Neural Network Deep Learning (23442)

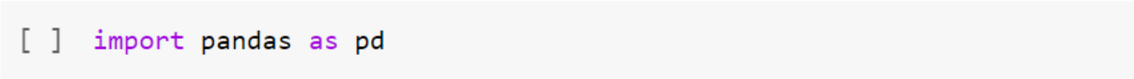
Assignment_4

https://github.com/niteesh0301/Assignment-_4.git

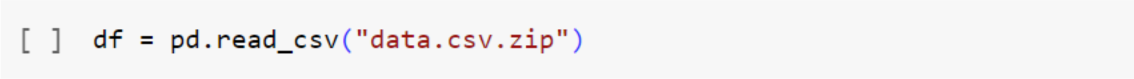
1 Question :- Data Manipulation

1(a) :- Read the provided CSV file ‘data.csv’.

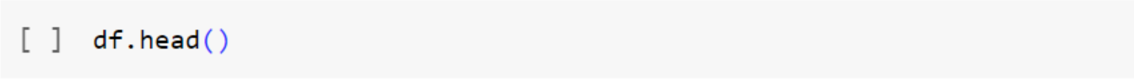
Code:-



```
[ ]  import pandas as pd
```

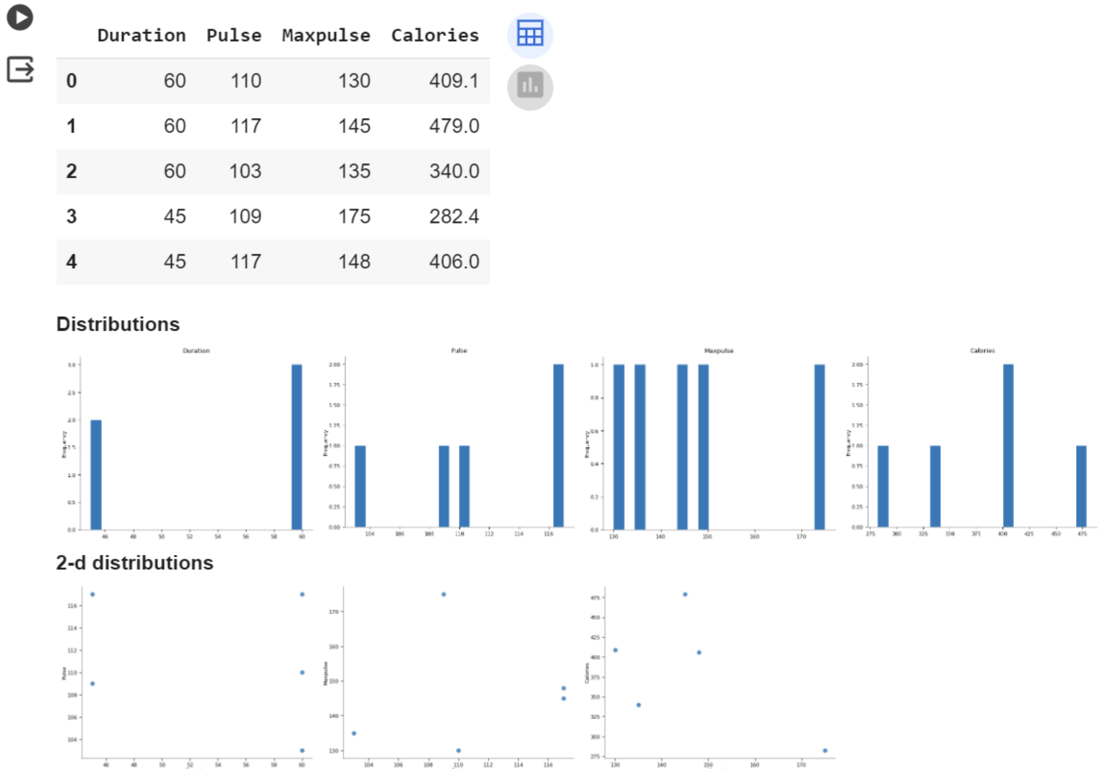


```
[ ]  df = pd.read_csv("data.csv.zip")
```



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

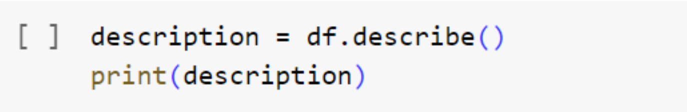
Output:-



	Duration	Pulse	Maxpulse	Calories
0	60	110	130	409.1
1	60	117	145	479.0
2	60	103	135	340.0
3	45	109	175	282.4
4	45	117	148	406.0

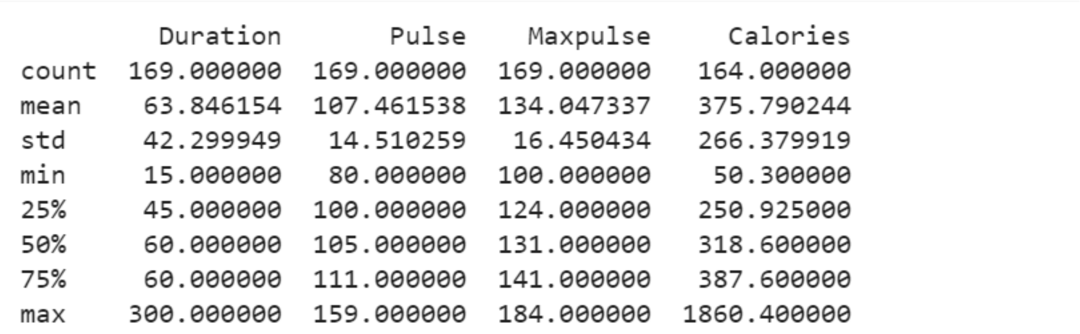
1(b) :- Show the basic statistical description about the data.

Code:-



```
[ ]  description = df.describe()
     print(description)
```

Output:-

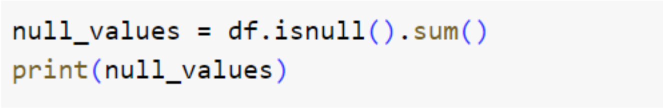


```
          Duration       Pulse    Maxpulse     Calories
count   169.000000  169.000000  169.000000   164.000000
mean     63.846154  107.461538  134.047337   375.790244
std      42.299949   14.510259   16.450434   266.379919
min      15.000000   80.000000  100.000000    50.300000
25%      45.000000  100.000000  124.000000   250.925000
50%      60.000000  105.000000  131.000000   318.600000
75%      60.000000  111.000000  141.000000   387.600000
max     300.000000  159.000000  184.000000  1860.400000
```

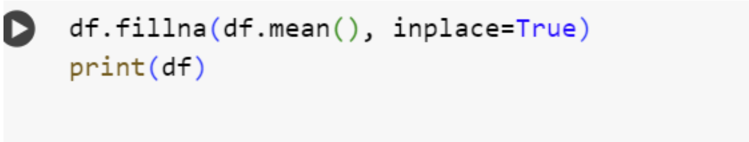
1(c) :- Check if the data has null values.

i. Replace the null values with the mean

Code:-

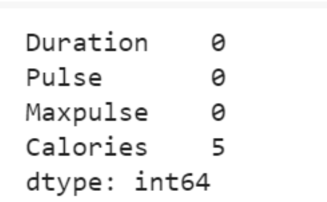


```
null_values = df.isnull().sum()
print(null_values)
```

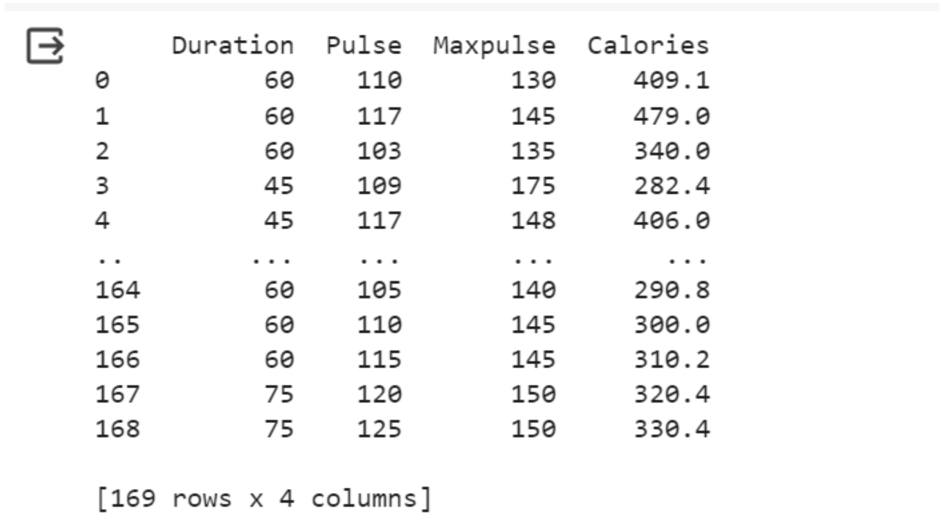


```
df.fillna(df.mean(), inplace=True)
print(df)
```

Output:-



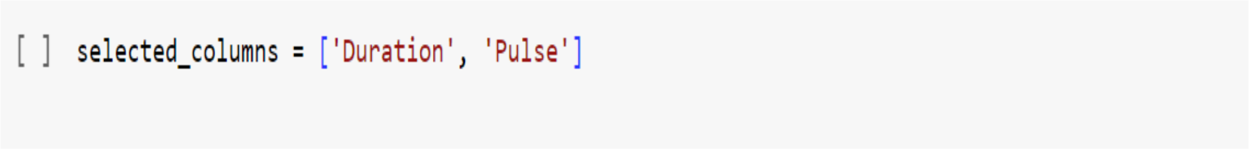
```
Duration    0
Pulse       0
Maxpulse    0
Calories    5
dtype: int64
```



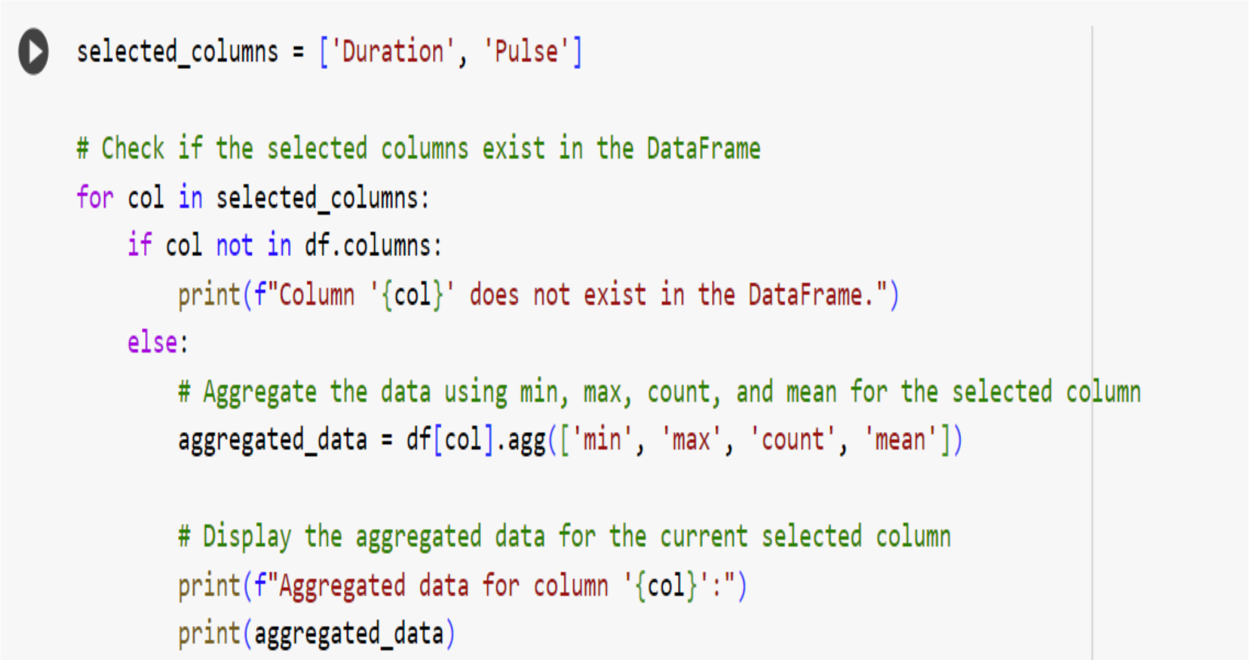
```
     Duration  Pulse  Maxpulse  Calories
0          60    110       130     409.1
1          60    117       145     479.0
2          60    103       135     340.0
3          45    109       175     282.4
4          45    117       148     406.0
..        ...    ...       ...       ...
164        60    105       140     290.8
165        60    110       145     300.0
166        60    115       145     310.2
167        75    120       150     320.4
168        75    125       150     330.4

[169 rows x 4 columns]
```

1(d) :- Select at least two columns and aggregate the data using: min, max, count, mean.

Code:-

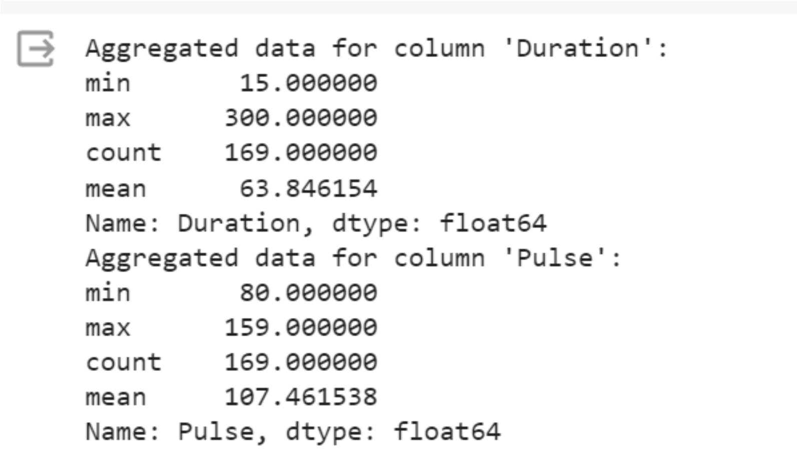


```
selected_columns = ['Duration', 'Pulse']
```



```
selected_columns = ['Duration', 'Pulse']

# Check if the selected columns exist in the DataFrame
for col in selected_columns:
    if col not in df.columns:
        print(f"Column '{col}' does not exist in the DataFrame.")
    else:
        # Aggregate the data using min, max, count, and mean for the selected column
        aggregated_data = df[col].agg(['min', 'max', 'count', 'mean'])

        # Display the aggregated data for the current selected column
        print(f"Aggregated data for column '{col}':")
        print(aggregated_data)
```

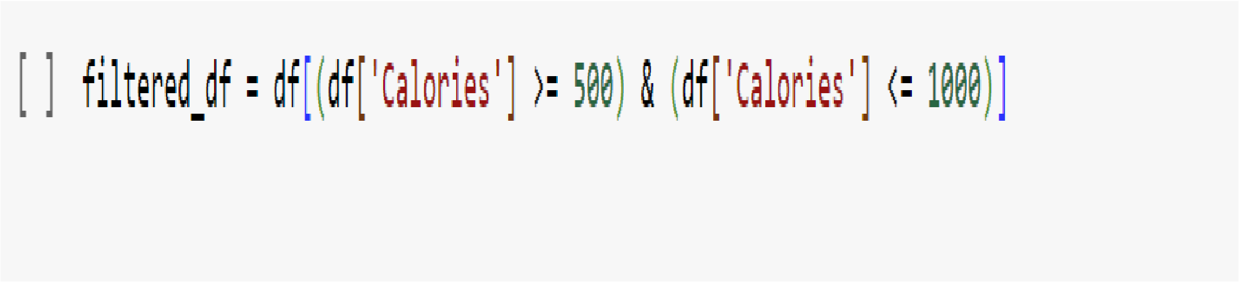
Output:-



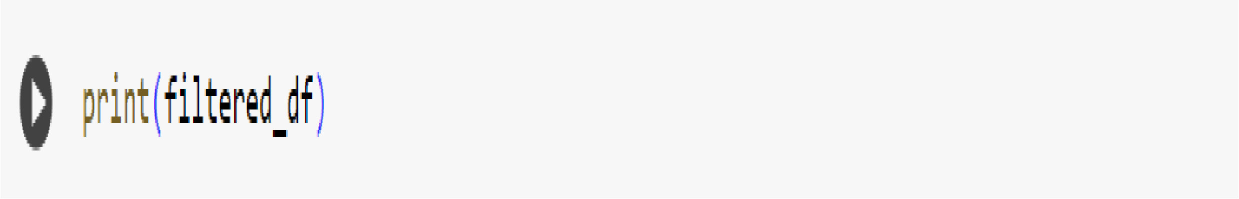
```
Aggregated data for column 'Duration':
min       15.000000
max      300.000000
count    169.000000
mean      63.846154
Name: Duration, dtype: float64
Aggregated data for column 'Pulse':
min       80.000000
max      159.000000
count    169.000000
mean     107.461538
Name: Pulse, dtype: float64
```

1(e) :- Filter the dataframe to select the rows with calories values between 500 and 1000.

Code:-

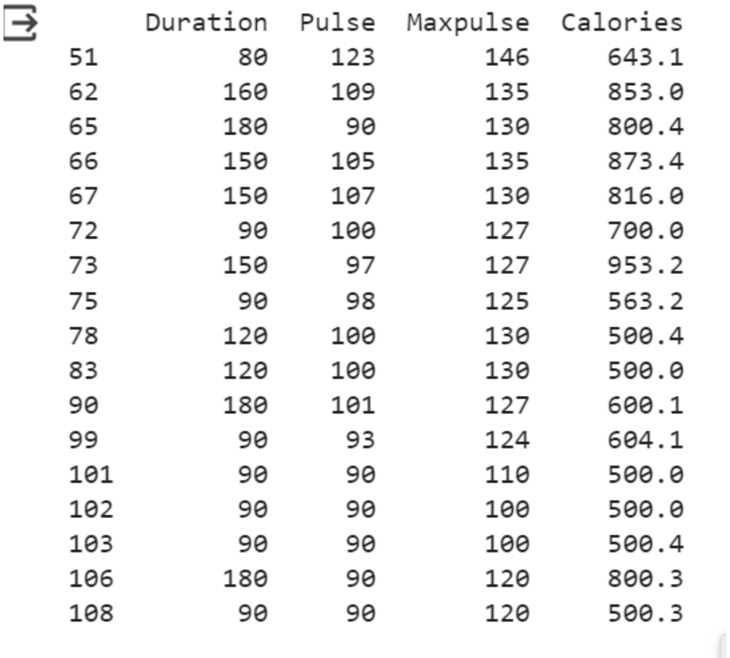


```
[ ] filtered_df = df[(df['Calories'] >= 500) & (df['Calories'] <= 1000)]
```



```
print(filtered_df)
```

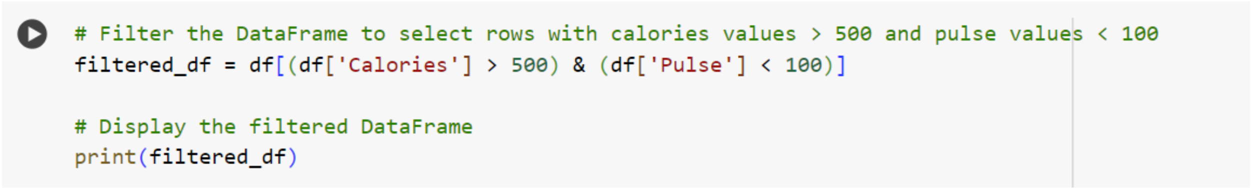
Output:-



```
     Duration  Pulse  Maxpulse  Calories
51         80    123       146     643.1
62        160    109       135     853.0
65        180     90       130     800.4
66        150    105       135     873.4
67        150    107       130     816.0
72         90    100       127     700.0
73        150     97       127     953.2
75         90     98       125     563.2
78        120    100       130     500.4
83        120    100       130     500.0
90        180    101       127     600.1
99         90     93       124     604.1
101        90     90       110     500.0
102        90     90       100     500.0
103        90     90       100     500.4
106       180     90       120     800.3
108        90     90       120     500.3
```

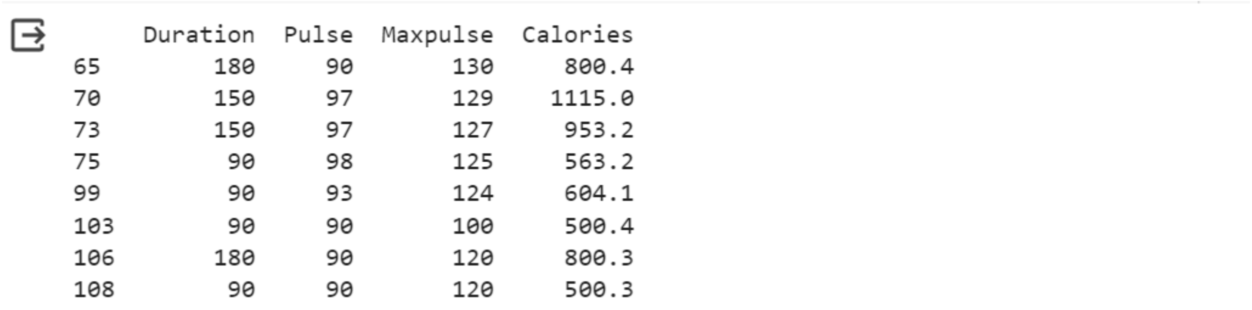
1(e) :- Filter the dataframe to select the rows with calories values > 500 and pulse < 100.

Code:-



```python
# Filter the DataFrame to select rows with calories values > 500 and pulse values < 100
filtered_df = df[(df['Calories'] > 500) & (df['Pulse'] < 100)]

# Display the filtered DataFrame
print(filtered_df)
```

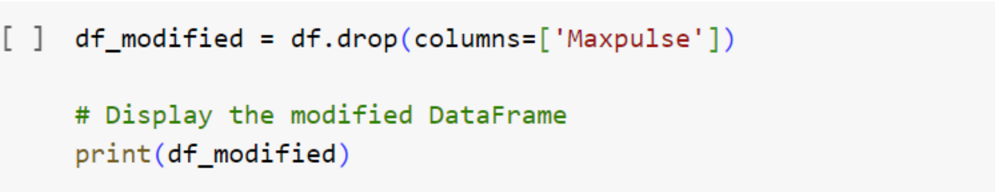
Output:-



```
     Duration  Pulse  Maxpulse  Calories
65        180     90       130     800.4
70        150     97       129    1115.0
73        150     97       127     953.2
75         90     98       125     563.2
99         90     93       124     604.1
103        90     90       100     500.4
106       180     90       120     800.3
108        90     90       120     500.3
```

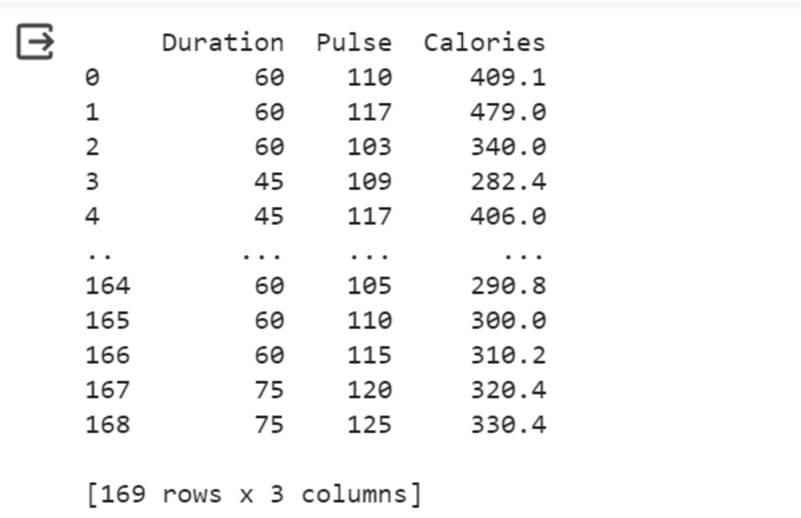
1(f) :- Create a new “df_modified” dataframe that contains all the columns from df except for “Maxpulse”.

Code:-



```
[ ]  df_modified = df.drop(columns=['Maxpulse'])

     # Display the modified DataFrame
     print(df_modified)
```

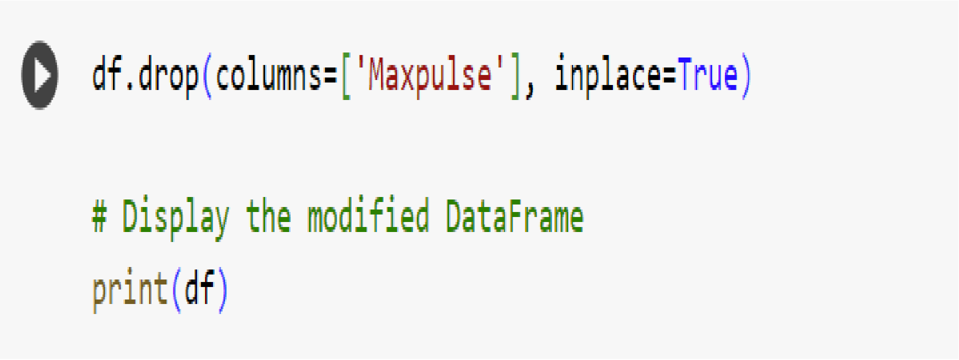
Output:-



```
     Duration  Pulse  Calories
0          60    110     409.1
1          60    117     479.0
2          60    103     340.0
3          45    109     282.4
4          45    117     406.0
..        ...    ...       ...
164        60    105     290.8
165        60    110     300.0
166        60    115     310.2
167        75    120     320.4
168        75    125     330.4

[169 rows x 3 columns]
```

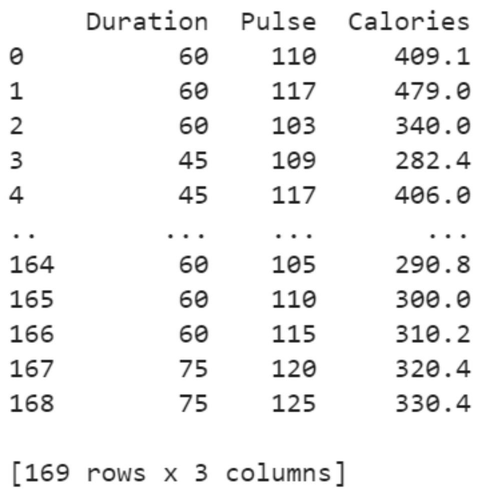
1(g) :- Delete the “Maxpulse” column from the main df dataframe

Code:-



```
df.drop(columns=['Maxpulse'], inplace=True)

# Display the modified DataFrame
print(df)
```

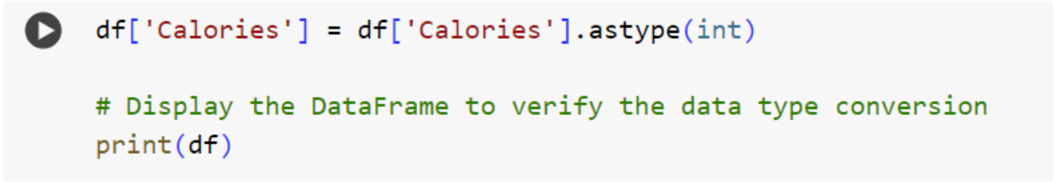
Output:-



```
     Duration  Pulse  Calories
0          60    110     409.1
1          60    117     479.0
2          60    103     340.0
3          45    109     282.4
4          45    117     406.0
..        ...    ...       ...
164        60    105     290.8
165        60    110     300.0
166        60    115     310.2
167        75    120     320.4
168        75    125     330.4

[169 rows x 3 columns]
```

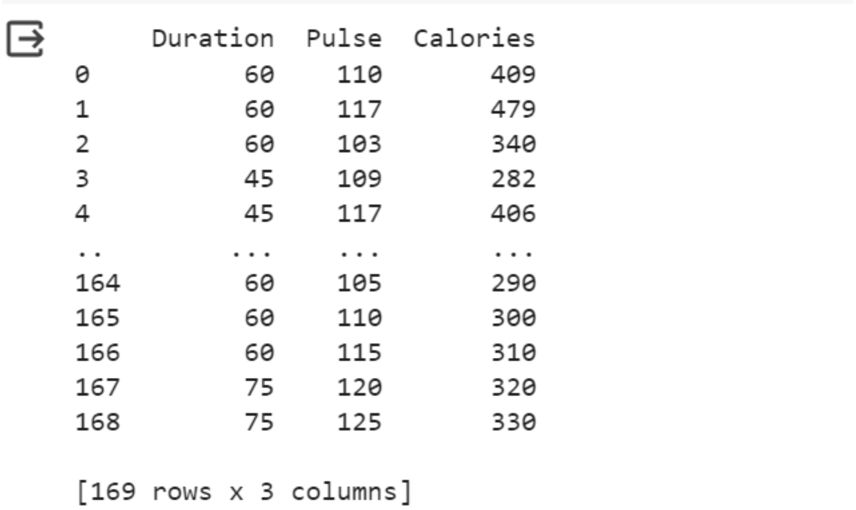
1(h) :- Convert the datatype of Calories column to int datatype.

Code:-



```
df['Calories'] = df['Calories'].astype(int)

# Display the DataFrame to verify the data type conversion
print(df)
```

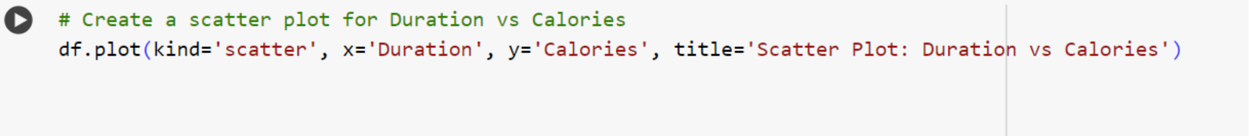
Output:-



```
     Duration  Pulse  Calories
0          60    110       409
1          60    117       479
2          60    103       340
3          45    109       282
4          45    117       406
..        ...    ...       ...
164        60    105       290
165        60    110       300
166        60    115       310
167        75    120       320
168        75    125       330

[169 rows x 3 columns]
```

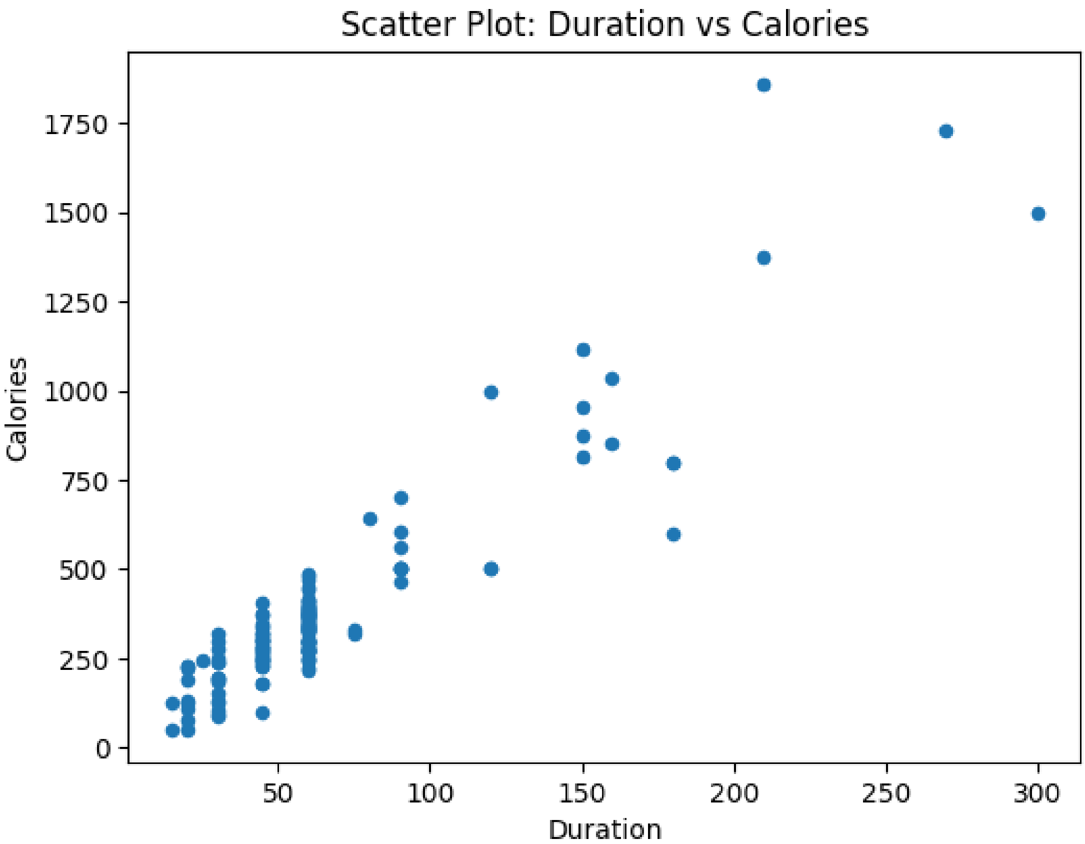
1(i) :- Using pandas create a scatter plot for the two columns (Duration and Calories).

Code:-



```
# Create a scatter plot for Duration vs Calories
df.plot(kind='scatter', x='Duration', y='Calories', title='Scatter Plot: Duration vs Calories')
```

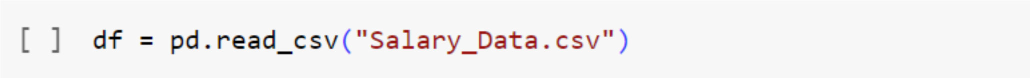
Output:-



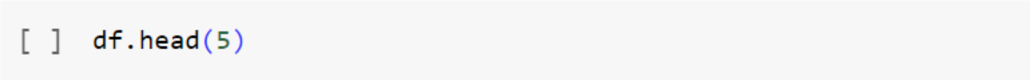
2 Question :- Linear Regression

2(a) :- Import the given “Salary_Data.csv”

Code:-

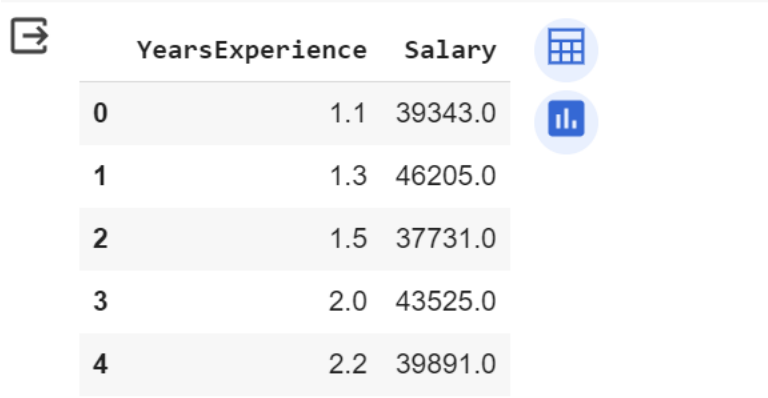


```
df = pd.read_csv("Salary_Data.csv")
```



```
df.head(5)
```

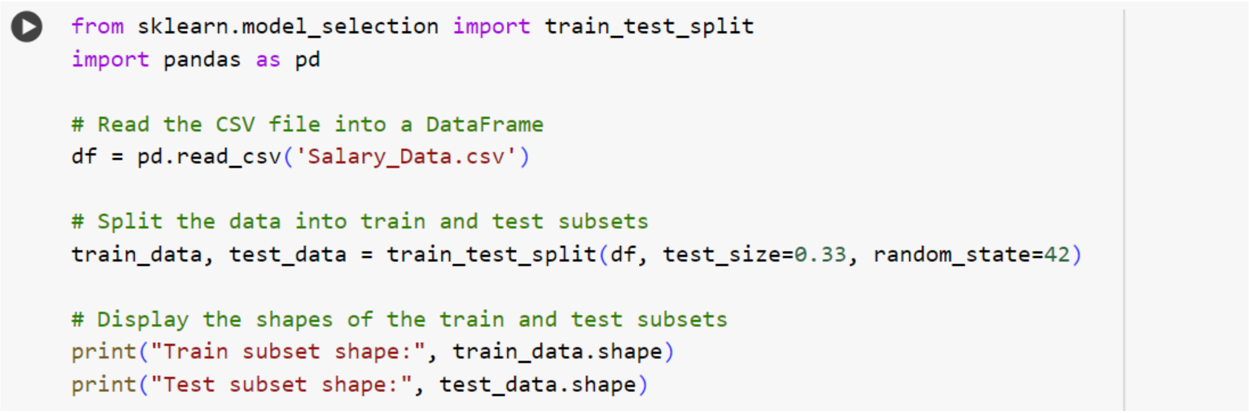
Output:-



	YearsExperience	Salary
0	1.1	39343.0
1	1.3	46205.0
2	1.5	37731.0
3	2.0	43525.0
4	2.2	39891.0

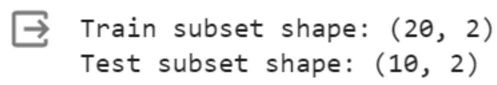
2(b) :- Split the data in train_test partitions, such that 1/3 of the data is reserved as test subset.

Code:-



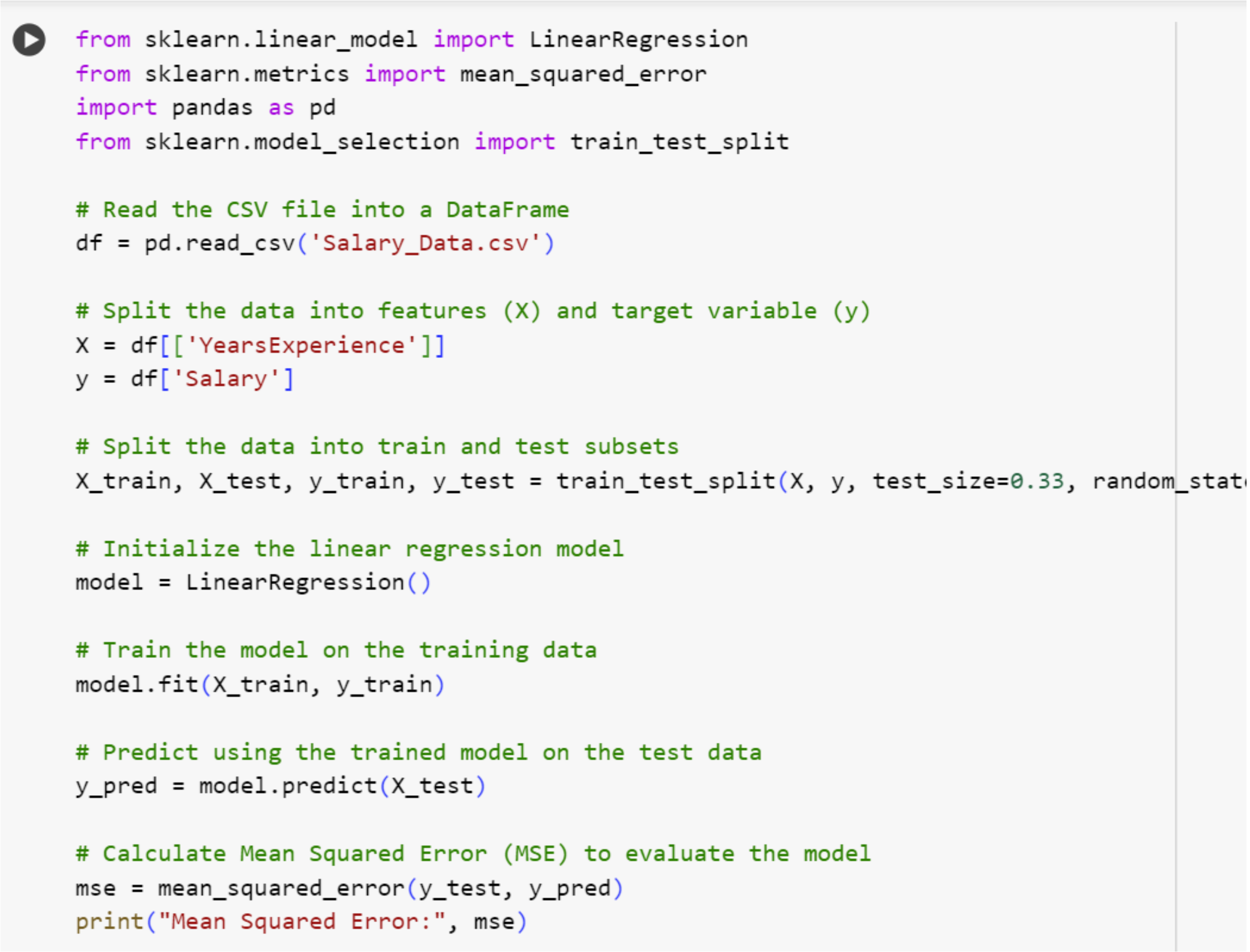
```
from sklearn.model_selection import train_test_split
import pandas as pd

# Read the CSV file into a DataFrame
df = pd.read_csv('Salary_Data.csv')

# Split the data into train and test subsets
train_data, test_data = train_test_split(df, test_size=0.33, random_state=42)

# Display the shapes of the train and test subsets
print("Train subset shape:", train_data.shape)
print("Test subset shape:", test_data.shape)
```

Output:-



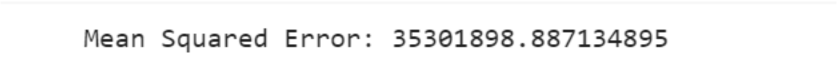
```
Train subset shape: (20, 2)
Test subset shape: (10, 2)
```

2(c) **:-** Train and predict the model.



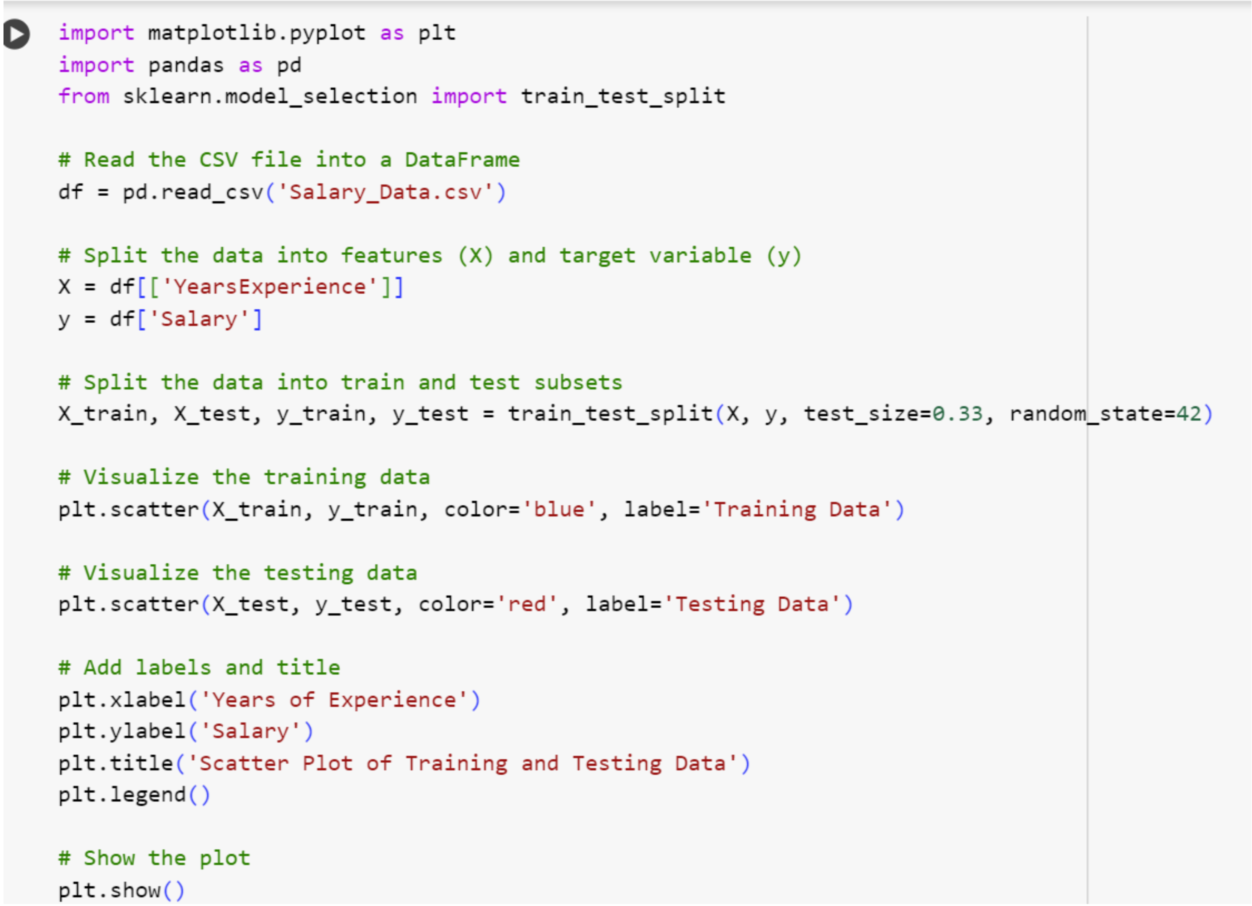
```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
import pandas as pd
from sklearn.model_selection import train_test_split

# Read the CSV file into a DataFrame
df = pd.read_csv('Salary_Data.csv')

# Split the data into features (X) and target variable (y)
X = df[['YearsExperience']]
y = df['Salary']

# Split the data into train and test subsets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_stat

# Initialize the linear regression model
model = LinearRegression()

# Train the model on the training data
model.fit(X_train, y_train)

# Predict using the trained model on the test data
y_pred = model.predict(X_test)

# Calculate Mean Squared Error (MSE) to evaluate the model
mse = mean_squared_error(y_test, y_pred)
print("Mean Squared Error:", mse)
```

Output:-



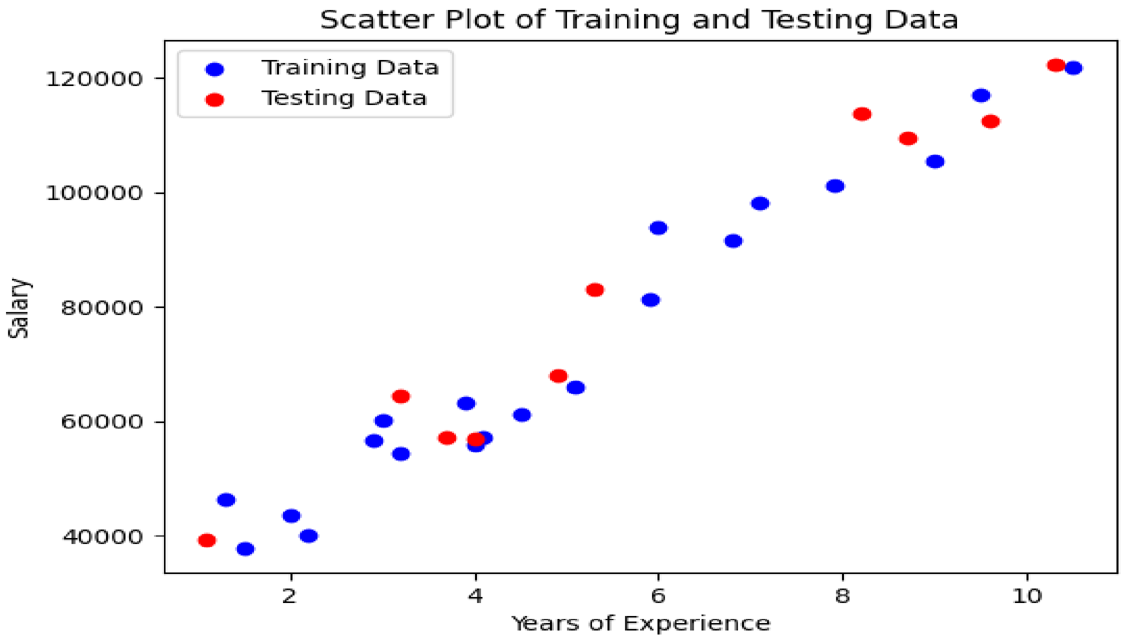
```
Mean Squared Error: 35301898.887134895
```

2(d) :- Visualize both train and test data using scatter plot.



```
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.model_selection import train_test_split

# Read the CSV file into a DataFrame
df = pd.read_csv('Salary_Data.csv')

# Split the data into features (X) and target variable (y)
X = df[['YearsExperience']]
y = df['Salary']

# Split the data into train and test subsets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=42)

# Visualize the training data
plt.scatter(X_train, y_train, color='blue', label='Training Data')

# Visualize the testing data
plt.scatter(X_test, y_test, color='red', label='Testing Data')

# Add labels and title
plt.xlabel('Years of Experience')
plt.ylabel('Salary')
plt.title('Scatter Plot of Training and Testing Data')
plt.legend()

# Show the plot
plt.show()
```

Output:-


Scatter Plot of Training and Testing Data
Training Data
Testing Data
Salary
Years of Experience