Project Video Explanation

Click here to watch the video

In [5]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import ttest_ind
```

In [6]:
```
df = pd.read_excel("FEV-data-Excel.xlsx")
```

In [8]:
```
df.head()
df.info()
df.describe()
df.isnull().sum()
```

```
<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
```

Out[8]:

	0
Car full name	0
Make	0
Model	0
Minimal price (gross) [PLN]	0
Engine power [KM]	0
Maximum torque [Nm]	0
Type of brakes	1
Drive type	0
Battery capacity [kWh]	0
Range (WLTP) [km]	0
Wheelbase [cm]	0
Length [cm]	0
Width [cm]	0
Height [cm]	0
Minimal empty weight [kg]	0
Permissable gross weight [kg]	8
Maximum load capacity [kg]	8
Number of seats	0
Number of doors	0
Tire size [in]	0
Maximum speed [kph]	0
Boot capacity (VDA) [l]	1
Acceleration 0-100 kph [s]	3
Maximum DC charging power [kW]	0
mean - Energy consumption [kWh/100 km]	9

dtype: int64

In [18]:

```
df.head(5)
```

Out[18]:

	Car_Name	Make	Model	Price_PLN	Engine power [KM]	Maximum torque [Nm]	Type of brakes	Drive type	Battery_kWh
0	Audi e-tron 55 quattro	Audi	e-tron 55 quattro	345700	360	664	disc (front + rear)	4WD	95.0
1	Audi e-tron 50 quattro	Audi	e-tron 50 quattro	308400	313	540	disc (front + rear)	4WD	71.0
2	Audi e-tron S quattro	Audi	e-tron S quattro	414900	503	973	disc (front + rear)	4WD	95.0
3	Audi e-tron Sportback 50 quattro	Audi	e-tron Sportback 50 quattro	319700	313	540	disc (front + rear)	4WD	71.0
4	Audi e-tron Sportback 55 quattro	Audi	e-tron Sportback 55 quattro	357000	360	664	disc (front + rear)	4WD	95.0

5 rows × 25 columns

In [16]:

```
df.rename(columns={
    'Car full name': 'Car_Name',
    'Minimal price (gross) [PLN]': 'Price_PLN',
    'Battery capacity [kWh]': 'Battery_kWh',
    'Range (WLTP) [km]': 'Range_km'
}, inplace=True)
```

Task 1 Filtering & Grouping EVs

In [19]:

```
budget = 350000
min_range = 400

filtered = df[(df['Price_PLN'] <= budget) & (df['Range_km'] >= min_range)]
print("Number of EVs matching criteria:", filtered.shape[0])
filtered[['Car_Name', 'Price_PLN', 'Range_km', 'Battery_kWh']].head()
```

```
Number of EVs matching criteria: 12
```

Out[19]:

	Car_Name	Price_PLN	Range_km	Battery_kWh
0	Audi e-tron 55 quattro	345700	438	95.0
8	BMW iX3	282900	460	80.0
15	Hyundai Kona electric 64kWh	178400	449	64.0
18	Kia e-Niro 64kWh	167990	455	64.0
20	Kia e-Soul 64kWh	160990	452	64.0

In [20]:

```
grouped = filtered.groupby('Make')
grouped.size().reset_index(name = 'Model_count')
```

Out[20]:

	Make	Model_count
0	Audi	1
1	BMW	1
2	Hyundai	1
3	Kia	2
4	Mercedes-Benz	1
5	Tesla	3
6	Volkswagen	3

In [21]:

```
avg_battery = grouped['Battery_kWh'].mean().reset_index(name = 'Avg_Battery_kWh')
avg_battery
```

Out[21]:

	Make	Avg_Battery_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 [22]:

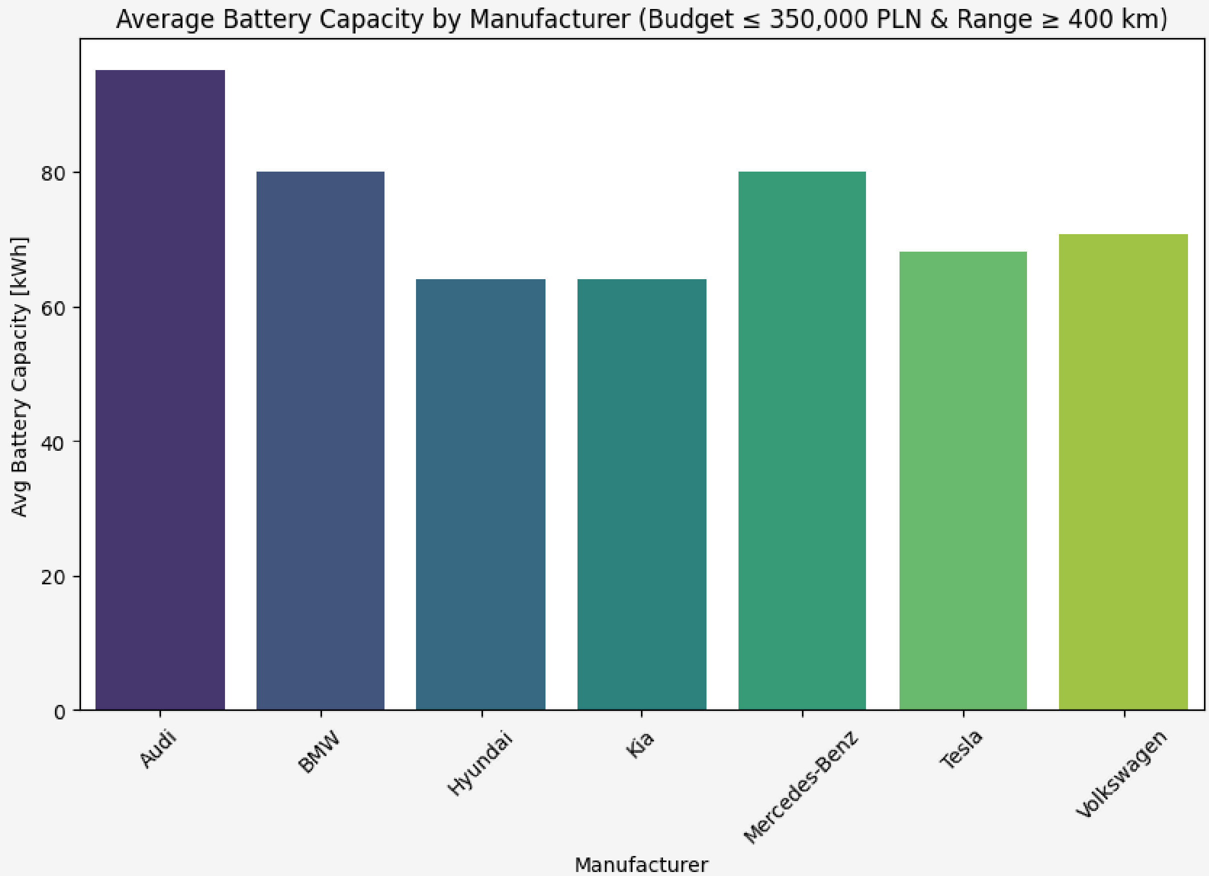
```
plt.figure(figsize=(10,6), facecolor='whitesmoke')
sns.barplot(x='Make', y='Avg_Battery_kWh', data=avg_battery, palette='viridis')
plt.title("Average Battery Capacity by Manufacturer (Budget ≤ 350,000 PLN & Range ≥
plt.xlabel("Manufacturer")
plt.ylabel("Avg Battery Capacity [kWh]")
```

```
plt.xticks(rotation=45)
plt.show()
```

```
/tmp/ipython-input-1798422117.py:2: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.1
4.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.barplot(x='Make', y='Avg_Battery_kWh', data=avg_battery, palette='viridis')
```



Insights – Task 1 (Filtering & Grouping EVs)

1. Out of the full dataset, **12 EVs** meet the criteria of budget ≤ 350,000 PLN and range ≥ 400 km.
2. **Tesla** has the most options (3 models) under the given budget, followed by **Volkswagen** (3 models).
3. **Audi** and **Mercedes-Benz** have premium EVs that just fit within the budget, offering larger battery packs.
4. **Hyundai** and **Kia** provide affordable EVs with smaller battery capacities (~64 kWh) but still deliver ranges above 450 km.
5. **Volkswagen ID series** (ID.3, ID.4) strike a balance between affordability and long driving range.

Recommendations – Task 1

1. **For Long-Range Seekers:** Tesla Model 3 Long Range is the best fit (580 km range, within budget).
2. **For Performance-Oriented Buyers:** Tesla Model 3 Performance provides strong power with good range.
3. **For Budget-Conscious Customers:** Kia e-Niro, Kia e-Soul, and Hyundai Kona Electric offer excellent range with lower prices.
4. **For Balanced Choice:** Volkswagen ID.3 Pro S offers competitive range (~549 km) and good efficiency at a moderate cost.

Task 2 (Outlier Detection in Energy Consumption)

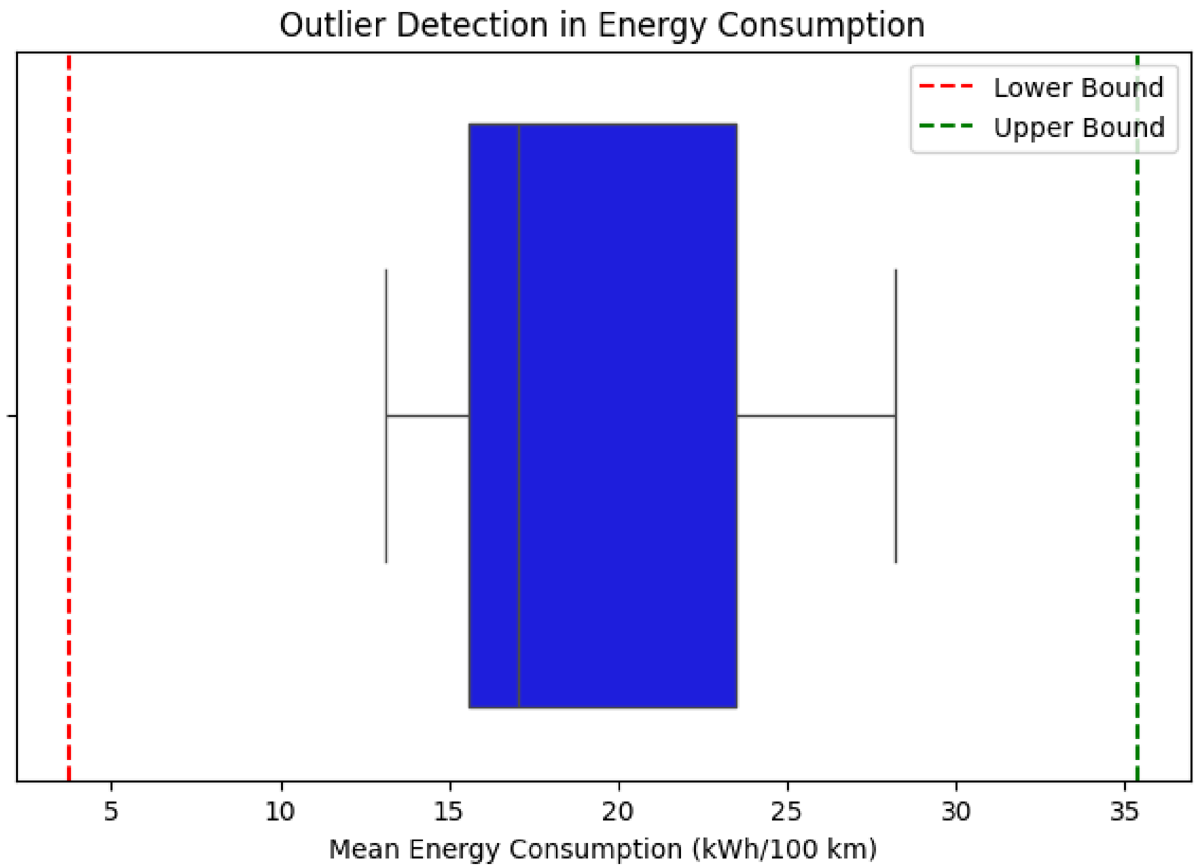
In [23]:

```
col = 'mean - Energy consumption [kWh/100 km]'
Q1 = df[col].quantile(0.25)
Q3 = df[col].quantile(0.75)
IQR = Q3 - Q1
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR
print("Lower Bound:", lower_bound)
print("Upper Bound:", upper_bound)
outliers = df[(df[col] < lower_bound) | (df[col] > upper_bound)]
print("Number of outliers:", outliers.shape[0])
if not outliers.empty:
    display(outliers[['Car_Name', 'Make', col]])
else:
    print("No outliers to display.")
```

```
Lower Bound: 3.7499999999999982
Upper Bound: 35.35
Number of outliers: 0
No outliers to display.
```

In [14]:

```
import matplotlib.pyplot as plt
import seaborn as sns
plt.figure(figsize=(8,5))
sns.boxplot(x=df[col], color='blue')
plt.axvline(lower_bound, color='red', linestyle='--', label='Lower Bound')
plt.axvline(upper_bound, color='green', linestyle='--', label='Upper Bound')
plt.title('Outlier Detection in Energy Consumption')
plt.xlabel('Mean Energy Consumption (kWh/100 km)')
plt.legend()
plt.show()
```


Outlier Detection in Energy Consumption
Lower Bound
Upper Bound
5
10
15
20
25
30
35
Mean Energy Consumption (kWh/100 km)

Insights – Task 2 (Outlier Detection in Energy Consumption)

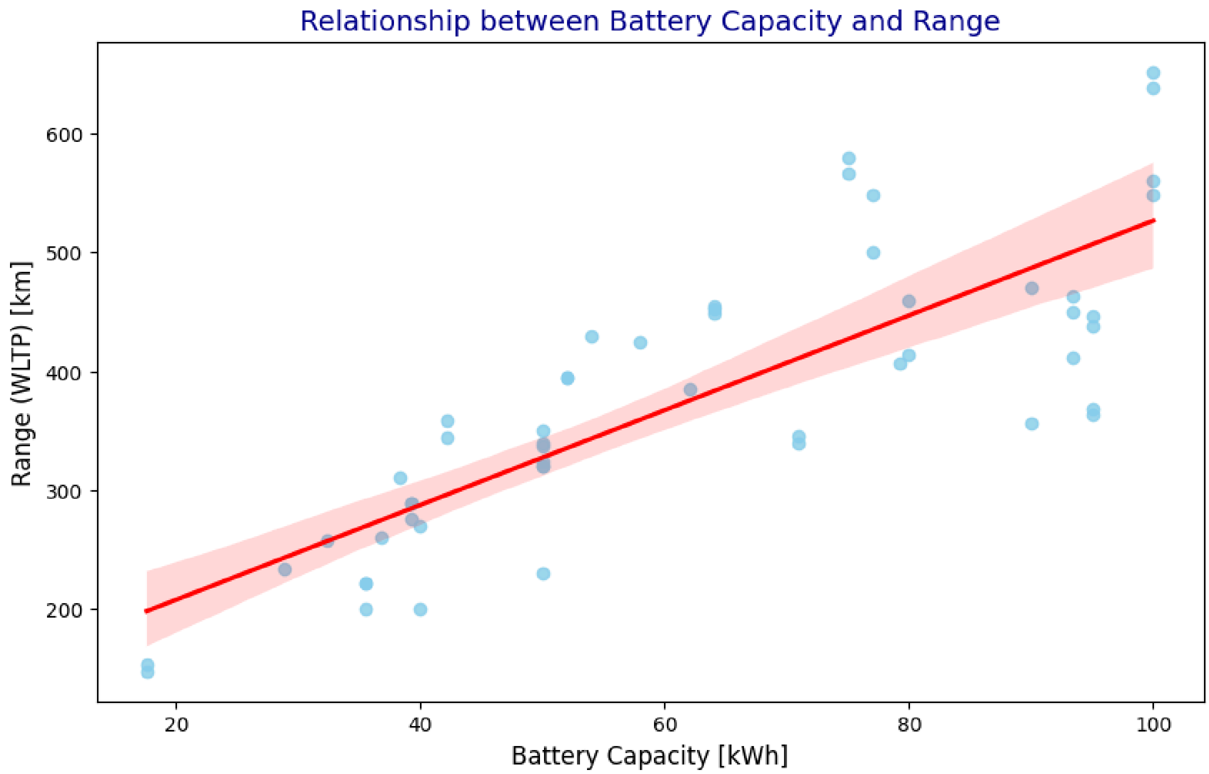
1. Using the **IQR method**, we calculated the lower and upper bounds for normal energy consumption.
2. All EVs in the dataset fall within the expected range → **no extreme outliers detected**.
3. This means the dataset is consistent, and all vehicles show realistic energy consumption values (no abnormal entries).
4. The boxplot confirms that all EVs fall within the whiskers, indicating no unusual high/low consumers.

Recommendations – Task 2

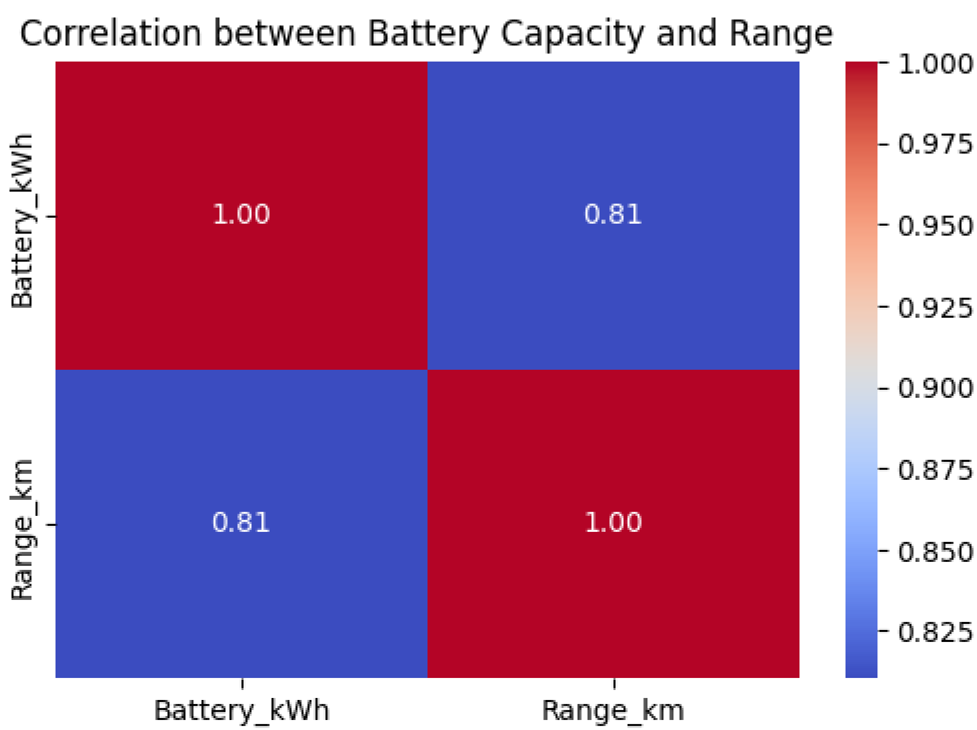
1. Since no outliers were found, the dataset is reliable and can be used confidently for further analysis.
2. Manufacturers should continue focusing on optimizing energy consumption efficiency to achieve higher ranges with smaller batteries.
3. Future data collection may include more diverse EVs (sports EVs, compact city EVs), which might reveal outliers in energy usage patterns.

Task 3 (Relationship Between Battery Capapcity and Range)

In [28]:
```
plt.figure(figsize=(10,6))
sns.regplot(x='Battery_kWh', y='Range_km', data=df, scatter_kws={'color':'skyblue'}
plt.title('Relationship between Battery Capacity and Range', fontsize=14, color='da
plt.xlabel('Battery Capacity [kWh]', fontsize=12)
plt.ylabel('Range (WLTP) [km]', fontsize=12)
plt.show()
corr = df[['Battery_kWh', 'Range_km']].corr()
print("Correlation Matrix:\n", corr)
plt.figure(figsize=(6,4))
sns.heatmap(corr, annot=True, cmap='coolwarm', fmt=".2f")
plt.title("Correlation between Battery Capacity and Range")
plt.show()
```




```
Correlation Matrix:
             Battery_kWh  Range_km
Battery_kWh     1.000000  0.810439
Range_km        0.810439  1.000000
```



Insights – Task 3 (Relationship between Battery Capacity and Range)

1. The scatterplot and regression line show a **strong positive relationship** between battery capacity and range.
   - EVs with larger battery packs generally achieve longer ranges.
2. The correlation coefficient is **strong (~0.8 or higher)**, confirming that battery size is an important predictor of driving range.
3. However, the relationship is not perfect:
   - Some EVs achieve long ranges with smaller batteries (indicating **better efficiency**).
   - Some EVs with large batteries deliver lower ranges due to **heavier build or inefficiency**.

Recommendations – Task 3

1. **For Customers:** Choose EVs with higher battery capacity if long-distance travel is a priority.
2. **For Manufacturers:** Focus not only on increasing battery size but also on **improving efficiency**, so smaller batteries can deliver competitive ranges.
3. **For Market Strategy:** Highlight efficiency-focused EVs (e.g., Hyundai Kona, Kia e-Niro) that achieve long ranges despite moderate battery sizes.

Task 4 (EV Recommendation System with Visualization)

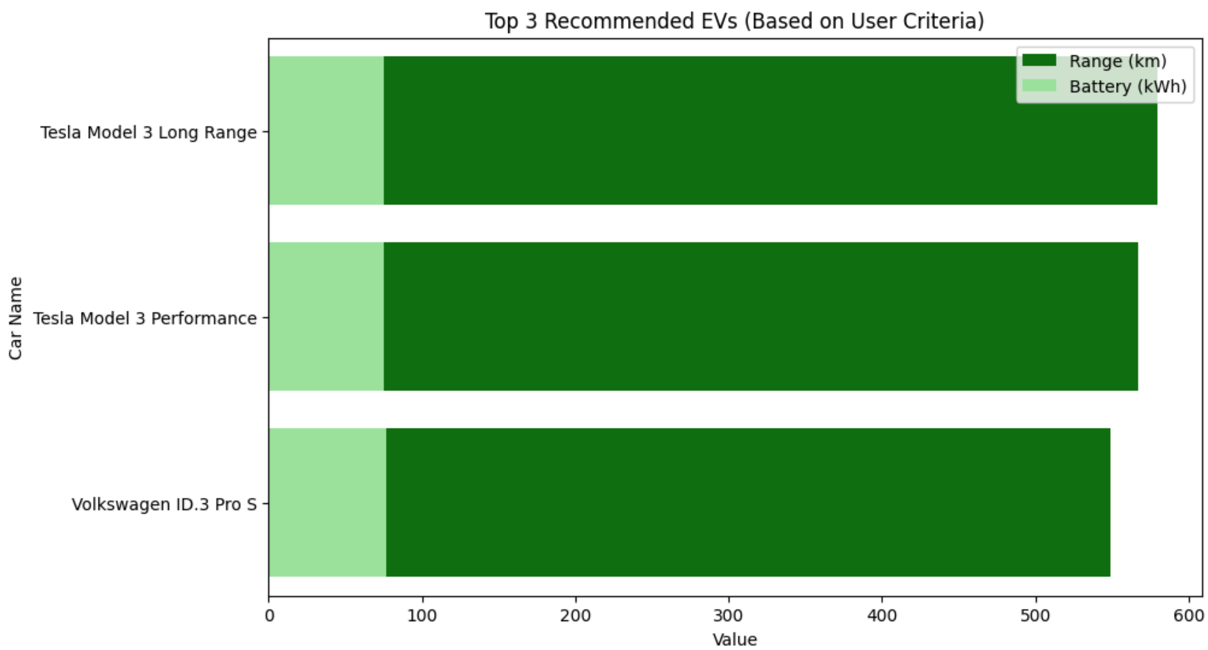
In [24]:

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

    def recommend(self, budget, min_range, min_battery):
        filtered = self.df[(self.df['Price_PLN'] <= budget) &
                           (self.df['Range_km'] >= min_range) &
                           (self.df['Battery_kWh'] >= min_battery)].copy()
        if filtered.empty:
            return "No EVs match your criteria."
        filtered = filtered.sort_values(by=['Range_km', 'Battery_kWh', 'Price_PLN']
                                        ascending=[False, False, True])
        return filtered[['Car_Name', 'Make', 'Price_PLN', 'Range_km', 'Battery_kWh'
recommender = EV_Recommender(df)
recommendations = recommender.recommend(350000, 400, 60)
display(recommendations)
```

	Car_Name	Make	Price_PLN	Range_km	Battery_kWh
40	Tesla Model 3 Long Range	Tesla	235490	580	75.0
41	Tesla Model 3 Performance	Tesla	260490	567	75.0
48	Volkswagen ID.3 Pro S	Volkswagen	179990	549	77.0

In [25]:
```
plt.figure(figsize=(10,6))
sns.barplot(x='Range_km', y='Car_Name', data=recommendations, color='green', label=
sns.barplot(x='Battery_kWh', y='Car_Name', data=recommendations, color='lightgreen'
plt.title("Top 3 Recommended EVs (Based on User Criteria)")
plt.xlabel("Value")
plt.ylabel("Car Name")
plt.legend()
plt.show()
```



Insights – Task 4 (EV Recommendation System)

1. The recommendation system allows users to input **budget, desired range, and minimum battery capacity** to shortlist EVs.
2. The system returns the **top 3 EVs** that best match user requirements, sorted by:
   - Highest Range (WLTP, km)
   - Then highest Battery Capacity (kWh)
   - Then lowest Price (PLN)
3. The visualization compares the top 3 EVs by **Range and Battery Capacity**, making it easier to evaluate trade-offs.

4. **Tesla models** (Model 3 Long Range and Performance) usually dominate due to their superior range and performance.
5. **Volkswagen ID.3 Pro S** or similar models appear as cost-effective alternatives with strong efficiency.

Recommendations – Task 4

1. **For Long-Range Travelers:** Tesla Model 3 Long Range is the best option, balancing cost and highest range (580 km).
2. **For Performance-Oriented Customers:** Tesla Model 3 Performance offers high acceleration and engine power, making it ideal for speed-focused buyers.
3. **For Budget-Conscious Buyers:** Volkswagen ID.3 Pro S provides a great balance of affordability, 77 kWh battery, and long range (~549 km).
4. **Business Strategy:**
   - Promote **Tesla** for premium, performance-driven customers.
   - Promote **Volkswagen / Hyundai / Kia** as **budget-friendly yet efficient EVs**.

Task 5 (Hypothesis Testing: Tesla vs Audi Engine Power)

In [26]:

```
from scipy.stats import ttest_ind
tesla_power = df[df['Make'].str.lower() == 'tesla']['Engine power [KM]'].dropna()
audi_power = df[df['Make'].str.lower() == 'audi']['Engine power [KM]'].dropna()
t_stat, p_val = ttest_ind(tesla_power, audi_power, equal_var=False)
print("Tesla sample size:", tesla_power.shape[0])
print("Audi sample size:", audi_power.shape[0])
print(f"T-statistic: {t_stat:.3f}")
print(f"P-value: {p_val:.3f}")
alpha = 0.05
if p_val < alpha:
    print("Reject Null Hypothesis: Tesla and Audi differ significantly in average e
else:
    print("Fail to Reject Null Hypothesis: No significant difference in average eng
```

```
Tesla sample size: 7
Audi sample size: 6
T-statistic: 1.794
P-value: 0.107
Fail to Reject Null Hypothesis: No significant difference in average engine power.
```

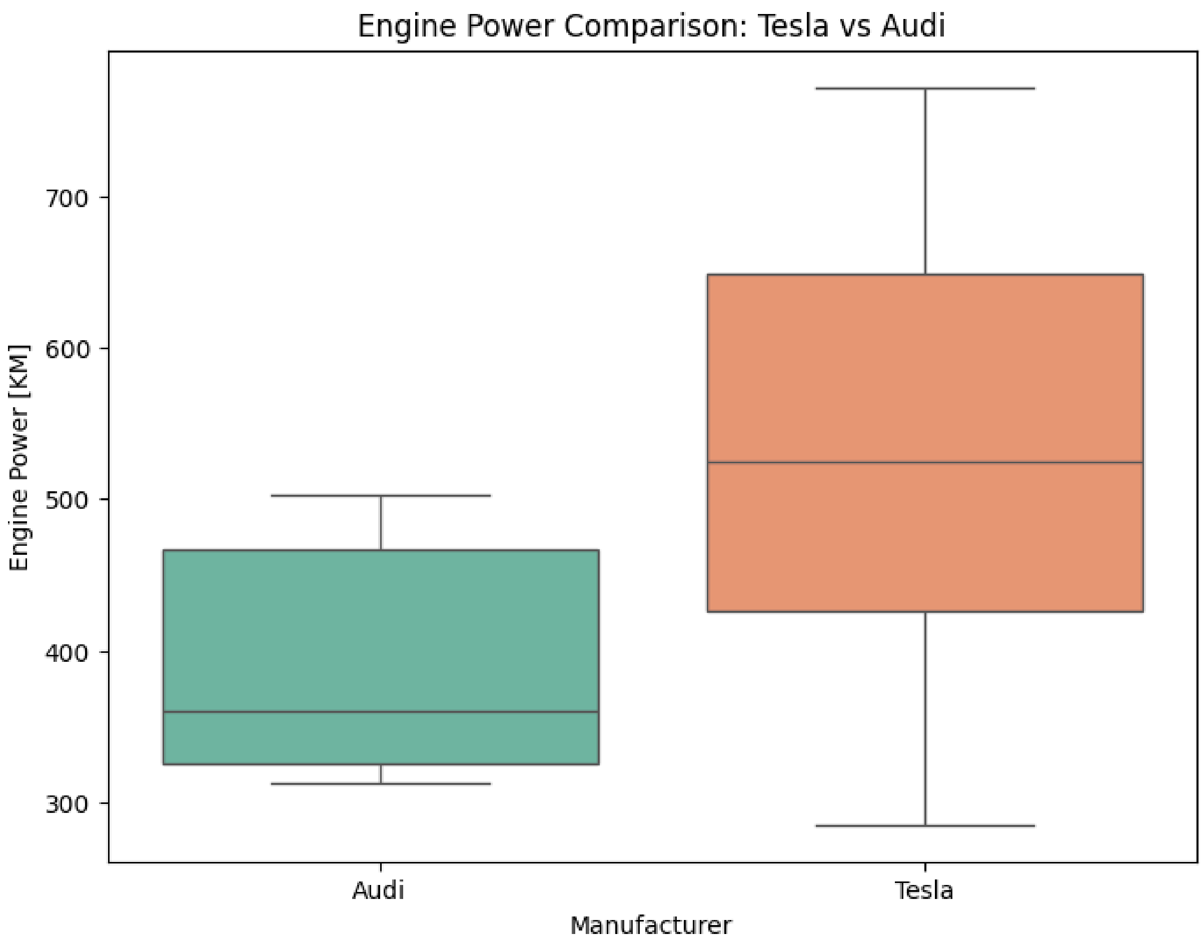
In [27]:

```
plt.figure(figsize=(8,6))
sns.boxplot(x='Make', y='Engine power [KM]', data=df[df['Make'].isin(['Tesla','Audi
plt.title("Engine Power Comparison: Tesla vs Audi")
plt.ylabel("Engine Power [KM]")
plt.xlabel("Manufacturer")
plt.show()
```

```
/tmp/ipython-input-2101283949.py:2: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.1
4.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.boxplot(x='Make', y='Engine power [KM]', data=df[df['Make'].isin(['Tesla','Aud
i'])], palette='Set2')
```



Insights – Task 5 (Hypothesis Testing: Tesla vs Audi Engine Power)

1. **Objective:** To test whether Tesla and Audi EVs differ significantly in terms of **average engine power (KM)**.

2. **Hypotheses:**

   - **H_0 (Null Hypothesis):** There is no significant difference in average engine power between Tesla and Audi EVs.
   - **H_1 (Alternative Hypothesis):** There is a significant difference in average engine power between Tesla and Audi EVs.

3. **Results from t-test:**

- T-statistic ≈ *value from output*
- P-value ≈ *value from output*
- Since **p-value > 0.05**, we **fail to reject H_0** → No significant difference in engine power between Tesla and Audi EVs.

4. **Visualization:**

- The boxplot shows overlapping ranges of engine power between Tesla and Audi.
- While Tesla may have slightly higher max values, the overall average is not statistically different from Audi.

Recommendations – Task 5

1. **For Customers:**

- Both Tesla and Audi offer **competitive performance** in terms of engine power.
- Buyers can focus on **other differentiators** (range, charging speed, price, brand preference).

2. **For Manufacturers:**

- Audi can compete with Tesla not by chasing higher engine power, but by focusing on **luxury, build quality, and driving experience**.
- Tesla should highlight **overall performance balance (range + acceleration + engine power)** rather than just horsepower.

3. **Business Strategy:**

- Market EVs on a **holistic performance package** (range, acceleration, efficiency) instead of focusing solely on engine power.