**Gestiunea activității unei clinici medicale**

Asavei Roxana, seria 25, grupa 251

2024 - 2025

### Cuprins

[Cuprins 2](#_clde9mj76ov1)

[Introducere 3](#_4v5w1h4p4d90)

[Descrierea modelului 4](#_kj3giflx2gpn)

[Diagrama ER 4](#_8w4zupn6zo53)

[Diagrama conceptuală 5](#_ejoworin74p)

[Implementarea diagramei 7](#_8t1xrucwfd6t)

[Popularea bazei de date 12](#_5g6uczdg50j5)

[Cerința 6 35](#_s3416nhii51m)

[Cerința 7 40](#_7ohgtd1q10sg)

[Cerința 8 44](#_95h7rzc263i)

[Cerința 9 48](#_bf7g17z2xspc)

[Cerința 10 53](#_eyacd17ax83y)

[Cerința 11 65](#_s2q9jw1j9qko)

[Cerința 12 72](#_gxeo8alojp9)

[Cerința 13 77](#_hcghlwwpwu0b)

### Introducere

Proiectul ilustrează activitatea din cadrul unei clinici medicale, având aplicații reale: se pot determina procedurile realizate de un medic într-un anumit interval, asistenții medicali care lucrează pe o anumită secție, angajații care ar putea interacționa cu un anumit pacient astfel încât să nu existe bariere lingvistice, cele mai cerute analize de către pacienți și multe alte aspecte relevante.

În implementarea acestui proiect, toate cerințele au fost executate pe serverul facultății, versiunea fiind Oracle 19c. Nu s-a folosit o mașină virtuală, sistemul de operare fiind Windows 11. Dispozitivul pe care s-a rulat codul are procesorul AMD Ryzen 7 8840HS w/ Radeon 780M Graphics 3.30 GHz și o memorie RAM de 16GB, din care s-au alocat 4GB pentru SGBD.

### Descrierea modelului

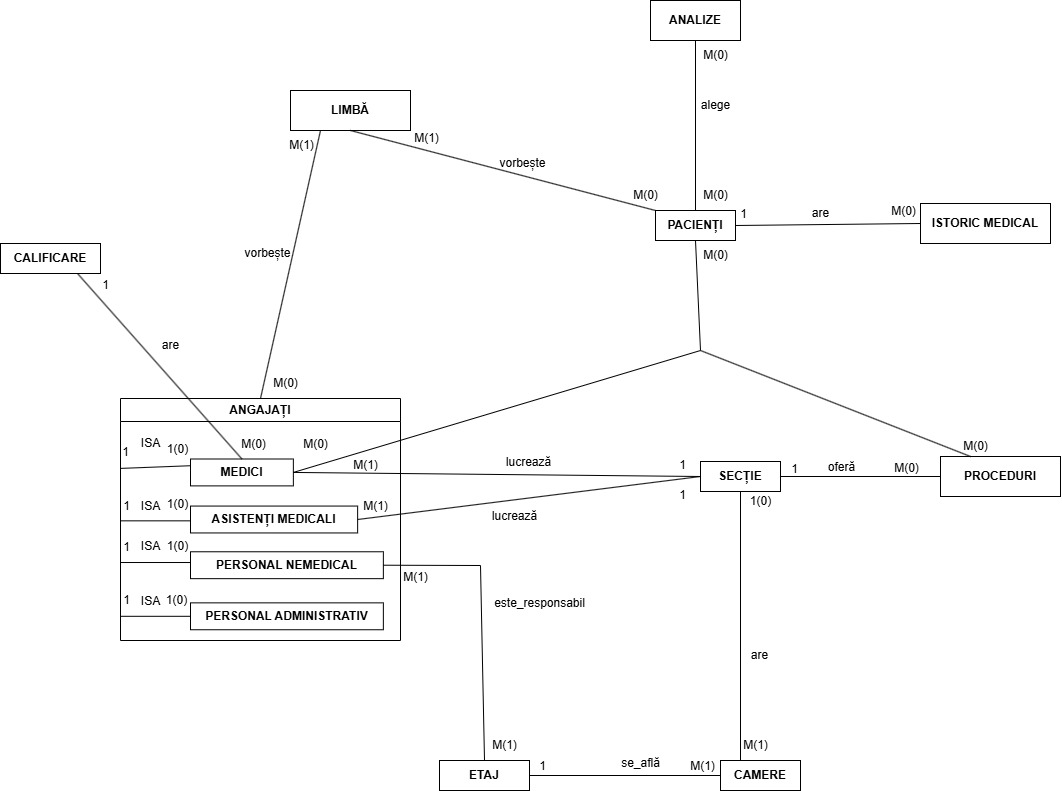
Pentru o bună organizare, angajații din cadrul clinicii se împart în 4 categorii: medici, asistenți medicali, personal nemedical (agent de curățenie, recepționer, agent de pază) și personal administrativ (manager, specialist resurse umane, specialist IT etc.). Medicii și asistenții medicali sunt repartizați unei anumite secții, în timp ce personalul nemedical este asignat unor etaje. Fiecare secție are diverse camere în care își desfășoară activitatea, iar fiecare cameră se află la un anumit etaj. În funcție de experiența acumulată, medicii au o anumită calificare (rezident, specialist, primar, generalist). Clinica este frecventată de pacienți. Pentru fiecare pacient se ține evidența afecțiunilor anterioare sub forma unui istoric medical. Pacienții pot alege să efectueze anumite analize la clinică. De asemenea, pacienții pot fi supuși anumitor proceduri specifice fiecărei specializări și realizate de către medicii din clinică. Pentru a limita barierele lingvistice care pot apărea între angajații clinicii și pacienți, se vor reține limbile cunoscute de către ambele categorii.

Pentru a menține integritatea bazei de date, se impune următorul set de constrângeri:

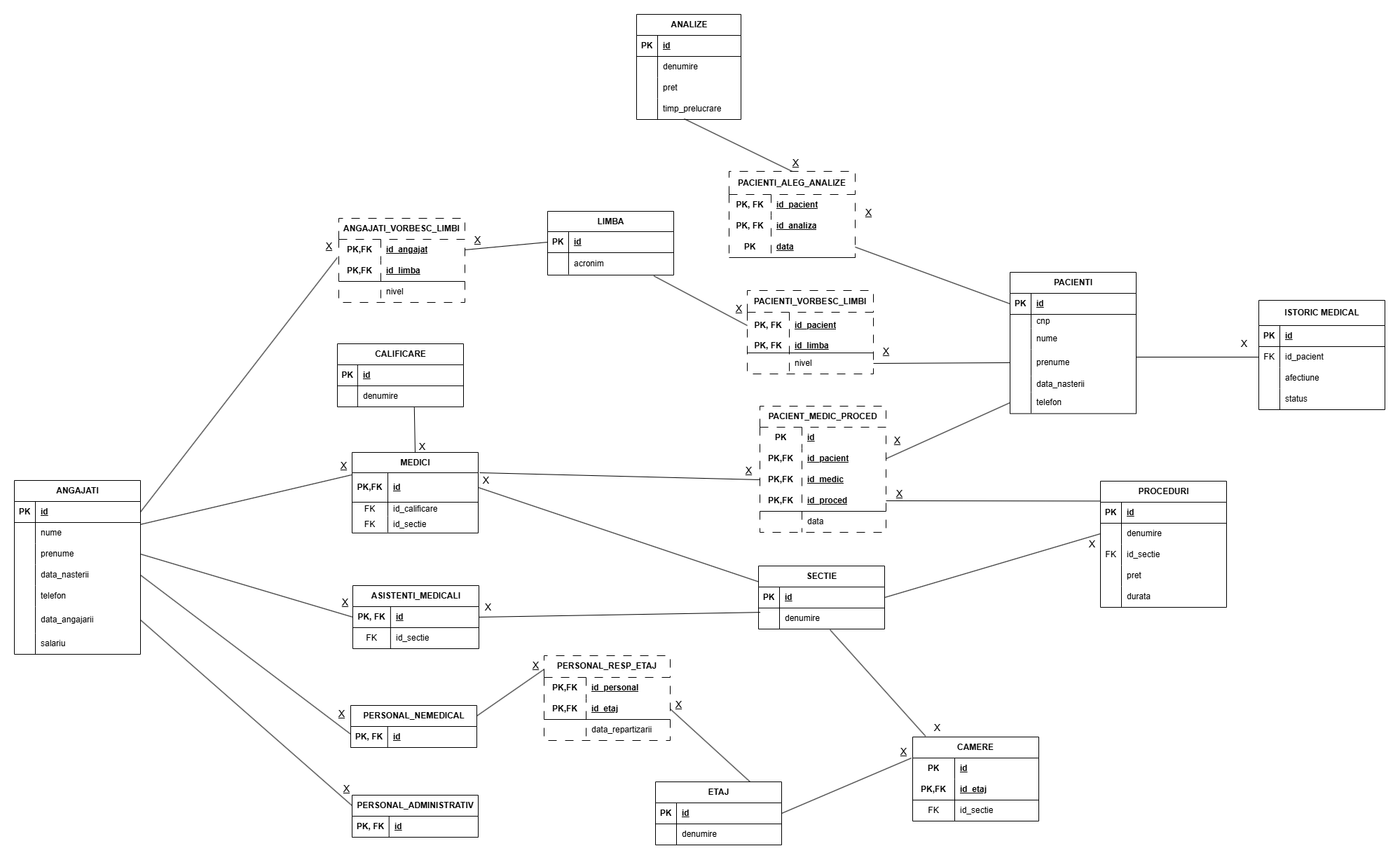
* Pacienții și angajații pot vorbi mai multe limbi; fiecare persoană cunoaște cel puțin o limbă (cea nativă);
* În baza de date pot exista limbi care la un moment dat să nu fie vorbite de niciun pacient, respectiv de către niciun angajat al clinicii;
* Pacienții pot alege să efectueze mai multe analize, iar o anumită analiză poate fi aleasă de mai mulți pacienți;
* Pot exista pacienți fără istoric medical. De asemenea, un pacient poate avea asociate mai multe istorice medicale (a suferit mai multe afecțiuni în trecut);
* Fiecare medic are o singură calificare;
* Atât medicii, cât și asistenții medicali trebuie să lucreze pe o singură secție;
* Există camere care nu sunt repartizate niciunei secții;
* Există secții care nu oferă nicio procedură specifică;
* Pe fiecare etaj trebuie să lucreze personal nemedical;
* Pot exista maxim 10 secții;
* La o procedură pot participa mai mulți pacienți și mai mulți medici simultan;

### 

### Diagrama ER



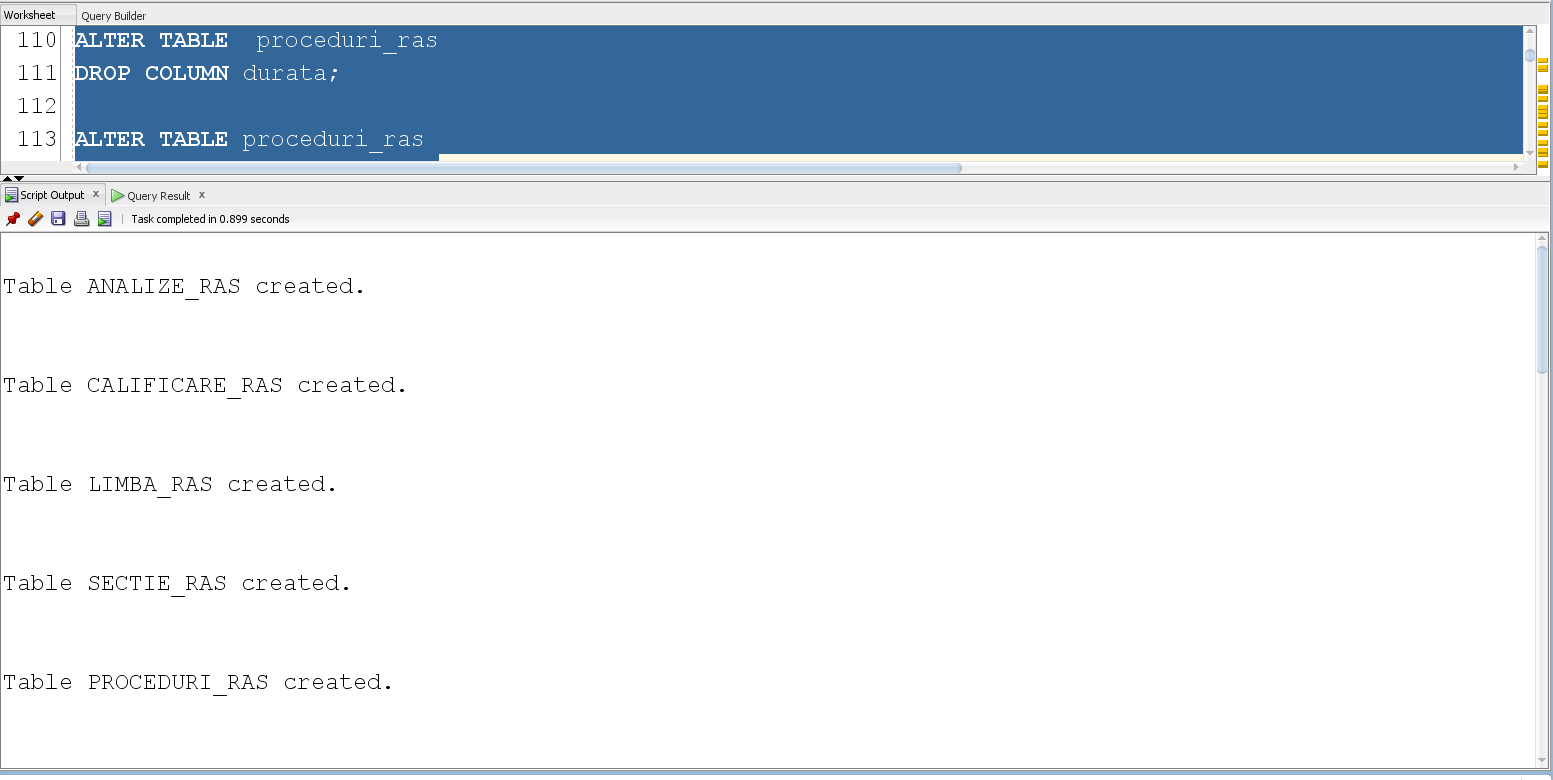
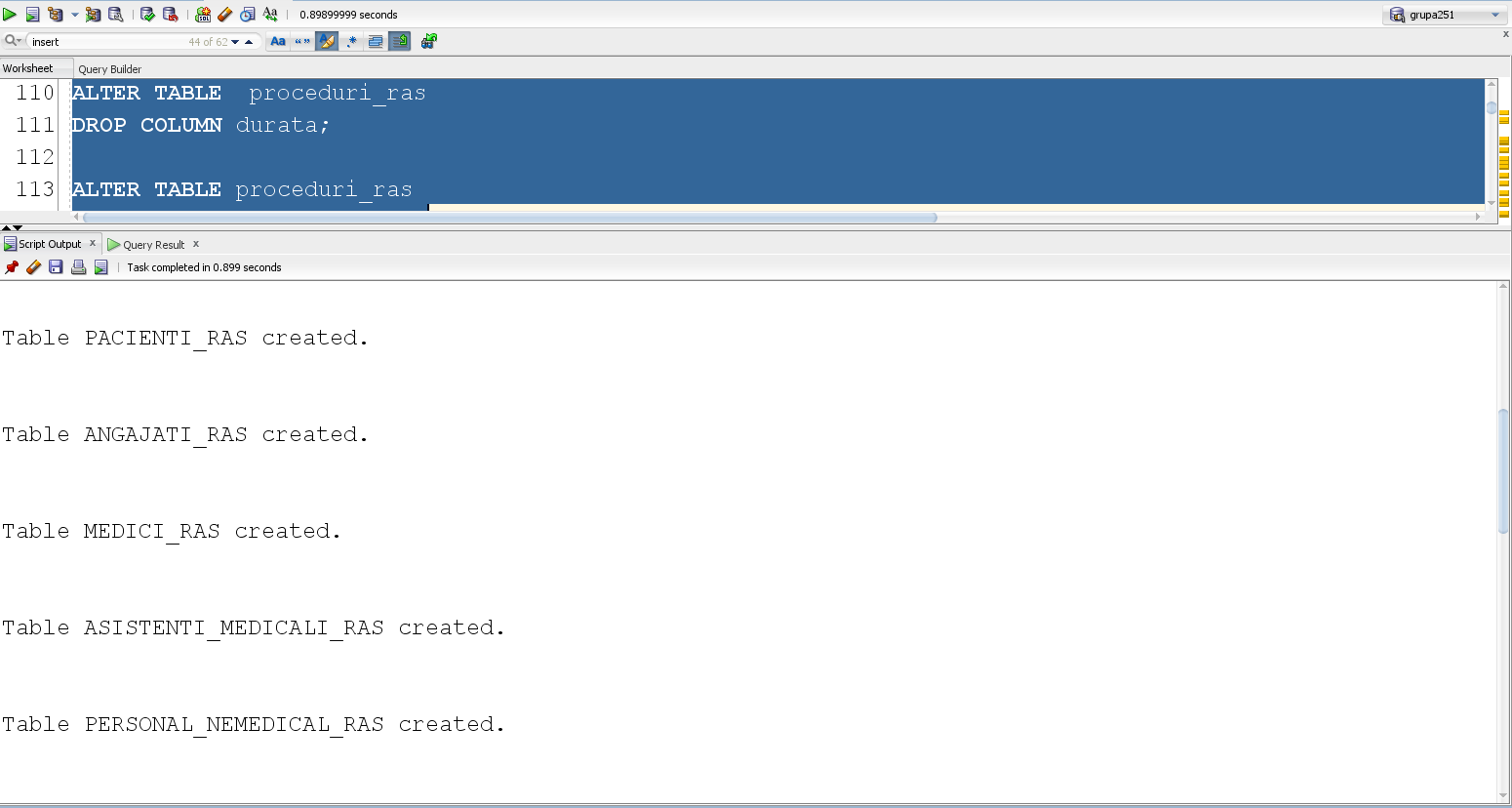
### Diagrama conceptuală

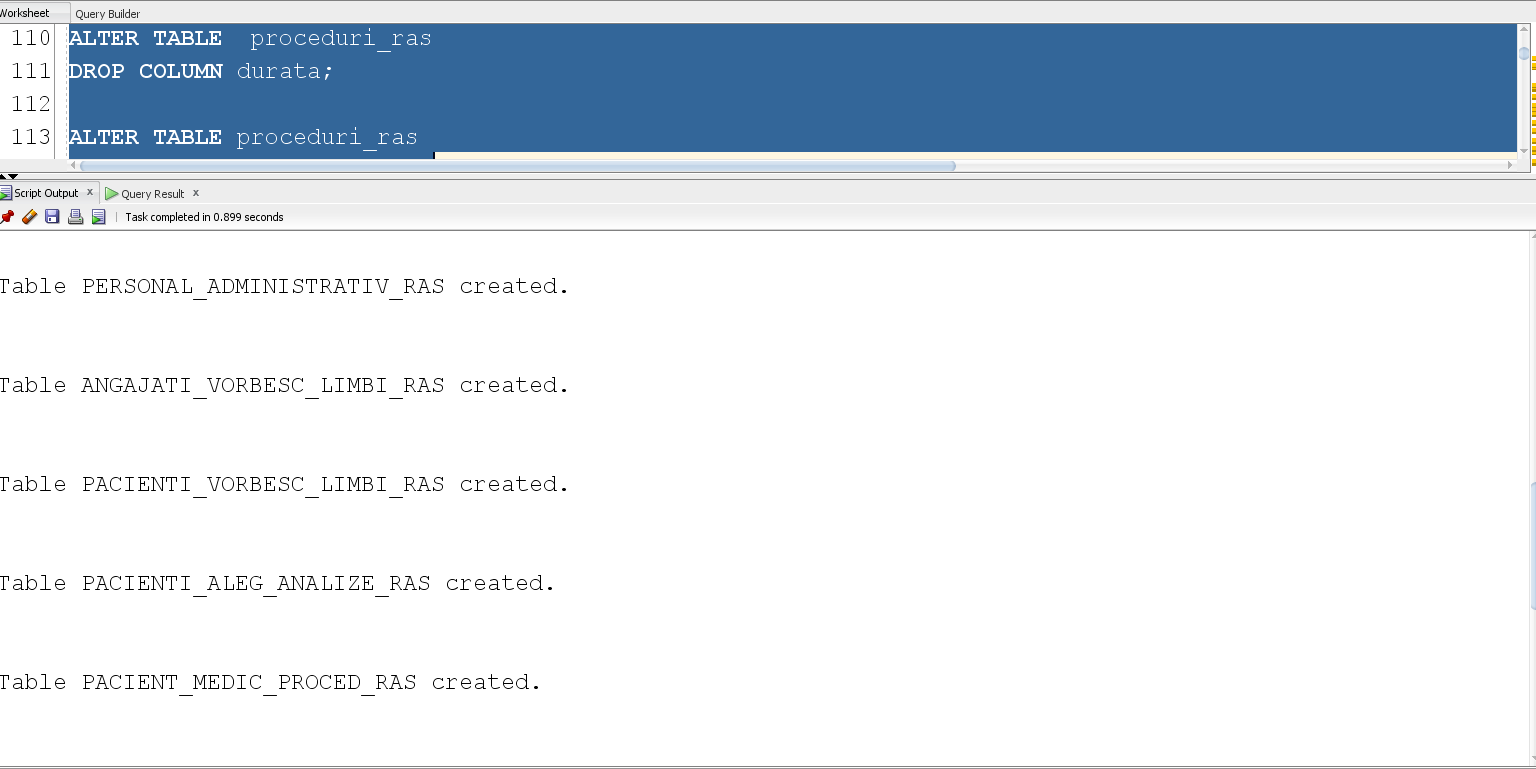
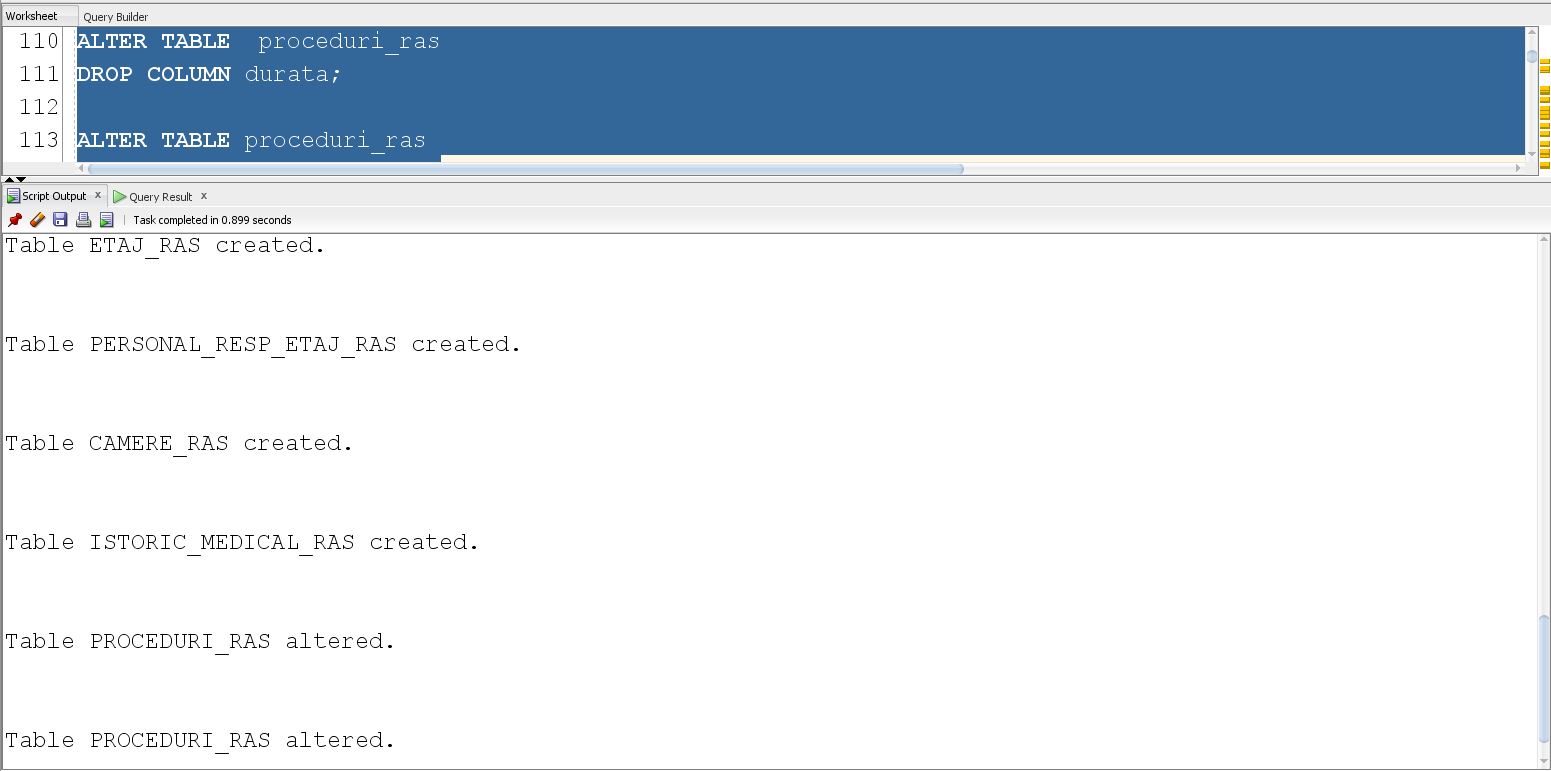


### 

### Implementarea diagramei

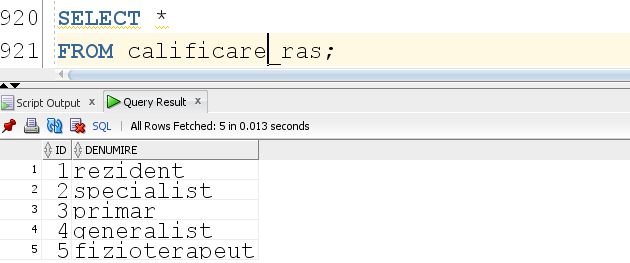
|  |
| --- |
| CREATE TABLE analize\_ras  (id NUMBER PRIMARY KEY,  denumire VARCHAR2(50) NOT NULL,  pret NUMBER NOT NULL,  timp\_prelucrare NUMBER);    CREATE TABLE calificare\_ras  (id NUMBER PRIMARY KEY,  denumire VARCHAR2(30) NOT NULL);  CREATE TABLE limba\_ras  (id NUMBER PRIMARY KEY,  acronim VARCHAR2(3) NOT NULL);    CREATE TABLE sectie\_ras  (id NUMBER PRIMARY KEY,  denumire VARCHAR2(30) NOT NULL);    CREATE TABLE proceduri\_ras  (id NUMBER PRIMARY KEY,  denumire VARCHAR2(50) NOT NULL,  id\_sectie NUMBER CONSTRAINT fk\_proced\_sectie REFERENCES sectie\_ras(id) ON DELETE CASCADE,  pret NUMBER DEFAULT 100,  durata NUMBER);    CREATE TABLE pacienti\_ras  (id NUMBER PRIMARY KEY,  cnp VARCHAR2(13) UNIQUE NOT NULL,  nume VARCHAR2(30),  prenume VARCHAR2(30),  data\_nasterii DATE,  telefon VARCHAR2(15));    CREATE TABLE angajati\_ras  (id NUMBER PRIMARY KEY,  nume VARCHAR2(30),  prenume VARCHAR2(30),  data\_nasterii DATE,  telefon VARCHAR2(15),  data\_angajarii DATE DEFAULT SYSDATE,  salariu NUMBER);    CREATE TABLE medici\_ras  (id NUMBER PRIMARY KEY CONSTRAINT fk\_medici\_ang REFERENCES angajati\_ras(id) ON DELETE CASCADE,  id\_calificare CONSTRAINT fk\_medici\_calif REFERENCES calificare\_ras(id) ON DELETE SET NULL,  id\_sectie CONSTRAINT fk\_medici\_sectie REFERENCES sectie\_ras(id) ON DELETE SET NULL);  CREATE TABLE asistenti\_medicali\_ras  (id NUMBER PRIMARY KEY CONSTRAINT fk\_asist\_medicali REFERENCES angajati\_ras(id) ON DELETE CASCADE,  id\_sectie CONSTRAINT fk\_asist\_medicali\_sectie REFERENCES sectie\_ras(id) ON DELETE SET NULL);    CREATE TABLE personal\_nemedical\_ras  (id NUMBER PRIMARY KEY CONSTRAINT fk\_nemedical REFERENCES angajati\_ras(id) ON DELETE CASCADE);  CREATE TABLE personal\_administrativ\_ras  (id NUMBER PRIMARY KEY CONSTRAINT fk\_adminsitrativ REFERENCES angajati\_ras(id) ON DELETE CASCADE);  CREATE TABLE angajati\_vorbesc\_limbi\_ras  (id\_angajat CONSTRAINT fk\_anglimba\_ang REFERENCES angajati\_ras(id) ON DELETE CASCADE,  id\_limba CONSTRAINT fk\_anglimba\_limba REFERENCES limba\_ras(id) ON DELETE CASCADE,  nivel NUMBER CHECK (nivel >= 1 AND nivel <= 10),  CONSTRAINT pk\_ang\_vorbesc\_limbi PRIMARY KEY (id\_angajat, id\_limba)  );    CREATE TABLE pacienti\_vorbesc\_limbi\_ras  (id\_pacient CONSTRAINT fk\_pacientlimba\_ang REFERENCES pacienti\_ras(id) ON DELETE CASCADE,  id\_limba CONSTRAINT fk\_pacientlimba\_limba REFERENCES limba\_ras(id) ON DELETE CASCADE,  nivel NUMBER CHECK (nivel >= 1 AND nivel <= 10),  CONSTRAINT pk\_pacienti\_vorbesc\_limbi PRIMARY KEY (id\_pacient, id\_limba)  );    CREATE TABLE pacienti\_aleg\_analize\_ras  (id\_pacient CONSTRAINT fk\_pacientanalize\_pacient REFERENCES pacienti\_ras(id) ON DELETE CASCADE,  id\_analiza CONSTRAINT fk\_pacientanalize\_analiza REFERENCES analize\_ras(id) ON DELETE CASCADE,  data DATE DEFAULT SYSDATE,  CONSTRAINT pk\_pacienti\_analize PRIMARY KEY (id\_pacient, id\_analiza, data)  );  CREATE TABLE pacient\_medic\_proced\_ras  (id NUMBER,  id\_pacient CONSTRAINT fk\_pacientproced\_pacient REFERENCES pacienti\_ras(id)ON DELETE CASCADE,  id\_medic CONSTRAINT fk\_pacientproced\_medic REFERENCES medici\_ras(id) ON DELETE CASCADE,  id\_proced CONSTRAINT fk\_pacienproced\_proced REFERENCES proceduri\_ras(id) ON DELETE CASCADE,  data DATE DEFAULT SYSDATE,  CONSTRAINT pk\_pacient\_medic\_proced PRIMARY KEY (id, id\_pacient, id\_medic, id\_proced)  );    CREATE TABLE etaj\_ras  (id NUMBER PRIMARY KEY,  nume VARCHAR2(30) NOT NULL);    CREATE TABLE personal\_resp\_etaj\_ras  (id\_personal CONSTRAINT fk\_personaletaj\_personal REFERENCES personal\_nemedical\_ras(id) ON DELETE CASCADE,  id\_etaj CONSTRAINT fk\_personaletaj\_etaj REFERENCES etaj\_ras(id) ON DELETE CASCADE,  data\_repartizarii DATE DEFAULT SYSDATE,  CONSTRAINT pk\_personaletaj PRIMARY KEY(id\_personal, id\_etaj));    CREATE TABLE camere\_ras -- entitate dependenta de etaj  (id NUMBER,  id\_sectie CONSTRAINT fk\_camere\_sectie REFERENCES sectie\_ras(id) ON DELETE SET NULL,  id\_etaj CONSTRAINT fk\_camere\_etaj REFERENCES etaj\_ras(id) ON DELETE CASCADE,  CONSTRAINT pk\_camere PRIMARY KEY (id, id\_etaj));    CREATE TABLE istoric\_medical\_ras  (id NUMBER PRIMARY KEY,  id\_pacient CONSTRAINT fk\_istoric\_pacient REFERENCES pacienti\_ras(id) ON DELETE CASCADE,  afectiune VARCHAR2(50),  status VARCHAR2(30));  ALTER TABLE proceduri\_ras  DROP COLUMN durata;  ALTER TABLE proceduri\_ras  ADD (durata VARCHAR2(10)); |

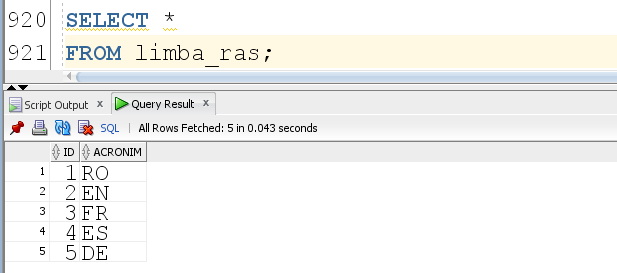


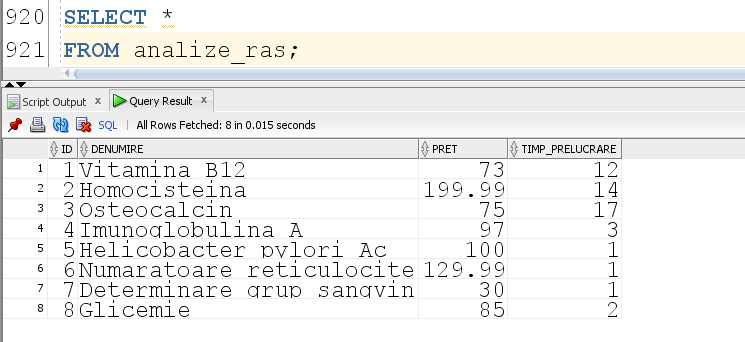


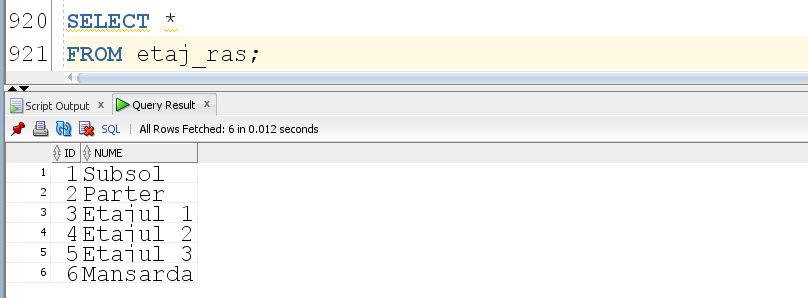
### Popularea bazei de date

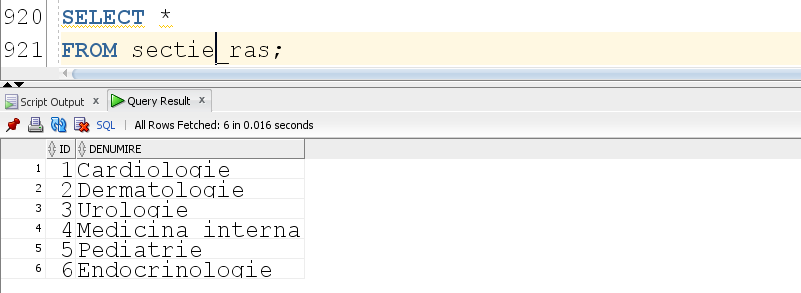
|  |
| --- |
| INSERT INTO calificare\_ras  VALUES (1, 'rezident');  INSERT INTO calificare\_ras  VALUES (2, 'specialist');  INSERT INTO calificare\_ras  VALUES (3, 'primar');  INSERT INTO calificare\_ras  VALUES (4, 'generalist');  INSERT INTO calificare\_ras  VALUES (5, 'fizioterapeut');  ---------------------------------------------------------------------  INSERT INTO limba\_ras  VALUES (1, 'RO');  INSERT INTO limba\_ras  VALUES (2, 'EN');  INSERT INTO limba\_ras  VALUES (3, 'FR');  INSERT INTO limba\_ras  VALUES (4, 'ES');  INSERT INTO limba\_ras  VALUES (5, 'DE');  --------------------------------------------------  INSERT INTO analize\_ras  VALUES (1, 'Vitamina B12', 73.0, 12);  INSERT INTO analize\_ras  VALUES (2, 'Homocisteina', 199.99, 14);  INSERT INTO analize\_ras  VALUES (3, 'Osteocalcin', 75.0, 17);  INSERT INTO analize\_ras  VALUES (4, 'Imunoglobulina A', 97.0, 3);  INSERT INTO analize\_ras  VALUES (5, 'Helicobacter pylori Ac', 100.0, 1);  INSERT INTO analize\_ras  VALUES (6, 'Numaratoare reticulocite', 129.99, 1);  INSERT INTO analize\_ras  VALUES (7, 'Determinare grup sangvin', 30.0, 1);  INSERT INTO analize\_ras  VALUES (8, 'Glicemie', 85, 2);  ---------------------------------------------------------------------  INSERT INTO etaj\_ras  VALUES (1, 'Subsol');  INSERT INTO etaj\_ras  VALUES (2, 'Parter');  INSERT INTO etaj\_ras  VALUES (3, 'Etajul 1');  INSERT INTO etaj\_ras  VALUES (4, 'Etajul 2');  INSERT INTO etaj\_ras  VALUES (5, 'Etajul 3');  INSERT INTO etaj\_ras  VALUES (6, 'Mansarda');  --------------------------------------------  INSERT INTO sectie\_ras  VALUES (1, 'Cardiologie');  INSERT INTO sectie\_ras  VALUES (2, 'Dermatologie');  INSERT INTO sectie\_ras  VALUES (3, 'Urologie');  INSERT INTO sectie\_ras  VALUES (4, 'Medicina interna');  INSERT INTO sectie\_ras  VALUES (5, 'Pediatrie');  INSERT INTO sectie\_ras  VALUES (6, 'Endocrinologie');  -----------------------------------------------------  INSERT INTO proceduri\_ras  VALUES (1, 'Hollywood Carbon Peel', 2, 1000, '20 min');  INSERT INTO proceduri\_ras  VALUES (2, 'Dermapen 4', 2, 1150, '35 min');  INSERT INTO proceduri\_ras  VALUES (3, 'Vasectomie', 3, 5500, '1 zi');  INSERT INTO proceduri\_ras  VALUES (4, 'Cistoscopie diagnostica', 3, 1000, '6 ore');  INSERT INTO proceduri\_ras  VALUES (5, 'Transplant incrucisat de rinichi', 3, 66000, '15 zile');  INSERT INTO proceduri\_ras  VALUES (6, 'Coronarografie', 1, 4000, '24 ore');  INSERT INTO proceduri\_ras  VALUES (7, ' Implantare de stimulator cardiac (pacemaker)', 1, 25000, '3 zile');  INSERT INTO proceduri\_ras  VALUES (8, 'Colonoscopie', 4, 450, '60 min');  INSERT INTO proceduri\_ras  VALUES (9, 'Punctie lombara', 5, 600, '2 zile');  INSERT INTO proceduri\_ras  VALUES (10, 'Bronhoscopie', 5, 1300, '1 zi');  INSERT INTO proceduri\_ras  VALUES (11, 'Toracenteză', 5, 500, '1 zi');  INSERT INTO proceduri\_ras  VALUES (12, 'Test de stimulare cu TRH', 6, 450, '3 ore');  INSERT INTO proceduri\_ras  VALUES (13, 'Biopsie tiroidiană cu ac fin', 6, 450, '30 min');  -----------------------------------------------------------  CREATE SEQUENCE secv\_id\_angajat  START WITH 1  INCREMENT BY 1  MINVALUE 1  MAXVALUE 999999  NOCYCLE  NOCACHE;  ---------------------------------------------------------  -- medici  INSERT INTO angajati\_ras  VALUES (secv\_id\_angajat.NEXTVAL, 'Popescu', 'Miruna', TO\_DATE('1974/12/23', 'yyyy/mm/dd'), '0702703707', TO\_DATE('2002/06/01', 'yyyy/mm/dd'), 12500);  INSERT INTO angajati\_ras  VALUES (secv\_id\_angajat.NEXTVAL, 'Tacu', 'Oleg', TO\_DATE('1982/05/21', 'yyyy/mm/dd'), '0795121428', TO\_DATE('2010/11/19', 'yyyy/mm/dd'), 10000);  INSERT INTO angajati\_ras  VALUES (secv\_id\_angajat.NEXTVAL, 'Popescu', 'Diana', TO\_DATE('1980/02/12', 'yyyy/mm/dd'), '0754753426', TO\_DATE('2010/10/21', 'yyyy/mm/dd'), 12500);  INSERT INTO angajati\_ras  VALUES (secv\_id\_angajat.NEXTVAL, 'Botez', 'Catalin', TO\_DATE('1988/02/10', 'yyyy/mm/dd'), '0721905168', TO\_DATE('2022/05/16', 'yyyy/mm/dd'), 9900);  INSERT INTO angajati\_ras  VALUES (secv\_id\_angajat.NEXTVAL, 'Carp', 'Lucia', TO\_DATE('1987/05/13', 'yyyy/mm/dd'), '0785795744', TO\_DATE('2022/05/18', 'yyyy/mm/dd'), 10000);  INSERT INTO angajati\_ras  VALUES (secv\_id\_angajat.NEXTVAL, 'Popa', 'Raluca', TO\_DATE('1982/07/17', 'yyyy/mm/dd'), '0710405579', TO\_DATE('2020/06/12', 'yyyy/mm/dd'), 11500);  INSERT INTO angajati\_ras  VALUES (secv\_id\_angajat.NEXTVAL, 'Munteanu', 'Romulus', TO\_DATE('1988/07/17', 'yyyy/mm/dd'), '0741068396', TO\_DATE('2022/05/11', 'yyyy/mm/dd'), 12500);  INSERT INTO angajati\_ras  VALUES (secv\_id\_angajat.NEXTVAL, 'Patrascu', 'Florin', TO\_DATE('1983/04/17', 'yyyy/mm/dd'), '0745196482', TO\_DATE('2022/04/13', 'yyyy/mm/dd'), 11000);  INSERT INTO angajati\_ras  VALUES (secv\_id\_angajat.NEXTVAL, 'Ochiana', 'Simona', TO\_DATE('1986/02/13', 'yyyy/mm/dd'), '0782587299', TO\_DATE('2022/11/02', 'yyyy/mm/dd'), 14000);  INSERT INTO angajati\_ras  VALUES (secv\_id\_angajat.NEXTVAL, 'Popa', 'Cezara', TO\_DATE('1990/09/21', 'yyyy/mm/dd'), '0751074467', TO\_DATE('2023/12/02', 'yyyy/mm/dd'), 14000);  INSERT INTO angajati\_ras  VALUES (secv\_id\_angajat.NEXTVAL, 'Castravet', 'Ion', TO\_DATE('1992/02/12', 'yyyy/mm/dd'), '0751281457', TO\_DATE('2024/05/12', 'yyyy/mm/dd'), 17000);  INSERT INTO angajati\_ras  VALUES (secv\_id\_angajat.NEXTVAL, 'Iordache', 'Flavius', TO\_DATE('1989/07/18', 'yyyy/mm/dd'), '0784632757', TO\_DATE('2024/05/12', 'yyyy/mm/dd'), 12500);  INSERT INTO angajati\_ras  VALUES (secv\_id\_angajat.NEXTVAL, 'Besliu', 'Geanina', TO\_DATE('1987/02/24', 'yyyy/mm/dd'), '0736332588', TO\_DATE('2023/06/13', 'yyyy/mm/dd'), 15000);  INSERT INTO angajati\_ras  VALUES (secv\_id\_angajat.NEXTVAL, 'Besliu', 'Madalina', TO\_DATE('1983/04/26', 'yyyy/mm/dd'), '0776298452', TO\_DATE('2022/04/10', 'yyyy/mm/dd'), 15000);  INSERT INTO angajati\_ras  VALUES (secv\_id\_angajat.NEXTVAL, 'Atanasoae', 'Ionut', TO\_DATE('1989/05/19', 'yyyy/mm/dd'), '0712559796', TO\_DATE('2021/03/12', 'yyyy/mm/dd'), 12500);  ------------------------------------------------------------------------------------------------------------------------------------------------------------------------  -- asistent medicali  INSERT INTO angajati\_ras  VALUES (secv\_id\_angajat.NEXTVAL, 'Bujor', 'Iuliana', TO\_DATE('1989/06/19', 'yyyy/mm/dd'), '0715214364', TO\_DATE('2021/03/12', 'yyyy/mm/dd'), 5000);  INSERT INTO angajati\_ras  VALUES (secv\_id\_angajat.NEXTVAL, 'Cojocaru', 'Camelia', TO\_DATE('1989/03/17', 'yyyy/mm/dd'), '0790361357', TO\_DATE('2020/05/09', 'yyyy/mm/dd'), 4800);  INSERT INTO angajati\_ras  VALUES (secv\_id\_angajat.NEXTVAL, 'Tomac', 'Iurie', TO\_DATE('1976/03/04', 'yyyy/mm/dd'), '0726634107', TO\_DATE('2008/09/01', 'yyyy/mm/dd'), 4800);  INSERT INTO angajati\_ras  VALUES (secv\_id\_angajat.NEXTVAL, 'Ilie', 'Marian', TO\_DATE('1989/02/14', 'yyyy/mm/dd'), '0731537535', TO\_DATE('2021/03/16', 'yyyy/mm/dd'), 6000);  INSERT INTO angajati\_ras  VALUES (secv\_id\_angajat.NEXTVAL, 'Lazar', 'Tudor', TO\_DATE('1987/10/08', 'yyyy/mm/dd'), '0755346116', TO\_DATE('2024/11/11', 'yyyy/mm/dd'), 6500);  INSERT INTO angajati\_ras  VALUES (secv\_id\_angajat.NEXTVAL, 'Luca', 'Alexandru', TO\_DATE('1991/10/09', 'yyyy/mm/dd'), '0741088619', TO\_DATE('2024/05/13', 'yyyy/mm/dd'), 5000);  INSERT INTO angajati\_ras  VALUES (secv\_id\_angajat.NEXTVAL, 'Maftei', 'Radu', TO\_DATE('1991/07/02', 'yyyy/mm/dd'), '0748058812', TO\_DATE('2023/02/23', 'yyyy/mm/dd'), 6800);  INSERT INTO angajati\_ras  VALUES (secv\_id\_angajat.NEXTVAL, 'Mogos', 'Raluca', TO\_DATE('1990/05/12', 'yyyy/mm/dd'), '0760668971', TO\_DATE('2022/12/22', 'yyyy/mm/dd'), 5400);  INSERT INTO angajati\_ras  VALUES (secv\_id\_angajat.NEXTVAL, 'Toma', 'Bogdan', TO\_DATE('1992/10/11', 'yyyy/mm/dd'), '0768992237', TO\_DATE('2021/10/11', 'yyyy/mm/dd'),5000);  INSERT INTO angajati\_ras  VALUES (secv\_id\_angajat.NEXTVAL, 'Rosu', 'Radu', TO\_DATE('1990/07/12', 'yyyy/mm/dd'), '0765982894', TO\_DATE('2022/11/21', 'yyyy/mm/dd'), 4800);  INSERT INTO angajati\_ras  VALUES (secv\_id\_angajat.NEXTVAL, 'Tanase', 'Adina', TO\_DATE('1992/08/02', 'yyyy/mm/dd'), '0728257769', TO\_DATE('2021/09/01', 'yyyy/mm/dd'), 4800);  INSERT INTO angajati\_ras  VALUES (secv\_id\_angajat.NEXTVAL, 'Boboc', 'Andreea', TO\_DATE('1987/03/12', 'yyyy/mm/dd'), '0718714101', TO\_DATE('2021/09/16', 'yyyy/mm/dd'), 5000);  INSERT INTO angajati\_ras  VALUES (secv\_id\_angajat.NEXTVAL, 'Toma', 'Loredana', TO\_DATE('1984/02/09', 'yyyy/mm/dd'), '0773060228', TO\_DATE('2022/10/21', 'yyyy/mm/dd'), 6000);  INSERT INTO angajati\_ras  VALUES (secv\_id\_angajat.NEXTVAL, 'Vieru', 'Razvan', TO\_DATE('1989/02/12', 'yyyy/mm/dd'), '0781905483', TO\_DATE('2022/08/30', 'yyyy/mm/dd'), 6000);  INSERT INTO angajati\_ras  VALUES (secv\_id\_angajat.NEXTVAL, 'Vasile', 'Cristina', TO\_DATE('1993/12/12', 'yyyy/mm/dd'), '0798360671', TO\_DATE('2022/03/02', 'yyyy/mm/dd'), 6000);  -------------------------------------------------------------------------------------------------------------------------------------------------------  -- personal nemedical  INSERT INTO angajati\_ras  VALUES (secv\_id\_angajat.NEXTVAL, 'Visli', 'Mariana', TO\_DATE('1994/11/22', 'yyyy/mm/dd'), '0775383475', TO\_DATE('2008/09/01', 'yyyy/mm/dd'), 3800);  INSERT INTO angajati\_ras  VALUES (secv\_id\_angajat.NEXTVAL, 'Acsinte', 'Vlad', TO\_DATE('1993/12/21', 'yyyy/mm/dd'), '0738800993', TO\_DATE('2010/01/10', 'yyyy/mm/dd'), 4000);  INSERT INTO angajati\_ras  VALUES (secv\_id\_angajat.NEXTVAL, 'Adam', 'Mihaela', TO\_DATE('1990/10/10', 'yyyy/mm/dd'), '0736908403', TO\_DATE('2023/07/22', 'yyyy/mm/dd'), 3800);  INSERT INTO angajati\_ras  VALUES (secv\_id\_angajat.NEXTVAL, 'Lupu', 'Ana', TO\_DATE('1989/03/08', 'yyyy/mm/dd'), '0736848609', TO\_DATE('2023/07/22', 'yyyy/mm/dd'), 3500);  INSERT INTO angajati\_ras  VALUES (secv\_id\_angajat.NEXTVAL, 'Alexa', 'Laura', TO\_DATE('1983/08/02', 'yyyy/mm/dd'), '0722053435', TO\_DATE('2019/08/02', 'yyyy/mm/dd'), 4000);  ---------------------------------------------------------------------------------------------------------------------------------------------------------------  -- personal administrativ  INSERT INTO angajati\_ras  VALUES (secv\_id\_angajat.NEXTVAL, 'Alexa', 'Rodica', TO\_DATE('1991/05/17', 'yyyy/mm/dd'), '0752491721', TO\_DATE('2018/10/11', 'yyyy/mm/dd'), 17000);  INSERT INTO angajati\_ras  VALUES (secv\_id\_angajat.NEXTVAL, 'Cozorici', 'Monica', TO\_DATE('1990/10/19', 'yyyy/mm/dd'), '0718452815', TO\_DATE('2024/03/01', 'yyyy/mm/dd'), 6300);  INSERT INTO angajati\_ras  VALUES (secv\_id\_angajat.NEXTVAL, 'Stan', 'Iuliana', TO\_DATE('1991/01/10', 'yyyy/mm/dd'), '0756968017', TO\_DATE('2024/03/01', 'yyyy/mm/dd'), 10000);  INSERT INTO angajati\_ras  VALUES (secv\_id\_angajat.NEXTVAL, 'Banu', 'Corina', TO\_DATE('1990/01/11', 'yyyy/mm/dd'), '0798400266', TO\_DATE('2020/10/21', 'yyyy/mm/dd'), 7800);  INSERT INTO angajati\_ras  VALUES (secv\_id\_angajat.NEXTVAL, 'Berea', 'Grigore', TO\_DATE('1991/09/04', 'yyyy/mm/dd'), '0755641392', TO\_DATE('2021/08/17', 'yyyy/mm/dd'), 9900);  ------------------------------------------------------------------------------------  CREATE SEQUENCE secv\_id\_medic  START WITH 1  INCREMENT BY 1  MINVALUE 1  MAXVALUE 999999  NOCYCLE  NOCACHE;  INSERT INTO medici\_ras  VALUES (secv\_id\_medic.NEXTVAL, 3, 4);  INSERT INTO medici\_ras  VALUES (secv\_id\_medic.NEXTVAL, 2, 4);  INSERT INTO medici\_ras  VALUES (secv\_id\_medic.NEXTVAL, 3, 1);  INSERT INTO medici\_ras  VALUES (secv\_id\_medic.NEXTVAL, 1, 1);  INSERT INTO medici\_ras  VALUES (secv\_id\_medic.NEXTVAL, 2, 3);  INSERT INTO medici\_ras  VALUES (secv\_id\_medic.NEXTVAL, 3, 3);  INSERT INTO medici\_ras  VALUES (secv\_id\_medic.NEXTVAL, 3, 3);  INSERT INTO medici\_ras  VALUES (secv\_id\_medic.NEXTVAL, 2, 2);  INSERT INTO medici\_ras  VALUES (secv\_id\_medic.NEXTVAL, 3, 2);  INSERT INTO medici\_ras  VALUES (secv\_id\_medic.NEXTVAL, 3, 5);  INSERT INTO medici\_ras  VALUES (secv\_id\_medic.NEXTVAL, 3, 5);  INSERT INTO medici\_ras  VALUES (secv\_id\_medic.NEXTVAL, 3, 6);  INSERT INTO medici\_ras  VALUES (secv\_id\_medic.NEXTVAL, 4, 1);  INSERT INTO medici\_ras  VALUES (secv\_id\_medic.NEXTVAL, 3, 4);  INSERT INTO medici\_ras  VALUES (secv\_id\_medic.NEXTVAL, 3, 1);  -----------------------------------------------------------------------------  CREATE SEQUENCE secv\_id\_asis  START WITH 16  INCREMENT BY 1  MINVALUE 16  MAXVALUE 30  NOCYCLE  NOCACHE;  INSERT INTO asistenti\_medicali\_ras  VALUES (secv\_id\_asis.NEXTVAL, 1);  INSERT INTO asistenti\_medicali\_ras  VALUES (secv\_id\_asis.NEXTVAL, 1);  INSERT INTO asistenti\_medicali\_ras  VALUES (secv\_id\_asis.NEXTVAL, 1);  INSERT INTO asistenti\_medicali\_ras  VALUES (secv\_id\_asis.NEXTVAL, 1);  INSERT INTO asistenti\_medicali\_ras  VALUES (secv\_id\_asis.NEXTVAL, 2);  INSERT INTO asistenti\_medicali\_ras  VALUES (secv\_id\_asis.NEXTVAL, 2);  INSERT INTO asistenti\_medicali\_ras  VALUES (secv\_id\_asis.NEXTVAL, 3);  INSERT INTO asistenti\_medicali\_ras  VALUES (secv\_id\_asis.NEXTVAL, 3);  INSERT INTO asistenti\_medicali\_ras  VALUES (secv\_id\_asis.NEXTVAL, 3);  INSERT INTO asistenti\_medicali\_ras  VALUES (secv\_id\_asis.NEXTVAL, 4);  INSERT INTO asistenti\_medicali\_ras  VALUES (secv\_id\_asis.NEXTVAL, 4);  INSERT INTO asistenti\_medicali\_ras  VALUES (secv\_id\_asis.NEXTVAL, 4);  INSERT INTO asistenti\_medicali\_ras  VALUES (secv\_id\_asis.NEXTVAL, 5);  INSERT INTO asistenti\_medicali\_ras  VALUES (secv\_id\_asis.NEXTVAL, 5);  INSERT INTO asistenti\_medicali\_ras  VALUES (secv\_id\_asis.NEXTVAL, 6);  ------------------------------------------------------------------------  CREATE SEQUENCE secv\_id\_nemed  START WITH 31  INCREMENT BY 1  MINVALUE 31  MAXVALUE 35  NOCYCLE  NOCACHE;  INSERT INTO personal\_nemedical\_ras  VALUES (secv\_id\_nemed.NEXTVAL);  INSERT INTO personal\_nemedical\_ras  VALUES (secv\_id\_nemed.NEXTVAL);  INSERT INTO personal\_nemedical\_ras  VALUES (secv\_id\_nemed.NEXTVAL);  INSERT INTO personal\_nemedical\_ras  VALUES (secv\_id\_nemed.NEXTVAL);  INSERT INTO personal\_nemedical\_ras  VALUES (secv\_id\_nemed.NEXTVAL);  -------------------------------------------------------------------------------------  CREATE SEQUENCE secv\_id\_adm  START WITH 36  INCREMENT BY 1  MINVALUE 36  MAXVALUE 40  NOCYCLE  NOCACHE;  INSERT INTO personal\_administrativ\_ras  VALUES (secv\_id\_adm.NEXTVAL);  INSERT INTO personal\_administrativ\_ras  VALUES (secv\_id\_adm.NEXTVAL);  INSERT INTO personal\_administrativ\_ras  VALUES (secv\_id\_adm.NEXTVAL);  INSERT INTO personal\_administrativ\_ras  VALUES (secv\_id\_adm.NEXTVAL);  INSERT INTO personal\_administrativ\_ras  VALUES (secv\_id\_adm.NEXTVAL);  ------------------------------------------------------------------------------  INSERT INTO camere\_ras  VALUES (1, 1, 1);  INSERT INTO camere\_ras  VALUES (2, 6, 1);  INSERT INTO camere\_ras  VALUES (3, 6, 1);  INSERT INTO camere\_ras  VALUES (1, 1, 2);  INSERT INTO camere\_ras  VALUES (2, 3, 2);  INSERT INTO camere\_ras  VALUES (3, 4, 2);  INSERT INTO camere\_ras  VALUES (1, 1, 3);  INSERT INTO camere\_ras  VALUES (2, 1, 3);  INSERT INTO camere\_ras  VALUES (3, 2, 3);  INSERT INTO camere\_ras  VALUES (1, 2, 4);  INSERT INTO camere\_ras  VALUES (2, 5, 4);  INSERT INTO camere\_ras  VALUES (3, 5, 4);  INSERT INTO camere\_ras  VALUES (1, 3, 5);  INSERT INTO camere\_ras  VALUES (2, 3, 5);  INSERT INTO camere\_ras  VALUES (3, 4, 5);  INSERT INTO camere\_ras  VALUES (1, 3, 6);  INSERT INTO camere\_ras  VALUES (2, 1, 6);  INSERT INTO camere\_ras  VALUES (3, 2, 6);  -----------------------------------------------------------------------------------------  INSERT INTO personal\_resp\_etaj\_ras  VALUES (31, 1, TO\_DATE('2023/10/23', 'yyyy/mm/dd'));  INSERT INTO personal\_resp\_etaj\_ras  VALUES (31, 4, TO\_DATE('2024/12/13', 'yyyy/mm/dd'));  INSERT INTO personal\_resp\_etaj\_ras  VALUES (31, 6, TO\_DATE('2024/03/04', 'yyyy/mm/dd'));  INSERT INTO personal\_resp\_etaj\_ras  VALUES (32, 2, TO\_DATE('2022/05/17', 'yyyy/mm/dd'));  INSERT INTO personal\_resp\_etaj\_ras  VALUES (32, 3, TO\_DATE('2021/02/21', 'yyyy/mm/dd'));  INSERT INTO personal\_resp\_etaj\_ras  VALUES (33, 3, TO\_DATE('2024/08/09', 'yyyy/mm/dd'));  INSERT INTO personal\_resp\_etaj\_ras  VALUES (33, 4, TO\_DATE('2024/09/03', 'yyyy/mm/dd'));  INSERT INTO personal\_resp\_etaj\_ras  VALUES (33, 5, TO\_DATE('2024/10/25', 'yyyy/mm/dd'));  INSERT INTO personal\_resp\_etaj\_ras  VALUES (34, 2, TO\_DATE('2024/08/02', 'yyyy/mm/dd'));  INSERT INTO personal\_resp\_etaj\_ras  VALUES (34, 4, TO\_DATE('2024/11/26', 'yyyy/mm/dd'));  INSERT INTO personal\_resp\_etaj\_ras  VALUES (34, 1, TO\_DATE('2024/09/22', 'yyyy/mm/dd'));  INSERT INTO personal\_resp\_etaj\_ras  VALUES (35, 3, TO\_DATE('2023/10/22', 'yyyy/mm/dd'));  INSERT INTO personal\_resp\_etaj\_ras  VALUES (35, 2, TO\_DATE('2023/07/02', 'yyyy/mm/dd'));  INSERT INTO personal\_resp\_etaj\_ras  VALUES (35, 5, TO\_DATE('2024/09/22', 'yyyy/mm/dd'));  ---------------------------------------------------------------------------------  INSERT INTO pacienti\_ras  VALUES (1, '5241226155713', 'Sava', 'Ion', TO\_DATE('2000/07/02', 'yyyy/mm/dd'), '0722568199');  INSERT INTO pacienti\_ras  VALUES (2, '5241226152670', 'Preda', 'Maria', TO\_DATE('2012/04/22', 'yyyy/mm/dd'), '0750232000');  INSERT INTO pacienti\_ras  VALUES (3, '5170326156302', 'Enache', 'Ana', TO\_DATE('1987/11/24', 'yyyy/mm/dd'), '0721886095');  INSERT INTO pacienti\_ras  VALUES (4, '5170326155650', 'Pop', 'Alina', TO\_DATE('1995/04/19', 'yyyy/mm/dd'), '0723355309');  INSERT INTO pacienti\_ras  VALUES (5, '5170326157865', 'Pop', 'Alex', TO\_DATE('2000/08/28', 'yyyy/mm/dd'), '0745337969');  INSERT INTO pacienti\_ras  VALUES (6, '5170326156322', 'Pop', 'Alex', TO\_DATE('1997/12/14', 'yyyy/mm/dd'), '0753662812');  INSERT INTO pacienti\_ras  VALUES (7, '5170826156322', 'Popa', 'Miruna', TO\_DATE('1999/09/23', 'yyyy/mm/dd'), '0758962812');  -----------------------------------------------------------------------------------------------------------  INSERT INTO pacienti\_aleg\_analize\_ras  VALUES (1, 1, TO\_DATE('2024/12/23', 'yyyy/mm/dd'));  INSERT INTO pacienti\_aleg\_analize\_ras  VALUES (1, 3, TO\_DATE('2024/12/23', 'yyyy/mm/dd'));  INSERT INTO pacienti\_aleg\_analize\_ras  VALUES (1, 5, TO\_DATE('2023/08/20', 'yyyy/mm/dd'));  INSERT INTO pacienti\_aleg\_analize\_ras  VALUES (1, 7, TO\_DATE('2022/10/29', 'yyyy/mm/dd'));  INSERT INTO pacienti\_aleg\_analize\_ras  VALUES (2, 2, TO\_DATE('2024/10/14', 'yyyy/mm/dd'));  INSERT INTO pacienti\_aleg\_analize\_ras  VALUES (2, 4, TO\_DATE('2024/10/14', 'yyyy/mm/dd'));  INSERT INTO pacienti\_aleg\_analize\_ras  VALUES (2, 6, TO\_DATE('2024/10/14', 'yyyy/mm/dd'));  INSERT INTO pacienti\_aleg\_analize\_ras  VALUES (2, 2, TO\_DATE('2024/12/14', 'yyyy/mm/dd'));  INSERT INTO pacienti\_aleg\_analize\_ras  VALUES (2, 4, TO\_DATE('2024/12/14', 'yyyy/mm/dd'));  INSERT INTO pacienti\_aleg\_analize\_ras  VALUES (2, 6, TO\_DATE('2024/12/14', 'yyyy/mm/dd'));  INSERT INTO pacienti\_aleg\_analize\_ras  VALUES (2, 2, TO\_DATE('2023/03/09', 'yyyy/mm/dd'));  INSERT INTO pacienti\_aleg\_analize\_ras  VALUES (2, 4, TO\_DATE('2023/02/08', 'yyyy/mm/dd'));  INSERT INTO pacienti\_aleg\_analize\_ras  VALUES (2, 6, TO\_DATE('2023/04/12', 'yyyy/mm/dd'));  INSERT INTO pacienti\_aleg\_analize\_ras  VALUES (3, 7, TO\_DATE('2024/11/04', 'yyyy/mm/dd'));  INSERT INTO pacienti\_aleg\_analize\_ras  VALUES (3, 3, TO\_DATE('2024/11/04', 'yyyy/mm/dd'));  INSERT INTO pacienti\_aleg\_analize\_ras  VALUES (3, 6, TO\_DATE('2024/08/12', 'yyyy/mm/dd'));  INSERT INTO pacienti\_aleg\_analize\_ras  VALUES (4, 1, TO\_DATE('2022/07/10', 'yyyy/mm/dd'));  INSERT INTO pacienti\_aleg\_analize\_ras  VALUES (4, 1, TO\_DATE('2024/03/22', 'yyyy/mm/dd'));  INSERT INTO pacienti\_aleg\_analize\_ras  VALUES (4, 6, TO\_DATE('2024/12/28', 'yyyy/mm/dd'));  INSERT INTO pacienti\_aleg\_analize\_ras  VALUES (4, 4, TO\_DATE('2024/08/07', 'yyyy/mm/dd'));  -------------------------------------------------------------------------------------------------  INSERT INTO pacienti\_vorbesc\_limbi\_ras  VALUES (1, 1, 10);  INSERT INTO pacienti\_vorbesc\_limbi\_ras  VALUES (1, 2, 8);  INSERT INTO pacienti\_vorbesc\_limbi\_ras  VALUES (1, 3, 7);  INSERT INTO pacienti\_vorbesc\_limbi\_ras  VALUES (2, 1, 10);  INSERT INTO pacienti\_vorbesc\_limbi\_ras  VALUES (2, 5, 4);  INSERT INTO pacienti\_vorbesc\_limbi\_ras  VALUES (2, 4, 8);  INSERT INTO pacienti\_vorbesc\_limbi\_ras  VALUES (3, 3, 7);  INSERT INTO pacienti\_vorbesc\_limbi\_ras  VALUES (3, 2, 3);  INSERT INTO pacienti\_vorbesc\_limbi\_ras  VALUES (3, 4, 9);  INSERT INTO pacienti\_vorbesc\_limbi\_ras  VALUES (4, 1, 6);  INSERT INTO pacienti\_vorbesc\_limbi\_ras  VALUES (4, 2, 10);  INSERT INTO pacienti\_vorbesc\_limbi\_ras  VALUES (4, 5, 4);  INSERT INTO pacienti\_vorbesc\_limbi\_ras  VALUES (5, 5, 10);  INSERT INTO pacienti\_vorbesc\_limbi\_ras  VALUES (5, 1, 2);  ----------------------------------------------------------------------------------------  INSERT INTO istoric\_medical\_ras  VALUES (1, 1, 'amigdalita', 'tratat');  INSERT INTO istoric\_medical\_ras  VALUES (2, 1, 'anemie', 'sub observatie');  INSERT INTO istoric\_medical\_ras  VALUES (3, 1, 'apnee', 'sub observatie');  INSERT INTO istoric\_medical\_ras  VALUES (4, 2, 'apendicita', 'tratat');  INSERT INTO istoric\_medical\_ras  VALUES (5, 3, 'dermatita seboreica', 'sub observatie');  INSERT INTO istoric\_medical\_ras  VALUES (6, 3, 'acnee vulgara', 'tratat');  INSERT INTO istoric\_medical\_ras  VALUES (7, 5, 'raceala', 'tratat');  ----------------------------------------------------------------------------------  DECLARE  i NUMBER := 1;  BEGIN  LOOP  INSERT INTO angajati\_vorbesc\_limbi\_ras  VALUES (i, 1, 10);  i := i + 1;  EXIT WHEN i > 40;  END LOOP;  END;  /  DECLARE  i NUMBER := 1;  BEGIN  LOOP  INSERT INTO angajati\_vorbesc\_limbi\_ras  VALUES (i, 2, 9);  i := i + 3;  EXIT WHEN i > 40;  END LOOP;  END;  /  DECLARE  i NUMBER := 2;  BEGIN  LOOP  INSERT INTO angajati\_vorbesc\_limbi\_ras  VALUES (i, 2, 8);  i := i + 3;  EXIT WHEN i > 40;  END LOOP;  END;  /  DECLARE  i NUMBER := 3;  BEGIN  LOOP  INSERT INTO angajati\_vorbesc\_limbi\_ras  VALUES (i, 5, 9);  i := i + 3;  EXIT WHEN i > 40;  END LOOP;  END;  /  INSERT INTO angajati\_vorbesc\_limbi\_ras  VALUES (1, 5, 10);  INSERT INTO angajati\_vorbesc\_limbi\_ras  VALUES (36, 4, 10);  INSERT INTO angajati\_vorbesc\_limbi\_ras  VALUES (39, 4, 7);  --------------------------------------------------------------------------------------  INSERT INTO pacient\_medic\_proced\_ras  VALUES (1, 1, 6, 5, TO\_DATE('2024/08/17', 'yyyy/mm/dd'));  INSERT INTO pacient\_medic\_proced\_ras  VALUES (1, 1, 7, 5, TO\_DATE('2024/08/17', 'yyyy/mm/dd'));  INSERT INTO pacient\_medic\_proced\_ras  VALUES (1, 3, 6, 5, TO\_DATE('2024/08/17', 'yyyy/mm/dd'));  INSERT INTO pacient\_medic\_proced\_ras  VALUES (1, 3, 7, 5, TO\_DATE('2024/08/17', 'yyyy/mm/dd'));  INSERT INTO pacient\_medic\_proced\_ras  VALUES (1, 4, 6, 5, TO\_DATE('2024/08/17', 'yyyy/mm/dd'));  INSERT INTO pacient\_medic\_proced\_ras  VALUES (1, 4, 7, 5, TO\_DATE('2024/08/17', 'yyyy/mm/dd'));  INSERT INTO pacient\_medic\_proced\_ras  VALUES (1, 5, 6, 5, TO\_DATE('2024/08/17', 'yyyy/mm/dd'));  INSERT INTO pacient\_medic\_proced\_ras  VALUES (1, 5, 7, 5, TO\_DATE('2024/08/17', 'yyyy/mm/dd'));  INSERT INTO pacient\_medic\_proced\_ras  VALUES (2, 2, 9, 1, TO\_DATE('2024/11/03', 'yyyy/mm/dd'));  INSERT INTO pacient\_medic\_proced\_ras  VALUES (3, 3, 15, 7, TO\_DATE('2024/03/04', 'yyyy/mm/dd'));  INSERT INTO pacient\_medic\_proced\_ras  VALUES (4, 3, 3, 7, TO\_DATE('2024/03/04', 'yyyy/mm/dd'));  INSERT INTO pacient\_medic\_proced\_ras  VALUES (5, 3, 4, 7, TO\_DATE('2024/03/04', 'yyyy/mm/dd'));  INSERT INTO pacient\_medic\_proced\_ras  VALUES (6, 3, 13, 7, TO\_DATE('2024/03/04', 'yyyy/mm/dd'));  INSERT INTO pacient\_medic\_proced\_ras  VALUES (7, 3, 9, 2, SYSDATE);  INSERT INTO pacient\_medic\_proced\_ras  VALUES (7, 3, 9, 1, SYSDATE);  COMMIT; |

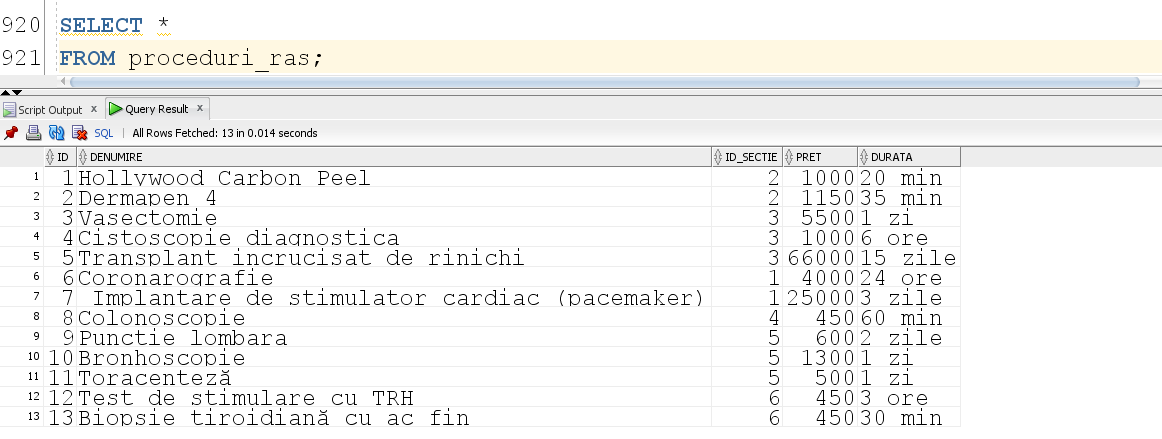


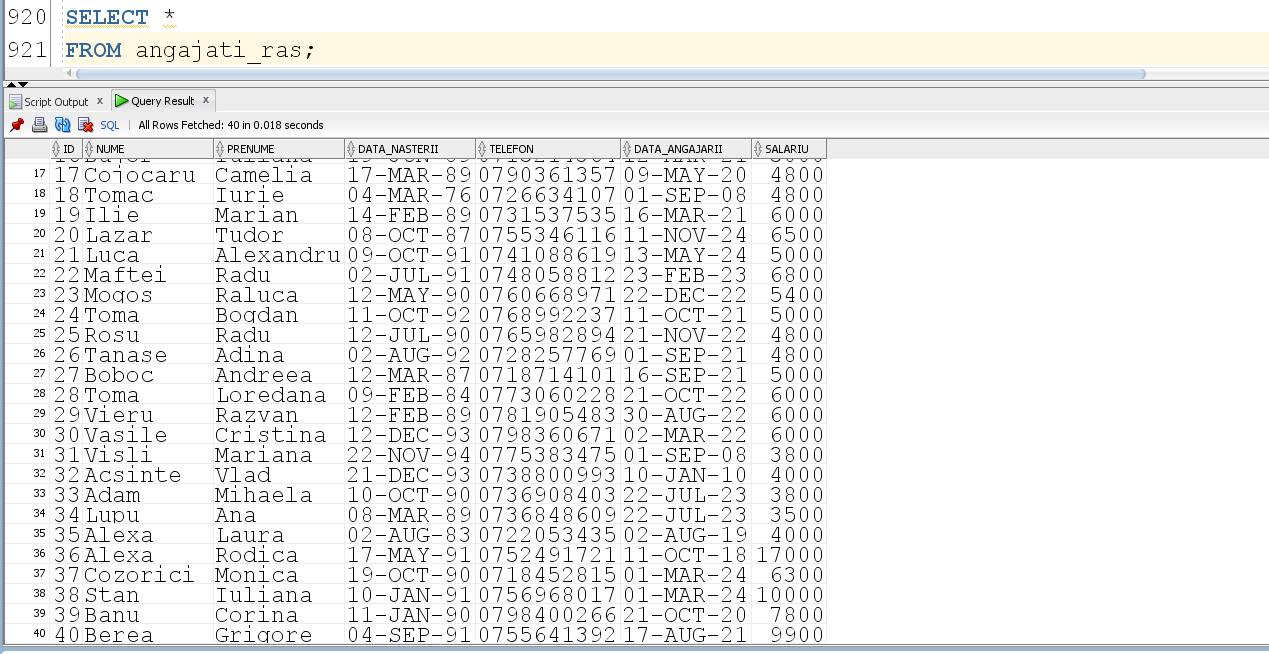


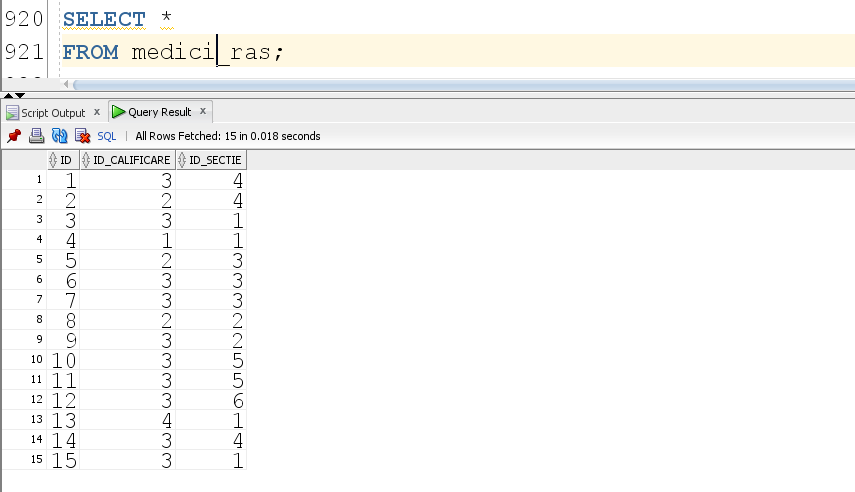


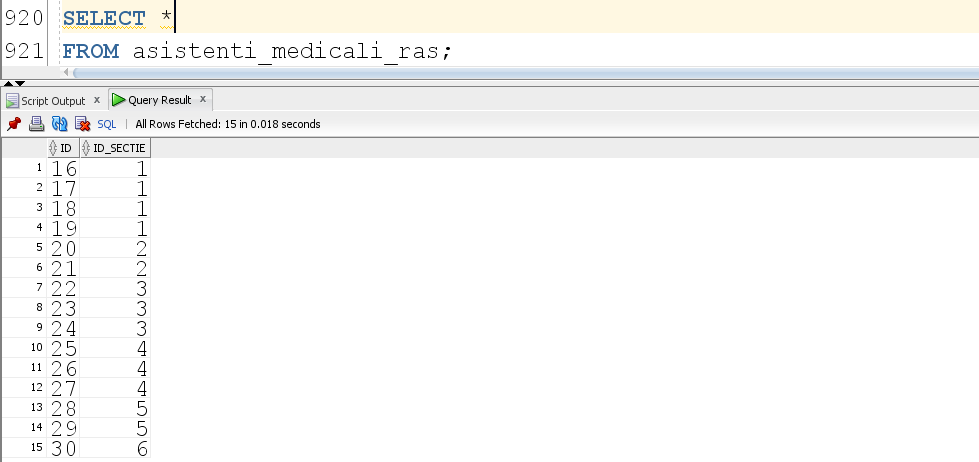


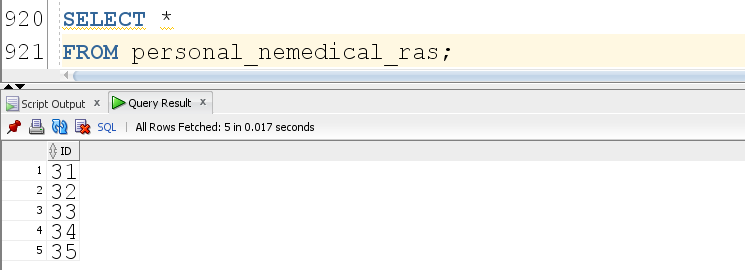


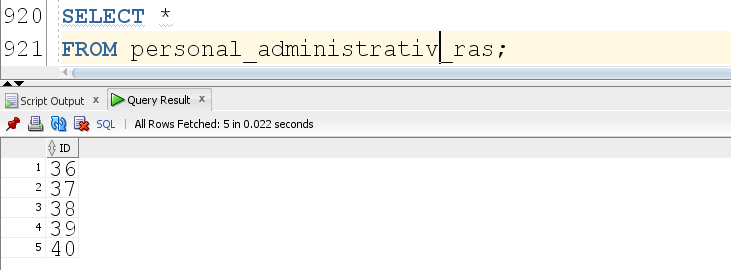


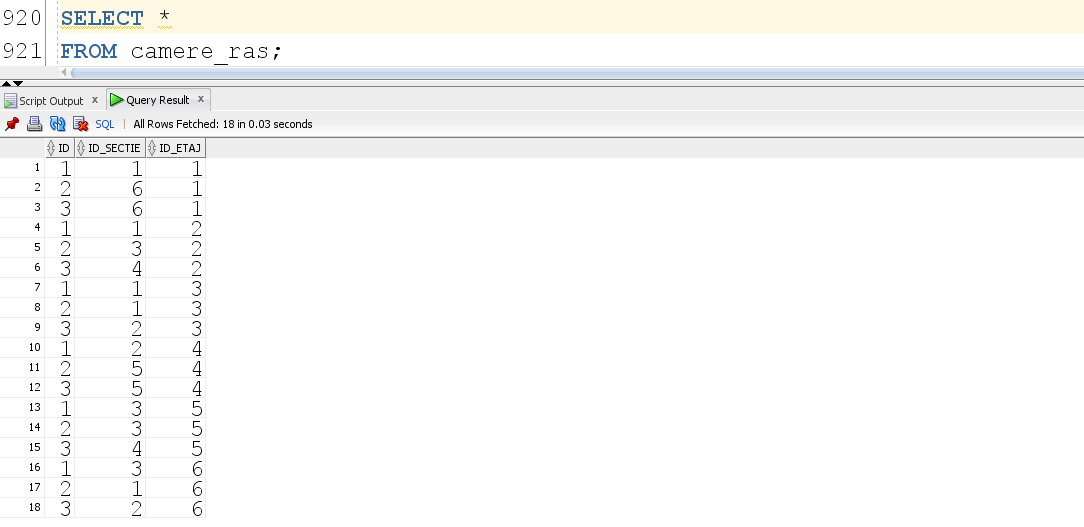


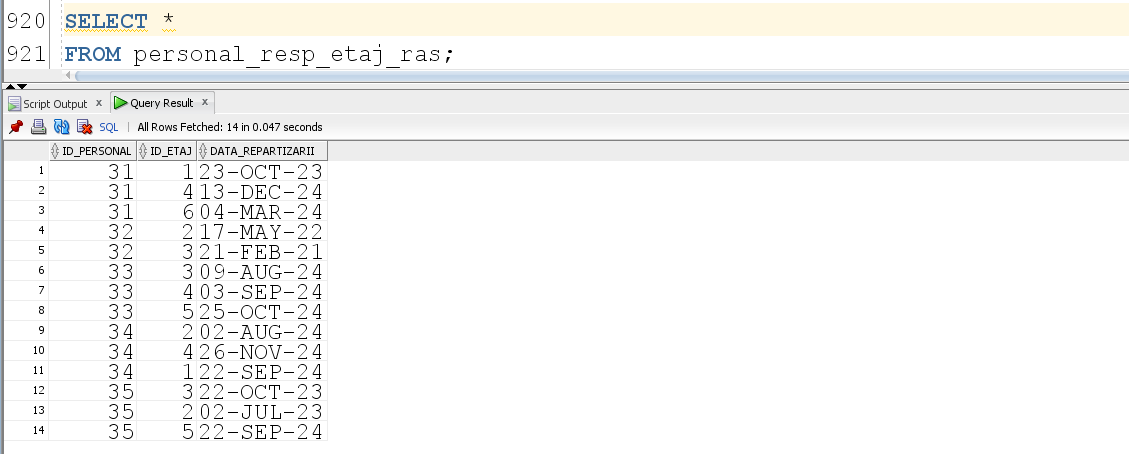


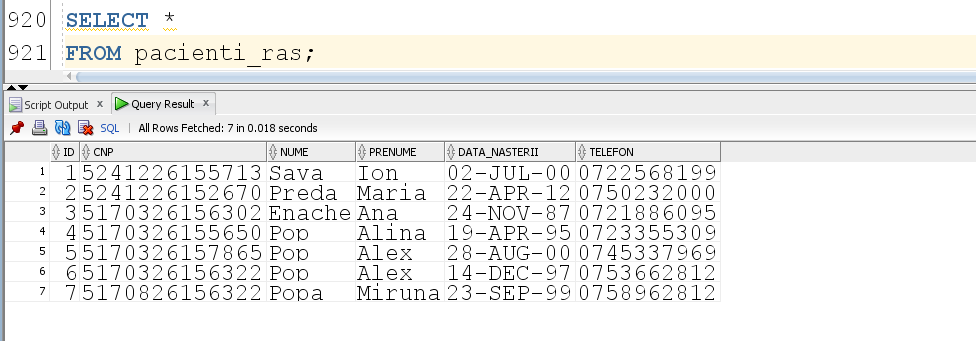


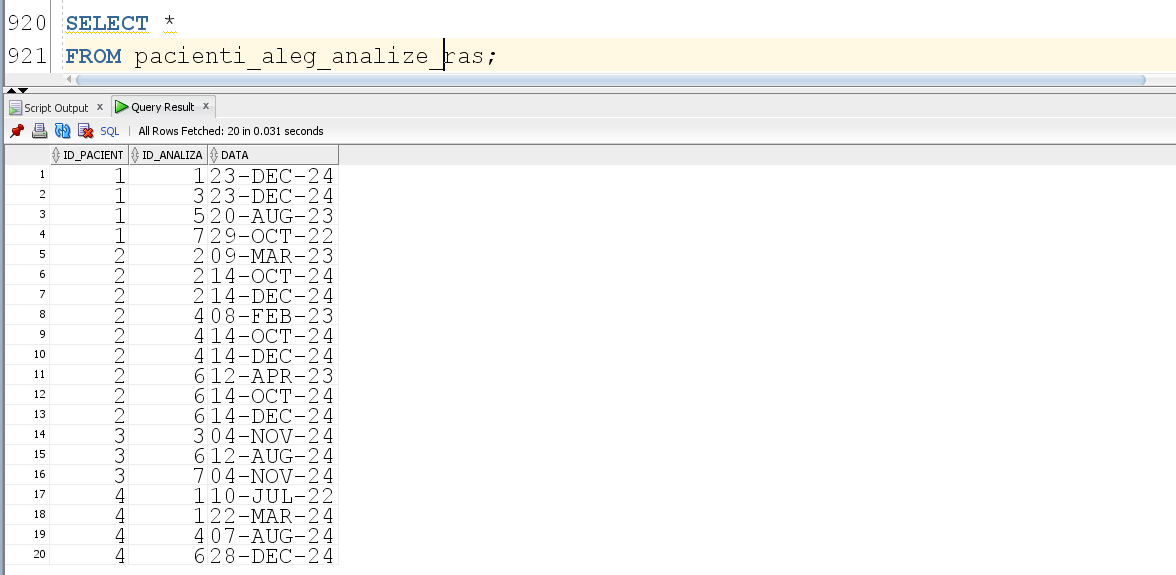


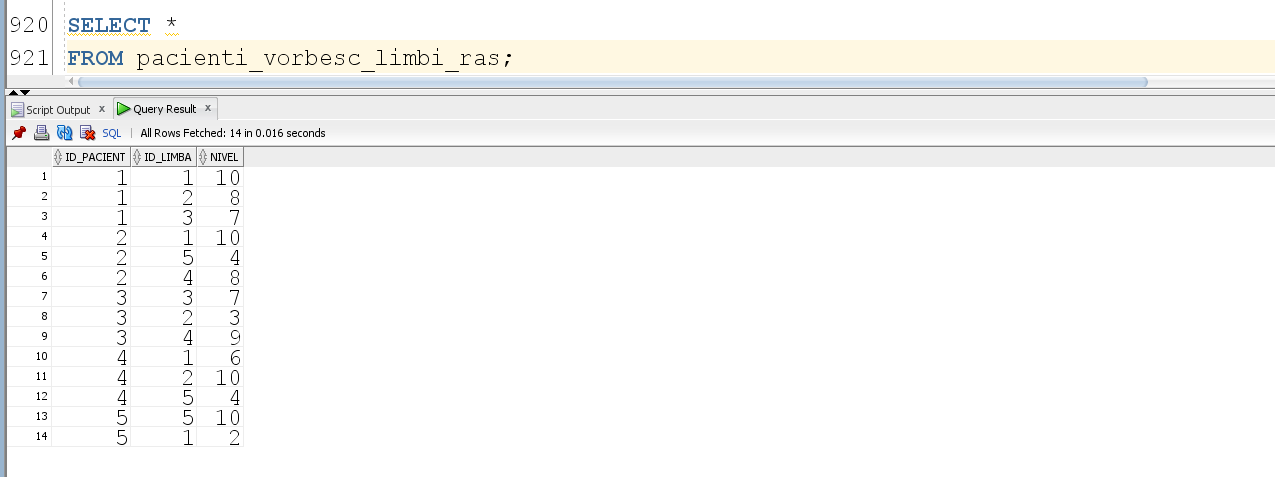


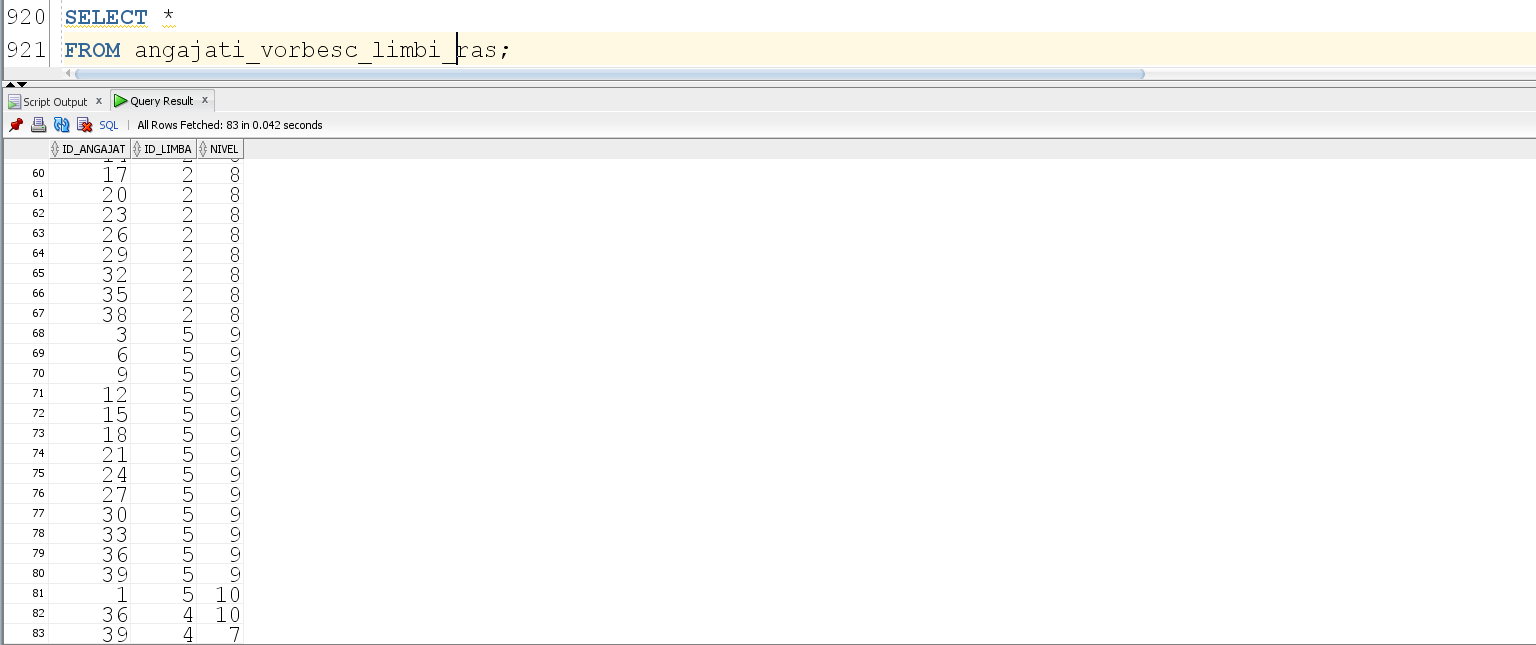


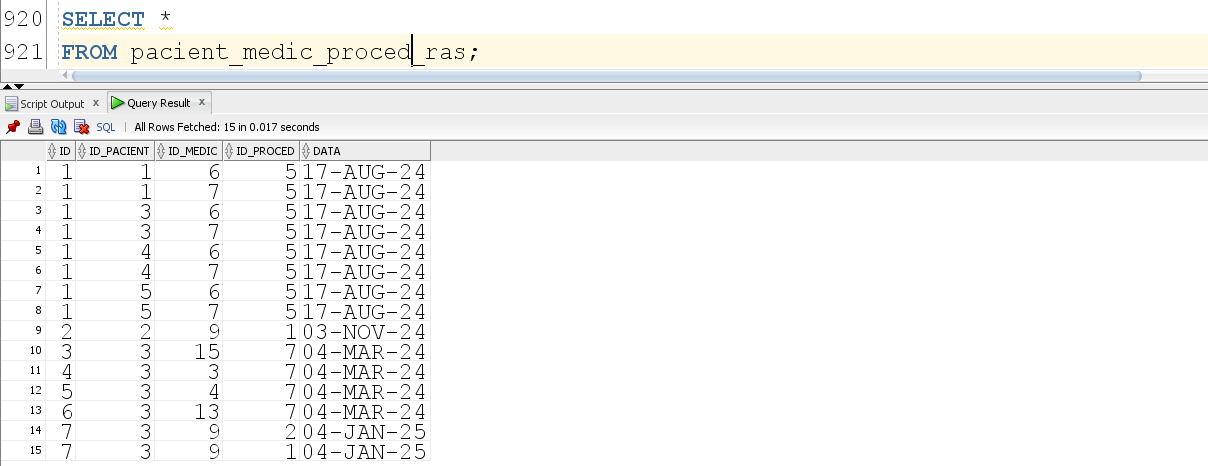










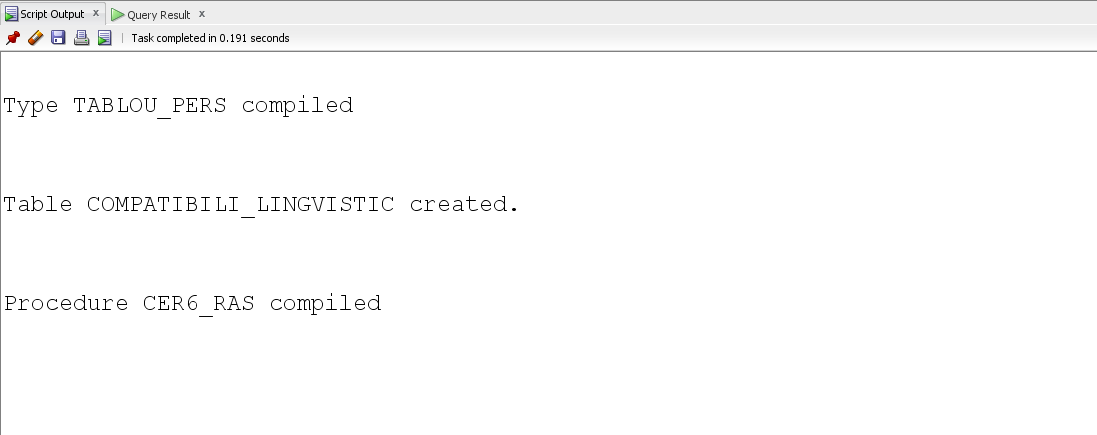


### Cerința 6

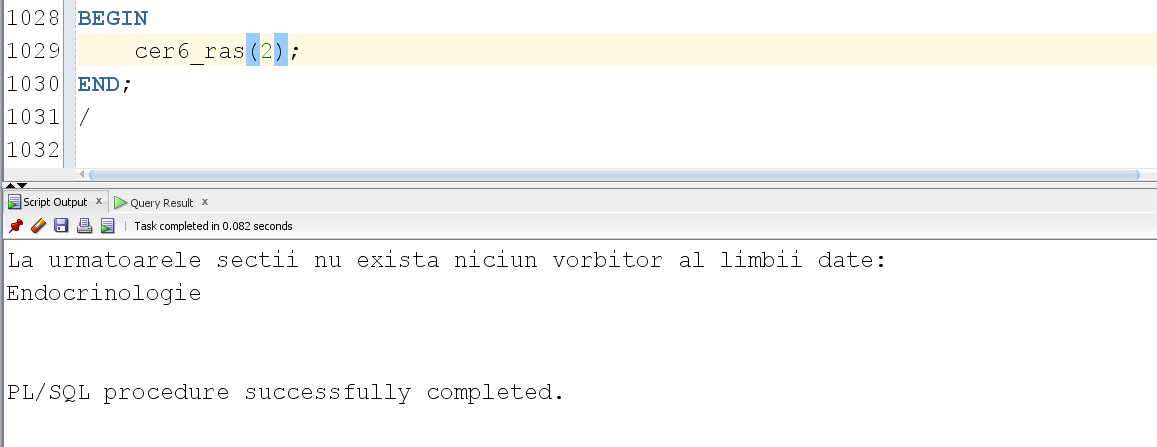
*Cerința*: Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat independent care să utilizeze toate cele 3 tipuri de colecții studiate. Apelați subprogramul.

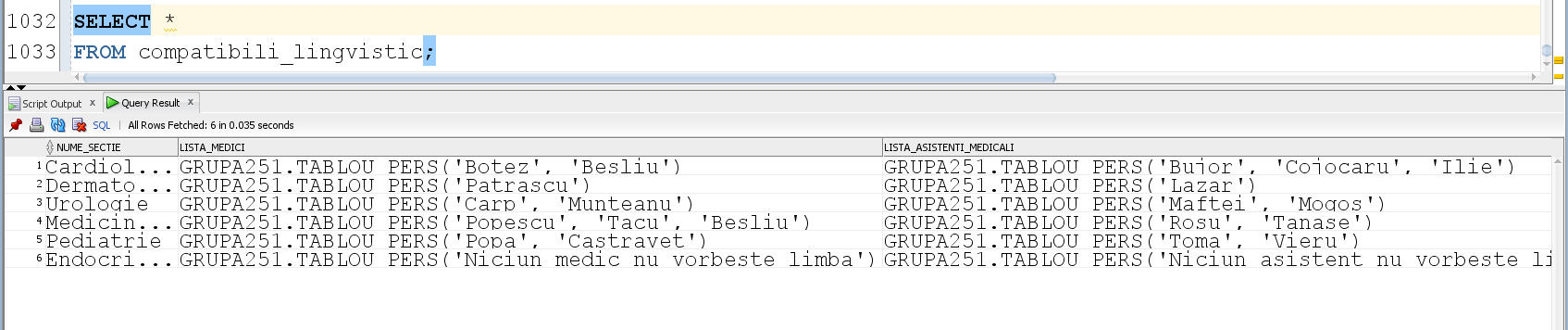
*Rezolvare*: Creați tabelul compatibili lingvistic cu următoarea structură: nume\_sectie, lista\_medici, lista\_asistenti\_medicali. Scrieți un subprogram stocat care, pe baza unui id al unei limbi, inserează în tabelul compatibili\_lingvistic pentru fiecare secție lista medicilor, respectiv a asistenților medicali (numele lor) care vorbesc limba cu id-ul dat ca parametru. În cazul în care pentru o secție nu există niciun medic / asistent medical care vorbește limba respectivă tabelul va conține, textul ‘Niciun medic / asistent nu vorbeste limba’, după caz. De asemenea, subprogramul va afișa lista secțiilor la care nu există niciun vorbitor (medic sau asistent medical) al limbii date.

|  |
| --- |
| SET SERVEROUTPUT ON;  CREATE OR REPLACE TYPE tablou\_pers IS TABLE OF VARCHAR2(100);  /  CREATE TABLE compatibili\_lingvistic  (nume\_sectie VARCHAR2(50),  lista\_medici tablou\_pers,  lista\_asistenti\_medicali tablou\_pers  )  NESTED TABLE lista\_medici STORE AS tabl\_medici  NESTED TABLE lista\_asistenti\_medicali STORE AS tabl\_asist;  /  CREATE OR REPLACE PROCEDURE cer6\_ras(v\_id limba\_ras.id%TYPE)  IS  TYPE tablou\_sectii IS TABLE OF sectie\_ras%ROWTYPE INDEX BY PLS\_INTEGER;  sectii tablou\_sectii;  medici tablou\_pers := tablou\_pers();  asistenti tablou\_pers := tablou\_pers();  niciun\_medic tablou\_pers := tablou\_pers();  niciun\_asist tablou\_pers := tablou\_pers();  ct\_medici NUMBER;  ct\_asist NUMBER;  TYPE niciun\_vorbitor IS VARRAY(10) OF sectie\_ras.denumire%TYPE;  sectii\_neacces niciun\_vorbitor := niciun\_vorbitor();  linie limba\_ras%ROWTYPE;  BEGIN  -- verificam daca codul dat ca parametru exista in baza de date  SELECT \*  INTO linie  FROM limba\_ras  WHERE id = v\_id;    niciun\_medic.EXTEND;  niciun\_medic(1) := 'Niciun medic nu vorbeste limba';    niciun\_asist.EXTEND;  niciun\_asist(1) := 'Niciun asistent nu vorbeste limba';  -- luam toate sectiile  SELECT \*  BULK COLLECT INTO sectii  FROM sectie\_ras;    -- pentru fiecare sectie luam medicii care vorbesc limba respectiva  FOR i IN 1..sectii.COUNT LOOP  -- numaram medicii de pe sectie care vorbesc limba data  SELECT COUNT(a.nume)  INTO ct\_medici  FROM medici\_ras m JOIN angajati\_vorbesc\_limbi\_ras ang ON (m.id = ang.id\_angajat)  JOIN angajati\_ras a ON (a.id = m.id)  WHERE m.id\_sectie = sectii(i).id AND ang.id\_limba = v\_id;    IF ct\_medici > 0 THEN  SELECT a.nume  BULK COLLECT INTO medici  FROM medici\_ras m JOIN angajati\_vorbesc\_limbi\_ras ang ON (m.id = ang.id\_angajat)  JOIN angajati\_ras a ON (a.id = m.id)  WHERE m.id\_sectie = sectii(i).id AND ang.id\_limba = v\_id;  ELSE  medici := niciun\_medic;  END IF;    -- numaram asistentii de pe sectie care vorbesc limba data  SELECT COUNT (a.nume)  INTO ct\_asist  FROM asistenti\_medicali\_ras asist JOIN angajati\_vorbesc\_limbi\_ras ang ON (asist.id = ang.id\_angajat)  JOIN angajati\_ras a ON (a.id = asist.id)  WHERE asist.id\_sectie = sectii(i).id AND ang.id\_limba = v\_id;    IF ct\_asist > 0 THEN  SELECT a.nume  BULK COLLECT INTO asistenti  FROM asistenti\_medicali\_ras asist JOIN angajati\_vorbesc\_limbi\_ras ang ON (asist.id = ang.id\_angajat)  JOIN angajati\_ras a ON (a.id = asist.id)  WHERE asist.id\_sectie = sectii(i).id AND ang.id\_limba = v\_id;  ELSE  asistenti := niciun\_asist;  END IF;    -- inseram in tabel  INSERT INTO compatibili\_lingvistic  VALUES (sectii(i).denumire, medici, asistenti);    IF ct\_medici = 0 AND ct\_asist = 0 THEN  sectii\_neacces.EXTEND;  sectii\_neacces(sectii\_neacces.LAST) := sectii(i).denumire;  END IF;    END LOOP;    -- daca exista sectii neaccesibile, le afisam  IF sectii\_neacces.COUNT > 0 THEN  DBMS\_OUTPUT.PUT\_LINE('La urmatoarele sectii nu exista niciun vorbitor al limbii date: ');  FOR i IN sectii\_neacces.FIRST..sectii\_neacces.LAST LOOP  DBMS\_OUTPUT.PUT\_LINE(sectii\_neacces(i) || ' ');  END LOOP;  END IF;  EXCEPTION  WHEN NO\_DATA\_FOUND THEN  RAISE\_APPLICATION\_ERROR(-20002,'Codul dat nu exista');  END cer6\_ras;  /  BEGIN  cer6\_ras(2);  END;  /  SELECT \*  FROM compatibili\_lingvistic;  -- stergem inserarile anterioare  DELETE FROM compatibili\_lingvistic;  BEGIN  cer6\_ras(3);  END;  /  SELECT \*  FROM compatibili\_lingvistic;  -- stergem inseratile anterioare  DELETE FROM compatibili\_lingvistic;  BEGIN  cer6\_ras(10);  END;  / |

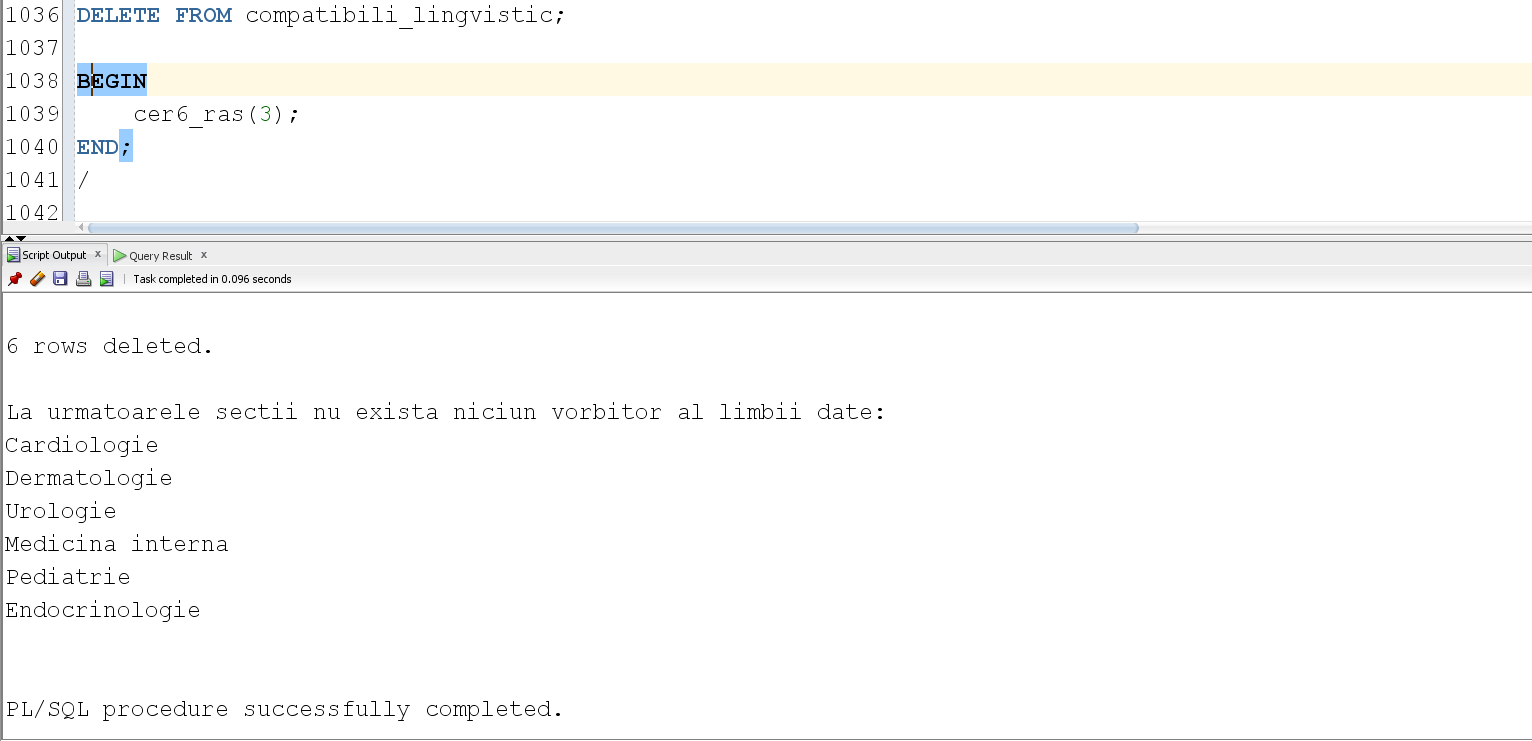
**

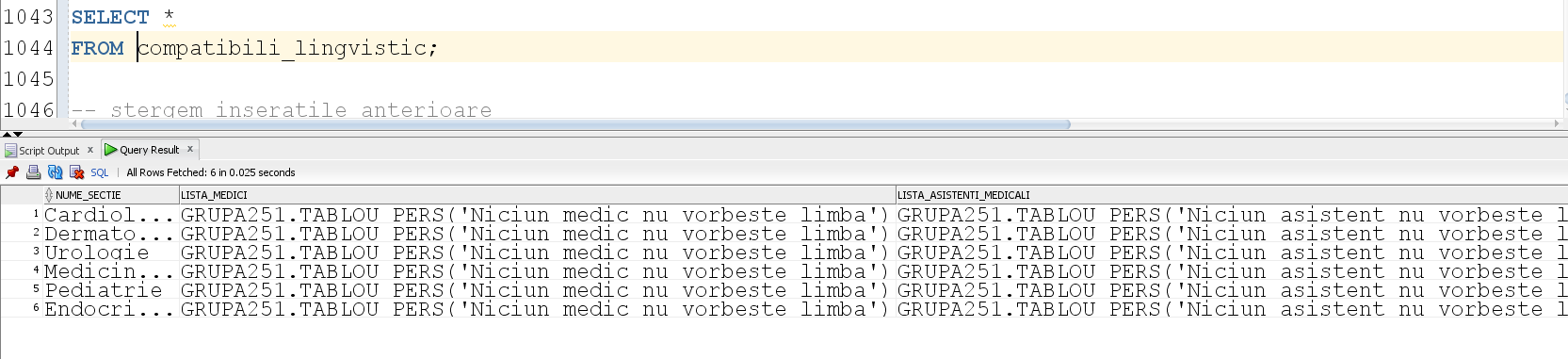
*Exemplul 1*



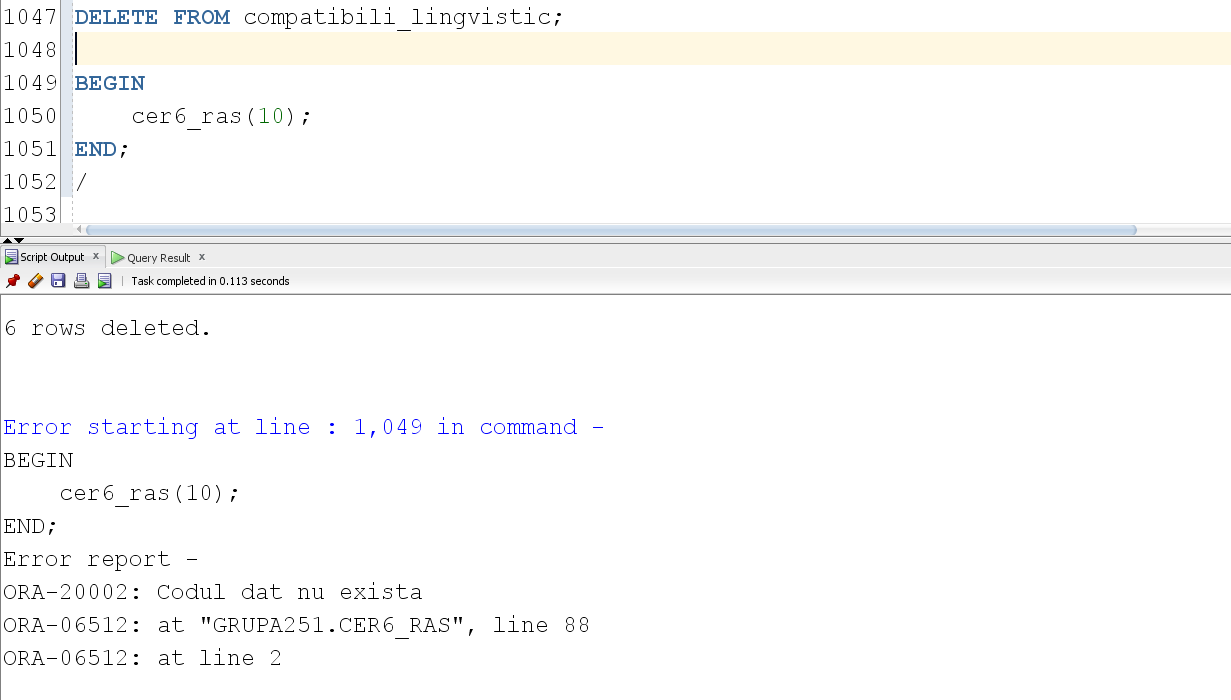


*Exemplul 2*





*Exemplul 3*



### Cerința 7

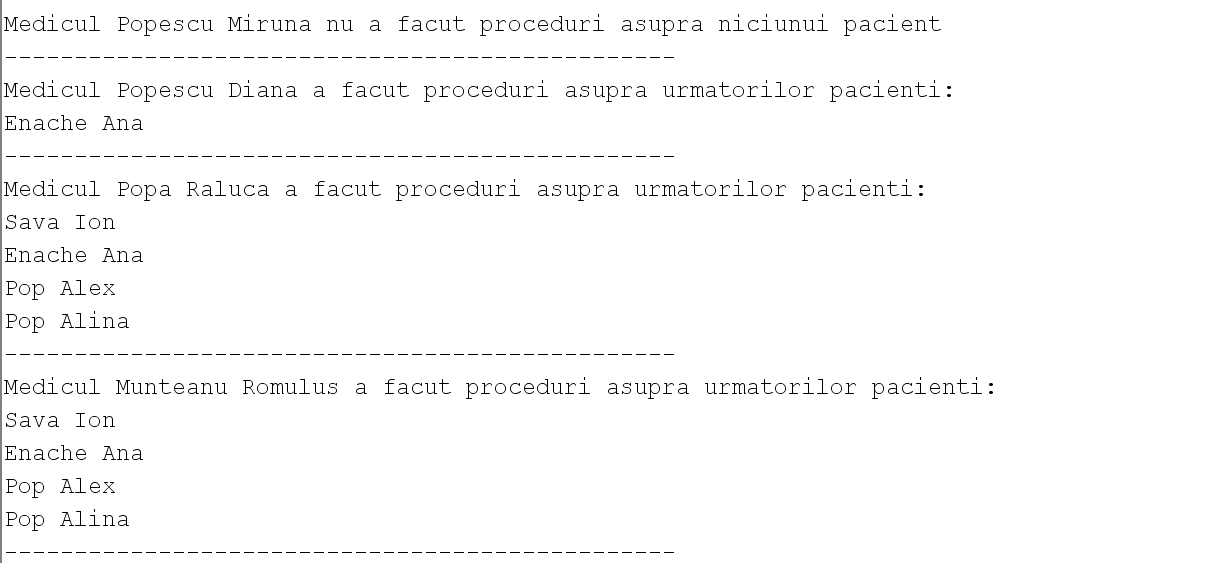
*Cerința*: Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat independent care să utilizeze 2 tipuri diferite de cursoare studiate, unul dintre acestea fiind cursor parametrizat, dependent de celălalt cursor. Apelați subprogramul.

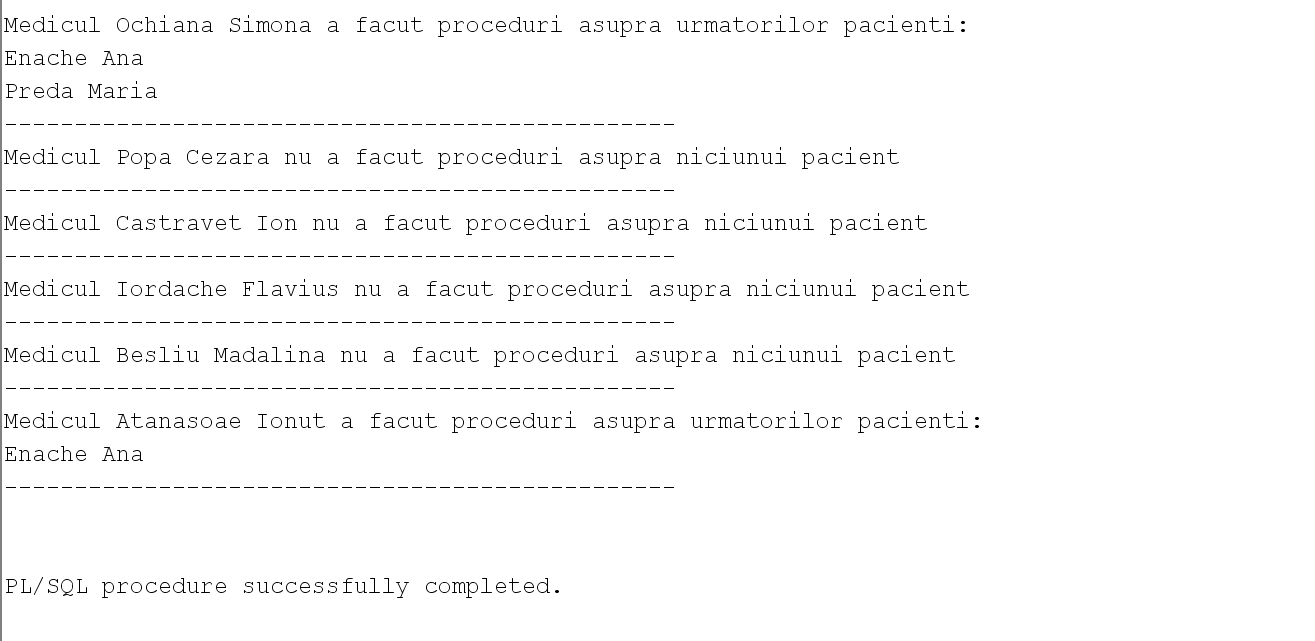
*Rezolvare:* Se citește de la tastatură o calificare. Să se afișeze medicii care au calificarea respectivă și pentru fiecare medic, lista pacienților asupra cărora au realizat proceduri. În cazul în care un medic nu a realizat proceduri asupra niciunui pacient, se va afișa textul ‘Niciun pacient’.

*În rezolvarea problemei s-au folosit 2 cursoare: un cursor explicit și parametrizat și un REF cursor cu SQL dinamic.*

|  |
| --- |
| SET SERVEROUTPUT ON;  SET VERIFY OFF;  CREATE OR REPLACE FUNCTION exista\_calificare(calif calificare\_ras.denumire%TYPE) RETURN NUMBER  IS  ct NUMBER;  BEGIN  SELECT COUNT(\*)  INTO ct  FROM calificare\_ras  WHERE UPPER(calif) = UPPER(denumire);    IF ct > 0 THEN  RETURN 1;  ELSE  RETURN 0;  END IF;  END exista\_calificare;  /  CREATE OR REPLACE PROCEDURE cer7\_ras  IS  TYPE lista\_medici IS REF CURSOR;  medici lista\_medici;  v\_calificare calificare\_ras.denumire%TYPE := '&p\_calificare';  CURSOR pacienti(v\_id\_medic pacient\_medic\_proced\_ras.id\_medic%TYPE) IS  SELECT DISTINCT p.id, p.nume, p.prenume  FROM pacient\_medic\_proced\_ras asoc JOIN pacienti\_ras p ON (asoc.id\_pacient = p.id)  WHERE asoc.id\_medic = v\_id\_medic;  v\_id\_m medici\_ras.id%TYPE;  v\_nume\_m angajati\_ras.nume%TYPE;  v\_prenume\_m angajati\_ras.prenume%TYPE;  v\_id\_p pacienti\_ras.id%TYPE;  v\_nume\_p pacienti\_ras.nume%TYPE;  v\_prenume\_p pacienti\_ras.prenume%TYPE;  calif\_inex EXCEPTION;  zero\_medici EXCEPTION;  exista NUMBER;  BEGIN  exista := exista\_calificare(v\_calificare);  IF exista = 0 THEN  RAISE calif\_inex;  END IF;  -- selectam medicii cu calificarea data  OPEN medici FOR  'SELECT m.id, a.nume, a.prenume  FROM medici\_ras m JOIN angajati\_ras a ON (m.id = a.id)  JOIN calificare\_ras c ON (m.id\_calificare = c.id)  WHERE UPPER(c.denumire) = UPPER(:v)'  USING v\_calificare;  FETCH medici INTO v\_id\_m, v\_nume\_m, v\_prenume\_m;  IF medici%NOTFOUND THEN  RAISE zero\_medici;  END IF;  --- parcurgem medicii  LOOP  DBMS\_OUTPUT.PUT('Medicul ' || v\_nume\_m || ' ' || v\_prenume\_m);  OPEN pacienti(v\_id\_m);  FETCH pacienti INTO v\_id\_p, v\_nume\_p, v\_prenume\_p;  IF pacienti%NOTFOUND THEN  DBMS\_OUTPUT.PUT\_LINE(' nu a facut proceduri asupra niciunui pacient');  ELSE  DBMS\_OUTPUT.PUT\_LINE(' a facut proceduri asupra urmatorilor pacienti: ');  WHILE pacienti%FOUND LOOP  DBMS\_OUTPUT.PUT\_LINE(v\_nume\_p || ' ' || v\_prenume\_p);  FETCH pacienti INTO v\_id\_p, v\_nume\_p, v\_prenume\_p;  END LOOP;  END IF;  CLOSE pacienti;  DBMS\_OUTPUT.PUT\_LINE('------------------------------------------------');  FETCH medici INTO v\_id\_m, v\_nume\_m, v\_prenume\_m;  EXIT WHEN medici%NOTFOUND;  END LOOP;  CLOSE medici;  EXCEPTION  WHEN calif\_inex THEN  RAISE\_APPLICATION\_ERROR(-20002, 'Calificarea nu exista');  WHEN zero\_medici THEN  RAISE\_APPLICATION\_ERROR(-20003, 'Nu exista medici cu calificarea data');  END cer7\_ras;  /  BEGIN  cer7\_ras;  END;  / |

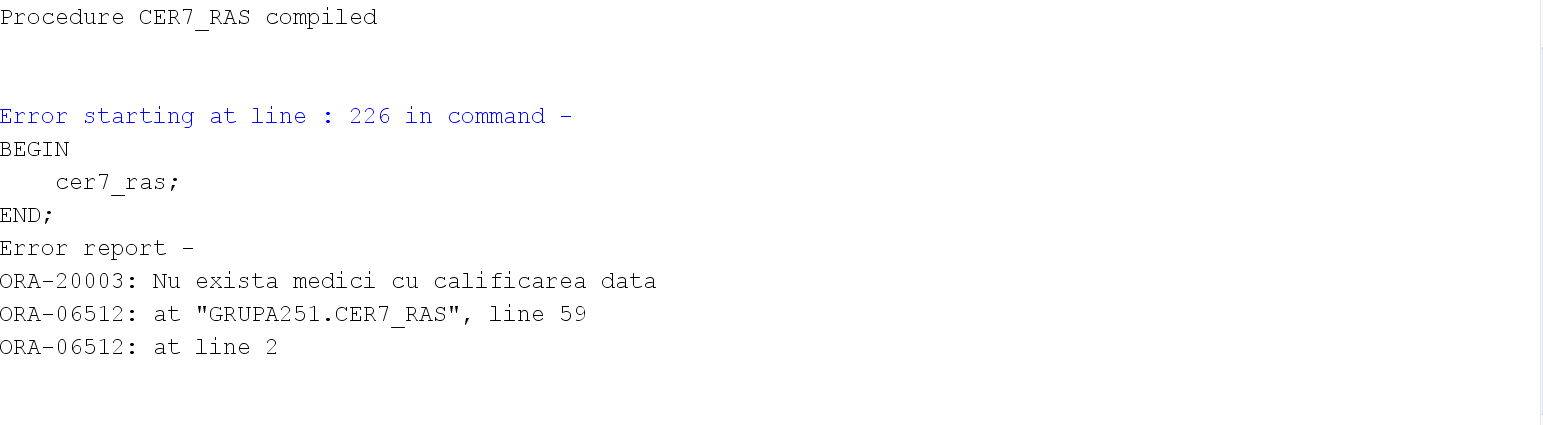
*Exemplul 1*Se compilează procedura și se introduce textul *primar.*





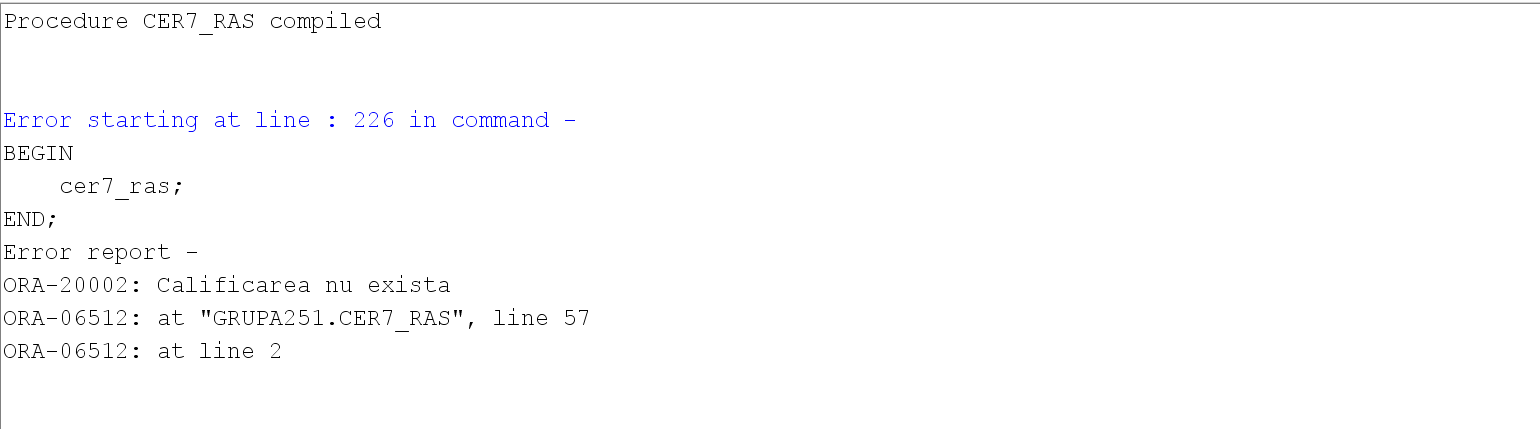
*Exemplul 2*

Se compilează procedura și se introduce textul *fizioterapeut.*



*Exemplul 3*

Se compilează procedura și se introduce textul *psiholog.*



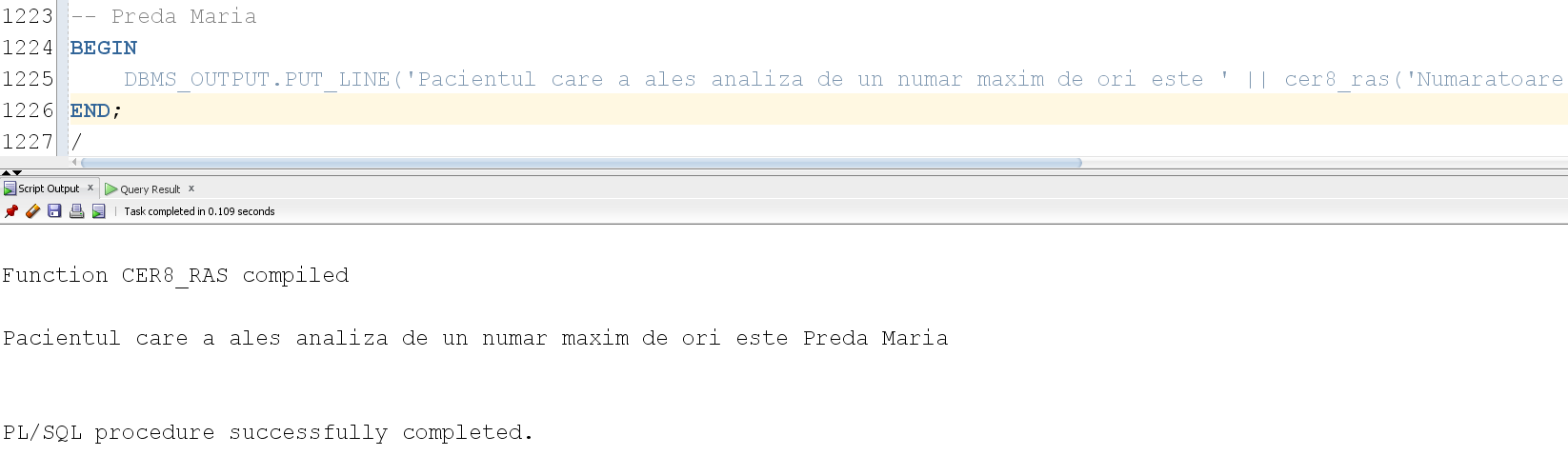
### Cerința 8

*Cerința:* Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat independent de tip funcție care să utilizeze într-o singură comandă SQL 3 dintre tabelele create. Tratați toate excepțiile care pot apărea, incluzând excepțiile predefinite NO\_DATA\_FOUND și TOO\_MANY\_ROWS. Apelați subprogramul astfel încât să evidențiați toate cazurile tratate.

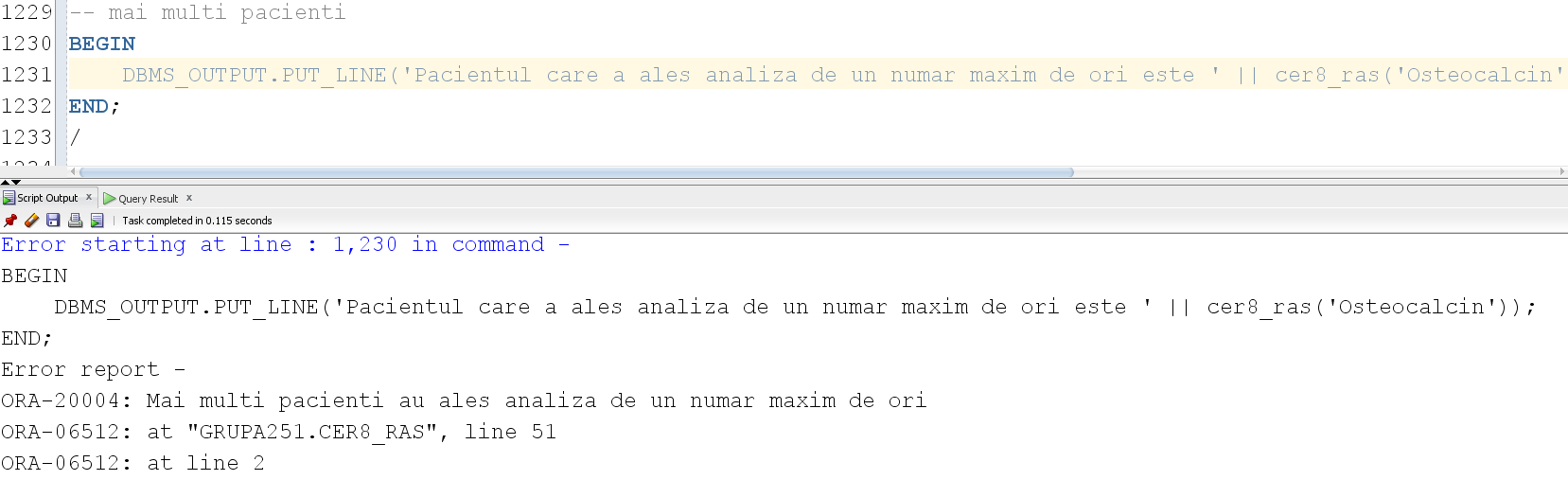
*Rezolvare:* Scrieți o funcție care primește numele unei analize și returnează numele și prenumele pacientului și-a făcut analiza respectivă de un număr maxim de ori. Se vor trata cazul în care analiza nu există, analiza nu a fost aleasă de nimeni, dar și cazul în care mai mulți pacienți au ales analiza de un număr maxim de ori.

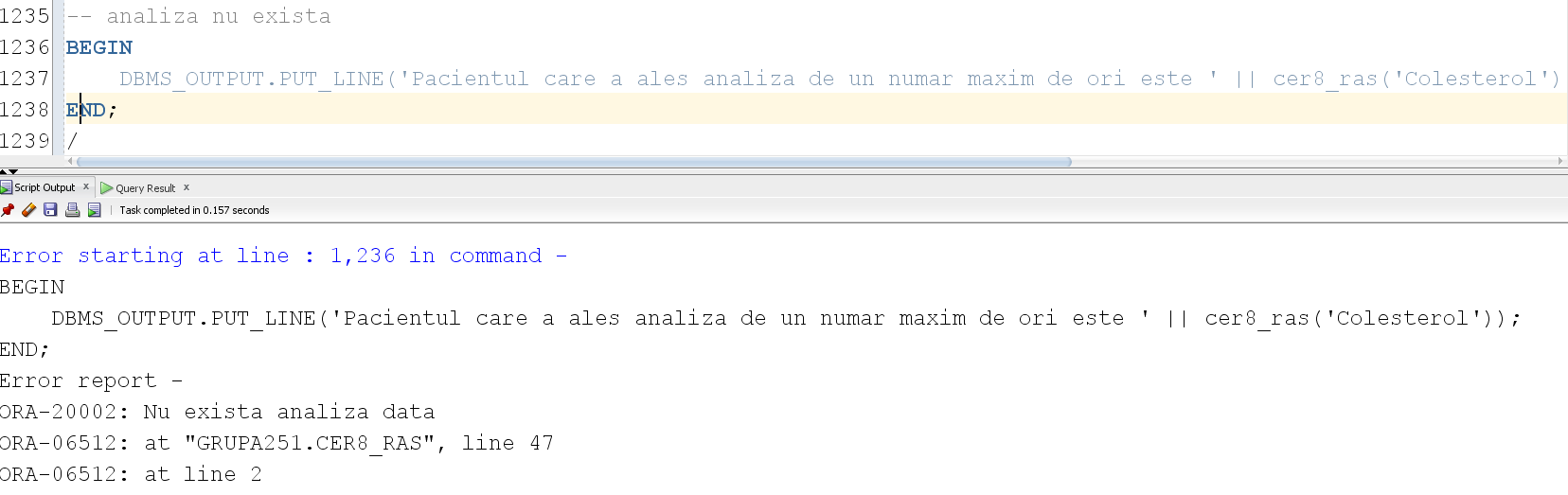
|  |
| --- |
| SET SERVEROUTPUT ON;  SET VERIFY OFF;  CREATE OR REPLACE FUNCTION cer8\_ras(v\_nume\_analiza analize\_ras.denumire%TYPE)  RETURN VARCHAR2  IS  nume\_p pacienti\_ras.nume%TYPE;  prenume\_p pacienti\_ras.prenume%TYPE;  val\_max NUMBER;  ans VARCHAR2(100);  v\_id analize\_ras.id%TYPE;  nimeni EXCEPTION;  ct NUMBER;  BEGIN    SELECT id  INTO v\_id  FROM analize\_ras  WHERE UPPER(denumire) = UPPER(v\_nume\_analiza);    SELECT COUNT(\*)  INTO ct  FROM pacienti\_aleg\_analize\_ras  WHERE id\_analiza = v\_id;    IF ct = 0 THEN  RAISE nimeni;  END IF;  -- calculam maximul  SELECT MAX(COUNT(\*))  INTO val\_max  FROM analize\_ras a JOIN pacienti\_aleg\_analize\_ras asoc ON (a.id = asoc.id\_analiza)  WHERE UPPER(a.denumire) = UPPER(v\_nume\_analiza)  GROUP BY asoc.id\_pacient;    -- selectam pacientul care a facut analiza de un numar maxim de ori  SELECT p.nume, p.prenume  INTO nume\_p, prenume\_p  FROM analize\_ras a JOIN pacienti\_aleg\_analize\_ras asoc ON (a.id = asoc.id\_analiza)  JOIN pacienti\_ras p ON (asoc.id\_pacient = p.id)  WHERE UPPER(a.denumire) = UPPER(v\_nume\_analiza)  GROUP BY p.id, p.nume, p.prenume  HAVING COUNT(a.id) = val\_max;  ans := nume\_p || ' ' || prenume\_p;  RETURN ans;    EXCEPTION  WHEN NO\_DATA\_FOUND THEN  RAISE\_APPLICATION\_ERROR(-20002, 'Nu exista analiza data');  WHEN nimeni THEN  RAISE\_APPLICATION\_ERROR(-20003, 'Niciun pacient nu a ales analiza data');  WHEN TOO\_MANY\_ROWS THEN  RAISE\_APPLICATION\_ERROR(-20004, 'Mai multi pacienti au ales analiza de un numar maxim de ori');  END cer8\_ras;  /  -- Preda Maria  BEGIN  DBMS\_OUTPUT.PUT\_LINE('Pacientul care a ales analiza de un numar maxim de ori este ' || cer8\_ras('Numaratoare reticulocite'));  END;  /  -- mai multi pacienti  BEGIN  DBMS\_OUTPUT.PUT\_LINE('Pacientul care a ales analiza de un numar maxim de ori este ' || cer8\_ras('Osteocalcin'));  END;  /  -- analiza nu exista  BEGIN  DBMS\_OUTPUT.PUT\_LINE('Pacientul care a ales analiza de un numar maxim de ori este ' || cer8\_ras('Colesterol'));  END;  /  -- analiza nu a fost aleasa de nimeni  BEGIN  DBMS\_OUTPUT.PUT\_LINE('Pacientul care a ales analiza de un numar maxim de ori este ' || cer8\_ras('Glicemie'));  END;  / |

*Exemplul 1*

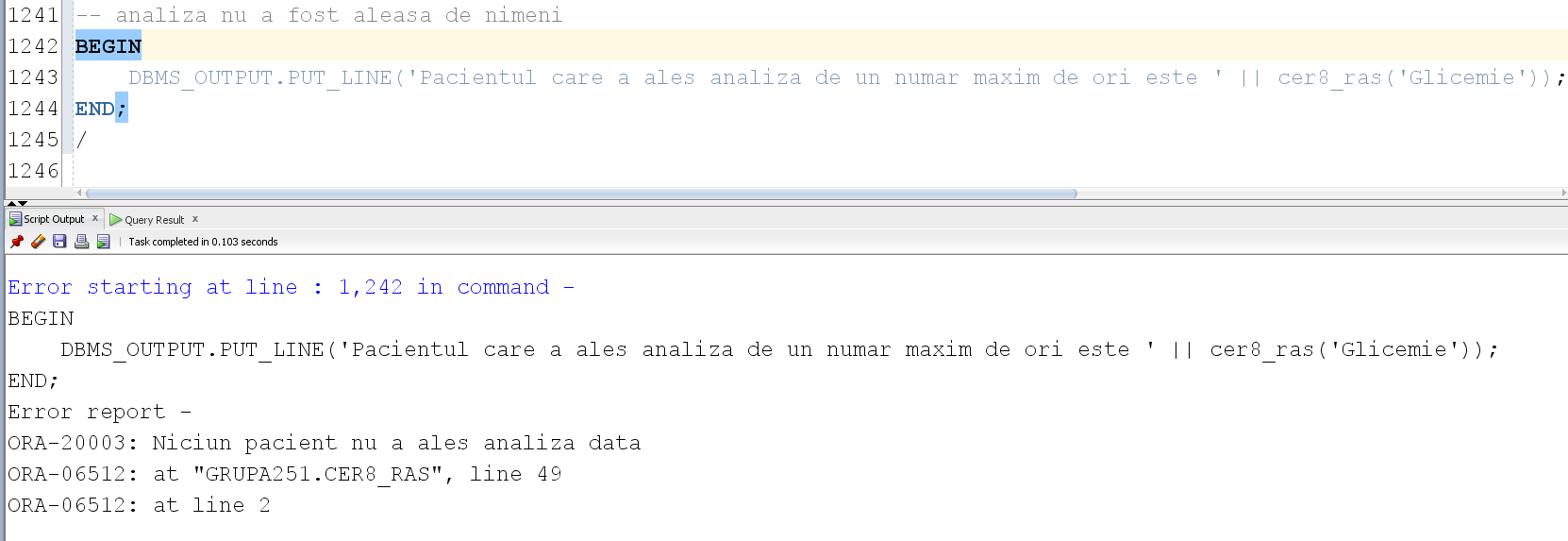


*Exemplul 2*



*Exemplul 3*

*Exemplul 4*



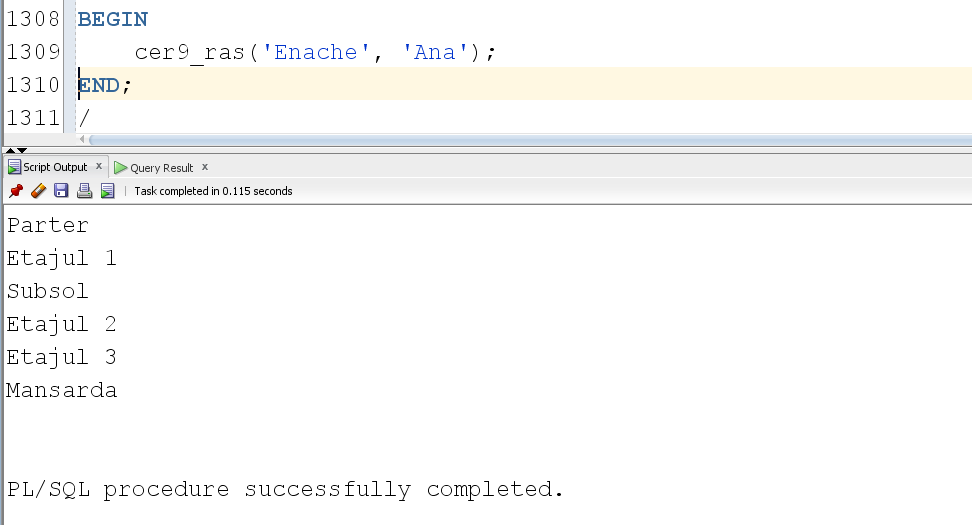
### Cerința 9

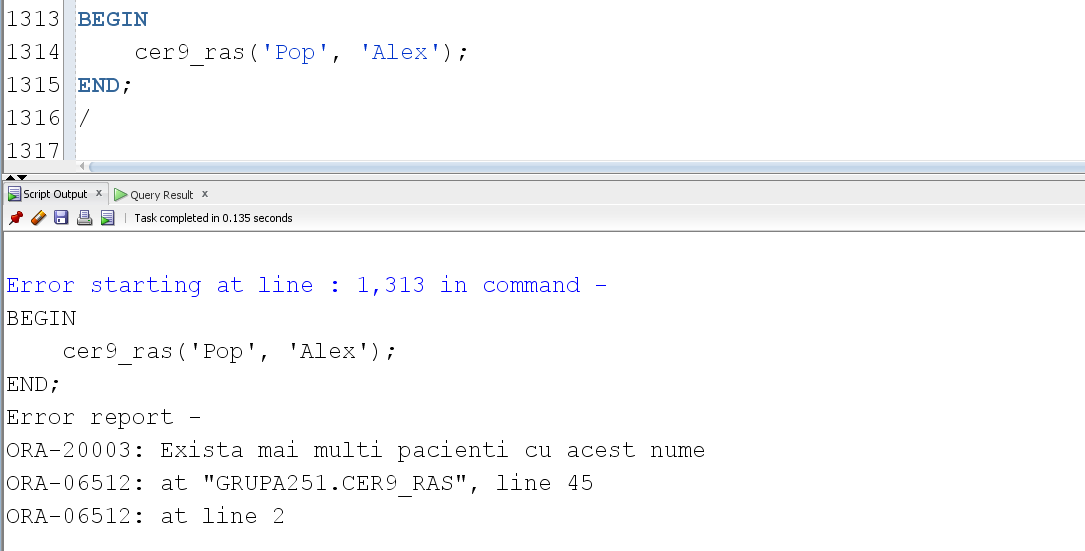
*Cerința*: Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat independent de tip procedură care să aibă minim 2 parametri și să utilizeze într-o singură comandă SQL 5 dintre tabelele create. Definiți minim 2 excepții proprii, altele decât cele predefinite la nivel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

*Rezolvare*: Scrieți o procedură care primește numele și prenumele unui pacient și afișează lista tuturor etajelor corespunzătoare secțiilor la care acesta a fost supus procedurilor. Pe lângă cazul de bază, se vor trata cazurile în care pacientul nu există în baza de date, există mai mulți pacienți cu numele și prenumele furnizate ca parametrii, pacientul nu a realizat nicio procedură sau datele introduse sunt invalide (lungimea numelui sau a prenumelui este strict mai mică decât 3).

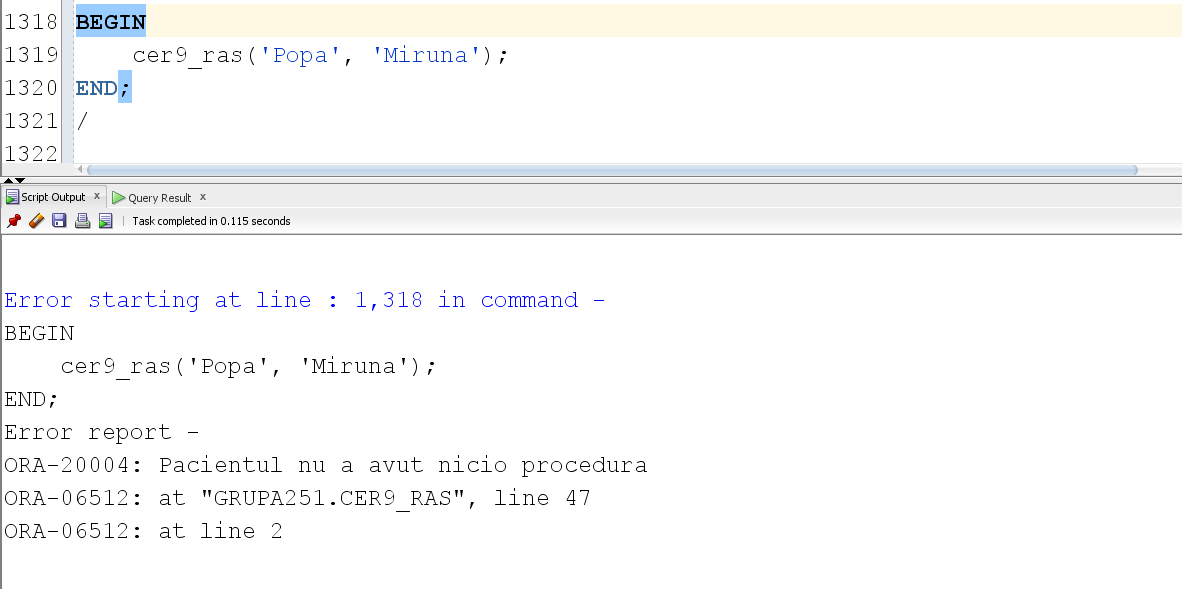
|  |
| --- |
| SET SERVEROUTPUT ON;  SET VERIFY OFF;  CREATE OR REPLACE PROCEDURE cer9\_ras (v\_nume pacienti\_ras.nume%TYPE, v\_prenume pacienti\_ras.prenume%TYPE)  IS  v\_id pacienti\_ras.id%TYPE;  TYPE tablou\_imbr IS TABLE OF etaj\_ras.nume%TYPE;  etaje tablou\_imbr := tablou\_imbr();  zero\_proceduri EXCEPTION;  date\_invalide EXCEPTION;  ct NUMBER;  BEGIN  IF LENGTH(v\_nume) < 3 OR LENGTH(v\_prenume) < 3 THEN  RAISE date\_invalide;  END IF;  -- selectam id-ul pacientului  SELECT id  INTO v\_id  FROM pacienti\_ras  WHERE UPPER(nume) = UPPER(v\_nume) AND UPPER(prenume) = UPPER(v\_prenume);    -- verificam daca pacientul a facut vreo procedura  SELECT COUNT(\*)  INTO ct  FROM pacient\_medic\_proced\_ras  WHERE id\_pacient = v\_id;    IF ct = 0 THEN  RAISE zero\_proceduri;  END IF;    SELECT DISTINCT e.nume  BULK COLLECT INTO etaje  FROM pacient\_medic\_proced\_ras asoc JOIN proceduri\_ras p ON (asoc.id\_proced = p.id)  JOIN sectie\_ras s ON (p.id\_sectie = s.id)  JOIN camere\_ras c ON (s.id = c.id\_sectie)  JOIN etaj\_ras e ON (e.id = c.id\_etaj)  WHERE asoc.id\_pacient = v\_id;    FOR i IN etaje.FIRST..etaje.LAST LOOP  DBMS\_OUTPUT.PUT\_LINE(etaje(i) || ' ');  END LOOP;  EXCEPTION  WHEN NO\_DATA\_FOUND THEN  RAISE\_APPLICATION\_ERROR(-20002, 'Nu exista niciun pacient cu numele dat');  WHEN TOO\_MANY\_ROWS THEN  RAISE\_APPLICATION\_ERROR(-20003, 'Exista mai multi pacienti cu acest nume');  WHEN zero\_proceduri THEN  RAISE\_APPLICATION\_ERROR(-20004, 'Pacientul nu a avut nicio procedura');  WHEN date\_invalide THEN  RAISE\_APPLICATION\_ERROR(-20005, 'Numele / prenumele introdus este prea scurt!');  END cer9\_ras;  /  BEGIN  cer9\_ras('Enache', 'Ana');  END;  /  BEGIN  cer9\_ras('Pop', 'Alex');  END;  /  BEGIN  cer9\_ras('Popa', 'Miruna');  END;  /  BEGIN  cer9\_ras('Dumitrescu', 'Ni');  END;  /  BEGIN  cer9\_ras('Sava', 'Matei');  END;  / |

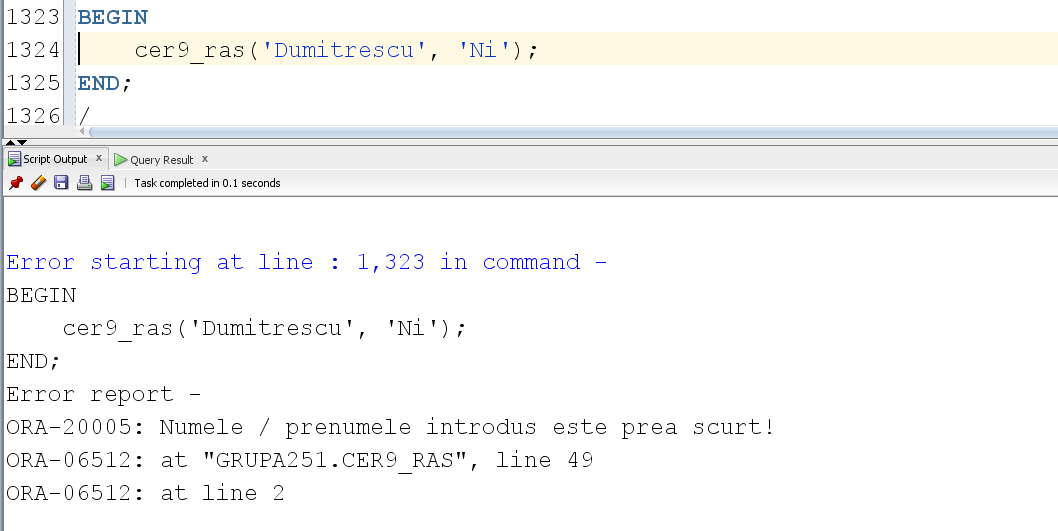
*Exemplul 1*

*Exemplul 2*

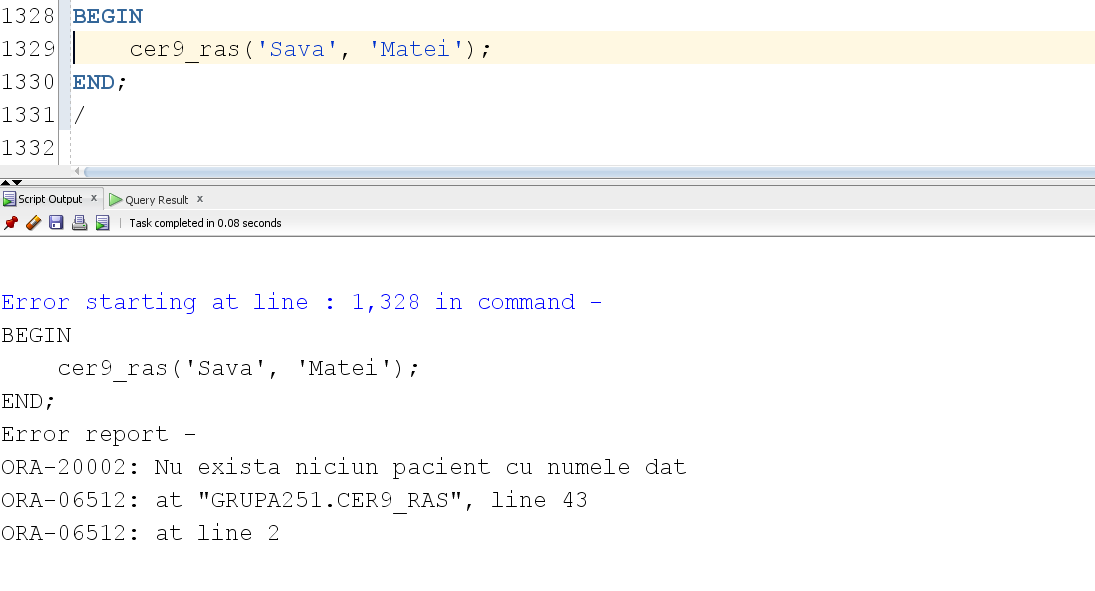


*Exemplul 3*



*Exemplul 4*

*Exemplul 5*



### 

### 

### 

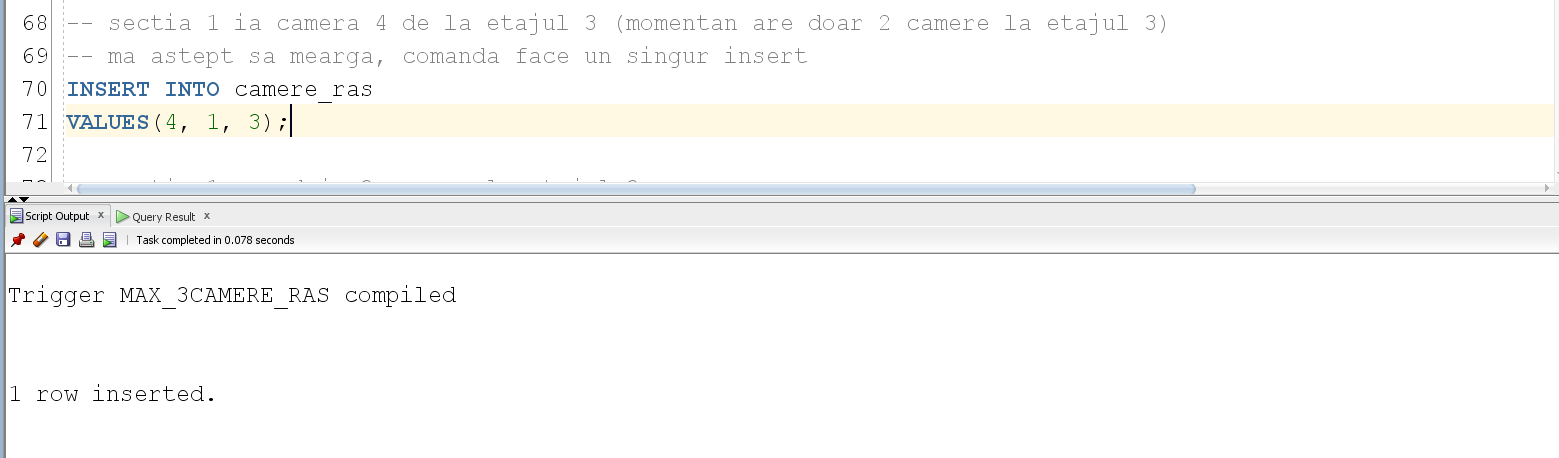
### Cerința 10

*Cerința*: Definiți un trigger de tip LMD la nivel de comandă. Declanșați trigger-ul.

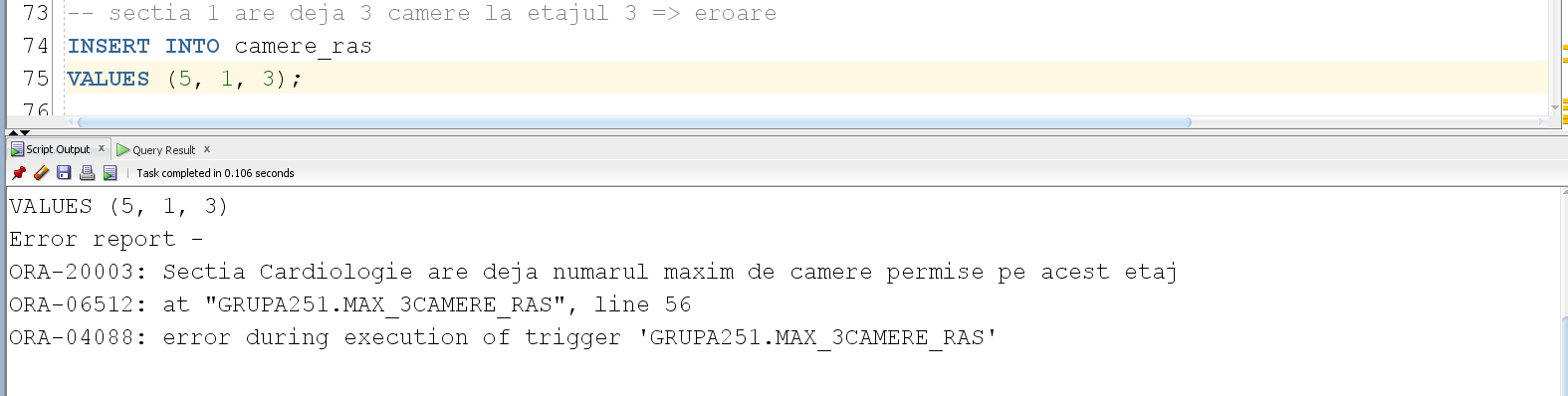
*Rezolvare*: Implementați, cu ajutorul triggerilor, următoarea restricție: o secție **nu** poate avea mai mult de 3 camere pe un etaj.

Inițial, încercăm implementarea unui trigger la nivel de linie care să transpună restricția dată.

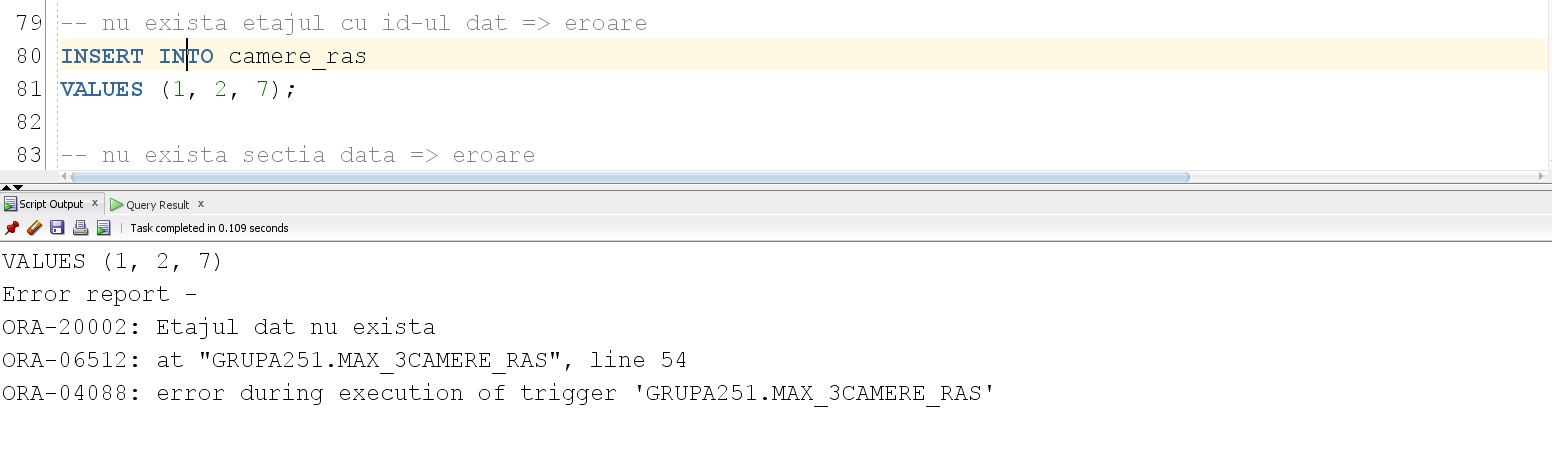
|  |
| --- |
| CREATE OR REPLACE TRIGGER max\_3camere\_ras  BEFORE INSERT OR UPDATE OF id\_etaj, id\_sectie ON camere\_ras  FOR EACH ROW  DECLARE  nr\_camere NUMBER;  nume\_sectie sectie\_ras.denumire%TYPE;  ct\_sectie NUMBER;  sectie\_inexistenta EXCEPTION;  nr\_maxim EXCEPTION;  etaj\_inexistent EXCEPTION;  ct\_etaj NUMBER;  mesaj\_trigger VARCHAR2(100);  BEGIN  -- vedem daca exista etajul dat  SELECT COUNT(\*)  INTO ct\_etaj  FROM etaj\_ras  WHERE id = :NEW.id\_etaj;    IF ct\_etaj = 0 THEN  RAISE etaj\_inexistent;  END IF;  SELECT COUNT(\*)  INTO nr\_camere  FROM camere\_ras  WHERE id\_etaj = :NEW.id\_etaj AND id\_sectie = :NEW.id\_sectie;  IF :NEW.id\_sectie IS NOT NULL THEN  -- sectie\_ras nu e mutating, pot sa il interoghez  SELECT COUNT(\*)  INTO ct\_sectie  FROM sectie\_ras  WHERE id = :NEW.id\_sectie;    IF ct\_sectie = 0 THEN  RAISE sectie\_inexistenta;  END IF;  -- selectam numele sectiei  SELECT denumire  INTO nume\_sectie  FROM sectie\_ras  WHERE id = :NEW.id\_sectie;    IF nr\_camere = 3 THEN  mesaj\_trigger := 'Sectia ' || nume\_sectie || ' are deja numarul maxim de camere permise pe acest etaj';  RAISE nr\_maxim;  END IF;    END IF;  EXCEPTION  WHEN sectie\_inexistenta THEN  RAISE\_APPLICATION\_ERROR(-20001, 'Sectia data nu exista');  WHEN etaj\_inexistent THEN  RAISE\_APPLICATION\_ERROR(-20002, 'Etajul dat nu exista');  WHEN nr\_maxim THEN  RAISE\_APPLICATION\_ERROR(-20003, mesaj\_trigger);  END;  /  -- sectia 1 ia camera 4 de la etajul 3 (momentan are doar 2 camere la etajul 3)  -- ma astept sa mearga, comanda face un singur insert  INSERT INTO camere\_ras  VALUES(4, 1, 3);  -- sectia 1 are deja 3 camere la etajul 3 => eroare  INSERT INTO camere\_ras  VALUES (5, 1, 3);  ROLLBACK;  -- nu exista etajul cu id-ul dat => eroare  INSERT INTO camere\_ras  VALUES (1, 2, 7);  -- nu exista sectia data => eroare  INSERT INTO camere\_ras  VALUES (5, 8, 1);  -- insert into ... select ... => mutating table  INSERT INTO camere\_ras  SELECT 4, 1, 3  FROM DUAL;  -- mutating table  -- sectia 6 are 2 camere la etajul 1  -- sectia 5 nu are nicio camera la etajul 1  -- => operatia ar trebui sa poata fi executata, dar obtinem mutating  UPDATE camere\_ras  SET id\_sectie = 5  WHERE id\_sectie = 6;  ROLLBACK; |

*Exemplul 1*

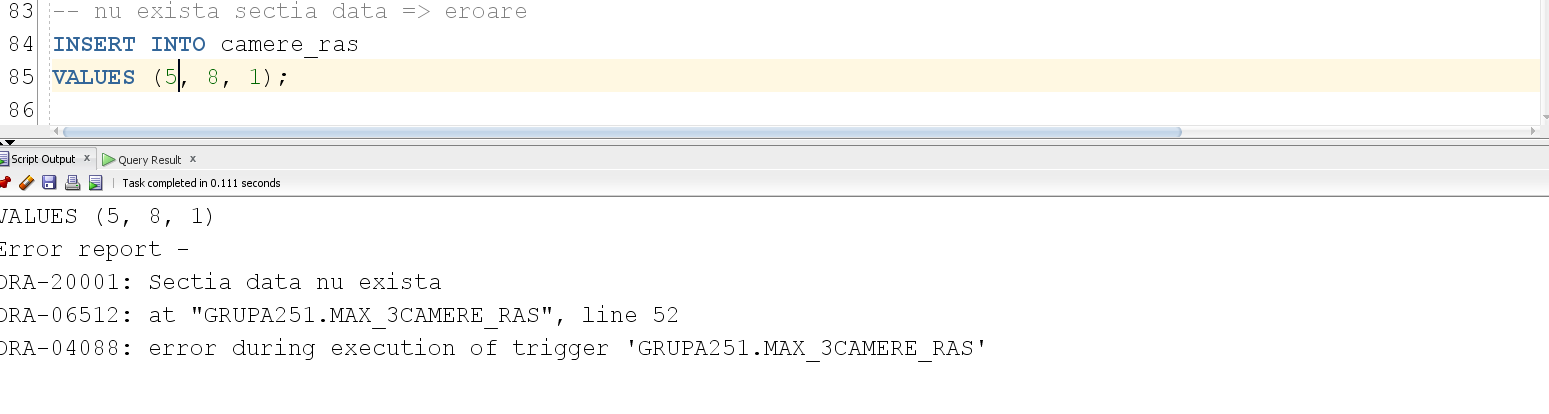
*Exemplul 2*

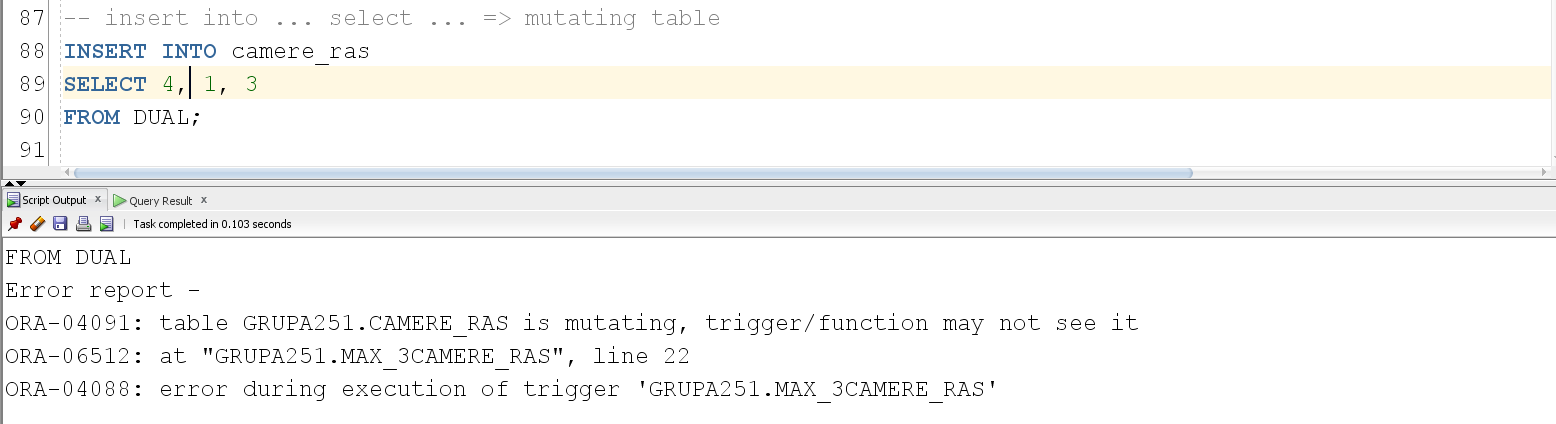


*Exemplul 3*

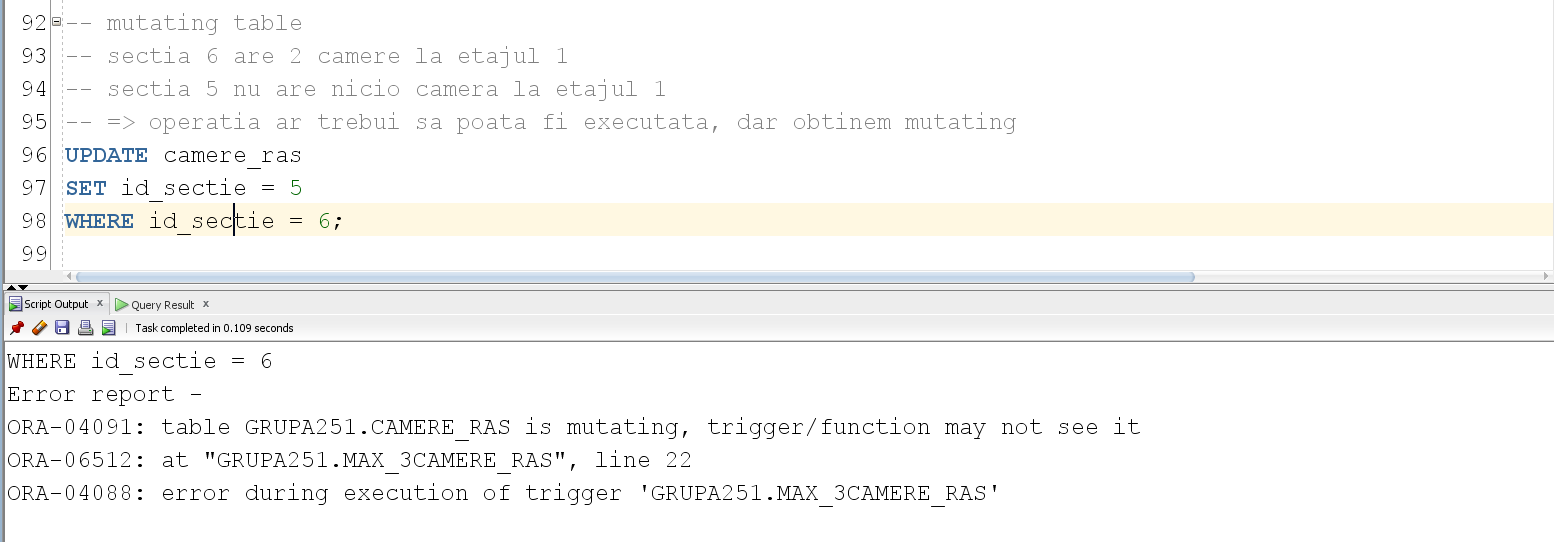


*Exemplul 4*

*Exemplul 5*

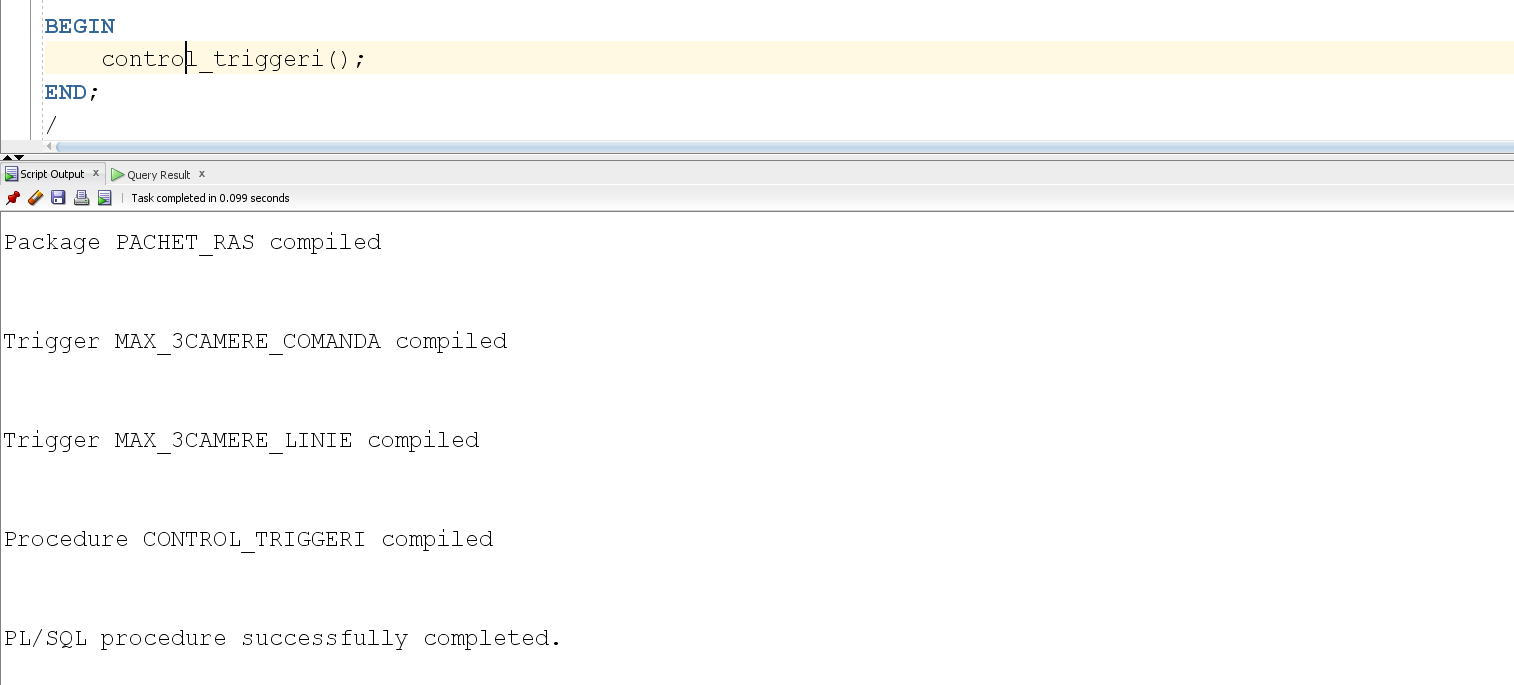


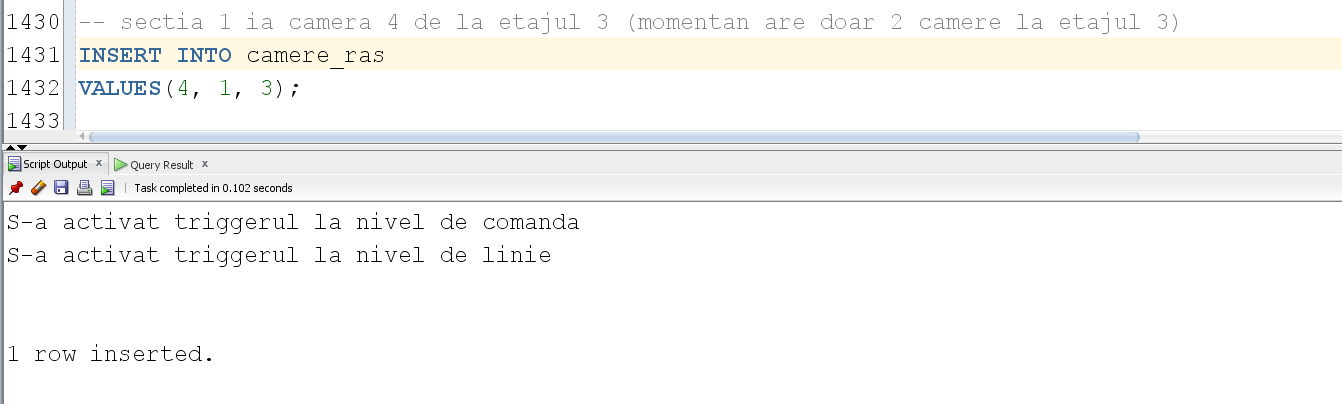
*Exemplul 6*



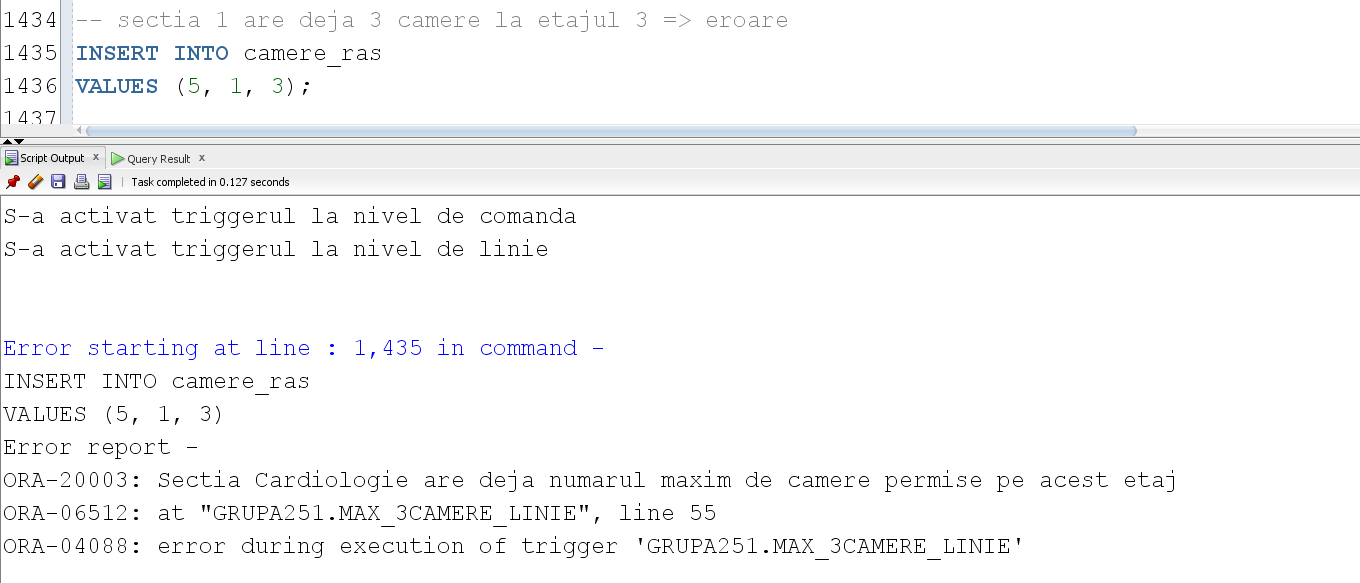
*Pentru a evita eroarea mutating table, vom folosi un pachet cu 2 triggeri, denumiți sugestiv:* ***max\_3camere\_comanda (triggerul la nivel de comandă cerut în enunț)*** *și* ***max\_3camere\_linie (trigger la nivel de linie)****. Vom folosi o procedură care activează și dezactivează triggerii, urmând apoi se executăm aceleași comenzi.*

|  |
| --- |
| CREATE OR REPLACE PACKAGE pachet\_ras  AS  TYPE linie IS RECORD  (id\_sectie camere\_ras.id\_sectie%TYPE,  id\_etaj camere\_ras.id\_etaj%TYPE,  ct\_camere NUMBER);  TYPE tabl\_ind IS TABLE OF linie  INDEX BY PLS\_INTEGER;  info tabl\_ind;  contor NUMBER := 0;  END pachet\_ras;  /  CREATE OR REPLACE TRIGGER max\_3camere\_comanda  BEFORE INSERT OR UPDATE OF id\_etaj, id\_sectie ON camere\_ras  BEGIN  pachet\_ras.contor := 0;  -- cate camere are sectia aia pe etajul dat  SELECT id\_sectie, id\_etaj, COUNT(\*)  BULK COLLECT INTO pachet\_ras.info  FROM camere\_ras  GROUP BY id\_sectie, id\_etaj;    DBMS\_OUTPUT.PUT\_LINE('S-a activat triggerul la nivel de comanda');    END;  /  CREATE OR REPLACE TRIGGER max\_3camere\_linie  BEFORE INSERT OR UPDATE OF id\_etaj, id\_sectie ON camere\_ras  FOR EACH ROW  DECLARE  -- verificari input  ct\_etaj NUMBER;  etaj\_inexistent EXCEPTION;  ct\_sectie NUMBER;  sectie\_inexistenta EXCEPTION;  -- variabile trigger  nume\_sectie sectie\_ras.denumire%TYPE;  mesaj\_trigger VARCHAR2(100);  nr\_maxim EXCEPTION;  BEGIN    DBMS\_OUTPUT.PUT\_LINE('S-a activat triggerul la nivel de linie');  SELECT COUNT(\*)  INTO ct\_etaj  FROM etaj\_ras  WHERE id = :NEW.id\_etaj;    IF ct\_etaj = 0 THEN  RAISE etaj\_inexistent;  END IF;    IF :NEW.id\_sectie IS NOT NULL THEN  SELECT COUNT(\*)  INTO ct\_sectie  FROM sectie\_ras  WHERE id = :NEW.id\_sectie;    IF ct\_sectie = 0 THEN  RAISE sectie\_inexistenta;  END IF;    FOR i IN 1..pachet\_ras.info.LAST LOOP  IF pachet\_ras.info(i).id\_sectie = :NEW.id\_sectie AND pachet\_ras.info(i).id\_etaj = :NEW.id\_etaj  AND pachet\_ras.info(i).ct\_camere + pachet\_ras.contor = 3 THEN  -- luam numele sectiei  SELECT denumire  INTO nume\_sectie  FROM sectie\_ras  WHERE id = :NEW.id\_sectie;    mesaj\_trigger := 'Sectia ' || nume\_sectie || ' are deja numarul maxim de camere permise pe acest etaj';  RAISE nr\_maxim;  END IF;  END LOOP;  pachet\_ras.contor := pachet\_ras.contor + 1;  END IF;  EXCEPTION  WHEN etaj\_inexistent THEN  RAISE\_APPLICATION\_ERROR(-20001, 'Etajul dat nu exista');  WHEN sectie\_inexistenta THEN  RAISE\_APPLICATION\_ERROR(-20002, 'Sectia data nu exista');  WHEN nr\_maxim THEN  RAISE\_APPLICATION\_ERROR(-20003, mesaj\_trigger);  END;  /  -- procedura care activeaza / dezactiveaza triggerii  CREATE OR REPLACE PROCEDURE control\_triggeri  IS  BEGIN  EXECUTE IMMEDIATE  'ALTER TRIGGER  max\_3camere\_ras DISABLE';  EXECUTE IMMEDIATE  'ALTER TRIGGER  max\_3camere\_comanda ENABLE';  EXECUTE IMMEDIATE  'ALTER TRIGGER  max\_3camere\_linie ENABLE';  END;  /  BEGIN  control\_triggeri();  END;  /  -- sectia 1 ia camera 4 de la etajul 3 (momentan are doar 2 camere la etajul 3)  INSERT INTO camere\_ras  VALUES(4, 1, 3);  -- sectia 1 are deja 3 camere la etajul 3 => eroare  INSERT INTO camere\_ras  VALUES (5, 1, 3);  ROLLBACK;  -- nu exista etajul cu id-ul dat => eroare  INSERT INTO camere\_ras  VALUES (1, 2, 7);  -- nu exista sectia data => eroare  INSERT INTO camere\_ras  VALUES (5, 8, 1);  -- insert into ... select ... => merge  INSERT INTO camere\_ras  SELECT 4, 1, 3  FROM DUAL;  ROLLBACK;  -- sectia 6 are 2 camere la etajul 1  -- sectia 5 nu are nicio camera la etajul 1  -- => operatia ar trebui sa poata fi executata  UPDATE camere\_ras  SET id\_sectie = 5  WHERE id\_sectie = 6;  ROLLBACK;  -- exemplul 7  INSERT INTO camere\_ras  VALUES(4, NULL, 1);  INSERT INTO camere\_ras  VALUES(5, NULL, 1);  INSERT INTO camere\_ras  VALUES(6, NULL, 1);  UPDATE camere\_ras  SET id\_sectie = 6  WHERE id\_sectie IS NULL AND id\_etaj = 1;  ROLLBACK; |

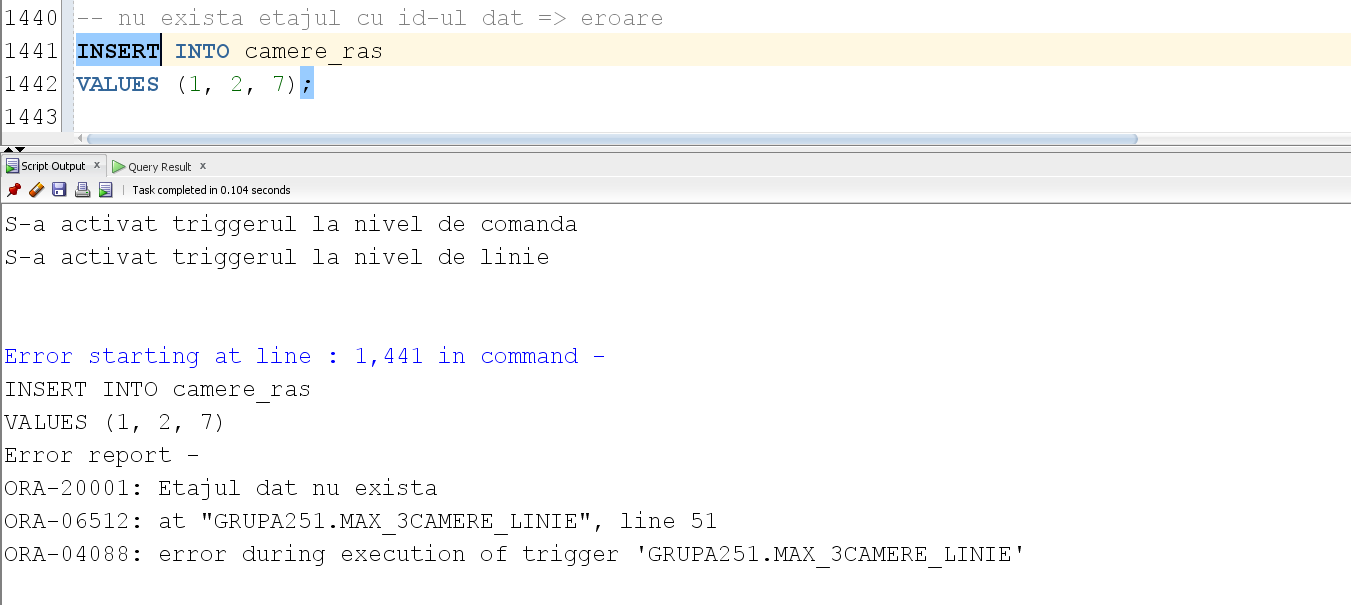


*Exemplul 1*

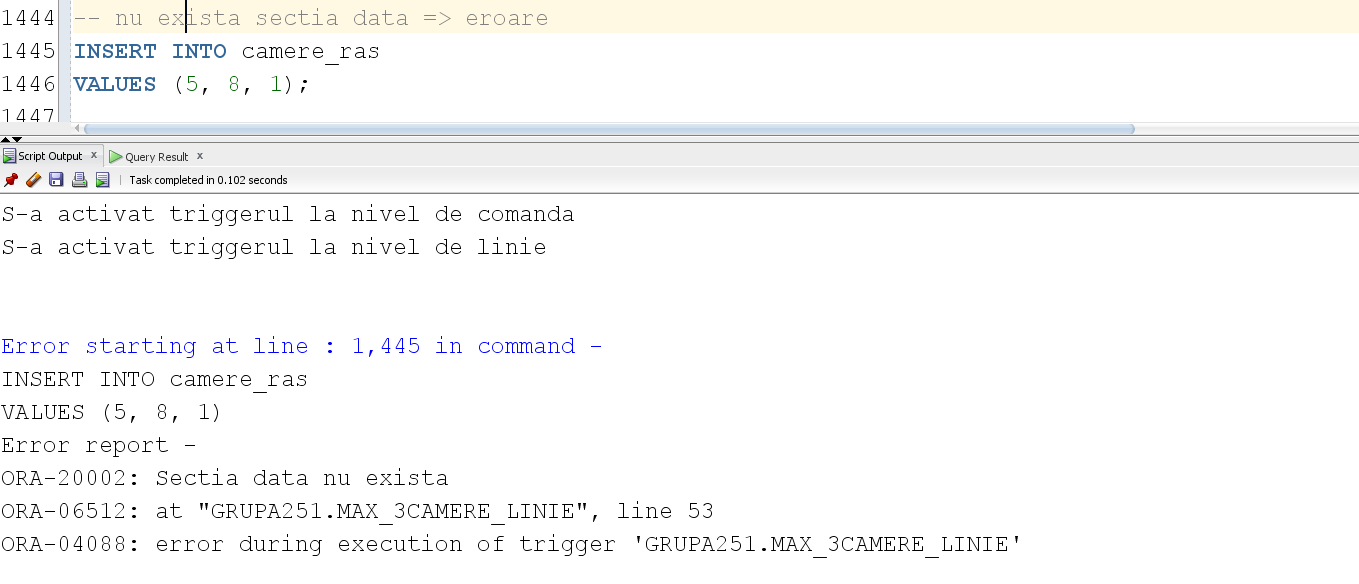
*Exemplul 2*



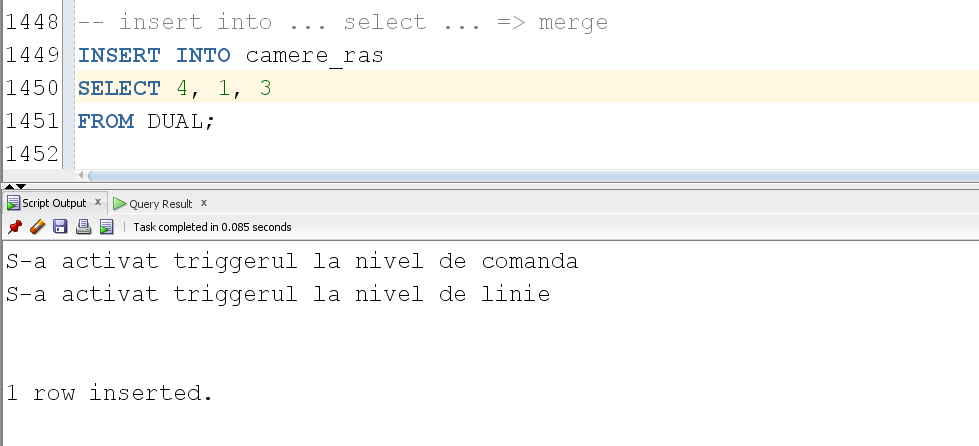
*Exemplul 3*



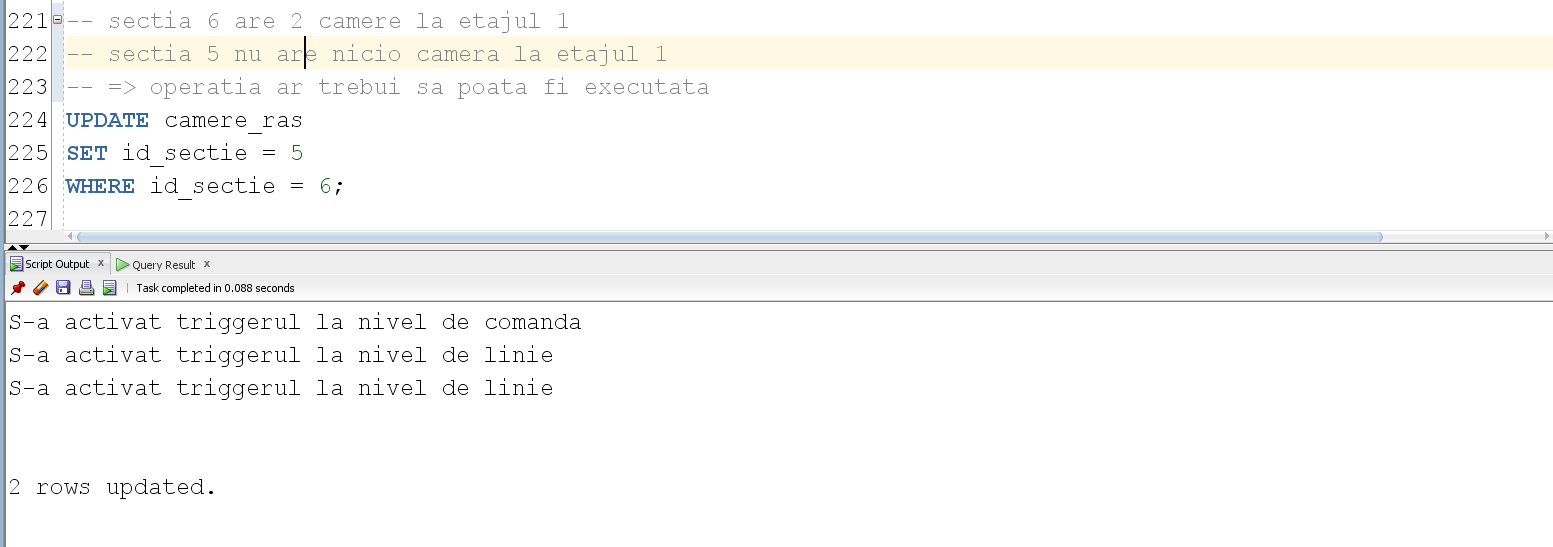
*Exemplul 4*

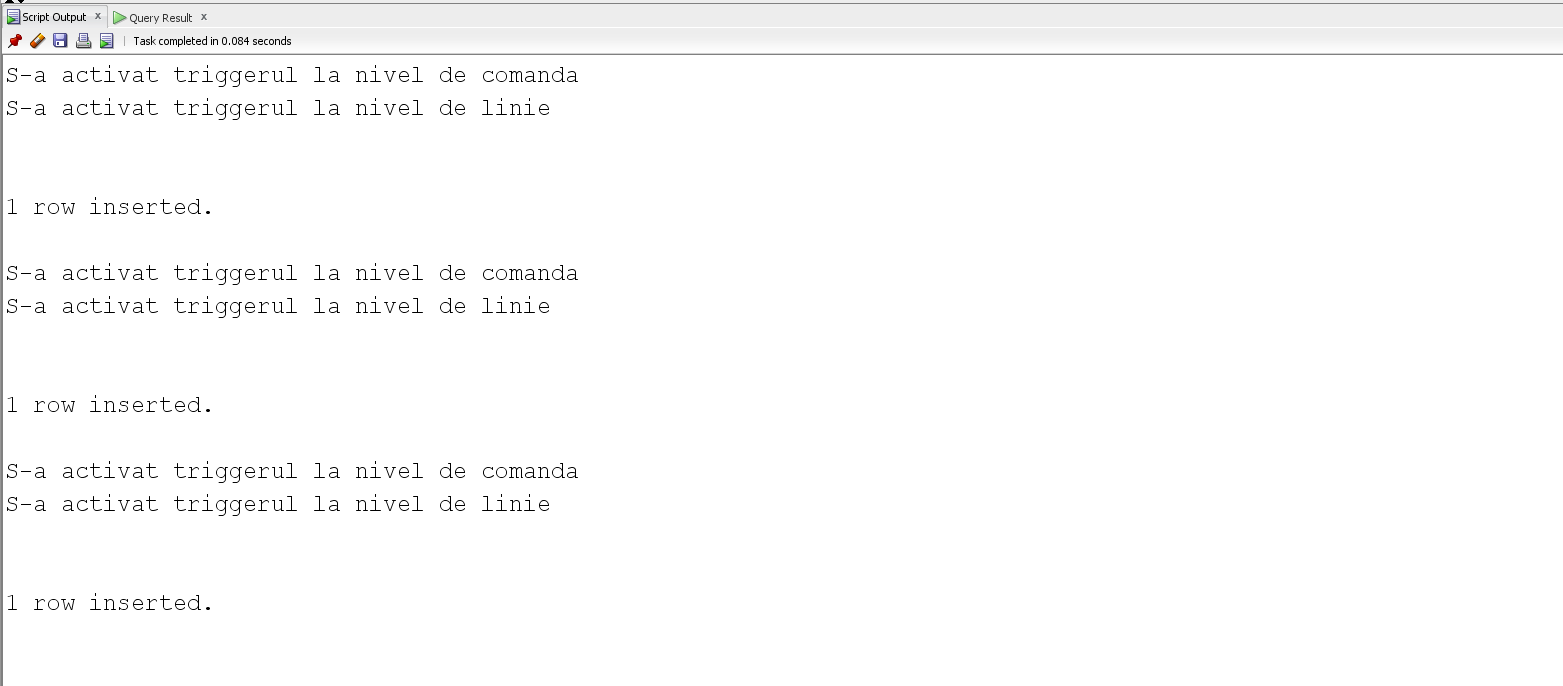


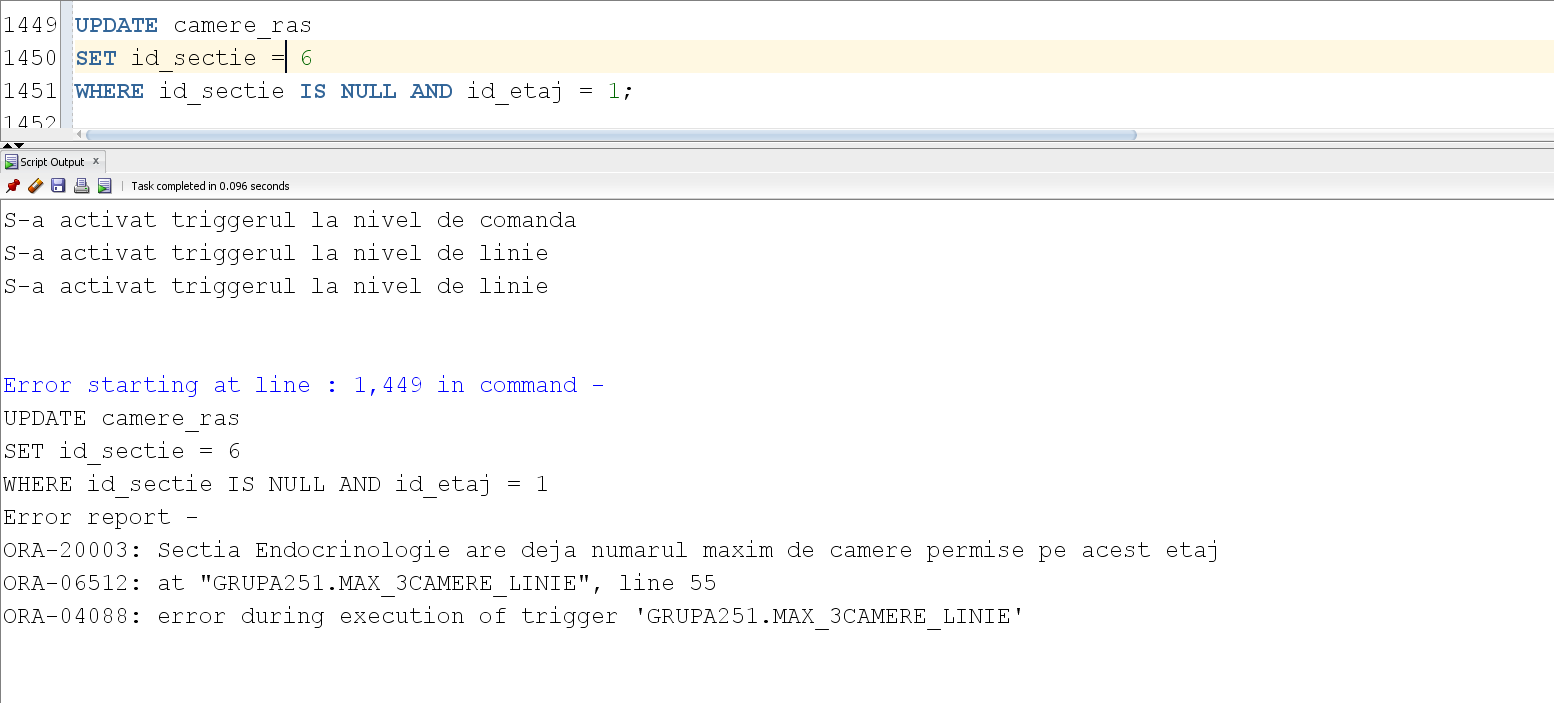
*Exemplul 5*



*Exemplul 6*



*Exemplul 7*



### 

### 

### 

### 

### 

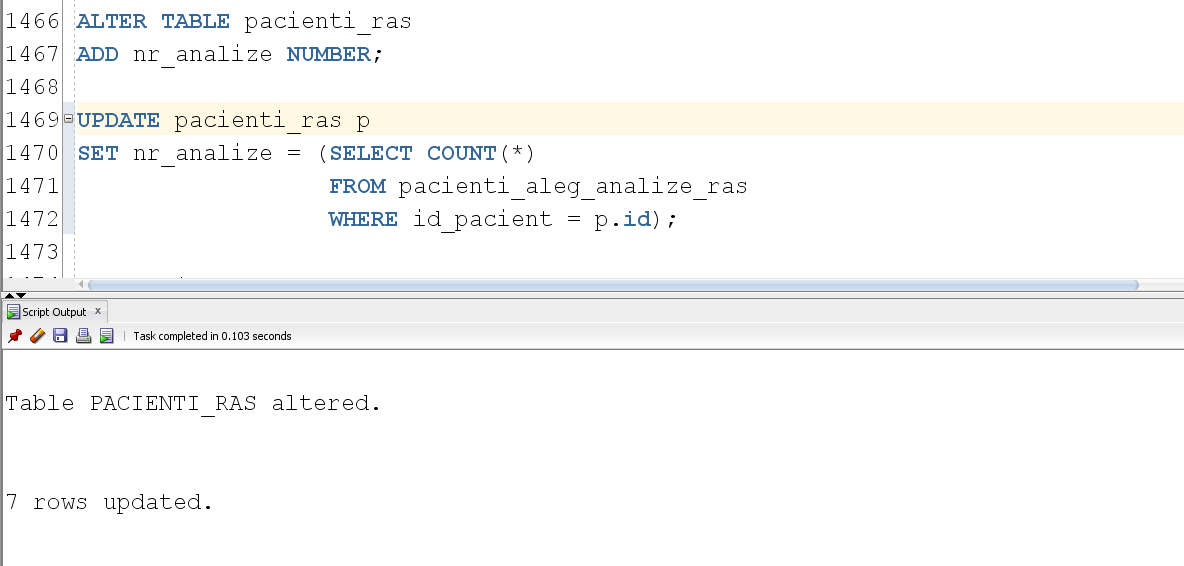
### Cerința 11

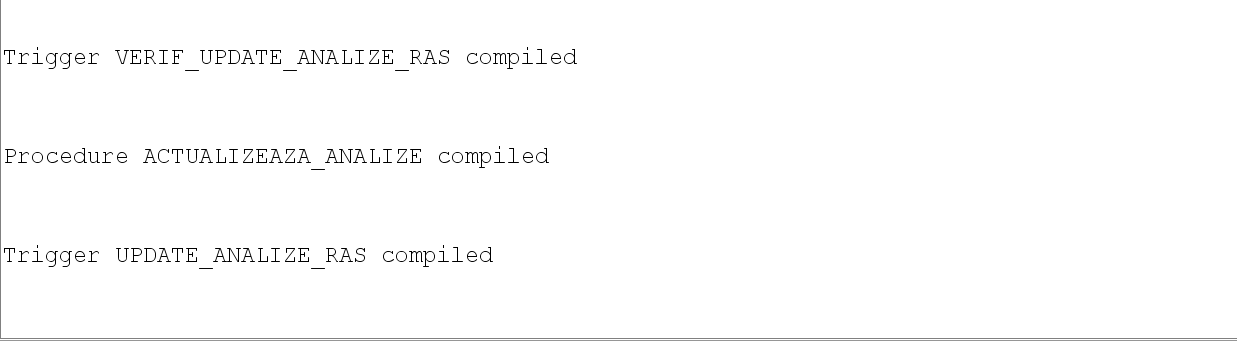
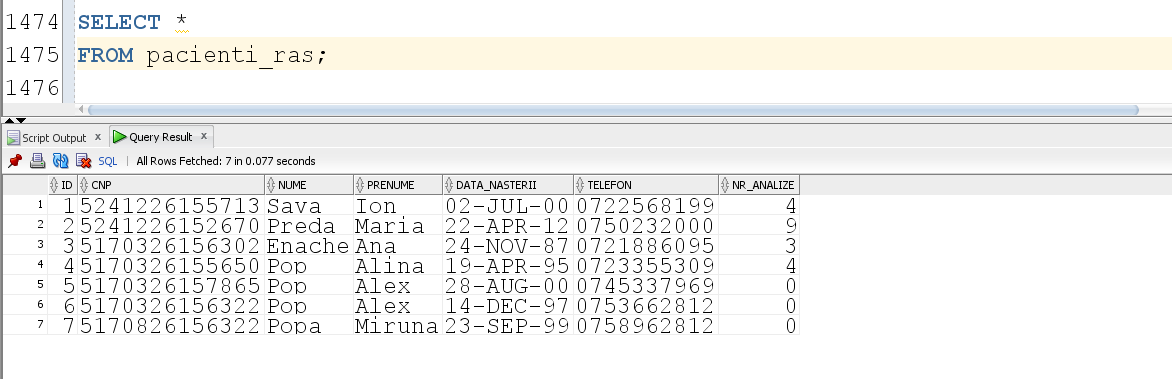
*Cerința*: Definiți un trigger de tip LMD la nivel de linie. Declanșați trigger-ul.

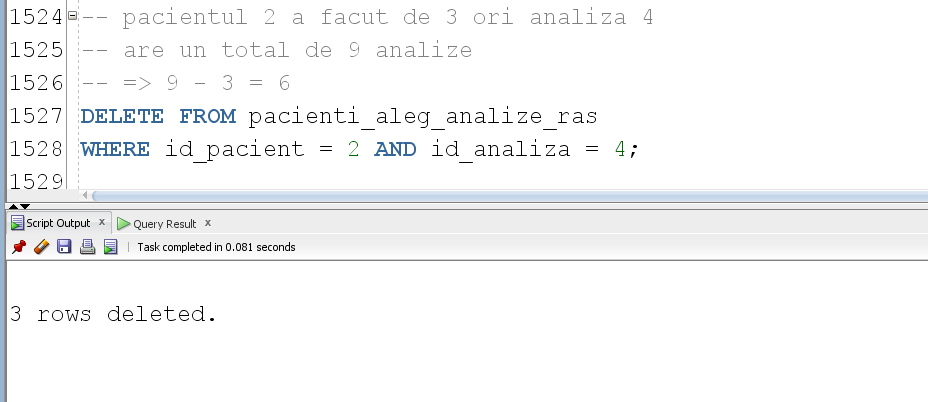
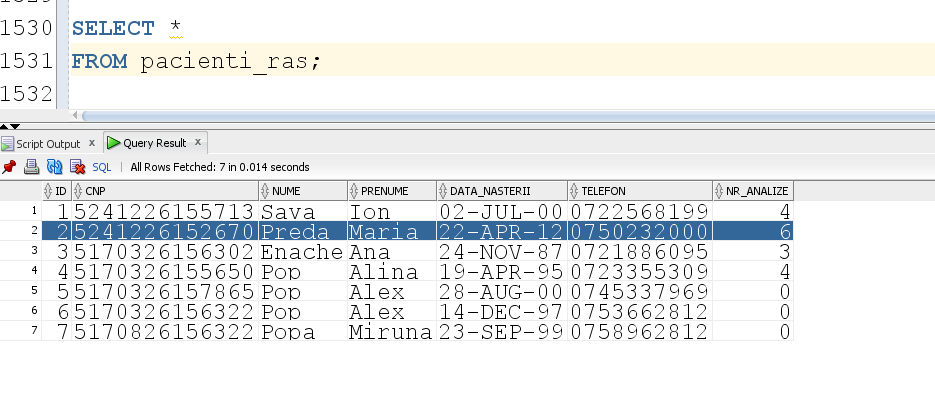
*Rezolvare*: Să se adauge o nouă coloană tabelului pacienti, *nr\_analize* NUMBER, care va reține numărul de analize realizate de un pacient. Coloana va fi actualizată de fiecare dată când se fac modificări în tabela *pacienti\_aleg\_analize*. De asemenea, înainte se va verifica corectitudinea datelor manipulate .

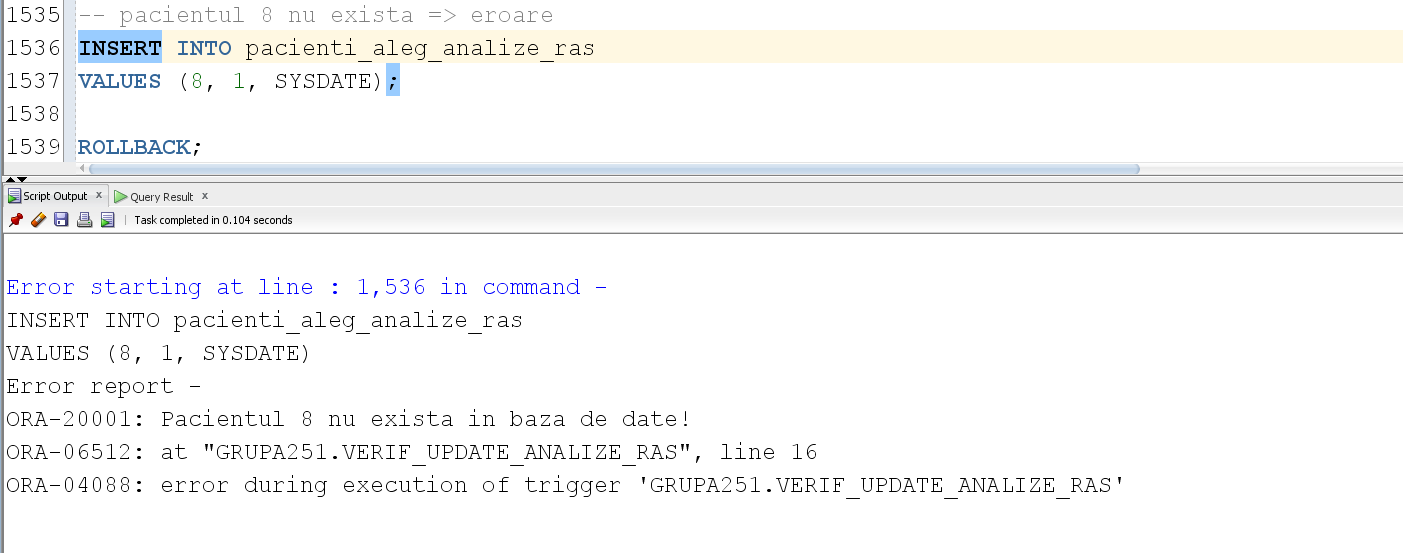
Vom adăuga triggerul *update\_analize\_ras* pentru rezolvarea cerinței, iar pentru a menține integritatea bazei de date, vom mai avea încă un trigger *verif\_update\_analize\_ras*, care verifică corectitudinea datelor manipulate.

|  |
| --- |
| ALTER TABLE pacienti\_ras  ADD nr\_analize NUMBER;  UPDATE pacienti\_ras p  SET nr\_analize = (SELECT COUNT(\*)  FROM pacienti\_aleg\_analize\_ras  WHERE id\_pacient = p.id);    SELECT \*  FROM pacienti\_ras;  CREATE OR REPLACE TRIGGER verif\_update\_analize\_ras  BEFORE INSERT OR UPDATE OF id\_pacient, id\_analiza ON pacienti\_aleg\_analize\_ras  FOR EACH ROW  DECLARE  pacient\_inexistent EXCEPTION;  analiza\_inexistenta EXCEPTION;  ct\_pacient NUMBER;  ct\_analiza NUMBER;  BEGIN  SELECT COUNT(\*)  INTO ct\_pacient  FROM pacienti\_ras  WHERE id = :NEW.id\_pacient;    IF ct\_pacient = 0 THEN  RAISE pacient\_inexistent;  END IF;  SELECT COUNT(\*)  INTO ct\_analiza  FROM analize\_ras  WHERE id = :NEW.id\_analiza;    IF ct\_analiza = 0 THEN  RAISE analiza\_inexistenta;  END IF;    EXCEPTION  WHEN pacient\_inexistent THEN  RAISE\_APPLICATION\_ERROR(-20001, 'Pacientul ' || :NEW.id\_pacient || ' nu exista in baza de date!');  WHEN analiza\_inexistenta THEN  RAISE\_APPLICATION\_ERROR(-20002, 'Analiza ' || :NEW.id\_analiza || ' nu exista in baza de date!');  END;  /  CREATE OR REPLACE PROCEDURE actualizeaza\_analize (id\_pacient pacienti\_ras.id%TYPE, ct NUMBER)  IS  BEGIN  UPDATE pacienti\_ras  SET nr\_analize = nr\_analize + ct  WHERE id = id\_pacient;  END actualizeaza\_analize;  /  CREATE OR REPLACE TRIGGER update\_analize\_ras  AFTER INSERT OR DELETE OR UPDATE OF id\_pacient ON pacienti\_aleg\_analize\_ras  FOR EACH ROW  BEGIN  IF DELETING THEN  actualizeaza\_analize(:OLD.id\_pacient, -1);  ELSIF INSERTING THEN  actualizeaza\_analize(:NEW.id\_pacient, 1);  ELSE -- update  actualizeaza\_analize(:OLD.id\_pacient, -1);  actualizeaza\_analize(:NEW.id\_pacient, 1);  END IF;  END;  /  -- pacientul 2 a facut de 3 ori analiza 4  -- are un total de 9 analize  -- => 9 - 3 = 6  DELETE FROM pacienti\_aleg\_analize\_ras  WHERE id\_pacient = 2 AND id\_analiza = 4;  SELECT \*  FROM pacienti\_ras;  ROLLBACK;  -- pacientul 8 nu exista => eroare  INSERT INTO pacienti\_aleg\_analize\_ras  VALUES (8, 1, SYSDATE);  ROLLBACK;  -- pacientul 1 are initial 4 analize  -- => 5 analize  INSERT INTO pacienti\_aleg\_analize\_ras  VALUES (1, 4, TO\_DATE('2024/12/31', 'yyyy/mm/dd'));  SELECT \*  FROM pacienti\_ras;  ROLLBACK;  -- initial pacientul 3 are 3 analize, pacientul 4 are 4 analize  -- dupa => pacientul 3 = 0, pacientul 4 = 7  UPDATE pacienti\_aleg\_analize\_ras  SET id\_pacient = 4  WHERE id\_pacient = 3;  SELECT \*  FROM pacienti\_ras  WHERE id = 3 OR id = 4;  ROLLBACK; |

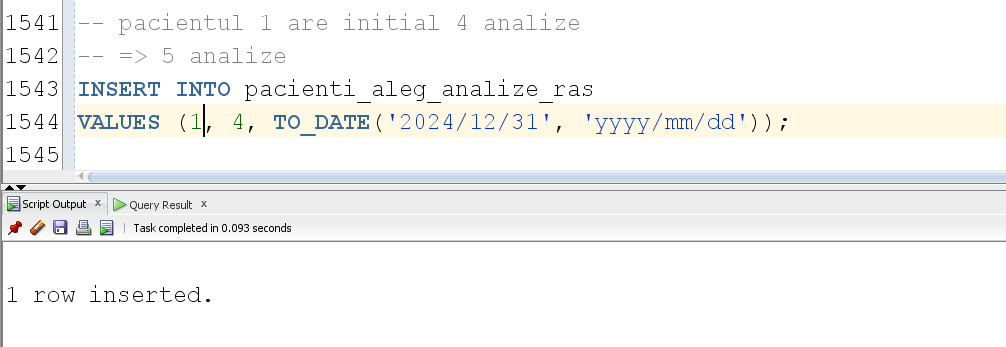


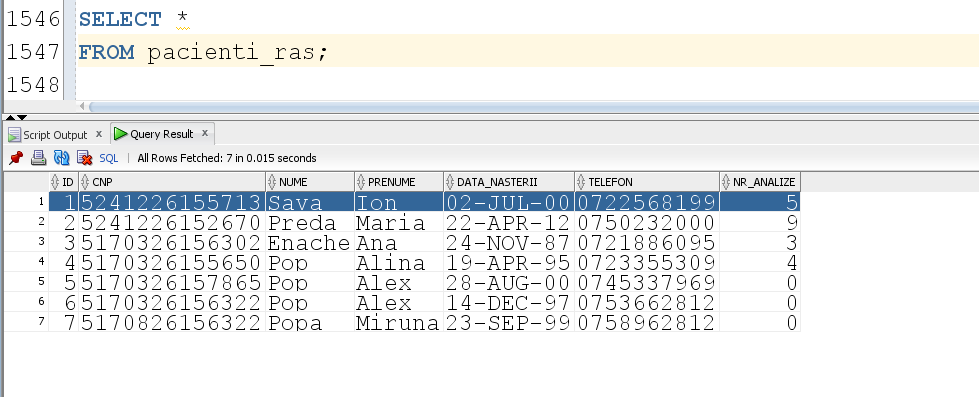
 *Exemplul 1*

**  
 *Exemplul 2*

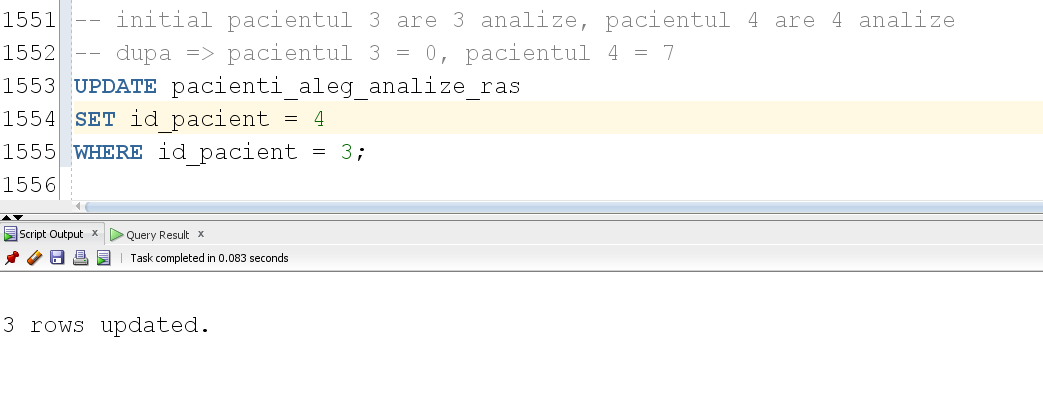


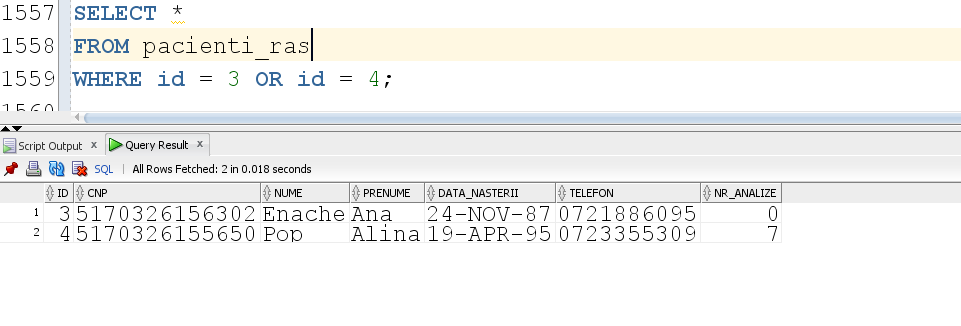
*Exemplul 3*



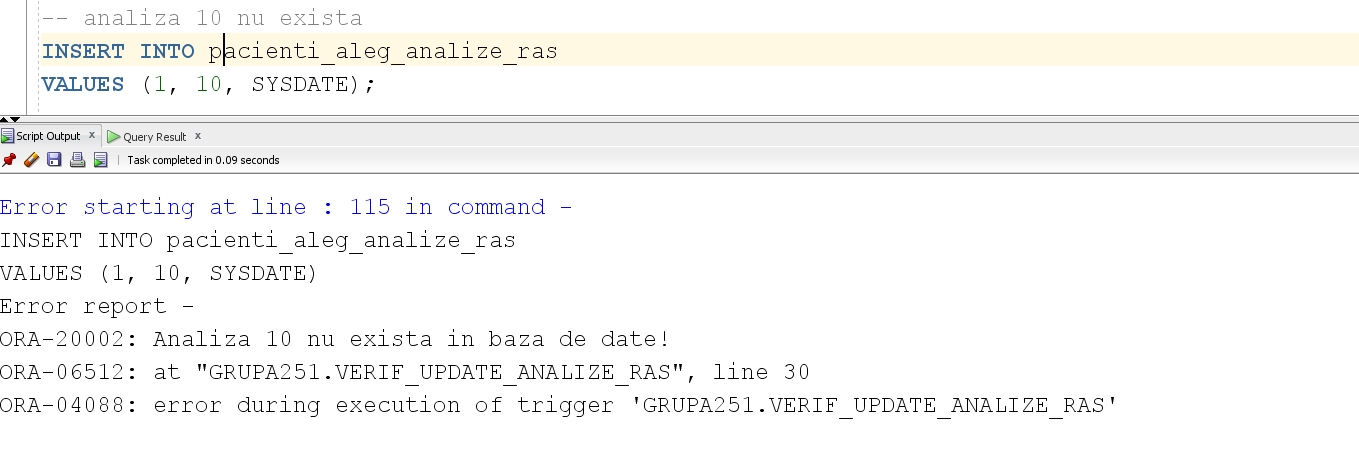


*Exemplul 4*





*Exemplul 5*



### 

### Cerința 12

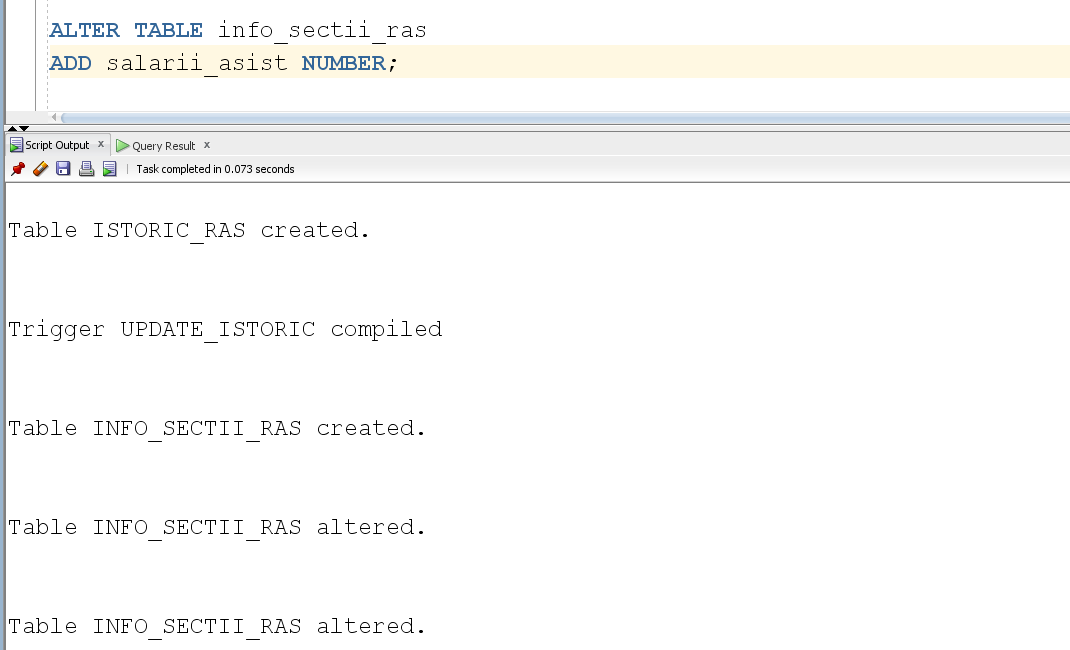
*Cerința*: Definiți un trigger de tip LDD. Declanșați trigger-ul.

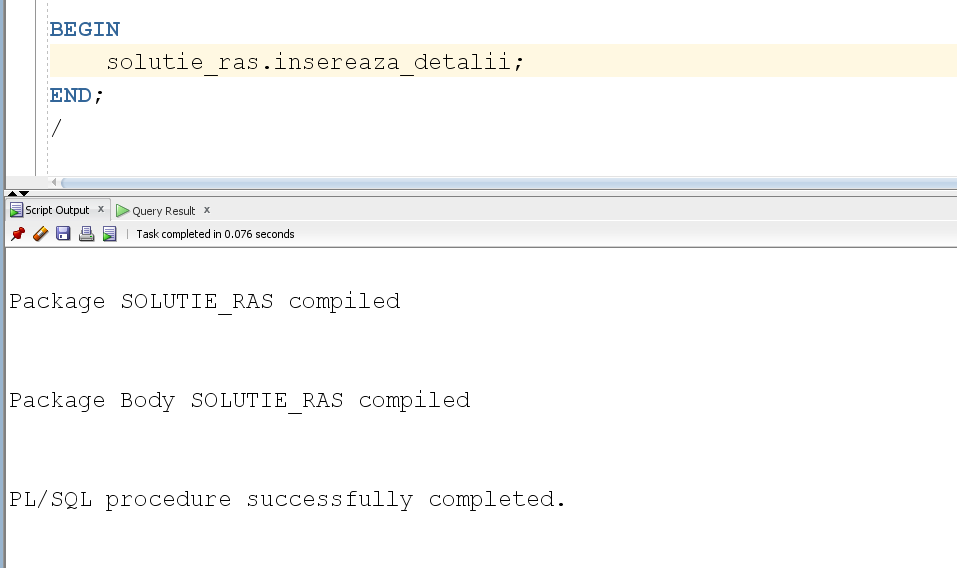
*Rezolvare*: Creați tabelul *istoric\_ras* cu următoarea structură: utilizator (numele utilizatorului), nume\_bd (numele bazei de date), eveniment (evenimentul sistem), nume\_obiect (numele obiectului), tip\_obiect (tipul obiectului), data (data producerii evenimentului).

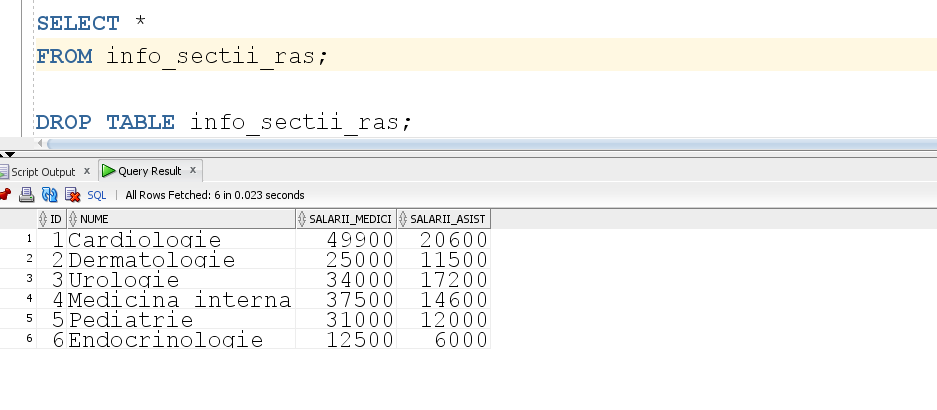
Popularea acestui tabel se va face cu ajutorul unui trigger, care la fiecare comandă de tip LDD (CREATE, ALTER, DROP), inserează în tabel o înregistrare cu detaliile operației.

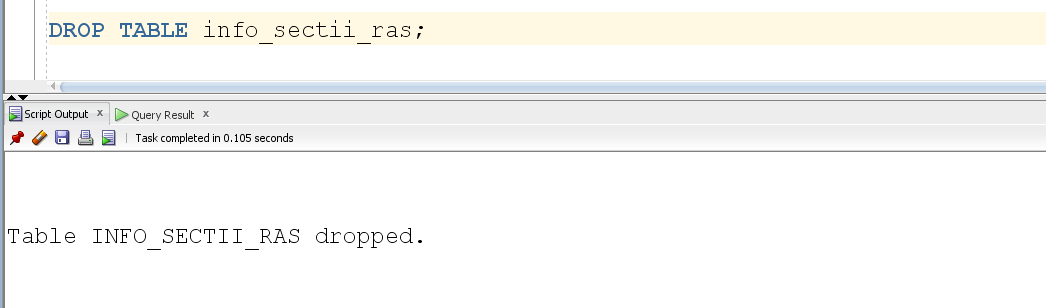
Pentru a vedea modul de funcționare a triggerului și implicit popularea tabelului *istoric\_ras*, vom rezolva următoarea problemă: creați tabelul *info\_sectii\_ras* cu următoarea structură: id-ul secției, numele ei. Adăugați alte 2 coloane pentru suma salariilor medicilor, respectiv suma salariilor asistenților medicali care lucrează pe secția respectivă. Inserați în tabelul creat informațiile necesare. La final, ștergeți tabelul.

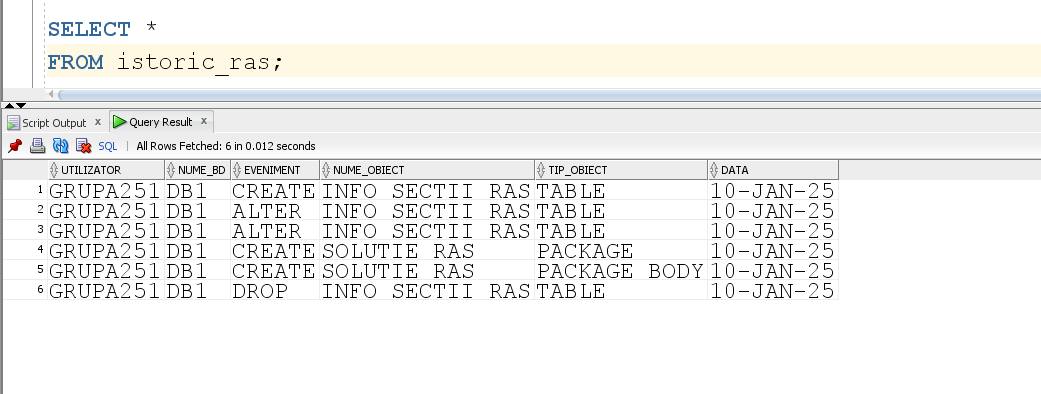
|  |
| --- |
| -- cream tabelul pentru istoric  CREATE TABLE istoric\_ras (utilizator VARCHAR2(50),  nume\_bd VARCHAR2(50),  eveniment VARCHAR2(50),  nume\_obiect VARCHAR2(50),  tip\_obiect VARCHAR2(50),  data DATE);  -- cream triggerul  CREATE OR REPLACE TRIGGER update\_istoric  AFTER CREATE OR DROP OR ALTER ON SCHEMA  BEGIN  INSERT INTO istoric\_ras  VALUES (SYS.LOGIN\_USER, SYS.DATABASE\_NAME, SYS.SYSEVENT, SYS.DICTIONARY\_OBJ\_NAME,  SYS.DICTIONARY\_OBJ\_TYPE, SYSDATE);  END;  /  -- cream tabelul info\_sectii\_ras  CREATE TABLE info\_sectii\_ras (id NUMBER,  nume VARCHAR2(50)  );    -- adaugam cele 2 coloane  ALTER TABLE info\_sectii\_ras  ADD salarii\_medici NUMBER;  ALTER TABLE info\_sectii\_ras  ADD salarii\_asist NUMBER;  CREATE OR REPLACE PACKAGE solutie\_ras  AS  CURSOR sectii IS  SELECT id, denumire  FROM sectie\_ras;  FUNCTION suma\_salarii\_medici(id\_sectie sectie\_ras.id%TYPE)  RETURN NUMBER;  FUNCTION suma\_salarii\_asist (id\_sectie sectie\_ras.id%TYPE)  RETURN NUMBER;  PROCEDURE insereaza\_detalii;  END solutie\_ras;  /  CREATE OR REPLACE PACKAGE BODY solutie\_ras  AS  FUNCTION suma\_salarii\_medici(id\_sectie sectie\_ras.id%TYPE)  RETURN NUMBER  IS  total NUMBER;  BEGIN  SELECT SUM(salariu)  INTO total  FROM medici\_ras m JOIN angajati\_ras a ON (m.id = a.id)  WHERE m.id\_sectie = id\_sectie;  RETURN total;  END suma\_salarii\_medici;    FUNCTION suma\_salarii\_asist (id\_sectie sectie\_ras.id%TYPE)  RETURN NUMBER  IS  total NUMBER;  BEGIN  SELECT SUM(salariu)  INTO total  FROM asistenti\_medicali\_ras asist JOIN angajati\_ras a ON (asist.id = a.id)  WHERE asist.id\_sectie = id\_sectie;  RETURN total;  END suma\_salarii\_asist;    PROCEDURE insereaza\_detalii  IS  v\_id NUMBER;  v\_denumire VARCHAR2(50);  v\_sal\_medici NUMBER;  v\_sal\_asist NUMBER;  BEGIN  OPEN sectii;  LOOP  FETCH sectii INTO v\_id, v\_denumire;  EXIT WHEN sectii%NOTFOUND;  v\_sal\_medici := suma\_salarii\_medici(v\_id);  v\_sal\_asist := suma\_salarii\_asist(v\_id);  INSERT INTO info\_sectii\_ras  VALUES (v\_id, v\_denummire, v\_sal\_medici, v\_sal\_asist);  END LOOP;  CLOSE sectii;    END insereaza\_detalii;    END solutie\_ras;  /  BEGIN  solutie\_ras.insereaza\_detalii;  END;  /  SELECT \*  FROM info\_sectii\_ras;  DROP TABLE info\_sectii\_ras;  SELECT \*  FROM istoric\_ras; |









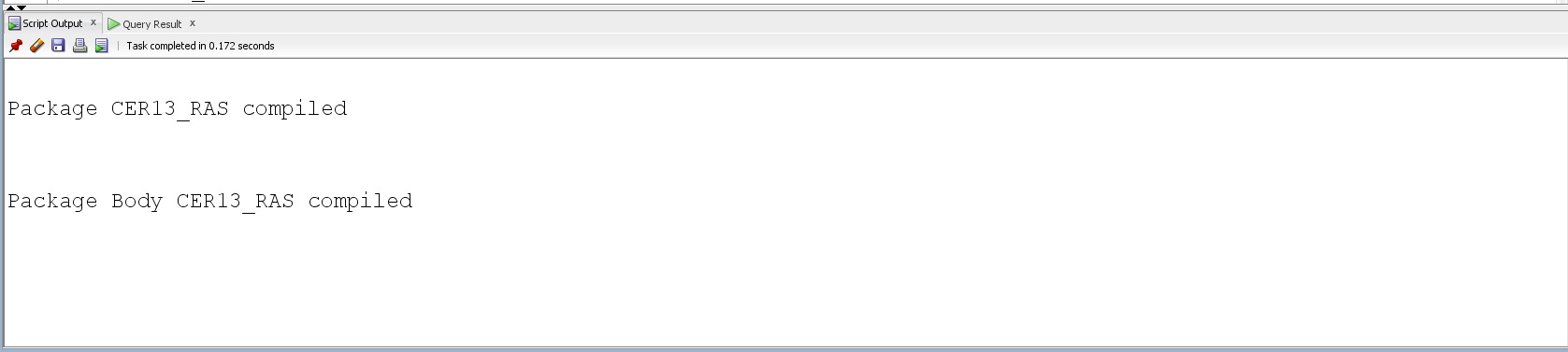


### Cerința 13

*Cerința*: Formulați în limbaj natural o problemă pe care să o rezolvați folosind 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).

*Rezolvare*: Definiți un pachet prin intermediul căruia să afișați pentru fiecare secție numărul de medici, numărul de pacienți care au realizat proceduri specifice secției, lista medicilor, dar și lista pacienților, alături de procedurile medicale realizate în cadrul secției.

|  |
| --- |
| SET SERVEROUTPUT ON;  CREATE OR REPLACE PACKAGE cer13\_ras AS  -- tipuri de date  TYPE rec\_sectie IS RECORD (id sectie\_ras.id%TYPE,  nume sectie\_ras.denumire%TYPE);  TYPE info\_sectii IS TABLE OF rec\_sectie;  TYPE tabl\_id\_pacienti IS TABLE OF NUMBER INDEX BY PLS\_INTEGER;  TYPE rec\_pacient IS RECORD (id pacienti\_ras.id%TYPE,  nume pacienti\_ras.nume%TYPE,  prenume pacienti\_ras.prenume%TYPE);  TYPE info\_pacienti IS TABLE OF rec\_pacient;    CURSOR nume\_proceduri (v\_id\_pacient pacienti\_ras.id%TYPE, v\_id\_sectie sectie\_ras.id%TYPE)  IS  SELECT DISTINCT proc.denumire  FROM pacient\_medic\_proced\_ras asoc JOIN proceduri\_ras proc ON (asoc.id\_proced = proc.id)  WHERE asoc.id\_pacient = v\_id\_pacient AND proc.id\_sectie = v\_id\_sectie;    CURSOR lista\_medici(v\_id\_sectie sectie\_ras.id%TYPE)  IS  SELECT nume, prenume  FROM medici\_ras m JOIN angajati\_ras a ON (m.id = a.id)  WHERE id\_sectie = v\_id\_sectie;    -- functii & proceduri  FUNCTION nr\_medici (v\_id\_sectie sectie\_ras.id%TYPE)  RETURN NUMBER;    FUNCTION nr\_pacienti (v\_id\_sectie sectie\_ras.id%TYPE)  RETURN NUMBER;  -- ret id, nume, prenume pentru pacientii care au avut proceduri la sectia respectiva  FUNCTION selecteaza\_pacienti(v\_id sectie\_ras.id%TYPE)  RETURN info\_pacienti;  -- functia main a pachetului  PROCEDURE afiseaza\_info\_sectii;  -- ret id, nume sectie  FUNCTION selecteaza\_sectii  RETURN info\_sectii;  PROCEDURE afiseaza\_info\_generale(id\_sectie sectie\_ras.id%TYPE);  PROCEDURE afiseaza\_info\_pacienti(id\_sectie sectie\_ras.id%TYPE);  PROCEDURE afiseaza\_medici(id\_sectie sectie\_ras.id%TYPE);  END cer13\_ras;  /  CREATE OR REPLACE PACKAGE BODY cer13\_ras AS  FUNCTION nr\_medici (v\_id\_sectie sectie\_ras.id%TYPE)  RETURN NUMBER  IS  ct NUMBER;  BEGIN  SELECT COUNT(\*)  INTO ct  FROM medici\_ras  WHERE v\_id\_sectie = id\_sectie;    RETURN ct;  END nr\_medici;    FUNCTION nr\_pacienti(v\_id\_sectie sectie\_ras.id%TYPE)  RETURN NUMBER  IS  lista\_pacienti tabl\_id\_pacienti;  BEGIN  -- selectam pacientii care au realizat proceduri la sectia respectiva  SELECT DISTINCT asoc.id\_pacient  BULK COLLECT INTO lista\_pacienti  FROM pacient\_medic\_proced\_ras asoc JOIN proceduri\_ras proc ON (asoc.id\_proced = proc.id)  WHERE proc.id\_sectie = v\_id\_sectie;    -- returnam numarul de pacienti  RETURN lista\_pacienti.COUNT;  END nr\_pacienti;    PROCEDURE afiseaza\_medici(id\_sectie sectie\_ras.id%TYPE)  IS  v\_nume angajati\_ras.nume%TYPE;  v\_prenume angajati\_ras.prenume%TYPE;  BEGIN  OPEN lista\_medici(id\_sectie);  DBMS\_OUTPUT.PUT\_LINE('...........................................');  DBMS\_OUTPUT.PUT\_LINE('Lista medicilor: ');  LOOP  FETCH lista\_medici INTO v\_nume, v\_prenume;  EXIT WHEN lista\_medici%NOTFOUND;  DBMS\_OUTPUT.PUT\_LINE(v\_nume || ' ' || v\_prenume);  END LOOP;  CLOSE lista\_medici;  END afiseaza\_medici;      PROCEDURE afiseaza\_info\_generale(id\_sectie sectie\_ras.id%TYPE)  IS  v\_medici NUMBER;  v\_pacienti NUMBER;  BEGIN  v\_medici := nr\_medici(id\_sectie);  v\_pacienti := nr\_pacienti(id\_sectie);  DBMS\_OUTPUT.PUT\_LINE('Nr medici: ' || v\_medici || ' Nr pacienti: ' || v\_pacienti);  IF v\_medici > 0 THEN  afiseaza\_medici(id\_sectie);  END IF;  END afiseaza\_info\_generale;    FUNCTION selecteaza\_sectii  RETURN info\_sectii  IS  lista\_sectii info\_sectii;  BEGIN  SELECT id, denumire  BULK COLLECT INTO lista\_sectii  FROM sectie\_ras;    RETURN lista\_sectii;  END selecteaza\_sectii;    FUNCTION selecteaza\_pacienti(v\_id sectie\_ras.id%TYPE)  RETURN info\_pacienti  IS  lista info\_pacienti;  BEGIN  SELECT DISTINCT p.id, p.nume, p.prenume  BULK COLLECT INTO lista  FROM pacient\_medic\_proced\_ras asoc JOIN proceduri\_ras proc ON (asoc.id\_proced = proc.id)  JOIN pacienti\_ras p ON (asoc.id\_pacient = p.id)  WHERE proc.id\_sectie = v\_id;    RETURN lista;  END selecteaza\_pacienti;    PROCEDURE afiseaza\_info\_pacienti(id\_sectie sectie\_ras.id%TYPE)  IS  lista\_pacienti info\_pacienti;  nume\_proc proceduri\_ras.denumire%TYPE;  BEGIN  lista\_pacienti := selecteaza\_pacienti(id\_sectie);  IF lista\_pacienti.COUNT > 0 THEN  DBMS\_OUTPUT.PUT\_LINE('...............................................');  DBMS\_OUTPUT.PUT\_LINE('Lista pacientilor:');  FOR i IN lista\_pacienti.FIRST..lista\_pacienti.LAST LOOP  DBMS\_OUTPUT.PUT\_LINE(lista\_pacienti(i).nume || ' ' || lista\_pacienti(i).prenume);  OPEN nume\_proceduri(lista\_pacienti(i).id ,id\_sectie);  LOOP  FETCH nume\_proceduri INTO nume\_proc;  EXIT WHEN nume\_proceduri%NOTFOUND;  DBMS\_OUTPUT.PUT\_LINE(' ' || nume\_proc);  END LOOP;  CLOSE nume\_proceduri;  END LOOP;  END IF;  END afiseaza\_info\_pacienti;    PROCEDURE afiseaza\_info\_sectii  IS  lista\_sectii info\_sectii;  BEGIN  lista\_sectii := selecteaza\_sectii;  FOR i IN lista\_sectii.FIRST..lista\_sectii.LAST LOOP  DBMS\_OUTPUT.PUT\_LINE('--------------------------------------------');  DBMS\_OUTPUT.PUT\_LINE('Sectia ' || lista\_sectii(i).nume);  DBMS\_OUTPUT.PUT\_LINE('............................................');  afiseaza\_info\_generale(lista\_sectii(i).id);  afiseaza\_info\_pacienti(lista\_sectii(i).id);  END LOOP;  END afiseaza\_info\_sectii;  END cer13\_ras;  /  BEGIN  cer13\_ras.afiseaza\_info\_sectii;  END; |



*Exemplu*

