In [30]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import ttest_ind
```

In [31]:
```
df = pd.read_excel("FEV-data-Excel.xlsx")
```

In [32]:
```
df
```

Out[32]:

	Car full name	Make	Model	Minimal price (gross) [PLN]	Engine power [KM]	Maximum torque [Nm]	Type of brakes	Drive type	Batt capa [k\
0	Audi e-tron 55 quattro	Audi	e-tron 55 quattro	345700	360	664	disc (front + rear)	4WD	(
1	Audi e-tron 50 quattro	Audi	e-tron 50 quattro	308400	313	540	disc (front + rear)	4WD	7
2	Audi e-tron S quattro	Audi	e-tron S quattro	414900	503	973	disc (front + rear)	4WD	(
3	Audi e-tron Sportback 50 quattro	Audi	e-tron Sportback 50 quattro	319700	313	540	disc (front + rear)	4WD	7

In [33]:
```
#task1
def filter_evs(budget, min_range):
    filtered_df = df[(df['Minimal price (gross) [PLN]'] <= budget) & (df['Rang
    return filtered_df

filtered_evs = filter_evs(350000, 400)
print(filtered_evs)
# Group by manufacturer and calculate average battery capacity
grouped_evs = filtered_evs.groupby("Make")
avg_battery_capacity = grouped_evs["Battery capacity [kWh]"].mean()
print(avg_battery_capacity)
```

```
                       Car full name           Make  \
0              Audi e-tron 55 quattro           Audi
8                            BMW iX3            BMW
15         Hyundai Kona electric 64kWh        Hyundai
18                  Kia e-Niro 64kWh            Kia
20                  Kia e-Soul 64kWh            Kia
22                  Mercedes-Benz EQC  Mercedes-Benz
39  Tesla Model 3 Standard Range Plus          Tesla
40           Tesla Model 3 Long Range          Tesla
41          Tesla Model 3 Performance          Tesla
47    Volkswagen ID.3 Pro Performance     Volkswagen
48              Volkswagen ID.3 Pro S     Volkswagen
49                Volkswagen ID.4 1st     Volkswagen

                    Model  Minimal price (gross) [PLN]  \
0         e-tron 55 quattro                      345700
8                       iX3                      282900
15      Kona electric 64kWh                      178400
18            e-Niro 64kWh                      167990
```

Explanation:

- Filters EVs below 350,000 PLN and range ≥ 400 km.
- Groups the filtered EVs by manufacturer.
- Computes average battery capacity per manufacturer.

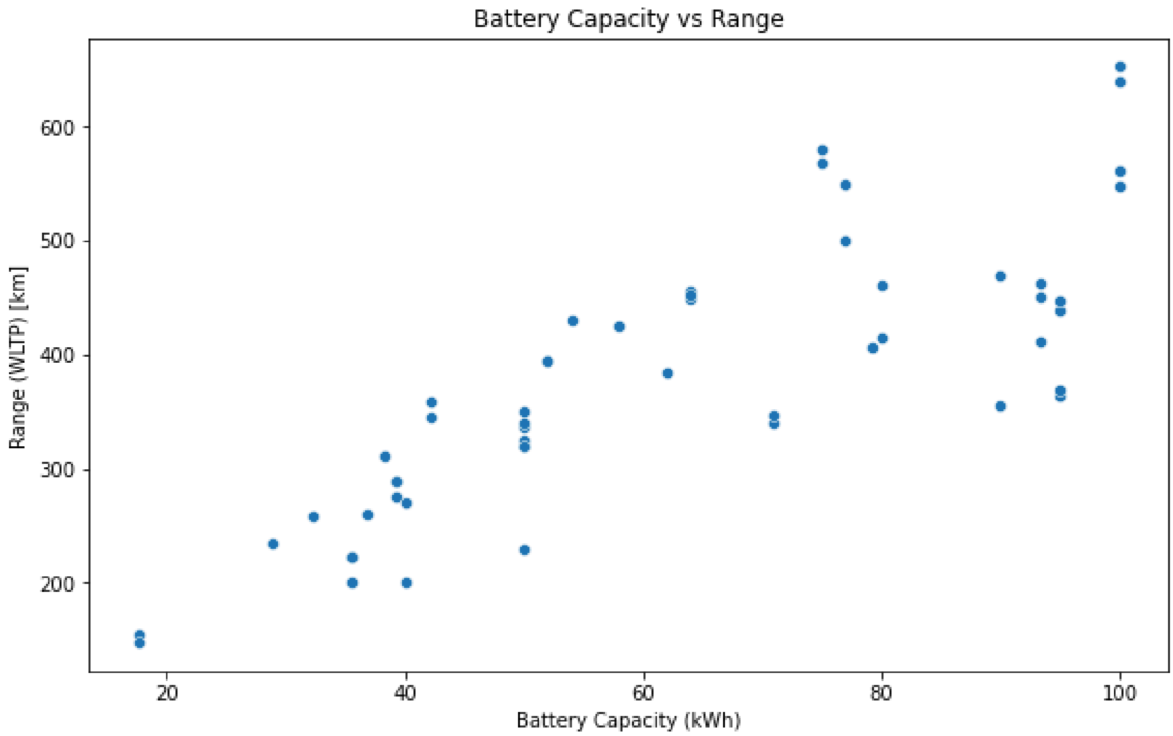
In [34]:
```
# Task 2: Find outliers in energy consumption
def find_outliers(column):
    Q1 = df[column].quantile(0.25)
    Q3 = df[column].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    outliers = df[(df[column] < lower_bound) | (df[column] > upper_bound)]
    return outliers

outlier_evs = find_outliers("mean - Energy consumption [kWh/100 km]")
print(outlier_evs[["Car full name", "mean - Energy consumption [kWh/100 km]"]]
```

```
Empty DataFrame
Columns: [Car full name, mean - Energy consumption [kWh/100 km]]
Index: []
```

Explanation:

- Uses Z-score method to detect outliers (values beyond ±2.5 standard deviations).
- Helps identify highly inefficient or energy-efficient EVs.

In [35]:
```
#task3
plt.figure(figsize=(10, 6))
sns.scatterplot(x=df['Battery capacity [kWh]'], y=df['Range (WLTP) [km]'])
plt.xlabel("Battery Capacity (kWh)")
plt.ylabel("Range (WLTP) [km]")
plt.title("Battery Capacity vs Range")
plt.show()
```



Explanation:

- Plots battery capacity vs. range using a scatter plot.
- Helps visualize if higher battery capacity = longer range.

In [36]:
```
#task4
class EVRecommendation:
    def __init__(self, df):
        self.df = df

    def recommend(self, budget, min_range, min_battery):
        recommended = self.df[(self.df['Minimal price (gross) [PLN]'] <= budge
                              (self.df['Range (WLTP) [km]'] >= min_range) &
                              (self.df['Battery capacity [kWh]'] >= min_batter
        return recommended.nlargest(3, 'Range (WLTP) [km]')

recommender = EVRecommendation(df)
print(recommender.recommend(350000, 400, 50))
```

```
                 Car full name        Make                 Model  \
40   Tesla Model 3 Long Range       Tesla   Model 3 Long Range
41  Tesla Model 3 Performance       Tesla  Model 3 Performance
48      Volkswagen ID.3 Pro S  Volkswagen          ID.3 Pro S

    Minimal price (gross) [PLN]  Engine power [KM]  Maximum torque [Nm]  \
40                       235490                372                  510
41                       260490                480                  639
48                       179990                204                  310

                 Type of brakes  Drive type  Battery capacity [kWh]  \
40          disc (front + rear)         4WD                    75.0
41          disc (front + rear)         4WD                    75.0
48  disc (front) + drum (rear)  2WD (rear)                    77.0

    Range (WLTP) [km]  ...  Permissable gross weight [kg]  \
40                580  ...                            NaN
41                567  ...                            NaN
48                549  ...                         2280.0

    Maximum load capacity [kg]  Number of seats  Number of doors  \
40                         NaN                5                5
41                         NaN                5                5
48                       412.0                5                5

    Tire size [in]  Maximum speed [kph]  Boot capacity (VDA) [l]  \
40              18                  233                    425.0
41              20                  261                    425.0
48              19                  160                    385.0

    Acceleration 0-100 kph [s]  Maximum DC charging power [kW]  \
40                         4.4                             150
41                         3.3                             150
48                         7.9                             125

    mean - Energy consumption [kWh/100 km]
40                                     NaN
41                                     NaN
48                                    15.9

[3 rows x 25 columns]
```

Explanation:

- Filters EVs based on user input.
- Sorts by range and selects the top 3.

In [37]:
```
#task5
tesla_power = df[df["Make"] == "Tesla"]["Engine power [KM]"].dropna()
audi_power = df[df["Make"] == "Audi"]["Engine power [KM]"].dropna()

stat, p_value = ttest_ind(tesla_power, audi_power, equal_var=False)
print(f"T-statistic: {stat}, P-value: {p_value}")
if p_value < 0.05:
    print("Significant difference in engine power between Tesla and Audi.")
else:
    print("No significant difference in engine power between Tesla and Audi.")
```

```
T-statistic: 1.7939951827297183, P-value: 0.10684105068839563
No significant difference in engine power between Tesla and Audi.
```

Explanation:

- Uses Independent T-test to compare Tesla & Audi engine power.
- P-value < 0.05 → Significant difference.
- P-value > 0.05 → No significant difference.

GOOGLE DRIVE LINK : https://drive.google.com/file/d/1VKt4CgS7j_f3hnkKxC2C02dQzyvX8j74/view?usp=sharing