ICP Assignment-4

Nirmala Yarlagadda
700733102

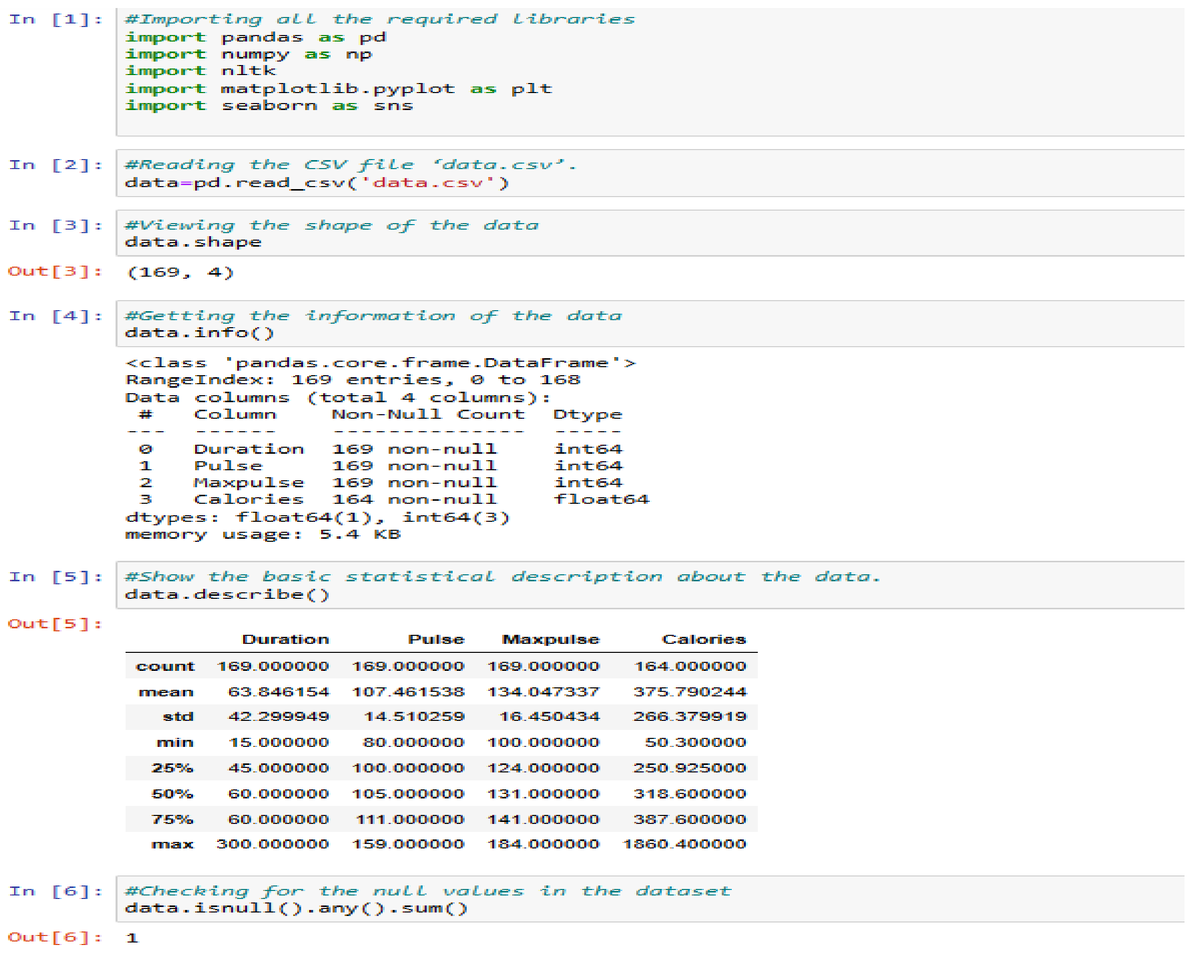
GitHub link: https://github.com/niryarjessy22/ICP-4.git

Videos link:

https://drive.google.com/file/d/1sAZAFFEnqPxI0XqLx-QSkQIXBEpQAHSG/view?usp=share_link

1. Data Manipulation

a. Read the provided CSV file 'data.csv'.



```
In [1]: #Importing all the required libraries
        import pandas as pd
        import numpy as np
        import nltk
        import matplotlib.pyplot as plt
        import seaborn as sns
```

```
In [2]: #Reading the CSV file 'data.csv'.
        data=pd.read_csv('data.csv')
```

```
In [3]: #Viewing the shape of the data
        data.shape

Out[3]: (169, 4)
```

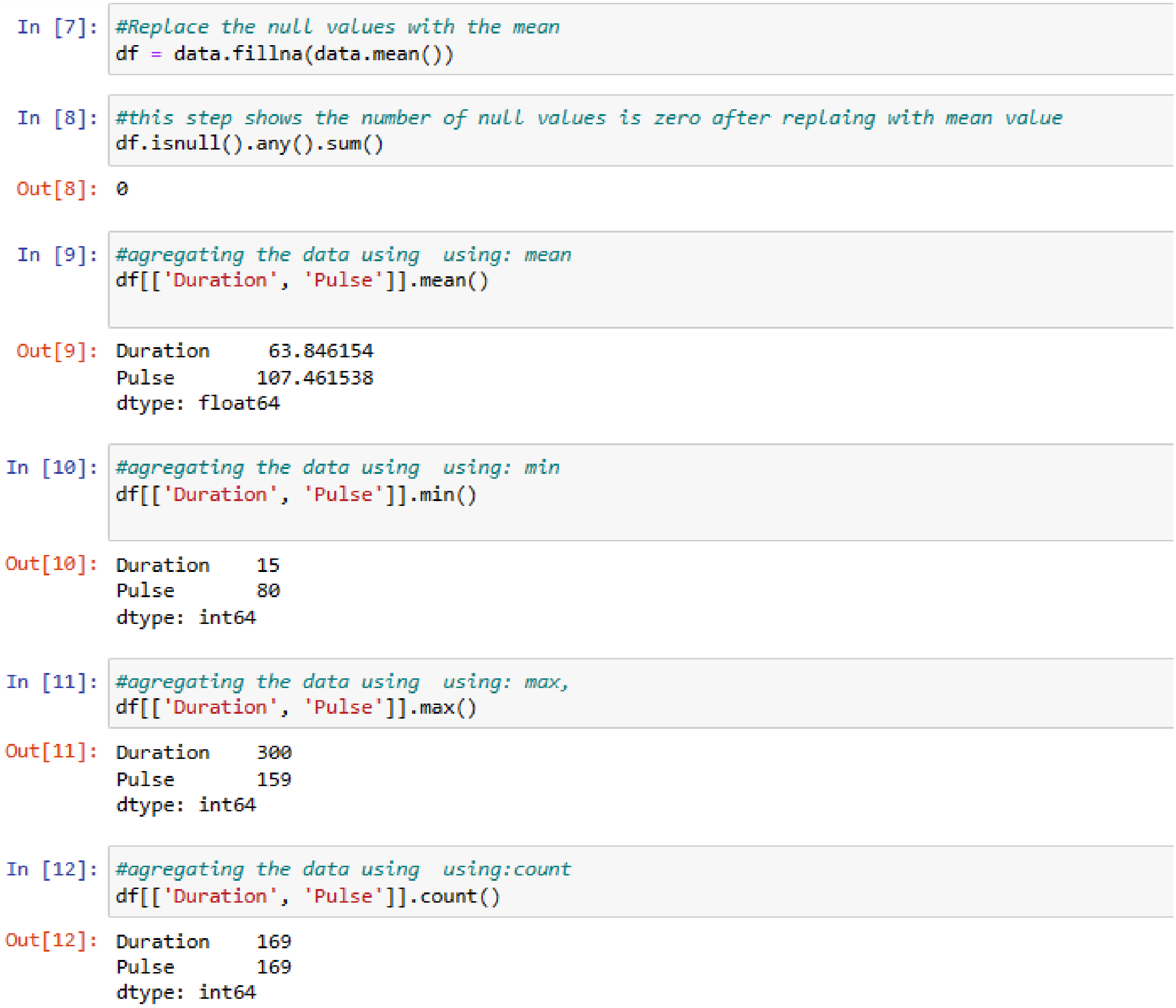
```
In [4]: #Getting the information of the data
        data.info()

        <class 'pandas.core.frame.DataFrame'>
        RangeIndex: 169 entries, 0 to 168
        Data columns (total 4 columns):
         #   Column    Non-Null Count  Dtype
        ---  ------    --------------  -----
         0   Duration  169 non-null    int64
         1   Pulse     169 non-null    int64
         2   Maxpulse  169 non-null    int64
         3   Calories  164 non-null    float64
        dtypes: float64(1), int64(3)
        memory usage: 5.4 KB
```

```
In [5]: #Show the basic statistical description about the data.
        data.describe()

Out[5]:
```

	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

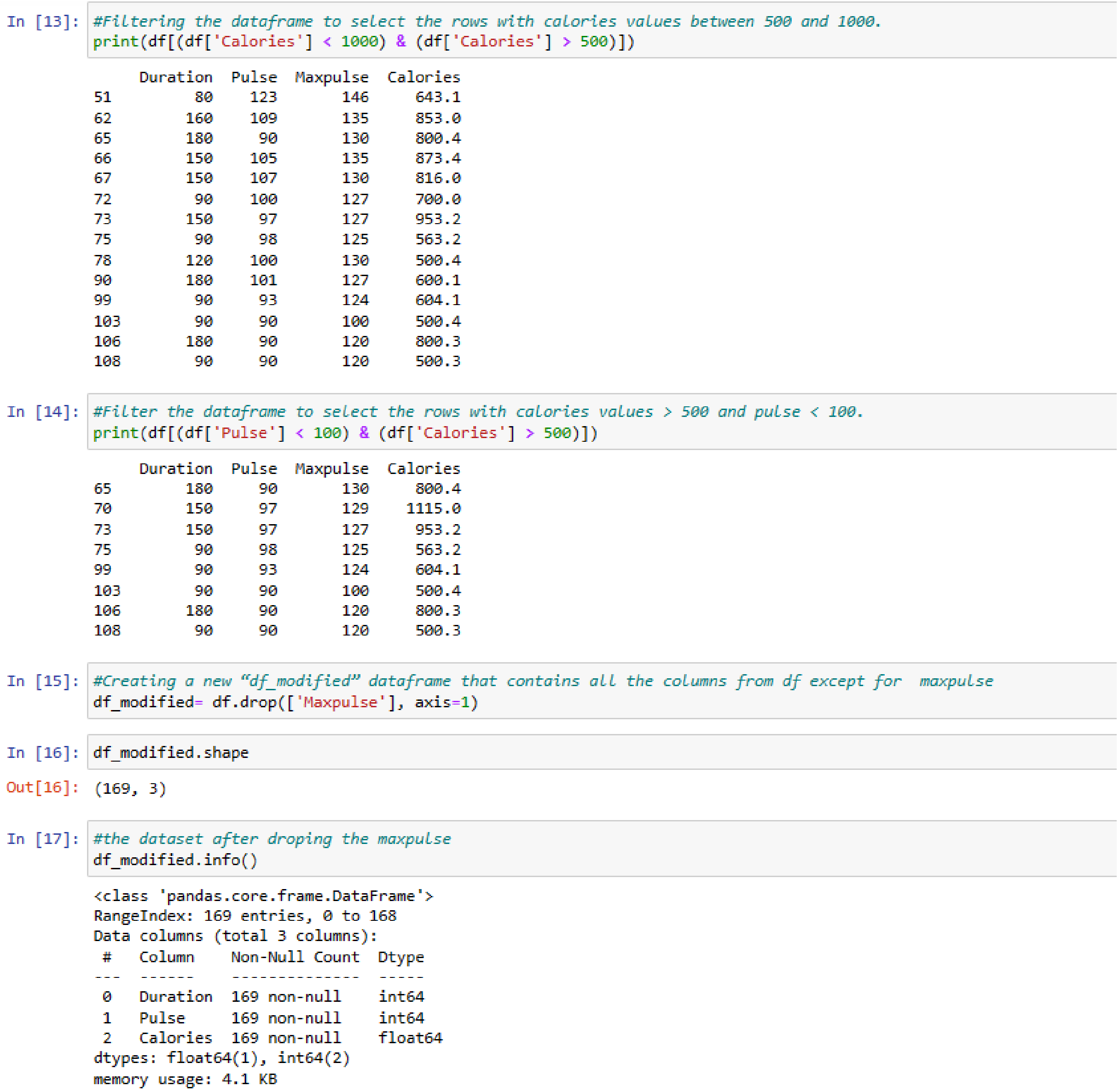
```
In [6]: #Checking for the null values in the dataset
        data.isnull().any().sum()

Out[6]: 1
```

d. Check if the data has null values.
e. Select at least two columns and aggregate the data using: min, max, count, mean.
f. Filter the dataframe to select the rows with calories values between 500 and 1000.
i. Replace the null values with the mean



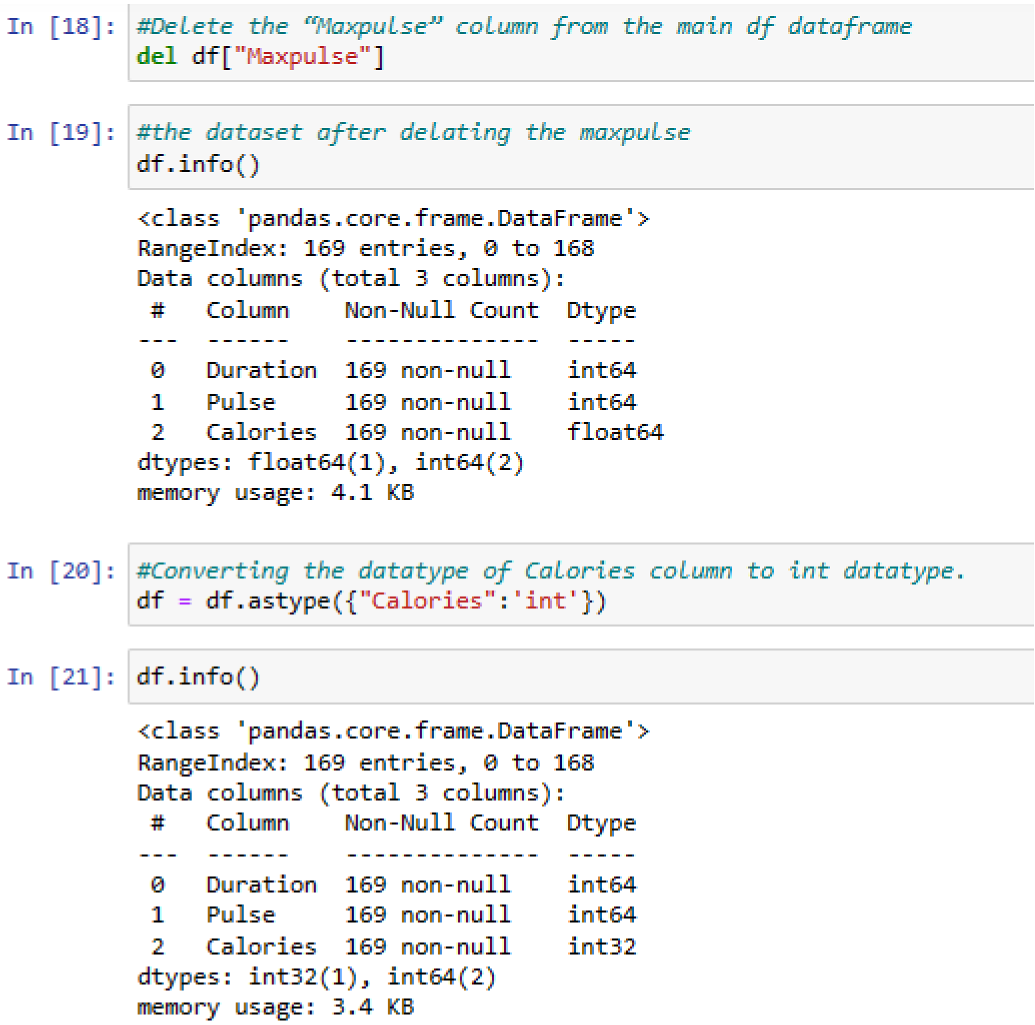
```
In [7]: #Replace the null values with the mean
        df = data.fillna(data.mean())

In [8]: #this step shows the number of null values is zero after replaing with mean value
        df.isnull().any().sum()

Out[8]: 0

In [9]: #agregating the data using  using: mean
        df[['Duration', 'Pulse']].mean()

Out[9]: Duration     63.846154
        Pulse       107.461538
        dtype: float64

In [10]: #agregating the data using  using: min
         df[['Duration', 'Pulse']].min()

Out[10]: Duration    15
         Pulse       80
         dtype: int64

In [11]: #agregating the data using  using: max,
         df[['Duration', 'Pulse']].max()

Out[11]: Duration    300
         Pulse       159
         dtype: int64

In [12]: #agregating the data using  using:count
         df[['Duration', 'Pulse']].count()

Out[12]: Duration    169
         Pulse       169
         dtype: int64
```

g. Filter the dataframe to select the rows with calories values > 500 and pulse < 100.



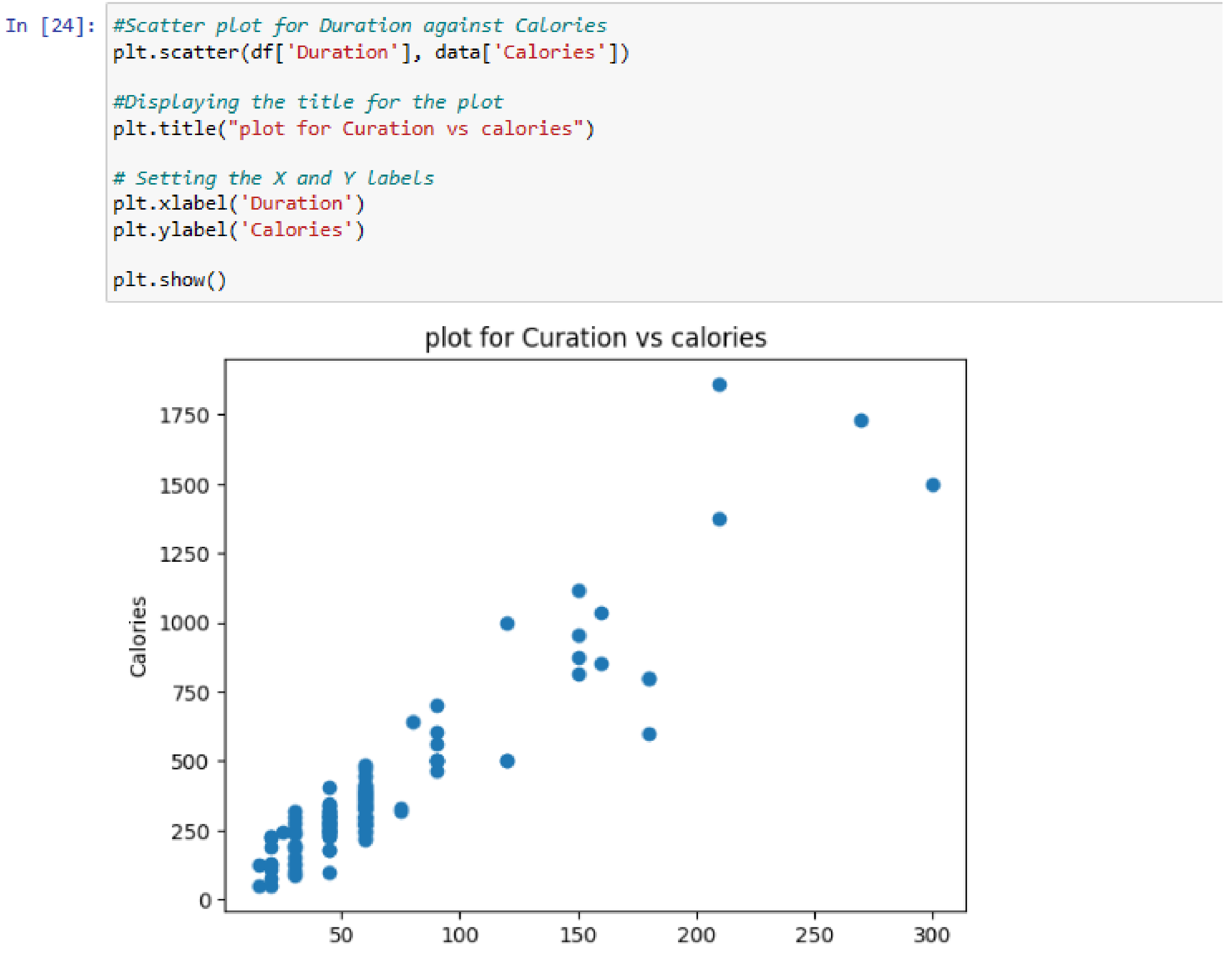
```
In [13]: #Filtering the dataframe to select the rows with calories values between 500 and 1000.
         print(df[(df['Calories'] < 1000) & (df['Calories'] > 500)])

              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
         90        180    101       127     600.1
         99         90     93       124     604.1
         103        90     90       100     500.4
         106       180     90       120     800.3
         108        90     90       120     500.3

In [14]: #Filter the dataframe to select the rows with calories values > 500 and pulse < 100.
         print(df[(df['Pulse'] < 100) & (df['Calories'] > 500)])

              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

In [15]: #Creating a new “df_modified” dataframe that contains all the columns from df except for  maxpulse
         df_modified= df.drop(['Maxpulse'], axis=1)

In [16]: df_modified.shape

Out[16]: (169, 3)

In [17]: #the dataset after droping the maxpulse
         df_modified.info()

         <class 'pandas.core.frame.DataFrame'>
         RangeIndex: 169 entries, 0 to 168
         Data columns (total 3 columns):
          #   Column    Non-Null Count  Dtype  
         ---  ------    --------------  -----  
          0   Duration  169 non-null    int64  
          1   Pulse     169 non-null    int64  
          2   Calories  169 non-null    float64
         dtypes: float64(1), int64(2)
         memory usage: 4.1 KB
```

h. Create a new “df_modified” dataframe that contains all the columns from df except for “Maxpulse”.
i. Delete the “Maxpulse” column from the main df dataframe
j. Convert the datatype of Calories column to int datatype.



```
In [18]: #Delete the “Maxpulse” column from the main df dataframe
         del df["Maxpulse"]

In [19]: #the dataset after delating the maxpulse
         df.info()

         <class 'pandas.core.frame.DataFrame'>
         RangeIndex: 169 entries, 0 to 168
         Data columns (total 3 columns):
          #   Column    Non-Null Count  Dtype  
         ---  ------    --------------  -----  
          0   Duration  169 non-null    int64  
          1   Pulse     169 non-null    int64  
          2   Calories  169 non-null    float64
         dtypes: float64(1), int64(2)
         memory usage: 4.1 KB

In [20]: #Converting the datatype of Calories column to int datatype.
         df = df.astype({"Calories":'int'})

In [21]: df.info()

         <class 'pandas.core.frame.DataFrame'>
         RangeIndex: 169 entries, 0 to 168
         Data columns (total 3 columns):
          #   Column    Non-Null Count  Dtype
         ---  ------    --------------  -----
          0   Duration  169 non-null    int64
          1   Pulse     169 non-null    int64
          2   Calories  169 non-null    int32
         dtypes: int32(1), int64(2)
         memory usage: 3.4 KB
```

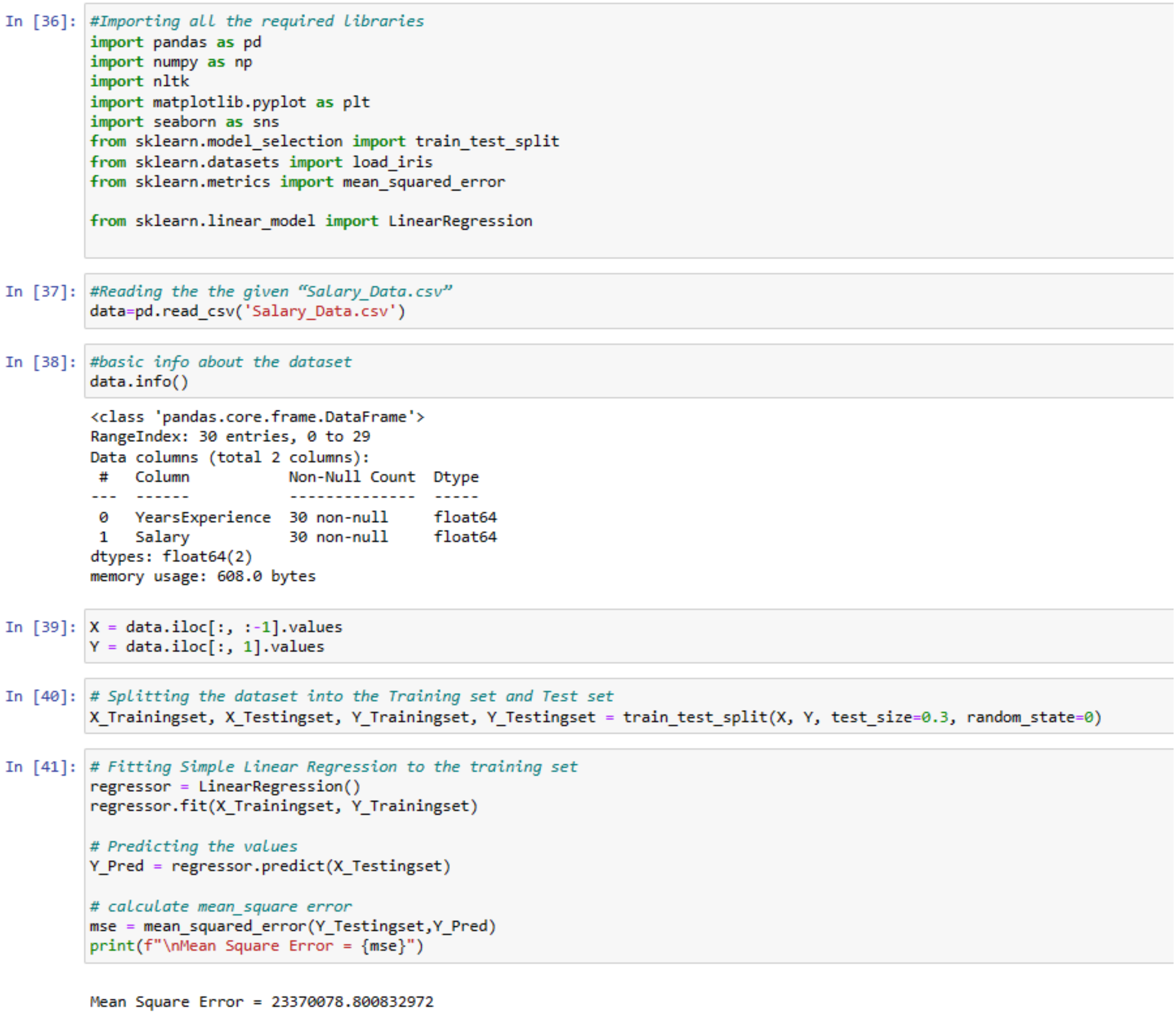
k. Using pandas create a scatter plot for the two columns (Duration and Calories).
Example



```
In [24]: #Scatter plot for Duration against Calories
         plt.scatter(df['Duration'], data['Calories'])

         #Displaying the title for the plot
         plt.title("plot for Curation vs calories")

         # Setting the X and Y labels
         plt.xlabel('Duration')
         plt.ylabel('Calories')

         plt.show()
```

2. Linear Regression

a) Import the given “Salary_Data.csv”
b) Split the data in train_test partitions, such that 1/3 of the data is reserved as test subset.
c) Train and predict the model.
d) Calculate the mean_squared error
e) Visualize both train and test data using scatter plot.



In [36]:

```
#Importing all the required libraries
import pandas as pd
import numpy as np
import nltk
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.datasets import load_iris
from sklearn.metrics import mean_squared_error

from sklearn.linear_model import LinearRegression
```

In [37]:

```
#Reading the the given “Salary_Data.csv”
data=pd.read_csv('Salary_Data.csv')
```

In [38]:

```
#basic info about the dataset
data.info()
```

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

In [39]:

```
X = data.iloc[:, :-1].values
Y = data.iloc[:, 1].values
```

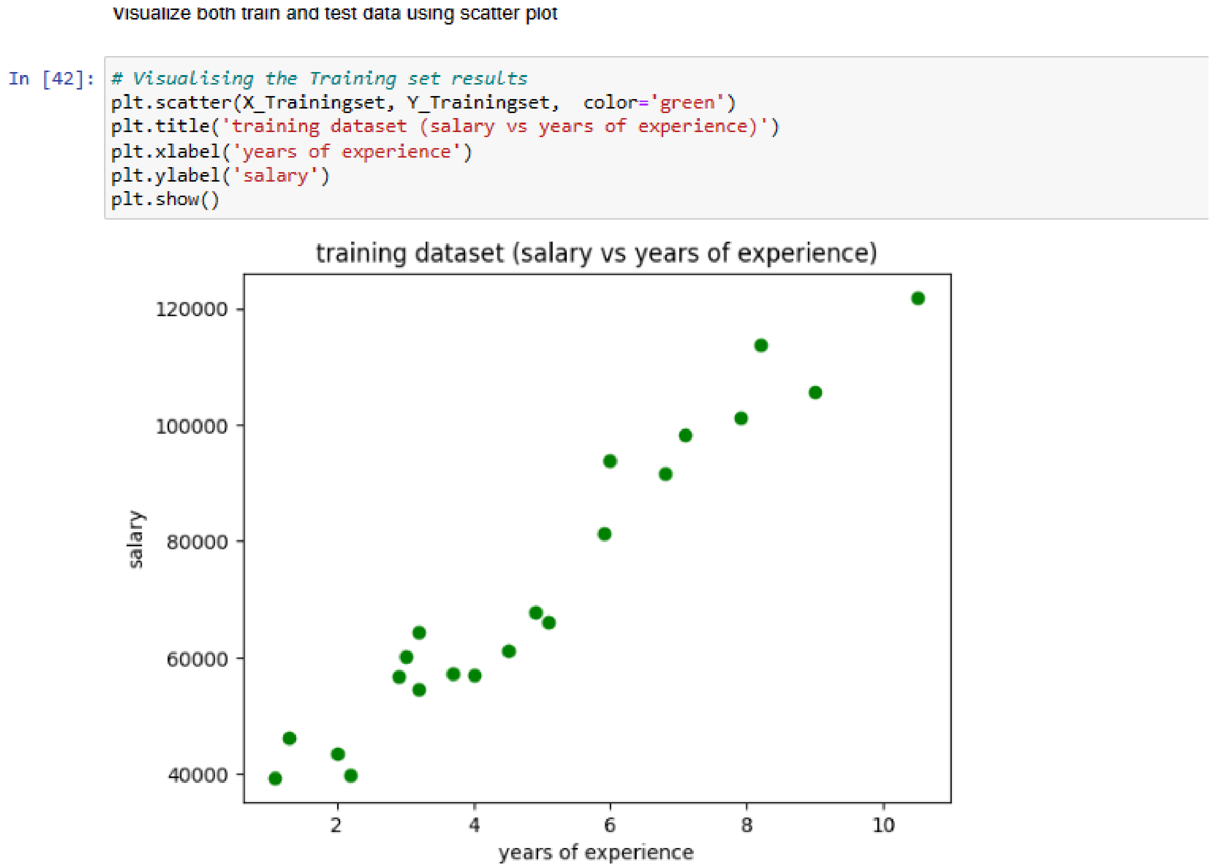
In [40]:

```
# Splitting the dataset into the Training set and Test set
X_Trainingset, X_Testingset, Y_Trainingset, Y_Testingset = train_test_split(X, Y, test_size=0.3, random_state=0)
```

In [41]:

```
# Fitting Simple Linear Regression to the training set
regressor = LinearRegression()
regressor.fit(X_Trainingset, Y_Trainingset)

# Predicting the values
Y_Pred = regressor.predict(X_Testingset)

# calculate mean_square error
mse = mean_squared_error(Y_Testingset,Y_Pred)
print(f"\nMean Square Error = {mse}")
```

```

Mean Square Error = 23370078.800832972
```

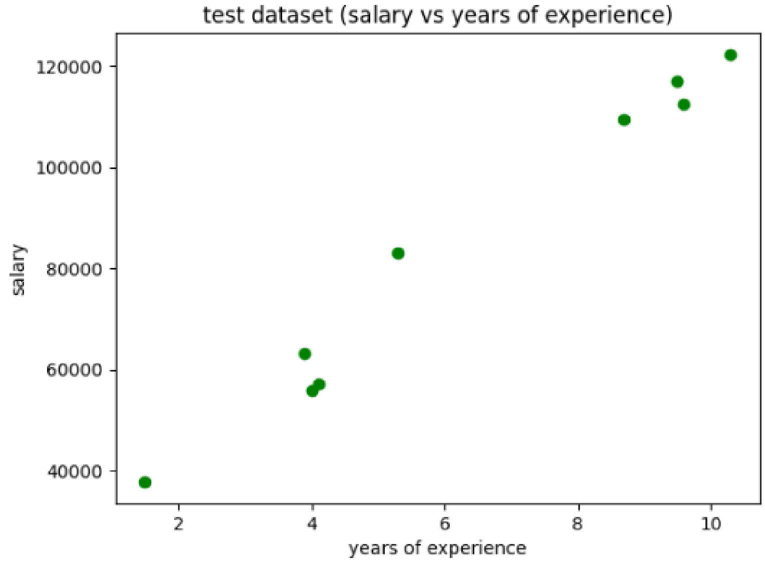


Visualize both train and test data using scatter plot

In [42]:
```
# Visualising the Training set results
plt.scatter(X_Trainingset, Y_Trainingset,  color='green')
plt.title('training dataset (salary vs years of experience)')
plt.xlabel('years of experience')
plt.ylabel('salary')
plt.show()
```

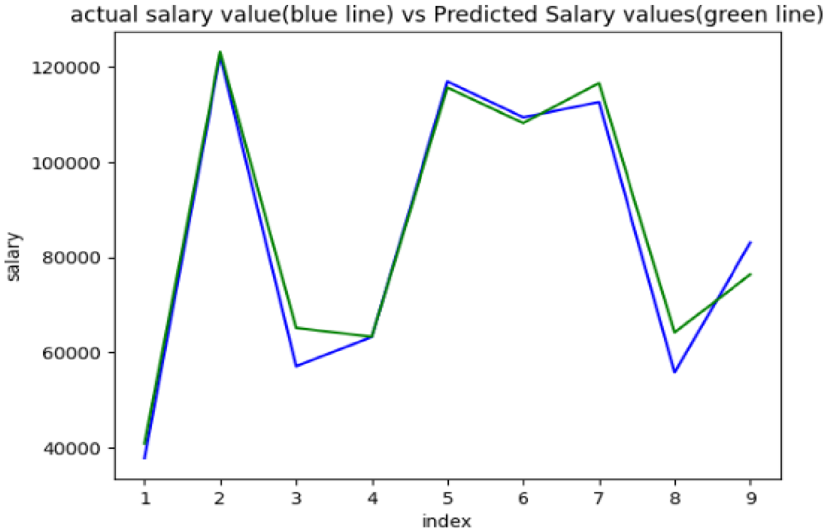
In [43]:

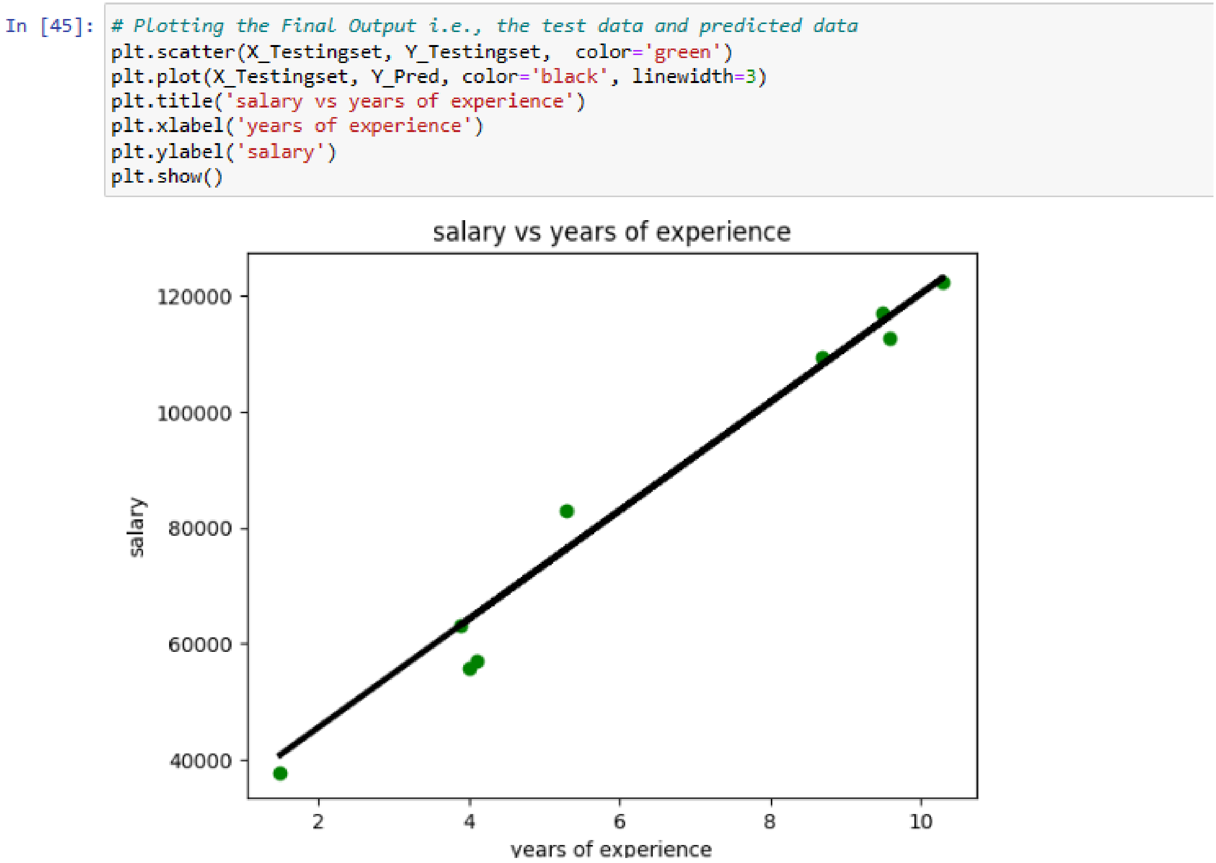
```
# Visualising the Test set results
plt.scatter(X_Testingset, Y_Testingset,  color='green')
plt.title('test dataset (salary vs years of experience)')
plt.xlabel('years of experience')
plt.ylabel('salary')
plt.show()
```



In [44]:

```
# plotting the actual and predicted values
compare = [i for i in range(1, len(Y_Testingset)+1, 1)]
plt.plot(compare, Y_Testingset, color='blue', linestyle='-')
plt.plot(compare, Y_Pred, color='green', linestyle='-')
plt.xlabel('index')
plt.ylabel('salary')
plt.title('actual salary value(blue line) vs Predicted Salary values(green line)')
plt.show()
```





```
In [45]: # Plotting the Final Output i.e., the test data and predicted data
         plt.scatter(X_Testingset, Y_Testingset,  color='green')
         plt.plot(X_Testingset, Y_Pred, color='black', linewidth=3)
         plt.title('salary vs years of experience')
         plt.xlabel('years of experience')
         plt.ylabel('salary')
         plt.show()
```

GitHub link: https://github.com/niryarjessy22/ICP-4.git

Videos link:

https://drive.google.com/file/d/1sAZAFFEnqPxI0XqLx-QSkQIXBEpQAHSG/view?usp=share_link