****

**MINISTERUL EDUCAȚIEI, CULTURII ȘI CERCETĂRII**

**AL REPUBLICII MOLDOVA**

**Universitatea Tehnică a Moldovei**

**Facultatea Calculatoare, Informatică şi Microelectronică**

**Departamentul Informatică şi Ingineria Sistemelor**

**Raport**

**pentru lucrarea de laborator Nr.3**

***la cursul de “Baze de Date”***

Efectuat:

**Chistol Maxim**, st. gr. IA-231

Verificat:

**Ganea Ion,**

Facultatea FCIM, UTM

**Chișinău – 2024**

# **Lucrare de laborator Nr. 3: COMPUNERI SQL**

**Scop:**

* Îmbunătățirea abilităților de combinare a datelor din tabele multiple prin utilizarea diferitelor tipuri de compuneri (JOIN) în interogări SQL.

**Obiective:**

Înțelegerea diferențelor dintre tipurile de compuneri:  *JOIN, LEFT JOIN, RIGHT JOIN, SELF JOIN*.

* Aplicarea compunerilor pentru corelarea datelor din tabele multiple.
* Crearea interogărilor complexe utilizând compuneri pentru extragerea datelor relevante.

**Cerințe:**

* Scrierea de interogări utilizând JOIN pentru extragerea datelor corelate din două sau mai multe tabele.
* Utilizarea LEFT JOIN și RIGHT JOIN pentru a include date din tabelele de bază, chiar și în lipsa unor corelări.
* Realizarea de interogări SELF JOIN pentru corelarea datelor din același tabel, dacă este necesar (ex: ierarhia angajaților).
* Testarea interogărilor cu date reale pentru validarea rezultatelor obținute.

# **Sarcini:**

1. ***Comenzi și clienți cu produse cumpărate.***

SELECT o.order\_id, c.first\_name, c.last\_name, p.product\_name, oi.quantity

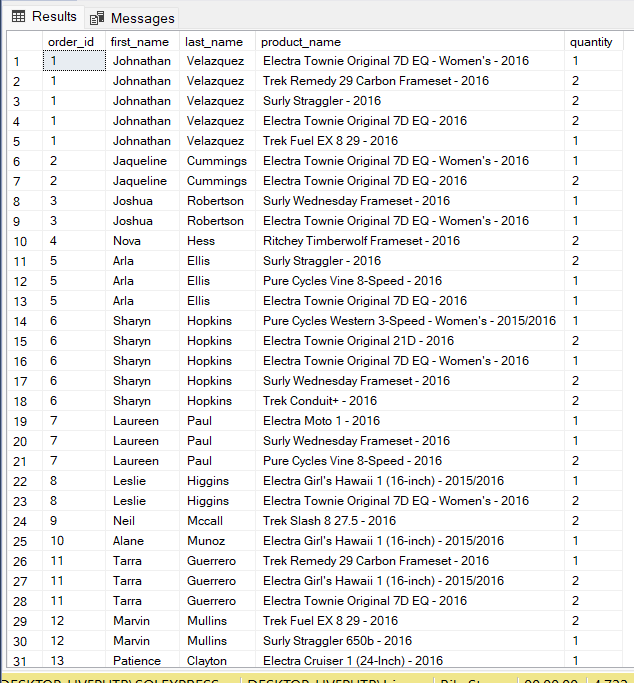
FROM sales.orders o

JOIN sales.customers c ON o.customer\_id = c.customer\_id

JOIN sales.order\_items oi ON o.order\_id = oi.order\_id

JOIN production.products p ON oi.product\_id = p.product\_id;

* Combinăm tabelele sales.orders, sales.customers, sales.order\_items și production.products folosind INNER JOIN.
* **Afișează**: ID-ul comenzii (order\_id), Numele clientului (first\_name, last\_name), Produsul comandat (product\_name), Cantitatea produsului (quantity).
* Leagă comenzile de clienți (customer\_id), de articolele din comandă (order\_id) și de produsele comandate (product\_id), pentru a afișa detaliile relevante.

****

**Figura 1 –** Rezultatele interogării

1. ***Stocuri disponibile în magazine pentru fiecare categorie de produse.***

USE [BikeStores]

SELECT st.store\_id,

s.store\_name,

c.category\_name,

SUM(st.quantity) AS total\_stock

FROM production.stocks st

JOIN sales.stores s ON st.store\_id = s.store\_id

JOIN production.products p ON st.product\_id = p.product\_id

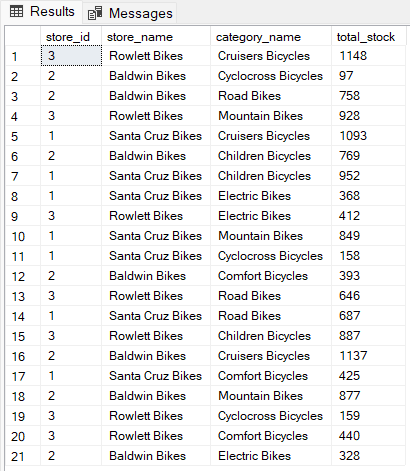
JOIN production.categories c ON p.category\_id = c.category\_id

GROUP BY st.store\_id, s.store\_name, c.category\_name;

GO

Se folosesc tabelele **production.stocks**, **sales.stores**, **production.products** și **production.categories**.

* **JOIN** între **stocks** și **stores** pe store\_id, între **stocks** și **products** pe product\_id, și între **products** și **categories** pe category\_id.
* Se selectează: **ID-ul magazinului**, **numele magazinului**, **numele categoriei**, și **totalul stocurilor** pentru fiecare combinație magazin-categorie.
* Rezultatele sunt grupate după **ID-ul magazinului**, **numele magazinului**, și **numele categoriei**, pentru a calcula suma stocurilor per combinație.

****

**Figura 2 –** Rezultatele interogării

1. ***Produse comandate, reduceri aplicate și stocuri disponibile.***

Pentru a determina **produsele comandate, reducerile aplicate și stocurile disponibile**, trebuie să corelăm tabelele care conțin informațiile despre comenzi, produse, reduceri și stocuri. Relațiile necesare sunt:

* **sales.order\_items**: Conține produsele comandate și reducerile aplicate.
* **production.products**: Conține informații despre produse.
* **production.stocks**: Conține stocurile disponibile pentru produse.

USE [BikeStores]

SELECT p.product\_name,

oi.quantity AS ordered\_quantity,

oi.discount AS applied\_discount,

COALESCE(SUM(st.quantity), 0) AS available\_stock

FROM sales.order\_items oi

JOIN production.products p ON oi.product\_id = p.product\_id

LEFT JOIN production.stocks st ON p.product\_id = st.product\_id

GROUP BY p.product\_name, oi.quantity, oi.discount;

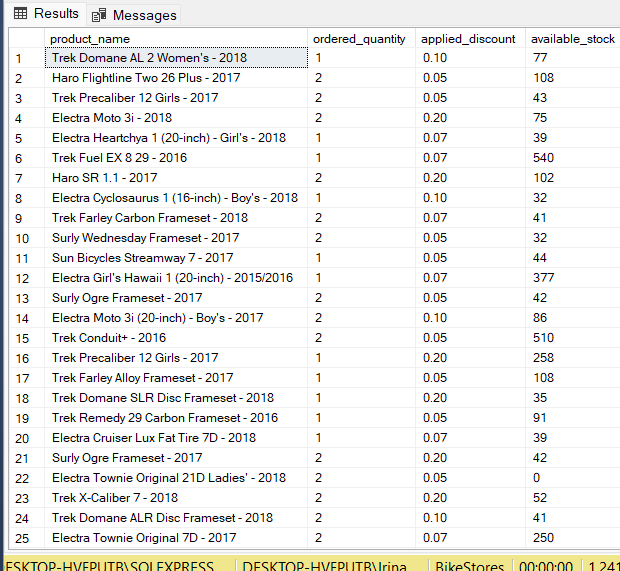
GO

**Coloane afișate:**

* product\_name: Numele produsului.
* ordered\_quantity: Cantitatea comandată pentru produs (din sales.order\_items).
* applied\_discount: Reducerea aplicată pe produs (din sales.order\_items).
* available\_stock: Stocul disponibil al produsului (calculat folosind SUM pe stocuri și COALESCE pentru a afișa 0 dacă nu există stoc).

**LEFT JOIN pentru stocuri:** Folosim **LEFT JOIN** între production.products și production.stocks pentru a include produsele care au fost comandate, chiar dacă nu au stocuri disponibile.

**Grupare:** Grupăm după product\_name, ordered\_quantity și applied\_discount, pentru a calcula totalul stocului per produs comandat.

****

**Figura 3 –** Rezultatele interogării

1. ***Angajați responsabili de comenzi și detalii ale clienților.***

Pentru a determina **angajații responsabili de comenzi și detaliile clienților**, trebuie să corelăm tabelele care conțin informațiile despre angajați, comenzi și clienți. Tabele implicate:

* **sales.orders**: Conține informații despre comenzi, inclusiv staff\_id, care identifică angajatul responsabil de comandă.
* **sales.staffs**: Conține informații despre angajați.
* **sales.customers**: Conține detalii ale clienților.

USE [BikeStores]

SELECT s.first\_name AS staff\_first\_name,

s.last\_name AS staff\_last\_name,

o.order\_id,

c.first\_name AS customer\_first\_name,

c.last\_name AS customer\_last\_name,

c.email AS customer\_email,

c.phone AS customer\_phone

FROM sales.orders o

JOIN sales.staffs s ON o.staff\_id = s.staff\_id

JOIN sales.customers c ON o.customer\_id = c.customer\_id;

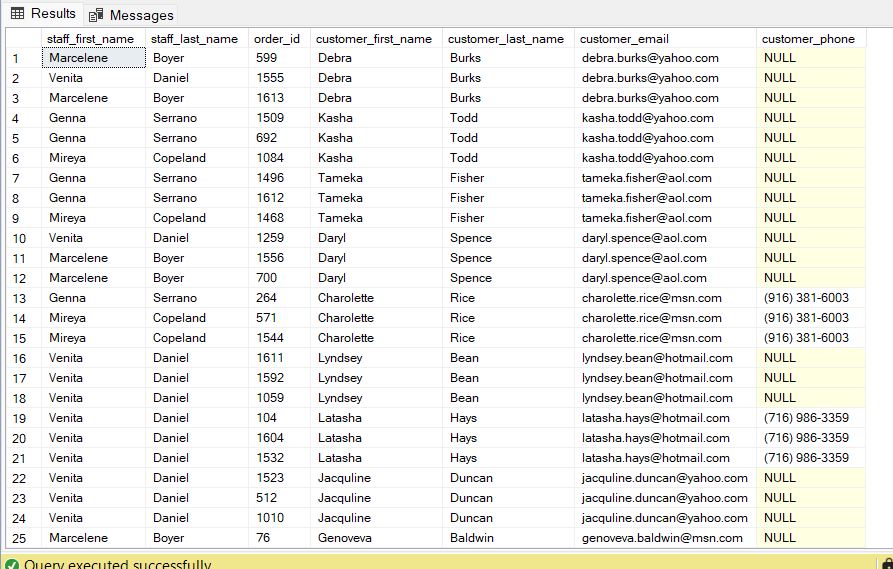
GO

**Coloane afișate:**

* **Numele angajatului responsabil**: staff\_first\_name și staff\_last\_name din tabelul sales.staffs.
* **ID-ul comenzii**: order\_id din tabelul sales.orders.
* **Detalii ale clientului**: customer\_first\_name, customer\_last\_name, customer\_email, și customer\_phone din tabelul sales.customers.

J**OIN-uri:**

* **Între sales.orders și sales.staffs**: Conectează comanda cu angajatul care a gestionat-o pe baza câmpului staff\_id.
* **Între sales.orders și sales.customers**: Conectează comanda cu clientul care a făcut-o pe baza câmpului customer\_id.

****

**Figura 4–** Rezultatele interogării

1. ***Produse comandate împreună cu magazinele unde sunt disponibile.***

Pentru a determina **produsele comandate împreună cu magazinele unde sunt disponibile**, trebuie să corelăm tabelele care conțin informațiile despre comenzi, produse și magazine. Tabele implicate:

* **sales.order\_items**: Conține informațiile despre produsele comandate.
* **production.products**: Conține informațiile despre produsele disponibile.
* **production.stocks**: Conține informațiile despre stocurile disponibile ale produselor în magazine.
* **sales.stores**: Conține informațiile despre magazine.

USE [BikeStores]

SELECT p.product\_name,

s.store\_name,

oi.quantity AS ordered\_quantity

FROM sales.order\_items oi

JOIN production.products p ON oi.product\_id = p.product\_id

JOIN production.stocks st ON p.product\_id = st.product\_id

JOIN sales.stores s ON st.store\_id = s.store\_id;

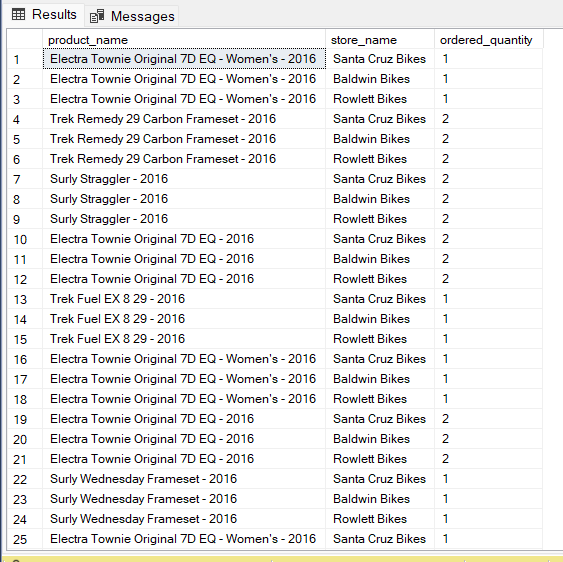
GO

**Coloane afișate:**

* **product\_name**: Numele produsului comandat (din tabelul **production.products**).
* **store\_name**: Numele magazinului unde este disponibil produsul (din tabelul **sales.stores**).
* **ordered\_quantity**: Cantitatea comandată a produsului (din tabelul **sales.order\_items**).

**JOIN-uri:**

* **Între sales.order\_items și production.products**: Leagă produsele comandate de detaliile produsului.
* **Între production.products și production.stocks**: Leagă produsele cu stocurile disponibile.
* **Între production.stocks și sales.stores**: Leagă stocurile de magazinele unde sunt disponibile produsele.

****

**Figura 5–** Rezultatele interogării

1. ***Produse, categorii și angajații care au gestionat vânzările.***

Pentru a determina **produsele, categoriile și angajații care au gestionat vânzările**, trebuie să corelăm tabelele care conțin informațiile despre produse, categorii și angajați, respectiv comenzile gestionate de aceștia. Tabele implicate:

* **sales.order\_items**: Conține produsele comandate în fiecare comandă.
* **production.products**: Conține informații despre produse.
* **production.categories**: Conține informații despre categoriile produselor.
* **sales.orders**: Conține informațiile despre comenzi, inclusiv staff\_id, care identifică angajatul responsabil de comandă.
* **sales.staffs**: Conține informații despre angajați.

USE [BikeStores]

SELECT p.product\_name,

c.category\_name,

s.first\_name AS staff\_first\_name,

s.last\_name AS staff\_last\_name

FROM sales.order\_items oi

JOIN production.products p ON oi.product\_id = p.product\_id

JOIN production.categories c ON p.category\_id = c.category\_id

JOIN sales.orders o ON oi.order\_id = o.order\_id

JOIN sales.staffs s ON o.staff\_id = s.staff\_id;

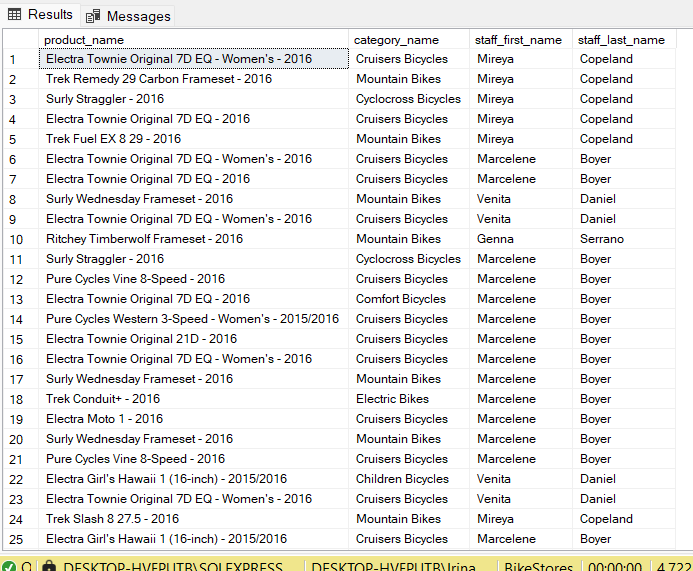
GO

**Coloane afișate:**

* **product\_name**: Numele produsului (din tabelul **production.products**).
* **category\_name**: Numele categoriei produsului (din tabelul **production.categories**).
* **staff\_first\_name**: Prenumele angajatului care a gestionat comanda (din tabelul **sales.staffs**).
* **staff\_last\_name**: Numele de familie al angajatului care a gestionat comanda (din tabelul **sales.staffs**).

**JOIN-uri:**

* **Între sales.order\_items și production.products**: Leagă produsele comandate de detaliile produselor.
* **Între production.products și production.categories**: Leagă produsele de categoriile lor.
* **Între sales.order\_items și sales.orders**: Leagă produsele comandate de comenzile respective.
* **Între sales.orders și sales.staffs**: Leagă comenzile de angajații care le-au gestionat.

****

**Figura 6 –** Rezultatele interogării

1. ***Magazine și numărul total de produse vândute.***

Pentru a determina **magazinele și numărul total de produse vândute**, trebuie să corelăm tabelele care conțin informațiile despre magazine, produsele comandate și comenzile. Vom calcula numărul total de produse vândute pentru fiecare magazin pe baza comenzilor și articolelor din comenzile respective. Tabele implicate:

* **sales.orders**: Conține informații despre comenzi, inclusiv store\_id, care identifică magazinul.
* **sales.order\_items**: Conține produsele comandate și cantitățile.
* **sales.stores**: Conține informațiile despre magazine.

USE [BikeStores]

SELECT st.store\_name,

SUM(oi.quantity) AS total\_products\_sold

FROM sales.orders o

JOIN sales.order\_items oi ON o.order\_id = oi.order\_id

JOIN sales.stores st ON o.store\_id = st.store\_id

GROUP BY st.store\_name;

GO

**Coloane afișate:**

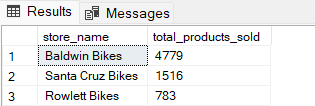
* **store\_name**: Numele magazinului (din tabelul **sales.stores**).
* **total\_products\_sold**: Numărul total de produse vândute de fiecare magazin, calculat prin sumarea cantităților din **sales.order\_items**.

**JOIN-uri:**

* **Între sales.orders și sales.order\_items**: Leagă comenzile de produsele comandate.
* **Între sales.orders și sales.stores**: Leagă comenzile de magazinul respectiv.

**Grupare:**

* Se grupează după store\_name pentru a calcula suma cantităților pentru fiecare magazin în parte.

****

**Figura 7 –** Rezultatele interogării

1. ***Clienți și comenzile lor (inclusiv clienții fără comenzi).***

Pentru a determina **clienții și comenzile lor**, inclusiv **clienții fără comenzi**, va trebui să utilizăm o interogare SQL care să includă toți clienții, chiar și cei care nu au plasat nicio comandă. Pentru a face acest lucru, vom folosi un **LEFT JOIN** între tabelul sales.customers și sales.orders. Acest tip de JOIN va asigura că toți clienții vor fi incluși în rezultat, chiar dacă nu au comenzi asociate. Tabele implicate:

* **sales.customers**: Conține informațiile despre clienți.
* **sales.orders**: Conține informațiile despre comenzi.

USE [BikeStores]

SELECT c.first\_name AS customer\_first\_name,

c.last\_name AS customer\_last\_name,

o.order\_id

FROM sales.customers c

LEFT JOIN sales.orders o ON c.customer\_id = o.customer\_id

ORDER BY c.first\_name, c.last\_name;

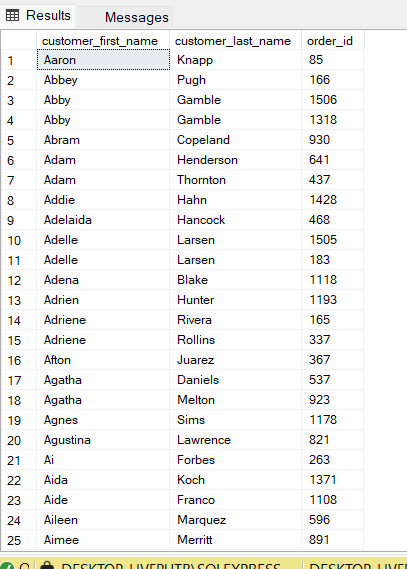
GO

**Coloane afișate:**

* **customer\_first\_name și customer\_last\_name**: Numele clientului (din tabelul **sales.customers**).
* **order\_id**: ID-ul comenzii (din tabelul **sales.orders**). Pentru clienții care nu au plasat nicio comandă, această coloană va fi NULL.

**LEFT JOIN:** este folosit pentru a include toți clienții din tabelul sales.customers, chiar și cei care nu au comenzi. Dacă un client nu a făcut nicio comandă, câmpul order\_id va fi NULL pentru acea persoană.

**ORDONARE:** Interogarea va ordona rezultatele după **numele** și **prenumele** clientului pentru a le face mai ușor de citit.

******

**Figura 8 –** Rezultatele interogării

1. ***Produse și stocuri disponibile (inclusiv produse fără stocuri).***

Pentru a determina **produsele și stocurile disponibile**, inclusiv **produsele fără stocuri**, trebuie să corelăm tabelele care conțin informațiile despre produse și stocurile lor. Vom folosi un **LEFT JOIN** între tabelele **production.products** și **production.stocks** pentru a include toate produsele, chiar și cele care nu au stocuri disponibile. Tabele implicate:

* **production.products**: Conține informațiile despre produse.
* **production.stocks**: Conține informațiile despre stocurile disponibile ale produselor în diferite magazine.

USE [BikeStores]

SELECT p.product\_name,

st.quantity AS stock\_quantity

FROM production.products p

LEFT JOIN production.stocks st ON p.product\_id = st.product\_id

ORDER BY p.product\_name;

GO

**Coloane afișate:**

* **product\_name**: Numele produsului (din tabelul **production.products**).
* **stock\_quantity**: Cantitatea disponibilă în stoc a fiecărui produs (din tabelul **production.stocks**). Dacă produsul nu are stocuri disponibile, această valoare va fi NULL.

**LEFT JOIN:** este folosit pentru a include toate produsele din tabelul **production.products**, chiar și produsele care nu au stocuri disponibile. Dacă un produs nu are stocuri, câmpul stock\_quantity va fi NULL pentru acel produs.

**ORDONARE:** Interogarea va ordona rezultatele după **numele produsului** pentru a face lista mai ușor de citit.

******

**Figura 9 –** Rezultatele interogării

1. ***Angajați și comenzile gestionate (inclusiv angajații fără comenzi).***

Pentru a determina **angajații și comenzile gestionate**, inclusiv **angajații care nu au gestionat nicio comandă**, va trebui să utilizăm un **LEFT JOIN** între tabelul **sales.staffs** și tabelul **sales.orders**. Acest tip de JOIN va asigura că toți angajații vor fi incluși în rezultat, chiar dacă nu au gestionat vreo comandă. Tabele implicate:

* **sales.staffs**: Conține informațiile despre angajați, inclusiv staff\_id.
* **sales.orders**: Conține informațiile despre comenzi, inclusiv staff\_id, care indică angajatul care a gestionat comanda.

USE [BikeStores]

SELECT s.first\_name AS staff\_first\_name,

s.last\_name AS staff\_last\_name,

o.order\_id

FROM sales.staffs s

LEFT JOIN sales.orders o ON s.staff\_id = o.staff\_id

ORDER BY s.first\_name, s.last\_name;

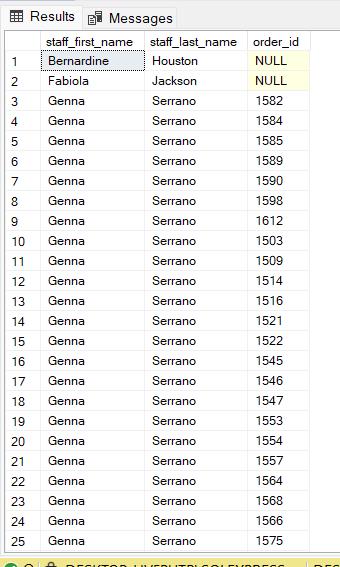
GO

**Coloane afișate:**

* **staff\_first\_name și staff\_last\_name**: Numele angajatului (din tabelul **sales.staffs**).
* **order\_id**: ID-ul comenzii (din tabelul **sales.orders**). Dacă un angajat nu a gestionat nicio comandă, această coloană va fi NULL.

**LEFT JOIN:** este folosit pentru a include toți angajații din tabelul **sales.staffs**, chiar și cei care nu au gestionat vreo comandă. Dacă un angajat nu a gestionat nicio comandă, câmpul order\_id va fi NULL.

**ORDONARE:** Interogarea va ordona rezultatele după **numele** și **prenumele** angajatului pentru a le face mai ușor de citit.

******

**Figura 10 –** Rezultatele interogării

1. ***Produse și categoriile lor (inclusiv categorii fără produse).***

Pentru a determina **produsele și categoriile lor**, inclusiv **categoriile care nu au produse**, vom utiliza o interogare SQL care combină tabelele **production.products** și **production.categories** folosind un **LEFT JOIN**. Acest tip de **JOIN** va asigura că toate categoriile vor fi incluse în rezultat, chiar și cele care nu au produse asociate. Tabele implicate:

* **production.products**: Conține informațiile despre produse, inclusiv product\_id și category\_id.
* **production.categories**: Conține informațiile despre categorii, inclusiv category\_id și category\_name.

USE [BikeStores]

SELECT c.category\_name,

p.product\_name

FROM production.categories c

LEFT JOIN production.products p ON c.category\_id = p.category\_id

ORDER BY c.category\_name;

GO

**Coloane afișate:**

* **category\_name**: Numele categoriei (din tabelul **production.categories**).
* **product\_name**: Numele produsului (din tabelul **production.products**). Dacă o categorie nu are produse asociate, această coloană va fi NULL.

**LEFT JOIN:** este folosit pentru a include toate categoriile din tabelul **production.categories**, chiar și categoriile care nu au produse asociate. Dacă o categorie nu are produse, câmpul product\_name va fi NULL pentru acea categorie.

**ORDONARE:** Interogarea va ordona rezultatele după **numele categoriei** pentru a face lista mai ușor de citit.

******

**Figura 11 –** Rezultatele interogării

1. ***Comenzi și magazinele aferente (inclusiv magazine fără comenzi).***

Pentru a determina **comenzile și magazinele aferente**, inclusiv **magazinele care nu au comenzi**, trebuie să utilizăm un **LEFT JOIN** între tabelele **sales.orders** și **sales.stores**. Astfel, vom include toate magazinele din tabelul **sales.stores**, chiar și cele care nu au plasat nicio comandă. Tabele implicate:

* **sales.orders**: Conține informațiile despre comenzi, inclusiv store\_id, care indică magazinul care a procesat comanda.
* **sales.stores**: Conține informațiile despre magazine, inclusiv store\_id și store\_name.

USE [BikeStores]

SELECT st.store\_name,

o.order\_id

FROM sales.stores st

LEFT JOIN sales.orders o ON st.store\_id = o.store\_id

ORDER BY st.store\_name;

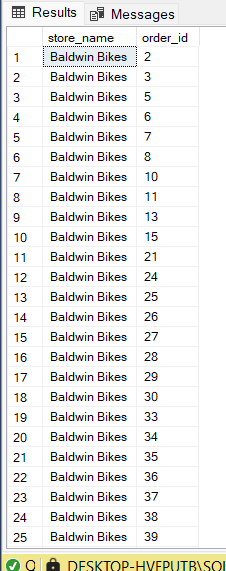
GO

**Coloane afișate:**

* **store\_name**: Numele magazinului (din tabelul **sales.stores**).
* **order\_id**: ID-ul comenzii plasate de magazin (din tabelul **sales.orders**). Dacă un magazin nu a plasat nicio comandă, această coloană va fi NULL.

**LEFT JOIN:** este folosit pentru a include toate magazinele din tabelul **sales.stores**, chiar și magazinele care nu au plasat nicio comandă. Dacă un magazin nu a plasat nicio comandă, câmpul order\_id va fi NULL pentru acea linie.

**ORDONARE:** Interogarea va ordona rezultatele după **numele magazinului** pentru a le face mai ușor de citit.

******

**Figura 12 –** Rezultatele interogării

1. ***Produse comandate și stocuri aferente (inclusiv stocuri fără produse comandate).***

Pentru a determina **produsele comandate și stocurile aferente**, inclusiv **stocurile fără produse comandate**, vom utiliza o interogare care combină tabelele **sales.order\_items** (care conține produsele comandate) și **production.stocks** (care conține informațiile despre stocuri). Vom folosi un **LEFT JOIN** între aceste tabele pentru a include toate stocurile, chiar și cele care nu au produse comandate. Tabele implicate:

* **sales.order\_items**: Conține informațiile despre produsele comandate, inclusiv product\_id și order\_id.
* **production.stocks**: Conține informațiile despre stocurile disponibile, inclusiv product\_id și quantity.

USE [BikeStores]

SELECT p.product\_name,

st.quantity AS stock\_quantity

FROM sales.order\_items oi

JOIN production.products p ON oi.product\_id = p.product\_id

LEFT JOIN production.stocks st ON p.product\_id = st.product\_id

ORDER BY p.product\_name;

GO

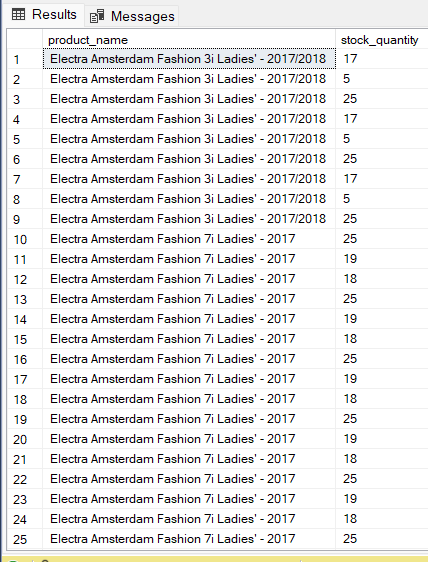
**Coloane afișate:**

* **product\_name**: Numele produsului (din tabelul **production.products**).
* **stock\_quantity**: Cantitatea disponibilă în stoc a fiecărui produs (din tabelul **production.stocks**). Dacă un produs nu are stocuri disponibile, această valoare va fi NULL.

**JOIN-uri:**

* **JOIN** între **sales.order\_items** și **production.products** este realizat pe product\_id pentru a obține numele produsului comandat.
* **LEFT JOIN** între **production.products** și **production.stocks** pentru a aduce informațiile despre stocuri. Acest **LEFT JOIN** asigură că toate produsele vor fi incluse în rezultat, chiar și cele care nu au stocuri disponibile (în acest caz, câmpul stock\_quantity va fi NULL).

**ORDONARE:** Interogarea va ordona rezultatele după **numele produsului** pentru a le face mai ușor de citit.

******

**Figura 13 –** Rezultatele interogării

1. ***Managerii angajaților.***

Pentru a determina **managerii angajaților**, presupunând că există o relație ierarhică în cadrul angajaților (un angajat poate fi manager al altui angajat), vom utiliza un **SELF JOIN** pe tabelul **sales.staffs**. În acest caz, fiecare angajat are un **manager\_id** care se leagă de **staff\_id** al unui alt angajat, care este managerul său. Tabele implicate:

* **sales.staffs**: Conține informațiile despre angajați, inclusiv staff\_id (ID-ul angajatului) și manager\_id (ID-ul managerului).

USE [BikeStores]

SELECT e.first\_name AS employee\_first\_name,

e.last\_name AS employee\_last\_name,

m.first\_name AS manager\_first\_name,

m.last\_name AS manager\_last\_name

FROM sales.staffs e

LEFT JOIN sales.staffs m ON e.manager\_id = m.staff\_id

ORDER BY e.first\_name, e.last\_name;

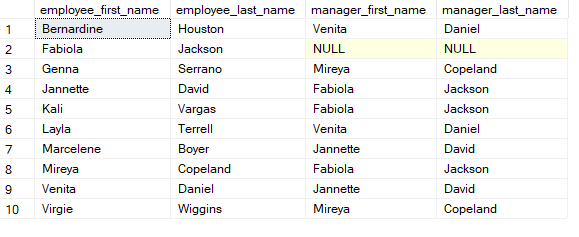
GO

**Coloane afișate:**

* **employee\_first\_name și employee\_last\_name**: Numele angajatului (din tabelul **sales.staffs**).
* **manager\_first\_name și manager\_last\_name**: Numele managerului angajatului. Dacă angajatul nu are un manager (adică manager\_id este NULL), aceste coloane vor fi și ele NULL.

**LEFT JOIN:** este folosit pentru a include toți angajații, chiar și aceia care nu au un manager (cazul în care manager\_id este NULL). Dacă un angajat nu are un manager, câmpurile manager\_first\_name și manager\_last\_name vor fi NULL.

**ORDONARE:** Interogarea va ordona rezultatele după **numele angajatului** pentru a le face mai ușor de citit.

******

**Figura 14 –** Rezultatele interogării

1. ***Magazine din același oraș.***

Pentru a determina **magazinele din același oraș**, va trebui să utilizăm un **SELF JOIN** pe tabelul **sales.stores**. În acest caz, fiecare magazin are un câmp **city** care indică orașul în care se află. Vom efectua o îmbinare între același tabel pentru a compara orașele magazinelor și a obține doar magazinele care se află în același oraș. Tabelul implicat:

* **sales.stores**: Conține informațiile despre magazine, inclusiv store\_id, store\_name și city.

USE [BikeStores]

SELECT s1.store\_name AS store1\_name,

s2.store\_name AS store2\_name,

s1.city

FROM sales.stores s1

JOIN sales.stores s2 ON s1.city = s2.city

WHERE s1.store\_id <> s2.store\_id

ORDER BY s1.city, s1.store\_name, s2.store\_name;

GO

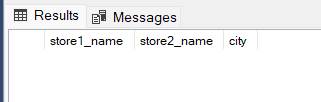
**Coloane afișate:**

* **store1\_name și store2\_name**: Numele celor două magazine din același oraș.
* **city**: Orașul în care se află ambele magazine.

**JOIN:** între cele două instanțe ale tabelului **sales.stores** (alias s1 și s2) pe câmpul **city**. Acesta va compara orașele magazinelor și va aduce doar magazinele care se află în același oraș.

**Filtrare (WHERE):** **WHERE s1.store\_id <> s2.store\_id**: Asigurăm că un magazin nu este comparat cu el însuși. Adică, nu vor apărea perechi de magazine care sunt același magazin.

**ORDONARE:** Interogarea va ordona rezultatele după **oraș** și apoi după numele magazinelor pentru a le face mai ușor de citit.



**Figura 15 –** Rezultatele interogării

# **CONCLUZII**

În această lucrare de laborator, am aprofundat metode avansate de interogare SQL, punând accent pe utilizarea diverselor tipuri de compuneri (JOIN) pentru a analiza și extrage date relevante din tabele multiple. Am evidențiat rolul crucial al relațiilor dintre tabele într-o bază de date relațională și am explorat provocările asociate interogărilor complexe.

Am aplicat în mod eficient principalele tipuri de JOIN, precum:

1. **INNER JOIN** – pentru a selecta doar datele comune între tabele.
2. **LEFT JOIN** și **RIGHT JOIN** – pentru a include toate înregistrările dintr-un tabel, chiar dacă nu există corespondențe în celălalt, utile pentru identificarea entităților fără relații directe (ex.: clienți fără comenzi sau produse nevândute).
3. **SELF JOIN** – pentru a analiza relațiile dintre înregistrări din același tabel (ex.: ierarhii sau asocieri între entități).

De asemenea, am utilizat funcții precum **GROUP BY**, **COUNT**, și **SUM** pentru agregarea și sintetizarea datelor complexe, cum ar fi numărul total de produse vândute sau analiza stocurilor disponibile.

Pe parcursul lucrării, am constatat că unele interogări, precum cea de la punctul 15, nu au returnat rezultate. Această situație poate fi atribuită lipsei relațiilor sau datelor corespunzătoare în tabelele implicate. De exemplu, absența stocurilor sau a condițiilor specifice în baza de date actuală poate duce la un rezultat gol, subliniind importanța verificării datelor existente și a proiectării corecte a interogărilor.

În concluzie, această activitate a oferit o perspectivă practică asupra interogărilor SQL avansate, subliniind atât capacitatea lor de a analiza datele, cât și limitele impuse de structura și conținutul bazei de date. Am dezvoltat abilități valoroase pentru manipularea datelor relaționale, necesare pentru gestionarea bazelor de date complexe și rezolvarea provocărilor din mediul profesional.