**PROIECT SGBD**

**Adapost de Animale**

Prof. Laborator: Marchidon A.

Prof. Curs: Mihai G.

**Mocanu Alexandra**

**Grupa 232**

**CUPRINS**

Descriere model real……………………………………………………………………..……………3

Constrangeri………………………………………………………………………………..……………..3

Entitati…………………………………………………………………………………………..……………4

Relatii si cardinalitate…………………………………………………………………..……………..6

Atribute………………………………………………………………………………………..…………….7

Diagrama entitate-relatie……………………………………………………………..….…………9

Diagrama conceptuala….…………………………………………………………..………………..9

Definirea tabelelor…..………………………….………………………………….…………………10

Inserare…………………….………………………………………………………….……….…………..14

Subprogram stocat independent cu 3 tipuri de colectii ………………..……….……21

Subprogram stocat independent cu 2 tipuri de cursoare…………….………………24

Subprogram stocat independent de tip functie……….. ………………..………………26

subprogram stocat independent de tip procedura………………………………………29

Trigger LMD(nivel comanda)……………………………………………………………………….33

Trigger LMD(nivel line)……………………………………….……………………………………….34

Trigger LDD………………………………………………………………………..……………………….36

Pachet cu toate obiectele definite……………………………..……………………………….38

Pachet cu tipuri de date complexe……………………………………………………………...40

**PROIECT BAZE DE DATE - ADAPOST DE ANIMALE**

1. Prezentarea pe scurt a bazei de date.

Acest proiect urmareste realizarea design-ului unei baze de date a unui adapost de animale. Utilitatea acestui design este aceea ca se poate tine cu usurinta evidenta angajatilor (veterinari, directori) din adapost, dar si a unor consumabile (medicamente, hrana), sau documente (cereri de adoptie).

In ceea ce priveste organizarea, adapostul este impartit in sectii, iar fiecare sectie exista sectoare in care se afla custile animalelor adapostite. Veterinarii pot face consultatii si noteaza in carnetul veterinary al fiecarui animal toate datele necesare.

1. Prezentarea constrangerilor(restrictii, regului) impuse asupra modelului.

Constrângerile din codul furnizat includ: chei primare, chei externe, restricții de referință, chei primare compuse, restricții de unicitate și restricții de integritate referențială. Acestea asigură unicitatea valorilor, relațiile între tabele, respectarea valorilor cheilor primare și integritatea datelor.

Cheie primară: O constrângere de cheie primară asigură că o coloană sau un set de coloane dintr-o tabelă au valori unice și identifică în mod unic fiecare înregistrare. De exemplu, o constrângere de cheie primară ar putea fi aplicată coloanei "id" dintr-o tabelă "Angajat", astfel încât fiecare angajatsă aibă un ID unic.

Cheie externă: O constrângere de cheie externă definește relații între tabele, asigurând integritatea referențială. De exemplu, o constrângere de cheie externă ar putea fi aplicată coloanei "id\_angajat" dintr-o tabelă "Contract", referindu-se la coloana "id" din tabela "Angajat", pentru a asigura că fiecare comandă este asociată cu un utilizator valid.

Restricție de unicitate: O restricție de unicitate asigură că o coloană sau un set de coloane dintr-o tabelă nu conține valori duplicate. De exemplu, o restricție de unicitate ar putea fi aplicată coloanei "CNP" dintr-o tabelă "Vizitator", astfel încât fiecare vizitator să aibe un CNP unic.

1. Descrierea entitatilor, incluzand precizarea cheii primare.

Tabela CONTRACT:

Cheia primară: NUMAR (de tip INT)

Tabela ANGAJAT:

Cheia primară: ID\_ANGAJAT (de tip INT)

Tabela DIRECTOR:

Cheia primară: ID\_DIRECTOR (de tip INT)

Tabela SECTIE:

Cheia primară: ID\_SECTIE (de tip INT)

Tabela SECTOR:

Cheia primară: ID\_SECTOR (de tip INT)

Tabela SECTIE\_SECTOR:

Cheile primare: ID\_SECTIE și ID\_SECTOR (ambele de tip INT)

Tabela FARMACIE:

Cheia primară: ID\_FARMACIE (de tip INT)

Tabela MEDICAMENT:

Cheia primară: NUMAR (de tip INT)

Tabela VIZITATOR:

Cheia primară: CNP (de tip VARCHAR(13))

Tabela CERERE\_ADOPTIE:

Cheia primară: SERIE, ID\_ANIMAL, ID\_VIZITATOR (toate de tip VARCHAR(255), INT, VARCHAR(13) respectiv)

Tabela ANIMAL:

Cheia primară: ID\_ANIMAL (de tip INT)

Tabela CUSCA:

Cheia primară: ID\_CUSCA (de tip INT)

Tabela VETERINAR:

Cheia primară: CNP (de tip VARCHAR(13))

Tabela MAGAZIN:

Cheia primară: ID\_MAGAZIN (de tip INT)

Tabela HRANA:

Cheia primară: NUMAR (de tip INT)

Tabela CARTE\_DE\_SANATATE:

Cheia primară: ID\_ANIMAL (de tip INT)

Tabela CONSULTATIE:

Cheia primară: ID\_CONSULTATIE (de tip INT)

Tabela CLINICA\_VETERINARA:

Cheia primară: ID\_CLINICA (de tip INT)

Vizualizarea v\_angajati\_sectii:

Această vizualizare nu are o cheie primară, deoarece este o vizualizare rezultată din interogarea datelor din mai multe tabele.

1. Descrierea relatiilor, incluzand precizarea cardinalitatii acestora.

Tabelul CONTRACT are o relație de tip 1 la N cu tabelul ANGAJAT. Un contract poate fi asociat unui singur angajat, iar un angajat poate avea asociat unul sau mai multe contracte.

Tabelul ANGAJAT are o relație de tip 1 la 1 cu tabelul CONTRACT. Fiecare angajat este asociat cu un singur contract.

Tabelul DIRECTOR are o relație de tip 1 la 1 cu tabelul CONTRACT. Fiecare director este asociat cu un singur contract.

Tabelul SECTIE are o relatie cu tabela Angajat si una cu tabela SECTIE\_SECTOR.

Tabelul SECTOR are relatie atat cu tablea SECTIE\_SECTOR cat si cu tabela CUSCA.

Tabelul SECTIE\_SECTOR are o relație de tip N la N cu tabelele SECTIE și SECTOR. O sectie poate fi asociata cu mai multe sectoare, iar un sector poate fi asociat cu mai multe sectii.

Tabelul FARMACIE are relațiE cu tabelele CONTRACT si FARMACIE.

Tabelul MEDICAMENT are o relație de tip 1 la N cu tabelul FARMACIE. Un medicament poate fi asociat unei singure farmacii, iar o farmacie poate avea asociate unul sau mai multe medicamente.

Tabelul VIZITATOR nu are relații cu alte tabele.

Tabelul CERERE\_ADOPTIE are o relație de tip N la N cu tabelele ANIMAL și VIZITATOR. O cerere de adoptie poate fi asociata cu mai multe animale si vizitatori, iar un animal si un vizitator pot fi asociati cu mai multe cereri de adoptie.

Tabelul ANIMAL are o relație de tip 1 la 1 cu tabelul CUSCA. Fiecare animal este asociat cu o singură cusca.

Tabelul CUSCA are relații cu tabelele SECTOR si ANIMAL.

Tabelul VETERINAR are o relație de tip 1 la N cu tabelul ANIMAL. Un veterinar poate fi asociat cu unul sau mai multe animale, iar un animal poate fi asociat cu un singur veterinar.

Tabelul MAGAZIN are relații cu tabelea HRANA si tabela CONTRACT.

Tabelul HRANA are o relație de tip 1 la N cu tabelul MAGAZIN. Un produs de hrana poate fi asociat unui singur magazin, iar un magazin poate avea asociate unul sau mai multe produse de hrana.

Tabelul CARTE\_DE\_SANATATE are o relație de tip 1 la 1 cu tabelul ANIMAL. Fiecare animal are asociată o singură carte de sănătate.

Tabelul CONSULTATIE are relatie cu tabela ANIMALl si cu tabela VETERINAR.

Tabelul CLINICA\_VETERINARA are relatie cu tabelele VETERINAR si tabela CONTRACT

1. Descrierea atributelor, incluzand tipul de date si eventualele constrangeri, valori implicite, valori posibile ale atributelor.

Tabelul CONTRACT:

Atribute: NUMAR (cheie primară), DATA\_EMITERE, DURATA, TIP, id\_angajat (cheie externă referențiază ANGAJAT.ID\_ANGAJAT).

Tabelul ANGAJAT:

Atribute: ID\_ANGAJAT (cheie primară), NUME, PRENUME, CNP, ADRESA, TELEFON, EMAIL, ID\_CONTRACT (cheie externă referențiază CONTRACT.NUMAR).

Tabelul DIRECTOR:

Atribute: ID\_DIRECTOR (cheie primară), ID\_CONTRACT (cheie externă referențiază CONTRACT.NUMAR).

Tabelul SECTIE:

Atribute: ID\_SECTIE (cheie primară), NUME.

Tabelul SECTOR:

Atribute: ID\_SECTOR (cheie primară), TIP\_NEVOIE, NUMAR\_LOCURI.

Tabelul SECTIE\_SECTOR:

Atribute: ID\_SECTIE, ID\_SECTOR (cheie primară compusă), ID\_SECTIE (cheie externă referențiază SECTIE.ID\_SECTIE), ID\_SECTOR (cheie externă referențiază SECTOR.ID\_SECTOR).

Tabelul FARMACIE:

Atribute: ID\_FARMACIE (cheie primară), NUME, TELEFON, PROGRAM.

Tabelul MEDICAMENT:

Atribute: NUMAR (cheie primară), DENUMIRE\_PRODUS, CANTITATE, ID\_FARMACIE (cheie externă referențiază FARMACIE.ID\_FARMACIE).

Tabelul VIZITATOR:

Atribute: CNP (cheie primară), NUME, PRENUME, TELEFON, EMAIL.

Tabelul CERERE\_ADOPTIE:

Atribute: SERIE, ID\_ANIMAL, ID\_VIZITATOR, DATA, APROBATA\_DE\_DIRECTOR, APROBATA\_DE\_ASOCIATIE (cheie primară compusă), ID\_ANIMAL (cheie externă referențiază ANIMAL.ID\_ANIMAL), ID\_VIZITATOR (cheie externă referențiază VIZITATOR.CNP).

Tabelul ANIMAL:

Atribute: ID\_ANIMAL (cheie primară), NUME, RASA, VARSTA, SEX, DATA\_SOSIRE, ID\_CUSCA (cheie externă referențiază CUSCA.ID\_CUSCA), ID\_VETERINAR (cheie externă referențiază VETERINAR.CNP).

Tabelul CUSCA:

Atribute: ID\_CUSCA (cheie primară), NUMAR\_LOCURI.

Tabelul VETERINAR:

Atribute: CNP (cheie primară), NUME, PRENUME, ID\_ANIMAL (cheie externă referențiază ANIMAL.ID\_ANIMAL).

Tabelul MAGAZIN:

Atribute: ID\_MAGAZIN (cheie primară), NUME, TELEFON, PROGRAM.

Tabelul HRANA:

Atribute: ID\_HRANA (cheie primară), DENUMIRE, CANTITATE, ID\_MAGAZIN (cheie externă referențiază MAGAZIN.ID\_MAGAZIN).

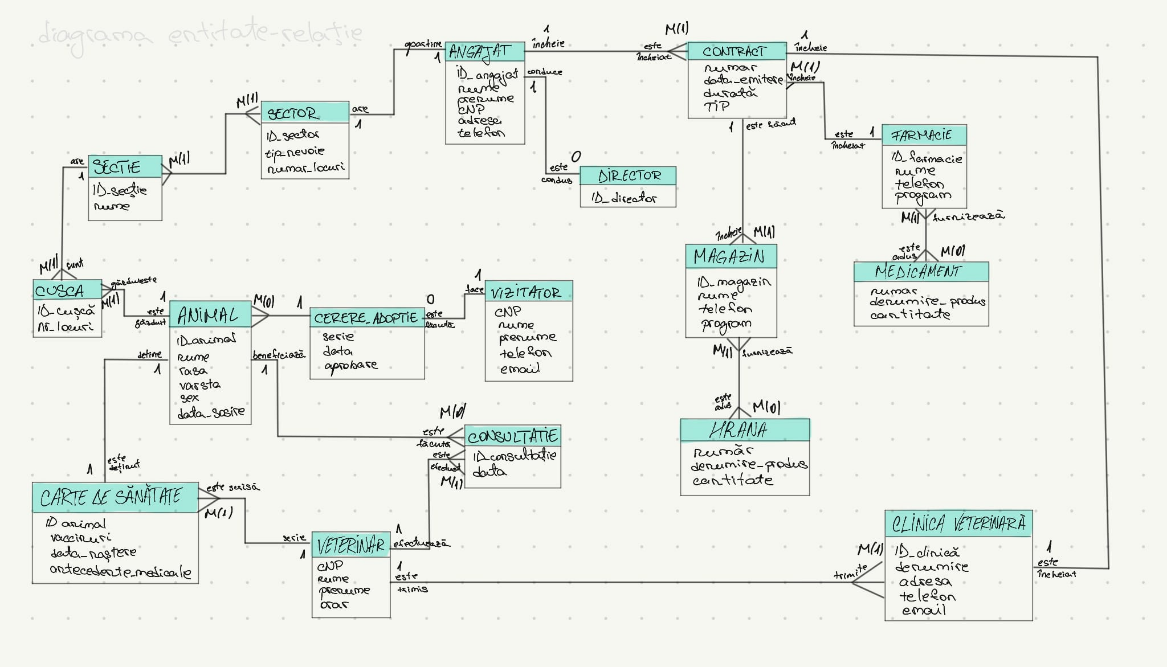
Tabelul COMANDA:

Atribute: ID\_COMANDA (cheie primară), DATA, SUMA\_TOTALA, ID\_MAGAZIN (cheie externă referențiază MAGAZIN.ID\_MAGAZIN).

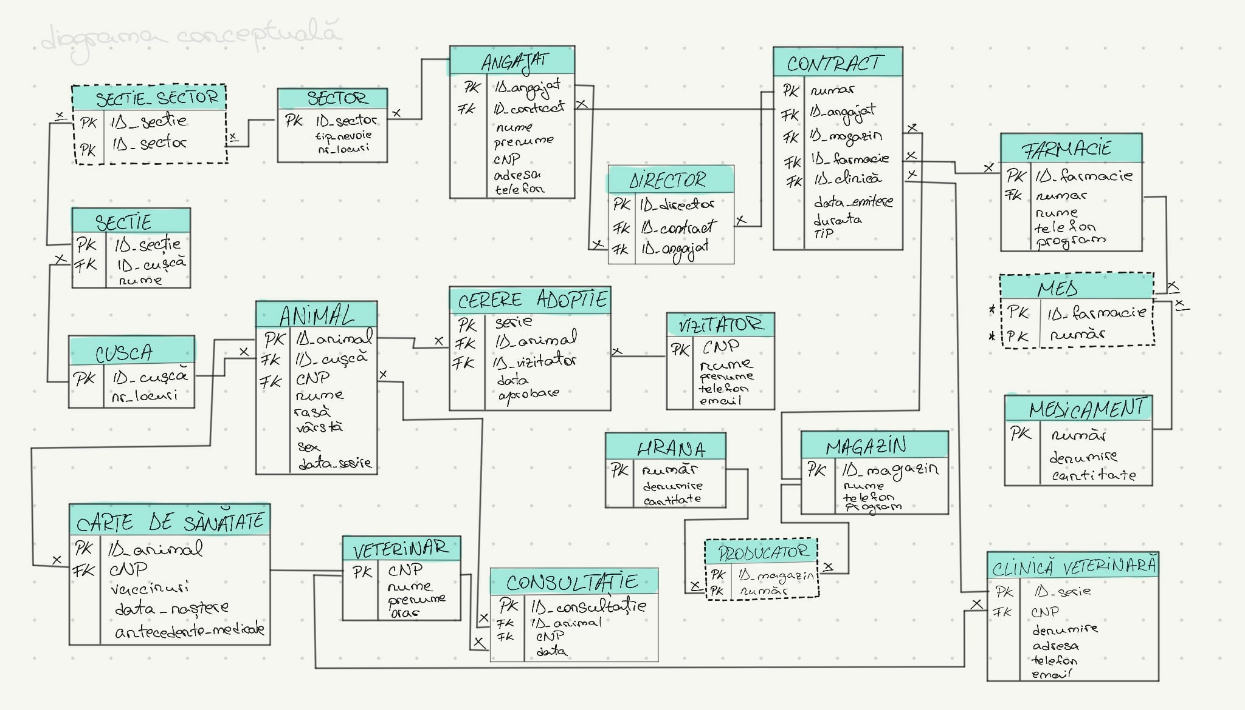
Tabelul DETALII\_COMANDA:

Atribute: ID\_COMANDA, ID\_HRANA (cheie primară compusă), ID\_COMANDA (cheie externă referențiază COMANDA.ID\_COMANDA), ID\_HRANA (cheie externă referențiază HRANA.ID\_HRANA).

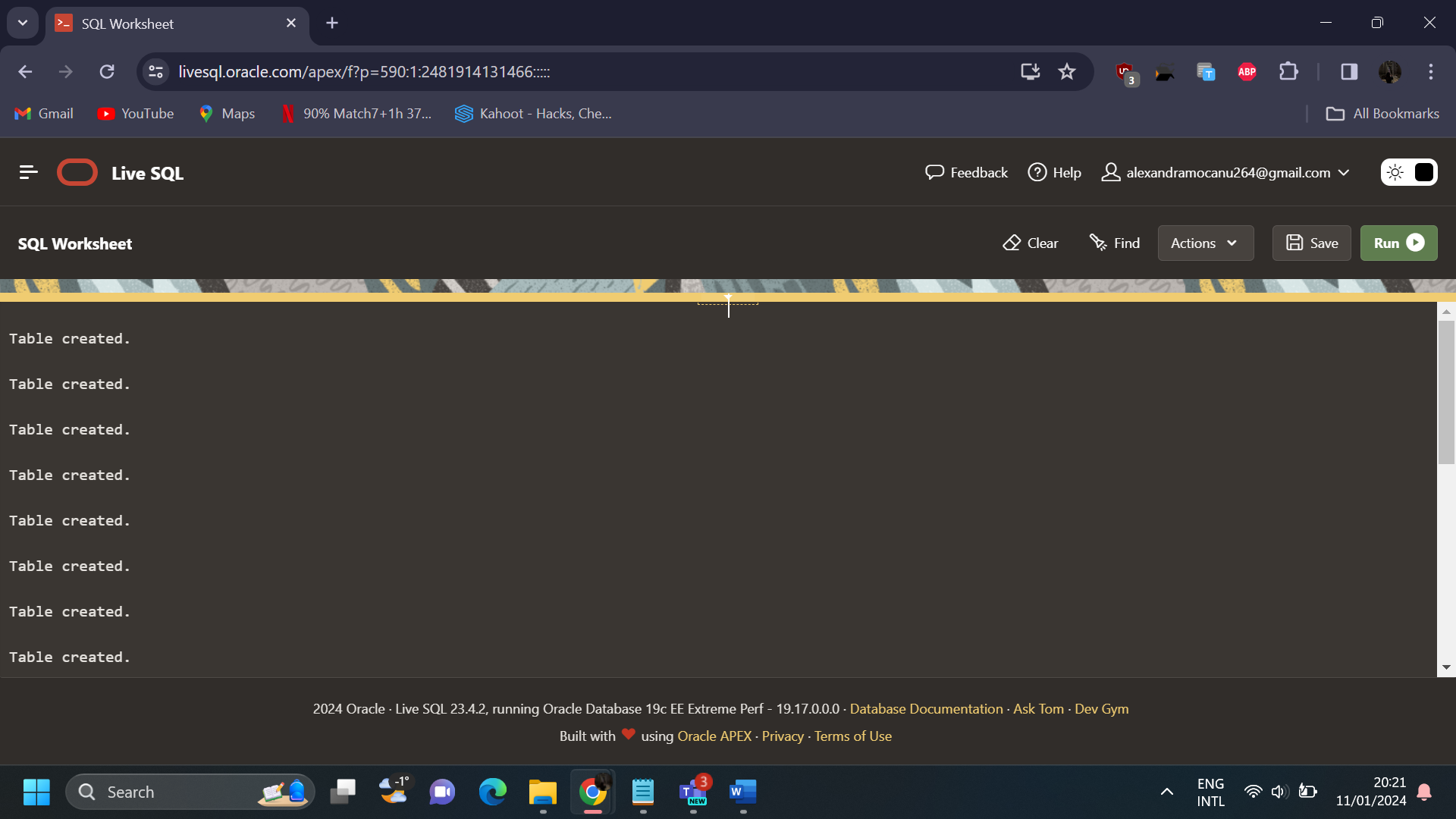
1. Diagrama entitate-relatie



1. Diagrama conceptuala:



1. Implementarea in Oracle a diagramei conceptuale realizate anterior



CREATE TABLE CUSCA (

ID\_CUSCA INT PRIMARY KEY,

NUMAR\_LOCURI INT

);

CREATE TABLE SECTOR (

ID\_SECTOR INT PRIMARY KEY,

TIP\_NEVOIE VARCHAR(255),

NUMAR\_LOCURI INT

);

CREATE TABLE SECTIE (

ID\_SECTIE INT PRIMARY KEY,

NUME VARCHAR(255),

ID\_CUSCA INT,

FOREIGN KEY (ID\_CUSCA) REFERENCES CUSCA (ID\_CUSCA)

);

CREATE TABLE SECTIE\_SECTOR (

ID\_SECTIE INT,

ID\_SECTOR INT,

PRIMARY KEY (ID\_SECTIE, ID\_SECTOR),

FOREIGN KEY (ID\_SECTIE) REFERENCES SECTIE (ID\_SECTIE),

FOREIGN KEY (ID\_SECTOR) REFERENCES SECTOR (ID\_SECTOR)

);

CREATE TABLE MEDICAMENT (

NUMAR INT PRIMARY KEY,

DENUMIRE\_PRODUS VARCHAR(255),

CANTITATE INT

);

CREATE TABLE FARMACIE (

ID\_FARMACIE INT PRIMARY KEY,

NUME VARCHAR(255),

TELEFON VARCHAR(255),

PROGRAM VARCHAR(255),

NUMAR INT,

FOREIGN KEY (NUMAR) REFERENCES MEDICAMENT (NUMAR)

);

CREATE TABLE MED(

ID\_FARMACIE INT,

NUMAR INT,

PRIMARY KEY(ID\_FARMACIE, NUMAR),

FOREIGN KEY (ID\_FARMACIE) REFERENCES FARMACIE (ID\_FARMACIE),

FOREIGN KEY (NUMAR) REFERENCES MEDICAMENT (NUMAR)

);

CREATE TABLE MAGAZIN (

ID\_MAGAZIN INT PRIMARY KEY,

ID\_CONTRACT INT,

NUME VARCHAR(255),

TELEFON VARCHAR(255),

PROGRAM VARCHAR(255)

);

CREATE TABLE HRANA (

NUMAR INT PRIMARY KEY,

DENUMIRE\_PRODUS VARCHAR(255),

CANTITATE INT,

ID\_MAGAZIN INT,

FOREIGN KEY (ID\_MAGAZIN) REFERENCES MAGAZIN (ID\_MAGAZIN)

);

CREATE TABLE VIZITATOR (

CNP VARCHAR(13) PRIMARY KEY,

NUME VARCHAR(255),

PRENUME VARCHAR(255),

TELEFON VARCHAR(255),

EMAIL VARCHAR(255)

);

CREATE TABLE VETERINAR (

CNP VARCHAR(13) PRIMARY KEY,

NUME VARCHAR(255),

PRENUME VARCHAR(255)

);

CREATE TABLE ANIMAL (

ID\_ANIMAL INT PRIMARY KEY,

NUME VARCHAR(255),

RASA VARCHAR(255),

VARSTA INT,

SEX VARCHAR(255),

DATA\_SOSIRE DATE,

ID\_CUSCA INT,

ID\_VETERINAR VARCHAR(13),

FOREIGN KEY (ID\_CUSCA) REFERENCES CUSCA (ID\_CUSCA),

FOREIGN KEY (ID\_VETERINAR) REFERENCES VETERINAR (CNP)

);

CREATE TABLE CARTE\_DE\_SANATATE (

ID\_ANIMAL INT PRIMARY KEY,

VACCINURI VARCHAR(255),

DATA\_NASTERE DATE,

ANTECEDENTE\_MEDICALE VARCHAR(255),

ID\_VETERINAR VARCHAR(13),

FOREIGN KEY (ID\_VETERINAR) REFERENCES VETERINAR (CNP)

);

CREATE TABLE CONSULTATIE (

ID\_CONSULTATIE INT PRIMARY KEY,

ID\_ANIMAL INT,

DATA DATE,

ID\_VETERINAR VARCHAR(13),

FOREIGN KEY (ID\_VETERINAR) REFERENCES VETERINAR (CNP)

);

CREATE TABLE CERERE\_ADOPTIE (

SERIE VARCHAR(255),

ID\_ANIMAL INT,

ID\_VIZITATOR VARCHAR(13),

DATA DATE,

--APROBATA\_DE\_DIRECTOR BOOLEAN DEFAULT FALSE,

PRIMARY KEY (SERIE, ID\_ANIMAL, ID\_VIZITATOR),

FOREIGN KEY (ID\_ANIMAL) REFERENCES ANIMAL (ID\_ANIMAL),

FOREIGN KEY (ID\_VIZITATOR) REFERENCES VIZITATOR (CNP)

);

CREATE TABLE CLINICA\_VETERINARA (

ID\_CLINICA INT PRIMARY KEY,

DENUMIRE VARCHAR(255),

ADRESA VARCHAR(255),

TELEFON VARCHAR(255),

EMAIL VARCHAR(255),

ID\_VETERINAR VARCHAR(13),

FOREIGN KEY (ID\_VETERINAR) REFERENCES VETERINAR (CNP)

);

CREATE TABLE CONTRACT (

NUMAR INT PRIMARY KEY,

DATA\_EMITERE DATE,

DURATA INT,

TIP VARCHAR(255),

id\_magazin INT,

id\_farmacie INT,

id\_clinica INT,

FOREIGN KEY (id\_magazin) REFERENCES MAGAZIN (id\_magazin),

FOREIGN KEY (id\_farmacie) REFERENCES FARMACIE (id\_farmacie),

FOREIGN KEY (id\_clinica) REFERENCES CLINICA\_VETERINARA (ID\_CLINICA)

);

CREATE TABLE ANGAJAT (

ID\_ANGAJAT INT PRIMARY KEY,

NUME VARCHAR(255),

PRENUME VARCHAR(255),

CNP VARCHAR(13),

ADRESA VARCHAR(255),

TELEFON VARCHAR(255),

EMAIL VARCHAR(255),

ID\_CONTRACT INT,

FOREIGN KEY (ID\_CONTRACT) REFERENCES CONTRACT (NUMAR)

);

CREATE TABLE DIRECTOR (

ID\_DIRECTOR INT,

ID\_CONTRACT INT,

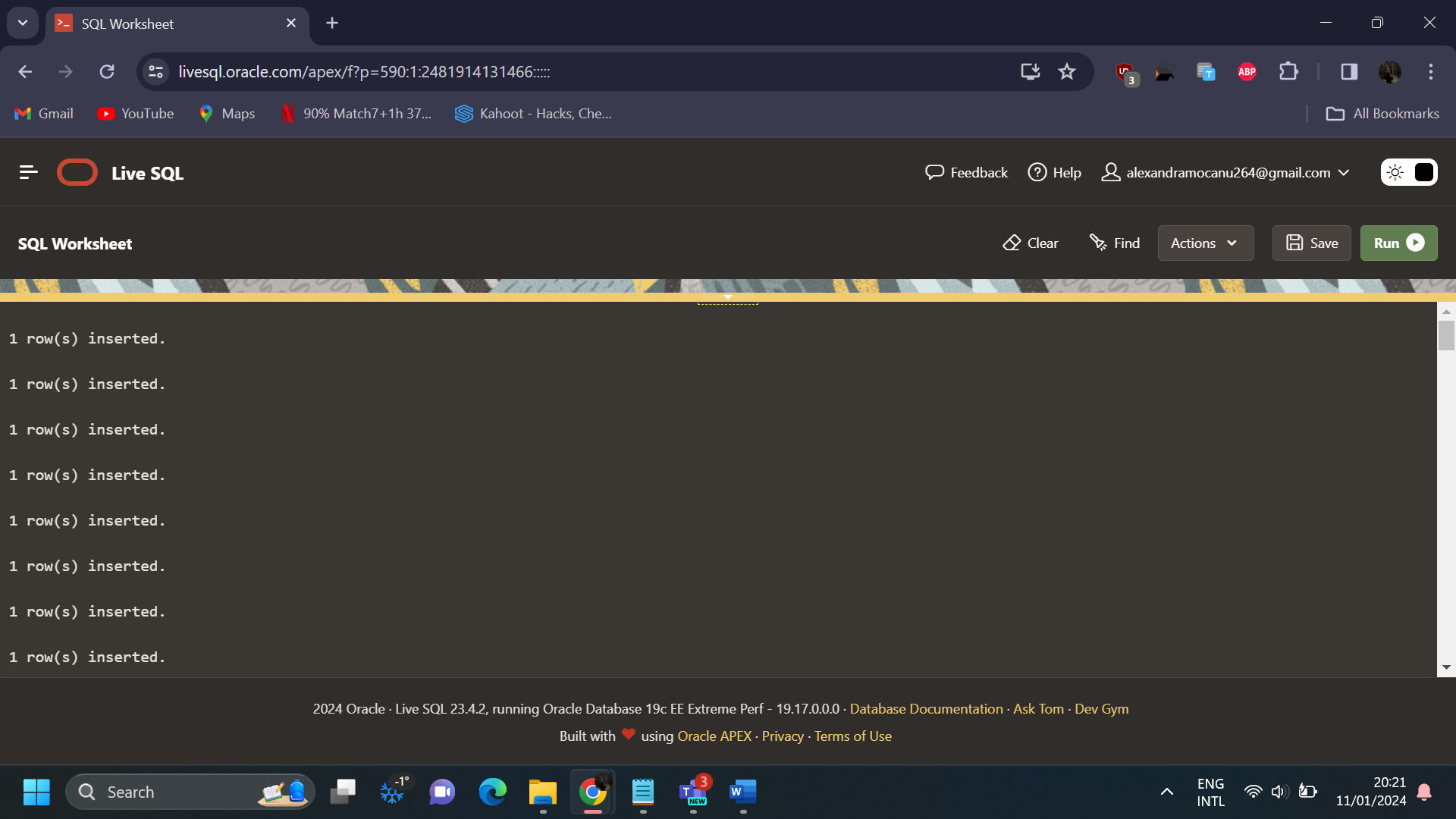
ID\_ANGAJAT INT,

FOREIGN KEY (ID\_CONTRACT) REFERENCES CONTRACT (NUMAR),

FOREIGN KEY (ID\_ANGAJAT) REFERENCES ANGAJAT (ID\_ANGAJAT)

);

1. Adaugarea informatiilor in tebelele create.



-- Tabela CUSCA --

INSERT INTO CUSCA (ID\_CUSCA, NUMAR\_LOCURI)

VALUES (1, 5);

INSERT INTO CUSCA (ID\_CUSCA, NUMAR\_LOCURI)

VALUES (2, 10);

INSERT INTO CUSCA (ID\_CUSCA, NUMAR\_LOCURI)

VALUES (3, 4);

INSERT INTO CUSCA (ID\_CUSCA, NUMAR\_LOCURI)

VALUES (4, 6);

INSERT INTO CUSCA (ID\_CUSCA, NUMAR\_LOCURI)

VALUES (5, 5);

-- Tabela SECTOR --

INSERT INTO SECTOR (ID\_SECTOR, TIP\_NEVOIE, NUMAR\_LOCURI)

VALUES (1, 'Deparazitare interna si externa', 100);

INSERT INTO SECTOR (ID\_SECTOR, TIP\_NEVOIE, NUMAR\_LOCURI)

VALUES (2, 'Analize de sange', 30);

INSERT INTO SECTOR (ID\_SECTOR, TIP\_NEVOIE, NUMAR\_LOCURI)

VALUES (3, 'Vaccinari', 10);

INSERT INTO SECTOR (ID\_SECTOR, TIP\_NEVOIE, NUMAR\_LOCURI)

VALUES (4, 'Chirurgie generala', 20);

INSERT INTO SECTOR (ID\_SECTOR, TIP\_NEVOIE, NUMAR\_LOCURI)

VALUES (5, 'Dermatologie', 40);

-- Tabela SECTIE --

INSERT INTO SECTIE (ID\_SECTIE, NUME, ID\_CUSCA)

VALUES (1, 'Sectia 1', 1);

INSERT INTO SECTIE (ID\_SECTIE, NUME, ID\_CUSCA)

VALUES (2, 'Sectia 2', 2);

INSERT INTO SECTIE (ID\_SECTIE, NUME, ID\_CUSCA)

VALUES (3, 'Sectia 3', 3);

INSERT INTO SECTIE (ID\_SECTIE, NUME, ID\_CUSCA)

VALUES (4, 'Sectia 4', 4);

INSERT INTO SECTIE (ID\_SECTIE, NUME, ID\_CUSCA)

VALUES (5, 'Sectia 5', 5);

-- Tabela SECTIE\_SECTOR --

INSERT INTO SECTIE\_SECTOR (ID\_SECTIE, ID\_SECTOR)

VALUES (1, 1);

INSERT INTO SECTIE\_SECTOR (ID\_SECTIE, ID\_SECTOR)

VALUES (1, 2);

INSERT INTO SECTIE\_SECTOR (ID\_SECTIE, ID\_SECTOR)

VALUES (1, 3);

INSERT INTO SECTIE\_SECTOR (ID\_SECTIE, ID\_SECTOR)

VALUES (2, 2);

INSERT INTO SECTIE\_SECTOR (ID\_SECTIE, ID\_SECTOR)

VALUES (2, 4);

INSERT INTO SECTIE\_SECTOR (ID\_SECTIE, ID\_SECTOR)

VALUES (3, 3);

INSERT INTO SECTIE\_SECTOR (ID\_SECTIE, ID\_SECTOR)

VALUES (3, 5);

INSERT INTO SECTIE\_SECTOR (ID\_SECTIE, ID\_SECTOR)

VALUES (4, 4);

INSERT INTO SECTIE\_SECTOR (ID\_SECTIE, ID\_SECTOR)

VALUES (4, 1);

INSERT INTO SECTIE\_SECTOR (ID\_SECTIE, ID\_SECTOR)

VALUES (5, 5);

-- Tabela MEDICAMENT --

INSERT INTO MEDICAMENT (NUMAR, DENUMIRE\_PRODUS, CANTITATE)

VALUES (1, 'Wyzium', 50);

INSERT INTO MEDICAMENT (NUMAR, DENUMIRE\_PRODUS, CANTITATE)

VALUES (2, 'KalmVet', 50);

INSERT INTO MEDICAMENT (NUMAR, DENUMIRE\_PRODUS, CANTITATE)

VALUES (3, 'RX Nutricalm', 50);

INSERT INTO MEDICAMENT (NUMAR, DENUMIRE\_PRODUS, CANTITATE)

VALUES (4, 'Entero-Chronic', 50);

INSERT INTO MEDICAMENT (NUMAR, DENUMIRE\_PRODUS, CANTITATE)

VALUES (5, 'Calmex', 50);

-- Tabela FARMACIE --

INSERT INTO FARMACIE (ID\_FARMACIE, NUME, TELEFON, PROGRAM, NUMAR)

VALUES (1, 'ANIMAX VET SRL', '33522396', '9 AM - 6 PM', 1);

--INSERT INTO FARMACIE (ID\_FARMACIE, NUME, TELEFON, PROGRAM, NUMAR)

--VALUES (1, 'ANIMAX VET SRL', '33522396', '9 AM - 6 PM', 2);

INSERT INTO FARMACIE (ID\_FARMACIE, NUME, TELEFON, PROGRAM, NUMAR)

VALUES (2, 'PET BOUTIQUE', '38440980', '10 AM - 8 PM', 1);

INSERT INTO FARMACIE (ID\_FARMACIE, NUME, TELEFON, PROGRAM, NUMAR)

VALUES (3, 'ANIMAL AREA', '39898404', '12 AM - 6 PM', 1);

INSERT INTO FARMACIE (ID\_FARMACIE, NUME, TELEFON, PROGRAM, NUMAR)

VALUES (4, ' GROOM VET C&A SRL', '41654738', '8 AM - 8 PM', 1);

INSERT INTO FARMACIE (ID\_FARMACIE, NUME, TELEFON, PROGRAM, NUMAR)

VALUES (5, 'PETMART ONLINE SRL', '32167601', '24/7', 1);

-- Tabela MAGAZIN --

INSERT INTO MAGAZIN (ID\_MAGAZIN, ID\_CONTRACT, NUME, TELEFON, PROGRAM)

VALUES (1, 1, 'Animax – Pet Store', '987-654-3456', '10 AM - 8 PM');

--INSERT INTO MAGAZIN (ID\_MAGAZIN, ID\_CONTRACT, NUME, TELEFON, PROGRAM)

--VALUES (1, 2, 'PetMart', '987-654-3210', '10 AM - 8 PM');

INSERT INTO MAGAZIN (ID\_MAGAZIN, ID\_CONTRACT, NUME, TELEFON, PROGRAM)

VALUES (2, 2, 'PetMart', '865-153-3210', '9 AM - 4 PM');

INSERT INTO MAGAZIN (ID\_MAGAZIN, ID\_CONTRACT, NUME, TELEFON, PROGRAM)

VALUES (3, 3, 'Maxi-Pet', '235-864-3210', '12 AM - 10 PM');

INSERT INTO MAGAZIN (ID\_MAGAZIN, ID\_CONTRACT, NUME, TELEFON, PROGRAM)

VALUES (4, 4, 'Shop4Pet', '987-654-2355', '10 AM - 8 PM');

INSERT INTO MAGAZIN (ID\_MAGAZIN, ID\_CONTRACT, NUME, TELEFON, PROGRAM)

VALUES (5, 5, 'PerfectPet', '987-654-3210', '10 AM - 8 PM');

-- Tabela HRANA --

INSERT INTO HRANA (NUMAR, DENUMIRE\_PRODUS, CANTITATE, ID\_MAGAZIN)

VALUES (1, 'Hrana uscata pentru caini Pedigree Adult', 100, 1);

INSERT INTO HRANA (NUMAR, DENUMIRE\_PRODUS, CANTITATE, ID\_MAGAZIN)

VALUES (2, 'Hrana uscata pentru caini Brit Premium, Adult', 120, 2);

INSERT INTO HRANA (NUMAR, DENUMIRE\_PRODUS, CANTITATE, ID\_MAGAZIN)

VALUES (3, 'Hrana uscata pentru caini Bosch Adult cu miel si orez', 80, 3);

INSERT INTO HRANA (NUMAR, DENUMIRE\_PRODUS, CANTITATE, ID\_MAGAZIN)

VALUES (4, 'Hrana umeda pentru caini Petkult Adult cu vanat', 100, 4);

INSERT INTO HRANA (NUMAR, DENUMIRE\_PRODUS, CANTITATE, ID\_MAGAZIN)

VALUES (5, 'Hrana umeda pentru caini Petkult Adult cu rata', 140, 5);

-- Tabela VIZITATOR --

INSERT INTO VIZITATOR (CNP, NUME, PRENUME, TELEFON, EMAIL)

VALUES ('1891123456789', 'Ion', 'Popescu', '0721123456', 'ion.popescu@example.com');

INSERT INTO VIZITATOR (CNP, NUME, PRENUME, TELEFON, EMAIL)

VALUES ('2900501234567', 'Elena', 'Ionescu', '0750123456', 'elena.ionescu@example.com');

INSERT INTO VIZITATOR (CNP, NUME, PRENUME, TELEFON, EMAIL)

VALUES ('1810412345678', 'Mihai', 'Georgescu', '0741123456', 'mihai.georgescu@example.com');

INSERT INTO VIZITATOR (CNP, NUME, PRENUME, TELEFON, EMAIL)

VALUES ('2930412345678', 'Maria', 'Stefanescu', '0771123456', 'maria.stefanescu@example.com');

INSERT INTO VIZITATOR (CNP, NUME, PRENUME, TELEFON, EMAIL)

VALUES ('1971001234567', 'Andrei', 'Popa', '0730123456', 'andrei.popa@example.com');

-- Tabela VETERINAR --

INSERT INTO VETERINAR (CNP, NUME, PRENUME)

VALUES ('1820701234567', 'Mihai', 'Popescu');

INSERT INTO VETERINAR (CNP, NUME, PRENUME)

VALUES ('1910501234567', 'Ana', 'Ionescu');

INSERT INTO VETERINAR (CNP, NUME, PRENUME)

VALUES ('1950412345678', 'Maria', 'Georgescu');

INSERT INTO VETERINAR (CNP, NUME, PRENUME)

VALUES ('1860301234567', 'Ion', 'Stefanescu');

INSERT INTO VETERINAR (CNP, NUME, PRENUME)

VALUES ('1830412345678', 'Elena', 'Popa');

-- Tabela ANIMAL --

INSERT INTO ANIMAL (ID\_ANIMAL, NUME, RASA, VARSTA, SEX, DATA\_SOSIRE, ID\_CUSCA, ID\_VETERINAR)

VALUES (1, 'Rex', 'Cocker Spaniel', 3, 'Masculin', TO\_DATE('2022-01-15', 'YYYY-MM-DD'), 1, '1820701234567');

INSERT INTO ANIMAL (ID\_ANIMAL, NUME, RASA, VARSTA, SEX, DATA\_SOSIRE, ID\_CUSCA, ID\_VETERINAR)

VALUES (2, 'Lola', 'Beagle', 2, 'Feminin', TO\_DATE('2023-03-22', 'YYYY-MM-DD'), 1, '1910501234567');

INSERT INTO ANIMAL (ID\_ANIMAL, NUME, RASA, VARSTA, SEX, DATA\_SOSIRE, ID\_CUSCA, ID\_VETERINAR)

VALUES (3, 'Max', 'Labrador Retriever', 1, 'Masculin', TO\_DATE('2023-05-10', 'YYYY-MM-DD'), 2, '1950412345678');

INSERT INTO ANIMAL (ID\_ANIMAL, NUME, RASA, VARSTA, SEX, DATA\_SOSIRE, ID\_CUSCA, ID\_VETERINAR)

VALUES (4, 'Daisy', 'Golden Retriever', 2, 'Feminin', TO\_DATE('2023-02-18', 'YYYY-MM-DD'), 2, '1860301234567');

INSERT INTO ANIMAL (ID\_ANIMAL, NUME, RASA, VARSTA, SEX, DATA\_SOSIRE, ID\_CUSCA, ID\_VETERINAR)

VALUES (5, 'Rocky', 'German Shepherd', 4, 'Masculin', TO\_DATE('2022-11-05', 'YYYY-MM-DD'), 3, '1830412345678');

-- Tabela CARTE\_DE\_SANATATE --

INSERT INTO CARTE\_DE\_SANATATE (ID\_ANIMAL, VACCINURI, DATA\_NASTERE, ANTECEDENTE\_MEDICALE, ID\_VETERINAR)

VALUES (1, 'Vaccin Leptospiroza, Vaccin Parvoviroza', TO\_DATE('2019-05-12', 'YYYY-MM-DD'), 'Niciuna', '1820701234567');

INSERT INTO CARTE\_DE\_SANATATE (ID\_ANIMAL, VACCINURI, DATA\_NASTERE, ANTECEDENTE\_MEDICALE, ID\_VETERINAR)

VALUES (2, 'Vaccin Leucemie Felina, Vaccin Rabie', TO\_DATE('2023-03-22', 'YYYY-MM-DD'), 'Vaccin alergie', '1910501234567');

INSERT INTO CARTE\_DE\_SANATATE (ID\_ANIMAL, VACCINURI, DATA\_NASTERE, ANTECEDENTE\_MEDICALE, ID\_VETERINAR)

VALUES (3, 'Vaccin Parvoviroza, Vaccin Distemper', TO\_DATE('2023-05-10', 'YYYY-MM-DD'), 'Niciuna', '1950412345678');

INSERT INTO CARTE\_DE\_SANATATE (ID\_ANIMAL, VACCINURI, DATA\_NASTERE, ANTECEDENTE\_MEDICALE, ID\_VETERINAR)

VALUES (4, 'Vaccin Rabie, Vaccin Parainfluenza', TO\_DATE('2023-02-18', 'YYYY-MM-DD'), 'Vaccin diabet', '1860301234567');

INSERT INTO CARTE\_DE\_SANATATE (ID\_ANIMAL, VACCINURI, DATA\_NASTERE, ANTECEDENTE\_MEDICALE, ID\_VETERINAR)

VALUES (5, 'Vaccin Bordeteloza, Vaccin Parvoviroza', TO\_DATE('2022-11-05', 'YYYY-MM-DD'), 'Niciuna', '1830412345678');

-- Tabela CONSULTATIE --

INSERT INTO CONSULTATIE (ID\_CONSULTATIE, ID\_ANIMAL, DATA, ID\_VETERINAR)

VALUES (2, 3, TO\_DATE('2023-08-01', 'YYYY-MM-DD'), '1950412345678');

INSERT INTO CONSULTATIE (ID\_CONSULTATIE, ID\_ANIMAL, DATA, ID\_VETERINAR)

VALUES (3, 2, TO\_DATE('2023-08-02', 'YYYY-MM-DD'), '1910501234567');

INSERT INTO CONSULTATIE (ID\_CONSULTATIE, ID\_ANIMAL, DATA, ID\_VETERINAR)

VALUES (4, 5, TO\_DATE('2023-08-03', 'YYYY-MM-DD'), '1830412345678');

INSERT INTO CONSULTATIE (ID\_CONSULTATIE, ID\_ANIMAL, DATA, ID\_VETERINAR)

VALUES (5, 4, TO\_DATE('2023-08-04', 'YYYY-MM-DD'), '1860301234567');

-- Tabela CERERE\_ADOPTIE --

INSERT INTO CERERE\_ADOPTIE (SERIE, ID\_ANIMAL, ID\_VIZITATOR, DATA)

VALUES ('DEF456', 3, '1971001234567', TO\_DATE('2023-08-02', 'YYYY-MM-DD'));

INSERT INTO CERERE\_ADOPTIE (SERIE, ID\_ANIMAL, ID\_VIZITATOR, DATA)

VALUES ('GHI789', 2, '1810412345678', TO\_DATE('2023-08-03', 'YYYY-MM-DD'));

INSERT INTO CERERE\_ADOPTIE (SERIE, ID\_ANIMAL, ID\_VIZITATOR, DATA)

VALUES ('JKL012', 5, '2900501234567', TO\_DATE('2023-08-04', 'YYYY-MM-DD'));

INSERT INTO CERERE\_ADOPTIE (SERIE, ID\_ANIMAL, ID\_VIZITATOR, DATA)

VALUES ('MNO345', 4, '2930412345678', TO\_DATE('2023-08-05', 'YYYY-MM-DD'));

-- Tabela CLINICA\_VETERINARA --

INSERT INTO CLINICA\_VETERINARA (ID\_CLINICA, DENUMIRE, ADRESA, TELEFON, EMAIL, ID\_VETERINAR)

VALUES (1, 'Clinica Veterinara "VetCare"', 'Strada Vesela, nr. 789', '555-111-2222', 'vetcare@example.com', '1830412345678');

INSERT INTO CLINICA\_VETERINARA (ID\_CLINICA, DENUMIRE, ADRESA, TELEFON, EMAIL, ID\_VETERINAR)

VALUES (2, 'Clinica Veterinara "AnimalCare"', 'Strada Centrala, nr. 456', '555-987-6543', 'clinicavet@example.com', '1950412345678');

INSERT INTO CLINICA\_VETERINARA (ID\_CLINICA, DENUMIRE, ADRESA, TELEFON, EMAIL, ID\_VETERINAR)

VALUES (3, 'Clinica Veterinara "VetExpert"', 'Strada Mica, nr. 789', '555-456-7890', 'vetexpert@example.com', '1820701234567');

INSERT INTO CLINICA\_VETERINARA (ID\_CLINICA, DENUMIRE, ADRESA, TELEFON, EMAIL, ID\_VETERINAR)

VALUES (4, 'Clinica Veterinara "PetHealth"', 'Strada Mare, nr. 101', '555-789-0123', 'pethealth@example.com', '1910501234567');

INSERT INTO CLINICA\_VETERINARA (ID\_CLINICA, DENUMIRE, ADRESA, TELEFON, EMAIL, ID\_VETERINAR)

VALUES (5, 'Clinica Veterinara "HappyPets"', 'Strada Munte, nr. 543', '555-012-3456', 'happypets@example.com', '1860301234567');

-- Tabela CONTRACT --

INSERT INTO CONTRACT (NUMAR, DATA\_EMITERE, DURATA, TIP, id\_magazin, id\_farmacie, id\_clinica)

VALUES (101, TO\_DATE('2023-07-15', 'YYYY-MM-DD'), 2, 'Servicii Veterinare', 1, NULL, 2);

INSERT INTO CONTRACT (NUMAR, DATA\_EMITERE, DURATA, TIP, id\_magazin, id\_farmacie, id\_clinica)

VALUES (102, TO\_DATE('2023-07-16', 'YYYY-MM-DD'), 2, 'Consultatii Medicale', 2, 1, 3);

INSERT INTO CONTRACT (NUMAR, DATA\_EMITERE, DURATA, TIP, id\_magazin, id\_farmacie, id\_clinica)

VALUES (103, TO\_DATE('2023-07-17', 'YYYY-MM-DD'), 2, 'Analize Medicale', 3, 3, 1);

INSERT INTO CONTRACT (NUMAR, DATA\_EMITERE, DURATA, TIP, id\_magazin, id\_farmacie, id\_clinica)

VALUES (104, TO\_DATE('2023-07-18', 'YYYY-MM-DD'), 2, 'Chirurgie Veterinara', 4, 2, 5);

INSERT INTO CONTRACT (NUMAR, DATA\_EMITERE, DURATA, TIP, id\_magazin, id\_farmacie, id\_clinica)

VALUES (105, TO\_DATE('2023-07-19', 'YYYY-MM-DD'), 2, 'Vaccinari Animale', 5, 5, 4)

-- Tabela ANGAJAT --

INSERT INTO ANGAJAT (ID\_ANGAJAT, NUME, PRENUME, CNP, ADRESA, TELEFON, EMAIL, ID\_CONTRACT)

VALUES (1, 'Ana', 'Popa', '5678901234567', 'Strada Angajatului, nr. 6', '555-555-6666', 'ana@example.com', 101);

INSERT INTO ANGAJAT (ID\_ANGAJAT, NUME, PRENUME, CNP, ADRESA, TELEFON, EMAIL, ID\_CONTRACT)

VALUES (2, 'Maria', 'Popescu', '1234567890123', 'Strada Angajatului, nr. 2', '555-111-2222', 'maria@example.com', 102);

INSERT INTO ANGAJAT (ID\_ANGAJAT, NUME, PRENUME, CNP, ADRESA, TELEFON, EMAIL, ID\_CONTRACT)

VALUES (3, 'Ion', 'Ionescu', '2345678901234', 'Strada Angajatului, nr. 3', '555-222-3333', 'ion@example.com', 103);

INSERT INTO ANGAJAT (ID\_ANGAJAT, NUME, PRENUME, CNP, ADRESA, TELEFON, EMAIL, ID\_CONTRACT)

VALUES (4, 'Elena', 'Georgescu', '3456789012345', 'Strada Angajatului, nr. 4', '555-333-4444', 'elena@example.com', 104);

INSERT INTO ANGAJAT (ID\_ANGAJAT, NUME, PRENUME, CNP, ADRESA, TELEFON, EMAIL, ID\_CONTRACT)

VALUES (5, 'Mihai', 'Stef?nescu', '4567890123456', 'Strada Angajatului, nr. 5', '555-444-5555', 'mihai@example.com', 105);

-- Sterg constragerile pentru tabela director, deoarece exista un singur director

--ALTER TABLE DIRECTOR

--DROP CONSTRAINT SYS\_C00129012018;

-- Tabela DIRECTOR --

INSERT INTO DIRECTOR (ID\_DIRECTOR, ID\_CONTRACT, ID\_ANGAJAT)

VALUES (1, 101, 1);

INSERT INTO DIRECTOR (ID\_DIRECTOR, ID\_CONTRACT, ID\_ANGAJAT)

VALUES (1, 102, 2);

INSERT INTO DIRECTOR (ID\_DIRECTOR, ID\_CONTRACT, ID\_ANGAJAT)

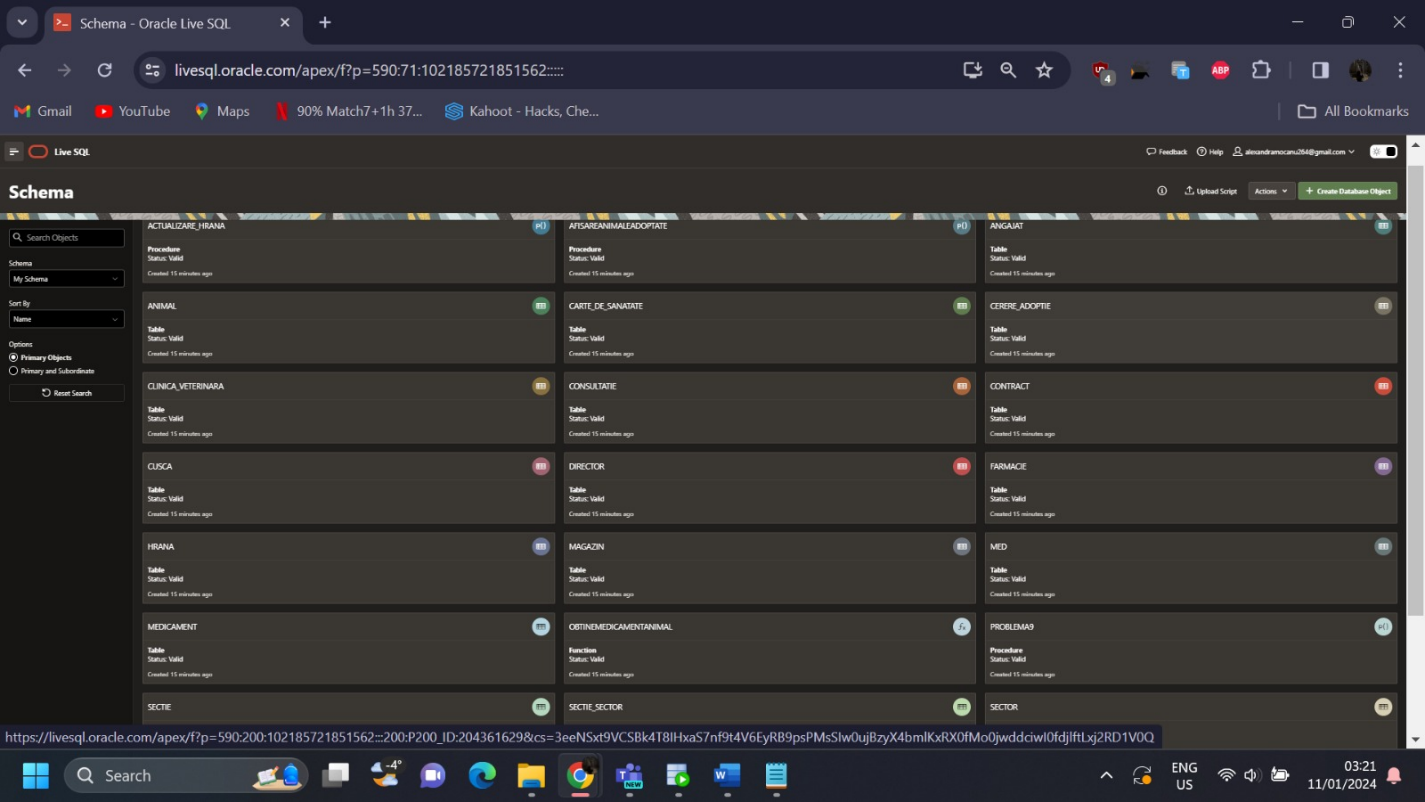
VALUES (1, 103, 3);

INSERT INTO DIRECTOR (ID\_DIRECTOR, ID\_CONTRACT, ID\_ANGAJAT)

VALUES (1, 104, 4);

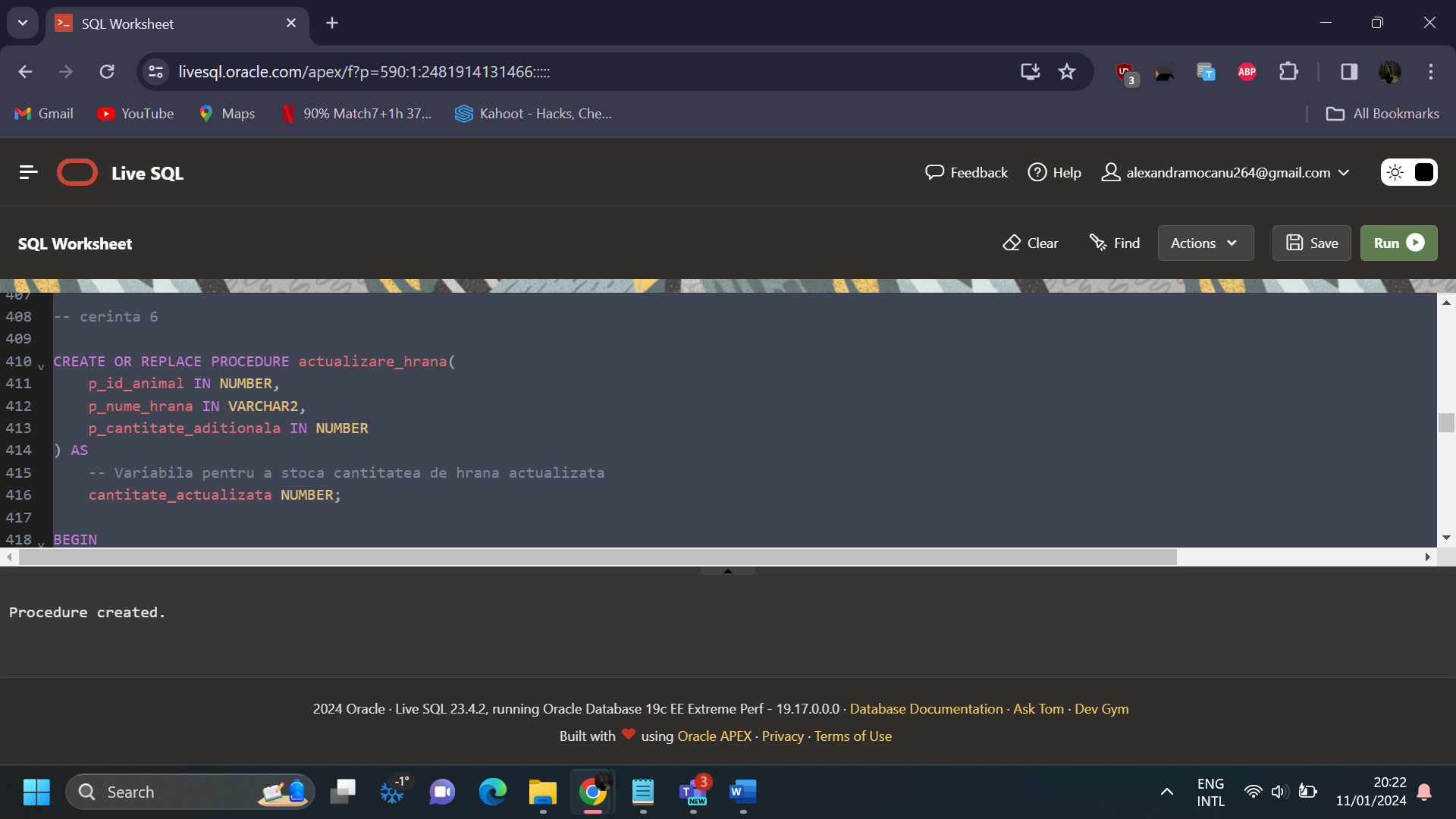
INSERT INTO DIRECTOR (ID\_DIRECTOR, ID\_CONTRACT, ID\_ANGAJAT)

VALUES (1, 105, 5);



1. Formularea in limbaj natural o problema rezolvata folosind un subprogram stocat independent.folosind 3 tipuri de colectii.

Procedura actualizare\_hrana primește trei parametri: ID-ul animalului, numele hranei și cantitatea adițională de hrana. Apoi, procedura utilizează o colecție de tip INDEX BY pentru a stoca ID-urile animalelor și hrana necesară. Se parcurge colecția și se actualizează cantitatea de hrana pentru fiecare animal în parte.

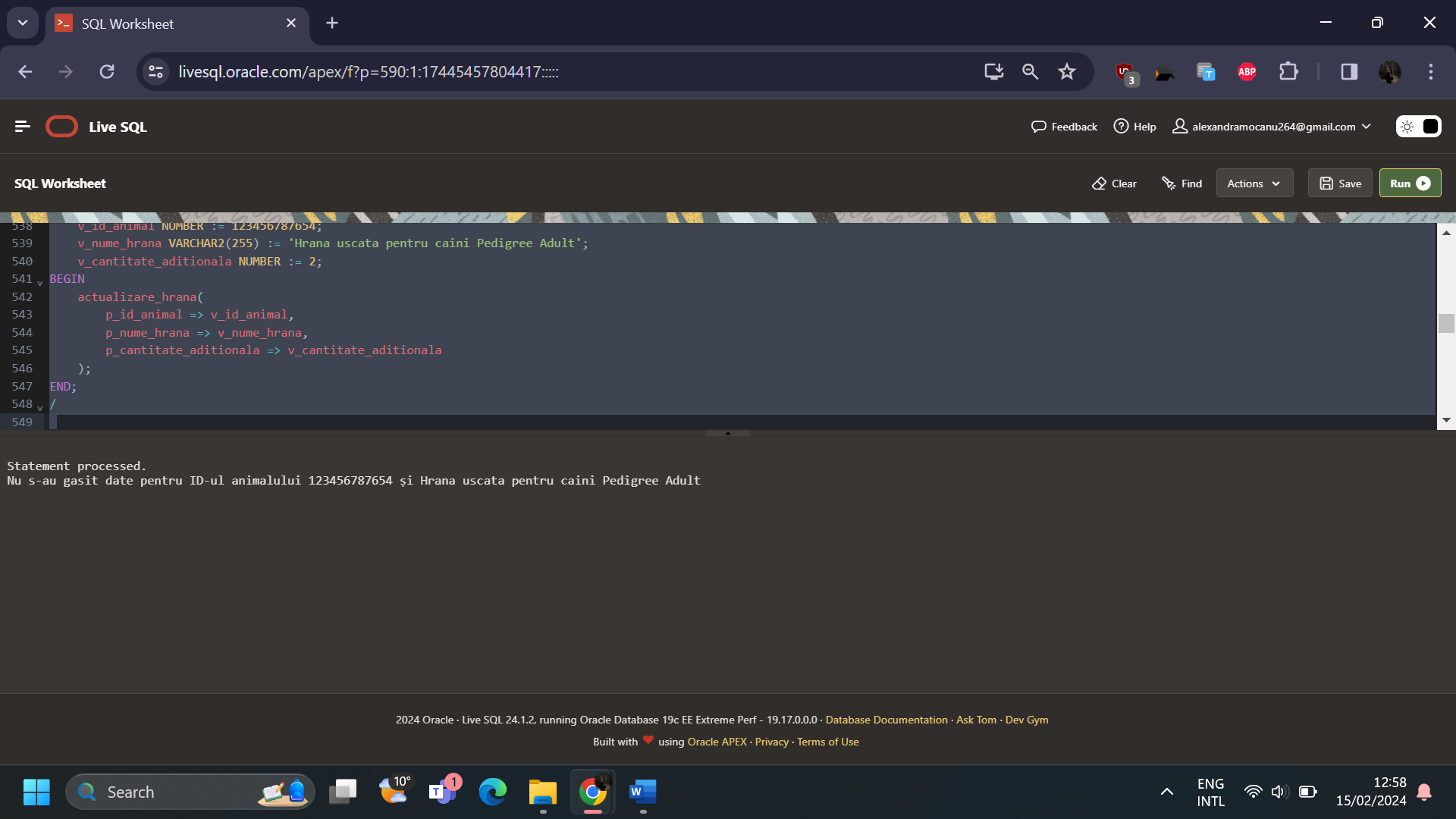


Apelarea procedurii:

* Pentru un ID valid:



* Pentru un ID invalid:



CREATE OR REPLACE PROCEDURE actualizare\_hrana(

p\_id\_animal IN NUMBER,

p\_nume\_hrana IN VARCHAR2,

p\_cantitate\_aditionala IN NUMBER

) AS

-- Variabile pentru a stoca cantitatea de hrana actualizata

cantitate\_actualizata NUMBER

-- Tabel indexat

TYPE Hrane\_Indexate IS TABLE OF VARCHAR2(255) INDEX BY PLS\_INTEGER;

hrane\_disponibile Hrane\_Indexate;

-- Tabel imbricat

TYPE Hrane\_Imbricate IS TABLE OF VARCHAR2(255);

hrane\_animal Hrane\_Imbricate := Hrane\_Imbricate();

BEGIN

-- Adăugarea hranei disponibile în tabelul indexat

hrane\_disponibile(1) := 'Hrana uscata pentru caini Pedigree Adult';

hrane\_disponibile(2) := 'Hrana umeda pentru pisici Whiskas';

-- Actualizarea cantitatii de hrana

UPDATE HRANA

SET CANTITATE = CANTITATE + p\_cantitate\_aditionala

WHERE ID\_MAGAZIN = p\_id\_animal AND DENUMIRE\_PRODUS = p\_nume\_hrana;

cantitate\_actualizata := SQL%ROWCOUNT;

IF cantitate\_actualizata > 0 THEN

COMMIT;

DBMS\_OUTPUT.PUT\_LINE('Cantitatea de hrana pentru animalul cu ID ' || p\_id\_animal || ' și ' || p\_nume\_hrana || ' a fost actualizata cu ' || p\_cantitate\_aditionala || ' unitați.');

ELSE

DBMS\_OUTPUT.PUT\_LINE('Nu s-au gasit date pentru ID-ul animalului ' || p\_id\_animal || ' și ' || p\_nume\_hrana);

END IF;

EXCEPTION

WHEN OTHERS THEN

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

END;

/

DECLARE

v\_id\_animal NUMBER := 1;

v\_nume\_hrana VARCHAR2(255) := 'Hrana uscata pentru caini Pedigree Adult';

v\_cantitate\_aditionala NUMBER := 2;

BEGIN

actualizare\_hrana(

p\_id\_animal => v\_id\_animal,

p\_nume\_hrana => v\_nume\_hrana,

p\_cantitate\_aditionala => v\_cantitate\_aditionala

);

END;

/

DECLARE

v\_id\_animal NUMBER := 123456787654;

v\_nume\_hrana VARCHAR2(255) := 'Hrana uscata pentru caini Pedigree Adult';

v\_cantitate\_aditionala NUMBER := 2;

BEGIN

actualizare\_hrana(

p\_id\_animal => v\_id\_animal,

p\_nume\_hrana => v\_nume\_hrana,

p\_cantitate\_aditionala => v\_cantitate\_aditionala

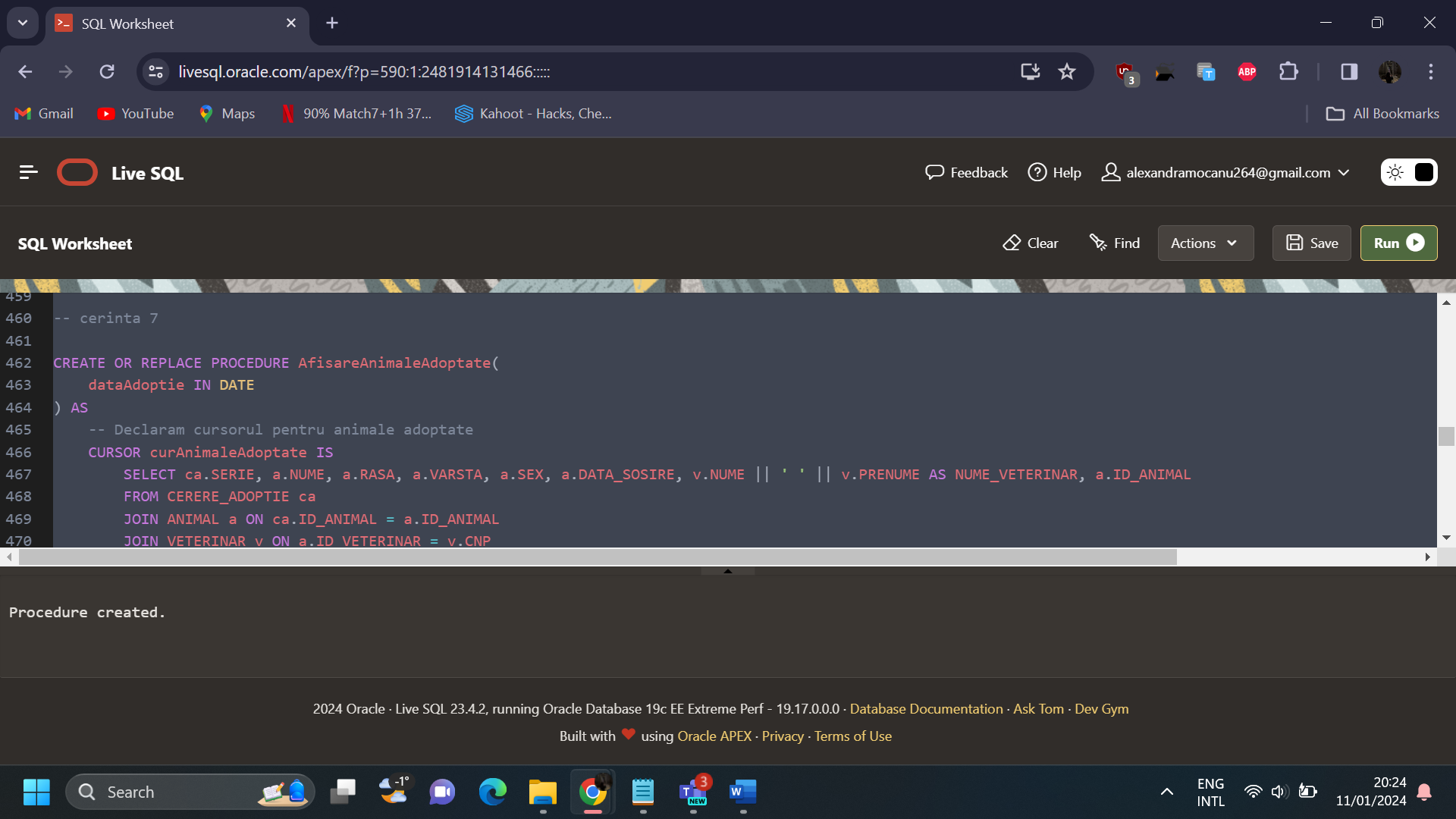
);

END;

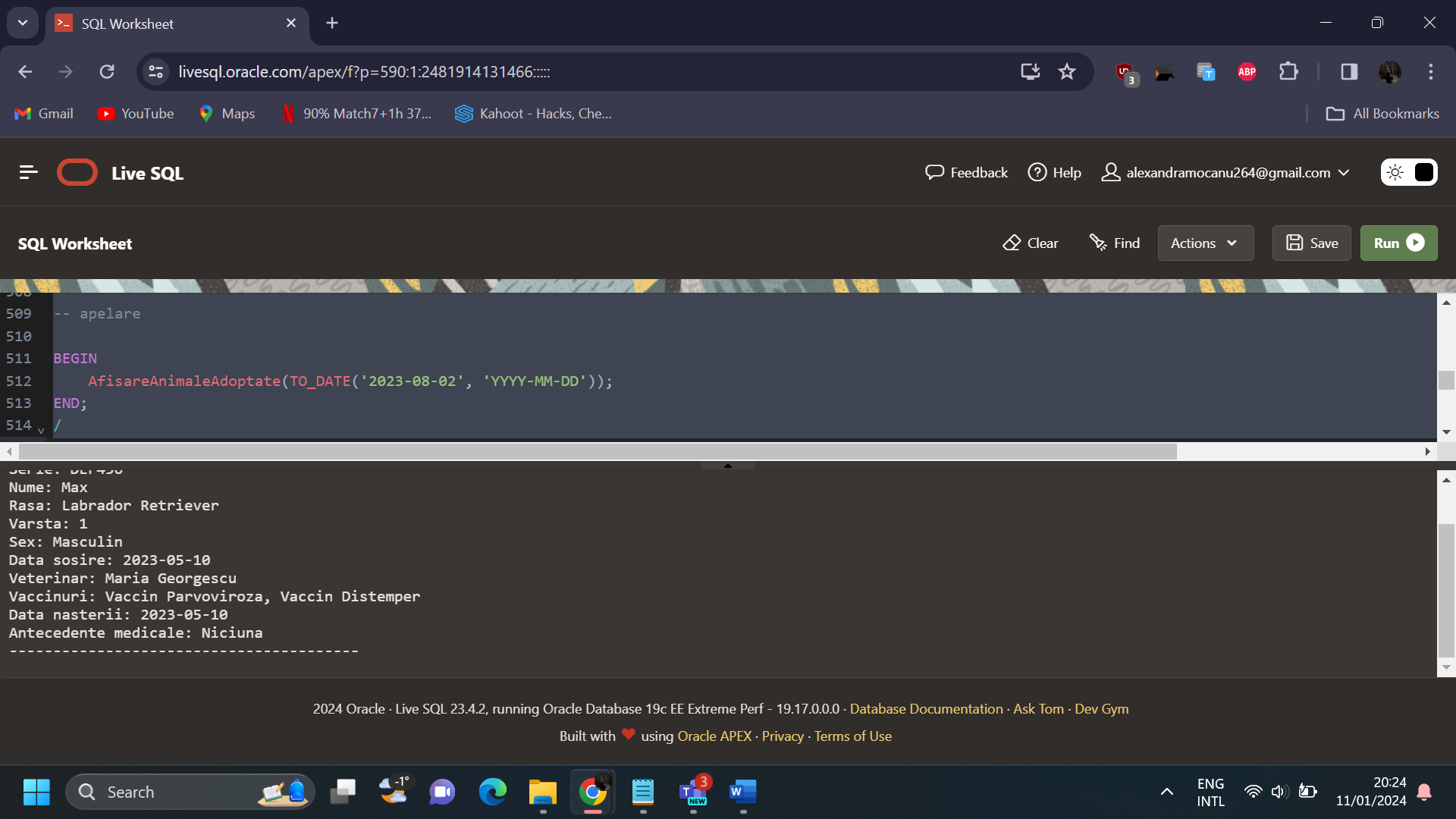
/

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.

Subprogramul stocat independent numit "AfisareAnimaleAdoptate" folosește două tipuri diferite de cursoare: unul pentru animalele adoptate și unul cursor parametrizat dependent de primul cursor. Apoi, am apelat subprogramul furnizând o dată specifică.



Apelare:



CREATE OR REPLACE PROCEDURE afisare\_animale\_adoptate(

dataAdoptie IN DATE

) AS

-- Declaram cursorul pentru animale adoptate

CURSOR curAnimaleAdoptate IS

SELECT ca.SERIE, a.NUME, a.RASA, a.VARSTA, a.SEX, a.DATA\_SOSIRE, v.NUME || ' ' || v.PRENUME AS NUME\_VETERINAR, a.ID\_ANIMAL

FROM CERERE\_ADOPTIE ca

JOIN ANIMAL a ON ca.ID\_ANIMAL = a.ID\_ANIMAL

JOIN VETERINAR v ON a.ID\_VETERINAR = v.CNP

WHERE ca.DATA = dataAdoptie;

-- Declaram variabila pentru ID\_ANIMAL

v\_ID\_ANIMAL INT;

v\_au\_fost\_adoptate BOOLEAN := FALSE;

BEGIN

-- Parcurgem animalele adoptate

FOR animalAdoptat IN curAnimaleAdoptate LOOP

-- Salvam ID\_ANIMAL in variabila

v\_ID\_ANIMAL := animalAdoptat.ID\_ANIMAL;

v\_au\_fost\_adoptate := TRUE;

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

DBMS\_OUTPUT.PUT\_LINE('Serie: ' || animalAdoptat.SERIE);

DBMS\_OUTPUT.PUT\_LINE('Nume: ' || animalAdoptat.NUME);

DBMS\_OUTPUT.PUT\_LINE('Rasa: ' || animalAdoptat.RASA);

DBMS\_OUTPUT.PUT\_LINE('Varsta: ' || animalAdoptat.VARSTA);

DBMS\_OUTPUT.PUT\_LINE('Sex: ' || animalAdoptat.SEX);

DBMS\_OUTPUT.PUT\_LINE('Data sosire: ' || TO\_CHAR(animalAdoptat.DATA\_SOSIRE, 'YYYY-MM-DD'));

DBMS\_OUTPUT.PUT\_LINE('Veterinar: ' || animalAdoptat.NUME\_VETERINAR);

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

-- Obtinem informatii suplimentare folosind cursorul parametrizat

FOR infoSuplimentare IN (

SELECT cs.VACCINURI, cs.DATA\_NASTERE, cs.ANTECEDENTE\_MEDICALE

FROM CARTE\_DE\_SANATATE cs

WHERE cs.ID\_ANIMAL = v\_ID\_ANIMAL

) LOOP

DBMS\_OUTPUT.PUT\_LINE('Vaccinuri: ' || infoSuplimentare.VACCINURI);

DBMS\_OUTPUT.PUT\_LINE('Data nasterii: ' || TO\_CHAR(infoSuplimentare.DATA\_NASTERE, 'YYYY-MM-DD'));

DBMS\_OUTPUT.PUT\_LINE('Antecedente medicale: ' || infoSuplimentare.ANTECEDENTE\_MEDICALE);

END LOOP;

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

END LOOP

IF NOT v\_au\_fost\_adoptate THEN

DBMS\_OUTPUT.PUT\_LINE('Nu au fost adoptate animale in data introdusa.');

END IF;

END afisare\_animale\_adoptate;

/

-- apelare

BEGIN

afisare\_animale\_adoptate(TO\_DATE('2023-08-02', 'YYYY-MM-DD'));

END;

/

BEGIN

afisare\_animale\_adoptate(TO\_DATE('2023-08-03', 'YYYY-MM-DD'));

END;

/

BEGIN

afisare\_animale\_adoptate(TO\_DATE('2025-01-20', 'YYYY-MM-DD'));

END;

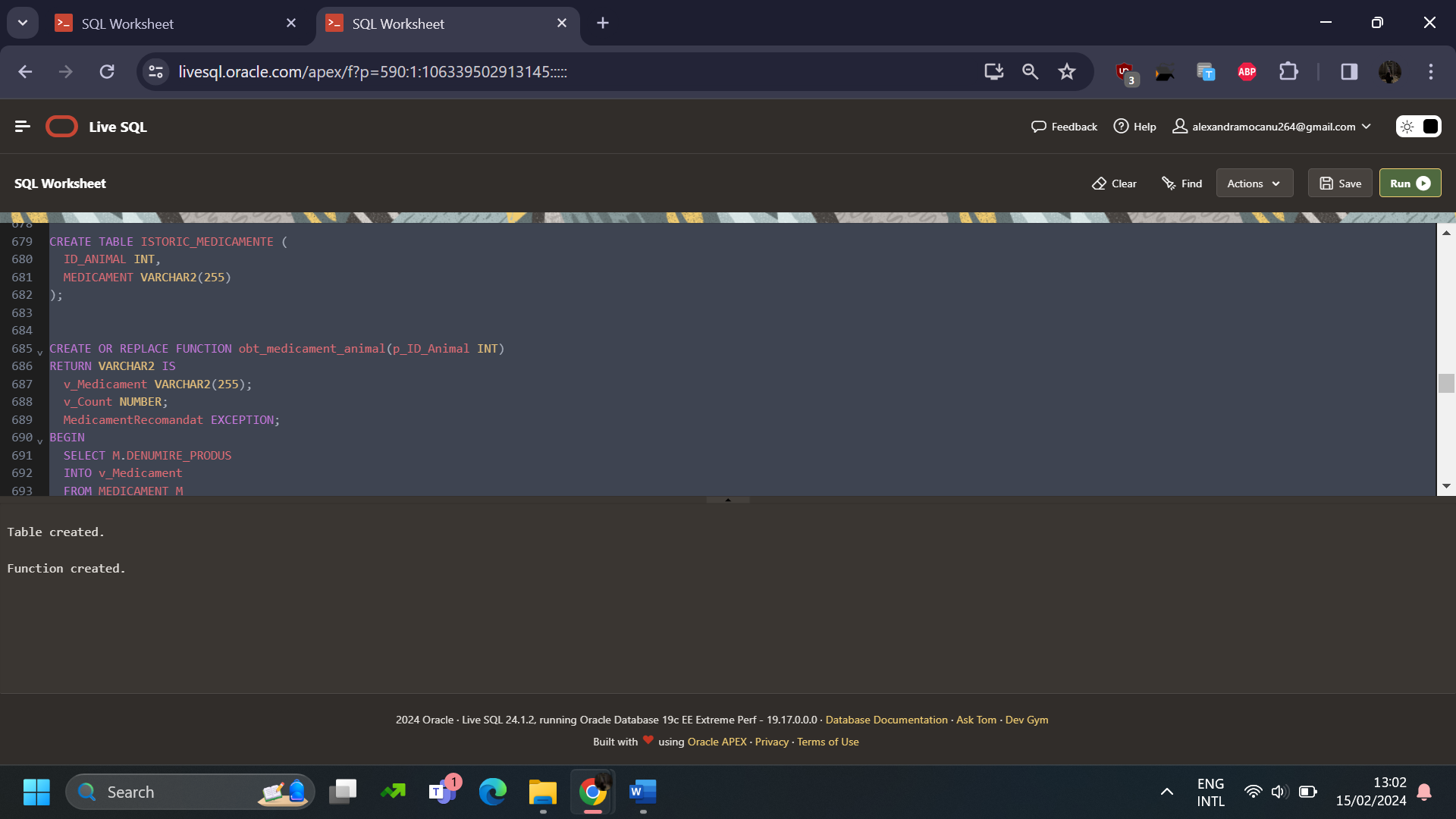
/

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.

Două erori proprii:

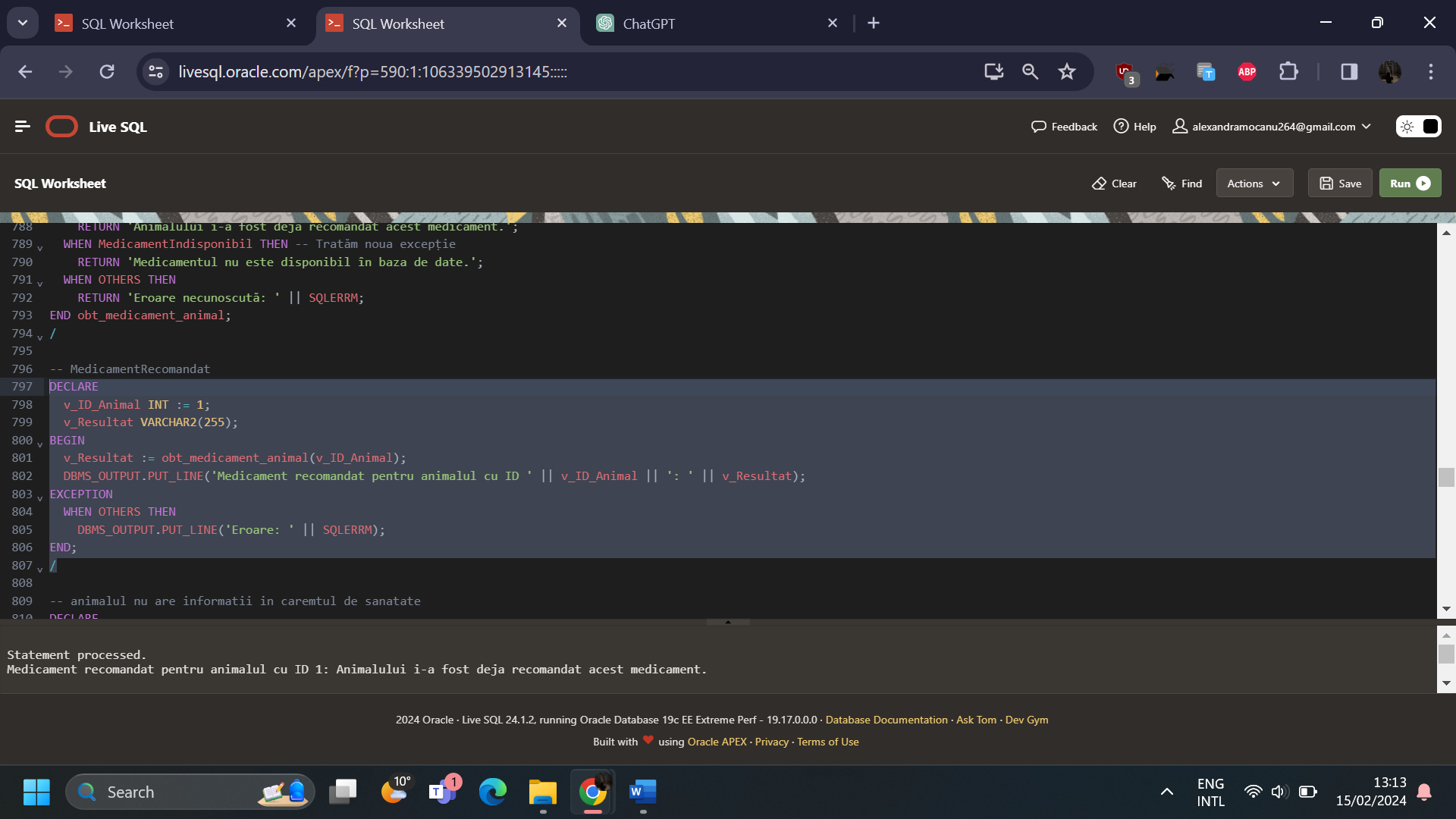
MedicamentRecomandat: Această excepție este ridicată atunci când animalului i-a fost deja recomandat același medicament.

MedicamentIndisponibil: Această excepție este ridicată atunci când medicamentul specificat nu este disponibil în baza de date. Mesajul asociat este "Medicamentul nu este disponibil în baza de date."

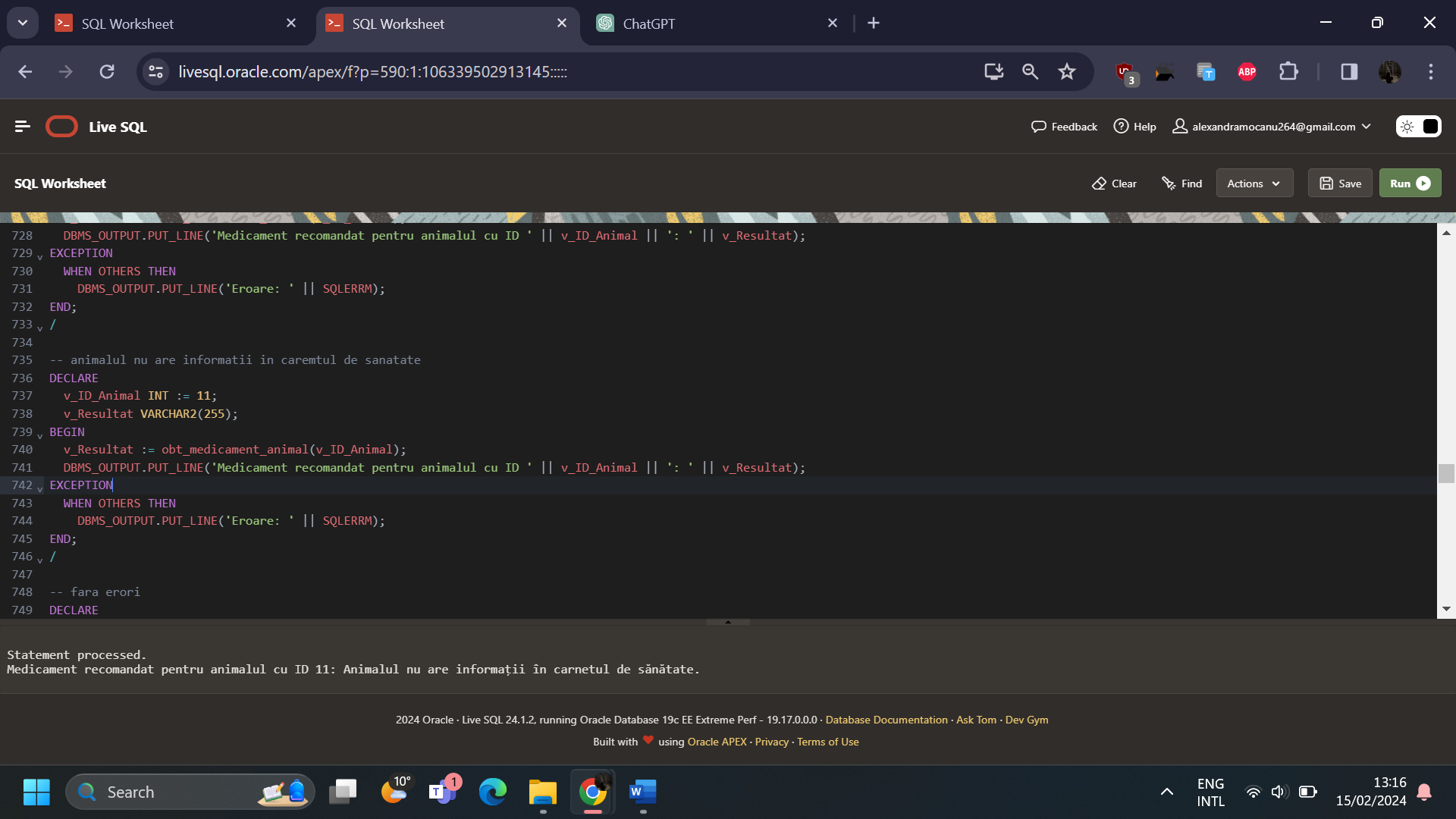


Apelare:

* Medicament deja recomandat:



* Nu exista informatii in carnet:



CREATE OR REPLACE FUNCTION obt\_medicament\_animal(p\_ID\_Animal INT)

RETURN VARCHAR2 IS

v\_Medicament VARCHAR2(255);

v\_Count NUMBER;

MedicamentRecomandat EXCEPTION;

MedicamentIndisponibil EXCEPTION;

BEGIN

SELECT M.DENUMIRE\_PRODUS

INTO v\_Medicament

FROM MEDICAMENT M

JOIN CARTE\_DE\_SANATATE CS ON M.NUMAR = CS.ID\_ANIMAL

WHERE CS.ID\_ANIMAL = p\_ID\_Animal;

SELECT COUNT(\*)

INTO v\_Count

FROM ISTORIC\_MEDICAMENTE

WHERE ID\_ANIMAL = p\_ID\_Animal AND MEDICAMENT = v\_Medicament;

IF v\_Count > 0 THEN

RAISE MedicamentRecomandat;

END IF;

-- Verificam daca medicamentul este disponibil

IF v\_Medicament IS NULL THEN

RAISE MedicamentIndisponibil;

END IF;

INSERT INTO ISTORIC\_MEDICAMENTE (ID\_ANIMAL, MEDICAMENT) VALUES (p\_ID\_Animal, v\_Medicament);

RETURN v\_Medicament;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

RETURN 'Animalul nu are informații în carnetul de sănătate.';

WHEN MedicamentRecomandat THEN

RETURN 'Animalului i-a fost deja recomandat acest medicament.';

WHEN MedicamentIndisponibil THEN -- Tratăm noua excepție

RETURN 'Medicamentul nu este disponibil în baza de date.';

WHEN OTHERS THEN

RETURN 'Eroare necunoscută: ' || SQLERRM;

END obt\_medicament\_animal;

/

-- MedicamentRecomandat

DECLARE

v\_ID\_Animal INT := 1;

v\_Resultat VARCHAR2(255);

BEGIN

v\_Resultat := obt\_medicament\_animal(v\_ID\_Animal);

DBMS\_OUTPUT.PUT\_LINE('Medicament recomandat pentru animalul cu ID ' || v\_ID\_Animal || ': ' || v\_Resultat);

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Eroare: ' || SQLERRM);

END;

/

-- animalul nu are informatii in caremtul de sanatate

DECLARE

v\_ID\_Animal INT := 11;

v\_Resultat VARCHAR2(255);

BEGIN

v\_Resultat := obt\_medicament\_animal(v\_ID\_Animal);

DBMS\_OUTPUT.PUT\_LINE('Medicament recomandat pentru animalul cu ID ' || v\_ID\_Animal || ': ' || v\_Resultat);

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Eroare: ' || SQLERRM);

END;

/

-- fara erori

DECLARE

v\_ID\_Animal INT := 6;

v\_Resultat VARCHAR2(255);

BEGIN

v\_Resultat := obt\_medicament\_animal(v\_ID\_Animal);

DBMS\_OUTPUT.PUT\_LINE('Medicament recomandat pentru animalul cu ID ' || v\_ID\_Animal || ': ' || v\_Resultat);

EXCEPTION

WHEN OTHERS THEN

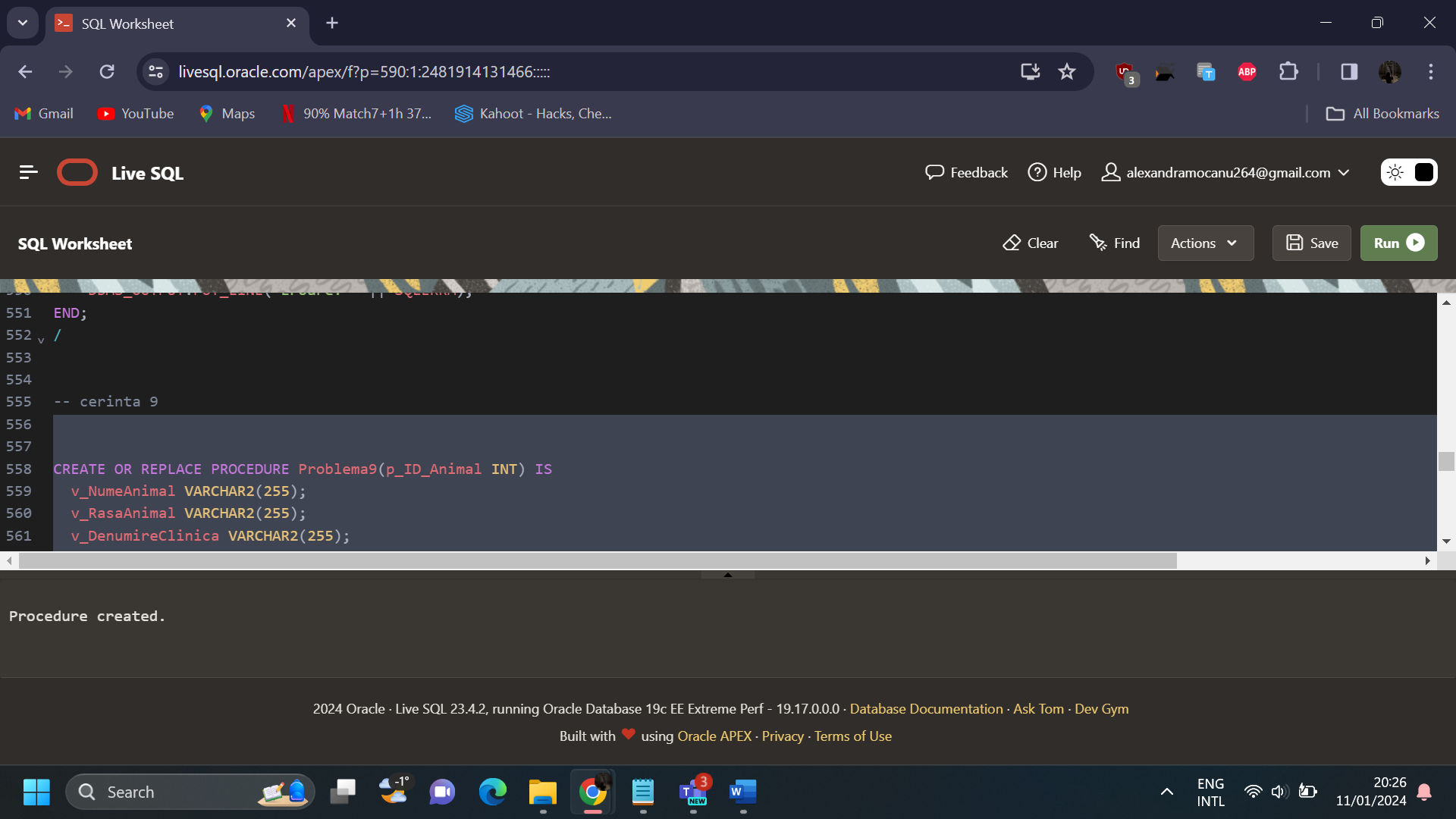
DBMS\_OUTPUT.PUT\_LINE('Eroare: ' || SQLERRM);

END;

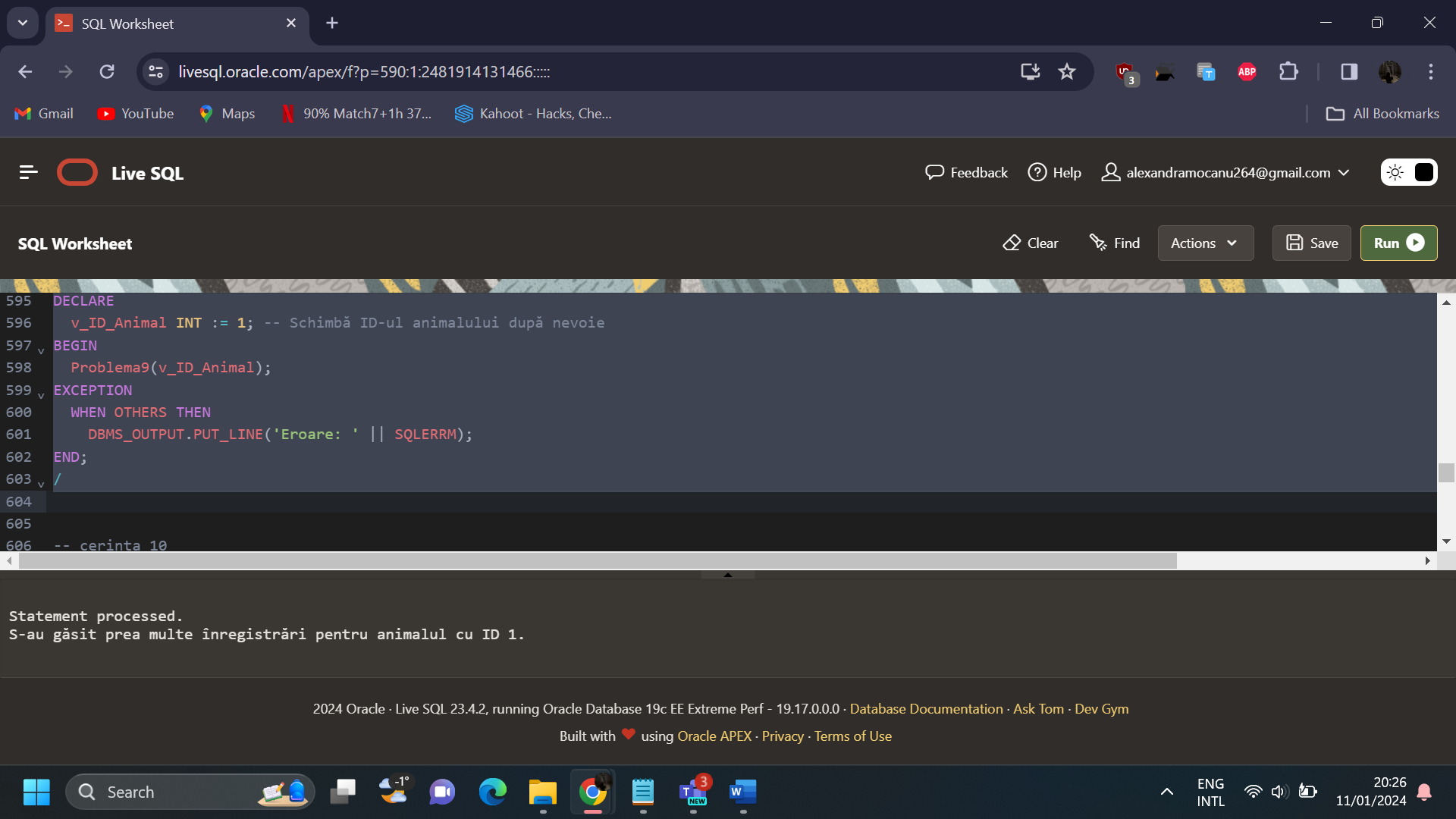
/

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

Procedura stocata independent utilizează o singură comandă SQL pentru a accesa și prelua date din sase tabele diferite: ANIMAL, CLINICA\_VETERINARA, SECTIE\_SECTOR, SECTOR, MED, FARMACIE. Procedura tratează explicit excepțiile NO\_DATA\_FOUND și TOO\_MANY\_ROWS. Apelul procedurii este gestionat într-un bloc DECLARE care gestionează excepțiile.



Testare:



CREATE OR REPLACE PROCEDURE informatii\_animal(p\_ID\_Animal INT) IS

v\_NumeAnimal VARCHAR2(255);

v\_RasaAnimal VARCHAR2(255);

v\_DenumireClinica VARCHAR2(255);

v\_TipNevoieSector VARCHAR2(255);

v\_NumeFarmacie VARCHAR2(255);

v\_RecordCount INT := 0;

BEGIN

SELECT A.NUME, A.RASA, C.DENUMIRE, S.TIP\_NEVOIE, F.NUME

INTO v\_NumeAnimal, v\_RasaAnimal, v\_DenumireClinica, v\_TipNevoieSector, v\_NumeFarmacie

FROM ANIMAL A

JOIN CLINICA\_VETERINARA C ON A.ID\_VETERINAR = C.ID\_VETERINAR

JOIN SECTIE\_SECTOR SS ON SS.ID\_SECTIE = A.ID\_CUSCA

JOIN SECTOR S ON S.ID\_SECTOR = SS.ID\_SECTOR

LEFT JOIN MED M ON M.ID\_FARMACIE = S.ID\_SECTOR

LEFT JOIN FARMACIE F ON F.ID\_FARMACIE = M.ID\_FARMACIE

WHERE A.ID\_ANIMAL = p\_ID\_Animal AND ROWNUM = 1

ORDER BY A.DATA\_SOSIRE DESC;

-- Numărul de înregistrări găsite

SELECT COUNT(\*)

INTO v\_RecordCount

FROM ANIMAL A

JOIN CLINICA\_VETERINARA C ON A.ID\_VETERINAR = C.ID\_VETERINAR

JOIN SECTIE\_SECTOR SS ON SS.ID\_SECTIE = A.ID\_CUSCA

JOIN SECTOR S ON S.ID\_SECTOR = SS.ID\_SECTOR

LEFT JOIN MED M ON M.ID\_FARMACIE = S.ID\_SECTOR

LEFT JOIN FARMACIE F ON F.ID\_FARMACIE = M.ID\_FARMACIE

WHERE A.ID\_ANIMAL = p\_ID\_Animal;

-- Poți adăuga aici operații suplimentare folosind variabilele create

-- Afisează doar prima înregistrare dacă sunt mai multe

IF v\_RecordCount > 1 THEN

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

DBMS\_OUTPUT.PUT\_LINE('S-au găsit prea multe înregistrări pentru animalul cu ID ' || p\_ID\_Animal || '. Se afișează doar prima înregistrare.');

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

END IF;

DBMS\_OUTPUT.PUT\_LINE('Numele animalului cu ID ' || p\_ID\_Animal || ': ' || v\_NumeAnimal);

DBMS\_OUTPUT.PUT\_LINE('Rasa animalului: ' || v\_RasaAnimal);

DBMS\_OUTPUT.PUT\_LINE('Clinica veterinară asociată: ' || v\_DenumireClinica);

DBMS\_OUTPUT.PUT\_LINE('Sector cu nevoie: ' || v\_TipNevoieSector);

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

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

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

DBMS\_OUTPUT.PUT\_LINE('Animalul cu ID ' || p\_ID\_Animal || ' nu a fost găsit.');

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

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Eroare necunoscută: ' || SQLERRM);

END informatii\_animal;

/

-- apelare

DECLARE

v\_ID\_Animal INT := 1;

BEGIN

informatii\_animal(v\_ID\_Animal);

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Eroare: ' || SQLERRM);

END;

/

DECLARE

v\_ID\_Animal INT := 9;

BEGIN

informatii\_animal(v\_ID\_Animal);

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Eroare: ' || SQLERRM);

END;

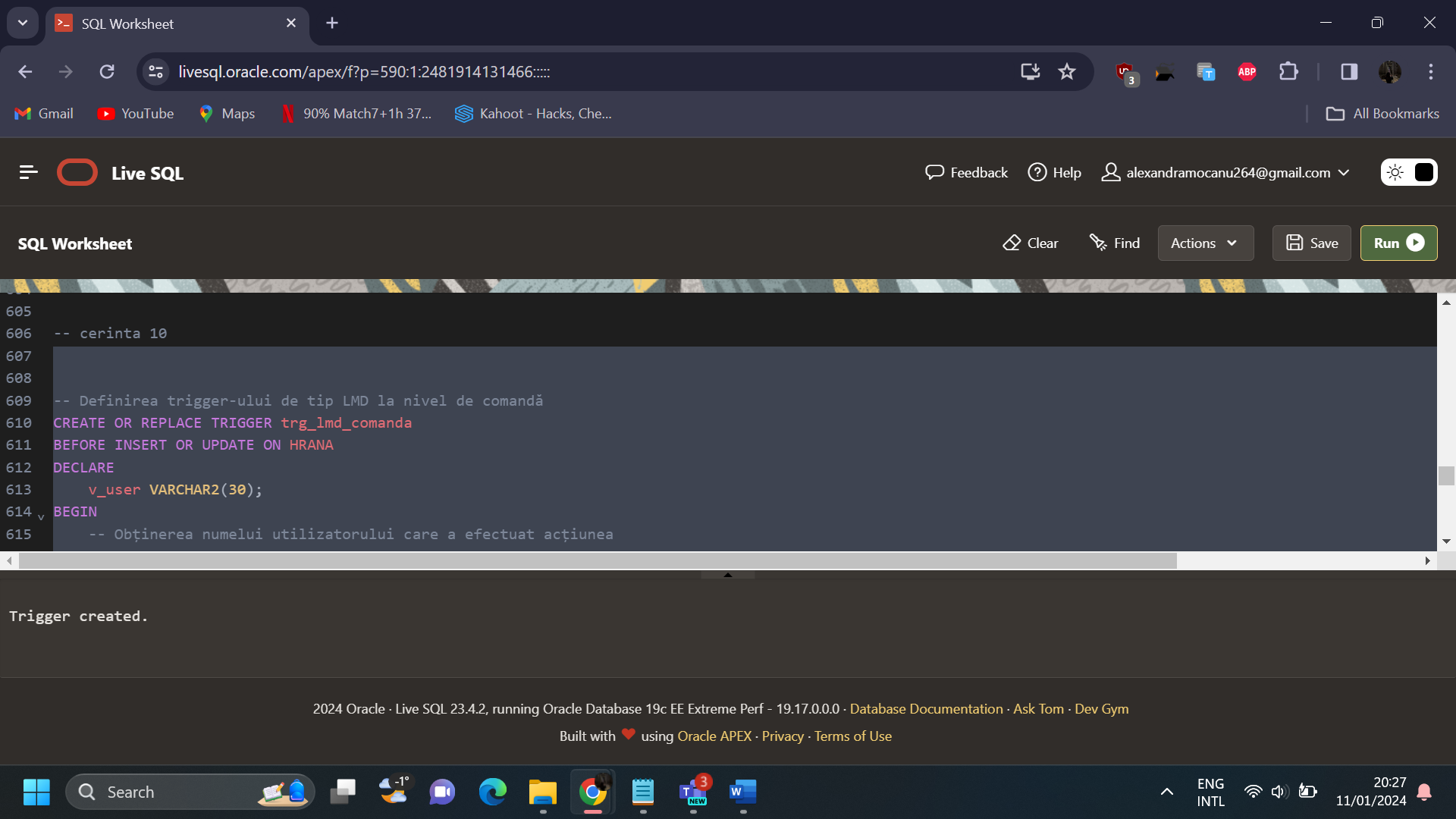
/

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

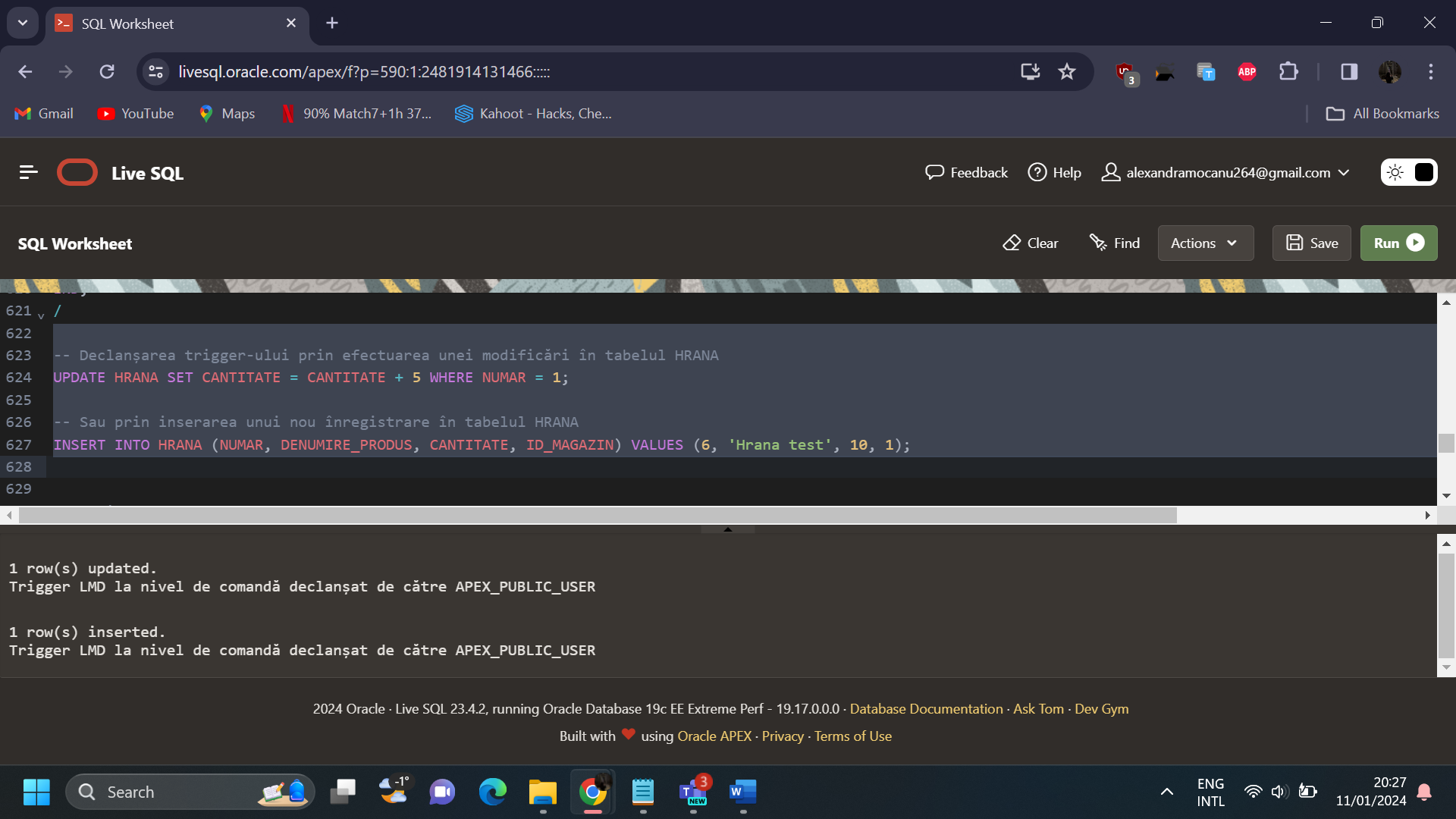
Triggerul de tip LMD (Logon/Modification/Delete) la nivel de comandă pentru tabela HRANA este declanșat înainte de operațiile de inserare (INSERT) sau actualizare (UPDATE) asupra tabelei menționate.

Trigger-ul afișează un mesaj în consola DBMS\_OUTPUT, indicând declanșarea acțiunii și numele utilizatorului care a efectuat operația.

Pentru a testa funcționalitatea trigger-ului, s-au efectuat operațiile de actualizare și inserare în tabela HRANA, generând astfel mesaje în consolă care confirmă declanșarea trigger-ului.



Declansarea Trigger-ului:



-- Definirea trigger-ului de tip LMD la nivel de comanda

CREATE OR REPLACE TRIGGER trg\_lmd\_comanda

BEFORE INSERT OR UPDATE ON HRANA

DECLARE

v\_user VARCHAR2(30);

BEGIN

-- Obținerea numelui utilizatorului care a efectuat acțiunea

SELECT USER INTO v\_user FROM DUAL;

-- Afișarea unui mesaj înainte de efectuarea operațiunii

DBMS\_OUTPUT.PUT\_LINE('Trigger LMD la nivel de comanda declanșat de catre ' || v\_user);

END;

-- Declanșarea trigger-ului prin efectuarea unei modificari în tabelul HRANA

UPDATE HRANA SET CANTITATE = CANTITATE + 5 WHERE NUMAR = 1;

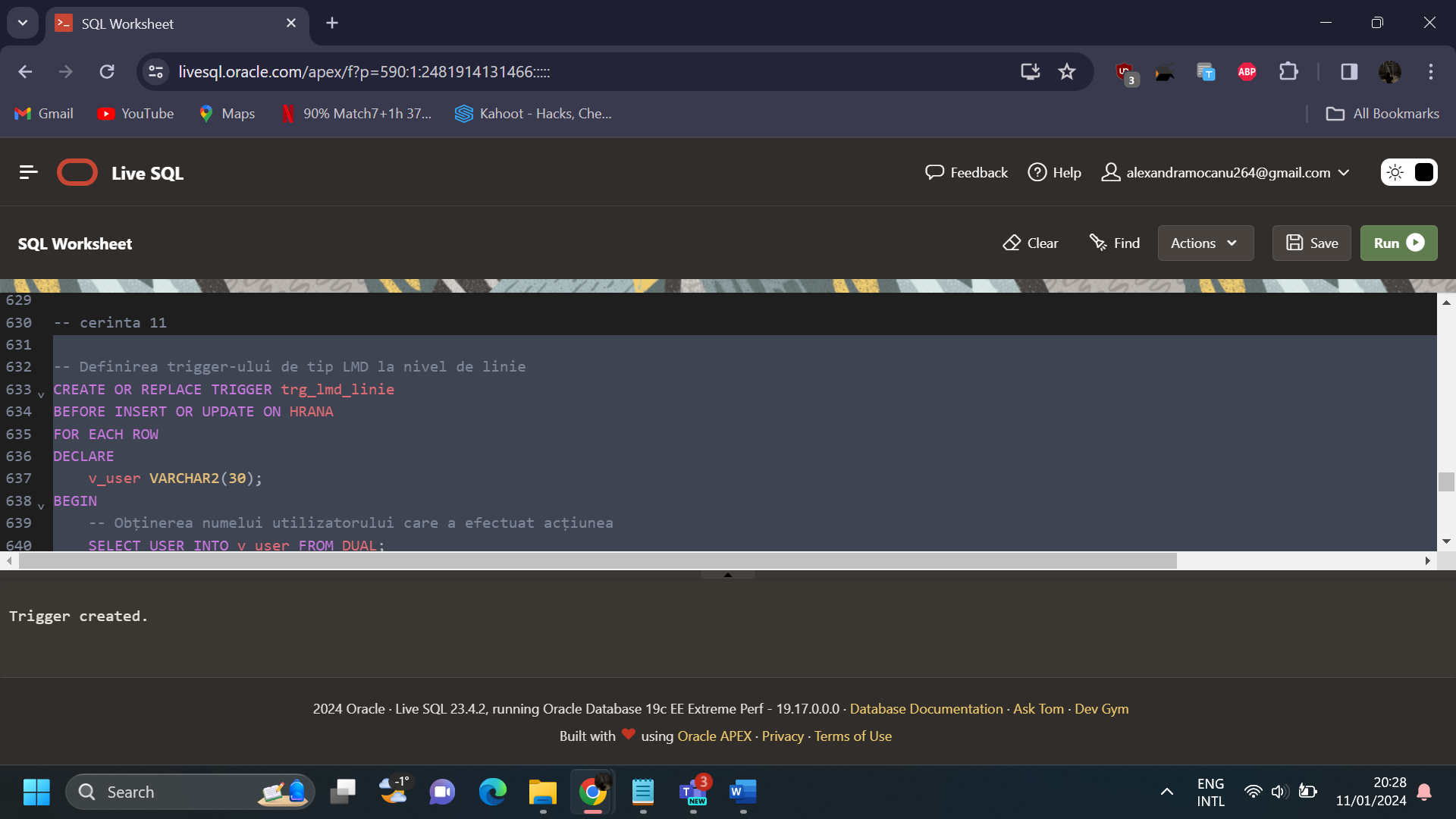
-- Sau prin inserarea unui nou înregistrare în tabelul HRANA

INSERT INTO HRANA (NUMAR, DENUMIRE\_PRODUS, CANTITATE, ID\_MAGAZIN) VALUES (6, 'Hrana test', 10, 1);

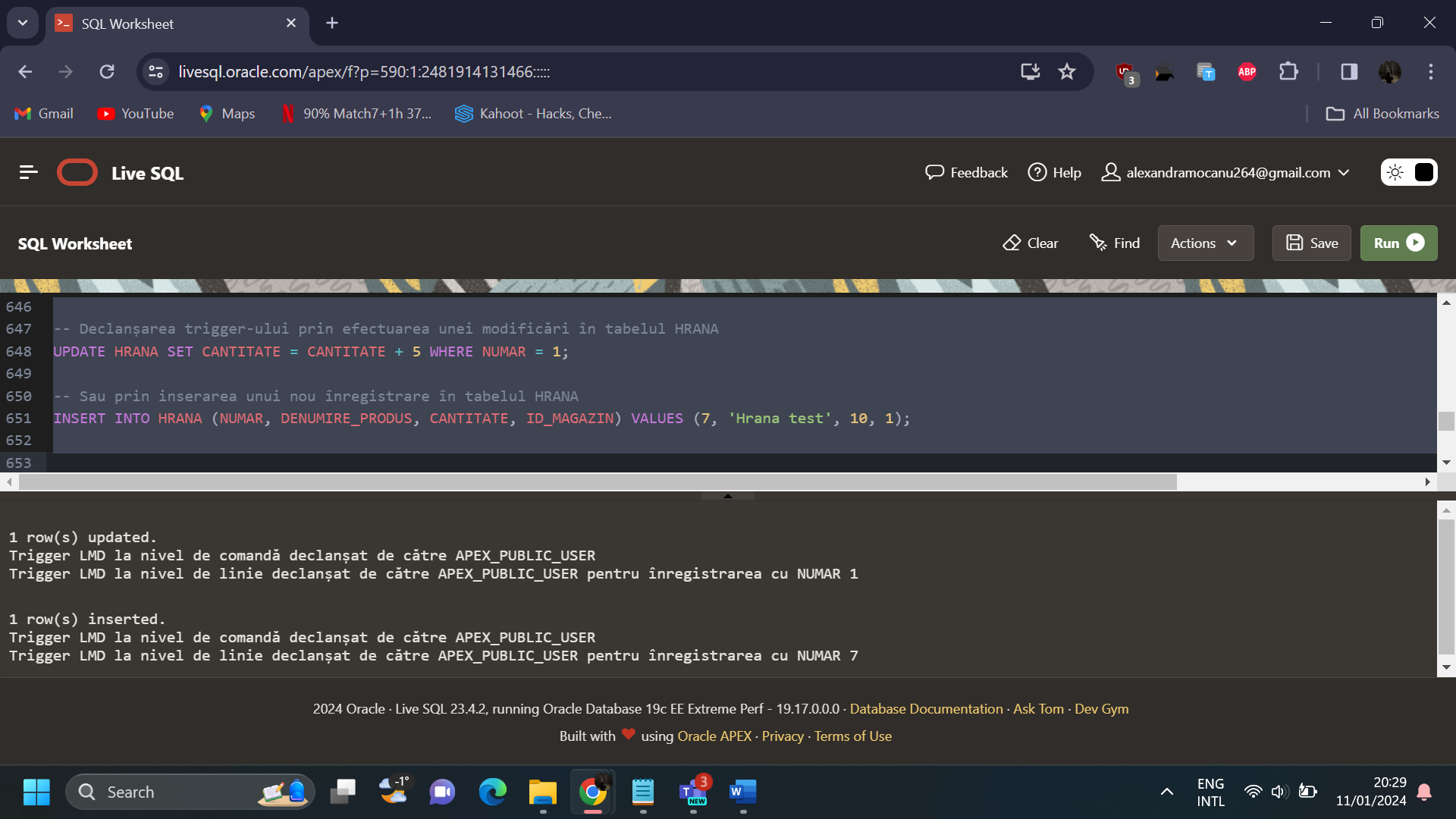
1. Definiți un trigger de tip LMD la nivel de linie. Declanșați trigger-ul.

Am definit un trigger de tip LMD la nivel de linie pentru tabela HRANA. Trigger-ul este declanșat înainte de operațiile de inserare (INSERT) sau actualizare (UPDATE), și este configurat pentru a se aplica pentru fiecare înregistrare (FOR EACH ROW).

În cadrul trigger-ului, obțin numele utilizatorului care a efectuat acțiunea și afișez un mesaj înainte de operațiunea pentru fiecare înregistrare. Astfel, trigger-ul reacționează la modificările de date la nivel de linie și furnizează informații utile despre cine a efectuat acțiunea și pentru ce înregistrare.



Declansarea trigger-ului:



-- Definirea trigger-ului de tip LMD la nivel de linie

CREATE OR REPLACE TRIGGER trg\_lmd\_linie

BEFORE INSERT OR UPDATE ON HRANA

FOR EACH ROW

DECLARE

v\_user VARCHAR2(30);

BEGIN

-- Obținerea numelui utilizatorului care a efectuat acțiunea

SELECT USER INTO v\_user FROM DUAL;

-- Afișarea unui mesaj înainte de efectuarea operațiunii pentru fiecare înregistrare

DBMS\_OUTPUT.PUT\_LINE('Trigger LMD la nivel de linie declanșat de catre ' || v\_user || ' pentru înregistrarea cu NUMAR ' || :NEW.NUMAR);

END;

/

-- Declanșarea trigger-ului prin efectuarea unei modificari în tabelul HRANA

UPDATE HRANA SET CANTITATE = CANTITATE + 5 WHERE NUMAR = 1;

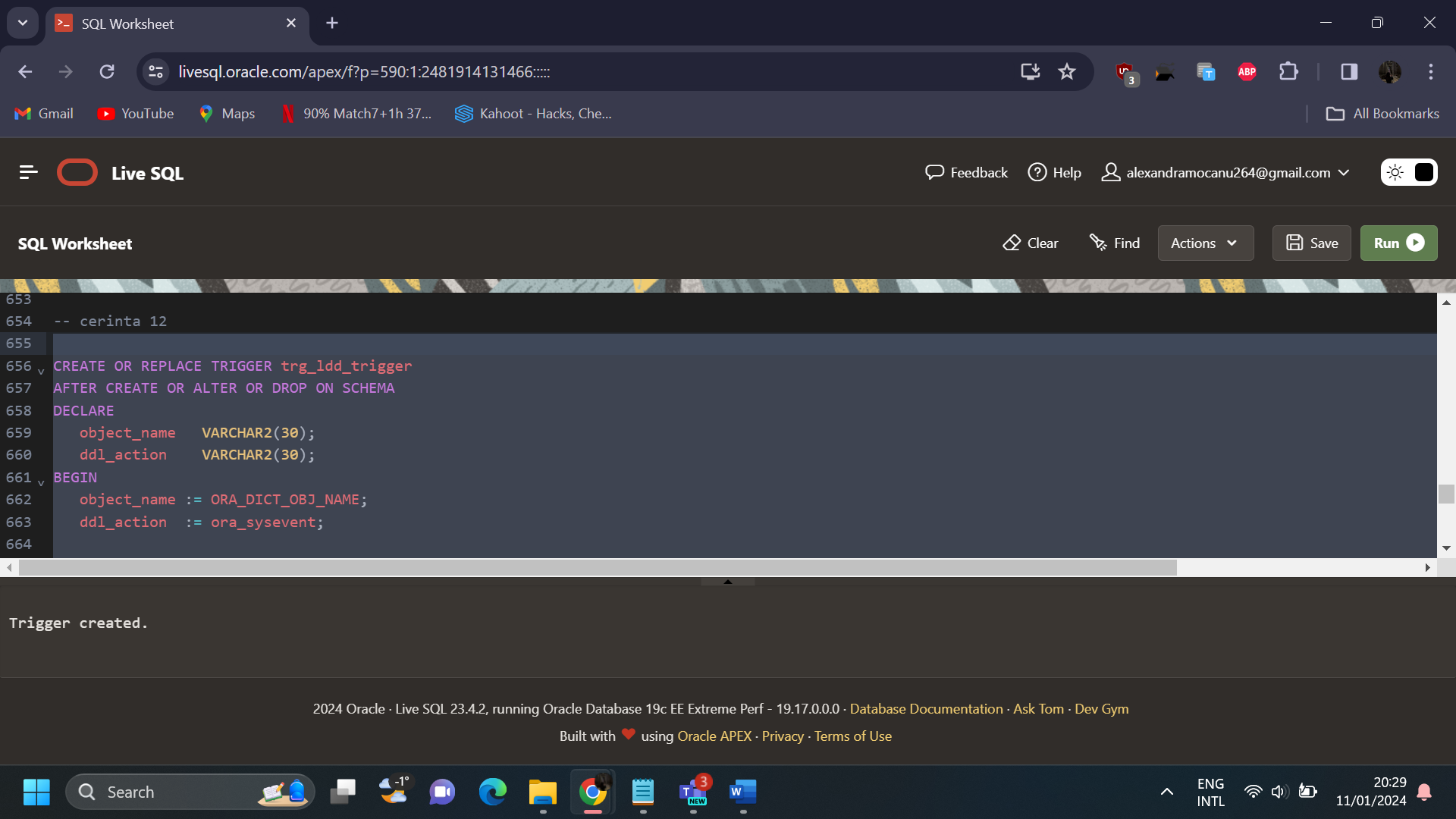
-- Sau prin inserarea unui nou înregistrare în tabelul HRANA

INSERT INTO HRANA (NUMAR, DENUMIRE\_PRODUS, CANTITATE, ID\_MAGAZIN) VALUES (7, 'Hrana test', 10, 1);

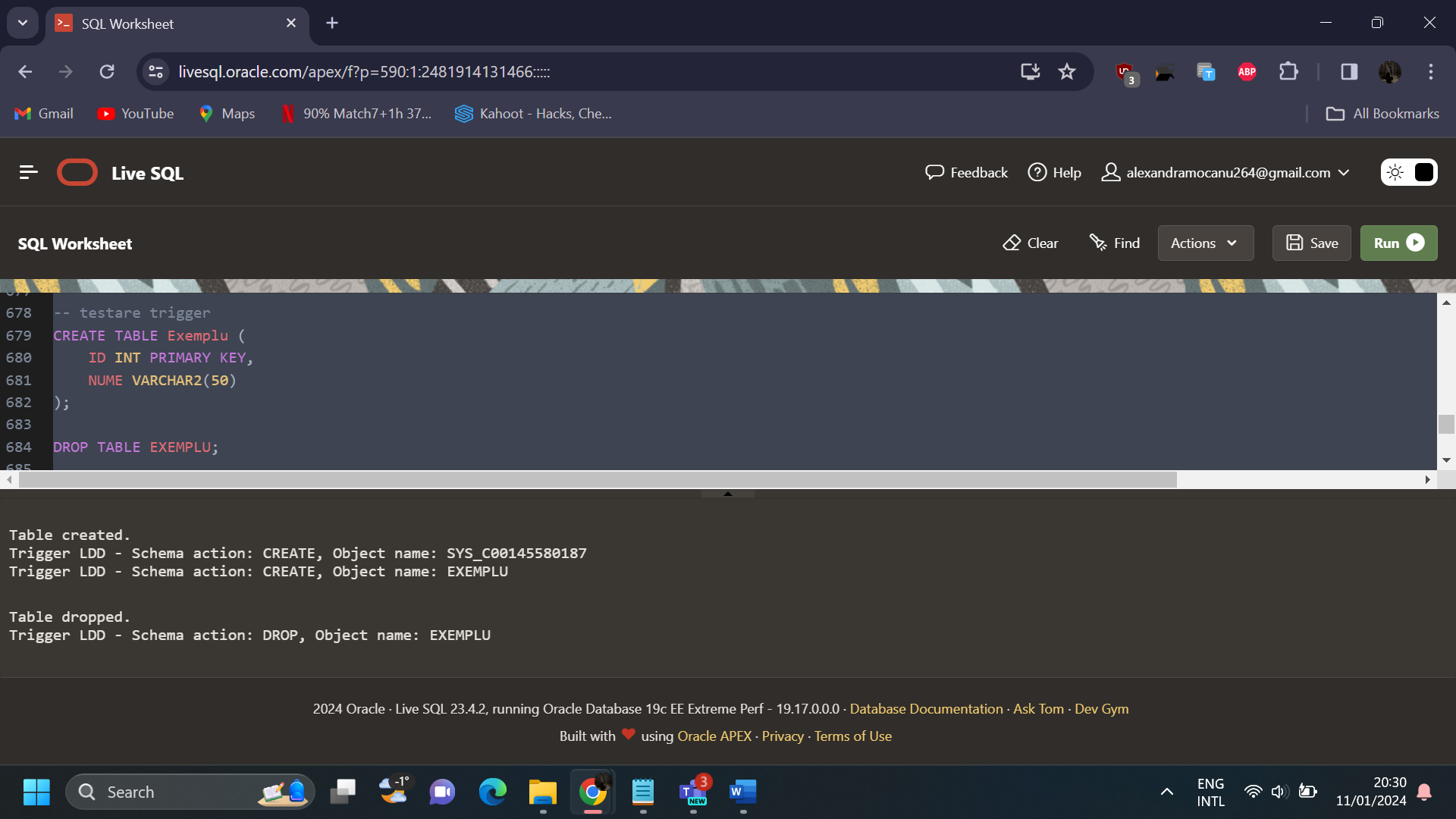
1. Definiți un trigger de tip LDD. Declanșați trigger-ul.

Trigger-ul de tip LDD (Language Definition or Declaration) este declanșat după acțiuni de creare (CREATE), modificare (ALTER) sau ștergere (DROP) asupra obiectelor din schema curentă.

În cadrul testului, a fost creată o tabelă numită "Exemplu" și ulterior ștearsă, declanșând astfel trigger-ul LDD. Mesajul afișat de trigger indică tipul acțiunii DDL (CREATE sau DROP) și numele obiectului asupra căruia s-a realizat acțiunea.



Declansarea trigger-ului:



CREATE OR REPLACE TRIGGER trg\_ldd\_trigger

AFTER CREATE OR ALTER OR DROP ON SCHEMA

DECLARE

object\_name VARCHAR2(30);

ddl\_action VARCHAR2(30);

BEGIN

object\_name := ORA\_DICT\_OBJ\_NAME;

ddl\_action := ora\_sysevent;

-- Afișeaza mesajul cu detaliile acțiunii

DBMS\_OUTPUT.PUT\_LINE('Trigger LDD - Schema action: ' || ddl\_action || ', Object name: ' || object\_name);

END;

/

--SET SERVEROUTPUT ON;

--SELECT \*

--FROM ALL\_ERRORS

--WHERE NAME = 'TRG\_LDD\_TRIGGER';

-- testare trigger

CREATE TABLE Exemplu (

ID INT PRIMARY KEY,

NUME VARCHAR2(50)

);

DROP TABLE EXEMPLU;

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

Pachetul ANIMAL\_SHET include toate obiectele și componente definite în cerințele anterioare (cerințele 6, 7, 8, 9, 10, 11 și 12). Acesta conține:

Tipuri de Date:

NumereAnimale - tip de tabel asociativ pentru numere de animale.

HranaCantitate - tip de tabel asociativ pentru cantități de hrană.

Proceduri:

actualizare\_hrana (cerința 6).

AfisareAnimaleAdoptate (cerința 7).

Problema9 - procedura (cerința 9).

Funcții:

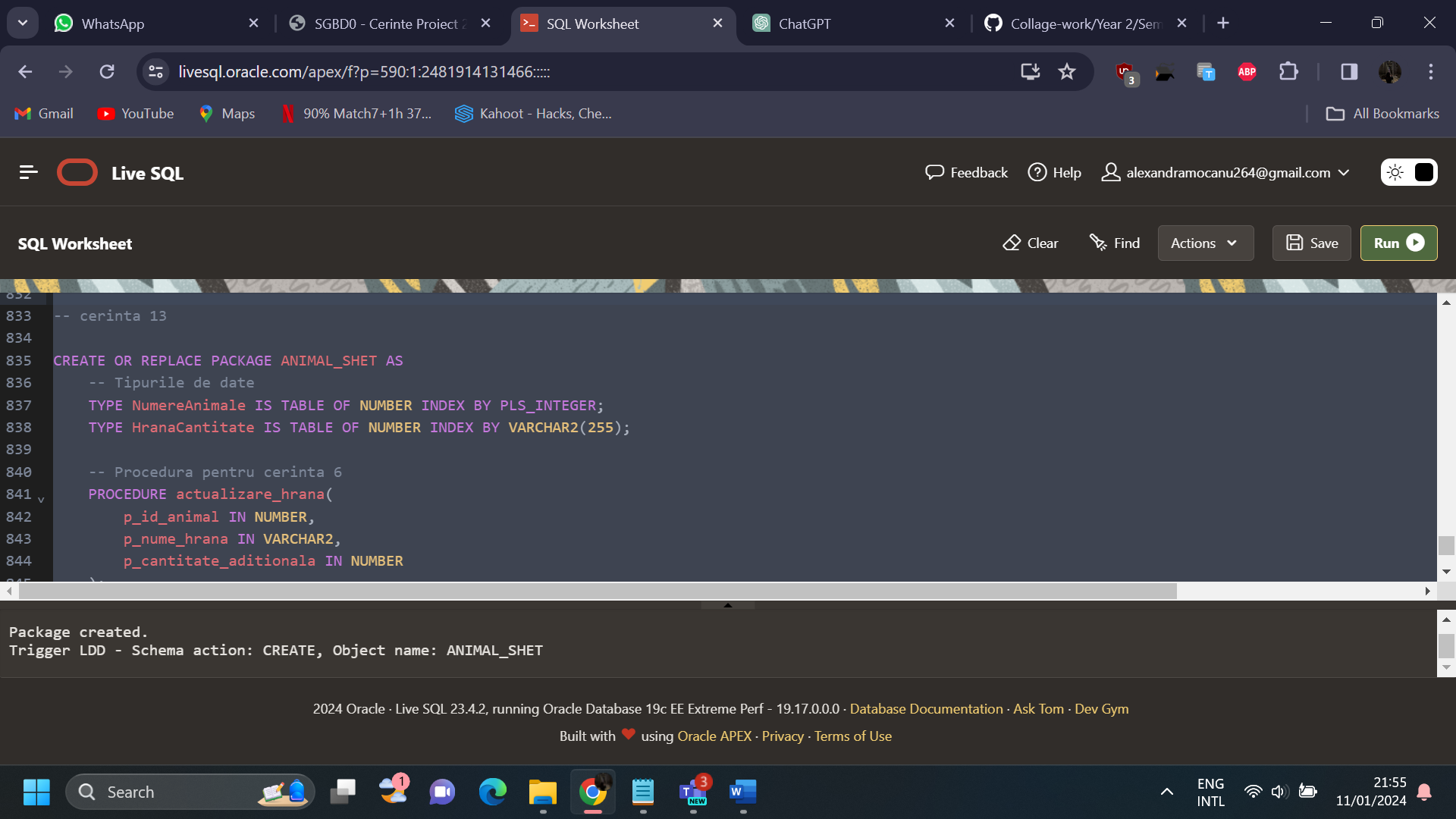
ObtineMedicamentAnimal - funcția pentru (cerința 8).

Triggers:

trg\_lmd\_comanda - trigger de tip LMD la nivel de comandă (cerința 10).

trg\_lmd\_linie - trigger de tip LMD la nivel de linie (cerința 11).

trg\_ldd\_trigger - trigger LDD (cerința 12).



CREATE OR REPLACE PACKAGE ANIMAL\_SHEET AS

-- Tipurile de date

TYPE NumereAnimale IS TABLE OF NUMBER INDEX BY PLS\_INTEGER;

TYPE HranaCantitate IS TABLE OF NUMBER INDEX BY VARCHAR2(255);

-- Procedura pentru cerinta 6

PROCEDURE actualizare\_hrana(

p\_id\_animal IN NUMBER,

p\_nume\_hrana IN VARCHAR2,

p\_cantitate\_aditionala IN NUMBER

);

-- Procedura pentru cerinta 7

PROCEDURE afisare\_animale\_adoptate(

dataAdoptie IN DATE

);

-- Functia pentru cerinta 8

FUNCTION obt\_medicament\_animal(p\_ID\_Animal INT) RETURN VARCHAR2;

-- Procedura pentru cerinta 9

PROCEDURE informatii\_animal(p\_ID\_Animal INT);

-- Trigger pentru cerinta 10

-- Trigger de tip LMD la nivel de comanda

PROCEDURE trg\_lmd\_comanda;

-- Trigger pentru cerinta 11

-- Trigger de tip LMD la nivel de linie

PROCEDURE trg\_lmd\_linie;

-- Trigger pentru cerinta 12

PROCEDURE trg\_ldd\_trigger;

END ANIMAL\_SHEET;

/

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).

Tipurile de Date Complexe:

DetaliiAnimal - un RECORD care stochează informații despre un animal.

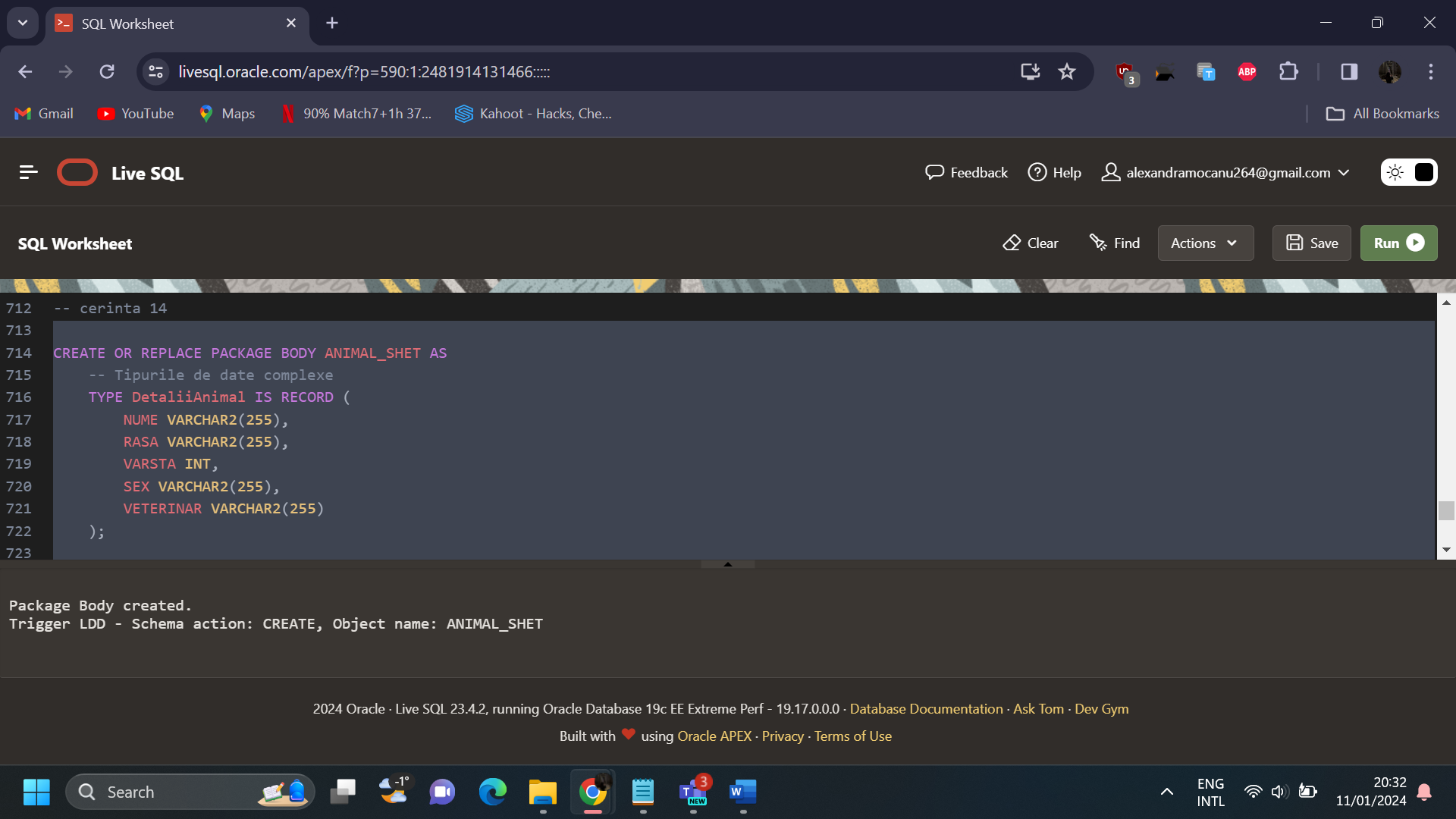
DetaliiHrana - un RECORD care stochează informații despre hrana unui animal.

Proceduri și Funcții:

actualizare\_hrana - procedura pentru actualizarea cantității de hrană, care utilizează tipurile de date complexe definite.

cursor\_parametrizat - funcție care deschide un cursor parametrizat pentru a returna informații despre hrana în funcție de numele hranei furnizat.

cursor\_dependent - funcție care deschide un cursor dependent pentru a returna informații despre animalele care consumă hrana cu un anumit nume.



CREATE OR REPLACE PACKAGE BODY ANIMAL\_SHEET AS

-- Tipurile de date complexe

TYPE DetaliiAnimal IS RECORD (

NUME VARCHAR2(255),

RASA VARCHAR2(255),

VARSTA INT,

SEX VARCHAR2(255),

VETERINAR VARCHAR2(255)

);

TYPE DetaliiHrana IS RECORD (

NUMAR INT,

DENUMIRE\_PRODUS VARCHAR2(255),

CANTITATE INT

);

-- Procedura pentru actualizarea hranei

PROCEDURE actualizare\_hrana(

p\_id\_animal IN NUMBER,

p\_nume\_hrana IN VARCHAR2,

p\_cantitate\_aditionala IN NUMBER

) AS

-- Definirea variabilei cantitate\_actualizata

cantitate\_actualizata NUMBER;

BEGIN

-- Detalii suplimentare pentru animale

FOR animal IN (SELECT A.NUME, A.RASA, A.VARSTA, A.SEX, V.NUME || ' ' || V.PRENUME AS VETERINAR

FROM ANIMAL A

JOIN VETERINAR V ON A.ID\_VETERINAR = V.CNP

WHERE A.ID\_ANIMAL = p\_id\_animal) LOOP

-- Afisarea detaliilor animalelor

DBMS\_OUTPUT.PUT\_LINE('Detalii despre animal: ' || animal.NUME || ', ' || animal.RASA || ', ' ||

animal.VARSTA || ' ani, ' || animal.SEX || ', Veterinar: ' || animal.VETERINAR);

END LOOP;

-- Initializarea colectiei cu cantitatile de hrana necesare

FOR hrana IN (SELECT NUMAR, DENUMIRE\_PRODUS, CANTITATE + p\_cantitate\_aditionala AS CANTITATE

FROM HRANA

WHERE NUMAR = p\_id\_animal AND DENUMIRE\_PRODUS = p\_nume\_hrana) LOOP

cantitate\_actualizata := hrana.CANTITATE;

-- Actualizarea cantitatii de hrana

UPDATE HRANA

SET CANTITATE = cantitate\_actualizata

WHERE NUMAR = hrana.NUMAR AND DENUMIRE\_PRODUS = hrana.DENUMIRE\_PRODUS;

-- Afisarea detaliilor hranei actualizate

DBMS\_OUTPUT.PUT\_LINE('Hrana actualizata: ' || hrana.DENUMIRE\_PRODUS || ', Cantitate: ' || cantitate\_actualizata);

END LOOP;

END actualizare\_hrana;

-- Functia cursor\_parametrizat

FUNCTION cursor\_parametrizat(p\_nume\_hrana VARCHAR2) RETURN SYS\_REFCURSOR IS

v\_cursor SYS\_REFCURSOR;

BEGIN

-- Deschiderea cursorului parametrizat

OPEN v\_cursor FOR

SELECT NUMAR, DENUMIRE\_PRODUS, CANTITATE

FROM HRANA

WHERE DENUMIRE\_PRODUS = p\_nume\_hrana;

-- Returnarea cursorului

RETURN v\_cursor;

END cursor\_parametrizat;

-- Functia cursor\_dependent

FUNCTION cursor\_dependent(p\_nume\_hrana VARCHAR2) RETURN SYS\_REFCURSOR IS

v\_cursor SYS\_REFCURSOR;

BEGIN

-- Deschiderea cursorului dependent

OPEN v\_cursor FOR

SELECT A.NUME, A.RASA, H.NUMAR, H.DENUMIRE\_PRODUS, H.CANTITATE

FROM ANIMAL A

JOIN HRANA H ON A.ID\_ANIMAL = H.NUMAR

WHERE H.DENUMIRE\_PRODUS = p\_nume\_hrana;

-- Returnarea cursorului

RETURN v\_cursor;

END cursor\_dependent;

-- Procedura pentru cerinta 7

PROCEDURE afisare\_animale\_adoptate(

dataAdoptie IN DATE

) IS

-- Definirea variabilei de tip NumereAnimale

numere\_animele NumereAnimale;

BEGIN

-- Initializarea variabilei de tip NumereAnimale

numere\_animele(1) := 1;

numere\_animele(2) := 2;

-- Afisarea numerelor animalelor adoptate

FOR i IN 1..numere\_animele.COUNT LOOP

DBMS\_OUTPUT.PUT\_LINE('Animal adoptat: ' || numere\_animele(i) || ', Data adoptiei: ' || TO\_CHAR(dataAdoptie, 'DD-MON-YYYY'));

END LOOP;

END afisare\_animale\_adoptate;

-- Functia pentru cerinta 8

FUNCTION obt\_medicament\_animal(p\_ID\_Animal INT) RETURN VARCHAR2 IS

-- Definirea variabilei medicament

medicament VARCHAR2(255);

BEGIN

-- Setarea valorii default pentru medicament

medicament := 'Nu are medicament';

-- Returnarea valorii medicamentului

RETURN medicament;

END obt\_medicament\_animal;

-- Procedura pentru cerinta 9

PROCEDURE informatii\_animal(p\_ID\_Animal INT) IS

-- Definirea variabilei de tip DetaliiAnimal

detalii\_animal DetaliiAnimal;

BEGIN

-- Obtine detaliile animalelor

SELECT A.NUME, A.RASA, A.VARSTA, A.SEX, V.NUME || ' ' || V.PRENUME

INTO detalii\_animal

FROM ANIMAL A

JOIN VETERINAR V ON A.ID\_VETERINAR = V.CNP

WHERE A.ID\_ANIMAL = p\_ID\_Animal;

-- Afisarea detaliilor animalelor

DBMS\_OUTPUT.PUT\_LINE('Detalii despre animal: ' || detalii\_animal.NUME || ', ' || detalii\_animal.RASA || ', ' ||

detalii\_animal.VARSTA || ' ani, ' || detalii\_animal.SEX || ', Veterinar: ' || detalii\_animal.VETERINAR);

END informatii\_animal;

-- Trigger pentru cerinta 10

-- Trigger de tip LMD la nivel de comanda

PROCEDURE trg\_lmd\_comanda IS

BEGIN

-- Implementarea logicii pentru trigger-ul LMD la nivel de comanda

NULL;

END trg\_lmd\_comanda;

-- Trigger pentru cerinta 11

-- Trigger de tip LMD la nivel de linie

PROCEDURE trg\_lmd\_linie IS

BEGIN

-- Implementarea logicii pentru trigger-ul LMD la nivel de linie

NULL;

END trg\_lmd\_linie;

-- Trigger pentru cerinta 12

PROCEDURE trg\_ldd\_trigger IS

BEGIN

-- Implementarea logicii pentru trigger-ul LDD

NULL;

END trg\_ldd\_trigger;

END ANIMAL\_SHEET;

/