Proiect

My Hospital

Nume: Vasile

Prenume: Daria-Gabriela

Seria: 24

Grupa: 241

Anul Universitar curent: 2024-2025

Contents

[ Tema proiectului: 4](#_Toc186129310)

[ Infrastructura utilizată pentru implementarea: 4](#_Toc186129311)

[1. Prezentați pe scurt baza de date (utilitatea ei). 5](#_Toc186129312)

[ Descrierea modelului 5](#_Toc186129313)

[ Utilitatea modelului 5](#_Toc186129314)

[2. Realizați diagrama entitate-relație (ERD): entitățile, relațiile și atributele trebuie definite în limba română. 6](#_Toc186129315)

[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. 7](#_Toc186129316)

[4. Implementați în Oracle diagrama conceptuală realizată: definiți toate tabelele, adăugând toate constrângerile de integritate necesare (chei primare, cheile externe etc). 8](#_Toc186129317)

[1) Medic 8](#_Toc186129318)

[2) Pacient 8](#_Toc186129319)

[3) Consultatie 9](#_Toc186129320)

[4) Afectiune 9](#_Toc186129321)

[5) Tratament 10](#_Toc186129322)

[6) Afectie\_tratament 10](#_Toc186129323)

[7) Istoric\_Medical 11](#_Toc186129324)

[8) Investigatie 11](#_Toc186129325)

[9) Fisa\_Medicala 12](#_Toc186129326)

[5. Adăugați informații coerente în tabelele create (minim 5 înregistrări pentru fiecare entitate independentă; minim 10 înregistrări pentru fiecare tabelă asociativă). 12](#_Toc186129327)

[1) Medic 12](#_Toc186129328)

[2) Pacient 13](#_Toc186129329)

[3) Consultatie 14](#_Toc186129330)

[4) Afectiune 15](#_Toc186129331)

[5) Tratament 16](#_Toc186129332)

[6) Afectiune\_tratament 17](#_Toc186129333)

[7) Istoric-Medical 18](#_Toc186129334)

[8) Investigatie 19](#_Toc186129335)

[9) Fisa\_Medicala 20](#_Toc186129336)

[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. 22](#_Toc186129337)

[Cernița limbaj natural: 22](#_Toc186129338)

[Apelare subprogram: 22](#_Toc186129339)

[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. 29](#_Toc186129340)

[Cernița limbaj natural: 29](#_Toc186129341)

[Apelare subprogram: 29](#_Toc186129342)

[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 create. Tratați toate excepțiile care pot apărea, incluzând excepțiile predefinite NO\_DATA\_FOUND și TOO\_MANY\_ROWS. Apelați subprogramul astfel încât să evidențiați toate cazurile tratate. 33](#_Toc186129343)

[Cernița limbaj natural: 33](#_Toc186129344)

[Apelare subprogram: 33](#_Toc186129345)

[9. Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat independent de tip procedură care să aibă minim 2 parametri și să utilizeze într-o singură comandă SQL 5 dintre tabelele create. Definiți minim 2 excepții proprii, altele decât cele predefinite la nivel de sistem. Apelați subprogramul astfel încât să evidențiați toate cazurile definite și tratate 36](#_Toc186129346)

[Cernița limbaj natural: 36](#_Toc186129347)

[Apelare subprogram: 37](#_Toc186129348)

[10. Definiți un trigger de tip LMD la nivel de comandă. Declanșați trigger-ul. 42](#_Toc186129349)

[11. Definiți un trigger de tip LMD la nivel de linie.Declanșați trigger-ul. 44](#_Toc186129350)

[12. Defiiți un trigger de tip LDD. Declanșați trigger-ul. 47](#_Toc186129351)

[13. Formulați în limbaj natural o problemă pe care să o rezolvați folosind un pachet care să include tipuri de date complexe și obiecte necesare unui flux de acțiuni integrate, specifice bazei de date. 49](#_Toc186129352)

# Introducere

## Tema proiectului:

* Sistem de gestionare al unui spital

## Infrastructura utilizată pentru implementarea:

* + Versiunea SGBD-ului: • Oracle Database 19c
  + Configurația hardware - RAM alocat: 8 GB
  + Configurația software - sistem de operare: Windows 11
  + Nu am utilizat mașina virtuala

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

## Descrierea modelului

Modelul de date descris reprezintă structura și relațiile între entitățile implicate într-un sistem medical. Acest model acoperă aspecte esențiale ale gestionării datelor mai multor medici, incluzând pacienți, consultații, fișe medicale, istoricul medical, afecțiuni, tratamente și investigații.

## Utilitatea modelului

Acest model de date este esențial pentru gestionarea eficientă a unui sistem medical, oferind următoarele beneficii:

1. *Organizare și accesibilitate:*

Permite organizarea structurată a informațiilor despre medici, pacienți și fișe medicale, facilitând accesul rapid la informații relevante.

1. *Monitorizarea activității:*

Ajută la urmărirea programărilor și consultațiilor realizate de medici, precum și la gestionarea programului acestora.

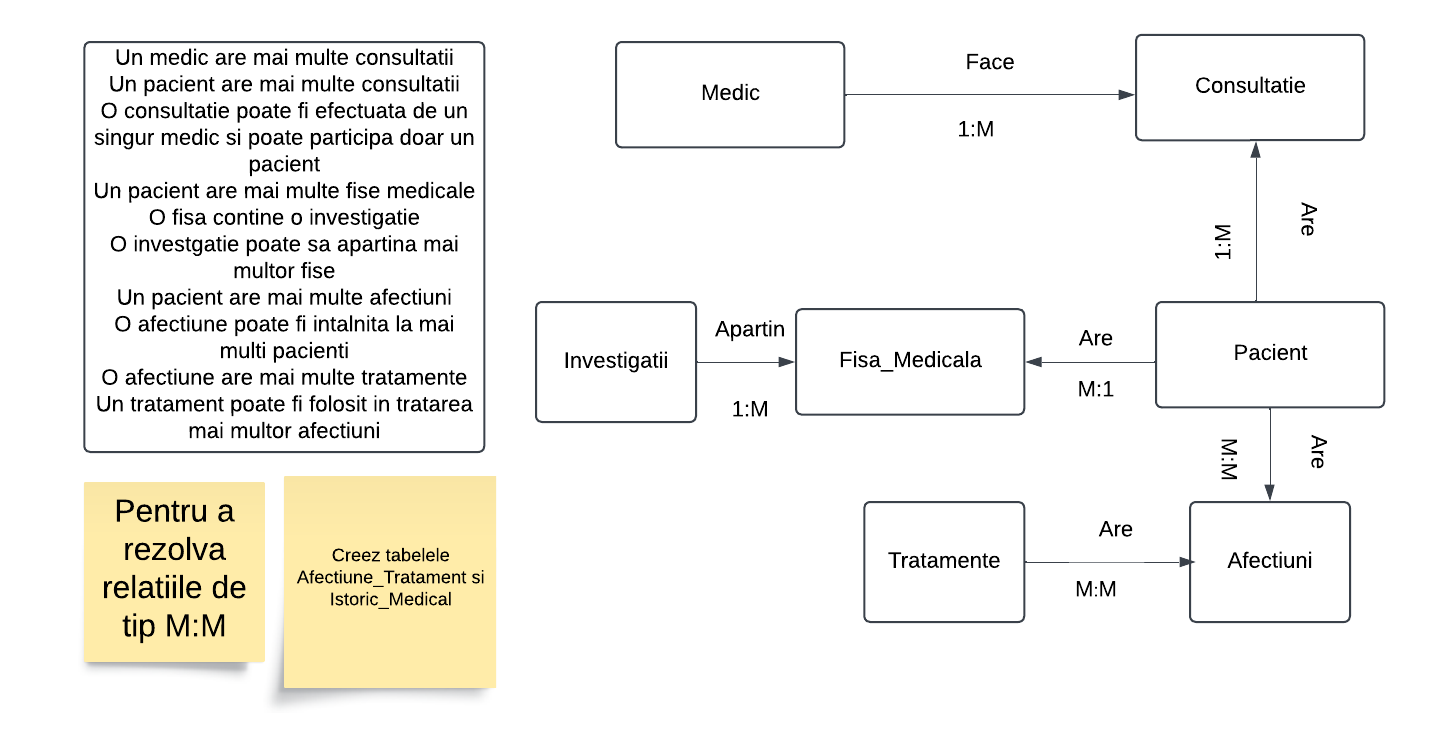
1. *Istoricul medical al pacienților:*

Stochează informații detaliate despre istoricul medical, afecțiuni, tratamente și investigații, oferind o imagine completă a stării de sănătate a pacienților.

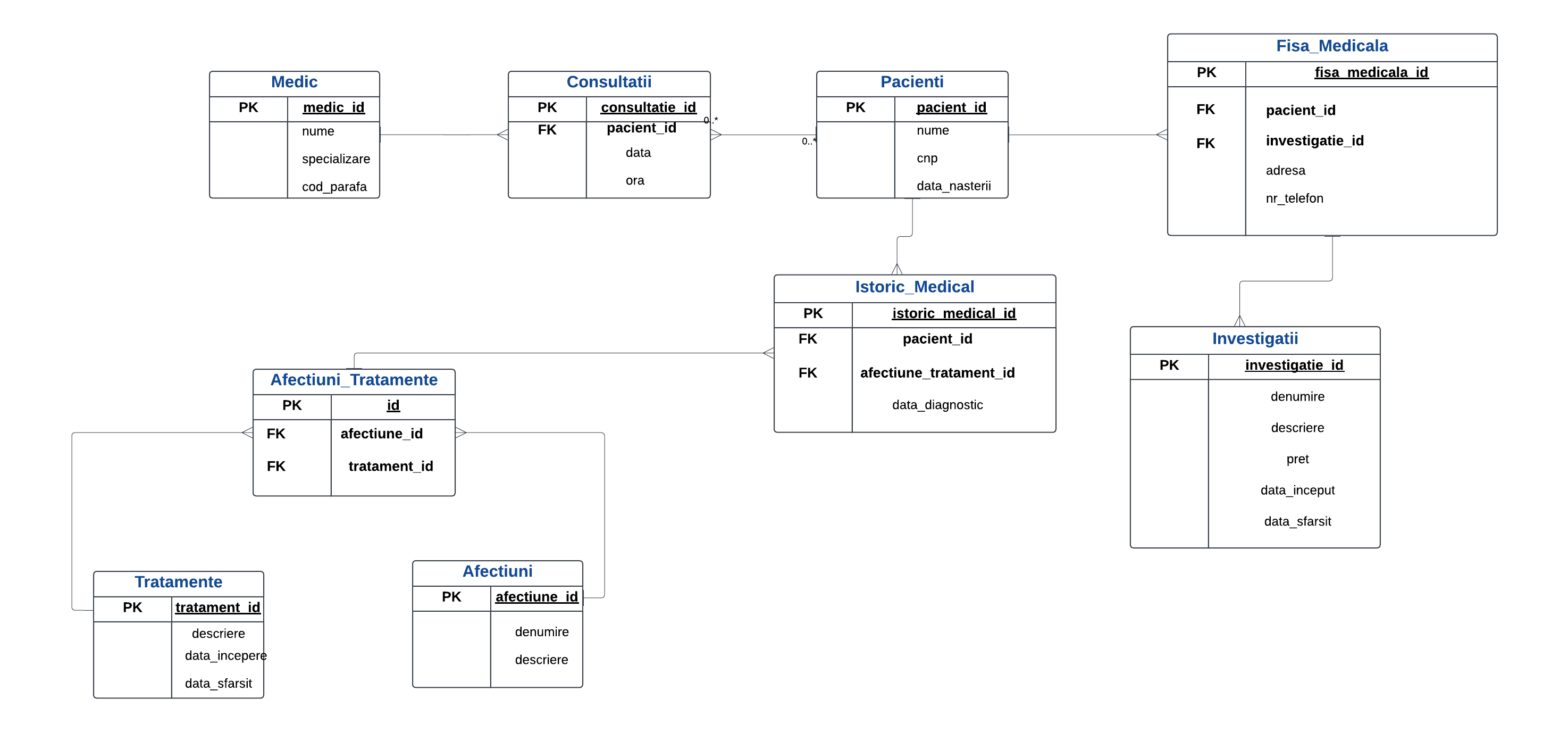
1. *Planificare și alocare:*

Facilitează planificare si alocarea tratamntelor.

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



# 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.



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

## Medic

-- Creare secvente pentru id-urile tabelului Medic

CREATE SEQUENCE medic\_seq START WITH 1 INCREMENT BY 1;

--  Creare tabel Medic

CREATE TABLE Medic (

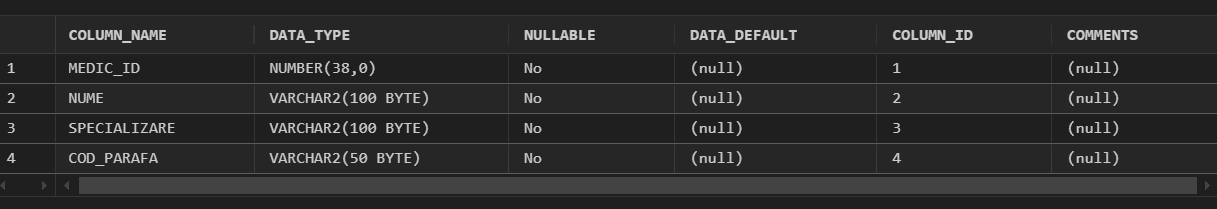
    medic\_id INT PRIMARY KEY,

    nume VARCHAR2(100) NOT NULL,

    specializare VARCHAR2(100) NOT NULL,

    cod\_parafa VARCHAR2(50) UNIQUE NOT NULL

);



## Pacient

CREATE SEQUENCE pacient\_seq START WITH 1 INCREMENT BY 1;

-- Creare tabel Pacient

CREATE TABLE Pacient (

    pacient\_id INT PRIMARY KEY,

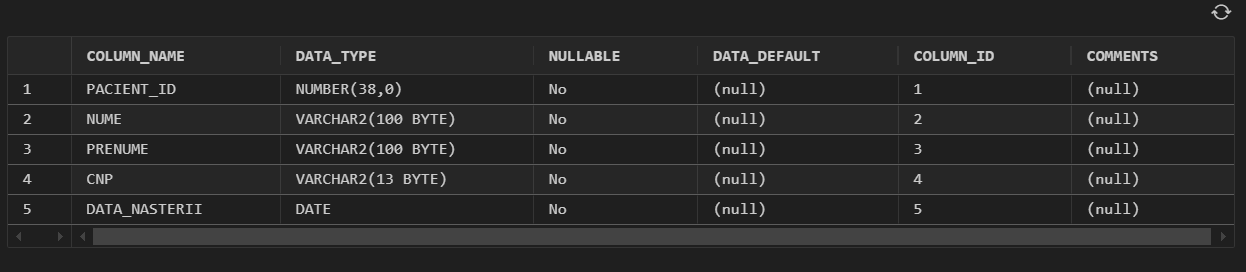
    nume VARCHAR2(100) NOT NULL,

    prenume VARCHAR2(100) NOT NULL,

    cnp VARCHAR2(13) UNIQUE NOT NULL,

    data\_nasterii DATE NOT NULL

);



## Consultatie

-- Creare secvente pentru id-urile tabelului Consultatie

CREATE SEQUENCE consultatie\_seq START WITH 1 INCREMENT BY 1;

-- Create table Consultatie

CREATE TABLE Consultatie (

    consultatie\_id INT PRIMARY KEY,

    medic\_id INT NOT NULL,

    pacient\_id INT NOT NULL,

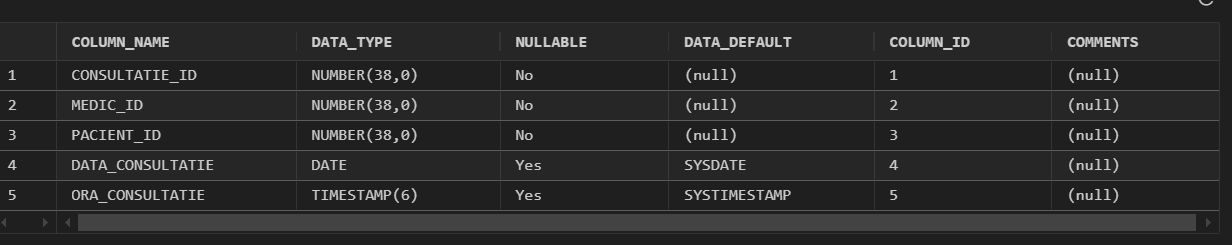
    data\_consultatie DATE DEFAULT SYSDATE,

    ora\_consultatie TIMESTAMP  DEFAULT SYSTIMESTAMP,

    FOREIGN KEY (medic\_id) REFERENCES Medic (medic\_id),

    FOREIGN KEY (pacient\_id) REFERENCES Pacient (pacient\_id)

);



## Afectiune

-- Creare secvente pentru id-urile tabelului  Afecțiune

CREATE SEQUENCE afectiune\_seq START WITH 1 INCREMENT BY 1;

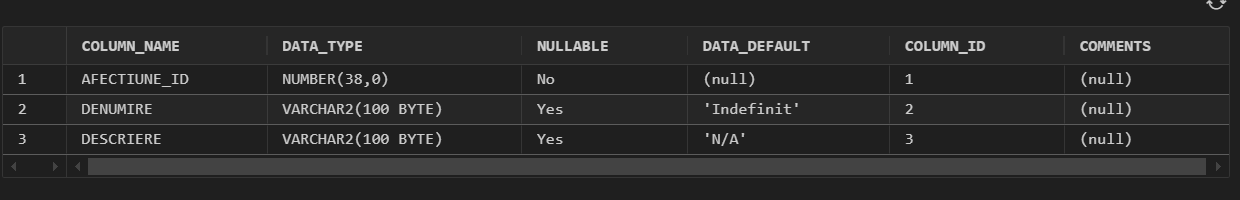
CREATE TABLE Afectiune (

    afectiune\_id INT PRIMARY KEY,

    denumire VARCHAR2(100)  DEFAULT 'Indefinit',

    descriere VARCHAR2(100) DEFAULT 'N/A'

);



## Tratament

-- Creare secvente pentru id-urile tabelului Tratament

CREATE SEQUENCE tratament\_seq START WITH 1 INCREMENT BY 1;

-- Crare table Tratamente

CREATE TABLE Tratament (

    tratament\_id INT PRIMARY KEY,

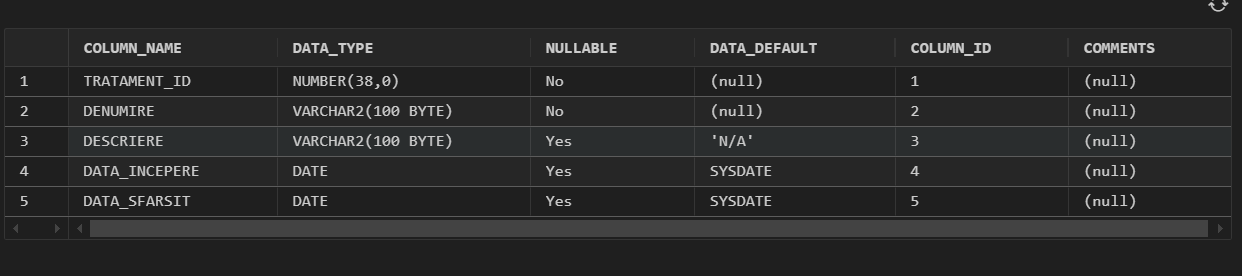
    denumire VARCHAR2(100) NOT NULL,

    descriere VARCHAR2(100) DEFAULT 'N/A',

    data\_incepere DATE DEFAULT SYSDATE,

    data\_sfarsit DATE DEFAULT SYSDATE

);



## Afectie\_tratament

--Creare secvente pentru id-urile tabelului Afectiune\_Tratament

CREATE SEQUENCE afectiune\_tratament\_seq START WITH 1 INCREMENT BY 1;

-- Creare tabel Afectiune\_Tratament

CREATE TABLE Afectiune\_Tratament (

    afectiune\_tratament\_id INT PRIMARY KEY,

    UNIQUE (afectiune\_id, tratament\_id),

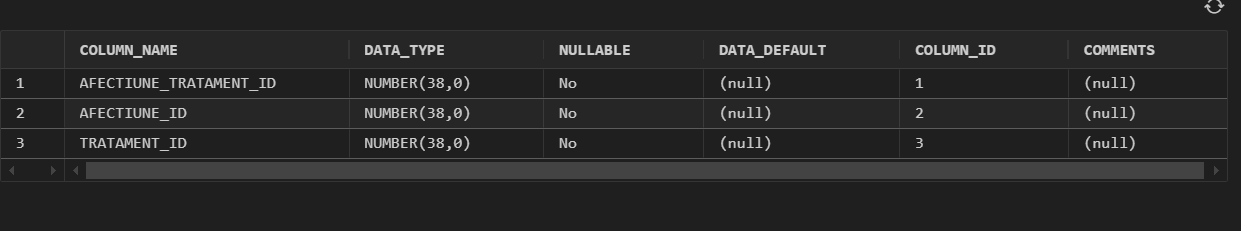
    afectiune\_id INT NOT NULL,

    tratament\_id INT NOT NULL,

    FOREIGN KEY (afectiune\_id) REFERENCES Afectiune (afectiune\_id),

    FOREIGN KEY (tratament\_id) REFERENCES Tratament (tratament\_id)

);



## Istoric\_Medical

--Creare secvente pentru id-urile tabelului Istoric\_Medical

CREATE SEQUENCE istoric\_medical\_seq START WITH 1 INCREMENT BY 1;

-- Creare tabel Istoric\_Medical

CREATE TABLE Istoric\_Medical (

    istoric\_medical\_id INT PRIMARY KEY,

    afectiune\_tratament\_id INT NOT NULL,

    pacient\_id INT NOT NULL,

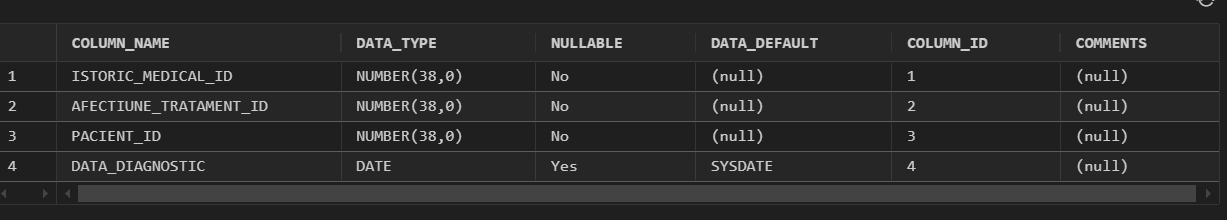
    data\_diagnostic DATE DEFAULT SYSDATE,

    UNIQUE (afectiune\_tratament\_id, pacient\_id),

    FOREIGN KEY (afectiune\_tratament\_id) REFERENCES Afectiune\_Tratament (afectiune\_tratament\_id),

    FOREIGN KEY (pacient\_id) REFERENCES Pacient (pacient\_id)

);



## Investigatie

--Creare secvente pentru id-urile tabelului Investigatie

CREATE SEQUENCE investigatie\_seq START WITH 1 INCREMENT BY 1;

-- Creare tabel Investigatie

CREATE TABLE Investigatie (

    investigatie\_id INT PRIMARY KEY,

    denumire VARCHAR2(100) NOT NULL,

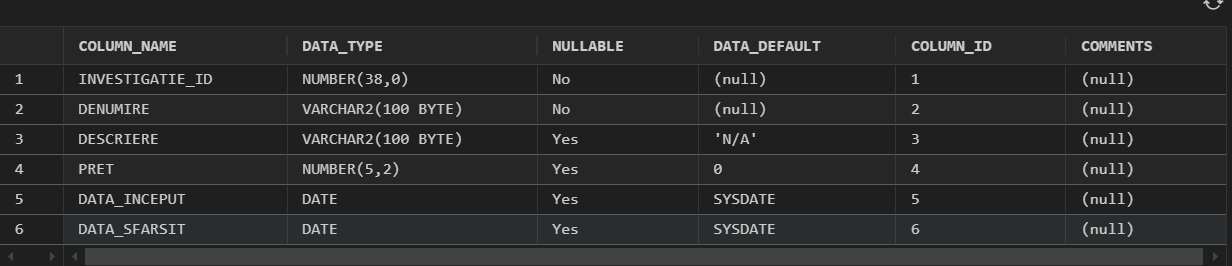
    descriere VARCHAR2(100) DEFAULT 'N/A',

    pret NUMBER(5, 2) DEFAULT 0,

    data\_inceput DATE DEFAULT SYSDATE,

    data\_sfarsit DATE DEFAULT SYSDATE

);



## Fisa\_Medicala

CREATE TABLE Fisa\_Medicala (

    fisa\_medicala\_id INT PRIMARY KEY,

    investigatie\_id INT NOT NULL,

    pacient\_id INT NOT NULL,

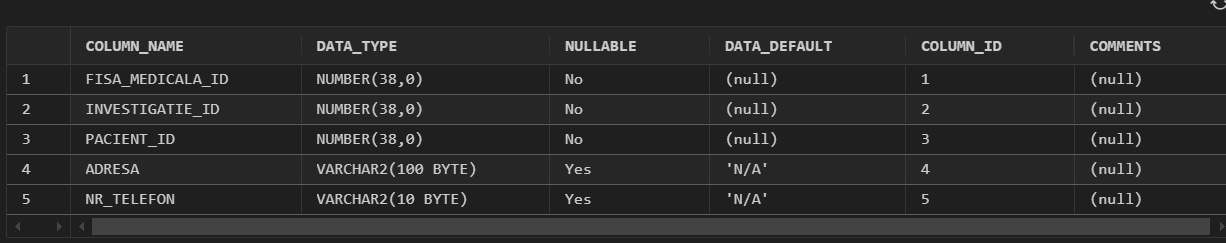
    adresa VARCHAR2(100) DEFAULT 'N/A',

    nr\_telefon VARCHAR2(10) DEFAULT 'N/A',

    FOREIGN KEY (investigatie\_id) REFERENCES Investigatie (investigatie\_id),

    FOREIGN KEY (pacient\_id) REFERENCES Pacient (pacient\_id)

);



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

## Medic

-- Inserare date in tabelul Medici

INSERT INTO Medic (medic\_id, nume, specializare, cod\_parafa) VALUES (medic\_seq.NEXTVAL, 'Dr. Ion Popescu', 'Cardiologie', 'CP123');

INSERT INTO Medic (medic\_id, nume, specializare, cod\_parafa) VALUES (medic\_seq.NEXTVAL, 'Dr. Maria Ionescu', 'Neurologie', 'NP456');

INSERT INTO Medic (medic\_id, nume, specializare, cod\_parafa) VALUES (medic\_seq.NEXTVAL, 'Dr. Andrei Georgescu', 'Pediatrie', 'PP789');

INSERT INTO Medic (medic\_id, nume, specializare, cod\_parafa) VALUES (medic\_seq.NEXTVAL, 'Dr. Elena Dumitrescu', 'Ortopedie', 'OP012');

INSERT INTO Medic (medic\_id, nume, specializare, cod\_parafa) VALUES (medic\_seq.NEXTVAL, 'Dr. Vasile Petrescu', 'Chirurgie', 'CP345');

INSERT INTO Medic (medic\_id, nume, specializare, cod\_parafa) VALUES (medic\_seq.NEXTVAL, 'Dr. Ana Popa', 'Ginecologie', 'GP678');

INSERT INTO Medic (medic\_id, nume, specializare, cod\_parafa) VALUES (medic\_seq.NEXTVAL, 'Dr. Andreea Vasile', 'Oftalmologie', 'OP901');

-- Selectare date din tabelul Medici

SELECT \* FROM Medic;



## Pacient

-- Inserare date in tabelul Pacienti

INSERT INTO Pacient (pacient\_id, prenume, nume, cnp, data\_nasterii) VALUES (pacient\_seq.NEXTVAL, 'Ana', 'Dedu', '2900101123456', TO\_DATE('01/01/90', 'DD/MM/YY'));

INSERT INTO Pacient (pacient\_id, prenume, nume, cnp, data\_nasterii) VALUES (pacient\_seq.NEXTVAL, 'George', 'Albina', '1850215123456', TO\_DATE('15/02/85', 'DD/MM/YY'));

INSERT INTO Pacient (pacient\_id, prenume, nume, cnp, data\_nasterii) VALUES (pacient\_seq.NEXTVAL, 'Elena', 'Vasile', '2780320123456', TO\_DATE('20/03/78', 'DD/MM/YY'));

INSERT INTO Pacient (pacient\_id, prenume, nume, cnp, data\_nasterii) VALUES (pacient\_seq.NEXTVAL, 'Cristina', 'Georgescu', '6000410123456', TO\_DATE('10/04/00', 'DD/MM/YY'));

INSERT INTO Pacient (pacient\_id, prenume, nume, cnp, data\_nasterii) VALUES (pacient\_seq.NEXTVAL, 'Ion', 'Dumitrescu','1950505123456', TO\_DATE('05/05/95', 'DD/MM/YY'));

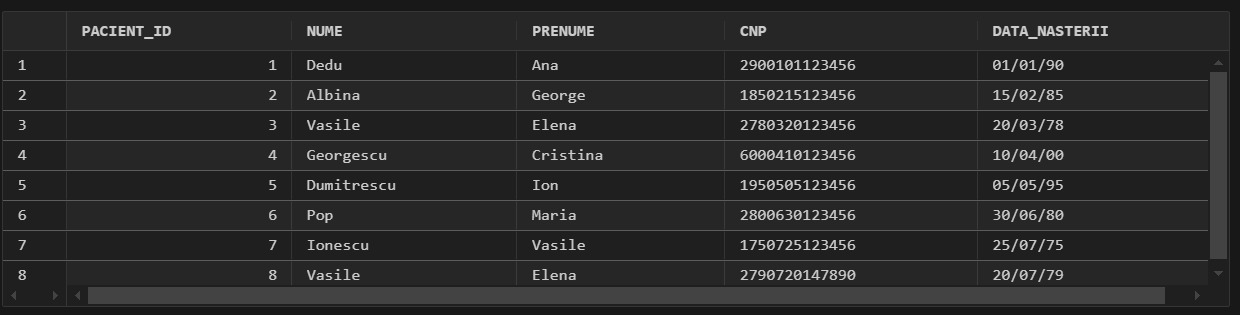
INSERT INTO Pacient (pacient\_id, prenume, nume, cnp, data\_nasterii) VALUES (pacient\_seq.NEXTVAL, 'Maria', 'Pop', '2800630123456', TO\_DATE('30/06/80', 'DD/MM/YY'));

INSERT INTO Pacient (pacient\_id, prenume, nume, cnp, data\_nasterii) VALUES (pacient\_seq.NEXTVAL, 'Vasile', 'Ionescu', '1750725123456', TO\_DATE('25/07/75', 'DD/MM/YY'));

INSERT INTO Pacient (pacient\_id, prenume, nume, cnp, data\_nasterii) VALUES (pacient\_seq.NEXTVAL, 'Elena', 'Vasile', '2790720147890', TO\_DATE('20/07/79', 'DD/MM/YY'));

-- Selectare date din tabelul Pacienti

SELECT \* FROM Pacient;



## Consultatie

-- Inserare date in tabelul Consultatie

INSERT INTO Consultatie (consultatie\_id, medic\_id, pacient\_id, data\_consultatie, ora\_consultatie) VALUES (consultatie\_seq.NEXTVAL, 1, 1, TO\_DATE('01/01/24', 'DD/MM/YY'), TO\_TIMESTAMP('01/10/24 08:30:00.000000000', 'DD/MM/YY HH24:MI:SS.FF'));

INSERT INTO Consultatie (consultatie\_id, medic\_id, pacient\_id, data\_consultatie, ora\_consultatie) VALUES (consultatie\_seq.NEXTVAL, 1, 1, TO\_DATE('01/01/24', 'DD/MM/YY'), TO\_TIMESTAMP('01/10/24 08:30:00.000000000', 'DD/MM/YY HH24:MI:SS.FF'));

INSERT INTO Consultatie (consultatie\_id, medic\_id, pacient\_id, data\_consultatie, ora\_consultatie) VALUES (consultatie\_seq.NEXTVAL, 7, 5, TO\_DATE('02/10/24', 'DD/MM/YY'), TO\_TIMESTAMP('01/10/24 09:00:00.000000000', 'DD/MM/YY HH24:MI:SS.FF'));

INSERT INTO Consultatie (consultatie\_id, medic\_id, pacient\_id, data\_consultatie, ora\_consultatie) VALUES (consultatie\_seq.NEXTVAL, 7, 3, TO\_DATE('03/11/24', 'DD/MM/YY'), TO\_TIMESTAMP('01/10/24 10:30:00.000000000', 'DD/MM/YY HH24:MI:SS.FF'));

INSERT INTO Consultatie (consultatie\_id, medic\_id, pacient\_id, data\_consultatie, ora\_consultatie) VALUES (consultatie\_seq.NEXTVAL, 4, 2, TO\_DATE('04/12/24', 'DD/MM/YY'), TO\_TIMESTAMP('01/10/24 11:00:00.000000000', 'DD/MM/YY HH24:MI:SS.FF'));

INSERT INTO Consultatie (consultatie\_id, medic\_id, pacient\_id, data\_consultatie, ora\_consultatie) VALUES (consultatie\_seq.NEXTVAL, 5, 4, TO\_DATE('05/02/24', 'DD/MM/YY'), TO\_TIMESTAMP('01/10/24 12:00:00.000000000', 'DD/MM/YY HH24:MI:SS.FF'));

INSERT INTO Consultatie (consultatie\_id, medic\_id, pacient\_id, data\_consultatie, ora\_consultatie) VALUES (consultatie\_seq.NEXTVAL, 6, 6, TO\_DATE('06/03/24', 'DD/MM/YY'), TO\_TIMESTAMP('01/10/24 13:30:00.000000000', 'DD/MM/YY HH24:MI:SS.FF'));

INSERT INTO Consultatie (consultatie\_id, medic\_id, pacient\_id, data\_consultatie, ora\_consultatie) VALUES (consultatie\_seq.NEXTVAL, 2, 7, TO\_DATE('07/04/24', 'DD/MM/YY'), TO\_TIMESTAMP('01/10/24 14:00:00.000000000', 'DD/MM/YY HH24:MI:SS.FF'));

INSERT INTO Consultatie (consultatie\_id, medic\_id, pacient\_id, data\_consultatie, ora\_consultatie) VALUES (consultatie\_seq.NEXTVAL, 1, 1, TO\_DATE('08/05/24', 'DD/MM/YY'), TO\_TIMESTAMP('01/10/24 15:00:00.000000000', 'DD/MM/YY HH24:MI:SS.FF'));

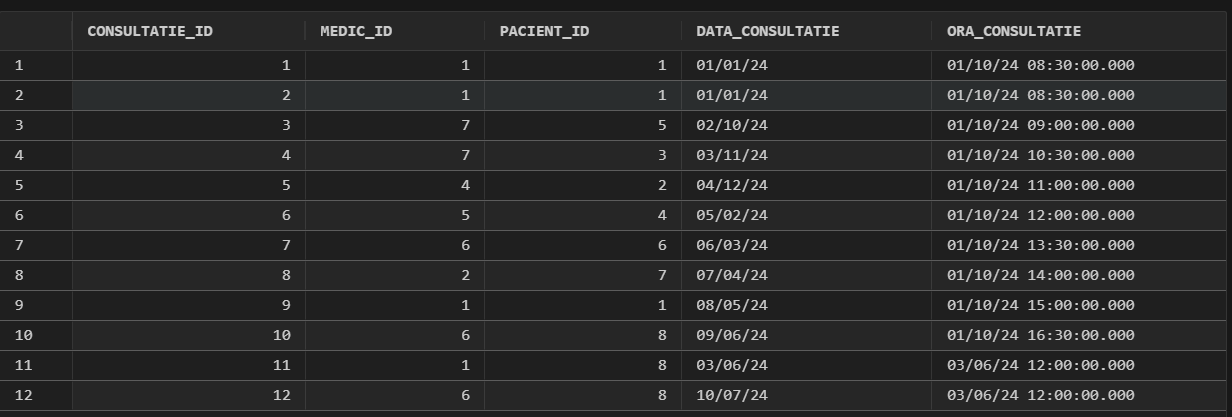
INSERT INTO Consultatie (consultatie\_id, medic\_id, pacient\_id, data\_consultatie, ora\_consultatie) VALUES (consultatie\_seq.NEXTVAL, 6, 8, TO\_DATE('09/06/24', 'DD/MM/YY'), TO\_TIMESTAMP('01/10/24 16:30:00.000000000', 'DD/MM/YY HH24:MI:SS.FF'));

INSERT INTO Consultatie (consultatie\_id, medic\_id, pacient\_id, data\_consultatie, ora\_consultatie) VALUES (consultatie\_seq.NEXTVAL, 1, 8, TO\_DATE('03/06/24', 'DD/MM/YY'), TO\_TIMESTAMP('03/06/24 12:00:00.000000000', 'DD/MM/YY HH24:MI:SS.FF'));

INSERT INTO Consultatie (consultatie\_id, medic\_id, pacient\_id, data\_consultatie, ora\_consultatie) VALUES (consultatie\_seq.NEXTVAL, 6, 8, TO\_DATE('10/07/24', 'DD/MM/YY'), TO\_TIMESTAMP('03/06/24 12:00:00.000000000', 'DD/MM/YY HH24:MI:SS.FF'));

-- Selectare date din tabelul Consultatie

SELECT \* FROM Consultatie;



## Afectiune

-- Inserare date in tabelul Afectiune

INSERT INTO Afectiune (afectiune\_id, denumire, descriere) VALUES (afectiune\_seq.NEXTVAL, 'Tahicardie', 'Ritmul cardiac accelerat');

INSERT INTO Afectiune (afectiune\_id, denumire, descriere) VALUES (afectiune\_seq.NEXTVAL, 'Migrena', 'Dureri de cap intense');

INSERT INTO Afectiune (afectiune\_id, denumire, descriere) VALUES (afectiune\_seq.NEXTVAL, 'Hernie de disc', 'Durere la nivelul spatelui si picioarelor');

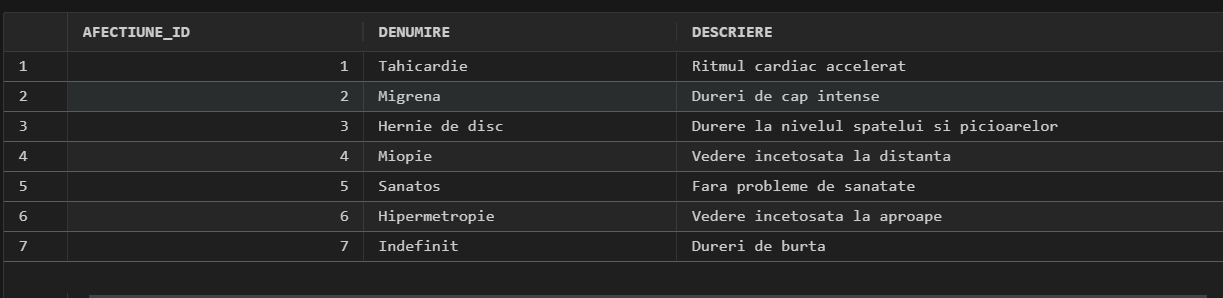
INSERT INTO Afectiune (afectiune\_id, denumire, descriere) VALUES (afectiune\_seq.NEXTVAL, 'Miopie', 'Vedere incetosata la distanta');

INSERT INTO Afectiune (afectiune\_id, denumire, descriere) VALUES (afectiune\_seq.NEXTVAL, 'Sanatos', 'Fara probleme de sanatate');

INSERT INTO Afectiune (afectiune\_id, denumire, descriere) VALUES (afectiune\_seq.NEXTVAL, 'Hipermetropie', 'Vedere incetosata la aproape');

INSERT INTO Afectiune (afectiune\_id, descriere) VALUES (afectiune\_seq.NEXTVAL, 'Dureri de burta');

-- Selectare date din tabelul Afectiune

SELECT \* FROM Afectiune;

## Tratament

-- Inserare date in tabelul Tratament

INSERT INTO Tratament (tratament\_id, denumire, descriere, data\_incepere, data\_sfarsit) VALUES (tratament\_seq.NEXTVAL, 'Tratament pentru tahicardie', 'Medicatie pentru reglarea ritmului cardiac', TO\_DATE('01/01/24', 'DD/MM/YY'), TO\_DATE('15/01/25', 'DD/MM/YY'));

INSERT INTO Tratament (tratament\_id, denumire, descriere, data\_incepere, data\_sfarsit) VALUES (tratament\_seq.NEXTVAL, 'Paracetamol', 'Medicatie pentru calmarea durerilor de cap', TO\_DATE('02/10/24', 'DD/MM/YY'), TO\_DATE('05/10/24', 'DD/MM/YY'));

INSERT INTO Tratament (tratament\_id, denumire, descriere, data\_incepere, data\_sfarsit) VALUES (tratament\_seq.NEXTVAL, 'Antiinflamatoare', 'Medicatie pentru reducerea inflamatiei', TO\_DATE('03/11/24', 'DD/MM/YY'), TO\_DATE('10/11/24', 'DD/MM/YY'));

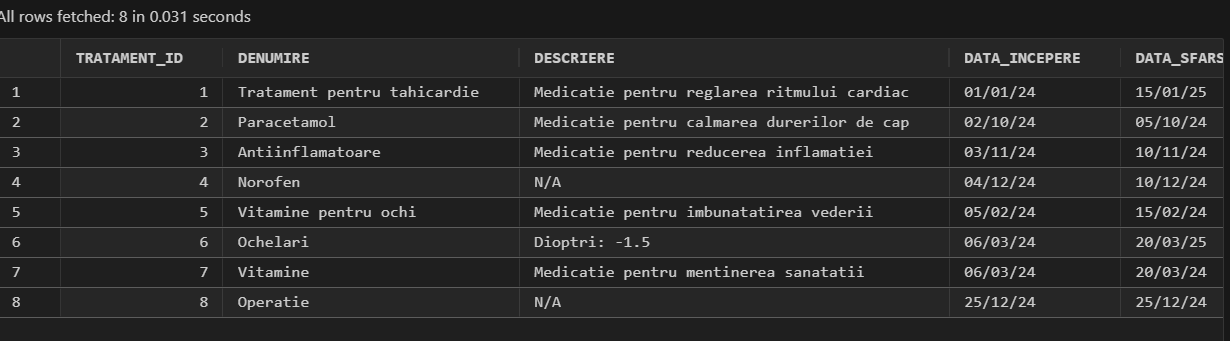
INSERT INTO Tratament (tratament\_id, denumire, descriere, data\_incepere, data\_sfarsit) VALUES (tratament\_seq.NEXTVAL, 'Norofen', 'N/A', TO\_DATE('04/12/24', 'DD/MM/YY'), TO\_DATE('10/12/24', 'DD/MM/YY'));

INSERT INTO Tratament (tratament\_id, denumire, descriere, data\_incepere, data\_sfarsit) VALUES (tratament\_seq.NEXTVAL, 'Vitamine pentru ochi', 'Medicatie pentru imbunatatirea vederii', TO\_DATE('05/02/24', 'DD/MM/YY'), TO\_DATE('15/02/24', 'DD/MM/YY'));

INSERT INTO Tratament (tratament\_id, denumire, descriere, data\_incepere, data\_sfarsit) VALUES (tratament\_seq.NEXTVAL, 'Ochelari', 'Dioptri: -1.5', TO\_DATE('06/03/24', 'DD/MM/YY'), TO\_DATE('20/03/25', 'DD/MM/YY'));

INSERT INTO Tratament (tratament\_id, denumire, descriere, data\_incepere, data\_sfarsit) VALUES (tratament\_seq.NEXTVAL, 'Vitamine', 'Medicatie pentru mentinerea sanatatii', TO\_DATE('06/03/24', 'DD/MM/YY'), TO\_DATE('20/03/24', 'DD/MM/YY'));

INSERT INTO Tratament (tratament\_id, denumire) VALUES (tratament\_seq.NEXTVAL, 'Operatie');



## Afectiune\_tratament

-- Inserare date in tabelul Afectiune\_Tratament

INSERT INTO Afectiune\_Tratament (afectiune\_tratament\_id, afectiune\_id, tratament\_id) VALUES (afectiune\_tratament\_seq.NEXTVAL, 1, 1);

INSERT INTO Afectiune\_Tratament (afectiune\_tratament\_id, afectiune\_id, tratament\_id) VALUES (afectiune\_tratament\_seq.NEXTVAL, 2, 2);

INSERT INTO Afectiune\_Tratament (afectiune\_tratament\_id, afectiune\_id, tratament\_id) VALUES (afectiune\_tratament\_seq.NEXTVAL, 2, 7);

INSERT INTO Afectiune\_Tratament (afectiune\_tratament\_id, afectiune\_id, tratament\_id) VALUES (afectiune\_tratament\_seq.NEXTVAL, 3, 3);

INSERT INTO Afectiune\_Tratament (afectiune\_tratament\_id, afectiune\_id, tratament\_id) VALUES (afectiune\_tratament\_seq.NEXTVAL, 3, 4);

INSERT INTO Afectiune\_Tratament (afectiune\_tratament\_id, afectiune\_id, tratament\_id) VALUES (afectiune\_tratament\_seq.NEXTVAL, 3, 8);

INSERT INTO Afectiune\_Tratament (afectiune\_tratament\_id, afectiune\_id, tratament\_id) VALUES (afectiune\_tratament\_seq.NEXTVAL, 4, 5);

INSERT INTO Afectiune\_Tratament (afectiune\_tratament\_id, afectiune\_id, tratament\_id) VALUES (afectiune\_tratament\_seq.NEXTVAL, 4, 6);

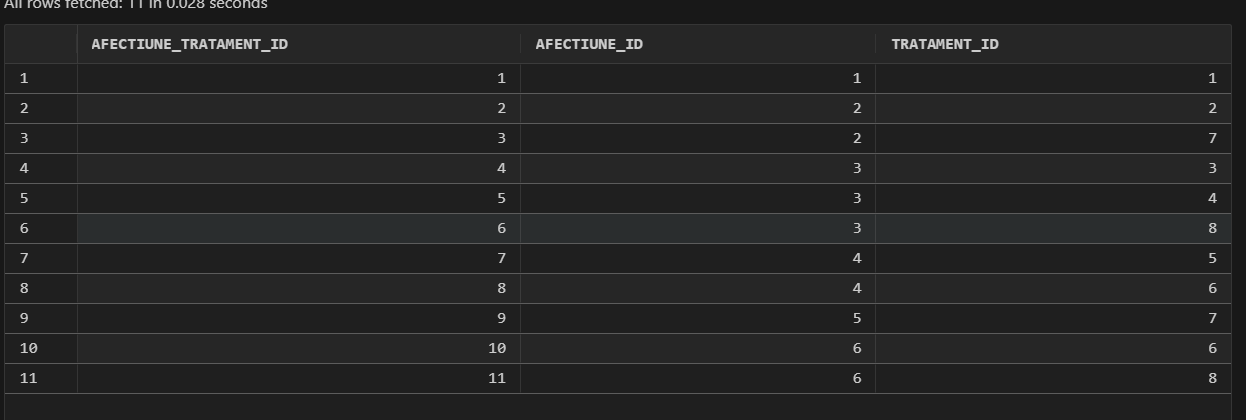
INSERT INTO Afectiune\_Tratament (afectiune\_tratament\_id, afectiune\_id, tratament\_id) VALUES (afectiune\_tratament\_seq.NEXTVAL, 5, 7);

INSERT INTO Afectiune\_Tratament (afectiune\_tratament\_id, afectiune\_id, tratament\_id) VALUES (afectiune\_tratament\_seq.NEXTVAL, 6, 6);

INSERT INTO Afectiune\_Tratament (afectiune\_tratament\_id, afectiune\_id, tratament\_id) VALUES (afectiune\_tratament\_seq.NEXTVAL, 6, 8);

-- Selectare date din tabelul Afectiune\_Tratament

SELECT \* FROM Afectiune\_Tratament;



## Istoric-Medical

-- Inserare date in tabelul Istoric\_Medical

INSERT INTO Istoric\_Medical (istoric\_medical\_id, afectiune\_tratament\_id, pacient\_id, data\_diagnostic) VALUES (istoric\_medical\_seq.NEXTVAL, 1, 1, TO\_DATE('01/01/24', 'DD/MM/YY'));

INSERT INTO Istoric\_Medical (istoric\_medical\_id, afectiune\_tratament\_id, pacient\_id, data\_diagnostic) VALUES (istoric\_medical\_seq.NEXTVAL, 2, 2, TO\_DATE('02/10/24', 'DD/MM/YY'));

INSERT INTO Istoric\_Medical (istoric\_medical\_id, afectiune\_tratament\_id, pacient\_id, data\_diagnostic) VALUES (istoric\_medical\_seq.NEXTVAL, 3, 2, TO\_DATE('03/11/24', 'DD/MM/YY'));

INSERT INTO Istoric\_Medical (istoric\_medical\_id, afectiune\_tratament\_id, pacient\_id, data\_diagnostic) VALUES (istoric\_medical\_seq.NEXTVAL, 4, 4, TO\_DATE('04/12/24', 'DD/MM/YY'));

INSERT INTO Istoric\_Medical (istoric\_medical\_id, afectiune\_tratament\_id, pacient\_id, data\_diagnostic) VALUES (istoric\_medical\_seq.NEXTVAL, 5, 4, TO\_DATE('05/02/24', 'DD/MM/YY'));

INSERT INTO Istoric\_Medical (istoric\_medical\_id, afectiune\_tratament\_id, pacient\_id, data\_diagnostic) VALUES (istoric\_medical\_seq.NEXTVAL, 6, 4, TO\_DATE('06/03/24', 'DD/MM/YY'));

INSERT INTO Istoric\_Medical (istoric\_medical\_id, afectiune\_tratament\_id, pacient\_id, data\_diagnostic) VALUES (istoric\_medical\_seq.NEXTVAL, 7, 6, TO\_DATE('07/04/24', 'DD/MM/YY'));

INSERT INTO Istoric\_Medical (istoric\_medical\_id, afectiune\_tratament\_id, pacient\_id, data\_diagnostic) VALUES (istoric\_medical\_seq.NEXTVAL, 8, 5, TO\_DATE('08/05/24', 'DD/MM/YY'));

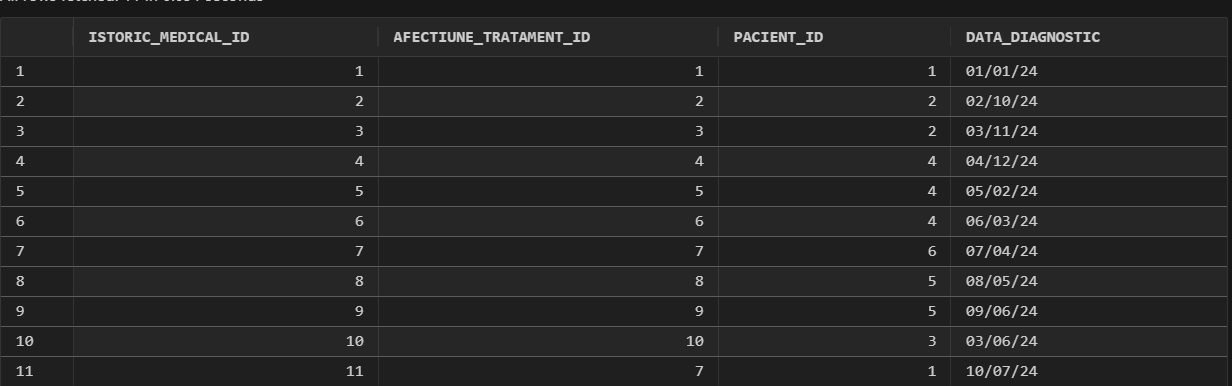
INSERT INTO Istoric\_Medical (istoric\_medical\_id, afectiune\_tratament\_id, pacient\_id, data\_diagnostic) VALUES (istoric\_medical\_seq.NEXTVAL, 9, 5, TO\_DATE('09/06/24', 'DD/MM/YY'));

INSERT INTO Istoric\_Medical (istoric\_medical\_id, afectiune\_tratament\_id, pacient\_id, data\_diagnostic) VALUES (istoric\_medical\_seq.NEXTVAL, 10, 3, TO\_DATE('03/06/24', 'DD/MM/YY'));

INSERT INTO Istoric\_Medical (istoric\_medical\_id, afectiune\_tratament\_id, pacient\_id, data\_diagnostic) VALUES (istoric\_medical\_seq.NEXTVAL, 7, 1, TO\_DATE('10/07/24', 'DD/MM/YY'));

-- Selectare date din tabelul Istoric\_Medical

SELECT \* FROM Istoric\_Medical;



## Investigatie

-- Inserare date in tabelul Investigatie

INSERT INTO Investigatie (investigatie\_id, denumire, descriere, pret, data\_inceput, data\_sfarsit) VALUES (investigatie\_seq.NEXTVAL, 'Electrocardiograma', 'Examinare ritm cardiac', 200.00, TO\_DATE('01/01/24', 'DD/MM/YY'), TO\_DATE('15/01/25', 'DD/MM/YY'));

INSERT INTO Investigatie (investigatie\_id, denumire, descriere, pret, data\_inceput, data\_sfarsit) VALUES (investigatie\_seq.NEXTVAL, 'Tomografie', 'Examinare creier', 500.00, TO\_DATE('02/10/24', 'DD/MM/YY'), TO\_DATE('05/10/24', 'DD/MM/YY'));

INSERT INTO Investigatie (investigatie\_id, denumire, descriere, pret, data\_inceput, data\_sfarsit) VALUES (investigatie\_seq.NEXTVAL, 'RMN', 'Examinare coloana vertebrala', 700.00, TO\_DATE('03/11/24', 'DD/MM/YY'), TO\_DATE('10/11/24', 'DD/MM/YY'));

INSERT INTO Investigatie (investigatie\_id, denumire, descriere, pret, data\_inceput, data\_sfarsit) VALUES (investigatie\_seq.NEXTVAL, 'Examinare ochi', 'Examinare vedere', 150.00, TO\_DATE('04/12/24', 'DD/MM/YY'), TO\_DATE('10/12/24', 'DD/MM/YY'));

INSERT INTO Investigatie (investigatie\_id, denumire, descriere, pret, data\_inceput, data\_sfarsit) VALUES (investigatie\_seq.NEXTVAL, 'Ecografie', 'Examinare organe interne', 300.00, TO\_DATE('05/02/24', 'DD/MM/YY'), TO\_DATE('15/02/24', 'DD/MM/YY'));

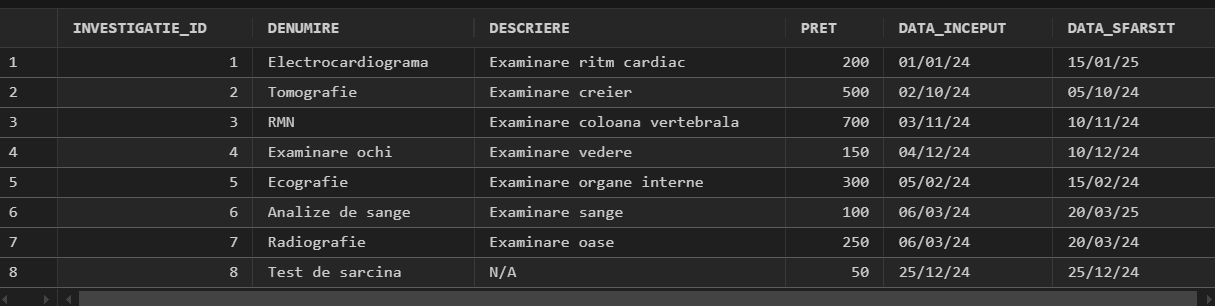
INSERT INTO Investigatie (investigatie\_id, denumire, descriere, pret, data\_inceput, data\_sfarsit) VALUES (investigatie\_seq.NEXTVAL, 'Analize de sange', 'Examinare sange', 100.00, TO\_DATE('06/03/24', 'DD/MM/YY'), TO\_DATE('20/03/25', 'DD/MM/YY'));

INSERT INTO Investigatie (investigatie\_id, denumire, descriere, pret, data\_inceput, data\_sfarsit) VALUES (investigatie\_seq.NEXTVAL, 'Radiografie', 'Examinare oase', 250.00, TO\_DATE('06/03/24', 'DD/MM/YY'), TO\_DATE('20/03/24', 'DD/MM/YY'));

INSERT INTO Investigatie (investigatie\_id, denumire, pret) VALUES (investigatie\_seq.NEXTVAL, 'Test de sarcina', 50.00);

-- Selectare date din tabelul Investigatie

SELECT \* FROM Investigatie;



## Fisa\_Medicala

-- Inserare date in tabelul Fisa\_Medicala

-- Inserare date in tabelul Fisa\_Medicala

INSERT INTO Fisa\_Medicala (fisa\_medicala\_id, investigatie\_id, pacient\_id, adresa, nr\_telefon) VALUES (fisa\_medicala\_seq.NEXTVAL, 3, 1, 'Str. Mihai Eminescu, nr. 1', '0723456789');

INSERT INTO Fisa\_Medicala (fisa\_medicala\_id, investigatie\_id, pacient\_id, adresa, nr\_telefon) VALUES (fisa\_medicala\_seq.NEXTVAL, 1, 1, 'Str. Mihai Eminescu, nr. 1', '0723456789');

INSERT INTO Fisa\_Medicala (fisa\_medicala\_id, investigatie\_id, pacient\_id, adresa, nr\_telefon) VALUES (fisa\_medicala\_seq.NEXTVAL, 7, 1, 'Str. Mihai Eminescu, nr. 1', '0723456789');

INSERT INTO Fisa\_Medicala (fisa\_medicala\_id, investigatie\_id, pacient\_id, adresa, nr\_telefon) VALUES (fisa\_medicala\_seq.NEXTVAL, 2, 2, 'Str. Ion Creanga, nr. 2', '0723456789');

INSERT INTO Fisa\_Medicala (fisa\_medicala\_id, investigatie\_id, pacient\_id, adresa, nr\_telefon) VALUES (fisa\_medicala\_seq.NEXTVAL, 5, 3, 'Str. Ciberneticii nr. 9', '0735856336');

INSERT INTO Fisa\_Medicala (fisa\_medicala\_id, investigatie\_id, pacient\_id, adresa, nr\_telefon) VALUES (fisa\_medicala\_seq.NEXTVAL, 4, 3, 'Str. Ciberneticii nr. 9', '0735856336');

INSERT INTO Fisa\_Medicala (fisa\_medicala\_id, investigatie\_id, pacient\_id, adresa, nr\_telefon) VALUES (fisa\_medicala\_seq.NEXTVAL, 6, 4, 'Str. Mihai Viteazu, nr. 3', '0733361690');

INSERT INTO Fisa\_Medicala (fisa\_medicala\_id, investigatie\_id, pacient\_id, adresa, nr\_telefon) VALUES (fisa\_medicala\_seq.NEXTVAL, 5, 4, 'Str. Mihai Viteazu, nr. 3', '0733361690');

INSERT INTO Fisa\_Medicala (fisa\_medicala\_id, investigatie\_id, pacient\_id, adresa, nr\_telefon) VALUES (fisa\_medicala\_seq.NEXTVAL, 6, 5, 'Str. Aviatorilor, nr. 5', '0733361690');

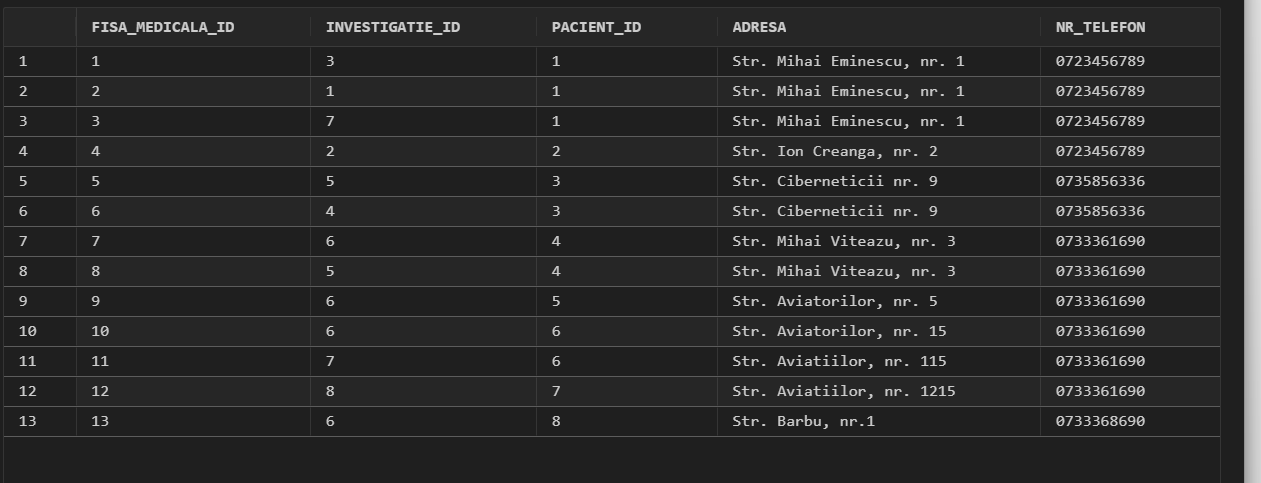
INSERT INTO Fisa\_Medicala (fisa\_medicala\_id, investigatie\_id, pacient\_id, adresa, nr\_telefon) VALUES (fisa\_medicala\_seq.NEXTVAL, 6, 6, 'Str. Aviatorilor, nr. 15', '0733361690');

INSERT INTO Fisa\_Medicala (fisa\_medicala\_id, investigatie\_id, pacient\_id, adresa, nr\_telefon) VALUES (fisa\_medicala\_seq.NEXTVAL, 7, 6, 'Str. Aviatiilor, nr. 115', '0733361690');

INSERT INTO Fisa\_Medicala (fisa\_medicala\_id, investigatie\_id, pacient\_id, adresa, nr\_telefon) VALUES (fisa\_medicala\_seq.NEXTVAL, 8, 7, 'Str. Aviatiilor, nr. 1215', '0733361690');

-- Selectare date din tabelul Fisa\_Medicala

SELECT \* FROM Fisa\_Medicala;



# 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.

## Cernița limbaj natural:

Scrieți un subprogram stocat independent (de tip procedură) care să calculeze numărul total de consultații efectuate de fiecare medic și să afișeze fisa medicala a fiecărui pacient care a beneficiat de consultații. Procedura trebuie să utilizeze cele trei tipuri de colecții pentru a modela medicii, consultațiile și pacienții și să trateze eventualele erori, cum ar fi:

* Medicii fără consultații înregistrate.
* Pacienții fără un diagnostic atribuit.

Acest subprogram poate fi utilizat pentru generarea rapoartelor săptămânale de activitate medicală.

## Apelare subprogram:

-- Definire tip la nivel de schemă pentru rândurile din tabelul Medici

CREATE OR REPLACE TYPE medic\_row AS OBJECT (

    id\_medic NUMBER,

    nume VARCHAR2(100)

);

/

-- Definire tip Nested Table bazat pe medic\_row

CREATE OR REPLACE TYPE medici\_nt AS TABLE OF medic\_row;

/

-- Definire VARRAY pentru pacienți

CREATE OR REPLACE TYPE pacient\_varray AS VARRAY(100) OF NUMBER; -- VARRAY cu dimensiune fixă in care se stochează id-urile pacienților

/

-- Creare procedură pentru generarea raportului de activitate

CREATE OR REPLACE PROCEDURE Raport\_Activitate IS

    -- Declarare Nested Table pentru medici

    medici medici\_nt;

    -- Declarare Associative Array pentru consultații

    TYPE consultatii\_array IS TABLE OF NUMBER INDEX BY PLS\_INTEGER;

    consultatii consultatii\_array;

    -- Declarare VARRAY pentru pacienți

    TYPE pacienti\_array IS TABLE OF pacient\_varray INDEX BY PLS\_INTEGER;

    pacient\_var pacienti\_array;

    -- Variabile locale pentru detalii pacient și afectiuni

    nume\_pacient Pacient.nume%TYPE;

    prenume\_pacient Pacient.prenume%TYPE;

    adresa Fisa\_Medicala.adresa%TYPE;

    nr\_telefon Fisa\_Medicala.nr\_telefon%TYPE;

    nr\_afectiuni NUMBER;

    denumire\_afectiune Afectiune.denumire%TYPE;

    nr\_consultatii NUMBER;

BEGIN

    DBMS\_OUTPUT.PUT\_LINE('Raport Activitate');

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

    -- Selectare medici

    SELECT medic\_row(medic\_id, nume)

    BULK COLLECT INTO medici

    FROM Medic;

    -- Inițializare consultații pentru fiecare medic

    FOR i IN 1..medici.COUNT LOOP

        consultatii(medici(i).id\_medic) := 0; -- Inițial, zero consultații

    END LOOP;

    -- Procesare pentru fiecare medic

    FOR i IN 1..medici.COUNT LOOP

        BEGIN

            pacient\_var(medici(i).id\_medic) := pacient\_varray(); -- VARRAY gol

            -- Afisare număr consultații efectuate de fiecare medic

            SELECT COUNT(\*)

            INTO consultatii(medici(i).id\_medic)

            FROM Consultatie

            WHERE medic\_id = medici(i).id\_medic;

            IF consultatii(medici(i).id\_medic) = 0 THEN

                RAISE NO\_DATA\_FOUND;

            END IF;

            DBMS\_OUTPUT.PUT\_LINE('Medic: ' || medici(i).nume || ' - Numar Consultatii: ' || consultatii(medici(i).id\_medic));

            -- Selectare pacienți consultați de medic

            SELECT DISTINCT pacient\_id -- Distinct pentru a evita pacienții cu mai multe consultații la același medic

            BULK COLLECT INTO pacient\_var(medici(i).id\_medic)

            FROM Consultatie

            WHERE medic\_id = medici(i).id\_medic;

            -- Iterare prin lista de pacienți

            FOR j IN 1..pacient\_var(medici(i).id\_medic).COUNT LOOP

                DBMS\_OUTPUT.PUT\_LINE('Pacient ID: ' || pacient\_var(medici(i).id\_medic)(j));

                -- Afisare nume și prenume pacient

                SELECT nume, prenume

                INTO nume\_pacient, prenume\_pacient

                FROM Pacient

                WHERE pacient\_id = pacient\_var(medici(i).id\_medic)(j);

                DBMS\_OUTPUT.PUT\_LINE('Nume: ' || nume\_pacient || ' Prenume: ' || prenume\_pacient);

                -- Afisare detalii fisa medicala

                SELECT Distinct adresa , nr\_telefon

                INTO adresa, nr\_telefon

                FROM Fisa\_Medicala

                WHERE pacient\_id = pacient\_var(medici(i).id\_medic)(j);

                DBMS\_OUTPUT.PUT\_LINE('Fisa Medicala: ' || adresa || ' nr. telefon: ' || nr\_telefon);

                --Verificare numar consultații pacient

                SELECT COUNT(\*)

                INTO nr\_consultatii

                FROM Consultatie

                WHERE pacient\_id = pacient\_var(medici(i).id\_medic)(j) AND medic\_id = medici(i).id\_medic;

                -- Afisare nr consultații pacient la acelasi medic

                DBMS\_OUTPUT.PUT\_LINE('Numar Consultatii la medicul curent: ' || nr\_consultatii);

                -- Verificare dacă pacientul are afecțiuni

                SELECT COUNT(\*)

                INTO nr\_afectiuni

                FROM Istoric\_Medical

                WHERE pacient\_id = pacient\_var(medici(i).id\_medic)(j);

                -- Afișare afecțiuni

                IF nr\_afectiuni > 0 THEN

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

                    FOR k IN (SELECT Distinct A.denumire

                              FROM Afectiune A

                              JOIN Afectiune\_Tratament AT ON A.afectiune\_id = AT.afectiune\_id

                              JOIN Istoric\_Medical IM ON AT.afectiune\_tratament\_id = IM.afectiune\_tratament\_id

                              WHERE IM.pacient\_id = pacient\_var(medici(i).id\_medic)(j)) LOOP

                        DBMS\_OUTPUT.PUT\_LINE('-'||k.denumire);

                    END LOOP;

                ELSE

                    DBMS\_OUTPUT.PUT\_LINE('Pacientul nu are afecțiuni înregistrate.');

                END IF;

            END LOOP;

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

        EXCEPTION

            WHEN NO\_DATA\_FOUND THEN

                DBMS\_OUTPUT.PUT\_LINE('Nu există consultații pentru medicul ' || medici(i).nume);

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

            WHEN TOO\_MANY\_ROWS THEN

                DBMS\_OUTPUT.PUT\_LINE('Prea multe înregistrări găsite ');

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

        END;

    END LOOP;

END;

/

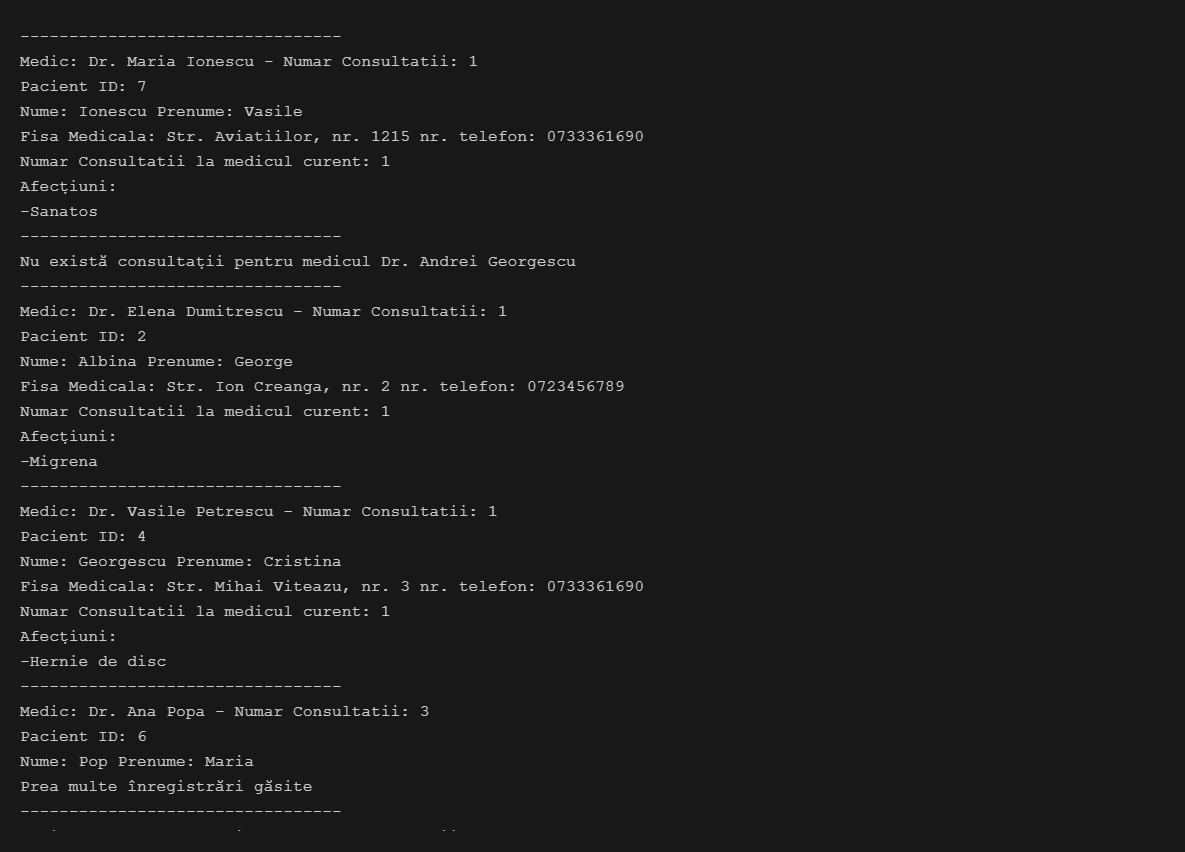
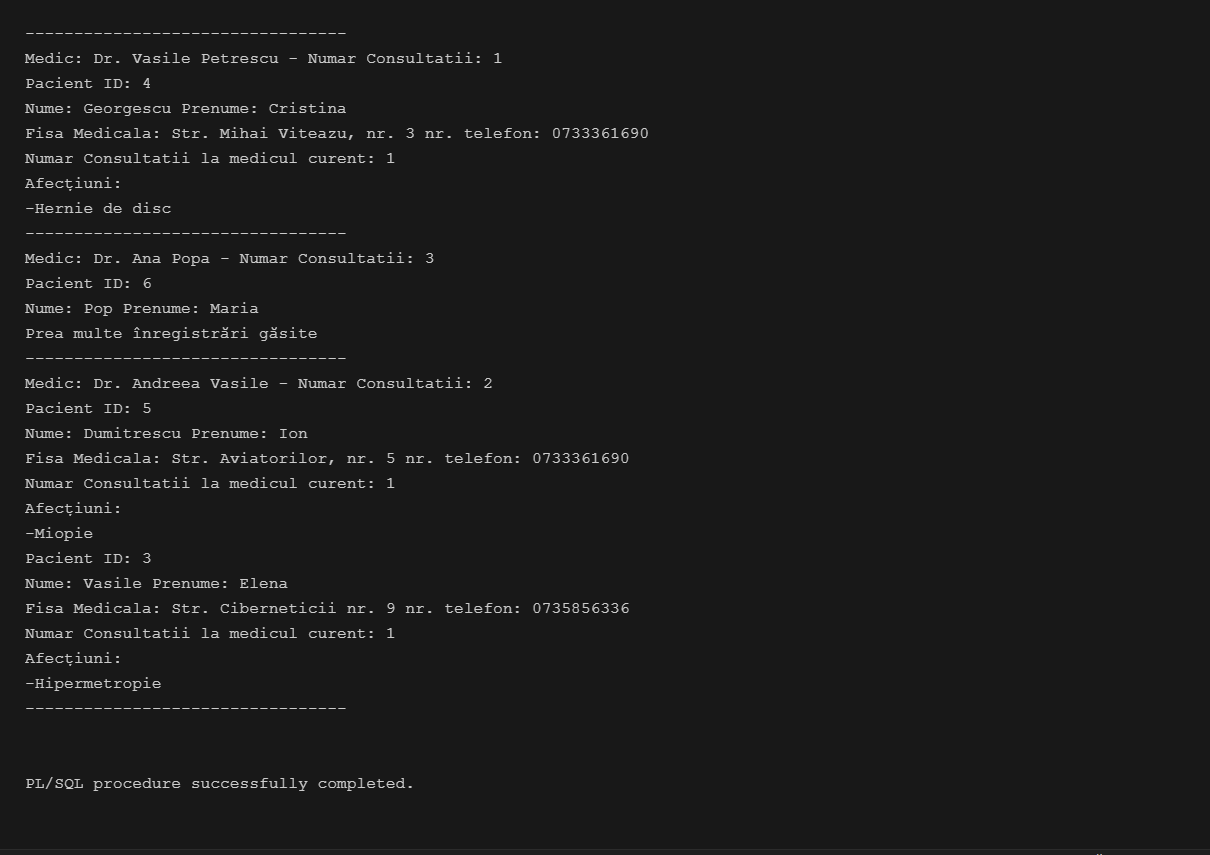
-- Apelare procedură

BEGIN

    Raport\_Activitate;

END;

## Print-screen



# 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.

## Cernița limbaj natural:

Scrieți un subprogram stocat independent (de tip procedură) care să parcurgă toți medicii din baza de date (folosind un cursor simplu). Pentru fiecare medic, să se extragă consultațiile efectuate de acesta într-o anumită perioadă (parametrii perioadei sunt transmiși subprogramului) și să se afișeze lista pacienților consultați in acea perioada. Să trateze eventualele erori, cum ar fi:

* Situația în care un medic nu are consultații în perioada specificată (NO\_DATA\_FOUND).
* Situația în care datele sunt ambigue, de exemplu, un medic este asociat cu mai multe înregistrări duplicate pentru aceeași consultație (TOO\_MANY\_ROWS).

Acest subprogram poate fi utilizat pentru a analiza activitatea spitalului într-un interval de timp specific și pentru a identifica medicii cu cea mai mare încărcare de muncă.

## Apelare subprogram:

CREATE OR REPLACE PROCEDURE Consultatii\_Perioada(

    data\_inceput IN DATE,

    data\_sfarsit IN DATE

) IS

    -- Cursor pentru medici

    CURSOR c\_medici IS

        SELECT medic\_id, nume

        FROM Medic;

    -- Cursor pentru consultații

    CURSOR c\_consultatii (medic\_id\_param NUMBER) IS

        SELECT pacient\_id, data\_consultatie, consultatie\_id

        FROM Consultatie

        WHERE medic\_id = medic\_id\_param AND data\_consultatie BETWEEN data\_inceput AND data\_sfarsit;

    -- Variabile pentru stocarea informațiilor despre pacient

    pacient\_nume   Pacient.nume%TYPE;

    pacient\_prenume Pacient.prenume%TYPE;

    nr\_pacienti NUMBER;

BEGIN

    -- Iterare prin medici

    FOR medici\_rec IN c\_medici LOOP

        DBMS\_OUTPUT.PUT\_LINE('Medic: ' || medici\_rec.nume);

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

        BEGIN

            -- Flag pentru a detecta dacă există consultații

            DECLARE

                exista\_consultatii BOOLEAN := FALSE;

            BEGIN

                -- Iterare prin consultațiile unui medic

                FOR consultatii\_rec IN c\_consultatii(medici\_rec.medic\_id) LOOP

                    exista\_consultatii := TRUE;

                    BEGIN

                        -- Obținerea numelui pacientului

                        SELECT nume, prenume

                        INTO pacient\_nume, pacient\_prenume

                        FROM Pacient

                        WHERE pacient\_id = consultatii\_rec.pacient\_id;

                        -- Verificare dacă există mai mulți pacienți cu același ID consultati de același medic

                        SELECT COUNT(\*)

                        INTO nr\_pacienti

                        FROM Consultatie

                        WHERE pacient\_id = consultatii\_rec.pacient\_id

                          AND medic\_id = medici\_rec.medic\_id

                          AND TO\_CHAR(data\_consultatie, 'YYYY-MM-DD') = TO\_CHAR(consultatii\_rec.data\_consultatie, 'YYYY-MM-DD');

                        IF nr\_pacienti > 1 THEN

                            RAISE TOO\_MANY\_ROWS;

                        END IF;

                        DBMS\_OUTPUT.PUT\_LINE('Pacient: ' || pacient\_nume || ' ' || pacient\_prenume || ' - Data Consultație: ' || consultatii\_rec.data\_consultatie);

                    EXCEPTION

                        WHEN TOO\_MANY\_ROWS THEN

                            DBMS\_OUTPUT.PUT\_LINE('Date ambigue pentru pacientul cu ID ' || consultatii\_rec.pacient\_id);

                    END;

                END LOOP;

                -- Dacă nu există consultații, ridică NO\_DATA\_FOUND

                IF NOT exista\_consultatii THEN

                    RAISE NO\_DATA\_FOUND;

                END IF;

            EXCEPTION

                WHEN NO\_DATA\_FOUND THEN

                    DBMS\_OUTPUT.PUT\_LINE('Nu există consultații pentru medicul ' || medici\_rec.nume || ' în perioada specificată.');

            END;

        END;

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

    END LOOP;

END;

/

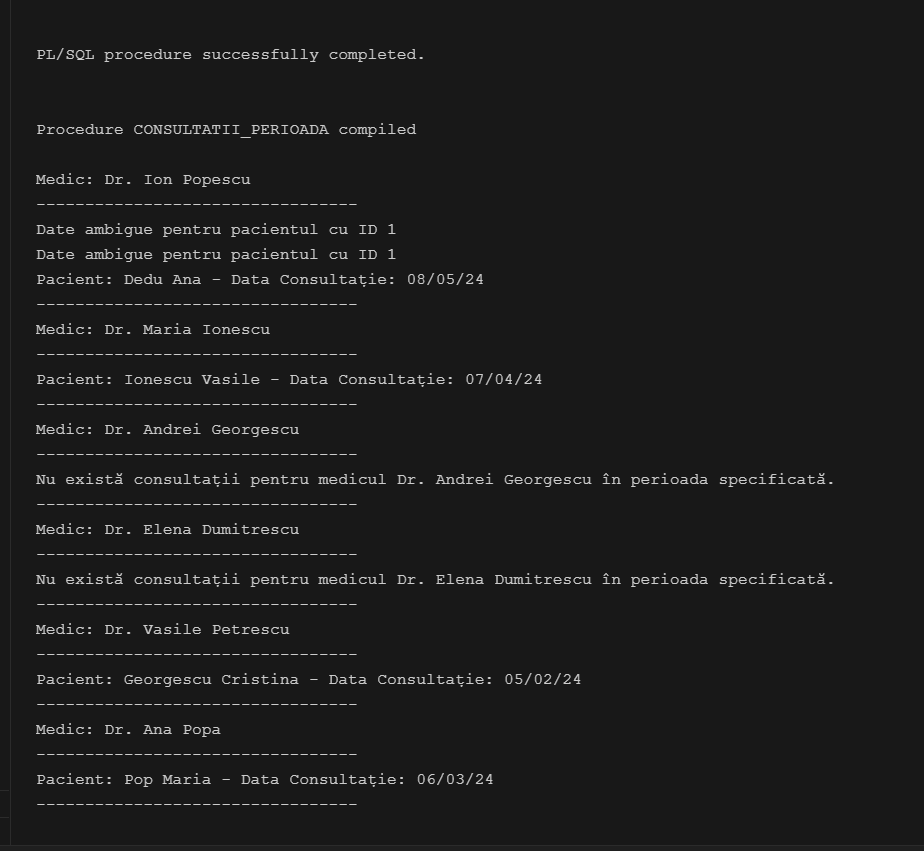
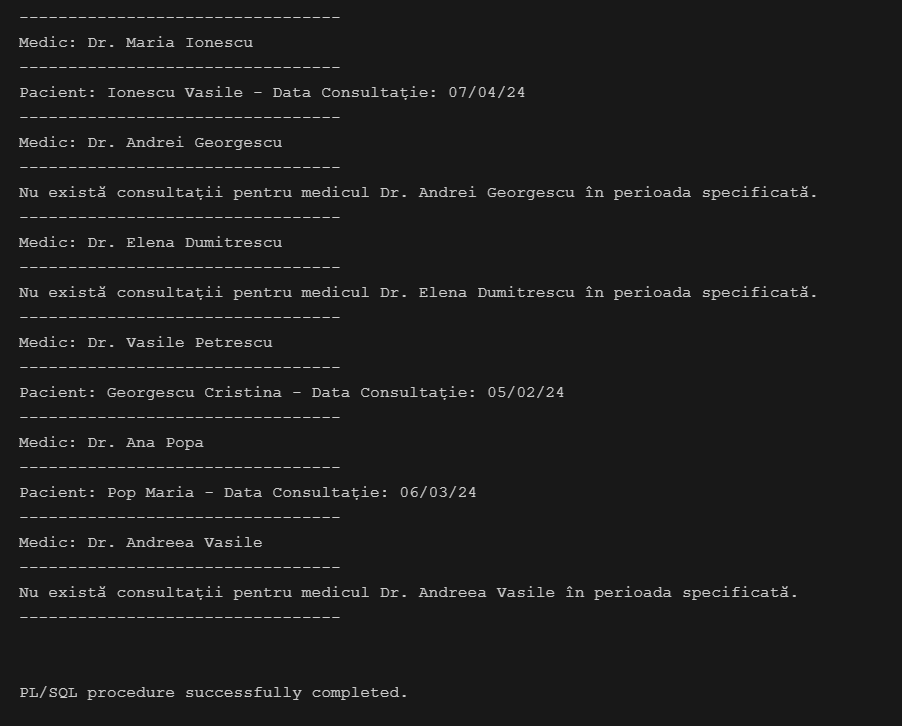
-- Apelare procedură

BEGIN

    Consultatii\_Perioada(TO\_DATE('2024-01-01', 'YYYY-MM-DD'), TO\_DATE('2024-05-9', 'YYYY-MM-DD'));

END;

## Print-screen



# 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 create. Tratați toate excepțiile care pot apărea, incluzând excepțiile predefinite NO\_DATA\_FOUND și TOO\_MANY\_ROWS. Apelați subprogramul astfel încât să evidențiați toate cazurile tratate.

## Cernița limbaj natural:

Un spital dorește să automatizeze procesul de obținere a detaliilor despre pacienții care urmează un tratament pentru o anumită afecțiune. Administratorul bazei de date dorește să implementeze un subprogram de tip funcție care să primească drept parametri următoarele informații: numele medicului si numele afectiunii. Funcția trebuie să returneze o listă de pacienți care primesc tratament pentru afecțiunea specificată, împreună cu detalii despre tratament si specialitatea medicului.

## Apelare subprogram:

CREATE OR REPLACE FUNCTION Detalii\_Tratament(

    nume\_medic IN VARCHAR2,

    denumire\_afectiune IN VARCHAR2

) RETURN VARCHAR2 IS

    -- Variabile pentru stocarea informațiilor despre pacient

    pacient\_nume   Pacient.nume%TYPE;

    pacient\_prenume Pacient.prenume%TYPE;

    detalii\_tratament Tratament.denumire%TYPE;

    specialitate\_medic Medic.specializare%TYPE;

    nr\_pacienti NUMBER;

    result VARCHAR2(32767);

BEGIN

    -- Obținerea specializării medicului

    BEGIN

        SELECT specializare

        INTO specialitate\_medic

        FROM Medic

        WHERE nume = nume\_medic;

    EXCEPTION

        WHEN NO\_DATA\_FOUND THEN

            RETURN 'Medicul ' || nume\_medic || ' nu a fost găsit.';

    END;

    -- Obținerea tratamentelor pentru afecțiunea specificată

    FOR tratament\_rec IN (

        SELECT p.nume, p.prenume, t.denumire

        FROM Tratament t

        JOIN Afectiune\_Tratament at ON t.tratament\_id = at.tratament\_id

        JOIN Afectiune a ON at.afectiune\_id = a.afectiune\_id

        JOIN Istoric\_Medical im ON at.afectiune\_tratament\_id = im.afectiune\_tratament\_id

        JOIN Pacient p ON im.pacient\_id = p.pacient\_id

        JOIN Consultatie c ON p.pacient\_id = c.pacient\_id

        JOIN Medic m ON c.medic\_id = m.medic\_id

        WHERE m.nume = nume\_medic AND a.denumire = denumire\_afectiune

    ) LOOP

        pacient\_nume := tratament\_rec.nume;

        pacient\_prenume := tratament\_rec.prenume;

        -- Verificare dacă există mai multe rânduri returnate pentru același pacient

        SELECT COUNT(\*)

        INTO nr\_pacienti

        FROM Consultatie c

        JOIN Medic m ON c.medic\_id = m.medic\_id

        JOIN Pacient p ON c.pacient\_id = p.pacient\_id

        WHERE p.nume = pacient\_nume AND p.prenume = pacient\_prenume and c.data\_consultatie = data\_consultatie;

        IF nr\_pacienti > 1 THEN

            RAISE TOO\_MANY\_ROWS;

        END IF;

        result := result || 'Pacient: ' || tratament\_rec.nume || ' ' || tratament\_rec.prenume || ' - Tratament: ' || tratament\_rec.denumire || ' - Specialitate Medic: ' || specialitate\_medic || CHR(10);

    END LOOP;

    IF result IS NULL THEN

        RETURN 'Nu există tratamente pentru medicul ' || nume\_medic || ' și afecțiunea ' || denumire\_afectiune;

    ELSE

        RETURN result;

    END IF;

EXCEPTION

    WHEN TOO\_MANY\_ROWS THEN

        RETURN 'Date ambigue pentru pacientul ' || pacient\_nume || ' ' || pacient\_prenume ;

END;

/

-- Apelare funcție

BEGIN

    --Apelare no\_data\_found

    DBMS\_OUTPUT.PUT\_LINE(Detalii\_Tratament('Dr. Ion', 'N/A'));

    --Apelare too\_many\_rows

    DBMS\_OUTPUT.PUT\_LINE(Detalii\_Tratament('Dr. Ion Popescu', 'Tahicardie'));

    --Apelare corecta

    DBMS\_OUTPUT.PUT\_LINE(Detalii\_Tratament('Dr. Elena Dumitrescu', 'Migrena'));

    --Apelare null

    DBMS\_OUTPUT.PUT\_LINE(Detalii\_Tratament('Dr. Elena Dumitrescu', 'N/A'));

END;

## Print-screen

# 9. Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat independent de tip procedură care să aibă minim 2 parametri și să utilizeze într-o singură comandă SQL 5 dintre tabelele create. Definiți minim 2 excepții proprii, altele decât cele predefinite la nivel de sistem. Apelați subprogramul astfel încât să evidențiați toate cazurile definite și tratate

## Cernița limbaj natural:

Un spital dorește să implementeze o procedură stocată care să genereze un raport detaliat despre pacienții care au fost consultați pentru o anumită afecțiune și care urmează un tratament specific. Procedura va include informații esențiale pentru a facilita urmărirea stării de sănătate a pacienților și colaborarea între medici.

Procedura va primi doi parametri:

* nume\_afectiune (tip VARCHAR2) – reprezintă numele afecțiunii pentru care se dorește raportul.
* data\_inceput (tip DATE) – reprezintă data de început pentru intervalul de timp în care să fie analizate consultațiile.

Raportul generat trebuie să includă următoarele informații:

* Numele fiecărui pacient.
* Numele medicului care l-a consultat.
* Numele afecțiunii diagnosticate.
* Numele tratamentului prescris și detalii destpre acesta.
* Data fiecărei consultații.
* Data inceperii tratamentului
* Data terminarii tratamentului

Trebuie tratate urmatoarele exceptii:

• AFECTIUNE\_INEXISTENTA

• TRATAMENT\_INEXISTENT - tratamentul este prescris de mai multe ori la aceeasi afectiune

• DATA\_INEXISTENTA - nu exista consultații pentru afecțiunea specificată la data specificată

• NO\_DATA\_FOUND

• TOO\_MANY\_ROWS

## Apelare subprogram:

CREATE OR REPLACE PROCEDURE Raport\_Afectiune(

    nume\_afectiune IN VARCHAR2,

    data\_inceput IN DATE

) IS

    -- Variabile pentru stocarea informațiilor despre pacient

    pacient\_nume   Pacient.nume%TYPE;

    pacient\_prenume Pacient.prenume%TYPE;

    medic\_nume Medic.nume%TYPE;

    denumire\_afectiune Afectiune.denumire%TYPE;

    detalii\_tratament Tratament.denumire%TYPE;

    data\_consultatie Consultatie.data\_consultatie%TYPE;

    data\_incepere\_tratament Tratament.data\_incepere%TYPE;

    data\_sfarsit\_tratament Tratament.data\_sfarsit%TYPE;

    nr\_pacienti NUMBER;

    nr\_tratamente NUMBER;

    nr\_consultatii NUMBER;

    result VARCHAR2(32767);

    ---Exceptii

    AFECTIUNE\_INEXISTENTA EXCEPTION;

    DATA\_INEXISTENTA EXCEPTION;

    TRATAMENT\_INEXISTENTA EXCEPTION;

    NO\_DATA\_FOUND EXCEPTION;

    TOO\_MANY\_ROWS EXCEPTION;

BEGIN

    -- Verificare existență afecțiune

    BEGIN

        SELECT COUNT(\*)

        INTO nr\_pacienti

        FROM Afectiune

        WHERE denumire = nume\_afectiune;

        IF nr\_pacienti = 0 THEN

            RAISE AFECTIUNE\_INEXISTENTA;

        END IF;

    END;

    -- Verificare existență tratament

    BEGIN

        SELECT COUNT(\*)

        INTO nr\_tratamente

        FROM Tratament

        JOIN Afectiune\_Tratament at ON Tratament.tratament\_id = at.tratament\_id

        JOIN Afectiune a ON at.afectiune\_id = a.afectiune\_id

        WHERE a.denumire = nume\_afectiune;

        IF nr\_tratamente = 0 THEN

            RAISE TRATAMENT\_INEXISTENTA;

        END IF;

    END;

    -- Verificare existență dată

    BEGIN

        SELECT COUNT(\*)

        INTO nr\_consultatii

        FROM Consultatie

        WHERE data\_consultatie = data\_inceput;

        IF nr\_consultatii = 0 THEN

            RAISE DATA\_INEXISTENTA;

        END IF;

    END;

    -- Obținerea tratamentelor pentru afecțiunea specificată

    FOR tratament\_rec IN (

        SELECT p.nume AS pacient\_nume, p.prenume AS pacient\_prenume, m.nume AS medic\_nume,

               a.denumire AS denumire\_afectiune, t.denumire AS detalii\_tratament, c.data\_consultatie, t.data\_incepere, t.data\_sfarsit

        FROM Tratament t

        JOIN Afectiune\_Tratament at ON t.tratament\_id = at.tratament\_id

        JOIN Afectiune a ON at.afectiune\_id = a.afectiune\_id

        JOIN Istoric\_Medical im ON at.afectiune\_tratament\_id = im.afectiune\_tratament\_id

        JOIN Pacient p ON im.pacient\_id = p.pacient\_id

        JOIN Consultatie c ON p.pacient\_id = c.pacient\_id

        JOIN Medic m ON c.medic\_id = m.medic\_id

        WHERE a.denumire = nume\_afectiune AND c.data\_consultatie >= data\_inceput

    ) LOOP

        pacient\_nume := tratament\_rec.pacient\_nume;

        pacient\_prenume := tratament\_rec.pacient\_prenume;

        medic\_nume := tratament\_rec.medic\_nume;

        denumire\_afectiune := tratament\_rec.denumire\_afectiune;

        detalii\_tratament := tratament\_rec.detalii\_tratament;

        data\_consultatie := tratament\_rec.data\_consultatie;

        data\_incepere\_tratament := tratament\_rec.data\_incepere;

        data\_sfarsit\_tratament := tratament\_rec.data\_sfarsit;

        -- Verificare dacă exista consultații pentru afecțiunea specificată la data specificată

        SELECT COUNT(\*)

        INTO nr\_consultatii

        FROM Consultatie

        WHERE pacient\_id = (

            SELECT pacient\_id

            FROM Pacient

            WHERE nume = pacient\_nume AND prenume = pacient\_prenume

        ) AND medic\_id = (

            SELECT medic\_id

            FROM Medic

            WHERE nume = medic\_nume

        ) AND data\_consultatie = data\_inceput;

        IF nr\_consultatii = 0 THEN

            RAISE NO\_DATA\_FOUND;

        END IF;

        -- Verificare dacă există mai multe rânduri returnate pentru același pacient

        SELECT COUNT(\*)

        INTO nr\_pacienti

        FROM Consultatie

        WHERE pacient\_id = (

            SELECT pacient\_id

            FROM Pacient

            WHERE nume = pacient\_nume AND prenume = pacient\_prenume

        );

        IF nr\_pacienti > 1 THEN

            RAISE TOO\_MANY\_ROWS;

        END IF;

        result := result || 'Pacient: ' || pacient\_nume || ' ' || pacient\_prenume || ' - Medic: ' || medic\_nume || ' - Afectiune: ' || denumire\_afectiune || ' - Tratament: ' || detalii\_tratament || ' - Data Consultație: ' || data\_consultatie || ' - Data Incepere Tratament: ' || data\_incepere\_tratament || ' - Data Sfarsit Tratament: ' || data\_sfarsit\_tratament || CHR(10);

    END LOOP;

    IF result IS NULL THEN

        RAISE NO\_DATA\_FOUND;

    ELSE

        DBMS\_OUTPUT.PUT\_LINE(result);

    END IF;

EXCEPTION

    WHEN AFECTIUNE\_INEXISTENTA THEN

        DBMS\_OUTPUT.PUT\_LINE('Afecțiunea ' || nume\_afectiune || ' nu există.');

    WHEN TRATAMENT\_INEXISTENTA THEN

        DBMS\_OUTPUT.PUT\_LINE('Tratament Inexistent pentru: ' || nume\_afectiune);

    WHEN DATA\_INEXISTENTA THEN

        DBMS\_OUTPUT.PUT\_LINE('Nu există consultații pentru afecțiunea ' || nume\_afectiune || ' la data ' || data\_inceput);

    WHEN NO\_DATA\_FOUND THEN

        DBMS\_OUTPUT.PUT\_LINE('Nu există date pentru afecțiunea ' || nume\_afectiune || ' la data ' || data\_inceput);

    WHEN TOO\_MANY\_ROWS THEN

        DBMS\_OUTPUT.PUT\_LINE('Date ambigue pentru pacientul ' || pacient\_nume || ' ' || pacient\_prenume);

END;

/

-- Apelare procedură

BEGIN

    --Apelare afecțiune inexistentă

    Raport\_Afectiune('Gripa', TO\_DATE('2024-01-01', 'YYYY-MM-DD'));

    --Apelare tratament inexistent

    Raport\_Afectiune('Indefinit', TO\_DATE('2024-10-02', 'YYYY-MM-DD'));

    --Apelare data inexistentă

    Raport\_Afectiune('Tahicardie', TO\_DATE('2024-01-02', 'YYYY-MM-DD'));

    --Apelare no data found

    Raport\_Afectiune('Migrena', TO\_DATE('2024-01-01', 'YYYY-MM-DD'));

    --Apelare too many rows

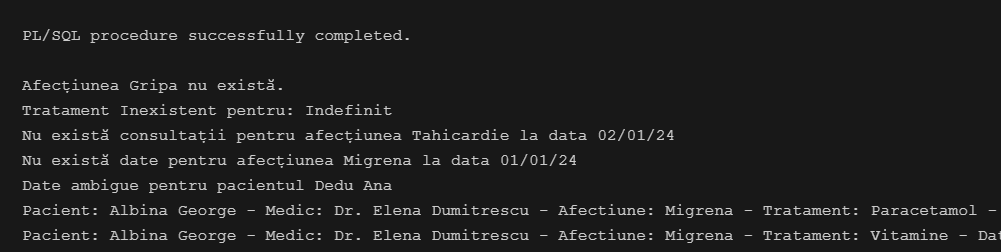
    Raport\_Afectiune('Tahicardie', TO\_DATE('2024-01-01', 'YYYY-MM-DD'));

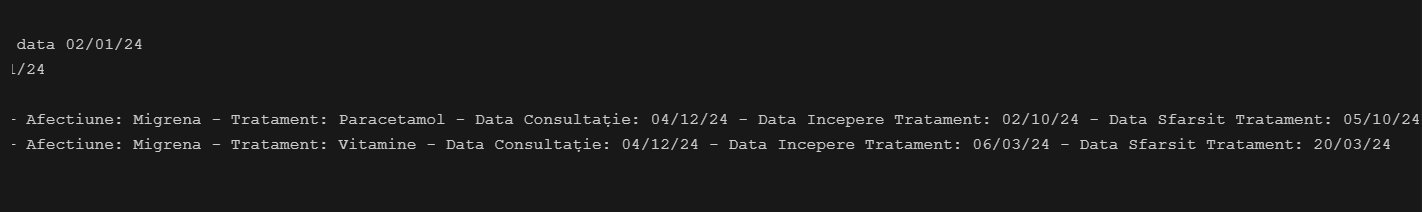
    --Apelare corectă

    Raport\_Afectiune('Migrena', TO\_DATE('2024-12-04', 'YYYY-MM-DD'));

END;

## Print-screen





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

CREATE Table Pacient\_copie AS SELECT \* FROM Pacient;

CREATE OR REPLACE TRIGGER LMD\_Pacient

AFTER INSERT OR UPDATE OR DELETE ON Pacient\_copie

BEGIN

    --Insert la ineputul anului in intervalul orar 8-12

    --Update la inceputul lunii in intervalul orar 12-16

    --Delete la sfarsitul anului in intervalul orar 8-18

    IF INSERTING THEN

        IF TO\_CHAR(SYSDATE, 'HH24') NOT BETWEEN 8 AND 12 OR To\_CHAR(SYSDATE, 'MM') <> '01' OR To\_CHAR(SYSDATE, 'DD') <> '01' THEN

            DBMS\_OUTPUT.PUT\_LINE('Insert se poate face doar pe 01.01  in intervalul orar 8-12');

            RAISE\_APPLICATION\_ERROR(-20001, 'Insert se poate face doar pe 01.01  in intervalul orar 8-12');

        END IF;

    ELSIF UPDATING THEN

        IF TO\_CHAR(SYSDATE, 'HH24') NOT BETWEEN 12 AND 16 OR To\_CHAR(SYSDATE, 'DD') <> '01' THEN

            DBMS\_OUTPUT.PUT\_LINE('Update se poate face doar la inceputul lunii in intervalul orar 12-16');

            RAISE\_APPLICATION\_ERROR(-20002, 'Update se poate face doar la inceputul lunii in intervalul orar 12-16');

        END IF;

    ELSIF DELETING THEN

        IF TO\_CHAR(SYSDATE, 'HH24') NOT BETWEEN 8 AND 18 OR To\_CHAR(SYSDATE, 'MM') <>'12' OR  To\_CHAR(SYSDATE, 'DD') <> '31' THEN

            DBMS\_OUTPUT.PUT\_LINE('Delete se poate face doar la sfarsitul anului in intervalul orar 8-18');

            RAISE\_APPLICATION\_ERROR(-20003, 'Delete se poate face doar la sfarsitul anului in intervalul orar 8-18');

        END IF;

    END IF;

END;

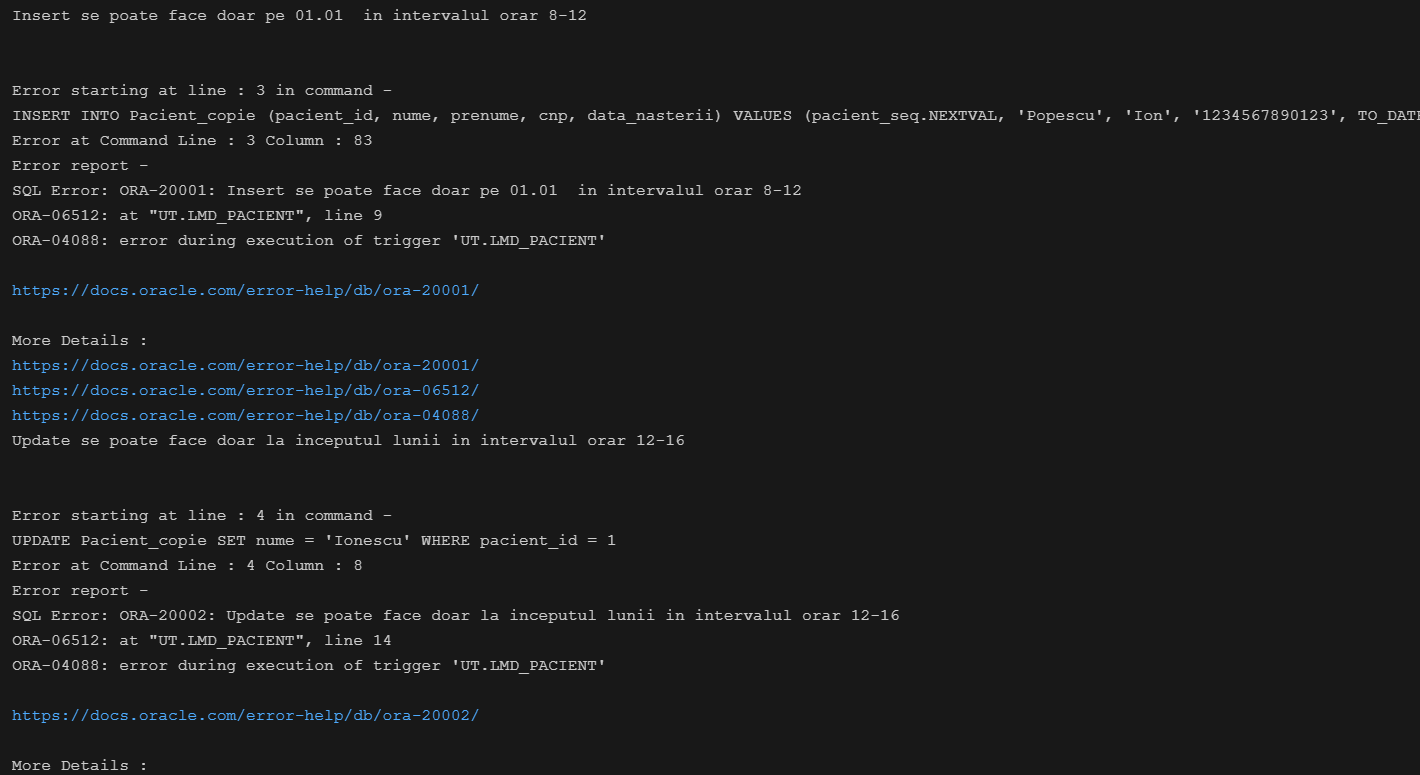
-- Apelare trigger

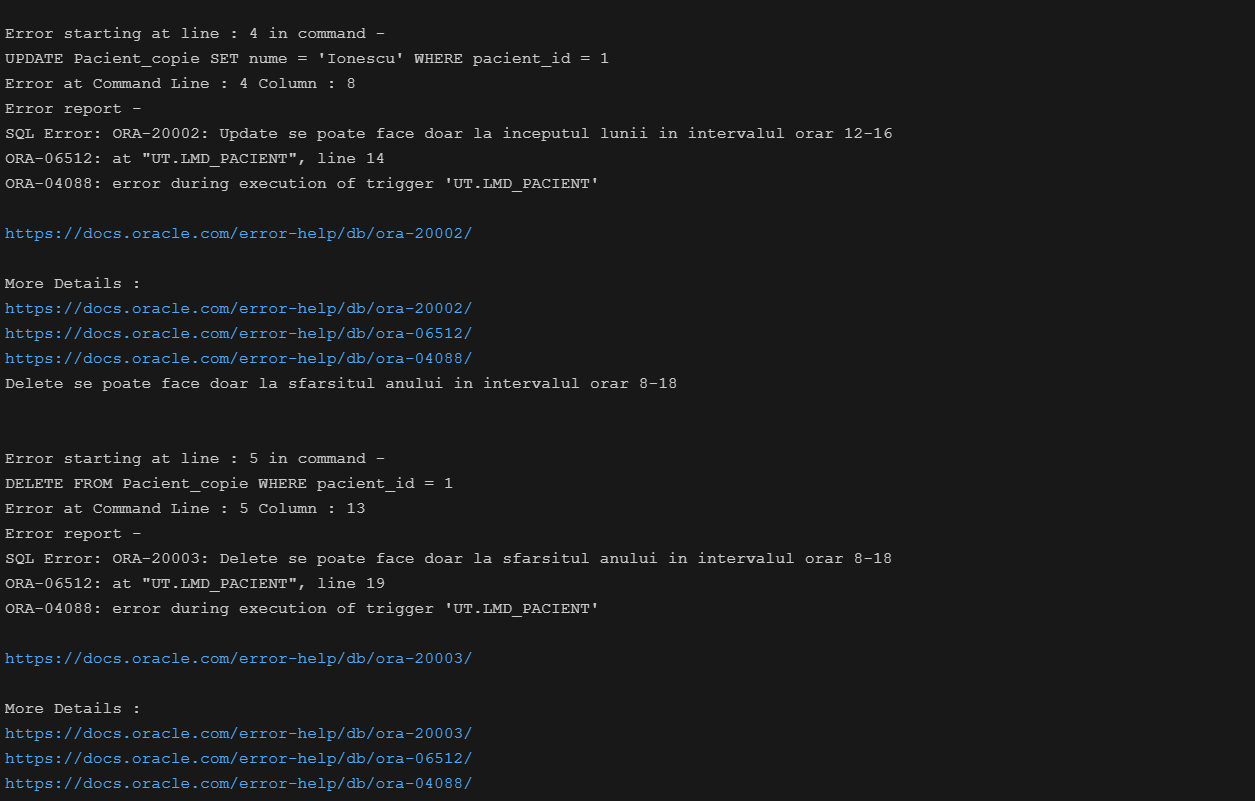
INSERT INTO Pacient\_copie (pacient\_id, nume, prenume, cnp, data\_nasterii) VALUES (pacient\_seq.NEXTVAL, 'Popescu', 'Ion', '1234567890123', TO\_DATE('1990-01-01', 'YYYY-MM-DD'));

UPDATE Pacient\_copie SET nume = 'Ionescu' WHERE pacient\_id = 1;

DELETE FROM Pacient\_copie WHERE pacient\_id = 1;

## Print-screen





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

--Creez un tabel CostProceduri care are pretul investigatiilor

CREATE TABLE CostProceduri(

    id NUMBER,

    pret NUMBER

);

--Adug in tabelul CostProceduri total pretul pentru toate investigatiile facute intr-o luna folosind un select

INSERT INTO CostProceduri(id, pret) SELECT 1, NVL(SUM(pret), 0) FROM INVESTIGATIE WHERE TO\_CHAR(data\_inceput, 'MM') = '01';

INSERT INTO CostProceduri(id, pret) SELECT 2, NVL(SUM(pret), 0) FROM INVESTIGATIE WHERE TO\_CHAR(data\_inceput, 'MM') = '02';

INSERT INTO CostProceduri(id, pret) SELECT 3, NVL(SUM(pret), 0) FROM INVESTIGATIE WHERE TO\_CHAR(data\_inceput, 'MM') = '03';

INSERT INTO CostProceduri(id, pret) SELECT 4, NVL(SUM(pret), 0) FROM INVESTIGATIE WHERE TO\_CHAR(data\_inceput, 'MM') = '04';

INSERT INTO CostProceduri(id, pret) SELECT 5, NVL(SUM(pret), 0) FROM INVESTIGATIE WHERE TO\_CHAR(data\_inceput, 'MM') = '05';

INSERT INTO CostProceduri(id, pret) SELECT 6, NVL(SUM(pret), 0) FROM INVESTIGATIE WHERE TO\_CHAR(data\_inceput, 'MM') = '06';

INSERT INTO CostProceduri(id, pret) SELECT 7, NVL(SUM(pret), 0) FROM INVESTIGATIE WHERE TO\_CHAR(data\_inceput, 'MM') = '07';

INSERT INTO CostProceduri(id, pret) SELECT 8, NVL(SUM(pret), 0) FROM INVESTIGATIE WHERE TO\_CHAR(data\_inceput, 'MM') = '08';

INSERT INTO CostProceduri(id, pret) SELECT 9, NVL(SUM(pret), 0) FROM INVESTIGATIE WHERE TO\_CHAR(data\_inceput, 'MM') = '09';

INSERT INTO CostProceduri(id, pret) SELECT 10, NVL(SUM(pret), 0) FROM INVESTIGATIE WHERE TO\_CHAR(data\_inceput, 'MM') = '10';

INSERT INTO CostProceduri(id, pret) SELECT 11, NVL(SUM(pret), 0) FROM INVESTIGATIE WHERE TO\_CHAR(data\_inceput, 'MM') = '11';

INSERT INTO CostProceduri(id, pret) SELECT 12, NVL(SUM(pret), 0) FROM INVESTIGATIE WHERE TO\_CHAR(data\_inceput, 'MM') = '12';

SELECT \* FROM CostProceduri;

CREATE TABLE Investigatie\_copie AS SELECT \* FROM Investigatie;

CREATE OR REPLACE PROCEDURE Pret\_Investigatii

    (p\_id IN NUMBER, pret\_investigatie IN NUMBER) IS

BEGIN

    UPDATE CostProceduri

    SET pret = GREATEST(pret + NVL(pret\_investigatie, 0), 0)

    WHERE id = p\_id;

END Pret\_Investigatii;

CREATE OR REPLACE TRIGGER LMD\_Investigatie

AFTER INSERT OR UPDATE OR DELETE ON Investigatie\_copie

FOR EACH ROW

BEGIN

    IF INSERTING THEN

        DBMS\_OUTPUT.PUT\_LINE('Inserting: ' || :NEW.data\_inceput || ' - ' || :NEW.pret);

        PRET\_INVESTIGATII(TO\_NUMBER(TO\_CHAR(:NEW.data\_inceput, 'MM')), :NEW.pret);

    ELSIF UPDATING THEN

        DBMS\_OUTPUT.PUT\_LINE('Updating: ' || :OLD.data\_inceput || ' - ' || :OLD.pret || ' to ' || :NEW.data\_inceput || ' - ' || :NEW.pret);

        PRET\_INVESTIGATII(TO\_NUMBER(TO\_CHAR(:OLD.data\_inceput, 'MM')), - :OLD.pret);

        PRET\_INVESTIGATII(TO\_NUMBER(TO\_CHAR(:NEW.data\_inceput, 'MM')), :NEW.pret);

    ELSIF DELETING THEN

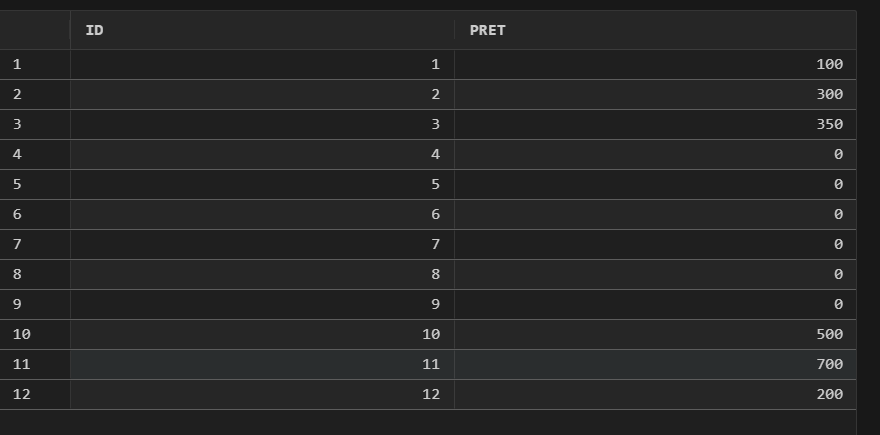
        DBMS\_OUTPUT.PUT\_LINE('Deleting: ' || :OLD.data\_inceput || ' - ' || :OLD.pret);

        PRET\_INVESTIGATII(TO\_NUMBER(TO\_CHAR(:OLD.data\_inceput, 'MM')), - :OLD.pret);

    END IF;

END;

## Print-screen



# 12. Defiiți un trigger de tip LDD. Declanșați trigger-ul.

CREATE TABLE ModificariTabele(

   nume\_bd VARCHAR2(100),

   user\_bd VARCHAR2(100),

   eveniment VARCHAR2(100),

   nume VARCHAR2(100),

   tip VARCHAR2(100),

   data TIMESTAMP(3)

);

--Creez trigger-ul

CREATE OR REPLACE TRIGGER Trg\_LDD

AFTER DROP OR ALTER OR CREATE ON SCHEMA

BEGIN

    INSERT INTO ModificariTabele(nume\_bd, user\_bd, eveniment, nume, tip, data)

    VALUES (ora\_database\_name, ora\_login\_user, ora\_sysevent, ora\_dict\_obj\_name, ora\_dict\_obj\_type, SYSTIMESTAMP);

END;

Select \* from ModificariTabele;

CREATE table Medic\_copie AS SELECT \* FROM Medic;

SELECT \* FROM ModificariTabele;

ALTER TABLE Medic\_copie ADD data\_angajare DATE;

SELECT \* FROM ModificariTabele;

DROP TABLE Medic\_copie;

SELECT \* FROM ModificariTabele;

Create or replace TYPE vectori as object

(

    x number,

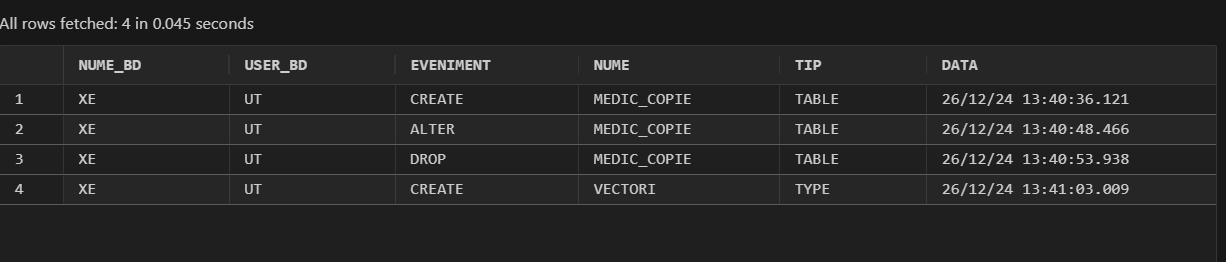
    y number

);

Create or replace type vectori\_t as table of vectori;

SELECT \* FROM ModificariTabele;

## Print-screen



# 13. Formulați în limbaj natural o problemă pe care să o rezolvați folosind un pachet care să include tipuri de date complexe și obiecte necesare unui flux de acțiuni integrate, specifice bazei de date.

## Limbaj natural:

Problema: Un spital dorește să automatizeze procesul de gestionare a afecțiunilor pacienților.

Pentru aceasta, se dorește implementarea unui pachet care să conțină următoarele funcționalități:

1. Crearea unei liste de tratamente pentru o afecțiune specificată.

2. Numarul de pacienti afectati de o anumita afectiune.

3. Afisarea pacientilor care au o anumita afectiune.(nume, prenume, varsta)

4. Durata medie a tratamentului pentru o anumita afectiune si daca depseste 3 luni sa fie incadrata in categoria "Boli Cronice"

5. La afectiunea cel mai des intalnita sa se afiseze un mesaj de avertizare, daca sunt mai multe se vor afisa toate

6. Afisarea tuturor datelor

-- Creerea tipului de obiect Tratament\_Obj

CREATE OR REPLACE TYPE Tratament\_Obj AS OBJECT(

    denumire VARCHAR2(100),

    durata NUMBER

);

-- Crearea tipului de tabel Tratament\_T

CREATE OR REPLACE TYPE Tratament\_T IS TABLE OF Tratament\_Obj;

-- Crearea pachetului

CREATE OR REPLACE PACKAGE Gestionare\_Afectiuni AS

    -- Crearea unei liste de tratamente pentru o afecțiune specificată

    FUNCTION Creare\_Tratamente(

        denumire\_afectiune IN VARCHAR2

    ) RETURN Tratament\_T;

    -- Numarul de pacienti afectati de o anumita afectiune

    FUNCTION Numar\_Pacienti(

        denumire\_afectiune IN VARCHAR2

    ) RETURN NUMBER;

    -- Afisarea pacientilor care au o anumita afectiune

    PROCEDURE Afisare\_Pacienti(

        denumire\_afectiune IN VARCHAR2

    );

    -- Durata medie a tratamentului pentru o anumita afectiune

    FUNCTION Durata\_Medie\_Tratament(

        denumire\_afectiune IN VARCHAR2

    ) RETURN NUMBER;

    -- Verificare daca durata medie a tratamentului depaseste 3 luni

    FUNCTION Verificare\_Durata\_Medie(

        denumire\_afectiune IN VARCHAR2

    ) RETURN VARCHAR2;

    --Functie care returneaza numarul de pacienti afectati de cele mai intalnite afectiuni

    FUNCTION Numar\_Pacienti\_Afectiuni RETURN NUMBER;

    --Procedura pentru afisarea celor mai intalnite afectiuni

    PROCEDURE AfisareAfectiuni;

    -- Procedura pentru afisare

    PROCEDURE Afisare;

END Gestionare\_Afectiuni;

-- Implementare pachet

CREATE OR REPLACE PACKAGE BODY Gestionare\_Afectiuni AS

    -- Crearea unei liste de tratamente pentru o afecțiune specificată

    FUNCTION Creare\_Tratamente(

        denumire\_afectiune IN VARCHAR2

    ) RETURN Tratament\_T IS

        tratamente Tratament\_T := Tratament\_T();

    BEGIN

        FOR tr IN (

            SELECT DISTINCT t.denumire, Floor(MONTHS\_BETWEEN(data\_sfarsit, data\_incepere)) as durata

            FROM Tratament t

            JOIN Afectiune\_Tratament at ON t.tratament\_id = at.tratament\_id

            JOIN Afectiune a ON at.afectiune\_id = a.afectiune\_id

            WHERE a.denumire = denumire\_afectiune

        ) LOOP

            tratamente.EXTEND; -- Extindem tabelul de tratamente

            tratamente(tratamente.LAST) := Tratament\_Obj(tr.denumire, tr.durata); -- Adaugam un nou tratament

        END LOOP;

        RETURN tratamente;

    END Creare\_Tratamente;

    -- Numarul de pacienti afectati de o anumita afectiune

    FUNCTION Numar\_Pacienti(

        denumire\_afectiune IN VARCHAR2

    ) RETURN NUMBER IS

        nr\_pacienti NUMBER;

    BEGIN

        SELECT COUNT(DISTINCT p.pacient\_id)

        INTO nr\_pacienti

        FROM Pacient p

        JOIN Istoric\_Medical im ON p.pacient\_id = im.pacient\_id

        JOIN Afectiune\_Tratament at ON im.afectiune\_tratament\_id = at.afectiune\_tratament\_id

        JOIN Afectiune a ON at.afectiune\_id = a.afectiune\_id

        WHERE a.denumire = denumire\_afectiune;

        RETURN nr\_pacienti;

    END Numar\_Pacienti;

    -- Afisarea pacientilor care au o anumita afectiune

    PROCEDURE Afisare\_Pacienti(

        denumire\_afectiune IN VARCHAR2

    ) IS

    BEGIN

        IF Numar\_Pacienti(denumire\_afectiune) = 0 THEN

            DBMS\_OUTPUT.PUT\_LINE('Nu exista pacienti pentru afectiunea ' || denumire\_afectiune);

            RETURN;

        END IF;

        FOR pacient IN (

            SELECT DISTINCT p.nume, p.prenume,  EXTRACT(YEAR FROM p.data\_nasterii)- EXTRACT(YEAR FROM SYSDATE) AS varsta

            FROM Pacient p

            JOIN Istoric\_Medical im ON p.pacient\_id = im.pacient\_id

            JOIN Afectiune\_Tratament at ON im.afectiune\_tratament\_id = at.afectiune\_tratament\_id

            JOIN Afectiune a ON at.afectiune\_id = a.afectiune\_id

            WHERE a.denumire = denumire\_afectiune

        ) LOOP

            DBMS\_OUTPUT.PUT\_LINE('Pacient: ' || pacient.nume || ' ' || pacient.prenume || ' - Varsta: ' || pacient.varsta);

        END LOOP;

    END Afisare\_Pacienti;

    -- Durata medie a tratamentului pentru o anumita afectiune

    FUNCTION Durata\_Medie\_Tratament(

        denumire\_afectiune IN VARCHAR2

    ) RETURN NUMBER IS

        durata\_medie NUMBER;

    BEGIN

        SELECT Floor(AVG(MONTHS\_BETWEEN(t.data\_sfarsit, t.data\_incepere)))

        INTO durata\_medie

        FROM Tratament t

        JOIN Afectiune\_Tratament at ON t.tratament\_id = at.tratament\_id

        JOIN Afectiune a ON at.afectiune\_id = a.afectiune\_id

        WHERE a.denumire = denumire\_afectiune;

        IF durata\_medie IS NULL THEN

            durata\_medie := 0;

        END IF;

        RETURN durata\_medie;

    END Durata\_Medie\_Tratament;

    -- Verificare daca durata medie a tratamentului depaseste 3 luni

    FUNCTION Verificare\_Durata\_Medie(

        denumire\_afectiune IN VARCHAR2

    ) RETURN VARCHAR2 IS

        durata\_medie NUMBER;

    BEGIN

        durata\_medie := Durata\_Medie\_Tratament(denumire\_afectiune);

        IF durata\_medie > 3 THEN

            RETURN 'Boli Cronice';

        ELSE

            RETURN 'Boli Acute';

        END IF;

    END Verificare\_Durata\_Medie;

    --Functie care returneaza numarul de pacienti afectati de cele mai intalnite afectiuni

    FUNCTION Numar\_Pacienti\_Afectiuni RETURN NUMBER IS

        nr\_pacienti NUMBER;

    BEGIN

        SELECT COUNT(DISTINCT p.pacient\_id)

        INTO nr\_pacienti

        FROM Pacient p

        JOIN Istoric\_Medical im ON p.pacient\_id = im.pacient\_id

        JOIN Afectiune\_Tratament at ON im.afectiune\_tratament\_id = at.afectiune\_tratament\_id

        JOIN Afectiune a ON at.afectiune\_id = a.afectiune\_id

        GROUP BY a.denumire

        ORDER BY COUNT(DISTINCT p.pacient\_id) DESC

        FETCH FIRST ROW ONLY;

        RETURN nr\_pacienti;

    END Numar\_Pacienti\_Afectiuni;

    --Procedura pentru afisarea celor mai intalnite afectiuni

    PROCEDURE AfisareAfectiuni IS

    BEGIN

        FOR afectiune\_rec IN (

            SELECT a.denumire, COUNT(DISTINCT p.pacient\_id) AS nr\_pacienti

            FROM Afectiune a

            JOIN Afectiune\_Tratament at ON a.afectiune\_id = at.afectiune\_id

            JOIN Istoric\_Medical im ON at.afectiune\_tratament\_id = im.afectiune\_tratament\_id

            JOIN Pacient p ON im.pacient\_id = p.pacient\_id

            GROUP BY a.denumire

            ORDER BY nr\_pacienti DESC

        ) LOOP

            DBMS\_OUTPUT.PUT\_LINE('Afectiune: ' || afectiune\_rec.denumire || ' - Numar Pacienti: ' || afectiune\_rec.nr\_pacienti);

        END LOOP;

    END AfisareAfectiuni;

    -- Procedura pentru afisare

    PROCEDURE Afisare IS

    --Creez un cursor pentru a afisa toate afectiunile

    CURSOR c\_afectiuni IS

        SELECT DISTINCT a.denumire

        FROM Afectiune a;

    BEGIN

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

        FOR afectiune\_rec IN c\_afectiuni LOOP

            DBMS\_OUTPUT.PUT\_LINE('Afectiune: ' || afectiune\_rec.denumire);

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

                IF Numar\_Pacienti(afectiune\_rec.denumire) = Numar\_Pacienti\_Afectiuni THEN

                    DBMS\_OUTPUT.PUT\_LINE('--Cea mai intalnita afectiune--');

                END IF;

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

            IF Creare\_Tratamente(afectiune\_rec.denumire).COUNT = 0 THEN

                DBMS\_OUTPUT.PUT\_LINE('Nu exista tratamente pentru afectiunea ' || afectiune\_rec.denumire);

            ELSE

            FOR i IN 1..Creare\_Tratamente(afectiune\_rec.denumire).COUNT LOOP

                DBMS\_OUTPUT.PUT\_LINE('Denumire: ' || Creare\_Tratamente(afectiune\_rec.denumire)(i).denumire || ' - Durata: ' || Creare\_Tratamente(afectiune\_rec.denumire)(i).durata);

            END LOOP;

            END IF;

            IF Numar\_Pacienti(afectiune\_rec.denumire) = 0 THEN

                DBMS\_OUTPUT.PUT\_LINE('Nu exista pacienti pentru afectiunea ' || afectiune\_rec.denumire);

            ELSE

                DBMS\_OUTPUT.PUT\_LINE('Numar Pacienti: ' || Numar\_Pacienti(afectiune\_rec.denumire));

            END IF;

            DBMS\_OUTPUT.PUT\_LINE('Durata Medie Tratament: ' || Durata\_Medie\_Tratament(afectiune\_rec.denumire));

            DBMS\_OUTPUT.PUT\_LINE('Tip Afectiune: ' || Verificare\_Durata\_Medie(afectiune\_rec.denumire));

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

            Afisare\_Pacienti(afectiune\_rec.denumire);

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

        END LOOP;

        DBMS\_OUTPUT.PUT\_LINE('Top afectiuni:');

        AfisareAfectiuni;

    END Afisare;

END Gestionare\_Afectiuni;

-- Apelare pachet

BEGIN

    Gestionare\_Afectiuni.Afisare;

END;

## Print-screen

