In [1]:
```python
import os
os.environ["PYSPARK_PYTHON"] = "/home/shiva/venv/bin/python"
os.environ["PYSPARK_DRIVER_PYTHON"] = "/home/shiva/venv/bin/python"
```

Practice Questions Solutions

Sources

- Questions source - ChatGPT, (Questions.md)
- Official answer source - ChatGPT, (Answers.md)
- CSV source - Kaggle - https://www.kaggle.com/datasets/urvishahir/electric-vehicle-specifications-dataset-2025

Setup

In [51]:
```python
import pyspark.sql as ps
import pyspark.sql.functions as psf
import pyspark.sql.types as pst
import pandas as pd
import pyspark.sql.window as psw
```

In [3]:
```python
spark = (ps.SparkSession
         .builder
         .appName("EV Spec Practice")
         .getOrCreate()
        )
```

```
WARNING: Using incubator modules: jdk.incubator.vector
Using Spark's default log4j profile: org/apache/spark/log4j2-defaults.prop
erties
25/07/03 22:55:17 WARN Utils: Your hostname, Victus, resolves to a loopbac
k address: 127.0.1.1; using 192.168.29.87 instead (on interface wlo1)
25/07/03 22:55:17 WARN Utils: Set SPARK_LOCAL_IP if you need to bind to an
other address
Using Spark's default log4j profile: org/apache/spark/log4j2-defaults.prop
erties
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setL
ogLevel(newLevel).
25/07/03 22:55:18 WARN NativeCodeLoader: Unable to load native-hadoop libr
ary for your platform... using builtin-java classes where applicable
```

In [4]:
```python
file_path = "electric_vehicles_spec_2025.csv.csv"
```

In [5]:
```python
df = (
    spark
    .read
    .option("samplingRatio", 0.001)
    .option("header", "true")
    .option("inferschema", "true")
    .csv(file_path)
)
```

In [6]:
```
df.columns
```

Out[6]:
```
['brand',
 'model',
 'top_speed_kmh',
 'battery_capacity_kWh',
 'battery_type',
 'number_of_cells',
 'torque_nm',
 'efficiency_wh_per_km',
 'range_km',
 'acceleration_0_100_s',
 'fast_charging_power_kw_dc',
 'fast_charge_port',
 'towing_capacity_kg',
 'cargo_volume_l',
 'seats',
 'drivetrain',
 'segment',
 'length_mm',
 'width_mm',
 'height_mm',
 'car_body_type',
 'source_url']
```

In [7]:
```
(
    df
    .createOrReplaceTempView("df_view")
)
```

Easy Questions

Focus: RDD/DataFrame basics, filtering, selecting, grouping, sorting, basic aggregations, handling nulls.

DataFrame Basics

1. Load the CSV file into a DataFrame and print the schema.

```
df = (
    spark
    .read
    .option("samplingRatio", 0.001)
    .option("header", "true")
    .option("inferschema", "true")
    .csv(file_path)
)```
```

In [52]:
```
df.printSchema()
```

```
root
 |-- brand: string (nullable = true)
 |-- model: string (nullable = true)
 |-- top_speed_kmh: integer (nullable = true)
 |-- battery_capacity_kWh: double (nullable = true)
 |-- battery_type: string (nullable = true)
 |-- number_of_cells: integer (nullable = true)
 |-- torque_nm: integer (nullable = true)
 |-- efficiency_wh_per_km: integer (nullable = true)
 |-- range_km: integer (nullable = true)
 |-- acceleration_0_100_s: double (nullable = true)
 |-- fast_charging_power_kw_dc: integer (nullable = true)
 |-- fast_charge_port: string (nullable = true)
 |-- towing_capacity_kg: integer (nullable = true)
 |-- cargo_volume_l: integer (nullable = true)
 |-- seats: integer (nullable = true)
 |-- drivetrain: string (nullable = true)
 |-- segment: string (nullable = true)
 |-- length_mm: integer (nullable = true)
 |-- width_mm: integer (nullable = true)
 |-- height_mm: integer (nullable = true)
 |-- car_body_type: string (nullable = true)
 |-- source_url: string (nullable = true)
```

In [53]:
```
df.schema
```

Out[53]:
```
StructType([StructField('brand', StringType(), True), StructField('mode
l', StringType(), True), StructField('top_speed_kmh', IntegerType(), Tru
e), StructField('battery_capacity_kWh', DoubleType(), True), StructField
('battery_type', StringType(), True), StructField('number_of_cells', Int
egerType(), True), StructField('torque_nm', IntegerType(), True), Struct
Field('efficiency_wh_per_km', IntegerType(), True), StructField('range_k
m', IntegerType(), True), StructField('acceleration_0_100_s', DoubleType
(), True), StructField('fast_charging_power_kw_dc', IntegerType(), Tru
e), StructField('fast_charge_port', StringType(), True), StructField('to
wing_capacity_kg', IntegerType(), True), StructField('cargo_volume_l', I
ntegerType(), True), StructField('seats', IntegerType(), True), StructFi
eld('drivetrain', StringType(), True), StructField('segment', StringType
(), True), StructField('length_mm', IntegerType(), True), StructField('w
idth_mm', IntegerType(), True), StructField('height_mm', IntegerType(),
True), StructField('car_body_type', StringType(), True), StructField('so
urce_url', StringType(), True)])
```

---

2. Display the first 10 rows of the DataFrame.

In [54]:
```
df.show(10)
```

```
+------+--------------------+-------------+--------------------+----------
--+---------------+---------+--------------------+--------+---------------
-----+-------------------------+----------------+------------------+------
--------+-----+----------+------------+---------+--------+---------+------
-------+--------------------+
| brand|               model|top_speed_kmh|battery_capacity_kWh|battery_ty
pe|number_of_cells|torque_nm|efficiency_wh_per_km|range_km|acceleration_0_
100_s|fast_charging_power_kw_dc|fast_charge_port|towing_capacity_kg|cargo_
volume_l|seats|drivetrain|     segment|length_mm|width_mm|height_mm|car_bo
dy_type|          source_url|
+------+--------------------+-------------+--------------------+----------
--+---------------+---------+--------------------+--------+---------------
-----+-------------------------+----------------+------------------+------
--------+-----+----------+------------+---------+--------+---------+------
-------+--------------------+
|Abarth|    500e Convertible|          155|                37.8| Lithium-i
on|            192|      235|                 156|     225|
7.0|                       67|             CCS|                 0|
185|    4|       FWD| B - Compact|     3673|    1683|     1518|    Hatchba
ck|https://ev-databa...|
|Abarth|      500e Hatchback|          155|                37.8| Lithium-i
on|            192|      235|                 149|     225|
7.0|                       67|             CCS|                 0|
185|    4|       FWD| B - Compact|     3673|    1683|     1518|    Hatchba
ck|https://ev-databa...|
|Abarth| 600e Scorpionissima|          200|                50.8| Lithium-i
on|            102|      345|                 158|     280|
5.9|                       79|             CCS|                 0|
360|    5|       FWD|JB - Compact|     4187|    1779|     1557|          S
UV|https://ev-databa...|
|Abarth|        600e Turismo|          200|                50.8| Lithium-i
on|            102|      345|                 158|     280|
6.2|                       79|             CCS|                 0|
360|    5|       FWD|JB - Compact|     4187|    1779|     1557|          S
UV|https://ev-databa...|
|Aiways|                  U5|          150|                60.0| Lithium-i
on|           NULL|      310|                 156|     315|
7.5|                       78|             CCS|              NULL|
496|    5|       FWD| JC - Medium|     4680|    1865|     1700|          S
UV|https://ev-databa...|
|Aiways|                  U6|          160|                60.0| Lithium-i
on|           NULL|      315|                 150|     350|
7.0|                       78|             CCS|              NULL|
472|    5|       FWD| JC - Medium|     4805|    1880|     1641|          S
UV|https://ev-databa...|
|  Alfa|Romeo Junior Elet...|          150|                50.8| Lithium-i
on|            102|      260|                 128|     320|
9.0|                       85|             CCS|                 0|
400|    5|       FWD|JB - Compact|     4173|    1781|     1532|          S
UV|https://ev-databa...|
|  Alfa|Romeo Junior Elet...|          200|                50.8| Lithium-i
on|            102|      345|                 164|     310|
6.0|                       85|             CCS|                 0|
400|    5|       FWD|JB - Compact|     4173|    1781|     1505|          S
UV|https://ev-databa...|
|Alpine|A290 Electric 180 hp|          160|                52.0| Lithium-i
on|            184|      285|                 138|     310|
7.4|                       70|             CCS|               500|
326|    5|       FWD| B - Compact|     3997|    1823|     1512|    Hatchba
ck|https://ev-databa...|
|Alpine|A290 Electric 220 hp|          170|               52.0| Lithium-i
on|            184|      300|                   144|     305|
6.4|                        70|               CCS|               500|
326|    5|        FWD| B - Compact|     3997|    1823|      1512|    Hatchba
ck|https://ev-databa...|
+------+--------------------+-------------+--------------------+----------
--+---------------+---------+--------------------+--------+---------------
-----+-----------------------+---------------+------------------+------
--------+-----+----------+------------+---------+--------+---------+------
-------+--------------------+
only showing top 10 rows
```

3. Select only `brand`, `model`, and `range_km` columns.

```
In [55]: ans_df_3 = (
             df
             .select("brand", "model", "range_km")
         )
```

```
In [56]: ans_df_3.show(10)
```

```
+------+--------------------+--------+
| brand|               model|range_km|
+------+--------------------+--------+
|Abarth|    500e Convertible|     225|
|Abarth|      500e Hatchback|     225|
|Abarth| 600e Scorpionissima|     280|
|Abarth|        600e Turismo|     280|
|Aiways|                  U5|     315|
|Aiways|                  U6|     350|
|  Alfa|Romeo Junior Elet...|     320|
|  Alfa|Romeo Junior Elet...|     310|
|Alpine|A290 Electric 180 hp|     310|
|Alpine|A290 Electric 220 hp|     305|
+------+--------------------+--------+
only showing top 10 rows
```

4. Rename the column `battery_capacity_kWh` to `battery_kWh`.

```
In [58]: df_renamed = (
             df
             .withColumnRenamed("battery_capacity_kwh", "battery_kwh")
         )
```

```
In [61]: df_renamed.columns
```

```
Out[61]: ['brand',
          'model',
          'top_speed_kmh',
          'battery_kwh',
          'battery_type',
          'number_of_cells',
          'torque_nm',
          'efficiency_wh_per_km',
          'range_km',
          'acceleration_0_100_s',
          'fast_charging_power_kw_dc',
          'fast_charge_port',
          'towing_capacity_kg',
          'cargo_volume_l',
          'seats',
          'drivetrain',
          'segment',
          'length_mm',
          'width_mm',
          'height_mm',
          'car_body_type',
          'source_url']
```

---

5. Drop the `source_url` column from the DataFrame.

```
In [62]: dropped_source_url_df = (
             df
             .drop("source_url")
         )
```

```
In [66]: len(df.columns)
```

```
Out[66]: 22
```

```
In [67]: len(dropped_source_url_df.columns)
```

```
Out[67]: 21
```

Filtering & Selection

6. Filter rows where `brand` is "Abarth".

```
In [72]: (
             df
             .filter(psf.expr("brand == 'Abarth'"))
             .select("brand", "model")
             .show(10)
         )
```

```
+------+-------------------+
| brand|              model|
+------+-------------------+
|Abarth|   500e Convertible|
|Abarth|     500e Hatchback|
|Abarth|600e Scorpionissima|
|Abarth|       600e Turismo|
+------+-------------------+
```

---

7. Find all vehicles with `top_speed_kmh` greater than 180.

In [79]:
```
query = """
    SELECT
        brand,
        model,
        top_speed_kmh
    FROM
        df_view
    WHERE
        top_speed_kmh > 180
"""
```

In [80]:
```
spark.sql(query).show(10)
```

```
+------+--------------------+-------------+
| brand|               model|top_speed_kmh|
+------+--------------------+-------------+
|Abarth| 600e Scorpionissima|          200|
|Abarth|        600e Turismo|          200|
|  Alfa|Romeo Junior Elet...|          200|
|  Audi|      A6 Avant e-tron|          210|
|  Audi|A6 Avant e-tron p...|          210|
|  Audi|A6 Avant e-tron q...|          210|
|  Audi|  A6 Sportback e-tron|          210|
|  Audi|A6 Sportback e-tr...|          210|
|  Audi|A6 Sportback e-tr...|          210|
|  Audi|           Q6 e-tron|          210|
+------+--------------------+-------------+
only showing top 10 rows
```

---

8. Filter rows where `torque_nm` is not null.

In [83]:
```
(
    df
    .select("brand", "model", "torque_nm")
    .filter(psf.col("torque_nm").isNotNull())
    .show(5)
)
```

```
+------+------------------+---------+
| brand|             model|torque_nm|
+------+------------------+---------+
|Abarth|   500e Convertible|      235|
|Abarth|     500e Hatchback|      235|
|Abarth|600e Scorpionissima|      345|
|Abarth|       600e Turismo|      345|
|Aiways|                 U5|      310|
+------+------------------+---------+
only showing top 5 rows
```

---

9. Retrieve vehicles where `range_km` is between 250 and 300.

In [85]:
```
query = """
    SELECT
        brand,
        model,
        range_km
    FROM
        df_view
    WHERE
        range_km BETWEEN 250 AND 300
    ORDER BY
        range_km
"""
```

In [86]:
```
spark.sql(query).show(10)
```

```
+--------+--------------------+--------+
|   brand|               model|range_km|
+--------+--------------------+--------+
|    Mini|            Cooper E|     250|
| firefly|                NULL|     250|
| Renault|5 E-Tech 40kWh 120hp|     250|
|   Smart|             #1 Pro|     250|
|   Smart|            #1 Pure|     250|
|Dongfeng|        Box 42.3 kWh|     255|
| Citroen|                e-C3|     255|
| Hyundai|INSTER Standard R...|     255|
| Renault| 5 E-Tech 40kWh 95hp|     255|
| Citroen|e-SpaceTourer M 7...|     260|
+--------+--------------------+--------+
only showing top 10 rows
```

---

10. Filter SUVs with more than 4 seats.

In [93]:
```
(
    df
    .select("brand", "model", "car_body_type", "seats")
    .filter(psf.col("car_body_type") == "SUV")
    .filter(psf.col("seats") > 4)
    .orderBy("seats")
    .show(10)
)
```

```
+------+--------------------+------------+-----+
| brand|               model|car_body_type|seats|
+------+--------------------+------------+-----+
|Abarth| 600e Scorpionissima|         SUV|    5|
|Abarth|        600e Turismo|         SUV|    5|
|Aiways|                  U5|         SUV|    5|
|Aiways|                  U6|         SUV|    5|
|  Alfa|Romeo Junior Elet...|         SUV|    5|
|  Alfa|Romeo Junior Elet...|         SUV|    5|
|  Audi|Q4 Sportback e-tr...|         SUV|    5|
|  Audi|Q4 Sportback e-tr...|         SUV|    5|
|  Audi|Q4 Sportback e-tr...|         SUV|    5|
|  Audi|Q4 Sportback e-tr...|         SUV|    5|
+------+--------------------+------------+-----+
only showing top 10 rows
```

Aggregations

11. Calculate the average `efficiency_wh_per_km` for each `brand`.

In [99]:
```
(
    df
    .groupBy("brand")
    .avg("efficiency_wh_per_km")
    .show(10)
)
```

```
+-----------+-------------------------+
|      brand|avg(efficiency_wh_per_km)|
+-----------+-------------------------+
|  Leapmotor|                    151.0|
| Volkswagen|       150.08695652173913|
|    Peugeot|        159.3684210526316|
|        NIO|       181.08333333333334|
|      Lexus|       150.66666666666666|
|   Polestar|       156.44444444444446|
|     Jaguar|                    208.0|
|   Maserati|                    200.0|
|Rolls-Royce|                    192.0|
|       Jeep|                    140.0|
+-----------+-------------------------+
only showing top 10 rows
```

In [105…
```
(
    df
    .groupBy("brand")
    .agg(psf.avg("efficiency_wh_per_km").alias("average"))
    .show(10)
)
```

```
+-----------+-----------------+
|      brand|          average|
+-----------+-----------------+
|  Leapmotor|            151.0|
| Volkswagen|150.08695652173913|
|    Peugeot| 159.3684210526316|
|        NIO|181.08333333333334|
|      Lexus|150.66666666666666|
|   Polestar|156.44444444444446|
|     Jaguar|            208.0|
|   Maserati|            200.0|
|Rolls-Royce|            192.0|
|       Jeep|            140.0|
+-----------+-----------------+
only showing top 10 rows
```

---

12. Find the maximum `top_speed_kmh` for each `brand`.

In [106…

```
(
    df
    .select("brand", "top_speed_kmh")
    .groupBy("brand")
    .agg(psf.max("top_speed_kmh").alias("Max_speed"))
    .show(10)
)
```

```
+-----------+---------+
|      brand|Max_speed|
+-----------+---------+
|  Leapmotor|      170|
| Volkswagen|      200|
|    Peugeot|      170|
|        NIO|      200|
|      Lexus|      160|
|   Polestar|      210|
|     Jaguar|      200|
|   Maserati|      325|
|Rolls-Royce|      250|
|       Jeep|      180|
+-----------+---------+
only showing top 10 rows
```

---

13. Count how many distinct `car_body_type` values exist.

In [107…

```
(
    df
    .agg(psf.count_distinct("car_body_type").alias("distinct num"))
    .show()
)
```

```
+------------+
|distinct num|
+------------+
|           8|
+------------+
```

In [108…
```
(
    df
    .select("car_body_type")
    .distinct()
    .count()
)
```

Out[108… 8

---

14. Compute the total number of vehicles for each `segment`.

In [113…
```
(
    df
    .groupBy("segment")
    .count()
    .show(10)
)
```

```
+-----------------+-----+
|          segment|count|
+-----------------+-----+
|     JB - Compact|   44|
|N - Passenger Van|   47|
|         A - Mini|    3|
|      B - Compact|   29|
|        F - Luxury|   51|
|       JD - Large|   58|
|       C - Medium|   34|
|        D - Large|   28|
|    E - Executive|   30|
|      JF - Luxury|   30|
+-----------------+-----+
only showing top 10 rows
```

---

15. Group by `drivetrain` and calculate the average `acceleration_0_100_s`.

In [115…
```
(
    df
    .groupBy("drivetrain")
    .agg(psf.avg("acceleration_0_100_s").alias("average"))
    .show(10)
)
```

```
+----------+-----------------+
|drivetrain|          average|
+----------+-----------------+
|       FWD| 9.730128205128207|
|       AWD|4.5539267015706795|
|       RWD| 6.887022900763362|
+----------+-----------------+
```

Sorting

16. Sort the DataFrame by `range_km` in descending order.

In [117…
```
(
    df
    .select("brand", "model", "range_km")
    .sort("range_km", ascending=False)
    .show(10)
)
```

```
+-------------+------------------+--------+
|        brand|             model|range_km|
+-------------+------------------+--------+
|Mercedes-Benz|         EQS 450+|     685|
|        Lucid|   Air Grand Touring|     665|
|Mercedes-Benz|      EQS 450 4MATIC|     655|
|Mercedes-Benz|      EQS 500 4MATIC|     640|
|Mercedes-Benz|      EQS 580 4MATIC|     640|
|         Audi|A6 Sportback e-tr...|     610|
|         Audi|A6 Sportback e-tr...|     590|
|Mercedes-Benz|  EQS AMG 53 4MATIC+|     585|
|        Lucid|         Air Touring|     580|
|         Audi|A6 Avant e-tron p...|     575|
+-------------+------------------+--------+
only showing top 10 rows
```

---

17. Sort vehicles by `battery_capacity_kWh`, then by `top_speed_kmh`.

In [121…
```
(
    df
    .select("brand", "model", "battery_capacity_kWh", "top_speed_kmh")
    .orderBy("battery_capacity_kWh", "top_speed_kmh").show()
)
```

```
+---------+-------------------+--------------------+------------+
|    brand|              model|battery_capacity_kWh|top_speed_kmh|
+---------+-------------------+--------------------+------------+
|     Fiat|     500e 3+1 24 kWh|                21.3|         135|
|     Fiat|  500e Cabrio 24 kWh|                21.3|         135|
|     Fiat|500e Hatchback 24...|                21.3|         135|
|    Dacia|  Spring Electric 65|                25.0|         125|
|    Dacia|  Spring Electric 45|                25.0|         125|
| Dongfeng|        Box 31.4 kWh|                29.0|         140|
|      BYD|DOLPHIN SURF 30 k...|                30.0|         150|
|Leapmotor|                 T03|                36.0|         130|
|     Mini|            Cooper E|                36.6|         160|
|     Fiat|     500e 3+1 42 kWh|                37.3|         150|
|     Fiat|  500e Cabrio 42 kWh|                37.3|         150|
|     Fiat|500e Hatchback 42...|                37.3|         150|
|   Abarth|    500e Convertible|                37.8|         155|
|   Abarth|      500e Hatchback|                37.8|         155|
|     Mini|            Aceman E|                38.5|         160|
|  Hyundai|INSTER Standard R...|                39.0|         140|
|  Renault| 5 E-Tech 40kWh 95hp|                40.0|         130|
| Dongfeng|        Box 42.3 kWh|                40.0|         140|
|  Renault|4 E-Tech 40kWh 120hp|                40.0|         150|
|  Renault|5 E-Tech 40kWh 120hp|                40.0|         150|
+---------+-------------------+--------------------+------------+
only showing top 20 rows
```

Handling Nulls

18. Count how many null values are present in each column.

In [11]:
```
(
    df
    .select([psf.sum(psf.col(c).isNull().cast("int")).alias(c) for c in d
    .transpose()
    .show(22)
)
```

```
+--------------------+---+
|                 key|  0|
+--------------------+---+
|               model|  1|
|       top_speed_kmh|  0|
|battery_capacity_kWh|  0|
|        battery_type|  0|
|     number_of_cells|202|
|           torque_nm|  7|
|efficiency_wh_per_km|  0|
|            range_km|  0|
|acceleration_0_100_s|  0|
|fast_charging_pow...|  1|
|    fast_charge_port|  1|
|  towing_capacity_kg| 26|
|       cargo_volume_l|  4|
|               seats|  0|
|          drivetrain|  0|
|             segment|  0|
|           length_mm|  0|
|            width_mm|  0|
|           height_mm|  0|
|       car_body_type|  0|
|          source_url|  0|
+--------------------+---+
```

---

19. Drop rows with any null values.

In [127…
```
df_dropped_rows = (
    df
    .na
    .drop()
)
```

In [130…
```
(
    df_dropped_rows
    .select("brand", "model")
    .show(10)
)
```

```
+------+--------------------+
| brand|               model|
+------+--------------------+
|Abarth|    500e Convertible|
|Abarth|      500e Hatchback|
|Abarth| 600e Scorpionissima|
|Abarth|        600e Turismo|
|  Alfa|Romeo Junior Elet...|
|  Alfa|Romeo Junior Elet...|
|Alpine|A290 Electric 180 hp|
|Alpine|A290 Electric 220 hp|
|  Audi|     A6 Avant e-tron|
|  Audi|A6 Avant e-tron p...|
+------+--------------------+
only showing top 10 rows
```

---

20. Fill nulls in `number_of_cells` with 0.

In [131…
```
df_filled = (
    df
    .select("number_of_cells")
    .fillna(0)
)
```

In [132…
```
(
    df_filled
    .select("number_of_cells")
    .show(10)
)
```

```
+---------------+
|number_of_cells|
+---------------+
|            192|
|            192|
|            102|
|            102|
|              0|
|              0|
|            102|
|            102|
|            184|
|            184|
+---------------+
only showing top 10 rows
```

---

21. Fill nulls in `towing_capacity_kg` with the mean value of the column.

In [140…
```
mean_val = (
    df
    .select(psf.mean("towing_capacity_kg"))
    .first()
)[0]
```

In [141…

```
df_filled_with_mean = (
    df
    .select("brand", "model", "towing_capacity_kg")
    .fillna({"towing_capacity_kg": mean_val})
)
```

In [142…

```
df_filled_with_mean.show(10)
```

```
+------+--------------------+------------------+
| brand|               model|towing_capacity_kg|
+------+--------------------+------------------+
|Abarth|    500e Convertible|                 0|
|Abarth|      500e Hatchback|                 0|
|Abarth| 600e Scorpionissima|                 0|
|Abarth|        600e Turismo|                 0|
|Aiways|                  U5|              1052|
|Aiways|                  U6|              1052|
|  Alfa|Romeo Junior Elet...|                 0|
|  Alfa|Romeo Junior Elet...|                 0|
|Alpine|A290 Electric 180 hp|               500|
|Alpine|A290 Electric 220 hp|               500|
+------+--------------------+------------------+
only showing top 10 rows
```

Distinct & Deduplication

22. Get distinct values from the `brand` column.

In [10]:

```
(
    df
    .select("brand")
    .distinct()
    .show(10)
)
```

```
+-----------+
|      brand|
+-----------+
|  Leapmotor|
| Volkswagen|
|    Peugeot|
|        NIO|
|      Lexus|
|   Polestar|
|     Jaguar|
|   Maserati|
|Rolls-Royce|
|       Jeep|
+-----------+
only showing top 10 rows
```

---

23. Drop duplicate rows from the DataFrame.

```
In [12]: df_without_duplicates = (
             df
             .dropDuplicates()
         )
```

```
In [14]: (
             df_without_duplicates
             .select("brand", "model")
             .show(10)
         )
```

```
+-------+--------------------+
|  brand|               model|
+-------+--------------------+
|   Ford|Capri Extended Ra...|
|Citroen|e-SpaceTourer XL ...|
|Hyundai|IONIQ 6 Standard ...|
|    NIO|      EL7 Long Range|
|Porsche|      Macan Electric|
|    BYD|SEAL U 87 kWh Design|
|   Fiat|  500e Cabrio 42 kWh|
|  Skoda|      Enyaq Coupe 60|
|  Tesla|Model 3 Long Rang...|
|   Audi|Q6 e-tron perform...|
+-------+--------------------+
only showing top 10 rows
```

String & Column Operations

24. Add a new column `brand_model` by concatenating `brand` and `model`.

```
In [21]: df_with_brand_model = (
             df
             .withColumn("brand_model", psf.concat_ws(" ", "brand", "model"))
         )
```

```
In [22]: (
             df_with_brand_model
             .select("brand", "model", "brand_model")
             .show(10, truncate=False)
         )
```

```
+------+------------------------------------+----------------------------
------------+
|brand |model                               |brand_model
|
+------+------------------------------------+----------------------------
------------+
|Abarth|500e Convertible                    |Abarth 500e Convertible
|
|Abarth|500e Hatchback                      |Abarth 500e Hatchback
|
|Abarth|600e Scorpionissima                 |Abarth 600e Scorpionissima
|
|Abarth|600e Turismo                        |Abarth 600e Turismo
|
|Aiways|U5                                  |Aiways U5
|
|Aiways|U6                                  |Aiways U6
|
|Alfa  |Romeo Junior Elettrica 54 kWh       |Alfa Romeo Junior Elettrica 5
4 kWh       |
|Alfa  |Romeo Junior Elettrica 54 kWh Veloce|Alfa Romeo Junior Elettrica 5
4 kWh Veloce|
|Alpine|A290 Electric 180 hp                |Alpine A290 Electric 180 hp
|
|Alpine|A290 Electric 220 hp                |Alpine A290 Electric 220 hp
|
+------+------------------------------------+----------------------------
------------+
only showing top 10 rows
```

---

25. Extract domain name from the `source_url`.

In [25]:
```python
(
    df
    .withColumn("domain", psf.regexp_extract("source_url", r"https?://([^
    .select("source_url", "domain")
    .show(10, truncate=False)
)
```

```
+--------------------------------------------------------------------------
-+---------------+
|source_url
|domain         |
+--------------------------------------------------------------------------
-+---------------+
|https://ev-database.org/car/1904/Abarth-500e-Convertible
|ev-database.org|
|https://ev-database.org/car/1903/Abarth-500e-Hatchback
|ev-database.org|
|https://ev-database.org/car/3057/Abarth-600e-Scorpionissima
|ev-database.org|
|https://ev-database.org/car/3056/Abarth-600e-Turismo
|ev-database.org|
|https://ev-database.org/car/1678/Aiways-U5
|ev-database.org|
|https://ev-database.org/car/1766/Aiways-U6
|ev-database.org|
|https://ev-database.org/car/2184/Alfa-Romeo-Junior-Elettrica-54-kWh
|ev-database.org|
|https://ev-database.org/car/2185/Alfa-Romeo-Junior-Elettrica-54-kWh-Veloc
e|ev-database.org|
|https://ev-database.org/car/2268/Alpine-A290-Electric-180-hp
|ev-database.org|
|https://ev-database.org/car/2269/Alpine-A290-Electric-220-hp
|ev-database.org|
+--------------------------------------------------------------------------
-+---------------+
only showing top 10 rows
```

Type Casting

26. Cast `battery_capacity_kWh` from float to integer.

In [26]:
```
df_battery_in_int = (
    df.withColumn("battery_capacity_kWh_int", psf.col("battery_capacity_k
    .drop("battery_capacity_kWh")
)
```

In [27]:
```
df_battery_in_int.select("brand", "battery_capacity_kWh_int").show(10)
```

```
+------+-----------------------+
| brand|battery_capacity_kWh_int|
+------+-----------------------+
|Abarth|                     37|
|Abarth|                     37|
|Abarth|                     50|
|Abarth|                     50|
|Aiways|                     60|
|Aiways|                     60|
|  Alfa|                     50|
|  Alfa|                     50|
|Alpine|                     52|
|Alpine|                     52|
+------+-----------------------+
only showing top 10 rows
```

---

27. Convert `range_km` to string.

In [29]:
```
df_range_km_in_string = (
    df
    .withColumn("range_km_str", psf.col("range_km").cast("string"))
    .drop("range_km")
)
```

In [32]:
```
(
    df_range_km_in_string
    .select("brand", "range_km_str")
    .show(10)
)
```

```
+------+------------+
| brand|range_km_str|
+------+------------+
|Abarth|         225|
|Abarth|         225|
|Abarth|         280|
|Abarth|         280|
|Aiways|         315|
|Aiways|         350|
|  Alfa|         320|
|  Alfa|         310|
|Alpine|         310|
|Alpine|         305|
+------+------------+
only showing top 10 rows
```

Basic UDFs

28. Create a UDF to categorize cars as "High Range" (>300 km) or "Low Range".

In [15]:
```
# Using pandas UDF
def categorize_cars(range_km: pd.Series) -> pd.Series:
    return range_km.apply(lambda x: "High Range" if x > 300 else "Low Ran
```

In [16]:
```
categorize_udf = psf.pandas_udf(categorize_cars, returnType=pst.StringTyp
```

In [17]:
```
df_with_categories = (
    df
    .withColumn("Category", categorize_udf(psf.col("range_km")))
)
```

In [18]:
```
(
    df_with_categories
    .select("brand", "range_km", "Category")
    .show(10)
)
```

```
+------+--------+----------+
| brand|range_km|  Category|
+------+--------+----------+
|Abarth|     225| Low Range|
|Abarth|     225| Low Range|
|Abarth|     280| Low Range|
|Abarth|     280| Low Range|
|Aiways|     315|High Range|
|Aiways|     350|High Range|
|  Alfa|     320|High Range|
|  Alfa|     310|High Range|
|Alpine|     310|High Range|
|Alpine|     305|High Range|
+------+--------+----------+
only showing top 10 rows
```

---

29. Create a UDF to compute `power_density = battery_capacity_kWh / length_mm`.

In [20]:
```
# Using pandas UDF
def compute_power_density(capacity: pd.Series, length: pd.Series) -> pd.S
    result = capacity * length
    return result
```

In [25]:
```
compute_power_density_udf = psf.pandas_udf(compute_power_density, returnT
```

In [26]:
```
df_with_power_density = (
    df
    .withColumn("power_density", compute_power_density_udf(psf.col("batte
)
```

In [27]:
```
(
    df_with_power_density
    .select("brand", "battery_capacity_kWh", "length_mm", "power_density"
    .show(10)
)
```

```
+------+--------------------+---------+-------------+
| brand|battery_capacity_kWh|length_mm|power_density|
+------+--------------------+---------+-------------+
|Abarth|                37.8|     3673|     138839.4|
|Abarth|                37.8|     3673|     138839.4|
|Abarth|                50.8|     4187|     212699.6|
|Abarth|                50.8|     4187|     212699.6|
|Aiways|                60.0|     4680|     280800.0|
|Aiways|                60.0|     4805|     288300.0|
|  Alfa|                50.8|     4173|     211988.4|
|  Alfa|                50.8|     4173|     211988.4|
|Alpine|                52.0|     3997|     207844.0|
|Alpine|                52.0|     3997|     207844.0|
+------+--------------------+---------+-------------+
only showing top 10 rows
```

More Selections

30. Create a column `is_fast_charge_supported` where `fast_charging_power_kw_dc` > 50.

In [35]:
```
df_with_new_col = (
    df
    .withColumn("is_fast_charge_supported", psf.expr("fast_charging_power
)
```

In [37]:
```
(
    df_with_new_col
    .select("brand", "fast_charging_power_kw_dc", "is_fast_charge_support
    .show(10)
)
```

```
+------+-------------------------+------------------------+
| brand|fast_charging_power_kw_dc|is_fast_charge_supported|
+------+-------------------------+------------------------+
|Abarth|                       67|                    true|
|Abarth|                       67|                    true|
|Abarth|                       79|                    true|
|Abarth|                       79|                    true|
|Aiways|                       78|                    true|
|Aiways|                       78|                    true|
|  Alfa|                       85|                    true|
|  Alfa|                       85|                    true|
|Alpine|                       70|                    true|
|Alpine|                       70|                    true|
+------+-------------------------+------------------------+
only showing top 10 rows
```

---

31. Filter all rows where `car_body_type` is "SUV".

In [41]:
```
df_SUV_only = (
    df
    .where(psf.expr("car_body_type == 'SUV'"))
)
```

In [42]:
```
(
    df_SUV_only
    .select("brand", "car_body_type")
    .show(10)
)
```

```
+------+-------------+
| brand|car_body_type|
+------+-------------+
|Abarth|          SUV|
|Abarth|          SUV|
|Aiways|          SUV|
|Aiways|          SUV|
|  Alfa|          SUV|
|  Alfa|          SUV|
|  Audi|          SUV|
|  Audi|          SUV|
|  Audi|          SUV|
|  Audi|          SUV|
+------+-------------+
only showing top 10 rows
```

DataFrame Metadata

32. Show all column names and their data types.

In [45]:
```
df.printSchema()
```

```
root
 |-- brand: string (nullable = true)
 |-- model: string (nullable = true)
 |-- top_speed_kmh: integer (nullable = true)
 |-- battery_capacity_kWh: double (nullable = true)
 |-- battery_type: string (nullable = true)
 |-- number_of_cells: integer (nullable = true)
 |-- torque_nm: integer (nullable = true)
 |-- efficiency_wh_per_km: integer (nullable = true)
 |-- range_km: integer (nullable = true)
 |-- acceleration_0_100_s: double (nullable = true)
 |-- fast_charging_power_kw_dc: integer (nullable = true)
 |-- fast_charge_port: string (nullable = true)
 |-- towing_capacity_kg: integer (nullable = true)
 |-- cargo_volume_l: integer (nullable = true)
 |-- seats: integer (nullable = true)
 |-- drivetrain: string (nullable = true)
 |-- segment: string (nullable = true)
 |-- length_mm: integer (nullable = true)
 |-- width_mm: integer (nullable = true)
 |-- height_mm: integer (nullable = true)
 |-- car_body_type: string (nullable = true)
 |-- source_url: string (nullable = true)
```

```
In [46]: df.dtypes
```

```
Out[46]: [('brand', 'string'),
          ('model', 'string'),
          ('top_speed_kmh', 'int'),
          ('battery_capacity_kWh', 'double'),
          ('battery_type', 'string'),
          ('number_of_cells', 'int'),
          ('torque_nm', 'int'),
          ('efficiency_wh_per_km', 'int'),
          ('range_km', 'int'),
          ('acceleration_0_100_s', 'double'),
          ('fast_charging_power_kw_dc', 'int'),
          ('fast_charge_port', 'string'),
          ('towing_capacity_kg', 'int'),
          ('cargo_volume_l', 'int'),
          ('seats', 'int'),
          ('drivetrain', 'string'),
          ('segment', 'string'),
          ('length_mm', 'int'),
          ('width_mm', 'int'),
          ('height_mm', 'int'),
          ('car_body_type', 'string'),
          ('source_url', 'string')]
```

---

33. Count total number of rows in the DataFrame.

```
In [47]: (
             df
             .count()
         )
```

```
Out[47]: 478
```

---

34. Check if all entries in the `drivetrain` column are the same.

```
In [48]: (
             df
             .select("drivetrain")
             .distinct()
             .show()
         )
```

```
+----------+
|drivetrain|
+----------+
|       FWD|
|       AWD|
|       RWD|
+----------+
```

---

35. Count the number of vehicles per number of seats.

In [50]:
```
(
    df
    .groupBy("seats")
    .agg(psf.count("*").alias("count_per_seats"))
    .show()
)
```

```
+-----+---------------+
|seats|count_per_seats|
+-----+---------------+
|    6|              5|
|    5|            383|
|    9|             15|
|    4|             27|
|    8|              7|
|    7|             38|
|    2|              3|
+-----+---------------+
```

Medium Level (15 Questions)

Focus: Window functions, advanced aggregations, joins, UDFs, pivot/unpivot, JSON export, date functions (simulated), performance tuning basics.

Window Functions & Advanced Aggregations

36. For each `brand`, rank vehicles by `range_km` in descending order.

In [54]:
```
windowSpec = psw.Window.partitionBy("brand").orderBy(psf.col("range_km").
```

In [55]:
```
df_with_range_rank = (
    df
    .withColumn("range_rank", psf.rank().over(windowSpec))
)
```

In [57]:
```
(
    df_with_range_rank
    .select("brand", "range_km", "range_rank")
    .show(10)
)
```

```
+------+--------+----------+
| brand|range_km|range_rank|
+------+--------+----------+
|Abarth|     280|         1|
|Abarth|     280|         1|
|Abarth|     225|         3|
|Abarth|     225|         3|
|Aiways|     350|         1|
|Aiways|     315|         2|
|  Alfa|     320|         1|
|  Alfa|     310|         2|
|Alpine|     310|         1|
|Alpine|     305|         2|
+------+--------+----------+
only showing top 10 rows
```

---

37. Calculate the average `battery_capacity_kWh` and standard deviation per `car_body_type`.

In [58]:
```
df_with_avg_stdDev = (
    df
    .groupBy("car_body_type")
    .agg(psf.mean("battery_capacity_kWh").alias("average"),
         psf.stddev("battery_capacity_kWh").alias("standard deviation"))
)
```

In [59]:
```
df_with_avg_stdDev.show(10)
```

```
+------------------+----------------+------------------+
|     car_body_type|         average|standard deviation|
+------------------+----------------+------------------+
|         Hatchback|49.73684210526315| 14.51156720367469|
|               SUV|76.68729508196722|18.244932845760577|
|             Sedan| 86.8047619047619| 16.47529361010512|
|     Liftback Sedan|85.52121212121212|12.055619995871156|
|Small Passenger Van|60.01489361702127|14.491211410389768|
|         Cabriolet|            58.08| 27.10289652417247|
|             Coupe|             92.5|13.435028842544403|
|     Station/Estate|83.67407407407407|15.848309606356047|
+------------------+----------------+------------------+
```

---

38. Find the vehicle with the longest `range_km` for each `segment`.

In [62]:
```
(
    df
    .groupBy("segment")
    .agg(psf.max("range_km").alias("longest_range"))
    .show()
)
```

```
+----------------+------------+
|         segment|longest_range|
+----------------+------------+
|     JB - Compact|         455|
|N - Passenger Van|         370|
|         A - Mini|         225|
|      B - Compact|         330|
|       F - Luxury|         685|
|       JD - Large|         545|
|       C - Medium|         495|
|        D - Large|         565|
|    E - Executive|         555|
|      JF - Luxury|         540|
|       I - Luxury|         465|
|        JA - Mini|         300|
|      JC - Medium|         500|
|       G - Sports|         425|
|   JE - Executive|         610|
+----------------+------------+
```

Pivoting

39. Pivot the data to show average `range_km` for each `car_body_type` per `drivetrain`.

In [76]:
```
(
    df
    .groupBy("drivetrain")
    .pivot("car_body_type")
    .agg(psf.avg("range_km"))
    .transpose()  # Just for printing it
    .show(10)
)
```

```
+------------------+----------------+-----------------+----------------
--+
|               key|             AWD|              FWD|               R
WD|
+------------------+----------------+-----------------+----------------
--+
|         Cabriolet|           395.0|            182.5|             42
5.0|
|             Coupe|           442.5|             NULL|              NU
LL|
|         Hatchback|           320.0|277.95454545454544|             39
2.5|
|     Liftback Sedan|           505.0|             NULL|             46
8.0|
|               SUV|424.1818181818182| 348.2258064516129| 395.83333333333
33|
|             Sedan|          510.75|            362.5|511.470588235294
14|
|Small Passenger Van|           340.0|234.73684210526315|277.857142857142
83|
|     Station/Estate|        488.4375|            302.5|497.857142857142
83|
+------------------+----------------+-----------------+----------------
--+
```

Advanced UDFs

40. Create a UDF to classify vehicles into: "City EV", "Highway EV", or "Performance EV" based on `acceleration_0_100_s` and `range_km`.

def classify_vehicle(accel, range_km): if accel is None or range_km is None: return "Unknown" if accel < 6: return "Performance EV" elif range_km > 300: return "Highway EV" else: return "City EV" from pyspark.sql.functions import udf from pyspark.sql.types import StringType classify_udf = udf(classify_vehicle, StringType()) df.withColumn("ev_type", classify_udf("acceleration_0_100_s", "range_km")).show()

In [78]:
```python
# Using pandas UDFs for better performance
def classify_vehicle(acceleration: pd.Series, range_km: pd.Series) -> pd.
    result = []
    for a, r in zip(acceleration, range_km):
        if pd.isnull(a) or pd.isnull(r):
            result.append("Unknown")
        elif a < 6:
            result.append("Performance EV")
        elif r > 300:
            result.append("Highway EV")
        else:
            result.append("City EV")

    return pd.Series(result)
```

In [79]:
```python
classify_vehicles_udf = psf.pandas_udf(classify_vehicle, returnType=pst.S
```

In [80]:
```python
df_with_classified_vehicles = (
    df
    .withColumn("class", classify_vehicles_udf(psf.col("acceleration_0_10
)
```

In [81]:
```
(
    df_with_classified_vehicles
    .select("brand", "acceleration_0_100_s", "range_km", "class")
    .show(10)
)
```

```
+------+--------------------+--------+--------------+
| brand|acceleration_0_100_s|range_km|         class|
+------+--------------------+--------+--------------+
|Abarth|                 7.0|     225|       City EV|
|Abarth|                 7.0|     225|       City EV|
|Abarth|                 5.9|     280|Performance EV|
|Abarth|                 6.2|     280|       City EV|
|Aiways|                 7.5|     315|    Highway EV|
|Aiways|                 7.0|     350|    Highway EV|
|  Alfa|                 9.0|     320|    Highway EV|
|  Alfa|                 6.0|     310|    Highway EV|
|Alpine|                 7.4|     310|    Highway EV|
|Alpine|                 6.4|     305|    Highway EV|
+------+--------------------+--------+--------------+
only showing top 10 rows
```

Joins (Synthetic example)

41. Assume you have another DataFrame with `brand` and `country`. Join it with the EV DataFrame on `brand`.

```
(
    df
    .join(df_country, on="brand", how="left")
    .show()
)
```

Complex Filtering & Conditions

42. Find SUVs with a top speed over 180 km/h and efficiency under 160 Wh/km.

In [85]:
```
df.columns
```

```
Out[85]: ['brand',
          'model',
          'top_speed_kmh',
          'battery_capacity_kWh',
          'battery_type',
          'number_of_cells',
          'torque_nm',
          'efficiency_wh_per_km',
          'range_km',
          'acceleration_0_100_s',
          'fast_charging_power_kw_dc',
          'fast_charge_port',
          'towing_capacity_kg',
          'cargo_volume_l',
          'seats',
          'drivetrain',
          'segment',
          'length_mm',
          'width_mm',
          'height_mm',
          'car_body_type',
          'source_url']
```

```
In [87]: (
             df
             .select("brand", "model", "top_speed_kmh", "efficiency_wh_per_km")
             .filter(psf.col("car_body_type") == "SUV")
             .filter(psf.col("top_speed_kmh") > 180)
             .filter(psf.col("efficiency_wh_per_km") < 160)
             .show(10)
         )
```

```
+-------+--------------------+-------------+--------------------+
|  brand|               model|top_speed_kmh|efficiency_wh_per_km|
+-------+--------------------+-------------+--------------------+
| Abarth| 600e Scorpionissima|          200|                 158|
| Abarth|        600e Turismo|          200|                 158|
|   Audi| Q6 e-tron Sportback|          210|                 139|
|   Audi|Q6 e-tron Sportba...|          210|                 145|
|   Audi|Q6 e-tron Sportba...|          210|                 149|
|     DS|  N°8 AWD Long Range|          190|                 146|
|     DS|             N°8 FWD|          190|                 141|
|     DS|  N°8 FWD Long Range|          190|                 136|
|Genesis|        GV60 Premium|          185|                 143|
|Genesis|          GV60 Sport|          200|                 157|
+-------+--------------------+-------------+--------------------+
only showing top 10 rows
```

JSON Export & Schema

43. Write the DataFrame to a JSON file with inferred schema.

```
In [92]: file_path = "exported_json"
In [93]: (
             df
             .write
             .mode("overwrite")
             .format("json")
             .save(file_path)
         )
```

---

44. Infer schema manually using `StructType` and load the json using it.

```
In [94]: file_path = "exported_json/part-00000-e1c2171c-df5c-4740-96bd-089ebb96ea1
```

```
In [97]: schema = pst.StructType([
             pst.StructField("brand", pst.StringType(), True),
             pst.StructField("model", pst.StringType(), True),
             pst.StructField("top_speed_kmh", pst.IntegerType(), True),
             pst.StructField("battery_capacity_kWh", pst.DoubleType(), True),
             pst.StructField("battery_type", pst.StringType(), True),
             pst.StructField("number_of_cells", pst.DoubleType(), True),
             pst.StructField("torque_nm", pst.DoubleType(), True),
             pst.StructField("efficiency_wh_per_km", pst.IntegerType(), True),
             pst.StructField("range_km", pst.IntegerType(), True),
             pst.StructField("acceleration_0_100_s", pst.DoubleType(), True),
             pst.StructField("fast_charging_power_kw_dc", pst.DoubleType(), True),
             pst.StructField("fast_charge_port", pst.StringType(), True),
             pst.StructField("towing_capacity_kg", pst.DoubleType(), True),
             pst.StructField("cargo_volume_l", pst.IntegerType(), True),
             pst.StructField("seats", pst.IntegerType(), True),
             pst.StructField("drivetrain", pst.StringType(), True),
             pst.StructField("segment", pst.StringType(), True),
             pst.StructField("length_mm", pst.IntegerType(), True),
             pst.StructField("width_mm", pst.IntegerType(), True),
             pst.StructField("height_mm", pst.IntegerType(), True),
             pst.StructField("car_body_type", pst.StringType(), True),
             pst.StructField("source_url", pst.StringType(), True)
         ])
```

```
In [99]: df_read_from_json = (
             spark
             .read
             .schema(schema)
             .json(file_path)
         )
```

```
In [100…  (
             df_read_from_json
             .select("brand", "model", "car_body_type", "source_url")
             .show(10)
         )
```

```
+------+-------------------+------------+--------------------+
| brand|              model|car_body_type|          source_url|
+------+-------------------+------------+--------------------+
|Abarth|    500e Convertible|    Hatchback|https://ev-databa...|
|Abarth|      500e Hatchback|    Hatchback|https://ev-databa...|
|Abarth| 600e Scorpionissima|          SUV|https://ev-databa...|
|Abarth|        600e Turismo|          SUV|https://ev-databa...|
|Aiways|                  U5|          SUV|https://ev-databa...|
|Aiways|                  U6|          SUV|https://ev-databa...|
|  Alfa|Romeo Junior Elet...|          SUV|https://ev-databa...|
|  Alfa|Romeo Junior Elet...|          SUV|https://ev-databa...|
|Alpine|A290 Electric 180 hp|    Hatchback|https://ev-databa...|
|Alpine|A290 Electric 220 hp|    Hatchback|https://ev-databa...|
+------+-------------------+------------+--------------------+
only showing top 10 rows
```

Unpivot (Melt-like)

45. Transform the column-based format into a long format for `top_speed_kmh`, `range_km`, and `efficiency_wh_per_km`.

In [103…

```python
from pyspark.sql.functions import posexplode, array, struct, lit, explode

unpivoted = df.select(
    "brand", "model",
    explode(array(
        struct(lit("top_speed_kmh").alias("metric"), col("top_speed_kmh")
        struct(lit("range_km").alias("metric"), col("range_km").alias("va
        struct(lit("efficiency_wh_per_km").alias("metric"), col("efficien
    )).alias("kv")
).select("brand", "model", col("kv.metric"), col("kv.value"))

unpivoted.show()
```

```
+------+--------------------+--------------------+-----+
| brand|               model|              metric|value|
+------+--------------------+--------------------+-----+
|Abarth|    500e Convertible|       top_speed_kmh|  155|
|Abarth|    500e Convertible|            range_km|  225|
|Abarth|    500e Convertible|efficiency_wh_per_km|  156|
|Abarth|      500e Hatchback|       top_speed_kmh|  155|
|Abarth|      500e Hatchback|            range_km|  225|
|Abarth|      500e Hatchback|efficiency_wh_per_km|  149|
|Abarth| 600e Scorpionissima|       top_speed_kmh|  200|
|Abarth| 600e Scorpionissima|            range_km|  280|
|Abarth| 600e Scorpionissima|efficiency_wh_per_km|  158|
|Abarth|        600e Turismo|       top_speed_kmh|  200|
|Abarth|        600e Turismo|            range_km|  280|
|Abarth|        600e Turismo|efficiency_wh_per_km|  158|
|Aiways|                  U5|       top_speed_kmh|  150|
|Aiways|                  U5|            range_km|  315|
|Aiways|                  U5|efficiency_wh_per_km|  156|
|Aiways|                  U6|       top_speed_kmh|  160|
|Aiways|                  U6|            range_km|  350|
|Aiways|                  U6|efficiency_wh_per_km|  150|
|  Alfa|Romeo Junior Elet...|       top_speed_kmh|  150|
|  Alfa|Romeo Junior Elet...|            range_km|  320|
+------+--------------------+--------------------+-----+
only showing top 20 rows
```

> below is the efficient version (using pandas UDF)

```
from pyspark.sql.functions import pandas_udf
from pyspark.sql.types import StructType, StructField,
StringType, DoubleType
import pandas as pd

# Define output schema for the long format
output_schema = StructType([
    StructField("brand", StringType()),
    StructField("model", StringType()),
    StructField("metric", StringType()),
    StructField("value", DoubleType())
])

# Pandas UDF for unpivoting
@pandas_udf(output_schema)
def unpivot_udf(pdf: pd.DataFrame) -> pd.DataFrame:
    id_vars = ["brand", "model"]
    value_vars = ["top_speed_kmh", "range_km",
"efficiency_wh_per_km"]

    # Melt to long format
    melted = pd.melt(
        pdf,
        id_vars=id_vars,
        value_vars=value_vars,
        var_name="metric",
        value_name="value"
    )
    return melted
```

```
unpivoted = df.groupBy("brand", "model").apply(unpivot_udf)
```

Map & ReduceByKey (RDD-style)

46. Convert DataFrame to RDD and find average `battery_capacity_kWh` per brand using `reduceByKey`.

```
rdd = df.select("brand", "battery_capacity_kWh").rdd \
        .filter(lambda x: x[1] is not None) \
        .map(lambda x: (x[0], (x[1], 1)))

brand_avg = rdd.reduceByKey(lambda a, b: (a[0] + b[0], a[1] + 
b[1])) \
                    .mapValues(lambda x: x[0] / x[1])

brand_avg.collect()
```

Performance Optimization

47. Cache the DataFrame and perform a groupBy operation. Measure execution time before and after caching.

In [104…
```
import time

start = time.time()
df.groupBy("car_body_type").count().show()
print("Without cache:", time.time() - start)

df.cache()

start = time.time()
df.groupBy("car_body_type").count().show()
print("With cache:", time.time() - start)
```

```
+------------------+-----+
|      car_body_type|count|
+------------------+-----+
|          Hatchback|   57|
|                SUV|  244|
|              Sedan|   63|
|      Liftback Sedan|   33|
|Small Passenger Van|   47|
|          Cabriolet|    5|
|              Coupe|    2|
|      Station/Estate|   27|
+------------------+-----+

Without cache: 0.4533965587615967
+------------------+-----+
|      car_body_type|count|
+------------------+-----+
|          Hatchback|   57|
|                SUV|  244|
|              Sedan|   63|
|      Liftback Sedan|   33|
|Small Passenger Van|   47|
|          Cabriolet|    5|
|              Coupe|    2|
|      Station/Estate|   27|
+------------------+-----+

With cache: 0.3217630386352539
```

Nested Column Creation

48. Create a struct column called `performance` with `top_speed_kmh`, `acceleration_0_100_s`, and `torque_nm`.

df.withColumn("performance", struct("top_speed_kmh", "torque_nm", "acceleration_0_100_s")).select("brand", "model", "performance").show(truncate=False)

In [105…
```
df_with_performance_col = (
    df
    .withColumn("performance", psf.struct("top_speed_kmh", "torque_nm", "
    .select("brand", "model", "performance")
)
```

In [106…
```
df_with_performance_col.show(10)
```

```
+------+-------------------+--------------+
| brand|              model|   performance|
+------+-------------------+--------------+
|Abarth|    500e Convertible|{155, 235, 7.0}|
|Abarth|      500e Hatchback|{155, 235, 7.0}|
|Abarth| 600e Scorpionissima|{200, 345, 5.9}|
|Abarth|        600e Turismo|{200, 345, 6.2}|
|Aiways|                  U5|{150, 310, 7.5}|
|Aiways|                  U6|{160, 315, 7.0}|
|  Alfa|Romeo Junior Elet...|{150, 260, 9.0}|
|  Alfa|Romeo Junior Elet...|{200, 345, 6.0}|
|Alpine|A290 Electric 180 hp|{160, 285, 7.4}|
|Alpine|A290 Electric 220 hp|{170, 300, 6.4}|
+------+-------------------+--------------+
only showing top 10 rows
```

Exploding Arrays

49. Create an array column of all performance metrics and explode it row-wise.

In [107…

```
(
    df
    .withColumn("performance_array", array("top_speed_kmh", "torque_nm",
    .withColumn("exploded_metric", explode("performance_array"))
    .select("brand", "model", "exploded_metric")
    .show(10)
)
```

```
+------+------------------+---------------+
| brand|             model|exploded_metric|
+------+------------------+---------------+
|Abarth|  500e Convertible|          155.0|
|Abarth|  500e Convertible|          235.0|
|Abarth|  500e Convertible|            7.0|
|Abarth|    500e Hatchback|          155.0|
|Abarth|    500e Hatchback|          235.0|
|Abarth|    500e Hatchback|            7.0|
|Abarth|600e Scorpionissima|          200.0|
|Abarth|600e Scorpionissima|          345.0|
|Abarth|600e Scorpionissima|            5.9|
|Abarth|      600e Turismo|          200.0|
+------+------------------+---------------+
only showing top 10 rows
```

Complex Sorting

50. Sort cars by acceleration (ascending), then by torque (descending), and then by range.

In [109…

```
(
    df
    .orderBy(
        psf.col("acceleration_0_100_s").asc(),
        psf.col("torque_nm").desc(),
        psf.col("range_km").desc()
    )
    .select("brand", "model", "acceleration_0_100_s", "torque_nm", "range
    .show()
)
```

```
+--------+--------------------+--------------------+---------+--------+
|   brand|               model|acceleration_0_100_s|torque_nm|range_km|
+--------+--------------------+--------------------+---------+--------+
| Porsche|Taycan Turbo GT W...|                 2.2|     1340|     475|
| Porsche|     Taycan Turbo GT|                 2.3|     1340|     475|
|   Tesla|       Model S Plaid|                 2.3|     NULL|     560|
| Porsche|      Taycan Turbo S|                 2.4|     1110|     525|
| Porsche|Taycan Turbo S Sp...|                 2.4|     1110|     505|
| Porsche|Taycan Turbo S Cr...|                 2.5|     1110|     485|
|    Audi|e-tron GT RS perf...|                 2.5|     1027|     525|
|Maserati| GranTurismo Folgore|                 2.7|     1350|     420|
| Porsche|        Taycan Turbo|                 2.7|      940|     535|
| Porsche|Taycan Turbo Spor...|                 2.7|      940|     505|
|   Tesla|       Model X Plaid|                 2.7|     NULL|     465|
|Maserati|  GranCabrio Folgore|                 2.8|     1350|     395|
|   Lotus|             Emeya R|                 2.8|      985|     465|
| Porsche|Taycan Turbo Cros...|                 2.8|      940|     495|
|    Audi|         e-tron GT RS|                 2.8|      865|     525|
|   Lotus|            Eletre R|                 2.9|      985|     455|
|   Lucid|   Air Grand Touring|                 3.0|     1200|     665|
|   Tesla|Model 3 Performan...|                 3.2|      741|     490|
|      MG|         Cyberster GT|                 3.2|      725|     395|
|   Lucid|         Air Touring|                 3.2|     NULL|     580|
+--------+--------------------+--------------------+---------+--------+
only showing top 20 rows
```