1. Data Manipulation
    a. Read the provided CSV file ‘data.csv’.
    b. https://drive.google.com/drive/folders/1h8C3mLsso-R-sIOLsvoYwPLzy2fJ4IOF?usp=sharing
    c. Show the basic statistical description about the data.
    d. Check if the data has null values. i. Replace the null values with the mean
    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.
    g. Filter the dataframe to select the rows with calories values > 500 and pulse < 100.
    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. k. Using pandas create a scatter plot for the two columns (Duration and Calories).

In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
```

In [2]:
```
#a. Read the provided CSV file ‘data.csv’.
df = pd.read_csv('C:/Users/koppu/Downloads/data.csv')
print(df.head())
```

```
   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
```

In [3]:
```
#c. Show the basic statistical description about the data.
print("Statistics Data:")
print(df.describe())
```

```
Statistics Data:
         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 [4]:
```
#d. Check if the data has null values
print("Checking for null values:")
print(df.isnull().sum())
```

```
Checking for null values:
Duration    0
Pulse       0
Maxpulse    0
Calories    5
dtype: int64
```

In [5]:
```
#d) i. Replace the null values with the mean
df.fillna(df.mean(), inplace=True)
print(df.isnull().sum())
```

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

In [6]:
```
#e. Select at least two columns and aggregate the data using: min, max, count, mean.
aggregated_data = df[['Pulse', 'Calories']].agg(['min', 'max', 'count', 'mean'])
print("Aggregated data for Pulse and Calories:")
print(aggregated_data)
```

```
Aggregated data for Pulse and Calories:
            Pulse     Calories
min     80.000000    50.300000
max    159.000000  1860.400000
count  169.000000   169.000000
mean   107.461538   375.790244
```

In [7]:
```
#f. Filter the dataframe to select the rows with calories values between 500 and 1000.
print("count before filtering :\n" + str(df.count()) + "\n\n")
data1 = df[(df['Calories'] >= 500) & (df['Calories'] <= 1000)]
print("count after filtering :\n" + str(data1.count()))
```

```
count before filtering :
Duration    169
Pulse       169
Maxpulse    169
Calories    169
dtype: int64


count after filtering :
Duration    17
Pulse       17
Maxpulse    17
Calories    17
dtype: int64
```

In [8]:
```
#g. Filter the dataframe to select the rows with calories values > 500 and pulse < 100.
print("count before filtering :\n" + str(df.count()) + "\n\n")
data2 = df[(df['Calories'] > 500) & (df['Pulse'] < 100)]
print("count after filtering :\n" + str(data2.count()))
```

```
count before filtering :
Duration    169
Pulse       169
Maxpulse    169
Calories    169
dtype: int64


count after filtering :
Duration    8
Pulse       8
Maxpulse    8
Calories    8
dtype: int64
```

In [9]:
```
#h. Create a new “df_modified” dataframe that contains all the columns from df except for “Maxpulse”.
df_modified = df.drop(columns=['Maxpulse'])
print(df_modified.head())
```

```
   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
```

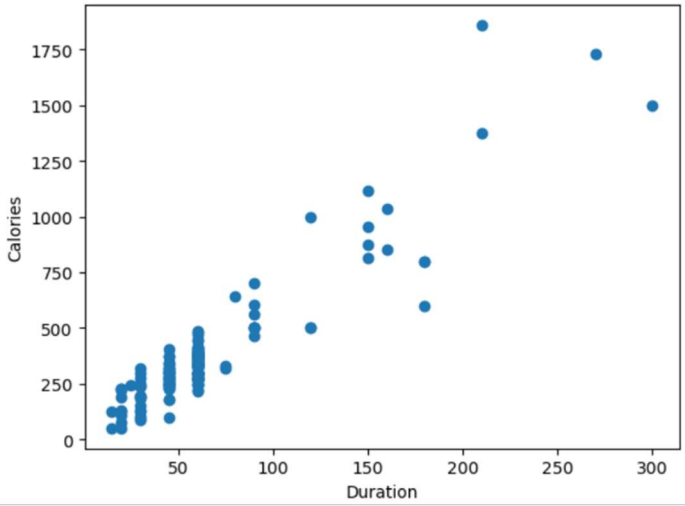
In [10]:
```
#i. Delete the “Maxpulse” column from the main df dataframe
df.drop(columns=['Maxpulse'], inplace=True)
print(df.head())
```

```
   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
```

In [11]:
```
#j. Convert the datatype of Calories column to int datatype.
df['Calories'] = df['Calories'].astype(int)
df.dtypes['Calories']
```

Out[11]:
```
dtype('int32')
```

In [27]:
```
#k. Using pandas create a scatter plot for the two columns (Duration and Calories).
plt.scatter(df['Duration'], df['Calories'])
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 [20]:
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
import matplotlib.pyplot as plt
```

In [21]:
```
#a) Import the given “Salary_Data.csv”
df = pd.read_csv('C:/Users/koppu/Downloads/Salary_Data2.csv')
print(df.head())
```

```
   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
```

In [11]:
```
#b) Split the data in train_test partitions, such that 1/3 of the data is reserved as test subset
x = df[['YearsExperience']]
y = df[['Salary']]
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=1/3, random_state=0)
print("train data count is " + str(x_train.size))
print("test data count is " + str(x_test.size))
```

```
train data count is 20
test data count is 10
```

In [22]:
```
#c) Train and predict the model.
reg = LinearRegression()
reg.fit(x_train, y_train)
y_pred = reg.predict(x_test)
print("tested data : \n" + str(y_test) + "\n\n")
print("Predicted data : \n" + str(y_pred))
```

```
tested data :
      Salary
2    37731.0
28  122391.0
13   57081.0
10   63218.0
26  116969.0
24  109431.0
27  112635.0
11   55794.0
17   83088.0
22  101302.0


Predicted data :
[[ 40835.10590871]
 [123079.39940819]
 [ 65134.55626083]
 [ 63265.36777221]
 [115602.64545369]
 [108125.8914992 ]
 [116537.23969801]
 [ 64199.96201652]
 [ 76349.68719258]
```

In [23]:
```
#d) Calculate the mean_squared error
error = mean_squared_error(y_test, y_pred)
print("Mean Squared Error: ", error)
```

```
Mean Squared Error:  21026037.329511296
```

In [18]:
```
#e) Visualize both train and test data using scatter plot
plt.scatter(x_train, y_train, color='black')
plt.scatter(x_test, y_test, color='red')
plt.plot(x_train, reg.predict(x_train), color='orange')
plt.xlabel('Years of Experience')
plt.ylabel('Salary')
plt.title('Training and Test data')
plt.show()
```