MONITORIZAREA activității unei agenții imobiliare

proiect sgbd

**CIURESCU IRINA - ALEXANDRA**

**GRUPA 242**

# Prezentați pe scurt baza de date (utilitatea ei).

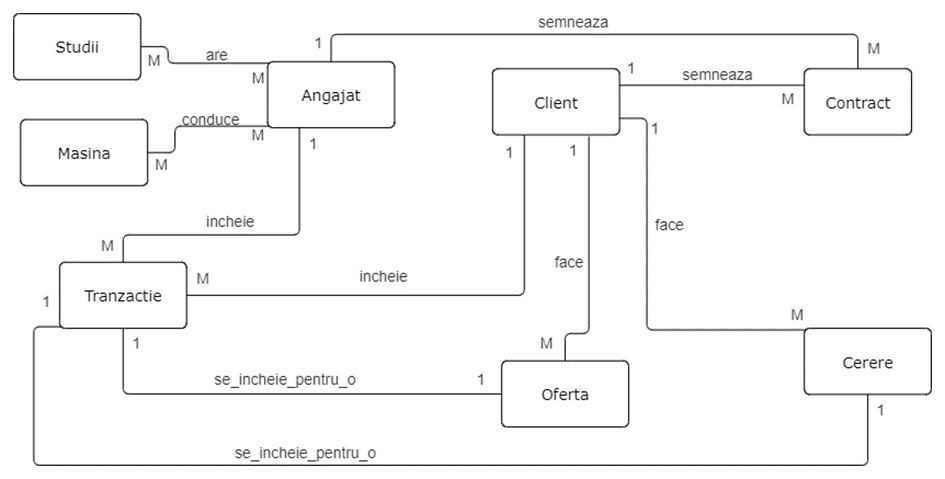
Sistemul informatic al agenției imobiliare a fost conceput pentru a oferi personalului agenției un instrument eficient de gestionare și de accesare a informațiilor despre proprietățile imobiliare și clienții agenției. Aplicația utilizează o bază de date pentru a gestiona ofertele și cererile prezentate, precum și pentru a înregistra contractele și tranzacțiile încheiate.

Principala utilitate a aplicației constă în stocarea eficientă a informațiilor necesare oricărei agenții de acest tip și accesul rapid la acestea, agentul imobiliar putând afla rapid prin intermediul unor interogări simple detalii despre orice proprietate sau contract de care se ocupă. În plus, acesta poate afișa cererile potrivite pentru o ofertă, ofertele potrivite pentru o cerere, ofertele existente pentru fiecare județ și localitate, numărul de contracte încheiate pe o perioadă specificată și altele.

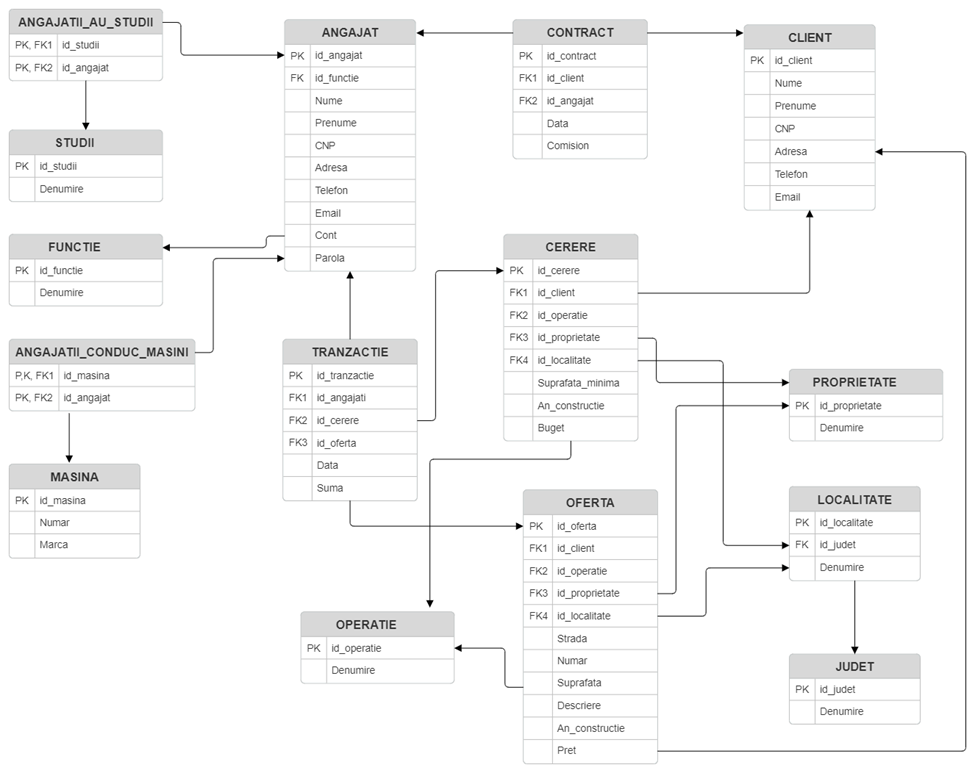
Așadar, activitățile de monitorizare constau în:

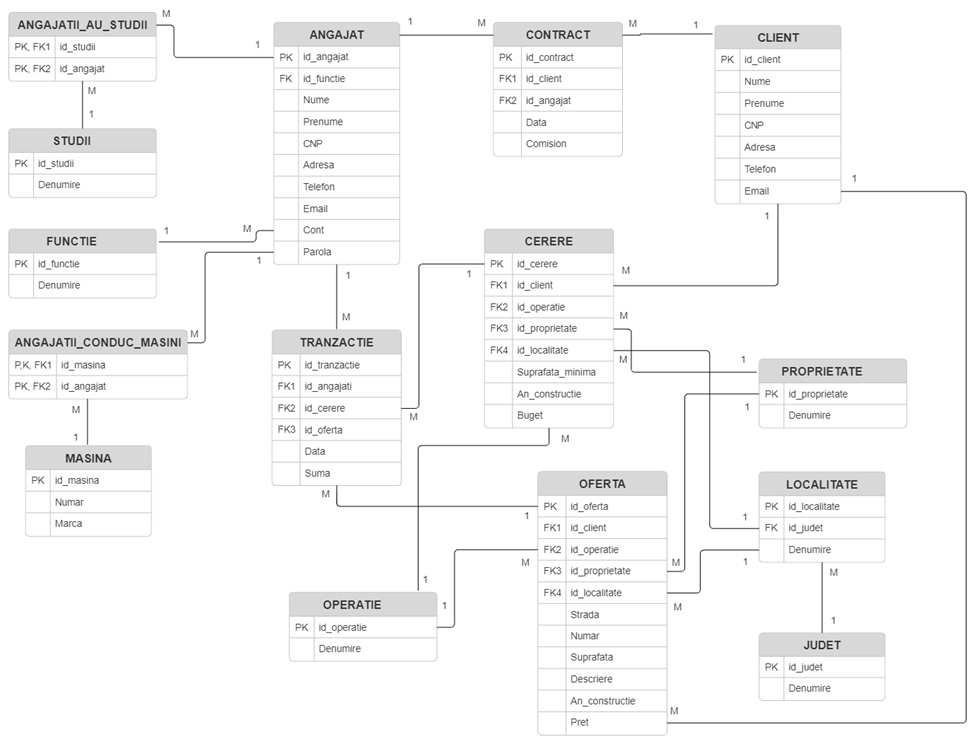
* **Gestiunea utilizatorilor (angajaților)**
  + Memorarea datelor personale ale angajaților (cod, nume, prenume, CNP, studii, funcție, adresă, telefon, email);
  + Stocarea datelor de logare în aplicație (cont și parolă).
* **Gestiunea clienților**
* Memorarea informațiilor personale ale clienților (cod, nume, prenume, CNP, adresă, telefon, email).
* **Gestionarea cererilor și ofertelor**
* Pentru fiecare cerere/ofertă se vor memora datele proprietarului (cod client) și principalele caracteristici: cod proprietate, tipul operației (cumpărare / vânzare/ închiriere), tipul imobilului (casa, apartament, teren), adresa, suprafață, an construcție, buget/ preț, descriere proprietate etc.
* **Gestiunea contractelor și tranzacțiilor încheiate**:
* Pentru fiecare client se întocmește un contract care conține detalii precum numărul contractului, codul agentului, codul clientului, data contractului și comisionul.
* Pentru fiecare tranzacție încheiată, se înregistrează detalii cum ar fi codul tranzacției, data, codul agentului, codurile clienților pentru vânzător și cumpărător, codul ofertei, codul cererii, suma plătită și comisioanele percepute pentru vânzător și cumpărător.

# Realizați diagrama entitate-relație (ERD)



# Pornind de la diagrama entitate-relație realizați diagrama conceptuală a modelului propus, integrând toate atributele necesare





# Implementați în Oracle diagrama conceptuală realizată: definiți toate tabelele, definind toate constrângerile de integritate necesare (chei primare, cheile externe etc).

CREATE TABLE JUDET

(ID\_JUDET VARCHAR2(2) PRIMARY KEY,

DENUMIRE VARCHAR2(20) NOT NULL

);

CREATE TABLE LOCALITATE

(ID\_LOCALITATE NUMBER(10) PRIMARY KEY,

ID\_JUDET VARCHAR2(2),

DENUMIRE VARCHAR2(20) NOT NULL

);

ALTER TABLE LOCALITATE

ADD CONSTRAINT FKJUDET FOREIGN KEY

(

ID\_JUDET

)

REFERENCES JUDET

(

ID\_JUDET

);

CREATE TABLE STUDII

(ID\_STUDII VARCHAR2(1) PRIMARY KEY,

DENUMIRE VARCHAR2(20) NOT NULL

);

CREATE TABLE FUNCTIE

(ID\_FUNCTIE NUMBER(2) PRIMARY KEY,

DENUMIRE VARCHAR2(20) NOT NULL

);

CREATE TABLE OPERATIE

(ID\_OPERATIE VARCHAR2(1) PRIMARY KEY,

DENUMIRE VARCHAR2(20) NOT NULL

);

CREATE TABLE PROPRIETATE

(ID\_PROPRIETATE VARCHAR2(1) PRIMARY KEY,

DENUMIRE VARCHAR2(20) NOT NULL

);

CREATE TABLE ANGAJAT

(Id\_angajat Number(3) PRIMARY KEY,

Id\_functie Number(2) Not null,

Nume Varchar2(20) Not null,

Prenume Varchar2(20) Not null,

Cnp Varchar2(13) Not null,

Adresa Varchar2(30) Not null,

Telefon Varchar2(10) Not null,

email Varchar2(20) Not null,

Cont Varchar2(20) Not null,

parola Varchar2(20) Not null

);

ALTER TABLE ANGAJAT

ADD CONSTRAINT FKFUNCTIE FOREIGN KEY

(

ID\_FUNCTIE

)

REFERENCES FUNCTIE

(

ID\_FUNCTIE

);

CREATE TABLE ANG\_STUDII

(Id\_angajat Number(3) Not null,

Id\_studii Varchar2(1) Not null

);

--ADAUGARE CHEIE PRIMARA LA ANG\_STUDII

--ADAUGARE KEY EXTERNE LA ANG\_STUDII

ALTER TABLE ANG\_STUDII ADD (

CONSTRAINT ANG\_STUD\_PK PRIMARY KEY (ID\_ANGAJAT,ID\_STUDII),

CONSTRAINT AS\_ID\_ANGAJAT\_FK FOREIGN KEY (ID\_ANGAJAT) REFERENCES ANGAJAT(ID\_ANGAJAT),

CONSTRAINT AS\_ID\_SUDII\_FK FOREIGN KEY (ID\_STUDII) REFERENCES STUDII(ID\_STUDII)

);

CREATE TABLE MASINA

(Id\_masina Number(3) primary key,

Numar Varchar2(10) Not null,

Marca Varchar2(20) Not null

);

CREATE TABLE ANG\_MAS

(Id\_masina Number(3) Not null,

Id\_angajat Number(3) Not null

);

--ADAUGARE CHEIE PRIMARA LA ANG\_MAS

--ADAUGARE KEY EXTERNE LA ANG\_MAS

ALTER TABLE ANG\_MAS ADD (

CONSTRAINT ANG\_MAS\_PK PRIMARY KEY (ID\_MASINA,ID\_ANGAJAT),

CONSTRAINT AS\_ID\_ANGMA\_FK FOREIGN KEY (ID\_ANGAJAT) REFERENCES ANGAJAT(ID\_ANGAJAT),

CONSTRAINT AS\_ID\_MASAN\_FK FOREIGN KEY (ID\_MASINA) REFERENCES MASINA(ID\_MASINA)

);

CREATE TABLE CLIENT

(Id\_client Number(5) primary key,

Nume Varchar2(20) Not null,

Prenume Varchar2(20) Not null,

Cnp Varchar2(13) Not null,

Adresa Varchar2(30) Not null,

Telefon Varchar2(10) Not null,

email Varchar2(20)

);

CREATE TABLE OFERTA

(Id\_oferta Number(5) PRIMARY KEY,

Id\_client Number(5) Not null,

Id\_operatie Varchar2(1) Not null,

Id\_proprietate Varchar2(1) Not null,

Id\_localitate Number(10) Not null,

Strada Varchar2(20) Not null,

Suprafata Number(10,2) CHECK (SUPRAFATA >0),

An\_constructie Number(4) CHECK (An\_constructie >0),

Pret Number(10) CHECK (Pret >0),

Descriere Varchar2(100),

Numar Varchar2(5)

);

ALTER TABLE oferta

ADD CONSTRAINT FKCLIENT FOREIGN KEY

(

ID\_CLIENT

)

REFERENCES CLIENT

(

ID\_CLIENT

);

ALTER TABLE oferta

ADD CONSTRAINT FKOPERATIE FOREIGN KEY

(

ID\_OPERATIE

)

REFERENCES OPERATIE

(

ID\_OPERATIE

);

ALTER TABLE oferta

ADD CONSTRAINT FKPROPRIETATE FOREIGN KEY

(

ID\_PROPRIETATE

)

REFERENCES PROPRIETATE

(

ID\_PROPRIETATE

);

ALTER TABLE oferta

ADD CONSTRAINT FKLOCALITATE FOREIGN KEY

(

ID\_LOCALITATE

)

REFERENCES LOCALITATE

(

ID\_LOCALITATE

);

CREATE TABLE CERERE

( Id\_cerere Number(5) PRIMARY KEY,

Id\_client Number(5) Not null,

Id\_operatie Varchar2(1) Not null,

Id\_proprietate Varchar2(1) Not null,

Id\_localitate Number(10) Not null,

Buget Number(10) check( buget>0),

Suprafata\_minima Number(10,2) CHECK(SUPRAFATA\_MINIMA>0),

An\_construcTie Number(4) CHECK(AN\_CONSTRUCTIE>0)

);

ALTER TABLE CERERE

ADD CONSTRAINT FKCLIENT2 FOREIGN KEY

(

ID\_CLIENT

)

REFERENCES CLIENT

(

ID\_CLIENT

);

ALTER TABLE CERERE

ADD CONSTRAINT FKOPERATIE2 FOREIGN KEY

(

ID\_OPERATIE

)

REFERENCES OPERATIE

(

ID\_OPERATIE

);

ALTER TABLE CERERE

ADD CONSTRAINT FKPROPRIETATE2 FOREIGN KEY

(

ID\_PROPRIETATE

)

REFERENCES PROPRIETATE

(

ID\_PROPRIETATE

);

ALTER TABLE CERERE

ADD CONSTRAINT FKLOCALITATE2 FOREIGN KEY

(

ID\_LOCALITATE

)

REFERENCES LOCALITATE

(

ID\_LOCALITATE

);

CREATE TABLE CONTRACT

(Id\_contract Number(5) PRIMARY KEY,

Id\_angajat Number(3) Not null,

Id\_client Number(5) Not null,

Data Date DEFAULT (SYSDATE),

Comision Number(2) CHECK(COMISION>0)

);

ALTER TABLE CONTRACT

ADD CONSTRAINT FKANGAJAT3 FOREIGN KEY

(

ID\_ANGAJAT

)

REFERENCES ANGAJAT

(

ID\_ANGAJAT

);

ALTER TABLE CONTRACT

ADD CONSTRAINT FKCLIENT3 FOREIGN KEY

(

ID\_CLIENT

)

REFERENCES CLIENT

(

ID\_CLIENT

);

CREATE TABLE TRANZACTIE

(Id\_tranzactie Number(5) PRIMARY KEY,

Id\_angajat Number(3) Not null,

Id\_cerere Number(5) Not null,

Id\_oferta Number(5) Not null,

Data Date DEFAULT SYSDATE,

Suma Number(10) CHECK(SUMA>0)

);

ALTER TABLE TRANZACTIE

ADD CONSTRAINT FKANGAJAT4 FOREIGN KEY

(

ID\_ANGAJAT

)

REFERENCES ANGAJAT

(

ID\_ANGAJAT

);

ALTER TABLE TRANZACTIE

ADD CONSTRAINT FKCERERE4 FOREIGN KEY

(

ID\_CERERE

)

REFERENCES CERERE

(

ID\_CERERE

);

ALTER TABLE TRANZACTIE

ADD CONSTRAINT FKOFERTA4 FOREIGN KEY

(

ID\_OFERTA

)

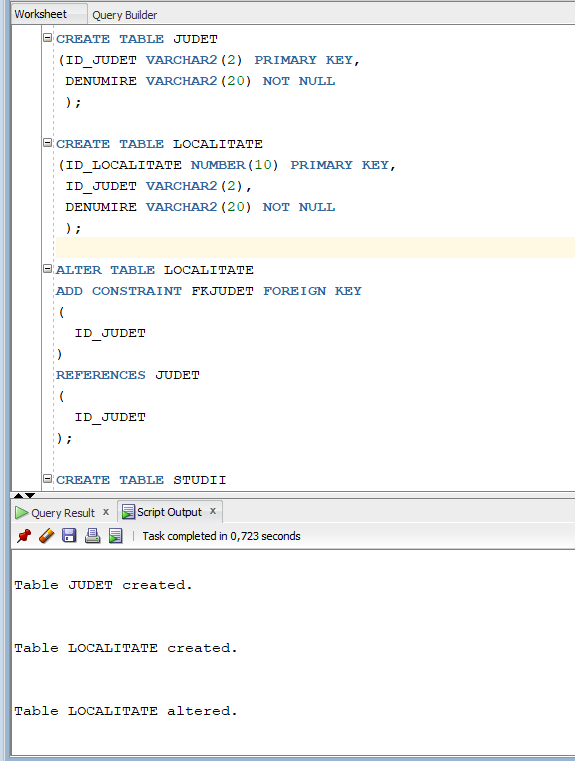
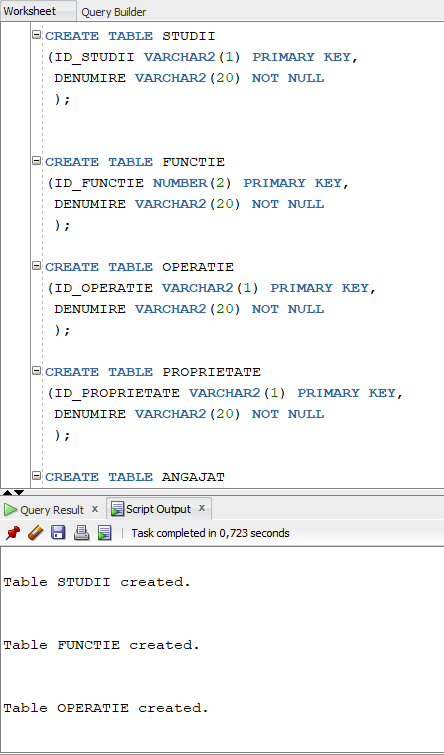
REFERENCES OFERTA

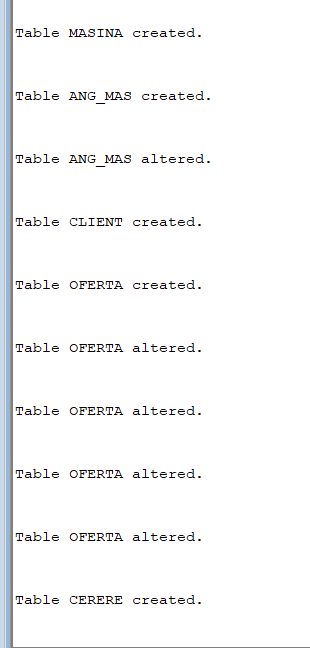
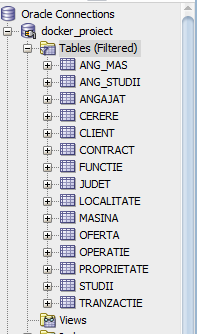
(

ID\_OFERTA

);

**Câteva printscreen-uri pentru a evidenția crearea tabelelor**:





# Adăugați informații coerente în tabelele create (minim 5 înregistrări pentru fiecare entitate independentă; minim 10 înregistrări pentru tabela asociativă).

INSERT INTO JUDET VALUES ('AB', 'Alba');

INSERT INTO JUDET VALUES ('AR', 'Arad');

INSERT INTO JUDET VALUES ('AG', 'Argeş');

INSERT INTO JUDET VALUES ('BC', 'Bacău');

INSERT INTO JUDET VALUES ('BH', 'Bihor');

INSERT INTO JUDET VALUES ('BN', 'Bistriţa-Năsăud');

INSERT INTO JUDET VALUES ('BT', 'Botoşani');

INSERT INTO JUDET VALUES ('BR', 'Brăila');

INSERT INTO JUDET VALUES ('BV', 'Braşov');

INSERT INTO JUDET VALUES ('BZ', 'Buzău');

INSERT INTO JUDET VALUES ('CL', 'Călăraşi');

INSERT INTO JUDET VALUES ('CS', 'Caraş-Severin');

INSERT INTO JUDET VALUES ('CJ', 'Cluj');

INSERT INTO JUDET VALUES ('CT', 'Constanţa');

INSERT INTO JUDET VALUES ('CV', 'Covasna');

INSERT INTO JUDET VALUES ('DB', 'Dâmboviţa');

INSERT INTO JUDET VALUES ('DJ', 'Dolj');

INSERT INTO JUDET VALUES ('GL', 'Galaţi');

INSERT INTO JUDET VALUES ('GR', 'Giurgiu');

INSERT INTO JUDET VALUES ('GJ', 'Gorj');

INSERT INTO JUDET VALUES ('HR', 'Harghita');

INSERT INTO JUDET VALUES ('HD', 'Hunedoara');

INSERT INTO JUDET VALUES ('IL', 'Ialomiţa');

INSERT INTO JUDET VALUES ('IS', 'Iaşi');

INSERT INTO JUDET VALUES ('IF', 'Ilfov');

INSERT INTO JUDET VALUES ('MM', 'Maramureş');

INSERT INTO JUDET VALUES ('MH', 'Mehedinţi');

INSERT INTO JUDET VALUES ('MS', 'Mureş');

INSERT INTO JUDET VALUES ('NT', 'Neamţ');

INSERT INTO JUDET VALUES ('OT', 'Olt');

INSERT INTO JUDET VALUES ('PH', 'Prahova');

INSERT INTO JUDET VALUES ('SJ', 'Sălaj');

INSERT INTO JUDET VALUES ('SM', 'Satu Mare');

INSERT INTO JUDET VALUES ('SB', 'Sibiu');

INSERT INTO JUDET VALUES ('SV', 'Suceava');

INSERT INTO JUDET VALUES ('TR', 'Teleorman');

INSERT INTO JUDET VALUES ('TM', 'Timiş');

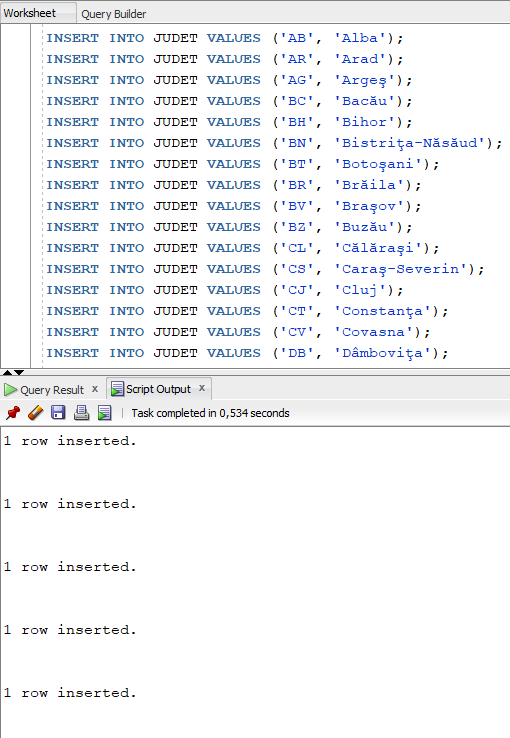
INSERT INTO JUDET VALUES ('TL', 'Tulcea');

INSERT INTO JUDET VALUES ('VL', 'Vâlcea');

INSERT INTO JUDET VALUES ('VS', 'Vaslui');

INSERT INTO JUDET VALUES ('VN', 'Vrancea');

INSERT INTO JUDET VALUES ('B', 'Bucureşti');



INSERT INTO LOCALITATE VALUES (1, 'IS', 'Iasi');

INSERT INTO LOCALITATE VALUES (2, 'B', 'Bucuresti');

INSERT INTO LOCALITATE VALUES (3, 'CT', 'Constanta');

INSERT INTO LOCALITATE VALUES (4, 'CT', 'Mangalia');

INSERT INTO LOCALITATE VALUES (5, 'GL', 'Galati');

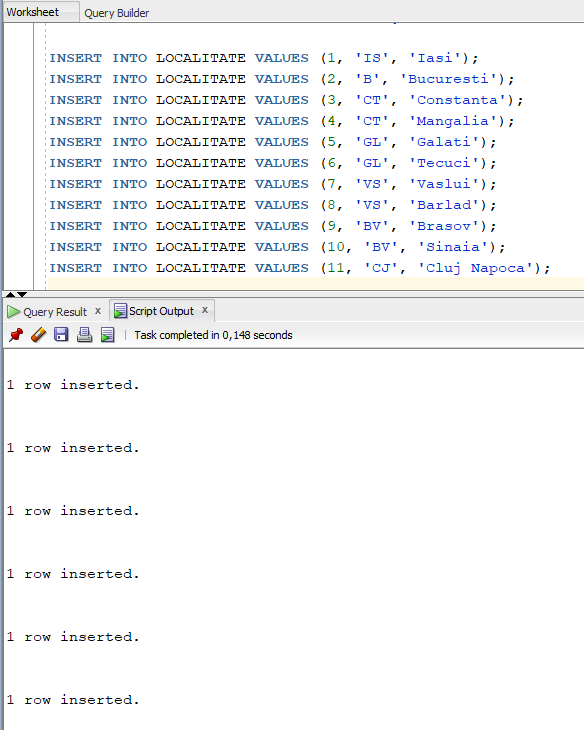
INSERT INTO LOCALITATE VALUES (6, 'GL', 'Tecuci');

INSERT INTO LOCALITATE VALUES (7, 'VS', 'Vaslui');

INSERT INTO LOCALITATE VALUES (8, 'VS', 'Barlad');

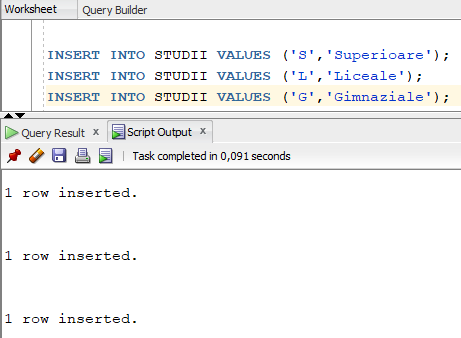
INSERT INTO LOCALITATE VALUES (9, 'BV', 'Brasov');

INSERT INTO LOCALITATE VALUES (10, 'BV', 'Sinaia');

INSERT INTO LOCALITATE VALUES (11, 'CJ', 'Cluj Napoca');

INSERT INTO STUDII VALUES ('S','Superioare');

INSERT INTO STUDII VALUES ('L','Liceale');

INSERT INTO STUDII VALUES ('G','Gimnaziale');

INSERT INTO FUNCTIE VALUES (1,'Agent');

INSERT INTO FUNCTIE VALUES (2,'Contabil');

INSERT INTO FUNCTIE VALUES (3,'Director');

INSERT INTO FUNCTIE VALUES (4,'Sofer');

INSERT INTO FUNCTIE VALUES (5,'Secretara');

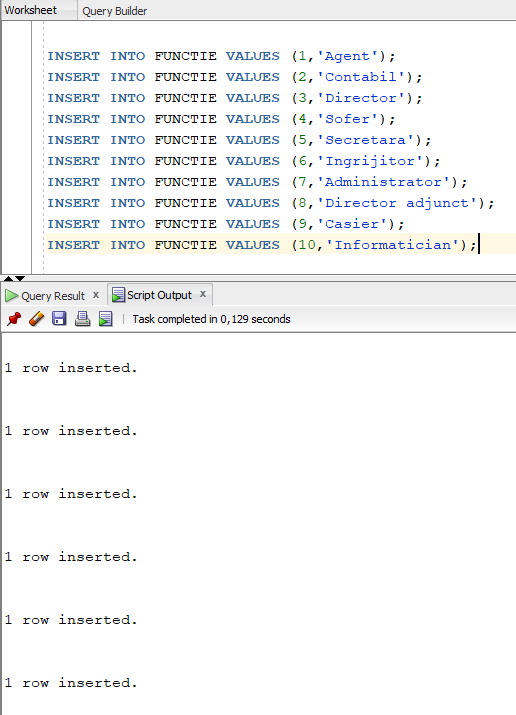
INSERT INTO FUNCTIE VALUES (6,'Ingrijitor');

INSERT INTO FUNCTIE VALUES (7,'Administrator');

INSERT INTO FUNCTIE VALUES (8,'Director adjunct');

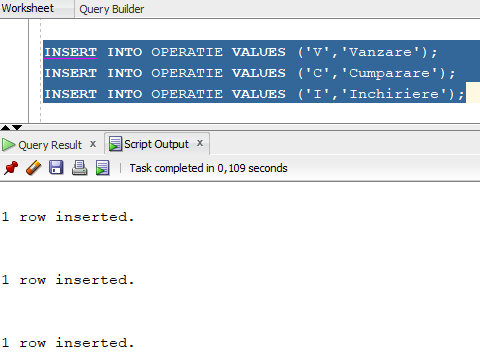
INSERT INTO FUNCTIE VALUES (9,'Casier');

INSERT INTO FUNCTIE VALUES (10,'Informatician');



INSERT INTO OPERATIE VALUES ('V','Vanzare');

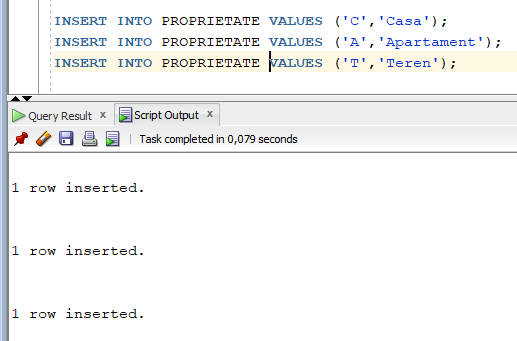
INSERT INTO OPERATIE VALUES ('C','Cumparare');

INSERT INTO OPERATIE VALUES ('I','Inchiriere');

INSERT INTO PROPRIETATE VALUES ('C','Casa');

INSERT INTO PROPRIETATE VALUES ('A','Apartament');

INSERT INTO PROPRIETATE VALUES ('T','Teren');



INSERT INTO ANGAJAT VALUES(1,1,'Ionescu','Vasile','1710612370028','Str. Fagului Nr. 2', '075142561','ionescuv@fx.ro','vionescu','sdkcn');

INSERT INTO ANGAJAT VALUES(2,1,'Popescu','Ionut','1710612370828','Str. Carpenului Nr. 3', '0751427140','popescui@fx.ro','ipopescu','sdkln');

INSERT INTO ANGAJAT VALUES(3,1,'Chirita','Maria','2710912370028','Str. Marului Nr. 12', '075199561','chiritam@fx.ro','mchirita','sdkcnio');

INSERT INTO ANGAJAT VALUES(4,1,'Mincu','Ion','1710612373028','Str. Stejarului Nr. 2', '072142561','mincui@fx.ro','imincu','sdttttkcn');

INSERT INTO ANGAJAT VALUES(5,2,'Cornea','Doina','2710618870028','Str. Macinului Nr. 28', '071192561','cornead@fx.ro','dcornea','aaadkcn');

INSERT INTO ANGAJAT VALUES(6,3,'Ciubotaru','Cornel','1710616660028','Str. Pacii Nr. 101', '075142001','ciubotaruc@fx.ro','cciubotaru','siiidkcn');

INSERT INTO ANGAJAT VALUES(7,5,'Vieru','Ana','2710612370118','Str. Muncii Nr. 2', '074442561','vierua@fx.ro','avieru','sdddfffkcn');

INSERT INTO ANGAJAT VALUES(8,8,'Carare','Raluca','2710612399928','Str. Donici Nr. 12', '075142092','cararer@fx.ro','rcarare','uulodkn');

INSERT INTO ANGAJAT VALUES(9,10,'Taciune','Mihai','1710123470028','Str. Salcamului Nr. 203', '079023561','taciunem@fx.ro','mtaciune','wwqdkcn');

INSERT INTO ANGAJAT VALUES(10,9,'Scanteie','Raluca','2723412370028','Str. Ciresului Nr. 62', '0759042361','scanteier@fx.ro','rscanteie','oosdkcn');



INSERT INTO ANG\_STUDII VALUES(1,'S');

INSERT INTO ANG\_STUDII VALUES(2,'S');

INSERT INTO ANG\_STUDII VALUES(3,'S');

INSERT INTO ANG\_STUDII VALUES(4,'S');

INSERT INTO ANG\_STUDII VALUES(5,'S');

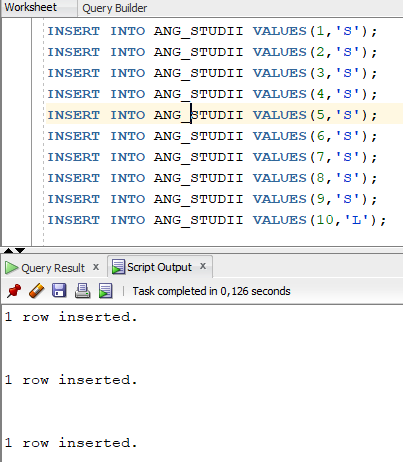
INSERT INTO ANG\_STUDII VALUES(6,'S');

INSERT INTO ANG\_STUDII VALUES(7,'S');

INSERT INTO ANG\_STUDII VALUES(8,'S');

INSERT INTO ANG\_STUDII VALUES(9,'S');

INSERT INTO ANG\_STUDII VALUES(10,'L');



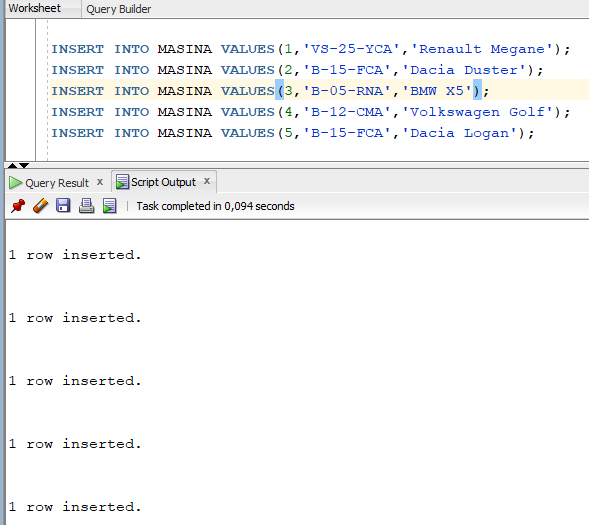
INSERT INTO MASINA VALUES(1,'VS-25-YCA','Renault Megane');

INSERT INTO MASINA VALUES(2,'B-15-FCA','Dacia Duster');

INSERT INTO MASINA VALUES(3,'B-05-RNA','BMW X5');

INSERT INTO MASINA VALUES(4,'B-12-CMA','Volkswagen Golf');

INSERT INTO MASINA VALUES(5,'B-15-FCA','Dacia Logan');



INSERT INTO ANG\_MAS VALUES(1,2);

INSERT INTO ANG\_MAS VALUES(1,4);

INSERT INTO ANG\_MAS VALUES(1,10);

INSERT INTO ANG\_MAS VALUES(2,2);

INSERT INTO ANG\_MAS VALUES(2,6);

INSERT INTO ANG\_MAS VALUES(3,2);

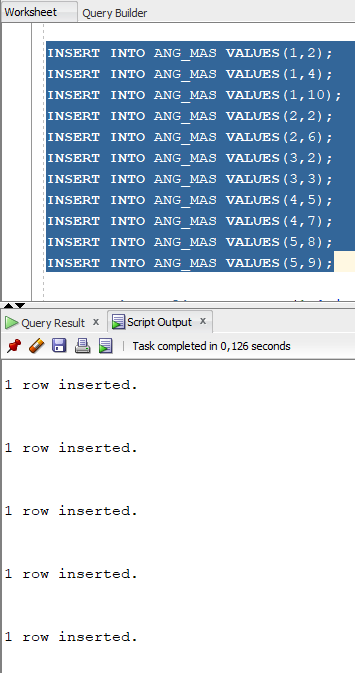
INSERT INTO ANG\_MAS VALUES(3,3);

INSERT INTO ANG\_MAS VALUES(4,5);

INSERT INTO ANG\_MAS VALUES(4,7);

INSERT INTO ANG\_MAS VALUES(5,8);

INSERT INTO ANG\_MAS VALUES(5,9);



INSERT into client VALUES (1,'Cimpoesu','Adrian','1720318174050','Str. Ion Creanga Nr. 4', '0741234518','acimpoesu@gmail.com');

INSERT into client VALUES (2,'Margineanu','Cosmin','1540318374050','Str. Pacii Nr. 4', '0741277518','margcos@gmail.com');

INSERT into client VALUES (3,'Purice','Adrian','1720318374234','Str. Muntelui Nr. 89', '0741890518','apurice@gmail.com');

INSERT into client VALUES (4,'Tarnoveanu','Costel','1720319174050','Str. Soarelui Nr. 5', '0788734518','costelt@gmail.com');

INSERT into client VALUES (5,'Maciuca','Tina','2720316234050','Str. Calugareni Nr. 74', '0741234255','tinamac@gmail.com');

INSERT into client VALUES (6,'Caragea','Adina','2730318274050','Str. Bratianu Nr. 101', '0741231234','adinuta@gmail.com');

INSERT into client VALUES (7,'Frunza','George','1751018874050','Str. Tezaurului Nr. 18', '0741480518','gfrunza@gmail.com');

INSERT into client VALUES (8,'Bulbuc','Elena','2720118366450','Str. Liliacului Nr. 90', '0791560518','bulbuce@gmail.com');

INSERT into client VALUES (9,'Pricope','Diana','2721495374050','Str. Bucuresti Nr. 49', '0712384518','pricopedia@gmail.com');

INSERT into client VALUES (10,'Stoica','Beatrice','2725141774050','Str. Traian Nr. 10', '0761291508','beastoi@gmail.com');

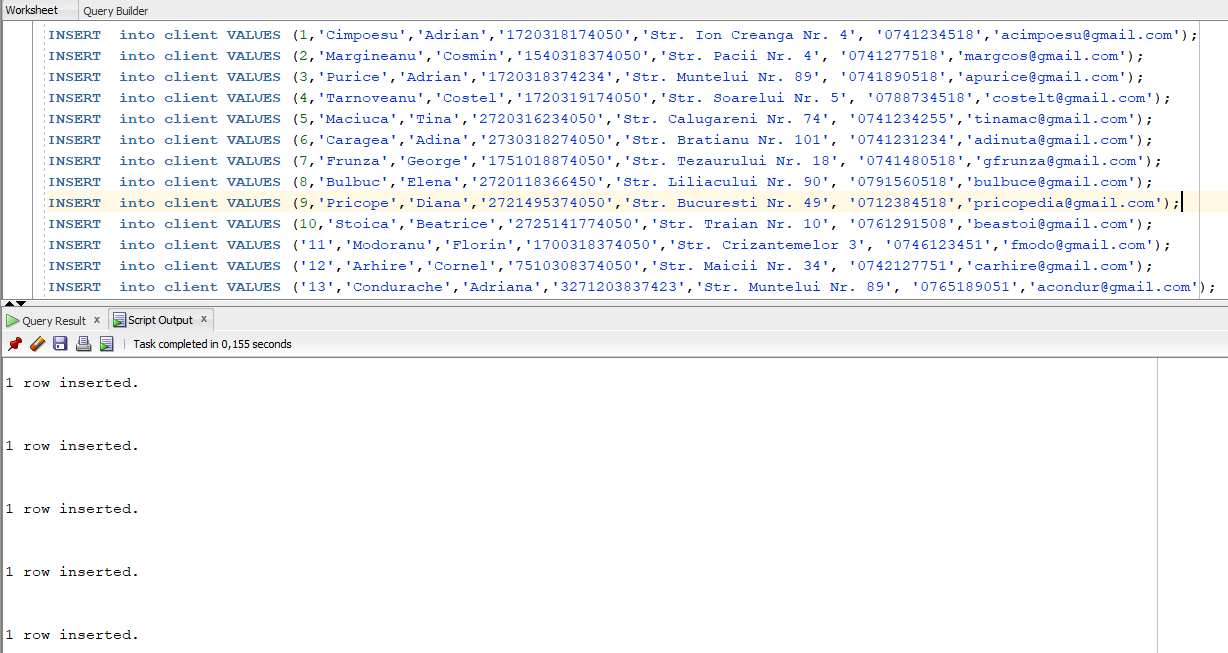
INSERT into client VALUES ('11','Modoranu','Florin','1700318374050','Str. Crizantemelor 3', '0746123451','fmodo@gmail.com');

INSERT into client VALUES ('12','Arhire','Cornel','7510308374050','Str. Maicii Nr. 34', '0742127751','carhire@gmail.com');

INSERT into client VALUES ('13','Condurache','Adriana','3271203837423','Str. Muntelui Nr. 89', '0765189051','acondur@gmail.com');

INSERT into client VALUES ('14','Pirnoveanu','Ioana','2720312174050','Str. Crangului Nr. 5', '0738873451','ipirno@gmail.com');

INSERT into client VALUES ('15','Viorescu','Titina','2720511234050','Str. Calugareni Nr. 4', '0765123425','tinav@gmail.com');



INSERT INTO OFERTA VALUES (1,1,'V','A',1,'Stejar', 150,1999,20000,'Mobilat','3');

INSERT INTO OFERTA VALUES (2,3,'I','C',2,'Pinului', 300,2001,70000,'Mobilat','6');

INSERT INTO OFERTA VALUES (3,2,'V','C',5,'Pacii', 700,2004,100000,'Mobilat','43');

INSERT INTO OFERTA VALUES (4,5,'V','T',6,'Vantului', 500,1995,10000,'Mobilat','89');

INSERT INTO OFERTA VALUES (5,6,'I','A',4,'Teiului', 200,1997,40000,'Mobilat','13');

INSERT INTO OFERTA VALUES (6,3,'V','A',9,'Feroviarilor', 150,2010,50000,'Mobilat','90');

INSERT INTO OFERTA VALUES (7,4,'V','C',5,'Garii', 800,2004,100000,'La rosu','50');

INSERT INTO OFERTA VALUES (8,8,'V','T',6,'Morii', 500,1995,10000,'Mobilat','56');

INSERT INTO OFERTA VALUES (9,7,'I','A',4,'Donici', 100,1997,40000,'Nemobilat','13');

INSERT INTO OFERTA VALUES (10,9,'V','A',9,'Unirii', 150,2010,50000,'Mobilat','100');

INSERT INTO OFERTA VALUES (11,10,'V','A',7,'Pinului', 180,1999,20000,null,'3');

INSERT INTO OFERTA VALUES (12,11,'V','C',2,'Stejar', 300,2001,70000,'Mobilat','6');

INSERT INTO OFERTA VALUES (13,12,'V','C',2,'Pacii', 700,2004,100000,'Mobilat','43');

INSERT INTO OFERTA VALUES (14,13,'V','A',2,'Muncii', 500,1995,10000,'Mobilat','89');

INSERT INTO OFERTA VALUES (15,14,'V','A',7,'Teiului', 200,1997,40000,'Mobilat','13');

INSERT INTO OFERTA VALUES (16,15,'V','A',9,'Tecuci', 150,2010,55000,'Mobilat','90');

INSERT INTO OFERTA VALUES (17,6,'V','C',7,'Florilor', 800,2004,100000,'La rosu','50');

INSERT INTO OFERTA VALUES (18,8,'V','C',2,'Smardan', 500,1995,10000,'Mobilat','56');

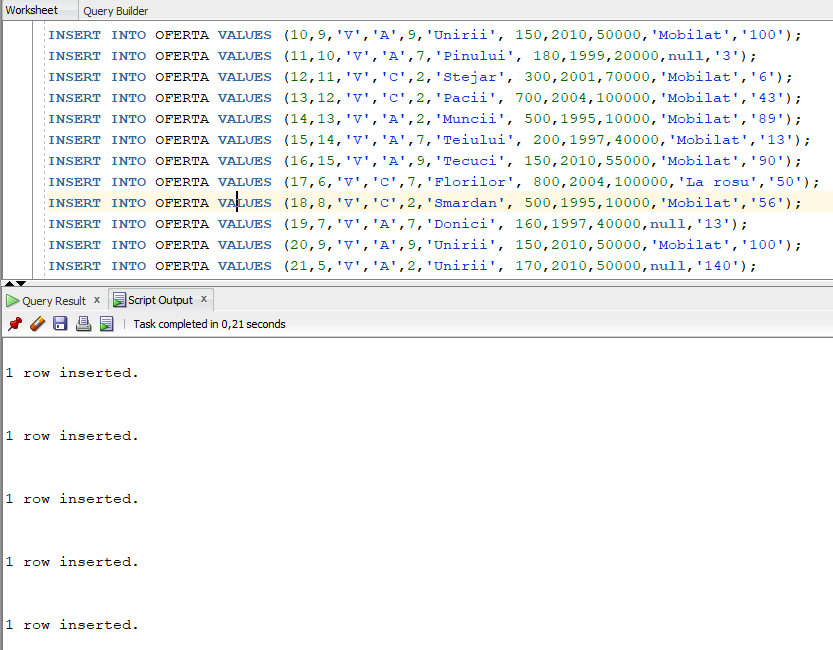
INSERT INTO OFERTA VALUES (19,7,'V','A',7,'Donici', 160,1997,40000,null,'13');

INSERT INTO OFERTA VALUES (20,11,'V','A',9,'Unirii', 150,2010,50000,'Mobilat','100');

INSERT INTO OFERTA VALUES (21,5,'V','A',2,'Unirii', 170,2010,50000,null,'140');

INSERT INTO OFERTA VALUES (22,6,'V','A',5,'Unirii', 170,2011,60000,null,'140');

INSERT INTO OFERTA VALUES (23,11,'V','C',7,'Unirii', 700,2006,200000,null,'23');



INSERT INTO cerere VALUES (1,1,'C','A',1,30000,150,2020);

INSERT INTO cerere VALUES (2,3,'I','C',3,70000,300,2002);

INSERT INTO cerere VALUES (3,8,'C','C',7,200000, 700,2005);

INSERT INTO cerere VALUES (4,10,'C','T',6,200000, 500,2021);

INSERT INTO cerere VALUES (5,2,'I','A',4,50000, 200,2001);

INSERT INTO cerere VALUES (6,7,'C','A',9,10000, 150,2011);

INSERT INTO cerere VALUES (7,4,'C','C',2,100000, 700,2019);

INSERT INTO cerere VALUES (8,5,'C','T',8,300000, 500,2021);

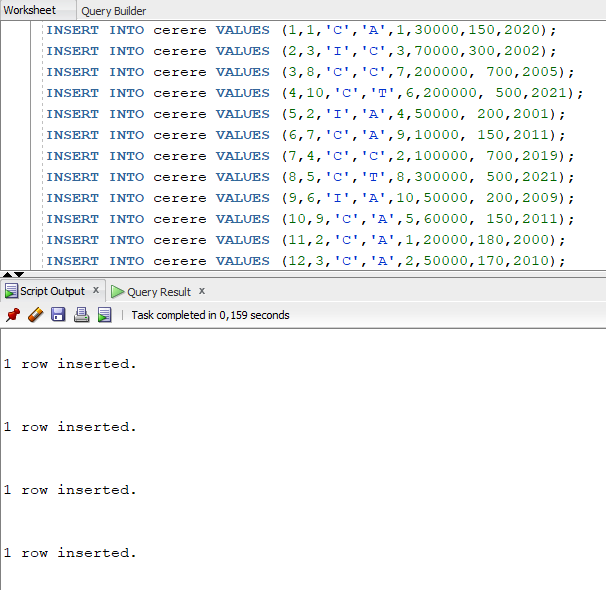
INSERT INTO cerere VALUES (9,6,'I','A',10,50000, 200,2009);

INSERT INTO cerere VALUES (10,9,'C','A',5,60000, 150,2011);

INSERT INTO cerere VALUES (11,2,'C','A',1,20000,180,2000);

INSERT INTO cerere VALUES (12,3,'C','A',2,50000,170,2010);

INSERT INTO cerere VALUES (13,1,'C','A',5,60000,160,2010);



INSERT INTO contract VALUES (1,2,1,SYSDATE,7);

INSERT INTO contract VALUES (2,4,2,SYSDATE,5);

INSERT INTO contract VALUES (3,4,6,SYSDATE,5);

INSERT INTO contract VALUES (4,4,7,SYSDATE,10);

INSERT INTO contract VALUES (5,2,2,SYSDATE,5);

INSERT INTO contract VALUES (6,3,7,SYSDATE,5);

INSERT INTO contract VALUES (7,3,10,SYSDATE,5);

INSERT INTO contract VALUES (8,4,2,SYSDATE,10);

INSERT INTO contract VALUES (9,4,6,SYSDATE,20);

INSERT INTO contract VALUES (10,1,1,SYSDATE,10);

INSERT INTO contract VALUES (11,1,11,SYSDATE,10);

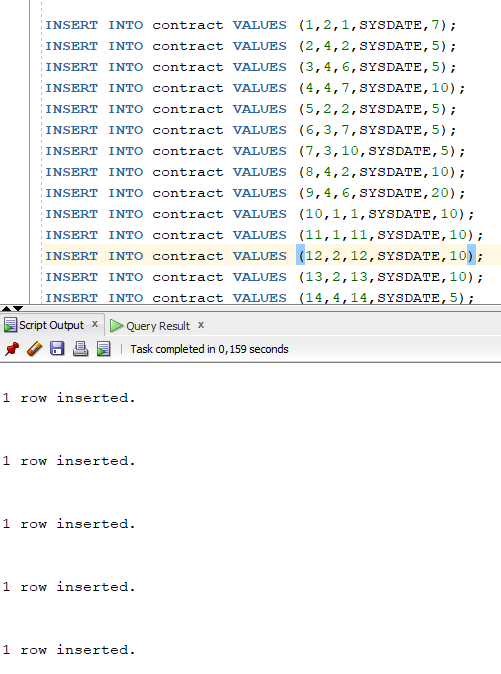
INSERT INTO contract VALUES (12,2,12,SYSDATE,10);

INSERT INTO contract VALUES (13,2,13,SYSDATE,10);

INSERT INTO contract VALUES (14,4,14,SYSDATE,5);

INSERT INTO contract VALUES (15,3,15,SYSDATE,10);

INSERT INTO contract VALUES (16,3,9,SYSDATE,10);



INSERT INTO TRANZACTIE VALUES(1,1,1,1,SYSDATE,10000);

INSERT INTO TRANZACTIE VALUES(2,2,2,2,SYSDATE,20000);

INSERT INTO TRANZACTIE VALUES(3,3,3,5,SYSDATE,300000);

INSERT INTO TRANZACTIE VALUES(4,3,4,4,SYSDATE,40000);

INSERT INTO TRANZACTIE VALUES(5,3,5,3,SYSDATE,50000);

INSERT INTO TRANZACTIE VALUES(6,4,6,2,SYSDATE,90000);

INSERT INTO TRANZACTIE VALUES(7,7,1,6,SYSDATE,10000);

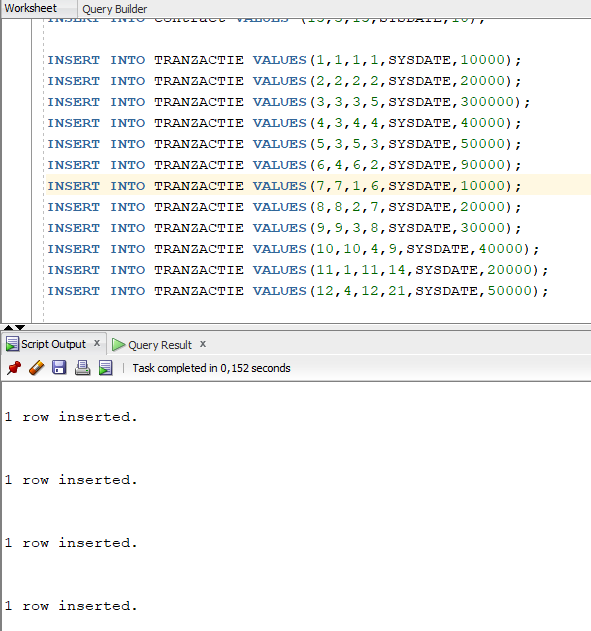
INSERT INTO TRANZACTIE VALUES(8,8,2,7,SYSDATE,20000);

INSERT INTO TRANZACTIE VALUES(9,9,3,8,SYSDATE,30000);

INSERT INTO TRANZACTIE VALUES(10,10,4,9,SYSDATE,40000);

INSERT INTO TRANZACTIE VALUES(11,1,11,14,SYSDATE,20000);

INSERT INTO TRANZACTIE VALUES(12,4,12,21,SYSDATE,50000);



# 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.

*Cerința***: Să se afișeze ofertele existente pe fiecare județ, respectiv localitate.**

CREATE OR REPLACE PROCEDURE cerinta6

IS

TYPE T\_id\_loc IS TABLE OF localitate.id\_localitate%TYPE INDEX BY PLS\_INTEGER;

TYPE T\_den\_loc IS TABLE OF localitate.denumire%TYPE;

TYPE T\_den\_jud IS VARRAY(50) OF judet.denumire%TYPE;

--**record pentru datele unei oferte ++ vom face join pt numele clientului(ofertantului)**

TYPE T\_of\_record IS RECORD ( loc NUMBER(10), ofertant VARCHAR(30),strada VARCHAR(30),numar VARCHAR(10), suprafata NUMBER(10), an\_constructie NUMBER(4), pret NUMBER(15), op CHAR(1), tip\_propr VARCHAR(15), descr VARCHAR(30));

TYPE T\_of IS TABLE OF T\_of\_record INDEX BY PLS\_INTEGER;

v\_id\_loc T\_id\_loc;

v\_den\_loc T\_den\_loc;

v\_den\_jud T\_den\_jud;

v\_of T\_of;

BEGIN

SELECT id\_localitate BULK COLLECT INTO v\_id\_loc FROM localitate ORDER BY id\_localitate;

DBMS\_OUTPUT.PUT('Tablou indexat: ');

FOR i IN v\_id\_loc.FIRST..v\_id\_loc.LAST LOOP

DBMS\_OUTPUT.PUT(v\_id\_loc(i) || ' ');

END LOOP;

DBMS\_OUTPUT.NEW\_LINE;

SELECT denumire BULK COLLECT INTO v\_den\_loc FROM localitate ORDER BY id\_localitate;

DBMS\_OUTPUT.PUT('Tablou imbricat: ');

FOR i IN v\_den\_loc.FIRST..v\_den\_loc.LAST LOOP

DBMS\_OUTPUT.PUT(v\_den\_loc(i) || ' ');

END LOOP;

DBMS\_OUTPUT.NEW\_LINE;

SELECT j.denumire BULK COLLECT INTO v\_den\_jud

FROM localitate l join judet j on l.id\_judet=j.id\_judet

order by l.id\_localitate;

DBMS\_OUTPUT.PUT('Vector: ');

FOR i IN v\_den\_jud.FIRST..v\_den\_jud.LAST LOOP

DBMS\_OUTPUT.PUT(v\_den\_jud(i) || ' ');

END LOOP;

DBMS\_OUTPUT.NEW\_LINE;

DBMS\_OUTPUT.NEW\_LINE;

SELECT O.ID\_LOCALITATE LOC, INITCAP(TRIM(C.NUME)) || ' ' ||

INITCAP(TRIM(C.PRENUME)) OFERTANT, STRADA, NUMAR, SUPRAFATA, AN\_CONSTRUCTIE, PRET, ID\_OPERATIE OP, DECODE(ID\_PROPRIETATE,'C','CASA','A','APARTAMENT','T','TEREN') TIP\_PROPR, NVL(O.DESCRIERE,'') DESCR

BULK COLLECT INTO v\_of

FROM OFERTA O

JOIN CLIENT C ON O.ID\_CLIENT=C.ID\_CLIENT

WHERE NOT EXISTS ( SELECT \* FROM TRANZACTIE T

WHERE T.ID\_OFERTA = O.ID\_OFERTA

) --preluam doar ofertele disponibile, nu si cele tranzactionate

ORDER BY O.ID\_LOCALITATE, O.ID\_OPERATIE, O.ID\_PROPRIETATE, O.STRADA;

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

DBMS\_OUTPUT.PUT\_LINE(' JUDET LOCALITATE OFERTANT STRADA NUMAR SUPRAFATA CONSTRUIT OP TIP\_IMOBIL DESCR');

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

FOR i IN v\_of.FIRST..v\_of.LAST LOOP

DBMS\_OUTPUT.PUT\_LINE(RPAD(v\_den\_jud(v\_of(i).loc),10,' ') || ' ' || RPAD(v\_den\_loc(v\_of(i).loc),10,' ') || ' '|| RPAD(v\_of(i).ofertant,20,' ') || ' '|| v\_of(i).strada || ' '|| v\_of(i).numar || ' '|| v\_of(i).suprafata || ' '

|| v\_of(i).an\_constructie || ' '|| v\_of(i).pret || ' '|| v\_of(i).op || ' '|| v\_of(i).tip\_propr || ' '|| v\_of(i).descr || ' ');

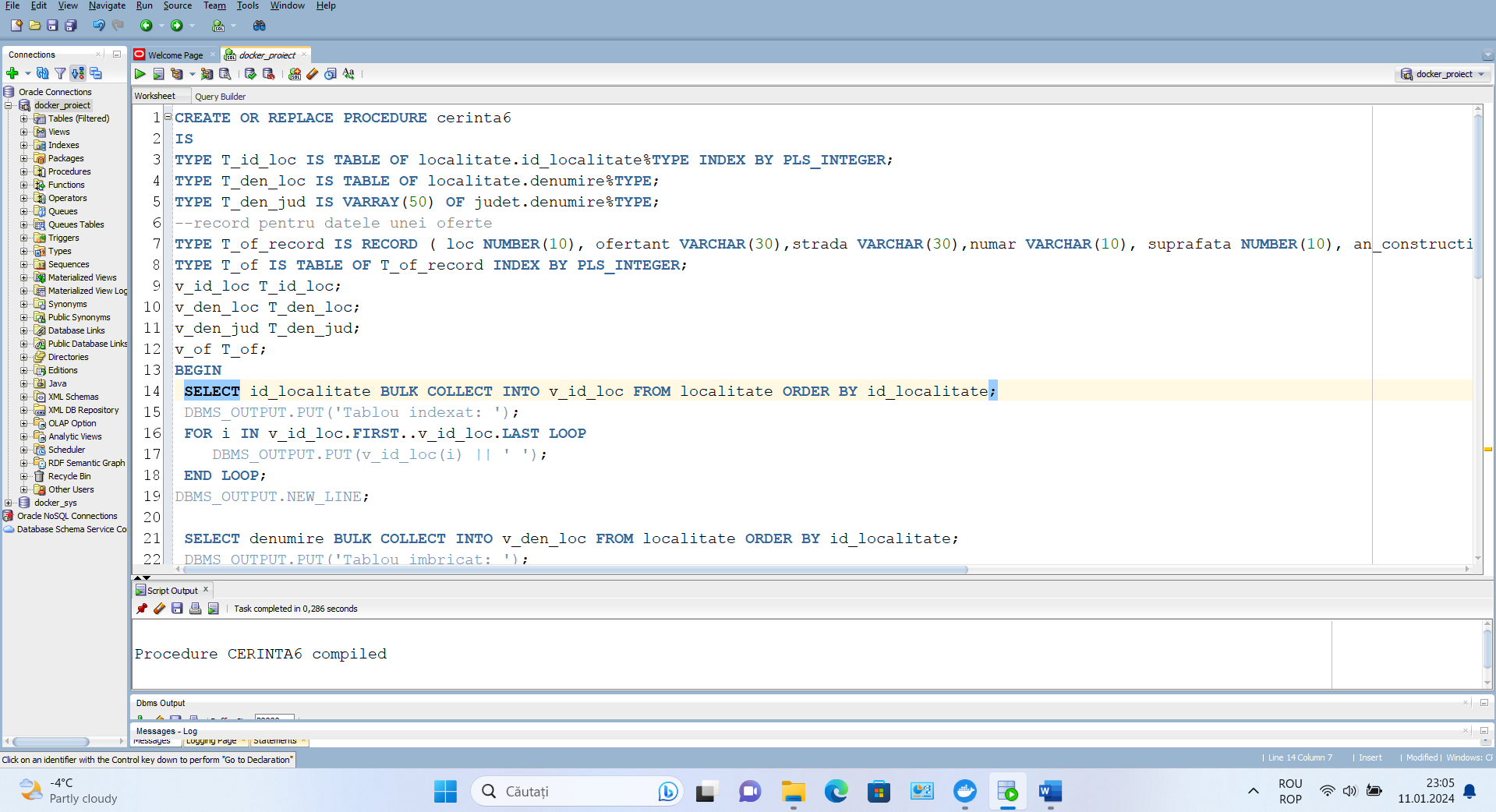
END LOOP;

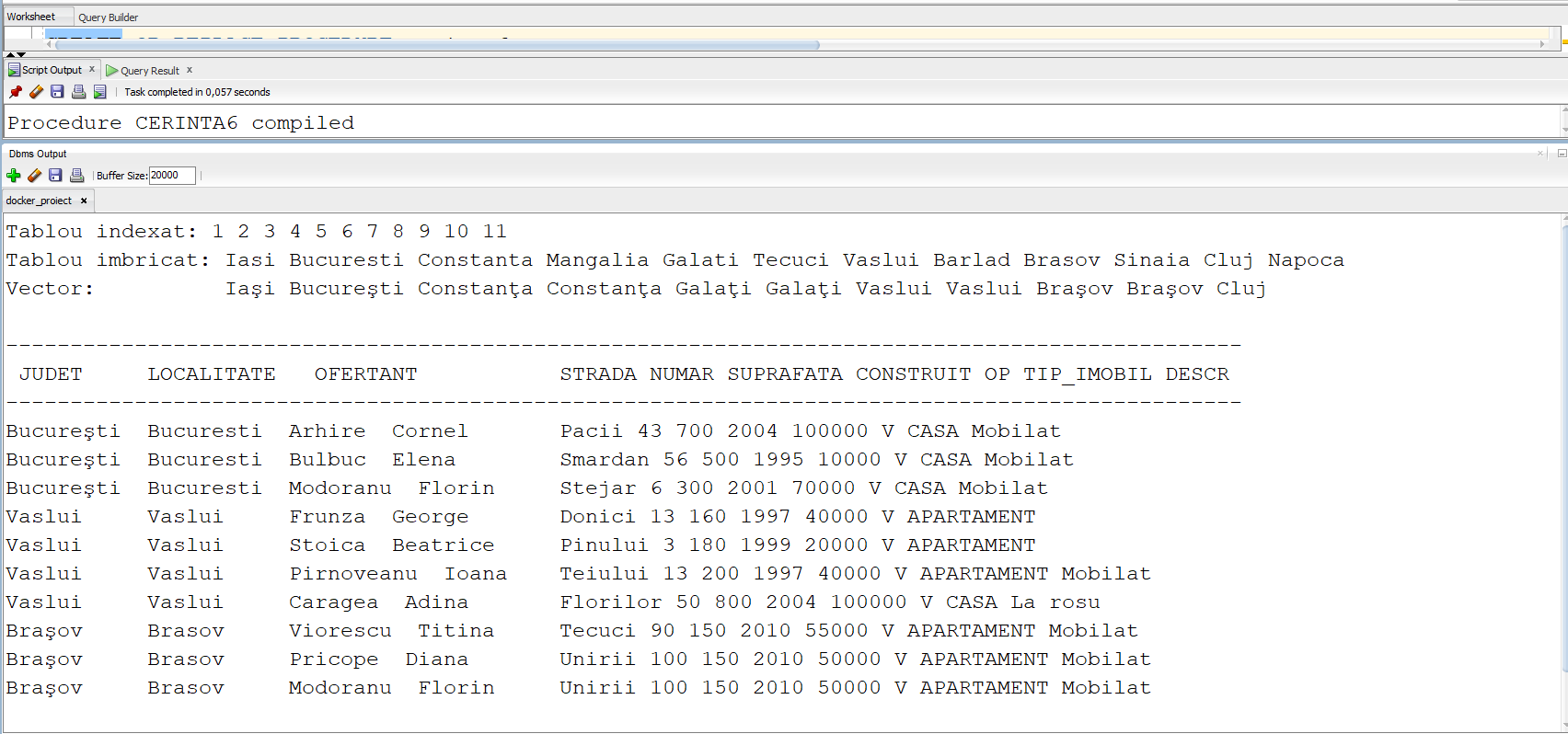
DBMS\_OUTPUT.NEW\_LINE;

END cerinta6;

/

EXECUTE cerinta6;





# 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.

*Cerința*: **Să se afișeze localitățile în care există mai puține cereri decât oferte pentru apartamente.**

CREATE OR REPLACE PROCEDURE cerinta7 IS

CURSOR c\_of (p\_id\_localitate localitate.id\_localitate%type) IS

(SELECT o.id\_localitate, COUNT(\*) AS nr\_of --**id localitate si numar de oferte/localitate**

FROM oferta o WHERE o.id\_localitate=p\_id\_localitate and o.id\_proprietate='A'

GROUP BY o.id\_localitate);

CURSOR c\_cer (p\_id\_localitate localitate.id\_localitate%type) IS

(SELECT c.id\_localitate, COUNT(\*) AS nr\_cer

FROM cerere c WHERE c.id\_localitate=p\_id\_localitate and c.id\_proprietate='A'

GROUP BY c.id\_localitate);

v\_id\_loc NUMBER(4);

v\_of NUMBER(4);

v\_cer NUMBER(4);

BEGIN

FOR i IN (SELECT id\_localitate, denumire FROM localitate) LOOP --**ciclu cursor cu subcereri**

OPEN c\_of(i.id\_localitate);

FETCH c\_of INTO v\_id\_loc, v\_of;

OPEN c\_cer(i.id\_localitate);

FETCH c\_cer INTO v\_id\_loc, v\_cer;

IF (v\_cer < v\_of) THEN

DBMS\_OUTPUT.PUT\_LINE(i.denumire || ' cereri ' || v\_cer || ' oferte ' || v\_of);

END IF;

CLOSE c\_of;

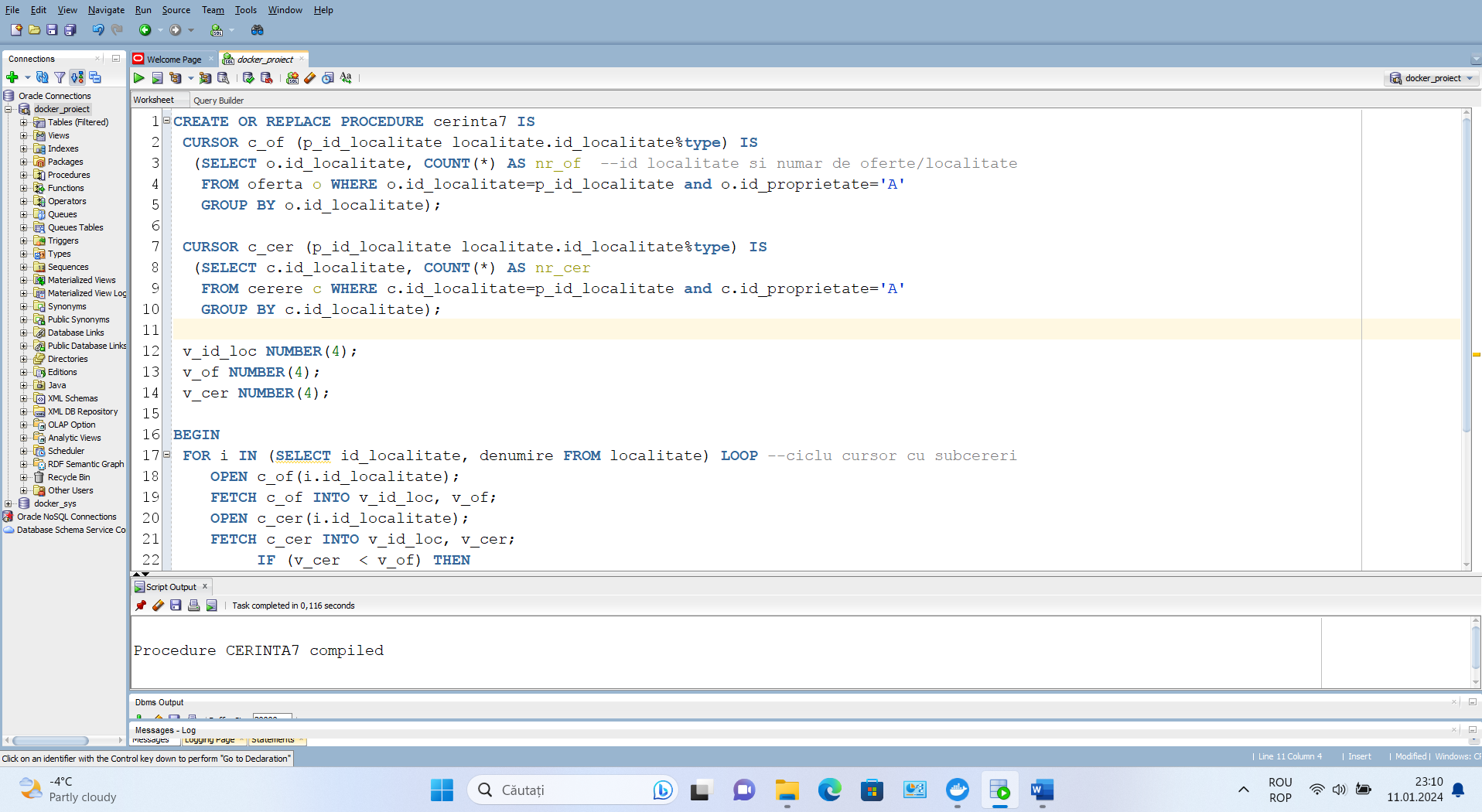
CLOSE c\_cer;

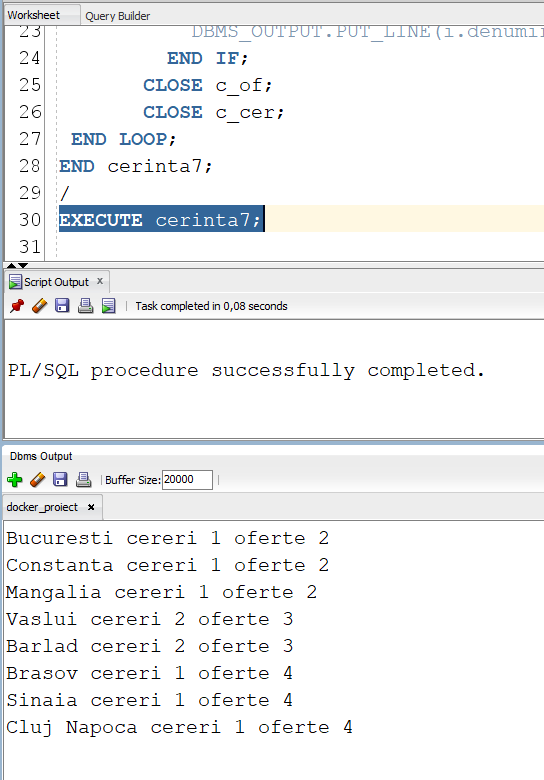
END LOOP;

END cerinta7;

/

EXECUTE cerinta7;





# 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.

*Cerința*: **Definiți un subprogram prin care să obțineți pentru un angajat al cărui prenume este specificat, numărul de contracte încheiate pe o perioadă specificată cu clienți care au cel puțin o cerere sau o ofertă introdusă.**

CREATE OR REPLACE FUNCTION f8 (v\_prenume angajat.prenume%TYPE DEFAULT 'Vasile', data\_init contract.data%TYPE DEFAULT TO\_DATE('01/01/2023','DD/MM/YYYY'), data\_fin contract.data%TYPE DEFAULT TO\_DATE('31/12/2024','DD/MM/YYYY'))

RETURN NUMBER IS

v\_nr\_contr NUMBER(4);

v\_nr\_ang NUMBER(2);

exception1 EXCEPTION;

exception2 EXCEPTION;

BEGIN

--**numaram cati angajati mai au prenumele respectiv pt a arunca erorile dupa caz**

SELECT COUNT(id\_angajat)

INTO v\_nr\_ang

FROM angajat

WHERE UPPER(prenume) = UPPER(v\_prenume);

IF (v\_nr\_ang = 0) THEN RAISE exception1;

END IF;

IF (v\_nr\_ang > 1) THEN RAISE exception2;

END IF;

--**in nc s-a calculat numarul de contracte pentru fiecare angajat care are minim o cerere sau o oferta**

SELECT nc.nr\_contr INTO v\_nr\_contr FROM angajat a

LEFT JOIN (SELECT c.id\_angajat, COUNT(\*) as nr\_contr FROM contract c

WHERE c.data BETWEEN data\_init AND data\_fin **-- verificam data**

AND ( (c.id\_client IN (SELECT distinct id\_client FROM cerere)) OR

(c.id\_client IN (SELECT distinct id\_client FROM oferta)) ) **--verificam sa aiba minim o cerere/ oferta**

GROUP BY c.id\_angajat) nc

ON a.id\_angajat = nc.id\_angajat

WHERE UPPER(a.prenume) = UPPER(v\_prenume);

RETURN NVL(v\_nr\_contr,0);

EXCEPTION

WHEN exception1 THEN

BEGIN

DBMS\_OUTPUT.PUT\_LINE('EROARE: Nu exista niciun angajat cu prenumele dat!');

RETURN -1;

END;

WHEN exception2 THEN

BEGIN

DBMS\_OUTPUT.PUT\_LINE('EROARE: Exista mai multi angajati cu prenumele dat!');

RETURN -1;

END;

WHEN OTHERS THEN RAISE\_APPLICATION\_ERROR(-20002,'Alta eroare!');

END f8;

/

BEGIN

DBMS\_OUTPUT.PUT\_LINE('Numarul de contracte incheiate este '|| f8);

END;

/

BEGIN

DBMS\_OUTPUT.PUT\_LINE('Numarul de contracte incheiate este '|| f8('MMM'));

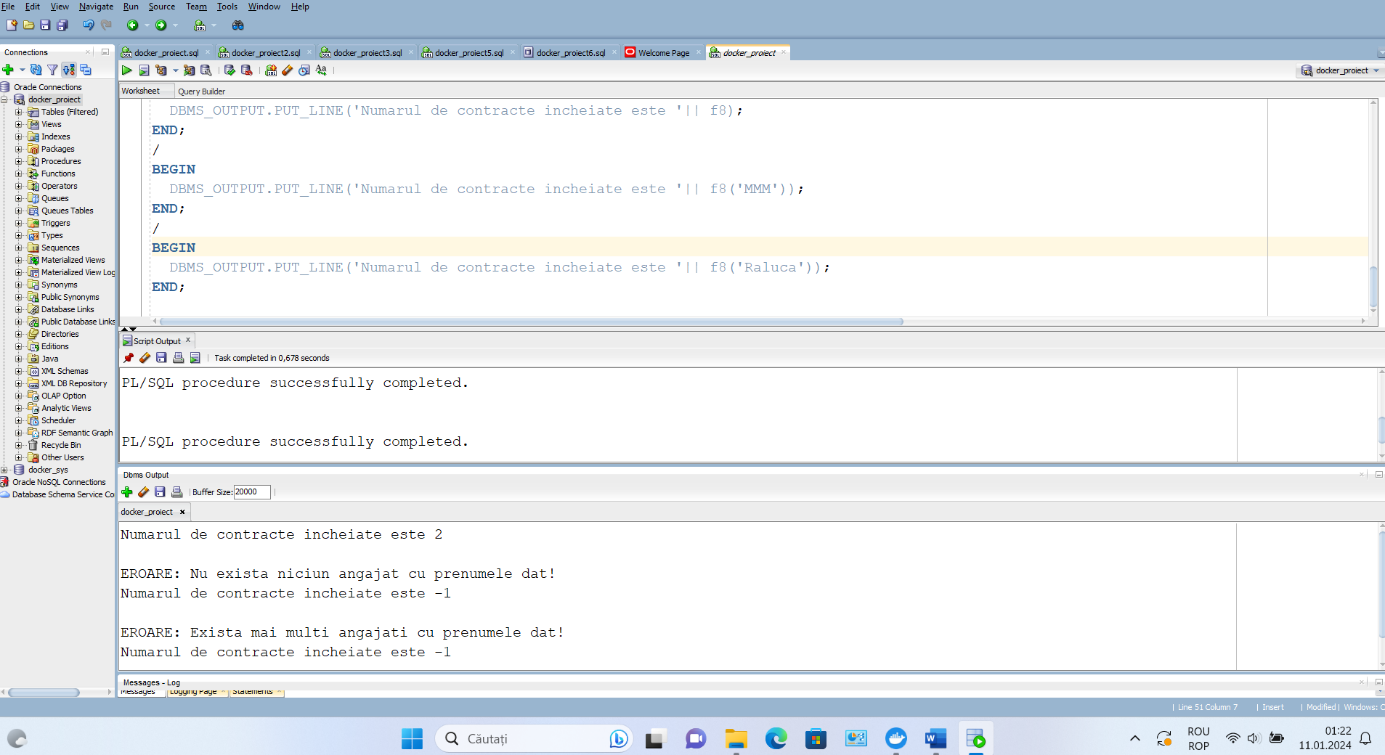
END;

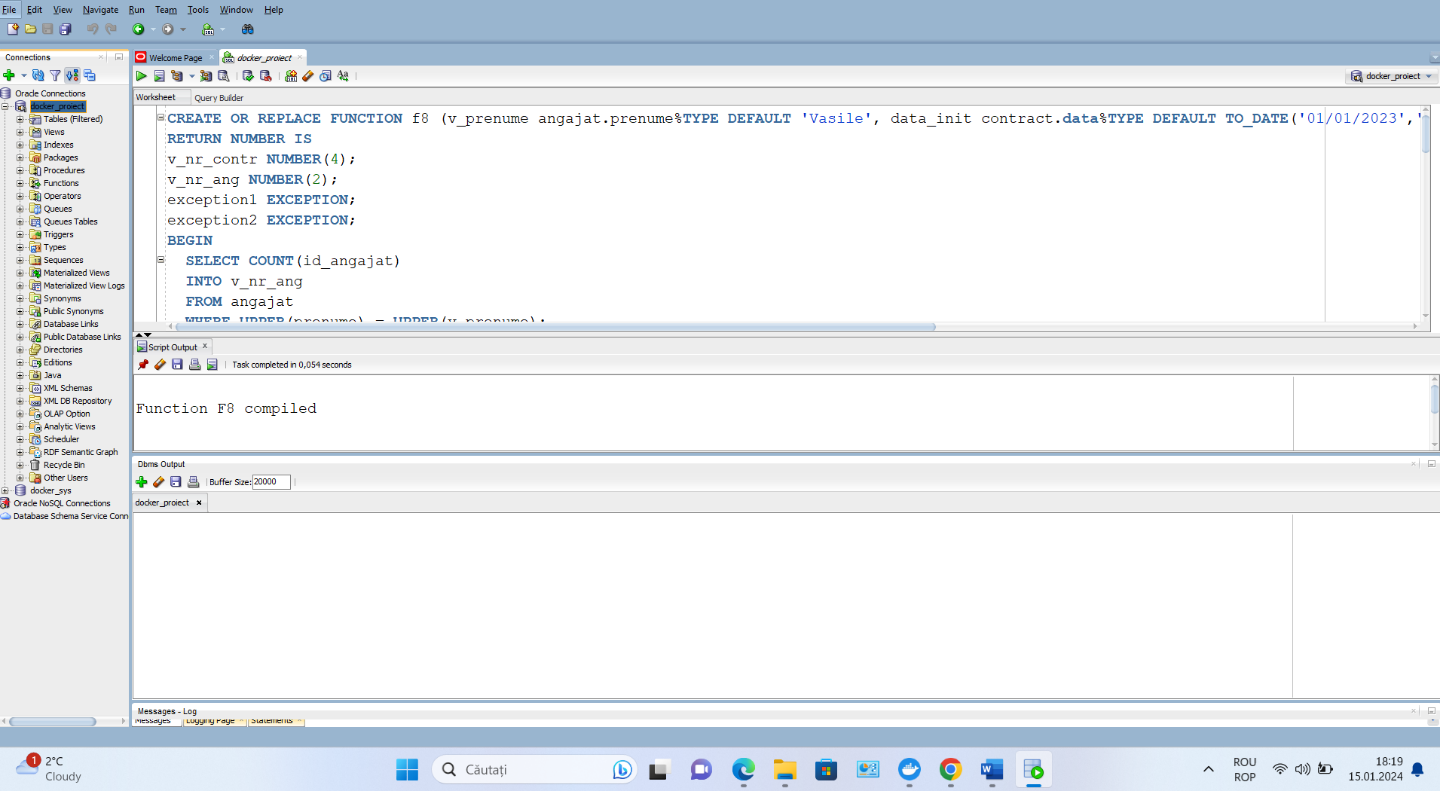
/

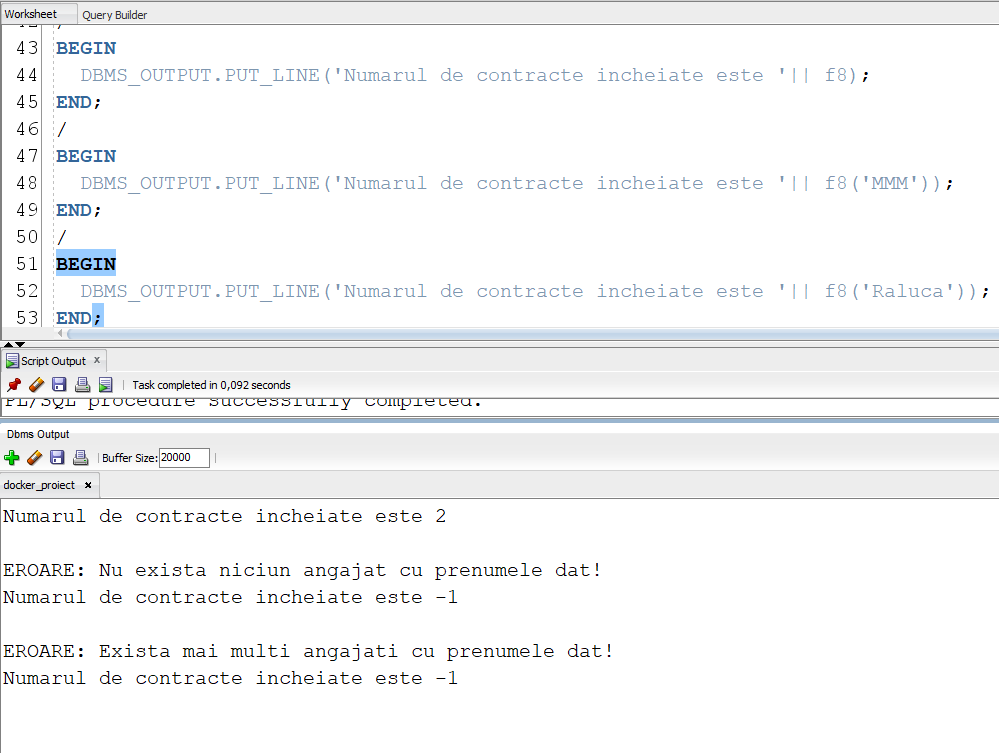
BEGIN

DBMS\_OUTPUT.PUT\_LINE('Numarul de contracte incheiate este '|| f8('Raluca'));

END;







# 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.

*Cerința*: **Pentru o cerere dată să se afișeze oferta potrivită (să corespundă caracteristicile: localitate, tip proprietate, suprafață, buget, an construcție etc.) care nu a fost deja tranzacționată. Tratați cazurile în care există mai multe oferte care corespund, respectiv când nicio ofertă nu este corespunzătoare cererii. În cazul favorabil afișați id-ul ofertei, numele clientului care a făcut-o și localitatea unde se află.**

CREATE OR REPLACE PROCEDURE

OF\_PT\_CER(v\_id\_cerere IN cerere.id\_cerere%TYPE,

v\_id\_oferta OUT oferta.id\_oferta%type) IS

v\_id\_localitate localitate.id\_localitate%TYPE;

v\_id\_proprietate proprietate.id\_proprietate%TYPE;

v\_buget cerere.buget%TYPE;

v\_supraf\_min cerere.suprafata\_minima%TYPE;

v\_an\_constructie cerere.an\_constructie%TYPE;

v\_id\_operatie operatie.id\_operatie%TYPE;

v\_nr NUMBER(3);

v\_nume client.nume%TYPE;

v\_prenume client.prenume%TYPE;

v\_denumire localitate.denumire%TYPE;

BEGIN

--**verificam daca exista cerere cu id-ul dat**

SELECT COUNT(\*) INTO v\_nr FROM cerere WHERE id\_cerere=v\_id\_cerere;

IF (v\_nr=0) THEN

DBMS\_OUTPUT.PUT\_LINE('Nu exista cerere cu id-ul dat!');

RETURN;

END IF;

--**pun toate info. despre cererea data in variabile**

SELECT id\_localitate, id\_proprietate,buget, suprafata\_minima, an\_constructie, id\_operatie

INTO v\_id\_localitate, v\_id\_proprietate, v\_buget, v\_supraf\_min, v\_an\_constructie, v\_id\_operatie

FROM cerere

WHERE id\_cerere=v\_id\_cerere;

SELECT id\_oferta,c.nume, c.prenume,l.denumire INTO v\_id\_oferta, v\_nume, v\_prenume, v\_denumire

FROM **oferta** o

JOIN **client** c on o.id\_client=c.id\_client

JOIN **localitate** l on o.id\_localitate=l.id\_localitate

WHERE o.id\_localitate = v\_id\_localitate AND o.id\_proprietate=v\_id\_proprietate AND

o.pret<=v\_buget AND o.suprafata>=v\_supraf\_min AND ABS(o.an\_constructie-v\_an\_constructie)<=10

AND ((v\_id\_operatie='C' AND o.id\_operatie='V') OR (v\_id\_operatie='I' AND o.id\_operatie='I')) AND o.id\_oferta NOT IN (SELECT id\_oferta

FROM **tranzactie**)

AND o.id\_client IN (SELECT id\_client

FROM **contract**);

DBMS\_OUTPUT.PUT\_LINE ('Oferta ' || v\_id\_oferta || ' facuta de ' || v\_nume || ' ' || v\_prenume || ' in localitatea ' || v\_denumire);

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

RAISE\_APPLICATION\_ERROR(-20000, 'Nu exista oferta care sa respecte cererea!');

WHEN TOO\_MANY\_ROWS THEN

RAISE\_APPLICATION\_ERROR(-20001,'Exista mai multe oferte care respecta cererea!');

WHEN OTHERS THEN

RAISE\_APPLICATION\_ERROR(-20002,'Alta eroare!');

END OF\_PT\_CER;

/

--APELARE

DECLARE

v\_oferta oferta.id\_oferta%type;

BEGIN

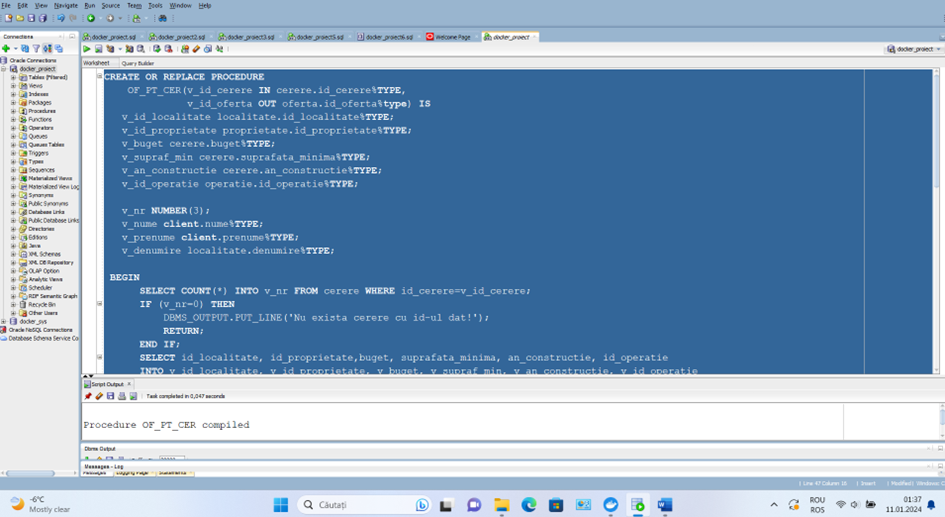
--OF\_PT\_CER(10,v\_oferta); -- o singura oferta potrivita

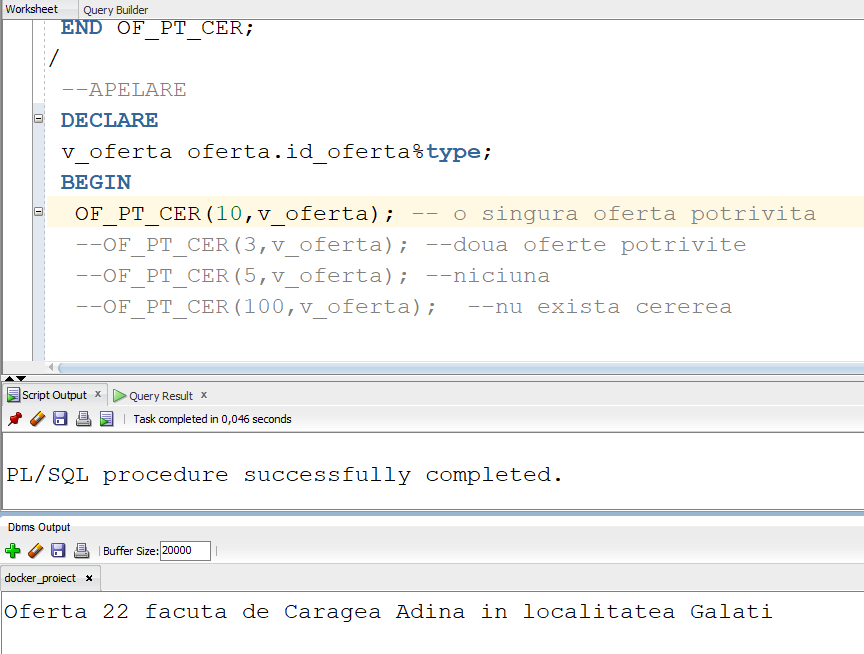
OF\_PT\_CER(3,v\_oferta); --doua oferte potrivite

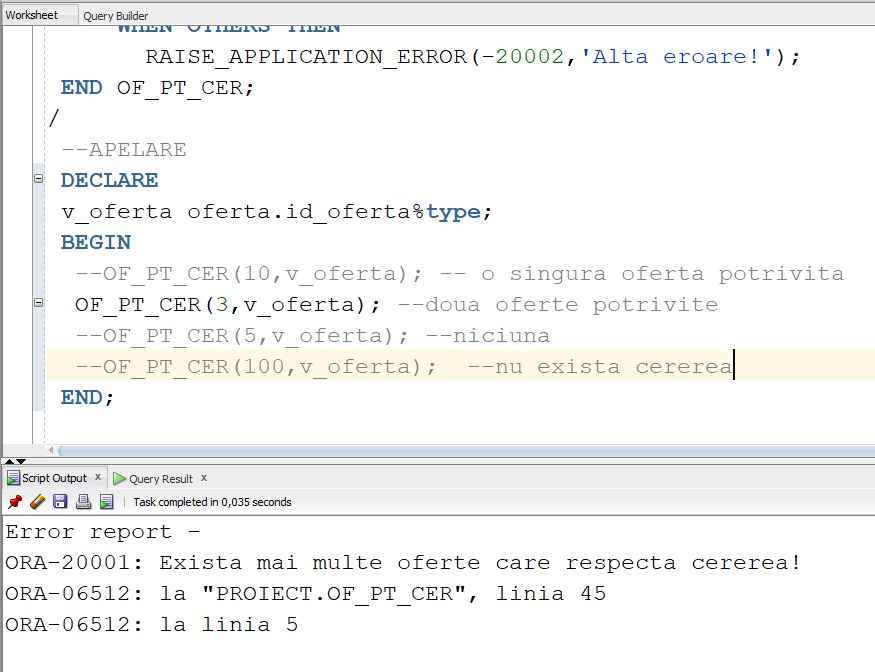
--OF\_PT\_CER(5,v\_oferta); --niciuna

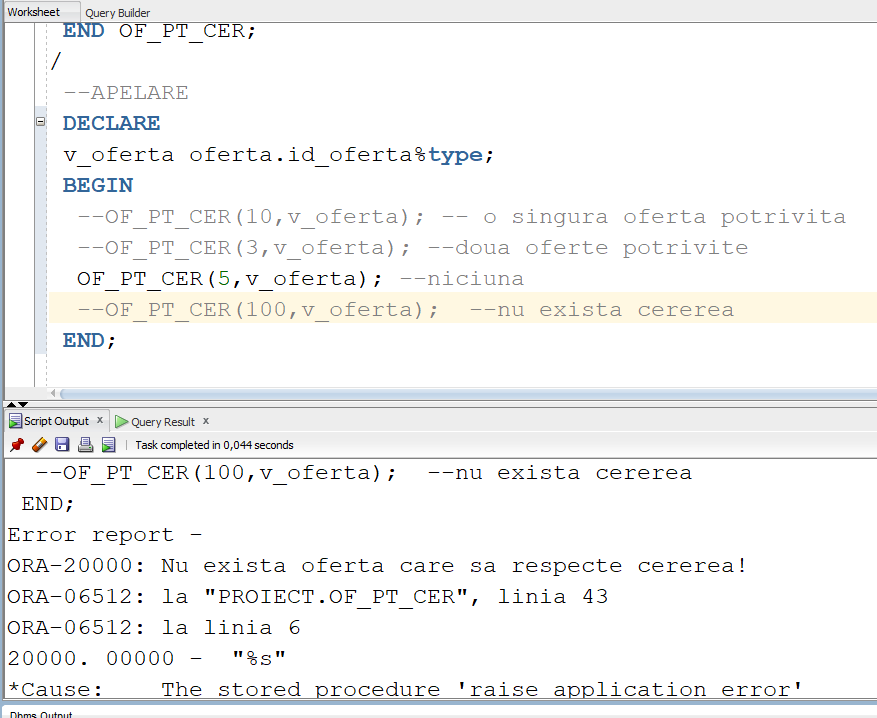
--OF\_PT\_CER(100,v\_oferta); --nu exista cererea

END;









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

**Trig\_10 blochează tabela PROPRIETATE la actualizări:**

CREATE OR REPLACE TRIGGER trig\_10

BEFORE INSERT OR UPDATE OR DELETE ON proprietate

BEGIN

CASE

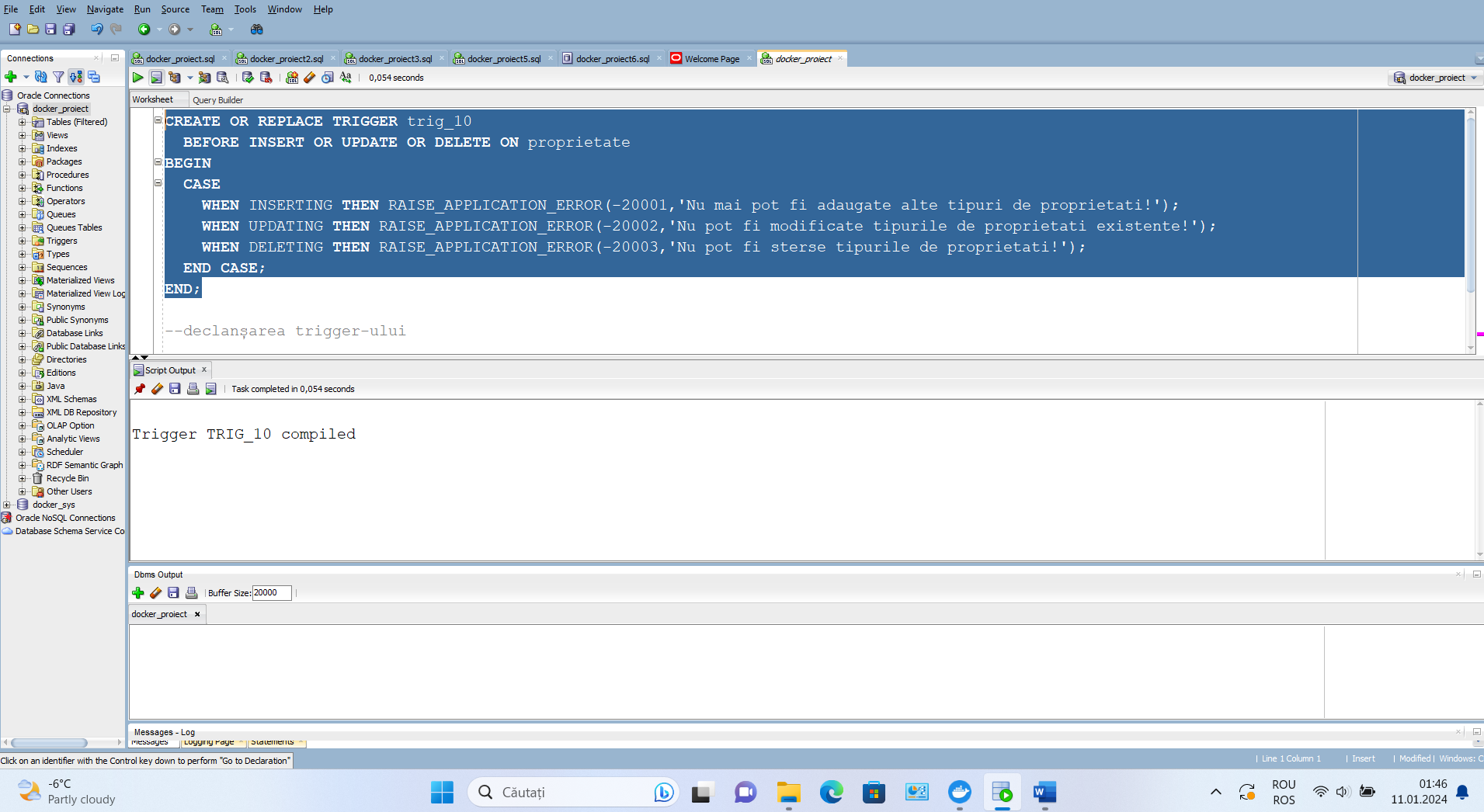
WHEN INSERTING THEN RAISE\_APPLICATION\_ERROR(-20001,'Nu mai pot fi adaugate alte tipuri de proprietati!');

WHEN UPDATING THEN RAISE\_APPLICATION\_ERROR(-20002,'Nu pot fi modificate tipurile de proprietati existente!');

WHEN DELETING THEN RAISE\_APPLICATION\_ERROR(-20003,'Nu pot fi sterse tipurile de proprietati!');

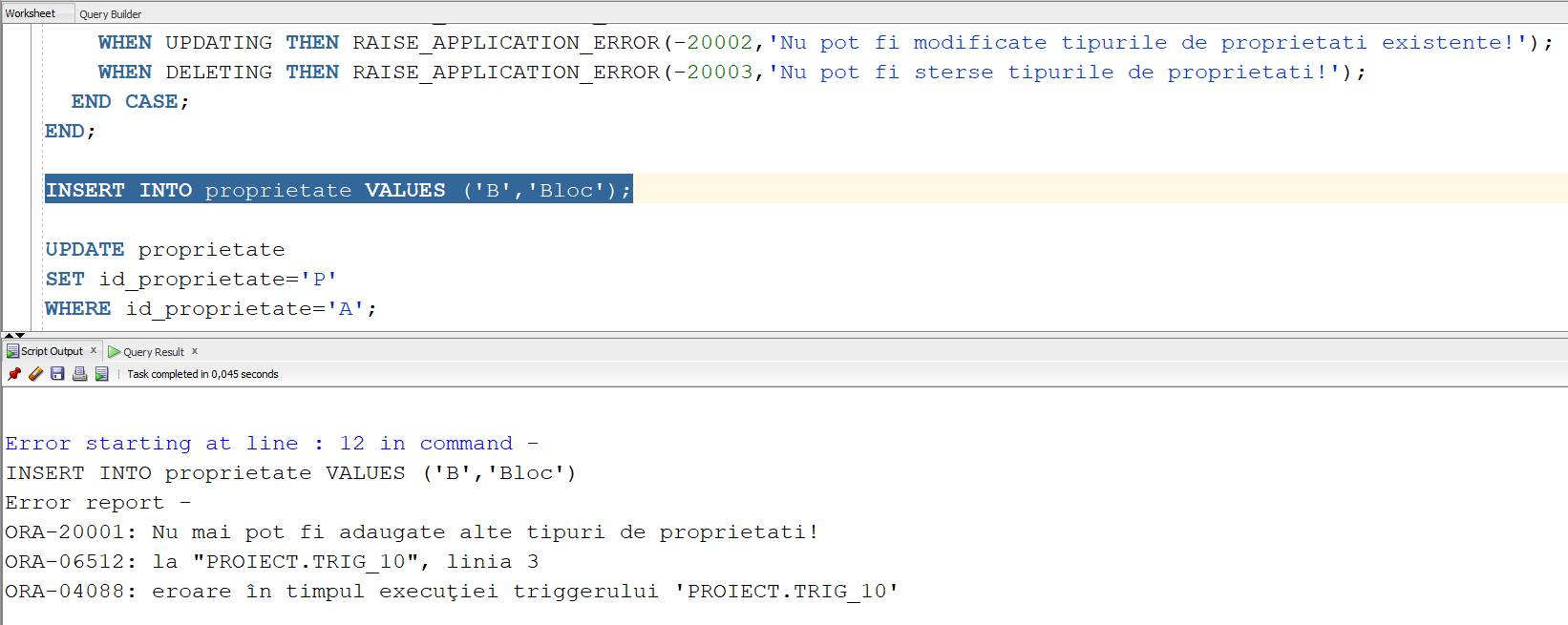
END CASE;

END;



--declanșarea trigger-ului

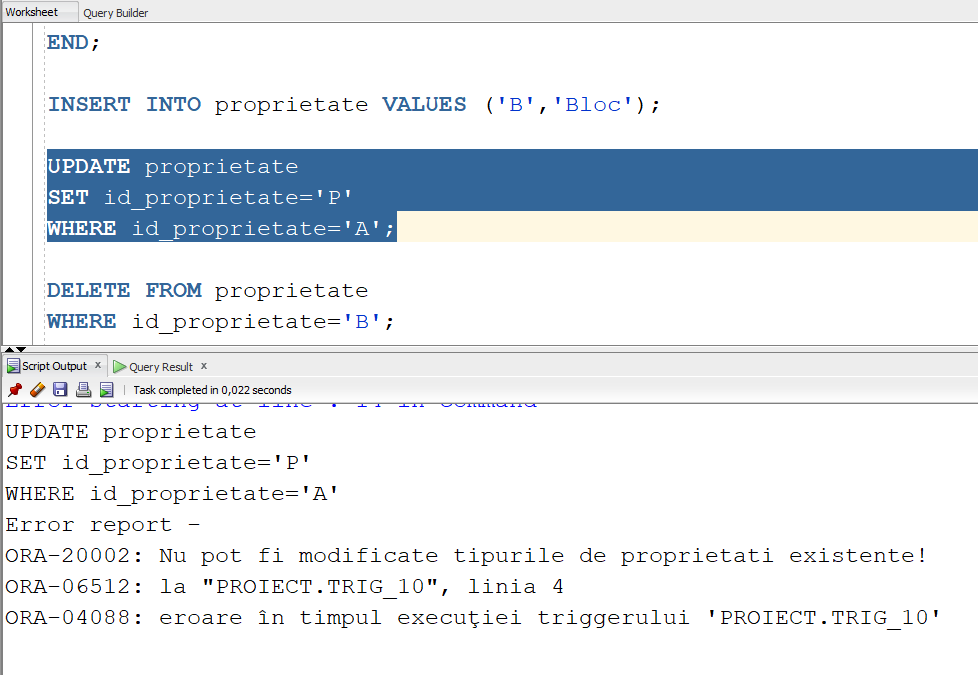
INSERT INTO proprietate VALUES ('B','Bloc');



UPDATE proprietate

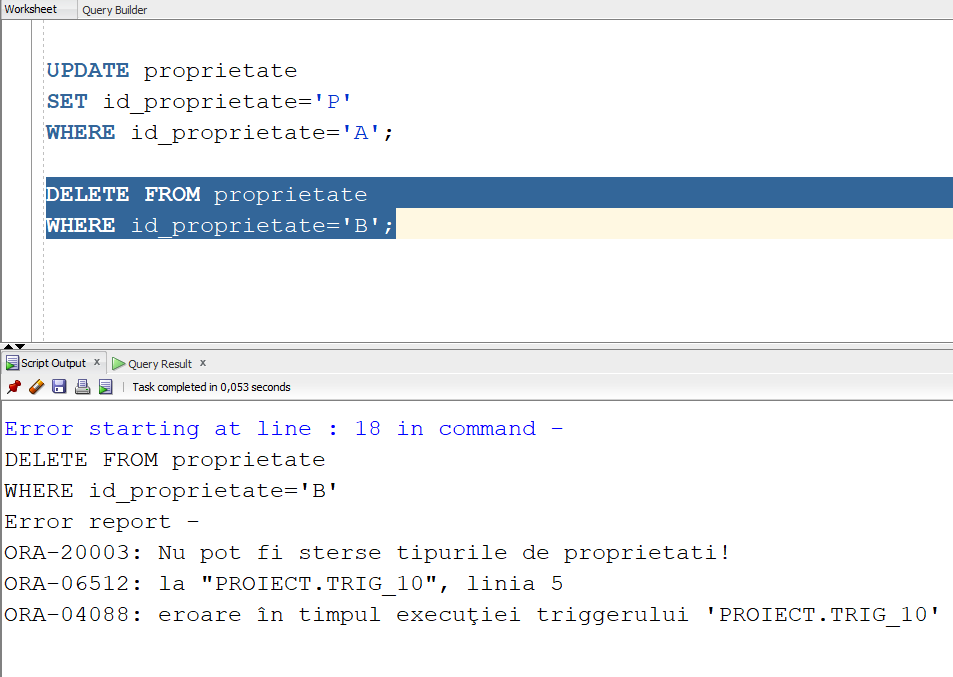
SET id\_proprietate='P'

WHERE id\_proprietate='A';



DELETE FROM proprietate

WHERE id\_proprietate='B';



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

**Trig\_11 impune ca înaintea realizării unei tranzacții, ambele părți să fi încheiat contract cu agenția:**

CREATE OR REPLACE TRIGGER trig\_11

BEFORE INSERT ON tranzactie

FOR EACH ROW

DECLARE

v\_nr\_contr\_cer NUMBER(2) := 0;

v\_nr\_contr\_of NUMBER(2) := 0;

BEGIN

SELECT COUNT(\*) -- **nr de contracte** **incheiate cu clientul care a facut cererea**

INTO v\_nr\_contr\_cer

FROM contract

WHERE id\_client IN (SELECT id\_client FROM cerere WHERE id\_cerere=:NEW.id\_cerere);

SELECT COUNT(\*)

INTO v\_nr\_contr\_of

FROM contract

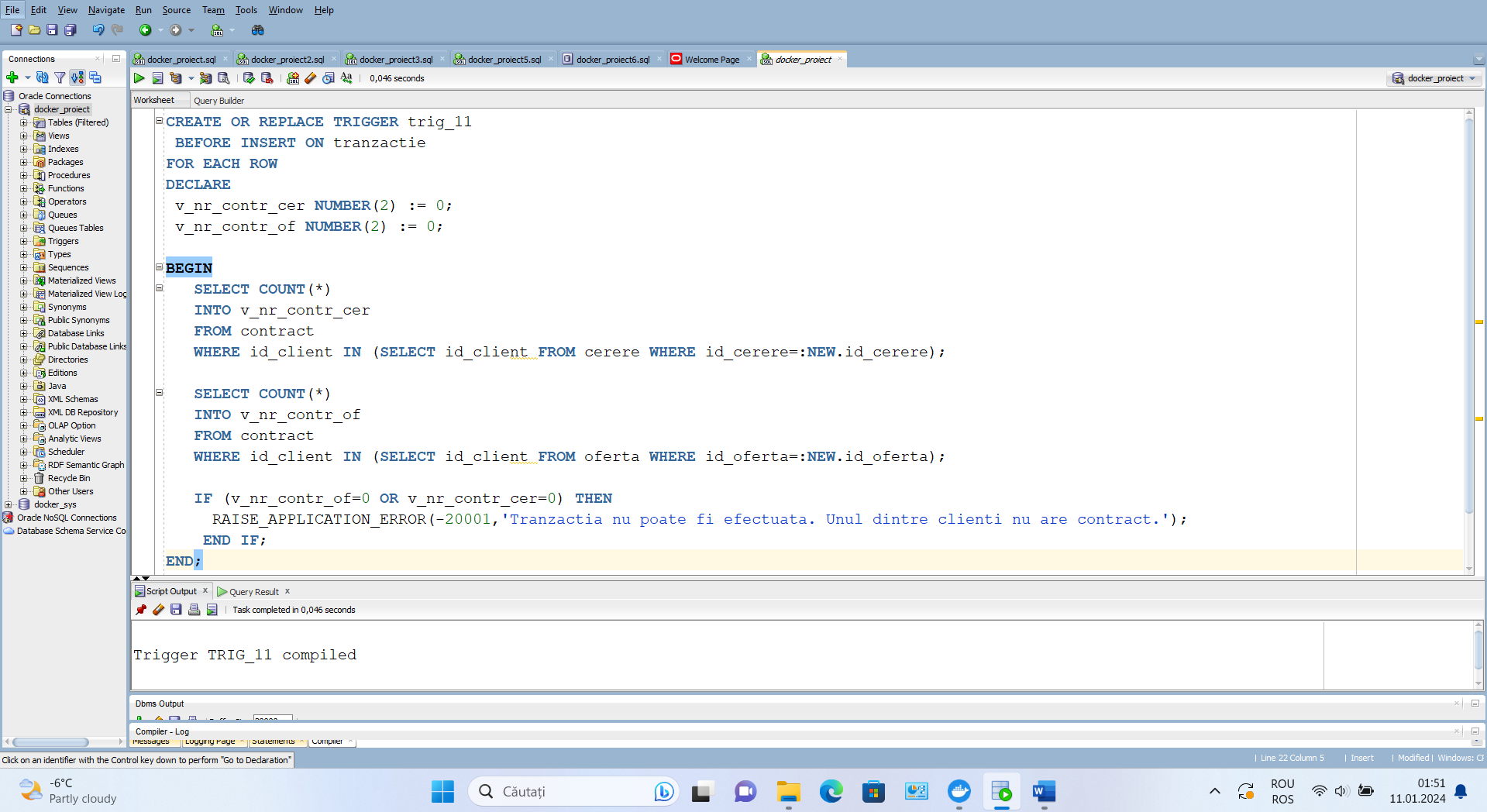
WHERE id\_client IN (SELECT id\_client FROM oferta WHERE id\_oferta=:NEW.id\_oferta);

IF (v\_nr\_contr\_of=0 OR v\_nr\_contr\_cer=0) THEN

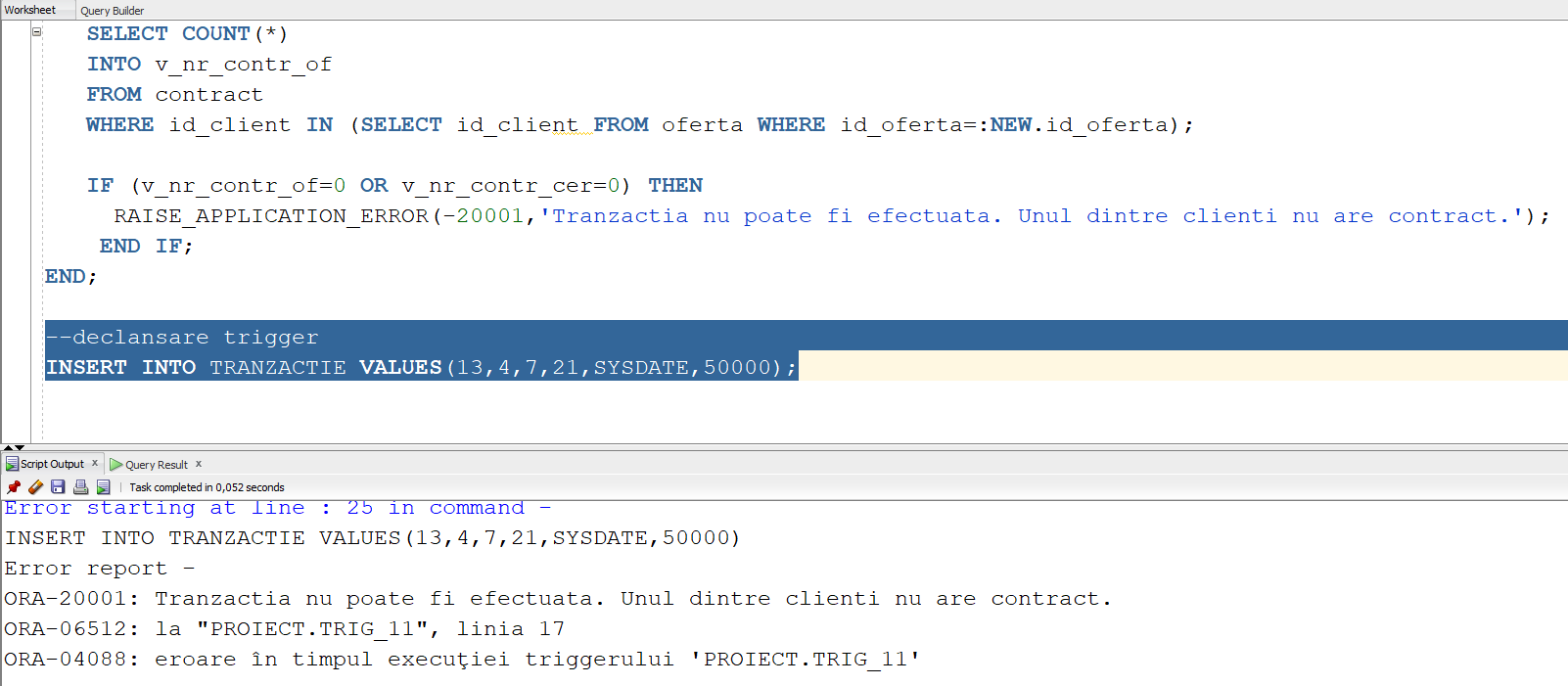
RAISE\_APPLICATION\_ERROR(-20001,'Tranzactia nu poate fi efectuata. Unul dintre clienti nu are contract.');

END IF;

END;



--declansare trigger

INSERT INTO TRANZACTIE VALUES(13,4,7,21,SYSDATE,50000);

# 12. Definiți un trigger de tip LDD. Declanșați trigger-ul.

**Trig\_12 blochează operația drop pentru tabele:**

CREATE OR REPLACE TRIGGER trig\_12

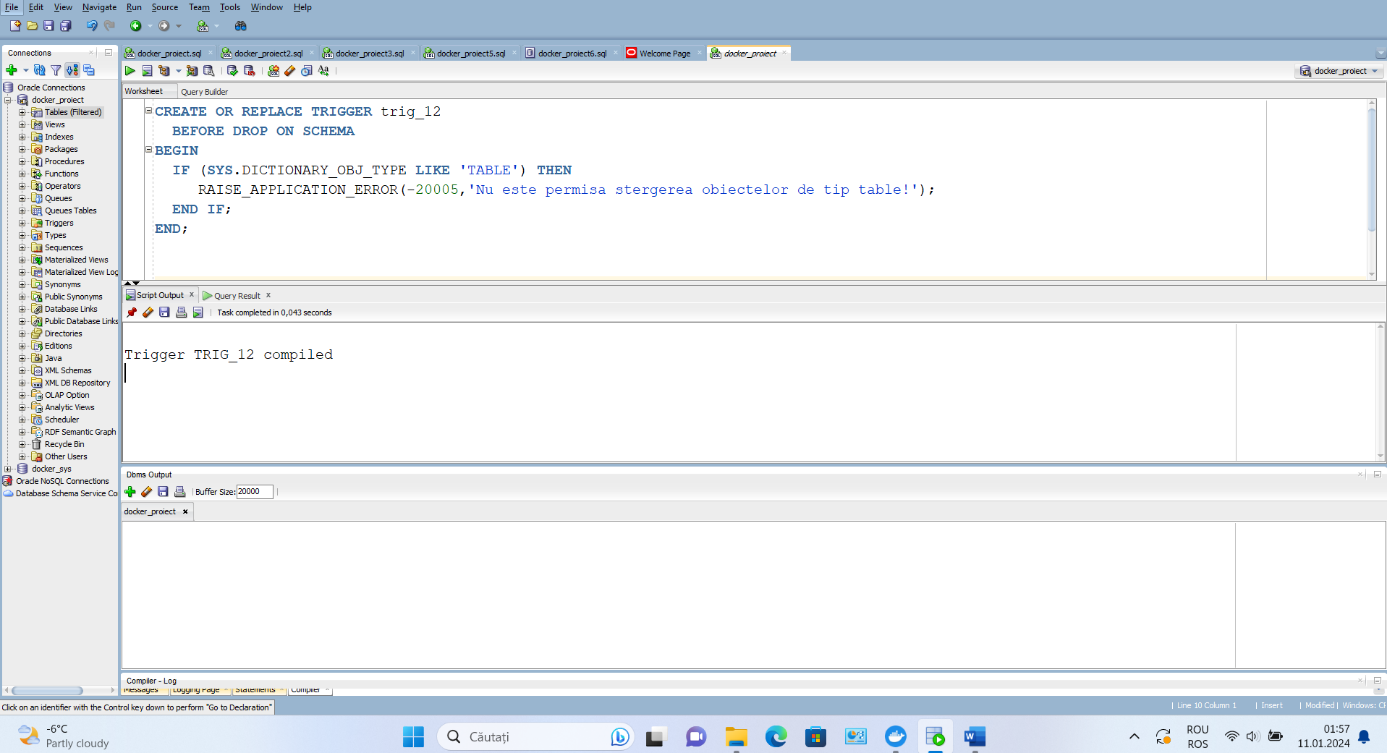
BEFORE DROP ON SCHEMA

BEGIN

IF (SYS.DICTIONARY\_OBJ\_TYPE LIKE 'TABLE') THEN

RAISE\_APPLICATION\_ERROR(-20005,'Nu este permisa stergerea obiectelor de tip table!');

END IF;

END;

--declanșare trigger

DROP TABLE MASINA;

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

CREATE OR REPLACE PACKAGE cerinta13 AS

PROCEDURE cerinta6;

PROCEDURE cerinta7;

FUNCTION f8(v\_prenume angajat.prenume%TYPE DEFAULT 'Vasile', data\_init contract.data%TYPE DEFAULT TO\_DATE('01/01/2023','DD/MM/YYYY'), data\_fin contract.data%TYPE DEFAULT TO\_DATE('31/12/2024','DD/MM/YYYY'))

RETURN NUMBER;

PROCEDURE OF\_PT\_CER(v\_id\_cerere IN cerere.id\_cerere%TYPE, v\_id\_oferta OUT oferta.id\_oferta%type);

END cerinta13;

/

CREATE OR REPLACE PACKAGE BODY cerinta13 as

--6

PROCEDURE cerinta6

IS

TYPE T\_id\_loc IS TABLE OF localitate.id\_localitate%TYPE INDEX BY PLS\_INTEGER;

TYPE T\_den\_loc IS TABLE OF localitate.denumire%TYPE;

TYPE T\_den\_jud IS VARRAY(50) OF judet.denumire%TYPE;

--record pt datele unei oferte

TYPE T\_of\_record IS RECORD ( loc NUMBER(10), ofertant VARCHAR(30),strada VARCHAR(30),numar VARCHAR(10),

suprafata NUMBER(10), an\_constructie NUMBER(4), pret NUMBER(15), op CHAR(1), tip\_propr VARCHAR(15), descr VARCHAR(30));

TYPE T\_of IS TABLE OF T\_of\_record INDEX BY PLS\_INTEGER;

v\_id\_loc T\_id\_loc;

v\_den\_loc T\_den\_loc;

v\_den\_jud T\_den\_jud;

v\_of T\_of;

BEGIN

SELECT id\_localitate BULK COLLECT INTO v\_id\_loc FROM localitate ORDER BY id\_localitate;

DBMS\_OUTPUT.PUT('Tablou indexat: ');

FOR i IN v\_id\_loc.FIRST..v\_id\_loc.LAST LOOP

DBMS\_OUTPUT.PUT(v\_id\_loc(i) || ' ');

END LOOP;

DBMS\_OUTPUT.NEW\_LINE;

SELECT denumire BULK COLLECT INTO v\_den\_loc FROM localitate ORDER BY id\_localitate;

DBMS\_OUTPUT.PUT('Tablou imbricat: ');

FOR i IN v\_den\_loc.FIRST..v\_den\_loc.LAST LOOP

DBMS\_OUTPUT.PUT(v\_den\_loc(i) || ' ');

END LOOP;

DBMS\_OUTPUT.NEW\_LINE;

SELECT j.denumire BULK COLLECT INTO v\_den\_jud

FROM localitate l join judet j on l.id\_judet=j.id\_judet

order by l.id\_localitate;

DBMS\_OUTPUT.PUT('Vector: ');

FOR i IN v\_den\_jud.FIRST..v\_den\_jud.LAST LOOP

DBMS\_OUTPUT.PUT(v\_den\_jud(i) || ' ');

END LOOP;

DBMS\_OUTPUT.NEW\_LINE;

DBMS\_OUTPUT.NEW\_LINE;

SELECT O.ID\_LOCALITATE LOC, INITCAP(TRIM(C.NUME)) || ' ' ||

INITCAP(TRIM(C.PRENUME)) OFERTANT, STRADA, NUMAR, SUPRAFATA, AN\_CONSTRUCTIE, PRET, ID\_OPERATIE OP, DECODE(ID\_PROPRIETATE,'C','CASA','A','APARTAMENT','T','TEREN') TIP\_PROPR, NVL(O.DESCRIERE,'') DESCR

BULK COLLECT INTO v\_of

FROM OFERTA O

JOIN CLIENT C ON O.ID\_CLIENT=C.ID\_CLIENT

WHERE NOT EXISTS ( SELECT \* FROM TRANZACTIE T

WHERE T.ID\_OFERTA = O.ID\_OFERTA

) --preluam doar ofertele disponibile, nu si cele tranzactionate

ORDER BY O.ID\_LOCALITATE, O.ID\_OPERATIE, O.ID\_PROPRIETATE, O.STRADA;

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

DBMS\_OUTPUT.PUT\_LINE(' JUDET LOCALITATE OFERTANT STRADA NUMAR SUPRAFATA CONSTRUIT OP TIP\_IMOBIL DESCR');

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

FOR i IN v\_of.FIRST..v\_of.LAST LOOP

DBMS\_OUTPUT.PUT\_LINE(RPAD(v\_den\_jud(v\_of(i).loc),10,' ') || ' ' || RPAD(v\_den\_loc(v\_of(i).loc),10,' ') || ' '|| RPAD(v\_of(i).ofertant,20,' ') || ' '|| v\_of(i).strada || ' '|| v\_of(i).numar || ' '|| v\_of(i).suprafata || ' '

|| v\_of(i).an\_constructie || ' '|| v\_of(i).pret || ' '|| v\_of(i).op || ' '|| v\_of(i).tip\_propr || ' '|| v\_of(i).descr || ' ');

END LOOP;

DBMS\_OUTPUT.NEW\_LINE;

END cerinta6;

--7

PROCEDURE cerinta7 IS

CURSOR c\_of (p\_id\_localitate localitate.id\_localitate%type) IS

(SELECT o.id\_localitate, COUNT(\*) AS nr\_of --id localitate si nr de oferte/localitate

FROM oferta o WHERE o.id\_localitate=p\_id\_localitate and o.id\_proprietate='A'

GROUP BY o.id\_localitate);

CURSOR c\_cer (p\_id\_localitate localitate.id\_localitate%type) IS

(SELECT c.id\_localitate, COUNT(\*) AS nr\_cer

FROM cerere c WHERE c.id\_localitate=p\_id\_localitate and c.id\_proprietate='A'

GROUP BY c.id\_localitate);

v\_id\_loc NUMBER(4);

v\_of NUMBER(4);

v\_cer NUMBER(4);

BEGIN

FOR i IN (SELECT id\_localitate, denumire FROM localitate) LOOP --ciclu cursor cu subcereri

OPEN c\_of(i.id\_localitate);

FETCH c\_of INTO v\_id\_loc, v\_of;

OPEN c\_cer(i.id\_localitate);

FETCH c\_cer INTO v\_id\_loc, v\_cer;

IF (v\_cer < v\_of) THEN

DBMS\_OUTPUT.PUT\_LINE(i.denumire || ' cereri ' || v\_cer || ' oferte ' || v\_of);

END IF;

CLOSE c\_of;

CLOSE c\_cer;

END LOOP;

END cerinta7;

--8

FUNCTION f8 (v\_prenume angajat.prenume%TYPE DEFAULT 'Vasile', data\_init contract.data%TYPE DEFAULT TO\_DATE('01/01/2023','DD/MM/YYYY'), data\_fin contract.data%TYPE DEFAULT TO\_DATE('31/12/2024','DD/MM/YYYY'))

RETURN NUMBER IS

v\_nr\_contr NUMBER(4);

v\_nr\_ang NUMBER(2);

exception1 EXCEPTION;

exception2 EXCEPTION;

BEGIN

SELECT COUNT(id\_angajat)

INTO v\_nr\_ang

FROM angajat

WHERE UPPER(prenume) = UPPER(v\_prenume);

IF (v\_nr\_ang = 0) THEN RAISE exception1;

END IF;

IF (v\_nr\_ang > 1) THEN RAISE exception2;

END IF;

--in nc s-a calculat nr de contracte pt fiecare angajat care are minim o cerere sau o o oferta

SELECT nc.nr\_contr INTO v\_nr\_contr FROM angajat a

LEFT JOIN (SELECT c.id\_angajat, COUNT(\*) as nr\_contr FROM contract c

WHERE c.data BETWEEN data\_init AND data\_fin AND ( (c.id\_client IN (SELECT distinct id\_client FROM cerere)) OR

(c.id\_client IN (SELECT distinct id\_client FROM oferta)) )

GROUP BY c.id\_angajat) nc

ON a.id\_angajat = nc.id\_angajat

WHERE UPPER(a.prenume) = UPPER(v\_prenume);

RETURN NVL(v\_nr\_contr,0);

EXCEPTION

WHEN exception1 THEN

BEGIN

DBMS\_OUTPUT.PUT\_LINE('EROARE: Nu exista niciun angajat cu prenumele dat!');

RETURN -1;

END;

WHEN exception2 THEN

BEGIN

DBMS\_OUTPUT.PUT\_LINE('EROARE: Exista mai multi angajati cu prenumele dat!');

RETURN -1;

END;

WHEN OTHERS THEN RAISE\_APPLICATION\_ERROR(-20002,'Alta eroare!');

END f8;

--9

PROCEDURE

OF\_PT\_CER(v\_id\_cerere IN cerere.id\_cerere%TYPE,

v\_id\_oferta OUT oferta.id\_oferta%type) IS

v\_id\_localitate localitate.id\_localitate%TYPE;

v\_id\_proprietate proprietate.id\_proprietate%TYPE;

v\_buget cerere.buget%TYPE;

v\_supraf\_min cerere.suprafata\_minima%TYPE;

v\_an\_constructie cerere.an\_constructie%TYPE;

v\_id\_operatie operatie.id\_operatie%TYPE;

v\_nr NUMBER(3);

v\_nume client.nume%TYPE;

v\_prenume client.prenume%TYPE;

v\_denumire localitate.denumire%TYPE;

BEGIN

SELECT COUNT(\*) INTO v\_nr FROM cerere WHERE id\_cerere=v\_id\_cerere;

IF (v\_nr=0) THEN

DBMS\_OUTPUT.PUT\_LINE('Nu exista cerere cu id-ul dat!');

RETURN;

END IF;

SELECT id\_localitate, id\_proprietate,buget, suprafata\_minima, an\_constructie, id\_operatie

INTO v\_id\_localitate, v\_id\_proprietate, v\_buget, v\_supraf\_min, v\_an\_constructie, v\_id\_operatie

FROM cerere

WHERE id\_cerere=v\_id\_cerere;

SELECT id\_oferta,c.nume, c.prenume,l.denumire INTO v\_id\_oferta, v\_nume, v\_prenume, v\_denumire

FROM oferta o

JOIN client c on o.id\_client=c.id\_client

JOIN localitate l on o.id\_localitate=l.id\_localitate

WHERE o.id\_localitate = v\_id\_localitate AND o.id\_proprietate=v\_id\_proprietate AND

o.pret<=v\_buget AND o.suprafata>=v\_supraf\_min AND ABS(o.an\_constructie-v\_an\_constructie)<=10

AND ((v\_id\_operatie='C' AND o.id\_operatie='V') OR (v\_id\_operatie='I' AND o.id\_operatie='I'))

AND o.id\_oferta NOT IN (SELECT id\_oferta

FROM tranzactie)

AND o.id\_client IN (SELECT id\_client

FROM contract);

DBMS\_OUTPUT.PUT\_LINE ('Oferta ' || v\_id\_oferta || ' facuta de ' || v\_nume || ' ' || v\_prenume || ' in localitatea ' || v\_denumire);

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

RAISE\_APPLICATION\_ERROR(-20000, 'Nu exista oferta care sa respecte cererea!');

WHEN TOO\_MANY\_ROWS THEN

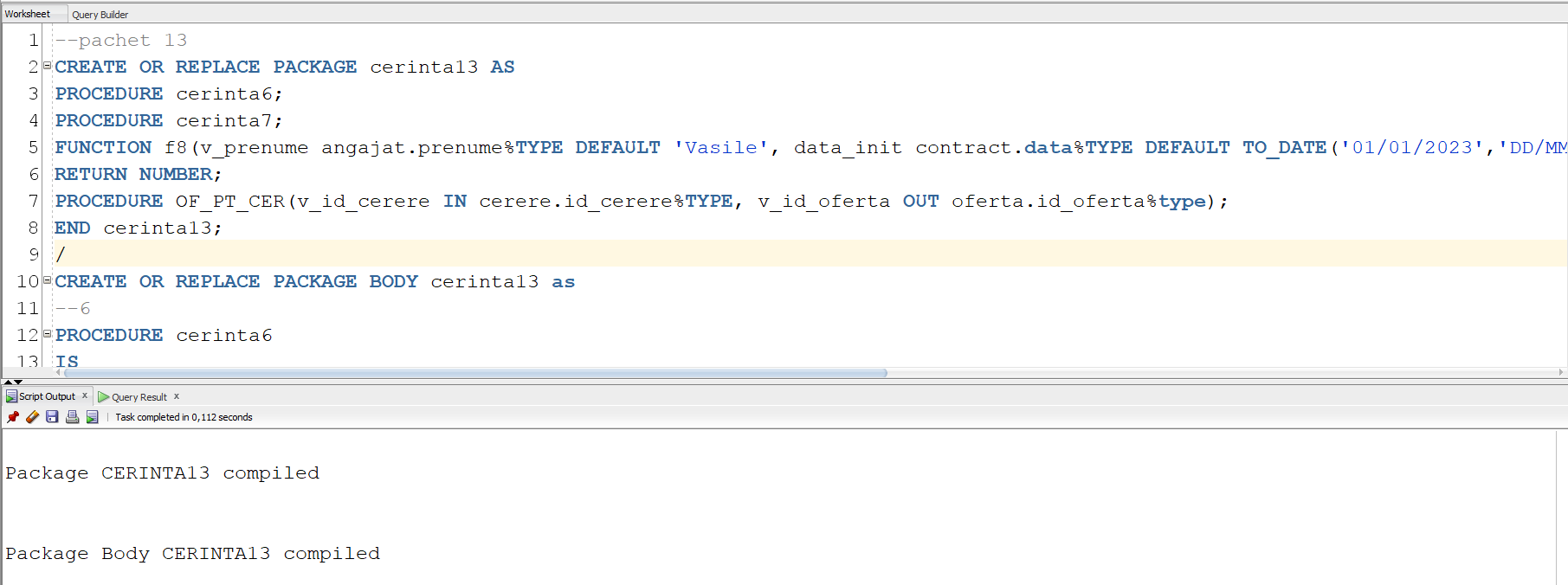
RAISE\_APPLICATION\_ERROR(-20001,'Exista mai multe oferte care respecta cererea!');

WHEN OTHERS THEN

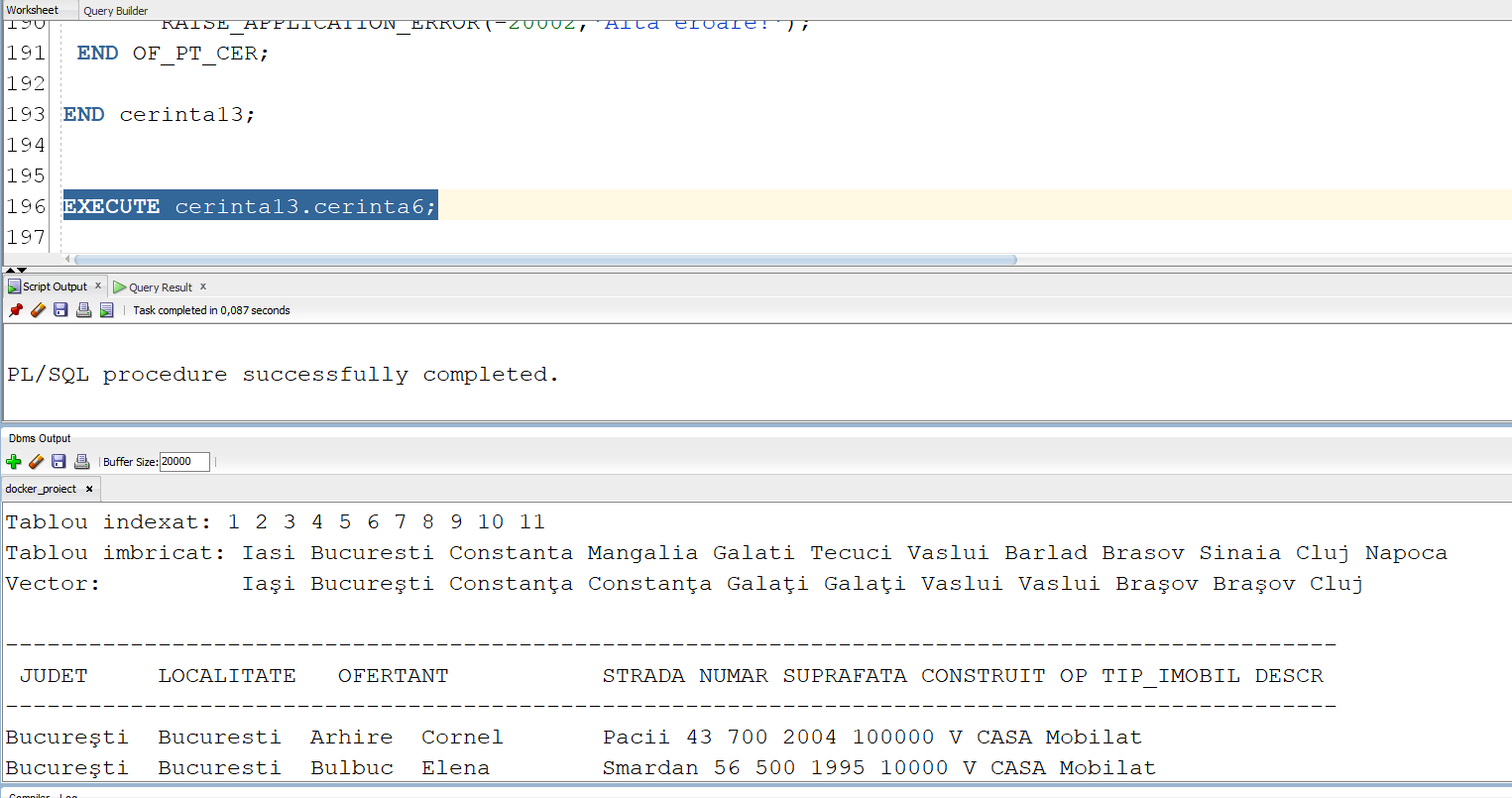
RAISE\_APPLICATION\_ERROR(-20002,'Alta eroare!');

END OF\_PT\_CER;

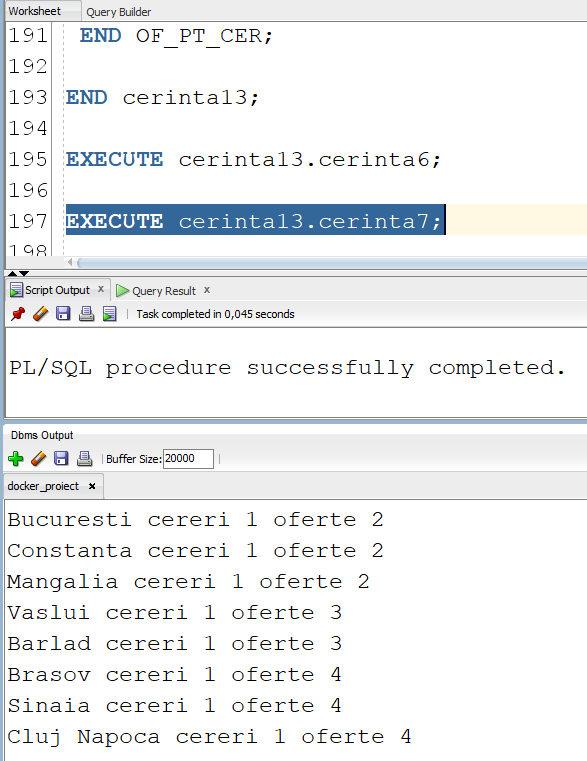
END cerinta13;



**Testare pachet**:

EXECUTE cerinta13.cerinta6;

EXECUTE cerinta13.cerinta7;



DECLARE

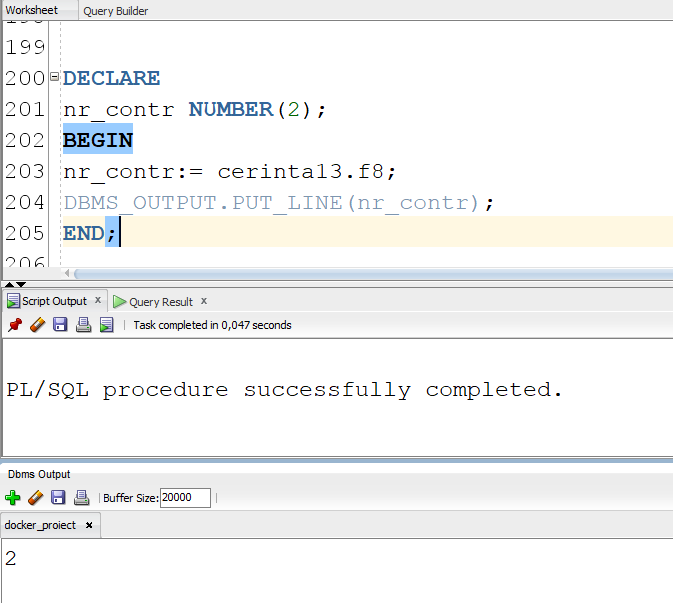
nr\_contr NUMBER(2);

BEGIN

nr\_contr:= cerinta13.f8;

DBMS\_OUTPUT.PUT\_LINE(nr\_contr);

END;



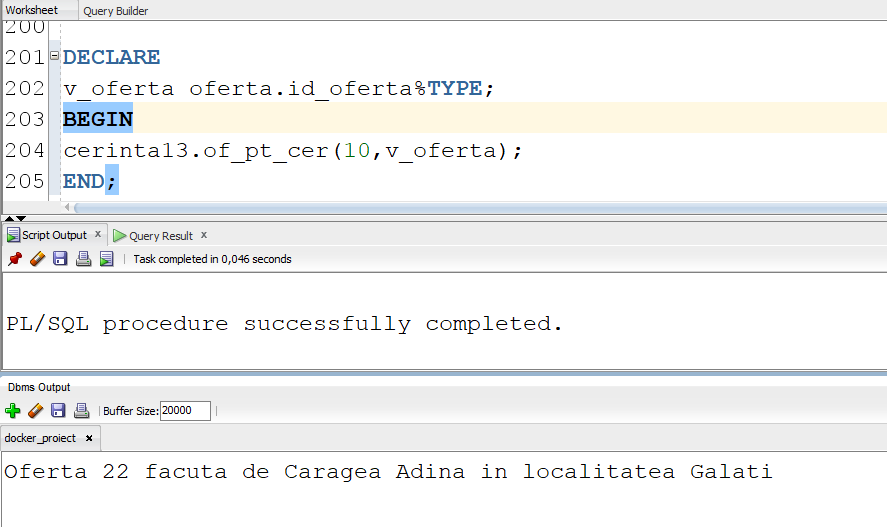
DECLARE

v\_oferta oferta.id\_oferta%TYPE;

BEGIN

cerinta13.of\_pt\_cer(10,v\_oferta);

END;



# 14. 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).

**Pachetul cerinta14 facilitează asocierea cererilor cu ofertele corespunzătoare și invers. El are ca scop afișarea tuturor ofertelor potrivite pentru o cerere sau tuturor cererilor care corespund unei oferte.**

CREATE OR REPLACE PACKAGE cerinta14 AS

CURSOR of\_disponibile(pid\_oferta oferta.id\_oferta%TYPE) RETURN dual%rowtype;

CURSOR cer\_disponibile(pid\_cerere cerere.id\_cerere%TYPE) RETURN dual%rowtype;

TYPE t\_detalii\_of IS RECORD (

id\_oferta oferta.id\_oferta%TYPE,

nume client.nume%TYPE, prenume client.prenume%TYPE,

id\_operatie oferta.id\_operatie%TYPE, id\_proprietate oferta.id\_proprietate%TYPE,

id\_localitate oferta.id\_localitate%TYPE, loc localitate.denumire%TYPE,

suprafata oferta.suprafata%TYPE, an\_constructie oferta.an\_constructie%TYPE, pret oferta.pret%TYPE,

descriere oferta.descriere%TYPE,strada oferta.strada%TYPE, numar oferta.numar%TYPE

);

TYPE t\_detalii\_cer IS RECORD (

id\_cerere cerere.id\_cerere%TYPE,

nume client.nume%TYPE, prenume client.prenume%TYPE,

id\_operatie cerere.id\_operatie%TYPE, id\_proprietate cerere.id\_proprietate%TYPE,

id\_localitate cerere.id\_localitate%TYPE, loc localitate.denumire%TYPE,

suprafata cerere.suprafata\_minima%TYPE, an\_constructie cerere.an\_constructie%TYPE, pret cerere.buget%TYPE

);

TYPE tablou\_indexat\_of IS TABLE OF t\_detalii\_of INDEX BY PLS\_INTEGER;

TYPE tablou\_indexat\_cer IS TABLE OF t\_detalii\_cer INDEX BY PLS\_INTEGER;

FUNCTION are\_contract\_cer(pid\_cerere cerere.id\_cerere%TYPE) RETURN NUMBER;

FUNCTION are\_contract\_of(pid\_oferta oferta.id\_oferta%TYPE) RETURN NUMBER;

FUNCTION disponibilitate\_cer(pid\_cerere cerere.id\_cerere%TYPE) RETURN NUMBER;

FUNCTION disponibilitate\_of(pid\_oferta oferta.id\_oferta%TYPE) RETURN NUMBER;

PROCEDURE afis\_of\_potriv(pid\_cerere cerere.id\_cerere%TYPE);

PROCEDURE afis\_cer\_potriv(pid\_oferta oferta.id\_oferta%TYPE);

END cerinta14;

/

CREATE OR REPLACE PACKAGE BODY cerinta14 AS

CURSOR of\_disponibile(pid\_oferta oferta.id\_oferta%TYPE) RETURN dual%rowtype IS SELECT COUNT(\*) FROM oferta WHERE id\_oferta=pid\_oferta AND id\_oferta NOT IN (SELECT id\_oferta FROM tranzactie); -- **verif daca oferta cu id-ul dat ca parametru nu e tranzactionata; 1=netranz.**

CURSOR cer\_disponibile(pid\_cerere cerere.id\_cerere%TYPE) RETURN dual%rowtype IS SELECT COUNT(\*) FROM cerere WHERE id\_cerere=pid\_cerere AND id\_cerere NOT IN (SELECT id\_cerere FROM tranzactie);

--**verificam daca clientul care face cererea are contract cu agentia**

FUNCTION are\_contract\_cer(pid\_cerere cerere.id\_cerere%TYPE) RETURN NUMBER IS

v\_nr NUMBER;

BEGIN

SELECT COUNT(\*) INTO v\_nr

FROM contract WHERE id\_client IN (SELECT id\_client FROM cerere WHERE id\_cerere=pid\_cerere);

RETURN v\_nr; --**v\_nr=0 -> clientul nu are contract cu agentia**

END;

--**verificam daca clientul care face oferta are contract cu agentia**

FUNCTION are\_contract\_of(pid\_oferta oferta.id\_oferta%TYPE) RETURN NUMBER IS

v\_nr NUMBER;

BEGIN

SELECT COUNT(\*) INTO v\_nr

FROM contract WHERE id\_client IN (SELECT id\_client FROM oferta WHERE id\_oferta=pid\_oferta);

RETURN v\_nr;

END;

FUNCTION disponibilitate\_cer(pid\_cerere cerere.id\_cerere%TYPE) RETURN NUMBER IS

v\_nr NUMBER;

BEGIN

OPEN cer\_disponibile(pid\_cerere); --**verificam disponibilitatea cer. cu ajutorul cursorului**

FETCH cer\_disponibile INTO v\_nr;

CLOSE cer\_disponibile;

RETURN v\_nr;

END;

FUNCTION disponibilitate\_of(pid\_oferta oferta.id\_oferta%TYPE) RETURN NUMBER IS

v\_nr NUMBER;

BEGIN

OPEN of\_disponibile(pid\_oferta);

FETCH of\_disponibile INTO v\_nr;

CLOSE of\_disponibile;

RETURN v\_nr;

END;

PROCEDURE afis\_of\_potriv(pid\_cerere cerere.id\_cerere%TYPE) IS

t tablou\_indexat\_of; --**tablou cu record-uri cu detalii despre fiecare oferta**

cer t\_detalii\_cer; --**record cu date despre cererea data ca parametru**

i NUMBER(3);

BEGIN

IF disponibilitate\_cer(pid\_cerere)=0 THEN

DBMS\_OUTPUT.PUT\_LINE ('Cererea ' || pid\_cerere || ' nu este disponibila!');

RETURN;

END IF;

--**preiau detalii cerere din tabela in variabila cer**

SELECT id\_cerere, c.nume, c.prenume, id\_operatie, id\_proprietate,o.id\_localitate, l.denumire as loc, suprafata\_minima as suprafata, an\_constructie, buget as pret

INTO cer

FROM cerere o

JOIN client c ON o.id\_client=c.id\_client

JOIN localitate l ON o.id\_localitate=l.id\_localitate

WHERE id\_cerere=pid\_cerere;

--**afisam cererea**

DBMS\_OUTPUT.PUT\_LINE ('Pentru cererea ' || pid\_cerere || ' ' || cer.nume|| ' ' || cer.prenume || ' ' || cer.id\_operatie || ' ' || cer.id\_proprietate || ' ' || cer.loc || ' ' || cer.suprafata || ' ' || cer.an\_constructie || ' ' || cer.pret);

--**creez tabloul cu oferte potrivite**

SELECT id\_oferta, c.nume, c.prenume, id\_operatie, o.id\_proprietate,o.id\_localitate, l.denumire as loc, suprafata, an\_constructie, pret, descriere, strada, numar

BULK COLLECT INTO T

FROM oferta o

JOIN client c ON o.id\_client=c.id\_client

JOIN localitate l ON o.id\_localitate=l.id\_localitate

**--verificam potrivirea caracteristicilor**

WHERE o.id\_proprietate=cer.id\_proprietate AND o.id\_localitate=cer.id\_localitate AND o.suprafata>=cer.suprafata AND ABS(an\_constructie - cer.an\_constructie)<=10 AND o.pret<=cer.pret

AND are\_contract\_of(id\_oferta)>0 AND disponibilitate\_of(o.id\_oferta)>0 AND ((cer.id\_operatie='C' AND o.id\_operatie='V') OR (cer.id\_operatie='I' AND o.id\_operatie='I'));

--**afisare oferte potrivite**

IF t.COUNT = 0 THEN

DBMS\_OUTPUT.PUT\_LINE ('Nu exista oferte pentru cererea ' || pid\_cerere);

RETURN;

END IF;

FOR i IN t.FIRST..t.LAST LOOP

DBMS\_OUTPUT.PUT(t(i).id\_oferta || ' ' || t(i).nume || ' ' || t(i).prenume || ' ' || t(i).id\_operatie || ' ' || t(i).id\_proprietate || ' ' || t(i).loc || ' ' || t(i).suprafata || ' ' || t(i).an\_constructie || ' ' || t(i).pret || ' ' || t(i).descriere || ' ' || t(i).strada || ' ' || t(i).numar);

DBMS\_OUTPUT.NEW\_LINE;

END LOOP;

END;

PROCEDURE afis\_cer\_potriv(pid\_oferta oferta.id\_oferta%TYPE) IS

t tablou\_indexat\_cer; --**tablou cu record-uri cu detalii despre fiecare cerere**

ofr t\_detalii\_of;

i NUMBER(3);

BEGIN

IF disponibilitate\_of(pid\_oferta)=0 THEN

DBMS\_OUTPUT.PUT\_LINE ('Oferta ' || pid\_oferta || ' nu este disponibila!');

RETURN;

END IF;

**--preiau detalii oferta din tabela in variabila ofr**

SELECT id\_oferta, c.nume, c.prenume, id\_operatie, id\_proprietate,o.id\_localitate, l.denumire as loc, suprafata as suprafata, an\_constructie,pret,descriere,strada, numar

INTO ofr

FROM oferta o

JOIN client c ON o.id\_client=c.id\_client

JOIN localitate l ON o.id\_localitate=l.id\_localitate

WHERE id\_oferta=pid\_oferta;

--**afiseaza oferta**

DBMS\_OUTPUT.PUT\_LINE ('Pentru oferta ' || pid\_oferta || ' ' || ofr.nume|| ' ' || ofr.prenume || ' ' || ofr.id\_operatie || ' ' || ofr.id\_proprietate || ' ' || ofr.loc || ' ' || ofr.suprafata || ' ' || ofr.an\_constructie || ' ' || ofr.pret || ' ' || ofr.descriere || ' ' || ofr.strada || ' ' || ofr.numar);

--**creeaza tabelul cu cereri potrivite**

SELECT id\_cerere, c.nume, c.prenume, id\_operatie, o.id\_proprietate,o.id\_localitate, l.denumire as loc, suprafata\_minima as suprafata, an\_constructie, buget as pret

BULK COLLECT INTO T

FROM cerere o

JOIN client c ON o.id\_client=c.id\_client

JOIN localitate l ON o.id\_localitate=l.id\_localitate

WHERE o.id\_proprietate=ofr.id\_proprietate AND o.id\_localitate=ofr.id\_localitate AND o.suprafata\_minima<=ofr.suprafata AND ABS(an\_constructie - ofr.an\_constructie)<=10 AND o.buget>=ofr.pret

AND are\_contract\_cer(o.id\_cerere)>0 AND disponibilitate\_cer(o.id\_cerere)>0 AND ((ofr.id\_operatie='V' AND o.id\_operatie='C') OR (ofr.id\_operatie='I' AND o.id\_operatie='I'));

--**afisare cereri potrivite**

IF t.COUNT = 0 THEN

DBMS\_OUTPUT.PUT\_LINE ('Nu exista cereri pentru oferta ' || pid\_oferta);

RETURN;

END IF;

FOR i IN t.FIRST..t.LAST LOOP

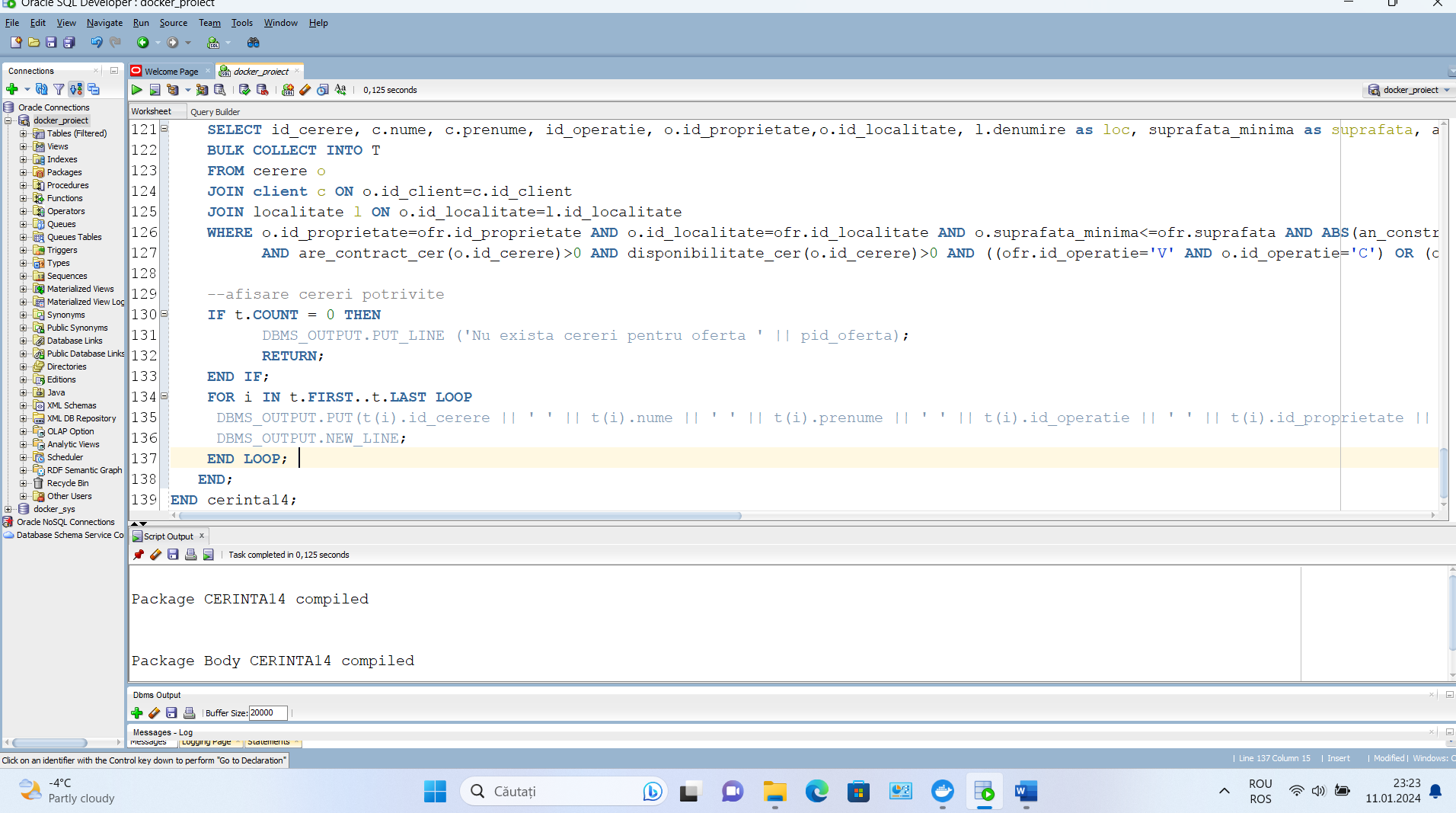
DBMS\_OUTPUT.PUT(t(i).id\_cerere || ' ' || t(i).nume || ' ' || t(i).prenume || ' ' || t(i).id\_operatie || ' ' || t(i).id\_proprietate || ' ' || t(i).loc || ' ' || t(i).suprafata || ' ' || t(i).an\_constructie || ' ' || t(i).pret);

DBMS\_OUTPUT.NEW\_LINE;

END LOOP;

END;

END cerinta14;



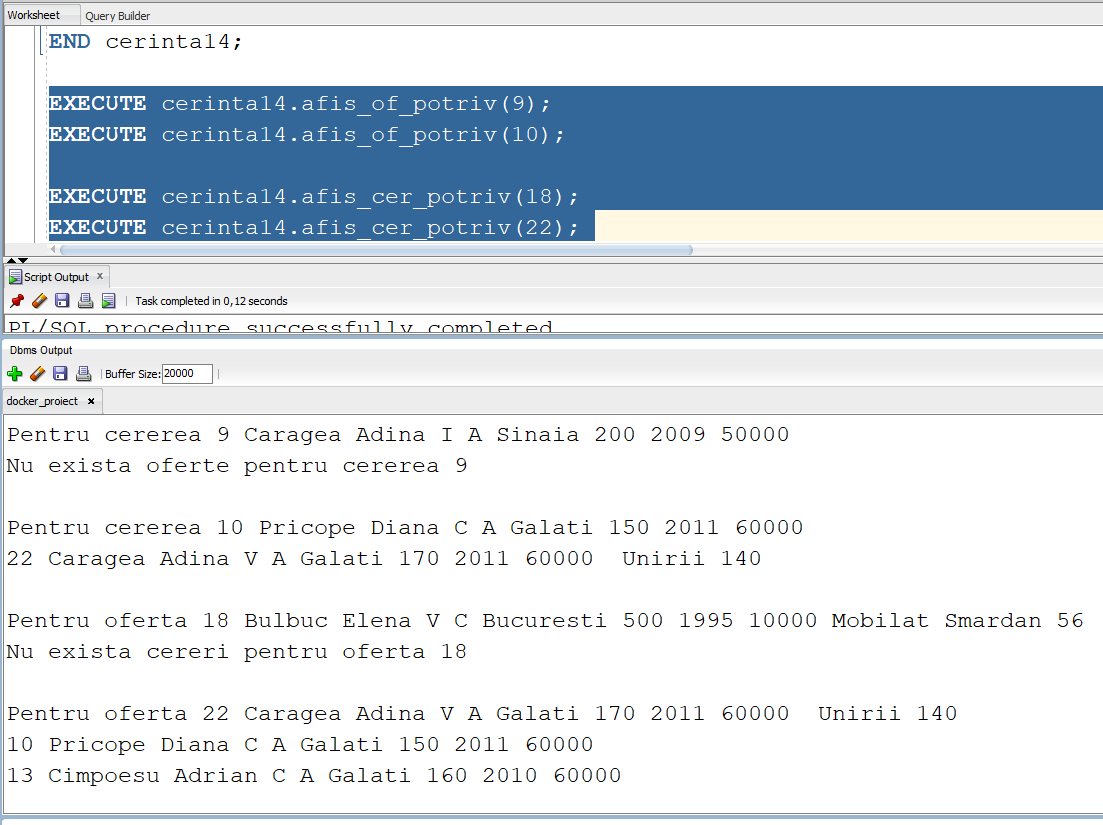
**Testare pachet**

EXECUTE cerinta14.afis\_of\_potriv(9);

EXECUTE cerinta14.afis\_of\_potriv(10);

EXECUTE cerinta14.afis\_cer\_potriv(18);

EXECUTE cerinta14.afis\_cer\_potriv(22);

****

Cuprins

[1. Prezentați pe scurt baza de date (utilitatea ei). 1](#_Toc155993435)

[2. Realizați diagrama entitate-relație (ERD) 2](#_Toc155993436)

[3. Pornind de la diagrama entitate-relație realizați diagrama conceptuală a modelului propus, integrând toate atributele necesare 3](#_Toc155993437)

[4. Implementați în Oracle diagrama conceptuală realizată: definiți toate tabelele, definind toate constrângerile de integritate necesare (chei primare, cheile externe etc). 5](#_Toc155993438)

[5. Adăugați informații coerente în tabelele create (minim 5 înregistrări pentru fiecare entitate independentă; minim 10 înregistrări pentru tabela asociativă). 12](#_Toc155993439)

[6. 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. 22](#_Toc155993440)

[7. 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. 25](#_Toc155993441)

[8. 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. 27](#_Toc155993442)

[9. 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. 30](#_Toc155993443)

[10. Definiți un trigger de tip LMD la nivel de comandă. Declanșați trigger-ul. 33](#_Toc155993444)

[11. Definiți un trigger de tip LMD la nivel de linie. Declanșați trigger-ul 36](#_Toc155993445)

[12. Definiți un trigger de tip LDD. Declanșați trigger-ul. 38](#_Toc155993446)

[13. Definiți un pachet care să conțină toate obiectele definite în cadrul proiectului. 38](#_Toc155993447)

[14. 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). 46](#_Toc155993448)