RAJALAKSHMI ENGINEERING COLLEGE

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

Department of Computer Science and Engineering

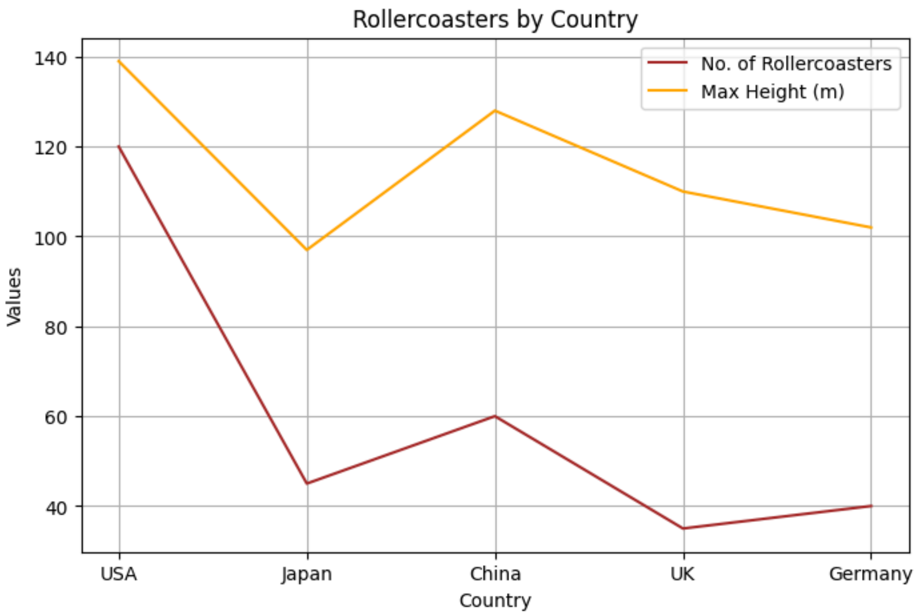
# CS23334 Fundamentals of Data Science Lab

III semester II Year (2023R)

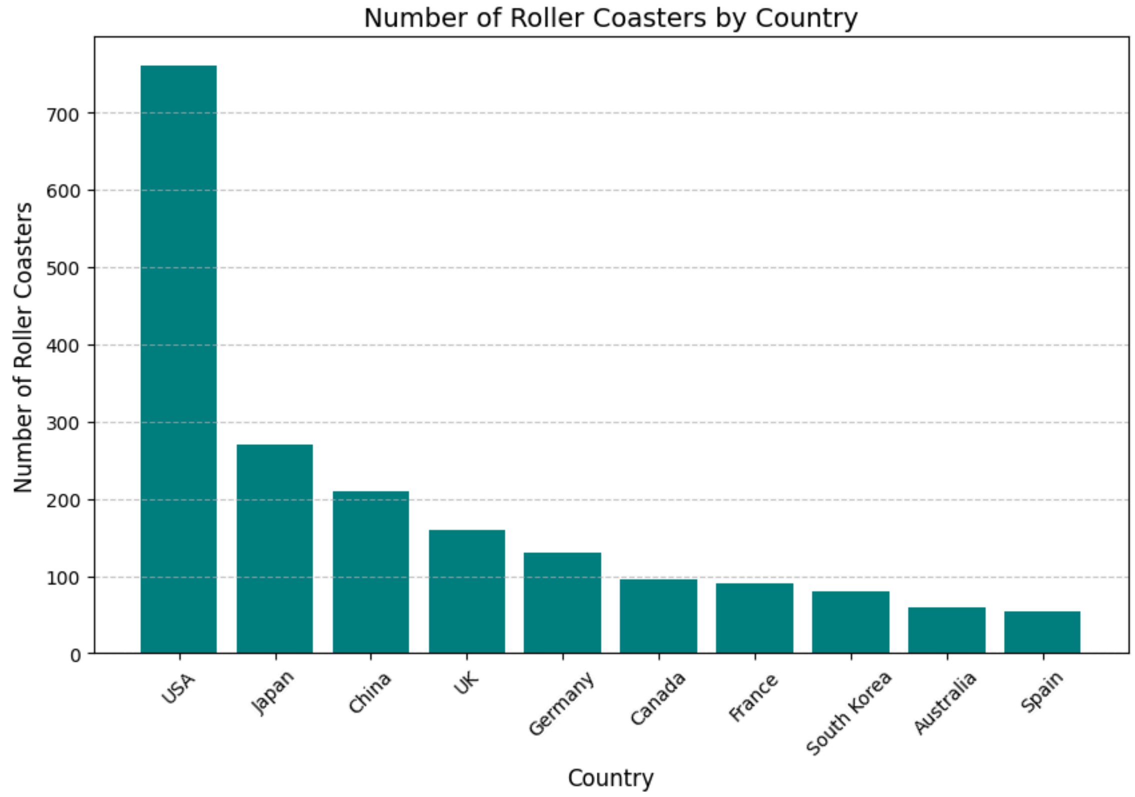
Name of the Student :

Register Number :

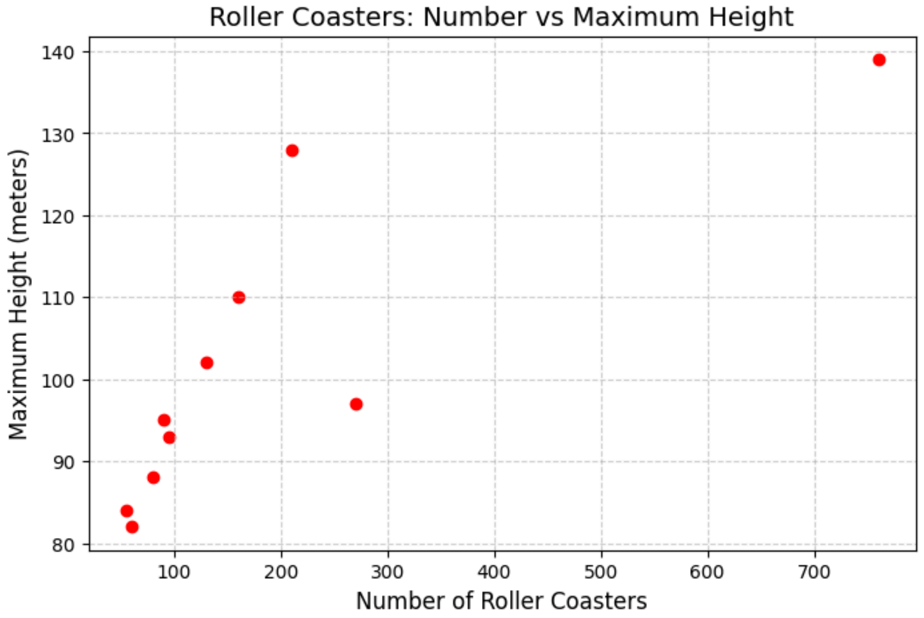
```
#reshma s
#240701426
#17/07/2025
#cse

import matplotlib.pyplot as plt
countries = ['USA', 'Japan', 'China', 'UK', 'Germany']
no_of_rollercoasters = [120, 45, 60, 35, 40]
max_height = [139, 97, 128, 110, 102]
plt.figure(figsize=(8, 5))

plt.plot(countries, no_of_rollercoasters, color='brown', label='No. of Rollercoasters')
plt.plot(countries, max_height, color='orange', label='Max Height (m)')

plt.title("Rollercoasters by Country")
plt.xlabel("Country")
plt.ylabel("Values")
plt.legend()
plt.grid(True)
plt.show()
```



```
#reshma s
#240701426
#17/07/2025
#cse

import matplotlib.pyplot as plt
countries = ['USA', 'Japan', 'China', 'UK', 'Germany', 'Canada', 'France', 'South Korea', 'Australia', 'Spain']

no_of_rollercoasters = [760, 270, 210, 160, 130, 95, 90, 80, 60, 55]

plt.figure(figsize=(10, 6))
plt.bar(countries, no_of_rollercoasters, color='teal')

plt.title("Number of Roller Coasters by Country", fontsize=14)
plt.xlabel("Country", fontsize=12)
plt.ylabel("Number of Roller Coasters", fontsize=12)
plt.xticks(rotation=45)
plt.grid(axis='y', linestyle='--', alpha=0.7)

plt.show()
```

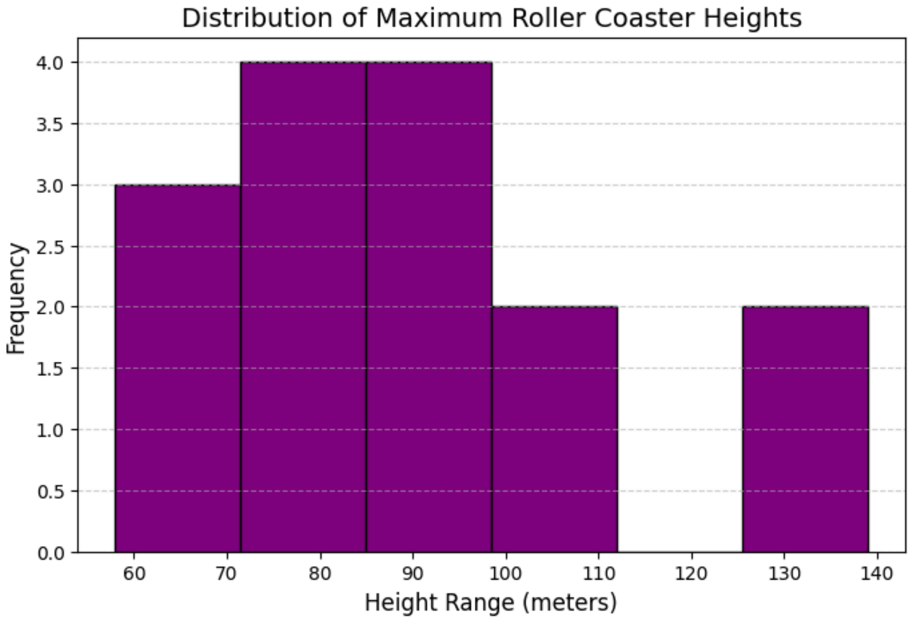


```
#reshma s
#240701426
#17/07/2025
#cse

import matplotlib.pyplot as plt
countries = ['USA', 'Japan', 'China', 'UK', 'Germany', 'Canada', 'France', 'South Korea', 'Australia', 'Spain']

no_of_rollercoasters = [760, 270, 210, 160, 130, 95, 90, 80, 60, 55]
max_height = [139, 97, 128, 110, 102, 93, 95, 88, 82, 84]

plt.figure(figsize=(8, 5))
plt.scatter(no_of_rollercoasters, max_height, color='red')

plt.title("Roller Coasters: Number vs Maximum Height", fontsize=14)
plt.xlabel("Number of Roller Coasters", fontsize=12)
plt.ylabel("Maximum Height (meters)", fontsize=12)
plt.grid(True, linestyle='--', alpha=0.6)

plt.show()
```



```
#reshma s
```

```
#240701426
#17/07/2025
#cse


import matplotlib.pyplot as plt
max_heights = [139, 97, 128, 110, 102, 93, 95, 88, 82, 84, 75, 69, 72, 65, 58]

plt.figure(figsize=(8, 5))
plt.hist(max_heights, bins=6, color='purple', edgecolor='black')

plt.title("Distribution of Maximum Roller Coaster Heights", fontsize=14)
plt.xlabel("Height Range (meters)", fontsize=12)
plt.ylabel("Frequency", fontsize=12)
plt.grid(axis='y', linestyle='--', alpha=0.6)

plt.show()
```



```
Start coding or generate with AI.
```

```
#reshma s
#240701426
#03/08/2025
#cse
from google.colab import files
uploaded = files.upload()

import numpy as np
import pandas as pd

df = pd.read_csv("pre_process_datasample.csv")
print(df)
print("\n")
print(df.info())
print("\n")

df['Country'] = df['Country'].fillna(df['Country'].mode()[0])
df['Age'] = df['Age'].fillna(df['Age'].median())
df['Salary'] = df['Salary'].fillna(round(df['Salary'].mean()))

print(df)
print("\n")
print(pd.get_dummies(df.Country))
print("\n")
updated_dataset = pd.concat([pd.get_dummies(df['Country']), df[['Age', 'Salary', 'Purchased']]], axis=1)
updated_dataset['Purchased'] = updated_dataset['Purchased'].replace(['No', 'Yes'], [0, 1])

print(updated_dataset)
print("\n")
updated_dataset.info()
```

Choose Files pre_proces…asample.csv

**pre_process_datasample.csv**(text/csv) - 226 bytes, last modified: 11/1/2025 - 100% done

```
Saving pre_process_datasample.csv to pre_process_datasample (4).csv
   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


<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
None


   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


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

```
#reshma s
#240701426
#07/08/2025
#cse
from google.colab import files
uploaded = files.upload()

import numpy as np
import pandas as pd

df = pd.read_csv("Hotel_Dataset.csv")
print(df, "\n")
print(df.duplicated(), "\n")
print(df.info(), "\n")
df = df.drop_duplicates()
print(df, "\n")
print("Length after removing duplicates:", len(df), "\n")
df = df.reset_index(drop=True)
print(df, "\n")
df = df.drop(columns=['Age_Group.1'])
print(df, "\n")
df.loc[df['CustomerID'] < 0, 'CustomerID'] = np.nan
df.loc[df['Bill'] < 0, 'Bill'] = np.nan
df.loc[df['EstimatedSalary'] < 0, 'EstimatedSalary'] = np.nan
print(df, "\n")
df.loc[(df['NoOfPax'] < 1) | (df['NoOfPax'] > 20), 'NoOfPax'] = np.nan
print(df, "\n")
print("Unique Age Groups:", df['Age_Group'].unique(), "\n")
print("Unique Hotels:", df['Hotel'].unique(), "\n")
df['Hotel'] = df['Hotel'].replace(['Ibys'], 'Ibis')
print("Before FoodPreference fix:", df['FoodPreference'].unique(), "\n")
df['FoodPreference'] = df['FoodPreference'].replace(['Vegetarian', 'veg'], 'Veg')
df['FoodPreference'] = df['FoodPreference'].replace(['non-Veg'], 'Non-Veg')
df['EstimatedSalary'] = df['EstimatedSalary'].fillna(round(df['EstimatedSalary'].mean()))
df['NoOfPax'] = df['NoOfPax'].fillna(round(df['NoOfPax'].median()))
df['Rating(1-5)'] = df['Rating(1-5)'].fillna(round(df['Rating(1-5)'].median()))
df['Bill'] = df['Bill'].fillna(round(df['Bill'].mean()))

print(df)
```

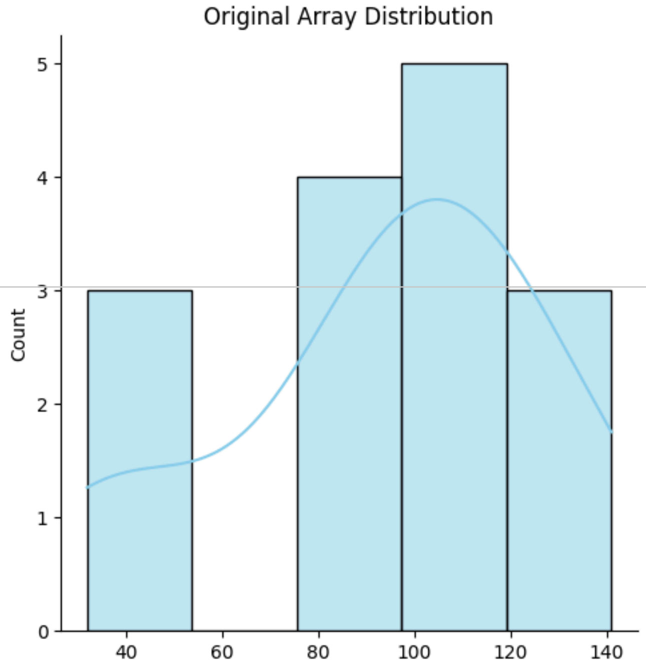
Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving Hotel_Dataset.csv to Hotel_Dataset (1).csv
    CustomerID Age_Group  Rating(1-5)      Hotel FoodPreference  Bill  \
0            1     20-25            4       Ibis            veg  1300   
1            2     30-35            5  LemonTree        Non-Veg  2000   
2            3     25-30            6     RedFox            Veg  1322   
3            4     20-25           -1  LemonTree            Veg  1234   
4            5       35+            3       Ibis     Vegetarian   989   
5            6       35+            3       Ibys        Non-Veg  1909   
6            7       35+            4     RedFox     Vegetarian  1000   
7            8     20-25            7  LemonTree            Veg  2999   
8            9     25-30            2       Ibis        Non-Veg  3456   
9            9     25-30            2       Ibis        Non-Veg  3456   
10          10     30-35            5     RedFox        non-Veg -6755   

    NoOfPax  EstimatedSalary Age_Group.1  
0         2            40000       20-25  
1         3            59000       30-35  
2         2            30000       25-30  
3         2           120000       20-25  
4         2            45000         35+  
5         2           122220         35+  
6        -1            21122         35+  
7       -10           345673       20-25  
8         3           -99999       25-30  
9         3           -99999       25-30  
10        4            87777       30-35  

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

<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
None

    CustomerID Age_Group  Rating(1-5)      Hotel FoodPreference  Bill  \
0            1     20-25            4       Ibis            veg  1300   
1            2     30-35            5  LemonTree        Non-Veg  2000   
2            3     25-30            6     RedFox            Veg  1322   
3            4     20-25           -1  LemonTree            Veg  1234   
4            5       35+            3       Ibis     Vegetarian   989   
5            6       35+            3       Ibys        Non-Veg  1909   
6            7       35+            4     RedFox     Vegetarian  1000   
7            8     20-25            7  LemonTree            Veg  2999   
8            9     25-30            2       Ibis        Non-Veg  3456   
10          10     30-35            5     RedFox        non-Veg -6755   

    NoOfPax  EstimatedSalary Age_Group.1  
0         2            40000       20-25  
1         3            59000       30-35  
2         2            30000       25-30  
3         2           120000       20-25  
4         2            45000         35+  
5         2           122220         35+  
6        -1            21122         35+  
7       -10           345673       20-25  
8         3           -99999       25-30  
10        4            87777       30-35  

Length after removing duplicates: 10

   CustomerID Age_Group  Rating(1-5)      Hotel FoodPreference  Bill  NoOfPax  \
0           1     20-25            4       Ibis            veg  1300        2   
1           2     30-35            5  LemonTree        Non-Veg  2000        3   
2           3     25-30            6     RedFox            Veg  1322        2   
```

```
3          4     20-25           -1  LemonTree            Veg  1234        2
4          5       35+            3       Ibis     Vegetarian   989        2
5          6       35+            3       Ibys        Non-Veg  1909        2
6          7       35+            4     RedFox     Vegetarian  1000       -1
7          8     20-25            7  LemonTree            Veg  2999      -10
8          9     25-30            2       Ibis        Non-Veg  3456        3
9         10     30-35            5     RedFox        non-Veg -6755        4

   EstimatedSalary Age_Group.1
0            40000       20-25
1            59000       30-35
2            30000       25-30
```

```
#reshma s
#240701426
#03/08/2025
#cse

import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

array = np.random.randint(5, 150, 15)
print("Array:", array)

print("\nMean:", array.mean())
print("\n25th percentile:", np.percentile(array, 25))
print("\n50th percentile (median):", np.percentile(array, 50))
print("\n75th percentile:", np.percentile(array, 75))
print("\n100th percentile (max):", np.percentile(array, 100))

def outDetection(array):
  array_sorted = np.sort(array)
  Q1, Q3 = np.percentile(array_sorted, [25, 75])
  IQR = Q3 - Q1
  lr = Q1 - (1.5 * IQR)
  ur = Q3 + (1.5 * IQR)
  return lr, ur
lr, ur = outDetection(array)
print("\nLower Range (LR):", lr)
print("\nUpper Range (UR):", ur)

plt.figure(figsize=(6,4))
sns.displot(array, kde=True, color='skyblue')
plt.title("Original Array Distribution")
plt.show()

new_array = array[(array > lr) & (array < ur)]
print("\nFiltered Array (no outliers):", new_array)

plt.figure(figsize=(6,4))
sns.distplot(new_array, kde=True, color='brown')
plt.title("Filtered Array Distribution")
plt.show()

lr1, ur1 = outDetection(new_array)
print("\nNew Lower Range:", lr1)
print("New Upper Range:", ur1)

final_array = new_array[(new_array > lr1) & (new_array < ur1)]
print("\nFinal Array:", final_array)

plt.figure(figsize=(6,4))
sns.distplot(final_array, kde=True, color='green')
plt.title("Final Array Distribution")
plt.show()
```

```
Array: [141  89  32 108 123  78  82 102  35 113  95  48 111 134 106]

Mean: 93.13333333333334

25th percentile: 80.0

50th percentile (median): 102.0

75th percentile: 112.0

100th percentile (max): 141.0

Lower Range (LR): 32.0

Upper Range (UR): 160.0
<Figure size 600x400 with 0 Axes>
```



```
Filtered Array (no outliers): [141  89 108 123  78  82 102  35 113  95  48 111 134 106]
/tmp/ipython-input-3007436746.py:34: UserWarning:

`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).
```

```
#reshma s
#240701426
#03/08/2025
#cse

from google.colab import files
uploaded = files.upload()

import numpy as np
import pandas as pd
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import LabelEncoder, StandardScaler, MinMaxScaler

df = pd.read_csv('pre_process_datasample.csv')
print("\nOriginal Dataset:")
display(df.head())

cat_imputer = SimpleImputer(strategy='most_frequent')
num_imputer = SimpleImputer(strategy='mean')

df[['Country']] = cat_imputer.fit_transform(df[['Country']])
df[['Age']] = num_imputer.fit_transform(df[['Age']])
df[['Salary']] = num_imputer.fit_transform(df[['Salary']])

print("\n")
display(df.head())

le = LabelEncoder()
df['Country'] = le.fit_transform(df['Country'])
print("\n")
display(df.head())

x1 = df.iloc[:, 0:1].values
x2 = df.iloc[:, 1:3].values
final_set = np.concatenate((x1, x2), axis=1)
print("\n")
print(final_set)

sc = StandardScaler()
feat_standard_scaler = sc.fit_transform(final_set)
print("\n")
print(feat_standard_scaler)

mms = MinMaxScaler(feature_range=(0,1))
feat_minmax_scaler = mms.fit_transform(final_set)
print("\n")
print(feat_minmax_scaler)
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving pre_process_datasample.csv to pre_process_datasample (1).csv

Original Dataset:
```

| | 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 |

| | Country | Age | Salary | Purchased |
|---|---|---|---|---|
| **0** | France | 44.0 | 72000.000000 | No |
| **1** | Spain | 27.0 | 48000.000000 | Yes |
| **2** | Germany | 30.0 | 54000.000000 | No |
| **3** | Spain | 38.0 | 61000.000000 | No |
| **4** | Germany | 40.0 | 63777.777778 | Yes |

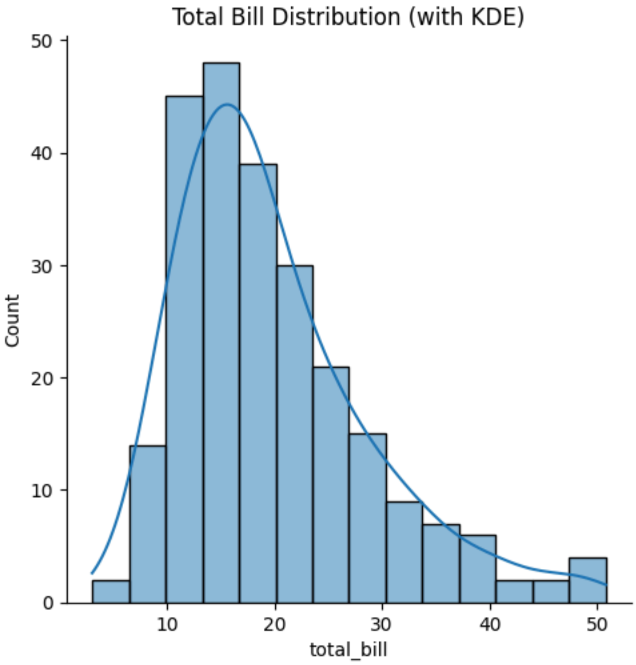
| | Country | Age | Salary | Purchased |
|---|---|---|---|---|
| **0** | 0 | 44.0 | 72000.000000 | No |
| **1** | 2 | 27.0 | 48000.000000 | Yes |
| **2** | 1 | 30.0 | 54000.000000 | No |
| **3** | 2 | 38.0 | 61000.000000 | No |
| **4** | 1 | 40.0 | 63777.777778 | Yes |

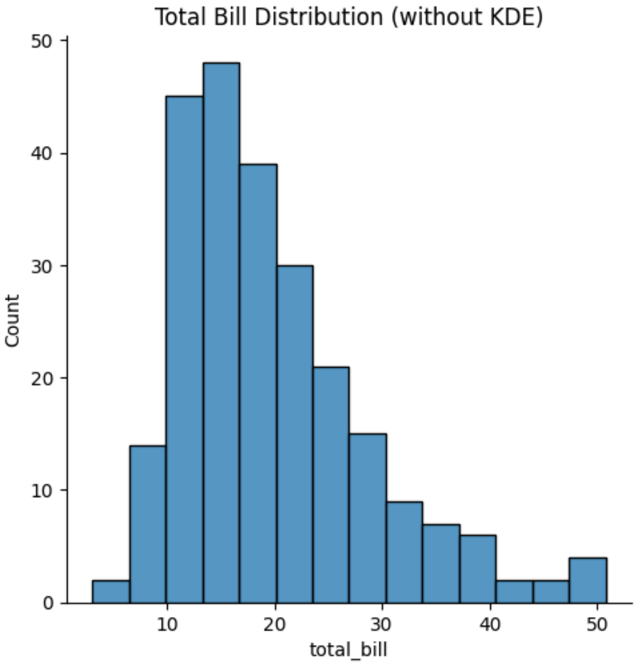
```
[[0.00000000e+00 4.40000000e+01 7.20000000e+04]
 [2.00000000e+00 2.70000000e+01 4.80000000e+04]
 [1.00000000e+00 3.00000000e+01 5.40000000e+04]
```

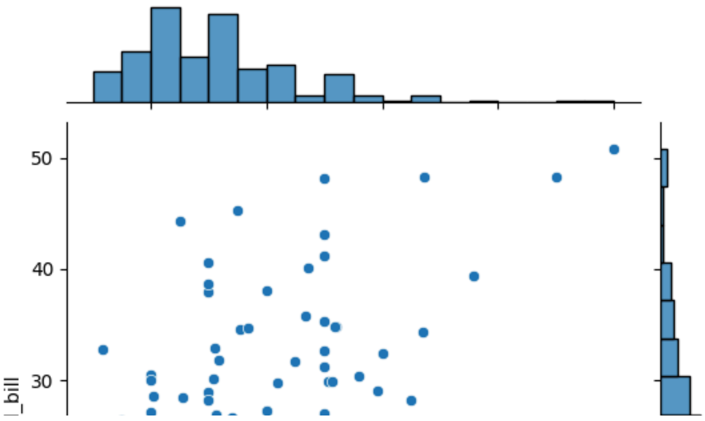
```
#reshma s
#240701426
#28/08/2025
#cse
import seaborn as sns
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline

tips = sns.load_dataset('tips')
print("\nDataset Preview:")
display(tips.head())

sns.displot(data=tips, x='total_bill', kde=True)
plt.title('Total Bill Distribution (with KDE)')
plt.show()

sns.displot(data=tips, x='total_bill', kde=False)
plt.title('Total Bill Distribution (without KDE)')
plt.show()

sns.jointplot(data=tips, x='tip', y='total_bill')
sns.jointplot(data=tips, x='tip', y='total_bill', kind="reg")
sns.jointplot(data=tips, x='tip', y='total_bill', kind="hex")

sns.pairplot(tips)
plt.show()

print("\nTime value counts:")
print(tips['time'].value_counts())

sns.pairplot(tips, hue='time')
sns.pairplot(tips, hue='day')

plt.figure(figsize=(8,5))
sns.heatmap(tips.corr(numeric_only=True), annot=True, cmap="YlGnBu")
plt.title("Correlation Heatmap")
plt.show()

sns.boxplot(data=tips, x='total_bill')
plt.title("Boxplot of Total Bill")
plt.show()

sns.boxplot(data=tips, x='tip')
plt.title("Boxplot of Tip")
plt.show()

sns.countplot(data=tips, x='day')
plt.title("Count of Records by Day")
plt.show()

sns.countplot(data=tips, x='sex')
plt.title("Count of Records by Gender")
plt.show()

tips['sex'].value_counts().plot(kind='pie', autopct='%1.1f%%', startangle=90)
plt.title("Gender Distribution (Pie)")
plt.ylabel("")
plt.show()

tips['sex'].value_counts().plot(kind='bar', color=['skyblue', 'salmon'])
plt.title("Gender Distribution (Bar)")
plt.xlabel("Sex")
plt.ylabel("Count")
plt.show()

sns.countplot(data=tips[tips['time'] == 'Dinner'], x='day')
plt.title("Dinner-Time Frequency by Day")
plt.show()
```

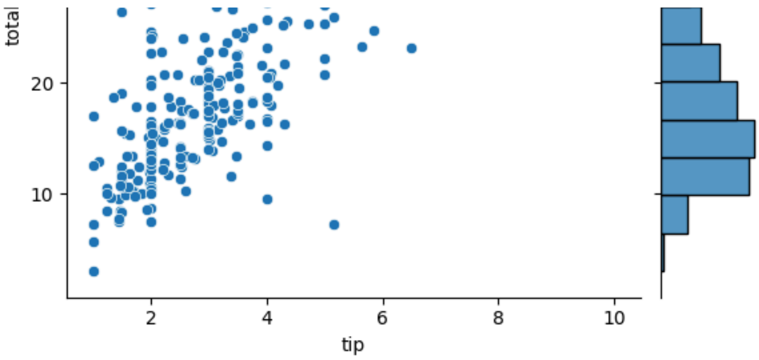
```
Dataset Preview:
```

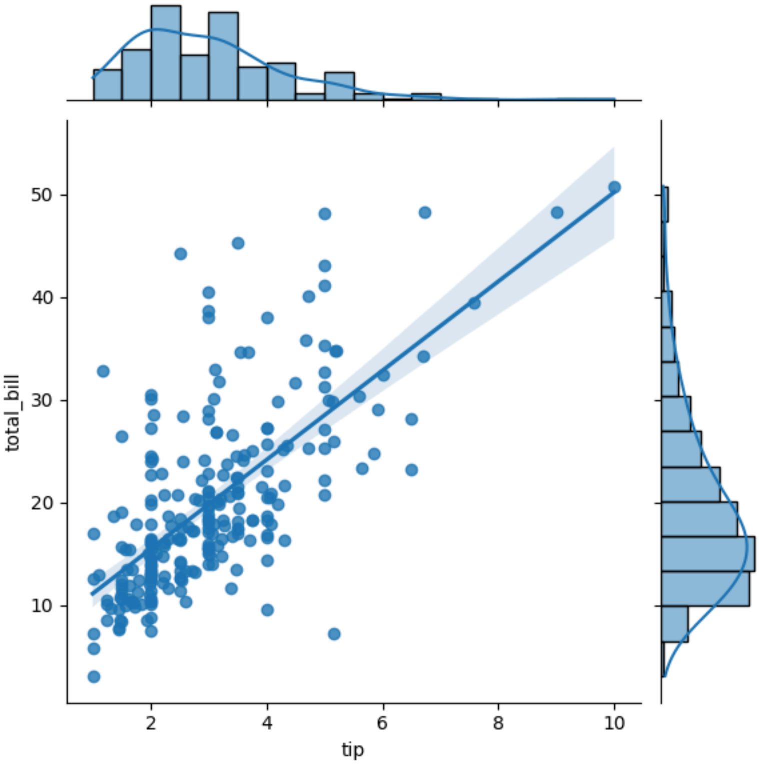
| | total_bill | tip | sex | smoker | day | time | size |
|---|---|---|---|---|---|---|---|
| **0** | 16.99 | 1.01 | Female | No | Sun | Dinner | 2 |
| **1** | 10.34 | 1.66 | Male | No | Sun | Dinner | 3 |
| **2** | 21.01 | 3.50 | Male | No | Sun | Dinner | 3 |
| **3** | 23.68 | 3.31 | Male | No | Sun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

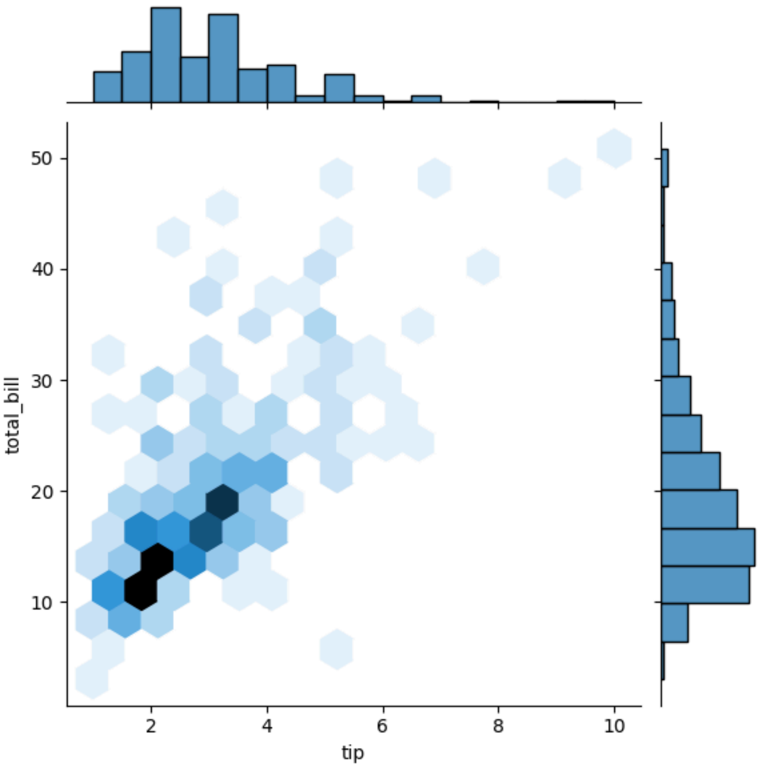


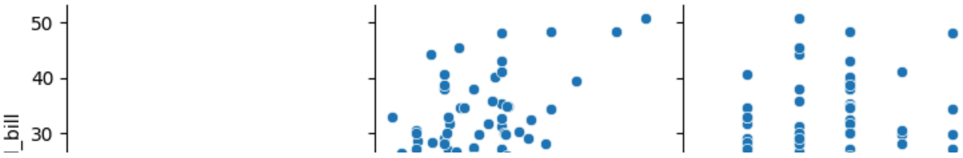


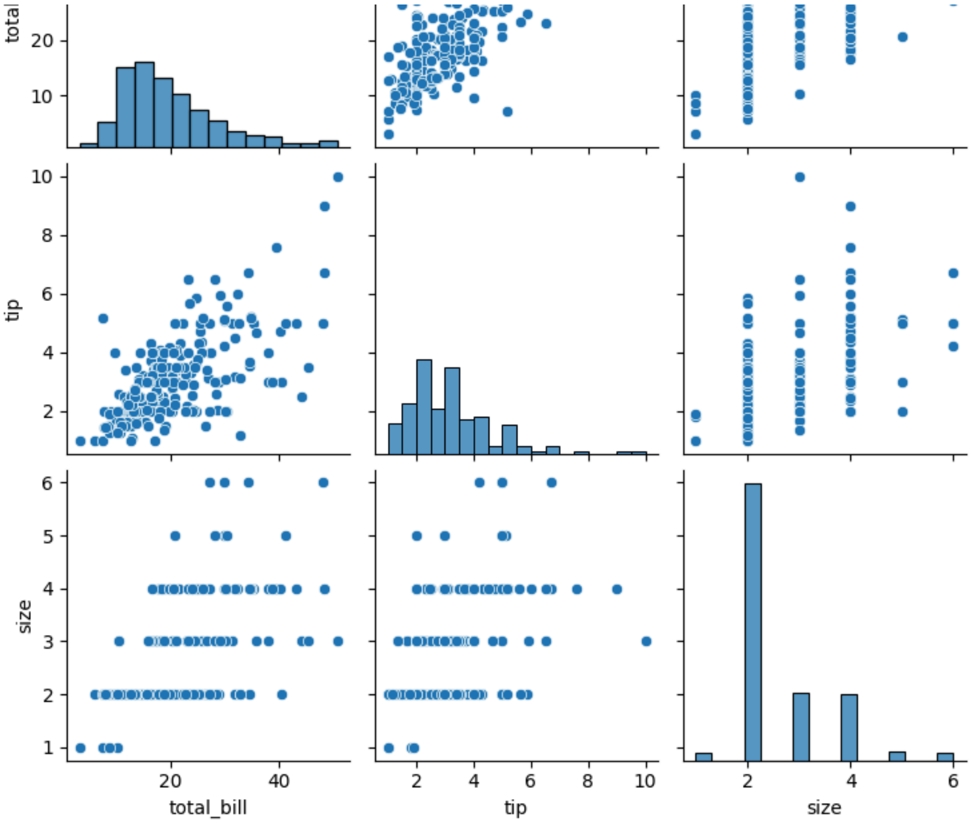



total
20
10
2
4
6
8
10
tip

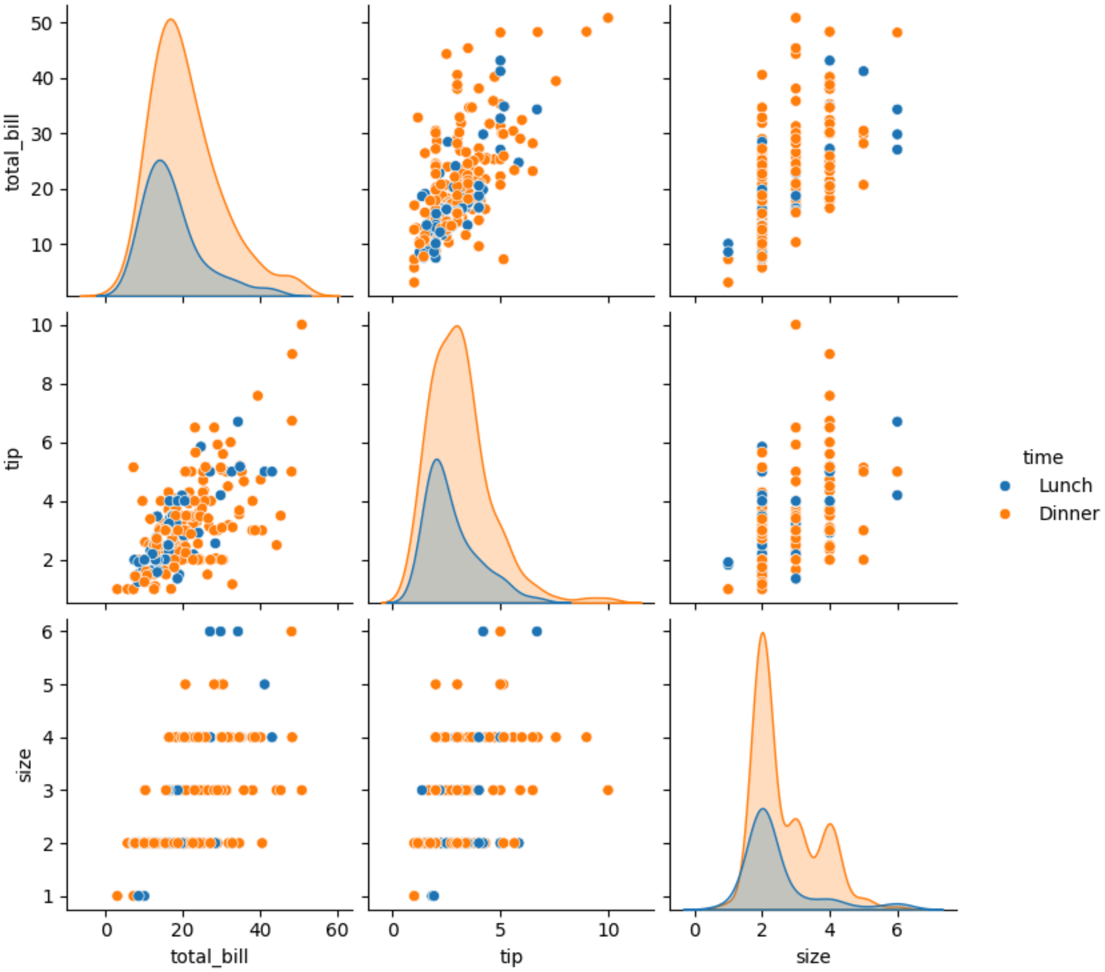

total_bill
50
40
30
20
10
2
4
6
8
10
tip

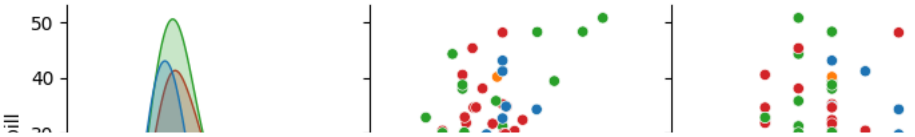

total_bill
50
40
30
20
10
2
4
6
8
10
tip

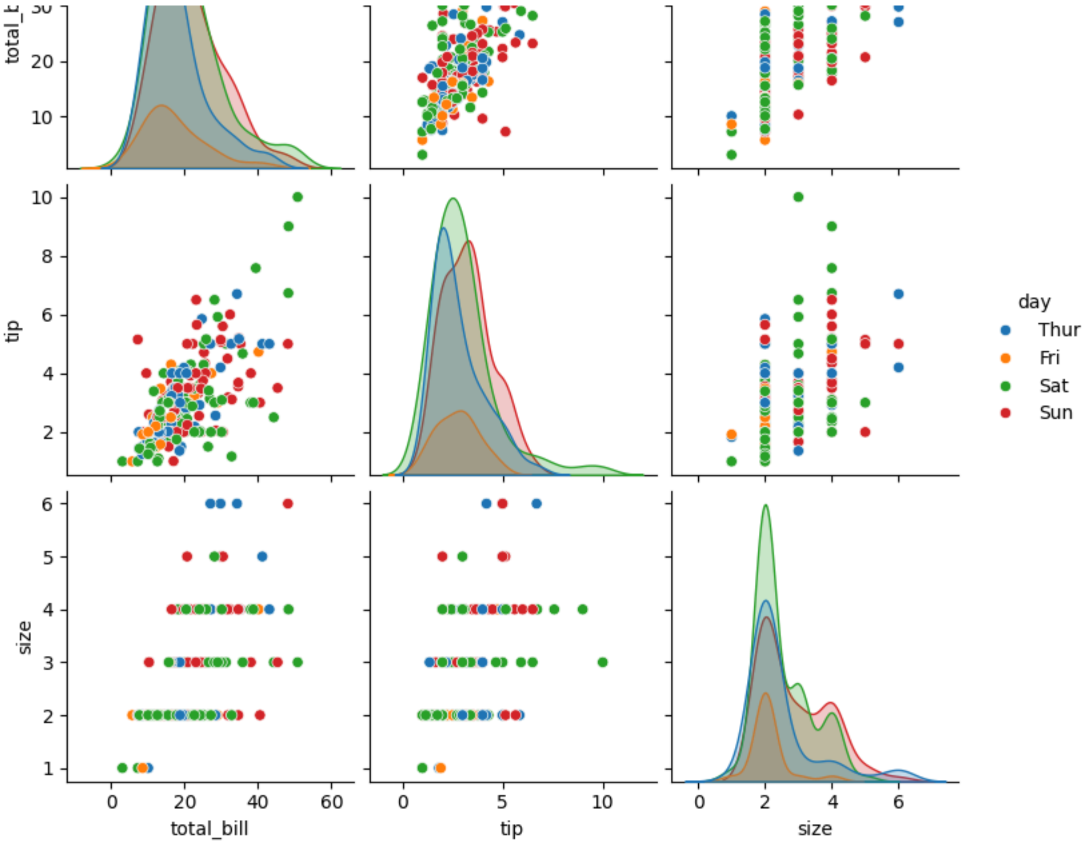

_bill
50
40
30

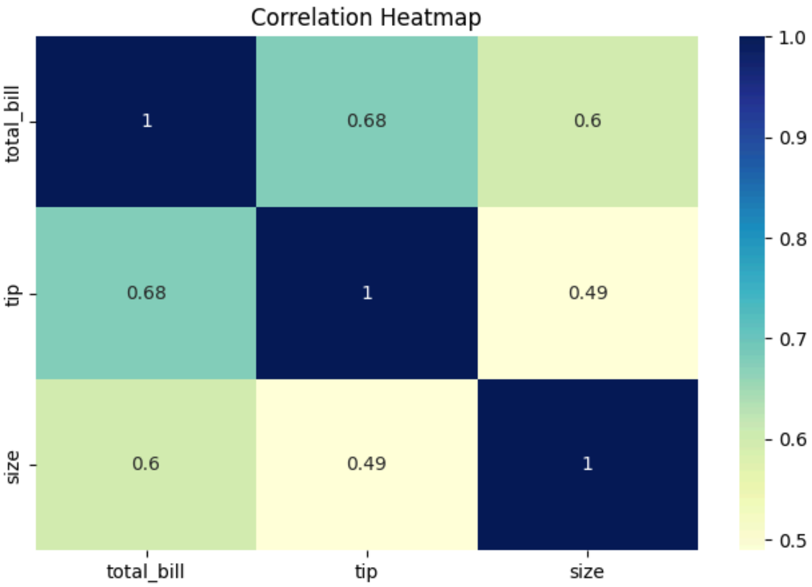


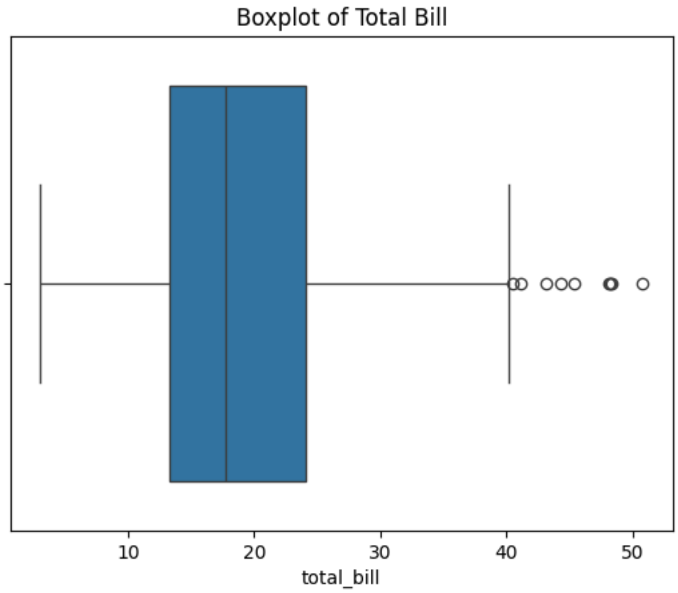
```
Time value counts:
time
Dinner    176
Lunch      68
Name: count, dtype: int64
```



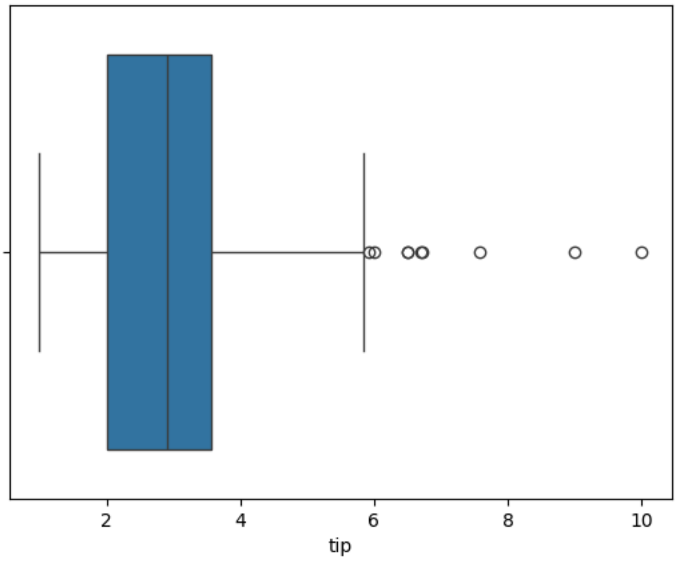


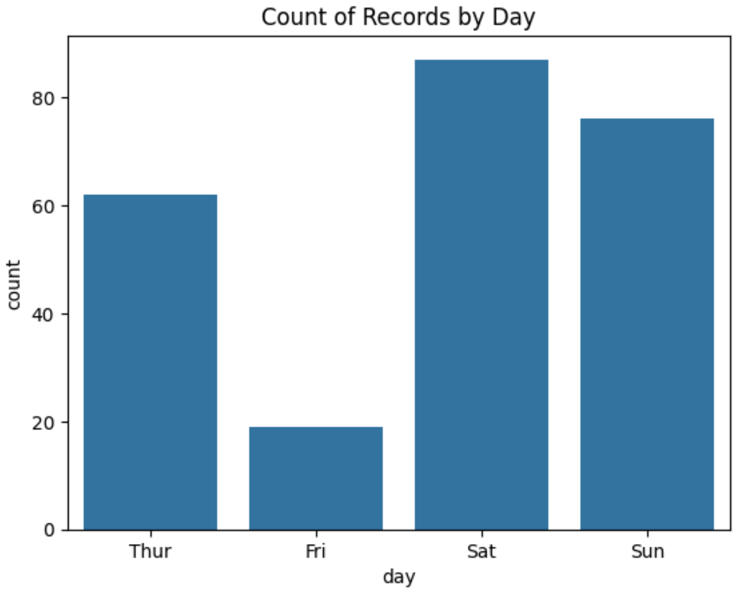


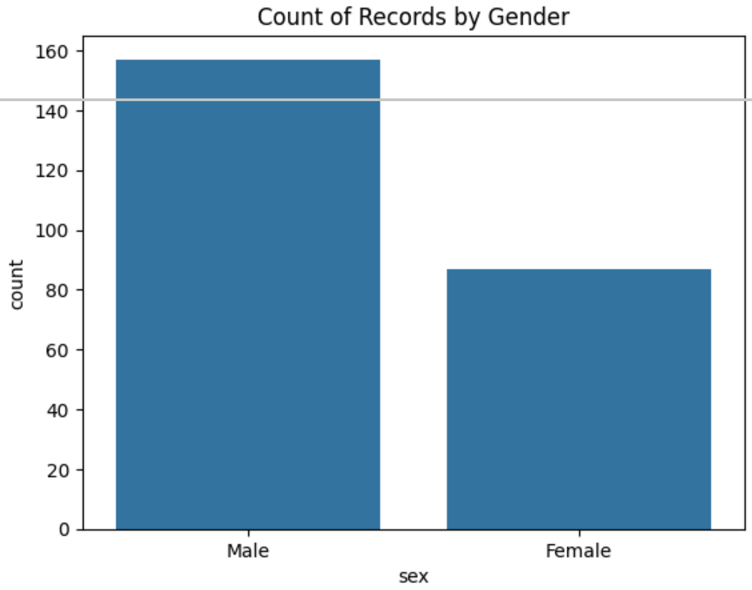

Correlation Heatmap


Boxplot of Total Bill

Boxplot of Tip







Gender Distribution (Pie)

```
#reshma s
#240701426
#17/08/2025
#cse

from google.colab import files
uploaded = files.upload()

import numpy as np
import pandas as pd
df=pd.read_csv('Salary_data.csv')
df
df.info()
df.dropna(inplace=True)
df.info()
df.describe()

features=df.iloc[:,[0]].values
label=df.iloc[:,[1]].values
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,rand
from sklearn.linear_model import LinearRegression
model=LinearRegression()
model.fit(x_train,y_train)

model.score(x_train,y_train)

model.score(x_test,y_test)

model.coef_

model.intercept_

import pickle
pickle.dump(model,open('SalaryPred.model','wb'))

model=pickle.load(open('SalaryPred.model','rb'))

yr_of_exp=float(input("Enter Years of Experience: "))
yr_of_exp_NP=np.array([[yr_of_exp]])
Salary=model.predict(yr_of_exp_NP)

print("Estimated Salary for {} years of experience is {}: " .format(yr_of_exp,Sa
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving Salary_data.csv to Salary_data.csv
<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: 612.0 bytes
<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: 612.0 bytes
Enter Years of Experience: 178
Estimated Salary for 178.0 years of experience is [[1679239.64461009]]:
```

```
#reshma s
#240701426
#17/08/2025
#cse

from google.colab import files
uploaded = files.upload()

import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report

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

print("---- Full Dataset ----")
print(df)

print("\n---- Head of Dataset ----")
print(df.head())

features = df.iloc[:, [2, 3]].values
label = df.iloc[:, 4].values

print("\n---- Features (Age, Estimated Salary) ----")
print(features)

print("\n---- Label (Purchased) ----")
print(label)

print("\n---- Finding Best Random State ----")
for i in range(1, 401):
    x_train, x_test, y_train, y_test = train_test_split(features, label, test_size=0.2, 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: {:.3f} | Train: {:.3f} | Random State: {}".format(test_score, train_score, i))

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)

print("\n---- Final Model Accuracy ----")
print("Train Accuracy:", finalModel.score(x_train, y_train))
print("Test Accuracy:", finalModel.score(x_test, y_test))

print("\n---- Classification Report ----")
print(classification_report(label, finalModel.predict(features)))
```

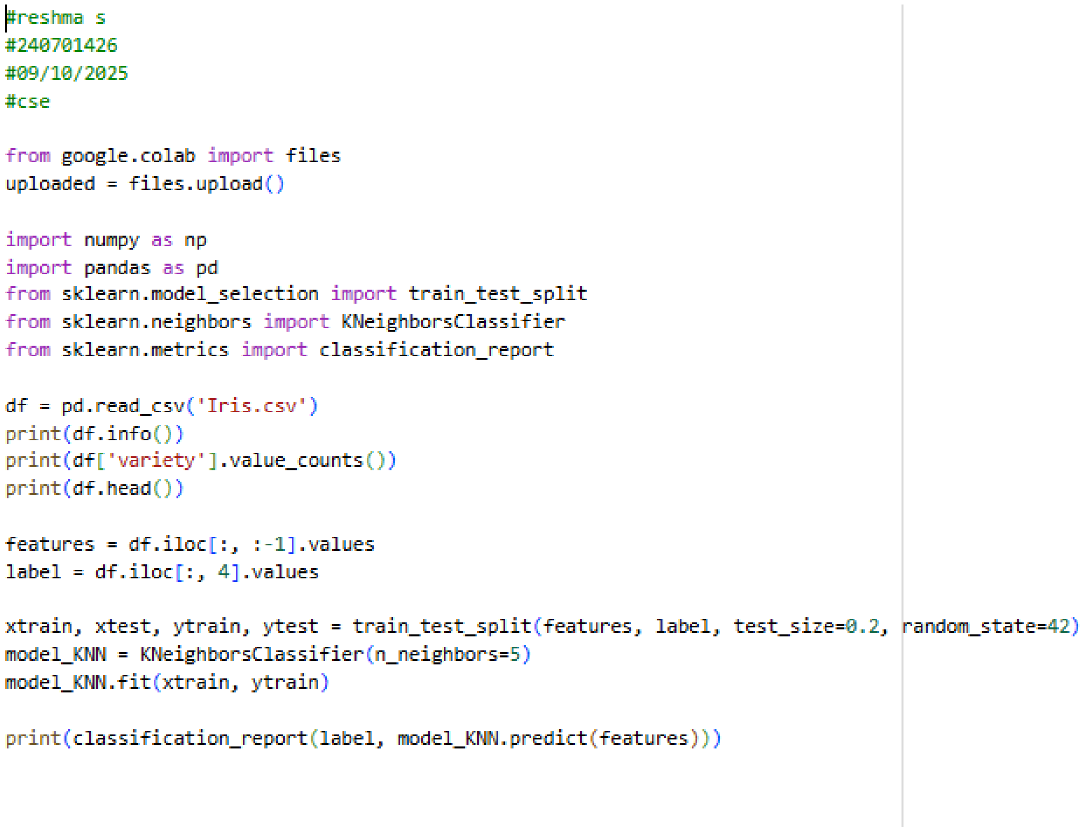
Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving Social_Network_Ads.csv to Social_Network_Ads.csv
---- Full Dataset ----
      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 x 5 columns]

---- Head of Dataset ----
    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

---- Features (Age, Estimated Salary) ----
[[    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]
 [    29  43000]
 [    31  18000]
 [    31  74000]
 [    27 137000]
 [    21  16000]
 [    28  44000]
 [    27  90000]
 [    35  27000]
 [    33  28000]
 [    30  49000]
 [    26  72000]
 [    27  31000]
 [    27  17000]
 [    33  51000]
 [    35 108000]
 [    30  15000]
 [    28  84000]
 [    23  20000]
 [    25  79000]
 [    27  54000]
 [    30 135000]
 [    31  89000]
 [    24  32000]
 [    18  44000]
 [    29  83000]
 [    35  23000]
 [    27  58000]
 [    24  55000]
 [    23  48000]
 [    28  79000]
 [    22  18000]
 [    32 117000]
 [    27  20000]
 [    25  87000]
 [    23  66000]
 [    32 120000]
 [    59  83000]
 [    24  58000]
 [    24  19000]
 [    23  82000]
 [    22  63000]
 [    31  68000]
 [    25  80000]
 [    24  27000]
 [    20  23000]
 [    33 113000]
 [    32  18000]
 [    34 112000]
 [    18  52000]
 [    22  27000]
 [    28  87000]
 [    26  17000]
 [    30  80000]
 [    39  42000]
 [    20  49000]
 [    35  88000]
 [    30  62000]
 [    31 118000]
 [    24  55000]
 [    28  85000]
 [    26  81000]
 [    35  50000]
 [    22  81000]
 [    30 116000]
 [    26  15000]
 [    29  28000]
 [    29  83000]
 [    35  44000]
 [    35  25000]
 [    28 123000]
 [    35  73000]
 [    28  37000]
 [    27  88000]
 [    28  59000]
 [    32  86000]
 [    33 149000]
 [    19  21000]
 [    21  72000]
 [    26  35000]
 [    27  89000]
 [    26  86000]
 [    38  80000]
 [    39  71000]
 [    37  71000]
 [    38  61000]
 [    37  55000]
 [    42  80000]
 [    40  57000]
 [    35  75000]
 [    36  52000]
 [    40  59000]
 [    41  59000]
 [    36  75000]
 [    37  72000]
 [    40  75000]
 [    35  53000]
 [    41  51000]
 [    39  61000]
 [    42  65000]
 [    26  32000]
 [    30  17000]
 [    26  84000]
 [    31  58000]
 [    33  31000]
 [    30  87000]
 [    21  68000]
 [    28  55000]
 [    23  63000]
 [    20  82000]
 [    30 107000]
 [    28  59000]
 [    19  25000]
 [    19  85000]
 [    18  68000]
 [    35  59000]
 [    30  89000]
 [    34  25000]
 [    24  89000]
 [    27  96000]
 [    41  30000]
 [    29  61000]
 [    20  74000]
 [    26  15000]
 [    41  45000]
 [    31  76000]
 [    36  50000]
 [    40  47000]
 [    31  15000]
 [    46  59000]
 [    29  75000]
 [    26  30000]
 [    32 135000]
 [    32 100000]
 [    25  90000]
 [    37  33000]
 [    35  38000]
 [    33  69000]
 [    18  86000]
 [    22  55000]
 [    35  71000]
 [    29 148000]
 [    29  47000]
 [    21  88000]
 [    34 115000]
 [    26 118000]
 [    34  43000]
 [    34  72000]
 [    23  28000]
 [    35  47000]
 [    25  22000]
 [    24  23000]
 [    31  34000]
 [    26  16000]
 [    31  71000]
 [    32 117000]
 [    33  43000]
 [    33  60000]
 [    31  66000]
 [    20  82000]
 [    33  41000]
 [    35  72000]
 [    28  32000]
 [    24  84000]
 [    19  26000]
 [    29  43000]
 [    19  70000]
 [    28  89000]
 [    34  43000]
 [    30  79000]
 [    20  36000]
 [    26  80000]
 [    35  22000]
 [    35  39000]
 [    49  74000]
 [    39 134000]
 [    41  71000]
 [    58 101000]
 [    47  47000]
 [    55 130000]
 [    52 114000]
 [    40 142000]
 [    46  22000]
 [    48  96000]
 [    52 150000]
 [    59  42000]
 [    35  58000]
 [    47  43000]
 [    60 108000]
 [    49  65000]
 [    40  78000]
 [    46  96000]
 [    59 143000]
 [    41  80000]
 [    35  91000]
 [    37 144000]
 [    60 102000]
 [    35  60000]
 [    37  53000]
 [    36 126000]
 [    56 133000]
 [    40  72000]
 [    42  80000]
 [    35 147000]
 [    39  42000]
 [    40 107000]
 [    49  86000]
 [    38 112000]
 [    46  79000]
 [    40  57000]
 [    37  80000]
 [    46  82000]
 [    53 143000]
 [    42 149000]
 [    38  59000]
 [    50  88000]
 [    56 104000]
 [    41  72000]
 [    51 146000]
 [    35  50000]
 [    57 122000]
 [    41  52000]
 [    35  97000]
 [    44  39000]
 [    37  52000]
 [    48 134000]
 [    37 146000]
 [    50  44000]
 [    52  90000]
 [    41  72000]
 [    40  57000]
 [    58  95000]
 [    45 131000]
 [    35  77000]
 [    36 144000]
 [    55 125000]
 [    35  72000]
 [    48  90000]
 [    42 108000]
 [    40  75000]
 [    37  74000]
 [    47 144000]
 [    40  61000]
 [    43 133000]
 [    59  76000]
 [    60  42000]
 [    39 106000]
 [    57  26000]
 [    57  74000]
 [    38  71000]
 [    49  88000]
 [    52  38000]
 [    50  36000]
 [    59  88000]
 [    35  61000]
 [    37  70000]
 [    52  21000]
 [    48 141000]
 [    37  93000]
 [    37  62000]
 [    48 138000]
 [    41  79000]
 [    37  78000]
 [    39 134000]
 [    49  89000]
 [    55  39000]
 [    37  77000]
 [    35  57000]
 [    36  63000]
 [    42  73000]
 [    43 112000]
 [    45  79000]
 [    46 117000]
 [    58  38000]
 [    48  74000]
 [    37 137000]
 [    37  79000]
 [    40  60000]
 [    42  54000]
 [    51 134000]
 [    47 113000]
 [    36 125000]
 [    38  50000]
 [    42  70000]
 [    39  96000]
 [    38  50000]
 [    49 141000]
 [    39  79000]
 [    39  75000]
 [    54 104000]
 [    35  55000]
 [    45  32000]
 [    36  60000]
 [    52 138000]
 [    53  82000]
 [    41  52000]
 [    48  30000]
 [    48 131000]
 [    41  60000]
 [    41  72000]
 [    42  75000]
 [    36 118000]
 [    47 107000]
 [    38  51000]
 [    48 119000]
 [    42  65000]
 [    40  65000]
 [    57  60000]
 [    36  54000]
 [    58 144000]
 [    35  79000]
 [    38  55000]
 [    39 122000]
 [    53 104000]
 [    35  75000]
 [    38  65000]
 [    47  51000]
 [    47 105000]
 [    41  63000]
 [    53  72000]
 [    54 108000]
 [    39  77000]
 [    38  61000]
 [    38 113000]
 [    37  75000]
 [    42  90000]
 [    37  57000]
 [    36  99000]
 [    60  34000]
 [    54  70000]
 [    41  72000]
 [    40  71000]
 [    42  54000]
 [    43 129000]
 [    53  34000]
 [    47  50000]
 [    42  79000]
 [    42 104000]
 [    59  29000]
 [    58  47000]
 [    46  88000]
 [    38  71000]
 [    54  26000]
 [    60  46000]
 [    60  83000]
 [    39  73000]
 [    59 130000]
 [    37  80000]
 [    46  32000]
 [    46  74000]
 [    42  53000]
 [    41  87000]
 [    58  23000]
 [    42  64000]
 [    48  33000]
 [    44 139000]
 [    49  28000]
 [    57  33000]
 [    56  60000]
 [    49  39000]
 [    39  71000]
 [    47  34000]
 [    48  35000]
 [    48  33000]
 [    47  23000]
 [    45  45000]
 [    60  42000]
 [    39  59000]
 [    46  41000]
 [    51  23000]
 [    50  20000]
 [    36  33000]
 [    49  36000]]

---- Label (Purchased) ----
[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 0 1 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 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]

---- Finding Best Random State ----
Test: 0.900 | Train: 0.841 | Random State: 4
Test: 0.863 | Train: 0.850 | Random State: 5
Test: 0.863 | Train: 0.859 | Random State: 6
Test: 0.887 | Train: 0.838 | Random State: 7
Test: 0.863 | Train: 0.838 | Random State: 9
Test: 0.900 | Train: 0.841 | Random State: 10
Test: 0.863 | Train: 0.856 | Random State: 14
Test: 0.850 | Train: 0.844 | Random State: 15
Test: 0.863 | Train: 0.856 | Random State: 16
```

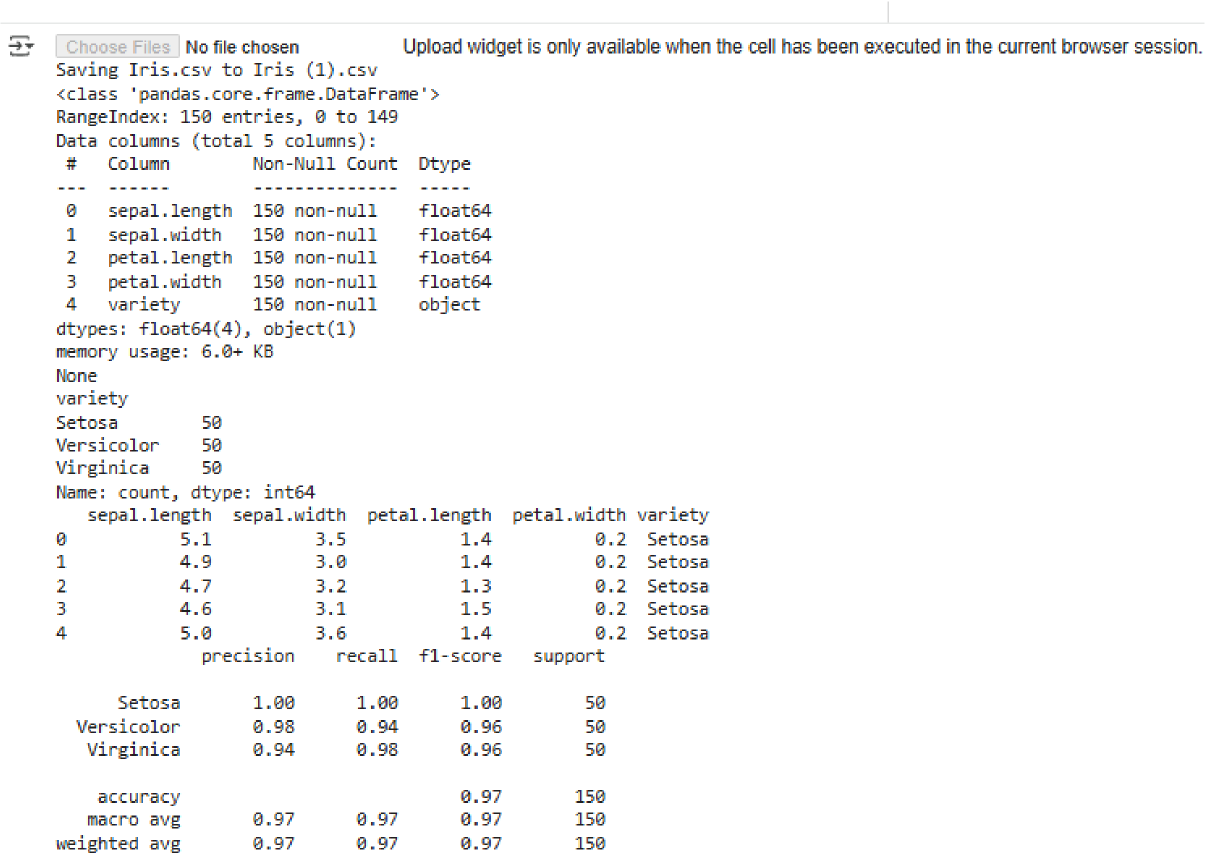
```
    Test: 0.875 | Train: 0.834 | Random State: 18
    Test: 0.850 | Train: 0.844 | Random State: 19
    Test: 0.875 | Train: 0.844 | Random State: 20
    Test: 0.863 | Train: 0.834 | Random State: 21
    Test: 0.875 | Train: 0.841 | Random State: 22
    Test: 0.875 | Train: 0.841 | Random State: 24
    Test: 0.850 | Train: 0.834 | Random State: 26
    Test: 0.850 | Train: 0.841 | Random State: 27
    Test: 0.863 | Train: 0.834 | Random State: 30
    Test: 0.863 | Train: 0.856 | Random State: 31
    Test: 0.875 | Train: 0.853 | Random State: 32
    Test: 0.863 | Train: 0.844 | Random State: 33
    Test: 0.875 | Train: 0.831 | Random State: 35
    Test: 0.863 | Train: 0.853 | Random State: 36
    Test: 0.887 | Train: 0.841 | Random State: 38
    Test: 0.875 | Train: 0.838 | Random State: 39
    Test: 0.887 | Train: 0.838 | Random State: 42
    Test: 0.875 | Train: 0.847 | Random State: 46
    Test: 0.912 | Train: 0.831 | Random State: 47
    Test: 0.875 | Train: 0.831 | Random State: 51
    Test: 0.900 | Train: 0.844 | Random State: 54
    Test: 0.850 | Train: 0.844 | Random State: 57
    Test: 0.875 | Train: 0.844 | Random State: 58
    Test: 0.925 | Train: 0.838 | Random State: 61
    Test: 0.887 | Train: 0.834 | Random State: 65
    Test: 0.887 | Train: 0.841 | Random State: 68
    Test: 0.900 | Train: 0.831 | Random State: 72
    Test: 0.887 | Train: 0.838 | Random State: 75
    Test: 0.925 | Train: 0.825 | Random State: 76
    Test: 0.863 | Train: 0.841 | Random State: 77
    Test: 0.863 | Train: 0.859 | Random State: 81
    Test: 0.875 | Train: 0.838 | Random State: 82
    Test: 0.887 | Train: 0.838 | Random State: 83
    Test: 0.863 | Train: 0.853 | Random State: 84
    Test: 0.863 | Train: 0.841 | Random State: 85
    Test: 0.863 | Train: 0.841 | Random State: 87
    Test: 0.875 | Train: 0.847 | Random State: 88
    Test: 0.912 | Train: 0.838 | Random State: 90
    Test: 0.863 | Train: 0.850 | Random State: 95
    Test: 0.875 | Train: 0.850 | Random State: 99
    Test: 0.850 | Train: 0.841 | Random State: 101
    Test: 0.850 | Train: 0.841 | Random State: 102
    Test: 0.900 | Train: 0.825 | Random State: 106
    Test: 0.863 | Train: 0.841 | Random State: 107
    Test: 0.850 | Train: 0.834 | Random State: 109
    Test: 0.850 | Train: 0.841 | Random State: 111
    Test: 0.912 | Train: 0.841 | Random State: 112
    Test: 0.863 | Train: 0.850 | Random State: 115
    Test: 0.863 | Train: 0.841 | Random State: 116
    Test: 0.875 | Train: 0.834 | Random State: 119
    Test: 0.912 | Train: 0.828 | Random State: 120
    Test: 0.863 | Train: 0.859 | Random State: 125
    Test: 0.850 | Train: 0.847 | Random State: 128
    Test: 0.875 | Train: 0.850 | Random State: 130
    Test: 0.900 | Train: 0.844 | Random State: 133
    Test: 0.925 | Train: 0.834 | Random State: 134
    Test: 0.863 | Train: 0.850 | Random State: 135
    Test: 0.875 | Train: 0.831 | Random State: 138
    Test: 0.863 | Train: 0.850 | Random State: 141
    Test: 0.850 | Train: 0.847 | Random State: 143
    Test: 0.850 | Train: 0.847 | Random State: 146
    Test: 0.850 | Train: 0.844 | Random State: 147
    Test: 0.863 | Train: 0.850 | Random State: 148
    Test: 0.875 | Train: 0.838 | Random State: 150
    Test: 0.887 | Train: 0.831 | Random State: 151
    Test: 0.925 | Train: 0.844 | Random State: 152
    Test: 0.850 | Train: 0.841 | Random State: 153
    Test: 0.900 | Train: 0.844 | Random State: 154
    Test: 0.900 | Train: 0.841 | Random State: 155
    Test: 0.887 | Train: 0.847 | Random State: 156
    Test: 0.887 | Train: 0.834 | Random State: 158
    Test: 0.875 | Train: 0.828 | Random State: 159
    Test: 0.900 | Train: 0.831 | Random State: 161
    Test: 0.850 | Train: 0.838 | Random State: 163
    Test: 0.875 | Train: 0.831 | Random State: 164
    Test: 0.863 | Train: 0.850 | Random State: 169
    Test: 0.875 | Train: 0.841 | Random State: 171
    Test: 0.850 | Train: 0.841 | Random State: 172
    Test: 0.900 | Train: 0.825 | Random State: 180
    Test: 0.850 | Train: 0.834 | Random State: 184
    Test: 0.925 | Train: 0.822 | Random State: 186
    Test: 0.900 | Train: 0.831 | Random State: 193
    Test: 0.863 | Train: 0.850 | Random State: 195
    Test: 0.863 | Train: 0.841 | Random State: 196
    Test: 0.863 | Train: 0.838 | Random State: 197
    Test: 0.875 | Train: 0.841 | Random State: 198
    Test: 0.887 | Train: 0.838 | Random State: 199
    Test: 0.887 | Train: 0.844 | Random State: 200
    Test: 0.863 | Train: 0.838 | Random State: 202
    Test: 0.863 | Train: 0.841 | Random State: 203
    Test: 0.887 | Train: 0.831 | Random State: 206
```

```
Test: 0.887 | Train: 0.831 | Random State: 206
Test: 0.863 | Train: 0.834 | Random State: 211
Test: 0.850 | Train: 0.844 | Random State: 212
Test: 0.863 | Train: 0.834 | Random State: 214
Test: 0.875 | Train: 0.831 | Random State: 217
Test: 0.963 | Train: 0.819 | Random State: 220
Test: 0.875 | Train: 0.844 | Random State: 221
Test: 0.850 | Train: 0.841 | Random State: 222
Test: 0.900 | Train: 0.844 | Random State: 223
Test: 0.863 | Train: 0.853 | Random State: 227
Test: 0.863 | Train: 0.834 | Random State: 228
```

# EXP 10 – KKN

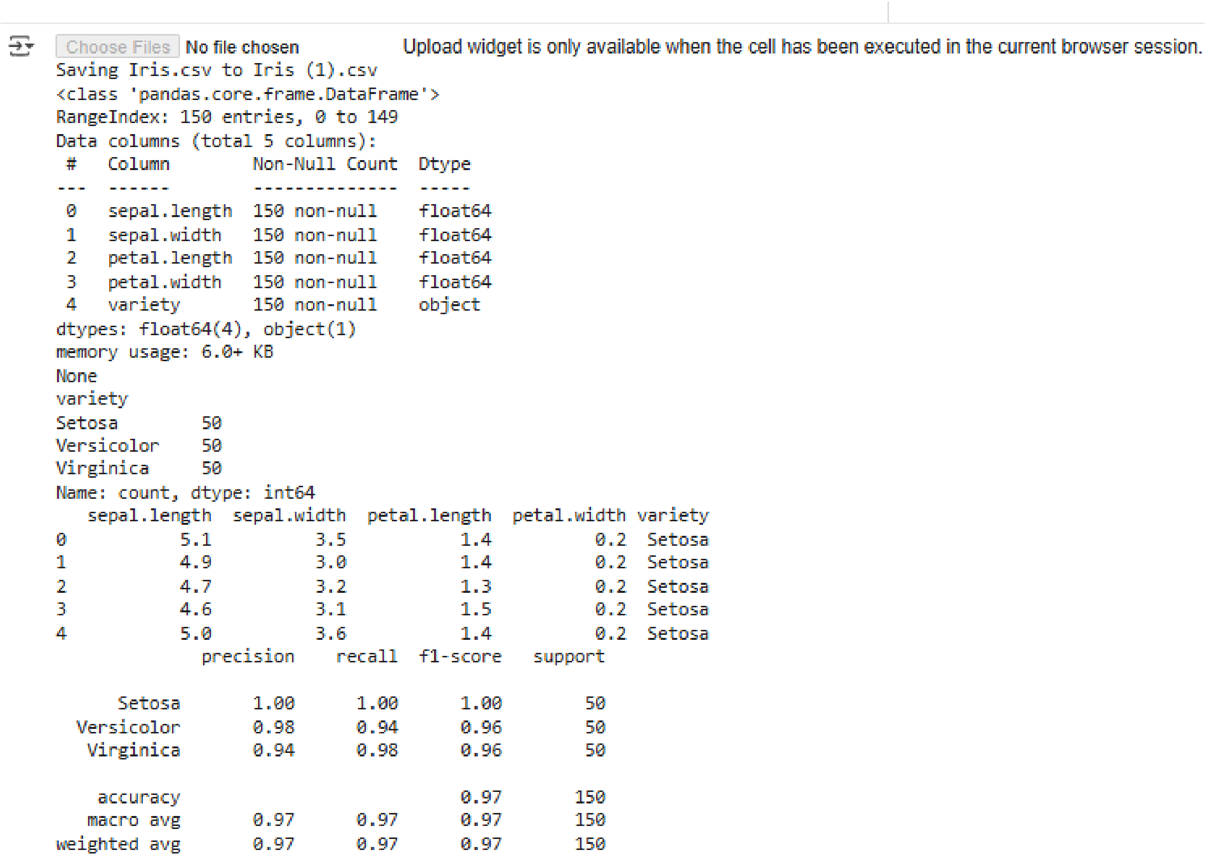


```python
#reshma s
#240701426
#09/10/2025
#cse

from google.colab import files
uploaded = files.upload()

import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import classification_report

df = pd.read_csv('Iris.csv')
print(df.info())
print(df['variety'].value_counts())
print(df.head())

features = df.iloc[:, :-1].values
label = df.iloc[:, 4].values

xtrain, xtest, ytrain, ytest = train_test_split(features, label, test_size=0.2, random_state=42)
model_KNN = KNeighborsClassifier(n_neighbors=5)
model_KNN.fit(xtrain, ytrain)

print(classification_report(label, model_KNN.predict(features)))
```



Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session.

```
Saving Iris.csv to Iris (1).csv
<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
None
variety
Setosa        50
Versicolor    50
Virginica     50
Name: count, dtype: int64
   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
              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
```



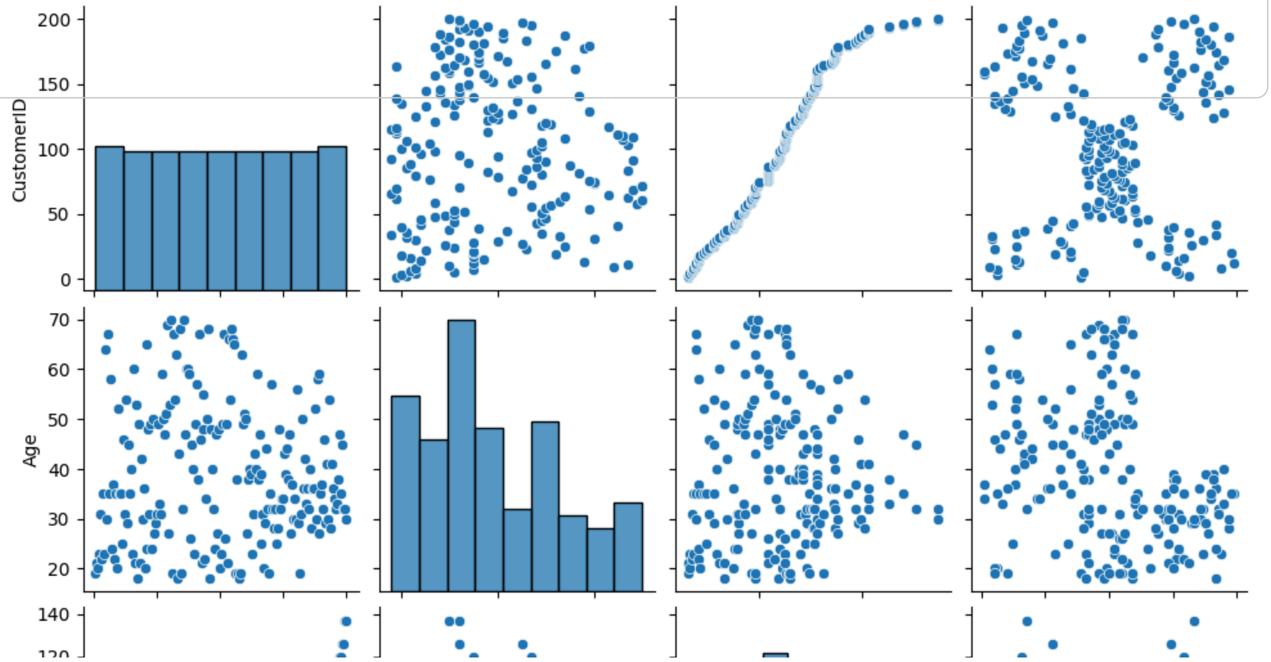
Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session.

```
Saving Iris.csv to Iris (1).csv
<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
None
variety
Setosa        50
Versicolor    50
Virginica     50
Name: count, dtype: int64
   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
              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
```

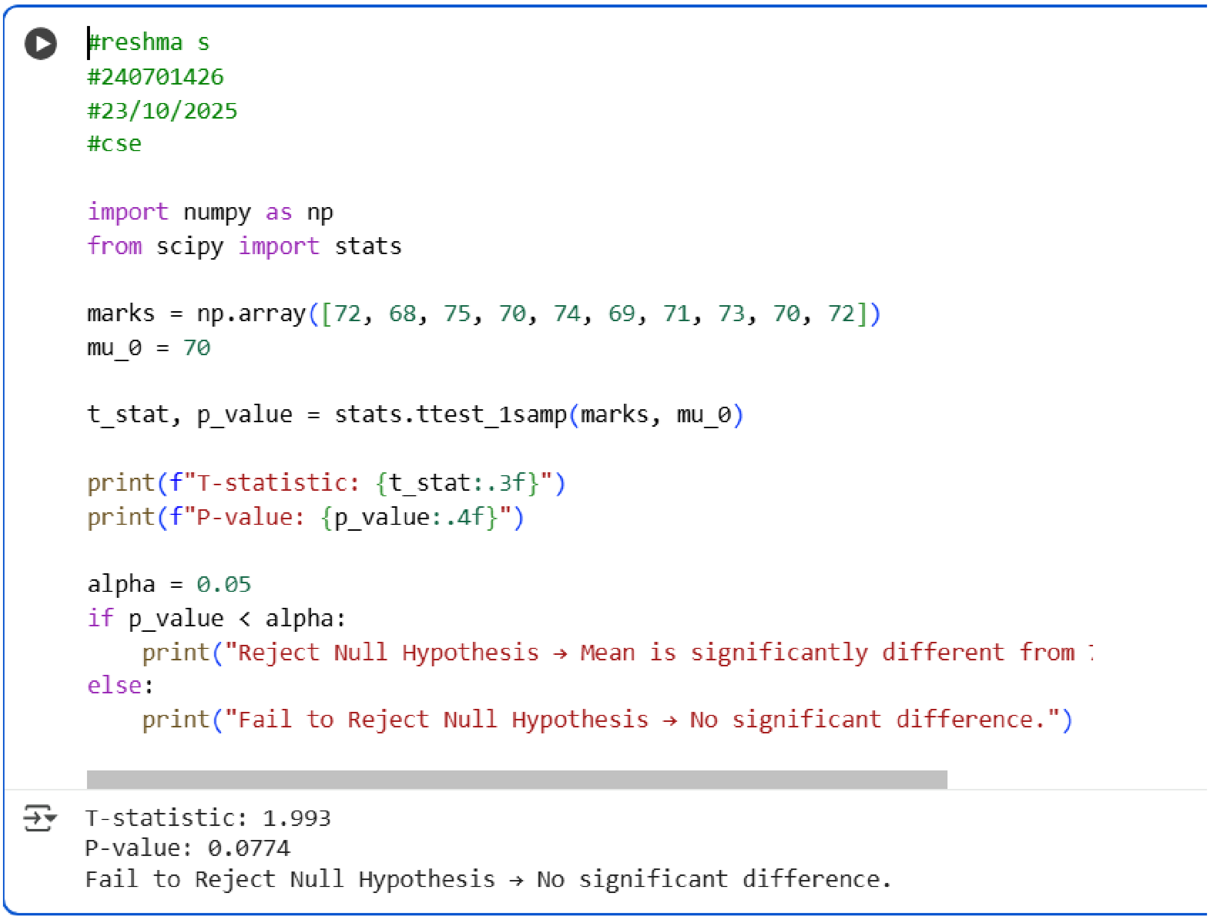
```
#reshma s
#240701426
#03/08/2025
#cse
from google.colab import files
uploaded = files.upload()


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

df = pd.read_csv('Mall_Customers.csv')
print(df.info())
print(df.head())

sns.pairplot(df)
plt.show()

features = df.iloc[:, [3, 4]].values
from sklearn.cluster import KMeans
model = KMeans(n_clusters=5, random_state=42)
model.fit(features)

Final = df.iloc[:, [3, 4]].copy()
Final['label'] = model.predict(features)
print(Final)

sns.set_style("whitegrid")
plt.figure(figsize=(8,6))
sns.scatterplot(data=Final, x="Annual Income (k$)", y="Spending Score (1-100)", hue="label", palette="Set2", s=80)
plt.title("K-Means Clustering of Customers")
plt.show()

features_el = df.iloc[:, [2, 3, 4]].values
wcss = []
for i in range(1, 10):
    model = KMeans(n_clusters=i, random_state=42)
    model.fit(features_el)
    wcss.append(model.inertia_)

plt.figure(figsize=(8,5))
plt.plot(range(1, 10), wcss, marker='o', color='blue')
plt.title("Elbow Method for Optimal K")
plt.xlabel("Number of Clusters")
plt.ylabel("WCSS")
plt.show()
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving Mall_Customers.csv to Mall_Customers.csv
<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$)      200 non-null    int64 
 4   Spending Score (1-100)  200 non-null    int64 
dtypes: int64(4), object(1)
memory usage: 7.9+ KB
None
   CustomerID  Gender  Age  Annual Income (k$)  Spending Score (1-100)
0           1    Male   19                  15                      39
1           2    Male   21                  15                      81
2           3  Female   20                  16                       6
3           4  Female   23                  16                      77
4           5  Female   31                  17                      40
```



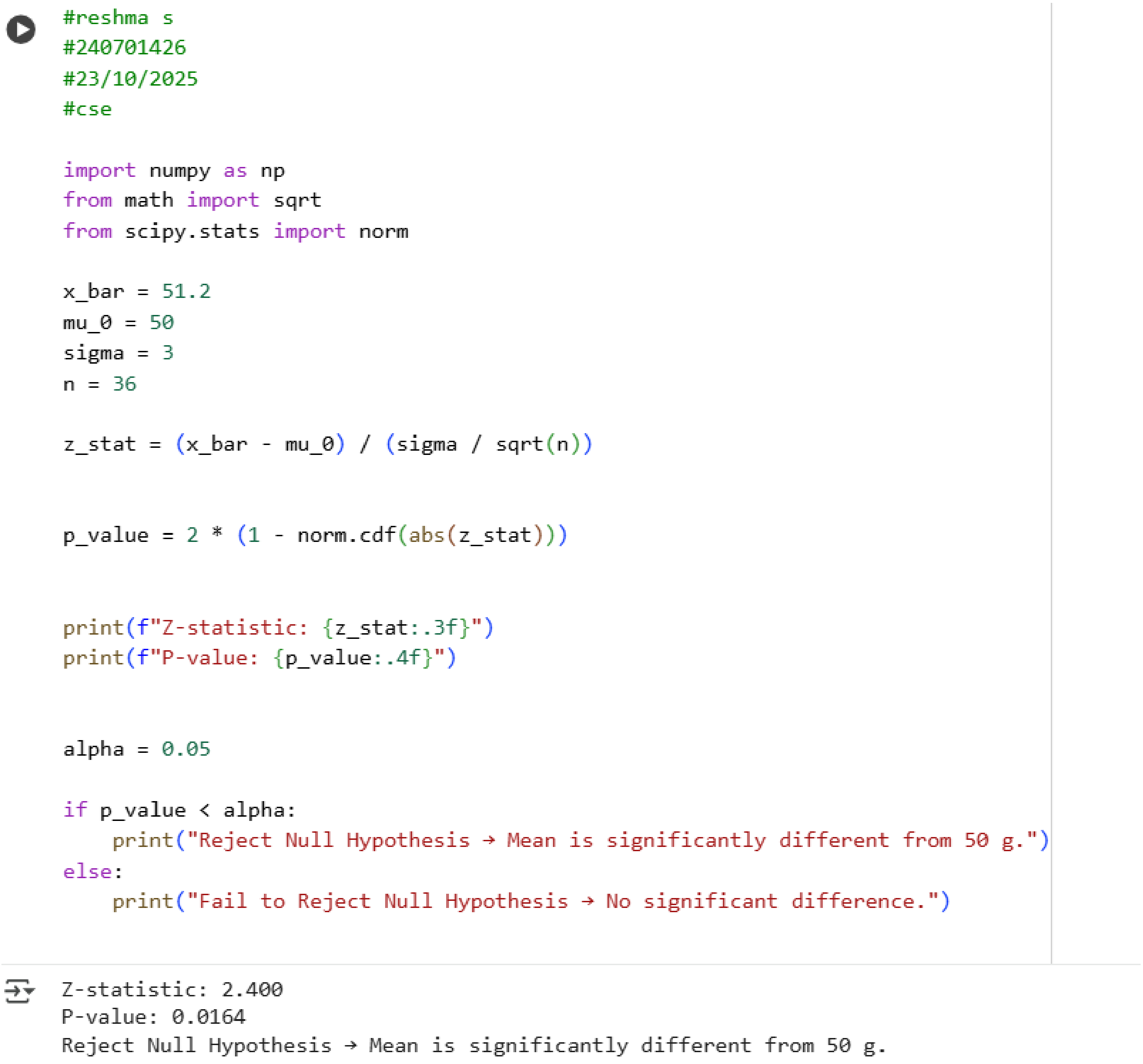
EXP 12 –  T Test



```python
#reshma s
#240701426
#23/10/2025
#cse

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
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.
```

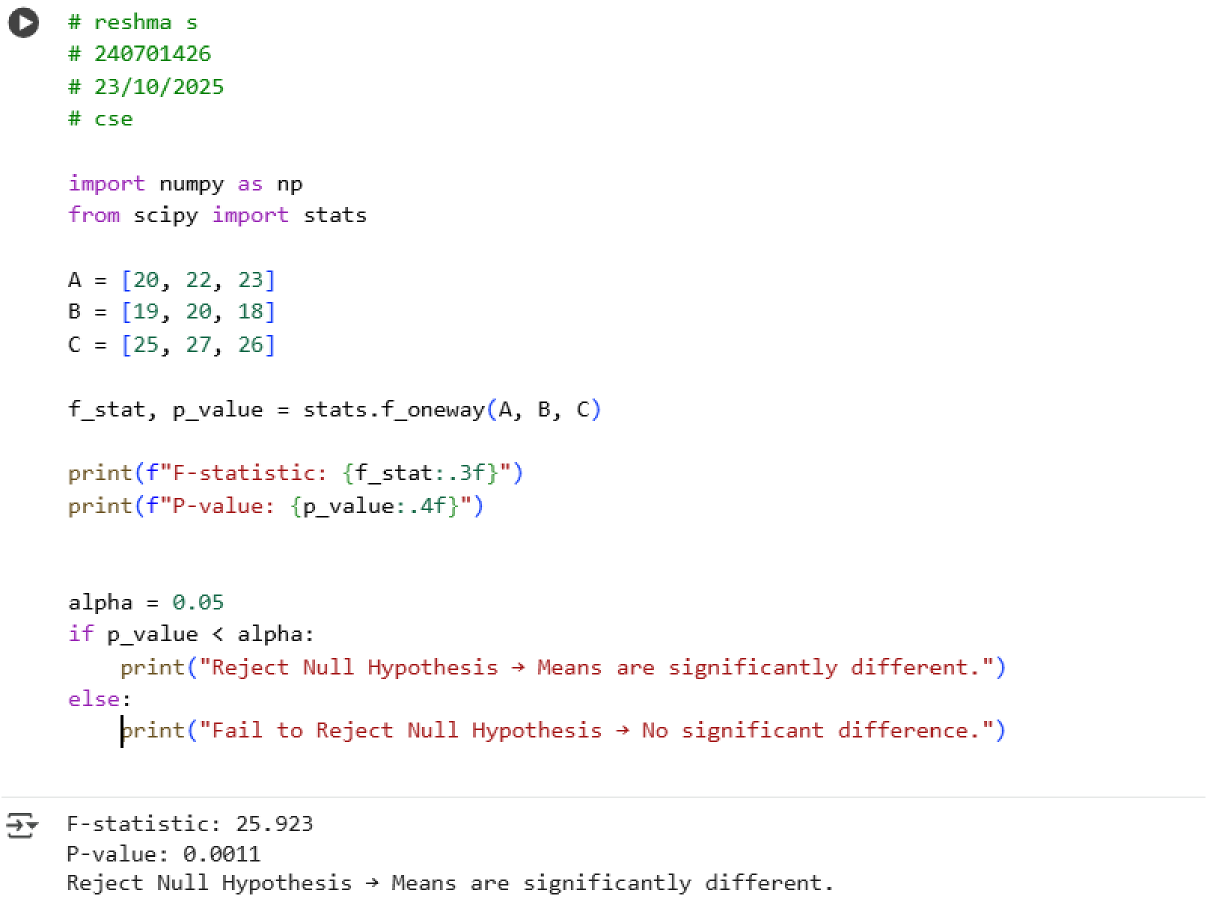
EXP 13 - z test



```
#reshma s
#240701426
#23/10/2025
#cse

import numpy as np
from math import sqrt
from scipy.stats import norm

x_bar = 51.2
mu_0 = 50
sigma = 3
n = 36

z_stat = (x_bar - mu_0) / (sigma / sqrt(n))


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.
```

# EXP 14 – Anova test



```
# reshma s
# 240701426
# 23/10/2025
# cse

import numpy as np
from scipy import stats

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

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.
```