In [1]:
```python
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
```

Task 1

Customer with a budget of 350,000 PLN and wants an EV with a minimum range of 400 km

In [2]:
```python
# import the EVs data

EVs = pd.read_excel('FEV-data-Excel.xlsx')

EVs.head()
```

Out[2]:

	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]	...	Permissable gross weight [kg]	Maximum load capacity [kg]	Number of seats	Number of doors	Tire size [in]	Maximum speed [kph]	Boot capacity (VDA) [l]	Acceleration 0-100 kph [s]	Maximum DC charging power [kW]	mean - Energy consumption [kWh/100 km]
0	Audi e-tron 55 quattro	Audi	e-tron 55 quattro	345700	360	664	disc (front + rear)	4WD	95.0	438	...	3130.0	640.0	5	5	19	200	660.0	5.7	150	24.45
1	Audi e-tron 50 quattro	Audi	e-tron 50 quattro	308400	313	540	disc (front + rear)	4WD	71.0	340	...	3040.0	670.0	5	5	19	190	660.0	6.8	150	23.80
2	Audi e-tron S quattro	Audi	e-tron S quattro	414900	503	973	disc (front + rear)	4WD	95.0	364	...	3130.0	565.0	5	5	20	210	660.0	4.5	150	27.55
3	Audi e-tron Sportback 50 quattro	Audi	e-tron Sportback 50 quattro	319700	313	540	disc (front + rear)	4WD	71.0	346	...	3040.0	640.0	5	5	19	190	615.0	6.8	150	23.30
4	Audi e-tron Sportback 55 quattro	Audi	e-tron Sportback 55 quattro	357000	360	664	disc (front + rear)	4WD	95.0	447	...	3130.0	670.0	5	5	19	200	615.0	5.7	150	23.85

5 rows × 25 columns

In [7]:
```python
EVs.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 53 entries, 0 to 52
Data columns (total 25 columns):
 #   Column                                  Non-Null Count  Dtype  
---  ------                                  --------------  -----  
 0   Car full name                           53 non-null     object 
 1   Make                                    53 non-null     object 
 2   Model                                   53 non-null     object 
 3   Minimal price (gross) [PLN]             53 non-null     int64  
 4   Engine power [KM]                       53 non-null     int64  
 5   Maximum torque [Nm]                     53 non-null     int64  
 6   Type of brakes                          52 non-null     object 
 7   Drive type                              53 non-null     object 
 8   Battery capacity [kWh]                  53 non-null     float64
 9   Range (WLTP) [km]                       53 non-null     int64  
 10  Wheelbase [cm]                          53 non-null     float64
 11  Length [cm]                             53 non-null     float64
 12  Width [cm]                              53 non-null     float64
 13  Height [cm]                             53 non-null     float64
 14  Minimal empty weight [kg]               53 non-null     int64  
 15  Permissable gross weight [kg]           45 non-null     float64
 16  Maximum load capacity [kg]              45 non-null     float64
 17  Number of seats                         53 non-null     int64  
 18  Number of doors                         53 non-null     int64  
 19  Tire size [in]                          53 non-null     int64  
 20  Maximum speed [kph]                     53 non-null     int64  
 21  Boot capacity (VDA) [l]                 52 non-null     float64
 22  Acceleration 0-100 kph [s]              50 non-null     float64
 23  Maximum DC charging power [kW]          53 non-null     int64  
 24  mean - Energy consumption [kWh/100 km]  44 non-null     float64
dtypes: float64(10), int64(10), object(5)
memory usage: 10.5+ KB
```

In [3]:
```python
budget_range= EVs[(EVs['Minimal price (gross) [PLN]'] <= 350000) & (EVs['Range (WLTP) [km]'] >= 400)]

budget_range
```

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]	Range (WLTP) [km]	...	Permissable gross weight [kg]	Maximum load capacity [kg]	Number of seats	Number of doors	Tire size [in]	Maximum speed [kph]	Boot capacity (VDA) [l]	Acceleration 0-100 kph [s]	Maximum DC charging power [kW]	mean - Energy consumption [kWh/100 km]
0	Audi e-tron 55 quattro	Audi	e-tron 55 quattro	345700	360	664	disc (front + rear)	4WD	95.0	438	...	3130.0	640.0	5	5	19	200	660.0	5.7	150	24.45
8	BMW iX3	BMW	iX3	282900	286	400	disc (front + rear)	2WD (rear)	80.0	460	...	2725.0	540.0	5	5	19	180	510.0	6.8	150	18.80
15	Hyundai Kona electric 64kWh	Hyundai	Kona electric 64kWh	178400	204	395	disc (front + rear)	2WD (front)	64.0	449	...	2170.0	485.0	5	5	17	167	332.0	7.6	100	15.40
18	Kia e-Niro 64kWh	Kia	e-Niro 64kWh	167990	204	395	disc (front + rear)	2WD (front)	64.0	455	...	2230.0	493.0	5	5	17	167	451.0	7.8	100	15.90
20	Kia e-Soul 64kWh	Kia	e-Soul 64kWh	160990	204	395	disc (front + rear)	2WD (front)	64.0	452	...	1682.0	498.0	5	5	17	167	315.0	7.9	100	15.70
22	Mercedes-Benz EQC	Mercedes-Benz	EQC	334700	408	760	disc (front + rear)	4WD	80.0	414	...	2940.0	445.0	5	5	19	180	500.0	5.1	110	21.85
39	Tesla Model 3 Standard Range Plus	Tesla	Model 3 Standard Range Plus	195490	285	450	disc (front + rear)	2WD (rear)	54.0	430	...	NaN	NaN	5	5	18	225	425.0	5.6	150	NaN
40	Tesla Model 3 Long Range	Tesla	Model 3 Long Range	235490	372	510	disc (front + rear)	4WD	75.0	580	...	NaN	NaN	5	5	18	233	425.0	4.4	150	NaN
41	Tesla Model 3 Performance	Tesla	Model 3 Performance	260490	480	639	disc (front + rear)	4WD	75.0	567	...	NaN	NaN	5	5	20	261	425.0	3.3	150	NaN
47	Volkswagen ID.3 Pro Performance	Volkswagen	ID.3 Pro Performance	155890	204	310	disc (front) + drum (rear)	2WD (rear)	58.0	425	...	2270.0	540.0	5	5	18	160	385.0	7.3	100	15.40
48	Volkswagen ID.3 Pro S	Volkswagen	ID.3 Pro S	179990	204	310	disc (front) + drum (rear)	2WD (rear)	77.0	549	...	2280.0	412.0	5	5	19	160	385.0	7.9	125	15.90
49	Volkswagen ID.4 1st	Volkswagen	ID.4 1st	202390	204	310	disc (front) + drum (rear)	2WD (rear)	77.0	500	...	2660.0	661.0	5	5	20	160	543.0	8.5	125	18.00

12 rows × 25 columns

In [14]:
```python
grp=budget_range.groupby('Make')['Battery capacity [kWh]'].mean()
grp
```

Out[14]:
```
Make
Audi             71.000000
BMW              42.200000
Citroën          50.000000
DS               50.000000
Honda            35.500000
Hyundai          38.750000
Kia              39.200000
Mazda            35.500000
Mercedes-Benz    90.000000
Mini             28.900000
Nissan           47.333333
Opel             50.000000
Peugeot          50.000000
Renault          52.000000
Skoda            36.800000
Smart            17.600000
Volkswagen       32.300000
Name: Battery capacity [kWh], dtype: float64
```

task2

In [15]:
```python
# 2: Identify outliers using the IQR method

# Calculate Percentiles
percentiles_energy_consumption = EVs["mean - Energy consumption [kWh/100 km]"].quantile([0.25,0.5,0.75])

percentiles_energy_consumption
```

Out[15]:
```
0.25    15.60
0.50    17.05
0.75    23.50
Name: mean - Energy consumption [kWh/100 km], dtype: float64
```

In [26]:
```python
q3=23.50
q1=15.60
iqr = q3 - q1 # Interquartile Range (IQR)

lower_bound = q1 - 3 * iqr  #formula for lower bound of outlier
upper_bound = q3 + 3 * iqr  #formula for upper bound of outlier
print(f'lower bound is {lower_bound}, upper bound is {upper_bound}')
extreme_outliers = EVs[(EVs['mean - Energy consumption [kWh/100 km]'] <= lower_bound) | (EVs['mean - Energy consumption [kWh/100 km]'] >= upper_bound)]
extreme_outliers
```

```
lower bound is -8.100000000000003, upper bound is 47.2
```

Out[26]:

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]	...	Permissable gross weight [kg]	Maximum load capacity [kg]	Number of seats	Number of doors	Tire size [in]	Maximum speed [kph]	Boot capacity (VDA) [l]	Acceleration 0-100 kph [s]	Maximum DC charging power [kW]	mean - Energy consumption [kWh/100 km]

0 rows × 25 columns

Task 3

relationship between battery capacity and range

In [5]:
```python
correlation=EVs[['Battery capacity [kWh]','Range (WLTP) [km]']].corr()
```

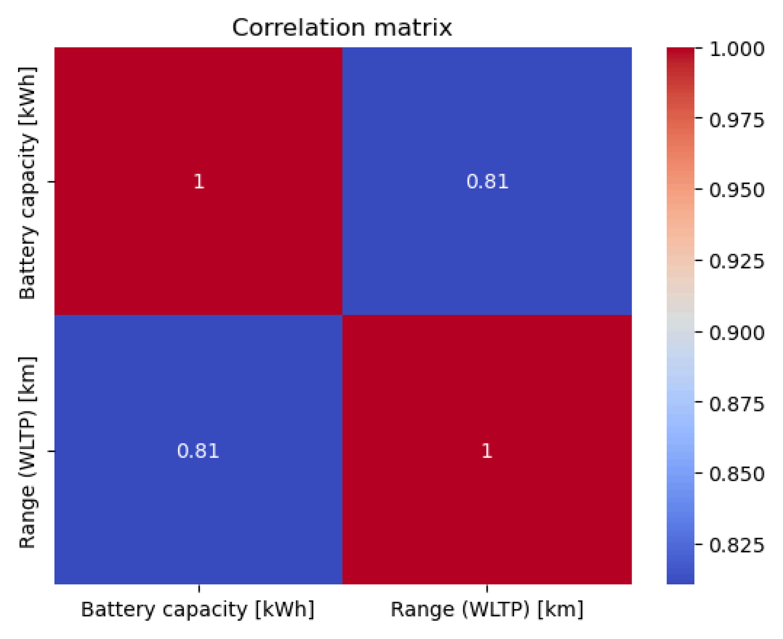
In [6]:
```python
correlation
```

Out[6]:

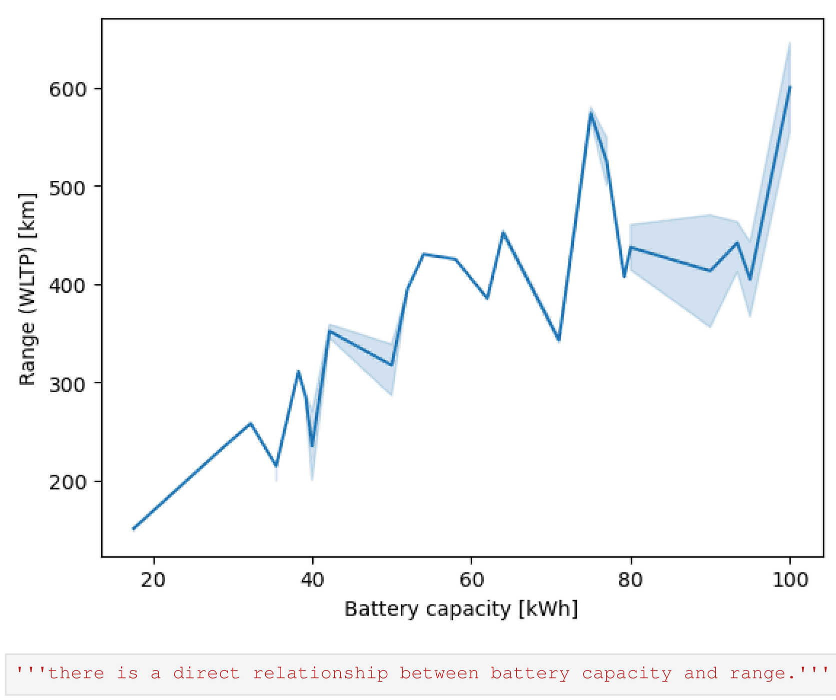
	Battery capacity [kWh]	Range (WLTP) [km]
Battery capacity [kWh]	1.000000	0.810439
Range (WLTP) [km]	0.810439	1.000000

In [10]:
```python
sns.heatmap(correlation,annot=True,cmap='coolwarm')
plt.title('Correlation matrix')
plt.show
```

Out[10]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```



In [7]:
```python
sns.lineplot(x='Battery capacity [kWh]',y='Range (WLTP) [km]',data=EVs)
plt.show()
```



In [ ]:
```python
'''there is a direct relationship between battery capacity and range.'''
```

Task 4

EV recommendation class

In [33]:
```python
class recommendation_:
    def __init__(self,a,b,c):
        self.a=a     #to store budget
        self.b=b     #to store desired range
        self.c=c     #to battery capacity
        if self.a==0:
            print("enter valid budget!!")
        elif self.b==0:
            print("enter valid range!!")
        elif self.c==0:
            print("enter valid battery capacity!!")
        else:
            print("TOP 3 EVs with these parameters are:")
            top_=EVs[(EVs['Minimal price (gross) [PLN]'] <= self.a) & (EVs['Range (WLTP) [km]'] >= self.b) & (EVs['Battery capacity [kWh]'] <= self.c)]
            top_affordable=top_.sort_values(by="Minimal price (gross) [PLN]")
            print(top_affordable.head(3))
x=int(input("enter your budget"))
y=int(input("enter your desired range"))
z=int(input("enter battery capacity"))
r=recommendation_(x,y,z)
```

```
enter your budget120000
enter your desired range50
enter battery capacity50
TOP 3 EVs with these parameters are:
        Car full name        Make        Model  Minimal price (gross) [PLN]  \
36  Skoda Citigo-e iV       Skoda  Citigo-e iV                        82050   
37    Smart fortwo EQ       Smart    fortwo EQ                        96900   
46   Volkswagen e-up!  Volkswagen        e-up!                        97990   

    Engine power [KM]  Maximum torque [Nm]              Type of brakes  \
36                 83                  212  disc (front) + drum (rear)   
37                 82                  160  disc (front) + drum (rear)   
46                 83                  210  disc (front) + drum (rear)   

     Drive type  Battery capacity [kWh]  Range (WLTP) [km]  ...  \
36  2WD (front)                    36.8                260  ...   
37   2WD (rear)                    17.6                154  ...   
46  2WD (front)                    32.3                258  ...   

    Permissable gross weight [kg]  Maximum load capacity [kg]  \
36                         1530.0                       367.0   
37                         1310.0                       290.0   
46                         1530.0                       370.0   

    Number of seats  Number of doors  Tire size [in]  Maximum speed [kph]  \
36                4                5              14                  130   
37                2                3              15                  130   
46                4                5              14                  130   

    Boot capacity (VDA) [l]  Acceleration 0-100 kph [s]  \
36                    250.0                        12.3   
37                    185.0                        11.6   
46                    250.0                        11.9   

    Maximum DC charging power [kW]  mean - Energy consumption [kWh/100 km]  
36                              40                                   15.45  
37                              22                                   16.35  
46                              40                                   14.00  

[3 rows x 25 columns]
```

In [ ]:
```python
'''these three cars are within the required budget with required range and battery capacity '''
```

Task 5

Testing whether there is a significant difference in the average Engine power [KM] of vehicles manufactured by two leading manufacturers i.e. Tesla and Audi

In [ ]:
```python
import warnings
warnings.filterwarnings('ignore',category=DeprecationWarning)# to ignore Deprecation warnings
```

In [58]:
```python
# H0: there is a significant difference in the average Engine power [KM] of vehicles
# H1 : No difference
from scipy.stats import*
audi_eng=EVs[EVs['Make']=="Audi"]
tes_eng=EVs[EVs['Make']=="Tesla"]
#since sample size is small therefore T test can be applied for comparing means of two group
stat,p=stats.ttest_ind(audi_eng['Engine power [KM]'],tes_eng['Engine power [KM]'],alternative='two-sided')
print(f"stats={stat} and p value ={p}")
alpha=0.05
if p<alpha:
    print("we reject the null hypothesis. There NO significant difference in average engine power of vehicles manufactured by Audi and Tesla")
else:
    print("There is a significant difference in average engine power of vehicles manufactured by Audi and Tesla")
```

```
stats=-1.7024444538261416 and p value =0.11672692675082785
There is a significant difference in average engine power of vehicles manufactured by Audi and Tesla
```

task6

In [ ]:
```
https://drive.google.com/file/d/17kdhIYYqcU4gaVjr-ukoyH2nmOJqCZML/view?usp=drivesdk
```