RAJALAKSHMI ENGINEERING COLLEGE

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

Department of Computer Science and Engineering

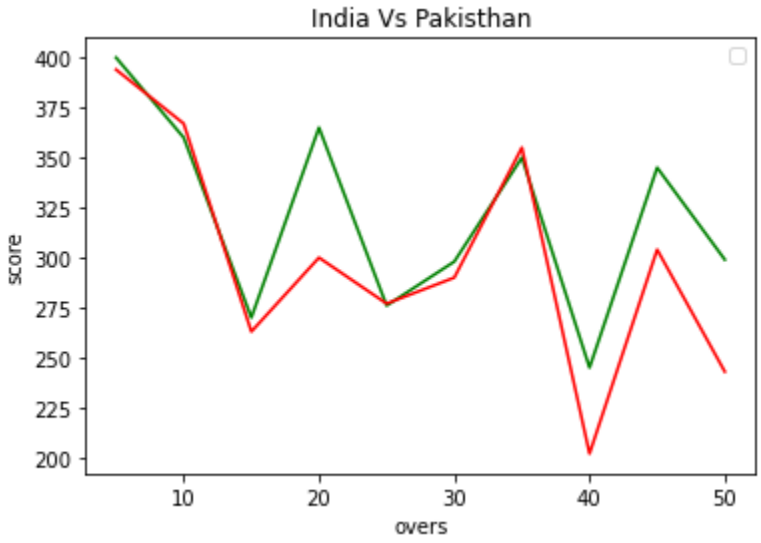
CS23334 Fundamentals of Data Science Lab

III semester II Year (2023R)

Name of the Student :M.Vishal

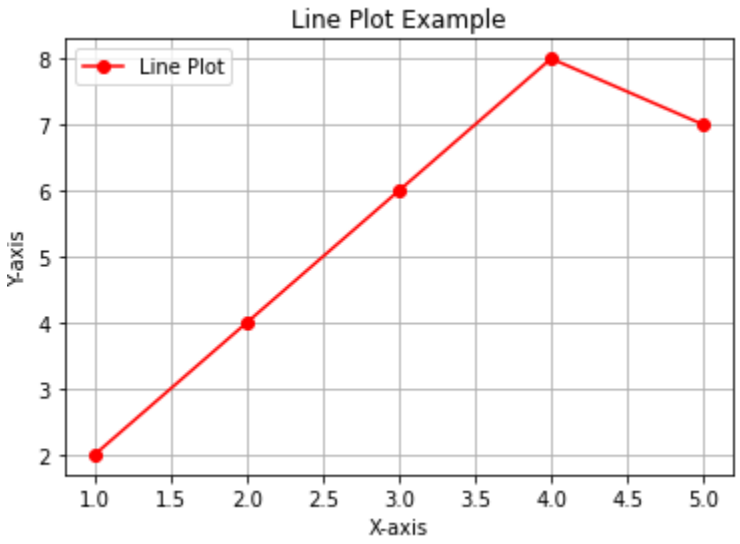
Register Number :2116240701598

```
#M VISHAL
#240701598
#22-07-2025
#Line plot
import matplotlib.pyplot as plt
overs=list(range(5,51,5))
India=[400,360,270,365,276,298,350,245,345,299]
Pakisthan=[394,367,263,300,277,290,355,202,304,243]
plt.plot(overs,India,'color'=='green')
plt.plot(overs,Pakisthan)
plt.show()
plt.title("India Vs Pakisthan")
plt.xlabel("overs")
plt.ylabel("score")
plt.legend()
plt.plot(overs,India,color="green",label="India")
plt.plot(overs,Pakisthan,color="red",label="Pakisthan")
```

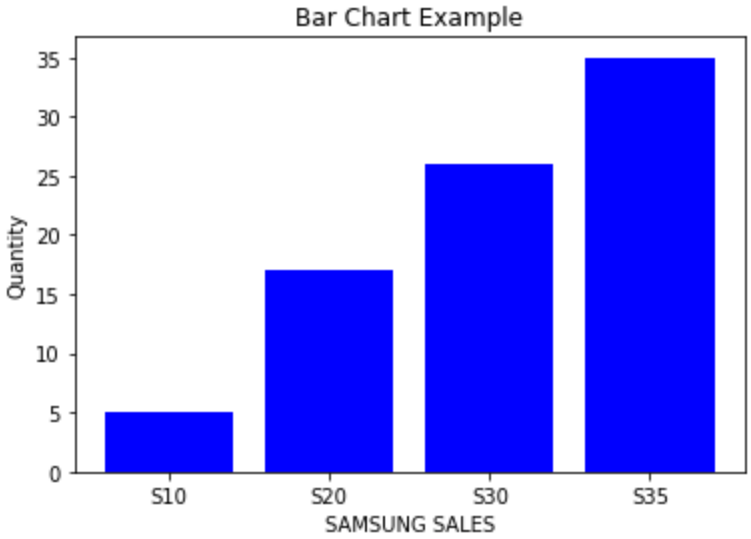


#M VISHAL
#240701598
#22-07-2025
#Line plot
import matplotlib.pyplot as plt
x = [1, 2, 3, 4, 5]
y = [2, 4, 6, 8, 7]
plt.figure(figsize=(6, 4)) # Set the figure size

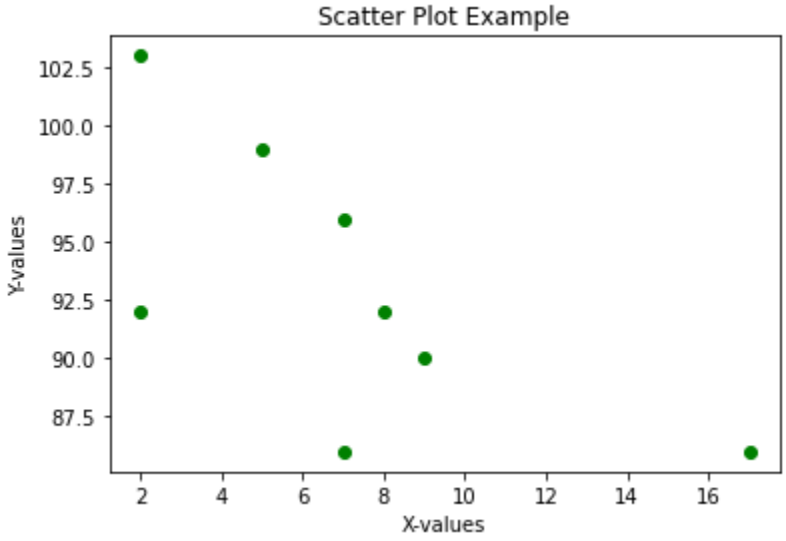
```
plt.plot(x, y, color='red', marker='o', linestyle='-', label='Line Plot')
plt.title("Line Plot Example")
plt.xlabel("X-axis")
plt.ylabel("Y-axis")
plt.legend()
plt.grid(True)
plt.show()
```




```
#M VISHAL
#240701598
#22-07-2025
#Line plot
import matplotlib.pyplot as plt
categories = ['S10', 'S20', 'S30', 'S35']
values = [5, 17, 26, 35]

plt.figure(figsize=(6, 4))
plt.bar(categories, values, color='blue')
plt.title("Bar Chart Example")
plt.xlabel("SAMSUNG SALES")
plt.ylabel("Quantity")
plt.show()
```

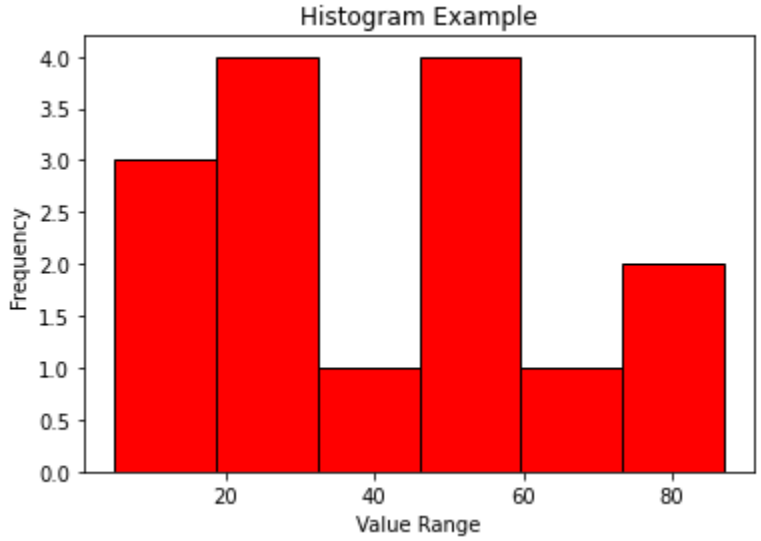



```
#M VISHAL
#240701598
#22-07-2025
#Line plot
import matplotlib.pyplot as plt
x_scatter = [5, 7, 8, 7, 2, 17, 2, 9]
y_scatter = [99, 86, 92, 96, 92, 86, 103, 90]
plt.figure(figsize=(6, 4))
plt.scatter(x_scatter, y_scatter, color='green')
plt.title("Scatter Plot Example")
plt.xlabel("X-values")
plt.ylabel("Y-values")
plt.show()
```

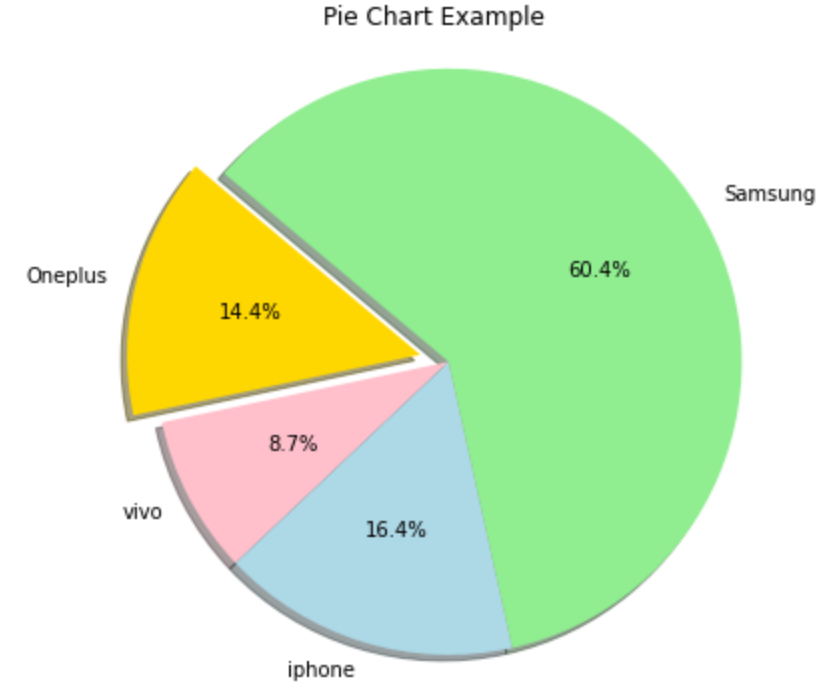


#M VISHAL

```
#240701598
#22-07-2025
#Line plot
import matplotlib.pyplot as plt
data = [22, 87, 5, 43, 56, 73, 55, 54, 11, 20, 51, 5, 79, 31, 27]
plt.figure(figsize=(6, 4))
plt.hist(data, bins=6, color='red', edgecolor='black')
plt.title("Histogram Example")
plt.xlabel("Value Range")
plt.ylabel("Frequency")
plt.show()
```




```
#M VISHAL
#240701598
#22-07-2025
#Line plot
import matplotlib.pyplot as plt
labels = ['Oneplus', 'vivo', 'iphone', 'Samsung']
sizes = [215, 130, 245, 900]
colors = ['gold', 'pink', 'lightblue', 'lightgreen']
explode = (0.1, 0, 0, 0)
plt.figure(figsize=(6, 6))
plt.pie(sizes, explode=explode, labels=labels, colors=colors,autopct='%1.1f%%', shadow=True,
startangle=140)
plt.title("Pie Chart Example")
plt.axis('equal')
plt.show()
```


Pie Chart Example
Samsung
60.4%
Oneplus
14.4%
8.7%
vivo
16.4%
iphone

Untitled27.ipynb

Code

Python 3

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
fp='./sales_data.csv'
df = pd.read_csv(fp)
print(df.head())
```

```
         Date    Product  Sales  Quantity Region
0  01-01-2023  Product A    200         4  North
1  02-01-2023  Product B    150         3  South
2  03-01-2023  Product A    220         5  North
3  04-01-2023  Product C    300         6   East
4  05-01-2023  Product B    180         4   West
```

[2]:
```
print(df.isnull().sum())

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

print(df.describe())
```

```
Date        0
Product     0
Sales       0
Quantity    0
Region      0
dtype: int64
           Sales   Quantity
count   16.000000  16.000000
mean   237.500000   5.375000
std     64.031242   1.746425
min    150.000000   3.000000
25%    187.500000   4.000000
50%    225.000000   5.500000
75%    302.500000   7.000000
max    340.000000   8.000000
```

[3]:
```
product_summary = df.groupby('Product').agg({
'Sales': 'sum',
'Quantity': 'sum'
}).reset_index()
print(product_summary)
```

```
     Product  Sales  Quantity
0  Product A   1350        33
1  Product B    850        17
2  Product C   1600        36
```

http://localhost:8888/lab/tree/Untitled27.ipynb

ernel Tabs Settings Help

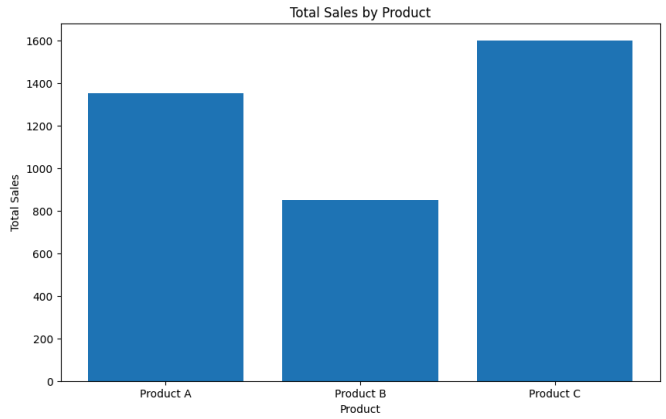
Untitled27.ipynb

Code

Python 3

```
0  Product A   1350        33
1  Product B    850        17
2  Product C   1600        36
```

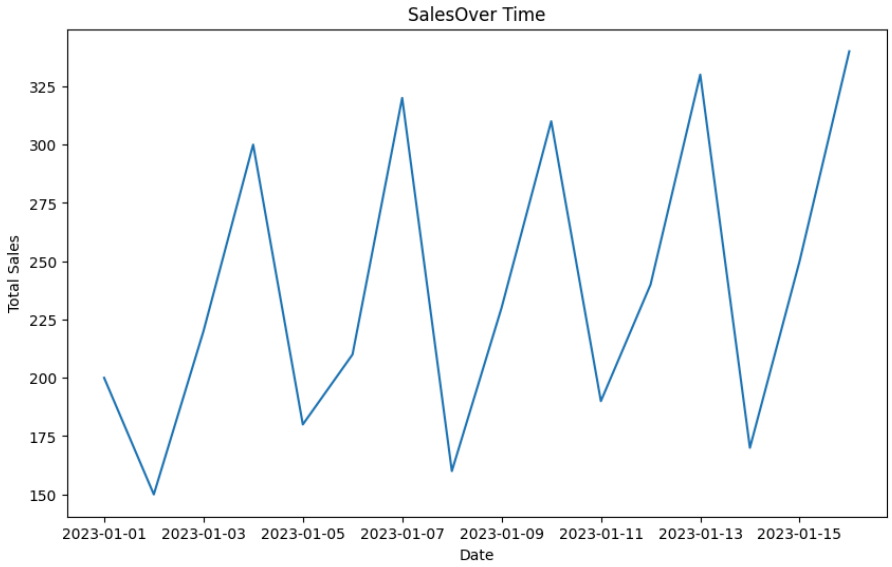
[4]:
```
plt.figure(figsize=(10, 6))
plt.bar(product_summary['Product'], product_summary['Sales'])
plt.xlabel('Product')
plt.ylabel('Total Sales')
plt.title('Total Sales by Product')
plt.show()
```



Modified
last yr.
now
last yr.
last yr.
2mo ago
2mo ago
last yr.
last yr.
2mo ago
last yr.
2mo ago
2mo ago
last yr.
last yr.
2mo ago
2mo ago
21d ago
2mo ago
2mo ago
21d ago
last yr.
last yr.
last yr.
2mo ago
13d ago
21d ago
21d ago
6d ago

Product

[6]:
```
df['Date'] = pd.to_datetime(df['Date'],format='%d-%m-%Y')
sales_over_time = df.groupby('Date').agg({'Sales': 'sum'}).reset_index()
plt.figure(figsize=(10, 6))
plt.plot(sales_over_time['Date'],sales_over_time['Sales'])
plt.xlabel('Date')
plt.ylabel('Total Sales')
plt.title('SalesOver Time')
plt.show()
```



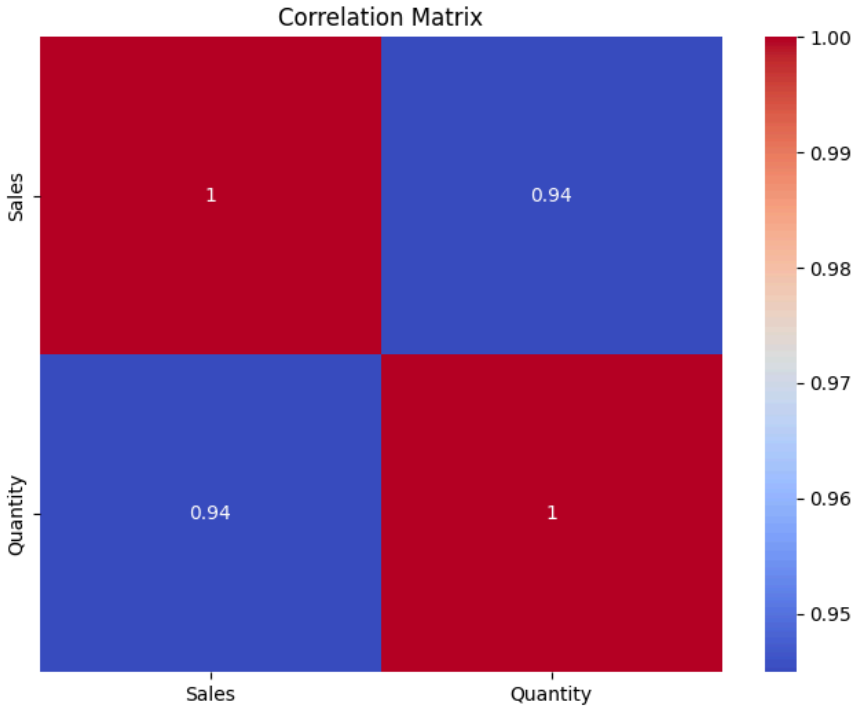
[7]:
```
pivot_table = df.pivot_table(values='Sales', index='Region', columns='Product',
aggfunc=np.sum, fill_value=0)
print(pivot_table)
```

```
Product  Product A  Product B  Product C
Region
East             0          0       1600
North         1350          0          0
South            0        480          0
West             0        370          0
```

[9]:
```
correlation_matrix = df.corr(numeric_only=True)
print(correlation_matrix)
```

```
              Sales  Quantity
Sales     1.000000  0.944922
Quantity  0.944922  1.000000
```

[10]:
```
import seaborn as sns
plt.figure(figsize=(8, 6))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')
plt.title('Correlation Matrix')
plt.show()
```



In [1]:

```
#M.vishal
#240701598
#8-5-2025
import numpy as np
import pandas as pd
df=pd.read_csv("pre_process_datasample.csv")
df
```

Out[1]:

	Country	Age	Salary	Purchased
0	France	44.0	72000.0	No
1	Spain	27.0	48000.0	Yes
2	Germany	30.0	54000.0	No
3	Spain	38.0	61000.0	No
4	Germany	40.0	NaN	Yes
5	France	35.0	58000.0	Yes
6	Spain	NaN	52000.0	No
7	France	48.0	79000.0	Yes
8	Germany	50.0	83000.0	No
9	France	37.0	67000.0	Yes

In [2]:

```
#M.vishal
#240701598
#8-5-2025
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10 entries, 0 to 9
Data columns (total 4 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   Country    10 non-null     object
 1   Age        9 non-null      float64
 2   Salary     9 non-null      float64
 3   Purchased  10 non-null     object
dtypes: float64(2), object(2)
memory usage: 452.0+ bytes
```

In [3]:

```
#M.vishal
#240701598
#8-5-2025
df.Country.mode()
```

Out[3]:

```
0    France
Name: Country, dtype: object
```

In [4]:

```
#M.vishal
#240701598
#8-5-2025
df.Country.mode()[0]
```

Out[4]:

```
'France'
```

In [5]:

```
#M.vishal
#240701598
#8-5-2025
type(df.Country.mode())
```

Out[5]:

```
pandas.core.series.Series
```

In [6]:
```
#M.vishal
#240701598
#8-5-2025
df.Country.fillna(df.Country.mode()[0],inplace=True)
df.Age.fillna(df.Age.median(),inplace=True)
df.Salary.fillna(round(df.Salary.mean()),inplace=True)
df
```

Out[6]:

	Country	Age	Salary	Purchased
0	France	44.0	72000.0	No
1	Spain	27.0	48000.0	Yes
2	Germany	30.0	54000.0	No
3	Spain	38.0	61000.0	No
4	Germany	40.0	63778.0	Yes
5	France	35.0	58000.0	Yes
6	Spain	38.0	52000.0	No
7	France	48.0	79000.0	Yes
8	Germany	50.0	83000.0	No
9	France	37.0	67000.0	Yes

In [7]:
```
#M.vishal
#240701598
#8-5-2025
pd.get_dummies(df.Country)
```

Out[7]:

	France	Germany	Spain
0	1	0	0
1	0	0	1
2	0	1	0
3	0	0	1
4	0	1	0
5	1	0	0
6	0	0	1
7	1	0	0
8	0	1	0
9	1	0	0

In [8]:
```
#M.vishal
#240701598
#8-5-2025
updated_dataset=pd.concat([pd.get_dummies(df.Country),df.iloc[:,[1,2,3]]],axis=1)
```

In [9]:
```
#M.vishal
#240701598
#8-5-2025
updated_dataset
```

Out[9]:

	France	Germany	Spain	Age	Salary	Purchased
0	1	0	0	44.0	72000.0	No
1	0	0	1	27.0	48000.0	Yes
2	0	1	0	30.0	54000.0	No
3	0	0	1	38.0	61000.0	No
4	0	1	0	40.0	63778.0	Yes
5	1	0	0	35.0	58000.0	Yes
6	0	0	1	38.0	52000.0	No
7	1	0	0	48.0	79000.0	Yes
8	0	1	0	50.0	83000.0	No
9	1	0	0	37.0	67000.0	Yes

In [10]:

```
#M.vishal
#240701598
#8-5-2025
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10 entries, 0 to 9
Data columns (total 4 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   Country    10 non-null     object 
 1   Age        10 non-null     float64
 2   Salary     10 non-null     float64
 3   Purchased  10 non-null     object 
dtypes: float64(2), object(2)
memory usage: 452.0+ bytes
```

In [11]:

```
#M.vishal
#240701598
#8-5-2025
updated_dataset.Purchased.replace(['No','Yes'],[0,1],inplace=True)
updated_dataset
```

Out[11]:

	France	Germany	Spain	Age	Salary	Purchased
0	1	0	0	44.0	72000.0	0
1	0	0	1	27.0	48000.0	1
2	0	1	0	30.0	54000.0	0
3	0	0	1	38.0	61000.0	0
4	0	1	0	40.0	63778.0	1
5	1	0	0	35.0	58000.0	1
6	0	0	1	38.0	52000.0	0
7	1	0	0	48.0	79000.0	1
8	0	1	0	50.0	83000.0	0
9	1	0	0	37.0	67000.0	1

In [2]:
```
#M.vishal
#240701598
#8-5-2025
import numpy as np
import pandas as pd
df=pd.read_csv("Hotel_Dataset.csv")
df
```

Out[2]:

	CustomerID	Age_Group	Rating(1-5)	Hotel	FoodPreference	Bill	NoOfPax	EstimatedSalary	Age_Group.1
0	1	20-25	4	Ibis	veg	1300	2	40000	20-25
1	2	30-35	5	LemonTree	Non-Veg	2000	3	59000	30-35
2	3	25-30	6	RedFox	Veg	1322	2	30000	25-30
3	4	20-25	-1	LemonTree	Veg	1234	2	120000	20-25
4	5	35+	3	Ibis	Vegetarian	989	2	45000	35+
5	6	35+	3	Ibys	Non-Veg	1909	2	122220	35+
6	7	35+	4	RedFox	Vegetarian	1000	-1	21122	35+
7	8	20-25	7	LemonTree	Veg	2999	-10	345673	20-25
8	9	25-30	2	Ibis	Non-Veg	3456	3	-99999	25-30
9	9	25-30	2	Ibis	Non-Veg	3456	3	-99999	25-30
10	10	30-35	5	RedFox	non-Veg	-6755	4	87777	30-35

In [3]:
```
#M.vishal
#240701598
#8-5-2025
df.duplicated()
```

Out[3]:
```
0     False
1     False
2     False
3     False
4     False
5     False
6     False
7     False
8     False
9      True
10    False
dtype: bool
```

In [4]:
```
#M.vishal
#240701598
#8-5-2025
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11 entries, 0 to 10
Data columns (total 9 columns):
 #   Column           Non-Null Count  Dtype 
---  ------           --------------  ----- 
 0   CustomerID       11 non-null     int64 
 1   Age_Group        11 non-null     object
 2   Rating(1-5)      11 non-null     int64 
 3   Hotel            11 non-null     object
 4   FoodPreference   11 non-null     object
 5   Bill             11 non-null     int64 
 6   NoOfPax          11 non-null     int64 
 7   EstimatedSalary  11 non-null     int64 
 8   Age_Group.1      11 non-null     object
dtypes: int64(5), object(4)
memory usage: 924.0+ bytes
```

In [5]:
```
#M.vishal
#240701598
#8-5-2025
df.drop_duplicates(inplace=True)
df
```

Out[5]:

	CustomerID	Age_Group	Rating(1-5)	Hotel	FoodPreference	Bill	NoOfPax	EstimatedSalary	Age_Group.1
0	1	20-25	4	Ibis	veg	1300	2	40000	20-25
1	2	30-35	5	LemonTree	Non-Veg	2000	3	59000	30-35
2	3	25-30	6	RedFox	Veg	1322	2	30000	25-30
3	4	20-25	-1	LemonTree	Veg	1234	2	120000	20-25
4	5	35+	3	Ibis	Vegetarian	989	2	45000	35+
5	6	35+	3	Ibys	Non-Veg	1909	2	122220	35+
6	7	35+	4	RedFox	Vegetarian	1000	-1	21122	35+
7	8	20-25	7	LemonTree	Veg	2999	-10	345673	20-25
8	9	25-30	2	Ibis	Non-Veg	3456	3	-99999	25-30
10	10	30-35	5	RedFox	non-Veg	-6755	4	87777	30-35

In [6]:
```
#M.vishal
#240701598
#8-5-2025
len(df)
```

Out[6]:
```
10
```

In [17]:
```
#M.vishal
#240701598
#8-5-2025
index=np.array(list(range(0,len(df))))
df.set_index(index,inplace=True)
index
```

Out[17]:
```
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

In [8]:
```
#M.vishal
#240701598
#8-5-2025
df
```

Out[8]:

	CustomerID	Age_Group	Rating(1-5)	Hotel	FoodPreference	Bill	NoOfPax	EstimatedSalary	Age_Group.1
0	1	20-25	4	Ibis	veg	1300	2	40000	20-25
1	2	30-35	5	LemonTree	Non-Veg	2000	3	59000	30-35
2	3	25-30	6	RedFox	Veg	1322	2	30000	25-30
3	4	20-25	-1	LemonTree	Veg	1234	2	120000	20-25
4	5	35+	3	Ibis	Vegetarian	989	2	45000	35+
5	6	35+	3	Ibys	Non-Veg	1909	2	122220	35+
6	7	35+	4	RedFox	Vegetarian	1000	-1	21122	35+
7	8	20-25	7	LemonTree	Veg	2999	-10	345673	20-25
8	9	25-30	2	Ibis	Non-Veg	3456	3	-99999	25-30
9	10	30-35	5	RedFox	non-Veg	-6755	4	87777	30-35

In [9]:
```
#M.vishal
#240701598
#8-5-2025
df.drop(['Age_Group.1'],axis=1,inplace=True)
df
```

Out[9]:

	CustomerID	Age_Group	Rating(1-5)	Hotel	FoodPreference	Bill	NoOfPax	EstimatedSalary
0	1	20-25	4	Ibis	veg	1300	2	40000
1	2	30-35	5	LemonTree	Non-Veg	2000	3	59000
2	3	25-30	6	RedFox	Veg	1322	2	30000
3	4	20-25	-1	LemonTree	Veg	1234	2	120000
4	5	35+	3	Ibis	Vegetarian	989	2	45000
5	6	35+	3	Ibys	Non-Veg	1909	2	122220
6	7	35+	4	RedFox	Vegetarian	1000	-1	21122
7	8	20-25	7	LemonTree	Veg	2999	-10	345673
8	9	25-30	2	Ibis	Non-Veg	3456	3	-99999
9	10	30-35	5	RedFox	non-Veg	-6755	4	87777

In [10]:
```
#M.vishal
#240701598
#8-5-2025
df.CustomerID.loc[df.CustomerID<0]=np.nan
df.Bill.loc[df.Bill<0]=np.nan
df.EstimatedSalary.loc[df.EstimatedSalary<0]=np.nan
df
```

```
C:\Users\hdc0422206\AppData\Local\Temp\ipykernel_84\2080958306.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing
rsus-a-copy
  df.CustomerID.loc[df.CustomerID<0]=np.nan
C:\Users\hdc0422206\AppData\Local\Temp\ipykernel_84\2080958306.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing
rsus-a-copy
  df.Bill.loc[df.Bill<0]=np.nan
C:\Users\hdc0422206\AppData\Local\Temp\ipykernel_84\2080958306.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing
rsus-a-copy
  df.EstimatedSalary.loc[df.EstimatedSalary<0]=np.nan
```

Out[10]:

	CustomerID	Age_Group	Rating(1-5)	Hotel	FoodPreference	Bill	NoOfPax	EstimatedSalary
0	1.0	20-25	4	Ibis	veg	1300.0	2	40000.0
1	2.0	30-35	5	LemonTree	Non-Veg	2000.0	3	59000.0
2	3.0	25-30	6	RedFox	Veg	1322.0	2	30000.0
3	4.0	20-25	-1	LemonTree	Veg	1234.0	2	120000.0
4	5.0	35+	3	Ibis	Vegetarian	989.0	2	45000.0
5	6.0	35+	3	Ibys	Non-Veg	1909.0	2	122220.0
6	7.0	35+	4	RedFox	Vegetarian	1000.0	-1	21122.0
7	8.0	20-25	7	LemonTree	Veg	2999.0	-10	345673.0
8	9.0	25-30	2	Ibis	Non-Veg	3456.0	3	NaN
9	10.0	30-35	5	RedFox	non-Veg	NaN	4	87777.0

In [11]:
```
#M.vishal
#240701598
#8-5-2025
df['NoOfPax'].loc[(df['NoOfPax']<1) | (df['NoOfPax']>20)]=np.nan
df
```

```
C:\Users\hdc0422206\AppData\Local\Temp\ipykernel_84\2129877948.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/i
rsus-a-copy
  df['NoOfPax'].loc[(df['NoOfPax']<1) | (df['NoOfPax']>20)]=np.nan
```

Out[11]:

	CustomerID	Age_Group	Rating(1-5)	Hotel	FoodPreference	Bill	NoOfPax	EstimatedSalary
0	1.0	20-25	4	Ibis	veg	1300.0	2.0	40000.0
1	2.0	30-35	5	LemonTree	Non-Veg	2000.0	3.0	59000.0
2	3.0	25-30	6	RedFox	Veg	1322.0	2.0	30000.0
3	4.0	20-25	-1	LemonTree	Veg	1234.0	2.0	120000.0
4	5.0	35+	3	Ibis	Vegetarian	989.0	2.0	45000.0
5	6.0	35+	3	Ibys	Non-Veg	1909.0	2.0	122220.0
6	7.0	35+	4	RedFox	Vegetarian	1000.0	NaN	21122.0
7	8.0	20-25	7	LemonTree	Veg	2999.0	NaN	345673.0
8	9.0	25-30	2	Ibis	Non-Veg	3456.0	3.0	NaN
9	10.0	30-35	5	RedFox	non-Veg	NaN	4.0	87777.0

In [12]:
```
#M.vishal
#240701598
#8-5-2025
df.Age_Group.unique()
```

Out[12]:
```
array(['20-25', '30-35', '25-30', '35+'], dtype=object)
```

In [13]:
```
#M.vishal
#240701598
#8-5-2025
df.Hotel.unique()
```

Out[13]:
```
array(['Ibis', 'LemonTree', 'RedFox', 'Ibys'], dtype=object)
```

In [14]:
```
#M.vishal
#240701598
#8-5-2025
df.Hotel.replace(['Ibys'],'Ibis',inplace=True)
df.FoodPreference.unique
```

Out[14]:
```
<bound method Series.unique of 0           veg
1       Non-Veg
2           Veg
3           Veg
4    Vegetarian
5       Non-Veg
6    Vegetarian
7           Veg
8       Non-Veg
9       non-Veg
Name: FoodPreference, dtype: object>
```

In [15]:
```
#M.vishal
#240701598
#8-5-2025
df.FoodPreference.replace(['Vegetarian','veg'],'Veg',inplace=True)
df.FoodPreference.replace(['non-Veg'],'Non-Veg',inplace=True)
```

In [16]:
```
#M.vishal
#240701598
#8-5-2025
df.EstimatedSalary.fillna(round(df.EstimatedSalary.mean()),inplace=True)
df.NoOfPax.fillna(round(df.NoOfPax.median()),inplace=True)
df['Rating(1-5)'].fillna(round(df['Rating(1-5)'].median()), inplace=True)
df.Bill.fillna(round(df.Bill.mean()),inplace=True)
df
```

Out[16]:

	CustomerID	Age_Group	Rating(1-5)	Hotel	FoodPreference	Bill	NoOfPax	EstimatedSalary
0	1.0	20-25	4	Ibis	Veg	1300.0	2.0	40000.0
1	2.0	30-35	5	LemonTree	Non-Veg	2000.0	3.0	59000.0
2	3.0	25-30	6	RedFox	Veg	1322.0	2.0	30000.0
3	4.0	20-25	-1	LemonTree	Veg	1234.0	2.0	120000.0
4	5.0	35+	3	Ibis	Veg	989.0	2.0	45000.0
5	6.0	35+	3	Ibis	Non-Veg	1909.0	2.0	122220.0
6	7.0	35+	4	RedFox	Veg	1000.0	2.0	21122.0
7	8.0	20-25	7	LemonTree	Veg	2999.0	2.0	345673.0
8	9.0	25-30	2	Ibis	Non-Veg	3456.0	3.0	96755.0
9	10.0	30-35	5	RedFox	Non-Veg	1801.0	4.0	87777.0

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

Out[1]:
```
array([47, 51, 74, 85, 65, 96, 77, 21, 85, 72, 80, 50, 51, 74, 99, 17])
```

In [2]:
```
array.mean()
```

Out[2]:
```
65.25
```

In [3]:
```
np.percentile(array,25)
```

Out[3]:
```
50.75
```

In [4]:
```
np.percentile(array,50)
```

Out[4]:
```
73.0
```

In [5]:
```
np.percentile(array,75)
```

Out[5]:
```
81.25
```

In [6]:
```
np.percentile(array,100)
```

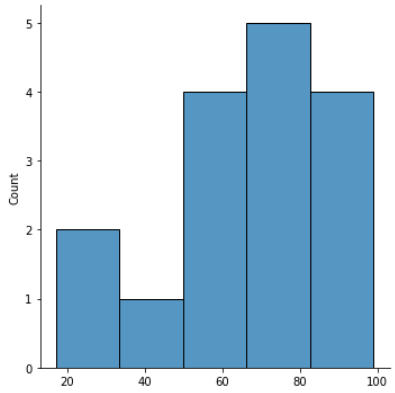
Out[6]:
```
99.0
```

In [8]:
```
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[8]:
```
(5.0, 127.0)
```

In [9]:
```
import seaborn as sns
%matplotlib inline
sns.displot(array)
```

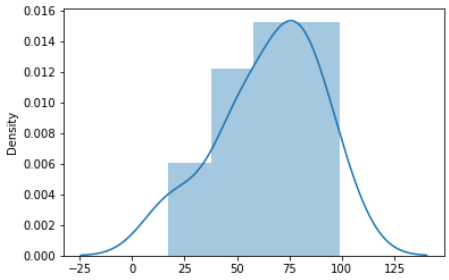
Out[9]:
```
<seaborn.axisgrid.FacetGrid at 0x233f706efa0>
```



In [10]:
```
sns.distplot(array)
```

```
C:\Users\HDC0422193\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated functi
on and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with simila
r flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[10]: <AxesSubplot:ylabel='Density'>

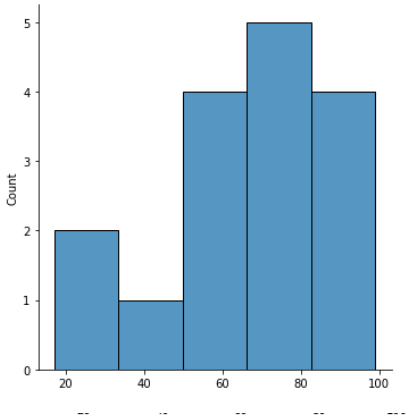


In [11]:
```
new_array=array[(array>lr) & (array<ur)]
new_array
```

Out[11]: array([47, 51, 74, 85, 65, 96, 77, 21, 85, 72, 80, 50, 51, 74, 99, 17])

In [12]:
```
sns.displot(new_array)
```

Out[12]: <seaborn.axisgrid.FacetGrid at 0x233f7de6100>



In [13]:
```
lr1,ur1=outDetection(new_array)
lr1,ur1
```

Out[13]: (5.0, 127.0)

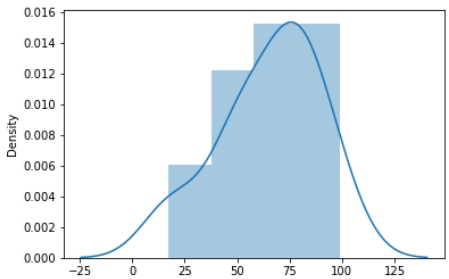
In [14]:
```
final_array=new_array[(new_array>lr1) & (new_array<ur1)]
final_array
```

Out[14]: array([47, 51, 74, 85, 65, 96, 77, 21, 85, 72, 80, 50, 51, 74, 99, 17])

In [15]:
```
sns.distplot(final_array)
```

```
C:\Users\HDC0422193\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated functi
on and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with simila
r flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[15]: <AxesSubplot:ylabel='Density'>



In [ ]:

[19]:
```
import numpy as np
import pandas as pd
df=pd.read_csv('pre_process_datasample.csv')
```

[23]:
```
df
```

[23]:

	Country	Age	Salary	Purchased
0	France	44.0	72000.0	No
1	Spain	27.0	48000.0	Yes
2	Germany	30.0	54000.0	No
3	Spain	38.0	61000.0	No
4	Germany	40.0	NaN	Yes
5	France	35.0	58000.0	Yes
6	Spain	NaN	52000.0	No
7	France	48.0	79000.0	Yes
8	Germany	50.0	83000.0	No
9	France	37.0	67000.0	Yes

[25]:
```
df.head()
```

[25]:

	Country	Age	Salary	Purchased
0	France	44.0	72000.0	No
1	Spain	27.0	48000.0	Yes
2	Germany	30.0	54000.0	No
3	Spain	38.0	61000.0	No
4	Germany	40.0	NaN	Yes

[27]:
```
df.Country.fillna(df.Country.mode()[0],inplace=True)
features=df.iloc[:,:-1].values
```

[29]:
```
label=df.iloc[:,-1].values
```

[31]:
```
from sklearn.impute import SimpleImputer
age=SimpleImputer(strategy="mean",missing_values=np.nan)
Salary=SimpleImputer(strategy="mean",missing_values=np.nan)
```

[39]:
```
age.fit(features[:,[1]])
```

[39]:
```
age.fit(features[:,[1]])
```

[39]:
```
SimpleImputer()
```

[41]:
```
Salary.fit(features[:,[2]])
```

[41]:
```
SimpleImputer()
```

[43]:
```
SimpleImputer()
```

[43]:
```
SimpleImputer()
```

[45]:
```
features[:,[1]]=age.transform(features[:,[1]])
features[:,[2]]=Salary.transform(features[:,[2]])
features
```

[45]:
```
array([['France', 44.0, 72000.0],
       ['Spain', 27.0, 48000.0],
       ['Germany', 30.0, 54000.0],
       ['Spain', 38.0, 61000.0],
       ['Germany', 40.0, 63777.77777777778],
       ['France', 35.0, 58000.0],
       ['Spain', 38.77777777777778, 52000.0],
       ['France', 48.0, 79000.0],
       ['Germany', 50.0, 83000.0],
       ['France', 37.0, 67000.0]], dtype=object)
```

[49]:
```
from sklearn.preprocessing import OneHotEncoder
oh = OneHotEncoder(sparse_output=False)
Country=oh.fit_transform(features[:,[0]])
Country
```

[49]:
```
array([[1., 0., 0.],
       [0., 0., 1.],
       [0., 1., 0.],
       [0., 0., 1.],
       [0., 1., 0.],
       [1., 0., 0.],
       [0., 0., 1.],
       [1., 0., 0.],
       [0., 1., 0.],
       [1., 0., 0.]])
```

[51]:
```
final_set=np.concatenate((Country,features[:,[1,2]]),axis=1)
final_set
```

[51]:
```
array([[1.0, 0.0, 0.0, 44.0, 72000.0],
       [0.0, 0.0, 1.0, 27.0, 48000.0],
       [0.0, 1.0, 0.0, 30.0, 54000.0],
       [0.0, 0.0, 1.0, 38.0, 61000.0],
       [0.0, 1.0, 0.0, 40.0, 63777.77777777778],
       [1.0, 0.0, 0.0, 35.0, 58000.0],
       [0.0, 0.0, 1.0, 38.77777777777778, 52000.0],
       [1.0, 0.0, 0.0, 48.0, 79000.0],
       [0.0, 1.0, 0.0, 50.0, 83000.0],
       [1.0, 0.0, 0.0, 37.0, 67000.0]], dtype=object)
```

[53]:
```
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
sc.fit(final_set)
feat_standard_scaler=sc.transform(final_set)
feat_standard_scaler
```

[53]:
```
array([[ 1.22474487e+00, -6.54653671e-01, -6.54653671e-01,
          7.58874362e-01,  7.49473254e-01],
       [-8.16496581e-01, -6.54653671e-01,  1.52752523e+00,
        -1.71150388e+00, -1.43817841e+00],
       [-8.16496581e-01,  1.52752523e+00, -6.54653671e-01,
        -1.27555478e+00, -8.91265492e-01],
       [-8.16496581e-01, -6.54653671e-01,  1.52752523e+00,
        -1.13023841e-01, -2.53200424e-01],
       [-8.16496581e-01,  1.52752523e+00, -6.54653671e-01,
         1.77608893e-01,  6.63219199e-16],
       [ 1.22474487e+00, -6.54653671e-01, -6.54653671e-01,
        -5.48972942e-01, -5.26656882e-01],
       [-8.16496581e-01, -6.54653671e-01,  1.52752523e+00,
         0.00000000e+00, -1.07356980e+00],
       [ 1.22474487e+00, -6.54653671e-01, -6.54653671e-01,
         1.34013983e+00,  1.38753832e+00],
       [-8.16496581e-01,  1.52752523e+00, -6.54653671e-01,
         1.63077256e+00,  1.75214693e+00],
       [ 1.22474487e+00, -6.54653671e-01, -6.54653671e-01,
        -2.58340208e-01,  2.93712492e-01]])
```

[55]:
```
from sklearn.preprocessing import MinMaxScaler
mms=MinMaxScaler(feature_range=(0,1))
mms.fit(final_set)
feat_minmax_scaler=mms.transform(final_set)
feat_minmax_scaler
```

[55]:
```
array([[1.        , 0.        , 0.        , 0.73913043, 0.68571429],
       [0.        , 0.        , 1.        , 0.        , 0.        ],
       [0.        , 1.        , 0.        , 0.13043478, 0.17142857],
       [0.        , 0.        , 1.        , 0.47826087, 0.37142857],
       [0.        , 1.        , 0.        , 0.56521739, 0.45079365],
       [1.        , 0.        , 0.        , 0.34782609, 0.28571429],
       [0.        , 0.        , 1.        , 0.51207729, 0.11428571],
       [1.        , 0.        , 0.        , 0.91304348, 0.88571429],
       [0.        , 1.        , 0.        , 1.        , 1.        ],
       [1.        , 0.        , 0.        , 0.43478261, 0.54285714]])
```

[ ]:

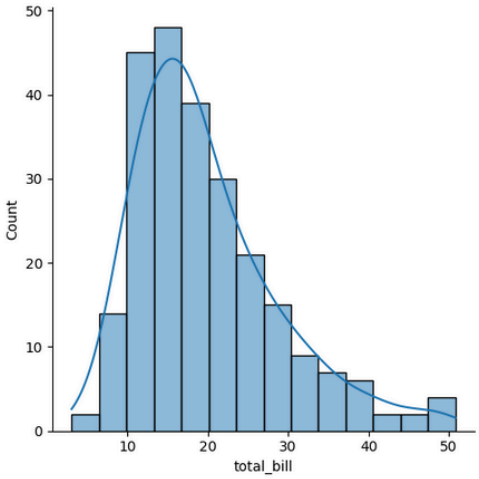
*[2]:
```
#M.Vishal
#240701598
#CSE
#26-08-2025

import seaborn as sns
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
tips=sns.load_dataset('tips')
tips.head()
```

[2]:

	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

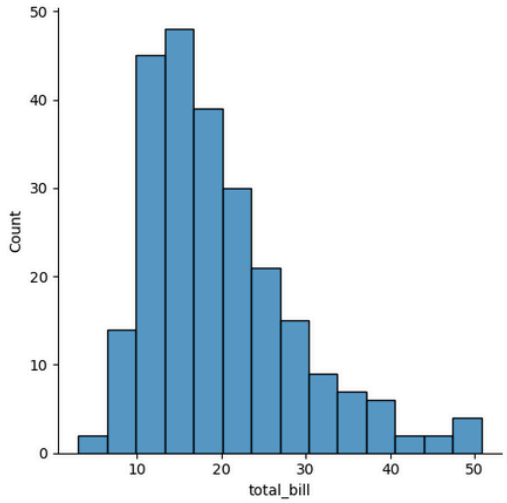
*[5]:
```
#M.Vishal
#240701598
#CSE
#26-08-2025
sns.displot(tips.total_bill,kde=True)
```

[5]:
```
<seaborn.axisgrid.FacetGrid at 0x29396ae4bd0>
```



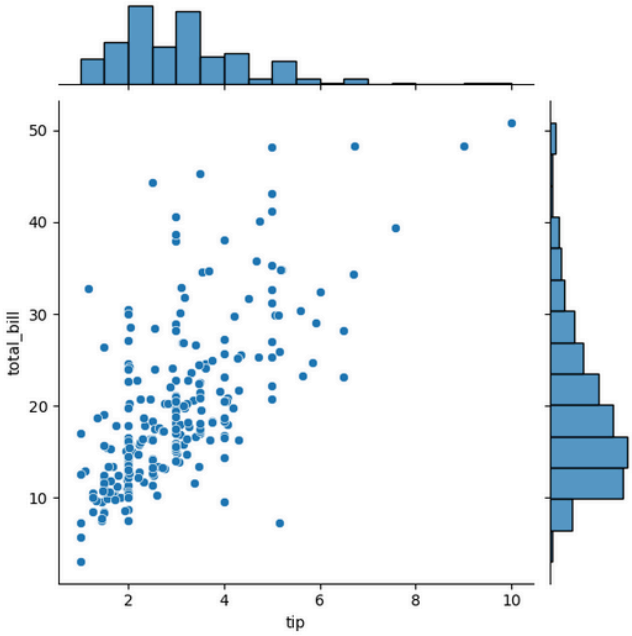
*[7]:
```
#M.Vishal
#240701598
#CSE
#26-08-2025
sns.displot(tips.total_bill,kde=False)
```

[7]: <seaborn.axisgrid.FacetGrid at 0x29396bf2a90>



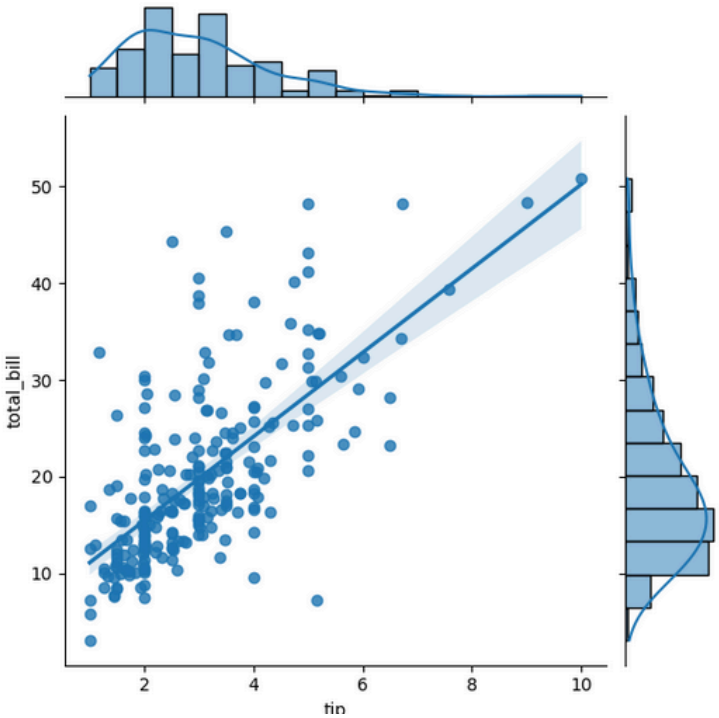
*[9]:
```
#M.Vishal
#240701598
#CSE
#26-08-2025
sns.jointplot(x=tips.tip,y=tips.total_bill)
```

[9]: <seaborn.axisgrid.JointGrid at 0x29396ae6410>



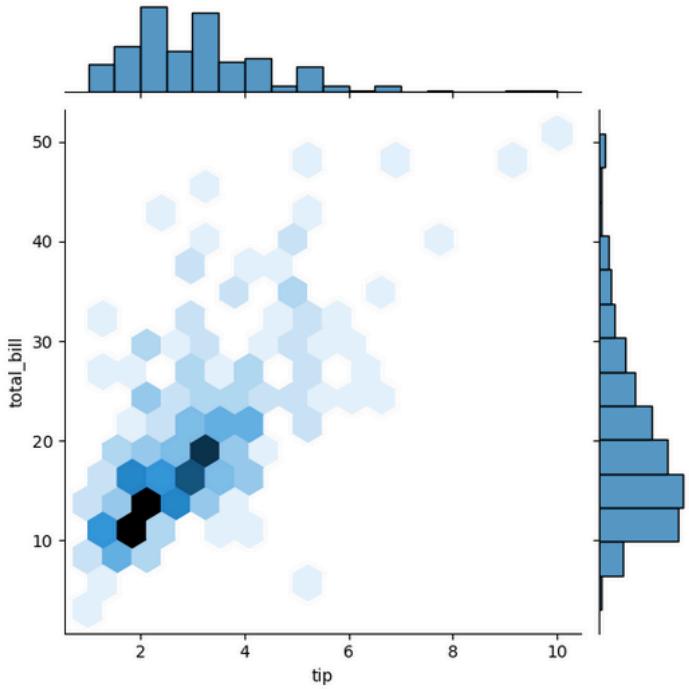
•[11]:
```
#M.Vishal
#240701598
#CSE
#26-08-2025
sns.jointplot(x=tips.tip,y=tips.total_bill,kind="reg")
```

[11]:
```
<seaborn.axisgrid.JointGrid at 0x29397a24090>
```



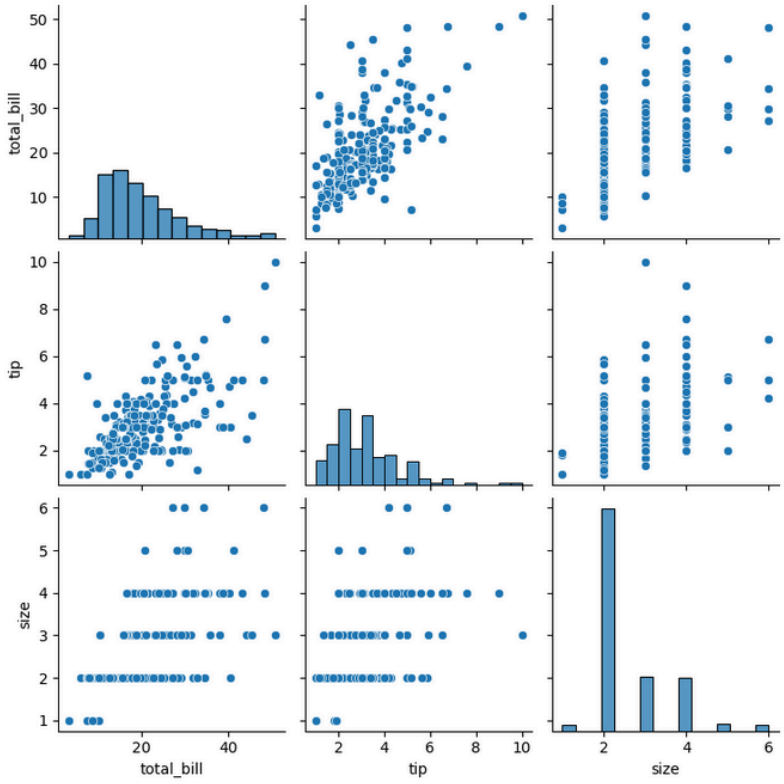
•[13]:
```
#M.Vishal
#240701598
#CSE
#26-08-2025
sns.jointplot(x=tips.tip,y=tips.total_bill,kind="hex")
```

[13]:
```
<seaborn.axisgrid.JointGrid at 0x29397cc32d0>
```



*[15]:
```
#M.Vishal
#240701598
#CSE
#26-08-2025
sns.pairplot(tips)
```

[15]:
```
<seaborn.axisgrid.PairGrid at 0x293977538d0>
```

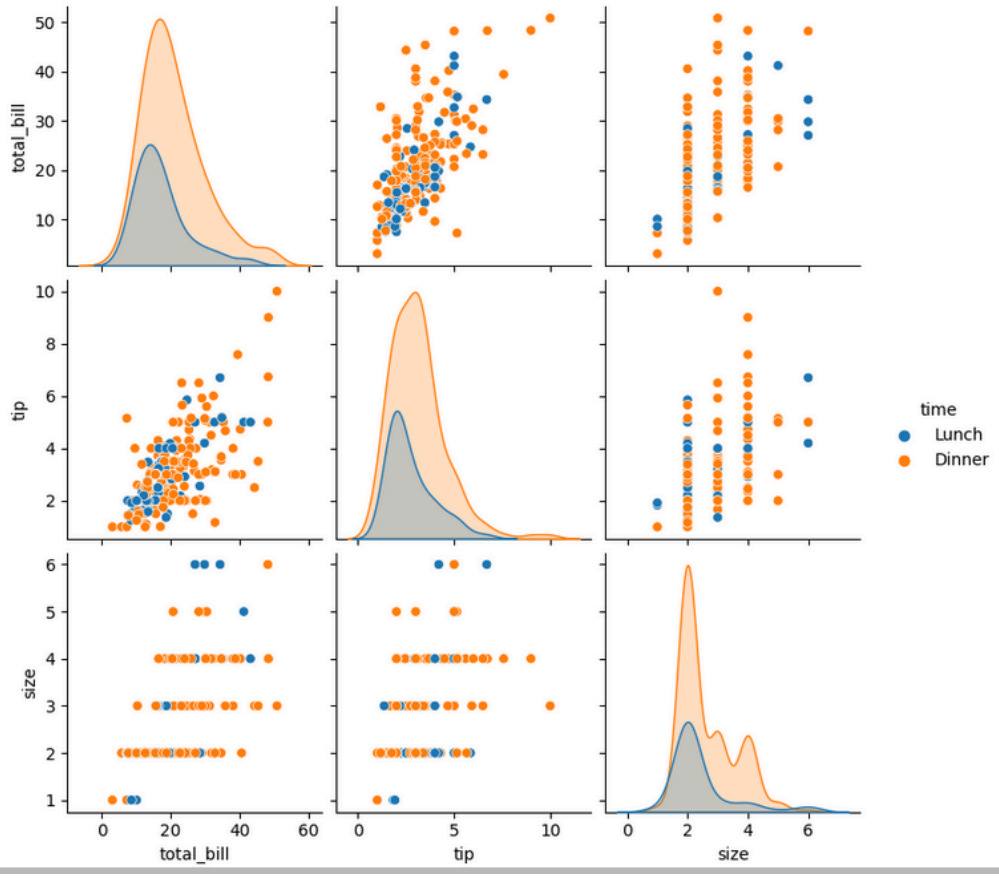


*[17]:
```
#M.Vishal
#240701598
#CSE
#26-08-2025
tips.time.value_counts()
```

[17]:
```
Dinner    176
Lunch      68
Name: time, dtype: int64
```

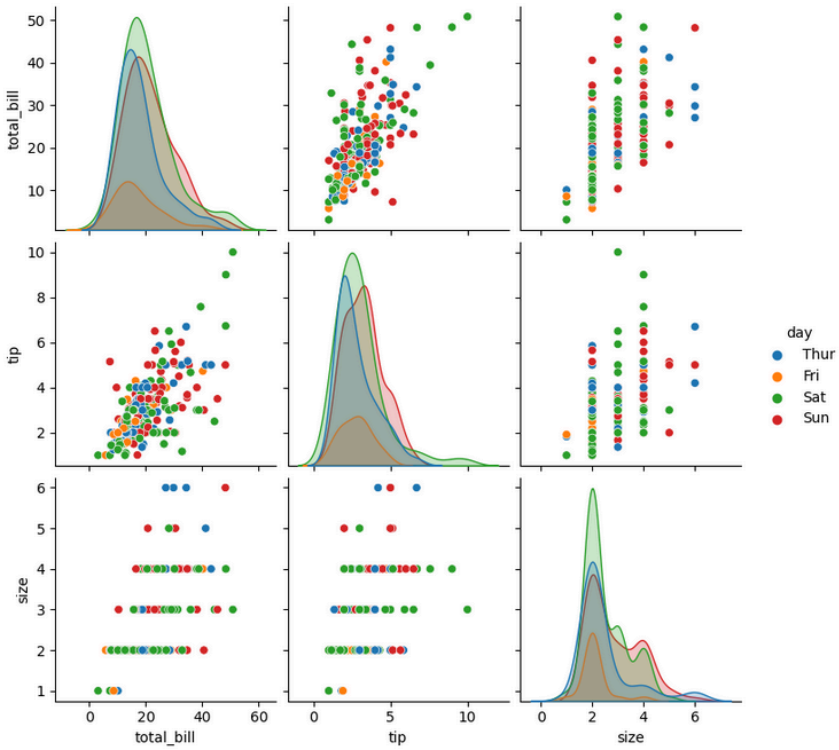
*[19]:
```
#M.Vishal
#240701598
#CSE
#26-08-2025
sns.pairplot(tips,hue='time')
```

[19]: <seaborn.axisgrid.PairGrid at 0x29398b6b090>



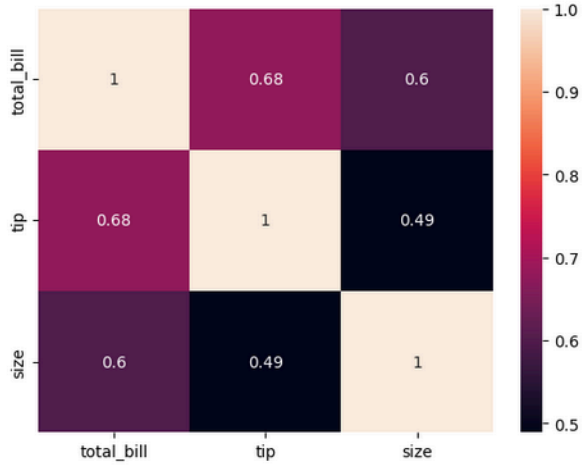
*[21]:
```
#M.Vishal
#240701598
#CSE
#26-08-2025
sns.pairplot(tips,hue='day')
```

[21]: <seaborn.axisgrid.PairGrid at 0x2939a43fb50>



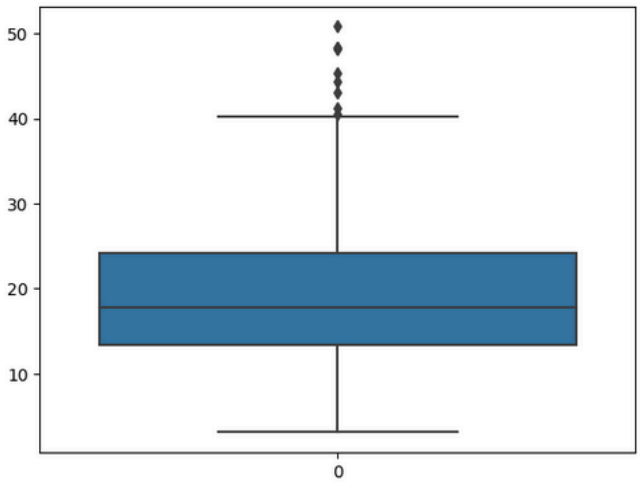
In [11]:
```
#M.Vishal
#240701598
#26-08-2025
sns.heatmap(tips.corr(numeric_only=True),annot=True)
```

Out[11]: <Axes: >



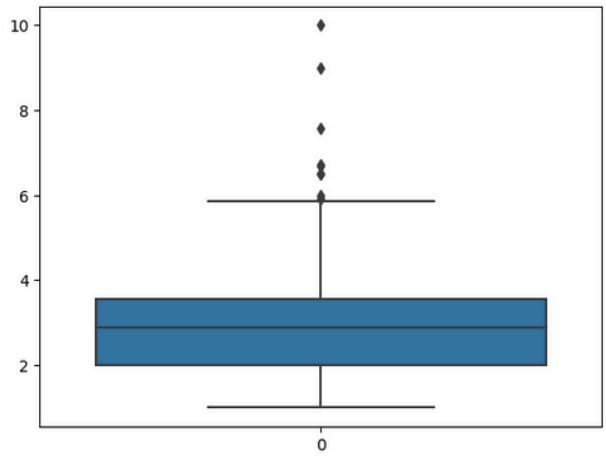
In [12]:
```
#M.Vishal
#240701598
#26-08-2025
sns.boxplot(tips.total_bill)
```

Out[12]: <Axes: >



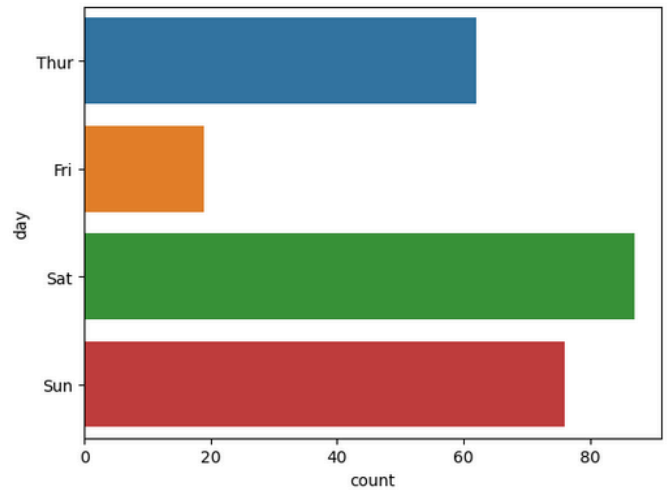
In [13]:
```
#M.Vishal
#240701598
#26-08-2025
sns.boxplot(tips.tip)
```

Out[13]: <Axes: >



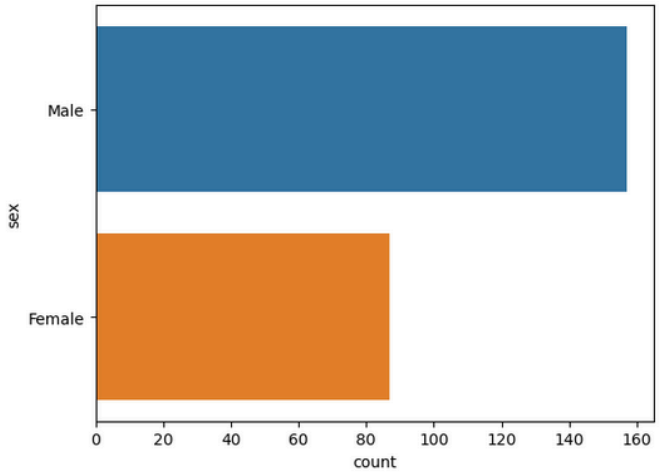
In [16]:
```
#M.Vishal
#240701598
#26-08-2025
sns.countplot(y='day', data=tips)
```

Out[16]: <Axes: xlabel='count', ylabel='day'>



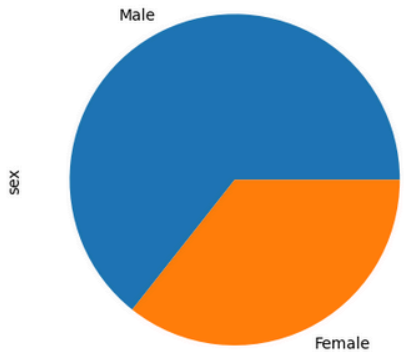
In [18]:
```
#M.Vishal
#240701598
#26-08-2025
sns.countplot(y='sex',data=tips)
```

Out[18]: <Axes: xlabel='count', ylabel='sex'>



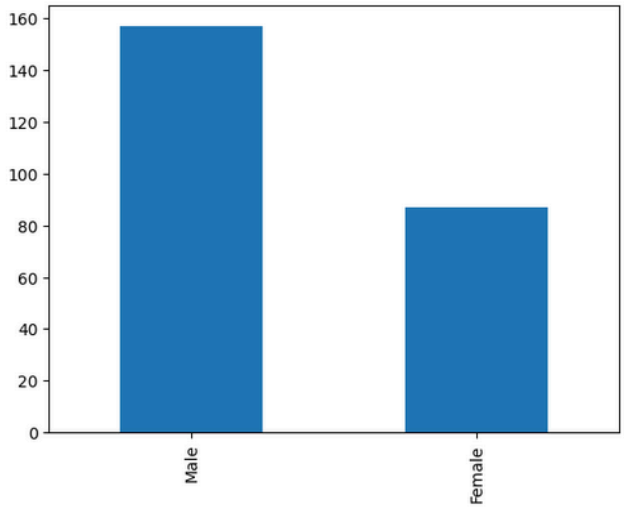
In [19]:
```
#M.Vishal
#240701598
#26-08-2025
tips.sex.value_counts().plot(kind='pie')
```

Out[19]: <Axes: ylabel='sex'>



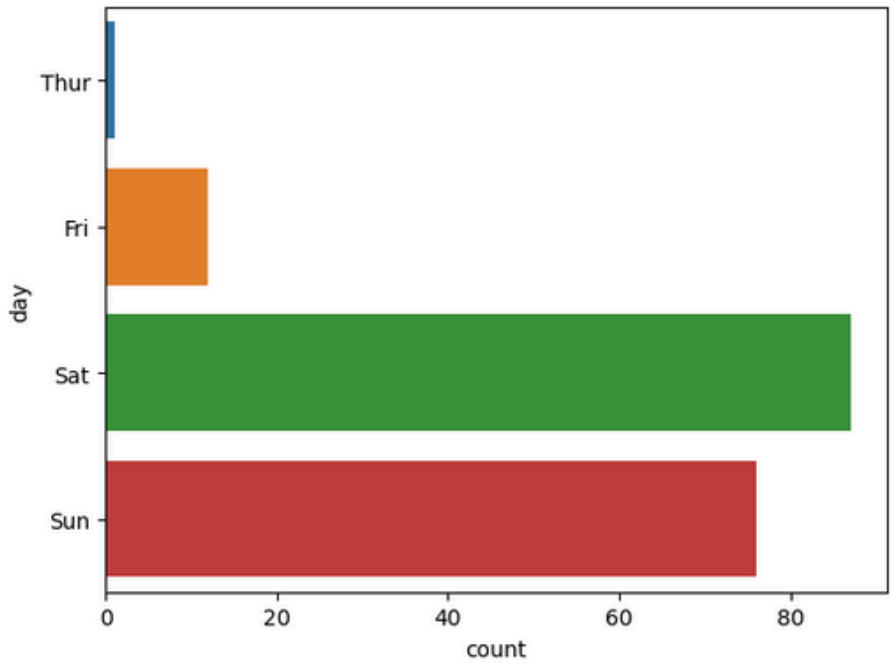
In [20]:
```
#M.Vishal
#240701598
#26-08-2025
tips.sex.value_counts().plot(kind='bar')
```

Out[20]: <Axes: >



In [23]:
```
#M.Vishal
#240701598
#26-08-2025
sns.countplot(y='day', data=tips[tips.time == 'Dinner'])
```

Out[23]: <Axes: xlabel='count', ylabel='day'>



In [1]:
```
import numpy as np
import pandas as pd
df=pd.read_csv('Salary_data.csv')
df
```

Out[1]:

	YearsExperience	Salary
0	1.1	39343
1	1.3	46205
2	1.5	37731
3	2.0	43525
4	2.2	39891
5	2.9	56642
6	3.0	60150
7	3.2	54445
8	3.2	64445
9	3.7	57189
10	3.9	63218
11	4.0	55794
12	4.0	56957
13	4.1	57081
14	4.5	61111
15	4.9	67938
16	5.1	66029
17	5.3	83088
18	5.9	81363
19	6.0	93940
20	6.8	91738
21	7.1	98273
22	7.9	101302
23	8.2	113812
24	8.7	109431
25	9.0	105582
26	9.5	116969
27	9.6	112635
28	10.3	122391
29	10.5	121872

In [1]:
```
import numpy as np
import pandas as pd
df=pd.read_csv('Salary_data.csv')
df
```

Out[1]:

	YearsExperience	Salary
0	1.1	39343

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30 entries, 0 to 29
Data columns (total 2 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   YearsExperience  30 non-null     float64
 1   Salary           30 non-null     int64  
dtypes: float64(1), int64(1)
memory usage: 608.0 bytes
```

In [3]:
```
df.dropna(inplace=True)
```

In [4]:
```
df
```

Out[4]:

In [6]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30 entries, 0 to 29
Data columns (total 2 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   YearsExperience  30 non-null     float64
 1   Salary           30 non-null     int64  
dtypes: float64(1), int64(1)
memory usage: 608.0 bytes
```

In [7]:
```
df.describe()
```

Out[7]:

	YearsExperience	Salary
count	30.000000	30.000000
mean	5.313333	76003.000000
std	2.837888	27414.429785
min	1.100000	37731.000000
25%	3.200000	56720.750000
50%	4.700000	65237.000000
75%	7.700000	100544.750000
max	10.500000	122391.000000

In [8]:
```
features=df.iloc[:,[0]].values
label=df.iloc[:,[1]].values
```

In [11]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(features,label,test_size=0.2,random_state=20)
```

In [12]:
```
from sklearn.linear_model import LinearRegression
model=LinearRegression()
model.fit(x_train,y_train)
```

Out[12]:
```
LinearRegression()
```

`

•[1]:
```
#M.vishal
#240701598
#7/10/2025
import numpy as np
import pandas as pd
df=pd.read_csv('Social_Network_Ads.csv')
df
```

[1]:

	User ID	Gender	Age	EstimatedSalary	Purchased
0	15624510	Male	19	19000	0
1	15810944	Male	35	20000	0
2	15668575	Female	26	43000	0
3	15603246	Female	27	57000	0
4	15804002	Male	19	76000	0
...	...	...	...	...	...
395	15691863	Female	46	41000	1
396	15706071	Male	51	23000	1
397	15654296	Female	50	20000	1
398	15755018	Male	36	33000	0
399	15594041	Female	49	36000	1

400 rows × 5 columns

•[2]:
```
#M.vishal
#240701598
#7/10/2025
df.head()
```

[2]:

	User ID	Gender	Age	EstimatedSalary	Purchased
0	15624510	Male	19	19000	0
1	15810944	Male	35	20000	0
2	15668575	Female	26	43000	0
3	15603246	Female	27	57000	0
4	15804002	Male	19	76000	0

•[3]:
```
#M.vishal
#240701598
#7/10/2025
features=df.iloc[:,[2,3]].values
label=df.iloc[:,4].values
features
```

[3]:
```
array([[    19,  19000],
       [    35,  20000],
       [    26,  43000],
       [    27,  57000],
       [    19,  76000],
       [    27,  58000],
       [    27,  84000],
       [    32, 150000],
       [    25,  33000],
       [    35,  65000],
       [    26,  80000],
       [    26,  52000],
       [    20,  86000],
       [    32,  18000],
       [    18,  82000],
       [    29,  80000],
       [    47,  25000],
       [    45,  26000],
       [    46,  28000],
       [    48,  29000],
       [    45,  22000],
       [    47,  49000],
       [    48,  41000],
       [    45,  22000],
       [    46,  23000],
       [    47,  20000],
       [    49,  28000],
       [    47,  30000],
```

[4]:
```
#M.vishal
#240701598
#7/10/2025
label
```

[4]:
```
array([0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1,
       0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0,
       1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0,
       1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1,
       0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1,
       1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1,
       0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0,
       1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1,
       0, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1,
       1, 1, 0, 1])
```

[5]:
```
#M.vishal
#240701598
#7/10/2025
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
```

[7]:
```
#M.vishal
#240701598
#7/10/2025
for i in range(1, 401):
    x_train, x_test, y_train, y_test = train_test_split(features, label, test_size=0.3, random_state=i)
    model=LogisticRegression()
    model.fit(x_train,y_train)
    train_score=model.score(x_train,y_train)
    test_score=model.score(x_test,y_test)
    if test_score>train_score:
        print("Test {} Train{} Random State {}".format(test_score,train_score,i))
```

```
Test 0.8833333333333333 Train0.8285714285714286 Random State 4
Test 0.8666666666666667 Train0.8464285714285714 Random State 5
Test 0.875 Train0.8464285714285714 Random State 6
Test 0.875 Train0.8392857142857143 Random State 7
Test 0.875 Train0.8464285714285714 Random State 10
Test 0.8583333333333333 Train0.8285714285714286 Random State 13
Test 0.8666666666666667 Train0.8535714285714285 Random State 14
Test 0.8833333333333333 Train0.8464285714285714 Random State 15
Test 0.8833333333333333 Train0.8464285714285714 Random State 16
Test 0.8583333333333333 Train0.8535714285714285 Random State 17
Test 0.8833333333333333 Train0.8285714285714286 Random State 18
```

[8]:
```
#M.vishal
#240701598
#7/10/2025
x_train, x_test, y_train, y_test = train_test_split(features, label, test_size=0.2,random_state=42)
finalModel = LogisticRegression()
finalModel.fit(x_train, y_train)
```

[8]:
LogisticRegression

▶ Parameters

[9]:
```
#M.vishal
#240701598
#7/10/2025
print(finalModel.score(x_train,y_train))
print(finalModel.score(x_test,y_test))
```

```
0.8375
0.8875
```

•[10]:
```
#M.vishal
#240701598
#7/10/2025
from sklearn.metrics import classification_report
print(classification_report(label,finalModel.predict(features)))
```

```
              precision    recall  f1-score   support

           0       0.85      0.93      0.89       257
           1       0.85      0.70      0.77       143

    accuracy                           0.85       400
   macro avg       0.85      0.81      0.83       400
weighted avg       0.85      0.85      0.84       400
```

[ ]:

In [1]:
```
#M.Vishal
#240701598
#10/7/2025



import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

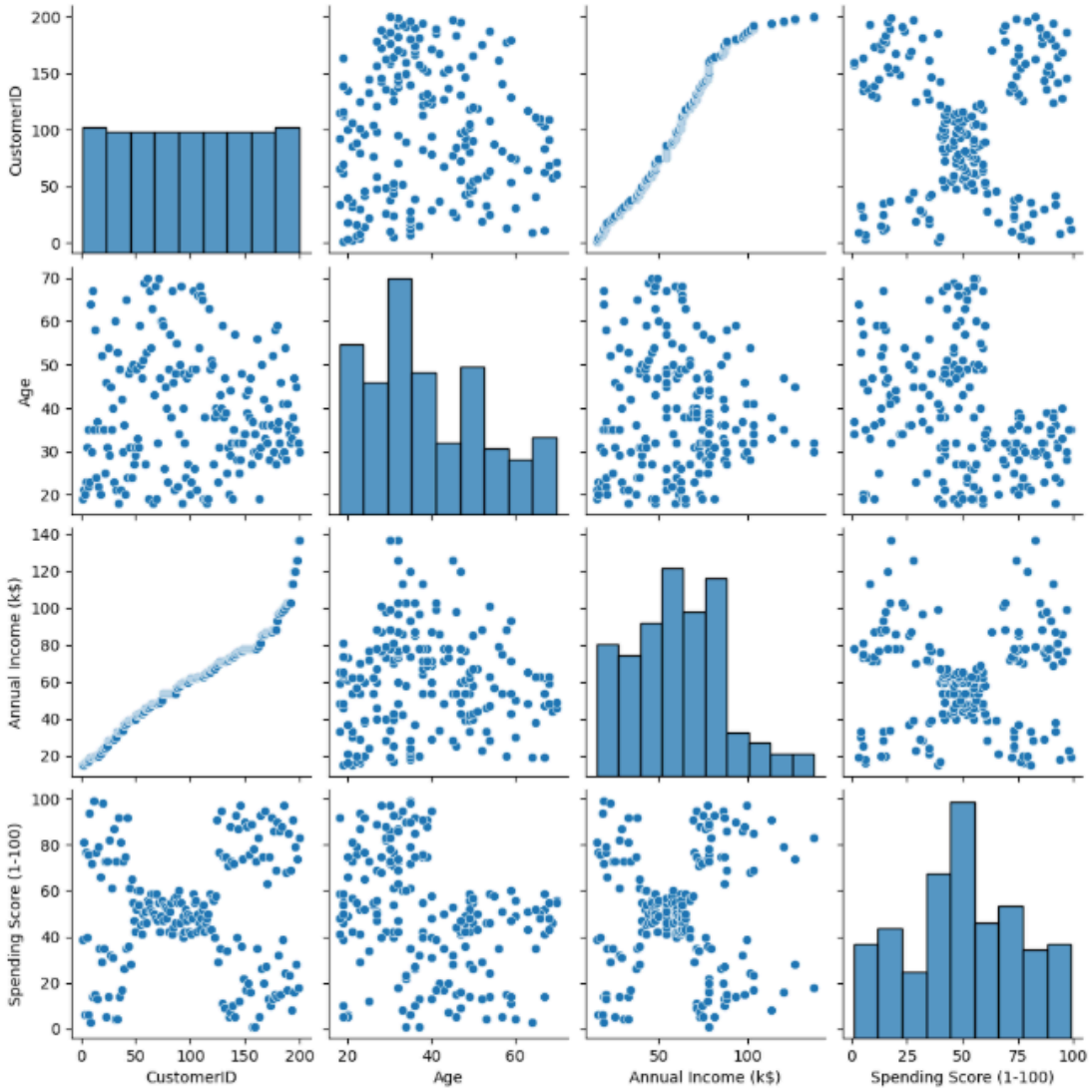
In [3]:
```
#M.Vishal
#240701598
#10/7/2025
df=pd.read_csv('Mall_Customers.csv')
```

In [4]:
```
#M.Vishal
#240701598
#10/7/2025
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 5 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   CustomerID              200 non-null    int64
 1   Gender                  200 non-null    object
 2   Age                     200 non-null    int64
 3   Annual Income (k$)      200 non-null    int64
 4   Spending Score (1-100)  200 non-null    int64
dtypes: int64(4), object(1)
memory usage: 7.9+ KB
```

In [5]:
```
#M.Vishal
#240701598
#10/7/2025
sns.pairplot(df)
```

Out[5]:
```
<seaborn.axisgrid.PairGrid at 0x2afaeb19ad0>
```



In [6]:
```
#M.Vishal
#240701598
#10/7/2025
features=df.iloc[:,[3,4]].values
```

In [7]:
```
#M.Vishal
#240701598
#10/7/2025
from sklearn.cluster import KMeans
model=KMeans(n_clusters=5)
model.fit(features)
KMeans(n_clusters=5)
```

```
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarning: The default value of `n_init` will c
hange from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarning: KMeans is known to have a memory leak
on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment variable OM
P_NUM_THREADS=1.
  warnings.warn(
```

Out[7]:

KMeans

KMeans(n_clusters=5)

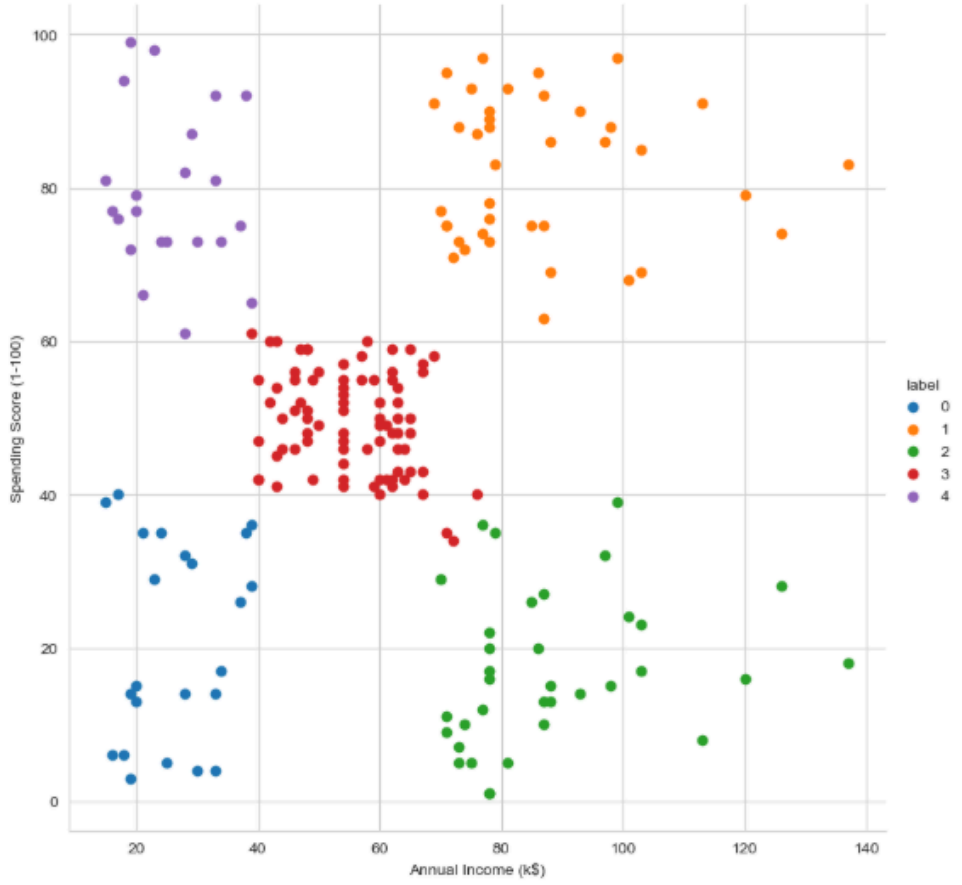
In [8]:
```
#M.Vishal
#240701598
#10/7/2025
Final=df.iloc[:,[3,4]]
Final['label']=model.predict(features)
Final.head()
```

```
C:\Users\hdc0422206\AppData\Local\Temp\ipykernel_16284\470183701.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-ve
rsus-a-copy
  Final['label']=model.predict(features)
```

Out[8]:

	Annual Income (k$)	Spending Score (1-100)	label
0	15	39	0
1	15	81	4
2	16	6	0
3	16	77	4
4	17	40	0

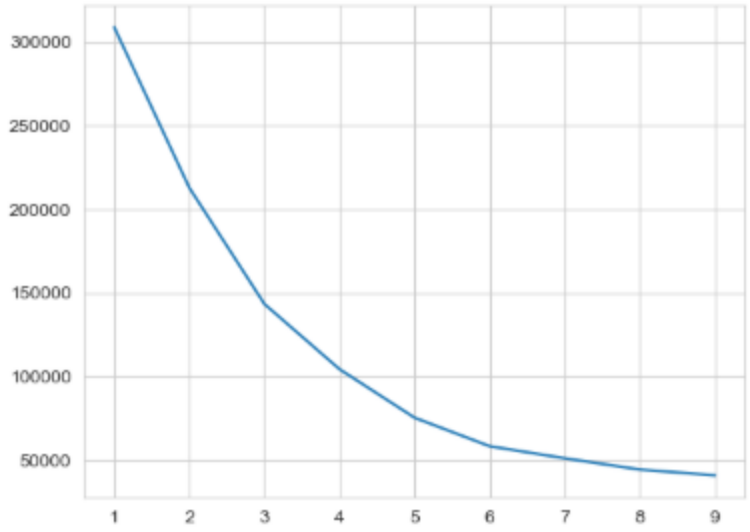
In [9]:
```
#M.Vishal
#240701598
#10/7/2025
sns.set_style("whitegrid")
sns.FacetGrid(Final,hue="label",height=8) \
.map(plt.scatter,"Annual Income (k$)", "Spending Score (1-100)") \
.add_legend();
plt.show()
```



In [11]:
```
#M.Vishal
#240701598
#10/7/2025
features_el=df.iloc[:,[2,3,4]].values
from sklearn.cluster import KMeans
wcss=[]
for i in range(1,10):
    model=KMeans(n_clusters=i)
    model.fit(features_el)
    wcss.append(model.inertia_)
plt.plot(range(1,10),wcss)
```

```
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarning: The default value of `n_init` will c
hange from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarning: KMeans is known to have a memory leak
on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment variable OM
P_NUM_THREADS=1.
  warnings.warn(
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarning: The default value of `n_init` will c
hange from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarning: KMeans is known to have a memory leak
on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment variable OM
P_NUM_THREADS=1.
  warnings.warn(
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarning: The default value of `n_init` will c
hange from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarning: KMeans is known to have a memory leak
on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment variable OM
P_NUM_THREADS=1.
  warnings.warn(
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarning: The default value of `n_init` will c
hange from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarning: KMeans is known to have a memory leak
on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment variable OM
P_NUM_THREADS=1.
  warnings.warn(
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarning: The default value of `n_init` will c
hange from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarning: KMeans is known to have a memory leak
on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment variable OM
P_NUM_THREADS=1.
  warnings.warn(
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarning: The default value of `n_init` will c
hange from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarning: KMeans is known to have a memory leak
on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment variable OM
P_NUM_THREADS=1.
  warnings.warn(
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarning: The default value of `n_init` will c
hange from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarning: KMeans is known to have a memory leak
on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment variable OM
P_NUM_THREADS=1.
```

Out[11]: [<matplotlib.lines.Line2D at 0x2afb104eb90>]



In [12]:
```
#M.vishal
#240701598
#10/7/2025
import numpy as np
import pandas as pd
```

In [14]:
```
#M.vishal
#240701598
#10/7/2025
df=pd.read_csv('Iris.csv')
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   sepal.length  150 non-null    float64
 1   sepal.width   150 non-null    float64
 2   petal.length  150 non-null    float64
 3   petal.width   150 non-null    float64
 4   variety       150 non-null    object
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```

In [15]:
```
#M.vishal
#240701598
#10/7/2025
df.variety.value_counts()
```

Out[15]:
```
Setosa        50
Versicolor    50
Virginica     50
Name: variety, dtype: int64
```

In [16]:
```
#M.vishal
#240701598
#10/7/2025
df.head()
```

Out[16]:

	sepal.length	sepal.width	petal.length	petal.width	variety
0	5.1	3.5	1.4	0.2	Setosa
1	4.9	3.0	1.4	0.2	Setosa
2	4.7	3.2	1.3	0.2	Setosa
3	4.6	3.1	1.5	0.2	Setosa
4	5.0	3.6	1.4	0.2	Setosa

In [17]:
```
#M.vishal
#240701598
#10/7/2025
features=df.iloc[:,:-1].values
label=df.iloc[:,4].values
```

In [18]:
```
#M.vishal
#240701598
#10/7/2025
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
```

In [19]:
```
#M.vishal
#240701598
#10/7/2025
xtrain,xtest,ytrain,ytest=train_test_split(features,label,test_size=.2,random_state=42)
model_KNN=KNeighborsClassifier(n_neighbors=5)
model_KNN.fit(xtrain,ytrain)
```

Out[19]:
```
KNeighborsClassifier
KNeighborsClassifier()
```

In [20]:
```
#M.vishal
#240701598
#10/7/2025
print(model_KNN.score(xtrain,ytrain))
print(model_KNN.score(xtest,ytest))
```

```
0.9666666666666667
1.0
```

In [21]:
```
#M.vishal
#240701598
#10/7/2025
from sklearn.metrics import confusion_matrix
confusion_matrix(label,model_KNN.predict(features))
```

Out[21]:
```
array([[50,  0,  0],
       [ 0, 47,  3],
       [ 0,  1, 49]], dtype=int64)
```

In [ ]:

In [22]:
```
#M.vishal
#240701598
#10/7/2025
from sklearn.metrics import classification_report
print(classification_report(label,model_KNN.predict(features)))
```

```
              precision    recall  f1-score   support

      Setosa       1.00      1.00      1.00        50
  Versicolor       0.98      0.94      0.96        50
   Virginica       0.94      0.98      0.96        50

    accuracy                           0.97       150
   macro avg       0.97      0.97      0.97       150
weighted avg       0.97      0.97      0.97       150
```

In [ ]:

[10]:

```python
# A sample of 10 students scored the following marks in an exam:
# [72, 68, 75, 70, 74, 69, 71, 73, 70, 72] We want to test whether the average mark = 70 (μ₀ = 70) at
# 5% significance level using python
#M.Vishal
#240701598
import numpy as np
from scipy import stats

marks = np.array([72, 68, 75, 70, 74, 69, 71, 73, 70, 72])

mu_0 = 70

t_stat, p_value = stats.ttest_1samp(marks, mu_0)
print(f"T-statistic: {t_stat:.3f}")
print(f"P-value: {p_value:.4f}")
alpha = 0.05
if p_value < alpha:
    print("Reject Null Hypothesis → Mean is significantly different from 70.")
else:
    print("Fail to Reject Null Hypothesis → No significant difference.")
```

```
T-statistic: 1.993
P-value: 0.0774
Fail to Reject Null Hypothesis → No significant difference.
```

```python
# Z-test
# A manufacturer claims that the average weight of packets is 50 g.A random sample of 36 packets has
# an average weight of 51.2 g with a known σ = 3 g.At a 5% significance
#M.Vishal
#240701598
import numpy as np
from math import sqrt
from scipy.stats import norm
# Given data
x_bar = 51.2   # sample mean
mu_0 = 50      # population mean
sigma = 3      # population standard deviation
n = 36         # sample size
# Calculate Z-statistic
z_stat = (x_bar - mu_0) / (sigma / sqrt(n))
# Two-tailed p-value
p_value = 2 * (1 - norm.cdf(abs(z_stat)))
print(f"Z-statistic: {z_stat:.3f}")
print(f"P-value: {p_value:.4f}")
alpha = 0.05
if p_value < alpha:
    print("Reject Null Hypothesis → Mean is significantly different from 50 g.")
else:
    print("Fail to Reject Null Hypothesis → No significant difference.")
```

```
Z-statistic: 2.400
P-value: 0.0164
Reject Null Hypothesis → Mean is significantly different from 50 g.
```

[15]:

```python
#M.Vishal
#240701598
import numpy as np
from scipy import stats

# Data
A = [20, 22, 23]
B = [19, 20, 18]
C = [25, 27, 26]

# Perform one-way ANOVA
f_stat, p_value = stats.f_oneway(A, B, C)

print(f"F-statistic: {f_stat:.3f}")
print(f"P-value: {p_value:.4f}")

alpha = 0.05
if p_value < alpha:
    print("Reject Null Hypothesis → Means are significantly different.")
else:
    print("Fail to Reject Null Hypothesis → No significant difference.")
```

```
F-statistic: 25.923
P-value: 0.0011
Reject Null Hypothesis → Means are significantly different.
```