

# **Proces ELT și Analiză a Datelor din Domeniul Auto**

**- proiect pentru Sisteme de Baze de Date Evolute -**

**Student: Neacșu David-Andrei**

**Grupa: 1109**

**An universitar: 2025–2026**

## **1. Introducere**

Proiectul de față își propune realizarea analizelor de date din domeniul auto care a trecut printr-un process complet de ELT (Extract, Load, Transform). Acesta are scopul de a demonstra utilizarea conceptelor avansate de proiectare, normalizare și analiză a bazelor de date relaționale. Prin intermediul acestui proiect, se dorește integrarea și analiza datelor tehnice și comerciale aferente vehiculelor din diferite mărci și modele.

## **2. Sursele de date**

Datele utilizate au fost prelucrate, curățate și normalizate în vederea integrării într-un model relațional unitar. Pentru implementarea proiectului au fost utilizate două surse principale de date:

- Dataset-ul „Vehicle Technical Specifications and Environmental Performance” – disponibil pe platforma Opendatabay, care conține informații tehnice și ecologice despre vehicule - <https://www.opendatabay.com/data/dataset/99b921ee-d99b-414f-af61-36978fa36d92>.
- Dataset-ul „Vehicle Sales Cleaned” – preluat de pe Kaggle, care oferă informații despre tranzacțiile de vânzare, prețuri, kilometraj și alte detalii comerciale - <https://www.kaggle.com/datasets/krishanukalita/vehicle-sales-cleaned>.

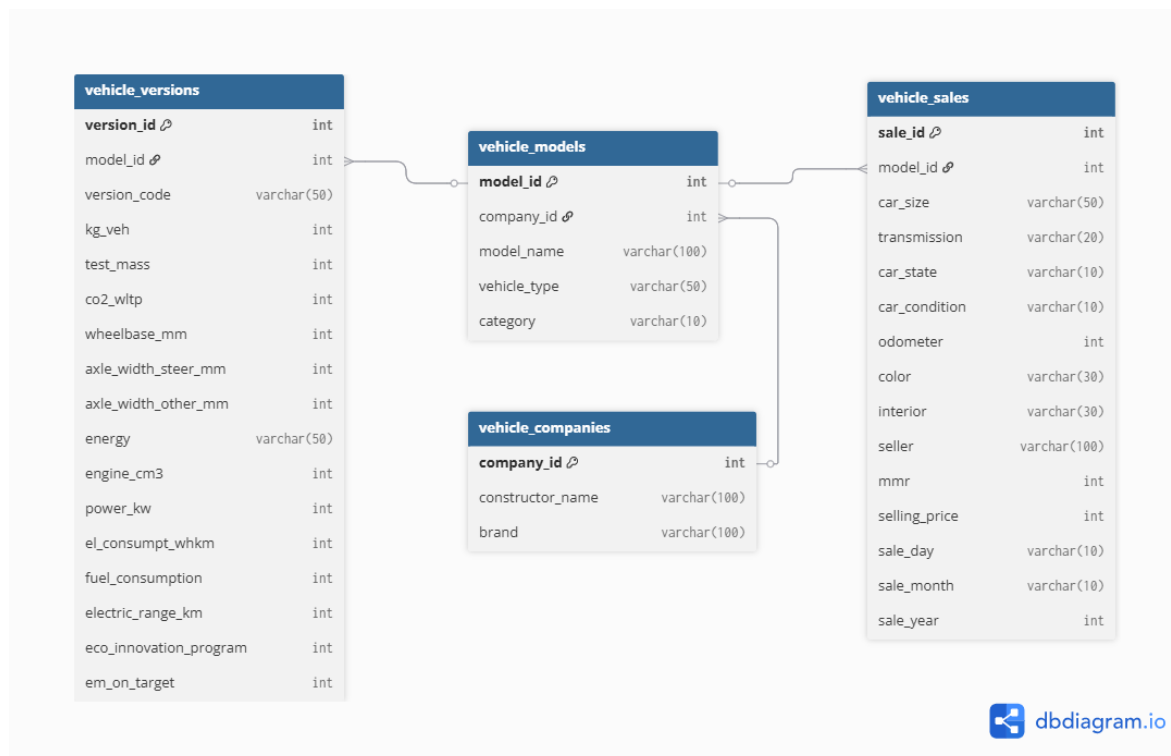
## **3. Modelarea bazei de date**

Baza de date a fost proiectată conform principiilor normalizării relaționale, fiind organizată în mai multe entități care reflectă structura domeniului auto. Schema finală conține următoarele tabele principale:

- vehicle\_companies – stochează informații despre constructorii și mărcile auto.
- vehicle\_models – conține detalii despre modelele de vehicule.
- vehicle\_versions – reține specificațiile tehnice ale fiecărei versiuni.
- vehicle\_sales – stochează informații referitoare la vânzările efectuate.

Relațiile dintre aceste tabele sunt de tipul 1–N, fiecare companie putând avea mai multe modele, iar fiecare model mai multe versiuni și înregistrări de vânzări.

Schema conceptuală a bazei de date:



## 4. Procesul ETL

Procesul ELT (Extract, Load, Transform) a fost implementat în mai multe etape:

1. Crearea tabelor brute (raw tables) care vor datele preluate direct din fișierele sursă utilizând SQL Developer.

```
CREATE TABLE raw_vehicle_data (  
  v_id NUMBER,  
  constructor_name VARCHAR2(100),  
  vehicle_type VARCHAR2(50),  
  version VARCHAR2(50),  
  brand VARCHAR2(100),  
  vehicle_model VARCHAR2(100),  
  vehicle_category VARCHAR2(10),
```

```
kg_veh NUMBER,  
test_mass NUMBER,  
co2_wltp NUMBER,  
wheelbase_mm NUMBER,  
axle_width_steer_mm NUMBER,  
axle_width_other_mm NUMBER,  
energy VARCHAR2(50),  
engine_cm3 NUMBER,  
power_kw NUMBER,  
el_consumpt_whkm NUMBER,  
year NUMBER(4),  
fuel_consumption NUMBER,  
electric_range_km NUMBER,  
eco_innovation_program NUMBER,  
em_on_target NUMBER  
);  
  
CREATE TABLE raw_vehicle_sales (  
    company VARCHAR2(50),  
    model VARCHAR2(100),  
    model_type VARCHAR2(50),  
    car_size VARCHAR2(50),  
    transmission VARCHAR2(20),  
    car_state VARCHAR2(10),  
    car_condition VARCHAR2(10),  
    odometer NUMBER,  
    color VARCHAR2(30),  
    interior VARCHAR2(30),  
    seller VARCHAR2(100),  
    mmr NUMBER,  
    selling_price NUMBER,  
    sale_day VARCHAR2(10),  
    sale_month VARCHAR2(10),  
    sale_year NUMBER  
);
```

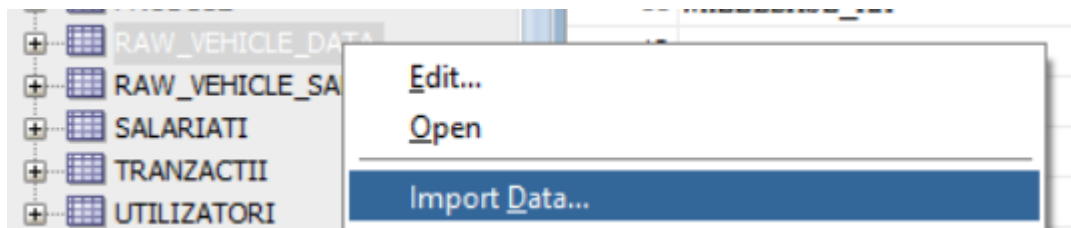
Table RAW\_VEHICLE\_DATA created.

Table RAW\_VEHICLE\_SALES created.

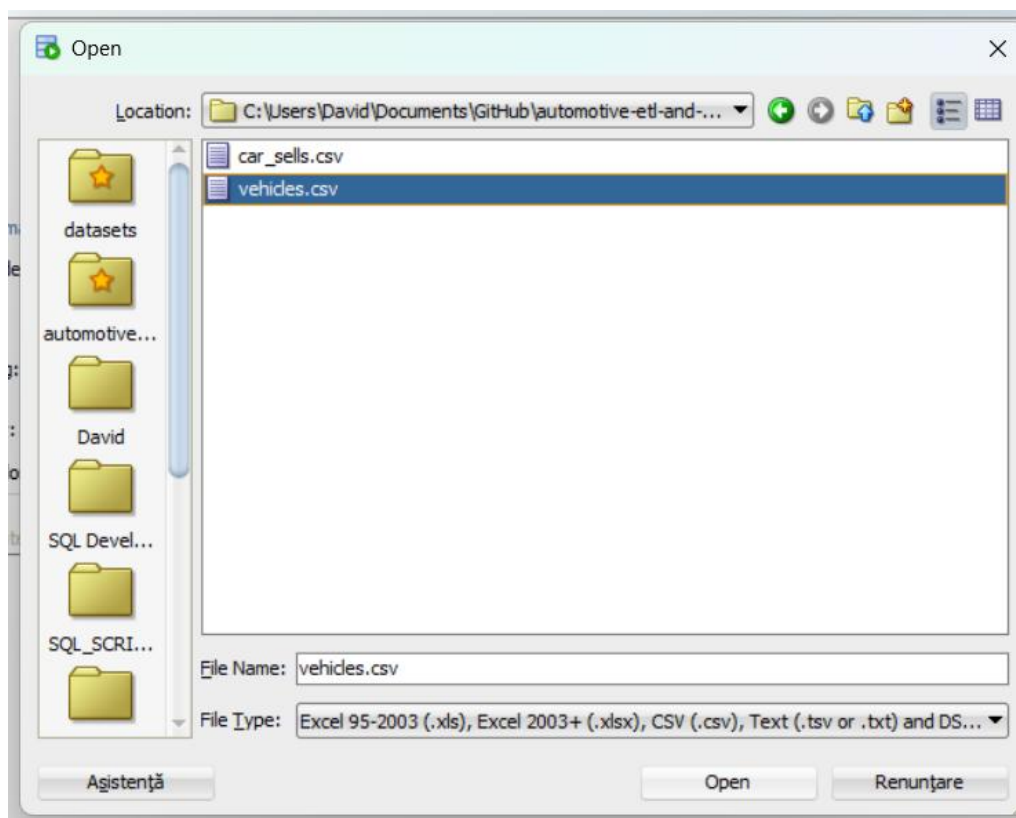
2. Importarea fișierelor CSV în tabelele brute, folosind instrumente din Oracle SQL

Developer și urmând pașii:

- a. Selectarea tabelului și acțiunii de import data.



- b. Selectarea fișierului sursă dorit.



- c. Selectarea modalității de inserare a datelor, în cazul acestei utilizare a comenzii de insert

Import Method: Insert

☐ Send Create Script to SQL Worksheet

Table Name: RAW\_VEHICLE\_DATA

- d. Maparea coloanelor din fișierul sursă cu denumirea coloanelor din tabela brută (raw).

For each column in the Source Data Columns list on the left, select a Target Table column on the right.

Match By: Name

Source Data Columns

- Version
- Brand
- Veh\_Model**
- Veh\_Category**
- Kg\_veh
- Test\_mass
- CO2\_wltp
- Wheelbase\_mm
- Axle\_width\_steer\_mm
- Axle\_width\_other\_mm
- Energy
- Engine\_cm3
- Power\_KW
- El\_Consumpt\_whkm
- year
- Fuel consumption
- Electric range (km)
- Eco-innovation program**
- Em\_on\_target

Target Table Columns

Name	Data Type	Size/Precision	Scale	Nullable?	Comment
V_ID	ENGINE_CM3				
	POWER_KW				
	EL_CONSUMPT_WHKM				
	YEAR				
	FUEL_CONSUMPTION				
	ELECTRIC_RANGE_KM				
	ECO_INNOVATION_PROGRAM				
	EM_ON_TARGET				

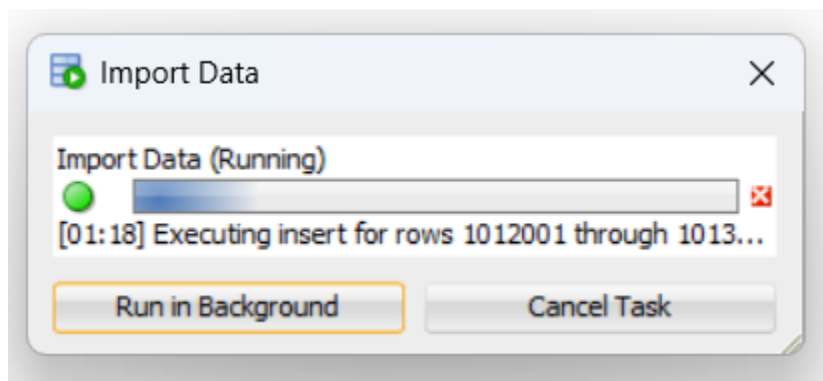
Data

0
0
0
0
0
0

- e. Finalizarea procesului

Import Summary

- Destination Connection: BDSA
- Source File: C:\Users\David\Documents\GitHub\automotive-etl-and-analysis\datasets\vehides.csv
- Selected Fields
- Fields Not Selected
- Import Method: Insert



Observație: Pentru al doilea fișier urmăm aceeași pași.


### 3. Realizarea schemei normalizate

```
CREATE TABLE vehicle_companies (  
    company_id NUMBER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,  
    constructor_name VARCHAR2(100),  
    brand VARCHAR2(100)  
);  
  
CREATE TABLE vehicle_models (  
    model_id NUMBER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,  
    company_id NUMBER,  
    model_name VARCHAR2(100),  
    vehicle_type VARCHAR2(50),  
    category VARCHAR2(10),  
    FOREIGN KEY (company_id) REFERENCES vehicle_companies(company_id)  
);  
  
CREATE TABLE vehicle_versions (  
    version_id NUMBER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,  
    model_id NUMBER,  
    version_code VARCHAR2(50),  
    kg_veh NUMBER,  
    test_mass NUMBER,  
    co2_wltp NUMBER,  
    wheelbase_mm NUMBER,  
    axle_width_steer_mm NUMBER,  
    axle_width_other_mm NUMBER,  
    energy VARCHAR2(50),  
    engine_cm3 NUMBER,  
    power_kw NUMBER,  
    el_consumpt_whkm NUMBER,
```

```
);

CREATE TABLE vehicle_sales (
    sale_id NUMBER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
    model_id NUMBER,
    car_size VARCHAR2(50),
    transmission VARCHAR2(20),
    car_state VARCHAR2(10),
    car_condition VARCHAR2(10),
    odometer NUMBER,
    color VARCHAR2(30),
    interior VARCHAR2(30),
    seller VARCHAR2(100),
    mmr NUMBER,
    selling_price NUMBER,
    sale_day VARCHAR2(10),
    sale_month VARCHAR2(10),
    sale_year NUMBER,
    FOREIGN KEY (model_id) REFERENCES vehicle_models(model_id)
);
```

```

setup >  create_normalized_schema.sql > ...
37 CREATE TABLE vehicle_sales (
54
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```

Table VEHICLE\_COMPANIES created.

Table VEHICLE\_MODELS created.

Table VEHICLE\_VERSIONS created.

Table VEHICLE\_SALES created.



Modelul propus reflectă relațiile dintre companii, modele, versiuni și vânzări de vehicule, prin următoarele tabele principale:

- **vehicle\_companies** – stochează informații despre constructori și mărcile auto.
- **vehicle\_models** – conține detalii despre modelele de vehicule și tipurile acestora.
- **vehicle\_versions** – reține specificațiile tehnice aferente fiecărei versiuni.
- **vehicle\_sales** – include informații comerciale referitoare la tranzacțiile de vânzare.

Relațiile dintre entitățile principale sunt de tip **unu-la-mai-mulți (1-N)**:

- o companie poate produce mai multe modele;
- fiecare model poate avea mai multe versiuni;
- fiecare versiune poate apărea în multiple tranzacții de vânzare.

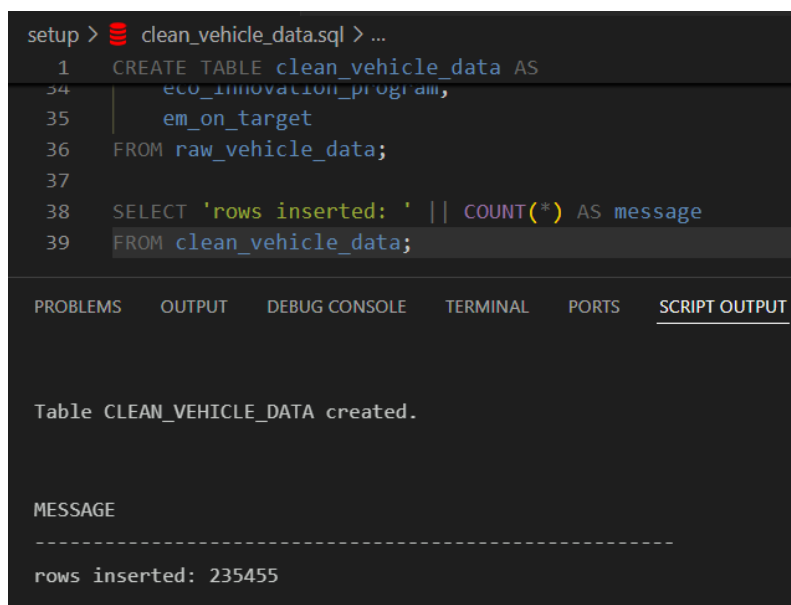
Această structură asigură integritatea referențială și permite realizarea de analize complexe asupra datelor tehnice și comerciale.

4. **Pregătirea tabeli de staging** – pentru datele provenite din fișierul *vehicles.csv* a fost creată o tabelă intermediară (*clean\_vehicle\_data*) în care informațiile denormalizate au fost curățate, standardizate și deduplicate. Această tabelă servește drept sursă pentru încărcarea ulterioară a datelor în schema relațională.

```
CREATE TABLE clean_vehicle_data AS
SELECT DISTINCT
    UPPER(TRIM(
        REGEXP_REPLACE(
            REGEXP_REPLACE(constructor_name, ',', ''),
            '\s+', ' '
        )) AS constructor_name,
    vehicle_type,
    version,
    UPPER(TRIM(
        REGEXP_REPLACE(
            REGEXP_REPLACE(brand, ',', ''),
            '\s+', ' '
        )) AS brand,
    UPPER(TRIM(
        REGEXP_REPLACE(
            REGEXP_REPLACE(vehicle_model, ',', ''),
```

```
        '\s+', ' ' ))
    ) AS vehicle_model,
    vehicle_category,
    kg_veh,
    test_mass,
    co2_wltp,
    wheelbase_mm,
    axle_width_steer_mm,
    axle_width_other_mm,
    energy,
    engine_cm3,
    power_kw,
    el_consumpt_whkm,
    year,
    fuel_consumption,
    electric_range_km,
    eco_innovation_program,
    em_on_target
FROM raw_vehicle_data;

SELECT 'rows inserted: ' || COUNT(*) AS message
FROM clean_vehicle_data;
```



The screenshot shows a SQL IDE interface with a script editor and a results pane. The script editor contains the following SQL code:

```
setup > clean_vehicle_data.sql > ...
1 CREATE TABLE clean_vehicle_data AS
34 |   eco_innovation_program,
35 |   em_on_target
36 FROM raw_vehicle_data;
37
38 SELECT 'rows inserted: ' || COUNT(*) AS message
39 FROM clean_vehicle_data;
```

The results pane shows the output of the script:

```
Table CLEAN_VEHICLE_DATA created.

MESSAGE
-----
rows inserted: 235455
```

Observație: De la un număr de intrări de peste 1 milion, tabela de staging a rămas după deduplicarea datelor cu doar 235 455 de intrări, evidențând necesitatea curățării fișierelor de date înainte de utilizare.

5. Popularea tabelelor normalizate a fost realizată prin mai mulți pași, adaptați structurii fiecărei surse de date:

- Datele din **clean\_vehicle\_data** (prima sursă) au fost utilizate pentru completarea tabelelor *vehicle\_companies*, *vehicle\_models* și *vehicle\_versions*. Pentru fiecare tabelă, inserările s-au realizat astfel încât să se evite duplicatele și să se păstreze consistența referințială între entități. Maparea se face prin potrivirea numelor standardizate ale constructorilor, mărcilor și modelelor.

```
INSERT INTO vehicle_companies (constructor_name, brand)
SELECT DISTINCT UPPER(TRIM(constructor_name)), UPPER(TRIM(brand))
FROM clean_vehicle_data;

INSERT INTO vehicle_models (company_id, model_name, vehicle_type,
category)
SELECT DISTINCT
    c.company_id,
    UPPER(TRIM(cl.vehicle_model)) AS model_name,
    cl.vehicle_type,
    cl.vehicle_category
FROM clean_vehicle_data cl
JOIN vehicle_companies c
    ON UPPER(TRIM(c.constructor_name)) = UPPER(TRIM(cl.constructor_name))
    AND UPPER(TRIM(c.brand)) = UPPER(TRIM(cl.brand));

INSERT INTO vehicle_versions (
    model_id, version_code, kg_veh, test_mass, co2_wltp,
    wheelbase_mm, axle_width_steer_mm, axle_width_other_mm,
    energy, engine_cm3, power_kw, el_consumpt_whkm,
    fuel_consumption, electric_range_km,
    eco_innovation_program, em_on_target
)
SELECT
    m.model_id, cl.version, cl.kg_veh, cl.test_mass, cl.co2_wltp,
    cl.wheelbase_mm, cl.axle_width_steer_mm, cl.axle_width_other_mm,
    cl.energy, cl.engine_cm3, cl.power_kw, cl.el_consumpt_whkm,
    cl.fuel_consumption, cl.electric_range_km,
    cl.eco_innovation_program, cl.em_on_target
FROM clean_vehicle_data cl
JOIN vehicle_companies c
    ON UPPER(TRIM(c.constructor_name)) = UPPER(TRIM(cl.constructor_name))
    AND UPPER(TRIM(c.brand)) = UPPER(TRIM(cl.brand))
```

```
JOIN vehicle_models m
  ON m.company_id = c.company_id
  AND UPPER(TRIM(m.model_name)) = UPPER(TRIM(c1.vehicle_model))
  AND UPPER(TRIM(m.vehicle_type)) = UPPER(TRIM(c1.vehicle_type))
  AND UPPER(TRIM(m.category)) = UPPER(TRIM(c1.vehicle_category));
```

54 rows inserted.

1,219 rows inserted.

235,900 rows inserted.

- Datele din **raw\_vehicle\_sales** (a doua sursă), deja parțial normalizate, au fost standardizate și mapate utilizând tabela existentă *vehicle\_models*, fiind apoi inserate în tabela *vehicle\_sales*. Nu s-a abandonat câmpul *company* și nu s-a legat de către tabela *vehicle\_company* deoarece legătura era realizată deja prin intermediul tabeli intermedierea *vehicle\_models*.

```
INSERT INTO vehicle_sales (
  model_id, car_size, transmission, car_state,
  car_condition, odometer, color, interior, seller,
  mmr, selling_price, sale_day, sale_month, sale_year
)
SELECT DISTINCT
  vm.model_id AS model_id,
  vs.car_size,
  vs.transmission,
  vs.car_state,
  vs.car_condition,
  vs.odometer,
  vs.color,
  vs.interior,
  vs.seller,
  vs.mmr,
  vs.selling_price,
  vs.sale_day,
```

```
vs.sale_month,  
vs.sale_year  
FROM raw_vehicle_sales vs  
JOIN vehicle_models vm  
  ON UPPER(TRIM(REGEXP_REPLACE(REGEXP_REPLACE(vs.model, ',', ''), '\s+',  
' '))) =  
      UPPER(TRIM(REGEXP_REPLACE(REGEXP_REPLACE(vm.model_name, ',', ''),  
'\s+', ' ')));
```

105,402 rows inserted.

Observație: De la un număr de intrări de peste 500 000, tabela finală a rămas după curățarea datelor cu doar o cincime din numărul initial de intrări, evidențând încă o data necesitatea curățării datelor înainte de utilizare.

Prin aceste etape, toate tabelele finale (vehicle\_companies, vehicle\_models, vehicle\_versions, vehicle\_sales) au fost populate cu date curate, standardizate și coerente, pregătite pentru interogări analitice și raportare. În acest proces s-au aplicat transformări pentru uniformizarea textului (majuscule, eliminarea spațiilor multiple și a caracterelor inutile) și asigurarea corelării corecte cu cheile primare ale tabelelor normalizate.

6. Opțional, tabelele pot fi șterse și recreate prin rularea scriptului drop\_tables.sql.

## 5. Analize și interogări SQL

După încărcarea completă a datelor, au fost dezvoltate o serie de interogări analitice destinate obținerii de informații utile privind performanțele vehiculelor și tendințele pieței.

- **Calculul mediei și deviației standard a autonomiei vehiculelor electrice per marcă** – avg\_and\_std\_km\_range.sql.

```
SELECT DISTINCT  
  C.brand, M.model_name,  
  ROUND(AVG(V.electric_range_km) OVER (PARTITION BY M.model_name), 2) AS  
Autonomie_medie_Km_Model,
```

```
ROUND(STDDEV(V.electric_range_km) OVER (PARTITION BY M.model_name), 2)
AS deviatia_standard_autonomie_Km_Model,
COUNT(V.version_id) OVER (PARTITION BY M.model_name) AS
nr_versiuni_model
FROM vehicle_versions V
JOIN vehicle_models M ON V.model_id = M.model_id
JOIN vehicle_companies C ON M.company_id = C.company_id
WHERE V.electric_range_km > 0 AND ENERGY = 'electric'
ORDER BY Autonomie_medie_Km_Model DESC;
```

All rows fetched: 74 in 0.246 seconds

	BRAND	MODEL_NAME	AUTONOMIE_MEDIE_KM_MODEL	DEVIATIA_STANDARD_AUTONOMIE_KM_MODEL	NR_VERSIUNI_MODEL
1	MERCEDES-BENZ	EQS 450+	715.18	12.12	39
2	MERCEDES-BENZ	EQS 580 4MATIC	644.12	15.07	26
3	PORSCHE	TAYCAN GTS	481	4.28	8
4	KIA	NIRO	446.22	0	1
5	OPEL	AMPERA-E	440.59	38.93	6
6	MG ROEWE	MG ZS EVROEWE ZS EV	440	0	1
7	PORSCHE	TAYCAN 4	437	0	167
8	HYUNDAI	IONIQ5	432.56	33.9	12
9	MERCEDES-BENZ	EQC 400 4MATIC	422.03	30.79	2542
10	MERCEDES-BENZ	EQA 300 4MATIC	421.75	4.58	119
11	PORSCHE	TAYCAN TURBO	420.79	1.83	435
12	MERCEDES-BENZ	EQA 350 4MATIC	420.13	4.65	75
13	MERCEDES-BENZ	EQA 250	415.5	9.75	1327

- **Calculul mediei și deviației standard a emisiilor de CO2 per marcă –**  
avg\_and\_std\_co2.sql.

```
SELECT C.brand, M.model_name, M.category,
ROUND(AVG(V.co2_wltp), 2) AS AVG_CO2_Model,
ROUND(STDDEV(V.co2_wltp), 2) AS STDDEV_CO2_Model
FROM vehicle_versions V
JOIN vehicle_models M ON V.model_id = M.model_id
JOIN vehicle_companies C ON M.company_id = C.company_id
WHERE V.co2_wltp IS NOT NULL AND V.co2_wltp > 0
GROUP BY C.brand, M.model_name, M.category
HAVING COUNT(V.version_id) >= 2
ORDER BY STDDEV_CO2_Model DESC;
```

Fetches 200 rows in 1.479 seconds

	BRAND	MODEL_NAME	CATEGORY	AVG_CO2_MODEL	STDDEV_CO2_MODEL
1	mitsubishi	MITSUBISHI OUTLANDER	M1	139.54	70.49
2	AUDI	A8	M1	162.46	68.76
3	CITROEN	C5 AIRCROSS	M1	88.04	64.87
4	LAND ROVER	RANGE ROVER EVOQUE	M1G	155	62.05
5	SKODA	OCTAVIA RS	M1	87.49	60.08
6	VOLVO	XC90	M1G	109.03	58.66
7	VOLVO	XC60	M1G	114.37	57.71
8	PORSCHE	PANAMERA TUBRO	M1	323.5	57.28
9	VOLVO	XC40	M1	101.15	56.08
10	VOLVO	V60	M1	100.63	55.9
11	PEUGEOT	508	M1	76.99	55.42
12	FORD	KUGA	M1	106.54	55.4
13	VOLVO	S90	M1	98.1	54.97

- **Determinarea celor mai populare combinații de culori per marcă –**  
best\_selling\_color\_combo\_per\_brand.sql.

```

SELECT  BRAND,
        "BEST INTERIOR-EXTERIOR COMBO",
        total_revenue,
        total_sales,
        avg_market_value,
        avg_selling_price
FROM (
    SELECT
        c.brand AS BRAND,
        s.color || '-' || s.interior AS "BEST INTERIOR-EXTERIOR COMBO",
        COUNT(*) AS total_sales,
        ROUND(AVG(s.mmr), 2) AS avg_market_value,
        ROUND(AVG(s.selling_price), 2) AS avg_selling_price,
        ROUND(SUM(s.selling_price), 2) AS total_revenue,
        RANK() OVER (
            PARTITION BY c.brand
            ORDER BY SUM(s.selling_price) DESC
        ) AS total_revenue_rank_within_brand
    FROM vehicle_sales s
    JOIN vehicle_models m ON s.model_id = m.model_id
    JOIN vehicle_companies c ON m.company_id = c.company_id
    WHERE s.color IS NOT NULL AND s.interior IS NOT NULL
    GROUP BY c.brand, s.color, s.interior
)
WHERE total_revenue_rank_within_brand = 1
ORDER BY total_revenue;

```

All rows fetched: 19 in 0.627 seconds

	BRAND	BEST INTERIOR-EXTERIOR COMBO	TOTAL_REVENUE	TOTAL_SALES	AVG_MARKET_VALUE	AVG_SELLING_PRICE
1	SUZUKI	silver-gray	4700	2	1625	2350
2	OPEL	silver-black	47500	11	4254.55	4318.18
3	PEUGEOT	silver-gray	431350	198	2227.27	2178.54
4	BMW I	white-brown	446500	3	155333.33	148833.33
5	FIAT	white-black	631500	65	9734.62	9715.38
6	SUBARU	blue-black	5995375	337	17809.5	17790.43
7	VOLVO	black-black	7813850	407	19529.18	19198.65
8	LAND ROVER	black-black	7993050	224	35834.71	35683.26
9	BMW	white-black	9671054	592	16360.22	16336.24
10	MERCEDES-BENZ	white-black	9696600	384	23892.19	25251.56
11	MAZDA	gray-black	9729256	970	10158.58	10030.16
12	KIA	silver-black	9864300	747	13337.88	13205.22
13	HYUNDAI	silver-gray	10407675	1088	9870.2	9648.6

- Lista modelelor auto cu cele mai mari vânzări pentru fiecare brand –  
best\_selling\_models.sql.

```
SELECT DISTINCT brand, model, total_sales
FROM (
    SELECT
        c.brand as brand,
        m.model_name as model,
        SUM(s.selling_price) AS total_sales,
        RANK() OVER (PARTITION BY c.company_id ORDER BY
SUM(s.selling_price) DESC) AS rang
    FROM vehicle_sales s
    JOIN vehicle_models m ON s.model_id = m.model_id
    JOIN vehicle_companies c ON m.company_id = c.company_id
    GROUP BY c.brand, c.company_id, m.model_name
)
WHERE rang = 1
ORDER BY total_sales DESC;
```

All rows fetched: 21 in 0.082 seconds

	BRAND	MODEL	TOTAL_SALES
1	JEEP	WRANGLER	155749716
2	FORD	EXPLORER	114229068
3	JEEP	WRANGLER	77874858
4	HYUNDAI	ELANTRA	61347125
5	MINI	COOPER	49651425
6	KIA	SORENTO	48551273
7	BMW	COOPER	33100950
8	MINI	COOPER	33100950
9	PORSCHE	CAYENNE	31256600
10	VOLKSWAGEN VW	PASSAT	27632758
11	AUDI	Q5	27209600
12	LAND ROVER	RANGE ROVER SPORT	26287250
13	MAZDA	MAZDA3	25748417



- **Identificarea vehiculelor electrice cu autonomie peste media generală –**  
electric\_cars\_above\_avg\_range.sql.

```
WITH electric_avg AS (  
    SELECT AVG(electric_range_km) AS avg_range  
    FROM vehicle_versions  
    WHERE LOWER(energy) LIKE '%electric%'  
)  
SELECT  
    c.constructor_name,  
    m.model_name,  
    v.version_code,  
    MAX(v.electric_range_km) AS electric_range_km  
FROM vehicle_versions v  
JOIN vehicle_models m ON v.model_id = m.model_id  
JOIN vehicle_companies c ON m.company_id = c.company_id  
WHERE LOWER(v.energy) LIKE '%electric%'  
GROUP BY c.constructor_name, m.model_name, v.version_code  
HAVING MAX(v.electric_range_km) > (SELECT avg_range FROM electric_avg)  
ORDER BY electric_range_km DESC;
```

Fetches 200 rows in 0.320 seconds

	CONSTRUCTOR_NAME	MODEL_NAME	VERSION_CODE	ELECTRIC_RANGE_KM
1	MERCEDES-BENZ	EQS 450+	ZZAL051B	739
2	MERCEDES-BENZ	EQS 450+	ZZAL050C	728
3	MERCEDES-BENZ	EQS 450+	ZZAA050C	716
4	MERCEDES-BENZ	EQS 580 4MATIC	ZZAL050B	660
5	OPEL	AMPERA-E	A1EA5AA1	520
6	MERCEDES-BENZ	EQA 250	ZZAAA50A	486
7	VM-SAIC	TAYCAN GTS	1151000	485
8	VM-SAIC	TAYCAN GTS	1141000	485
9	HYUNDAI	KONA KAUAI	E11D11	484
10	HYUNDAI	KONA KAUAI	E11B11	484
11	HYUNDAI	IONIQ5	E11A11	481
12	MERCEDES-BENZ	EQC 400 4MATIC	ZZAA050A	471
13	MERCEDES-BENZ	EQC 400 4MATIC	ZZAA050A	471

- **Distribuția procentuală a tipurilor de energie pentru fiecare constructor**  
(electric, hibrid, combustibil) per marcă – energy\_percentange\_per\_brand.sql.

```
SELECT  
    c.brand,  
    v.energy as CONSUMPTION_TYPE,  
    ROUND(  

```

```

COUNT(DISTINCT m.model_name) * 100.0
/ SUM(COUNT(DISTINCT m.model_name)) OVER (PARTITION BY c.brand),
2
) || '%' AS BRAND_MODELS_RATIO,
COUNT(DISTINCT m.model_name) AS nr_modele
FROM vehicle_versions v
JOIN vehicle_models m ON v.model_id = m.model_id
JOIN vehicle_companies c ON m.company_id = c.company_id
GROUP BY c.brand, v.energy
ORDER BY c.brand, v.energy;

```

Fetches 109 rows in 0.205 seconds

	BRAND	CONSUMPTION_TYPE	BRAND_MODELS_RATIO	NR_MODELE
1	ALFA ROMEO	diesel	30%	3
2	ALFA ROMEO	petrol	70%	7
3	AUDI	diesel	28,7%	31
4	AUDI	electric	6,48%	7
5	AUDI	hybrid petrol	20,37%	22
6	AUDI	petrol	44,44%	48
7	BMW	diesel	39,79%	76
8	BMW	hybrid petrol	10,47%	20
9	BMW	petrol	49,74%	95
10	BMW I	electric	66,67%	2
11	BMW I	hybrid petrol	33,33%	1
12	CITROEN	diesel	38,1%	8

- Afișarea ierarhică a vehiculelor după marcă și model –  
hierarchical\_cars\_display.sql.

```

SELECT
    LPAD(' ', LEVEL * 5) || name AS hierarchy_element, LEVEL, entity_type
FROM (
    SELECT
        'C' || company_id AS id,
        NULL AS parent_id,
        brand AS name,
        'Company' AS entity_type
    FROM vehicle_companies
    UNION ALL
    SELECT
        'M' || model_id AS id,
        'C' || company_id AS parent_id,
        model_name AS name,
        'Model' AS entity_type
    FROM vehicle_models
    UNION ALL

```

```

SELECT
    'V' || MAX(version_id) AS id,
    'M' || MAX(model_id) AS parent_id,
    version_code AS name,
    'Version' AS entity_type
FROM vehicle_versions
GROUP BY version_code
)
START WITH parent_id IS NULL
CONNECT BY PRIOR id = parent_id
ORDER SIBLINGS BY name;

```

Fetches 200 rows in 0.201 seconds

	HIERARCHY_ELEMENT	LEVEL	ENTITY_TYPE
1	ALFA ROMEO	1	Company
2	ALFA GIULIETTA	2	Model
3	18AR	3	Version
4	18AS	3	Version
5	E29D5	3	Version
6	E31D5	3	Version
7	GIULIA	2	Model
8	5A1DG	3	Version
9	5A1DGD	3	Version
10	5ADG	3	Version
11	5ADGD	3	Version
12	5CDG	3	Version
13	5CDGD	3	Version

- Afișarea într-o structură ierarhică a top 5 vânzări pe grup de date calendaristice – hierarchical\_sales\_display\_by\_date.sql.

```

SELECT
    LPAD(' ', LEVEL * 5) || name AS hierarchy_element, entity_type,
    selling_price
FROM (
    SELECT
        'Y-' || sale_year AS id,
        NULL AS parent_id,
        TO_CHAR(sale_year) AS name,
        'Year' AS entity_type,
        NULL AS selling_price
    FROM vehicle_sales
    GROUP BY sale_year
    UNION ALL
    SELECT
        'YM-' || sale_year || '-' || sale_month AS id,
        'Y-' || sale_year AS parent_id,

```

```

        sale_month AS name,
        'Month' AS entity_type,
        NULL AS selling_price
    FROM vehicle_sales
    GROUP BY sale_year, sale_month
    UNION ALL
    SELECT
        'S-' || sale_id AS id,
        'YM-' || sale_year || '-' || sale_month AS parent_id,
        'Seller: ' || UPPER(seller) ||
        ' | Price: ' || selling_price ||
        ' | Car: ' || c.brand || ' ' || m.model_name AS name,
        'Sale' AS entity_type,
        selling_price
    FROM (
        SELECT vs.*,
               ROW_NUMBER() OVER (PARTITION BY sale_year, sale_month ORDER
    BY selling_price DESC) AS rn
        FROM vehicle_sales vs
    ) vs
    JOIN vehicle_models m ON vs.model_id = m.model_id
    JOIN vehicle_companies c ON m.company_id = c.company_id
    WHERE rn <= 5
)
START WITH parent_id IS NULL
CONNECT BY PRIOR id = parent_id
ORDER SIBLINGS BY selling_price DESC , ID ASC

```

	HIERARCHY_ELEMENT	ENTITY_TYPE	SELLING_PRICE
1	2014	Year	(null)
2	Dec	Month	(null)
3	Seller: THE COLLECTION   Price: 156000   Car: VOLKSWAGEN VW CALIFORNIA	Sale	156000
4	Seller: CHICAGO MOTOR CAR CORPORATION   Price: 154000   Car: VOLKSWAGEN VW CALI...	Sale	154000
5	Seller: EUROCAR   Price: 124000   Car: VOLKSWAGEN VW CALIFORNIA	Sale	124000
6	Seller: THE COLLECTION   Price: 84000   Car: LAND ROVER RANGE ROVER SPORT	Sale	84000
7	Seller: THE COLLECTION   Price: 79800   Car: AUDI S8	Sale	79800
8	Feb	Month	(null)
9	Seller: KIA MOTORS AMERICA INC   Price: 10500   Car: KIA RIO	Sale	10500
10	Jan	Month	(null)
11	Seller: ARS/ENTERPRISE   Price: 33800   Car: FORD EXPLORER	Sale	33800
12	Seller: ARS/ENTERPRISE   Price: 33500   Car: FORD EXPLORER	Sale	33500

- Analiza influenței kilometrajului asupra vânzărilor utilizând diverse măsuri – odometer\_sales\_influence.sql.

```
SELECT
```

```
CASE
    WHEN s.odometer BETWEEN 0 AND 10000 THEN '0-10k'
    WHEN s.odometer BETWEEN 10001 AND 20000 THEN '10k-20k'
    WHEN s.odometer BETWEEN 20001 AND 30000 THEN '20k-30k'
    WHEN s.odometer BETWEEN 30001 AND 50000 THEN '30k-50k'
    WHEN s.odometer BETWEEN 50001 AND 70000 THEN '50k-70k'
    WHEN s.odometer BETWEEN 70001 AND 100000 THEN '70k-100k'
    ELSE '100k+'
END AS km_ran,
ROUND(AVG(s.selling_price), 2) AS avg_price,
DENSE_RANK() OVER (ORDER BY ROUND(AVG(s.selling_price), 2) DESC) AS
avg_price_rank,
ROUND(AVG(s.mmr), 2) AS avg_estimated_market_value,
DENSE_RANK() OVER (ORDER BY ROUND(AVG(s.mmr), 2) DESC) AS
avg_estimated_market_value_rank,
ROUND(AVG(s.selling_price - s.mmr), 2) AS avg_price_dev_from_market,
DENSE_RANK() OVER (ORDER BY ROUND(AVG(s.selling_price - s.mmr), 2)
DESC) AS avg_price_dev_from_market_rank,
COUNT(*) AS sales_count,
DENSE_RANK() OVER (ORDER BY COUNT(*) DESC) AS sales_count_rank
FROM vehicle_sales s
WHERE s.odometer IS NOT NULL
GROUP BY
    CASE
        WHEN s.odometer BETWEEN 0 AND 10000 THEN '0-10k'
        WHEN s.odometer BETWEEN 10001 AND 20000 THEN '10k-20k'
        WHEN s.odometer BETWEEN 20001 AND 30000 THEN '20k-30k'
        WHEN s.odometer BETWEEN 30001 AND 50000 THEN '30k-50k'
        WHEN s.odometer BETWEEN 50001 AND 70000 THEN '50k-70k'
        WHEN s.odometer BETWEEN 70001 AND 100000 THEN '70k-100k'
        ELSE '100k+'
    END
ORDER BY avg_price_rank, avg_estimated_market_value_rank,
avg_price_dev_from_market_rank, sales_count_rank;
```

All rows fetched: 7 in 0.120 seconds

	KM_RAN	AVG_PRICE	AVG_PRICE_RANK	AVG_ESTIMATED_MARKET_VALUE	AVG_ESTIMATED_MARKET_VALUE_RANK
1	0-10k	22367.33	1	22552.46	1
2	10k-20k	20148.39	2	20354.62	2
3	20k-30k	18393.9	3	18552.48	3
4	30k-50k	16468.65	4	16579.96	4
5	50k-70k	13400.47	5	13448.05	5
6	70k-100k	9747.92	6	9901.47	6
7	100k+	4774.15	7	4958.97	7

	AVG_PRICE_DEV_FROM_MARKET	AVG_PRICE_DEV_FROM_MARKET_RANK	SALES_COUNT	SALES_COUNT_RANK
	-185.12	6	5586	7
	-206.22	7	13667	5
	-158.58	4	13271	6
	-111.31	2	24768	1
	-47.58	1	15545	3
	-153.55	3	14711	4
	-184.82	5	17849	2

- **Clasificarea mărcilor după proporția modelelor eco-friendly –**  
rank\_brands\_by\_green\_models\_distribution.sql.

```
WITH modele_firma AS (
    SELECT
        c.brand,
        COUNT(DISTINCT v.model_id) AS total_modele,
        COUNT(DISTINCT CASE WHEN v.eco_innovation_program = 1 OR
v.em_on_target = 1 THEN v.model_id END) AS modelele_verzi
    FROM vehicle_versions v
    JOIN vehicle_models m ON v.model_id = m.model_id
    JOIN vehicle_companies c ON m.company_id = c.company_id
    GROUP BY c.brand
)
SELECT
    brand,
    total_modele,
    modelele_verzi,
    ROUND(modelele_verzi * 100.0 / total_modele, 2) AS
procent_modele_verzi,
    DENSE_RANK() OVER (ORDER BY modelele_verzi * 100.0 / total_modele
DESC) AS rang_firma
FROM modele_firma ORDER BY rang_firma;
```

All rows fetched: 35 in 0.198 seconds

	BRAND	TOTAL_MODELE	MODELELE_VERZI	PROCENT_MODELE_VERZI	RANG_FIRMA
1	POLESTAR	1	1	100	1
2	BMW I	3	3	100	1
3	MG ROEWE	6	6	100	1
4	SMART	9	8	88.89	2
5	VOLVO	19	15	78.95	3
6	CUPRA	9	7	77.78	4
7	PEUGEOT	22	14	63.64	5
8	MITSUBISHI	8	5	62.5	6
9	CITROEN	15	8	53.33	7
10	JEEP	16	6	37.5	8
11	OPEL	35	13	37.14	9
12	AUDI	100	36	36	10
13	NISSAN	14	5	35.71	11

- **Calculul raportului dintre volumul vânzărilor și venituri per marcă sau model – sales\_and\_revenue\_ratio.sql.**

```
SELECT
    v.energy AS energy_type,
    COUNT(s.sale_id) AS total_sales,
    ROUND(COUNT(s.sale_id) * 100.0 / SUM(COUNT(s.sale_id)) OVER (), 2) AS
sales_ratio_pct,
    ROUND(AVG(s.selling_price), 2) AS avg_selling_price,
    ROUND(AVG(s.mmr), 2) AS avg_market_price,
    ROUND(SUM(s.selling_price), 2) AS total_revenue,
    ROUND(SUM(s.selling_price) * 100.0 / SUM(SUM(s.selling_price)) OVER
()), 2) AS revenue_ratio_pct
FROM vehicle_sales s
JOIN vehicle_models m ON s.model_id = m.model_id
JOIN vehicle_versions v ON v.model_id = m.model_id
GROUP BY v.energy
ORDER BY total_revenue DESC;
```

All rows fetched: 4 in 11.078 seconds

	ENERGY_TYPE	TOTAL_SALES	SALES_RATIO_PCT	AVG_SELLING_PRICE	AVG_MARKET_PRICE	TOTAL_REVENUE	REVENUE_RATIO_PCT
1	diesel	25364234	53.86	11873.02	12028.08	301150172711	56.56
2	petrol	14673638	31.16	10465.69	10639.3	153569815119	28.84
3	hybrid petrol	6929622	14.71	11054.97	11197.73	76606755277	14.39
4	electric	124781	0.26	8916.05	9078.08	1112553233	0.21

- **Analiza relației dintre greutatea vehiculului și prețul de vânzare, inclusiv prețul mediu și prețul per kilogram – vehicle\_weight\_price\_analysis.sql.**

```
SELECT * FROM (
  SELECT
    c.brand,
    m.model_name,
    ROUND(AVG(v.fuel_consumption), 2) AS avg_fuel_consumption
  FROM vehicle_versions v
  JOIN vehicle_models m ON v.model_id = m.model_id
  JOIN vehicle_companies c ON m.company_id = c.company_id
  WHERE v.fuel_consumption IS NOT NULL AND v.fuel_consumption > 0
  GROUP BY c.brand, m.model_name
  ORDER BY avg_fuel_consumption DESC
) WHERE ROWNUM <= 10;
```

All rows fetched: 10 in 0.189 seconds

	BRAND	MODEL_NAME	AVG_FUEL_CONSUMPTION
1	JEEP	GRAND CHEROKEE TRACKHAWK	16.7
2	MERCEDES-BENZ	AMG S 65	15.2
3	JEEP	GRAND CHEROKEE SRT	14.9
4	MERCEDES-BENZ	G 500	14.85
5	MERCEDES-BENZ	AMG G 63	14.79
6	FIAT	640 SB	14
7	MERCEDES-BENZ	S 680 MAYBACH	13.82
8	BMW	M760LI XDRIVE	13.4
9	PORSCHE	CAYENNE GTS	13.38

- **Lista modelelor cu cel mai bun și cel mai slab consum de combustibil –**  
top\_worst\_models\_by\_fuel\_consumption.sql.

```
SELECT
  c.constructor_name,
  m.model_name,
  ROUND(AVG(v.kg_veh), 2) AS avg_weight,
  ROUND(AVG(s.selling_price), 2) AS avg_price,
  ROUND(AVG(s.selling_price)/NULLIF(AVG(v.kg_veh),0), 2) AS price_per_kg
FROM vehicle_sales s
JOIN vehicle_models m ON s.model_id = m.model_id
JOIN vehicle_versions v ON v.model_id = m.model_id
JOIN vehicle_companies c ON m.company_id = c.company_id
GROUP BY c.constructor_name, m.model_name
ORDER BY price_per_kg DESC;
```



All rows fetched: 63 in 14.472 seconds

	CONSTRUCTOR_NAME	MODEL_NAME	AVG_WEIGHT	AVG_PRICE	PRICE_PER_KG
1	BMW	I8	1660	154222.22	92.9
2	VW-SAIC	CALIFORNIA	2516	134363.64	53.4
3	BMW	M4	1687.7	67943.75	40.26
4	VW-SAIC	MACAN	1870	60881.25	32.56
5	VW-SAIC	PANAMERA	1986	55780.3	28.09
6	STELLANTIS	GRAND CHEROKEE SRT	2418	61000	25.23
7	VW-SAIC	SQ5	2083.28	47660	22.88
8	BMW	M5	1962.4	39743.06	20.25
9	VW-SAIC	S8	2305	44476.32	19.3
10	BMW	X5 M	2305	44000.35	19.09

- Creșterea anuală a vânzărilor per marcă – yearly\_growth\_per\_brand.sql.

```
WITH brand_revenue AS (
    SELECT c.brand, s.sale_year, SUM(s.selling_price) AS total_revenue
    FROM vehicle_sales s
    JOIN vehicle_models m ON s.model_id = m.model_id
    JOIN vehicle_companies c ON m.company_id = c.company_id
    GROUP BY c.brand, s.sale_year
)
SELECT
    brand,
    sale_year,
    total_revenue,
    LAG(total_revenue) OVER (PARTITION BY brand ORDER BY sale_year) AS
prev_year_revenue
FROM brand_revenue;
```

	BRAND	SALE_YEAR	TOTAL_REVENUE	PREV_YEAR_REVENUE
1	AUDI	2014	4746550	(null)
2	AUDI	2015	54324950	4746550
3	BMW	2014	5716106	(null)
4	BMW	2015	52599854	5716106
5	BMW I	2015	1388000	(null)
6	FIAT	2014	236400	(null)
7	FIAT	2015	4589677	236400
8	FORD	2014	24611208	(null)
9	FORD	2015	383896651	24611208
10	HYUNDAI	2014	4746000	(null)
11	HYUNDAI	2015	56900150	4746000

## **6. Concluzie**

Proiectul „Proces ELT și Analiză a Datelor din Domeniul Auto” demonstrează aplicarea practică a conceptelor evolute de proiectare, implementare și analiză a bazelor de date. Prin integrarea mai multor surse de date și prin normalizarea informațiilor, s-a obținut o structură coerentă, capabilă să susțină analize complexe și vizualizări ierarhice asupra domeniului auto. Rezultatele obținute pot fi utilizate în cercetări privind eficiența energetică, sustenabilitatea și tendințele pieței auto moderne.