In [4]:
```
#installation of libraries.
!pip install pandas
!pip install matplotlib
!pip install seaborn
!pip install scipy
```

```
Requirement already satisfied: pandas in c:\users\lap\anaconda3\lib\site-packages
(2.2.2)
Requirement already satisfied: numpy>=1.26.0 in c:\users\lap\anaconda3\lib\site-p
ackages (from pandas) (1.26.4)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\lap\anaconda3\l
ib\site-packages (from pandas) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in c:\users\lap\anaconda3\lib\site-pa
ckages (from pandas) (2024.1)
Requirement already satisfied: tzdata>=2022.7 in c:\users\lap\anaconda3\lib\site-
packages (from pandas) (2023.3)
Requirement already satisfied: six>=1.5 in c:\users\lap\anaconda3\lib\site-packag
es (from python-dateutil>=2.8.2->pandas) (1.16.0)
Requirement already satisfied: matplotlib in c:\users\lap\anaconda3\lib\site-pack
ages (3.9.2)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\lap\anaconda3\lib\sit
e-packages (from matplotlib) (1.2.0)
Requirement already satisfied: cycler>=0.10 in c:\users\lap\anaconda3\lib\site-pa
ckages (from matplotlib) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\lap\anaconda3\lib\si
te-packages (from matplotlib) (4.51.0)
Requirement already satisfied: kiwisolver>=1.3.1 in c:\users\lap\anaconda3\lib\si
te-packages (from matplotlib) (1.4.4)
Requirement already satisfied: numpy>=1.23 in c:\users\lap\anaconda3\lib\site-pac
kages (from matplotlib) (1.26.4)
Requirement already satisfied: packaging>=20.0 in c:\users\lap\anaconda3\lib\site
-packages (from matplotlib) (24.1)
Requirement already satisfied: pillow>=8 in c:\users\lap\anaconda3\lib\site-packa
ges (from matplotlib) (10.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in c:\users\lap\anaconda3\lib\sit
e-packages (from matplotlib) (3.1.2)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\lap\anaconda3\lib
\site-packages (from matplotlib) (2.9.0.post0)
Requirement already satisfied: six>=1.5 in c:\users\lap\anaconda3\lib\site-packag
es (from python-dateutil>=2.7->matplotlib) (1.16.0)
Requirement already satisfied: seaborn in c:\users\lap\anaconda3\lib\site-package
s (0.13.2)
Requirement already satisfied: numpy!=1.24.0,>=1.20 in c:\users\lap\anaconda3\lib
\site-packages (from seaborn) (1.26.4)
Requirement already satisfied: pandas>=1.2 in c:\users\lap\anaconda3\lib\site-pac
kages (from seaborn) (2.2.2)
Requirement already satisfied: matplotlib!=3.6.1,>=3.4 in c:\users\lap\anaconda3
\lib\site-packages (from seaborn) (3.9.2)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\lap\anaconda3\lib\sit
e-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (1.2.0)
Requirement already satisfied: cycler>=0.10 in c:\users\lap\anaconda3\lib\site-pa
ckages (from matplotlib!=3.6.1,>=3.4->seaborn) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\lap\anaconda3\lib\si
te-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (4.51.0)
Requirement already satisfied: kiwisolver>=1.3.1 in c:\users\lap\anaconda3\lib\si
te-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (1.4.4)
Requirement already satisfied: packaging>=20.0 in c:\users\lap\anaconda3\lib\site
-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (24.1)
Requirement already satisfied: pillow>=8 in c:\users\lap\anaconda3\lib\site-packa
ges (from matplotlib!=3.6.1,>=3.4->seaborn) (10.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in c:\users\lap\anaconda3\lib\sit
e-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (3.1.2)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\lap\anaconda3\lib
\site-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in c:\users\lap\anaconda3\lib\site-pa
ckages (from pandas>=1.2->seaborn) (2024.1)
```

```
Requirement already satisfied: tzdata>=2022.7 in c:\users\lap\anaconda3\lib\site-
packages (from pandas>=1.2->seaborn) (2023.3)
Requirement already satisfied: six>=1.5 in c:\users\lap\anaconda3\lib\site-packag
es (from python-dateutil>=2.7->matplotlib!=3.6.1,>=3.4->seaborn) (1.16.0)
Requirement already satisfied: scipy in c:\users\lap\anaconda3\lib\site-packages
(1.13.1)
Requirement already satisfied: numpy<2.3,>=1.22.4 in c:\users\lap\anaconda3\lib\s
ite-packages (from scipy) (1.26.4)
```

In [36]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import ttest_ind
from scipy.stats import pearsonr
from typing import List, Tuple, Dict
```

In [12]:
```
#Datasets.
dataset = pd.read_csv("C:/Users/lap/Downloads/FEV-data-Excel.xlsx - Auta elektry
dataset.head()
```

Out[12]:

	Car full name	Make	Model	Minimal price (gross) [PLN]	Engine power [KM]	Maximum torque [Nm]	Type of brakes	Drive type	Battery capacity [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 [24]:
```
dataset.isnull().sum()
dataset.dropna(inplace=True)
```

In [26]:
```
dataset['Minimal price (gross) [PLN]'] = dataset['Minimal price (gross) [PLN]'].
```

In [46]:
```
dataset.rename(columns={'Minimal price (gross) [PLN]': 'Price', 'Range (WLTP) [k
```

In [48]:
```
#TASK 1.
filtered_dataset = dataset[(dataset['Price'] <= 350000) & (dataset['Range'] >= 4
print(filtered_dataset[['Make', 'Model', 'Price', 'Range']])
```

```
            Make                  Model     Price  Range
0           Audi     e-tron 55 quattro  345700.0    438
8            BMW                   iX3  282900.0    460
15       Hyundai   Kona electric 64kWh  178400.0    449
18           Kia          e-Niro 64kWh  167990.0    455
20           Kia          e-Soul 64kWh  160990.0    452
22  Mercedes-Benz                   EQC  334700.0    414
47     Volkswagen  ID.3 Pro Performance  155890.0    425
48     Volkswagen            ID.3 Pro S  179990.0    549
49     Volkswagen              ID.4 1st  202390.0    500
```

In [50]:
```
grouped = filtered_dataset.groupby('Make').size()
print(grouped)
```

```
Make
Audi             1
BMW              1
Hyundai          1
Kia              2
Mercedes-Benz    1
Volkswagen       3
dtype: int64
```

In [52]:
```
battery_avg = filtered_dataset.groupby('Make')['Battery capacity [kWh]'].mean()
print(battery_avg)
```

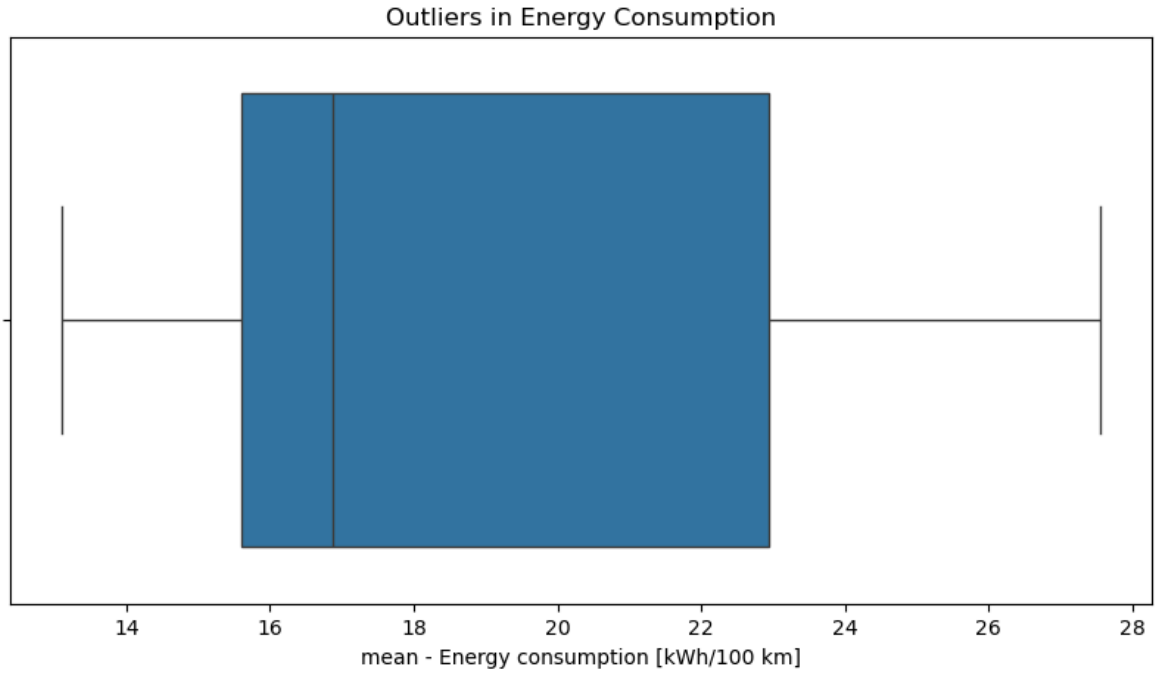
```
Make
Audi             95.000000
BMW              80.000000
Hyundai          64.000000
Kia              64.000000
Mercedes-Benz    80.000000
Volkswagen       70.666667
Name: Battery capacity [kWh], dtype: float64
```

In [54]:
```
#TASK 2.
q1 = dataset['mean - Energy consumption [kWh/100 km]'].quantile(0.25)
q3 = dataset['mean - Energy consumption [kWh/100 km]'].quantile(0.75)
iqr = q3 - q1

lower_bound = q1 - 1.5 * iqr
upper_bound = q3 + 1.5 * iqr

outliers = dataset[(dataset['mean - Energy consumption [kWh/100 km]'] < lower_bo
              (dataset['mean - Energy consumption [kWh/100 km]'] > upper_bound)]
print(outliers[['Make', 'Model', 'mean - Energy consumption [kWh/100 km]']])
```

```
Empty DataFrame
Columns: [Make, Model, mean - Energy consumption [kWh/100 km]]
Index: []
```

In [56]:
```
plt.figure(figsize=(10,5))
sns.boxplot(x=dataset['mean - Energy consumption [kWh/100 km]'])
```

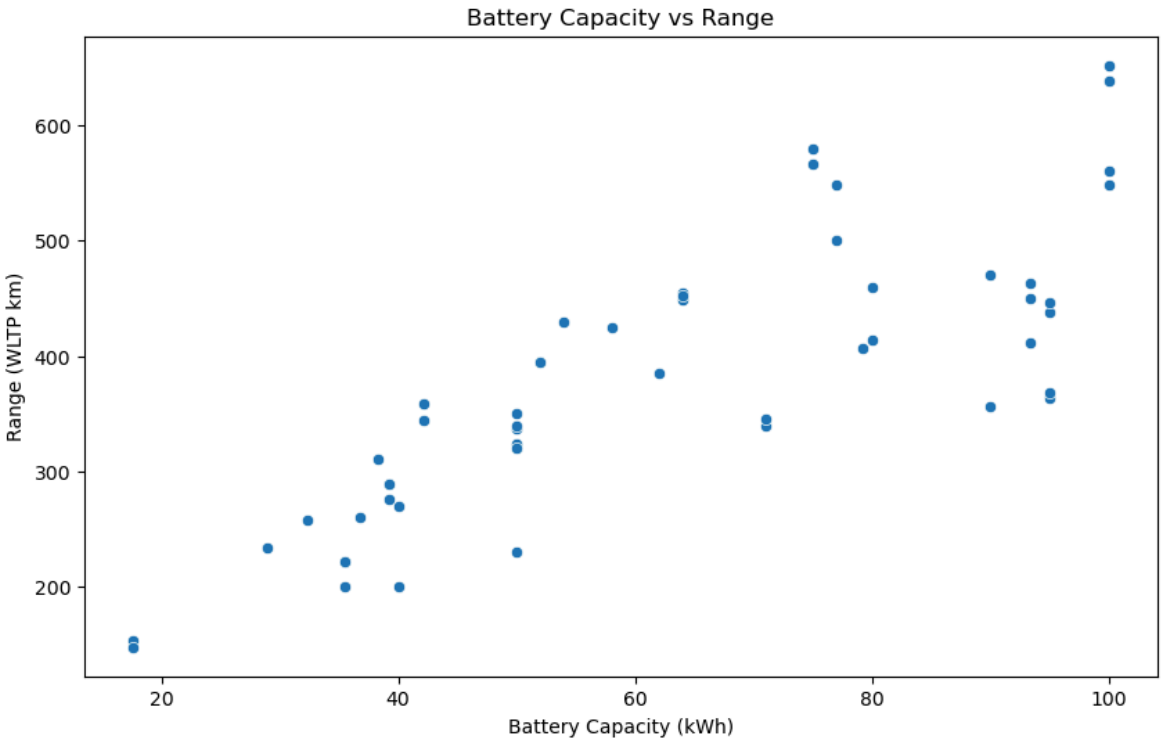
```
plt.title("Outliers in Energy Consumption")
plt.show()
```



In [172…

```
#Task 3.
def analyze_battery_range_relationship(df: pd.DataFrame):

    # Create visualization
    plt.figure(figsize=(10, 6))
    sns.scatterplot(data=df, x='Battery capacity [kWh]', y='Range (WLTP) [km]')
    plt.title('Battery Capacity vs Range')
    plt.xlabel('Battery Capacity (kWh)')
    plt.ylabel('Range (WLTP km)')

    # Calculate correlation coefficient
    correlation = df['Battery capacity [kWh]'].corr(df['Range (WLTP) [km]'])
    return correlation

# Example usage

if __name__ == "__main__":

    # Read the data
    df = pd.read_csv('C:/Users/lap/Downloads/FEV-data-Excel.xlsx - Auta elektryc

    # Get correlation
    correlation = analyze_battery_range_relationship(df)

    # Print result
    print(f"\nCorrelation between Battery Capacity and Range: {correlation:.2f}"

    # Show plot
    plt.show()
```

```
Correlation between Battery Capacity and Range: 0.81
```



In [174…

```python
#Task 4.

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

    def get_recommendations(self, budget: float, desired_range: int, min_battery
        # Create mask for filtering
        mask = (
            (self.df['Minimal price (gross) [PLN]'] <= budget) &
            (self.df['Range (WLTP) [km]'] >= desired_range) &
            (self.df['Battery capacity [kWh]'] >= min_battery_capacity)
        )

        # Create a copy of the filtered DataFrame
        filtered_df = self.df[mask].copy()

        if filtered_df.empty:
            print("No vehicles found matching your criteria.")
            return pd.DataFrame()

        # Calculate scores
        filtered_df.loc[:, 'score'] = (
            (filtered_df['Range (WLTP) [km]'] / desired_range) * 0.4 +
            (budget / filtered_df['Minimal price (gross) [PLN]']) * 0.4 +
            (filtered_df['Battery capacity [kWh]'] / min_battery_capacity) * 0.2
        )

        # Get top 3 recommendations
        columns_to_show = [
            'Car full name', 'Make', 'Model',
            'Minimal price (gross) [PLN]', 'Range (WLTP) [km]',
            'Battery capacity [kWh]', 'score'
        ]

        recommendations = filtered_df.nlargest(3, 'score')[columns_to_show]
        return recommendations
```

```python
def format_price(price):
    """Format price with thousands separator"""
    return f"{price:,.0f}"

# Example usage
if __name__ == "__main__":
    try:
        # Read the data
        df = pd.read_csv('C:/Users/lap/Downloads/FEV-data-Excel.xlsx - Auta elek

        # Create recommender instance
        recommender = EVRecommender(df)

        # Get recommendations
        recommendations = recommender.get_recommendations(
            budget=350000,
            desired_range=400,
            min_battery_capacity=60
        )

        if not recommendations.empty:
            print("\nTop 3 Recommended Electric Vehicles:")
            print("-" * 50)
            for idx, row in recommendations.iterrows():
                print(f"\nRecommendation {recommendations.index.get_loc(idx) + 1
                print(f"Model: {row['Car full name']}")
                print(f"Make: {row['Make']}")
                print(f"Price: {format_price(row['Minimal price (gross) [PLN]'])
                print(f"Range: {row['Range (WLTP) [km]']} km")
                print(f"Battery: {row['Battery capacity [kWh]']} kWh")
                print(f"Match Score: {row['score']:.2f}")

    except FileNotFoundError:
        print("Error: CSV file not found. Make sure 'FEV-data-Excel.xlsx - Auta
    except Exception as e:
        print(f"An error occurred: {str(e)}")
```

```
Top 3 Recommended Electric Vehicles:
--------------------------------------------------

Recommendation 1:
Model: Volkswagen ID.3 Pro S
Make: Volkswagen
Price: 179,990 PLN
Range: 549 km
Battery: 77.0 kWh
Match Score: 1.58

Recommendation 2:
Model: Kia e-Soul 64kWh
Make: Kia
Price: 160,990 PLN
Range: 452 km
Battery: 64.0 kWh
Match Score: 1.53

Recommendation 3:
Model: Kia e-Niro 64kWh
Make: Kia
Price: 167,990 PLN
Range: 455 km
Battery: 64.0 kWh
Match Score: 1.50
```

In [176…

```python
#Tsk 5.
from scipy.stats import ttest_ind, shapiro, levene

# Extract Tesla and Audi engine power data
tesla_power = df[df["Make"] == "Tesla"]["Engine power [KM]"].dropna()
audi_power = df[df["Make"] == "Audi"]["Engine power [KM]"].dropna()

# Check normality assumption using Shapiro-Wilk test
shapiro_tesla = shapiro(tesla_power)
shapiro_audi = shapiro(audi_power)

# Check homogeneity of variances using Levene's test
levene_test = levene(tesla_power, audi_power)

# Perform independent t-test
t_stat, p_value = ttest_ind(tesla_power, audi_power, equal_var=True)
# Interpret the results
alpha = 0.05
# Significance level

if p_value < alpha:
    print("There is a significant difference in the average engine power between
else:
    print("There is no significant difference in the average engine power betwee

# Display results
{
    "Tesla Mean Power": round(tesla_power.mean(), 2),
    "Audi Mean Power": round(audi_power.mean(), 2),
    "Shapiro-Wilk Tesla p-value": round(shapiro_tesla.pvalue, 4),
    "Shapiro-Wilk Audi p-value": round(shapiro_audi.pvalue, 4),
    "Levene Test p-value": round(levene_test.pvalue, 4),
    "T-test Statistic": round(t_stat, 4),
```

```
    "P-value": round(p_value, 4)
}
```

There is no significant difference in the average engine power between Tesla and Audi vehicles.

Out[176…

```
{'Tesla Mean Power': 533.0,
 'Audi Mean Power': 392.0,
 'Shapiro-Wilk Tesla p-value': 0.3819,
 'Shapiro-Wilk Audi p-value': 0.0441,
 'Levene Test p-value': 0.2196,
 'T-test Statistic': 1.7024,
 'P-value': 0.1167}
```

In [ ]:

In [ ]: