**PROIECT SGDB**

- MAGAZIN ONLINE -

OCNARU MIHAI-OCTAVIAN

1. **Prezentați pe scurt baza de date (utilitatea ei).**

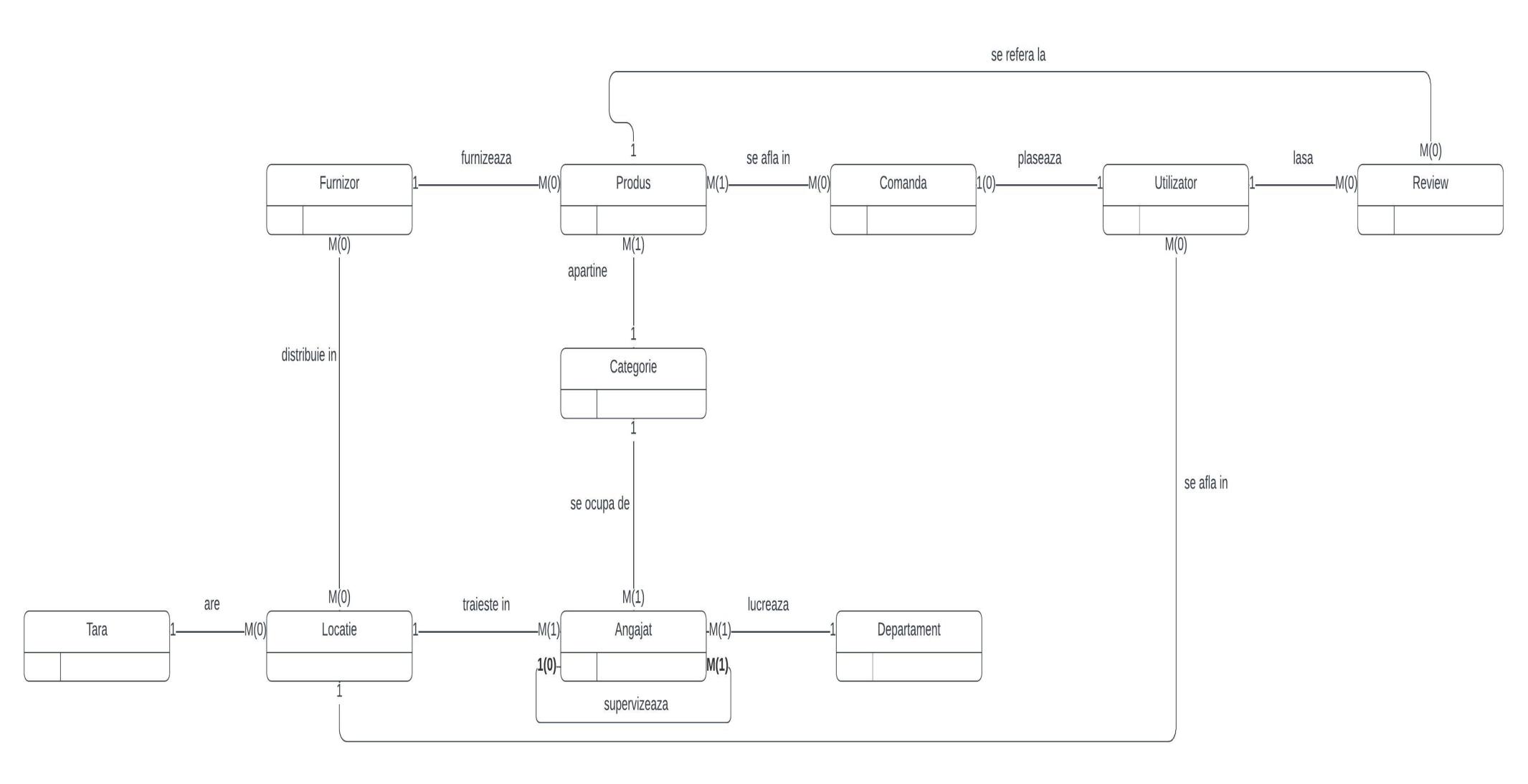
Modelul real al bazei de date pentru un magazin online este acela al unei afaceri care vinde produse online, primind comenzi de la clienți și livrând produsele la adresele specificate de aceștia. Magazinul online are o gamă largă de produse, inclusiv electronice, îmbrăcăminte, produse de îngrijire personală și multe altele.

Utilitatea bazei de date este de a gestiona și stoca informațiile despre produse, comenzi, clienți, plăți, recenzii, furnizori și angajați, permițând afacerii să-și gestioneze eficient operațiunile, să-și urmărească stocurile, să proceseze comenzi și să furnizeze un serviciu excelent clienților săi. Astfel pe baza acestor metrice putând să îmbunătațească

atât recomandările de produse, dar și alegerea furnizorilor potriviți cererii pieței.

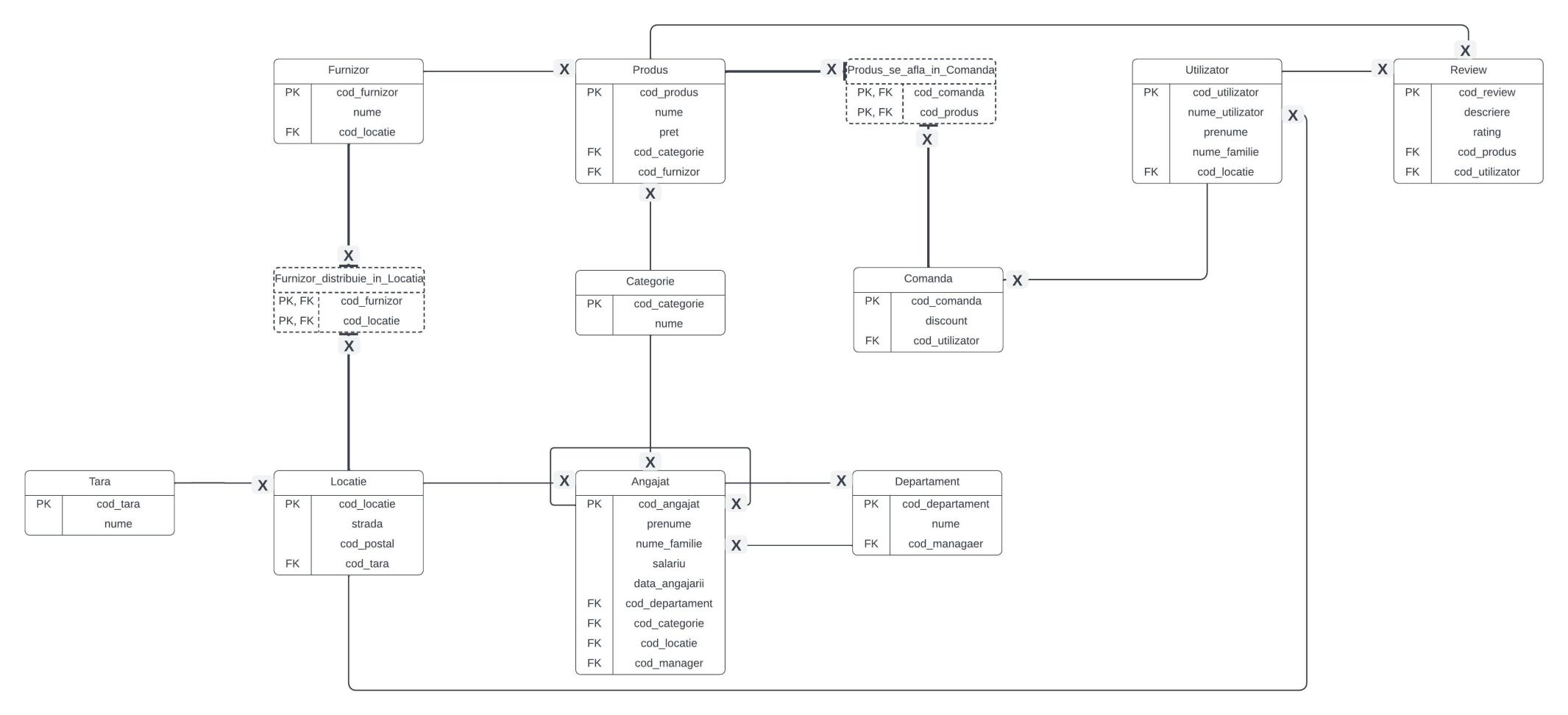
**2. Realizați diagrama entitate-relație (ERD): entitățile, relațiile și atributele trebuie definite în limba**

**română (vezi curs SGBD / model de diagrama ERD; nu se va accepta alt format).**



**3. Pornind de la diagrama entitate-relație realizați diagrama conceptuală a modelului propus, integrând**

**toate atributele necesare: entitățile, relațiile și atributele trebuie definite în limba română.**



**4. Implementați în Oracle diagrama conceptuală realizată: definiți toate tabelele, definind toate**

**constrângerile de integritate necesare (chei primare, cheile externe etc).**

– secventa pt o tabela

-- Tabela **Tara**

CREATE TABLE Tara (

    cod\_tara INT PRIMARY KEY,

    nume VARCHAR(50)

);

-- Tabela **Categorie**

CREATE TABLE Categorie (

    cod\_categorie INT PRIMARY KEY,

    nume VARCHAR(50)

);

-- Tabela **Locatie**

CREATE TABLE Locatie (

    cod\_locatie INT PRIMARY KEY,

    strada VARCHAR(50),

    cod\_postal VARCHAR(10),

    cod\_tara INT,

    FOREIGN KEY (cod\_tara) REFERENCES Tara(cod\_tara)

);

-- Tabela **Departament**

CREATE TABLE Departament (

    cod\_departament INT PRIMARY KEY,

    nume VARCHAR(50),

    cod\_manager INT

);

-- Tabela Angajat

CREATE TABLE **Angajat** (

    cod\_angajat INT PRIMARY KEY,

    prenume VARCHAR(50),

    nume\_familie VARCHAR(50),

    cod\_departament INT,

    cod\_categorie INT,

    cod\_locatie INT,

    cod\_manager INT,

    FOREIGN KEY (cod\_departament) REFERENCES Departament(cod\_departament),

    FOREIGN KEY (cod\_categorie) REFERENCES Categorie(cod\_categorie),

    FOREIGN KEY (cod\_locatie) REFERENCES Locatie(cod\_locatie),

    FOREIGN KEY (cod\_manager) REFERENCES Angajat(cod\_angajat)

);

-- Alterare tabel Departament pentru adăugarea restricției cheii externe către tabela Angajat

ALTER TABLE Departament

ADD CONSTRAINT FK\_Departament\_Angajat FOREIGN KEY (cod\_manager) REFERENCES Angajat(cod\_angajat);

-- Tabela Furnizor

CREATE TABLE **Furnizor** (

    cod\_furnizor INT PRIMARY KEY,

    nume VARCHAR(50),

    cod\_locatie INT,

    FOREIGN KEY (cod\_locatie) REFERENCES Locatie(cod\_locatie)

);

-- Tabela Produs

CREATE TABLE **Produs** (

    cod\_produs INT PRIMARY KEY,

    nume VARCHAR(50),

    pret DECIMAL(10,2),

    cod\_categorie INT,

    cod\_furnizor INT,

    FOREIGN KEY (cod\_categorie) REFERENCES Categorie(cod\_categorie),

    FOREIGN KEY (cod\_furnizor) REFERENCES Furnizor(cod\_furnizor)

);

-- Tabela Utilizator

CREATE TABLE **Utilizator** (

    cod\_utilizator INT PRIMARY KEY,

    nume\_utilizator VARCHAR(50) NOT NULL,

    prenume VARCHAR(50),

    nume\_familie VARCHAR(50),

    cod\_locatie INT,

    FOREIGN KEY (cod\_locatie) REFERENCES Locatie(cod\_locatie)

);

-- Tabela Comanda

CREATE TABLE **Comanda** (

    cod\_comanda INT PRIMARY KEY,

    suma DECIMAL(10,2),

    cod\_utilizator INT,

    FOREIGN KEY (cod\_utilizator) REFERENCES Utilizator(cod\_utilizator)

);

-- Tabela Review

CREATE TABLE **Review** (

    cod\_review INT PRIMARY KEY,

    descriere VARCHAR(255),

    rating INT,

    cod\_produs INT,

    cod\_utilizator INT,

    FOREIGN KEY (cod\_produs) REFERENCES Produs(cod\_produs),

    FOREIGN KEY (cod\_utilizator) REFERENCES Utilizator(cod\_utilizator)

);

-- Tabela Produs\_se\_afla\_in\_Comanda

CREATE TABLE **Produs\_se\_afla\_in\_Comanda** (

    cod\_comanda INT,

    cod\_produs INT,

    PRIMARY KEY (cod\_comanda, cod\_produs),

    FOREIGN KEY (cod\_comanda) REFERENCES Comanda(cod\_comanda),

    FOREIGN KEY (cod\_produs) REFERENCES Produs(cod\_produs)

);

-- Tabela Furnizor\_distribuie\_in\_Loc

CREATE TABLE **Furnizor\_distribuie\_in\_Locati**a (

    cod\_furnizor INT,

    cod\_locatie INT,

    PRIMARY KEY (cod\_furnizor, cod\_locatie),

    FOREIGN KEY (cod\_furnizor) REFERENCES Furnizor(cod\_furnizor),

    FOREIGN KEY (cod\_locatie) REFERENCES Locatie(cod\_locatie)

);

**5. Adăugați informații coerente în tabelele create (minim 5 înregistrări pentru fiecare entitate**

**independentă; minim 10 înregistrări pentru tabela asociativă).**

**Tabela CATEGORIE:**

INSERT INTO Categorie (cod\_categorie, nume) VALUES (1, 'Electronice');

INSERT INTO Categorie (cod\_categorie, nume) VALUES (2, 'Îmbrăcăminte');

INSERT INTO Categorie (cod\_categorie, nume) VALUES (3, 'Mobila');

INSERT INTO Categorie (cod\_categorie, nume) VALUES (4, 'Cosmetice');

INSERT INTO Categorie (cod\_categorie, nume) VALUES (5, 'Alimente');

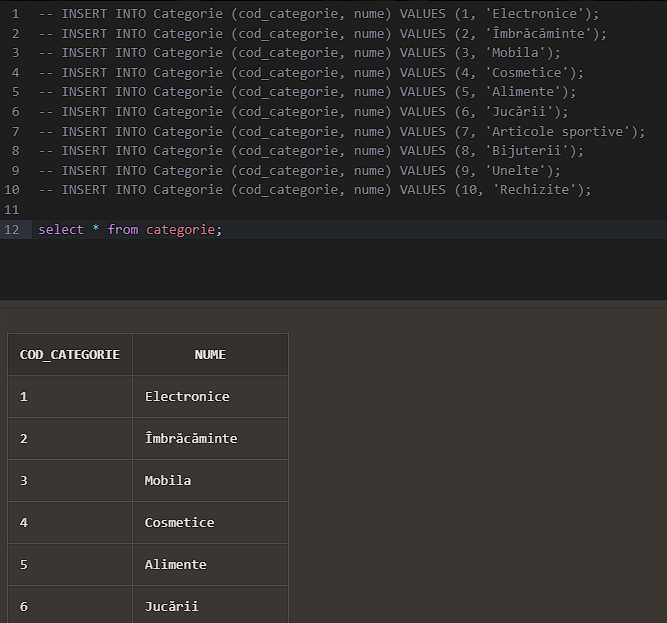
INSERT INTO Categorie (cod\_categorie, nume) VALUES (6, 'Jucării');

INSERT INTO Categorie (cod\_categorie, nume) VALUES (7, 'Articole sportive');

INSERT INTO Categorie (cod\_categorie, nume) VALUES (8, 'Bijuterii');

INSERT INTO Categorie (cod\_categorie, nume) VALUES (9, 'Unelte');

INSERT INTO Categorie (cod\_categorie, nume) VALUES (10, 'Rechizite');



**Tabela TARA:**

INSERT INTO Tara (cod\_tara, nume) VALUES (1, 'România');

INSERT INTO Tara (cod\_tara, nume) VALUES (2, 'Germania');

INSERT INTO Tara (cod\_tara, nume) VALUES (3, 'Franța');

INSERT INTO Tara (cod\_tara, nume) VALUES (4, 'Spania');

INSERT INTO Tara (cod\_tara, nume) VALUES (5, 'Italia');

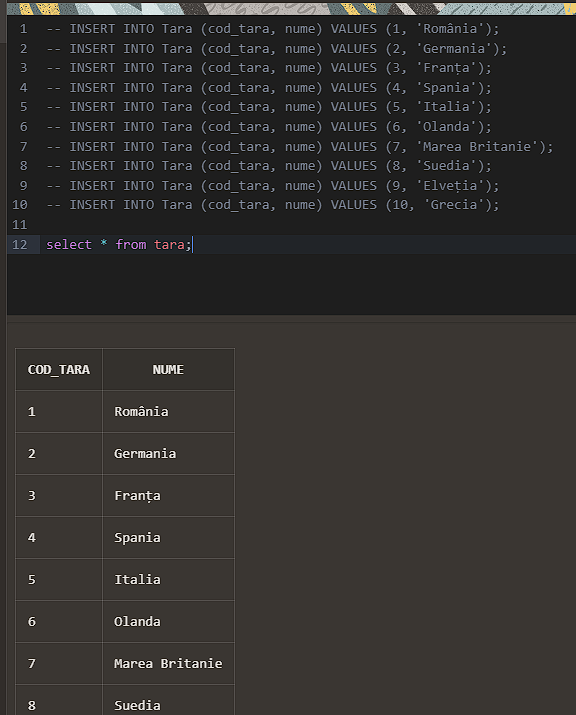
INSERT INTO Tara (cod\_tara, nume) VALUES (6, 'Olanda');

INSERT INTO Tara (cod\_tara, nume) VALUES (7, 'Marea Britanie');

INSERT INTO Tara (cod\_tara, nume) VALUES (8, 'Suedia');

INSERT INTO Tara (cod\_tara, nume) VALUES (9, 'Elveția');

INSERT INTO Tara (cod\_tara, nume) VALUES (10, 'Grecia');



**Tabela LOCATIE:**

-- Romania

INSERT INTO Locatie (cod\_locatie, strada, cod\_postal, cod\_tara) VALUES (1, 'Strada Victoriei', '010051', 1);

INSERT INTO Locatie (cod\_locatie, strada, cod\_postal, cod\_tara) VALUES (2, 'Bulevardul Unirii', '030167', 1);

-- Germania

INSERT INTO Locatie (cod\_locatie, strada, cod\_postal, cod\_tara) VALUES (3, 'Alexanderplatz', '10178', 2);

INSERT INTO Locatie (cod\_locatie, strada, cod\_postal, cod\_tara) VALUES (4, 'Kurfürstendamm', '10719', 2);

-- Franta

INSERT INTO Locatie (cod\_locatie, strada, cod\_postal, cod\_tara) VALUES (5, 'Avenue des Champs-Élysées', '75008', 3);

INSERT INTO Locatie (cod\_locatie, strada, cod\_postal, cod\_tara) VALUES (6, 'Rue de Rivoli', '75001', 3);

-- Spania

INSERT INTO Locatie (cod\_locatie, strada, cod\_postal, cod\_tara) VALUES (7, 'Paseo de la Castellana', '08046', 4);

INSERT INTO Locatie (cod\_locatie, strada, cod\_postal, cod\_tara) VALUES (8, 'Rambla de Catalunya', '08007', 4);

-- Italia

INSERT INTO Locatie (cod\_locatie, strada, cod\_postal, cod\_tara) VALUES (9, 'Via del Corso', '00187', 5);

INSERT INTO Locatie (cod\_locatie, strada, cod\_postal, cod\_tara) VALUES (10, 'Piazza San Marco', '00124', 5);

-- Olanda

INSERT INTO Locatie (cod\_locatie, strada, cod\_postal, cod\_tara) VALUES (11, 'Dam Square', '1012JS', 6);

INSERT INTO Locatie (cod\_locatie, strada, cod\_postal, cod\_tara) VALUES (12, 'Leidseplein', '1017PT', 6);

-- Marea Britanie

INSERT INTO Locatie (cod\_locatie, strada, cod\_postal, cod\_tara) VALUES (13, 'Baker Street', 'NW1 5LA', 7);

INSERT INTO Locatie (cod\_locatie, strada, cod\_postal, cod\_tara) VALUES (14, 'Oxford Street', 'W1D 1BS', 7);

-- Suedia

INSERT INTO Locatie (cod\_locatie, strada, cod\_postal, cod\_tara) VALUES (15, 'Drottninggatan', '111 21', 8);

INSERT INTO Locatie (cod\_locatie, strada, cod\_postal, cod\_tara) VALUES (16, 'Kungsgatan', '111 43', 8);

-- Elvetia

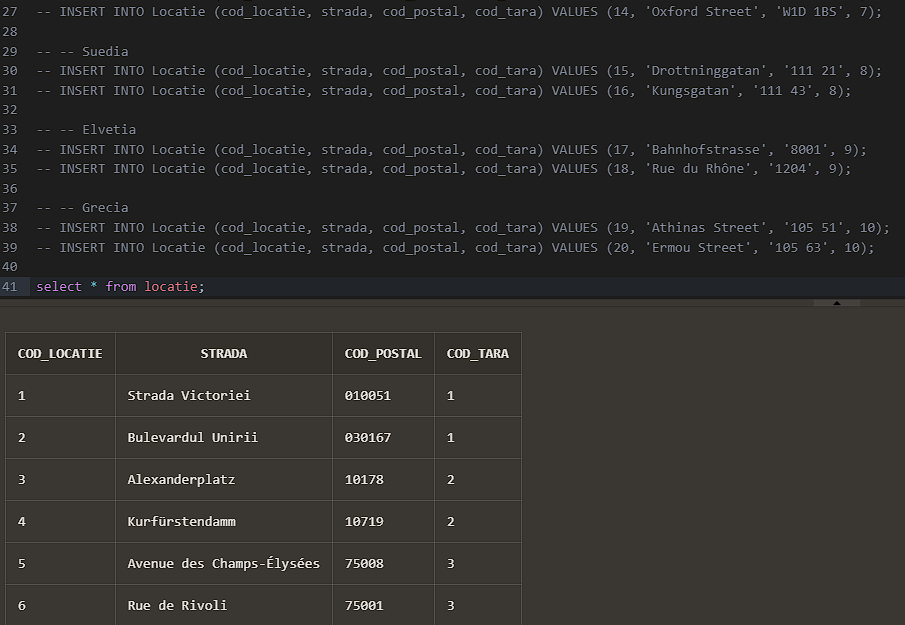
INSERT INTO Locatie (cod\_locatie, strada, cod\_postal, cod\_tara) VALUES (17, 'Bahnhofstrasse', '1001', 9);

INSERT INTO Locatie (cod\_locatie, strada, cod\_postal, cod\_tara) VALUES (18, 'Rue du Rhône', '1204', 9);

-- Grecia

INSERT INTO Locatie (cod\_locatie, strada, cod\_postal, cod\_tara) VALUES (19, 'Athinas Street', '105 51', 10);

INSERT INTO Locatie (cod\_locatie, strada, cod\_postal, cod\_tara) VALUES (20, 'Ermou Street', '105 63', 10);



**Tabela DEPARTAMENTE:**

-- Department 1

INSERT INTO Departament (cod\_departament, nume, cod\_manager) VALUES (1, 'IT', NULL);

-- Department 2

INSERT INTO Departament (cod\_departament, nume, cod\_manager) VALUES (2, 'Vânzări', NULL);

-- Department 3

INSERT INTO Departament (cod\_departament, nume, cod\_manager) VALUES (3, 'Resurse Umane', NULL);

-- Department 4

INSERT INTO Departament (cod\_departament, nume, cod\_manager) VALUES (4, 'Marketing', NULL);

-- Department 5

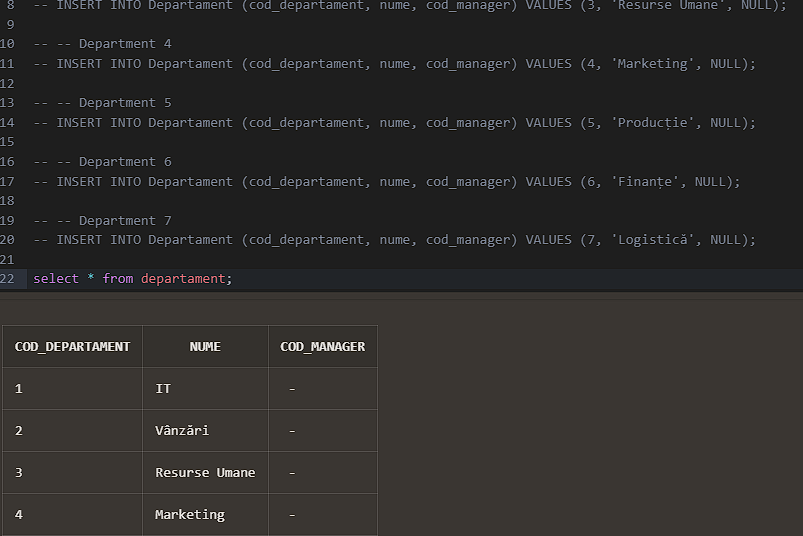
INSERT INTO Departament (cod\_departament, nume, cod\_manager) VALUES (5, 'Producție', NULL);

-- Department 6

INSERT INTO Departament (cod\_departament, nume, cod\_manager) VALUES (6, 'Finanțe', NULL);

-- Department 7

INSERT INTO Departament (cod\_departament, nume, cod\_manager) VALUES (7, 'Logistică', NULL);



-- Manager for Department 1

INSERT INTO Angajat (cod\_angajat, prenume, nume\_familie, salariu, data\_angajarii, cod\_departament, cod\_categorie, cod\_locatie, cod\_manager)

VALUES (1, 'John', 'Smith', 20000, TO\_DATE('2002-05-21', 'YYYY-MM-DD'), 1, 5, 3, NULL);

-- Manager for Department 2

INSERT INTO Angajat (cod\_angajat, prenume, nume\_familie, salariu, data\_angajarii, cod\_departament, cod\_categorie, cod\_locatie, cod\_manager)

VALUES (2, 'Maria', 'Johnson', 20500, TO\_DATE('2000-04-02', 'YYYY-MM-DD'), 2, 2, 4, NULL);

-- Manager for Department 3

INSERT INTO Angajat (cod\_angajat, prenume, nume\_familie, salariu, data\_angajarii, cod\_departament, cod\_categorie, cod\_locatie, cod\_manager)

VALUES (3, 'Robert', 'Brown', 21000, TO\_DATE('2001-10-23', 'YYYY-MM-DD'), 3, 1, 6, NULL);

-- Manager for Department 4

INSERT INTO Angajat (cod\_angajat, prenume, nume\_familie, salariu, data\_angajarii, cod\_departament, cod\_categorie, cod\_locatie, cod\_manager)

VALUES (4, 'Sophia', 'Wilson', 21500, TO\_DATE('2003-12-29', 'YYYY-MM-DD'), 4, 7, 8, NULL);

-- Manager for Department 5

INSERT INTO Angajat (cod\_angajat, prenume, nume\_familie, salariu, data\_angajarii, cod\_departament, cod\_categorie, cod\_locatie, cod\_manager)

VALUES (5, 'Daniel', 'Taylor', 22000, TO\_DATE('2000-09-05', 'YYYY-MM-DD'), 5, 3, 10, NULL);

-- Manager for Department 6

INSERT INTO Angajat (cod\_angajat, prenume, nume\_familie, salariu, data\_angajarii, cod\_departament, cod\_categorie, cod\_locatie, cod\_manager)

VALUES (6, 'Olivia', 'Miller', 22500, TO\_DATE('2001-01-30', 'YYYY-MM-DD'), 6, 6, 12, NULL);

-- Manager for Department 7

INSERT INTO Angajat (cod\_angajat, prenume, nume\_familie, salariu, data\_angajarii, cod\_departament, cod\_categorie, cod\_locatie, cod\_manager)

VALUES (7, 'James', 'Anderson', 23000, TO\_DATE('2002-03-17', 'YYYY-MM-DD'), 7, 4, 14, NULL);

**-- dupa inserare manageri**

**UPDATE Departament SET cod\_manager = cod\_departament;**

**Tabela ANGAJAT:**

-- Department 1

INSERT INTO Angajat (cod\_angajat, prenume, nume\_familie, salariu, data\_angajarii, cod\_departament, cod\_categorie, cod\_locatie, cod\_manager)

VALUES (36, 'Ion', 'Popescu', 5500, TO\_DATE('2010-05-23', 'YYYY-MM-DD'), 1, 1, 5, 1);

INSERT INTO Angajat (cod\_angajat, prenume, nume\_familie, salariu, data\_angajarii, cod\_departament, cod\_categorie, cod\_locatie, cod\_manager)

VALUES (37, 'Maria', 'Ionescu', 8000, TO\_DATE('2007-09-15', 'YYYY-MM-DD'), 1, 3, 3, 2);

INSERT INTO Angajat (cod\_angajat, prenume, nume\_familie, salariu, data\_angajarii, cod\_departament, cod\_categorie, cod\_locatie, cod\_manager)

VALUES (38, 'Alexandru', 'Stoica', 5000, TO\_DATE('2018-02-10', 'YYYY-MM-DD'), 1, 9, 4, 3);

INSERT INTO Angajat (cod\_angajat, prenume, nume\_familie, salariu, data\_angajarii, cod\_departament, cod\_categorie, cod\_locatie, cod\_manager)

VALUES (39, 'Andreea', 'Popa', 3500, TO\_DATE('2015-11-07', 'YYYY-MM-DD'), 1, 10, 1, 3);

INSERT INTO Angajat (cod\_angajat, prenume, nume\_familie, salariu, data\_angajarii, cod\_departament, cod\_categorie, cod\_locatie, cod\_manager)

VALUES (40, 'Victor', 'Dumitru', 12000, TO\_DATE('2012-04-18', 'YYYY-MM-DD'), 1, 5, 2, 1);

-- Department 2

INSERT INTO Angajat (cod\_angajat, prenume, nume\_familie, salariu, data\_angajarii, cod\_departament, cod\_categorie, cod\_locatie, cod\_manager)

VALUES (41, 'Ana', 'Baciu', 5000, TO\_DATE('2009-12-05', 'YYYY-MM-DD'), 2, 3, 13, 6);

INSERT INTO Angajat (cod\_angajat, prenume, nume\_familie, salariu, data\_angajarii, cod\_departament, cod\_categorie, cod\_locatie, cod\_manager)

VALUES (42, 'Mihai', 'Pavel', 9500, TO\_DATE('2014-07-30', 'YYYY-MM-DD'), 2, 7, 11, 7);

INSERT INTO Angajat (cod\_angajat, prenume, nume\_familie, salariu, data\_angajarii, cod\_departament, cod\_categorie, cod\_locatie, cod\_manager)

VALUES (8, 'Andrei', 'Stefan', 8000, TO\_DATE('2006-11-19', 'YYYY-MM-DD'), 2, 2, 14, 6);

INSERT INTO Angajat (cod\_angajat, prenume, nume\_familie, salariu, data\_angajarii, cod\_departament, cod\_categorie, cod\_locatie, cod\_manager)

VALUES (9, 'Mihaela', 'Coman', 3500, TO\_DATE('2017-08-12', 'YYYY-MM-DD'), 2, 8, 12, 4);

INSERT INTO Angajat (cod\_angajat, prenume, nume\_familie, salariu, data\_angajarii, cod\_departament, cod\_categorie, cod\_locatie, cod\_manager)

VALUES (10, 'George', 'Nita', 11000, TO\_DATE('2019-03-26', 'YYYY-MM-DD'), 2, 5, 15, 7);

-- Department 3

INSERT INTO Angajat (cod\_angajat, prenume, nume\_familie, salariu, data\_angajarii, cod\_departament, cod\_categorie, cod\_locatie, cod\_manager)

VALUES (11, 'Laura', 'Munteanu', 4000, TO\_DATE('2008-09-17', 'YYYY-MM-DD'), 3, 2, 1, 7);

INSERT INTO Angajat (cod\_angajat, prenume, nume\_familie, salariu, data\_angajarii, cod\_departament, cod\_categorie, cod\_locatie, cod\_manager)

VALUES (12, 'Cristian', 'Dinu', 6000, TO\_DATE('2010-06-23', 'YYYY-MM-DD'), 3, 10, 5, 4);

INSERT INTO Angajat (cod\_angajat, prenume, nume\_familie, salariu, data\_angajarii, cod\_departament, cod\_categorie, cod\_locatie, cod\_manager)

VALUES (13, 'Andreea', 'Popescu', 4500, TO\_DATE('2005-11-09', 'YYYY-MM-DD'), 3, 4, 2, 2);

INSERT INTO Angajat (cod\_angajat, prenume, nume\_familie, salariu, data\_angajarii, cod\_departament, cod\_categorie, cod\_locatie, cod\_manager)

VALUES (14, 'Ionut', 'Popa', 9000, TO\_DATE('2014-03-18', 'YYYY-MM-DD'), 3, 9, 14, 6);

INSERT INTO Angajat (cod\_angajat, prenume, nume\_familie, salariu, data\_angajarii, cod\_departament, cod\_categorie, cod\_locatie, cod\_manager)

VALUES (15, 'Elena', 'Radu', 3500, TO\_DATE('2019-08-29', 'YYYY-MM-DD'), 3, 6, 13, 3);

-- Department 4

INSERT INTO Angajat (cod\_angajat, prenume, nume\_familie, salariu, data\_angajarii, cod\_departament, cod\_categorie, cod\_locatie, cod\_manager)

VALUES (16, 'Daniel', 'Stan', 5500, TO\_DATE('2006-07-12', 'YYYY-MM-DD'), 4, 10, 2, 2);

INSERT INTO Angajat (cod\_angajat, prenume, nume\_familie, salariu, data\_angajarii, cod\_departament, cod\_categorie, cod\_locatie, cod\_manager)

VALUES (17, 'Ana', 'Popescu', 7500, TO\_DATE('2012-02-05', 'YYYY-MM-DD'), 4, 9, 11, 3);

INSERT INTO Angajat (cod\_angajat, prenume, nume\_familie, salariu, data\_angajarii, cod\_departament, cod\_categorie, cod\_locatie, cod\_manager)

VALUES (18, 'Mihai', 'Dumitru', 4000, TO\_DATE('2007-09-21', 'YYYY-MM-DD'), 4, 6, 9, 5);

INSERT INTO Angajat (cod\_angajat, prenume, nume\_familie, salariu, data\_angajarii, cod\_departament, cod\_categorie, cod\_locatie, cod\_manager)

VALUES (19, 'Andreea', 'Munteanu', 6000, TO\_DATE('2013-11-30', 'YYYY-MM-DD'), 4, 2, 5, 6);

INSERT INTO Angajat (cod\_angajat, prenume, nume\_familie, salariu, data\_angajarii, cod\_departament, cod\_categorie, cod\_locatie, cod\_manager)

VALUES (20, 'Marius', 'Ionescu', 9000, TO\_DATE('2018-05-09', 'YYYY-MM-DD'), 4, 8, 14, 1);

-- Department 5

INSERT INTO Angajat (cod\_angajat, prenume, nume\_familie, salariu, data\_angajarii, cod\_departament, cod\_categorie, cod\_locatie, cod\_manager)

VALUES (21, 'Mihai', 'Popa', 4000, TO\_DATE('2005-06-17', 'YYYY-MM-DD'), 5, 4, 9, 5);

INSERT INTO Angajat (cod\_angajat, prenume, nume\_familie, salariu, data\_angajarii, cod\_departament, cod\_categorie, cod\_locatie, cod\_manager)

VALUES (22, 'Andreea', 'Stoica', 5500, TO\_DATE('2010-10-29', 'YYYY-MM-DD'), 5, 1, 15, 1);

INSERT INTO Angajat (cod\_angajat, prenume, nume\_familie, salariu, data\_angajarii, cod\_departament, cod\_categorie, cod\_locatie, cod\_manager)

VALUES (23, 'Ionut', 'Munteanu', 7500, TO\_DATE('2008-08-12', 'YYYY-MM-DD'), 5, 2, 7, 2);

INSERT INTO Angajat (cod\_angajat, prenume, nume\_familie, salariu, data\_angajarii, cod\_departament, cod\_categorie, cod\_locatie, cod\_manager)

VALUES (24, 'Elena', 'Popescu', 6000, TO\_DATE('2013-04-24', 'YYYY-MM-DD'), 5, 3, 12, 6);

INSERT INTO Angajat (cod\_angajat, prenume, nume\_familie, salariu, data\_angajarii, cod\_departament, cod\_categorie, cod\_locatie, cod\_manager)

VALUES (25, 'Cristian', 'Dinu', 9000, TO\_DATE('2019-09-07', 'YYYY-MM-DD'), 5, 5, 6, 4);

-- Department 6

INSERT INTO Angajat (cod\_angajat, prenume, nume\_familie, salariu, data\_angajarii, cod\_departament, cod\_categorie, cod\_locatie, cod\_manager)

VALUES (26, 'Laura', 'Radu', 4000, TO\_DATE('2006-07-21', 'YYYY-MM-DD'), 6, 7, 11, 2);

INSERT INTO Angajat (cod\_angajat, prenume, nume\_familie, salariu, data\_angajarii, cod\_departament, cod\_categorie, cod\_locatie, cod\_manager)

VALUES (27, 'Cristina', 'Dumitru', 5500, TO\_DATE('2011-12-15', 'YYYY-MM-DD'), 6, 1, 19, 1);

INSERT INTO Angajat (cod\_angajat, prenume, nume\_familie, salariu, data\_angajarii, cod\_departament, cod\_categorie, cod\_locatie, cod\_manager)

VALUES (28, 'Andrei', 'Popa', 7500, TO\_DATE('2008-09-03', 'YYYY-MM-DD'), 6, 3, 14, 4);

INSERT INTO Angajat (cod\_angajat, prenume, nume\_familie, salariu, data\_angajarii, cod\_departament, cod\_categorie, cod\_locatie, cod\_manager)

VALUES (29, 'Mihaela', 'Ionescu', 6000, TO\_DATE('2014-03-11', 'YYYY-MM-DD'), 6, 4, 18, 7);

INSERT INTO Angajat (cod\_angajat, prenume, nume\_familie, salariu, data\_angajarii, cod\_departament, cod\_categorie, cod\_locatie, cod\_manager)

VALUES (30, 'George', 'Munteanu', 9000, TO\_DATE('2020-11-19', 'YYYY-MM-DD'), 6, 7, 8, 5);

-- Department 7

INSERT INTO Angajat (cod\_angajat, prenume, nume\_familie, salariu, data\_angajarii, cod\_departament, cod\_categorie, cod\_locatie, cod\_manager)

VALUES (31, 'Alexandru', 'Popescu', 4000, TO\_DATE('2011-08-22', 'yyyy-mm-dd'), 7, 2, 13, 5);

INSERT INTO Angajat (cod\_angajat, prenume, nume\_familie, salariu, data\_angajarii, cod\_departament, cod\_categorie, cod\_locatie, cod\_manager)

VALUES (32, 'Gabriela', 'Stoica', 5500, TO\_DATE('2017-12-03', 'yyyy-mm-dd'), 7, 10, 10, 4);

INSERT INTO Angajat (cod\_angajat, prenume, nume\_familie, salariu, data\_angajarii, cod\_departament, cod\_categorie, cod\_locatie, cod\_manager)

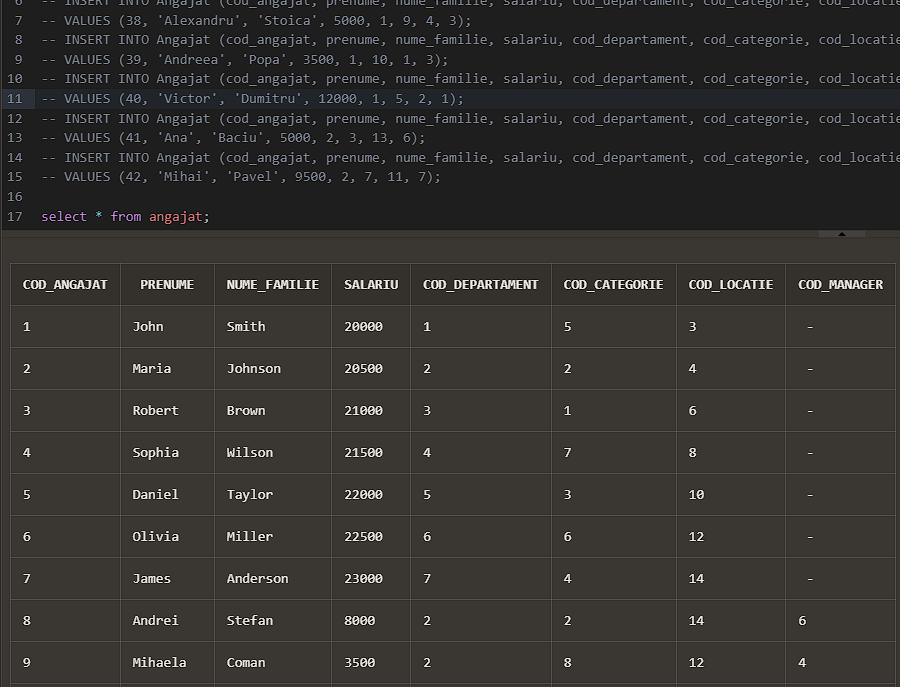
VALUES (33, 'Mihai', 'Munteanu', 7500, TO\_DATE('2013-10-18', 'yyyy-mm-dd'), 7, 3, 17, 1);

INSERT INTO Angajat (cod\_angajat, prenume, nume\_familie, salariu, data\_angajarii, cod\_departament, cod\_categorie, cod\_locatie, cod\_manager)

VALUES (34, 'Andreea', 'Popa', 6000, TO\_DATE('2016-05-07', 'yyyy-mm-dd'), 7, 8, 6, 2);

INSERT INTO Angajat (cod\_angajat, prenume, nume\_familie, salariu, data\_angajarii, cod\_departament, cod\_categorie, cod\_locatie, cod\_manager)

VALUES (35, 'Cristian', 'Dinu', 9000, TO\_DATE('2019-09-21', 'yyyy-mm-dd'), 7, 5, 3, 3);



**Tabela FURNIZOR:**

-- Furnizor for Location 1

INSERT INTO Furnizor (cod\_furnizor, nume, cod\_locatie)

VALUES (1, 'Kaufland', 1);

-- Furnizor for Location 2

INSERT INTO Furnizor (cod\_furnizor, nume, cod\_locatie)

VALUES (2, 'Carrefour', 2);

-- Furnizor for Location 3

INSERT INTO Furnizor (cod\_furnizor, nume, cod\_locatie)

VALUES (3, 'Metro', 3);

-- Furnizor for Location 4

INSERT INTO Furnizor (cod\_furnizor, nume, cod\_locatie)

VALUES (4, 'Lidl', 4);

-- Furnizor for Location 5

INSERT INTO Furnizor (cod\_furnizor, nume, cod\_locatie)

VALUES (5, 'Auchan', 5);

-- Furnizor for Location 6

INSERT INTO Furnizor (cod\_furnizor, nume, cod\_locatie)

VALUES (6, 'Mega Image', 6);

-- Furnizor for Location 7

INSERT INTO Furnizor (cod\_furnizor, nume, cod\_locatie)

VALUES (7, 'Selgros', 7);

-- Furnizor for Location 8

INSERT INTO Furnizor (cod\_furnizor, nume, cod\_locatie)

VALUES (8, 'Mega Store', 8);

-- Furnizor for Location 9

INSERT INTO Furnizor (cod\_furnizor, nume, cod\_locatie)

VALUES (9, 'Eco Market', 9);

-- Furnizor for Location 10

INSERT INTO Furnizor (cod\_furnizor, nume, cod\_locatie)

VALUES (10, 'Super Mart', 10);

-- Furnizor for Location 11

INSERT INTO Furnizor (cod\_furnizor, nume, cod\_locatie)

VALUES (11, 'Fresh Foods', 11);

-- Furnizor for Location 12

INSERT INTO Furnizor (cod\_furnizor, nume, cod\_locatie)

VALUES (12, 'Discount Depot', 12);

-- Furnizor for Location 13

INSERT INTO Furnizor (cod\_furnizor, nume, cod\_locatie)

VALUES (13, 'Market Square', 13);

-- Furnizor for Location 14

INSERT INTO Furnizor (cod\_furnizor, nume, cod\_locatie)

VALUES (14, 'Value Mart', 14);

-- Furnizor for Location 15

INSERT INTO Furnizor (cod\_furnizor, nume, cod\_locatie)

VALUES (15, 'Fresh Mart', 15);

-- Furnizor for Location 16

INSERT INTO Furnizor (cod\_furnizor, nume, cod\_locatie)

VALUES (16, 'City Market', 16);

-- Furnizor for Location 17

INSERT INTO Furnizor (cod\_furnizor, nume, cod\_locatie)

VALUES (17, 'Gourmet Express', 17);

-- Furnizor for Location 18

INSERT INTO Furnizor (cod\_furnizor, nume, cod\_locatie)

VALUES (18, 'Healthy Choices', 18);

-- Furnizor for Location 19

INSERT INTO Furnizor (cod\_furnizor, nume, cod\_locatie)

VALUES (19, 'Farmers Market', 19);

-- Furnizor for Location 20

INSERT INTO Furnizor (cod\_furnizor, nume, cod\_locatie)

VALUES (20, 'Quality Foods', 20);



**Tabela PRODUS:**

-- Produse pentru Categoria "Electronice"

INSERT INTO Produs (cod\_produs, nume, pret, cod\_categorie, cod\_furnizor)

VALUES (1, 'Laptop', 2499.99, 1, 1);

INSERT INTO Produs (cod\_produs, nume, pret, cod\_categorie, cod\_furnizor)

VALUES (2, 'Smartphone', 1299.99, 1, 6);

INSERT INTO Produs (cod\_produs, nume, pret, cod\_categorie, cod\_furnizor)

VALUES (3, 'Tabletă', 899.99, 1, 7);

INSERT INTO Produs (cod\_produs, nume, pret, cod\_categorie, cod\_furnizor)

VALUES (4, 'Televizor', 1999.99, 1, 2);

INSERT INTO Produs (cod\_produs, nume, pret, cod\_categorie, cod\_furnizor)

VALUES (5, 'Cameră foto', 799.99, 1, 3);

-- Produse pentru Categoria "Îmbrăcăminte"

INSERT INTO Produs (cod\_produs, nume, pret, cod\_categorie, cod\_furnizor)

VALUES (6, 'Cămașă', 99.99, 2, 8);

INSERT INTO Produs (cod\_produs, nume, pret, cod\_categorie, cod\_furnizor)

VALUES (7, 'Pantaloni', 79.99, 2, 7);

INSERT INTO Produs (cod\_produs, nume, pret, cod\_categorie, cod\_furnizor)

VALUES (8, 'Rochie', 149.99, 2, 19);

INSERT INTO Produs (cod\_produs, nume, pret, cod\_categorie, cod\_furnizor)

VALUES (9, 'Geacă', 199.99, 2, 2);

INSERT INTO Produs (cod\_produs, nume, pret, cod\_categorie, cod\_furnizor)

VALUES (10, 'Papuci', 49.99, 2, 5);

-- Produse pentru Categoria "Mobilă"

INSERT INTO Produs (cod\_produs, nume, pret, cod\_categorie, cod\_furnizor)

VALUES (11, 'Canapea', 999.99, 3, 3);

INSERT INTO Produs (cod\_produs, nume, pret, cod\_categorie, cod\_furnizor)

VALUES (12, 'Masă de dining', 799.99, 3, 9);

INSERT INTO Produs (cod\_produs, nume, pret, cod\_categorie, cod\_furnizor)

VALUES (13, 'Scaun de birou', 199.99, 3, 4);

INSERT INTO Produs (cod\_produs, nume, pret, cod\_categorie, cod\_furnizor)

VALUES (14, 'Dulap', 1499.99, 3, 2);

INSERT INTO Produs (cod\_produs, nume, pret, cod\_categorie, cod\_furnizor)

VALUES (15, 'Pat', 699.99, 3, 1);

-- Produse pentru Categoria "Cosmetice"

INSERT INTO Produs (cod\_produs, nume, pret, cod\_categorie, cod\_furnizor)

VALUES (16, 'Fond de ten', 59.99, 4, 12);

INSERT INTO Produs (cod\_produs, nume, pret, cod\_categorie, cod\_furnizor)

VALUES (17, 'Ruj', 19.99, 4, 14);

INSERT INTO Produs (cod\_produs, nume, pret, cod\_categorie, cod\_furnizor)

VALUES (18, 'Mascara', 29.99, 4, 13);

INSERT INTO Produs (cod\_produs, nume, pret, cod\_categorie, cod\_furnizor)

VALUES (19, 'Ser facial', 99.99, 4, 9);

INSERT INTO Produs (cod\_produs, nume, pret, cod\_categorie, cod\_furnizor)

VALUES (20, 'Parfum', 149.99, 4, 16);

-- Produse pentru Categoria "Alimente"

INSERT INTO Produs (cod\_produs, nume, pret, cod\_categorie, cod\_furnizor)

VALUES (21, 'Cafea', 19.99, 5, 8);

INSERT INTO Produs (cod\_produs, nume, pret, cod\_categorie, cod\_furnizor)

VALUES (22, 'Ciocolată', 4.99, 5, 11);

INSERT INTO Produs (cod\_produs, nume, pret, cod\_categorie, cod\_furnizor)

VALUES (23, 'Ulei de măsline', 9.99, 5, 19);

INSERT INTO Produs (cod\_produs, nume, pret, cod\_categorie, cod\_furnizor)

VALUES (24, 'Biscuiți', 2.99, 5, 16);

INSERT INTO Produs (cod\_produs, nume, pret, cod\_categorie, cod\_furnizor)

VALUES (25, 'Suc', 3.99, 5, 14);

-- Produse pentru Categoria "Jucării"

INSERT INTO Produs (cod\_produs, nume, pret, cod\_categorie, cod\_furnizor)

VALUES (26, 'Păpușă', 29.99, 6, 20);

INSERT INTO Produs (cod\_produs, nume, pret, cod\_categorie, cod\_furnizor)

VALUES (27, 'Mașinuță', 24.99, 6, 9);

INSERT INTO Produs (cod\_produs, nume, pret, cod\_categorie, cod\_furnizor)

VALUES (28, 'Puzzle', 14.99, 6, 15);

INSERT INTO Produs (cod\_produs, nume, pret, cod\_categorie, cod\_furnizor)

VALUES (29, 'Lego', 49.99, 6, 10);

INSERT INTO Produs (cod\_produs, nume, pret, cod\_categorie, cod\_furnizor)

VALUES (30, 'Joc de societate', 39.99, 6, 17);

-- Produse pentru Categoria "Articole"

INSERT INTO Produs (cod\_produs, nume, pret, cod\_categorie, cod\_furnizor)

VALUES (31, 'Caiet', 4.99, 7, 3);

INSERT INTO Produs (cod\_produs, nume, pret, cod\_categorie, cod\_furnizor)

VALUES (32, 'Pix', 1.99, 7, 8);

INSERT INTO Produs (cod\_produs, nume, pret, cod\_categorie, cod\_furnizor)

VALUES (33, 'Adeziv', 2.99, 7, 5);

INSERT INTO Produs (cod\_produs, nume, pret, cod\_categorie, cod\_furnizor)

VALUES (34, 'Creion colorat', 0.99, 7, 7);

INSERT INTO Produs (cod\_produs, nume, pret, cod\_categorie, cod\_furnizor)

VALUES (35, 'Foarfecă', 3.99, 7, 16);

-- Produse pentru Categoria "Bijuterii"

INSERT INTO Produs (cod\_produs, nume, pret, cod\_categorie, cod\_furnizor)

VALUES (36, 'Inel', 49.99, 8, 8);

INSERT INTO Produs (cod\_produs, nume, pret, cod\_categorie, cod\_furnizor)

VALUES (37, 'Cercei', 39.99, 8, 2);

INSERT INTO Produs (cod\_produs, nume, pret, cod\_categorie, cod\_furnizor)

VALUES (38, 'Colier', 59.99, 8, 19);

INSERT INTO Produs (cod\_produs, nume, pret, cod\_categorie, cod\_furnizor)

VALUES (39, 'Brățară', 29.99, 8, 1);

INSERT INTO Produs (cod\_produs, nume, pret, cod\_categorie, cod\_furnizor)

VALUES (40, 'Ceas', 99.99, 8, 14);

-- Produse pentru Categoria "Unelte"

INSERT INTO Produs (cod\_produs, nume, pret, cod\_categorie, cod\_furnizor)

VALUES (41, 'Șurubelniță', 9.99, 9, 20);

INSERT INTO Produs (cod\_produs, nume, pret, cod\_categorie, cod\_furnizor)

VALUES (42, 'Foarfecă de grădină', 14.99, 9, 9);

INSERT INTO Produs (cod\_produs, nume, pret, cod\_categorie, cod\_furnizor)

VALUES (43, 'Ciocan', 19.99, 9, 15);

INSERT INTO Produs (cod\_produs, nume, pret, cod\_categorie, cod\_furnizor)

VALUES (44, 'Ruletă', 7.99, 9, 10);

INSERT INTO Produs (cod\_produs, nume, pret, cod\_categorie, cod\_furnizor)

VALUES (45, 'Mașină de găurit', 99.99, 9, 17);

-- Produse pentru Categoria "Rechizite"

INSERT INTO Produs (cod\_produs, nume, pret, cod\_categorie, cod\_furnizor)

VALUES (46, 'Penar', 9.99, 10, 8);

INSERT INTO Produs (cod\_produs, nume, pret, cod\_categorie, cod\_furnizor)

VALUES (47, 'Creion HB', 0.99, 10, 11);

INSERT INTO Produs (cod\_produs, nume, pret, cod\_categorie, cod\_furnizor)

VALUES (48, 'Radieră', 1.99, 10, 19);

INSERT INTO Produs (cod\_produs, nume, pret, cod\_categorie, cod\_furnizor)

VALUES (49, 'Caiet dictando', 2.99, 10, 16);

INSERT INTO Produs (cod\_produs, nume, pret, cod\_categorie, cod\_furnizor)

VALUES (50, 'Stilou', 4.99, 10, 14);



**Tabela UTILIZATOR:**

-- Utilizatori

INSERT INTO Utilizator (cod\_utilizator, nume\_utilizator, prenume, nume\_familie, cod\_locatie)

VALUES (1, 'popescu\_1', 'Alexandra', 'Popescu', 1);

INSERT INTO Utilizator (cod\_utilizator, nume\_utilizator, prenume, nume\_familie, cod\_locatie)

VALUES (2, 'ionescu\_2', 'Andrei', 'Ionescu', 2);

INSERT INTO Utilizator (cod\_utilizator, nume\_utilizator, prenume, nume\_familie, cod\_locatie)

VALUES (3, 'vasilescu\_3', 'Maria', 'Vasilescu', 3);

INSERT INTO Utilizator (cod\_utilizator, nume\_utilizator, prenume, nume\_familie, cod\_locatie)

VALUES (4, 'georgescu\_4', 'Mihai', 'Georgescu', 4);

INSERT INTO Utilizator (cod\_utilizator, nume\_utilizator, prenume, nume\_familie, cod\_locatie)

VALUES (5, 'popa\_5', 'Elena', 'Popa', 5);

INSERT INTO Utilizator (cod\_utilizator, nume\_utilizator, prenume, nume\_familie, cod\_locatie)

VALUES (6, 'dumitru\_6', 'Andreea', 'Dumitru', 6);

INSERT INTO Utilizator (cod\_utilizator, nume\_utilizator, prenume, nume\_familie, cod\_locatie)

VALUES (7, 'stoica\_7', 'Adrian', 'Stoica', 7);

INSERT INTO Utilizator (cod\_utilizator, nume\_utilizator, prenume, nume\_familie, cod\_locatie)

VALUES (8, 'constantin\_8', 'Cristina', 'Constantin', 8);

INSERT INTO Utilizator (cod\_utilizator, nume\_utilizator, prenume, nume\_familie, cod\_locatie)

VALUES (9, 'ilie\_9', 'Gabriel', 'Ilie', 9);

INSERT INTO Utilizator (cod\_utilizator, nume\_utilizator, prenume, nume\_familie, cod\_locatie)

VALUES (10, 'dobre\_10', 'Andrei', 'Dobre', 10);

INSERT INTO Utilizator (cod\_utilizator, nume\_utilizator, prenume, nume\_familie, cod\_locatie)

VALUES (11, 'popovici\_11', 'Diana', 'Popovici', 11);

INSERT INTO Utilizator (cod\_utilizator, nume\_utilizator, prenume, nume\_familie, cod\_locatie)

VALUES (12, 'marinescu\_12', 'Ionut', 'Marinescu', 12);

INSERT INTO Utilizator (cod\_utilizator, nume\_utilizator, prenume, nume\_familie, cod\_locatie)

VALUES (13, 'mihai\_13', 'Andreea', 'Mihai', 13);

INSERT INTO Utilizator (cod\_utilizator, nume\_utilizator, prenume, nume\_familie, cod\_locatie)

VALUES (14, 'sandu\_14', 'Ana', 'Sandu', 14);

INSERT INTO Utilizator (cod\_utilizator, nume\_utilizator, prenume, nume\_familie, cod\_locatie)

VALUES (15, 'radu\_15', 'Marius', 'Radu', 15);

INSERT INTO Utilizator (cod\_utilizator, nume\_utilizator, prenume, nume\_familie, cod\_locatie)

VALUES (16, 'petrescu\_16', 'Elena', 'Petrescu', 16);

INSERT INTO Utilizator (cod\_utilizator, nume\_utilizator, prenume, nume\_familie, cod\_locatie)

VALUES (17, 'stan\_17', 'Gabriel', 'Stan', 17);

INSERT INTO Utilizator (cod\_utilizator, nume\_utilizator, prenume, nume\_familie, cod\_locatie)

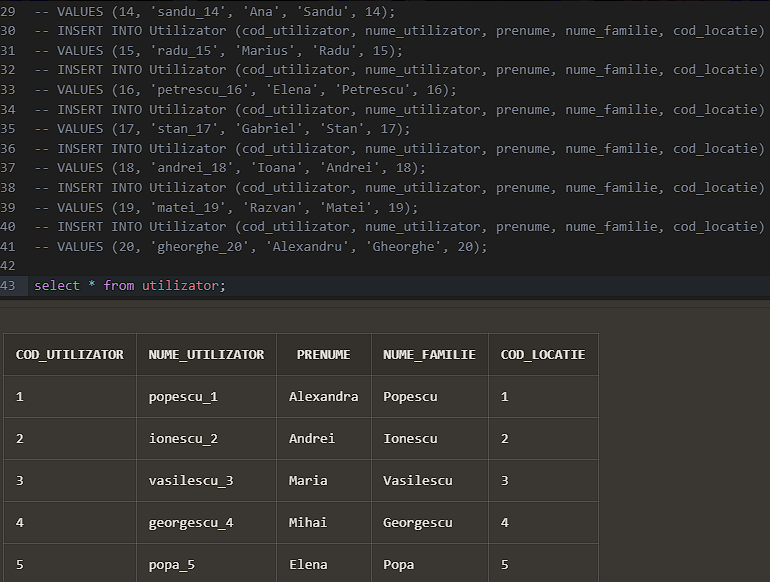
VALUES (18, 'andrei\_18', 'Ioana', 'Andrei', 18);

INSERT INTO Utilizator (cod\_utilizator, nume\_utilizator, prenume, nume\_familie, cod\_locatie)

VALUES (19, 'matei\_19', 'Razvan', 'Matei', 19);

INSERT INTO Utilizator (cod\_utilizator, nume\_utilizator, prenume, nume\_familie, cod\_locatie)

VALUES (20, 'gheorghe\_20', 'Alexandru', 'Gheorghe', 20);



**Tabela COMANDA:**

-- Comenzi

INSERT INTO Comanda (cod\_comanda, discount, cod\_utilizator)

VALUES (1, 0.25, 5);

INSERT INTO Comanda (cod\_comanda, discount, cod\_utilizator)

VALUES (2, 0.12, 17);

INSERT INTO Comanda (cod\_comanda, discount, cod\_utilizator)

VALUES (3, 0.45, 9);

INSERT INTO Comanda (cod\_comanda, discount, cod\_utilizator)

VALUES (4, 0.33, 12);

INSERT INTO Comanda (cod\_comanda, discount, cod\_utilizator)

VALUES (5, 0.58, 4);

INSERT INTO Comanda (cod\_comanda, discount, cod\_utilizator)

VALUES (6, 0.27, 19);

INSERT INTO Comanda (cod\_comanda, discount, cod\_utilizator)

VALUES (7, 0.61, 3);

INSERT INTO Comanda (cod\_comanda, discount, cod\_utilizator)

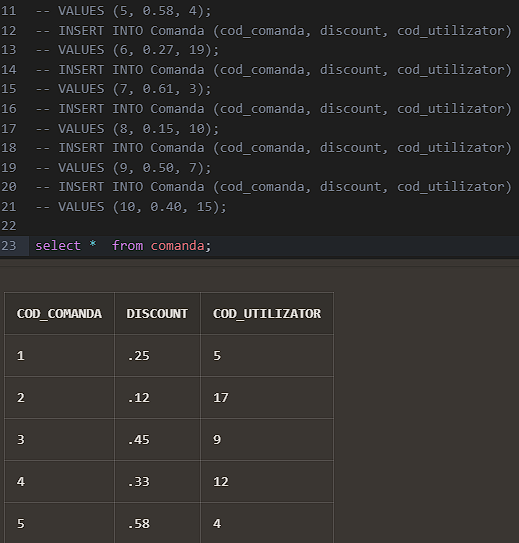
VALUES (8, 0.15, 10);

INSERT INTO Comanda (cod\_comanda, discount, cod\_utilizator)

VALUES (9, 0.50, 7);

INSERT INTO Comanda (cod\_comanda, discount, cod\_utilizator)

VALUES (10, 0.40, 15);



**Tabela REVIEW:**

-- Reviews

INSERT INTO Review (cod\_review, descriere, rating, cod\_produs, cod\_utilizator)

VALUES (1, 'Bun produs!', 4, 15, 8);

INSERT INTO Review (cod\_review, descriere, rating, cod\_produs, cod\_utilizator)

VALUES (2, 'Slab produs...', 2, 27, 12);

INSERT INTO Review (cod\_review, descriere, rating, cod\_produs, cod\_utilizator)

VALUES (3, 'Produs excelent!', 5, 6, 3);

INSERT INTO Review (cod\_review, descriere, rating, cod\_produs, cod\_utilizator)

VALUES (4, 'Nu recomand...', 1, 42, 19);

INSERT INTO Review (cod\_review, descriere, rating, cod\_produs, cod\_utilizator)

VALUES (5, 'Super calitate!', 5, 10, 5);

INSERT INTO Review (cod\_review, descriere, rating, cod\_produs, cod\_utilizator)

VALUES (6, 'Mai mult decat ma asteptam.', 4, 35, 11);

INSERT INTO Review (cod\_review, descriere, rating, cod\_produs, cod\_utilizator)

VALUES (7, 'Dezamagitor...', 2, 20, 2);

INSERT INTO Review (cod\_review, descriere, rating, cod\_produs, cod\_utilizator)

VALUES (8, 'Recomand cu incredere!', 5, 17, 14);

INSERT INTO Review (cod\_review, descriere, rating, cod\_produs, cod\_utilizator)

VALUES (9, 'Calitate medie.', 3, 8, 9);

INSERT INTO Review (cod\_review, descriere, rating, cod\_produs, cod\_utilizator)

VALUES (10, 'Foarte rau...', 1, 49, 16);

INSERT INTO Review (cod\_review, descriere, rating, cod\_produs, cod\_utilizator)

VALUES (11, 'Produs ok.', 3, 12, 7);

INSERT INTO Review (cod\_review, descriere, rating, cod\_produs, cod\_utilizator)

VALUES (12, 'Foarte bun raport calitate-pret.', 4, 33, 13);

INSERT INTO Review (cod\_review, descriere, rating, cod\_produs, cod\_utilizator)

VALUES (13, 'Nu a meritat investitia...', 2, 24, 18);

INSERT INTO Review (cod\_review, descriere, rating, cod\_produs, cod\_utilizator)

VALUES (14, 'Satisfacut de produs.', 3, 9, 4);

INSERT INTO Review (cod\_review, descriere, rating, cod\_produs, cod\_utilizator)

VALUES (15, 'Slab calitate.', 2, 38, 15);

INSERT INTO Review (cod\_review, descriere, rating, cod\_produs, cod\_utilizator)

VALUES (16, 'Excelent produs!', 5, 14, 10);

INSERT INTO Review (cod\_review, descriere, rating, cod\_produs, cod\_utilizator)

VALUES (17, 'Nu recomand acest produs...', 1, 45, 17);

INSERT INTO Review (cod\_review, descriere, rating, cod\_produs, cod\_utilizator)

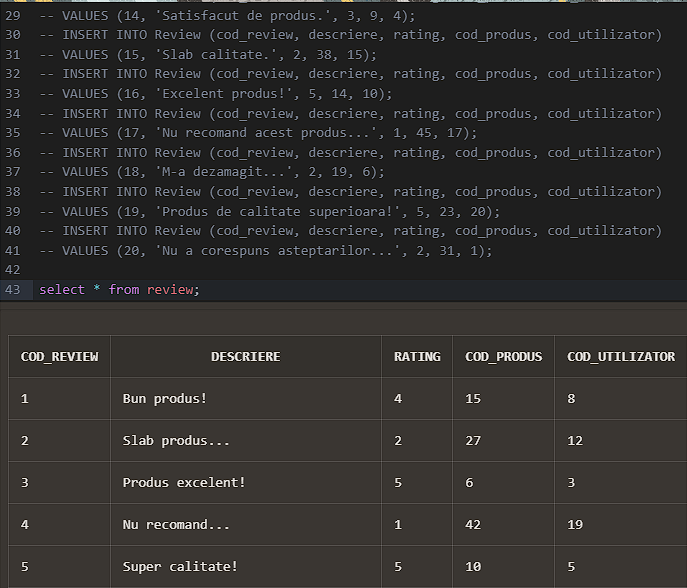
VALUES (18, 'M-a dezamagit...', 2, 19, 6);

INSERT INTO Review (cod\_review, descriere, rating, cod\_produs, cod\_utilizator)

VALUES (19, 'Produs de calitate superioara!', 5, 4, 20);

INSERT INTO Review (cod\_review, descriere, rating, cod\_produs, cod\_utilizator)

VALUES (20, 'Nu a corespuns asteptarilor...', 2, 31, 1);



**Tabela PRODUS\_se\_afla\_in\_COMANDA:**

-- Produs\_se\_afla\_in\_Comanda

--1

INSERT INTO Produs\_se\_afla\_in\_Comanda (cod\_comanda, cod\_produs) VALUES (1, 4);

INSERT INTO Produs\_se\_afla\_in\_Comanda (cod\_comanda, cod\_produs) VALUES (1, 9);

INSERT INTO Produs\_se\_afla\_in\_Comanda (cod\_comanda, cod\_produs) VALUES (1, 14);

--2

INSERT INTO Produs\_se\_afla\_in\_Comanda (cod\_comanda, cod\_produs) VALUES (2, 21);

INSERT INTO Produs\_se\_afla\_in\_Comanda (cod\_comanda, cod\_produs) VALUES (2, 27);

INSERT INTO Produs\_se\_afla\_in\_Comanda (cod\_comanda, cod\_produs) VALUES (2, 4);

--3

INSERT INTO Produs\_se\_afla\_in\_Comanda (cod\_comanda, cod\_produs) VALUES (3, 4);

INSERT INTO Produs\_se\_afla\_in\_Comanda (cod\_comanda, cod\_produs) VALUES (3, 15);

INSERT INTO Produs\_se\_afla\_in\_Comanda (cod\_comanda, cod\_produs) VALUES (3, 25);

--4

INSERT INTO Produs\_se\_afla\_in\_Comanda (cod\_comanda, cod\_produs) VALUES (4, 1);

INSERT INTO Produs\_se\_afla\_in\_Comanda (cod\_comanda, cod\_produs) VALUES (4, 7);

INSERT INTO Produs\_se\_afla\_in\_Comanda (cod\_comanda, cod\_produs) VALUES (4, 4);

--5

INSERT INTO Produs\_se\_afla\_in\_Comanda (cod\_comanda, cod\_produs) VALUES (5, 18);

INSERT INTO Produs\_se\_afla\_in\_Comanda (cod\_comanda, cod\_produs) VALUES (5, 4);

INSERT INTO Produs\_se\_afla\_in\_Comanda (cod\_comanda, cod\_produs) VALUES (5, 33);

--6

INSERT INTO Produs\_se\_afla\_in\_Comanda (cod\_comanda, cod\_produs) VALUES (6, 2);

INSERT INTO Produs\_se\_afla\_in\_Comanda (cod\_comanda, cod\_produs) VALUES (6, 11);

INSERT INTO Produs\_se\_afla\_in\_Comanda (cod\_comanda, cod\_produs) VALUES (6, 4);

--7

INSERT INTO Produs\_se\_afla\_in\_Comanda (cod\_comanda, cod\_produs) VALUES (7, 4);

INSERT INTO Produs\_se\_afla\_in\_Comanda (cod\_comanda, cod\_produs) VALUES (7, 13);

INSERT INTO Produs\_se\_afla\_in\_Comanda (cod\_comanda, cod\_produs) VALUES (7, 24);

--8

INSERT INTO Produs\_se\_afla\_in\_Comanda (cod\_comanda, cod\_produs) VALUES (8, 10);

INSERT INTO Produs\_se\_afla\_in\_Comanda (cod\_comanda, cod\_produs) VALUES (8, 4);

INSERT INTO Produs\_se\_afla\_in\_Comanda (cod\_comanda, cod\_produs) VALUES (8, 28);

--9

INSERT INTO Produs\_se\_afla\_in\_Comanda (cod\_comanda, cod\_produs) VALUES (9, 9);

INSERT INTO Produs\_se\_afla\_in\_Comanda (cod\_comanda, cod\_produs) VALUES (9, 4);

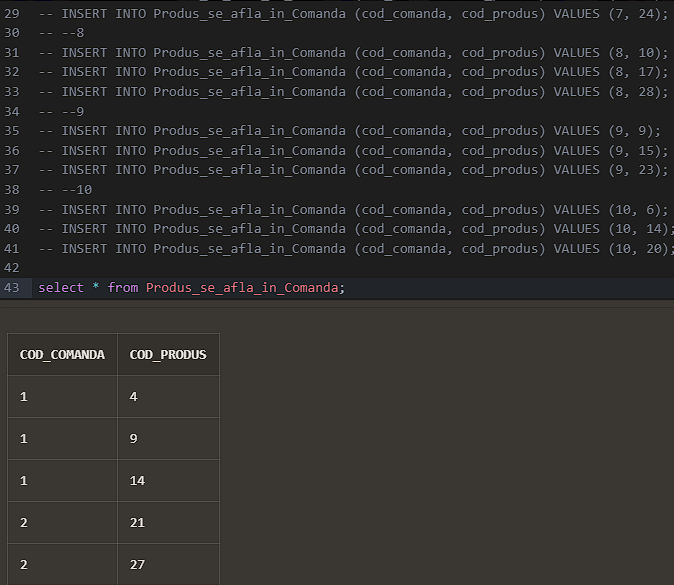
INSERT INTO Produs\_se\_afla\_in\_Comanda (cod\_comanda, cod\_produs) VALUES (9, 23);

--10

INSERT INTO Produs\_se\_afla\_in\_Comanda (cod\_comanda, cod\_produs) VALUES (10, 6);

INSERT INTO Produs\_se\_afla\_in\_Comanda (cod\_comanda, cod\_produs) VALUES (10, 4);

INSERT INTO Produs\_se\_afla\_in\_Comanda (cod\_comanda, cod\_produs) VALUES (10, 20);



**Tabela FURNIZOR\_distribuie\_in\_LOCATIA:**

-- Furnizor\_distribuie\_in\_Locatia

INSERT INTO Furnizor\_distribuie\_in\_Locatia (cod\_furnizor, cod\_locatie) VALUES (1, 1);

INSERT INTO Furnizor\_distribuie\_in\_Locatia (cod\_furnizor, cod\_locatie) VALUES (2, 2);

INSERT INTO Furnizor\_distribuie\_in\_Locatia (cod\_furnizor, cod\_locatie) VALUES (3, 3);

INSERT INTO Furnizor\_distribuie\_in\_Locatia (cod\_furnizor, cod\_locatie) VALUES (4, 4);

INSERT INTO Furnizor\_distribuie\_in\_Locatia (cod\_furnizor, cod\_locatie) VALUES (5, 5);

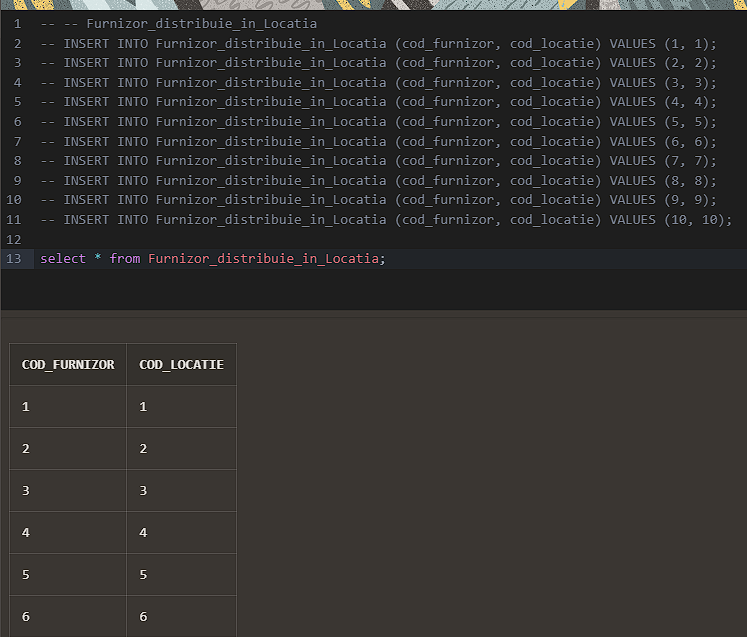
INSERT INTO Furnizor\_distribuie\_in\_Locatia (cod\_furnizor, cod\_locatie) VALUES (6, 6);

INSERT INTO Furnizor\_distribuie\_in\_Locatia (cod\_furnizor, cod\_locatie) VALUES (7, 7);

INSERT INTO Furnizor\_distribuie\_in\_Locatia (cod\_furnizor, cod\_locatie) VALUES (8, 8);

INSERT INTO Furnizor\_distribuie\_in\_Locatia (cod\_furnizor, cod\_locatie) VALUES (9, 9);

INSERT INTO Furnizor\_distribuie\_in\_Locatia (cod\_furnizor, cod\_locatie) VALUES (10, 10);



1. **Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat independent care să utilizeze toate cele 3 tipuri de colecții studiate. Apelați subprogramul.**

**Sa se afiseze pentru fiecare utilizator comenzile sale, respectiv pentru fiecare comanda, numele, codul si pretul produsului din comanda respectiva.**

CREATE OR REPLACE PROCEDURE display\_user\_orders

IS

TYPE produs\_record IS RECORD (

cod\_produs produs.cod\_produs%TYPE,

pret produs.pret%TYPE,

nume produs.nume%TYPE

);

TYPE vector\_produse IS VARRAY(200) OF produs\_record;

TYPE comanda\_cu\_produse\_record IS RECORD (

cod\_comanda comanda.cod\_comanda%TYPE,

produse vector\_produse

);

TYPE comanda\_cu\_produse\_nested\_table IS TABLE OF comanda\_cu\_produse\_record;

TYPE utilizator\_comanda\_indexed\_table IS TABLE OF comanda\_cu\_produse\_nested\_table INDEX BY PLS\_INTEGER;

utilizatori\_cu\_comenzi utilizator\_comanda\_indexed\_table;

BEGIN

FOR utilizator IN (SELECT DISTINCT cod\_utilizator FROM comanda)

LOOP

utilizatori\_cu\_comenzi(utilizator.cod\_utilizator) := comanda\_cu\_produse\_nested\_table();

DBMS\_OUTPUT.PUT\_LINE('Utilizator: ' || utilizator.cod\_utilizator);

FOR comanda IN (SELECT cod\_comanda FROM comanda WHERE cod\_utilizator = utilizator.cod\_utilizator)

LOOP

DECLARE

produse vector\_produse := vector\_produse();

BEGIN

FOR produs IN (SELECT p.cod\_produs, p.pret, p.nume

FROM produs p, produs\_se\_afla\_in\_comanda pc

WHERE p.cod\_produs = pc.cod\_produs

AND pc.cod\_comanda = comanda.cod\_comanda)

LOOP

produse.extend();

produse(produse.last) := produs\_record(

cod\_produs => produs.cod\_produs,

pret => produs.pret,

nume => produs.nume

);

END LOOP;

utilizatori\_cu\_comenzi(utilizator.cod\_utilizator).EXTEND;

utilizatori\_cu\_comenzi(utilizator.cod\_utilizator)(utilizatori\_cu\_comenzi(utilizator.cod\_utilizator).LAST) :=

comanda\_cu\_produse\_record(

cod\_comanda => comanda.cod\_comanda,

produse => produse

);

DBMS\_OUTPUT.PUT\_LINE(' Comanda: ' || comanda.cod\_comanda);

FOR i IN 1..produse.count

LOOP

DBMS\_OUTPUT.PUT\_LINE(' Produs: ' || produse(i).cod\_produs ||

', Nume: ' || produse(i).nume ||

', Pret: ' || produse(i).pret);

END LOOP;

END;

END LOOP;

DBMS\_OUTPUT.PUT\_LINE('-------------');

DBMS\_OUTPUT.NEW\_LINE;

END LOOP;

END display\_user\_orders;

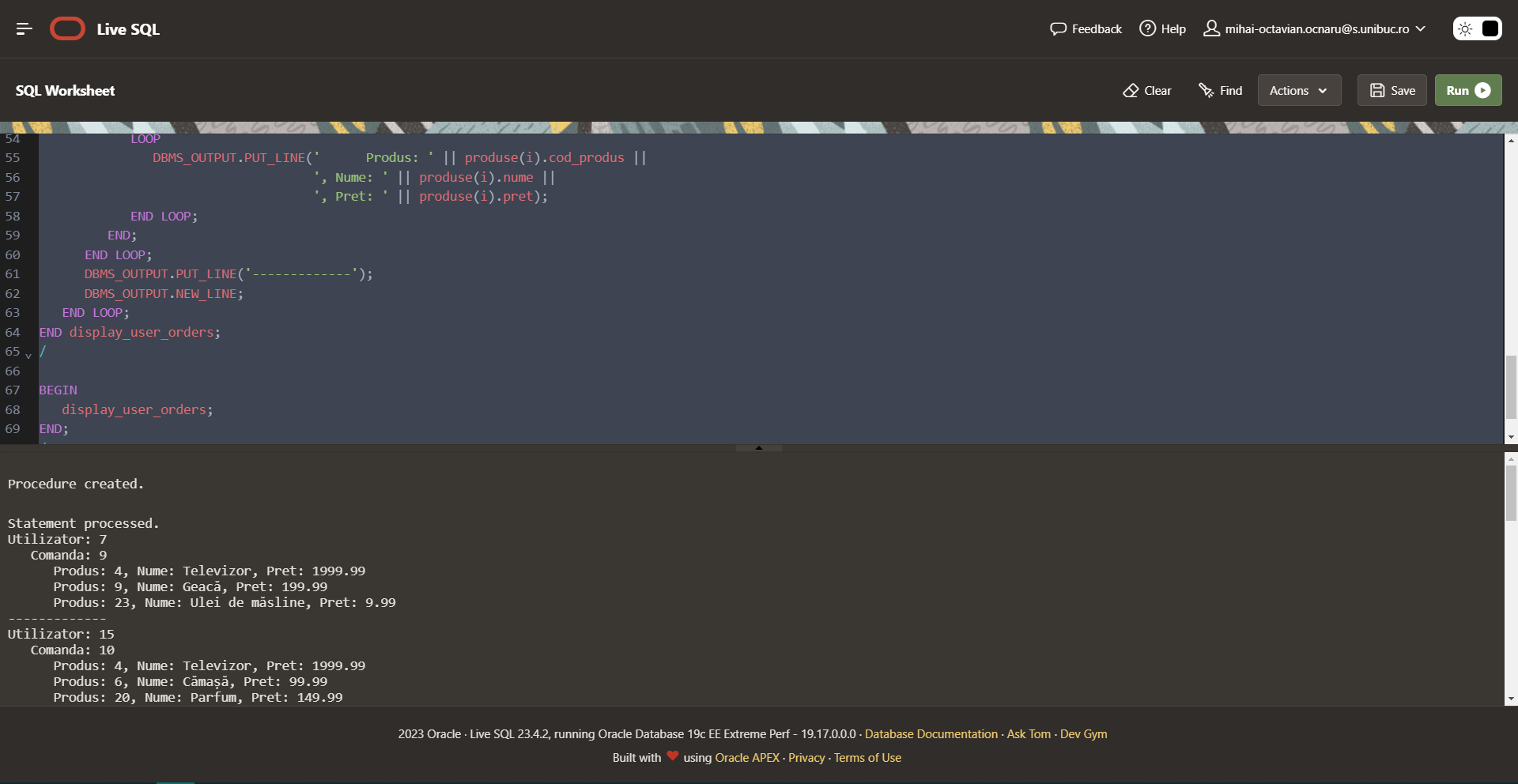
/

BEGIN

display\_user\_orders;

END;

/



1. **Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat independent care să utilizeze 2 tipuri diferite de cursoare studiate, unul dintre acestea fiind cursor parametrizat, dependent de celălalt cursor. Apelați subprogramul.**

**Sa afiseze numele, pretul, reviurile si recenziile produselor care se afla intr-o comanda plasata de un utilizator dintr-o tara latina.**

CREATE OR REPLACE PROCEDURE raport\_produse\_recenzii IS

CURSOR curson\_produse IS

SELECT distinct p.cod\_produs, p.nume, p.pret

FROM produs p, produs\_se\_afla\_in\_comanda pc, comanda c

where pc.cod\_produs = p.cod\_produs

and pc.cod\_comanda = c.cod\_comanda

and c.cod\_utilizator in (select distinct u.cod\_utilizator

from locatie l, utilizator u, tara t

where l.cod\_locatie = u.cod\_locatie

and l.cod\_tara = t.cod\_tara

and initcap(t.nume) in ('România', 'Franța', 'Spania', 'Italia'));

CURSOR curson\_recenzii (v\_cod\_produs INT) IS

SELECT descriere, rating

FROM Review

WHERE cod\_produs = v\_cod\_produs;

v\_cod\_produs Produs.cod\_produs%TYPE;

v\_nume\_produs Produs.nume%TYPE;

v\_pret Produs.pret%TYPE;

v\_descriere Review.descriere%TYPE;

v\_rating Review.rating%TYPE;

BEGIN

OPEN curson\_produse;

LOOP

FETCH curson\_produse INTO v\_cod\_produs, v\_nume\_produs, v\_pret;

EXIT WHEN curson\_produse%NOTFOUND;

OPEN curson\_recenzii(v\_cod\_produs);

LOOP

FETCH curson\_recenzii INTO v\_descriere, v\_rating;

EXIT WHEN curson\_recenzii%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE('Produs: ' || v\_nume\_produs || ', Pret: ' || v\_pret);

DBMS\_OUTPUT.PUT\_LINE(' Recenzie: ' || v\_descriere || ', Rating: ' || v\_rating);

DBMS\_OUTPUT.PUT\_LINE('------');

DBMS\_OUTPUT.NEW\_LINE;

END LOOP;

CLOSE curson\_recenzii;

END LOOP;

CLOSE curson\_produse;

END raport\_produse\_recenzii;

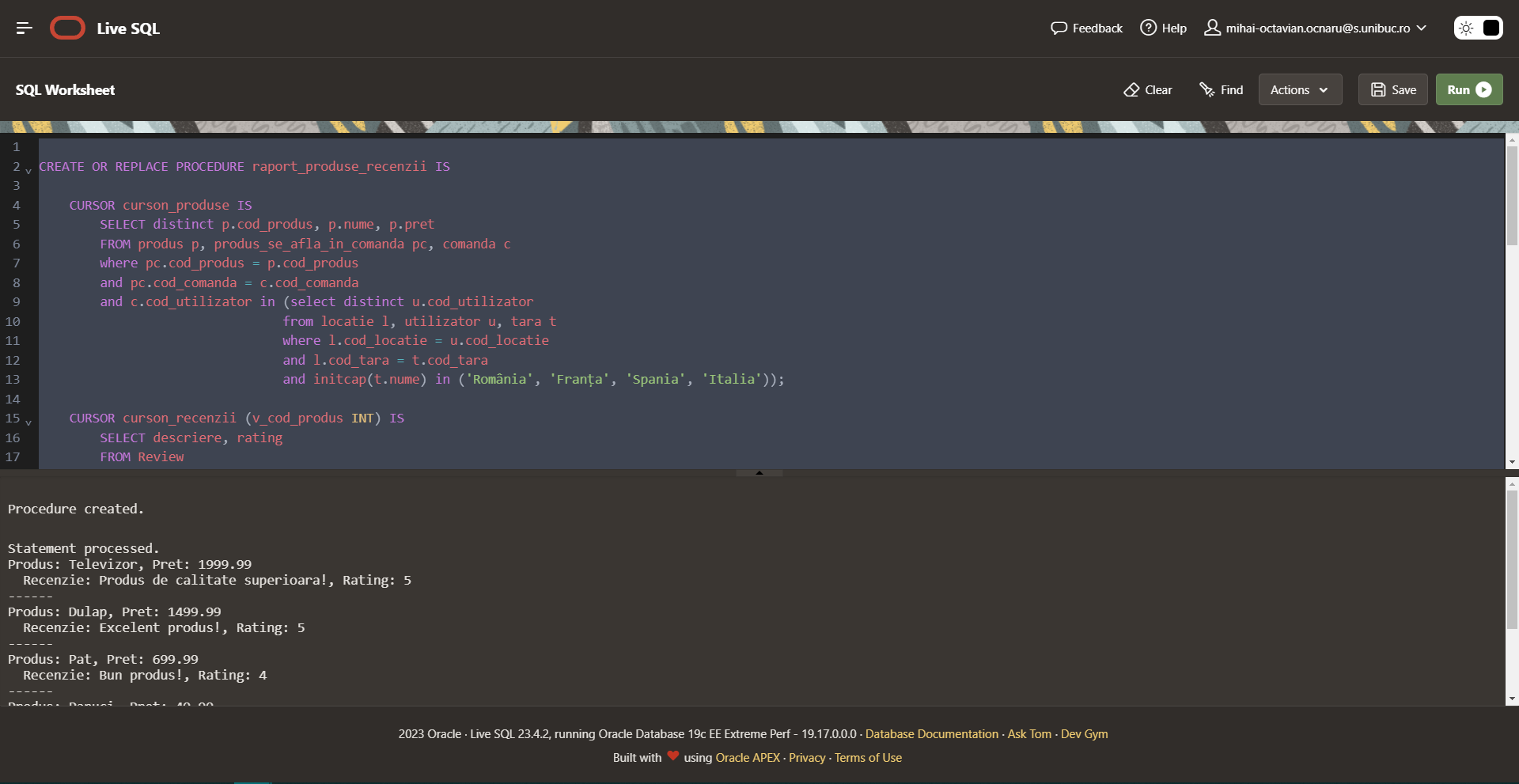
/

BEGIN

raport\_produse\_recenzii;

END;

/



1. **Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat independent de tip funcție care să utilizeze într-o singură comandă SQL 3 dintre tabelele definite. Definiți minim 2 excepții proprii. Apelați subprogramul astfel încât să evidențiați toate cazurile definite și tratate.**

**Sa se afiseze un raport care sa afiseze numele si pretul produselor livrate de un furnizor dat, intr-o anumita tara si sub o anumita limita de pret. In cazul in care tara nu exista sau pretul este prea mic se vor afisa erori adecvate.**

CREATE OR REPLACE FUNCTION raport\_produse\_tara\_categorie(

v\_nume\_tara IN VARCHAR2,

v\_nume\_categorie IN VARCHAR2,

v\_max\_pret IN NUMBER

) RETURN VARCHAR2

IS

v\_raport VARCHAR2(4000);

invalid\_country\_name EXCEPTION;

PRAGMA EXCEPTION\_INIT(invalid\_country\_name, -1333);

too\_small\_price EXCEPTION;

PRAGMA EXCEPTION\_INIT(too\_small\_price, -1332);

BEGIN

v\_raport := 'Products in ' || v\_nume\_tara || ', Category: ' || v\_nume\_categorie || ' with price <= ' || v\_max\_pret || ':';

DECLARE

min\_price produs.pret%TYPE;

BEGIN

select min(p.pret)

into min\_price

FROM tara t, locatie l, furnizor f, furnizor\_distribuie\_in\_locatia fl, produs p

WHERE t.cod\_tara = l.cod\_tara

AND f.cod\_furnizor = fl.cod\_furnizor

AND fl.cod\_locatie = l.cod\_locatie

AND f.cod\_furnizor = p.cod\_furnizor;

if min\_price > v\_max\_pret then

RAISE too\_small\_price;

end if;

END;

DECLARE

flag Boolean := false;

BEGIN

FOR country IN (

SELECT 1

FROM dual

WHERE EXISTS (

SELECT 1

FROM tara t, locatie l, furnizor f, furnizor\_distribuie\_in\_locatia fl

WHERE t.cod\_tara = l.cod\_tara

AND f.cod\_furnizor = fl.cod\_furnizor

AND fl.cod\_locatie = l.cod\_locatie

AND t.nume = v\_nume\_tara

)

) LOOP

flag := true;

EXIT;

END LOOP;

IF NOT flag THEN

RAISE invalid\_country\_name;

END IF;

END;

FOR prod\_rec IN (

SELECT distinct p.nume, p.pret

FROM produs p, produs\_se\_afla\_in\_comanda pc, comanda c, locatie l, tara t, categorie cat, furnizor f, furnizor\_distribuie\_in\_locatia fl

where p.cod\_produs = pc.cod\_produs

and pc.cod\_comanda = c.cod\_comanda

and l.cod\_tara = t.cod\_tara

and p.cod\_categorie = cat.cod\_categorie

and f.cod\_furnizor = fl.cod\_furnizor

and fl.cod\_locatie = l.cod\_locatie

and f.cod\_furnizor = p.cod\_furnizor

and initcap(t.nume) = initcap(v\_nume\_tara)

AND initcap(cat.nume)= initcap(v\_nume\_categorie)

AND p.pret <= v\_max\_pret

) LOOP

v\_raport := v\_raport || CHR(10) || 'Product: ' || prod\_rec.nume || ', Price: ' || prod\_rec.pret;

END LOOP;

IF v\_raport = 'Products in ' || v\_nume\_tara || ', Category: ' || v\_nume\_categorie || ' with price <= ' || v\_max\_pret || ':' THEN

RAISE NO\_DATA\_FOUND;

END IF;

RETURN v\_raport;

EXCEPTION

WHEN invalid\_country\_name THEN

DBMS\_OUTPUT.PUT\_LINE('Error: The country name given is not in the database.');

RAISE\_APPLICATION\_ERROR(-20001, 'Invalid country name');

RETURN NULL;

WHEN too\_small\_price THEN

DBMS\_OUTPUT.PUT\_LINE('Error: The max price given is too small compared to the database prices.');

RAISE\_APPLICATION\_ERROR(-20001, 'Price too small');

RETURN NULL;

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Error: No products found for the given criteria.');

RAISE\_APPLICATION\_ERROR(-20002, 'No products found for the given criteria');

RETURN NULL;

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('An error occurred: ' || SQLERRM);

RETURN NULL;

END raport\_produse\_tara\_categorie;

/

**Declaram codul pentru un apel reusit al functiei:**

DECLARE

v\_result VARCHAR2(4000);

BEGIN

v\_result := raport\_produse\_tara\_categorie('România', 'Electronice', 3000);

IF v\_result IS NOT NULL THEN

DBMS\_OUTPUT.PUT\_LINE(v\_result);

END IF;

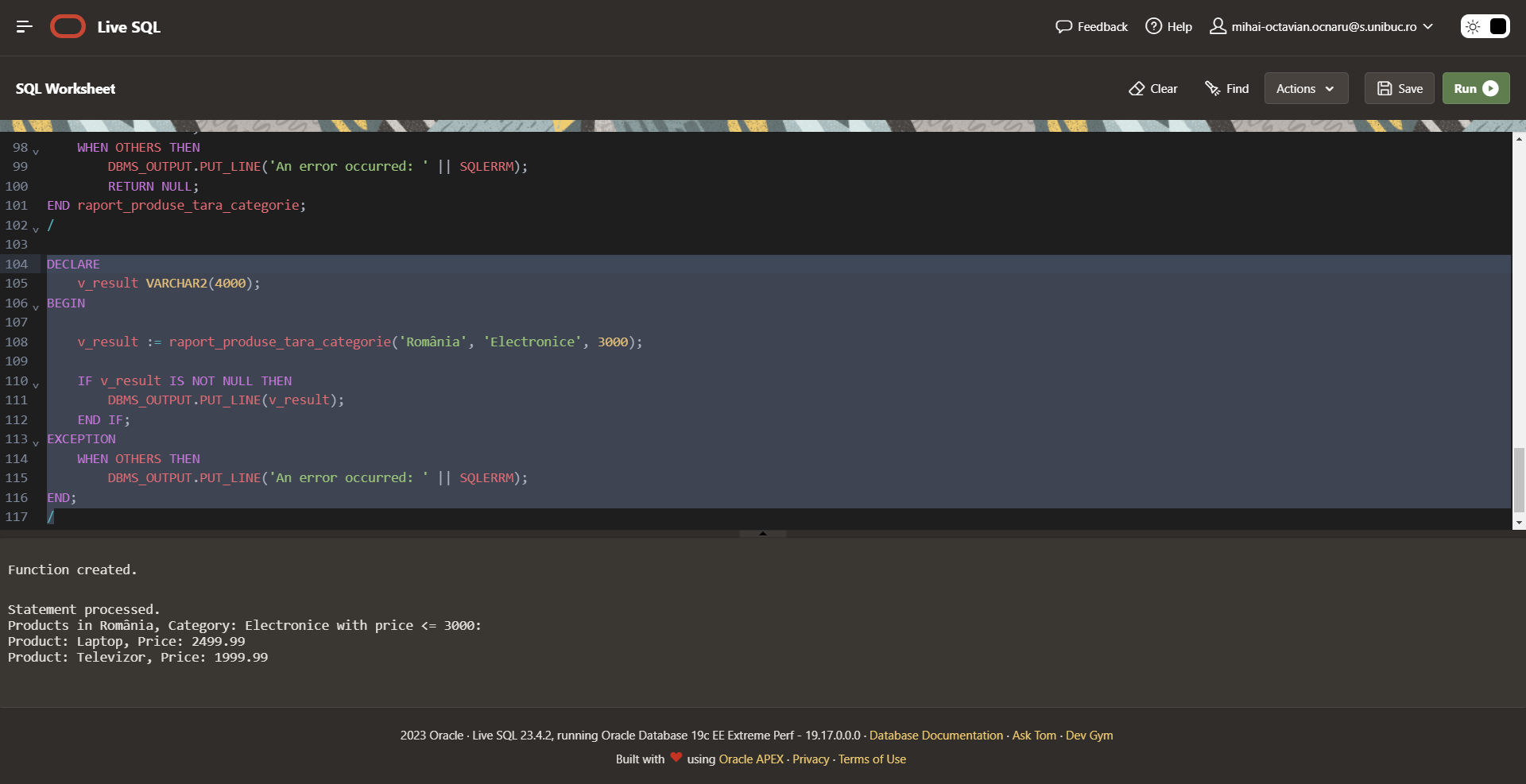
EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('An error occurred: ' || SQLERRM);

END;

/



**Declaram codul pentru o tara inexistenta in baza de date:**

DECLARE

v\_result VARCHAR2(4000);

BEGIN

v\_result := raport\_produse\_tara\_categorie('China', 'Electronice', 3000);

IF v\_result IS NOT NULL THEN

DBMS\_OUTPUT.PUT\_LINE(v\_result);

END IF;

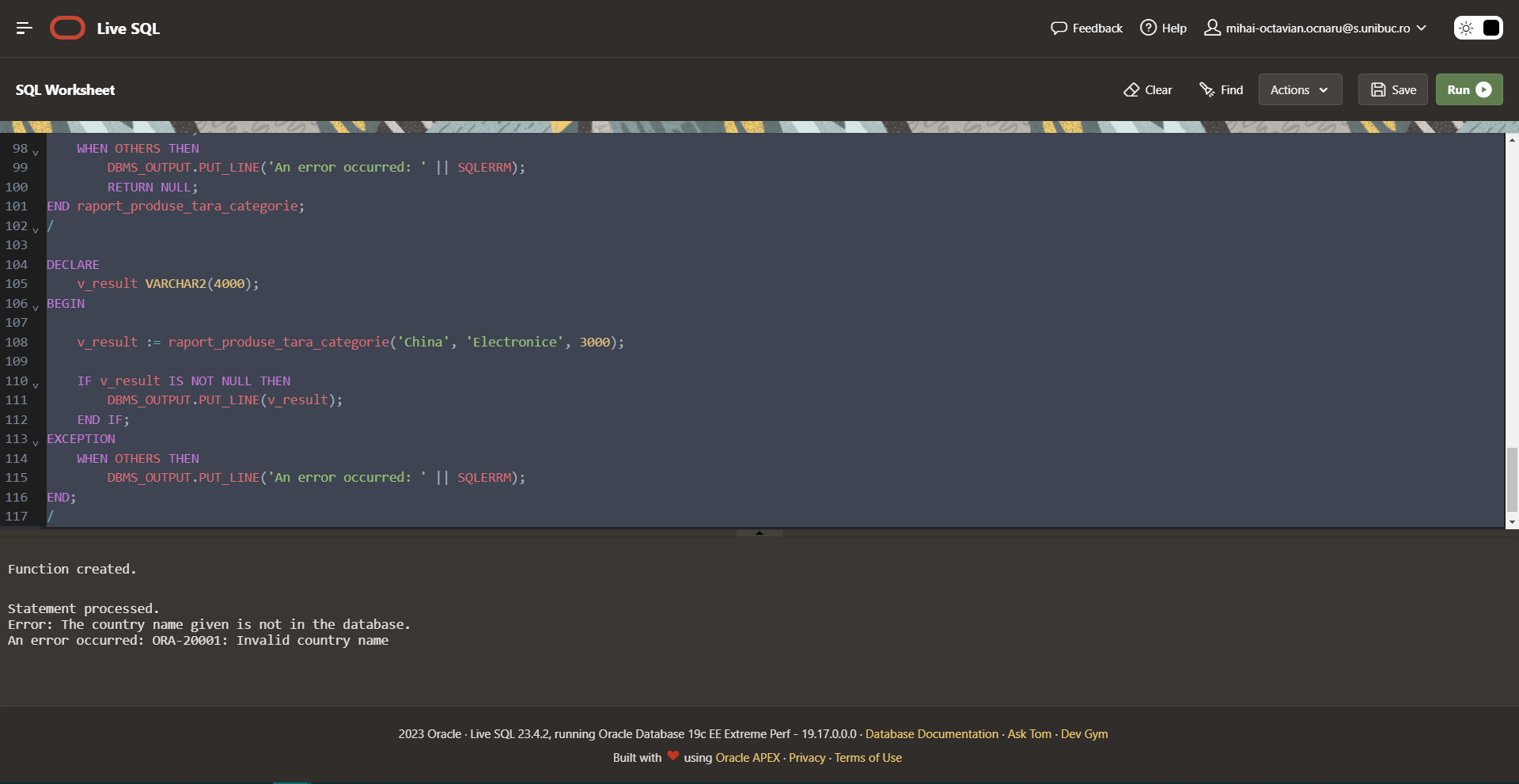
EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('An error occurred: ' || SQLERRM);

END;

/



**Declaram codul pentru un pret maxim prea mic:**

DECLARE

v\_result VARCHAR2(4000);

BEGIN

v\_result := raport\_produse\_tara\_categorie('România', 'Electronice', 0.5);

IF v\_result IS NOT NULL THEN

DBMS\_OUTPUT.PUT\_LINE(v\_result);

END IF;

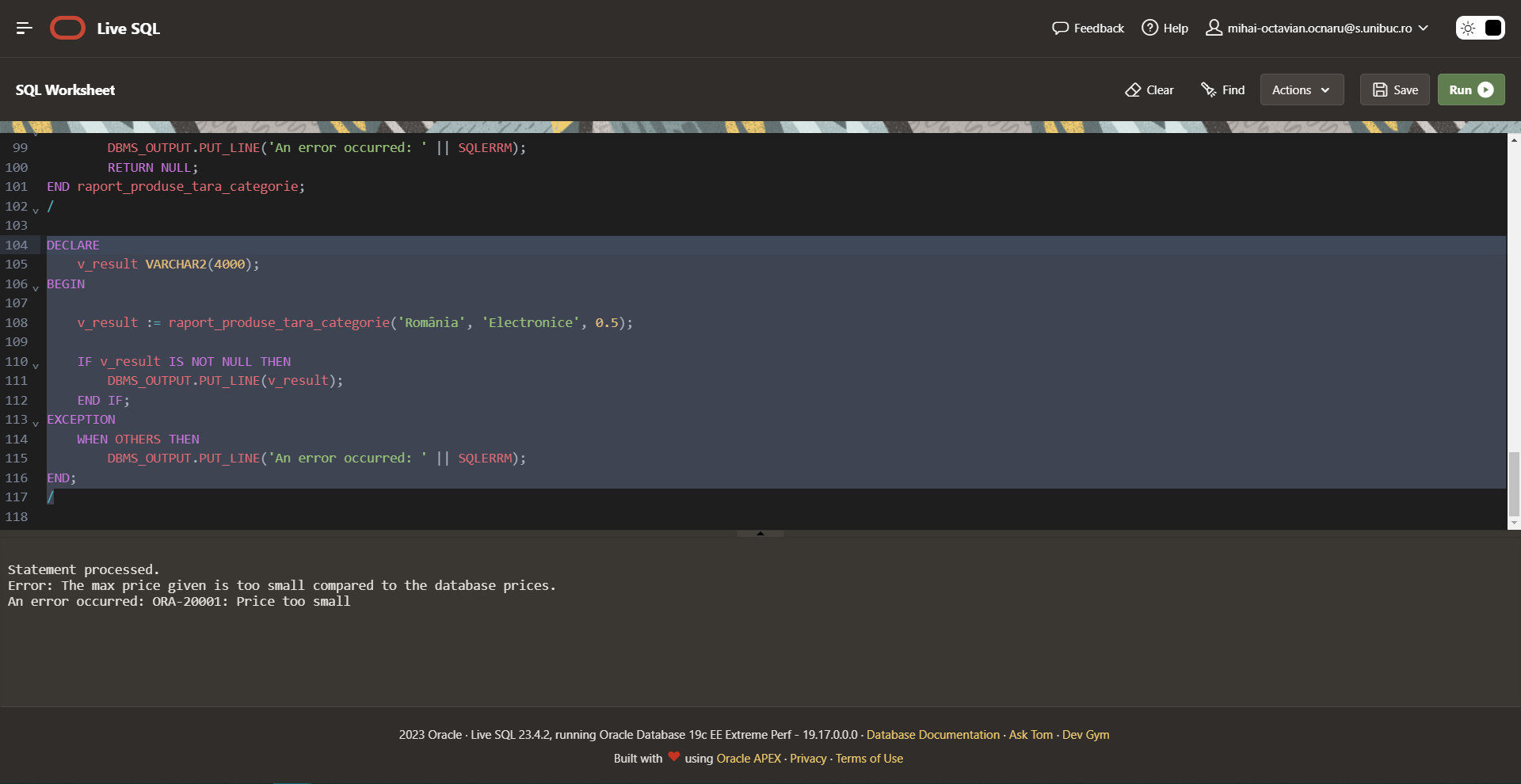
EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('An error occurred: ' || SQLERRM);

END;

/



1. **Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat independent de tip procedură care să utilizeze într-o singură comandă SQL 5 dintre tabelele definite. Tratați toate excepțiile care pot apărea, incluzând excepțiile NO\_DATA\_FOUND și TOO\_MANY\_ROWS. Apelați subprogramul astfel încât să evidențiați toate cazurile tratate.**

**Sa se afiseze numele tarii si numarul angajatilor in care traiesc cei mai multi lucratori care presteaza servicii in domeniul categoriei cu pretul mediu al produselor cel mai ridicat. Sa se afiseze de asemenea si numele acestei categorii.**

CREATE OR REPLACE PROCEDURE afisare\_tara\_max\_angajati\_categorie\_max\_pret AS

v\_nume\_tara Tara.nume%TYPE;

v\_numar\_angajati Number;

v\_nume\_categorie categorie.nume%TYPE;

BEGIN

WITH tari as (

SELECT DISTINCT t.nume, COUNT(COD\_ANGAJAT) AS nr\_angajati, c.nume as nume\_categorie

FROM tara t

JOIN locatie l ON t.cod\_tara = l.cod\_tara

JOIN angajat a ON l.cod\_locatie = a.cod\_locatie

JOIN categorie c ON a.cod\_categorie = c.cod\_categorie

WHERE a.cod\_categorie IN (

SELECT COD\_CATEGORIE

FROM produs

GROUP BY COD\_CATEGORIE

HAVING AVG(PRET) = (

SELECT MAX(AVG(PRET))

FROM PRODUS

GROUP BY COD\_CATEGORIE

)

)

GROUP BY t.cod\_tara, t.nume, c.nume

)

SELECT tari.nume, tari.nr\_angajati, tari.nume\_categorie

INTO v\_nume\_tara, v\_numar\_angajati, v\_nume\_categorie

FROM tari

GROUP BY tari.nume, tari.nr\_angajati, tari.nume\_categorie

HAVING tari.nr\_angajati = (SELECT MAX(nr\_angajati)

FROM tari

);

DBMS\_OUTPUT.PUT\_LINE('Tara cu cei mai multi angajati care lucreaza la categoria cu numele: '|| v\_nume\_categorie || ' si cu pretul mediu cel mai mare: ' || v\_nume\_tara || ' - numar angajati: ' || v\_numar\_angajati);

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Nu există date pentru cerință.');

WHEN TOO\_MANY\_ROWS THEN

DBMS\_OUTPUT.PUT\_LINE('Eroare: Prea multe rânduri returnate.');

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('A apărut o eroare: ' || SQLERRM);

END afisare\_tara\_max\_angajati\_categorie\_max\_pret;

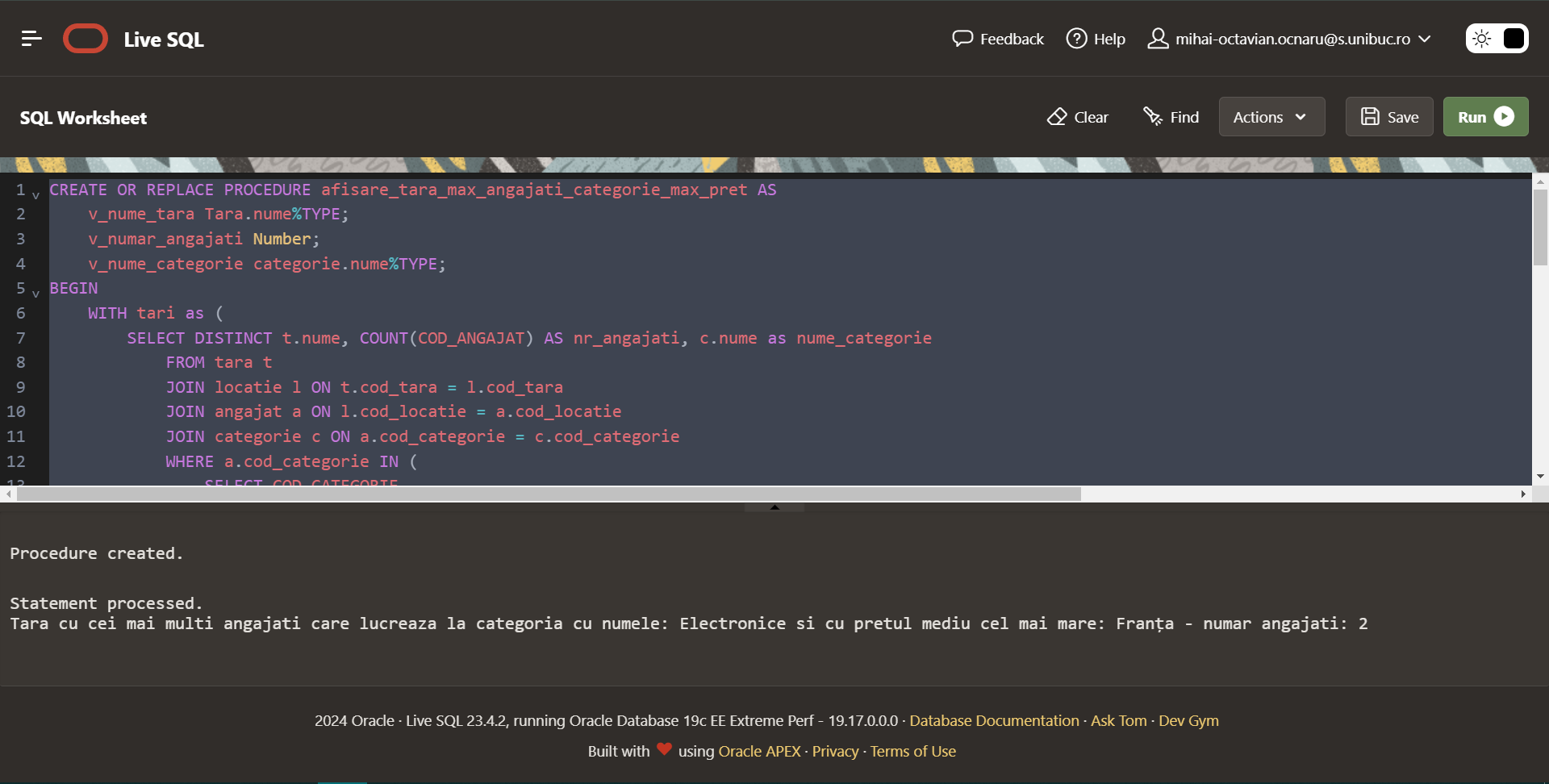
/

BEGIN

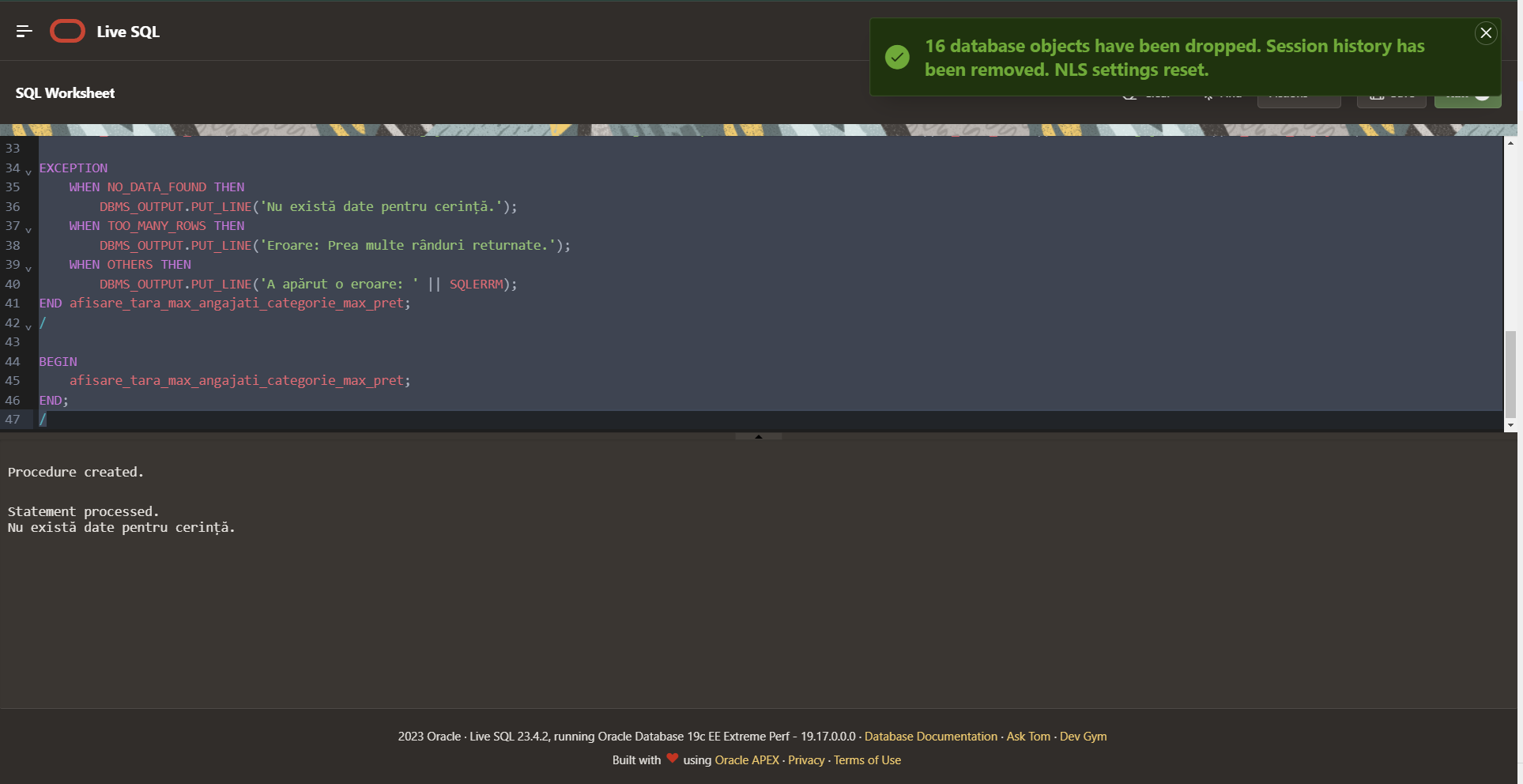
afisare\_tara\_max\_angajati\_categorie\_max\_pret;

END;

/



**Pentru exemplificarea cazului NO\_DATA\_FOUND nu am mai rulat scriptul de inserare in baza de date:**



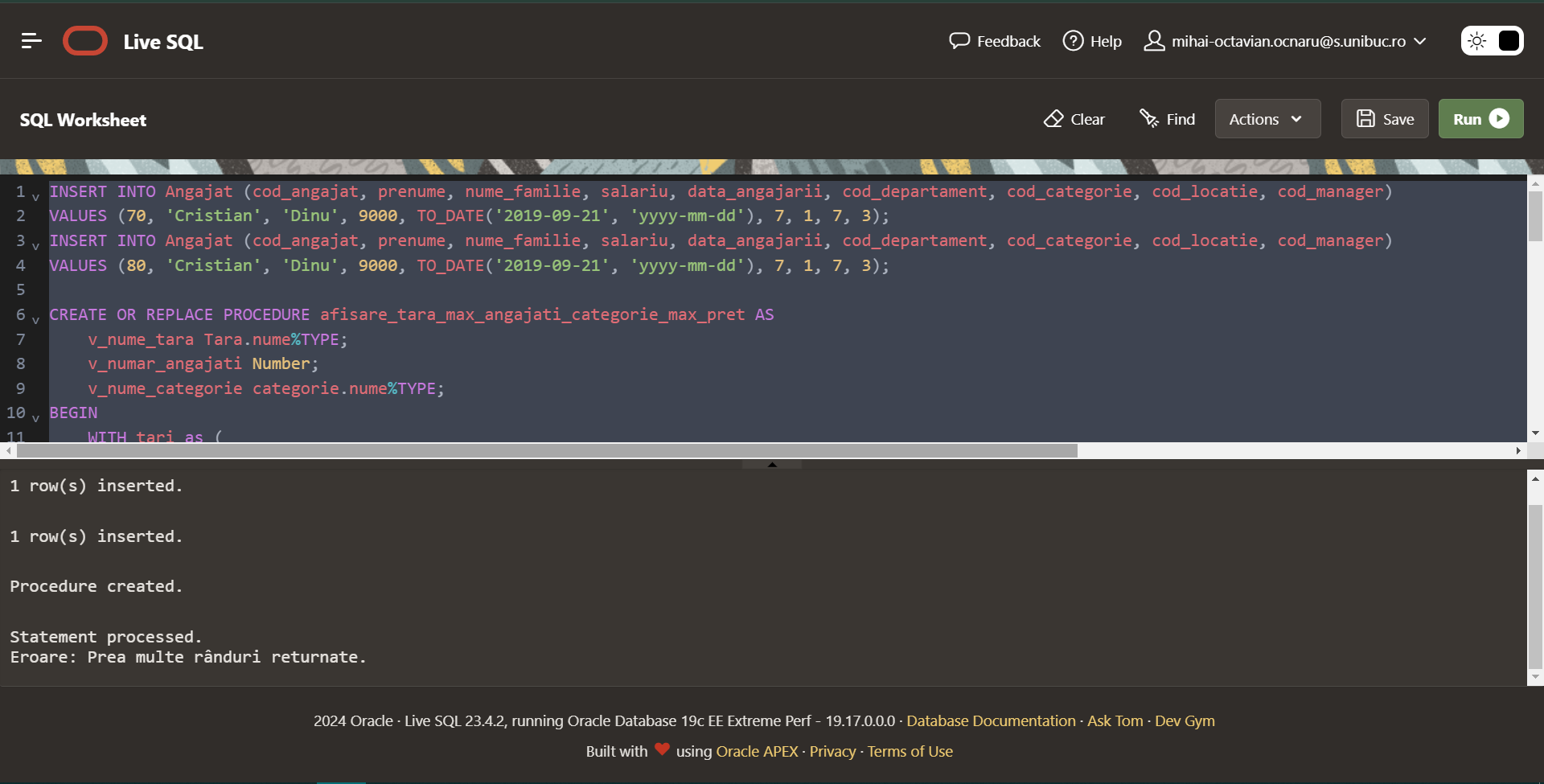
**Pentru exemplificarea cazului TOO\_MANY\_ROWS am inserat doi angajati ce lucreaza in categoria 1 si sunt situati in Spania:**

INSERT INTO Angajat (cod\_angajat, prenume, nume\_familie, salariu, data\_angajarii, cod\_departament, cod\_categorie, cod\_locatie, cod\_manager)

VALUES (70, 'Cristian', 'Dinu', 9000, TO\_DATE('2019-09-21', 'yyyy-mm-dd'), 7, 1, 7, 3);

INSERT INTO Angajat (cod\_angajat, prenume, nume\_familie, salariu, data\_angajarii, cod\_departament, cod\_categorie, cod\_locatie, cod\_manager)

VALUES (80, 'Cristian', 'Dinu', 9000, TO\_DATE('2019-09-21', 'yyyy-mm-dd'), 7, 1, 7, 3);



1. **Definiți un trigger de tip LMD la nivel de comandă. Declanșați trigger-ul.**

**Sa se creeze un declansator astfel incat adaugarea unui produs intr-o comanda sa fie interzisa daca valoarea cumulata a reducerilor oferite prin discount comenzilor depaseste suma de 10.000.**

CREATE OR REPLACE TRIGGER discount\_comenzi

BEFORE INSERT OR UPDATE ON PRODUS\_SE\_AFLA\_IN\_COMANDA

DECLARE

red\_total produs.pret%TYPE;

BEGIN

WITH tabela\_comenzi AS (

SELECT c.cod\_comanda, SUM(p.pret) AS pret\_initial, SUM(p.pret \* (1 - c.discount)) AS pret\_redus, c.discount \* SUM(p.pret) AS reducere

FROM comanda c, produs p, produs\_se\_afla\_in\_comanda pc

WHERE c.cod\_comanda = pc.cod\_comanda

AND pc.cod\_produs = p.cod\_produs

GROUP BY c.cod\_comanda, c.discount

)

SELECT SUM(t.reducere)

INTO red\_total

FROM tabela\_comenzi t;

IF red\_total > 10000 THEN

RAISE\_APPLICATION\_ERROR(-20101, 'Nu puteti sa mai introduceti o noua comanda cu discount nenul. Suma reducerilor depasita.');

END IF;

END;

/

WITH tabela\_comenzi AS (

SELECT c.cod\_comanda, SUM(p.pret) AS pret\_initial, SUM(p.pret \* (1 - c.discount)) AS pret\_redus, c.discount \* SUM(p.pret) AS reducere

FROM comanda c, produs p, produs\_se\_afla\_in\_comanda pc

WHERE c.cod\_comanda = pc.cod\_comanda

AND pc.cod\_produs = p.cod\_produs

GROUP BY c.cod\_comanda, c.discount

)

SELECT SUM(t.reducere) suma\_reduceri

FROM tabela\_comenzi t;

**Pentru evidentiere am creat o noua comanda si am inserat produse intr-o comanda creata recent, astfel suma totala fiind initial 9897.2102, dupa prima inserare ar depasi suma target de 10.000 ceea ce rezulta intr-o eroare.**

BEGIN

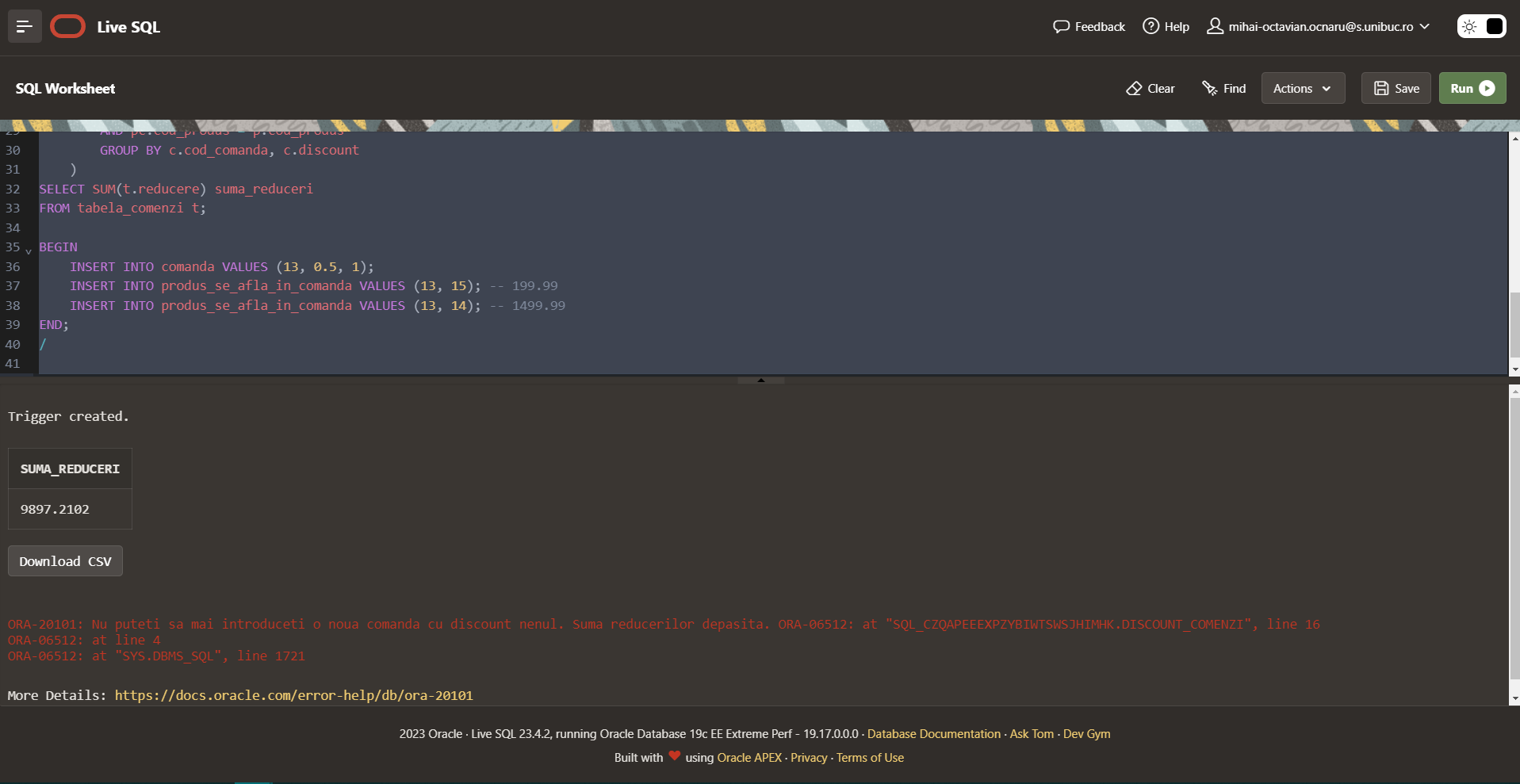
INSERT INTO comanda VALUES (13, 0.5, 1);

INSERT INTO produs\_se\_afla\_in\_comanda VALUES (13, 15); -- 199.99

INSERT INTO produs\_se\_afla\_in\_comanda VALUES (13, 14); -- 1499.99

END;

/



1. **Definiți un trigger de tip LMD la nivel de linie. Declanșați trigger-ul.**

**Sa se creeze un declansator care sa interzica inserarea unui angajat daca acesta are salariu mai mic decat media departamentului sau, de asemenea se interzice cresterea salariala daca acest raport este mai mare decat cel mai mic salariu maxim pe departamente, iar procentul micsorarii salariului nu poate fi mai mic decat raportul dintre cele mai mici doua salarii.**

**Exemplificare micsorare: 3500 si 4000 sunt cele mai mici salarii, avand raportul 3500/4000 = 0.875. Un salariat cu salariul de 4000 nu va putea primii un salariu mai mic de 3500, atfel raportul va fi mai mic decat 0.875.**

CREATE OR REPLACE TRIGGER salariu\_angajat

BEFORE INSERT OR UPDATE ON ANGAJAT

FOR EACH ROW

DECLARE

PRAGMA AUTONOMOUS\_TRANSACTION;

v\_dep\_sal\_mediu Angajat.salariu%TYPE;

v\_sal\_aux Angajat.salariu%TYPE;

TYPE sal\_array IS VARRAY(2) OF Angajat.salariu%TYPE;

v\_sal\_array sal\_array;

v\_proc Number;

BEGIN

IF INSERTING THEN

SELECT AVG(salariu)

INTO v\_dep\_sal\_mediu

FROM angajat

WHERE cod\_departament = :NEW.cod\_departament;

IF v\_dep\_sal\_mediu > :NEW.salariu THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Noul angajat trebuie să aibă cel puțin un salariu egal cu ' || v\_dep\_sal\_mediu || '.');

END IF;

ELSIF UPDATING THEN

-- crestere salariala

IF :NEW.salariu > :OLD.salariu THEN

WITH sal\_max AS (

SELECT cod\_departament, MAX(salariu) maxim

FROM angajat

GROUP BY cod\_departament

)

SELECT MIN(s.maxim)

INTO v\_sal\_aux

FROM sal\_max s;

IF :NEW.salariu > v\_sal\_aux THEN

RAISE\_APPLICATION\_ERROR(-20002, 'Majorarea salarială nu poate depăși minimul salariilor maxime pe departament!');

END IF;

-- scadere salariala

ELSE

WITH salarii AS (

SELECT DISTINCT salariu

FROM angajat

ORDER BY salariu

)

SELECT salariu

BULK COLLECT INTO v\_sal\_array

FROM salarii

WHERE ROWNUM <= 2;

v\_proc := v\_sal\_array(1) / v\_sal\_array(2);

IF :NEW.salariu / :OLD.salariu < v\_proc THEN

RAISE\_APPLICATION\_ERROR(-20003, 'Micșorarea salariului nu poate fi mai mică decât diferența procentuală a celor mai mici 2 salarii');

END IF;

END IF;

END IF;

END;/

/

**Pentru cazul de inserare am afisat media pe departamentul 1 si am adaugat un nou angajat in departamentul respectiv cu salariul mai mic decat media.**

DECLARE

v\_avg\_sal Angajat.salariu%TYPE;

v\_cod\_dep Angajat.cod\_departament%TYPE;

BEGIN

v\_cod\_dep := 1;

select avg(salariu)

into v\_avg\_sal

from angajat

where cod\_departament = v\_cod\_dep

group by cod\_departament;

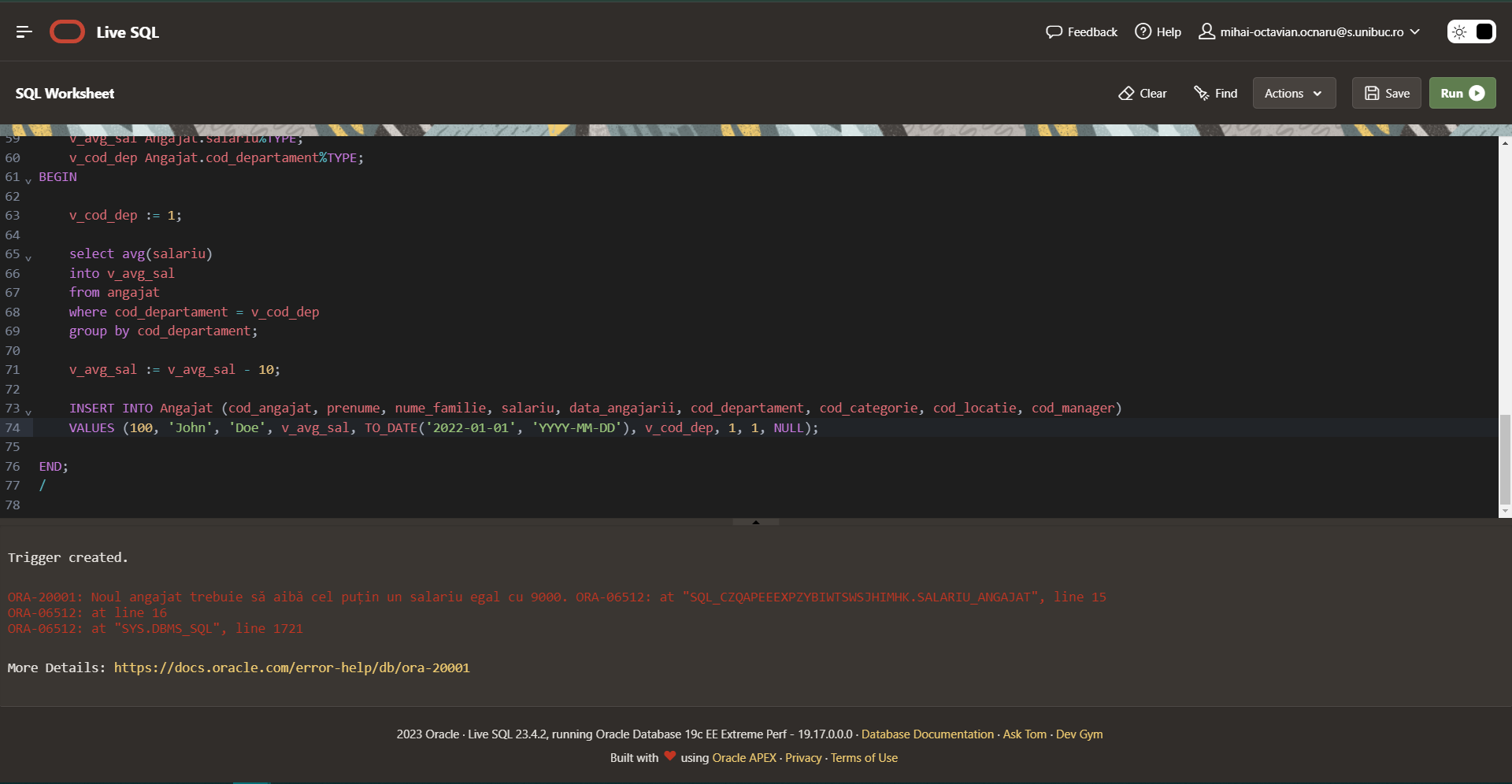
v\_avg\_sal := v\_avg\_sal - 10;

INSERT INTO Angajat (cod\_angajat, prenume, nume\_familie, salariu, data\_angajarii, cod\_departament, cod\_categorie, cod\_locatie, cod\_manager)

VALUES (100, 'John', 'Doe', v\_avg\_sal, TO\_DATE('2022-01-01', 'YYYY-MM-DD'), v\_cod\_dep, 1, 1, NULL);

END;

/



**Pentru cazul micsorarii am luat angajatul din exemplu cu salariul 4000 si am incercat sa ii updatez salariul la 3200:**

DECLARE

v\_cod\_angajat Angajat.cod\_angajat%TYPE;

v\_sal Angajat.salariu%TYPE;

BEGIN

v\_cod\_angajat := 11;

v\_sal := 3200;

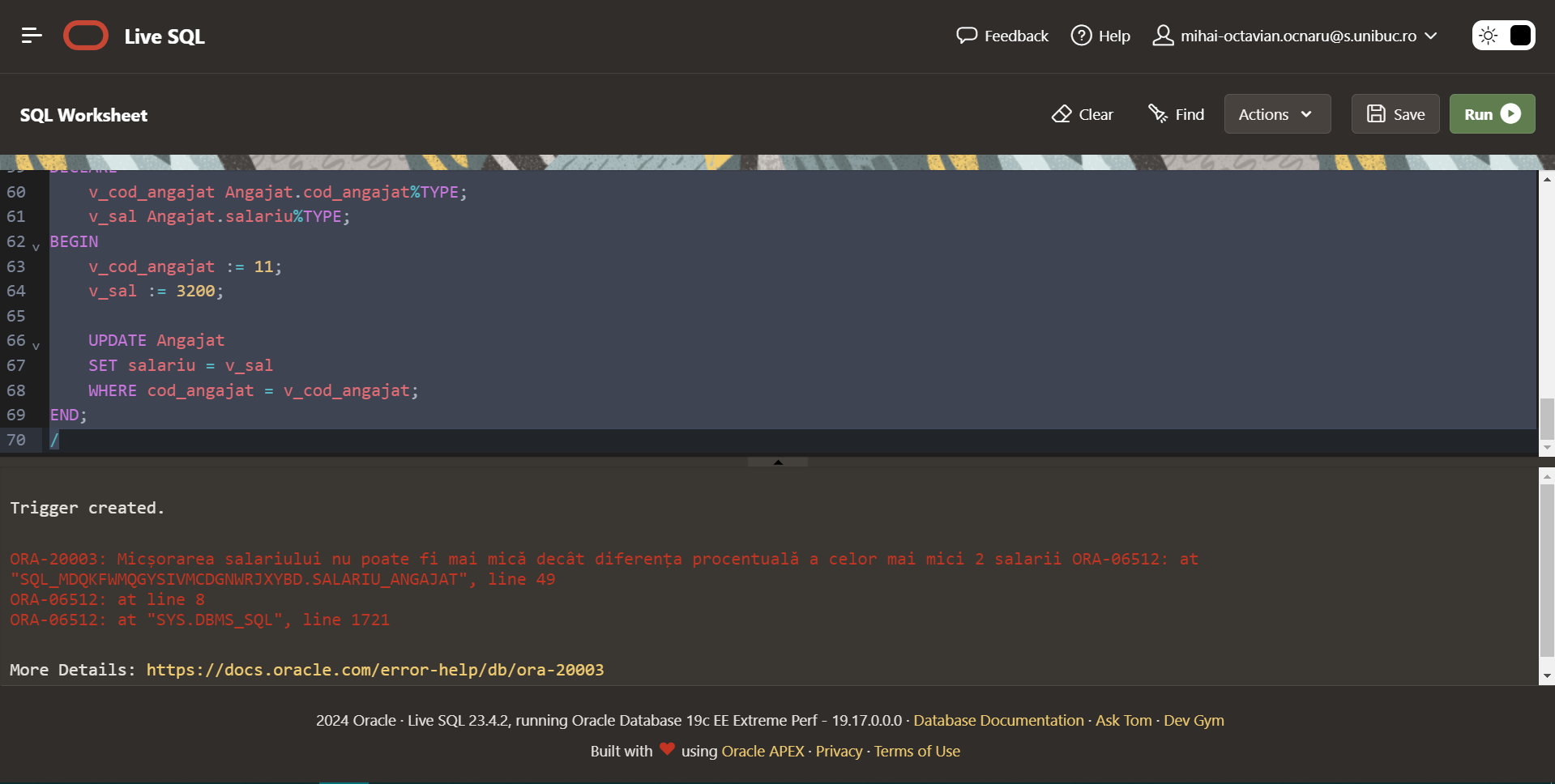
UPDATE Angajat

SET salariu = v\_sal

WHERE cod\_angajat = v\_cod\_angajat;

END;

/



**Iar pentru cazul cresterii salariale am ales un salariu de 150000 ce nu se regaseste in inserarile bazei de date:**

DECLARE

v\_cod\_angajat Angajat.cod\_angajat%TYPE;

v\_sal Angajat.salariu%TYPE;

BEGIN

v\_cod\_angajat := 11;

v\_sal := 150000;

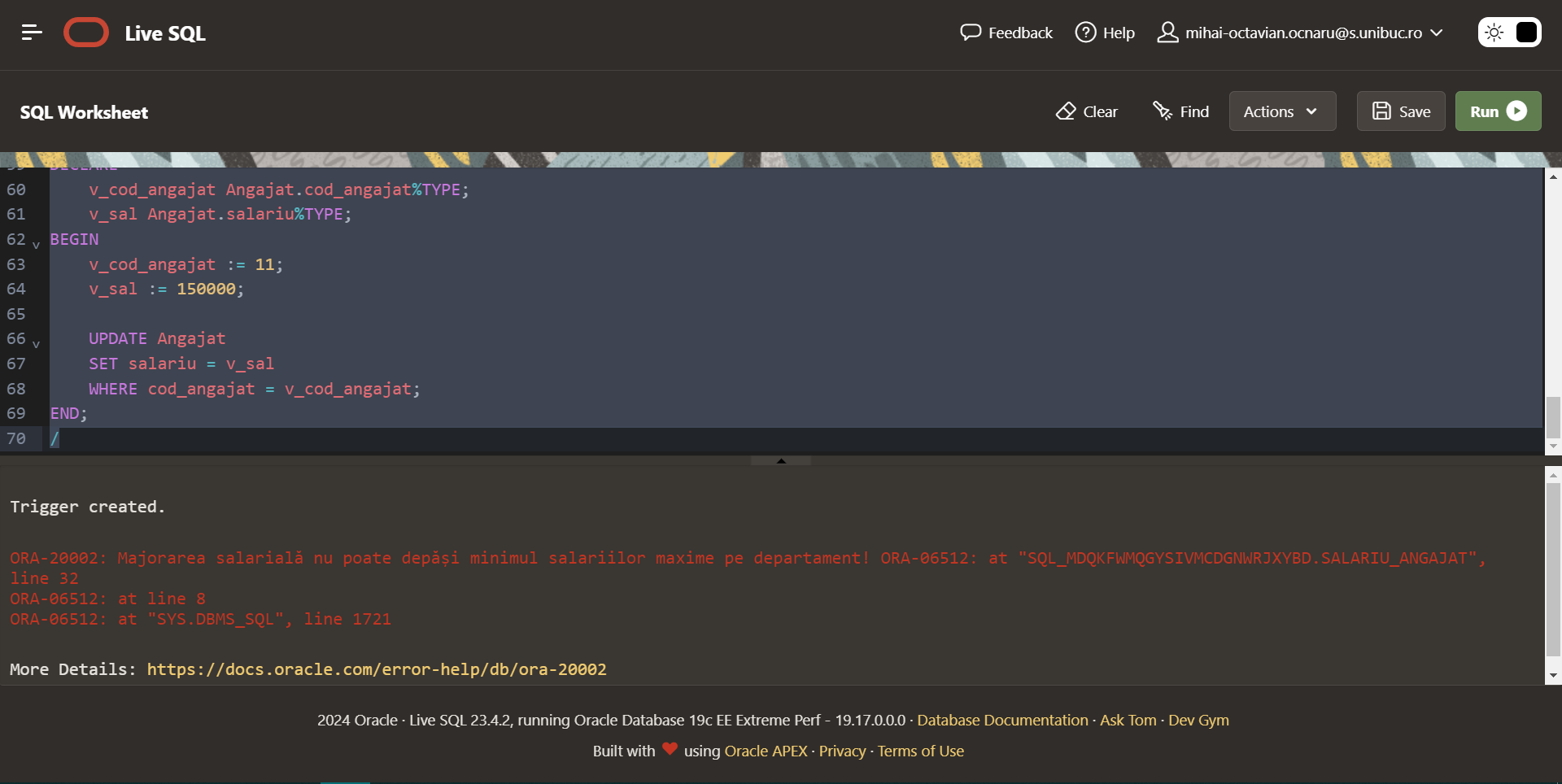
UPDATE Angajat

SET salariu = v\_sal

WHERE cod\_angajat = v\_cod\_angajat;

END;

/



1. **Definiți un trigger de tip LDD. Declanșați trigger-ul.**

**Sa se creeze un declansator care sa nu permita creerea/editarea/stergerea tabelelor decat in timpul programului de munca. Prin program de munca se inteleg zilele de Luni pana Vineri de la ora 8:00 - 16:00. Sa se creeze un tabel in care sa se pastreze arhive ale utilizatorului, data, actiunea si statusul comenzii efectuate.**

CREATE TABLE logs (

username VARCHAR2(30),

log\_date DATE,

sysevent VARCHAR2(20),

status VARCHAR2(10)

);

CREATE OR REPLACE TRIGGER time\_control\_trigger

BEFORE CREATE OR ALTER OR DROP ON SCHEMA

DECLARE

valid\_time BOOLEAN;

v\_status VARCHAR2(10);

BEGIN

v\_status := 'Completed';

valid\_time := TO\_CHAR(SYSDATE, 'Dy', 'NLS\_DATE\_LANGUAGE=American') IN ('Mon', 'Tue', 'Wed', 'Thu', 'Fri') AND

TO\_NUMBER(TO\_CHAR(SYSDATE, 'HH24')) BETWEEN 8 AND 16;

IF NOT valid\_time THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Action not allowed outside working hours.');

v\_status := 'Failed';

END IF;

INSERT INTO logs (username, log\_date, sysevent, status)

VALUES (USER, SYSDATE, SYS.SYSEVENT, v\_status);

END;

/

**Am creat o tabela dummy inainte te initializarea scriptului:**

CREATE TABLE dummy\_table (

id NUMBER,

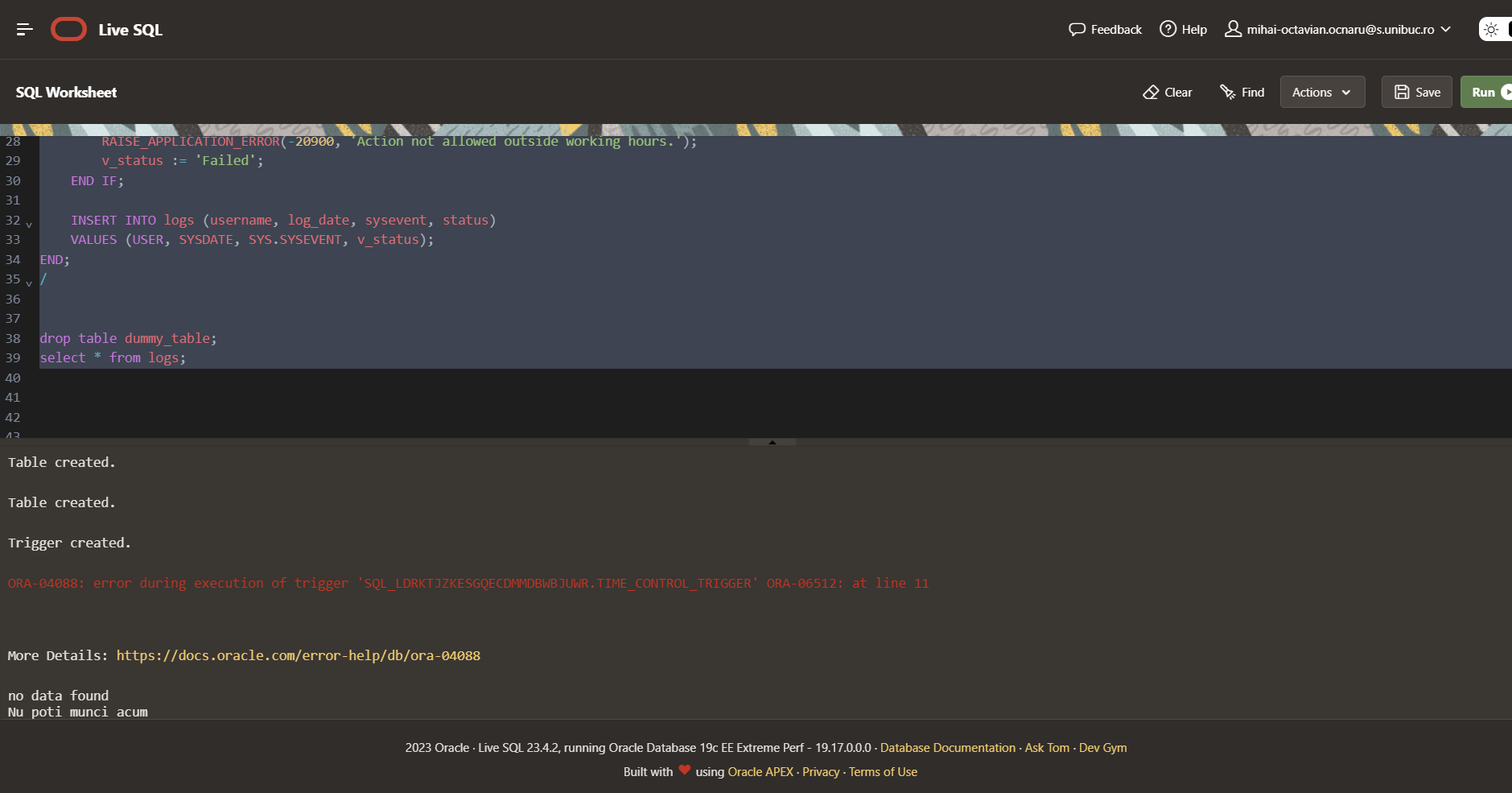
name VARCHAR2(50),

created\_at TIMESTAMP

);

**Si am incercat sa o sterg dupa initializarea scriptului:**

DROP TABLE dummy\_table;



1. **Definiți un pachet care să conțină toate obiectele definite în cadrul proiectului.**

CREATE OR REPLACE PACKAGE proiect\_ocnaru\_mihai AS

PROCEDURE display\_user\_orders;

PROCEDURE raport\_produse\_recenzii;

FUNCTION raport\_produse\_tara\_categorie(

v\_nume\_tara IN VARCHAR2,

v\_nume\_categorie IN VARCHAR2,

v\_max\_pret IN NUMBER

) RETURN VARCHAR2;

PROCEDURE afisare\_tara\_max\_angajati\_categorie\_max\_pret;

END;

/

CREATE OR REPLACE PACKAGE BODY proiect\_ocnaru\_mihai AS

--6

PROCEDURE display\_user\_orders

AS

TYPE produs\_record IS RECORD (

cod\_produs produs.cod\_produs%TYPE,

pret produs.pret%TYPE,

nume produs.nume%TYPE

);

TYPE vector\_produse IS VARRAY(200) OF produs\_record;

TYPE comanda\_cu\_produse\_record IS RECORD (

cod\_comanda comanda.cod\_comanda%TYPE,

produse vector\_produse

);

TYPE comanda\_cu\_produse\_nested\_table IS TABLE OF comanda\_cu\_produse\_record;

TYPE utilizator\_comanda\_indexed\_table IS TABLE OF comanda\_cu\_produse\_nested\_table INDEX BY PLS\_INTEGER;

utilizatori\_cu\_comenzi utilizator\_comanda\_indexed\_table;

BEGIN

FOR utilizator IN (SELECT DISTINCT cod\_utilizator FROM comanda)

LOOP

utilizatori\_cu\_comenzi(utilizator.cod\_utilizator) := comanda\_cu\_produse\_nested\_table();

DBMS\_OUTPUT.PUT\_LINE('Utilizator: ' || utilizator.cod\_utilizator);

FOR comanda IN (SELECT cod\_comanda FROM comanda WHERE cod\_utilizator = utilizator.cod\_utilizator)

LOOP

DECLARE

produse vector\_produse := vector\_produse();

BEGIN

FOR produs IN (SELECT p.cod\_produs, p.pret, p.nume

FROM produs p, produs\_se\_afla\_in\_comanda pc

WHERE p.cod\_produs = pc.cod\_produs

AND pc.cod\_comanda = comanda.cod\_comanda)

LOOP

produse.extend();

produse(produse.last) := produs\_record(

cod\_produs => produs.cod\_produs,

pret => produs.pret,

nume => produs.nume

);

END LOOP;

utilizatori\_cu\_comenzi(utilizator.cod\_utilizator).EXTEND;

utilizatori\_cu\_comenzi(utilizator.cod\_utilizator)(utilizatori\_cu\_comenzi(utilizator.cod\_utilizator).LAST) :=

comanda\_cu\_produse\_record(

cod\_comanda => comanda.cod\_comanda,

produse => produse

);

DBMS\_OUTPUT.PUT\_LINE(' Comanda: ' || comanda.cod\_comanda);

FOR i IN 1..produse.count

LOOP

DBMS\_OUTPUT.PUT\_LINE(' Produs: ' || produse(i).cod\_produs ||

', Nume: ' || produse(i).nume ||

', Pret: ' || produse(i).pret);

END LOOP;

END;

END LOOP;

DBMS\_OUTPUT.PUT\_LINE('-------------');

DBMS\_OUTPUT.NEW\_LINE;

END LOOP;

END display\_user\_orders;

--7

PROCEDURE raport\_produse\_recenzii AS

CURSOR curson\_produse IS

SELECT distinct p.cod\_produs, p.nume, p.pret

FROM produs p, produs\_se\_afla\_in\_comanda pc, comanda c

where pc.cod\_produs = p.cod\_produs

and pc.cod\_comanda = c.cod\_comanda

and c.cod\_utilizator in (select distinct u.cod\_utilizator

from locatie l, utilizator u, tara t

where l.cod\_locatie = u.cod\_locatie

and l.cod\_tara = t.cod\_tara

and initcap(t.nume) in ('România', 'Franța', 'Spania', 'Italia'));

CURSOR curson\_recenzii (v\_cod\_produs INT) IS

SELECT descriere, rating

FROM Review

WHERE cod\_produs = v\_cod\_produs;

v\_cod\_produs Produs.cod\_produs%TYPE;

v\_nume\_produs Produs.nume%TYPE;

v\_pret Produs.pret%TYPE;

v\_descriere Review.descriere%TYPE;

v\_rating Review.rating%TYPE;

BEGIN

OPEN curson\_produse;

LOOP

FETCH curson\_produse INTO v\_cod\_produs, v\_nume\_produs, v\_pret;

EXIT WHEN curson\_produse%NOTFOUND;

OPEN curson\_recenzii(v\_cod\_produs);

LOOP

FETCH curson\_recenzii INTO v\_descriere, v\_rating;

EXIT WHEN curson\_recenzii%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE('Produs: ' || v\_nume\_produs || ', Pret: ' || v\_pret);

DBMS\_OUTPUT.PUT\_LINE(' Recenzie: ' || v\_descriere || ', Rating: ' || v\_rating);

DBMS\_OUTPUT.PUT\_LINE('------');

DBMS\_OUTPUT.NEW\_LINE;

END LOOP;

CLOSE curson\_recenzii;

END LOOP;

CLOSE curson\_produse;

END raport\_produse\_recenzii;

--8

FUNCTION raport\_produse\_tara\_categorie(

v\_nume\_tara IN VARCHAR2,

v\_nume\_categorie IN VARCHAR2,

v\_max\_pret IN NUMBER

) RETURN VARCHAR2

AS

v\_raport VARCHAR2(4000);

invalid\_country\_name EXCEPTION;

PRAGMA EXCEPTION\_INIT(invalid\_country\_name, -1333);

too\_small\_price EXCEPTION;

PRAGMA EXCEPTION\_INIT(too\_small\_price, -1332);

BEGIN

v\_raport := 'Products in ' || v\_nume\_tara || ', Category: ' || v\_nume\_categorie || ' with price <= ' || v\_max\_pret || ':';

DECLARE

min\_price produs.pret%TYPE;

BEGIN

select min(p.pret)

into min\_price

FROM tara t, locatie l, furnizor f, furnizor\_distribuie\_in\_locatia fl, produs p

WHERE t.cod\_tara = l.cod\_tara

AND f.cod\_furnizor = fl.cod\_furnizor

AND fl.cod\_locatie = l.cod\_locatie

AND f.cod\_furnizor = p.cod\_furnizor;

if min\_price > v\_max\_pret then

RAISE too\_small\_price;

end if;

END;

DECLARE

flag Boolean := false;

BEGIN

FOR country IN (

SELECT 1

FROM dual

WHERE EXISTS (

SELECT 1

FROM tara t, locatie l, furnizor f, furnizor\_distribuie\_in\_locatia fl

WHERE t.cod\_tara = l.cod\_tara

AND f.cod\_furnizor = fl.cod\_furnizor

AND fl.cod\_locatie = l.cod\_locatie

AND t.nume = v\_nume\_tara

)

) LOOP

flag := true;

EXIT;

END LOOP;

IF NOT flag THEN

RAISE invalid\_country\_name;

END IF;

END;

FOR prod\_rec IN (

SELECT distinct p.nume, p.pret

FROM produs p, produs\_se\_afla\_in\_comanda pc, comanda c, locatie l, tara t, categorie cat, furnizor f, furnizor\_distribuie\_in\_locatia fl

where p.cod\_produs = pc.cod\_produs

and pc.cod\_comanda = c.cod\_comanda

and l.cod\_tara = t.cod\_tara

and p.cod\_categorie = cat.cod\_categorie

and f.cod\_furnizor = fl.cod\_furnizor

and fl.cod\_locatie = l.cod\_locatie

and f.cod\_furnizor = p.cod\_furnizor

and initcap(t.nume) = initcap(v\_nume\_tara)

AND initcap(cat.nume)= initcap(v\_nume\_categorie)

AND p.pret <= v\_max\_pret

) LOOP

v\_raport := v\_raport || CHR(10) || 'Product: ' || prod\_rec.nume || ', Price: ' || prod\_rec.pret;

END LOOP;

IF v\_raport = 'Products in ' || v\_nume\_tara || ', Category: ' || v\_nume\_categorie || ' with price <= ' || v\_max\_pret || ':' THEN

RAISE NO\_DATA\_FOUND;

END IF;

RETURN v\_raport;

EXCEPTION

WHEN invalid\_country\_name THEN

DBMS\_OUTPUT.PUT\_LINE('Error: The country name given is not in the database.');

RAISE\_APPLICATION\_ERROR(-20001, 'Invalid country name');

RETURN NULL;

WHEN too\_small\_price THEN

DBMS\_OUTPUT.PUT\_LINE('Error: The max price given is too small compared to the database prices.');

RAISE\_APPLICATION\_ERROR(-20001, 'Price too small');

RETURN NULL;

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Error: No products found for the given criteria.');

RAISE\_APPLICATION\_ERROR(-20002, 'No products found for the given criteria');

RETURN NULL;

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('An error occurred: ' || SQLERRM);

RETURN NULL;

END raport\_produse\_tara\_categorie;

--9

PROCEDURE afisare\_tara\_max\_angajati\_categorie\_max\_pret AS

v\_nume\_tara Tara.nume%TYPE;

v\_numar\_angajati Number;

BEGIN

WITH tari as (

SELECT DISTINCT t.nume, COUNT(COD\_ANGAJAT) AS nr\_angajati

FROM tara t

JOIN locatie l ON t.cod\_tara = l.cod\_tara

JOIN angajat a ON l.cod\_locatie = a.cod\_locatie

WHERE a.cod\_categorie IN (

SELECT COD\_CATEGORIE

FROM produs

GROUP BY COD\_CATEGORIE

HAVING AVG(PRET) = (

SELECT MAX(AVG(PRET))

FROM PRODUS

GROUP BY COD\_CATEGORIE

)

)

GROUP BY t.cod\_tara, t.nume

)

SELECT tari.nume, tari.nr\_angajati

INTO v\_nume\_tara, v\_numar\_angajati

FROM tari

GROUP BY tari.nume, tari.nr\_angajati

HAVING tari.nr\_angajati = (SELECT MAX(nr\_angajati)

FROM tari

);

DBMS\_OUTPUT.PUT\_LINE('Tara cu cei mai multi angajati care lucreaza la categoria cu pretul mediu cel mai mare: ' || v\_nume\_tara || ' numar angajati: ' || v\_numar\_angajati);

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Nu există date pentru cerință.');

WHEN TOO\_MANY\_ROWS THEN

DBMS\_OUTPUT.PUT\_LINE('Eroare: Prea multe rânduri returnate.');

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('A apărut o eroare: ' || SQLERRM);

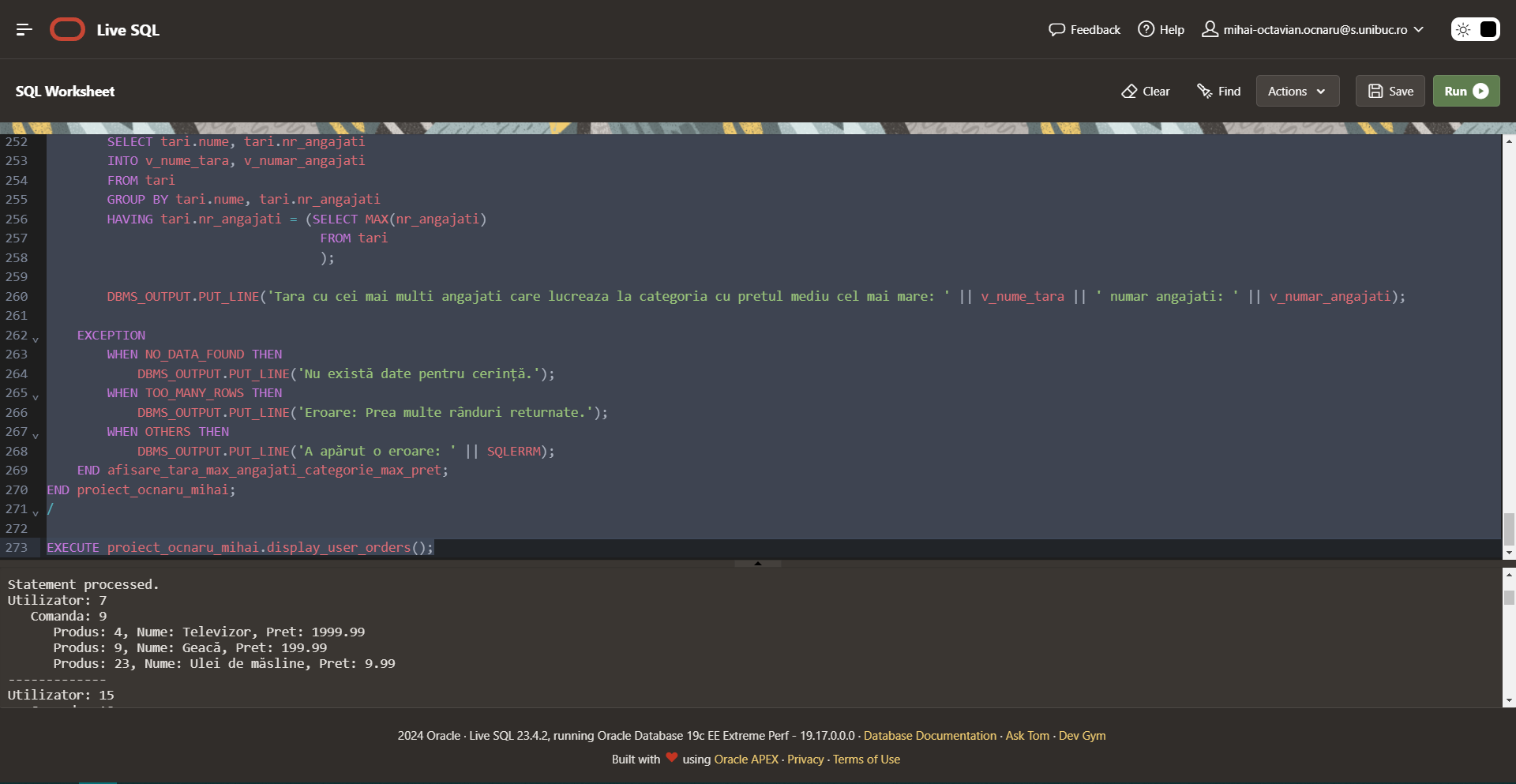
END afisare\_tara\_max\_angajati\_categorie\_max\_pret;

END proiect\_ocnaru\_mihai;

/

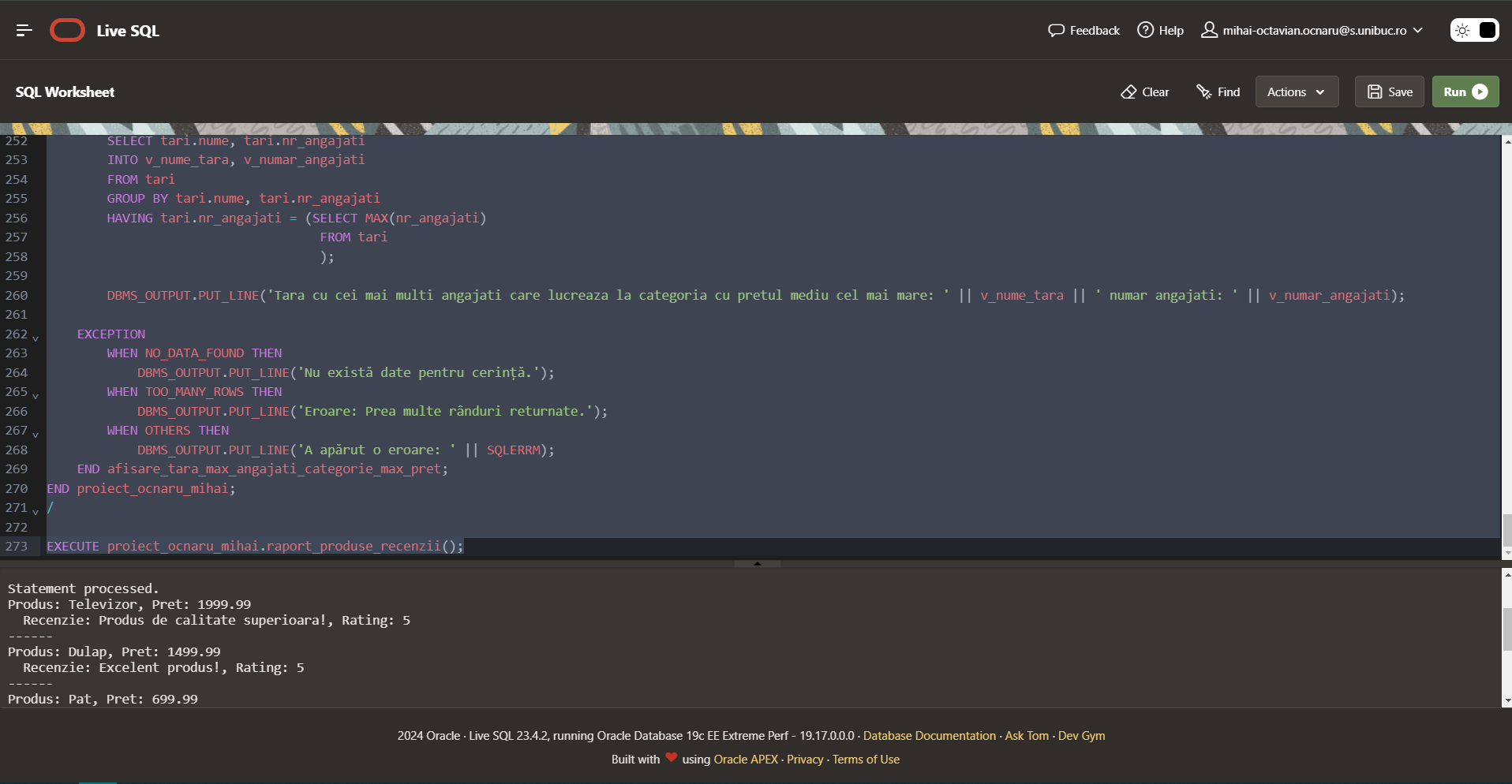
**Apelam ex. 6 din pachet:**

EXECUTE proiect\_ocnaru\_mihai.display\_user\_orders();



**Apelam ex. 7 din pachet:**

EXECUTE proiect\_ocnaru\_mihai.raport\_produse\_recenzii();



**Apelam ex. 8 din pachet:**

**Apel reusit al functiei**

DECLARE

v\_result VARCHAR2(4000);

BEGIN

v\_result := proiect\_ocnaru\_mihai.raport\_produse\_tara\_categorie('România', 'Electronice', 3000);

IF v\_result IS NOT NULL THEN

DBMS\_OUTPUT.PUT\_LINE(v\_result);

END IF;

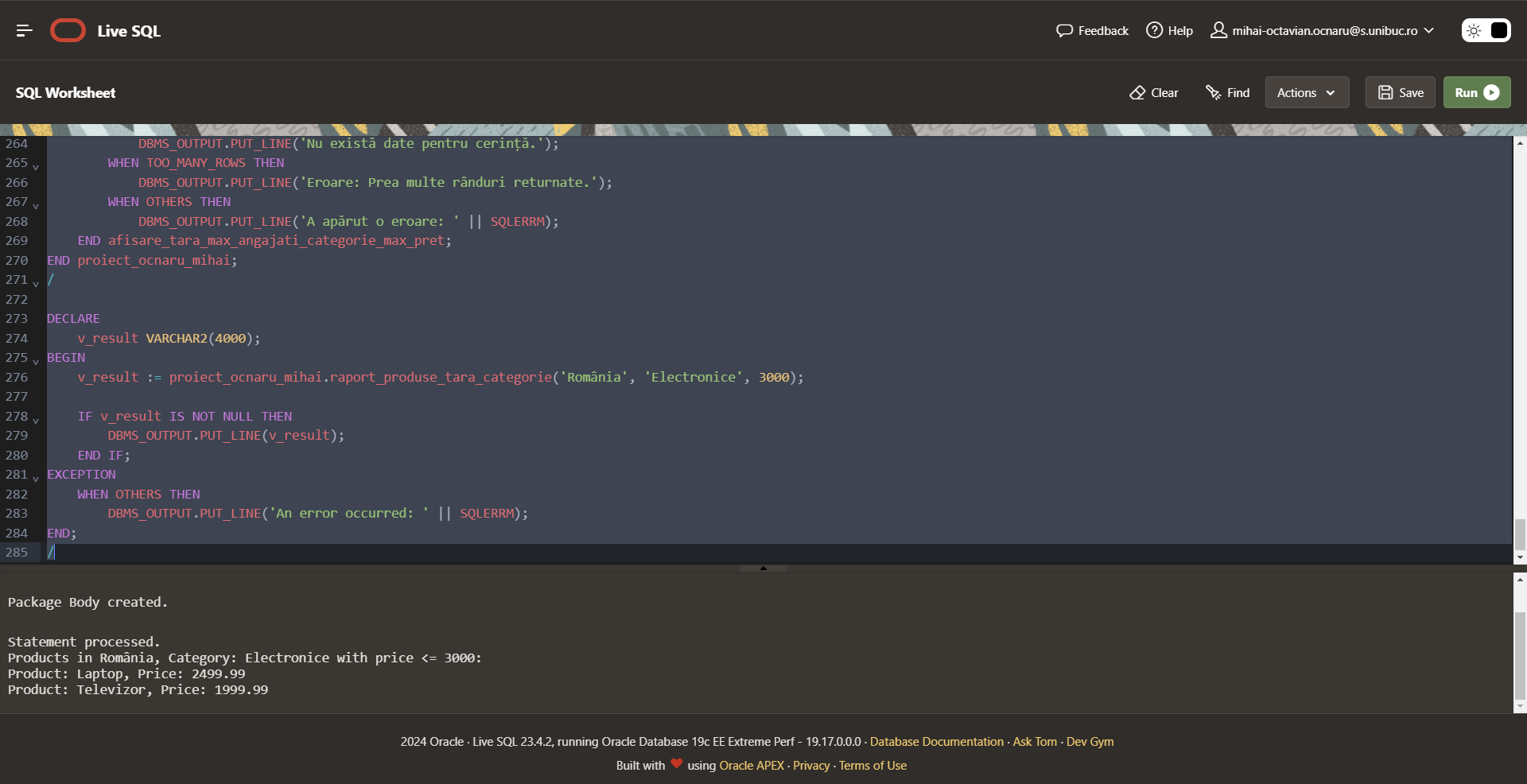
EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('An error occurred: ' || SQLERRM);

END;

/



**Apel pentru o tara inexistenta**

**Declaram codul pentru o tara inexistenta in baza de date:**

DECLARE

v\_result VARCHAR2(4000);

BEGIN

v\_result := proiect\_ocnaru\_mihai.raport\_produse\_tara\_categorie('China', 'Electronice', 3000);

IF v\_result IS NOT NULL THEN

DBMS\_OUTPUT.PUT\_LINE(v\_result);

END IF;

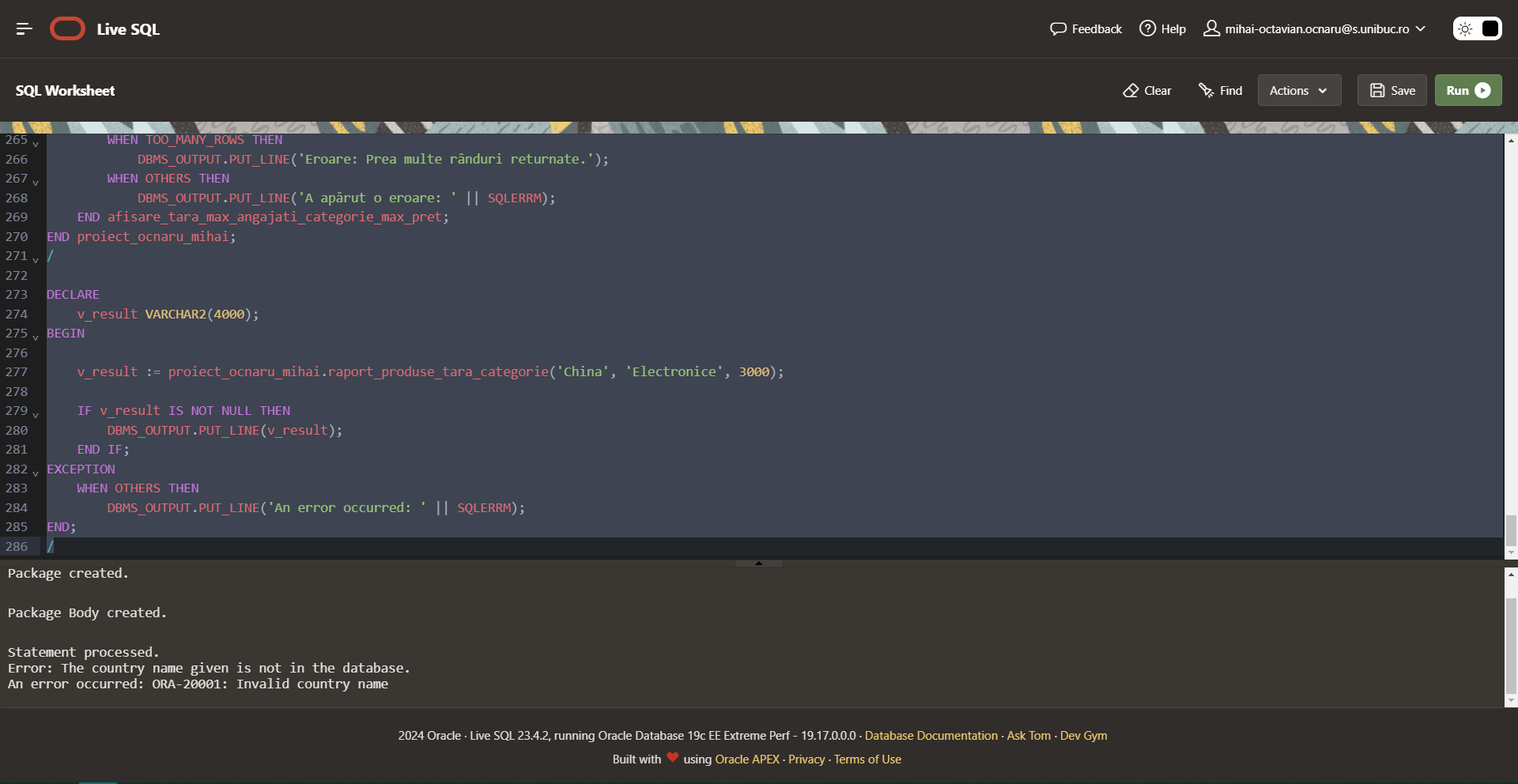
EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('An error occurred: ' || SQLERRM);

END;

/



**Apel pentru un pret maxim inexistent**

**Declaram codul pentru un pret maxim prea mic:**

DECLARE

v\_result VARCHAR2(4000);

BEGIN

v\_result := proiect\_ocnaru\_mihai.raport\_produse\_tara\_categorie('România', 'Electronice', 0.5);

IF v\_result IS NOT NULL THEN

DBMS\_OUTPUT.PUT\_LINE(v\_result);

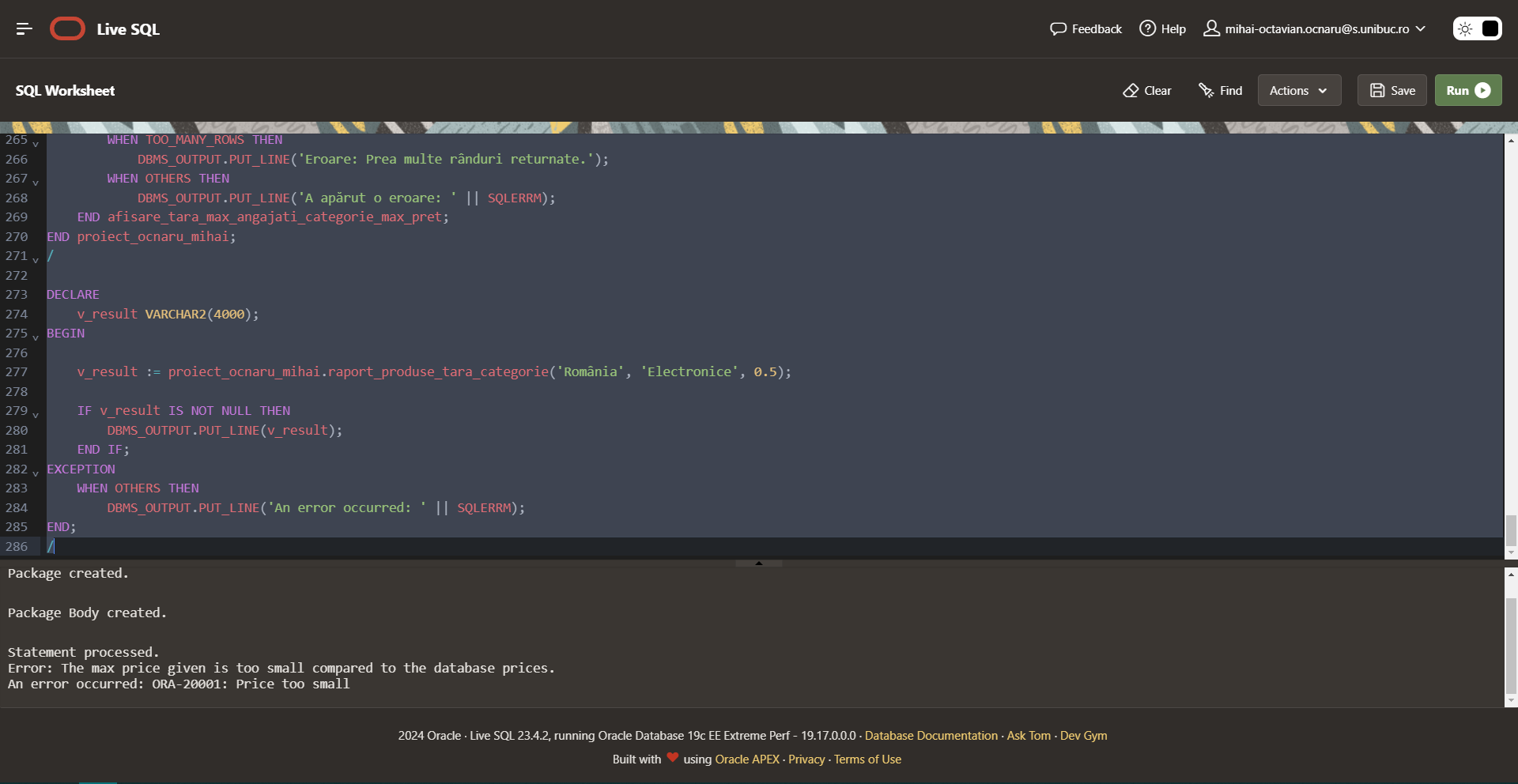
END IF;

EXCEPTION

WHEN OTHERS THEN

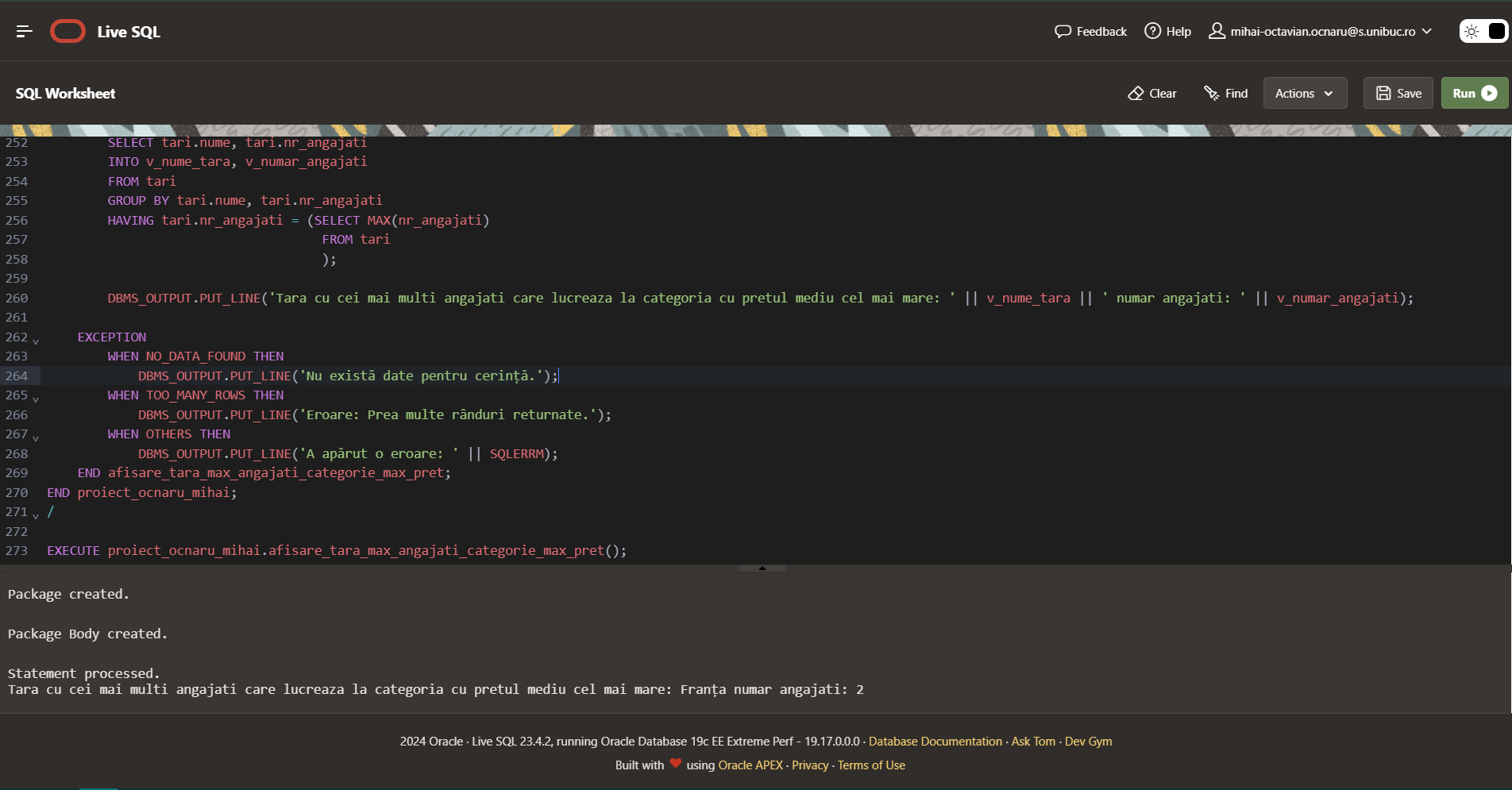
DBMS\_OUTPUT.PUT\_LINE('An error occurred: ' || SQLERRM);

END;/



**Apelam ex. 9 din pachet:**

EXECUTE proiect\_ocnaru\_mihai.afisare\_tara\_max\_angajati\_categorie\_max\_pret();



**Apel pentru TOO\_MANY\_ROWS**

BEGIN

--Pentru exemplificarea cazului TOO\_MANY\_ROWS am inserat doi

-- angajati ce lucreaza in categoria 1 si sunt situati in Spania:

INSERT INTO Angajat (cod\_angajat, prenume, nume\_familie, salariu, data\_angajarii, cod\_departament, cod\_categorie, cod\_locatie, cod\_manager)

VALUES (70, 'Cristian', 'Dinu', 9000, TO\_DATE('2019-09-21', 'yyyy-mm-dd'), 7, 1, 7, 3);

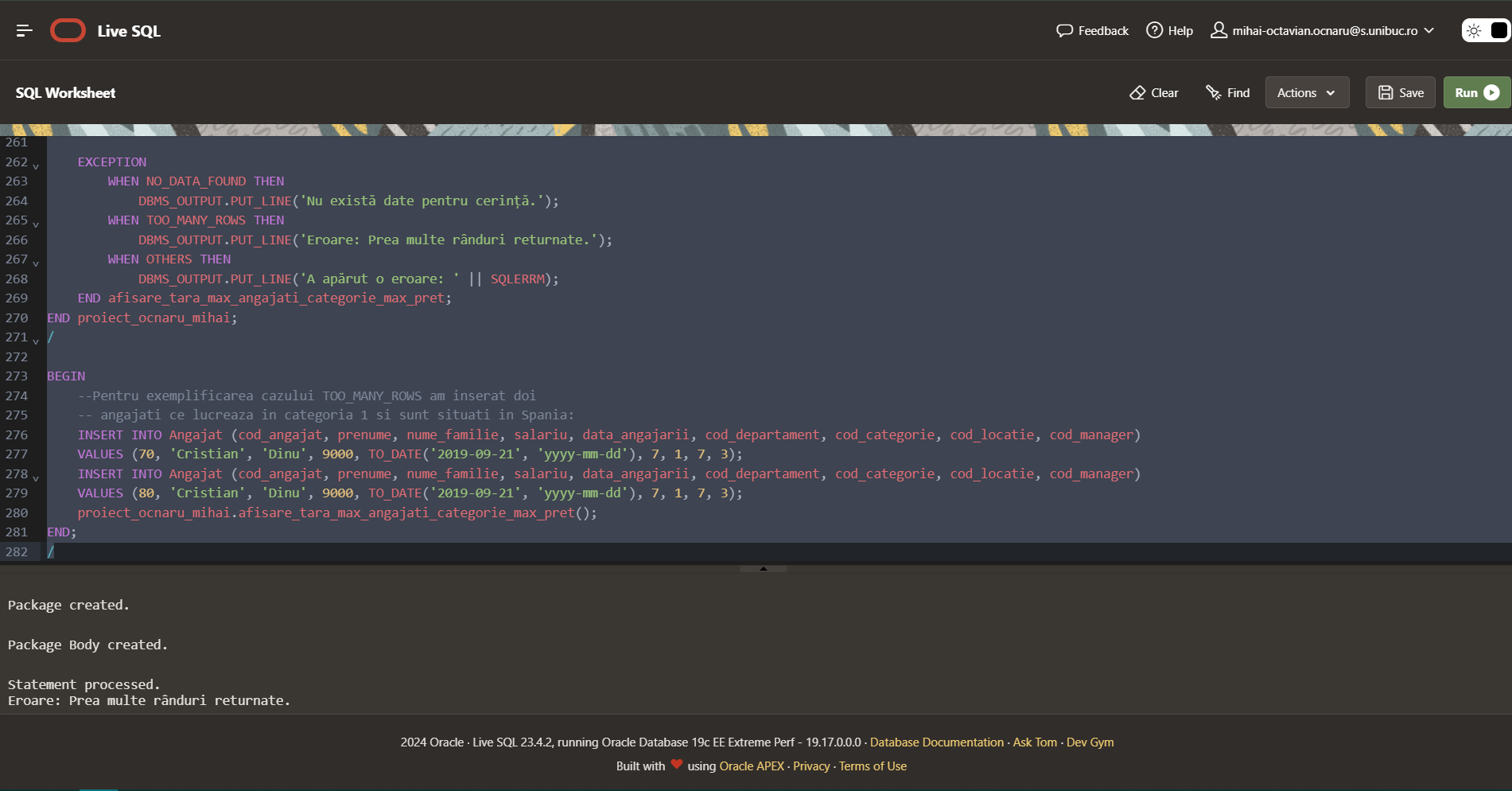
INSERT INTO Angajat (cod\_angajat, prenume, nume\_familie, salariu, data\_angajarii, cod\_departament, cod\_categorie, cod\_locatie, cod\_manager)

VALUES (80, 'Cristian', 'Dinu', 9000, TO\_DATE('2019-09-21', 'yyyy-mm-dd'), 7, 1, 7, 3);

proiect\_ocnaru\_mihai.afisare\_tara\_max\_angajati\_categorie\_max\_pret();

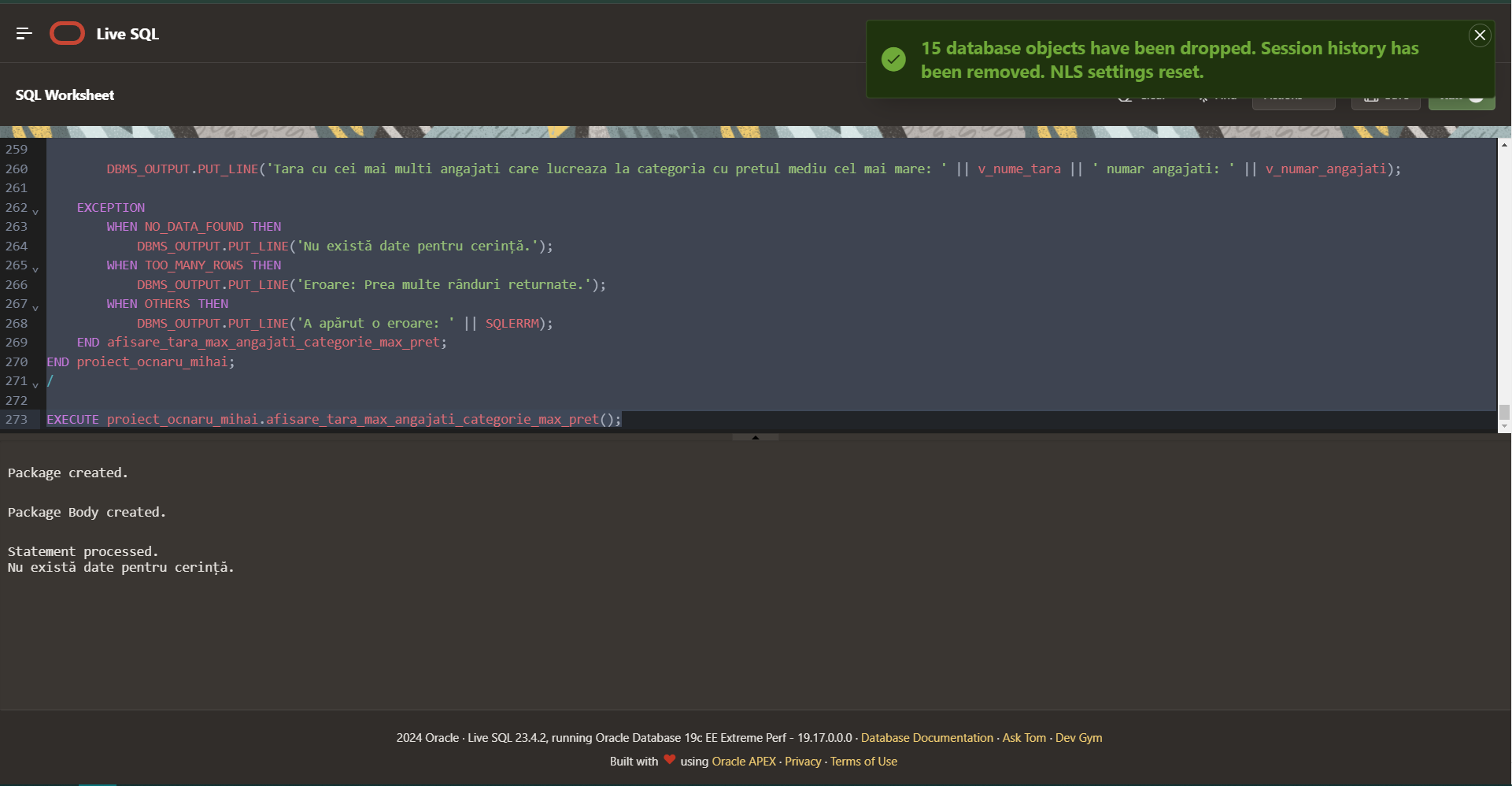
END;

/



**Apel pentru NO\_DATA\_FOUND**

Am rulat fara a face inserarea in baza de date



1. **Definiți un pachet care să includă tipuri de date complexe și obiecte necesare unui flux de acțiuni integrate, specifice bazei de date definite (minim 2 tipuri de date, minim 2 funcții, minim 2 proceduri).**

**Sa se defineasca un pachet care sa cuprinda:**

* **O procedura pentru afisarea codului comenzilor, username-ul si numele produselor pentru un utilizator dat**
* **O procedura pentru afisarea codului de review, al descrierii, al ratingului si numele produselor care au fost evaluate de un utilizator dat**
* **O functie care va returna un array ce cuprinde suma produselor dintr-o comanda, impreuna cu codul comenzii**
* **O functie ce returneaza ratingul mediu al produselor evaluate de utilizator**

CREATE OR REPLACE PACKAGE EX14\_Package AS

TYPE ProductArray is VARRAY(200) of produs.nume%Type;

TYPE OrderRecord IS RECORD (

cod\_comanda Comanda.cod\_comanda%TYPE,

username Utilizator.nume\_utilizator%TYPE,

product\_names ProductArray

);

TYPE TotalCostRecord IS RECORD (

cod\_comanda Comanda.cod\_comanda%TYPE,

total\_cost produs.pret%TYPE

);

TYPE TotalOrderCostArray is VARRAY(500) OF TotalCostRecord;

TYPE AverageRatingRecord IS RECORD (

product\_id Produs.cod\_produs%TYPE,

product\_name Produs.nume%TYPE,

average\_rating NUMBER

);

TYPE AverageRatingArray is VARRAY(500) OF AverageRatingRecord;

PROCEDURE DisplayOrderDetails(userId IN Utilizator.cod\_utilizator%TYPE);

PROCEDURE DisplayReviewData(userId IN Utilizator.cod\_utilizator%TYPE);

FUNCTION CalculateTotalCost(userId IN Utilizator.cod\_utilizator%TYPE)

RETURN TotalOrderCostArray;

FUNCTION AverageRatingForUser(userId IN Utilizator.cod\_utilizator%TYPE)

RETURN AverageRatingArray;

END EX14\_Package;

/

CREATE OR REPLACE PACKAGE BODY Ex14\_Package as

PROCEDURE DisplayOrderDetails(userId IN Utilizator.cod\_utilizator%TYPE) IS

CURSOR order\_cursor IS

SELECT c.cod\_comanda, u.nume\_utilizator, p.nume

FROM Comanda c

JOIN Utilizator u ON c.cod\_utilizator = u.cod\_utilizator

JOIN Produs\_se\_afla\_in\_Comanda pc ON c.cod\_comanda = pc.cod\_comanda

JOIN Produs p ON pc.cod\_produs = p.cod\_produs

WHERE u.cod\_utilizator = userId;

order\_rec order\_cursor%ROWTYPE;

BEGIN

OPEN order\_cursor;

LOOP

FETCH order\_cursor INTO order\_rec;

EXIT WHEN order\_cursor%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE('ID comanda: ' || order\_rec.cod\_comanda || ' | username: ' || order\_rec.nume\_utilizator || ' | nume produs: ' || order\_rec.nume);

END LOOP;

CLOSE order\_cursor;

END DisplayOrderDetails;

PROCEDURE DisplayReviewData(userId IN Utilizator.cod\_utilizator%TYPE) IS

CURSOR review\_cursor IS

SELECT r.cod\_review, r.descriere, r.rating, p.nume

FROM Review r

JOIN Produs p ON r.cod\_produs = p.cod\_produs

JOIN Utilizator u ON r.cod\_utilizator = u.cod\_utilizator

WHERE u.cod\_utilizator = userId;

review\_rec review\_cursor%ROWTYPE;

BEGIN

OPEN review\_cursor;

LOOP

FETCH review\_cursor INTO review\_rec;

EXIT WHEN review\_cursor%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE('ID review: ' || review\_rec.cod\_review || ' | descriere: ' || review\_rec.descriere || ' | rating: ' || review\_rec.rating || ' | nume produs: ' || review\_rec.nume);

END LOOP;

CLOSE review\_cursor;

END DisplayReviewData;

FUNCTION CalculateTotalCost(userId IN Utilizator.cod\_utilizator%TYPE)

RETURN TotalOrderCostArray IS

total\_cost\_array TotalOrderCostArray := TotalOrderCostArray();

total\_cost\_record TotalCostRecord;

total\_cost produs.pret%TYPE;

BEGIN

FOR order\_rec IN (

SELECT c.cod\_comanda, SUM(p.pret) AS total\_cost

FROM Comanda c

JOIN Utilizator u ON c.cod\_utilizator = u.cod\_utilizator

JOIN Produs\_se\_afla\_in\_Comanda pc ON c.cod\_comanda = pc.cod\_comanda

JOIN Produs p ON pc.cod\_produs = p.cod\_produs

WHERE u.cod\_utilizator = userId

GROUP BY c.cod\_comanda

) LOOP

total\_cost\_record.cod\_comanda := order\_rec.cod\_comanda;

total\_cost\_record.total\_cost := order\_rec.total\_cost;

total\_cost\_array.extend;

total\_cost\_array(total\_cost\_array.last) := total\_cost\_record;

END LOOP;

RETURN total\_cost\_array;

END CalculateTotalCost;

FUNCTION AverageRatingForUser(userId IN Utilizator.cod\_utilizator%TYPE)

RETURN AverageRatingArray IS

average\_rating\_array AverageRatingArray := AverageRatingArray();

average\_rating\_record AverageRatingRecord;

average\_rating NUMBER;

BEGIN

FOR product\_rec IN (

SELECT p.cod\_produs, p.nume

FROM Produs p

JOIN Review r ON p.cod\_produs = r.cod\_produs

JOIN Utilizator u ON r.cod\_utilizator = u.cod\_utilizator

WHERE u.cod\_utilizator = userId

GROUP BY p.cod\_produs, p.nume

) LOOP

SELECT AVG(r.rating) INTO average\_rating

FROM Review r

JOIN Produs p ON r.cod\_produs = p.cod\_produs

JOIN Utilizator u ON r.cod\_utilizator = u.cod\_utilizator

WHERE u.cod\_utilizator = userId AND p.cod\_produs = product\_rec.cod\_produs;

average\_rating\_record.product\_id := product\_rec.cod\_produs;

average\_rating\_record.product\_name := product\_rec.nume;

average\_rating\_record.average\_rating := average\_rating;

average\_rating\_array.extend;

average\_rating\_array(average\_rating\_array.last) := average\_rating\_record;

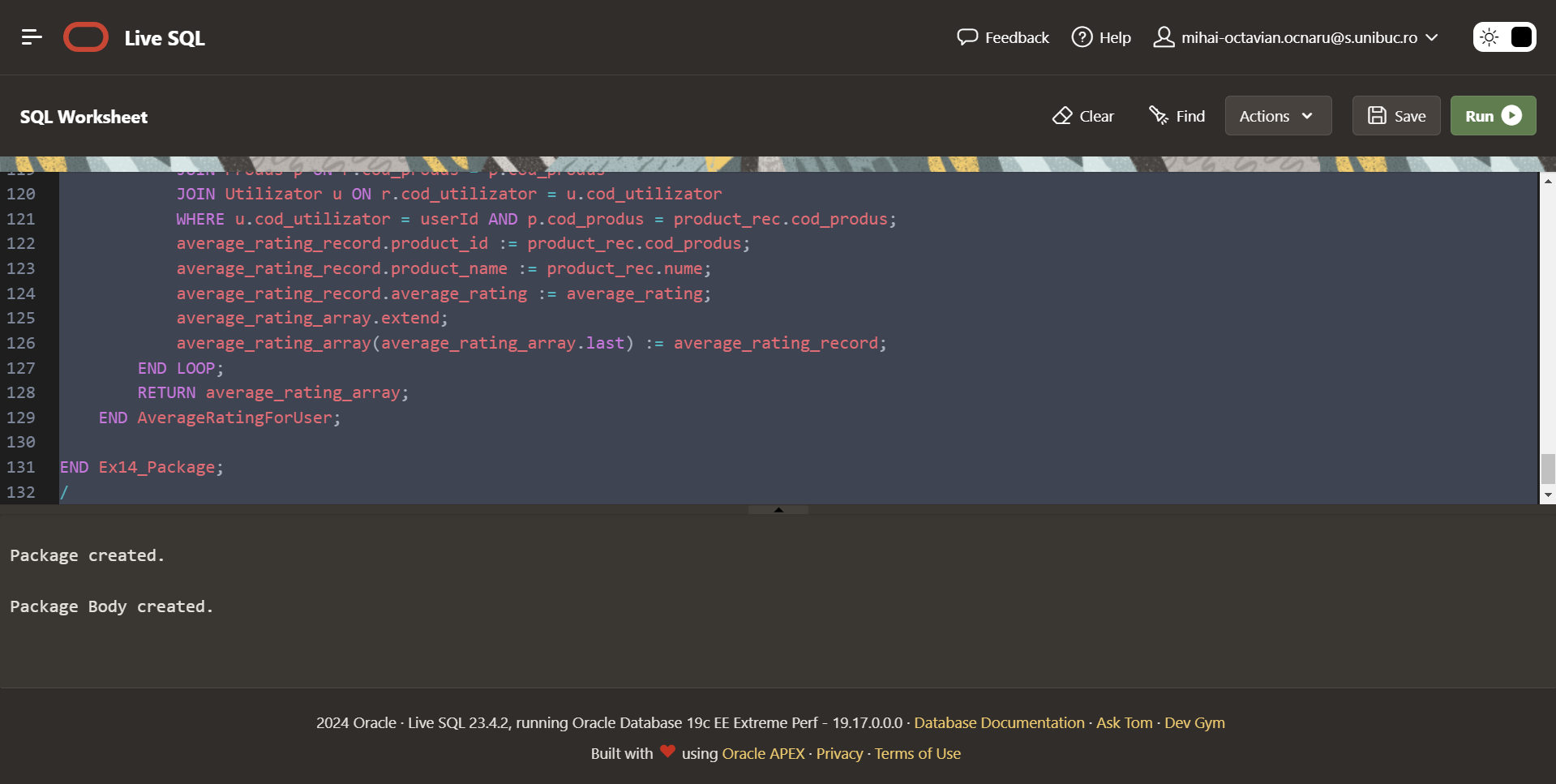
END LOOP;

RETURN average\_rating\_array;

END AverageRatingForUser;

END Ex14\_Package;

/



**In continuare vom testa functionalitatea pentru un utilizator dat:**

DECLARE

userId Utilizator.cod\_utilizator%TYPE := 3;

total\_cost\_array Ex14\_Package.TotalOrderCostArray;

average\_rating\_array Ex14\_Package.AverageRatingArray;

BEGIN

DBMS\_OUTPUT.PUT\_LINE('Afisare detalii comenzi si review-uri pentru utilizatorul cu ID-ul ' || userId);

Ex14\_Package.DisplayOrderDetails(userId);

DBMS\_OUTPUT.PUT\_LINE('------------');

DBMS\_OUTPUT.PUT\_LINE('Afisare rating pentru fiecare produs evaluat de utilizatorul cu ID-ul ' || userId);

Ex14\_Package.DisplayReviewData(userId);

DBMS\_OUTPUT.PUT\_LINE('------------ ');

DBMS\_OUTPUT.PUT\_LINE('Afisare cost total pentru fiecare comanda a utilizatorului cu ID-ul ' || userId);

total\_cost\_array := Ex14\_Package.CalculateTotalCost(userId);

FOR i IN 1..total\_cost\_array.count LOOP

DBMS\_OUTPUT.PUT\_LINE('ID comanda: ' || total\_cost\_array(i).cod\_comanda || ' | cost total: ' || total\_cost\_array(i).total\_cost);

END LOOP;

DBMS\_OUTPUT.PUT\_LINE('------------ ');

DBMS\_OUTPUT.PUT\_LINE('Afisare rating mediu pentru fiecare produs evaluat de utilizatorul cu ID-ul ' || userId);

average\_rating\_array := Ex14\_Package.AverageRatingForUser(userId);

FOR i IN 1..average\_rating\_array.count LOOP

DBMS\_OUTPUT.PUT\_LINE('ID produs: ' || average\_rating\_array(i).product\_id || ' | nume produs: ' || average\_rating\_array(i).product\_name || ' | rating mediu: ' || average\_rating\_array(i).average\_rating);

END LOOP;

END;

/

