Sisteme de Gestiune a Bazelor de Date

Proiect Final

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

Andrei este un pasionat al mediului audio-video și dorește să transforme pasiunea sa într-o afacere, sub forma unui magazin online. Categoriile principale de produse aflate la vânzare pe site sunt microfoane si căști.

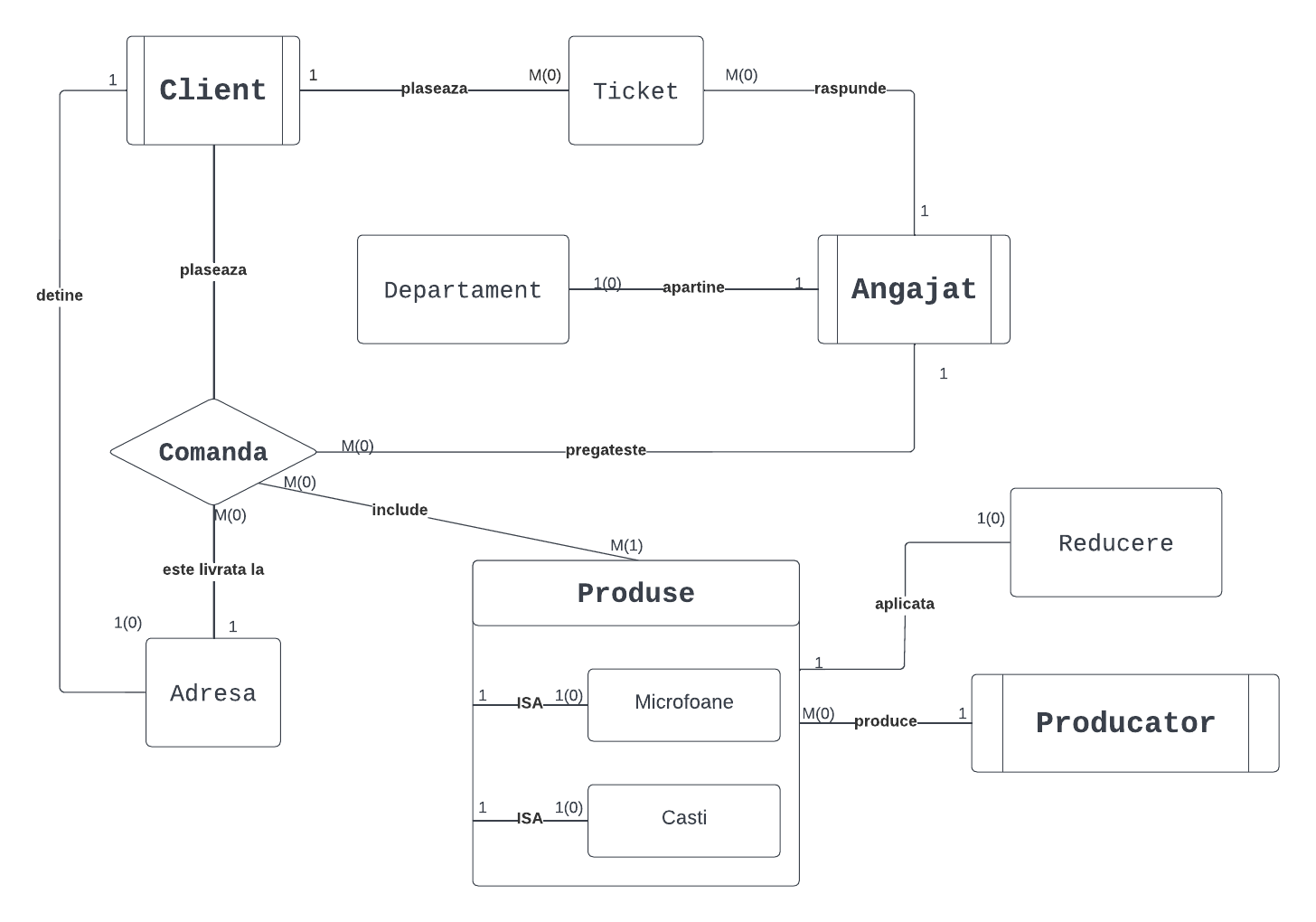
Pe pagina site-ului, clienții vor trebui să își creeze un cont pentru a putea plasa o comandă. Contul de client va reține numele si prenumele, data creării contului și adresa principala de livrare.

Clienții pot plasa pe site întrebări despre diversele produse sau servicii, iar fiecare astfel de ticket va fi atribuit unui angajat.

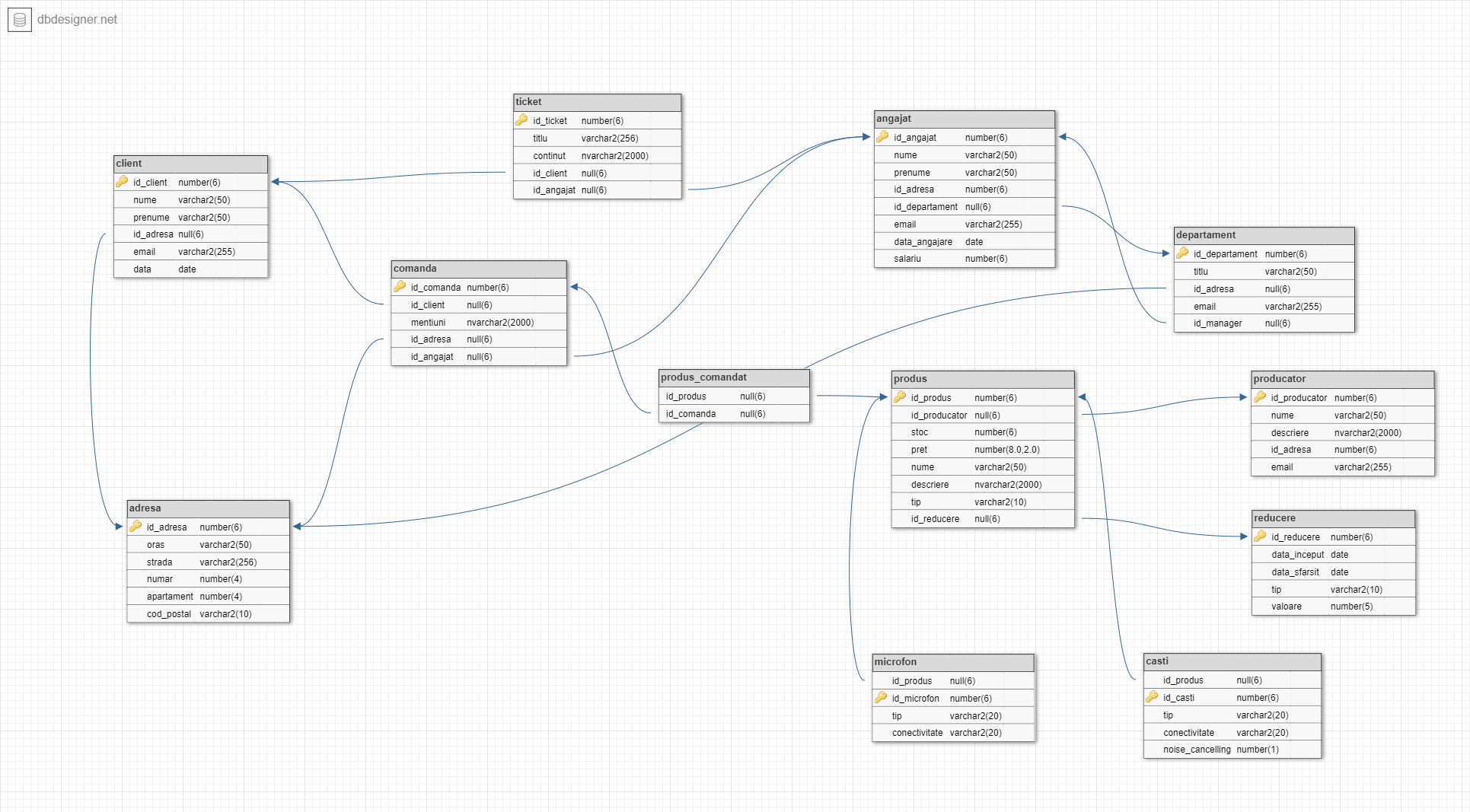
Totodată, o comandă plasată va fi atribuită unui angajat din departamentul „Comenzi” care o va împacheta și trimite.

O adresă reține țara, orașul, strada, numărul și alte informații necesare.

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

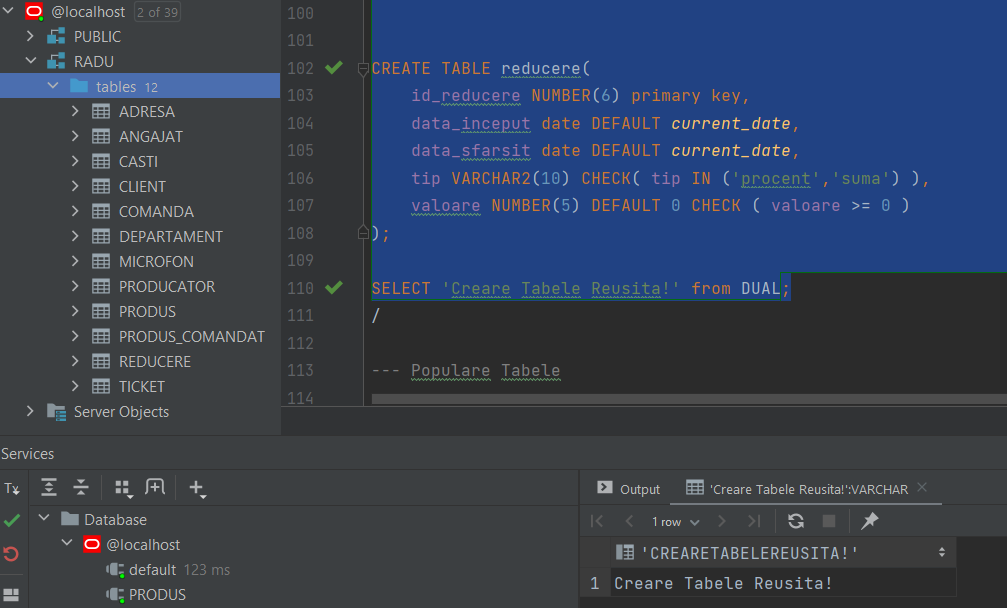


3. **Pornind de la diagrama entitate-relație realizați diagrama conceptuală a modelului propus, integrând toate atributele necesare.**



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

CREATE TABLE client(  
 id\_client NUMBER(6) primary key,  
 nume VARCHAR2(50),  
 prenume VARCHAR2(50) NOT NULL,  
 id\_adresa NUMBER(6),  
 email VARCHAR2(255) NOT NULL,  
 data date DEFAULT *current\_date*);  
  
  
CREATE TABLE ticket(  
 id\_ticket NUMBER(6) primary key,  
 titlu VARCHAR2(256) NOT NULL,  
 continut NVARCHAR2(2000) NOT NULL,  
 id\_client NUMBER(6) NOT NULL,  
 id\_angajat NUMBER(6)  
);  
  
  
CREATE TABLE adresa(  
 id\_adresa NUMBER(6) primary key,  
 oras VARCHAR2(50) NOT NULL,  
 strada VARCHAR2(256) NOT NULL,  
 numar NUMBER(4) NOT NULL,  
 apartament NUMBER(4),  
 cod\_postal VARCHAR2(10) NOT NULL  
);  
  
CREATE TABLE angajat(  
 id\_angajat NUMBER(6) primary key,  
 nume VARCHAR2(50) NOT NULL,  
 prenume VARCHAR2(50) NOT NULL,  
 id\_adresa NUMBER(6) NOT NULL,  
 id\_departament NUMBER(6),  
 email VARCHAR2(255) NOT NULL,  
 data\_angajare date DEFAULT *current\_date*,  
 salariu NUMBER(6) DEFAULT 3000  
);  
  
CREATE TABLE departament(  
 id\_departament NUMBER(6) primary key,  
 titlu VARCHAR2(50) NOT NULL,  
 id\_adresa NUMBER(6),  
 email VARCHAR2(255) NOT NULL,  
 id\_manager NUMBER(6) DEFAULT 1  
);  
  
  
CREATE TABLE producator(  
 id\_producator NUMBER(6) primary key,  
 nume VARCHAR2(50) NOT NULL,  
 descriere NVARCHAR2(2000),  
 id\_adresa NUMBER(6) ,  
 email VARCHAR2(255) NOT NULL  
);  
  
CREATE TABLE produs(  
 id\_produs NUMBER(6) PRIMARY KEY,  
 id\_producator NUMBER(6) NOT NULL,  
 stoc NUMBER(6) CHECK ( stoc >= 0 ),  
 pret NUMBER(8,2) CHECK ( pret >= 0 ),  
 nume VARCHAR2(50) NOT NULL,  
 descriere NVARCHAR2(2000),  
 tip VARCHAR2(10) CHECK( tip IN ('microfon','casti') ),  
 id\_reducere NUMBER(6)  
);  
  
CREATE TABLE microfon(  
 id\_produs NUMBER(6) NOT NULL,  
 id\_microfon NUMBER(6) PRIMARY KEY,  
 tip VARCHAR2(20) CHECK( tip IN ('dinamic','condensator') ),  
 conectivitate VARCHAR2(20) CHECK( conectivitate IN ('XLR','USB', 'JACK') )  
);  
  
CREATE TABLE casti(  
 id\_produs NUMBER(6) NOT NULL,  
 id\_casti NUMBER(6) PRIMARY KEY,  
 tip VARCHAR2(20) CHECK( tip IN ('In Ear','Over Ear','On Ear') ),  
 conectivitate VARCHAR2(20) CHECK( conectivitate IN ('wireless','cu fir') ),  
 noise\_cancelling NUMBER(1)  
);  
  
  
  
CREATE TABLE produs\_comandat(  
 id\_produs NUMBER(6) NOT NULL,  
 id\_comanda NUMBER(6) NOT NULL,  
 PRIMARY KEY (id\_produs, id\_comanda)  
);  
  
CREATE TABLE comanda(  
 id\_comanda NUMBER(6) PRIMARY KEY,  
 id\_client NUMBER(6) NOT NULL,  
 mentiuni NVARCHAR2(2000),  
 id\_adresa NUMBER(6),  
 id\_angajat NUMBER(6)  
);  
  
  
CREATE TABLE reducere(  
 id\_reducere NUMBER(6) primary key,  
 data\_inceput date DEFAULT *current\_date*,  
 data\_sfarsit date DEFAULT *current\_date*,  
 tip VARCHAR2(10) CHECK( tip IN ('procent','suma') ),  
 valoare NUMBER(5) DEFAULT 0 CHECK ( valoare >= 0 )  
);  
  
SELECT 'Creare Tabele Reusita!' from DUAL;



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

INSERT ALL  
 INTO client (id\_client, prenume,nume, id\_adresa, email) VALUES (1,'Ion', 'Al Glanetasului', 1, 'IonSiPamantul@gmail.com')  
 INTO client (id\_client, prenume,nume, id\_adresa, email) VALUES (2,'Marcel', 'Voinea', 2, 'Marcel2387@7yahoo.com')  
 INTO client (id\_client, prenume,nume, id\_adresa, email) VALUES (3,'Horia', 'Scarlat', 3, 'pavelbartos@gmail.com')  
 INTO client (id\_client, prenume,nume, id\_adresa, email) VALUES (4,'Radu', 'Marius', 4, 'n-am@gmail.com')  
 INTO client (id\_client, prenume,nume, id\_adresa, email) VALUES (5,'John', 'Smith', 5, 'numeBasic@gmail.com')  
 INTO client (id\_client, prenume,nume, id\_adresa, email) VALUES (6,'Gabriel', 'Scarlat', 3, 'gabiscarlat22@gmail.com')  
SELECT \* FROM DUAL;  
  
INSERT ALL  
 INTO ticket (id\_ticket, id\_client,titlu, continut, ID\_ANGAJAT ) VALUES (10, 1, 'Cum cumpar?', 'Cum cumpar un produs?', 2)  
 INTO ticket (id\_ticket, id\_client,titlu, continut, ID\_ANGAJAT ) VALUES (4, 1, 'Cum platesc?', 'Cum platesc un produs?', NULL)  
 INTO ticket (id\_ticket, id\_client,titlu, continut, ID\_ANGAJAT ) VALUES (5, 1, 'Salut ma poti ajuta?', 'Salut, ma poti ajuta?', NULL)  
 INTO ticket (id\_ticket, id\_client,titlu, continut, ID\_ANGAJAT ) VALUES (12, 4, 'Locatie sediu', 'Unde se afla sediul?', 2)  
 INTO ticket (id\_ticket, id\_client,titlu, continut, ID\_ANGAJAT ) VALUES (9, 4, 'Plata', 'Ce modalitati de plata exista??', 2)  
 INTO reducere(id\_reducere, DATA\_INCEPUT,DATA\_SFARSIT, tip, VALOARE) VALUES (4,'15-JAN-2004', '15-JAN-2006', 'procent', 15)  
 INTO reducere(id\_reducere, DATA\_INCEPUT,DATA\_SFARSIT, tip, VALOARE) VALUES (5,'15-FEB-2022', '24-SEP-2023', 'suma', 240)  
 INTO reducere(id\_reducere, DATA\_INCEPUT,DATA\_SFARSIT, tip, VALOARE) VALUES (6,'15-OCT-2022', '23-NOV-2022', 'procent', 15)  
 INTO reducere(id\_reducere, DATA\_INCEPUT, tip, VALOARE) VALUES

(7, '25-MAR-2022', 'procent', 10)  
 INTO reducere(id\_reducere, DATA\_SFARSIT, tip, VALOARE) VALUES

(8, '1-AUG-2022', 'suma', 69)  
 INTO producator (ID\_PRODUCATOR, nume, ID\_ADRESA, email ) VALUES

(1, 'Audio-Technica', 0, 'contact@audiotechnica.com')  
 INTO producator (ID\_PRODUCATOR, nume, ID\_ADRESA, email ) VALUES

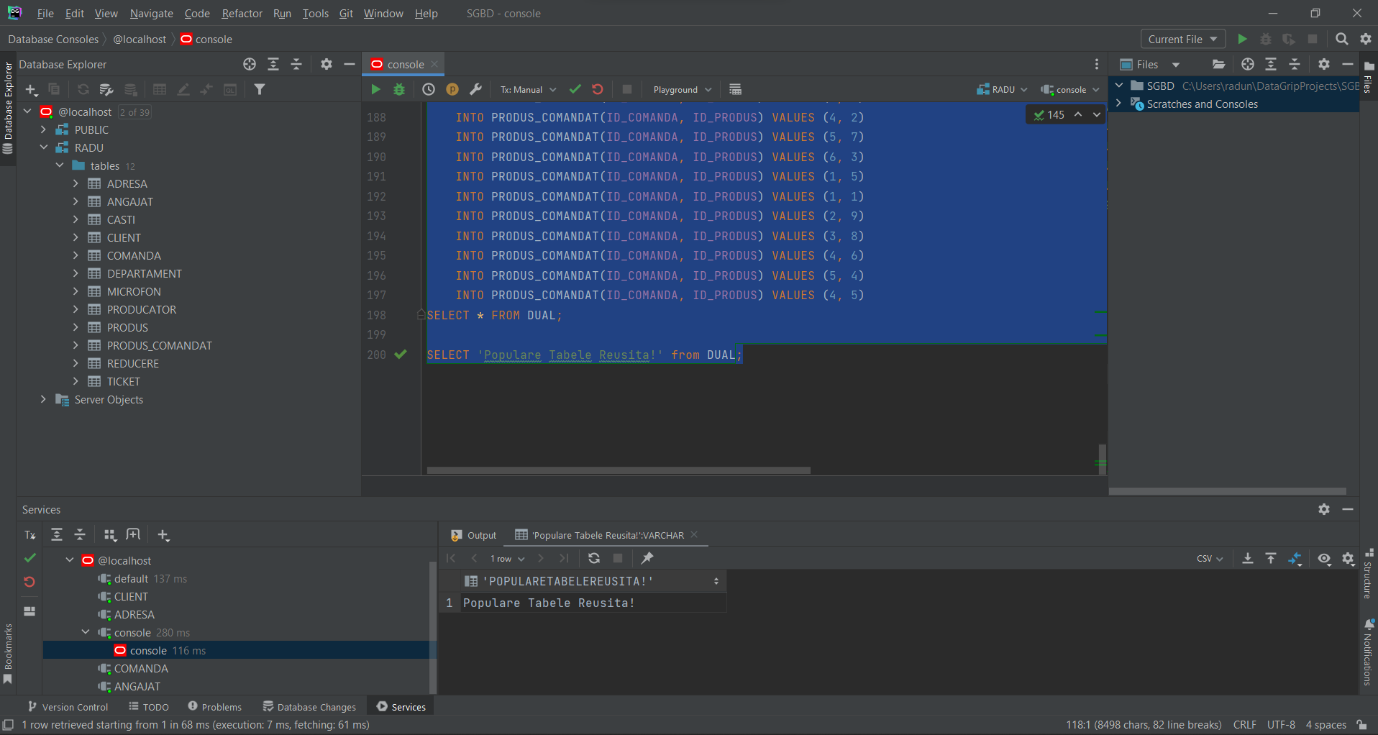
(2, 'SENNHEISER', 0, 'contact@sennheiser.com')  
 INTO producator (ID\_PRODUCATOR, nume, ID\_ADRESA, email ) VALUES

(3, 'Shure', 0, 'shure@shure.com')  
 INTO producator (ID\_PRODUCATOR, nume, ID\_ADRESA, email ) VALUES

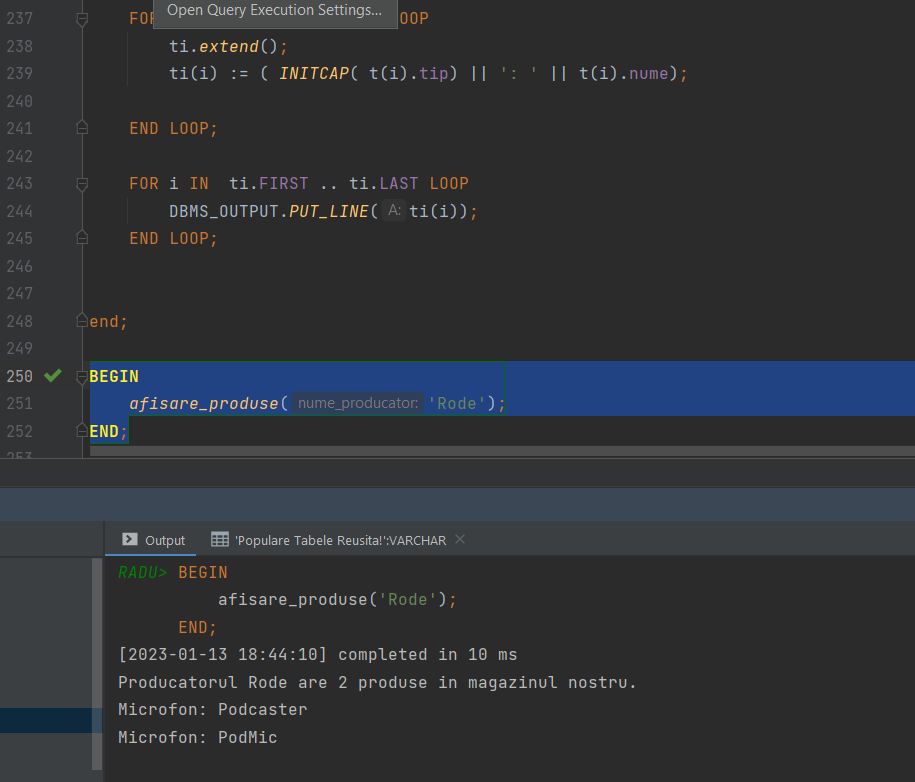
(4, 'Rode', 0, 'contact@rode.com')  
 INTO producator (ID\_PRODUCATOR, nume, ID\_ADRESA, email ) VALUES

(5, 'Behringer', 0, 'behringer@gmail.com')  
 INTO producator (ID\_PRODUCATOR, nume, ID\_ADRESA, email ) VALUES

(6, 'KZ Audio', 0, 'kz-audio@gmail.com')  
 INTO produs (id\_produs, ID\_PRODUCATOR,NUME, pret, TIP,ID\_REDUCERE, stoc ) VALUES (1, 4, 'Podcaster', 840.50, 'microfon', NULL, 5)  
 INTO produs (id\_produs, ID\_PRODUCATOR,NUME, pret, TIP,ID\_REDUCERE, stoc ) VALUES (2, 4, 'PodMic', 455, 'microfon', NULL, 10)  
 INTO produs (id\_produs, ID\_PRODUCATOR,NUME, pret, TIP,ID\_REDUCERE, stoc ) VALUES (3, 3, 'SM58',400, 'microfon', NULL, 15)  
 INTO produs (id\_produs, ID\_PRODUCATOR,NUME, pret, TIP,ID\_REDUCERE, stoc ) VALUES (4, 2, 'HD 600', 1530, 'casti', NULL, 14)  
 INTO produs (id\_produs, ID\_PRODUCATOR,NUME, pret, TIP,ID\_REDUCERE, stoc ) VALUES (5, 1, 'ATH-M50x', 579.40, 'casti', NULL, 100)  
 INTO produs (id\_produs, ID\_PRODUCATOR,NUME, pret, TIP,ID\_REDUCERE, stoc ) VALUES (6, 1, 'ATH-M20x', 249.80, 'casti', 5, 11)  
 INTO produs (id\_produs, ID\_PRODUCATOR,NUME, pret, TIP,ID\_REDUCERE, stoc ) VALUES (7, 6, 'ZR10', 335, 'casti', NULL, 29)  
 INTO produs (id\_produs, ID\_PRODUCATOR,NUME, pret, TIP,ID\_REDUCERE, stoc ) VALUES (8, 1, 'ATH-M30x', 239.67, 'casti', NULL, 42)  
 INTO produs (id\_produs, ID\_PRODUCATOR,NUME, pret, TIP,ID\_REDUCERE, stoc ) VALUES (9, 5, 'C-1U', 229.99, 'microfon', NULL, 50)  
 INTO produs (id\_produs, ID\_PRODUCATOR,NUME, pret, TIP,ID\_REDUCERE, stoc ) VALUES (10, 6, 'ZEX PRO', 230, 'casti', NULL, 22)  
 INTO microfon (id\_produs, ID\_MICROFON, TIP, conectivitate ) VALUES (9, 1, 'condensator', 'USB')  
 INTO microfon (id\_produs, ID\_MICROFON, TIP, conectivitate ) VALUES (3, 2, 'dinamic', 'XLR')  
 INTO microfon (id\_produs, ID\_MICROFON, TIP, conectivitate ) VALUES (2, 3, 'dinamic', 'XLR')  
 INTO microfon (id\_produs, ID\_MICROFON, TIP, conectivitate ) VALUES (1, 4, 'condensator', 'XLR')  
 INTO casti (id\_produs, id\_casti, TIP, conectivitate, NOISE\_CANCELLING ) VALUES (4, 1,'Over Ear', 'cu fir', 1)  
 INTO casti (id\_produs, id\_casti, TIP, conectivitate, NOISE\_CANCELLING ) VALUES (5, 2,'Over Ear', 'cu fir', 0)  
 INTO casti (id\_produs, id\_casti, TIP, conectivitate, NOISE\_CANCELLING ) VALUES (6, 3,'Over Ear', 'cu fir', 0)  
 INTO casti (id\_produs, id\_casti, TIP, conectivitate, NOISE\_CANCELLING ) VALUES (7, 4,'In Ear', 'wireless', 1)  
 INTO casti (id\_produs, id\_casti, TIP, conectivitate, NOISE\_CANCELLING ) VALUES (8, 5,'Over Ear', 'wireless', 0)  
 INTO casti (id\_produs, id\_casti, TIP, conectivitate, NOISE\_CANCELLING ) VALUES (10, 6,'In Ear', 'cu fir', 1)  
 INTO departament (ID\_DEPARTAMENT, TITLU, ID\_MANAGER, ID\_ADRESA, email) VALUES (1, 'Vanzari',2, 2, 'contact@showroom.ro')  
 INTO departament (ID\_DEPARTAMENT, TITLU, ID\_MANAGER, ID\_ADRESA, email) VALUES (2, 'Impachetare comenzi',1, 2, 'contact@showroom.ro')  
 INTO departament (ID\_DEPARTAMENT, TITLU, ID\_MANAGER, ID\_ADRESA, email) VALUES (3, 'Website',2, 3, 'contact@showroom.ro')  
 INTO departament (ID\_DEPARTAMENT, TITLU, ID\_MANAGER, ID\_ADRESA, email) VALUES (4, 'Showroom',1, 2, 'contact@showroom.ro')  
 INTO adresa (id\_adresa,oras, strada, numar, apartament, COD\_POSTAL) VALUES (1, 'Bucuresti','Alexandru Macedon', 14, 66, '071432')  
 INTO adresa (id\_adresa,oras, strada, numar, apartament, COD\_POSTAL) VALUES (2, 'Bucuresti','Soldan Ion enache', 14, 66, '32498')  
 INTO adresa (id\_adresa,oras, strada, numar, apartament, COD\_POSTAL) VALUES (3, 'Bucuresti','I C Bratianu', 14, 66, '139821')  
 INTO adresa (id\_adresa,oras, strada, numar, apartament, COD\_POSTAL) VALUES (4, 'Bucuresti','Iuliu Maniu', 66, NULL, '23873')  
 INTO adresa (id\_adresa,oras, strada, numar, apartament, COD\_POSTAL) VALUES (5, 'Bucuresti','Splaiul Independentei', 145, NULL, '071432')  
 INTO adresa (id\_adresa,oras, strada, numar, apartament, COD\_POSTAL) VALUES (6, 'Bucuresti','Straduintei', 12, 66, '091380')  
 INTO adresa (id\_adresa,oras, strada, numar, apartament, COD\_POSTAL) VALUES (7, 'Bucuresti','Aliorului', 5, 66, '231234')  
 INTO angajat (ID\_ANGAJAT, ID\_DEPARTAMENT, nume, prenume, ID\_ADRESA, SALARIU, email) VALUES (2, 1,'Andrei', 'Balea', 7, 7500, 'emailGeneric@gmail.com')  
 INTO angajat (ID\_ANGAJAT, ID\_DEPARTAMENT, nume, prenume, ID\_ADRESA, email) VALUES (1, 4,'Marian', 'Popovici', 7, 'emailGeneric@gmail.com')  
 INTO angajat (ID\_ANGAJAT, ID\_DEPARTAMENT, nume, prenume, ID\_ADRESA, SALARIU, email) VALUES (3, 3,'Theodor', 'Ionescu', 7, 1200, 'emailGeneric@gmail.com')  
 INTO angajat (ID\_ANGAJAT, ID\_DEPARTAMENT, nume, prenume, ID\_ADRESA, SALARIU, email) VALUES (4, 2,'Theodora', 'Apostol', 7, 4800, 'emailGeneric@gmail.com')  
 INTO angajat (ID\_ANGAJAT, ID\_DEPARTAMENT, nume, prenume, ID\_ADRESA, SALARIU, email) VALUES (5, 2,'Gabriel', 'Popescu', 7, 5500, 'emailGeneric@gmail.com')  
 INTO comanda (ID\_COMANDA, ID\_ADRESA, ID\_ANGAJAT, ID\_CLIENT, MENTIUNI) VALUES (1, 3,4, 1, 'Nu am nimic de declarat.')  
 INTO comanda (ID\_COMANDA, ID\_ADRESA, ID\_ANGAJAT, ID\_CLIENT, MENTIUNI) VALUES (2, 4,4, 2, NULL)  
 INTO comanda (ID\_COMANDA, ID\_ADRESA, ID\_ANGAJAT, ID\_CLIENT, MENTIUNI) VALUES (3, 5,4, 3, NULL)  
 INTO comanda (ID\_COMANDA, ID\_ADRESA, ID\_ANGAJAT, ID\_CLIENT, MENTIUNI) VALUES (4, 2,4, 2, NULL)  
 INTO comanda (ID\_COMANDA, ID\_ADRESA, ID\_ANGAJAT, ID\_CLIENT, MENTIUNI) VALUES (5, 3,4, 1, NULL)  
 INTO comanda (ID\_COMANDA, ID\_ADRESA, ID\_ANGAJAT, ID\_CLIENT, MENTIUNI) VALUES (6, 3,4, 1, 'Sa impachetati discret.')  
 INTO PRODUS\_COMANDAT(ID\_COMANDA, ID\_PRODUS) VALUES (1, 3)  
 INTO PRODUS\_COMANDAT(ID\_COMANDA, ID\_PRODUS) VALUES (2, 4)  
 INTO PRODUS\_COMANDAT(ID\_COMANDA, ID\_PRODUS) VALUES (3, 5)  
 INTO PRODUS\_COMANDAT(ID\_COMANDA, ID\_PRODUS) VALUES (4, 2)  
 INTO PRODUS\_COMANDAT(ID\_COMANDA, ID\_PRODUS) VALUES (5, 7)  
 INTO PRODUS\_COMANDAT(ID\_COMANDA, ID\_PRODUS) VALUES (6, 3)  
 INTO PRODUS\_COMANDAT(ID\_COMANDA, ID\_PRODUS) VALUES (1, 5)  
 INTO PRODUS\_COMANDAT(ID\_COMANDA, ID\_PRODUS) VALUES (1, 1)  
 INTO PRODUS\_COMANDAT(ID\_COMANDA, ID\_PRODUS) VALUES (2, 9)  
 INTO PRODUS\_COMANDAT(ID\_COMANDA, ID\_PRODUS) VALUES (3, 8)  
 INTO PRODUS\_COMANDAT(ID\_COMANDA, ID\_PRODUS) VALUES (4, 6)  
 INTO PRODUS\_COMANDAT(ID\_COMANDA, ID\_PRODUS) VALUES (5, 4)  
 INTO PRODUS\_COMANDAT(ID\_COMANDA, ID\_PRODUS) VALUES (4, 5)  
SELECT \* FROM DUAL;  
  
SELECT 'Populare Tabele Reusita!' from DUAL;

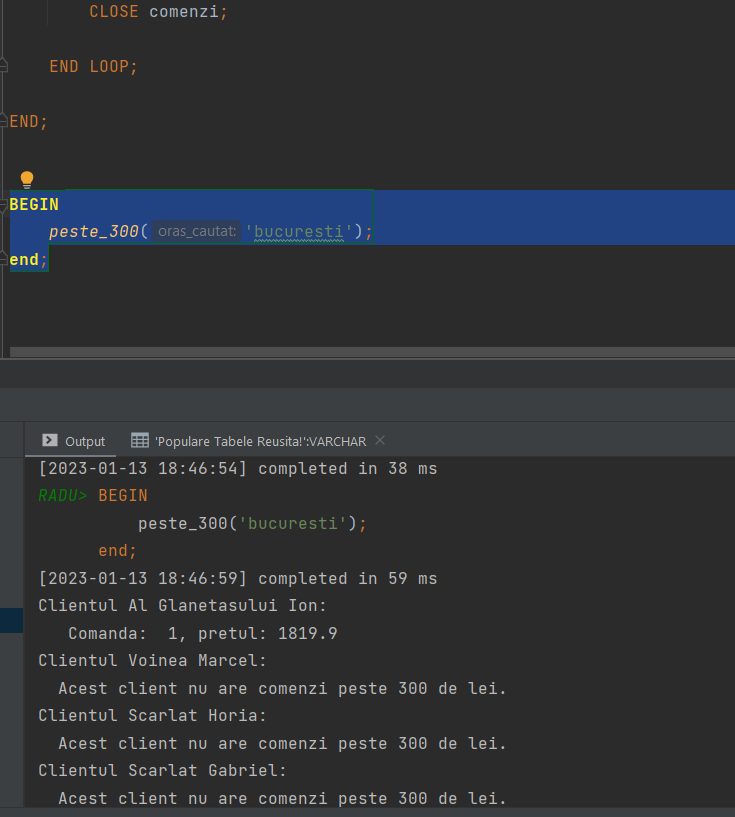


6. Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat independent care să utilizeze două tipuri de colecție studiate. Apelați subprogramul.

-- Un subprogram care afiseaza toate produsele unui producator dat ca parametru.  
-- Acestea sunt salvatate intr-o colectie de tip tabel indexat.  
CREATE OR REPLACE PROCEDURE *afisare\_produse* (nume\_producator producator.id\_producator%TYPE)  
AS  
 TYPE tabel\_produse IS TABLE OF produs%ROWTYPE INDEX BY PLS\_INTEGER;  
 t tabel\_produse;  
  
BEGIN  
 SELECT \* BULK COLLECT INTO t  
 FROM PRODUS  
 WHERE id\_producator = nume\_producator;  
  
 DBMS\_OUTPUT.*PUT\_LINE*('Producatorul ' || nume\_producator || ' are ' || t.COUNT || ' produse in magazinul nostru.' );  
  
 FOR i IN t.FIRST..t.LAST LOOP  
 DBMS\_OUTPUT.*PUT\_LINE*( *INITCAP*(t(i).tip) || ': ' || t(i).nume);  
  
 END LOOP;  
  
end;  
BEGIN  
 *afisare\_produse*(2);  
END;

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

-- Afisam toti clientii dintr-un oras dat, care au facut comenzi de peste 300 de lei  
  
  
CREATE OR REPLACE PROCEDURE *peste\_300*(oras\_cautat adresa.oras%type)  
AS  
 CURSOR clienti (oras\_c adresa.oras%type) IS  
 SELECT (nume || ' ' || prenume) as nume , id\_client  
 FROM client  
 JOIN ADRESA A2 on client.id\_adresa = A2.ID\_ADRESA  
 WHERE *LOWER*(a2.ORAS) = *LOWER*(oras\_c);  
  
 CURSOR comenzi is  
 SELECT c.id\_comanda, *SUM*(p.pret) as pret, c.id\_client  
 FROM comanda c  
 JOIN produs\_comandat pc ON c.id\_comanda=pc.id\_comanda  
 JOIN produs p ON p.id\_produs = pc.id\_produs  
 group by c.id\_comanda, c.id\_client;  
  
 comanda comenzi%rowtype;  
 COUNTER INT;  
  
Begin  
  
 FOR client in clienti(oras\_cautat) LOOP  
 counter :=0;  
  
 DBMS\_OUTPUT.*PUT\_LINE*('Clientul ' || client.nume || ':');  
 OPEN comenzi;  
 FETCH comenzi INTO comanda;  
  
 IF comanda.id\_client = client.id\_client AND comanda.pret >= 300 THEN  
 DBMS\_OUTPUT.*PUT\_LINE*(' Comanda: ' || comanda.ID\_COMANDA || ', pretul: ' || comanda.pret);  
 counter := counter +1;  
  
 end if;  
  
 IF counter = 0 THEN  
 DBMS\_OUTPUT.*PUT\_LINE*(' Acest client nu are comenzi peste 300 de lei.');  
 end if;  
  
 CLOSE comenzi;  
  
 END LOOP;  
  
END;  
  
  
BEGIN  
 *peste\_300*('bucuresti');  
end;

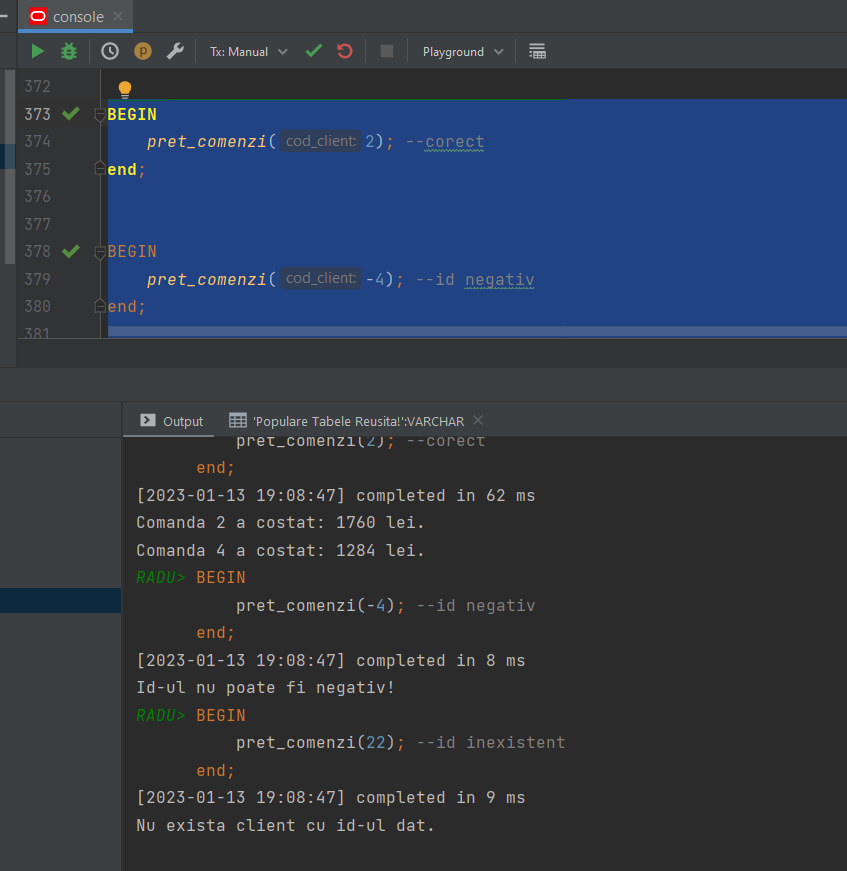


8. Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat independent de tip funcție care să utilizeze într-o singură comandă SQL 3 dintre tabelele definite.

Definiți minim 2 excepții. Apelați subprogramul astfel încât să evidențiați toate cazurile tratate.

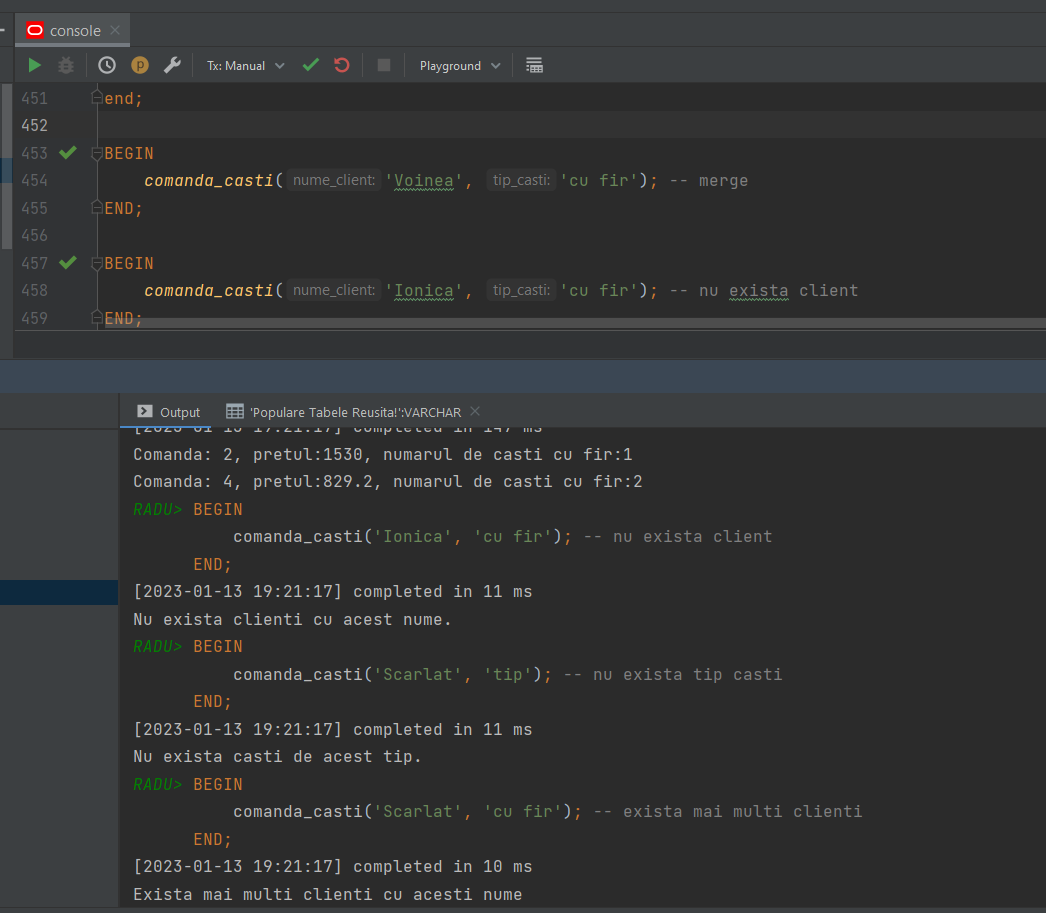
--pentru un client dat sa se afiseze valorile comenzilor, fara reducere  
-- functia returneaza numarul comenzilor

CREATE OR REPLACE FUNCTION *pret\_comenzi*(cod\_client client.id\_client%TYPE) return NUMBER  
AS  
 vr\_pret int;  
 vr\_id int;  
 clienti int;  
  
  
 CURSOR c IS  
 SELECT c.id\_comanda, *SUM*(p.pret)  
 FROM comanda c  
 JOIN produs\_comandat pc ON c.id\_comanda=pc.id\_comanda  
 JOIN produs p ON p.id\_produs = pc.id\_produs  
 WHERE c.id\_client = cod\_client  
 group by c.id\_comanda;  
 counter int;  
 NEGATIVE\_NUMBER EXCEPTION;  
 NO\_DATA\_FOUND1 EXCEPTION;  
  
BEGIN  
  
 counter := 0;  
  
 IF cod\_client <0 THEN  
 RAISE NEGATIVE\_NUMBER;  
 end if;  
  
 SELECT *count*(id\_client) INTO clienti  
 FROM client  
 WHERE id\_client = cod\_client;  
  
 IF clienti < 1 THEN  
 RAISE NO\_DATA\_FOUND1;  
 end if;  
  
 OPEN c;  
 LOOP  
 FETCH c INTO vr\_id,vr\_pret;  
 EXIT WHEN c%NOTFOUND;  
  
 DBMS\_OUTPUT.*PUT\_LINE*('Comanda ' || vr\_id || ' a costat: ' || vr\_pret || ' lei.');  
 counter := counter +1;  
  
 END LOOP;  
 CLOSE c;  
  
 return counter;  
  
EXCEPTION  
  
 WHEN INVALID\_NUMBER THEN  
 *RAISE\_APPLICATION\_ERROR*(-20000, 'Id-ul introdus nu este valid.');  
 return 0;  
  
 WHEN NEGATIVE\_NUMBER THEN  
 DBMS\_OUTPUT.*PUT\_LINE*('Id-ul nu poate fi negativ!');  
 return 0;  
 WHEN NO\_DATA\_FOUND1 THEN  
 DBMS\_OUTPUT.*PUT\_LINE*( 'Nu exista client cu id-ul dat.');  
 return 0;  
END;  
  
  
  
BEGIN  
 DBMS\_OUTPUT.*PUT\_LINE* (*pret\_comenzi*(2)); --corect  
end;  
  
  
BEGIN  
 *pret\_comenzi*(-4); --id negativ  
end;  
  
BEGIN  
 *pret\_comenzi*(22); --id inexistent  
end;



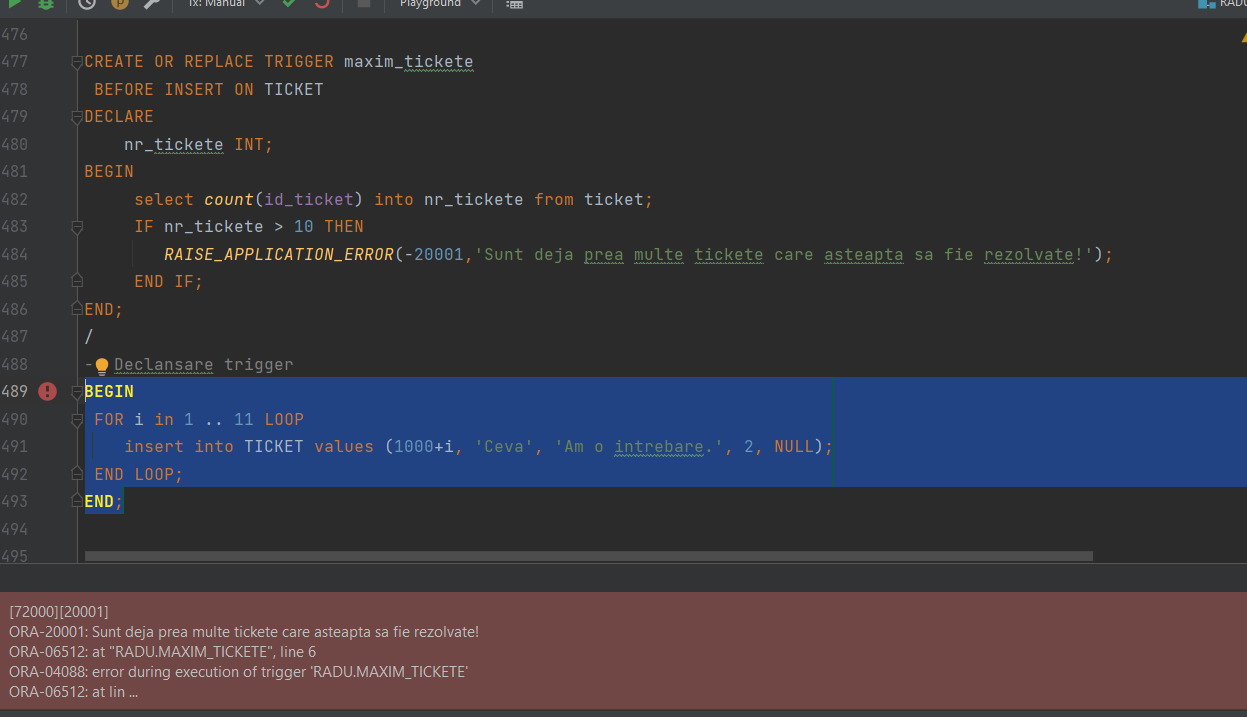
9. Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat independent de tip procedură care să utilizeze într-o singură comandă SQL 5 dintre tabelele definite.

Tratați toate excepțiile care pot apărea, incluzând excepțiile NO\_DATA\_FOUND și TOO\_MANY\_ROWS. Apelați subprogramul astfel încât să evidențiați toate cazurile tratate.

CREATE OR REPLACE PROCEDURE *comanda\_casti*(nume\_client client.nume%TYPE, tip\_casti casti.tip%TYPE)  
AS  
 cursor c IS  
 SELECT comanda.id\_comanda as id\_comanda, *sum*(p.pret) as pret, *COUNT*(p.id\_produs) as nr\_casti  
 FROM comanda  
 JOIN client c2 on comanda.id\_client = c2.id\_client  
 JOIN produs\_comandat pc on comanda.id\_comanda = pc.id\_comanda  
 JOIN produs p on pc.id\_produs = p.id\_produs  
 JOIN casti c3 on p.id\_produs = c3.id\_produs  
 WHERE *LOWER*(c3.CONECTIVITATE) = *LOWER*(tip\_casti) AND *LOWER*(c2.nume) = *LOWER*(nume\_client)  
 GROUP BY comanda.id\_comanda;  
  
 t c%rowtype;  
 k int;  
  
 NO\_DATA\_FOUND1 EXCEPTION;  
 NO\_DATA\_FOUND2 EXCEPTION;  
 TOO\_MANY\_ROWS1 EXCEPTION;  
begin  
 SELECT *count*(\*) into k  
 FROM CASTI  
 WHERE *LOWER*(casti.conectivitate) = *LOWER*(tip\_casti);  
  
 if k = 0 then  
 raise NO\_DATA\_FOUND1;  
 end if;  
  
  
 SELECT *count*(\*) into k  
 FROM client  
 WHERE *LOWER*(nume\_client) = *LOWER*(nume);  
  
 if k = 0 then  
 raise NO\_DATA\_FOUND2;  
 end if;  
  
 if k > 1 then  
 raise TOO\_MANY\_ROWS1;  
 end if;  
  
  
 FOR t in c LOOP  
 DBMS\_OUTPUT.*PUT\_LINE*('Comanda: ' || t.id\_comanda || ', pretul:' || t.pret || ', numarul de casti ' || tip\_casti|| ':' || t.nr\_casti);  
  
 END LOOP;  
  
EXCEPTION  
 WHEN NO\_DATA\_FOUND1 THEN  
 DBMS\_OUTPUT.*PUT\_LINE*('Nu exista casti de acest tip.');  
  
 WHEN NO\_DATA\_FOUND2 THEN  
 DBMS\_OUTPUT.*PUT\_LINE*('Nu exista clienti cu acest nume.');  
  
 WHEN TOO\_MANY\_ROWS1 THEN  
 DBMS\_OUTPUT.*PUT\_LINE*('Exista mai multi clienti cu acesti nume');  
  
end;  
  
BEGIN  
 *comanda\_casti*('Voinea', 'cu fir'); -- merge  
END;  
  
BEGIN  
 *comanda\_casti*('Ionica', 'cu fir'); -- nu exista client  
END;  
  
BEGIN  
 *comanda\_casti*('Scarlat', 'tip'); -- nu exista tip casti  
END;  
  
BEGIN  
 *comanda\_casti*('Scarlat', 'cu fir'); -- exista mai multi clienti  
END;

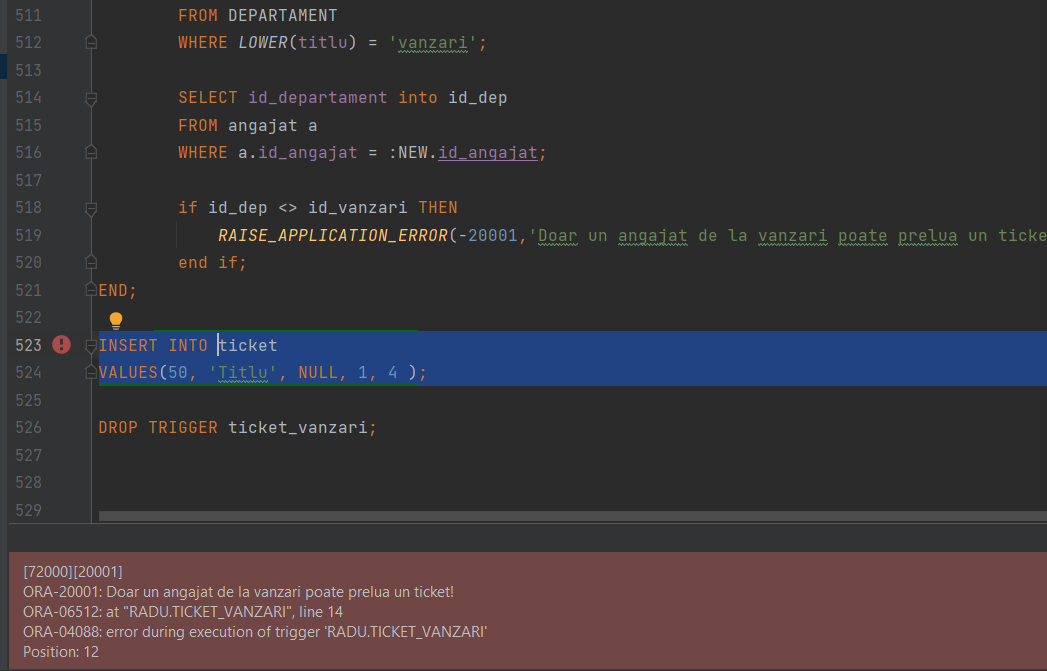
10. Definiți un trigger de tip LMD la nivel de comandă. Declanșați trigger-ul.

--Definim un trigger care sa nu permita adaugarea a mai mult de 10 tickete de support.  
  
CREATE OR REPLACE TRIGGER maxim\_tickete  
 BEFORE INSERT ON TICKET  
DECLARE  
 nr\_tickete INT;  
BEGIN  
 select *count*(id\_ticket) into nr\_tickete from ticket;  
 IF nr\_tickete > 10 THEN  
 *RAISE\_APPLICATION\_ERROR*(-20001,'Sunt deja prea multe tickete care asteapta sa fie rezolvate!');  
 END IF;  
END;  
/  
-- Declansare trigger  
BEGIN  
 FOR i in 1 .. 11 LOOP  
 insert into TICKET values (1000+i, 'Ceva', 'Am o intrebare.', 2, NULL);  
 END LOOP;  
END;  
  
  
-- Sterge trigger  
DROP TRIGGER maxim\_tickete;

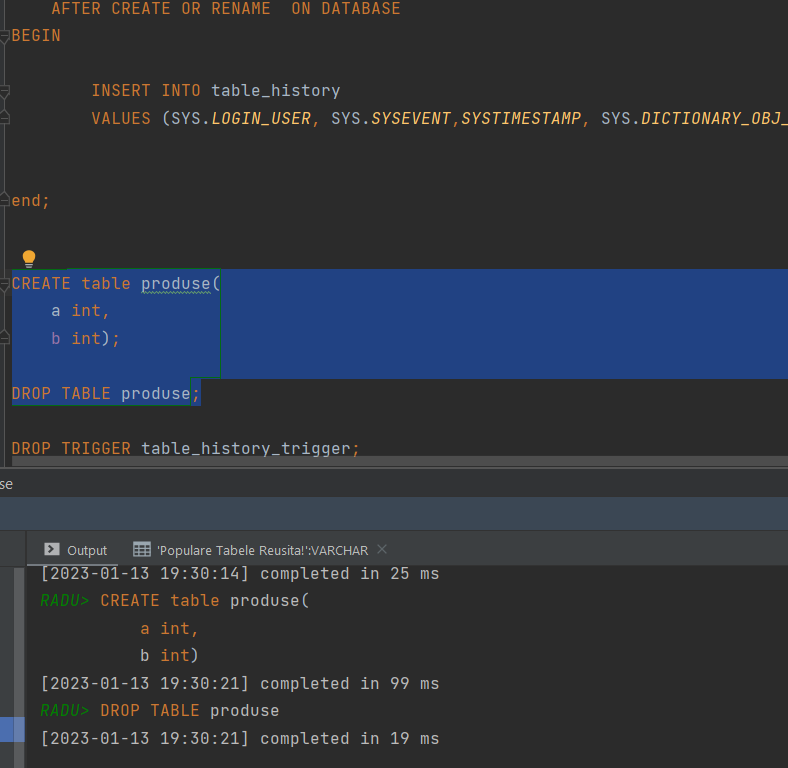


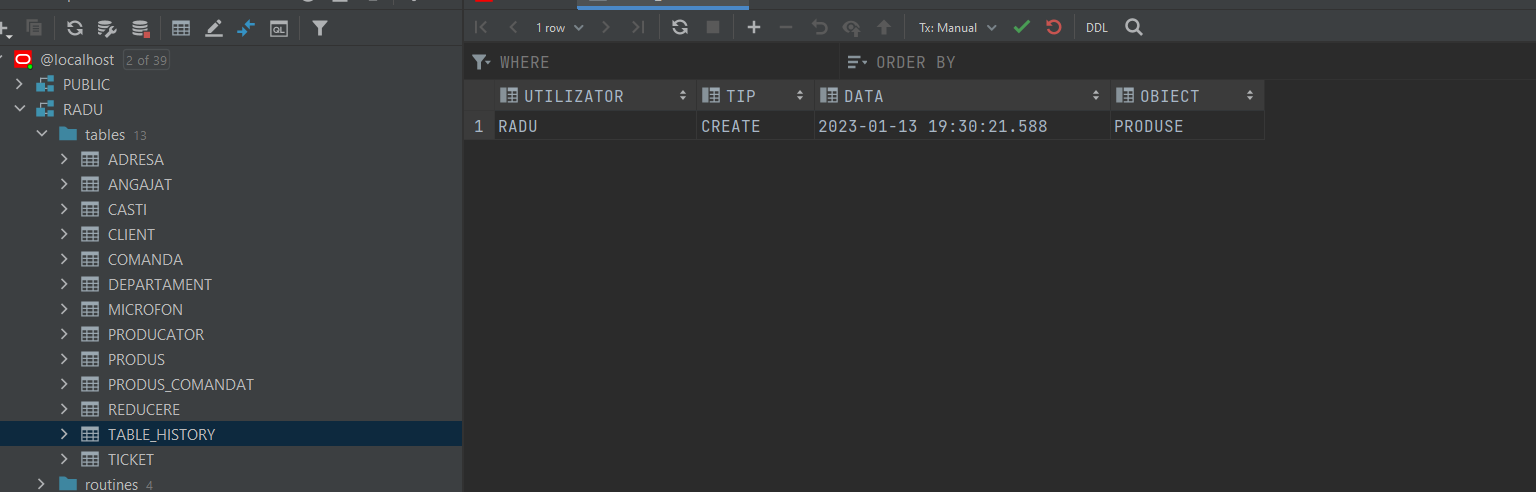
11. Definiți un trigger de tip LMD la nivel de linie. Declanșați trigger-ul.

-- Definim un trigger care nu ne lasa sa adaugam un ticket decat pentru un angajat din departamentul vanzari  
  
CREATE OR REPLACE TRIGGER ticket\_vanzari  
 BEFORE INSERT ON ticket  
 FOR EACH ROW  
 DECLARE  
 id\_vanzari INT;  
 id\_dep INT;  
BEGIN  
 SELECT id\_departament into id\_vanzari  
 FROM DEPARTAMENT  
 WHERE *LOWER*(titlu) = 'vanzari';  
  
 SELECT id\_departament into id\_dep  
 FROM angajat a  
 WHERE a.id\_angajat = :NEW.id\_angajat;  
  
 if id\_dep <> id\_vanzari THEN  
 *RAISE\_APPLICATION\_ERROR*(-20001,'Doar un angajat de la vanzari poate prelua un ticket!');  
 end if;  
END;  
  
INSERT INTO ticket  
VALUES(50, 'Titlu', NULL, 1, 4 );



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

-- Definim un trigger care memoreazea in tabela insert\_history toate operatiile de tip insert sau rename  
  
CREATE TABLE table\_history  
(  
 utilizator VARCHAR(30),  
 tip VARCHAR(10),  
 data TIMESTAMP(3),  
 obiect VARCHAR(30)  
);  
  
  
CREATE OR REPLACE TRIGGER table\_history\_trigger  
 AFTER CREATE OR RENAME ON DATABASE  
BEGIN  
  
 INSERT INTO table\_history  
 VALUES (SYS.*LOGIN\_USER*, SYS.*SYSEVENT*,*SYSTIMESTAMP*, SYS.*DICTIONARY\_OBJ\_NAME* );  
  
  
end;  
  
  
CREATE table produse(  
 a int,  
 b int);



Cerințe opționale pentru nota finală N >= 6:

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

CREATE OR REPLACE PACKAGE pachet\_magazin AS  
 PROCEDURE *afisare\_produse* (nume\_producator producator.NUME%TYPE); --6  
 PROCEDURE *peste\_300*(oras\_cautat adresa.oras%type); --7  
 PROCEDURE *pret\_comenzi*(cod\_client client.id\_client%TYPE); --8  
 PROCEDURE *comanda\_casti*(nume\_client client.nume%TYPE, tip\_casti casti.tip%TYPE); --9  
END pachet\_magazin;  
  
  
CREATE OR REPLACE PACKAGE BODY pachet\_magazin AS  
  
  
 PROCEDURE *afisare\_produse* (nume\_producator producator.NUME%TYPE)  
 AS  
 TYPE tabel\_produse IS TABLE OF produs%ROWTYPE INDEX BY PLS\_INTEGER;  
 t tabel\_produse;  
  
 TYPE tablou\_imbricat IS TABLE OF VARCHAR2(50);  
 ti tablou\_imbricat := tablou\_imbricat();  
  
 id\_prod INT;  
  
 BEGIN  
  
 SELECT id\_producator INTO id\_prod  
 FROM producator  
 WHERE *LOWER*(nume) = *LOWER*(nume\_producator);  
  
 SELECT \* BULK COLLECT INTO t  
 FROM PRODUS  
 WHERE id\_producator = id\_prod;  
  
 DBMS\_OUTPUT.*PUT\_LINE*('Producatorul ' || nume\_producator || ' are ' || t.COUNT || ' produse in magazinul nostru.' );  
  
 FOR i IN t.FIRST..t.LAST LOOP  
 ti.*extend*();  
 ti(i) := ( *INITCAP*( t(i).tip) || ': ' || t(i).nume);  
  
 END LOOP;  
  
 FOR i IN ti.FIRST .. ti.LAST LOOP  
 DBMS\_OUTPUT.*PUT\_LINE*(ti(i));  
 END LOOP;  
  
  
 end *afisare\_produse*;  
  
  
  
  
-- 7. Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat independent care să utilizeze  
-- 2 tipuri de cursoare studiate, unul dintre acestea fiind cursor parametrizat.  
-- Apelați subprogramul.  
  
-- Afisam toti clientii dintr-un oras dat, care au facut comenzi de peste 300 de lei  
  
  
 PROCEDURE *peste\_300*(oras\_cautat adresa.oras%type)  
AS  
 CURSOR clienti (oras\_c adresa.oras%type) IS  
 SELECT (nume || ' ' || prenume) as nume , id\_client  
 FROM client  
 JOIN ADRESA A2 on client.id\_adresa = A2.ID\_ADRESA  
 WHERE *LOWER*(a2.ORAS) = *LOWER*(oras\_c);  
  
 CURSOR comenzi is  
 SELECT c.id\_comanda, *SUM*(p.pret) as pret, c.id\_client  
 FROM comanda c  
 JOIN produs\_comandat pc ON c.id\_comanda=pc.id\_comanda  
 JOIN produs p ON p.id\_produs = pc.id\_produs  
 group by c.id\_comanda, c.id\_client;  
  
 comanda comenzi%rowtype;  
 COUNTER INT;  
  
Begin  
  
 FOR client in clienti(oras\_cautat) LOOP  
 counter :=0;  
  
 DBMS\_OUTPUT.*PUT\_LINE*('Clientul ' || client.nume || ':');  
 OPEN comenzi;  
 FETCH comenzi INTO comanda;  
  
 IF comanda.id\_client = client.id\_client AND comanda.pret >= 300 THEN  
 DBMS\_OUTPUT.*PUT\_LINE*(' Comanda: ' || comanda.ID\_COMANDA || ', pretul: ' || comanda.pret);  
 counter := counter +1;  
  
 end if;  
  
 IF counter = 0 THEN  
 DBMS\_OUTPUT.*PUT\_LINE*(' Acest client nu are comenzi peste 300 de lei.');  
 end if;  
  
 CLOSE comenzi;  
  
 END LOOP;  
  
END *peste\_300*;  
  
  
  
  
  
  
-- 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.  
  
  
  
--pentru un client dat sa se afiseze valorile comenzilor, fara reducere  
  
PROCEDURE *pret\_comenzi*(cod\_client client.id\_client%TYPE)  
AS  
 vr\_pret int;  
 vr\_id int;  
 clienti int;  
  
  
 CURSOR c IS  
 SELECT c.id\_comanda, *SUM*(p.pret)  
 FROM comanda c  
 JOIN produs\_comandat pc ON c.id\_comanda=pc.id\_comanda  
 JOIN produs p ON p.id\_produs = pc.id\_produs  
 WHERE c.id\_client = cod\_client  
 group by c.id\_comanda;  
  
 NEGATIVE\_NUMBER EXCEPTION;  
 NO\_DATA\_FOUND1 EXCEPTION;  
BEGIN  
  
 IF cod\_client <0 THEN  
 RAISE NEGATIVE\_NUMBER;  
 end if;  
  
 SELECT *count*(id\_client) INTO clienti  
 FROM client  
 WHERE id\_client = cod\_client;  
  
 IF clienti < 1 THEN  
 RAISE NO\_DATA\_FOUND1;  
 end if;  
  
 OPEN c;  
 LOOP  
 FETCH c INTO vr\_id,vr\_pret;  
 EXIT WHEN c%NOTFOUND;  
  
 DBMS\_OUTPUT.*PUT\_LINE*('Comanda ' || vr\_id || ' a costat: ' || vr\_pret || ' lei.');  
  
 END LOOP;  
 CLOSE c;  
  
EXCEPTION  
  
 WHEN INVALID\_NUMBER THEN  
 *RAISE\_APPLICATION\_ERROR*(-20000, 'Id-ul introdus nu este valid.');  
  
 WHEN NEGATIVE\_NUMBER THEN  
 DBMS\_OUTPUT.*PUT\_LINE*('Id-ul nu poate fi negativ!');  
 WHEN NO\_DATA\_FOUND1 THEN  
 DBMS\_OUTPUT.*PUT\_LINE*( 'Nu exista client cu id-ul dat.');  
END *pret\_comenzi*;  
  
  
  
  
  
  
  
  
--9  
  
-- Pentru un nume de client dat afisam toate comenzile ce contin casti de un anumit tip  
  
 PROCEDURE *comanda\_casti*(nume\_client client.nume%TYPE, tip\_casti casti.tip%TYPE)  
AS  
 cursor c IS  
 SELECT comanda.id\_comanda as id\_comanda, *sum*(p.pret) as pret, *COUNT*(p.id\_produs) as nr\_casti  
 FROM comanda  
 JOIN client c2 on comanda.id\_client = c2.id\_client  
 JOIN produs\_comandat pc on comanda.id\_comanda = pc.id\_comanda  
 JOIN produs p on pc.id\_produs = p.id\_produs  
 JOIN casti c3 on p.id\_produs = c3.id\_produs  
 WHERE *LOWER*(c3.CONECTIVITATE) = *LOWER*(tip\_casti) AND *LOWER*(c2.nume) = *LOWER*(nume\_client)  
 GROUP BY comanda.id\_comanda;  
  
 t c%rowtype;  
 k int;  
  
 NO\_DATA\_FOUND1 EXCEPTION;  
 NO\_DATA\_FOUND2 EXCEPTION;  
 TOO\_MANY\_ROWS1 EXCEPTION;  
begin  
 SELECT *count*(\*) into k  
 FROM CASTI  
 WHERE *LOWER*(casti.conectivitate) = *LOWER*(tip\_casti);  
  
 if k = 0 then  
 raise NO\_DATA\_FOUND1;  
 end if;  
  
  
 SELECT *count*(\*) into k  
 FROM client  
 WHERE *LOWER*(nume\_client) = *LOWER*(nume);  
  
 if k = 0 then  
 raise NO\_DATA\_FOUND2;  
 end if;  
  
 if k > 1 then  
 raise TOO\_MANY\_ROWS1;  
 end if;  
  
  
 FOR t in c LOOP  
 DBMS\_OUTPUT.*PUT\_LINE*('Comanda: ' || t.id\_comanda || ', pretul:' || t.pret || ', numarul de casti ' || tip\_casti|| ':' || t.nr\_casti);  
  
 END LOOP;  
  
EXCEPTION  
 WHEN NO\_DATA\_FOUND1 THEN  
 DBMS\_OUTPUT.*PUT\_LINE*('Nu exista casti de acest tip.');  
  
 WHEN NO\_DATA\_FOUND2 THEN  
 DBMS\_OUTPUT.*PUT\_LINE*('Nu exista clienti cu acest nume.');  
  
 WHEN TOO\_MANY\_ROWS1 THEN  
 DBMS\_OUTPUT.*PUT\_LINE*('Exista mai multi clienti cu acesti nume');  
  
end *comanda\_casti*;  
  
END pachet\_magazin;  
  
  
-- testare pachet  
BEGIN  
  
 PACHET\_MAGAZIN.*afisare\_produse*('Rode');  
 PACHET\_MAGAZIN.*peste\_300*('bucuresti');  
 PACHET\_MAGAZIN.*pret\_comenzi*('2');  
 PACHET\_MAGAZIN.*comanda\_casti*('voinea', 'cu fir');  
end;