In [123…

```
# video - google drive link
# https://drive.google.com/file/d/1Bk-fU02NyAwYkYU4Zogcq_IEdseUug7Z/view?usp=sha


import pandas as pd

# Load dataset
df = pd.read_excel("FEV-data-Excel-cleaned.xlsx")

# info
# print("Dataset shape:", df.shape)
# print("\nColumn names:", df.columns.tolist())
# print("\nFirst 5 rows:")
print(df.head())

# Check missing values
# print("\nMissing values:")
# print(df.isnull().sum())
```

```
                      Car full name  Make                        Model  \
0            Audi e-tron 55 quattro  Audi            e-tron 55 quattro   
1            Audi e-tron 50 quattro  Audi            e-tron 50 quattro   
2             Audi e-tron S quattro  Audi             e-tron S quattro   
3  Audi e-tron Sportback 50 quattro  Audi  e-tron Sportback 50 quattro   
4  Audi e-tron Sportback 55 quattro  Audi  e-tron Sportback 55 quattro   

   Minimal price (gross) [PLN]  Engine power [KM]  Maximum torque [Nm]  \
0                       345700                360                  664   
1                       308400                313                  540   
2                       414900                503                  973   
3                       319700                313                  540   
4                       357000                360                  664   

        Type of brakes Drive type  Battery capacity [kWh]  Range (WLTP) [km]  \
0  disc (front + rear)        4WD                    95.0                438   
1  disc (front + rear)        4WD                    71.0                340   
2  disc (front + rear)        4WD                    95.0                364   
3  disc (front + rear)        4WD                    71.0                346   
4  disc (front + rear)        4WD                    95.0                447   

   ...  Permissable gross weight [kg]  Maximum load capacity [kg]  \
0  ...                           3130                         640   
1  ...                           3040                         670   
2  ...                           3130                         565   
3  ...                           3040                         640   
4  ...                           3130                         670   

   Number of seats  Number of doors  Tire size [in]  Maximum speed [kph]  \
0                5                5              19                  200   
1                5                5              19                  190   
2                5                5              20                  210   
3                5                5              19                  190   
4                5                5              19                  200   

   Boot capacity (VDA) [l]  Acceleration 0-100 kph [s]  \
0                      660                         5.7   
1                      660                         6.8   
2                      660                         4.5   
3                      615                         6.8   
4                      615                         5.7   

   Maximum DC charging power [kW]  mean - Energy consumption [kWh/100 km]  
0                             150                                   24.45  
1                             150                                   23.80  
2                             150                                   27.55  
3                             150                                   23.30  
4                             150                                   23.85  

[5 rows x 25 columns]
```

In [125…

```
# Task 1: A customer has a budget of 350,000 PLN and wants an EV with a minimum
# of 400 km.

# a) Your task is to filter out EVs that meet these criteria.(2 Marks)
filtered_df = df[(df['Minimal price (gross) [PLN]'] <= 350000) & (df['Range (WLT

# b) Group them by the manufacturer (Make).(6 marks)
# Group by manufacturer (Make)
grouped = filtered_df.groupby('Make')
# c) Calculate the average battery capacity for each manufacturer. (8 Marks)
# Calculate average battery capacity per manufacturer
average_battery_capacity = grouped['Battery capacity [kWh]'].mean()

# Display the result
average_battery_capacity
```

Out[125…

```
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 [127…

```
# Task 2: You suspect some EVs have unusually high or low energy consumption. Fi
# outliers in the mean - Energy consumption [kWh/100 km] column.(16 Marks)


import pandas as pd
import numpy as np
from scipy.stats import norm


# Mean and std of the energy consumption
mu = df['mean - Energy consumption [kWh/100 km]'].mean()
sigma = df['mean - Energy consumption [kWh/100 km]'].std()

# Z-test for high outliers
df['z_stat'] = (df['mean - Energy consumption [kWh/100 km]'] - mu) / sigma

# cdf- Cumulative distribution function
df['p_value'] = 1 - norm.cdf(df['z_stat'])  # Right tail

df['high_outlier'] = df['p_value'] < 0.05

# Show only high outliers
high_outliers = df[df['high_outlier']]
high_outliers
```

Out[127…

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]

0 rows × 28 columns

In [129…

```
# Z-test for low outliers
df['z_stat'] = (df['mean - Energy consumption [kWh/100 km]'] - mu) / sigma
df['p_value'] = norm.cdf(df['z_stat'])  # Left tail

# Flag low outliers at alpha = 0.05
df['low_outlier'] = df['p_value'] < 0.05
```

```
# Show only low outliers
low_outliers = df[df['low_outlier']]
low_outliers
```

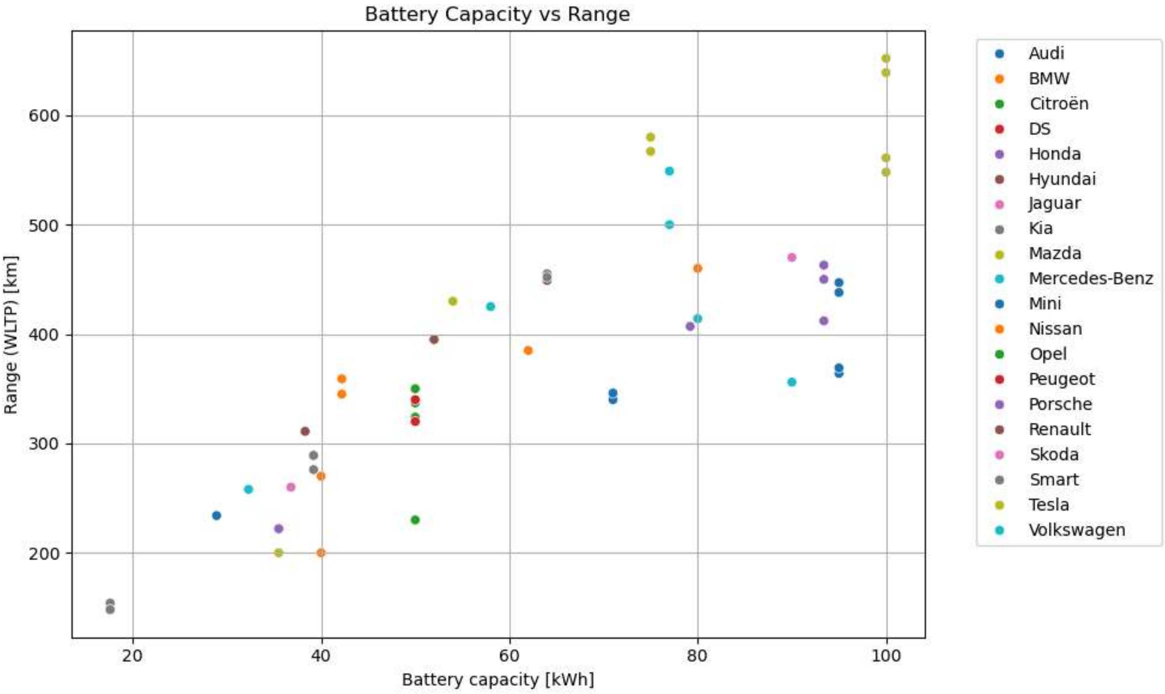
Out[129…

	Car full name	Make	Model	Minimal price (gross) [PLN]	Engine power [KM]	Maximum torque [Nm]	Type of brakes	Drive type	Ba cap [
9	Citroën ë-C4	Citroën	ë-C4	125000	136	260	disc (front + rear)	2WD (front)	
29	Peugeot e-2008	Peugeot	e-2008	149400	136	260	disc (front + rear)	2WD (front)	
39	Tesla Model 3 Standard Range Plus	Tesla	Model 3 Standard Range Plus	195490	285	450	disc (front + rear)	2WD (rear)	
40	Tesla Model 3 Long Range	Tesla	Model 3 Long Range	235490	372	510	disc (front + rear)	4WD	
41	Tesla Model 3 Performance	Tesla	Model 3 Performance	260490	480	639	disc (front + rear)	4WD	
42	Tesla Model S Long Range Plus	Tesla	Model S Long Range Plus	368990	525	755	disc (front + rear)	4WD	
43	Tesla Model S Performance	Tesla	Model S Performance	443990	772	1140	disc (front + rear)	4WD	
44	Tesla Model X Long Range Plus	Tesla	Model X Long Range Plus	407990	525	755	disc (front + rear)	4WD	
45	Tesla Model X Performance	Tesla	Model X Performance	482990	772	1140	disc (front + rear)	4WD	

9 rows × 29 columns

In [131…

```
# Task 3: Your manager wants to know if there's a strong relationship between ba
# capacity and range.
# a) Create a suitable plot to visualize.(8 Marks)
# b) Highlight any insights.(8 Marks)


import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(10, 6))

sns.scatterplot(data=df, x='Battery capacity [kWh]', y='Range (WLTP) [km]', hue=
plt.title('Battery Capacity vs Range')

plt.xlabel('Battery capacity [kWh]')
plt.ylabel('Range (WLTP) [km]')

plt.legend(bbox_to_anchor=(1.05, 1), loc='upper left')

plt.grid(True)
plt.tight_layout()

plt.show()
# higher battery capacity result in longer ranges.
```



In [132…

```
# Task 4: Build an EV recommendation class. The class should allow users to inpu
# budget, desired range, and battery capacity. The class should then return the
# matching their criteria. (8+8 Marks)

class recommendEV:
    def __init__(self, data):
        self.data = data

    def recommend(self, budget, min_range, min_battery_capacity):
        filtered = self.data[
            (self.data['Minimal price (gross) [PLN]'] <= budget) &
            (self.data['Range (WLTP) [km]'] >= min_range) &
            (self.data['Battery capacity [kWh]'] >= min_battery_capacity)
        ]
        top_three = filtered.sort_values(by='Minimal price (gross) [PLN]').head(
        return top_three[['Make', 'Model', 'Minimal price (gross) [PLN]', 'Range

# Example:
recommender = recommendEV(df)
recommender.recommend(budget=145490, min_range=200, min_battery_capacity=20)
```

Out[132…

	Make	Model	Minimal price (gross) [PLN]	Range (WLTP) [km]	Battery capacity [kWh]
36	Skoda	Citigo-e iV	82050	260	36.8
46	Volkswagen	e-up!	97990	258	32.3
24	Nissan	Leaf	122900	270	40.0

In [133…

```
# Task 5: Inferential Statistics – Hypothesis Testing: Test whether there is a s
# difference in the average Engine power [KM] of vehicles manufactured by two le
# manufacturers i.e. Tesla and Audi. What insights can you draw from the test re
# Recommendations and Conclusion: Provide actionable insights based on your anal
# (Conduct a two sample t-test using ttest_ind from scipy.stats module)


from scipy.stats import ttest_ind

# Filter data for Tesla and Audi
tesla_power = df[df['Make'] == 'Tesla']['Engine power [KM]']
audi_power = df[df['Make'] == 'Audi']['Engine power [KM]']

# Perform two-sample t-test (Welch’s t-test assumes unequal variances)
t_stat, p_value = ttest_ind(tesla_power, audi_power, equal_var=False)

print("T-statistic:", round(t_stat, 2))
print("P-value:", round(p_value, 4))

# Interpretation
if p_value < 0.05:
    print("There is a statistically significant difference in average engine pow
else:
    print("No statistically significant difference in average engine power betwe
```

```
T-statistic: 1.79
P-value: 0.1068
No statistically significant difference in average engine power between Tesla and
Audi.
```

In [ ]: