

SGBD Project

Macovei Catalina
Grupa 242

1. Utilitatea bazei de date.....	2
2. Diagrama Entitate-Relație ERD.....	3
3. Diagrama conceptuală.....	4
4. Implementarea diagramei conceptuale.....	5
5. Adăugare informații în tabele - Inserări.....	13
6. Subprogram stocat independent - Colecții.....	30
7. Subprogram stocat independent - 2 tipuri diferite de cursoare.....	33
8. Subprogram stocat - funcție & excepții proprii.....	36
9. Procedură: tratare excepții TOO_MANY_ROWS & NO_DATA_FOUND.....	39
10. Definiți un trigger de tip LMD la nivel de linie.....	42
11. Definiți un trigger de tip LMD la nivel de comandă 11. Definiți un trigger de tip LMD la nivel de comandă. Declanșați trigger-ul.....	46
12. Definiți un trigger de tip LDD. Declanșați trigger-ul.....	47
13. Definiți un pachet care să conțină toate obiectele definite în cadrul proiectului.....	49
14. Definiți un pachet care să includă tipuri de date complexe și obiecte necesare unui flux de acțiuni integrate, specific bazei de date definite (minim 2 tipuri de date, minim 2 funcții, minim 2 proceduri).....	57

Utilitatea bazei de date

1. Descrierea TEMEI DE PROIECT:

"MediCare" este o rețea de spitale. Baza sa de date este un sistem complex care gestionează relațiile dintre spitale, medici, pacienți și secții.

Spitalele sunt localizate în zone ale sectoarelor, iar fiecare spital poate avea mai multe secții specializate, cum ar fi cardiologie, reanimare, ortopedie, ginecologie și obstetrică, pediatrie și neurologie.

Medicii pot lucra într-un spital sau mai multe spitale și sunt repartizați în secții în funcție de specializare. Pacienții pot avea mai multe internări în secții. Înainte de fiecare tratament, ei sunt programați la consultații, unde li se stabilește un diagnostic. Tratamentul poate fi de diverse tipuri, inclusiv intervenții chirurgicale, tratamente medicamentoase sau fizioterapie, în funcție de afecțiunea diagnosticată.

În plus, pacienții au un istoric medical actualizat periodic, care conține informații despre patologii anterioare și contraindicații pentru diverse preparate.

Baza de date "MediCare" facilitează gestionarea acestor relații complexe pentru a asigura o îngrijire medicală de calitate în cadrul acestei rețele de spitale.

Diagrama Entitate-Relație ERD

2. Diagrama entitate relație:

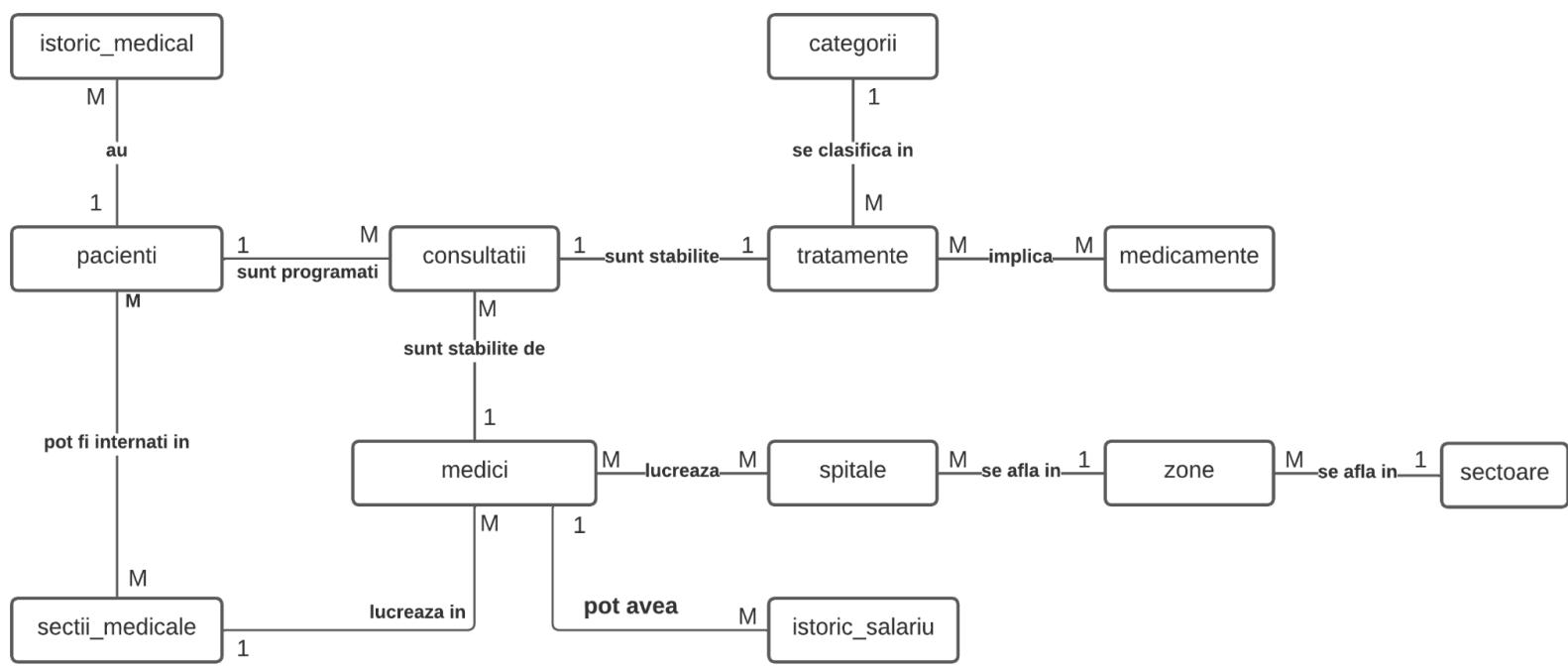
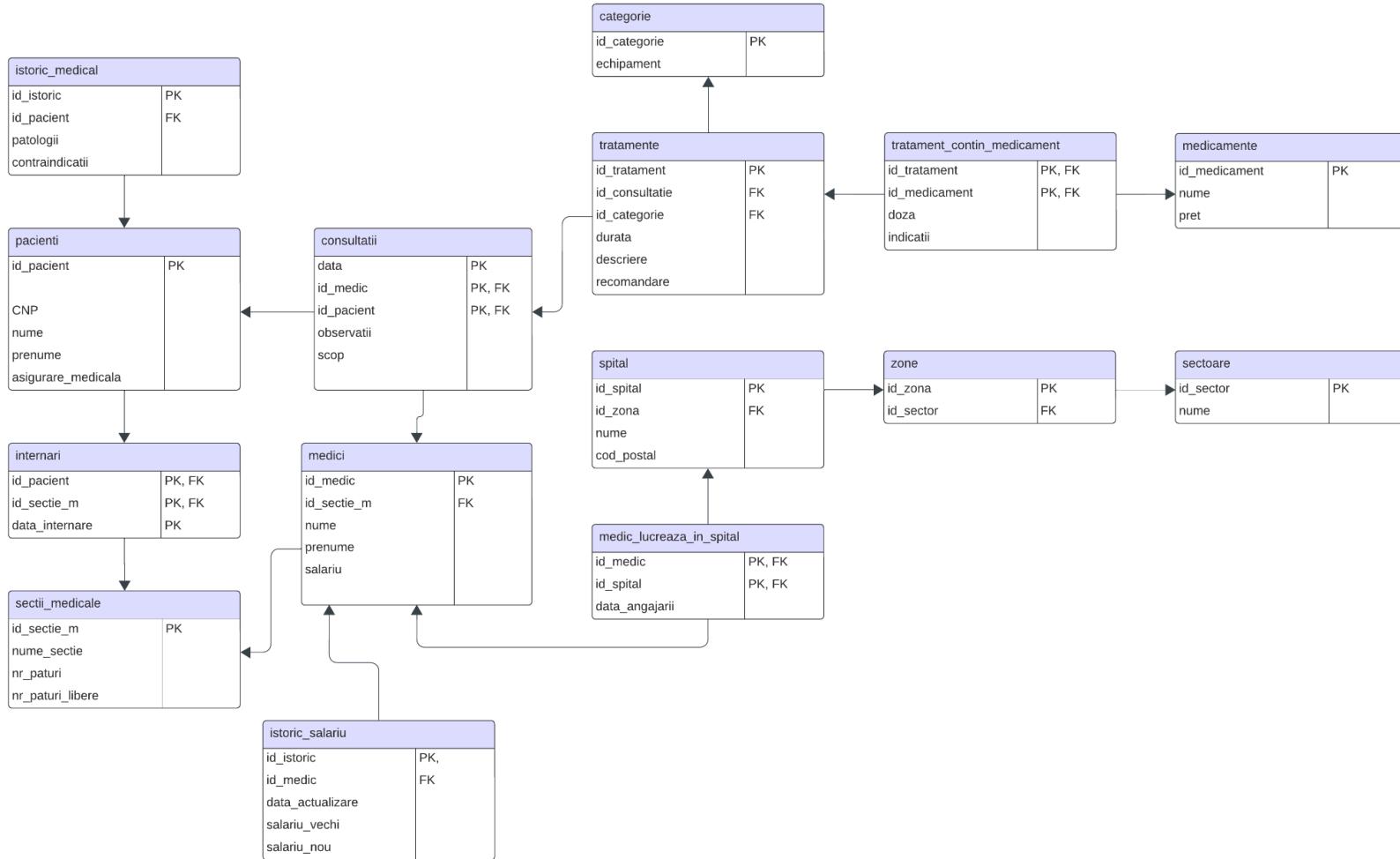


Diagrama conceptuală

3. Diagrama conceptuală:

* FK ID_CONSULTATIE cheie compusă, formată din: (ID_PACIENT FK, ID_MEDIC FK, DATA FK)



Implementarea diagramei conceptuale

4. Definirea tabelelor, constrângerilor necesare

Această secțiune conține:

- Cod sursă
- Print Screen

```
CREATE TABLE SECTII_MEDICALE (
    ID_SECTIE_M VARCHAR2(20) PRIMARY KEY,
    NUME_SECTIE VARCHAR2(100),
    NR_PATURI NUMBER(4),
    NR_PATURI_LIBERE NUMBER(4)
);
```

```
CREATE TABLE PACIENTI (
    ID_PACIENT NUMBER(4) PRIMARY KEY,
    CNP NUMBER(13),
    NUME VARCHAR2(100),
    PRENUME VARCHAR2(100),
    ASIGURARE_MEDICALA VARCHAR2(20)
);
```

```
CREATE TABLE INTERNARI (
    ID_PACIENT NUMBER(4),
    ID_SECTIE_M VARCHAR2(20),
    DATA_INTERNARE DATE,
    CONSTRAINT ID_INTERNARE PRIMARY KEY (ID_PACIENT, ID_SECTIE_M,
DATA_INTERNARE),
    CONSTRAINT FK_ID_PACIENT FOREIGN KEY (ID_PACIENT)
        REFERENCES PACIENTI(ID_PACIENT),
    CONSTRAINT FK_ID_SECTIE_M FOREIGN KEY (ID_SECTIE_M)
        REFERENCES SECTII_MEDICALE(ID_SECTIE_M)
);
```

```
CREATE TABLE ISTORIC_MEDICAL (
    ID_PACIENT NUMBER(4),
    ID_ISTORIC NUMBER(4) PRIMARY KEY,
    PATOLOGII VARCHAR2(100),
    CONTRAINDICATII VARCHAR2(100),
    CONSTRAINT FK_ID_PACIENT1 FOREIGN KEY (ID_PACIENT)
        REFERENCES PACIENTI(ID_PACIENT)
```

);

```
CREATE TABLE MEDICI (
    ID_MEDIC NUMBER(4) PRIMARY KEY,
    NUME VARCHAR2(100),
    PRENUME VARCHAR2(100),
    SALARIU NUMBER(10),
    MEDIC_SEF VARCHAR(100),
    ID_SECTIE_M VARCHAR2(20),
    CONSTRAINT FK_ID_SECTIE_M1 FOREIGN KEY (ID_SECTIE_M)
        REFERENCES SECTII_MEDICALE(ID_SECTIE_M)
```

);

```
CREATE TABLE ISTORIC_SALARIU (
    ID_ISTORIC_SAL NUMBER(4) PRIMARY KEY,
    ID_MEDIC NUMBER(4),
    SALARIU_VECI NUMBER(10),
    SALARIU_NOU NUMBER(10),
    DATA_ACTUALIZARE DATE,
    CONSTRAINT FK_ID_SAL_M1 FOREIGN KEY (ID_MEDIC)
        REFERENCES MEDICI(ID_MEDIC)
```

);

```
CREATE TABLE CONSULTATII (
    data_cons DATE,
    ID_MEDIC NUMBER(4),
    ID_PACIENT NUMBER(4),
    CONSTRAINT ID_CONSULTATIE_P PRIMARY KEY (ID_PACIENT, ID_MEDIC,
    data_cons),
    CONSTRAINT FK_ID_PACIENT2 FOREIGN KEY (ID_PACIENT)
        REFERENCES PACIENTI(ID_PACIENT),
    CONSTRAINT FK_ID_MEDIC FOREIGN KEY (ID_MEDIC)
        REFERENCES MEDICI(ID_MEDIC)
```

);

```
CREATE TABLE CATEGORIE (
    ID_CATEGORIE VARCHAR2(100) PRIMARY KEY,
    ECHIPAMENT VARCHAR2(100)
```

);

```

CREATE TABLE TRATAMENTE (
    ID_TRATAMENT NUMBER(4) PRIMARY KEY,
    data_cons DATE,
    ID_CATEGORIE VARCHAR2(100),
    DURATA VARCHAR2(20),
    DESCRIERE VARCHAR2(150),
    RECOMANDARI VARCHAR2(150),
    ID_MEDIC NUMBER(4),
    ID_PACIENT NUMBER(4),
    CONSTRAINT FK_ID_CONSULTATIE112 FOREIGN KEY (data_cons, ID_PACIENT,
    ID_MEDIC)
        REFERENCES CONSULTATII(data_cons, ID_PACIENT, ID_MEDIC),
    CONSTRAINT FK_ID_CATEGORIE112 FOREIGN KEY (ID_CATEGORIE)
        REFERENCES CATEGORIE(ID_CATEGORIE)
);

```

```

CREATE TABLE MEDICAMENTE (
    ID_MEDICAMENT VARCHAR2(20) PRIMARY KEY,
    NUME VARCHAR2(100),
    PRET NUMBER(10)
);

```

```

CREATE TABLE TRATAMENTE_CONTIN_MEDICAMENTE (
    ID_TRATAMENT NUMBER(4),
    ID_MEDICAMENT VARCHAR2(20),
    DOZA VARCHAR2(100),
    INDICATII VARCHAR2(100),
    CONSTRAINT FK_ID_TRATAMENT FOREIGN KEY (ID_TRATAMENT)
        REFERENCES TRATAMENTE(ID_TRATAMENT),
    CONSTRAINT FK_IID_MEDICAMENT FOREIGN KEY (ID_MEDICAMENT)
        REFERENCES MEDICAMENTE(ID_MEDICAMENT),
    CONSTRAINT ID_MEDICAMENT_TR PRIMARY KEY (ID_TRATAMENT,
    ID_MEDICAMENT)
);

```

```

CREATE TABLE SECTOARE (
    ID_SECTOR VARCHAR2(20) PRIMARY KEY,
    NUME VARCHAR2(20)
);

```

```
CREATE TABLE ZONE (
    ID_ZONA VARCHAR2(20) PRIMARY KEY,
    ID_SECTOR VARCHAR2(20),
    CONSTRAINT FK_ID_SECTOR FOREIGN KEY (ID_SECTOR)
        REFERENCES SECTOARE(ID_SECTOR)
```

);

```
CREATE TABLE SPITAL (
    ID_SPITAL VARCHAR2(20) PRIMARY KEY,
    NUME_SPITAL VARCHAR2(100),
    COD_POSTAL VARCHAR2(20),
    ID_ZONA VARCHAR2(20),
    CONSTRAINT FK_ID_ZONA FOREIGN KEY (ID_ZONA)
        REFERENCES ZONE(ID_ZONA)
```

);

```
CREATE TABLE MEDIC_LUCREAZA_IN_SPITAL (
    ID_MEDIC NUMBER(4),
    ID_SPITAL VARCHAR2(20),
    DATA_ANGAJARII DATE,
    CONSTRAINT FK_ID_SPITAL FOREIGN KEY (ID_SPITAL)
        REFERENCES SPITAL(ID_SPITAL),
    CONSTRAINT FK_ID_MEDIC1 FOREIGN KEY (ID_MEDIC)
        REFERENCES MEDICI(ID_MEDIC),
    CONSTRAINT ID_MEDIC_LUCREAZA_IN_SPITAL PRIMARY KEY (ID_MEDIC,
    ID_SPITAL)
);
```

...ql 242_Macovei_Catalina_tema3.sql 242_Macovei_Catalina_inserari.sql

SQL Worksheet History

Worksheet Query Builder

```
CREATE TABLE SECTII_MEDICALE (
    ID_SECTIE_M VARCHAR2(20) PRIMARY KEY,
    NUME_SECTIE VARCHAR2(100),
    NR_PATURI NUMBER(4)
);

CREATE TABLE PACIENTI (
    ID_PACIENT NUMBER(4) PRIMARY KEY,
    CNP NUMBER(13),
    NUME VARCHAR2(100),
    PRENUME VARCHAR2(100),
    ASIGURARE_MEDICALA VARCHAR2(20),
    ID_SECTIE_M VARCHAR2(20),
    FOREIGN KEY (ID_SECTIE_M) REFERENCES SECTII_MEDICALE(ID_SECTIE_M)
);

CREATE TABLE INTERNARI (
    ID_PACIENT NUMBER(4),
    ID_SECTIE_M VARCHAR2(20),
    DATA_INTERNARE DATE,
    CONSTRAINT ID_INTERNARE PRIMARY KEY (ID_PACIENT, ID_SECTIE_M, DATA_INTERNAR),
    CONSTRAINT FK_ID_PACIENT FOREIGN KEY (ID_PACIENT)
        REFERENCES PACIENTI(ID_PACIENT),
    CONSTRAINT FK_ID_SECTIE_M FOREIGN KEY (ID_SECTIE_M)
        REFERENCES SECTII_MEDICALE(ID_SECTIE_M)
```

Script Output

Task completed in 0.148 seconds

Table SECTII_MEDICALE created.

Table PACIENTI created.

Table INTERNARI created.

...ql 242_Macovei_Catalina_tema3.sql 242_Macovei_Catalina_inserari.sql

SQL Worksheet History

Worksheet Query Builder

```
CREATE TABLE MEDICI (
    ID_MEDIC NUMBER(4) PRIMARY KEY,
    NUME VARCHAR2(100),
    PRENUME VARCHAR2(100),
    SALARIU NUMBER(10),
    MEDIC_SEF VARCHAR(100),
    ID_SECTIE_M VARCHAR2(20),
    CONSTRAINT FK_ID_SECTIE_M1 FOREIGN KEY (ID_SECTIE_M)
        REFERENCES SECTII_MEDICALE(ID_SECTIE_M)
);

CREATE TABLE CONSULTATII (
    ID_CONSULTATIE NUMBER(4),
    ID_MEDIC NUMBER(4),
    ID_PACIENT NUMBER(4),
    CONSTRAINT ID_CONSULTATIE_PK PRIMARY KEY (ID_PACIENT, ID_MEDIC, ID_CONSULTATIE)
    CONSTRAINT FK_ID_PACIENT2 FOREIGN KEY (ID_PACIENT)
        REFERENCES PACIENTI(ID_PACIENT),
    CONSTRAINT FK_ID_MEDIC FOREIGN KEY (ID_MEDIC)
        REFERENCES MEDICI(ID_MEDIC)
);

CREATE TABLE CATEGORIE (
    ID_CATEGORIE VARCHAR2(100) PRIMARY KEY.
```

Script Output

Task completed in 0.167 seconds

Table MEDICI created.

Table CONSULTATII created.

Table CATEGORIE created.

...ql 242_Macovei_Catalina_tema3.sql x 242_Macovei_Catalina_inserari.sql x DropD

SQL Worksheet History

Worksheet Query Builder

```
CREATE TABLE TRATAMENTE (
    ID_TRATAMENT NUMBER(4) PRIMARY KEY,
    ID_CONSULTATIE NUMBER(4),
    ID_CATEGORIE VARCHAR2(100),
    DURATA VARCHAR2(20),
    DESCRIERE VARCHAR2(150),
    RECOMANDARI VARCHAR2(150),
    ID_MEDIC NUMBER(4),
    ID_PACIENT NUMBER(4),
    CONSTRAINT FK_ID_CONSULTATIE112 FOREIGN KEY (ID_CONSULTATIE, ID_PACIENT, ID_MEDIC)
        REFERENCES CONSULTATII(ID_CONSULTATIE, ID_PACIENT, ID_MEDIC),
    CONSTRAINT FK_ID_CATEGORIE112 FOREIGN KEY (ID_CATEGORIE)
        REFERENCES CATEGORIE(ID_CATEGORIE)
);

CREATE TABLE MEDICAMENTE (
    ID_MEDICAMENT VARCHAR2(20) PRIMARY KEY,
    NUME VARCHAR2(100),
    PRET NUMBER(10)
);

CREATE TABLE TRATAMENTE_CONTIN_MEDICAMENTE (
    ID_TRATAMENT NUMBER(4),
    TD_MEDICAMENT VARCHAR2(20).
```

Script Output X

Task completed in 0.116 seconds

Table CATEGORIE created.

Table TRATAMENTE created.

Table MEDICAMENTE created.

Table TRATAMENTE_CONTIN_MEDICAMENTE created.

...ql 242_Macovei_Catalina_tema3.sql 242_Macovei_Catalina_inserari.sql

SQL Worksheet History

Worksheet Query Builder

```
CREATE TABLE SECTOARE (
    ID_SECTOR VARCHAR2(20) PRIMARY KEY,
    NUME VARCHAR2(20)
);

CREATE TABLE ZONE (
    ID_ZONA VARCHAR2(20) PRIMARY KEY,
    ID_SECTOR VARCHAR2(20),
    CONSTRAINT FK_ID_SECTOR FOREIGN KEY (ID_SECTOR)
        REFERENCES SECTOARE(ID_SECTOR)
);

CREATE TABLE SPITAL (
    ID_SPITAL VARCHAR2(20) PRIMARY KEY,
    NUME_SPITAL VARCHAR2(100),
    COD_POSTAL VARCHAR2(20),
    ID_ZONA VARCHAR2(20),
    CONSTRAINT FK_ID_ZONA FOREIGN KEY (ID_ZONA)
        REFERENCES ZONE(ID_ZONA)
);

CREATE TABLE MEDIC_LUCREAZA_IN_SPITAL (
    ID_MEDIC NUMBER(4),
    ID_SPITAL VARCHAR2(20).
```

Script Output

Task completed in 0.170 seconds

```
Table SECTOARE created.

Table ZONE created.

Table SPITAL created.

Table MEDIC_LUCREAZA_IN_SPITAL created.
```

Adăugare informații în tabele - Inserări

5. Adăugarea informațiilor necesare în tabele.

Această secțiune conține:

- Cod sursă
- Print Screen

```
INSERT INTO SECTII_MEDICALE (ID_SECTIE_M, NUME_SECTIE, NR_PATURI,
NR_PATURI_LIBERE)
```

```
VALUES ('SEC1', 'Chirurgie', 30, 25);
```

```
INSERT INTO SECTII_MEDICALE (ID_SECTIE_M, NUME_SECTIE, NR_PATURI,
NR_PATURI_LIBERE)
```

```
VALUES ('SEC2', 'Cardiologie', 25, 20);
```

```
INSERT INTO SECTII_MEDICALE (ID_SECTIE_M, NUME_SECTIE, NR_PATURI,
NR_PATURI_LIBERE)
```

```
VALUES ('SEC3', 'Pediatrie', 40, 35);
```

```
INSERT INTO SECTII_MEDICALE (ID_SECTIE_M, NUME_SECTIE, NR_PATURI,
NR_PATURI_LIBERE)
```

```
VALUES ('SEC4', 'Ortopedie', 20, 1);
```

```
INSERT INTO SECTII_MEDICALE (ID_SECTIE_M, NUME_SECTIE, NR_PATURI,
NR_PATURI_LIBERE)
```

```
VALUES ('SEC5', 'Neurologie', 15, 10);
```

```
INSERT INTO SECTII_MEDICALE (ID_SECTIE_M, NUME_SECTIE, NR_PATURI,
NR_PATURI_LIBERE)
```

```
VALUES ('SEC6', 'Oftalmologie', 18, 10);
```

--Inserare Pacienti

```
INSERT INTO PACIENTI (ID_PACIENT, CNP, NUME, PRENUME, ASIGURARE_MEDICALA)
VALUES (1, 1234567890123, 'Popescu', 'Ana', 'Da');
```

```
INSERT INTO PACIENTI (ID_PACIENT, CNP, NUME, PRENUME, ASIGURARE_MEDICALA)
VALUES (2, 2345678901234, 'Ionescu', 'Mihai', 'Nu');
```

```
INSERT INTO PACIENTI (ID_PACIENT, CNP, NUME, PRENUME, ASIGURARE_MEDICALA)
VALUES (3, 3456789012345, 'Dumitrescu', 'Maria', 'Da');
```

```
INSERT INTO PACIENTI (ID_PACIENT, CNP, NUME, PRENUME, ASIGURARE_MEDICALA)
VALUES (4, 4567890123456, 'Georgescu', 'Ion', 'Nu');
```

```
INSERT INTO PACIENTI (ID_PACIENT, CNP, NUME, PRENUME, ASIGURARE_MEDICALA)
VALUES (5, 5678901234567, 'Constantinescu', 'Elena', 'Da');
```

```
INSERT INTO PACIENTI (ID_PACIENT, CNP, NUME, PRENUME, ASIGURARE_MEDICALA)
VALUES (6, 6789012345678, 'Radulescu', 'Andrei', 'Nu');
```

```
INSERT INTO PACIENTI (ID_PACIENT, CNP, NUME, PRENUME, ASIGURARE_MEDICALA)
VALUES (7, 7890123456789, 'Mihai', 'Ana-Maria', 'Da');
```

```
INSERT INTO PACIENTI (ID_PACIENT, CNP, NUME, PRENUME, ASIGURARE_MEDICALA)
VALUES (8, 8901234567890, 'Stoica', 'Gabriel', 'Nu');
```

```
INSERT INTO PACIENTI (ID_PACIENT, CNP, NUME, PRENUME, ASIGURARE_MEDICALA)
VALUES (9, 9012345678901, 'Popa', 'Cristina', 'Da');
```

```
INSERT INTO PACIENTI (ID_PACIENT, CNP, NUME, PRENUME, ASIGURARE_MEDICALA)
VALUES (10, 1234509876543, 'Marinescu', 'Alexandru', 'Nu');
```

```
INSERT INTO PACIENTI (ID_PACIENT, CNP, NUME, PRENUME, ASIGURARE_MEDICALA)
VALUES (11, 2345098765432, 'Florescu', 'Andreea', 'Da');
```

```
INSERT INTO PACIENTI (ID_PACIENT, CNP, NUME, PRENUME, ASIGURARE_MEDICALA)
VALUES (12, 3450987654321, 'Dobre', 'George', 'Nu');
```

```
INSERT INTO PACIENTI (ID_PACIENT, CNP, NUME, PRENUME, ASIGURARE_MEDICALA)
VALUES (13, 4509876543210, 'Neagu', 'Raluca', 'Da');
```

```
INSERT INTO PACIENTI (ID_PACIENT, CNP, NUME, PRENUME, ASIGURARE_MEDICALA)
VALUES (14, 5098765432109, 'Ionita', 'Robert', 'Nu');
```

```
INSERT INTO PACIENTI (ID_PACIENT, CNP, NUME, PRENUME, ASIGURARE_MEDICALA)
VALUES (15, 5987654321098, 'Gheorghe', 'Ioana', 'Da');
```

--Inserare INTERNARI

-- Inserare a 15 înregistrări în tabelul INTERNARI folosind datele din tabela PACIENTI
INSERT INTO INTERNARI (ID_PACIENT, ID_SECTIE_M, DATA_INTERNARE)

```
VALUES (1, 'SEC1', TO_DATE('2023-11-01', 'YYYY-MM-DD'));  
  
INSERT INTO INTERNARI (ID_PACIENT, ID_SECTIE_M, DATA_INTERNARE)  
VALUES (2, 'SEC3', TO_DATE('2023-10-15', 'YYYY-MM-DD'));  
  
INSERT INTO INTERNARI (ID_PACIENT, ID_SECTIE_M, DATA_INTERNARE)  
VALUES (3, 'SEC2', TO_DATE('2023-10-20', 'YYYY-MM-DD'));  
  
INSERT INTO INTERNARI (ID_PACIENT, ID_SECTIE_M, DATA_INTERNARE)  
VALUES (4, 'SEC5', TO_DATE('2023-11-10', 'YYYY-MM-DD'));  
  
INSERT INTO INTERNARI (ID_PACIENT, ID_SECTIE_M, DATA_INTERNARE)  
VALUES (5, 'SEC4', TO_DATE('2023-09-25', 'YYYY-MM-DD'));  
  
INSERT INTO INTERNARI (ID_PACIENT, ID_SECTIE_M, DATA_INTERNARE)  
VALUES (6, 'SEC1', TO_DATE('2023-08-18', 'YYYY-MM-DD'));  
  
INSERT INTO INTERNARI (ID_PACIENT, ID_SECTIE_M, DATA_INTERNARE)  
VALUES (7, 'SEC3', TO_DATE('2023-08-22', 'YYYY-MM-DD'));  
  
INSERT INTO INTERNARI (ID_PACIENT, ID_SECTIE_M, DATA_INTERNARE)  
VALUES (8, 'SEC2', TO_DATE('2023-09-05', 'YYYY-MM-DD'));  
  
INSERT INTO INTERNARI (ID_PACIENT, ID_SECTIE_M, DATA_INTERNARE)  
VALUES (9, 'SEC4', TO_DATE('2023-10-03', 'YYYY-MM-DD'));  
  
INSERT INTO INTERNARI (ID_PACIENT, ID_SECTIE_M, DATA_INTERNARE)  
VALUES (10, 'SEC5', TO_DATE('2023-11-22', 'YYYY-MM-DD'));  
  
INSERT INTO INTERNARI (ID_PACIENT, ID_SECTIE_M, DATA_INTERNARE)  
VALUES (11, 'SEC1', TO_DATE('2023-11-12', 'YYYY-MM-DD'));  
  
INSERT INTO INTERNARI (ID_PACIENT, ID_SECTIE_M, DATA_INTERNARE)  
VALUES (12, 'SEC3', TO_DATE('2023-09-30', 'YYYY-MM-DD'));  
  
INSERT INTO INTERNARI (ID_PACIENT, ID_SECTIE_M, DATA_INTERNARE)  
VALUES (13, 'SEC2', TO_DATE('2023-10-10', 'YYYY-MM-DD'));  
  
INSERT INTO INTERNARI (ID_PACIENT, ID_SECTIE_M, DATA_INTERNARE)  
VALUES (14, 'SEC4', TO_DATE('2023-08-29', 'YYYY-MM-DD'));
```

```
INSERT INTO INTERNARI (ID_PACIENT, ID_SECTIE_M, DATA_INTERNARE)
VALUES (15, 'SEC5', TO_DATE('2023-09-15', 'YYYY-MM-DD'));
```

-- Inserare tabela Istoric Medical

```
INSERT INTO ISTORIC_MEDICAL (ID_PACIENT, ID_ISTORIC, PATOLOGII,
CONTRAINdicatii)
VALUES (1, 101, 'Hipertensiune', 'Alergie la penicilină');
```

```
INSERT INTO ISTORIC_MEDICAL (ID_PACIENT, ID_ISTORIC, PATOLOGII,
CONTRAINdicatii)
VALUES (2, 102, 'Diabet', 'Intoleranță la antiinflamatoare');
```

```
INSERT INTO ISTORIC_MEDICAL (ID_PACIENT, ID_ISTORIC, PATOLOGII,
CONTRAINdicatii)
VALUES (3, 103, 'Astm', 'Alergie la aspirină');
```

```
INSERT INTO ISTORIC_MEDICAL (ID_PACIENT, ID_ISTORIC, PATOLOGII,
CONTRAINdicatii)
VALUES (4, 104, 'Insuficiență cardiacă', 'Alergie la sulfați');
```

```
INSERT INTO ISTORIC_MEDICAL (ID_PACIENT, ID_ISTORIC, PATOLOGII,
CONTRAINdicatii)
VALUES (5, 105, 'Osteoporoză', 'Intoleranță la anumite antibiotice');
```

```
INSERT INTO ISTORIC_MEDICAL (ID_PACIENT, ID_ISTORIC, PATOLOGII,
CONTRAINdicatii)
VALUES (6, 106, 'Ulcer', NULL);
```

```
INSERT INTO ISTORIC_MEDICAL (ID_PACIENT, ID_ISTORIC, PATOLOGII,
CONTRAINdicatii)
VALUES (7, 107, 'Ulcer', 'Intoleranță la lactoză');
```

```
INSERT INTO ISTORIC_MEDICAL (ID_PACIENT, ID_ISTORIC, PATOLOGII,
CONTRAINdicatii)
VALUES (8, 108, 'Anemie', NULL);
```

```
INSERT INTO ISTORIC_MEDICAL (ID_PACIENT, ID_ISTORIC, PATOLOGII,
CONTRAINdicatii)
VALUES (9, 109, 'Artrită', 'Intoleranță la gluten');
```

```
INSERT INTO ISTORIC_MEDICAL (ID_PACIENT, ID_ISTORIC, PATOLOGII,  
CONTRAINDICATII)
```

```
VALUES (10, 110, 'Anemie', 'Alergie la aspirină');
```

```
INSERT INTO ISTORIC_MEDICAL (ID_PACIENT, ID_ISTORIC, PATOLOGII,  
CONTRAINDICATII)
```

```
VALUES (11, 111, 'Colesterol crescut', NULL);
```

```
INSERT INTO ISTORIC_MEDICAL (ID_PACIENT, ID_ISTORIC, PATOLOGII,  
CONTRAINDICATII)
```

```
VALUES (12, 112, 'Probleme tiroidiene', NULL);
```

```
INSERT INTO ISTORIC_MEDICAL (ID_PACIENT, ID_ISTORIC, PATOLOGII,  
CONTRAINDICATII)
```

```
VALUES (13, 113, 'Fibroză chistică', 'Intoleranță la cafeină');
```

```
INSERT INTO ISTORIC_MEDICAL (ID_PACIENT, ID_ISTORIC, PATOLOGII,  
CONTRAINDICATII)
```

```
VALUES (14, 114, 'Colesterol crescut', 'Intoleranță la sulfiti');
```

```
INSERT INTO ISTORIC_MEDICAL (ID_PACIENT, ID_ISTORIC, PATOLOGII,  
CONTRAINDICATII)
```

```
VALUES (15, 115, 'Migrene', 'Intoleranță la sulfiti');
```

-- Inserari Medici

```
INSERT INTO MEDICI (ID_MEDIC, NUME, PRENUME, SALARIU, MEDIC_SEF,  
ID_SECTIE_M)
```

```
VALUES (1, 'Popescu', 'Andrei', 8000, 'Dr. Ionescu', 'SEC1');
```

```
INSERT INTO MEDICI (ID_MEDIC, NUME, PRENUME, SALARIU, MEDIC_SEF,  
ID_SECTIE_M)
```

```
VALUES (2, 'Ionescu', 'Maria', 7500, NULL, 'SEC2');
```

```
INSERT INTO MEDICI (ID_MEDIC, NUME, PRENUME, SALARIU, MEDIC_SEF,  
ID_SECTIE_M)
```

```
VALUES (3, 'Dumitrescu', 'Mihai', 8200, NULL, 'SEC3');
```

```
INSERT INTO MEDICI (ID_MEDIC, NUME, PRENUME, SALARIU, MEDIC_SEF,
ID_SECTIE_M)
VALUES (4, 'Georgescu', 'Elena', 7800, 'Dr. Popescu', 'SEC4');
```

```
INSERT INTO MEDICI (ID_MEDIC, NUME, PRENUME, SALARIU, MEDIC_SEF,
ID_SECTIE_M)
VALUES (5, 'Constantinescu', 'Vlad', 7900, NULL, 'SEC5');
```

```
INSERT INTO MEDICI (ID_MEDIC, NUME, PRENUME, SALARIU, MEDIC_SEF,
ID_SECTIE_M)
VALUES (6, 'Radulescu', 'Ana', 8200, 'Dr. Ionescu', 'SEC1');
```

```
INSERT INTO MEDICI (ID_MEDIC, NUME, PRENUME, SALARIU, MEDIC_SEF,
ID_SECTIE_M)
VALUES (7, 'Mihai', 'Alexandru', 7700, NULL, 'SEC2');
```

```
INSERT INTO MEDICI (ID_MEDIC, NUME, PRENUME, SALARIU, MEDIC_SEF,
ID_SECTIE_M)
VALUES (8, 'Stoica', 'Raluca', 8100, NULL, 'SEC3');
```

```
INSERT INTO MEDICI (ID_MEDIC, NUME, PRENUME, SALARIU, MEDIC_SEF,
ID_SECTIE_M)
VALUES (9, 'Popa', 'George', 7900, 'Dr. Georgescu', 'SEC4');
```

```
INSERT INTO MEDICI (ID_MEDIC, NUME, PRENUME, SALARIU, MEDIC_SEF,
ID_SECTIE_M)
VALUES (10, 'Marinescu', 'Ioana', 7800, NULL, 'SEC5');
```

-- istoric medici

```
INSERT INTO ISTORIC_SALARIU (ID_ISTORIC_SAL, ID_MEDIC, SALARIU_VECHI,
SALARIU_NOU, DATA_ACTUALIZARE)
VALUES (1, 1, 1000, (SELECT SALARIU FROM MEDICI WHERE ID_MEDIC = 1),
TO_DATE('2018-01-01', 'YYYY-MM-DD') + 1);
```

```
INSERT INTO ISTORIC_SALARIU (ID_ISTORIC_SAL, ID_MEDIC, SALARIU_VECHI,
SALARIU_NOU, DATA_ACTUALIZARE)
VALUES (2, 2, 1200, (SELECT SALARIU FROM MEDICI WHERE ID_MEDIC = 2),
TO_DATE('2018-01-01', 'YYYY-MM-DD') + 2);
```

```
INSERT INTO ISTORIC_SALARIU (ID_ISTORIC_SAL, ID_MEDIC, SALARIU_VECI,  
SALARIU_NOU, DATA_ACTUALIZARE)  
VALUES (3, 3, 1500, (SELECT SALARIU FROM MEDICI WHERE ID_MEDIC = 3),  
TO_DATE('2018-01-01', 'YYYY-MM-DD') + 3);
```

```
INSERT INTO ISTORIC_SALARIU (ID_ISTORIC_SAL, ID_MEDIC, SALARIU_VECI,  
SALARIU_NOU, DATA_ACTUALIZARE)  
VALUES (4, 4, 1000, (SELECT SALARIU FROM MEDICI WHERE ID_MEDIC = 4),  
TO_DATE('2018-01-01', 'YYYY-MM-DD') + 4);
```

```
INSERT INTO ISTORIC_SALARIU (ID_ISTORIC_SAL, ID_MEDIC, SALARIU_VECI,  
SALARIU_NOU, DATA_ACTUALIZARE)  
VALUES (5, 5, 2000, (SELECT SALARIU FROM MEDICI WHERE ID_MEDIC = 5),  
TO_DATE('2018-01-01', 'YYYY-MM-DD') + 5);
```

```
INSERT INTO ISTORIC_SALARIU (ID_ISTORIC_SAL, ID_MEDIC, SALARIU_VECI,  
SALARIU_NOU, DATA_ACTUALIZARE)  
VALUES (6, 6, 1000, (SELECT SALARIU FROM MEDICI WHERE ID_MEDIC = 6),  
TO_DATE('2018-01-01', 'YYYY-MM-DD') + 6);
```

```
INSERT INTO ISTORIC_SALARIU (ID_ISTORIC_SAL, ID_MEDIC, SALARIU_VECI,  
SALARIU_NOU, DATA_ACTUALIZARE)  
VALUES (7, 7, 1250, (SELECT SALARIU FROM MEDICI WHERE ID_MEDIC = 7),  
TO_DATE('2018-01-01', 'YYYY-MM-DD') + 7);
```

```
INSERT INTO ISTORIC_SALARIU (ID_ISTORIC_SAL, ID_MEDIC, SALARIU_VECI,  
SALARIU_NOU, DATA_ACTUALIZARE)  
VALUES (8, 8, 1500, (SELECT SALARIU FROM MEDICI WHERE ID_MEDIC = 8),  
TO_DATE('2018-01-01', 'YYYY-MM-DD') + 8);
```

```
INSERT INTO ISTORIC_SALARIU (ID_ISTORIC_SAL, ID_MEDIC, SALARIU_VECI,  
SALARIU_NOU, DATA_ACTUALIZARE)  
VALUES (9, 9, 1500, (SELECT SALARIU FROM MEDICI WHERE ID_MEDIC = 9),  
TO_DATE('2018-01-01', 'YYYY-MM-DD') + 9);
```

```
INSERT INTO ISTORIC_SALARIU (ID_ISTORIC_SAL, ID_MEDIC, SALARIU_VECI,  
SALARIU_NOU, DATA_ACTUALIZARE)  
VALUES (10, 10, 1500, (SELECT SALARIU FROM MEDICI WHERE ID_MEDIC = 10),  
TO_DATE('2018-01-01', 'YYYY-MM-DD') + 10);
```

--INSERIRE CONSULTATII

```
INSERT INTO CONSULTATII (data_cons, ID_MEDIC, ID_PACIENT)
VALUES (TO_DATE('2023-11-18', 'YYYY-MM-DD'), 1, 1);
```

```
INSERT INTO CONSULTATII (data_cons, ID_MEDIC, ID_PACIENT)
VALUES (TO_DATE('2023-11-18', 'YYYY-MM-DD'), 3, 2);
```

```
INSERT INTO CONSULTATII (data_cons, ID_MEDIC, ID_PACIENT)
VALUES (TO_DATE('2023-12-18', 'YYYY-MM-DD'), 1, 1);
```

```
INSERT INTO CONSULTATII (data_cons, ID_MEDIC, ID_PACIENT)
VALUES (TO_DATE('2023-12-18', 'YYYY-MM-DD'), 3, 2);
```

```
INSERT INTO CONSULTATII (data_cons, ID_MEDIC, ID_PACIENT)
VALUES (TO_DATE('2023-12-18', 'YYYY-MM-DD'), 2, 3);
```

```
INSERT INTO CONSULTATII (data_cons, ID_MEDIC, ID_PACIENT)
VALUES (TO_DATE('2023-12-18', 'YYYY-MM-DD'), 5, 4);
```

```
INSERT INTO CONSULTATII (data_cons, ID_MEDIC, ID_PACIENT)
VALUES (TO_DATE('2023-12-18', 'YYYY-MM-DD'), 4, 5);
```

```
INSERT INTO CONSULTATII (data_cons, ID_MEDIC, ID_PACIENT)
VALUES (TO_DATE('2023-12-18', 'YYYY-MM-DD'), 6, 6);
```

```
INSERT INTO CONSULTATII (data_cons, ID_MEDIC, ID_PACIENT)
VALUES (TO_DATE('2023-12-18', 'YYYY-MM-DD'), 8, 7);
```

```
INSERT INTO CONSULTATII (data_cons, ID_MEDIC, ID_PACIENT)
VALUES (TO_DATE('2023-12-18', 'YYYY-MM-DD'), 7, 8);
```

```
INSERT INTO CONSULTATII (data_cons, ID_MEDIC, ID_PACIENT)
VALUES (TO_DATE('2023-12-18', 'YYYY-MM-DD'), 9, 9);
```

```
INSERT INTO CONSULTATII (data_cons, ID_MEDIC, ID_PACIENT)
VALUES (TO_DATE('2023-12-18', 'YYYY-MM-DD'), 5, 10);
```

```
INSERT INTO CONSULTATII (data_cons, ID_MEDIC, ID_PACIENT)
VALUES (TO_DATE('2023-12-19', 'YYYY-MM-DD'), 5, 10);
```

```
select * from consultatii;
```

--Inserare Categorii

```
INSERT INTO CATEGORIE (ID_CATEGORIE, ECHIPAMENT)
VALUES ('Medicamentos', 'Medicamente');
```

```
INSERT INTO CATEGORIE (ID_CATEGORIE, ECHIPAMENT)
VALUES ('Chirurgicale', 'Instrumente chirurgicale, materiale sterile, echipamente de monitorizare post-operatorie');
```

```
INSERT INTO CATEGORIE (ID_CATEGORIE, ECHIPAMENT)
VALUES ('Ortopedice', 'Dispozitive ortopedice, proteze, aparate ortopedice specifice');
```

```
INSERT INTO CATEGORIE (ID_CATEGORIE, ECHIPAMENT)
VALUES ('FIZIOTERAPIE', 'Mese de masaj, role de masaj, aparate de electroterapie, aparate cu ultrasunet');
```

```
INSERT INTO CATEGORIE (ID_CATEGORIE, ECHIPAMENT)
VALUES ('PSIHOTERAPIE', 'Scaune și mobilier de consiliere');
```

-- Inserare Tratamente

```
INSERT INTO TRATAMENTE (ID_TRATAMENT, data_cons, ID_CATEGORIE, DURATA,
DESCRIERE, RECOMANDARI, ID_MEDIC, ID_PACIENT)
VALUES (1, TO_DATE('2023-11-18', 'YYYY-MM-DD'), 'Chirurgicale', '7 zile', 'Intervenție chirurgicală', 'Repaus post-operator', 1, 1);
```

```
INSERT INTO TRATAMENTE (ID_TRATAMENT, data_cons, ID_CATEGORIE, DURATA,
DESCRIERE, RECOMANDARI, ID_MEDIC, ID_PACIENT)
VALUES (2, TO_DATE('2023-11-18', 'YYYY-MM-DD'), 'Medicamentos', '20 zile', 'Tratament antibiotice', 'Repaus și hidratare', 3, 2);
```

```
INSERT INTO TRATAMENTE (ID_TRATAMENT, data_cons, ID_CATEGORIE, DURATA,
DESCRIERE, RECOMANDARI, ID_MEDIC, ID_PACIENT)
VALUES (3, TO_DATE('2023-12-18', 'YYYY-MM-DD'), 'Medicamentos', '30 zile', 'Tratament curativ pentru dilatarea vaselor sanguine', 'Repaus și hidratare', 2, 3);
```

```
INSERT INTO TRATAMENTE (ID_TRATAMENT, data_cons, ID_CATEGORIE, DURATA,
DESCRIERE, RECOMANDARI, ID_MEDIC, ID_PACIENT)
VALUES (4, TO_DATE('2023-12-18', 'YYYY-MM-DD'), 'FIZIOTERAPIE', '2 luni', 'Sesiuni de fizioterapie', 'Exerciții specifice', 4, 5);
```

```
INSERT INTO TRATAMENTE (ID_TRATAMENT, data_cons, ID_CATEGORIE, DURATA,  
DESCRIERE, RECOMANDARI, ID_MEDIC, ID_PACIENT)  
VALUES (5, TO_DATE('2023-12-18', 'YYYY-MM-DD'), 'PSIHOTERAPIE', '3 luni', 'Sesiuni de  
consiliere psihologica', 'Repaus și îngrijire', 5, 4);
```

```
INSERT INTO TRATAMENTE (ID_TRATAMENT, data_cons, ID_CATEGORIE, DURATA,  
DESCRIERE, RECOMANDARI, ID_MEDIC, ID_PACIENT)  
VALUES (6, TO_DATE('2023-12-18', 'YYYY-MM-DD'), 'Chirurgicale', '21 zile', 'Intervenție  
chirurgicală', 'Repaus post-operator', 6, 6);
```

```
INSERT INTO TRATAMENTE (ID_TRATAMENT, data_cons, ID_CATEGORIE, DURATA,  
DESCRIERE, RECOMANDARI, ID_MEDIC, ID_PACIENT)  
VALUES (7, TO_DATE('2023-12-18', 'YYYY-MM-DD'), 'Ortopedice', '3 luni', 'Masaj',  
'Urmărire periodică', 8, 7);
```

```
INSERT INTO TRATAMENTE (ID_TRATAMENT, data_cons, ID_CATEGORIE, DURATA,  
DESCRIERE, RECOMANDARI, ID_MEDIC, ID_PACIENT)  
VALUES (8, TO_DATE('2023-12-18', 'YYYY-MM-DD'), 'Medicamentos', '14 zile', 'Tratament  
curativ cardiovascular', 'Repaus și monitorizare', 7, 8);
```

```
INSERT INTO TRATAMENTE (ID_TRATAMENT, data_cons, ID_CATEGORIE, DURATA,  
DESCRIERE, RECOMANDARI, ID_MEDIC, ID_PACIENT)  
VALUES (9, TO_DATE('2023-12-18', 'YYYY-MM-DD'), 'Ortopedice', '30 zile', 'Masaj si  
medicamente', 'Repaus și monitorizare', 9, 9);
```

```
INSERT INTO TRATAMENTE (ID_TRATAMENT, data_cons, ID_CATEGORIE, DURATA,  
DESCRIERE, RECOMANDARI, ID_MEDIC, ID_PACIENT)  
VALUES (10, TO_DATE('2023-12-18', 'YYYY-MM-DD'), 'PSIHOTERAPIE', '3 luni', 'Sesiuni  
complexe de PSIHOTERAPIE', null, 5, 10);
```

```
INSERT INTO TRATAMENTE (ID_TRATAMENT, data_cons, ID_CATEGORIE, DURATA,  
DESCRIERE, RECOMANDARI, ID_MEDIC, ID_PACIENT)  
VALUES (11, TO_DATE('2023-12-18', 'YYYY-MM-DD'), 'PSIHOTERAPIE', '1 luna',  
'Tratament cu antidepresive', null, 5, 10);
```

```
INSERT INTO TRATAMENTE (ID_TRATAMENT, data_cons, ID_CATEGORIE, DURATA,  
DESCRIERE, RECOMANDARI, ID_MEDIC, ID_PACIENT)  
VALUES (12, TO_DATE('2023-12-18', 'YYYY-MM-DD'), 'Medicamentos', '20 zile',  
'Recuperare', 'Repaus și hidratare', 3, 2);
```

```
INSERT INTO TRATAMENTE (ID_TRATAMENT, data_cons, ID_CATEGORIE, DURATA,  
DESCRIERE, RECOMANDARI, ID_MEDIC, ID_PACIENT)  
VALUES (13, TO_DATE('2023-12-19', 'YYYY-MM-DD'), 'Medicamentos', '20 zile',  
'Recuperare', 'Repaus și hidratare', 5, 10);
```

```
select * from tratamente;  
--Inserare Medicamente  
INSERT INTO MEDICAMENTE (ID_MEDICAMENT, NUME, PRET)  
VALUES ('MED1', 'Paracetamol', 10.50);
```

```
INSERT INTO MEDICAMENTE (ID_MEDICAMENT, NUME, PRET)  
VALUES ('MED2', 'Ibuprofen', 15.75);
```

```
INSERT INTO MEDICAMENTE (ID_MEDICAMENT, NUME, PRET)  
VALUES ('MED3', 'Amoxicilină', 20.20);
```

```
INSERT INTO MEDICAMENTE (ID_MEDICAMENT, NUME, PRET)  
VALUES ('MED4', 'Aspirină', 8.30);
```

```
INSERT INTO MEDICAMENTE (ID_MEDICAMENT, NUME, PRET)  
VALUES ('MED5', 'Claritromycină', 30.00);
```

```
INSERT INTO MEDICAMENTE (ID_MEDICAMENT, NUME, PRET)  
VALUES ('MED6', 'Omeprazol', 12.60);
```

```
INSERT INTO MEDICAMENTE (ID_MEDICAMENT, NUME, PRET)  
VALUES ('MED7', 'Diclofenac', 18.90);
```

```
INSERT INTO MEDICAMENTE (ID_MEDICAMENT, NUME, PRET)  
VALUES ('MED8', 'Ranitidină', 22.40);
```

```
INSERT INTO MEDICAMENTE (ID_MEDICAMENT, NUME, PRET)  
VALUES ('MED9', 'Furosemid', 25.80);
```

```
INSERT INTO MEDICAMENTE (ID_MEDICAMENT, NUME, PRET)  
VALUES ('MED10', 'Atorvastatin', 28.00);
```

```
INSERT INTO MEDICAMENTE (ID_MEDICAMENT, NUME, PRET)
```

```
VALUES ('MED11', 'Metformin', 17.30);
```

```
INSERT INTO MEDICAMENTE (ID_MEDICAMENT, NUME, PRET)  
VALUES ('MED12', 'Losartan', 21.50);
```

```
INSERT INTO MEDICAMENTE (ID_MEDICAMENT, NUME, PRET)  
VALUES ('MED13', 'Amlodipină', 19.70);
```

```
INSERT INTO MEDICAMENTE (ID_MEDICAMENT, NUME, PRET)  
VALUES ('MED14', 'Levotiroxină', 16.90);
```

```
INSERT INTO MEDICAMENTE (ID_MEDICAMENT, NUME, PRET)  
VALUES ('MED15', 'Tramadol', 24.90);
```

```
-- inserare tabela TRATAMENTE_CONTIN_MEDICAMENTE
```

```
INSERT INTO TRATAMENTE_CONTIN_MEDICAMENTE (ID_TRATAMENT,  
ID_MEDICAMENT, DOZA, INDICATII)  
VALUES (1, 'MED1', '500mg', 'O dată pe zi, după masă');
```

```
INSERT INTO TRATAMENTE_CONTIN_MEDICAMENTE (ID_TRATAMENT,  
ID_MEDICAMENT, DOZA, INDICATII)  
VALUES (1, 'MED3', '750mg', 'De două ori pe zi');
```

```
INSERT INTO TRATAMENTE_CONTIN_MEDICAMENTE (ID_TRATAMENT,  
ID_MEDICAMENT, DOZA, INDICATII)  
VALUES (2, 'MED5', '250mg', 'Dimineața și seara');
```

```
INSERT INTO TRATAMENTE_CONTIN_MEDICAMENTE (ID_TRATAMENT,  
ID_MEDICAMENT, DOZA, INDICATII)  
VALUES (2, 'MED7', '100mg', 'La nevoie, pentru durere');
```

```
INSERT INTO TRATAMENTE_CONTIN_MEDICAMENTE (ID_TRATAMENT,  
ID_MEDICAMENT, DOZA, INDICATII)  
VALUES (3, 'MED9', '40mg', 'O dată pe zi, dimineață');
```

```
INSERT INTO TRATAMENTE_CONTIN_MEDICAMENTE (ID_TRATAMENT,  
ID_MEDICAMENT, DOZA, INDICATII)  
VALUES (3, 'MED11', '500mg', 'După fiecare masă');
```

```
INSERT INTO TRATAMENTE_CONTIN_MEDICAMENTE (ID_TRATAMENT,
```

ID_MEDICAMENT, DOZA, INDICATII)

VALUES (4, 'MED13', '10mg', 'O dată pe zi, seara');

INSERT INTO TRATAMENTE_CONTIN_MEDICAMENTE (ID_TRATAMENT,
ID_MEDICAMENT, DOZA, INDICATII)

VALUES (4, 'MED15', '50mg', 'La nevoie, pentru durere moderată');

INSERT INTO TRATAMENTE_CONTIN_MEDICAMENTE (ID_TRATAMENT,
ID_MEDICAMENT, DOZA, INDICATII)

VALUES (5, 'MED2', '400mg', 'De trei ori pe zi, după mese');

INSERT INTO TRATAMENTE_CONTIN_MEDICAMENTE (ID_TRATAMENT,
ID_MEDICAMENT, DOZA, INDICATII)

VALUES (5, 'MED4', '1000mg', 'Dimineața și seara');

INSERT INTO TRATAMENTE_CONTIN_MEDICAMENTE (ID_TRATAMENT,
ID_MEDICAMENT, DOZA, INDICATII)

VALUES (8, 'MED10', '20mg', 'O dată pe zi, seara');

INSERT INTO TRATAMENTE_CONTIN_MEDICAMENTE (ID_TRATAMENT,
ID_MEDICAMENT, DOZA, INDICATII)

VALUES (8, 'MED15', '20mg', 'O dată pe zi, seara');

INSERT INTO TRATAMENTE_CONTIN_MEDICAMENTE (ID_TRATAMENT,
ID_MEDICAMENT, DOZA, INDICATII)

VALUES (8, 'MED12', '50mg', 'Dimineața și seara');

INSERT INTO TRATAMENTE_CONTIN_MEDICAMENTE (ID_TRATAMENT,
ID_MEDICAMENT, DOZA, INDICATII)

VALUES (3, 'MED6', '10mg', 'După fiecare masă');

INSERT INTO TRATAMENTE_CONTIN_MEDICAMENTE (ID_TRATAMENT,
ID_MEDICAMENT, DOZA, INDICATII)

VALUES (3, 'MED10', '30mg', 'După fiecare masă');

-- Inserare în tabela SECTOARE

INSERT INTO SECTOARE (ID_SECTOR, NUME)

VALUES ('SECTOR1', 'Sector 1');

INSERT INTO SECTOARE (ID_SECTOR, NUME)

```
VALUES ('SECTOR2', 'Sector 2');
```

```
INSERT INTO SECTOARE (ID_SECTOR, NUME)  
VALUES ('SECTOR3', 'Sector 3');
```

```
INSERT INTO SECTOARE (ID_SECTOR, NUME)  
VALUES ('SECTOR4', 'Sector 4');
```

```
INSERT INTO SECTOARE (ID_SECTOR, NUME)  
VALUES ('SECTOR5', 'Sector 5');
```

-- Inserare în tabela ZONE

```
INSERT INTO ZONE (ID_ZONA, ID_SECTOR)  
VALUES ('ZONA1', 'SECTOR1');
```

```
INSERT INTO ZONE (ID_ZONA, ID_SECTOR)  
VALUES ('ZONA2-1', 'SECTOR1');
```

```
INSERT INTO ZONE (ID_ZONA, ID_SECTOR)  
VALUES ('ZONA2', 'SECTOR2');
```

```
INSERT INTO ZONE (ID_ZONA, ID_SECTOR)  
VALUES ('ZONA3', 'SECTOR3');
```

```
INSERT INTO ZONE (ID_ZONA, ID_SECTOR)  
VALUES ('ZONA4', 'SECTOR4');
```

```
INSERT INTO ZONE (ID_ZONA, ID_SECTOR)  
VALUES ('ZONA5', 'SECTOR5');
```

-- Inserare în tabela SPITAL

```
INSERT INTO SPITAL (ID_SPITAL, NUME_SPITAL, COD_POSTAL, ID_ZONA)  
VALUES ('SPITAL1', 'Spitalul Clinic de Urgență Floreasca', '014461', 'ZONA1');
```

```
INSERT INTO SPITAL (ID_SPITAL, NUME_SPITAL, COD_POSTAL, ID_ZONA)  
VALUES ('SPITAL6', 'Spitalul Clinic de Urgență Floreasca 2', '014461', 'ZONA2-1');
```

```
INSERT INTO SPITAL (ID_SPITAL, NUME_SPITAL, COD_POSTAL, ID_ZONA)  
VALUES ('SPITAL2', 'Spitalul Clinic de Urgență Bucur', '021012', 'ZONA2');
```

```
INSERT INTO SPITAL (ID_SPITAL, NUME_SPITAL, COD_POSTAL, ID_ZONA)
VALUES ('SPITAL3', 'Spitalul Universitar de Urgență Elias', '030475', 'ZONA3');
```

```
INSERT INTO SPITAL (ID_SPITAL, NUME_SPITAL, COD_POSTAL, ID_ZONA)
VALUES ('SPITAL4', 'Spitalul Militar Central', '041441', 'ZONA4');
```

```
INSERT INTO SPITAL (ID_SPITAL, NUME_SPITAL, COD_POSTAL, ID_ZONA)
VALUES ('SPITAL5', 'Spitalul Clinic de Urgență Sfântul Ioan', '050098', 'ZONA5');
```

--Inserarea MEDICI_LUCREAZA_IN_SPITALE

```
INSERT INTO MEDIC_LUCREAZA_IN_SPITAL (ID_MEDIC, ID_SPITAL,
DATA_ANGAJARII)
VALUES (1, 'SPITAL1', TO_DATE('2023-01-15', 'YYYY-MM-DD'));
```

```
INSERT INTO MEDIC_LUCREAZA_IN_SPITAL (ID_MEDIC, ID_SPITAL,
DATA_ANGAJARII)
VALUES (2, 'SPITAL2', TO_DATE('2022-11-20', 'YYYY-MM-DD'));
```

```
INSERT INTO MEDIC_LUCREAZA_IN_SPITAL (ID_MEDIC, ID_SPITAL,
DATA_ANGAJARII)
VALUES (3, 'SPITAL3', TO_DATE('2023-03-10', 'YYYY-MM-DD'));
```

```
INSERT INTO MEDIC_LUCREAZA_IN_SPITAL (ID_MEDIC, ID_SPITAL,
DATA_ANGAJARII)
VALUES (4, 'SPITAL4', TO_DATE('2022-09-05', 'YYYY-MM-DD'));
```

```
INSERT INTO MEDIC_LUCREAZA_IN_SPITAL (ID_MEDIC, ID_SPITAL,
DATA_ANGAJARII)
VALUES (5, 'SPITAL5', TO_DATE('2022-08-18', 'YYYY-MM-DD'));
```

```
INSERT INTO MEDIC_LUCREAZA_IN_SPITAL (ID_MEDIC, ID_SPITAL,
DATA_ANGAJARII)
VALUES (6, 'SPITAL1', TO_DATE('2021-02-28', 'YYYY-MM-DD'));
```

```
INSERT INTO MEDIC_LUCREAZA_IN_SPITAL (ID_MEDIC, ID_SPITAL,
DATA_ANGAJARII)
VALUES (7, 'SPITAL2', TO_DATE('2021-01-03', 'YYYY-MM-DD'));
```

```
INSERT INTO MEDIC_LUCREAZA_IN_SPITAL (ID_MEDIC, ID_SPITAL,
```

```
DATA_ANGAJARII)
```

```
VALUES (7, 'SPITAL1', TO_DATE('2021-01-06', 'YYYY-MM-DD'));
```

```
INSERT INTO MEDIC_LUCREAZA_IN_SPITAL (ID_MEDIC, ID_SPITAL,  
DATA_ANGAJARII)
```

```
VALUES (8, 'SPITAL3', TO_DATE('2021-04-12', 'YYYY-MM-DD'));
```

```
INSERT INTO MEDIC_LUCREAZA_IN_SPITAL (ID_MEDIC, ID_SPITAL,  
DATA_ANGAJARII)
```

```
VALUES (8, 'SPITAL4', TO_DATE('2021-06-12', 'YYYY-MM-DD'));
```

```
INSERT INTO MEDIC_LUCREAZA_IN_SPITAL (ID_MEDIC, ID_SPITAL,  
DATA_ANGAJARII)
```

```
VALUES (9, 'SPITAL4', TO_DATE('2021-06-20', 'YYYY-MM-DD'));
```

```
INSERT INTO MEDIC_LUCREAZA_IN_SPITAL (ID_MEDIC, ID_SPITAL,  
DATA_ANGAJARII)
```

```
VALUES (10, 'SPITAL5', TO_DATE('2023-05-07', 'YYYY-MM-DD'));
```

```
INSERT INTO MEDIC_LUCREAZA_IN_SPITAL (ID_MEDIC, ID_SPITAL,  
DATA_ANGAJARII)
```

```
VALUES (10, 'SPITAL4', TO_DATE('2022-05-07', 'YYYY-MM-DD'));
```

```
COMMIT;
```

The screenshot shows a SQL Server Management Studio interface. At the top, there are three tabs: "...ql", "242_Macovei_Catalina_tema3.sql", "242_Macovei_Catalina_inserari.sql" (which is the active tab), and "DropDatabaseMediCare.sql". Below the tabs is a toolbar with various icons. The main area is divided into two panes: "Worksheet" and "Query Builder". The "Worksheet" pane contains the following SQL code:

```
INSERT INTO TRATAMENTE (ID_TRATAMENT, ID_CONSULTATIE, ID_CATEGORIE, DURATA, DESCRIERE, RECOMANDARI, ID_MEDIC, ID_PACIENT)
VALUES (7, 7, 'Ortopedice', '3 luni', 'Masaj', 'Urmărire periodică', 8, 7);

INSERT INTO TRATAMENTE (ID_TRATAMENT, ID_CONSULTATIE, ID_CATEGORIE, DURATA, DESCRIERE, RECOMANDARI, ID_MEDIC, ID_PACIENT)
VALUES (8, 8, 'Medicamentos', '14 zile', 'Tratament curativ cardiovascular', 'Repaus și monitorizare', 7, 8);

INSERT INTO TRATAMENTE (ID_TRATAMENT, ID_CONSULTATIE, ID_CATEGORIE, DURATA, DESCRIERE, RECOMANDARI, ID_MEDIC, ID_PACIENT)
VALUES (9, 9, 'Ortopedice', '30 zile', 'Masaj și medicamente', 'Repaus și monitorizare', 9, 9);

INSERT INTO TRATAMENTE (ID_TRATAMENT, ID_CONSULTATIE, ID_CATEGORIE, DURATA, DESCRIERE, RECOMANDARI, ID_MEDIC, ID_PACIENT)
VALUES (10, 10, 'PSIHOTERAPIE', '3 luni', 'Sesiuni complexe de PSIHOTERAPIE', null, 5, 10);

INSERT INTO TRATAMENTE (ID_TRATAMENT, ID_CONSULTATIE, ID_CATEGORIE, DURATA, DESCRIERE, RECOMANDARI, ID_MEDIC, ID_PACIENT)
VALUES (11, 10, 'PSIHOTERAPIE', '1 luna', 'Tratament cu antidepresive', null, 5, 10);

--Inserare Medicamente
INSERT INTO MEDICAMENTE (ID_MEDICAMENT, NUME, PRET)
VALUES ('MED1', 'Paracetamol', 10.50);

INSERT INTO MEDICAMENTE (ID_MEDICAMENT, NUME, PRET)
VALUES ('MED2', 'Ibuprofen', 15.75);

INSERT INTO MEDICAMENTE (ID_MEDICAMENT, NUME, PRET)
```

Below the Worksheet pane is a "Script Output" window with the following log:

```
1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.
```

The screenshot shows the SQL Server Management Studio interface. At the top, there are three tabs: ...sql, 242_Macovei_Catalina_tema3.sql, 242_Macovei_Catalina_inserari.sql, and DropDatabaseMediCare.sql. Below the tabs is a toolbar with various icons. The main area is divided into two panes: 'Worksheet' and 'Query Builder'. The 'Worksheet' pane contains a block of SQL code for inserting data into SECTOARE and ZONE tables. The 'Script Output' pane at the bottom shows the results of the execution, indicating 1 row inserted for each of the five insert statements.

```
VALUES ('SECTOR2', 'Sector 2');

INSERT INTO SECTOARE (ID_SECTOR, NUME)
VALUES ('SECTOR3', 'Sector 3');

INSERT INTO SECTOARE (ID_SECTOR, NUME)
VALUES ('SECTOR4', 'Sector 4');

INSERT INTO SECTOARE (ID_SECTOR, NUME)
VALUES ('SECTOR5', 'Sector 5');

-- Inserare in tabela ZONE
INSERT INTO ZONE (ID_ZONA, ID_SECTOR)
VALUES ('ZONA1', 'SECTOR1');

INSERT INTO ZONE (ID_ZONA, ID_SECTOR)
VALUES ('ZONA2-1', 'SECTOR2');

INSERT INTO ZONE (ID_ZONA, ID_SECTOR)
VALUES ('ZONA2', 'SECTOR2');

INSERT INTO ZONE (ID_ZONA, ID_SECTOR)
VALUES ('ZONA3', 'SECTOR3');
```

Script Output

```
1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.
```

...ql 242_Macovei_Catalina_tema3.sql x 242_Macovei_Catalina_inserari.sql x DropDatabaseMediCare.sql x

SQL Worksheet History

Worksheet Query Builder

```
VALUES ('SPITAL2', 'Spitalul Clinic de Urgență Bucur', '021012', 'ZONA2');

INSERT INTO SPITAL (ID_SPITAL, NUME_SPITAL, COD_POSTAL, ID_ZONA)
VALUES ('SPITAL3', 'Spitalul Universitar de Urgență Elias', '030475', 'ZONA3');

INSERT INTO SPITAL (ID_SPITAL, NUME_SPITAL, COD_POSTAL, ID_ZONA)
VALUES ('SPITAL4', 'Spitalul Militar Central', '041441', 'ZONA4');

INSERT INTO SPITAL (ID_SPITAL, NUME_SPITAL, COD_POSTAL, ID_ZONA)
VALUES ('SPITAL5', 'Spitalul Clinic de Urgență Sfântul Ioan', '050098', 'ZONA5');

--Inserarea MEDICI_LUCREAZA_IN_SPITALE

INSERT INTO MEDIC_LUCREAZA_IN_SPITAL (ID_MEDIC, ID_SPITAL, DATA_ANGAJARII)
VALUES (1, 'SPITAL1', TO_DATE('2023-01-15', 'YYYY-MM-DD'));

INSERT INTO MEDIC_LUCREAZA_IN_SPITAL (ID_MEDIC, ID_SPITAL, DATA_ANGAJARII)
VALUES (2, 'SPITAL2', TO_DATE('2022-11-20', 'YYYY-MM-DD'));

INSERT INTO MEDIC_LUCREAZA_IN_SPITAL (ID_MEDIC, ID_SPITAL, DATA_ANGAJARII)
VALUES (3, 'SPITAL3', TO_DATE('2023-03-10', 'YYYY-MM-DD'));

INSERT INTO MEDIC_LUCREAZA_IN_SPITAL (ID_MEDIC, ID_SPITAL, DATA_ANGAJARII)
VALUES (4, 'SPITAL4', TO_DATE('2022-09-05', 'YYYY-MM-DD'));
```

Script Output x

Task completed in 1.052 seconds

```
1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.
```

Subprogram stocat independent - Colecții

6. Crearea unui subprogram stocat independent care să utilizeze toate cele 3 tipuri de colecții studiate.

Această secțiune conține:

- Problema în limbaj natural
- Cod sursă
- Print Screen

Problema: Creați un subprogram stocat independent în PL/SQL pentru a identifica primii trei medici cu cel mai mic număr de consultații înregistrate. După identificarea acestora, salvați în colecții diferențe pacienții și consultațiile asociate. Afisați, din cele trei colecții de date: medicii, data consultării și numele pacienților.

```

File Edit View Navigate Run Source Team Tools Window Help
Fri 12 Jan 10:23
Oracle SQL Developer : /Users/catalinamacovei/sqldeveloper/exercitii_medicare.sql
242_Macovei_Catalina_inserari.sql * exercitii_medicare.sql * toDo.sql * DropIndexMediCare.sql * DOCKER_PLSQL *
SQL Worksheet History
DOCKER_PLSQL * Buffer Size: 20000
Worksheet Query Builder
Dbms Output *
+ - Buffer Size: 20000
DOCKER_PLSQL *
Id medic: 8 Stoica Raluca
Consultatii:
Data consultatie: 18-DEC-23
Pacient:
Mihai Ana-Maria
Id medic: 9 Popa George
Consultatii:
Data consultatie: 18-DEC-23
Pacient:
Popa Cristina
Id medic: 4 Georgescu Elena
Consultatii:
Data consultatie: 18-DEC-23
Pacient:
Constantinescu Elena
-- Afisare medici
FOR k IN 1..med.COUNT LOOP
    DBMS_OUTPUT.PUT_LINE('Id medic: ' || med(k).id_medic || ' ' || med(k).nume || ' ' || med(k).prenume);
    DBMS_OUTPUT.PUT_LINE('Consultatii:');
    -- Afisare consultatii pentru fiecare medic
    FOR j IN 1..cons_med.COUNT LOOP
        IF cons_med(j).id_medic = med(k).id_medic THEN
            DBMS_OUTPUT.PUT_LINE('Data consultatie: ' || cons_med(j).data_cons || ' ');
        END IF;
    END LOOP;
    -- Afisare pacienti pentru fiecare consultatie
    DBMS_OUTPUT.PUT_LINE('Pacient:');
    FOR i IN 1..pacienti_med.COUNT LOOP
        IF cons_med(j).id_pacient = pacienti_med(i).id_pacient THEN
            DBMS_OUTPUT.PUT_LINE(pacienti_med(i).nume || ' ' || pacienti_med(i).prenume);
        END IF;
    END LOOP;
    DBMS_OUTPUT.NEW_LINE;
END LOOP;
END;
/
BEGIN
    evaluare_medici;
END;

```

Script Output x
Task completed in 0.325 seconds
Procedure EVALUARE_MEDICI compiled
PL/SQL procedure successfully completed.

```

CREATE OR REPLACE PROCEDURE evaluare_medici IS
TYPE tablou_idx IS TABLE OF medici%ROWTYPE INDEX BY BINARY_INTEGER;
TYPE tablou_imbr IS TABLE OF pacienti%ROWTYPE;
TYPE vector IS VARRAY(3) OF consultatii%ROWTYPE;

med tablou_idx;
pacienti_med tablou_imbr := tablou_imbr();
cons_med vector := vector();
BEGIN
-- Salvare in tablou
SELECT ID_MEDIC, NUME, PRENUME, SALARIU, MEDIC_SEF, ID_SECTIE_M
BULK COLLECT INTO med
FROM (
  SELECT COUNT(*) AS NR_CONS, M.ID_MEDIC, M.NUME, M.PRENUME, M.SALARIU,
M.MEDIC_SEF, M.ID_SECTIE_M FROM MEDICI M
  INNER JOIN CONSULTATII C ON C.ID_MEDIC = M.ID_MEDIC
  GROUP BY M.ID_MEDIC, M.ID_MEDIC, M.NUME, M.PRENUME, M.SALARIU, M.MEDIC_SEF,
M.ID_SECTIE_M
  ORDER BY NR_CONS ASC) T
WHERE ROWNUM <= 3;

FOR i IN 1..med.COUNT LOOP
-- Pentru pacienti
FOR patient_rec IN (
  SELECT pa.*
  FROM pacienti pa
  INNER JOIN consultatii c ON c.id_pacient = pa.id_pacient
  WHERE c.id_medic = med(i).id_medic
) LOOP
  pacienti_med.extend;
  pacienti_med(pacienti_med.last) := patient_rec;
END LOOP;

-- Pentru consultatii
FOR cons_rec IN (
  SELECT *
  FROM consultatii
  WHERE id_medic = med(i).id_medic
) LOOP
  cons_med.extend;
  cons_med(cons_med.last) := cons_rec;
END LOOP;
END LOOP;

```

```

-- Afisare medici
FOR k IN 1..med.COUNT LOOP
    DBMS_OUTPUT.PUT_LINE('Id medic: ' || med(k).id_medic || ' ' || med(k).nume || ' ' ||
    med(k).prenume);
    DBMS_OUTPUT.PUT_LINE('Consultatii:');

    -- Afisare consultatii pentru fiecare medic
    FOR j IN 1..cons_med.COUNT LOOP
        IF cons_med(j).id_medic = med(k).id_medic THEN
            DBMS_OUTPUT.PUT_LINE('Data consultatie: ' || cons_med(j).data_cons || ' ');

            -- Afisare pacienti pentru fiecare consultatie
            DBMS_OUTPUT.PUT_LINE('Pacient: ');
            FOR i IN 1..pacienti_med.COUNT LOOP
                IF cons_med(j).id_pacient = pacienti_med(i).id_pacient THEN
                    DBMS_OUTPUT.PUT_LINE(pacienti_med(i).nume || ' ' || pacienti_med(i).prenume);
                END IF;
            END LOOP;

            END IF;
        END LOOP;
        DBMS_OUTPUT.NEW_LINE;
    END LOOP;
END;
/

BEGIN
    evaluare_medici;
END;

```

Subprogram stocat independent - 2 tipuri diferite de cursoare

7. Creati un subprogram stocat independent care sa utilizeze 2 tipuri diferite de cursoare studiate, unul dintre acestea fiind cursor parametrizat, dependent de celalalt cursor. Apelați subprogramul.

Această secțiune conține:

- Problema limbaj natural

- Cod sursă
- Print Screen

Problema: Creați un subprogram stocat independent în PL/SQL pentru a identifica pacienții care consumă medicamente mai scumpe decât media prețului medicamentelor din farmacia locală a spitalului. Din lista acestor pacienți, afișați doar cei care au avut cel puțin o internare.

The screenshot shows the Oracle SQL Developer interface. The top menu bar includes File, Edit, View, Navigate, Run, Source, Team, Tools, Window, and Help. The title bar indicates the session is connected to 'Oracle SQL Developer : /Users/catalinamacovei/.sqldeveloper/exercitii_medicare.sql'. The main workspace contains several tabs: '242_Macovei_Catalina_inserari.sql', 'exercitii_medicare.sql' (which is the active tab), 'toDo.sql', and 'DropDatabaseMediCare.sql'. Below the tabs, there are two panes: 'Worksheet' and 'Query Builder'. The 'Worksheet' pane displays the PL/SQL code for the stored procedure. The 'Query Builder' pane shows the results of the query, listing patient names: Popescu Ana, Ionescu Mihai, Stoica Gabriel, Constantinescu Elena, and Dumitrescu Maria. A vertical scroll bar is visible on the right side of the main workspace. At the bottom, the 'Script Output' pane shows the message 'Task completed in 0.181 seconds' and 'Procedure AFISEAZA_PACIENTI CU_MED_SCUMPE compiled'. Another message in the same pane states 'PL/SQL procedure successfully completed.'

```

c_id_pacient tratament.id_pacient%TYPE;
nume pacienti.nume%TYPE;
prenume pacienti.prenume%TYPE;
v_cursor SYS_REFCURSOR;
BEGIN
  OPEN tratament;
  LOOP
    FETCH tratament INTO c_id_pacient;
    EXIT WHEN tratament%NOTFOUND;

    OPEN pacient (c_id_pacient);
    LOOP
      FETCH pacient INTO nume, prenume;
      EXIT WHEN pacient%NOTFOUND OR pacient%ROWCOUNT > 3;
      DBMS_OUTPUT.PUT_LINE('pacientul ' || nume || ' ' || prenume );
    END LOOP;

    CLOSE pacient;
    DBMS_OUTPUT.PUT_LINE('-----');
  END LOOP;
  CLOSE tratament;
END;
/
BEGIN
afiseaza_pacienti_cu_med_scumpe;
END;
/

```

Cod sursă:

```

CREATE OR REPLACE PROCEDURE afiseaza_pacienti_cu_med_scumpe IS
CURSOR tratament IS
  SELECT DISTINCT ID_PACIENT
  FROM (

```

```

SELECT t.id_pacient, m.pret
FROM tratamente t
JOIN tratamente_contin_medicamente tcm ON t.id_tratament = tcm.id_tratament
JOIN medicamente m ON m.id_medicament = tcm.id_medicament
GROUP BY t.id_pacient, m.pret
HAVING m.pret > (SELECT AVG(pret) FROM medicamente)
);

CURSOR pacient (v_id_pacient pacienti.id_pacient%TYPE) IS
SELECT DISTINCT p.nume, p.prenume
FROM pacienti p
JOIN internari i ON i.id_pacient = p.id_pacient
JOIN sectii_medicale s ON i.id_sectie_m = s.id_sectie_m
WHERE v_id_pacient = p.id_pacient;

c_id_pacient tratamente.id_pacient%TYPE;
nume pacienti.nume%TYPE;
prenume pacienti.prenume%TYPE;
v_cursor SYS_REFCURSOR;
BEGIN
OPEN tratament;
LOOP
FETCH tratament INTO c_id_pacient;
EXIT WHEN tratament%NOTFOUND;

OPEN pacient (c_id_pacient);
LOOP
FETCH pacient INTO nume, prenume;
EXIT WHEN pacient%NOTFOUND OR pacient%ROWCOUNT > 3;
DBMS_OUTPUT.PUT_LINE('pacientul ' || nume || ' ' || prenume );
END LOOP;

CLOSE pacient;
DBMS_OUTPUT.PUT_LINE('-----');
END LOOP;
CLOSE tratament;
END;
/

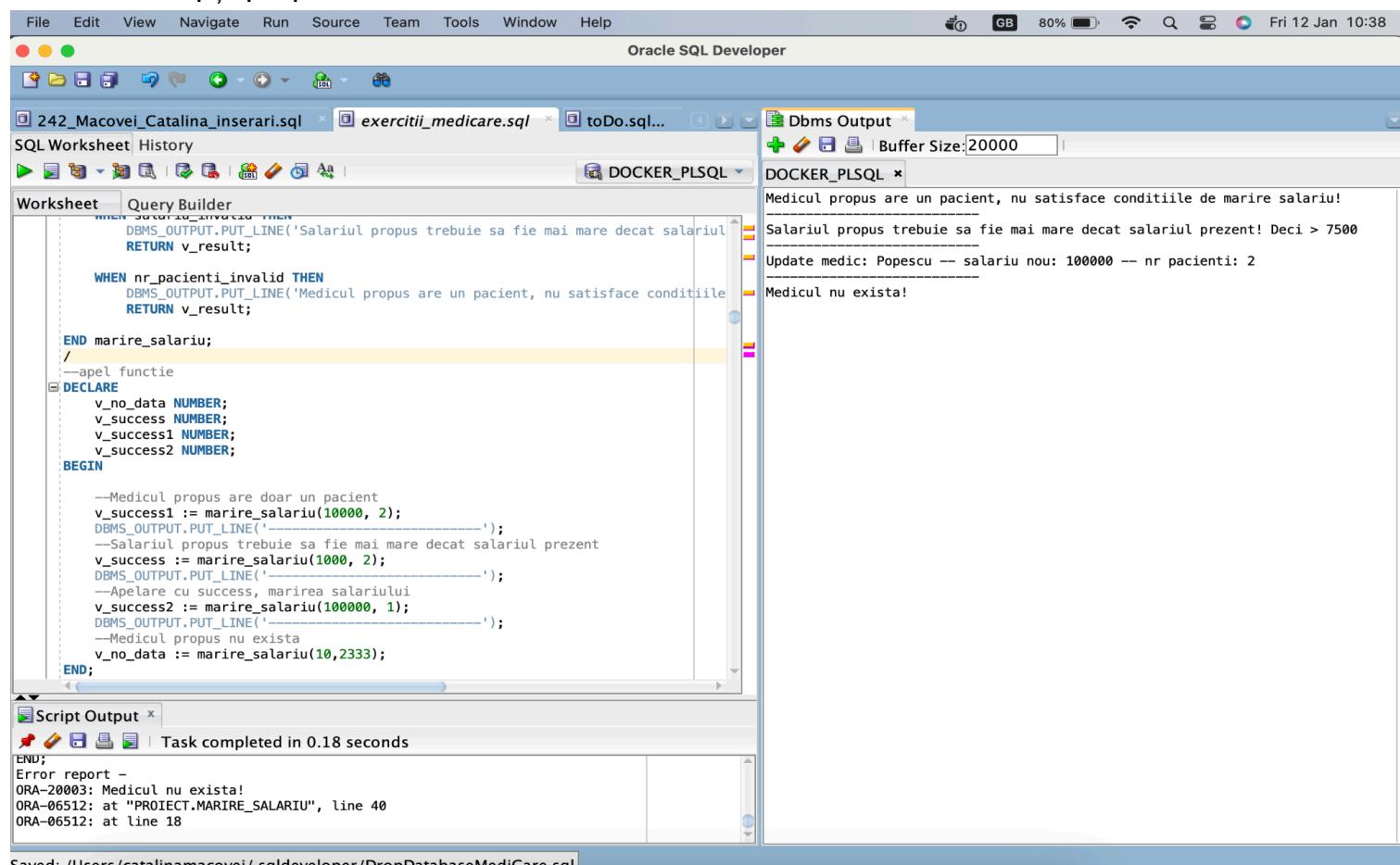
BEGIN
afiseaza_pacienti_cu_med_scumpe;
END;
/

```

Subprogram stocat - funcție & excepții proprii

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.

Problema: Creați un subprogram stocat independent tip funcție **marire_salariu**, care să primească 2 parametri - salariu propus și id medic. Dacă medicul a avut consultații cu cel puțin 2 pacienți, este valid pentru mărarea salariului, altfel nu. Să se utilizeze într-o singură comandă SQL 3 tabele definite (Medici, Pacienți, Consultații). Definiți minim 2 excepții proprii.



```

File Edit View Navigate Run Source Team Tools Window Help
Oracle SQL Developer
Fri 12 Jan 10:38
242_Macovei_Catalina_inserari.sql exercitii_medicare.sql ToDo.sql...
SQL Worksheet History
DOCKER_PLSQL
Worksheet Query Builder
WHEN nr_pacienti_invalid THEN
    DBMS_OUTPUT.PUT_LINE('Salariul propus trebuie sa fie mai mare decat salariul prezent');
    RETURN v_result;
END marire_salariu;
/
--apel functie
DECLARE
    v_no_data NUMBER;
    v_success NUMBER;
    v_success1 NUMBER;
    v_success2 NUMBER;
BEGIN
    --Medicul propus are doar un pacient
    v_success1 := marire_salariu(10000, 2);
    DBMS_OUTPUT.PUT_LINE('-----');
    --Salariul propus trebuie sa fie mai mare decat salariul prezent
    v_success := marire_salariu(1000, 2);
    DBMS_OUTPUT.PUT_LINE('-----');
    --Apelare cu success, mărarea salariului
    v_success2 := marire_salariu(1000000, 1);
    DBMS_OUTPUT.PUT_LINE('-----');
    --Medicul propus nu există
    v_no_data := marire_salariu(10,2333);
END;

```

Dbms Output

Medicul propus are un pacient, nu satisface condițiile de marire salariu!

Salariul propus trebuie sa fie mai mare decat salariul prezent! Deci > 7500

Update medic: Popescu -- salariu nou: 100000 -- nr pacienti: 2

Medicul nu există!

Script Output

Task completed in 0.18 seconds

END;
Error report -
ORA-20003: Medicul nu există!
ORA-06512: at "PROIECT.MARIRE_SALARIU", line 40
ORA-06512: at line 18

Saved: /Users/catalinamacovei/.sqldeveloper/DropDatabaseMediCare.sql

The screenshot shows the Oracle SQL Developer interface. In the top menu bar, the following items are visible: File, Edit, View, Navigate, Run, Source, Team, Tools, Window, Help. The title bar says "Oracle SQL Developer". The main window contains several tabs: "242_Macovei_Catalina_inserari.sql", "exercitii_medicare.sql", and "toDo.sql...". Below these tabs is a toolbar with icons for file operations like Open, Save, Print, and a magnifying glass. The central workspace is titled "Worksheet" and "Query Builder". It displays the following PL/SQL code:

```

CREATE OR REPLACE FUNCTION marire_salariu (
    v_salariu_propus medici.salariu%TYPE,
    v_id NUMBER
)
RETURN NUMBER IS
    v_result NUMBER := 1;
    v_salariu medici.salariu%TYPE;
    v_nume medici.nume%TYPE;
    v_nr_pacienti NUMBER;

    salariu_invalid EXCEPTION;
    nr_pacienti_invalid EXCEPTION;

BEGIN
    SELECT m.nume, m.salariu, COUNT(p.ID_PACIENT)
    INTO v_nume, v_salariu, v_nr_pacienti
    FROM medici m
    JOIN consultatii c ON c.id_medic = m.id_medic
    JOIN pacienti p ON p.id_pacient = c.id_pacient
    WHERE v_id = m.id_medic
    GROUP BY m.nume, m.salariu;

    IF v_salariu_propus <= v_salariu THEN
        RAISE salariu_invalid;
    ELSIF v_nr_pacienti < 2 THEN
        RAISE nr_pacienti_invalid;
    ELSE
        UPDATE MEDICI

```

To the right of the code editor is the "Dbms Output" window, which shows the results of the function execution:

```

Medicul propus are un pacient, nu satisface conditiile de mare salariu!
Salariul propus trebuie sa fie mai mare decat salariul prezent! Deci > 7500
Update medic: Popescu -- salariu nou: 100000 -- nr pacienti: 2
Medicul nu exista!

```

Below the code editor is the "Script Output" window, which shows the message:

```

Function MARIRE_SALARIU compiled
Task completed in 0.18 seconds

```

Procedură: tratare excepții TOO_MANY_ROWS & NO_DATA_FOUND

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.

Această secțiune conține:

- Problema limbaj natural
- Cod sursă

- Print Screen

Problema: Creați un subprogram stocat independent de tip procedură **verifica_pacienti_medicamente**, care ia ca parametru numele unui medicament și va afișa numele, sectia de internare a pacientului care îl folosește, medicamentul și prețul acestuia, dacă și numai dacă acest pacient consumă medicamente mai scumpe decât prețul mediu al medicamentelor din farmacia locală a spitalului. În acest scop, folosiți 5 dintre tabelele definite și tratați toate exceptiile inclusiv: NO_DATA_FOUND și TOO_MANY_ROWS.

- Am folosit 7 tabele in total (6 joinuri)

The screenshot shows the Oracle SQL Developer interface. In the central workspace, a PL/SQL script named `exercitiu_medicare.sql` is being edited. The script defines a procedure `verifica_pacienti_medicamente` that takes a medicine name as input and prints information about patients consuming it. It uses joins to find patients, calculates the average price, and compares individual prices to the average. It handles exceptions for no data found or too many rows. The `Dbms Output` window on the right shows the results of running the procedure, including a message about a patient named Stoica and the medicine Losartan.

```

CREATE OR REPLACE PROCEDURE verifica_pacienti_medicamente
(v_nume_medicament medicamente.nume%TYPE)
AS
    count_id NUMBER:=0;
    v_id NUMBER;
    v_nume_pacienti.nume%TYPE;
    v_pret_med medicamente.pret%TYPE;
    v_nume_med medicamente.nume%TYPE;

    WHERE m.nume = v_nume_medicament
    GROUP BY p.id_pacient, p.nume, m.pret, m.nume, sm.nume_sectie
    HAVING m.pret > (SELECT AVG(pret)
                      FROM medicamente);

    DBMS_OUTPUT.PUT_LINE('PACIENTUL: ' || v_nume || ' --> internat in sectia: ' || v_sectie_m);
    DBMS_OUTPUT.PUT_LINE('MEDICAMENT: ' || v_nume_med || ' -> pret: ' || v_pret_med);

EXCEPTION
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('Nu exista Pacienti care consuma acest medicament!');
    WHEN TOO_MANY_ROWS THEN
        DBMS_OUTPUT.PUT_LINE('Mai multi pacienti folosesc acest medicament!');
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Alta eroare!');
END verifica_pacienti_medicamente;
/

```

```

DECLARE
    v_nume_medicament_no_data_found medicamente.nume%TYPE := 'Ranitidină';
    v_nume_medicament_too_many_rows medicamente.nume%TYPE := 'Tramadol'; -- LA FEL Tramadol
    v_nume_medicament_no_err medicamente.nume%TYPE := 'Losartan'; -- Amoxicilină
BEGIN
    verifica_pacienti_medicamente(v_nume_medicament_no_data_found);
    DBMS_OUTPUT.PUT_LINE('-----');
    verifica_pacienti_medicamente(v_nume_medicament_too_many_rows);
    DBMS_OUTPUT.PUT_LINE('-----');
    verifica_pacienti_medicamente(v_nume_medicament_no_err);
END;

```

Script Output x
Task completed in 0.108 seconds
Procedure VERIFICA_PACIENTI_MEDICAMENTE compiled
PL/SQL procedure successfully completed.

```

CREATE OR REPLACE PROCEDURE verifica_pacienti_medicamente
(v_nume_medicament medicamente.nume%TYPE)
AS
    count_id NUMBER:=0;
    v_id NUMBER;
    v_nume_pacienti.nume%TYPE;
    v_pret_med medicamente.pret%TYPE;
    v_nume_med medicamente.nume%TYPE;

```

```

v_sectie_m sectii_medicale.nume_sectie%TYPE;

BEGIN

SELECT distinct p.id_pacient, p.nume, m.pret, m.nume, sm.nume_sectie
INTO v_id, v_nume, v_pret_med, v_nume_med, v_sectie_m
FROM PACIENTI P

JOIN CONSULTATII C ON C.ID_PACIENT = P.ID_PACIENT
JOIN TRATAMENTE T ON T.ID_PACIENT = C.ID_PACIENT
JOIN TRATAMENTE_CONTIN_MEDICAMENTE TCM ON TCM.ID_TRATAMENT =
T.ID_TRATAMENT
JOIN MEDICAMENTE M ON M.ID_MEDICAMENT = TCM.ID_MEDICAMENT
JOIN INTERNARI I ON I.ID_PACIENT = P.ID_PACIENT
JOIN SECTII_MEDICALE SM ON SM.ID_SECTIE_M = I.ID_SECTIE_M

WHERE m.nume = v_nume_medicament
GROUP BY p.id_pacient, p.nume, m.pret, m.nume, sm.nume_sectie
HAVING m.pret > (SELECT AVG(pret)
                  FROM medicamente);

DBMS_OUTPUT.PUT_LINE('PACIENTUL: ' || v_nume || ' --> internat in
sectia: ' || v_sectie_m);
DBMS_OUTPUT.PUT_LINE('MEDICAMENT: ' || v_nume_med || ' -> pret: ' ||
v_pret_med);
EXCEPTION
  WHEN NO_DATA_FOUND THEN
    DBMS_OUTPUT.PUT_LINE('Nu exista Pacienti care consuma acest
medicament!');
  WHEN TOO_MANY_ROWS THEN
    DBMS_OUTPUT.PUT_LINE('Mai multi pacienti folosesc acest
medicament!');
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('Alta eroare!');
END verifica_pacienti_medicamente;
/

DECLARE
  v_nume_medicament_no_data_found medicamente.nume%TYPE := 'Ranitidină';
  v_nume_medicament_too_many_rows medicamente.nume%TYPE := 'Tramadol'; --
LA FEL Tramadol
  v_nume_medicament_no_err medicamente.nume%TYPE := 'Losartan';
--Amoxicilină
BEGIN
  verifica_pacienti_medicamente(v_nume_medicament_no_data_found);

```

```

DBMS_OUTPUT.PUT_LINE('-----');
----- ');
verifica_pacienti_medicamente(v_numere_medicament_too_many_rows);

DBMS_OUTPUT.PUT_LINE('-----');
----- ');
verifica_pacienti_medicamente(v_numere_medicament_no_err);
END;
/

```

Definiți un trigger de tip LMD la nivel de linie.

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

Această secțiune conține:

- Problema limbaj natural
- Cod sursă
- Print Screen

Problema: Creați un trigger care folosește o procedură

MODIFIC_NR_PATURI_LIBERE, care ia 2 parametri, id_sectie_m și operația (1, -1), care urmează să modifice numărul de paturi libere în secție, pe moment ce au loc adaugarea de internări, modificarea sau ștergerea lor. Dacă există paturi libere, atunci se va insera internarea, altfel vom declanșa o excepție.

Pentru început voi folosi secția cu id = SEC4, deoarece are 1 pat liber. Astfel, o inserare va avea loc cu succes, iar la următoarea se va declanșa excepția **nu_exista_paturi_libere.

File Edit View Navigate Run Source Team Tools Window Help Fri 12 Jan 11:24

Oracle SQL Developer : /Users/catalinamacovei/.sqldeveloper/exercitii_medicare.sql

242_Macovei_Catalina_inserari.sql exercitii_medicare.sql Compiler - Log ToDo.sql DropDatabaseM... DOCKER_PLSQL DOCKER_PLSQL

SQL Worksheet History

Worksheet Query Builder

```

--SE INTRODUC O INTERNARE
SELECT NR_PATURI_LIBERE INTO exista_paturi FROM SECTII_MEDICALE WHERE :NEW.ID_SECTIE_M = ID_SECTIE_M ;

IF exista_paturi = 0
    THEN RAISE nu_exista_paturi_libere;
ELSE
    MODIFIC_NR_PATURI_LIBERE(:NEW.ID_SECTIE_M, -1);
END IF;
END IF;

EXCEPTION
    WHEN nu_exista_paturi_libere THEN
        RAISE_APPLICATION_ERROR(-20002,'NU MAI EXISTA PATURI!');
END;
/

INSERT INTO INTERNARI (ID_PACIENT, ID_SECTIE_M, DATA_INTERNARE)
VALUES (3, 'SEC4', TO_DATE('2023-05-29', 'YYYY-MM-DD'));

INSERT INTO INTERNARI (ID_PACIENT, ID_SECTIE_M, DATA_INTERNARE)
VALUES (6, 'SEC4', TO_DATE('2023-05-29', 'YYYY-MM-DD'));

```

Script Output Query Result Task completed in 0.1 seconds

Trigger TRIGGER_INTERNARI compiled

1 row inserted.

Error starting at line : 345 in command -
`INSERT INTO INTERNARI (ID_PACIENT, ID_SECTIE_M, DATA_INTERNARE)
VALUES (6, 'SEC4', TO_DATE('2023-05-29', 'YYYY-MM-DD'))`
Error report -
ORA-20002: NU MAI EXISTA PATURI!
ORA-06512: at "PROJECT.TRIGGER_INTERNARI", line 35
ORA-04081: error during execution of trigger 'PROJECT.TRIGGER_INTERNARI'

Saved: /Users/catalinamacovei/.sqldeveloper/DropDatabaseMediCare.sql | Line 342 Column 1 | Insert | Modified | Unix/Mac: LF

File Edit View Navigate Run Source Team Tools Window Help Fri 12 Jan 11:26

Oracle SQL Developer : /Users/catalinamacovei/.sqldeveloper/exercitii_medicare.sql

242_Macovei_Catalina_inserari.sql exercitii_medicare.sql Compiler - Log ToDo.sql DropDatabaseM... DOCKER_PLSQL DOCKER_PLSQL

SQL Worksheet History

Worksheet Query Builder

```

CREATE OR REPLACE PROCEDURE MODIFIC_NR_PATURI_LIBERE
(v_id_sec sectii_medicale.id_sectie_m%TYPE, operatie NUMBER) AS
BEGIN
    UPDATE sectii_medicale
    SET NR_PATURI_LIBERE = NVL(NR_PATURI_LIBERE, 0) + operatie
    WHERE id_sectie_m = v_id_sec;
END;
/

CREATE OR REPLACE TRIGGER trigger_internari
    BEFORE DELETE OR UPDATE OR INSERT OF ID_SECTIE_M ON internari
    FOR EACH ROW
DECLARE
    exista_paturi SECTII_MEDICALE.NR_PATURI_LIBERE%TYPE;
    nu_exista_paturi_libere EXCEPTION;
BEGIN
    -- am rescris select query deoarece la delete primeam "no data found", nu gasesc :OLD.ID_SECTIE_M

    IF DELETING THEN
        SELECT NR_PATURI_LIBERE INTO exista_paturi FROM SECTII_MEDICALE WHERE :OLD.ID_SECTIE_M = ID_SECTIE_M ;
        -- se sterge o internare
        MODIFIC_NR_PATURI_LIBERE(:OLD.ID_SECTIE_M, 1);
    END IF;

```

Script Output Query Result All Rows Fetched: 6 in 0.004 seconds

ID_SECTIE_M	NUME_SECTIE	NR_PATURI	NR_PATURI_LIBERE
1 SEC1	Chirurgie	30	25
2 SEC2	Cardiologie	25	20
3 SEC3	Pediatrie	40	35
4 SEC4	Ortopedie	20	0
5 SEC5	Neurologie	15	10
6 SEC6	Oftalmologie	18	10

SQL Worksheet History

Worksheet Query Builder

```

ELSE
    --SE INTRODUC O INTERNARE
    SELECT NR_PATURI_LIBERE INTO exista_paturi FROM SECTII_MEDICALE WHERE :NEW.ID_SECTIE_M = ID_SECTIE_M ;

    IF exista_paturi = 0
        THEN RAISE nu_exista_paturi_libere;
    ELSE
        MODIFIC_NR_PATURI_LIBERE(:NEW.ID_SECTIE_M, -1);
    END IF;
END IF;
EXCEPTION
    WHEN nu_exista_paturi_libere THEN
        RAISE_APPLICATION_ERROR(-20002,'NU MAI EXISTA PATURI!');
END;
/

INSERT INTO INTERNARI (ID_PACIENT, ID_SECTIE_M, DATA_INTERNARE)
VALUES (3, 'SEC4', TO_DATE('2023-05-29', 'YYYY-MM-DD'));

INSERT INTO INTERNARI (ID_PACIENT, ID_SECTIE_M, DATA_INTERNARE)
VALUES (6, 'SEC4', TO_DATE('2023-05-29', 'YYYY-MM-DD'));

UPDATE INTERNARI
SET ID_SECTIE_M = 'SEC4'
WHERE id_patient = 3 AND TO_DATE(DATA_INTERNARE) = TO_DATE('2023-MAY-29', 'YYYY-MM-DD');

```

Script Output x | Task completed in 0.072 seconds

```

Error starting at line : 348 in command -
UPDATE INTERNARI
SET ID_SECTIE_M = 'SEC4'
WHERE id_patient = 3 AND TO_DATE(DATA_INTERNARE) = TO_DATE('2023-MAY-29', 'YYYY-MM-DD')
Error report -
ORA-20002: NU MAI EXISTA PATURI!
ORA-06512: at "PROIECT.TRIGGER_INTERNARI", line 35
ORA-04088: error during execution of trigger 'PROIECT.TRIGGER_INTERNARI'

```

SQL Worksheet History

Worksheet Query Builder

```

    THEN RAISE nu_exista_paturi_libere;
ELSE
    MODIFIC_NR_PATURI_LIBERE(:NEW.ID_SECTIE_M, -1);
END IF;
END IF;
EXCEPTION
    WHEN nu_exista_paturi_libere THEN
        RAISE_APPLICATION_ERROR(-20002, 'NU MAI EXISTA PATURI!');
END;
/

INSERT INTO INTERNARI (ID_PACIENT, ID_SECTIE_M, DATA_INTERNARE)
VALUES (3, 'SEC4', TO_DATE('2023-05-29', 'YYYY-MM-DD'));

INSERT INTO INTERNARI (ID_PACIENT, ID_SECTIE_M, DATA_INTERNARE)
VALUES (6, 'SEC4', TO_DATE('2023-05-29', 'YYYY-MM-DD'));

UPDATE INTERNARI
SET ID_SECTIE_M = 'SEC4'
WHERE id_pacient = 3 AND TO_DATE(DATA_INTERNARE) = TO_DATE('2023-MAY-29', 'YYYY-MM-DD');

DELETE FROM INTERNARI
WHERE INTERNARI.id_pacient = 5
    AND INTERNARI.ID_SECTIE_M = 'SEC4'
    AND TO_DATE(DATA_INTERNARE, 'DD-MM-YYYY') = TO_DATE('25-SEP-23', 'DD-MM-YYYY');
SELECT * FROM SECTII_MEDICALE;

```

Script Output x Query Result x

SQL | All Rows Fetched: 6 in 0.004 seconds

ID_SECTIE_M	NUME_SECTIE	NR_PATURI	NR_PATURI_LIBERE
1 SEC1	Chirurgie	30	25
2 SEC2	Cardiologie	25	20
3 SEC3	Pediatrie	40	35
4 SEC4	Ortopedie	20	1
5 SEC5	Neurologie	15	10
6 SEC6	Oftalmologie	18	10

Cod sursă:

```

CREATE OR REPLACE PROCEDURE MODIFIC_NR_PATURI_LIBERE
    (v_id_sec sectii_medicale.id_sectie_m%TYPE, operatie NUMBER) AS
BEGIN
    UPDATE sectii_medicale
    SET NR_PATURI_LIBERE = NVL(NR_PATURI_LIBERE, 0) + operatie
    WHERE id_sectie_m = v_id_sec;
END;
/

```

```

CREATE OR REPLACE TRIGGER trigger_internari
  BEFORE DELETE OR UPDATE OR INSERT OF ID_SECTIE_M ON internari
  FOR EACH ROW
DECLARE
  exista_paturi SECTII_MEDICALE.NR_PATURI_LIBERE%TYPE;
  nu_exista_paturi_libere EXCEPTION;
BEGIN
  -- am rescris select query deoarece la delete primeam "no data found", nu gasea
  :OLD.ID_SECTIE_M

  IF DELETING THEN
    SELECT NR_PATURI_LIBERE INTO exista_paturi FROM SECTII_MEDICALE WHERE
  :OLD.ID_SECTIE_M = ID_SECTIE_M ;
    -- se sterge o internare
    MODIFIC_NR_PATURI_LIBERE(:OLD.ID_SECTIE_M, 1);

  ELSIF UPDATING THEN
    --SE MODIFICA SECTIA UNEI INTERNARI
    SELECT NR_PATURI_LIBERE INTO exista_paturi FROM SECTII_MEDICALE WHERE
  :NEW.ID_SECTIE_M = ID_SECTIE_M ;

    MODIFIC_NR_PATURI_LIBERE(:OLD.ID_SECTIE_M, 1);

  IF exista_paturi = 0
    THEN RAISE nu_exista_paturi_libere;
  MODIFIC_NR_PATURI_LIBERE(:NEW.ID_SECTIE_M, -1);
  END IF;

  ELSE
    --SE INTRODUCE O INTERNARE
    SELECT NR_PATURI_LIBERE INTO exista_paturi FROM SECTII_MEDICALE WHERE
  :NEW.ID_SECTIE_M = ID_SECTIE_M ;

    IF exista_paturi = 0
      THEN RAISE nu_exista_paturi_libere;
    ELSE
      MODIFIC_NR_PATURI_LIBERE(:NEW.ID_SECTIE_M, -1);
    END IF;
  END IF;
EXCEPTION
  WHEN nu_exista_paturi_libere THEN
    RAISE_APPLICATION_ERROR(-20002,'NU MAI EXISTA PATURI!');
END;

```

/

```

INSERT INTO INTERNARI (ID_PACIENT, ID_SECTIE_M, DATA_INTERNARE)
VALUES (3, 'SEC4', TO_DATE('2023-05-29', 'YYYY-MM-DD'));

INSERT INTO INTERNARI (ID_PACIENT, ID_SECTIE_M, DATA_INTERNARE)
VALUES (6, 'SEC4', TO_DATE('2023-05-29', 'YYYY-MM-DD'));

UPDATE INTERNARI
SET ID_SECTIE_M = 'SEC4'
WHERE id_pacient = 3 AND TO_DATE(DATA_INTERNARE) = TO_DATE('2023-MAY-29',
'YYYY-MM-DD');

DELETE FROM INTERNARI
WHERE INTERNARI.id_pacient = 5
AND INTERNARI.ID_SECTIE_M = 'SEC4'
AND TO_DATE(DATA_INTERNARE, 'DD-MM-YYYY') = TO_DATE('25-SEP-23', 'DD-MM-YYYY');

SELECT * FROM SECTII_MEDICALE;
SELECT * FROM internari;
DROP TRIGGER trigger_internari;

```

Definiți un trigger de tip LMD la nivel de comandă

11. Definiți un *trigger* de tip LMD la nivel de comandă. Declansați *trigger-ul*.

Problema: Sa presupunem ca un medic poate îngriji zilnic 3 pacienți. Definiți un trigger care se declanșează dacă numărul de internări depășește limita.

The screenshot shows the Oracle SQL Developer interface with the following details:

- File Bar:** File, Edit, View, Navigate, Run, Source, Team, Tools, Window, Help.
- Title Bar:** Oracle SQL Developer : /Users/catalinamacovei/.sqldeveloper/exercitii_medicare.sql
- Toolbar:** Standard toolbar with icons for file operations.
- Tab Bar:** 242_Macovei_Catalina_inserari.sql, exercitii_medicare.sql, Compiler - Log, ToDo.sql, DropDatabaseM..., Dbms Output, DOCKER_PLSQL, DOCKER_PLSQL.
- Worksheet:** The main area contains the PL/SQL code for the trigger. A portion of the code is highlighted in blue.

```

INTO v_medici
FROM medici;

SELECT DISTINCT COUNT(id_pacient)
INTO v_internari
FROM INTERNARI;

v_medici := v_medici * 3;

IF v_internari > v_medici THEN
RAISE_APPLICATION_ERROR(-20001,'Ati depasit numarul maxim de internari!');
END IF;
END;
/
drop trigger gestioneaza_nr_internari;
BEGIN
FOR j in 10..30 LOOP
INSERT INTO INTERNARI VALUES (1, 'SEC2', TO_DATE('2023-10-' || j), 'YYYY-MM-DD');
END LOOP;
END;

```

- Script Output:** Shows the trigger compilation message: "Trigger GESTIONEAZA_NR_INTERNARI compiled".
- Error Output:** Displays the error starting at line 390: "Error starting at line : 390 in command - BEGIN FOR j in 10..30 LOOP INSERT INTO INTERNARI VALUES (1, 'SEC2', TO_DATE('2023-10-' || j), 'YYYY-MM-DD'); END LOOP; END; Error report - ORA-20001: Ati depasit numarul maxim de internari! ORA-06512: at "PROJECT.GESTIONEAZA_NR_INTERNARI", line 17 ORA-04088: error during execution of trigger 'PROJECT.GESTIONEAZA_NR_INTERNARI'".
- Status Bar:** Saved: /Users/catalinamacovei/.sqldeveloper/DropDatabaseMediCare.sql | Line 388 Column 1 | Insert | Modified | Unix/Mac: LF

CREATE OR REPLACE TRIGGER gestioneaza_nr_internari
BEFORE INSERT ON internari

DECLARE

```
v_internari NUMBER;
v_medici NUMBER;
```

BEGIN

```
SELECT DISTINCT COUNT(*)
INTO v_medici
FROM medici;
```

```
SELECT DISTINCT COUNT(id_pacient)
INTO v_internari
FROM INTERNARI;
```

```
v_medici := v_medici * 3;
```

```
IF v_internari > v_medici THEN
RAISE_APPLICATION_ERROR(-20001,'Ati depasit numarul maxim de internari!');
END IF;
```

END;

```

/
drop trigger gestioneaza_nr_internari;
BEGIN
  FOR j in 10..30 LOOP
    INSERT INTO INTERNARI VALUES (1, 'SEC2', TO_DATE(('2023-10-' || j), 'YYYY-MM-DD'));
  END LOOP;
END;

```

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

12. **Problema:** Creați un trigger care se va declanșa la nivel de sistem. Dacă sunt efectuate schimbări din alt cont decat 'PROIECT', atunci se va declanșa o eroare. Salvati modificarile asupra schemei in tabela audit_trigger.

Declansare trigger, istoric modificari:

The screenshot shows the Oracle SQL Developer interface with the following details:

- File Bar:** File, Edit, View, Navigate, Run, Source, Team, Tools, Window, Help.
- Title Bar:** Oracle SQL Developer : /Users/catalinamacovei/.sqldeveloper/exercitii_medicare.sql
- Toolbar:** Standard toolbar with various icons for file operations.
- Script Area:** Contains the SQL code for creating a trigger named 'gestionare_modificari_ldd'. The trigger is created after an object is created or altered, and it checks if the user is 'PROIECT'. If not, it raises an application error. It also inserts a record into the 'audit_trigger' table with the current user, database name, event type, object name, and timestamp.
- Output Area:** Shows the results of the query 'SELECT * FROM audit_trigger;'. The results are as follows:

UTILIZATOR	NUME_BD	EVENIMENT	NUME_OBJECT	DATA
1 PROIECT	ORCLPDB1	ALTER	CONSULTATII	12-JAN-24
2 PROIECT	ORCLPDB1	ALTER	CONSULTATII	12-JAN-24
3 PROIECT	ORCLPDB1	ALTER	CONSULTATII	12-JAN-24

```
-- trigger LDD - system
CREATE TABLE audit_trigger
(utilizator VARCHAR2(30),
 nume_bd VARCHAR2(50),
 eveniment VARCHAR2(20),
 nume_object VARCHAR2(30),
 data DATE);

CREATE OR REPLACE TRIGGER gestionare_modificari_ldd
AFTER CREATE OR DROP OR ALTER ON SCHEMA
BEGIN
IF 'PROIECT' != USER
THEN
    RAISE_APPLICATION_ERROR(-20900, 'Aveti voie sa faceti modificari doar din contul
<<PROIECT>>');
END IF;
INSERT INTO audit_trigger
VALUES (SYS.LOGIN_USER, SYS.DATABASE_NAME, SYS.SYSEVENT,
SYS.DICTIONARY_OBJ_NAME, SYSDATE);
END;
/
--testing
alter table consultatii add test varchar2(10);
alter table consultatii add test2 varchar2(10);
alter table consultatii drop column test2;

SELECT * FROM audit_trigger;
DROP TRIGGER gestionare_modificari_ldd;
drop table audit_trigger;
```

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

```

File Edit View Navigate Run Source Team Tools Window Help
Oracle SQL Developer : /Users/catalinamacovei/.sqldeveloper/exercitii_medicare.sql
45% Fri 12 Jan 11:56
242_Macovei_Catalina_inserari.sql exercitii_medicare.sql Compiler - Log toDo.sql DropDatabaseM...
SQL Worksheet History DOCKER_PLSQL
Worksheet Query Builder
--Testare pachet:
--6
execute proiect_medicare.evaluare_medici();
--7
execute proiect_medicare.afiseaza_pacienti_cu_med_scumpe();
-- proc trigger
execute proiect_medicare.MODIFIC_NR_PATURI_LIBERE('SEC2', 1);
-- testez intr-un bloc PLSQL:
DECLARE
    v_salariu_propus medici.salariu%TYPE := 1000;
    v_id NUMBER := 2;
    v_success NUMBER;
    v_success1 NUMBER;
    v_success2 NUMBER; --APEL SUCCES

```

Script Output x Query Result x Task completed in 0.128 seconds

```

Package Body PROIECT_MEDICARE compiled

PL/SQL procedure successfully completed.

PL/SQL procedure successfully completed.

PL/SQL procedure successfully completed.

PL/SQL procedure successfully completed.

```

Dbms Output x Buffer Size:20000 DOCKER_PLSQL

```

Id medic: 8 Stoica Raluca
Consultatii:
Data consultatie: 18-DEC-23
Pacient:
Mihai Ana-Maria

Id medic: 4 Georgescu Elena
Consultatii:
Data consultatie: 18-DEC-23
Pacient:
Constantinescu Elena

Id medic: 9 Popa George
Consultatii:
Data consultatie: 18-DEC-23
Pacient:
Popa Cristina

pacientul Popescu Ana
pacientul Ionescu Mihai
pacientul Stoica Gabriel
pacientul Dumitrescu Maria

Medicul propus are un pacient, nu satisfac!
Actualizarea salariului a esuat!

Salariul propus trebuie sa fie mai mare !
Actualizarea salariului a esuat!

Salariul propus trebuie sa fie mai mare !
Actualizarea salariului a esuat!

```

```

CREATE OR REPLACE PACKAGE proiect_medicare AS
    PROCEDURE evaluare_medici;
    PROCEDURE afiseaza_pacienti_cu_med_scumpe;
    FUNCTION marire_salariu (
        v_salariu_propus medici.salariu%TYPE,
        v_id NUMBER)
        RETURN NUMBER;
    PROCEDURE MODIFIC_NR_PATURI_LIBERE
        (v_id_sec sectii_medicale.id_sectie_m%TYPE, operatie NUMBER);

END proiect_medicare;
/

```

```
CREATE OR REPLACE PACKAGE BODY proiect_medicare AS
```

```
-- sarcina 6
```

```
PROCEDURE evaluare_medici AS
```

```
    TYPE tablou_idx IS TABLE OF medici%ROWTYPE INDEX BY BINARY_INTEGER;
```

```
    TYPE tablou_imbr IS TABLE OF pacienti%ROWTYPE;
```

```
    TYPE vector IS VARRAY(3) OF consultatii%ROWTYPE;
```

```
med tablou_idx;
```

```
pacienti_med tablou_imbr := tablou_imbr();
```

```
cons_med vector := vector();
```

```
BEGIN
```

```
-- Salvare in tablou
```

```
SELECT ID_MEDIC, NUME, PRENUME, SALARIU, MEDIC_SEF, ID_SECTIE_M
```

```
BULK COLLECT INTO med
```

```
FROM (
```

```
    SELECT COUNT(*) AS NR_CONS, M.ID_MEDIC, M.NUME, M.PRENUME, M.SALARIU,
```

```
M.MEDIC_SEF, M.ID_SECTIE_M FROM MEDICI M
```

```
    INNER JOIN CONSULTATII C ON C.ID_MEDIC = M.ID_MEDIC
```

```
    GROUP BY M.ID_MEDIC, M.ID_MEDIC, M.NUME, M.PRENUME, M.SALARIU,
```

```
M.MEDIC_SEF, M.ID_SECTIE_M
```

```
    ORDER BY NR_CONS ASC) T
```

```
WHERE ROWNUM <= 3;
```

```
FOR i IN 1..med.COUNT LOOP
```

```
-- Pentru pacienti
```

```
FOR patient_rec IN (
```

```
    SELECT pa.*
```

```
    FROM pacienti pa
```

```
    INNER JOIN consultatii c ON c.id_pacient = pa.id_pacient
```

```
    WHERE c.id_medic = med(i).id_medic
```

```
) LOOP
```

```
    pacienti_med.extend;
```

```
    pacienti_med(pacienti_med.last) := patient_rec;
```

```
END LOOP;
```

```
-- Pentru consultatii
```

```
FOR cons_rec IN (
```

```
    SELECT *
```

```

        FROM consultatii
        WHERE id_medic = med(i).id_medic
    ) LOOP
        cons_med.extend;
        cons_med(cons_med.last) := cons_rec;
    END LOOP;
END LOOP;

-- Afisare medici
FOR k IN 1..med.COUNT LOOP
    DBMS_OUTPUT.PUT_LINE('Id medic: ' || med(k).id_medic || ' ' || med(k).nume || ' ' ||
med(k).prenume);
    DBMS_OUTPUT.PUT_LINE('Consultatii:');
    -- Afisare consultatii pentru fiecare medic
    FOR j IN 1..cons_med.COUNT LOOP
        IF cons_med(j).id_medic = med(k).id_medic THEN
            DBMS_OUTPUT.PUT_LINE('Data consultatie: ' || cons_med(j).data_cons || ' ');
            -- Afisare pacienti pentru fiecare consultatie
            DBMS_OUTPUT.PUT_LINE('Pacient: ');
            FOR i IN 1..pacienti_med.COUNT LOOP
                IF cons_med(j).id_pacient = pacienti_med(i).id_pacient THEN
                    DBMS_OUTPUT.PUT_LINE(pacienti_med(i).nume || ' ' ||
pacienti_med(i).prenume);
                END IF;
            END LOOP;
        END IF;
    END LOOP;
    DBMS_OUTPUT.NEW_LINE;
END LOOP;

END evaluare_medici;

```

```

--- sarcina 7
PROCEDURE afiseaza_pacienti_cu_med_scumpe AS
CURSOR tratament IS
    SELECT DISTINCT t.id_pacient
    FROM tratamente t
    WHERE t.id_pacient IN (
        SELECT t.id_pacient

```

```

FROM tratamente t
JOIN tratamente_contin_medicamente tcm ON t.id_tratament = tcm.id_tratament
JOIN medicamente m ON m.id_medicament = tcm.id_medicament
WHERE t.id_pacient = t.id_pacient
GROUP BY m.pret, t.id_pacient
HAVING m.pret > (SELECT AVG(pret) FROM medicamente)
);

CURSOR pacient (v_id_pacient pacienti.id_pacient%TYPE) IS
SELECT DISTINCT p.nume, p.prenume
FROM pacienti p
RIGHT JOIN internari i ON i.id_pacient = p.id_pacient
WHERE v_id_pacient = p.id_pacient;

c_id_pacient tratamente.id_pacient%TYPE;
nume pacienti.nume%TYPE;
prenume pacienti.prenume%TYPE;
v_cursor SYS_REFCURSOR;
BEGIN
OPEN tratament;
LOOP
  FETCH tratament INTO c_id_pacient;
  EXIT WHEN tratament%NOTFOUND;

  OPEN pacient (c_id_pacient);
  LOOP
    FETCH pacient INTO nume, prenume;
    EXIT WHEN pacient%NOTFOUND OR pacient%ROWCOUNT > 3;
    DBMS_OUTPUT.PUT_LINE('pacientul ' || nume || ' ' || prenume);
  END LOOP;

  CLOSE pacient;
  DBMS_OUTPUT.PUT_LINE('-----');
END LOOP;
CLOSE tratament;
END afiseaza_pacienti_cu_med_scumpe;

--- sarcina 8
FUNCTION marire_salariu (

```

```

v_salariu_propus medici.salariu%TYPE,
v_id NUMBER
)
RETURN NUMBER IS
  v_result NUMBER := 2;
  v_salariu medici.salariu%TYPE;
  v_nume medici.nume%TYPE;
  v_nr_pacienti NUMBER;

  salariu_invalid EXCEPTION;

  nr_pacienti_invalid EXCEPTION;

BEGIN
  SELECT m.nume, m.salariu, COUNT(P.ID_PACIENT)
    INTO v_nume, v_salariu, v_nr_pacienti
    FROM medici m
    JOIN consultatii c ON c.id_medic = m.id_medic
    JOIN pacienti p ON p.id_pacient = c.id_pacient
    WHERE v_id = m.id_medic
    GROUP BY m.nume, m.salariu;

  IF v_salariu_propus <= v_salariu THEN
    RAISE salariu_invalid;
  ELSIF v_nr_pacienti < 2 THEN
    RAISE nr_pacienti_invalid;
  ELSE
    UPDATE MEDICI
      SET salariu = v_salariu_propus
      WHERE id_medic = v_id;
    v_result := 1; -- REZULTAT SUCCES
    DBMS_OUTPUT.PUT_LINE('Update medic: ' || v_nume || '-- salariu nou: ' ||
v_salariu_propus || '-- nr pacienti: ' || v_nr_pacienti);
  END IF;

  RETURN v_result;
EXCEPTION
  WHEN NO_DATA_FOUND THEN
    DBMS_OUTPUT.PUT_LINE('Medicul nu exista!');
    RAISE_APPLICATION_ERROR(-20003, 'Medicul nu exista!');

```

```

    RETURN v_result;

WHEN salariu_invalid THEN
    DBMS_OUTPUT.PUT_LINE('Salariul propus trebuie sa fie mai mare decat salariul
prezent! Deci > ' || v_salariu);
    RETURN v_result;

WHEN nr_pacienti_invalid THEN
    DBMS_OUTPUT.PUT_LINE('Medicul propus are un pacient, nu satisface conditiile de
marire salariu!');
    RETURN v_result;

END marire_salariu;

PROCEDURE MODIFIC_NR_PATURI_LIBERE
(v_id_sec sectii_medicale.id_sectie_m%TYPE, operatie NUMBER) AS
BEGIN
    UPDATE sectii_medicale
    SET NR_PATURI_LIBERE = NVL(NR_PATURI_LIBERE, 0) + operatie
    WHERE id_sectie_m = v_id_sec;
END MODIFIC_NR_PATURI_LIBERE;

END proiect_medicare;
/

--Testare pachet:

--6
execute proiect_medicare.evaluare_medici();

--7
execute proiect_medicare.afiseaza_pacienti_cu_med_scumpe();

-- proc trigger
execute proiect_medicare.MODIFIC_NR_PATURI_LIBERE('SEC2', 1);
-- testez intr-un bloc PLSQL:
DECLARE
    v_salariu_propus medici.salariu%TYPE := 1000;
    v_id NUMBER := 2;
    v_success NUMBER;

```

```

v_success1 NUMBER;
v_success2 NUMBER; --APEL SUCCES

BEGIN
--Medicul propus are doar un pacient
v_success1 := proiect_medicare.marire_salariu(10000, 2);

IF v_success1 = 1 THEN
    DBMS_OUTPUT.PUT_LINE('Salariul a fost actualizat cu succes!');
ELSE
    DBMS_OUTPUT.PUT_LINE('Actualizarea salariului a esuat!');
END IF;

DBMS_OUTPUT.PUT_LINE('-----');

--Salariul propus trebuie sa fie mai mare decat salariul prezent
v_success := proiect_medicare.marire_salariu(v_salariu_propus, v_id);
IF v_success = 1 THEN
    DBMS_OUTPUT.PUT_LINE('Salariul a fost actualizat cu succes!');
ELSE
    DBMS_OUTPUT.PUT_LINE('Actualizarea salariului a esuat!');
END IF;

DBMS_OUTPUT.PUT_LINE('-----');

--Apelare cu success, marirea salariului
v_success2 := proiect_medicare.marire_salariu(10000, 1);
IF v_success2 = 1 THEN
    DBMS_OUTPUT.PUT_LINE('Salariul a fost actualizat cu succes!');
ELSE
    DBMS_OUTPUT.PUT_LINE('Actualizarea salariului a esuat!');
END IF;
END;
/

```

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).

Problema: Creați un pachet care să mărească salariul a top 3 medici în funcție de numărul de consultații în ordine descrescătoare. Coeficientul de mărire este de 10%. După efectuarea modificărilor, inserati in tabela **istoric_salariu**, pentru fiecare pacient: data actualizării salariului, salariul vechi, salariul nou. Inițializați aceste date, afișați date despre medic (id, nume, salariu), consultațiile împreună cu data și pacienții respectivi. Organizați aceste funcționalități după scop în subprograme, evitând structuri repetitive.

Structura pachetului PACHET_ACTUALIZARI_MEDICARE:

1. Tipuri de date complexe :

- **Colecții :**
 - Vector - medici
 - Tablou imbricat - consultații, pacienți
- **V_istoric_salariu** - va colecta inregistrarea care urmează să fie inserată dinamic în istoric_salariu după mărire.

2. Cursor:

- Inițializarea tabloului - medici

3. Proceduri:

- Inițializarea datelor, tablourile: medici, pacienți, consultații
- Afisarea datelor: medici, pacienți, consultații

4. Funcții:

- Recalcularea salariului, include formula de calcul și returnează salariul nou
- Actualizarea datelor din BD. Acest program integrează structurile create anterior, adică sunt apelate procedurile de inițializare, afișare a datelor și funcția de recalculare a salariului.

Inserează modificările salariului în tabela istoric_salariu pentru a tine evidența modificărilor salariului pentru fiecare medic. Astfel dacă actualizările au avut loc cu succes, atunci funcția returnează 1, altfel

returnează -2. Aceasta ne ajuta sa testam daca actualizarea a avut loc cu succes.

The screenshot shows the Oracle SQL Developer interface with the following details:

- File Bar:** File, Edit, View, Navigate, Run, Source, Team, Tools, Window, Help.
- Title Bar:** Oracle SQL Developer : /Users/catalinamacovei/.sqldeveloper/exercitii_medicare.sql
- Toolbar:** Standard icons for file operations.
- Tab Bar:** 242_Macovei_Catalina_inserari.sql, exercitii_medicare.sql (active), Compiler - Log, ToDo.sql... , DOCKER_PLSQL.
- Worksheet:** Shows the PL/SQL code for the package body PACHET_ACTUALIZARI_MEDICARE. The code includes a procedure marire_salariu_pac and a function pachet_actualizari_medicare. It uses DBMS_OUTPUT.PUT_LINE to print messages to the console.
- Script Output:** Shows the results of the compilation and execution of the package body.
- Query Result:** Shows the output from the DBMS_OUTPUT.PUT_LINE statements.
- Dbms Output:** Shows the actualized data for three doctors (Constantinescu Vlad, Popescu Andrei, Ionescu Mihai) across different patient consultancies.
- DOCKER_PLSQL:** A log window showing the execution of the PL/SQL code.
- Status Bar:** Fri 12 Jan 12:26.

```

PL/SQL procedure successfully completed.

Saved: /Users/catalinamacovei/.sqldeveloper/exercitii_medicare.sql

```

```

);
      v_result := 1; -- REZULTAT SUCCES
END LOOP;

RETURN v_result;

END marire_salariu_pac;

END pachet_actualizari_medicare;
/

--utilizare pachet
DECLARE
  v_success NUMBER;
BEGIN
  v_success := pachet_actualizari_medicare.marire_salariu_pac();
  IF v_success = 1 THEN
    DBMS_OUTPUT.PUT_LINE('Actualizare cu succes!');
  ELSE
    DBMS_OUTPUT.PUT_LINE('Actualizarea a esuat!');
  END IF;

```

```

-----  

Id medic: 5 Constantinescu Vlad  

Salariu: 7900  

Consultatii:  

Data consultatie: 18-DEC-23  

Pacient:  

Georgescu Ion  

Data consultatie: 18-DEC-23  

Pacient:  

Marinescu Alexandru  

Data consultatie: 19-DEC-23  

Pacient:  

Marinescu Alexandru  

-----  

Id medic: 1 Popescu Andrei  

Salariu: 8000  

Consultatii:  

Data consultatie: 18-NOV-23  

Pacient:  

Popescu Ana  

Data consultatie: 18-DEC-23  

Pacient:  

Popescu Ana  

-----  

Id medic: 3 Dumitrescu Mihai  

Salariu: 8200  

Consultatii:  

Data consultatie: 18-NOV-23  

Pacient:  

Ionescu Mihai  

Data consultatie: 18-DEC-23  

Pacient:  

Ionescu Mihai  

Actualizare cu succes!
-----  


```

File Edit View Navigate Run Source Team Tools Window Help

Oracle SQL Developer : /Users/catalinamacovei/.sqldeveloper/exercitii_medicare.sql

242_Macovei_Catalina_inserari.sql exercitii_medicare.sql Compiler - Log toDo.sql... DOCKER_PLSQL

SQL Worksheet History

Worksheet Query Builder

```
--utilizare pachet
DECLARE
    v_success NUMBER;
BEGIN
    v_success := pachet_actualizari_medicare.marire_salariu_pac();
    IF v_success = 1 THEN
        DBMS_OUTPUT.PUT_LINE('Actualizare cu succes!');
    ELSE
        DBMS_OUTPUT.PUT_LINE('Actualizarea a esuat!');
    END IF;
    DBMS_OUTPUT.PUT_LINE('-----');
END;
/
rollback;
select * from istoric_salariu;
drop package pachet_actualizari_medicare;
COMMIT;
```

Script Output | Query Result

All Rows Fetched: 13 in 0.01 seconds

ID_ISTORIC_SAL	ID_MEDIC	SALARIU_VECI	SALARIU_NOU	DATA_ACTUALIZARE
7	7	1250	7700	08-JAN-18
8	8	1500	8100	09-JAN-18
9	9	1500	7900	10-JAN-18
10	10	1500	7800	11-JAN-18
11	11	7900	8690	12-JAN-24
12	13	8000	8800	12-JAN-24
13	16	8200	9020	12-JAN-24

Saved: /Users/catalinamacovei/.sqldeveloper/exercitii_medicare.sql

Id medic: 5 Constantinescu Vlad
Salariu: 7900
Consultatii:
Data consultatie: 18-DEC-23
Pacient:
Georgescu Ion
Data consultatie: 18-DEC-23
Pacient:
Marinescu Alexandru
Data consultatie: 19-DEC-23
Pacient:
Marinescu Alexandru

Id medic: 1 Popescu Andrei
Salariu: 8000
Consultatii:
Data consultatie: 18-NOV-23
Pacient:
Popescu Ana
Data consultatie: 18-DEC-23
Pacient:
Popescu Ana

Id medic: 3 Dumitrescu Mihai
Salariu: 8200
Consultatii:
Data consultatie: 18-NOV-23
Pacient:
Ionescu Mihai
Data consultatie: 18-DEC-23
Pacient:
Ionescu Mihai

Actualizare cu succes!

CREATE OR REPLACE PACKAGE pachet_actualizari_medicare **AS**

```
TYPE tablou_imbr IS TABLE OF pacienti%ROWTYPE;
TYPE tablou_imbricat IS TABLE OF consultatii%ROWTYPE;
TYPE vector IS VARRAY(3) OF medici%ROWTYPE;
v_istoric_salariu ISTORIC_SALARIU%ROWTYPE;
```

med vector;

CURSOR med_cursor **IS**

```
SELECT ID_MEDIC, NUME, PRENUME, SALARIU, MEDIC_SEF, ID_SECTIE_M
FROM (
    SELECT COUNT(*) AS NR_CONS, M.ID_MEDIC, M.NUME, M.PRENUME, M.SALARIU,
    M.MEDIC_SEF, M.ID_SECTIE_M
    FROM MEDICI M
    INNER JOIN CONSULTATII C ON C.ID_MEDIC = M.ID_MEDIC
    GROUP BY M.ID_MEDIC, M.ID_MEDIC, M.NUME, M.PRENUME, M.SALARIU, M.MEDIC_SEF,
```

```

M.ID_SECTIE_M
    ORDER BY NR_CONS DESC
) T
WHERE ROWNUM <= 3;

PROCEDURE initializare_date;

PROCEDURE afisare_medici;

FUNCTION creste_salariu(p_salariu NUMBER)
RETURN NUMBER;

FUNCTION marire_salariu_pac RETURN NUMBER;

END pachet_actualizari_medicare;
/

CREATE OR REPLACE PACKAGE BODY pachet_actualizari_medicare AS
pacienti_med tablou_imbr := tablou_imbr();
cons_med tablou_imbricat := tablou_imbricat();

PROCEDURE initializare_date IS
BEGIN
pacienti_med.DELETE;
cons_med.DELETE;

OPEN med_cursor;
FETCH med_cursor BULK COLLECT INTO med;
CLOSE med_cursor;

FOR i IN 1..med.COUNT LOOP
-- Pentru pacienti
FOR patient_rec IN (
    SELECT distinct pa.*
    FROM pacienti pa
    INNER JOIN consultatii c ON c.id_patient = pa.id_patient
    WHERE c.id_medic = med(i).id_medic
) LOOP
    pacienti_med.extend;
    pacienti_med(pacienti_med.last) := patient_rec;
END LOOP;

-- Pentru consultatii

```

```

FOR cons_rec IN (
  SELECT *
  FROM consultatii
  WHERE id_medic = med(i).id_medic
) LOOP
  cons_med.extend;
  cons_med(cons_med.last) := cons_rec;
END LOOP;
END LOOP;

END initializare_date;

PROCEDURE afisare_medici
IS
BEGIN
initializare_date();
  FOR k IN 1..med.COUNT LOOP
    DBMS_OUTPUT.PUT_LINE('-----');
    DBMS_OUTPUT.PUT_LINE('Id medic: ' || med(k).id_medic || ' ' || med(k).nume || ' ' ||
med(k).prenume);
    DBMS_OUTPUT.PUT_LINE('Salariu: ' || med(k).salariu);
    DBMS_OUTPUT.PUT_LINE('Consultatii:');

    FOR j IN 1..cons_med.COUNT LOOP
      IF cons_med(j).id_medic = med(k).id_medic THEN
        DBMS_OUTPUT.PUT_LINE('Data consultatie: ' || cons_med(j).data_cons || ' ');
        DBMS_OUTPUT.PUT_LINE('Pacient: ');

        FOR i IN 1..pacienti_med.COUNT LOOP
          IF cons_med(j).id_pacient = pacienti_med(i).id_pacient THEN
            DBMS_OUTPUT.PUT_LINE(pacienti_med(i).nume || ' ' || pacienti_med(i).prenume);
          END IF;
        END LOOP;
      END IF;
    END LOOP;
  END IF;
END LOOP;

DBMS_OUTPUT.NEW_LINE;
END LOOP;
END afisare_medici;

FUNCTION creste_salariu(p_salariu NUMBER)
RETURN NUMBER IS
  salariu_crescut NUMBER;
BEGIN

```

```

salariu_crescut := p_salariu * 1.1; -- Crestere cu 10%
RETURN salariu_crescut;
END;

FUNCTION marire_salariu_pac

RETURN NUMBER IS
  v_result NUMBER := -2;
  id_nou NUMBER;
BEGIN
  initializare_date();
  afisare_medici();

FOR i IN 1..med.COUNT LOOP

  SELECT MAX(ID_ISTORIC_SAL) INTO id_nou FROM ISTORIC_SALARIU;

  v_istoric_salariu.ID_ISTORIC_SAL := id_nou + i;
  v_istoric_salariu.ID_MEDIC := med(i).id_medic;
  v_istoric_salariu.SALARIU_NOU := creste_salariu(med(i).salariu);
  v_istoric_salariu.SALARIU_VECHI := med(i).salariu;
  v_istoric_salariu.DATA_ACTUALIZARE := SYSDATE;

  UPDATE MEDICI
  SET salariu = creste_salariu(salariu)
  WHERE id_medic = med(i).id_medic;

  INSERT INTO ISTORIC_SALARIU (ID_ISTORIC_SAL, ID_MEDIC, SALARIU_VECHI,
SALARIU_NOU, DATA_ACTUALIZARE)
  VALUES (
    v_istoric_salariu.ID_ISTORIC_SAL,
    v_istoric_salariu.ID_MEDIC,
    v_istoric_salariu.SALARIU_VECHI,
    v_istoric_salariu.SALARIU_NOU,
    v_istoric_salariu.DATA_ACTUALIZARE
  );

  v_result := 1; -- REZULTAT SUCCES
END LOOP;

RETURN v_result;

END marire_salariu_pac;

```

```
END pachet_actualizari_medicare;
/
--utilizare pachet
DECLARE
    v_success NUMBER;
BEGIN
    v_success := pachet_actualizari_medicare.marire_salariu_pac();

    IF v_success = 1 THEN
        DBMS_OUTPUT.PUT_LINE('Actualizare cu succes!');
    ELSE
        DBMS_OUTPUT.PUT_LINE('Actualizarea a esuat!');
    END IF;

    DBMS_OUTPUT.PUT_LINE('-----');

END;
/
select * from istoric_salariu;
```