

```
In [ ]: 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

```
In [ ]: Q: Read a CSV file named data.csv into a DataFrame and print the first 5 rows.
```

```
In [2]: filtered_df = df[df['Age'] > 28]
print(filtered_df)
```

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

```
In [ ]: 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

```
In [ ]: Q: Given a DataFrame with columns Department and Salary, find the average salary per
```

```
In [6]: import pandas as pd
data = {
    'Department': ['HR', 'IT', 'HR', 'Finance', 'IT', 'Finance'],
    'Salary': [50000, 60000, 55000, 70000, 65000, 72000]
}

df = pd.DataFrame(data)
average_salary = df.groupby('Department')['Salary'].mean()

print(average_salary)
```

Department	
Finance	71000.0
HR	52500.0
IT	62500.0

Name: Salary, dtype: float64

In []: Q: Replace all occurrences of 'New York' in City with 'NYC'.

```
In [12]: 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

In []: Q: Drop the Age column from the DataFrame.

```
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 = df.drop('Age', axis=1)
print(df)
```

	Name	City
0	Alice	New York
1	Bob	Los Angeles
2	Charlie	Chicago

In []: Q: Sort the DataFrame by Salary in descending order.

```
In [14]: import pandas as pd

data = {
    'Name': ['Alice', 'Bob', 'Charlie'],
    'Salary': [70000, 85000, 60000],
    'Department': ['HR', 'IT', 'Finance']
}

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

	Name	Salary	Department
1	Bob	85000	IT
0	Alice	70000	HR
2	Charlie	60000	Finance

In []: Q: Check for missing values in the DataFrame.

```
In [15]: import pandas as pd
import numpy as np
```

```
data = {
    'Name': ['Alice', 'Bob', 'Charlie', 'David'],
    'Age': [25, np.nan, 35, 40],
    'City': ['New York', 'Los Angeles', None, 'Chicago'],
    'Salary': [70000, 85000, 60000, None]
}

df = pd.DataFrame(data)

missing_values = df.isnull().sum()
print("Missing values per column:")
print(missing_values)

has_missing = df.isnull().values.any()
print("\nIs there any missing value in the DataFrame?")
print(has_missing)
```

Missing values per column:

```
Name      0
Age        1
City        1
Salary      1
dtype: int64
```

```
Is there any missing value in the DataFrame?
True
```

In []:

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

In [1]: **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))
```

	Name	Age	City
0	Alice	24	New York
1	Bob	27	Los Angeles

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:

b) Print the column names

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.columns)
```

```
Index(['Name', 'Age', 'City'], dtype='object')
```

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:

c) Show the shape of the DataFrame

In [3]: **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.shape)
```

(4, 3)

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:

d) Display the summary info of the DataFrame

In [4]: **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.info())
```

```
<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
```

In []: 2.a) Select only the Name column

In [5]: **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[['Name', 'City']])
```

```
      Name      City
0   Alice  New York
1    Bob  Los Angeles
2  Charlie   Chicago
3   David   Houston
```

In []: 2.b) Select both Name **and** City columns

In [6]: **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[['Name', 'City']])
```

	Name	City
0	Alice	New York
1	Bob	Los Angeles
2	Charlie	Chicago
3	David	Houston

In []: 2.c) Select the second row using `.iloc`

In [7]: `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.iloc[1])
```

Name	Bob
Age	27
City	Los Angeles

Name: 1, dtype: object

In []: 2.d) Select the row where Name is 'Charlie' using `.loc`

In [8]: `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.loc[df['Name'] == 'Charlie'])
```

	Name	Age	City
2	Charlie	22	Chicago

In []: 3.a) People older than 25

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)
print(df[df['Age'] > 25])
```

	Name	Age	City
1	Bob	27	Los Angeles
3	David	32	Houston

In []: 3.b) People living in 'Chicago' or 'Houston'

In [10]: `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[df['City'].isin(['Chicago', 'Houston'])])
```

	Name	Age	City
2	Charlie	22	Chicago
3	David	32	Houston

In []: 3.c) People whose age is between 23 and 30

```
In [11]: 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[(df['Age'] >= 23) & (df['Age'] <= 30)])
```

	Name	Age	City
0	Alice	24	New York
1	Bob	27	Los Angeles

In []: 4.a) Add a new column Score with values [85, 90, 88, 95]

```
In [12]: 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['Score'] = [85, 90, 88, 95]
print(df)
```

	Name	Age	City	Score
0	Alice	24	New York	85
1	Bob	27	Los Angeles	90
2	Charlie	22	Chicago	88
3	David	32	Houston	95

In []: 4.b) Change Bob's age to 28

```
In [13]: 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.at[1, 'Age'] = 28
print(df)
```

	Name	Age	City
0	Alice	24	New York
1	Bob	28	Los Angeles
2	Charlie	22	Chicago
3	David	32	Houston

In []: 4.c) Remove the City column

```
In [14]: 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 = df.drop(columns=['City'])
print(df)
```

	Name	Age
0	Alice	24
1	Bob	27
2	Charlie	22
3	David	32

In []: 4.d) Drop the row of David

```
In [15]: 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 = df[df['Name'] != 'David']
print(df)
```

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

In []: 5. 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

```
In [16]: import pandas as pd

data = {
    'Department': ['HR', 'IT', 'HR', 'IT'],
    'Salary': [30000, 50000, 35000, 55000],
    'Experience': [2, 5, 3, 6]
}
df = pd.DataFrame(data)
print(df.groupby('Department')['Salary'].mean())
```

Department	Salary
HR	32500.0
IT	52500.0

Name: Salary, dtype: float64

In []: 5. Create a new DataFrame:

```
data = {
    'Department': ['HR', 'IT', 'HR', 'IT'],
    'Salary': [30000, 50000, 35000, 55000],
```



```
'Experience': [2, 5, 3, 6]
}
```

Write code to:

b) Find maximum Experience **in** each Department

In [17]: **import** pandas **as** pd

```
data = {
    'Department': ['HR', 'IT', 'HR', 'IT'],
    'Salary': [30000, 50000, 35000, 55000],
    'Experience': [2, 5, 3, 6]
}
df = pd.DataFrame(data)
print(df.groupby('Department')['Experience'].max())
```

Department

HR 3

IT 6

Name: Experience, dtype: int64

In []: 5. Create a new DataFrame:

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

Write code to:

c) Calculate total Salary paid

In [18]: **import** pandas **as** pd

```
data = {
    'Department': ['HR', 'IT', 'HR', 'IT'],
    'Salary': [30000, 50000, 35000, 55000],
    'Experience': [2, 5, 3, 6]
}
df = pd.DataFrame(data)
print(df['Salary'].sum())
```

170000

In []: 6. 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

In [19]: **import** pandas **as** pd

```
data = {
    'Student': ['John', 'Emma', 'Sam', 'Olivia'],
    'Marks': [80, None, 75, 90]
}
df = pd.DataFrame(data)
df['Marks'] = df['Marks'].fillna(0) # safer than inplace
print(df)
```

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

In []: 6. Given a DataFrame **with** missing values:

```
data = {
    'Student': ['John', 'Emma', 'Sam', 'Olivia'],
    'Marks': [80, None, 75, 90]
}
```

Write code to:

b) Drop rows **with** missing values

```
In [20]: import pandas as pd

data = {
    'Student': ['John', 'Emma', 'Sam', 'Olivia'],
    'Marks': [80, None, 75, 90]
}
df = pd.DataFrame(data)
df = df.dropna()
print(df)
```

	Student	Marks
0	John	80.0
2	Sam	75.0
3	Olivia	90.0

In []: 6. Given a DataFrame **with** missing values:

```
data = {
    'Student': ['John', 'Emma', 'Sam', 'Olivia'],
    'Marks': [80, None, 75, 90]
}
```

Write code to:

c) Sort the DataFrame by Marks **in** descending order

```
In [21]: import pandas as pd

data = {
    'Student': ['John', 'Emma', 'Sam', 'Olivia'],
    'Marks': [80, None, 75, 90]
}
df = pd.DataFrame(data)
df = df.sort_values(by='Marks', ascending=False)
print(df)
```

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

In []: 7.a) Save it **as** a CSV file named students.csv

```
In [22]: import pandas as pd

df = pd.DataFrame({
    'Name': ['Alice', 'Bob'],
    'Age': [25, 28]
})
df.to_csv('students.csv', index=False)
```

In []: 7.b) Load a CSV file named `employees.csv`

In [23]: `import pandas as pd`

```
df = pd.read_csv('students.csv')
print(df)
```

	Name	Age
0	Alice	25
1	Bob	28

In []: 7.c) Set Name `as` the index

In [24]: `import pandas as pd`

```
df = pd.DataFrame({
    'Name': ['Alice', 'Bob'],
    'Age': [25, 28]
})

df.set_index('Name', inplace=True)
print(df)
```

	Age
Name	
Alice	25
Bob	28

In []: 7.d) Reset the index

In [25]: `import pandas as pd`

```
df = pd.DataFrame({
    'Name': ['Alice', 'Bob'],
    'Age': [25, 28]
})

df.set_index('Name', inplace=True)

df.reset_index(inplace=True)
print(df)
```

	Name	Age
0	Alice	25
1	Bob	28

In []: 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.
Expected Output:
Product Price Stock
0 Laptop 70000 10
2 Smartphone 25000 50
```

```
In [26]: 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

In []: 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'.

Expected Output:

Subject Science

Marks 92

Name: Student2, dtype: object

```
In [27]: 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

In []:

In []: 1. You are assigned a data creation task. Based on the field/topic given to you, Pandas to create a CSV file containing at least 25 entries. Each record should have relevant columns (fields) suitable for your dataset.

```
In [1]: import pandas as pd

data = {
    'Dish_ID': ['D001', 'D002', 'D003', 'D004', 'D005',
               'D006', 'D007', 'D008', 'D009', 'D010',
               'D011', 'D012', 'D013', 'D014', 'D015',
               'D016', 'D017', 'D018', 'D019', 'D020',
               'D021', 'D022', 'D023', 'D024', 'D025'],

    'Dish_Name': ['Spring Rolls', 'Paneer Tikka', 'Chicken Wings', 'Garlic Bread',
                  'Butter Chicken', 'Paneer Butter Masala', 'Veg Biryani', 'Chic',
                  'Palak Paneer', 'Mutton Rogan Josh', 'Gulab Jamun', 'Rasgulla',
                  'Chocolate Brownie', 'Cheesecake', 'Masala Chai', 'Cold Coffee',
                  'Lemonade', 'Green Tea', 'Paneer Tikka Special 23', 'Veg Biry',
                  'Chocolate Brownie Special 25'],

    'Category': ['Starters', 'Starters', 'Starters', 'Starters', 'Starters',
                 'Main Course', 'Main Course', 'Main Course', 'Main Course', 'Main C',
                 'Main Course', 'Main Course', 'Dessert', 'Dessert', 'Dessert',
                 'Dessert', 'Dessert', 'Beverage', 'Beverage', 'Beverage',
                 'Beverage', 'Beverage', 'Starters', 'Main Course', 'Dessert'],

    'Price': [207, 128, 106, 112, 198,
              415, 311, 480, 316, 337,
              419, 453, 157, 167, 145,
              172, 161, 78, 87, 88,
              73, 68, 248, 482, 155],

    'Ingredients': [
        'Rice, Onion, Flour, Paneer, Garlic',
        'Herbs, Paneer, Spices, Milk, Onion',
        'Butter, Garlic, Paneer, Onion, Flour',
        'Butter, Onion, Herbs, Spices, Rice',
        'Onion, Garlic, Paneer, Butter, Spices',
        'Mutton, Garlic, Onion, Spices, Butter',
        'Cheese, Spices, Paneer, Onion, Garlic',
        'Rice, Tomato, Onion, Garlic, Spices',
        'Rice, Chicken, Garlic, Spices, Onion',
        'Garlic, Onion, Tomato, Spices, Herbs',
        'Paneer, Onion, Garlic, Herbs, Spices',
        'Mutton, Onion, Tomato, Garlic, Spices',
        'Milk, Flour, Sugar, Butter, Paneer',
        'Sugar, Paneer, Milk, Butter, Cheese',
        'Milk, Chocolate, Sugar, Butter, Paneer',
        'Sugar, Chocolate, Flour, Butter, Milk',
        'Cheese, Milk, Sugar, Butter, Flour',
        'Tea Leaves, Milk, Sugar, Spices, Herbs',
        'Milk, Sugar, Chocolate, Butter, Paneer',
        'Milk, Sugar, Paneer, Butter, Herbs',
        'Sugar, Herbs, Lemon, Water, Ice',
        'Tea Leaves, Water, Herbs, Lemon',
        'Onion, Spices, Paneer, Garlic, Tomato',
        'Rice, Spices, Onion, Garlic, Tomato',
        'Chocolate, Flour, Sugar, Butter, Milk']
}
```

```
],  
  
    'Vegetarian': ['Yes', 'Yes', 'No', 'Yes', 'Yes',  
                  'No', 'Yes', 'Yes', 'No', 'Yes',  
                  'Yes', 'No', 'Yes', 'Yes', 'Yes',  
                  'Yes', 'Yes', 'Yes', 'Yes', 'Yes',  
                  'Yes', 'Yes', 'Yes', 'Yes', 'Yes'],  
  
    'Preparation_Time (mins)': [20, 19, 18, 18, 13,  
                                24, 25, 34, 36, 37,  
                                35, 23, 14, 13, 14,  
                                13, 9, 7, 9, 8,  
                                7, 6, 20, 38, 12]  
}  
df = pd.DataFrame(data)  
df.to_csv('restaurant_menu.csv', index=False)  
print("CSV file 'restaurant_menu.csv' created successfully.")
```

CSV file 'restaurant_menu.csv' created successfully.

In []:

Dish_ID	Dish_Name	Category	Price	Ingredients	Vegetarian	Preparation_Time (mins)
D001	Spring Rolls	Starters	207	"Rice, Onion, Flour, Paneer, Garlic"	Yes	20
D002	Paneer Tikka	Starters	128	"Herbs, Paneer, Spices, Milk, Onion"	Yes	19
D003	Chicken Wings	Starters	106	"Butter, Garlic, Paneer, Onion, Flour"	No	18
D004	Garlic Bread	Starters	112	"Butter, Onion, Herbs, Spices, Rice"	Yes	18
D005	Hummus Platter	Starters	198	"Onion, Garlic, Paneer, Butter, Spices"	Yes	13
D006	Butter Chicken	Main Course	415	"Mutton, Garlic, Onion, Spices, Butter"	No	24
D007	Paneer Butter Masala	Main Course	311	"Cheese, Spices, Paneer, Onion, Garlic"	Yes	25
D008	Veg Biryani	Main Course	480	"Rice, Tomato, Onion, Garlic, Spices"	Yes	34
D009	Chicken Biryani	Main Course	316	"Rice, Chicken, Garlic, Spices, Onion"	No	36
D010	Dal Tadka	Main Course	337	"Garlic, Onion, Tomato, Spices, Herbs"	Yes	37
D011	Palak Paneer	Main Course	419	"Paneer, Onion, Garlic, Herbs, Spices"	Yes	35
D012	Mutton Rogan Josh	Main Course	453	"Mutton, Onion, Tomato, Garlic, Spices"	No	23
D013	Gulab Jamun	Dessert	157	"Milk, Flour, Sugar, Butter, Paneer"	Yes	14
D014	Rasgulla	Dessert	167	"Sugar, Paneer, Milk, Butter, Cheese"	Yes	13
D015	Ice Cream Sundae	Dessert	145	"Milk, Chocolate, Sugar, Butter, Paneer"	Yes	14
D016	Chocolate Brownie	Dessert	172	"Sugar, Chocolate, Flour, Butter, Milk"	Yes	13
D017	Cheesecake	Dessert	161	"Cheese, Milk, Sugar, Butter, Flour"	Yes	9
D018	Masala Chai	Beverage	78	"Tea Leaves, Milk, Sugar, Spices, Herbs"	Yes	7
D019	Cold Coffee	Beverage	87	"Milk, Sugar, Chocolate, Butter, Paneer"	Yes	9
D020	Mango Lassi	Beverage	88	"Milk, Sugar, Paneer, Butter, Herbs"	Yes	8
D021	Lemonade	Beverage	73	"Sugar, Herbs, Lemon, Water, Ice"	Yes	7
D022	Green Tea	Beverage	68	"Tea Leaves, Water, Herbs, Lemon"	Yes	6
D023	Paneer Tikka Special 23	Starters	248	"Onion, Spices, Paneer, Garlic, Tomato"	Yes	20
D024	Veg Biryani Special 24	Main Course	482	"Rice, Spices, Onion, Garlic, Tomato"	Yes	38
D025	Chocolate Brownie Special 25	Dessert	155	"Chocolate, Flour, Sugar, Butter, Milk"	Yes	12

In []: 1. The following table shows the number of Data Science job postings recorded over the years from 2010 to 2020.

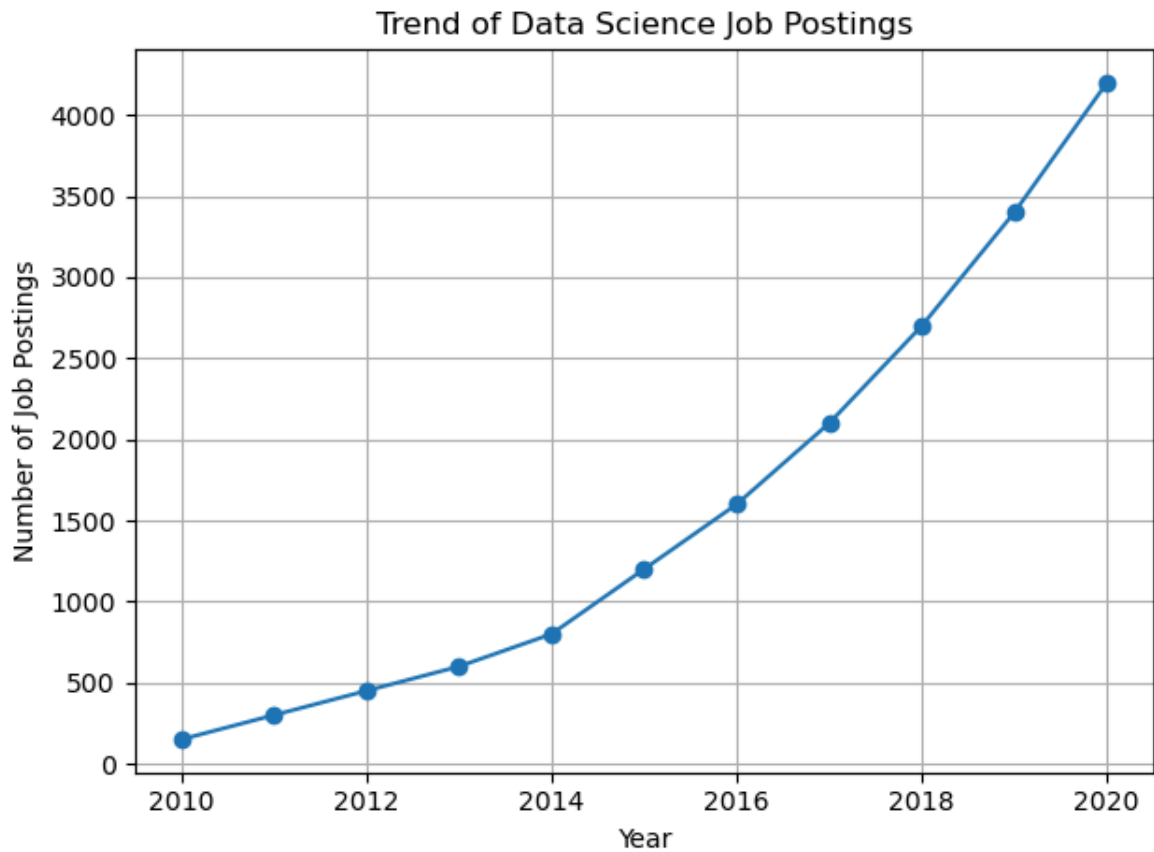
Year	Job Postings
2010	150
2011	300
2012	450
2013	600
2014	800
2015	1200
2016	1600
2017	2100
2018	2700
2019	3400
2020	4200

Using Python (Pandas and Matplotlib):

1. Create a DataFrame for the given data.
2. Plot a line graph showing the trend of Data Science job postings over the years on data points.
3. Add suitable title and axis labels to the graph.

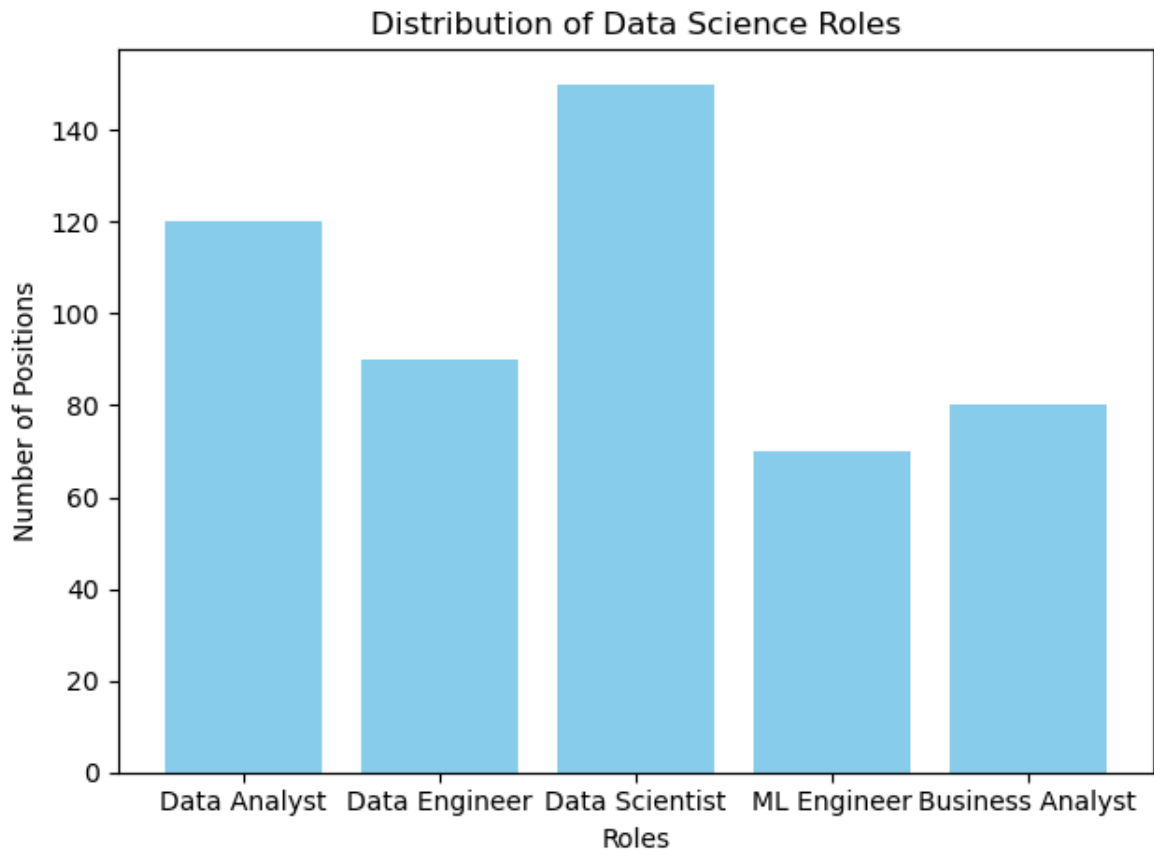
Expected Output: A line chart showing steady growth of job postings from 2010 to 2020.

```
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.grid(True)
plt.tight_layout()
plt.show()
```

In []: 2. 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, Business Analyst) with their respective counts. Add appropriate axis labels and chart.

```
In [2]: import matplotlib.pyplot as plt
roles = ['Data Analyst', 'Data Engineer', 'Data Scientist', 'ML Engineer', 'Business Analyst']
counts = [120, 90, 150, 70, 80]
plt.bar(roles, counts, color='skyblue')
plt.title('Distribution of Data Science Roles')
plt.xlabel('Roles')
plt.ylabel('Number of Positions')
plt.tight_layout()
plt.show()
```



In []: 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 for unstructured data, and a dictionary for semi-structured data. Print each type with clear labels.

```
In [3]: import pandas as pd
a = pd.DataFrame({
    'Name': ['Alice', 'Bob'],
    'Age': [25, 30],
    'Role': ['Data Scientist', 'Data Engineer']
})
b = "This is an example of unstructured data. It could be a paragraph, email, or"
c = {'Name': 'Alice',
    'Contact': {
        'Email': 'alice@example.com',
        'Phone': '1234567890'
    },
    'Skills': ['Python', 'ML', 'SQL']
}
print("Structured Data (DataFrame):\n", a, '\n')
print("Unstructured Data (Text):\n", b, '\n')
print("Semi-structured Data (Dictionary):\n", c)
```

Structured Data (DataFrame):

	Name	Age	Role
0	Alice	25	Data Scientist
1	Bob	30	Data Engineer

Unstructured Data (Text):

This is an example of unstructured data. It could be a paragraph, email, or social media post.

Semi-structured Data (Dictionary):

```
{'Name': 'Alice', 'Contact': {'Email': 'alice@example.com', 'Phone': '1234567890'}, 'Skills': ['Python', 'ML', 'SQL']}
```

In []: 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. Print the original, encrypted, and decrypted data.

```
In [6]: from cryptography.fernet import Fernet
key = Fernet.generate_key()
f = Fernet(key)
token = f.encrypt(b"Rajalakshmi Engineering College")
token
f.decrypt(token)
key = Fernet.generate_key()
cipher_suite = Fernet(key)
plain_text = b"Rajalakshmi Engineering College."
cipher_text = cipher_suite.encrypt(plain_text)
decrypted_text = cipher_suite.decrypt(cipher_text)
print("Original Data:", plain_text)
print("Encrypted Data:", cipher_text)
print("Decrypted Data:", decrypted_text)
```

Original Data: b'Rajalakshmi Engineering College.'

Encrypted Data: b'gAAAAABovnoT2NDFnlvelT4EXmiHKVwqrCwx2ngKagNrPmxqGsTFgK14pWCn9yvS5qTUc_yBuS7UeM2cUHLT4V_dDPx18MJFbcXTC7wIq3qp408cW-M3gNN2s5jpp-30U0111VtLP_Bg'

Decrypted Data: b'Rajalakshmi Engineering College.'

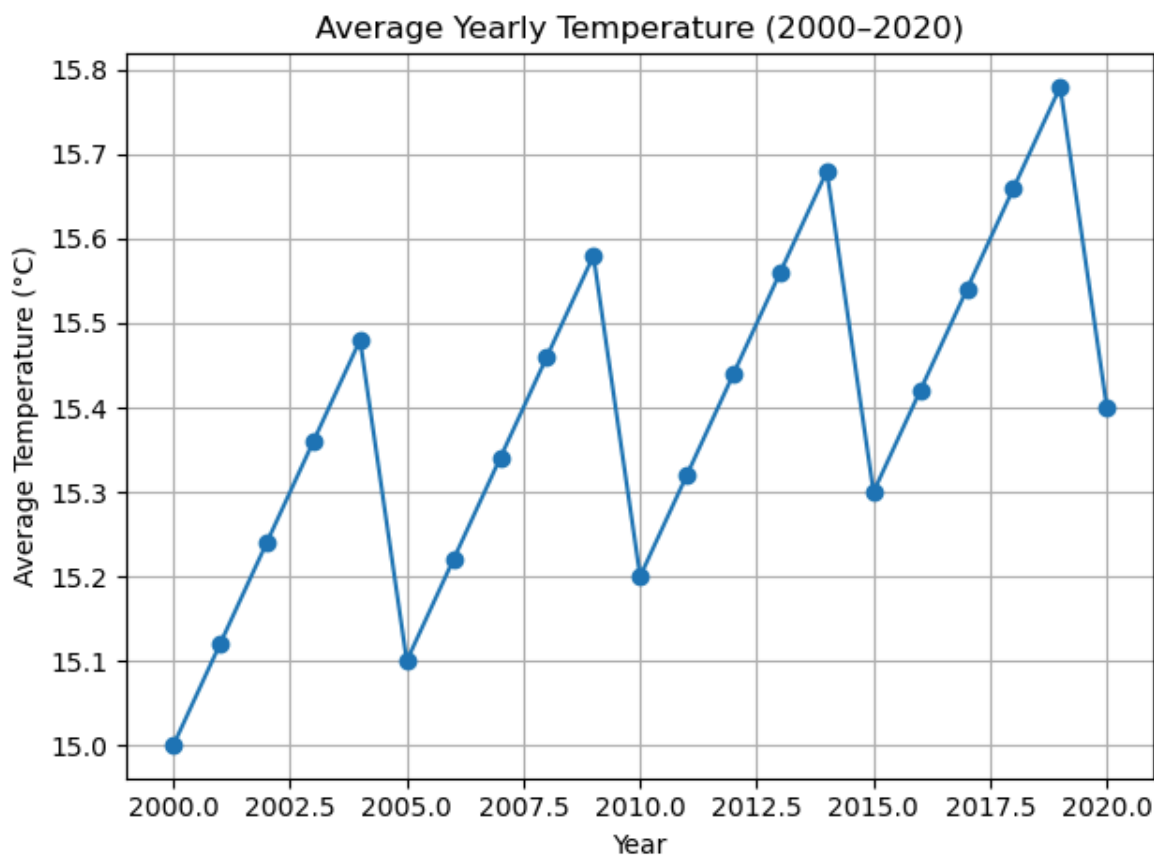
In []: Q1. Line Plot from CSV (Temperature Trends)

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

data = {
    'Year': list(range(2000, 2021)),
    'AvgTemperature': [15 + (i % 5) * 0.1 + i*0.02 for i in range(21)]
}

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

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



In []: Q2. Bar Chart from CSV (Sales Data)

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

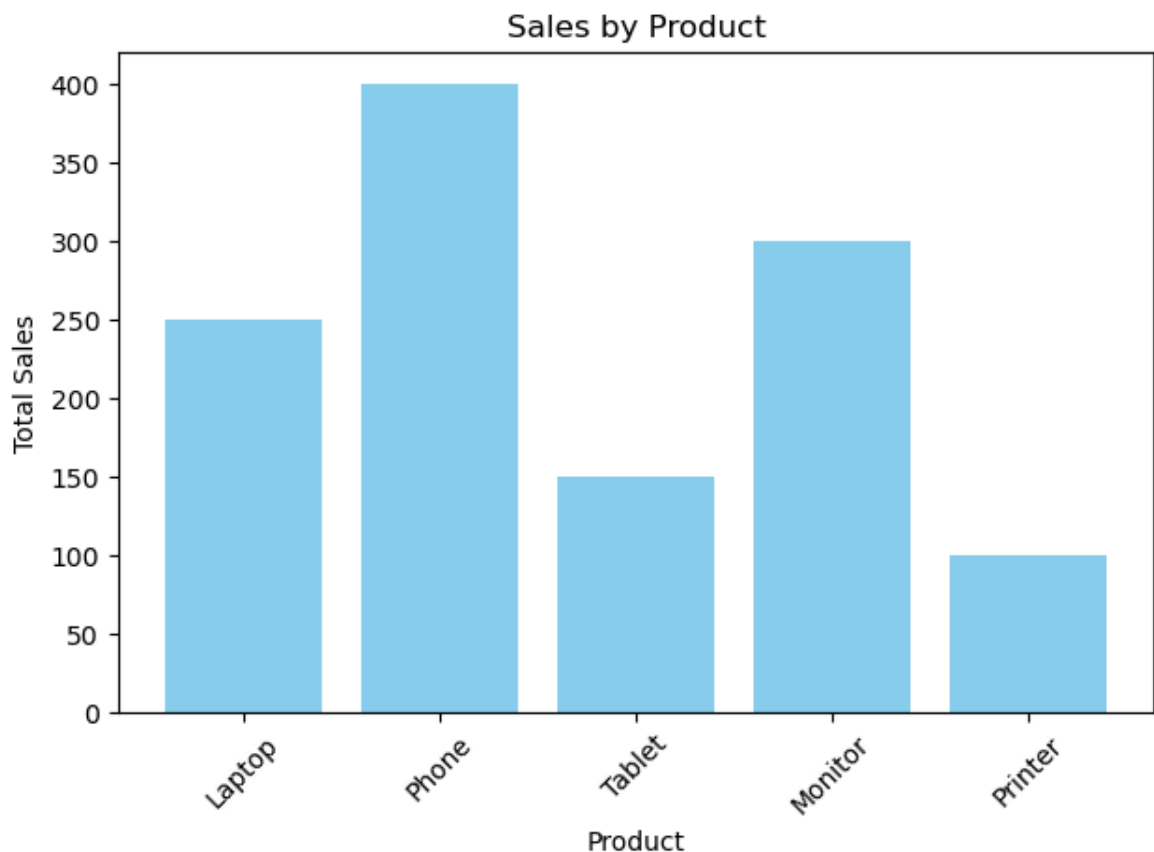
data = {
```

```

    'Product': ['Laptop', 'Phone', 'Tablet', 'Monitor', 'Printer'],
    'Sales': [250, 400, 150, 300, 100]
}
df = pd.DataFrame(data)
df.to_csv('sales.csv', index=False)

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

```



In []: Q3. Pie Chart (Movie Genres Distribution)

```

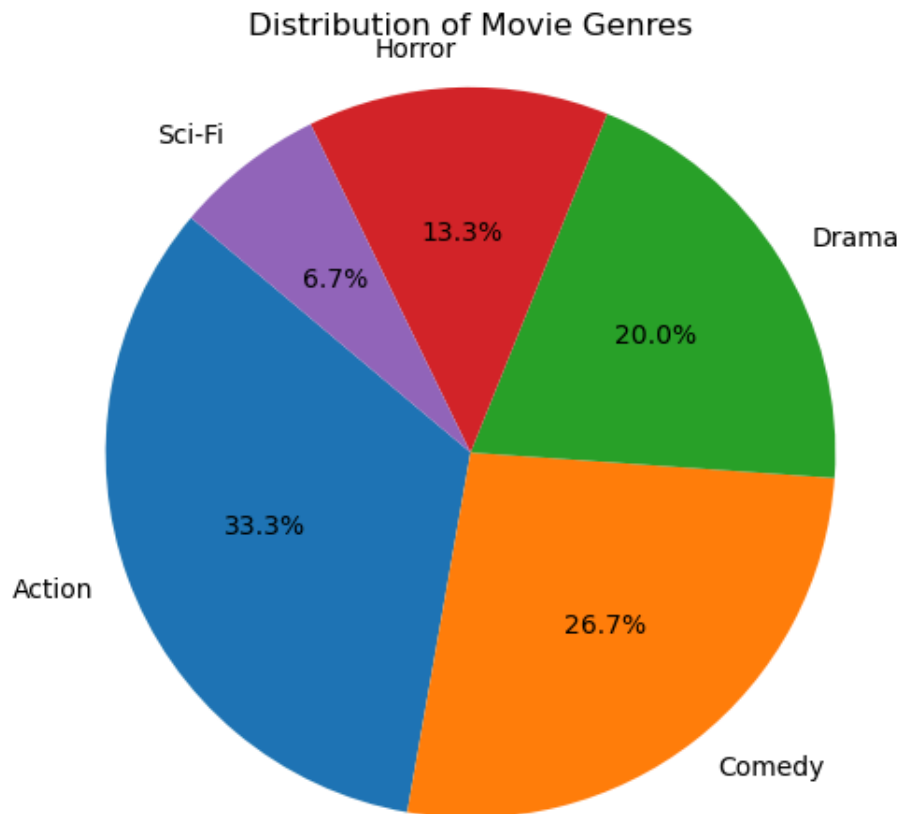
In [4]: import pandas as pd
import matplotlib.pyplot as plt

data = {
    'Genre': ['Action', 'Comedy', 'Drama', 'Horror', 'Sci-Fi'],
    'Count': [50, 40, 30, 20, 10]
}
df = pd.DataFrame(data)
df.to_csv('movies.csv', index=False)

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

```

```
plt.axis('equal')
plt.tight_layout()
plt.show()
```

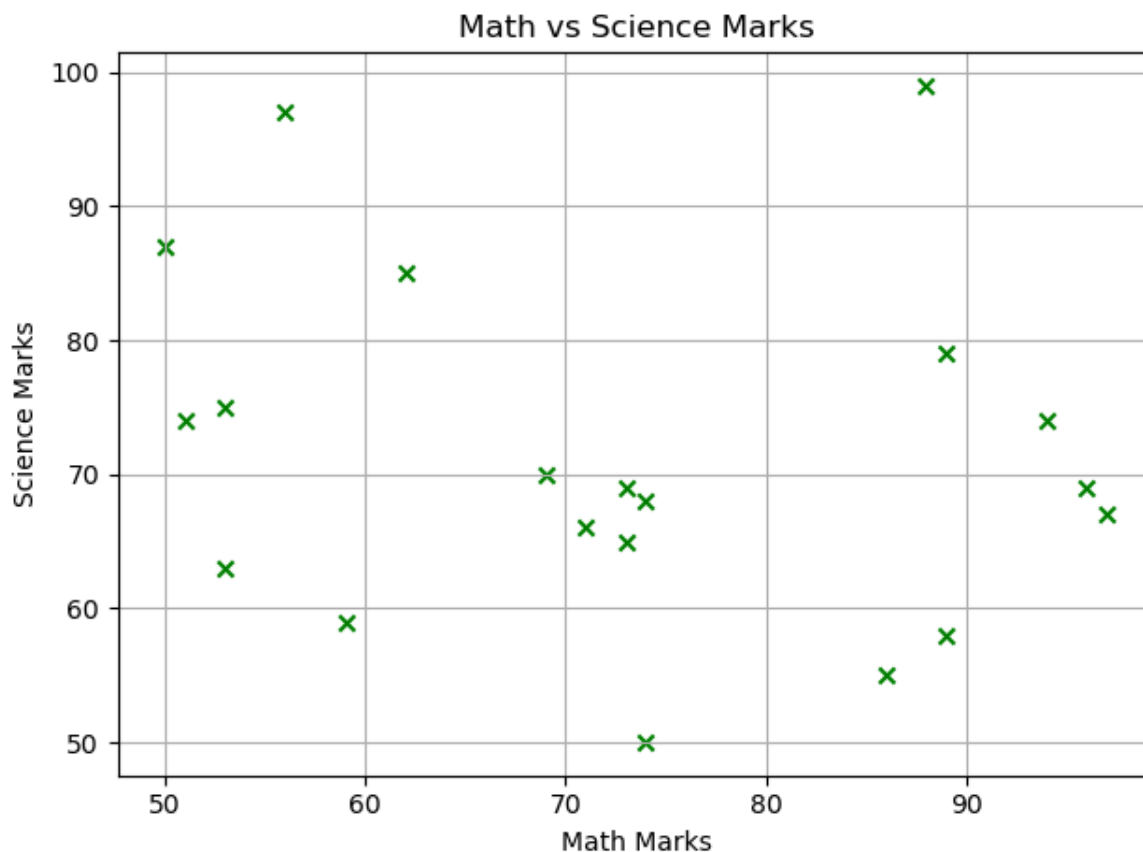


In []: Q4. Scatter Plot (Students' Marks)

```
In [5]: import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

np.random.seed(0)
data = {
    'MathMarks': np.random.randint(50, 100, 20),
    'ScienceMarks': np.random.randint(50, 100, 20)
}
df = pd.DataFrame(data)
df.to_csv('students.csv', index=False)

df = pd.read_csv('students.csv')
plt.scatter(df['MathMarks'], df['ScienceMarks'], color='green', marker='x')
plt.xlabel('Math Marks')
plt.ylabel('Science Marks')
plt.title('Math vs Science Marks')
plt.grid(True)
plt.tight_layout()
plt.show()
```

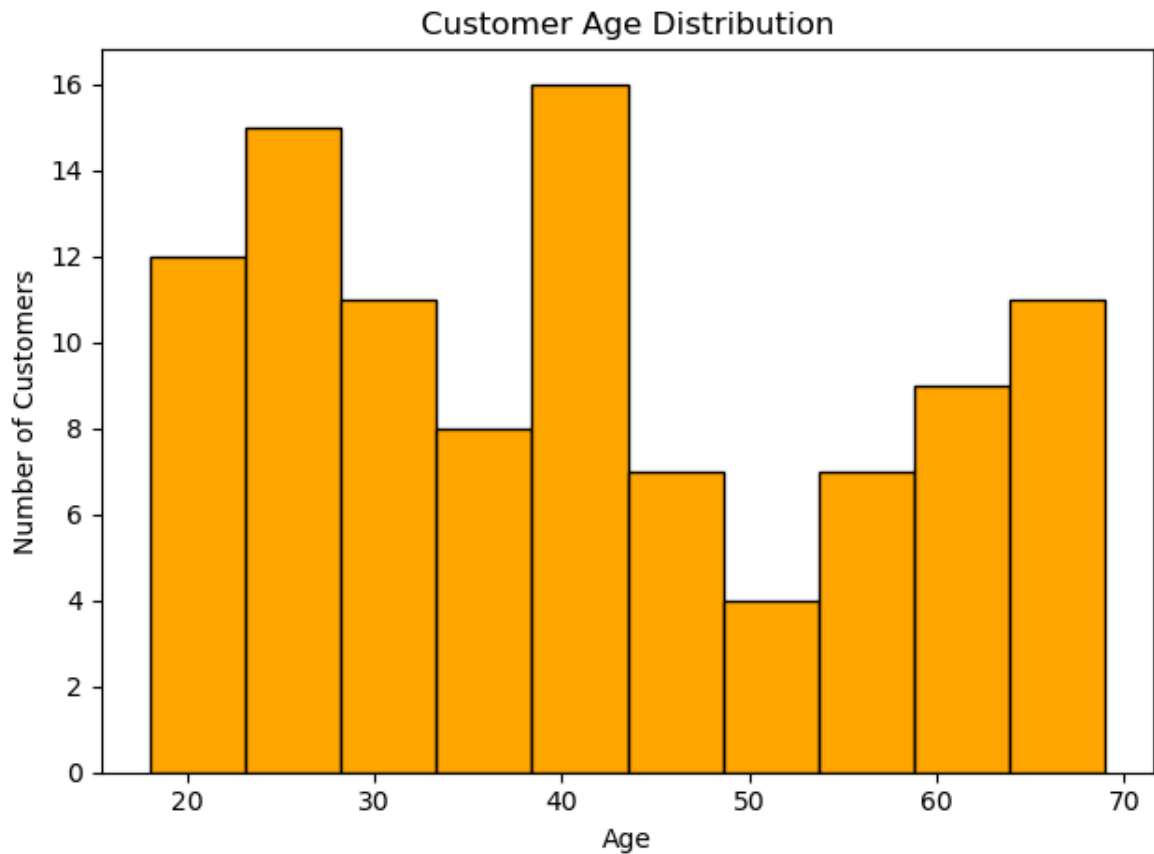


In []: Q5. Histogram (Customer Ages)

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

np.random.seed(1)
df = pd.DataFrame({'Age': np.random.randint(18, 70, 100)})
df.to_csv('customers.csv', index=False)

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

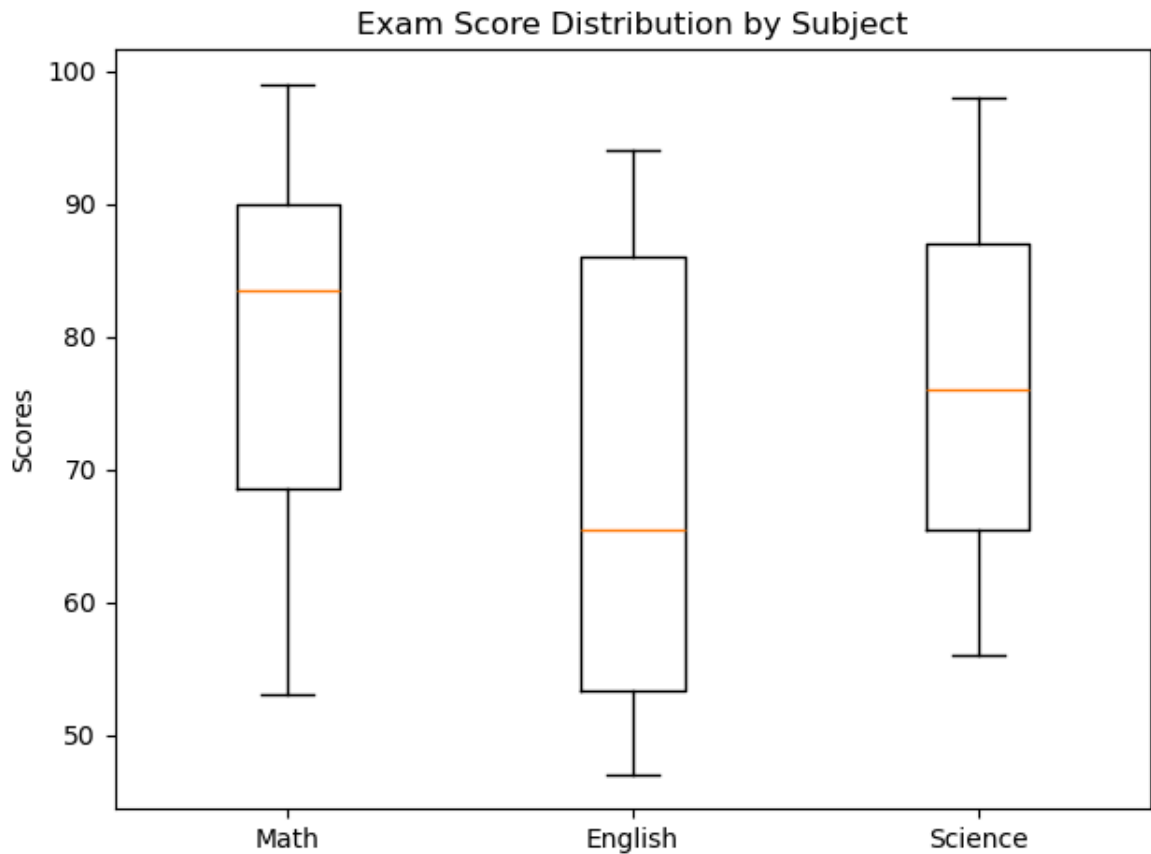


In []: Q6. Boxplot (Exam Scores Comparison)

```
In [9]: import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

np.random.seed(2)
df = pd.DataFrame({
    'Math': np.random.randint(50, 100, 30),
    'English': np.random.randint(45, 95, 30),
    'Science': np.random.randint(55, 100, 30)
})
df.to_csv('exam_scores.csv', index=False)

df = pd.read_csv('exam_scores.csv')
plt.boxplot(
    [df['Math'], df['English'], df['Science']],
    tick_labels=['Math', 'English', 'Science']
)
plt.ylabel('Scores')
plt.title('Exam Score Distribution by Subject')
plt.tight_layout()
plt.show()
```

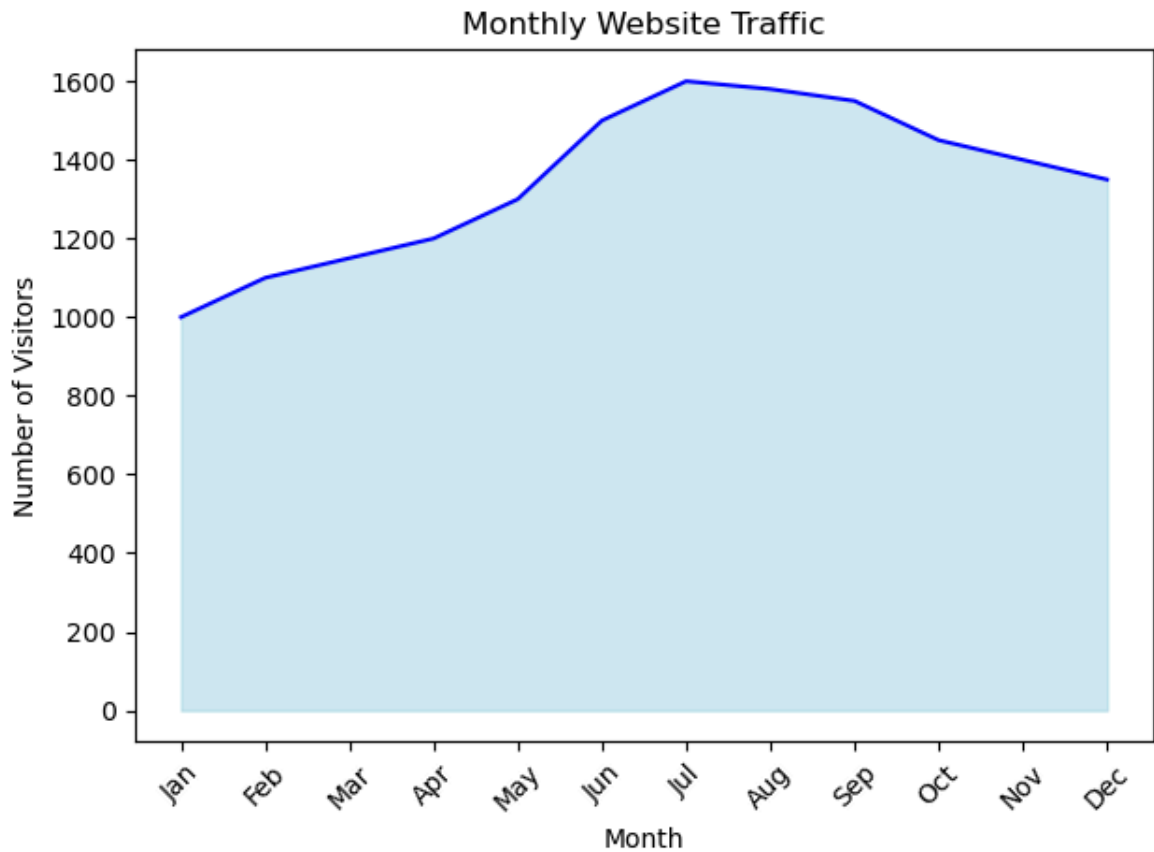



In []: Q7. Area Chart (Website Traffic)

```
In [10]: import pandas as pd
import matplotlib.pyplot as plt

data = {
    'Month': ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec'],
    'Visitors': [1000, 1100, 1150, 1200, 1300, 1500, 1600, 1580, 1550, 1450, 1400, 1350]
}
df = pd.DataFrame(data)
df.to_csv('traffic.csv', index=False)

df = pd.read_csv('traffic.csv')
plt.fill_between(df['Month'], df['Visitors'], color='lightblue', alpha=0.6)
plt.plot(df['Month'], df['Visitors'], color='blue')
plt.xlabel('Month')
plt.ylabel('Number of Visitors')
plt.title('Monthly Website Traffic')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



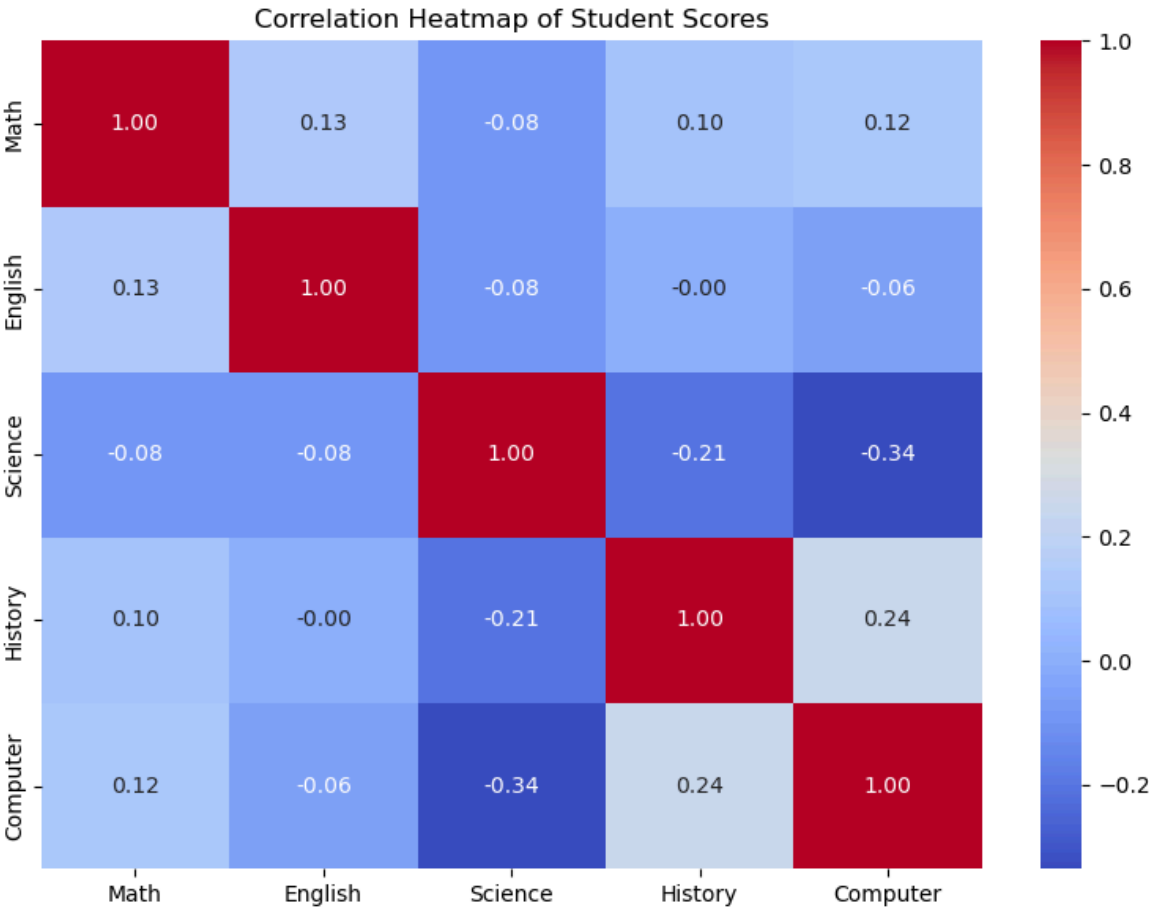
In []: Q8. Heatmap (Correlation Matrix)

```
In [11]: import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np

np.random.seed(3)
df = pd.DataFrame({
    'Math': np.random.randint(60, 100, 30),
    'English': np.random.randint(55, 95, 30),
    'Science': np.random.randint(65, 100, 30),
    'History': np.random.randint(50, 90, 30),
    'Computer': np.random.randint(70, 100, 30)
})
df.to_csv('students_scores.csv', index=False)

df = pd.read_csv('students_scores.csv')
corr = df.corr()

plt.figure(figsize=(8, 6))
sns.heatmap(corr, annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Correlation Heatmap of Student Scores')
plt.tight_layout()
plt.show()
```



In []: 1) A retail company has provided a CSV file (sales_data.csv) containing sales transactions with the following columns: Date, Product, Quantity, Sales, and Region. As a Data Analyst, you are to perform an Exploratory Data Analysis (EDA) using Python (Pandas, NumPy, Matplotlib, 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 of Sales for 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 heatmap.

Discuss the insights obtained from each step.

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

data = {
    'Date': pd.date_range(start='2024-01-01', periods=15, freq='D'),
    'Product': ['A', 'B', 'C', 'A', 'B', 'C', 'A', 'B', 'C', 'A', 'B', 'C', 'A', 'B', 'C'],
    'Quantity': [10, 15, 20, 12, 18, 25, 8, 14, 22, 11, 16, 19, 9, 13, 21],
    'Sales': [1000, 1500, 2000, 1100, 1600, 2100, 900, 1400, 2200, 1050, 1550, 1950, 950, 1450, 2050],
    'Region': ['North', 'South', 'East', 'West', 'North', 'South', 'East', 'West', 'North', 'South', 'East', 'West', 'North', 'South', 'East']
}
df = pd.DataFrame(data)

df.loc[2, 'Sales'] = np.nan
df.loc[5, 'Product'] = np.nan

df['Sales'] = df['Sales'].fillna(df['Sales'].mean())
df = df.dropna(subset=['Product', 'Quantity', 'Region'])
print(df.isnull().sum())

print(df.describe())

product_summary = df.groupby('Product', as_index=False).agg({'Sales': 'sum', 'Quantity': 'sum'})
print(product_summary)

plt.figure(figsize=(8, 5))
sns.barplot(data=product_summary, x='Product', y='Sales', hue='Product', palette='magma')
plt.title('Total Sales by Product', fontsize=14, color='blue')
plt.xlabel('Product')
plt.ylabel('Total Sales')
plt.show()

df['Date'] = pd.to_datetime(df['Date'])
sales_over_time = df.groupby('Date', as_index=False)['Sales'].sum()

plt.figure(figsize=(10, 5))
sns.lineplot(data=sales_over_time, x='Date', y='Sales', color='green')
plt.title('Total Sales Over Time', fontsize=14, color='darkgreen')
plt.xlabel('Date')
plt.ylabel('Sales')
plt.show()
```

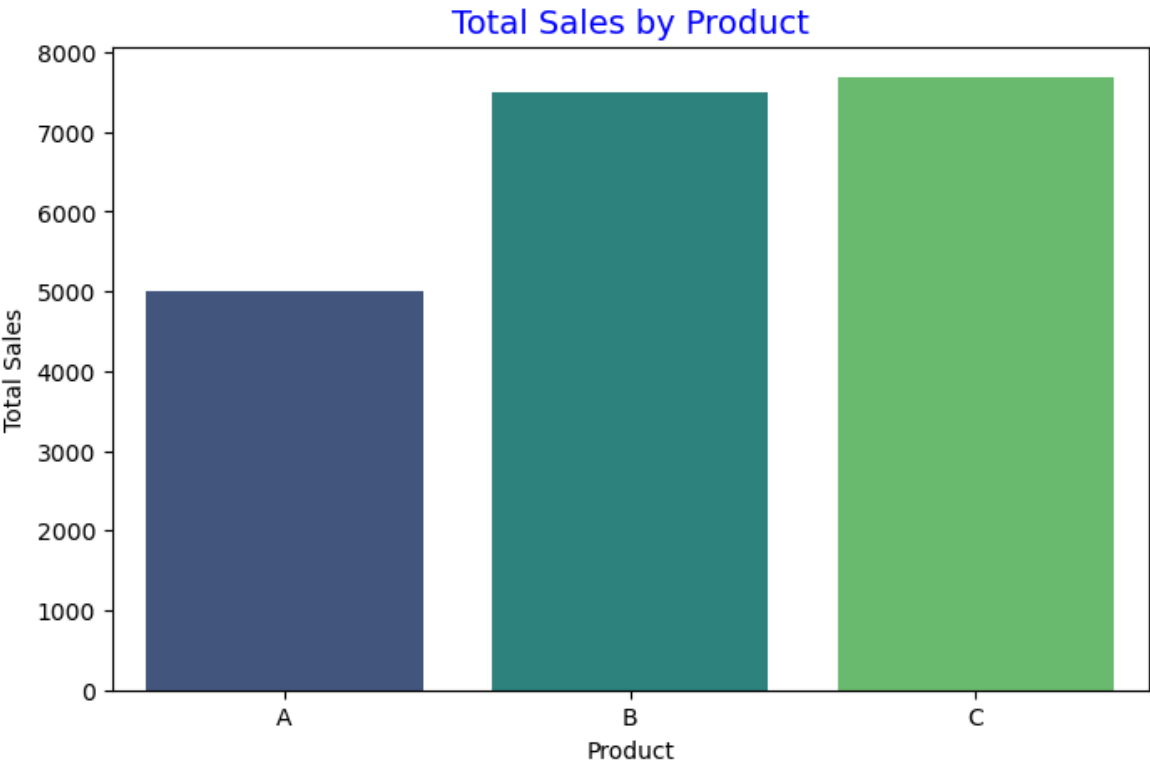
```
pivot_table = pd.pivot_table(df, values='Sales', index='Region', columns='Product')
print(pivot_table)

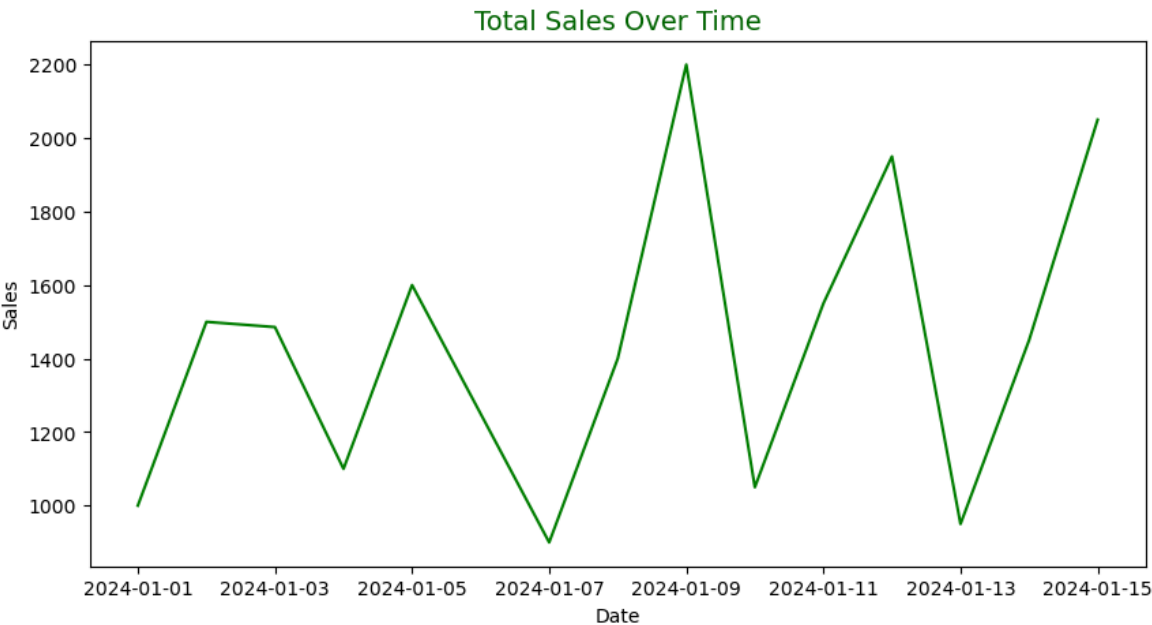
plt.figure(figsize=(6,4))
sns.heatmap(df.corr(numeric_only=True), annot=True, cmap='coolwarm')
plt.title('Correlation Matrix')
plt.show()
```

Date	0
Product	0
Quantity	0
Sales	0
Region	0
dtype:	int64

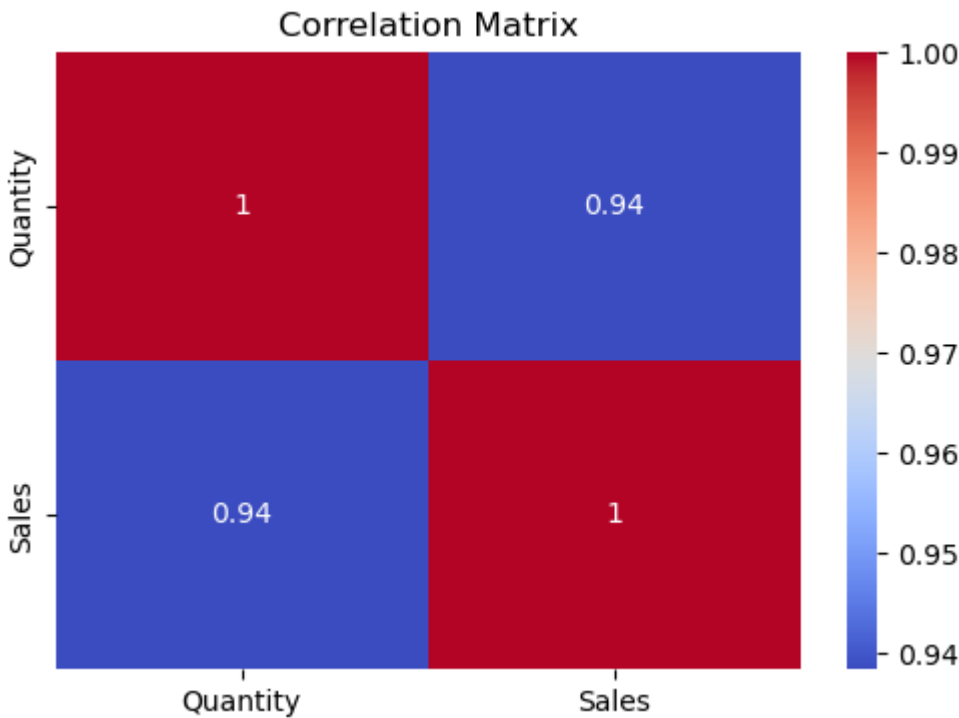
		Date	Quantity	Sales
count		14	14.000000	14.000000
mean	2024-01-08 03:25:42.857142784	14.857143	1441.836735	
min	2024-01-01 00:00:00	8.000000	900.000000	
25%	2024-01-04 06:00:00	11.250000	1062.500000	
50%	2024-01-08 12:00:00	14.500000	1467.857143	
75%	2024-01-11 18:00:00	18.750000	1587.500000	
max	2024-01-15 00:00:00	22.000000	2200.000000	
std		NaN	4.605300	415.825644

	Product	Sales	Quantity
0	A	5000.000000	50
1	B	7500.000000	76
2	C	7685.714286	82





Product	A	B	C
Region			
East	900.0	1550.0	3535.714286
North	1950.0	1600.0	2200.000000
South	1050.0	2950.0	0.000000
West	1100.0	1400.0	1950.000000

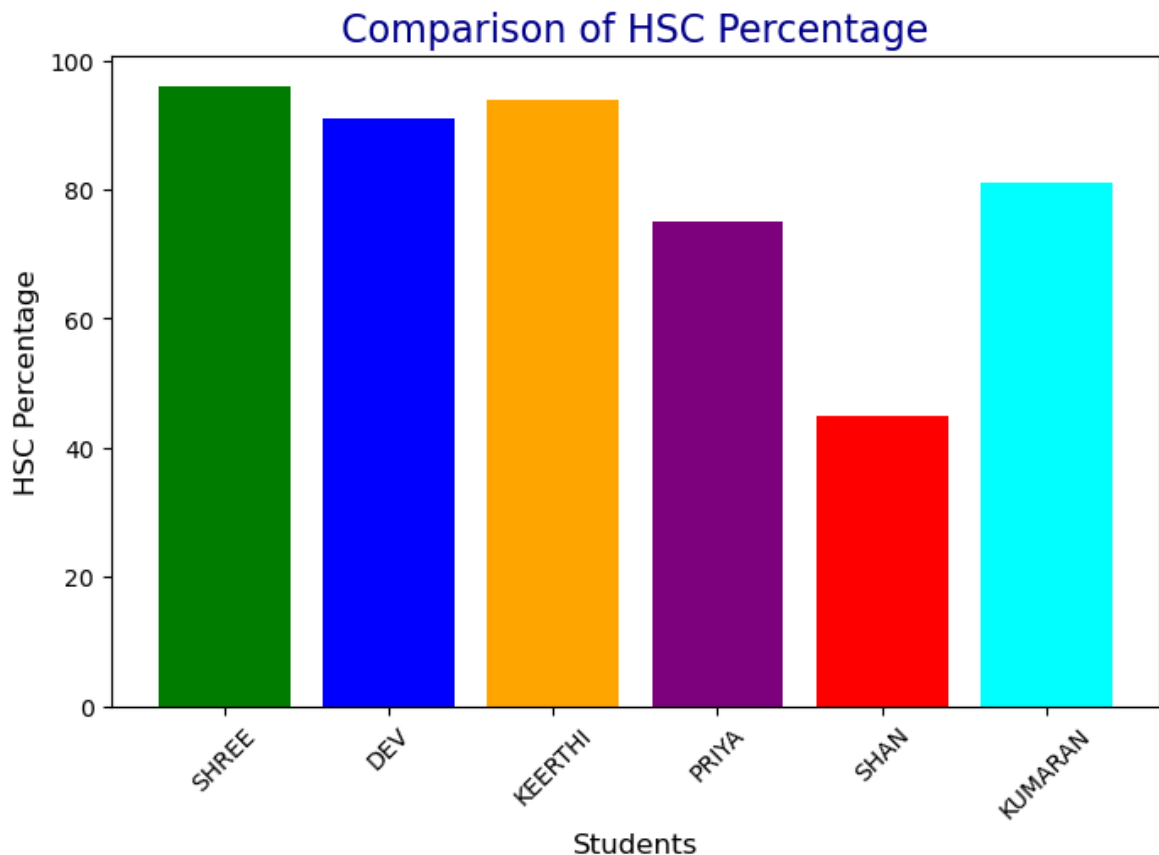


```
In [ ]: 2) A dataset contains the names of six students (SHREE, DEV, KEERTHI, PRIYA, SHA
along with their Higher Secondary Certificate (HSC) exam percentages: 96, 91, 94
Using Matplotlib and NumPy in Python:
Plot a bar chart comparing the HSC percentages of the students.
Set appropriate labels for the x-axis and y-axis.
Rotate the x-axis tick labels for better readability.
Add a title ("Comparison of HSC Percentage") with customized font size and color
Display the chart using show().
Also, interpret the chart to identify the student with the highest and lowest pe
```

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

students = ['SHREE', 'DEV', 'KEERTHI', 'PRIYA', 'SHAN', 'KUMARAN']
percentages = [96, 91, 94, 75, 45, 81]

plt.figure(figsize=(8,5))
plt.bar(students, percentages, color=['green','blue','orange','purple','red','cyan'])
plt.xlabel('Students', fontsize=12)
plt.ylabel('HSC Percentage', fontsize=12)
plt.title('Comparison of HSC Percentage', fontsize=16, color='darkblue')
plt.xticks(rotation=45)
plt.show()
```



```
In [ ]: 3) The following table shows the number of votes received by four candidates in
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").
```

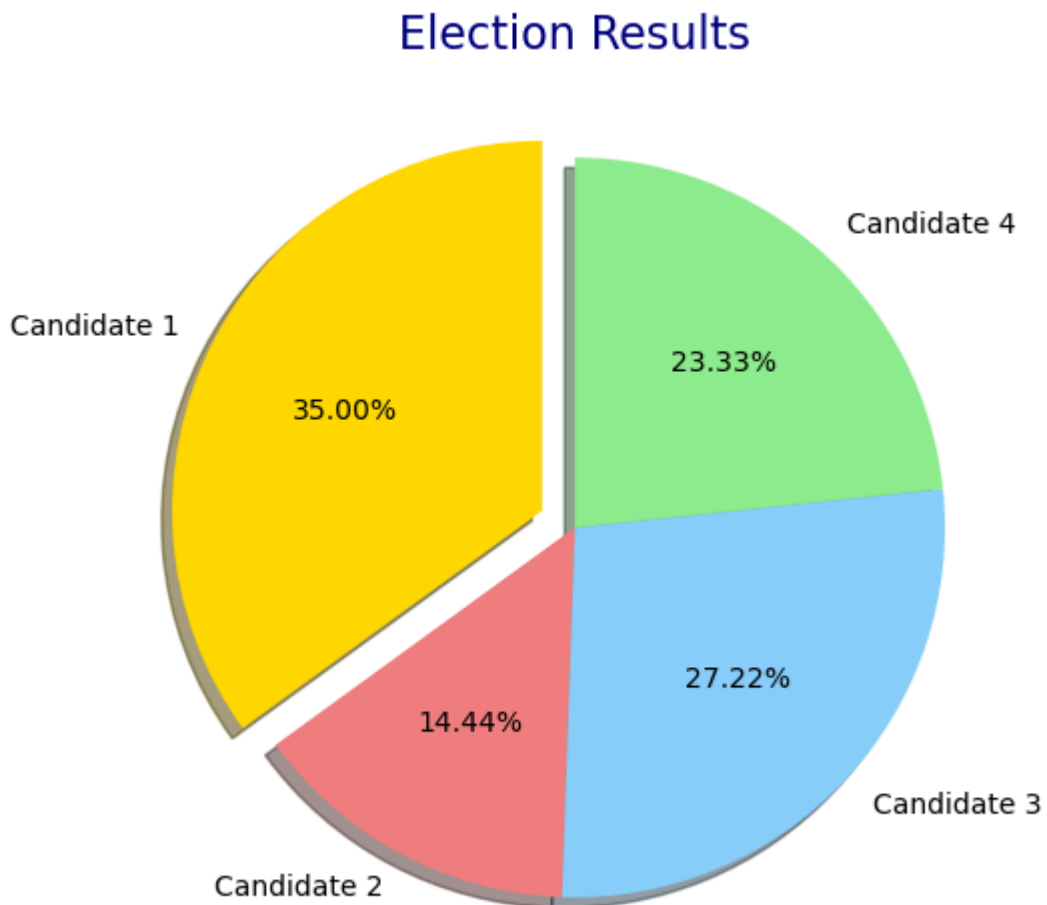
```
In [4]: import matplotlib.pyplot as plt
```

```

candidates = ['Candidate 1', 'Candidate 2', 'Candidate 3', 'Candidate 4']
votes = [315, 130, 245, 210]
colors = ['gold', 'lightcoral', 'lightskyblue', 'lightgreen']
explode = [0.1, 0, 0, 0]

plt.figure(figsize=(6,6))
plt.pie(votes, labels=candidates, autopct='%1.2f%%', startangle=90, colors=colors)
plt.title('Election Results', fontsize=16, color='navy')
plt.show()

```



In []: 4) To Count the frequency of occurrence of a word in a body of text is often need word processing.
Description: Import the word_tokenize function and nltk.

```

In [6]: import nltk
nltk.download('gutenberg')
nltk.download('punkt')

```

```

[nltk_data] Downloading package gutenberg to
[nltk_data]   C:\Users\RISHASRI\AppData\Roaming\nltk_data...
[nltk_data]   Package gutenberg is already up-to-date!
[nltk_data] Downloading package punkt to
[nltk_data]   C:\Users\RISHASRI\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!

```

Out[6]: True

```

In [8]: import nltk
nltk.download('punkt_tab')

```



```
[nltk_data] Downloading package punkt_tab to
[nltk_data]   C:\Users\RISHASRI\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping tokenizers\punkt_tab.zip.
```

Out[8]: True

```
In [9]: from nltk.tokenize import word_tokenize
        from nltk.corpus import gutenberg

        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))))
```

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), ('unite', 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)]
```

In []: 1) Write a Python program to collect, load, and perform initial exploration of the dataset using Pandas. Display the first few records of the dataset and summarize structure and contents.

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

data = {
    "Pregnancies": [2, 5, 3, 1, 4, 6, 8, 0, 2, 7],
    "Glucose": [120, 150, 85, 95, 130, 140, 155, 100, 110, 160],
    "BloodPressure": [70, 80, 60, 75, 78, 85, 90, 72, 68, 88],
    "SkinThickness": [35, 29, 30, 32, 28, 25, 31, 27, 33, 36],
    "Insulin": [100, 130, 85, 90, 120, 140, 150, 95, 105, 160],
    "BMI": [33.6, 28.1, 26.2, 30.5, 27.8, 29.4, 31.1, 25.7, 32.3, 34.8],
    "DiabetesPedigreeFunction": [0.627, 0.351, 0.672, 0.245, 0.587, 0.478, 0.513, 0.167, 0.431, 0.354],
    "Age": [50, 31, 29, 41, 35, 45, 55, 28, 38, 60],
    "Outcome": [1, 0, 0, 0, 1, 1, 1, 0, 0, 1]
}

df = pd.DataFrame(data)
df.to_csv("diabetes.csv", index=False)
print("diabetes.csv created successfully!\n")

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()
```

diabetes.csv created successfully!

First 5 rows of the dataset:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	\
0	2	120	70	35	100	33.6	
1	5	150	80	29	130	28.1	
2	3	85	60	30	85	26.2	
3	1	95	75	32	90	30.5	
4	4	130	78	28	120	27.8	

	DiabetesPedigreeFunction	Age	Outcome
0	0.627	50	1
1	0.351	31	0
2	0.672	29	0
3	0.245	41	0
4	0.587	35	1

Dataset Information:

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

RangeIndex: 10 entries, 0 to 9

Data columns (total 9 columns):

#	Column	Non-Null Count	Dtype
0	Pregnancies	10 non-null	int64
1	Glucose	10 non-null	int64
2	BloodPressure	10 non-null	int64
3	SkinThickness	10 non-null	int64
4	Insulin	10 non-null	int64
5	BMI	10 non-null	float64
6	DiabetesPedigreeFunction	10 non-null	float64
7	Age	10 non-null	int64
8	Outcome	10 non-null	int64

dtypes: float64(2), int64(7)

memory usage: 852.0 bytes

None

Statistical Summary:

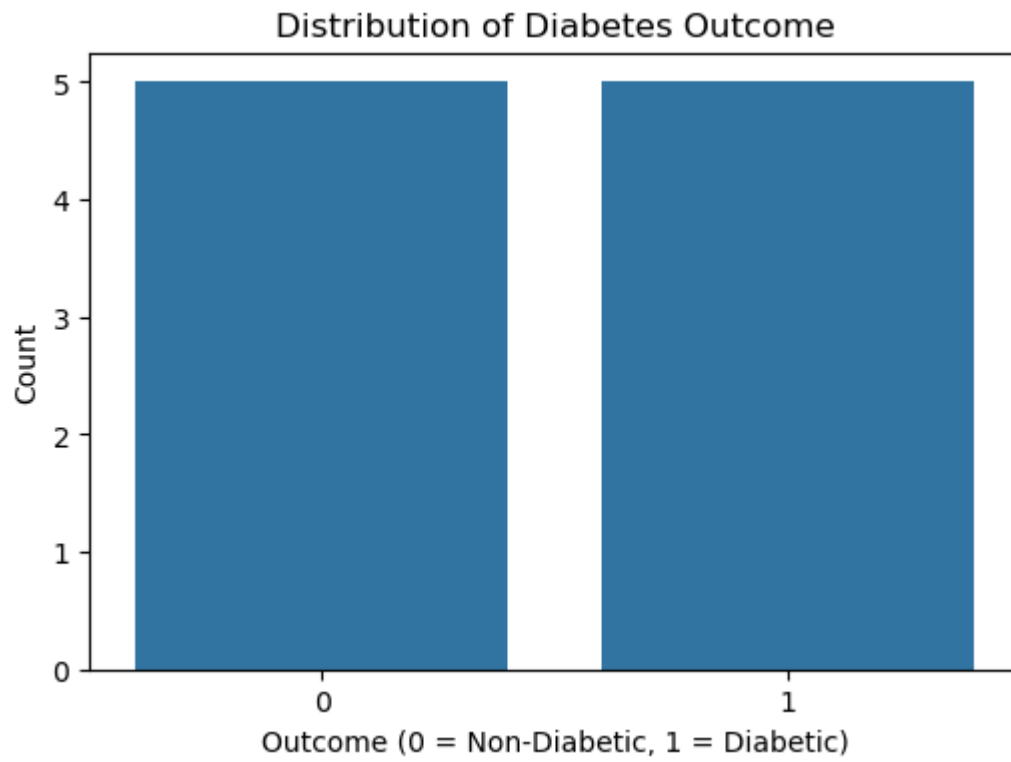
	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	\
count	10.00000	10.000000	10.000000	10.00000	10.000000	
mean	3.80000	124.500000	76.600000	30.60000	117.500000	
std	2.65832	26.609313	9.489175	3.50238	26.483748	
min	0.00000	85.000000	60.000000	25.00000	85.000000	
25%	2.00000	102.500000	70.500000	28.25000	96.250000	
50%	3.50000	125.000000	76.500000	30.50000	112.500000	
75%	5.75000	147.500000	83.750000	32.75000	137.500000	
max	8.00000	160.000000	90.000000	36.00000	160.000000	

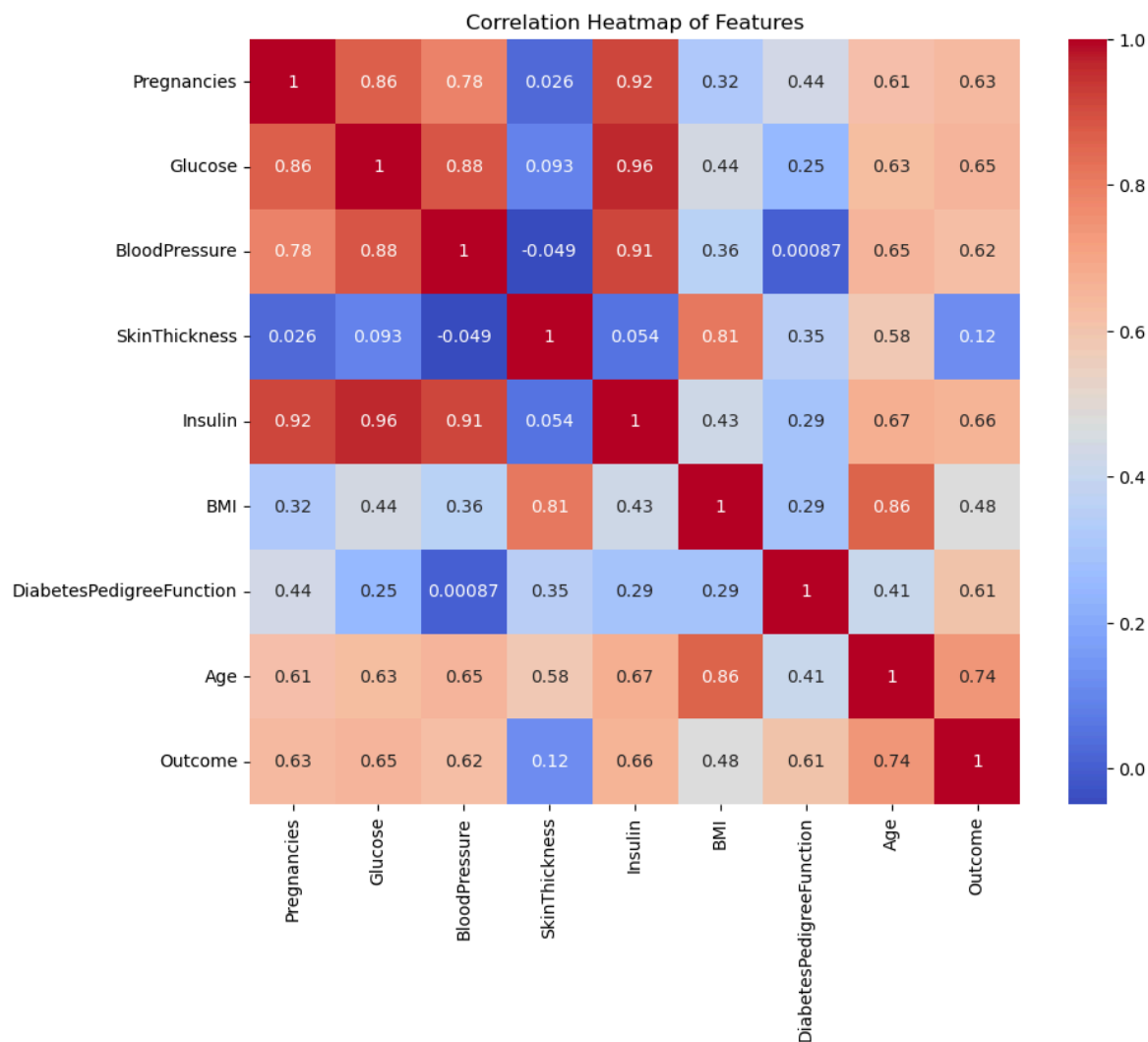
	BMI	DiabetesPedigreeFunction	Age	Outcome
count	10.000000	10.000000	10.000000	10.000000
mean	29.950000	0.490600	41.200000	0.500000
std	3.063495	0.153845	11.113555	0.527046
min	25.700000	0.245000	28.000000	0.000000
25%	27.875000	0.366750	32.000000	0.000000
50%	29.950000	0.495500	39.500000	0.500000
75%	32.000000	0.617000	48.750000	1.000000
max	34.800000	0.690000	60.000000	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
```





In []: 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 trend and seasonality, visualize key metrics using moving averages and seasonal plots, and detect anomalies using statistical methods.

```
In [2]: 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

dates = pd.date_range(start='2024-01-01', periods=100, freq='D')
np.random.seed(42)
page_views = np.random.randint(1000, 5000, size=100) + np.linspace(0, 2000, 100)
unique_visitors = page_views * np.random.uniform(0.6, 0.9, size=100)
bounce_rate = np.random.uniform(40, 70, size=100)

df = pd.DataFrame({
    'Date': dates,
    'Page_Views': page_views,
    'Unique_Visitors': unique_visitors,
    'Bounce_Rate': bounce_rate
})

df = df.sort_values('Date')
df.set_index('Date', inplace=True)
```

```

df = df.interpolate()

print("First few rows of the dataset:")
print(df.head())

print("\nDataset Information:")
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()

```

First few rows of the dataset:

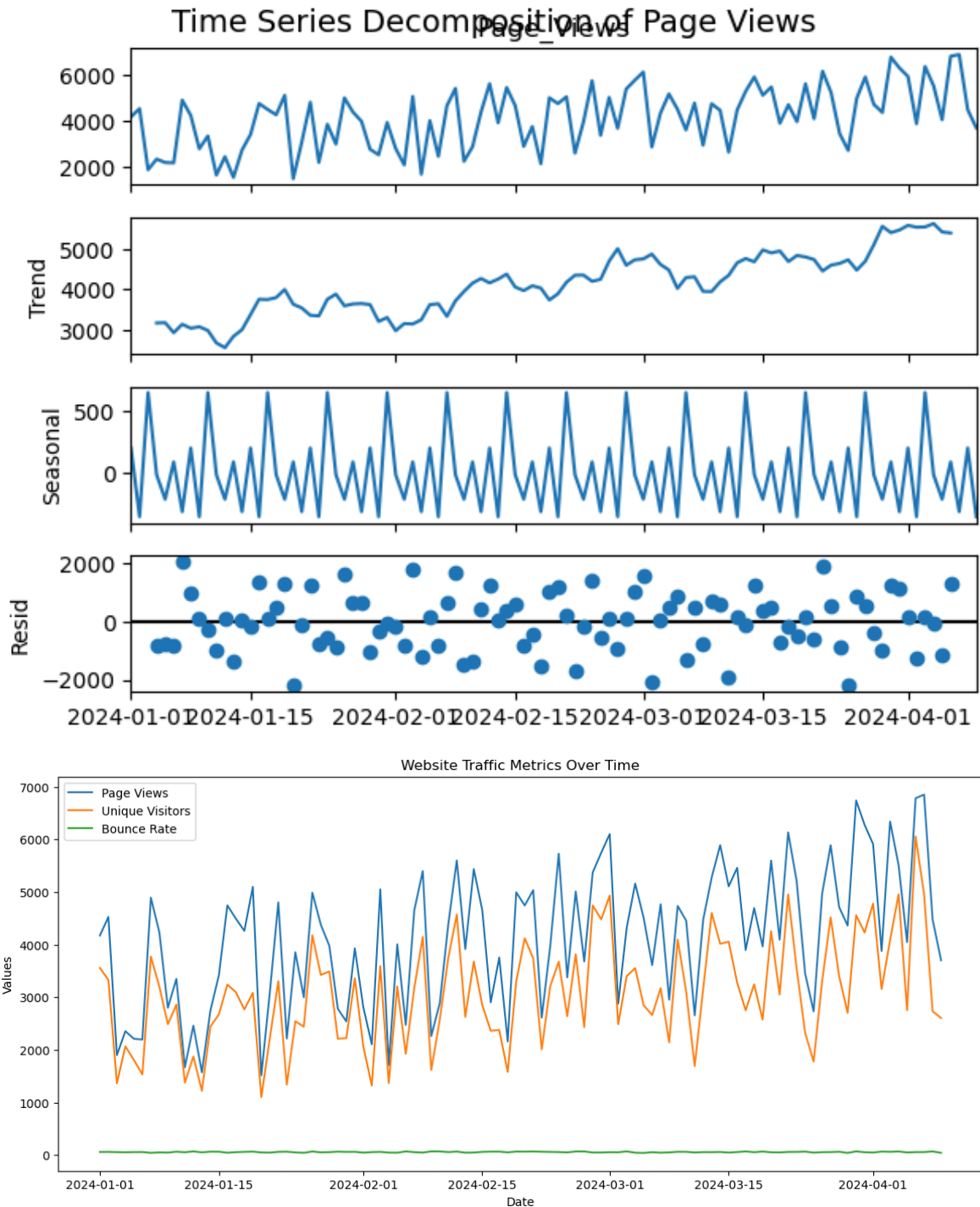
	Page_Views	Unique_Visitors	Bounce_Rate
Date			
2024-01-01	4174.000000	3559.108995	59.030540
2024-01-02	4527.202020	3327.159558	60.421164
2024-01-03	1900.404040	1365.525956	55.928037
2024-01-04	2354.606061	2067.338611	53.433495
2024-01-05	2210.808081	1808.842490	56.586793

Dataset Information:

```

<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 100 entries, 2024-01-01 to 2024-04-09
Data columns (total 3 columns):
 #   Column          Non-Null Count  Dtype
---  -
 0   Page_Views      100 non-null   float64
 1   Unique_Visitors 100 non-null   float64
 2   Bounce_Rate     100 non-null   float64
dtypes: float64(3)
memory usage: 3.1 KB
None
<Figure size 1200x800 with 0 Axes>

```



In []: 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 trend and seasonality, visualize key metrics using moving averages and seasonal plots, and detect anomalies using statistical methods.

```
In [4]: 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

dates = pd.date_range(start='2023-01-01', periods=60, freq='D')
np.random.seed(42)
page_views = np.random.poisson(lam=200, size=60) + (np.sin(np.linspace(0, 3*np.pi, 60)) * 100)
unique_visitors = page_views - np.random.randint(10, 50, size=60)
```

```

bounce_rate = np.random.uniform(40, 70, size=60)

data = {
    'Date': dates,
    'Page_Views': page_views,
    'Unique_Visitors': unique_visitors,
    'Bounce_Rate': bounce_rate
}

df_sample = pd.DataFrame(data)
df_sample.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()

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=2)
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()

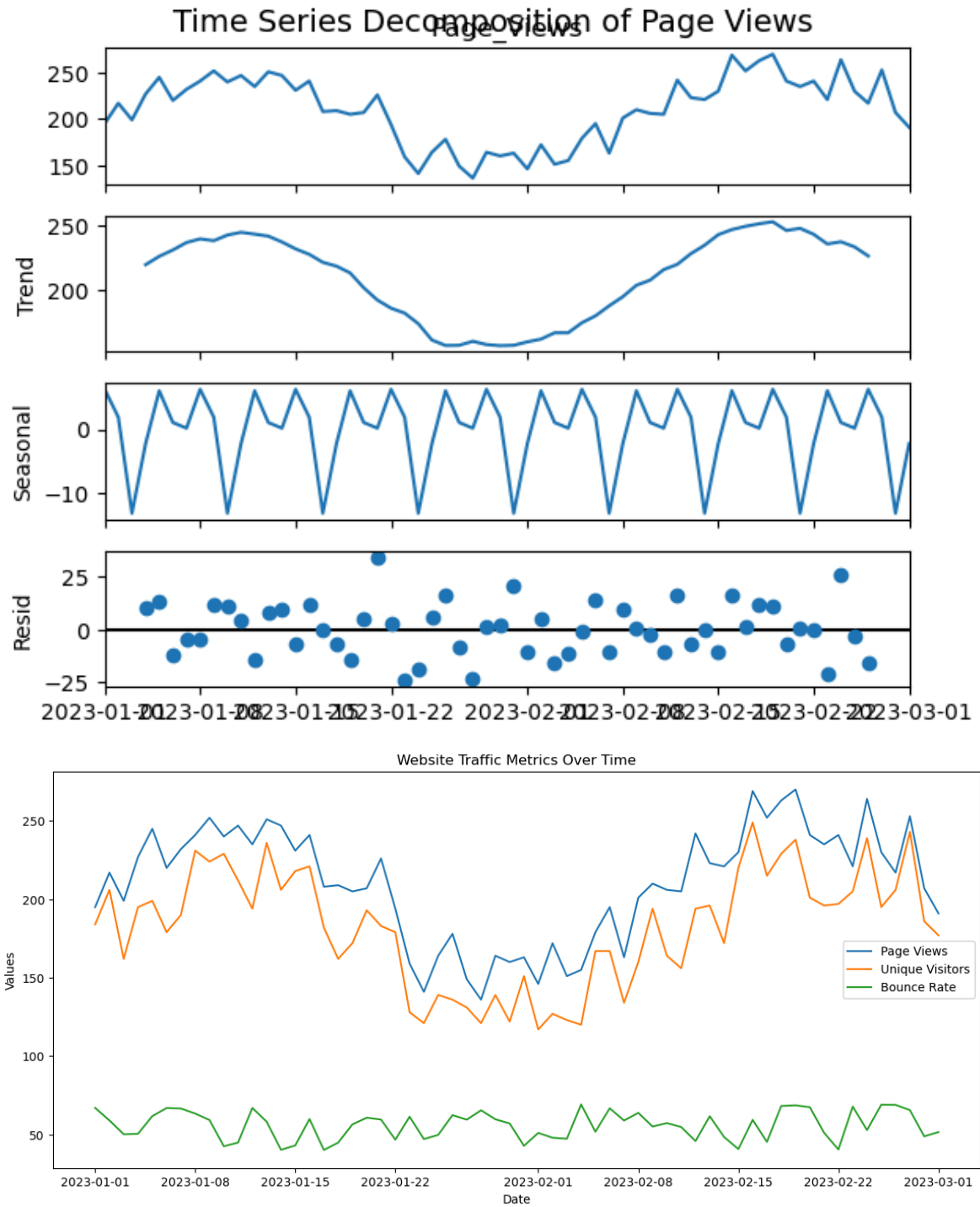
```

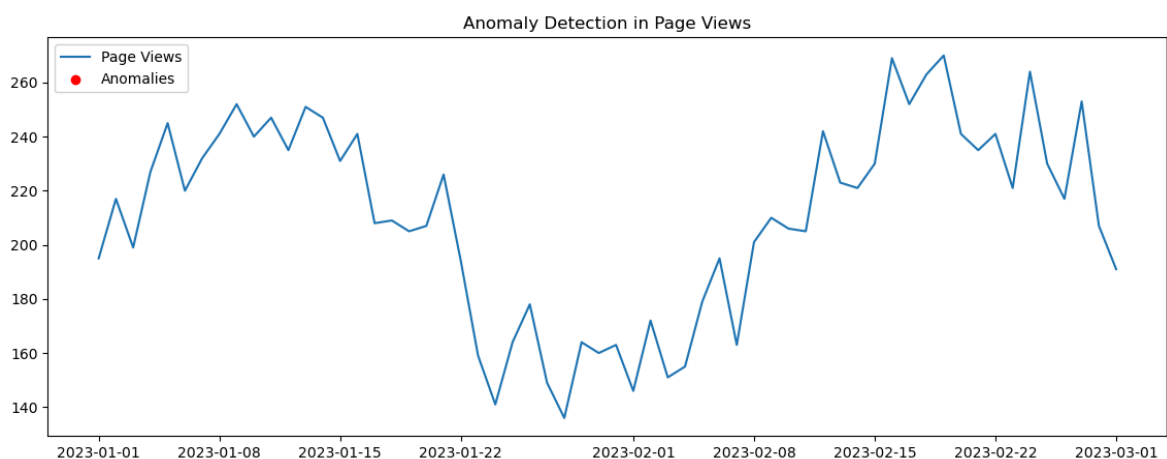
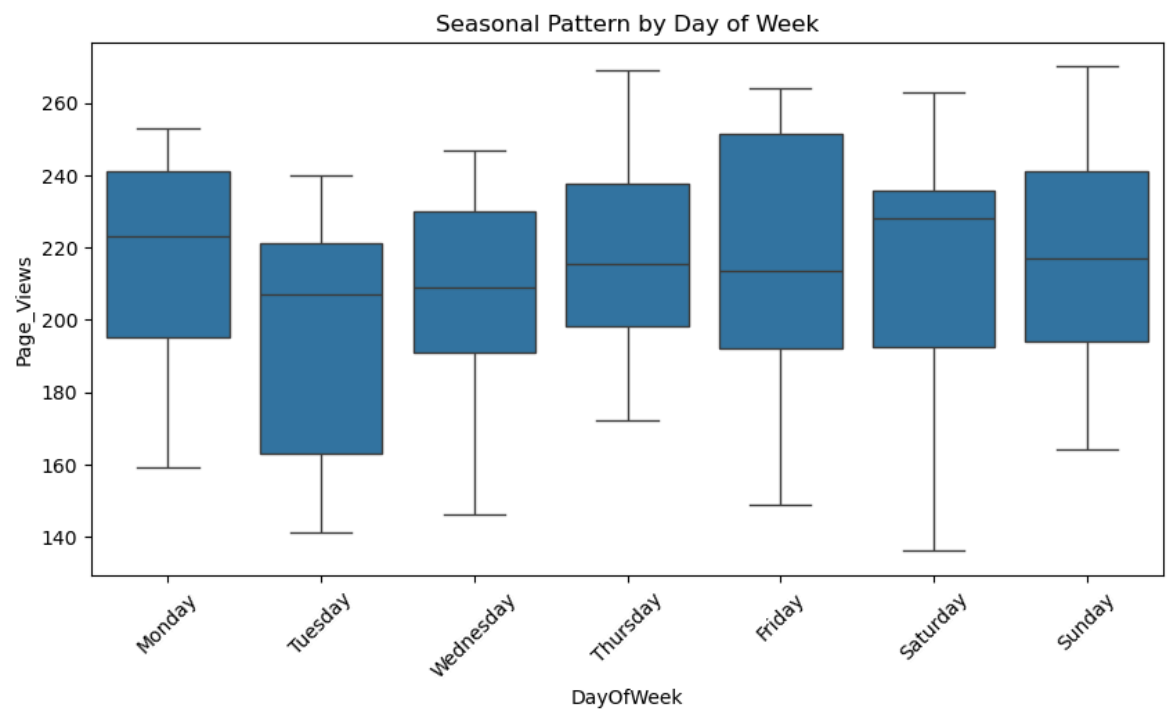
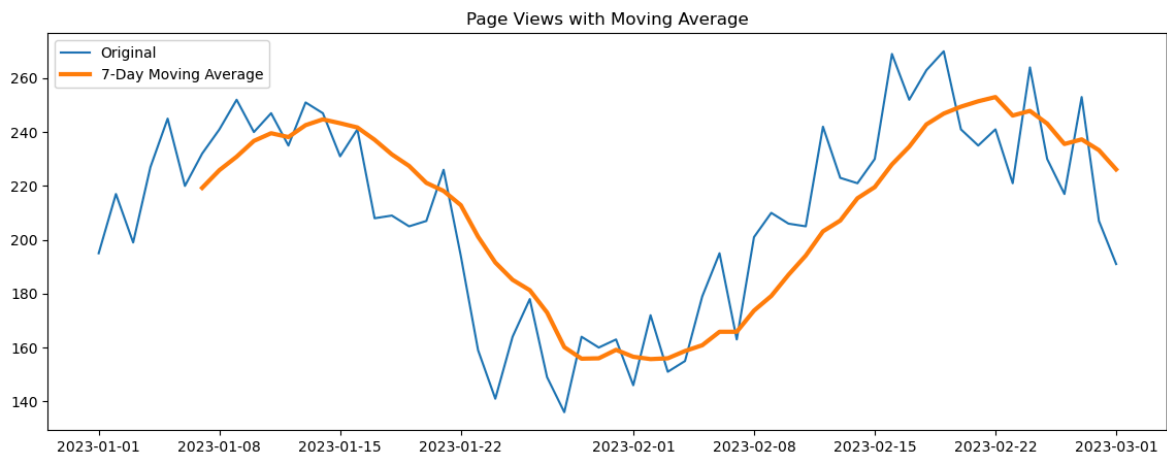


```
plt.show()

print("\nDetected Anomalies:")
print(anomalies[['Page_Views', 'Z_Score']])
```

<Figure size 1200x800 with 0 Axes>





Detected Anomalies:

Empty DataFrame

Columns: [Page_Views, Z_Score]

Index: []

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

```
In [5]: import numpy as np
        import matplotlib.pyplot as plt
```

```

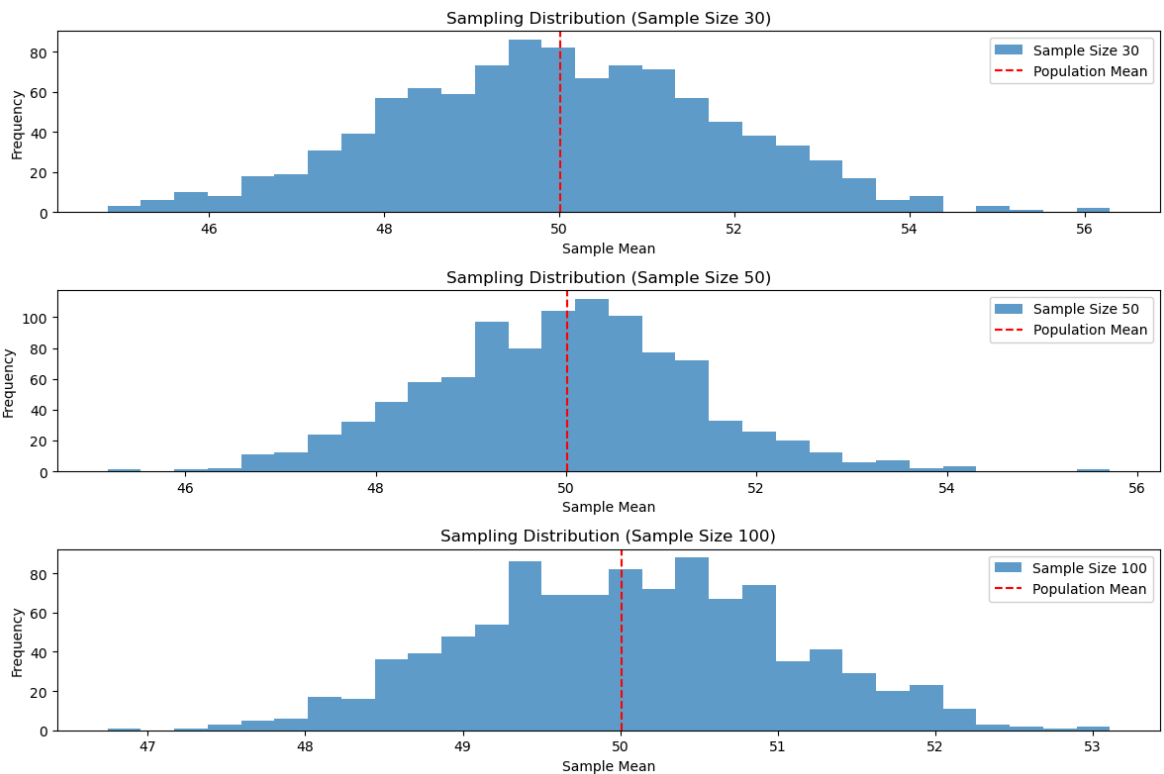
population_mean = 50
population_std = 10
population_size = 100000
population = np.random.normal(population_mean, population_std, population_size)

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=2)
    plt.title(f'Sampling Distribution (Sample Size {size})')
    plt.xlabel('Sample Mean')
    plt.ylabel('Frequency')
    plt.legend()
plt.tight_layout()
plt.show()

```



In []: Lab Experiment 8: 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.

```
In [1]: import numpy as np
import scipy.stats as stats

sample_data = np.array([152,148,151,149,147,153,150,148,152,149,
                        151,150,149,152,151,148,150,152,149,150,
                        148,153,151,150,149,152,148,151,150,153])

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(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")
else:
    print("Fail to reject the null hypothesis")
```

Sample Mean: 150.20

Z-Statistic: 0.6406

P-Value: 0.5218

Fail to reject the null hypothesis

In []: Exp No:9 Experiment to understand Matplotlib library use cases in Data Science through visualization

Description: Use Iris data set to understand the Matplotlib library

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

iris = datasets.load_iris()
iris_df = pd.DataFrame(iris.data, columns=iris.feature_names)
iris_df['species'] = pd.Categorical.from_codes(iris.target, iris.target_names)

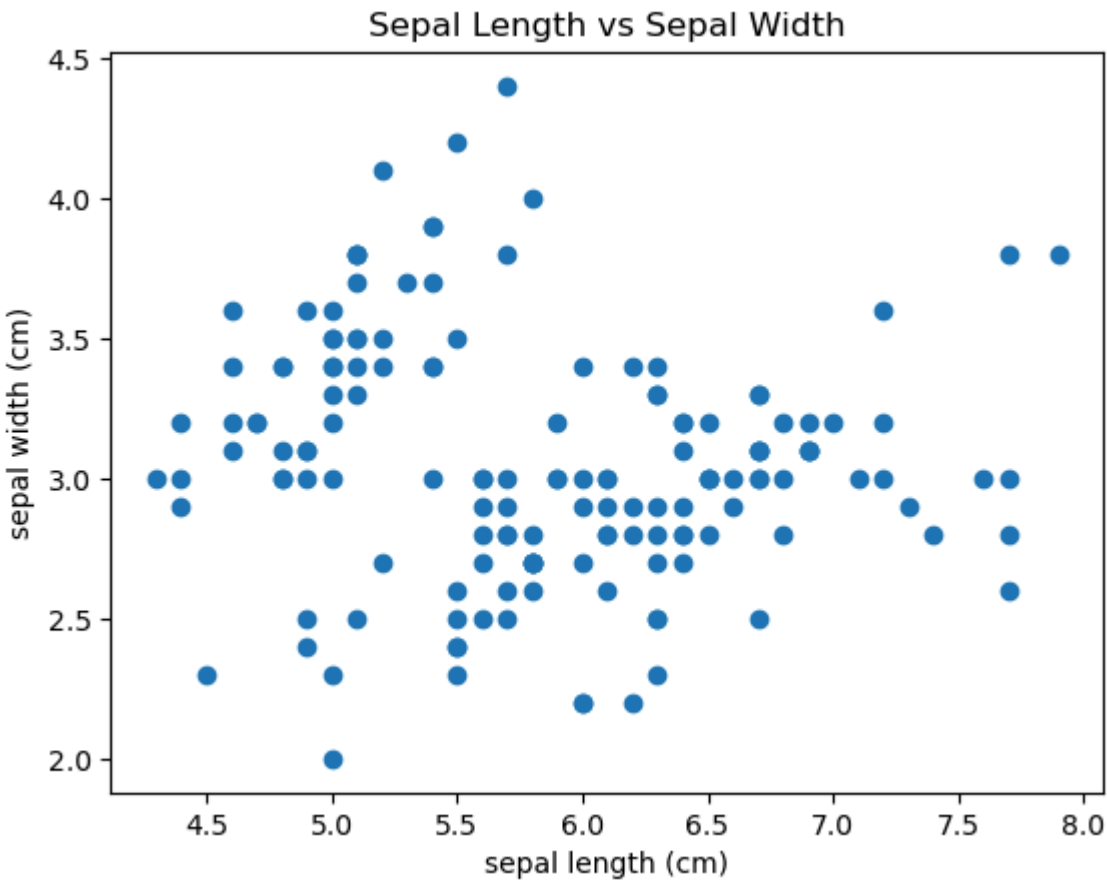
print(iris_df.head())

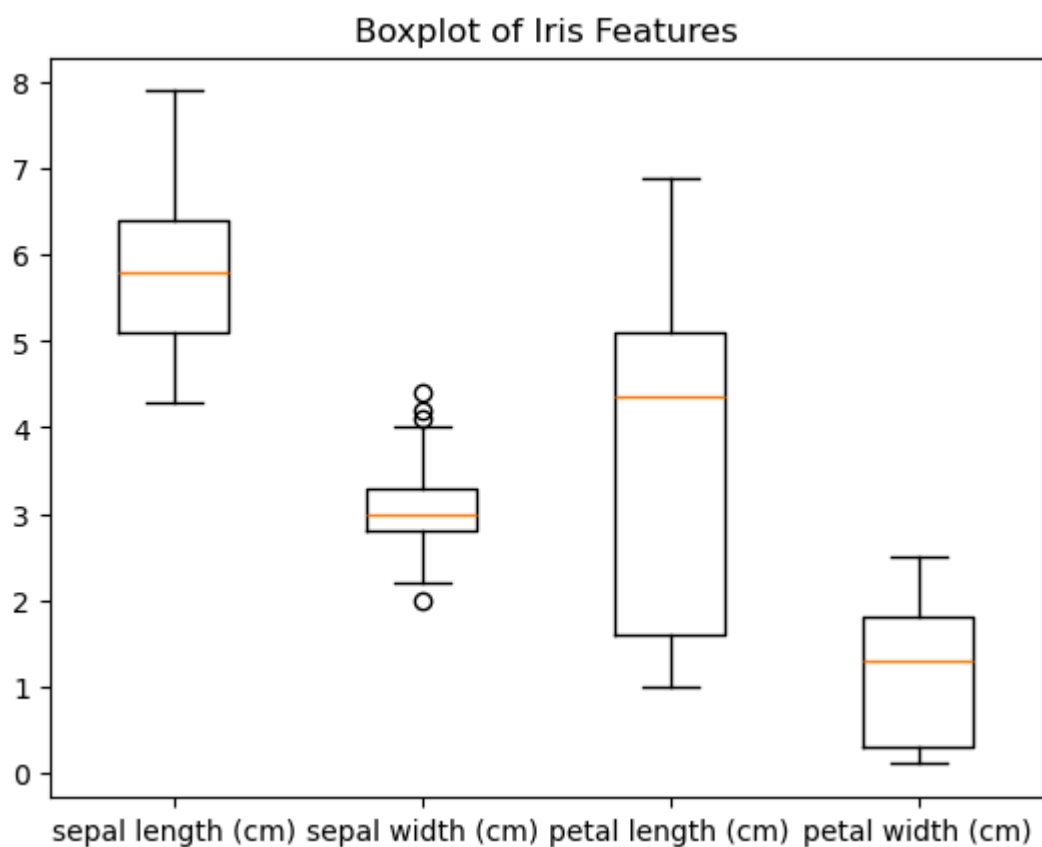
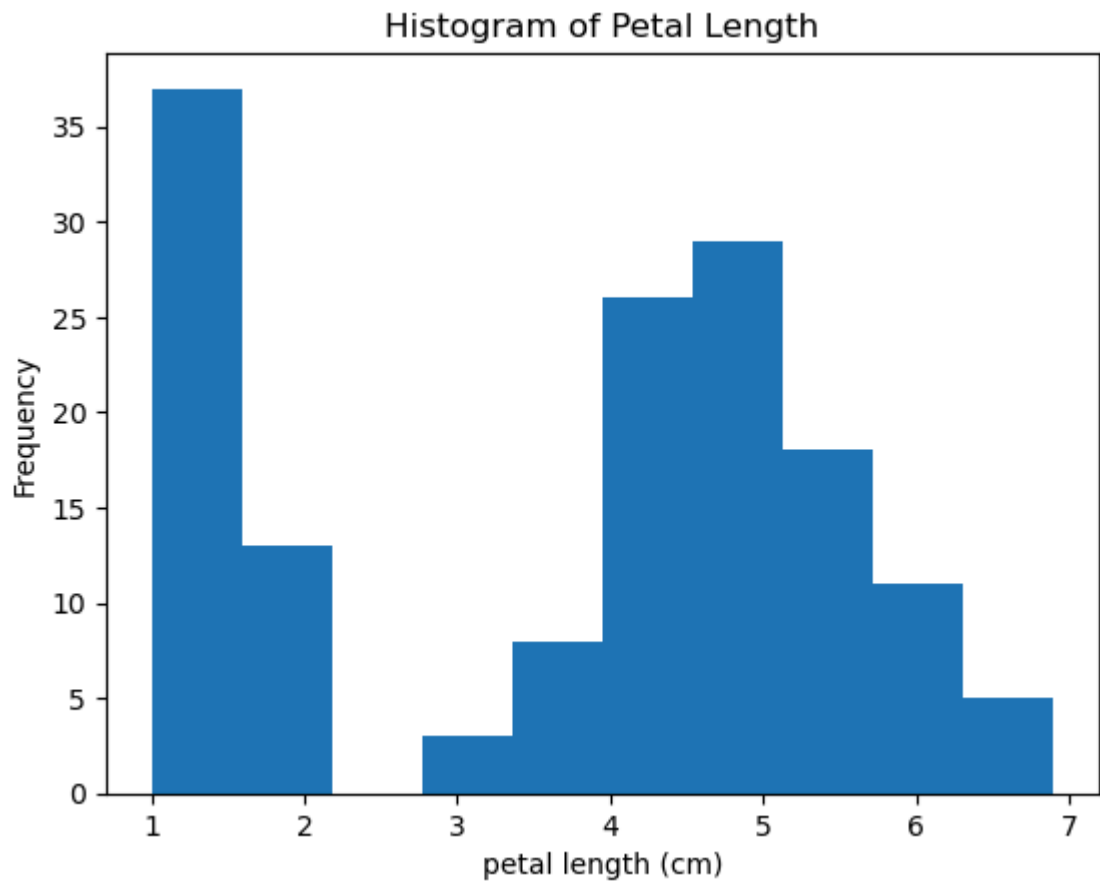
plt.scatter(iris_df['sepal length (cm)'], iris_df['sepal width (cm)'])
plt.title('Sepal Length vs Sepal Width')
plt.xlabel('sepal length (cm)')
plt.ylabel('sepal width (cm)')
plt.show()

plt.hist(iris_df['petal length (cm)'])
plt.title('Histogram of Petal Length')
plt.xlabel('petal length (cm)')
plt.ylabel('Frequency')
plt.show()
```

```
plt.boxplot([iris_df[col] for col in iris.feature_names], tick_labels=iris.featu
plt.title('Boxplot of Iris Features')
plt.show()
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	\
0	5.1	3.5	1.4	0.2	
1	4.9	3.0	1.4	0.2	
2	4.7	3.2	1.3	0.2	
3	4.6	3.1	1.5	0.2	
4	5.0	3.6	1.4	0.2	
	species				
0	setosa				
1	setosa				
2	setosa				
3	setosa				
4	setosa				





In []: Exp No:10 Experiment to understand array function in Data science.
Description: Understand array function using Numpy library

```
In [3]: import numpy as np  
  
arr = np.arange(1,13)
```

```

print("Original array:", arr)
reshaped = arr.reshape(3,4)
print("Reshaped (3x4):\n", reshaped)
print("Transpose:\n", reshaped.T)
print("Mean:", arr.mean())
print("Standard Deviation:", arr.std(ddof=1))
print("Sliced array (2:8):", arr[2:8])
concat = np.concatenate([arr, arr])
print("Concatenated length:", len(concat))

```

Original array: [1 2 3 4 5 6 7 8 9 10 11 12]

Reshaped (3x4):

```

[[ 1  2  3  4]
 [ 5  6  7  8]
 [ 9 10 11 12]]

```

Transpose:

```

[[ 1  5  9]
 [ 2  6 10]
 [ 3  7 11]
 [ 4  8 12]]

```

Mean: 6.5

Standard Deviation: 3.605551275463989

Sliced array (2:8): [3 4 5 6 7 8]

Concatenated length: 24

In []: Exp No:11 Experiment to understand pandas library use cases in Data science.
Description: Understand data frame use cases using pandas library

```

In [5]: import pandas as pd
        from sklearn import datasets

        iris = datasets.load_iris()
        df = pd.DataFrame(iris.data, columns=iris.feature_names)
        df['species'] = pd.Categorical.from_codes(iris.target, iris.target_names)

        print(df.head())
        print("\nSummary Statistics:\n", df.describe())

        grouped = df.groupby('species', observed=False).mean()
        print("\nGroup Mean by Species:\n", grouped)

        other = pd.DataFrame({'species': ['setosa','versicolor','virginica'],
                                'group_code': [1,2,3]})
        merged = pd.merge(df, other, on='species', how='left')
        print("\nMerged DataFrame (first 5 rows):\n", merged.head())

```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	\
0	5.1	3.5	1.4	0.2	
1	4.9	3.0	1.4	0.2	
2	4.7	3.2	1.3	0.2	
3	4.6	3.1	1.5	0.2	
4	5.0	3.6	1.4	0.2	

	species
0	setosa
1	setosa
2	setosa
3	setosa
4	setosa

Summary Statistics:

	sepal length (cm)	sepal width (cm)	petal length (cm)	\
count	150.000000	150.000000	150.000000	
mean	5.843333	3.057333	3.758000	
std	0.828066	0.435866	1.765298	
min	4.300000	2.000000	1.000000	
25%	5.100000	2.800000	1.600000	
50%	5.800000	3.000000	4.350000	
75%	6.400000	3.300000	5.100000	
max	7.900000	4.400000	6.900000	

	petal width (cm)
count	150.000000
mean	1.199333
std	0.762238
min	0.100000
25%	0.300000
50%	1.300000
75%	1.800000
max	2.500000

Group Mean by Species:

	sepal length (cm)	sepal width (cm)	petal length (cm)	\
species				
setosa	5.006	3.428	1.462	
versicolor	5.936	2.770	4.260	
virginica	6.588	2.974	5.552	

	petal width (cm)
species	
setosa	0.246
versicolor	1.326
virginica	2.026

Merged DataFrame (first 5 rows):

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	\
0	5.1	3.5	1.4	0.2	
1	4.9	3.0	1.4	0.2	
2	4.7	3.2	1.3	0.2	
3	4.6	3.1	1.5	0.2	
4	5.0	3.6	1.4	0.2	

	species	group_code
0	setosa	1
1	setosa	1
2	setosa	1


```
3 setosa      1
4 setosa      1
```

In []: Exp No:12 Experiment to detect outliers in a given data set.
Description: Understand the procedure to identify the outliers in a given dataset

```
In [6]: import pandas as pd
import numpy as np
from sklearn import datasets
from scipy import stats

sample_data = pd.Series([152,148,151,149,147,153,150,148,152,149,
                          151,150,149,152,151,148,150,152,149,150,
                          148,153,151,150,149,152,148,151,150,153])

def iqr_outliers(series):
    q1 = series.quantile(0.25)
    q3 = series.quantile(0.75)
    iqr = q3 - q1
    lower = q1 - 1.5 * iqr
    upper = q3 + 1.5 * iqr
    return series[(series < lower) | (series > upper)]

iqr_out = iqr_outliers(sample_data)
print("IQR Outliers:", iqr_out.values)

iris = datasets.load_iris()
iris_df = pd.DataFrame(iris.data, columns=iris.feature_names)
z_scores = stats.zscore(iris_df)
outliers = {}
for i, col in enumerate(iris.feature_names):
    mask = np.abs(z_scores[:, i]) > 3
    if mask.any():
        outliers[col] = iris_df[col][mask].tolist()
print("Z-score Outliers:", outliers)
```

IQR Outliers: []

Z-score Outliers: {'sepal width (cm)': [4.4]}

In []: Experiment 6: Handling Missing and Inappropriate Data

```
In [1]: import pandas as pd
import numpy as np

data = {
    'CustomerID':[1,2,3,4,5,6,7,8,9,9,10],
    'Age_Group':['20-25','30-35','25-30','20-25','35+','35+','35+','20-25','25-30',
    'Rating(1-5)':[4,5,6,-1,3,3,4,7,2,2,5],
    'Hotel':['Ibis','LemonTree','RedFox','LemonTree','Ibis','Ibys','RedFox','Lem
    'FoodPreference':['veg','Non-Veg','Veg','Veg','Vegetarian','Non-Veg','Vegeta
    'Bill':[1300,2000,1322,1234,989,1909,1000,2999,3456,3456,-6755],
    'NoOfPax':[2,3,2,2,2,2,-1,-10,3,3,4],
    'EstimatedSalary':[40000,59000,30000,120000,45000,122220,21122,345673,-99999
    'Age_Group.1':['20-25','30-35','25-30','20-25','35+','35+','35+','20-25','25
}
df=pd.DataFrame(data)
df.to_csv("Hotel_Dataset.csv",index=False)
df=pd.read_csv("Hotel_Dataset.csv")
df=df.drop_duplicates()
df=df.drop(['Age_Group.1'],axis=1)
df.loc[df['Bill']<0,'Bill']=np.nan
df.loc[df['EstimatedSalary']<0,'EstimatedSalary']=np.nan
df.loc[(df['NoOfPax']<1)|(df['NoOfPax']>20),'NoOfPax']=np.nan
df['Hotel']=df['Hotel'].replace({'Ibys':'Ibis'})
df['FoodPreference']=df['FoodPreference'].replace({'veg':'Veg','Vegetarian':'Veg
df['EstimatedSalary']=df['EstimatedSalary'].fillna(round(df['EstimatedSalary'].m
df['NoOfPax']=df['NoOfPax'].fillna(round(df['NoOfPax'].median()))
df['Bill']=df['Bill'].fillna(round(df['Bill'].mean()))
df['Rating(1-5)']=df['Rating(1-5)'].fillna(round(df['Rating(1-5)'].median()))
print(df)
```

	CustomerID	Age_Group	Rating(1-5)	Hotel	FoodPreference	Bill \
0	1	20-25	4	Ibis	Veg	1300.0
1	2	30-35	5	LemonTree	Non-Veg	2000.0
2	3	25-30	6	RedFox	Veg	1322.0
3	4	20-25	-1	LemonTree	Veg	1234.0
4	5	35+	3	Ibis	Veg	989.0
5	6	35+	3	Ibis	Non-Veg	1909.0
6	7	35+	4	RedFox	Veg	1000.0
7	8	20-25	7	LemonTree	Veg	2999.0
8	9	25-30	2	Ibis	Non-Veg	3456.0
10	10	30-35	5	RedFox	Non-Veg	1801.0

	NoOfPax	EstimatedSalary
0	2.0	40000.0
1	3.0	59000.0
2	2.0	30000.0
3	2.0	120000.0
4	2.0	45000.0
5	2.0	122220.0
6	2.0	21122.0
7	2.0	345673.0
8	3.0	96755.0
10	4.0	87777.0

In []: Experiment 14: Feature Scaling

```
In [2]: import pandas as pd
from sklearn.preprocessing import MinMaxScaler, StandardScaler

data={'Age':[20,25,30,35,40], 'Salary':[20000,30000,40000,50000,60000]}
df=pd.DataFrame(data)
scaler=MinMaxScaler()
df['Normalized_Salary']=scaler.fit_transform(df[['Salary']])
std_scaler=StandardScaler()
df['Standardized_Salary']=std_scaler.fit_transform(df[['Salary']])
print(df)
```

	Age	Salary	Normalized_Salary	Standardized_Salary
0	20	20000	0.00	-1.414214
1	25	30000	0.25	-0.707107
2	30	40000	0.50	0.000000
3	35	50000	0.75	0.707107
4	40	60000	1.00	1.414214

In []: Experiment 15: Data Preprocessing in Data Science

```
In [3]: import pandas as pd
from sklearn.preprocessing import LabelEncoder

data={'Name':['A','B','C','D','E'], 'Gender':['Male','Female','Male','Female','Fe
df=pd.DataFrame(data)
df.to_csv("data.csv",index=False)
df=pd.read_csv("data.csv")
df.fillna(df.mean(numeric_only=True),inplace=True)
encoder=LabelEncoder()
df['Gender']=encoder.fit_transform(df['Gender'])
print(df)
```

	Name	Gender	Age	Salary
0	A	1	22	35000
1	B	0	25	40000
2	C	1	28	50000
3	D	0	26	42000
4	E	0	23	38000

In []: Experiment 16: EDA - Quantitative and Qualitative Analysis

```
In [4]: import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

data={'Category':['A','B','A','C','B','C','A','B'], 'Sales':[100,150,120,200,130,
df=pd.DataFrame(data)
df.to_csv("dataset.csv",index=False)
df=pd.read_csv("dataset.csv")
print(df.describe())
print(df['Category'].value_counts())
sns.boxplot(data=df,x='Category',y='Sales')
plt.show()
```

```

Sales
count    8.000000
mean    143.750000
std      35.025501
min     100.000000
25%     117.500000
50%     140.000000
75%     165.000000
max     200.000000

```

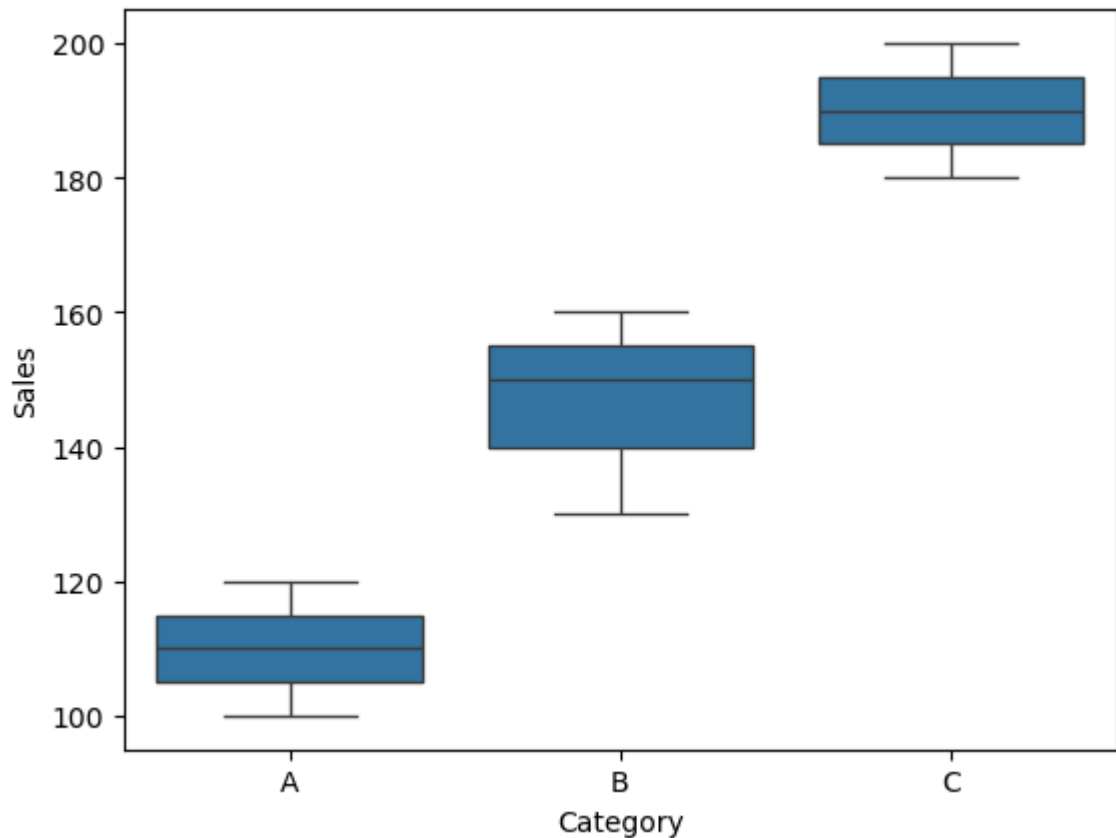
Category

A 3

B 3

C 2

Name: count, dtype: int64



In []: Experiment 17: Linear Regression

```

In [5]: import pandas as pd
        from sklearn.model_selection import train_test_split
        from sklearn.linear_model import LinearRegression

        data={'YearsExperience':[1.1,1.5,2.0,2.2,2.9,3.0,3.2,3.7,3.9,4.0,4.5,4.9,5.1,5.3
                                'Salary':[39343,46205,37731,43525,39891,56642,60150,54445,63218,55794,5695

        df=pd.DataFrame(data)
        df.to_csv("Salary_Data.csv",index=False)
        df=pd.read_csv("Salary_Data.csv")
        X=df[['YearsExperience']]
        y=df['Salary']
        X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=0)
        model=LinearRegression()
        model.fit(X_train,y_train)
        y_pred=model.predict(X_test)
        print(y_pred)

```

[90474.58768564 36770.69366503 92991.95771786 56909.65392276]

In []: Experiment 18: K-Nearest Neighbors (KNN) Algorithm

```
In [6]: from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score

iris=load_iris()
X,y=iris.data,iris.target
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2)
knn=KNeighborsClassifier(n_neighbors=3)
knn.fit(X_train,y_train)
y_pred=knn.predict(X_test)
print("Accuracy:",accuracy_score(y_test,y_pred))
```

Accuracy: 0.9333333333333333

In []: Experiment 19: Logistic Regression

```
In [7]: from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

iris=load_iris()
X,y=iris.data,(iris.target!=0)*1
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2)
model=LogisticRegression()
model.fit(X_train,y_train)
y_pred=model.predict(X_test)
print("Accuracy:",accuracy_score(y_test,y_pred))
```

Accuracy: 1.0

In []: Experiment 20: K-Means Clustering

```
In [9]: import os
os.environ["OMP_NUM_THREADS"] = "1"

from sklearn.cluster import KMeans
from sklearn.datasets import load_iris
import pandas as pd

iris = load_iris()
X = iris.data
kmeans = KMeans(n_clusters=3, random_state=42)
kmeans.fit(X)
print("Cluster Centers:\n", kmeans.cluster_centers_)
print("Labels:\n", kmeans.labels_)
```

Cluster Centers:

```
[[6.85384615 3.07692308 5.71538462 2.05384615]
 [5.006      3.428      1.462      0.246      ]
 [5.88360656 2.74098361 4.38852459 1.43442623]]
```

Labels:

[illegible]

```
C:\Users\RISHASRI\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1419: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
```