

Gestionarea unui lanț de supermarket-uri

Smădu Andrei

Seria 24, Grupa 243

Anul 2025-2026

Cuprins:

1. Descrierea modelului real, a utilității acestuia și a regulilor de funcționare	4
2. Diagrama ERD (entity-relationship)	5
3. Diagrama conceptuală.....	6
4. Implementarea în Oracle a diagramei conceptuale realizate.....	7
5. Adăugarea de informații coerente în tabelele create.....	10
6. Formularea în limbaj natural a unei probleme care se rezolvă folosind un subprogram stocat independent care utilizează toate cele 3 tipuri de colecții studiate.....	22
7. Formularea în limbaj natural a unei probleme care se rezolvă folosind un subprogram stocat independent care utilizează 2 tipuri diferite de cursoare studiate, unul dintre acestea fiind cursor parametrizat, dependent de celălalt cursor.....	26
8. Formularea în limbaj natural a unei probleme care se rezolvă folosind un subprogram stocat independent de tip funcție care utilizează într-o singură comandă SQL 3 dintre tabelele create.....	28
9. Formularea în limbaj natural a unei probleme care se rezolvă folosind un subprogram stocat independent de tip procedură care primește minim 2 parametri și utilizează într-o singură comandă SQL 5 dintre tabelele create și definirea a minim 2 exceptii proprii....	34
10. Definirea unui trigger de tip LMD la nivel de comandă și declanșarea trigger-ului	38
11. Definirea unui trigger de tip LMD la nivel de linie și declanșarea trigger-ului.....	44
12. Definirea unui trigger de tip LDD și declanșarea trigger-ului.....	48
13. Formularea în limbaj natural a unei probleme care se rezolvă folosind un pachet care include tipuri de date complexe și obiecte necesare unui flux de acțiuni integrate	50

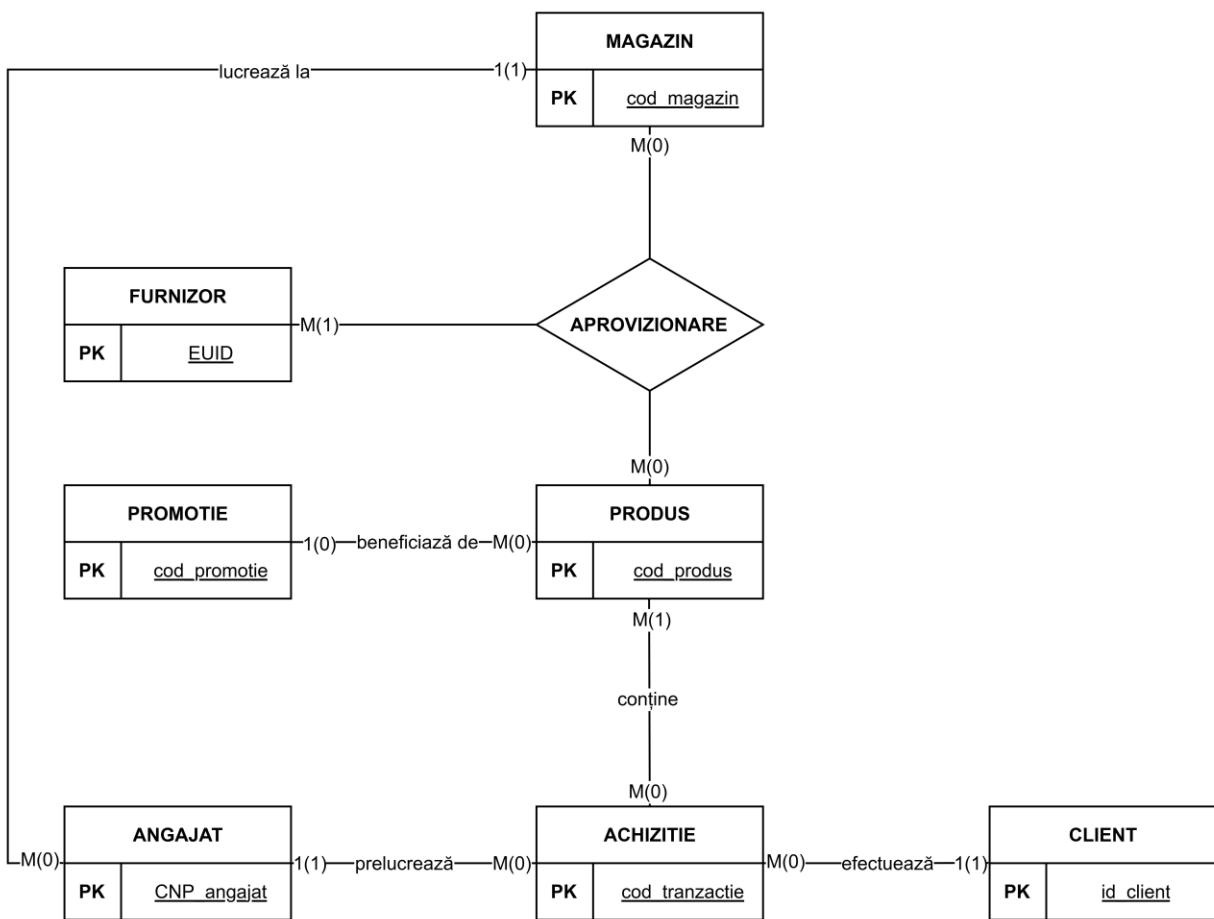
Introducere:

Pentru implementarea temei alese, un sistem de gestionare pentru un lanț de supermarket-uri, am utilizat motorul de baze de date Oracle Database 21c Express Edition. Ca mediu de lucru am utilizat Oracle SQL Developer 24.3 pe care l-am rulat pe sistemul de operare Windows 11 Pro.

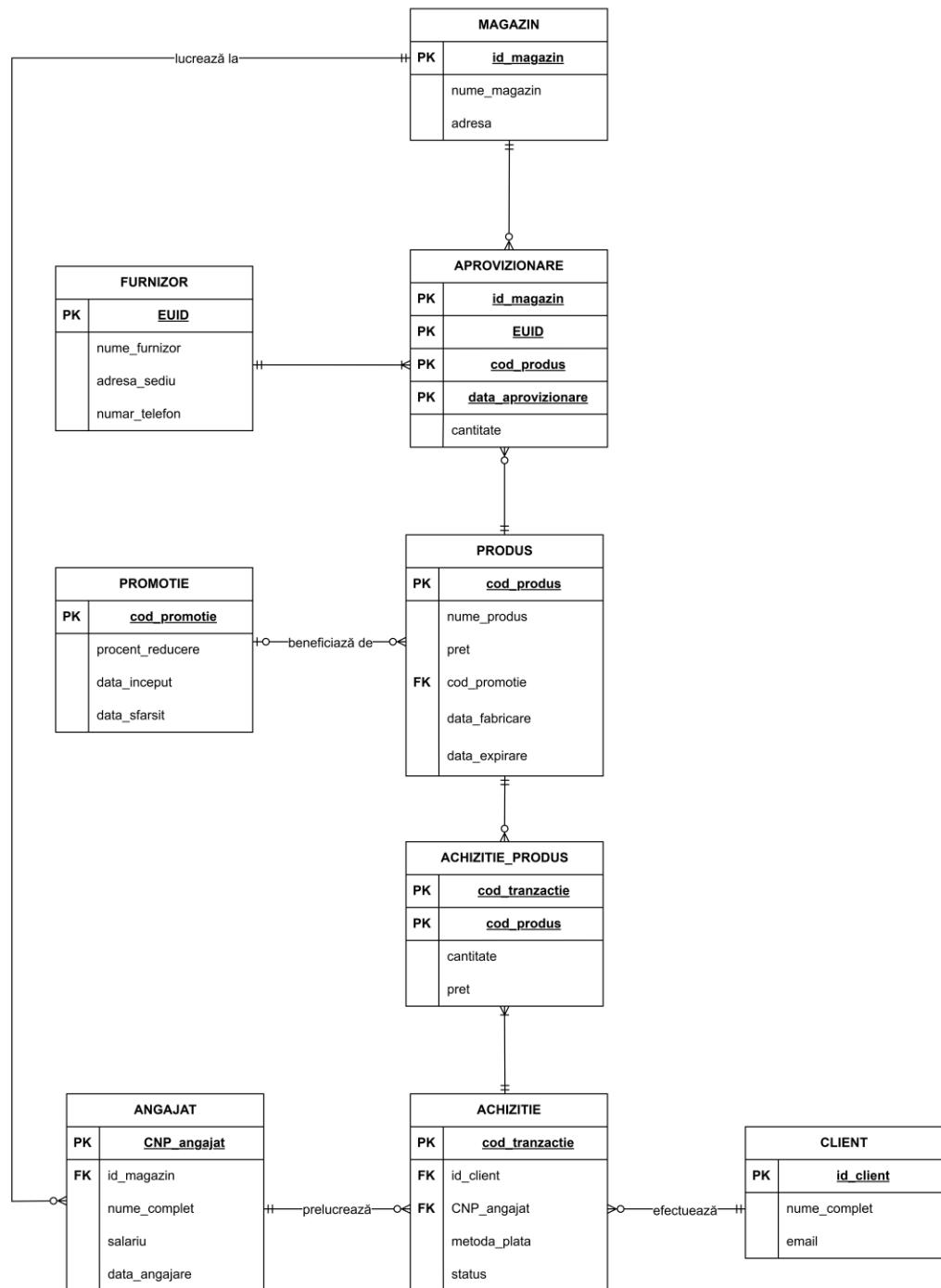
1. Descrierea modelului real, a utilității acestuia și a regulilor de funcționare

- 1) Proiectul are ca scop dezvoltarea unui sistem de gestionare pentru un lanț de supermarket-uri. Pentru fiecare magazin este gestionat sistemul de aprovizionare, produsele, promoțiile aferente lor, tranzacțiile realizate, dar și clienții și personalul.
- 2) Fiecare produs din fiecare magazin este adus de către un furnizor la o dată anume. Astfel, avem o relație ternară între MAGAZIN, FURNIZOR și PRODUS, având cheia primară compusă din cheile celor 3 tabele și data la care a fost făcută aprovizionarea. În cazul în care data nu ar face parte din cheia primară, un furnizor ar putea aduce un produs la un magazin o singură dată.
- 3) Un produs nu poate beneficia de mai multe promoții simultan, iar o promoție nu poate avea niciodată 0 produse. Pentru a respecta în totalitate această regulă impusă în model, acțiunea de creare a tabelului PROMOTIE și de adăugare a atributului cod_promotie în tabelul PRODUS trebuie realizate concurent.
- 4) O tranzacție poate corespunde unui singur client. Vom presupune că fiecare client are un card cu id-ul său pe care îl scanează de fiecare dată când plătește.
- 5) Similar, o achiziție este prelucrată de un singur angajat, dar un angajat nu este obligatoriu să prelucreze nicio achiziție.
- 6) Un angajat nu poate lucra în mai multe magazine simultan. Este obligatoriu ca un angajat să facă parte dintr-un magazin deoarece, dacă nu ar lucra în niciunul, ar însemna că nu ar mai fi angajat (trebuie eliminat din baza de date). Un magazin poate să nu aibă niciun angajat (închis temporar, în proces de restructurări etc.).

2. Diagrama ERD (entity-relationship)



3. Diagrama conceptuală



4. Implementarea în Oracle a diagramei conceptuale realizate

- Crearea tabelului **MAGAZIN**:

```
CREATE TABLE MAGAZIN (
    id_magazin NUMBER,
    nume_magazin VARCHAR2(50),
    adresa VARCHAR2(50) UNIQUE,
    PRIMARY KEY (id_magazin)
);
```

- Crearea tabelului **FURNIZOR**:

```
CREATE TABLE FURNIZOR (
    EUID VARCHAR2(10),
    nume_furnizor VARCHAR2(30),
    adresa_sediu VARCHAR2(50) UNIQUE,
    numar_telefon NUMBER(15,0) UNIQUE,
    PRIMARY KEY (EUID)
);
```

- Crearea tabelului **PROMOTIE**:

```
CREATE TABLE PROMOTIE (
    cod_promotie NUMBER,
    procent_reducere NUMBER(3,0),
    data_inceput DATE,
    data_sfarsit DATE,
    PRIMARY KEY (cod_promotie),
    CHECK (data_inceput < data_sfarsit)
);
```

- Crearea tabelului **PRODUS**:

```
CREATE TABLE PRODUS (
    cod_produs NUMBER,
    nume_produs VARCHAR2(30),
    pret NUMBER(6,2),
    cod_promotie NUMBER,
    data_fabricare DATE,
    data_expirare DATE,
    PRIMARY KEY (cod_produs),
    FOREIGN KEY (cod_promotie)
        REFERENCES PROMOTIE(cod_promotie),
    CHECK (data_fabricare < data_expirare)
);
```

- Crearea tabelului **APROVIZIONARE**:

```
CREATE TABLE APROVIZIONARE (
    id_magazin NUMBER REFERENCES MAGAZIN(id_magazin) ON DELETE CASCADE,
    cod_produs NUMBER REFERENCES PRODUS(cod_produs) ON DELETE CASCADE,
    EUID VARCHAR2(10) REFERENCES FURNIZOR(EUID) ON DELETE CASCADE,
    data_aprovizionare DATE,
    cantitate NUMBER,
    PRIMARY KEY (id_magazin, cod_produs, EUID, data_aprovizionare)
);
```

- Crearea tabelului **CLIENT**:

```
CREATE TABLE CLIENT (
    id_client NUMBER,
    nume_complet VARCHAR2(30),
    email VARCHAR2(30) CHECK (email LIKE '%@%.%') UNIQUE,
    PRIMARY KEY (id_client)
);
```

- Crearea tabeluluiui **ANGAJAT**:

```
CREATE TABLE ANGAJAT (
    CNP_angajat NUMBER(13,0),
    id_magazin NUMBER,
    nume_complet VARCHAR2(30),
    salariu NUMBER(5,0),
    data_angajare DATE,
    PRIMARY KEY (CNP_angajat),
    FOREIGN KEY (id_magazin)
        REFERENCES MAGAZIN(id_magazin) ON DELETE SET NULL
);
```

- Crearea tabeluluiui **ACHIZITIE**:

```
CREATE TABLE ACHIZITIE (
    cod_tranzactie NUMBER,
    id_client NUMBER,
    CNP_angajat NUMBER(13,0),
    metoda_plata VARCHAR2(30) CHECK (metoda_plata IN ('Card','Numerar','PayPal')),
    status VARCHAR2(30),
    PRIMARY KEY (cod_tranzactie),
    FOREIGN KEY (id_client)
        REFERENCES CLIENT(id_client) ON DELETE SET NULL,
    FOREIGN KEY (CNP_angajat)
        REFERENCES ANGAJAT(CNP_angajat) ON DELETE SET NULL
);
```

- Crearea tabeluluiui asociativ **ACHIZITIE-PRODUS**:

```
CREATE TABLE ACHIZITIE_PRODUS (
    cod_produs NUMBER REFERENCES PRODUS(cod_produs) ON DELETE CASCADE,
    cod_tranzactie NUMBER REFERENCES ACHIZITIE(cod_tranzactie) ON DELETE
CASCADE,
    cantitate NUMBER,
    pret NUMBER(6,2),
    PRIMARY KEY (cod_produs, cod_tranzactie)
);
```

5. Adăugarea de informații coerente în tabelele create

- Tabelul **MAGAZIN**:

INSERT ALL

```
INTO MAGAZIN (id_magazin, nume_magazin, adresa) VALUES (1, 'Magazin Lipsani',  
'Strada Lipsani 22')  
INTO MAGAZIN (id_magazin, nume_magazin, adresa) VALUES (2, 'Magazin Eroilor',  
'Bulevardul Eroilor 15')  
INTO MAGAZIN (id_magazin, nume_magazin, adresa) VALUES (3, 'Unirii Shopping  
Center', 'Bulevardul Unirii 10')  
INTO MAGAZIN (id_magazin, nume_magazin, adresa) VALUES (4, 'Paris Plaza', 'Avenue  
de Champs-Elysees 45')  
INTO MAGAZIN (id_magazin, nume_magazin, adresa) VALUES (5, 'Magazin Central',  
'Bulevardul St. Pellegrino 30')  
INTO MAGAZIN (id_magazin, nume_magazin, adresa) VALUES (6, 'Madrid Center',  
'Gran Vía 58')  
INTO MAGAZIN (id_magazin, nume_magazin, adresa) VALUES (7, 'City Plaza', 'Calea  
Bucuresti 23')  
INTO MAGAZIN (id_magazin, nume_magazin, adresa) VALUES (8, 'Munich Shopping  
Center', 'Maximilianstrasse 7')  
INTO MAGAZIN (id_magazin, nume_magazin, adresa) VALUES (9, 'Central Boutique',  
'Rue de la Republique 19')  
INTO MAGAZIN (id_magazin, nume_magazin, adresa) VALUES (10, 'Barcelona Central  
Plaza', 'La Rambla 42')  
SELECT * FROM dual;
```

- Tabelul **FURNIZOR**:

INSERT ALL

```
INTO FURNIZOR (EUID, nume_furnizor, adresa_sediu, numar_telefon) VALUES  
('RO73925', 'ElectroTech SRL', 'Sibiu, Drumul Fermei 45', 40721234567)  
INTO FURNIZOR (EUID, nume_furnizor, adresa_sediu, numar_telefon) VALUES  
('RO15347', 'FashionStyle SA', 'Galati, Str. Domneasca 12', 40734567890)  
INTO FURNIZOR (EUID, nume_furnizor, adresa_sediu, numar_telefon) VALUES  
('RO28964', 'AgroFood SRL', 'Cluj-Napoca, Calea Turzii 8', 40756789012)
```

```

INTO FURNIZOR (EUID, nume_furnizor, adresa_sediu, numar_telefon) VALUES
('RO42681', 'BeautyCosmetics SA', 'Timisoara, Bd. Republicii 23', 40767890123)
INTO FURNIZOR (EUID, nume_furnizor, adresa_sediu, numar_telefon) VALUES
('RO59713', 'SportLife SRL', 'Iasi, Str. Garii 56', 40778901234)
INTO FURNIZOR (EUID, nume_furnizor, adresa_sediu, numar_telefon) VALUES
('DE84296', 'BricoWarehouse GMBH', 'Hamburg, Zona Industriala 32', 49789012345)
INTO FURNIZOR (EUID, nume_furnizor, adresa_sediu, numar_telefon) VALUES
('DE37102', 'BookWorld SA', 'Munchen', 49790123456)
INTO FURNIZOR (EUID, nume_furnizor, adresa_sediu, numar_telefon) VALUES
('FR63851', 'ToysFrance SA', 'Paris', 33701234567)
INTO FURNIZOR (EUID, nume_furnizor, adresa_sediu, numar_telefon) VALUES
('FR92467', 'AutoParts SRL', 'Lyon', 33712345678)
INTO FURNIZOR (EUID, nume_furnizor, adresa_sediu, numar_telefon) VALUES
('IT18539', 'Bricolitalia GMBH', 'Roma', 39723456789)
INTO FURNIZOR (EUID, nume_furnizor, adresa_sediu, numar_telefon) VALUES
('ES24973', 'ElectroEspana SA', 'Madrid', 34745678901)
INTO FURNIZOR (EUID, nume_furnizor, adresa_sediu, numar_telefon) VALUES
('ES50681', 'ModaBarcelona SA', 'Barcelona, La Rambla 22', 34756789012)
INTO FURNIZOR (EUID, nume_furnizor, adresa_sediu, numar_telefon) VALUES
('IT74295', 'AgroItalia SRL', 'Milano', 39767890123)
INTO FURNIZOR (EUID, nume_furnizor, adresa_sediu, numar_telefon) VALUES
('DE61834', 'SportDeutsch GMBH', 'Hamburg', 49778901234)
INTO FURNIZOR (EUID, nume_furnizor, adresa_sediu, numar_telefon) VALUES
('FR45720', 'MonJardin SA', 'Marseille', 33789012345)
SELECT * FROM dual;

```

- Tabelul **PROMOTIE**:

```

INSERT ALL
INTO PROMOTIE (cod_promotie, procent_reducere, data_inceput, data_sfarsit)
VALUES (1, 10, TO_DATE('01-05-2024', 'DD-MM-YYYY'), TO_DATE('10-05-2024', 'DD-MM-YYYY'))
INTO PROMOTIE (cod_promotie, procent_reducere, data_inceput, data_sfarsit)
VALUES (2, 25, TO_DATE('15-06-2024', 'DD-MM-YYYY'), TO_DATE('20-06-2024', 'DD-MM-YYYY'))
INTO PROMOTIE (cod_promotie, procent_reducere, data_inceput, data_sfarsit)
VALUES (3, 50, TO_DATE('15-11-2024', 'DD-MM-YYYY'), TO_DATE('18-11-2024', 'DD-MM-YYYY'))

```

```

INTO PROMOTIE (cod_promotie, procent_reducere, data_inceput, data_sfarsit)
VALUES (4, 15, TO_DATE('01-09-2024', 'DD-MM-YYYY'), TO_DATE('15-09-2024', 'DD-MM-
YYYY'))
INTO PROMOTIE (cod_promotie, procent_reducere, data_inceput, data_sfarsit)
VALUES (5, 30, TO_DATE('01-07-2024', 'DD-MM-YYYY'), TO_DATE('31-07-2024', 'DD-MM-
YYYY'))
INTO PROMOTIE (cod_promotie, procent_reducere, data_inceput, data_sfarsit)
VALUES (6, 20, TO_DATE('20-12-2024', 'DD-MM-YYYY'), TO_DATE('31-12-2024', 'DD-MM-
YYYY'))
INTO PROMOTIE (cod_promotie, procent_reducere, data_inceput, data_sfarsit)
VALUES (7, 5, TO_DATE('01-01-2024', 'DD-MM-YYYY'), TO_DATE('31-12-2024', 'DD-MM-
YYYY'))
SELECT * FROM dual;

```

- Tabelul **PRODUS**:

INSERT ALL

```

INTO PRODUS (cod_produs, nume_produs, pret, cod_promotie, data_fabricare,
data_expirare) VALUES (1, 'Televizor Smart LED', 2500, 1, TO_DATE('15-03-2024', 'DD-
MM-YYYY'), NULL)
INTO PRODUS (cod_produs, nume_produs, pret, cod_promotie, data_fabricare,
data_expirare) VALUES (2, 'Laptop Gaming', 3899.99, 2, TO_DATE('10-04-2024', 'DD-MM-
YYYY'), NULL)
INTO PRODUS (cod_produs, nume_produs, pret, cod_promotie, data_fabricare,
data_expirare) VALUES (3, 'Smartphone', 1799.99, 3, TO_DATE('05-05-2024', 'DD-MM-
YYYY'), NULL)
INTO PRODUS (cod_produs, nume_produs, pret, cod_promotie, data_fabricare,
data_expirare) VALUES (4, 'Blugi', 149.99, 4, TO_DATE('20-01-2024', 'DD-MM-YYYY'),
NULL)
INTO PRODUS (cod_produs, nume_produs, pret, cod_promotie, data_fabricare,
data_expirare) VALUES (5, 'Tricou', 60, 5, TO_DATE('15-02-2024', 'DD-MM-YYYY'), NULL)
INTO PRODUS (cod_produs, nume_produs, pret, cod_promotie, data_fabricare,
data_expirare) VALUES (6, 'Geaca', 300, 6, TO_DATE('10-10-2023', 'DD-MM-YYYY'), NULL)
INTO PRODUS (cod_produs, nume_produs, pret, cod_promotie, data_fabricare,
data_expirare) VALUES (7, 'Lapte', 6.99, 7, TO_DATE('12-05-2024', 'DD-MM-YYYY'),
TO_DATE('26-05-2024', 'DD-MM-YYYY'))
INTO PRODUS (cod_produs, nume_produs, pret, cod_promotie, data_fabricare,
data_expirare) VALUES (8, 'Paine', 4.50, 1, TO_DATE('15-05-2024', 'DD-MM-YYYY'),
TO_DATE('20-05-2024', 'DD-MM-YYYY'))

```

```
INTO PRODUS (cod_produs, nume_produs, pret, cod_promotie, data_fabricare,
data_expirare) VALUES (9, 'Oua', 12.99, 2, TO_DATE('10-05-2024', 'DD-MM-YYYY'),
TO_DATE('01-06-2024', 'DD-MM-YYYY'))  
INTO PRODUS (cod_produs, nume_produs, pret, cod_promotie, data_fabricare,
data_expirare) VALUES (10, 'Crema fata', 89.99, 3, TO_DATE('05-12-2023', 'DD-MM-
YYYY'), TO_DATE('05-12-2025', 'DD-MM-YYYY'))  
INTO PRODUS (cod_produs, nume_produs, pret, cod_promotie, data_fabricare,
data_expirare) VALUES (11, 'Parfum', 245.50, 4, TO_DATE('15-11-2023', 'DD-MM-YYYY'),
TO_DATE('15-11-2026', 'DD-MM-YYYY'))  
INTO PRODUS (cod_produs, nume_produs, pret, cod_promotie, data_fabricare,
data_expirare) VALUES (12, 'Sampon', 24.99, 5, TO_DATE('25-01-2024', 'DD-MM-YYYY'),
TO_DATE('25-01-2026', 'DD-MM-YYYY'))  
INTO PRODUS (cod_produs, nume_produs, pret, cod_promotie, data_fabricare,
data_expirare) VALUES (13, 'Minge fotbal', 79.99, 6, TO_DATE('20-09-2023', 'DD-MM-
YYYY'), NULL)  
INTO PRODUS (cod_produs, nume_produs, pret, cod_promotie, data_fabricare,
data_expirare) VALUES (14, 'Racheta tenis', 179.50, 7, TO_DATE('15-08-2023', 'DD-MM-
YYYY'), NULL)  
INTO PRODUS (cod_produs, nume_produs, pret, cod_promotie, data_fabricare,
data_expirare) VALUES (15, 'Bicicleta', 900.50, 1, TO_DATE('10-07-2023', 'DD-MM-YYYY'),
NULL)  
INTO PRODUS (cod_produs, nume_produs, pret, cod_promotie, data_fabricare,
data_expirare) VALUES (16, 'Set gradinarit', 125, 3, TO_DATE('05-04-2023', 'DD-MM-
YYYY'), NULL)  
INTO PRODUS (cod_produs, nume_produs, pret, cod_promotie, data_fabricare,
data_expirare) VALUES (17, 'Scaun gradina', 160, NULL, TO_DATE('15-05-2023', 'DD-MM-
YYYY'), NULL)  
INTO PRODUS (cod_produs, nume_produs, pret, cod_promotie, data_fabricare,
data_expirare) VALUES (18, 'Masa exterior', 350.99, NULL, TO_DATE('20-05-2023', 'DD-
MM-YYYY'), NULL)  
INTO PRODUS (cod_produs, nume_produs, pret, cod_promotie, data_fabricare,
data_expirare) VALUES (19, 'Roman', 45.99, NULL, TO_DATE('10-02-2022', 'DD-MM-
YYYY'), NULL)  
INTO PRODUS (cod_produs, nume_produs, pret, cod_promotie, data_fabricare,
data_expirare) VALUES (20, 'Atlas', 79.99, NULL, TO_DATE('15-06-2021', 'DD-MM-YYYY'),
NULL)  
INTO PRODUS (cod_produs, nume_produs, pret, cod_promotie, data_fabricare,
data_expirare) VALUES (21, 'Carte de colorat', 19.99, NULL, TO_DATE('20-01-2023', 'DD-
MM-YYYY'), NULL)  
INTO PRODUS (cod_produs, nume_produs, pret, cod_promotie, data_fabricare,
data_expirare) VALUES (22, 'Papusa', 70.99, NULL, TO_DATE('10-03-2023', 'DD-MM-
YYYY'), NULL)
```

```

INTO PRODUS (cod_produs, nume_produs, pret, cod_promotie, data_fabricare,
data_expirare) VALUES (23, 'Set Lego', 249.99, NULL, TO_DATE('05-04-2023', 'DD-MM-
YYYY'), NULL)
INTO PRODUS (cod_produs, nume_produs, pret, cod_promotie, data_fabricare,
data_expirare) VALUES (24, 'Puzzle', 37, NULL, TO_DATE('15-02-2023', 'DD-MM-YYYY'),
NULL)
INTO PRODUS (cod_produs, nume_produs, pret, cod_promotie, data_fabricare,
data_expirare) VALUES (25, 'Ulei motor', 50, NULL, TO_DATE('05-01-2023', 'DD-MM-
YYYY'), TO_DATE('05-01-2026', 'DD-MM-YYYY'))
INTO PRODUS (cod_produs, nume_produs, pret, cod_promotie, data_fabricare,
data_expirare) VALUES (26, 'Antigel', 29.99, NULL, TO_DATE('10-02-2023', 'DD-MM-
YYYY'), TO_DATE('10-02-2028', 'DD-MM-YYYY'))
INTO PRODUS (cod_produs, nume_produs, pret, cod_promotie, data_fabricare,
data_expirare) VALUES (27, 'Odorizant auto', 15.99, NULL, TO_DATE('15-03-2023', 'DD-
MM-YYYY'), TO_DATE('15-03-2025', 'DD-MM-YYYY'))
INTO PRODUS (cod_produs, nume_produs, pret, cod_promotie, data_fabricare,
data_expirare) VALUES (28, 'Bormasina', 299.99, NULL, TO_DATE('10-12-2022', 'DD-MM-
YYYY'), NULL)
INTO PRODUS (cod_produs, nume_produs, pret, cod_promotie, data_fabricare,
data_expirare) VALUES (29, 'Set surubelnite', 89.99, NULL, TO_DATE('15-01-2023', 'DD-
MM-YYYY'), NULL)
INTO PRODUS (cod_produs, nume_produs, pret, cod_promotie, data_fabricare,
data_expirare) VALUES (30, 'Ciocan', 50, NULL, TO_DATE('20-02-2023', 'DD-MM-YYYY'),
NULL)
SELECT * FROM dual;

```

- Tabelul **APROVIZIONARE**:

```

INSERT ALL
INTO APROVIZIONARE (id_magazin, cod_produs, EUID, cantitate, data_aprovizionare)
VALUES (2, 17, 'RO73925', 50, TO_DATE('20-03-2024', 'DD-MM-YYYY'))
INTO APROVIZIONARE (id_magazin, cod_produs, EUID, cantitate, data_aprovizionare)
VALUES (9, 10, 'RO73925', 30, TO_DATE('15-04-2024', 'DD-MM-YYYY'))
INTO APROVIZIONARE (id_magazin, cod_produs, EUID, cantitate, data_aprovizionare)
VALUES (1, 25, 'ES24973', 40, TO_DATE('10-05-2024', 'DD-MM-YYYY'))
INTO APROVIZIONARE (id_magazin, cod_produs, EUID, cantitate, data_aprovizionare)
VALUES (1, 30, 'RO15347', 100, TO_DATE('25-01-2024', 'DD-MM-YYYY'))
INTO APROVIZIONARE (id_magazin, cod_produs, EUID, cantitate, data_aprovizionare)
VALUES (10, 14, 'RO15347', 120, TO_DATE('20-02-2024', 'DD-MM-YYYY'))

```

```

INTO APROVIZIONARE (id_magazin, cod_produs, EUID, cantitate, data_aprovizionare)
VALUES (7, 27, 'RO15347', 80, TO_DATE('15-10-2023', 'DD-MM-YYYY'))
INTO APROVIZIONARE (id_magazin, cod_produs, EUID, cantitate, data_aprovizionare)
VALUES (6, 21, 'RO28964', 200, TO_DATE('14-05-2024', 'DD-MM-YYYY'))
INTO APROVIZIONARE (id_magazin, cod_produs, EUID, cantitate, data_aprovizionare)
VALUES (8, 23, 'RO28964', 150, TO_DATE('16-05-2024', 'DD-MM-YYYY'))
INTO APROVIZIONARE (id_magazin, cod_produs, EUID, cantitate, data_aprovizionare)
VALUES (5, 10, 'RO28964', 120, TO_DATE('12-05-2024', 'DD-MM-YYYY'))
INTO APROVIZIONARE (id_magazin, cod_produs, EUID, cantitate, data_aprovizionare)
VALUES (3, 19, 'RO42681', 60, TO_DATE('10-12-2023', 'DD-MM-YYYY'))
INTO APROVIZIONARE (id_magazin, cod_produs, EUID, cantitate, data_aprovizionare)
VALUES (2, 12, 'RO42681', 45, TO_DATE('20-11-2023', 'DD-MM-YYYY'))
INTO APROVIZIONARE (id_magazin, cod_produs, EUID, cantitate, data_aprovizionare)
VALUES (3, 26, 'RO42681', 70, TO_DATE('30-01-2024', 'DD-MM-YYYY'))
INTO APROVIZIONARE (id_magazin, cod_produs, EUID, cantitate, data_aprovizionare)
VALUES (10, 26, 'RO59713', 25, TO_DATE('25-09-2023', 'DD-MM-YYYY'))
INTO APROVIZIONARE (id_magazin, cod_produs, EUID, cantitate, data_aprovizionare)
VALUES (1, 28, 'RO59713', 15, TO_DATE('20-08-2023', 'DD-MM-YYYY'))
INTO APROVIZIONARE (id_magazin, cod_produs, EUID, cantitate, data_aprovizionare)
VALUES (2, 24, 'RO59713', 10, TO_DATE('15-07-2023', 'DD-MM-YYYY'))
INTO APROVIZIONARE (id_magazin, cod_produs, EUID, cantitate, data_aprovizionare)
VALUES (6, 16, 'DE84296', 30, TO_DATE('10-04-2023', 'DD-MM-YYYY'))
INTO APROVIZIONARE (id_magazin, cod_produs, EUID, cantitate, data_aprovizionare)
VALUES (9, 20, 'DE84296', 25, TO_DATE('20-05-2023', 'DD-MM-YYYY'))
INTO APROVIZIONARE (id_magazin, cod_produs, EUID, cantitate, data_aprovizionare)
VALUES (3, 23, 'DE84296', 20, TO_DATE('25-05-2023', 'DD-MM-YYYY'))
INTO APROVIZIONARE (id_magazin, cod_produs, EUID, cantitate, data_aprovizionare)
VALUES (10, 11, 'DE37102', 40, TO_DATE('15-02-2022', 'DD-MM-YYYY'))
INTO APROVIZIONARE (id_magazin, cod_produs, EUID, cantitate, data_aprovizionare)
VALUES (7, 24, 'DE37102', 35, TO_DATE('20-06-2021', 'DD-MM-YYYY'))
SELECT * FROM dual;

```

- Tabelul **CLIENT**:

INSERT ALL

```

INTO CLIENT (id_client, nume_complet, email) VALUES (1, 'Popescu Ion',
'ion.popescu@email.com')
INTO CLIENT (id_client, nume_complet, email) VALUES (2, 'Ionescu Maria',
'maria.ionescu@email.com')

```

```
INTO CLIENT (id_client, nume_complet, email) VALUES (3, 'Nastase Ilie',
'inastase@gmail.com')
INTO CLIENT (id_client, nume_complet, email) VALUES (4, 'Popa Elena',
'elena.popa@gmail.com')
INTO CLIENT (id_client, nume_complet, email) VALUES (5, 'Stanescu Mihai',
'mihai.s@gmail.com')
INTO CLIENT (id_client, nume_complet, email) VALUES (6, 'Hans Flick',
'hans.flick@gmail.com')
INTO CLIENT (id_client, nume_complet, email) VALUES (7, 'Ana Maria',
'anna.maria@email.com')
INTO CLIENT (id_client, nume_complet, email) VALUES (8, 'Thomas Tuchel',
'thomas.tuchel@email.com')
INTO CLIENT (id_client, nume_complet, email) VALUES (9, 'Becker Allison',
'allison.becker@email.com')
INTO CLIENT (id_client, nume_complet, email) VALUES (10, 'Rashford Markus',
'markus.r@email.com')
INTO CLIENT (id_client, nume_complet, email) VALUES (11, 'Jean Polnareff',
'jean.pol@email.com')
INTO CLIENT (id_client, nume_complet, email) VALUES (12, 'Martin Sophie',
'sophie.martin@email.com')
INTO CLIENT (id_client, nume_complet, email) VALUES (13, 'Leroy Sane',
'sane.leroy@email.com')
INTO CLIENT (id_client, nume_complet, email) VALUES (14, 'Bernard Emilio',
'emi.bernard@yahoo.com')
INTO CLIENT (id_client, nume_complet, email) VALUES (15, 'Petit Michael',
'mike.petit@hotmail.com')
INTO CLIENT (id_client, nume_complet, email) VALUES (16, 'Rossi Francesco',
'fran.rossi@hotmail.com')
INTO CLIENT (id_client, nume_complet, email) VALUES (17, 'Ferrari Enzo',
'enzo.ferrari@email.com')
INTO CLIENT (id_client, nume_complet, email) VALUES (18, 'Giancarlo Esposito',
'gesposito@yahoo.com')
INTO CLIENT (id_client, nume_complet, email) VALUES (19, 'Andra Sofia',
'sofiaandraa@yahoo.com')
INTO CLIENT (id_client, nume_complet, email) VALUES (20, 'Enache Luca',
'lucaenache@email.com')
INTO CLIENT (id_client, nume_complet, email) VALUES (21, 'Silva Miguel',
'miguel.silva@email.com')
INTO CLIENT (id_client, nume_complet, email) VALUES (22, 'Rodriguez Ana',
'ana.rodriguez@email.com')
INTO CLIENT (id_client, nume_complet, email) VALUES (23, 'Fernando Torres',
'fertorres@email.com')
INTO CLIENT (id_client, nume_complet, email) VALUES (24, 'Lopez Jennifer',
'jlopez@email.com')
```

```
INTO CLIENT (id_client, nume_complet, email) VALUES (25, 'Martinez Emiliano',
'emimartinez@email.com')
SELECT * FROM dual;
```

- Tabelul **ANGAJAT**:

INSERT ALL

```
INTO ANGAJAT (CNP_angajat, id_magazin, nume_complet, salariu, data_angajare)
VALUES (1890302234567, 5, 'Ionescu Elena', 3800, TO_DATE('10-02-2021', 'DD-MM-
YYYY'))
INTO ANGAJAT (CNP_angajat, id_magazin, nume_complet, salariu, data_angajare)
VALUES (1750215789012, 3, 'Roby Will', 6000, TO_DATE('20-05-2025', 'DD-MM-YYYY'))
INTO ANGAJAT (CNP_angajat, id_magazin, nume_complet, salariu, data_angajare)
VALUES (1880723345678, 8, 'Popa Andreea', 3900, TO_DATE('05-07-2021', 'DD-MM-
YYYY'))
INTO ANGAJAT (CNP_angajat, id_magazin, nume_complet, salariu, data_angajare)
VALUES (1760812456789, 7, 'Stanescu Cristian', 5800, TO_DATE('12-11-2019', 'DD-MM-
YYYY'))
INTO ANGAJAT (CNP_angajat, id_magazin, nume_complet, salariu, data_angajare)
VALUES (1870405567890, 2, 'Mario Klaus', 4200, TO_DATE('30-09-2020', 'DD-MM-YYYY'))
INTO ANGAJAT (CNP_angajat, id_magazin, nume_complet, salariu, data_angajare)
VALUES (1740628901234, 4, 'George Kross', 5700, TO_DATE('15-08-2018', 'DD-MM-
YYYY'))
INTO ANGAJAT (CNP_angajat, id_magazin, nume_complet, salariu, data_angajare)
VALUES (1860917678901, 6, 'Marco Raul', 4100, TO_DATE('22-04-2021', 'DD-MM-YYYY'))
INTO ANGAJAT (CNP_angajat, id_magazin, nume_complet, salariu, data_angajare)
VALUES (1770330012345, 10, 'Martin Leonard', 5900, TO_DATE('03-12-2018', 'DD-MM-
YYYY'))
INTO ANGAJAT (CNP_angajat, id_magazin, nume_complet, salariu, data_angajare)
VALUES (1850528789012, 1, 'Rocco Martin', 4300, TO_DATE('18-06-2020', 'DD-MM-
YYYY'))
INTO ANGAJAT (CNP_angajat, id_magazin, nume_complet, salariu, data_angajare)
VALUES (1780204123456, 9, 'Mitch Rodrygo', 5800, TO_DATE('25-01-2019', 'DD-MM-
YYYY'))
INTO ANGAJAT (CNP_angajat, id_magazin, nume_complet, salariu, data_angajare)
VALUES (1840809890123, 3, 'Garcia Sanchez', 4000, TO_DATE('12-05-2021', 'DD-MM-
YYYY'))
INTO ANGAJAT (CNP_angajat, id_magazin, nume_complet, salariu, data_angajare)
VALUES (1790519234567, 5, 'Roberto Carlos', 5600, TO_DATE('08-07-2019', 'DD-MM-
YYYY'))
```

```

INTO ANGAJAT (CNP_angajat, id_magazin, nume_complet, salariu, data_angajare)
VALUES (1830210901234, 7, 'Kirk Hammett', 4200, TO_DATE('30-11-2020', 'DD-MM-
YYYY'))
INTO ANGAJAT (CNP_angajat, id_magazin, nume_complet, salariu, data_angajare)
VALUES (1800726345678, 8, 'Szabo Laszlo', 5500, TO_DATE('17-03-2019', 'DD-MM-
YYYY'))
INTO ANGAJAT (CNP_angajat, id_magazin, nume_complet, salariu, data_angajare)
VALUES (1820601012345, 2, 'Lars Ulrich', 4100, TO_DATE('20-01-2021', 'DD-MM-YYYY'))
INTO ANGAJAT (CNP_angajat, id_magazin, nume_complet, salariu, data_angajare)
VALUES (1810915456789, 10, 'James Hetfield', 5700, TO_DATE('05-04-2019', 'DD-MM-
YYYY'))
INTO ANGAJAT (CNP_angajat, id_magazin, nume_complet, salariu, data_angajare)
VALUES (1891112123456, 6, 'Axel Rose', 4000, TO_DATE('15-08-2021', 'DD-MM-YYYY'))
INTO ANGAJAT (CNP_angajat, id_magazin, nume_complet, salariu, data_angajare)
VALUES (1830617567890, 4, 'Mike Pinkman', 5600, TO_DATE('22-09-2019', 'DD-MM-
YYYY'))
INTO ANGAJAT (CNP_angajat, id_magazin, nume_complet, salariu, data_angajare)
VALUES (1881023234567, 1, 'Smith Will', 4300, TO_DATE('12-10-2020', 'DD-MM-YYYY'))
INTO ANGAJAT (CNP_angajat, id_magazin, nume_complet, salariu, data_angajare)
VALUES (1841129678901, 3, 'Walter Blue', 5900, TO_DATE('28-06-2018', 'DD-MM-YYYY'))
SELECT * FROM dual;

```

- Tabelul **ACHIZITIE**:

```

INSERT ALL
INTO ACHIZITIE (cod_tranzactie, id_client, CNP_angajat, metoda_plata, status)
VALUES (1001, 1, 1810915456789, 'Card', 'Finalizata')
INTO ACHIZITIE (cod_tranzactie, id_client, CNP_angajat, metoda_plata, status)
VALUES (1002, 2, 1890302234567, 'Numerar', 'Finalizata')
INTO ACHIZITIE (cod_tranzactie, id_client, CNP_angajat, metoda_plata, status)
VALUES (1003, 3, 1750215789012, 'Card', 'Finalizata')
INTO ACHIZITIE (cod_tranzactie, id_client, CNP_angajat, metoda_plata, status)
VALUES (1004, 4, 1880723345678, 'Numerar', 'In procesare')
INTO ACHIZITIE (cod_tranzactie, id_client, CNP_angajat, metoda_plata, status)
VALUES (1005, 5, 1760812456789, 'Card', 'Finalizata')
INTO ACHIZITIE (cod_tranzactie, id_client, CNP_angajat, metoda_plata, status)
VALUES (1006, 6, 1870405567890, 'PayPal', 'Finalizata')
INTO ACHIZITIE (cod_tranzactie, id_client, CNP_angajat, metoda_plata, status)
VALUES (1007, 7, 1740628901234, 'Card', 'In procesare')

```

```
INTO ACHIZITIE (cod_tranzactie, id_client, CNP_angajat, metoda_plata, status)
VALUES (1008, 8, 1860917678901, 'Numerar', 'Finalizata')
    INTO ACHIZITIE (cod_tranzactie, id_client, CNP_angajat, metoda_plata, status)
VALUES (1009, 9, 1770330012345, 'Card', 'Finalizata')
    INTO ACHIZITIE (cod_tranzactie, id_client, CNP_angajat, metoda_plata, status)
VALUES (1010, 10, 1850528789012, 'PayPal', 'In procesare')
    INTO ACHIZITIE (cod_tranzactie, id_client, CNP_angajat, metoda_plata, status)
VALUES (1011, 11, 1780204123456, 'Card', 'Finalizata')
    INTO ACHIZITIE (cod_tranzactie, id_client, CNP_angajat, metoda_plata, status)
VALUES (1012, 12, 1840809890123, 'Numerar', 'Finalizata')
    INTO ACHIZITIE (cod_tranzactie, id_client, CNP_angajat, metoda_plata, status)
VALUES (1013, 13, 1790519234567, 'Card', 'In procesare')
    INTO ACHIZITIE (cod_tranzactie, id_client, CNP_angajat, metoda_plata, status)
VALUES (1014, 14, 1830210901234, 'Card', 'Finalizata')
    INTO ACHIZITIE (cod_tranzactie, id_client, CNP_angajat, metoda_plata, status)
VALUES (1015, 15, 1800726345678, 'PayPal', 'Finalizata')
    INTO ACHIZITIE (cod_tranzactie, id_client, CNP_angajat, metoda_plata, status)
VALUES (1016, 16, 1820601012345, 'Card', 'In procesare')
    INTO ACHIZITIE (cod_tranzactie, id_client, CNP_angajat, metoda_plata, status)
VALUES (1017, 17, 1810915456789, 'Numerar', 'Finalizata')
    INTO ACHIZITIE (cod_tranzactie, id_client, CNP_angajat, metoda_plata, status)
VALUES (1018, 18, 1891112123456, 'Card', 'Finalizata')
    INTO ACHIZITIE (cod_tranzactie, id_client, CNP_angajat, metoda_plata, status)
VALUES (1019, 19, 1830617567890, 'PayPal', 'In procesare')
    INTO ACHIZITIE (cod_tranzactie, id_client, CNP_angajat, metoda_plata, status)
VALUES (1020, 20, 1881023234567, 'Card', 'Finalizata')
    INTO ACHIZITIE (cod_tranzactie, id_client, CNP_angajat, metoda_plata, status)
VALUES (1021, 21, 1841129678901, 'Numerar', 'Finalizata')
    INTO ACHIZITIE (cod_tranzactie, id_client, CNP_angajat, metoda_plata, status)
VALUES (1022, 22, 1810915456789, 'Card', 'In procesare')
    INTO ACHIZITIE (cod_tranzactie, id_client, CNP_angajat, metoda_plata, status)
VALUES (1023, 23, 1890302234567, 'PayPal', 'Finalizata')
    INTO ACHIZITIE (cod_tranzactie, id_client, CNP_angajat, metoda_plata, status)
VALUES (1024, 24, 1750215789012, 'Card', 'In procesare')
    INTO ACHIZITIE (cod_tranzactie, id_client, CNP_angajat, metoda_plata, status)
VALUES (1025, 25, 1810915456789, 'PayPal', 'In procesare')
SELECT * FROM dual;
```

- Tabelul **ACHIZITIE_PRODUS**:

INSERT ALL

```
INTO ACHIZITIE_PRODUS (cod_produs, cod_tranzactie, cantitate, pret) VALUES (14, 1001, 1, 179.50)
INTO ACHIZITIE_PRODUS (cod_produs, cod_tranzactie, cantitate, pret) VALUES (13, 1002, 2, 79.99)
INTO ACHIZITIE_PRODUS (cod_produs, cod_tranzactie, cantitate, pret) VALUES (26, 1003, 1, 29.99)
INTO ACHIZITIE_PRODUS (cod_produs, cod_tranzactie, cantitate, pret) VALUES (23, 1004, 1, 249.99)
INTO ACHIZITIE_PRODUS (cod_produs, cod_tranzactie, cantitate, pret) VALUES (17, 1005, 4, 160)
INTO ACHIZITIE_PRODUS (cod_produs, cod_tranzactie, cantitate, pret) VALUES (24, 1006, 1, 37)
INTO ACHIZITIE_PRODUS (cod_produs, cod_tranzactie, cantitate, pret) VALUES (11, 1007, 1, 245.50)
INTO ACHIZITIE_PRODUS (cod_produs, cod_tranzactie, cantitate, pret) VALUES (19, 1008, 2, 45.99)
INTO ACHIZITIE_PRODUS (cod_produs, cod_tranzactie, cantitate, pret) VALUES (16, 1009, 1, 125)
INTO ACHIZITIE_PRODUS (cod_produs, cod_tranzactie, cantitate, pret) VALUES (20, 1010, 1, 79.99)
INTO ACHIZITIE_PRODUS (cod_produs, cod_tranzactie, cantitate, pret) VALUES (15, 1011, 1, 900.50)
INTO ACHIZITIE_PRODUS (cod_produs, cod_tranzactie, cantitate, pret) VALUES (28, 1012, 1, 299.99)
INTO ACHIZITIE_PRODUS (cod_produs, cod_tranzactie, cantitate, pret) VALUES (10, 1013, 3, 89.99)
INTO ACHIZITIE_PRODUS (cod_produs, cod_tranzactie, cantitate, pret) VALUES (27, 1014, 2, 15.99)
INTO ACHIZITIE_PRODUS (cod_produs, cod_tranzactie, cantitate, pret) VALUES (21, 1015, 5, 19.99)
INTO ACHIZITIE_PRODUS (cod_produs, cod_tranzactie, cantitate, pret) VALUES (12, 1016, 1, 24.99)
INTO ACHIZITIE_PRODUS (cod_produs, cod_tranzactie, cantitate, pret) VALUES (10, 1017, 2, 89.99)
INTO ACHIZITIE_PRODUS (cod_produs, cod_tranzactie, cantitate, pret) VALUES (18, 1018, 1, 350.99)
INTO ACHIZITIE_PRODUS (cod_produs, cod_tranzactie, cantitate, pret) VALUES (25, 1019, 1, 50)
```

```
INTO ACHIZITIE_PRODUS (cod_produs, cod_tranzactie, cantitate, pret) VALUES (20,  
1020, 1, 79.99)  
INTO ACHIZITIE_PRODUS (cod_produs, cod_tranzactie, cantitate, pret) VALUES (15,  
1021, 1, 900.50)  
INTO ACHIZITIE_PRODUS (cod_produs, cod_tranzactie, cantitate, pret) VALUES (22,  
1022, 1, 70.99)  
INTO ACHIZITIE_PRODUS (cod_produs, cod_tranzactie, cantitate, pret) VALUES (13,  
1023, 2, 79.99)  
INTO ACHIZITIE_PRODUS (cod_produs, cod_tranzactie, cantitate, pret) VALUES (29,  
1024, 1, 89.99)  
INTO ACHIZITIE_PRODUS (cod_produs, cod_tranzactie, cantitate, pret) VALUES (17,  
1025, 2, 160)  
SELECT * FROM dual;
```

6. Formularea în limbaj natural a unei probleme care se rezolvă folosind un subprogram stocat independent care utilizează toate cele 3 tipuri de colecții studiate

Să se creeze o colecție care reține numele fiecărui produs care beneficiază de o promoție de cel puțin 50%, împreună cu noul preț și lista cu furnizori de la care provine. Să se adauge în această colecție și cele mai puțin vândute 5 produse și să se ieftinească cu 75%. Să se afișeze conținutul acestei colecții.

```
CREATE OR REPLACE PROCEDURE ieftinire_produse IS
    TYPE lista_furnizori_type IS TABLE OF VARCHAR2(30);

    TYPE rec_tabel_indexat IS RECORD (
        nume_produs PRODUS.nume_produs%TYPE,
        pret_redus NUMBER,
        lista_furnizori lista_furnizori_type);

    TYPE tabel_indexat_type IS TABLE OF rec_tabel_indexat INDEX BY PLS_INTEGER;

    TYPE vector_type IS VARRAY(5) OF PRODUS%ROWTYPE;

    t_produse_nevandute vector_type := vector_type();
    t_info_produse tabel_indexat_type;

    v_index PLS_INTEGER;

BEGIN
    SELECT * BULK COLLECT INTO t_produse_nevandute
    FROM (SELECT *
            FROM PRODUS p
            ORDER BY (SELECT COUNT(*)
                      FROM ACHIZITIE_PRODUS ap
                      WHERE p.cod_produs = ap.cod_produs) ASC
        )
    WHERE ROWNUM <= 5;

    FOR i IN (SELECT P.cod_produs, P.nume_produs, P.pret, PR.procent_reducere
              FROM PRODUS P
```

```

JOIN PROMOTIE PR ON P.cod_promotie = PR.cod_promotie
WHERE PR.procent_reducere >= 50) LOOP

t_info_produse(i.cod_produs).nume_produs := i.nume_produs;
t_info_produse(i.cod_produs).pret_redus := ROUND(i.pret * (1 - i.procent_reducere /
100), 2);

t_info_produse(i.cod_produs).lista_furnizori := lista_furnizori_type();

SELECT F.nume_furnizor BULK COLLECT
INTO t_info_produse(i.cod_produs).lista_furnizori
FROM FURNIZOR F
JOIN APROVIZIONARE AP ON AP.EUID = F.EUID
WHERE AP.cod_produs = i.cod_produs;

END LOOP;

FOR i IN 1..t_produse_nevandute.COUNT LOOP

    t_info_produse(t_produse_nevandute(i).cod_produs).nume_produs :=
    t_produse_nevandute(i).nume_produs;
    t_info_produse(t_produse_nevandute(i).cod_produs).pret_redus :=
    ROUND(t_produse_nevandute(i).pret * 0.25, 2);

    t_info_produse(t_produse_nevandute(i).cod_produs).lista_furnizori :=
    lista_furnizori_type();

    SELECT F.nume_furnizor BULK COLLECT
    INTO t_info_produse(t_produse_nevandute(i).cod_produs).lista_furnizori
    FROM FURNIZOR F
    JOIN APROVIZIONARE AP ON AP.EUID = F.EUID
    WHERE AP.cod_produs = t_produse_nevandute(i).cod_produs;

END LOOP;

v_index := t_info_produse.FIRST;

WHILE v_index IS NOT NULL LOOP

    DBMS_OUTPUT.PUT_LINE('Produs: ' || t_info_produse(v_index).nume_produs);
    DBMS_OUTPUT.PUT_LINE(' *preț cu reducere: ' ||
    t_info_produse(v_index).pret_redus);
    DBMS_OUTPUT.PUT_LINE(' *lista furnizorilor: ');


```

```

IF t_info_produse(v_index).lista_furnizori.COUNT > 0 THEN
    FOR i IN 1..t_info_produse(v_index).lista_furnizori.COUNT LOOP
        DBMS_OUTPUT.PUT_LINE(' -> '|| t_info_produse(v_index).lista_furnizori(i));
    END LOOP;
ELSE
    DBMS_OUTPUT.PUT_LINE(' Nu s-au înregistrat furnizori');
END IF;

DBMS_OUTPUT.PUT_LINE(' ');

v_index := t_info_produse.NEXT(v_index);
END LOOP;

END;
/

```

- Am utilizat o colecție de tip varray pentru a reține cele mai puțin vândute 5 produse deoarece știm că dimensiunea acestuia nu se modifică niciodată, nu sunt șterse elemente din colecție deci aceasta este mereu contiguă, iar numărul de elemente nu este unul mare.
- Am utilizat o colecție de tip tabel imbricat pentru a reține lista furnizorilor pentru fiecare produs în parte deoarece nu cunoaștem numărul de elemente din colecție, dar și deoarece nu ne interesează indecșii elementelor (dacă un furnizor a fost adăugat, acesta va rămâne acolo definitiv).
- Am utilizat o colecție de tip tabel indexat pentru a reține toate produsele și datele despre acestea. În această colecție cheile perechilor sunt reprezentate de codurile produselor, iar valorile sunt RECORD-uri ce conțin datele cerute despre fiecare produs. Am ales acest tip de colecție deoarece am folosit ca index pentru fiecare element codul produsului. De asemenea, conținând produsele care beneficiază de promoție, elementele din colecție sunt adăugate și eliminate des.

Rezultatul rulării acestei proceduri în SQL:

```
Procedure IEFTINIRE_PRODUSE compiled
```

```
Produs: Televizor Smart LED
*preț cu reducere: 625
*lista furnizorilor:
Nu s-au înregistrat furnizori
```

```
Produs: Laptop Gaming
*preț cu reducere: 975
*lista furnizorilor:
Nu s-au înregistrat furnizori
```

```
Produs: Smartphone
*preț cu reducere: 450
*lista furnizorilor:
Nu s-au înregistrat furnizori
```

```
Produs: Blugi
*preț cu reducere: 37.5
*lista furnizorilor:
Nu s-au înregistrat furnizori
```

```
Produs: Tricou
*preț cu reducere: 15
*lista furnizorilor:
Nu s-au înregistrat furnizori
```

```
Produs: Crema fata
*preț cu reducere: 45
*lista furnizorilor:
-> AgroFood SRL
-> ElectroTech SRL
```

```
Produs: Set gradinarit
*preț cu reducere: 62.5
*lista furnizorilor:
-> BricoWarehouse GMBH
```

```
PL/SQL procedure successfully completed.
```

7. Formularea în limbaj natural a unei probleme care se rezolvă folosind un subprogram stocat independent care utilizează 2 tipuri diferite de cursoare studiate, unul dintre acestea fiind cursor parametrizat, dependent de celălalt cursor

Să se afișeze pentru fiecare achiziție realizată la un magazin cu exact 3 angajați lista de produse din comandă.

```
CREATE OR REPLACE PROCEDURE lista_produse IS
  CURSOR c_achizitii(p_id_magazin NUMBER) IS
    SELECT AC.cod_tranzactie
      FROM ACHIZITIE AC
     JOIN ANGAJAT AN ON AN.CNP_angajat = AC.CNP_angajat
    WHERE AN.id_magazin = p_id_magazin;

  v_cod_tranzactie NUMBER;
BEGIN
  FOR i IN (SELECT id_magazin
             FROM MAGAZIN M
            WHERE (SELECT COUNT(DISTINCT(CNP_angajat))
                   FROM ANGAJAT A
                  WHERE M.id_magazin = A.id_magazin) = 3) LOOP
    OPEN c_achizitii(i.id_magazin);

    LOOP
      FETCH c_achizitii INTO v_cod_tranzactie;
      EXIT WHEN c_achizitii%NOTFOUND;

      DBMS_OUTPUT.PUT_LINE('Cod achiziție: ' || v_cod_tranzactie);
      DBMS_OUTPUT.PUT_LINE('*Lista de produse:');

      FOR j IN (SELECT P.nume_produs
                 FROM PRODUS P
                JOIN ACHIZITIE_PRODUS AP ON AP.cod_produs = P.cod_produs
               WHERE AP.cod_tranzactie = v_cod_tranzactie) LOOP
        DBMS_OUTPUT.PUT_LINE(P.nume_produs);
      END LOOP;
    END LOOP;
  END LOOP;
END;
```

```

DBMS_OUTPUT.PUT_LINE(' -> ' || j.numa_produs);
END LOOP;

DBMS_OUTPUT.PUT_LINE('');
END LOOP;

CLOSE c_achizitii;
END LOOP;
END;
/

```

- Am utilizat un ciclu-cursor implicit cu subcerere pentru a parcurge magazinele cu exact 3 angajați.
- Am utilizat un cursor explicit parametrizat pentru a parcurge angajații magazinelor găsite, parametrul fiind id_ul magazinului returnat de cursorul anterior.
- Am utilizat un ciclu-cursor implicit cu subcerere pentru a parcurge și afișa lista de produse din fiecare achiziție.

Rezultatul rulării acestei proceduri în SQL:

```

Procedure LISTA_PRODUSE compiled

Cod achiziție: 1024
*Lista de produse:
-> Set surubelnite

Cod achiziție: 1003
*Lista de produse:
-> Antigel

Cod achiziție: 1012
*Lista de produse:
-> Bormasina

Cod achiziție: 1021
*Lista de produse:
-> Bicicleta

PL/SQL procedure successfully completed.

```

8. Formularea în limbaj natural a unei probleme care se rezolvă folosind un subprogram stocat independent de tip funcție care utilizează într-o singură comandă SQL 3 dintre tabelele create

Să se afișeze cel mai bine plătit angajat cu salariul mai mare sau egal decât o sumă dată, care a procesat cel puțin o comandă în valoare de minim 500 de lei.

```
CREATE OR REPLACE FUNCTION angajat_salariu_max(p_suma ANGAJAT.salariu%TYPE)
RETURN VARCHAR2 IS

v_nume_angajat ANGAJAT.nume_complet%TYPE;

BEGIN
    SELECT A.nume_complet INTO v_nume_angajat
    FROM ANGAJAT A
    WHERE A.salariu >= p_suma
    AND EXISTS (SELECT *
                FROM ACHIZITIE AC
                JOIN ACHIZITIE_PRODUS AP ON AP.cod_tranzactie = AC.cod_tranzactie
                WHERE AC.CNP_angajat = A.CNP_angajat
                GROUP BY AC.cod_tranzactie
                HAVING SUM(AP.cantitate * AP.pret) >= 500
            )
    AND A.salariu = (SELECT MAX(A2.salariu)
                      FROM ANGAJAT A2
                      WHERE A2.salariu > p_suma
                      AND EXISTS (SELECT *
                                  FROM ACHIZITIE AC2
                                  JOIN ACHIZITIE_PRODUS AP2 ON
                                      AP2.cod_tranzactie = AC2.cod_tranzactie
                                  WHERE AC2.CNP_angajat = A2.CNP_angajat
                                  GROUP BY AC2.cod_tranzactie
                                  HAVING SUM(AP2.cantitate * AP2.pret) >= 500));

    RETURN v_nume_angajat;

EXCEPTION
    WHEN NO_DATA_FOUND THEN
```

```
    RAISE_APPLICATION_ERROR(-20000, 'Nu exista angajati care indeplindesc  
conditiile');  
WHEN TOO_MANY_ROWS THEN  
    RAISE_APPLICATION_ERROR(-20001, 'Mai multi angajati care respecta cerintele cu  
acelasi salariu maxim');  
WHEN OTHERS THEN  
    RAISE_APPLICATION_ERROR(-20002, 'Alta eroare:' || SQLERRM);  
  
END angajat_salariu_max;  
/
```

- Pentru primul caz vom da ca parametru pentru funcție valoarea 5000:

```
BEGIN  
    DBMS_OUTPUT.PUT_LINE(angajat_salariu_max(5000));  
END;  
/
```

La rularea acestui bloc vom primi următorul rezultat:

```
Function ANGAJAT_SALARIU_MAX compiled  
  
Walter Blue  
  
PL/SQL procedure successfully completed.
```

Pentru a verifica dacă rezultatul este corect, vom rula interogarea SQL:

```
SELECT A.nume_complet, A.salariu
FROM ANGAJAT A
WHERE A.salariu >= 5000
AND EXISTS (
    SELECT *
    FROM ACHIZITIE AC
    JOIN ACHIZITIE_PRODUS AP ON AP.cod_tranzactie = AC.cod_tranzactie
    WHERE AC.CNP_angajat = A.CNP_angajat
    GROUP BY AC.cod_tranzactie
    HAVING SUM(AP.cantitate * AP.pret) >= 500
)
ORDER BY A.salariu DESC;
```

Rezultatul acestei cereri este următorul:

	NUME_COMPLET	SALARIU
1	Walter Blue	5900
2	Stanescu Cristian	5800
3	Mitch Rodrygo	5800

Astfel, observăm că angajatul Walter Blue, care respectă toate condițiile, are într-adevăr cel mai mare salariu. De asemenea, este singurul cu acest salariu maxim, deci rezultatul funcției este corect.

- Pentru al doilea caz, vom modifica salariul angajatului Walter Blue la 5800, fiind astfel la egalitate cu ceilalți doi angajați:

```
UPDATE "C##ANDREISGBD"."ANGAJAT" SET SALARIU = '5800' WHERE ROWID = 'AAATXRAAHAAAIPAAT' AND ORA_ROWSCN = '109405305'
Commit Successful
```

Acum vom rula același bloc PL/SQL ca în cazul anterior:

```
BEGIN
    DBMS_OUTPUT.PUT_LINE(angajat_salariu_max(5000));
END;
/
```

La rularea acestui bloc vom primi următorul rezultat:

```
BEGIN
*
ERROR at line 1:
ORA-20001: Mai multi angajati care respecta cerintele cu acelasi salariu maxim
ORA-06512: at "C##ANDREISGBD.ANGAJAT_SALARIU_MAX", line 34
ORA-06512: at line 2

https://docs.oracle.com/error-help/db/ora-20001/

More Details :
https://docs.oracle.com/error-help/db/ora-20001/
https://docs.oracle.com/error-help/db/ora-06512/
```

Pentru a verifica dacă rezultatul este corect, vom rula interogarea SQL de la cazul anterior, având rezultatul următor:

	NUME_COMPLET	SALARIU
1	Stanescu Cristian	5800
2	Walter Blue	5800
3	Mitch Rodrygo	5800

Observăm că cei trei angajați eligibili au același salariu maxim, deci funcția returnează corect eroarea TOO_MANY_ROWS;

- Pentru al treilea caz vom da ca parametru pentru funcție valoarea 7000:

```
BEGIN
    DBMS_OUTPUT.PUT_LINE(angajat_salariu_max(7000));
END;
/
```

La rularea acestui bloc vom primi următorul rezultat:

```
BEGIN
*
ERROR at line 1:
ORA-20000: Nu exista angajati care indeplindesc conditiile
ORA-06512: at "C##ANDREISGBD.ANGAJAT_SALARIU_MAX", line 32
ORA-06512: at line 2
```

<https://docs.oracle.com/error-help/db/ora-20000/>

More Details :

<https://docs.oracle.com/error-help/db/ora-20000/>

<https://docs.oracle.com/error-help/db/ora-06512/>

Pentru a verifica dacă rezultatul este corect, vom rula interogarea SQL:

```
SELECT A.nume_complet, A.salariu
FROM ANGAJAT A
WHERE A.salariu >= 7000
AND EXISTS (
    SELECT *
    FROM ACHIZITIE AC
    JOIN ACHIZITIE_PRODUS AP ON AP.cod_tranzactie = AC.cod_tranzactie
    WHERE AC.CNP_angajat = A.CNP_angajat
    GROUP BY AC.cod_tranzactie
    HAVING SUM(AP.cantitate * AP.pret) >= 500
)
ORDER BY A.salariu DESC;
```

Rezultatul acestei cereri este următorul:

The screenshot shows a 'Query Result' window from MySQL Workbench. The window title is 'Query Result'. Below the title bar are several icons: a green play button, a red stop button, a blue refresh button, a blue document icon, and a red X close button. To the right of these icons is the text 'SQL | All Rows Fetched: 0 in 0.013 seconds'. Below the toolbar is a header row with two columns: 'NUME_CO...' and 'SALARIU'. The main body of the table is empty, indicating no results were found.

NUME_CO...	SALARIU

Astfel, observăm că nu există niciun angajat cu salariul peste suma dată ca parametru, deci funcția returnează corect eroarea NO_DATA_FOUND.

9. Formularea în limbaj natural a unei probleme care se rezolvă folosind un subprogram stocat independent de tip procedură care primește minim 2 parametri și utilizează într-o singură comandă SQL 5 dintre tabelele create și definirea a minim 2 exceptii proprii

Să se afișeze suma totală a produselor achiziționate de un client cu un id client primit ca parametru care beneficiază de o reducere mai mare sau egală decât un procent primit ca parametru.

```
CREATE OR REPLACE PROCEDURE statistica_client(
    p_id_client CLIENT.id_client%TYPE,
    p_procent_minim PROMOTIE.procent_reducere%TYPE) IS

    v_exista_client NUMBER;
    v_suma_totala NUMBER;

    NU_EXISTA_CLIENT EXCEPTION;
    PROCENT_INVALID EXCEPTION;

    BEGIN
        SELECT COUNT(*) INTO v_exista_client
        FROM CLIENT
        WHERE id_client = p_id_client;

        IF v_exista_client = 0 THEN
            RAISE NU_EXISTA_CLIENT;
        END IF;

        IF (p_procent_minim < 1) OR (p_procent_minim > 100) THEN
            RAISE PROCENT_INVALID;
        END IF;

        SELECT NVL(SUM(AP.pret * AP.cantitate), 0) INTO v_suma_totala
        FROM ACHIZITIE_PRODUS AP
        JOIN PRODUS P ON P.cod_produs = AP.cod_produs
        JOIN PROMOTIE PR ON PR.cod_promotie = P.cod_promotie
        JOIN ACHIZITIE AC ON AC.cod_tranzactie = AP.cod_tranzactie
        JOIN CLIENT C ON AC.id_client = C.id_client
```

```

WHERE C.id_client = p_id_client
AND PR.procent_reducere >= p_procent_minim;

DBMS_OUTPUT.PUT_LINE('Clientul cu id-ul ' || p_id_client || ' a cheltuit ' ||
v_suma_totala ||
' pe produse reduse cu peste ' || p_procent_minim || '%');

EXCEPTION
WHEN NU_EXISTA_CLIENT THEN
    RAISE_APPLICATION_ERROR(-20000, 'Nu exista clientul cu id-ul dat');
WHEN PROCENT_INVALID THEN
    RAISE_APPLICATION_ERROR(-20001, 'Procent invalid');
WHEN OTHERS THEN
    RAISE_APPLICATION_ERROR(-20002, 'Alta eroare:' || SQLERRM);
END;
/

```

- Pentru primul caz vom da ca parametru pentru procedură id-ul client 17 (care există) și valoare procentuală 10 (care este validă):

```

BEGIN
    statistica_client(17, 10);
END;
/

```

Rezultatul acestei cereri este următorul:

```

Procedure STATISTICA_CLIENT compiled

Clientul cu id-ul 17 a cheltuit 179.98 pe produse reduse cu peste 10%

PL/SQL procedure successfully completed.

```

- Pentru al doilea caz vom da ca parametru pentru procedură id-ul client 100 (care **NU** există) și valoare procentuală 10 (care este validă):

```
BEGIN
    statistica_client(100, 10);
END;
/
```

Rezultatul acestei cereri este următorul:

```
Procedure STATISTICA_CLIENT compiled

BEGIN
*
ERROR at line 1:
ORA-20000: Nu exista clientul cu id-ul dat
ORA-06512: at "C##ANDREISGBD.STATISTICA_CLIENT", line 38
ORA-06512: at line 2
```

<https://docs.oracle.com/error-help/db/ora-20000/>

More Details :

<https://docs.oracle.com/error-help/db/ora-20000/>

<https://docs.oracle.com/error-help/db/ora-06512/>

- Pentru al treilea caz vom da ca parametru pentru procedură id-ul client 17 (care există) și valoare procentuală 150 (care **NU** este validă):

```
BEGIN
    statistica_client(17, 150);
END;
/
```

Rezultatul acestei cereri este următorul:

```
Procedure STATISTICA_CLIENT compiled

BEGIN
*
ERROR at line 1:
ORA-20001: Procent invalid
ORA-06512: at "C##ANDREISGBD.STATISTICA_CLIENT", line 40
ORA-06512: at line 2
```

<https://docs.oracle.com/error-help/db/ora-20001/>

More Details :

<https://docs.oracle.com/error-help/db/ora-20001/>

<https://docs.oracle.com/error-help/db/ora-06512/>

10. Definirea unui trigger de tip LMD la nivel de comandă și declanșarea trigger-ului

Să se permită doar utilizatorului ADMIN_MAGAZIN să adauge, să modifice doar câmpul cantitate, dar nu să șteargă aprovizionările în intervalul orar 22:00 - 08:00. Utilizatorul MANAGER_LOGISTICA nu are restricții.

```
CREATE OR REPLACE TRIGGER actiuni_aprovizionare
BEFORE INSERT OR UPDATE OR DELETE ON APROVIZIONARE
BEGIN
    IF (USER = 'MANAGER_LOGISTICA') THEN
        NULL;
    ELSIF (USER LIKE 'ADMIN_MAGAZIN_%' AND TO_NUMBER(SUBSTR(USER, 15))
BETWEEN 1 AND 10) THEN
        IF (TO_CHAR(SYSDATE, 'HH24:MI') >= '08:00' AND TO_CHAR(SYSDATE, 'HH24:MI') <
'22:00') THEN
            RAISE_APPLICATION_ERROR(-20000, 'Nu se realizeaza aprovizionari in timpul
orelor de program');
        ELSE
            IF UPDATING THEN
                IF UPDATING('id_magazin') OR UPDATING('EUID') OR UPDATING('cod_produs')
OR UPDATING('data_aprovizionare') THEN
                    RAISE_APPLICATION_ERROR(-20001, 'Nu se poate modifica acest atribut');
                END IF;
            END IF;

            IF DELETING THEN
                RAISE_APPLICATION_ERROR(-20002, 'Nu aveti acces la stergere');
            END IF;
        END IF;
    ELSE
        RAISE_APPLICATION_ERROR(-20003, 'Nu aveti acces la comenzi de
administrator');
    END IF;
END;
```

- Pentru primul caz vom rula comanda **DELETE** la o oră nepermisă de pe user-ul MANAGER_LOGISTICA care nu are restricții:

```
BEGIN
    DBMS_OUTPUT.PUT_LINE('User curent: ' || USER);
    DBMS_OUTPUT.PUT_LINE('Ora curenta: ' || TO_CHAR(SYSDATE, 'HH24:MI'));
    DELETE FROM APROVIZIONARE WHERE id_magazin = 2;
    DBMS_OUTPUT.PUT_LINE('Randuri sterse: ' || SQL%ROWCOUNT);
END;
/
```

Rezultatul acestei cereri este următorul:

```
Trigger ACTIUNI_APROVIZIONARE compiled

User curent: MANAGER_LOGISTICA
Ora curenta: 19:12
Randuri sterse: 3

PL/SQL procedure successfully completed.
```

Observăm că operația **DELETE** s-a produs cu succes, deci trigger-ul a funcționat conform regulilor.

- Pentru al doilea caz vom rula aceeași operație **DELETE** la o oră nepermisă de pe user-ul ADMIN_MAGAZIN_3, rezultatul fiind următorul:

```
Trigger ACTIUNI_APROVIZIONARE compiled

User curent: ADMIN_MAGAZIN_3
Ora curenta: 19:20

BEGIN
*
ERROR at line 1:
ORA-20000: Nu se realizeaza aprovisionari in timpul orelor de program
ORA-06512: at "ADMIN_MAGAZIN_3.ACTIUNI_APROVIZIONARE", line 6
ORA-04088: error during execution of trigger 'ADMIN_MAGAZIN_3.ACTIUNI_APROVIZIONARE'
ORA-06512: at line 4

https://docs.oracle.com/error-help/db/ora-20000/
```

Observăm că acțiunea a fost blocată din cauză că ne aflăm în intervalul orar nepermis, deci trigger-ul funcționează corect.

Dacă rulăm aceeași comandă de pe același user la o oră permisă vom obține următorul rezultat:

```
Trigger ACTIUNI_APROVIZIONARE compiled

User curent: ADMIN_MAGAZIN_3
Ora curenta: 00:01

BEGIN
*
ERROR at line 1:
ORA-20002: Nu aveti acces la stergere
ORA-06512: at "ADMIN_MAGAZIN_3.ACTIUNI_APROVIZIONARE", line 15
ORA-04088: error during execution of trigger 'ADMIN_MAGAZIN_3.ACTIUNI_APROVIZIONARE'
ORA-06512: at line 7

https://docs.oracle.com/error-help/db/ora-20002/
```

Observăm că trigger-ul a blocat acțiunea deoarece user-ul nu are drepturi de ștergere.

- Pentru al treilea caz vom rula comanda **UPDATE** pe o coloană nepermisă la o oră permisă:

```
BEGIN
```

```
    DBMS_OUTPUT.PUT_LINE('User curent: ' || USER);
    DBMS_OUTPUT.PUT_LINE('Ora curenta: ' || TO_CHAR(SYSDATE, 'HH24:MI'));
    UPDATE APROVIZIONARE SET cod_produs = 10 WHERE id_magazin = 3;
    DBMS_OUTPUT.PUT_LINE('Randuri modificate: ' || SQL%ROWCOUNT);
END;
/
```

```
Trigger ACTIUNI_APROVIZIONARE compiled

User curent: ADMIN_MAGAZIN_3
Ora curenta: 00:36

BEGIN
*
ERROR at line 1:
ORA-20001: Nu se poate modifica acest atribut
ORA-06512: at "ADMIN_MAGAZIN_3.ACTIUNI_APROVIZIONARE", line 10
ORA-04088: error during execution of trigger 'ADMIN_MAGAZIN_3.ACTIUNI_APROVIZIONARE'
ORA-06512: at line 7

https://docs.oracle.com/error-help/db/ora-20001/
```

Observăm că acțiunea a fost blocată deoarece user-ul ADMIN_MAGAZIN are restricții pe operațiile LMD.

Vom rula aceeași comandă pe o coloană permisă:

```
BEGIN
```

```
    DBMS_OUTPUT.PUT_LINE('User curent: ' || USER);
    DBMS_OUTPUT.PUT_LINE('Ora curenta: ' || TO_CHAR(SYSDATE, 'HH24:MI'));
    UPDATE APROVIZIONARE SET cantitate = 10 WHERE cod_produs = 16;
    DBMS_OUTPUT.PUT_LINE('Randuri modificate: ' || SQL%ROWCOUNT);
END;
/
```

Rezultatul este următorul:

```
Trigger ACTIUNI_APROVIZIONARE compiled

User curent: ADMIN_MAGAZIN_3
Ora curenta: 00:39
Randuri modificate: 1

PL/SQL procedure successfully completed.
```

- Pentru al patrulea caz vom rula o comandă **INSERT** la o oră permisă:

```
BEGIN
    DBMS_OUTPUT.PUT_LINE('User curent: ' || USER);
    DBMS_OUTPUT.PUT_LINE('Ora curenta: ' || TO_CHAR(SYSDATE, 'HH24:MI'));
    INSERT INTO APROVIZIONARE (id_magazin, cod_produs, EUID, cantitate,
        data_aprovizionare)
        VALUES (3, 21, 'RO73925', 44, TO_DATE('11-02-2023', 'DD-MM-YYYY'));
    DBMS_OUTPUT.PUT_LINE('Randuri adaugate: ' || SQL%ROWCOUNT);
END;
/
```

Rezultatul acestei cereri este următorul:

```
Trigger ACTIUNI_APROVIZIONARE compiled

User curent: ADMIN_MAGAZIN_3
Ora curenta: 00:47
Randuri adaugate: 1

PL/SQL procedure successfully completed.
```

Observăm că insert-ul s-a realizat cu succes, deci trigger-ul funcționează corect.

- În final vom rula aceeași operație **DELETE** de la început la o oră nepermisă de pe user-ul CLIENT_MAGAZIN, rezultatul fiind următorul:

```
Trigger ACTIUNI_APROVIZIONARE compiled

User curent: CLIENT_MAGAZIN
Ora curenta: 19:32

BEGIN
*
ERROR at line 1:
ORA-20003: Nu aveti acces la comenzile de administrator
ORA-06512: at "CLIENT_MAGAZIN.ACTIUNI_APROVIZIONARE", line 19
ORA-04088: error during execution of trigger 'CLIENT_MAGAZIN.ACTIUNI_APROVIZIONARE'
ORA-06512: at line 4

https://docs.oracle.com/error-help/db/ora-20003/
```

Acțiunea este blocată indiferent de oră deoarece user-ul nu are acces la aceste comenzi.

11. Definirea unui trigger de tip LMD la nivel de linie și declanșarea trigger-ului

Să se interzică achiziționarea produselor expirate și, în cazul în care achiziția conține produse cu prețul peste 1000 de lei iar angajatul care o procesează are sub 1 an vechime, să se modifice statusul achiziției în "Necesită revizuire supervizor".

```
CREATE OR REPLACE TRIGGER revizuire_produse
BEFORE INSERT ON ACHIZITIE_PRODUS
FOR EACH ROW
DECLARE
    v_pret PRODUS.pret%TYPE;
    v_data_expirare PRODUS.data_expirare%TYPE;
    v_data_angajare ANGAJAT.data_angajare%TYPE;

BEGIN
    SELECT pret, data_expirare INTO v_pret, v_data_expirare
    FROM PRODUS
    WHERE cod_produs = :NEW.cod_produs;

    SELECT data_angajare INTO v_data_angajare
    FROM ANGAJAT
    JOIN ACHIZITIE AC ON AC.CNP_angajat = AN.CNP_angajat
    WHERE AC.cod_tranzactie = :NEW.cod_tranzactie;

    IF (v_data_expirare < TRUNC(SYSDATE)) THEN
        RAISE_APPLICATION_ERROR(-20000, 'Produs expirat');
    END IF;

    IF (v_pret > 1000) AND (TRUNC(MONTHS_BETWEEN(SYSDATE, v_data_angajare)) < 12)
    THEN
        UPDATE ACHIZITIE
        SET status = 'Necesita revizuire supervizor'
        WHERE cod_tranzactie = :NEW.cod_tranzactie;
    END IF;
END;
/
```

- Pentru primul caz vom inseră un produs expirat, iar apoi vom încerca să îl achiziționăm (să îl inserăm în tabelul ACHIZITIE_PRODUS):

```
INSERT INTO PRODUS (cod_produs, nume_produs, pret, cod_promotie, data_fabricare, data_expirare) VALUES (50, 'Paine', 5, NULL, TO_DATE('15-03-2024', 'DD-MM-YYYY'), TO_DATE('20-03-2024', 'DD-MM-YYYY'));
```

```
INSERT INTO ACHIZITIE_PRODUS (cod_produs, cod_tranzactie, cantitate, pret) VALUES (50, 1001, 2, 5);
```

Rezultatul acestor comenzi este:

```
Trigger REVIZUIRE_PRODUSE compiled

Error starting at line : 327 in command -
INSERT INTO ACHIZITIE_PRODUS (cod_produs, cod_tranzactie, cantitate, pret) VALUES (50, 1001, 2, 5)
Error at Command Line : 327 Column : 13
Error report -
SQL Error: ORA-20000: Produs expirat
ORA-06512: at "C##ANDREISGBD.REVIZUIRE_PRODUSE", line 17
ORA-04088: error during execution of trigger 'C##ANDREISGBD.REVIZUIRE_PRODUSE'

https://docs.oracle.com/error-help/db/ora-20000/20000.00000 - %s
*Cause: The stored procedure 'raise_application_error'
        was called which causes this error to be generated.
*Action: Correct the problem as described in the error message or contact
        the application administrator or DBA for more information.
```

Observăm că primim eroare din cauza datei de expirare, deci trigger-ul funcționează corect.

- Pentru al doilea caz, vom inseră într-o achiziție procesată de un angajat cu vechime sub 1 an un produs de peste 1000 de lei:

```
DECLARE
  v_pret PRODUS.pret%TYPE;
  v_data_angajare ANGAJAT.data_angajare%TYPE;
  v_status ACHIZITIE.status%TYPE;
```

```

BEGIN
    SELECT P.pret, AN.data_angajare, AC.status INTO v_pret, v_data_angajare, v_status
    FROM PRODUS P, ANGAJAT AN, ACHIZITIE AC
    WHERE P.cod_produs = 3 AND AN.CNP_angajat = 1750215789012 AND
    AC.cod_tranzactie = 1024;

    DBMS_OUTPUT.PUT_LINE('Inseram produsul cu pretul de ' || v_pret ||
    ' in achizitia procesata de un angajat cu vechimea de ' ||
    (TRUNC(MONTHS_BETWEEN(SYSDATE, v_data_angajare) / 12, 1))
    || ' ani');
    DBMS_OUTPUT.PUT_LINE('Status nou achizitie 1024: ' || v_status);

    INSERT INTO ACHIZITIE_PRODUS (cod_produs, cod_tranzactie, cantitate, pret)
    VALUES (3, 1024, 1, 1799.99);

    SELECT status INTO v_status
    FROM ACHIZITIE
    WHERE cod_tranzactie = 1024;

    DBMS_OUTPUT.PUT_LINE('Status nou achizitie 1024: ' || v_status);
END;
/

```

Rezultatul acestui bloc PL/SQL este următorul:

```

Trigger REVIZUIRE_PRODUSE compiled

Inseram produsul cu pretul de 1799.99 in achizitia procesata de un angajat cu vechimea de .6 ani
Status initial achizitie 1024: In procesare
Status actual achizitie 1024: Necesita revizuire supervisor

PL/SQL procedure successfully completed.

```

Observăm că, dacă încercăm să inserăm produsul cu codul 3 (Telefon mobil) cu prețul peste 1000 de lei în achiziția cu codul 1024 procesată de angajatul cu CNP-ul 1750215789012 (vechime 0.6 ani), atunci statusul tranzacției se modifică conform regulilor, deci trigger-ul funcționează corect.

- Pentru al treilea caz, vom inseră într-o achiziție procesată de un angajat cu vechime peste 1 an un produs de peste 1000 de lei. Vom rula blocul PL/SQL anterior doar ca vom modifica tranzacția cu una procesată de alt angajat, rezultatul fiind următorul:

```
Trigger REVIZUIRE_PRODUSE compiled

Inseram produsul cu pretul de 1799.99 in achizitia procesata de un angajat cu vechimea de 5.5 ani
Status initial achizitie 1010: In procesare
Status actual achizitie 1010: In procesare

PL/SQL procedure successfully completed.
```

Observăm că, dacă încercăm să inserăm produsul cu codul 3 (Telefon mobil) cu prețul peste 1000 de lei în achiziția cu codul 1010 procesată de angajatul cu CNP-ul 1850528789012 (vechime 5.5 ani), atunci statusul tranzacției nu se modifică, deci trigger-ul funcționează corect.

12. Definirea unui trigger de tip LDD și declanșarea trigger-ului

Să se permită doar user-ului MANAGER_LOGISTICA operațiile LDD doar în intervalul orar 22:00 - 08:00. Vom lăsa userii SYS și SYSTEM fără restricții pentru a păstra o metodă de a avea controlul deplin asupra bazei de date.

```
CREATE OR REPLACE TRIGGER permisiuni_LDD
BEFORE DDL ON DATABASE
BEGIN
    IF (USER IN ('SYS', 'SYSTEM')) THEN
        NULL;
    ELSIF (USER = 'MANAGER_LOGISTICA') THEN
        IF (TO_CHAR(SYSDATE, 'HH24:MI') >= '08:00' AND TO_CHAR(SYSDATE, 'HH24:MI') < '22:00') THEN
            RAISE_APPLICATION_ERROR(-20000, 'Nu se pot realiza modificari in timpul orelor de program');
        ELSE
            NULL;
        END IF;
    ELSE
        RAISE_APPLICATION_ERROR(-20001, 'Nu aveti acces la comenziile asupra bazei de date');
    END IF;
END;
/
```

- Pentru primul caz vom rula comanda **DROP** pe tabelul CLIENT de pe user-ul MANAGER_LOGISTICA la o oră nepermisă:

```
BEGIN
    DBMS_OUTPUT.PUT_LINE('User curent: ' || USER);
    DBMS_OUTPUT.PUT_LINE('Ora curenta: ' || TO_CHAR(SYSDATE, 'HH24:MI'));
    EXECUTE IMMEDIATE 'DROP TABLE CLIENT CASCADE CONSTRAINTS';
    DBMS_OUTPUT.PUT_LINE('Tabelul CLIENT a fost sters');
END;
/
```

Rezultatul acestei cereri este următorul:

```
Trigger PERMISIUNI_LDD compiled

User current: MANAGER_LOGISTICA
Ora curenta: 19:48

BEGIN
*
ERROR at line 1:
ORA-04088: error during execution of trigger 'MANAGER_LOGISTICA.PERMISIUNI_LDD'
ORA-00604: error occurred at recursive SQL level 2
ORA-20000: Nu se pot realiza modificari in timpul orelor de program
ORA-06512: at line 6
ORA-06512: at line 4
```

Observăm că acțiunea a fost blocată din cauză că ne aflăm în intervalul orar nepermis, deci trigger-ul funcționează corect.

- Pentru al doilea caz vom rula aceeași operație **DROP** la o oră nepermisă de pe user- ul ADMIN_MAGAZIN_3, rezultatul fiind următorul:

```
Trigger PERMISIUNI_LDD compiled

User current: ADMIN_MAGAZIN_3
Ora curenta: 19:59

BEGIN
*
ERROR at line 1:
ORA-04088: error during execution of trigger 'ADMIN_MAGAZIN_3.PERMISIUNI_LDD'
ORA-00604: error occurred at recursive SQL level 2
ORA-20001: Nu aveti acces la comenzile asupra bazei de date
ORA-06512: at line 11
ORA-06512: at line 4
```

<https://docs.oracle.com/error-help/db/ora-04088/>

Observăm că acțiunea este blocată indiferent de oră deoarece user-ul nu are acces la aceste comenzi.

- Pentru al treilea caz vom rula comandă **DROP** pe tabelul CLIENT de pe user-ul MANAGER_LOGISTICA la o oră permisă:

```
Trigger PERMISIUNI_LDD compiled

User curent: MANAGER_LOGISTICA
Ora curenta: 23:47
Tabelul CLIENT a fost sters

PL/SQL procedure successfully completed.
```

Observăm că operația s-a realizat cu succes, deci trigger-ul a respectat regulile impuse.

- Vom rula aceeași comanda **DROP** de pe user-ul SYS, rezultatul fiind următorul:

```
Trigger PERMISIUNI_LDD compiled

User curent: SYS
Ora curenta: 18:32

PL/SQL procedure successfully completed.
```

Observăm că user-ul SYS nu are restricții, deci trigger-ul funcționează corect.

13. Formularea în limbaj natural a unei probleme care se rezolvă folosind un pachet care include tipuri de date complexe și obiecte necesare unui flux de acțiuni integrate

Să se creeze un pachet ce analizează performanța angajaților pe baza vânzărilor procesate. Vom calcula vânzările totale pentru fiecare angajat dintr-un magazin dat și, pentru angajații care au vânzări peste un prag stabilit, vom crește salariul în funcție de cât au vândut. Înainte de a crește salariul vom putea afișa o colecție cu angajații eligibili, iar abia apoi se va lua decizia de a se modifica sau nu.

```
CREATE OR REPLACE PACKAGE performanta_angajati IS
    TYPE rec_raport_angajati IS RECORD (
        CNP_angajat ANGAJAT.CNP_angajat%TYPE,
        nume_complet ANGAJAT.nume_complet%TYPE,
        salariu_curent ANGAJAT.salariu%TYPE,
        total_vanzari NUMBER,
        salariu_nou ANGAJAT.salariu%TYPE
    );
    TYPE raport_angajati_type IS TABLE OF rec_raport_angajati;
    FUNCTION total_vanzari_angajat(p_CNP_angajat ANGAJAT.CNP_angajat%TYPE)
    RETURN NUMBER;
    FUNCTION calcul_salariu_nou(p_salariu ANGAJAT.salariu%TYPE, p_vanzari_angajat
    NUMBER) RETURN NUMBER;
    PROCEDURE angajati_eligibili(p_id_magazin MAGAZIN.id_magazin%TYPE,
    p_prag_vanzari NUMBER);
    PROCEDURE majorare_salarii(p_id_magazin MAGAZIN.id_magazin%TYPE,
    p_prag_vanzari NUMBER);
    MAGAZIN_INEXISTENT EXCEPTION;
END performanta_angajati;
/

CREATE OR REPLACE PACKAGE BODY performanta_angajati IS
    FUNCTION total_vanzari_angajat(p_CNP_angajat ANGAJAT.CNP_angajat%TYPE)
    RETURN NUMBER IS
        v_total NUMBER := 0;
```

```

BEGIN
    SELECT NVL(SUM(AP.cantitate * AP.pret), 0)
    INTO v_total
    FROM ACHIZITIE_PRODUS AP
    JOIN ACHIZITIE A ON A.cod_tranzactie = AP.cod_tranzactie
    WHERE A.CNP_angajat = p_CNP_angajat;

    RETURN v_total;
END total_vanzari_angajat;

FUNCTION calcul_salariu_nou(p_salariu ANGAJAT.salariu%TYPE, p_vanzari_angajat
NUMBER) RETURN NUMBER IS
BEGIN
    IF p_vanzari_angajat >= (p_salariu * 5) THEN
        RETURN p_salariu * 1.10;
    ELSIF p_vanzari_angajat >= (p_salariu * 3) THEN
        RETURN p_salariu * 1.05;
    ELSE
        RETURN p_salariu;
    END IF;
END calcul_salariu_nou;

PROCEDURE angajati_eligibili(p_id_magazin MAGAZIN.id_magazin%TYPE,
p_prag_vanzari NUMBER) IS
    v_count_magazin NUMBER;
    t_raport_angajati raport_angajati_type;
BEGIN
    t_raport_angajati := raport_angajati_type();

    SELECT COUNT(*) INTO v_count_magazin
    FROM MAGAZIN
    WHERE id_magazin = p_id_magazin;

    IF v_count_magazin = 0 THEN
        RAISE MAGAZIN_INEXISTENT;
    END IF;

    FOR i IN (SELECT *
              FROM ANGAJAT
              WHERE id_magazin = p_id_magazin) LOOP

        IF total_vanzari_angajat(i.CNP_angajat) >= p_prag_vanzari THEN

```

```

        IF calcul_salariu_nou(i.salariu, total_vanzari_angajat(i.CNP_angajat)) > i.salariu
    THEN

        t_raport_angajati.EXTEND;
        t_raport_angajati(t_raport_angajati.LAST).CNP_angajat := i.CNP_angajat;
        t_raport_angajati(t_raport_angajati.LAST).nume_complet := i.nume_complet;
        t_raport_angajati(t_raport_angajati.LAST).salariu_curent := i.salariu;
        t_raport_angajati(t_raport_angajati.LAST).total_vanzari :=
        total_vanzari_angajat(i.CNP_angajat);
        t_raport_angajati(t_raport_angajati.LAST).salariu_nou :=
        calcul_salariu_nou(i.salariu, total_vanzari_angajat(i.CNP_angajat));
    END IF;
    END IF;
END LOOP;

IF t_raport_angajati.COUNT > 0 THEN
    FOR i IN 1..t_raport_angajati.COUNT LOOP
        DBMS_OUTPUT.PUT_LINE('*Angajat ' || i || ':');
        DBMS_OUTPUT.PUT_LINE('CNP: ' || t_raport_angajati(i).CNP_angajat);
        DBMS_OUTPUT.PUT_LINE('Nume: ' || t_raport_angajati(i).nume_complet);
        DBMS_OUTPUT.PUT_LINE('Salariu curent: ' || t_raport_angajati(i).salariu_curent);
        DBMS_OUTPUT.PUT_LINE('Vanzari totale: ' || t_raport_angajati(i).total_vanzari);
        DBMS_OUTPUT.PUT_LINE('Salariu nou: ' || t_raport_angajati(i).salariu_nou);
        DBMS_OUTPUT.PUT_LINE(' ');
    END LOOP;
ELSE
    DBMS_OUTPUT.PUT_LINE('Niciun angajat eligibil');
END IF;
EXCEPTION
    WHEN MAGAZIN_INEXISTENT THEN
        RAISE_APPLICATION_ERROR(-20000, 'Magazinul nu exista');
END angajati_eligibili;

PROCEDURE majorare_salarii(p_id_magazin MAGAZIN.id_magazin%TYPE,
p_prag_vanzari NUMBER) IS
    v_count_magazin NUMBER;
    v_modificari NUMBER := 0;
BEGIN
    SELECT COUNT(*) INTO v_count_magazin
    FROM MAGAZIN
    WHERE id_magazin = p_id_magazin;

    IF v_count_magazin = 0 THEN
        RAISE MAGAZIN_INEXISTENT;
    END IF;
    IF p_prag_vanzari > 0 THEN
        v_modificari := p_prag_vanzari;
    ELSE
        v_modificari := 1;
    END IF;
    FOR i IN 1..v_count_magazin LOOP
        IF t_raport_angajati(i).total_vanzari <= v_modificari THEN
            t_raport_angajati(i).total_vanzari := t_raport_angajati(i).total_vanzari + 1;
        END IF;
    END LOOP;
    DBMS_OUTPUT.PUT_LINE('Salariile au fost majorate cu ' ||
        v_modificari || ' unitati');
END majorare_salarii;

```

```

END IF;

FOR i IN (SELECT *
           FROM ANGAJAT
          WHERE id_magazin = p_id_magazin) LOOP

  IF total_vanzari_angajat(i.CNP_angajat) >= p_prag_vanzari THEN
    IF calcul_salariu_nou(i.salariu, total_vanzari_angajat(i.CNP_angajat)) > i.salariu
    THEN

      UPDATE ANGAJAT
      SET salariu = calcul_salariu_nou(i.salariu,
total_vanzari_angajat(i.CNP_angajat))
      WHERE CNP_angajat = i.CNP_angajat;

      v_modificari := v_modificari + 1;
    END IF;
  END IF;
END LOOP;

IF v_modificari > 0 THEN
  COMMIT;
  DBMS_OUTPUT.PUT_LINE('Salarii majorate: ' || v_modificari);
ELSE
  DBMS_OUTPUT.PUT_LINE('Niciun angajat eligibil pentru majorare');
END IF;

EXCEPTION
  WHEN MAGAZIN_INEXISTENT THEN
    RAISE_APPLICATION_ERROR(-20000, 'Magazinul nu exista');
  END majorare_salarii;
END performanta_angajati;
/

```

Vom insera următorul produs într-o tranzacție:

```
INSERT INTO ACHIZITIE_PRODUS (cod_produs, cod_tranzactie, cantitate, pret) VALUES
(1, 1003, 50, 2500);
```

Pentru a afla cărui magazin îi corespunde această tranzacție (cod_tranzactie = 1003) vom rula următoarea interogare SQL:

```
SELECT M.id_magazin
FROM MAGAZIN M
JOIN ANGAJAT AN ON AN.id_magazin = M.id_magazin
JOIN ACHIZITIE AC ON AC.CNP_angajat = AN.CNP_angajat
WHERE AC.cod_tranzactie = 1003;
```

Rezultatul interogării este următorul:

ID_MAGAZIN
1
3

- Pentru primul caz vom rula procedura angajati_eligibili pentru magazinul cu id-ul 3 și pragul 1000:

```
BEGIN
    performanta_angajati.angajati_eligibili(3, 1000);
END;
/
```

Rezultatul acestei comenzi este următorul:

```
Package PERFORMANTA_ANGAJATI compiled

Package Body PERFORMANTA_ANGAJATI compiled

*Angajat 1:
CNP: 1750215789012
Nume: Roby Will
Salariu curent: 6000
Vanzari totale: 126919.97
Salariu nou: 6600

PL/SQL procedure successfully completed.
```

- Pentru al doilea caz vom rula procedura majorare_salarii pentru magazinul cu id-ul 3 și pragul 1000:

```
BEGIN
    performanta_angajati.majorare_salarii(3, 1000);
END;
/
```

Rezultatul acestei cereri este următorul:

```
Package PERFORMANTA_ANGAJATI compiled

Package Body PERFORMANTA_ANGAJATI compiled

Salarii majorate: 1

PL/SQL procedure successfully completed.
```

Observăm că a fost modificat salariul unui angajat, același număr de angajați afișați de procedura anterioară.

- Pentru al treilea vom rula procedura anterioară, dar vom da ca parametru un id de magazin inexistent:

```
BEGIN
    performanta_angajati.majorare_salarii(100, 1000);
END;
/
```

Rezultatul acestei cereri este următorul:

```
Package PERFORMANTA_ANGAJATI compiled

Package Body PERFORMANTA_ANGAJATI compiled

BEGIN
*
ERROR at line 1:
ORA-20000: Magazinul nu exista
ORA-06512: at "C##ANDREISGBD.PERFORMANTA_ANGAJATI", line 113
ORA-06512: at line 2
```

Pentru acest **flux de acțiuni integrate** am utilizat 2 funcții ce calculează informații necesare în proceduri și 2 proceduri “principale”, una pentru afișarea angajaților eligibili pentru modificări, iar una care efectuează modificările. Ca tipuri de date am utilizat un RECORD și un tabel imbricat, ambele fiind utilizate pentru a reține și afișa angajații eligibili pentru modificări.