COMPANIA DE COSMETICE

BERLINSCHI ALEXANDRA MARINA

GRUPA 242

Cerinta 1

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

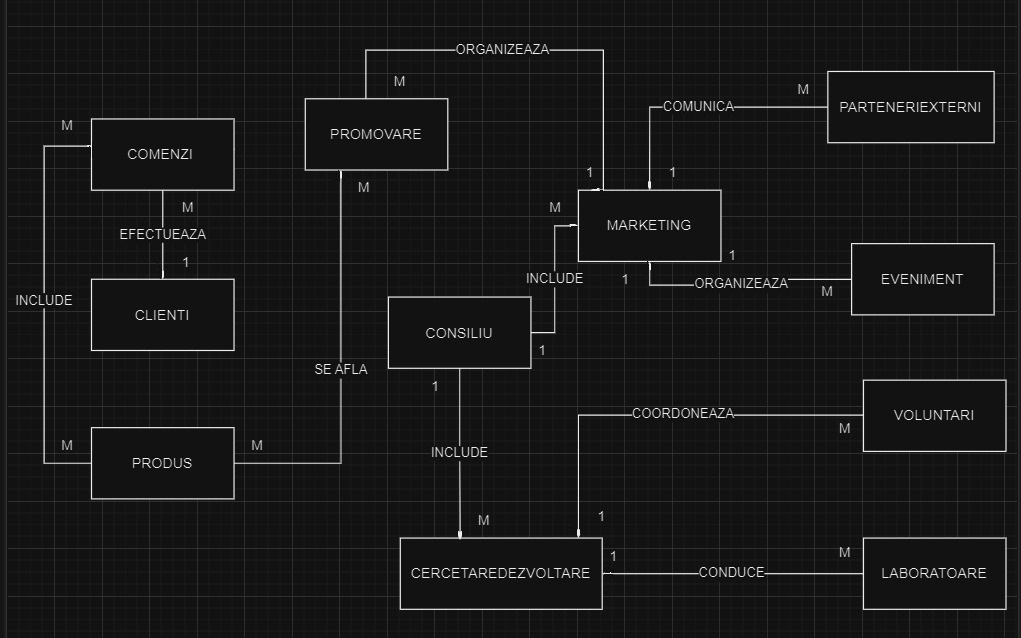
Compania de cosmetice produce și distribuie produse cosmetice de calitate superioara în intreaga lume. De asemenea, are mai multe linii de produse, inclusiv ingrijirea pielii, machiaj, parfumuri și produse de ingrijire personala.

Echipa de cercetare și dezvoltare dezvolta și testeaza noi produse, iar aceste produse sunt testate pe un grup de voluntari inainte de a fi lansate pe piața. Detine propriile laboratoare pentru productia produselor cosmetice si are parteneri externi pentru a livra produsele la clienti.

Compania are un departament de marketing care se concentreaza pe strategii de branding și promovare pentru a isi promova produsele. Acest department colaboreaza cu departamentul de vanzari pentru a gasi noi piete si pentru a creste vanzarile existente. Baza de date relationala va fi utilizata pentru a gestiona toate informatiile legate de produse, cercetare si dezvoltare, productie, distributie, vanzari și marketing. Acesta va permite companiei sa isi gestioneze eficient toate aspectele operationale ale afacerii.

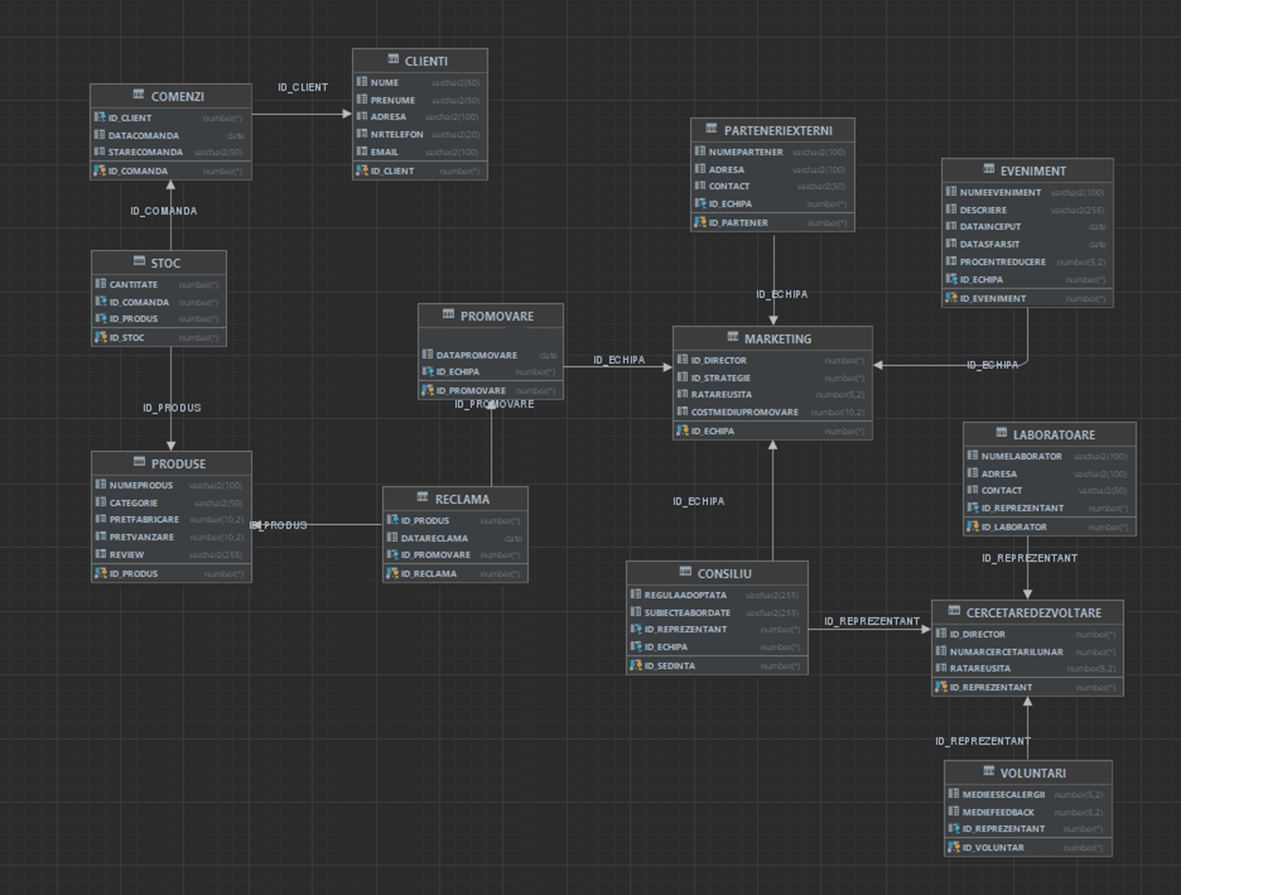
Cerinta 2

**Realizați diagrama entitate-relație (ERD): entitățile, relațiile și atributele trebuie definite în limba română (vezi curs SGBD / model de diagrama ERD; nu se va accepta alt format).**

****

Cerinta 3

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

****

Cerinta 4

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

CREATE TABLE Comenzi (

ID\_Comanda INT PRIMARY KEY,

ID\_Client INT,

DataComanda DATE,

StareComanda VARCHAR(50),

FOREIGN KEY (ID\_Client) REFERENCES Clienti (ID\_Client)

);

CREATE TABLE Clienti (

ID\_Client INT PRIMARY KEY,

Nume VARCHAR(50),

Prenume VARCHAR(50),

Adresa VARCHAR(100),

NrTelefon VARCHAR(20),

Email VARCHAR(100)

);

CREATE TABLE Produse

(

ID\_Produs INT PRIMARY KEY,

NumeProdus VARCHAR(100),

Categorie VARCHAR(50),

PretFabricare DECIMAL(10, 2),

PretVanzare DECIMAL(10, 2),

Review VARCHAR(255)

);

CREATE TABLE Stoc (

ID\_Stoc INT PRIMARY KEY,

Cantitate INT,

ID\_Comanda INT,

ID\_Produs INT,

FOREIGN KEY (ID\_Comanda) REFERENCES Comenzi (ID\_Comanda),

FOREIGN KEY (ID\_Produs) REFERENCES Produse (ID\_Produs)

);

CREATE TABLE Consiliu (

ID\_Sedinta INT PRIMARY KEY,

RegulaAdoptata VARCHAR(255),

SubiecteAbordate VARCHAR(255),

ID\_Reprezentant INT,

ID\_Echipa INT,

FOREIGN KEY (ID\_Echipa) REFERENCES Marketing (ID\_Echipa),

FOREIGN KEY (ID\_Reprezentant) REFERENCES CercetareDezvoltare (ID\_Reprezentant)

);

CREATE TABLE CercetareDezvoltare (

ID\_Reprezentant INT PRIMARY KEY,

ID\_Director INT,

NumarCercetariLunar INT,

RataReusita DECIMAL(5, 2)

);

CREATE TABLE Marketing (

ID\_Echipa INT PRIMARY KEY,

Id\_Director INT,

ID\_Strategie INT,

RataReusita DECIMAL(5, 2),

CostMediuPromovare DECIMAL(10, 2)

);

CREATE TABLE ParteneriExterni (

ID\_Partener INT PRIMARY KEY,

NumePartener VARCHAR(100),

Adresa VARCHAR(100),

Contact VARCHAR(50),

ID\_Echipa INT,

FOREIGN KEY (ID\_Echipa) REFERENCES Marketing (ID\_Echipa)

);

CREATE TABLE Eveniment (

ID\_Eveniment INT PRIMARY KEY,

NumeEveniment VARCHAR(100),

Descriere VARCHAR(255),

DataInceput DATE,

DataSfarsit DATE,

ProcentReducere DECIMAL(5, 2),

ID\_Echipa INT,

FOREIGN KEY (ID\_Echipa) REFERENCES Marketing (ID\_Echipa)

);

CREATE TABLE Promovare (

ID\_Promovare INT PRIMARY KEY,

DataPromovare DATE,

ID\_Echipa INT,

FOREIGN KEY (ID\_Echipa) REFERENCES Marketing (ID\_Echipa)

);

CREATE TABLE Reclama (

ID\_Reclama INT PRIMARY KEY,

ID\_Produs INT,

DataReclama DATE,

ID\_Promovare INT,

FOREIGN KEY (ID\_Produs) REFERENCES Produse (ID\_Produs),

FOREIGN KEY (ID\_Promovare) REFERENCES Promovare (ID\_Promovare)

);

CREATE TABLE Voluntari (

ID\_Voluntar INT PRIMARY KEY,

MedieEsecAlergii DECIMAL(5, 2),

MedieFeedback DECIMAL(5, 2),

ID\_Reprezentant INT,

FOREIGN KEY (ID\_Reprezentant) REFERENCES CercetareDezvoltare (ID\_Reprezentant)

);

CREATE TABLE Laboratoare (

ID\_Laborator INT PRIMARY KEY,

NumeLaborator VARCHAR(100),

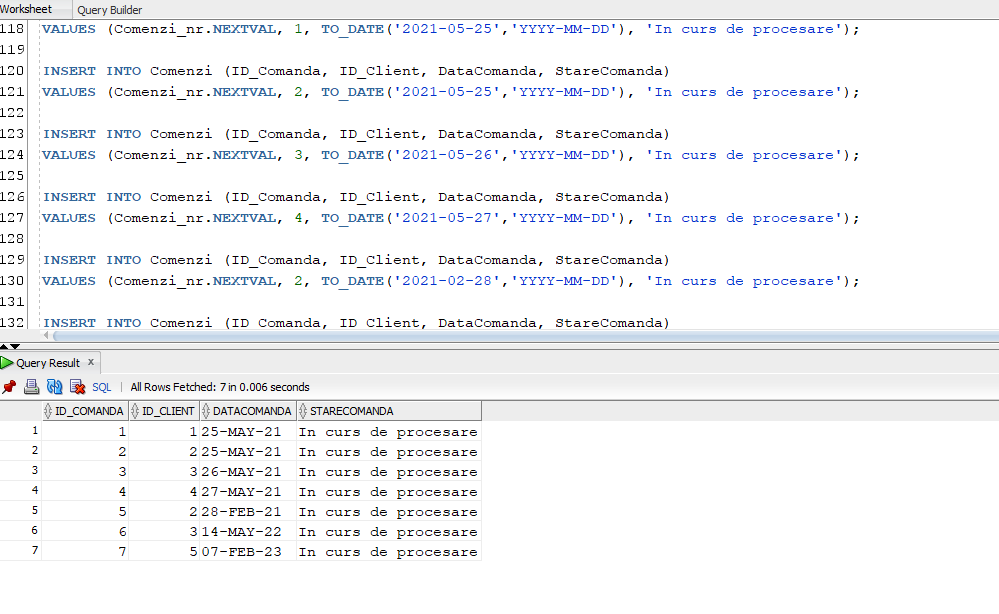
Adresa VARCHAR(100),

Contact VARCHAR(50),

ID\_Reprezentant INT,

FOREIGN KEY (ID\_Reprezentant) REFERENCES CercetareDezvoltare (ID\_Reprezentant)

);



Cerinta 5

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

CREATE SEQUENCE Comenzi\_nr

START WITH 1

INCREMENT BY 1

MAXVALUE 7

NOCYCLE;

INSERT INTO Comenzi (ID\_Comanda, ID\_Client, DataComanda, StareComanda)

VALUES (Comenzi\_nr.NEXTVAL, 1, TO\_DATE('2021-05-25','YYYY-MM-DD'), 'In curs de procesare');

INSERT INTO Comenzi (ID\_Comanda, ID\_Client, DataComanda, StareComanda)

VALUES (Comenzi\_nr.NEXTVAL, 2, TO\_DATE('2021-05-25','YYYY-MM-DD'), 'In curs de procesare');

INSERT INTO Comenzi (ID\_Comanda, ID\_Client, DataComanda, StareComanda)

VALUES (Comenzi\_nr.NEXTVAL, 3, TO\_DATE('2021-05-26','YYYY-MM-DD'), 'In curs de procesare');

INSERT INTO Comenzi (ID\_Comanda, ID\_Client, DataComanda, StareComanda)

VALUES (Comenzi\_nr.NEXTVAL, 4, TO\_DATE('2021-05-27','YYYY-MM-DD'), 'In curs de procesare');

INSERT INTO Comenzi (ID\_Comanda, ID\_Client, DataComanda, StareComanda)

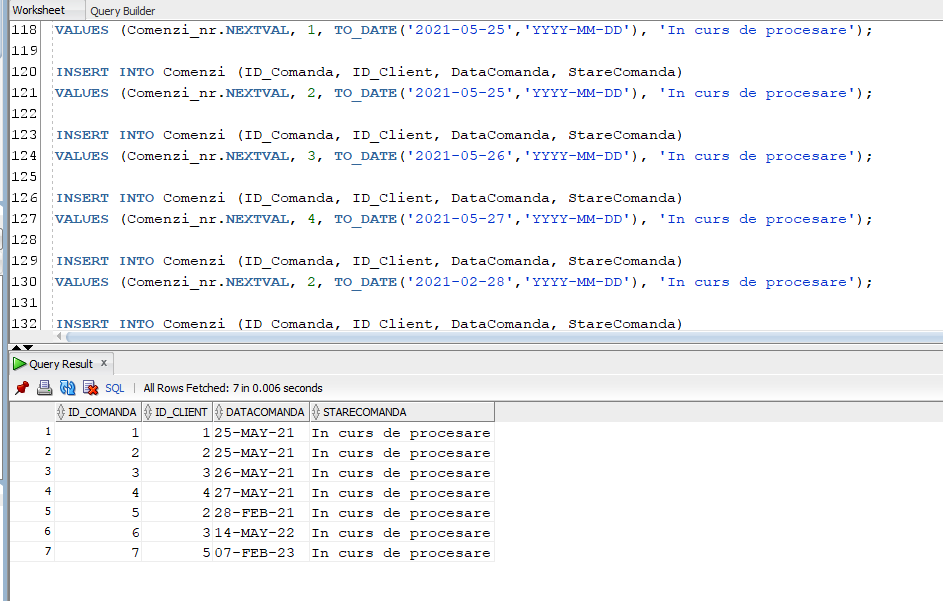
VALUES (Comenzi\_nr.NEXTVAL, 2, TO\_DATE('2021-02-28','YYYY-MM-DD'), 'In curs de procesare');

INSERT INTO Comenzi (ID\_Comanda, ID\_Client, DataComanda, StareComanda)

VALUES (Comenzi\_nr.NEXTVAL, 3, TO\_DATE('2022-05-14','YYYY-MM-DD'), 'In curs de procesare');

INSERT INTO Comenzi (ID\_Comanda, ID\_Client, DataComanda, StareComanda)

VALUES (Comenzi\_nr.NEXTVAL, 5, TO\_DATE('2023-02-07','YYYY-MM-DD'), 'In curs de procesare');



CREATE SEQUENCE Client\_nr

START WITH 1

INCREMENT BY 1

MAXVALUE 5

NOCYCLE;

INSERT INTO Clienti ( ID\_Client, Nume, Prenume, Adresa, NrTelefon, Email )

VALUES (Client\_nr.NEXTVAL, 'Popescu', 'Ana', 'Str. Florilor, Nr. 10', '0721122334', 'ana.popescu@gmail.com');

INSERT INTO Clienti ( ID\_Client, Nume, Prenume, Adresa, NrTelefon, Email )

VALUES (Client\_nr.NEXTVAL, 'Ionescu', 'Maria', 'Str. Florilor, Nr. 11', '0721122335', 'ionescu.maria@gmail.com');

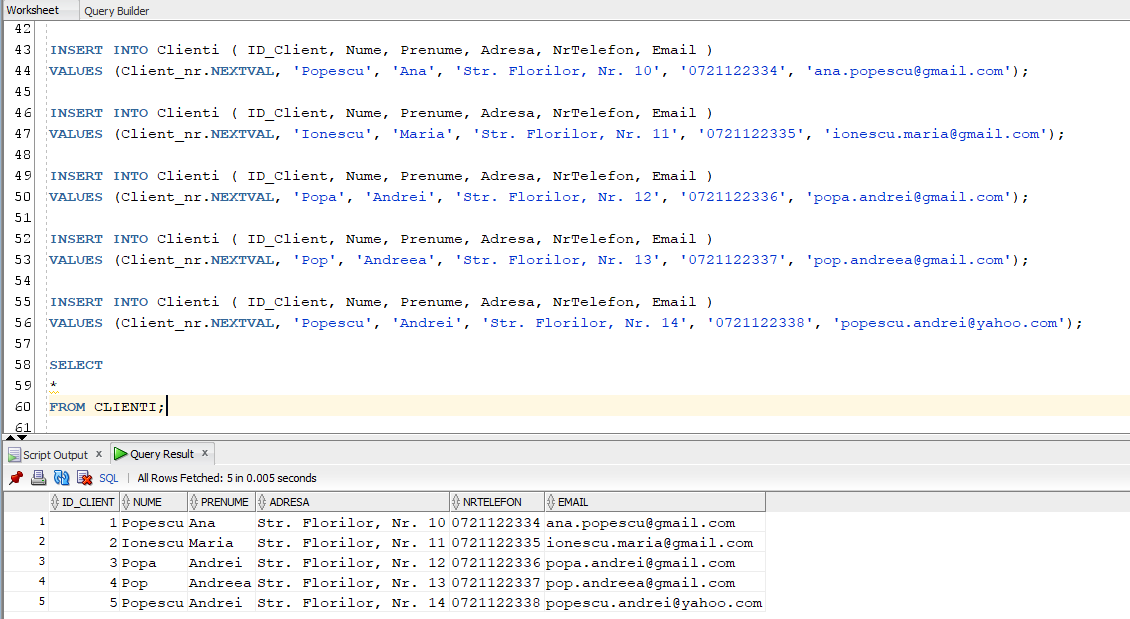
INSERT INTO Clienti ( ID\_Client, Nume, Prenume, Adresa, NrTelefon, Email )

VALUES (Client\_nr.NEXTVAL, 'Popa', 'Andrei', 'Str. Florilor, Nr. 12', '0721122336', 'popa.andrei@gmail.com');

INSERT INTO Clienti ( ID\_Client, Nume, Prenume, Adresa, NrTelefon, Email )

VALUES (Client\_nr.NEXTVAL, 'Pop', 'Andreea', 'Str. Florilor, Nr. 13', '0721122337', 'pop.andreea@gmail.com');

INSERT INTO Clienti ( ID\_Client, Nume, Prenume, Adresa, NrTelefon, Email )

VALUES (Client\_nr.NEXTVAL, 'Popescu', 'Andrei', 'Str. Florilor, Nr. 14', '0721122338', 'popescu.andrei@yahoo.com');

INSERT INTO Marketing (ID\_Echipa, Id\_Director, ID\_STRATEGIE, RataReusita, CostMediuPromovare)

VALUES (1, 1, 12, 0.9, 500.00);

INSERT INTO Marketing (ID\_Echipa, Id\_Director, ID\_STRATEGIE, RataReusita, CostMediuPromovare)

VALUES (2, 2, 10, 0.8, 600.00);

INSERT INTO Marketing (ID\_Echipa, Id\_Director, ID\_STRATEGIE, RataReusita, CostMediuPromovare)

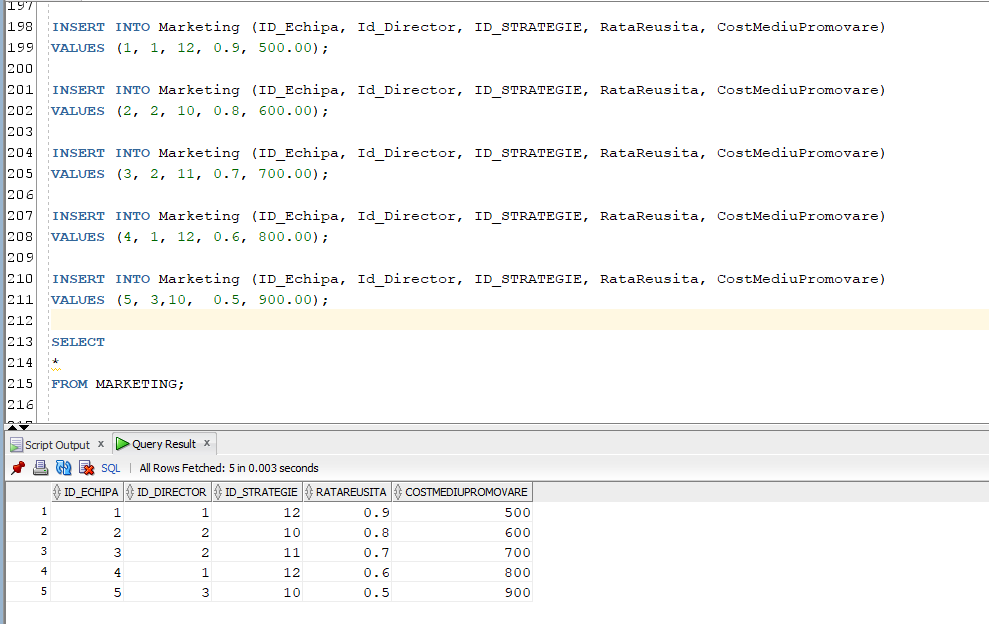
VALUES (3, 2, 11, 0.7, 700.00);

INSERT INTO Marketing (ID\_Echipa, Id\_Director, ID\_STRATEGIE, RataReusita, CostMediuPromovare)

VALUES (4, 1, 12, 0.6, 800.00);

INSERT INTO Marketing (ID\_Echipa, Id\_Director, ID\_STRATEGIE, RataReusita, CostMediuPromovare)

VALUES (5, 3,10, 0.5, 900.00);



INSERT INTO Produse (ID\_Produs, NumeProdus, Categorie, PretFabricare, PretVanzare, Review)

VALUES (1, 'Rimel', 'Machiaj', 10.00, 20.00, 'Bun');

INSERT INTO Produse (ID\_Produs, NumeProdus, Categorie, PretFabricare, PretVanzare, Review)

VALUES (2, 'Ruj', 'Machiaj', 15.00, 30.00, 'Foarte bun');

INSERT INTO Produse (ID\_Produs, NumeProdus, Categorie, PretFabricare, PretVanzare, Review)

VALUES (3, 'Crema de fata', 'Ingrijirea pielii', 10.00, 20.00, 'Bun');

INSERT INTO Produse (ID\_Produs, NumeProdus, Categorie, PretFabricare, PretVanzare, Review)

VALUES (4, 'Fond de ten', 'Machiaj', 12.00, 25.00, 'Foarte bun');

INSERT INTO Produse (ID\_Produs, NumeProdus, Categorie, PretFabricare, PretVanzare, Review)

VALUES (5, 'Gel de curatare ', 'Ingrijirea tenului', 15.00, 30.00, 'Excelent');

INSERT INTO Produse (ID\_Produs, NumeProdus, Categorie, PretFabricare, PretVanzare, Review)

VALUES (6, 'SPF', 'Ingrijirea pielii', 30.00, 70.00, 'Excelent');

INSERT INTO Produse (ID\_Produs, NumeProdus, Categorie, PretFabricare, PretVanzare, Review)

VALUES (7, 'Pasta de dinti', 'Ingrijirea dentara', 5.00, 12.00, 'Excelent');

INSERT INTO Produse (ID\_Produs, NumeProdus, Categorie, PretFabricare, PretVanzare, Review)

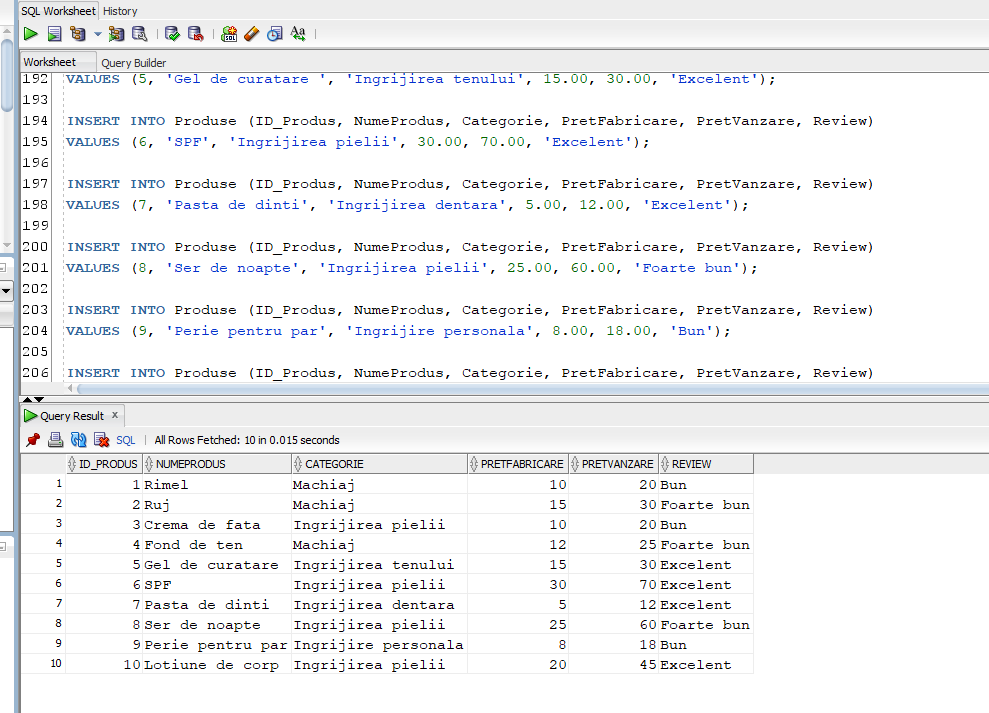
VALUES (8, 'Ser de noapte', 'Ingrijirea pielii', 25.00, 60.00, 'Foarte bun');

INSERT INTO Produse (ID\_Produs, NumeProdus, Categorie, PretFabricare, PretVanzare, Review)

VALUES (9, 'Perie pentru par', 'Ingrijire personala', 8.00, 18.00, 'Bun');

INSERT INTO Produse (ID\_Produs, NumeProdus, Categorie, PretFabricare, PretVanzare, Review)

VALUES (10, 'Lotiune de corp', 'Ingrijirea pielii', 20.00, 45.00, 'Excelent');



INSERT INTO Stoc (ID\_Stoc, Cantitate, ID\_Comanda, ID\_Produs)

VALUES (101, 100, 7, 5);

INSERT INTO Stoc (ID\_Stoc, Cantitate, ID\_Comanda, ID\_Produs)

VALUES (102, 200, 2, 4);

INSERT INTO Stoc (ID\_Stoc, Cantitate, ID\_Comanda, ID\_Produs)

VALUES (103, 300, 1, 9);

INSERT INTO Stoc (ID\_Stoc, Cantitate, ID\_Comanda, ID\_Produs)

VALUES (107, 300, 5, 2);

INSERT INTO Stoc (ID\_Stoc, Cantitate, ID\_Comanda, ID\_Produs)

VALUES (104, 450, 3, 3);

INSERT INTO Stoc (ID\_Stoc, Cantitate, ID\_Comanda, ID\_Produs)

VALUES (105, 520, 1, 1);

INSERT INTO Stoc (ID\_Stoc, Cantitate, ID\_Comanda, ID\_Produs)

VALUES (106, 128, 6, 6);

INSERT INTO Stoc (ID\_Stoc, Cantitate, ID\_Comanda, ID\_Produs)

VALUES (108, 89, 5, 8);

INSERT INTO Stoc (ID\_Stoc, Cantitate, ID\_Comanda, ID\_Produs)

VALUES (109, 382, 7, 7)

INSERT INTO Stoc (ID\_Stoc, Cantitate, ID\_Comanda, ID\_Produs)

VALUES (110, 203, 1, 10);

INSERT INTO Stoc (ID\_Stoc, Cantitate, ID\_Comanda, ID\_Produs)

VALUES (111, 100, 4, 2);

INSERT INTO Stoc (ID\_Stoc, Cantitate, ID\_Comanda, ID\_Produs)

VALUES (112, 200, 3, 1);

INSERT INTO Stoc (ID\_Stoc, Cantitate, ID\_Comanda, ID\_Produs)

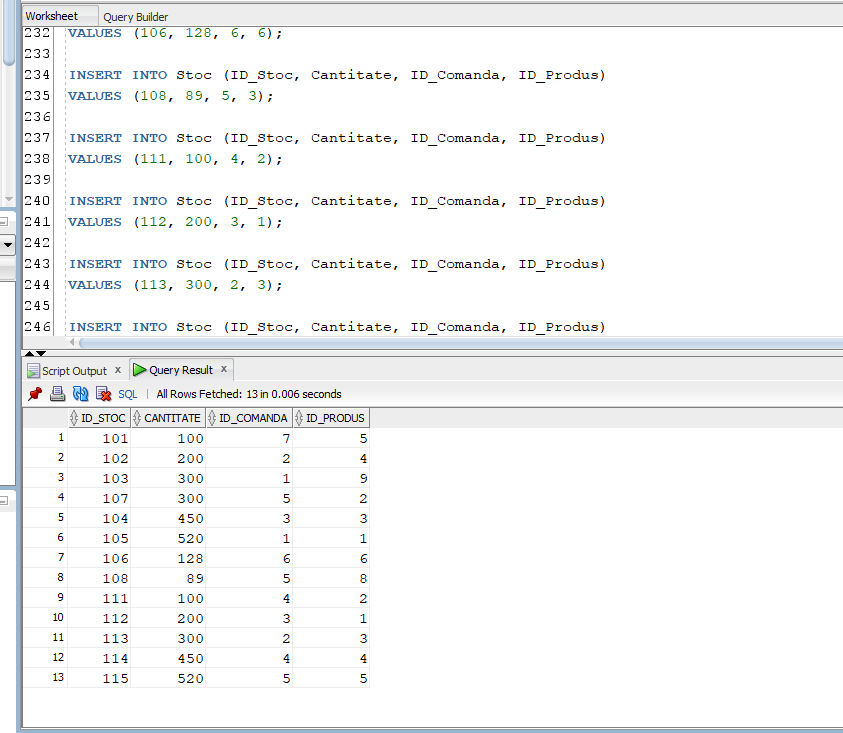
VALUES (113, 300, 2, 3);

INSERT INTO Stoc (ID\_Stoc, Cantitate, ID\_Comanda, ID\_Produs)

VALUES (114, 450, 4, 4);

INSERT INTO Stoc (ID\_Stoc, Cantitate, ID\_Comanda, ID\_Produs)

VALUES (115, 520, 5, 5);



INSERT INTO CONSILIU(ID\_Sedinta,RegulaAdoptata, SubiecteAbordate, ID\_ECHIPA, ID\_Reprezentant)

VALUES (1, 'Regula 1', 'Subiect 1', 2, 4);

INSERT INTO CONSILIU(ID\_Sedinta,RegulaAdoptata, SubiecteAbordate, ID\_ECHIPA, ID\_Reprezentant)

VALUES (2, 'Regula 2', 'Subiect 2', 5, 4);

INSERT INTO CONSILIU(ID\_Sedinta,RegulaAdoptata, SubiecteAbordate, ID\_ECHIPA, ID\_Reprezentant)

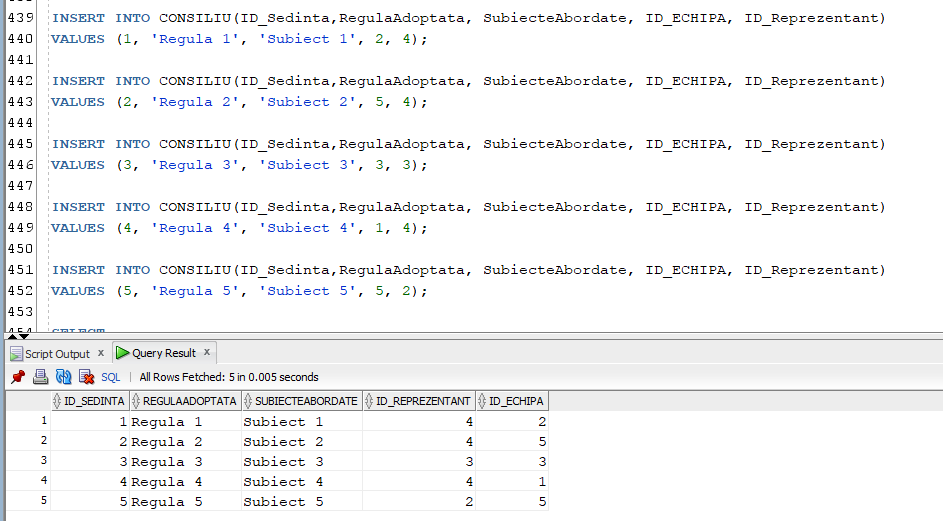
VALUES (3, 'Regula 3', 'Subiect 3', 3, 3);

INSERT INTO CONSILIU(ID\_Sedinta,RegulaAdoptata, SubiecteAbordate, ID\_ECHIPA, ID\_Reprezentant)

VALUES (4, 'Regula 4', 'Subiect 4', 1, 4);

INSERT INTO CONSILIU(ID\_Sedinta,RegulaAdoptata, SubiecteAbordate, ID\_ECHIPA, ID\_Reprezentant)

VALUES (5, 'Regula 5', 'Subiect 5', 5, 2);



INSERT INTO CercetareDezvoltare (ID\_Reprezentant, ID\_Director, NumarCercetariLunar, RataReusita)

VALUES (1, 1, 5, 0.8);

INSERT INTO CercetareDezvoltare (ID\_Reprezentant, ID\_Director, NumarCercetariLunar, RataReusita)

VALUES (2, 2, 8, 0.7);

INSERT INTO CercetareDezvoltare (ID\_Reprezentant, ID\_Director, NumarCercetariLunar, RataReusita)

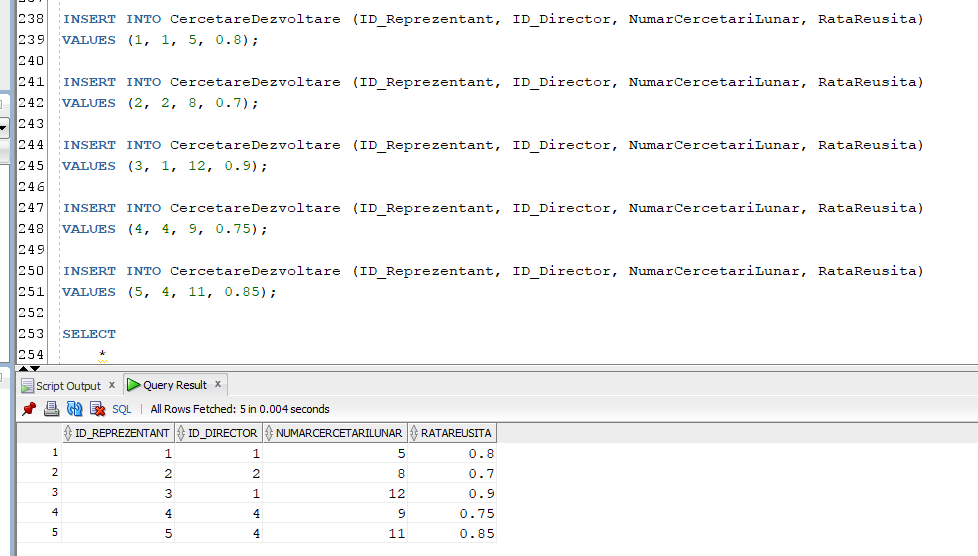
VALUES (3, 1, 12, 0.9);

INSERT INTO CercetareDezvoltare (ID\_Reprezentant, ID\_Director, NumarCercetariLunar, RataReusita)

VALUES (4, 4, 9, 0.75);

INSERT INTO CercetareDezvoltare (ID\_Reprezentant, ID\_Director, NumarCercetariLunar, RataReusita)

VALUES (5, 4, 11, 0.85);



INSERT INTO ParteneriExterni (ID\_Partener, NumePartener, Adresa, Contact, ID\_Echipa)

VALUES (1, 'Compania ABC', 'Str. Principala 123, Orasul X', 'Mihai Popa', 1);

INSERT INTO ParteneriExterni (ID\_Partener, NumePartener, Adresa, Contact, ID\_Echipa)

VALUES (2, 'Corporatia XYZ', 'Str. Secundara 456, Orasul Y', 'Andreea Maria', 2);

INSERT INTO ParteneriExterni (ID\_Partener, NumePartener, Adresa, Contact, ID\_Echipa)

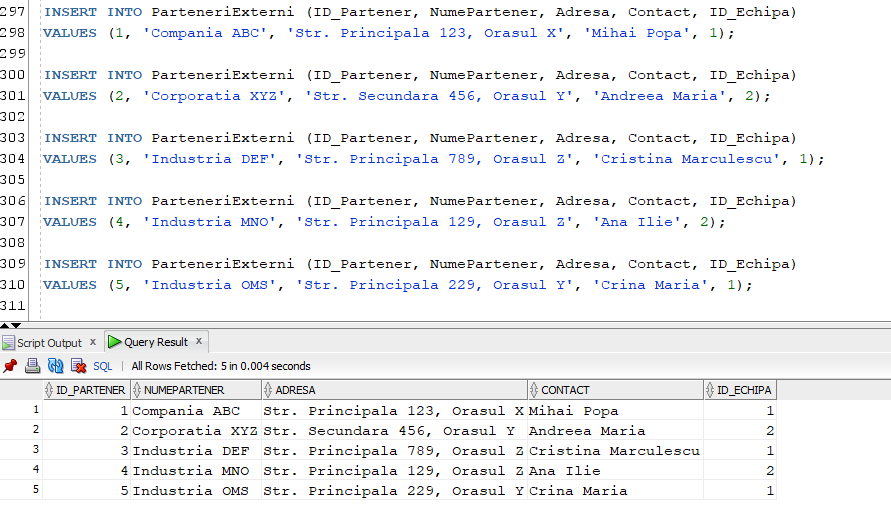
VALUES (3, 'Industria DEF', 'Str. Principala 789, Orasul Z', 'Cristina Marculescu', 1);

INSERT INTO ParteneriExterni (ID\_Partener, NumePartener, Adresa, Contact, ID\_Echipa)

VALUES (4, 'Industria MNO', 'Str. Principala 129, Orasul Z', 'Ana Ilie', 2);

INSERT INTO ParteneriExterni (ID\_Partener, NumePartener, Adresa, Contact, ID\_Echipa)

VALUES (5, 'Industria OMS', 'Str. Principala 229, Orasul Y', 'Crina Maria', 1);



INSERT INTO Eveniment(ID\_Eveniment, NumeEveniment, Descriere, DataInceput, DataSfarsit, ProcentReducere, ID\_Echipa)

VALUES (1, 'Black Friday', 'Reduceri de 50% la toate produsele', TO\_DATE('2020-11-27','YYYY-MM-DD'), TO\_DATE('2020-11-29','YYYY-MM-DD'), 50, 1);

INSERT INTO Eveniment(ID\_Eveniment, NumeEveniment, Descriere, DataInceput, DataSfarsit, ProcentReducere, ID\_Echipa)

VALUES (2, 'Craciun', 'Reduceri de 30% la toate produsele', TO\_DATE('2020-12-24','YYYY-MM-DD'), TO\_DATE('2020-12-26','YYYY-MM-DD'), 30, 2);

INSERT INTO Eveniment(ID\_Eveniment, NumeEveniment, Descriere, DataInceput, DataSfarsit, ProcentReducere, ID\_Echipa)

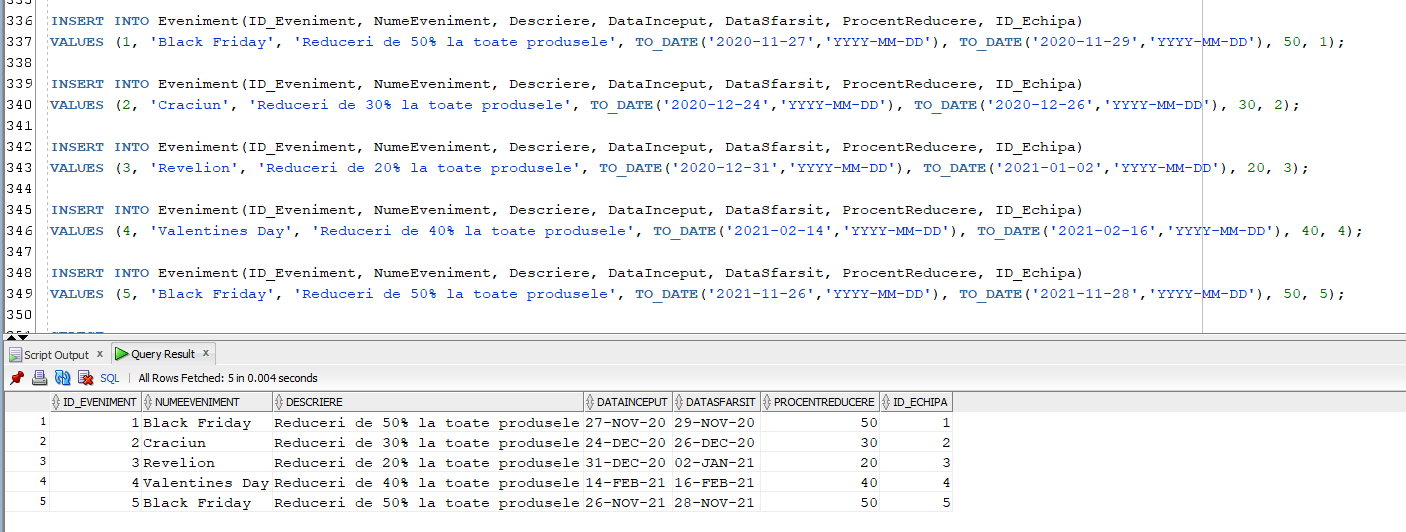
VALUES (3, 'Revelion', 'Reduceri de 20% la toate produsele', TO\_DATE('2020-12-31','YYYY-MM-DD'), TO\_DATE('2021-01-02','YYYY-MM-DD'), 20, 3);

INSERT INTO Eveniment(ID\_Eveniment, NumeEveniment, Descriere, DataInceput, DataSfarsit, ProcentReducere, ID\_Echipa)

VALUES (4, 'Valentines Day', 'Reduceri de 40% la toate produsele', TO\_DATE('2021-02-14','YYYY-MM-DD'), TO\_DATE('2021-02-16','YYYY-MM-DD'), 40, 4);

INSERT INTO Eveniment(ID\_Eveniment, NumeEveniment, Descriere, DataInceput, DataSfarsit, ProcentReducere, ID\_Echipa)

VALUES (5, 'Black Friday', 'Reduceri de 50% la toate produsele', TO\_DATE('2021-11-26','YYYY-MM-DD'), TO\_DATE('2021-11-28','YYYY-MM-DD'), 50, 5);



INSERT INTO Promovare(ID\_PROMOVARE, DATAPROMOVARE, ID\_ECHIPA)

VALUES (1, TO\_DATE('2020-11-27','YYYY-MM-DD'), 1);

INSERT INTO Promovare(ID\_PROMOVARE, DATAPROMOVARE, ID\_ECHIPA)

VALUES (2, TO\_DATE('2020-11-27','YYYY-MM-DD'), 2);

INSERT INTO Promovare(ID\_PROMOVARE, DATAPROMOVARE, ID\_ECHIPA)

VALUES (3, TO\_DATE('2020-11-27','YYYY-MM-DD'), 3);

INSERT INTO Promovare(ID\_PROMOVARE, DATAPROMOVARE, ID\_ECHIPA)

VALUES (4, TO\_DATE('2020-11-27','YYYY-MM-DD'), 4);

INSERT INTO Promovare(ID\_PROMOVARE, DATAPROMOVARE, ID\_ECHIPA)

VALUES (5, TO\_DATE('2020-11-27','YYYY-MM-DD'), 5);

INSERT INTO Promovare(ID\_PROMOVARE, DATAPROMOVARE, ID\_ECHIPA)

VALUES (6, TO\_DATE('2020-11-27','YYYY-MM-DD'), 1);

INSERT INTO Promovare(ID\_PROMOVARE, DATAPROMOVARE, ID\_ECHIPA)

VALUES (7, TO\_DATE('2020-11-27','YYYY-MM-DD'), 2);

INSERT INTO Promovare(ID\_PROMOVARE, DATAPROMOVARE, ID\_ECHIPA)

VALUES (8, TO\_DATE('2020-11-27','YYYY-MM-DD'), 2);

INSERT INTO Promovare(ID\_PROMOVARE, DATAPROMOVARE, ID\_ECHIPA)

VALUES (9, TO\_DATE('2020-11-27','YYYY-MM-DD'), 2);

INSERT INTO Promovare(ID\_PROMOVARE, DATAPROMOVARE, ID\_ECHIPA)

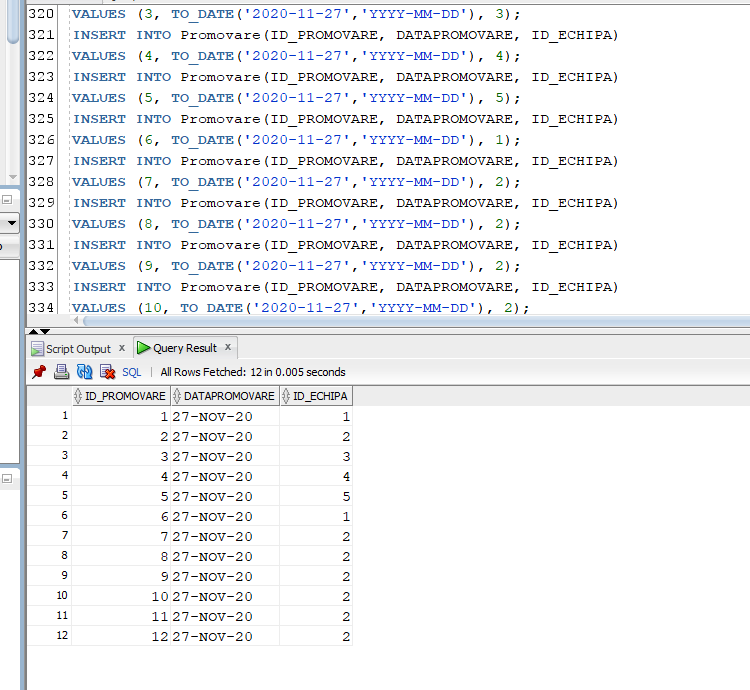
VALUES (10, TO\_DATE('2020-11-27','YYYY-MM-DD'), 2);

INSERT INTO Promovare(ID\_PROMOVARE, DATAPROMOVARE, ID\_ECHIPA)

VALUES (11, TO\_DATE('2020-11-27','YYYY-MM-DD'), 2);

INSERT INTO Promovare(ID\_PROMOVARE, DATAPROMOVARE, ID\_ECHIPA)

VALUES (12, TO\_DATE('2020-11-27','YYYY-MM-DD'), 2);



INSERT INTO Reclama(ID\_Reclama, DataReclama, ID\_PRODUS, ID\_PROMOVARE)

VALUES (1, TO\_DATE('2020-11-27','YYYY-MM-DD'), 1, 1);

INSERT INTO Reclama(ID\_Reclama, DataReclama, ID\_PRODUS, ID\_PROMOVARE)

VALUES (2, TO\_DATE('2020-11-27','YYYY-MM-DD'), 2, 2);

INSERT INTO Reclama(ID\_Reclama, DataReclama, ID\_PRODUS, ID\_PROMOVARE)

VALUES (3, TO\_DATE('2020-11-27','YYYY-MM-DD'), 3, 3);

INSERT INTO Reclama(ID\_Reclama, DataReclama, ID\_PRODUS, ID\_PROMOVARE)

VALUES (4, TO\_DATE('2020-11-27','YYYY-MM-DD'), 4, 4);

INSERT INTO Reclama(ID\_Reclama, DataReclama, ID\_PRODUS, ID\_PROMOVARE)

VALUES (5, TO\_DATE('2020-11-27','YYYY-MM-DD'), 7, 5);

INSERT INTO Reclama(ID\_Reclama, DataReclama, ID\_PRODUS, ID\_PROMOVARE)

VALUES (6, TO\_DATE('2020-11-27','YYYY-MM-DD'), 2, 6);

INSERT INTO Reclama(ID\_Reclama, DataReclama, ID\_PRODUS, ID\_PROMOVARE)

VALUES (7, TO\_DATE('2020-11-27','YYYY-MM-DD'), 7, 7);

INSERT INTO Reclama(ID\_Reclama, DataReclama, ID\_PRODUS, ID\_PROMOVARE)

VALUES (8, TO\_DATE('2020-11-27','YYYY-MM-DD'), 4, 8);

INSERT INTO Reclama(ID\_Reclama, DataReclama, ID\_PRODUS, ID\_PROMOVARE)

VALUES (9, TO\_DATE('2020-11-27','YYYY-MM-DD'), 3, 9);

INSERT INTO Reclama(ID\_Reclama, DataReclama, ID\_PRODUS, ID\_PROMOVARE)

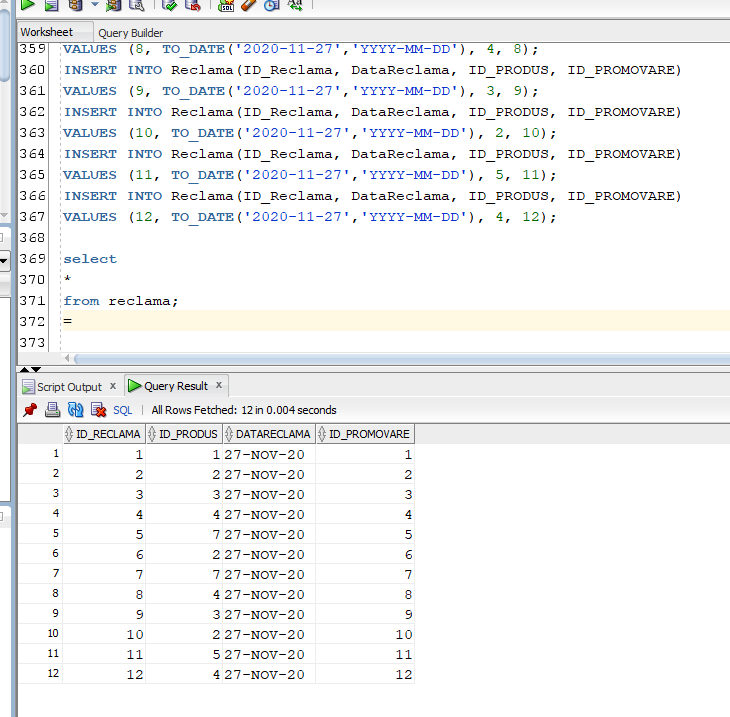
VALUES (10, TO\_DATE('2020-11-27','YYYY-MM-DD'), 2, 10);

INSERT INTO Reclama(ID\_Reclama, DataReclama, ID\_PRODUS, ID\_PROMOVARE)

VALUES (11, TO\_DATE('2020-11-27','YYYY-MM-DD'), 5, 11);

INSERT INTO Reclama(ID\_Reclama, DataReclama, ID\_PRODUS, ID\_PROMOVARE)

VALUES (12, TO\_DATE('2020-11-27','YYYY-MM-DD'), 4, 12);



INSERT INTO Voluntari (ID\_Voluntar, MedieEsecAlergii, MedieFeedback, ID\_Reprezentant)

VALUES (1, 0.2, 4.5, 1);

INSERT INTO Voluntari (ID\_Voluntar, MedieEsecAlergii, MedieFeedback, ID\_Reprezentant)

VALUES (2, 0.5, 4.2, 2);

INSERT INTO Voluntari (ID\_Voluntar, MedieEsecAlergii, MedieFeedback, ID\_Reprezentant)

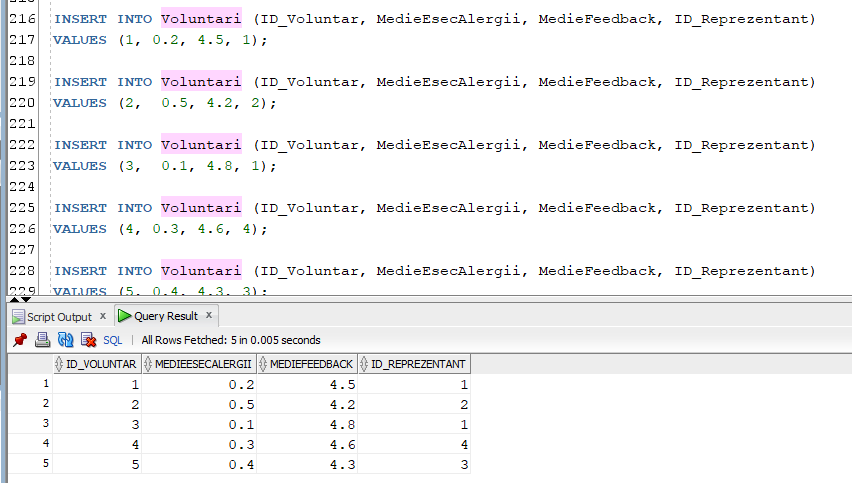
VALUES (3, 0.1, 4.8, 1);

INSERT INTO Voluntari (ID\_Voluntar, MedieEsecAlergii, MedieFeedback, ID\_Reprezentant)

VALUES (4, 0.3, 4.6, 4);

INSERT INTO Voluntari (ID\_Voluntar, MedieEsecAlergii, MedieFeedback, ID\_Reprezentant)

VALUES (5, 0.4, 4.3, 3);



INSERT INTO Laboratoare (ID\_Laborator, NumeLaborator, Adresa, Contact, ID\_Reprezentant)

VALUES (1, 'Laboratorul 1', 'Str. Principala 102, Orasul X', 'Claudia Marin', 1);

INSERT INTO Laboratoare (ID\_Laborator, NumeLaborator, Adresa, Contact, ID\_Reprezentant)

VALUES (2, 'Laboratorul 2', 'Str. Secundara 212, Orasul Y', 'Marina Dobos', 2);

INSERT INTO Laboratoare (ID\_Laborator, NumeLaborator, Adresa, Contact, ID\_Reprezentant)

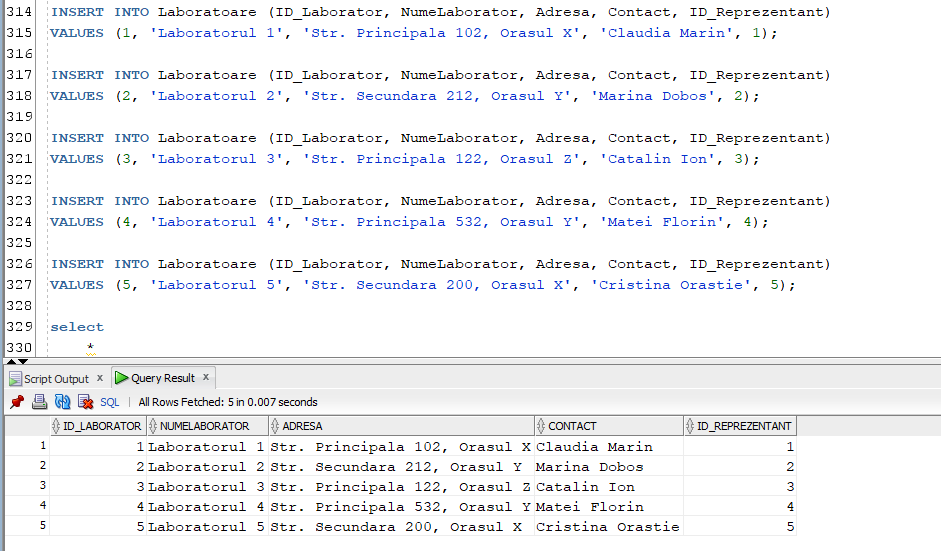
VALUES (3, 'Laboratorul 3', 'Str. Principala 122, Orasul Z', 'Catalin Ion', 3);

INSERT INTO Laboratoare (ID\_Laborator, NumeLaborator, Adresa, Contact, ID\_Reprezentant)

VALUES (4, 'Laboratorul 4', 'Str. Principala 532, Orasul Y', 'Matei Florin', 4);

INSERT INTO Laboratoare (ID\_Laborator, NumeLaborator, Adresa, Contact, ID\_Reprezentant)

VALUES (5, 'Laboratorul 5', 'Str. Secundara 200, Orasul X', 'Cristina Orastie', 5);



Cerinta 6

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

Un manager de produs dintr o companie are nevoie de un raport detaliat pentru un anumit produs din inventarul companiei. Raportul trebuie sa include informatii esentiale despre un produs cum ar fi numele, categoria si pretul de vanzare. De asemenea, managerul doreste sa afle detalii despre stocul disponibil pentru acest produs, incluzand cantitatea din fiecare stoc si comenzile asociate cu aceste stocuri. Mai mult, pentru a evalua impactul campaniilor de marketing, raportul trebuie sa prezinte o lista a tuturor reclamelor asociate cu produsul, cu datele lor de difuzare. In final, raportul trebuie sa ofere o imagine de ansamblu asupra categoriilor de produse disponibile in companie, pentru a asista la planificarea viitoare a stocurilor si a promotiilor.

CREATE OR REPLACE PROCEDURE RaportDetaliiProdus (

p\_ID\_Produs IN INT

) AS

TYPE ProdusInfo IS RECORD (

NumeProdus VARCHAR2(100),

Categorie VARCHAR2(50),

PretVanzare DECIMAL(10, 2)

);

TYPE StocInfo IS RECORD (

ID\_Stoc INT,

Cantitate INT,

ID\_Comanda INT

);

TYPE StocList IS TABLE OF StocInfo INDEX BY PLS\_INTEGER;

TYPE ReclamaType IS RECORD (

ID\_Reclama INT,

DataReclama DATE

);

TYPE ReclamaList IS TABLE OF ReclamaType;

TYPE CategorieList IS VARRAY(50) OF VARCHAR2(50);

produs\_info ProdusInfo;

stoc\_list StocList;

reclama\_list ReclamaList := ReclamaList();

categorie\_list CategorieList := CategorieList();

idx INT := 1;

BEGIN

SELECT NumeProdus, Categorie, PretVanzare

INTO produs\_info.NumeProdus, produs\_info.Categorie, produs\_info.PretVanzare

FROM Produse

WHERE ID\_Produs = p\_ID\_Produs;

SELECT ID\_Stoc, Cantitate, ID\_Comanda

BULK COLLECT INTO stoc\_list

FROM Stoc

WHERE ID\_Produs = p\_ID\_Produs;

SELECT ID\_Reclama, DataReclama

BULK COLLECT INTO reclama\_list

FROM Reclama

WHERE ID\_Produs = p\_ID\_Produs;

SELECT DISTINCT Categorie

BULK COLLECT INTO categorie\_list

FROM Produse

WHERE ROWNUM <= 50;

DBMS\_OUTPUT.PUT\_LINE('Raport pentru produsul: ' || produs\_info.NumeProdus);

DBMS\_OUTPUT.PUT\_LINE('Categorie: ' || produs\_info.Categorie || ', Pret de Vanzare: ' || produs\_info.PretVanzare);

DBMS\_OUTPUT.PUT\_LINE('Stocuri:');

FOR i IN stoc\_list.FIRST..stoc\_list.LAST LOOP

IF stoc\_list.EXISTS(i) THEN

DBMS\_OUTPUT.PUT\_LINE('ID Stoc: ' || stoc\_list(i).ID\_Stoc || ', Cantitate: ' || stoc\_list(i).Cantitate || ', ID Comanda: ' || stoc\_list(i).ID\_Comanda);

END IF;

END LOOP;

DBMS\_OUTPUT.PUT\_LINE('Reclame asociate:');

FOR i IN 1..reclama\_list.COUNT LOOP

DBMS\_OUTPUT.PUT\_LINE('ID Reclama: ' || reclama\_list(i).ID\_Reclama || ', Data: ' || TO\_CHAR(reclama\_list(i).DataReclama, 'DD-MM-YYYY'));

END LOOP;

DBMS\_OUTPUT.PUT\_LINE('Categorii de produse disponibile:');

FOR i IN 1..categorie\_list.COUNT LOOP

DBMS\_OUTPUT.PUT\_LINE(categorie\_list(i));

END LOOP;

END RaportDetaliiProdus;

/

DECLARE

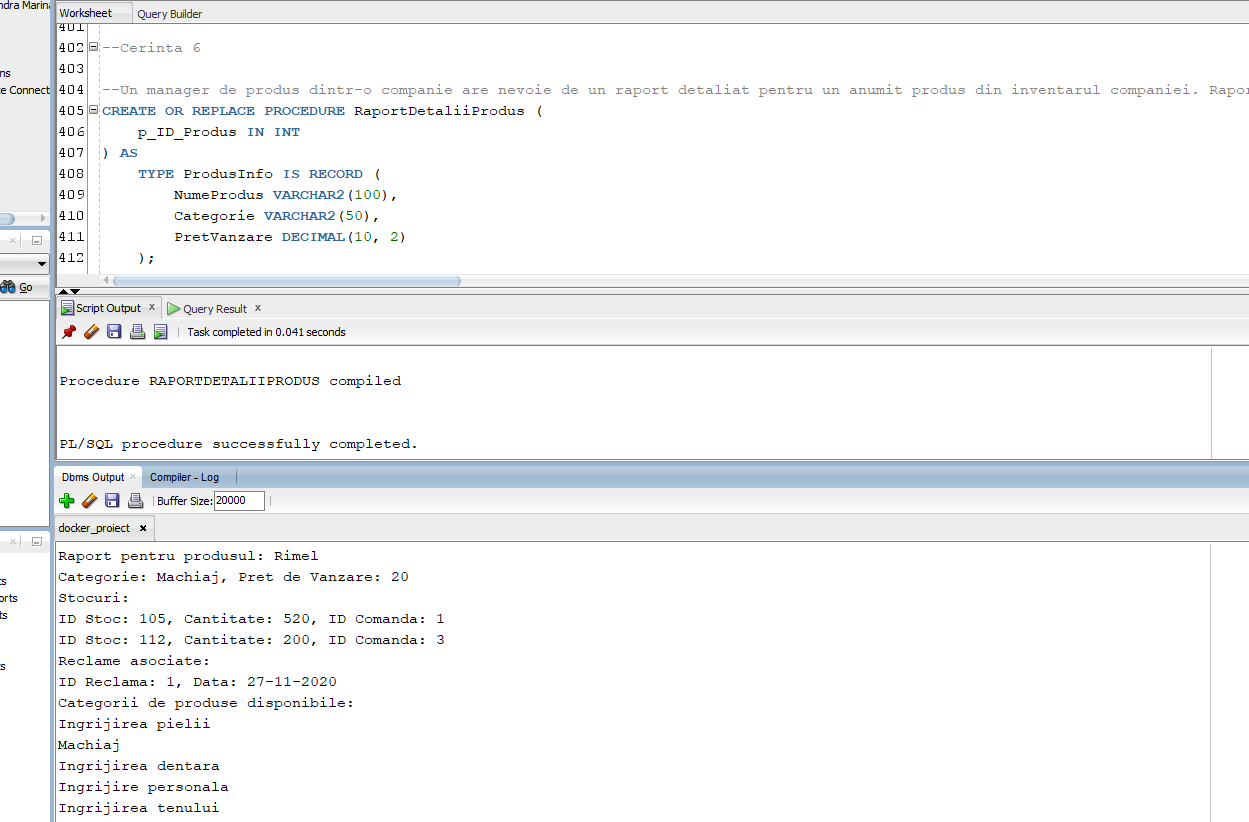
v\_ID\_Produs INT := 1;

BEGIN

RaportDetaliiProdus(v\_ID\_Produs);

END;

/



Cerinta 7

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

Un manager al departamentului de logistica al unei companii trebuie sa genereze rapoarte detaliate despre comenzile din baza de date. Acesta doreste sa afle informatii precum ID-ul comenzii, data si starea acesteia. De asemenea, managerul vrea sa includa in raport detalii despre produsele din stoc asociate fiecarei comenzi, cum ar fi ID-ul stocului, numele produsului, cantitatea si pretul de vanzare. Pentru a realiza aceasta, foloseste un subprogram stocat care utilizeaza doua tipuri de cursoare: unul implicit pentru a obtine informatii generale despre comenzile din baza de date si unul explicit parametrizat pentru a obtine detalii specifice despre produsele din stoc asociate fiecarei comenzi

CREATE OR REPLACE PROCEDURE RaportComenziBazaDate IS

CURSOR cursorComenzi IS

SELECT c.ID\_Comanda, c.DataComanda, c.StareComanda, cl.Nume

FROM Comenzi c

JOIN Clienti cl ON c.ID\_Client = cl.ID\_Client;

BEGIN

FOR comanda\_rec IN cursorComenzi LOOP

DBMS\_OUTPUT.PUT\_LINE('ID Comanda: ' || comanda\_rec.ID\_Comanda || ', Data: ' || comanda\_rec.DataComanda || ', Stare: ' || comanda\_rec.StareComanda || ', Nume Client: ' || comanda\_rec.Nume);

DECLARE

CURSOR cursorDetaliiComanda(p\_ID\_Comanda INT) IS

SELECT s.ID\_Stoc, s.Cantitate, p.NumeProdus, p.PretVanzare

FROM Stoc s

JOIN Produse p ON s.ID\_Produs = p.ID\_Produs

WHERE s.ID\_Comanda = p\_ID\_Comanda;

BEGIN

FOR detalii\_rec IN cursorDetaliiComanda(comanda\_rec.ID\_Comanda) LOOP

DBMS\_OUTPUT.PUT\_LINE(' Detalii Stoc: ID\_Stoc ' || detalii\_rec.ID\_Stoc || ', Produs: ' || detalii\_rec.NumeProdus || ', Cantitate: ' || detalii\_rec.Cantitate || ', Pret: ' || detalii\_rec.PretVanzare);

END LOOP;

END;

END LOOP;

END RaportComenziBazaDate;

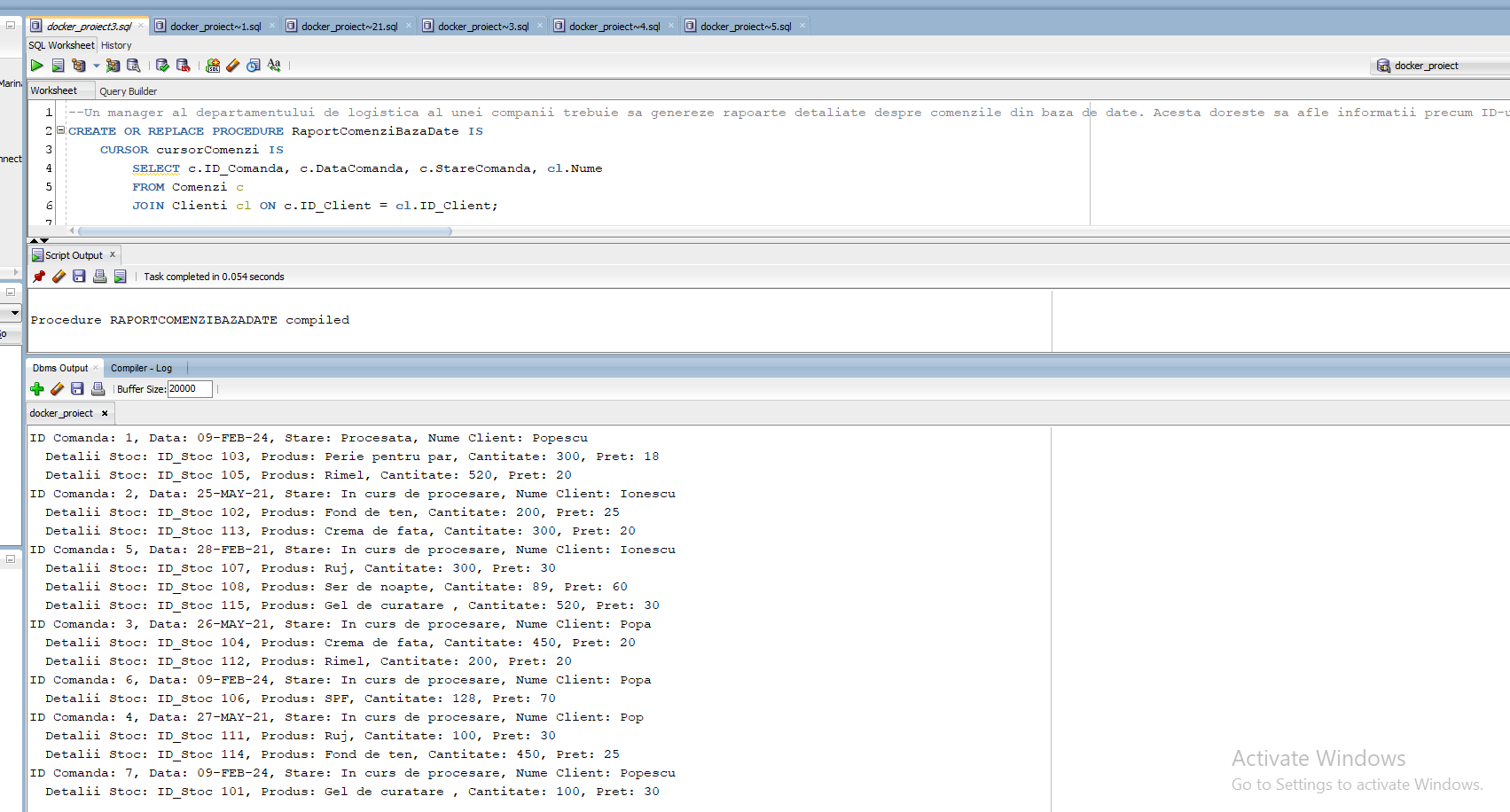
/

BEGIN

RaportComenziBazaDate;

END;

/



Cerinta 8

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

Un administrator al bazei de date trebuie sa creeze un raport detaliat pentru comenzile, clientii si produsele asociate. Raportul trebuie sa includa informatii despre ID-ul comenzii,informatii despre clienti, data si starea comenzii, precum si numele si categoria produsului. Pentru a realiza acest lucru, va folosi o functie stocata care va returna toate informatiile necesare, folosind o singura comanda SQL pentru a interoga tabelele Comenzi, Stoc si Produse. Functia trebuie sa poata gestiona doua exceptii personalizate: una pentru situatia in care nu exista informatii despre comenzile efectuate si alta pentru situatia in care nu exista informatii despre produse. Pentru a demonstra functionarea corecta a functiei, aceasta va fi apelata in trei scenarii diferite: apelul normal pentru a genera raportul, simularea unei situatii in care nu exista informatii despre comenzile efectuate si simularea unei situatii in care nu exista informatii despre produse

CREATE OR REPLACE FUNCTION RaportComenziClientiProduse(p\_simulare\_exception IN VARCHAR2 DEFAULT NULL) RETURN VARCHAR2 IS

TYPE RaportType IS TABLE OF VARCHAR2(4000);

v\_raport RaportType := RaportType();

COMENZI\_INEXISTENTE EXCEPTION;

PRODUSE\_INEXISTENTE EXCEPTION;

BEGIN

IF p\_simulare\_exception = 'PRODUSE\_INEXISTENTE' THEN

RAISE PRODUSE\_INEXISTENTE;

END IF;

IF p\_simulare\_exception = 'COMENZI\_INEXISTENTE' THEN

RAISE COMENZI\_INEXISTENTE;

END IF;

FOR rec IN (

SELECT

c.ID\_Comanda, c.DataComanda, c.StareComanda, cl.Nume, cl.Prenume,

p.NumeProdus, p.Categorie

FROM

Comenzi c

JOIN Clienti cl ON c.ID\_Client = cl.ID\_Client

JOIN Stoc s ON c.ID\_Comanda = s.ID\_Comanda

JOIN Produse p ON s.ID\_Produs = p.ID\_Produs

) LOOP

v\_raport.EXTEND;

v\_raport(v\_raport.LAST) := 'Comanda: ' || rec.ID\_Comanda || ', Data: ' || rec.DataComanda ||

', Stare: ' || rec.StareComanda || ', Client: ' || rec.Nume || ' ' || rec.Prenume ||

', Produs: ' || rec.NumeProdus || ', Categorie: ' || rec.Categorie;

END LOOP;

IF v\_raport.COUNT = 0 THEN

RAISE COMENZI\_INEXISTENTE;

ELSIF p\_simulare\_exception = 'PRODUSE\_INEXISTENTE' THEN

RAISE PRODUSE\_INEXISTENTE;

END IF;

FOR i IN 1..v\_raport.COUNT LOOP

DBMS\_OUTPUT.PUT\_LINE(v\_raport(i));

END LOOP;

RETURN 'Raport generat cu succes.';

EXCEPTION

WHEN COMENZI\_INEXISTENTE THEN

RETURN 'Nu exista informatii despre comenzile efectuate.';

WHEN PRODUSE\_INEXISTENTE THEN

RETURN 'Nu exista informatii despre produse.';

END RaportComenziClientiProduse;

/

DECLARE

rezultat VARCHAR2(4000);

BEGIN

rezultat := RaportComenziClientiProduse;

DBMS\_OUTPUT.PUT\_LINE(rezultat);

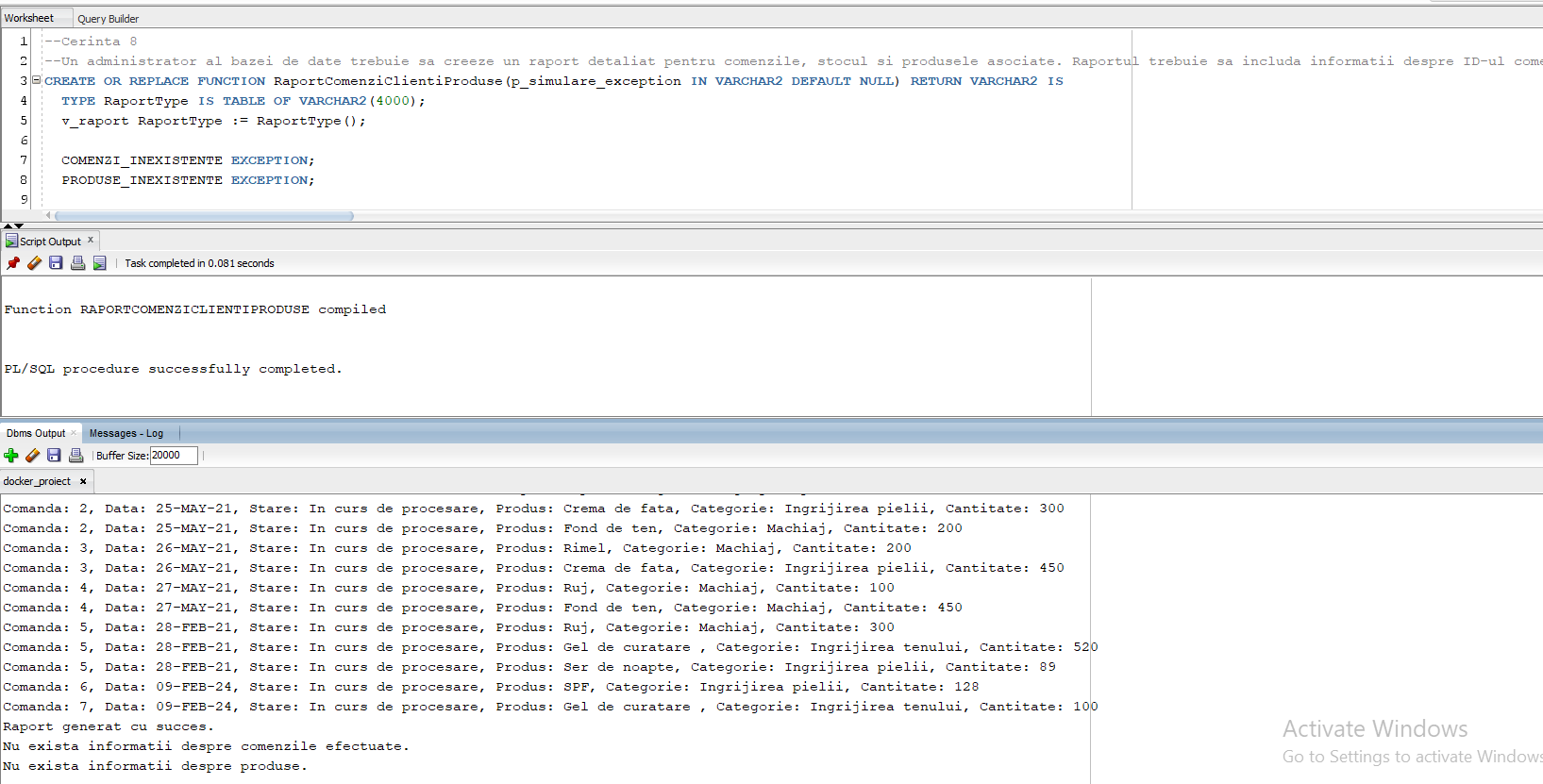
rezultat := RaportComenziClientiProduse('COMENZI\_INEXISTENTE');

DBMS\_OUTPUT.PUT\_LINE(rezultat);

rezultat := RaportComenziClientiProduse('PRODUSE\_INEXISTENTE');

DBMS\_OUTPUT.PUT\_LINE(rezultat);

END;

/

Cerinta 9

Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat independent de tip procedură care să utilizeze într-o singură comandă SQL 5 dintre tabelele definite. Tratați toate excepțiile care pot apărea, incluzând excepțiile NO\_DATA\_FOUND și TOO\_MANY\_ROWS. Apelați subprogramul astfel încât să evidențiați toate cazurile tratate

Creeaza o procedura stocata in baza de date care sa genereze un raport de analiza utilizand date din cinci tabele definite: Comenzi, Clienti, Stoc, Produse si CercetareDezvoltare. Raportul trebuie sa includa informatii precum valoarea totala a vanzarilor, numarul total de clienti, media preturilor de vanzare, totalul cantitatii din stoc si media ratei de reusita in cercetare. Procedura trebuie sa gestioneze corect toate situatiile, inclusiv cazurile in care nu exista date disponibile sau sunt prea multe randuri de date. Pentru a evidenția functionarea corecta a procedurii, aceasta trebuie apelata in trei scenarii diferite: un apel normal pentru a genera raportul, un apel pentru a simula lipsa de date si unul pentru a simula prezenta unui numar prea mare de randuri de date.

CREATE OR REPLACE PROCEDURE RaportAnaliza(p\_simulare\_scenariu IN VARCHAR2 DEFAULT NULL) AS

v\_total\_valoare\_vanzari DECIMAL(10, 2);

v\_numar\_clienti INT;

v\_medie\_pret\_vanzare DECIMAL(10, 2);

v\_total\_cantitate\_stoc INT;

v\_medie\_reusita\_cercetare DECIMAL(5, 2);

PROCEDURE SimulareNoData IS

BEGIN

RAISE NO\_DATA\_FOUND;

END;

PROCEDURE SimulareTooManyRows IS

BEGIN

RAISE TOO\_MANY\_ROWS;

END;

BEGIN

IF p\_simulare\_scenariu = 'NO\_DATA' THEN

SimulareNoData;

ELSIF p\_simulare\_scenariu = 'TOO\_MANY\_ROWS' THEN

SimulareTooManyRows;

END IF;

SELECT

COALESCE(SUM(p.PretVanzare \* s.Cantitate), 0),

COUNT(DISTINCT cl.ID\_Client),

COALESCE(AVG(p.PretVanzare), 0),

SUM(s.Cantitate),

COALESCE(AVG(cd.RataReusita), 0)

INTO

v\_total\_valoare\_vanzari,

v\_numar\_clienti,

v\_medie\_pret\_vanzare,

v\_total\_cantitate\_stoc,

v\_medie\_reusita\_cercetare

FROM

Comenzi c

JOIN Clienti cl ON c.ID\_Client = cl.ID\_Client

JOIN Stoc s ON c.ID\_Comanda = s.ID\_Comanda

JOIN Produse p ON p.ID\_Produs = s.ID\_Produs

LEFT JOIN CercetareDezvoltare cd ON cl.ID\_Client = cd.ID\_Reprezentant;

DBMS\_OUTPUT.PUT\_LINE('Total valoare vanzari: ' || v\_total\_valoare\_vanzari);

DBMS\_OUTPUT.PUT\_LINE('Numar total de clienti: ' || v\_numar\_clienti);

DBMS\_OUTPUT.PUT\_LINE('Media preturilor de vanzare: ' || v\_medie\_pret\_vanzare);

DBMS\_OUTPUT.PUT\_LINE('Total cantitate in stoc: ' || v\_total\_cantitate\_stoc);

DBMS\_OUTPUT.PUT\_LINE('Media ratei de reusita in cercetare: ' || v\_medie\_reusita\_cercetare);

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Nu s-au gasit date pentru raport.');

WHEN TOO\_MANY\_ROWS THEN

DBMS\_OUTPUT.PUT\_LINE('Prea multe randuri gasite pentru raport.');

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('O eroare neasteptata a aparut.');

END RaportAnaliza;

/

DECLARE

rezultat VARCHAR2(2000);

BEGIN

RaportAnaliza;

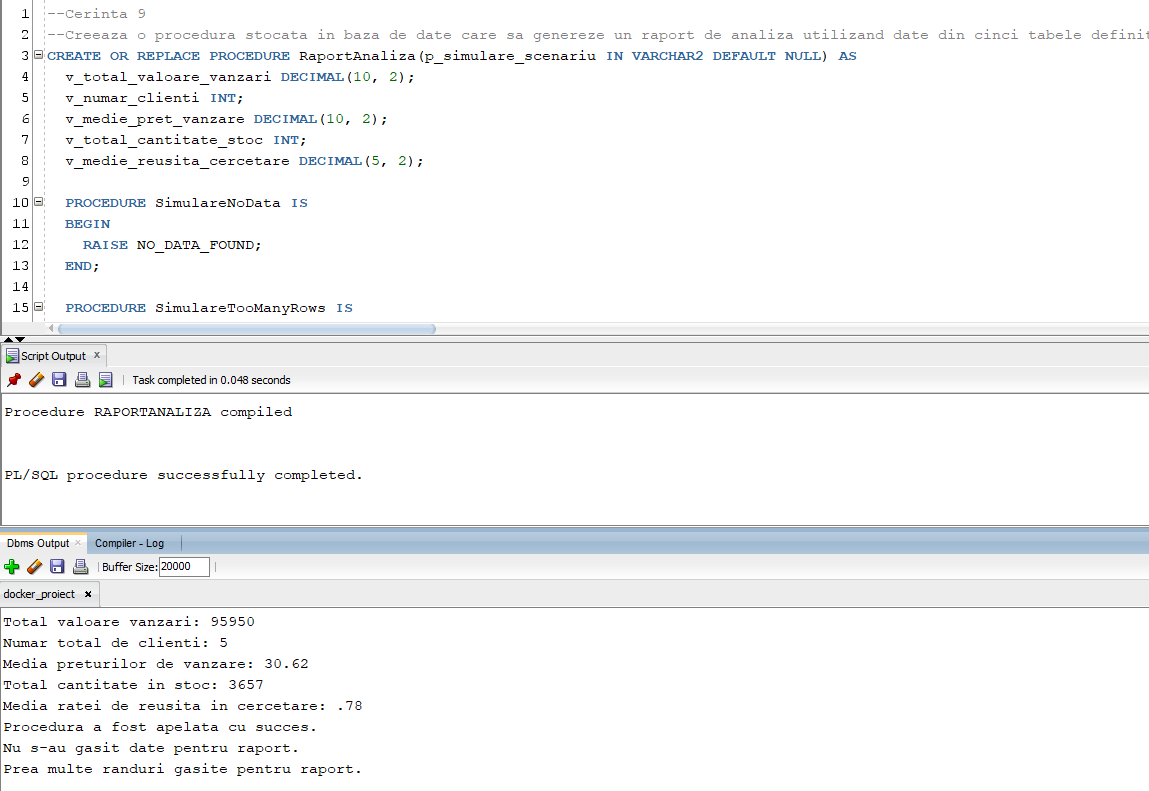
DBMS\_OUTPUT.PUT\_LINE('Procedura a fost apelata cu succes.');

RaportAnaliza ('NO\_DATA');

RaportAnaliza ('TOO\_MANY\_ROWS');

END;

/



Cerinta 10

Definiți un trigger de tip LMD la nivel de comandă. Declanșați trigger-ul. ( rularea a fost facuta dupa crearea triggerului de la cerinta 11 )

CREATE OR REPLACE TRIGGER trg\_update\_comenzi

AFTER INSERT OR UPDATE ON Comenzi

FOR EACH ROW

DECLARE

v\_mesaj VARCHAR2(100);

BEGIN

IF INSERTING THEN

v\_mesaj := 'A fost inserat un nou rand în tabela Comenzi.';

ELSIF UPDATING THEN

v\_mesaj := 'Un rand existent în tabela Comenzi a fost actualizat.';

END IF;

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

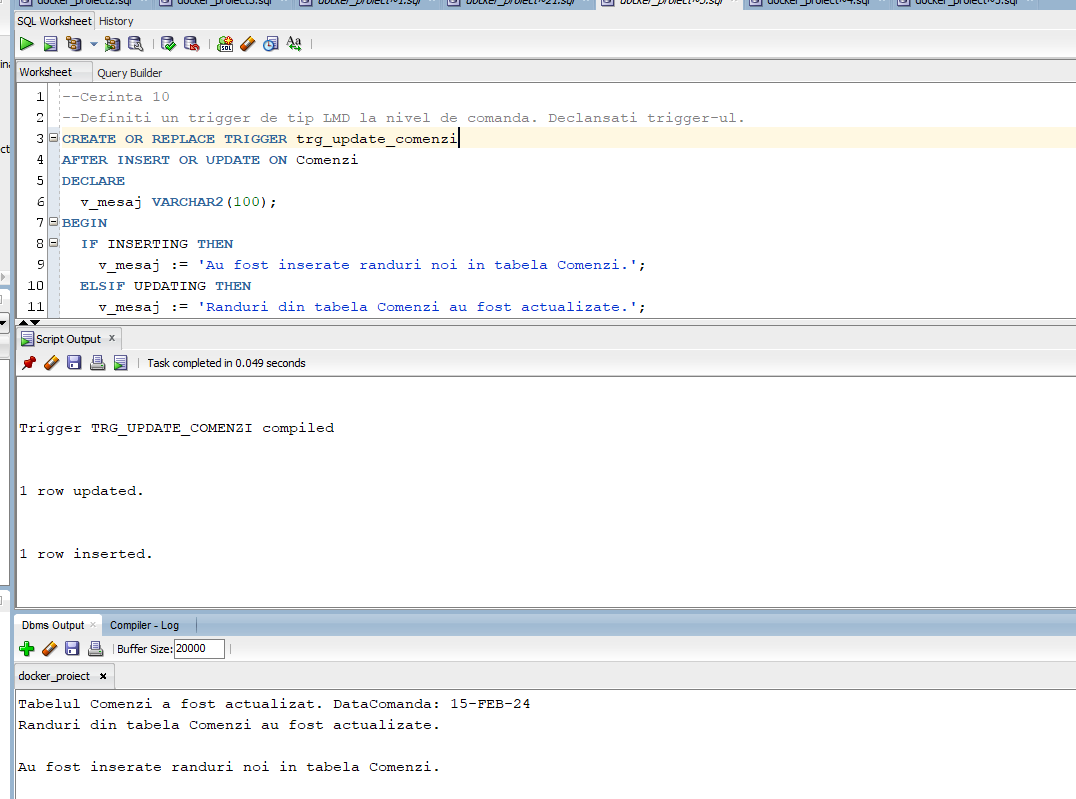
END trg\_update\_comenzi;

/

INSERT INTO Comenzi (ID\_Comanda, ID\_Client, DataComanda, ID\_Stoc, StareComanda)

VALUES (20, 5, SYSDATE, 104, 'In curs de procesare');

UPDATE Comenzi SET StareComanda = 'Procesata' WHERE ID\_Comanda = 1;



Cerinta 11

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

CREATE OR REPLACE TRIGGER trg\_update\_comenzi1

BEFORE UPDATE ON Comenzi

FOR EACH ROW

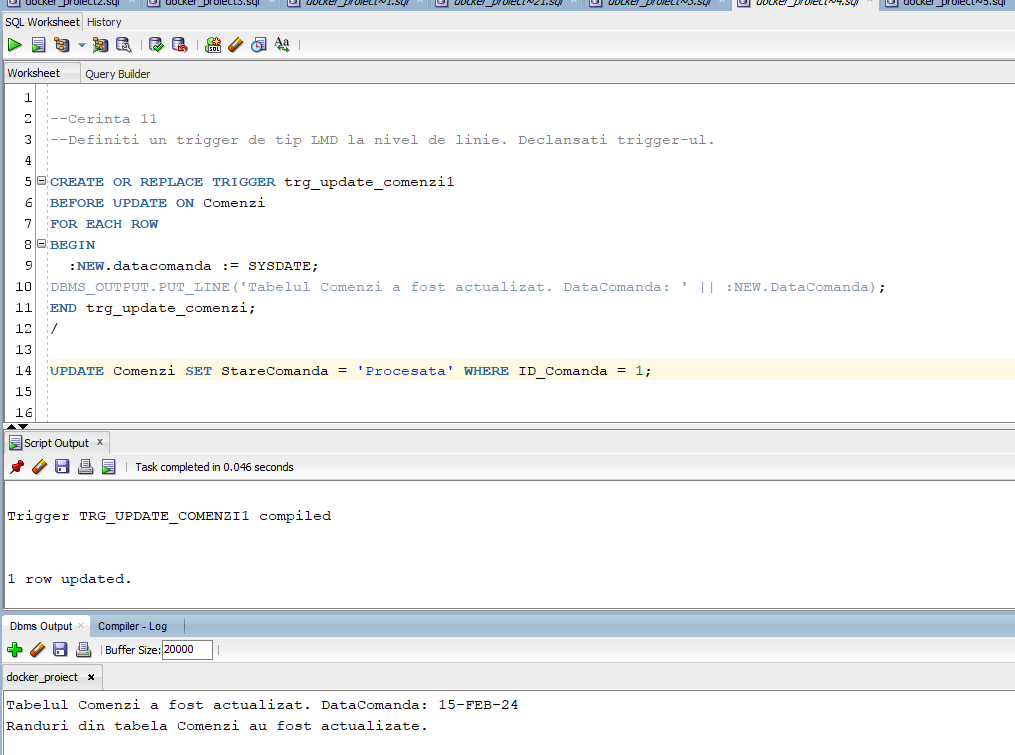
BEGIN

:NEW.datacomanda := SYSDATE;

DBMS\_OUTPUT.PUT\_LINE('Tabelul Comenzi a fost actualizat. DataComanda: ' || :NEW.DataComanda);

END trg\_update\_comenzi;

/



Cerinta 12

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

CREATE TABLE audit\_events (

utilizator VARCHAR2(30),

nume\_bd VARCHAR2(50),

eveniment VARCHAR2(20),

nume\_obiect VARCHAR2(30),

data DATE

);

CREATE OR REPLACE TRIGGER trg\_audit\_events

AFTER CREATE OR DROP OR ALTER ON SCHEMA

BEGIN

INSERT INTO audit\_events

VALUES (SYS.LOGIN\_USER, SYS.DATABASE\_NAME, SYS.SYSEVENT,

SYS.DICTIONARY\_OBJ\_NAME, SYSDATE);

END;

/

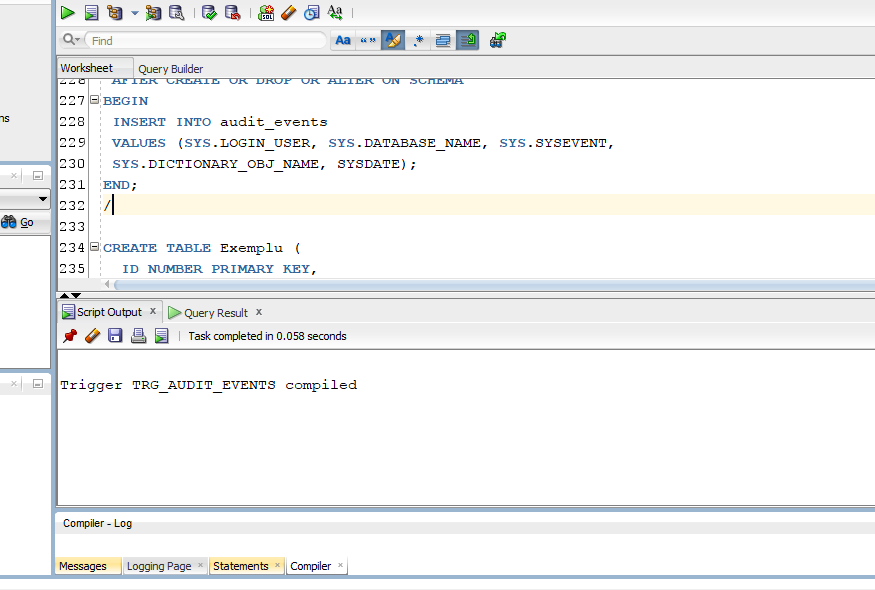
CREATE TABLE Exemplu (

ID NUMBER PRIMARY KEY,

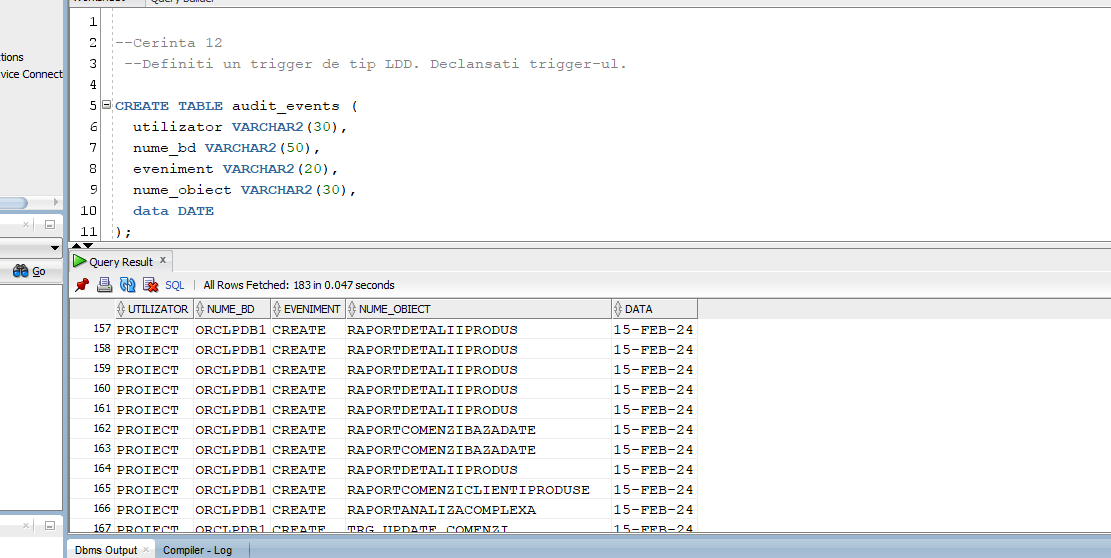
Nume VARCHAR2(50)

);

drop table exemplu;

SELECT \* FROM audit\_events;

Tabelul audit\_events :



Cerinta 13

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

CREATE OR REPLACE PACKAGE ProiectPachet AS

PROCEDURE RaportDetaliiProdus(p\_ID\_Produs IN INT);

PROCEDURE RaportComenziBazaDate;

FUNCTION RaportComenziClientiProduse(p\_simulare\_exception IN VARCHAR2 DEFAULT NULL) RETURN VARCHAR2;

PROCEDURE RaportAnaliza(p\_simulare\_scenariu IN VARCHAR2 DEFAULT NULL);

END ProiectPachet;

/

CREATE OR REPLACE PACKAGE BODY ProiectPachet AS

--Cerinta 6

PROCEDURE RaportDetaliiProdus(p\_ID\_Produs IN INT) AS

TYPE ProdusInfo IS RECORD (

NumeProdus VARCHAR2(100),

Categorie VARCHAR2(50),

PretVanzare DECIMAL(10, 2)

);

TYPE StocInfo IS RECORD (

ID\_Stoc INT,

Cantitate INT,

ID\_Comanda INT

);

TYPE StocList IS TABLE OF StocInfo INDEX BY PLS\_INTEGER;

TYPE ReclamaType IS RECORD (

ID\_Reclama INT,

DataReclama DATE

);

TYPE ReclamaList IS TABLE OF ReclamaType;

TYPE CategorieList IS VARRAY(50) OF VARCHAR2(50);

produs\_info ProdusInfo;

stoc\_list StocList;

reclama\_list ReclamaList := ReclamaList();

categorie\_list CategorieList := CategorieList();

idx INT := 1;

BEGIN

SELECT NumeProdus, Categorie, PretVanzare

INTO produs\_info.NumeProdus, produs\_info.Categorie, produs\_info.PretVanzare

FROM Produse

WHERE ID\_Produs = p\_ID\_Produs;

SELECT ID\_Stoc, Cantitate, ID\_Comanda

BULK COLLECT INTO stoc\_list

FROM Stoc

WHERE ID\_Produs = p\_ID\_Produs;

SELECT ID\_Reclama, DataReclama

BULK COLLECT INTO reclama\_list

FROM Reclama

WHERE ID\_Produs = p\_ID\_Produs;

SELECT DISTINCT Categorie

BULK COLLECT INTO categorie\_list

FROM Produse

WHERE ROWNUM <= 50;

DBMS\_OUTPUT.PUT\_LINE('Raport pentru produsul: ' || produs\_info.NumeProdus);

DBMS\_OUTPUT.PUT\_LINE('Categorie: ' || produs\_info.Categorie || ', Pret de Vanzare: ' || produs\_info.PretVanzare);

DBMS\_OUTPUT.PUT\_LINE('Stocuri:');

FOR i IN stoc\_list.FIRST..stoc\_list.LAST LOOP

IF stoc\_list.EXISTS(i) THEN

DBMS\_OUTPUT.PUT\_LINE('ID Stoc: ' || stoc\_list(i).ID\_Stoc || ', Cantitate: ' || stoc\_list(i).Cantitate || ', ID Comanda: ' || stoc\_list(i).ID\_Comanda);

END IF;

END LOOP;

DBMS\_OUTPUT.PUT\_LINE('Reclame asociate:');

FOR i IN 1..reclama\_list.COUNT LOOP

DBMS\_OUTPUT.PUT\_LINE('ID Reclama: ' || reclama\_list(i).ID\_Reclama || ', Data: ' || TO\_CHAR(reclama\_list(i).DataReclama, 'DD-MM-YYYY'));

END LOOP;

DBMS\_OUTPUT.PUT\_LINE('Categorii de produse disponibile:');

FOR i IN 1..categorie\_list.COUNT LOOP

DBMS\_OUTPUT.PUT\_LINE(categorie\_list(i));

END LOOP;

END RaportDetaliiProdus;

-- Cerinta 7

PROCEDURE RaportComenziBazaDate AS

CURSOR cursorComenzi IS

SELECT c.ID\_Comanda, c.DataComanda, c.StareComanda, cl.Nume

FROM Comenzi c

JOIN Clienti cl ON c.ID\_Client = cl.ID\_Client;

BEGIN

FOR comanda\_rec IN cursorComenzi LOOP

DBMS\_OUTPUT.PUT\_LINE('ID Comanda: ' || comanda\_rec.ID\_Comanda || ', Data: ' || comanda\_rec.DataComanda || ', Stare: ' || comanda\_rec.StareComanda || ', Nume Client: ' || comanda\_rec.Nume);

DECLARE

CURSOR cursorDetaliiComanda(p\_ID\_Comanda INT) IS

SELECT s.ID\_Stoc, s.Cantitate, p.NumeProdus, p.PretVanzare

FROM Stoc s

JOIN Produse p ON s.ID\_Produs = p.ID\_Produs

WHERE s.ID\_Comanda = p\_ID\_Comanda;

BEGIN

FOR detalii\_rec IN cursorDetaliiComanda(comanda\_rec.ID\_Comanda) LOOP

DBMS\_OUTPUT.PUT\_LINE(' Detalii Stoc: ID\_Stoc ' || detalii\_rec.ID\_Stoc || ', Produs: ' || detalii\_rec.NumeProdus || ', Cantitate: ' || detalii\_rec.Cantitate || ', Pret: ' || detalii\_rec.PretVanzare);

END LOOP;

END;

END LOOP;

END RaportComenziBazaDate;

-- Cerinta 8

FUNCTION RaportComenziClientiProduse(p\_simulare\_exception IN VARCHAR2 DEFAULT NULL) RETURN VARCHAR2 IS

TYPE RaportType IS TABLE OF VARCHAR2(4000);

v\_raport RaportType := RaportType();

COMENZI\_INEXISTENTE EXCEPTION;

PRODUSE\_INEXISTENTE EXCEPTION;

BEGIN

IF p\_simulare\_exception = 'PRODUSE\_INEXISTENTE' THEN

RAISE PRODUSE\_INEXISTENTE;

END IF;

IF p\_simulare\_exception = 'COMENZI\_INEXISTENTE' THEN

RAISE COMENZI\_INEXISTENTE;

END IF;

FOR rec IN (

SELECT

c.ID\_Comanda, c.DataComanda, c.StareComanda, cl.Nume, cl.Prenume,

p.NumeProdus, p.Categorie

FROM

Comenzi c

JOIN Clienti cl ON c.ID\_Client = cl.ID\_Client

JOIN Stoc s ON c.ID\_Comanda = s.ID\_Comanda

JOIN Produse p ON s.ID\_Produs = p.ID\_Produs

) LOOP

v\_raport.EXTEND;

v\_raport(v\_raport.LAST) := 'Comanda: ' || rec.ID\_Comanda || ', Data: ' || rec.DataComanda ||

', Stare: ' || rec.StareComanda || ', Client: ' || rec.Nume || ' ' || rec.Prenume ||

', Produs: ' || rec.NumeProdus || ', Categorie: ' || rec.Categorie;

END LOOP;

IF v\_raport.COUNT = 0 THEN

RAISE COMENZI\_INEXISTENTE;

ELSIF p\_simulare\_exception = 'PRODUSE\_INEXISTENTE' THEN

RAISE PRODUSE\_INEXISTENTE;

END IF;

FOR i IN 1..v\_raport.COUNT LOOP

DBMS\_OUTPUT.PUT\_LINE(v\_raport(i));

END LOOP;

RETURN 'Raport generat cu succes.';

EXCEPTION

WHEN COMENZI\_INEXISTENTE THEN

RETURN 'Nu exista informatii despre comenzile efectuate.';

WHEN PRODUSE\_INEXISTENTE THEN

RETURN 'Nu exista informatii despre produse.';

END RaportComenziClientiProduse;

-- Cerinta 9

PROCEDURE RaportAnaliza(p\_simulare\_scenariu IN VARCHAR2 DEFAULT NULL) AS

v\_total\_valoare\_vanzari DECIMAL(10, 2);

v\_numar\_clienti INT;

v\_medie\_pret\_vanzare DECIMAL(10, 2);

v\_total\_cantitate\_stoc INT;

v\_medie\_reusita\_cercetare DECIMAL(5, 2);

PROCEDURE SimulareNoData IS

BEGIN

RAISE NO\_DATA\_FOUND;

END;

PROCEDURE SimulareTooManyRows IS

BEGIN

RAISE TOO\_MANY\_ROWS;

END;

BEGIN

IF p\_simulare\_scenariu = 'NO\_DATA' THEN

SimulareNoData;

ELSIF p\_simulare\_scenariu = 'TOO\_MANY\_ROWS' THEN

SimulareTooManyRows;

END IF;

SELECT

COALESCE(SUM(p.PretVanzare \* s.Cantitate), 0),

COUNT(DISTINCT cl.ID\_Client),

COALESCE(AVG(p.PretVanzare), 0),

SUM(s.Cantitate),

COALESCE(AVG(cd.RataReusita), 0)

INTO

v\_total\_valoare\_vanzari,

v\_numar\_clienti,

v\_medie\_pret\_vanzare,

v\_total\_cantitate\_stoc,

v\_medie\_reusita\_cercetare

FROM

Comenzi c

JOIN Clienti cl ON c.ID\_Client = cl.ID\_Client

JOIN Stoc s ON c.ID\_Comanda = s.ID\_Comanda

JOIN Produse p ON p.ID\_Produs = s.ID\_Produs

LEFT JOIN CercetareDezvoltare cd ON cl.ID\_Client = cd.ID\_Reprezentant;

DBMS\_OUTPUT.PUT\_LINE('Total valoare vanzari: ' || v\_total\_valoare\_vanzari);

DBMS\_OUTPUT.PUT\_LINE('Numar total de clienti: ' || v\_numar\_clienti);

DBMS\_OUTPUT.PUT\_LINE('Media preturilor de vanzare: ' || v\_medie\_pret\_vanzare);

DBMS\_OUTPUT.PUT\_LINE('Total cantitate in stoc: ' || v\_total\_cantitate\_stoc);

DBMS\_OUTPUT.PUT\_LINE('Media ratei de reusita in cercetare: ' || v\_medie\_reusita\_cercetare);

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Nu s-au gasit date pentru raport.');

WHEN TOO\_MANY\_ROWS THEN

DBMS\_OUTPUT.PUT\_LINE('Prea multe randuri gasite pentru raport.');

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('O eroare neasteptata a aparut.');

END RaportAnaliza;

END ProiectPachet;

/

BEGIN

ProiectPachet.RaportDetaliiProdus(1);

END;

/

BEGIN

ProiectPachet.RaportComenziBazaDate;

END;

/

DECLARE

v\_raport VARCHAR2(4000);

BEGIN

v\_raport := ProiectPachet.RaportComenziClientiProduse;

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

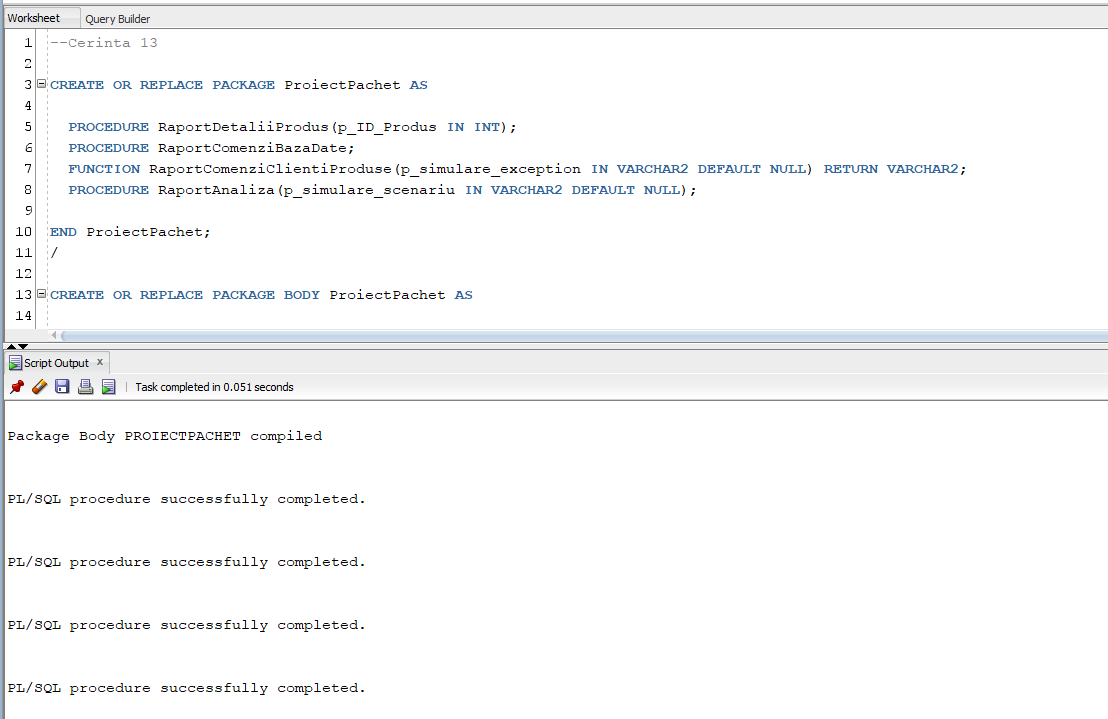
END;

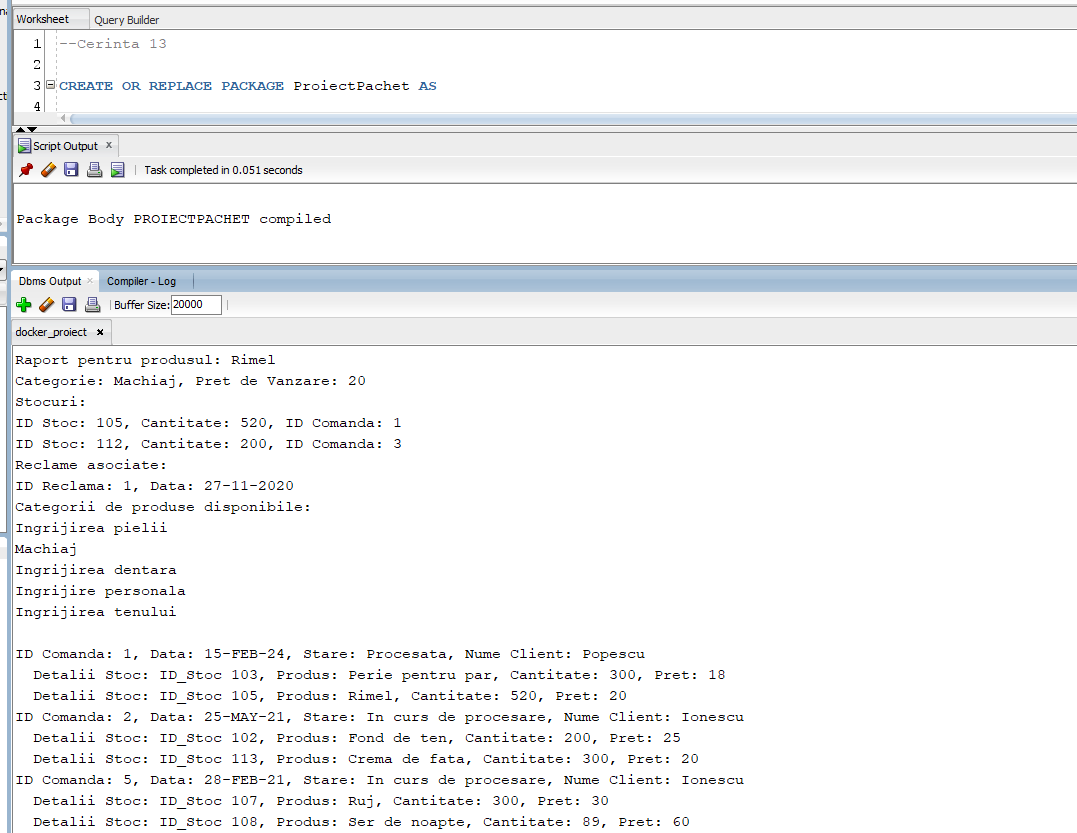
/

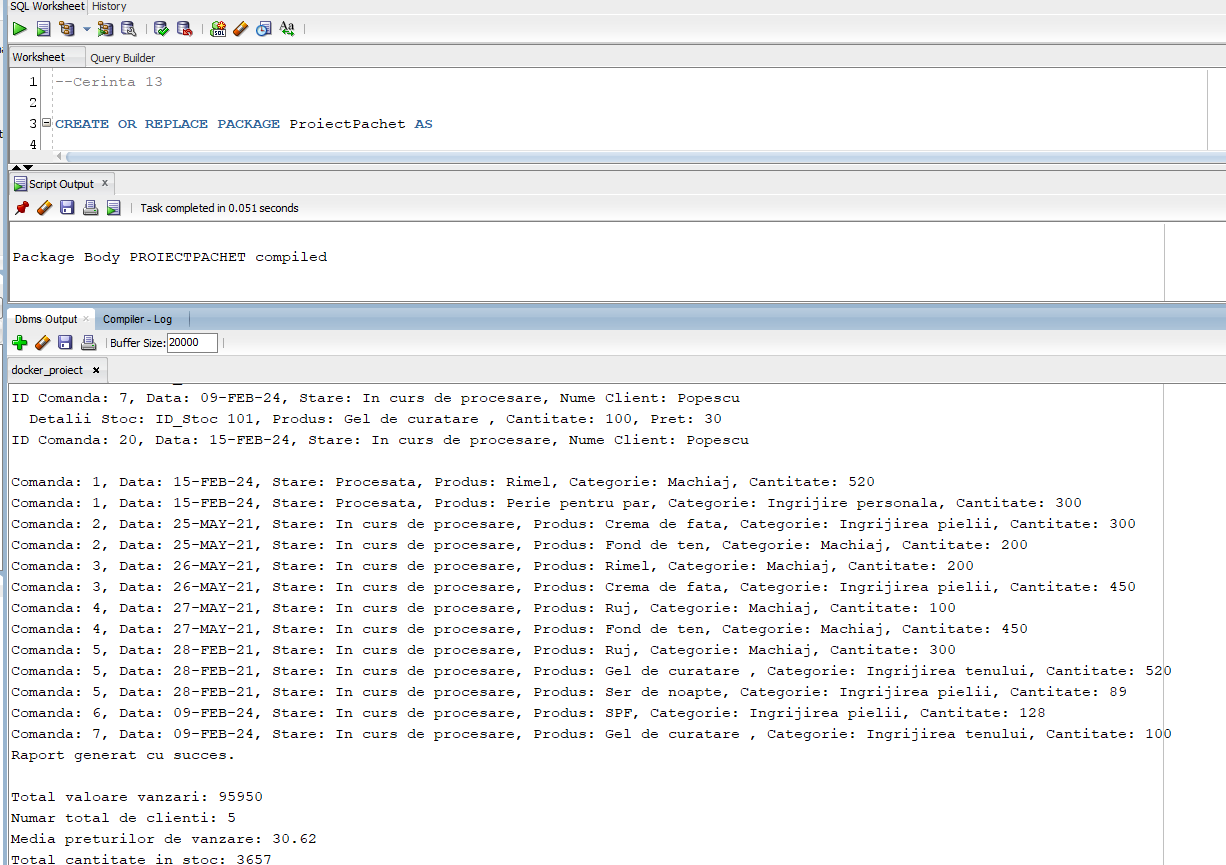
BEGIN

ProiectPachet.RaportAnaliza;

END;

/





Cerinta 14

14. Definiți un pachet care să includă tipuri de date complexe și obiecte necesare unui flux de acțiuni integrate, specifice bazei de date definite (minim 2 tipuri de date, minim 2 funcții, minim 2 proceduri).

CREATE OR REPLACE PACKAGE FluxComenzi AS

TYPE ComandaType IS RECORD (

ID\_Comanda Comenzi.ID\_Comanda%TYPE,

DataComanda Comenzi.DataComanda%TYPE,

StareComanda Comenzi.StareComanda%TYPE

);

TYPE ClientType IS RECORD (

ID\_Client Clienti.ID\_Client%TYPE,

Nume Clienti.Nume%TYPE,

Prenume Clienti.Prenume%TYPE,

Adresa Clienti.Adresa%TYPE,

NrTelefon Clienti.NrTelefon%TYPE,

Email Clienti.Email%TYPE

);

PROCEDURE AfisareDetaliiComanda(p\_ID\_Comanda IN Comenzi.ID\_Comanda%TYPE);

FUNCTION ObtinereDetaliiClient(p\_ID\_Client IN Clienti.ID\_Client%TYPE) RETURN ClientType;

PROCEDURE InserareComanda(

p\_ID\_Comanda IN Comenzi.ID\_Comanda%TYPE,

p\_ID\_Client IN Clienti.ID\_Client%TYPE,

p\_DataComanda IN Comenzi.DataComanda%TYPE,

p\_StareComanda IN Comenzi.StareComanda%TYPE

);

PROCEDURE NumarTotalComenziClient(

p\_ID\_Client IN Clienti.ID\_Client%TYPE,

p\_NumarComenzi OUT NUMBER

);

END FluxComenzi;

/

CREATE OR REPLACE PACKAGE BODY FluxComenzi AS

PROCEDURE AfisareDetaliiComanda(p\_ID\_Comanda IN Comenzi.ID\_Comanda%TYPE) AS

v\_Comanda ComandaType;

BEGIN

SELECT c.ID\_Comanda, c.DataComanda, c.StareComanda

INTO v\_Comanda

FROM Comenzi c

WHERE c.ID\_Comanda = p\_ID\_Comanda;

DBMS\_OUTPUT.PUT\_LINE('Detalii comanda:');

DBMS\_OUTPUT.PUT\_LINE('ID Comanda: ' || v\_Comanda.ID\_Comanda);

DBMS\_OUTPUT.PUT\_LINE('Data Comanda: ' || TO\_CHAR(v\_Comanda.DataComanda, 'DD-MM-YYYY'));

DBMS\_OUTPUT.PUT\_LINE('Stare Comanda: ' || v\_Comanda.StareComanda);

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Comanda cu ID-ul ' || p\_ID\_Comanda || ' nu exista.');

END AfisareDetaliiComanda;

FUNCTION ObtinereDetaliiClient(p\_ID\_Client IN Clienti.ID\_Client%TYPE) RETURN ClientType AS

v\_Client ClientType;

BEGIN

SELECT ID\_Client, Nume, Prenume, Adresa, NrTelefon, Email

INTO v\_Client

FROM Clienti

WHERE ID\_Client = p\_ID\_Client;

RETURN v\_Client;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

RETURN NULL;

END ObtinereDetaliiClient;

PROCEDURE InserareComanda(

p\_ID\_Comanda IN Comenzi.ID\_Comanda%TYPE,

p\_ID\_Client IN Clienti.ID\_Client%TYPE,

p\_DataComanda IN Comenzi.DataComanda%TYPE,

p\_StareComanda IN Comenzi.StareComanda%TYPE

) AS

BEGIN

INSERT INTO Comenzi (ID\_Comanda, ID\_Client, DataComanda, StareComanda)

VALUES (p\_ID\_Comanda, p\_ID\_Client, p\_DataComanda, p\_StareComanda);

DBMS\_OUTPUT.PUT\_LINE('Comanda cu ID-ul ' || p\_ID\_Comanda || ' a fost inserata cu succes.');

END InserareComanda;

PROCEDURE NumarTotalComenziClient(

p\_ID\_Client IN Clienti.ID\_Client%TYPE,

p\_NumarComenzi OUT NUMBER

) AS

BEGIN

SELECT COUNT(\*)

INTO p\_NumarComenzi

FROM Comenzi

WHERE ID\_Client = p\_ID\_Client;

END NumarTotalComenziClient;

END FluxComenzi;

/

DECLARE

v\_ID\_Comanda Comenzi.ID\_Comanda%TYPE := 4;

BEGIN

FluxComenzi.AfisareDetaliiComanda(v\_ID\_Comanda);

END;

/

DECLARE

v\_ID\_Client Clienti.ID\_Client%TYPE := 3;

v\_Client FluxComenzi.ClientType;

BEGIN

v\_Client := FluxComenzi.ObtinereDetaliiClient(v\_ID\_Client);

IF v\_Client.ID\_Client IS NOT NULL THEN

DBMS\_OUTPUT.PUT\_LINE('Detalii client:');

DBMS\_OUTPUT.PUT\_LINE('ID Client: ' || v\_Client.ID\_Client);

DBMS\_OUTPUT.PUT\_LINE('Nume: ' || v\_Client.Nume);

DBMS\_OUTPUT.PUT\_LINE('Prenume: ' || v\_Client.Prenume);

DBMS\_OUTPUT.PUT\_LINE('Adresa: ' || v\_Client.Adresa);

DBMS\_OUTPUT.PUT\_LINE('Nr Telefon: ' || v\_Client.NrTelefon);

DBMS\_OUTPUT.PUT\_LINE('Email: ' || v\_Client.Email);

ELSE

DBMS\_OUTPUT.PUT\_LINE('Clientul cu ID-ul ' || v\_ID\_Client || ' nu exista.');

END IF;

END;

/

BEGIN

FluxComenzi.InserareComanda(29, 3, SYSDATE, 'Noua');

END;

/

DECLARE

v\_ID\_Client Clienti.ID\_Client%TYPE := 2;

v\_NumarComenzi NUMBER;

BEGIN

FluxComenzi.NumarTotalComenziClient(v\_ID\_Client, v\_NumarComenzi);

DBMS\_OUTPUT.PUT\_LINE('Numar total comenzi pentru clientul cu ID-ul ' || v\_ID\_Client || ': ' || v\_NumarComenzi);

END;

/

