5):
    df.loc[random.randint(0, num_records - 1), 'Sales'] = np.nan
for _ in range(3):
    df.loc[random.randint(0, num_records - 1), 'Product'] = np.nan

# Save to CSV
df.to_csv("sales_data.csv", index=False)
print("✓ Sample CSV file 'sales_data.csv' created successfully!\n")

# 2 Load dataset
```

```

df = pd.read_csv("sales_data.csv")
print("First five records:")
print(df.head())

# [3] Handle missing values
print("\nMissing values before cleaning:")
print(df.isnull().sum())

# Replace missing Sales with mean
df['Sales'] = df['Sales'].fillna(df['Sales'].mean())

# Drop rows with missing Product, Quantity, or Region
df = df.dropna(subset=['Product', 'Quantity', 'Region'])

print("\nMissing values after cleaning:")
print(df.isnull().sum())

# [4] Summary statistics
print("\nSummary statistics:")
print(df.describe())

# [5] Group by Product – Total Sales and Quantity
product_summary = df.groupby('Product').agg(
    Total_Sales=('Sales', 'sum'),
    Total_Quantity=('Quantity', 'sum')
).reset_index()
print("\nTotal Sales and Quantity by Product:")
print(product_summary)

# [6] Bar Chart – Total Sales by Product
plt.figure(figsize=(10, 6))
sns.barplot(x='Product', y='Total_Sales', data=product_summary, palette='viridis')
plt.title("Total Sales by Product")
plt.xlabel("Product")
plt.ylabel("Total Sales")
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()

# [7] Line Chart – Total Sales Over Time
df['Date'] = pd.to_datetime(df['Date'])
sales_over_time = df.groupby('Date')['Sales'].sum().reset_index()

plt.figure(figsize=(12, 6))
sns.lineplot(x='Date', y='Sales', data=sales_over_time, color='blue')
plt.title("Total Sales Over Time")
plt.xlabel("Date")
plt.ylabel("Total Sales")
plt.tight_layout()
plt.show()

# [8] Pivot Table – Sales by Region and Product
pivot_table = pd.pivot_table(
    df,
    values='Sales',
    index='Region',
    columns='Product',
    aggfunc='sum',
    fill_value=0
)
print("\nPivot Table: Sales by Region and Product")
print(pivot_table)

# [9] Correlation Matrix – Heatmap

```

```
corr_matrix = df[['Quantity', 'Sales']].corr()
plt.figure(figsize=(6, 4))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.title("Correlation Matrix Heatmap")
plt.show()
```

Sample CSV file 'sales_data.csv' created successfully!

First five records:

	Date	Product	Quantity	Sales	Region
0	2024-02-21	Smartphone	7	1334.302508	North
1	2024-04-02	Laptop	2	1713.524480	East
2	2024-01-15	Headphones	3	1349.464495	South
3	2024-03-12	Headphones	1	NaN	West
4	2024-03-01	Headphones	5	277.982059	South

Missing values before cleaning:

	Date	Product	Quantity	Sales	Region
	0	3	0	5	0

dtype: int64

Missing values after cleaning:

	Date	Product	Quantity	Sales	Region
	0	0	0	0	0

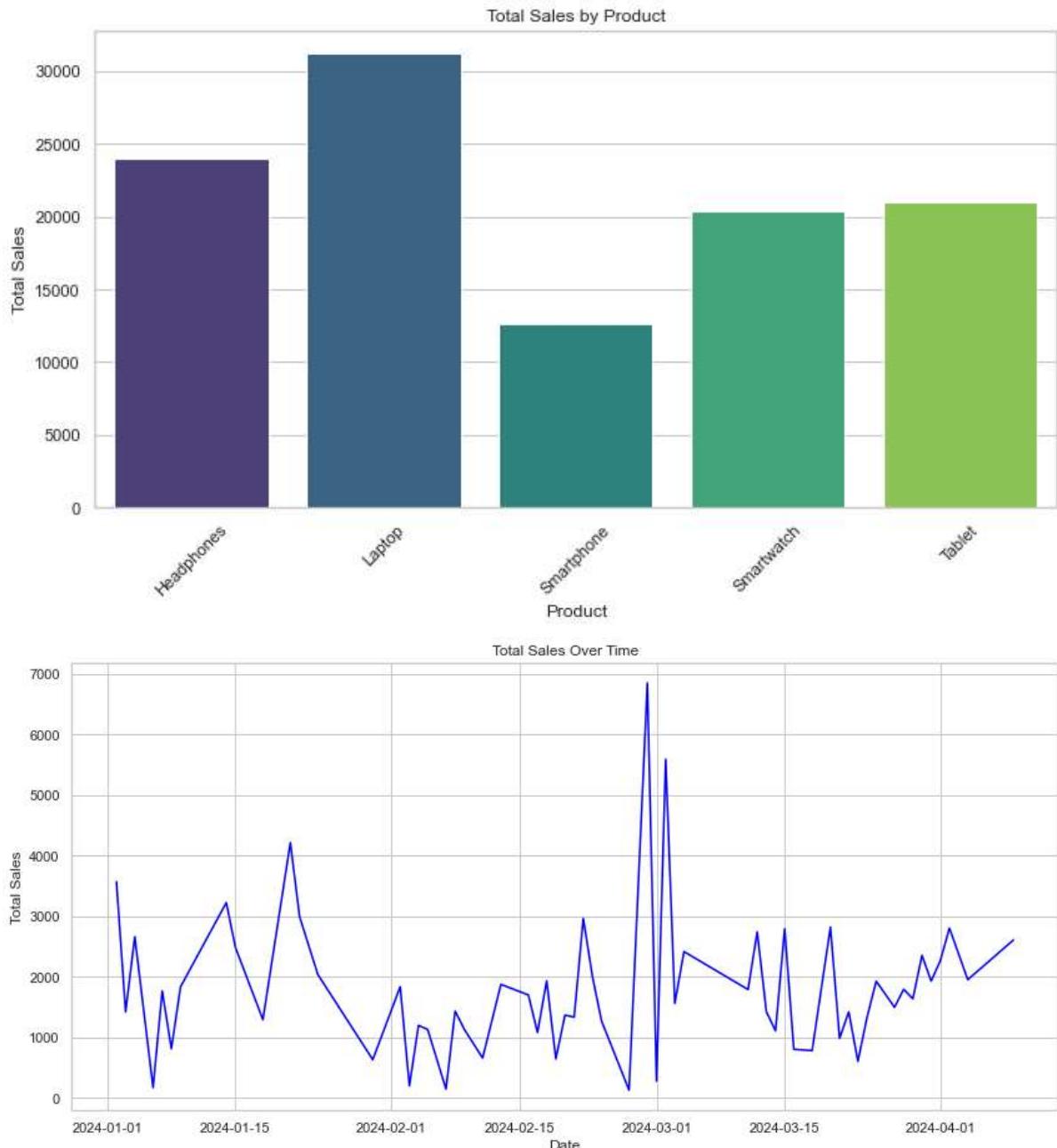
dtype: int64

Summary statistics:

	Quantity	Sales
count	97.000000	97.000000
mean	4.649485	1126.820268
std	2.704322	539.355367
min	1.000000	129.367571
25%	2.000000	713.617299
50%	4.000000	1134.084929
75%	7.000000	1610.853770
max	9.000000	1981.102315

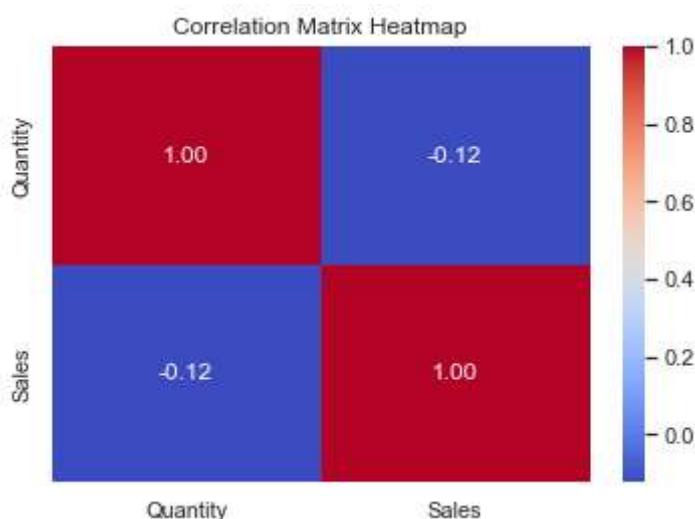
Total Sales and Quantity by Product:

	Product	Total_Sales	Total_Quantity
0	Headphones	23998.741843	100
1	Laptop	31241.931740	124
2	Smartphone	12606.626689	65
3	Smartwatch	20424.290501	78
4	Tablet	21029.975240	84



Pivot Table: Sales by Region and Product

Region	Product	Headphones	Laptop	Smartphone	Smartwatch	Tablet
East	Headphones	8048.963042	3467.978061	2353.044568	436.510291	6037.021855
North	Headphones	3830.124412	12615.386051	2419.128848	5445.806603	4789.152582
South	Headphones	6763.301677	5334.269222	3703.474446	9704.481686	5286.381219
West	Headphones	5356.352712	9824.298406	4130.978828	4837.491921	4917.419585



2) A dataset contains the names of six students (SHREE, DEV, KEERTHI, PRIYA, SHAN, KUMARAN) along with their Higher Secondary Certificate (HSC) exam percentages: 96, 91, 94, 75, 45, 81. Using Matplotlib and NumPy in Python:

1. Plot a bar chart comparing the HSC percentages of the students.
2. Set appropriate labels for the x-axis and y-axis.
3. Rotate the x-axis tick labels for better readability.
4. Add a title ("Comparison of HSC Percentage") with customized font size and color.
5. Display the chart using show().

```
In [3]: import numpy as np
import matplotlib.pyplot as plt

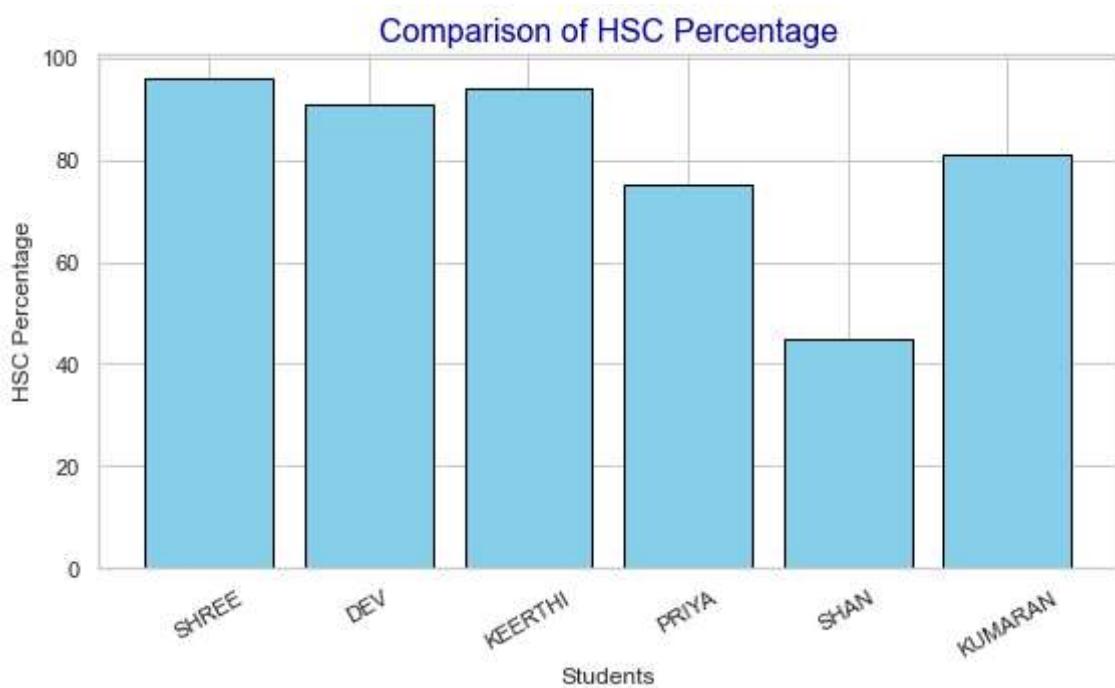
students = np.array(["SHREE", "DEV", "KEERTHI", "PRIYA", "SHAN", "KUMARAN"])
percentages = np.array([96, 91, 94, 75, 45, 81])

plt.figure(figsize=(8, 5))
plt.bar(students, percentages, color='skyblue', edgecolor='black')

plt.xlabel("Students", fontsize=12)
plt.ylabel("HSC Percentage", fontsize=12)
plt.title("Comparison of HSC Percentage", fontsize=16, color='darkblue')

plt.xticks(rotation=30)

plt.tight_layout()
plt.show()
```



3) The following table shows the number of votes received by four candidates in a college election: Candidate Votes Candidate 1 315 Candidate 2 130

Candidate Votes Candidate 3 245 Candidate 4 210

Using Matplotlib in Python, write a program to:

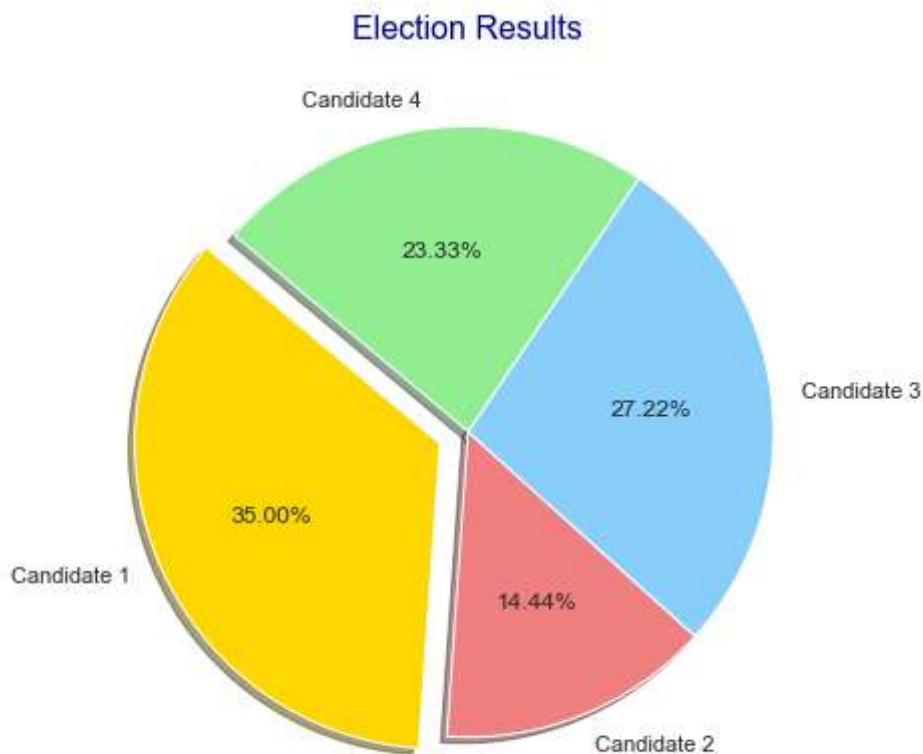
1. Plot a pie chart to represent the election results.
2. Highlight Candidate 1 using the explode parameter.
3. Use different colors for each candidate.
4. Display percentage values on the chart up to two decimal places.
5. Add a suitable title ("Election Results").

```
In [4]: import matplotlib.pyplot as plt

candidates = ["Candidate 1", "Candidate 2", "Candidate 3", "Candidate 4"]
votes = [315, 130, 245, 210]

explode = [0.1, 0, 0, 0]
colors = ["gold", "lightcoral", "lightskyblue", "lightgreen"]

plt.figure(figsize=(7, 7))
plt.pie(votes, labels=candidates, autopct=".2f%%", explode=explode, colors=colors,
plt.title("Election Results", fontsize=16, color="darkblue")
plt.show()
```



- 4) To Count the frequency of occurrence of a word in a body of text is often needed during text processing.

```
In [5]: import nltk
from nltk.tokenize import word_tokenize
from nltk.corpus import gutenberg

nltk.download('gutenberg')
nltk.download('punkt')

sample = gutenberg.raw("austen-emma.txt")
token = word_tokenize(sample)
```

```
wlist = [token[i] for i in range(50)]
wordfreq = [wlist.count(w) for w in wlist]

print("Pairs\n" + str(list(zip(wlist, wordfreq))))
```

```
[nltk_data] Downloading package gutenberg to
[nltk_data]      C:\Users\HDC0422049\AppData\Roaming\nltk_data...
[nltk_data] Package gutenberg is already up-to-date!
[nltk_data] Downloading package punkt to
[nltk_data]      C:\Users\HDC0422049\AppData\Roaming\nltk_data...
[nltk_data] Package punkt is already up-to-date!
Pairs
[('[', 1), ('Emma', 2), ('by', 1), ('Jane', 1), ('Austen', 1), ('1816', 1), (']', 1),
('VOLUME', 1), ('I', 2), ('CHAPTER', 1), ('I', 2), ('Emma', 2), ('Woodhouse', 1),
(',', 5), ('handsome', 1), (',', 5), ('clever', 1), (',', 5), ('and', 3), ('rich', 1),
(',', 5), ('with', 2), ('a', 1), ('comfortable', 1), ('home', 1), ('and', 3),
('happy', 1), ('disposition', 1), (',', 5), ('seemed', 1), ('to', 1), ('unit',
'e', 1), ('some', 1), ('of', 2), ('the', 2), ('best', 1), ('blessings', 1), ('of',
2), ('existence', 1), (';', 1), ('and', 3), ('had', 1), ('lived', 1), ('nearly',
1), ('twenty-one', 1), ('years', 1), ('in', 1), ('the', 2), ('world', 1), ('with',
2)]
```

WEEK 7

1. Write a Python program to collect, load, and perform initial exploration of the Diabetes dataset using Pandas. Display the first few records of the dataset and summarize its structure and contents.

```
In [2]: import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
data = {
    'Pregnancies': np.random.randint(0, 17, size=768),
    'Glucose': np.random.randint(50, 200, size=768),
    'BloodPressure': np.random.randint(30, 122, size=768),
    'SkinThickness': np.random.randint(0, 99, size=768),
    'Insulin': np.random.randint(0, 846, size=768),
    'BMI': np.round(np.random.uniform(15, 67, size=768), 1),
    'DiabetesPedigreeFunction': np.round(np.random.uniform(0.1, 2.5, size=768), 3),
    'Age': np.random.randint(21, 81, size=768),
    'Outcome': np.random.randint(0, 2, size=768)
}
diabetes_df = pd.DataFrame(data)
diabetes_df.to_csv("diabetes.csv", index=False)
diabetes_df = pd.read_csv("diabetes.csv")
print("First 5 rows of the dataset:")
print(diabetes_df.head())
print("\nDataset Information:")
print(diabetes_df.info())
print("\nStatistical Summary:")
print(diabetes_df.describe())
print("\nMissing Values in Each Column:")
print(diabetes_df.isnull().sum())
plt.figure(figsize=(6,4))
sns.countplot(x='Outcome', data=diabetes_df)
plt.title('Distribution of Diabetes Outcome')
plt.xlabel('Outcome (0 = Non-Diabetic, 1 = Diabetic)')
plt.ylabel('Count')
plt.show()
plt.figure(figsize=(10,8))
sns.heatmap(diabetes_df.corr(), annot=True, cmap='coolwarm')
plt.title('Correlation Heatmap of Features')
plt.show()
```

First 5 rows of the dataset:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	\
0	5	150	74	44	31	40.1	
1	1	157	82	21	781	34.5	
2	13	114	42	66	215	35.2	
3	15	79	30	29	52	28.0	
4	10	59	104	41	729	42.6	

	DiabetesPedigreeFunction	Age	Outcome
0	0.469	35	0
1	1.410	30	1
2	0.881	68	1
3	1.714	66	1
4	0.853	40	0

Dataset Information:

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 768 entries, 0 to 767

Data columns (total 9 columns):

#	Column	Non-Null Count	Dtype
0	Pregnancies	768 non-null	int64
1	Glucose	768 non-null	int64
2	BloodPressure	768 non-null	int64
3	SkinThickness	768 non-null	int64
4	Insulin	768 non-null	int64
5	BMI	768 non-null	float64
6	DiabetesPedigreeFunction	768 non-null	float64
7	Age	768 non-null	int64
8	Outcome	768 non-null	int64

dtypes: float64(2), int64(7)

memory usage: 54.1 KB

None

Statistical Summary:

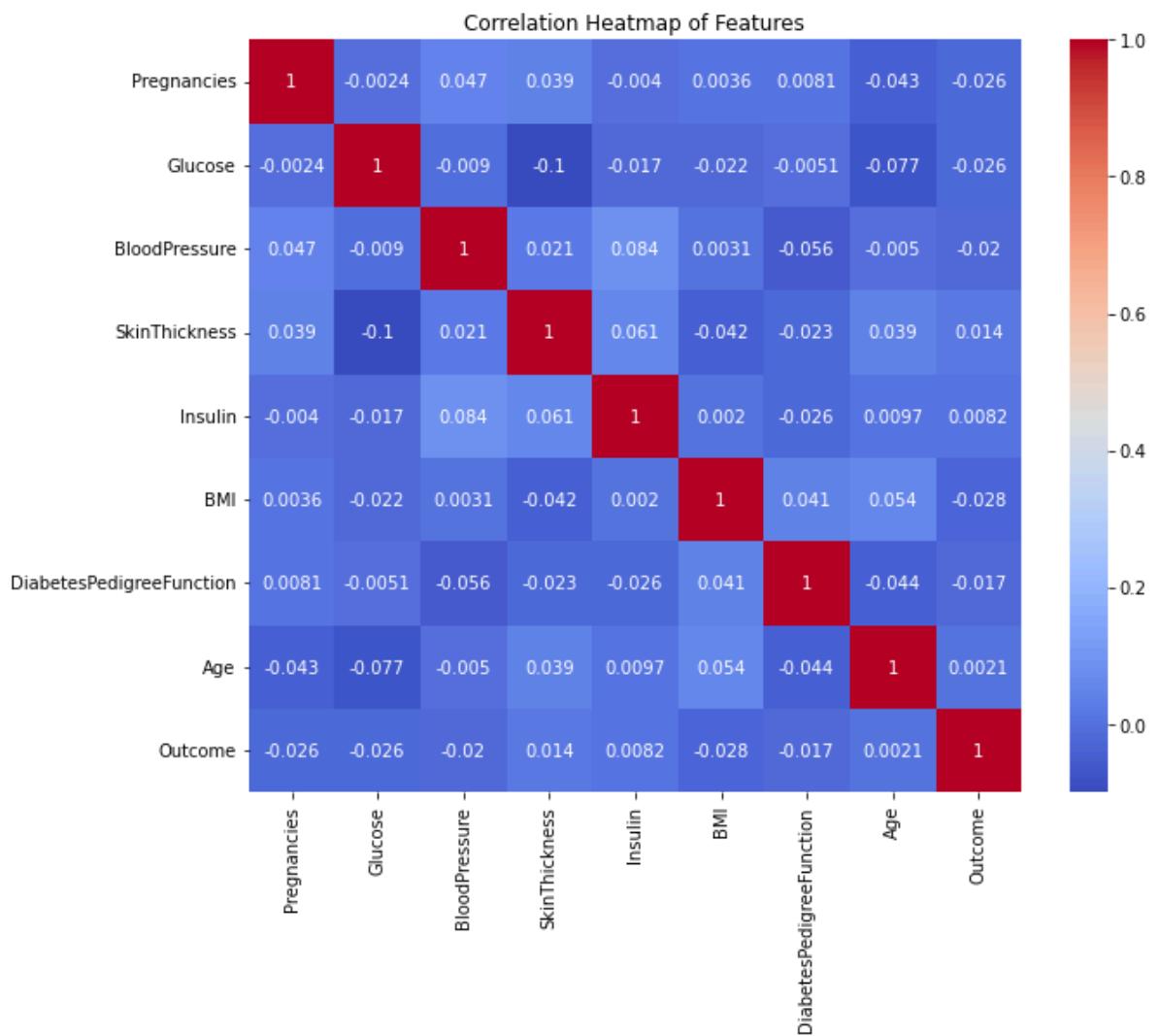
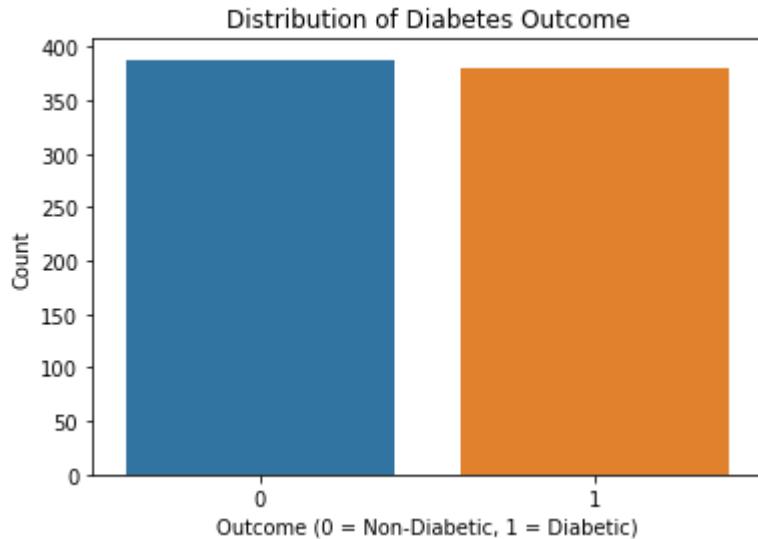
	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	\
count	768.000000	768.000000	768.000000	768.000000	768.000000	
mean	8.188802	126.669271	74.615885	48.457031	418.147135	
std	4.799992	44.242870	26.610859	28.087742	247.307610	
min	0.000000	50.000000	30.000000	0.000000	0.000000	
25%	4.000000	88.750000	52.000000	25.000000	203.750000	
50%	8.000000	128.000000	73.000000	46.000000	407.500000	
75%	12.000000	165.000000	98.000000	72.000000	631.000000	
max	16.000000	199.000000	121.000000	98.000000	845.000000	

	BMI	DiabetesPedigreeFunction	Age	Outcome
count	768.000000	768.000000	768.000000	768.000000
mean	41.405990	1.321742	50.723958	0.494792
std	14.939334	0.695430	17.338319	0.500299
min	15.000000	0.101000	21.000000	0.000000
25%	28.750000	0.722750	35.000000	0.000000
50%	41.900000	1.337500	52.000000	0.000000
75%	54.525000	1.915000	65.000000	1.000000
max	66.800000	2.498000	80.000000	1.000000

Missing Values in Each Column:

Pregnancies	0
Glucose	0
BloodPressure	0
SkinThickness	0
Insulin	0
BMI	0
DiabetesPedigreeFunction	0
Age	0

Outcome
dtype: int64



2) Write a Python program to perform Time Series Analysis on a website traffic dataset. The program should load and clean the dataset, decompose the time series to identify trends and seasonality, visualize key metrics using moving averages and seasonal plots, and detect anomalies using statistical methods. Dataset: A dataset containing daily website traffic data with the following columns:

- Date — Date of observation
- Page_VIEWS — Total number of page views per day
- Unique_Visitors — Number of unique visitors per day
- Bounce_Rate — Percentage of visitors who leave after viewing only one page (You can create a sample CSV file like website_traffic.csv for practice.)

Methodology:

1. Load and Clean Data: • Import the dataset and parse the Date column as a datetime object. • Handle missing values by interpolation.
2. Time Series Decomposition: • Decompose the time series data of Page_VIEWS to identify trend, seasonality, and residuals.
3. Visualization: • Plot the time series data for Page_VIEWS, Unique_Visitors, and Bounce_Rate. • Use moving averages to smooth the time series and highlight trends. • Create seasonal plots to visualize patterns over time.
4. Anomaly Detection: • Identify and visualize anomalies in the time series data using statistical methods such as the Z-score.

```
In [3]: import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from statsmodels.tsa.seasonal import seasonal_decompose
from scipy import stats
date_range = pd.date_range(start='2023-01-01', periods=180, freq='D')
data = {
    'Date': date_range,
    'Page_VIEWS': np.random.poisson(lam=500, size=180) + np.sin(np.arange(180)/10)*
    'Unique_Visitors': np.random.poisson(lam=300, size=180),
    'Bounce_Rate': np.clip(np.random.normal(loc=50, scale=10, size=180), 20, 100)
}
df = pd.DataFrame(data)
df.to_csv('website_traffic.csv', index=False)
df = pd.read_csv('website_traffic.csv', parse_dates=['Date'])
df = df.sort_values('Date')
df.set_index('Date', inplace=True)
df = df.interpolate()
print(df.head())
print(df.info())
decomposition = seasonal_decompose(df['Page_VIEWS'], model='additive', period=7)
plt.figure(figsize=(12, 8))
decomposition.plot()
plt.suptitle('Time Series Decomposition of Page Views', fontsize=14)
plt.show()
plt.figure(figsize=(14, 6))
plt.plot(df.index, df['Page_VIEWS'], label='Page Views')
plt.plot(df.index, df['Unique_Visitors'], label='Unique Visitors')
plt.plot(df.index, df['Bounce_Rate'], label='Bounce Rate')
plt.title('Website Traffic Metrics Over Time')
plt.xlabel('Date')
plt.ylabel('Values')
plt.legend()
plt.show()
df['PageViews_MA7'] = df['Page_VIEWS'].rolling(window=7).mean()
plt.figure(figsize=(14, 5))
plt.plot(df.index, df['Page_VIEWS'], label='Original')
plt.plot(df.index, df['PageViews_MA7'], label='7-Day Moving Average', linewidth=3)
plt.title('Page Views with Moving Average')
plt.legend()
plt.show()
df['DayOfWeek'] = df.index.day_name()
plt.figure(figsize=(10, 5))
sns.boxplot(x='DayOfWeek', y='Page_VIEWS', data=df.reset_index(), order=['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday'])
plt.title('Seasonal Pattern by Day of Week')
plt.xticks(rotation=45)
plt.show()

df['Z_Score'] = np.abs(stats.zscore(df['Page_VIEWS']))
```

```

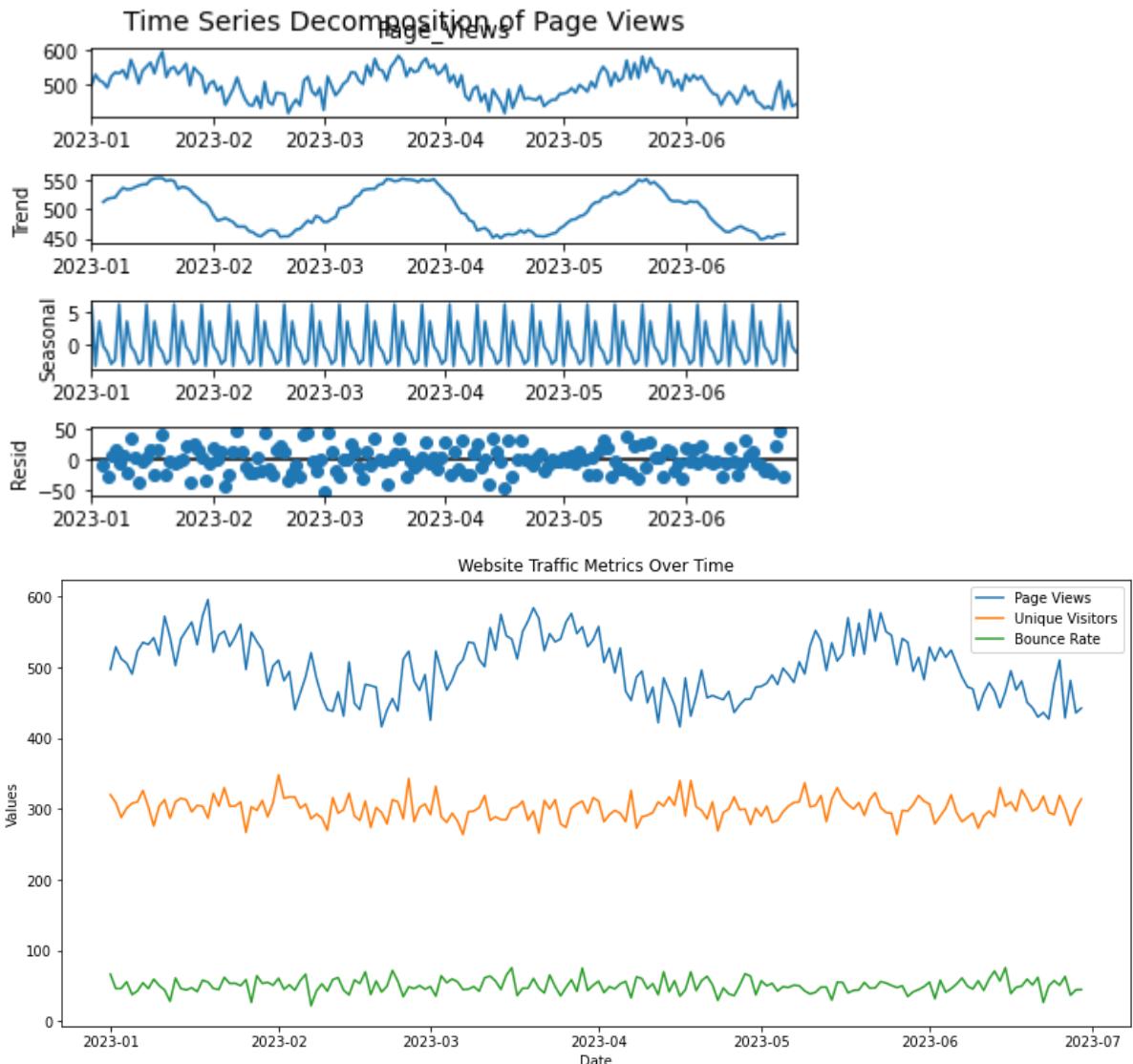
anomalies = df[df['Z_Score'] > 3]
plt.figure(figsize=(14, 5))
plt.plot(df.index, df['Page_VIEWS'], label='Page Views')
plt.scatter(anomalies.index, anomalies['Page_VIEWS'], color='red', label='Anomalies')
plt.title('Anomaly Detection in Page Views')
plt.legend()
plt.show()

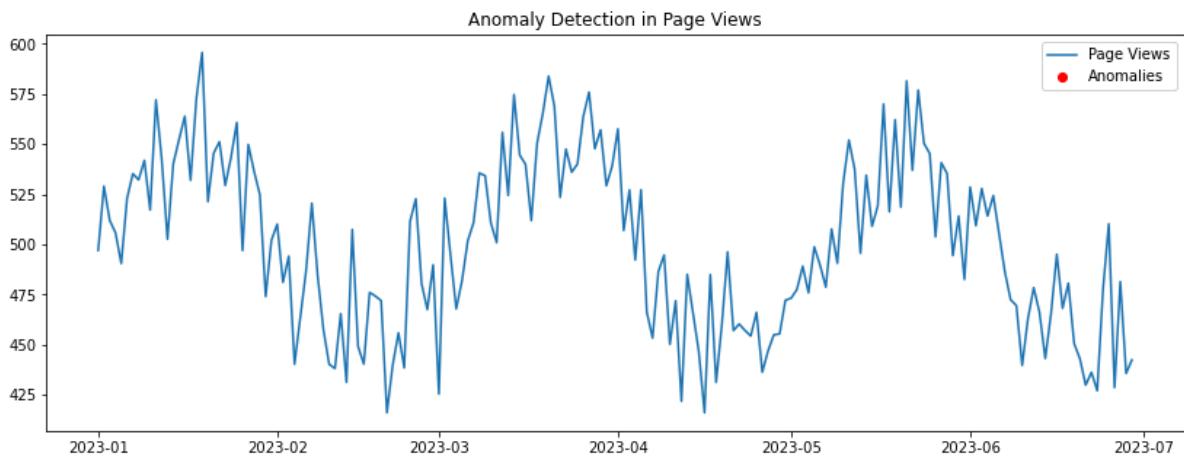
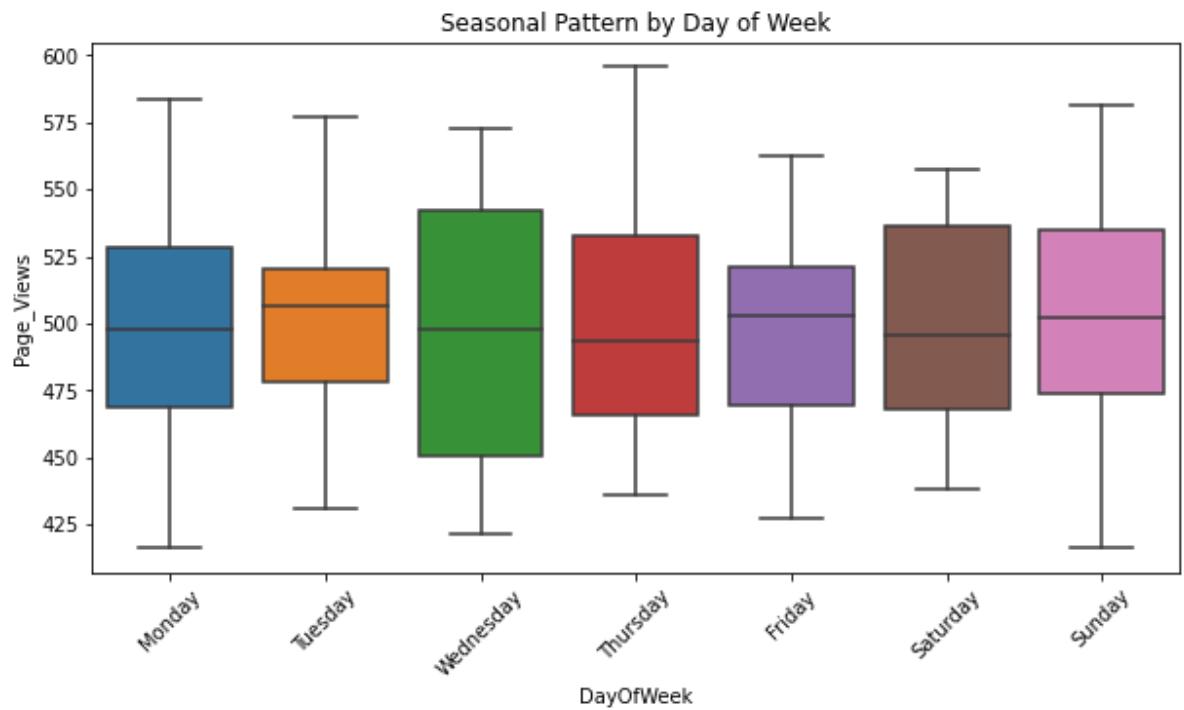
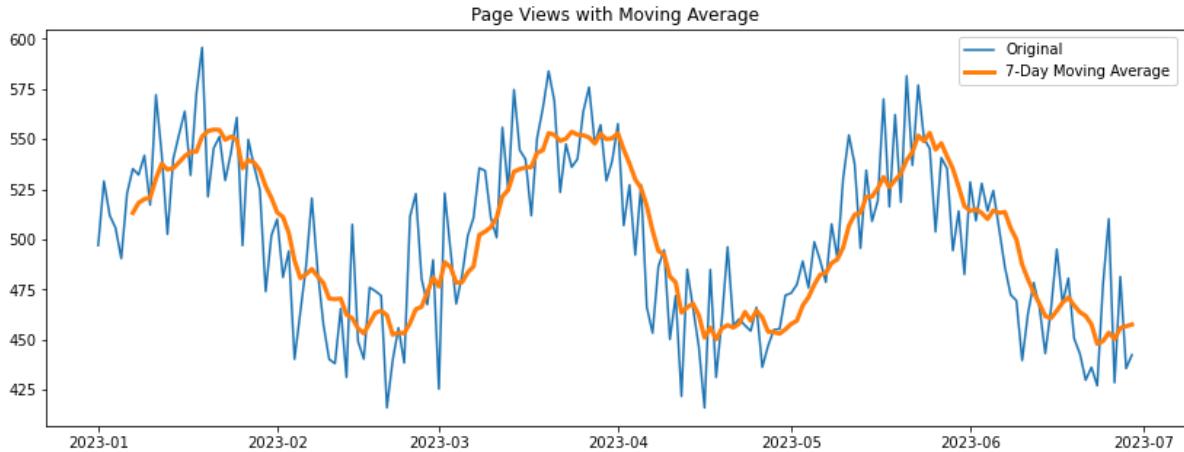
print(anomalies[['Page_VIEWS', 'Z_Score']])

```

	Page_VIEWS	Unique_Visitors	Bounce_Rate
Date			
2023-01-01	497.000000	320	66.606963
2023-01-02	528.991671	309	46.365357
2023-01-03	511.933467	288	46.485806
2023-01-04	505.776010	301	55.963921
2023-01-05	490.470917	308	37.869061

<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 180 entries, 2023-01-01 to 2023-06-29
Data columns (total 3 columns):
Column Non-Null Count Dtype
--- --
0 Page_VIEWS 180 non-null float64
1 Unique_Visitors 180 non-null int64
2 Bounce_Rate 180 non-null float64
dtypes: float64(2), int64(1)
memory usage: 5.6 KB
None
<Figure size 864x576 with 0 Axes>





Empty DataFrame
Columns: [Page_VIEWS, Z_Score]
Index: []

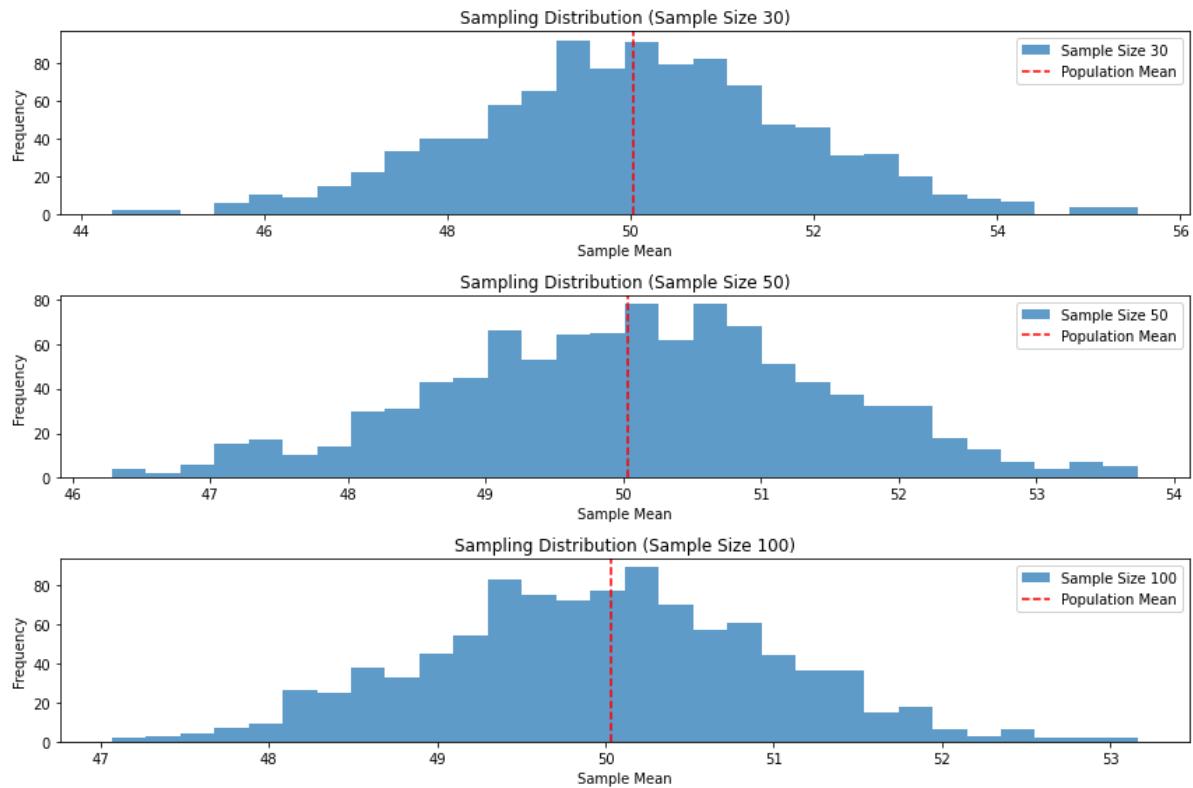
3) Random Sampling and Sampling Distribution To explore random sampling from a population and understand the concept of sampling distribution using Python in Jupyter Notebook. Steps:

1. Generate a Population: • Create a population of data with a specified distribution (e.g., normal distribution).

2. Random Sampling: • Perform random sampling from the population to create multiple samples of different sizes. • Compute sample statistics (mean, standard deviation, etc.) for each sample.
3. Sampling Distribution: • Plot histograms or density plots of sample statistics (e.g., sample means). • Compare the sampling distribution of the sample statistic (mean) with the population distribution.
4. Central Limit Theorem (Optional): • Demonstrate the Central Limit Theorem by showing that as sample size increases, the sampling distribution of the sample mean approaches a normal distribution regardless of the population distribution.

```
In [4]: import numpy as np
import matplotlib.pyplot as plt
population_mean = 50
population_std = 10
population_size = 100000
population = np.random.normal(population_mean, population_std, population_size)
sample_sizes = [30, 50, 100]
num_samples = 1000
sample_means = {}
for size in sample_sizes:
    sample_means[size] = []
    for _ in range(num_samples):
        sample = np.random.choice(population, size=size, replace=False)
        sample_means[size].append(np.mean(sample))

plt.figure(figsize=(12, 8))
for i, size in enumerate(sample_sizes):
    plt.subplot(len(sample_sizes), 1, i+1)
    plt.hist(sample_means[size], bins=30, alpha=0.7, label=f'Sample Size {size}')
    plt.axvline(np.mean(population), color='red', linestyle='dashed', linewidth=1.5)
    plt.title(f'Sampling Distribution (Sample Size {size})')
    plt.xlabel('Sample Mean')
    plt.ylabel('Frequency')
    plt.legend()
plt.tight_layout()
plt.show()
```



WEEK 8

1. Conduct Z test for the Data Given using Python Objective: To test whether the average weight of a species of birds differs from 150 grams. Procedure:
2. Null Hypothesis The average weight of the birds is 150 grams.
3. Alternative Hypothesis The average weight of the birds is not 150 grams.
4. Sample: Measure the weights of 30 birds randomly selected from the population.
5. Z-Test: Conduct a Z-test to compare the sample mean to 150 grams.
6. Decision Rule: Use a significance level of = 0.05.

```
In [2]: import numpy as np
import scipy.stats as stats
sample_data = np.array([152, 148, 151, 149, 147, 153, 150, 148, 152, 149, 151, 150,
148, 150, 152, 149, 150, 148, 153, 151, 150, 149, 152, 148,
population_mean = 150
sample_mean = np.mean(sample_data)
sample_std = np.std(sample_data, ddof=1)
n = len(sample_data)
z_statistic = (sample_mean - population_mean) / (sample_std / np.sqrt(n))
p_value = 2 * (1 - stats.norm.cdf(np.abs(z_statistic)))
alpha = 0.05

print(f"Sample Mean: {sample_mean:.2f}")
print(f"Z-Statistic: {z_statistic:.4f}")
print(f"P-Value: {p_value:.4f}")
if p_value < alpha:
    print("Reject the null hypothesis: The average weight is significantly different")
else:
    print("Fail to reject the null hypothesis: There is no significant difference in average weight from 150 grams.")

Sample Mean: 150.20
Z-Statistic: 0.6406
P-Value: 0.5218
Fail to reject the null hypothesis: There is no significant difference in average weight from 150 grams.
```

1. Experiment to understand Matplotlib library use cases in Data Science through visualization. Description: Use Iris data set to understand the Matplotlib library

```
In [3]: import pandas as pd
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
iris = load_iris()
df = pd.DataFrame(iris.data, columns=iris.feature_names)
df['species'] = pd.Categorical.from_codes(iris.target, iris.target_names)

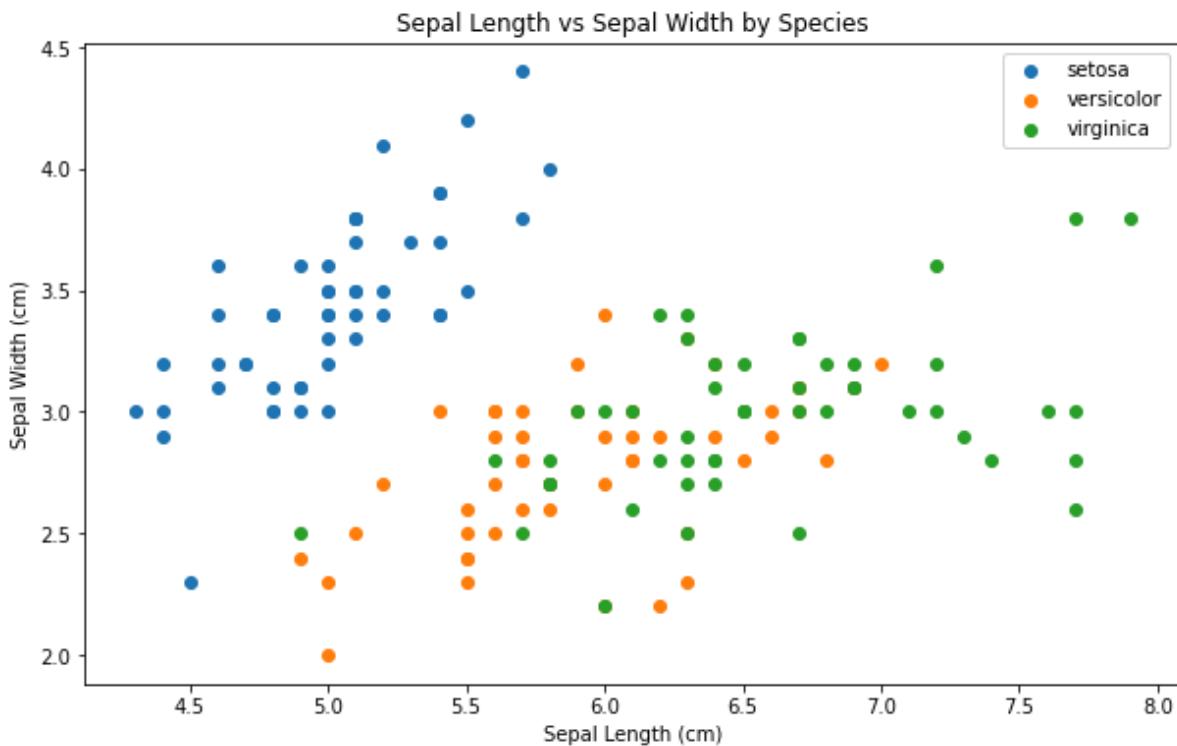
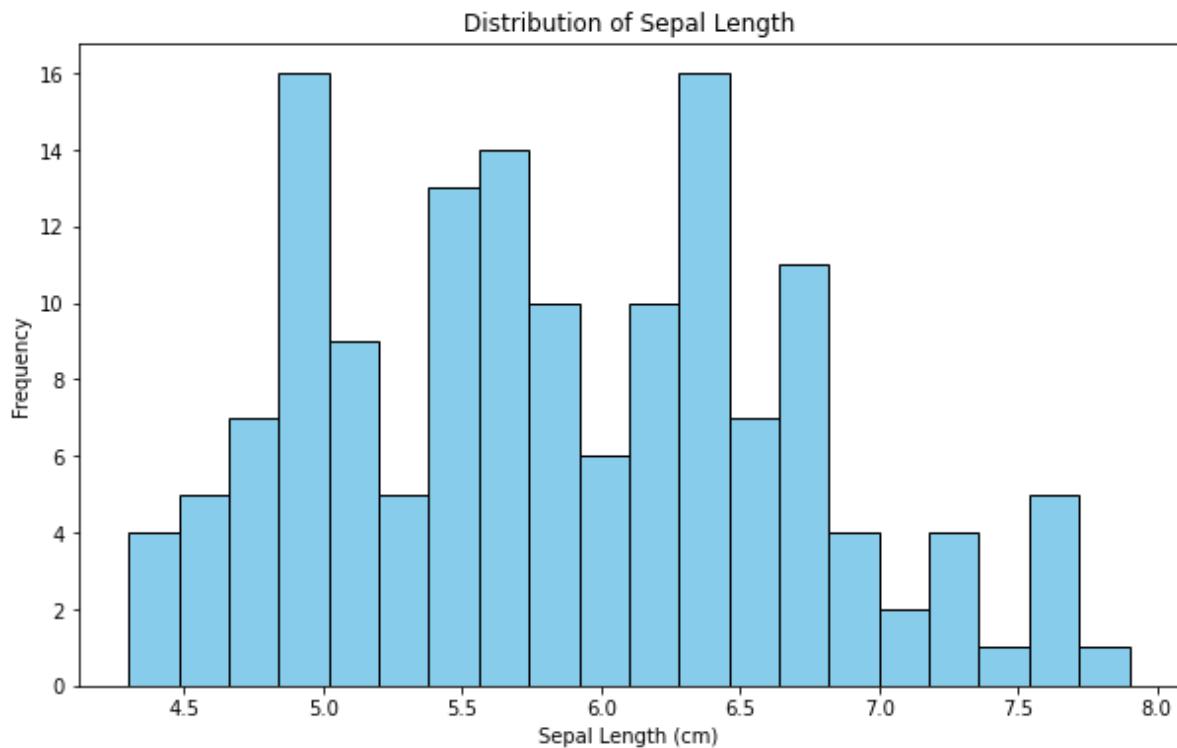
plt.figure(figsize=(10, 6))
plt.hist(df['sepal length (cm)'], bins=20, color='skyblue', edgecolor='black')
plt.title('Distribution of Sepal Length')
plt.xlabel('Sepal Length (cm)')
plt.ylabel('Frequency')
plt.show()

plt.figure(figsize=(10, 6))
for species in df['species'].unique():
    subset = df[df['species'] == species]
    plt.scatter(subset['sepal length (cm)'], subset['sepal width (cm)'], label=species)
plt.title('Sepal Length vs Sepal Width by Species')
```

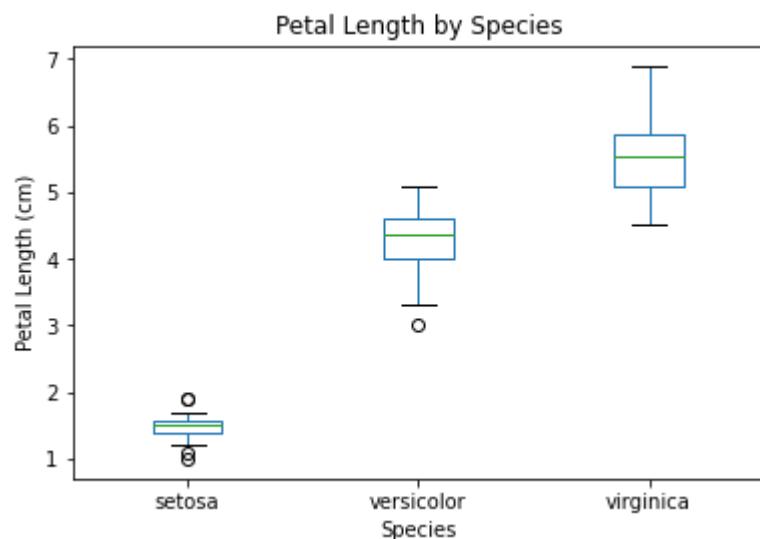
```

plt.xlabel('Sepal Length (cm)')
plt.ylabel('Sepal Width (cm)')
plt.legend()
plt.show()
plt.figure(figsize=(10, 6))
df.boxplot(column='petal length (cm)', by='species', grid=False)
plt.title('Petal Length by Species')
plt.suptitle('')
plt.xlabel('Species')
plt.ylabel('Petal Length (cm)')
plt.show()

```



<Figure size 720x432 with 0 Axes>



You are provided monthly sales data for four regions (North, South, East, West) for January–March. Use Python (pandas, seaborn, matplotlib) or a spreadsheet to analyze and visualize the data, then interpret results and recommend business actions.

	January	February	March
North	1500	3100	1600
South	900	0	2000
East	1100	1500	0
West	1300	0	0

1. Create a pandas DataFrame named df using the above data. Set the Region column as the index of the DataFrame. Display the DataFrame.
2. Use the seaborn library to create a heat map that shows sales values by region and month.

Requirements:

- Display numeric values inside each cell (annot=True).
- Use a color map (for example 'YlGnBu').
- Add a suitable figure size and title (Sales Heat Map by Region and Month). Display the heat map using plt.show().

1. After generating the heat map:

Observe which region and month show the highest and lowest sales. Print the following results in the output: Region with highest total sales. Region with lowest total sales. Month with highest total sales. Month with lowest total sales.

```
In [1]: import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

data = {
    'Region': ['North', 'South', 'East', 'West'],
    'January': [1500, 900, 1100, 1300],
    'February': [3100, 0, 1500, 0],
    'March': [1600, 2000, 0, 0]
}
df = pd.DataFrame(data)
df.set_index('Region', inplace=True)
print("Sales DataFrame:")
print(df)

plt.figure(figsize=(8, 6))
sns.heatmap(df, annot=True, cmap='YlGnBu', fmt='g')
plt.title("Sales Heat Map by Region and Month")
plt.xlabel("Month")
plt.ylabel("Region")
plt.show()

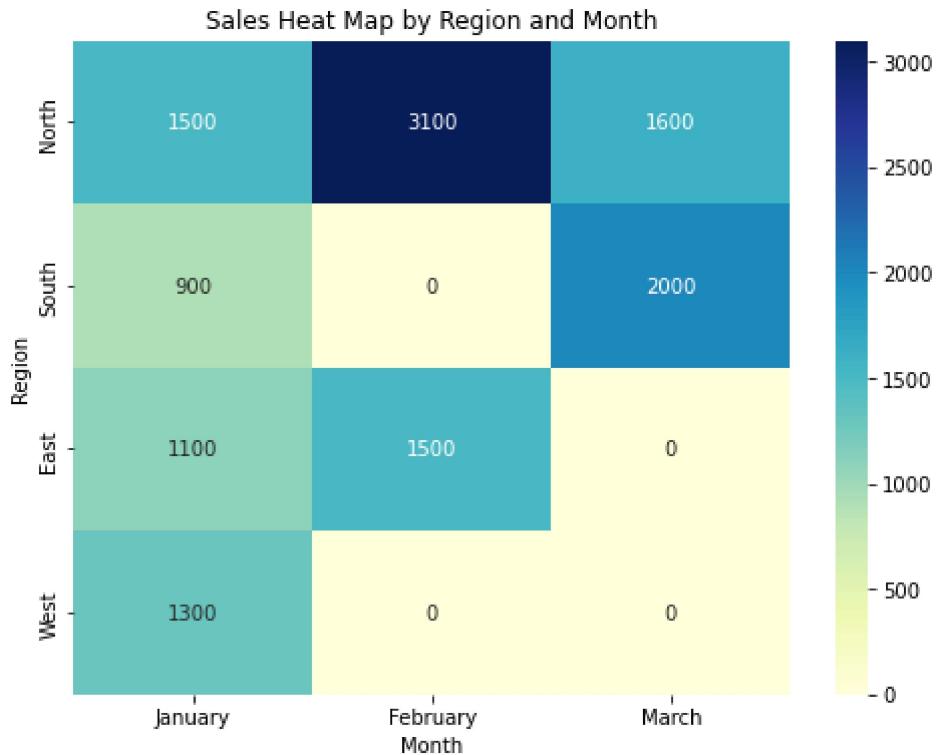
total_sales_by_region = df.sum(axis=1)
total_sales_by_month = df.sum(axis=0)

region_highest_sales = total_sales_by_region.idxmax()
region_lowest_sales = total_sales_by_region.idxmin()
month_highest_sales = total_sales_by_month.idxmax()
month_lowest_sales = total_sales_by_month.idxmin()

print(f"Region with highest total sales: {region_highest_sales}")
print(f"Region with lowest total sales: {region_lowest_sales}")
print(f"Month with highest total sales: {month_highest_sales}")
print(f"Month with lowest total sales: {month_lowest_sales}")
```

Sales DataFrame:

	January	February	March
Region			
North	1500	3100	1600
South	900	0	2000
East	1100	1500	0
West	1300	0	0



Region with highest total sales: North

Region with lowest total sales: West

Month with highest total sales: January

Month with lowest total sales: March

2) You are given the average test scores of students in four subjects across four classes.

Class Math Science English Computer A 78 85 88 92 B 65 70 75 80 C 90 88 84 89 D 55 60 65

70 1 Create a DataFrame using pandas and set Class as the index. 2 Plot a heat map using

Seaborn to visualize the performance of each class across subjects. • Use `annot=True`,

`fmt='0f'`, and a colormap like 'coolwarm'. 3 Calculate and print: • The subject with the

highest overall average score. • The class with the highest total score. 4 Display the heat map

and print the results.

```
In [2]: import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

data = {
    'Class': ['A', 'B', 'C', 'D'],
    'Math': [78, 65, 90, 55],
    'Science': [85, 70, 88, 60],
    'English': [88, 75, 84, 65],
    'Computer': [92, 80, 89, 70]
}
df = pd.DataFrame(data)
df.set_index('Class', inplace=True)

plt.figure(figsize=(8, 6))
```

```

sns.heatmap(df, annot=True, fmt='0f', cmap='coolwarm')
plt.title('Performance of Classes Across Subjects')
plt.xlabel('Subjects')
plt.ylabel('Class')

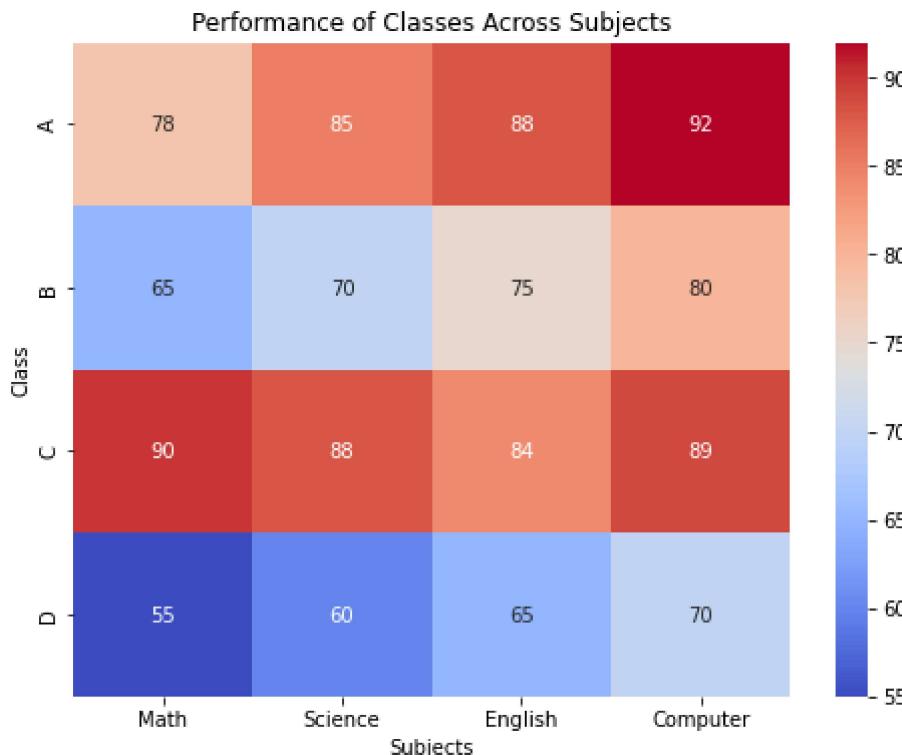
subject_avg = df.mean(axis=0)
subject_highest_avg = subject_avg.idxmax()

class_total = df.sum(axis=1)
class_highest_total = class_total.idxmax()

plt.show()

print(f"Subject with highest overall average score: {subject_highest_avg}")
print(f"Class with highest total score: {class_highest_total}")

```



Subject with highest overall average score: Computer
 Class with highest total score: C

A hospital recorded the number of patients treated in different departments over three months.

Department January February March Cardiology 120 150 130 Neurology 100 90 110
 Orthopedics 140 160 170 Pediatrics 80 100 95

1 Create a pandas DataFrame from the above data.
 2 Set Department as the index and display it.
 3 Create a heat map with:

- annot=True, fmt='d', cmap='Greens'

 4 Calculate and print:

- Department with the highest total patients.
- Month with the lowest total patients.

 5 Add a suitable title: "Hospital Patient Count Heat Map" and display it.

In [3]:

```

import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

```

```

data = {
    'Department': ['Cardiology', 'Neurology', 'Orthopedics', 'Pediatrics'],
    'January': [120, 100, 140, 80],
    'February': [150, 90, 160, 100],
    'March': [130, 110, 170, 95]
}
df = pd.DataFrame(data)
df.set_index('Department', inplace=True)
print("Hospital Patient Count DataFrame:")
print(df)

plt.figure(figsize=(8,6))
sns.heatmap(df, annot=True, fmt='d', cmap='Greens')
plt.title("Hospital Patient Count Heat Map")
plt.xlabel("Month")
plt.ylabel("Department")

total_by_department = df.sum(axis=1)
total_by_month = df.sum(axis=0)

department_highest = total_by_department.idxmax()
month_lowest = total_by_month.idxmin()

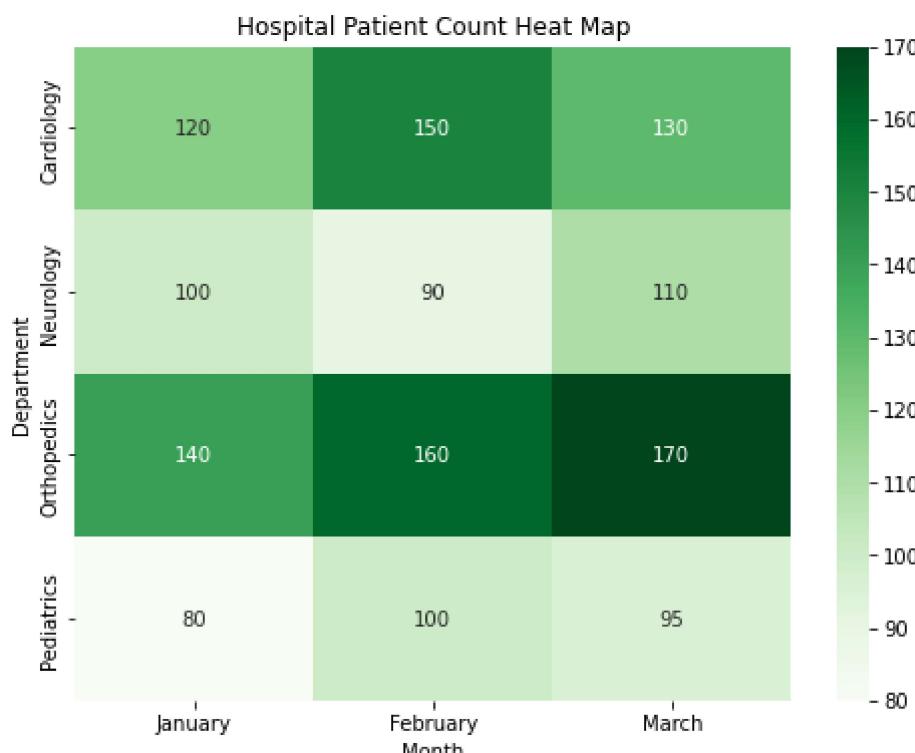
plt.show()

print(f"Department with the highest total patients: {department_highest}")
print(f"Month with the lowest total patients: {month_lowest}")

```

Hospital Patient Count DataFrame:

	January	February	March
Department			
Cardiology	120	150	130
Neurology	100	90	110
Orthopedics	140	160	170
Pediatrics	80	100	95



Department with the highest total patients: Orthopedics
Month with the lowest total patients: January