PROIECT SGBD

MAGAZIN MATERIALE DE CONSTRUCȚII

***Cerinta 1***

**Prezentarea bazei de date:**

Baza de date pentru magazinul de materiale de constructii este conceputa pentru a gestiona eficient toate aspectele operationale ale afacerii, oferind un sistem relational robust si bine structurat. Modelul asigura o evidenta clara a clientilor, comenzilor, produselor si proceselor logistice, fiind astfel adaptat pentru a sprijini functionarea unui magazin modern.

Clientii joaca un rol central in aceasta schema, fiind conectati direct cu comenzile pe care le plaseaza. Fiecare comanda poate contine mai multe produse, iar aceste produse sunt organizate in categorii clare pentru o gestionare mai usoara. Produsele sunt asociate cu depozitele in care sunt stocate, ceea ce permite magazinului sa urmareasca stocurile si sa optimizeze procesul de livrare catre clienti.

In cadrul magazinului, fiecare produs poate face parte din una sau mai multe promotii, ceea ce permite crearea de campanii de marketing eficiente. Detaliile despre produse, cum ar fi specificatiile tehnice, sunt gestionate separat pentru a asigura un nivel ridicat de flexibilitate in administrarea informatiilor.

Comenzile sunt gestionate impreuna cu detalii precum metodele de plata si optiunile de livrare, ceea ce ofera clientilor o experienta personalizata si usor de adaptat la nevoile lor. De asemenea, exista o relatie bine definita intre personalul magazinului si locatiile in care acestia lucreaza. Fiecare angajat este legat de un contract de angajare, iar vanzatorii sunt asociati magazinelor in care activeaza.

Baza de date este construita in jurul unei structuri care permite extinderea usoara. Relatiile dintre entitati sunt clar definite, ceea ce face ca sistemul sa fie scalabil si usor de gestionat. De exemplu, daca magazinul doreste sa isi extinda inventarul sau sa introduca noi metode de livrare, aceasta poate fi implementata fara modificari majore ale structurii existente.

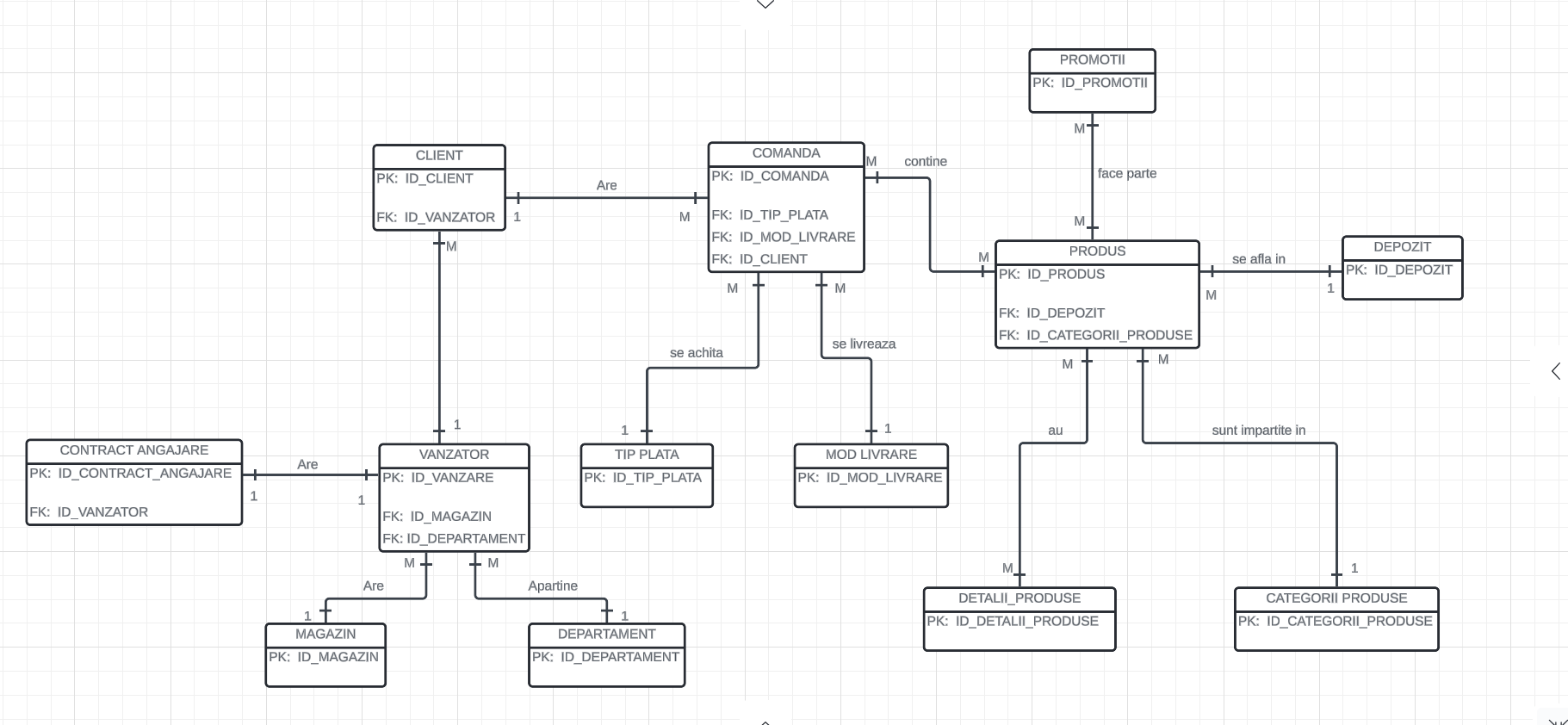
**Baza de date contine urmatoarele tabele principale:**

* **Magazin** (ID\_MAGAZIN, nume\_magazin, adresa\_magazin, program\_magazin) – pastreaza informatii despre locatiile fizice ale magazinelor.
* **Vanzator** (ID\_VANZATOR, ID\_MAGAZIN, nume\_vanzator, prenume\_vanzator, telefon\_vanzator, data\_angajare\_vanzator) – retine informatii despre angajatii magazinului si in ce locatie activeaza.
* **Client** (ID\_CLIENT, ID\_VANZATOR, nume\_client, prenume\_client) – stocheaza datele clientilor si vanzatorii cu care au interactionat.
* **Contract Angajare** (ID\_CONTRACT\_ANGAJARE, ID\_VANZATOR, salariu, tip\_contract) – gestioneaza informatiile despre contractele de munca ale vanzatorilor.
* **Depozit** (ID\_DEPOZIT, capacitate\_depozit, locatie\_depozit) – ofera detalii despre locatiile unde sunt stocate materialele de constructie.
* **Categorii Produse** (ID\_CATEGORIE\_PRODUS, denumire\_categorie, descriere\_categorie) – clasifica produsele in functie de tipul materialelor.
* **Produse** (ID\_PRODUS, ID\_CATEGORIE\_PRODUS, ID\_DEPOZIT, nume\_produs, descriere\_produs, pret\_produs, stoc\_produs) – pastreaza detalii despre articolele disponibile la vanzare.
* **Comanda** (ID\_COMANDA, ID\_CLIENT, ID\_TIP\_PLATA, ID\_MOD\_LIVRARE, data\_comanda, interval\_livrare) – stocheaza comenzile plasate de clienti si detalii despre plata si livrare.
* **Promotii** (ID\_PROMOTII, nume\_promotie, descriere\_promotie) – gestioneaza ofertele speciale aplicate produselor.
* **Detalii Produse** (ID\_DETALII\_PRODUSE, material, dimensiuni, greutate) – contine informatii suplimentare despre produsele din magazin.
* **Departamente** (ID\_DEPARTAMENT, denumire\_departament, status) – contine informatii despre diferitele sectiuni sau divizii din cadrul magazinului, fiecare avand un rol specific in gestionarea activitatilor si a resurselor.

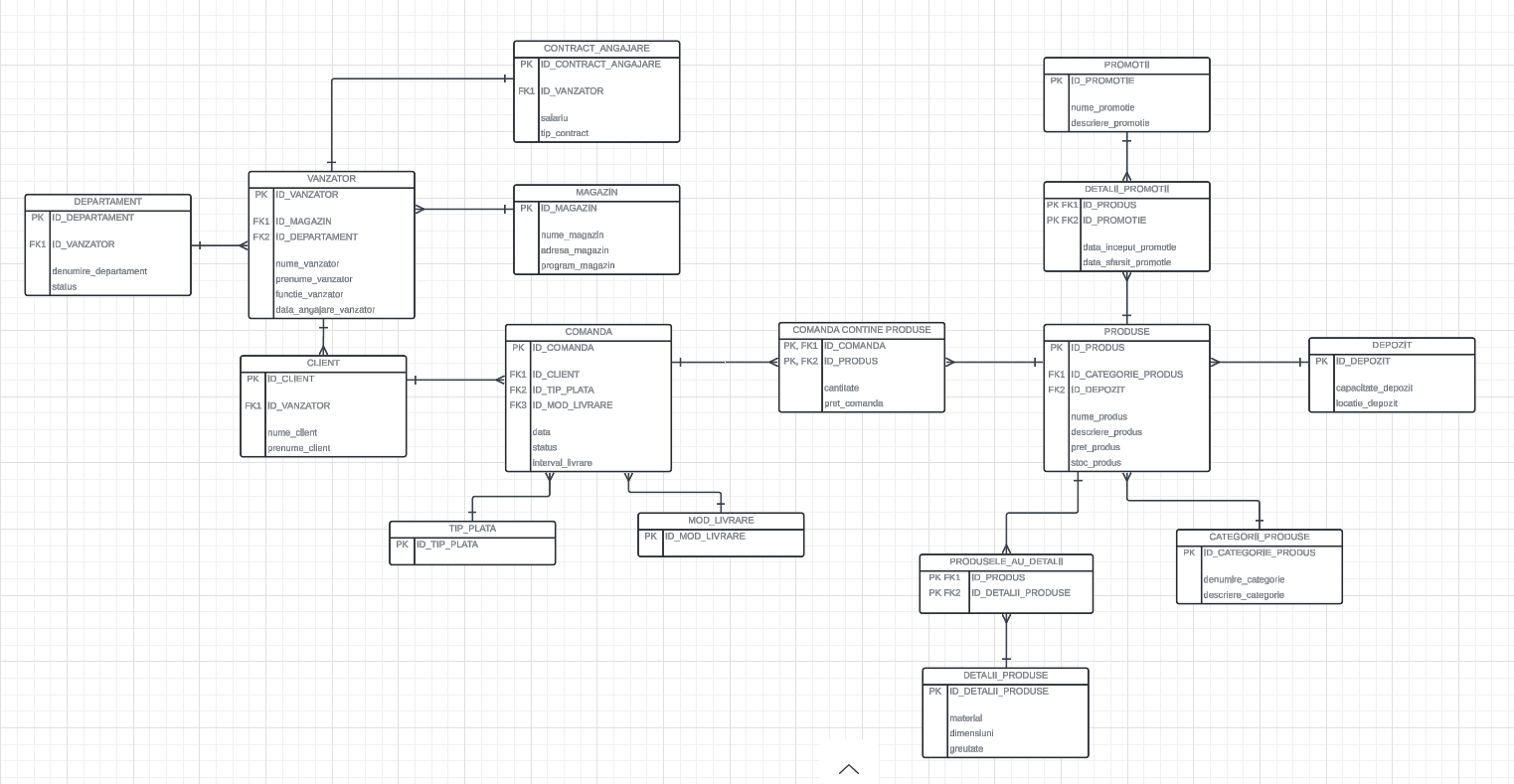
**Reguli de functionalitate ale bazei de date:**

1. **Fiecare magazin** are un ID unic si poate avea mai multi vanzatori asociati.
2. **Fiecare vanzator** este alocat unui magazin si poate interactiona cu mai multi clienti.
3. **Clientii pot plasa comenzi** si pot selecta metode diferite de plata si livrare.
4. **Produsele sunt clasificate** in categorii distincte si pot fi stocate in depozite diferite.
5. **Fiecare comanda** este asociata unui client si contine unul sau mai multe produse.
6. **Promotiile sunt aplicate** anumitor produse si au o perioada de valabilitate clar definita.
7. **Fiecare produs poate avea caracteristici suplimentare**, cum ar fi materialul, dimensiunile si greutatea.
8. **Comenzile sunt inregistrate impreuna cu detalii despre pret, cantitate si livrare**.

***Cerinta 2***

** Diagrama entitate-relatie(ERD):**

***Cerinta 3***

**Diagrama conceptuala:**

***Cerinta 4***

**Urmează să implementăm în Oracle diagrama conceptuală realizată pentru baza de date a magazinului de materiale de construcții. Aceasta va include definirea tuturor tabelelor necesare, asigurându-ne că fiecare tabel respectă structura stabilită. Vom adăuga toate constrângerile de integritate esențiale, inclusiv chei primare și chei externe, pentru a garanta consistența și corectitudinea datelor.**

**Fiecare tabel va fi creat cu atenție pentru a reflecta relațiile dintre entități și pentru a permite gestionarea eficientă a informațiilor despre magazine, vânzători, clienți, produse, depozite, comenzi și promoții. De asemenea, vom include detaliile relevante pentru fiecare entitate, astfel încât baza de date să fie scalabilă și ușor de administrat.**

**Prin implementarea acestor elemente, ne asigurăm că sistemul va putea gestiona datele într-un mod robust, permițând extinderea facilă a structurii în viitor, în cazul în care apar noi cerințe operaționale.**

**-- 1. TABEL MAGAZIN**

CREATE TABLE MAGAZIN (

ID\_MAGAZIN INT PRIMARY KEY,

nume\_magazin VARCHAR2(100),

adresa\_magazin VARCHAR2(255),

program\_magazin VARCHAR2(100)

);

**-- 2. TABEL VANZATOR**

CREATE TABLE VANZATOR (

ID\_VANZATOR INT PRIMARY KEY,

ID\_MAGAZIN INT NOT NULL,

ID\_DEPARTAMENT INT,

nume\_vanzator VARCHAR2(50),

prenume\_vanzator VARCHAR2(50),

telefon\_vanzator VARCHAR2(15),

data\_angajare\_vanzator DATE,

CONSTRAINT FK\_DEPARTAMENTFOREIGN KEY (ID\_DEPARTAMENT) REFERENCES DEPARTAMENT(ID\_DEPARTAMENT)

FOREIGN KEY (ID\_MAGAZIN) REFERENCES MAGAZIN(ID\_MAGAZIN)

);

**-- 3. TABEL CLIENT**

CREATE TABLE CLIENT (

ID\_CLIENT INT PRIMARY KEY,

ID\_VANZATOR INT NOT NULL,

nume\_client VARCHAR2(50),

prenume\_client VARCHAR2(50),

FOREIGN KEY (ID\_VANZATOR) REFERENCES VANZATOR(ID\_VANZATOR)

);

**-- 4. TABEL CONTRACT\_ANGAJARE**

CREATE TABLE CONTRACT\_ANGAJARE (

ID\_CONTRACT\_ANGAJARE INT PRIMARY KEY,

ID\_VANZATOR INT NOT NULL,

salariu DECIMAL(10,2),

tip\_contract VARCHAR2(50),

FOREIGN KEY (ID\_VANZATOR) REFERENCES VANZATOR(ID\_VANZATOR)

);

**-- 5. TABEL TIP\_PLATA**

CREATE TABLE TIP\_PLATA (

ID\_TIP\_PLATA INT PRIMARY KEY,

nume\_tip\_plata VARCHAR2(50) NOT NULL

);

**-- 6. TABEL MOD\_LIVRARE**

CREATE TABLE MOD\_LIVRARE (

ID\_MOD\_LIVRARE INT PRIMARY KEY

);

**-- 7. TABEL COMANDA**

CREATE TABLE COMANDA (

ID\_COMANDA INT PRIMARY KEY,

ID\_CLIENT INT NOT NULL,

ID\_TIP\_PLATA INT NOT NULL,

ID\_MOD\_LIVRARE INT NOT NULL,

data\_comanda DATE,

interval\_livrare VARCHAR2(100),

FOREIGN KEY (ID\_CLIENT) REFERENCES CLIENT(ID\_CLIENT),

FOREIGN KEY (ID\_TIP\_PLATA) REFERENCES TIP\_PLATA(ID\_TIP\_PLATA),

FOREIGN KEY (ID\_MOD\_LIVRARE) REFERENCES MOD\_LIVRARE(ID\_MOD\_LIVRARE)

);

**-- 8. TABEL DEPOZIT**

CREATE TABLE DEPOZIT (

ID\_DEPOZIT INT PRIMARY KEY,

capacitate\_depozit INT,

locatie\_depozit VARCHAR2(255)

);

**-- 9. TABEL CATEGORII\_PRODUSE**

CREATE TABLE CATEGORII\_PRODUSE (

ID\_CATEGORIE\_PRODUS INT PRIMARY KEY,

denumire\_categorie VARCHAR2(100),

descriere\_categorie CLOB

);

**-- 10. TABEL PRODUSE**

CREATE TABLE PRODUSE (

ID\_PRODUS INT PRIMARY KEY,

ID\_CATEGORIE\_PRODUS INT NOT NULL,

ID\_DEPOZIT INT NOT NULL,

nume\_produs VARCHAR2(50),

descriere\_produs CLOB,

pret\_produs DECIMAL(10,2),

stoc\_produs INT,

FOREIGN KEY (ID\_CATEGORIE\_PRODUS) REFERENCES CATEGORII\_PRODUSE(ID\_CATEGORIE\_PRODUS),

FOREIGN KEY (ID\_DEPOZIT) REFERENCES DEPOZIT(ID\_DEPOZIT)

);

**-- 11. TABEL DETALII\_PRODUSE**

CREATE TABLE DETALII\_PRODUSE (

ID\_DETALII\_PRODUSE INT PRIMARY KEY,

material VARCHAR2(50),

dimensiuni VARCHAR2(50),

greutate DECIMAL(10,2)

);

**-- 12. TABEL PRODUSE\_AU\_DETALII**

CREATE TABLE PRODUSE\_AU\_DETALII (

ID\_PRODUS INT,

ID\_DETALII\_PRODUSE INT,

PRIMARY KEY (ID\_PRODUS, ID\_DETALII\_PRODUSE),

FOREIGN KEY (ID\_PRODUS) REFERENCES PRODUSE(ID\_PRODUS),

FOREIGN KEY (ID\_DETALII\_PRODUSE) REFERENCES DETALII\_PRODUSE(ID\_DETALII\_PRODUSE)

);

**-- 13. TABEL PROMOTII**

CREATE TABLE PROMOTII (

ID\_PROMOTII INT PRIMARY KEY,

nume\_promotie VARCHAR2(100),

descriere\_promotie CLOB

);

**-- 14. TABEL DETALII\_PROMOTII**

CREATE TABLE DETALII\_PROMOTII (

ID\_PRODUS INT,

ID\_PROMOTIE INT,

data\_inceput\_promotie DATE,

data\_sfarsit\_promotie DATE,

PRIMARY KEY (ID\_PRODUS, ID\_PROMOTIE),

FOREIGN KEY (ID\_PRODUS) REFERENCES PRODUSE(ID\_PRODUS),

FOREIGN KEY (ID\_PROMOTIE) REFERENCES PROMOTII(ID\_PROMOTII)

);

**-- 15. TABEL COMANDA\_CONTINE\_PRODUSE**

CREATE TABLE COMANDA\_CONTINE\_PRODUSE (

ID\_COMANDA INT,

ID\_PRODUS INT,

cantitate INT,

pret\_comanda DECIMAL(10,2),

PRIMARY KEY (ID\_COMANDA, ID\_PRODUS),

FOREIGN KEY (ID\_COMANDA) REFERENCES COMANDA(ID\_COMANDA),

FOREIGN KEY (ID\_PRODUS) REFERENCES PRODUSE(ID\_PRODUS)

);

**-- 16. TABEL DEPARTAMENT**

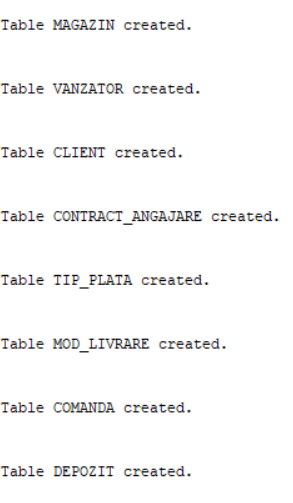
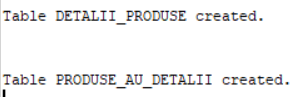
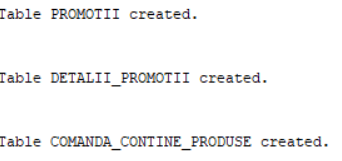
CREATE TABLE DEPARTAMENT (

ID\_DEPARTAMENT INT PRIMARY KEY,

denumire\_departament VARCHAR2(100),

status VARCHAR2(10) CHECK (status IN ('Active', 'Inactive'))

);



***Cerinta 5***

**După definirea și implementarea tabelelor în Oracle, urmează popularea acestora cu informații relevante și coerente. Vom introduce cel puțin cinci înregistrări pentru fiecare entitate independentă, astfel încât baza de date să conțină suficiente date pentru a reflecta scenariile reale de utilizare.**

**Pentru tabelele asociative, unde relațiile dintre entități sunt gestionate, vom adăuga un număr minim de zece înregistrări. Acest lucru va permite testarea corectă a interdependențelor dintre date și asigurarea că baza de date funcționează conform așteptărilor.**

**Prin adăugarea acestor date, vom putea verifica integritatea constrângerilor impuse, precum și modul în care informațiile sunt organizate și utilizate în sistem. Această etapă este esențială pentru validarea funcționalității bazei de date și pentru pregătirea acesteia în vederea utilizării într-un mediu operațional.**

**-- 1. INSERARE MAGAZIN**

INSERT INTO MAGAZIN (ID\_MAGAZIN, nume\_magazin, adresa\_magazin, program\_magazin)

VALUES (1, 'Magazin Central', 'Strada Unirii 10', '08:00 - 22:00');

INSERT INTO MAGAZIN (ID\_MAGAZIN, nume\_magazin, adresa\_magazin, program\_magazin)

VALUES (2, 'Magazin Vest', 'Strada Libertatii 5', '09:00 - 21:00');

INSERT INTO MAGAZIN (ID\_MAGAZIN, nume\_magazin, adresa\_magazin, program\_magazin)

VALUES (3, 'Magazin Est', 'Strada Independentei 15', '07:00 - 23:00');

INSERT INTO MAGAZIN (ID\_MAGAZIN, nume\_magazin, adresa\_magazin, program\_magazin)

VALUES (4, 'Magazin Nord', 'Strada Mihai Viteazul 3', '10:00 - 20:00');

INSERT INTO MAGAZIN (ID\_MAGAZIN, nume\_magazin, adresa\_magazin, program\_magazin)

VALUES (5, 'Magazin Sud', 'Strada Carol 7', '08:30 - 21:30');



**-- 2. INSERARE VANZATOR**

INSERT INTO VANZATOR (ID\_VANZATOR, ID\_MAGAZIN, nume\_vanzator, prenume\_vanzator, telefon\_vanzator, data\_angajare\_vanzator)

VALUES (1, 1, 'Popescu', 'Ion', '0745123456', TO\_DATE('2018-06-15', 'YYYY-MM-DD'));

INSERT INTO VANZATOR (ID\_VANZATOR, ID\_MAGAZIN, nume\_vanzator, prenume\_vanzator, telefon\_vanzator, data\_angajare\_vanzator)

VALUES (2, 1, 'Ionescu', 'Maria', '0745234567', TO\_DATE('2019-07-20', 'YYYY-MM-DD'));

INSERT INTO VANZATOR (ID\_VANZATOR, ID\_MAGAZIN, nume\_vanzator, prenume\_vanzator, telefon\_vanzator, data\_angajare\_vanzator)

VALUES (3, 1, 'Georgescu', 'Ana', '0745345678', TO\_DATE('2020-01-10', 'YYYY-MM-DD'));

INSERT INTO VANZATOR (ID\_VANZATOR, ID\_MAGAZIN, nume\_vanzator, prenume\_vanzator, telefon\_vanzator, data\_angajare\_vanzator)

VALUES (4, 2, 'Dumitru', 'Mihai', '0745456789', TO\_DATE('2021-03-05', 'YYYY-MM-DD'));

INSERT INTO VANZATOR (ID\_VANZATOR, ID\_MAGAZIN, nume\_vanzator, prenume\_vanzator, telefon\_vanzator, data\_angajare\_vanzator)

VALUES (5, 2, 'Constantin', 'Elena', '0745567890', TO\_DATE('2017-11-25', 'YYYY-MM-DD'));

INSERT INTO VANZATOR (ID\_VANZATOR, ID\_MAGAZIN, nume\_vanzator, prenume\_vanzator, telefon\_vanzator, data\_angajare\_vanzator)

VALUES (6, 3, 'Marinescu', 'Adrian', '0745678901', TO\_DATE('2020-09-10', 'YYYY-MM-DD'));

INSERT INTO VANZATOR (ID\_VANZATOR, ID\_MAGAZIN, nume\_vanzator, prenume\_vanzator, telefon\_vanzator, data\_angajare\_vanzator)

VALUES (7, 3, 'Stan', 'Gabriel', '0745789012', TO\_DATE('2018-11-15', 'YYYY-MM-DD'));

INSERT INTO VANZATOR (ID\_VANZATOR, ID\_MAGAZIN, nume\_vanzator, prenume\_vanzator, telefon\_vanzator, data\_angajare\_vanzator)

VALUES (8, 4, 'Enache', 'Cristian', '0745890123', TO\_DATE('2019-07-08', 'YYYY-MM-DD'));

INSERT INTO VANZATOR (ID\_VANZATOR, ID\_MAGAZIN, nume\_vanzator, prenume\_vanzator, telefon\_vanzator, data\_angajare\_vanzator)

VALUES (9, 4, 'Nistor', 'Andreea', '0745901234', TO\_DATE('2021-02-10', 'YYYY-MM-DD'));

INSERT INTO VANZATOR (ID\_VANZATOR, ID\_MAGAZIN, nume\_vanzator, prenume\_vanzator, telefon\_vanzator, data\_angajare\_vanzator)

VALUES (10, 5, 'Lupu', 'Bianca', '0745012345', TO\_DATE('2023-01-12', 'YYYY-MM-DD'));



**-- 3. INSERARE CLIENT**

INSERT INTO CLIENT (ID\_CLIENT, ID\_VANZATOR, nume\_client, prenume\_client) VALUES (1, 1, 'Popescu', 'Ion');

INSERT INTO CLIENT (ID\_CLIENT, ID\_VANZATOR, nume\_client, prenume\_client) VALUES (2, 1, 'Ionescu', 'Maria');

INSERT INTO CLIENT (ID\_CLIENT, ID\_VANZATOR, nume\_client, prenume\_client) VALUES (3, 2, 'Georgescu', 'Ana');

INSERT INTO CLIENT (ID\_CLIENT, ID\_VANZATOR, nume\_client, prenume\_client) VALUES (4, 1, 'Dumitru', 'Mihai');

INSERT INTO CLIENT (ID\_CLIENT, ID\_VANZATOR, nume\_client, prenume\_client) VALUES (5, 2, 'Constantin', 'Elena');

INSERT INTO CLIENT (ID\_CLIENT, ID\_VANZATOR, nume\_client, prenume\_client) VALUES (6, 6, 'Marinescu', 'Adrian');

INSERT INTO CLIENT (ID\_CLIENT, ID\_VANZATOR, nume\_client, prenume\_client) VALUES (7, 7, 'Stan', 'Gabriel');

INSERT INTO CLIENT (ID\_CLIENT, ID\_VANZATOR, nume\_client, prenume\_client) VALUES (8, 4, 'Enache', 'Cristian');

INSERT INTO CLIENT (ID\_CLIENT, ID\_VANZATOR, nume\_client, prenume\_client) VALUES (9, 4, 'Nistor', 'Andreea');

INSERT INTO CLIENT (ID\_CLIENT, ID\_VANZATOR, nume\_client, prenume\_client) VALUES (10, 9, 'Lupu', 'Bianca');

INSERT INTO CLIENT (ID\_CLIENT, ID\_VANZATOR, nume\_client, prenume\_client) VALUES (11, 10, 'Vasilescu', 'Marius');

INSERT INTO CLIENT (ID\_CLIENT, ID\_VANZATOR, nume\_client, prenume\_client) VALUES (12, 3, 'Dobre', 'Ioana');

INSERT INTO CLIENT (ID\_CLIENT, ID\_VANZATOR, nume\_client, prenume\_client) VALUES (13, 5, 'Matei', 'Alexandru');

INSERT INTO CLIENT (ID\_CLIENT, ID\_VANZATOR, nume\_client, prenume\_client) VALUES (14, 10, 'Petrescu', 'Carmen');

INSERT INTO CLIENT (ID\_CLIENT, ID\_VANZATOR, nume\_client, prenume\_client) VALUES (15, 3, 'Dinu', 'Florin');

****

**-- 4. INSERARE CONTRACT\_ANGAJARE**

INSERT INTO CONTRACT\_ANGAJARE (ID\_CONTRACT\_ANGAJARE, ID\_VANZATOR, salariu, tip\_contract)

VALUES (1, 1, 4000.00, 'Full-time');

INSERT INTO CONTRACT\_ANGAJARE (ID\_CONTRACT\_ANGAJARE, ID\_VANZATOR, salariu, tip\_contract)

VALUES (2, 2, 3800.00, 'Part-time');

INSERT INTO CONTRACT\_ANGAJARE (ID\_CONTRACT\_ANGAJARE, ID\_VANZATOR, salariu, tip\_contract)

VALUES (3, 3, 4200.00, 'Full-time');

INSERT INTO CONTRACT\_ANGAJARE (ID\_CONTRACT\_ANGAJARE, ID\_VANZATOR, salariu, tip\_contract)

VALUES (4, 4, 3900.00, 'Full-time');

INSERT INTO CONTRACT\_ANGAJARE (ID\_CONTRACT\_ANGAJARE, ID\_VANZATOR, salariu, tip\_contract)

VALUES (5, 5, 4500.00, 'Full-time');

INSERT INTO CONTRACT\_ANGAJARE (ID\_CONTRACT\_ANGAJARE, ID\_VANZATOR, salariu, tip\_contract)

VALUES (6, 6, 4700.00, 'Part-time');

INSERT INTO CONTRACT\_ANGAJARE (ID\_CONTRACT\_ANGAJARE, ID\_VANZATOR, salariu, tip\_contract)

VALUES (7, 7, 3600.00, 'Full-time');

INSERT INTO CONTRACT\_ANGAJARE (ID\_CONTRACT\_ANGAJARE, ID\_VANZATOR, salariu, tip\_contract)

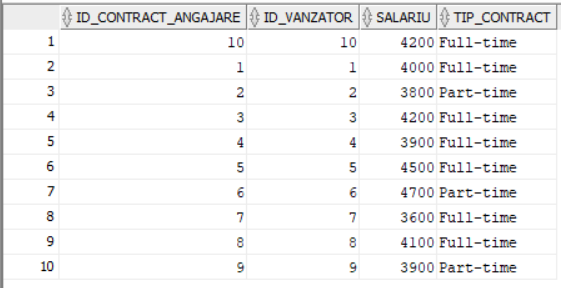
VALUES (8, 8, 4100.00, 'Full-time');

INSERT INTO CONTRACT\_ANGAJARE (ID\_CONTRACT\_ANGAJARE, ID\_VANZATOR, salariu, tip\_contract)

VALUES (9, 9, 3900.00, 'Part-time');

INSERT INTO CONTRACT\_ANGAJARE (ID\_CONTRACT\_ANGAJARE, ID\_VANZATOR, salariu, tip\_contract)

VALUES (10, 10, 4200.00, 'Full-time');



**-- 5. INSERARE TIP\_PLATA**

INSERT INTO TIP\_PLATA (ID\_TIP\_PLATA, nume\_tip\_plata) VALUES (1, 'Cash');

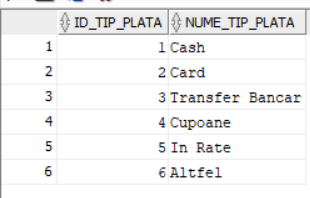
INSERT INTO TIP\_PLATA (ID\_TIP\_PLATA, nume\_tip\_plata) VALUES (2, 'Card');

INSERT INTO TIP\_PLATA (ID\_TIP\_PLATA, nume\_tip\_plata) VALUES (3, 'Transfer Bancar');

INSERT INTO TIP\_PLATA (ID\_TIP\_PLATA, nume\_tip\_plata) VALUES (4, 'Cupoane');

INSERT INTO TIP\_PLATA (ID\_TIP\_PLATA, nume\_tip\_plata) VALUES (5, 'In Rate');

INSERT INTO TIP\_PLATA (ID\_TIP\_PLATA, nume\_tip\_plata) VALUES (6, 'Altfel');



**-- 6. INSERARE TABEL MOD\_LIVRARE**

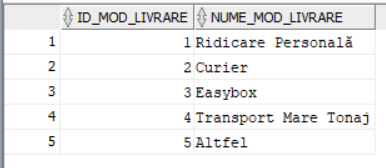
INSERT INTO MOD\_LIVRARE (ID\_MOD\_LIVRARE, nume\_mod\_livrare) VALUES (1, 'Ridicare Personala');

INSERT INTO MOD\_LIVRARE (ID\_MOD\_LIVRARE, nume\_mod\_livrare) VALUES (2, 'Curier');

INSERT INTO MOD\_LIVRARE (ID\_MOD\_LIVRARE, nume\_mod\_livrare) VALUES (3, 'Easybox');

INSERT INTO MOD\_LIVRARE (ID\_MOD\_LIVRARE, nume\_mod\_livrare) VALUES (4, 'Transport Mare Tonaj');

INSERT INTO MOD\_LIVRARE (ID\_MOD\_LIVRARE, nume\_mod\_livrare) VALUES (5, 'Altfel');



**-- 7. INSERARE COMANDA**

INSERT INTO COMANDA (ID\_COMANDA, ID\_CLIENT, ID\_TIP\_PLATA, ID\_MOD\_LIVRARE, data\_comanda, interval\_livrare)

VALUES (1, 1, 1, 2, TO\_DATE('2024-01-10', 'YYYY-MM-DD'), '3 zile');

INSERT INTO COMANDA (ID\_COMANDA, ID\_CLIENT, ID\_TIP\_PLATA, ID\_MOD\_LIVRARE, data\_comanda, interval\_livrare)

VALUES (2, 1, 2, 1, TO\_DATE('2024-02-15', 'YYYY-MM-DD'), '1 zi');

INSERT INTO COMANDA (ID\_COMANDA, ID\_CLIENT, ID\_TIP\_PLATA, ID\_MOD\_LIVRARE, data\_comanda, interval\_livrare)

VALUES (3, 2, 3, 3, TO\_DATE('2024-01-12', 'YYYY-MM-DD'), '2 zile');

INSERT INTO COMANDA (ID\_COMANDA, ID\_CLIENT, ID\_TIP\_PLATA, ID\_MOD\_LIVRARE, data\_comanda, interval\_livrare)

VALUES (4, 2, 4, 4, TO\_DATE('2024-02-20', 'YYYY-MM-DD'), '5 zile');

INSERT INTO COMANDA (ID\_COMANDA, ID\_CLIENT, ID\_TIP\_PLATA, ID\_MOD\_LIVRARE, data\_comanda, interval\_livrare)

VALUES (5, 3, 2, 2, TO\_DATE('2024-01-14', 'YYYY-MM-DD'), 'Livrare rapida');

INSERT INTO COMANDA (ID\_COMANDA, ID\_CLIENT, ID\_TIP\_PLATA, ID\_MOD\_LIVRARE, data\_comanda, interval\_livrare)

VALUES (6, 3, 1, 3, TO\_DATE('2024-02-25', 'YYYY-MM-DD'), 'Easybox');

INSERT INTO COMANDA (ID\_COMANDA, ID\_CLIENT, ID\_TIP\_PLATA, ID\_MOD\_LIVRARE, data\_comanda, interval\_livrare)

VALUES (7, 4, 4, 1, TO\_DATE('2024-01-16', 'YYYY-MM-DD'), 'Ridicare personala');

INSERT INTO COMANDA (ID\_COMANDA, ID\_CLIENT, ID\_TIP\_PLATA, ID\_MOD\_LIVRARE, data\_comanda, interval\_livrare)

VALUES (8, 4, 5, 2, TO\_DATE('2024-03-02', 'YYYY-MM-DD'), '4 zile');

INSERT INTO COMANDA (ID\_COMANDA, ID\_CLIENT, ID\_TIP\_PLATA, ID\_MOD\_LIVRARE, data\_comanda, interval\_livrare)

VALUES (9, 5, 3, 4, TO\_DATE('2024-01-17', 'YYYY-MM-DD'), 'Transport mare tonaj');

INSERT INTO COMANDA (ID\_COMANDA, ID\_CLIENT, ID\_TIP\_PLATA, ID\_MOD\_LIVRARE, data\_comanda, interval\_livrare)

VALUES (10, 5, 1, 5, TO\_DATE('2024-03-05', 'YYYY-MM-DD'), 'Standard 7 zile');

INSERT INTO COMANDA (ID\_COMANDA, ID\_CLIENT, ID\_TIP\_PLATA, ID\_MOD\_LIVRARE, data\_comanda, interval\_livrare)

VALUES (11, 6, 5, 3, TO\_DATE('2024-01-19', 'YYYY-MM-DD'), 'Express');

INSERT INTO COMANDA (ID\_COMANDA, ID\_CLIENT, ID\_TIP\_PLATA, ID\_MOD\_LIVRARE, data\_comanda, interval\_livrare)

VALUES (12, 6, 2, 4, TO\_DATE('2024-03-08', 'YYYY-MM-DD'), 'Livrare in 48h');

INSERT INTO COMANDA (ID\_COMANDA, ID\_CLIENT, ID\_TIP\_PLATA, ID\_MOD\_LIVRARE, data\_comanda, interval\_livrare)

VALUES (13, 7, 3, 1, TO\_DATE('2024-01-22', 'YYYY-MM-DD'), 'Ridicare rapida');

INSERT INTO COMANDA (ID\_COMANDA, ID\_CLIENT, ID\_TIP\_PLATA, ID\_MOD\_LIVRARE, data\_comanda, interval\_livrare)

VALUES (14, 7, 4, 2, TO\_DATE('2024-03-15', 'YYYY-MM-DD'), '3 zile');

INSERT INTO COMANDA (ID\_COMANDA, ID\_CLIENT, ID\_TIP\_PLATA, ID\_MOD\_LIVRARE, data\_comanda, interval\_livrare)

VALUES (15, 8, 1, 3, TO\_DATE('2024-01-25', 'YYYY-MM-DD'), 'Easybox livrare standard');

INSERT INTO COMANDA (ID\_COMANDA, ID\_CLIENT, ID\_TIP\_PLATA, ID\_MOD\_LIVRARE, data\_comanda, interval\_livrare)

VALUES (16, 8, 5, 5, TO\_DATE('2024-03-20', 'YYYY-MM-DD'), '7 zile');

INSERT INTO COMANDA (ID\_COMANDA, ID\_CLIENT, ID\_TIP\_PLATA, ID\_MOD\_LIVRARE, data\_comanda, interval\_livrare)

VALUES (17, 9, 2, 4, TO\_DATE('2024-01-28', 'YYYY-MM-DD'), 'Livrare urgenta');

INSERT INTO COMANDA (ID\_COMANDA, ID\_CLIENT, ID\_TIP\_PLATA, ID\_MOD\_LIVRARE, data\_comanda, interval\_livrare)

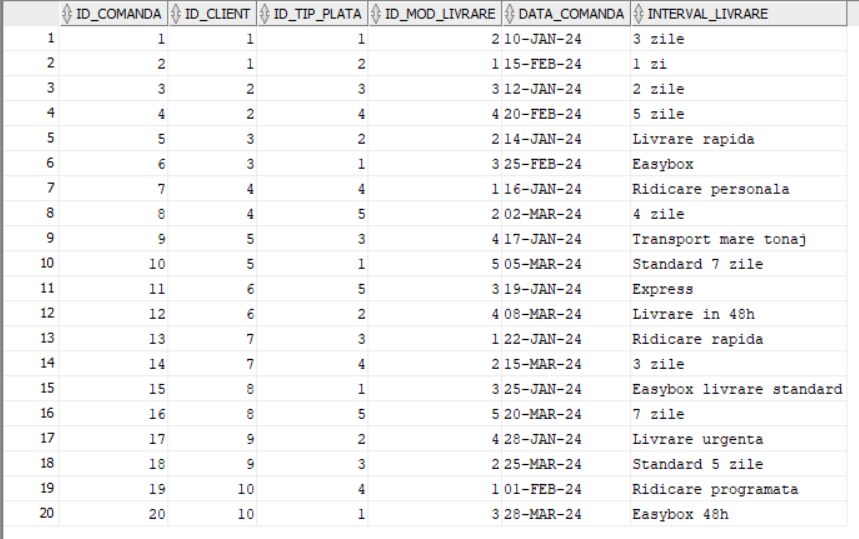
VALUES (18, 9, 3, 2, TO\_DATE('2024-03-25', 'YYYY-MM-DD'), 'Standard 5 zile');

INSERT INTO COMANDA (ID\_COMANDA, ID\_CLIENT, ID\_TIP\_PLATA, ID\_MOD\_LIVRARE, data\_comanda, interval\_livrare)

VALUES (19, 10, 4, 1, TO\_DATE('2024-02-01', 'YYYY-MM-DD'), 'Ridicare programata');

INSERT INTO COMANDA (ID\_COMANDA, ID\_CLIENT, ID\_TIP\_PLATA, ID\_MOD\_LIVRARE, data\_comanda, interval\_livrare)

VALUES (20, 10, 1, 3, TO\_DATE('2024-03-28', 'YYYY-MM-DD'), 'Easybox 48h');



**-- 8. INSERARE DEPOZIT**

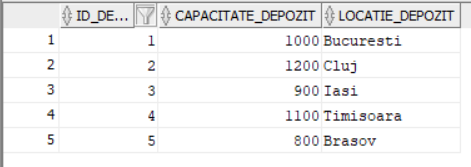
INSERT INTO DEPOZIT (ID\_DEPOZIT, capacitate\_depozit, locatie\_depozit) VALUES (1, 1000, 'Bucuresti');

INSERT INTO DEPOZIT (ID\_DEPOZIT, capacitate\_depozit, locatie\_depozit) VALUES (2, 1200, 'Cluj');

INSERT INTO DEPOZIT (ID\_DEPOZIT, capacitate\_depozit, locatie\_depozit) VALUES (3, 900, 'Iasi');

INSERT INTO DEPOZIT (ID\_DEPOZIT, capacitate\_depozit, locatie\_depozit) VALUES (4, 1100, 'Timisoara');

INSERT INTO DEPOZIT (ID\_DEPOZIT, capacitate\_depozit, locatie\_depozit) VALUES (5, 800, 'Brasov');



**-- 9. INSERARE TABEL CATEGORII\_PRODUSE**

INSERT INTO CATEGORII\_PRODUSE (ID\_CATEGORIE\_PRODUS, denumire\_categorie, descriere\_categorie)

VALUES (1, 'Ciment si Adezivi',

'Cimentul si adezivii sunt elemente esentiale in orice proiect de constructie.

Gama include ciment Portland, ciment alb, mortar de zidarie, adezivi pentru gresie si faianta,

precum si solutii speciale pentru hidroizolatii si termoizolatii. Aceste produse asigura o rezistenta

sporita si o aderenta perfecta intre materiale.');

INSERT INTO CATEGORII\_PRODUSE (ID\_CATEGORIE\_PRODUS, denumire\_categorie, descriere\_categorie)

VALUES (2, 'Caramizi si BCA',

'Caramizile si blocurile BCA sunt utilizate in constructia peretilor exteriori si interiori,

asigurand izolatie termica si fonica superioara. Caramizile ceramice sunt ideale pentru constructii rezistente,

in timp ce blocurile BCA sunt usoare, durabile si eficiente energetic. Gama include si caramizi aparente pentru decor.');

INSERT INTO CATEGORII\_PRODUSE (ID\_CATEGORIE\_PRODUS, denumire\_categorie, descriere\_categorie)

VALUES (3, 'Lemn si Cherestea',

'Materialele din lemn sunt esentiale pentru structuri, acoperisuri si finisaje.

Oferta include cherestea de rasinoase, scanduri, grinzi si paneluri lemnoase de diverse dimensiuni.

Produsele sunt tratate pentru a rezista la umiditate, insecte si fungi, fiind perfecte pentru constructii durabile.');

INSERT INTO CATEGORII\_PRODUSE (ID\_CATEGORIE\_PRODUS, denumire\_categorie, descriere\_categorie)

VALUES (4, 'Fier si Armaturi',

'Fierul si armaturile sunt componente esentiale in structurile din beton armat.

Gama include bare de otel beton, plase sudate, profile metalice si accesorii pentru sustinerea structurilor.

Aceste materiale ofera rezistenta si stabilitate constructiilor, fiind indispensabile in proiectele civile si industriale.');

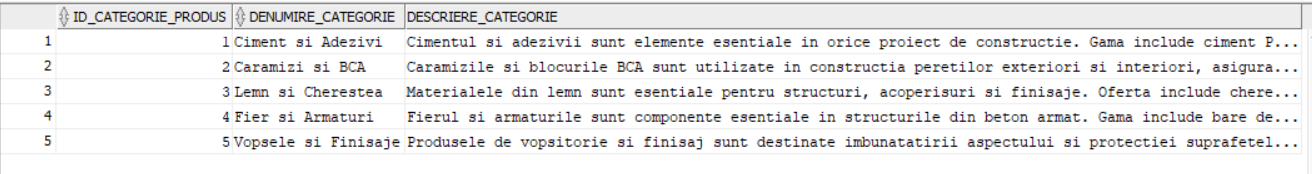
INSERT INTO CATEGORII\_PRODUSE (ID\_CATEGORIE\_PRODUS, denumire\_categorie, descriere\_categorie)

VALUES (5, 'Vopsele si Finisaje',

'Produsele de vopsitorie si finisaj sunt destinate imbunatatirii aspectului si protectiei suprafetelor.

Gama include vopsele lavabile, emailuri pentru metal si lemn, grunduri, tencuieli decorative si gleturi pentru pereti.

Aceste produse asigura atat o estetica placuta, cat si o durabilitate crescuta.');



**-- 10. INSERARE TABEL PRODUSE**

INSERT INTO PRODUSE (ID\_PRODUS, ID\_CATEGORIE\_PRODUS, ID\_DEPOZIT, nume\_produs, descriere\_produs, pret\_produs, stoc\_produs)

VALUES (1, 1, 1, 'Ciment Portland 50kg', 'Ciment de calitate superioara pentru constructii civile si industriale.', 45.00, 500);

INSERT INTO PRODUSE (ID\_PRODUS, ID\_CATEGORIE\_PRODUS, ID\_DEPOZIT, nume\_produs, descriere\_produs, pret\_produs, stoc\_produs)

VALUES (2, 1, 1, 'Adeziv pentru gresie si faianta 25kg', 'Adeziv flexibil, rezistent la umiditate si temperaturi extreme.', 35.00, 300);

INSERT INTO PRODUSE (ID\_PRODUS, ID\_CATEGORIE\_PRODUS, ID\_DEPOZIT, nume\_produs, descriere\_produs, pret\_produs, stoc\_produs)

VALUES (3, 2, 2, 'Caramida Porotherm 25cm', 'Caramida termoizolanta pentru pereti exteriori si interiori.', 3.50, 5000);

INSERT INTO PRODUSE (ID\_PRODUS, ID\_CATEGORIE\_PRODUS, ID\_DEPOZIT, nume\_produs, descriere\_produs, pret\_produs, stoc\_produs)

VALUES (4, 2, 2, 'BCA 60x25x20cm', 'Bloc de zidarie usor, termoizolant, ideal pentru constructii rapide.', 6.00, 4000);

INSERT INTO PRODUSE (ID\_PRODUS, ID\_CATEGORIE\_PRODUS, ID\_DEPOZIT, nume\_produs, descriere\_produs, pret\_produs, stoc\_produs)

VALUES (5, 3, 3, 'Cherestea rasinoasa 4m', 'Cherestea uscata, tratata impotriva umiditatii si daunatorilor.', 25.00, 1000);

INSERT INTO PRODUSE (ID\_PRODUS, ID\_CATEGORIE\_PRODUS, ID\_DEPOZIT, nume\_produs, descriere\_produs, pret\_produs, stoc\_produs)

VALUES (6, 3, 3, 'Grinzi din lemn stratificat 6m', 'Lemn uscat, rezistent la incovoiere, ideal pentru structuri grele.', 120.00, 500);

INSERT INTO PRODUSE (ID\_PRODUS, ID\_CATEGORIE\_PRODUS, ID\_DEPOZIT, nume\_produs, descriere\_produs, pret\_produs, stoc\_produs)

VALUES (7, 4, 4, 'Plasa sudata 4mm 2x3m', 'Plasa din otel beton, ideala pentru armaturi si structuri de rezistenta.', 120.00, 300);

INSERT INTO PRODUSE (ID\_PRODUS, ID\_CATEGORIE\_PRODUS, ID\_DEPOZIT, nume\_produs, descriere\_produs, pret\_produs, stoc\_produs)

VALUES (8, 4, 4, 'Bare otel beton 12mm 6m', 'Bare din otel de calitate superioara pentru armaturi beton.', 75.00, 1000);

INSERT INTO PRODUSE (ID\_PRODUS, ID\_CATEGORIE\_PRODUS, ID\_DEPOZIT, nume\_produs, descriere\_produs, pret\_produs, stoc\_produs)

VALUES (9, 5, 5, 'Vopsea lavabila alba 10L', 'Vopsea superlavabila, rezistenta la murdarie, ideala pentru interior.', 80.00, 200);

INSERT INTO PRODUSE (ID\_PRODUS, ID\_CATEGORIE\_PRODUS, ID\_DEPOZIT, nume\_produs, descriere\_produs, pret\_produs, stoc\_produs)

VALUES (10, 5, 5, 'Tencuiala decorativa siliconata 25kg', 'Tencuiala elastica, rezistenta la umiditate si murdarie.', 110.00, 150);

INSERT INTO PRODUSE (ID\_PRODUS, ID\_CATEGORIE\_PRODUS, ID\_DEPOZIT, nume\_produs, descriere\_produs, pret\_produs, stoc\_produs)

VALUES (11, 1, 1, 'Glet de finisaj 20kg', 'Glet fin, aplicabil mecanizat sau manual pentru pereti perfect netezi.', 45.00, 350);

INSERT INTO PRODUSE (ID\_PRODUS, ID\_CATEGORIE\_PRODUS, ID\_DEPOZIT, nume\_produs, descriere\_produs, pret\_produs, stoc\_produs)

VALUES (12, 2, 2, 'Boltari beton 39x19x14cm', 'Boltari din beton pentru zidarie, cu rezistenta ridicata la compresiune.', 7.50, 2500);

INSERT INTO PRODUSE (ID\_PRODUS, ID\_CATEGORIE\_PRODUS, ID\_DEPOZIT, nume\_produs, descriere\_produs, pret\_produs, stoc\_produs)

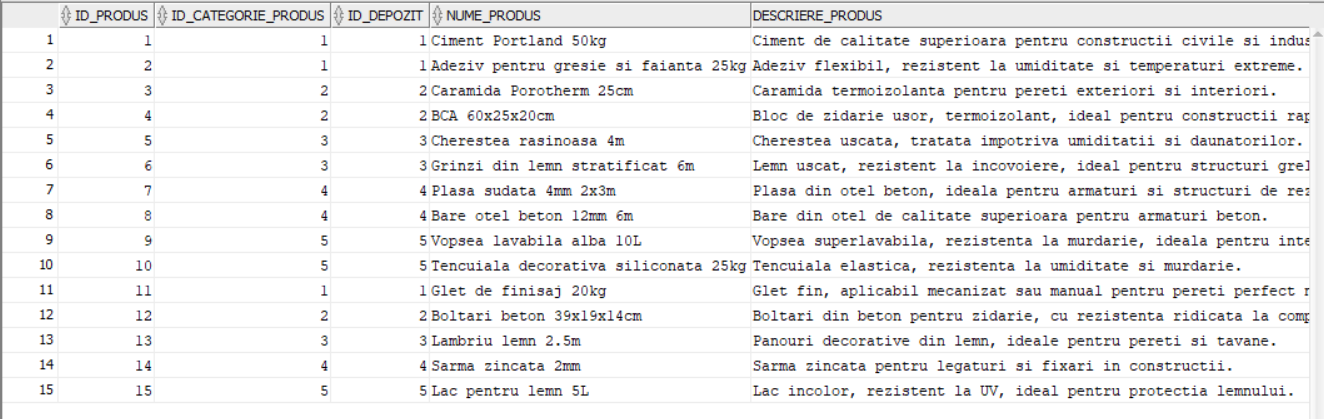
VALUES (13, 3, 3, 'Lambriu lemn 2.5m', 'Panouri decorative din lemn, ideale pentru pereti si tavane.', 30.00, 800);

INSERT INTO PRODUSE (ID\_PRODUS, ID\_CATEGORIE\_PRODUS, ID\_DEPOZIT, nume\_produs, descriere\_produs, pret\_produs, stoc\_produs)

VALUES (14, 4, 4, 'Sarma zincata 2mm', 'Sarma zincata pentru legaturi si fixari in constructii.', 10.00, 500);

INSERT INTO PRODUSE (ID\_PRODUS, ID\_CATEGORIE\_PRODUS, ID\_DEPOZIT, nume\_produs, descriere\_produs, pret\_produs, stoc\_produs)

VALUES (15, 5, 5, 'Lac pentru lemn 5L', 'Lac incolor, rezistent la UV, ideal pentru protectia lemnului.', 95.00, 250);



**-- 11. INSERARE TABEL DETALII\_PRODUSE**

INSERT INTO DETALII\_PRODUSE (ID\_DETALII\_PRODUSE, material, dimensiuni, greutate)

VALUES (1, 'Ciment', '50kg sac', 50.00);

INSERT INTO DETALII\_PRODUSE (ID\_DETALII\_PRODUSE, material, dimensiuni, greutate)

VALUES (2, 'Adeziv', '25kg sac', 25.00);

INSERT INTO DETALII\_PRODUSE (ID\_DETALII\_PRODUSE, material, dimensiuni, greutate)

VALUES (3, 'Caramida', '25x12x14cm', 3.50);

INSERT INTO DETALII\_PRODUSE (ID\_DETALII\_PRODUSE, material, dimensiuni, greutate)

VALUES (4, 'BCA', '60x25x20cm', 6.00);

INSERT INTO DETALII\_PRODUSE (ID\_DETALII\_PRODUSE, material, dimensiuni, greutate)

VALUES (5, 'Cherestea', '4m lungime', 20.00);

INSERT INTO DETALII\_PRODUSE (ID\_DETALII\_PRODUSE, material, dimensiuni, greutate)

VALUES (6, 'Fier beton', '6m lungime, 12mm grosime', 50.00);

INSERT INTO DETALII\_PRODUSE (ID\_DETALII\_PRODUSE, material, dimensiuni, greutate)

VALUES (7, 'Plasa sudata', '2x3m, 4mm grosime', 10.00);

INSERT INTO DETALII\_PRODUSE (ID\_DETALII\_PRODUSE, material, dimensiuni, greutate)

VALUES (8, 'Vopsea lavabila', '10L galeata', 10.00);

INSERT INTO DETALII\_PRODUSE (ID\_DETALII\_PRODUSE, material, dimensiuni, greutate)

VALUES (9, 'Lac lemn', '5L recipient', 5.00);

INSERT INTO DETALII\_PRODUSE (ID\_DETALII\_PRODUSE, material, dimensiuni, greutate)

VALUES (10, 'Tencuiala decorativa', '25kg sac', 25.00);



**-- 12. INSERARE TABEL PRODUSE\_AU\_DETALII**

INSERT INTO PRODUSE\_AU\_DETALII (ID\_PRODUS, ID\_DETALII\_PRODUSE) VALUES (1, 1);

INSERT INTO PRODUSE\_AU\_DETALII (ID\_PRODUS, ID\_DETALII\_PRODUSE) VALUES (2, 2);

INSERT INTO PRODUSE\_AU\_DETALII (ID\_PRODUS, ID\_DETALII\_PRODUSE) VALUES (3, 3);

INSERT INTO PRODUSE\_AU\_DETALII (ID\_PRODUS, ID\_DETALII\_PRODUSE) VALUES (4, 4);

INSERT INTO PRODUSE\_AU\_DETALII (ID\_PRODUS, ID\_DETALII\_PRODUSE) VALUES (5, 5);

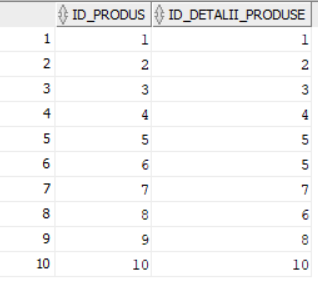
INSERT INTO PRODUSE\_AU\_DETALII (ID\_PRODUS, ID\_DETALII\_PRODUSE) VALUES (6, 5);

INSERT INTO PRODUSE\_AU\_DETALII (ID\_PRODUS, ID\_DETALII\_PRODUSE) VALUES (7, 7);

INSERT INTO PRODUSE\_AU\_DETALII (ID\_PRODUS, ID\_DETALII\_PRODUSE) VALUES (8, 6);

INSERT INTO PRODUSE\_AU\_DETALII (ID\_PRODUS, ID\_DETALII\_PRODUSE) VALUES (9, 8);

INSERT INTO PRODUSE\_AU\_DETALII (ID\_PRODUS, ID\_DETALII\_PRODUSE) VALUES (10, 10);



**-- 13. INSERARE TABEL PROMOTII**

INSERT INTO PROMOTII (ID\_PROMOTII, nume\_promotie, descriere\_promotie)

VALUES (1, 'Reducere Ciment', 'Reducere de 10% la toate tipurile de ciment pentru comenzi peste 500 RON.');

INSERT INTO PROMOTII (ID\_PROMOTII, nume\_promotie, descriere\_promotie)

VALUES (2, 'Oferta Caramida', 'La fiecare 1000 de caramizi Porotherm cumparate, primesti 50 gratuit.');

INSERT INTO PROMOTII (ID\_PROMOTII, nume\_promotie, descriere\_promotie)

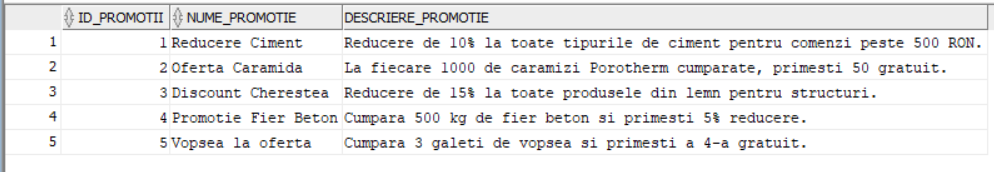
VALUES (3, 'Discount Cherestea', 'Reducere de 15% la toate produsele din lemn pentru structuri.');

INSERT INTO PROMOTII (ID\_PROMOTII, nume\_promotie, descriere\_promotie)

VALUES (4, 'Promotie Fier Beton', 'Cumpara 500 kg de fier beton si primesti 5% reducere.');

INSERT INTO PROMOTII (ID\_PROMOTII, nume\_promotie, descriere\_promotie)

VALUES (5, 'Vopsea la oferta', 'Cumpara 3 galeti de vopsea si primesti a 4-a gratuit.');



**-- 14. INSERARE TABEL DETALII\_PROMOTII**

INSERT INTO DETALII\_PROMOTII (ID\_PRODUS, ID\_PROMOTIE, data\_inceput\_promotie, data\_sfarsit\_promotie)

VALUES (1, 1, TO\_DATE('2024-04-01', 'YYYY-MM-DD'), TO\_DATE('2024-04-30', 'YYYY-MM-DD'));

INSERT INTO DETALII\_PROMOTII (ID\_PRODUS, ID\_PROMOTIE, data\_inceput\_promotie, data\_sfarsit\_promotie)

VALUES (3, 2, TO\_DATE('2024-04-01', 'YYYY-MM-DD'), TO\_DATE('2024-04-30', 'YYYY-MM-DD'));

INSERT INTO DETALII\_PROMOTII (ID\_PRODUS, ID\_PROMOTIE, data\_inceput\_promotie, data\_sfarsit\_promotie)

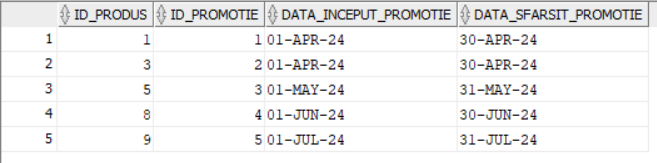
VALUES (5, 3, TO\_DATE('2024-05-01', 'YYYY-MM-DD'), TO\_DATE('2024-05-31', 'YYYY-MM-DD'));

INSERT INTO DETALII\_PROMOTII (ID\_PRODUS, ID\_PROMOTIE, data\_inceput\_promotie, data\_sfarsit\_promotie)

VALUES (8, 4, TO\_DATE('2024-06-01', 'YYYY-MM-DD'), TO\_DATE('2024-06-30', 'YYYY-MM-DD'));

INSERT INTO DETALII\_PROMOTII (ID\_PRODUS, ID\_PROMOTIE, data\_inceput\_promotie, data\_sfarsit\_promotie)

VALUES (9, 5, TO\_DATE('2024-07-01', 'YYYY-MM-DD'), TO\_DATE('2024-07-31', 'YYYY-MM-DD'));



**-- 15. INSERARE TABEL COMANDA\_CONTINE\_PRODUSE**

-- Comanda 1

INSERT INTO COMANDA\_CONTINE\_PRODUSE (ID\_COMANDA, ID\_PRODUS, cantitate, pret\_comanda)

VALUES (1, 1, 10, 10 \* 45.00); -- 450.00

INSERT INTO COMANDA\_CONTINE\_PRODUSE (ID\_COMANDA, ID\_PRODUS, cantitate, pret\_comanda)

VALUES (1, 3, 500, 500 \* 3.50); -- 1750.00

-- Comanda 2

INSERT INTO COMANDA\_CONTINE\_PRODUSE (ID\_COMANDA, ID\_PRODUS, cantitate, pret\_comanda)

VALUES (2, 5, 100, 100 \* 25.00); -- 2500.00

INSERT INTO COMANDA\_CONTINE\_PRODUSE (ID\_COMANDA, ID\_PRODUS, cantitate, pret\_comanda)

VALUES (2, 9, 50, 50 \* 80.00); -- 4000.00

-- Comanda 3

INSERT INTO COMANDA\_CONTINE\_PRODUSE (ID\_COMANDA, ID\_PRODUS, cantitate, pret\_comanda)

VALUES (3, 8, 200, 200 \* 75.00); -- 15000.00

INSERT INTO COMANDA\_CONTINE\_PRODUSE (ID\_COMANDA, ID\_PRODUS, cantitate, pret\_comanda)

VALUES (3, 7, 30, 30 \* 120.00); -- 3600.00

-- Comanda 4

INSERT INTO COMANDA\_CONTINE\_PRODUSE (ID\_COMANDA, ID\_PRODUS, cantitate, pret\_comanda)

VALUES (4, 10, 50, 50 \* 110.00); -- 5500.00

INSERT INTO COMANDA\_CONTINE\_PRODUSE (ID\_COMANDA, ID\_PRODUS, cantitate, pret\_comanda)

VALUES (4, 4, 200, 200 \* 6.00); -- 1200.00

-- Comanda 5

INSERT INTO COMANDA\_CONTINE\_PRODUSE (ID\_COMANDA, ID\_PRODUS, cantitate, pret\_comanda)

VALUES (5, 6, 20, 20 \* 120.00); -- 2400.00

INSERT INTO COMANDA\_CONTINE\_PRODUSE (ID\_COMANDA, ID\_PRODUS, cantitate, pret\_comanda)

VALUES (5, 2, 40, 40 \* 35.00); -- 1400.00

-- Comanda 6

INSERT INTO COMANDA\_CONTINE\_PRODUSE (ID\_COMANDA, ID\_PRODUS, cantitate, pret\_comanda)

VALUES (6, 2, 15, 15 \* 35.00); -- 525.00

INSERT INTO COMANDA\_CONTINE\_PRODUSE (ID\_COMANDA, ID\_PRODUS, cantitate, pret\_comanda)

VALUES (6, 12, 30, 30 \* 7.50); -- 225.00

-- Comanda 7

INSERT INTO COMANDA\_CONTINE\_PRODUSE (ID\_COMANDA, ID\_PRODUS, cantitate, pret\_comanda)

VALUES (7, 7, 40, 40 \* 120.00); -- 4800.00

INSERT INTO COMANDA\_CONTINE\_PRODUSE (ID\_COMANDA, ID\_PRODUS, cantitate, pret\_comanda)

VALUES (7, 11, 25, 25 \* 45.00); -- 1125.00

-- Comanda 8

INSERT INTO COMANDA\_CONTINE\_PRODUSE (ID\_COMANDA, ID\_PRODUS, cantitate, pret\_comanda)

VALUES (8, 9, 25, 25 \* 80.00); -- 2000.00

INSERT INTO COMANDA\_CONTINE\_PRODUSE (ID\_COMANDA, ID\_PRODUS, cantitate, pret\_comanda)

VALUES (8, 5, 10, 10 \* 25.00); -- 250.00

-- Comanda 9

INSERT INTO COMANDA\_CONTINE\_PRODUSE (ID\_COMANDA, ID\_PRODUS, cantitate, pret\_comanda)

VALUES (9, 4, 300, 300 \* 6.00); -- 1800.00

INSERT INTO COMANDA\_CONTINE\_PRODUSE (ID\_COMANDA, ID\_PRODUS, cantitate, pret\_comanda)

VALUES (9, 14, 100, 100 \* 10.00); -- 1000.00

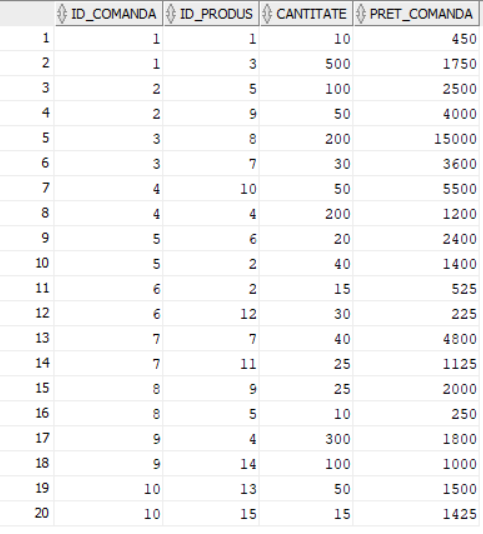
-- Comanda 10

INSERT INTO COMANDA\_CONTINE\_PRODUSE (ID\_COMANDA, ID\_PRODUS, cantitate, pret\_comanda)

VALUES (10, 13, 50, 50 \* 30.00); -- 1500.00

INSERT INTO COMANDA\_CONTINE\_PRODUSE (ID\_COMANDA, ID\_PRODUS, cantitate, pret\_comanda)

VALUES (10, 15, 15, 15 \* 95.00); -- 1425.00



**-- 16. INSERARE DEPARTAMENT**

INSERT INTO DEPARTAMENT (ID\_DEPARTAMENT, denumire\_departament, status)

VALUES (1, 'Vanzari', 'Active');

INSERT INTO DEPARTAMENT (ID\_DEPARTAMENT, denumire\_departament, status)

VALUES (2, 'Marketing', 'Active');

INSERT INTO DEPARTAMENT (ID\_DEPARTAMENT, denumire\_departament, status)

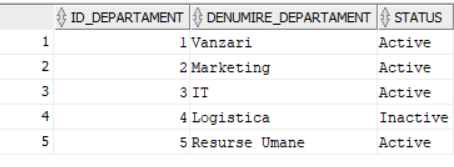
VALUES (3, 'IT', 'Active');

INSERT INTO DEPARTAMENT (ID\_DEPARTAMENT, denumire\_departament, status)

VALUES (4, 'Logistica', 'Inactive');

INSERT INTO DEPARTAMENT (ID\_DEPARTAMENT, denumire\_departament, status)

VALUES (5, 'Resurse Umane', 'Active');



***Cerinta 6***

Un magazin de materiale de constructii gestioneaza stocurile produselor sale. Din cauza unei erori de aprovizionare, toate produsele care au un stoc mai mic de 500 de unitati trebuie suplimentate pana la acest prag. Se doreste identificarea acestor produse si calcularea cantitatilor necesare pentru completarea stocului.

De asemenea, clientii care au comandat aceste produse in ultimul an trebuie informati in legatura cu aceasta situatie. Pentru a retine aceste date, se utilizeaza un nested table pentru lista clientilor afectati.

Doi dintre acesti clienti, alesi aleatoriu, vor primi un voucher de reducere de 20% la urmatoarea comanda. Lista clientilor care vor primi acest beneficiu va fi gestionata folosind un varray.

Pentru rezolvarea problemei, se vor utiliza urmatoarele structuri de date:

1. **Associative Array (Index-By Table)** – pentru memorarea produselor cu stoc insuficient si a cantitatii necesare pentru suplimentare.
2. **Nested Table** – pentru retinerea clientilor afectati, adica acei clienti care au comandat aceste produse in ultimul an.
3. **VARRAY** – pentru selectarea a doi clienti, alesi aleatoriu, care vor primi un **voucher de reducere de 20%** la urmatoarea comanda.

CREATE OR REPLACE PROCEDURE gestionare\_comenzi\_afectate IS

TYPE t\_stoc\_insuficient IS TABLE OF NUMBER INDEX BY PLS\_INTEGER;

produse\_problema t\_stoc\_insuficient;

TYPE t\_client IS RECORD (

id\_client CLIENT.ID\_CLIENT%TYPE,

nume CLIENT.nume\_client%TYPE,

prenume CLIENT.prenume\_client%TYPE

);

TYPE t\_lista\_clienti IS TABLE OF t\_client;

clienti\_afectati t\_lista\_clienti := t\_lista\_clienti();

TYPE t\_vouchere IS VARRAY(2) OF CLIENT.ID\_CLIENT%TYPE;

clienti\_voucher t\_vouchere := t\_vouchere();

v\_idx PLS\_INTEGER;

v\_random1 PLS\_INTEGER;

v\_random2 PLS\_INTEGER;

v\_count\_produse NUMBER := 0;

v\_count\_clienti NUMBER := 0;

PRAG\_MINIM CONSTANT NUMBER := 500;

PERIOADA CONSTANT NUMBER := 365;

v\_output CLOB; -- Variabila pentru stocarea mesajelor

BEGIN

DBMS\_OUTPUT.ENABLE;

v\_output := 'Inceperea executiei procedurii' || CHR(10);

-- Identificam produsele cu stoc insuficient

FOR produs IN (SELECT ID\_PRODUS, stoc\_produs FROM PRODUSE WHERE stoc\_produs < PRAG\_MINIM) LOOP

produse\_problema(produs.ID\_PRODUS) := PRAG\_MINIM - produs.stoc\_produs;

v\_count\_produse := v\_count\_produse + 1;

END LOOP;

IF v\_count\_produse = 0 THEN

v\_output := v\_output || 'Nu exista produse cu stoc insuficient' || CHR(10);

DBMS\_OUTPUT.PUT\_LINE(v\_output);

RETURN;

END IF;

-- Identificam clientii afectati

FOR comanda IN (

SELECT DISTINCT C.ID\_CLIENT, C.nume\_client, C.prenume\_client

FROM CLIENT C

JOIN COMANDA CO ON C.ID\_CLIENT = CO.ID\_CLIENT

JOIN COMANDA\_CONTINE\_PRODUSE CCP ON CO.ID\_COMANDA = CCP.ID\_COMANDA

WHERE CCP.ID\_PRODUS IN (SELECT ID\_PRODUS FROM PRODUSE WHERE stoc\_produs < PRAG\_MINIM)

AND CO.data\_comanda >= SYSDATE - PERIOADA

) LOOP

clienti\_afectati.EXTEND;

clienti\_afectati(clienti\_afectati.LAST) := t\_client(comanda.ID\_CLIENT, comanda.nume\_client, comanda.prenume\_client);

v\_count\_clienti := v\_count\_clienti + 1;

END LOOP;

IF v\_count\_clienti = 0 THEN

v\_output := v\_output || 'Nu exista clienti afectati' || CHR(10);

DBMS\_OUTPUT.PUT\_LINE(v\_output);

RETURN;

END IF;

-- Alegem aleatoriu doi clienti pentru vouchere

IF clienti\_afectati.COUNT >= 2 THEN

LOOP

v\_random1 := TRUNC(DBMS\_RANDOM.VALUE(1, clienti\_afectati.COUNT + 1));

v\_random2 := TRUNC(DBMS\_RANDOM.VALUE(1, clienti\_afectati.COUNT + 1));

EXIT WHEN v\_random2 <> v\_random1;

END LOOP;

clienti\_voucher := t\_vouchere(clienti\_afectati(v\_random1).id\_client, clienti\_afectati(v\_random2).id\_client);

END IF;

-- Adaugare mesaje in variabila CLOB

v\_output := v\_output || 'Produse cu stoc insuficient (sub ' || PRAG\_MINIM || ' unitati):' || CHR(10);

v\_idx := produse\_problema.FIRST;

WHILE v\_idx IS NOT NULL LOOP

v\_output := v\_output || 'Produs ID ' || v\_idx || ' necesita suplimentare cu ' || produse\_problema(v\_idx) || ' unitati' || CHR(10);

v\_idx := produse\_problema.NEXT(v\_idx);

END LOOP;

v\_output := v\_output || 'Lista clientilor afectati:' || CHR(10);

FOR i IN 1..clienti\_afectati.COUNT LOOP

v\_output := v\_output || clienti\_afectati(i).nume || ' ' || clienti\_afectati(i).prenume || CHR(10);

END LOOP;

IF clienti\_voucher.COUNT = 2 THEN

v\_output := v\_output || 'Voucher 20% reducere acordat clientilor: ' || clienti\_voucher(1) || ' si ' || clienti\_voucher(2) || CHR(10);

ELSE

v\_output := v\_output || 'Nu sunt suficienti clienti afectati pentru a primi vouchere' || CHR(10);

END IF;

v\_output := v\_output || 'Executia procedurii s-a incheiat' || CHR(10);

-- Afisam rezultatele doar in DBMS Output

DBMS\_OUTPUT.PUT\_LINE(v\_output);

END gestionare\_comenzi\_afectate;

/

SET SERVEROUTPUT ON;

BEGIN

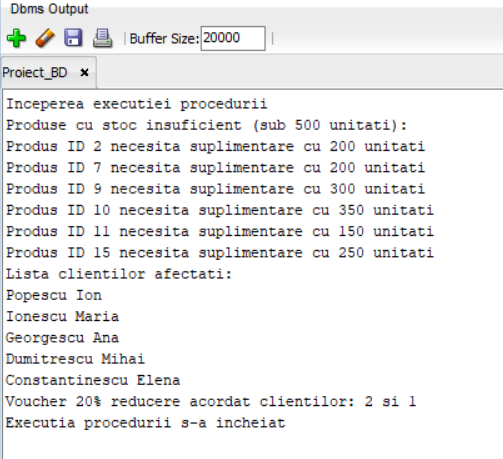
gestionare\_comenzi\_afectate;

END;

/







***Cerinta 7***

Problema:

Proprietarul magazinului de materiale de constructii doreste sa analizeze comenzile care au o valoare mai mare de 2000 de lei, pentru a identifica produsele care genereaza cele mai mari vanzari. Astfel, se va realiza un raport care sa contina lista comenzilor ce depasesc aceasta suma si, pentru fiecare comanda, detaliile produselor achizitionate: numele produsului, cantitatea comandata si valoarea totala a acestuia in cadrul comenzii.

Pentru a rezolva aceasta problema, se va folosi un **subprogram stocat independent**, care utilizeaza **doua cursoare**:

* **Un cursor principal** care va selecta comenzile cu o valoare totala de peste 2000 de lei.
* **Un cursor parametrizat** care va prelua produsele din fiecare comanda identificata de primul cursor.

La executia procedurii, se vor afisa toate comenzile care respecta criteriul stabilit, impreuna cu produsele aferente fiecarei comenzi.

CREATE OR REPLACE PROCEDURE raport\_comenzi\_mari IS

-- Cursor 1: NU are parametri si selecteaza comenzile care depasesc 2000 lei.

-- Acest cursor returneaza ID-ul comenzii si valoarea totala a comenzii.

-- Practic, acest cursor identifica comenzile relevante pentru raport.

CURSOR comenzi\_mari IS

SELECT ID\_COMANDA, SUM(PRET\_COMANDA) AS VALOARE\_TOTALA

FROM COMANDA\_CONTINE\_PRODUSE

GROUP BY ID\_COMANDA

HAVING SUM(PRET\_COMANDA) > 2000;

-- Cursor 2: ESTE parametrizat si depinde de primul cursor.

-- Primeste un ID\_COMANDA ca parametru si returneaza toate produsele din acea comanda.

-- Astfel, pentru fiecare comanda identificata de primul cursor, al doilea cursor extrage produsele aferente.

CURSOR produse\_comanda(v\_id\_comanda NUMBER) IS

SELECT P.NUME\_PRODUS, CCP.CANTITATE, CCP.PRET\_COMANDA

FROM COMANDA\_CONTINE\_PRODUSE CCP

JOIN PRODUSE P ON CCP.ID\_PRODUS = P.ID\_PRODUS

WHERE CCP.ID\_COMANDA = v\_id\_comanda;

-- Variabile pentru a stoca datele produselor in timpul preluarii din cursorul parametrizat.

v\_nume\_produs PRODUSE.NUME\_PRODUS%TYPE;

v\_cantitate COMANDA\_CONTINE\_PRODUSE.CANTITATE%TYPE;

v\_pret\_total COMANDA\_CONTINE\_PRODUSE.PRET\_COMANDA%TYPE;

BEGIN

-- Parcurgem comenzile mai mari de 2000 de lei (folosind primul cursor)

FOR comanda IN comenzi\_mari LOOP

-- Afisam detalii despre comanda curenta

DBMS\_OUTPUT.PUT\_LINE('Comanda ID ' || comanda.ID\_COMANDA ||

' - Valoare totala: ' || comanda.VALOARE\_TOTALA || ' lei');

-- Parcurgem produsele asociate comenzii curente (folosind al doilea cursor, care primeste ID\_COMANDA)

OPEN produse\_comanda(comanda.ID\_COMANDA);

LOOP

FETCH produse\_comanda INTO v\_nume\_produs, v\_cantitate, v\_pret\_total;

EXIT WHEN produse\_comanda%NOTFOUND; -- Oprim bucla cand nu mai sunt produse

-- Afisam detalii despre fiecare produs din aceasta comanda

DBMS\_OUTPUT.PUT\_LINE(' - Produs: ' || v\_nume\_produs ||

', Cantitate: ' || v\_cantitate ||

', Pret total: ' || v\_pret\_total || ' lei');

END LOOP;

CLOSE produse\_comanda;

-- Separator vizual intre comenzi

DBMS\_OUTPUT.PUT\_LINE('--------------------------------------------------');

END LOOP;

END raport\_comenzi\_mari;

/

BEGIN

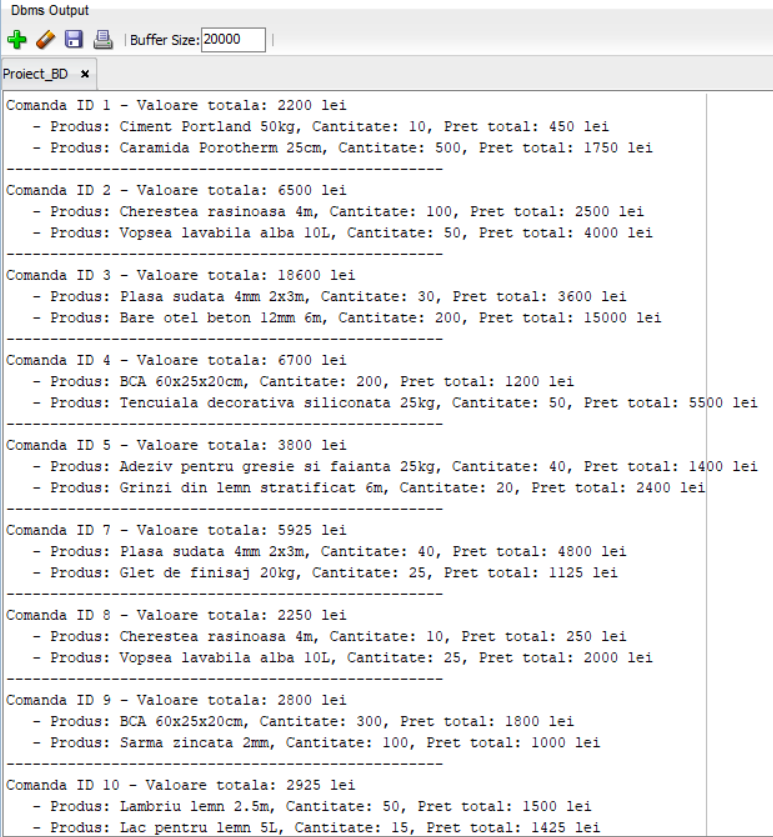
raport\_comenzi\_mari;

END;

/







***Cerinta 8***

Problema:

Managerul unui magazin doreste sa obtina o statistica despre vanzatorii din magazin, in care sa poata afla cati clienti activi au avut in ultimul an. Functia va primi numele si prenumele unui vanzator ca parametru si va verifica cati clienti au plasat comenzi pentru acest vanzator in ultimele 12 luni. Functia va folosi o comanda SQL pentru a interoga trei tabele: **VANZATOR**, **COMANDA** si **CLIENT**.

* Functia va returna numarul de clienti activi pentru un vanzator intr-o perioada de 12 luni. Daca vanzatorul nu are clienti in aceasta perioada, functia va returna un mesaj corespunzator fara a ridica o exceptie.
* In cazul in care vanzatorul nu exista in sistem, va fi ridicata o exceptie care va transmite un mesaj clar.
* Daca exista mai multi vanzatori cu acelasi nume, va fi ridicata o exceptie care sa precizeze acest lucru.
* Functia va trata toate exceptiile posibile folosind exceptiile predefinite NO\_DATA\_FOUND si TOO\_MANY\_ROWS, dar si altele pentru cazuri neprevazute.

CREATE OR REPLACE FUNCTION CLIENTI\_VANZATOR\_AN (

p\_nume\_vanzator IN VARCHAR2

) RETURN VARCHAR2 IS

v\_id\_vanzator VANZATOR.ID\_VANZATOR%TYPE;

v\_rezultat VARCHAR2(4000);

v\_numar\_clienti INT; -- Variabila pentru a numara clientii

BEGIN

-- Cautam ID-ul vanzatorului pe baza numelui complet

DBMS\_OUTPUT.PUT\_LINE('Cautam vanzatorul: ' || p\_nume\_vanzator); -- Debugging

SELECT ID\_VANZATOR

INTO v\_id\_vanzator

FROM VANZATOR

WHERE nume\_vanzator || ' ' || prenume\_vanzator = p\_nume\_vanzator;

DBMS\_OUTPUT.PUT\_LINE('ID Vanzator gasit: ' || v\_id\_vanzator); -- Verificam ID-ul vanzatorului

-- Verifica numarul de clienti care au plasat comenzi pentru acest vanzator in ultimul an

SELECT COUNT(DISTINCT c.ID\_CLIENT)

INTO v\_numar\_clienti

FROM COMANDA cm

JOIN CLIENT c ON cm.ID\_CLIENT = c.ID\_CLIENT

WHERE c.ID\_VANZATOR = v\_id\_vanzator

AND cm.data\_comanda > ADD\_MONTHS(SYSDATE, -12); -- Comenzi din ultimul an

DBMS\_OUTPUT.PUT\_LINE('Numar clienti gasiti: ' || v\_numar\_clienti); -- Debugging

-- Daca nu sunt clienti, returneaza un mesaj informativ

IF v\_numar\_clienti < 1 THEN

RETURN 'Vanzatorul nu are clienti in ultimul an.';

END IF;

-- Daca sunt clienti, returneaza un mesaj cu numarul acestora

RETURN 'Vanzatorul are ' || v\_numar\_clienti || ' clienti activi in ultimele 12 luni.';

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

-- Ridicam exceptia pentru cazurile in care vanzatorul nu a fost gasit

DBMS\_OUTPUT.PUT\_LINE('Eroare: Nu exista vanzator cu acest nume'); -- Debugging

RAISE\_APPLICATION\_ERROR(-20000, 'Nu exista un vanzator cu acest nume');

WHEN TOO\_MANY\_ROWS THEN

-- Ridicam exceptia pentru cazul in care sunt mai multi vanzatori cu acelasi nume

DBMS\_OUTPUT.PUT\_LINE('Eroare: Exista mai multi vanzatori cu acest nume'); -- Debugging

RAISE\_APPLICATION\_ERROR(-20001, 'Exista mai multi vanzatori cu acest nume');

WHEN OTHERS THEN

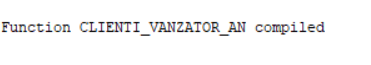
-- Ridicam o exceptie generala pentru alte cazuri

DBMS\_OUTPUT.PUT\_LINE('Eroare necunoscuta: ' || SQLERRM); -- Debugging

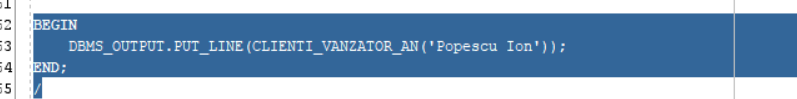
RAISE\_APPLICATION\_ERROR(-20002, 'Alta eroare necunoscuta.');

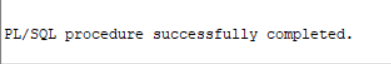
END CLIENTI\_VANZATOR\_AN;

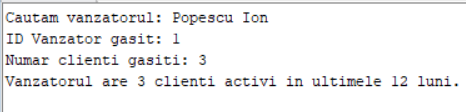
/

******

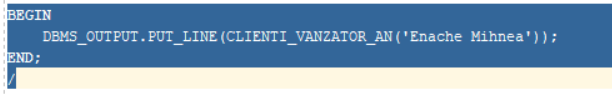
* + 1. Un apel care afiseaza bine:

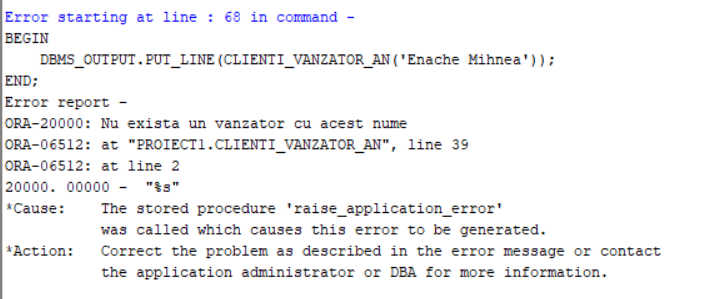






* + 1. Un apel pentru cazul in care nu exista vanzatorul:







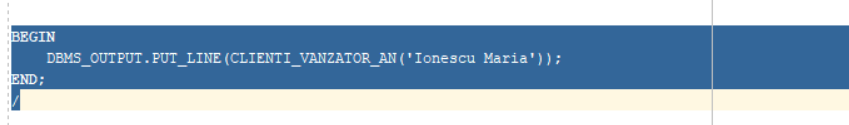
* + 1. Un apel pentru cazul in care avem mai multi vanzatori cu acest nume

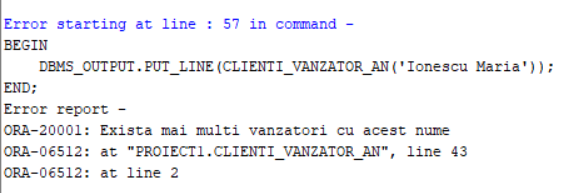
Inseram un nou angajat cu un nume deja folosit

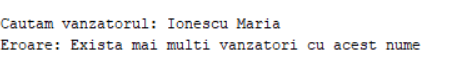
INSERT INTO VANZATOR (ID\_VANZATOR, ID\_MAGAZIN, nume\_vanzator, prenume\_vanzator, telefon\_vanzator, data\_angajare\_vanzator)

VALUES (13, 3, 'Ionescu', 'Maria', '0745123456', TO\_DATE('2024-01-01', 'YYYY-MM-DD'));

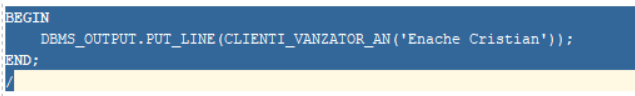




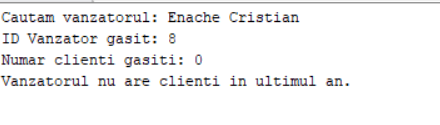




* + 1. Un apel pentru un vanzator fara clienti in ultimul an







***Cerinta 9***

Managerul unui magazin de materiale de constructii doreste sa obtina un raport detaliat despre o comanda plasata de un client. Raportul trebuie sa contina informatii despre client, magazinul de unde a fost gestionata comanda, vanzatorul responsabil, intervalul de livrare si produsele achizitionate.

Pentru a realiza acest lucru, se va implementa o procedura stocata independenta care primeste ca parametri numele si prenumele clientului si ID-ul comenzii. Procedura va interoga 5 tabele pentru a obtine toate informatiile necesare si va trata mai multe exceptii care pot aparea, cum ar fi:

1. Client inexistent – Daca nu exista niciun client cu numele si prenumele specificat, procedura va semnala o eroare.
2. Client duplicat – Daca exista mai multi clienti cu acelasi nume si prenume, se va cere o specificare suplimentara.
3. Comanda care nu apartine clientului – Daca ID-ul comenzii nu este asociat clientului respectiv, se va genera o eroare.
4. Comanda fara produse – Daca o comanda valida nu contine produse, se va semnala aceasta situatie.
5. ID de comanda invalid – Daca ID-ul comenzii nu este un numar valid, procedura va semnala o eroare.
6. Orice alta eroare neasteptata – Va fi tratata printr-un mesaj general de eroare.

Procedura va fi apelata in diferite scenarii pentru a evidentia toate cazurile tratate.

Tabele utilizate si scopul lor in procedura:

1. CLIENT – Folosit pentru a verifica daca clientul exista in baza de date pe baza numelui si prenumelui.
2. COMANDA – Folosit pentru a verifica daca exista o comanda cu ID-ul respectiv si daca aceasta apartine clientului.
3. VANZATOR – Folosit pentru a obtine informatii despre vanzatorul asociat clientului care a plasat comanda.
4. MAGAZIN – Folosit pentru a afla magazinul de unde a fost gestionata comanda.
5. COMANDA\_CONTINE\_PRODUSE – Folosit pentru a extrage produsele incluse in comanda si a verifica daca exista produse in aceasta.

CREATE OR REPLACE PROCEDURE raport\_comanda\_client(

p\_nume\_client IN VARCHAR2,

p\_prenume\_client IN VARCHAR2,

p\_id\_comanda IN VARCHAR2

) IS

-- Variabile pentru datele extrase

v\_id\_client CLIENT.ID\_CLIENT%TYPE;

v\_nume\_magazin MAGAZIN.NUME\_MAGAZIN%TYPE;

v\_nume\_vanzator VANZATOR.NUME\_VANZATOR%TYPE;

v\_prenume\_vanzator VANZATOR.PRENUME\_VANZATOR%TYPE;

v\_interval\_livrare COMANDA.INTERVAL\_LIVRARE%TYPE;

v\_id\_comanda NUMBER;

-- Cursori

CURSOR curs\_produse\_comanda(v\_id\_comanda NUMBER) IS

SELECT P.NUME\_PRODUS, CCP.CANTITATE, CCP.PRET\_COMANDA

FROM COMANDA\_CONTINE\_PRODUSE CCP

JOIN PRODUSE P ON CCP.ID\_PRODUS = P.ID\_PRODUS

WHERE CCP.ID\_COMANDA = v\_id\_comanda;

-- Variabile pentru produse

v\_nume\_produs PRODUSE.NUME\_PRODUS%TYPE;

v\_cantitate COMANDA\_CONTINE\_PRODUSE.CANTITATE%TYPE;

v\_pret\_total COMANDA\_CONTINE\_PRODUSE.PRET\_COMANDA%TYPE;

-- Exceptii definite de utilizator

ex\_comanda\_fara\_produse EXCEPTION;

ex\_comanda\_nu\_apartine\_clientului EXCEPTION;

ex\_id\_invalid EXCEPTION;

ex\_param\_invalid EXCEPTION;

BEGIN

-- Validare parametri nume si prenume

IF NOT REGEXP\_LIKE(p\_nume\_client, '^[A-Za-z ]+$') OR NOT REGEXP\_LIKE(p\_prenume\_client, '^[A-Za-z ]+$') THEN

RAISE ex\_param\_invalid;

END IF;

-- Conversie id comanda

BEGIN

v\_id\_comanda := TO\_NUMBER(p\_id\_comanda);

IF v\_id\_comanda < 1 THEN

RAISE ex\_id\_invalid;

END IF;

EXCEPTION

WHEN VALUE\_ERROR THEN

RAISE ex\_id\_invalid;

END;

-- Verificare existenta client

BEGIN

SELECT ID\_CLIENT INTO v\_id\_client

FROM CLIENT

WHERE UPPER(NUME\_CLIENT) = UPPER(p\_nume\_client)

AND UPPER(PRENUME\_CLIENT) = UPPER(p\_prenume\_client);

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Eroare: Clientul nu exista.');

RETURN;

WHEN TOO\_MANY\_ROWS THEN

DBMS\_OUTPUT.PUT\_LINE('Eroare: Exista mai multi clienti cu acelasi nume.');

RETURN;

END;

-- Verificare existenta comanda si apartenenta la client

BEGIN

SELECT M.NUME\_MAGAZIN, V.NUME\_VANZATOR, V.PRENUME\_VANZATOR, C.INTERVAL\_LIVRARE

INTO v\_nume\_magazin, v\_nume\_vanzator, v\_prenume\_vanzator, v\_interval\_livrare

FROM COMANDA C

JOIN CLIENT CL ON C.ID\_CLIENT = CL.ID\_CLIENT

JOIN VANZATOR V ON CL.ID\_VANZATOR = V.ID\_VANZATOR

JOIN MAGAZIN M ON V.ID\_MAGAZIN = M.ID\_MAGAZIN

WHERE C.ID\_COMANDA = v\_id\_comanda AND C.ID\_CLIENT = v\_id\_client;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

RAISE ex\_comanda\_nu\_apartine\_clientului;

END;

-- Afisare informatii comanda

DBMS\_OUTPUT.PUT\_LINE('Client: ' || p\_nume\_client || ' ' || p\_prenume\_client);

DBMS\_OUTPUT.PUT\_LINE('Magazin: ' || v\_nume\_magazin);

DBMS\_OUTPUT.PUT\_LINE('Vanzator: ' || v\_nume\_vanzator || ' ' || v\_prenume\_vanzator);

DBMS\_OUTPUT.PUT\_LINE('Interval livrare: ' || v\_interval\_livrare);

DBMS\_OUTPUT.PUT\_LINE('Produse comandate:');

-- Verificare existenta produse in comanda

OPEN curs\_produse\_comanda(v\_id\_comanda);

FETCH curs\_produse\_comanda INTO v\_nume\_produs, v\_cantitate, v\_pret\_total;

IF curs\_produse\_comanda%NOTFOUND THEN

CLOSE curs\_produse\_comanda;

RAISE ex\_comanda\_fara\_produse;

END IF;

-- Afisare produse

DBMS\_OUTPUT.PUT\_LINE(' - Produs: ' || v\_nume\_produs ||

', Cantitate: ' || v\_cantitate ||

', Pret total: ' || v\_pret\_total);

LOOP

FETCH curs\_produse\_comanda INTO v\_nume\_produs, v\_cantitate, v\_pret\_total;

EXIT WHEN curs\_produse\_comanda%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE(' - Produs: ' || v\_nume\_produs ||

', Cantitate: ' || v\_cantitate ||

', Pret total: ' || v\_pret\_total);

END LOOP;

CLOSE curs\_produse\_comanda;

EXCEPTION

WHEN ex\_id\_invalid THEN

DBMS\_OUTPUT.PUT\_LINE('Eroare: ID-ul comenzii trebuie sa fie un numar natural.');

WHEN ex\_param\_invalid THEN

DBMS\_OUTPUT.PUT\_LINE('Eroare: Parametrii introdusi sunt invalidi.');

WHEN ex\_comanda\_fara\_produse THEN

DBMS\_OUTPUT.PUT\_LINE('Eroare: Comanda nu contine produse.');

WHEN ex\_comanda\_nu\_apartine\_clientului THEN

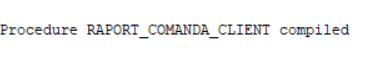
DBMS\_OUTPUT.PUT\_LINE('Eroare: Comanda nu apartine clientului.');

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('A aparut o eroare neprevazuta: ' || SQLERRM);

END raport\_comanda\_client;

/



Cazul in care exista si clientul si comanda, iar aceasta nu este goala:

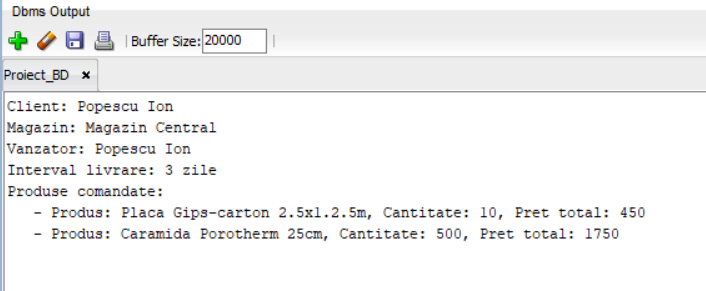
BEGIN

raport\_comanda\_client('Popescu', 'Ion', 1);

END;

/





Cazul in care nu exista comanda respectiva pentru clientul respectiv:

BEGIN

raport\_comanda\_client('Popescu', 'Ion', 100);

END;

/





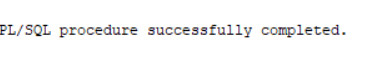
Cazul in care clientul nu exista:

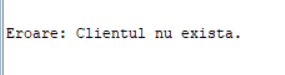
BEGIN

raport\_comanda\_client('POPPP', 'Ion', 100);

END;

/





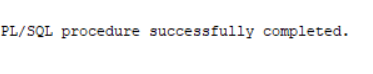
Cazul in care comanda este goala:

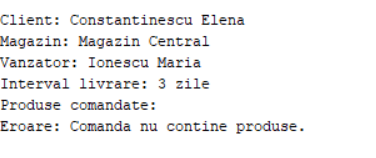
BEGIN

raport\_comanda\_client('Constantinescu', 'Elena', 21);

END;

/





Cazul in care nu se ofera date valide pentru apel

BEGIN

raport\_comanda\_client('Popescu', 'Ion', -100);

END;

/





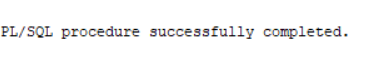
Cazul in care parametrii sunt invalizi

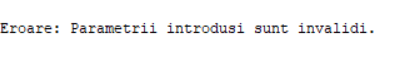
BEGIN

raport\_comanda\_client('Enache123', 'Cristian', '5');

END;

/





***Cerinta 10***

Problema:

Managerul magazinului doreste sa monitorizeze orice schimbare importanta in procesul de gestionare a comenzilor. Astfel, se solicita crearea unui trigger de tip LMD (insert, update, delete) care sa inregistreze toate operatiile efectuate asupra tabelei COMANDA. Trigger-ul va trebui sa afiseze un mesaj in consola de iesire cu privire la tipul operatiei (INSERT, UPDATE, DELETE) si ID-ul comenzii asupra careia s-a efectuat operatia respectiva.

CREATE OR REPLACE TRIGGER TRIG\_LOG\_COMANDA

AFTER INSERT OR UPDATE OR DELETE ON COMANDA

DECLARE

v\_tip\_operatiune VARCHAR2(20);

BEGIN

-- Determinam tipul operatiei

IF INSERTING THEN

v\_tip\_operatiune := 'INSERT';

ELSIF UPDATING THEN

v\_tip\_operatiune := 'UPDATE';

ELSIF DELETING THEN

v\_tip\_operatiune := 'DELETE';

END IF;

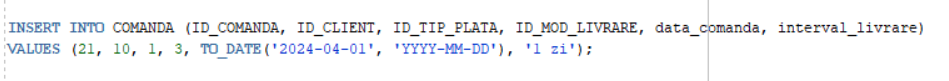
-- Inseram in tabela de log pentru operatia respectiva

DBMS\_OUTPUT.PUT\_LINE('Operatia executata: ' || v\_tip\_operatiune);

END TRIG\_LOG\_COMANDA;

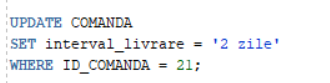


1. Operatia de inseare:



1. Operatia de modificare:



1. Operatia de stergere:



***Cerinta 11***

Problema:

Managerul magazinului doreste sa monitorizeze modificarile efectuate asupra produselor din stoc. Acesta vrea sa pastreze un istoric al modificarilor, inclusiv adaugarea unui nou produs, actualizarea unui produs existent sau stergerea unui produs. In acest sens, se cere crearea unui trigger de tip LMD (INSERT, UPDATE, DELETE) la nivel de linie care sa salveze urmatoarele informatii pentru fiecare modificare efectuata asupra tabelului PRODUSE:

* ID-ul produsului asupra caruia s-a efectuat operatia.
* Data si ora la care a fost efectuata modificarea.
* Coloana care a fost modificata (in caz de UPDATE).
* Vechea valoare a coloanei modificate (pentru UPDATE).
* Noua valoare a coloanei modificate (pentru UPDATE).
* In cazul unui INSERT, se va salva doar noua valoare a produsului adaugat.
* In cazul unui DELETE, se va salva valoarea veche a produsului sters.

Trigger-ul va salva aceste informatii intr-un tabel de log PRODUSE\_UPDATE pentru a putea urmari istoricul modificarilor din sistem.

--Inseram tabelul PRODUSE\_UPDATE in baza noastra de date:

CREATE TABLE PRODUSE\_UPDATE (

ID\_PRODUS INT,

DATA\_MODIFICARE DATE,

COLUMNA\_MODIFICATA VARCHAR2(50),

VALOARE\_VEche VARCHAR2(255),

VALOARE\_NOUA VARCHAR2(255)

);

CREATE OR REPLACE TRIGGER TRIG\_UPDATE\_PRODUSE

AFTER INSERT OR UPDATE OR DELETE ON PRODUSE

FOR EACH ROW

DECLARE

v\_col\_modificat VARCHAR2(50);

BEGIN

-- Cazul pentru INSERT

IF INSERTING THEN

v\_col\_modificat := 'INSERT';

INSERT INTO PRODUSE\_UPDATE (ID\_PRODUS, DATA\_MODIFICARE, COLUMNA\_MODIFICATA, VALOARE\_VEche, VALOARE\_NOUA)

VALUES (:NEW.ID\_PRODUS, SYSDATE, v\_col\_modificat, NULL, :NEW.nume\_produs);

END IF;

-- Cazul pentru UPDATE

IF UPDATING THEN

-- Verificam care coloana a fost modificata si salvam vechea si noua valoare

IF :OLD.nume\_produs != :NEW.nume\_produs THEN

v\_col\_modificat := 'nume\_produs';

INSERT INTO PRODUSE\_UPDATE (ID\_PRODUS, DATA\_MODIFICARE, COLUMNA\_MODIFICATA, VALOARE\_VEche, VALOARE\_NOUA)

VALUES (:OLD.ID\_PRODUS, SYSDATE, v\_col\_modificat, :OLD.nume\_produs, :NEW.nume\_produs);

END IF;

IF :OLD.descriere\_produs != :NEW.descriere\_produs THEN

v\_col\_modificat := 'descriere\_produs';

INSERT INTO PRODUSE\_UPDATE (ID\_PRODUS, DATA\_MODIFICARE, COLUMNA\_MODIFICATA, VALOARE\_VEche, VALOARE\_NOUA)

VALUES (:OLD.ID\_PRODUS, SYSDATE, v\_col\_modificat, :OLD.descriere\_produs, :NEW.descriere\_produs);

END IF;

IF :OLD.pret\_produs != :NEW.pret\_produs THEN

v\_col\_modificat := 'pret\_produs';

INSERT INTO PRODUSE\_UPDATE (ID\_PRODUS, DATA\_MODIFICARE, COLUMNA\_MODIFICATA, VALOARE\_VEche, VALOARE\_NOUA)

VALUES (:OLD.ID\_PRODUS, SYSDATE, v\_col\_modificat, :OLD.pret\_produs, :NEW.pret\_produs);

END IF;

IF :OLD.stoc\_produs != :NEW.stoc\_produs THEN

v\_col\_modificat := 'stoc\_produs';

INSERT INTO PRODUSE\_UPDATE (ID\_PRODUS, DATA\_MODIFICARE, COLUMNA\_MODIFICATA, VALOARE\_VEche, VALOARE\_NOUA)

VALUES (:OLD.ID\_PRODUS, SYSDATE, v\_col\_modificat, :OLD.stoc\_produs, :NEW.stoc\_produs);

END IF;

END IF;

-- Cazul pentru DELETE

IF DELETING THEN

v\_col\_modificat := 'DELETE';

INSERT INTO PRODUSE\_UPDATE (ID\_PRODUS, DATA\_MODIFICARE, COLUMNA\_MODIFICATA, VALOARE\_VEche, VALOARE\_NOUA)

VALUES (:OLD.ID\_PRODUS, SYSDATE, v\_col\_modificat, :OLD.nume\_produs, NULL);

END IF;

END TRIG\_UPDATE\_PRODUSE;

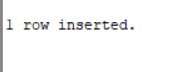
/



Inseram un nou produs in tabelul PRODUSE:

INSERT INTO PRODUSE (ID\_PRODUS, ID\_CATEGORIE\_PRODUS, ID\_DEPOZIT, nume\_produs, descriere\_produs, pret\_produs, stoc\_produs)

VALUES (19, 3, 3, 'Lemn de rasinoase 5m', 'Lemn de rasinoase uscat, ideal pentru constructii', 55.00, 200);





Modificam produsul inserat:

UPDATE PRODUSE

SET nume\_produs = 'Lemn de rasinoase 6m', pret\_produs = 30.00

WHERE ID\_PRODUS = 19;

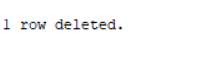




Stergerea produsului intrdous anterior:

DELETE FROM PRODUSE

WHERE ID\_PRODUS = 19;





***Cerinta 12***

Managerul bazei de date doreste sa monitorizeze toate modificarile structurii bazei de date, in special modificarile care implica tabelele din schema bazei de date. Acesta vrea sa aiba un istoric al actiunilor importante, cum ar fi adaugarea unui nou tabel (CREATE), modificarea unui tabel existent (ALTER) si stergerea unui tabel (DROP). Mai mult, managerul doreste ca stergerea unui tabel sa fie interzisa pentru a preveni pierderea accidentala a datelor.

In acest sens, se solicita implementarea unui trigger care sa se declanseze inainte de efectuarea acestor actiuni (LDD - BEFORE DROP, ALTER, CREATE). Trigger-ul trebuie sa:

1. Interzica stergerea tabelelor (DROP).
2. Salveze informatii inregistrand tipul actiunii (CREATE, ALTER) si numele tabelului afectat, precum si data la care a avut loc modificarea.

Tabelul MONITORIZARE\_TABELE are rolul de a inregistra actiunile efectuate asupra structurii bazei de date, in special asupra tabelelor. Acesta este folosit pentru a salva un istoric al actiunilor de tip CREATE, ALTER si DROP care sunt efectuate asupra tabelelor din baza de date.

CREATE TABLE MONITORIZARE\_TABELE(

id NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,

actiune VARCHAR2(50),

tabel VARCHAR2(50),

data\_actiune DATE

);

****

CREATE OR REPLACE TRIGGER trigger\_ldd

BEFORE DROP OR ALTER OR CREATE ON SCHEMA

DECLARE

v\_tip\_actiune VARCHAR2(50); -- Variabila pentru tipul actiunii (CREATE, ALTER, DROP)

BEGIN

-- Detectam actiunea si setam tipul acesteia

IF ora\_sysevent = 'DROP' AND ora\_dict\_obj\_type = 'TABLE' THEN

v\_tip\_actiune := 'DROP'; -- Salvam tipul actiunii

-- Interzicem stergerea tabelelor

RAISE\_APPLICATION\_ERROR(-20001, 'Stergerea tabelelor este interzisa in aceasta baza de date.');

ELSIF ora\_sysevent IN ('CREATE', 'ALTER') AND ora\_dict\_obj\_type = 'TABLE' THEN

v\_tip\_actiune := ora\_sysevent; -- Salvam tipul actiunii (CREATE sau ALTER)

-- Logam actiunea in tabelul de monitorizare

INSERT INTO MONITORIZARE\_TABELE (actiune, tabel, data\_actiune)

VALUES (v\_tip\_actiune, ora\_dict\_obj\_name, SYSDATE);

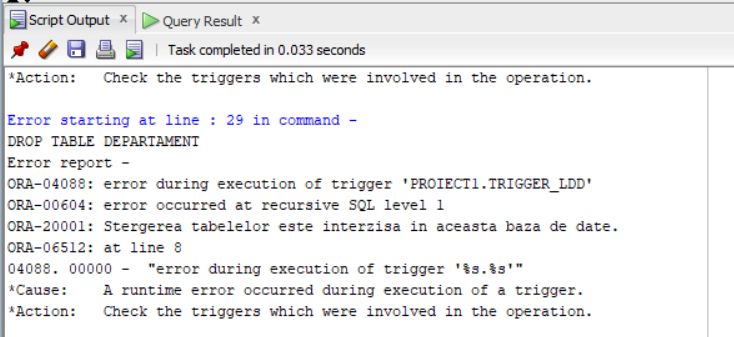
END IF;

END;

/

Incercarea de tergere a unui tabel:

DROP TABLE DEPARTAMENT;



Inserarea unui nou tabel in baza de date:

CREATE TABLE TEST (

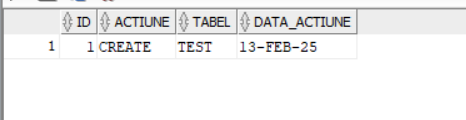
ID\_PRODUS INT GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,

DESCRIERE\_PRODUS VARCHAR2(100),

CANTITATE INT

);





Modificarea unui tabel:

ALTER TABLE TEST

ADD nume VARCHAR2(100);

