Proiectarea și Gestionarea unei Baze de Date

Clinică privată

Versiune: Oracle Database 19c

Cuprins

[Introducere: 3](#_Toc187347514)

[Cerinta 1: 5](#_Toc187347515)

[Cerinta 2: 6](#_Toc187347516)

[Cerinta 3: 7](#_Toc187347517)

[Cerinta 4: 8](#_Toc187347518)

[Cerinta 5: 11](#_Toc187347519)

[Cerinta 6: 15](#_Toc187347520)

[Cerinta 7: 19](#_Toc187347521)

[Cerinta 8: 22](#_Toc187347522)

[Cerinta 9: 25](#_Toc187347523)

[Cerinta 10: 30](#_Toc187347524)

[Cerinta 11: 32](#_Toc187347525)

[Cerinta 12: 37](#_Toc187347526)

[Cerinta 13: 39](#_Toc187347527)

## Cerinta 1:

Prezentați pe scurt baza de date (utilitatea ei).

Modelul de date va gestiona informatiile legate de activitatea unei clinici

private. Clinica are mai multi angajati, printre care asistente si doctori.

Fiecare asistenta se ocupa de mai multe camere, respectiv de paturile din

acele camere in care pot fi internati pacientii.

Pacientii isi pot programa o consultatie cu un anumit doctor.

Doctorii prescriu tratamente pentru pacienti.

Pacientii, care pot suferi de mai multe boli, vor primi cate un tratament

pentru fiecare boala de care sufera (consideram in plus ca un tratament poate

fi acordat mai multor pacienti odata ce s-a dovedit eficient si ca unele tratamente pot fi

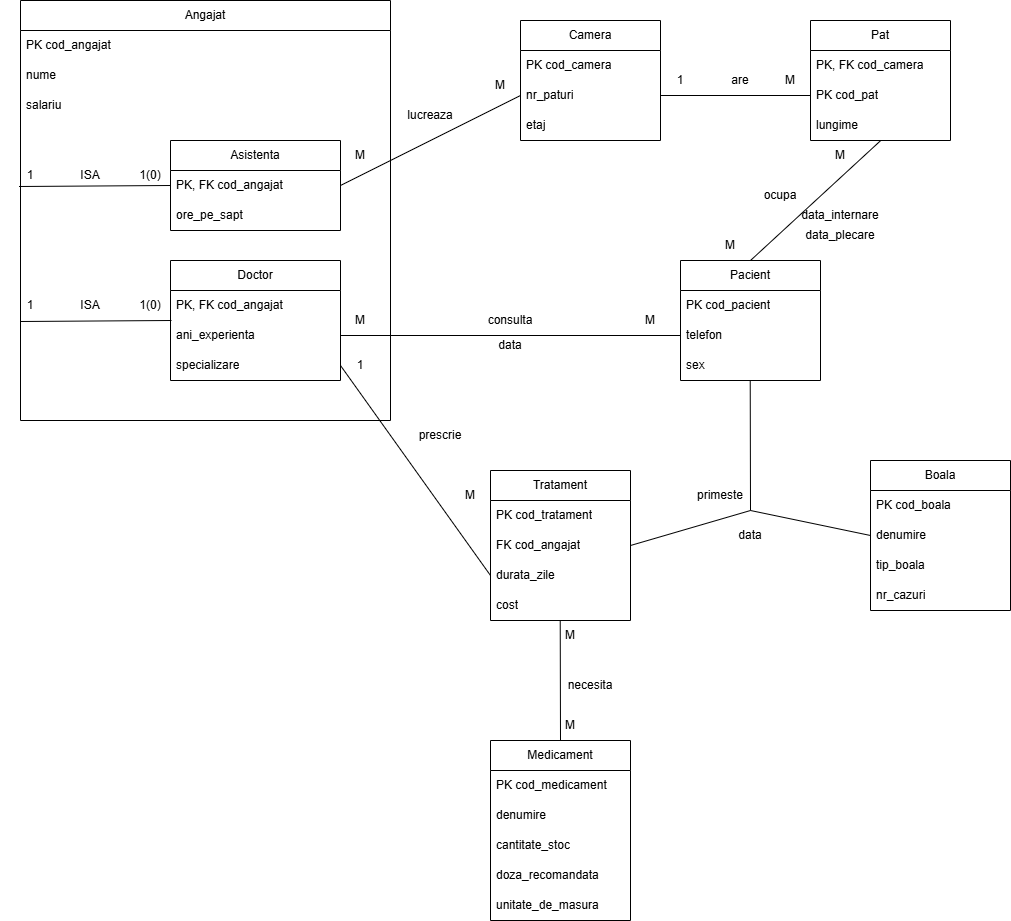
folosite pentru combaterea mai multor boli).

Se vor pastra informatii despre numarul de pacienti care au fost tratati

pentru o anumita boala, cat si stocul clinicii de medicamente necesare pentru

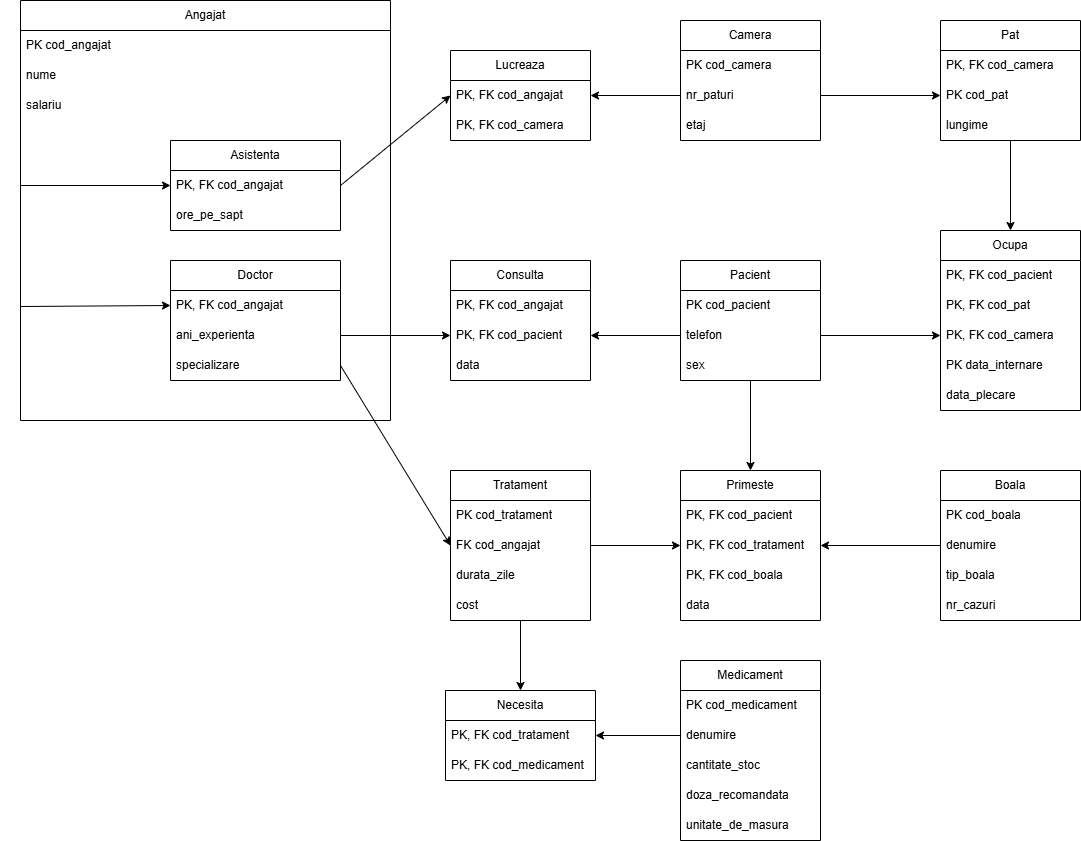
tratarea bolilor.

## Cerinta 2:

Diagrama ER:

## Cerinta 3:

Diagrama conceptuala:



## Cerinta 4:

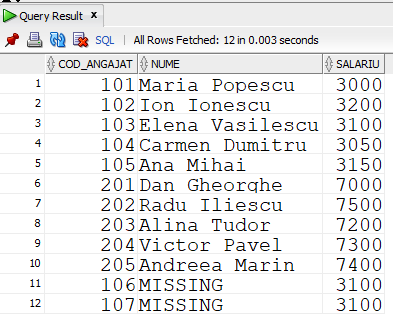
Creare tabel (+ constrangeri):

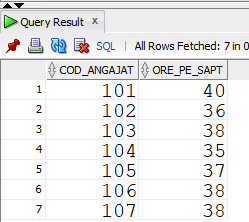
|  |
| --- |
| CREATE TABLE angajat(  cod\_angajat NUMBER(4) PRIMARY KEY,  nume VARCHAR2(50) NOT NULL,  salariu NUMBER(6),  CONSTRAINT salariu\_pozitiv CHECK (salariu > 0));    CREATE TABLE asistenta(  cod\_angajat NUMBER(4) PRIMARY KEY REFERENCES angajat(cod\_angajat) ON DELETE CASCADE,  ore\_pe\_sapt NUMBER(3));    CREATE TABLE doctor(  cod\_angajat NUMBER(4) PRIMARY KEY REFERENCES angajat(cod\_angajat) ON DELETE CASCADE,  ani\_experieta NUMBER(3),  specializare VARCHAR2(50));    CREATE TABLE camera(  cod\_camera NUMBER(4) PRIMARY KEY,  nr\_paturi NUMBER(2),  etaj NUMBER(2));    CREATE TABLE lucreaza(  cod\_angajat NUMBER(4),  cod\_camera NUMBER(4),  CONSTRAINT lucreaza\_pk PRIMARY KEY (cod\_angajat, cod\_camera),  CONSTRAINT fk\_cod\_angajat FOREIGN KEY (cod\_angajat) REFERENCES asistenta(cod\_angajat) ON DELETE CASCADE,  CONSTRAINT fk\_cod\_camera FOREIGN KEY (cod\_camera) REFERENCES camera(cod\_camera) ON DELETE CASCADE);    CREATE TABLE pat(  cod\_pat NUMBER(4) PRIMARY KEY,  cod\_camera NUMBER(4),  lungime NUMBER(4, 2),  PRIMARY KEY (cod\_pat, cod\_camera),  FOREIGN KEY (cod\_camera) REFERENCES camera(cod\_camera) ON DELETE CASCADE);    CREATE TABLE pacient(  cod\_pacient NUMBER(4) PRIMARY KEY,  telefon VARCHAR2(10) NOT NULL,  sex CHAR(1),  CONSTRAINT verif\_sex CHECK (sex IN ('M', 'F')));    CREATE TABLE ocupa(  cod\_pacient NUMBER(4),  cod\_pat NUMBER(4),  cod\_camera NUMBER(4),  data\_internare DATE NOT NULL,  data\_plecare DATE NOT NULL,  PRIMARY KEY (cod\_pacient, cod\_camera, cod\_pat),  FOREIGN KEY (cod\_pacient) REFERENCES pacient(cod\_pacient) ON DELETE CASCADE,  FOREIGN KEY (cod\_pat, cod\_camera) REFERENCES pat(cod\_pat, cod\_camera) ON DELETE CASCADE);  -- am schimbat niste detalii  ALTER TABLE ocupa MODIFY data\_plecare NULL;  ALTER TABLE ocupa DROP PRIMARY KEY;  ALTER TABLE ocupa  ADD CONSTRAINT pk\_ocupa PRIMARY KEY (cod\_pacient, cod\_pat, cod\_camera, data\_internare);    CREATE TABLE consulta(  cod\_angajat NUMBER(4),  cod\_pacient NUMBER(4),  data DATE DEFAULT SYSDATE,  PRIMARY KEY (cod\_angajat, cod\_pacient),  FOREIGN KEY (cod\_angajat) REFERENCES doctor(cod\_angajat) ON DELETE CASCADE,  FOREIGN KEY (cod\_pacient) REFERENCES pacient(cod\_pacient) ON DELETE CASCADE);    CREATE TABLE tratament(  cod\_tratament NUMBER(4) PRIMARY KEY,  cod\_angajat NUMBER(4),  durata\_zile NUMBER(5),  cost NUMBER(5,2),  FOREIGN KEY (cod\_angajat) REFERENCES doctor(cod\_angajat));    CREATE TABLE medicament(  cod\_medicament NUMBER(4) PRIMARY KEY,  denumire VARCHAR2(30),  cantitate\_stoc INT DEFAULT 0,  doza\_recomandata NUMBER(10, 2),  unitate\_de\_masura VARCHAR2(10) NOT NULL);    CREATE TABLE necesita(  cod\_tratament NUMBER(4),  cod\_medicament NUMBER(4),  PRIMARY KEY (cod\_tratament, cod\_medicament),  FOREIGN KEY (cod\_tratament) REFERENCES tratament(cod\_tratament) ON DELETE CASCADE,  FOREIGN KEY (cod\_medicament) REFERENCES medicament(cod\_medicament) ON DELETE CASCADE);    CREATE TABLE boala(  cod\_boala NUMBER(4) PRIMARY KEY,  denumire VARCHAR2(50) UNIQUE,  tip\_boala VARCHAR2(30),  nr\_cazuri INT);    CREATE TABLE primeste(  cod\_pacient NUMBER(4),  cod\_tratament NUMBER(4),  cod\_boala NUMBER(4),  data DATE default SYSDATE,  PRIMARY KEY (cod\_pacient, cod\_tratament, cod\_boala),  FOREIGN KEY (cod\_pacient) REFERENCES pacient(cod\_pacient) ON DELETE CASCADE,  FOREIGN KEY (cod\_tratament) REFERENCES tratament(cod\_tratament) ON DELETE CASCADE,  FOREIGN KEY (cod\_boala) REFERENCES boala(cod\_boala) ON DELETE CASCADE); |

## Cerinta 5:

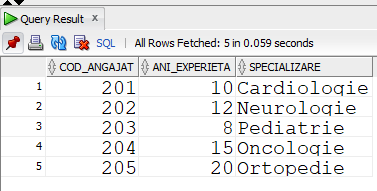
Adaugare informatii (unele intrari care apar in screen shot-uri au fost introduse/schimbate cand testam cerintele):

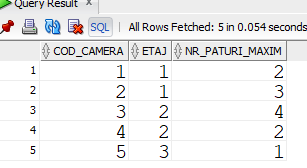
|  |
| --- |
| -- Inseram in tabelul 'angajat' pentru asistente  INSERT INTO angajat (cod\_angajat, nume, salariu) VALUES (101, 'Maria Popescu', 3000);  INSERT INTO angajat (cod\_angajat, nume, salariu) VALUES (102, 'Ion Ionescu', 3200);  INSERT INTO angajat (cod\_angajat, nume, salariu) VALUES (103, 'Elena Vasilescu', 3100);  INSERT INTO angajat (cod\_angajat, nume, salariu) VALUES (104, 'Carmen Dumitru', 3050);  INSERT INTO angajat (cod\_angajat, nume, salariu) VALUES (105, 'Ana Mihai', 3150);  -- Inseram in tabelul 'asistenta'  INSERT INTO asistenta (cod\_angajat, ore\_pe\_sapt) VALUES (101, 40);  INSERT INTO asistenta (cod\_angajat, ore\_pe\_sapt) VALUES (102, 36);  INSERT INTO asistenta (cod\_angajat, ore\_pe\_sapt) VALUES (103, 38);  INSERT INTO asistenta (cod\_angajat, ore\_pe\_sapt) VALUES (104, 35);  INSERT INTO asistenta (cod\_angajat, ore\_pe\_sapt) VALUES (105, 37);  -- Inseram in tabelul 'angajat' pentru doctori  INSERT INTO angajat (cod\_angajat, nume, salariu) VALUES (201, 'Dan Gheorghe', 7000);  INSERT INTO angajat (cod\_angajat, nume, salariu) VALUES (202, 'Radu Iliescu', 7500);  INSERT INTO angajat (cod\_angajat, nume, salariu) VALUES (203, 'Alina Tudor', 7200);  INSERT INTO angajat (cod\_angajat, nume, salariu) VALUES (204, 'Victor Pavel', 7300);  INSERT INTO angajat (cod\_angajat, nume, salariu) VALUES (205, 'Andreea Marin', 7400);  -- Inseram in tabelul 'doctor'  INSERT INTO doctor (cod\_angajat, ani\_experieta, specializare) VALUES (201, 10, 'Cardiologie');  INSERT INTO doctor (cod\_angajat, ani\_experieta, specializare) VALUES (202, 12, 'Neurologie');  INSERT INTO doctor (cod\_angajat, ani\_experieta, specializare) VALUES (203, 8, 'Pediatrie');  INSERT INTO doctor (cod\_angajat, ani\_experieta, specializare) VALUES (204, 15, 'Oncologie');  INSERT INTO doctor (cod\_angajat, ani\_experieta, specializare) VALUES (205, 20, 'Ortopedie');  -- Inseram in tabelul 'camera'  INSERT INTO camera (cod\_camera, nr\_paturi, etaj) VALUES (1, 2, 1);  INSERT INTO camera (cod\_camera, nr\_paturi, etaj) VALUES (2, 3, 1);  INSERT INTO camera (cod\_camera, nr\_paturi, etaj) VALUES (3, 4, 2);  INSERT INTO camera (cod\_camera, nr\_paturi, etaj) VALUES (4, 2, 2);  INSERT INTO camera (cod\_camera, nr\_paturi, etaj) VALUES (5, 1, 3);  -- Inseram in tabelul 'lucreaza'  INSERT INTO lucreaza (cod\_angajat, cod\_camera) VALUES (101, 1);  INSERT INTO lucreaza (cod\_angajat, cod\_camera) VALUES (101, 2);  INSERT INTO lucreaza (cod\_angajat, cod\_camera) VALUES (102, 2);  INSERT INTO lucreaza (cod\_angajat, cod\_camera) VALUES (102, 3);  INSERT INTO lucreaza (cod\_angajat, cod\_camera) VALUES (103, 3);  INSERT INTO lucreaza (cod\_angajat, cod\_camera) VALUES (103, 4);  INSERT INTO lucreaza (cod\_angajat, cod\_camera) VALUES (104, 4);  INSERT INTO lucreaza (cod\_angajat, cod\_camera) VALUES (104, 5);  INSERT INTO lucreaza (cod\_angajat, cod\_camera) VALUES (105, 1);  INSERT INTO lucreaza (cod\_angajat, cod\_camera) VALUES (105, 5);  -- Inseram in tabelul 'pat'  INSERT INTO pat (cod\_pat, cod\_camera, lungime) VALUES (101, 1, 2.00);  INSERT INTO pat (cod\_pat, cod\_camera, lungime) VALUES (102, 1, 1.95);  INSERT INTO pat (cod\_pat, cod\_camera, lungime) VALUES (103, 2, 2.10);  INSERT INTO pat (cod\_pat, cod\_camera, lungime) VALUES (104, 2, 1.85);  INSERT INTO pat (cod\_pat, cod\_camera, lungime) VALUES (105, 2, 2.00);  INSERT INTO pat (cod\_pat, cod\_camera, lungime) VALUES (106, 3, 2.00);  INSERT INTO pat (cod\_pat, cod\_camera, lungime) VALUES (107, 3, 2.10);  INSERT INTO pat (cod\_pat, cod\_camera, lungime) VALUES (108, 3, 1.90);  INSERT INTO pat (cod\_pat, cod\_camera, lungime) VALUES (109, 3, 2.05);  INSERT INTO pat (cod\_pat, cod\_camera, lungime) VALUES (110, 4, 1.80);  INSERT INTO pat (cod\_pat, cod\_camera, lungime) VALUES (111, 4, 1.95);  INSERT INTO pat (cod\_pat, cod\_camera, lungime) VALUES (112, 5, 2.20);  -- Inseram in tabelul 'pacient'  INSERT INTO pacient (cod\_pacient, telefon, sex) VALUES (201, '0712345678', 'M');  INSERT INTO pacient (cod\_pacient, telefon, sex) VALUES (202, '0723456789', 'F');  INSERT INTO pacient (cod\_pacient, telefon, sex) VALUES (203, '0734567890', 'M');  INSERT INTO pacient (cod\_pacient, telefon, sex) VALUES (204, '0745678901', 'F');  INSERT INTO pacient (cod\_pacient, telefon, sex) VALUES (205, '0756789012', 'M');  -- Inseram in tabelul 'ocupa'  INSERT INTO ocupa (cod\_pacient, cod\_pat, cod\_camera, data\_internare, data\_plecare) VALUES (201, 101, 1, TO\_DATE('2024-01-01', 'YYYY-MM-DD'), TO\_DATE('2024-01-10', 'YYYY-MM-DD'));  INSERT INTO ocupa (cod\_pacient, cod\_pat, cod\_camera, data\_internare, data\_plecare) VALUES (201, 102, 1, TO\_DATE('2024-01-11', 'YYYY-MM-DD'), TO\_DATE('2024-01-20', 'YYYY-MM-DD'));  INSERT INTO ocupa (cod\_pacient, cod\_pat, cod\_camera, data\_internare, data\_plecare) VALUES (202, 103, 2, TO\_DATE('2024-01-05', 'YYYY-MM-DD'), TO\_DATE('2024-01-15', 'YYYY-MM-DD'));  INSERT INTO ocupa (cod\_pacient, cod\_pat, cod\_camera, data\_internare, data\_plecare) VALUES (202, 104, 2, TO\_DATE('2024-01-16', 'YYYY-MM-DD'), TO\_DATE('2024-01-25', 'YYYY-MM-DD'));  INSERT INTO ocupa (cod\_pacient, cod\_pat, cod\_camera, data\_internare, data\_plecare) VALUES (202, 105, 2, TO\_DATE('2024-01-26', 'YYYY-MM-DD'), TO\_DATE('2024-02-05', 'YYYY-MM-DD'));  INSERT INTO ocupa (cod\_pacient, cod\_pat, cod\_camera, data\_internare, data\_plecare) VALUES (203, 106, 3, TO\_DATE('2024-01-10', 'YYYY-MM-DD'), TO\_DATE('2024-01-20', 'YYYY-MM-DD'));  INSERT INTO ocupa (cod\_pacient, cod\_pat, cod\_camera, data\_internare, data\_plecare) VALUES (204, 110, 4, TO\_DATE('2024-01-15', 'YYYY-MM-DD'), TO\_DATE('2024-01-25', 'YYYY-MM-DD'));  INSERT INTO ocupa (cod\_pacient, cod\_pat, cod\_camera, data\_internare, data\_plecare) VALUES (204, 111, 4, TO\_DATE('2024-01-26', 'YYYY-MM-DD'), TO\_DATE('2024-02-05', 'YYYY-MM-DD'));  INSERT INTO ocupa (cod\_pacient, cod\_pat, cod\_camera, data\_internare, data\_plecare) VALUES (205, 112, 5, TO\_DATE('2024-01-01', 'YYYY-MM-DD'), TO\_DATE('2024-01-10', 'YYYY-MM-DD'));  INSERT INTO ocupa (cod\_pacient, cod\_pat, cod\_camera, data\_internare, data\_plecare) VALUES (205, 102, 1, TO\_DATE('2024-01-11', 'YYYY-MM-DD'), TO\_DATE('2024-01-20', 'YYYY-MM-DD'));  -- Inseram in tabelul 'consulta'  INSERT INTO consulta (cod\_angajat, cod\_pacient, data) VALUES (201, 201, TO\_DATE('2024-01-01', 'YYYY-MM-DD'));  INSERT INTO consulta (cod\_angajat, cod\_pacient, data) VALUES (201, 202, TO\_DATE('2024-01-02', 'YYYY-MM-DD'));  INSERT INTO consulta (cod\_angajat, cod\_pacient, data) VALUES (201, 203, TO\_DATE('2024-01-03', 'YYYY-MM-DD'));  INSERT INTO consulta (cod\_angajat, cod\_pacient, data) VALUES (202, 201, TO\_DATE('2024-01-04', 'YYYY-MM-DD'));  INSERT INTO consulta (cod\_angajat, cod\_pacient, data) VALUES (202, 204, TO\_DATE('2024-01-05', 'YYYY-MM-DD'));  INSERT INTO consulta (cod\_angajat, cod\_pacient, data) VALUES (203, 205, TO\_DATE('2024-01-06', 'YYYY-MM-DD'));  INSERT INTO consulta (cod\_angajat, cod\_pacient, data) VALUES (204, 202, TO\_DATE('2024-01-07', 'YYYY-MM-DD'));  INSERT INTO consulta (cod\_angajat, cod\_pacient, data) VALUES (204, 203, TO\_DATE('2024-01-08', 'YYYY-MM-DD'));  INSERT INTO consulta (cod\_angajat, cod\_pacient, data) VALUES (205, 204, TO\_DATE('2024-01-09', 'YYYY-MM-DD'));  INSERT INTO consulta (cod\_angajat, cod\_pacient, data) VALUES (205, 205, TO\_DATE('2024-01-10', 'YYYY-MM-DD'));  -- Inseram in tabelul 'tratament'  INSERT INTO tratament (cod\_tratament, cod\_angajat, durata\_zile, cost) VALUES (401, 201, 10, 150.00);  INSERT INTO tratament (cod\_tratament, cod\_angajat, durata\_zile, cost) VALUES (402, 202, 15, 200.00);  INSERT INTO tratament (cod\_tratament, cod\_angajat, durata\_zile, cost) VALUES (403, 203, 7, 800.50);  INSERT INTO tratament (cod\_tratament, cod\_angajat, durata\_zile, cost) VALUES (404, 204, 20, 900.75);  INSERT INTO tratament (cod\_tratament, cod\_angajat, durata\_zile, cost) VALUES (405, 205, 5, 500.00);  -- Inseram in tabelul 'medicament'  INSERT INTO medicament (cod\_medicament, denumire, cantitate\_stoc, doza\_recomandata, unitate\_de\_masura) VALUES (501, 'Paracetamol', 500, 2.00, 'mg');  INSERT INTO medicament (cod\_medicament, denumire, cantitate\_stoc, doza\_recomandata, unitate\_de\_masura) VALUES (502, 'Ibuprofen', 300, 1.50, 'mg');  INSERT INTO medicament (cod\_medicament, denumire, cantitate\_stoc, doza\_recomandata, unitate\_de\_masura) VALUES (503, 'Amoxicilina', 200, 3.00, 'capsule');  INSERT INTO medicament (cod\_medicament, denumire, cantitate\_stoc, doza\_recomandata, unitate\_de\_masura) VALUES (504, 'Vitamina C', 1000, 1.00, 'g');  INSERT INTO medicament (cod\_medicament, denumire, cantitate\_stoc, doza\_recomandata, unitate\_de\_masura) VALUES (505, 'Omeprazol', 150, 0.50, 'capsule');  -- Inseram in tabelul 'necesita'  INSERT INTO necesita (cod\_tratament, cod\_medicament) VALUES (401, 501);  INSERT INTO necesita (cod\_tratament, cod\_medicament) VALUES (401, 502);  INSERT INTO necesita (cod\_tratament, cod\_medicament) VALUES (402, 503);  INSERT INTO necesita (cod\_tratament, cod\_medicament) VALUES (402, 504);  INSERT INTO necesita (cod\_tratament, cod\_medicament) VALUES (402, 505);  INSERT INTO necesita (cod\_tratament, cod\_medicament) VALUES (403, 501);  INSERT INTO necesita (cod\_tratament, cod\_medicament) VALUES (404, 502);  INSERT INTO necesita (cod\_tratament, cod\_medicament) VALUES (404, 503);  INSERT INTO necesita (cod\_tratament, cod\_medicament) VALUES (405, 504);  INSERT INTO necesita (cod\_tratament, cod\_medicament) VALUES (405, 505);  -- Inseram in tabelul 'boala'  INSERT INTO boala (cod\_boala, denumire, tip\_boala, nr\_cazuri) VALUES (601, 'Gripa', 'Virala', 250);  INSERT INTO boala (cod\_boala, denumire, tip\_boala, nr\_cazuri) VALUES (602, 'Pneumonie', 'Bacteriala', 180);  INSERT INTO boala (cod\_boala, denumire, tip\_boala, nr\_cazuri) VALUES (603, 'Diabet', 'Cronica', 300);  INSERT INTO boala (cod\_boala, denumire, tip\_boala, nr\_cazuri) VALUES (604, 'Cancer pulmonar', 'Neoplazica', 120);  INSERT INTO boala (cod\_boala, denumire, tip\_boala, nr\_cazuri) VALUES (605, 'Hipertensiune arteriala', 'Cardiovasculara', 400);  -- Inseram in tabelul 'primeste'  INSERT INTO primeste (cod\_pacient, cod\_tratament, cod\_boala, data) VALUES (201, 401, 601, TO\_DATE('2024-01-01', 'YYYY-MM-DD'));  INSERT INTO primeste (cod\_pacient, cod\_tratament, cod\_boala, data) VALUES (201, 402, 601, TO\_DATE('2024-01-05', 'YYYY-MM-DD'));  INSERT INTO primeste (cod\_pacient, cod\_tratament, cod\_boala, data) VALUES (202, 403, 602, TO\_DATE('2024-01-10', 'YYYY-MM-DD'));  INSERT INTO primeste (cod\_pacient, cod\_tratament, cod\_boala, data) VALUES (202, 404, 602, TO\_DATE('2024-01-12', 'YYYY-MM-DD'));  INSERT INTO primeste (cod\_pacient, cod\_tratament, cod\_boala, data) VALUES (202, 405, 602, TO\_DATE('2024-01-15', 'YYYY-MM-DD'));  INSERT INTO primeste (cod\_pacient, cod\_tratament, cod\_boala, data) VALUES (203, 401, 603, TO\_DATE('2024-01-20', 'YYYY-MM-DD'));  INSERT INTO primeste (cod\_pacient, cod\_tratament, cod\_boala, data) VALUES (204, 402, 604, TO\_DATE('2024-01-22', 'YYYY-MM-DD'));  INSERT INTO primeste (cod\_pacient, cod\_tratament, cod\_boala, data) VALUES (204, 403, 604, TO\_DATE('2024-01-25', 'YYYY-MM-DD'));  INSERT INTO primeste (cod\_pacient, cod\_tratament, cod\_boala, data) VALUES (205, 404, 605, TO\_DATE('2024-01-30', 'YYYY-MM-DD'));  INSERT INTO primeste (cod\_pacient, cod\_tratament, cod\_boala, data) VALUES (205, 405, 601, TO\_DATE('2024-02-01', 'YYYY-MM-DD'));  INSERT INTO primeste (cod\_pacient, cod\_tratament, cod\_boala, data) VALUES (205, 401, 603, TO\_DATE('2024-02-05', 'YYYY-MM-DD')); |

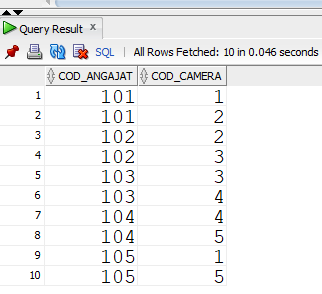


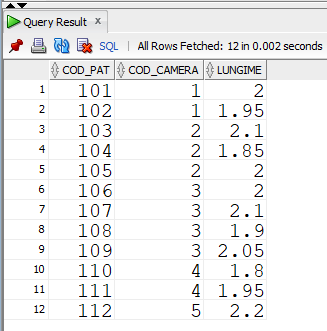


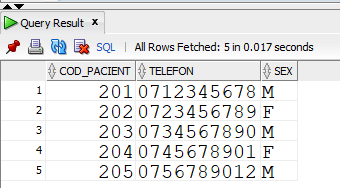


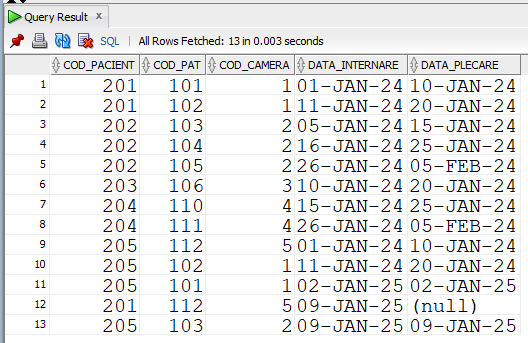


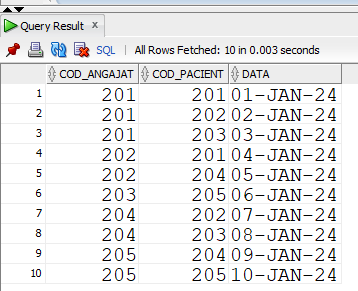


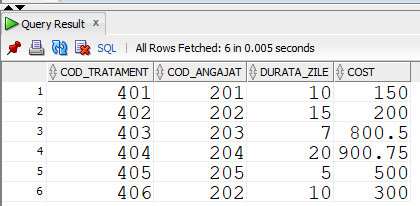


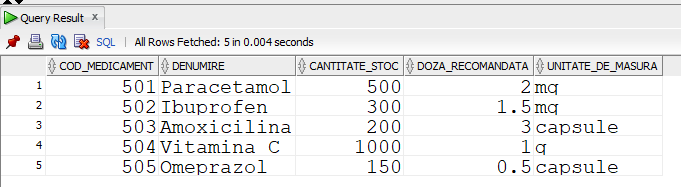


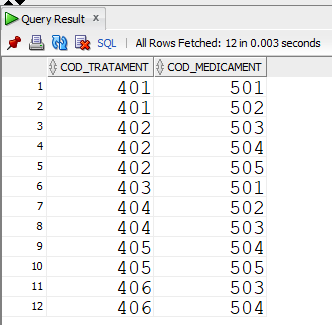


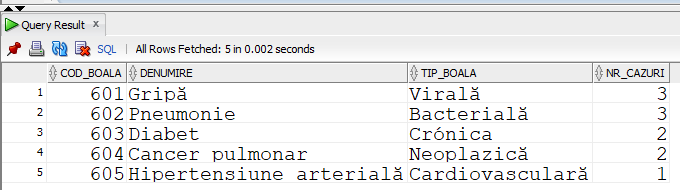


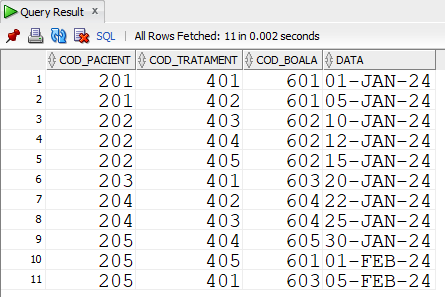












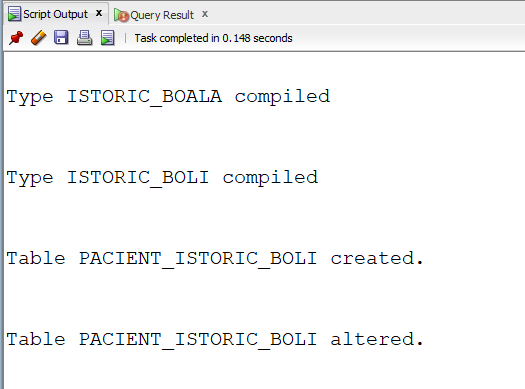
## Cerinta 6:

Sa se creeze un tabel pacient\_istoric\_boli in care sa se retina toate bolile unui paicent si data

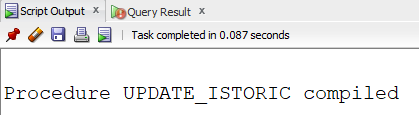
la care a fost diagnosticat cu aceasta (10 cele mai recente pentru fiecare boala). Sa se creeze

o procedura care primeste numele unui pacient si le update-aza istoricul.

|  |
| --- |
| CREATE OR REPLACE TYPE istoric\_boala IS OBJECT (d DATE, denumire VARCHAR2(50));  /  CREATE OR REPLACE TYPE istoric\_boli IS TABLE OF istoric\_boala;  /  CREATE TABLE pacient\_istoric\_boli AS  SELECT cod\_pacient FROM pacient;  ALTER TABLE pacient\_istoric\_boli  ADD (istoric istoric\_boli) NESTED TABLE istoric STORE AS boli\_tab; |

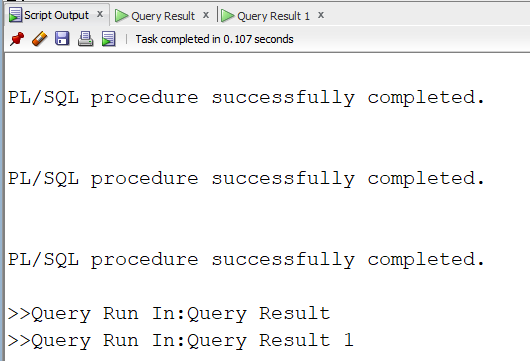


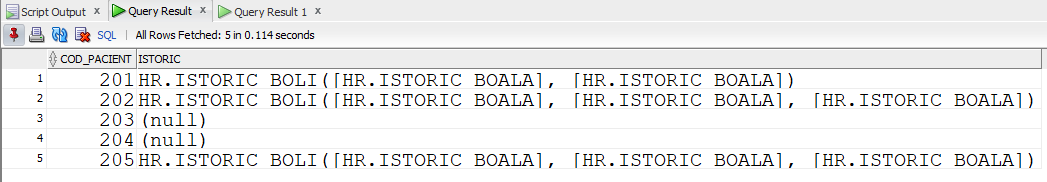
|  |
| --- |
| CREATE OR REPLACE PROCEDURE update\_istoric (cod pacient.cod\_pacient%TYPE) IS  TYPE tab\_imb IS TABLE OF boala.cod\_boala%TYPE;  TYPE tab\_ind IS TABLE OF boala.denumire%TYPE INDEX BY PLS\_INTEGER;  TYPE vec IS VARRAY(10) OF primeste.data%TYPE;  boli tab\_imb := tab\_imb();  boala\_denumire tab\_ind;  date\_boli vec := vec();  denumire boala.denumire%TYPE;  cheie boala.cod\_boala%TYPE;  istoric\_nou istoric\_boli := istoric\_boli();  BEGIN  -- colectez codurile bolilor pacientului  SELECT cod\_boala  BULK COLLECT INTO boli  FROM primeste  WHERE cod\_pacient = cod;    -- salvez in 'boala\_denumire' perechi de forma -> cod\_boala : denumire  FOR i IN boli.FIRST..boli.LAST LOOP  SELECT denumire  INTO denumire  FROM boala  WHERE cod\_boala = boli(i);    boala\_denumire(boli(i)) := denumire;  END LOOP;    -- iterez prin toate cheile tabelului 'boala\_denumire'  cheie := boala\_denumire.FIRST;    WHILE cheie IS NOT NULL LOOP  -- salvez in vectorul 'date' ultimelele 10 date pentru pacientul si boala respectiva  SELECT \*  BULK COLLECT INTO date\_boli  FROM (  SELECT data  FROM primeste  WHERE cod\_pacient = cod  AND cod\_boala = cheie  ORDER BY data DESC  )  WHERE ROWNUM <= 10;  -- adaug intrarile gasite in noul istoric  FOR i IN date\_boli.FIRST..date\_boli.LAST LOOP  istoric\_nou.EXTEND;  istoric\_nou(istoric\_nou.LAST) := istoric\_boala(date\_boli(i), boala\_denumire(cheie));  END LOOP;  cheie := boala\_denumire.NEXT(cheie);  END LOOP;    -- update pe coloana  UPDATE pacient\_istoric\_boli  SET istoric = istoric\_nou  WHERE cod\_pacient = cod;    EXCEPTION  WHEN TOO\_MANY\_ROWS THEN  RAISE\_APPLICATION\_ERROR(20001, 'Nu exista angajatul cu codul ' || cod);  WHEN OTHERS THEN  RAISE\_APPLICATION\_ERROR(20000, 'Nu s-a putut updata istoricul pacientului ' || cod);  END;  / |

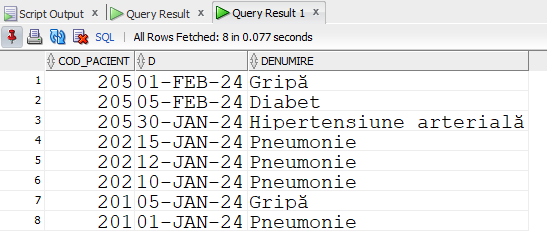


Testez:

|  |
| --- |
| EXECUTE update\_istoric(201);  EXECUTE update\_istoric(202);  EXECUTE update\_istoric(205);  select \* from pacient\_istoric\_boli;  SELECT p.cod\_pacient, t.\*  FROM pacient\_istoric\_boli p, TABLE (p.istoric) t; |



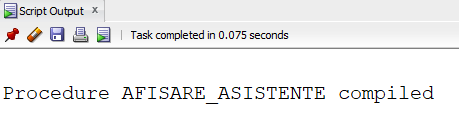




## Cerinta 7:

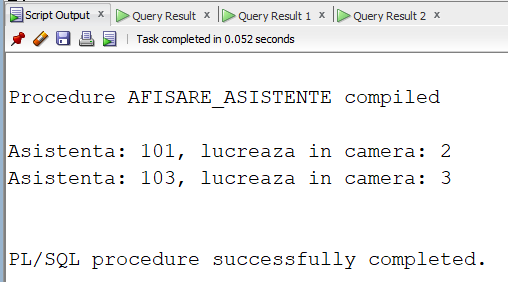
Sa se creeze o procedura care sa afiseze toate asistentele care lucreaza in cel putin o camera cu mai mult de x paturi si cel putin y ore pe saptamana.

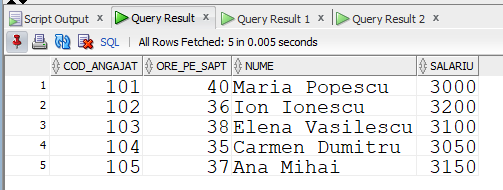
|  |
| --- |
| CREATE OR REPLACE PROCEDURE afisare\_asistente(nr\_paturi NUMBER, ore\_minime asistenta.ore\_pe\_sapt%TYPE) IS  TYPE tab\_imb IS TABLE OF asistenta.cod\_angajat%TYPE;  asistente\_afisate tab\_imb := tab\_imb();  ang asistenta.cod\_angajat%TYPE;  -- cursor cu parametru care intoarce asistentele care lucreaza intr-o anumita camera si au mai mult de y ore pe sapt  CURSOR c\_asistente(cod camera.cod\_camera%TYPE) IS  SELECT cod\_angajat  FROM asistenta a JOIN lucreaza l USING(cod\_angajat)  WHERE a.ore\_pe\_sapt >= ore\_minime  AND l.cod\_camera = cod;  BEGIN  -- cicilu cursor cu subcerere care intoarce camerele cu mai mult de x paturi  FOR i IN (SELECT cod\_camera  FROM camera c JOIN pat p USING(cod\_camera)  GROUP BY cod\_camera  HAVING COUNT(\*) > nr\_paturi)  LOOP  OPEN c\_asistente(i.cod\_camera);  LOOP  FETCH c\_asistente INTO ang;  EXIT WHEN c\_asistente%NOTFOUND;  -- verific daca am afisat deja asistenta  IF ang MEMBER OF asistente\_afisate THEN  CONTINUE;  END IF;  DBMS\_OUTPUT.PUT\_LINE('Asistenta: ' || ang || ', lucreaza in camera: ' || i.cod\_camera);  -- adaug asistenta in lista pentru a nu mai fi luata in considerare ulterior  asistente\_afisate.EXTEND;  asistente\_afisate(asistente\_afisate.LAST) := ang;  END LOOP;  CLOSE c\_asistente;  END LOOP;  END;  / |

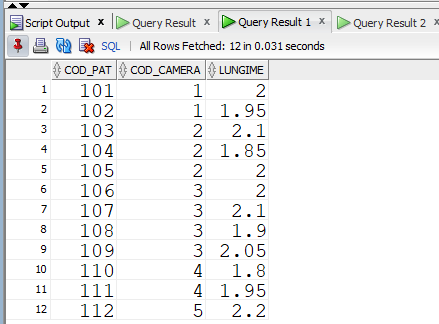


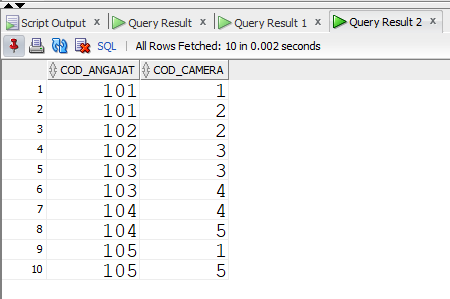
Testez:

|  |
| --- |
| select \* from asistenta join angajat using (cod\_angajat);  select \* from pat;  select \* from lucreaza;  EXECUTE afisare\_asistente(2, 37); |







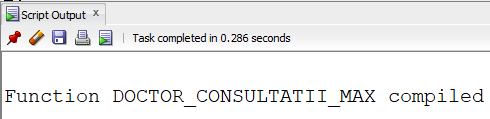


## Cerinta 8:

Sa se creeze o functie care returneaza numele doctorului care a consultat cei mai multi

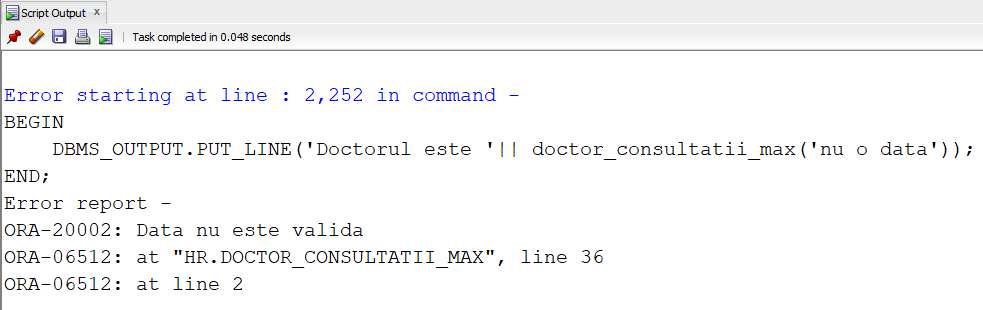
pacienti de sex masculin incepand cu o data primita ca parametru.

|  |
| --- |
| CREATE OR REPLACE FUNCTION doctor\_consultatii\_max(data\_inceput VARCHAR2) RETURN angajat.nume%TYPE IS  nume angajat.nume%TYPE;  data\_validata consulta.data%TYPE;  BEGIN  -- validez mai intai data  BEGIN  data\_validata := TO\_DATE(data\_inceput, 'DD-MON-YY');  EXCEPTION  WHEN OTHERS THEN  RAISE\_APPLICATION\_ERROR(-20002, 'Data nu este valida');  END;  SELECT a.nume  INTO nume  FROM angajat a JOIN doctor d USING(cod\_angajat)  JOIN consulta c USING(cod\_angajat)  JOIN pacient p USING(cod\_pacient)  WHERE c.data > TO\_DATE(data\_inceput, 'DD-MON-YY')  AND p.sex = 'M'  GROUP BY cod\_angajat, nume  HAVING COUNT(\*) = (  SELECT MAX(COUNT(\*))  FROM doctor d JOIN consulta c USING(cod\_angajat)  JOIN pacient p USING(cod\_pacient)  WHERE c.data > TO\_DATE(data\_inceput, 'DD-MON-YY')  AND p.sex = 'M'  GROUP BY cod\_angajat);  RETURN nume;  EXCEPTION  WHEN TOO\_MANY\_ROWS THEN  RAISE\_APPLICATION\_ERROR(-20000, 'Sunt mai multi doctori cu numarul maxim de consultatii');  WHEN NO\_DATA\_FOUND THEN  RAISE\_APPLICATION\_ERROR(-20001, 'Nu exista consultatii atat de recente');  WHEN OTHERS THEN  IF SQLCODE = -20002 THEN  RAISE\_APPLICATION\_ERROR(-20002, 'Data nu este valida');  ELSE  RAISE\_APPLICATION\_ERROR(-20003, 'Nu stiu ce s-a intamplat ' || SQLERRM);  END IF;  END doctor\_consultatii\_max;  / |

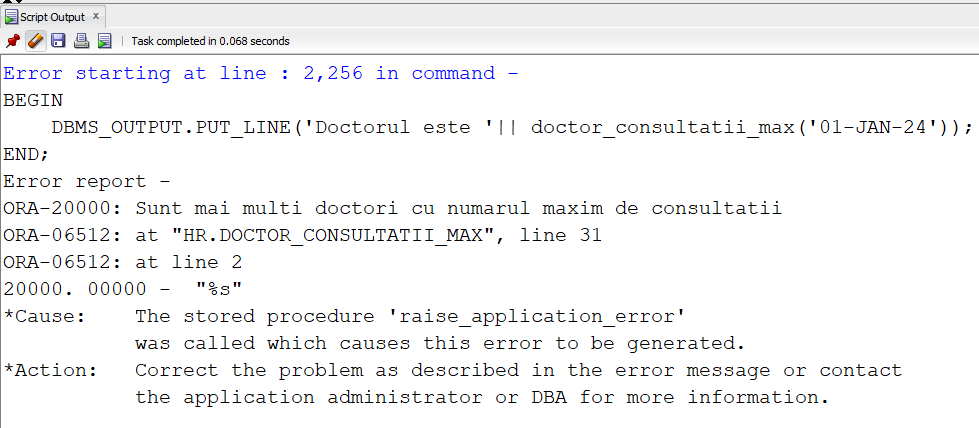


Testez:

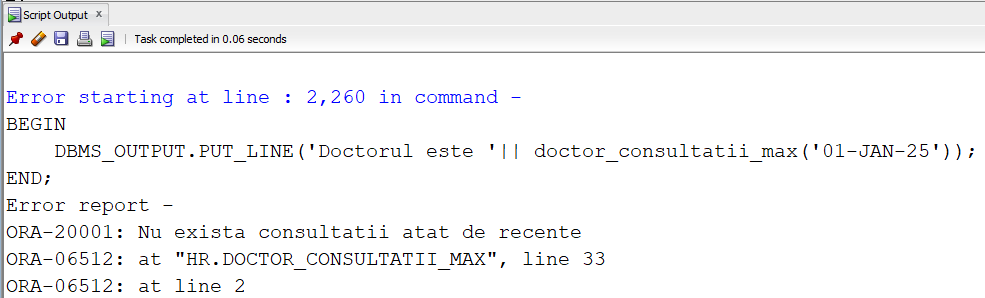
|  |
| --- |
| BEGIN  DBMS\_OUTPUT.PUT\_LINE('Doctorul este '|| doctor\_consultatii\_max('nu o data'));  END;  / |



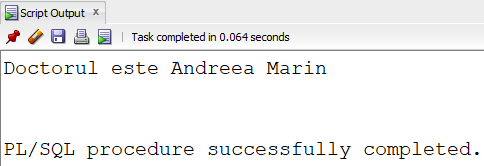
|  |
| --- |
| BEGIN  DBMS\_OUTPUT.PUT\_LINE('Doctorul este '|| doctor\_consultatii\_max('01-JAN-24'));  END;  / |



|  |
| --- |
| BEGIN  DBMS\_OUTPUT.PUT\_LINE('Doctorul este '|| doctor\_consultatii\_max('01-JAN-25'));  END;  / |



|  |
| --- |
| BEGIN  DBMS\_OUTPUT.PUT\_LINE('Doctorul este '|| doctor\_consultatii\_max('09-JAN-24'));  END;  / |



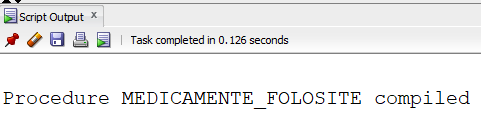
## Cerinta 9:

Sa se creeze o procedura care primeste ca parametri numele unui doctor si un string x (care

reprezinta un numar) si intoarce prin al doilea parametru primele x cele mai folosite

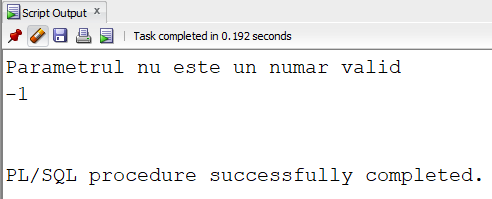
medicamente in tratamentele prescrise de acesta.

|  |
| --- |
| CREATE OR REPLACE PROCEDURE medicamente\_folosite(nume\_doctor angajat.nume%TYPE, nr\_medicamente IN OUT VARCHAR2) IS  v\_nr NUMBER;  v\_ang angajat.nume%TYPE;  NU\_DOCTOR EXCEPTION;  PREA\_PUTINE\_MEDICAMENTE EXCEPTION;  TYPE refc IS REF CURSOR;  c\_medicamente refc;  denumire\_med medicament.denumire%TYPE;  rez VARCHAR2(100) := '';  BEGIN  -- verific ca parametrul al doilea e numar  v\_nr := TO\_NUMBER(nr\_medicamente);  -- verific ca primul parametru e numele unui angajat  SELECT nume  INTO v\_ang  FROM angajat  WHERE nume = nume\_doctor;  -- verific ca primul parametru e numele unui doctor  SELECT COUNT(\*)  INTO v\_nr  FROM angajat a JOIN doctor d USING(cod\_angajat)  WHERE a.nume = nume\_doctor;    IF v\_nr = 0 THEN  RAISE NU\_DOCTOR;  ELSIF v\_nr > 1 THEN  RAISE TOO\_MANY\_ROWS;  END IF;    -- verific ca sunt destule medicamente diferite folosite de acel doctor  SELECT COUNT(COUNT(\*))  INTO v\_nr  FROM angajat a JOIN doctor d USING(cod\_angajat)  JOIN tratament t USING(cod\_angajat)  JOIN necesita n USING(cod\_tratament)  JOIN medicament m USING(cod\_medicament)  WHERE a.nume = nume\_doctor  GROUP BY cod\_medicament;    IF v\_nr < TO\_NUMBER(nr\_medicamente) THEN  RAISE PREA\_PUTINE\_MEDICAMENTE;  END IF;    v\_nr := TO\_NUMBER(nr\_medicamente);  v\_ang := nume\_doctor;  -- cursor dinamic care returneaza denimrea si numarul celor mai folosite x medicamente  OPEN c\_medicamente FOR  'SELECT denumire, COUNT(\*) "numar"  FROM angajat a JOIN doctor d USING(cod\_angajat)  JOIN tratament t USING(cod\_angajat)  JOIN necesita n USING(cod\_tratament)  JOIN medicament m USING(cod\_medicament)  WHERE nume = :v\_ang AND  cod\_medicament IN (  SELECT cod\_medicament  FROM (  SELECT cod\_medicament  FROM angajat a JOIN doctor d USING(cod\_angajat)  JOIN tratament t USING(cod\_angajat)  JOIN necesita n USING(cod\_tratament)  JOIN medicament m USING(cod\_medicament)  WHERE nume = :v\_ang  GROUP BY cod\_medicament  ORDER BY COUNT(\*) DESC)  WHERE ROWNUM < :v\_nr)  GROUP BY denumire  '  USING v\_ang, v\_ang, v\_nr;  LOOP  FETCH c\_medicamente INTO denumire\_med, v\_nr;  EXIT WHEN c\_medicamente%NOTFOUND;  rez := rez || ' ' || denumire\_med || ':' || TO\_CHAR(v\_nr);  END LOOP;  DBMS\_OUTPUT.PUT\_LINE(rez);  nr\_medicamente := rez;  CLOSE c\_medicamente;  EXCEPTION  WHEN VALUE\_ERROR THEN  DBMS\_OUTPUT.PUT\_LINE('Parametrul nu este un numar valid');  nr\_medicamente := '-1';  WHEN NO\_DATA\_FOUND THEN  DBMS\_OUTPUT.PUT\_LINE('Numele dat nu este al unui angajat');  nr\_medicamente := '-1';  WHEN TOO\_MANY\_ROWS THEN  DBMS\_OUTPUT.PUT\_LINE('Mai multi doctori cu acest nume');  nr\_medicamente := '-1';  WHEN NU\_DOCTOR THEN  DBMS\_OUTPUT.PUT\_LINE('Numele dat nu este al unui doctor');  nr\_medicamente := '-1';  WHEN PREA\_PUTINE\_MEDICAMENTE THEN  DBMS\_OUTPUT.PUT\_LINE('Nu sunt destule medicamente diferite folosite de doctor');  nr\_medicamente := '-1';  WHEN OTHERS THEN  DBMS\_OUTPUT.PUT\_LINE('Alta eroare: ' || SQLERRM);  END;  / |

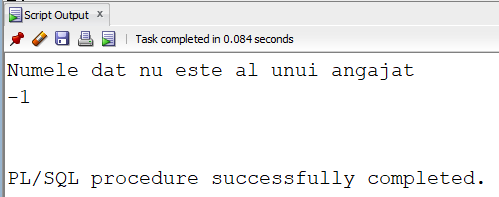


Testez:

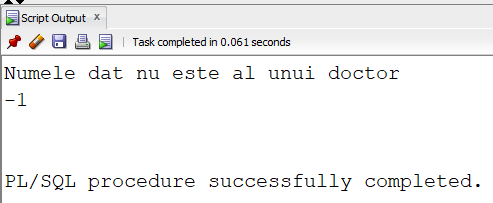
|  |
| --- |
| DECLARE  nr\_medicamente VARCHAR2(20) := 'nu numar';  BEGIN  medicamente\_folosite('Dan Gheorghe', nr\_medicamente);  DBMS\_OUTPUT.PUT\_LINE(nr\_medicamente);  END;  / |



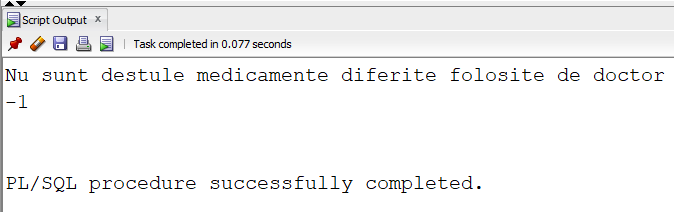
|  |
| --- |
| DECLARE  nr\_medicamente VARCHAR2(20) := '3';  BEGIN  medicamente\_folosite('Gigel', nr\_medicamente);  DBMS\_OUTPUT.PUT\_LINE(nr\_medicamente);  END;  / |



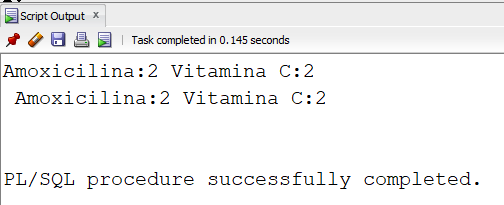
|  |
| --- |
| DECLARE  nr\_medicamente VARCHAR2(20) := '3';  BEGIN  medicamente\_folosite('Maria Popescu', nr\_medicamente);  DBMS\_OUTPUT.PUT\_LINE(nr\_medicamente);  END;  / |



|  |
| --- |
| DECLARE  nr\_medicamente VARCHAR2(20) := '11';  BEGIN  medicamente\_folosite('Radu Iliescu', nr\_medicamente);  DBMS\_OUTPUT.PUT\_LINE(nr\_medicamente);  END;  / |



|  |
| --- |
| DECLARE  nr\_medicamente VARCHAR2(50) := '3';  BEGIN  medicamente\_folosite('Radu Iliescu', nr\_medicamente);  DBMS\_OUTPUT.PUT\_LINE(nr\_medicamente);  END;  / |



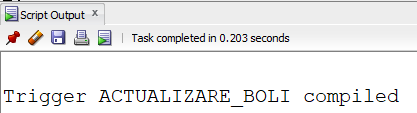
|  |
| --- |
| -- pentru teste am mai inserat:  INSERT INTO tratament VALUES(406, 202, 10, 300);  INSERT INTO necesita VALUES(406, 503);  INSERT INTO necesita VALUES(406, 504); |

## Cerinta 10:

Pentru orice schimbari in tabelul 'primeste', sa se actualizeze in tabelul 'boala' numarul de

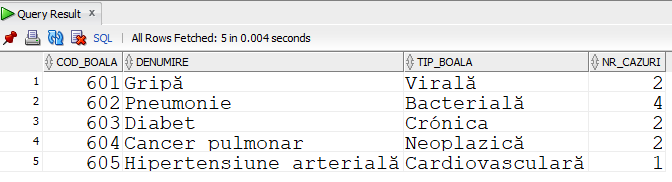
cazuri

|  |
| --- |
| CREATE OR REPLACE TRIGGER actualizare\_boli  AFTER UPDATE OR DELETE OR INSERT ON primeste  DECLARE  v\_nr NUMBER;  BEGIN  FOR i IN (SELECT cod\_boala FROM boala) LOOP  SELECT COUNT(\*)  INTO v\_nr  FROM primeste  WHERE cod\_boala = i.cod\_boala;    UPDATE boala  SET nr\_cazuri = v\_nr  WHERE cod\_boala = i.cod\_boala;  END LOOP;  END;  / |

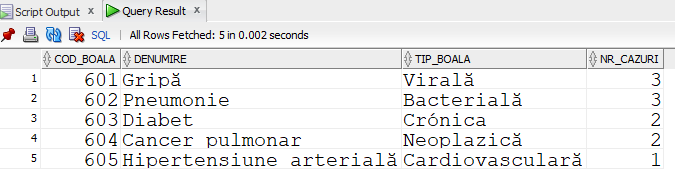


Testez:

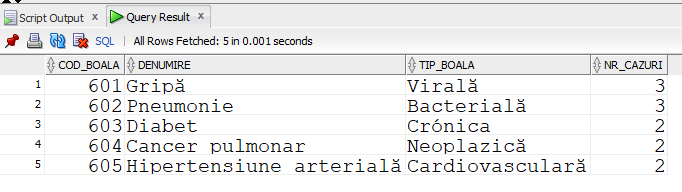
|  |
| --- |
| select \* from boala; |



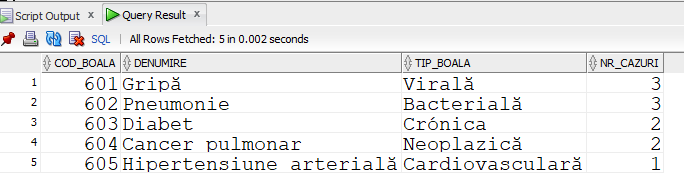
|  |
| --- |
| UPDATE primeste  SET cod\_boala = 602  WHERE cod\_pacient = 201 AND cod\_tratament = 401 AND cod\_boala = 601; |



|  |
| --- |
| INSERT INTO primeste(cod\_pacient, cod\_tratament, cod\_boala) VALUES (201, 402, 605); |



|  |
| --- |
| DELETE FROM primeste WHERE cod\_pacient = 201 AND cod\_tratament = 402 AND cod\_boala = 605; |



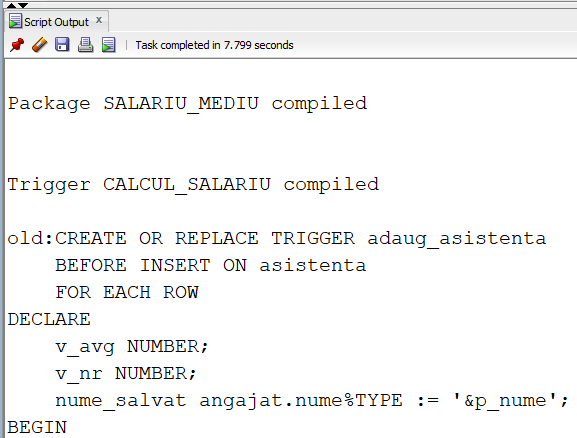
## Cerinta 11:

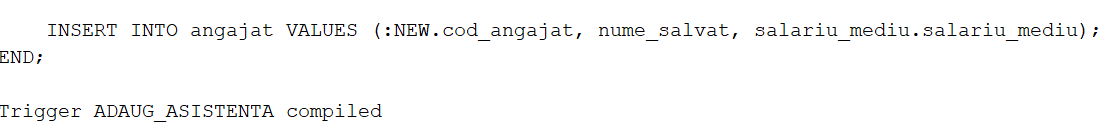
Sa se creeze un trigger care sa introduca automat o intrare in tabelul 'angajat' atunci cand o

noua asistenta este introudsa in baza de date. Noua asistenta va avea salariul mediu al tuturor

asistentelor.

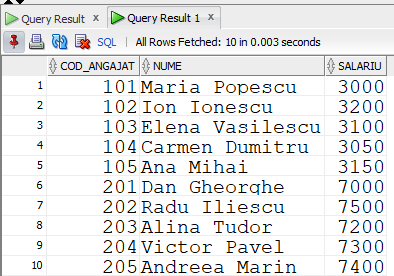
|  |
| --- |
| CREATE OR REPLACE PACKAGE salariu\_mediu AS  salariu\_mediu NUMBER;  END;  /  CREATE OR REPLACE TRIGGER calcul\_salariu  BEFORE INSERT ON asistenta  BEGIN  SELECT AVG(salariu)  INTO salariu\_mediu.salariu\_mediu  FROM angajat JOIN asistenta USING(cod\_angajat);  END;  /  CREATE OR REPLACE TRIGGER adaug\_asistenta  BEFORE INSERT ON asistenta  FOR EACH ROW  DECLARE  v\_avg NUMBER;  v\_nr NUMBER;  nume\_salvat angajat.nume%TYPE := '&p\_nume';  BEGIN  -- verific ca nu exista deja angajatul  SELECT COUNT(\*)  INTO v\_nr  FROM angajat  WHERE cod\_angajat = :NEW.cod\_angajat;    IF v\_nr = 1 THEN  RAISE\_APPLICATION\_ERROR(-20000, 'Angajatul cu codul ' || :NEW.cod\_angajat || ' exista deja');  END IF;  INSERT INTO angajat VALUES (:NEW.cod\_angajat, nume\_salvat, salariu\_mediu.salariu\_mediu);  END;  / |

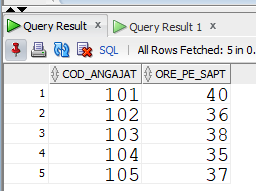




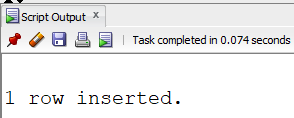
Testez:

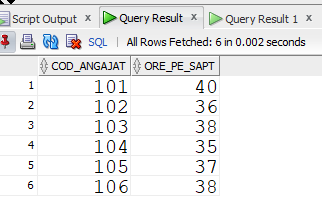
|  |
| --- |
| select \* from asistenta;  select \* from angajat; |

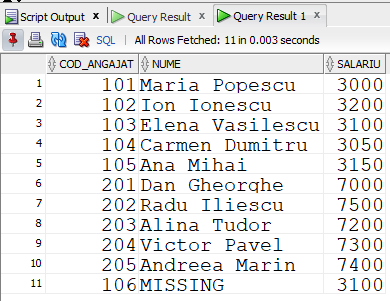




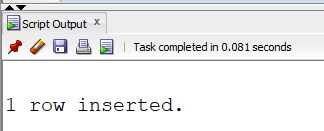
|  |
| --- |
| -- merge daca inserez doar o linie  INSERT INTO asistenta VALUES (105, 38); |

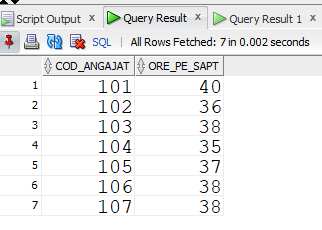


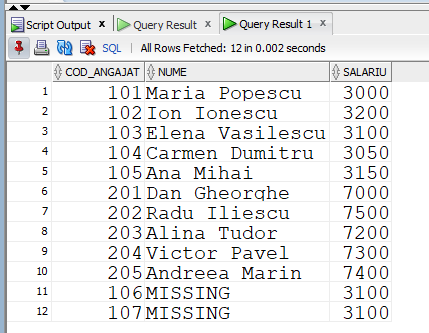




|  |
| --- |
| -- ar fi aparut eroarea mutating table daca folosesc un select (posibil inserez mai multe linii)  INSERT INTO asistenta SELECT 107, 38 FROM DUAL; |





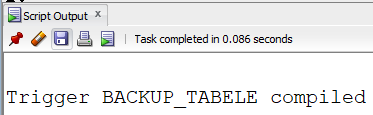


## Cerinta 12:

Sa se creeze backup-uri automate pentru tabelele medicament, boala si pacient in cazul in

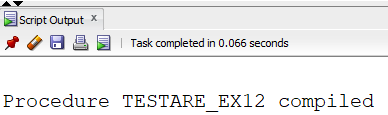
care sunt sterse

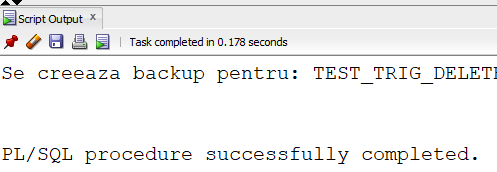
|  |
| --- |
| CREATE OR REPLACE TRIGGER backup\_tabele  BEFORE DROP ON SCHEMA  DECLARE  nume VARCHAR2(100);  BEGIN  IF SYS.DICTIONARY\_OBJ\_TYPE = 'TABLE'  AND LOWER(SYS.DICTIONARY\_OBJ\_NAME) IN ('medicament', 'boala', 'pacient', 'test\_trig\_delete') THEN  DBMS\_OUTPUT.PUT\_LINE('Se creeaza backup pentru: ' || SYS.DICTIONARY\_OBJ\_NAME);  nume := SYS.DICTIONARY\_OBJ\_NAME || '\_backup\_' || TO\_CHAR(SYSDATE, 'YYYYMMDDHH24MISS');  EXECUTE IMMEDIATE 'CREATE TABLE ' || nume || ' AS SELECT \* FROM ' || SYS.DICTIONARY\_OBJ\_NAME;  END IF;  END;  / |

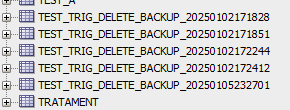


Testez (folosesc tabelul test\_trig\_delete pentru ca nu vreau sa imi sterg tabelele importante):

|  |
| --- |
| CREATE TABLE test\_trig\_delete AS SELECT cod\_angajat FROM angajat;  DROP TABLE test\_trig\_delete;  CREATE OR REPLACE PROCEDURE testare\_ex12 IS  v\_nr NUMBER;  BEGIN  SELECT COUNT(\*)  INTO v\_nr  FROM USER\_TABLES  WHERE TABLE\_NAME = 'TEST\_TRIG\_DELETE';  IF v\_nr = 0 THEN  EXECUTE IMMEDIATE 'CREATE TABLE test\_trig\_delete AS SELECT cod\_angajat FROM angajat';  END IF;  EXECUTE IMMEDIATE 'DROP TABLE test\_trig\_delete';  END;  / |

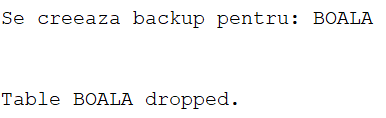


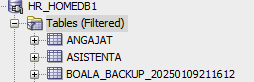




(L-am rulat de mai multe ori)

|  |
| --- |
| DROP TABLE boala CASCADE CONSTRAINTS; |





## Cerinta 13:

Sa se creeze un pachet cu 3 proceduri: actualizare a valabilitatii tuturor paturilor

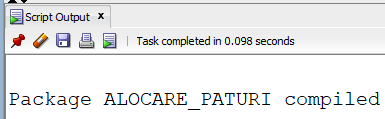
(stocata ca tabel imbricat in pachet), afisare paturi disponibile, eliberare pat.

In plus, 2 functii: verificare daca o camera este plina (returneaza boolean), alocare pat pentru

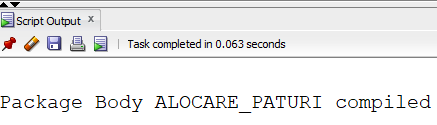
un pacient nou internat, intr-o anumita camera (returneaza numarul patului in care va fi

internat)

|  |
| --- |
| CREATE OR REPLACE PACKAGE alocare\_paturi AS  TYPE rec\_pat IS RECORD (  cod\_pat pat.cod\_pat%TYPE,  valabil NUMBER(1)  );  TYPE tab\_paturi IS TABLE OF rec\_pat;  paturi tab\_paturi := tab\_paturi();    PROCEDURE ACTUALIZARE\_PATURI\_VALABILE;  FUNCTION CAMERA\_PLINA(cod camera.cod\_camera%TYPE) RETURN BOOLEAN;  PROCEDURE AFISARE\_PATURI\_VALABILE;  FUNCTION ALOCARE\_PAT(  p\_cod\_pacient NUMBER,  p\_cod\_camera NUMBER,  p\_data\_internare DATE  ) RETURN pat.cod\_pat%TYPE;  PROCEDURE ELIBERARE\_PAT(  p\_cod\_pat NUMBER  );  END alocare\_paturi;  / |

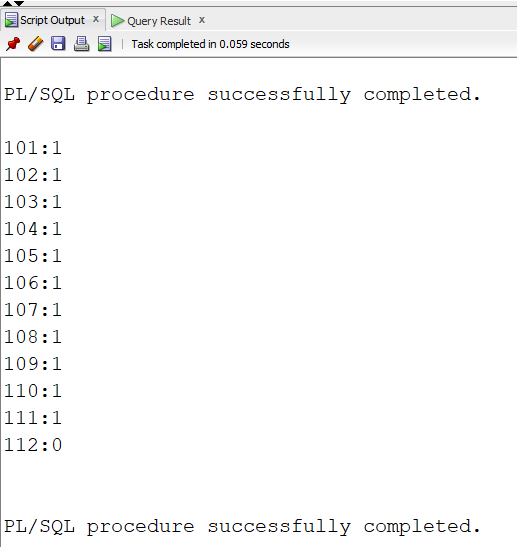


|  |
| --- |
| CREATE OR REPLACE PACKAGE BODY alocare\_paturi AS  PROCEDURE ACTUALIZARE\_PATURI\_VALABILE IS  BEGIN  -- initial setez pentru fiecare pat ca e gol (valoare '0')  SELECT cod\_pat, 0  BULK COLLECT INTO paturi  FROM pat;  -- trec prin fiecare pat si verific daca e cineva internat in el  FOR i IN paturi.FIRST..paturi.LAST LOOP  SELECT CASE  WHEN NOT EXISTS (SELECT 1  FROM ocupa  WHERE cod\_pat = paturi(i).cod\_pat  AND data\_plecare IS NULL)  THEN 1  ELSE 0  END  INTO paturi(i).valabil  FROM dual;  END LOOP;  END ACTUALIZARE\_PATURI\_VALABILE;    PROCEDURE AFISARE\_PATURI\_VALABILE IS  BEGIN  FOR i IN paturi.FIRST..paturi.LAST LOOP  DBMS\_OUTPUT.PUT\_LINE(paturi(i).cod\_pat || ':' || TO\_CHAR(paturi(i).valabil));  END LOOP;  END AFISARE\_PATURI\_VALABILE;    FUNCTION CAMERA\_PLINA(cod camera.cod\_camera%TYPE) RETURN BOOLEAN IS  paturi\_totale NUMBER;  paturi\_ocupate NUMBER;  BEGIN  -- numar paturi totale din camera  SELECT COUNT(\*) INTO paturi\_totale FROM pat WHERE cod\_camera = cod;  -- numar paturi ocupate din camera  SELECT COUNT(\*) INTO paturi\_ocupate  FROM ocupa  WHERE cod\_camera = cod AND data\_plecare IS NULL;  RETURN paturi\_totale = paturi\_ocupate;  END CAMERA\_PLINA;      FUNCTION ALOCARE\_PAT(p\_cod\_pacient IN NUMBER, p\_cod\_camera IN NUMBER, p\_data\_internare IN DATE) RETURN pat.cod\_pat%TYPE IS  v\_pat pat.cod\_pat%TYPE := 1;  internat NUMBER;  BEGIN  ACTUALIZARE\_PATURI\_VALABILE;  -- verific daca pacientul nu e deja internat  SELECT COUNT(\*)  INTO internat  FROM ocupa  WHERE cod\_pacient = p\_cod\_pacient AND data\_plecare IS NULL;    IF internat > 0 THEN  RAISE\_APPLICATION\_ERROR(-20001, 'Pacientul este deja internat');  END IF;  -- verifica ca camera in care vrea sa fie internat nu e plina  IF CAMERA\_PLINA(p\_cod\_camera) THEN  RAISE\_APPLICATION\_ERROR(-20000, 'Camera e plina, pacientul nu poate fi internat');  END IF;  -- gasesc primul pat neocupat din acea camera  SELECT cod\_pat  INTO v\_pat  FROM TABLE (paturi)  WHERE ROWNUM <= 1 AND valabil = 1 AND cod\_pat IN (  SELECT cod\_pat  FROM camera c JOIN pat p USING(cod\_camera)  WHERE cod\_camera = p\_cod\_camera  );    DBMS\_OUTPUT.PUT\_LINE(v\_pat);    INSERT INTO ocupa (cod\_pacient, cod\_camera, cod\_pat, data\_internare, data\_plecare)  VALUES (p\_cod\_pacient, p\_cod\_camera, v\_pat, p\_data\_internare, NULL);    ACTUALIZARE\_PATURI\_VALABILE;  RETURN v\_pat;  END ALOCARE\_PAT;      PROCEDURE ELIBERARE\_PAT(p\_cod\_pat NUMBER) IS  v\_nr NUMBER(1);  BEGIN  ACTUALIZARE\_PATURI\_VALABILE;  -- verific daca patul e ocupat  SELECT valabil  INTO v\_nr  FROM TABLE (paturi)  WHERE cod\_pat = p\_cod\_pat;    IF v\_nr = 1 THEN  RAISE\_APPLICATION\_ERROR(-20001, 'Patul nu este ocupat');  END IF;    UPDATE ocupa  SET data\_plecare = SYSDATE  WHERE cod\_pat = p\_cod\_pat AND data\_plecare IS NULL;    ACTUALIZARE\_PATURI\_VALABILE;    END ELIBERARE\_PAT;  END alocare\_paturi;  / |

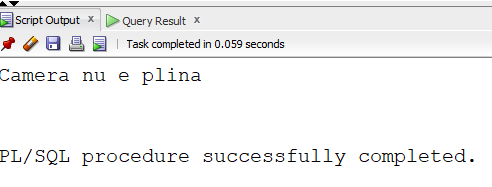


Testez:

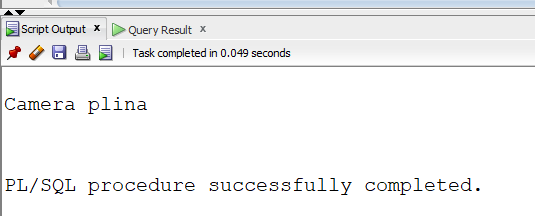
|  |
| --- |
| -- ocup camera 5 (are un singur pat)  INSERT INTO ocupa VALUES (201, 112, 5, SYSDATE, NULL);  EXECUTE alocare\_paturi.ACTUALIZARE\_PATURI\_VALABILE;  EXECUTE alocare\_paturi.AFISARE\_PATURI\_VALABILE; |



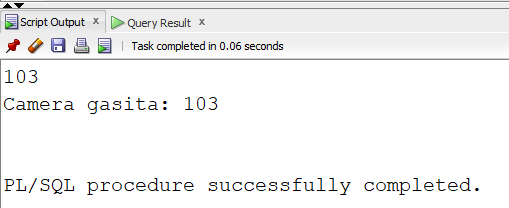
|  |
| --- |
| DECLARE  x BOOLEAN;  BEGIN  x := alocare\_paturi.CAMERA\_PLINA(1);  IF x THEN  DBMS\_OUTPUT.PUT\_LINE('Camera plina');  ELSE  DBMS\_OUTPUT.PUT\_LINE('Camera nu e plina');  END IF;  END;  / |

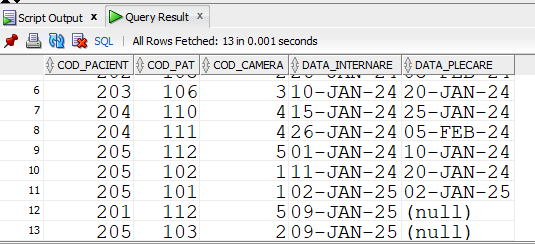


|  |
| --- |
| DECLARE  x BOOLEAN;  BEGIN  x := alocare\_paturi.CAMERA\_PLINA(5);  IF x THEN  DBMS\_OUTPUT.PUT\_LINE('Camera plina');  ELSE  DBMS\_OUTPUT.PUT\_LINE('Camera nu e plina');  END IF;  END;  / |



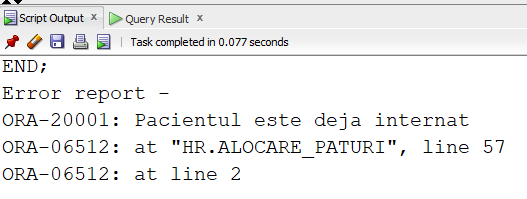
|  |
| --- |
| BEGIN  DBMS\_OUTPUT.PUT\_LINE('Camera gasita: ' || TO\_CHAR(alocare\_paturi.ALOCARE\_PAT(205, 2, SYSDATE)));  END;  / |



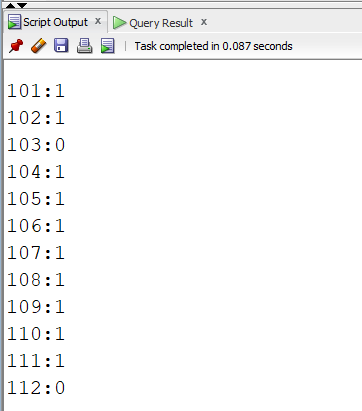


Dam acelasi input (acum pacientul e deja internat):

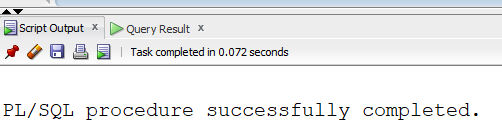
|  |
| --- |
| BEGIN  DBMS\_OUTPUT.PUT\_LINE('Camera gasita: ' || TO\_CHAR(alocare\_paturi.ALOCARE\_PAT(205, 2, SYSDATE)));  END;  / |

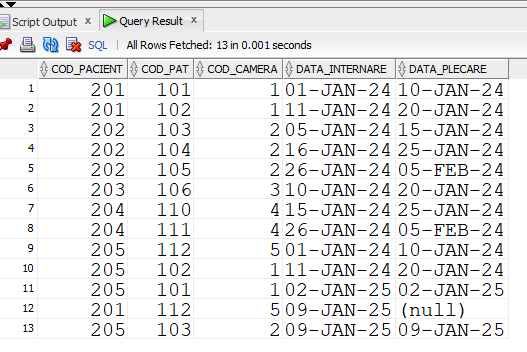


|  |
| --- |
| EXECUTE alocare\_paturi.AFISARE\_PATURI\_VALABILE; |



|  |
| --- |
| EXECUTE alocare\_paturi.ELIBERARE\_PAT(103); |





Acelasi input (patul e acum liber):

|  |
| --- |
| EXECUTE alocare\_paturi.ELIBERARE\_PAT(103); |

