**Proiect SGBD 2022 – 2023**

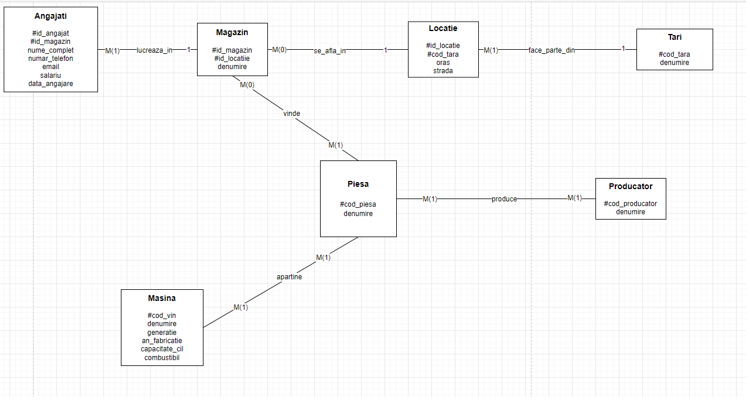
**Dina George-Alexandru**

**Grupa 233**

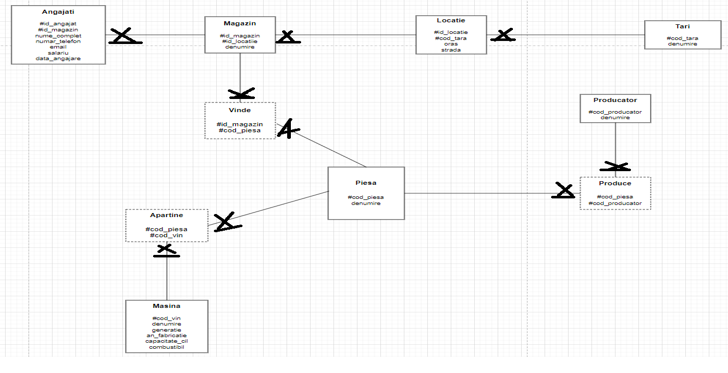
**1. Prezentați pe scurt baza de date (utilitatea ei).**

**Acest proiect prezintă baza de date a unui lanț de magazine de piese auto. Putem lua drept exemplu lanțul Unix. Acesta conține mai multe magazine, situate în zone diverse ale țării. După acest exemplu funcționează și baza aceasta de date. Astfel pentru fiecare magazin vom știi locația (orașul, strada), dar și țara în care acesta se află (îl gândim ca fiind internațional). La fel de bine avem, pentru fiecare magazin, o listă cu angajații acestuia. Fiind un magazin de piese auto, automat vom reține undeva și piesele auto care pot fi vândute în cadrul acestuia. Pentru fiecare piesă auto, știm cu ce mașină este compatibilă. Prin intermediul acestei baze de date, un utilizator poate să își caute o anumită piesă, cunoscând seria mașinii, codul piesei și totodată va primi și informații despre magazinile în care se află și locația acestora.**

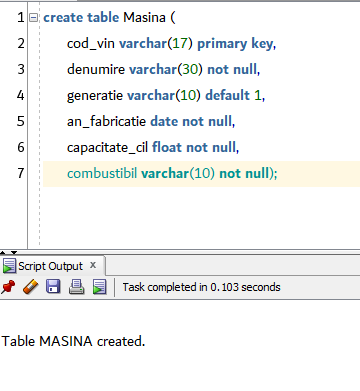
**2.** **Realizați diagrama entitate-relație (ERD).**

****

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

****

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



**create table Masina (**

**cod\_vin varchar(17) primary key,**

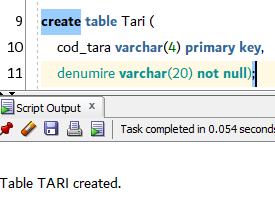
**denumire varchar(30) not null,**

**generatie varchar(10) default 1,**

**an\_fabricatie date not null,**

**capacitate\_cil float not null,**

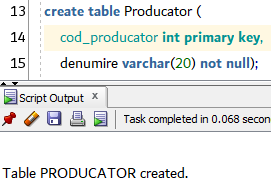
**combustibil varchar(10) not null);**

****

**create table Tari (**

**cod\_tara varchar(4) primary key,**

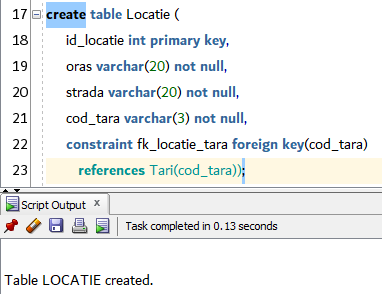
**denumire varchar(20) not null);**

****

**create table Producator (**

**cod\_producator int primary key,**

**denumire varchar(20) not null);**

****

**create table Locatie (**

**id\_locatie int primary key,**

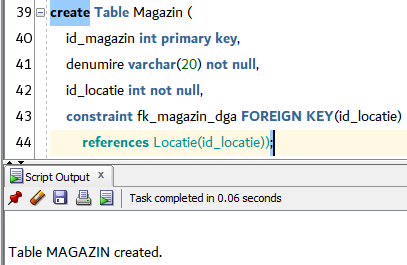
**oras varchar(20) not null,**

**strada varchar(20) not null,**

**cod\_tara varchar(3) not null,**

**constraint fk\_locatie\_tara foreign key(cod\_tara)**

**references Tari(cod\_tara));**

****

**create Table Magazin (**

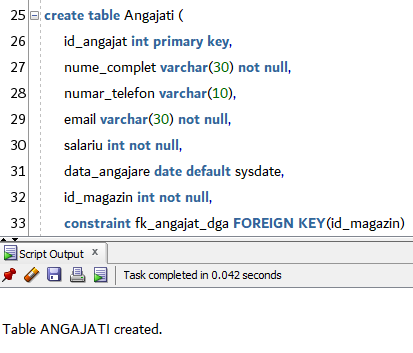
**id\_magazin int primary key,**

**denumire varchar(20) not null,**

**id\_locatie int not null,**

**constraint fk\_magazin\_dga FOREIGN KEY(id\_locatie)**

**references Locatie(id\_locatie));**

****

**create table Angajati (**

**id\_angajat int primary key,**

**nume\_complet varchar(30) not null,**

**numar\_telefon varchar(10),**

**email varchar(30) not null,**

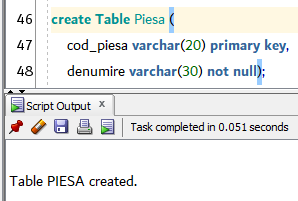
**salariu int not null,**

**data\_angajare date default sysdate,**

**id\_magazin int not null,**

**constraint fk\_angajat\_dga FOREIGN KEY(id\_magazin)**

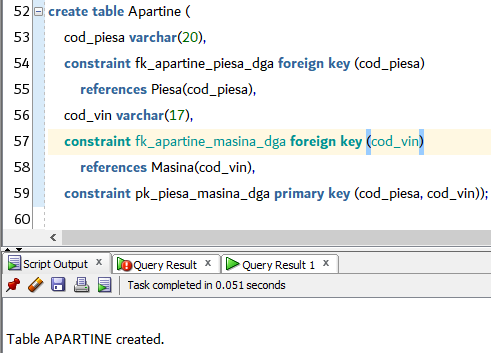
**references Magazin(id\_magazin));**

****

**create Table Piesa (**

**cod\_piesa varchar(20) primary key,**

**denumire varchar(30) not null);**

****

**create table Apartine (**

**cod\_piesa varchar(20),**

**constraint fk\_apartine\_piesa\_dga foreign key (cod\_piesa)**

**references Piesa(cod\_piesa),**

**cod\_vin varchar(17),**

**constraint fk\_apartine\_masina\_dga foreign key (cod\_vin)**

**references Masina(cod\_vin),**

**constraint pk\_piesa\_masina\_dga primary key (cod\_piesa, cod\_vin));**

**create table Produce (**

**cod\_piesa varchar(20),**

**constraint fk\_produce\_piesa\_dga foreign key (cod\_piesa)**

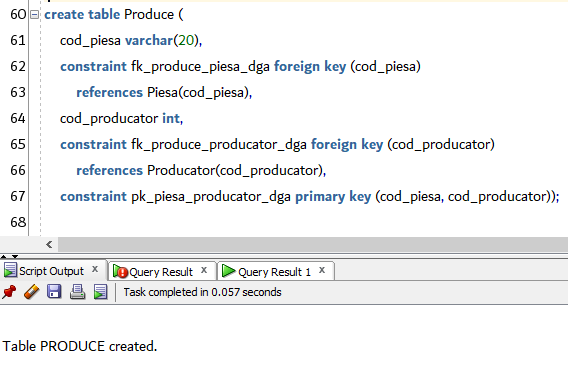
**references Piesa(cod\_piesa),**

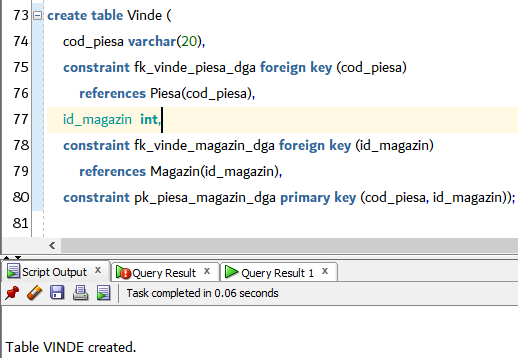
**cod\_producator int,**

**constraint fk\_produce\_producator\_dga foreign key (cod\_producator)**

**references Producator(cod\_producator),**

**constraint pk\_piesa\_producator\_dga primary key (cod\_piesa, cod\_producator));**

****

****

**create table Vinde (**

**cod\_piesa varchar(20),**

**constraint fk\_vinde\_piesa\_dga foreign key (cod\_piesa)**

**references Piesa(cod\_piesa),**

**id\_magazin int,**

**constraint fk\_vinde\_magazin\_dga foreign key (id\_magazin)**

**references Magazin(id\_magazin),**

**constraint pk\_piesa\_magazin\_dga primary key (cod\_piesa, id\_magazin));**

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

insert into Masina values('WAUDK78T29A014036', 'Audi A5', 'B8', to\_date('01-01-2009', 'DD-MM-YYYY'), 2.0, 'motorina');

insert into Masina values('WAUDH74F08N050846', 'Audi A6', 'C6', to\_date('05-05-2008', 'DD-MM-YYYY'), 3.2, 'benzina');

insert into Masina (cod\_vin, denumire, an\_fabricatie, capacitate\_cil, combustibil) values('3VWPP7AJXDM604357', 'Volkswagen Jetta Sportwagen', to\_date('06-05-2013', 'DD-MM-YYYY'), 2.5, 'benzina');

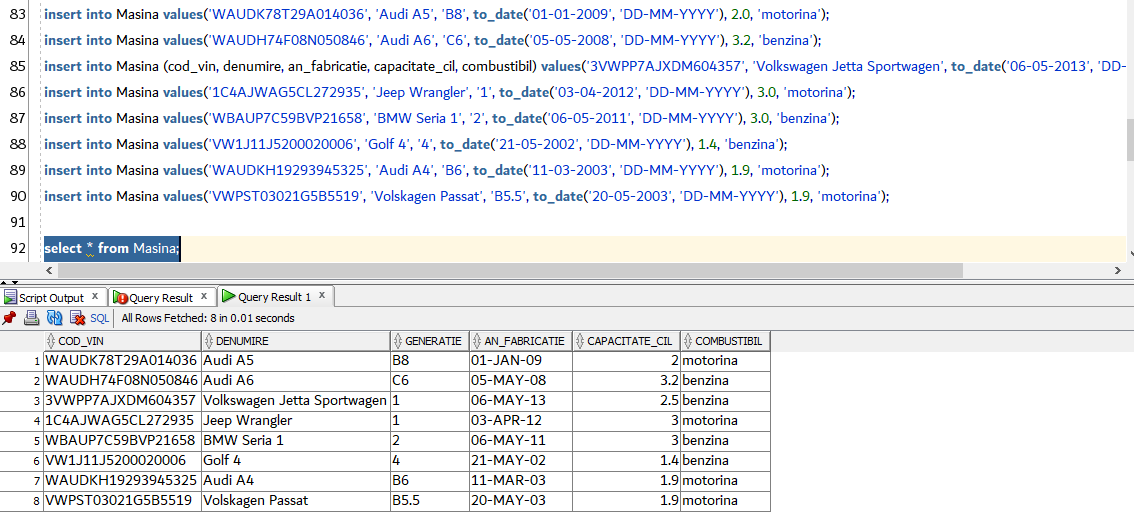
insert into Masina values('1C4AJWAG5CL272935', 'Jeep Wrangler', '1', to\_date('03-04-2012', 'DD-MM-YYYY'), 3.0, 'motorina');

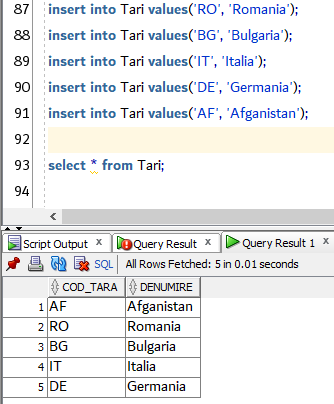
insert into Masina values('WBAUP7C59BVP21658', 'BMW Seria 1', '2', to\_date('06-05-2011', 'DD-MM-YYYY'), 3.0, 'benzina');

insert into Masina values('VW1J11J5200020006', 'Golf 4', '4', to\_date('21-05-2002', 'DD-MM-YYYY'), 1.4, 'benzina');

insert into Masina values('WAUDKH19293945325', 'Audi A4', 'B6', to\_date('11-03-2003', 'DD-MM-YYYY'), 1.9, 'motorina');

insert into Masina values('VWPST03021G5B5519', 'Volskagen Passat', 'B5.5', to\_date('20-05-2003', 'DD-MM-YYYY'), 1.9, 'motorina');

****

****

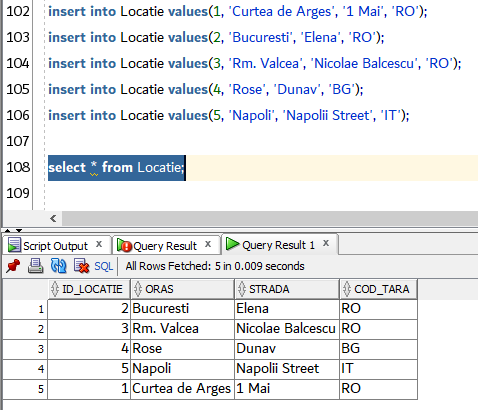
insert into Tari values('RO', 'Romania');

insert into Tari values('BG', 'Bulgaria');

insert into Tari values('IT', 'Italia');

insert into Tari values('DE', 'Germania');

insert into Tari values('AF', 'Afganistan');

****

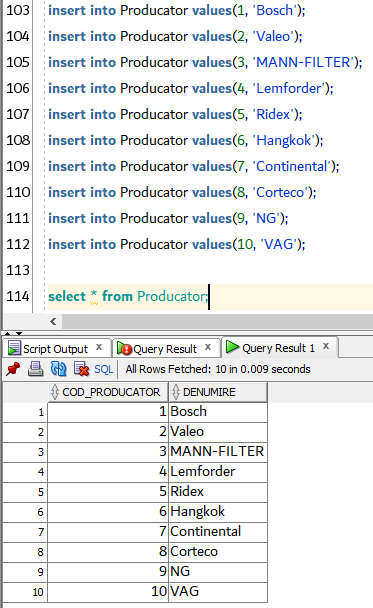
insert into Locatie values(1, 'Curtea de Arges', '1 Mai', 'RO');

insert into Locatie values(2, 'Bucuresti', 'Elena', 'RO');

insert into Locatie values(3, 'Rm. Valcea', 'Nicolae Balcescu', 'RO');

insert into Locatie values(4, 'Rose', 'Dunav', 'BG');

insert into Locatie values(5, 'Napoli', 'Napolii Street', 'IT');

****

insert into Producator values(1, 'Bosch');

insert into Producator values(2, 'Valeo');

insert into Producator values(3, 'MANN-FILTER');

insert into Producator values(4, 'Lemforder');

insert into Producator values(5, 'Ridex');

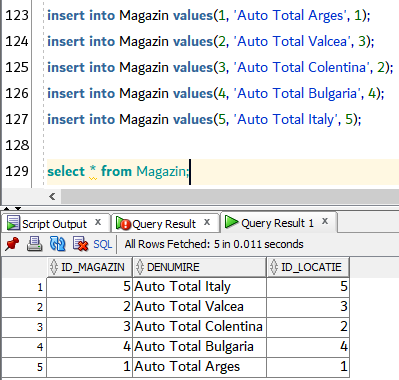
insert into Producator values(6, 'Hangkok');

insert into Producator values(7, 'Continental');

insert into Producator values(8, 'Corteco');

insert into Producator values(9, 'NG');

insert into Producator values(10, 'VAG');

****

insert into Magazin values(1, 'Auto Total Arges', 1);

insert into Magazin values(2, 'Auto Total Valcea', 3);

insert into Magazin values(3, 'Auto Total Colentina', 2);

insert into Magazin values(4, 'Auto Total Bulgaria', 4);

insert into Magazin values(5, 'Auto Total Italy', 5);

CREATE SEQUENCE genereaza\_ang

MINVALUE 0

START WITH 1

INCREMENT BY 1

nocycle nocache;

insert into Angajati values (genereaza\_ang.nextval, 'Simion Petrica', null, 'spetrica81@gmail.com', 20000, to\_date('25-03-2021', 'DD-MM-YYYY'), 2);

insert into Angajati values (genereaza\_ang.nextval, 'Andrei Cristian', null, 'dumitrache@gmail.com', 3000, to\_date('30-05-2005', 'DD-MM-YYYY'), 1);

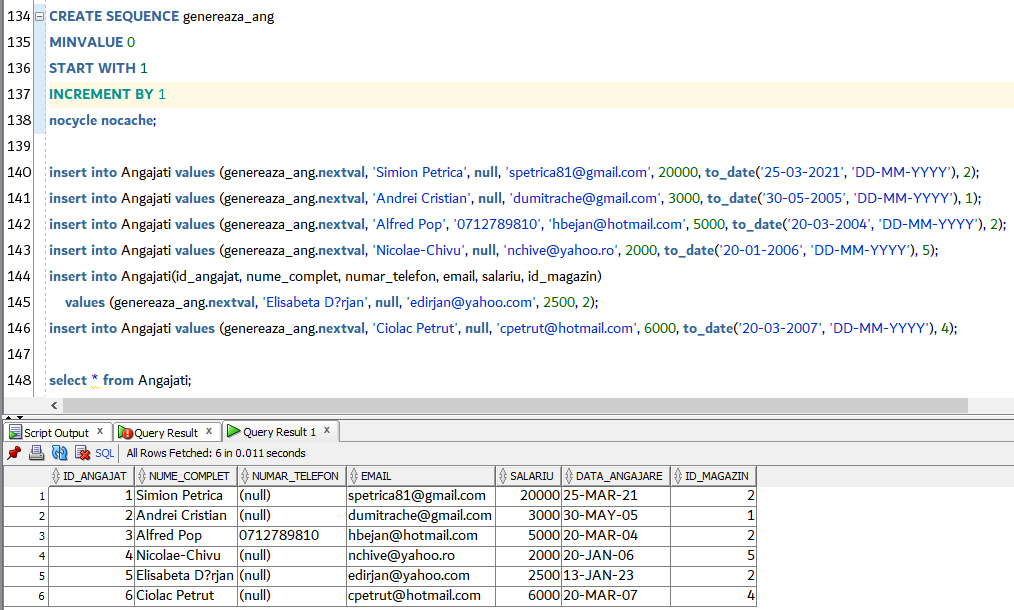
insert into Angajati values (genereaza\_ang.nextval, 'Alfred Pop', '0712789810', 'hbejan@hotmail.com', 5000, to\_date('20-03-2004', 'DD-MM-YYYY'), 2);

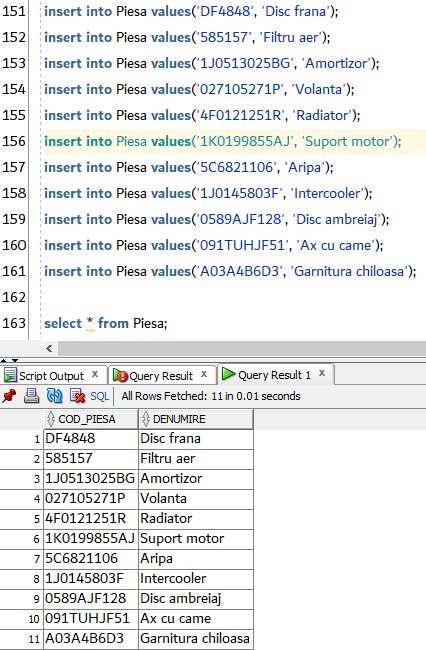
insert into Angajati values (genereaza\_ang.nextval, 'Nicolae-Chivu', null, 'nchive@yahoo.ro', 2000, to\_date('20-01-2006', 'DD-MM-YYYY'), 5);

insert into Angajati(id\_angajat, nume\_complet, numar\_telefon, email, salariu, id\_magazin)

values (genereaza\_ang.nextval, 'Elisabeta D?rjan', null, 'edirjan@yahoo.com', 2500, 2);

insert into Angajati values (genereaza\_ang.nextval, 'Ciolac Petrut', null, 'cpetrut@hotmail.com', 6000, to\_date('20-03-2007', 'DD-MM-YYYY'), 4);

****

****

insert into Piesa values('DF4848', 'Disc frana');

insert into Piesa values('585157', 'Filtru aer');

insert into Piesa values('1J0513025BG', 'Amortizor');

insert into Piesa values('027105271P', 'Volanta');

insert into Piesa values('4F0121251R', 'Radiator');

insert into Piesa values('1K0199855AJ', 'Suport motor');

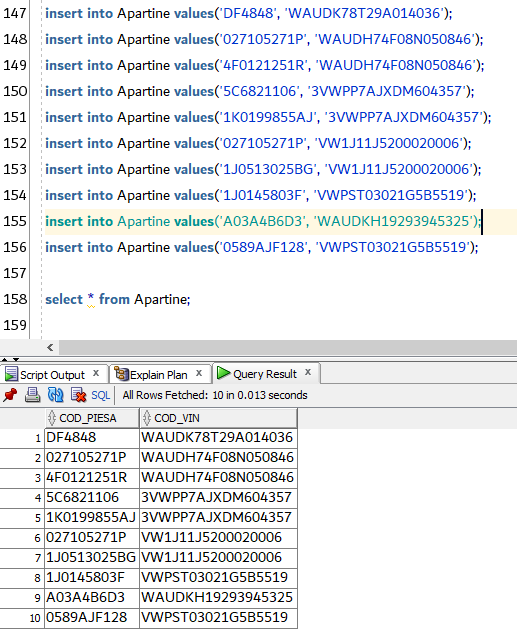
insert into Piesa values('5C6821106', 'Aripa');

insert into Piesa values('1J0145803F', 'Intercooler');

insert into Piesa values('0589AJF128', 'Disc ambreiaj');

insert into Piesa values('091TUHJF51', 'Ax cu came');

insert into Piesa values('A03A4B6D3', 'Garnitura chiloasa');

****

insert into Apartine values('DF4848', 'WAUDK78T29A014036');

insert into Apartine values('027105271P', 'WAUDH74F08N050846');

insert into Apartine values('4F0121251R', 'WAUDH74F08N050846');

insert into Apartine values('5C6821106', '3VWPP7AJXDM604357');

insert into Apartine values('1K0199855AJ', '3VWPP7AJXDM604357');

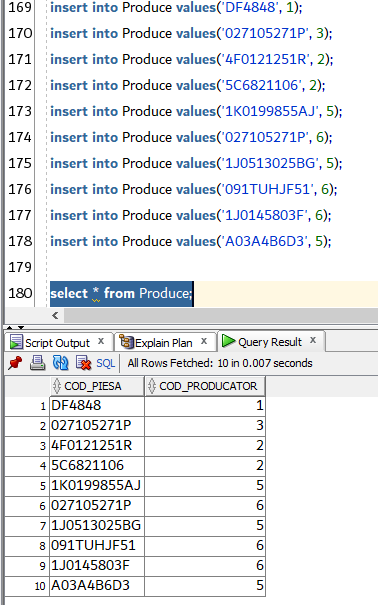
insert into Apartine values('027105271P', 'VW1J11J5200020006');

insert into Apartine values('1J0513025BG', 'VW1J11J5200020006');

insert into Apartine values('1J0145803F', 'VWPST03021G5B5519');

insert into Apartine values('A03A4B6D3', 'WAUDKH19293945325');

insert into Apartine values('0589AJF128', 'VWPST03021G5B5519');

****

insert into Produce values('DF4848', 1);

insert into Produce values('027105271P', 3);

insert into Produce values('4F0121251R', 2);

insert into Produce values('5C6821106', 2);

insert into Produce values('1K0199855AJ', 5);

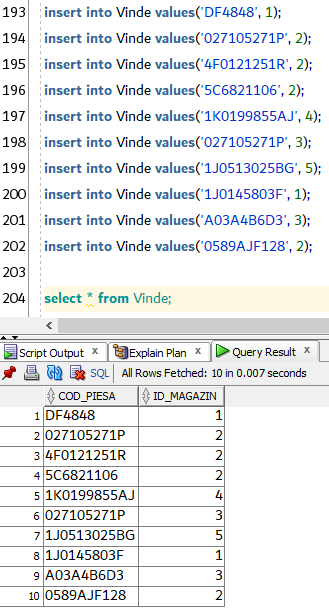
insert into Produce values('027105271P', 6);

insert into Produce values('1J0513025BG', 5);

insert into Produce values('091TUHJF51', 6);

insert into Produce values('1J0145803F', 6);

insert into Produce values('A03A4B6D3', 5);

****

insert into Vinde values('DF4848', 1);

insert into Vinde values('027105271P', 2);

insert into Vinde values('4F0121251R', 2);

insert into Vinde values('5C6821106', 2);

insert into Vinde values('1K0199855AJ', 4);

insert into Vinde values('027105271P', 3);

insert into Vinde values('1J0513025BG', 5);

insert into Vinde values('1J0145803F', 1);

insert into Vinde values('A03A4B6D3', 3);

insert into Vinde values('0589AJF128', 2);

**6. Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat independent care să utilizeze două tipuri diferite de colecții studiate. Apelați subprogramul.**

**R: Creati o procedura in cadrul careia sa afisati, pentru fiecare magazin, id-ul acestuia si denumirea pieselor existente in el.**

**- Pentru rezolvare am folosit VARRAY si Index-By Table.**

**CREATE OR REPLACE PROCEDURE afisare\_piese**

**IS**

**TYPE vector IS VARRAY(100) OF Magazin.id\_magazin%TYPE;**

**TYPE vector\_auxiliar IS VARRAY(100) OF piesa.denumire%TYPE;**

**TYPE ibtable IS TABLE OF vector\_auxiliar INDEX BY PLS\_INTEGER;**

**arr\_mag vector;**

**arr\_piese ibtable;**

**BEGIN**

**SELECT id\_magazin**

**BULK COLLECT INTO arr\_mag**

**FROM magazin;**

**FOR i in 1..arr\_mag.count LOOP**

**SELECT denumire**

**BULK COLLECT INTO arr\_piese(i)**

**FROM piesa**

**JOIN vinde USING (cod\_piesa)**

**WHERE id\_magazin = arr\_mag(i);**

**END LOOP;**

**FOR i in 1..arr\_mag.count LOOP**

**DBMS\_OUTPUT.PUT\_LINE('Cod magazin: ' || arr\_mag(i));**

**FOR j in 1..arr\_piese(i).count loop**

**DBMS\_OUTPUT.PUT\_LINE(arr\_piese(i)(j));**

**END LOOP;**

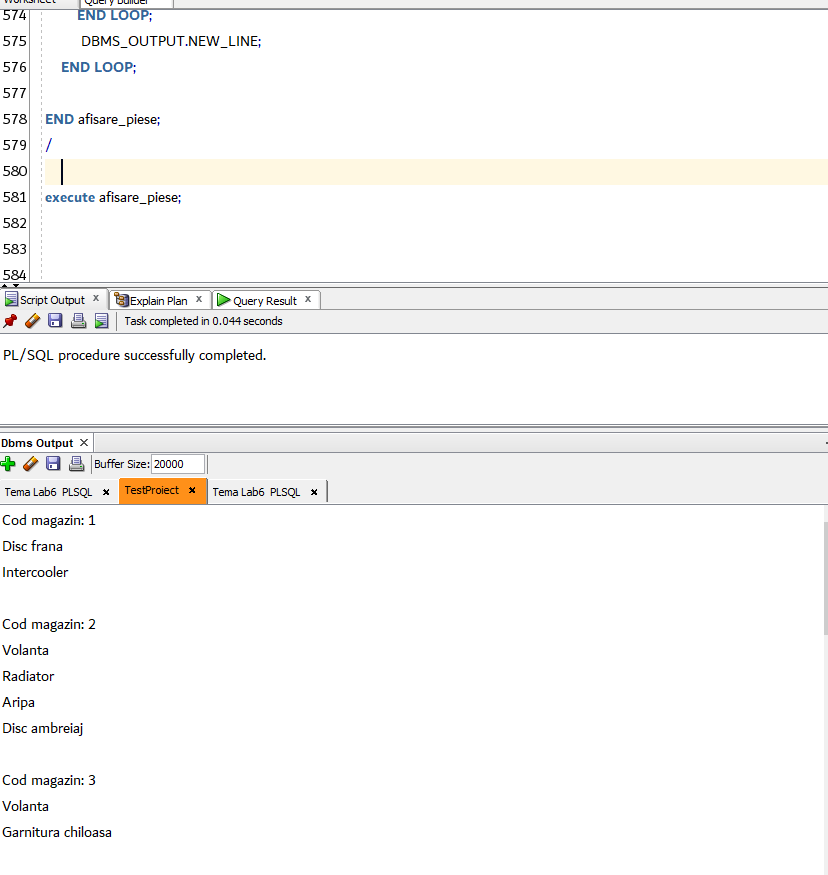
**DBMS\_OUTPUT.NEW\_LINE;**

**END LOOP;**

**END afisare\_piese;**

**/**

**execute afisare\_piese;**

****

**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. Apelați subprogramul.**

**R: Creati o procedurea prin care sa afisati, pentru fiecare magazin numele acestuia, cat si numele complete si datele de contact (numar telefon si email) ale angajatilor care lucreaza in respectivul magazin. In cazul in care nu exista angajati se va afisa un mesaj specific.**

**- Pentru rezolvare am folosit cursor explicit si ciclu cursor (cel de-al doilea fiind parametrizat)**

**CREATE OR REPLACE PROCEDURE afisare\_angajati\_dga IS**

**CURSOR magazine IS**

**SELECT id\_magazin, denumire**

**FROM magazin;**

**CURSOR angajati (magazin magazin.id\_magazin%TYPE) IS**

**SELECT nume\_complet, email, numar\_telefon**

**FROM angajati**

**WHERE id\_magazin = magazin;**

**idMagazin magazin.id\_magazin%TYPE;**

**numeMagazin magazin.denumire%TYPE;**

**counter NUMBER;**

**BEGIN**

**OPEN magazine;**

**LOOP**

**FETCH magazine into idMagazin, numeMagazin;**

**EXIT WHEN magazine%NOTFOUND;**

**DBMS\_OUTPUT.PUT\_LINE('Magazinul cu ID-ul ' || idMagazin || ' si numele ' || numeMagazin || ' are angajatii:');**

**counter := 0;**

**FOR i in angajati(idMagazin) LOOP**

**DBMS\_OUTPUT.PUT\_LINE(i.nume\_complet || ' ' || i.numar\_telefon || ' ' || i.email);**

**counter := counter + 1;**

**END LOOP;**

**IF counter = 0**

**THEN**

**DBMS\_OUTPUT.PUT\_LINE('Nu exista angajati in acest magazin!');**

**END IF;**

**DBMS\_OUTPUT.NEW\_LINE;**

**END LOOP;**

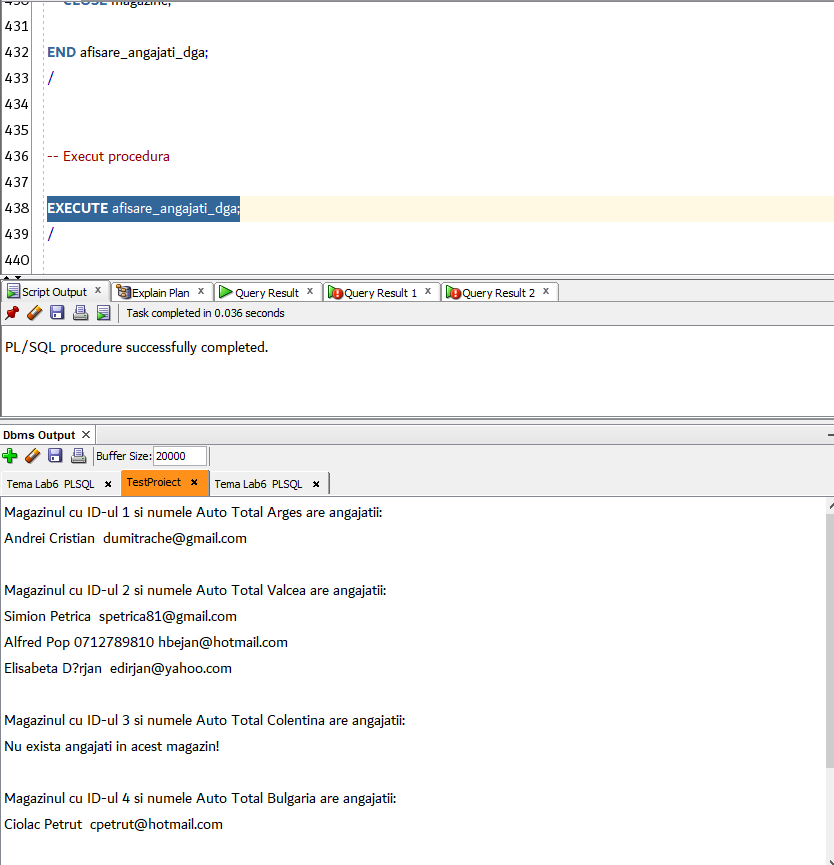
**CLOSE magazine;**

**END afisare\_angajati\_dga;**

**/**

**-- Execut procedura**

**EXECUTE afisare\_angajati\_dga;**

****

**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. Apelați subprogramul astfel încât să evidențiați toate cazurile tratate.**

**R: Scrieti o functie PLSQL care sa returneze ID-ul angajatului care lucreaza la un magazin dintr-o țară dată ca parametru si nu are setat un numar de telefon.**

**CREATE OR REPLACE FUNCTION numar\_angajati\_dga (tara Tari.cod\_tara%TYPE)**

**RETURN NUMBER IS**

**id\_ang Angajati.id\_angajat%type;**

**BEGIN**

**SELECT id\_angajat**

**INTO id\_ang**

**FROM angajati a**

**JOIN magazin m on (m.id\_magazin = a.id\_magazin)**

**JOIN locatie l on (l.id\_locatie = m.id\_locatie)**

**WHERE upper(l.cod\_tara) = upper(tara)**

**AND numar\_telefon is null;**

**RETURN id\_ang;**

**EXCEPTION**

**WHEN NO\_DATA\_FOUND**

**THEN**

**RAISE\_APPLICATION\_ERROR(-20001, 'Nu exista niciun angajat care sa corespunda cautarii!');**

**WHEN TOO\_MANY\_ROWS**

**THEN**

**RAISE\_APPLICATION\_ERROR(-20002, 'Sunt prea multi angajati care corespund!');**

**END numar\_angajati\_dga;**

**/**

****

**-- Facem join între Angajați, Magazin și Locatie. Verificăm cod\_tara din Locatie, comparativ cu cel dat ca parametru.**

**-- Și reținem rezultatul din SELECT în id\_ang. Aici pot apărea câteva cazuri nefericite:**

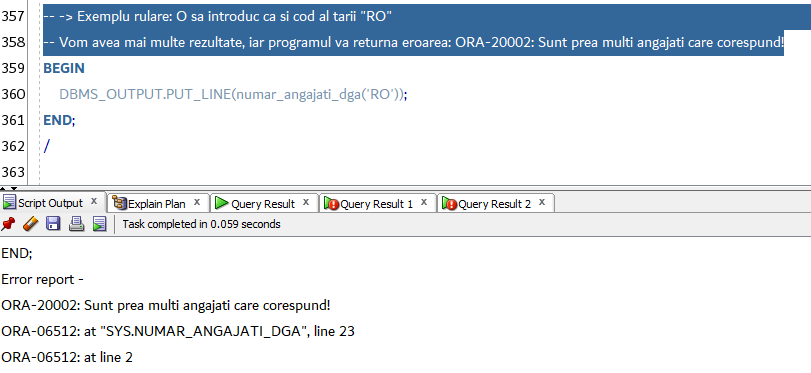
**1. Există mai mulți angajați în această situație. Pentru asta avem excepția TOO\_MANY\_ROWS, care afișează un mesaj specific.**

**2. Nu există niciun angajat care corespuned, asta poate fi datorată și introducerii unei țări care nu există.**

**Tratăm prin NO\_DATA\_FOUND.**

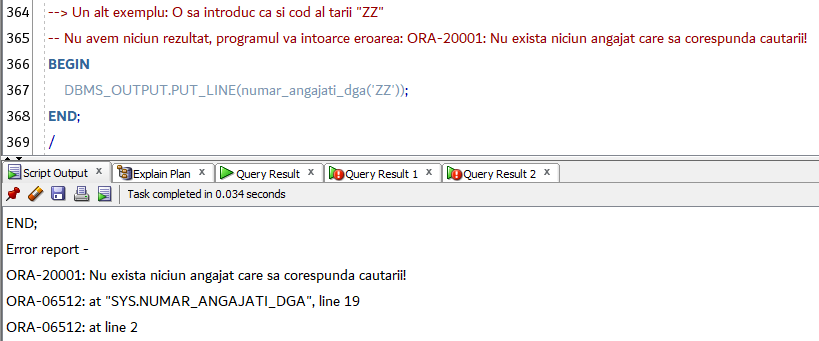
**1. -- -> Exemplu rulare: O sa introduc ca si cod al tarii "RO"**

**-- Vom avea mai multe rezultate, iar programul va returna eroarea: ORA-20002: Sunt prea multi angajati care corespund!**

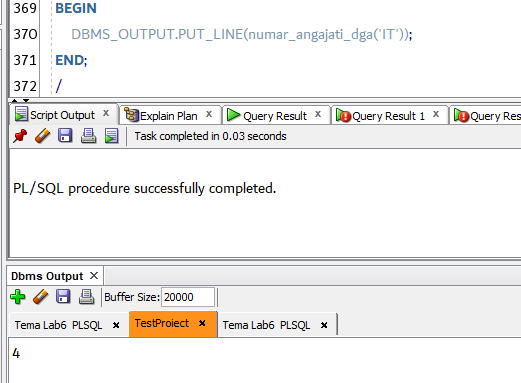
****

**2. --> Un alt exemplu: O sa introduc ca si cod al tarii "ZZ"**

**-- Nu avem niciun rezultat, programul va intoarce eroarea: ORA-20001: Nu exista niciun angajat care sa corespunda cautarii!**

****

**-- Acum vine cazul fericit, când totul este în regulă.**

****

**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.**

**R: Scrieti o procedura PLSQL care sa afiseze numele producaturilui ale carui piese sunt vandute in magazinele dintr-o tara data ca parametru si are lungimea numelui data tot ca parametru**

**CREATE OR REPLACE PROCEDURE afisare\_producator(tara Tari.cod\_tara%TYPE, lungime NUMBER)**

**IS**

**den\_prod Producator.denumire%TYPE;**

**BEGIN**

**SELECT unique producator.denumire**

**INTO den\_prod**

**FROM producator**

**JOIN produce USING (cod\_producator)**

**JOIN vinde USING (cod\_piesa)**

**JOIN magazin USING (id\_magazin)**

**JOIN locatie USING (id\_locatie)**

**WHERE cod\_tara = 'RO'**

**AND LENGTH(producator.denumire) = lungime;**

**DBMS\_OUTPUT.PUT\_LINE('Producatorul care corespunde are denumirea: ' || den\_prod);**

**EXCEPTION**

**WHEN TOO\_MANY\_ROWS**

**THEN**

**RAISE\_APPLICATION\_ERROR(-20010, 'Prea multi producatori corespund cerintelor!');**

**WHEN NO\_DATA\_FOUND**

**THEN**

**RAISE\_APPLICATION\_ERROR(-20011, 'Nu a fost gasit niciun producator care sa corespunda cerintelor!');**

**END afisare\_producator;**

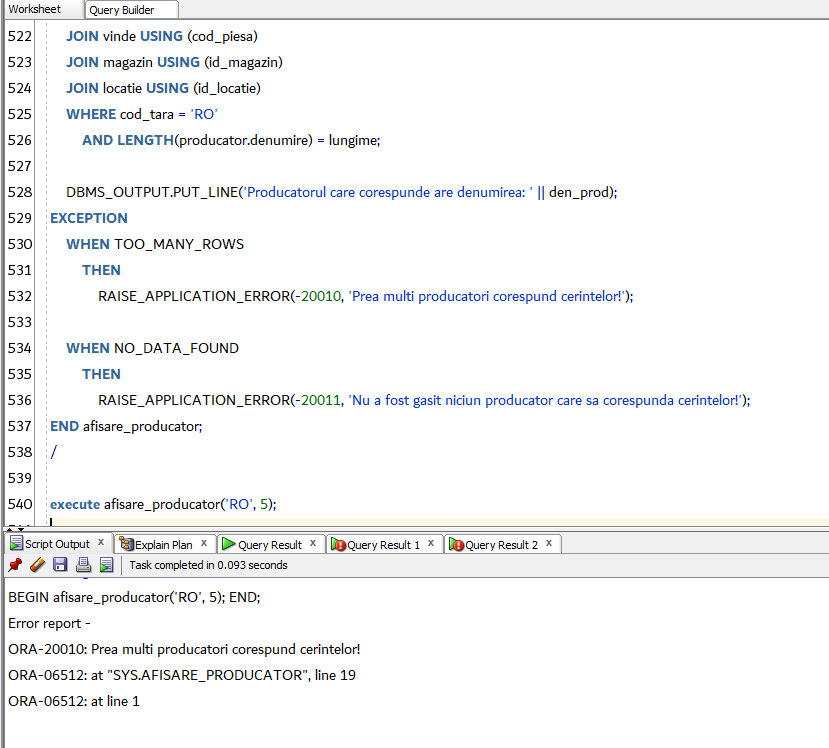
**/**

****

**Excepțiile care pot apărea sunt:**

**1. TOO\_MANY\_ROWS**

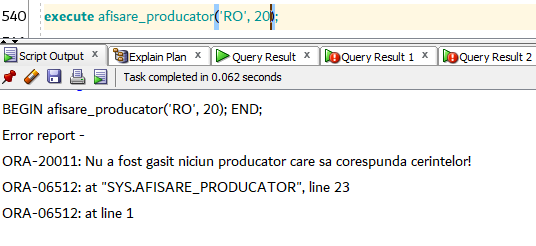
**-- Se întâmplă asta pentru că în România sunt mai ulți producători care oferă piese și au numele format din 5 litere, spre exemplu Valeo și Ridex.**

****

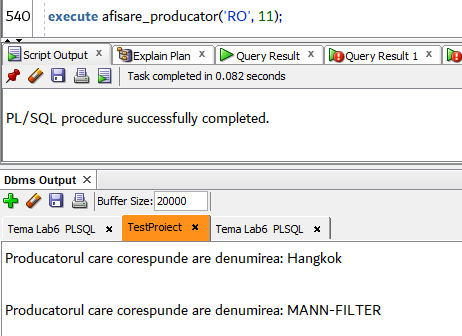
**2. NO\_DATA\_FOUND**

**- Apare când nu este niciun producător care să se încadreze în cerințe. Afișează un mesaj specific.**

**În exemplul meu, se observă că-mi afișează acest mesaj, dat fiind faptul că nu am niciun producător cu lungimea numelui de 20 de litere care să ofere piese magazinelor din România.**

****

**-- Acum vine cazul fericit, când nu avem probleme / excepții.**

****

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

**-- Creati un trigger LMD la nivel de comanda care sa nu permita adaugarea de noi piese in timpul programului de munca al angajatilor (se cunoaste ca un angajat lucreaza de luni pana vineri, intre orele 9 si 17).**

**CREATE OR REPLACE TRIGGER trigger\_piese**

**BEFORE INSERT ON piesa**

**DECLARE**

**BEGIN**

**IF (TO\_CHAR(SYSDATE, 'D') BETWEEN 2 AND 7) OR**

**(TO\_CHAR(SYSDATE, 'HH24') BETWEEN 9 AND 17)**

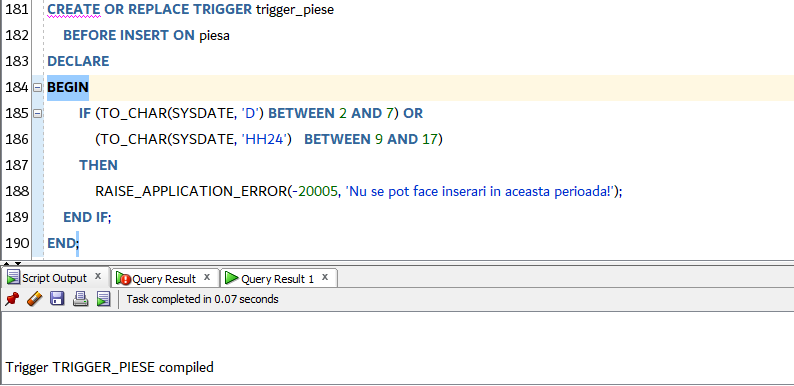
**THEN**

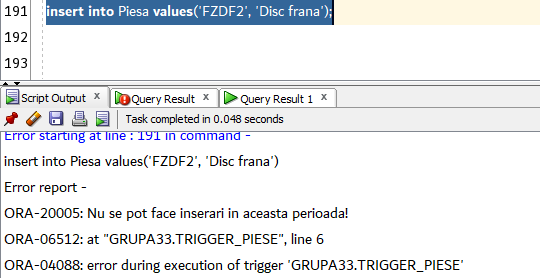
**RAISE\_APPLICATION\_ERROR(-20005, 'Nu se pot face inserari in aceasta perioada!');**

**END IF;**

**END;**

**/**

****

****

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

**-- Adăugați un trigger LMD la nivel de linie care esa nu permită, la modificarea salariului unui angajat, să se adauge un salariu mai mic decât cel pe economie (luăm 1900 ca salariu minim).**

**CREATE OR REPLACE TRIGGER previne\_salariu\_gresit**

**BEFORE UPDATE ON angajati**

**FOR EACH ROW**

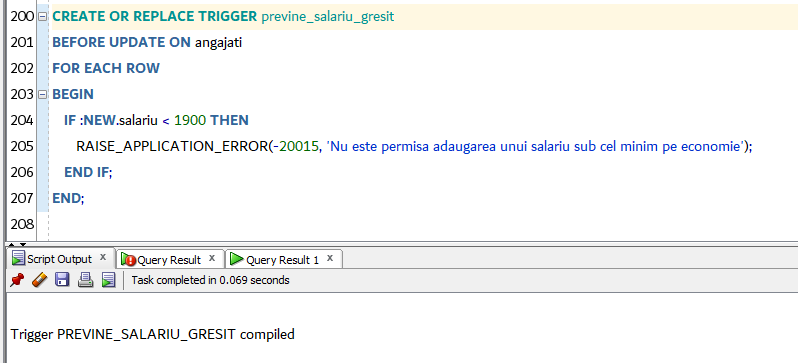
**BEGIN**

**IF :NEW.salariu < 1900 THEN**

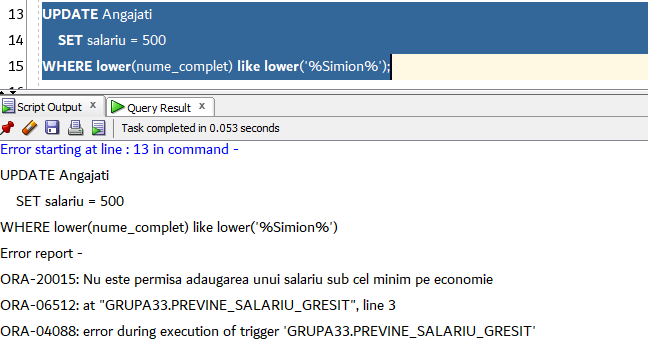
**RAISE\_APPLICATION\_ERROR(-20015, 'Nu este permisa adaugarea unui salariu sub cel minim pe economie');**

**END IF;**

**END;**

****

**- Aici un moment când este declanșat. Se observă că dacă încercăm să-i modificăm salariul angajatului care are ‘Simion’ în nume se declanșează triggerul și nu ne permite.**

****

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

**-- Creati un trigger LDD care sa nu permita modificarea (alter/create/drop) in timpul weekendului. Se presupune ca in cazul in care apar greseli nu ar avea cine sa le sesizeze, fiind weekend.**

**CREATE OR REPLACE TRIGGER previne\_modificari\_nepotrivite**

**AFTER CREATE OR DROP OR ALTER ON SCHEMA**

**BEGIN**

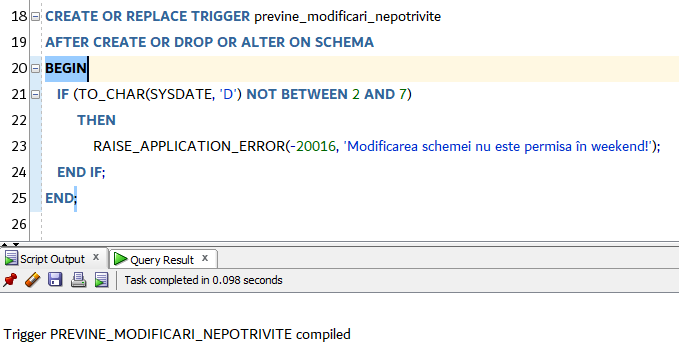
**IF (TO\_CHAR(SYSDATE, 'D') NOT BETWEEN 2 AND 7)**

**THEN**

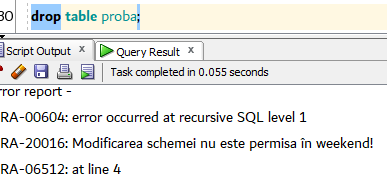
**RAISE\_APPLICATION\_ERROR(-20016, 'Modificarea schemei nu este permisa în weekend!');**

**END IF;**

**END;**

****

**-- Incerc un drop pe un tabel facut de proba:**

****

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

**CREATE OR REPLACE PACKAGE pachet\_rezolvari**

**AS**

**PROCEDURE afisare\_piese;**

**PROCEDURE afisare\_angajati\_dga;**

**FUNCTION numar\_angajati\_dga (tara Tari.cod\_tara%TYPE) RETURN NUMBER;**

**PROCEDURE afisare\_producator(tara Tari.cod\_tara%TYPE, lungime NUMBER);**

**END pachet\_rezolvari;**

**/**

**CREATE OR REPLACE PACKAGE BODY pachet\_rezolvari**

**IS**

**PROCEDURE afisare\_piese IS**

**TYPE vector IS VARRAY(100) OF Magazin.id\_magazin%TYPE;**

**TYPE vector\_auxiliar IS VARRAY(100) OF piesa.denumire%TYPE;**

**TYPE ibtable IS TABLE OF vector\_auxiliar INDEX BY PLS\_INTEGER;**

**arr\_mag vector;**

**arr\_piese ibtable;**

**BEGIN**

**SELECT id\_magazin**

**BULK COLLECT INTO arr\_mag**

**FROM magazin;**

**FOR i in 1..arr\_mag.count LOOP**

**SELECT denumire**

**BULK COLLECT INTO arr\_piese(i)**

**FROM piesa**

**JOIN vinde USING (cod\_piesa)**

**WHERE id\_magazin = arr\_mag(i);**

**END LOOP;**

**FOR i in 1..arr\_mag.count LOOP**

**DBMS\_OUTPUT.PUT\_LINE('Cod magazin: ' || arr\_mag(i));**

**FOR j in 1..arr\_piese(i).count loop**

**DBMS\_OUTPUT.PUT\_LINE(arr\_piese(i)(j));**

**END LOOP;**

**DBMS\_OUTPUT.NEW\_LINE;**

**END LOOP;**

**END afisare\_piese;**

**PROCEDURE afisare\_angajati\_dga IS**

**CURSOR magazine IS**

**SELECT id\_magazin, denumire**

**FROM magazin;**

**CURSOR angajati (magazin magazin.id\_magazin%TYPE) IS**

**SELECT nume\_complet, email, numar\_telefon**

**FROM angajati**

**WHERE id\_magazin = magazin;**

**idMagazin magazin.id\_magazin%TYPE;**

**numeMagazin magazin.denumire%TYPE;**

**counter NUMBER;**

**BEGIN**

**OPEN magazine;**

**LOOP**

**FETCH magazine into idMagazin, numeMagazin;**

**EXIT WHEN magazine%NOTFOUND;**

**DBMS\_OUTPUT.PUT\_LINE('Magazinul cu ID-ul ' || idMagazin || ' si numele ' || numeMagazin || ' are angajatii:');**

**counter := 0;**

**FOR i in angajati(idMagazin) LOOP**

**DBMS\_OUTPUT.PUT\_LINE(i.nume\_complet || ' ' || i.numar\_telefon || ' ' || i.email);**

**counter := counter + 1;**

**END LOOP;**

**IF counter = 0**

**THEN**

**DBMS\_OUTPUT.PUT\_LINE('Nu exista angajati in acest magazin!');**

**END IF;**

**DBMS\_OUTPUT.NEW\_LINE;**

**END LOOP;**

**CLOSE magazine;**

**END afisare\_angajati\_dga;**

**FUNCTION numar\_angajati\_dga (tara Tari.cod\_tara%TYPE)**

**RETURN NUMBER IS**

**id\_ang Angajati.id\_angajat%type;**

**BEGIN**

**SELECT id\_angajat**

**INTO id\_ang**

**FROM angajati a**

**JOIN magazin m on (m.id\_magazin = a.id\_magazin)**

**JOIN locatie l on (l.id\_locatie = m.id\_locatie)**

**WHERE upper(l.cod\_tara) = upper(tara)**

**AND numar\_telefon is null;**

**RETURN id\_ang;**

**EXCEPTION**

**WHEN NO\_DATA\_FOUND**

**THEN**

**RAISE\_APPLICATION\_ERROR(-20001, 'Nu exista niciun angajat care sa corespunda cautarii!');**

**WHEN TOO\_MANY\_ROWS**

**THEN**

**RAISE\_APPLICATION\_ERROR(-20002, 'Sunt prea multi angajati care corespund!');**

**END numar\_angajati\_dga;**

**PROCEDURE afisare\_producator(tara Tari.cod\_tara%TYPE, lungime NUMBER)**

**IS**

**den\_prod Producator.denumire%TYPE;**

**BEGIN**

**SELECT unique producator.denumire**

**INTO den\_prod**

**FROM producator**

**JOIN produce USING (cod\_producator)**

**JOIN vinde USING (cod\_piesa)**

**JOIN magazin USING (id\_magazin)**

**JOIN locatie USING (id\_locatie)**

**WHERE cod\_tara = 'RO'**

**AND LENGTH(producator.denumire) = lungime;**

**DBMS\_OUTPUT.PUT\_LINE('Producatorul care corespunde are denumirea: ' || den\_prod);**

**EXCEPTION**

**WHEN TOO\_MANY\_ROWS**

**THEN**

**RAISE\_APPLICATION\_ERROR(-20010, 'Prea multi producatori corespund cerintelor!');**

**WHEN NO\_DATA\_FOUND**

**THEN**

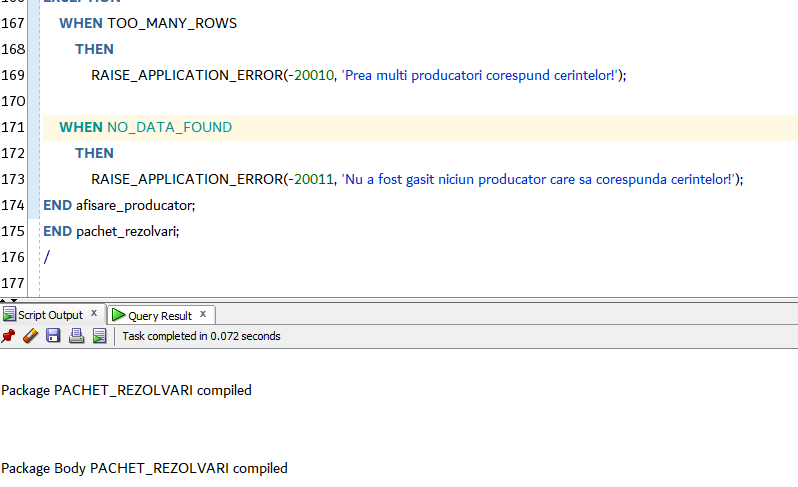
**RAISE\_APPLICATION\_ERROR(-20011, 'Nu a fost gasit niciun producator care sa corespunda cerintelor!');**

**END afisare\_producator;**

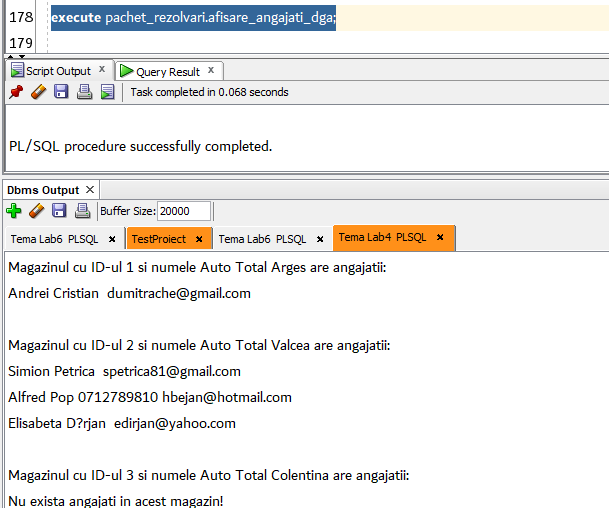
**END pachet\_rezolvari;**

**/**

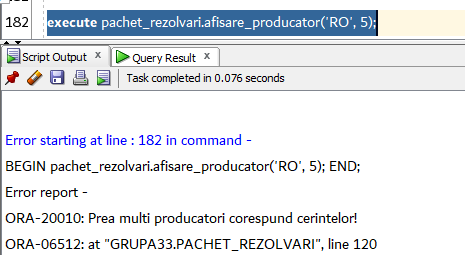
**-- Pur si simplu am creat un pachet in care am adaugat functiile / procedurile de la exercitiile 6 – 9.**

****

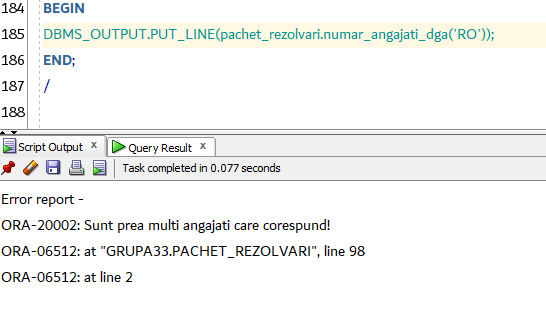
**- Acum testez subprogramele din pachet.**

****

**-- merge și partea de tratarea excepțiilor – am explicat la exercițiul 9 de ce apare această eroare**

****

**-- Înca un exemplu – am explicat la exercițiul 8 situația**

****