

Sisteme de Gestiune a Bazelor de Date

Proiect Final

1. Prezențați 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 și căști.

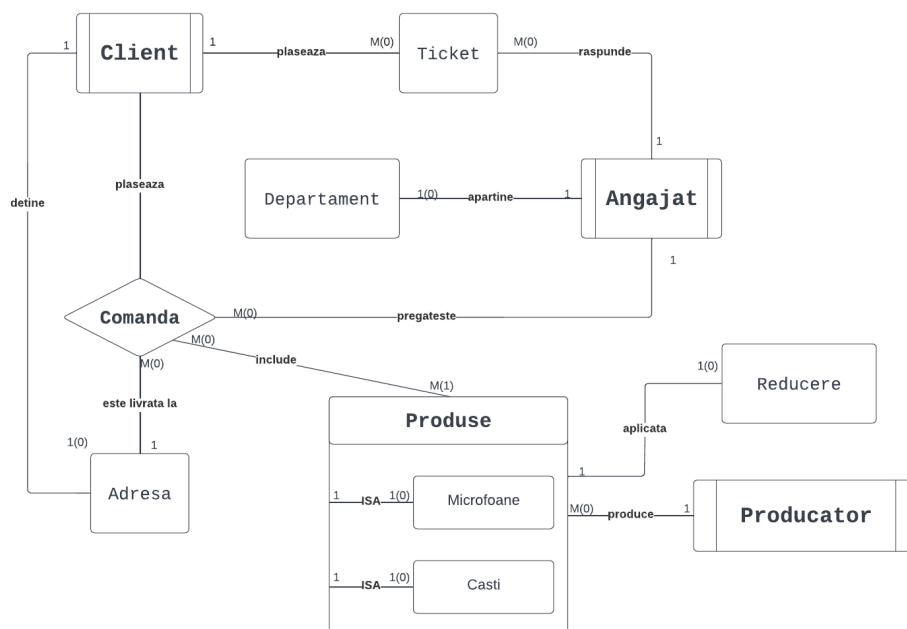
Pe pagina site-ului, clienții vor trebui să își creeze un cont pentru a putea plasa o comandă. Contul de client va reține numele și prenumele, data creării contului și adresa principală de livrare.

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

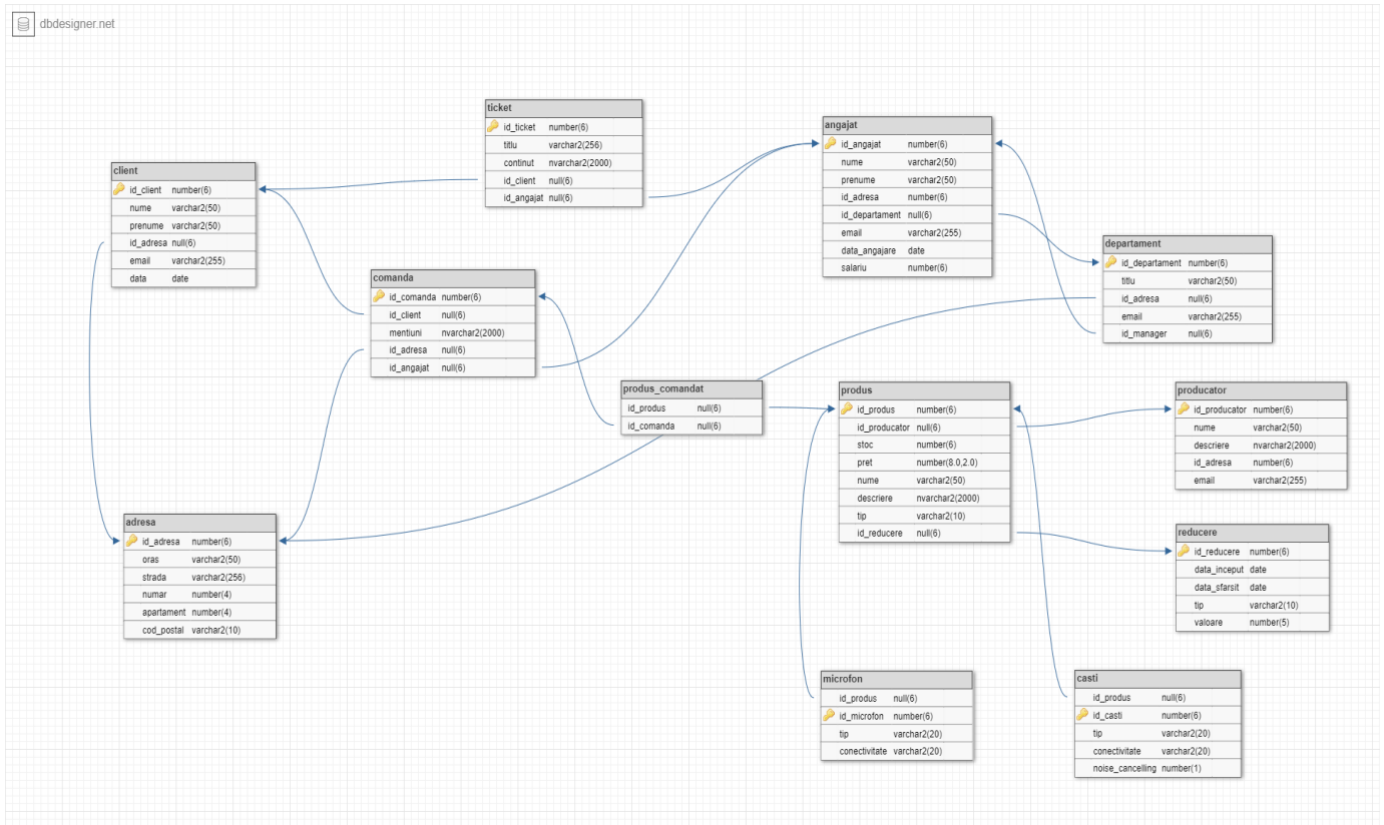
Totodată, o comandă plasată va fi atribuită unui angajat din departamentul „Comenzi” care o va împacheta și trimite.

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

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 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
);

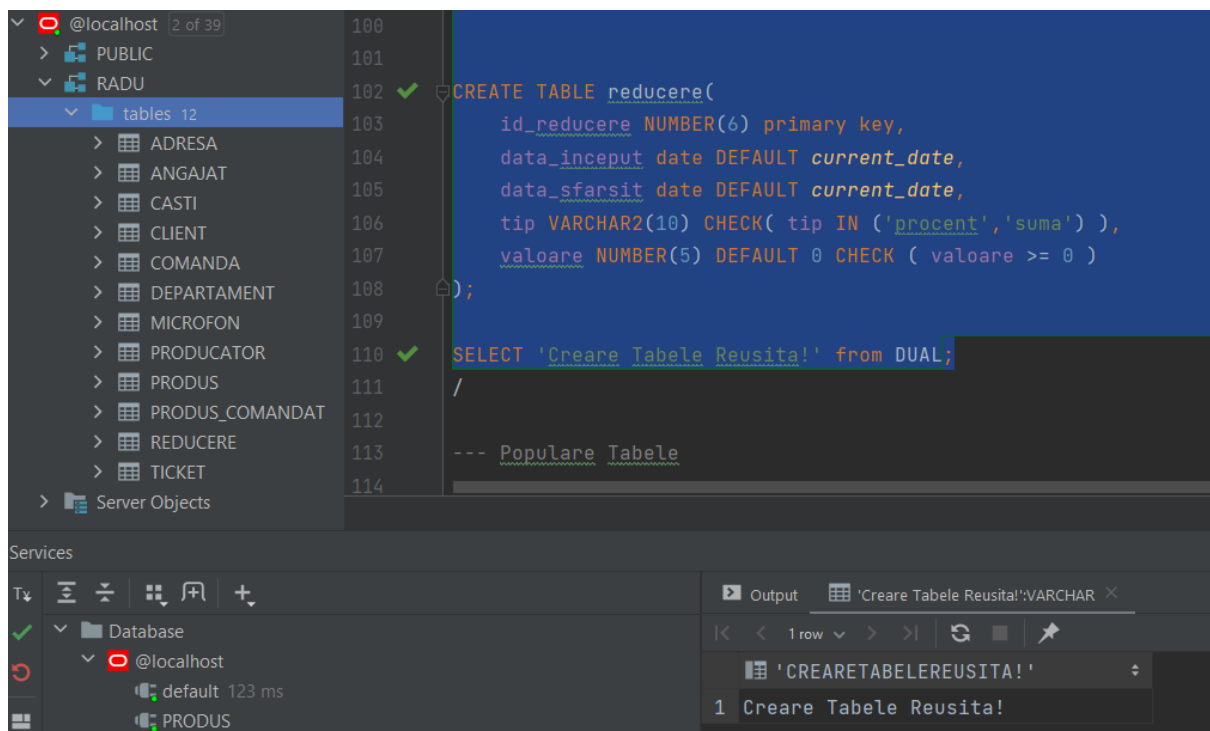
CREATE TABLE departament(
    id_departament NUMBER(6) primary key,
    titlu VARCHAR2(50) NOT NULL,
    id_adresa NUMBER(6),
    email VARCHAR2(255) NOT NULL,
    id_manager NUMBER(6) DEFAULT 1
);

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 'Create 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@yahoo.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)
```

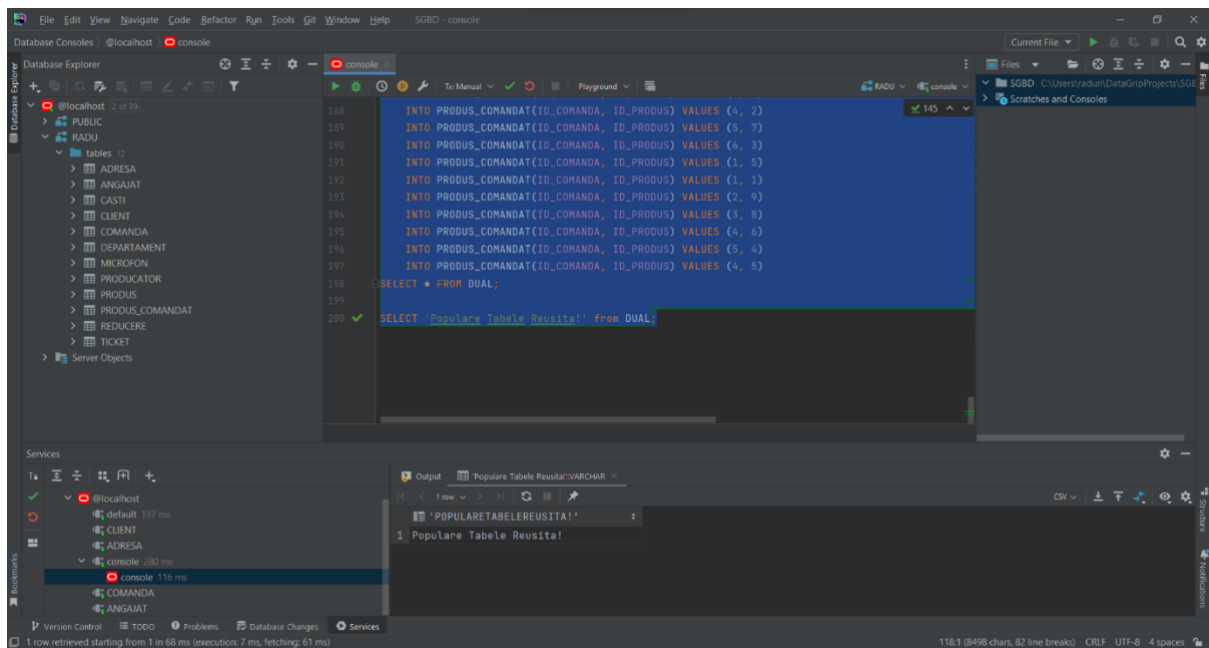
```
INTO ticket (id_ticket, id_client, titlu, continut, ID_ANGAJAT ) VALUES
(9, 4, 'Plata', 'Ce modalitati de plata exista??', 2)
INTO reducere(id_reducere, DATA_INCEPUT, DATA_SFARSIT, tip, VALOARE)
VALUES (4, '15-JAN-2004', '15-JAN-2006', 'procent', 15)
INTO reducere(id_reducere, DATA_INCEPUT, DATA_SFARSIT, 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_REDCERE,
stoc ) VALUES (1, 4, 'Podcaster', 840.50, 'microfon', NULL, 5)
INTO produs (id_produs, ID_PRODUCATOR, NUME, pret, TIP, ID_REDCERE,
stoc ) VALUES (2, 4, 'PodMic', 455, 'microfon', NULL, 10)
INTO produs (id_produs, ID_PRODUCATOR, NUME, pret, TIP, ID_REDCERE,
stoc ) VALUES (3, 3, 'SM58', 400, 'microfon', NULL, 15)
INTO produs (id_produs, ID_PRODUCATOR, NUME, pret, TIP, ID_REDCERE,
stoc ) VALUES (4, 2, 'HD 600', 1530, 'casti', NULL, 14)
INTO produs (id_produs, ID_PRODUCATOR, NUME, pret, TIP, ID_REDCERE,
stoc ) VALUES (5, 1, 'ATH-M50x', 579.40, 'casti', NULL, 100)
INTO produs (id_produs, ID_PRODUCATOR, NUME, pret, TIP, ID_REDCERE,
stoc ) VALUES (6, 1, 'ATH-M20x', 249.80, 'casti', 5, 11)
INTO produs (id_produs, ID_PRODUCATOR, NUME, pret, TIP, ID_REDCERE,
stoc ) VALUES (7, 6, 'ZR10', 335, 'casti', NULL, 29)
INTO produs (id_produs, ID_PRODUCATOR, NUME, pret, TIP, ID_REDCERE,
stoc ) VALUES (8, 1, 'ATH-M30x', 239.67, 'casti', NULL, 42)
INTO produs (id_produs, ID_PRODUCATOR, NUME, pret, TIP, ID_REDCERE,
stoc ) VALUES (9, 5, 'C-1U', 229.99, 'microfon', NULL, 50)
INTO produs (id_produs, ID_PRODUCATOR, NUME, pret, TIP, ID_REDCERE,
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 salvate intr-o colectie de tip tabel indexat.
CREATE OR REPLACE PROCEDURE afisare_produce (nume_producator
producator.id_producator%TYPE)
AS
    TYPE tabel_produce IS TABLE OF produs%ROWTYPE INDEX BY PLS_INTEGER;
    t tabel_produce;

BEGIN
    SELECT * BULK COLLECT INTO t
    FROM PRODUS
    WHERE id_producator = nume_producator;
```

```
237 FOR i IN t.FIRST .. t.LAST LOOP
238     ti.extend();
239     ti(i) := ( INITCAP( t(i).tip) || ': ' || t(i).nume);
240
241 END LOOP;
242
243 FOR i IN ti.FIRST .. ti.LAST LOOP
244     DBMS_OUTPUT.PUT_LINE( A: ti(i));
245 END LOOP;
246
247
248 end;
249
250 BEGIN
251     afisare_produce( nume_producator: 'Rode');
252 END;
```

Output

'Populare Tabele Reusita':VARCHAR X

```
RADU> BEGIN
        afisare_produce('Rode');
    END;

[2023-01-13 18:44:10] completed in 10 ms
Producatorul Rode are 2 produse in magazinul nostru.
Microfon: Podcaster
Microfon: PodMic
```

```
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_produce(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;  
    END LOOP;
```

```
        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;
```

```
    CLOSE comenzi;  
  
END LOOP;  
  
END;  
  
BEGIN  
    peste_300( oras_cautat: 'bucuresti');  
end;
```

Output 'Populare Tabele Reusita':VARCHAR X

```
[2023-01-13 18:46:54] completed in 38 ms  
RADU> BEGIN  
        peste_300('bucuresti');  
    end;  
[2023-01-13 18:46:59] completed in 59 ms  
Clientul Al Glanetasului Ion:  
    Comanda: 1, pretul: 1819.9  
Clientul Voinea Marcel:  
    Acest client nu are comenzi peste 300 de lei.  
Clientul Scarlat Horia:  
    Acest client nu are comenzi peste 300 de lei.  
Clientul Scarlat Gabriel:  
    Acest client nu are comenzi peste 300 de lei.
```

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
```

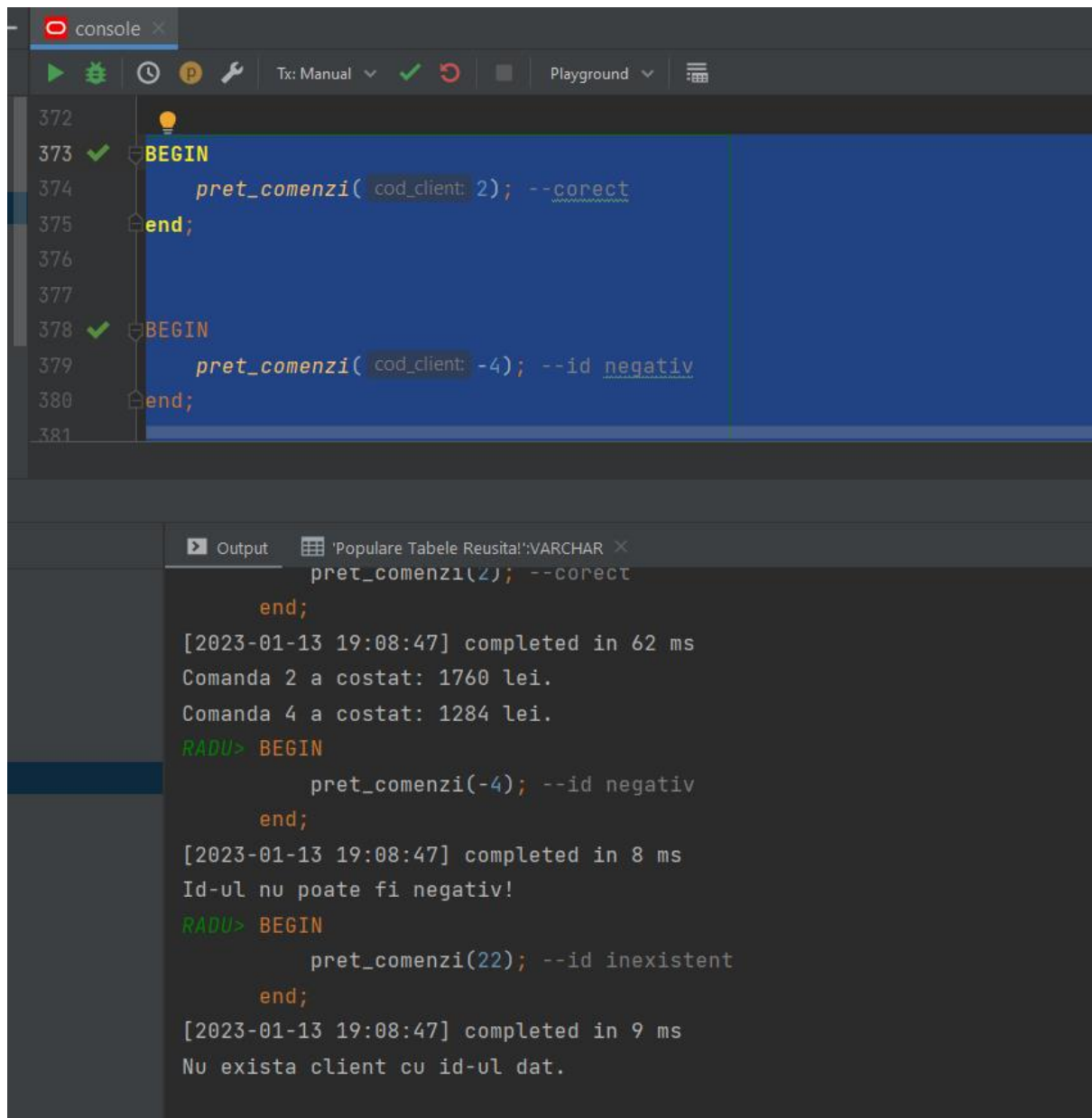
```
end;
```



```
BEGIN
```

```
    pret_comenzi(22); --id inexistent
```

```
end;
```



The screenshot shows a database playground interface with a code editor and an output console.

Code Editor:

```
372  
373 ✓ BEGIN  
374     pret_comenzi( cod_client: 2); --corect  
375 end;  
376  
377  
378 ✓ BEGIN  
379     pret_comenzi( cod_client: -4); --id negativ  
380 end;  
381
```

Output Console:

```
Output 'Populare Tabele Reusita':VARCHAR X  
pret_comenzi(2); --corect  
  
end;  
[2023-01-13 19:08:47] completed in 62 ms  
Comanda 2 a costat: 1760 lei.  
Comanda 4 a costat: 1284 lei.  
RADU> BEGIN  
      pret_comenzi(-4); --id negativ  
      end;  
[2023-01-13 19:08:47] completed in 8 ms  
Id-ul nu poate fi negativ!  
RADU> BEGIN  
      pret_comenzi(22); --id inexistent  
      end;  
[2023-01-13 19:08:47] completed in 9 ms  
Nu exista client cu id-ul dat.
```


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_produc) 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_produc = p.id_produc
    JOIN casti c3 on p.id_produc = c3.id_produc
    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;
end;
```

```
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

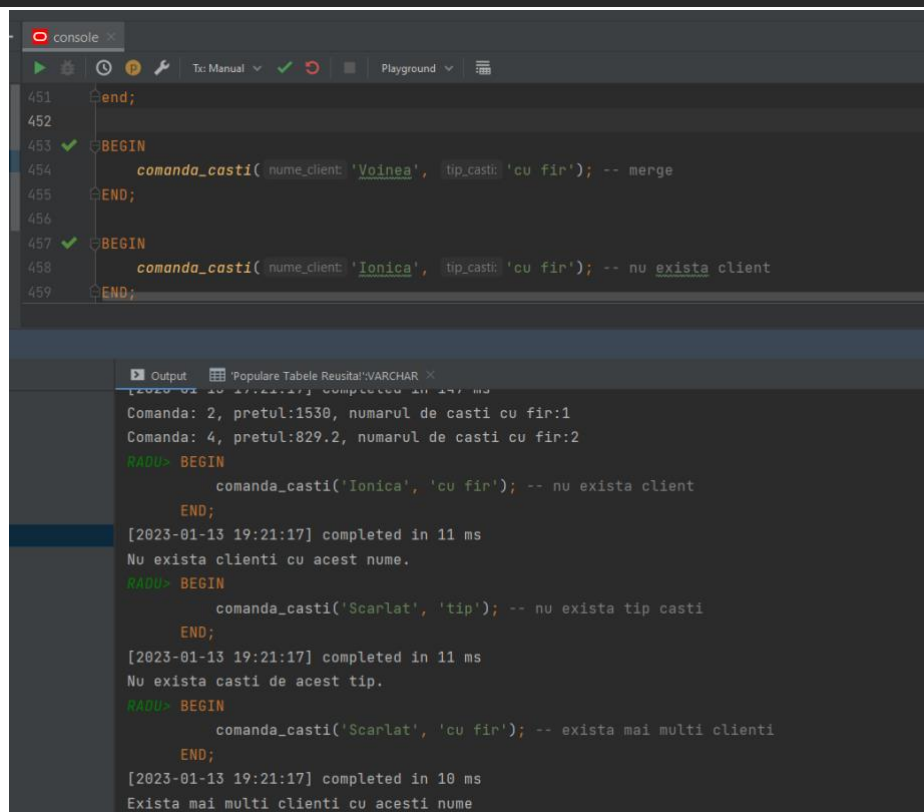
```
  comanda_casti('Scarlat', 'tip'); -- nu exista tip casti
```

END;

BEGIN

```
  comanda_casti('Scarlat', 'cu fir'); -- exista mai multi clienti
```

END;



The screenshot shows a SQL IDE interface. The top pane displays SQL code with line numbers 451 to 459. The code includes several calls to a procedure `comanda_casti` with comments indicating expected outcomes: 'merge', 'nu exista client', 'nu exista tip casti', and 'exista mai multi clienti'. The bottom pane shows the 'Output' window with the following text:

```
[2023-01-13 19:21:17] completed in 11 ms
Comanda: 2, pretul:1530, numarul de casti cu fir:1
Comanda: 4, pretul:829.2, numarul de casti cu fir:2
RADU> BEGIN
      comanda_casti('Ionica', 'cu fir'); -- nu exista client
      END;
[2023-01-13 19:21:17] completed in 11 ms
Nu exista clienti cu acest nume.
RADU> BEGIN
      comanda_casti('Scarlat', 'tip'); -- nu exista tip casti
      END;
[2023-01-13 19:21:17] completed in 11 ms
Nu exista casti de acest tip.
RADU> BEGIN
      comanda_casti('Scarlat', 'cu fir'); -- exista mai multi clienti
      END;
[2023-01-13 19:21:17] completed in 10 ms
Exista mai multi clienti cu acesti nume
```

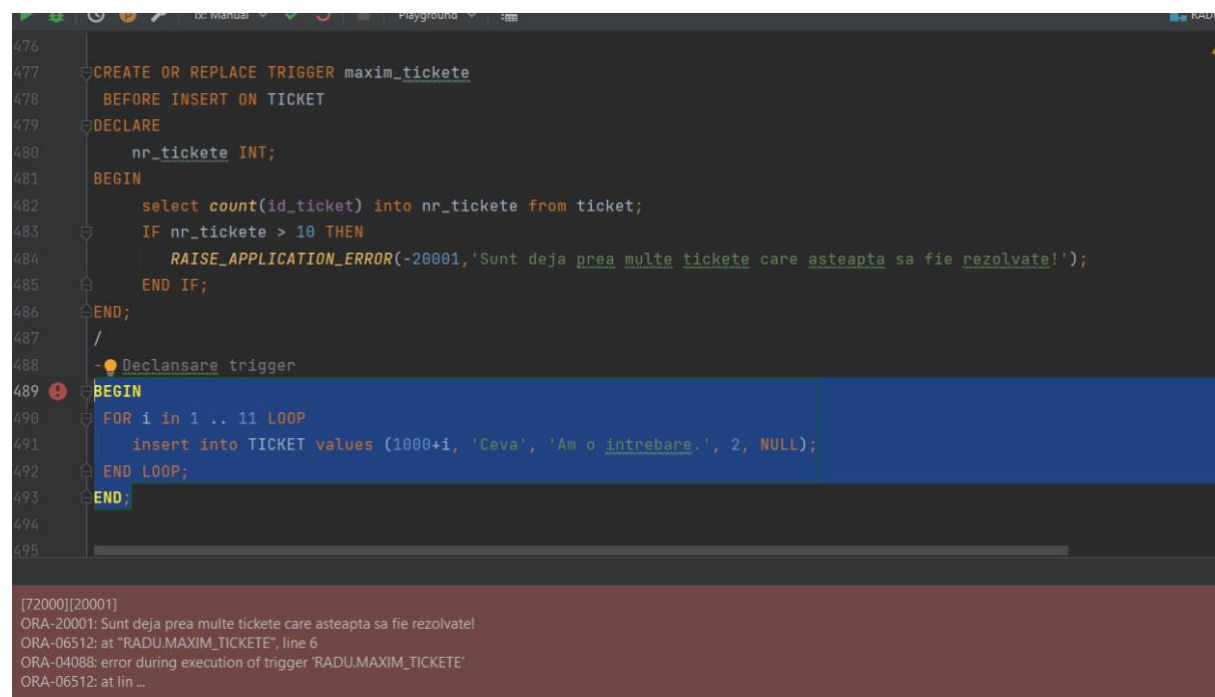
10. Definiți un trigger de tip LMD la nivel de comandă. Declanșați trigger-ul.

```
--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;
```



```
476
477 CREATE OR REPLACE TRIGGER maxim_tickete
478   BEFORE INSERT ON TICKET
479 DECLARE
480   nr_tickete INT;
481 BEGIN
482   select count(id_ticket) into nr_tickete from ticket;
483   IF nr_tickete > 10 THEN
484     RAISE_APPLICATION_ERROR(-20001,'Sunt deja prea multe tickete care asteapta sa fie rezolvate!');
485   END IF;
486 END;
487 /
488 -- Declansare trigger
489 BEGIN
490   FOR i in 1 .. 11 LOOP
491     insert into TICKET values (1000+i, 'Ceva', 'Am o intrebare.', 2, NULL);
492   END LOOP;
493 END;
494
495
```

[72000][20001]
ORA-20001: Sunt deja prea multe tickete care asteapta sa fie rezolvate!
ORA-06512: at "RADU.MAXIM_TICKETE", line 6
ORA-04088: error during execution of trigger 'RADU.MAXIM_TICKETE'
ORA-06512: at lin ...

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 );
```

```
511      FROM DEPARTAMENT
512      WHERE LOWER(titlu) = 'vanzari';
513
514      SELECT id_departament into id_dep
515      FROM angajat a
516      WHERE a.id_angajat = :NEW.id_angajat;
517
518      if id_dep <> id_vanzari THEN
519          RAISE_APPLICATION_ERROR(-20001,'Doar un angajat de la vanzari poate prelua un ticke
520      end if;
521  END;
522
523  INSERT INTO ticket
524  VALUES(50, 'Titlu', NULL, 1, 4 );
525
526  DROP TRIGGER ticket_vanzari;
527
528
529
```

```
[72000][20001]
ORA-20001: Doar un angajat de la vanzari poate prelua un ticket!
ORA-06512: at "RADU.TICKET_VANZARI", line 14
ORA-04088: error during execution of trigger 'RADU.TICKET_VANZARI'
Position: 12
```

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

```
-- Definim un trigger care memoreaza 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);
```

```
    AFTER CREATE OR RENAME ON DATABASE
BEGIN

    INSERT INTO table_history
    VALUES (SYS.LOGIN_USER, SYS.SYSEVENT, SYSTIMESTAMP, SYS.DICTIONARY_OBJ_

end;

CREATE table produse(
    a int,
    b int);

DROP TABLE produse;

DROP TRIGGER table_history_trigger;
```

Output Populara Tabele Reusital: VARCHAR X

[2023-01-13 19:30:14] completed in 25 ms

RADU> CREATE table produse(

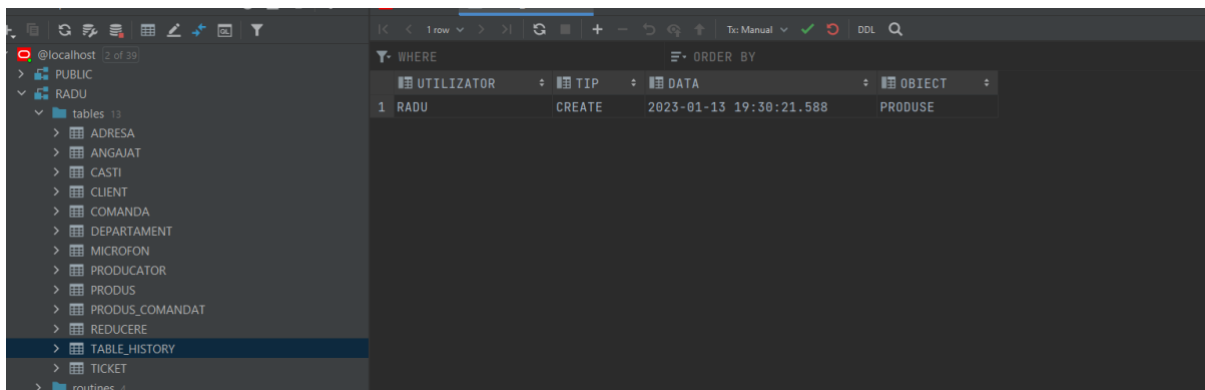
 a int,

 b int)

[2023-01-13 19:30:21] completed in 99 ms

RADU> DROP TABLE produse

[2023-01-13 19:30:21] completed in 19 ms



Cerințe opționale pentru nota finală $N \geq 6$:

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

```
CREATE OR REPLACE PACKAGE pachet_magazin AS
    PROCEDURE afisare_produce (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_produce (nume_producator producator.NUME%TYPE)
    AS
        TYPE tabel_produce IS TABLE OF produs%ROWTYPE INDEX BY PLS_INTEGER;
        t tabel_produce;

        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_produce;

-- 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;
    END LOOP;
END;
```

```
        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_produș) 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_produș = p.id_produș
    JOIN casti c3 on p.id_produș = c3.id_produș
    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(num_client) = LOWER(num);

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_produce('Rode');
    PACHET_MAGAZIN.pestes_300('bucuresti');
    PACHET_MAGAZIN.pret_comenzi('2');
    PACHET_MAGAZIN.comanda_casti('voinea', 'cu fir');
end;
```