Python Final Project

Name = Manoj Kumar

Batch = 1st September Batch

Course = Data Science Placement Guarantee Course

Email Id = manojkumarrajput9990@gmail.com

Here is Python Project Explaination Video Link below:

https://drive.google.com/file/d/1aZXHkWLCVZm_tKN__rJI7KA-3sxCpOHS/view?usp=sharing

Imported Libraries pandas, os.

In [1]:
```
import pandas as pd
import os
```

Loading Data with Pandas

We need to load data `FEV_data_Excel.csv` using pandas `read_csv()` method. here below.Loaded the dataset as `FEV_Data` using `read_csv()`

Then to take look of dataset so that we can understand the data and it's different-different columns and rows.

For that We need to use `head()` method See `top 5 rows` of the Data.

In [3]:
```
FEV_Data = pd.read_csv("C:\\Users\\msgme\\Downloads\\FEV_data_Excel.csv")
FEV_Data.head()
```

Out[3]:

	Car full name	Make	Model	Minimal price (gross) [PLN]	Engine power [KM]	Maximum torque [Nm]	Type of brakes	Drive type	Battery capacity [kWh]	Ran (WL [k
0	Audi e-tron 55 quattro	Audi	e-tron 55 quattro	345700	360	664	disc (front + rear)	4WD	95.0	4
1	Audi e-tron 50 quattro	Audi	e-tron 50 quattro	308400	313	540	disc (front + rear)	4WD	71.0	3
2	Audi e-tron S quattro	Audi	e-tron S quattro	414900	503	973	disc (front + rear)	4WD	95.0	3
3	Audi e-tron Sportback 50 quattro	Audi	e-tron Sportback 50 quattro	319700	313	540	disc (front + rear)	4WD	71.0	3
4	Audi e-tron Sportback 55 quattro	Audi	e-tron Sportback 55 quattro	357000	360	664	disc (front + rear)	4WD	95.0	4

5 rows × 25 columns

Task 1: A customer has a `budget of 350,000 PLN` and wants an `EV with a minimum range`

of 400 km.

Task 1.(A) Your task is to filter out EVs that meet these criteria.

Filtering The data

For the task given above, First of all we need to Filtered the data,where `Minimal price (gross)` is

less than equal to `350000` and `Range (WLTP)` is between `(400, 600)` as per the Question.

So i filtered out the FEV_Data where minimal price is less than equal to 350000 and range between 400 to 600

because custmoer has budget of 350000 so we need the data of EV Car of 350000 or less than that, and range also

should be minimum 400 so we need to the data of that EVs which price is less than or equal to 350000 and range 400

or can be more than that.so i filtered out the below.

And Dislayed the Filtered_FEV_Data below:

In [5]:
```
Filtered_FEV_Data = FEV_Data[(FEV_Data["Minimal price (gross) [PLN]"] <= 350000) &
```

Displayed the filtered data

In [7]:
```
print(Filtered_FEV_Data)
```

	Car full name	Make	Model	Minimal price (gross) [PLN]	Engine power [KM]	Maximum torque [Nm]	Type of brakes	Drive type	Battery capacity [kWh]	Range (WLTP) [km]	...
0	Audi e-tron 55 quattro	Audi	e-tron 55 quattro	345700	360	664	disc (front + rear)	4WD	95.0	438	...
8	BMW iX3	BMW	iX3	282900	286	400	disc (front + rear)	2WD (rear)	80.0	460	...
15	Hyundai Kona electric 64kWh	Hyundai	Kona electric 64kWh	178400	204	395	disc (front + rear)	2WD (front)	64.0	449	...
18	Kia e-Niro 64kWh	Kia	e-Niro 64kWh	167990	204	395	disc (front + rear)	2WD (front)	64.0	455	...
20	Kia e-Soul 64kWh	Kia	e-Soul 64kWh	160990	204	395	disc (front + rear)	2WD (front)	64.0	452	...
22	Mercedes-Benz EQC	Mercedes-Benz	EQC	334700	408	760	disc (front + rear)	4WD	80.0	414	...
39	Tesla Model 3 Standard Range Plus	Tesla	Model 3 Standard Range Plus	195490	285	450	disc (front + rear)	2WD (rear)	54.0	430	...
40	Tesla Model 3 Long Range	Tesla	Model 3 Long Range	235490	372	510	disc (front + rear)	4WD	75.0	580	...
41	Tesla Model 3 Performance	Tesla	Model 3 Performance	260490	480	639	disc (front + rear)	4WD	75.0	567	...
47	Volkswagen ID.3 Pro Performance	Volkswagen	ID.3 Pro Performance	155890	204	310	disc (front) + drum (rear)	2WD (rear)	58.0	425	...
48	Volkswagen ID.3 Pro S	Volkswagen	ID.3 Pro S	179990	204	310	disc (front) + drum (rear)	2WD (rear)	77.0	549	...
49	Volkswagen ID.4 1st	Volkswagen	ID.4 1st	202390	204	310	disc (front) + drum (rear)	2WD (rear)	77.0	500	...

	Permissable gross weight [kg]	Maximum load capacity [kg]
0	3130.0	640.0
8	2725.0	540.0
15	2170.0	485.0
18	2230.0	493.0
20	1682.0	498.0
22	2940.0	445.0
39	NaN	NaN
40	NaN	NaN
41	NaN	NaN
47	2270.0	540.0
48	2280.0	412.0
49	2660.0	661.0

	Number of seats	Number of doors	Tire size [in]	Maximum speed [kph]
0	5	5	19	200
8	5	5	19	180
15	5	5	17	167
18	5	5	17	167
20	5	5	17	167
22	5	5	19	180
39	5	5	18	225
40	5	5	18	233
41	5	5	20	261
47	5	5	18	160
48	5	5	19	160
49	5	5	20	160

	Boot capacity (VDA) [l]	Acceleration 0-100 kph [s]
0	660.0	5.7
8	510.0	6.8
15	332.0	7.6
18	451.0	7.8
20	315.0	7.9
22	500.0	5.1
39	425.0	5.6
40	425.0	4.4
41	425.0	3.3
47	385.0	7.3
48	385.0	7.9
49	543.0	8.5

	Maximum DC charging power [kW]	mean - Energy consumption [kWh/100 km]
0	150	24.45
8	150	18.80
15	100	15.40
18	100	15.90
20	100	15.70
22	110	21.85
39	150	NaN
40	150	NaN
41	150	NaN
47	100	15.40
48	125	15.90
49	125	18.00

```
[12 rows x 25 columns]
```

Task 1.(B) Group them by the manufacturer (Make).

After Filtering the Data,Our second Task is to `Group the Data by Manufacturer or Maker` ,For that i used

`groupby()` method than used `For Loop` to get data by Manufacturer Like Audi's Filtered_FEV_Data,Tesla's

Filtered_FEV_Data etc. and got the Data The EVs car which price is less than equal to 350000 and Range

between 400 to 600 and i got the data below as you can see below:

In [9]:
```
Grouped_by_Maker = Filtered_FEV_Data.groupby("Make")

for i, j in Grouped_by_Maker:
    print(f"\nManufacturer: {i}")
    print(j)
```

```
Manufacturer: Audi
           Car full name  Make              Model  \
0  Audi e-tron 55 quattro  Audi  e-tron 55 quattro

   Minimal price (gross) [PLN]  Engine power [KM]  Maximum torque [Nm]  \
0                       345700                360                  664

        Type of brakes Drive type  Battery capacity [kWh]  Range (WLTP) [km]  \
0  disc (front + rear)        4WD                    95.0                438

   ...  Permissable gross weight [kg]  Maximum load capacity [kg]  \
0  ...                         3130.0                       640.0

   Number of seats  Number of doors  Tire size [in]  Maximum speed [kph]  \
0                5                5              19                  200

   Boot capacity (VDA) [l]  Acceleration 0-100 kph [s]  \
0                    660.0                         5.7

   Maximum DC charging power [kW]  mean - Energy consumption [kWh/100 km]
0                             150                                   24.45

[1 rows x 25 columns]

Manufacturer: BMW
  Car full name Make Model  Minimal price (gross) [PLN]  Engine power [KM]  \
8       BMW iX3  BMW   iX3                        282900                286

   Maximum torque [Nm]       Type of brakes  Drive type  \
8                  400  disc (front + rear)  2WD (rear)

   Battery capacity [kWh]  Range (WLTP) [km]  ...  \
8                    80.0                460  ...

   Permissable gross weight [kg]  Maximum load capacity [kg]  Number of seats  \
8                         2725.0                       540.0                5

   Number of doors  Tire size [in]  Maximum speed [kph]  \
8                5              19                  180

   Boot capacity (VDA) [l]  Acceleration 0-100 kph [s]  \
8                    510.0                         6.8

   Maximum DC charging power [kW]  mean - Energy consumption [kWh/100 km]
8                             150                                   18.8

[1 rows x 25 columns]

Manufacturer: Hyundai
                  Car full name     Make                Model  \
15  Hyundai Kona electric 64kWh  Hyundai  Kona electric 64kWh

    Minimal price (gross) [PLN]  Engine power [KM]  Maximum torque [Nm]  \
15                       178400                204                  395

          Type of brakes   Drive type  Battery capacity [kWh]  \
```

```
15  disc (front + rear)  2WD (front)                    64.0

    Range (WLTP) [km]  ...  Permissable gross weight [kg]  \
15               449  ...                         2170.0

    Maximum load capacity [kg]  Number of seats  Number of doors  \
15                       485.0               5                5

    Tire size [in]  Maximum speed [kph]  Boot capacity (VDA) [l]  \
15             17                  167                    332.0

    Acceleration 0-100 kph [s]  Maximum DC charging power [kW]  \
15                         7.6                             100

    mean - Energy consumption [kWh/100 km]
15                                   15.4

[1 rows x 25 columns]

Manufacturer: Kia
       Car full name Make         Model  Minimal price (gross) [PLN]  \
18  Kia e-Niro 64kWh  Kia  e-Niro 64kWh                       167990
20  Kia e-Soul 64kWh  Kia  e-Soul 64kWh                       160990

    Engine power [KM]  Maximum torque [Nm]       Type of brakes   Drive type  \
18                204                  395  disc (front + rear)  2WD (front)
20                204                  395  disc (front + rear)  2WD (front)

    Battery capacity [kWh]  Range (WLTP) [km]  ...  \
18                    64.0               455  ...
20                    64.0               452  ...

    Permissable gross weight [kg]  Maximum load capacity [kg]  \
18                         2230.0                       493.0
20                         1682.0                       498.0

    Number of seats  Number of doors  Tire size [in]  Maximum speed [kph]  \
18                5                5              17                  167
20                5                5              17                  167

    Boot capacity (VDA) [l]  Acceleration 0-100 kph [s]  \
18                    451.0                          7.8
20                    315.0                          7.9

    Maximum DC charging power [kW]  mean - Energy consumption [kWh/100 km]
18                             100                                    15.9
20                             100                                    15.7

[2 rows x 25 columns]

Manufacturer: Mercedes-Benz
        Car full name           Make Model  Minimal price (gross) [PLN]  \
22  Mercedes-Benz EQC  Mercedes-Benz   EQC                       334700

    Engine power [KM]  Maximum torque [Nm]       Type of brakes Drive type  \
22                408                  760  disc (front + rear)        4WD
```

```
    Battery capacity [kWh]  Range (WLTP) [km]  ...  \
22                    80.0                414  ...

    Permissable gross weight [kg]  Maximum load capacity [kg]  \
22                         2940.0                       445.0

    Number of seats  Number of doors  Tire size [in]  Maximum speed [kph]  \
22               5                5              19                  180

    Boot capacity (VDA) [l]  Acceleration 0-100 kph [s]  \
22                    500.0                         5.1

    Maximum DC charging power [kW]  mean - Energy consumption [kWh/100 km]
22                             110                                   21.85

[1 rows x 25 columns]

Manufacturer: Tesla
                        Car full name   Make                          Model  \
39  Tesla Model 3 Standard Range Plus  Tesla  Model 3 Standard Range Plus
40           Tesla Model 3 Long Range  Tesla           Model 3 Long Range
41          Tesla Model 3 Performance  Tesla          Model 3 Performance

    Minimal price (gross) [PLN]  Engine power [KM]  Maximum torque [Nm]  \
39                       195490                285                  450
40                       235490                372                  510
41                       260490                480                  639

         Type of brakes  Drive type  Battery capacity [kWh]  \
39  disc (front + rear)  2WD (rear)                    54.0
40  disc (front + rear)         4WD                    75.0
41  disc (front + rear)         4WD                    75.0

    Range (WLTP) [km]  ...  Permissable gross weight [kg]  \
39                430  ...                           NaN
40                580  ...                           NaN
41                567  ...                           NaN

    Maximum load capacity [kg]  Number of seats  Number of doors  \
39                         NaN                5                5
40                         NaN                5                5
41                         NaN                5                5

    Tire size [in]  Maximum speed [kph]  Boot capacity (VDA) [l]  \
39              18                  225                    425.0
40              18                  233                    425.0
41              20                  261                    425.0

    Acceleration 0-100 kph [s]  Maximum DC charging power [kW]  \
39                         5.6                             150
40                         4.4                             150
41                         3.3                             150

    mean - Energy consumption [kWh/100 km]
39                                     NaN
```

```
40                                    NaN
41                                    NaN

[3 rows x 25 columns]

Manufacturer: Volkswagen
                     Car full name        Make                  Model  \
47  Volkswagen ID.3 Pro Performance  Volkswagen  ID.3 Pro Performance
48            Volkswagen ID.3 Pro S  Volkswagen            ID.3 Pro S
49              Volkswagen ID.4 1st  Volkswagen              ID.4 1st

    Minimal price (gross) [PLN]  Engine power [KM]  Maximum torque [Nm]  \
47                       155890               204                  310
48                       179990               204                  310
49                       202390               204                  310

               Type of brakes  Drive type  Battery capacity [kWh]  \
47  disc (front) + drum (rear)  2WD (rear)                    58.0
48  disc (front) + drum (rear)  2WD (rear)                    77.0
49  disc (front) + drum (rear)  2WD (rear)                    77.0

    Range (WLTP) [km]  ...  Permissable gross weight [kg]  \
47                425  ...                         2270.0
48                549  ...                         2280.0
49                500  ...                         2660.0

    Maximum load capacity [kg]  Number of seats  Number of doors  \
47                       540.0               5                5
48                       412.0               5                5
49                       661.0               5                5

    Tire size [in]  Maximum speed [kph]  Boot capacity (VDA) [l]  \
47              18                  160                    385.0
48              19                  160                    385.0
49              20                  160                    543.0

    Acceleration 0-100 kph [s]  Maximum DC charging power [kW]  \
47                         7.3                             100
48                         7.9                             125
49                         8.5                             125

    mean - Energy consumption [kWh/100 km]
47                                    15.4
48                                    15.9
49                                    18.0

[3 rows x 25 columns]
```

Taks 1.(C) Calculate the average battery capacity for each manufacturer.

Here in Task1(C) To get Average battery capacity for each manufacturer,We need to group the Filtered_FEV_Data

by Maker/ Manufacturer and Battery capacity and then we need `Mean()` of it, and need index also to get best match

First then Second,Third and so on.So i grouped both the column `maker` and `Battery capacity` and used `mean()`

method to get Average of battery capacity and used `Reset_index()` method to get Sequence of it, so that i can get

Best match first,So I Calculated the average battery capacity for each manufacturer like: Audi, BMW, Tesla etc.

then Displayed the result using `print()` statement.As you can see below:

In [11]:
```
Average_Bettery_Capacity = (Filtered_FEV_Data.groupby("Make")["Battery capacity [kW
print(" average battery capacity for each manufacturer.")
print(Average_Bettery_Capacity)
```

```
average battery capacity for each manufacturer.
           Make  Battery capacity [kWh]
0          Audi               95.000000
1           BMW               80.000000
2       Hyundai               64.000000
3           Kia               64.000000
4  Mercedes-Benz              80.000000
5         Tesla               68.000000
6    Volkswagen               70.666667
```

Task 2: You suspect some EVs have unusually high or low energy consumption. Find the

outliers in the mean - Energy consumption [kWh/100 km] column.

Imported Packages

For this Task We need to Import libraries like `pandas, numpy, matplotlib.pyplot, seaborn` .

So i impoerted these Packages first.

In [13]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

Missing Value Treatment

For this Task First of all We need to do `Missing value treatment` for that we need to remove all the missing

value from `mean - Energy consumption` column, so i deleted all the missing values, using `dropna()` ,because

missing value can effect our result.

In [15]:
```
FEV_Data = FEV_Data.dropna(subset=["mean - Energy consumption [kWh/100 km]"])
```

Statistical Method Quantile()

Here we divide the dataset into four equal part and `25th percentile represent the lowest 25% data`.

`75th percentile represent top 25% data`.

Then we need to find the outliers, So we need to get `25% and 75% quantile` for that, So for getting quantile

of `mean - Energy consumption` column, I used `quantile()` method, so that i can get `25th percentile and 75th`

percentile. and i difined a New variable `First_Quantile` and saved `25th percentile` in it, and defined another

variable `Third_Quantile` and saved `75th percentile` in it.

In [17]:
```
First_Quantile = FEV_Data["mean - Energy consumption [kWh/100 km]"].quantile(0.25)
Third_Quantile = FEV_Data["mean - Energy consumption [kWh/100 km]"].quantile(0.75)
```

IQR(Interquartile Range) Mid_Range

As we divided the dataset into `four equal part and 25th percentile represent the lowest 25% data`.

`75th percentile represent top 25% data`, After that we need to get `IQR`,And `IQR` means the data which

is spreaded in middle of the 50% part, it is the area between First and third quantile.

So with help of the middle 50% which is called `IQR` or here I named it Mid_range we can understand

the variation of it.

for getting `Mid_Range(IQR)` we need to substract `First_Quantile from Third_Quantile`,

Here I Calculated `IQR` below:

In [19]:
```
Mid_Range= Third_Quantile - First_Quantile
```

Created Lower and Upper Bounds to getting Outliers

Here I am using `Lower bound and Upper bound` formula to identify `outliers` With help of calculation of upper

bound and lower bound we difine the normal range of the data.

`1.` Lower bound tells us the acceptable minimum value, which we can get by `subtracting 1.5 times` the mid_range

from `First quantile` and if any value is lower than this boundary than that value can be considered as outliers.

`2.` Upper bound tells us, the acceptable maximum value, which we can get by `adding 1.5 times` the mid_range from

`Third quantile` and if any value is upper than this boundary than that value can be considered as `outliers`.

and Displayed Lower and Upper Bounds Value below:

In [21]:
```
Lower_Bound = First_Quantile - 1.5 * Mid_Range
Upper_Bound = Third_Quantile + 1.5 * Mid_Range
```

print(f"Lower Bound: {Lower_Bound}") print(f"Upper Bound: {Upper_Bound}")

Detecting Outliers below :

Here below I wanted to get the values of mean energy consumption column which is less than lower

bound which I created. Or the values of mean energy consumption column which is greater than upper

bound which I created.and i displayed it using `print()` statement.

We get rows which match the condition, and that value can be consider as outliers.

`There is no Outliers detected in EVs mean energy consumption column.`

In [24]:
```
Outliers = FEV_Data[(FEV_Data["mean - Energy consumption [kWh/100 km]"] < Lower_Bou
                    (FEV_Data["mean - Energy consumption [kWh/100 km]"] > Upper_Boun
print("Outliers found in the dataset:")
print(Outliers)
```

```
Outliers found in the dataset:
Empty DataFrame
Columns: [Car full name, Make, Model, Minimal price (gross) [PLN], Engine power [K
M], Maximum torque [Nm], Type of brakes, Drive type, Battery capacity [kWh], Range
(WLTP) [km], Wheelbase [cm], Length [cm], Width [cm], Height [cm], Minimal empty wei
ght [kg], Permissable gross weight [kg], Maximum load capacity [kg], Number of seat
s, Number of doors, Tire size [in], Maximum speed [kph], Boot capacity (VDA) [l], Ac
celeration 0-100 kph [s], Maximum DC charging power [kW], mean - Energy consumption
[kWh/100 km]]
Index: []

[0 rows x 25 columns]
```

Created boxplot to ensure Outliers.

Here we need to create boxplot as asked in the question. So I created boxplot which displays the distribution of

mean energy consumption column data.

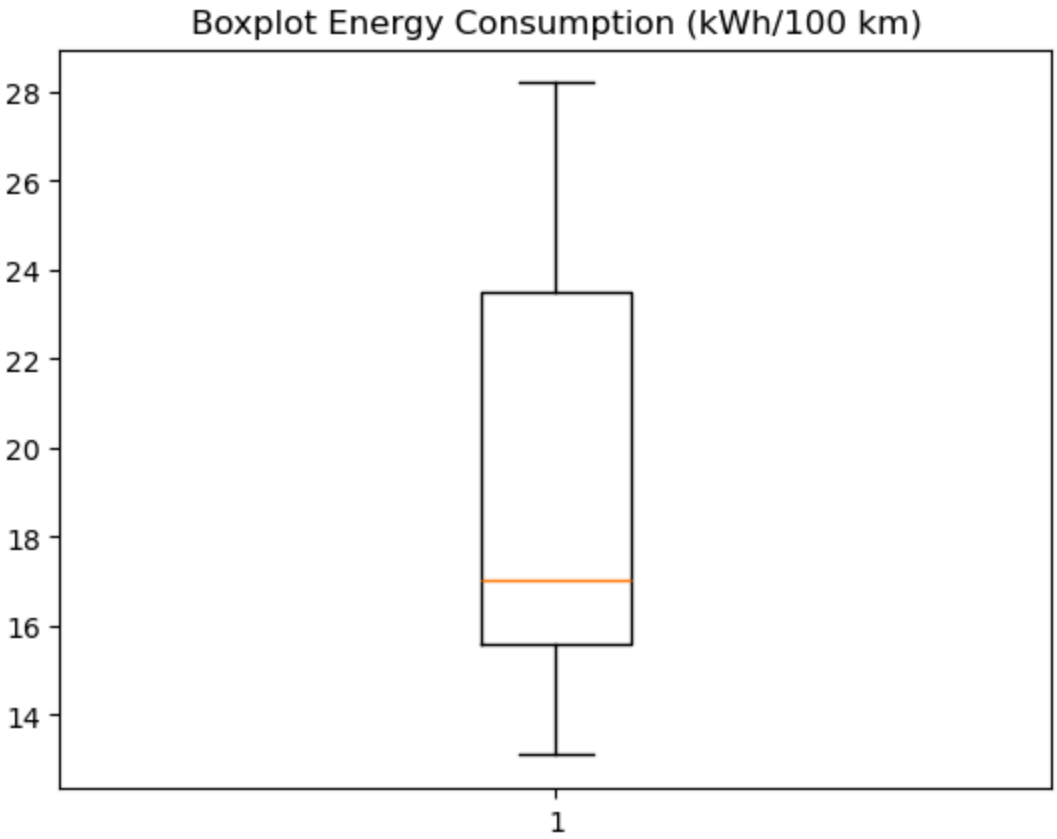
With help of this boxplot we can see median of data and `interquartile range (IQR)` and this boxplot shows outliers.

And with help of `plt.title()` we set the title of the boxplot.

And with help of `plt.Show()` we can display the plot.

****I found there is no Outliers here in mean energy consumption column.****

In [26]:
```
plt.boxplot(FEV_Data["mean - Energy consumption [kWh/100 km]"])
plt.title("Boxplot Energy Consumption (kWh/100 km)")
plt.show()
```



So i found that `there is no Outliers` here in `EVs mean energy consumption` column, there might be some other factor

because of that EVs have unusually high or low energy consumption.

Task 3: Your manager wants to know if there's a strong relationship between battery

capacity and range.

(a) Create a suitable plot to visualize.

(b) Highlight any insights

*Need to create `Scatter plot` *

Here we need to create a `scatter plot` according to the question to know, is there any `relationship`

between `battery capacity` and `Range`, if yes than is it a `strong relationship`? so I created a `scatter`

`plot` with help of which we can know is there any relationship between battery capacity and Range column or not ?

Created a scatter plot.

So for creating a scatter plot where I set size of plot like which size scatter plot we should create so

I used `plt.figure()` method and choosed the `8,5 size` for my plot. and than I defined column for `X-axis` and

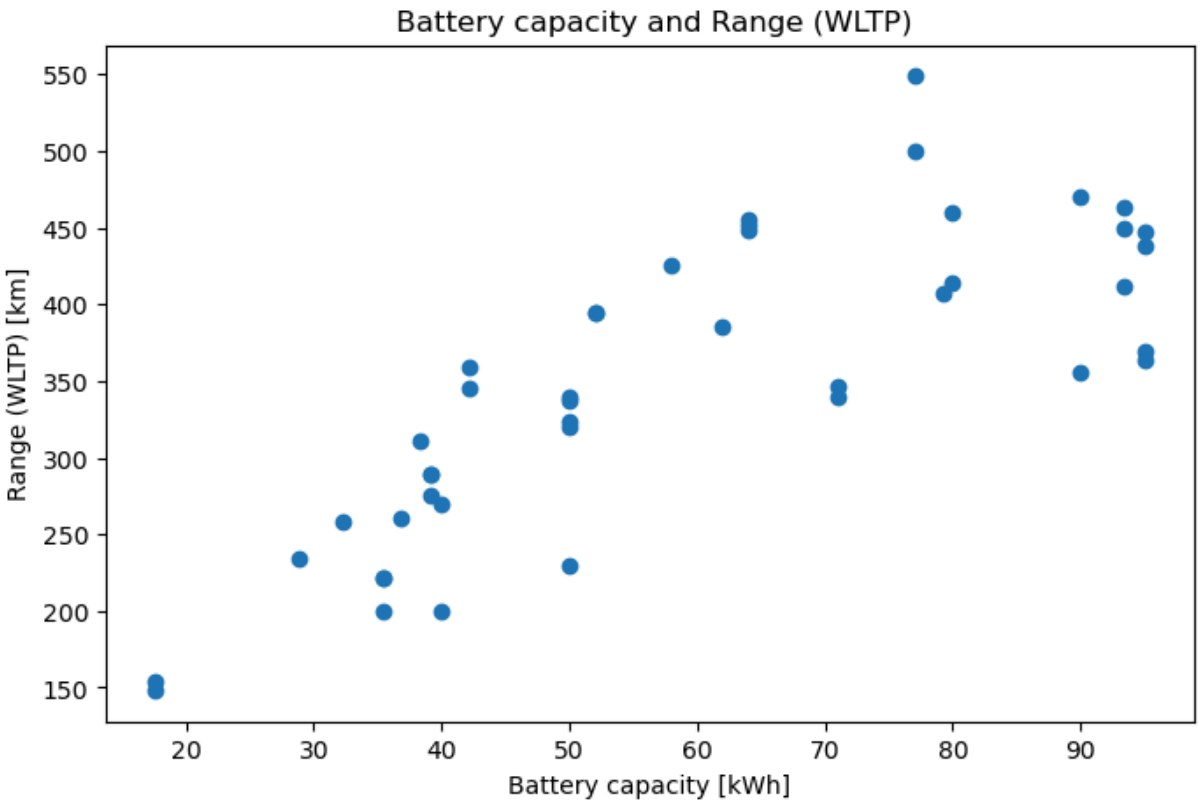
`Y-axis` than I defined title for the plot `Battery capacity` and `Range (WLTP)` using `plt.title()` method and

also I defined `Xlabel` and `Ylabel` also using `plt.xlabel()` and `plt.ylabel()` method and I displayed the plot

using `plt.show()`.

In [28]:
```
plt.figure(figsize=(8, 5))
plt.scatter(FEV_Data["Battery capacity [kWh]"], FEV_Data["Range (WLTP) [km]"])

plt.title(" Battery capacity and Range (WLTP)")
plt.xlabel("Battery capacity [kWh]")
plt.ylabel("Range (WLTP) [km]")
plt.show()
```



Here as we can see in `scatter plot` the dots are going to `upward direction`. the dots going upward in a line.

Which means there is a `strong positive relationship` between `battery capacity and Range`.

*Correlation between `Battery capacity [kWh]` and Range `(WLTP) [km]` *

Here below to calculate the `correlation` between `battery capacity` and `Range` using `corr()` method, so that I can ensure

that there is a `strong positive relationship` between `battery capacity and Range`.

So I used `corr()` method to get `correlation` and used `print()` statement to display the result.

It will provide us a value which can be between `-1 to +1`.

If the value is near `+1` it is called `strong positive relationship`.

If the value is near `-1` than it is called `strong negative relationship`.

If value is `0` than there is `no relationship` between the columns.

In [30]:
```
Correlation = FEV_Data["Battery capacity [kWh]"].corr(FEV_Data["Range (WLTP) [km]"]
print("Correlation between Bettery Capacity and Range is:", Correlation)
```

```
Correlation between Bettery Capacity and Range is: 0.7950960301829404
```

Here is just gave a condition if `correlation` is `greater` than `0.7` it will display that there is a `strong`

`positive relationship` between `battery capacity and Range` , and if correlation is not greater than 0.7 than

it will display that there is weak or negetive relationship between battery capacity and Range using `if()`

`else()` statement.

And as we already got the value `0.7950960` so it displayed that there is `strong positive relationship` .

In [32]:
```
if Correlation > 0.7:
    print("There is strong positive relationship between battery capacity and range
else:
    print("There is weak or negetive relationship between battery capacity and rang
```

```
There is strong positive relationship between battery capacity and range.
```

Insights

`Yes` , there is `strong positive relationship` because `0.7950960301829404 is colse to 1` ,

and here we got the `correlation` value `0.7950960` which means there is a `strong positive relationship` between

`battery capacity` and `Range` if we `increase battery capacity` than `range tends to increase` , it is a strong

positive relationship,as we saw in scatter plot also.

So as we can see the value is `0.7950960` which is `colse` to `+1` which means there is strong and direct relationship

between both of the variables.

Which shows that `higher battery capacity vehicles have higher range` .

it means `Car with larger Battery capacity have long range` aslo.

Task 4: Build an EV recommendation class.

The class should allow users to input their budget, desired range,and battery capacity. The class should

then return the top three EVs matching their criteria.

*Created `Class EV_recommendation` *

Here we need create a `class` first, which can help to recommed `EVs based on user input` . However user input we will give later.

First of all I created a `class` named `EV_Recommendation` than used the `_init_ method()` `to initialise the class` with my dataset

`FEV_Data` than I stored my data named `FEV_Data` inside of the `class` and in `self.FEV_Data` here is my `data saved` which I can use

after for filtering.

* `Recommendation Function:` *

Than I created a `function` named `Recommendation` , this function `recommends EVs based on 3 criteria` .

`Budget:` the budget of EVs should be less than or equal to `User's budget` .

`Desired_Range:` The Range of EVs should be greater than or equal to User's `desired range` .

`Battery_Capacity:` The Battery capacity of EVs should be more than or equal to `User's given capacity` .

Than this function will put a condition for filtering and returned that which will match all the conditions given.

`nlargest()` method will give us `Top 3 Recommendation` , it will return `Top 3 vehicle` from `Filtered EVs` where range will be high.

With help of this function User can get best `Top 3 EVs recommendation` which will be `budget friendly` and with the

`range` what user want to get and with a good `battery capacity` .

In [34]:
```
class EV_recommendation:
    def __init__(self, FEV_Data):
        self.FEV_Data = FEV_Data

## filtered data based on user's requirement like Budget, Desired_Range, Battery_ca

    def Recommendation(self, Budget, Desired_range, Battery_capacity):
        return self.FEV_Data[(self.FEV_Data["Minimal price (gross) [PLN]"] <= Budge
                             (self.FEV_Data["Range (WLTP) [km]"] >= Desired_range)
                             (self.FEV_Data["Battery capacity [kWh]"] >= Battery_ca
```

* `Recommender variable` *

Here below we created a `object` named `Recommender` of `EV_Recommendation` which connects the `FEV_Data dataset` to the `Class` .

* `User input` *

Than here I used `input()` function to get `input` values from user for `Budget`, `Desired Range and battery capacity`.

Here the `Top_Three_EVs statement` will call to the `Recommendatio`n function which we created above and `filtered out`

The `Top 3 EVs` based on `user's Budget`, `Desired Range and battery capacity`. And `print ()` statement will display the result

based on user's requirement.

Here with help of the function which we created above it will take `input` from `user` and based on the `user's requirement`

it'll display `Top 3 EVs`.

In [36]:
```
Recommender = EV_recommendation(FEV_Data)

Budget = float(input("Enter Your Budget Here:"))
Desired_range = float(input("Enter Your Desired Range Here:"))
Battery_capacity = float(input("Enter Your Battery capacity Here:"))

## for getting Top 3 EV based on user requirement I used this statement below:
Top_Three_EVs = Recommender.Recommendation(Budget, Desired_range, Battery_capacity)

## to Display Top 3 EV used print statement.
print(Top_Three_EVs)
```

```
            Car full name        Make       Model  \
48  Volkswagen ID.3 Pro S  Volkswagen  ID.3 Pro S
49    Volkswagen ID.4 1st  Volkswagen    ID.4 1st
8                 BMW iX3         BMW         iX3

    Minimal price (gross) [PLN]  Engine power [KM]  Maximum torque [Nm]  \
48                       179990                204                  310
49                       202390                204                  310
8                        282900                286                  400

                Type of brakes  Drive type  Battery capacity [kWh]  \
48  disc (front) + drum (rear)  2WD (rear)                    77.0
49  disc (front) + drum (rear)  2WD (rear)                    77.0
8          disc (front + rear)  2WD (rear)                    80.0

    Range (WLTP) [km]  ...  Permissable gross weight [kg]  \
48                549  ...                         2280.0
49                500  ...                         2660.0
8                 460  ...                         2725.0

    Maximum load capacity [kg]  Number of seats  Number of doors  \
48                       412.0                5                5
49                       661.0                5                5
8                        540.0                5                5

    Tire size [in]  Maximum speed [kph]  Boot capacity (VDA) [l]  \
48              19                  160                    385.0
49              20                  160                    543.0
8               19                  180                    510.0

    Acceleration 0-100 kph [s]  Maximum DC charging power [kW]  \
48                         7.9                             125
49                         8.5                             125
8                          6.8                             150

    mean - Energy consumption [kWh/100 km]
48                                    15.9
49                                    18.0
8                                     18.8

[3 rows x 25 columns]
```

Task 5: Inferential Statistics – Hypothesis Testing:

Test whether there is a significant difference in the average Engine power [KM] of vehicles manufactured

by two leading manufacturers i.e. Tesla and Audi. What insights can you draw from the test results?

Recommendations and Conclusion: Provide actionable insights based on your analysis. (Conduct a two sample

t-test using ttest_ind from scipy.stats module).

Imported `Packages` and `Loaded EV Data` Again

Here First of all we need to `import` the `library ttest_ind` from `scipy.stats`, so I imported the library.

here we need `fresh dataset` so again I we need to `load` the data so again I loaded `FEV_data` using `read_csv()`

method and named it `EV_Dataset` this time.

In [38]:
```
from scipy.stats import ttest_ind
```

In [40]:
```
EV_Dataset=pd.read_csv("C:\\Users\\msgme\\Downloads\\FEV_data_Excel.csv")
```

First of we need to define `Null Hypothesis` and `Alternative Hypothesis`, so here below i defined both.

-> Null Hypothesis(Ho):

`There is no difference in average engine power between Tesla and Audi vehicles`.

-> Alternative Hypothesis (Ha):

`There is a significant difference in average engine power between Tesla and Audi` .

Extracted Tesla and Audi Data with it's Engine Power

After defining `Null Hypothesis` and `Alternative Hypothesis`, we need to extract `engine power` of `Audi and Tesla`.

In `Tesla manufacturer` here I filtered data to extract where `Make(Manufacturer)` is `Tesla` and it's `engine power`.

Same I did in `Audi manufacturer` here also I filtered the data where `Make` is `Audi` and what is it's `engine power`.

In [42]:
```
Tesla_Mnufacturer = EV_Dataset[EV_Dataset["Make"]=="Tesla"]["Engine power [KM]"]
Audi_Mnufacturer = EV_Dataset[EV_Dataset["Make"]=="Audi"]["Engine power [KM]"]
```

Missing value treatment

Dropped missing values here:

Then I did `missing value treatment` , I `removed` all of the `null values` from `Tesla manufacturer` variable which I created

above, and I `removed` all of the `null values` from `Audi manufacturer` variable as well, using `dropna()` method.

In [44]:
```
Tesla_Mnufacturer = Tesla_Mnufacturer.dropna()
Audi_Mnufacturer = Audi_Mnufacturer.dropna()
```

Used len() function to get `Total numbers Tesla and Audi Cars` .

Now we need to get the `total number` of the `Tesla cars` and `total number` of `Audi cars` , so for getting the total

number of Tesla and Audi I used `len()` function here, and used this method inside of the `print()` statement so that

I can display the `result` as well, and I got there is `7 Tesla cars` and `6 Audi cars` .

In [46]:
```
print("Total Car of Tesla Manufacturer:", len(Tesla_Mnufacturer))
print("Total Car of Audi Manufacturer:", len(Audi_Mnufacturer))
```

```
Total Car of Tesla Manufacturer: 7
Total Car of Audi Manufacturer: 6
```

Performed `Two-Samle t-test` here:

Here I performed `Two-sample t-test` to check, `statiscally` is there any `significant difference` between `Tesla`

and `Audi engine power average` or not ?

`tstat` give us the `T-Statistic value` so that test result can be measured.

`p_value` is a `probability value` , so that we can know how many `chances` is there to `reject null hypothesis` .

In [49]:
```
t_stat, p_value = ttest_ind(Tesla_Mnufacturer, Audi_Mnufacturer)
```

Displayed the Result

Used `print ()` statement to display `t_stat` and `p_value` .

Here I got `T-Statistic value` which is `1.7024444`

And `P-Value` which is `0.1167269`

In [51]:
```
print("T-Statistic is:", t_stat)
print("P-Value is:", p_value)
```

```
T-Statistic is: 1.7024444538261416
P-Value is: 0.11672692675082785
```

Here as we got `T-Statistic value 1.702444` which measured the `result` it is a moderate value which `indicates`

that there can be `some difference between Tesla and Audi's engine power` but it is not a `strong evidence` .

I got `P-Value 0.1167269` which is `higher` than `0.05` , as we know if `P-Value>0.05` than we `don't reject the null`

`hypothesis` , it means statiscally we don't see any `significant difference` here.

Which means `statiscally` there is `no significant difference` between `Tesla` and `Audi's engine power` .

As we know Our Null and Alternative Hypothesis was

Null Hypothesis(Ho):

There is no difference in average engine power between Tesla and Audi vehicles.

Alternative Hypothesis (Ha):

There is a significant difference in average engine power between Tesla and Audi .

Result

And what Result we got is,

T-Statistic:

I got `T-Statistic` value `1.7024` , it shows the difference in `average engine`

`power of Tesla and Audi` .

If `T-Statistic` value is `higher` ,it means there is some `difference` between the groups.

P-Value:

Got `P-Value 0.1167` .

As I learnt from module video as Sir told that we compare this `P-Value` to a `significant level` called `Alpha(α)`

and `researchers` set that `0.05` means `5% significant level` .

I got `P-Value 0.1167` which is `higher` than `0.05` that why I * **`failed to reject Null hypothesis`** *.

Insights:

Basis of `P-Value`, statistically there is `no significant difference` in `average engine power between Tesla and Audi`.

The `visible difference` can be because of `some random variation`, not because of any `true difference in population`.

Recommendation:

Recommendation for Management:

If `Goal` is to `compare marketing` or `product strategy` of `Tesla` and `Audi`, then should focus to `other factors` like `range, price` because there is `no significant difference in engine power`.

Further Analysis:

Should `analysis` of `other manufacturers` or `factors` like `battery capacity` also, so that we can identify the `unique selling points` of `Tesla and Audi`.

Conclusion:

According to The `suggestion` of `Hypothesis Test` `engine power` is not major `differentiator` between `Tesla` and `Audi`, so it can not be a `key differentiator`.

Here is Python Project Explaination Video Link below:

https://drive.google.com/file/d/1aZXHkWLCVZm_tKN__rJI7KA-3sxCpOHS/view?usp=sharing