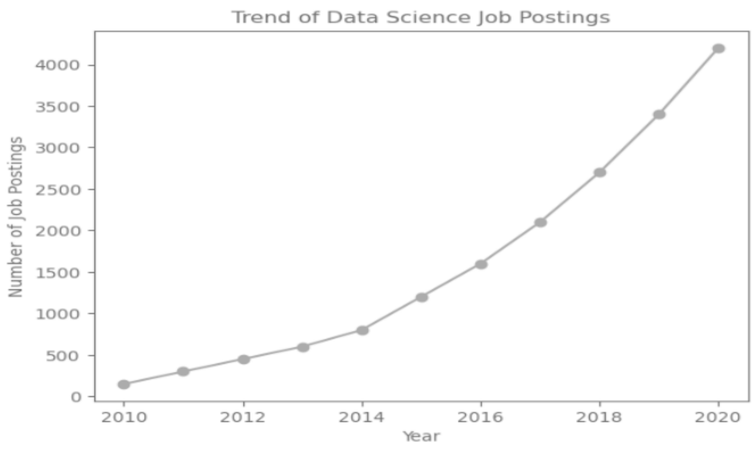
# Name: Athiena Rachel

# Roll no.: 230701016

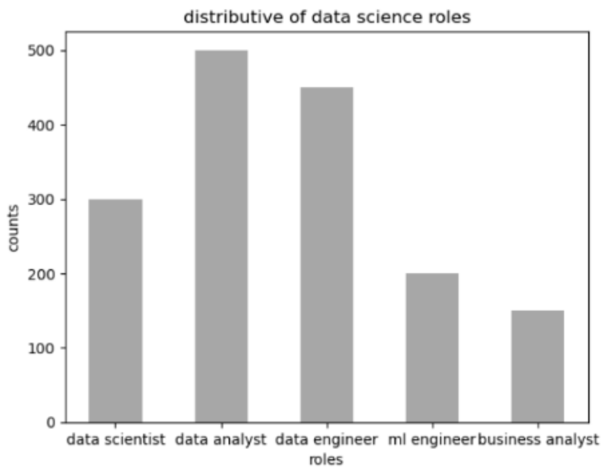
# Date: 30.07.2024

#Exercise: Matplotseaborn

```
import pandas as pd
import matplotlib.pyplot as plt
data = {&#39;Year&#39;: list(range(2010, 2021)),
&#39;Job Postings&#39;: [150, 300, 450, 600, 800, 1200, 1600, 2100, 2700, 3400, 4200]}
df = pd.DataFrame(data)
plt.plot(df[&#39;Year&#39;], df[&#39;Job Postings&#39;], marker=&#39;o&#39;)
plt.title(&#39;Trend of Data Science Job Postings&#39;)
plt.xlabel(&#39;Year&#39;)
plt.ylabel(&#39;Number of Job Postings&#39;)
plt.show()
```



```
import pandas as pd

import matplotlib.pyplot as plt

roles=[&#39;data scientist&#39;,&#39;data analyst&#39;,&#39;data
engineer&#39;,&#39;ml engineer&#39;,&#39;business analyst&#39;]

counts=[300,500,450,200,150]

plt.bar(roles,counts,width=0.5,color=&#39;yellow&#39;)

plt.title(&#39;distributive of data science roles&#39;)

plt.xlabel(&#39;roles&#39;)

plt.ylabel(&#39;counts&#39;)

plt.show()
```



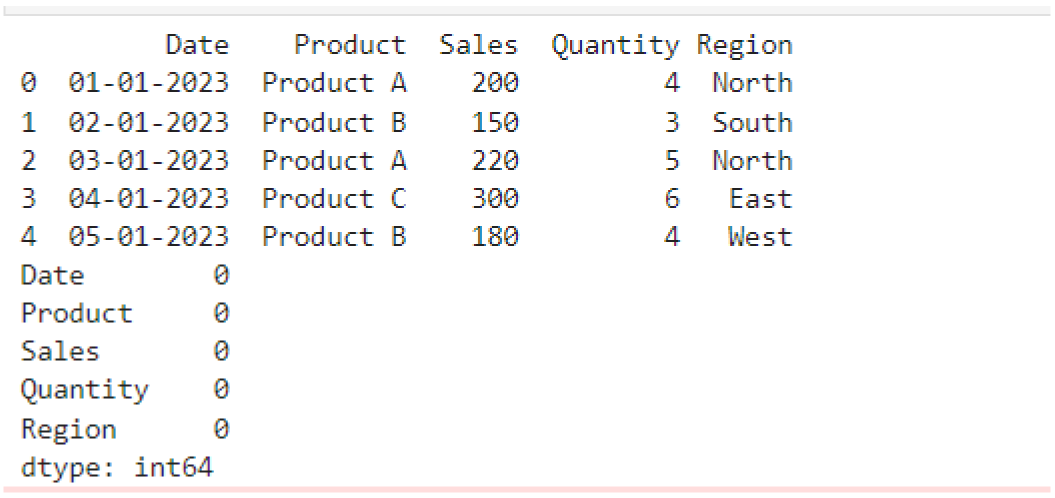
```
import pandas as pd

structured_data = pd. DataFrame ({'ID's [1, 2, 3], 'NAME': ['Aish', ''Betty', 'Cathy'], 'AGE':
[18, 20, 25], 'GRADE': ['O', 'A', 'B'],

'SKILL': ['Art', 'Music', 'Dance']})

print (''Structured Data: \n'', structured_data)

unstructured_data = "This is an unstructured data. It can be text, an image, or a video"

print(''unstructured data: \n'', unstructured_data)

semistrictured_clata = {'ID': 1, 'NAME': 'Alice', 'ATTRIBUTES'':
```

```
'HEIGHT': 170, 'WEIGHT: 45}}
print("Semistructured Data: \n", semistrictured_data)
```

OUTPUT-

```
Structured Data:
'ID's [1, 2, 3], 'NAME': ['Aish', "Betty', 'Cathy'], 'AGE': [18, 20, 25], 'GRADE': ['O',
'A', 'B'],
'SKILL': ['Art', 'Music', 'Dance']

unstructured data:
This is an unstructured data. It can be text, an image, or a video
Semistructured Data:
'ID': 1, 'NAME': 'Alice', 'ATTRIBUTES":
'HEIGHT': 170, 'WEIGHT: 45}
```

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
df=pd.read_csv ('sales_data.csv');
print(df.head());
print (df.isnull().sum())
df ['Sales ']. fillna(df ['Sales']. mean(), inplace = True)
df. dropna (subset = ['Product', 'Quantity', 'Region'], inplace = True)
print (df. describe())
prodsummary= df.groupby ('Product'). agg ({'sales': 'sum', 'Quantity': 'sum'}).reset_index()
print (prodsummary)
```



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

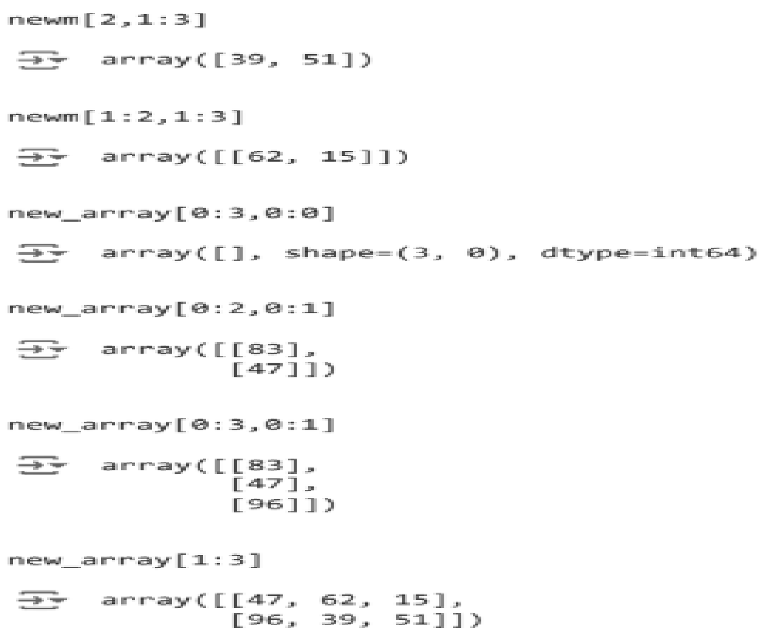
# Name: Athiena Rachel

# Roll no.: 230701016

# Date: 06.08.2024

#Exercise: Pandas Buit in function; Numpy Buit in fuction- Array slicing, Ravel,Reshape,ndim

```
import numpy as np
array = np.random.randint(1, 100, 9)
np.sqrt(array)
array.ndim
new_array = array.reshape(3, 3)
new_array.ndim
new_array.ravel()
newm = new_array.reshape(3, 3)
newm[2, 1:3]
newm[1:2, 1:3]
new_array[0:3, 0:0]
new_array[0:2, 0:1]
new_array[0:3, 0:1]
new_array[1:3]
```



```
newm[2,1:3]

array([39, 51])

newm[1:2,1:3]

array([[62, 15]])

new_array[0:3,0:0]

array([], shape=(3, 0), dtype=int64)

new_array[0:2,0:1]

array([[83],
       [47]])

new_array[0:3,0:1]

array([[83],
       [47],
       [96]])

new_array[1:3]

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

# Name: Athiena Rachel

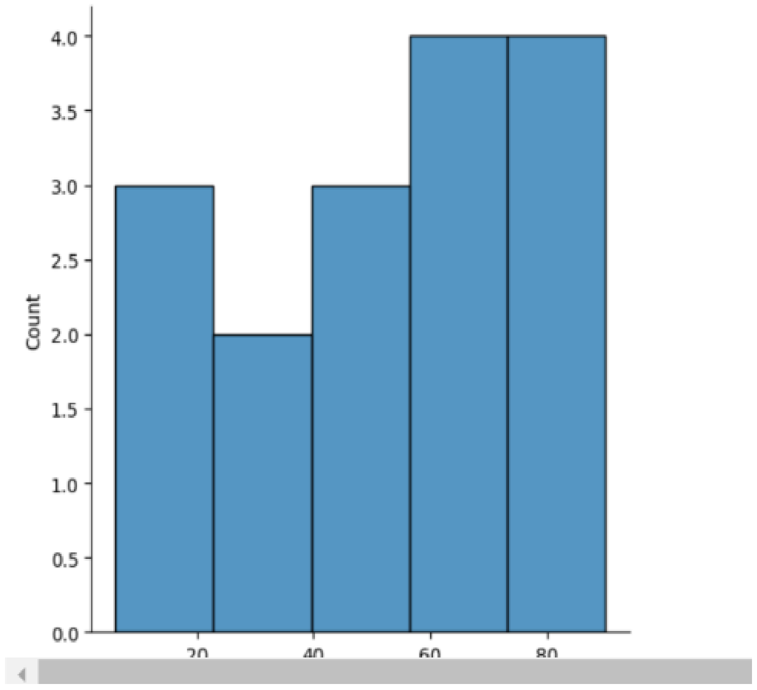
# Roll no.: 230701016

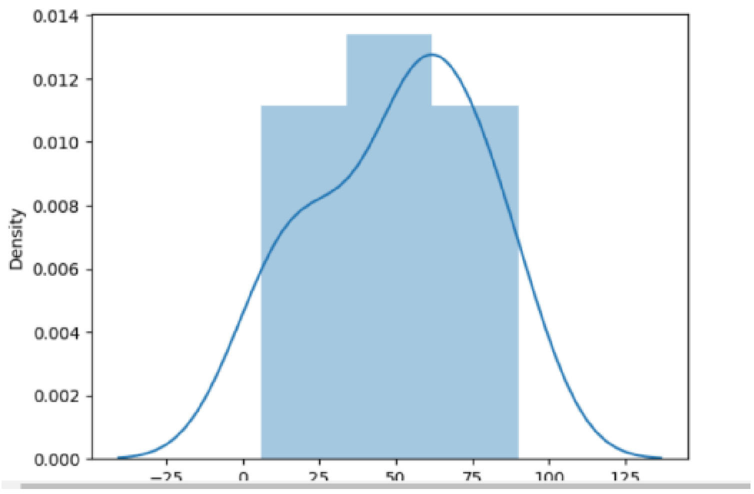
# Date: 28.08.2024

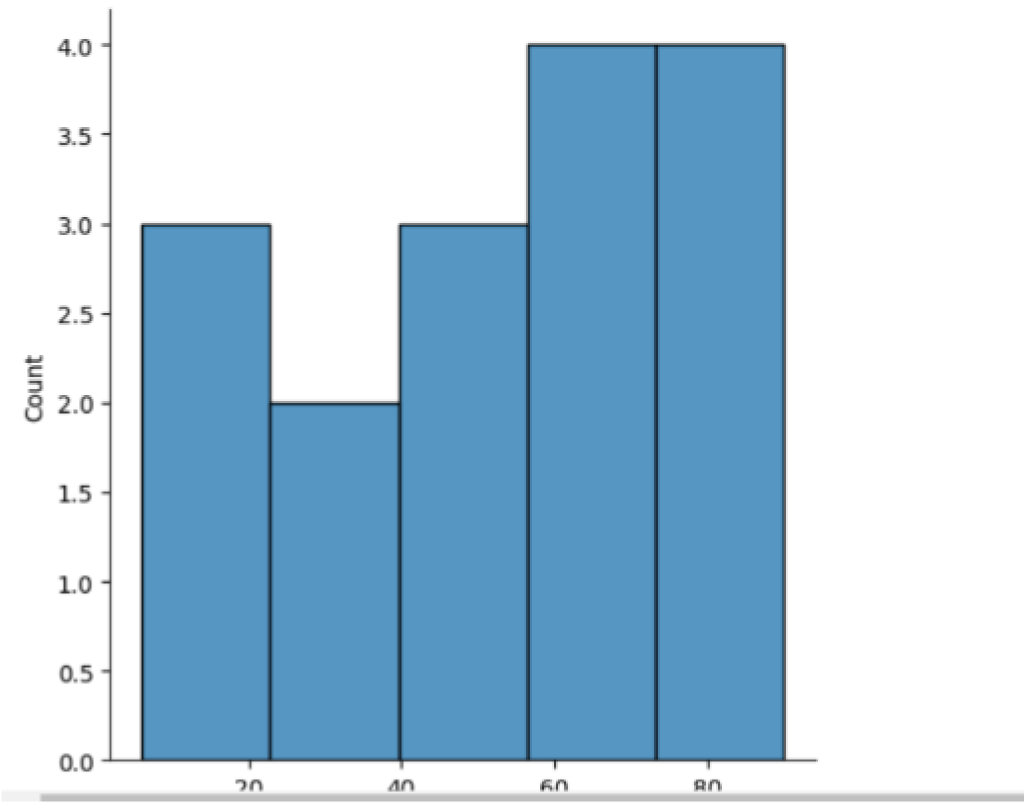
#Exercise: Outlier detection

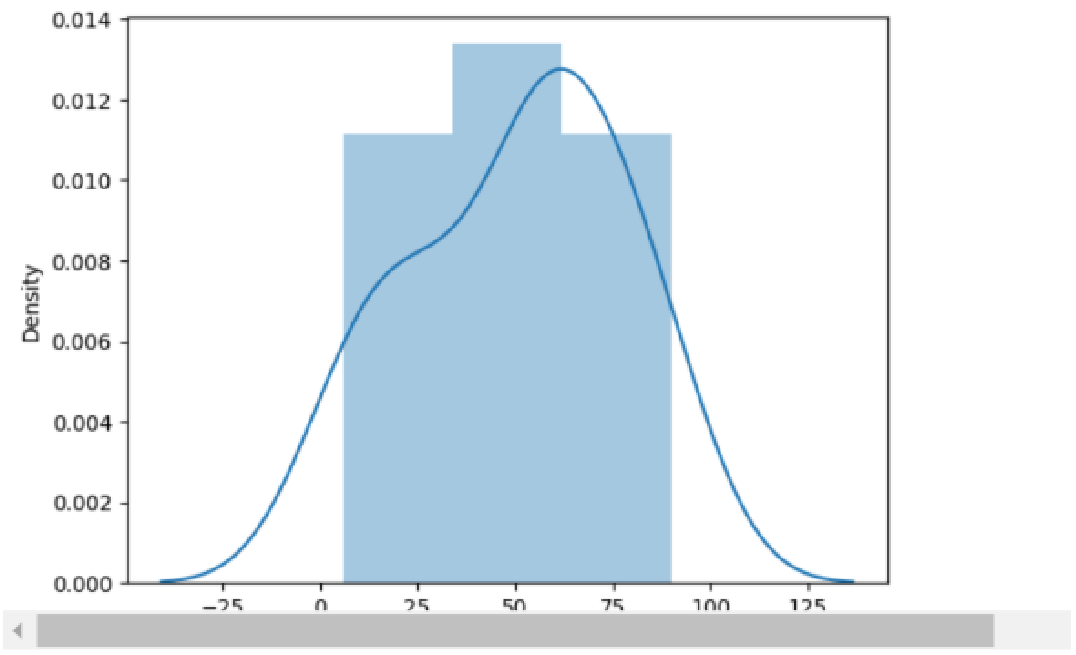
```
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
array = np.random.randint(1, 100, 16)
print(array)
print(array.mean())
print(np.percentile(array, 25))
print(np.percentile(array, 50))
print(np.percentile(array, 75))
print(np.percentile(array, 100))
def outDetection(array):
    sorted_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)
print(f"Lower Range: {lr}, Upper Range: {ur}")

sns.displot(array)
```

```
plt.show()


new_array = array[(array > lr) & (array < ur)]
print("Array after outlier removal:", new_array)



sns.displot(new_array)
plt.show()
lr1, ur1 = outDetection(new_array)
print(f"New Lower Range: {lr1}, New Upper Range: {ur1}")
final_array = new_array[(new_array > lr1) & (new_array < ur1)]
print("Final array after second outlier removal:", final_array)
sns.displot(final_array)
plt.show()
```




0.014
0.012
0.010
0.008
0.006
0.004
0.002
0.000
Density
−25
0
25
50
75
100
125


4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Count
20
40
60
80


Density
0.014
0.012
0.010
0.008
0.006
0.004
0.002
0.000
−25
0
25
50
75
100
125

# Name: Athiena Rachel

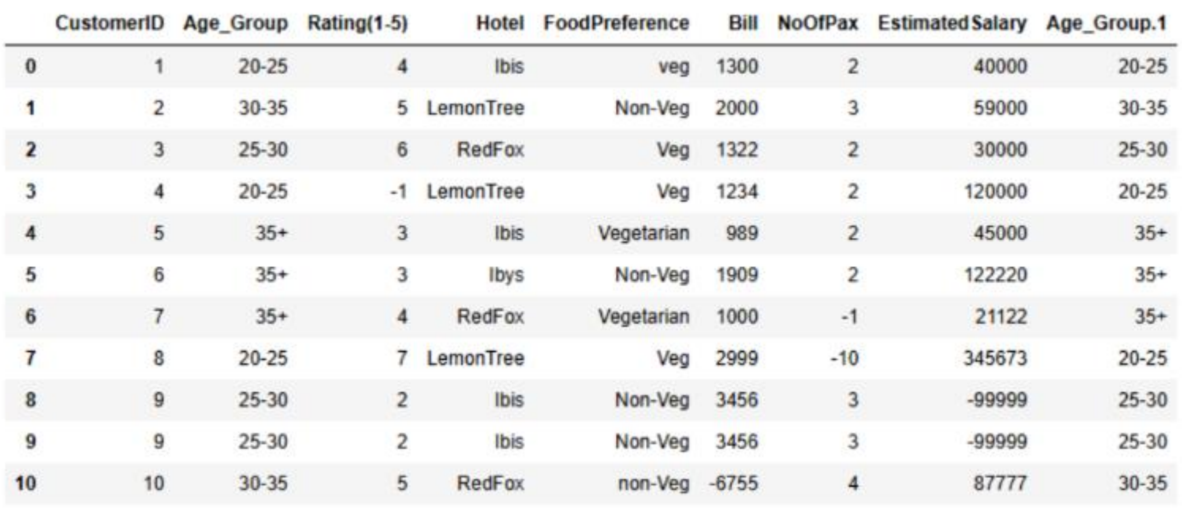
# Roll no.: 230701016

# Date: 27.08.2024

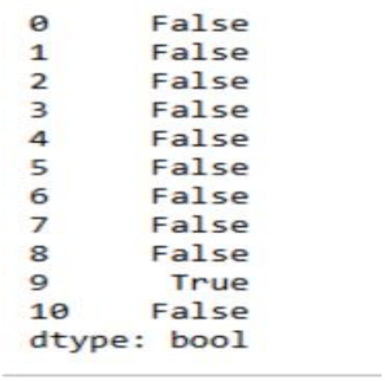
#Exercise: Missing and inappropriate data

```
import numpy as np
import pandas as pd
df = pd.read_csv("Hotel_Dataset.csv")
df
df.duplicated()
df.info()
df.drop_duplicates(inplace=True)
df
len(df)
index = np.array(list(range(0, len(df))))
df.set_index(index, inplace=True)
index

df.drop(['Age_Group.1'], axis=1, inplace=True)
df



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

df['NoOfPax'].loc[(df['NoOfPax'] < 1) | (df['NoOfPax'] > 20)] = np.nan
df
```

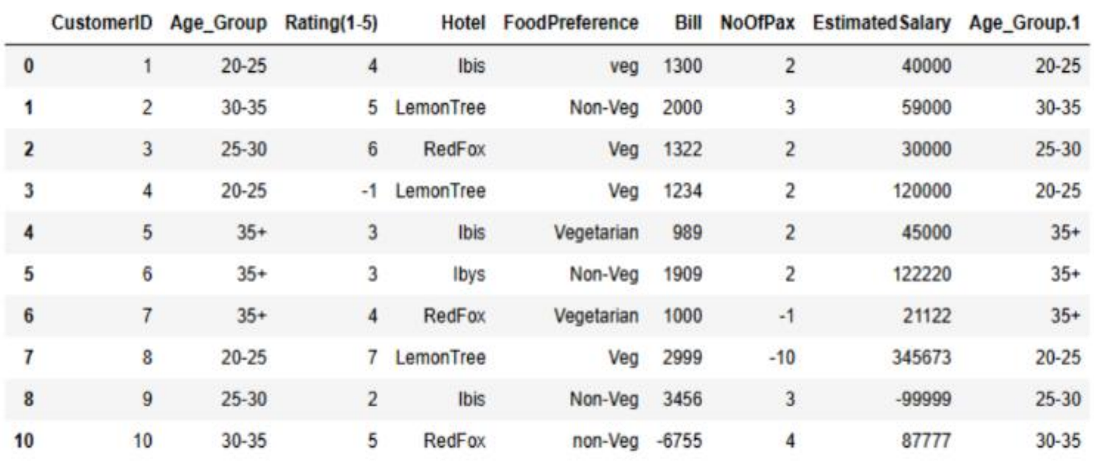
```
df.Age_Group.unique()
df.Hotel.unique()

df.Hotel.replace(['Ibys'], 'Ibis', inplace=True)

df.FoodPreference.unique()

df.FoodPreference.replace(['Vegetarian', 'veg'], 'Veg', inplace=True)
df.FoodPreference.replace(['non-Veg'], 'Non-Veg', inplace=True)

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
```



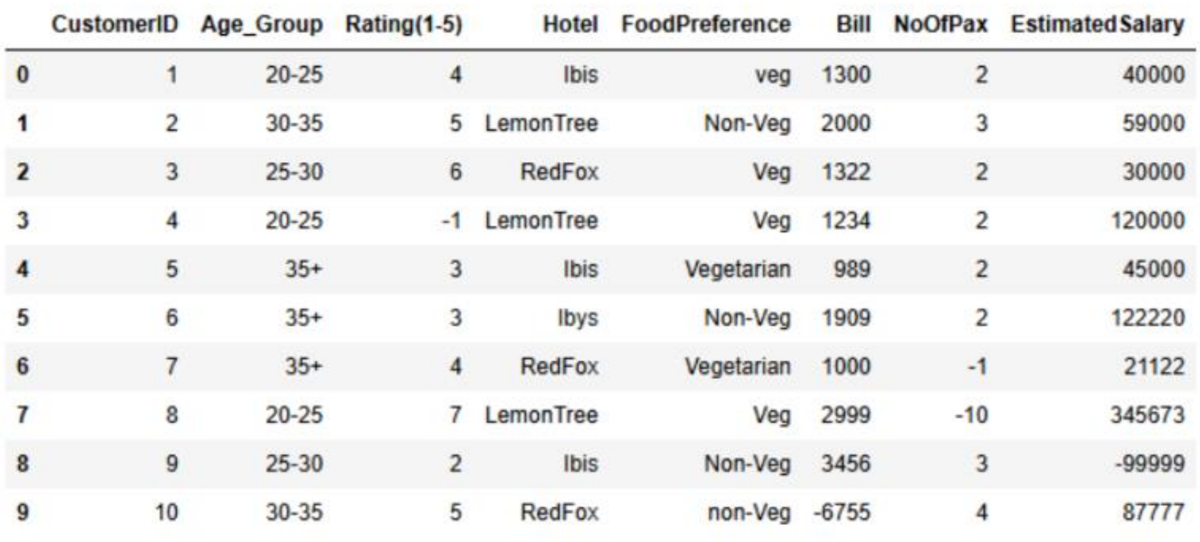
	CustomerID	Age_Group	Rating(1-5)	Hotel	FoodPreference	Bill	NoOfPax	EstimatedSalary	Age_Group.1
0	1	20-25	4	Ibis	veg	1300	2	40000	20-25
1	2	30-35	5	LemonTree	Non-Veg	2000	3	59000	30-35
2	3	25-30	6	RedFox	Veg	1322	2	30000	25-30
3	4	20-25	-1	LemonTree	Veg	1234	2	120000	20-25
4	5	35+	3	Ibis	Vegetarian	989	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



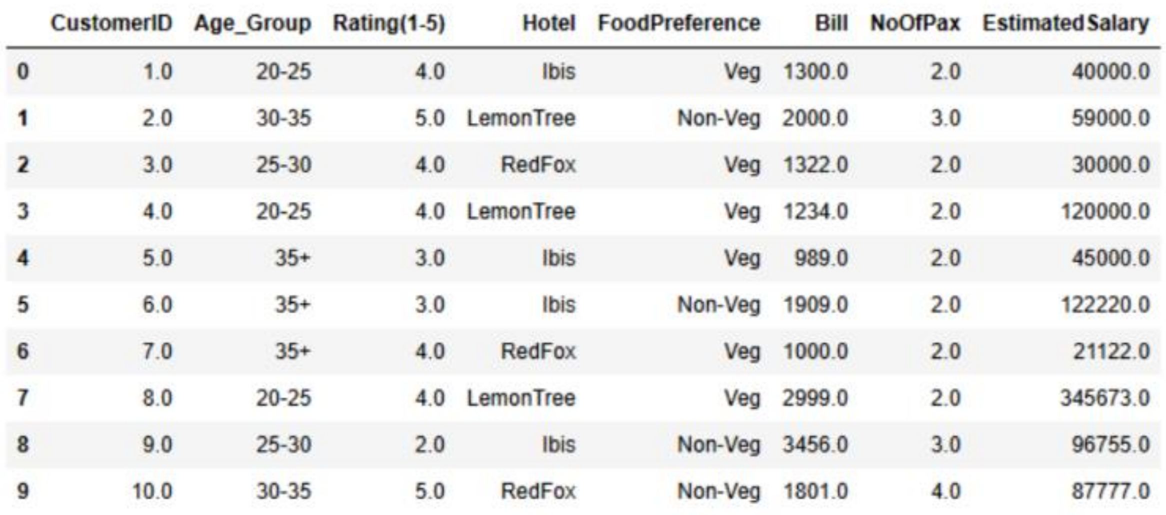
```
0      False
1      False
2      False
3      False
4      False
5      False
6      False
7      False
8      False
9       True
10     False
dtype: bool
```



	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



	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



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

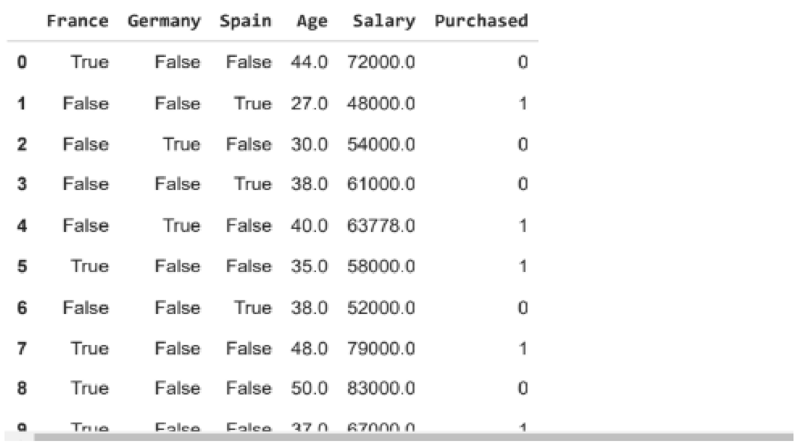
# Name: Athiena Rachel

# Roll no.: 230701016

# Date: 30.08.2024

#Exercise: Data Preprocessing

```
import numpy as np
import pandas as pd

df = pd.read_csv("/content/pre-process_datasample.csv")

df.info()

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)

encoded_countries = pd.get_dummies(df.Country)

updated_dataset = pd.concat([encoded_countries, df.iloc[:, [1, 2, 3]]], axis=1)

updated_dataset.Purchased.replace(['No', 'Yes'], [0, 1], inplace=True)

print(updated_dataset)
```



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

# Name: Athiena Rachel

# Roll no.: 230701016

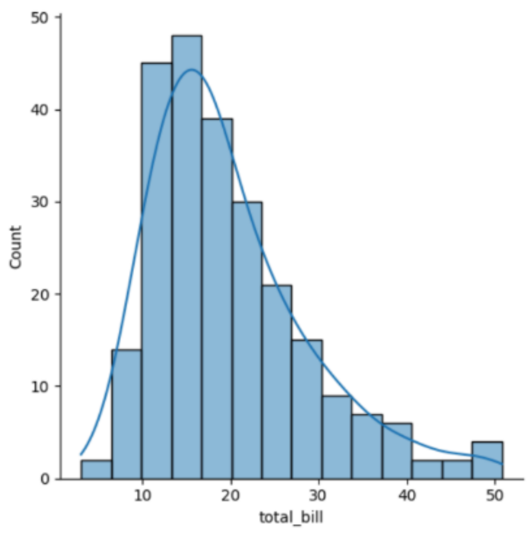
# Date: 03.09.2024

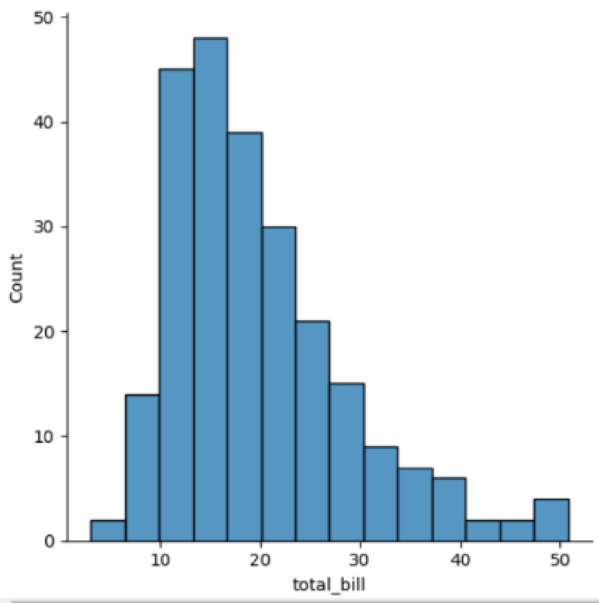
#Exercise: EDA-Quantitative and Qualitative plots

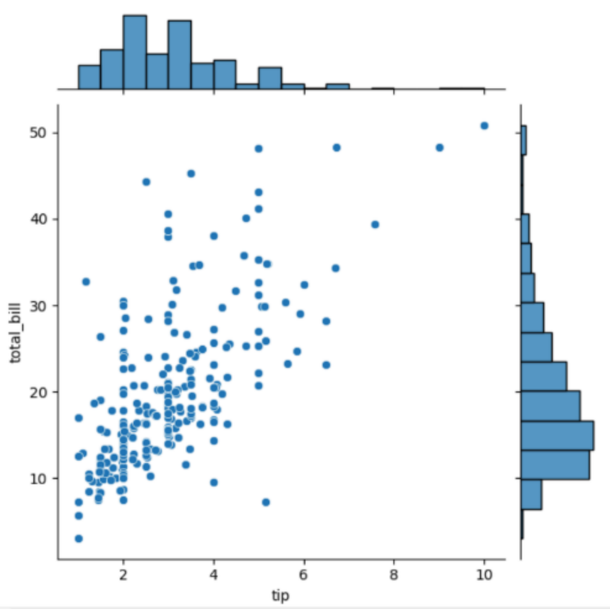
```
import seaborn as sns
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline

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

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

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

sns.jointplot(x=tips.tip, y=tips.total_bill)
plt.show()

sns.jointplot(x=tips.tip, y=tips.total_bill, kind="reg")
plt.show()

sns.jointplot(x=tips.tip, y=tips.total_bill, kind="hex")
plt.show()
```

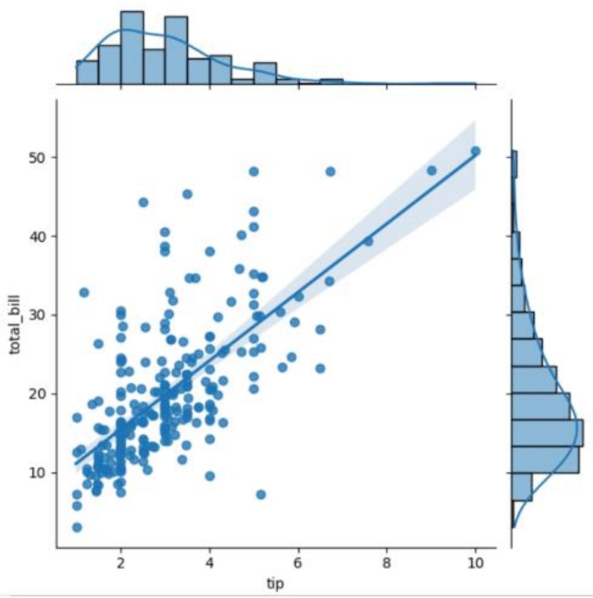
```
sns.pairplot(tips)
plt.show()

print(tips.time.value_counts())

sns.pairplot(tips, hue='time')
plt.show()

sns.pairplot(tips, hue='day')
plt.show()

sns.heatmap(tips.corr(numeric_only=True), annot=True)
plt.show()

sns.boxplot(x=tips.total_bill)
plt.show()
sns.boxplot(x=tips.tip)
plt.show()

sns.countplot(x=tips.day)
plt.show()
sns.countplot(x=tips.sex)
plt.show()

tips.sex.value_counts().plot(kind='pie', autopct='%1.1f%%')
plt.show()

tips.sex.value_counts().plot(kind='bar')
plt.show()
sns.countplot(x=tips[tips.time == 'Dinner']['day'])
```

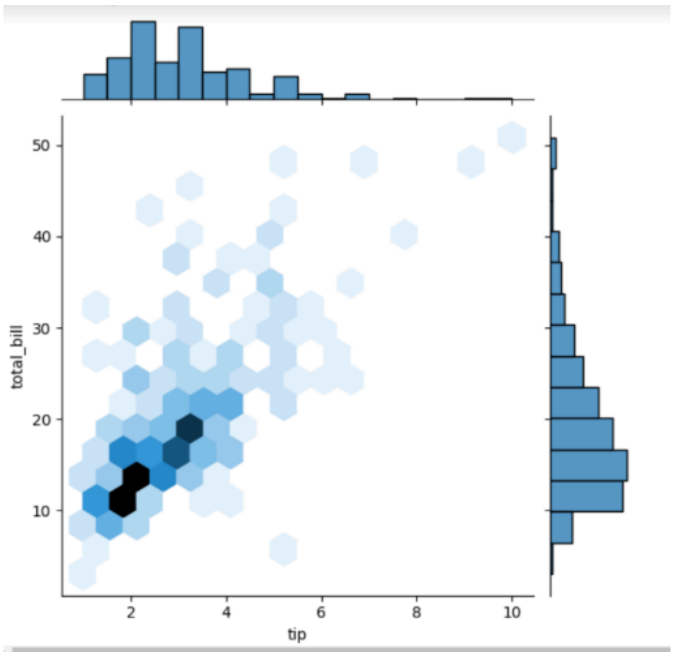
```
plt.show()
```

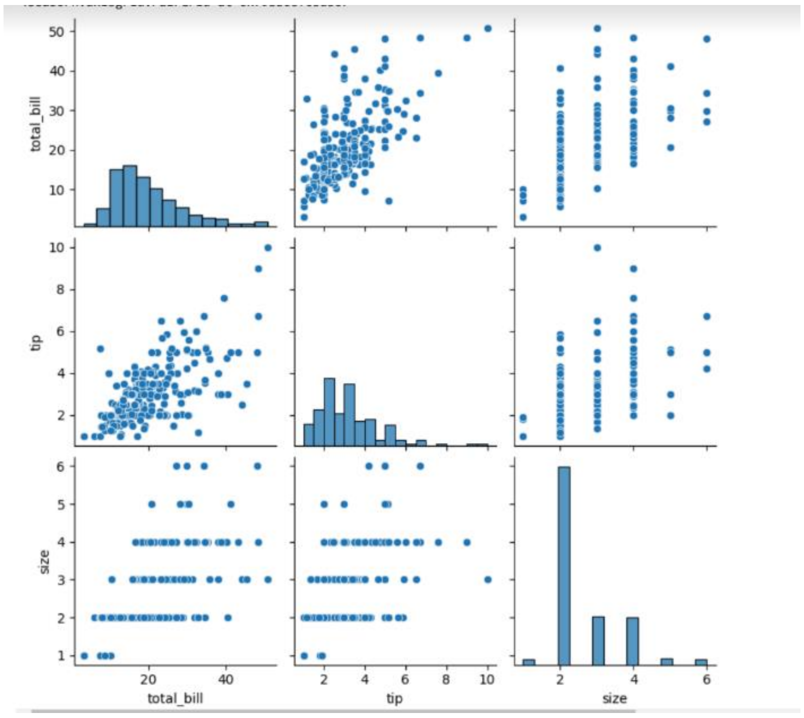

Count
total_bill
0
10
20
30
40
50

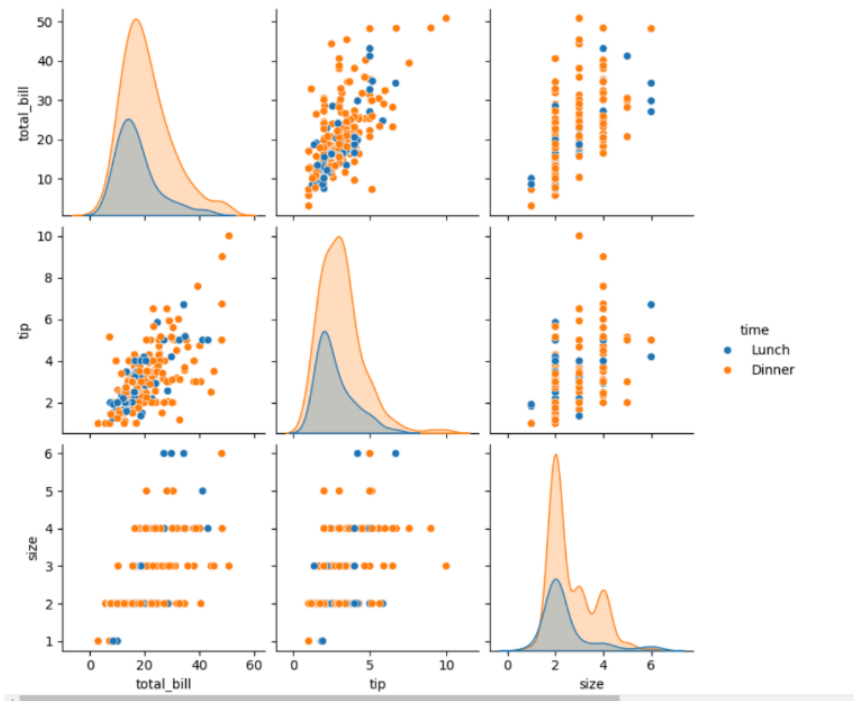

Count
total_bill
0
10
20
30
40
50


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


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


total_bill
tip


total_bill
tip
size


total_bill
tip
size
time
● Lunch
● Dinner

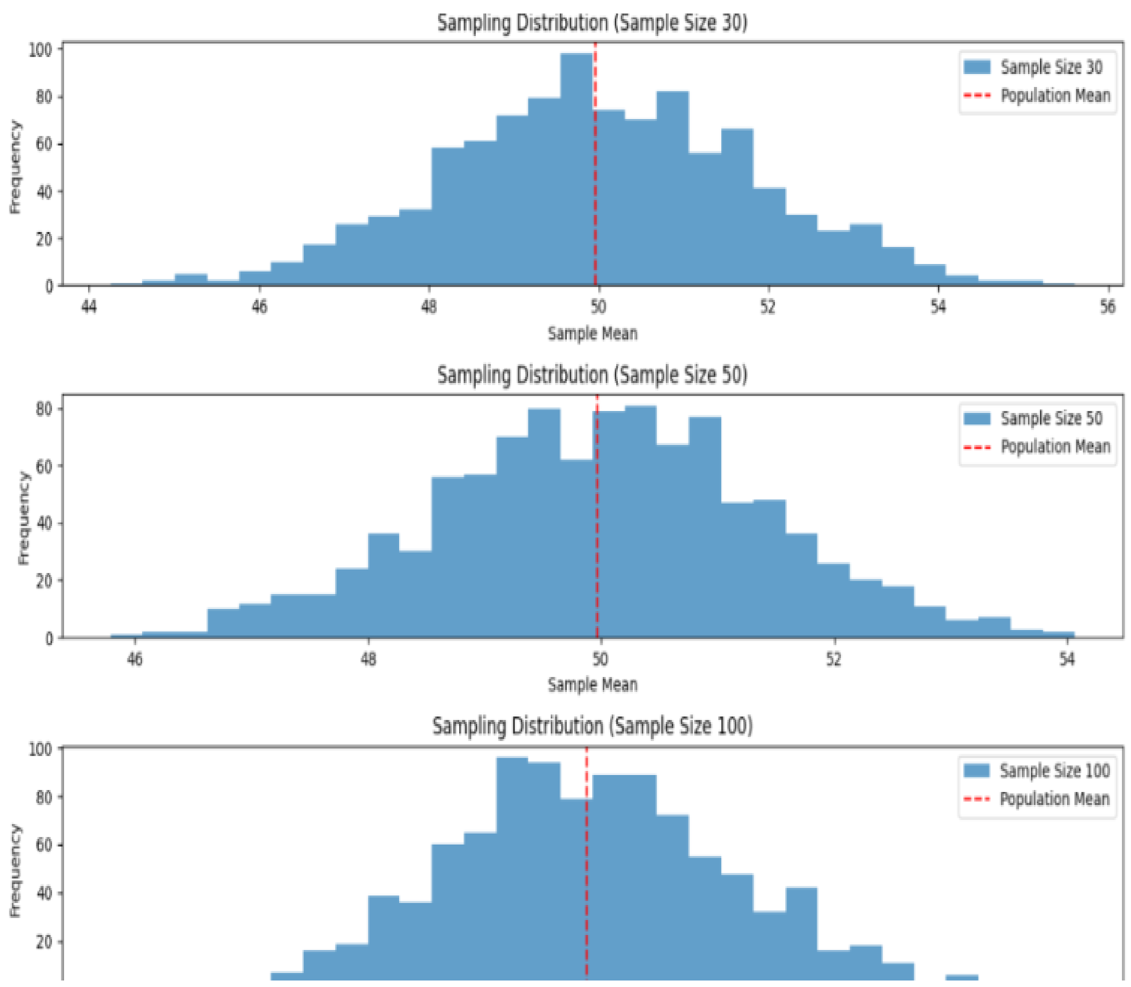
# Name: Athiena Rachel

# Roll no.: 230701016

# Date: 10.09.2024

#Exercise: Random Sampling and Sampling Distribution

```
import numpy as np

import matplotlib.pyplot as plt


population_mean = 50

population_std = 10

population_size = 100000

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



plt.figure(figsize=(8, 5))

plt.hist(population, bins=50, color='skyblue', edgecolor='black', alpha=0.7)

plt.title('Population Distribution')

plt.xlabel('Value')

plt.ylabel('Frequency')

plt.axvline(population_mean, color='red', linestyle='dashed', linewidth=1.5, label='Population
Mean')

plt.legend()

plt.show()



sample_sizes = [30, 50, 100]

num_samples = 1000

sample_means = {}


for size in sample_sizes:

    sample_means[size] = []
    for _ in range(num_samples):
        sample = np.random.choice(population, size=size, replace=False)
        sample_means[size].append(np.mean(sample))



plt.figure(figsize=(12, 8))
for i, size in enumerate(sample_sizes):
    plt.subplot(len(sample_sizes), 1, i + 1)
    plt.hist(sample_means[size], bins=30, alpha=0.7, color='orange', edgecolor='black',
             label=f'Sample Size {size}')
    plt.axvline(np.mean(population), color='red', linestyle='dashed', linewidth=1.5,
label='Population Mean')
    plt.title(f'Sampling Distribution of the Sample Mean (Sample Size {size})')
    plt.xlabel('Sample Mean')
    plt.ylabel('Frequency')
    plt.legend()

plt.tight_layout()
plt.show()

plt.figure(figsize=(12, 8))
for i, size in enumerate(sample_sizes):
    plt.subplot(len(sample_sizes), 1, i + 1)
    plt.hist(sample_means[size], bins=30, alpha=0.7, color='purple', edgecolor='black',
             label=f'Sample Size {size}', density=True)
    plt.axvline(np.mean(population), color='red', linestyle='dashed', linewidth=1.5,
label='Population Mean')
    plt.title(f'Sampling Distribution (Sample Size {size}) - CLT Demonstration')
    plt.xlabel('Sample Mean')
    plt.ylabel('Density')
    plt.legend()
plt.tight_layout()
plt.show()
```



# Name: Athiena Rachel

# Roll no.: 230701016

# Date: 10.09.2024

#Exercise: Z-Test

```
import numpy as np
import scipy.stats as stats

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

population_mean = 150
sample_mean = np.mean(sample_data)
sample_std = np.std(sample_data, ddof=1)

n = len(sample_data)
z_statistic = (sample_mean - population_mean) / (sample_std / np.sqrt(n))
p_value = 2 * (1 - stats.norm.cdf(np.abs(z_statistic)))
print(f"Sample Mean: {sample_mean:.2f}")
print(f"Z-Statistic: {z_statistic:.4f}")
print(f"P-Value: {p_value:.4f}")

alpha = 0.05
if p_value < alpha:
    print("Reject the null hypothesis: The average weight is significantly different from 150 grams.")
else:
    print("Fail to reject the null hypothesis: There is no significant difference in average
weight from 150 grams.")
```

OUTPUT:

```
Sample Mean: 150.20
Z-Statistic: 0.6406
P-Value: 0.5218
Fail to reject the null hypothesis: There is no significant difference in
average weight from 150 grams.
```

# Name: Athiena Rachel

# Roll no.: 230701016

# Date: 08.10.2024

#Exercise: T-Test

```
import numpy as np
import scipy.stats as stats

np.random.seed(42)

sample_size = 25
sample_data = np.random.normal(loc=102, scale=15, size=sample_size)

population_mean = 100

sample_mean = np.mean(sample_data)
sample_std = np.std(sample_data, ddof=1)

n = len(sample_data)

t_statistic, p_value = stats.ttest_1samp(sample_data, population_mean)

print(f"Sample Mean: {sample_mean:.2f}")
print(f"T-Statistic: {t_statistic:.4f}")
print(f"P-Value: {p_value:.4f}")

alpha = 0.05
if p_value < alpha:
    print("Reject the null hypothesis: The average IQ score is significantly different from 
100.")
```

```
else:
    print("Fail to reject the null hypothesis: There is no significant difference in average IQ score from 100.")
```

OUTPUT:

```
Sample Mean: 99.55
T-Statistic: -0.1577
P-Value: 0.8760
Fail to reject the null hypothesis: There is no significant
difference in average IQ score from 100.
```

# Name: Athiena Rachel

# Roll no.: 230701016

# Date: 08.10.2024

#Exercise: ANOVA Test

```
import numpy as np
import scipy.stats as stats
np.random.seed(42)
n_plants = 25
growth_A = np.random.normal(loc=10, scale=2, size=n_plants)
growth_B = np.random.normal(loc=12, scale=3, size=n_plants)
growth_C = np.random.normal(loc=15, scale=2.5, size=n_plants)
f_statistic, p_value = stats.f_oneway(growth_A, growth_B, growth_C)
print("Treatment A Mean Growth:", np.mean(growth_A))
print("Treatment B Mean Growth:", np.mean(growth_B))
print("Treatment C Mean Growth:", np.mean(growth_C))
print()
print(f"F-Statistic: {f_statistic:.4f}")
print(f"P-Value: {p_value:.4f}")
alpha = 0.05
if p_value < alpha:
    print("Reject the null hypothesis: There is a significant difference in mean growth rates
among the three treatments.")
else:
    print("Fail to reject the null hypothesis: There is no significant difference in mean growth
rates among the three treatments.")
if p_value < alpha:
    all_data = np.concatenate([growth_A, growth_B, growth_C])
    treatment_labels = ['A'] * n_plants + ['B'] * n_plants + ['C'] * n_plants

    tukey_results = pairwise_tukeyhsd(all_data, treatment_labels, alpha=0.05)
    print("\nTukey's HSD Post-hoc Test:")
    print(tukey_results)
```

OUTPUT:

```
Treatment A Mean Growth: 9.672983882683818
Treatment B Mean Growth: 11.137680744437432
Treatment C Mean Growth: 15.265234904828972
F-Statistic: 36.1214
P-Value: 0.0000
Reject the null hypothesis: There is a significant difference in mean
growth rates among the three treatments.
```

# Name: Athiena Rachel

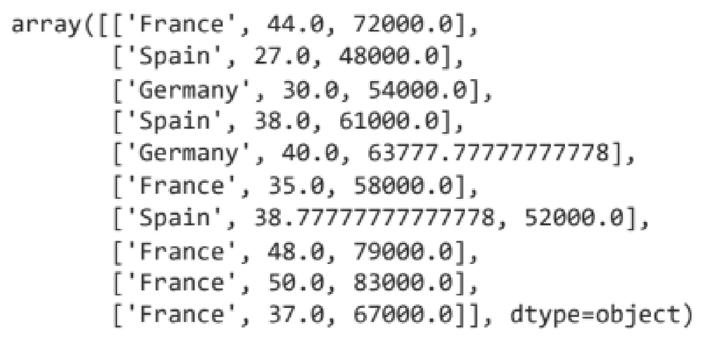
# Roll no.: 230701016

# Date: 22.10.2024

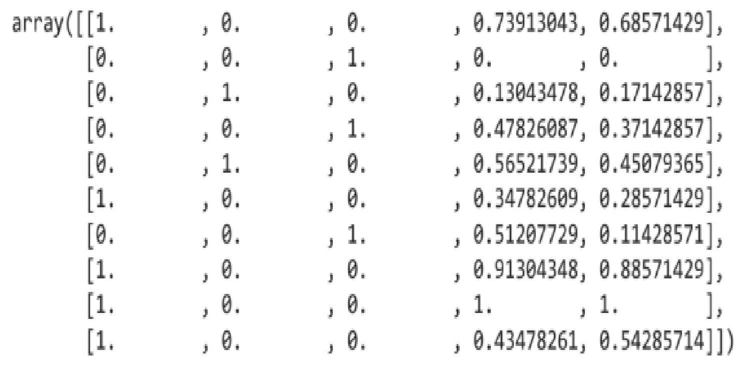
#Exercise: Feature Scaling

```
import numpy as np
import pandas as pd

df = pd.read_csv('/content/pre-process_datasample.csv')

print("Original Data:")
print(df)

df['Country'].fillna(df['Country'].mode()[0], inplace=True)

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

from sklearn.impute import SimpleImputer

age_imputer = SimpleImputer(strategy="mean")
salary_imputer = SimpleImputer(strategy="mean")

age_imputer.fit(features[:, [1]])
salary_imputer.fit(features[:, [2]])

features[:, [1]] = age_imputer.transform(features[:, [1]])
features[:, [2]] = salary_imputer.transform(features[:, [2]])

print("Features after handling missing values:")
print(features)

from sklearn.preprocessing import OneHotEncoder
oh = OneHotEncoder(sparse_output=False)

Country = oh.fit_transform(features[:, [0]])

print("OneHotEncoded 'Country' column:")
print(Country)

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

print("Final dataset with OneHotEncoded 'Country' and other features:")
print(final_set)

from sklearn.preprocessing import StandardScaler
sc = StandardScaler()

sc.fit(final_set)
feat_standard_scaler = sc.transform(final_set)

print("Standardized features:")
print(feat_standard_scaler)

from sklearn.preprocessing import MinMaxScaler
mms = MinMaxScaler(feature_range=(0, 1))

mms.fit(final_set)
feat_minmax_scaler = mms.transform(final_set)
```

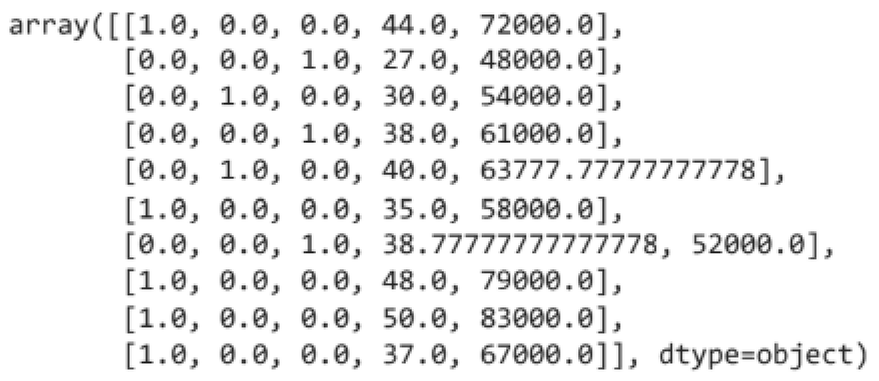
```
print("Normalized features:")

print(feat_minmax_scaler)
```



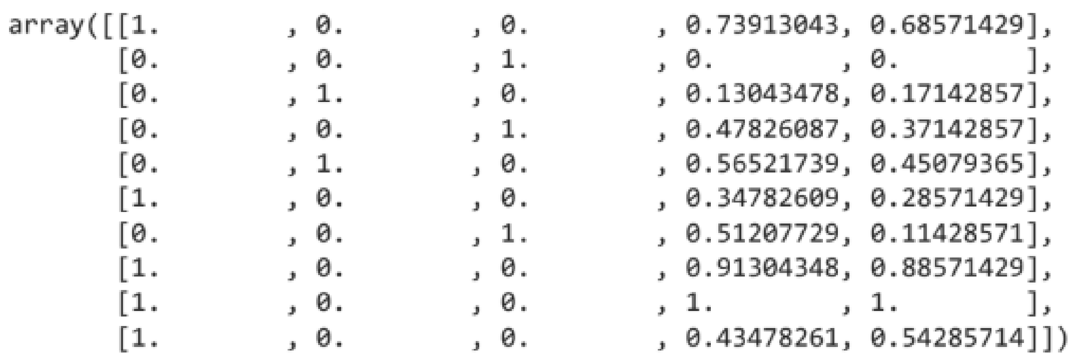
```
array([['France', 44.0, 72000.0],
       ['Spain', 27.0, 48000.0],
       ['Germany', 30.0, 54000.0],
       ['Spain', 38.0, 61000.0],
       ['Germany', 40.0, 63777.77777777778],
       ['France', 35.0, 58000.0],
       ['Spain', 38.77777777777778, 52000.0],
       ['France', 48.0, 79000.0],
  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



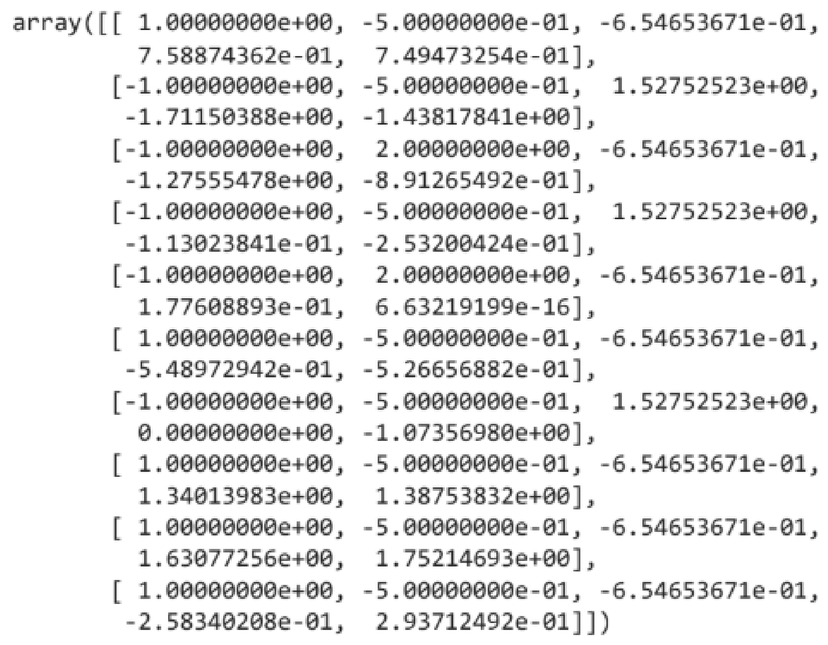
```
array([[1.        , 0.        , 0.        , 0.73913043, 0.68571429],
       [0.        , 0.        , 1.        , 0.        , 0.        ],
       [0.        , 1.        , 0.        , 0.13043478, 0.17142857],
       [0.        , 0.        , 1.        , 0.47826087, 0.37142857],
       [0.        , 1.        , 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],
       [1.        , 0.        , 0.        , 1.        , 1.        ],
       [1.        , 0.        , 0.        , 0.43478261, 0.54285714]])
```



```
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],
       [1.0, 0.0, 0.0, 50.0, 83000.0],
       [1.0, 0.0, 0.0, 37.0, 67000.0]], dtype=object)
```



```
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],
       [1.        , 0.        , 0.        , 1.        , 1.        ],
       [1.        , 0.        , 0.        , 0.43478261, 0.54285714]])
```



```
array([[ 1.00000000e+00, -5.00000000e-01, -6.54653671e-01,
         7.58874362e-01,  7.49473254e-01],
       [-1.00000000e+00, -5.00000000e-01,  1.52752523e+00,
        -1.71150388e+00, -1.43817841e+00],
       [-1.00000000e+00,  2.00000000e+00, -6.54653671e-01,
        -1.27555478e+00, -8.91265492e-01],
       [-1.00000000e+00, -5.00000000e-01,  1.52752523e+00,
        -1.13023841e-01, -2.53200424e-01],
       [-1.00000000e+00,  2.00000000e+00, -6.54653671e-01,
         1.77608893e-01,  6.63219199e-16],
       [ 1.00000000e+00, -5.00000000e-01, -6.54653671e-01,
        -5.48972942e-01, -5.26656882e-01],
       [-1.00000000e+00, -5.00000000e-01,  1.52752523e+00,
         0.00000000e+00, -1.07356980e+00],
       [ 1.00000000e+00, -5.00000000e-01, -6.54653671e-01,
         1.34013983e+00,  1.38753832e+00],
       [ 1.00000000e+00, -5.00000000e-01, -6.54653671e-01,
         1.63077256e+00,  1.75214693e+00],
       [ 1.00000000e+00, -5.00000000e-01, -6.54653671e-01,
        -2.58340208e-01,  2.93712492e-01]])
```

# Name: Athiena Rachel

# Roll no.: 230701016

# Date: 29.10.2024

#Exercise: Linear Regression

```
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,random_state=23)

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,Salary)
```

OUTPUT-

```
Estimated Salary for 44.0 years of experience is [[435544.30953887]]:
```

# Name: Athiena Rachel

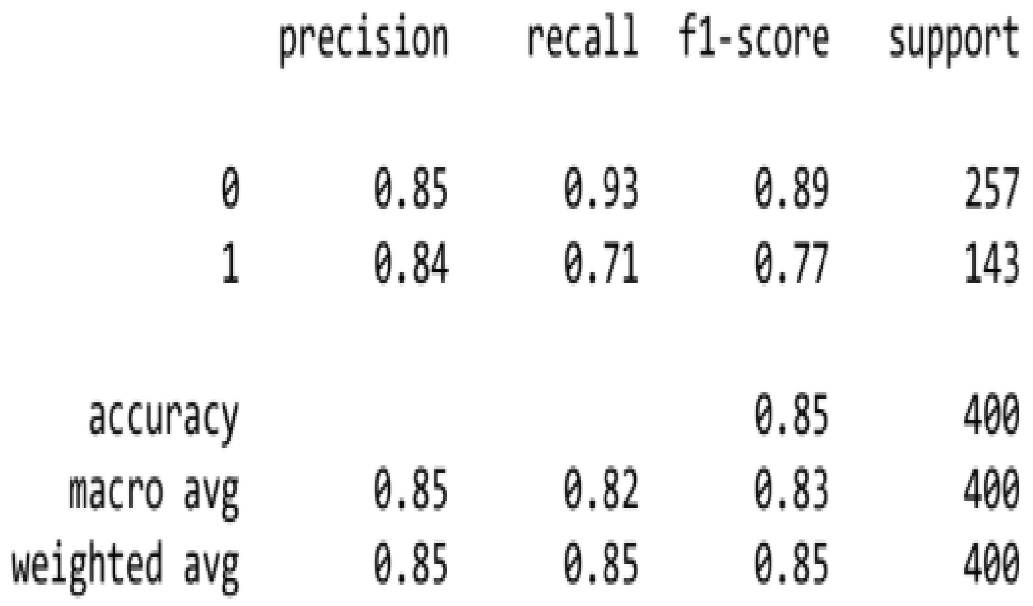
# Roll no.:05.11.2024

# Date: 29.10.2024

#Exercise: Logistic Regression

```
import numpy as np
import pandas as pd
df=pd.read_csv('Social_Network_Ads.csv')
df
df.head()

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

from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression

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 {} Train{} 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=314)
finalModel=LogisticRegression()
```

```
finalModel.fit(x_train,y_train)


print(finalModel.score(x_train,y_train))
print(finalModel.score(x_test,y_test))


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       0.84      0.71      0.77       143

    accuracy                           0.85       400
   macro avg       0.85      0.82      0.83       400
weighted avg       0.85      0.85      0.85       400
```