In [6]:
```
## 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.




import pandas as pd

# Load the dataset
df = pd.read_excel("Electric Vehicle.xlsx")

# Filter EVs that meet the criteria
filtered_evs = df[(df["Minimal price (gross) [PLN]"] <= 350000) & (df["Range (WLTP) [km]"] >= 400)]

# Group by manufacturer and calculate average battery capacity
manufacturer_avg_battery = (filtered_evs.groupby("Make")["Battery capacity [kWh]"].mean()
                            .reset_index().rename(columns={"Battery capacity [kWh]": "Average Battery Capacity (kWh)"}))

# Display the filtered results and battery capacity by manufacturer
print("Filtered EVs:")
print(filtered_evs[["Car full name", "Make", "Minimal price (gross) [PLN]", "Range (WLTP) [km]", "Battery capacity [kWh]"]])

print("\nAverage Battery Capacity by Manufacturer:")
print(manufacturer_avg_battery)
```

```
Filtered EVs:
                        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

    Minimal price (gross) [PLN]  Range (WLTP) [km]  Battery capacity [kWh]
0                        345700                438                    95.0
8                        282900                460                    80.0
15                       178400                449                    64.0
18                       167990                455                    64.0
20                       160990                452                    64.0
22                       334700                414                    80.0
39                       195490                430                    54.0
40                       235490                580                    75.0
41                       260490                567                    75.0
47                       155890                425                    58.0
48                       179990                549                    77.0
49                       202390                500                    77.0

Average Battery Capacity by Manufacturer:
           Make  Average 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
```

In [ ]:
```

```

In [14]:
```
## 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.

import numpy as np
Q1 = df["mean - Energy consumption [kWh/100 km]"].quantile(0.25)
Q3 = df["mean - Energy consumption [kWh/100 km]"].quantile(0.75)
IQR = Q3 - Q1

# Define lower and upper bounds for outliers
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

# Identify outliers
df_outliers = df[(df["mean - Energy consumption [kWh/100 km]"] < lower_bound) | (df["mean - Energy consumption [kWh/100 km]"] > upper_bound)]

# Display outliers
print(df_outliers)
```

```
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]
```

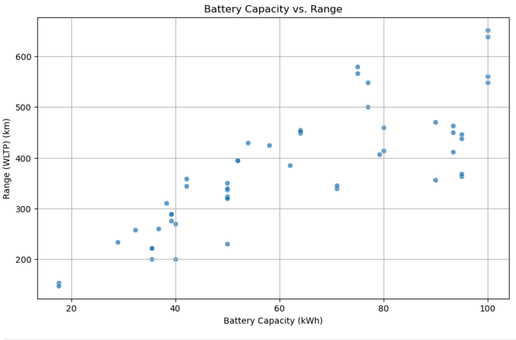
In [ ]:
```

```

In [16]:
```
## 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.


import matplotlib.pyplot as plt
import seaborn as sns
plt.figure(figsize=(10, 6))
sns.scatterplot(x=df["Battery capacity [kWh]"], y=df["Range (WLTP) [km]"], alpha=0.7)
plt.xlabel("Battery Capacity (kWh)")
plt.ylabel("Range (WLTP) (km)")
plt.title("Battery Capacity vs. Range")
plt.grid(True)
plt.show()

# Insights:
# From the scatter plot, we can observe if there is a positive correlation between battery capacity and range.
# If the points follow an upward trend, this suggests that larger battery capacities generally lead to longer ranges.
# However, some outliers may indicate EVs with highly efficient or inefficient energy consumption.
```



In [ ]:
```

```

In [17]:
```
## 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.


class EVRecommender:
    def __init__(self, dataframe):
        self.df = dataframe

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

        return recommended.sort_values(by=["Range (WLTP) [km]"], ascending=False).head(3)

# Create an instance of the recommender
ev_recommender = EVRecommender(df)

# Example usage:
user_budget = 300000
user_range = 350
user_battery = 60
print(ev_recommender.recommend(user_budget, user_range, user_battery))
```

```
                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]
```

In [ ]:
```

```

In [18]:
```
# 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)


from scipy.stats import ttest_ind

tesla_power = df[df["Make"] == "Tesla"]["Engine power [KM]"].dropna()
audi_power = df[df["Make"] == "Audi"]["Engine power [KM]"].dropna()

# Perform independent two-sample t-test
t_stat, p_value = ttest_ind(tesla_power, audi_power, equal_var=False)  # Welch's t-test

print(f"T-statistic: {t_stat}")
print(f"P-value: {p_value}")

# Insights:
# - If the p-value is below 0.05, we reject the null hypothesis and conclude that there is a significant difference
#   in average engine power between Tesla and Audi.
```

```
# - If the p-value is above 0.05, we fail to reject the null hypothesis, meaning there is no significant difference.
# - This analysis helps us understand whether Tesla and Audi offer significantly different engine power in their EVs.
```

```
T-statistic: 1.7939951827297178
P-value: 0.10684105068839565
```