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 (2025)

Name of the Student : B J KAVIYA

Register Number : 240701246

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 [7]:
```
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.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 [10]:
```
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
```

3. 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 [16]:
```
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 [22]:
```
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 [37]:
```
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 [40]:
```
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
```

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 [45]:
```
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 [48]:
```
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
```

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 [50]:
```
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
```

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). Submit your .csv file along with the code used to generate it. Company Employee Records Employee_ID, Name, Department, Designation, Salary, Join_Date, Email, Location

In [2]:
```
import pandas as pd
data = {
'Employee_ID': ['E001', 'E002', 'E003', 'E004', 'E005',
    'E006', 'E007', 'E008', 'E009', 'E010',
    'E011', 'E012', 'E013', 'E014', 'E015',
    'E016', 'E017', 'E018', 'E019', 'E020',
    'E021', 'E022', 'E023', 'E024', 'E025'],
'Name': ['Amit', 'Priya', 'Rahul', 'Sneha', 'Karan',
    'Anita', 'Ravi', 'Pooja', 'Vikram', 'Neha',
    'Arjun', 'Kavya', 'Suresh', 'Meena', 'Nitin',
    'Divya', 'Manoj', 'Ritu', 'Rakesh', 'Alka',
    'Tarun', 'Shreya', 'Deepak', 'Payal', 'Sunny'],
'Department': ['IT', 'HR', 'Sales', 'Finance', 'Marketing',
    'Operations', 'Customer Support', 'IT', 'HR', 'Sales',
    'Finance', 'Marketing', 'Operations', 'Customer Support', 'IT',
    'HR', 'Sales', 'Finance', 'Marketing', 'Operations',
    'Customer Support', 'IT', 'HR', 'Sales', 'Finance'],
    'Designation': ['Engineer', 'Executive', 'Manager', 'Analyst', 'Assistant',
    'Coordinator', 'Specialist', 'Engineer', 'Executive', 'Manager',
    'Analyst', 'Assistant', 'Coordinator', 'Specialist', 'Engineer',
    'Executive', 'Manager', 'Analyst', 'Assistant', 'Coordinator',
    'Specialist', 'Engineer', 'Executive', 'Manager', 'Analyst'],
'Salary': [50000, 40000, 60000, 55000, 45000,
        42000, 43000, 52000, 61000, 53000,
        46000, 51000, 43000, 62000, 54000,
        47000, 44000, 53000, 63000, 55000,
        48000, 45000, 54000, 64000, 56000],
'Join_Date': ['2021-01-15', '2021-02-10', '2021-03-05', '2021-04-12', '2021-05-1
        '2021-06-20', '2021-07-25', '2021-08-30', '2021-09-10', '2021-10-15',
        '2021-11-01', '2021-12-05', '2022-01-10', '2022-02-14', '2022-03-18',
        '2022-04-20', '2022-05-25', '2022-06-30', '2022-07-10', '2022-08-15',
        '2022-09-20', '2022-10-25', '2022-11-30', '2022-12-05', '2023-01-10'],
 'Email': ['amit@company.com', 'priya@company.com', 'rahul@company.com', 'sneha@
        'anita@company.com', 'ravi@company.com', 'pooja@company.com', 'vikram@co
        'arjun@company.com', 'kavya@company.com', 'suresh@company.com', 'meena@c
        'divya@company.com', 'manoj@company.com', 'ritu@company.com', 'rakesh@co
        'tarun@company.com', 'shreya@company.com', 'deepak@company.com', 'payal@
'Location': ['Mumbai', 'Delhi', 'Chennai', 'Bangalore', 'Hyderabad',
        'Pune', 'Kolkata', 'Mumbai', 'Delhi', 'Chennai',
         'Bangalore', 'Hyderabad', 'Pune', 'Kolkata', 'Mumbai',
        'Delhi', 'Chennai', 'Bangalore', 'Hyderabad', 'Pune',
        'Kolkata', 'Mumbai', 'Delhi', 'Chennai', 'Bangalore']
}
df = pd.DataFrame(data)
```

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

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

In [ ]:

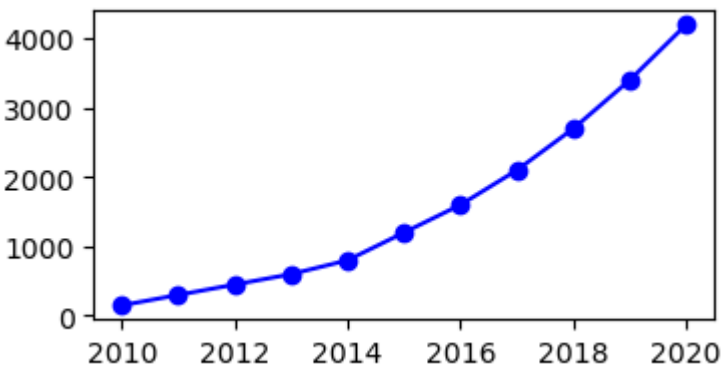
1. The following table shows the number of Data Science job postings recorded over the years 2010 to 2020. Year Job Postings 2010 150 2011 300 2012 450 2013 600 2014 800 2015 1200 2016 1600 2017 2100 2018 2700 2019 3400 2020 4200 Using Python (Pandas and Matplotlib): Create a DataFrame for the given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

In [4]:
```
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
```

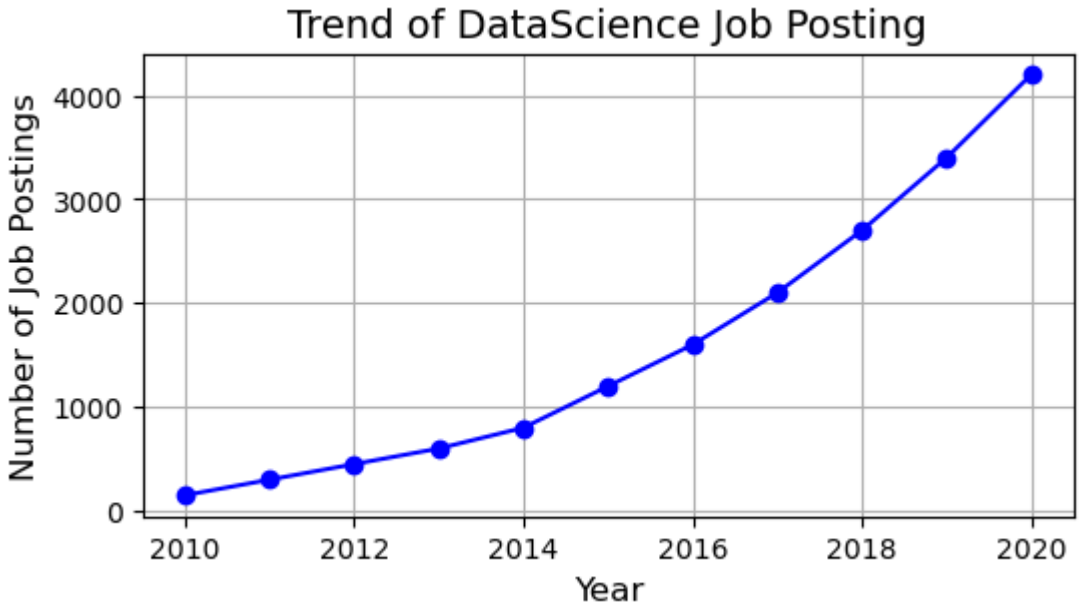
In [ ]:
```
2. Plot a line graph showing the trend of Data Science job postings over the yea
on data points.
```

In [3]:
```
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.figure(figsize=(4,2))
plt.plot(df['Year'],df['Job Postings'],marker='o',linestyle='-',color='b')
plt.show()
```



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

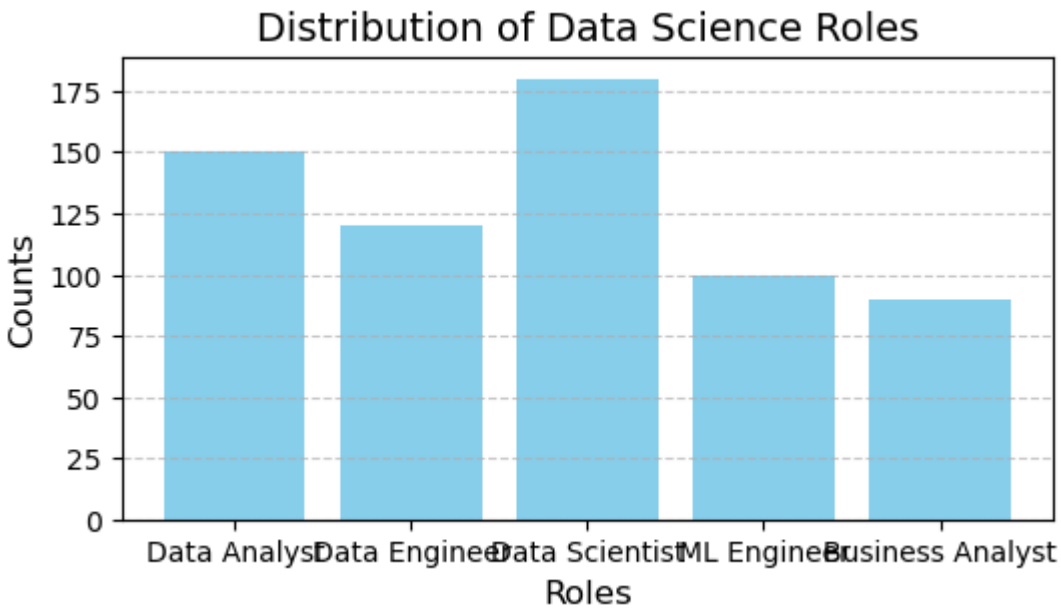
In [8]:
```
import pandas as pd
import matplotlib.pyplot as plt
data={
    'Year':[2010, 2011, 2012, 2013, 2014, 2015, 2016, 2017, 2018, 2019, 2020],
'Job Postings': [150, 300, 450, 600, 800, 1200, 1600, 2100, 2700, 3400, 4200]
}
df=pd.DataFrame(data)
plt.figure(figsize=(6,3))
plt.plot(df['Year'],df['Job 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 [8]:
```
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.figure(figsize=(6,3))
plt.bar(df['roles'],df['counts'],color='skyblue')
plt.title('Distribution of Data Science Roles',fontsize=14)
plt.xlabel('Roles',fontsize=12)
plt.ylabel('Counts',fontsize=12)
plt.grid(axis='y', linestyle='--', alpha=0.7)
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 [14]:
```
import pandas as pd
data ={
    'name':['kaushika','kaviya','manisha'],
    'age':[19,17,18],
    'city':['kpm','chennai','chennai']
}
df=pd.DataFrame(data)
print("___structured_data___")
print(df,"\n")
unstructured_data="we three are studying in rajalakshmi college"
print("___unstructured_data___")
print(unstructured_data,"\n")
semi_structured_data = {
    "student": {
        "id": 101,
        "name": "John Doe",
        "courses": ["Math", "Science", "History"],
        "address": {
            "city": "New York",
            "zipcode": "10001"
        }
    }
}
print("___semi_structured_data___")
print(semi_structured_data)
```

```
___structured_data___
       name  age     city
0  kaushika   19      kpm
1    kaviya   17  chennai
2   manisha   18  chennai

___unstructured_data___
we three are studying in rajalakshmi college

___semi_structured_data___
{'student': {'id': 101, 'name': 'John Doe', 'courses': ['Math', 'Science', 'Histo
ry'], 'address': {'city': 'New York', 'zipcode': '10001'}}}
```

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 [16]:
```python
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)
```

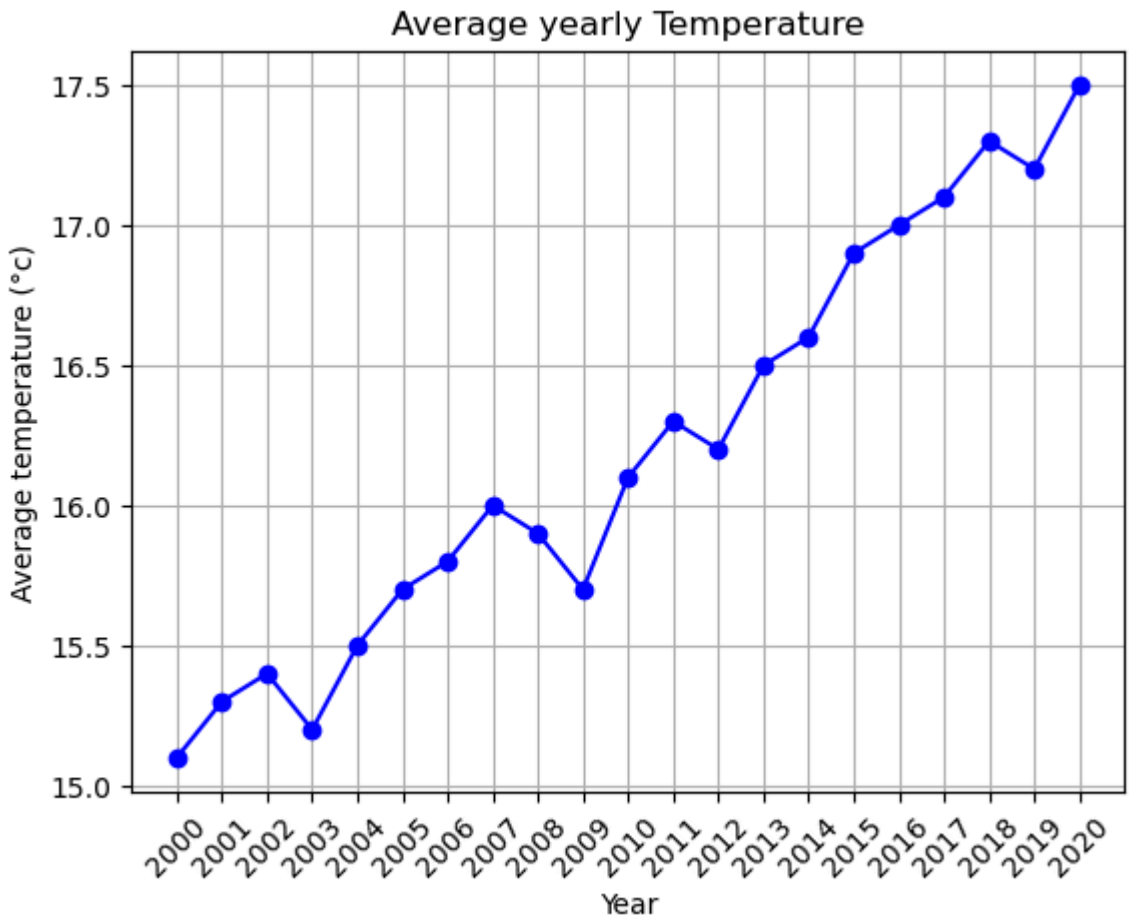
```
Original Data: b'Rajalakshmi Engineering College.'
Encrypted Data: b'gAAAAABotqMVLSHIvEqKOCsKwcTrBobp3mODH6UcPgNqfIfN7gbF0a05Lwfum5Z
UW3368Lmi4Be5E0c1NcA-fePsP-FKX6qbz64pFdx6gHnOqrwERwUaDW6FNYgcf8M1v8qUM0XiqOEp'
Decrypted Data: b'Rajalakshmi Engineering College.'
```

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.

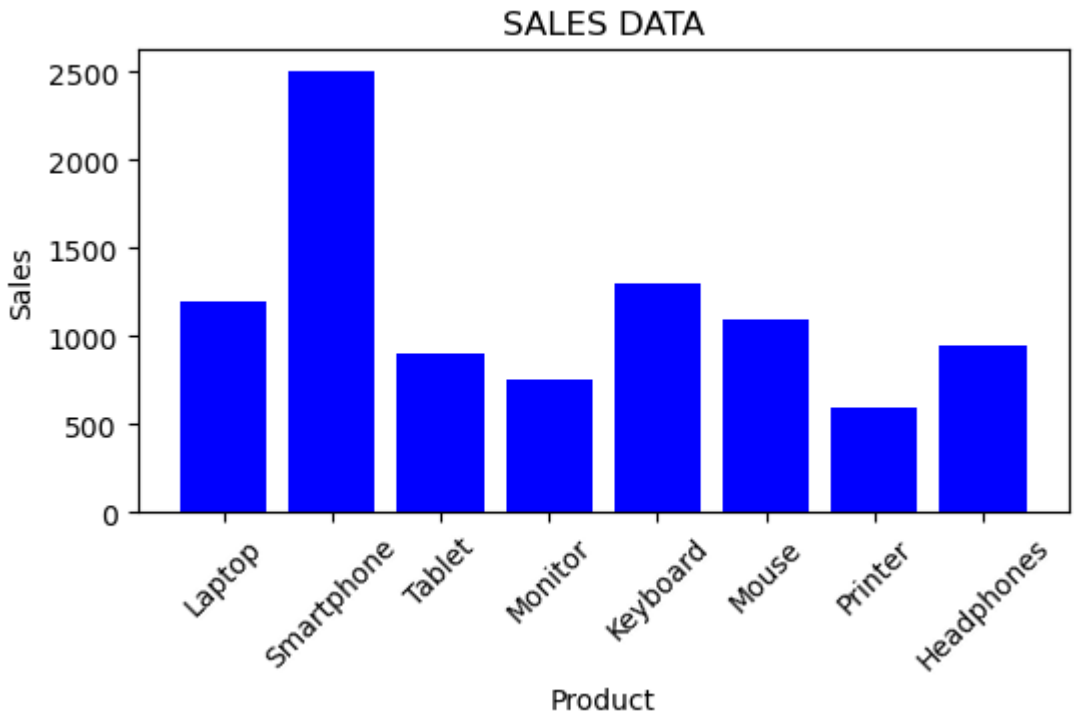
In [6]:
```
import pandas as pd
import matplotlib.pyplot as plt
data = {
    'Year': list(range(2000,2021)),
    'AvgTemperature': [
        15.1, 15.3, 15.4, 15.2, 15.5,
        15.7, 15.8, 16.0, 15.9, 15.7,
        16.1, 16.3, 16.2, 16.5, 16.6,
        16.9, 17.0, 17.1, 17.3, 17.2,
        17.5
    ]
}
df=pd.DataFrame(data)
df.to_csv('weather.csv',index=False)
print("weather.csv file created using pandas.")
df=pd.read_csv('weather.csv')
plt.plot(df['Year'],df['AvgTemperature'],marker='o',linestyle='-',color='blue')
plt.xlabel('Year')
plt.ylabel('Average temperature (°c)')
plt.title('Average yearly Temperature')
plt.xticks(df['Year'], rotation=45)
plt.grid(True)
plt.show()
```

```
weather.csv file created using pandas.
```



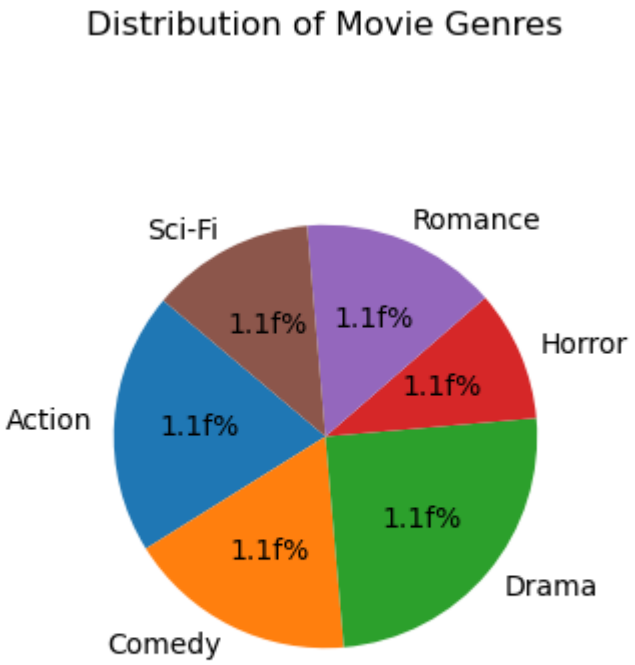
Q2. Bar Chart from CSV (Sales Data) Given a CSV file sales.csv with columns: Product, Sales. Write a Python program to plot a bar chart showing the total sales of each product. • Add axis labels and a title. • Rotate product names if necessary for better visibility.

In [3]:
```
import pandas as pd
import matplotlib.pyplot as plt
sales_data = {
    'Product': ['Laptop', 'Smartphone', 'Tablet', 'Monitor', 'Keyboard', 'Mouse'
    'Sales': [1200, 2500, 900, 750, 1300, 1100, 600, 950]
}
df=pd.DataFrame(sales_data)
df.to_csv('sales.csv',index=False)
df=pd.read_csv('sales.csv')
plt.figure(figsize=(6,3))
plt.bar(df['Product'],df['Sales'],color='blue')
plt.xlabel('Product')
plt.ylabel('Sales')
plt.title('SALES DATA')
plt.xticks(df['Product'],rotation=45)
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".

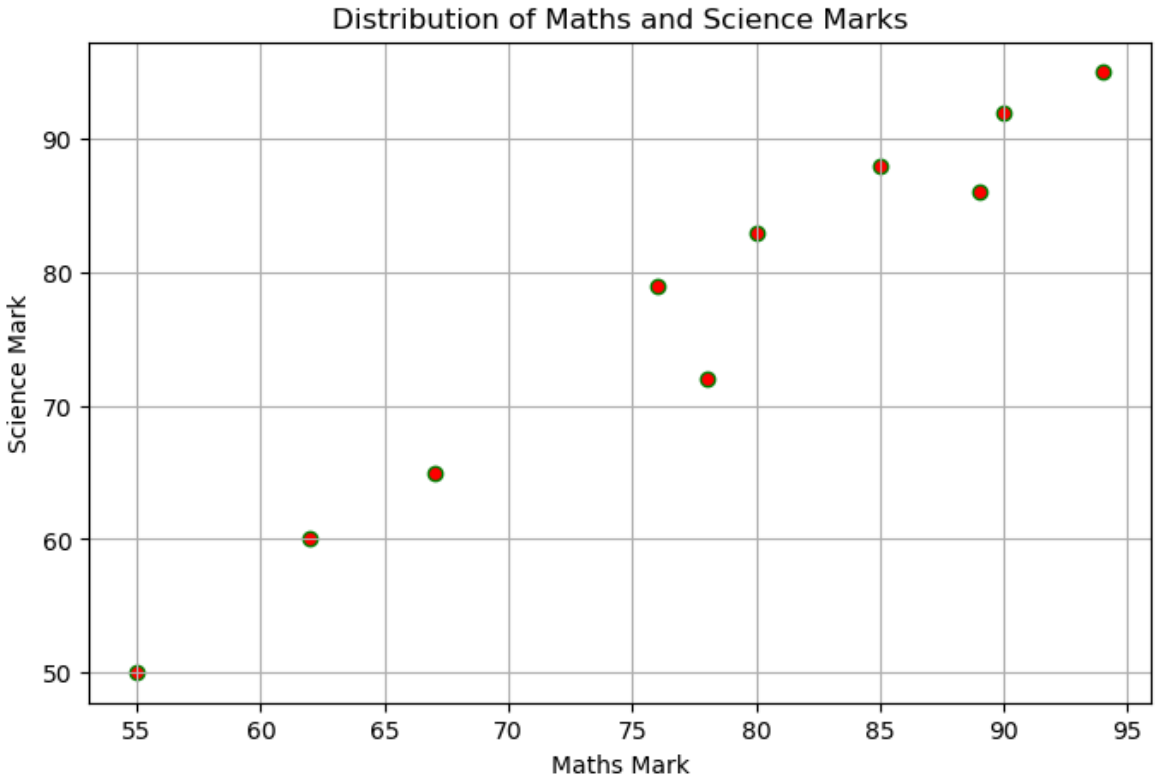
In [6]:
```
import pandas as pd
import matplotlib.pyplot as plt
movie_data = {
    'Genre': ['Action', 'Comedy', 'Drama', 'Horror', 'Romance', 'Sci-Fi'],
    'Count': [40, 35, 50, 20, 30, 25]
}
df=pd.DataFrame(movie_data)
df.to_csv('movie.csv',index=False)
df=pd.read_csv('movie.csv')
genre=df['Genre']
count=df['Count']
plt.figure(figsize=(3,5))
plt.pie(count,labels=genre,autopct='1.1f%%',startangle=140)
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.

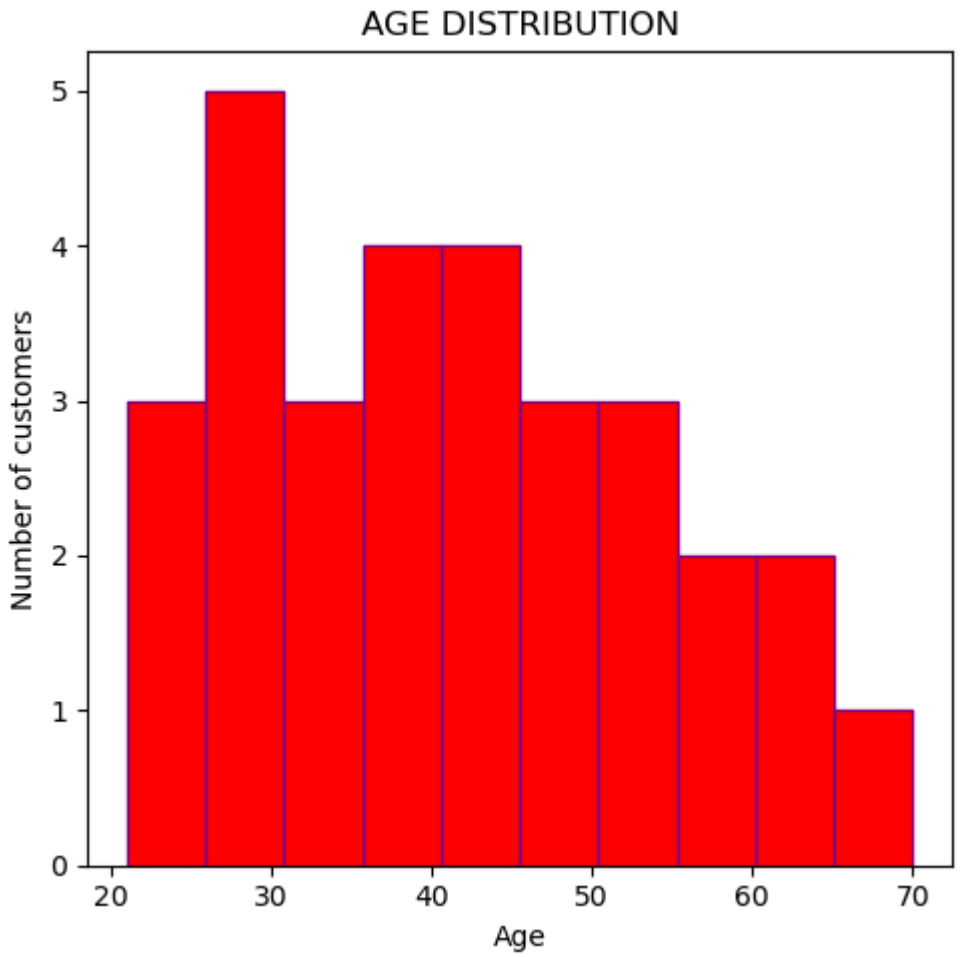
In [12]:
```
import pandas as pd
import matplotlib.pyplot as plt
student_data = {
    'MathMarks': [78, 85, 62, 90, 55, 89, 76, 94, 67, 80],
    'ScienceMarks': [72, 88, 60, 92, 50, 86, 79, 95, 65, 83]
}
df=pd.DataFrame(student_data)
df.to_csv('student.csv',index=False)
print("Student.csv file created")
df=pd.read_csv('student.csv')
plt.figure(figsize=(8,5))
plt.scatter(df['MathMarks'],df['ScienceMarks'],color='red',marker='o',edgecolors
plt.xlabel('Maths Mark')
plt.ylabel('Science Mark')
plt.title('Distribution of Maths and Science Marks')
plt.grid(True)
plt.show()
```

```
Student.csv file created
```



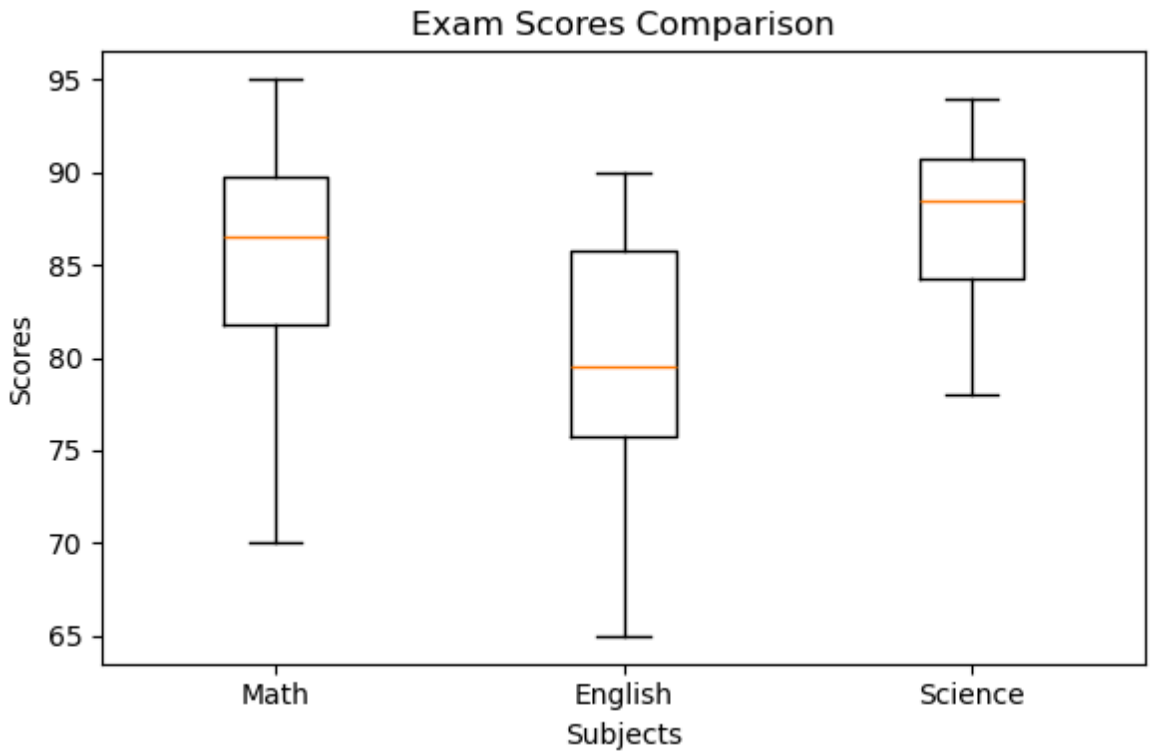
Q5. Histogram (Customer Ages) Given a CSV file customers.csv with a column: Age. Write a Python program to plot a histogram showing the age distribution of customers. • Use 10 bins. • Add axis labels and a title.

In [18]:
```
import pandas as pd
import matplotlib.pyplot as plt
customer_data = {
    'Age': [22, 25, 29, 21, 34, 45, 52, 48, 37, 26, 30, 31, 33, 42, 41,
            55, 60, 28, 27, 36, 38, 40, 43, 46, 50, 53, 58, 61, 65, 70]
}
df=pd.DataFrame(customer_data)
df.to_csv('customer.csv',index=False)
df=pd.read_csv('customer.csv')
plt.figure(figsize=(5,5))
plt.hist(df['Age'],bins=10,color='red',edgecolor='purple')
plt.xlabel('Age')
plt.ylabel('Number of customers')
plt.title('AGE DISTRIBUTION')
plt.tight_layout()#to prevent overlaps
plt.show()
```



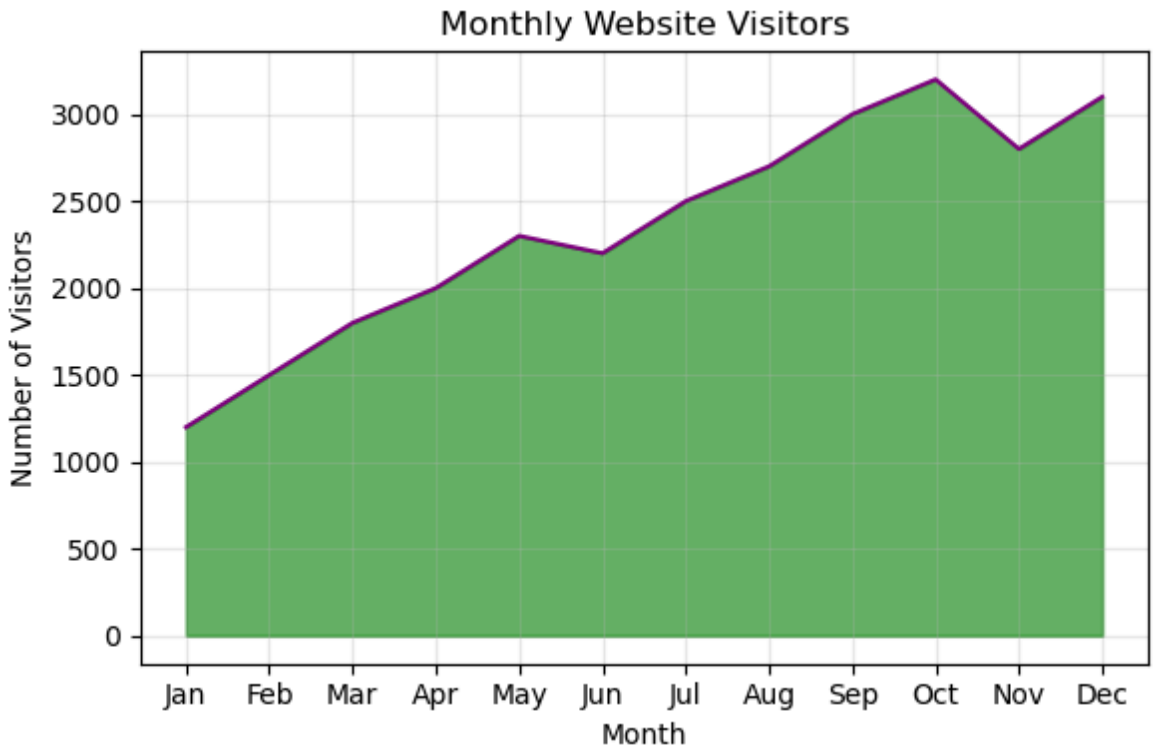
Q6. Boxplot (Exam Scores Comparison) Create a CSV file exam_scores.csv with columns: Math, English, Science. Write a Python program to plot a boxplot comparing score distributions across subjects. • Add axis labels and a title.

In [25]:
```
import pandas as pd
import matplotlib.pyplot as plt
data = {
    'Math':    [85, 88, 90, 76, 95, 89, 70, 84, 92, 81],
    'English': [78, 74, 80, 85, 90, 88, 65, 79, 86, 75],
    'Science': [92, 89, 94, 83, 91, 85, 78, 88, 90, 84]
}
df=pd.DataFrame(data)
df.to_csv('score.csv',index=False)
df=pd.read_csv('score.csv')
plt.figure(figsize=(6,4))
plt.boxplot([df['Math'],df['English'],df['Science']],tick_labels=['Math','Englis
plt.title('Exam Scores Comparison')
plt.xlabel('Subjects')
plt.ylabel('Scores')
plt.tight_layout()
plt.show()
```



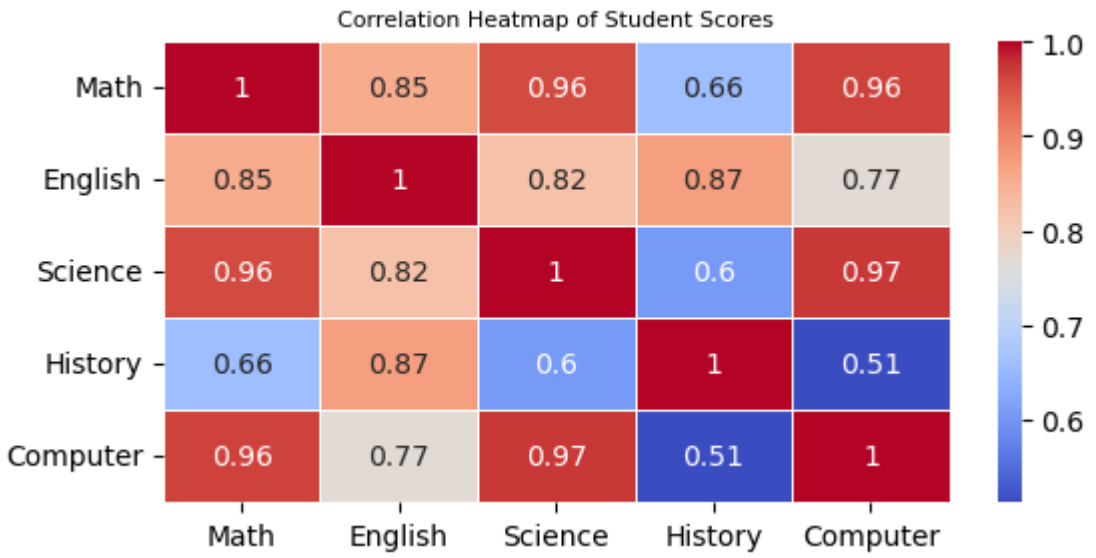
Q7. Area Chart (Website Traffic) Create a CSV file traffic.csv with columns: Month, Visitors. Write a Python program to plot an area chart showing monthly website visitors for a year. • Add axis labels and a title. • Use a different color for the filled area.

In [30]:
```
import pandas as pd
import matplotlib.pyplot as plt
data = {
    'Month': ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep', 'Oc
    'Visitors': [1200, 1500, 1800, 2000, 2300, 2200, 2500, 2700, 3000, 3200, 280
}
df = pd.DataFrame(data)
df.to_csv('traffic.csv', index=False)
df = pd.read_csv('traffic.csv')
plt.figure(figsize=(6, 4))
plt.fill_between(df['Month'], df['Visitors'], color='green', alpha=0.6)
plt.plot(df['Month'], df['Visitors'], color='purple')
plt.title('Monthly Website Visitors')
plt.xlabel('Month')
plt.ylabel('Number of Visitors')
plt.grid(alpha=0.3)# alpha is for visiblity
plt.tight_layout()
plt.show()
```



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

In [5]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
data = {
    'Math': [88, 92, 80, 89, 100, 67, 76, 94, 85, 91],
    'English': [78, 85, 82, 88, 90, 72, 79, 84, 81, 83],
    'Science': [90, 95, 85, 88, 98, 70, 77, 89, 84, 92],
    'History': [70, 75, 78, 80, 85, 68, 72, 76, 79, 74],
    'Computer': [95, 98, 90, 93, 100, 85, 88, 96, 91, 97]
}
df = pd.DataFrame(data)
df.to_csv('Student_scores.csv',index=False)
df = pd.read_csv('student_scores.csv')
corr_mat=df.corr()
plt.figure(figsize=(6,3))
sns.heatmap(corr_mat,annot=True,cmap='coolwarm',linewidths=0.5)
plt.title('Correlation Heatmap of Student Scores',fontsize=8)
plt.tight_layout()
plt.show()
```


Correlation Heatmap of Student Scores
Math
English
Science
History
Computer
1
0.85
0.96
0.66
0.96
0.85
1
0.82
0.87
0.77
0.96
0.82
1
0.6
0.97
0.66
0.87
0.6
1
0.51
0.96
0.77
0.97
0.51
1
1.0
0.9
0.8
0.7
0.6

In [ ]:

In [1]:
```
import matplotlib
import seaborn
print("Matplotlib:", matplotlib.__version__)
print("Seaborn:", seaborn.__version__)
```

```
Matplotlib: 3.8.4
Seaborn: 0.13.2
```

In [4]:
```
import matplotlib
import seaborn
print("Matplotlib:", matplotlib.__version__, "→", matplotlib.__file__)
print("Seaborn:", seaborn.__version__, "→", seaborn.__file__)
```

```
Matplotlib: 3.8.4 → C:\Users\Lenovo\AppData\Roaming\Python\Python39\site-packages\matplo
tlib\__init__.py
Seaborn: 0.13.2 → C:\Users\Lenovo\AppData\Roaming\Python\Python39\site-packages\seaborn
\__init__.py
```

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

In [ ]:
```
1) A retail company has provided a CSV file (sales_data.csv) containing sales transacti
with the following columns: Date, Product, Quantity, Sales, and Region. As a Data Analy
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
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 heatm
Discuss the insights obtained from each step.
```

In [2]:
```
sample_data = """Date,Product,Quantity,Sales,Region
2025-01-05,Smartphone,10,5000,North
2025-01-06,Laptop,5,7500,South
2025-01-07,Tablet,8,3200,East
2025-01-08,Smartwatch,15,2250,West
2025-01-09,Headphones,20,2000,North
2025-01-10,Laptop,3,4500,East
2025-01-11,Smartphone,7,3500,South
2025-01-12,Tablet,10,4000,West
2025-01-13,Smartwatch,12,1800,North
```

```
2025-01-14,Headphones,25,2500,South
2025-01-15,Laptop,6,,West
2025-01-16,,10,5000,East
2025-01-17,Smartphone,,4000,North
2025-01-18,Tablet,9,3600,
"""

with open("sales_data.csv", "w") as f:
    f.write(sample_data)
```

In [3]:

```
import pandas as pd

df = pd.read_csv('sales_data.csv')
print(df.head())
```

```
         Date     Product  Quantity   Sales Region
0  2025-01-05  Smartphone      10.0  5000.0  North
1  2025-01-06      Laptop       5.0  7500.0  South
2  2025-01-07      Tablet       8.0  3200.0   East
3  2025-01-08  Smartwatch      15.0  2250.0   West
4  2025-01-09  Headphones      20.0  2000.0  North
```

In [4]:

```
import pandas as pd

df = pd.read_csv('sales_data.csv')

print(df.isnull().sum())

mean_sales = df['Sales'].mean()
df['Sales'] = df['Sales'].fillna(mean_sales)

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

print(df.isnull().sum())
```

```
Date        0
Product     1
Quantity    1
Sales       1
Region      1
dtype: int64
Date        0
Product     0
Quantity    0
Sales       0
Region      0
dtype: int64
```

In [5]:

```
print(df.describe())
```

```
        Quantity        Sales
count  11.000000    11.000000
mean   11.000000  3637.062937
std     6.678323  1647.794180
min     3.000000  1800.000000
25%     6.500000  2375.000000
50%    10.000000  3500.000000
```

```
75%    13.500000  4250.000000
max    25.000000  7500.000000
```

In [6]:

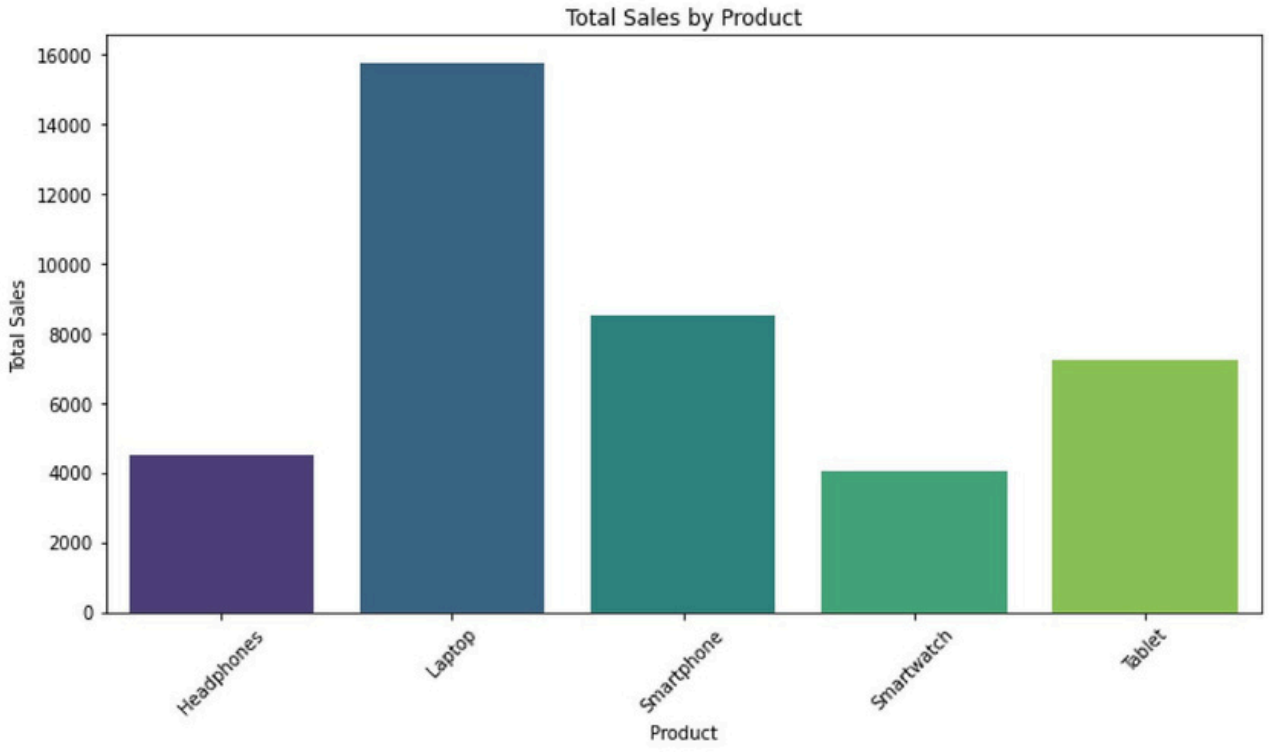
```
product_group = df.groupby('Product').agg({'Sales':'sum', 'Quantity':'sum'}).reset_inde
print(product_group)
```

```
      Product         Sales  Quantity
0  Headphones   4500.000000      45.0
1      Laptop  15757.692308      14.0
2  Smartphone   8500.000000      17.0
3  Smartwatch   4050.000000      27.0
4      Tablet   7200.000000      18.0
```

In [7]:

```
plt.figure(figsize=(10,6))
sns.barplot(data=product_group, x='Product', y='Sales', palette='viridis')
plt.title('Total Sales by Product')
plt.ylabel('Total Sales')
plt.xlabel('Product')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

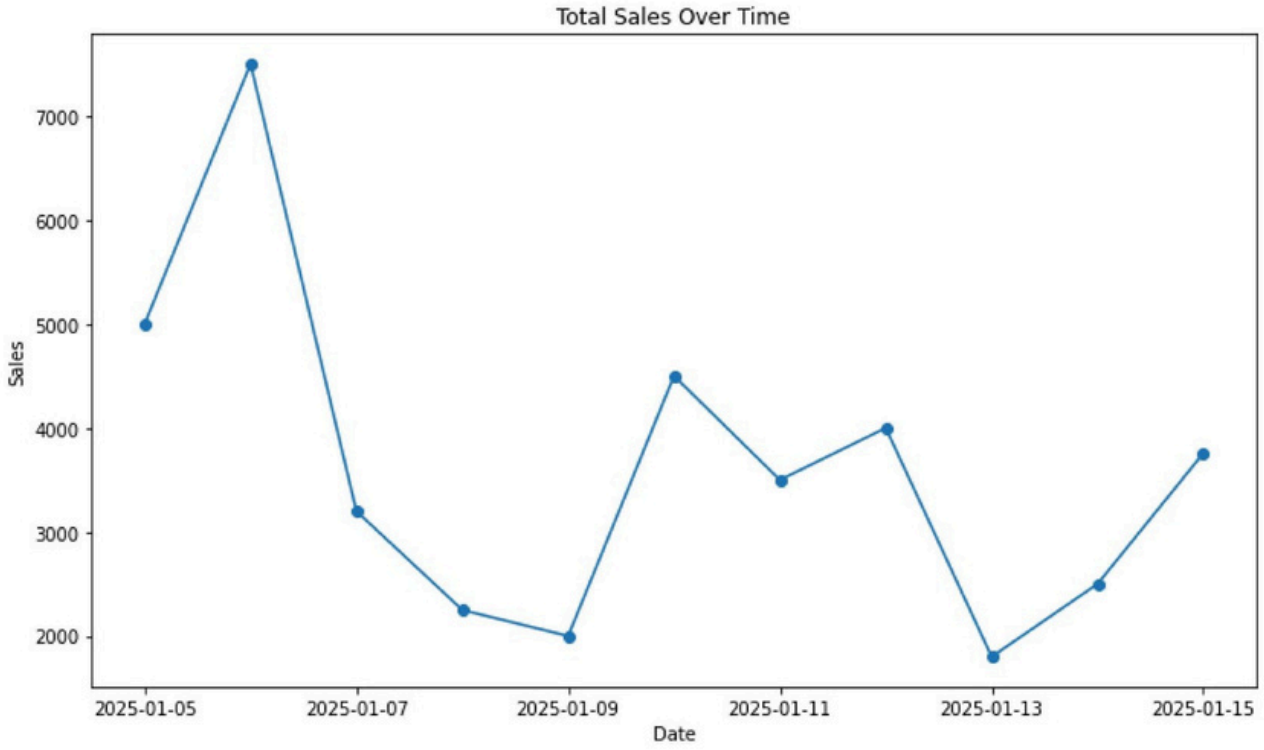
```
C:\Users\Lenovo\AppData\Local\Temp/ipykernel_14500/3120654797.py:2: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0.
Assign the `x` variable to `hue` and set `legend=False` for the same effect.

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



In [8]:

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

plt.figure(figsize=(10,6))
plt.plot(time_series['Date'], time_series['Sales'], marker='o')
```

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



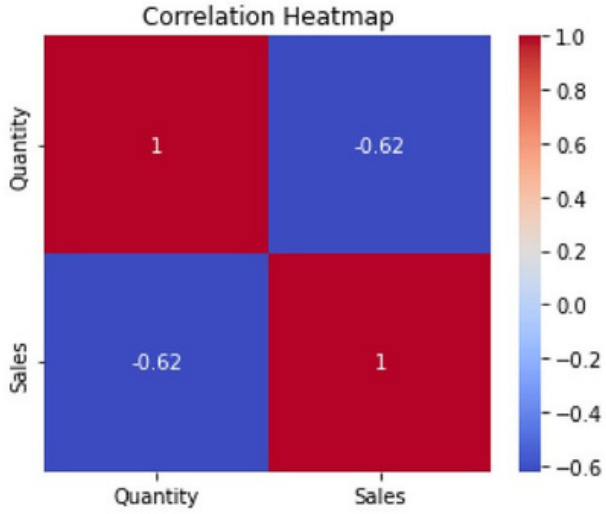
In [9]:

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

Product	Headphones	Laptop	Smartphone	Smartwatch	Tablet
Region					
East	0.0	4500.000000	0.0	0.0	3200.0
North	2000.0	0.000000	5000.0	1800.0	0.0
South	2500.0	7500.000000	3500.0	0.0	0.0
West	0.0	3757.692308	0.0	2250.0	4000.0

In [10]:

```
corr = df[['Quantity', 'Sales']].corr()

plt.figure(figsize=(5,4))
sns.heatmap(corr, annot=True, cmap='coolwarm')
plt.title('Correlation Heatmap')
plt.show()
```

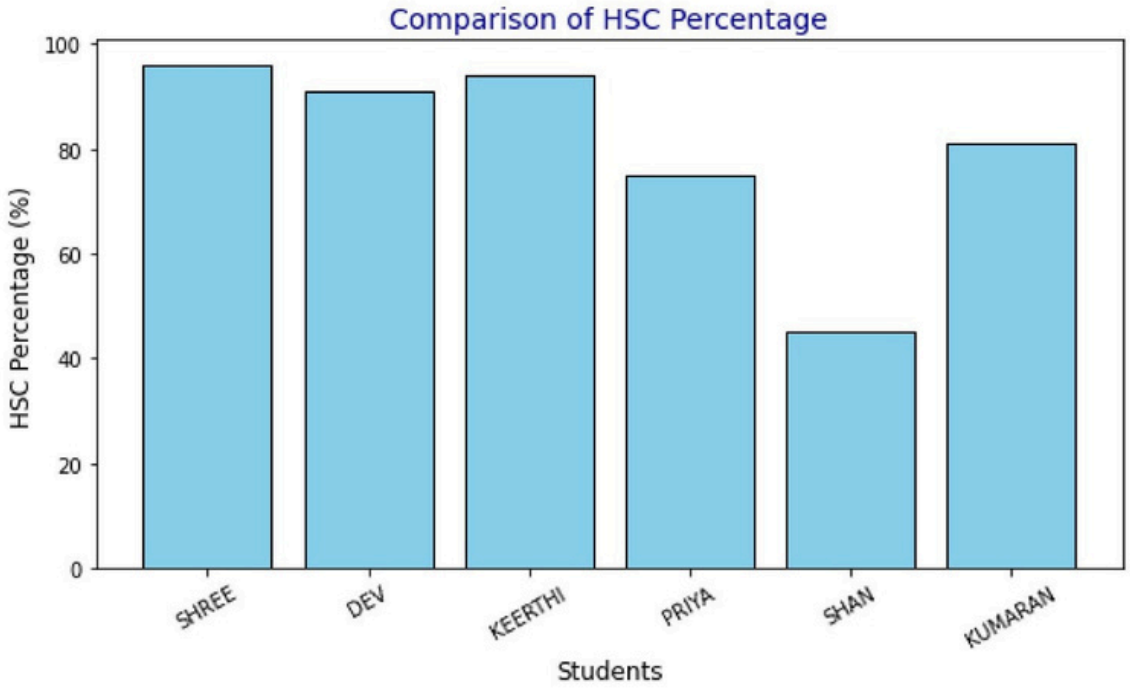


In [ ]:

```
2) A dataset contains the names of six students (SHREE, DEV, KEERTHI, PRIYA, SHAN, KUMA
along with their Higher Secondary Certificate (HSC) exam percentages: 96, 91, 94, 75, 4
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().
Also, interpret the chart to identify the student with the highest and lowest percentag
```

In [11]:

```
import numpy as np
import matplotlib.pyplot as plt

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

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

plt.xlabel('Students', fontsize=12)
plt.ylabel('HSC Percentage (%)', fontsize=12)
plt.title('Comparison of HSC Percentage', fontsize=14, color='darkblue')

plt.xticks(rotation=30)

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

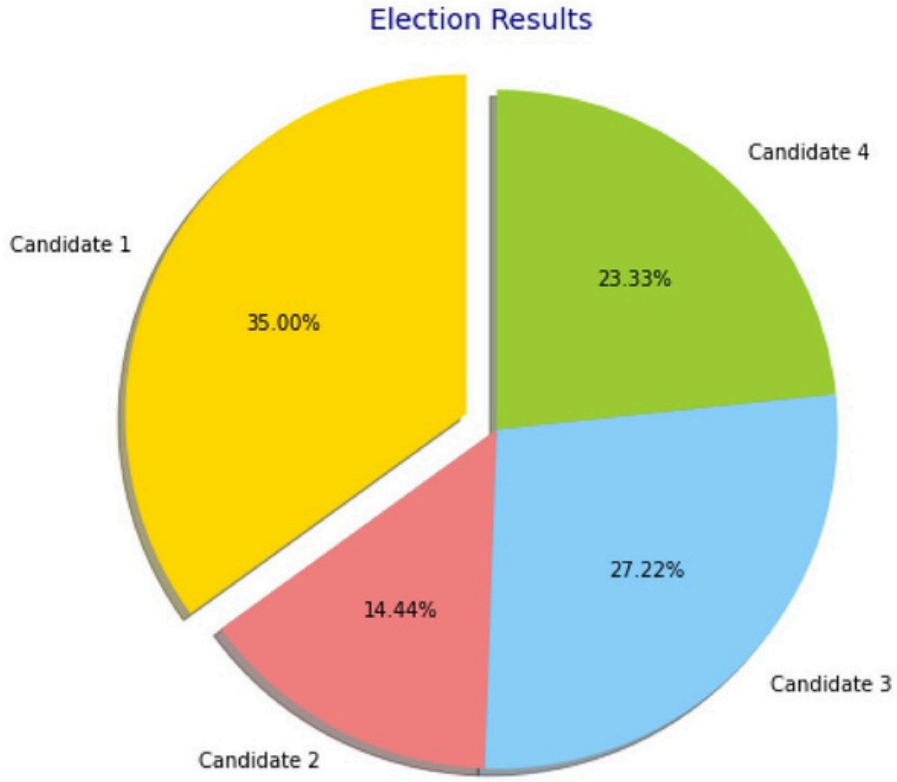


In [ ]:

```
3) The following table shows the number of votes received by four candidates in a colle
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 [12]:

```
import matplotlib.pyplot as plt

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

explode = [0.1, 0, 0, 0]

colors = ['gold', 'lightcoral', 'lightskyblue', 'yellowgreen']

plt.figure(figsize=(7,7))
plt.pie(votes, labels=candidates, autopct='%1.2f%%', startangle=90,
        explode=explode, colors=colors, shadow=True)

plt.title('Election Results', fontsize=14, color='darkblue')

plt.axis('equal')

plt.show()
```

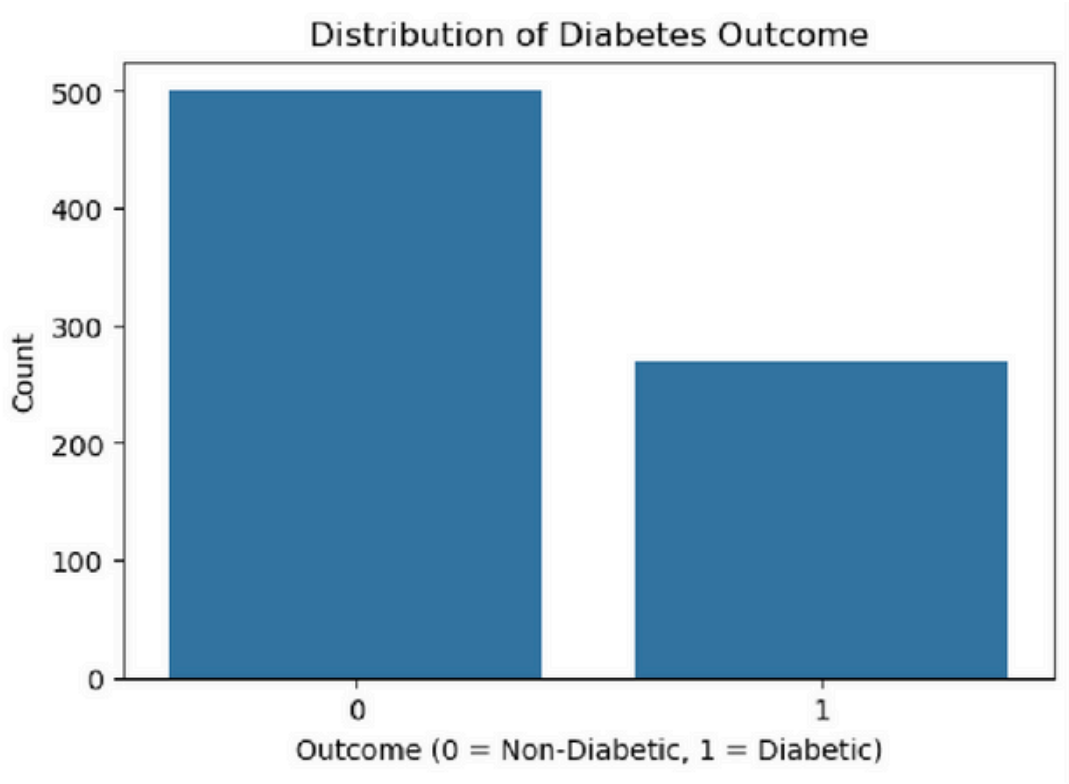

Election Results
Candidate 4
23.33%
Candidate 1
35.00%
27.22%
14.44%
Candidate 3
Candidate 2

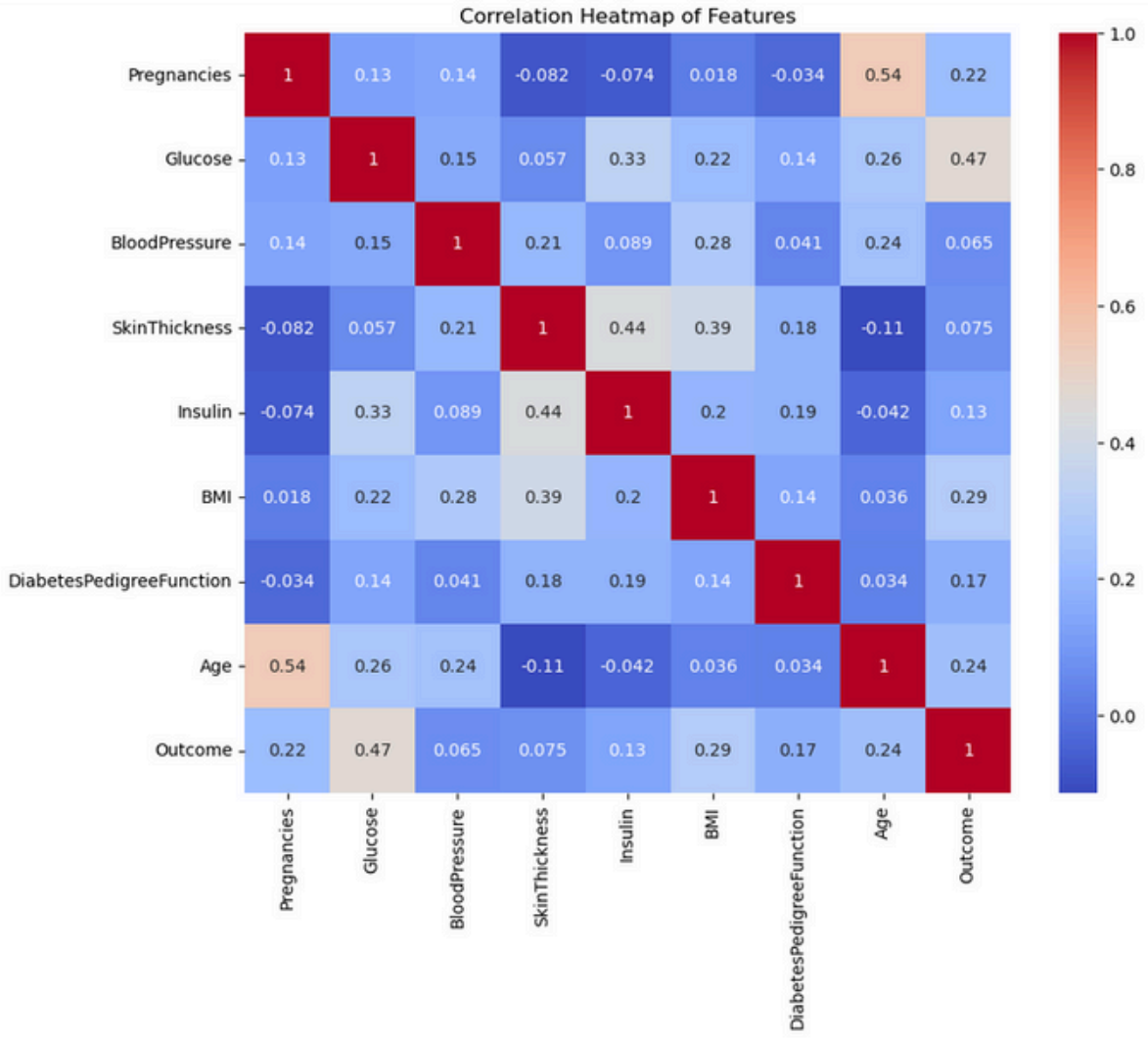
In [ ]:

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    GlucoseB7l6o8SkinThickness uInlnt64
 1   DiabetesPedigreeFunctionsul7i8n7 6 m8o-nuBlMlIt64
 2   onu-lnlu  ll  768  non-null  768  non-null  int64
 3   768 non-null                                int64
 4                                               int64
 5                                               float64
 6                                               float64
 7   Age                       768   non-null   int64
 8   Outcome                   768 non-null    int64
dtypes: float64(2), int64(7)
memory usage: 54.1 KB None

count mPeraeng nasntcdi emsi n 25G%l u5c0o%se BloodPressure SkinThicknessInsulin \
75% max    768.000000 768.000000     768.000000     768.000000 768.000000
             3.845052 120.894531      69.105469     20.536458    79.799479
         3.3695781 0.907020601080      19.355807      15.952218 115.244002
         1.000000      03.000000       0.000000       0.000000    0.000000
         117.000000   99.60.00000 0    62.000000       0.000000    0.000000
         140.250000    17.000000      72.000000     23.000000    30.500000
         199.000000                   80.000000      32.000000 127.250000
                                    122.000000      99.000000 846.000000

              BMI DiabetesPedigreeFunction          Age        Outcome
count 768.000000                         768.000000 768.000000 768.000000
mean std 3m1i.n9 925%7 850% 75%    0.471876   33.240885       0.348958
max  Pregn7a.n8c8i4e1s6 0Glucose   0.331329   11.760232       0.476951
BloodPressu0r.e000000              0.078000   21.000000       0.000000
SkinThickn2e7s.s3 00I0n0s0ulin BMI 0.243750   24.000000       0.000000
DiabetesPe3d2i.g0r0e0e0F0u0nction  0.372500   29.000000       0.000000
Age Outcom3e6 .d6t0y0p0e0:0 int64  0.626250   41.000000       1.000000
        67.100000                  2.420000   81.000000     1.000000
                           0
                           0
                           0
                           0
                           0
                           0
                           0
                           0
                           0
```



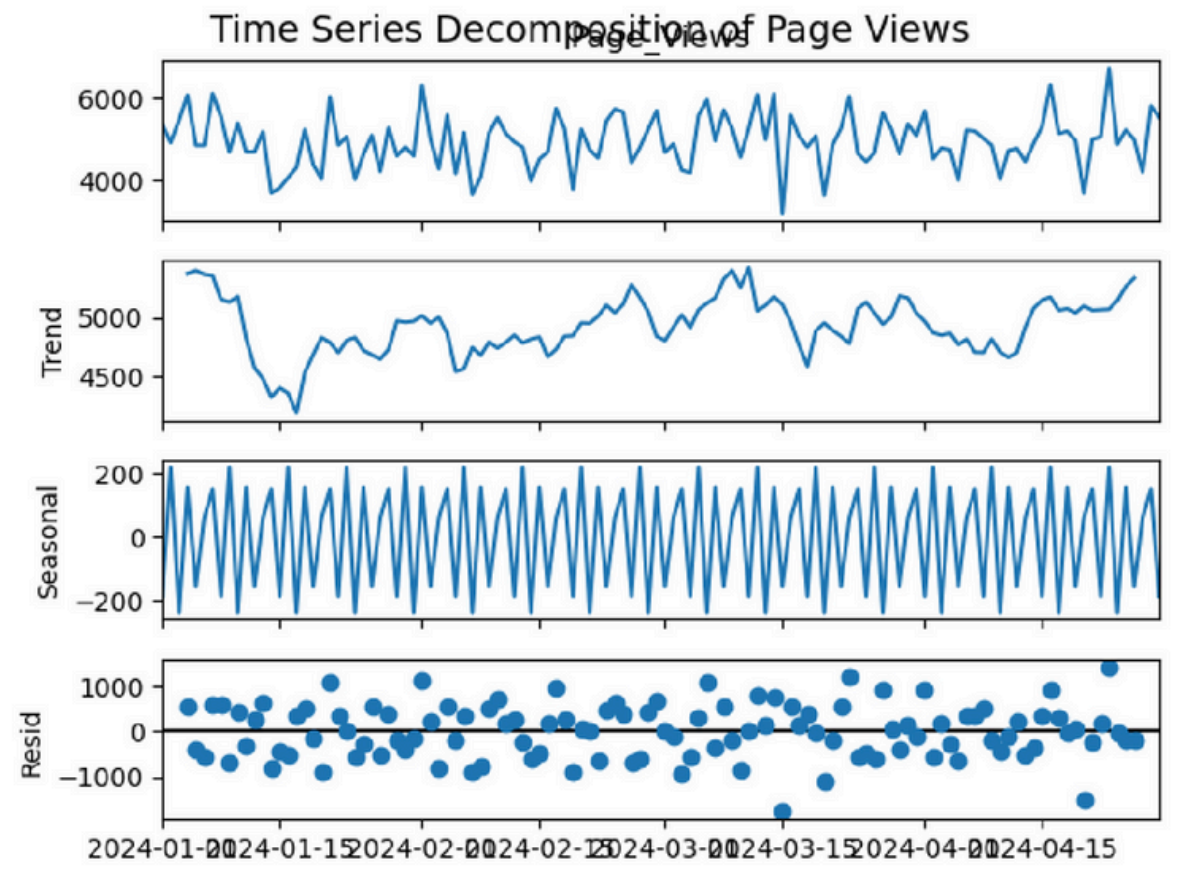


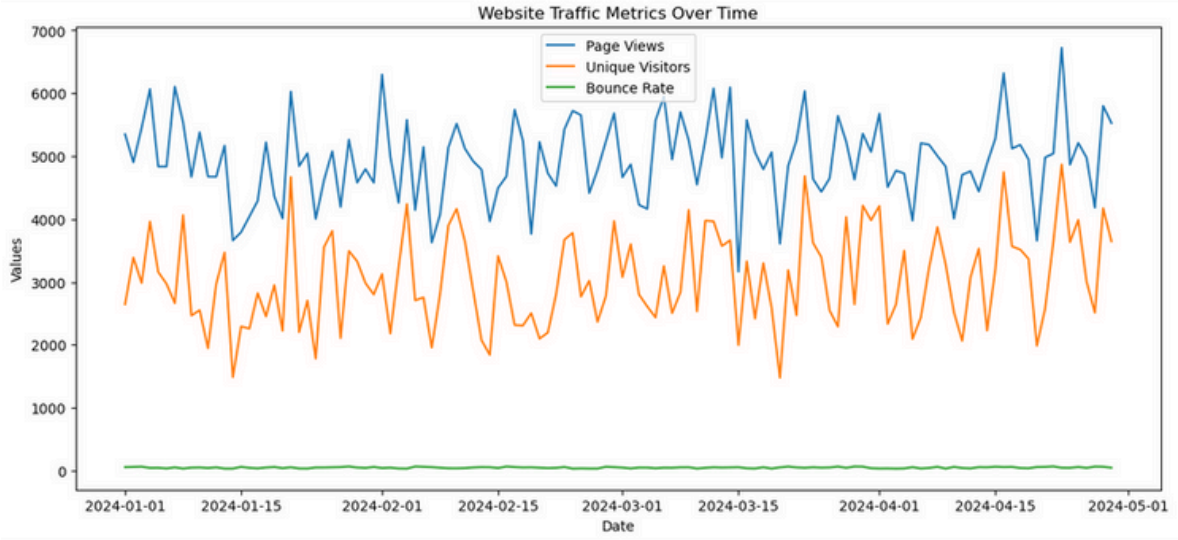
2.Write a Python program to perform Time Series Analysis on a website traffic dataset. The program should load and clean the dataset, decompose the time series to identify

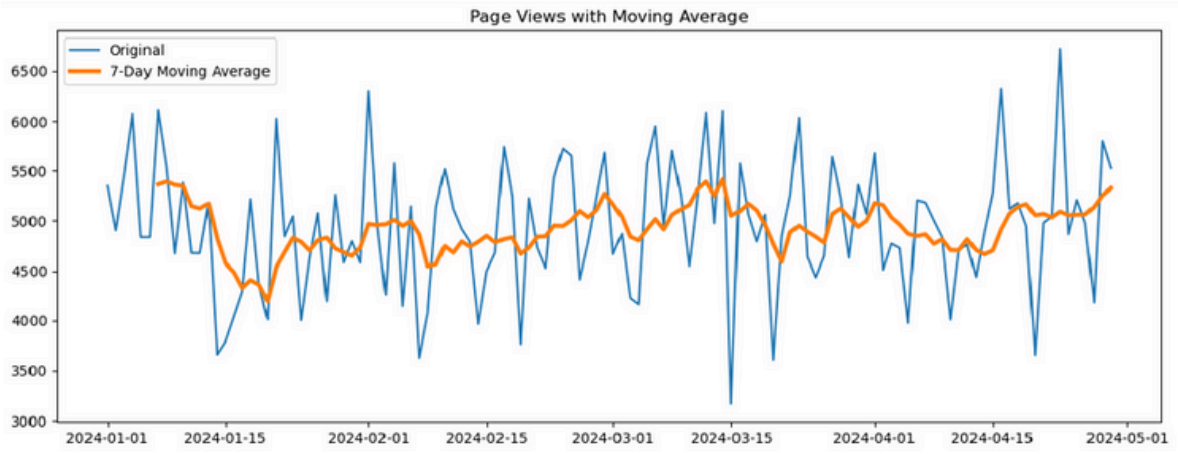
trends and seasonality, visualize key metrics using moving averages and seasonal plots, and detect anomalies using statistical methods.

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>
```





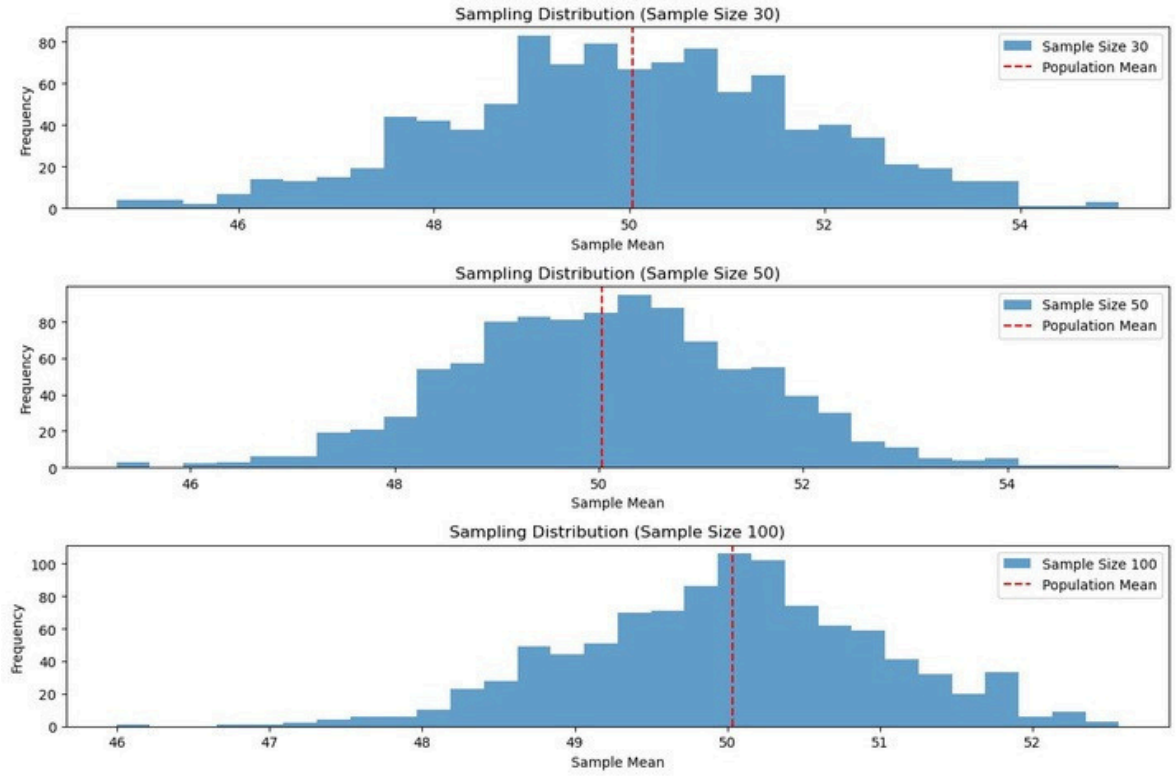


3. Random Sampling and Sampling Distribution To explore random sampling from a population and understand the concept of sampling distribution using Python in

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,  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.
```

jupyter EXP-10 Last Checkpoint: 5 minutes ago

File Edit View Run Kernel Settings Help

Code JupyterLab Python 3 (ipykernel)

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

[5]:
```
import numpy as np
array=np.random.randint(1,100,9)
array
```

[5]:
```
array([63, 44, 70, 75, 89, 59, 75, 19, 41], dtype=int32)
```

[6]:
```
np.sqrt(array)
```

[6]:
```
array([7.93725393, 6.63324958, 8.36660027, 8.66025404, 9.43398113,
       7.68114575, 8.66025404, 4.35889894, 6.40312424])
```

[3]:
```
array.ndim
```

[3]:
```
1
```

[4]:
```
import numpy as np
array=np.random.randint(1,100,9)
new_array=array.reshape(3,3)
new_array
```

[4]:
```
array([[20, 18, 49],
[15, 65, 62],
[46, 69, 59]], dtype=int32)
```

[7]:
```
new_array.ravel()
```

[7]:
```
array([20, 18, 49, 15, 65, 62, 46, 69, 59], dtype=int32)
```

[8]:
```
newm=new_array.reshape(3,3)
newm
```

Experiment to understand pandas library use cases in Data science. Description: Understand data frame use cases using pandas library

In [2]:
```
import pandas as pd
import numpy as np
data = {
    'Name': ['Alice', 'Bob', 'Charlie', 'David', 'Eva'],
    'Age': [25, 30, 35, np.nan, 40],
    'City': ['New York', 'Paris', 'London', 'Berlin', 'Tokyo'],
    'Salary': [50000, 54000, 58000, 62000, np.nan]
}

df = pd.DataFrame(data)
print("Original DataFrame:\n", df)
print("\nBasic Info:")
print(df.info())
print("\nSummary Statistics:")
print(df.describe())
print("\nFirst 3 Rows:")
print(df.head(3))
print("\nLast 3 rows:")
print(df.tail(3))
df['Age'].fillna(df['Age'].mean())
df['Salary'].fillna(df['Salary'].mean())
sorted_df = df.sort_values(by='Salary', ascending=False)
print("\nCleaned and Sorted DataFrame:\n", sorted_df)
print("\nAges:\n", df['Age'])
high_salary = df[df['Salary'] > 55000]
print("\nEmployees with Salary > 55000:\n", high_salary)
df['Experience'] = [2, 4, 6, 8, 10]
print("\nAverage Salary by City:\n", df.groupby('City')['Salary'].mean())
```

```
Original DataFrame:
      Name   Age      City   Salary
0   1   2 Alice 25.0 New York 50000.0
35.0 NaN Bob 30.0      Paris 54000.0
                     London 58000.0
3    David           Berlin 62000.0
4                   Tokyo       NaN

Basic Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5 entries, 0 to 4
Data columns (total 4 columns):
 #       Column Non-Null Count Dtype
    --- ------ -------------- -----
Name 0          5 non-null     object
Age 1           4 non-null     float64
City 2         5 non-null      object
Salary 3 non-null              float64
dtypes: float64(2), object(2)
memory usage: 292.0+ bytes
None

Summary Statistics:
              Age        Salary
count    4.000000       4.000000
mean    32.500000 56000.000000
std     6.454972    5163.977795
min     25.000000 50000.000000
25%     28.750000 53000.000000
50%     32.500000 56000.000000
75%     36.250000 59000.000000
max     40.000000 62000.000000

First 3 Rows:
      Name   Age      City    Salary
0   1   2 Alice 25.0 New York 50000.0
35.0    Bob 30.0      Paris 54000.0
                     London 58000.0

Last 3 rows:
      Name   Age    City    Salary
     2 Charlie 35.0 London 58000.0
3    David      NaN Berlin 62000.0
4      Eva 40.0    Tokyo       NaN

Cleaned and Sorted DataFrame:
       Name   Age      City   Salary
3    David   NaN      Berlin 62000.0
2 Charlie 35.0        London 58000.0
1   Bob 30.0           Paris 54000.0
0        Alice 25.0 New York 50000.0
4      Eva 40.0      Tokyo       NaN

Ages:
0 1 2 25.0
3 4  30.0
     35.0
      NaN
     40.0
Name: Age, dtype: float64
```

```
Employees with Salary > 55000:
       Name   Age    City   Salary
     2 Charlie 35.0 London 58000.0
3    David      NaN Berlin 62000.0

Average Salary by City:
 City
Berlin      62000.0
London      58000.0
New York    50000.0
Paris       54000.0
Tokyo           NaN
Name: Salary, dtype: float64
```

In [ ]:

Experiment to detect outliers in a given data set.

In [39]:
```
import numpy as np
array=np.random.randint(1,100,16)
array
```

Out[39]:
```
array([ 2, 79, 19, 17, 1, 79, 97, 11, 46, 80, 45, 46, 27, 28,   7, 65],
      dtype=int32)
```

In [40]:
```
array.mean()
```

Out[40]:
```
np.float64(40.5625)
```

In [41]:
```
np.percentile(array,25)
```

Out[41]:
```
np.float64(15.5)
```

In [42]:
```
np.percentile(array,50)
```

Out[42]:
```
np.float64(36.5)
```

In [43]:
```
np.percentile(array,75)
```

Out[43]:
```
np.float64(68.5)
```

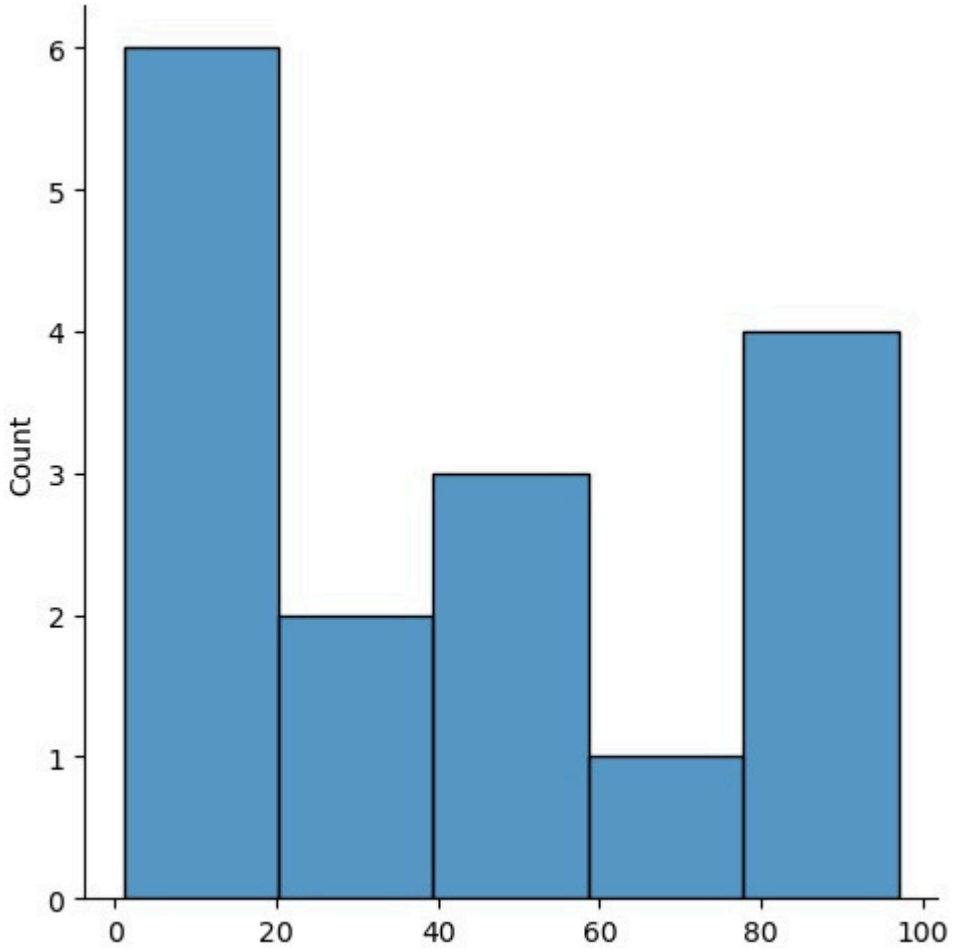
In [44]:
```
np.percentile(array,100)
```

Out[44]:
```
np.float64(97.0)
```

In [45]:
```
def outDetection(array):
    sorted(array)
    Q1,Q3=np.percentile(array,[25,75])
    IQR=Q3-Q1   lr=Q1-(1.5*IQR)   ur=Q3+
    (1.5*IQR) return lr,ur


lr,ur=outDetection(array)
lr,ur
```

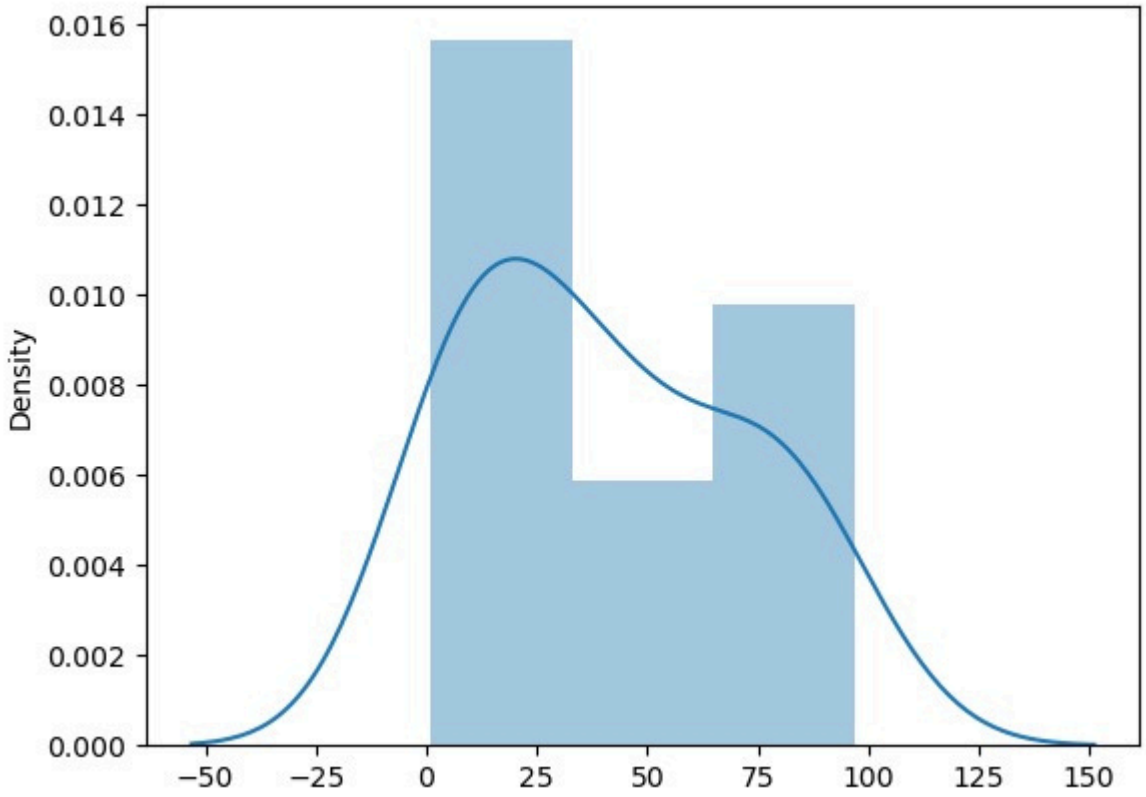
Out[45]:
```
(np.float64(-64.0), np.float64(148.0))
```

In [46]:
```
import seaborn as sns
import matplotlib.pyplot as plt
sns.displot(array)
plt.show()
```



In [47]:
```
sns.distplot(array)
plt.show()
```

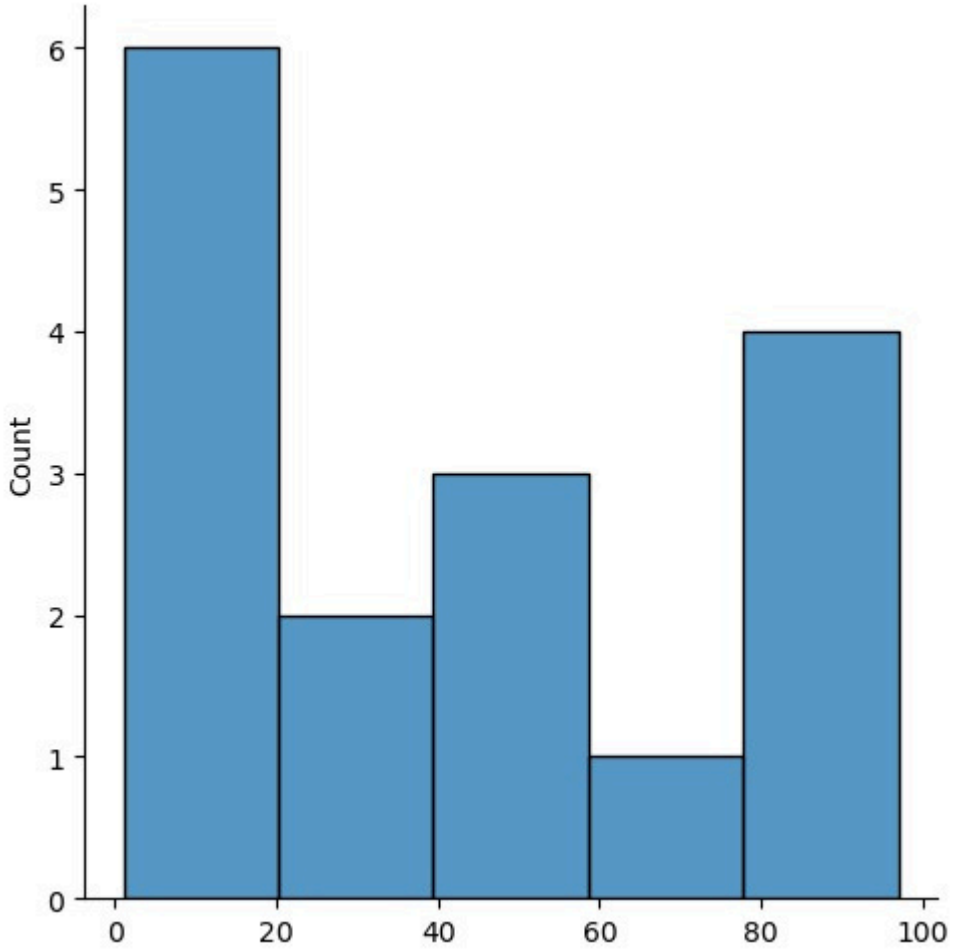
```
`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(array)
C:\Users\Kaviya\AppData\Local\Temp\ipykernel_35088\2106234926.py:1: UserWarning:
```



In [51]:
```
new_array=array[(array>lr)&(array<ur)]
new_array
```

Out[51]:
```
array([ 2, 79, 19, 17, 1, 79, 97, 11, 46, 80, 45, 46, 27, 28,  7, 65],
      dtype=int32)
```

In [52]:
```
sns.displot(new_array)
plt.show()
```



In [53]:
```
lr1,ur1=outDetection(new_array)
lr1,ur1
```

Out[53]: (np.float64(-64.0), np.float64(148.0))

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	Salary
0	John	28.0	New York	50000.0
1	Anna	22.0	Paris	54000.0
2	Peter	30.0	Berlin	58000.0
3	Linda	32.0	New York	56000.0
4	James	40.0	Berlin	62000.0
5	Anna	22.0	Paris	54000.0
6	NaN	35.0	Berlin	58000.0

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

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

EXPERIMENT-15

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

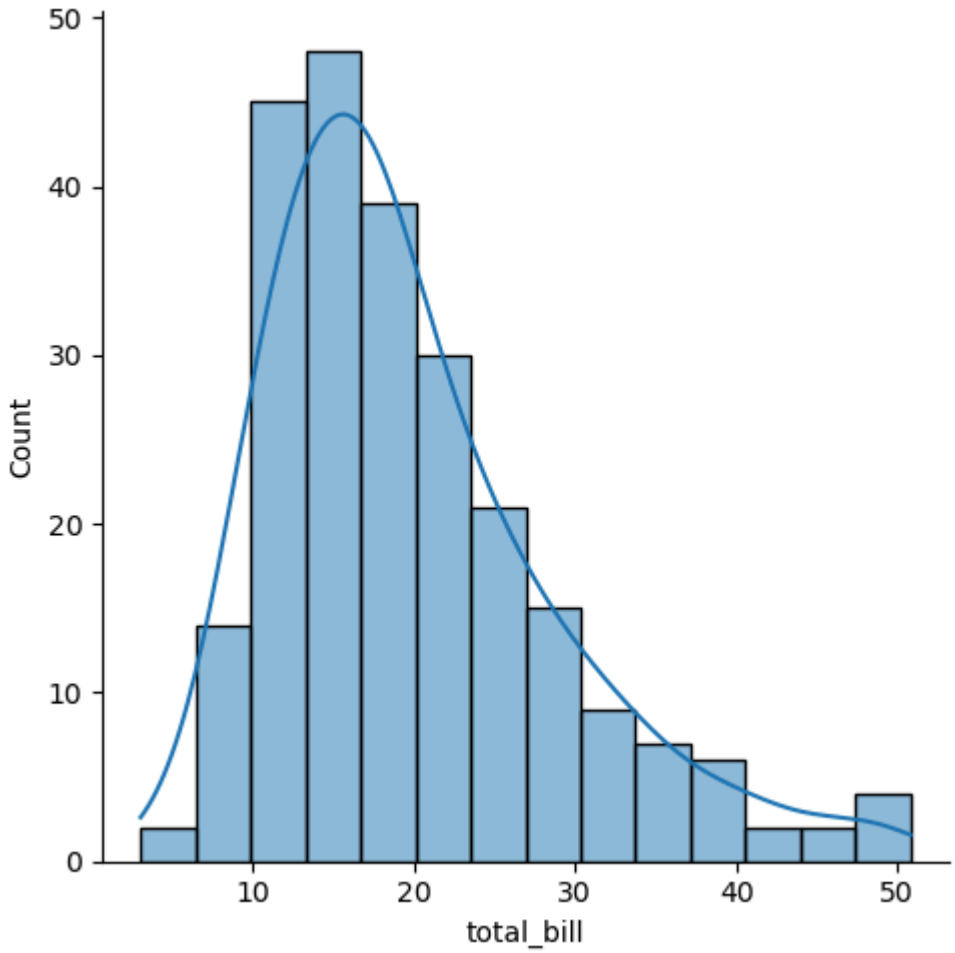
In [18]:
```
import seaborn as sns
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

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

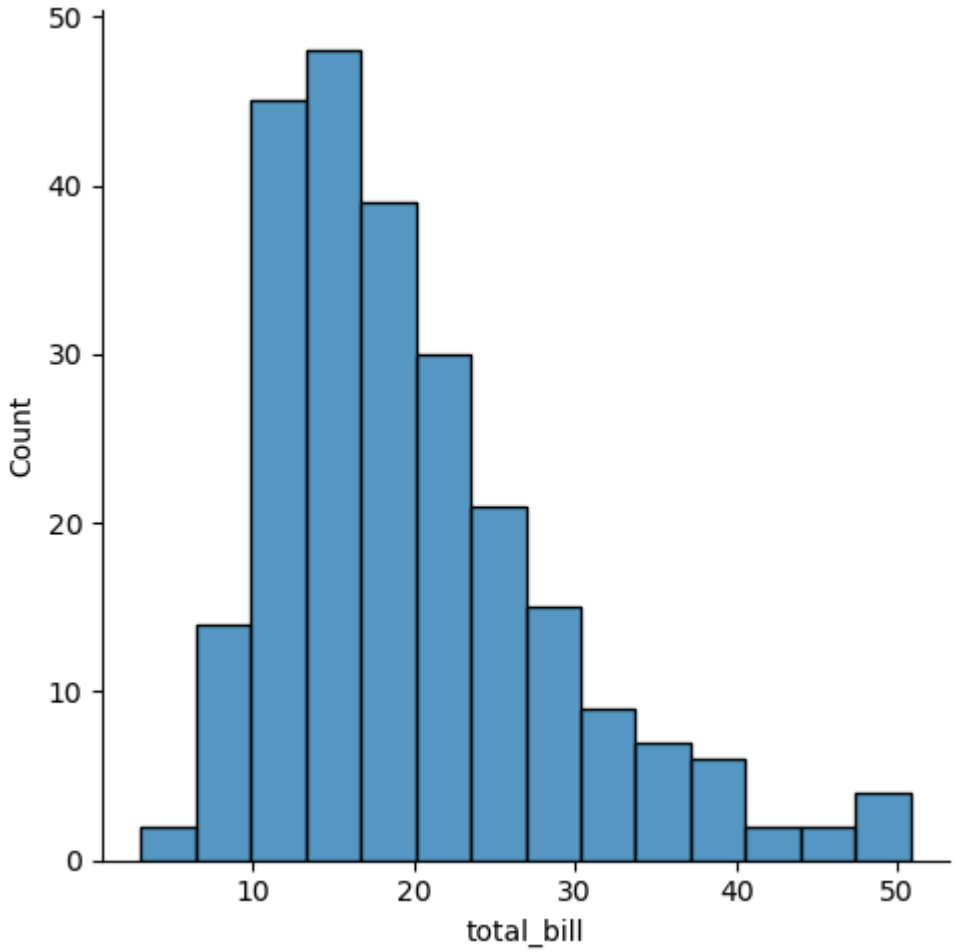
Out[18]:

	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

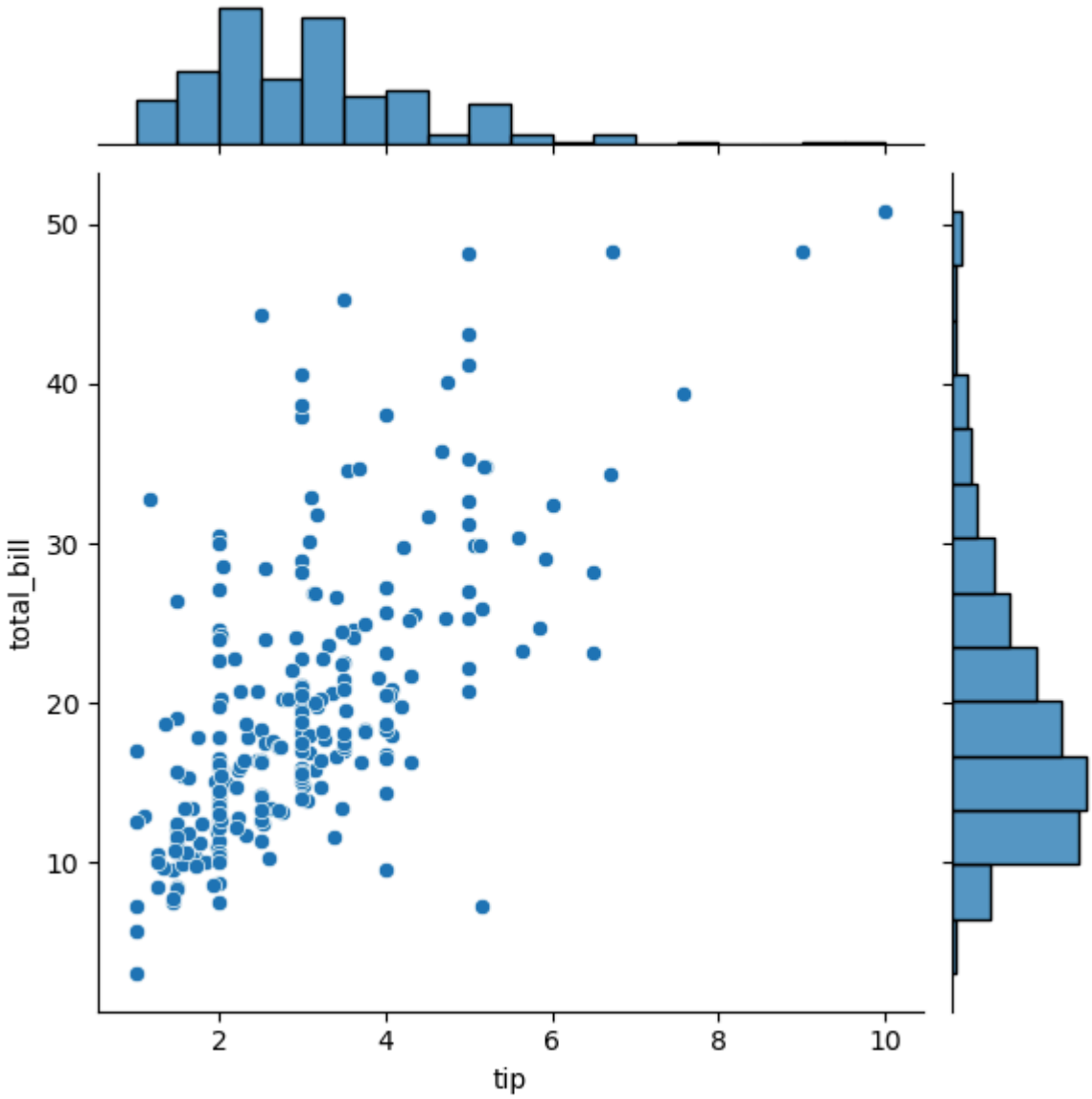
In [21]:
```
sns.displot(tips.total_bill,kde=True)
plt.show()
```



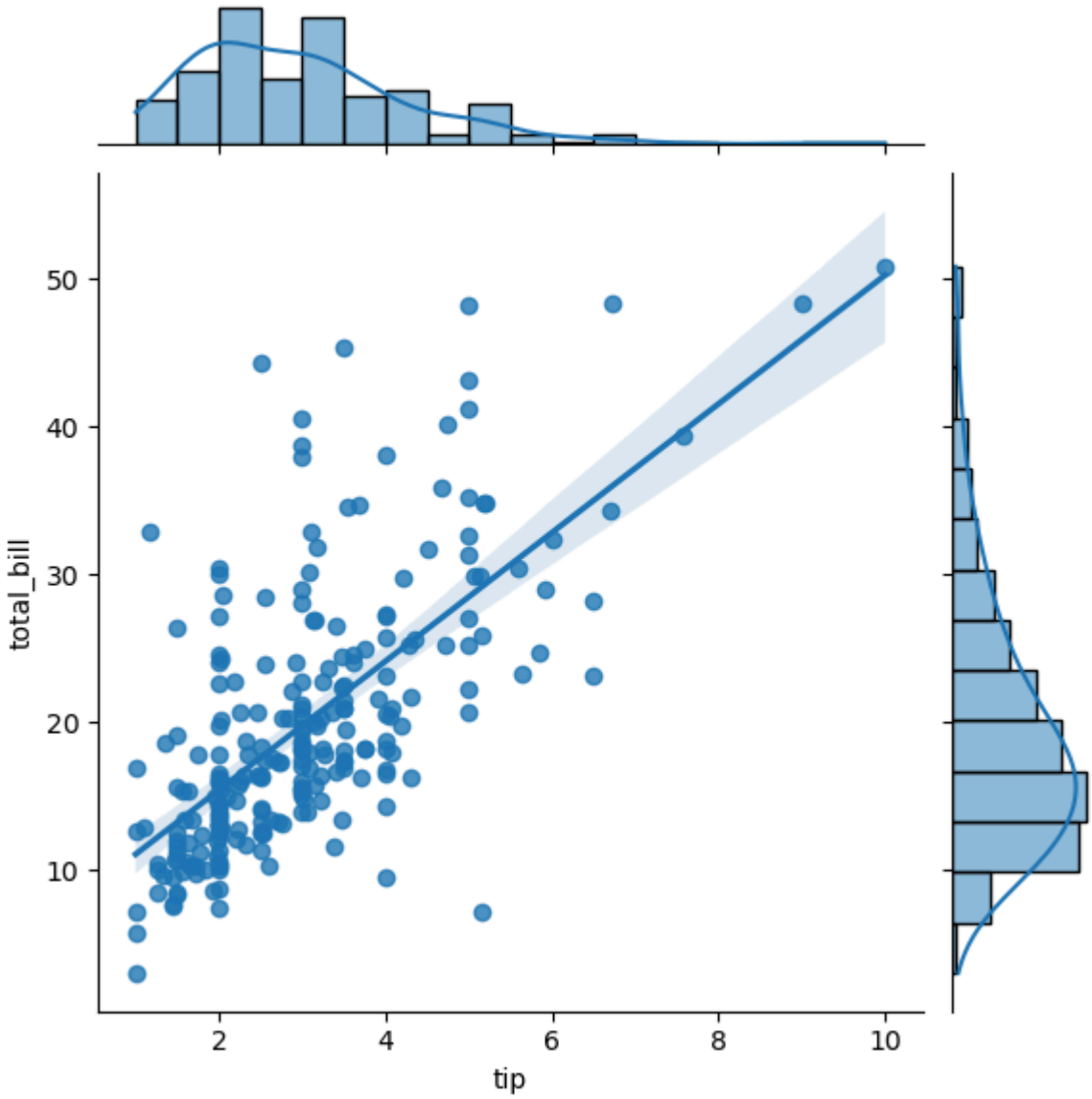
In [22]:
```
sns.displot(tips.total_bill,kde=False)
plt.show()
```



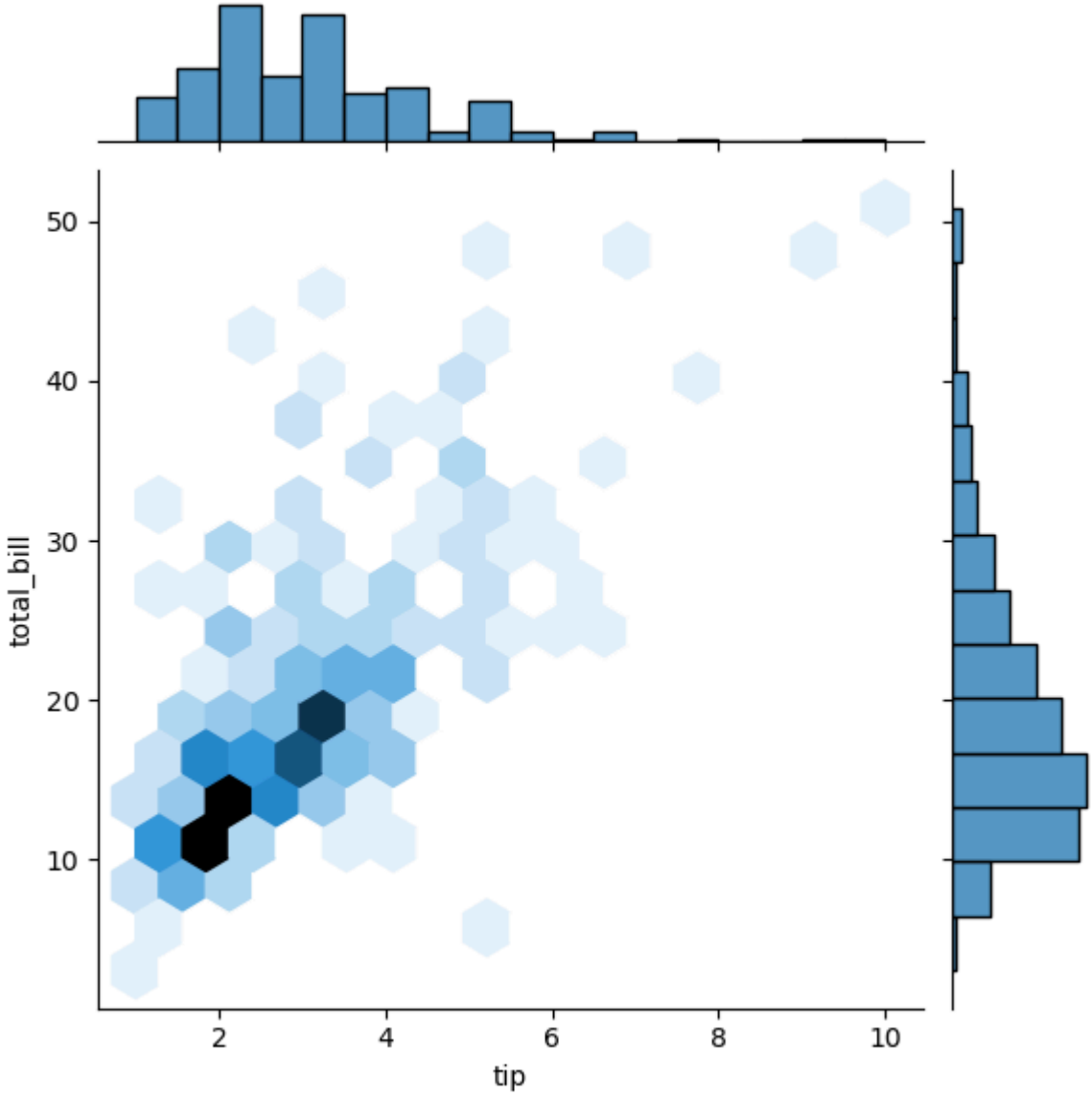
In [23]:
```
sns.jointplot(x=tips.tip,y=tips.total_bill)
plt.show()
```



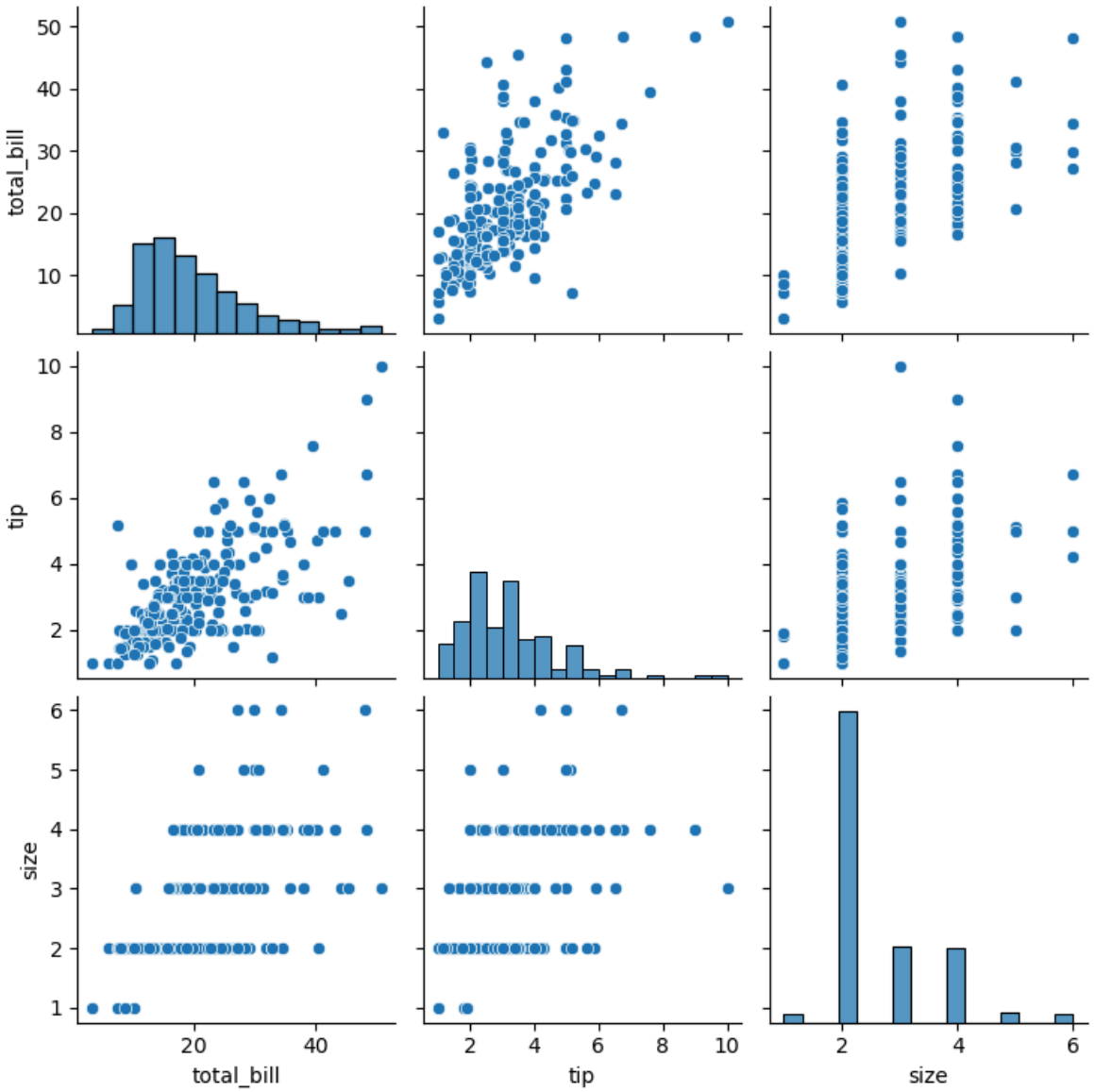
In [34]:
```
sns.jointplot(x=tips.tip,y=tips.total_bill,kind="reg")
plt.show()
```



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



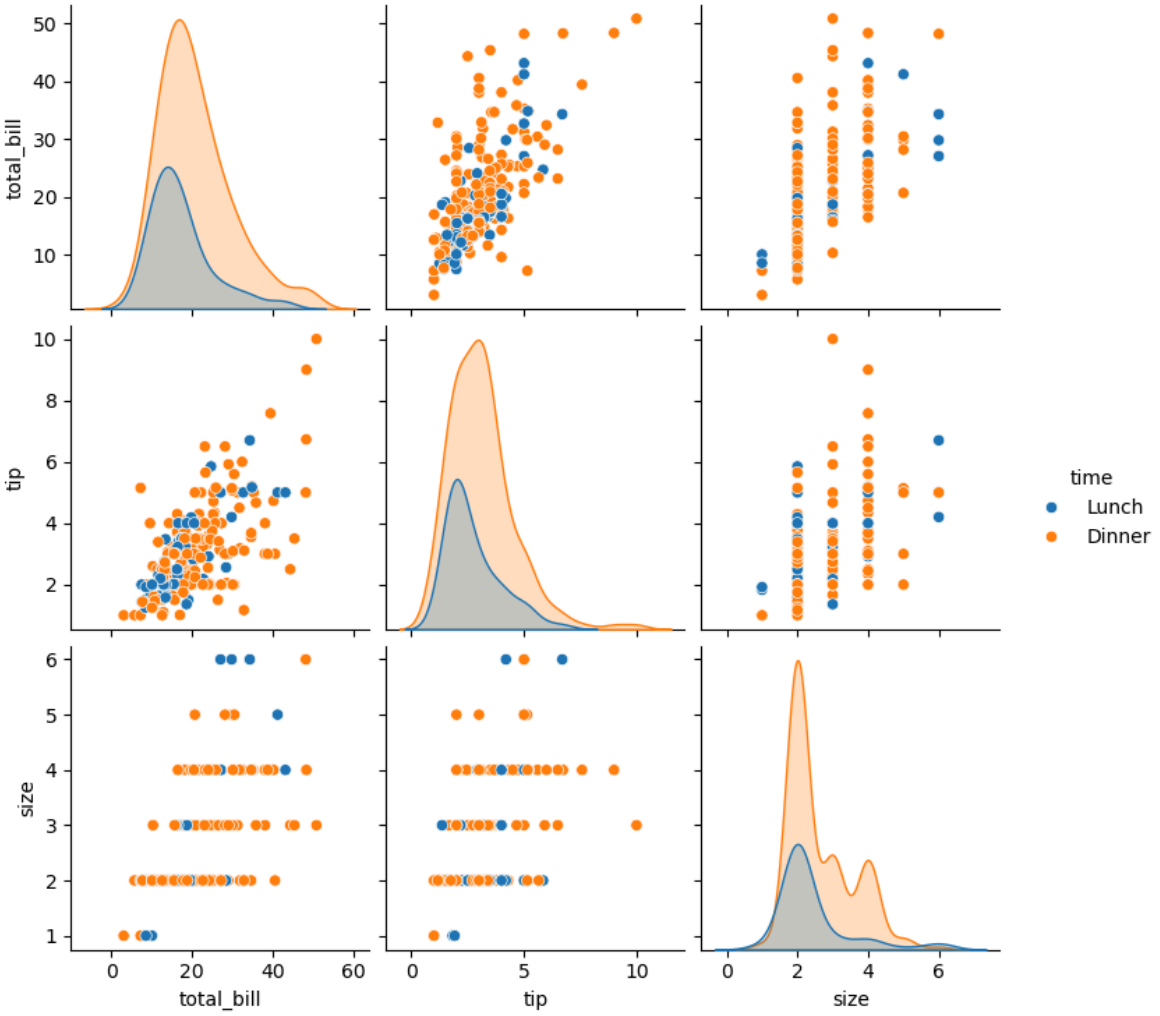
In [29]:
```
sns.pairplot(tips)
plt.show()
```



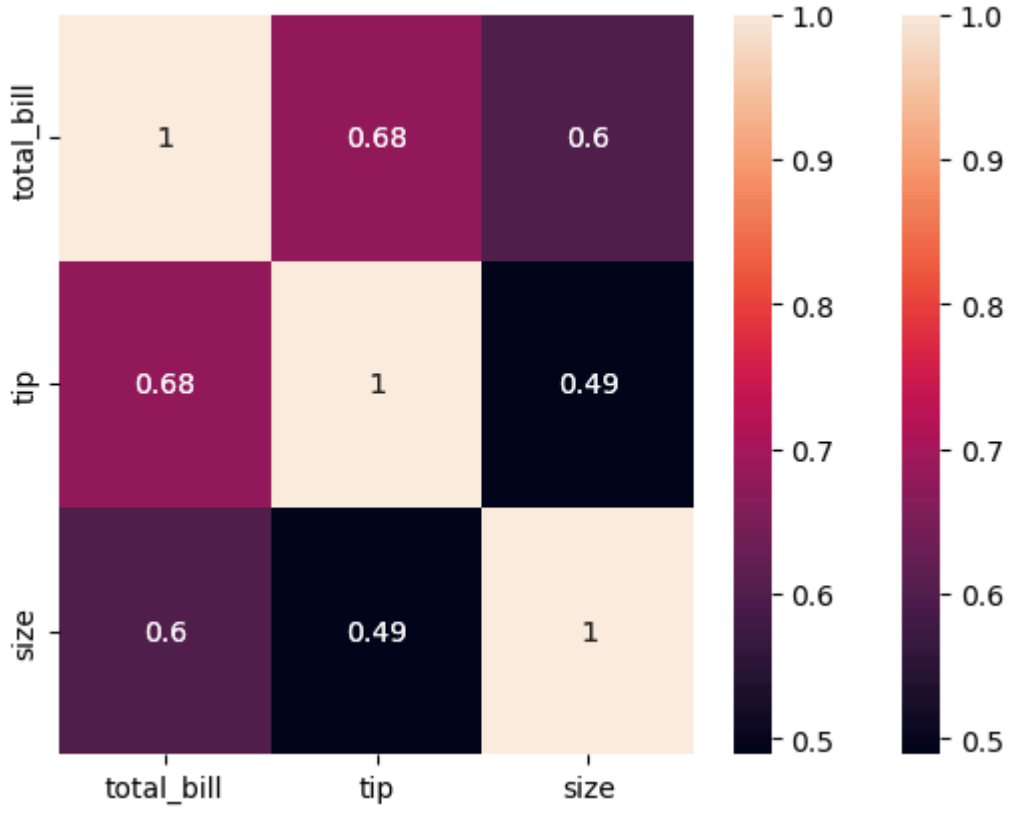
In [30]:
```
tips.time.value_counts()
```

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

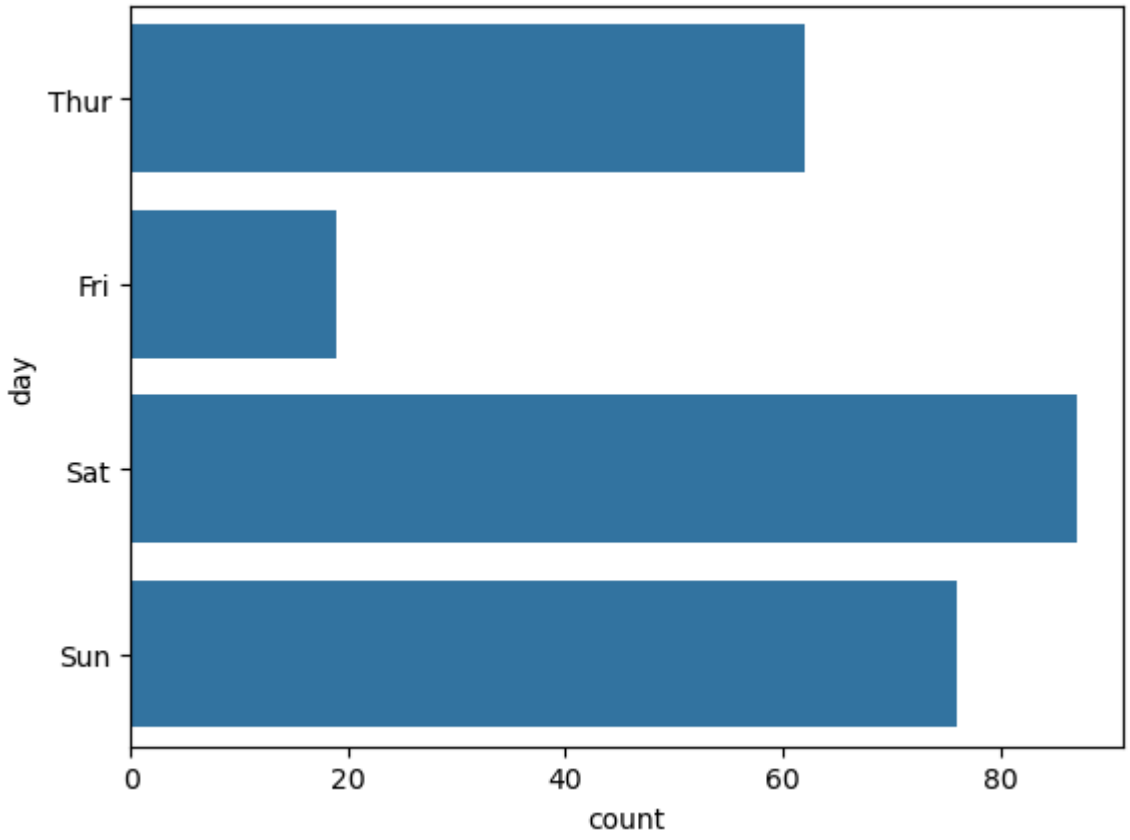
In [33]:
```
sns.pairplot(tips,hue='time')
plt.show()
```



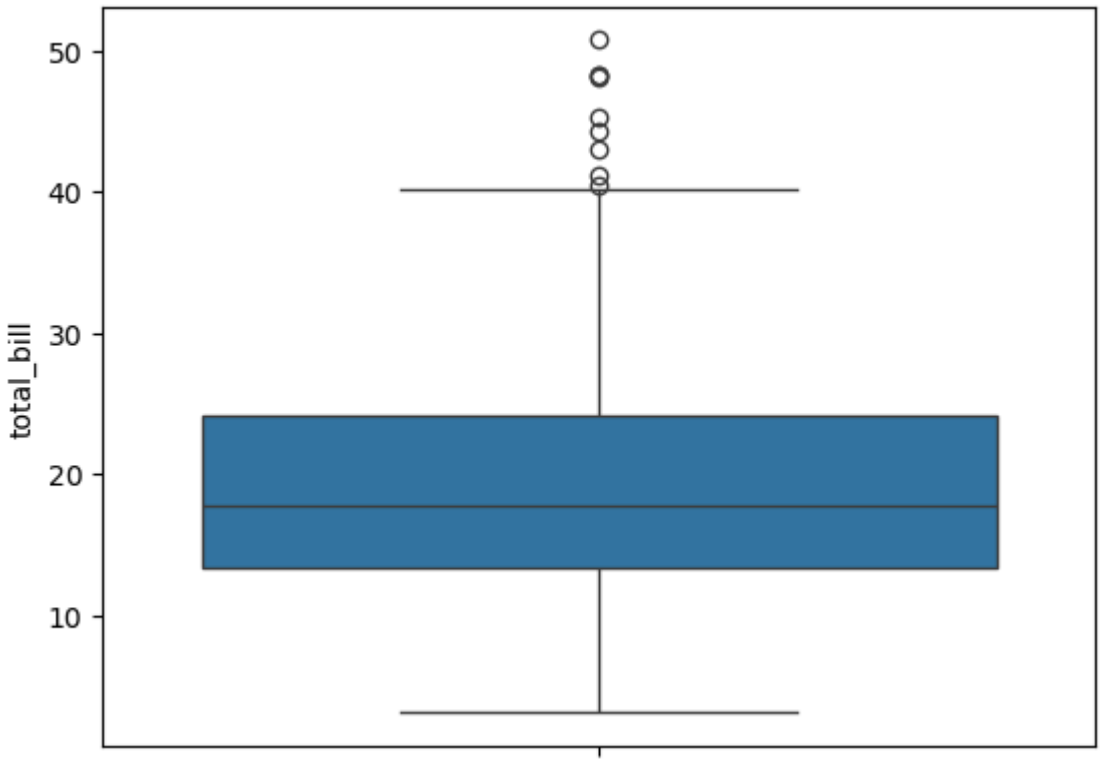
In [36]:
```
sns.heatmap(tips.corr(numeric_only=True),annot=True)
plt.show()
```



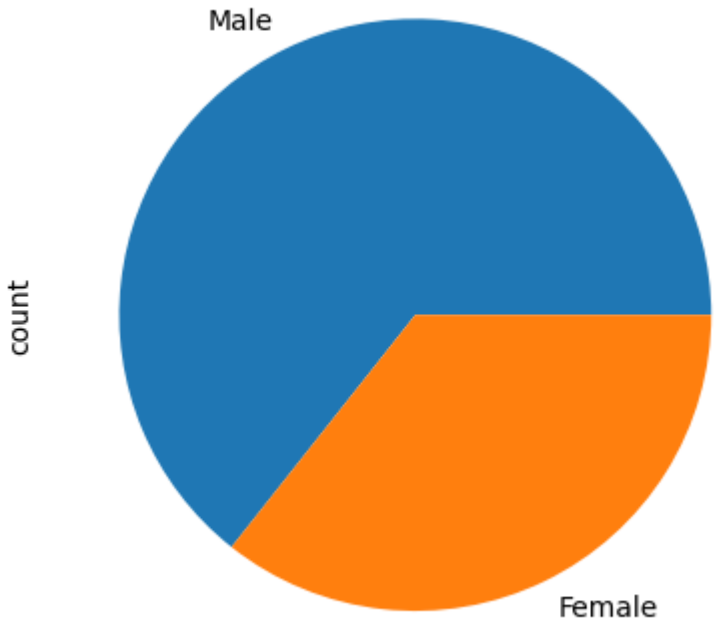
In [37]:
```
sns.countplot(tips.day)
plt.show()
```



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

Out[46]:

	EmployeeID	Age	Experience	Salary
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()
```

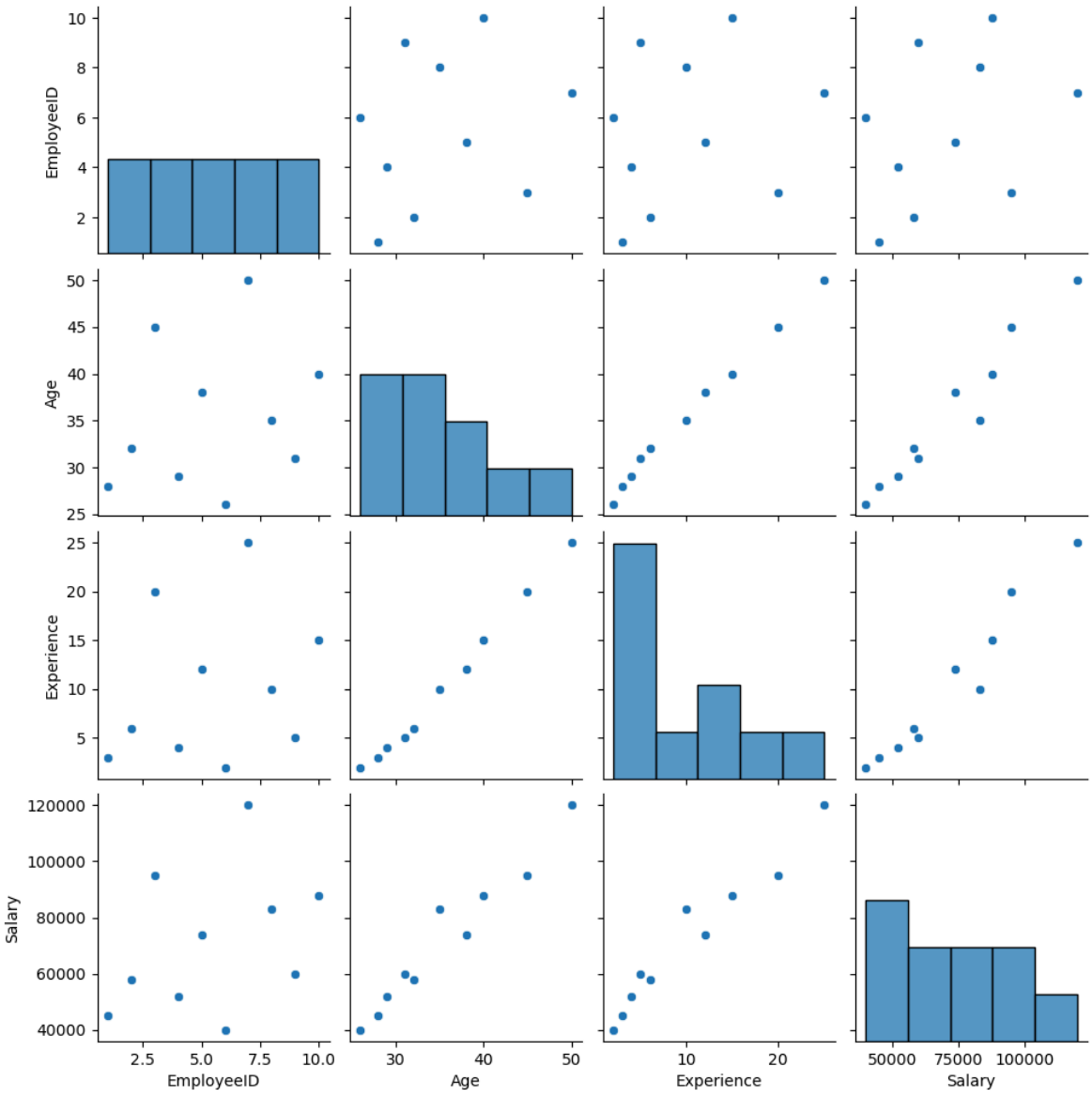
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10 entries, 0 to 9
Data columns (total 8 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   EmployeeID  10 non-null     int64
 1   Name        10 non-null     object
 2   Gender      10 non-null     object
 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 [64]:
```
df.head()
```

Out[64]:

	EmployeeID	Name	Gender	Age	Department	Experience	Education	Salary
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
EmployeeID
Age
Experience
Salary

In [ ]: