MACHINE LEARNING

GITHUB LINK: https://github.com/vijender6/vijender

RECORDING VIDEO LINK:
https://drive.google.com/file/d/1QfBjJF11W6inRNQWCPqluqrYbqFBxueg/view?usp=drive_link

NAME : VIJENDER REDDY KOOTURU

700765220

Q1. Read the provided CSV file '***data. csv***'.

https://drive.google.com/file/d/1-Ir3AXK1A77A-qCDu5gGkAxv-_nbmWlHO/view?usp=drive_link

Here in the code, I have used ***read_csv***( ) and ***head***( ) functions to read CSV file and to display the first 5 rows of the *DataFrame* respectively.

```
import pandas as pd

import io
df=pd.read_csv(io.BytesIO(uploaded['data.csv']))

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

Q2. Here in the code, I have used the ***describe***( ) function to generate descriptive statistics of a *DataFrame*. Then the description is printed.

```
description=df.describe()
description
```

	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

Q3. Here in the code, I have used the ***isnull***( ) function to identify null values in the $DataFrame$, and the ***values***. ***any***( ) function to check if any of the values are null.

Then ***sum***( ) function is called on this Boolean DataFrame and displayed a new DataFrame containing the sum of ***True*** values for each column in ***df***.

```
[7] print( 'Are there any null values: ', df. isnull().values.any ())

    # Checking for null values in the Dataframe
    null_values = df. isnull(). sum()

    # Printing the number of null values for each column
    null_values
```

```
Are there any null values:  True
Duration    0
Pulse       0
Maxpulse    0
Calories    5
dtype: int64
```

Q3(a). Replace the null values with the mean.

```
#Replacing null values with the mean of the respective column
mean_values = df.mean()
df. fillna(mean_values, inplace=True)
print('Are there any null values after replacing: ', df.isnull().values.any())

# Checking for null values in the Dataframe (shguld return all 0s)
null_values= df.isnull().sum()

# Printing the number of null values for each column
null_values
```

```
Are there any null values after replacing:  False
Duration    0
Pulse       0
Maxpulse    0
Calories    0
dtype: int64
```

Q4. I have used ***agg***( ) function to perform the aggregation operations on the selected columns, with a dictionary as an argument, where the keys are the column names to be aggregated and the values are lists of aggregation functions to be applied to each column.

```
#Selecting two columns maxpulse,calories and aggregating using min, max, count, and mean

agg_df=df.agg({'Maxpulse': ['min', 'max', 'count', 'mean'],
               'Calories': ['min', 'max', 'count', 'mean']})


# Printing the aggregated data
agg_df
```

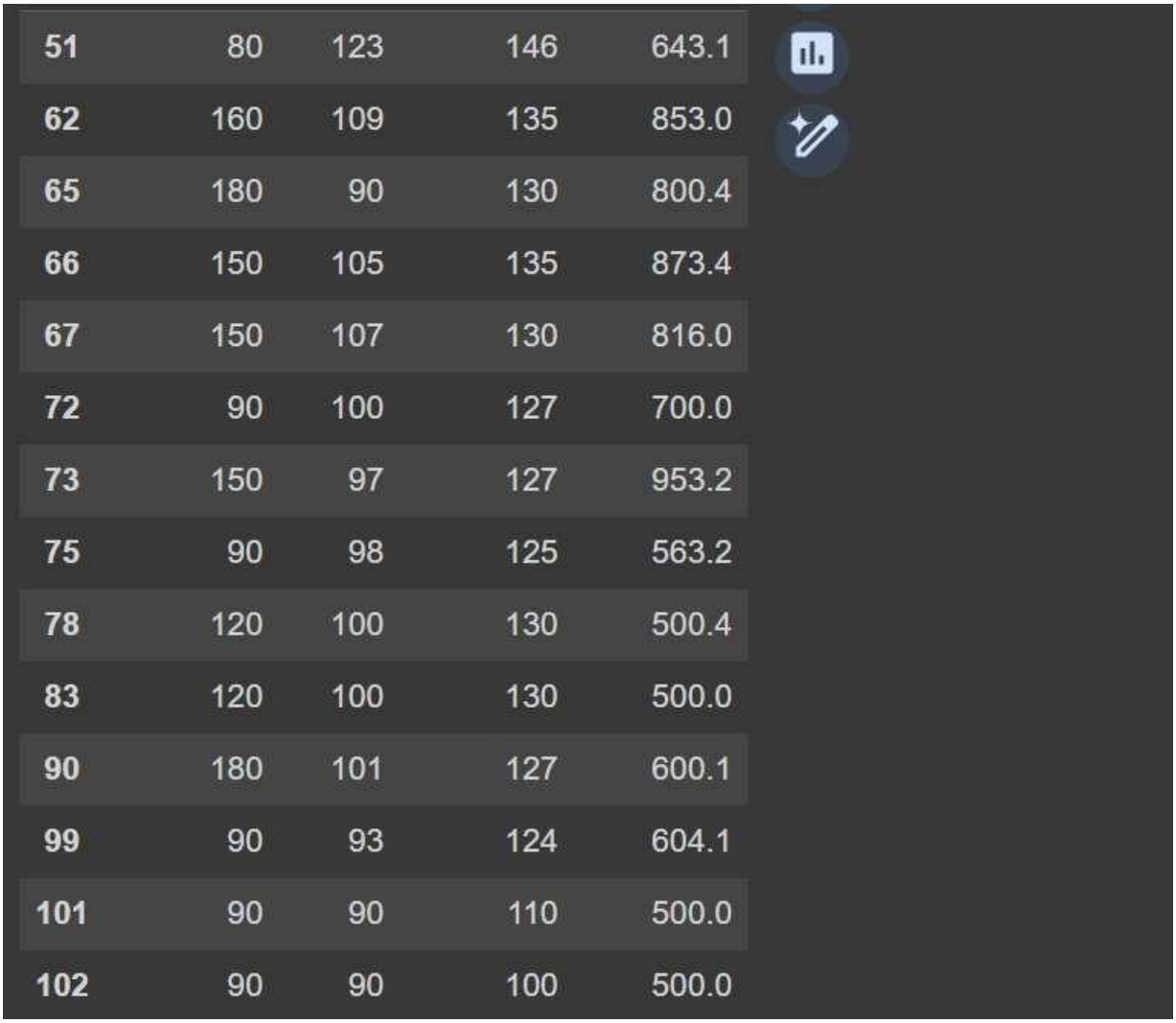
	Maxpulse	Calories
min	100.000000	50.300000
max	184.000000	1860.400000
count	169.000000	169.000000
mean	134.047337	375.790244

Q5. The ***loc***[ ] function is used to select rows based on a boolean condition. The condition is specified inside the square brackets of the ***loc***[ ] function using the '$Calories$' column of df. Specifically, ***df***['***Calories***'] >= **500** and ***df***['***Calories***'] <= **1000** are two separate boolean conditions, which are combined using the & operator to specify the filter condition.

```
#Filtering the Dataframe to select rws with calorie values between 500 and 1000
filtered_df=df.loc[(df['Calories']>=500) & (df['Calories']<=1000)]

filtered_df
```

OUTPUT:



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

Q6. Filter the DataFrame to select the rows with calories values > 500 and pulse < 100.

```
#Filtering the Dataframe to select rows with calorie values >500 and values pulse < 100
filtered_df=df.loc[(df['Calories']>500) & (df['Pulse']<100)]

filtered_df
```

	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

✓ 0s completed at 10:36 AM

Q7. Create a new "$df\_modified$" dataframe that contains all the columns from df except for "$Maxpulse$".

```
# Dropping the 'Maxpulse' column and creating a new Dataframe
df_modified=df.drop(columns=['Maxpulse'])

#Printing the first 5 rows of new Dataframe
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

Q8. Delete the "$Maxpulse$" column from the main df dataframe.

```
# Dropping the 'Maxpulse' column from the orginal  Dataframe
df.drop(columns=['Maxpulse'], inplace=True)

#Printing the first 5 rows of new Dataframe
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

Q9. Convert the datatype of $Calories$ column to int datatype.

```
#astype() function is used to convert the data type
df['Calories']=df['Calories'].astype('int64')
df.dtypes
```

```
Duration    int64
Pulse       int64
Calories    int64
dtype: object
```