In [5]:
```
import pandas as pd
import numpy as np
```

In [7]:
```
FEV_df=pd.read_excel("FEV-data-Excel.xlsx")
print(FEV_df.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
None
```

In [27]:
```
#Task 1: A customer has a budget of 350,000 PLN and wants an EV with a minimum range of 400 km.
#a) Your task is to filter out EVs that meet these criteria.
#b) Group them by the manufacturer (Make)
#c) Calculate the average battery capacity for each manufacturer.

EV_name=FEV_df[(FEV_df["Minimal price (gross) [PLN]"]<=350000) & (FEV_df["Range (WLTP) [km]"]>=400)]
avg_bat_cap=EV_name.groupby("Make")["Battery capacity [kWh]"].mean()
print(avg_bat_cap)
```

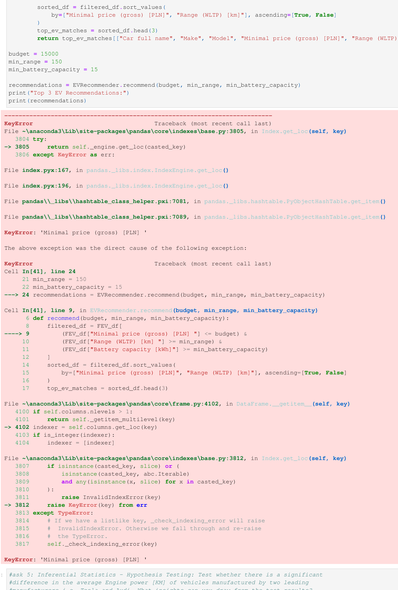
```
Make
Audi             95.000000
BMW              80.000000
Hyundai          64.000000
Kia              64.000000
Mercedes-Benz    80.000000
Tesla            68.000000
Volkswagen       70.666667
Name: Battery capacity [kWh], dtype: float64
```

In [15]:
```
# You suspect some EVs have unusually high or low energy consumption. Find the outliers in the mean - Energy consumption [kWh/100 km] column
column_name = "mean - Energy consumption [kWh/100 km]"
data = FEV_df[column_name]
Q1 = data.quantile(0.25)
Q3 = data.quantile(0.75)
IQR = Q3 - Q1

lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

outliers = FEV_df[(data < lower_bound) | (data > upper_bound)]

print("Outliers based on IQR method:")
print(outliers)
```

```
Outliers based on IQR method:
Empty DataFrame
Columns: [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], Wheelbase [cm], Length [cm], Width [cm], Height [cm], Minimal empty weight [kg], 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]]
Index: []

[0 rows x 25 columns]
```

In [37]:
```
#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.
# find correlation coefficient for realtion between two table from table we get mean and standard deviation so from
# correlation coefficnt formula we get it close to 1 so this is strong positive correlation.
print(FEV_df.describe())

import matplotlib.pyplot as plt
import seaborn as sns

battery_capacity = FEV_df["Battery capacity [kWh]"]
range_wltp = FEV_df["Range (WLTP) [km]"]

plt.figure(figsize=(10, 6))
sns.scatterplot(x=battery_capacity, y=range_wltp, hue=FEV_df["Make"], palette="viridis", s=100)

sns.regplot(x=battery_capacity, y=range_wltp, scatter=False, color="red", ci=None)

plt.title("Relationship Between Battery Capacity and Range (WLTP)", fontsize=16)
plt.xlabel("Battery Capacity [kWh]", fontsize=12)
plt.ylabel("Range (WLTP) [Km]", fontsize=12)
plt.grid(True)
plt.legend(title="Make")
plt.show()

correlation = battery_capacity.corr(range_wltp)
print(f"Correlation Coefficient: {correlation:.2f}")
```

```
       Minimal price (gross) [PLN]  Engine power [KM]  Maximum torque [Nm]  \
count                    53.000000          53.000000            53.000000   
mean                 246158.509434         269.773585           460.037736   
std                  149187.485190         181.298589           261.647000   
min                      82050.000000          82.000000           160.000000   
25%                  142900.000000         136.000000           260.000000   
50%                  178400.000000         204.000000           362.000000   
75%                  339480.000000         372.000000           640.000000   
max                  794000.000000         772.000000          1140.000000   

       Battery capacity [kWh]  Range (WLTP) [km]  Wheelbase [cm]  Length [cm]  \
count               53.000000          53.000000       53.000000    53.000000   
mean                62.366038         376.905660      273.581132   442.509434   
std                 24.170913         118.817938       22.740518    48.863280   
min                 17.600000         148.000000      187.300000   269.500000   
25%                 40.000000         289.000000      258.800000   411.800000   
50%                 58.000000         364.000000      270.000000   447.000000   
75%                 80.000000         450.000000      290.000000   490.100000   
max                100.000000         652.000000      327.500000   514.000000   

       Width [cm]  Height [cm]  Minimal empty weight [kg]  \
count   53.000000   53.000000                  53.000000   
mean   186.241509  155.422642                1868.452830   
std     14.280641   11.275358                 470.880867   
min    164.500000  137.800000                1035.000000   
25%    178.800000  148.100000                1530.000000   
50%    180.900000  155.600000                1685.000000   
75%    193.500000  161.500000                2370.000000   
max    255.800000  191.000000                2710.000000   

       Permissable gross weight [kg]  Maximum load capacity [kg]  \
count                      45.000000                   45.000000   
mean                     2288.844444                  520.466667   
std                       557.796026                  140.682848   
min                      1310.000000                  290.000000   
25%                      1916.000000                  440.000000   
50%                      2115.000000                  486.000000   
75%                      2870.000000                  575.000000   
max                      3500.000000                 1056.000000   

       Number of seats  Number of doors  Tire size [in]  Maximum speed [kph]  \
count        53.000000        53.000000       53.000000            53.000000   
mean          4.905660         4.849057       17.679245           178.169811   
std           0.838133         0.455573        1.868500            43.056196   
min           2.000000         3.000000       14.000000           123.000000   
25%           5.000000         5.000000       16.000000           150.000000   
50%           5.000000         5.000000       17.000000           160.000000   
75%           5.000000         5.000000       19.000000           200.000000   
max           8.000000         5.000000       21.000000           261.000000   

       Boot capacity (VDA) [l]  Acceleration 0-100 kph [s]  \
count                52.000000                   50.00000   
mean                465.096154                    7.36000   
std                 180.178480                    2.78663   
min                 171.000000                    2.50000   
25%                 315.000000                    4.87500   
50%                 425.000000                    7.70000   
75%                 558.000000                    9.37500   
max                 870.000000                   13.10000   

       Maximum DC charging power [kW]  mean - Energy consumption [kWh/100 km]  
count                       53.000000                               44.000000  
mean                       113.509434                               18.994318  
std                         57.166970                                4.418253  
min                         22.000000                               13.100000  
25%                        100.000000                               15.400000  
50%                        100.000000                               17.050000  
75%                        150.000000                               23.500000  
max                        270.000000                               28.200000  
```




```
Correlation Coefficient: 0.81
```

In [41]:
```
#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.
import pandas as pd
class EVRecommender:

    def recommend(budget, min_range, min_battery_capacity):

        filtered_df = FEV_df[
            (FEV_df["Minimal price (gross) [PLN] "] <= budget) &
            (FEV_df["Range (WLTP) [km] "] >= min_range) &
            (FEV_df["Battery capacity [kWh]"] >= min_battery_capacity)
        ]


        sorted_df = filtered_df.sort_values(
            by="Minimal price (gross) [PLN]", "Range (WLTP) [km]"], ascending=[True, False]
        )
        top_ev_matches = sorted_df.head(3)
        return top_ev_matches[["Car full name", "Make", "Model", "Minimal price (gross) [PLN]", "Range (WLTP) [km]", "Battery capacity [kWh]"]]

budget = 15000
min_range = 150
min_battery_capacity = 15

recommendations = EVRecommender.recommend(budget, min_range, min_battery_capacity)
print("Top 3 EV Recommendations:")
print(recommendations)
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
File ~\anaconda3\Lib\site-packages\pandas\core\indexes\base.py:3805, in Index.get_loc(self, key)
   3804 try:
-> 3805     return self._engine.get_loc(casted_key)
   3806 except KeyError as err:

File index.pyx:167, in pandas._libs.index.IndexEngine.get_loc()

File index.pyx:196, in pandas._libs.index.IndexEngine.get_loc()

File pandas\\_libs\hashtable_class_helper.pxi:7081, in pandas._libs.hashtable.PyObjectHashTable.get_item()

File pandas\\_libs\hashtable_class_helper.pxi:7089, in pandas._libs.hashtable.PyObjectHashTable.get_item()

KeyError: 'Minimal price (gross) [PLN] '

The above exception was the direct cause of the following exception:

KeyError                                  Traceback (most recent call last)
Cell In[41], line 24
     21 min_range = 150
     22 min_battery_capacity = 15
---> 24 recommendations = EVRecommender.recommend(budget, min_range, min_battery_capacity)

Cell In[41], line 9, in EVRecommender.recommend(budget, min_range, min_battery_capacity)
      6 def recommend(budget, min_range, min_battery_capacity):
      8     filtered_df = FEV_df[
----> 9         (FEV_df["Minimal price (gross) [PLN] "] <= budget) &
     10         (FEV_df["Range (WLTP) [km] "] >= min_range) &
     11         (FEV_df["Battery capacity [kWh]"] >= min_battery_capacity)
     12     ]
     14     sorted_df = filtered_df.sort_values(
     15         by="Minimal price (gross) [PLN]", "Range (WLTP) [km]"], ascending=[True, False]
     16     )
     17     top_ev_matches = sorted_df.head(3)

File ~\anaconda3\Lib\site-packages\pandas\core\frame.py:4102, in DataFrame.__getitem__(self, key)
   4100 if self.columns.nlevels > 1:
   4101     return self._getitem_multilevel(key)
-> 4102 indexer = self.columns.get_loc(key)
   4103 if is_integer(indexer):
   4104     indexer = [indexer]

File ~\anaconda3\Lib\site-packages\pandas\core\indexes\base.py:3812, in Index.get_loc(self, key)
   3807     if isinstance(casted_key, slice) or (
   3808         isinstance(casted_key, abc.Iterable)
   3809         and any(isinstance(x, slice) for x in casted_key)
   3810     ):
   3811         raise InvalidIndexError(key)
-> 3812     raise KeyError(key) from err
   3813 except TypeError:
   3814     # If we have a listlike key, _check_indexing_error will raise
   3815     #  InvalidIndexError. Otherwise we fall through and re-raise
   3816     #  the TypeError.
   3817     self._check_indexing_error(key)

KeyError: 'Minimal price (gross) [PLN] '
```

In [45]:
```
#ask 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)
from scipy.stats import ttest_ind

tesla_power =FEV_df[FEV_df["Make"].str.lower() == "tesla"]["Engine power [KM]"]
audi_power =FEV_df[FEV_df["Make"].str.lower() == "audi"]["Engine power [KM]"]

t_stat, p_value = ttest_ind(tesla_power, audi_power, equal_var=False)

print(f"T-Statistic: {t_stat:.2f}")
print(f"P-Value: {p_value:.4f}")

alpha = 0.05
if p_value < alpha:
    print("Reject the null hypothesis: There is a significant difference in average engine power.")
else:
    print("Fail to reject the null hypothesis: No significant difference in average engine power.")
```

```
T-Statistic: 1.79
P-Value: 0.1068
Fail to reject the null hypothesis: No significant difference in average engine power.
```

In [ ]:
```
#video drive link
[Watch the Analysis Video](https://drive.google.com/file/d/1kcRTOpXUDmv9f2J-7DFBpJfdjc4r3kI/view?usp=sharing)
```