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 : Gayathri Jayam S

Register Number : 240701142

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
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 • 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
```

1. Given a DataFrame with missing values: data = { 'Student': ['John', 'Emma', 'Sam', 'Olivia'], 'Marks': [80, None, 75, 90] } Write code to: • a) Fill missing marks with 0 • b) Drop rows with missing values • c) 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
```

1. For any DataFrame: • a) Save it as a CSV file named students.csv • b) Load a CSV file named employees.csv • c) Set Name as the index • d) 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, '
    {"Name": "Joe Root", "Age": 33, "Country": "England", "Matches_Played": 215, "R
    {"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": 130
    {"Name": "Pat Cummins", "Age": 32, "Country": "Australia", "Matches_Played": 14
    {"Name": "Rashid Khan", "Age": 26, "Country": "Afghanistan", "Matches_Played":
    {"Name": "Trent Boult", "Age": 35, "Country": "New Zealand", "Matches_Played":
    {"Name": "Mohammed Shami", "Age": 34, "Country": "India", "Matches_Played": 150
    {"Name": "KL Rahul", "Age": 32, "Country": "India", "Matches_Played": 140, "Run
    {"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":
    {"Name": "Shaheen Afridi", "Age": 25, "Country": "Pakistan", "Matches_Played":
    {"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":
]
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
    },
    {
        "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",
        "Price": 190,
        "Ingredients": ["Mango", "Yogurt", "Sugar", "Ice"],
        "Vegetarian": "Yes",
        "Preparation_Time": 10
    },
    {
        "Dish_ID": "DISH026",
        "Dish_Name": "Gulab Jamun",
        "Category": "Dessert",
        "Price": 160,
        "Ingredients": ["Milk solids", "Sugar syrup", "Cardamom", "Rose water"],
        "Vegetarian": "Yes",
        "Preparation_Time": 20
    },
    {
        "Dish_ID": "DISH027",
        "Dish_Name": "Chicken Butter Masala",
        "Category": "Main Course",
        "Price": 420,
        "Ingredients": ["Chicken", "Butter", "Tomato", "Cream", "Spices"],
        "Vegetarian": "No",
        "Preparation_Time": 35
    },
    {
        "Dish_ID": "DISH028",
        "Dish_Name": "Sizzling Brownie",
        "Category": "Dessert",
        "Price": 260,
        "Ingredients": ["Chocolate", "Flour", "Butter", "Ice cream", "Syrup"],
        "Vegetarian": "Yes",
        "Preparation_Time": 20
    },
    {
        "Dish_ID": "DISH029",
        "Dish_Name": "Laddu",
        "Category": "Dessert",
        "Price": 140,
        "Ingredients": ["Gram flour", "Sugar", "Ghee", "Cardamom"],
        "Vegetarian": "Yes",
        "Preparation_Time": 15
    },
    {
        "Dish_ID": "DISH030",
        "Dish_Name": "Rasmalai",
        "Category": "Dessert",
        "Price": 180,
        "Ingredients": ["Milk", "Sugar", "Cardamom", "Saffron", "Nuts"],
        "Vegetarian": "Yes",
        "Preparation_Time": 25
    },
    {
```

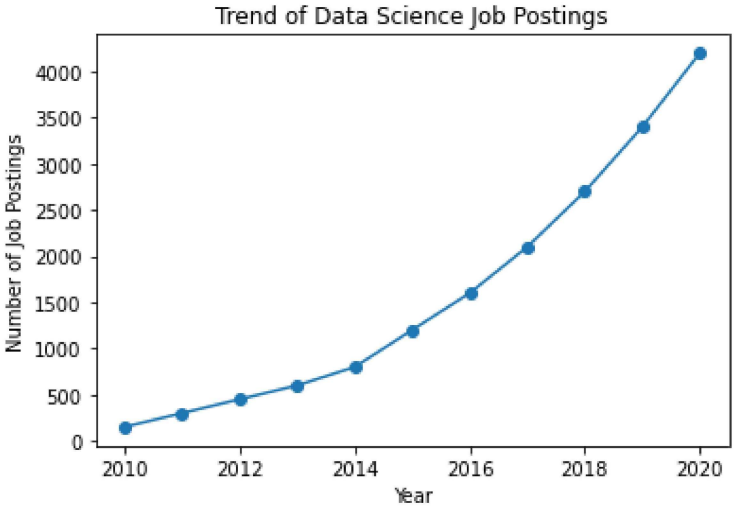
```
        "Dish_ID": "DISH031",
        "Dish_Name": "Pani Puri",
        "Category": "Starter",
        "Price": 120,
        "Ingredients": ["Semolina", "Potato", "Tamarind", "Mint", "Spices"],
        "Vegetarian": "Yes",
        "Preparation_Time": 15
    },
    {
        "Dish_ID": "DISH032",
        "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()
```

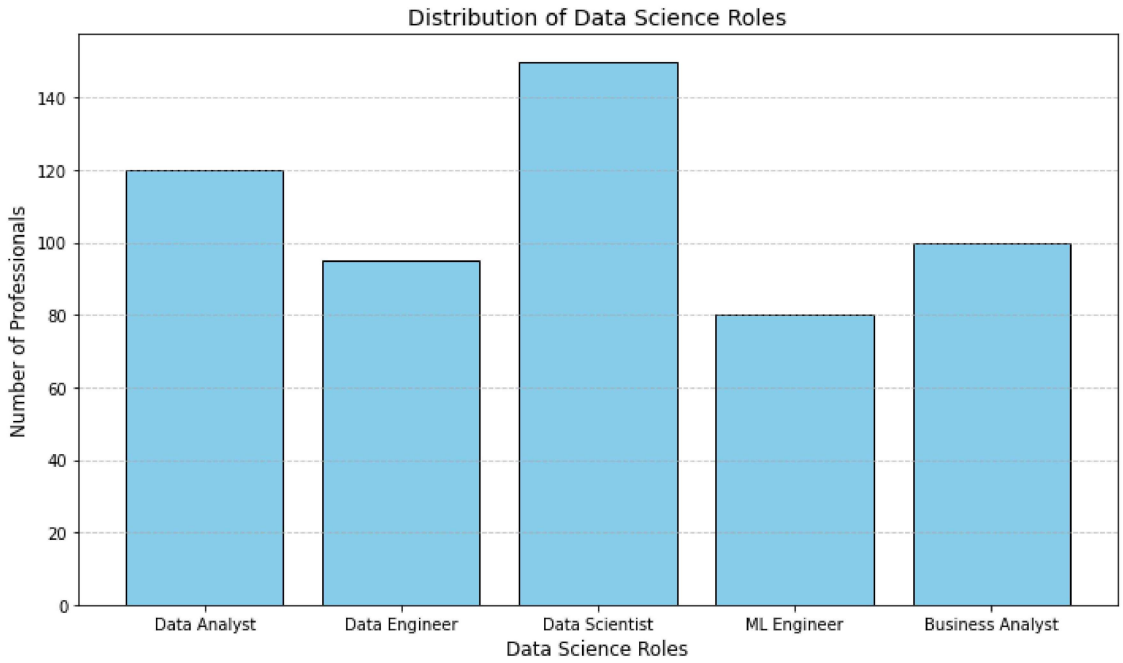


1. "Write a Python program using Matplotlib to create a bar chart that shows the distribution of different Data Science roles (Data Analyst, Data Engineer, Data Scientist, ML Engineer, and

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', 'Busines
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'], 'experie
nce': 5}, 'employee2': {'name': 'Bob', 'skills': ['C++', 'Cloud Computing']}, 'emp
loyee3': {'name': 'Charlie', 'department': 'Marketing'}}
```

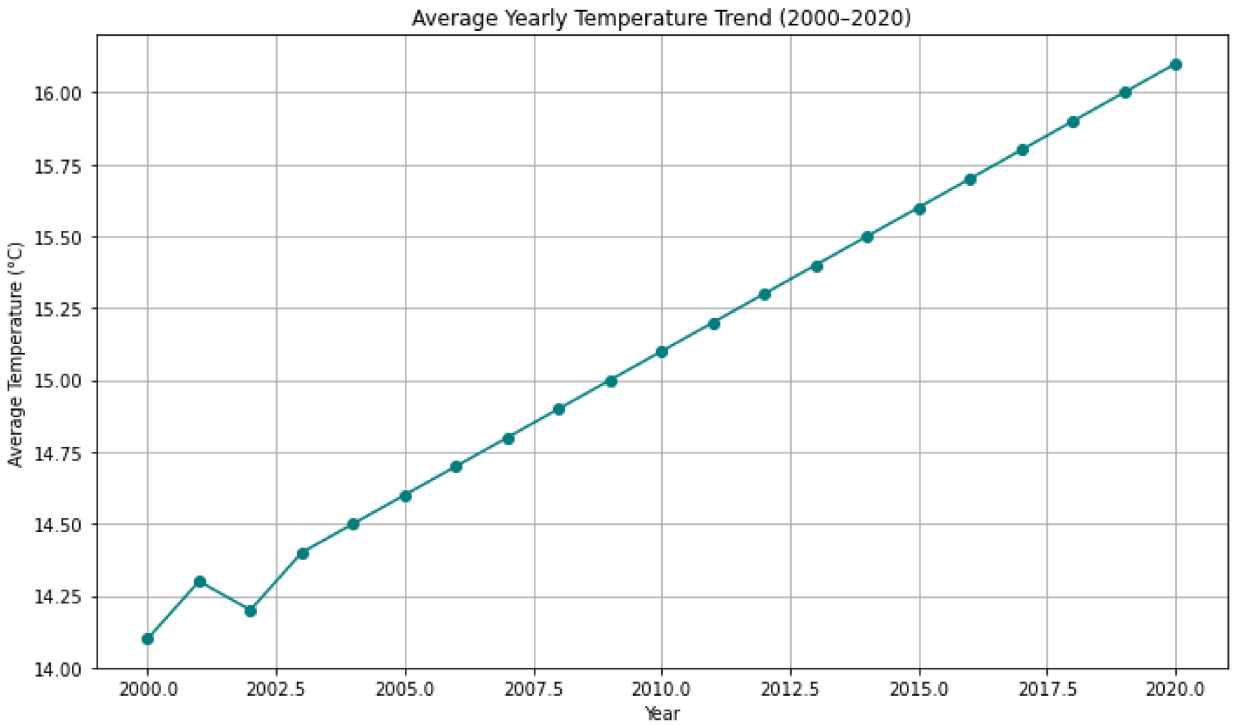
4 "Write a Python program using the cryptography.fernet module to demonstrate symmetric key encryption and decryption. Encrypt the text 'Rajalakshmi Engineering College' using a generated key, display the encrypted ciphertext, and then decrypt it back to the original text. Print the original, encrypted, 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_uTN5OLDSO_lqERPosR6Ua2VJBFLKtADCHpngDAxo8osarRd3di
Mhd7lnSgq4J0DNPvt5LG0mlMbme5fmsFn08vxMqcIw2l3kLJHUO-IMRhdBUqC3WEDJRdsR_oHjW1'
Decrypted Data: b'Rajalakshmi Engineering College.'
```

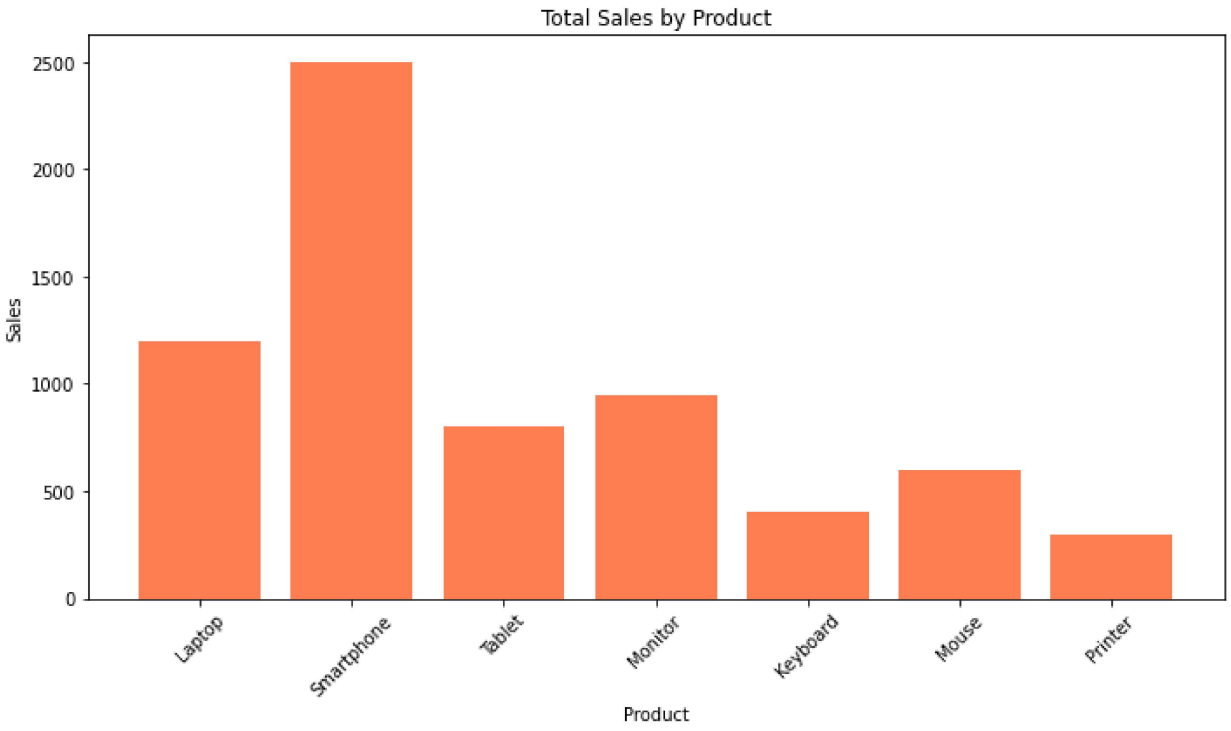
Q1. Line Plot from CSV (Temperature Trends) Download or create a CSV file weather.csv with the following columns: Year, AvgTemperature. Write a Python program using Pandas and Matplotlib to plot a line chart showing the trend of average yearly temperature from 2000 to 2020. • Add markers on the line.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

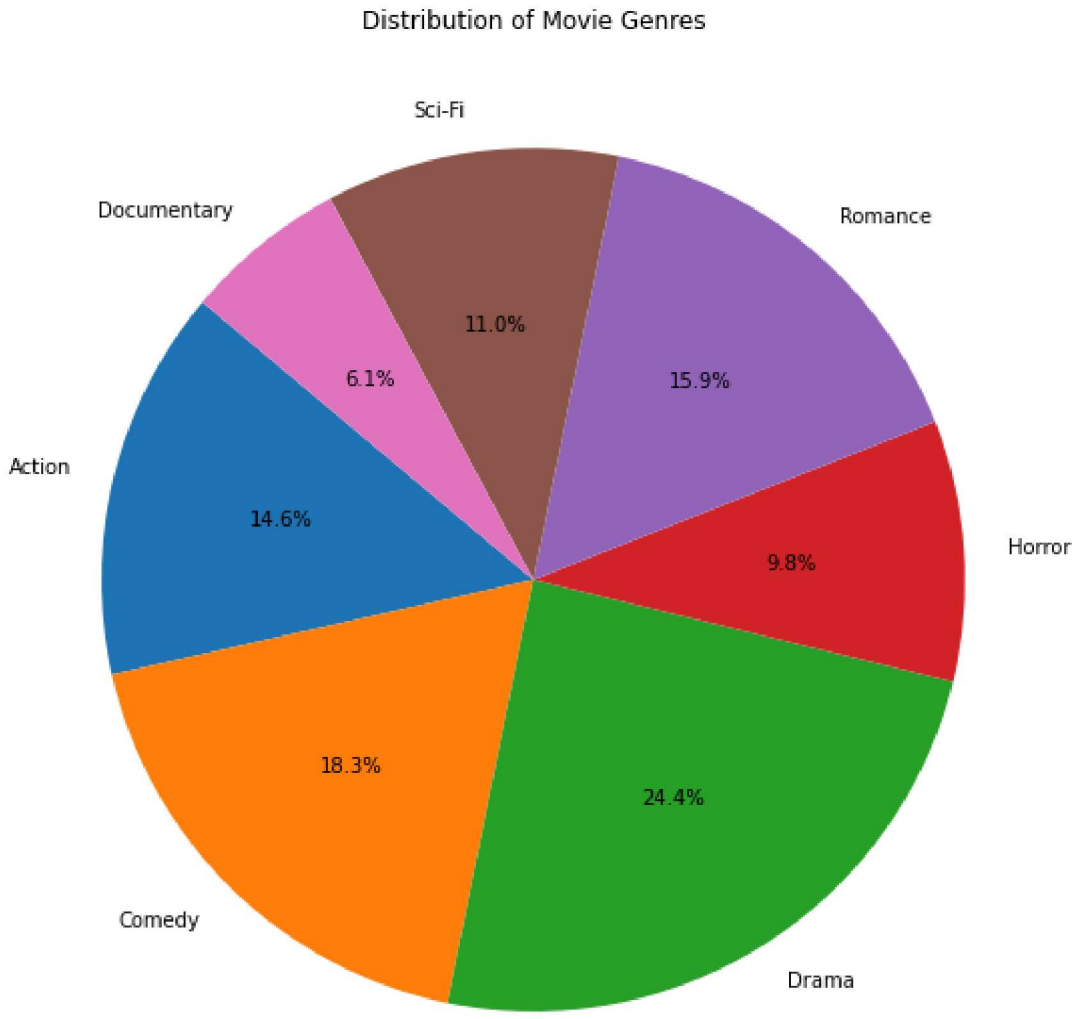
In [2]:
```
import pandas as pd
import matplotlib.pyplot as plt

data = {
    'Product': ['Laptop', 'Smartphone', 'Tablet', 'Monitor', 'Keyboard', 'Mouse', 'Pri
    '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
```

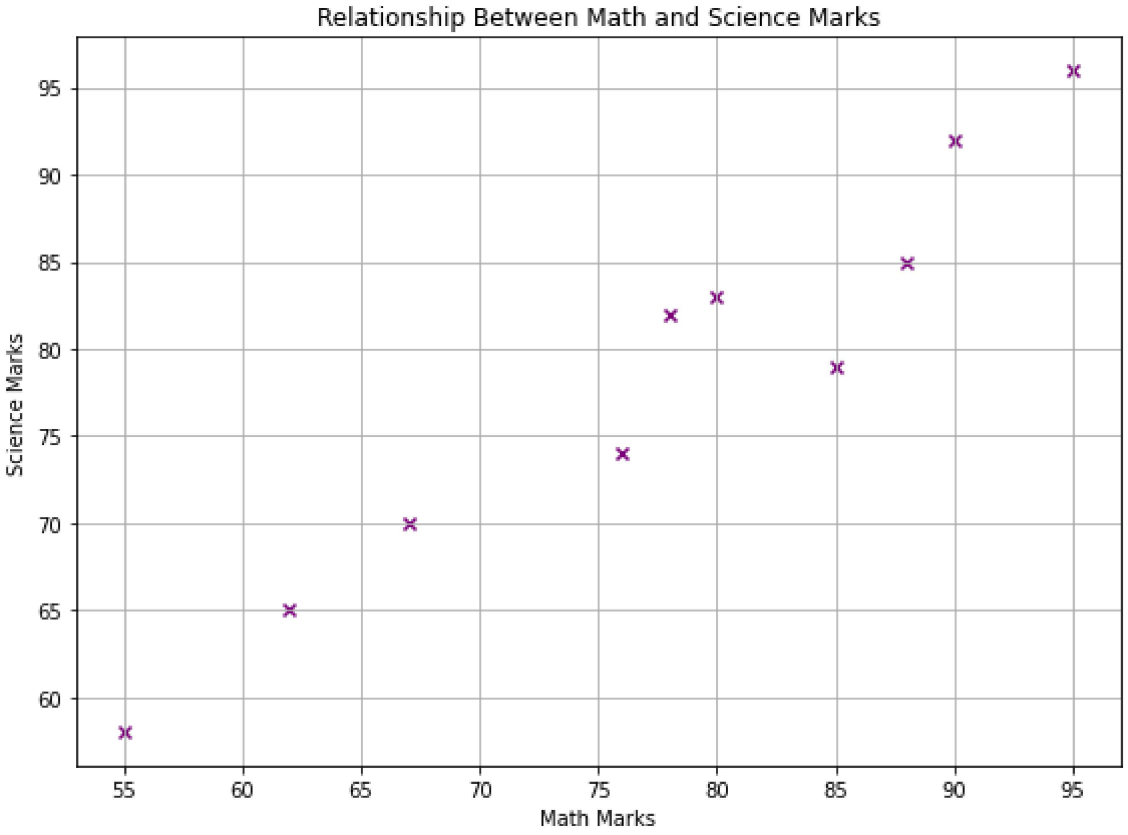
```
data = {
    'Genre': ['Action', 'Comedy', 'Drama', 'Horror', 'Romance', 'Sci-Fi', 'Documentary
    'Count': [120, 150, 200, 80, 130, 90, 50]
}

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()
```



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
```

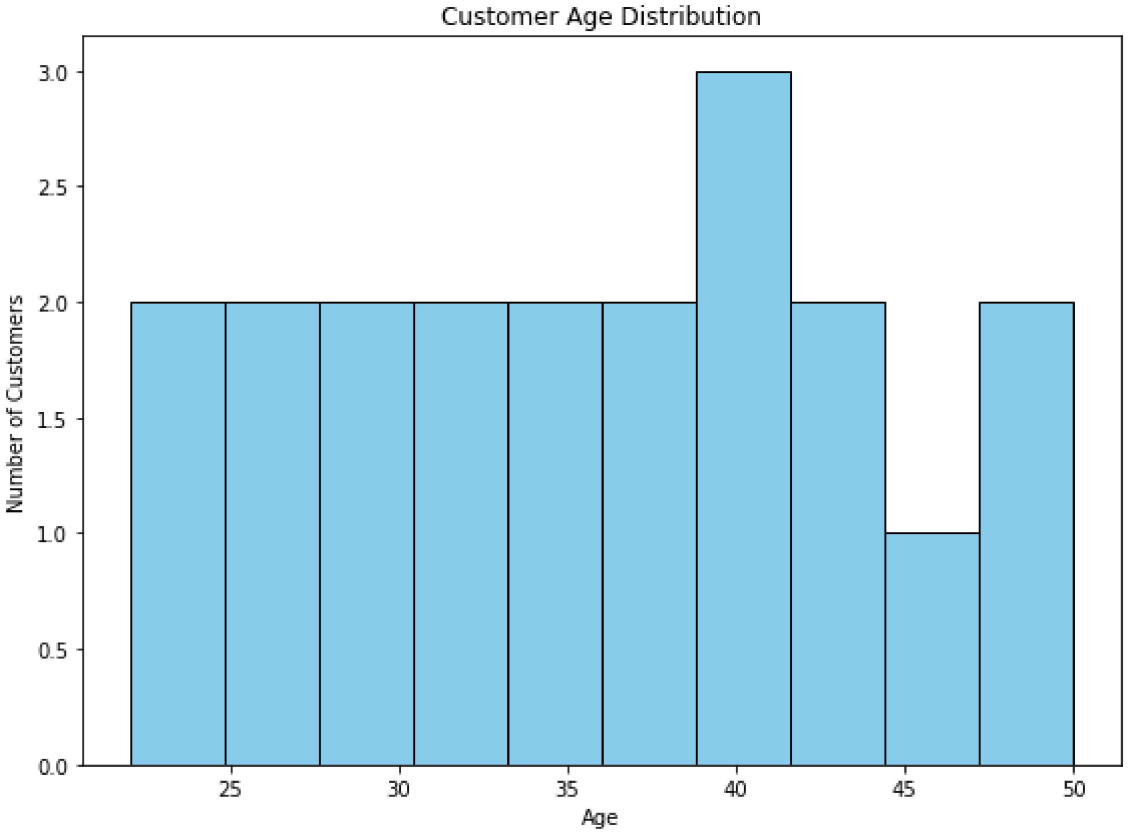
```
data = {
    'MathMarks': [78, 85, 62, 90, 55, 88, 76, 95, 67, 80],
    'ScienceMarks': [82, 79, 65, 92, 58, 85, 74, 96, 70, 83]
}

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, 30
}

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

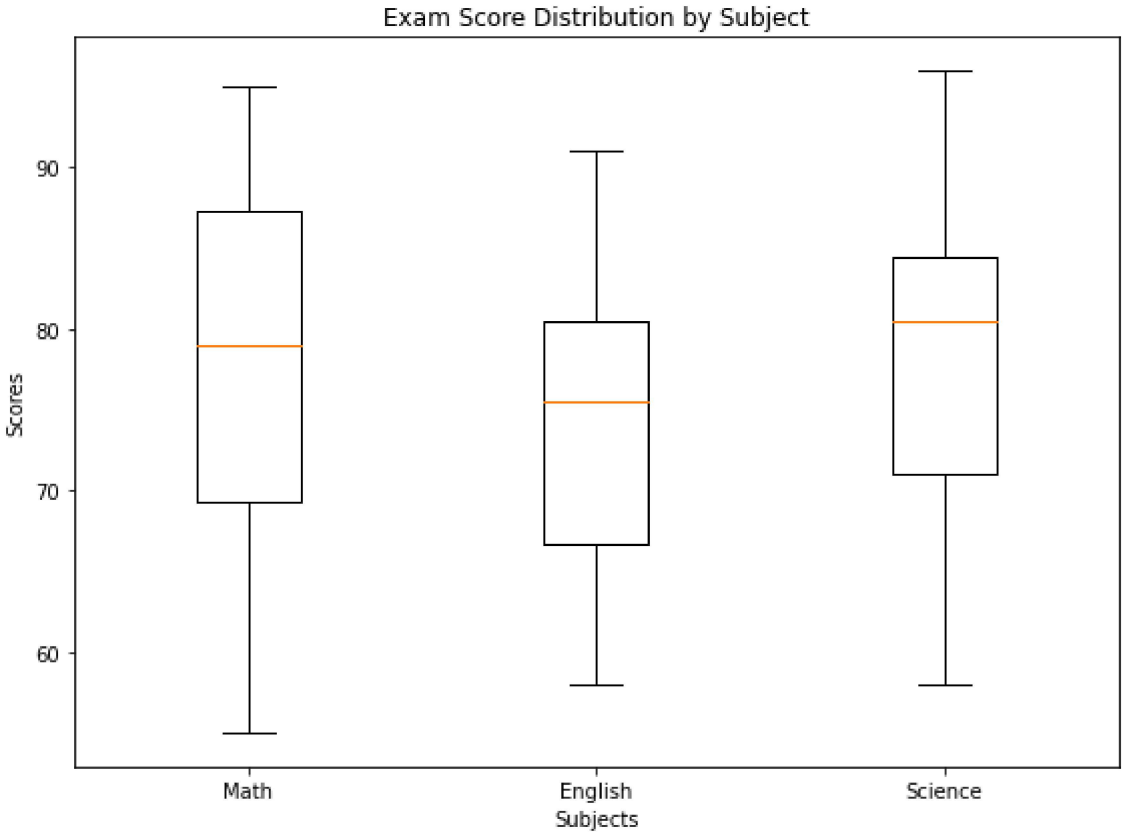
```
df = pd.read_csv('customers.csv')
plt.figure(figsize=(8, 6))
plt.hist(df['Age'], bins=10, color='skyblue', edgecolor='black')
plt.xlabel('Age')
plt.ylabel('Number of Customers')
plt.title('Customer Age Distribution')
plt.tight_layout()
plt.show()
```



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.

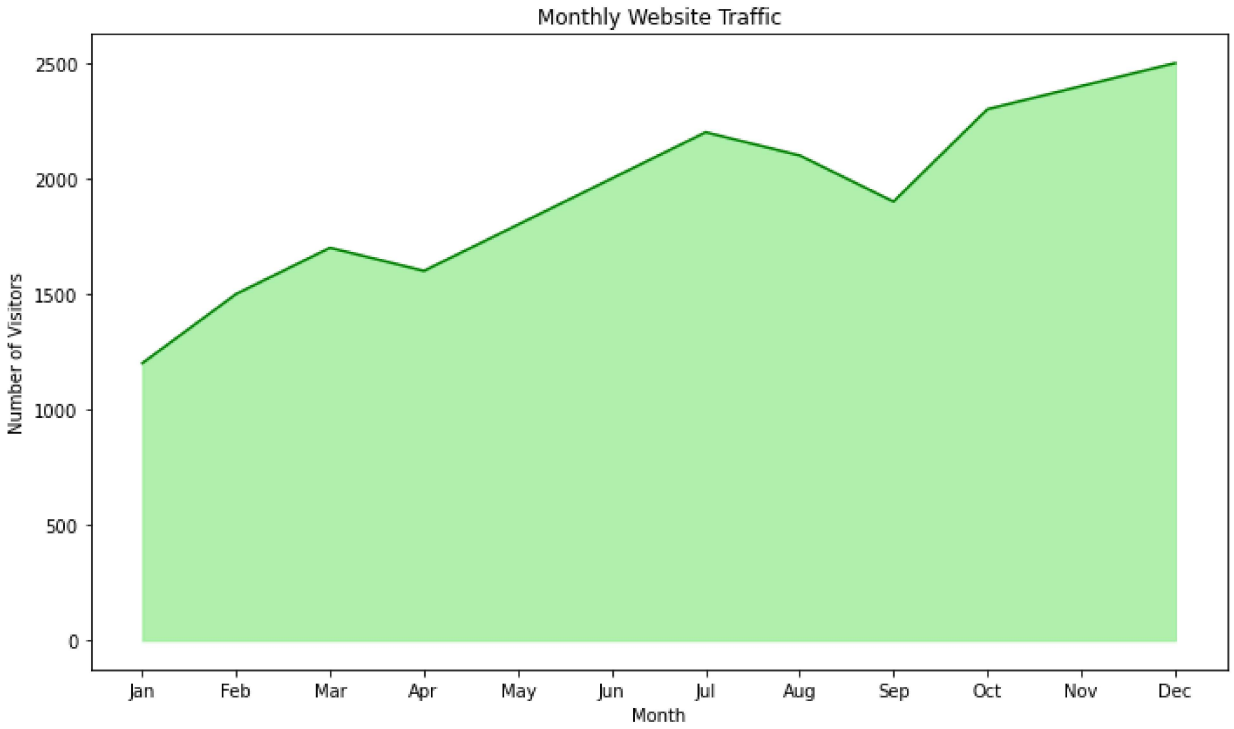
In [6]:
```
import pandas as pd
import matplotlib.pyplot as plt

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', 'Sc
plt.xlabel('Subjects')
plt.ylabel('Scores')
plt.title('Exam Score Distribution by Subject')
```

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



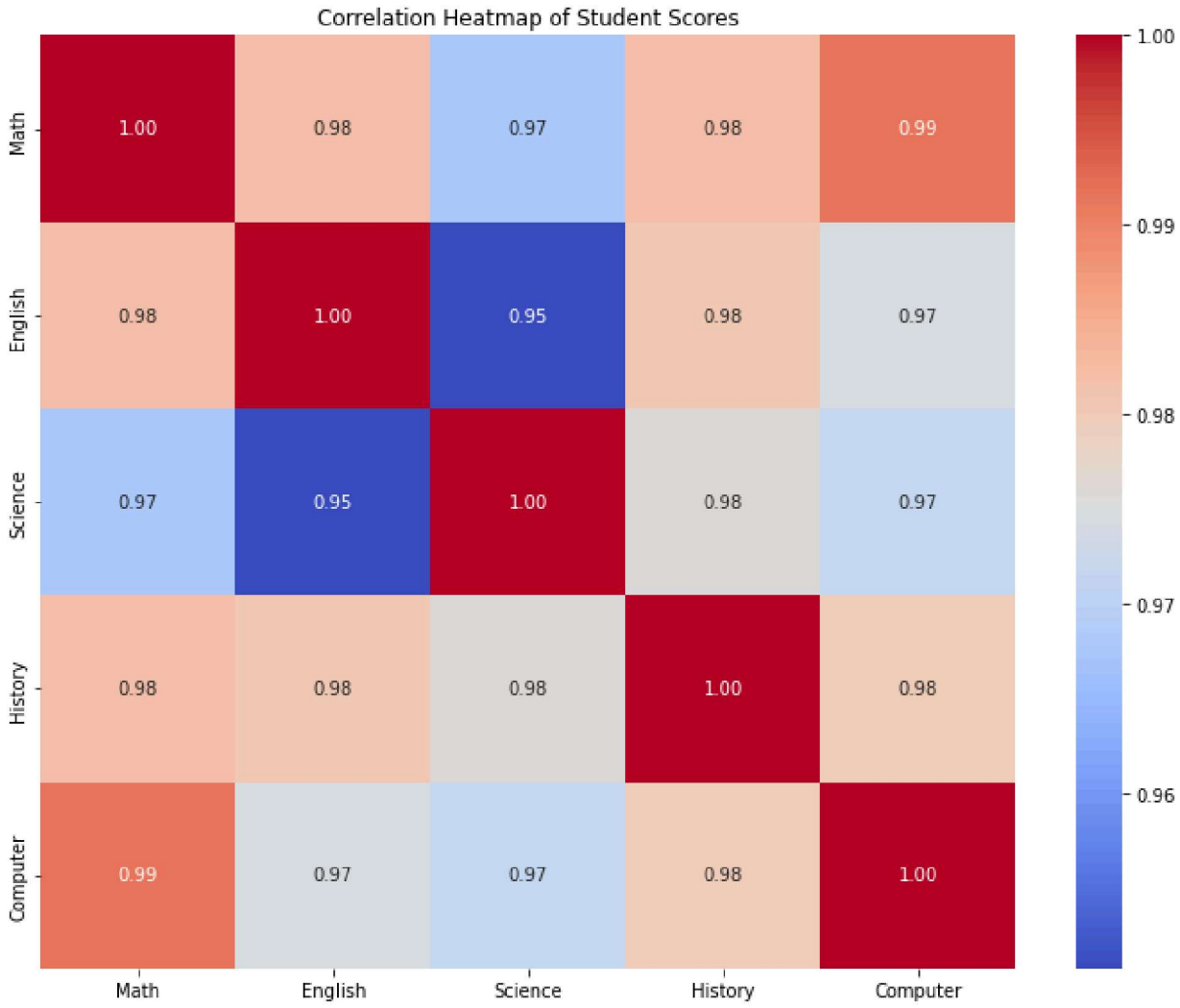
Q7. Area Chart (Website Traffic) Create a CSV file traffic.csv with columns: Month, Visitors. Write a Python program to plot an area chart showing monthly website visitors for a year. • Add axis labels and a title. • Use a different color for the filled area.

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

data = {
    'Month': ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep', 'Oct', 'N
    'Visitors': [1200, 1500, 1700, 1600, 1800, 2000, 2200, 2100, 1900, 2300, 2400, 250
}

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()
```


Correlation Heatmap of Student Scores
Math
English
Science
History
Computer
1.00 0.98 0.97 0.98 0.99
0.98 1.00 0.95 0.98 0.97
0.97 0.95 1.00 0.98 0.97
0.98 0.98 0.98 1.00 0.98
0.99 0.97 0.97 0.98 1.00
1.00
0.99
0.98
0.97
0.96

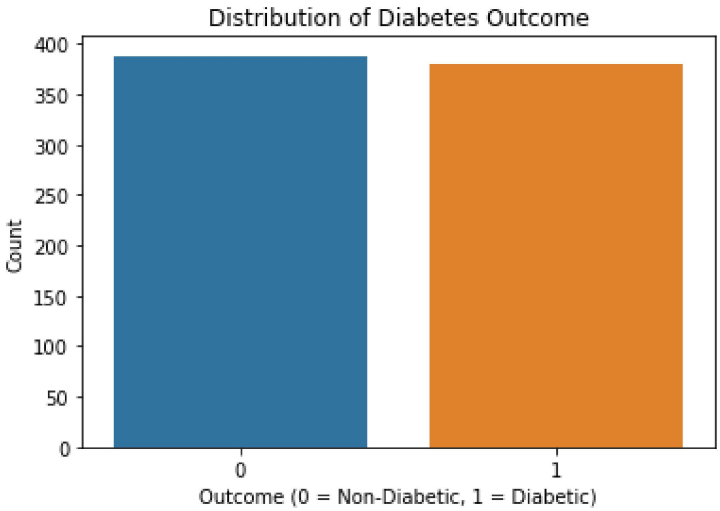
WEEK 7

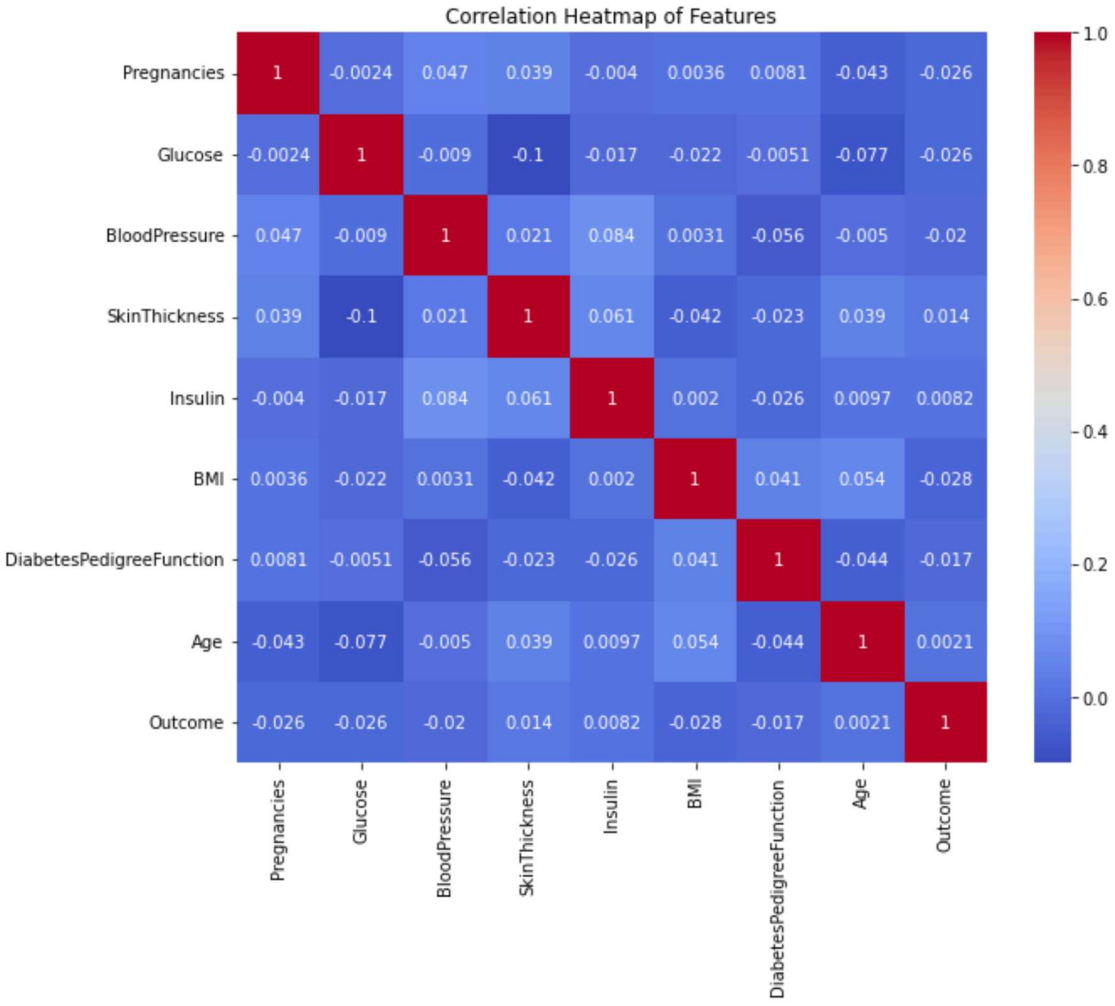
1. Write a Python program to collect, load, and perform initial exploration of the Diabetes dataset using Pandas. Display the first few records of the dataset and summarize its structure and contents.

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                   0
dtype: int64
```





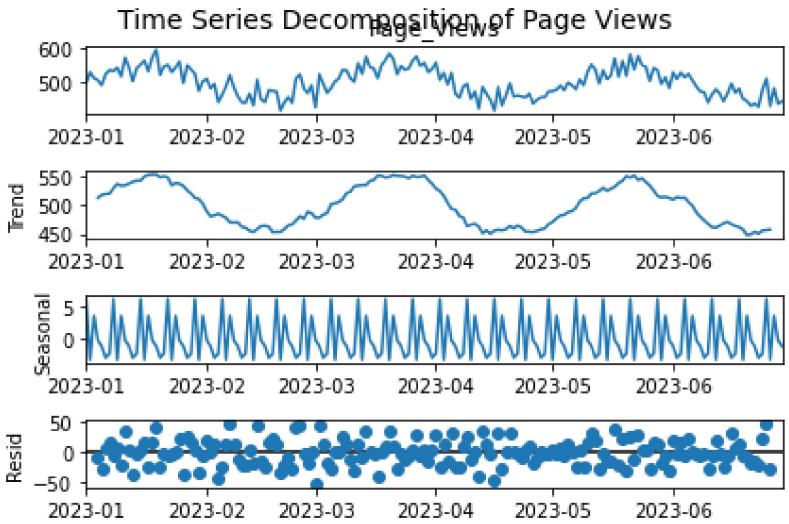
2) Write a Python program to perform Time Series Analysis on a website traffic dataset. The program should load and clean the dataset, decompose the time series to identify trends and seasonality, visualize key metrics using moving averages and seasonal plots, and detect anomalies using statistical methods.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

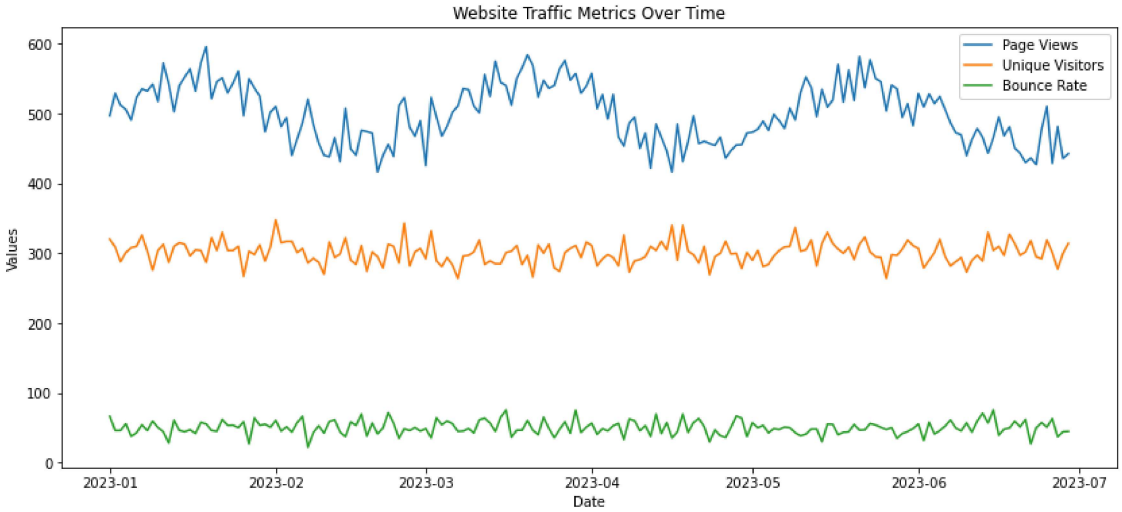
1. Load and Clean Data: • Import the dataset and parse the Date column as a datetime object. • Handle missing values by interpolation.
2. Time Series Decomposition: • Decompose the time series data of Page_Views to identify trend, seasonality, and residuals.
3. Visualization: • Plot the time series data 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.

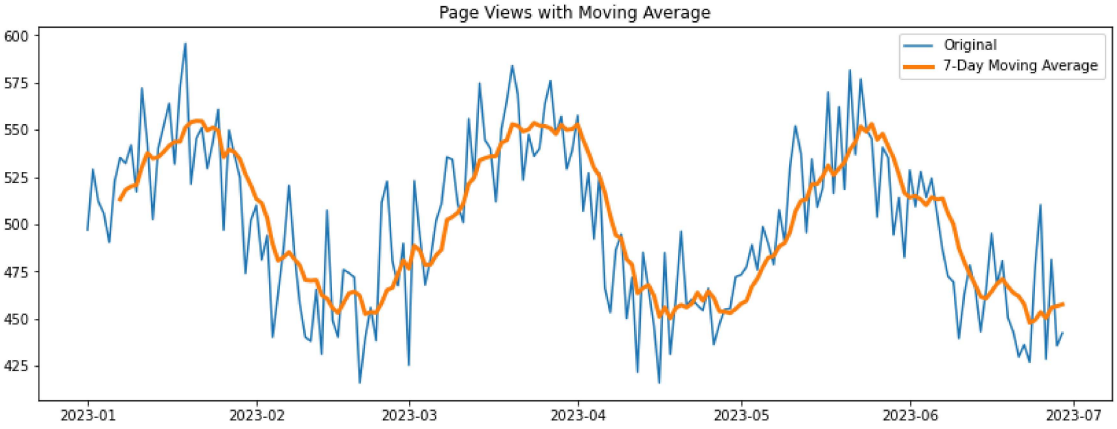
In [3]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from statsmodels.tsa.seasonal import seasonal_decompose
from scipy import stats
date_range = pd.date_range(start='2023-01-01', periods=180, freq='D')
data = {
    'Date': date_range,
    'Page_Views': 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','
plt.title('Seasonal Pattern by Day of Week')
plt.xticks(rotation=45)
plt.show()

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

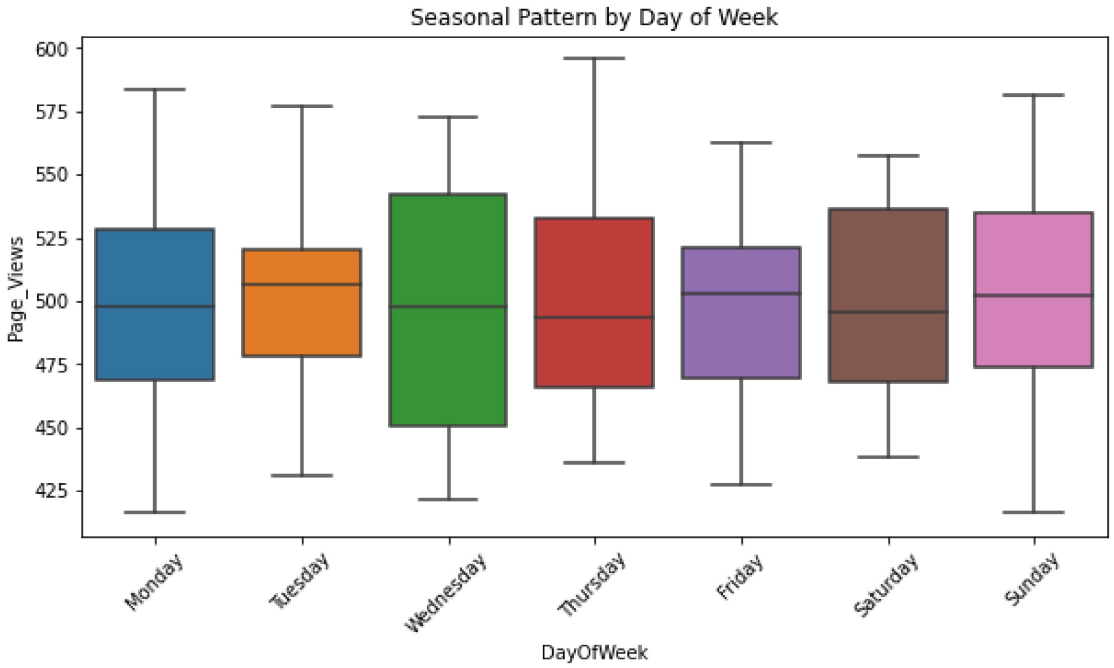
```
anomalies = df[df['Z_Score'] > 3]
plt.figure(figsize=(14, 5))
plt.plot(df.index, df['Page_Views'], label='Page Views')
plt.scatter(anomalies.index, anomalies['Page_Views'], color='red', label='Anomalies
plt.title('Anomaly Detection in Page Views')
plt.legend()
plt.show()

print(anomalies[['Page_Views', 'Z_Score']])
```

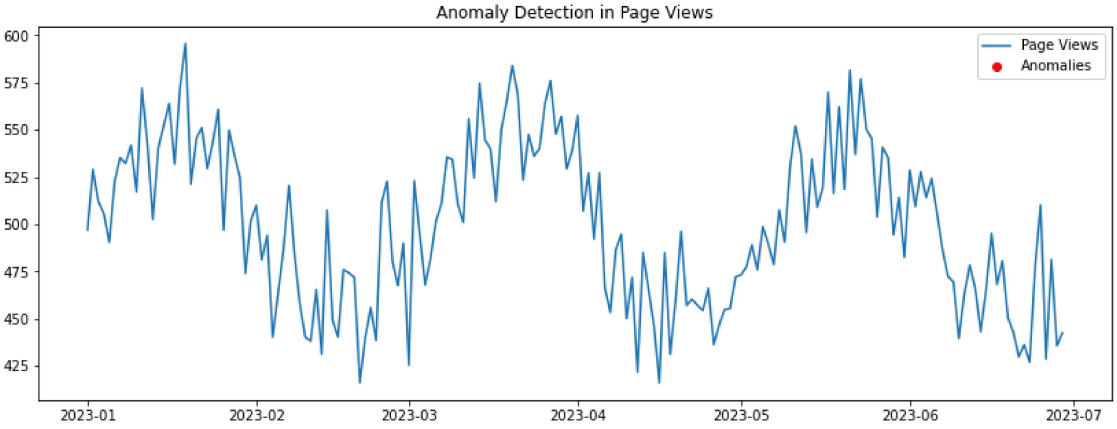
```
            Page_Views  Unique_Visitors  Bounce_Rate
Date
2023-01-01  497.000000              320    66.606963
2023-01-02  528.991671              309    46.365357
2023-01-03  511.933467              288    46.485806
2023-01-04  505.776010              301    55.963921
2023-01-05  490.470917              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>
```











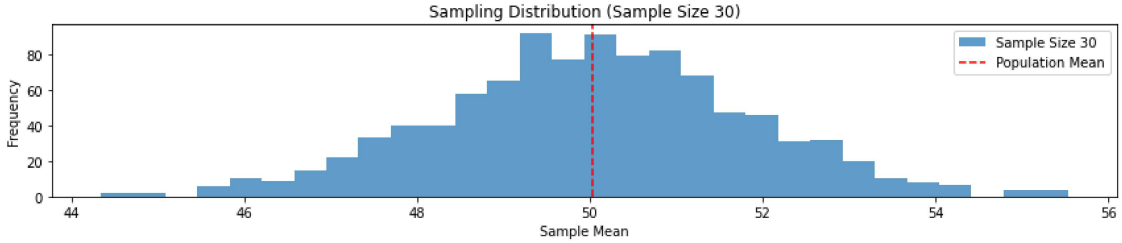

```
Empty DataFrame
Columns: [Page_Views, Z_Score]
Index: []
```

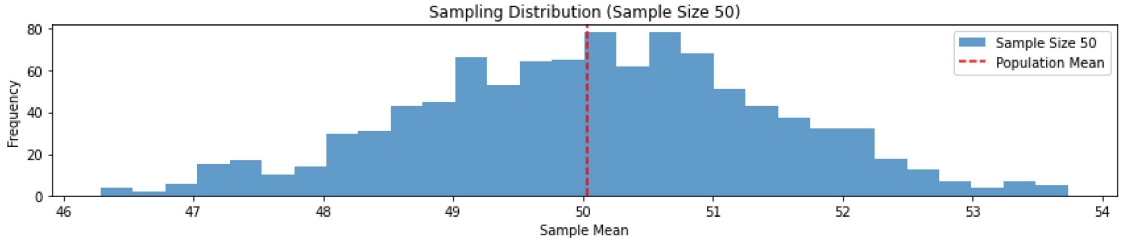
3) Random Sampling and Sampling Distribution To explore random sampling from a population and understand the concept of sampling distribution using Python in Jupyter Notebook. Steps:

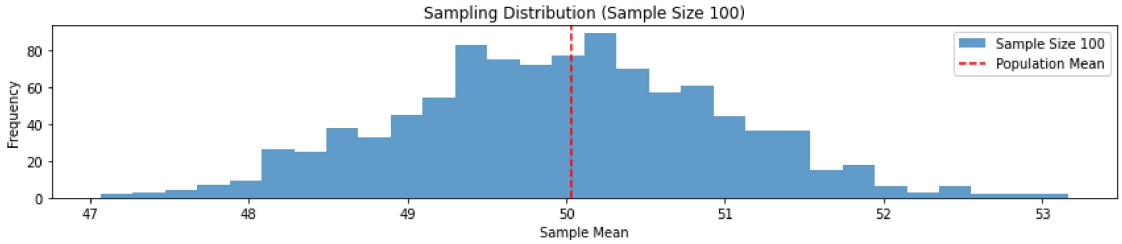
1. Generate a Population: • Create a population of data with a specified distribution (e.g., normal distribution).

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()
```


Sampling Distribution (Sample Size 30)
Sample Size 30
Population Mean
Frequency
Sample Mean
0
20
40
60
80
44
46
48
50
52
54
56


Sampling Distribution (Sample Size 50)
Sample Size 50
Population Mean
Frequency
Sample Mean
0
20
40
60
80
46
47
48
49
50
51
52
53
54


Sampling Distribution (Sample Size 100)
Sample Size 100
Population Mean
Frequency
Sample Mean
0
20
40
60
80
47
48
49
50
51
52
53

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.

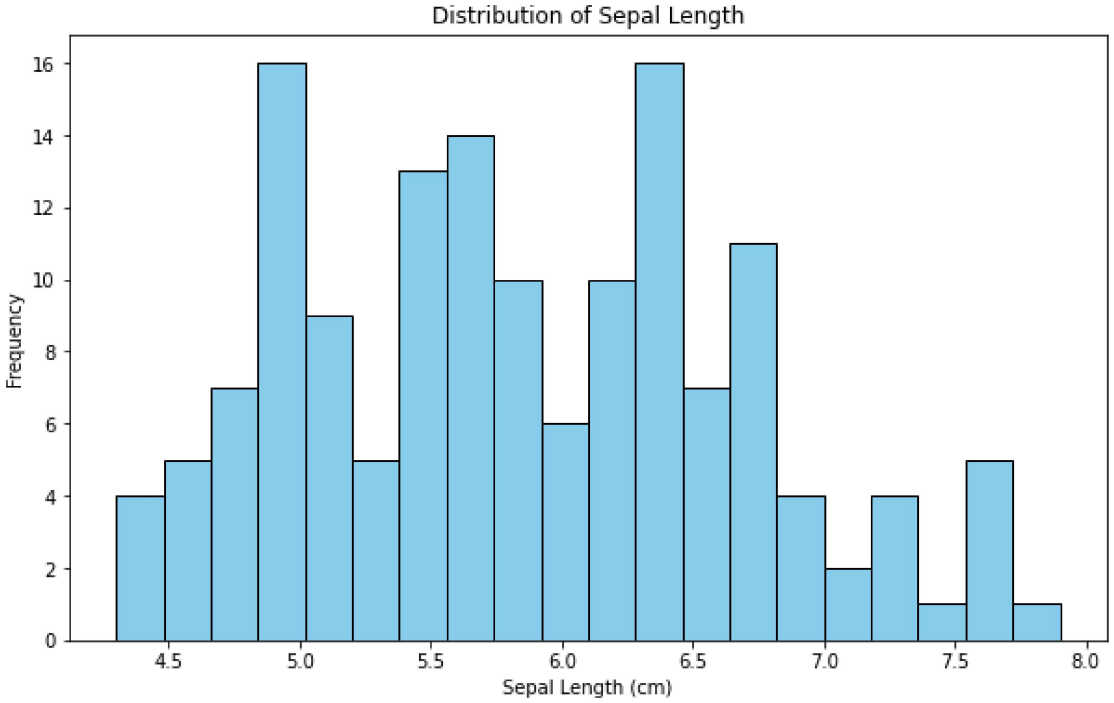
In [2]:
```
import numpy as np
import scipy.stats as stats
sample_data = np.array([152, 148, 151, 149, 147, 153, 150, 148, 152, 149, 151, 150,
                        148, 150, 152, 149, 150, 148, 153, 151, 150, 149, 152, 148,
population_mean = 150
sample_mean = np.mean(sample_data)
sample_std = np.std(sample_data, 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 differer
else:
    print("Fail to reject the null hypothesis: There is no significant difference i
```

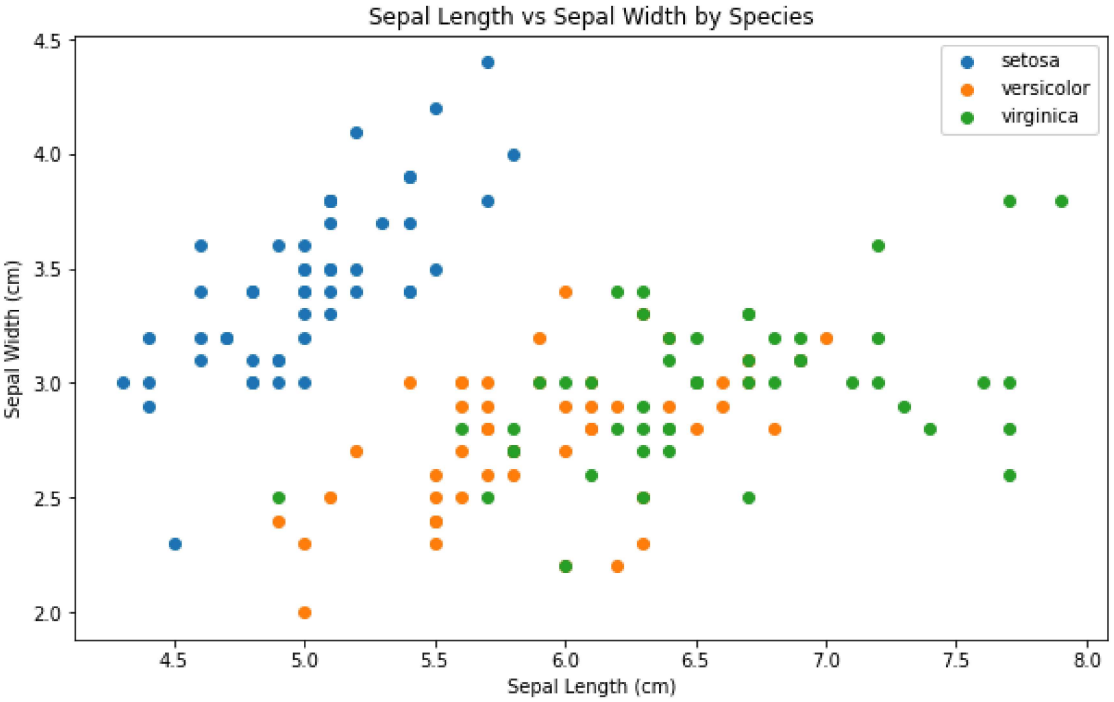
```
Sample Mean: 150.20
Z-Statistic: 0.6406
P-Value: 0.5218
Fail to reject the null hypothesis: There is no significant difference in average
weight from 150 grams.
```

1. Experiment to understand Matplotlib library use cases in Data Sciencethrough visualization. Description: Use Iris data set to understand the Matplot lib library

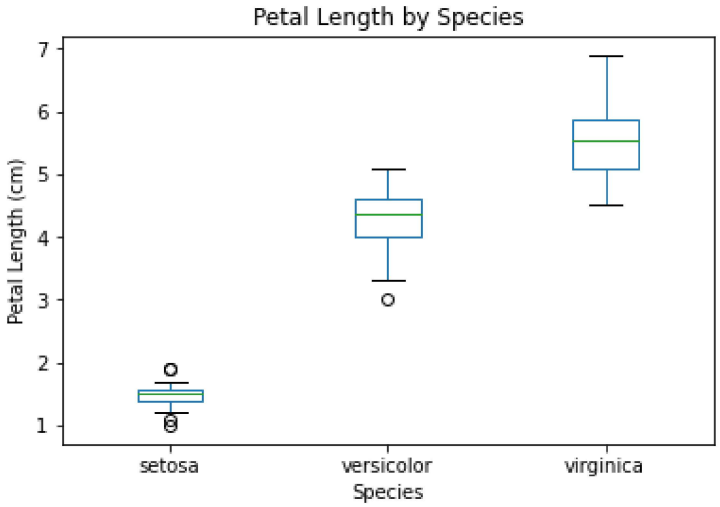
In [3]:
```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
iris = load_iris()
df = pd.DataFrame(iris.data, columns=iris.feature_names)
df['species'] = pd.Categorical.from_codes(iris.target, iris.target_names)

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=spec
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>
```


Petal Length by Species
Petal Length (cm)
7
6
5
4
3
2
1
setosa
versicolor
virginica
Species

In [ ]:
```
1.Experiment to understand Matplotlib library use cases in Data Science
through visualization
Description: Use Iris data set to understand the Matplot lib library
```

In [8]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
data=pd.read_csv('Iris_Dataset.csv')
print(data)
```

```
    sepal_length  sepal_width  petal_length  petal_width          species
0            5.1          3.5           1.4          0.2      Iris-setosa
1            4.9          3.0           1.4          0.2      Iris-setosa
2            4.7          3.2           1.3          0.2      Iris-setosa
3            4.6          3.1           1.5          0.2      Iris-setosa
4            5.0          3.6           1.4          0.2      Iris-setosa
5            5.4          3.9           1.7          0.4      Iris-setosa
6            4.6          3.4           1.4          0.3      Iris-setosa
7            5.0          3.4           1.5          0.2      Iris-setosa
8            4.4          2.9           1.4          0.2      Iris-setosa
9            4.9          3.1           1.5          0.1      Iris-setosa
10           7.0          3.2           4.7          1.4  Iris-versicolor
11           6.4          3.2           4.5          1.5  Iris-versicolor
12           6.9          3.1           4.9          1.5  Iris-versicolor
13           5.5          2.3           4.0          1.3  Iris-versicolor
14           6.5          2.8           4.6          1.5  Iris-versicolor
15           5.7          2.8           4.5          1.3  Iris-versicolor
16           6.3          3.3           4.7          1.6  Iris-versicolor
17           4.9          2.4           3.3          1.0  Iris-versicolor
18           6.6          2.9           4.6          1.3  Iris-versicolor
19           5.2          2.7           3.9          1.4  Iris-versicolor
20           6.3          2.5           4.9          1.5  Iris-versicolor
21           5.0          2.0           3.5          1.0  Iris-versicolor
22           5.9          3.0           4.2          1.5  Iris-versicolor
23           6.7          3.1           4.4          1.4   Iris-virginica
24           5.6          3.0           4.5          1.5   Iris-virginica
25           6.2          2.2           4.5          1.5   Iris-virginica
26           5.6          2.5           3.9          1.1   Iris-virginica
27           6.7          3.0           5.0          1.7   Iris-virginica
28           5.9          3.2           4.8          1.8   Iris-virginica
29           6.9          3.1           5.4          2.1   Iris-virginica
```

In [10]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30 entries, 0 to 29
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   sepal_length  30 non-null     float64
 1   sepal_width   30 non-null     float64
 2   petal_length  30 non-null     float64
 3   petal_width   30 non-null     float64
 4   species       30 non-null     object 
dtypes: float64(4), object(1)
memory usage: 1.3+ KB
```

In [11]:
```
data.describe()
```

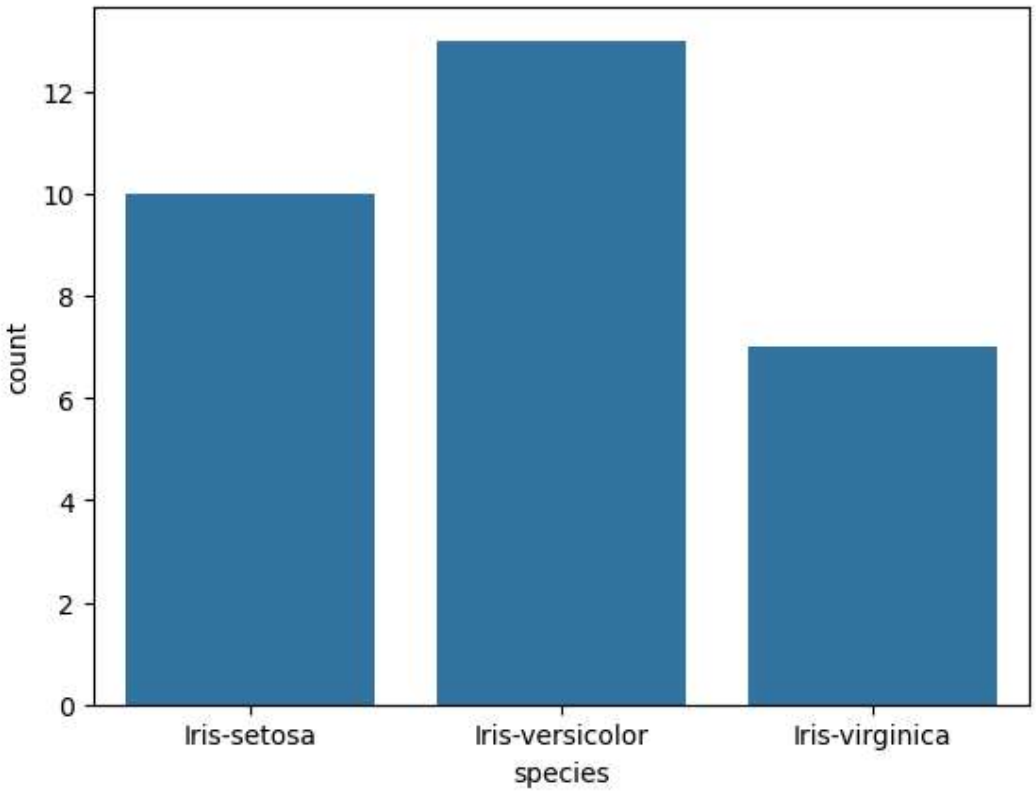
Out[11]:

	sepal_length	sepal_width	petal_length	petal_width
count	30.000000	30.000000	30.000000	30.000000
mean	5.680000	2.980000	3.443333	1.036667
std	0.805327	0.423776	1.493823	0.624491
min	4.400000	2.000000	1.300000	0.100000
25%	5.000000	2.800000	1.500000	0.225000
50%	5.600000	3.050000	4.100000	1.300000
75%	6.375000	3.200000	4.600000	1.500000
max	7.000000	3.900000	5.400000	2.100000

In [13]:
```
data.value_counts('species')
```

Out[13]:
```
species
Iris-versicolor    13
Iris-setosa        10
Iris-virginica      7
Name: count, dtype: int64
```

In [14]:
```
sns.countplot(x='species',data=data)
plt.show()
```



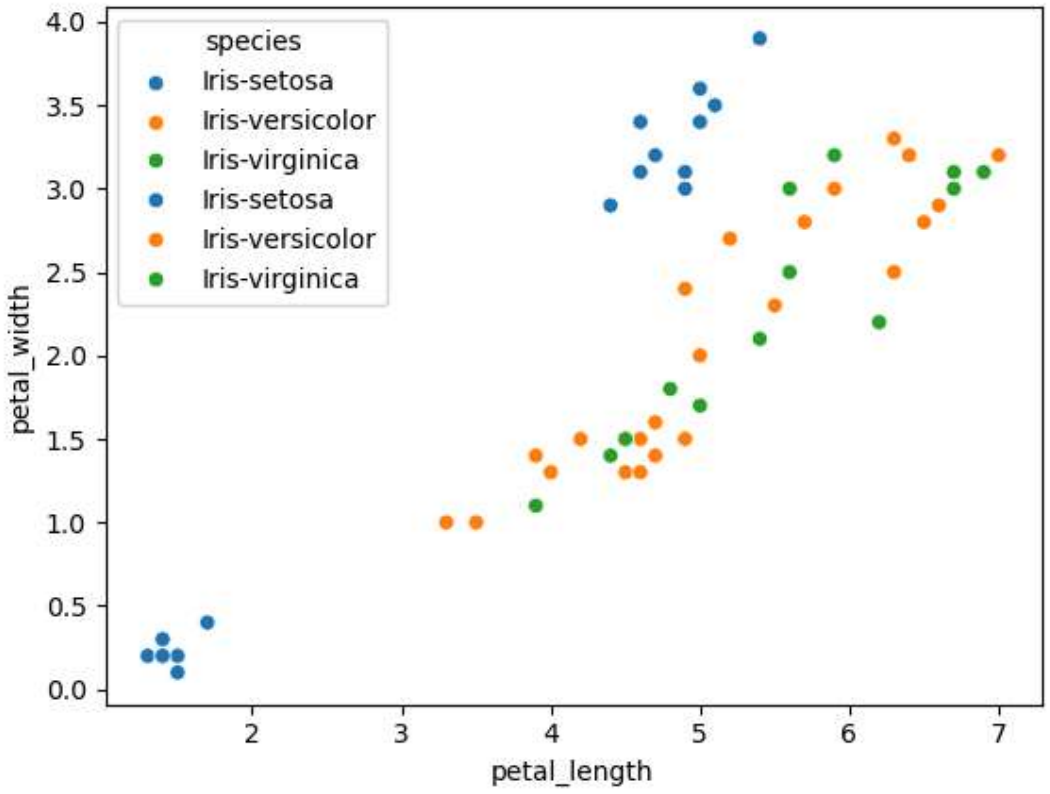
In [15]:
```
dummies=pd.get_dummies(data.species)

FinalDataset=pd.concat([pd.get_dummies(data.species),data.iloc[:, [0,1,2,3]]], a
```

```
FinalDataset.head()
```

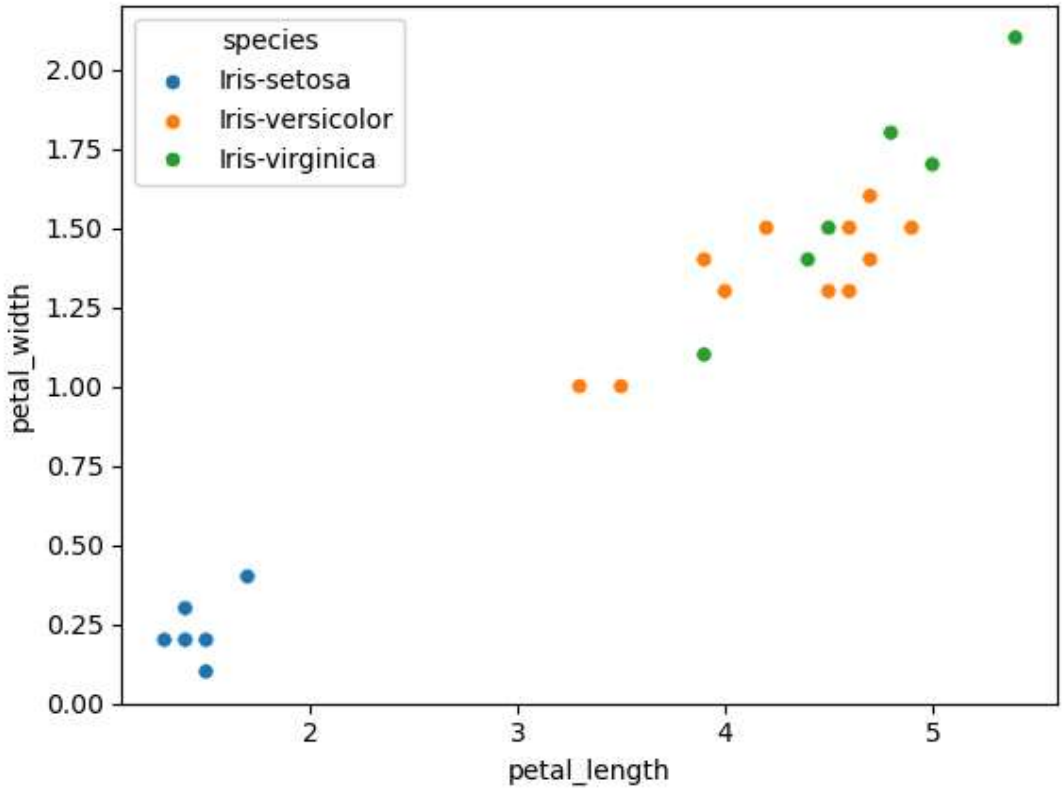
Out[15]:

	Iris-setosa	Iris-versicolor	Iris-virginica	sepal_length	sepal_width	petal_length	petal_width
0	True	False	False	5.1	3.5	1.4	0.2
1	True	False	False	4.9	3.0	1.4	0.2
2	True	False	False	4.7	3.2	1.3	0.2
3	True	False	False	4.6	3.1	1.5	0.2
4	True	False	False	5.0	3.6	1.4	0.2

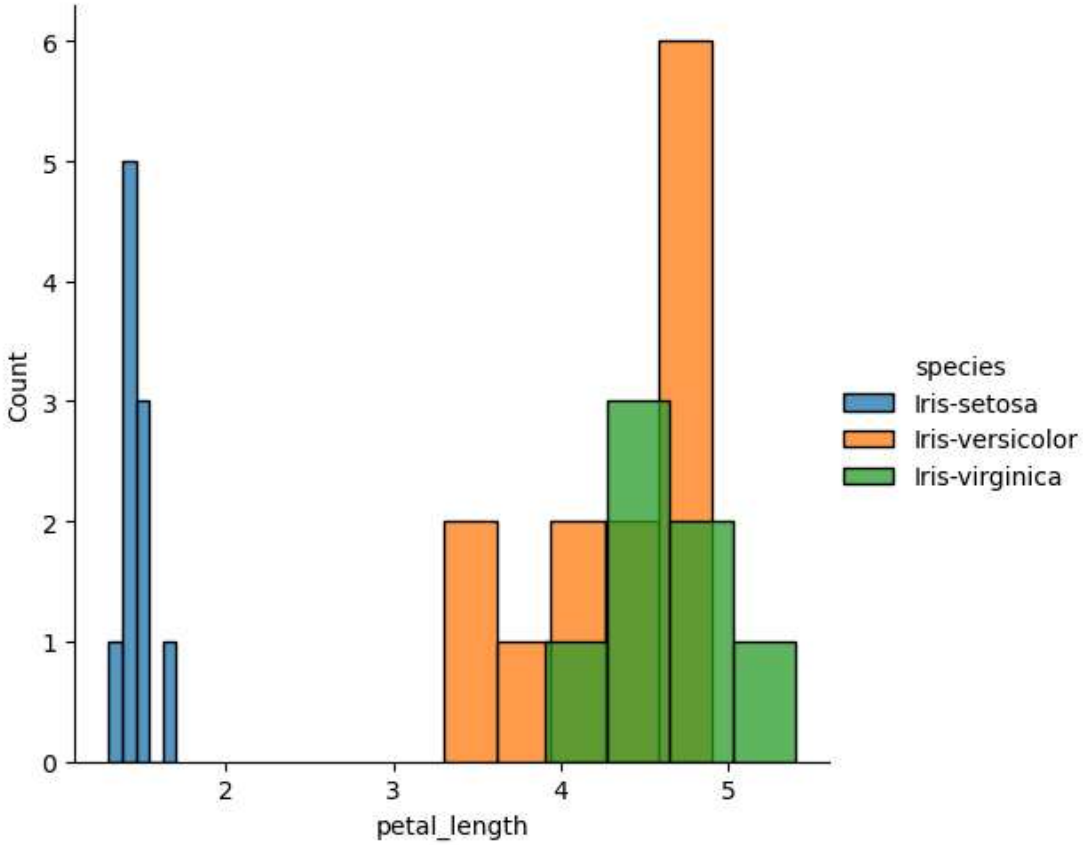
In [22]:
```
sns.scatterplot(x='sepal_length', y='sepal_width', hue='species', data=data)
plt.show()
```

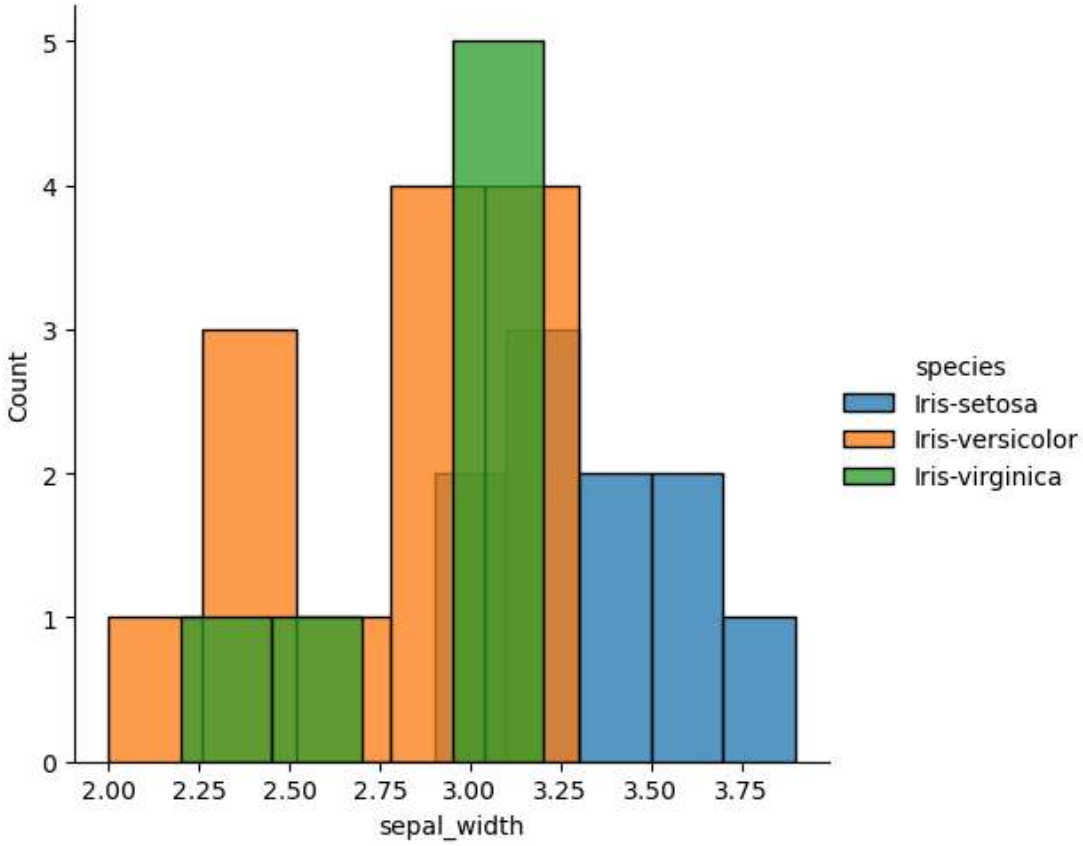


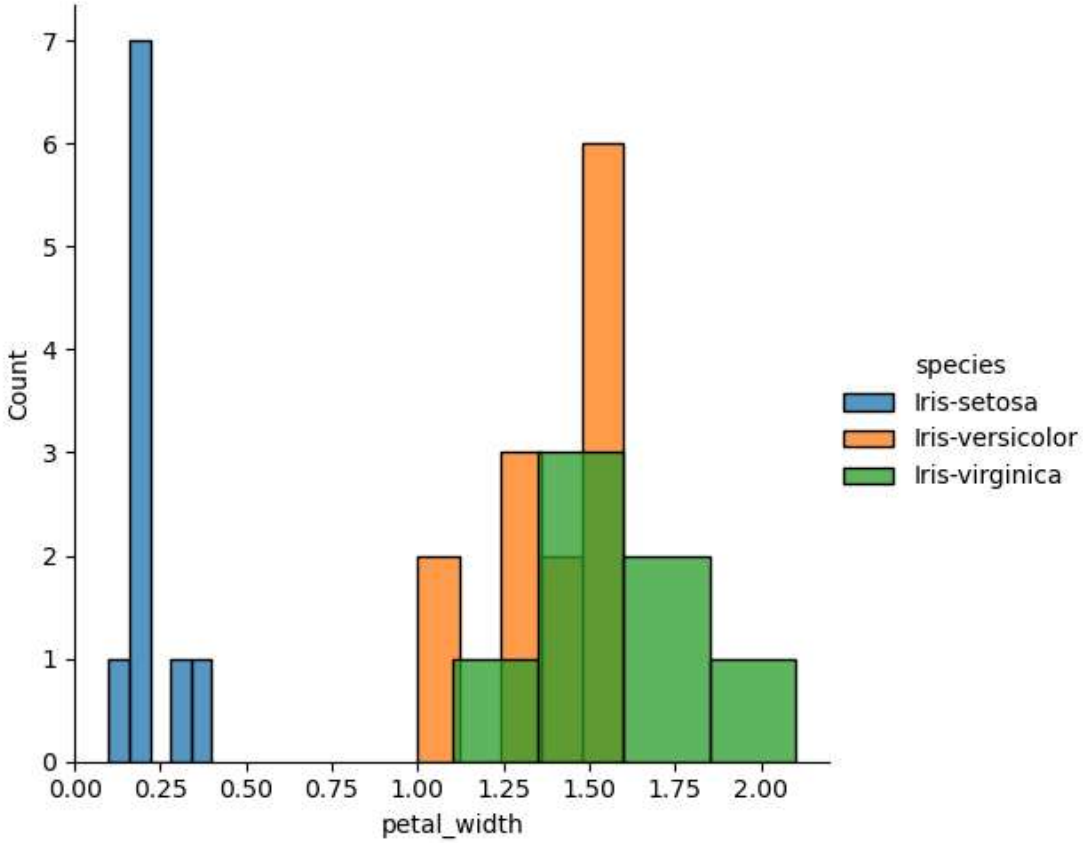
In [23]:
```
sns.scatterplot(x='petal_length', y='petal_width', hue='species', data=data)
plt.show()
```

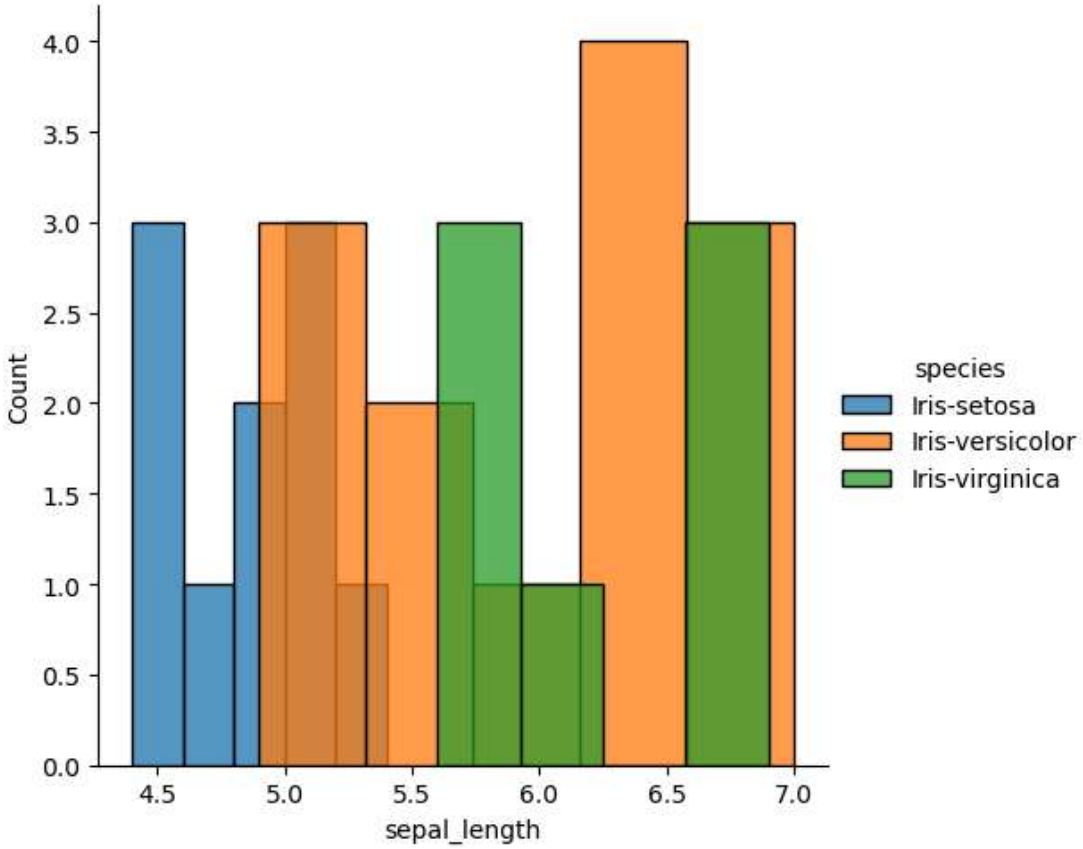


In [24]:
```
sns.FacetGrid(data, hue='species', height=5).map(sns.histplot, 'petal_length').a
plt.show()
sns.FacetGrid(data, hue='species', height=5).map(sns.histplot, 'sepal_width').ad
plt.show()
sns.FacetGrid(data, hue='species', height=5).map(sns.histplot, 'petal_width').ad
plt.show()
sns.FacetGrid(data, hue='species', height=5).map(sns.histplot, 'sepal_length').a
plt.show()
```


Count
petal_length
species
Iris-setosa
Iris-versicolor
Iris-virginica


Count
sepal_width
species
Iris-setosa
Iris-versicolor
Iris-virginica


Count
petal_width
species
Iris-setosa
Iris-versicolor
Iris-virginica


Count
sepal_length
species
Iris-setosa
Iris-versicolor
Iris-virginica

In [ ]: