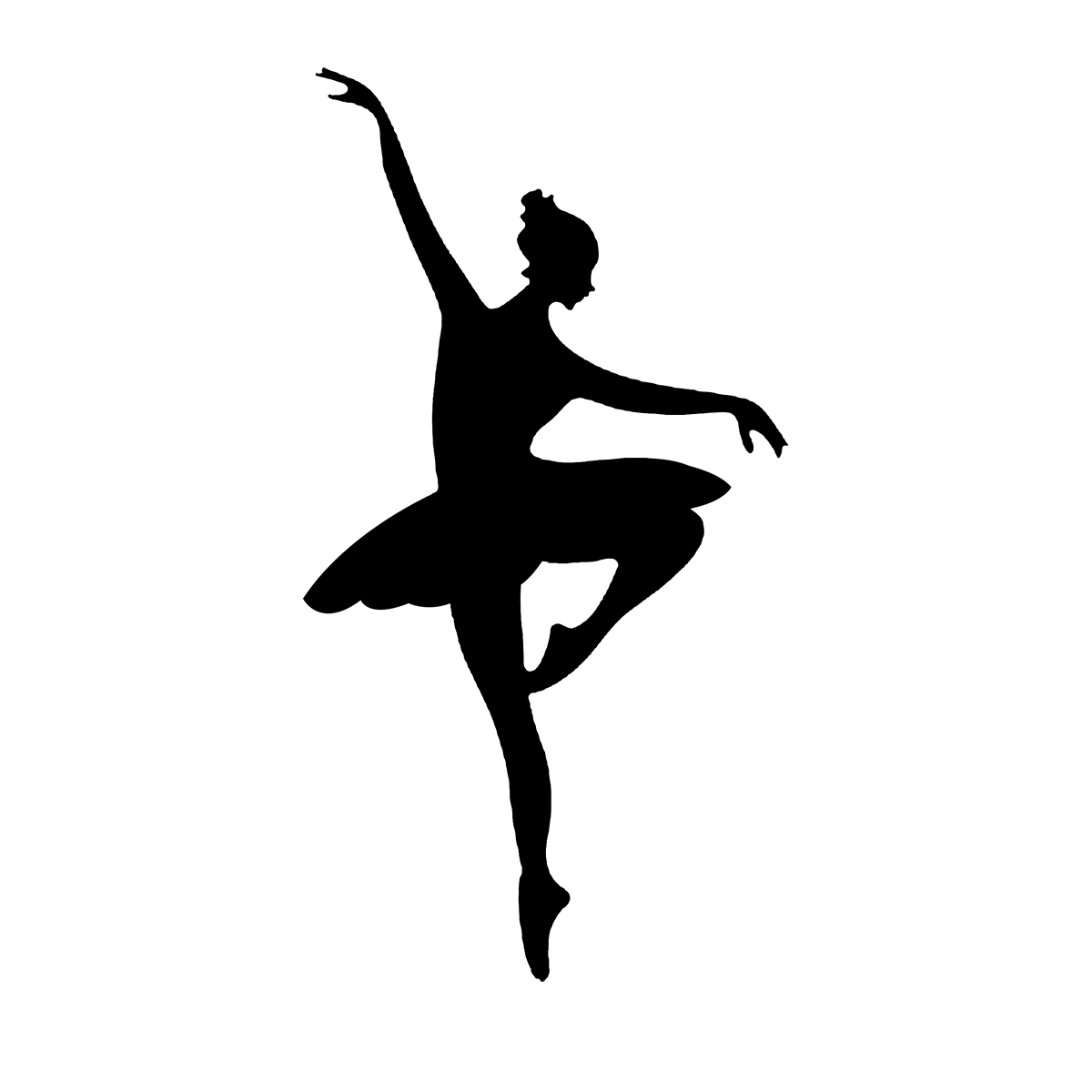
Proiect SGBD

Managementul unei Academii de Dans

Mihăilă Denisa

Seria 25 - Grupa 252

# Cuprins

# 

[**Cuprins 2**](#_wvjfimjfp3bi)

[**Introducere 2**](#_6shl0mpozf2p)

[**1. Prezentarea bazei de date 3**](#_81ruv56r51s5)

[**2. Diagrama entitate-relație (ERD) 4**](#_y2hk6kd9mytx)

[**3. Diagrama conceptuală 5**](#_vezf38f2sm9r)

[**4. Implementarea diagramei conceptuale 6**](#_uhunzje9n6tm)

[**5. Adăugarea informațiilor în tabele 12**](#_5td9qjjmd5gd)

[**6. Cerința 6 23**](#_55pl24ly01o1)

[**7. Cerința 7 26**](#_iyyve6hbfeb1)

[**8. Cerința 8 29**](#_oto5y3x409k7)

[**9. Cerința 9 31**](#_e29kkkytew52)

[**10. Cerința 10 34**](#_88w5ixfk3yeu)

[**11. Cerința 11 36**](#_fa77ug911kbc)

[**12. Cerința 12 38**](#_f2ifj2n36aj0)

[**13. Cerința 13 42**](#_zeb42wqe2g0w)

# 

# Introducere

Tema aleasă: „Managementul unei Academii de Dans”

Versiunea SGBD-ului: Oracle Database 21c Express Edition

RAM alocat: 2 GB

Configurația software: Windows 11 Pro

Nu am folosit o mașină virtuală.

# **Prezentarea bazei de date**

Prezentați pe scurt baza de date (utilitatea ei).

Modelul ales reprezintă baza de date a unei academii de dans, având rolul de a facilita gestionarea eficientă a școlilor partenere, a elevilor și instructorilor, a sălilor, locațiilor, competițiilor, abonamentelor și cursurilor.

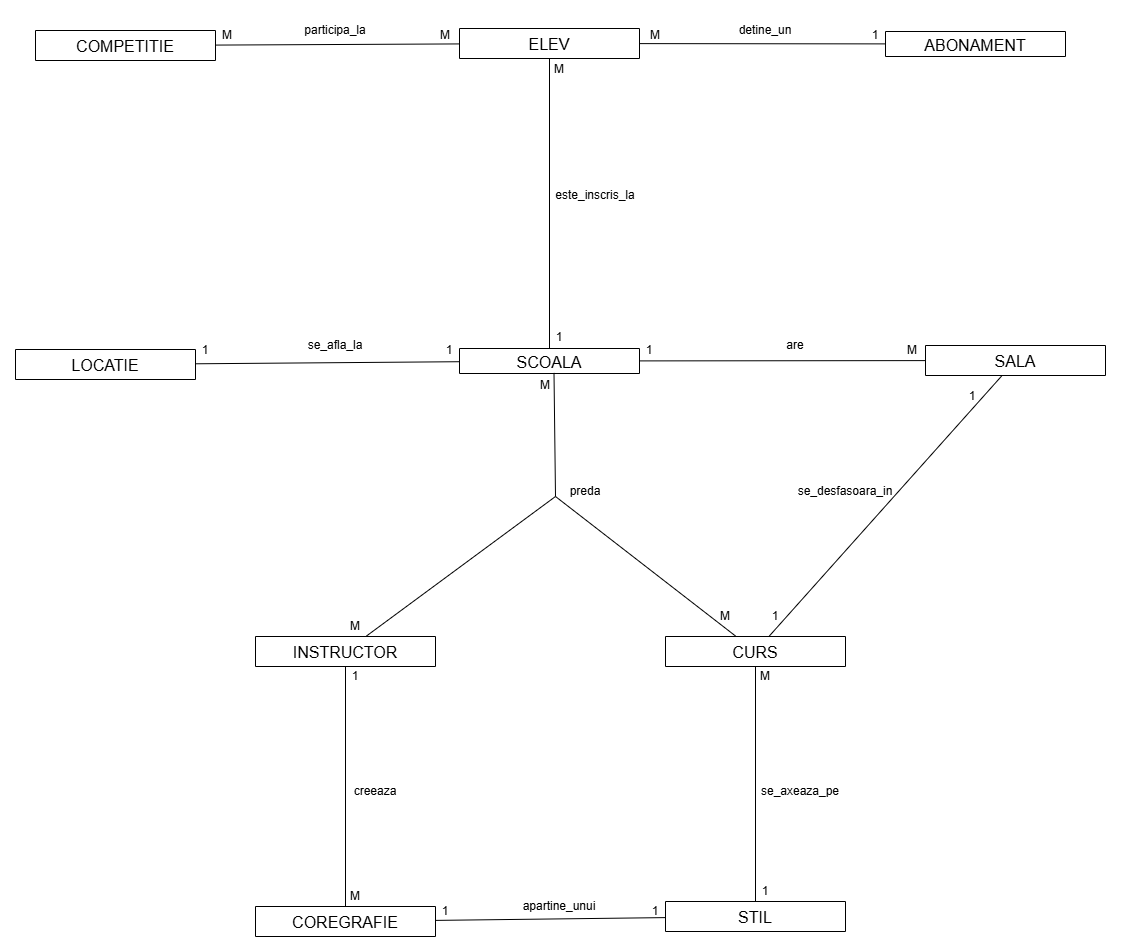
Această academie de dans administrează numeroase școli de dans din întreaga țară, fiecare aflându-se într-o locație diferită. De asemenea, fiecare școală dispune de una sau mai multe săli în care se pot desfășura cursuri.

Academia colaborează cu un număr mare de instructori care pot preda la mai multe școli diverse cursuri. Fiecare curs este asociat unui stil specific de dans și se desfășoară într-o sală anume, asigurând o organizare clară a activităților. În plus, instructorii dezvoltă propriile coregrafii originale, fiecare asociată unui stil de dans.

Elevii academiei sunt înscriși la una dintre școlile partenere, având posibilitatea de a alege dintr-o varietate de abonamente, personalizate în funcție de numărul de ședințe dorite lunar. Baza de date permite, de asemenea, gestionarea participării elevilor la competiții naționale și monitorizarea premiilor obținute.

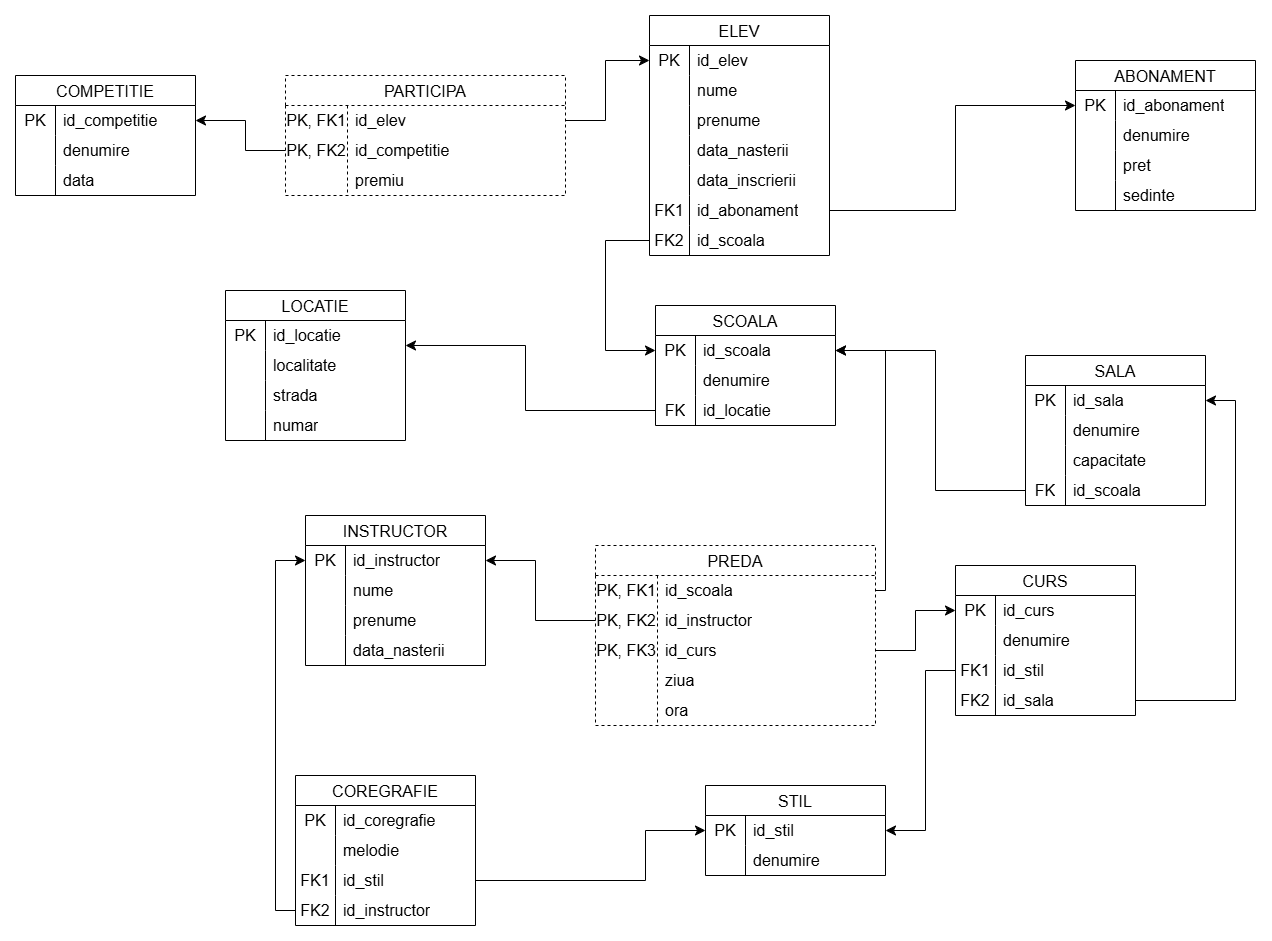
# **Diagrama entitate-relație (ERD)**

Realizați diagrama entitate-relație (ERD): entitățile, relațiile și atributele trebuie definite în limba română (vezi curs SGBD, model de diagrama entitate-relație; nu se va accepta alt format).



# **Diagrama conceptuală**

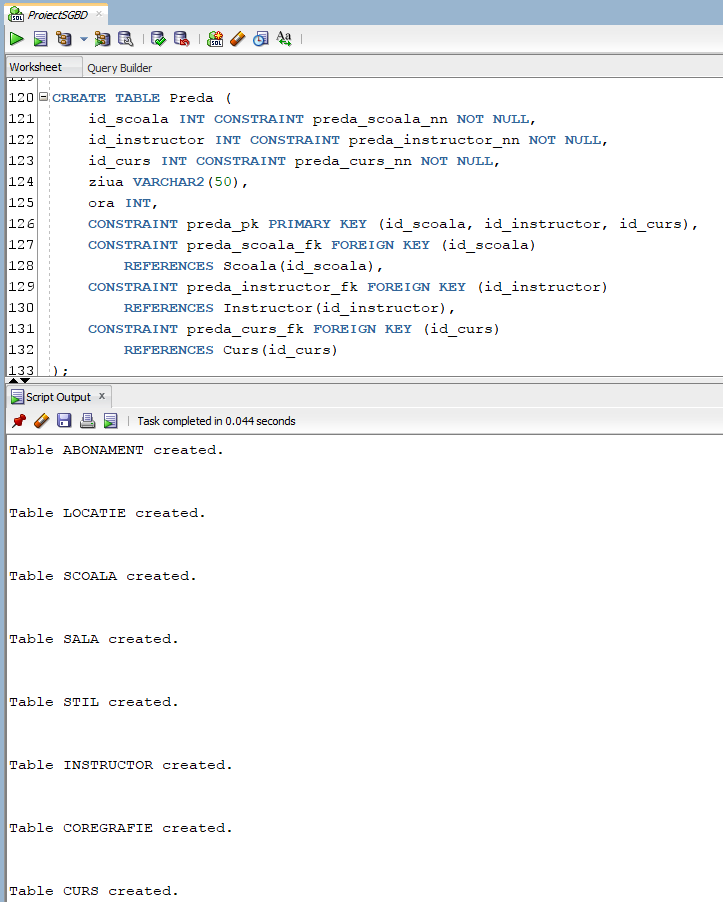
Pornind de la diagrama entitate-relație realizați diagrama conceptuală a modelului propus, integrând toate atributele necesare: entitățile, relațiile și atributele trebuie definite în limba română.

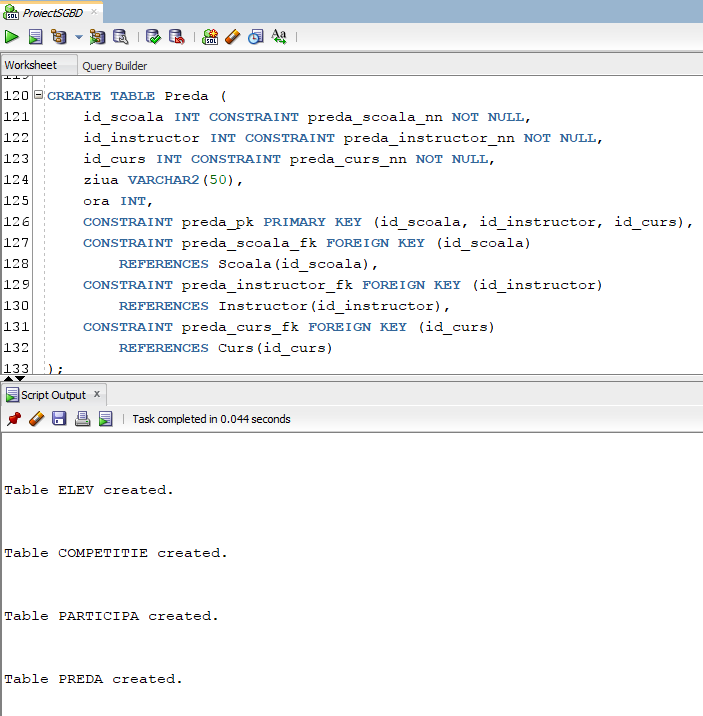


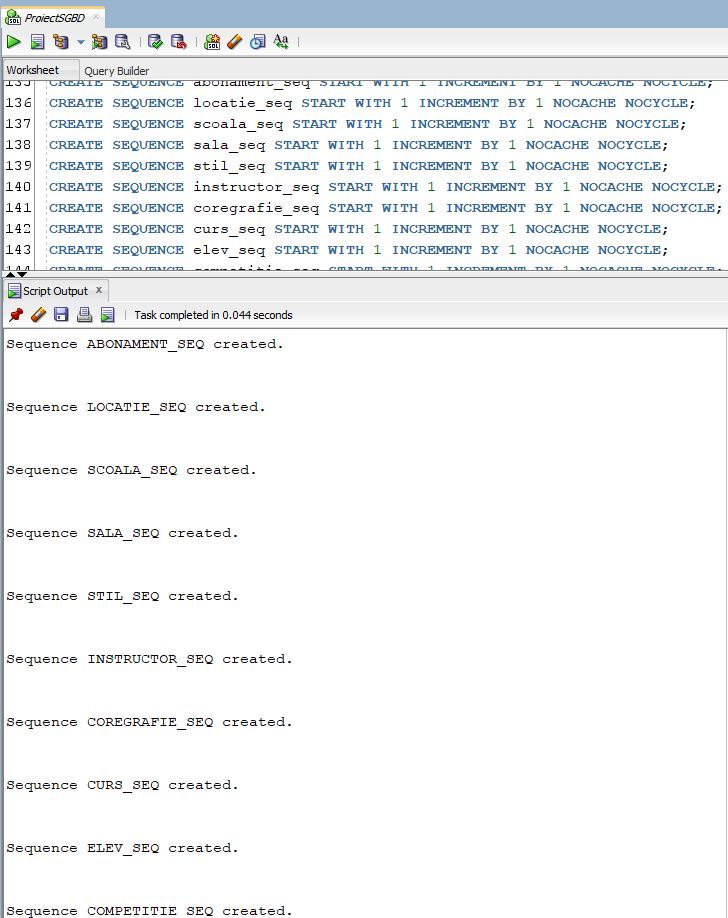
# **Implementarea diagramei conceptuale**

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

| DROP TABLE Participa CASCADE CONSTRAINTS;  DROP TABLE Preda CASCADE CONSTRAINTS;  DROP TABLE Coregrafie CASCADE CONSTRAINTS;  DROP TABLE Curs CASCADE CONSTRAINTS;  DROP TABLE Elev CASCADE CONSTRAINTS;  DROP TABLE Instructor CASCADE CONSTRAINTS;  DROP TABLE Competitie CASCADE CONSTRAINTS;  DROP TABLE Sala CASCADE CONSTRAINTS;  DROP TABLE Scoala CASCADE CONSTRAINTS;  DROP TABLE Locatie CASCADE CONSTRAINTS;  DROP TABLE Abonament CASCADE CONSTRAINTS;  DROP TABLE Stil CASCADE CONSTRAINTS;  CREATE TABLE Abonament (  id\_abonament INT,  denumire VARCHAR2(100),  pret INT,  sedinte INT,  CONSTRAINT abonament\_pk PRIMARY KEY (id\_abonament)  );  CREATE TABLE Locatie (  id\_locatie INT,  localitate VARCHAR2(100),  strada VARCHAR2(100),  numar INT,  CONSTRAINT locatie\_pk PRIMARY KEY (id\_locatie)  );  CREATE TABLE Scoala (  id\_scoala INT,  denumire VARCHAR2(100),  id\_locatie INT CONSTRAINT scoala\_locatie\_nn NOT NULL,  CONSTRAINT scoala\_pk PRIMARY KEY (id\_scoala),  CONSTRAINT scoala\_locatie\_fk FOREIGN KEY (id\_locatie)  REFERENCES Locatie(id\_locatie)  );  CREATE TABLE Sala (  id\_sala INT,  denumire VARCHAR2(100),  capacitate INT,  id\_scoala INT CONSTRAINT sala\_scoala\_nn NOT NULL,  CONSTRAINT sala\_pk PRIMARY KEY (id\_sala),  CONSTRAINT sala\_scoala\_fk FOREIGN KEY (id\_scoala)  REFERENCES Scoala(id\_scoala)  );  CREATE TABLE Stil (  id\_stil INT,  denumire VARCHAR2(100),  CONSTRAINT stil\_pk PRIMARY KEY (id\_stil)  );  CREATE TABLE Instructor (  id\_instructor INT,  nume VARCHAR2(100),  prenume VARCHAR2(100),  data\_nasterii DATE,  CONSTRAINT instructor\_pk PRIMARY KEY (id\_instructor)  );  CREATE TABLE Coregrafie (  id\_coregrafie INT,  melodie VARCHAR2(100),  id\_stil INT CONSTRAINT coregrafie\_stil\_nn NOT NULL,  id\_instructor INT CONSTRAINT coregrafie\_instructor\_nn NOT NULL,  CONSTRAINT coregrafie\_pk PRIMARY KEY (id\_coregrafie),  CONSTRAINT coregrafie\_stil\_fk FOREIGN KEY (id\_stil)  REFERENCES Stil(id\_stil),  CONSTRAINT coregrafie\_instructor\_fk FOREIGN KEY (id\_instructor)  REFERENCES Instructor(id\_instructor)  );  CREATE TABLE Curs (  id\_curs INT,  denumire VARCHAR2(100),  id\_stil INT CONSTRAINT curs\_stil\_nn NOT NULL,  id\_sala INT CONSTRAINT curs\_sala\_nn NOT NULL,  CONSTRAINT curs\_pk PRIMARY KEY (id\_curs),  CONSTRAINT curs\_stil\_fk FOREIGN KEY (id\_stil)  REFERENCES Stil(id\_stil),  CONSTRAINT curs\_sala\_fk FOREIGN KEY (id\_sala)  REFERENCES Sala(id\_sala)  );  CREATE TABLE Elev (  id\_elev INT,  nume VARCHAR2(100),  prenume VARCHAR2(100),  data\_nasterii DATE,  data\_inscrierii DATE,  id\_abonament INT CONSTRAINT elev\_abonament\_nn NOT NULL,  id\_scoala INT CONSTRAINT elev\_scoala\_nn NOT NULL,  CONSTRAINT elev\_pk PRIMARY KEY (id\_elev),  CONSTRAINT elev\_abonament\_fk FOREIGN KEY (id\_abonament)  REFERENCES Abonament(id\_abonament) ON DELETE SET NULL,  CONSTRAINT elev\_scoala\_fk FOREIGN KEY (id\_scoala)  REFERENCES Scoala(id\_scoala)  );  CREATE TABLE Competitie (  id\_competitie INT,  denumire VARCHAR2(100),  data DATE,  CONSTRAINT competitie\_pk PRIMARY KEY (id\_competitie)  );  CREATE TABLE Participa (  id\_elev INT CONSTRAINT participa\_elev\_nn NOT NULL,  id\_competitie INT CONSTRAINT participa\_competitie\_nn NOT NULL,  premiu INT,  CONSTRAINT participa\_pk PRIMARY KEY (id\_elev, id\_competitie),  CONSTRAINT participa\_elev\_fk FOREIGN KEY (id\_elev)  REFERENCES Elev(id\_elev),  CONSTRAINT participa\_competitie\_fk FOREIGN KEY (id\_competitie)  REFERENCES Competitie(id\_competitie)  );  CREATE TABLE Preda (  id\_scoala INT CONSTRAINT preda\_scoala\_nn NOT NULL,  id\_instructor INT CONSTRAINT preda\_instructor\_nn NOT NULL,  id\_curs INT CONSTRAINT preda\_curs\_nn NOT NULL,  ziua VARCHAR2(50),  ora INT,  CONSTRAINT preda\_pk PRIMARY KEY (id\_scoala, id\_instructor, id\_curs),  CONSTRAINT preda\_scoala\_fk FOREIGN KEY (id\_scoala)  REFERENCES Scoala(id\_scoala),  CONSTRAINT preda\_instructor\_fk FOREIGN KEY (id\_instructor)  REFERENCES Instructor(id\_instructor),  CONSTRAINT preda\_curs\_fk FOREIGN KEY (id\_curs)  REFERENCES Curs(id\_curs)  );  CREATE SEQUENCE abonament\_seq START WITH 1 INCREMENT BY 1 NOCACHE NOCYCLE;  CREATE SEQUENCE locatie\_seq START WITH 1 INCREMENT BY 1 NOCACHE NOCYCLE;  CREATE SEQUENCE scoala\_seq START WITH 1 INCREMENT BY 1 NOCACHE NOCYCLE;  CREATE SEQUENCE sala\_seq START WITH 1 INCREMENT BY 1 NOCACHE NOCYCLE;  CREATE SEQUENCE stil\_seq START WITH 1 INCREMENT BY 1 NOCACHE NOCYCLE;  CREATE SEQUENCE instructor\_seq START WITH 1 INCREMENT BY 1 NOCACHE NOCYCLE;  CREATE SEQUENCE coregrafie\_seq START WITH 1 INCREMENT BY 1 NOCACHE NOCYCLE;  CREATE SEQUENCE curs\_seq START WITH 1 INCREMENT BY 1 NOCACHE NOCYCLE;  CREATE SEQUENCE elev\_seq START WITH 1 INCREMENT BY 1 NOCACHE NOCYCLE;  CREATE SEQUENCE competitie\_seq START WITH 1 INCREMENT BY 1 NOCACHE NOCYCLE;  COMMIT; |
| --- |



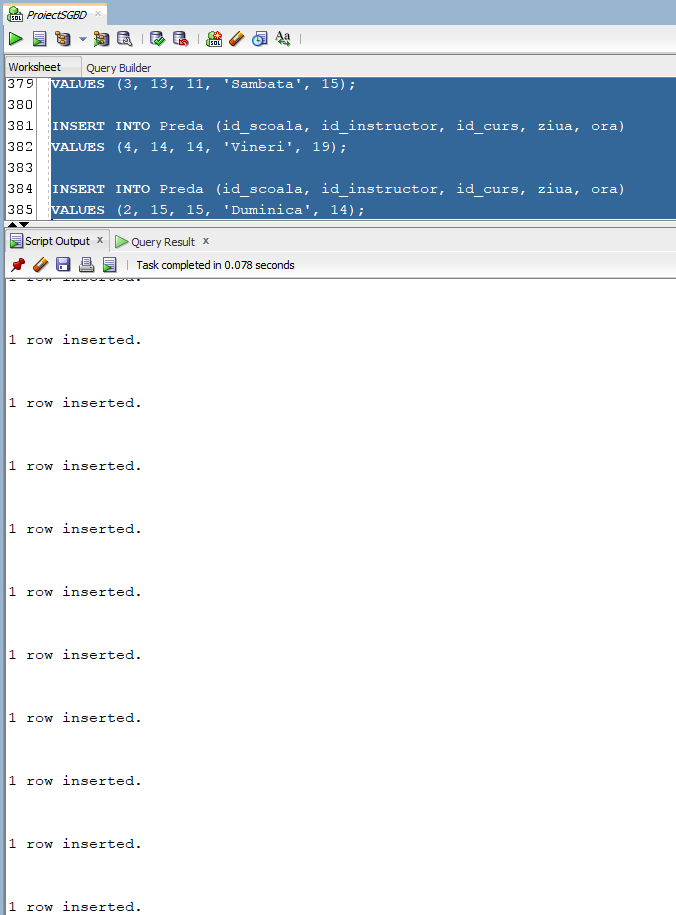


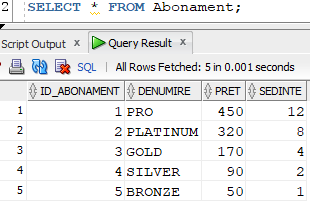
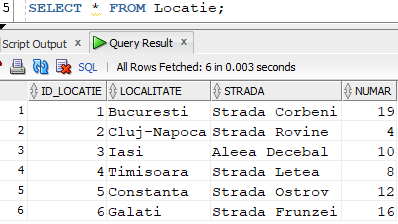


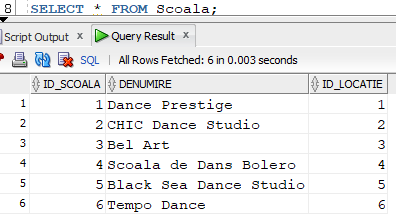
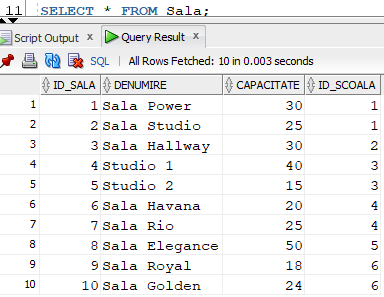
# **Adăugarea informațiilor în tabele**

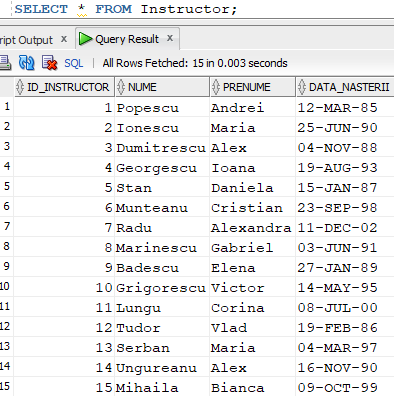
Adăugați informații coerente în tabelele create (minim 5 înregistrări pentru fiecare entitate independentă; minim 10 înregistrări pentru fiecare tabelă asociativă).

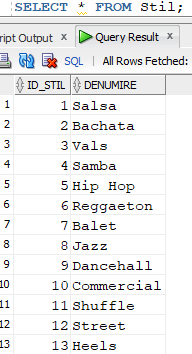
| INSERT INTO Abonament (id\_abonament, denumire, pret, sedinte)  VALUES (abonament\_seq.NEXTVAL, 'PRO', 450, 12);  INSERT INTO Abonament (id\_abonament, denumire, pret, sedinte)  VALUES (abonament\_seq.NEXTVAL, 'PLATINUM', 320, 8);  INSERT INTO Abonament (id\_abonament, denumire, pret, sedinte)  VALUES (abonament\_seq.NEXTVAL, 'GOLD', 170, 4);  INSERT INTO Abonament (id\_abonament, denumire, pret, sedinte)  VALUES (abonament\_seq.NEXTVAL, 'SILVER', 90, 2);  INSERT INTO Abonament (id\_abonament, denumire, pret, sedinte)  VALUES (abonament\_seq.NEXTVAL, 'BRONZE', 50, 1);  INSERT INTO Locatie (id\_locatie, localitate, strada, numar)  VALUES (locatie\_seq.NEXTVAL, 'Bucuresti', 'Strada Corbeni', 19);  INSERT INTO Locatie (id\_locatie, localitate, strada, numar)  VALUES (locatie\_seq.NEXTVAL, 'Cluj-Napoca', 'Strada Rovine', 4);  INSERT INTO Locatie (id\_locatie, localitate, strada, numar)  VALUES (locatie\_seq.NEXTVAL, 'Iasi', 'Aleea Decebal', 10);  INSERT INTO Locatie (id\_locatie, localitate, strada, numar)  VALUES (locatie\_seq.NEXTVAL, 'Timisoara', 'Strada Letea', 8);  INSERT INTO Locatie (id\_locatie, localitate, strada, numar)  VALUES (locatie\_seq.NEXTVAL, 'Constanta', 'Strada Ostrov', 12);  INSERT INTO Locatie (id\_locatie, localitate, strada, numar)  VALUES (locatie\_seq.NEXTVAL, 'Galati', 'Strada Frunzei', 16);  INSERT INTO Scoala (id\_scoala, denumire, id\_locatie)  VALUES (scoala\_seq.NEXTVAL, 'Dance Prestige', 1);  INSERT INTO Scoala (id\_scoala, denumire, id\_locatie)  VALUES (scoala\_seq.NEXTVAL, 'CHIC Dance Studio', 2);  INSERT INTO Scoala (id\_scoala, denumire, id\_locatie)  VALUES (scoala\_seq.NEXTVAL, 'Bel Art', 3);  INSERT INTO Scoala (id\_scoala, denumire, id\_locatie)  VALUES (scoala\_seq.NEXTVAL, 'Scoala de Dans Bolero', 4);  INSERT INTO Scoala (id\_scoala, denumire, id\_locatie)  VALUES (scoala\_seq.NEXTVAL, 'Black Sea Dance Studio', 5);  INSERT INTO Scoala (id\_scoala, denumire, id\_locatie)  VALUES (scoala\_seq.NEXTVAL, 'Tempo Dance', 6);  INSERT INTO Sala (id\_sala, denumire, capacitate, id\_scoala)  VALUES (sala\_seq.NEXTVAL, 'Sala Power', 30, 1);  INSERT INTO Sala (id\_sala, denumire, capacitate, id\_scoala)  VALUES (sala\_seq.NEXTVAL, 'Sala Studio', 25, 1);  INSERT INTO Sala (id\_sala, denumire, capacitate, id\_scoala)  VALUES (sala\_seq.NEXTVAL, 'Sala Hallway', 30, 2);  INSERT INTO Sala (id\_sala, denumire, capacitate, id\_scoala)  VALUES (sala\_seq.NEXTVAL, 'Studio 1', 40, 3);  INSERT INTO Sala (id\_sala, denumire, capacitate, id\_scoala)  VALUES (sala\_seq.NEXTVAL, 'Studio 2', 15, 3);  INSERT INTO Sala (id\_sala, denumire, capacitate, id\_scoala)  VALUES (sala\_seq.NEXTVAL, 'Sala Havana', 20, 4);  INSERT INTO Sala (id\_sala, denumire, capacitate, id\_scoala)  VALUES (sala\_seq.NEXTVAL, 'Sala Rio', 25, 4);  INSERT INTO Sala (id\_sala, denumire, capacitate, id\_scoala)  VALUES (sala\_seq.NEXTVAL, 'Sala Elegance', 50, 5);  INSERT INTO Sala (id\_sala, denumire, capacitate, id\_scoala)  VALUES (sala\_seq.NEXTVAL, 'Sala Royal', 18, 6);  INSERT INTO Sala (id\_sala, denumire, capacitate, id\_scoala)  VALUES (sala\_seq.NEXTVAL, 'Sala Golden', 24, 6);  INSERT INTO Stil (id\_stil, denumire)  VALUES (stil\_seq.NEXTVAL, 'Salsa');  INSERT INTO Stil (id\_stil, denumire)  VALUES (stil\_seq.NEXTVAL, 'Bachata');  INSERT INTO Stil (id\_stil, denumire)  VALUES (stil\_seq.NEXTVAL, 'Vals');  INSERT INTO Stil (id\_stil, denumire)  VALUES (stil\_seq.NEXTVAL, 'Samba');  INSERT INTO Stil (id\_stil, denumire)  VALUES (stil\_seq.NEXTVAL, 'Hip Hop');  INSERT INTO Stil (id\_stil, denumire)  VALUES (stil\_seq.NEXTVAL, 'Reggaeton');  INSERT INTO Stil (id\_stil, denumire)  VALUES (stil\_seq.NEXTVAL, 'Balet');  INSERT INTO Stil (id\_stil, denumire)  VALUES (stil\_seq.NEXTVAL, 'Jazz');  INSERT INTO Stil (id\_stil, denumire)  VALUES (stil\_seq.NEXTVAL, 'Dancehall');  INSERT INTO Stil (id\_stil, denumire)  VALUES (stil\_seq.NEXTVAL, 'Commercial');  INSERT INTO Stil (id\_stil, denumire)  VALUES (stil\_seq.NEXTVAL, 'Shuffle');  INSERT INTO Stil (id\_stil, denumire)  VALUES (stil\_seq.NEXTVAL, 'Street');  INSERT INTO Stil (id\_stil, denumire)  VALUES (stil\_seq.NEXTVAL, 'Heels');  INSERT INTO Instructor (id\_instructor, nume, prenume, data\_nasterii)  VALUES (instructor\_seq.NEXTVAL, 'Popescu', 'Andrei', TO\_DATE('12-03-1985', 'DD-MM-YYYY'));  INSERT INTO Instructor (id\_instructor, nume, prenume, data\_nasterii)  VALUES (instructor\_seq.NEXTVAL, 'Ionescu', 'Maria', TO\_DATE('25-06-1990', 'DD-MM-YYYY'));  INSERT INTO Instructor (id\_instructor, nume, prenume, data\_nasterii)  VALUES (instructor\_seq.NEXTVAL, 'Dumitrescu', 'Alex', TO\_DATE('04-11-1988', 'DD-MM-YYYY'));  INSERT INTO Instructor (id\_instructor, nume, prenume, data\_nasterii)  VALUES (instructor\_seq.NEXTVAL, 'Georgescu', 'Ioana', TO\_DATE('19-08-1993', 'DD-MM-YYYY'));  INSERT INTO Instructor (id\_instructor, nume, prenume, data\_nasterii)  VALUES (instructor\_seq.NEXTVAL, 'Stan', 'Daniela', TO\_DATE('15-01-1987', 'DD-MM-YYYY'));  INSERT INTO Instructor (id\_instructor, nume, prenume, data\_nasterii)  VALUES (instructor\_seq.NEXTVAL, 'Munteanu', 'Cristian', TO\_DATE('23-09-1998', 'DD-MM-YYYY'));  INSERT INTO Instructor (id\_instructor, nume, prenume, data\_nasterii)  VALUES (instructor\_seq.NEXTVAL, 'Radu', 'Alexandra', TO\_DATE('11-12-2002', 'DD-MM-YYYY'));  INSERT INTO Instructor (id\_instructor, nume, prenume, data\_nasterii)  VALUES (instructor\_seq.NEXTVAL, 'Marinescu', 'Gabriel', TO\_DATE('03-06-1991', 'DD-MM-YYYY'));  INSERT INTO Instructor (id\_instructor, nume, prenume, data\_nasterii)  VALUES (instructor\_seq.NEXTVAL, 'Badescu', 'Elena', TO\_DATE('27-01-1989', 'DD-MM-YYYY'));  INSERT INTO Instructor (id\_instructor, nume, prenume, data\_nasterii)  VALUES (instructor\_seq.NEXTVAL, 'Grigorescu', 'Victor', TO\_DATE('14-05-1995', 'DD-MM-YYYY'));  INSERT INTO Instructor (id\_instructor, nume, prenume, data\_nasterii)  VALUES (instructor\_seq.NEXTVAL, 'Lungu', 'Corina', TO\_DATE('08-07-2000', 'DD-MM-YYYY'));  INSERT INTO Instructor (id\_instructor, nume, prenume, data\_nasterii)  VALUES (instructor\_seq.NEXTVAL, 'Tudor', 'Vlad', TO\_DATE('19-02-1986', 'DD-MM-YYYY'));  INSERT INTO Instructor (id\_instructor, nume, prenume, data\_nasterii)  VALUES (instructor\_seq.NEXTVAL, 'Serban', 'Maria', TO\_DATE('04-03-1997', 'DD-MM-YYYY'));  INSERT INTO Instructor (id\_instructor, nume, prenume, data\_nasterii)  VALUES (instructor\_seq.NEXTVAL, 'Ungureanu', 'Alex', TO\_DATE('16-11-1990', 'DD-MM-YYYY'));  INSERT INTO Instructor (id\_instructor, nume, prenume, data\_nasterii)  VALUES (instructor\_seq.NEXTVAL, 'Mihaila', 'Bianca', TO\_DATE('09-10-1999', 'DD-MM-YYYY'));  INSERT INTO Coregrafie (id\_coregrafie, melodie, id\_stil, id\_instructor)  VALUES (coregrafie\_seq.NEXTVAL, 'Havana', 1, 1);  INSERT INTO Coregrafie (id\_coregrafie, melodie, id\_stil, id\_instructor)  VALUES (coregrafie\_seq.NEXTVAL, 'Bailando', 2, 2);  INSERT INTO Coregrafie (id\_coregrafie, melodie, id\_stil, id\_instructor)  VALUES (coregrafie\_seq.NEXTVAL, 'Perfect', 3, 3);  INSERT INTO Coregrafie (id\_coregrafie, melodie, id\_stil, id\_instructor)  VALUES (coregrafie\_seq.NEXTVAL, 'Magalenha', 4, 4);  INSERT INTO Coregrafie (id\_coregrafie, melodie, id\_stil, id\_instructor)  VALUES (coregrafie\_seq.NEXTVAL, 'In Da Club', 5, 5);  INSERT INTO Coregrafie (id\_coregrafie, melodie, id\_stil, id\_instructor)  VALUES (coregrafie\_seq.NEXTVAL, 'Despacito', 6, 6);  INSERT INTO Coregrafie (id\_coregrafie, melodie, id\_stil, id\_instructor)  VALUES (coregrafie\_seq.NEXTVAL, 'Swan Lake Theme', 7, 7);  INSERT INTO Coregrafie (id\_coregrafie, melodie, id\_stil, id\_instructor)  VALUES (coregrafie\_seq.NEXTVAL, 'Feeling Good', 8, 8);  INSERT INTO Coregrafie (id\_coregrafie, melodie, id\_stil, id\_instructor)  VALUES (coregrafie\_seq.NEXTVAL, 'Temperature', 9, 9);  INSERT INTO Coregrafie (id\_coregrafie, melodie, id\_stil, id\_instructor)  VALUES (coregrafie\_seq.NEXTVAL, 'Shape of You', 10, 10);  INSERT INTO Coregrafie (id\_coregrafie, melodie, id\_stil, id\_instructor)  VALUES (coregrafie\_seq.NEXTVAL, 'Faded', 11, 11);  INSERT INTO Coregrafie (id\_coregrafie, melodie, id\_stil, id\_instructor)  VALUES (coregrafie\_seq.NEXTVAL, 'Uptown Funk', 12, 12);  INSERT INTO Coregrafie (id\_coregrafie, melodie, id\_stil, id\_instructor)  VALUES (coregrafie\_seq.NEXTVAL, 'Single Ladies', 13, 13);  INSERT INTO Coregrafie (id\_coregrafie, melodie, id\_stil, id\_instructor)  VALUES (coregrafie\_seq.NEXTVAL, 'We Don’t Talk About Bruno', 2, 14);  INSERT INTO Coregrafie (id\_coregrafie, melodie, id\_stil, id\_instructor)  VALUES (coregrafie\_seq.NEXTVAL, 'Lose Yourself', 5, 15);  INSERT INTO Curs (id\_curs, denumire, id\_stil, id\_sala)  VALUES (curs\_seq.NEXTVAL, 'Salsa Tropical', 1, 1);  INSERT INTO Curs (id\_curs, denumire, id\_stil, id\_sala)  VALUES (curs\_seq.NEXTVAL, 'Bachata Sensual', 2, 2);  INSERT INTO Curs (id\_curs, denumire, id\_stil, id\_sala)  VALUES (curs\_seq.NEXTVAL, 'Vals Vienez', 3, 3);  INSERT INTO Curs (id\_curs, denumire, id\_stil, id\_sala)  VALUES (curs\_seq.NEXTVAL, 'Samba Carnaval', 4, 4);  INSERT INTO Curs (id\_curs, denumire, id\_stil, id\_sala)  VALUES (curs\_seq.NEXTVAL, 'Hip-Hop Coregrafie', 5, 5);  INSERT INTO Curs (id\_curs, denumire, id\_stil, id\_sala)  VALUES (curs\_seq.NEXTVAL, 'Reggaeton Beat', 6, 2);  INSERT INTO Curs (id\_curs, denumire, id\_stil, id\_sala)  VALUES (curs\_seq.NEXTVAL, 'Balet pentru Începători', 7, 3);  INSERT INTO Curs (id\_curs, denumire, id\_stil, id\_sala)  VALUES (curs\_seq.NEXTVAL, 'Jazz Modern', 8, 4);  INSERT INTO Curs (id\_curs, denumire, id\_stil, id\_sala)  VALUES (curs\_seq.NEXTVAL, 'Dancehall Freestyle', 9, 1);  INSERT INTO Curs (id\_curs, denumire, id\_stil, id\_sala)  VALUES (curs\_seq.NEXTVAL, 'Commercial Moves', 10, 2);  INSERT INTO Curs (id\_curs, denumire, id\_stil, id\_sala)  VALUES (curs\_seq.NEXTVAL, 'Shuffle Steps', 11, 3);  INSERT INTO Curs (id\_curs, denumire, id\_stil, id\_sala)  VALUES (curs\_seq.NEXTVAL, 'Street Performance', 12, 4);  INSERT INTO Curs (id\_curs, denumire, id\_stil, id\_sala)  VALUES (curs\_seq.NEXTVAL, 'Fierce on Heels', 13, 5);  INSERT INTO Curs (id\_curs, denumire, id\_stil, id\_sala)  VALUES (curs\_seq.NEXTVAL, 'Salsa Social', 1, 3);  INSERT INTO Curs (id\_curs, denumire, id\_stil, id\_sala)  VALUES (curs\_seq.NEXTVAL, 'Bachata Intensiv', 2, 4);  INSERT INTO Elev (id\_elev, nume, prenume, data\_nasterii, data\_inscrierii, id\_abonament, id\_scoala)  VALUES (elev\_seq.NEXTVAL, 'Popa', 'Ana', TO\_DATE('14-07-2005', 'DD-MM-YYYY'), TO\_DATE('10-01-2024', 'DD-MM-YYYY'), 1, 1);  INSERT INTO Elev (id\_elev, nume, prenume, data\_nasterii, data\_inscrierii, id\_abonament, id\_scoala)  VALUES (elev\_seq.NEXTVAL, 'Ionescu', 'Vlad', TO\_DATE('03-11-2002', 'DD-MM-YYYY'), TO\_DATE('12-03-2023', 'DD-MM-YYYY'), 2, 1);  INSERT INTO Elev (id\_elev, nume, prenume, data\_nasterii, data\_inscrierii, id\_abonament, id\_scoala)  VALUES (elev\_seq.NEXTVAL, 'Radu', 'Mihai', TO\_DATE('25-12-1998', 'DD-MM-YYYY'), TO\_DATE('01-02-2022', 'DD-MM-YYYY'), 3, 2);  INSERT INTO Elev (id\_elev, nume, prenume, data\_nasterii, data\_inscrierii, id\_abonament, id\_scoala)  VALUES (elev\_seq.NEXTVAL, 'Dobre', 'Maria', TO\_DATE('19-08-2000', 'DD-MM-YYYY'), TO\_DATE('02-04-2023', 'DD-MM-YYYY'), 1, 2);  INSERT INTO Elev (id\_elev, nume, prenume, data\_nasterii, data\_inscrierii, id\_abonament, id\_scoala)  VALUES (elev\_seq.NEXTVAL, 'Munteanu', 'Ioan', TO\_DATE('18-10-2000', 'DD-MM-YYYY'), TO\_DATE('05-03-2024', 'DD-MM-YYYY'), 4, 3);  INSERT INTO Elev (id\_elev, nume, prenume, data\_nasterii, data\_inscrierii, id\_abonament, id\_scoala)  VALUES (elev\_seq.NEXTVAL, 'Iliescu', 'Teodora', TO\_DATE('22-06-1999', 'DD-MM-YYYY'), TO\_DATE('28-01-2021', 'DD-MM-YYYY'), 2, 3);  INSERT INTO Elev (id\_elev, nume, prenume, data\_nasterii, data\_inscrierii, id\_abonament, id\_scoala)  VALUES (elev\_seq.NEXTVAL, 'Dima', 'Alexandra', TO\_DATE('30-03-1997', 'DD-MM-YYYY'), TO\_DATE('20-04-2023', 'DD-MM-YYYY'), 5, 4);  INSERT INTO Elev (id\_elev, nume, prenume, data\_nasterii, data\_inscrierii, id\_abonament, id\_scoala)  VALUES (elev\_seq.NEXTVAL, 'Petrescu', 'Andrei', TO\_DATE('14-05-2001', 'DD-MM-YYYY'), TO\_DATE('04-05-2022', 'DD-MM-YYYY'), 1, 5);  INSERT INTO Elev (id\_elev, nume, prenume, data\_nasterii, data\_inscrierii, id\_abonament, id\_scoala)  VALUES (elev\_seq.NEXTVAL, 'Voinea', 'Cristian', TO\_DATE('22-08-2003', 'DD-MM-YYYY'), TO\_DATE('15-05-2024', 'DD-MM-YYYY'), 5, 5);  INSERT INTO Elev (id\_elev, nume, prenume, data\_nasterii, data\_inscrierii, id\_abonament, id\_scoala)  VALUES (elev\_seq.NEXTVAL, 'Ciobanu', 'Diana', TO\_DATE('08-12-2004', 'DD-MM-YYYY'), TO\_DATE('10-06-2023', 'DD-MM-YYYY'), 4, 6);  INSERT INTO Competitie (id\_competitie, denumire, data)  VALUES (competitie\_seq.NEXTVAL, 'Campionatul National de Dans', TO\_DATE('25-03-2023', 'DD-MM-YYYY'));  INSERT INTO Competitie (id\_competitie, denumire, data)  VALUES (competitie\_seq.NEXTVAL, 'Gala Ritmurilor', TO\_DATE('10-05-2023', 'DD-MM-YYYY'));  INSERT INTO Competitie (id\_competitie, denumire, data)  VALUES (competitie\_seq.NEXTVAL, 'Festivalul Dansului', TO\_DATE('15-06-2024', 'DD-MM-YYYY'));  INSERT INTO Competitie (id\_competitie, denumire, data)  VALUES (competitie\_seq.NEXTVAL, 'Noaptea Ritmurilor', TO\_DATE('20-07-2024', 'DD-MM-YYYY'));  INSERT INTO Competitie (id\_competitie, denumire, data)  VALUES (competitie\_seq.NEXTVAL, 'Euforia Dansului', TO\_DATE('30-08-2024', 'DD-MM-YYYY'));  INSERT INTO Participa (id\_elev, id\_competitie, premiu)  VALUES (1, 1, 1);  INSERT INTO Participa (id\_elev, id\_competitie, premiu)  VALUES (2, 1, 2);  INSERT INTO Participa (id\_elev, id\_competitie, premiu)  VALUES (3, 2, 3);  INSERT INTO Participa (id\_elev, id\_competitie, premiu)  VALUES (4, 2, 1);  INSERT INTO Participa (id\_elev, id\_competitie, premiu)  VALUES (5, 3, 2);  INSERT INTO Participa (id\_elev, id\_competitie, premiu)  VALUES (6, 3, 3);  INSERT INTO Participa (id\_elev, id\_competitie, premiu)  VALUES (7, 4, 1);  INSERT INTO Participa (id\_elev, id\_competitie, premiu)  VALUES (8, 4, 2);  INSERT INTO Participa (id\_elev, id\_competitie, premiu)  VALUES (9, 5, 1);  INSERT INTO Participa (id\_elev, id\_competitie, premiu)  VALUES (10, 5, 3);  INSERT INTO Preda (id\_scoala, id\_instructor, id\_curs, ziua, ora)  VALUES (1, 1, 1, 'Luni', 18);  INSERT INTO Preda (id\_scoala, id\_instructor, id\_curs, ziua, ora)  VALUES (1, 2, 9, 'Miercuri', 19);  INSERT INTO Preda (id\_scoala, id\_instructor, id\_curs, ziua, ora)  VALUES (2, 3, 2, 'Marti', 17);  INSERT INTO Preda (id\_scoala, id\_instructor, id\_curs, ziua, ora)  VALUES (2, 4, 10, 'Vineri', 20);  INSERT INTO Preda (id\_scoala, id\_instructor, id\_curs, ziua, ora)  VALUES (3, 5, 3, 'Joi', 16);  INSERT INTO Preda (id\_scoala, id\_instructor, id\_curs, ziua, ora)  VALUES (3, 6, 7, 'Luni', 18);  INSERT INTO Preda (id\_scoala, id\_instructor, id\_curs, ziua, ora)  VALUES (4, 7, 4, 'Sambata', 14);  INSERT INTO Preda (id\_scoala, id\_instructor, id\_curs, ziua, ora)  VALUES (4, 8, 12, 'Duminica', 16);  INSERT INTO Preda (id\_scoala, id\_instructor, id\_curs, ziua, ora)  VALUES (5, 9, 5, 'Marti', 19);  INSERT INTO Preda (id\_scoala, id\_instructor, id\_curs, ziua, ora)  VALUES (5, 10, 13, 'Joi', 20);  INSERT INTO Preda (id\_scoala, id\_instructor, id\_curs, ziua, ora)  VALUES (6, 11, 6, 'Luni', 17);  INSERT INTO Preda (id\_scoala, id\_instructor, id\_curs, ziua, ora)  VALUES (6, 12, 8, 'Miercuri', 18);  INSERT INTO Preda (id\_scoala, id\_instructor, id\_curs, ziua, ora)  VALUES (3, 13, 11, 'Sambata', 15);  INSERT INTO Preda (id\_scoala, id\_instructor, id\_curs, ziua, ora)  VALUES (4, 14, 14, 'Vineri', 19);  INSERT INTO Preda (id\_scoala, id\_instructor, id\_curs, ziua, ora)  VALUES (2, 15, 15, 'Duminica', 14); |
| --- |

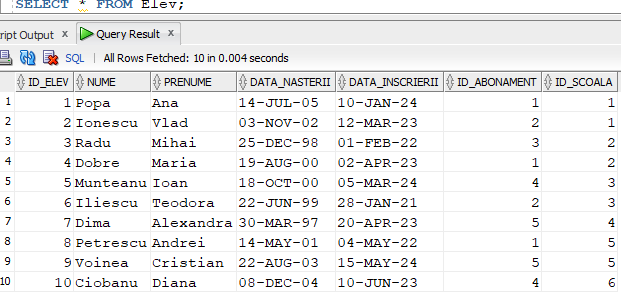


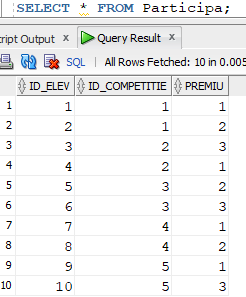
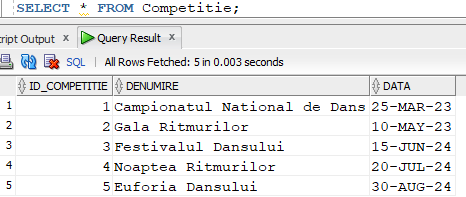


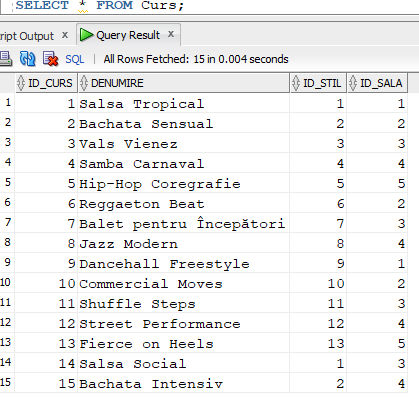
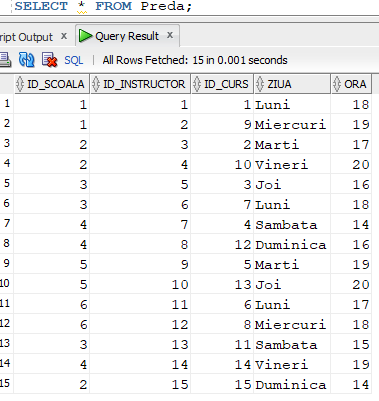


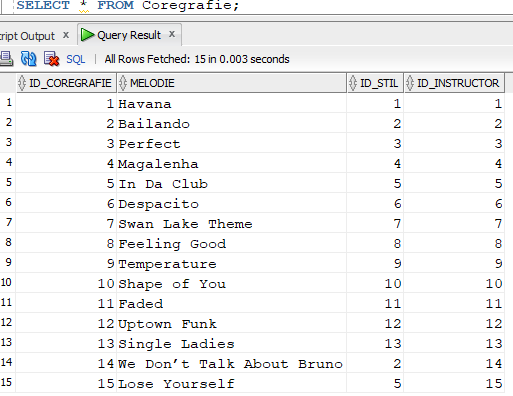










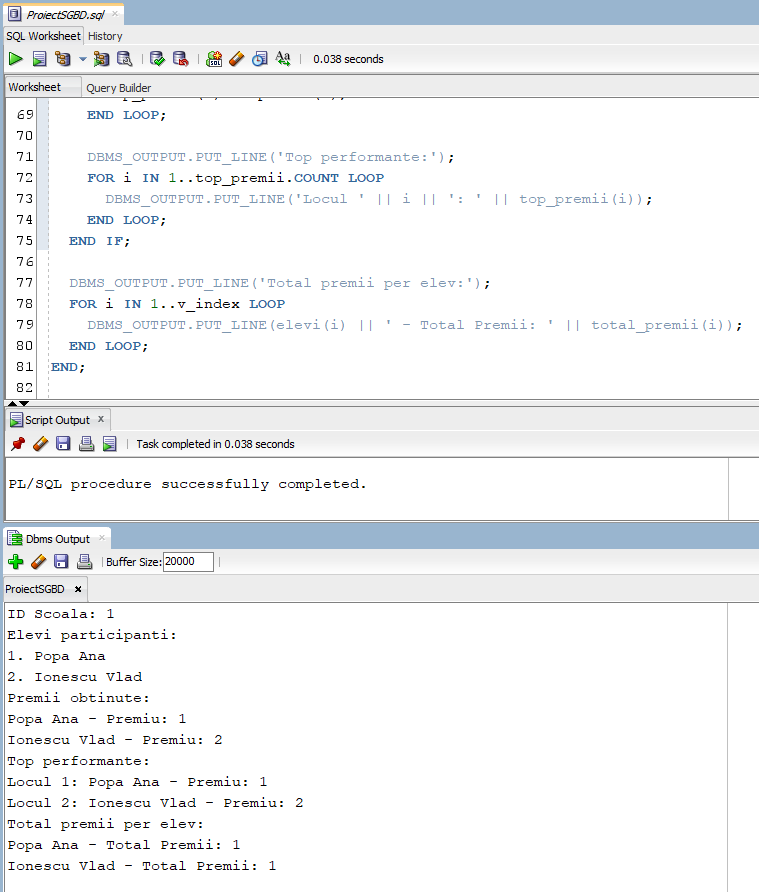


# **Cerința 6**

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.

Cerința problemei: Să se gestioneze participarea elevilor la competiții pentru o anumită școală. Folosiți un tabel indexat pentru a stoca numele elevilor participanți, un tabel imbricat pentru a reține premiile obținute la competiții (excluzând premiile NULL) și un vector pentru a păstra cele mai bune 2 performanțe ale elevilor din școala respectivă. Subprogramul va calcula, de asemenea, numărul total de premii pentru fiecare elev și va organiza aceste date, afișându-le structurat. Dacă școala nu are elevi sau elevii nu au obținut premii valide, subprogramul va trata aceste cazuri corespunzător.

| DECLARE  TYPE elevi\_table IS TABLE OF VARCHAR2(200) INDEX BY PLS\_INTEGER;  elevi elevi\_table;  TYPE premii\_table IS TABLE OF VARCHAR2(200);  premii premii\_table := premii\_table();  TYPE top\_performante IS VARRAY(2) OF VARCHAR2(200);  top\_premii top\_performante := top\_performante();  TYPE premiu\_counter IS TABLE OF PLS\_INTEGER INDEX BY PLS\_INTEGER;  total\_premii premiu\_counter;  v\_index PLS\_INTEGER := 0;  v\_scoala\_id NUMBER := 1;  BEGIN  FOR rec IN (  SELECT e.id\_elev, e.nume || ' ' || e.prenume AS nume\_complet  FROM Elev e  WHERE e.id\_scoala = v\_scoala\_id  ) LOOP  v\_index := v\_index + 1;  elevi(v\_index) := rec.nume\_complet;  total\_premii(v\_index) := 0;  END LOOP;  IF v\_index = 0 THEN  DBMS\_OUTPUT.PUT\_LINE('ID Scoala: ' || v\_scoala\_id);  DBMS\_OUTPUT.PUT\_LINE('Nu exista elevi in aceasta scoala');  RETURN;  END IF;  DBMS\_OUTPUT.PUT\_LINE('ID Scoala: ' || v\_scoala\_id);  DBMS\_OUTPUT.PUT\_LINE('Elevi participanti:');  FOR i IN 1..v\_index LOOP  DBMS\_OUTPUT.PUT\_LINE(i || '. ' || elevi(i));  END LOOP;  FOR rec IN (  SELECT e.nume || ' ' || e.prenume || ' - Premiu: ' || p.premiu AS detalii\_premiu, e.nume || ' ' || e.prenume AS nume\_complet, p.premiu  FROM Participa p  JOIN Elev e ON p.id\_elev = e.id\_elev  WHERE e.id\_scoala = v\_scoala\_id  ORDER BY p.premiu ASC  ) LOOP  IF rec.premiu IS NOT NULL THEN  premii.EXTEND;  premii(premii.COUNT) := rec.detalii\_premiu;  FOR i IN 1..v\_index LOOP  IF elevi(i) = rec.nume\_complet THEN  total\_premii(i) := total\_premii(i) + 1;  END IF;  END LOOP;  END IF;  END LOOP;  IF premii.COUNT = 0 THEN  DBMS\_OUTPUT.PUT\_LINE('Nu exista premii pentru elevii din aceasta scoala');  ELSE  DBMS\_OUTPUT.PUT\_LINE('Premii obtinute:');  FOR i IN 1..premii.COUNT LOOP  DBMS\_OUTPUT.PUT\_LINE(premii(i));  END LOOP;  FOR i IN 1..LEAST(2, premii.COUNT) LOOP  top\_premii.EXTEND;  top\_premii(i) := premii(i);  END LOOP;  DBMS\_OUTPUT.PUT\_LINE('Top performante:');  FOR i IN 1..top\_premii.COUNT LOOP  DBMS\_OUTPUT.PUT\_LINE('Locul ' || i || ': ' || top\_premii(i));  END LOOP;  END IF;  DBMS\_OUTPUT.PUT\_LINE('Total premii per elev:');  FOR i IN 1..v\_index LOOP  DBMS\_OUTPUT.PUT\_LINE(elevi(i) || ' - Total Premii: ' || total\_premii(i));  END LOOP;  END; |
| --- |

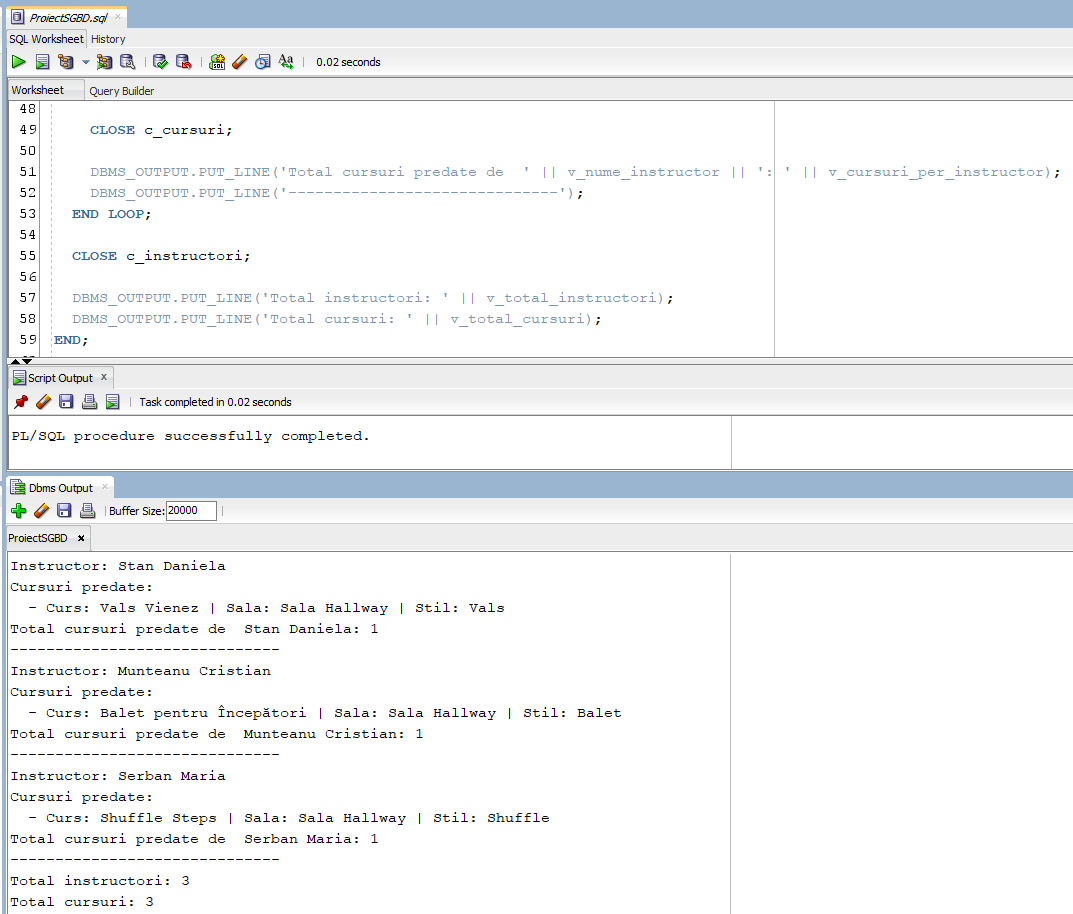


# **Cerința 7**

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.

Cerința problemei: Identificați instructorii care predau cursuri într-o anumită școală și detaliile cursurilor predate de aceștia, incluzând denumirea cursurilor, sala în care se desfășoară și stilul de dans asociat. Se vor utiliza două cursoare: unul pentru a prelua instructorii care predau la școala specificată și un alt cursor parametrizat, dependent de primul, pentru a obține detaliile cursurilor fiecărui instructor. Subprogramul va calcula numărul total de cursuri predate de fiecare instructor și va afișa, la final, numărul total de instructori și cursuri din școală.

| DECLARE  CURSOR c\_instructori IS  SELECT DISTINCT i.id\_instructor, i.nume || ' ' || i.prenume AS nume\_complet  FROM Instructor i  JOIN Preda p ON i.id\_instructor = p.id\_instructor  WHERE p.id\_scoala = 3;  CURSOR c\_cursuri (p\_id\_instructor NUMBER) IS  SELECT c.denumire, s.denumire AS sala, st.denumire AS stil  FROM Curs c  JOIN Preda p ON c.id\_curs = p.id\_curs  JOIN Sala s ON c.id\_sala = s.id\_sala  JOIN Stil st ON c.id\_stil = st.id\_stil  WHERE p.id\_instructor = p\_id\_instructor;  v\_instructor\_id NUMBER;  v\_nume\_instructor VARCHAR2(200);  v\_denumire\_curs VARCHAR2(200);  v\_sala VARCHAR2(200);  v\_stil VARCHAR2(200);  v\_total\_cursuri NUMBER := 0;  v\_total\_instructori NUMBER := 0;  v\_cursuri\_per\_instructor NUMBER := 0;  BEGIN  OPEN c\_instructori;  LOOP  FETCH c\_instructori INTO v\_instructor\_id, v\_nume\_instructor;  EXIT WHEN c\_instructori%NOTFOUND;  v\_total\_instructori := v\_total\_instructori + 1;  v\_cursuri\_per\_instructor := 0;  DBMS\_OUTPUT.PUT\_LINE('Instructor: ' || v\_nume\_instructor);  DBMS\_OUTPUT.PUT\_LINE('Cursuri predate:');  OPEN c\_cursuri(v\_instructor\_id);  LOOP  FETCH c\_cursuri INTO v\_denumire\_curs, v\_sala, v\_stil;  EXIT WHEN c\_cursuri%NOTFOUND;  v\_cursuri\_per\_instructor := v\_cursuri\_per\_instructor + 1;  v\_total\_cursuri := v\_total\_cursuri + 1;  DBMS\_OUTPUT.PUT\_LINE(' - Curs: ' || v\_denumire\_curs || ' | Sala: ' || v\_sala || ' | Stil: ' || v\_stil);  END LOOP;  CLOSE c\_cursuri;  DBMS\_OUTPUT.PUT\_LINE('Total cursuri predate de ' || v\_nume\_instructor || ': ' || v\_cursuri\_per\_instructor);  DBMS\_OUTPUT.PUT\_LINE('------------------------------');  END LOOP;  CLOSE c\_instructori;  DBMS\_OUTPUT.PUT\_LINE('Total instructori: ' || v\_total\_instructori);  DBMS\_OUTPUT.PUT\_LINE('Total cursuri: ' || v\_total\_cursuri);  END; |
| --- |

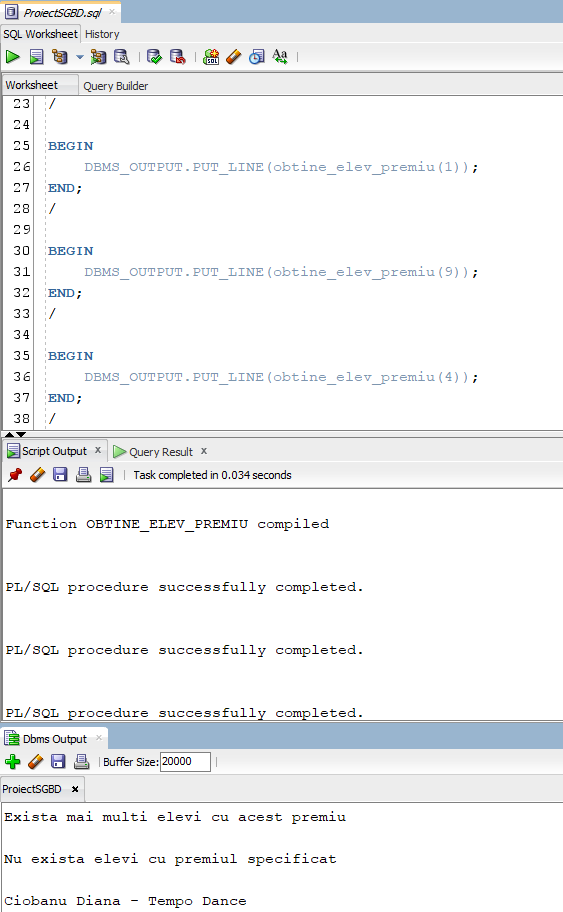


# **Cerința 8**

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

Cerința problemei: Identificați numele complet al unui singur elev și școala la care acesta este înscris, care a obținut un anumit premiu într-o competiție. În cazul în care nu există niciun elev cu premiul specificat, trebuie returnat un mesaj corespunzător. Dacă există mai mulți elevi cu același premiu, se va semnala o eroare indicând această situație.

| CREATE OR REPLACE FUNCTION obtine\_elev\_premiu (  p\_premiu INT  ) RETURN VARCHAR2 IS  v\_rezultat VARCHAR2(4000);  BEGIN  SELECT e.nume || ' ' || e.prenume || ' - ' || s.denumire  INTO v\_rezultat  FROM Elev e  JOIN Scoala s ON e.id\_scoala = s.id\_scoala  JOIN Participa p ON e.id\_elev = p.id\_elev  WHERE p.premiu = p\_premiu;  RETURN v\_rezultat;  EXCEPTION  WHEN NO\_DATA\_FOUND THEN  RETURN 'Nu exista elevi cu premiul specificat';  WHEN TOO\_MANY\_ROWS THEN  RETURN 'Exista mai multi elevi cu acest premiu';  WHEN OTHERS THEN  RETURN 'Alta eroare!';  END obtine\_elev\_premiu;  /  BEGIN  DBMS\_OUTPUT.PUT\_LINE(obtine\_elev\_premiu(1));  END;  /  BEGIN  DBMS\_OUTPUT.PUT\_LINE(obtine\_elev\_premiu(9));  END;  /  BEGIN  DBMS\_OUTPUT.PUT\_LINE(obtine\_elev\_premiu(4));  END;  / |
| --- |

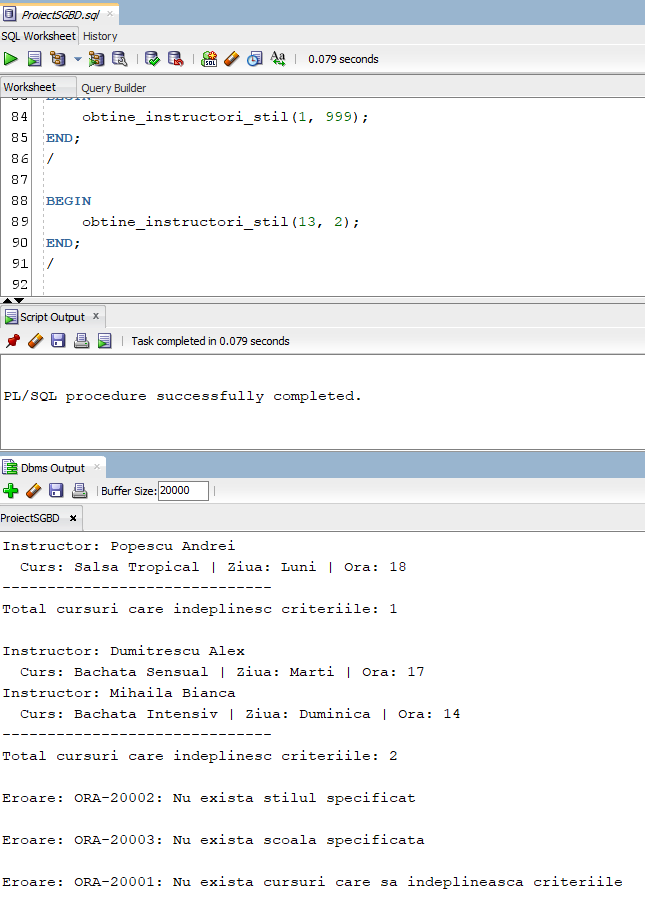


# **Cerința 9**

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

Cerința problemei: Obțineți lista instructorilor care predau cursuri dintr-un anumit stil de dans într-o anumită școală, incluzând denumirea cursurilor, ziua și ora acestora. Procedura trebuie să verifice existența stilului de dans și a școlii specificate și să semnaleze lipsa acestora prin mesaje de eroare personalizate. În cazul în care nu există cursuri care să corespundă criteriilor, procedura va semnala acest lucru printr-o altă eroare personalizată. Lista instructorilor și cursurilor va fi afișată într-un mod structurat, iar procedura va indica la final numărul total de cursuri care respectă criteriile. Procedura trebuie să utilizeze, într-o singură interogare SQL, cinci tabele din baza de date: Instructor, Preda, Curs, Stil și Scoala.

| CREATE OR REPLACE PROCEDURE obtine\_instructori\_stil (  p\_id\_stil IN INT,  p\_id\_scoala IN INT  ) IS  v\_count\_stil INT;  v\_count\_scoala INT;  v\_count\_cursuri INT;  BEGIN  SELECT COUNT(\*)  INTO v\_count\_stil  FROM Stil  WHERE id\_stil = p\_id\_stil;  IF v\_count\_stil = 0 THEN  RAISE\_APPLICATION\_ERROR(-20002, 'Nu exista stilul specificat');  END IF;  SELECT COUNT(\*)  INTO v\_count\_scoala  FROM Scoala  WHERE id\_scoala = p\_id\_scoala;  IF v\_count\_scoala = 0 THEN  RAISE\_APPLICATION\_ERROR(-20003, 'Nu exista scoala specificata');  END IF;  SELECT COUNT(\*)  INTO v\_count\_cursuri  FROM Instructor i  JOIN Preda pr ON i.id\_instructor = pr.id\_instructor  JOIN Curs c ON pr.id\_curs = c.id\_curs  JOIN Stil s ON c.id\_stil = s.id\_stil  JOIN Scoala sc ON pr.id\_scoala = sc.id\_scoala  WHERE c.id\_stil = p\_id\_stil AND pr.id\_scoala = p\_id\_scoala;  IF v\_count\_cursuri = 0 THEN  RAISE\_APPLICATION\_ERROR(-20001, 'Nu exista cursuri care sa indeplineasca criteriile');  END IF;  FOR rec IN (  SELECT i.nume || ' ' || i.prenume AS nume\_instructor,  c.denumire AS curs,  pr.ziua AS ziua,  pr.ora AS ora  FROM Instructor i  JOIN Preda pr ON i.id\_instructor = pr.id\_instructor  JOIN Curs c ON pr.id\_curs = c.id\_curs  JOIN Stil s ON c.id\_stil = s.id\_stil  JOIN Scoala sc ON pr.id\_scoala = sc.id\_scoala  WHERE c.id\_stil = p\_id\_stil AND pr.id\_scoala = p\_id\_scoala  ORDER BY i.nume, c.denumire  ) LOOP  DBMS\_OUTPUT.PUT\_LINE('Instructor: ' || rec.nume\_instructor);  DBMS\_OUTPUT.PUT\_LINE(' Curs: ' || rec.curs || ' | Ziua: ' || rec.ziua || ' | Ora: ' || rec.ora);  END LOOP;    DBMS\_OUTPUT.PUT\_LINE('------------------------------');  DBMS\_OUTPUT.PUT\_LINE('Total cursuri care indeplinesc criteriile: ' || v\_count\_cursuri);  EXCEPTION  WHEN NO\_DATA\_FOUND THEN  RAISE\_APPLICATION\_ERROR(-20004, 'Date inexistente');  WHEN OTHERS THEN  DBMS\_OUTPUT.PUT\_LINE('Eroare: ' || SQLERRM);  END obtine\_instructori\_stil;  /  BEGIN  obtine\_instructori\_stil(1, 1);  END;  /  BEGIN  obtine\_instructori\_stil(2, 2);  END;  /  BEGIN  obtine\_instructori\_stil(999, 1);  END;  /  BEGIN  obtine\_instructori\_stil(1, 999);  END;  /  BEGIN  obtine\_instructori\_stil(13, 2);  END;  / |
| --- |

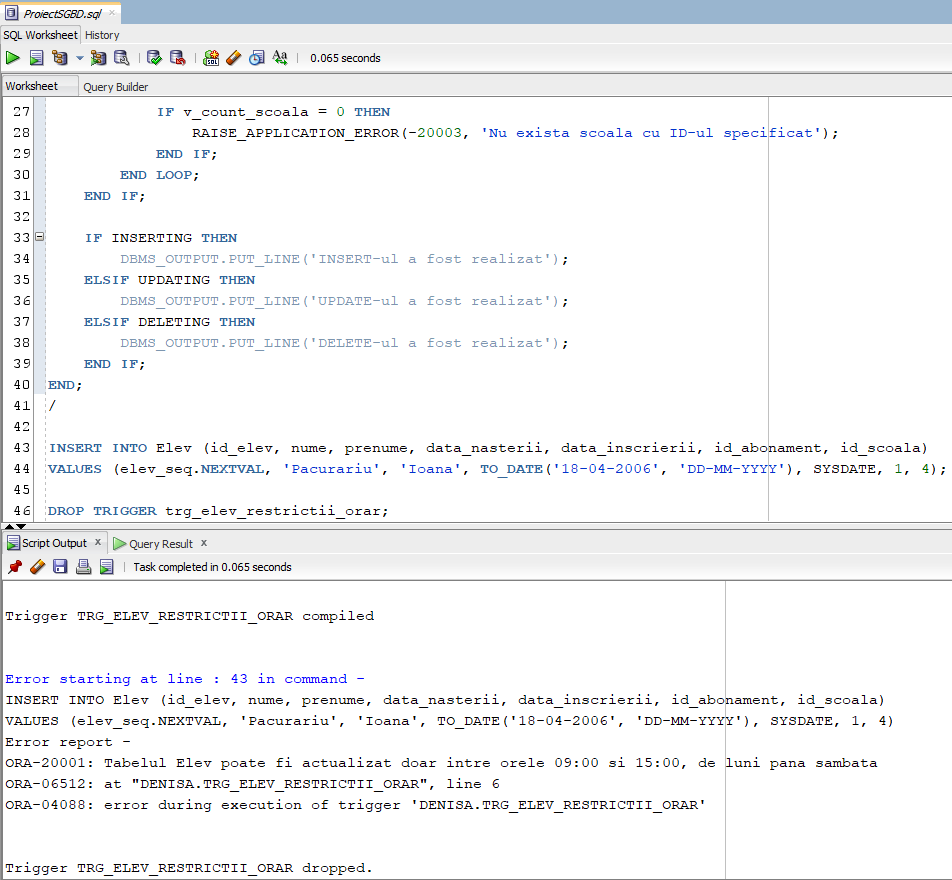


# **Cerința 10**

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

Acest trigger restricționează operațiunile de INSERT, UPDATE și DELETE asupra tabelului Elev la intervalul orar 9:00 - 15:00, de luni până sâmbătă, și permite accesul doar utilizatorului specificat (DENISA). De asemenea, validează existența școlii (id\_scoala) pentru operațiunile de tip INSERT și UPDATE, blocând modificările dacă școala nu există. La fiecare operațiune efectuată, trigger-ul afișează un mesaj specific.

| CREATE OR REPLACE TRIGGER trg\_elev\_restrictii\_orar  BEFORE INSERT OR UPDATE OR DELETE ON Elev  DECLARE  v\_username VARCHAR2(100);  v\_count\_scoala NUMBER;  BEGIN  IF TO\_CHAR(SYSDATE, 'D') = 1 OR TO\_CHAR(SYSDATE, 'HH24') NOT BETWEEN 9 AND 15 THEN  RAISE\_APPLICATION\_ERROR(-20001, 'Tabelul Elev poate fi actualizat doar intre orele 09:00 si 15:00, de luni pana sambata');  END IF;  SELECT USER INTO v\_username FROM DUAL;  IF v\_username NOT IN ('DENISA') THEN  RAISE\_APPLICATION\_ERROR(-20002, 'Doar utilizatorul DENISA poate actualiza tabelul Elev');  END IF;  IF INSERTING OR UPDATING THEN  FOR rec IN (  SELECT DISTINCT id\_scoala  FROM Elev  ) LOOP  SELECT COUNT(\*)  INTO v\_count\_scoala  FROM Scoala  WHERE id\_scoala = rec.id\_scoala;  IF v\_count\_scoala = 0 THEN  RAISE\_APPLICATION\_ERROR(-20003, 'Nu exista scoala cu ID-ul specificat');  END IF;  END LOOP;  END IF;  IF INSERTING THEN  DBMS\_OUTPUT.PUT\_LINE('INSERT-ul a fost realizat');  ELSIF UPDATING THEN  DBMS\_OUTPUT.PUT\_LINE('UPDATE-ul a fost realizat');  ELSIF DELETING THEN  DBMS\_OUTPUT.PUT\_LINE('DELETE-ul a fost realizat');  END IF;  END;  /  INSERT INTO Elev (id\_elev, nume, prenume, data\_nasterii, data\_inscrierii, id\_abonament, id\_scoala)  VALUES (elev\_seq.NEXTVAL, 'Pacurariu', 'Ioana', TO\_DATE('18-04-2006', 'DD-MM-YYYY'), SYSDATE, 1, 4);  DROP TRIGGER trg\_elev\_restrictii\_orar; |
| --- |

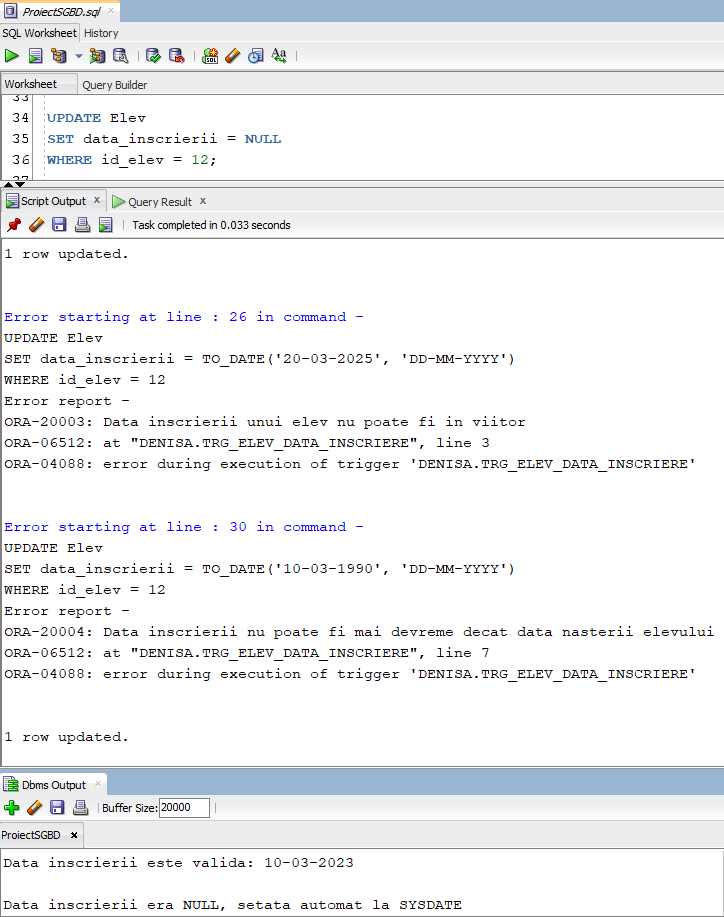


# **Cerința 11**

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

Acest trigger verifică, pentru fiecare modificare a câmpului data\_inscrierii din tabela Elev, ca data introdusă să nu fie în viitor și să nu fie mai devreme decât data nașterii elevului. De asemenea, dacă valoarea este NULL, trigger-ul o setează automat la data curentă (SYSDATE) și afișează un mesaj, indicând validitatea sau corectarea automată a datei.

| CREATE OR REPLACE TRIGGER trg\_elev\_data\_inscriere  BEFORE UPDATE OF data\_inscrierii ON Elev  FOR EACH ROW  BEGIN  IF :NEW.data\_inscrierii > SYSDATE THEN  RAISE\_APPLICATION\_ERROR(-20003, 'Data inscrierii unui elev nu poate fi in viitor');  END IF;  IF :NEW.data\_inscrierii < :OLD.data\_nasterii THEN  RAISE\_APPLICATION\_ERROR(-20004, 'Data inscrierii nu poate fi mai devreme decat data nasterii elevului');  END IF;  IF :NEW.data\_inscrierii IS NULL THEN  :NEW.data\_inscrierii := SYSDATE;  DBMS\_OUTPUT.PUT\_LINE('Data inscrierii era NULL, setata automat la SYSDATE');  ELSE  DBMS\_OUTPUT.PUT\_LINE('Data inscrierii este valida: ' || TO\_CHAR(:NEW.data\_inscrierii, 'DD-MM-YYYY'));  END IF;  END;  /  UPDATE Elev  SET data\_inscrierii = TO\_DATE('10-03-2023', 'DD-MM-YYYY')  WHERE id\_elev = 12;  UPDATE Elev  SET data\_inscrierii = TO\_DATE('20-03-2025', 'DD-MM-YYYY')  WHERE id\_elev = 12;  UPDATE Elev  SET data\_inscrierii = TO\_DATE('10-03-1990', 'DD-MM-YYYY')  WHERE id\_elev = 12;  UPDATE Elev  SET data\_inscrierii = NULL  WHERE id\_elev = 12;  DROP TRIGGER trg\_elev\_data\_inscriere; |
| --- |

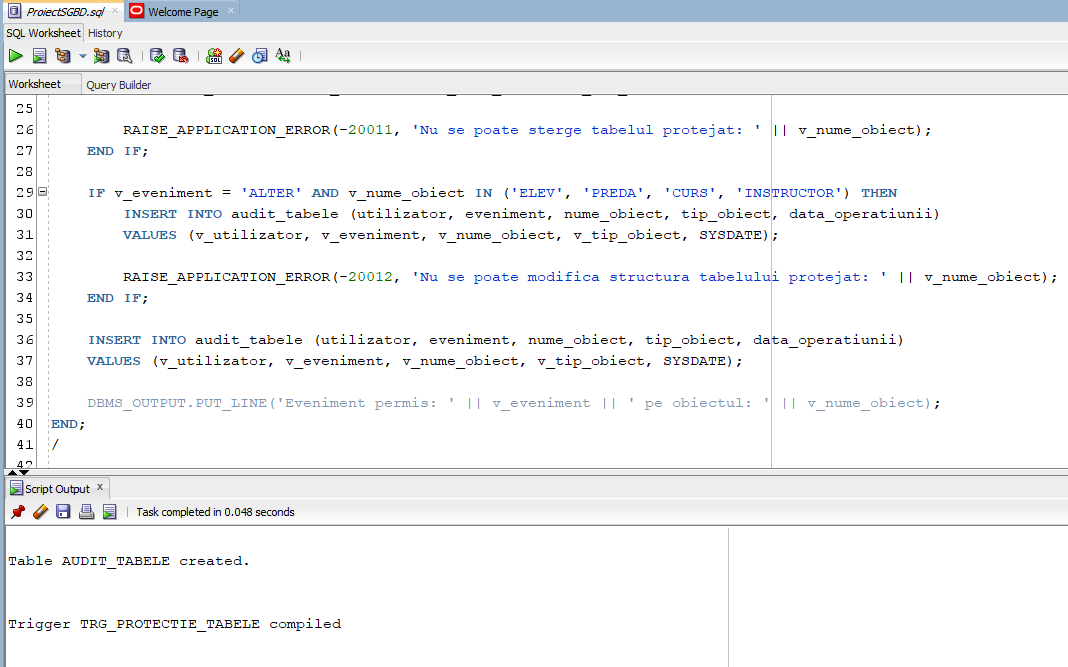


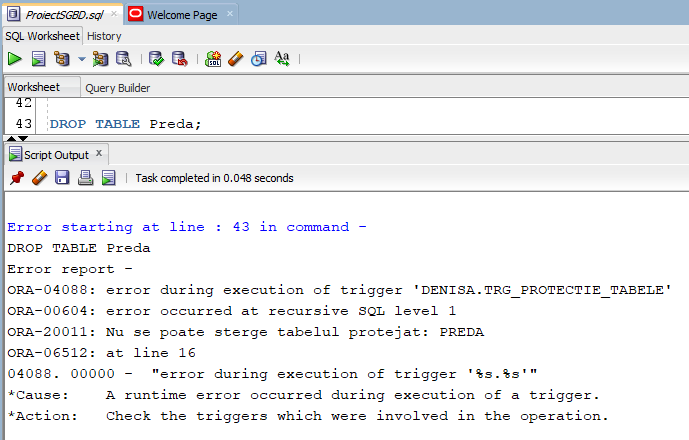
# **Cerința 12**

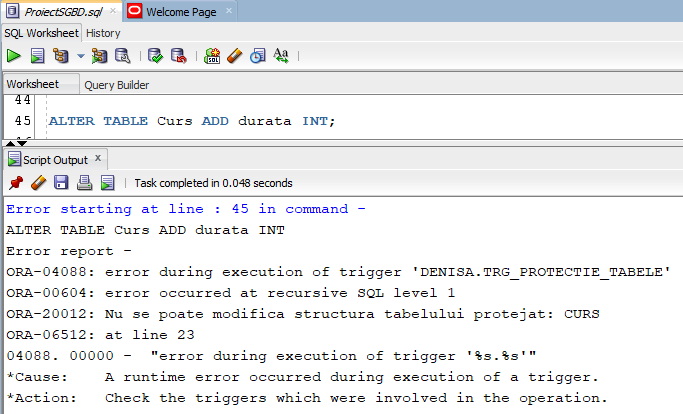
Definiți un trigger de tip LDD. Declanșați trigger-ul.

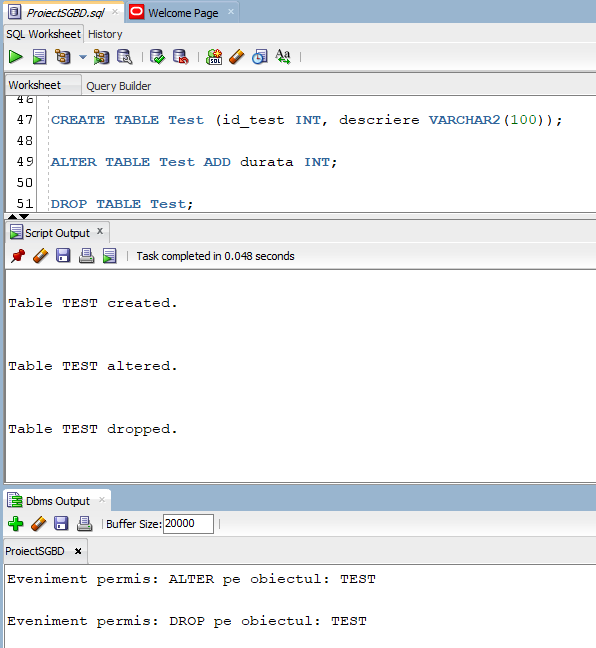
Acest trigger protejează tabelele critice ELEV, PREDA, CURS și INSTRUCTOR împotriva ștergerii și modificării structurilor lor, adăugând toate operațiunile DROP și ALTER într-un tabel de audit (audit\_tabele). Pentru fiecare eveniment, trigger-ul înregistrează utilizatorul care a inițiat operațiunea, tipul operațiunii, numele obiectului afectat, tipul obiectului și data operațiunii. Dacă operațiunea vizează un tabel protejat, aceasta este blocată și se generează o eroare personalizată. Pentru operațiunile permise, trigger-ul oferă feedback prin mesaje clare și loghează detalii în tabelul de audit.

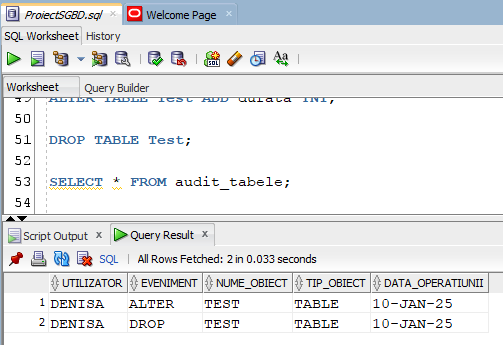
| CREATE TABLE audit\_tabele (  utilizator VARCHAR2(30),  eveniment VARCHAR2(20),  nume\_obiect VARCHAR2(30),  tip\_obiect VARCHAR2(30),  data\_operatiunii DATE  );  CREATE OR REPLACE TRIGGER trg\_protectie\_tabele  AFTER DROP OR ALTER ON SCHEMA  DECLARE  v\_utilizator VARCHAR2(30);  v\_eveniment VARCHAR2(20);  v\_nume\_obiect VARCHAR2(30);  v\_tip\_obiect VARCHAR2(30);  BEGIN  v\_utilizator := SYS.LOGIN\_USER;  v\_eveniment := ORA\_SYSEVENT;  v\_nume\_obiect := ORA\_DICT\_OBJ\_NAME;  v\_tip\_obiect := ORA\_DICT\_OBJ\_TYPE;  IF v\_eveniment = 'DROP' AND v\_nume\_obiect IN ('ELEV', 'PREDA', 'CURS', 'INSTRUCTOR') THEN  INSERT INTO audit\_tabele (utilizator, eveniment, nume\_obiect, tip\_obiect, data\_operatiunii)  VALUES (v\_utilizator, v\_eveniment, v\_nume\_obiect, v\_tip\_obiect, SYSDATE);  RAISE\_APPLICATION\_ERROR(-20011, 'Nu se poate sterge tabelul protejat: ' || v\_nume\_obiect);  END IF;  IF v\_eveniment = 'ALTER' AND v\_nume\_obiect IN ('ELEV', 'PREDA', 'CURS', 'INSTRUCTOR') THEN  INSERT INTO audit\_tabele (utilizator, eveniment, nume\_obiect, tip\_obiect, data\_operatiunii)  VALUES (v\_utilizator, v\_eveniment, v\_nume\_obiect, v\_tip\_obiect, SYSDATE);  RAISE\_APPLICATION\_ERROR(-20012, 'Nu se poate modifica structura tabelului protejat: ' || v\_nume\_obiect);  END IF;  INSERT INTO audit\_tabele (utilizator, eveniment, nume\_obiect, tip\_obiect, data\_operatiunii)  VALUES (v\_utilizator, v\_eveniment, v\_nume\_obiect, v\_tip\_obiect, SYSDATE);  DBMS\_OUTPUT.PUT\_LINE('Eveniment permis: ' || v\_eveniment || ' pe obiectul: ' || v\_nume\_obiect);  END;  /  DROP TABLE Preda;  ALTER TABLE Curs ADD durata INT;  CREATE TABLE Test (id\_test INT, descriere VARCHAR2(100));  ALTER TABLE Test ADD durata INT;  DROP TABLE Test;  SELECT \* FROM audit\_tabele;  DROP TRIGGER trg\_protectie\_tabele; |
| --- |











# **Cerința 13**

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).

Cerința problemei: Dezvoltați un pachet care să gestioneze procesul de organizare și monitorizare a competițiilor academiei de dans, utilizând funcționalități integrate și tipuri de date complexe. Pachetul trebuie să includă un tabel imbricat pentru gestionarea premiilor obținute într-o competiție și un tip RECORD pentru stocarea detaliilor competițiilor. De asemenea, trebuie să conțină două funcții: una care calculează numărul total de premii obținute de o școală într-o competiție și alta care verifică eligibilitatea organizării unei competiții la o anumită dată, asigurându-se că nu există suprapuneri (să nu existe două competiții la distanță mai mică de 7 zile). În plus, pachetul trebuie să includă două proceduri: una pentru înregistrarea unei noi competiții, care să integreze regulile de eligibilitate, și alta pentru generarea unui raport detaliat despre competițiile organizate și premiile acordate.

| CREATE OR REPLACE PACKAGE pkg\_gestionare\_competitii IS  TYPE premiu\_table IS TABLE OF NUMBER;  TYPE competitie\_record IS RECORD (  id\_competitie NUMBER,  denumire VARCHAR2(100),  data\_competitie DATE  );  FUNCTION calculeaza\_premii\_scoala(p\_id\_competitie NUMBER, p\_id\_scoala NUMBER) RETURN NUMBER;  FUNCTION verifica\_eligibilitate\_competitie(p\_data DATE) RETURN BOOLEAN;  PROCEDURE inregistreaza\_competitie(p\_id\_competitie NUMBER, p\_denumire VARCHAR2, p\_data DATE);  PROCEDURE genereaza\_raport\_competitii;  END pkg\_gestionare\_competitii;  /  CREATE OR REPLACE PACKAGE BODY pkg\_gestionare\_competitii IS  FUNCTION calculeaza\_premii\_scoala(p\_id\_competitie NUMBER, p\_id\_scoala NUMBER) RETURN NUMBER IS  v\_numar\_premii NUMBER;  BEGIN  SELECT COUNT(p.premiu)  INTO v\_numar\_premii  FROM Participa p  JOIN Elev e ON p.id\_elev = e.id\_elev  WHERE p.id\_competitie = p\_id\_competitie AND e.id\_scoala = p\_id\_scoala;  RETURN v\_numar\_premii;  END;  FUNCTION verifica\_eligibilitate\_competitie(p\_data DATE) RETURN BOOLEAN IS  v\_numar\_competitii NUMBER;  BEGIN  SELECT COUNT(\*)  INTO v\_numar\_competitii  FROM Competitie  WHERE ABS(TRUNC(p\_data) - TRUNC(data)) < 7;  RETURN v\_numar\_competitii = 0;  END;  PROCEDURE inregistreaza\_competitie(p\_id\_competitie NUMBER, p\_denumire VARCHAR2, p\_data DATE) IS  v\_eligibil BOOLEAN;  BEGIN  v\_eligibil := verifica\_eligibilitate\_competitie(p\_data);  IF NOT v\_eligibil THEN  RAISE\_APPLICATION\_ERROR(-20001, 'Competitia nu poate fi organizata. Exista deja o alta competitie planificata in aceeasi saptamana');  END IF;  INSERT INTO Competitie (id\_competitie, denumire, data)  VALUES (p\_id\_competitie, p\_denumire, p\_data);  DBMS\_OUTPUT.PUT\_LINE('Competitia a fost inregistrata cu succes');  END;  PROCEDURE genereaza\_raport\_competitii IS  CURSOR c\_competitii IS  SELECT c.id\_competitie, c.denumire, c.data  FROM Competitie c;  v\_competitie c\_competitii%ROWTYPE;  v\_numar\_premii NUMBER;  BEGIN  OPEN c\_competitii;  LOOP  FETCH c\_competitii INTO v\_competitie;  EXIT WHEN c\_competitii%NOTFOUND;  DBMS\_OUTPUT.PUT\_LINE('Competitie: ' || v\_competitie.denumire || ' | Data: ' || TO\_CHAR(v\_competitie.data, 'DD-MM-YYYY'));  SELECT COUNT(p.premiu)  INTO v\_numar\_premii  FROM Participa p  WHERE p.id\_competitie = v\_competitie.id\_competitie;  DBMS\_OUTPUT.PUT\_LINE(' Total premii acordate: ' || v\_numar\_premii);  DBMS\_OUTPUT.PUT\_LINE('------------------------------');  END LOOP;  CLOSE c\_competitii;  END;  END pkg\_gestionare\_competitii;  /  BEGIN  DBMS\_OUTPUT.PUT\_LINE('Premii scoala: ' || pkg\_gestionare\_competitii.calculeaza\_premii\_scoala(1, 1));  END;  /  BEGIN  pkg\_gestionare\_competitii.inregistreaza\_competitie(competitie\_seq.NEXTVAL, 'Bucurie in Miscare', TO\_DATE('15-04-2025', 'DD-MM-YYYY'));  END;  /  BEGIN  IF pkg\_gestionare\_competitii.verifica\_eligibilitate\_competitie(TO\_DATE('17-04-2025', 'DD-MM-YYYY')) THEN  DBMS\_OUTPUT.PUT\_LINE('Competitia este eligibila');  ELSE  DBMS\_OUTPUT.PUT\_LINE('Competitia NU este eligibila');  END IF;  END;  /  BEGIN  pkg\_gestionare\_competitii.genereaza\_raport\_competitii;  END;  / |
| --- |

