

# **Proiect Sisteme de Gestiune a Bazelor de Date**

**Glazov Maria - Liana**

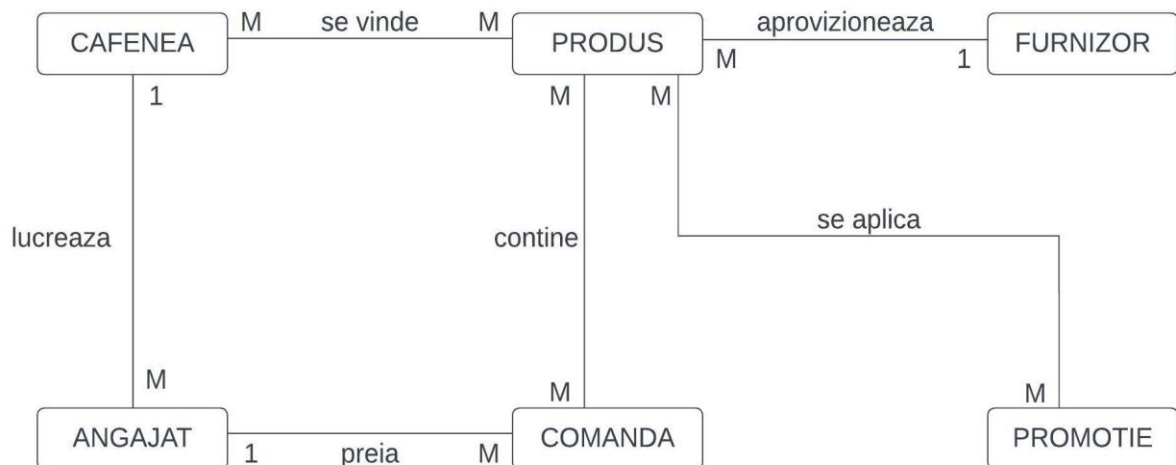
**Grupa 244**

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

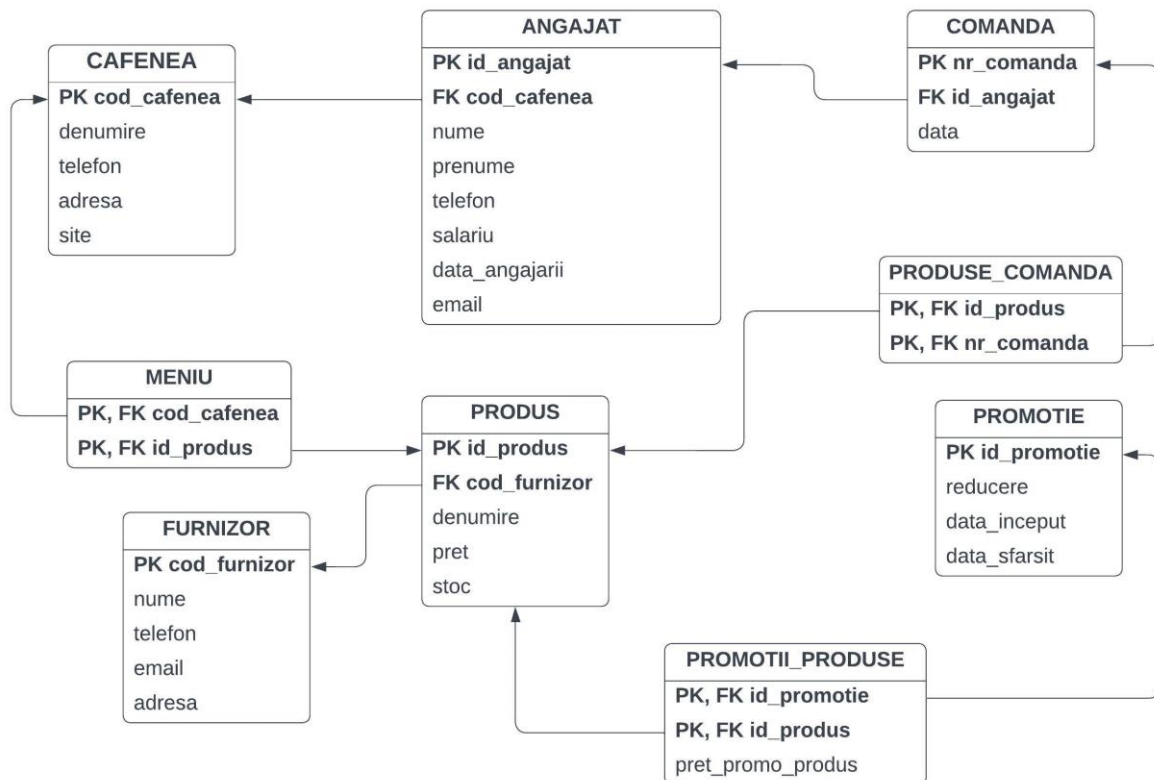
În această bază de date sunt stocate informații relevante pentru administrarea unui lanț de cafenele. Baza de date conține tabelele CAFENEA, ANGAJAT, PRODUS, COMANDA, FURNIZOR, PROMOTIE, precum și tabelele asociative MENIU, care face legătura între cafenea și produs, PRODUSE\_COMANDA și PROMOTII\_PRODUSE.

La o cafenea lucrează cel puțin un angajat, care poate prelua comenzi conținând unul sau mai multe produse din meniu. Fiecare produs este cumpărat de la un furnizor, iar uneori se aplică promoții asupra produselor disponibile, care le reduc prețul.

2) Realizați diagrama entitate-relație (ERD): entitățile, relațiile și atributele trebuie definite în limba română.



3) Pornind de la diagrama entitate-relație realizați diagrama conceptuală a modelului propus, integrând toate atributele necesare: entitățile, relațiile și atributele trebuie definite în limba română.



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

CREATE TABLE CAFENEA (

cod\_cafenea NUMBER(5) PRIMARY KEY,

telefon VARCHAR2(10) NOT NULL,

adresa VARCHAR2(100) NOT NULL,

site VARCHAR2(100)

);

CREATE TABLE ANGAJAT (

id\_angajat NUMBER(5) PRIMARY KEY,

nume VARCHAR2(50) NOT NULL,

prenume VARCHAR2(50) NOT NULL,

telefon VARCHAR2(10) NOT NULL,

cod\_cafenea NUMBER(5) NOT NULL REFERENCES CAFENEA (cod\_cafenea),

salariu NUMBER(4) NOT NULL,

data\_angajarii DATE NOT NULL,

email VARCHAR2(40)

```
);  
  
CREATE TABLE COMANDA (  
    nr_comanda NUMBER(4) PRIMARY KEY,  
    data DATE NOT NULL,  
    id_angajat NUMBER(5) NOT NULL REFERENCES ANGAJAT (id_angajat)  
);  
  
CREATE TABLE FURNIZOR(  
    cod_furnizor NUMBER(4) PRIMARY KEY,  
    nume VARCHAR2(100) NOT NULL,  
    telefon VARCHAR2(10) NOT NULL,  
    adresa VARCHAR2(100) NOT NULL,  
    email VARCHAR2(30)  
);  
  
CREATE TABLE PRODUS(  
    id_produs NUMBER(4) PRIMARY KEY,  
    denumire VARCHAR2(30) NOT NULL,  
    pret NUMBER(3) NOT NULL,  
    stoc NUMBER(4),  
    cod_furnizor NUMBER(4) NOT NULL REFERENCES FURNIZOR (cod_furnizor)  
);  
  
CREATE TABLE PROMOTIE(  
    id_promotie NUMBER(4) PRIMARY KEY,  
    reducere NUMBER(3) NOT NULL,  
    data_start DATE NOT NULL,  
    data_fin DATE NOT NULL  
);  
  
CREATE TABLE PRODUSE_COMANDA(  
    id_produs NUMBER(4) NOT NULL REFERENCES PRODUS(id_produs),  
    nr_comanda NUMBER(4) NOT NULL REFERENCES COMANDA(nr_comanda)  
);  
  
CREATE TABLE MENIU (
```

SGBD  
Seria 24

```
cod_cafenea NUMBER(5) NOT NULL REFERENCES CAFENEA(cod_cafenea),  
id_produș NUMBER(4) NOT NULL REFERENCES PRODUS(id_produș)  
);
```

```
CREATE TABLE PROMOTII_PRODUSE(  
    id_promotie NUMBER(4) NOT NULL REFERENCES PROMOTIE(id_promotie),  
    id_produș NUMBER(4) NOT NULL REFERENCES PRODUS(id_produș),  
    pret_promo_produș NUMBER(3)  
);
```

```
CREATE SEQUENCE promotie_id_seq  
    START WITH 1  
    INCREMENT BY 1  
    NOCACHE  
    NOCYCLE;
```

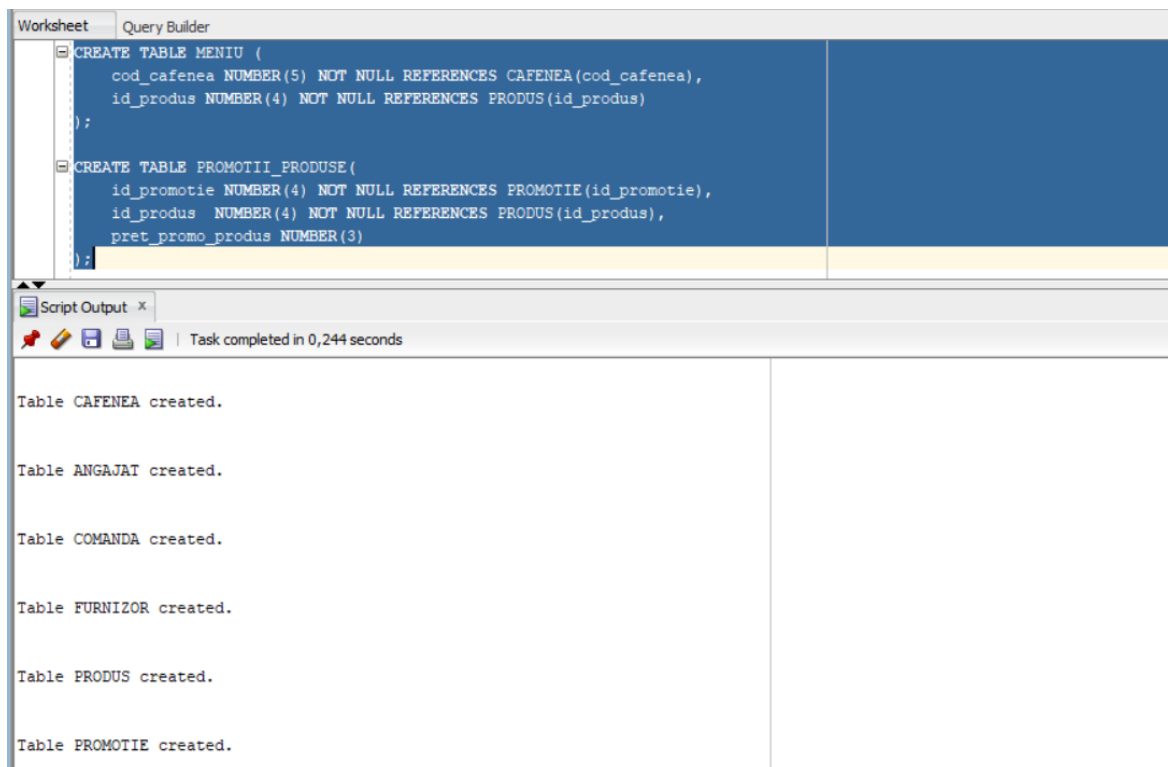
```
CREATE SEQUENCE cafenea_cod_seq  
    START WITH 1  
    INCREMENT BY 1  
    NOCACHE  
    NOCYCLE;
```

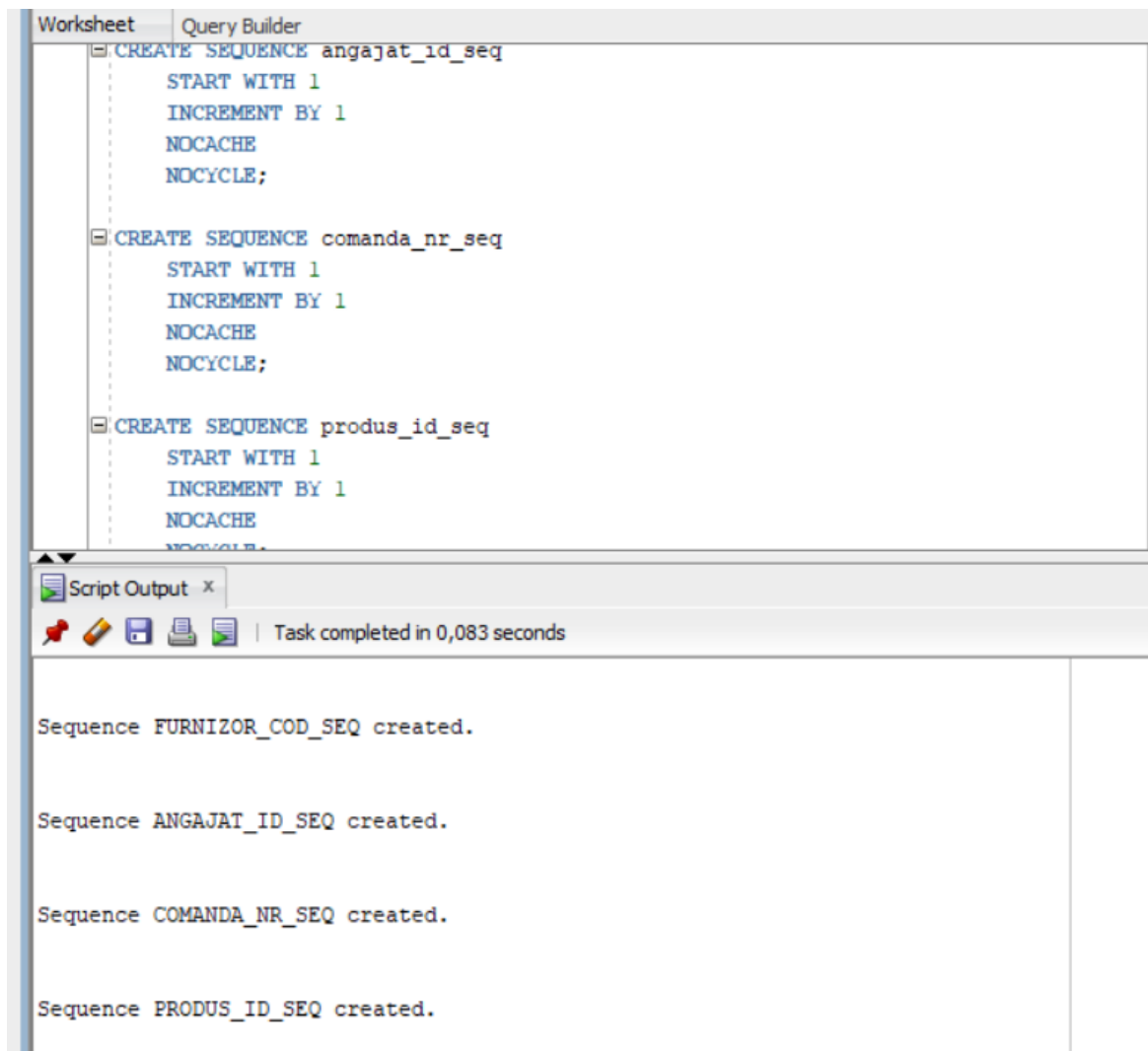
```
CREATE SEQUENCE furnizor_cod_seq  
    START WITH 1  
    INCREMENT BY 1  
    NOCACHE  
    NOCYCLE;
```

```
CREATE SEQUENCE angajat_id_seq  
    START WITH 1  
    INCREMENT BY 1  
    NOCACHE  
    NOCYCLE;
```

```
CREATE SEQUENCE comanda_nr_seq  
  
START WITH 1  
  
INCREMENT BY 1  
  
NOCACHE  
  
NOCYCLE;
```

```
CREATE SEQUENCE produs_id_seq  
  
START WITH 1  
  
INCREMENT BY 1  
  
NOCACHE  
  
NOCYCLE;
```





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

INSERT INTO CAFENEA (cod\_cafenea, telefon, adresa, site)

VALUES (cafenea\_cod\_seq.NEXTVAL, '0745638273', 'STR. MIHAI VITEAZU bl. 6, RADAUTI', NULL);

INSERT INTO CAFENEA (cod\_cafenea, telefon, adresa, site)

VALUES (cafenea\_cod\_seq.NEXTVAL, '0764729387', 'STR. VLADIMIRESCU TUDOR nr. 28, VALCEA', 'cafeVAL.ce');

INSERT INTO CAFENEA (cod\_cafenea, telefon, adresa, site)

VALUES (cafenea\_cod\_seq.NEXTVAL, '0764836274', 'BD. MOROIANU GEORGE nr. 27, SACELE', NULL);

INSERT INTO CAFENEA (cod\_cafenea, telefon, adresa, site)

SGBD  
Seria 24

```
VALUES (cafenea_cod_seq.NEXTVAL, '0737847583', 'STR. 22 DECEMBRIE 1989 nr. 38, MURES', NULL);
```

```
INSERT INTO CAFENEA (cod_cafenea, telefon, adresa, site)
```

```
VALUES (cafenea_cod_seq.NEXTVAL, '0758493849', 'STR. CUZA I. AL. nr. 42, DOLJ', NULL);
```

```
INSERT INTO ANGAJAT (id_angajat, nume, prenume, telefon, cod_cafenea, salariu, data_angajarii, email)
```

```
VALUES (angajat_id_seq.NEXTVAL, 'Mironescu', 'Sorina', '0764839478', '1', '2780', DATE '2021-08-23', NULL);
```

```
INSERT INTO ANGAJAT (id_angajat, nume, prenume, telefon, cod_cafenea, salariu, data_angajarii, email)
```

```
VALUES (angajat_id_seq.NEXTVAL, 'Silivasi', 'Claudia', '0763787489', '3', '4500', DATE '2017-03-02', NULL);
```

```
INSERT INTO ANGAJAT (id_angajat, nume, prenume, telefon, cod_cafenea, salariu, data_angajarii, email)
```

```
VALUES (angajat_id_seq.NEXTVAL, 'Petran', 'Gheorghe', '0765748930', '4', '2900', DATE '2021-07-09', 'ghegheor_ptrn@gmail.com');
```

```
INSERT INTO ANGAJAT (id_angajat, nume, prenume, telefon, cod_cafenea, salariu, data_angajarii, email)
```

```
VALUES (angajat_id_seq.NEXTVAL, 'Blaga', 'Constanta', '0754637893', '5', '3480', DATE '2020-01-26', NULL);
```

```
INSERT INTO ANGAJAT (id_angajat, nume, prenume, telefon, cod_cafenea, salariu, data_angajarii, email)
```

```
VALUES (angajat_id_seq.NEXTVAL, 'Popa', 'Raul', '0765789487', '4', '3120', DATE '2019-12-03', NULL);
```

```
INSERT INTO COMANDA (nr_comanda,data,id_angajat)
```

```
VALUES (comanda_nr_seq.NEXTVAL, DATE '2023-11-10', '2');
```

```
INSERT INTO COMANDA (nr_comanda, data, id_angajat)
```

```
VALUES (comanda_nr_seq.NEXTVAL, DATE '2023-11-29','2');
```

```
INSERT INTO COMANDA (nr_comanda, data,id_angajat)
```

```
VALUES (comanda_nr_seq.NEXTVAL, DATE '2023-11-21','4');
```

```
INSERT INTO COMANDA (nr_comanda, data,id_angajat)
```



```
VALUES (comanda_nr_seq.NEXTVAL, DATE '2023-11-19', '3');
```

```
INSERT INTO COMANDA (nr_comanda,data, id_angajat)
```

```
VALUES (comanda_nr_seq.NEXTVAL, DATE '2023-11-13', '3');
```

```
INSERT INTO FURNIZOR (cod_furnizor, nume, telefon, adresa, email)
```

```
VALUES (furnizor_cod_seq.NEXTVAL, 'Magazinul cu Produse', '0748374657', 'STR.  
PATULEA, ING. nr. 4B, BUCURESTI SECTOR 1', NULL);
```

```
INSERT INTO FURNIZOR (cod_furnizor, nume, telefon, adresa, email)
```

```
VALUES (furnizor_cod_seq.NEXTVAL, 'La Cafele', '0748392749', 'STR.  
VLADIMIRESCU T. nr. 30A, CONSTANTA', 'lacafe@gmail.com');
```

```
INSERT INTO FURNIZOR (cod_furnizor, nume, telefon, adresa, email)
```

```
VALUES (furnizor_cod_seq.NEXTVAL, 'De Toate', '0728398476', 'Strada 22 Decembrie 31,  
Zalau', NULL);
```

```
INSERT INTO FURNIZOR (cod_furnizor, nume, telefon, adresa, email)
```

```
VALUES (furnizor_cod_seq.NEXTVAL, 'ADCF', '0798476387', 'STR. CASTELULUI nr.  
110 ap. 2, BRASOV', NULL);
```

```
INSERT INTO FURNIZOR (cod_furnizor, nume, telefon, adresa, email)
```

```
VALUES (furnizor_cod_seq.NEXTVAL, 'MAJI PRAJI', '0764738476', 'STR. OVAZULUI  
nr. 10, SIBIU', 'maji_praji@yahoo.com');
```

```
INSERT INTO PRODUS (id_produs,denumire, pret, stoc, cod_furnizor)
```

```
VALUES (produs_id_seq.NEXTVAL, 'cafe latte', '15', '100', '1');
```

```
INSERT INTO PRODUS (id_produs,denumire, pret, stoc, cod_furnizor)
```

```
VALUES (produs_id_seq.NEXTVAL, 'fursec cu ciocolata', '5', '29', '1');
```

```
INSERT INTO PRODUS (id_produs,denumire, pret, stoc, cod_furnizor)
```

```
VALUES (produs_id_seq.NEXTVAL, 'tiramisu', '20', '45', '2');
```

```
INSERT INTO PRODUS (id_produs,denumire, pret, stoc, cod_furnizor)
```

```
VALUES (produs_id_seq.NEXTVAL, 'cappuccino', '10', '140', '3');
```

```
INSERT INTO PRODUS (id_produs,denumire, pret, stoc, cod_furnizor)
```

```
VALUES (produs_id_seq.NEXTVAL, 'milkshake', '19', '15', '5');
```

```
INSERT INTO PROMOTIE(id_promotie, reducere, data_start,data_fin)
```

```
VALUES(promotie_id_seq.NEXTVAL,'20',DATE '2023-11-24',DATE '2023-11-29');
INSERT INTO PROMOTIE(id_promotie, reducere, data_start,data_fin)
VALUES(promotie_id_seq.NEXTVAL,'15',DATE '2023-11-12',DATE '2023-11-23');
INSERT INTO PROMOTIE(id_promotie, reducere, data_start,data_fin)
VALUES(promotie_id_seq.NEXTVAL,'10',DATE '2023-11-30',DATE '2023-12-02');
INSERT INTO PROMOTIE(id_promotie, reducere, data_start,data_fin)
VALUES(promotie_id_seq.NEXTVAL,'25',DATE '2023-12-06',DATE '2023-12-24');
INSERT INTO PROMOTIE(id_promotie, reducere, data_start,data_fin)
VALUES(promotie_id_seq.NEXTVAL,'5',DATE '2023-11-07',DATE '2023-11-10');
```

```
INSERT INTO PRODUSE_COMANDA(id_produs, nr_comanda)
VALUES ('2','3');
INSERT INTO PRODUSE_COMANDA(id_produs, nr_comanda)
VALUES ('2','5');
INSERT INTO PRODUSE_COMANDA(id_produs, nr_comanda)
VALUES ('2','4');
INSERT INTO PRODUSE_COMANDA(id_produs, nr_comanda)
VALUES ('3','1');
INSERT INTO PRODUSE_COMANDA(id_produs, nr_comanda)
VALUES ('4','2');
INSERT INTO PRODUSE_COMANDA(id_produs, nr_comanda)
VALUES ('4','5');
INSERT INTO PRODUSE_COMANDA(id_produs, nr_comanda)
VALUES ('1','5');
INSERT INTO PRODUSE_COMANDA(id_produs, nr_comanda)
VALUES ('1','4');
```

```
INSERT INTO PRODUSE_COMANDA(id_produs, nr_comanda)
VALUES ('1','2');
INSERT INTO PRODUSE_COMANDA(id_produs, nr_comanda)
VALUES ('2','1');
```

SGBD  
Seria 24

```
SELECT * FROM produse_comanda;

INSERT INTO MENIU(cod_cafenea, id_produs)
VALUES ('1','2');

INSERT INTO MENIU(cod_cafenea, id_produs)
VALUES ('1','5');

INSERT INTO MENIU(cod_cafenea, id_produs)
VALUES ('1','3');

INSERT INTO MENIU(cod_cafenea, id_produs)
VALUES ('4','3');

INSERT INTO MENIU(cod_cafenea, id_produs)
VALUES ('4','4');

INSERT INTO MENIU(cod_cafenea, id_produs)
VALUES ('5','2');

INSERT INTO MENIU(cod_cafenea, id_produs)
VALUES ('5','5');

INSERT INTO MENIU(cod_cafenea, id_produs)
VALUES ('2','4');

INSERT INTO MENIU(cod_cafenea, id_produs)
VALUES ('2','3');

INSERT INTO MENIU(cod_cafenea, id_produs)
VALUES ('2','2');


INSERT INTO PROMOTII_PRODUSE(id_produs,id_promotie,pret_promo_produs)
VALUES('1','2','17');


INSERT INTO PROMOTII_PRODUSE(id_produs,id_promotie,pret_promo_produs)
VALUES('1','1','16');

INSERT INTO PROMOTII_PRODUSE(id_produs,id_promotie,pret_promo_produs)
VALUES('1','4','15');

INSERT INTO PROMOTII_PRODUSE(id_produs,id_promotie,pret_promo_produs)
VALUES('2','2','7');
```

```
INSERT INTO PROMOTII_PRODUSE(id_produs,id_promotie,pret_promo_produs)
VALUES('2','5','9');
```

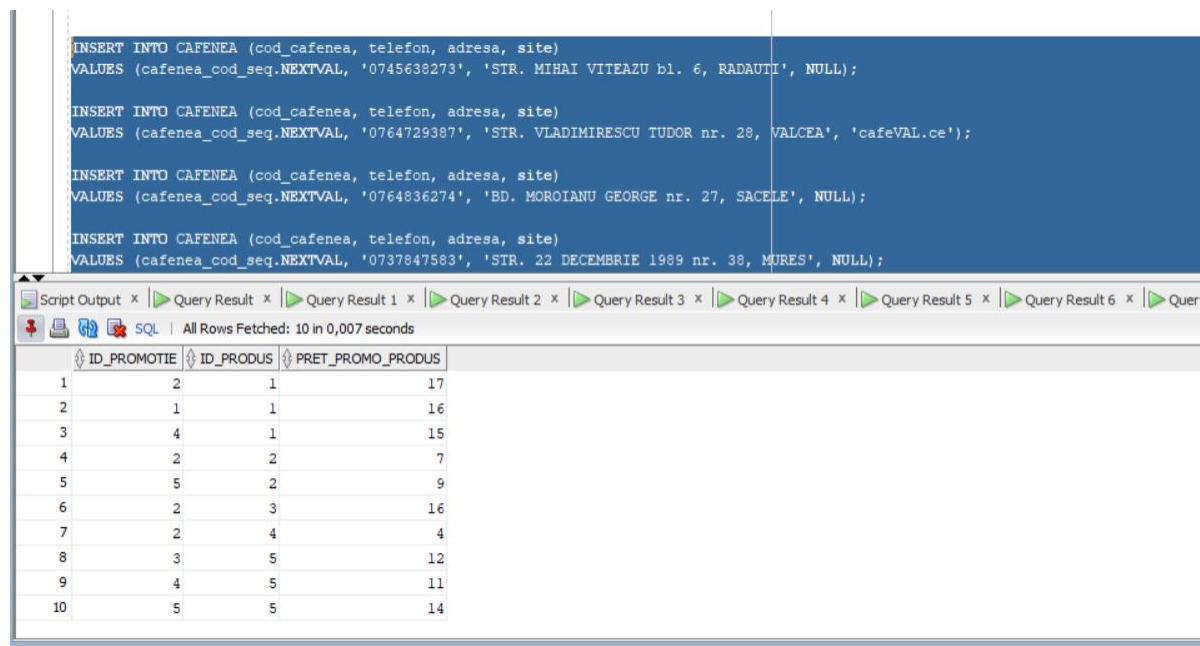
```
INSERT INTO PROMOTII_PRODUSE(id_produs,id_promotie,pret_promo_produs)
VALUES('3','2','16');
```

```
INSERT INTO PROMOTII_PRODUSE(id_produs,id_promotie,pret_promo_produs)
VALUES('4','2','4');
```

```
INSERT INTO PROMOTII_PRODUSE(id_produs,id_promotie,pret_promo_produs)
VALUES('5','3','12');
```

```
INSERT INTO PROMOTII_PRODUSE(id_produs,id_promotie,pret_promo_produs)
VALUES('5','4','11');
```

```
INSERT INTO PROMOTII_PRODUSE(id_produs,id_promotie,pret_promo_produs)
VALUES('5','5','14');
```



```
INSERT INTO CAFENEA (cod_cafenea, telefon, adresa, site)
VALUES (cafenea_cod_seq.NEXTVAL, '0745638273', 'STR. MIHAI VITEAZU bl. 6, RADAUTII', NULL);

INSERT INTO CAFENEA (cod_cafenea, telefon, adresa, site)
VALUES (cafenea_cod_seq.NEXTVAL, '0764729387', 'STR. VLADIMIRESCU TUDOR nr. 28, VALCEA', 'cafeVAL.ce');

INSERT INTO CAFENEA (cod_cafenea, telefon, adresa, site)
VALUES (cafenea_cod_seq.NEXTVAL, '0764836274', 'BD. MOROIANU GEORGE nr. 27, SACELE', NULL);

INSERT INTO CAFENEA (cod_cafenea, telefon, adresa, site)
VALUES (cafenea_cod_seq.NEXTVAL, '0737847583', 'STR. 22 DECEMBRIE 1989 nr. 38, MURES', NULL);
```

ID_PROMOTIE	ID_PRODUS	PRET_PROMO_PRODUS
1	2	17
2	1	16
3	4	15
4	2	7
5	5	9
6	2	16
7	2	4
8	3	12
9	4	11
10	5	14

6) Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat independent care să utilizeze toate cele 3 tipuri de colecții studiate. Apelați subprogramul.

Să se definească o procedură care primește ca parametru mai multe comenzi, iar pentru fiecare dintre comenzile respective calculează prețul acestora. Prețul unei comenzi este suma tuturor produselor din comandă. Unele produse se pot afla la reducere la data la care a fost data comanda, dacă este cazul reducerea se scade din prețul produsului.

```
CREATE OR REPLACE TYPE t_nr_comenzi AS TABLE OF NUMBER(4); --tablou imbricat
```

/

```
CREATE OR REPLACE PROCEDURE p_comenzi (nr_comenzi t_nr_comenzi)
IS
TYPE t_id_prod IS VARRAY(10) OF produs.id_produs%TYPE; --vector
produse_in_comanda t_id_prod;
detalii_comanda comanda%ROWTYPE;
pret_total NUMBER;
pret_curent produs.pret%TYPE;
TYPE t_promo IS TABLE OF promotie%ROWTYPE INDEX BY PLS_INTEGER; --tabel
indexat
detalii_promotie t_promo;
BEGIN
FOR i in nr_comenzi.FIRST..nr_comenzi.LAST LOOP --parcurgerea comenzilor
    --se preiau detaliile comenzilor
    SELECT *
    INTO detalii_comanda
    FROM comanda WHERE nr_comanda = nr_comenzi(i);
    --determinarea produselor dintr-o comanda
    SELECT pc.id_produs
    BULK COLLECT INTO produse_in_comanda
    FROM produs p, produse_comanda pc
    WHERE p.id_produs = pc.id_produs AND pc.nr_comanda = nr_comenzi(i);
    --se verifica pt fiecare produs daca este aplicata vreo promotie la data comenzii
    pret_total := 0;
    FOR j IN produse_in_comanda.FIRST..produse_in_comanda.LAST LOOP
        -- se selecteaza toate promotiile care au fost aplicate unui produs
        SELECT p.*
        BULK COLLECT INTO detalii_promotie
        FROM promotie p, promotii_produse pr
        WHERE pr.id_produs = produse_in_comanda(j) AND pr.id_promotie = p.id_promotie;
```

```
SELECT pret
INTO pret_curent
FROM produs
WHERE id_produs = produse_in_comanda(j);

-- se parcurg promotiile si se verifica daca este una din ele activa la data comenzii
curente

-- si se adauga reducerea in acest caz

FOR k IN detalii_promotie.FIRST..detalii_promotie.LAST LOOP

    IF detalii_comanda.data BETWEEN detalii_promotie(k).data_start AND
detalii_promotie(k).data_fin THEN

        pret_curent := pret_curent - ROUND(detalii_promotie(k).reducere/100 *
pret_curent);

        END IF;

    END LOOP;

    pret_total := pret_total + pret_curent;

END LOOP;

DBMS_OUTPUT.PUT_LINE('Comanda cu numarul ' || nr_comenzi(i) || ' costa ' ||
pret_total);

END LOOP;

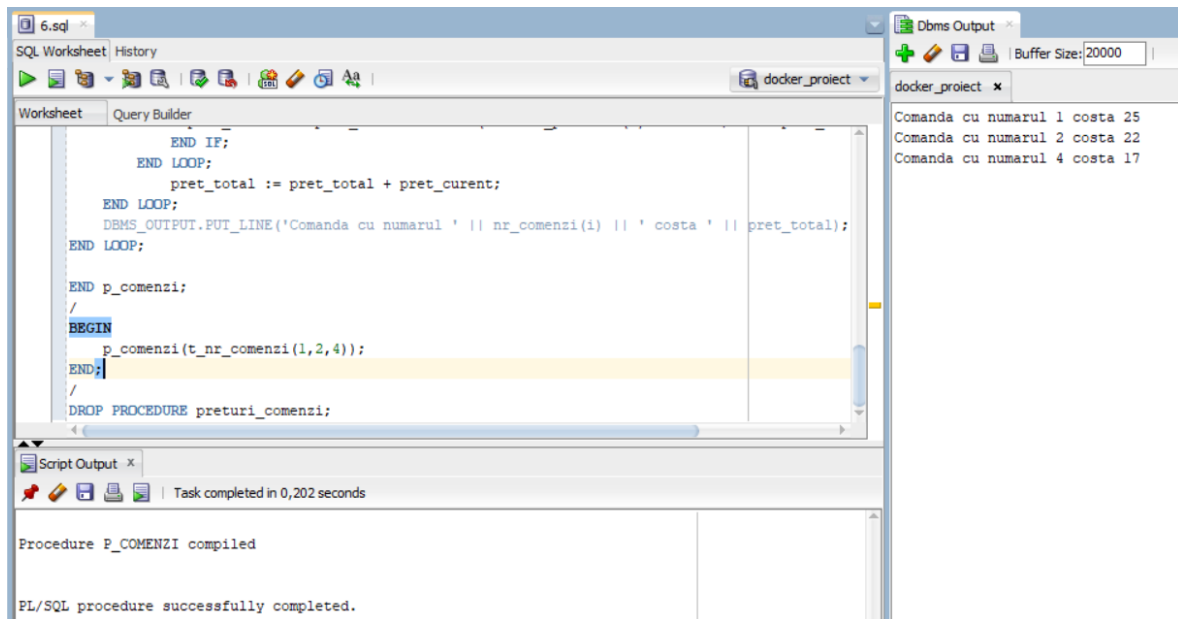
END p_comenzi;

/

BEGIN

    p_comenzi(t_nr_comenzi(1,2,4));

END;
```



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

Să se definească o procedură care generează unui raport cu informații despre produsele comandate într-o anumită perioadă.

CREATE OR REPLACE PROCEDURE produse\_in\_perioada(

data\_inceput DATE,

data\_sfarsit DATE

)

IS

-- cursor care obtine toate comenzile într-o anumita perioada

CURSOR c\_com IS

SELECT \*

FROM comanda

WHERE comanda.data BETWEEN data\_inceput AND data\_sfarsit;

-- cursor care obtine produsele din fiecare comanda din primul cursos

CURSOR c\_prod\_com (nr\_com comanda.nr\_comanda%TYPE) IS

SELECT p.id\_produs, p.denumire,

MIN(CASE

WHEN c.data BETWEEN pr.data\_start AND pr.data\_fin THEN

pret\_promo\_produs

```
        ELSE
            pret
        END) AS pret_curent
FROM produs p, promotie pr, promotii_produce prp, comanda c, produse_comanda pc
WHERE p.id_produs = prp.id_produs AND pr.id_promotie = prp.id_promotie AND
c.nr_comanda = nr_com
AND pc.nr_comanda = c.nr_comanda AND pc.id_produs = p.id_produs
GROUP BY p.id_produs, p.denumire;

nr_com comanda.nr_comanda%TYPE;
ang angajat%ROWTYPE;
BEGIN
    FOR com IN c_com LOOP
        nr_com := com.nr_comanda;

        SELECT * INTO ang
        FROM angajat a
        WHERE com.id_angajat = a.id_angajat;

        FOR prod IN c_prod_com(nr_com) LOOP
            DBMS_OUTPUT.PUT_LINE('Produsul cu denumirea ' || prod.denumire || ' a fost
cumparat la data de ' || com.data);

            DBMS_OUTPUT.PUT_LINE('intr-o comanda preluata de catre angajatul ' ||
ang.numa || ' ' || ang.prenume);

            DBMS_OUTPUT.PUT_LINE('la cafeaua cu id-ul ' || ang.cod_cafenea || ' si a costat
' || prod.pret_curent || ' RON.');
```

```
            DBMS_OUTPUT.PUT_LINE("");
        END LOOP;
    END LOOP;
END produse_in_perioada;

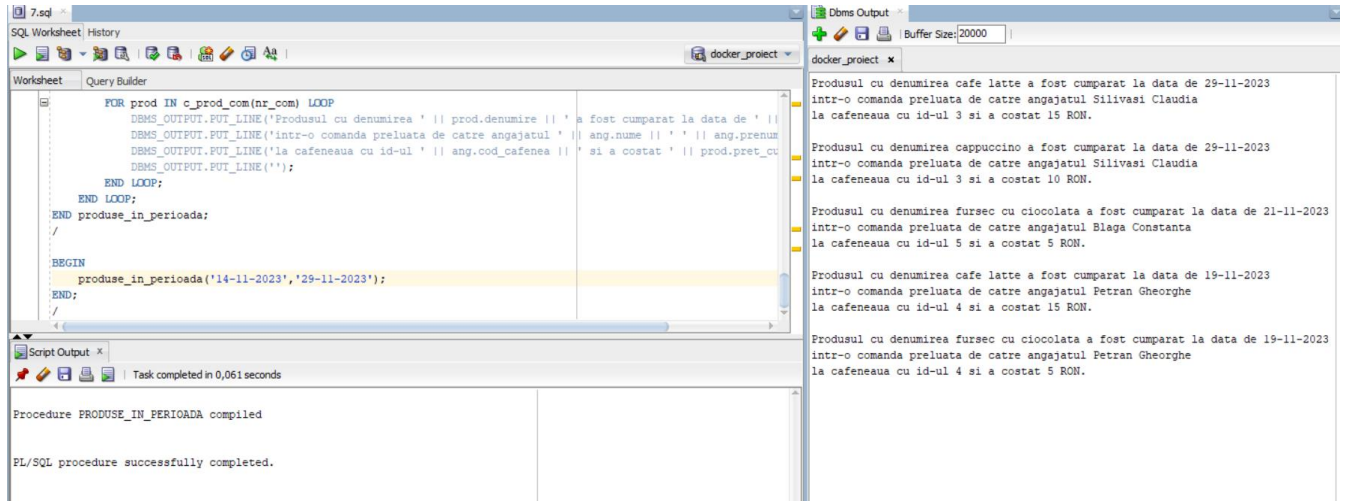
/
```



BEGIN

    produse\_in\_perioada('14-11-2023','29-11-2023');

END;



8) Formulati în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat independent de tip funcție care să utilizeze într-o singură comandă SQL 3 dintre tabelele definite. Definiți minim 2 excepții proprii. Apelați subprogramul astfel încât să evidențiați toate cazurile definite și tratate.

Să se definească p funcție care returnează furnizorii care au aprovizionat o cafea data ca parametru.

CREATE OR REPLACE TYPE t\_furn IS TABLE OF VARCHAR2(100);

/

CREATE OR REPLACE FUNCTION furnizori\_cafenea(caf cafenea.cod\_cafenea%TYPE)

RETURN t\_furn

AS

furnizori t\_furn;

TYPE t\_meniu IS TABLE OF meniu.id\_produs%TYPE;

men t\_meniu;

meniu\_gol EXCEPTION;

cafenea\_inexistenta EXCEPTION;

v\_exists NUMBER;

BEGIN

    SELECT CASE WHEN EXISTS (SELECT 1 FROM cafenea WHERE cod\_cafenea = caf)  
    THEN 1 ELSE 0 END

SGBD  
Seria 24

INTO v\_exists

FROM dual;

IF v\_exists = 0 THEN

RAISE cafenea\_inexistenta;

END IF;

SELECT id\_produc

BULK COLLECT INTO men

FROM meniu

WHERE cod\_cafenea = caf;

IF men.COUNT = 0 THEN

RAISE meniu\_gol;

END IF;

SELECT f.nume

BULK COLLECT INTO furnizori

FROM produs p, meniu m, furnizor f

WHERE m.cod\_cafenea = caf AND p.id\_produc = m.id\_produc AND p.cod\_furnizor =  
f.cod\_furnizor;

RETURN furnizori;

EXCEPTION

WHEN cafenea\_inexistenta THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Nu exista cafeaua data');

WHEN meniu\_gol THEN

RAISE\_APPLICATION\_ERROR(-20002, 'La cafeaua data nu se vinde niciun  
produc');

END furnizori\_cafenea;

/

SGBD  
Seria 24

DECLARE

furnizori t\_furn;

id cafenea.cod\_cafenea%TYPE;

BEGIN

id := 2;

furnizori := furnizori\_cafenea(id);

FOR i IN 1..furnizori.count LOOP

DBMS\_OUTPUT.PUT\_LINE(furnizori(i));

END LOOP;

END;

/

DECLARE

furnizori t\_furn;

id cafenea.cod\_cafenea%TYPE;

BEGIN

id := 7;

furnizori := furnizori\_cafenea(id);

FOR i IN 1..furnizori.count LOOP

DBMS\_OUTPUT.PUT\_LINE(furnizori(i));

END LOOP;

END;

/

DECLARE

furnizori t\_furn;

id cafenea.cod\_cafenea%TYPE;

BEGIN

id := 3;

furnizori := furnizori\_cafenea(id);

FOR i IN 1..furnizori.count LOOP

DBMS\_OUTPUT.PUT\_LINE(furnizori(i));

```
END LOOP;  
  
END;  
  
/
```

The screenshot shows the SQL Developer interface with a script named '8.sql' being executed. The script contains a PL/SQL block that declares a table type, initializes a variable, and loops through a table to output its contents. The 'Dbms Output' window on the right shows the results of the execution, listing the contents of the 'furnizori' table. The 'Script Output' window at the bottom indicates that the task was completed successfully in 0.035 seconds.

```
RAISE_APPLICATION_ERROR(-20002, 'La cafeaua data nu se vinde niciun produs');  
END furnizori_cafenea;  
/  
DECLARE  
    furnizori t_furn;  
    id cafeana.cod_cafenea%TYPE;  
BEGIN  
    id := 2;  
    furnizori := furnizori_cafenea(id);  
    FOR i IN 1..furnizori.count LOOP  
        DBMS_OUTPUT.PUT_LINE(furnizori(i));  
    END LOOP;  
END;  
/  
DECLARE
```

De Toate  
La Cafele  
Magazinul cu Produse

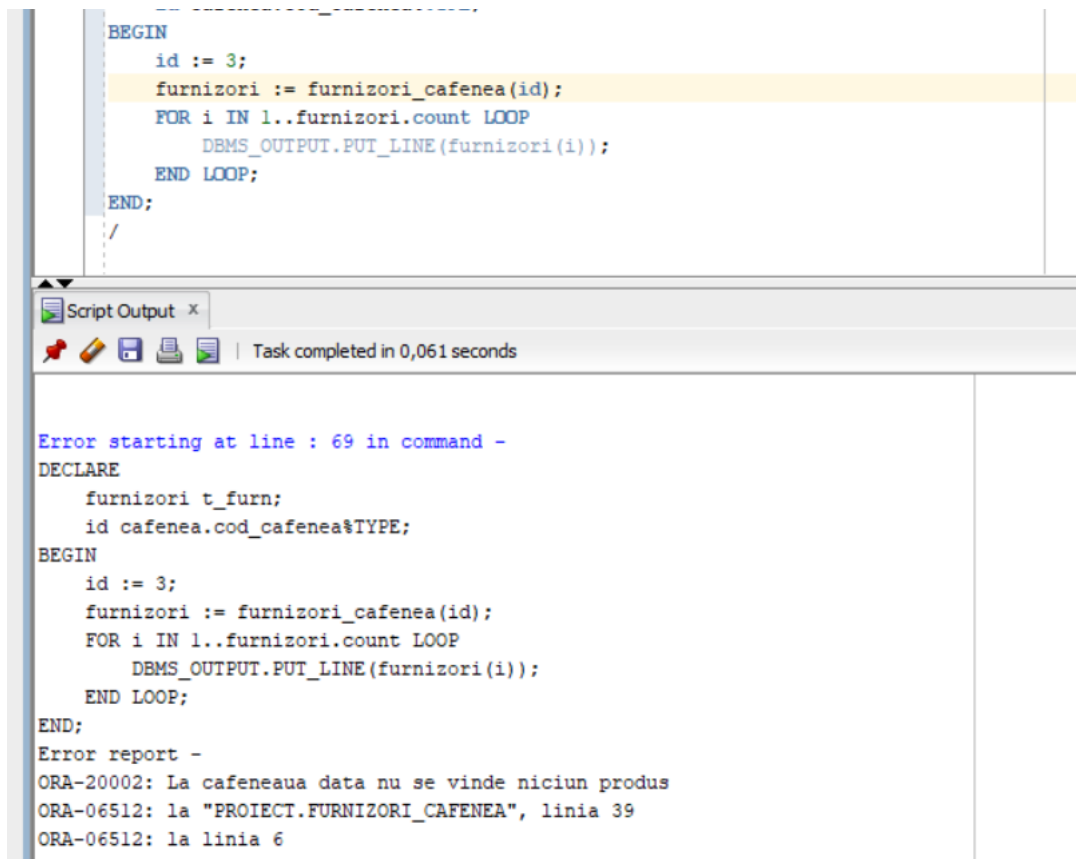
Task completed in 0,035 seconds

Function FURNIZORI\_CAFENEA compiled

PL/SQL procedure successfully completed.

The screenshot shows the same SQL Developer interface, but with a different script. This script has a typo in the table name 'cafeana' instead of 'cafenea'. The 'Script Output' window at the bottom shows an error report with three messages: a general error at line 57, and two specific errors (ORA-20001 and ORA-06512) pointing to lines 37 and 6 respectively.

```
DECLARE  
    furnizori t_furn;  
    id cafeana.cod_cafenea%TYPE;  
BEGIN  
    id := 7;  
    furnizori := furnizori_cafenea(id);  
    FOR i IN 1..furnizori.count LOOP  
        DBMS_OUTPUT.PUT_LINE(furnizori(i));  
    END LOOP;  
END;  
Error report -  
ORA-20001: Nu exista cafeaua data  
ORA-06512: la "PROIECT.FURNIZORI_CAFENEA", linia 37  
ORA-06512: la linia 6
```



```
-----  
BEGIN  
  id := 3;  
  furnizori := furnizori_cafenea(id);  
  FOR i IN 1..furnizori.count LOOP  
    DBMS_OUTPUT.PUT_LINE(furnizori(i));  
  END LOOP;  
END;  
/  
  
Script Output x  
Task completed in 0,061 seconds  
  
Error starting at line : 69 in command -  
DECLARE  
  furnizori t_furn;  
  id cafenea.cod_cafenea%TYPE;  
BEGIN  
  id := 3;  
  furnizori := furnizori_cafenea(id);  
  FOR i IN 1..furnizori.count LOOP  
    DBMS_OUTPUT.PUT_LINE(furnizori(i));  
  END LOOP;  
END;  
Error report -  
ORA-20002: La cafeneaua data nu se vinde niciun produs  
ORA-06512: la "PROIECT.FURNIZORI_CAFENEA", linia 39  
ORA-06512: la linia 6
```

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.

Să se definească o procedură care primește ca parametru o denumire a unui produs și un id de cafenea, iar dacă produsul a fost vândut într-o comandă din cafeneaua dată, îl adaugă în meniul cafenelei.

CREATE OR REPLACE PROCEDURE adauga\_meniu (prod produs.denumire%TYPE, caf  
cafenea.cod\_cafenea%TYPE)

IS

```
v_id_prod produs.id_produs%TYPE;  
TYPE t_tabel_produce IS TABLE OF produs.id_produs%TYPE;  
tabel_produce t_tabel_produce;  
prod_meniu t_tabel_produce;  
exista EXCEPTION;  
nu_vandut EXCEPTION;  
v_exista NUMBER;  
v_vandut NUMBER;
```

```
e_caf NUMBER;

BEGIN

    --luam id-ul produsului dat
    SELECT id_produs INTO v_id_prod
    FROM produs
    WHERE denumire = prod;

    --verificam daca exista cafeaua
    SELECT 1 INTO e_caf
    FROM cafea
    WHERE cod_cafea = caf;

    --verificam daca produsul a fost vandut la cafeaua data
    SELECT count(*) INTO v_vandut
    FROM comanda c, angajat a, produse_comanda pc
    WHERE a.cod_cafea = caf AND c.id_angajat = a.id_angajat
    AND c.nr_comanda = pc.nr_comanda AND pc.id_produs = v_id_prod;

    IF v_vandut = 0 THEN
        RAISE nu_vandut;
    END IF;

    --cautam produsele comandate care apar in meniu
    SELECT p.id_produs BULK COLLECT INTO prod_meniu
    FROM meniu m, angajat a, comanda c, produs p, produse_comanda pc
    WHERE m.cod_cafea = caf AND a.cod_cafea = caf AND
    c.id_angajat = a.id_angajat AND c.nr_comanda = pc.nr_comanda
    AND p.id_produs = pc.id_produs AND p.id_produs = m.id_produs
    GROUP BY p.id_produs;

    --verificam daca produsul dat exista in meniul cafelei, iar daca nu exista este adaugat
    v_exista := 0;

    FOR i IN prod_meniu.FIRST..prod_meniu.LAST LOOP
```

SGBD  
Seria 24

```
        IF v_id_prod = prod_meniu(i) THEN
            v_exista := 1;
        END IF;
    END LOOP;
    IF v_exista = 0 THEN
        INSERT INTO meniu (cod_cafenea, id_produs)
        VALUES(caf, v_id_prod);
        DBMS_OUTPUT.PUT_LINE('Produsul ' || prod || ' a fost adaugat in meniul cafelei ' ||
caf);
    ELSE
        RAISE exista;
    END IF;

EXCEPTION

    WHEN NO_DATA_FOUND THEN
        RAISE_APPLICATION_ERROR(-20001, 'Produsul si/sau cafeaua nu exista');
    WHEN TOO_MANY_ROWS THEN
        RAISE_APPLICATION_ERROR(-20002, 'Sunt mai multe produse cu denumirea data');
    WHEN exista THEN
        RAISE_APPLICATION_ERROR(-20003, 'Produsul exista deja in meniul cafelei');
    WHEN nu_vandut THEN
        RAISE_APPLICATION_ERROR(-20004, 'Produsul nu a fost vandut la cafeaua data');
    WHEN OTHERS THEN
        RAISE_APPLICATION_ERROR(-20005, 'Alta eroare');
END;

/

BEGIN
    adauga_meniu('tiramisu',7);
END;

/
```

SGBD  
Seria 24

BEGIN

INSERT INTO PRODUS

VALUES(6,'tiramisu',10,10,2);

adauga\_meniu('tiramisu',1);

END;

/

select \* from produs;

BEGIN

adauga\_meniu('fursec cu ciocolata',5);

END;

/

BEGIN

adauga\_meniu('tiramisu',1);

END;

/

BEGIN

adauga\_meniu('cafe latte',4);

END;

/



```
178
179 BEGIN
180   adauga_meniu('tiramisu',7);
181 END;
182 /
183 BEGIN
184   INSERT INTO PRODUS
185   VALUES(6,'tiramisu',10,10,2);
186   adauga_meniu('tiramisu',1);
187 END;
188 /
189 ROLLBACK;
190
191 BEGIN
```

Script Output x Query Result x

Task completed in 0,068 seconds

Procedure ADAUGA\_MENIU compiled

Error starting at line : 179 in command -  
BEGIN  
adauga\_meniu('tiramisu',7);  
END;  
Error report -  
ORA-20001: Produsul si/sau cafeaneaua nu exista  
ORA-06512: la "PROIECT.ADAUGA\_MENIU", linia 56  
ORA-06512: la linia 2

```
184 INSERT INTO PRODUS
185 VALUES(6,'tiramisu',10,10,2);
186 adauga_meniu('tiramisu',1);
187 END;
188 /
189 select * from produs;
190
191 BEGIN
192   adauga_meniu('fursec cu ciocolata',5);
193 END;
194 /
195 BEGIN
```

Script Output x Query Result x

Task completed in 0,078 seconds

Error starting at line : 183 in command -  
BEGIN  
INSERT INTO PRODUS  
VALUES(6,'tiramisu',10,10,2);  
adauga\_meniu('tiramisu',1);  
END;  
Error report -  
ORA-20002: Sunt mai multe produse cu denumirea data  
ORA-06512: la "PROIECT.ADAUGA\_MENIU", linia 58  
ORA-06512: la linia 4

```
190
191 BEGIN
192 adauga_meniu('fursec cu ciocolata',5);
193 END;
194 /
195 BEGIN
196 adauga_meniu('tiramisu',1);
197 END;
198 /
199 BEGIN
200 adauga_meniu('cafe latte',4);
201 END;
```

Script Output x Query Result x

Task completed in 0,077 seconds

ORA-06512: la linia 4

Error starting at line : 191 in command -  
BEGIN  
adauga\_meniu('fursec cu ciocolata',5);  
END;  
Error report -  
ORA-20003: Produsul exista deja in meniul cafelelei  
ORA-06512: la "PROIECT.ADAUGA\_MENIU", linia 60  
ORA-06512: la linia 2

```
194 /
195 BEGIN
196 adauga_meniu('tiramisu',1);
197 END;
198 /
199 BEGIN
200 adauga_meniu('cafe latte',4);
201 END;
```

Script Output x Query Result x

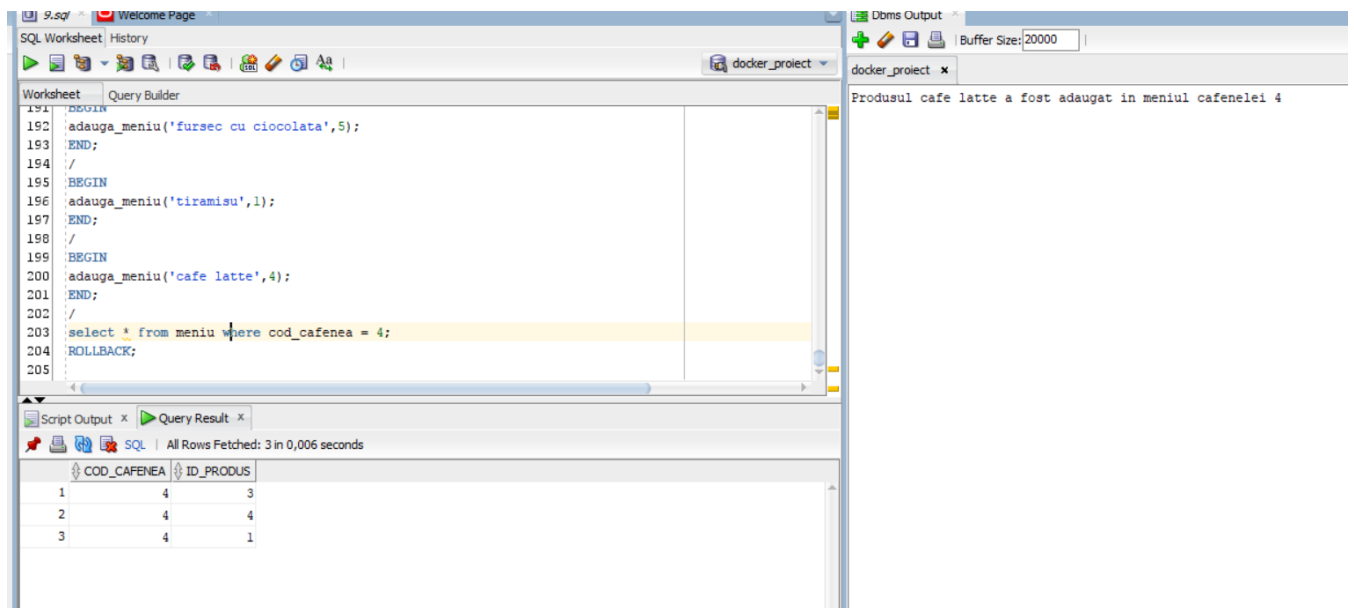
Task completed in 0,072 seconds

ORA-06512: la linia 2

Error starting at line : 195 in command -  
BEGIN  
adauga\_meniu('tiramisu',1);  
END;  
Error report -  
ORA-20004: Produsul nu a fost vandut la cafenea data  
ORA-06512: la "PROIECT.ADAUGA\_MENIU", linia 62  
ORA-06512: la linia 2

## SGBD

### Seria 24



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

Să se definească un trigger care nu permite inserarea a mai mult de 15 linii in tabela de angajati.

```
CREATE OR REPLACE TRIGGER max_promotii
```

```
BEFORE INSERT ON angajat
```

```
DECLARE
```

```
nr INT;
```

```
BEGIN
```

```
    SELECT COUNT(*)
```

```
    INTO nr
```

```
    FROM angajat;
```

```
    IF nr > 15 THEN
```

```
        RAISE_APPLICATION_ERROR(-20001, 'Nu se pot angaja mai mult de 15 persoane');
```

```
    END IF;
```

```
END;
```

```
/
```

```
BEGIN
```

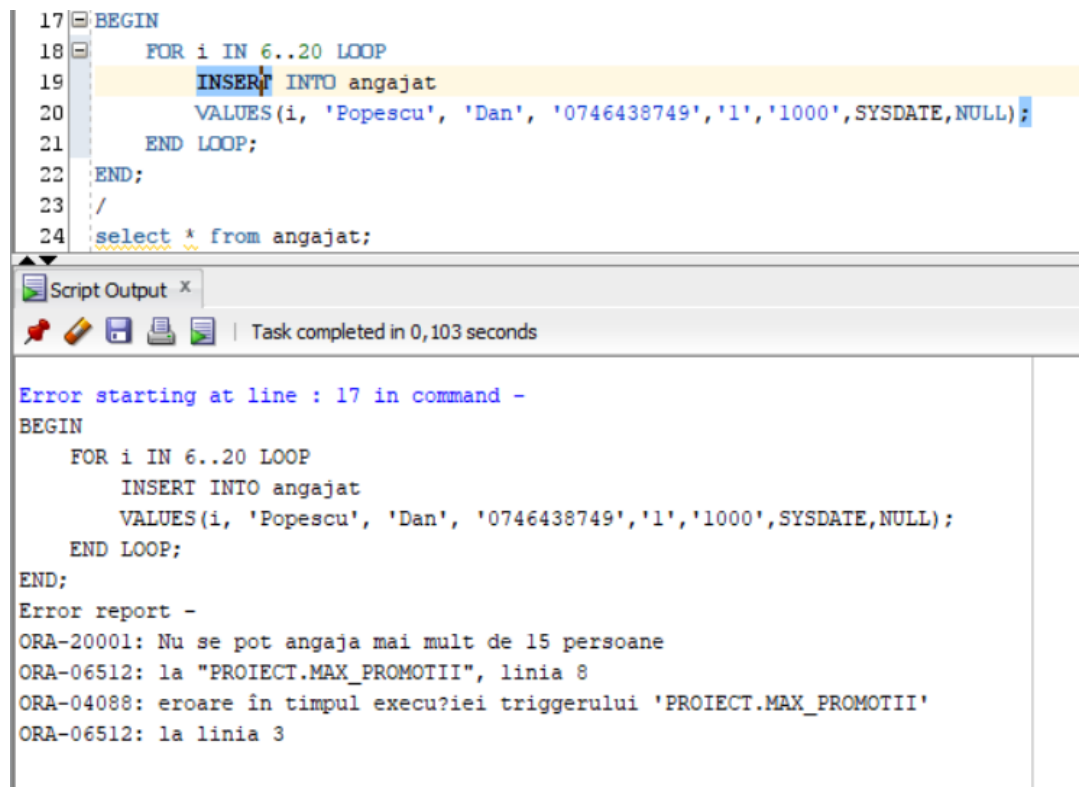
```
    FOR i IN 6..20 LOOP
```

```
        INSERT INTO angajat
```

```
        VALUES(i, 'Popescu', 'Dan', '0746438749','1','1000',SYSDATE,NULL);
```

```
    END LOOP;
```

END;



The screenshot shows a SQL script in a text editor and its execution output in a separate window. The script is as follows:

```
17 BEGIN
18   FOR i IN 6..20 LOOP
19     INSERT INTO angajat
20       VALUES(i, 'Popescu', 'Dan', '0746438749','1','1000',SYSDATE,NULL);
21   END LOOP;
22 END;
23 /
24 select * from angajat;
```

The 'Script Output' window shows the following error messages:

```
Error starting at line : 17 in command -
BEGIN
  FOR i IN 6..20 LOOP
    INSERT INTO angajat
      VALUES(i, 'Popescu', 'Dan', '0746438749','1','1000',SYSDATE,NULL);
  END LOOP;
END;
Error report -
ORA-20001: Nu se pot angaja mai mult de 15 persoane
ORA-06512: la "PROIECT.MAX_PROMOTII", linia 8
ORA-04088: eroare în timpul execuției triggerului 'PROIECT.MAX_PROMOTII'
ORA-06512: la linia 3
```

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

Definiți un trigger care atunci când se inserează o comanda, mărește cu 5% salariul angajatului care a preluat comanda.

CREATE OR REPLACE TRIGGER marire\_salariu

AFTER INSERT OR UPDATE ON comanda

FOR EACH ROW

DECLARE

v\_salariu angajat.salariu%TYPE;

BEGIN

SELECT salariu INTO v\_salariu

FROM angajat

WHERE id\_angajat = :NEW.id\_angajat;

DBMS\_OUTPUT.PUT\_LINE('Salariul angajatul cu id ' || :NEW.id\_angajat || ' a crescut de la '

|| v\_salariu || ' la ' || v\_salariu \* 1.05);

UPDATE angajat

## SGBD Seria 24

SET salariu = salariu \* 1.05

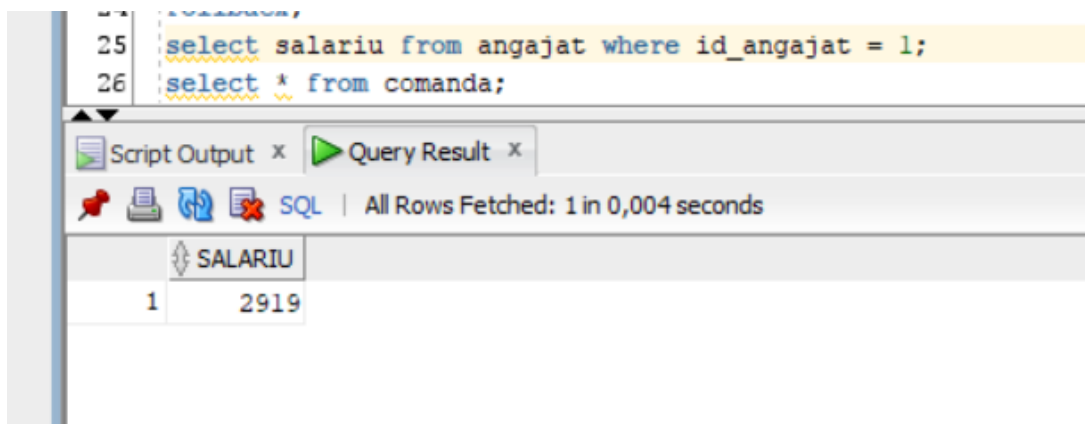
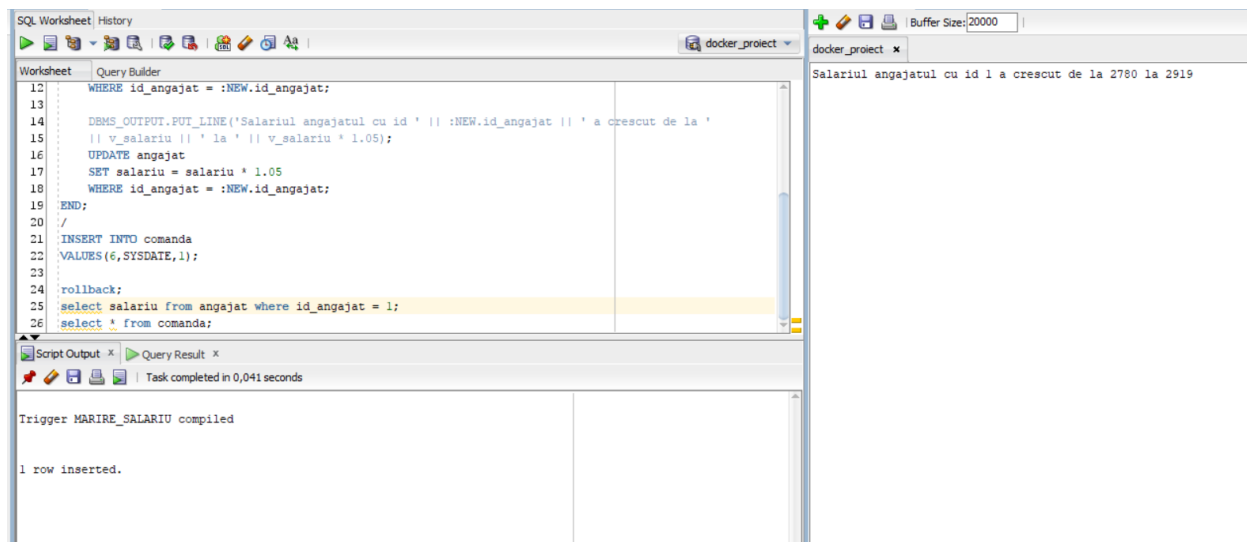
WHERE id\_angajat = :NEW.id\_angajat;

END;

/

INSERT INTO comanda

VALUES(6,SYSDATE,1);



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

Să se definească un trigger care afișează un mesaj corespunzător atunci când se adaugă, alterează sau șterge un obiect din baza de date.

CREATE OR REPLACE TRIGGER schimbare

AFTER CREATE OR ALTER OR DROP ON SCHEMA

DECLARE

v\_numa\_tabel VARCHAR2(50);

BEGIN

```
v_num_tabel := SYS.DICTIONARY_OBJ_NAME;

IF SYS.SYSEVENT = 'CREATE' THEN

    DBMS_OUTPUT.PUT_LINE('Utilizatorul ' || SYS.LOGIN_USER || ' a creat obiectul ' ||
v_num_tabel || ' de tip ' || SYS.DICTIONARY_OBJ_TYPE);

ELSIF SYS.SYSEVENT = 'ALTER' THEN

    DBMS_OUTPUT.PUT_LINE('Utilizatorul ' || SYS.LOGIN_USER || ' a modificat obiectul ' ||
v_num_tabel || ' de tip ' || SYS.DICTIONARY_OBJ_TYPE);

ELSIF SYS.SYSEVENT = 'DROP' THEN

    DBMS_OUTPUT.PUT_LINE('Utilizatorul ' || SYS.LOGIN_USER || ' a sters obiectul ' ||
v_num_tabel || ' de tip ' || SYS.DICTIONARY_OBJ_TYPE);

END IF;

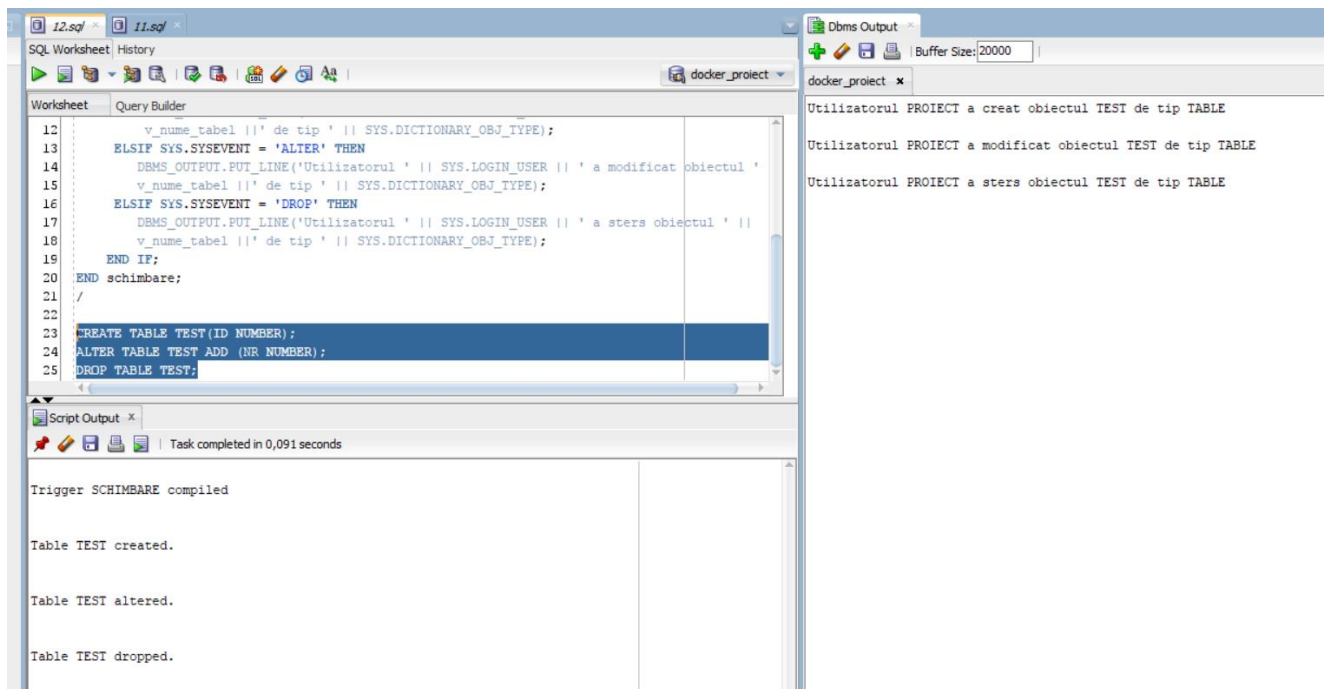
END schimbare;

/

CREATE TABLE TEST(ID NUMBER);

ALTER TABLE TEST ADD (NR NUMBER);

DROP TABLE TEST;
```



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

CREATE OR REPLACE PACKAGE pachet\_proiect AS

SGBD  
Seria 24

```
TYPE t_nr_comenzi IS TABLE OF NUMBER(4);
PROCEDURE p_comenzi (nr_comenzi t_nr_comenzi);
PROCEDURE produse_in_perioada(data_inceput DATE, data_sfarsit DATE);
TYPE t_furn IS TABLE OF VARCHAR2(100);
FUNCTION furnizori_cafenea(caf cafenea.cod_cafenea%TYPE)
RETURN t_furn;
PROCEDURE adauga_meniu (prod produs.denumire%TYPE, caf
cafenea.cod_cafenea%TYPE);
END pachet_proiect;
/

CREATE OR REPLACE PACKAGE BODY pachet_proiect AS
--6
PROCEDURE p_comenzi (nr_comenzi t_nr_comenzi)
IS
TYPE t_id_prod IS VARRAY(10) OF produs.id_produs%TYPE; --vector
produse_in_comanda t_id_prod;
detalii_comanda comanda%ROWTYPE;
pret_total NUMBER;
pret_curent produs.pret%TYPE;
TYPE t_promo IS TABLE OF promotie%ROWTYPE INDEX BY PLS_INTEGER; --
tabel indexat
detalii_promotie t_promo;
BEGIN
FOR i in nr_comenzi.FIRST..nr_comenzi.LAST LOOP --parcurea comenzilor
--se preiau detaliile comenzilor
SELECT *
INTO detalii_comanda
FROM comanda WHERE nr_comanda = nr_comenzi(i);
--determinarea produselor dintr-o comanda
SELECT pc.id_produs
BULK COLLECT INTO produse_in_comanda
```

```
FROM produs p, produse_comanda pc
WHERE p.id_produs = pc.id_produs AND pc.nr_comanda = nr_comenzi(i);
--se verifica pt fiecare produs daca este aplicata vreo promotie la data comenzii
pret_total := 0;
FOR j IN produse_in_comanda.FIRST..produse_in_comanda.LAST LOOP
    -- se selecteaza toate promotiile care au fost aplicate unui produs
    SELECT p.*
    BULK COLLECT INTO detalii_promotie
    FROM promotie p, promotii_produce pr
    WHERE pr.id_produs = produse_in_comanda(j) AND pr.id_promotie =
p.id_promotie;

    SELECT pret
    INTO pret_curent
    FROM produs
    WHERE id_produs = produse_in_comanda(j);
    -- se parcurg promotiile si se verifica daca este una din ele activa la data comenzii
curente
    -- si se adauga reducerea in acest caz
    FOR k IN detalii_promotie.FIRST..detalii_promotie.LAST LOOP
        IF detalii_comanda.data BETWEEN detalii_promotie(k).data_start AND
detalii_promotie(k).data_fin THEN
            pret_curent := pret_curent - ROUND(detalii_promotie(k).reducere/100 *
pret_curent);
            END IF;
        END LOOP;
        pret_total := pret_total + pret_curent;
    END LOOP;

    DBMS_OUTPUT.PUT_LINE('Comanda cu numarul ' || nr_comenzi(i) || ' costa ' ||
pret_total);
END LOOP;

END p_comenzi;
```



--7

```
PROCEDURE produse_in_perioada(
    data_inceput DATE,
    data_sfarsit DATE
)
IS
-- cursor care obtine toate comenzile într-o anumita perioada
    CURSOR c_com IS
    SELECT *
    FROM comanda
    WHERE comanda.data BETWEEN data_inceput AND data_sfarsit;
-- cursor care obtine produsele din fiecare comanda din primul cursos
    CURSOR c_prod_com (nr_com comanda.nr_comanda%TYPE) IS
    SELECT p.id_produc, p.denumire,
    MIN(CASE
        WHEN c.data BETWEEN pr.data_start AND pr.data_fin THEN
            pret_promo_produc
        ELSE
            pret
    END) AS pret_curent
    FROM produs p, promotie pr, promotii_produce prp, comanda c, produse_comanda pc
    WHERE p.id_produc = prp.id_produc AND pr.id_promotie = prp.id_promotie AND
c.nr_comanda = nr_com
    AND pc.nr_comanda = c.nr_comanda AND pc.id_produc = p.id_produc
    GROUP BY p.id_produc, p.denumire;

    nr_com comanda.nr_comanda%TYPE;
    ang angajat%ROWTYPE;
BEGIN
    FOR com IN c_com LOOP
        nr_com := com.nr_comanda;
```

```
SELECT * INTO ang
FROM angajat a
WHERE com.id_angajat = a.id_angajat;

FOR prod IN c_prod_com(nr_com) LOOP
    DBMS_OUTPUT.PUT_LINE('Produsul cu denumirea ' || prod.denumire || ' a fost
cumparat la data de ' || com.data);

    DBMS_OUTPUT.PUT_LINE('intr-o comanda preluata de catre angajatul ' ||
ang.ume || ' ' || ang.prenume);

    DBMS_OUTPUT.PUT_LINE('la cafeaua cu id-ul ' || ang.cod_cafenea || ' si a
costat ' || prod.pret_curent || ' RON.');
```

```
    DBMS_OUTPUT.PUT_LINE("");
END LOOP;
END LOOP;
END produse_in_perioada;
--8
FUNCTION furnizori_cafenea(caf cafenea.cod_cafenea%TYPE)
RETURN t_furn
IS
furnizori t_furn;
TYPE t_meniu IS TABLE OF meniu.id_produs%TYPE;
men t_meniu;
meniu_gol EXCEPTION;
cafenea_inexistenta EXCEPTION;
v_exists NUMBER;
BEGIN
    SELECT CASE WHEN EXISTS (SELECT 1 FROM cafenea WHERE cod_cafenea =
caf) THEN 1 ELSE 0 END
    INTO v_exists
    FROM dual;
```

SGBD  
Seria 24

```
IF v_exists = 0 THEN
    RAISE cafenea_inexistenta;
END IF;

SELECT id_produc
BULK COLLECT INTO men
FROM meniu
WHERE cod_cafenea = caf;

IF men.COUNT = 0 THEN
    RAISE meniu_gol;
END IF;

SELECT f.nume
BULK COLLECT INTO furnizori
FROM produs p, meniu m, furnizor f
WHERE m.cod_cafenea = caf AND p.id_produc = m.id_produc AND p.cod_furnizor =
f.cod_furnizor;

RETURN furnizori;

EXCEPTION
    WHEN cafenea_inexistenta THEN
        RAISE_APPLICATION_ERROR(-20001, 'Nu exista cafeneaua data');
    WHEN meniu_gol THEN
        RAISE_APPLICATION_ERROR(-20002, 'La cafeneaua data nu se vinde niciun
produs');
END furnizori_cafenea;
--9

PROCEDURE adauga_meniu (prod produs.denumire%TYPE, caf
cafenea.cod_cafenea%TYPE)
IS
```

```
v_id_prod produs.id_produs%TYPE;
TYPE t_tabel_produce IS TABLE OF produs.id_produs%TYPE;
tabel_produce t_tabel_produce;
prod_meniu t_tabel_produce;
exista EXCEPTION;
nu_vandut EXCEPTION;
v_exista NUMBER;
v_vandut NUMBER;
e_caf NUMBER;
BEGIN
    --luam id-ul produsului dat
    SELECT id_produs INTO v_id_prod
    FROM produs
    WHERE denumire = prod;
    --verificam daca exista cafeaua
    SELECT 1 INTO e_caf
    FROM cafea
    WHERE cod_cafea = caf;
    --verificam daca produsul a fost vandut la cafeaua data
    SELECT count(*) INTO v_vandut
    FROM comanda c, angajat a, produse_comanda pc
    WHERE a.cod_cafea = caf AND c.id_angajat = a.id_angajat
    AND c.nr_comanda = pc.nr_comanda AND pc.id_produs = v_id_prod;

    IF v_vandut = 0 THEN
        RAISE nu_vandut;
    END IF;

    --cautam produsele comdate care apar in meniu
    SELECT p.id_produs BULK COLLECT INTO prod_meniu
    FROM meniu m, angajat a, comanda c, produs p, produse_comanda pc
```

```
WHERE m.cod_cafenea = caf AND a.cod_cafenea = caf AND
c.id_angajat = a.id_angajat AND c.nr_comanda = pc.nr_comanda
AND p.id_produș = pc.id_produș AND p.id_produș = m.id_produș
GROUP BY p.id_produș;

--verificam daca produsul dat exista in meniul cafelei, iar daca nu exista este adaugat
v_exista := 0;
FOR i IN prod_meniu.FIRST..prod_meniu.LAST LOOP
    IF v_id_prod = prod_meniu(i) THEN
        v_exista := 1;
    END IF;
END LOOP;
IF v_exista = 0 THEN
    INSERT INTO meniu (cod_cafenea, id_produș)
    VALUES(caf, v_id_prod);
    DBMS_OUTPUT.PUT_LINE('Produsul ' || prod || ' a fost adaugat in meniul cafelei
' || caf);
ELSE
    RAISE exista;
END IF;

EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RAISE_APPLICATION_ERROR(-20001, 'Produsul si/sau cafeaua nu exista');
    WHEN TOO_MANY_ROWS THEN
        RAISE_APPLICATION_ERROR(-20002, 'Sunt mai multe produse cu denumirea
data');
    WHEN exista THEN
        RAISE_APPLICATION_ERROR(-20003, 'Produsul exista deja in meniul cafelei');
    WHEN nu_vandut THEN
        RAISE_APPLICATION_ERROR(-20004, 'Produsul nu a fost vandut la cafeaua
data');
```

SGBD  
Seria 24

```
        WHEN OTHERS THEN

            RAISE_APPLICATION_ERROR(-20005, 'Alta eroare');

        END adauga_meniu;
    END pachet_proiect;

/

--testare

EXECUTE proiect.p_comenzi(t_nr_comenzi(1,2,4));
EXECUTE pachet_proiect.produce_in_perioada('14-11-2023','29-11-2023');
DECLARE

    furnizori pachet_proiect.t_furn;

    id cafenea.cod_cafenea%TYPE;
BEGIN
    id := 2;

    furnizori := pachet_proiect.furnizori_cafenea(id);

    FOR i IN 1..furnizori.count LOOP

        DBMS_OUTPUT.PUT_LINE(furnizori(i));

    END LOOP;
END;

/

DECLARE

    furnizori pachet_proiect.t_furn;

    id cafenea.cod_cafenea%TYPE;
BEGIN
    id := 7;

    furnizori := pachet_proiect.furnizori_cafenea(id);

    FOR i IN 1..furnizori.count LOOP

        DBMS_OUTPUT.PUT_LINE(furnizori(i));

    END LOOP;
END;

/

DECLARE
```

SGBD  
Seria 24

```
furnizori pachet_proiect.t_furn;
id cafenea.cod_cafenea%TYPE;
BEGIN
    id := 3;
    furnizori := pachet_proiect.furnizori_cafenea(id);
    FOR i IN 1..furnizori.count LOOP
        DBMS_OUTPUT.PUT_LINE(furnizori(i));
    END LOOP;
END;
/
EXECUTE pachet_proiect.adauga_meniu('tiramisu',7);
EXECUTE pachet_proiect.adauga_meniu('fursec cu ciocolata',5);
EXECUTE pachet_proiect.adauga_meniu('tiramisu',1);
EXECUTE pachet_proiect.adauga_meniu('cafe latte',4);
EXECUTE pachet_proiect.adauga_meniu('tiramisu',7);
BEGIN
INSERT INTO PRODUS
VALUES(6,'tiramisu',10,10,2);
pachet_proiect.adauga_meniu('tiramisu',1);
END;
/
```

## SGBD

### Seria 24

SQL Worksheet | History

Worksheet | Query Builder

```
7 RETURN t_return;
8 PROCEDURE adauga_menu (prod produs.denumire%TYPE, caf cafea.cod_cafea%TYPE);
9 END pachet_proiect;
10 /
11
12 CREATE OR REPLACE PACKAGE BODY pachet_proiect AS
13 --6
14 PROCEDURE p_comenzi (nr_comenzi t_nr_comenzi)
15 IS
16 TYPE t_id_prod IS VARRAY(10) OF produs.id_produs%TYPE; --vector
17 produse_in_comanda t_id_prod;
18 detalii_comanda comanda%ROWTYPE;
19 pret_total NUMBER;
20 pret_curent produs.pret%TYPE;
21 TYPE t_promo IS TABLE OF promotie%ROWTYPE INDEX BY PLS_INTEGER; --tabel indexat
22 detalii_promotie t_promo;
23 BEGIN
24 FOR i in nr_comenzi.FIRST..nr_comenzi.LAST LOOP --parcurerea comenzilor
25 --se preiau detaliile comenzilor
26 SELECT *
```

Script Output x

Task completed in 0,048 seconds

Package PACHET\_PROIECT compiled

Package Body PACHET\_PROIECT compiled