PROIECT SGBD

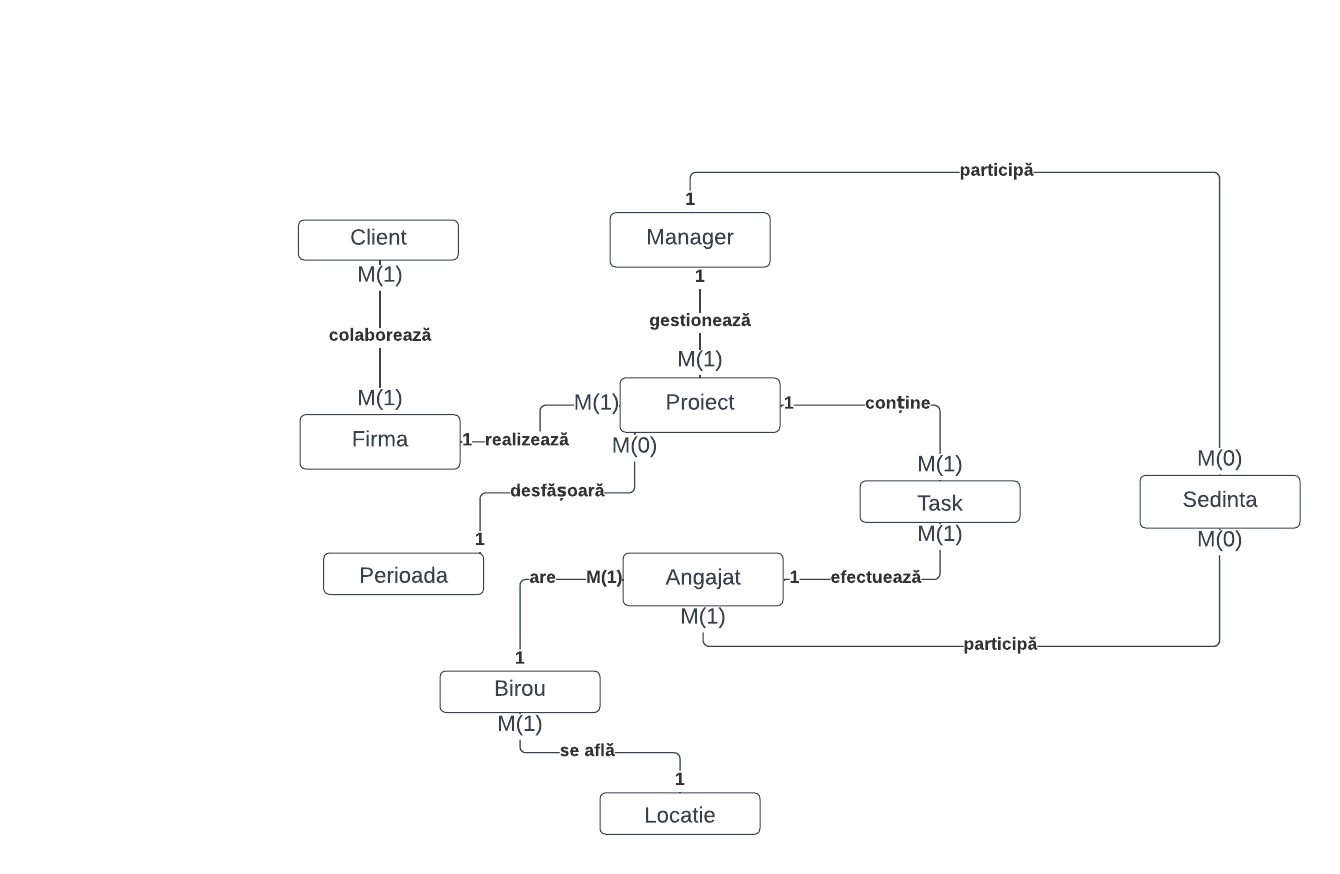
*Tema proiectului* - Managerierea Proiectelor

Nume: Pincu Iulia Maria Andreea  
Grupa: 242

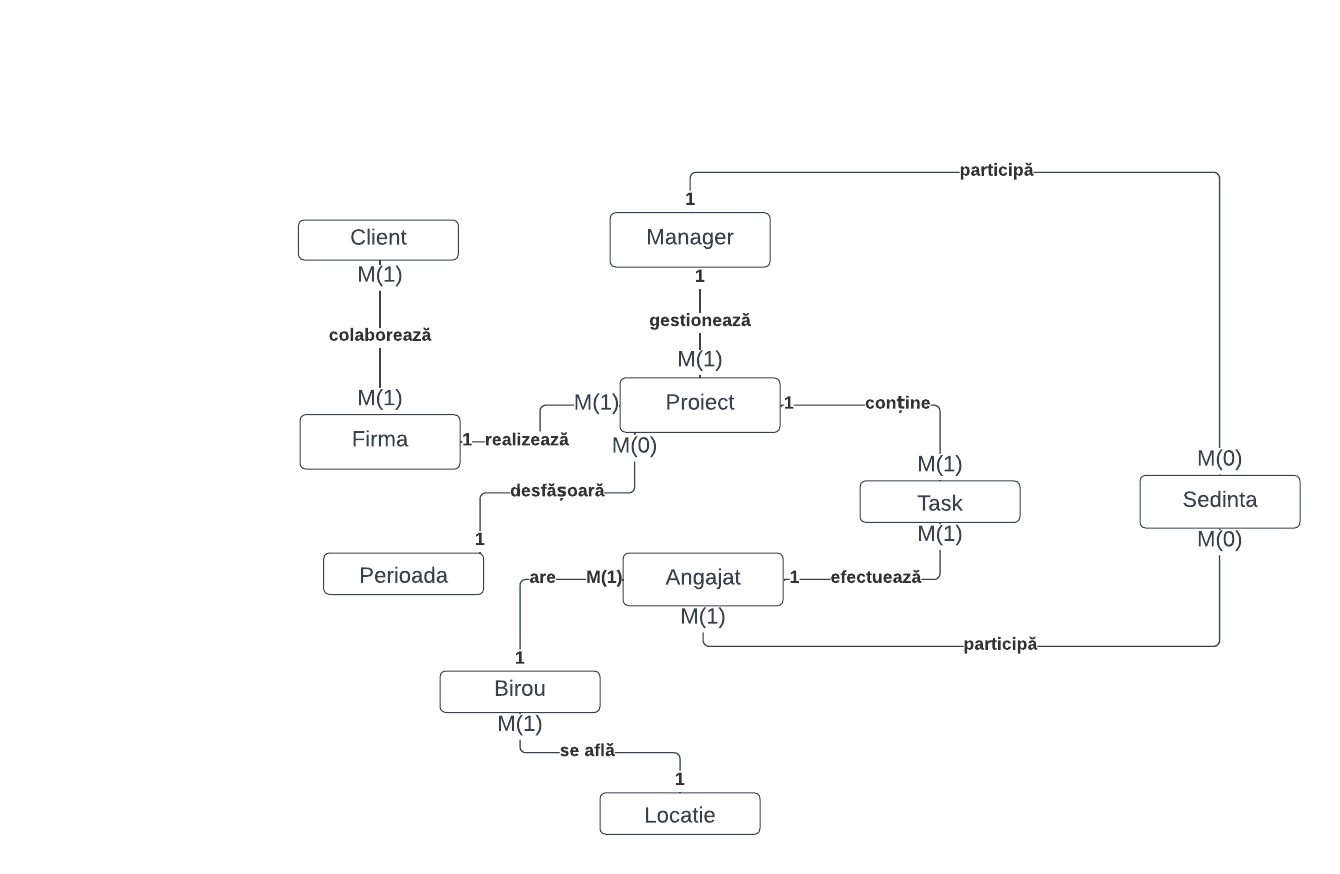
**1.** **Utilitatea bazei de date**

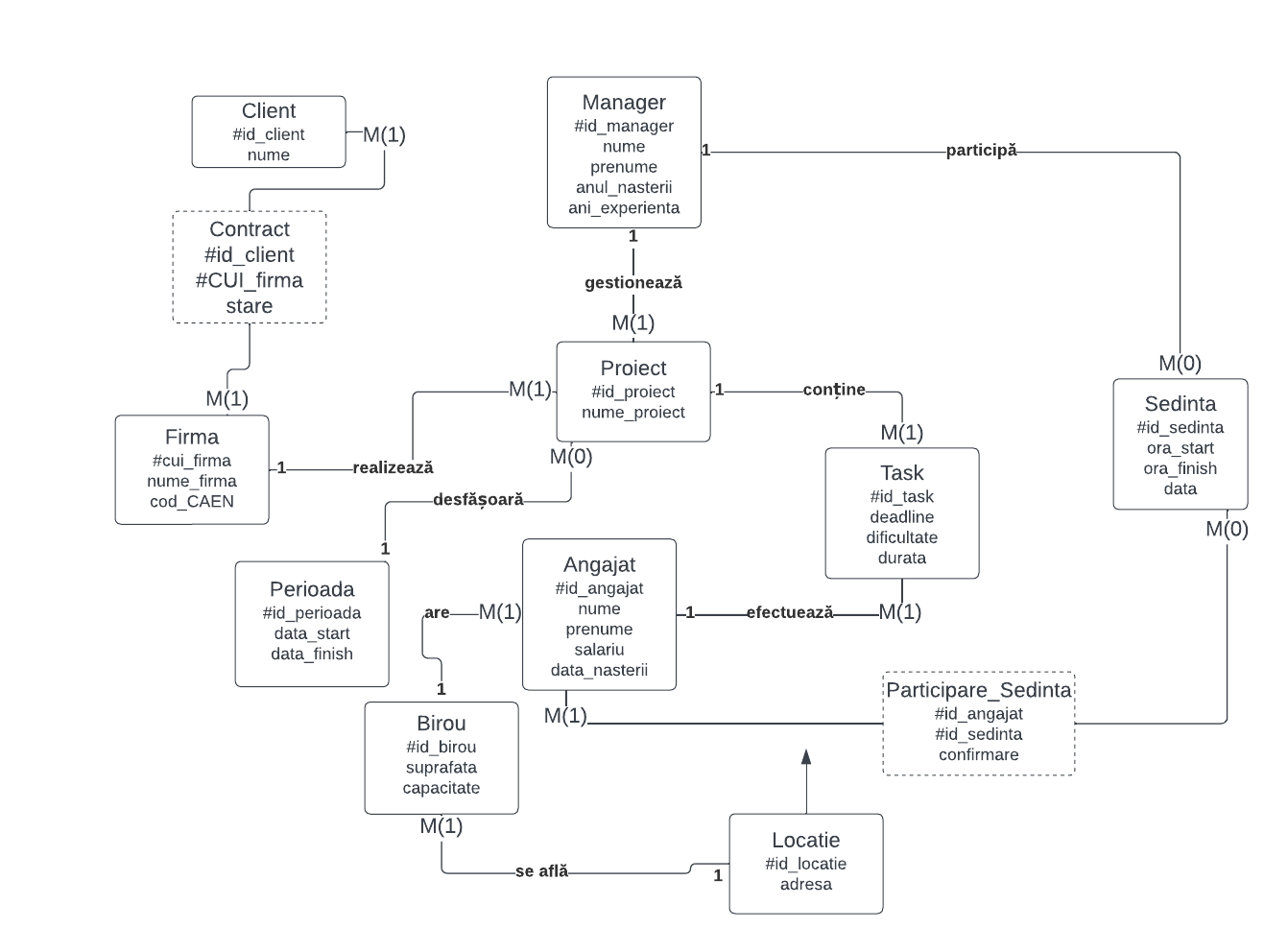
Proiectul este o aplicație de gestionare a proiectelor care implică colaborarea între clienți, firme, angajați și manageri. Scopul acestui proiect este de a oferi o platformă centralizată pentru administrarea și monitorizarea proiectelor într-un mod eficient și organizat.

Aplicația de gestionare a proiectelor facilitează comunicarea și coordonarea între diferitele părți implicate într-un proiect. Ea permite clienților să înregistreze proiectele lor, să monitorizeze stadiile și să primească actualizări în timp real. Firmele pot utiliza aplicația pentru a organiza și aloca resursele necesare proiectelor. Pentru angajați și manageri, aplicația de gestionare a proiectelor reprezintă un instrument esențial pentru administrarea sarcinilor și raportarea progresului. Angajații pot vedea sarcinile alocate, termenele limită și prioritățile acestora, având astfel o imagine clară a responsabilităților lor.

Prin centralizarea datelor și informațiilor într-o bază de date, aplicația de gestionare a proiectelor permite accesul rapid și ușor la informații actualizate despre proiecte, clienți, firme, angajați și alte detalii relevante. Astfel, se asigură transparența, eficiența și coerența în gestionarea proiectelor și se optimizează procesele de luare a deciziilor.

**2. ERD**



**3. Diagrama conceptuală**

**4. Implementarea bazei de date  
  
CREATE TABLE CLIENT(**

**ID\_CLIENT NUMBER(4) PRIMARY KEY,**

**NUME VARCHAR2(20) NOT NULL**

**);**

**CREATE TABLE FIRMA(**

**CUI\_FIRMA NUMBER(10) PRIMARY KEY,**

**NUME\_FIRMA VARCHAR2(20) NOT NULL,**

**COD\_CAEN NUMBER(10) NOT NULL**

**);**

**CREATE TABLE CONTRACT(**

**ID\_CLIENT NUMBER(4),**

**CUI\_FIRMA NUMBER(10),**

**PRIMARY KEY (ID\_CLIENT, CUI\_FIRMA),**

**FOREIGN KEY (ID\_CLIENT) REFERENCES CLIENT(ID\_CLIENT),**

**FOREIGN KEY (CUI\_FIRMA) REFERENCES FIRMA(CUI\_FIRMA)**

**);**

**CREATE TABLE PERIOADA(**

**ID\_PERIOADA NUMBER(4),**

**DATA\_START DATE DEFAULT SYSDATE,**

**DATA\_FINISH DATE DEFAULT SYSDATE,**

**PRIMARY KEY (ID\_PERIOADA)**

**);**

**CREATE TABLE LOCATIE(**

**ID\_LOCATIE NUMBER(4),**

**ADRESA VARCHAR2(20),**

**PRIMARY KEY (ID\_LOCATIE)**

**);**

**CREATE TABLE BIROU(**

**ID\_BIROU NUMBER(4),**

**ID\_LOCATIE NUMBER(4),**

**CAPACITATE NUMBER(1),**

**PRIMARY KEY (ID\_BIROU),**

**FOREIGN KEY (ID\_LOCATIE) REFERENCES LOCATIE(ID\_LOCATIE)**

**);**

**CREATE TABLE ANGAJAT(**

**ID\_ANGAJAT NUMBER(4),**

**NUME VARCHAR2(10),**

**PRENUME VARCHAR2(10),**

**SALARIU NUMBER(6,2),**

**DATA\_NASTERII DATE,**

**ID\_BIROU NUMBER(4),**

**PRIMARY KEY (ID\_ANGAJAT),**

**FOREIGN KEY (ID\_BIROU) REFERENCES BIROU(ID\_BIROU)**

**);**

**CREATE TABLE MANAGER(**

**ID\_MANAGER NUMBER(4),**

**NUME VARCHAR2(20),**

**PRENUME VARCHAR2(20),**

**DATA\_NASTERII DATE DEFAULT SYSDATE,**

**ANI\_EXPERIENTA NUMBER(4),**

**PRIMARY KEY (ID\_MANAGER)**

**);**

**CREATE TABLE PROIECT(**

**ID\_PROIECT NUMBER(4) PRIMARY KEY,**

**NUME\_PROIECT VARCHAR2(15),**

**ID\_MANAGER NUMBER(4),**

**ID\_PERIOADA NUMBER(4),**

**CUI\_FIRMA NUMBER(10),**

**FOREIGN KEY (ID\_MANAGER) REFERENCES MANAGER(ID\_MANAGER),**

**FOREIGN KEY (ID\_PERIOADA) REFERENCES PERIOADA(ID\_PERIOADA),**

**FOREIGN KEY (CUI\_FIRMA) REFERENCES FIRMA(CUI\_FIRMA)**

**);**

**CREATE TABLE TASK(**

**ID\_TASK NUMBER(4),**

**DEADLINE DATE,**

**DIFICULTATE VARCHAR2(5),**

**DURATA\_ORE NUMBER(4,1),**

**ID\_ANGAJAT NUMBER(4),**

**ID\_PROIECT NUMBER(4),**

**PRIMARY KEY (ID\_TASK),**

**FOREIGN KEY (ID\_ANGAJAT) REFERENCES ANGAJAT(ID\_ANGAJAT),**

**FOREIGN KEY (ID\_PROIECT) REFERENCES PROIECT(ID\_PROIECT)**

**);**

**CREATE TABLE SEDINTA(**

**ID\_SEDINTA NUMBER(4),**

**ORA\_START VARCHAR2(8),**

**ORA\_FINISH VARCHAR2(8),**

**ID\_MANAGER NUMBER(4),**

**DATA DATE DEFAULT SYSDATE,**

**PRIMARY KEY(ID\_SEDINTA),**

**FOREIGN KEY (ID\_MANAGER) REFERENCES MANAGER(ID\_MANAGER)**

**);**

**CREATE TABLE PARTICIPARE\_SEDINTA(**

**ID\_SEDINTA NUMBER(4),**

**ID\_ANGAJAT NUMBER(4),**

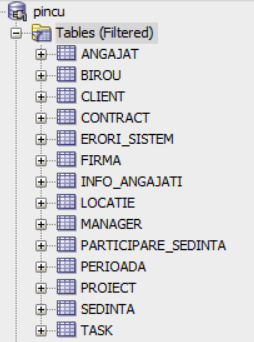
**CONFIRMARE VARCHAR(20),**

**FOREIGN KEY (ID\_SEDINTA) REFERENCES SEDINTA(ID\_SEDINTA),**

**FOREIGN KEY(ID\_ANGAJAT) REFERENCES ANGAJAT(ID\_ANGAJAT)**

**);**

**select \* from PARTICIPARE\_SEDINTA ;**

****

**5. Inserări**

**--INSERARI**

**--CLIENT**

**INSERT INTO CLIENT (ID\_CLIENT, NUME) VALUES (987, 'John');**

**INSERT INTO CLIENT (ID\_CLIENT, NUME) VALUES (234, 'Alice');**

**INSERT INTO CLIENT (ID\_CLIENT, NUME) VALUES (512, 'Michael');**

**INSERT INTO CLIENT (ID\_CLIENT, NUME) VALUES (789, 'Emma');**

**INSERT INTO CLIENT (ID\_CLIENT, NUME) VALUES (345, 'William');**

**INSERT INTO CLIENT (ID\_CLIENT, NUME) VALUES (111, 'Sophia');**

**INSERT INTO CLIENT (ID\_CLIENT, NUME) VALUES (222, 'Oliver');**

**INSERT INTO CLIENT (ID\_CLIENT, NUME) VALUES (333, 'Ava');**

**INSERT INTO CLIENT (ID\_CLIENT, NUME) VALUES (444, 'Liam');**

**INSERT INTO CLIENT (ID\_CLIENT, NUME) VALUES (555, 'Isabella');**

**--FIRMA**

**INSERT INTO FIRMA (CUI\_FIRMA, NUME\_FIRMA, COD\_CAEN) VALUES ('123456789', 'ABC Company', '6201');**

**INSERT INTO FIRMA (CUI\_FIRMA, NUME\_FIRMA, COD\_CAEN) VALUES ('987654321', 'XYZ Corporation', '6202');**

**INSERT INTO FIRMA (CUI\_FIRMA, NUME\_FIRMA, COD\_CAEN) VALUES ('456789123', 'DEF Ltd.', '6203');**

**INSERT INTO FIRMA (CUI\_FIRMA, NUME\_FIRMA, COD\_CAEN) VALUES ('654321987', 'PQR Solutions', '6204');**

**INSERT INTO FIRMA (CUI\_FIRMA, NUME\_FIRMA, COD\_CAEN) VALUES ('789123456', 'MNO Technologies', '6205');**

**--Contract**

**INSERT INTO CONTRACT (ID\_CLIENT, CUI\_FIRMA)**

**VALUES (**

**(SELECT ID\_CLIENT FROM CLIENT WHERE NUME = 'John'),**

**(SELECT CUI\_FIRMA FROM FIRMA WHERE NUME\_FIRMA = 'ABC Company')**

**);**

**INSERT INTO CONTRACT (ID\_CLIENT, CUI\_FIRMA)**

**VALUES (**

**(SELECT ID\_CLIENT FROM CLIENT WHERE NUME = 'Alice'),**

**(SELECT CUI\_FIRMA FROM FIRMA WHERE NUME\_FIRMA = 'XYZ Corporation')**

**);**

**INSERT INTO CONTRACT (ID\_CLIENT, CUI\_FIRMA)**

**VALUES (**

**(SELECT ID\_CLIENT FROM CLIENT WHERE NUME = 'Michael'),**

**(SELECT CUI\_FIRMA FROM FIRMA WHERE NUME\_FIRMA = 'DEF Ltd.')**

**);**

**INSERT INTO CONTRACT (ID\_CLIENT, CUI\_FIRMA)**

**VALUES (**

**(SELECT ID\_CLIENT FROM CLIENT WHERE NUME = 'Emma'),**

**(SELECT CUI\_FIRMA FROM FIRMA WHERE NUME\_FIRMA = 'PQR Solutions')**

**);**

**INSERT INTO CONTRACT (ID\_CLIENT, CUI\_FIRMA)**

**VALUES (**

**(SELECT ID\_CLIENT FROM CLIENT WHERE NUME = 'William'),**

**(SELECT CUI\_FIRMA FROM FIRMA WHERE NUME\_FIRMA = 'MNO Technologies')**

**);**

**INSERT INTO CONTRACT (ID\_CLIENT, CUI\_FIRMA)**

**VALUES (**

**(SELECT ID\_CLIENT FROM CLIENT WHERE NUME = 'Michael'),**

**(SELECT CUI\_FIRMA FROM FIRMA WHERE NUME\_FIRMA = 'ABC Company')**

**);**

**INSERT INTO CONTRACT (ID\_CLIENT, CUI\_FIRMA)**

**VALUES (**

**(SELECT ID\_CLIENT FROM CLIENT WHERE NUME = 'Emma'),**

**(SELECT CUI\_FIRMA FROM FIRMA WHERE NUME\_FIRMA = 'MNO Technologies')**

**);**

**INSERT INTO CONTRACT (ID\_CLIENT, CUI\_FIRMA)**

**VALUES (**

**(SELECT ID\_CLIENT FROM CLIENT WHERE NUME = 'William'),**

**(SELECT CUI\_FIRMA FROM FIRMA WHERE NUME\_FIRMA = 'PQR Solutions')**

**);**

**INSERT INTO CONTRACT (ID\_CLIENT, CUI\_FIRMA)**

**VALUES (**

**(SELECT ID\_CLIENT FROM CLIENT WHERE NUME = 'Sophia'),**

**(SELECT CUI\_FIRMA FROM FIRMA WHERE NUME\_FIRMA = 'ABC Company')**

**);**

**INSERT INTO CONTRACT (ID\_CLIENT, CUI\_FIRMA)**

**VALUES (**

**(SELECT ID\_CLIENT FROM CLIENT WHERE NUME = 'Oliver'),**

**(SELECT CUI\_FIRMA FROM FIRMA WHERE NUME\_FIRMA = 'XYZ Corporation')**

**);**

**INSERT INTO CONTRACT (ID\_CLIENT, CUI\_FIRMA)**

**VALUES (**

**(SELECT ID\_CLIENT FROM CLIENT WHERE NUME = 'Ava'),**

**(SELECT CUI\_FIRMA FROM FIRMA WHERE NUME\_FIRMA = 'DEF Ltd.')**

**);**

**INSERT INTO CONTRACT (ID\_CLIENT, CUI\_FIRMA)**

**VALUES (**

**(SELECT ID\_CLIENT FROM CLIENT WHERE NUME = 'Liam'),**

**(SELECT CUI\_FIRMA FROM FIRMA WHERE NUME\_FIRMA = 'PQR Solutions')**

**);**

**INSERT INTO CONTRACT (ID\_CLIENT, CUI\_FIRMA)**

**VALUES (**

**(SELECT ID\_CLIENT FROM CLIENT WHERE NUME = 'Isabella'),**

**(SELECT CUI\_FIRMA FROM FIRMA WHERE NUME\_FIRMA = 'MNO Technologies')**

**);**

**--Perioada UTILIZAM O SECVENTA PENTRU A INSERA VALORI IN TABEL**

**CREATE SEQUENCE SEQ\_PERIOADA**

**INCREMENT BY 1**

**START WITH 1**

**MAXVALUE 9999**

**NOCYCLE;**

**INSERT INTO PERIOADA (ID\_PERIOADA, DATA\_START, DATA\_FINISH)**

**VALUES (SEQ\_PERIOADA.NEXTVAL, TO\_DATE('01-01-2023', 'DD-MM-YYYY'), TO\_DATE('31-01-2023', 'DD-MM-YYYY'));**

**INSERT INTO PERIOADA (ID\_PERIOADA, DATA\_START, DATA\_FINISH)**

**VALUES (SEQ\_PERIOADA.NEXTVAL, TO\_DATE('01-02-2023', 'DD-MM-YYYY'), TO\_DATE('28-02-2023', 'DD-MM-YYYY'));**

**INSERT INTO PERIOADA (ID\_PERIOADA, DATA\_START, DATA\_FINISH)**

**VALUES (SEQ\_PERIOADA.NEXTVAL, TO\_DATE('01-03-2023', 'DD-MM-YYYY'), TO\_DATE('31-03-2023', 'DD-MM-YYYY'));**

**INSERT INTO PERIOADA (ID\_PERIOADA, DATA\_START, DATA\_FINISH)**

**VALUES (SEQ\_PERIOADA.NEXTVAL, TO\_DATE('01-04-2023', 'DD-MM-YYYY'), TO\_DATE('30-04-2023', 'DD-MM-YYYY'));**

**INSERT INTO PERIOADA (ID\_PERIOADA, DATA\_START, DATA\_FINISH)**

**VALUES (SEQ\_PERIOADA.NEXTVAL, TO\_DATE('01-05-2023', 'DD-MM-YYYY'), TO\_DATE('31-05-2023', 'DD-MM-YYYY'));**

**--Locatie**

**INSERT INTO LOCATIE (ID\_LOCATIE, ADRESA)**

**VALUES (1, '123 Main Street');**

**INSERT INTO LOCATIE (ID\_LOCATIE, ADRESA)**

**VALUES (2, '456 Elm Street');**

**INSERT INTO LOCATIE (ID\_LOCATIE, ADRESA)**

**VALUES (3, '789 Oak Street');**

**INSERT INTO LOCATIE (ID\_LOCATIE, ADRESA)**

**VALUES (4, '321 Pine Street');**

**INSERT INTO LOCATIE (ID\_LOCATIE, ADRESA)**

**VALUES (5, '654 Maple Street');**

**--Birou**

**INSERT INTO BIROU (ID\_BIROU, ID\_LOCATIE, CAPACITATE)**

**VALUES (1, 5,5);**

**INSERT INTO BIROU (ID\_BIROU, ID\_LOCATIE,CAPACITATE)**

**VALUES (2, 4,4);**

**INSERT INTO BIROU (ID\_BIROU, ID\_LOCATIE,CAPACITATE)**

**VALUES (3, 3,7);**

**INSERT INTO BIROU (ID\_BIROU, ID\_LOCATIE,CAPACITATE)**

**VALUES (4, 2,5);**

**INSERT INTO BIROU (ID\_BIROU, ID\_LOCATIE,CAPACITATE)**

**VALUES (5, 1,4);**

**--Angajat**

**INSERT INTO ANGAJAT (ID\_ANGAJAT, NUME, PRENUME, SALARIU, DATA\_NASTERII, ID\_BIROU)**

**VALUES (1, 'Smith', 'Ken', 5000, TO\_DATE('15-05-1990', 'DD-MM-YYYY'), 4);**

**INSERT INTO ANGAJAT (ID\_ANGAJAT, NUME, PRENUME, SALARIU, DATA\_NASTERII, ID\_BIROU)**

**VALUES (2, 'Johnson', 'Emily', 4500, TO\_DATE('22-09-1988', 'DD-MM-YYYY'), 2);**

**INSERT INTO ANGAJAT (ID\_ANGAJAT, NUME, PRENUME, SALARIU, DATA\_NASTERII, ID\_BIROU)**

**VALUES (3, 'Brown', 'David', 5200, TO\_DATE('07-11-1991', 'DD-MM-YYYY'), 3);**

**INSERT INTO ANGAJAT (ID\_ANGAJAT, NUME, PRENUME, SALARIU, DATA\_NASTERII, ID\_BIROU)**

**VALUES (4, 'Wilson', 'Jessica', 4800, TO\_DATE('12-03-1989', 'DD-MM-YYYY'), 4);**

**INSERT INTO ANGAJAT (ID\_ANGAJAT, NUME, PRENUME, SALARIU, DATA\_NASTERII, ID\_BIROU)**

**VALUES (5, 'Miller', 'Andrew', 5100, TO\_DATE('29-07-1992', 'DD-MM-YYYY'), 2);**

**INSERT INTO ANGAJAT (ID\_ANGAJAT, NUME, PRENUME, SALARIU, DATA\_NASTERII, ID\_BIROU)**

**VALUES (6, 'Davis', 'Olivia', 4900, TO\_DATE('05-12-1987', 'DD-MM-YYYY'), 3);**

**INSERT INTO ANGAJAT (ID\_ANGAJAT, NUME, PRENUME, SALARIU, DATA\_NASTERII, ID\_BIROU)**

**VALUES (7, 'Taylor', 'William', 5300, TO\_DATE('18-10-1993', 'DD-MM-YYYY'), 1);**

**INSERT INTO ANGAJAT (ID\_ANGAJAT, NUME, PRENUME, SALARIU, DATA\_NASTERII, ID\_BIROU)**

**VALUES (8, 'Anderson', 'Sophia', 4600, TO\_DATE('03-02-1991', 'DD-MM-YYYY'), 2);**

**INSERT INTO ANGAJAT (ID\_ANGAJAT, NUME, PRENUME, SALARIU, DATA\_NASTERII, ID\_BIROU)**

**VALUES (9, 'Clark', 'Ethan', 4800, TO\_DATE('11-06-1989', 'DD-MM-YYYY'), 1);**

**INSERT INTO ANGAJAT (ID\_ANGAJAT, NUME, PRENUME, SALARIU, DATA\_NASTERII, ID\_BIROU)**

**VALUES (10, 'Walker', 'Ava', 5100, TO\_DATE('26-04-1992', 'DD-MM-YYYY'), 1);**

**INSERT INTO ANGAJAT (ID\_ANGAJAT, NUME, PRENUME, SALARIU, DATA\_NASTERII, ID\_BIROU)**

**VALUES (11, 'Smith', 'John', 4800, TO\_DATE('12-03-1991', 'DD-MM-YYYY'), 4);**

**INSERT INTO ANGAJAT (ID\_ANGAJAT, NUME, PRENUME, SALARIU, DATA\_NASTERII, ID\_BIROU)**

**VALUES (12, 'Johnson', 'Emma', 5100, TO\_DATE('29-07-1988', 'DD-MM-YYYY'), 5);**

**INSERT INTO ANGAJAT (ID\_ANGAJAT, NUME, PRENUME, SALARIU, DATA\_NASTERII, ID\_BIROU)**

**VALUES (13, 'Brown', 'Daniel', 5200, TO\_DATE('07-11-1993', 'DD-MM-YYYY'), 3);**

**INSERT INTO ANGAJAT (ID\_ANGAJAT, NUME, PRENUME, SALARIU, DATA\_NASTERII, ID\_BIROU)**

**VALUES (14, 'Wilson', 'Sophia', 4800, TO\_DATE('12-03-1990', 'DD-MM-YYYY'),5 );**

**INSERT INTO ANGAJAT (ID\_ANGAJAT, NUME, PRENUME, SALARIU, DATA\_NASTERII, ID\_BIROU)**

**VALUES (15, 'Miller', 'Matthew', 5300, TO\_DATE('29-07-1987', 'DD-MM-YYYY'), 5);**

**INSERT INTO ANGAJAT (ID\_ANGAJAT, NUME, PRENUME, SALARIU, DATA\_NASTERII, ID\_BIROU)**

**VALUES (16, 'Davis', 'Olivia', 4900, TO\_DATE('05-12-1992', 'DD-MM-YYYY'),5 );**

**INSERT INTO ANGAJAT (ID\_ANGAJAT, NUME, PRENUME, SALARIU, DATA\_NASTERII, ID\_BIROU)**

**VALUES (17, 'Taylor', 'Benjamin', 5100, TO\_DATE('18-10-1991', 'DD-MM-YYYY'),3 );**

**INSERT INTO ANGAJAT (ID\_ANGAJAT, NUME, PRENUME, SALARIU, DATA\_NASTERII, ID\_BIROU)**

**VALUES (18, 'Anderson', 'Isabella', 4600, TO\_DATE('03-02-1989', 'DD-MM-YYYY'), 1);**

**INSERT INTO ANGAJAT (ID\_ANGAJAT, NUME, PRENUME, SALARIU, DATA\_NASTERII, ID\_BIROU)**

**VALUES (19, 'Clark', 'Ethan', 4800, TO\_DATE('11-06-1994', 'DD-MM-YYYY'), 2);**

**INSERT INTO ANGAJAT (ID\_ANGAJAT, NUME, PRENUME, SALARIU, DATA\_NASTERII, ID\_BIROU)**

**VALUES (20, 'Walker', 'Mia', 5100, TO\_DATE('26-04-1993', 'DD-MM-YYYY'), 3);**

**--Manager**

**INSERT INTO MANAGER (ID\_MANAGER, NUME, PRENUME, DATA\_NASTERII, ANI\_EXPERIENTA)**

**VALUES**

**(1, 'Popescu', 'Ion', TO\_DATE('01-01-1985', 'DD-MM-YYYY'), 6);**

**INSERT INTO MANAGER (ID\_MANAGER, NUME, PRENUME, DATA\_NASTERII, ANI\_EXPERIENTA)**

**VALUES**

**(2, 'Ionescu', 'Maria', TO\_DATE('15-07-1990', 'DD-MM-YYYY'), 4);**

**INSERT INTO MANAGER (ID\_MANAGER, NUME, PRENUME, DATA\_NASTERII, ANI\_EXPERIENTA)**

**VALUES**

**(3, 'Constantin', 'Andrei', TO\_DATE('10-03-1988', 'DD-MM-YYYY'), 8);**

**INSERT INTO MANAGER (ID\_MANAGER, NUME, PRENUME, DATA\_NASTERII, ANI\_EXPERIENTA)**

**VALUES**

**(4, 'Georgescu', 'Elena', TO\_DATE('05-09-1993', 'DD-MM-YYYY'), 3);**

**INSERT INTO MANAGER (ID\_MANAGER, NUME, PRENUME, DATA\_NASTERII, ANI\_EXPERIENTA)**

**VALUES**

**(5, 'Popa', 'Mihai', TO\_DATE('20-11-1982', 'DD-MM-YYYY'), 9);**

**--PROIECT**

**INSERT INTO PROIECT (ID\_PROIECT, NUME\_PROIECT, ID\_MANAGER, ID\_PERIOADA, CUI\_FIRMA)**

**VALUES (1, 'Iscoada', 1, 1, 123456789);**

**INSERT INTO PROIECT (ID\_PROIECT, NUME\_PROIECT, ID\_MANAGER, ID\_PERIOADA, CUI\_FIRMA)**

**VALUES (2, 'Stefanini Dev', 2, 2, 987654321);**

**INSERT INTO PROIECT (ID\_PROIECT, NUME\_PROIECT, ID\_MANAGER, ID\_PERIOADA, CUI\_FIRMA)**

**VALUES (3, 'Vinarte', 3, 3, 456789123);**

**INSERT INTO PROIECT (ID\_PROIECT, NUME\_PROIECT, ID\_MANAGER, ID\_PERIOADA, CUI\_FIRMA)**

**VALUES (4, 'CCS', 4, 4, 654321987);**

**INSERT INTO PROIECT (ID\_PROIECT, NUME\_PROIECT, ID\_MANAGER, ID\_PERIOADA, CUI\_FIRMA)**

**VALUES (5, 'Fashion House', 5, 5, 789123456);**

**--Task--task**

**INSERT INTO TASK (ID\_TASK, DEADLINE, DIFICULTATE, DURATA\_ORE, ID\_ANGAJAT, ID\_PROIECT)**

**VALUES**

**(1, TO\_DATE('22-05-2023', 'DD-MM-YYYY'), 'USOR', 3.5,4,2);**

**INSERT INTO TASK (ID\_TASK, DEADLINE, DIFICULTATE, DURATA\_ORE, ID\_ANGAJAT,ID\_PROIECT)**

**VALUES**

**(2, TO\_DATE('23-05-2023', 'DD-MM-YYYY'), 'MEDIU', 5.2, 3,3);**

**INSERT INTO TASK (ID\_TASK, DEADLINE, DIFICULTATE, DURATA\_ORE, ID\_ANGAJAT,ID\_PROIECT)**

**VALUES**

**(3, TO\_DATE('24-05-2023', 'DD-MM-YYYY'), 'GREU', 8.0, 7,4);**

**INSERT INTO TASK (ID\_TASK, DEADLINE, DIFICULTATE, DURATA\_ORE, ID\_ANGAJAT,ID\_PROIECT)**

**VALUES**

**(4, TO\_DATE('25-05-2023', 'DD-MM-YYYY'), 'USOR', 2.1, 6,3);**

**INSERT INTO TASK (ID\_TASK, DEADLINE, DIFICULTATE, DURATA\_ORE, ID\_ANGAJAT,ID\_PROIECT)**

**VALUES**

**(5, TO\_DATE('26-05-2023', 'DD-MM-YYYY'), 'MEDIU', 6.3,1,2 );**

**INSERT INTO TASK (ID\_TASK, DEADLINE, DIFICULTATE, DURATA\_ORE, ID\_ANGAJAT,ID\_PROIECT)**

**VALUES**

**(6, TO\_DATE('27-05-2023', 'DD-MM-YYYY'), 'GREU', 7.8, 13,3);**

**INSERT INTO TASK (ID\_TASK, DEADLINE, DIFICULTATE, DURATA\_ORE, ID\_ANGAJAT,ID\_PROIECT)**

**VALUES**

**(7, TO\_DATE('28-05-2023', 'DD-MM-YYYY'), 'USOR', 4.2, 17,3);**

**INSERT INTO TASK (ID\_TASK, DEADLINE, DIFICULTATE, DURATA\_ORE, ID\_ANGAJAT,ID\_PROIECT)**

**VALUES**

**(8, TO\_DATE('29-05-2023', 'DD-MM-YYYY'), 'MEDIU', 5.9, 11,2);**

**INSERT INTO TASK (ID\_TASK, DEADLINE, DIFICULTATE, DURATA\_ORE, ID\_ANGAJAT,ID\_PROIECT)**

**VALUES**

**(9, TO\_DATE('30-05-2023', 'DD-MM-YYYY'), 'GREU', 9.5,4,2);**

**INSERT INTO TASK (ID\_TASK, DEADLINE, DIFICULTATE, DURATA\_ORE, ID\_ANGAJAT,ID\_PROIECT)**

**VALUES**

**(10, TO\_DATE('31-05-2023', 'DD-MM-YYYY'), 'USOR', 2.8,9,4 );**

**INSERT INTO TASK (ID\_TASK, DEADLINE, DIFICULTATE, DURATA\_ORE, ID\_ANGAJAT,ID\_PROIECT)**

**VALUES**

**(11, TO\_DATE('01-06-2023', 'DD-MM-YYYY'), 'MEDIU', 6.1,20,3);**

**INSERT INTO TASK (ID\_TASK, DEADLINE, DIFICULTATE, DURATA\_ORE, ID\_ANGAJAT,ID\_PROIECT)**

**VALUES**

**(12, TO\_DATE('02-06-2023', 'DD-MM-YYYY'), 'GREU', 8.4, 12,5);**

**INSERT INTO TASK (ID\_TASK, DEADLINE, DIFICULTATE, DURATA\_ORE, ID\_ANGAJAT,ID\_PROIECT)**

**VALUES**

**(13, TO\_DATE('03-06-2023', 'DD-MM-YYYY'), 'USOR', 3.2,14,5 );**

**INSERT INTO TASK (ID\_TASK, DEADLINE, DIFICULTATE, DURATA\_ORE, ID\_ANGAJAT,ID\_PROIECT)**

**VALUES**

**(14, TO\_DATE('04-06-2023', 'DD-MM-YYYY'), 'MEDIU', 5.7, 15,5);**

**INSERT INTO TASK (ID\_TASK, DEADLINE, DIFICULTATE, DURATA\_ORE, ID\_ANGAJAT,ID\_PROIECT)**

**VALUES**

**(15, TO\_DATE('05-06-2023', 'DD-MM-YYYY'), 'GREU', 7.2,16,5 );**

**INSERT INTO TASK (ID\_TASK, DEADLINE, DIFICULTATE, DURATA\_ORE, ID\_ANGAJAT,ID\_PROIECT)**

**VALUES**

**(16, TO\_DATE('06-06-2023', 'DD-MM-YYYY'), 'USOR', 2.9, 15,5);**

**INSERT INTO TASK (ID\_TASK, DEADLINE, DIFICULTATE, DURATA\_ORE, ID\_ANGAJAT,ID\_PROIECT)**

**VALUES**

**(17, TO\_DATE('07-06-2023', 'DD-MM-YYYY'), 'MEDIU', 6.5, 12,5);**

**INSERT INTO TASK (ID\_TASK, DEADLINE, DIFICULTATE, DURATA\_ORE, ID\_ANGAJAT,ID\_PROIECT)**

**VALUES**

**(18, TO\_DATE('08-06-2023', 'DD-MM-YYYY'), 'GREU', 8.8, 14,5);**

**INSERT INTO TASK (ID\_TASK, DEADLINE, DIFICULTATE, DURATA\_ORE, ID\_ANGAJAT,ID\_PROIECT)**

**VALUES**

**(19, TO\_DATE('09-06-2023', 'DD-MM-YYYY'), 'USOR', 4.1, 19,1);**

**INSERT INTO TASK (ID\_TASK, DEADLINE, DIFICULTATE, DURATA\_ORE, ID\_ANGAJAT,ID\_PROIECT)**

**VALUES**

**(20, TO\_DATE('10-06-2023', 'DD-MM-YYYY'), 'MEDIU', 5.6,16,5 );**

**INSERT INTO TASK (ID\_TASK, DEADLINE, DIFICULTATE, DURATA\_ORE, ID\_ANGAJAT,ID\_PROIECT)**

**VALUES**

**(21, TO\_DATE('11-06-2023', 'DD-MM-YYYY'), 'GREU', 7.9,18,4 );**

**INSERT INTO TASK (ID\_TASK, DEADLINE, DIFICULTATE, DURATA\_ORE, ID\_ANGAJAT,ID\_PROIECT)**

**VALUES**

**(22, TO\_DATE('12-06-2023', 'DD-MM-YYYY'), 'USOR', 3.8, 9,4);**

**INSERT INTO TASK (ID\_TASK, DEADLINE, DIFICULTATE, DURATA\_ORE, ID\_ANGAJAT,ID\_PROIECT)**

**VALUES**

**(23, TO\_DATE('13-06-2023', 'DD-MM-YYYY'), 'MEDIU', 6.7, 14,5);**

**INSERT INTO TASK (ID\_TASK, DEADLINE, DIFICULTATE, DURATA\_ORE, ID\_ANGAJAT,ID\_PROIECT)**

**VALUES**

**(24, TO\_DATE('14-06-2023', 'DD-MM-YYYY'), 'GREU', 9.2,5,1);**

**INSERT INTO TASK (ID\_TASK, DEADLINE, DIFICULTATE, DURATA\_ORE, ID\_ANGAJAT,ID\_PROIECT)**

**VALUES**

**(25, TO\_DATE('15-06-2023', 'DD-MM-YYYY'), 'USOR', 2.5,20,3 );**

**INSERT INTO TASK (ID\_TASK, DEADLINE, DIFICULTATE, DURATA\_ORE, ID\_ANGAJAT,ID\_PROIECT)**

**VALUES**

**(26, TO\_DATE('16-06-2023', 'DD-MM-YYYY'), 'MEDIU', 5.3,2, 1);**

**INSERT INTO TASK (ID\_TASK, DEADLINE, DIFICULTATE, DURATA\_ORE, ID\_ANGAJAT,ID\_PROIECT)**

**VALUES**

**(27, TO\_DATE('17-06-2023', 'DD-MM-YYYY'), 'GREU', 7.6, 19,1);**

**INSERT INTO TASK (ID\_TASK, DEADLINE, DIFICULTATE, DURATA\_ORE, ID\_ANGAJAT,ID\_PROIECT)**

**VALUES**

**(28, TO\_DATE('18-06-2023', 'DD-MM-YYYY'), 'USOR', 3.7,8, 1);**

**INSERT INTO TASK (ID\_TASK, DEADLINE, DIFICULTATE, DURATA\_ORE, ID\_ANGAJAT,ID\_PROIECT)**

**VALUES**

**(29, TO\_DATE('19-06-2023', 'DD-MM-YYYY'), 'MEDIU', 6.4,15,5 );**

**INSERT INTO TASK (ID\_TASK, DEADLINE, DIFICULTATE, DURATA\_ORE, ID\_ANGAJAT,ID\_PROIECT)**

**VALUES**

**(30, TO\_DATE('20-06-2023', 'DD-MM-YYYY'), 'GREU', 8.7,19, 1);**

**--sedinta**

**INSERT INTO SEDINTA (ID\_SEDINTA, ORA\_START, ORA\_FINISH, ID\_MANAGER, DATA)**

**VALUES (1, '09:00', '10:30', 2, TO\_DATE('22-05-2023', 'DD-MM-YYYY'));**

**INSERT INTO SEDINTA (ID\_SEDINTA, ORA\_START, ORA\_FINISH, ID\_MANAGER, DATA)**

**VALUES (2, '14:00', '15:30', 3, TO\_DATE('23-05-2023', 'DD-MM-YYYY'));**

**INSERT INTO SEDINTA (ID\_SEDINTA, ORA\_START, ORA\_FINISH, ID\_MANAGER, DATA)**

**VALUES (3, '11:30', '12:30', 1, TO\_DATE('24-05-2023', 'DD-MM-YYYY'));**

**INSERT INTO SEDINTA (ID\_SEDINTA, ORA\_START, ORA\_FINISH, ID\_MANAGER, DATA)**

**VALUES (4, '16:00', '17:30',5 , TO\_DATE('25-05-2023', 'DD-MM-YYYY'));**

**INSERT INTO SEDINTA (ID\_SEDINTA, ORA\_START, ORA\_FINISH, ID\_MANAGER, DATA)**

**VALUES (5, '09:30', '11:00',4, TO\_DATE('26-05-2023', 'DD-MM-YYYY'));**

**--Participare sedinta**

**--1**

**INSERT INTO PARTICIPARE\_SEDINTA (ID\_SEDINTA, ID\_ANGAJAT, CONFIRMARE)**

**VALUES (3, 2, 'CONFIRMAT');**

**INSERT INTO PARTICIPARE\_SEDINTA (ID\_SEDINTA, ID\_ANGAJAT, CONFIRMARE)**

**VALUES (3, 5, 'NECONFIRMAT');**

**INSERT INTO PARTICIPARE\_SEDINTA (ID\_SEDINTA, ID\_ANGAJAT, CONFIRMARE)**

**VALUES (3, 8, 'CONFIRMAT');**

**INSERT INTO PARTICIPARE\_SEDINTA (ID\_SEDINTA, ID\_ANGAJAT, CONFIRMARE)**

**VALUES (3, 19, 'CONFIRMAT');**

**--2**

**INSERT INTO PARTICIPARE\_SEDINTA (ID\_SEDINTA, ID\_ANGAJAT, CONFIRMARE)**

**VALUES (1, 1, 'CONFIRMAT');**

**INSERT INTO PARTICIPARE\_SEDINTA (ID\_SEDINTA, ID\_ANGAJAT, CONFIRMARE)**

**VALUES (1, 4, 'NECONFIRMAT');**

**INSERT INTO PARTICIPARE\_SEDINTA (ID\_SEDINTA, ID\_ANGAJAT, CONFIRMARE)**

**VALUES (1, 11, 'CONFIRMAT');**

**INSERT INTO PARTICIPARE\_SEDINTA (ID\_SEDINTA, ID\_ANGAJAT, CONFIRMARE)**

**VALUES (2, 3, 'CONFIRMAT');**

**INSERT INTO PARTICIPARE\_SEDINTA (ID\_SEDINTA, ID\_ANGAJAT, CONFIRMARE)**

**VALUES (2, 6, 'NECONFIRMAT');**

**INSERT INTO PARTICIPARE\_SEDINTA (ID\_SEDINTA, ID\_ANGAJAT, CONFIRMARE)**

**VALUES (2, 13, 'CONFIRMAT');**

**INSERT INTO PARTICIPARE\_SEDINTA (ID\_SEDINTA, ID\_ANGAJAT, CONFIRMARE)**

**VALUES (2, 17, 'CONFIRMAT');**

**INSERT INTO PARTICIPARE\_SEDINTA (ID\_SEDINTA, ID\_ANGAJAT, CONFIRMARE)**

**VALUES (2, 20, 'NECONFIRMAT');**

**INSERT INTO PARTICIPARE\_SEDINTA (ID\_SEDINTA, ID\_ANGAJAT, CONFIRMARE)**

**VALUES (5, 7, 'CONFIRMAT');**

**INSERT INTO PARTICIPARE\_SEDINTA (ID\_SEDINTA, ID\_ANGAJAT, CONFIRMARE)**

**VALUES (5, 9, 'CONFIRMAT');**

**INSERT INTO PARTICIPARE\_SEDINTA (ID\_SEDINTA, ID\_ANGAJAT, CONFIRMARE)**

**VALUES (4, 12, 'NECONFIRMAT');**

**INSERT INTO PARTICIPARE\_SEDINTA (ID\_SEDINTA, ID\_ANGAJAT, CONFIRMARE)**

**VALUES (4, 14, 'CONFIRMAT');**

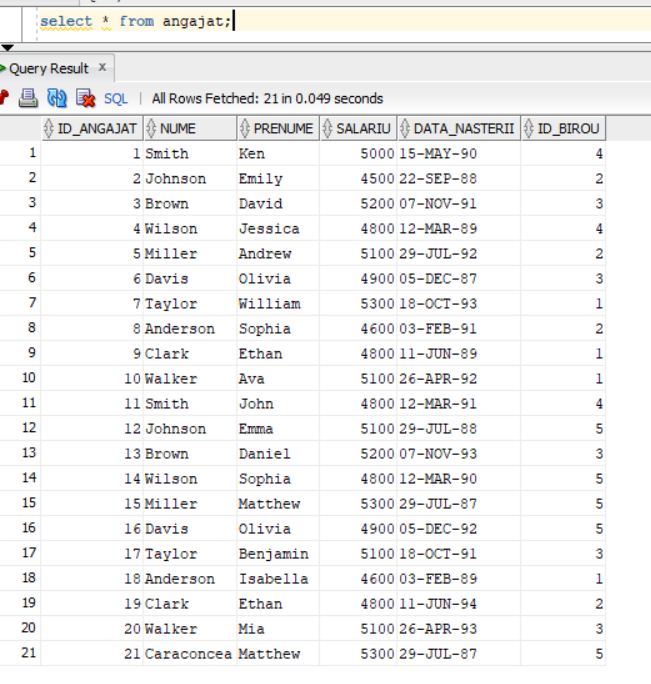
**INSERT INTO PARTICIPARE\_SEDINTA (ID\_SEDINTA, ID\_ANGAJAT, CONFIRMARE)**

**VALUES (4, 15, 'NECONFIRMAT');**

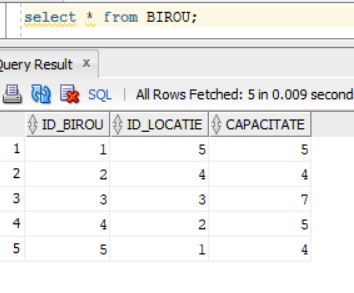
**INSERT INTO PARTICIPARE\_SEDINTA (ID\_SEDINTA, ID\_ANGAJAT, CONFIRMARE)**

**VALUES (4, 16, 'NECONFIRMAT');**

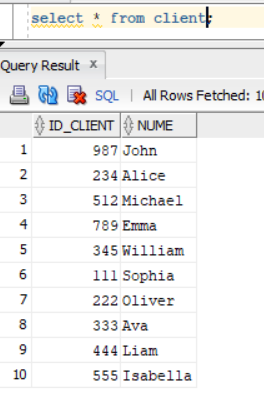
**ANGAJAT:**

****

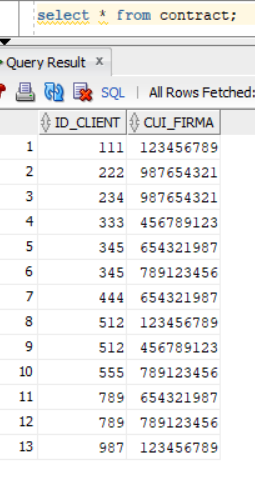
**BIROU:**

****

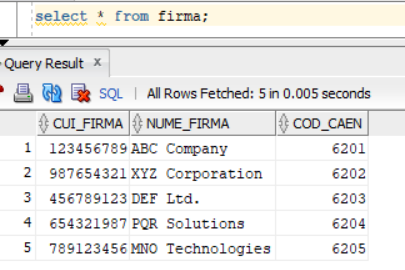
**CLIENT:**

****

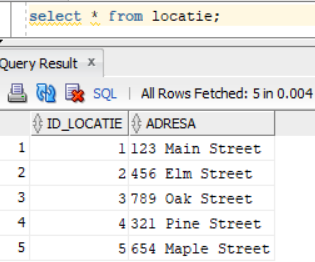
**CONTRACT:**

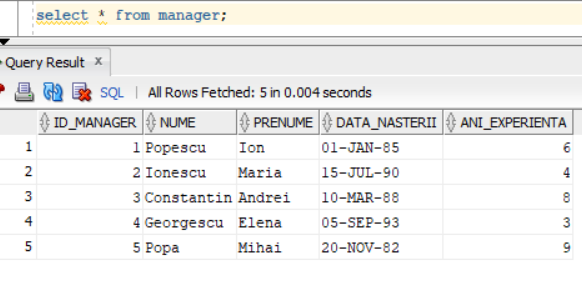
****

**FIRMA:**

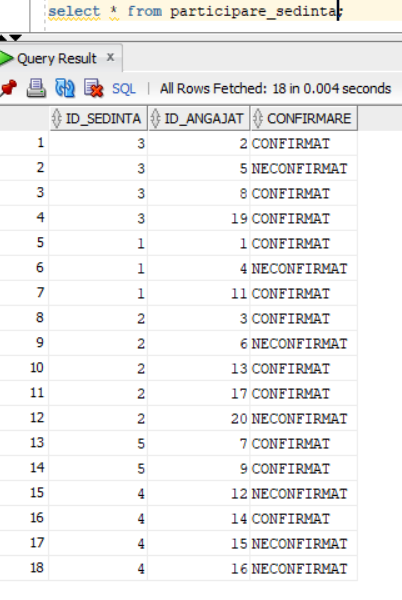
****

**LOCATIE:**

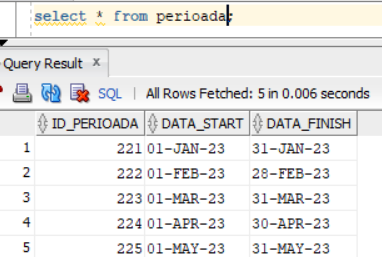
**  
MANAGER:**

****

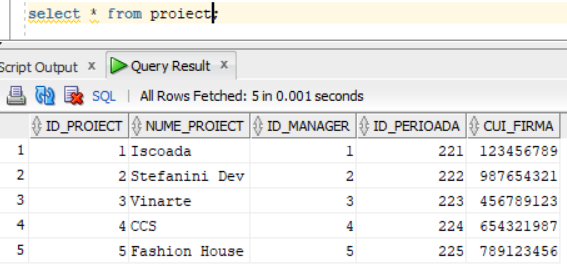
**PARTICIPARE SEDINTA:**

****

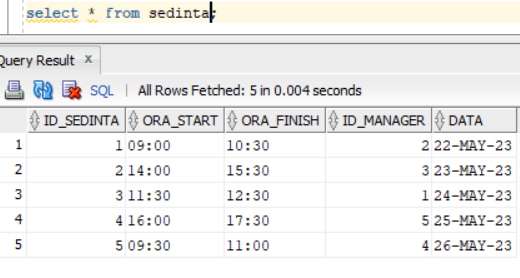
**PERIOADA:**

****

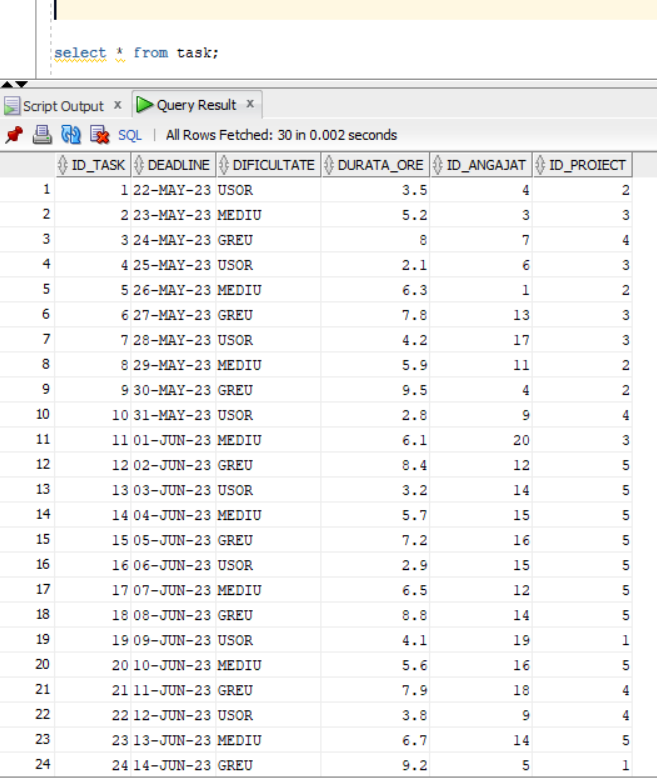
**PROIECT:**

****

**SEDINTA:**

****

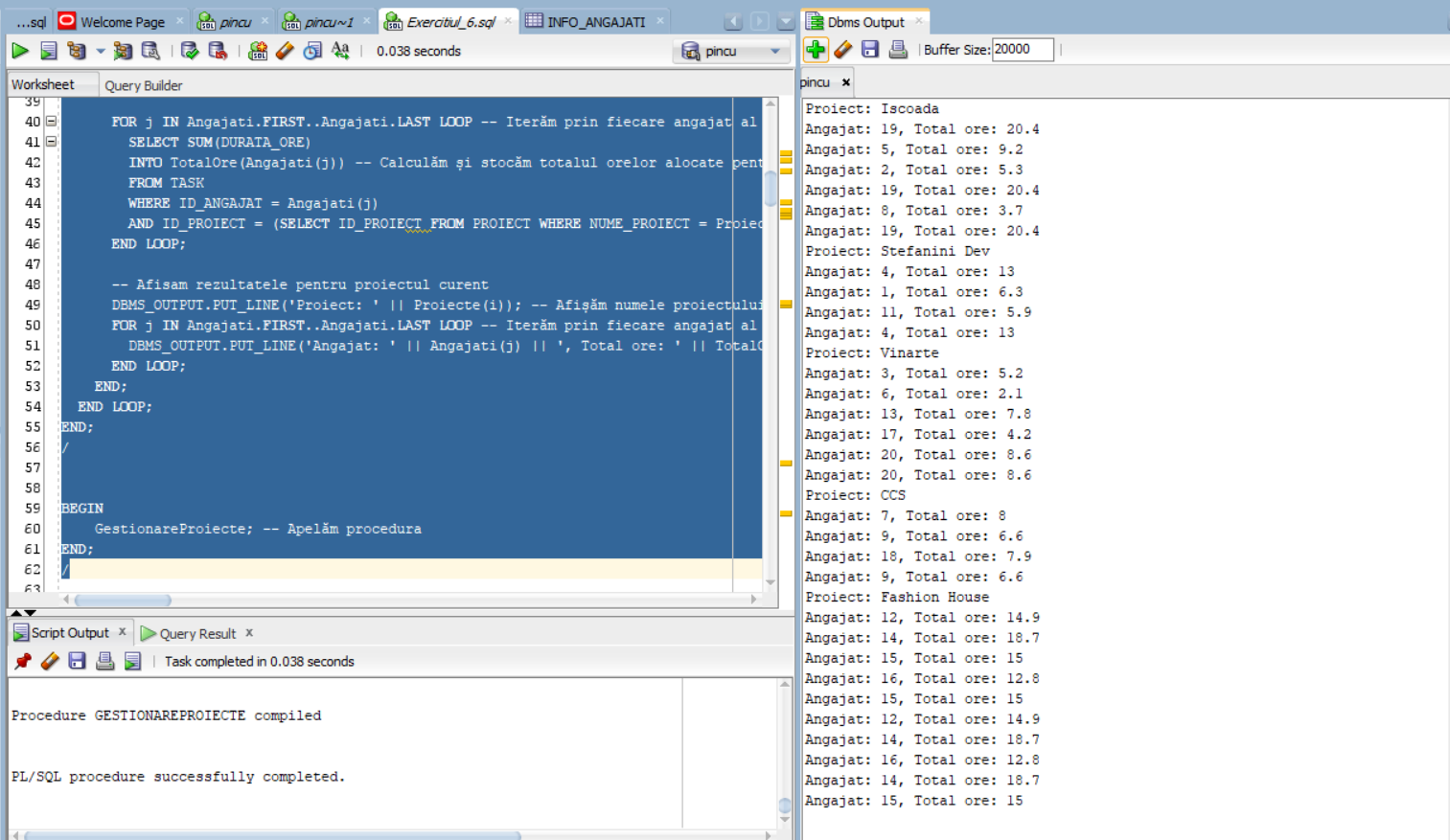
**TASK:**

****

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

Pentru toate proiectele să se afișeze id-urile anagajaților care lucrează pe fiecare proiect în parte și să se calculeze numărul total de ore pentru fiecare angajat.

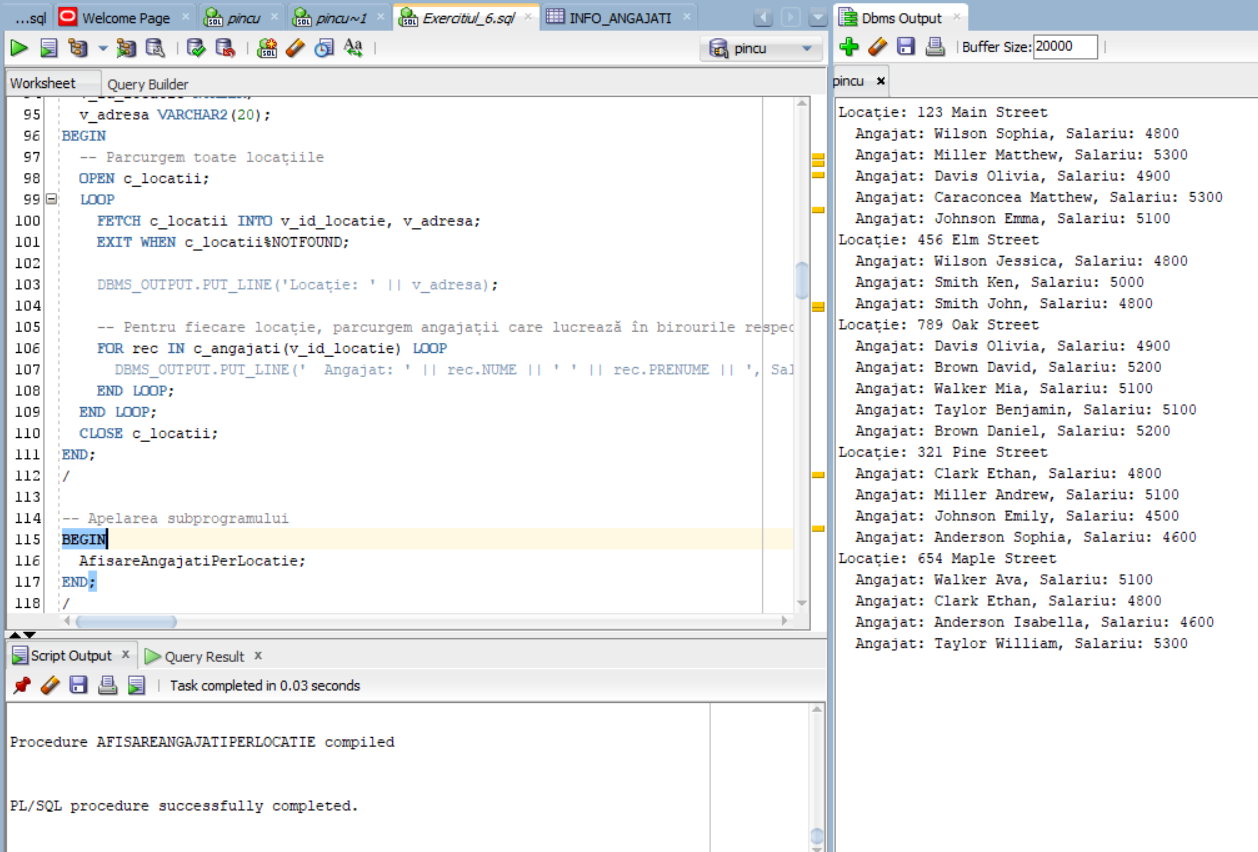
| CREATE OR REPLACE PROCEDURE GestionareProiecte AS  -- Definim tipurile de colectii  TYPE NumeProiecte IS VARRAY(10) OF VARCHAR2(15); -- VARRAY pentru stocarea numelor proiectelor  TYPE AngajatiProiect IS TABLE OF VARCHAR2(20); -- Nested Table pentru stocarea angaja?ilor  TYPE OreAlocate IS TABLE OF NUMBER INDEX BY VARCHAR2(20); -- Tabel indexat pentru stocarea orelor alocate  Proiecte NumeProiecte := NumeProiecte(); -- Initializam VARRAY-ul de proiecte  BEGIN  -- identificam toate proiectele si colectam numele lor  SELECT NUME\_PROIECT  BULK COLLECT INTO Proiecte -- extragem si stocam numele proiectelor în VARRAY  FROM PROIECT;  -- iteram prin fiecare proiect  FOR i IN 1..Proiecte.COUNT LOOP  DECLARE  Angajati AngajatiProiect := AngajatiProiect(); -- Initializam Nested Table-ul de angajati pentru proiectul curent  TotalOre OreAlocate;  BEGIN  -- identificam angajatii alocati pentru proiectul curent si calculam totalul orelor alocate pentru ei  SELECT a.id\_angajat  BULK COLLECT INTO Angajati -- Extragem si stocam numele angajatilor în Nested Table  FROM ANGAJAT a  JOIN TASK t ON a.ID\_ANGAJAT = t.ID\_ANGAJAT  WHERE t.ID\_PROIECT = (SELECT ID\_PROIECT FROM PROIECT WHERE NUME\_PROIECT = Proiecte(i));  FOR j IN Angajati.FIRST..Angajati.LAST LOOP  SELECT SUM(DURATA\_ORE)  INTO TotalOre(Angajati(j)) -- calculam si stocam totalul orelor alocate pentru fiecare angajat  FROM TASK  WHERE ID\_ANGAJAT = Angajati(j)  AND ID\_PROIECT = (SELECT ID\_PROIECT FROM PROIECT WHERE NUME\_PROIECT = Proiecte(i));  END LOOP;  -- afisam rezultatele pentru proiectul curent  DBMS\_OUTPUT.PUT\_LINE('Proiect: ' || Proiecte(i)); -- afisam numele proiectului  FOR j IN Angajati.FIRST..Angajati.LAST LOOP  DBMS\_OUTPUT.PUT\_LINE('Angajat: ' || Angajati(j) || ', Total ore: ' || TotalOre(Angajati(j))); -- afisam numele angajatului si totalul orelor alocate  END LOOP;  END;  END LOOP;  END;  /  BEGIN  GestionareProiecte; -- Apelăm procedura  END;  / |
| --- |

****

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

Să se identifice toate locațiile în care există birouri ale companiei și, pentru fiecare astfel de locație, să se listeze toți angajații care lucrează în acele birouri. Pentru fiecare angajat identificat să se afișeze numele, prenumele și salariul.

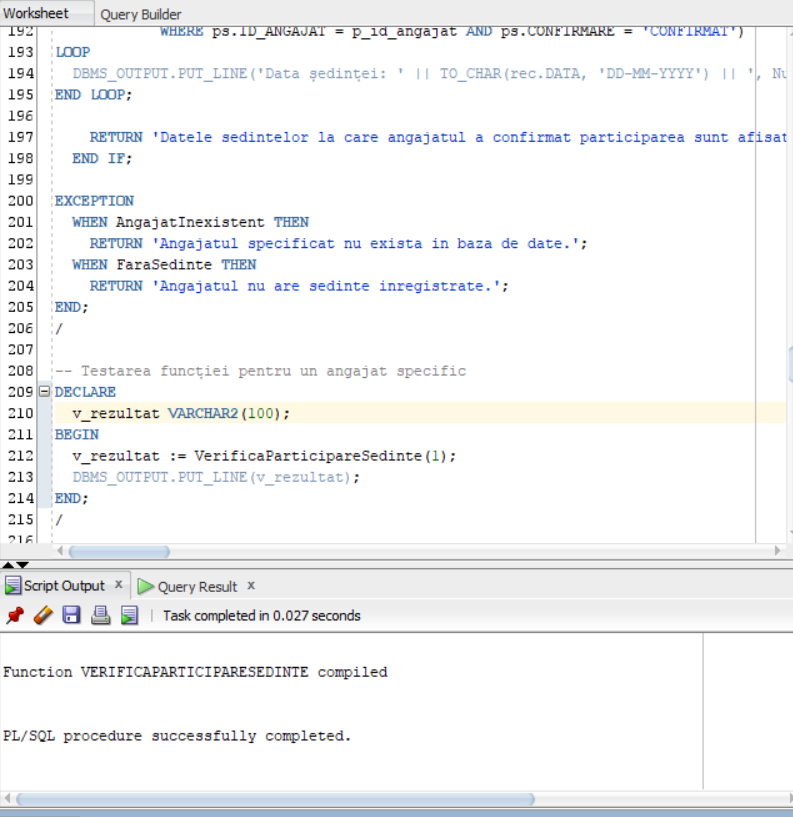
| CREATE OR REPLACE PROCEDURE AfisareAngajatiPerLocatie AS  -- cursor pentru selectarea locatiilor  CURSOR c\_locatii IS  SELECT DISTINCT l.ID\_LOCATIE, l.ADRESA  FROM LOCATIE l  JOIN BIROU b ON l.ID\_LOCATIE = b.ID\_LOCATIE;  -- cursor parametrizat pentru selectarea angajatilor dintr-un birou specific  CURSOR c\_angajati (p\_id\_locatie NUMBER) IS  SELECT a.NUME, a.PRENUME, a.SALARIU  FROM ANGAJAT a  JOIN BIROU b ON a.ID\_BIROU = b.ID\_BIROU  WHERE b.ID\_LOCATIE = p\_id\_locatie;  v\_id\_locatie NUMBER;  v\_adresa VARCHAR2(20);  BEGIN  -- parcurgem toate locatiile  OPEN c\_locatii;  LOOP  FETCH c\_locatii INTO v\_id\_locatie, v\_adresa;  EXIT WHEN c\_locatii%NOTFOUND;  DBMS\_OUTPUT.PUT\_LINE('Loca?ie: ' || v\_adresa);  -- pentru fiecare locatie, parcurgem angajatii care lucreaza în birourile respective  FOR rec IN c\_angajati(v\_id\_locatie) LOOP  DBMS\_OUTPUT.PUT\_LINE(' Angajat: ' || rec.NUME || ' ' || rec.PRENUME || ', Salariu: ' || rec.SALARIU);  END LOOP;  END LOOP;  CLOSE c\_locatii;  END;  /  -- Apelarea subprogramului  BEGIN  AfisareAngajatiPerLocatie;  END;  / |
| --- |

****

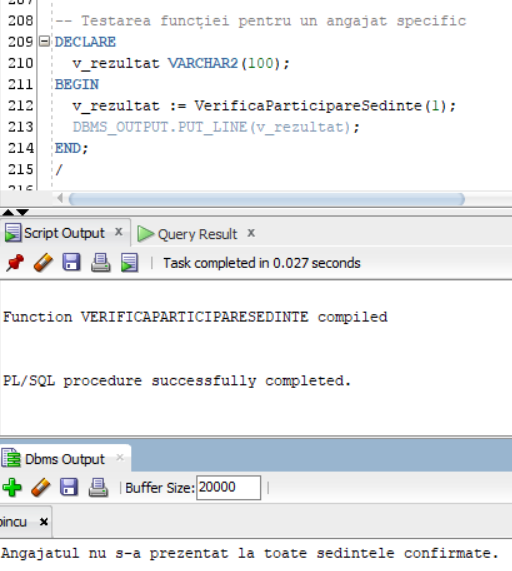
**8. Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat independent de tip funcție care să utilizeze într-o singură comandă SQL 3 dintre tabelele definite. Definiți minim 2 excepții proprii. Apelați subprogramul astfel încât să evidențiați toate cazurile definite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

Să se verifice dacă există un angajat cu un anumit ID, iar dacă există, se vor număra ședințele la care acesta este înregistrat și numărul confirmărilor. Dacă angajatatul a participat la toate ședințele la care este înregistrat, atunci se vor afișa datele acestor ședințe și numele managerului care participă.

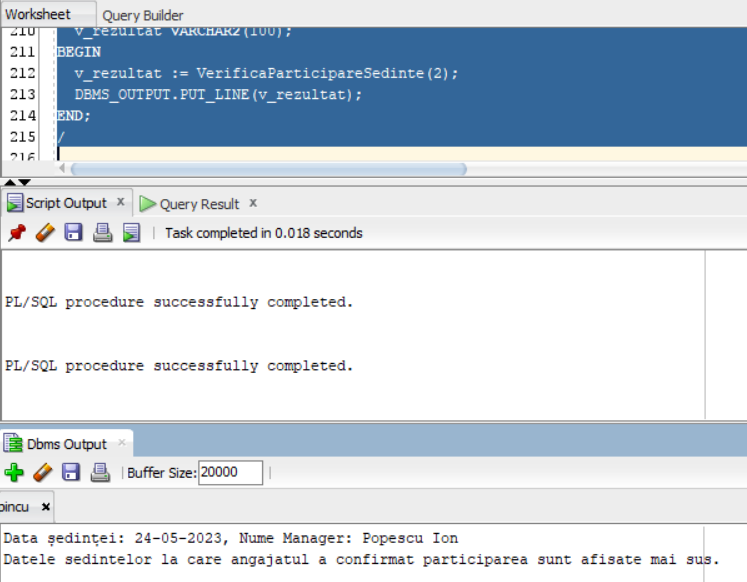
| CREATE OR REPLACE FUNCTION VerificaParticipareSedinte (p\_id\_angajat IN NUMBER)  RETURN VARCHAR2 IS  AngajatInexistent EXCEPTION;  FaraSedinte EXCEPTION;  v\_numar\_sedinte NUMBER;  v\_numar\_confirmari NUMBER;  BEGIN  -- verificam daca angajatul exista  SELECT COUNT(\*)  INTO v\_numar\_sedinte  FROM ANGAJAT  WHERE ID\_ANGAJAT = p\_id\_angajat;  IF v\_numar\_sedinte = 0 THEN  RAISE AngajatInexistent;  END IF;  -- verificam numarul de sedinte confirmate  SELECT COUNT(\*)  INTO v\_numar\_confirmari  FROM PARTICIPARE\_SEDINTA  WHERE ID\_ANGAJAT = p\_id\_angajat AND CONFIRMARE = 'CONFIRMAT';  -- verificam dacă exista sedinte inregistrate pentru angajat  SELECT COUNT(\*)  INTO v\_numar\_sedinte  FROM PARTICIPARE\_SEDINTA  WHERE ID\_ANGAJAT = p\_id\_angajat;  IF v\_numar\_sedinte = 0 THEN  RAISE FaraSedinte;  ELSIF v\_numar\_confirmari < v\_numar\_sedinte THEN  RETURN 'Angajatul nu s-a prezentat la toate sedintele confirmate.';  ELSE  -- afisam data sedintelor confirmate  FOR rec IN (SELECT s.DATA, m.NUME || ' ' || m.PRENUME AS NUME\_MANAGER  FROM PARTICIPARE\_SEDINTA ps  JOIN SEDINTA s ON ps.ID\_SEDINTA = s.ID\_SEDINTA  JOIN MANAGER m ON s.ID\_MANAGER = m.ID\_MANAGER  WHERE ps.ID\_ANGAJAT = p\_id\_angajat AND ps.CONFIRMARE = 'CONFIRMAT')  LOOP  DBMS\_OUTPUT.PUT\_LINE('Data ședinței: ' || TO\_CHAR(rec.DATA, 'DD-MM-YYYY') || ', Nume Manager: ' || rec.NUME\_MANAGER);  END LOOP;  RETURN 'Datele sedintelor la care angajatul a confirmat participarea sunt afisate mai sus.';  END IF;  EXCEPTION  WHEN AngajatInexistent THEN  RETURN 'Angajatul specificat nu exista in baza de date.';  WHEN FaraSedinte THEN  RETURN 'Angajatul nu are sedinte inregistrate.';  END;  /  -- Testarea funcției pentru un angajat specific  DECLARE  v\_rezultat VARCHAR2(100);  BEGIN  v\_rezultat := VerificaParticipareSedinte(123);  DBMS\_OUTPUT.PUT\_LINE(v\_rezultat);  END;  / |
| --- |



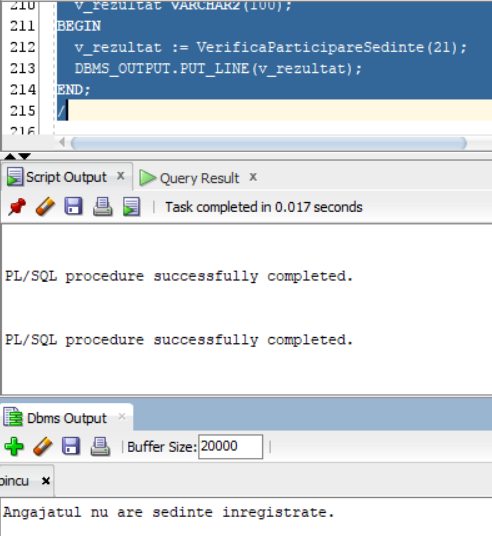
**Pentru Angajatul cu ID-ul = 1**



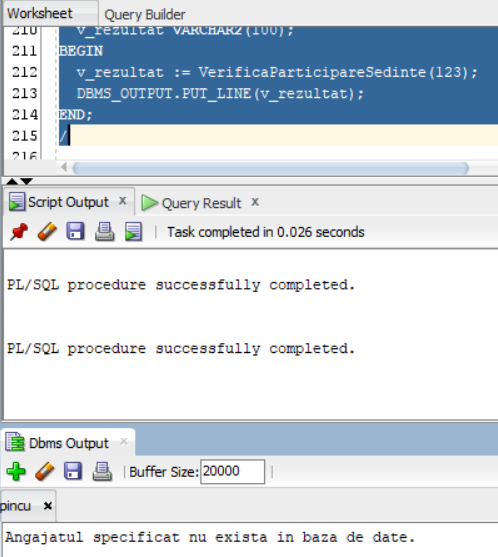
**Pentru Angajatul cu ID-ul = 2**

****

**Pentru Angajatul cu ID-ul = 21**

****

**Pentru Angajatul cu ID-ul = 123**

****

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

Să se afișeze toate locațiile unde lucrează angajații implicați într-un anumit proiect.

**Colectarea proiectelor:**

* Prima parte a procedurii constă în colectarea proiectelor cu numele specificat din tabela "PROIECT" într-un tabel indexat numit "proiecte".
* Dacă nu sunt găsite proiecte cu numele specificat, se generează excepția "NO\_DATA\_FOUND\_PROIECT".

**Verificare număr de proiecte:**

* Se verifică dacă există mai mult de un proiect cu același nume în tabela "PROIECT".
* Dacă există mai multe proiecte, se generează excepția "TOO\_MANY\_ROWS\_PROIECT".

**Colectarea locațiilor pentru proiect:**

* Procedura colectează locațiile asociate cu proiectul specificat din tabelul "LOCATIE" pe baza unei interogări complexe care implică tabelele "PROIECT", "TASK", "ANGAJAT", "BIROU" și "LOCATIE".
* Dacă nu sunt găsite locații pentru proiect, se generează excepția implicită "NO\_DATA\_FOUND".

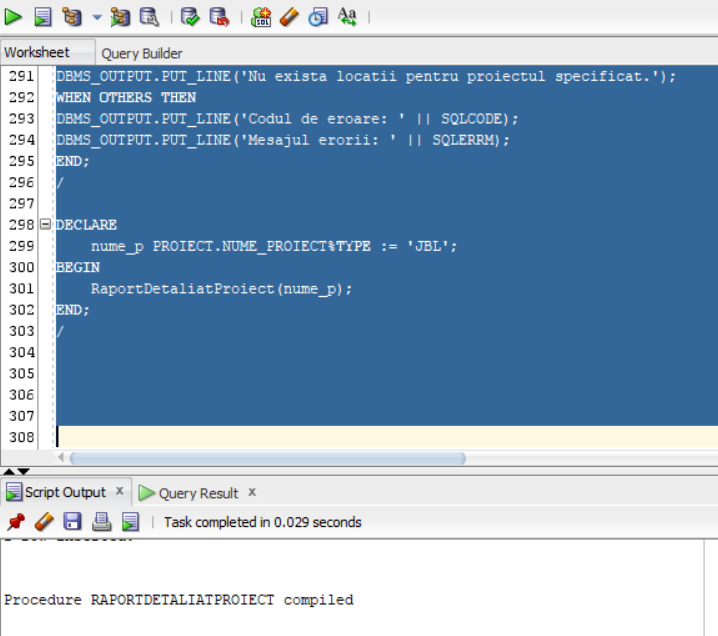
**Afișarea locațiilor:**

* Daca nu este întâlnit niciun caz de mai sus, procedura afișează locațiile găsite pentru proiectul specificat.

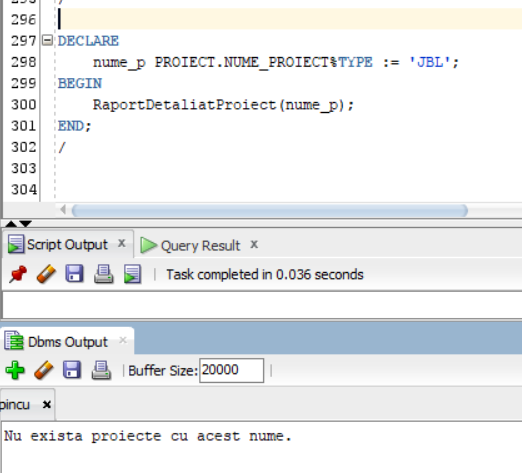
Pentru a scoate în evidență fiecare dintre cazuri, este necesar să mai inserăm date in tabelul PROIECT:

| INSERT INTO PROIECT (ID\_PROIECT, NUME\_PROIECT, ID\_MANAGER, ID\_PERIOADA, CUI\_FIRMA)  VALUES (6, 'Jaguar', 3, 221, 123456789);  INSERT INTO PROIECT (ID\_PROIECT, NUME\_PROIECT, ID\_MANAGER, ID\_PERIOADA, CUI\_FIRMA)  VALUES (7, 'Vinarte', 4, 224, 123456789); |
| --- |

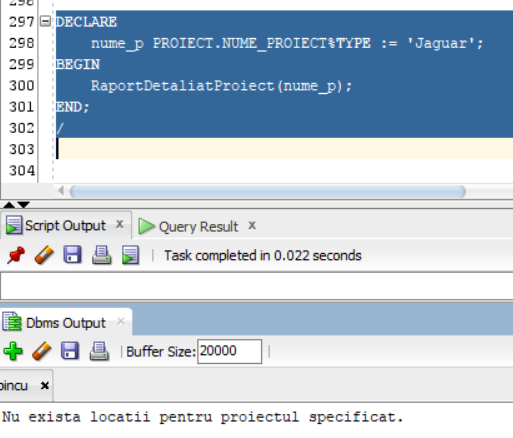
| CREATE OR REPLACE PROCEDURE RaportDetaliatProiect(nume\_p PROIECT.NUME\_PROIECT%TYPE) AS  TYPE TabelIndexLocatii IS TABLE OF LOCATIE.ADRESA%TYPE INDEX BY PLS\_INTEGER;  locatii TabelIndexLocatii;  TYPE TabelIndexProiecte IS TABLE OF PROIECT%ROWTYPE INDEX BY PLS\_INTEGER;  proiecte TabelIndexProiecte;  NO\_DATA\_FOUND\_PROIECT EXCEPTION;  TOO\_MANY\_ROWS\_PROIECT EXCEPTION;  BEGIN  -- Colectarea proiectelor  SELECT \*  BULK COLLECT INTO proiecte  FROM PROIECT  WHERE UPPER(NUME\_PROIECT) = UPPER(nume\_p);  DBMS\_OUTPUT.PUT\_LINE('Numar proiecte si nume: '||proiecte.count || nume\_p);  IF SQL%NOTFOUND THEN  RAISE NO\_DATA\_FOUND\_PROIECT;  END IF;  IF proiecte.count >= 2 THEN  RAISE TOO\_MANY\_ROWS\_PROIECT;  END IF;  -- Colectarea locațiilor pentru proiect  SELECT DISTINCT l.ADRESA  BULK COLLECT INTO locatii  FROM PROIECT p  JOIN TASK t ON p.ID\_PROIECT = t.ID\_PROIECT  JOIN ANGAJAT a ON t.ID\_ANGAJAT = a.ID\_ANGAJAT  JOIN BIROU b ON a.ID\_BIROU = b.ID\_BIROU  JOIN LOCATIE l ON b.ID\_LOCATIE = l.ID\_LOCATIE  WHERE UPPER(p.NUME\_PROIECT) = UPPER(nume\_p);  IF locatii.COUNT = 0 THEN  RAISE NO\_DATA\_FOUND;  END IF;  -- Afișarea locațiilor  DBMS\_OUTPUT.PUT\_LINE('Locatiile pentru proiectul ' || nume\_p || ':');  FOR i IN 1..locatii.COUNT LOOP  DBMS\_OUTPUT.PUT\_LINE(locatii(i));  END LOOP;  EXCEPTION  WHEN NO\_DATA\_FOUND\_PROIECT THEN  DBMS\_OUTPUT.PUT\_LINE('Nu exista proiecte cu acest nume.');  WHEN TOO\_MANY\_ROWS\_PROIECT THEN  DBMS\_OUTPUT.PUT\_LINE('Exista mai mult de un proiect cu acest nume.');  WHEN NO\_DATA\_FOUND THEN  DBMS\_OUTPUT.PUT\_LINE('Nu exista locatii pentru proiectul specificat.');  WHEN OTHERS THEN  DBMS\_OUTPUT.PUT\_LINE('Codul de eroare: ' || SQLCODE);  DBMS\_OUTPUT.PUT\_LINE('Mesajul erorii: ' || SQLERRM);  END;  /  DECLARE  nume\_p PROIECT.NUME\_PROIECT%TYPE := 'JBL';  BEGIN  RaportDetaliatProiect(nume\_p);  END;  / |
| --- |

****

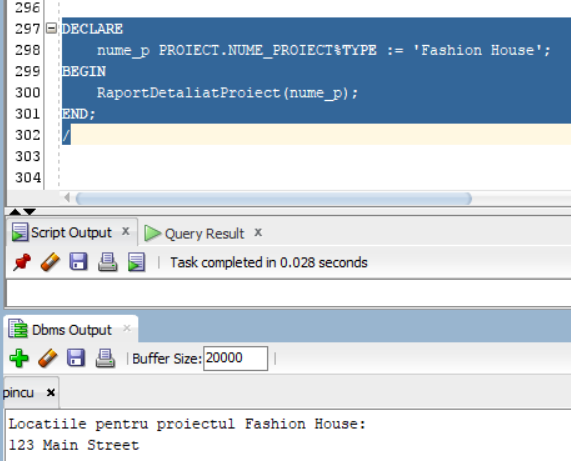
**Pentru Proiectul cu numele ‘JBL’**

****

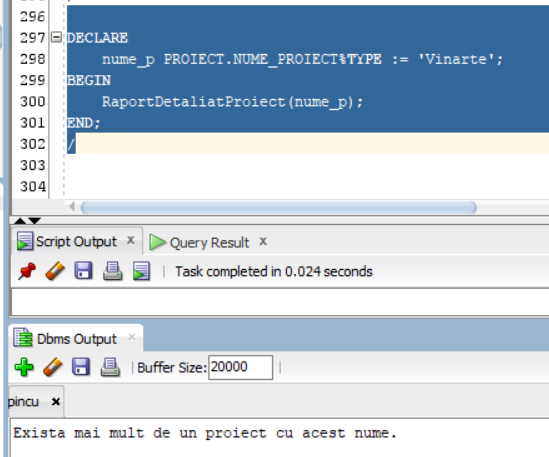
**Pentru Proiectul cu numele ‘Jaguar’**

****

**Pentru Proiectul cu numele ‘Fashion House’**

****

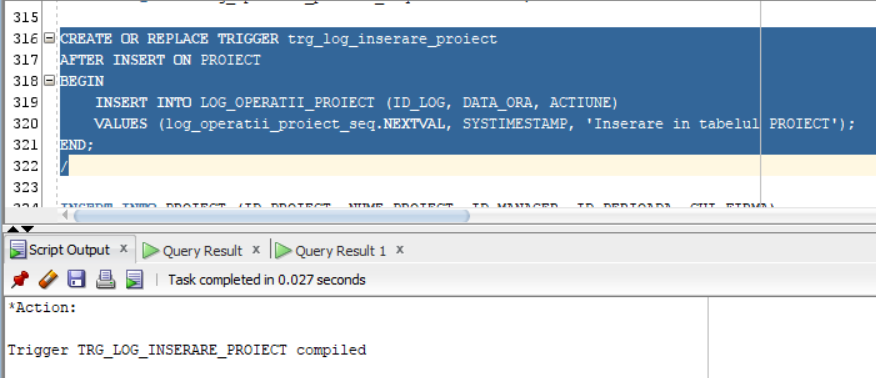
**Pentru Proiectul cu numele ‘Vinarte’’**

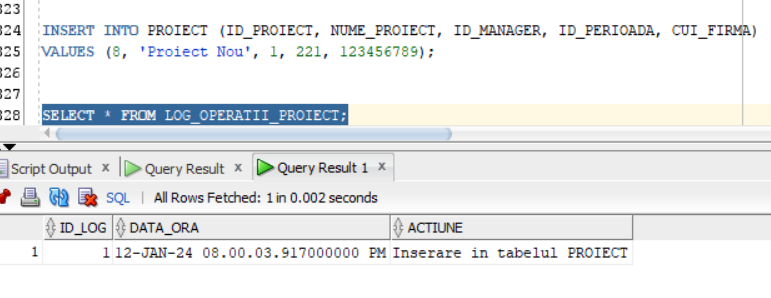
****

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

**–Acesta este un exemplu de trigger DML la nivel de comandă care înregistrează fiecare operație de inserare în tabelul PROIECT.**

| CREATE TABLE LOG\_OPERATII\_PROIECT (  ID\_LOG NUMBER(4) PRIMARY KEY,  DATA\_ORA TIMESTAMP,  ACTIUNE VARCHAR2(50)  );  CREATE SEQUENCE log\_operatii\_proiect\_seq START WITH 1;  CREATE OR REPLACE TRIGGER trg\_log\_inserare\_proiect  AFTER INSERT ON PROIECT  BEGIN  INSERT INTO LOG\_OPERATII\_PROIECT (ID\_LOG, DATA\_ORA, ACTIUNE)  VALUES (log\_operatii\_proiect\_seq.NEXTVAL, SYSTIMESTAMP, 'Inserare in tabelul PROIECT');  END;  /  INSERT INTO PROIECT (ID\_PROIECT, NUME\_PROIECT, ID\_MANAGER, ID\_PERIOADA, CUI\_FIRMA)  VALUES (8, 'Proiect Nou', 1, 221, 123456789);  SELECT \* FROM LOG\_OPERATII\_PROIECT; |
| --- |

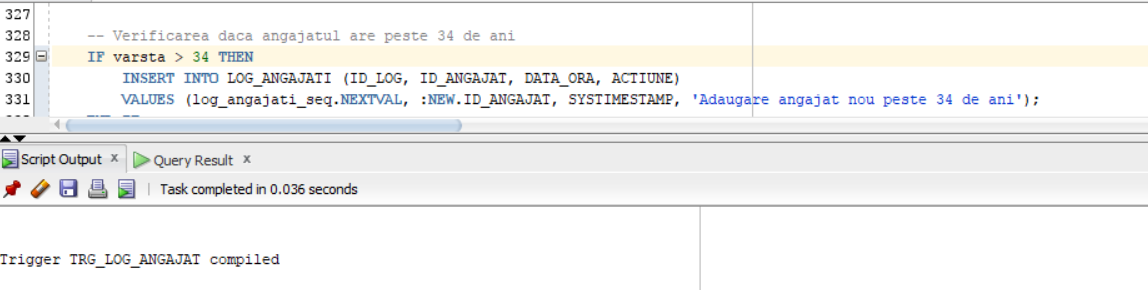
****

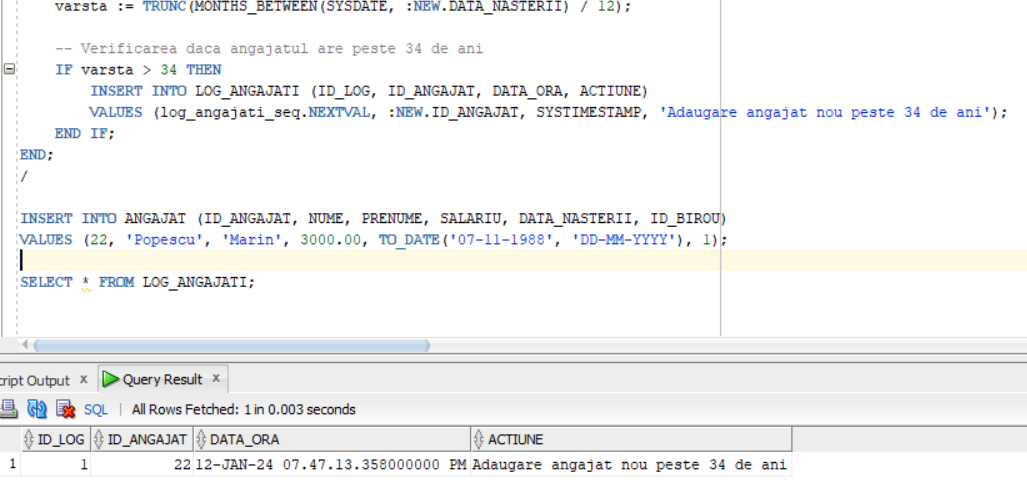
****

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

**-Adauga angajatii care au peste 34 de ani in tabelul LOG\_ANGAJATI dupa fiecare inserare**

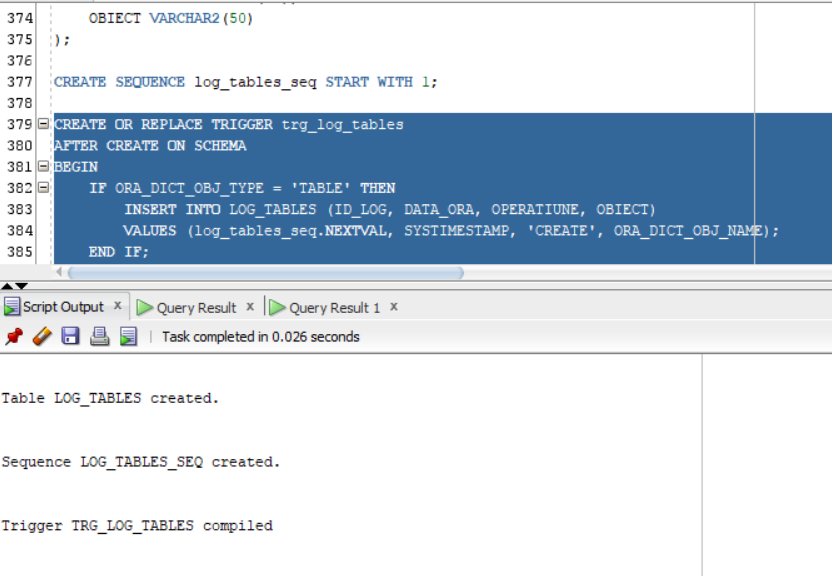
| CREATE TABLE LOG\_ANGAJATI (  ID\_LOG NUMBER(4) PRIMARY KEY,  ID\_ANGAJAT NUMBER(4),  DATA\_ORA TIMESTAMP,  ACTIUNE VARCHAR2(50)  );  -- Crearea unui sequence pentru ID-ul logurilor  CREATE SEQUENCE log\_angajati\_seq START WITH 1;  CREATE OR REPLACE TRIGGER trg\_log\_angajat  AFTER INSERT ON ANGAJAT  FOR EACH ROW  DECLARE  varsta NUMBER;  BEGIN  -- Calcularea varstei angajatului  varsta := TRUNC(MONTHS\_BETWEEN(SYSDATE, :NEW.DATA\_NASTERII) / 12);  -- Verificarea daca angajatul are peste 34 de ani  IF varsta > 34 THEN  INSERT INTO LOG\_ANGAJATI (ID\_LOG, ID\_ANGAJAT, DATA\_ORA, ACTIUNE)  VALUES (log\_angajati\_seq.NEXTVAL, :NEW.ID\_ANGAJAT, SYSTIMESTAMP, 'Adaugare angajat nou peste 34 de ani');  END IF;  END;  /  INSERT INTO ANGAJAT (ID\_ANGAJAT, NUME, PRENUME, SALARIU, DATA\_NASTERII, ID\_BIROU)  VALUES (22, 'Popescu', 'Marin', 3000.00, TO\_DATE('07-11-1988', 'DD-MM-YYYY'), 1);  SELECT \* FROM LOG\_ANGAJATI; |
| --- |

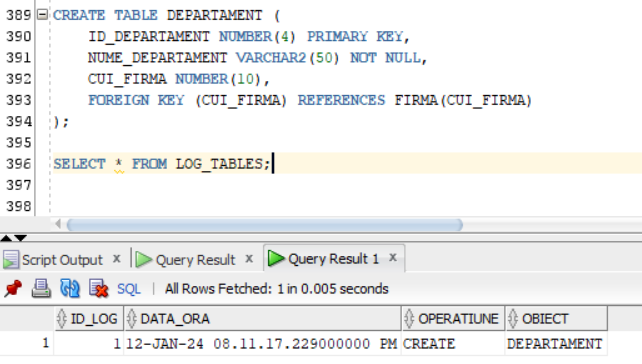
****

****

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

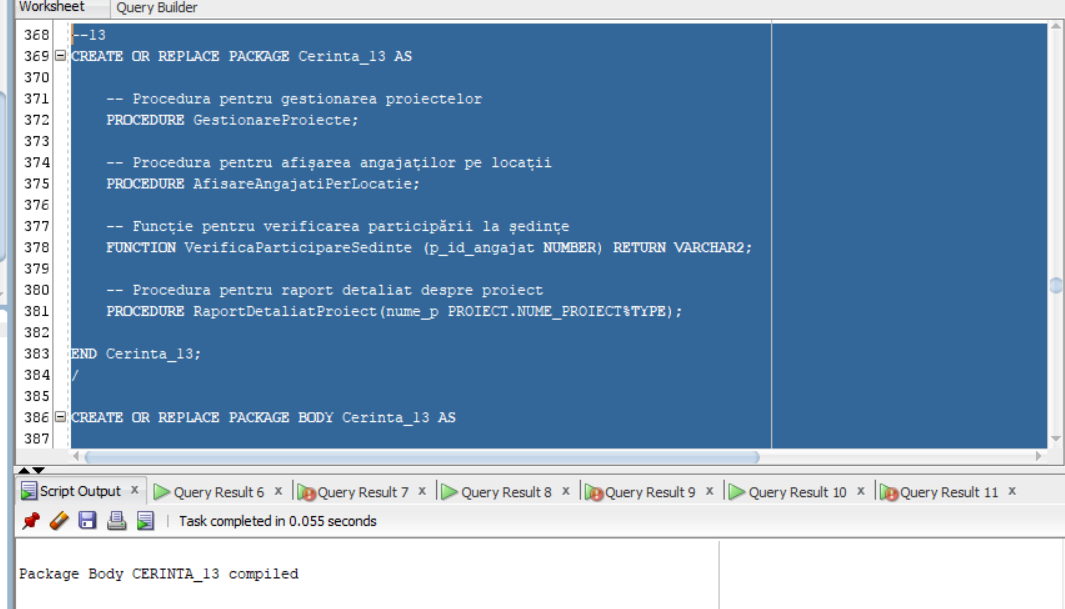
| CREATE TABLE LOG\_TABLES (  ID\_LOG NUMBER(4) PRIMARY KEY,  DATA\_ORA TIMESTAMP,  OPERATIUNE VARCHAR2(50),  OBIECT VARCHAR2(50)  );  CREATE SEQUENCE log\_tables\_seq START WITH 1;  CREATE OR REPLACE TRIGGER trg\_log\_tables  AFTER CREATE ON SCHEMA  BEGIN  IF ORA\_DICT\_OBJ\_TYPE = 'TABLE' THEN  INSERT INTO LOG\_TABLES (ID\_LOG, DATA\_ORA, OPERATIUNE, OBIECT)  VALUES (log\_tables\_seq.NEXTVAL, SYSTIMESTAMP, 'CREATE', ORA\_DICT\_OBJ\_NAME);  END IF;  END;  /  CREATE TABLE DEPARTAMENT (  ID\_DEPARTAMENT NUMBER(4) PRIMARY KEY,  NUME\_DEPARTAMENT VARCHAR2(50) NOT NULL,  CUI\_FIRMA NUMBER(10),  FOREIGN KEY (CUI\_FIRMA) REFERENCES FIRMA(CUI\_FIRMA)  );  SELECT \* FROM LOG\_TABLES; |
| --- |

****

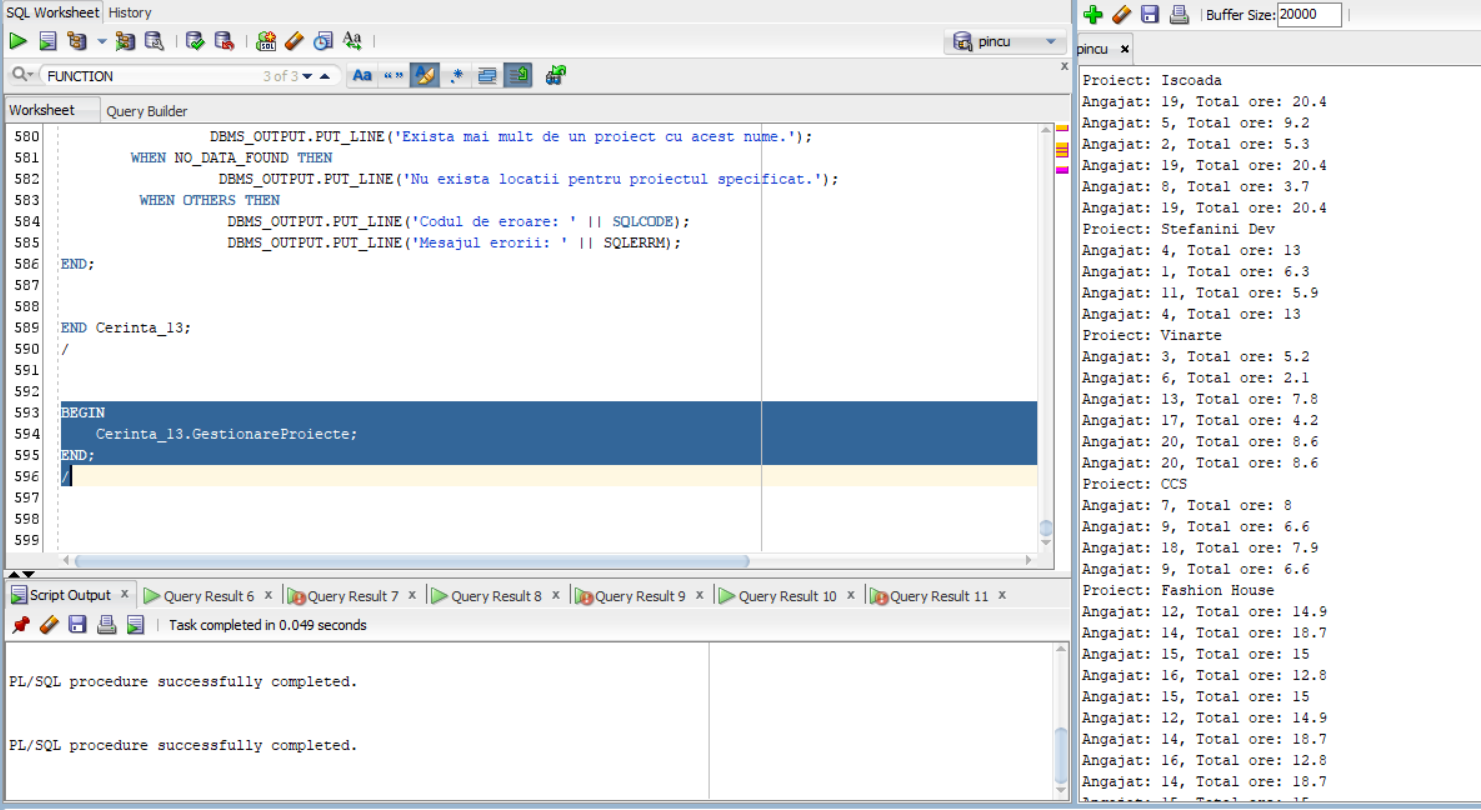
****

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

| --13  CREATE OR REPLACE PACKAGE Cerinta\_13 AS  -- Procedura pentru gestionarea proiectelor  PROCEDURE GestionareProiecte;  -- Procedura pentru afișarea angajaților pe locații  PROCEDURE AfisareAngajatiPerLocatie;  -- Funcție pentru verificarea participării la ședințe  FUNCTION VerificaParticipareSedinte (p\_id\_angajat NUMBER) RETURN VARCHAR2;  -- Procedura pentru raport detaliat despre proiect  PROCEDURE RaportDetaliatProiect(nume\_p PROIECT.NUME\_PROIECT%TYPE);  END Cerinta\_13;  /  CREATE OR REPLACE PACKAGE BODY Cerinta\_13 AS  PROCEDURE GestionareProiecte AS  -- Definim tipurile de colecții  TYPE NumeProiecte IS VARRAY(10) OF VARCHAR2(15); -- VARRAY pentru stocarea numelor proiectelor  TYPE AngajatiProiect IS TABLE OF VARCHAR2(20); -- Nested Table pentru stocarea angajaților  TYPE OreAlocate IS TABLE OF NUMBER INDEX BY VARCHAR2(20); -- Indexed table pentru stocarea orelor alocate  -- declaram variabilele colectiilor  Proiecte NumeProiecte := NumeProiecte(); -- initializam VARRAY-ul de proiecte  BEGIN  -- identificam toate proiectele si colectam numele lor  SELECT NUME\_PROIECT  BULK COLLECT INTO Proiecte -- extragem și stocam numele proiectelor în VARRAY  FROM PROIECT;  -- iteram prin fiecare proiect  FOR i IN 1..Proiecte.COUNT LOOP  DECLARE  Angajati AngajatiProiect := AngajatiProiect();  TotalOre OreAlocate;  BEGIN  -- identificam angajatii alocati pentru proiectul curent si calculam totalul orelor alocate pentru ei  SELECT a.id\_angajat  BULK COLLECT INTO Angajati -- extragem și stocam numele angajatilor  FROM ANGAJAT a  JOIN TASK t ON a.ID\_ANGAJAT = t.ID\_ANGAJAT  WHERE t.ID\_PROIECT = (SELECT ID\_PROIECT FROM PROIECT WHERE NUME\_PROIECT = Proiecte(i));  FOR j IN Angajati.FIRST..Angajati.LAST LOOP -- iteram prin fiecare angajat al proiectului  SELECT SUM(DURATA\_ORE)  INTO TotalOre(Angajati(j)) -- calculam si stocam totalul orelor alocate pentru fiecare angajat  FROM TASK  WHERE ID\_ANGAJAT = Angajati(j)  AND ID\_PROIECT = (SELECT ID\_PROIECT FROM PROIECT WHERE NUME\_PROIECT = Proiecte(i));  END LOOP;  -- afisam rezultatele pentru proiectul curent  DBMS\_OUTPUT.PUT\_LINE('Proiect: ' || Proiecte(i)); -- afisam numele proiectului  FOR j IN Angajati.FIRST..Angajati.LAST LOOP -- iteram prin fiecare angajat al proiectului  DBMS\_OUTPUT.PUT\_LINE('Angajat: ' || Angajati(j) || ', Total ore: ' || TotalOre(Angajati(j))); -- afisam numele angajatului si totalul orelor alocate  END LOOP;  END;  END LOOP;  END;  --7  PROCEDURE AfisareAngajatiPerLocatie AS  -- cursor pentru selectarea locatiilor  CURSOR c\_locatii IS  SELECT DISTINCT l.ID\_LOCATIE, l.ADRESA  FROM LOCATIE l  JOIN BIROU b ON l.ID\_LOCATIE = b.ID\_LOCATIE;  -- cursor parametrizat pentru selectarea angajatilor dintr-un birou specific  CURSOR c\_angajati (p\_id\_locatie NUMBER) IS  SELECT a.NUME, a.PRENUME, a.SALARIU  FROM ANGAJAT a  JOIN BIROU b ON a.ID\_BIROU = b.ID\_BIROU  WHERE b.ID\_LOCATIE = p\_id\_locatie;  v\_id\_locatie NUMBER;  v\_adresa VARCHAR2(20);  BEGIN  -- parcurgem toate locatiile  OPEN c\_locatii;  LOOP  FETCH c\_locatii INTO v\_id\_locatie, v\_adresa;  EXIT WHEN c\_locatii%NOTFOUND;  DBMS\_OUTPUT.PUT\_LINE('Locație: ' || v\_adresa);  -- pentru fiecare locatie, parcurgem angajatii care lucreaza in birourile respective  FOR rec IN c\_angajati(v\_id\_locatie) LOOP  DBMS\_OUTPUT.PUT\_LINE(' Angajat: ' || rec.NUME || ' ' || rec.PRENUME || ', Salariu: ' || rec.SALARIU);  END LOOP;  END LOOP;  CLOSE c\_locatii;  END;  --8  FUNCTION VerificaParticipareSedinte (p\_id\_angajat IN NUMBER)  RETURN VARCHAR2 IS  AngajatInexistent EXCEPTION;  FaraSedinte EXCEPTION;  v\_numar\_sedinte NUMBER;  v\_numar\_confirmari NUMBER;  BEGIN  -- verificam daca angajatul exista  SELECT COUNT(\*)  INTO v\_numar\_sedinte  FROM ANGAJAT  WHERE ID\_ANGAJAT = p\_id\_angajat;  IF v\_numar\_sedinte = 0 THEN  RAISE AngajatInexistent;  END IF;  -- verificam numarul de sedinte confirmate  SELECT COUNT(\*)  INTO v\_numar\_confirmari  FROM PARTICIPARE\_SEDINTA  WHERE ID\_ANGAJAT = p\_id\_angajat AND CONFIRMARE = 'CONFIRMAT';  -- verificam daca exista sedinte înregistrate pentru angajat  SELECT COUNT(\*)  INTO v\_numar\_sedinte  FROM PARTICIPARE\_SEDINTA  WHERE ID\_ANGAJAT = p\_id\_angajat;  IF v\_numar\_sedinte = 0 THEN  RAISE FaraSedinte;  ELSIF v\_numar\_confirmari < v\_numar\_sedinte THEN  RETURN 'Angajatul nu s-a prezentat la toate sedintele confirmate.';  ELSE  -- afisam data sedintelor confirmate  FOR rec IN (SELECT s.DATA, m.NUME || ' ' || m.PRENUME AS NUME\_MANAGER  FROM PARTICIPARE\_SEDINTA ps  JOIN SEDINTA s ON ps.ID\_SEDINTA = s.ID\_SEDINTA  JOIN MANAGER m ON s.ID\_MANAGER = m.ID\_MANAGER  WHERE ps.ID\_ANGAJAT = p\_id\_angajat AND ps.CONFIRMARE = 'CONFIRMAT')  LOOP  DBMS\_OUTPUT.PUT\_LINE('Data ședinței: ' || TO\_CHAR(rec.DATA, 'DD-MM-YYYY') || ', Nume Manager: ' || rec.NUME\_MANAGER);  END LOOP;  RETURN 'Datele sedintelor la care angajatul a confirmat participarea sunt afisate mai sus.';  END IF;  EXCEPTION  WHEN AngajatInexistent THEN  RETURN 'Angajatul specificat nu exista in baza de date.';  WHEN FaraSedinte THEN  RETURN 'Angajatul nu are sedinte inregistrate.';  END;  --9  PROCEDURE RaportDetaliatProiect(nume\_p PROIECT.NUME\_PROIECT%TYPE) AS  TYPE TabelIndexLocatii IS TABLE OF LOCATIE.ADRESA%TYPE INDEX BY PLS\_INTEGER;  locatii TabelIndexLocatii;  TYPE TabelIndexProiecte IS TABLE OF PROIECT%ROWTYPE INDEX BY PLS\_INTEGER;  proiecte TabelIndexProiecte;  NO\_DATA\_FOUND\_PROIECT EXCEPTION;  TOO\_MANY\_ROWS\_PROIECT EXCEPTION;  BEGIN  -- Colectarea proiectelor  SELECT \*  BULK COLLECT INTO proiecte  FROM PROIECT  WHERE UPPER(NUME\_PROIECT) = UPPER(nume\_p);  DBMS\_OUTPUT.PUT\_LINE('Numar proiecte si nume: '||proiecte.count || nume\_p);  IF SQL%NOTFOUND THEN  RAISE NO\_DATA\_FOUND\_PROIECT;  END IF;  IF proiecte.count >= 2 THEN  RAISE TOO\_MANY\_ROWS\_PROIECT;  END IF;  -- colectarea locatiilor pentru proiect  SELECT DISTINCT l.ADRESA  BULK COLLECT INTO locatii  FROM PROIECT p  JOIN TASK t ON p.ID\_PROIECT = t.ID\_PROIECT  JOIN ANGAJAT a ON t.ID\_ANGAJAT = a.ID\_ANGAJAT  JOIN BIROU b ON a.ID\_BIROU = b.ID\_BIROU  JOIN LOCATIE l ON b.ID\_LOCATIE = l.ID\_LOCATIE  WHERE UPPER(p.NUME\_PROIECT) = UPPER(nume\_p);  IF locatii.COUNT = 0 THEN  RAISE NO\_DATA\_FOUND;  END IF;  -- afisarea locatiilor  DBMS\_OUTPUT.PUT\_LINE('Locatiile pentru proiectul ' || nume\_p || ':');  FOR i IN 1..locatii.COUNT LOOP  DBMS\_OUTPUT.PUT\_LINE(locatii(i));  END LOOP;  EXCEPTION  WHEN NO\_DATA\_FOUND\_PROIECT THEN  DBMS\_OUTPUT.PUT\_LINE('Nu exista proiecte cu acest nume.');  WHEN TOO\_MANY\_ROWS\_PROIECT THEN  DBMS\_OUTPUT.PUT\_LINE('Exista mai mult de un proiect cu acest nume.');  WHEN NO\_DATA\_FOUND THEN  DBMS\_OUTPUT.PUT\_LINE('Nu exista locatii pentru proiectul specificat.');  WHEN OTHERS THEN  DBMS\_OUTPUT.PUT\_LINE('Codul de eroare: ' || SQLCODE);  DBMS\_OUTPUT.PUT\_LINE('Mesajul erorii: ' || SQLERRM);  END;  END Cerinta\_13;  / |
| --- |



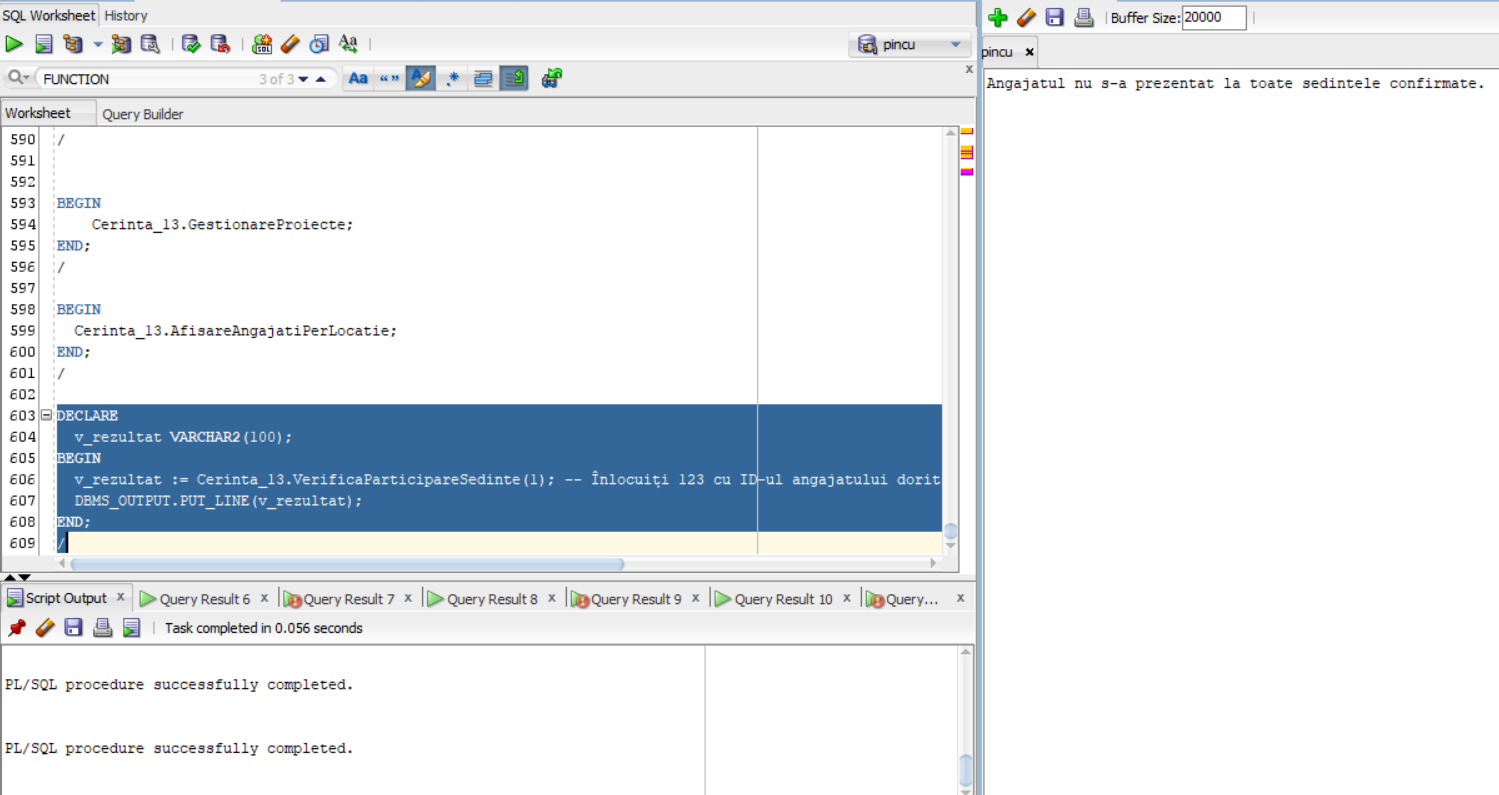
Exercitiul 6:



Exercitiul 7:



Exercitiul 8:



Exercitiul 9:

