# **Proiect SQL-bază de date papetărie**

**Baza de date:**

1. Produse: Aceasta este esența afacerii. Aici sunt stocate detalii despre fiecare produs, cum ar fi numele, o descriere succintă, prețul și numărul de produse din stoc. Exemple de produse ar putea fi "Caiet A4" sau "Pix albastru".
2. Clienți: Fidelizarea clienților este o prioritate pentru "Notițe și Idei". În această secțiune, sunt stocate detalii despre clienții care comandă produse, cum ar fi numele, adresa, e-mailul și numărul de telefon.
3. Comenzi: Fiecare dată când un client face o achiziție, aceasta este înregistrată. Această entitate stochează data comenzii și legătura către clientul care a efectuat-o.
4. DetaliiComenzi: Pentru a ști ce produse specifice au fost cumpărate într-o comandă, această entitate face legătura dintre comenzile specifice și produsele achiziționate. Astfel, dacă Ion Popescu a cumpărat două caiete și cinci pixuri, aceasta va fi evidențiată aici.

**Funcționalitatea și Utilitatea:**

O dată ce baza de date a fost creată și populată cu produse și clienți, personalul poate accesa ușor informațiile atunci când este nevoie. De exemplu, pot verifica rapid stocurile de produse și pot vedea ce produse sunt cele mai populare sau pot identifica clienții care fac comenzi frecvente. De asemenea, această bază de date ajută la planificarea stocurilor, urmărind produsele care sunt la mare căutare și cele care rămân nevândute.

**Concluzie:**

Într-o lume în care digitalizarea a devenit normă, chiar și o afacere tradițională precum o papetărie poate beneficia de avantajele unei baze de date bine structurate. Aceasta îmbunătățește eficiența, oferă informații valoroase și contribuie la un serviciu client mai bun.

Cerințele proiectului includ:

1. identificarea entităților de business care necesită stocarea informațiilor despre acestea;Entități și atribute:

Produse:

* produs\_id (INTEGER, PRIMARY KEY, AUTO\_INCREMENT)
* nume (VARCHAR(100), NOT NULL)
* descriere (VARCHAR(255))
* pret (DECIMAL(8, 2), NOT NULL)
* stoc (INTEGER, NOT NULL)

Clienti:

* client\_id (INTEGER, PRIMARY KEY, AUTO\_INCREMENT)
* nume (VARCHAR(100), NOT NULL)
* adresa (VARCHAR(255), NOT NULL)
* email (VARCHAR(100))
* telefon (VARCHAR(20))

Comenzi:

* comanda\_id (INTEGER, PRIMARY KEY, AUTO\_INCREMENT)
* client\_id (INTEGER, FOREIGN KEY -> Clienti.client\_id)
* data\_comanda (DATE, NOT NULL)

DetaliiComenzi:

* detaliu\_id (INTEGER, PRIMARY KEY, AUTO\_INCREMENT)
* comanda\_id (INTEGER, FOREIGN KEY -> Comenzi.comanda\_id)
* produs\_id (INTEGER, FOREIGN KEY -> Produse.produs\_id)
* cantitate (INTEGER, NOT NULL)

2. identificarea relatiilor intre entitatile mentionate anterior;

Relații între entități:

***Relația dintre Clienti și Comenzi:***

* Un Client poate plasa mai multe Comenzi.
* Comandă este plasată de un singur Client.
* Acesta este un tip de relație "one-to-many" (unul la mulți) dintre Clienti și Comenzi.
* Atributul client\_id în tabelul Comenzi este o cheie străină care face referire la client\_id din tabelul Clienti.

***Relația dintre Comenzi și DetaliiComenzi:***

* Comandă poate conține mai multe produse, fiecare cu o anumită cantitate. Aceste detalii sunt stocate în tabela DetaliiComenzi.
* înregistrare în DetaliiComenzi se referă la o singură Comandă.
* Aceasta este o relație "one-to-many" (unul la mulți) dintre Comenzi și DetaliiComenzi.
* Atributul comanda\_id în DetaliiComenzi este o cheie străină care face referire la comanda\_id în tabelul Comenzi.

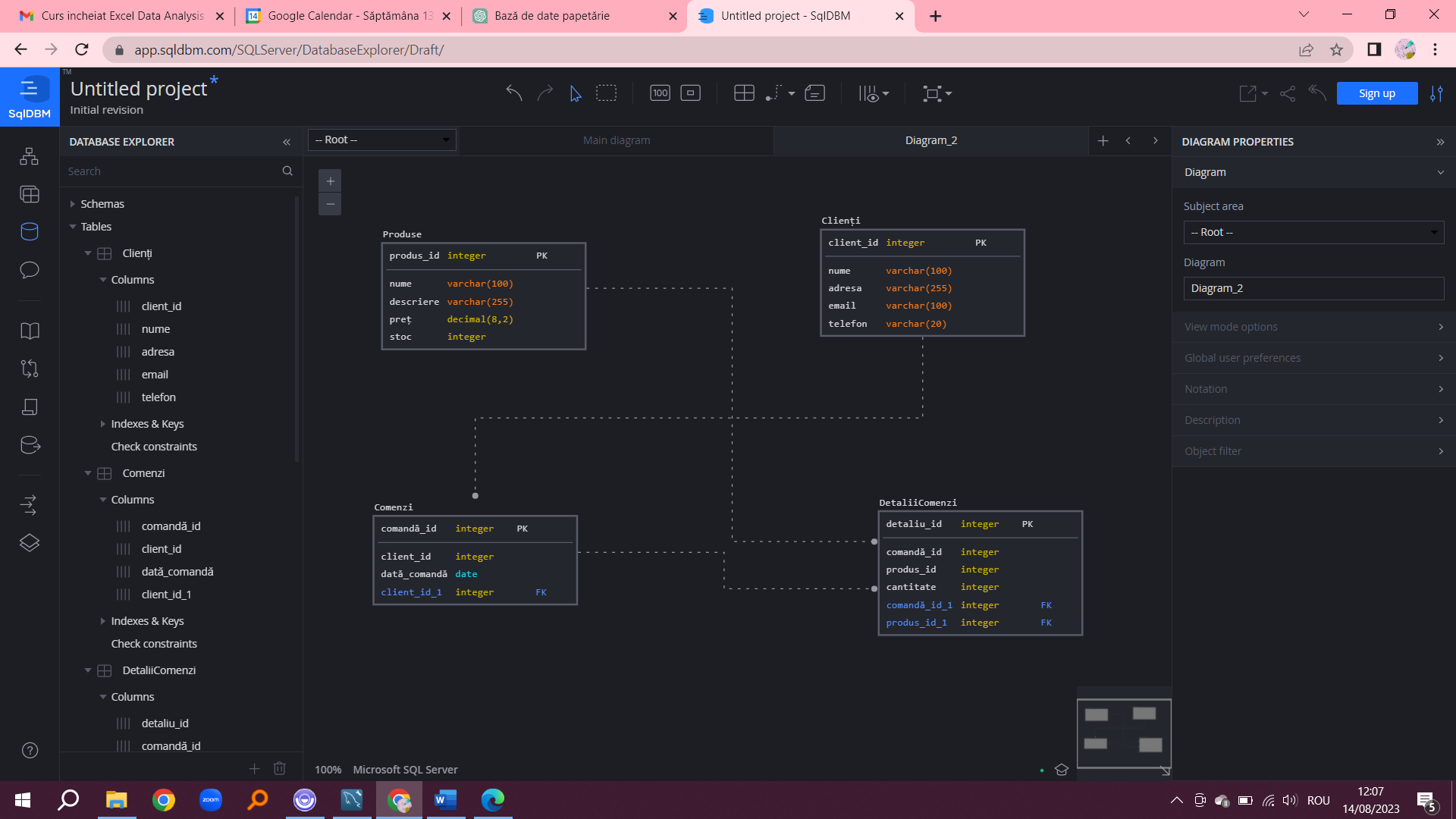
***Relația dintre Produse și DetaliiComenzi:***

* Un Produs poate fi parte a mai multor Comenzi. Aceste detalii sunt reprezentate în tabela DetaliiComenzi.
* înregistrare în DetaliiComenzi se referă la un singur Produs.
* Acesta este un tip de relație "one-to-many" (unul la mulți) dintre Produse și DetaliiComenzi.
* Atributul produs\_id în DetaliiComenzi este o cheie străină care face referire la produs\_id în tabelul Produse.

3. crearea diagramei entitate-relatie (substantiv-verb);

4. stabilirea schemei fiecarei entitati, a atributelor acesteia si a domeniilor atributelor (stabilirea tipurilor de data pentru fiecare atribut);

5. stabilirea eventualelor restrictii suplimentare (chei unice, chei straine, NOT NULL,…);



***Restrictii suplimentare:***

* client\_id, produs\_id și comanda\_id sunt setate ca chei unice (PRIMARY KEY) pentru entitățile lor respective și sunt AUTO\_INCREMENT pentru a genera automat un ID nou pentru fiecare înregistrare nouă.
* Atributele client\_id din Comenzi și produs\_id, comanda\_id din DetaliiComenzi sunt setate ca chei străine (FOREIGN KEY) pentru a păstra relațiile dintre tabele.
* Atributele care sunt esențiale pentru operațiunile de afaceri, cum ar fi nume în Clienti, nume\_produs și pret în Produse, client\_id și data\_comanda în Comenzi, sunt setate ca NOT NULL pentru a se asigura că aceste câmpuri sunt întotdeauna populate.

6. crearea bazei de date in SGBDR MySQL (CREATE DATABASE proiect;);

7. crearea relatiilor intre tabele (FOREIGN KEY);

CREATE DATABASE papetarie;

USE papetarie;

CREATE TABLE Produse (

produs\_id INT AUTO\_INCREMENT PRIMARY KEY,

nume VARCHAR(100) NOT NULL,

descriere VARCHAR(255),

pret DECIMAL(8, 2) NOT NULL,

stoc INT NOT NULL);

CREATE TABLE Clienti (

client\_id INT AUTO\_INCREMENT PRIMARY KEY,

nume VARCHAR(100) NOT NULL,

adresa VARCHAR(255) NOT NULL,

email VARCHAR(100),

telefon VARCHAR(20));

CREATE TABLE Comenzi (

comanda\_id INT AUTO\_INCREMENT PRIMARY KEY,

client\_id INT,

data\_comanda DATE NOT NULL,

FOREIGN KEY (client\_id) REFERENCES Clienti(client\_id));

CREATE TABLE DetaliiComenzi (

detaliu\_id INT AUTO\_INCREMENT PRIMARY KEY,

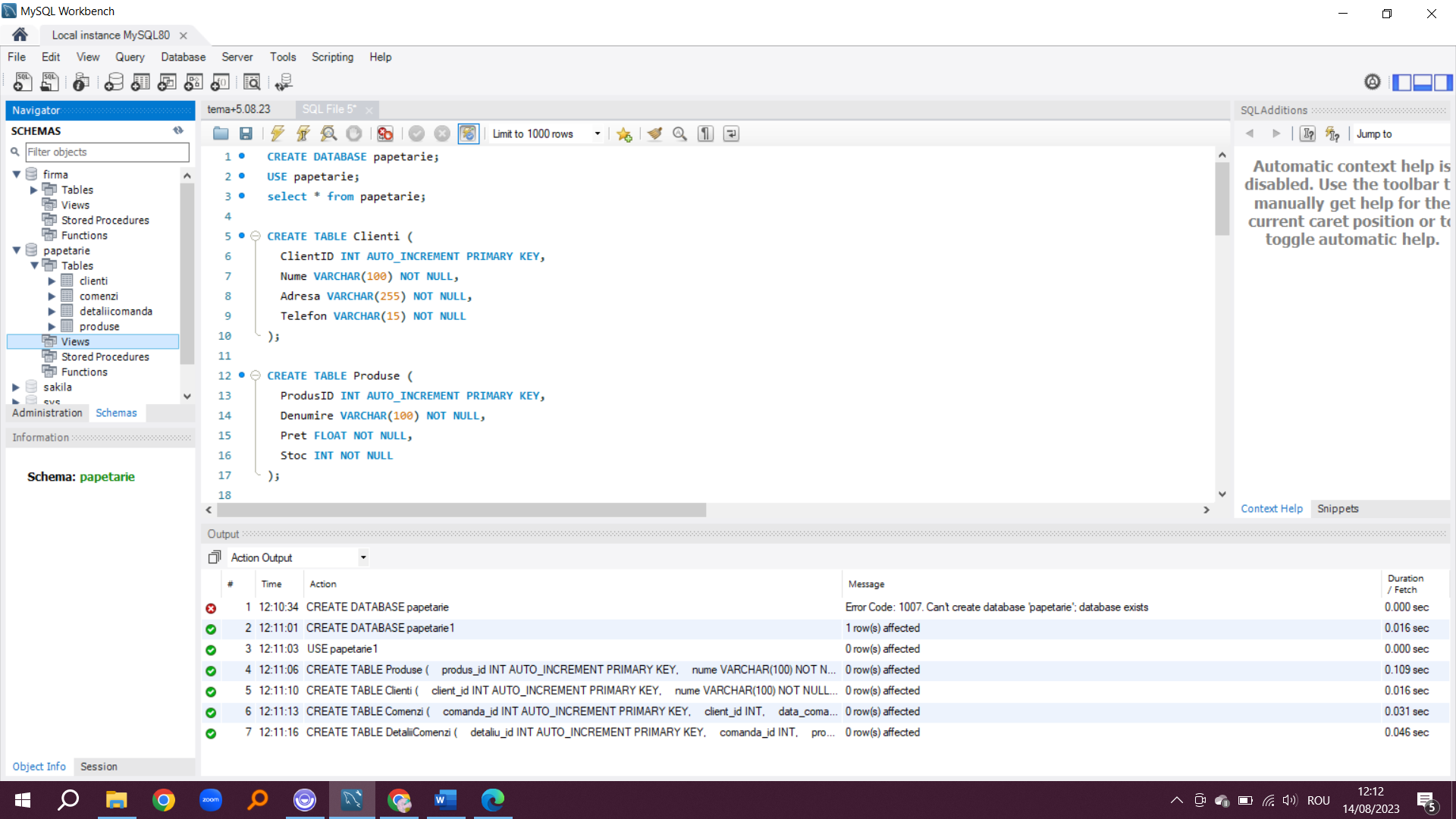
comanda\_id INT,

produs\_id INT,

cantitate INT NOT NULL,

FOREIGN KEY (comanda\_id) REFERENCES Comenzi(comanda\_id),

FOREIGN KEY (produs\_id) REFERENCES Produse(produs\_id));



8. crearea efectiva a tabelelor in baza de date, pentru fiecare entitate, respectand tipul de data

stabilit anterior pentru fiecare atribut (CREATE TABLE(…));+ DEMONSTRAȚIE

Înregistrări de demonstrație

**-- Adăugare produse**

INSERT INTO Produse (nume, descriere, pret, stoc)

VALUES ('Caiet A4', 'Caiet 80 file', 5.50, 100),

('Pix albastru', 'Pix cu gel', 1.50, 200);

**-- Adăugare clienți**

INSERT INTO Clienti (nume, adresa, email, telefon)

VALUES ('Ion Popescu', 'Strada Victoriei, nr. 10', 'ion.popescu@email.com', '0722001122'),

('Ana Ionescu', 'Strada Lalelelor, nr. 15', 'ana.ionescu@email.com', '0733004455');

**-- Adăugare comenzi**

INSERT INTO Comenzi (client\_id, data\_comanda)

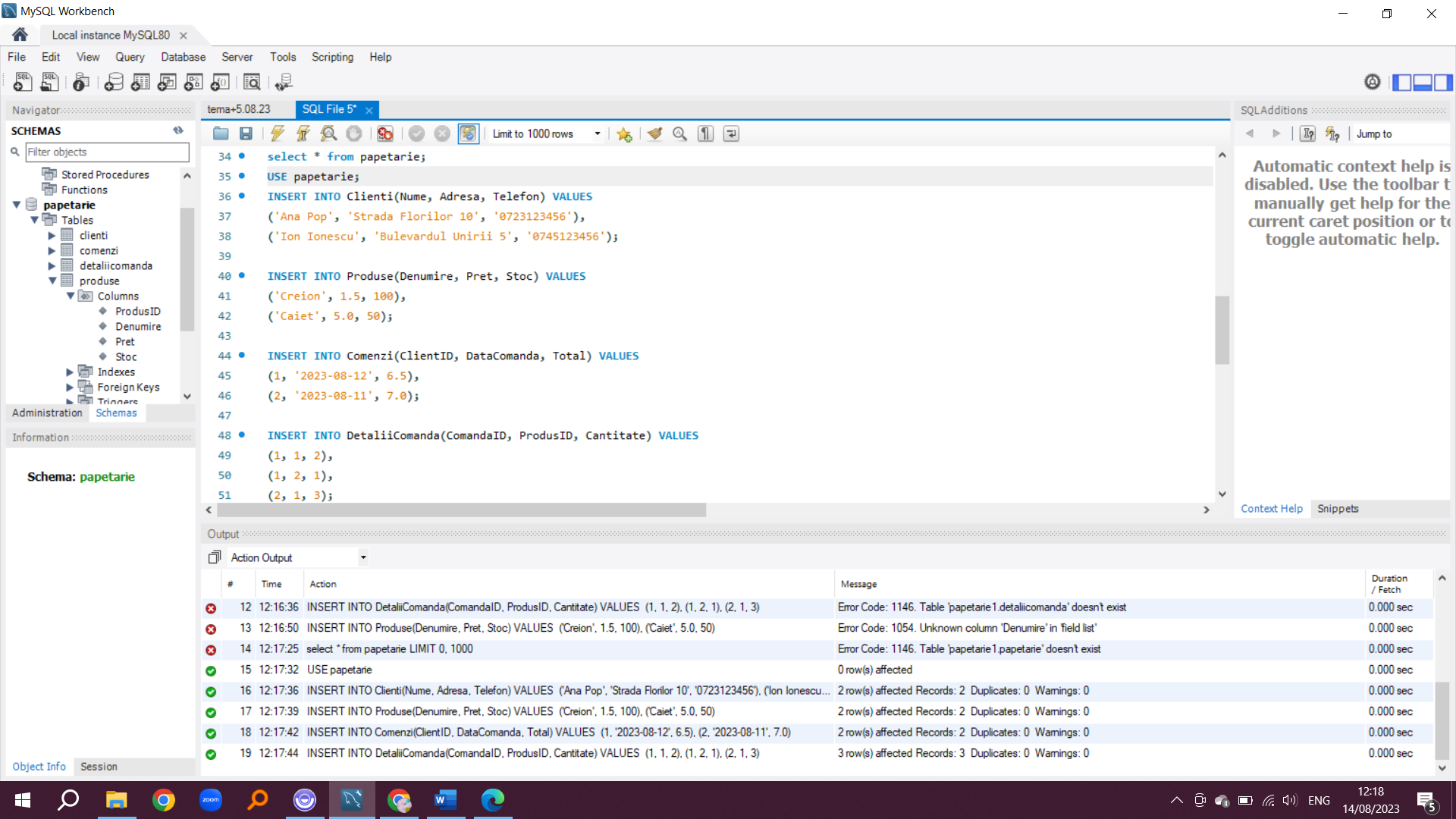
VALUES (1, '2023-08-10'), (2, '2023-08-11');

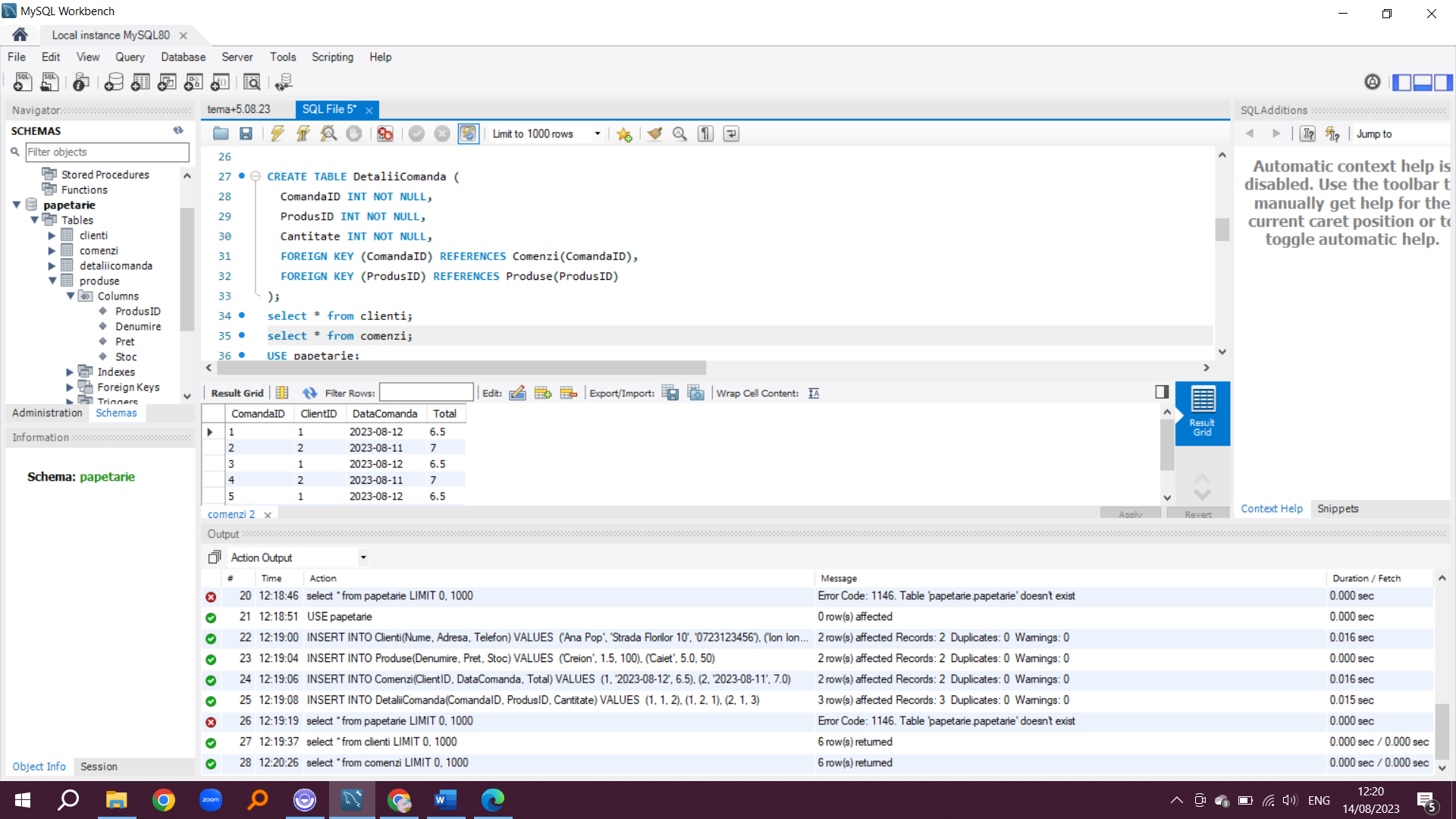
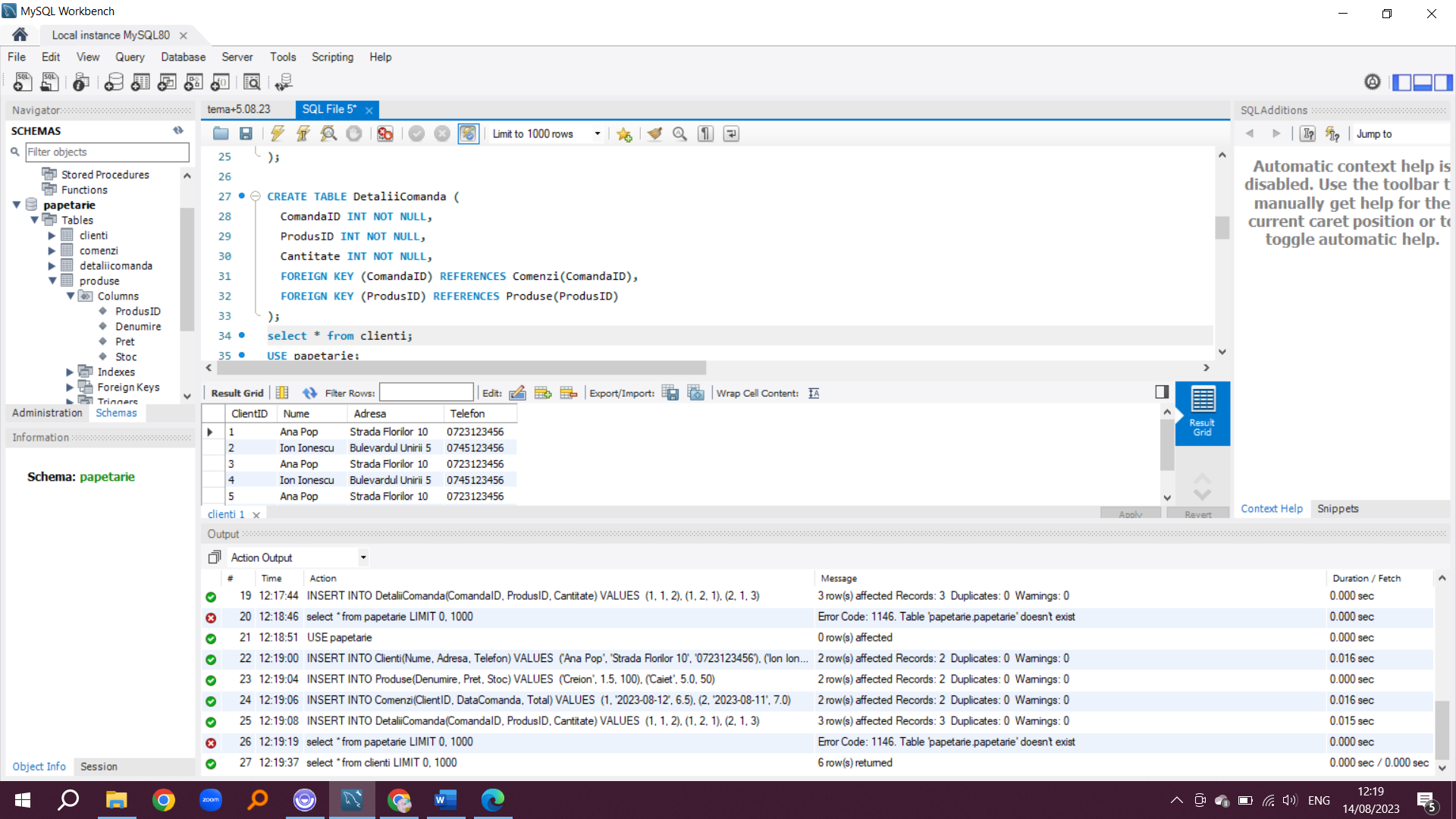
**-- Adăugare detalii comenzi**

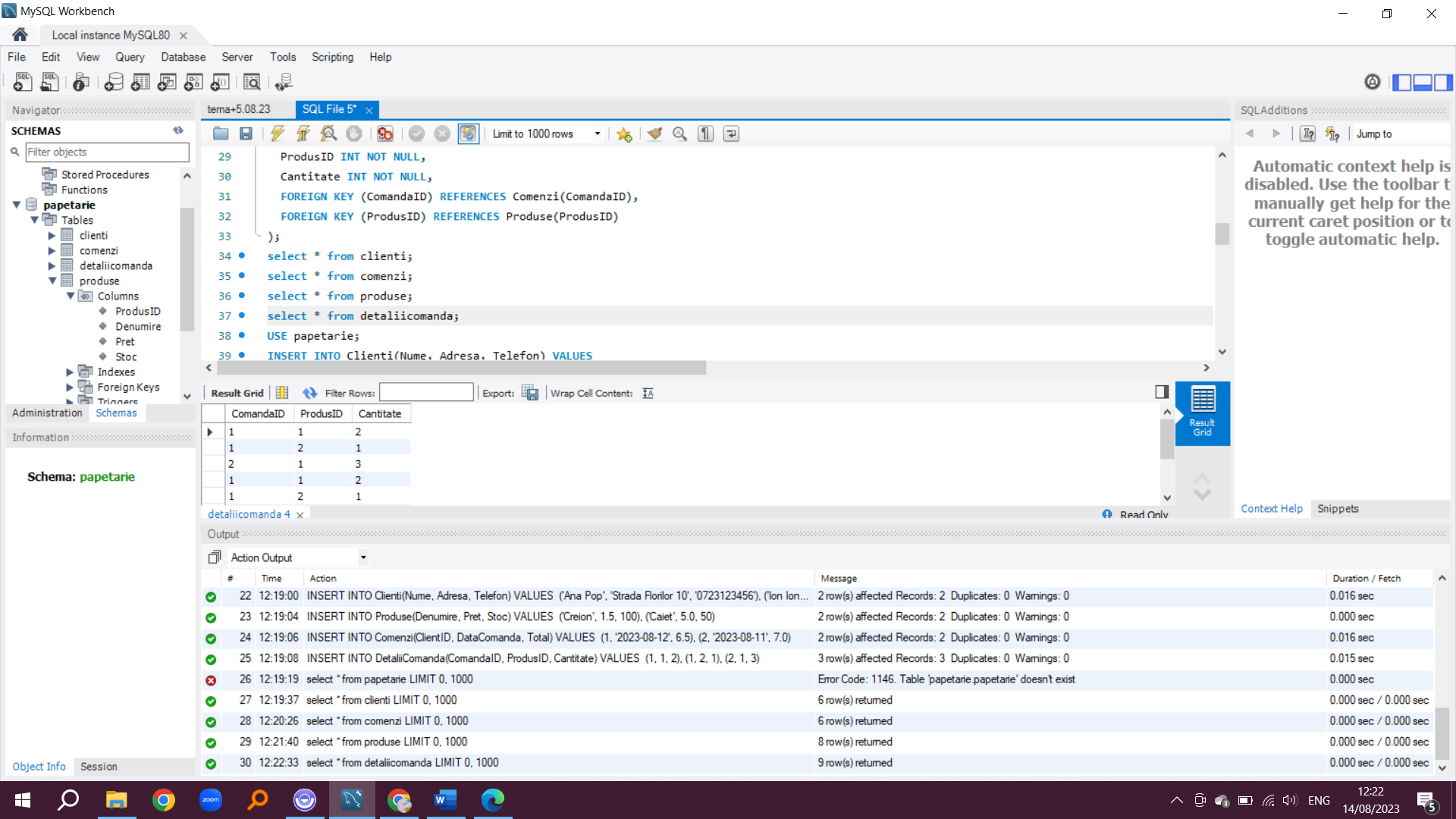
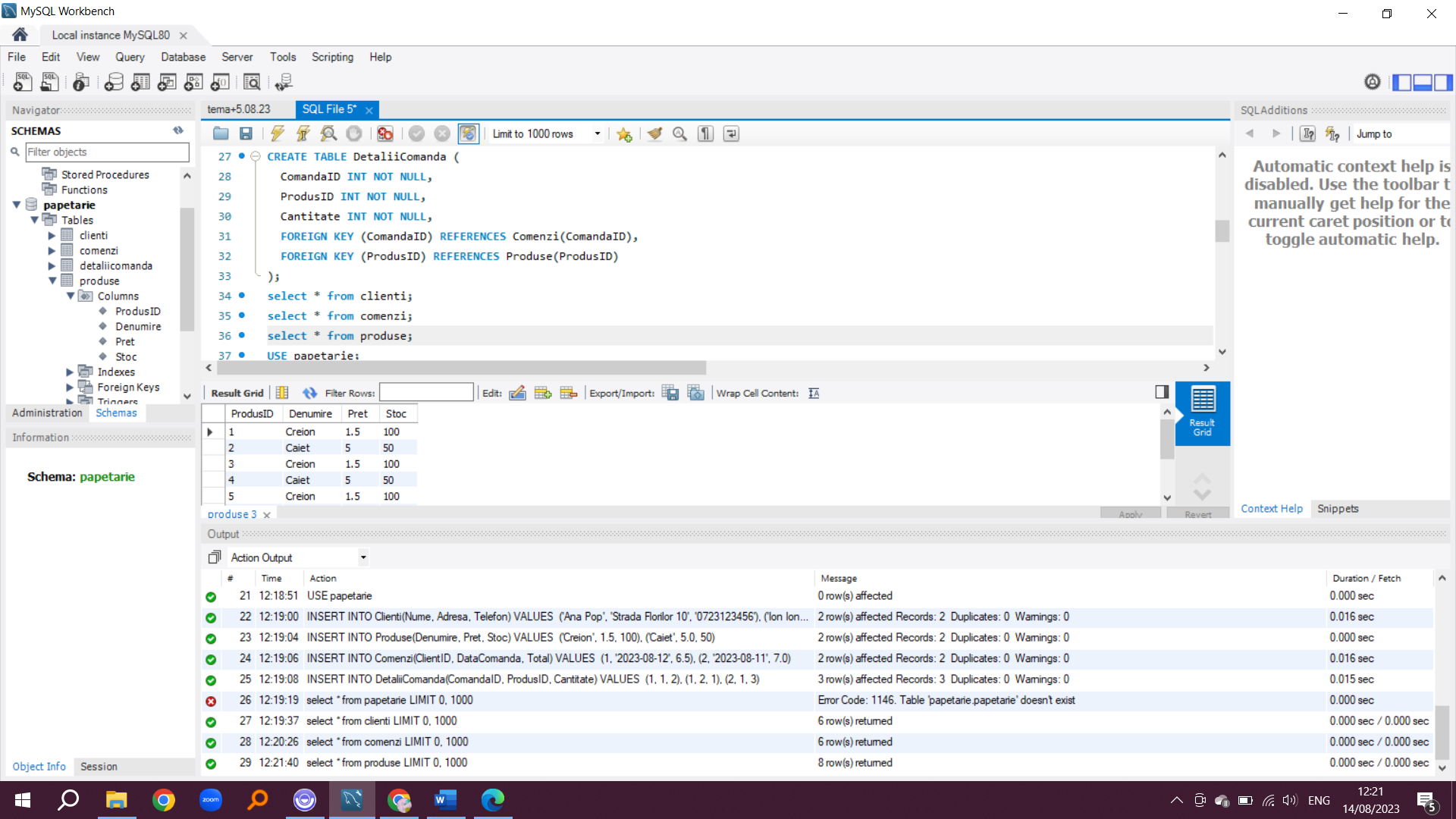
INSERT INTO DetaliiComenzi (comanda\_id, produs\_id, cantitate)

VALUES (1, 1, 2), -- Ion Popescu a comandat 2 caiete

(2, 2, 5); -- Ana Ionescu a comandat 5 pixuri







SCRIPTURILE CARE DOVEDESC REZULTATELE DE MAI SUS:

***-- Lista produselor vandabile***

SELECT \* FROM Produse;

***-- Lista clienților***

SELECT \* FROM Clienti;

***-- Lista comenzilor***

SELECT \* FROM Comenzi;

***-- Lista comenzilor și clienților***

SELECT c.ComandaID, cl.Nume, cl.Adresa, c.DataComanda FROM Comenzi c

JOIN Clienti cl ON c.ClientID = cl.ClientID;

***-- Ultima comanda a clientului 'Ana Pop'***

SELECT \* FROM Comenzi WHERE ClientID = 1 ORDER BY DataComanda DESC LIMIT 1;

***-- Statistici (un exemplu)***

SELECT p.Denumire, SUM(dc.Cantitate) as TotalVandut FROM Produse p

JOIN DetaliiComanda dc ON p.ProdusID = dc.ProdusID

GROUP BY p.Denumire;

***-- Folosind subinterogări***

SELECT Denumire,

(SELECT SUM(Cantitate) FROM DetaliiComanda WHERE ProdusID = p.ProdusID) as TotalVandut

FROM Produse p;

9. inserarea unui numar de inregistrari rezonabil pentru a putea fi demonstrata utilizarea bazei de date (5-20 de inregistrari per tabela sunt de regula suficiente);

**1. PRODUSE:**

CREATE TABLE Produse (

ProdusID INT PRIMARY KEY,

NumeProdus VARCHAR(50),

Pret DECIMAL(10, 2),

Stoc INT);

INSERT INTO Produse (ProdusID, NumeProdus, Pret, Stoc)

VALUES(1, 'Pix', 1.5, 100),

(2, 'Caiet', 5.0, 50),

(3, 'Gumă de șters', 0.5, 200),

(4, 'Marker', 2.5, 30),

(5, 'Acuarele', 15.0, 20);

**2. FURNIZORI:**

CREATE TABLE Furnizori (

FurnizorID INT PRIMARY KEY,

NumeFurnizor VARCHAR(50),

Contact VARCHAR(50));

INSERT INTO Furnizori (FurnizorID, NumeFurnizor, Contact)

VALUES

(1, 'Furnizor1', 'contact1@email.com'),

(2, 'Furnizor2', 'contact2@email.com'),

(3, 'Furnizor3', 'contact3@email.com'),

(4, 'Furnizor4', 'contact4@email.com'),

(5, 'Furnizor5', 'contact5@email.com');

**3. VANZARI:**

CREATE TABLE Vanzari (

VanzareID INT PRIMARY KEY,

ProdusID INT,

FurnizorID INT,

CantitateVanduta INT,

DataVanzare DATE,

FOREIGN KEY (ProdusID) REFERENCES Produse(ProdusID),

FOREIGN KEY (FurnizorID) REFERENCES Furnizori(FurnizorID));

INSERT INTO Vanzari (VanzareID, ProdusID, FurnizorID, CantitateVanduta, DataVanzare)

VALUES

(1, 1, 1, 10, '2023-01-01'),

(2, 2, 1, 5, '2023-01-02'),

(3, 3, 2, 20, '2023-01-03'),

(4, 4, 3, 6, '2023-01-04'),

(5, 5, 3, 4, '2023-01-05');

10. demonstrarea modului de utilizare a datelor prin crearea scripturilor (instructiunilor SQL) care furnizeaza urmatoarele rezultate:

* lista produselor vandabile (exemplu: lista tipurilor de pizza produsa sau a articolelor

sportive importabile);

SELECT \* FROM Produse;

* lista clientilor;

SELECT \* FROM Clienti;

* lista comenzilor inregistrate in sistem;

SELECT \* FROM Comenzi;

* lista comenzilor primite de la clienti, impreuna cu date ale respectivilor clienti (exemplu: lista comenzilor de pizza si numele si adresa cumparatorilor);

SELECT Comenzi.comanda\_id, Clienti.nume AS nume\_client, Clienti.adresa, Produse.nume AS nume\_produs, Comenzi.data\_comanda

FROM Comenzi

JOIN Clienti ON Comenzi.client\_id = Clienti.client\_id

JOIN Produse ON Comenzi.produs\_id = Produse.produs\_id;

* ultima comanda lansata de un anumit client

SELECT \* FROM Comenzi

WHERE client\_id = 5

ORDER BY data\_comanda DESC

LIMIT 1;

* 3 agregari de date in scop statistic (GROUP BY) care sa implice cel putin o legatura intre tabelele de date (exemplu cate pizza din fiecare tip au fost comandate, cati bani a cheltuit fiecare client sau lista clientilor care au comandat de minim doua ori/care au comandat minim doua produse diferite);

1. ***Câte produse din fiecare tip au fost comandate:***

SELECT p.nume\_produs,SUM(dc.cantitate) AS total\_comandat

FROM Produse p

JOIN DetaliiComenzi dc ON p.produs\_id = dc.produs\_id

GROUP BY p.nume\_produs;

***2. Câți bani a cheltuit fiecare client:***

SELECT c.nume, SUM(p.pret \* dc.cantitate) AS total\_cheltuit

FROM Clienti c

JOIN Comenzi co ON c.client\_id = co.client\_id

JOIN DetaliiComenzi dc ON co.comanda\_id = dc.comanda\_id

JOIN Produse p ON dc.produs\_id = p.produs\_id

GROUP BY c.nume;

***3. Lista clienților care au comandat minim două produse diferite:***

SELECT c.nume FROM Clienti c

JOIN Comenzi co ON c.client\_id = co.client\_id

JOIN DetaliiComenzi dc ON co.comanda\_id = dc.comanda\_id

GROUP BY c.nume

HAVING COUNT(DISTINCT dc.produs\_id) >= 2;

* oferirea cel putin pentru unul dintre scripturile de mai sus a unei alternative de scriere: pe baza de subinterogare, cu JOIN, cu vederi, etc.SELECT c.nume FROM Clienti c WHERE c.client\_id IN (

SELECT co.client\_id FROM Comenzi co

JOIN DetaliiComenzi dc ON co.comanda\_id = dc.comanda\_id

GROUP BY co.client\_id

HAVING COUNT(DISTINCT dc.produs\_id) >= 2 );