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 : T KAUSHIKA

Register Number : 240701239

Create a pandas DataFrame using the following dictionary:

In [3]:
```
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
```

Given a DataFrame with columns Department and Salary, find the average salary per department.

In [5]:
```
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
```

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

In [8]:
```
import pandas as pd

data = {
'Name': ['Alice', 'Bob', 'Charlie'],
'Age': [25, 30, 35],
'City': ['New York', 'Los Angeles', 'Chicago']
}
df = pd.DataFrame(data)
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
```

Replaced all occurrence of 'New york'in city with'NYK'

In [14]:
```
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
```

Drop the Age column from the DataFrame.

In [15]:
```
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(columns=['Age'])
print(df)
```

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

Sort the DataFrame by Salary in descending order.

In [23]:
```
import pandas as pd
import numpy as np
data ={
    'Name' : ['kaushi','alice','bob'],
    'city':['New York','los angeles','china'],
    'Salary':[50000,60000,70000]
}
df = pd.DataFrame(data)
df_sorted =df.sort_values(by='Salary',ascending=False)
print(df_sorted)
```

```
     Name         city  Salary
2     bob        china   70000
1   alice  los angeles   60000
0  kaushi     New York   50000
```

From a DataFrame df, print only the rows where Age is greater than 28.

In [24]:
```
import pandas as pd
data = {
'Name': ['Alice', 'Bob', 'Charlie'],
'Age': [25, 30, 35],
'City': ['New York', 'Los Angeles', 'Chicago']
}
df = pd.DataFrame(data)
filtered_df = df[df['Age']>28]
print(filtered_df)
```

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

Check for missing values in the DataFrame.

In [26]:
```
import pandas as pd
import numpy as np
data ={
    'Name':['bob','kaushi','abc',None],
    'Age':[25,np.nan,35,19],
    'City': [None,'New York', 'Los Angeles', 'Chicago']
}
df=pd.DataFrame(data)
print(df.isnull())
print(df.isnull().sum())
```

```
    Name    Age   City
0  False  False   True
1  False   True  False
2  False  False  False
3   True  False  False
Name    1
Age     1
City    1
dtype: int64
```

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 [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),"\n")
print(list(df.columns))
print(df.shape,"\n")
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: 228.0+ bytes
```

2.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 [2]:
```
import pandas as pd
import numpy as np
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'],"\n")
print(df[['Name','City']],"\n")
print(df.iloc[2],"\n")
print(df.loc[df['Name']=='Charlie'],"\n")
```

```
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    Charlie
Age          22
City    Chicago
Name: 2, dtype: object

      Name  Age     City
2  Charlie   22  Chicago
```

2.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 [3]:
```
import pandas as pd
import numpy as np
data = {
'Name': ['Alice', 'Bob', 'Charlie', 'David'],
'Age': [24, 27, 22, 32],
'City': ['New York', 'Los Angeles', 'Chicago', 'Houston']
}
df=pd.DataFrame(data);
a=df[df['Age']>25]
print(a,"\n")
b=df[(df['City']=='Chicago')|(df['City']=='Houston')]
print(b,"\n")
c=df[(df['Age']>23)&(df['Age']<30)]
print(c,"\n")
```

```
    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
```

4. 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 [4]:
```
import pandas as pd
import numpy as np
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,"\n")
df.loc[1,'Age']=28
print(df,"\n")
df=df.drop('City',axis = 1)
print(df,"\n")
df=df.drop(3)
print(df,"\n")
```

```
      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

      Name  Age         City  Score
0    Alice   24     New York     85
1      Bob   28  Los Angeles     90
2  Charlie   22      Chicago     88
3    David   32      Houston     95

      Name  Age  Score
0    Alice   24     85
1      Bob   28     90
2  Charlie   22     88
3    David   32     95

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

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 • b) Find maximum Experience in each Department • c) Calculate total Salary paid

In [5]:
```
import pandas as pd
import numpy as np
data = {
'Department': ['HR', 'IT', 'HR', 'IT'],
'Salary': [30000, 50000, 35000, 55000],
'Experience': [2, 5, 3, 6]
}
df=pd.DataFrame(data);
a=df.groupby('Department')['Salary'].mean()
print(a,"\n")
c=df.groupby('Department')['Experience'].max()
print(c,"\n")
b=df['Salary'].sum()
print(b,"\n")
```

```
Department
HR    32500.0
IT    52500.0
Name: Salary, dtype: float64

Department
HR    3
IT    6
Name: Experience, dtype: int64

170000
```

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 • b) Drop rows with missing values • c) Sort the DataFrame by Marks in descending order

In [6]:
```
import pandas as pd
import numpy as np
data = {
'Student': ['John', 'Emma', 'Sam', 'Olivia'],
'Marks': [80, np.nan, 75, 90]
}
df=pd.DataFrame(data);
b= df.dropna()
print(b,"\n")
df=df.fillna(0)
print(df,"\n")
df=df.sort_values(by=['Marks'],ascending=[False])
print(df,"\n")
```

```
  Student  Marks
0    John   80.0
2     Sam   75.0
3  Olivia   90.0

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

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

7. 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 [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);
df.to_csv('students.csv', index=False)
print("csv files'output.csv'created successful.\n")
df = pd.read_csv('students.csv')
print(df,"\n")
```

```
csv files'output.csv'created successful.

      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. Expected Output: Product Price Stock 0 Laptop 70000 10 2 Smartphone 25000 50

In [8]:
```
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'. Expected Output: Subject Science Marks 92 Name: Student2, dtype: object

In [9]:
```
import pandas as pd
data = {
'Subject': ['Math', 'Science', 'English'],
'Marks': [88, 92, 85]
}
df = pd.DataFrame(data,index=['Student1', 'Student2', 'Student3'])
print(df,"\n")
df=df.loc['Student2']
print(df,"\n")
```

```
          Subject  Marks
Student1     Math     88
Student2  Science     92
Student3  English     85

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

In [ ]:

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). Hospital Staff Information Staff_ID, Name, Role, Department, Shift, Phone, Email, Joining_Date

In [1]:
```
import pandas as pd
data = {
    'Staff_ID': [f'S{i}' for i in range(1, 26)],
'Name': ['Anil', 'Sunita', 'Ramesh', 'Priya', 'Sanjay',
'Neha', 'Vikram', 'Kavita', 'Rajesh', 'Deepa',
'Manish', 'Ritu', 'Karan', 'Pooja', 'Suresh',
'Meena', 'Nitin', 'Divya', 'Ajay', 'Alka',
'Tarun', 'Shreya', 'Deepak', 'Payal', 'Sunny'],

'Role': ['Doctor', 'Nurse', 'Technician', 'Receptionist', 'Surgeon',
'Nurse', 'Technician', 'Doctor', 'Receptionist', 'Surgeon',
'Technician', 'Nurse', 'Doctor', 'Receptionist', 'Surgeon',
'Technician', 'Nurse', 'Doctor', 'Receptionist', 'Surgeon',
'Technician', 'Nurse', 'Doctor', 'Receptionist', 'Surgeon'],

'Department': ['Cardiology', 'Emergency', 'Radiology', 'Reception', 'Surgery',
'Emergency', 'Radiology', 'Cardiology', 'Reception', 'Surgery',
'Radiology', 'Emergency', 'Cardiology', 'Reception', 'Surgery',
'Radiology', 'Emergency', 'Cardiology', 'Reception', 'Surgery',
'Radiology', 'Emergency', 'Cardiology', 'Reception', 'Surgery'],

'Shift': ['Morning', 'Afternoon', 'Night', 'Morning', 'Afternoon',
'Night', 'Morning', 'Afternoon', 'Night', 'Morning',
'Afternoon', 'Night', 'Morning', 'Afternoon', 'Night',
'Morning', 'Afternoon', 'Night', 'Morning', 'Afternoon',
'Night', 'Morning', 'Afternoon', 'Night', 'Morning'],

'Phone': ['9876543210', '8765432109', '7654321098', '6543210987', '5432109876',
'9876501234', '8765401235', '7654301236', '6543201237', '5432101238'
'9876504321', '8765404322', '7654304323', '6543204324', '5432104325',
'9876541230', '8765431231', '7654321232', '6543211233', '5432101234',
'9876505678', '8765405679', '7654305680', '6543205681', '5432105682'],

'Email': ['anil@hospital.com', 'sunita@hospital.com', 'ramesh@hospital.com', 'pr
'neha@hospital.com', 'vikram@hospital.com', 'kavita@hospital.com', 'rajesh@hospi
'manish@hospital.com', 'ritu@hospital.com', 'karan@hospital.com', 'pooja@hospita
'meena@hospital.com', 'nitin@hospital.com', 'divya@hospital.com', 'ajay@hospital
'tarun@hospital.com', 'shreya@hospital.com', 'deepak@hospital.com', 'payal@hospi

'Joining_Date': ['2019-01-10', '2018-05-15', '2020-07-20', '2019-08-25', '2021-0
'2020-03-14', '2018-09-30', '2019-12-05', '2021-04-10', '2020-11-11',
'2019-06-18', '2018-08-21', '2021-01-25', '2020-10-30', '2019-03-22',
'2020-07-07', '2018-12-12', '2019-05-05', '2021-06-16', '2020-02-20',
'2019-09-29', '2018-11-14', '2020-04-19', '2019-07-24', '2021-03-13']
}
df = pd.DataFrame(data)
```

```
df.to_csv('hospital_staff_s1_s2.csv', index=False)
print("CSV file 'hospital_staff_s1_s2.csv' created successfully.")
```

```
CSV file 'hospital_staff_s1_s2.csv' created successfully.
```

In [ ]:

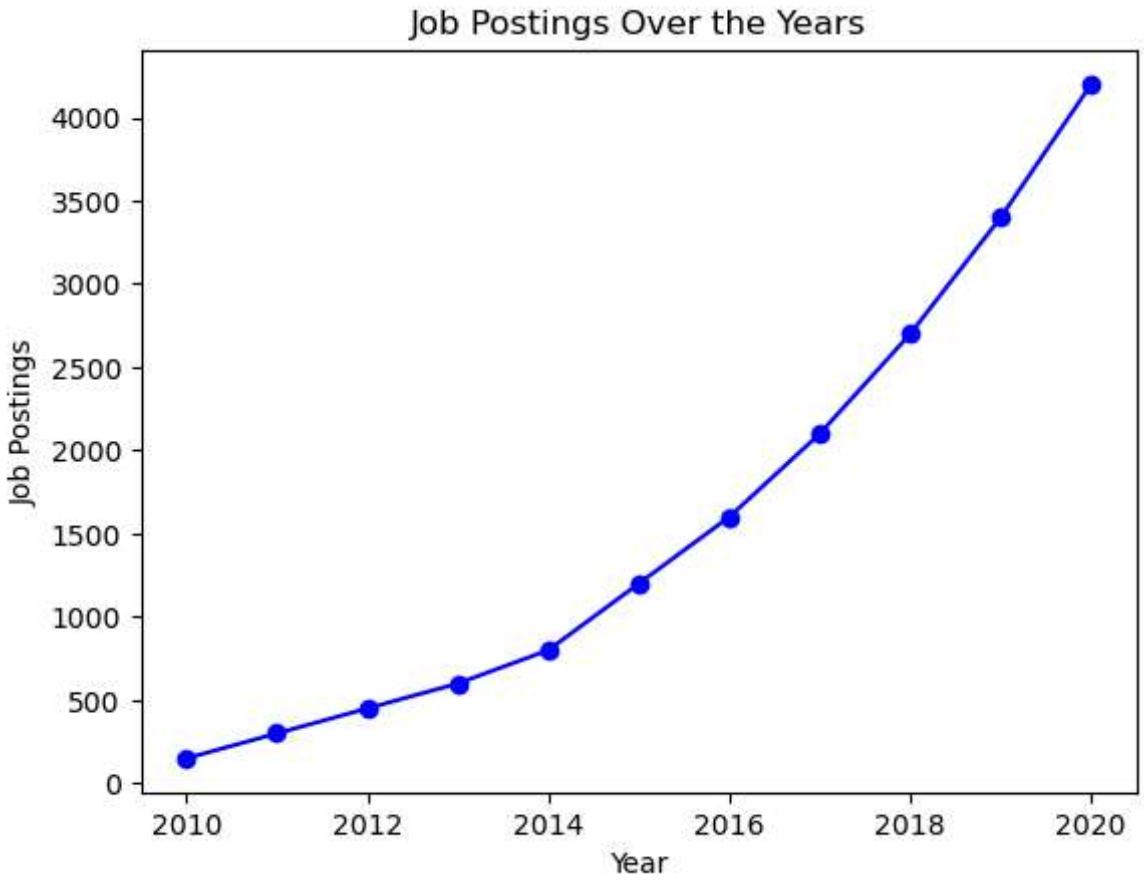
1. The following table shows the number of Data Science job postings recorded over the years 2010 to 2020. Year Job Postings 2010 150 2011 300 2012 450 2013 600 2014 800 2015 1200 2016 1600 2017 2100 2018 2700 2019 3400 2020 4200 Using Python (Pandas and Matplotlib): Create a DataFrame for the given data

In [1]:
```
import pandas as pd
data={
'Year':[2010, 2011, 2012, 2013, 2014, 2015, 2016, 2017, 2018, 2019, 2020],
'Job Postings': [150, 300, 450, 600, 800, 1200, 1600, 2100, 2700, 3400, 4200]
}
df=pd.DataFrame(data)
print(df)
```

```
    Year  Job Postings
0   2010           150
1   2011           300
2   2012           450
3   2013           600
4   2014           800
5   2015          1200
6   2016          1600
7   2017          2100
8   2018          2700
9   2019          3400
10  2020          4200
```

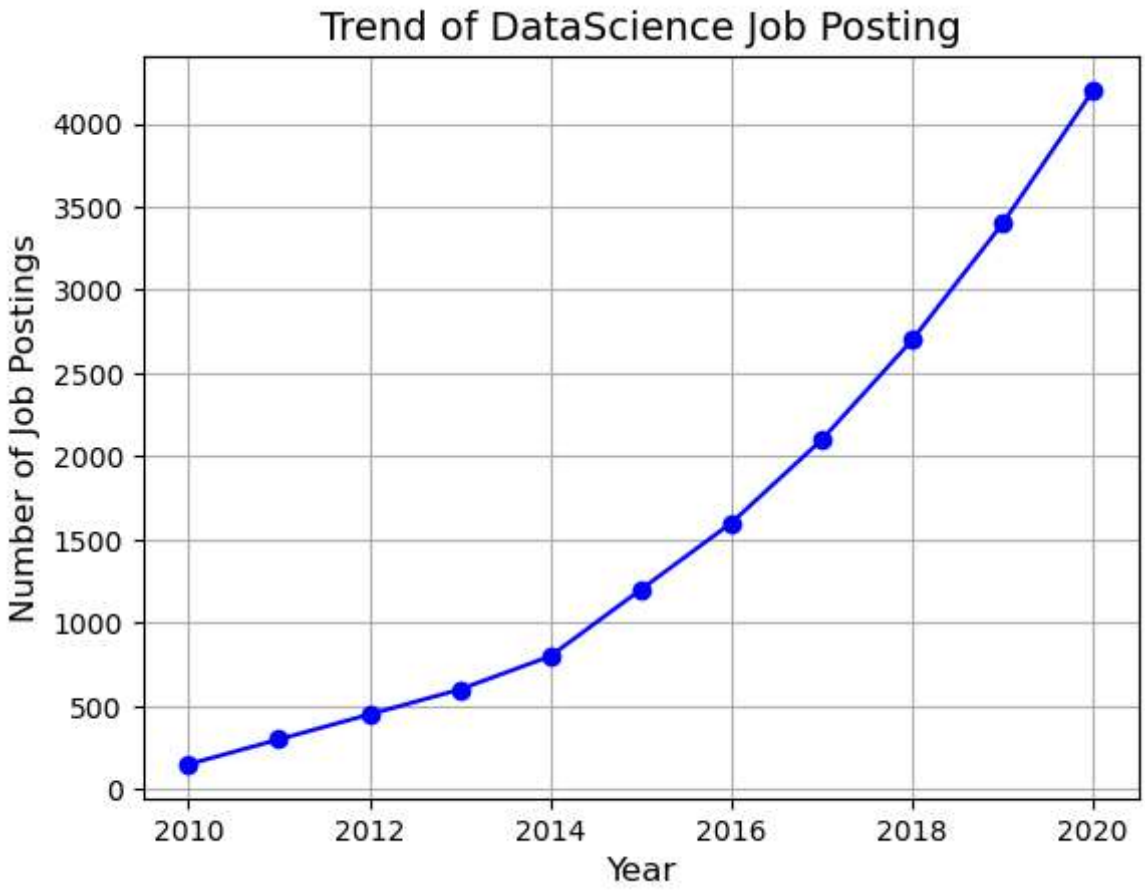
2. Plot a line graph showing the trend of Data Science job postings over the year on data points.

In [4]:
```
import pandas as pd
import matplotlib.pyplot as plt
data = {
    'Year': [2010, 2011, 2012, 2013, 2014, 2015, 2016, 2017, 2018, 2019, 2020],
    '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', linestyle='-', color='b')
plt.xlabel('Year')
plt.ylabel('Job Postings')
plt.title('Job Postings Over the Years')
plt.show()
```



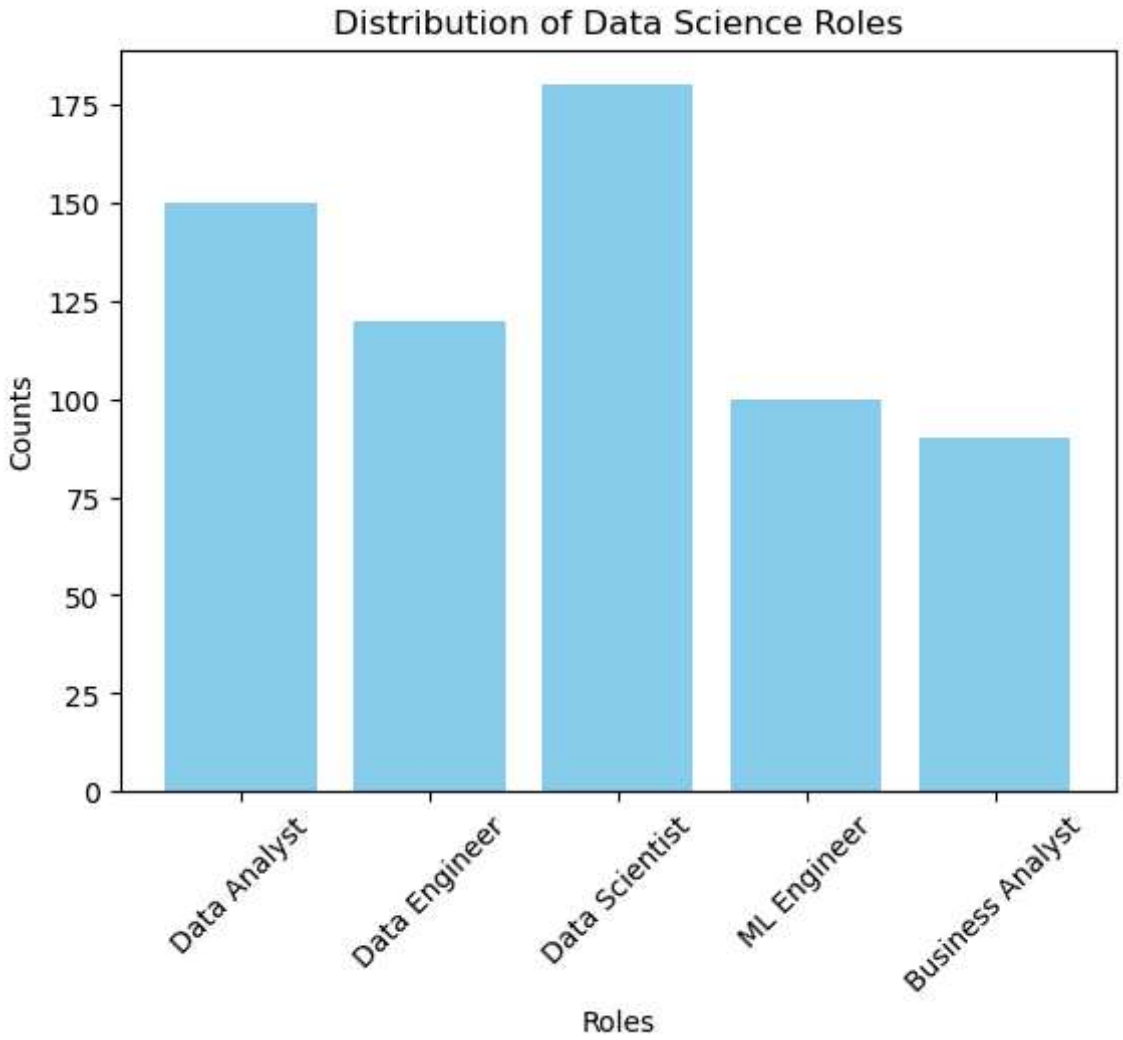
3. Add suitable title and axis labels to the graph. Expected Output: A line chart showing steady growth of job postings from 2010 to 2020.

In [5]:
```
import pandas as pd
import matplotlib.pyplot as plt
data={
'Year':[2010, 2011, 2012, 2013, 2014, 2015, 2016, 2017, 2018, 2019, 2020],
'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',linestyle='-',color='b')
plt.title('Trend of DataScience Job Posting',fontsize=14)
plt.xlabel('Year',fontsize=12)
plt.ylabel('Number of Job Postings',fontsize=12)
plt.grid(True)
plt.show()
```



4.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 [9]:
```
import pandas as pd
import matplotlib.pyplot as plt
data = {
    'roles': ['Data Analyst', 'Data Engineer', 'Data Scientist', 'ML Engineer',
    'counts': [150, 120, 180, 100, 90]
}
df = pd.DataFrame(data)
plt.bar(df['roles'], df['counts'], color='skyblue')
plt.title('Distribution of Data Science Roles')
plt.xlabel('Roles')
plt.ylabel('Counts')
plt.xticks(rotation=45)
plt.show()
```



4."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 [11]:
```
import pandas as pd
data = {
    'name': ['Arjun', 'Meera', 'Sahana'],
    'age': [22, 20, 21],
    'city': ['Mumbai', 'Bangalore', 'Delhi']
}
df = pd.DataFrame(data)
print("structured_data\n")
print(df, "\n")
unstructured_data = "We all love learning data science and AI."
print("unstructured_data\n")
print(unstructured_data, "\n")
semi_structured_data = {
    "student": {
        "id": 202,
        "name": "Alice Smith",
        "courses": ["Physics", "Chemistry", "English"],
        "address": {
            "city": "San Francisco",
            "zipcode": "94105"
        }
```

```
    }
}
print("semi_structured_data\n")
print(semi_structured_data)
```

```
structured_data

     name  age       city
0   Arjun   22     Mumbai
1   Meera   20  Bangalore
2  Sahana   21      Delhi

unstructured_data

We all love learning data science and AI.

semi_structured_data

{'student': {'id': 202, 'name': 'Alice Smith', 'courses': ['Physics', 'Chemistr
y', 'English'], 'address': {'city': 'San Francisco', 'zipcode': '94105'}}}
```

5."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."

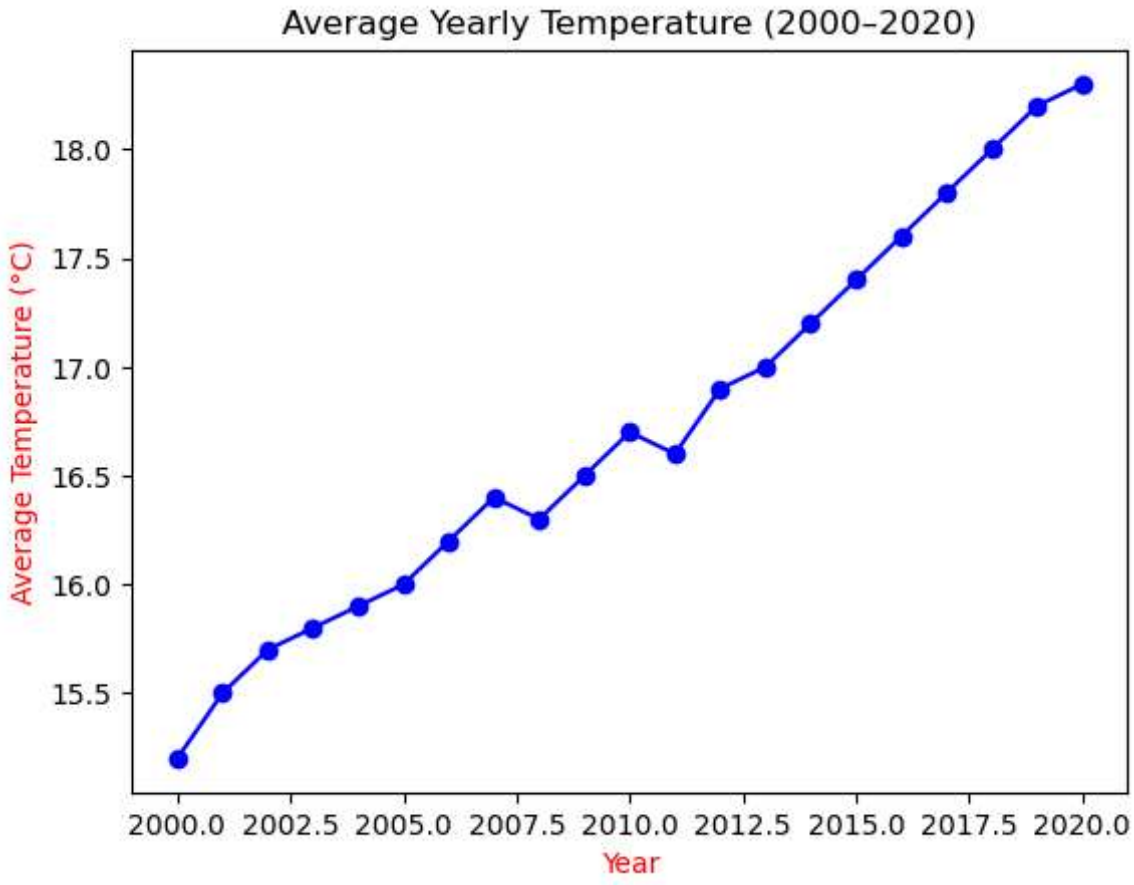
In [12]:
```
from cryptography.fernet import Fernet
key = Fernet.generate_key()
f = Fernet(key)
token = f.encrypt(b"Rajalakshmi Engineering College")
token
f.decrypt(token)
key = Fernet.generate_key()
cipher_suite = Fernet(key)
plain_text = b"Rajalakshmi Engineering College."
cipher_text = cipher_suite.encrypt(plain_text)
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'gAAAAABo1DZDvjW2SniX1zxYBtdqzF_ftYm4W11KctBl3G8QI96wJZbOYSSmnkj
gK-lms_pKF3OuNc9Hm_eoK_N0wCfBI1zTfbcz-vKSuzIoJ8owWCAOWzTwu3bfVQhWKmHjpLqxwOrM'
Decrypted Data: b'Rajalakshmi Engineering College.'
```

In [ ]:
```
```

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.

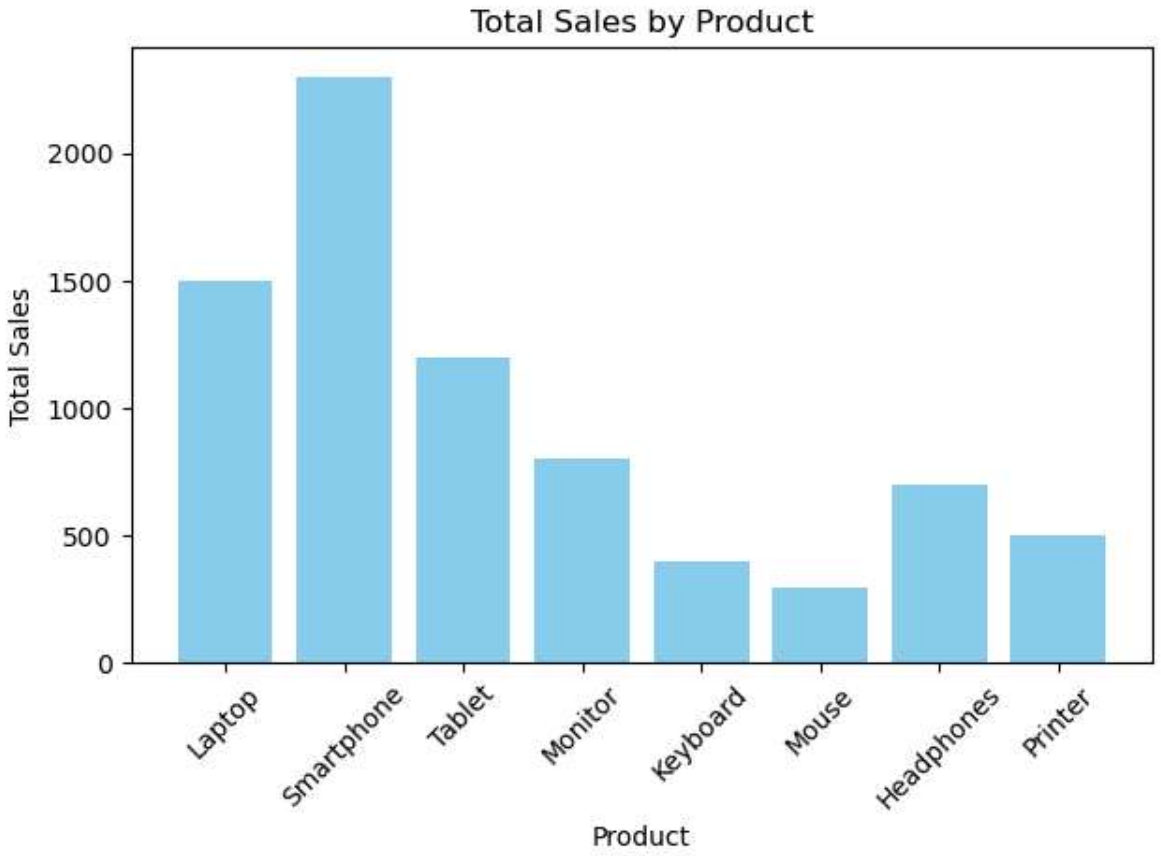
In [38]:
```
import pandas as pd
import matplotlib.pyplot as plt
years = list(range(2000, 2021))
temperatures = [
    15.2, 15.5, 15.7, 15.8, 15.9, 16.0, 16.2,
    16.4, 16.3, 16.5, 16.7, 16.6, 16.9, 17.0,
    17.2, 17.4, 17.6, 17.8, 18.0, 18.2, 18.3
]
data = pd.DataFrame({
    'Year': years,
    'AvgTemperature': temperatures
})
data.to_csv('weather.csv', index=False)
df = pd.read_csv('weather.csv')
df = df[(df['Year'] >= 2000) & (df['Year'] <= 2020)]
plt.plot(df['Year'], df['AvgTemperature'], marker='o', linestyle='-', color='blu
plt.xlabel('Year',color='red')
plt.ylabel('Average Temperature (°C)',color='red')
plt.title('Average Yearly Temperature (2000-2020)')
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.

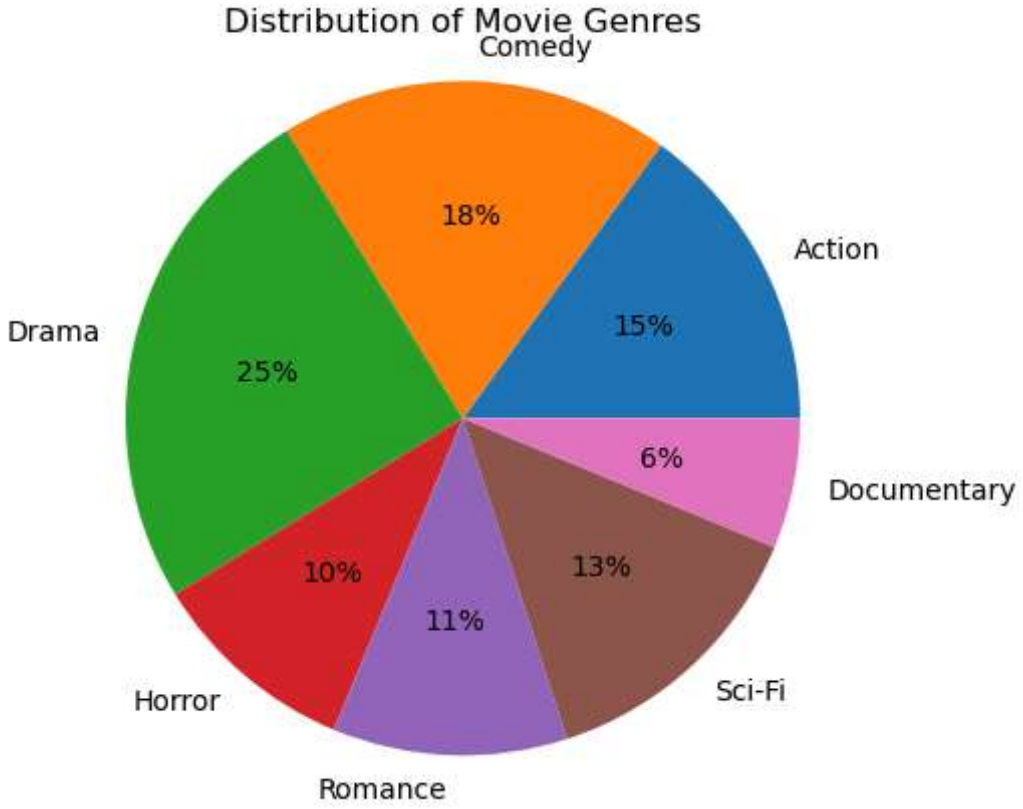
In [12]:
```
import pandas as pd
import matplotlib.pyplot as plt
data = {
    'Product': ['Laptop', 'Smartphone', 'Tablet', 'Monitor', 'Keyboard', 'Mouse'
    'Sales': [1500, 2300, 1200, 800, 400, 300, 700, 500]
}
df = pd.DataFrame(data)
df.to_csv('sales.csv', index=False)
df = pd.read_csv('sales.csv')
plt.bar(df['Product'], df['Sales'], color='skyblue')
plt.xlabel('Product')
plt.ylabel('Total Sales')
plt.title('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".

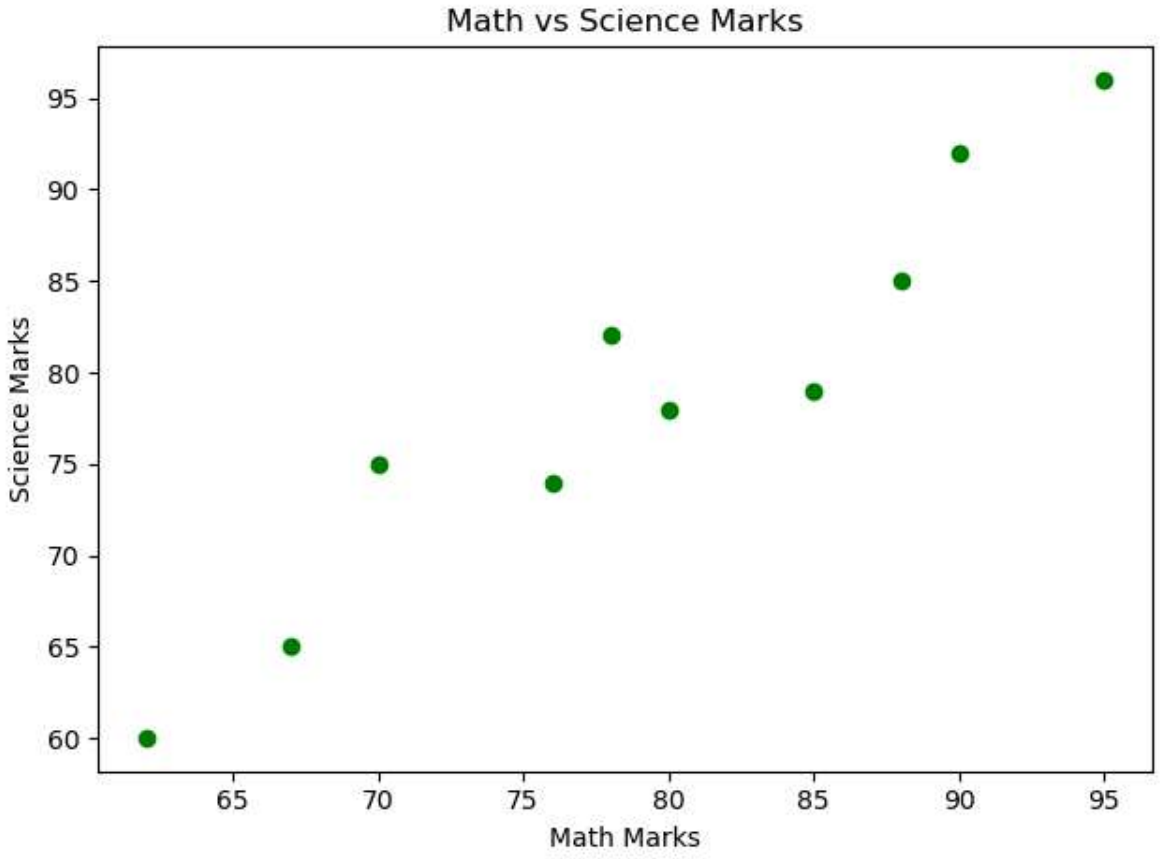
In [22]:
```
import pandas as pd
import matplotlib.pyplot as plt
data = {
    'Genre': ['Action', 'Comedy', 'Drama', 'Horror', 'Romance', 'Sci-Fi', 'Docum
    'Count': [120, 150, 200, 80, 90, 110, 50]
}
```

```
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='%d%%')
plt.title('Distribution of Movie Genres')
plt.axis('equal')
plt.show()
```



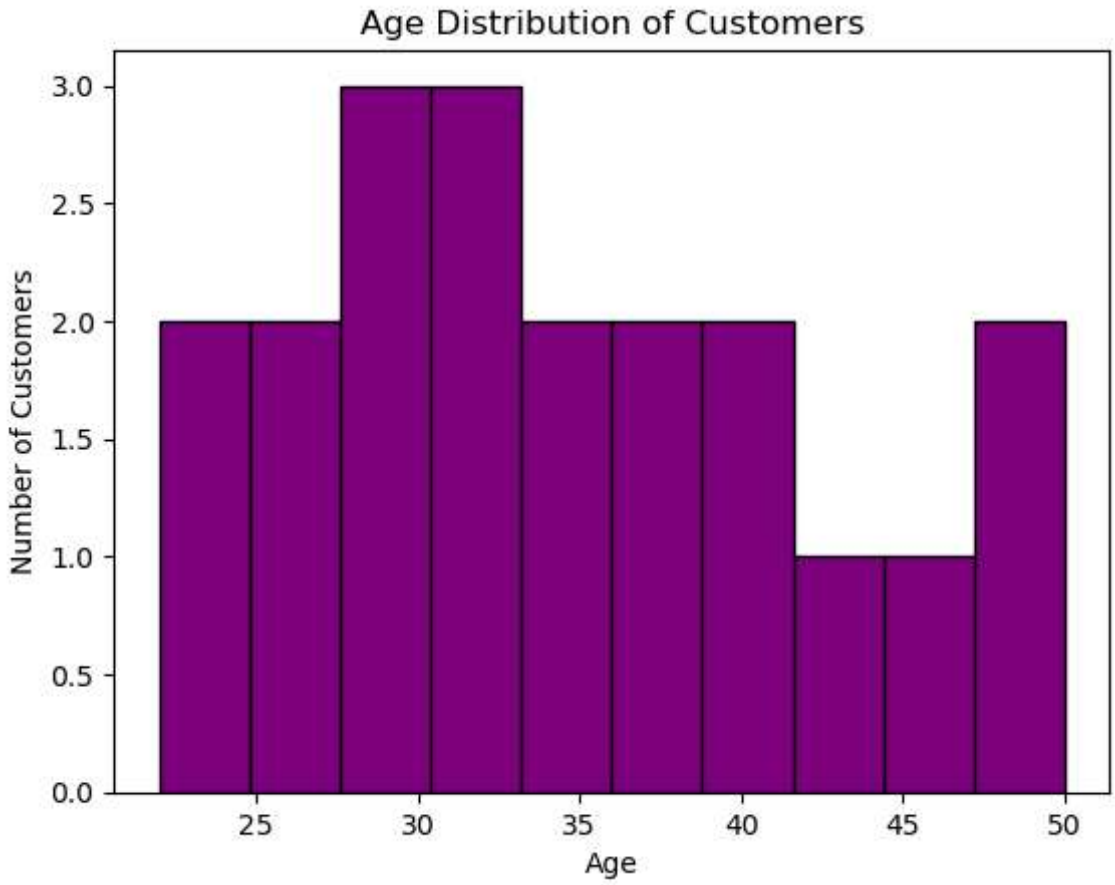
Q4. Scatter Plot (Students' Marks) Given a CSV file students.csv with columns: MathMarks, ScienceMarks. Write a Python program to plot a scatter plot showing the relationship between marks in Mathematics and Science. • Add axis labels and a title. • Use different colors/markers if possible.

In [24]:
```
import pandas as pd
import matplotlib.pyplot as plt
data = {
    'MathMarks': [78, 85, 62, 90, 70, 88, 76, 95, 67, 80],
    'ScienceMarks': [82, 79, 60, 92, 75, 85, 74, 96, 65, 78]
}
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='o')
plt.xlabel('Math Marks')
plt.ylabel('Science Marks')
plt.title("Math vs Science Marks")
plt.tight_layout()
plt.show()
```



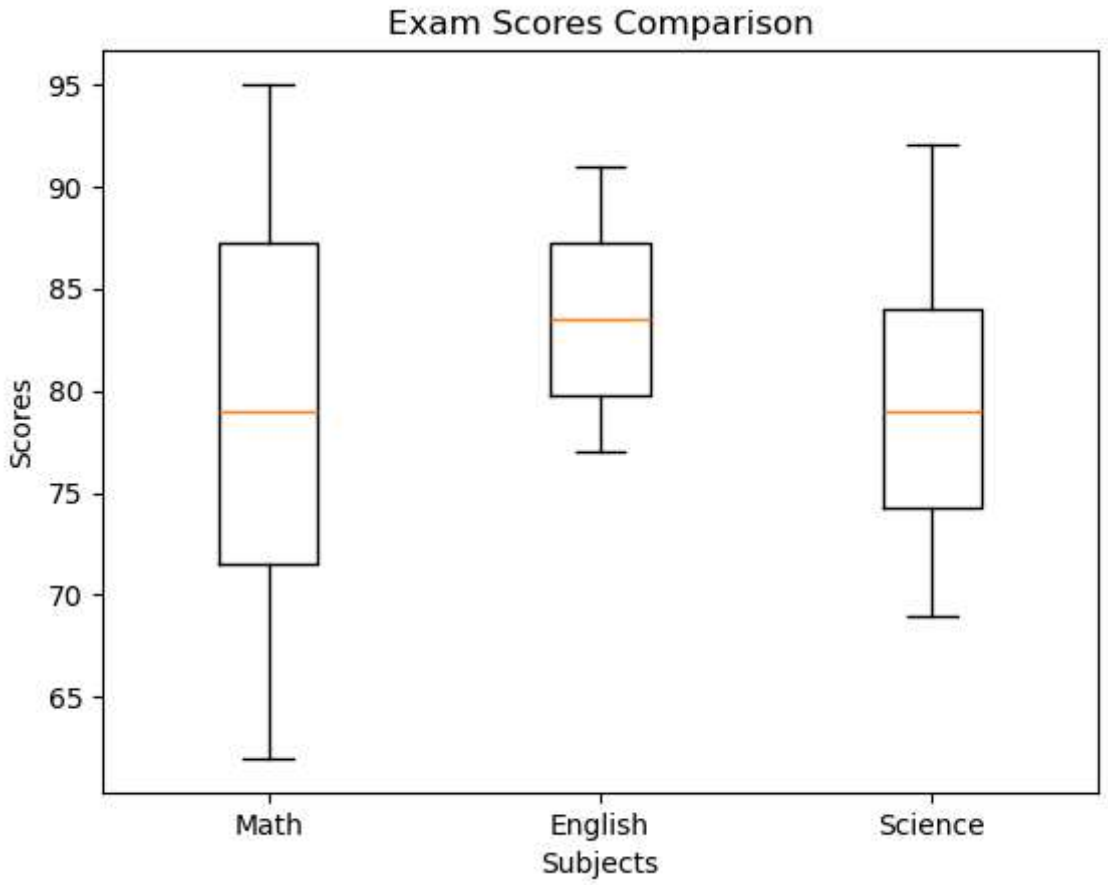
Q5. Histogram (Customer Ages) Given a CSV file customers.csv with a column: Age. Write a Python program to plot a histogram showing the age distribution of customers. • Use 10 bins. • Add axis labels and a title.

In [27]:
```
import pandas as pd
import matplotlib.pyplot as plt
data = {
    'Age': [23, 45, 31, 35, 40, 22, 27, 29, 34, 50, 41, 38, 33, 26, 44, 48, 30,
}
df = pd.DataFrame(data)
df.to_csv('customers.csv', index=False)
df = pd.read_csv('customers.csv')
plt.hist(df['Age'], bins=10, color='purple', edgecolor='black')
plt.xlabel('Age')
plt.ylabel('Number of Customers')
plt.title('Age Distribution of Customers')
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 [30]:
```
import pandas as pd
import matplotlib.pyplot as plt
data = {
    'Math': [78, 85, 62, 90, 70, 88, 76, 95, 67, 80],
    'English': [82, 79, 88, 85, 90, 91, 77, 84, 79, 83],
    'Science': [75, 80, 72, 88, 78, 85, 69, 92, 74, 81]
}
df = pd.DataFrame(data)
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', 'En
plt.xlabel('Subjects')
plt.ylabel('Scores')
plt.title('Exam Scores Comparison')
plt.show()
```

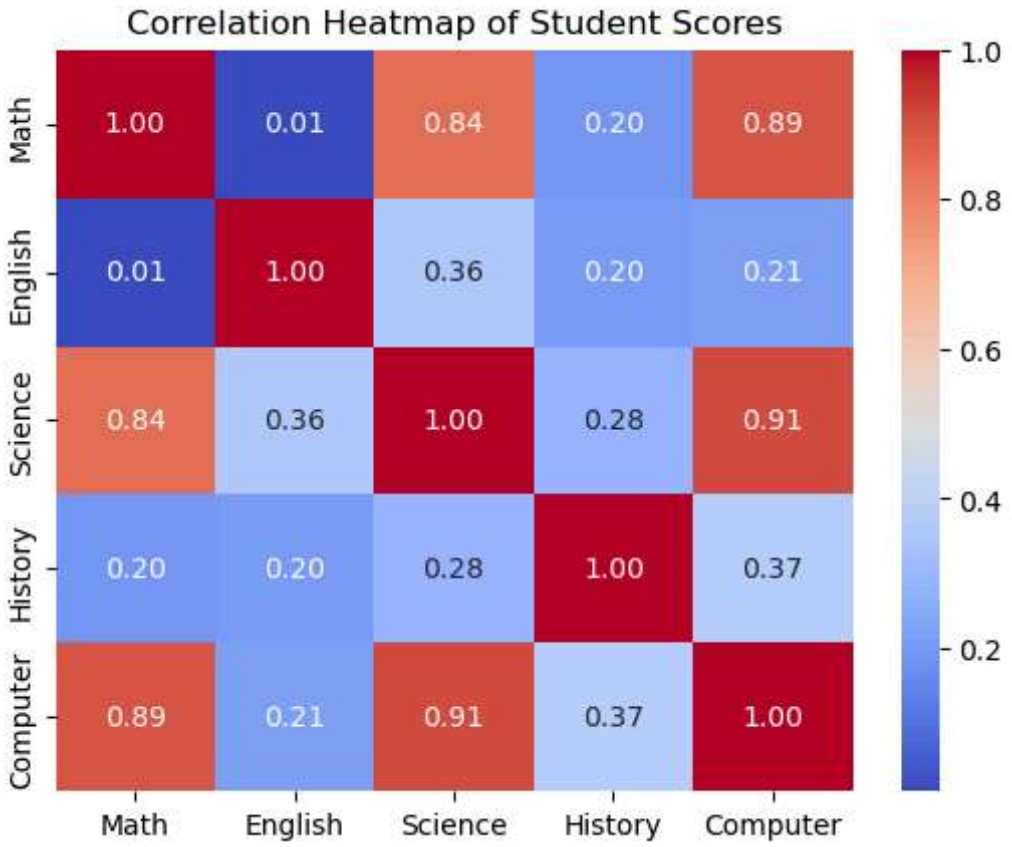


Q7. Area Chart (Website Traffic) Create a CSV file traffic.csv with columns: Month, Visitors. Write a Python program to plot an area chart showing monthly website visitors for a year. • Add axis labels and a title. • Use a different color for the filled area.

In [31]:
```
import pandas as pd
import matplotlib.pyplot as plt
data = {
    'Month': ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep', 'Oc
    'Visitors': [1200, 1500, 1700, 1600, 1800, 2100, 2300, 2200, 2000, 1900, 175
}
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='skyblue', alpha=0.5)
plt.plot(df['Month'], df['Visitors'], color='Slateblue', linewidth=2)
plt.xlabel('Month')
plt.ylabel('Number of Visitors')
plt.title('Monthly Website Visitors')
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 [37]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
data = {
    'Math': [78, 85, 62, 90, 70, 88, 76, 95, 67, 80],
    'English': [82, 79, 88, 85, 90, 91, 77, 84, 79, 83],
    'Science': [75, 80, 72, 88, 78, 85, 69, 92, 74, 81],
    'History': [70, 75, 80, 72, 68, 74, 71, 78, 69, 73],
    'Computer': [88, 92, 85, 95, 87, 90, 84, 96, 82, 89]
}
df = pd.DataFrame(data)
df.to_csv('students_scores.csv', index=False)
df = pd.read_csv('students_scores.csv')
correlation = df.corr()
sns.heatmap(correlation, annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Correlation Heatmap of Student Scores')
plt.show()
```


Correlation Heatmap of Student Scores
Math
English
Science
History
Computer
1.00 0.01 0.84 0.20 0.89
0.01 1.00 0.36 0.20 0.21
0.84 0.36 1.00 0.28 0.91
0.20 0.20 0.28 1.00 0.37
0.89 0.21 0.91 0.37 1.00
Math
English
Science
History
Computer
1.0
0.8
0.6
0.4
0.2

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

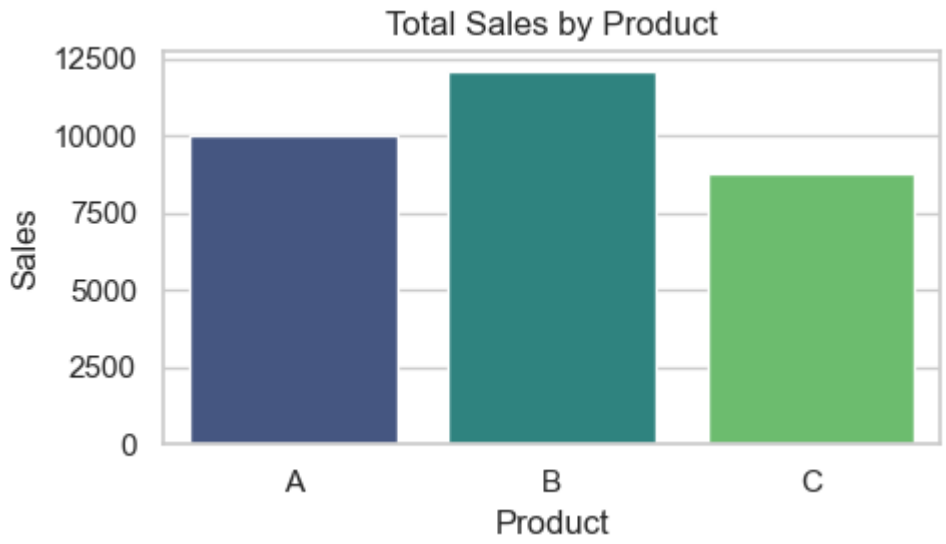
1. Load the dataset into a Pandas DataFrame and display the first few records.
2. Identify and handle missing values (replace missing Sales values with the mean, and drop rows with missing Product, Quantity, or Region).
3. Generate summary statistics of the numerical columns.
4. Group the data by Product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

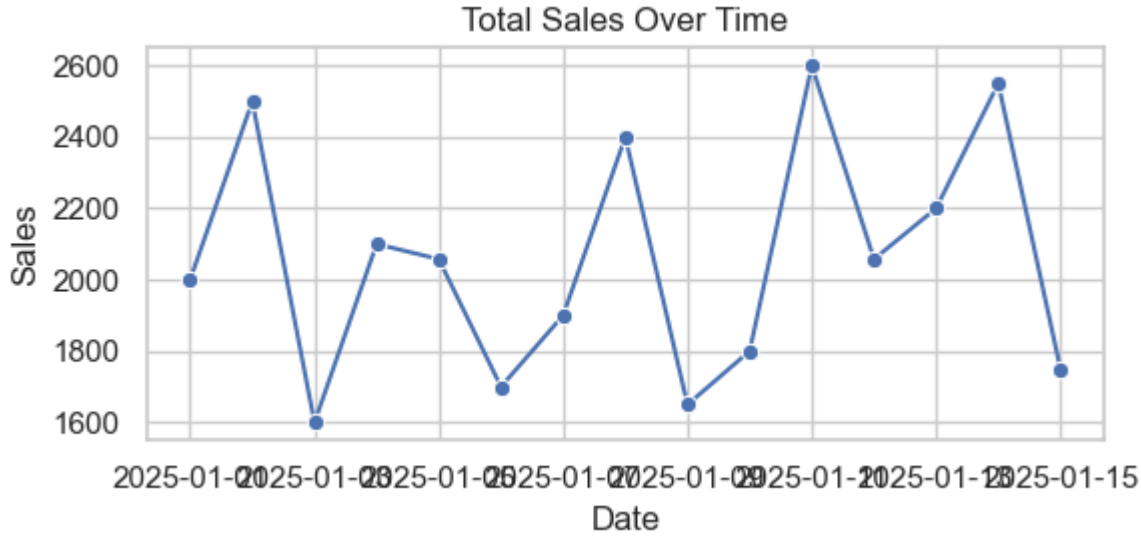
In [2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

sns.set(style="whitegrid")

data = {
    'Date': pd.date_range(start='2025-01-01', periods=15, freq='D'),
    'Product': ['A', 'B', 'C', 'A', 'B', 'C', 'A', 'B', 'C', 'A', 'B', 'C', 'A',
    'Quantity': [10, 15, 8, 12, 14, 9, 11, 13, 10, 9, 16, 8, 15, 14, 9],
    'Sales': [2000, 2500, 1600, 2100, np.nan, 1700, 1900, 2400, 1650, 1800, 2600
    'Region': ['North', 'South', 'East', 'North', 'South', 'West', 'North', 'Sou
}

df = pd.DataFrame(data)

df['Sales'] = df['Sales'].fillna(df['Sales'].mean())
df.dropna(subset=['Product', 'Quantity', 'Region'], inplace=True)

print(df.head())
print(df.describe())

product_summary = df.groupby('Product')[['Sales', 'Quantity']].sum().reset_index
print(product_summary)

plt.figure(figsize=(5, 3))
sns.barplot(data=product_summary, x='Product', y='Sales', palette='viridis')
plt.title('Total Sales by Product')
plt.tight_layout()
plt.show()

df['Date'] = pd.to_datetime(df['Date'])
daily_sales = df.groupby('Date')['Sales'].sum().reset_index()

plt.figure(figsize=(6, 3))
sns.lineplot(data=daily_sales, x='Date', y='Sales', marker='o')
```

```
plt.title('Total Sales Over Time')
plt.tight_layout()
plt.show()

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

plt.figure(figsize=(5, 3))
sns.heatmap(pivot_table, annot=True, fmt='.1f', cmap='coolwarm')
plt.title('Sales by Region and Product')
plt.tight_layout()
plt.show()

corr = df.corr(numeric_only=True)
print(corr)

plt.figure(figsize=(4, 3))
sns.heatmap(corr, annot=True, cmap='Blues', fmt='.2f')
plt.title('Correlation Heatmap')
plt.tight_layout()
plt.show()
```

```
        Date Product  Quantity        Sales Region
0 2025-01-01       A        10  2000.000000  North
1 2025-01-02       B        15  2500.000000  South
2 2025-01-03       C         8  1600.000000   East
3 2025-01-04       A        12  2100.000000  North
4 2025-01-05       B        14  2057.692308  South
                        Date   Quantity        Sales
count                     15  15.000000    15.000000
mean   2025-01-08 00:00:00  11.533333  2057.692308
min    2025-01-01 00:00:00   8.000000  1600.000000
25%    2025-01-04 12:00:00   9.000000  1775.000000
50%    2025-01-08 00:00:00  11.000000  2057.692308
75%    2025-01-11 12:00:00  14.000000  2300.000000
max    2025-01-15 00:00:00  16.000000  2600.000000
std                    NaN   2.774029   334.261528
  Product         Sales  Quantity
0       A  10000.000000        57
1       B  12107.692308        72
2       C   8757.692308        44
```

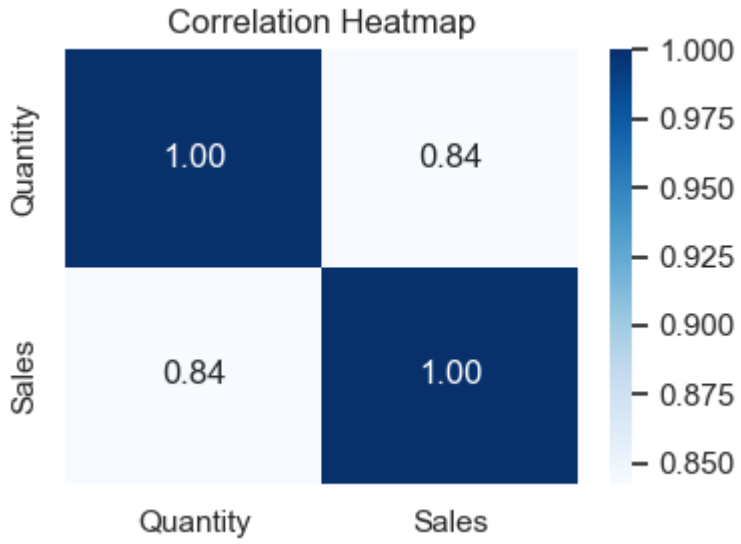
```
C:\Users\kaushika\AppData\Local\Temp\ipykernel_15448\2323665925.py:28: FutureWarn
ing:

Passing `palette` without assigning `hue` is deprecated and will be removed in v
0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effe
ct.

  sns.barplot(data=product_summary, x='Product', y='Sales', palette='viridis')
```





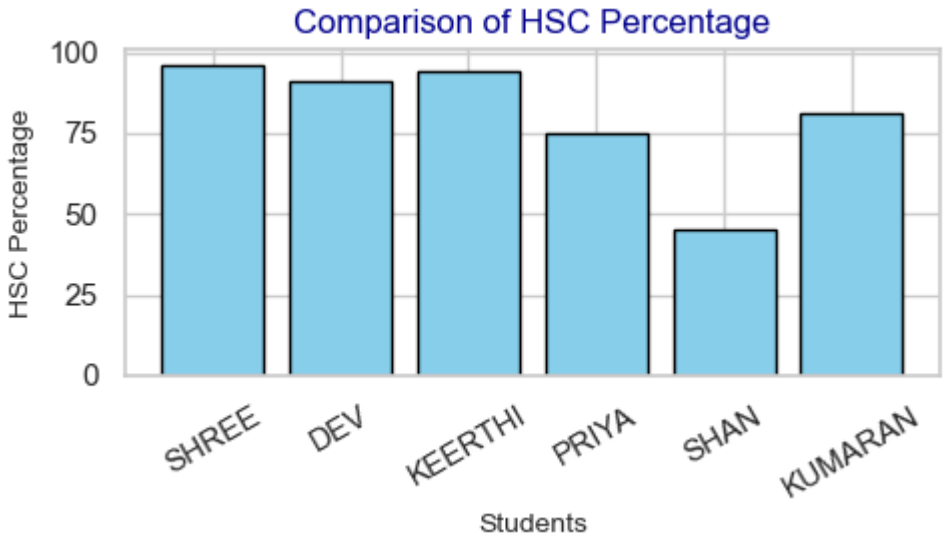
Product	A	B	C
Region			
East	2200.0	0.000000	3250.000000
North	6000.0	2600.000000	1750.000000
South	0.0	6957.692308	2057.692308
West	1800.0	2550.000000	1700.000000

	Quantity	Sales
Quantity	1.000000	0.842618
Sales	0.842618	1.000000



2. A dataset contains the names of six students (SHREE, DEV, KEERTHI, PRIYA, SHAN, KUMARAN) along with their Higher Secondary Certificate (HSC) exam percentages: 96, 91, 94, 75, 45, 81. Using Matplotlib and NumPy in Python:
3. Plot a bar chart comparing the HSC percentages of the students.
4. Set appropriate labels for the x-axis and y-axis.
5. Rotate the x-axis tick labels for better readability.
6. Add a title ("Comparison of HSC Percentage") with customized font size and color.
7. Display the chart using show().

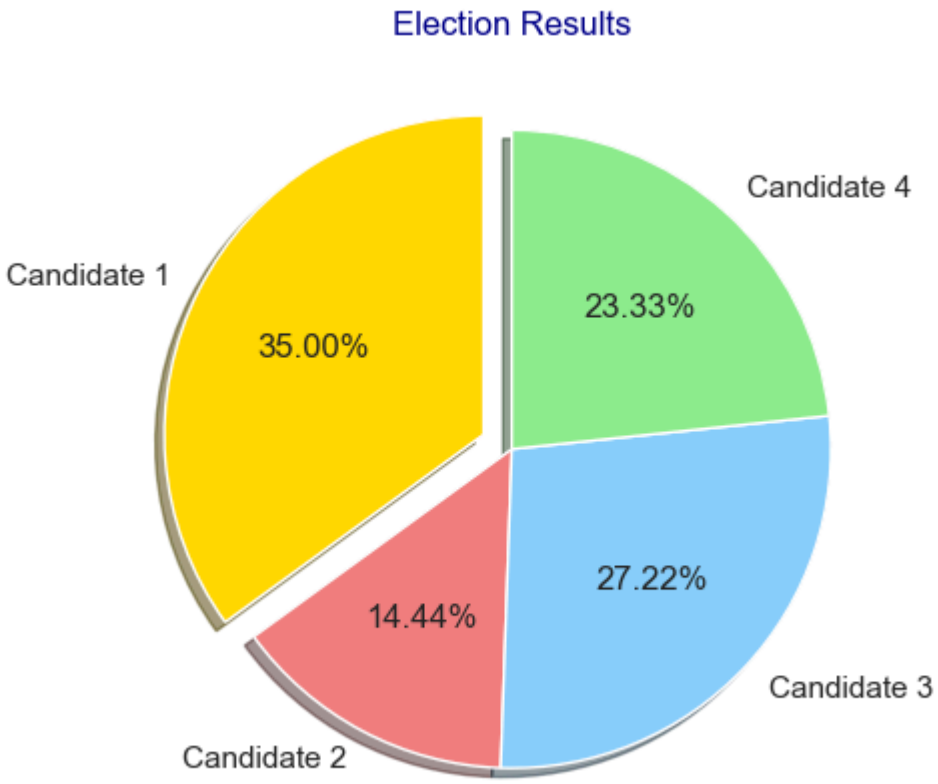
In [3]:
```
import numpy as np
import matplotlib.pyplot as plt

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

plt.figure(figsize=(5, 3))
plt.bar(students, percentages, color='skyblue', edgecolor='black')

plt.xlabel('Students', fontsize=10)
plt.ylabel('HSC Percentage', fontsize=10)
plt.title('Comparison of HSC Percentage', fontsize=12, color='darkblue')

plt.xticks(rotation=30)
plt.tight_layout()
plt.show()
```



3.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=(5, 5))
plt.pie(votes, labels=candidates, colors=colors, explode=explode,
        autopct='%.2f%%', startangle=90, shadow=True)

plt.title('Election Results', fontsize=12, color='darkblue')
plt.tight_layout()
plt.show()
```



4. To Count the frequency of occurrence of a word in a body of text is often needed during text processing. Description: Import the word_tokenize function and gutenberg.

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

```
[nltk_data] Downloading package gutenberg to
[nltk_data]     C:\Users\kaushika\AppData\Roaming\nltk_data...
[nltk_data]   Package gutenberg is already up-to-date!
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\kaushika\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package punkt_tab to
[nltk_data]     C:\Users\kaushika\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping tokenizers\punkt_tab.zip.
```

Out[6]: True

In [8]:
```
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))))
```

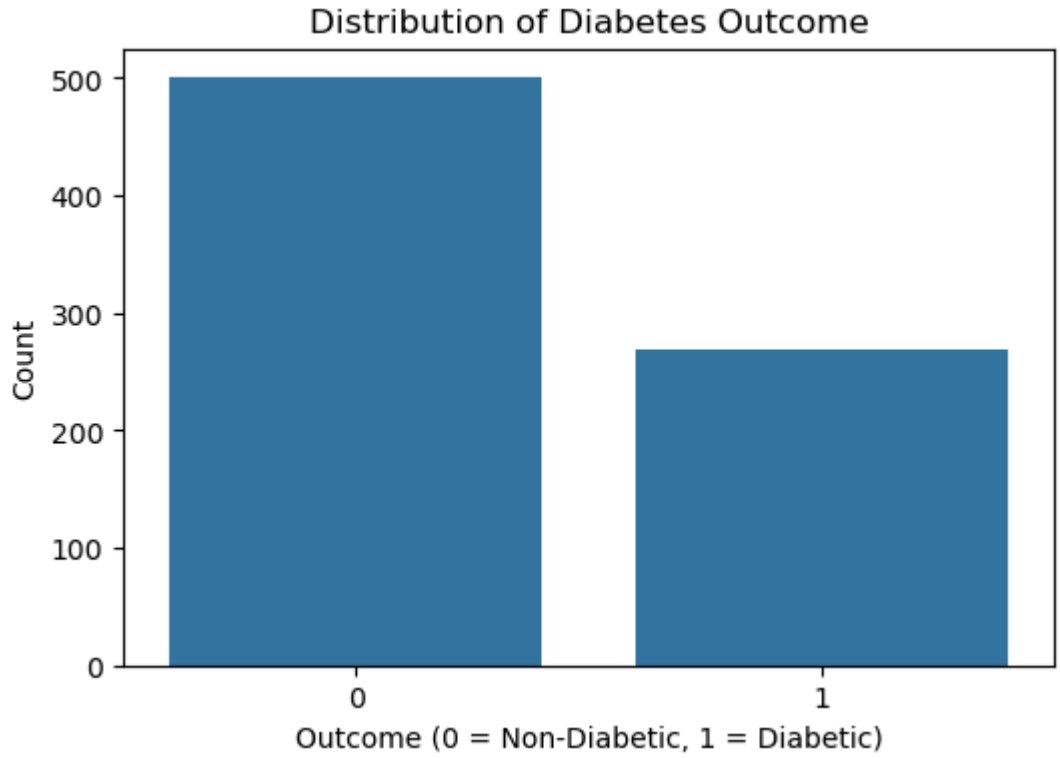
```
Pairs
[('[', 1), ('Emma', 2), ('by', 1), ('Jane', 1), ('Austen', 1), ('1816', 1), (']',
1), ('VOLUME', 1), ('I', 2), ('CHAPTER', 1), ('I', 2), ('Emma', 2), ('Woodhouse',
1), (',', 5), ('handsome', 1), (',', 5), ('clever', 1), (',', 5), ('and', 3), ('r
ich', 1), (',', 5), ('with', 2), ('a', 1), ('comfortable', 1), ('home', 1), ('an
d', 3), ('happy', 1), ('disposition', 1), (',', 5), ('seemed', 1), ('to', 1), ('u
nite', 1), ('some', 1), ('of', 2), ('the', 2), ('best', 1), ('blessings', 1), ('o
f', 2), ('existence', 1), (';', 1), ('and', 3), ('had', 1), ('lived', 1), ('nearl
y', 1), ('twenty-one', 1), ('years', 1), ('in', 1), ('the', 2), ('world', 1), ('w
ith', 2)]
```

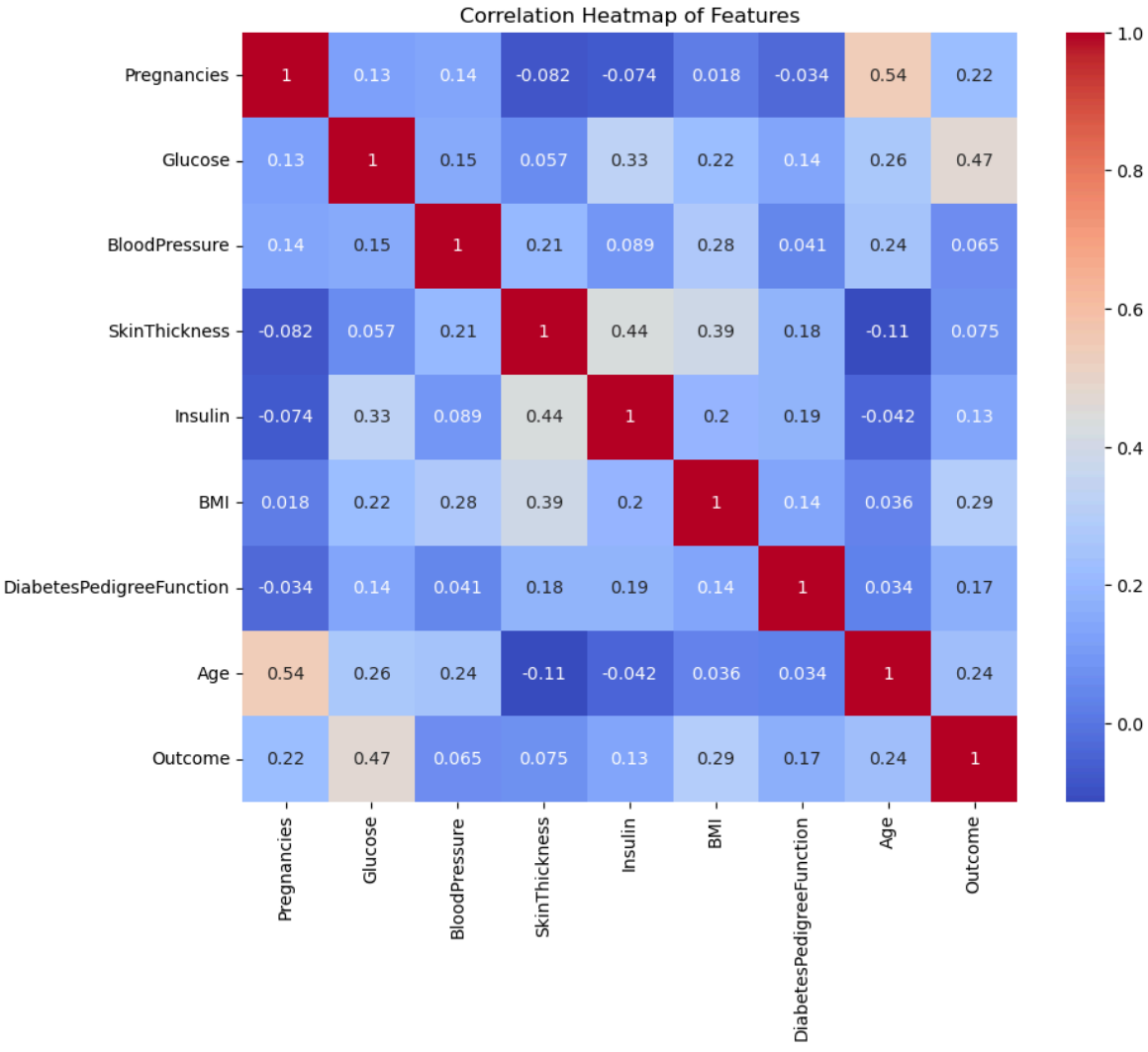
In [ ]:

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 [8]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
data = pd.read_csv("diabetes.csv")
print(data.head())
print(data.info())
print(data.describe())
print(data.isnull().sum())
plt.figure(figsize=(6,4))
sns.countplot(x='Outcome', data=data)
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(data.corr(), annot=True, cmap='coolwarm')
plt.title('Correlation Heatmap of Features')
plt.show()
```

```
   Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
0            6      148             72             35        0  33.6
1            1       85             66             29        0  26.6
2            8      183             64              0        0  23.3
3            1       89             66             23       94  28.1
4            0      137             40             35      168  43.1

   DiabetesPedigreeFunction  Age  Outcome
0                     0.627   50        1
1                     0.351   31        0
2                     0.672   32        1
3                     0.167   21        0
4                     2.288   33        1
<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
       Pregnancies     Glucose  BloodPressure  SkinThickness     Insulin  \
count   768.000000  768.000000     768.000000     768.000000  768.000000
mean      3.845052  120.894531      69.105469      20.536458   79.799479
std       3.369578   31.972618      19.355807      15.952218  115.244002
min       0.000000    0.000000       0.000000       0.000000    0.000000
25%       1.000000   99.000000      62.000000       0.000000    0.000000
50%       3.000000  117.000000      72.000000      23.000000   30.500000
75%       6.000000  140.250000      80.000000      32.000000  127.250000
max      17.000000  199.000000     122.000000      99.000000  846.000000

              BMI  DiabetesPedigreeFunction         Age     Outcome
count  768.000000                768.000000  768.000000  768.000000
mean    31.992578                  0.471876   33.240885    0.348958
std      7.884160                  0.331329   11.760232    0.476951
min      0.000000                  0.078000   21.000000    0.000000
25%     27.300000                  0.243750   24.000000    0.000000
50%     32.000000                  0.372500   29.000000    0.000000
75%     36.600000                  0.626250   41.000000    1.000000
max     67.100000                  2.420000   81.000000    1.000000
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```



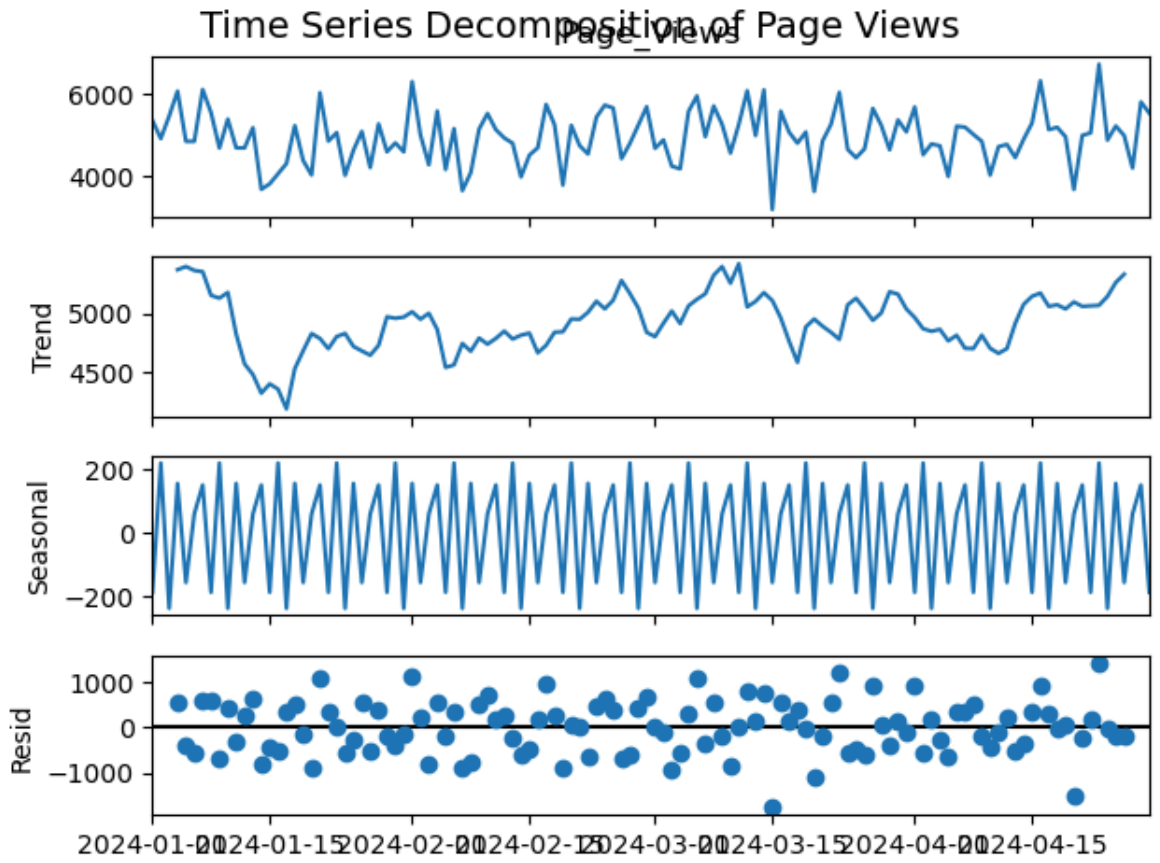


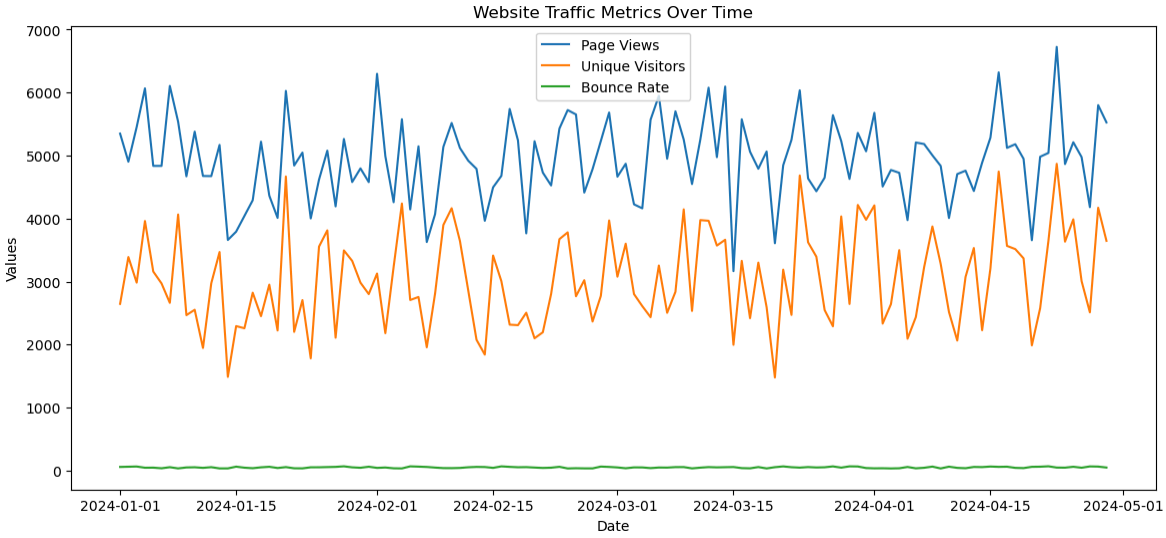
2.Write a Python program to perform Time Series Analysis on a website traffic dataset. The program should load and clean the dataset, decompose the time series to identify

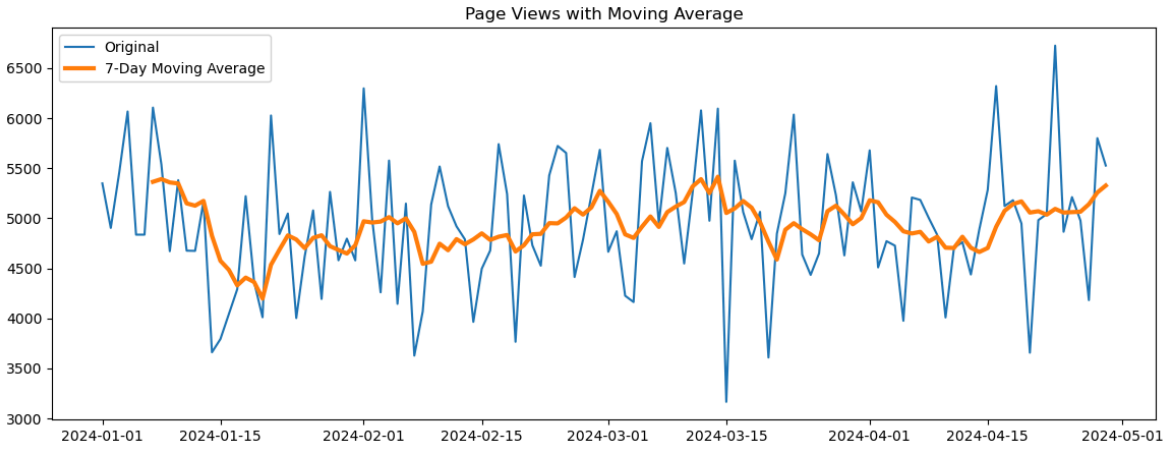
trends and seasonality, visualize key metrics using moving averages and seasonal plots, and detect anomalies using statistical methods.

In [7]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from statsmodels.tsa.seasonal import seasonal_decompose
from scipy import stats
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=
plt.title('Page Views with Moving Average')
plt.legend()
plt.show()
```

```
<Figure size 1200x800 with 0 Axes>
```





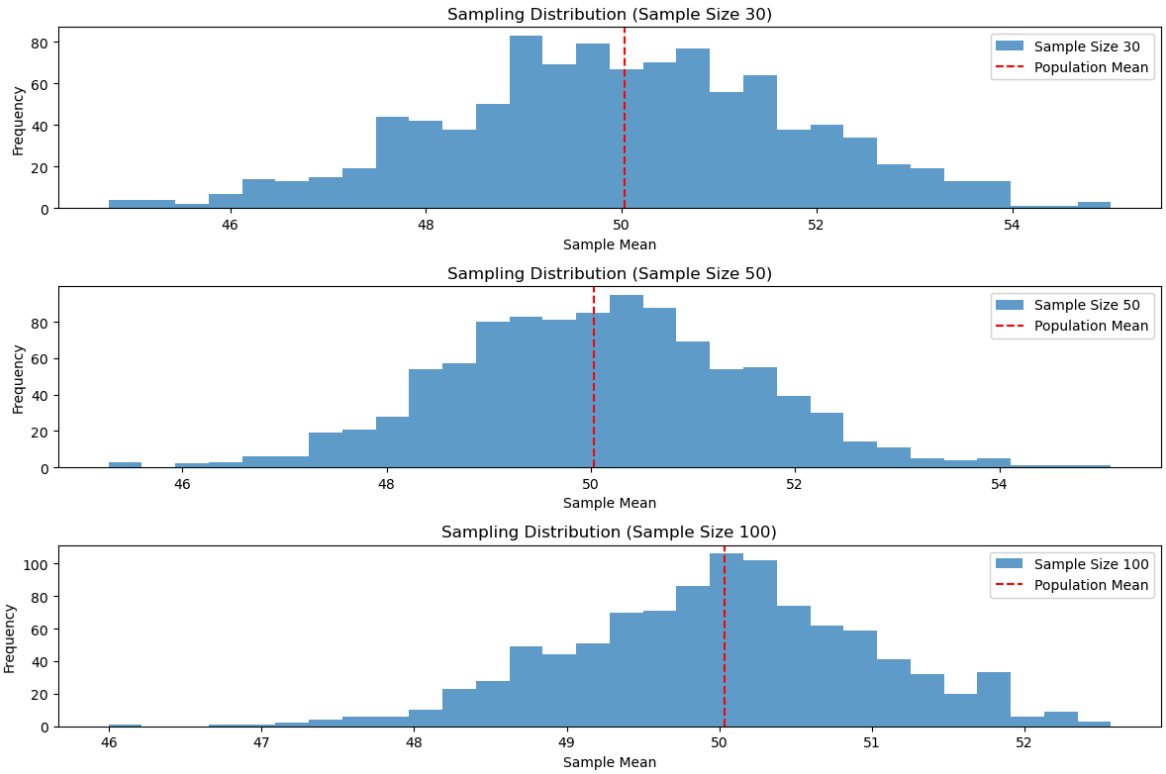


3. Random Sampling and Sampling Distribution To explore random sampling from a population and understand the concept of sampling distribution using Python in

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 [6]:
```
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)
        mean = np.mean(sample)
        sample_means[size].append(mean)
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=
    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
Sampling Distribution (Sample Size 50)
Sample Size 50
Population Mean
Frequency
Sample Mean
Sampling Distribution (Sample Size 100)
Sample Size 100
Population Mean
Frequency
Sample Mean

In [ ]:

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.

In [3]:
```
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(np.abs(z_statistic)))
print(f"Sample Mean: {sample_mean:.2f}")
print(f"Z-Statistic: {z_statistic:.4f}")
print(f"P-Value: {p_value:.4f}")
alpha = 0.05
if p_value < alpha:
    print("Reject the null hypothesis: The average weight is significantly diffe
else:
    print("Fail to reject the null hypothesis: There is no significant differenc
```

```
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.
```

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()
```

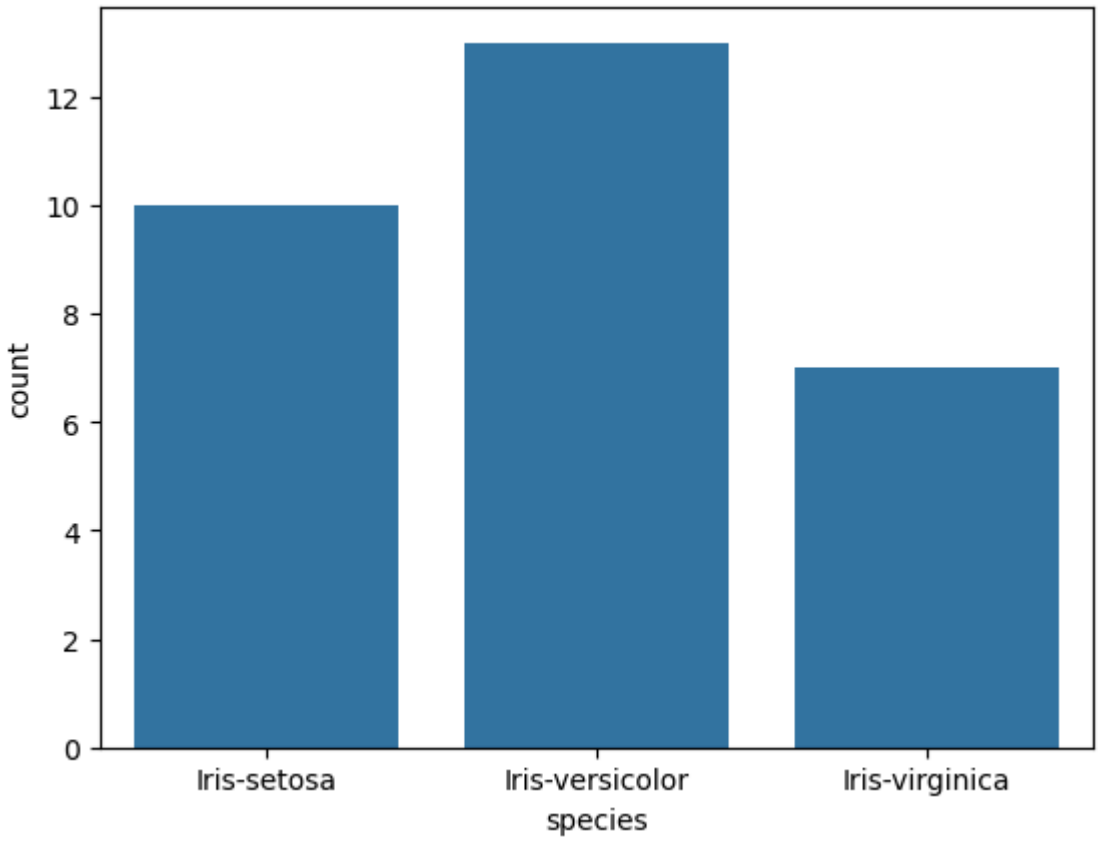
Out[11]:

	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()
```



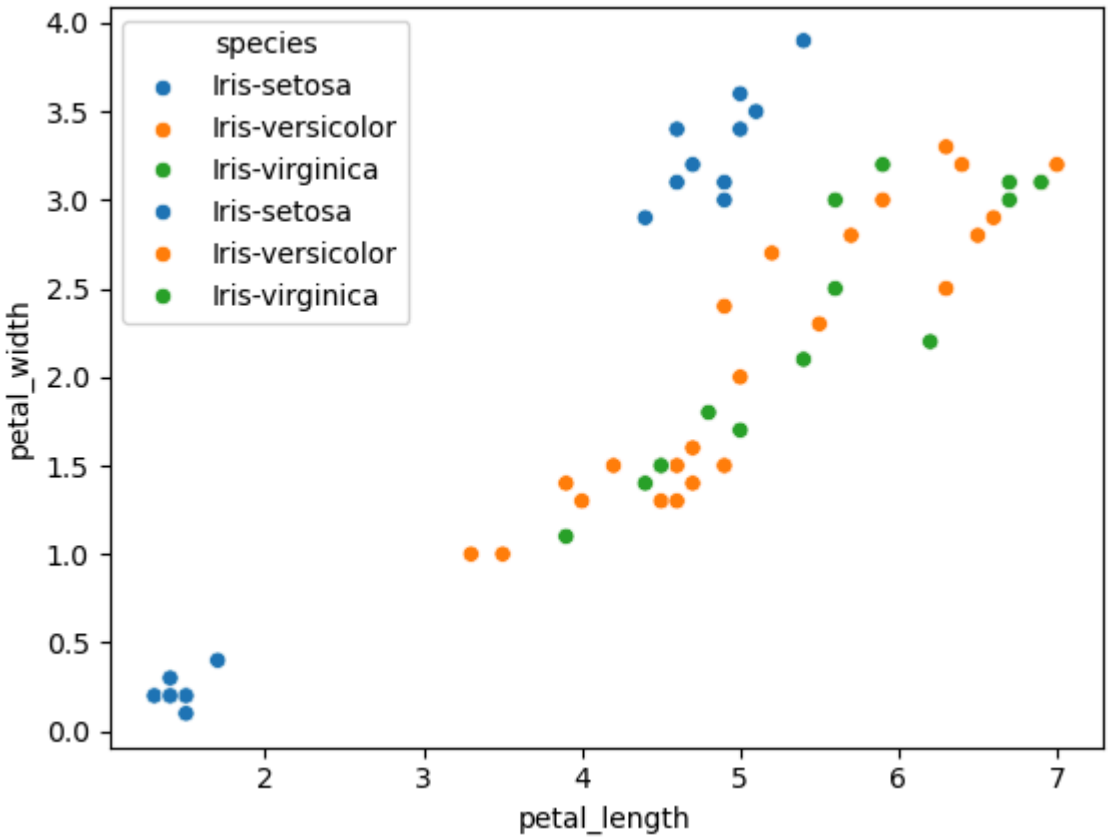
In [15]:
```
dummies=pd.get_dummies(data.species)

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

```
FinalDataset.head()
```

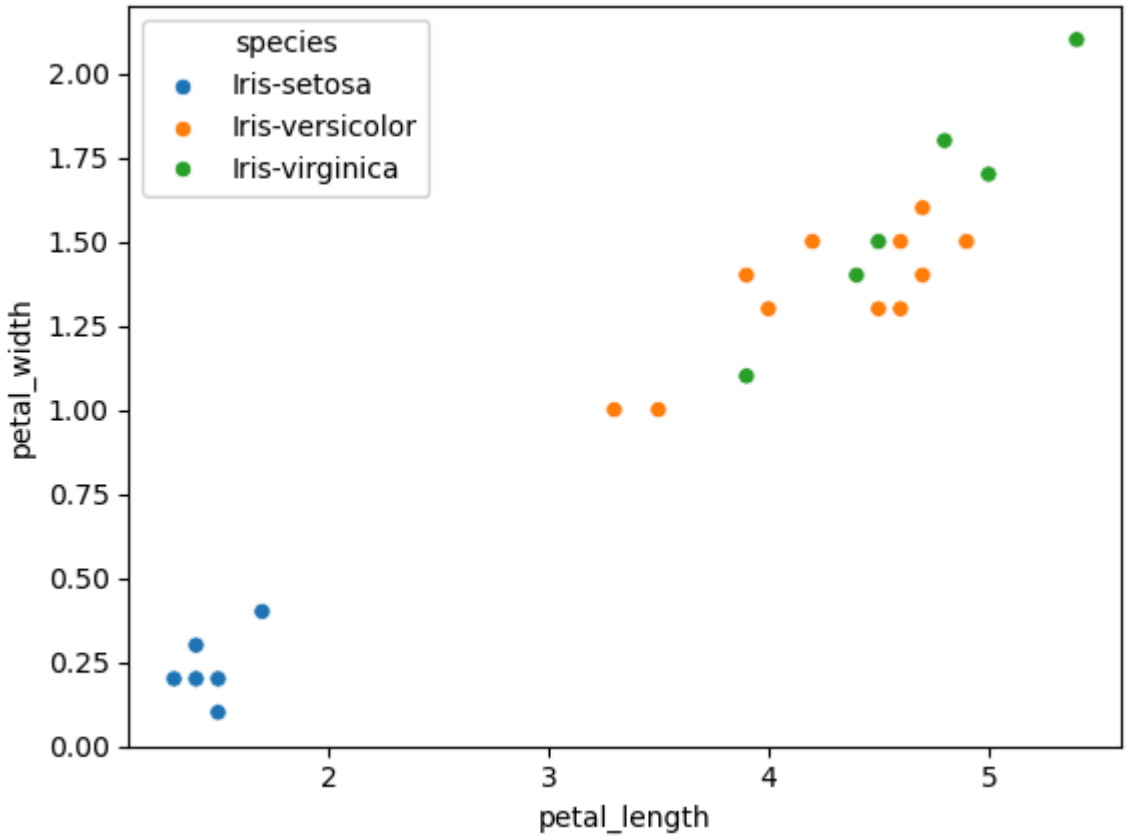
Out[15]:

	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

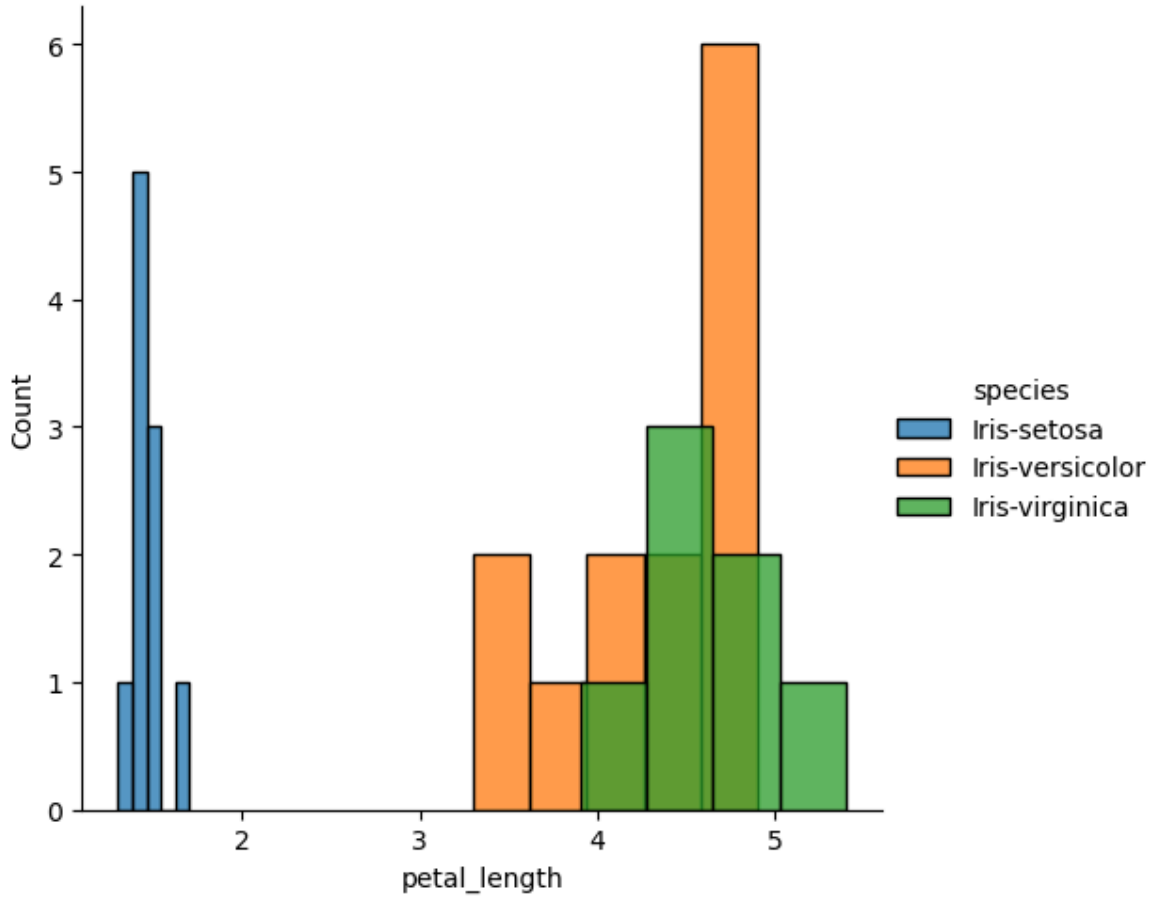
In [22]:
```
sns.scatterplot(x='sepal_length', y='sepal_width', hue='species', data=data)
plt.show()
```

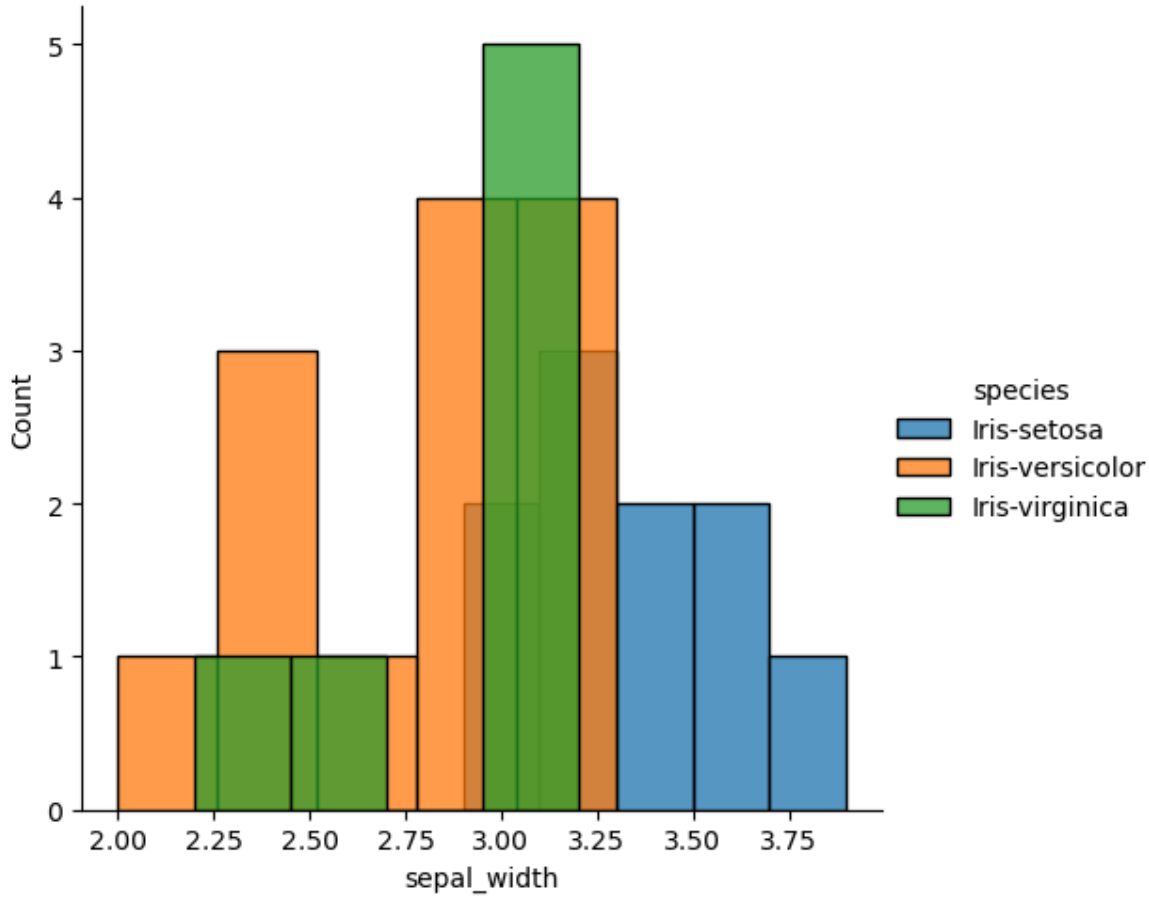


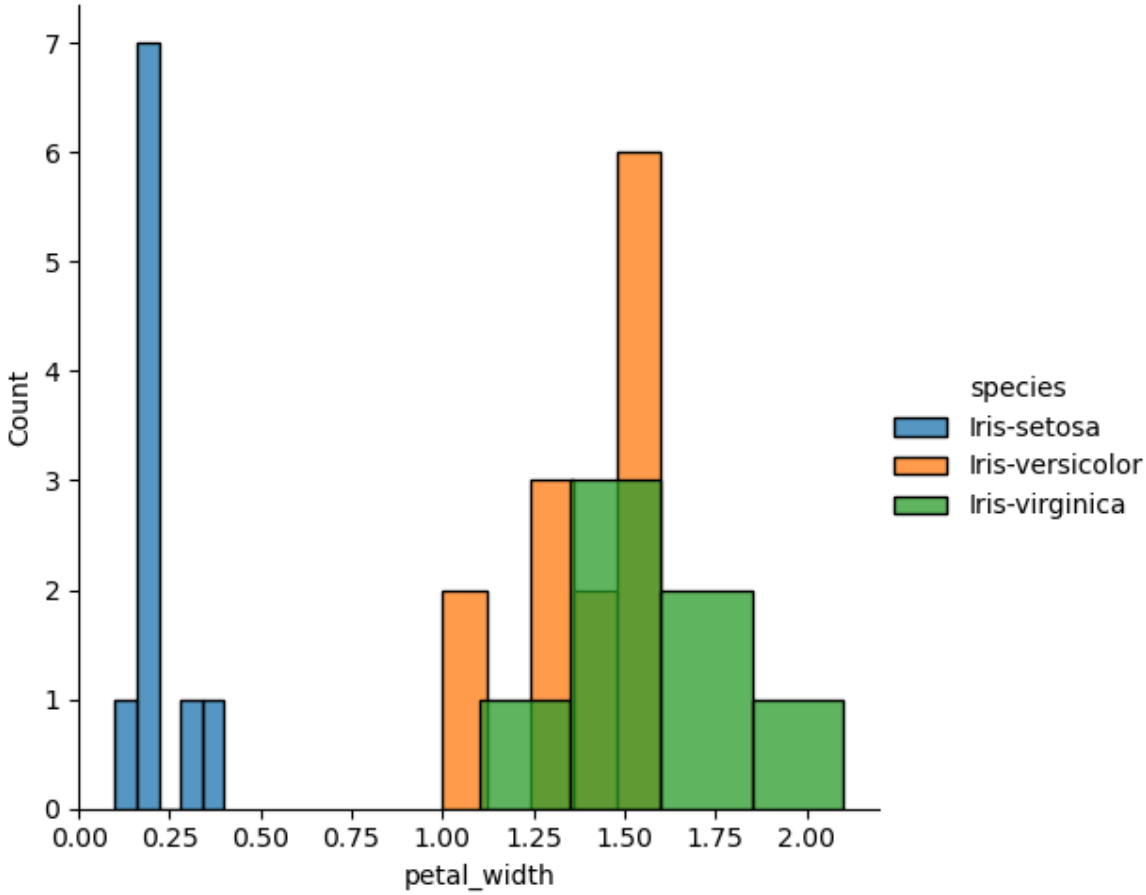
In [23]:
```
sns.scatterplot(x='petal_length', y='petal_width', hue='species', data=data)
plt.show()
```

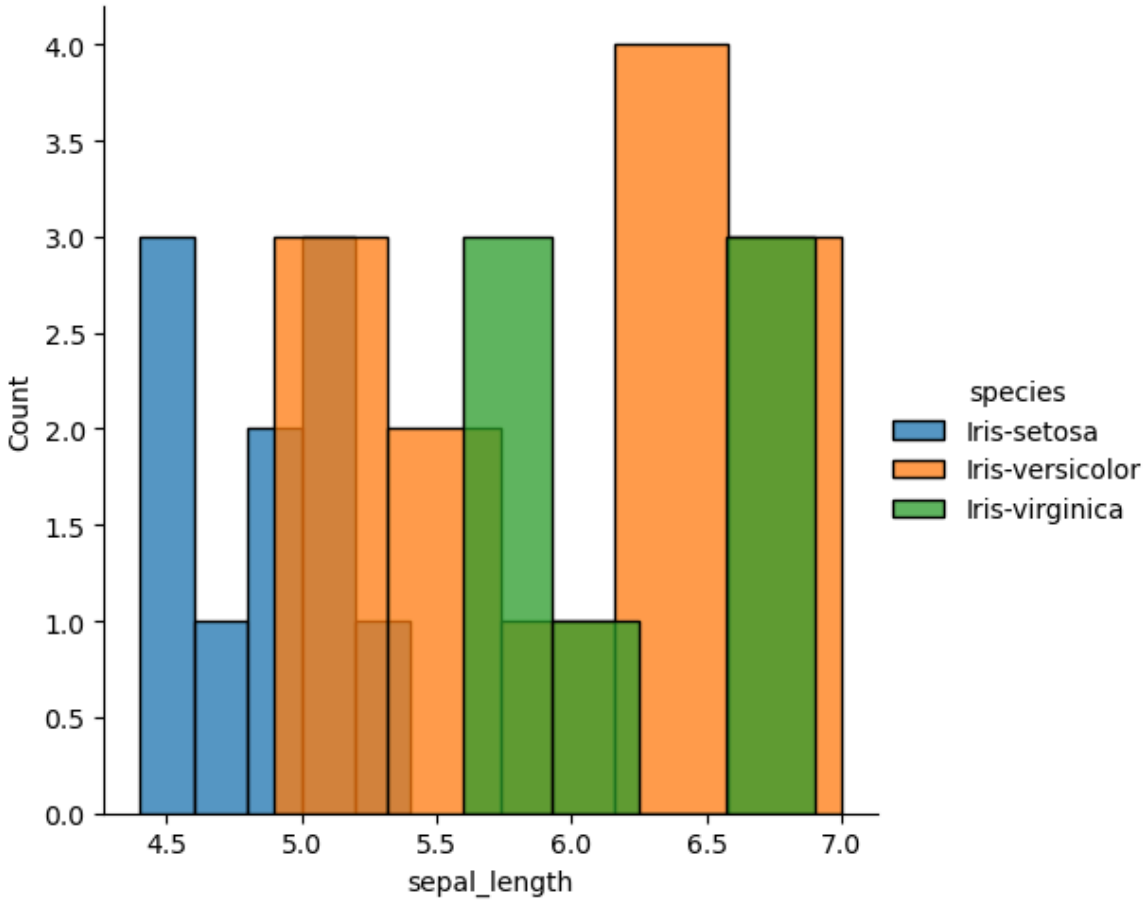


In [24]:
```
sns.FacetGrid(data, hue='species', height=5).map(sns.histplot, 'petal_length').a
plt.show()
sns.FacetGrid(data, hue='species', height=5).map(sns.histplot, 'sepal_width').ad
plt.show()
sns.FacetGrid(data, hue='species', height=5).map(sns.histplot, 'petal_width').ad
plt.show()
sns.FacetGrid(data,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

1.Experiment to understand array function in Data science. Description: Understand array function using Numpy library

In [4]:
```
import numpy as np
array = np.random.randint(1, 100, 9)
array = np.array([83, 25, 15, 47, 62, 15, 96, 39, 51])
np.sqrt(array)
```

Out[4]:
```
array([9.11043358, 5.        , 3.87298335, 6.8556546 , 7.87400787,
       3.87298335, 9.79795897, 6.244998  , 7.14142843])
```

In [5]:
```
array.ndim
```

Out[5]:
```
1
```

In [7]:
```
new_array = array.reshape(3, 3)
new_array
```

Out[7]:
```
array([[83, 25, 15],
       [47, 62, 15],
       [96, 39, 51]])
```

In [8]:
```
new_array.ndim
```

Out[8]:
```
2
```

In [9]:
```
new_array.ravel()
```

Out[9]:
```
array([83, 25, 15, 47, 62, 15, 96, 39, 51])
```

In [11]:
```
new_new = new_array.reshape(3, 3)
new_new
```

Out[11]:
```
array([[83, 25, 15],
       [47, 62, 15],
       [96, 39, 51]])
```

In [13]:
```
new_new[2,1:3]
```

Out[13]:
```
array([39, 51])
```

In [14]:
```
new_new[1:2, 1:3]
```

Out[14]:
```
array([[62, 15]])
```

In [15]:
```
new_array[0:3, 0:0]
```

Out[15]:
```
array([], shape=(3, 0), dtype=int64)
```

In [16]:
```
new_array[0:2, 0:1]
```

Out[16]:
```
array([[83],
       [47]])
```

In [17]:
```
new_array[0:3, 0:1]
```

Out[17]:
```
array([[83],
       [47],
       [96]])
```

In [18]:
```
new_array[1:3]
```

Out[18]:
```
array([[47, 62, 15],
       [96, 39, 51]])
```

In [19]:
```
import numpy as np
import pandas as pd
list=[[1,'Smith',50000], [2,'Jones',60000]]
df=pd.DataFrame(list)
print("\n",df)
df.columns=['Empd', 'Name' ,'Salary' ]
print("\n",df)
df.info()
df = pd.read_csv("50_Startups.csv")
df.info()
print(df.head())
df.tail()
```

```
   0      1      2
0  1  Smith  50000
1  2  Jones  60000

    Empd   Name  Salary
0     1  Smith   50000
1     2  Jones   60000
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2 entries, 0 to 1
Data columns (total 3 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Empd    2 non-null      int64
 1   Name    2 non-null      object
 2   Salary  2 non-null      int64
dtypes: int64(2), object(1)
memory usage: 180.0+ bytes
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50 entries, 0 to 49
Data columns (total 5 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   R&D Spend        50 non-null     float64
 1   Administration   50 non-null     float64
 2   Marketing Spend  50 non-null     float64
 3   State            50 non-null     object
 4   Profit           50 non-null     float64
dtypes: float64(4), object(1)
memory usage: 2.1+ KB
   R&D Spend  Administration  Marketing Spend       State     Profit
0  165349.20       136897.80        471784.10    New York  192261.83
1  162597.70       151377.59        443898.53  California  191792.06
2  153441.51       101145.55        407934.54     Florida  191050.39
3  144372.41       118671.85        383199.62    New York  182901.99
4  142107.34        91391.77        366168.42     Florida  166187.94
```

Out[19]:

	R&D Spend	Administration	Marketing Spend	State	Profit
45	1000.23	124153.04	1903.93	Florida	64926.08
46	1315.46	115816.21	297114.46	New York	49490.75
47	0.00	135426.92	0.00	California	42559.73
48	542.05	51743.15	45173.06	Florida	35673.41
49	0.00	116983.80	0.00	New York	14681.40

In [27]:
```
import pandas as pd
import numpy as np
df = pd.read_csv("Employee_Salary.csv")
print(df)
df.head()
```

```
   EmpID      Name Department  Salary
0    101     Smith         HR   50000
1    102     Jones    Finance   60000
2    103     Kavya         IT   55000
3    104  Kaushika         IT   58000
4    105      Ravi  Marketing   62000
5    106     Sneha      Sales   49000
6    107    Ramesh    Finance   65000
7    108     Divya         HR   53000
8    109      Asha      Sales   57000
9    110     Kiran         IT   60000
```

Out[27]:

	EmpID	Name	Department	Salary
0	101	Smith	HR	50000
1	102	Jones	Finance	60000
2	103	Kavya	IT	55000
3	104	Kaushika	IT	58000
4	105	Ravi	Marketing	62000

In [22]:
```
df.tail()
```

Out[22]:

	EmpID	Name	Department	Salary
5	106	Sneha	Sales	49000
6	107	Ramesh	Finance	65000
7	108	Divya	HR	53000
8	109	Asha	Sales	57000
9	110	Kiran	IT	60000

In [23]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10 entries, 0 to 9
Data columns (total 4 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   EmpID       10 non-null     int64
 1   Name        10 non-null     object
 2   Department  10 non-null     object
 3   Salary      10 non-null     int64
dtypes: int64(2), object(2)
memory usage: 452.0+ bytes
```

In [28]:
```
print(df['Salary'])
```

```
0    50000
1    60000
2    55000
3    58000
4    62000
5    49000
6    65000
7    53000
8    57000
9    60000
Name: Salary, dtype: int64
```

In [35]:
```
print("Mean:", df.Salary.mean())
print("Median:", df.Salary.median())
print("Mode:\n", df.Salary.mode())
print("Variance:", df.Salary.var())
print("Standard Deviation:", df.Salary.std())
df.describe()
```

```
Mean: 56900.0
Median: 57500.0
Mode:
 0    60000
Name: Salary, dtype: int64
Variance: 26766666.666666668
Standard Deviation: 5173.651192984184
```

Out[35]:

	EmpID	Name	Department	Salary
count	10.00000	10	10	10.000000
unique	NaN	10	5	NaN
top	NaN	Smith	IT	NaN
freq	NaN	1	3	NaN
mean	105.50000	NaN	NaN	56900.000000
std	3.02765	NaN	NaN	5173.651193
min	101.00000	NaN	NaN	49000.000000
25%	103.25000	NaN	NaN	53500.000000
50%	105.50000	NaN	NaN	57500.000000
75%	107.75000	NaN	NaN	60000.000000
max	110.00000	NaN	NaN	65000.000000

In [36]:
```
df.describe(include="all")
```

Out[36]:

	EmpID	Name	Department	Salary
count	10.00000	10	10	10.000000
unique	NaN	10	5	NaN
top	NaN	Smith	IT	NaN
freq	NaN	1	3	NaN
mean	105.50000	NaN	NaN	56900.000000
std	3.02765	NaN	NaN	5173.651193
min	101.00000	NaN	NaN	49000.000000
25%	103.25000	NaN	NaN	53500.000000
50%	105.50000	NaN	NaN	57500.000000
75%	107.75000	NaN	NaN	60000.000000
max	110.00000	NaN	NaN	65000.000000

In [41]:
```
empCol=df.columns
empCol
```

Out[41]:
```
Index(['EmpID', 'Name', 'Department', 'Salary'], dtype='object')
```

In [42]:
```
emparray=df. values
emparray
```

Out[42]:
```
array([[101, 'Smith', 'HR', 50000],
       [102, 'Jones', 'Finance', 60000],
       [103, 'Kavya', 'IT', 55000],
       [104, 'Kaushika', 'IT', 58000],
       [105, 'Ravi', 'Marketing', 62000],
       [106, 'Sneha', 'Sales', 49000],
       [107, 'Ramesh', 'Finance', 65000],
       [108, 'Divya', 'HR', 53000],
       [109, 'Asha', 'Sales', 57000],
       [110, 'Kiran', 'IT', 60000]], dtype=object)
```

In [43]:
```
employee_DF=pd.DataFrame(emparray,columns=empCol)
employee_DF
```

Out[43]:

	EmpID	Name	Department	Salary
0	101	Smith	HR	50000
1	102	Jones	Finance	60000
2	103	Kavya	IT	55000
3	104	Kaushika	IT	58000
4	105	Ravi	Marketing	62000
5	106	Sneha	Sales	49000
6	107	Ramesh	Finance	65000
7	108	Divya	HR	53000
8	109	Asha	Sales	57000
9	110	Kiran	IT	60000

In [ ]:

Exp No:15 Experiment to understand the data preprocessing in Data science

In [1]:
```
import numpy as np
import pandas as pd
data = {
    'Name': ['John', 'Anna', 'Peter', 'Linda', 'James', 'Anna', None],
    'Age': [28, 22, np.nan, 32, 40, 22, 35],
    'City': ['New York', 'Paris', 'Berlin', 'New York', None, 'Paris', 'Berlin']
    'Salary': [50000, 54000, 58000, np.nan, 62000, 54000, 58000]
}          df=pd.DataFrame(data)
df.to_csv('emp.csv',index=False)
df=pd.read_csv('emp.csv') df
```

Out[1]:

	Name	Age	City	Salary
0	John	28.0	New York	50000.0
1	Anna	22.0	Paris	54000.0
2	Peter	NaN	Berlin	58000.0
3	Linda	32.0	New York	NaN
4	James	40.0	NaN	62000.0
5	Anna	22.0	Paris	54000.0
6	NaN	35.0	Berlin	58000.0

In [2]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7 entries, 0 to 6
Data columns (total 4 columns):
 #      Column Non-Null Count Dtype
    --- ------ -------------- -----
0   Name          6 non-null      object
1   Age           6 non-null      float64
2   City          6 non-null      object
3   Salary 6 non-null             float64
dtypes: float64(2), object(2)
memory usage: 356.0+ bytes
```

In [4]:
```
df.City.mode()
```

Out[4]:
```
0      Berlin
1    New York
2       Paris
Name: City, dtype: object
```

In [5]:
```
df.City.mode()[0]
```

Out[5]: 'Berlin'

In [10]:
```
df.City.fillna(df.City.mode()[0],inplace=True)
df.Age.fillna(df.Age.median(),inplace=True)
```

```
df.Salary.fillna(round(df.Salary.mean()),inplace=True)
df
```

```
C:\Users\Kaviya\AppData\Local\Temp\ipykernel_21492\4129364907.py:2: FutureWarnin
g: A value is trying to be set on a copy of a DataFrame or Series through chained
assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work becau
se the intermediate object on which we are setting values always behaves as a cop
y.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.meth
od({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to pe
rform the operation inplace on the original object.


  df.Age.fillna(df.Age.median(),inplace=True)
C:\Users\Kaviya\AppData\Local\Temp\ipykernel_21492\4129364907.py:3: FutureWarnin g:
A value is trying to be set on a copy of a DataFrame or Series through chained
assignment using an inplace method. The behavior will change in pandas 3.0. This
inplace method will never work becau se the intermediate object on which we are
setting values always behaves as a cop y.


For example, when doing 'df[col].method(value, inplace=True)', try using 'df.meth
od({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to pe
rform the operation inplace on the original object.


  df.Salary.fillna(round(df.Salary.mean()),inplace=True)
```

Out[10]:

	Name	Age	City	Salar y
0	John	28.	New York	50000.0
1	Anna	0	Paris	54000.0
2	Peter	22.	Berlin	58000.0
3	Linda	0	New York	56000.0
4	James	30.	Berlin	62000.0
5	Anna	0	Paris	54000.0
6	NaN	32.	Berlin	58000.0

0

In [11]:
```
pd.get_dummies(df.City)
```

40.

0

22.

0

35.

0

Out[11]:

	Berlin	New York	Paris
0	False	True	False
1	False	False	True
2	True	False	False
3	False	True	False
4	True	False	False
5	False	False	True
6	True	False	False

In [15]:
```
updated_dataset=pd.concat([pd.get_dummies(df.City),df.iloc[:,[1,2,3]]],axis=1)
```

In [13]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7 entries, 0 to 6
Data columns (total 4 columns):
 #      Column Non-Null Count Dtype
    --- ------ -------------- -----
0   Name         6 non-null      object
1   Age          7 non-null      float64
2   City         7 non-null      object
3   Salary 7 non-null            float64
dtypes: float64(2), object(2)
memory usage: 356.0+ bytes
```

In [16]:
```
updated_dataset
```

Out[16]:

	Berlin	New York	Paris	Age	City	Salar y
0	False	True	False	28.0New	York	50000.0
1	False	False	True	22.0	Paris	30.0 54000.0
2	True	False	False	Berlin	32.0New	58000.0
3	False	True	False	York	40.0 Berlin	56000.0
4	True	False	False	22.0	Paris	35.0 62000.0
5	False	False	True	Berlin		54000.0
6	True	False	False			58000.0

EXPERIMENT-15

In [ ]:
```
Experiment to understand EDA-Quantitative and
Qualitative analysis.
```

In [18]:
```
import seaborn as sns
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

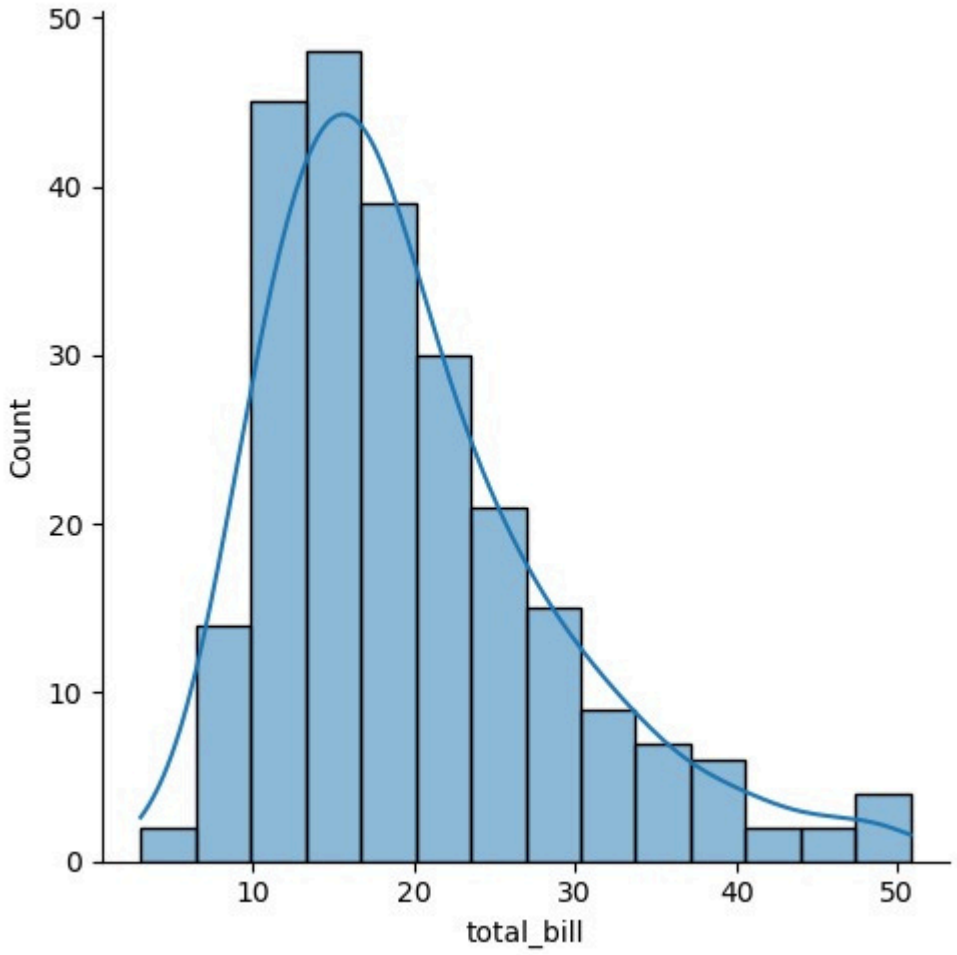
```
tips=sns.load_dataset('tips')
tips.head()
```

Out[18]:

	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4

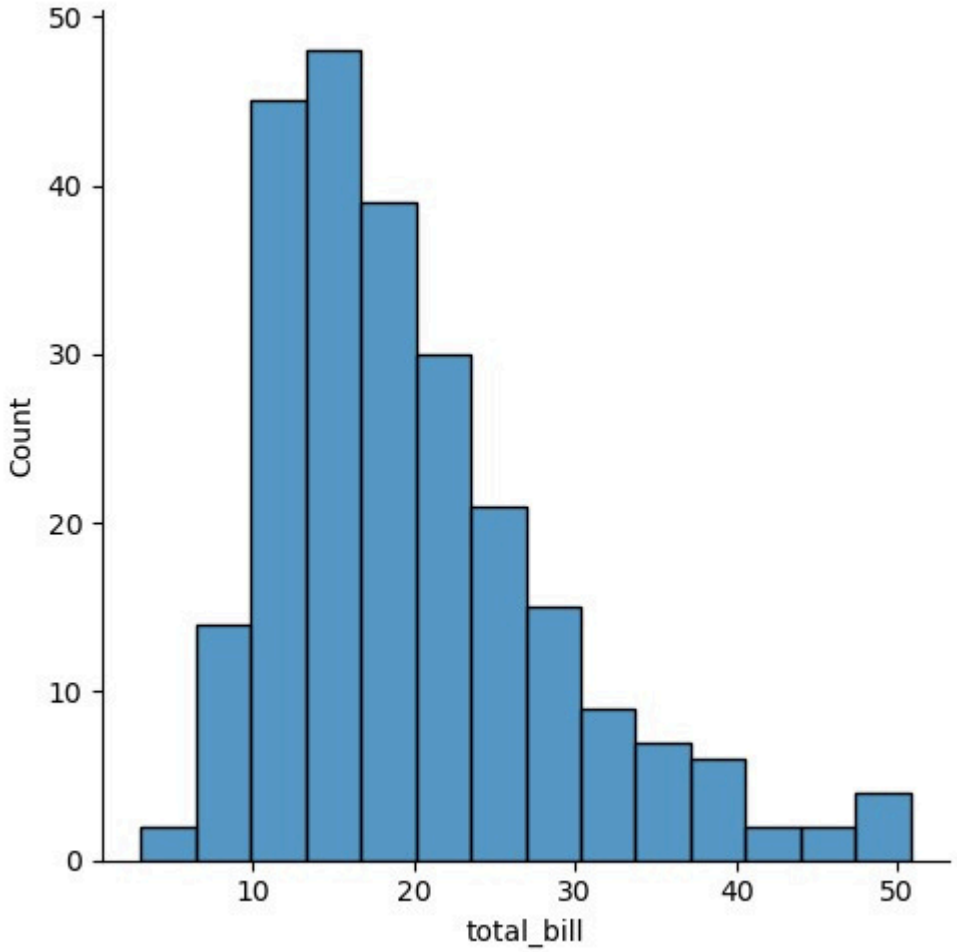
In [21]:

```
sns.displot(tips.total_bill,kde=True)
plt.show()
```

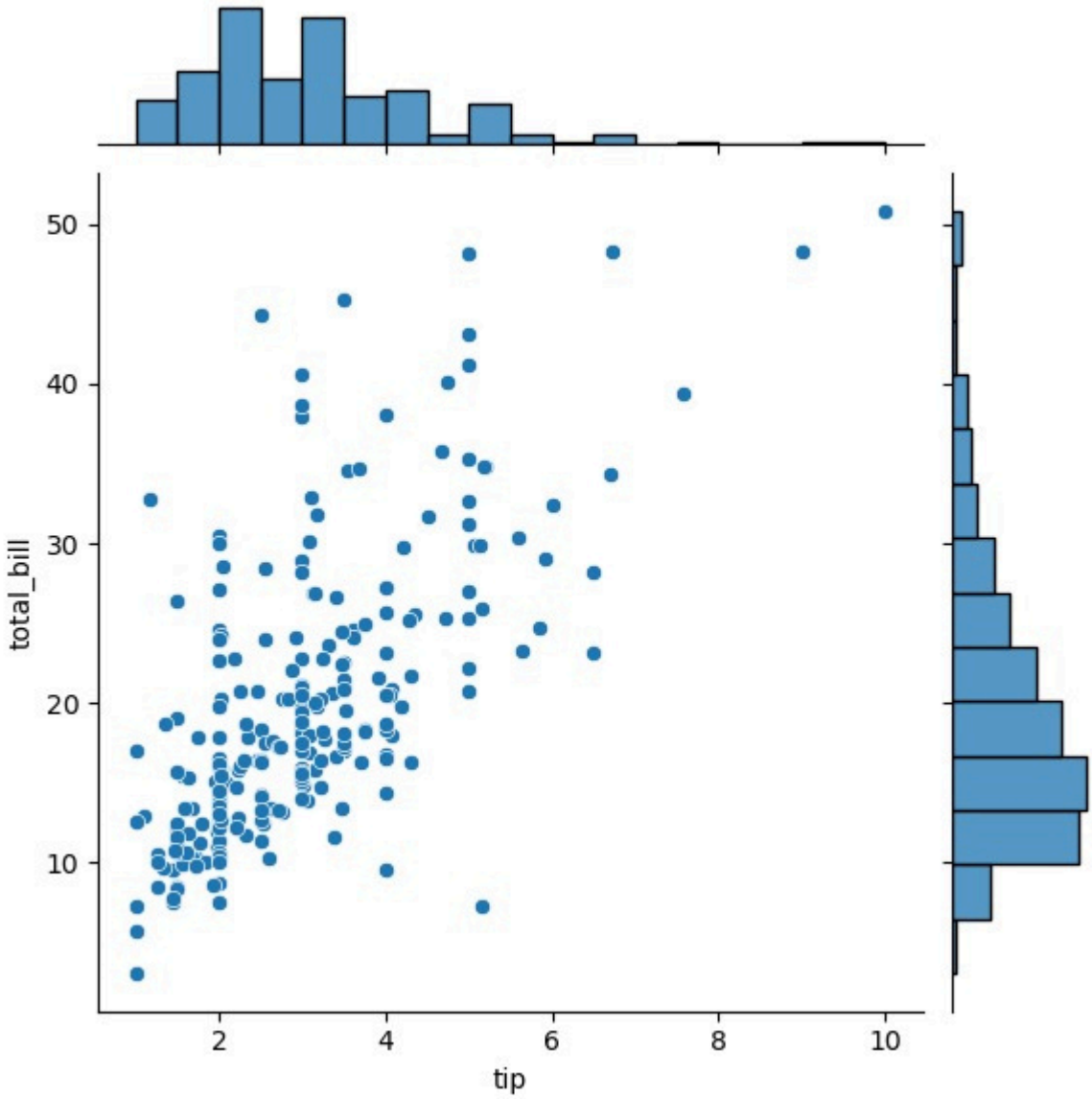


In [22]:

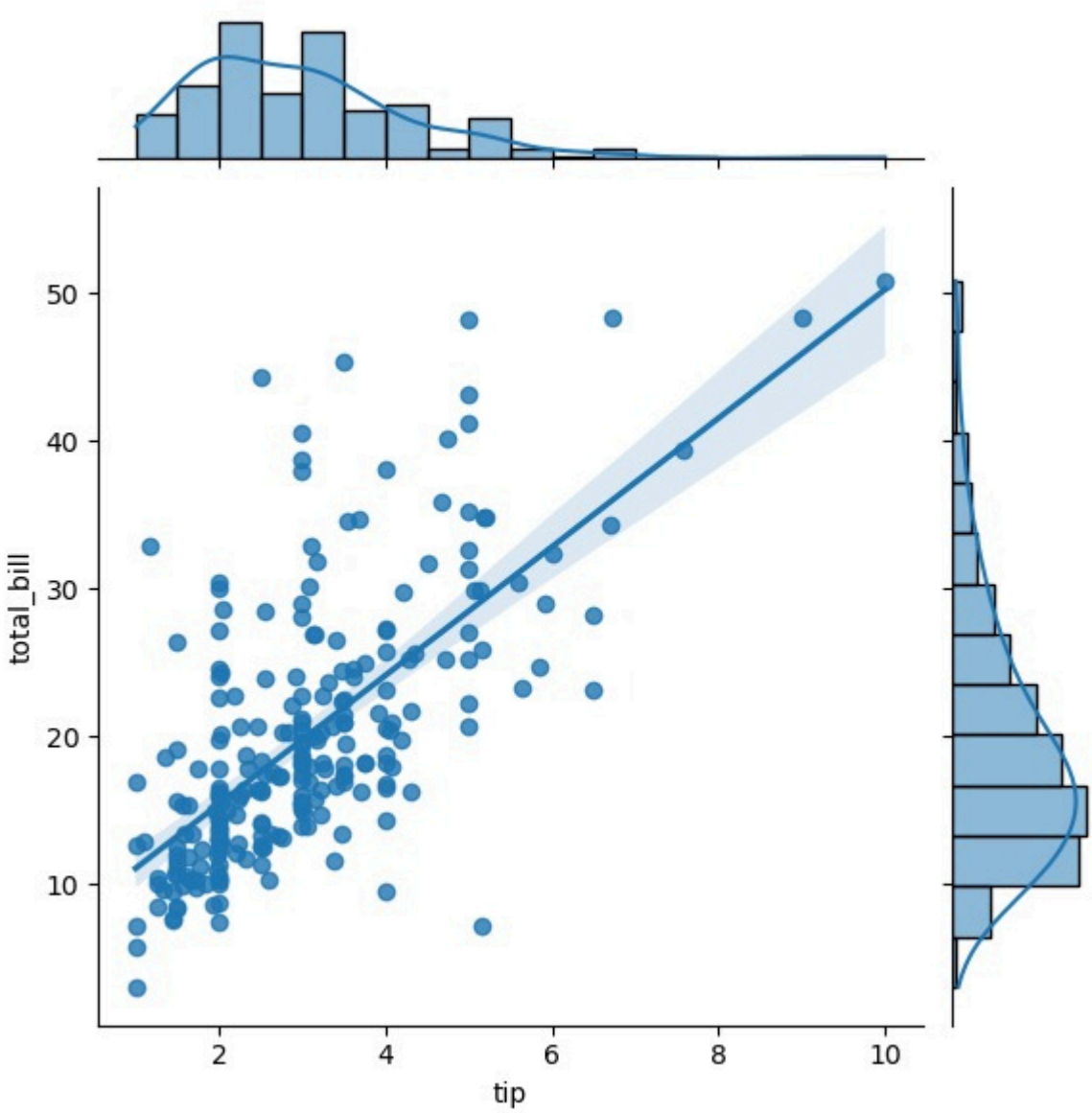
```
sns.displot(tips.total_bill,kde=False)
plt.show()
```



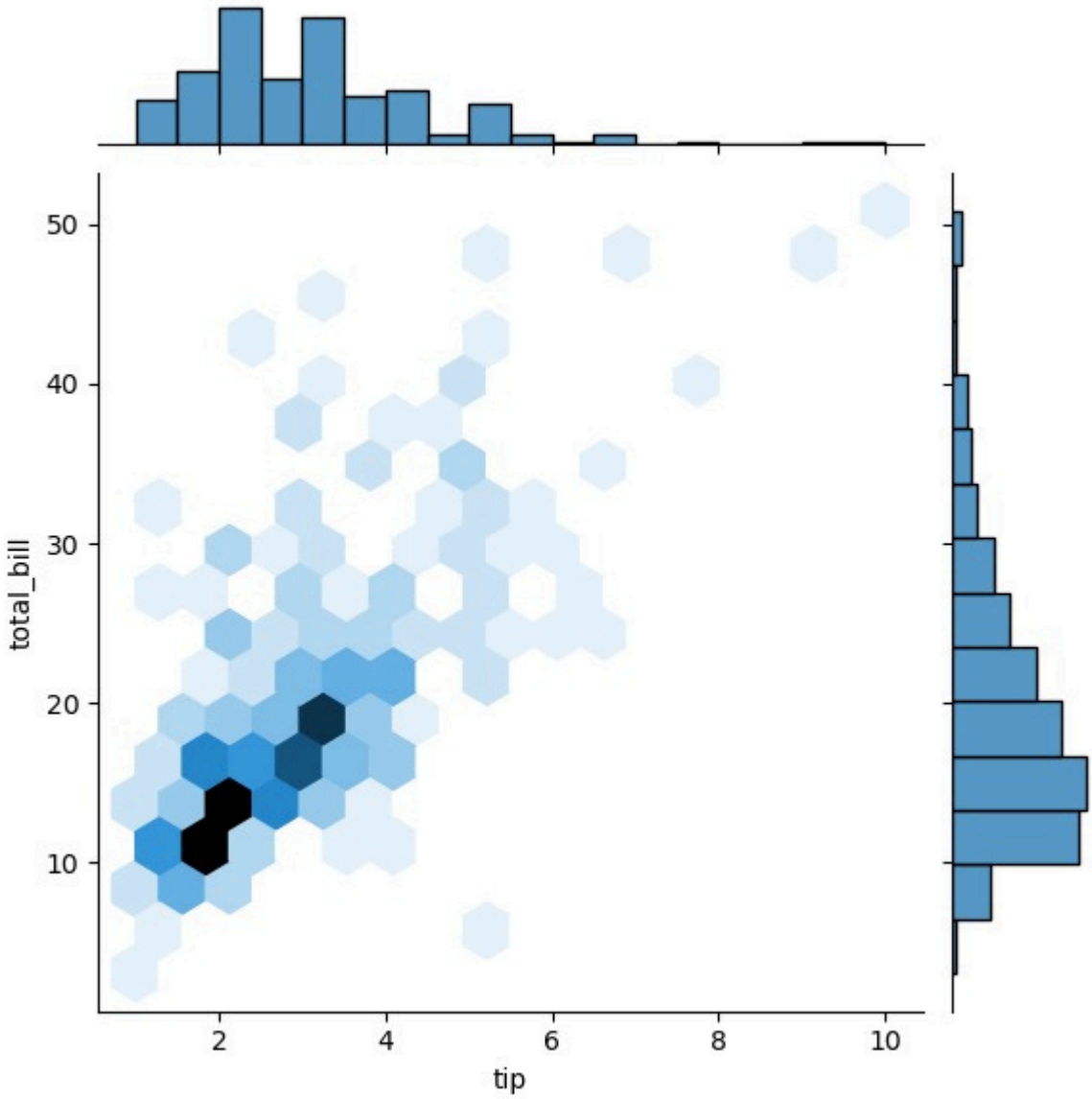
In [23]:
```
sns.jointplot(x=tips.tip,y=tips.total_bill)
plt.show()
```



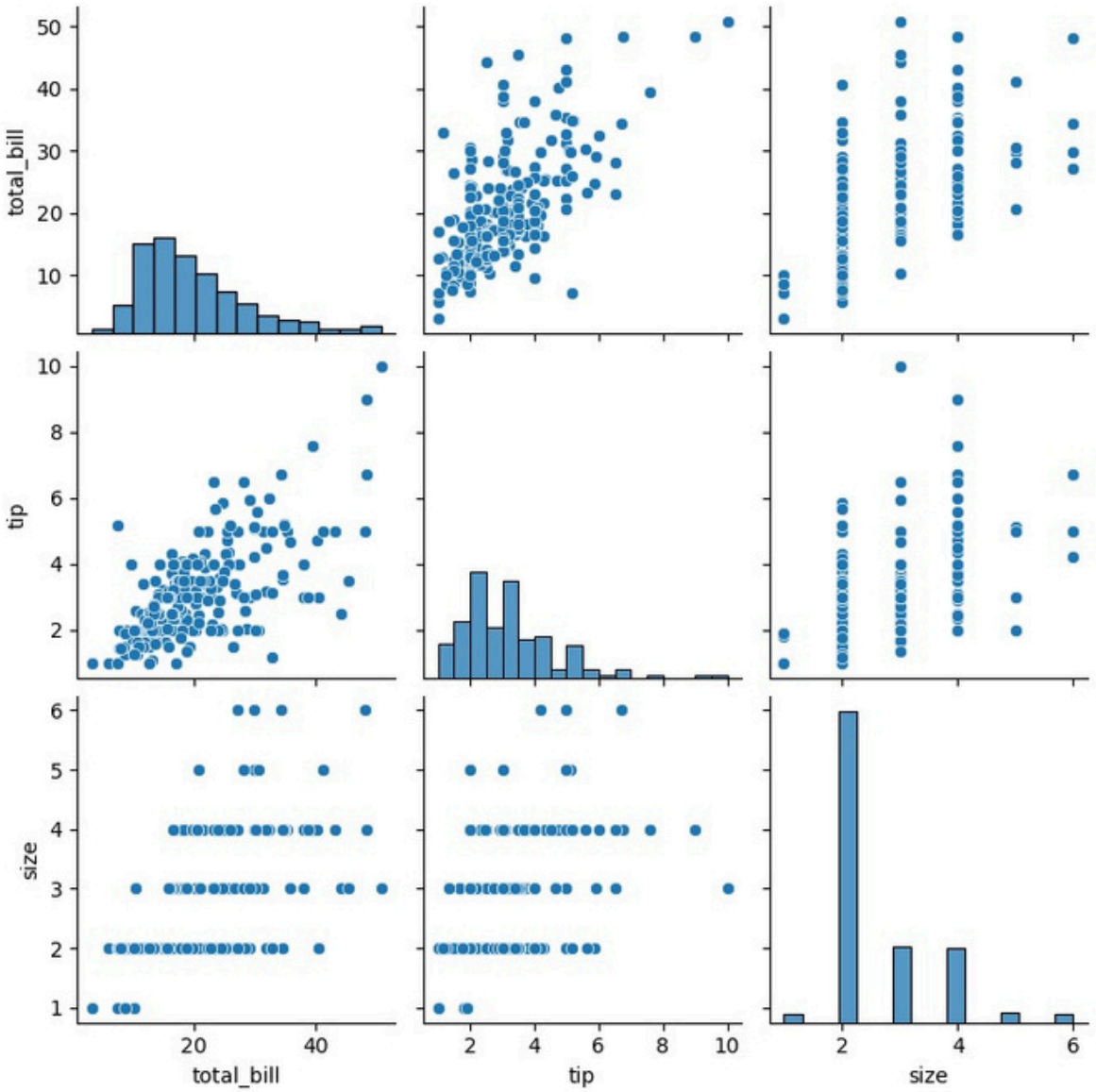
In [34]:
```
sns.jointplot(x=tips.tip,y=tips.total_bill,kind="reg")
plt.show()
```



In [28]:
```
sns.jointplot(x=tips.tip,y=tips.total_bill,kind="hex")
plt.show()
```



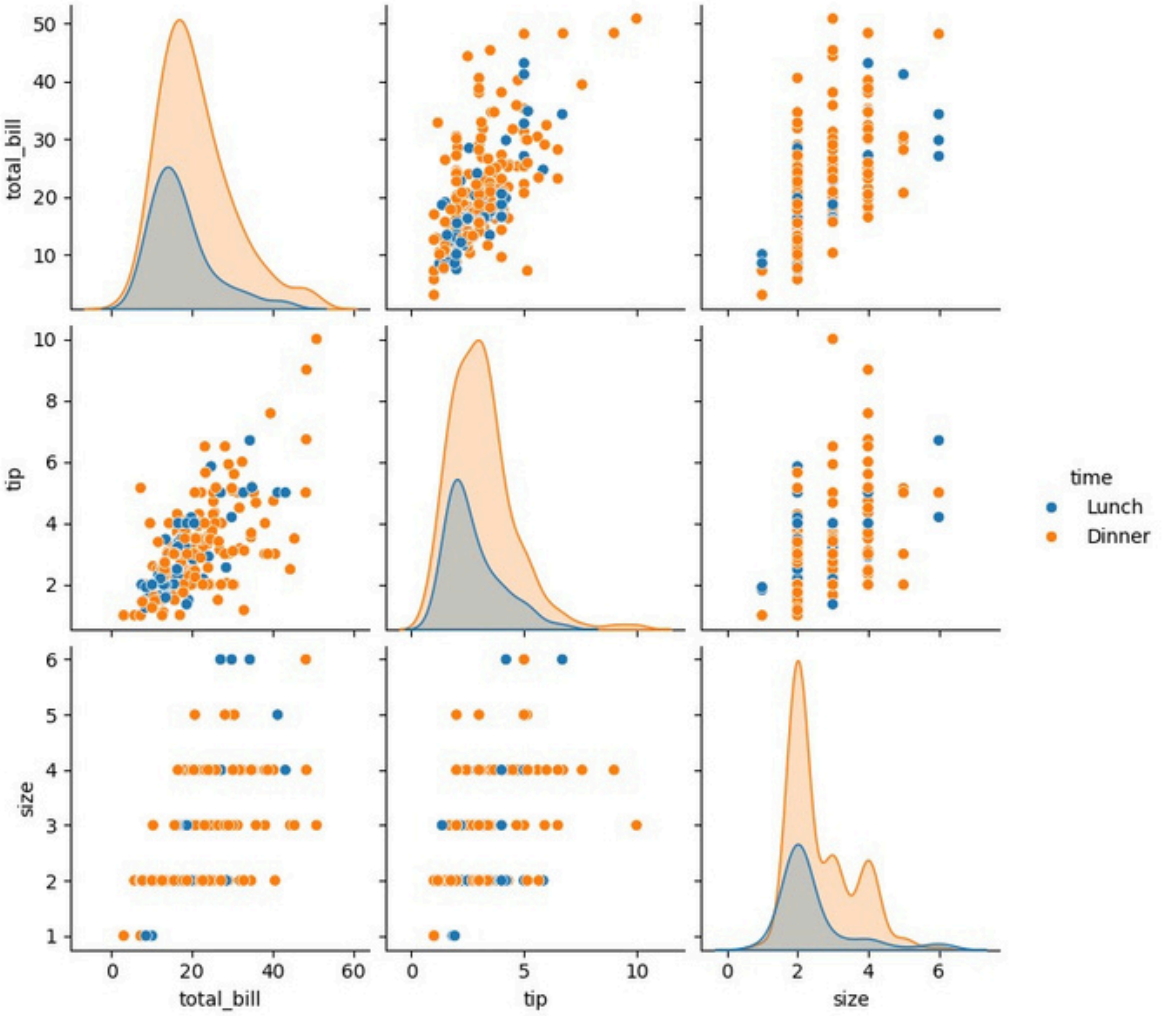
In [29]:
```
sns.pairplot(tips)
plt.show()
```



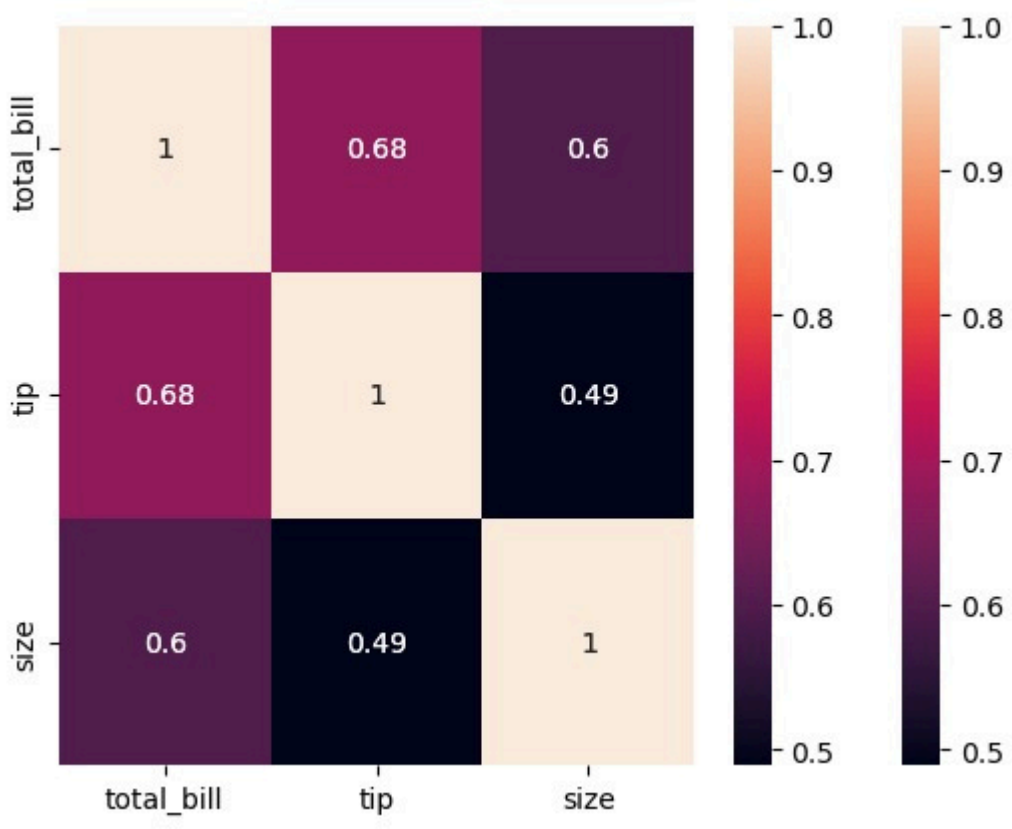
In [30]:
```
tips.time.value_counts()
```

Out[30]:
```
time
Dinner    176
Lunch      68
Name: count, dtype: int64
```

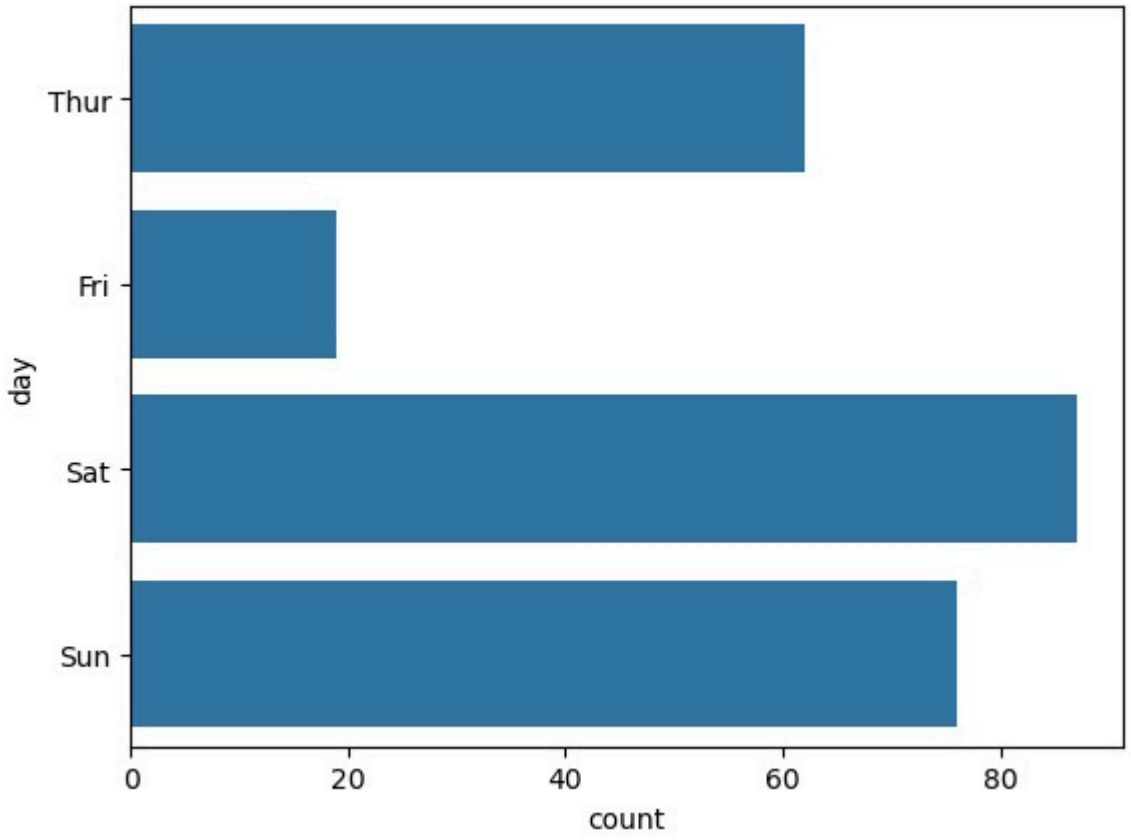
In [33]:
```
sns.pairplot(tips,hue='time')
plt.show()
```



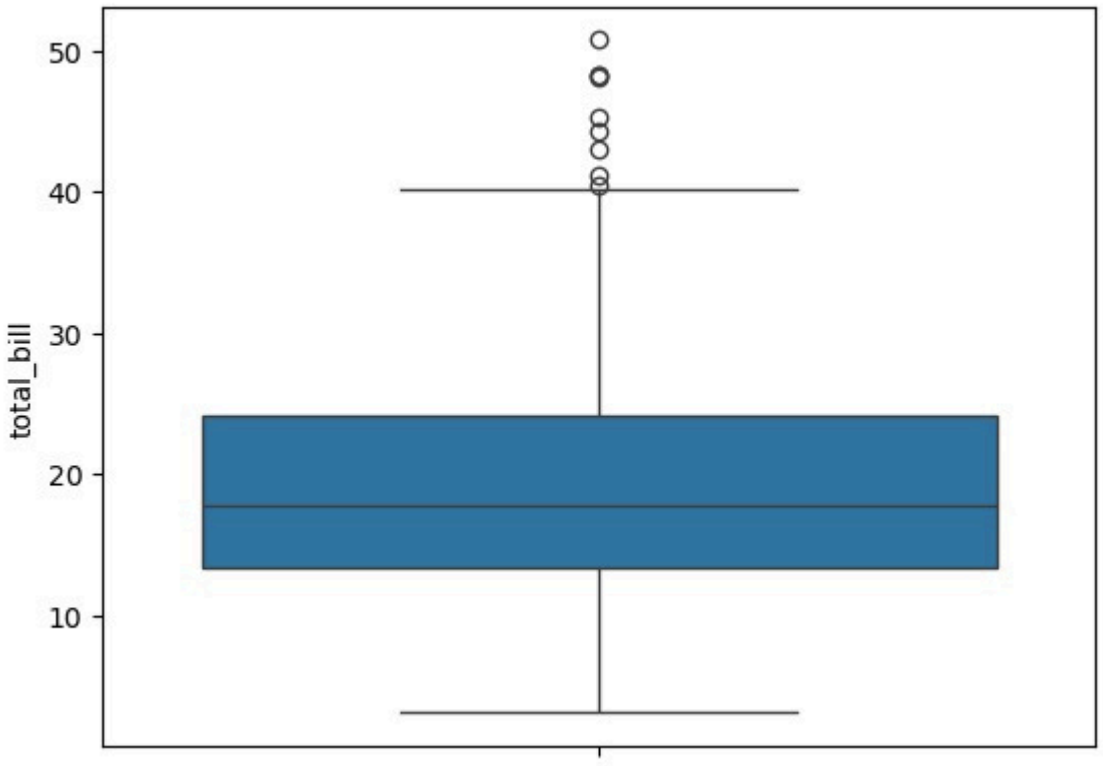
In [36]:
```
sns.heatmap(tips.corr(numeric_only=True),annot=True)
plt.show()
```



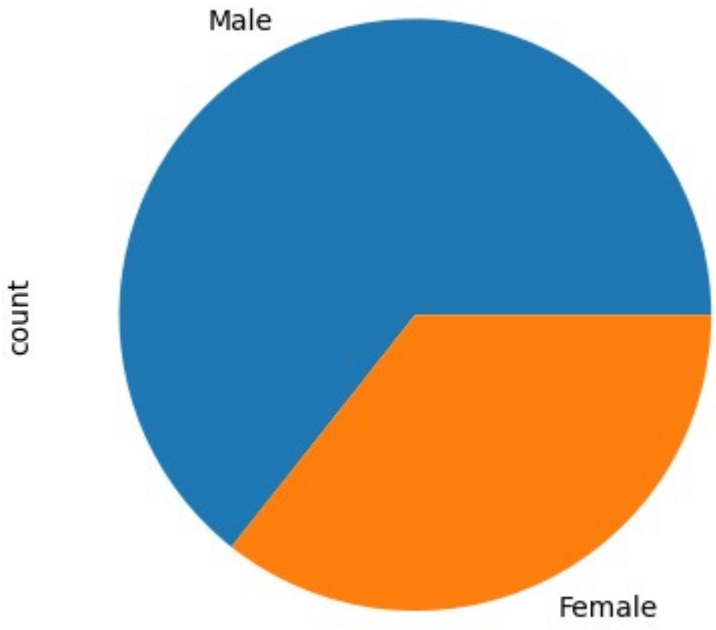
In [37]:
```
sns.countplot(tips.day)
plt.show()
```



In [38]:
```
sns.boxplot(tips.total_bill)
plt.show()
```



In [39]:
```
tips.sex.value_counts().plot(kind='pie')
plt.show()
```



EXPERIMENT-17

In [ ]:
```
Experiment to understand Linear Regression for a given
data set.
```

In [42]:
```
import numpy as np
import pandas as pd
data = {
    'EmployeeID': [1,2,3,4,5,6,7,8,9,10],
    'Name': ['John Smith','Priya Sharma','David Lee','Sarah Johnson','Karan Pate
    'Aisha Khan','Michael Brown','Meena Reddy','Rajesh Gupta','Emily Da
    'Gender': ['Male','Female','Male','Female','Male','Female','Male','Female','
    'Age': [28,32,45,29,38,26,50,35,31,40],
    'Department': ['IT','HR','Finance','IT','Marketing','HR','Management','Finan
    'Experience': [3,6,20,4,12,2,25,10,5,15],
    'Education': ['Bachelor','Master','Master','Bachelor','MBA','Bachelor','MBA'
    'Salary': [45000,58000,95000,52000,74000,40000,120000,83000,60000,88000]
}               df=pd.DataFrame(data)
df.to_csv('sal_data.csv',index=False)
df=pd.read_csv('sal_data.csv') df
```

Out[42]:

	EmployeeID	Name	Gender	Age	Department	Experience	Education	Salar y
0	1	John Smith	Male	28	IT	3	Bachelor	45000
1	2	Priya Sharma	Female	32	HR	6	Master	58000
2	3	David Lee	Male	45	Finance	20	Master	95000
3	4	Sarah Johnson	Female	29	IT	4	Bachelor	52000
4	5	Karan Patel	Male	38	Marketing	12	MBA	74000
5	6	Aisha Khan	Female	26	HR	2	Bachelor	40000
6	7	Michael Brown	Male	50	Management	25	MBA	120000
7	8	Meena Reddy	Female	35	Finance	10	Master	83000
8	9	Rajesh Gupta	Male	31	IT	5	Bachelor	60000
9	10	Emily Davis	Female	40	Marketing	15	MBA	88000

In [43]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10 entries, 0 to 9
Data columns (total 8 columns):
 #   Column       Non-Null Count Dtype  
---  ------       -------------- -----  
 0   EmployeeID 10 non-null      int64  
 1   Name         10 non-null     object 
 2   Gender       10 non-null     object 
 3   Age          10 non-null     int64  
 4   Department 10 non-null      object 
 5   Experience 10 non-null      int64  
 6   Education    10 non-null     object 
 7   Salary       10 non-null     int64  
dtypes: int64(4), object(4)
memory usage: 772.0+ bytes
```

In [44]:
```
df.dropna(inplace=True)
```

In [45]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10 entries, 0 to 9
Data columns (total 8 columns):
 #   Column         Non-Null Count Dtype
--- ------         -------------- -----
 0   EmployeeID 10 non-null       int64
 1   Name       Gender 10 non-null    object
 2   Department  10 non-null    object
 3   Experience 10 non-null      int64
 4                              object
 5                               int64
 6   Education   10  non-null    object
 7   Salary      10 non-null     int64
dtypes: int64(4), object(4)
memory usage: 772.0+ bytes
```

In [46]:
```
df.describe()
```

Out[46]:

	EmployeeID	Age	Experience	Salar y
count	10.00000	10.000000	10.000000	10.000000
mean	5.50000	35.400000	10.200000	71500.000000
std	3.02765	7.805981	7.771744	25176.046817
min	1.00000	26.000000	2.000000	40000.000000
25%	3.25000	29.500000	4.250000	53500.000000
50%	5.50000	33.500000	8.000000	67000.000000
75%	7.75000	39.500000	14.250000	86750.000000
max	10.00000	50.000000	25.000000	120000.000000

In [56]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
data = {
    'EmployeeID': [1,2,3,4,5,6,7,8,9,10],
    'Name': ['John Smith','Priya Sharma','David Lee','Sarah Johnson','Karan Pate
    'Aisha Khan','Michael Brown','Meena Reddy','Rajesh Gupta','Emily Da
    'Gender': ['Male','Female','Male','Female','Male','Female','Male','Female','
    'Age': [28,32,45,29,38,26,50,35,31,40],
    'Department': ['IT','HR','Finance','IT','Marketing','HR','Management','Finan
    'Experience': [3,6,20,4,12,2,25,10,5,15],
    'Education': ['Bachelor','Master','Master','Bachelor','MBA','Bachelor','MBA'
    'Salary': [45000,58000,95000,52000,74000,40000,120000,83000,60000,88000]
}  df  =  pd.DataFrame(data)  features  =  df[['Experience']].values  label  =
df[['Salary']].values      x_train,      x_test,      y_train,      y_test      =
train_test_split(features,  label,  test_size=0  model  =  LinearRegression()
model.fit(x_train,  y_train)  y_pred  =  model.predict(x_test)  print("Training
complete ") print("\nTest Data (Experience):") print(x_test)
```

```
print("\nPredicted Salaries:")
print(y_pred)
```

```
Training complete

Test Data (Experience):
[[5]
 [6]]

Predicted Salaries:
[[54108.37817064]
 [57326.67179093]]
```

In [57]:
```
model.score(x_train, y_train)
```

Out[57]: 0.9512880612893012

In [58]:
```
model.score(x_test, y_test)
```

Out[58]: -16.58228932867233

In [59]:
```
model.coef_
```

Out[59]: array([[3218.29362029]])

In [60]:
```
model.intercept_
```

Out[60]: array([38016.91006918])

Kmeans clustering

In [63]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
data = {
    'CustomerID': [1, 2, 3, 4, 5],
    'Gender': ['Male', 'Male', 'Female', 'Female', 'Female'],
    'Age': [19, 21, 20, 23, 31],
    'Annual Income (k$)': [15, 15, 16, 16, 17],
    'Spending Score (1-100)': [39, 81, 6, 77, 40]
}
df.to_csv('cust_data.csv',index=False)
df=pd.read_csv('cust_data.csv')
df.info()
```

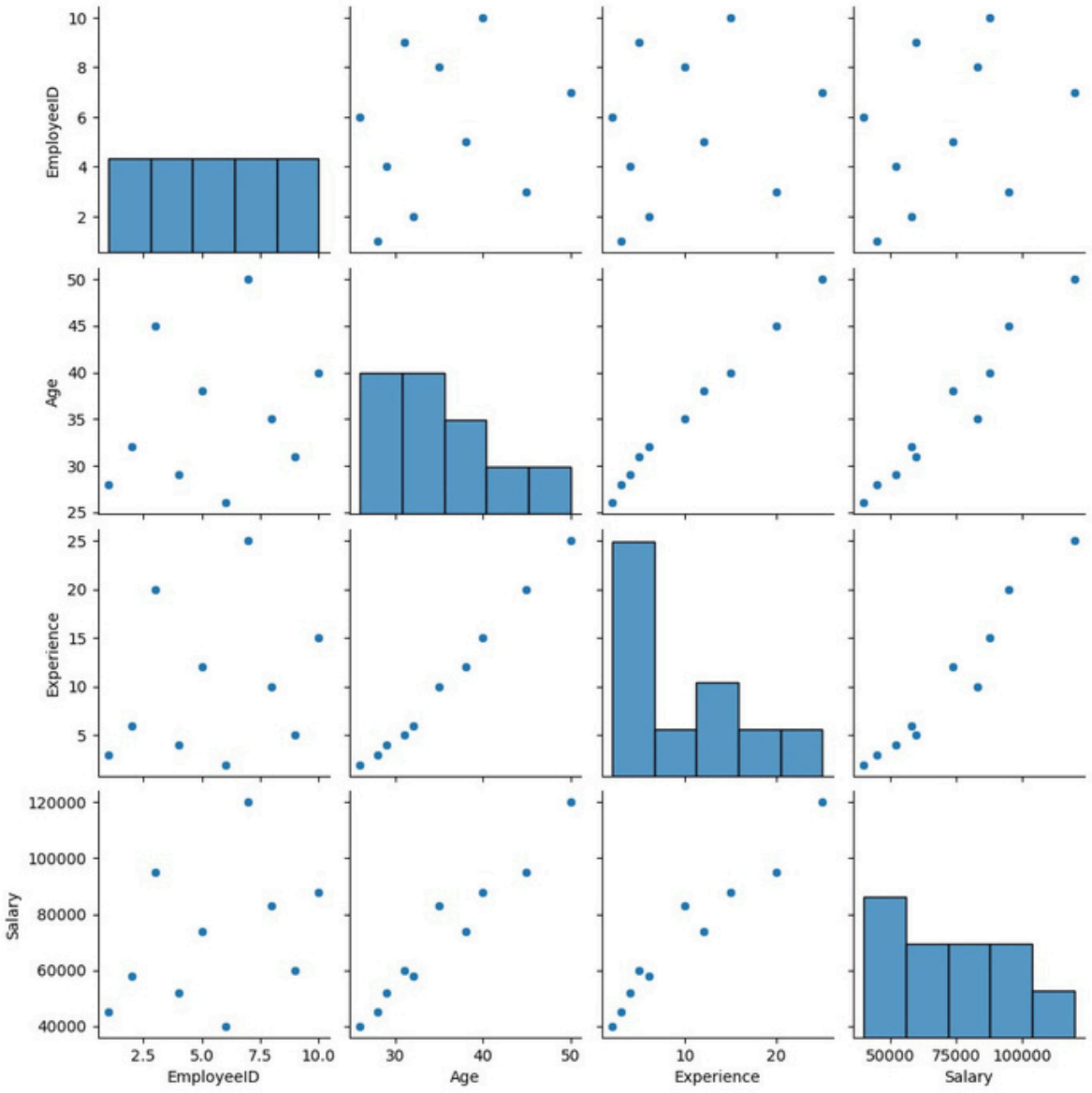
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10 entries, 0 to 9
Data columns (total 8 columns):
 #   Column        Non-Null Count Dtype
--- ------        -------------- -----
 0   EmployeeID 10 non-null       int64
 1   Name      Gender 10 non-null    object
 2   Department  10 non-null    object
 3   Experience 10 non-null      int64
 4                               object
 5                               int64
 6   Education   10  non-null    object
 7   Salary      10 non-null     int64
dtypes: int64(4), object(4)
memory usage: 772.0+ bytes
```

In [64]:
```
df.head()
```

Out[64]:

	EmployeeID	Name	Gender	Age	Department	Experience	Education	Salar y
0	1	John Smith	Male	28	IT	3	Bachelor	45000
1	2	Priya Sharma	Female	32	HR	6	Master	58000
2	3	David Lee	Male	45	Finance	20	Master	95000
3	4	Sarah Johnson	Female	29	IT	4	Bachelor	52000
4	5	Karan Patel	Male	38	Marketing	12	MBA	74000

In [68]:
```
sns.pairplot(df)
plt.show()
```


EmployeeID
Age
Experience
Salary
2.5
5.0
7.5
10.0
30
40
50
10
20
50000
75000
100000

In [ ]: