**Sisteme de Gestiune a Bazelor de Date**

**Gestiunea turneului de fotbal al ASMI**

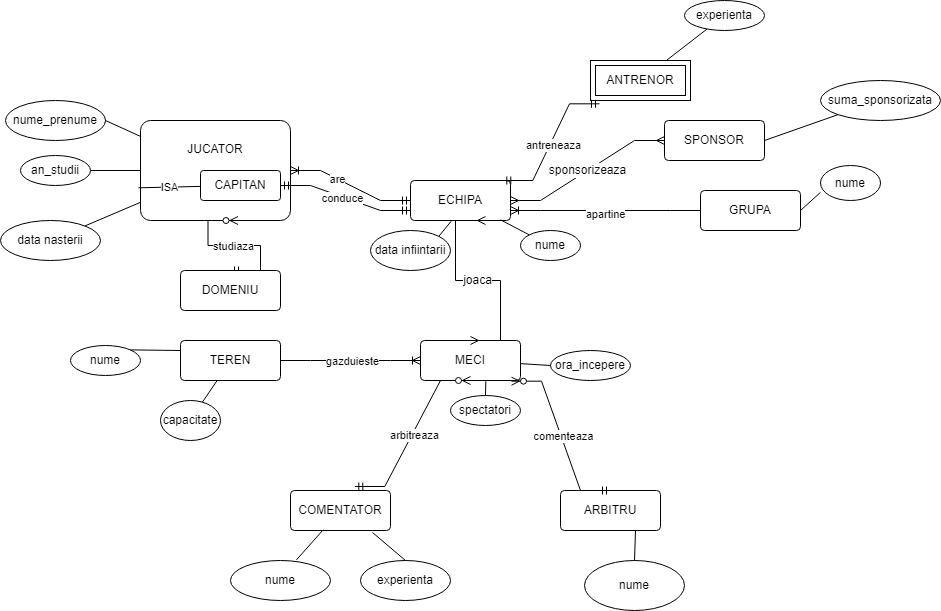
Pascu Ionuț Alexandru

Grupa 252

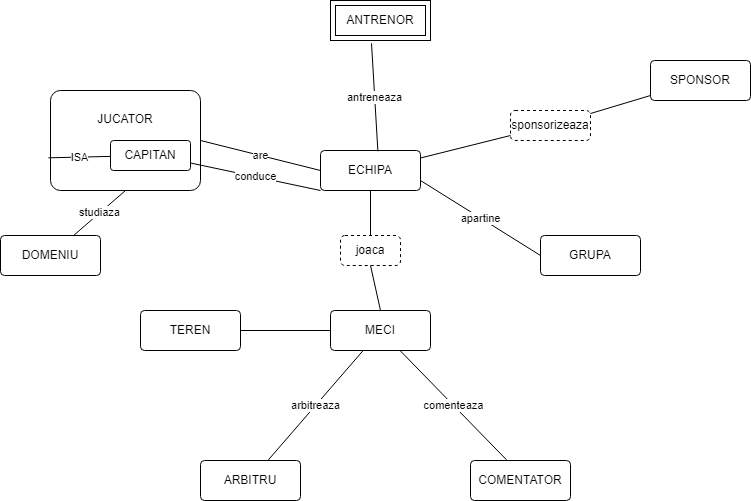
**1. Descrierea modelului real, a utilității acestuia și a regulilor de funcționare.**

În fiecare an, Asociația Studenților la Matematică și Informatică organizează un turneu de fotbal la care se pot înscrie studenți din toată Facultatea de Matematică-Informatică. La turneu participă mai multe echipe formate din studenți (jucători). Echipele sunt împărțite în mai multe grupe. În fiecare grupă se desfășoară mai multe meciuri între echipele din acea grupă. Meciurile se joacă pe mai multe terenuri și au o oră de început. Fiecare meci este arbitrat de un arbitru și are un comentator. Fiecare echipă are un antrenor și un căpitan. Fiecare echipă poate avea mai mulți sponsori. Fiecare jucător este student la o anumit domeniu de licență.

**2. Realizați diagrama entitate-relație (ERD).**



**3. Pornind de la diagrama entitate-relație realizați diagrama conceptuală a modelului propus, integrând toate atributele necesare.**

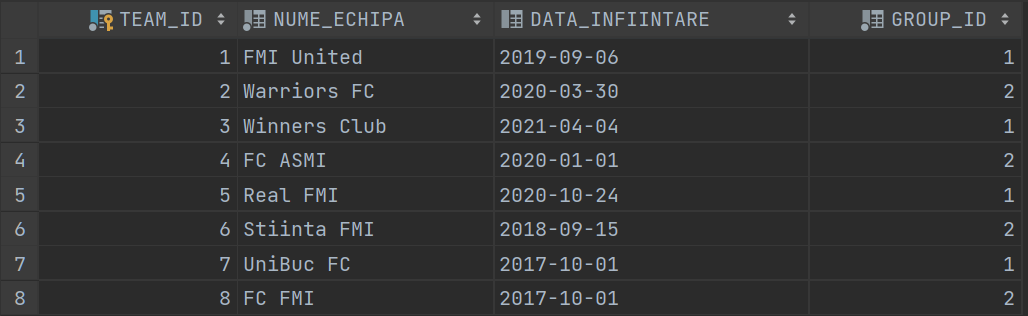
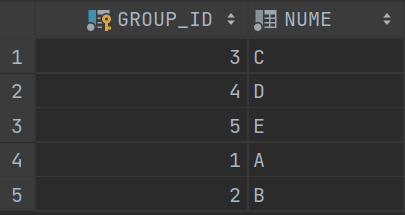


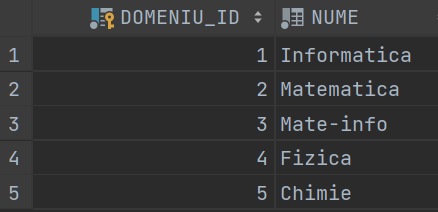
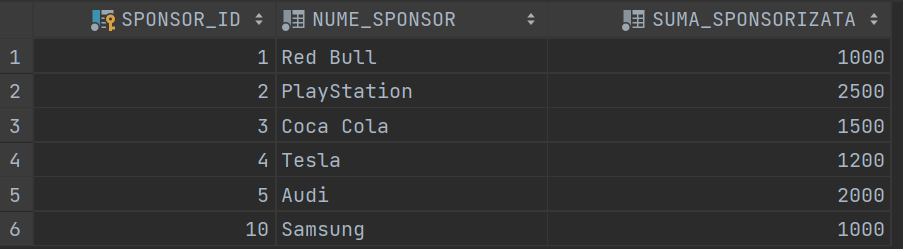
**4. Implementați în Oracle diagrama conceptuală realizată: definiți toate tabelele, implementând toate constrângerile de integritate necesare (chei primare, cheile externe etc).**

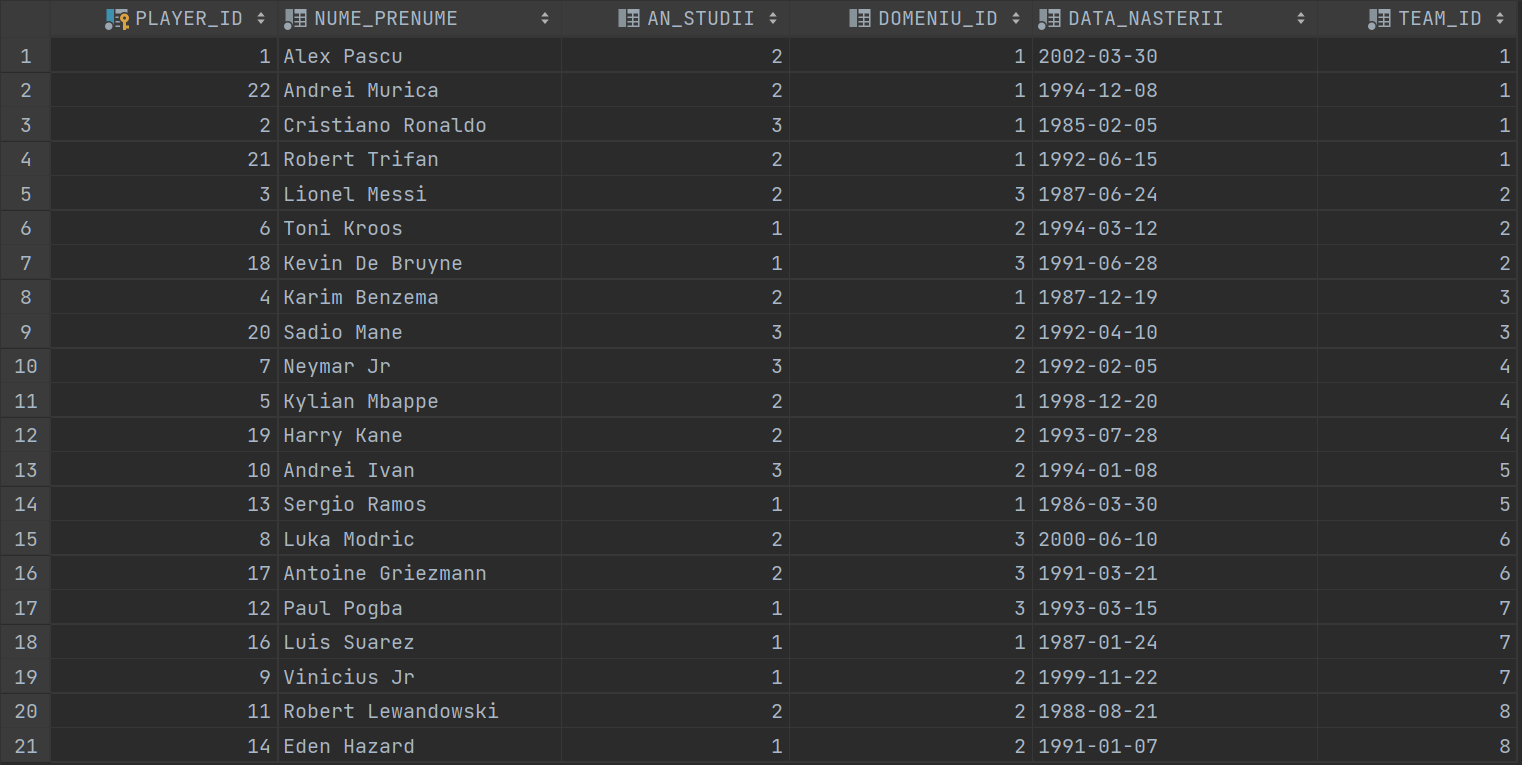
DROP TABLE echipa;  
CREATE TABLE echipa(  
 team\_id number(5) PRIMARY KEY,  
 nume\_echipa varchar2(30) NOT NULL,  
 data\_infiintare date default '01-01-2022',  
 group\_id number(5) NOT NULL  
);  
  
DROP TABLE sponsor;  
CREATE TABLE sponsor(  
 sponsor\_id number(5) PRIMARY KEY,  
 nume\_sponsor varchar2(30) NOT NULL,  
 suma\_sponsorizata number(8) NOT NULL  
);  
  
DROP TABLE jucator;  
CREATE TABLE jucator(  
 player\_id number(5) PRIMARY KEY,  
 nume\_prenume varchar2(30) NOT NULL,  
 an\_studii number(3),  
 domeniu\_id number(5),  
 data\_nasterii date NOT NULL,  
 team\_id number(5) NOT NULL  
);  
  
DROP TABLE sponsorizeaza;  
CREATE TABLE sponsorizeaza(  
 team\_id number(5) NOT NULL,  
 sponsor\_id number(5) NOT NULL,  
 CONSTRAINT sponsorizeaza\_id PRIMARY KEY(team\_id, sponsor\_id)  
);  
  
  
DROP TABLE joaca;  
CREATE TABLE joaca(  
 team\_id number(5) NOT NULL,  
 match\_id number(5) NOT NULL,  
 CONSTRAINT joaca\_id PRIMARY KEY(team\_id, match\_id)  
);  
  
DROP TABLE antrenor;  
CREATE TABLE antrenor(  
 coach\_id number(5) PRIMARY KEY,  
 team\_id number(5) NOT NULL,  
 experienta number(3) NOT NULL,  
 nume varchar2(30) NOT NULL,  
 CONSTRAINT team\_fk FOREIGN KEY (team\_id) REFERENCES echipa(team\_id)  
);  
  
DROP TABLE meci;  
CREATE TABLE meci(  
 match\_id number(5) PRIMARY KEY,  
 ora\_incepere varchar(10) NOT NULL,  
 spectatori number(10) NOT NULL,  
 commentator\_id number(5) NOT NULL,  
 referee\_id number(5) NOT NULL,  
 field\_id number(5) NOT NULL  
);  
  
DROP TABLE arbitru;  
CREATE TABLE arbitru(  
 referee\_id number(5) PRIMARY KEY,  
 nume varchar2(30) NOT NULL,  
 varsta number(3) NOT NULL  
);  
  
DROP TABLE comentator;  
CREATE TABLE comentator(  
 commentator\_id number(5) PRIMARY KEY,  
 nume varchar2(30) NOT NULL,  
 experienta number(3) NOT NULL  
);  
  
DROP TABLE teren;  
CREATE TABLE teren(  
 field\_id number(5) PRIMARY KEY,  
 nume varchar2(30) NOT NULL,  
 capacitate number(10) NOT NULL  
);  
  
DROP TABLE grupa;  
CREATE TABLE grupa(  
 group\_id number(5) PRIMARY KEY,  
 nume varchar2(30) NOT NULL  
);  
  
DROP TABLE domeniu;  
CREATE TABLE domeniu(  
 domeniu\_id number(5) PRIMARY KEY,  
 nume varchar2(30) NOT NULL  
);  
  
DROP TABLE capitan;  
CREATE TABLE capitan(  
 player\_id number(5) NOT NULL REFERENCES jucator(player\_id),  
 team\_id number(5) not null  
);

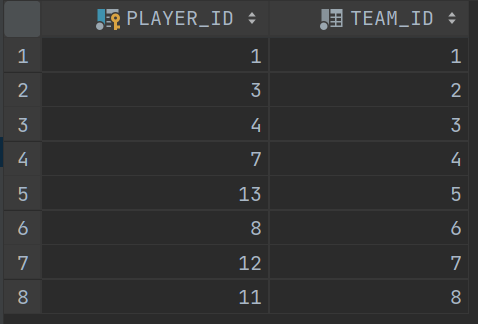
**5. Adăugați informații coerente în tabelele create (minim 5 înregistrări pentru fiecare entitate independentă; minim 10 înregistrări pentru tabela asociativă).**

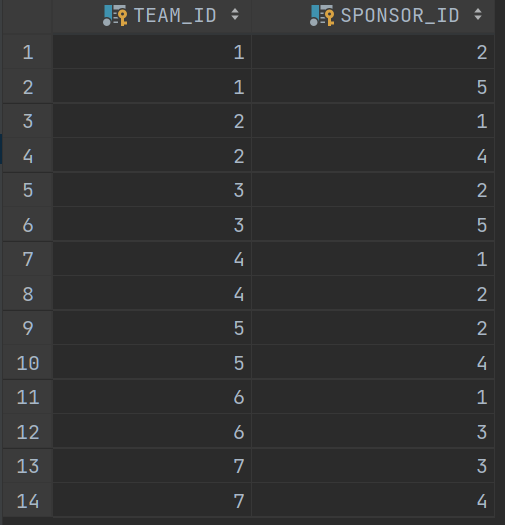
INSERT INTO GRUPA VALUES (1, 'A');  
INSERT INTO GRUPA VALUES (2, 'B');  
INSERT INTO GRUPA VALUES (3, 'C');  
INSERT INTO GRUPA VALUES (4, 'D');  
INSERT INTO GRUPA VALUES (5, 'E');  
-- select \* from GRUPA;  
  
  
CREATE SEQUENCE id\_echipa\_seq  
START WITH 1  
INCREMENT BY 1;  
  
INSERT INTO echipa VALUES(id\_echipa\_seq.nextval, 'FMI United', *TO\_DATE*('06-09-2019'), 1);  
INSERT INTO echipa VALUES(id\_echipa\_seq.nextval, 'Warriors FC', *TO\_DATE*('30-03-2020'), 2);  
INSERT INTO echipa VALUES(id\_echipa\_seq.nextval, 'Winners Club', *TO\_DATE*('04-04-2021'), 1);  
INSERT INTO echipa VALUES(id\_echipa\_seq.nextval, 'FC ASMI', *TO\_DATE*('01-01-2020'), 2);  
INSERT INTO echipa VALUES(id\_echipa\_seq.nextval, 'Real FMI', *TO\_DATE*('24-10-2020'), 1);  
INSERT INTO echipa VALUES(id\_echipa\_seq.nextval, 'Stiinta FMI', *TO\_DATE*('15-09-2018'), 2);  
INSERT INTO echipa VALUES(id\_echipa\_seq.nextval, 'UniBuc FC', *TO\_DATE*('01-10-2017'), 1);  
INSERT INTO echipa VALUES(id\_echipa\_seq.nextval, 'FC FMI', *TO\_DATE*('01-10-2017'), 2);  
-- SELECT \* FROM echipa ORDER BY GROUP\_ID;  
  
  
INSERT INTO sponsor VALUES(1, 'Red Bull', 1000);  
INSERT INTO sponsor VALUES(2, 'PlayStation', 2500);  
INSERT INTO sponsor VALUES(3, 'Coca Cola', 1500);  
INSERT INTO sponsor VALUES(4, 'Tesla', 1200);  
INSERT INTO sponsor VALUES(5, 'Audi', 2000);  
INSERT INTO SPONSOR VALUES (10, 'Samsung', 1000);  
-- SELECT \* FROM sponsor ORDER BY suma\_sponsorizata;  
  
  
INSERT INTO DOMENIU VALUES(1, 'Informatica');  
INSERT INTO DOMENIU VALUES(2, 'Matematica');  
INSERT INTO DOMENIU VALUES(3, 'Mate-info');  
INSERT INTO DOMENIU VALUES(4, 'Fizica');  
INSERT INTO DOMENIU VALUES(5, 'Chimie');  
-- SELECT \* FROM DOMENIU;  
  
  
INSERT INTO jucator VALUES(1, 'Alex Pascu', 2, 1, *TO\_DATE*('30-03-2002'), 1);  
INSERT INTO jucator VALUES(2, 'Cristiano Ronaldo', 3, 1, *TO\_DATE*('05-02-1985'), 1);  
INSERT INTO jucator VALUES(3, 'Lionel Messi', 2, 3, *TO\_DATE*('24-06-1987'), 2);  
INSERT INTO jucator VALUES(4, 'Karim Benzema', 2, 1, *TO\_DATE*('19-12-1987'), 3);  
INSERT INTO jucator VALUES(5, 'Kylian Mbappe', 2, 1, *TO\_DATE*('20-12-1998'), 4);  
INSERT INTO jucator VALUES(6, 'Toni Kroos', 1, 2, *TO\_DATE*('12-03-1994'), 2);  
INSERT INTO jucator VALUES(7, 'Neymar Jr', 3, 2, *TO\_DATE*('05-02-1992'), 4);  
INSERT INTO jucator VALUES(8, 'Luka Modric', 2, 3, *TO\_DATE*('10-06-2000'), 6);  
INSERT INTO jucator VALUES(9, 'Vinicius Jr', 1, 2, *TO\_DATE*('22-11-1999'), 7);  
INSERT INTO jucator VALUES(10, 'Andrei Ivan', 3, 2, *TO\_DATE*('08-01-1994'), 5);  
INSERT INTO jucator VALUES(11, 'Robert Lewandowski', 2, 2, *TO\_DATE*('21-08-1988'), 8);  
INSERT INTO jucator VALUES(12, 'Paul Pogba', 1, 3, *TO\_DATE*('15-03-1993'), 7);  
INSERT INTO jucator VALUES(13, 'Sergio Ramos', 1, 1, *TO\_DATE*('30-03-1986'), 5);  
INSERT INTO jucator VALUES(14, 'Eden Hazard', 1, 2, *TO\_DATE*('07-01-1991'), 8);  
INSERT INTO jucator VALUES(16, 'Luis Suarez', 1, 1, *TO\_DATE*('24-01-1987'), 7);  
INSERT INTO jucator VALUES(17, 'Antoine Griezmann', 2, 3, *TO\_DATE*('21-03-1991'), 6);  
INSERT INTO jucator VALUES(18, 'Kevin De Bruyne', 1, 3, *TO\_DATE*('28-06-1991'), 2);  
INSERT INTO jucator VALUES(19, 'Harry Kane', 2, 2, *TO\_DATE*('28-07-1993'), 4);  
INSERT INTO jucator VALUES(20, 'Sadio Mane', 3, 2, *TO\_DATE*('10-04-1992'), 3);  
INSERT INTO jucator VALUES(21, 'Robert Trifan', 2, 1, *TO\_DATE*('15-06-1992'), 1);  
INSERT INTO jucator VALUES(22, 'Andrei Murica', 2, 1, *TO\_DATE*('08-12-1994'), 1);  
-- SELECT \* FROM jucator ORDER BY TEAM\_ID;  
  
  
INSERT INTO CAPITAN VALUES(1, 1);  
INSERT INTO CAPITAN VALUES(3, 2);  
INSERT INTO CAPITAN VALUES(4, 3);  
INSERT INTO CAPITAN VALUES(7, 4);  
INSERT INTO CAPITAN VALUES(13, 5);  
INSERT INTO CAPITAN VALUES(8, 6);  
INSERT INTO CAPITAN VALUES(12, 7);  
INSERT INTO CAPITAN VALUES(11, 8);  
-- SELECT \* FROM CAPITAN ORDER BY TEAM\_ID;  
  
INSERT INTO sponsorizeaza (team\_id, sponsor\_id) VALUES(1, 5);  
INSERT INTO sponsorizeaza (team\_id, sponsor\_id) VALUES(1, 2);  
INSERT INTO sponsorizeaza (team\_id, sponsor\_id) VALUES(2, 1);  
INSERT INTO sponsorizeaza (team\_id, sponsor\_id) VALUES(2, 4);  
INSERT INTO sponsorizeaza (team\_id, sponsor\_id) VALUES(3, 5);  
INSERT INTO sponsorizeaza (team\_id, sponsor\_id) VALUES(3, 2);  
INSERT INTO sponsorizeaza (team\_id, sponsor\_id) VALUES(4, 2);  
INSERT INTO sponsorizeaza (team\_id, sponsor\_id) VALUES(4, 1);  
INSERT INTO sponsorizeaza (team\_id, sponsor\_id) VALUES(5, 2);  
INSERT INTO sponsorizeaza (team\_id, sponsor\_id) VALUES(5, 4);  
INSERT INTO sponsorizeaza (team\_id, sponsor\_id) VALUES(6, 1);  
INSERT INTO sponsorizeaza (team\_id, sponsor\_id) VALUES(6, 3);  
INSERT INTO sponsorizeaza (team\_id, sponsor\_id) VALUES(7, 3);  
INSERT INTO sponsorizeaza (team\_id, sponsor\_id) VALUES(7, 4);  
-- SELECT \* FROM sponsorizeaza;  
  
INSERT INTO COMENTATOR VALUES(1, 'Alexandru Popescu', 1);  
INSERT INTO COMENTATOR VALUES(2, 'Mihai Stoica', 2);  
INSERT INTO COMENTATOR VALUES(3, 'Andrei Ionescu', 3);  
INSERT INTO COMENTATOR VALUES(4, 'Cristian Ionescu', 3);  
INSERT INTO COMENTATOR VALUES(5, 'Mihai Popescu', 1);  
-- SELECT \* FROM comentator;  
  
INSERT INTO ARBITRU VALUES(1, 'Mihai Manea', 30);  
INSERT INTO ARBITRU VALUES(2, 'Andrei Stamate', 23);  
INSERT INTO ARBITRU VALUES(3, 'Cristian Ciobanu', 34);  
INSERT INTO ARBITRU VALUES(4, 'Alexandru Gheorghe', 27);  
INSERT INTO ARBITRU VALUES(5, 'Iustin Stoica', 20);  
-- SELECT \* FROM arbitru;  
  
INSERT INTO TEREN VALUES(1, 'Stadionul National', 5000);  
INSERT INTO TEREN VALUES(2, 'Stadionul Olimpic', 4000);  
INSERT INTO TEREN VALUES(3, 'Arena Leilor', 8000);  
INSERT INTO TEREN VALUES(4, 'Herastrau Park', 6400);  
INSERT INTO TEREN VALUES(5, 'Cotroceni Stadium', 7500);  
-- SELECT \* FROM teren ORDER BY capacitate;  
  
INSERT into MECI VALUES(1, '16:00', 7200, 1, 3, 3);  
INSERT into MECI VALUES(2, '18:30', 4500, 4, 2, 4);  
INSERT into MECI VALUES(3, '15:00', 5700, 5, 3, 5);  
INSERT into MECI VALUES(4, '19:00', 5000, 4, 2, 1);  
INSERT into MECI VALUES(5, '17:00', 3700, 3, 1, 2);  
INSERT into MECI VALUES(6, '20:00', 5000, 2, 4, 1);  
INSERT into MECI VALUES(7, '16:30', 3800, 2, 5, 4);  
INSERT into MECI VALUES(8, '18:00', 7000, 1, 4, 5);  
-- SELECT \* FROM meci ORDER BY ora\_inceput;  
  
INSERT INTO joaca (match\_id, team\_id) VALUES(1, 1);  
INSERT INTO joaca (match\_id, team\_id) VALUES(1, 3);  
INSERT INTO joaca (match\_id, team\_id) VALUES(2, 4);  
INSERT INTO joaca (match\_id, team\_id) VALUES(2, 2);  
INSERT INTO joaca (match\_id, team\_id) VALUES(3, 5);  
INSERT INTO joaca (match\_id, team\_id) VALUES(3, 7);  
INSERT INTO joaca (match\_id, team\_id) VALUES(4, 8);  
INSERT INTO joaca (match\_id, team\_id) VALUES(4, 6);  
INSERT INTO joaca (match\_id, team\_id) VALUES(5, 7);  
INSERT INTO joaca (match\_id, team\_id) VALUES(5, 1);  
INSERT INTO joaca (match\_id, team\_id) VALUES(6, 3);  
INSERT INTO joaca (match\_id, team\_id) VALUES(6, 5);  
INSERT INTO joaca (match\_id, team\_id) VALUES(7, 2);  
INSERT INTO joaca (match\_id, team\_id) VALUES(7, 8);  
INSERT INTO joaca (match\_id, team\_id) VALUES(8, 6);  
INSERT INTO joaca (match\_id, team\_id) VALUES(8, 4);  
-- SELECT \* FROM joaca ORDER BY match\_id;  
  
INSERT INTO antrenor VALUES(1, 1, 3, 'Carlo Ancelotti');  
INSERT INTO antrenor VALUES(2, 2, 1, 'Pep Guardiola');  
INSERT INTO antrenor VALUES(3, 3, 2, 'Dan Petrescu');  
INSERT INTO antrenor VALUES(4, 4, 2, 'Zinedine Zidane');  
INSERT INTO antrenor VALUES(5, 5, 2, 'Laurentiu Reghecampf');  
INSERT INTO antrenor VALUES(6, 6, 3, 'Gigi Becali');  
INSERT INTO antrenor VALUES(7, 7, 1, 'Mihai Rotaru');  
INSERT INTO antrenor VALUES(8, 8, 2, 'Andrei Pavel');  
-- SELECT \* FROM antrenor;  
COMMIT;  
ROLLBACK;

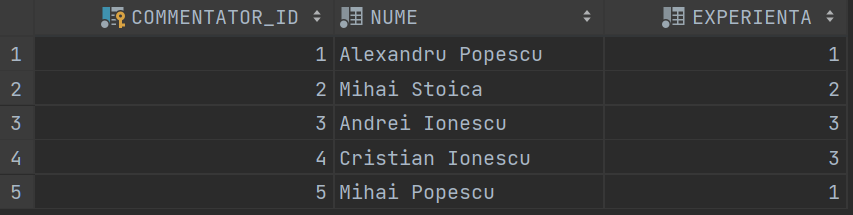
****

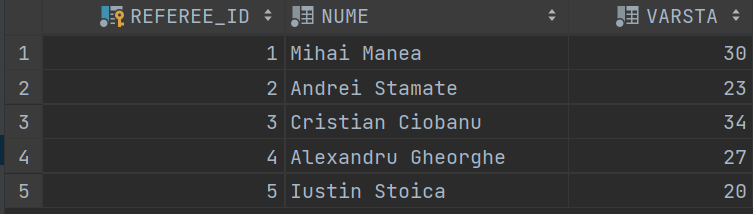
****

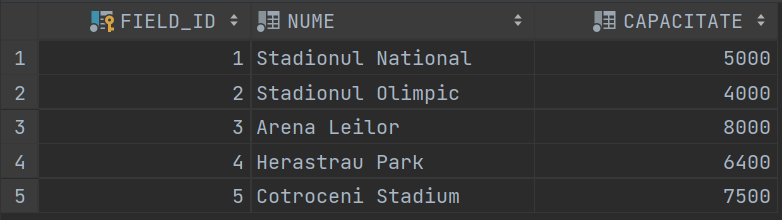
****

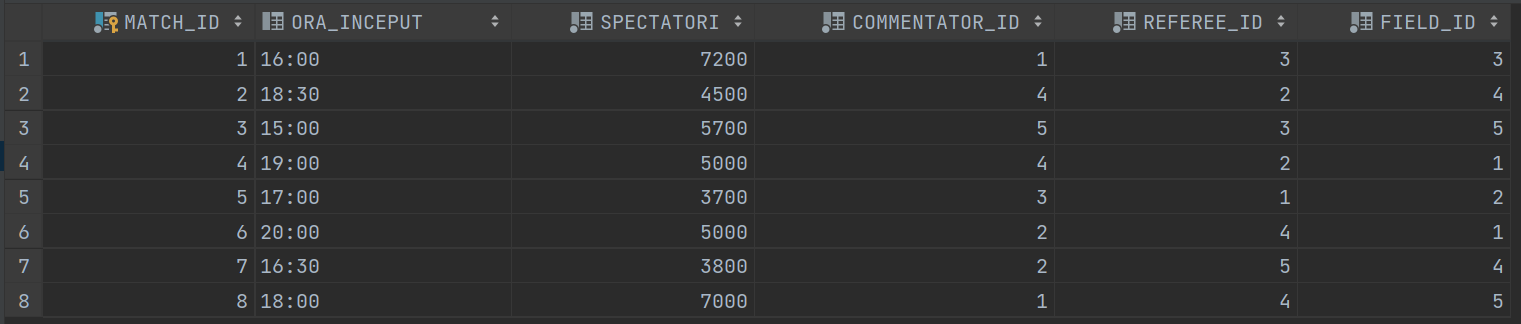
****

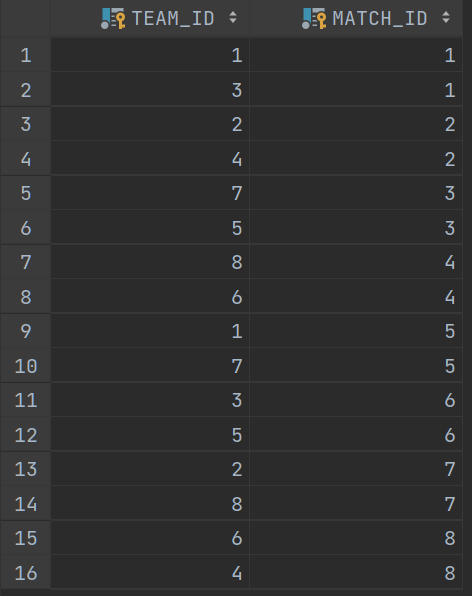
****

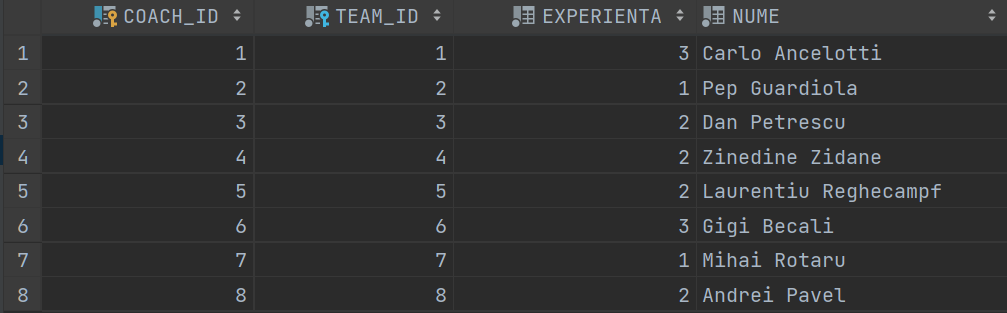
****

****

****

****

****

****

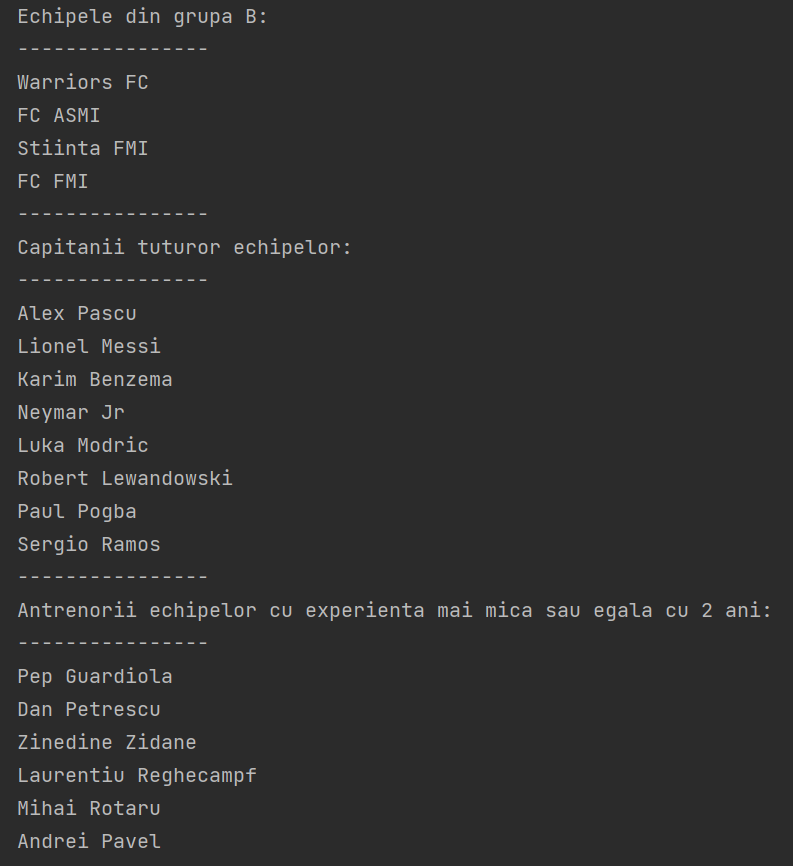
**6. Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat independent care să utilizeze două tipuri de colecție studiate. Apelați subprogramul.**

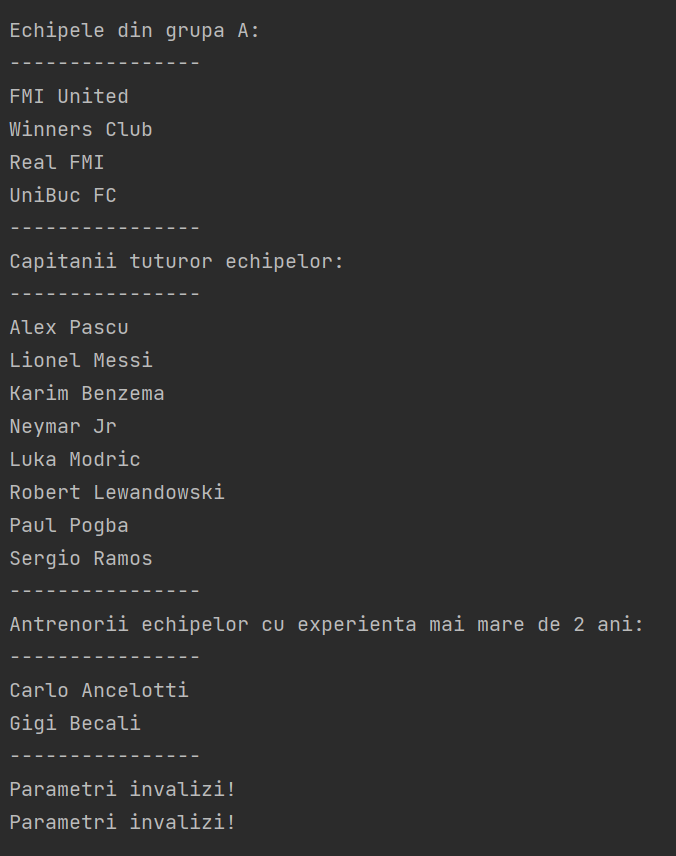
-- exercitiul 6  
-- Afisati toti jucatorii care participa la turneu, apoi doar jucatorii care  
-- fac parte din echipa cu un id dat  
CREATE OR REPLACE PROCEDURE *get\_players\_from\_team*(id\_echipa IN NUMBER)  
IS  
 TYPE jucatori is varray(100) of jucator%rowtype;  
 v\_jucatori jucatori;  
 v\_jucator jucator%rowtype;  
 TYPE toti\_jucatorii is table of jucator%rowtype;  
 v\_toti\_jucatorii toti\_jucatorii;  
 ECHIPA\_NEGASITA EXCEPTION;  
BEGIN  
 SELECT \* BULK COLLECT INTO v\_toti\_jucatorii from JUCATOR;  
 DBMS\_OUTPUT.*PUT\_LINE*('Toti jucatorii:');  
 DBMS\_OUTPUT.*PUT\_LINE*('----------------');  
 FOR i in 1..v\_toti\_jucatorii.COUNT loop  
 v\_jucator := v\_toti\_jucatorii(i);  
 DBMS\_OUTPUT.*PUT\_LINE*(v\_jucator.NUME\_PRENUME);  
 end loop;  
 SELECT \* bulk collect into v\_jucatori FROM JUCATOR WHERE TEAM\_ID = id\_echipa;  
 IF v\_jucatori.COUNT = 0 THEN  
 RAISE ECHIPA\_NEGASITA;  
 end if;  
 DBMS\_OUTPUT.*PUT\_LINE*('----------------');  
 DBMS\_OUTPUT.*PUT\_LINE*('Jucatorii din echipa cu id-ul ' || id\_echipa || ':');  
 DBMS\_OUTPUT.*PUT\_LINE*('----------------');  
 FOR i IN 1 .. v\_jucatori.COUNT LOOP  
 v\_jucator := v\_jucatori(i);  
 DBMS\_OUTPUT.*PUT\_LINE*(v\_jucator.NUME\_PRENUME);  
 END LOOP;  
 DBMS\_OUTPUT.*PUT\_LINE*('----------------');  
  
 EXCEPTION  
 WHEN ECHIPA\_NEGASITA THEN  
 DBMS\_OUTPUT.*PUT\_LINE*('Nu exista echipa cu id-ul ' || id\_echipa);  
  
END;  
-- apelarea procedurii  
BEGIN  
 *get\_players\_from\_team*(1);  
 *get\_players\_from\_team*(152);  
END;

****

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

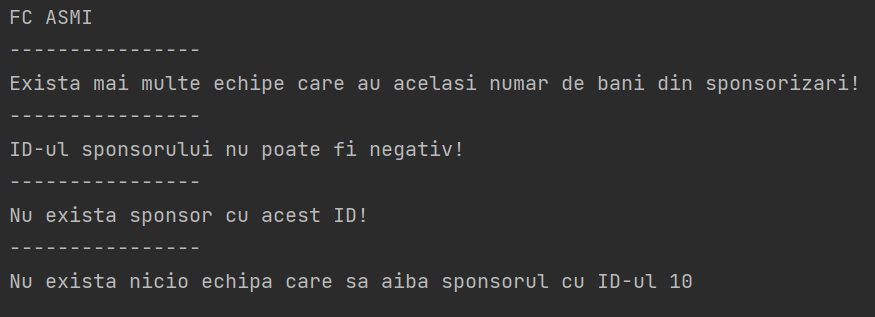
-- exercitiul 7  
-- Afisati echipele dintr-o grupa data, apoi capitanii tuturor echipelor  
-- apoi afisati antrenorii echipelor din grupa respectiva cu mai mult/mai putin de 2 ani experienta (daca p\_cursor = 1, mai mult, daca p\_cursor = 2, mai putin)  
CREATE OR REPLACE PROCEDURE *get\_teams\_captains\_coaches*(p\_grupa IN GRUPA.NUME%type, p\_cursor IN NUMBER)  
AS  
 cursor c\_teams (p\_grupa IN GRUPA.NUME%type) is  
 select \* from echipa e  
 join grupa g on e.GROUP\_ID = g.GROUP\_ID  
 where g.nume = p\_grupa;  
 cursor c\_captains is  
 select \* from jucator j  
 join capitan c on c.PLAYER\_ID = j.PLAYER\_ID;  
  
 v\_echipa c\_teams%rowtype;  
 v\_capitan c\_captains%rowtype;  
 TYPE tip\_cursor IS REF CURSOR RETURN antrenor%ROWTYPE;  
 v\_coach\_cursor tip\_cursor;  
  
 v\_coach ANTRENOR%rowtype;  
 INVALID\_INPUT EXCEPTION;  
BEGIN  
 IF p\_cursor > 2 OR p\_cursor < 1 THEN  
 RAISE INVALID\_INPUT;  
 end if;  
 DBMS\_OUTPUT.*PUT\_LINE*('Echipele din grupa ' || p\_grupa || ':');  
 DBMS\_OUTPUT.*PUT\_LINE*('----------------');  
 OPEN c\_teams(p\_grupa);  
 LOOP  
 FETCH c\_teams INTO v\_echipa;  
 EXIT WHEN c\_teams%NOTFOUND;  
 DBMS\_OUTPUT.*PUT\_LINE*(v\_echipa.NUME\_ECHIPA);  
 END LOOP;  
 CLOSE c\_teams;  
 DBMS\_OUTPUT.*PUT\_LINE*('----------------');  
 DBMS\_OUTPUT.*PUT\_LINE*('Capitanii tuturor echipelor:');  
 DBMS\_OUTPUT.*PUT\_LINE*('----------------');  
 OPEN c\_captains;  
 LOOP  
 FETCH c\_captains INTO v\_capitan;  
 EXIT WHEN c\_captains%NOTFOUND;  
 DBMS\_OUTPUT.*PUT\_LINE*(v\_capitan.nume\_prenume);  
 END LOOP;  
 CLOSE c\_captains;  
  
 IF p\_cursor = 1 THEN  
 OPEN v\_coach\_cursor FOR  
 SELECT \* FROM ANTRENOR WHERE EXPERIENTA > 2;  
 ELSIF p\_cursor = 2 THEN  
 OPEN v\_coach\_cursor FOR  
 SELECT \* FROM ANTRENOR WHERE EXPERIENTA <= 2;  
 END IF;  
  
 DBMS\_OUTPUT.*PUT\_LINE*('----------------');  
 if p\_cursor = 1 then  
 DBMS\_OUTPUT.*PUT\_LINE*('Antrenorii echipelor cu experienta mai mare de 2 ani:');  
 else  
 DBMS\_OUTPUT.*PUT\_LINE*('Antrenorii echipelor cu experienta mai mica sau egala cu 2 ani:');  
 end if;  
 DBMS\_OUTPUT.*PUT\_LINE*('----------------');  
 LOOP  
 FETCH v\_coach\_cursor INTO v\_coach;  
 EXIT WHEN v\_coach\_cursor%NOTFOUND;  
 DBMS\_OUTPUT.*PUT\_LINE*(v\_coach.nume);  
 END LOOP;  
 DBMS\_OUTPUT.*PUT\_LINE*('----------------');  
  
 EXCEPTION  
 WHEN INVALID\_INPUT THEN  
 DBMS\_OUTPUT.*PUT\_LINE*('Parametri invalizi!');  
  
END;  
-- apelarea procedurii  
BEGIN  
 *get\_teams\_captains\_coaches*('B', 2);  
 *get\_teams\_captains\_coaches*('A', 1);  
 *get\_teams\_captains\_coaches*('A', 3);  
 *get\_teams\_captains\_coaches*('C', -3);  
END;

****

****

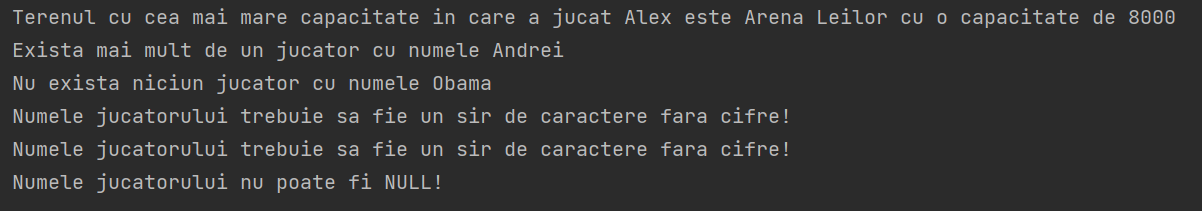
**8. Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat independent de tip funcție care să utilizeze într-o singură comandă SQL 3 dintre tabelele definite. Definiți minim 2 excepții. Apelați subprogramul astfel încât să evidențiați toate cazurile tratate.**

-- exercitiul 8  
-- Afisati echipa care primeste cei mai multi bani din sponsorizari  
-- avand un sponsor dat  
CREATE OR REPLACE FUNCTION *get\_team\_with\_most\_money*(p\_sponsor\_id IN NUMBER) RETURN VARCHAR2 AS  
 v\_nume\_echipa ECHIPA.nume\_echipa%type;  
 v\_suma NUMBER;  
 v\_s NUMBER;  
 v\_cnt NUMBER;  
 v\_sol ECHIPA.nume\_echipa%type;  
 v\_sponsor\_id NUMBER;  
 v\_cursor SYS\_REFCURSOR;  
 NUMAR\_NEGATIV EXCEPTION;  
 FARA\_ECHIPE EXCEPTION;  
 PREA\_MULTE\_ECHIPE EXCEPTION;  
BEGIN  
 IF p\_sponsor\_id < 0 THEN  
 RAISE NUMAR\_NEGATIV;  
 END IF;  
 SELECT SPONSOR\_ID  
 INTO v\_sponsor\_id  
 FROM SPONSOR  
 WHERE SPONSOR\_ID = p\_sponsor\_id;  
 OPEN v\_cursor FOR  
 SELECT e.NUME\_ECHIPA  
 FROM ECHIPA e  
 JOIN SPONSORIZEAZA ON e.TEAM\_ID = SPONSORIZEAZA.TEAM\_ID  
 JOIN SPONSOR s ON SPONSORIZEAZA.SPONSOR\_ID = s.SPONSOR\_ID  
 WHERE s.SPONSOR\_ID = p\_sponsor\_id;  
 v\_s := 0;  
 v\_suma := 0;  
 v\_cnt := 0;  
 LOOP  
 FETCH v\_cursor INTO v\_nume\_echipa;  
 EXIT WHEN v\_cursor%NOTFOUND;  
 SELECT *SUM*(SUMA\_SPONSORIZATA)  
 INTO v\_s  
 FROM SPONSORIZEAZA  
 JOIN SPONSOR s ON SPONSORIZEAZA.SPONSOR\_ID = s.SPONSOR\_ID  
 WHERE TEAM\_ID = (SELECT TEAM\_ID FROM ECHIPA WHERE NUME\_ECHIPA = v\_nume\_echipa);  
 IF v\_s > v\_suma THEN  
 v\_suma := v\_s;  
 v\_sol := v\_nume\_echipa;  
 v\_cnt := 1;  
 ELSIF v\_s = v\_suma THEN  
 v\_cnt := v\_cnt + 1;  
 END IF;  
 END LOOP;  
 close v\_cursor;  
 IF v\_sol IS NULL THEN  
 RAISE FARA\_ECHIPE;  
 END IF;  
 IF v\_cnt > 1 THEN  
 RAISE PREA\_MULTE\_ECHIPE;  
 end if;  
 RETURN v\_sol;  
  
 EXCEPTION  
 WHEN NUMAR\_NEGATIV THEN  
 DBMS\_OUTPUT.*PUT\_LINE*('ID-ul sponsorului nu poate fi negativ!');  
 RETURN NULL;  
 WHEN FARA\_ECHIPE THEN  
 DBMS\_OUTPUT.*PUT\_LINE*('Nu exista nicio echipa care sa aiba sponsorul cu ID-ul ' || p\_sponsor\_id);  
 RETURN NULL;  
 WHEN PREA\_MULTE\_ECHIPE THEN  
 DBMS\_OUTPUT.*PUT\_LINE*('Exista mai multe echipe care au acelasi numar de bani din sponsorizari!');  
 RETURN NULL;  
 WHEN NO\_DATA\_FOUND THEN  
 DBMS\_OUTPUT.*PUT\_LINE*('Nu exista sponsor cu acest ID!');  
 RETURN NULL;  
  
END;  
-- apelarea functiei  
BEGIN  
 DBMS\_OUTPUT.*PUT\_LINE*(*get\_team\_with\_most\_money*(1));  
 DBMS\_OUTPUT.*PUT\_LINE*('----------------');  
 DBMS\_OUTPUT.*PUT\_LINE*(*get\_team\_with\_most\_money*(2));  
 DBMS\_OUTPUT.*PUT\_LINE*('----------------');  
 DBMS\_OUTPUT.*PUT\_LINE*(*get\_team\_with\_most\_money*(-3));  
 DBMS\_OUTPUT.*PUT\_LINE*('----------------');  
 DBMS\_OUTPUT.*PUT\_LINE*(*get\_team\_with\_most\_money*(123));  
 DBMS\_OUTPUT.*PUT\_LINE*('----------------');  
 DBMS\_OUTPUT.*PUT\_LINE*(*get\_team\_with\_most\_money*(10));  
END;

****

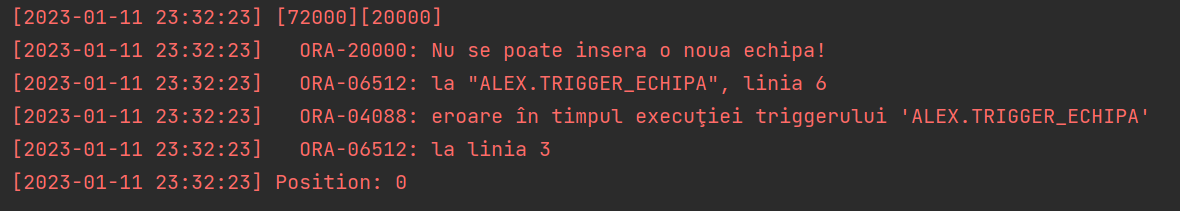
**9. Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat independent de tip procedură care să utilizeze într-o singură comandă SQL 5 dintre tabelele definite. Tratați toate excepțiile care pot apărea, incluzând excepțiile NO\_DATA\_FOUND și TOO\_MANY\_ROWS. Apelați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

-- exercitiul 9  
-- Pentru un jucator al carui prenume este dat, afisati terenul cu cea mai mare capacitate  
-- in care a jucat  
CREATE OR REPLACE PROCEDURE *get\_field\_with\_most\_capacity*(p\_nume\_jucator IN VARCHAR2) AS  
 v\_nume\_teren TEREN.NUME%type;  
 v\_capacitate TEREN.capacitate%type;  
 aux\_capacitate TEREN.capacitate%type;  
 aux\_nume\_teren TEREN.NUME%type;  
 v\_playerid JUCATOR.player\_id%type;  
 v\_init NUMBER := -1;  
 CURSOR c\_teren IS  
 SELECT t.CAPACITATE, t.NUME, JUCATOR.PLAYER\_ID  
 FROM teren t  
 JOIN meci m on m.FIELD\_ID = t.FIELD\_ID  
 JOIN joaca on joaca.MATCH\_ID = m.MATCH\_ID  
 JOIN ECHIPA on ECHIPA.TEAM\_ID = joaca.TEAM\_ID  
 join JUCATOR on JUCATOR.TEAM\_ID = ECHIPA.TEAM\_ID  
 WHERE *UPPER*(JUCATOR.NUME\_PRENUME) LIKE '%' || *UPPER*(p\_nume\_jucator) || '%';  
 INVALID\_INPUT EXCEPTION;  
 TYPE\_MISMATCH EXCEPTION;  
BEGIN  
 IF p\_nume\_jucator IS NULL THEN  
 RAISE INVALID\_INPUT;  
 END IF;  
 IF *regexp\_like*(p\_nume\_jucator, '[0-9]') THEN  
 RAISE TYPE\_MISMATCH;  
 END IF;  
 OPEN c\_teren;  
 v\_capacitate := -1;  
 LOOP  
 FETCH c\_teren INTO aux\_capacitate, aux\_nume\_teren, v\_playerid;  
 EXIT WHEN c\_teren%NOTFOUND;  
 if v\_playerid != v\_init and v\_init != -1 then  
 raise too\_many\_rows;  
 end if;  
 if aux\_capacitate > v\_capacitate then  
 v\_capacitate := aux\_capacitate;  
 v\_nume\_teren := aux\_nume\_teren;  
 end if;  
 v\_init := v\_playerid;  
 end loop;  
 CLOSE c\_teren;  
 IF v\_capacitate = -1 THEN  
 raise no\_data\_found;  
 end if;  
 DBMS\_OUTPUT.*PUT\_LINE*('Terenul cu cea mai mare capacitate in care a jucat ' || p\_nume\_jucator || ' este ' || v\_nume\_teren || ' cu o capacitate de ' || v\_capacitate);  
  
 EXCEPTION  
 WHEN NO\_DATA\_FOUND THEN  
 DBMS\_OUTPUT.*PUT\_LINE*('Nu exista niciun jucator cu numele ' || p\_nume\_jucator);  
 RETURN;  
 WHEN TOO\_MANY\_ROWS THEN  
 DBMS\_OUTPUT.*PUT\_LINE*('Exista mai mult de un jucator cu numele ' || p\_nume\_jucator);  
 RETURN;  
 WHEN TYPE\_MISMATCH THEN  
 DBMS\_OUTPUT.*PUT\_LINE*('Numele jucatorului trebuie sa fie un sir de caractere fara cifre!');  
 RETURN;  
 WHEN INVALID\_INPUT THEN  
 DBMS\_OUTPUT.*PUT\_LINE*('Numele jucatorului nu poate fi NULL!');  
 RETURN;  
 WHEN OTHERS THEN  
 DBMS\_OUTPUT.*PUT\_LINE*('Alta eroare!');  
  
END;  
-- apelarea procedurii  
BEGIN  
 *get\_field\_with\_most\_capacity*('Alex');  
 *get\_field\_with\_most\_capacity*('Andrei');  
 *get\_field\_with\_most\_capacity*('Obama');  
 *get\_field\_with\_most\_capacity*(6);  
 *get\_field\_with\_most\_capacity*('ale32');  
 *get\_field\_with\_most\_capacity*('');  
END;

****

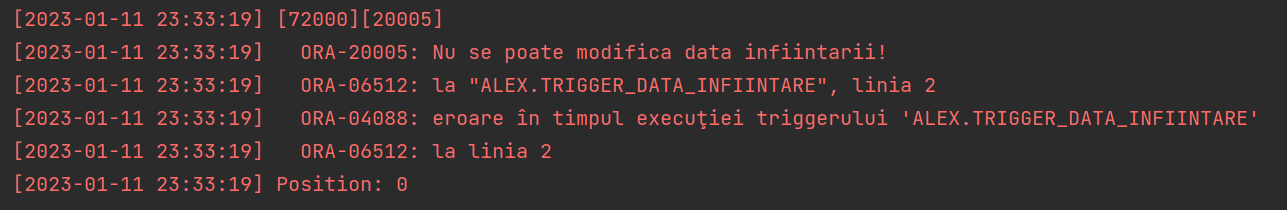
**10. Definiți un trigger de tip LMD la nivel de comandă. Declanșați trigger-ul.**

-- exercitiul 10  
-- Creati un trigger care sa nu permita sa fie mai mult de 16 echipe in turneu  
-- intrucat organizarea nu permite mai multe echipe  
CREATE OR REPLACE TRIGGER trigger\_echipa  
 AFTER INSERT ON ECHIPA  
DECLARE  
 v\_count NUMBER;  
BEGIN  
 SELECT *COUNT*(\*) INTO v\_count FROM ECHIPA;  
 IF v\_count > 16 THEN  
 *RAISE\_APPLICATION\_ERROR*(-20004, 'Nu se poate insera o noua echipa!');  
 END IF;  
END;  
-- declansarea triggerului  
BEGIN  
 FOR i in 1..10 LOOP  
 INSERT INTO ECHIPA(team\_id, nume\_echipa, group\_id) VALUES(30+i, 'test ' || i, 1);  
 end loop;  
 delete from ECHIPA where DATA\_INFIINTARE = '01-01-2022';  
END;

****

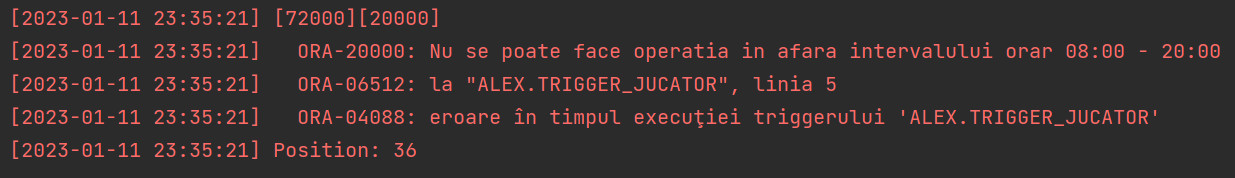
**11. Definiți un trigger de tip LMD la nivel de linie. Declanșați trigger-ul.**

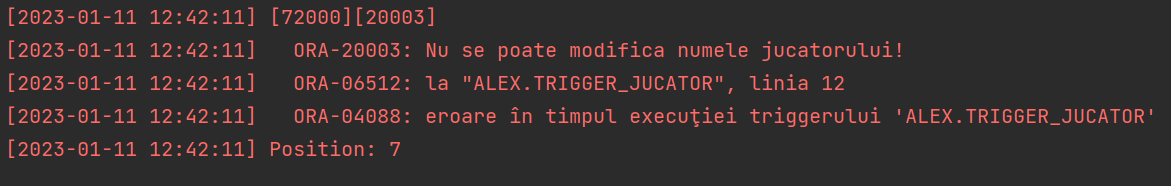
-- exercitiul 11  
-- Creati un trigger care sa nu permita modificarea datei de infiintare a unei echipe  
-- (acesta este un camp ce nu trebuie modificat)  
CREATE OR REPLACE TRIGGER trigger\_data\_infiintare  
 BEFORE UPDATE OF DATA\_INFIINTARE ON ECHIPA  
 FOR EACH ROW  
 WHEN (NEW.DATA\_INFIINTARE != OLD.DATA\_INFIINTARE)  
BEGIN  
 *RAISE\_APPLICATION\_ERROR*(-20005, 'Nu se poate modifica data infiintarii!');  
END;  
-- declansarea triggerului  
BEGIN  
 UPDATE ECHIPA SET DATA\_INFIINTARE = '01-01-1990' WHERE ECHIPA.TEAM\_ID < 3;  
END;

****

**Pentru exercițiile 10 și 11 am făcut și un trigger compus care rezolvă ambele exerciții deodată.**

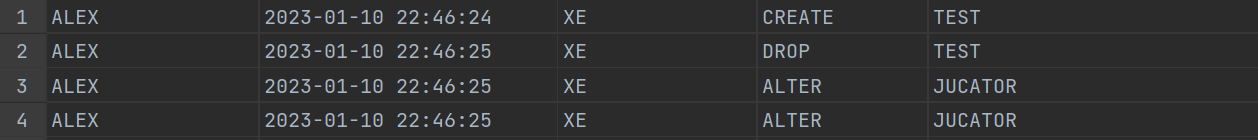
-- exercitiile 10 si 11 combinate  
CREATE OR REPLACE TRIGGER trigger\_jucator  
 FOR INSERT OR UPDATE OR DELETE ON JUCATOR  
 COMPOUND TRIGGER  
 BEFORE STATEMENT IS  
 BEGIN  
 IF *TO\_CHAR*(*SYSDATE*, 'HH24') < '08' OR *TO\_CHAR*(*SYSDATE*, 'HH24') > '20' THEN  
 *RAISE\_APPLICATION\_ERROR*(-20002, 'Nu se poate face operatia in afara intervalului orar 08:00 - 20:00');  
 END IF;  
 END BEFORE STATEMENT;  
 BEFORE EACH ROW IS  
 BEGIN  
 IF UPDATING THEN  
 IF :OLD.NUME\_PRENUME != :NEW.NUME\_PRENUME THEN  
 *RAISE\_APPLICATION\_ERROR*(-20003, 'Nu se poate modifica numele jucatorului!');  
 END IF;  
 END IF;  
 END BEFORE EACH ROW;  
END;  
-- apelarea triggerului  
BEGIN  
 INSERT INTO JUCATOR VALUES(50, 'John Cena', 2, 1, '1990-01-01', 1);  
 UPDATE JUCATOR SET NUME\_PRENUME = 'John Cena' WHERE PLAYER\_ID = 3;  
END;

****

****

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

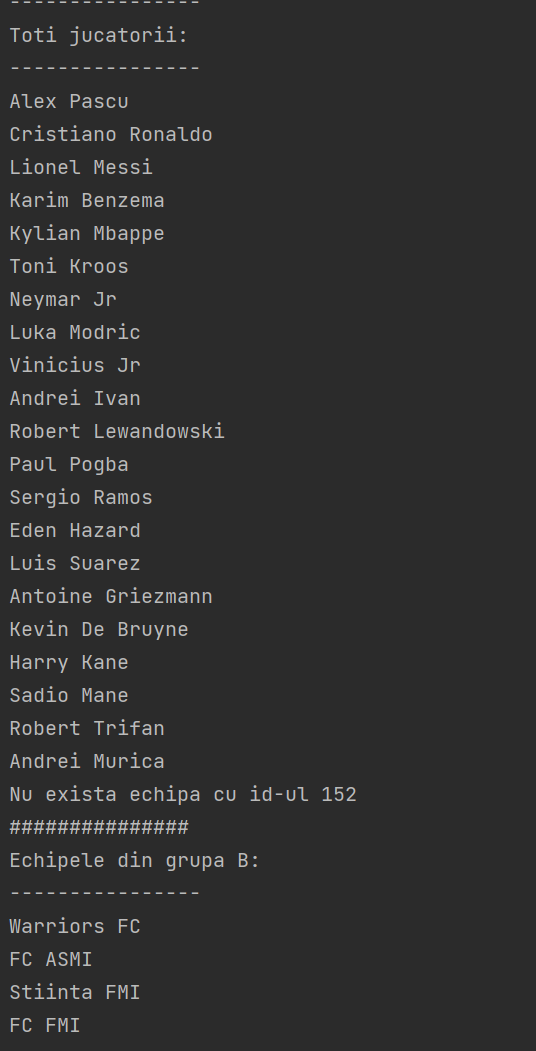
-- exercitiul 12  
-- Creati un trigger care sa permita modificarea schemei doar de utilizatorul alex  
-- si salvati modificarile facute asupra schemei intr-un tabel  
CREATE TABLE log\_history  
(  
 username VARCHAR2(20),  
 log\_date DATE,  
 db\_name VARCHAR2(20),  
 event VARCHAR2(100),  
 obj\_name VARCHAR2(100)  
);  
  
CREATE OR REPLACE TRIGGER trigger\_schema  
 AFTER CREATE OR DROP OR ALTER ON SCHEMA  
BEGIN  
 IF *USER* != 'ALEX' THEN  
 *RAISE\_APPLICATION\_ERROR*(-20910, 'Numai administratorul bazei de date poate realiza operatia!');  
 END IF;  
 INSERT INTO log\_history VALUES(SYS.*LOGIN\_USER*, *SYSDATE*, SYS.*DATABASE\_NAME*, SYS.*SYSEVENT*, SYS.*DICTIONARY\_OBJ\_NAME*);  
END;  
  
-- declansarea triggerului  
CREATE TABLE test (id NUMBER);  
DROP TABLE test;  
ALTER TABLE JUCATOR ADD (test VARCHAR2(20));  
ALTER TABLE JUCATOR DROP COLUMN test;  
-- vizualizarea modificarilor  
SELECT \* FROM log\_history;  
ROLLBACK;

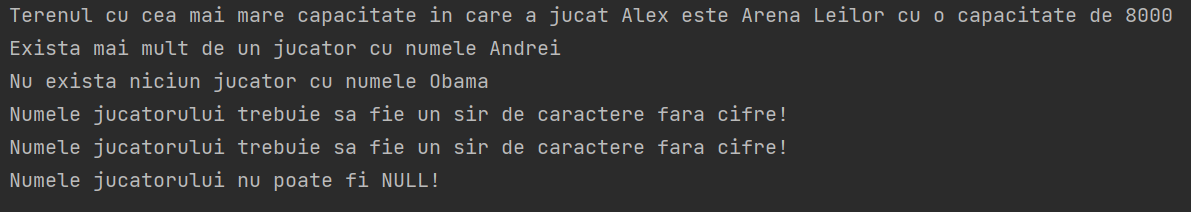
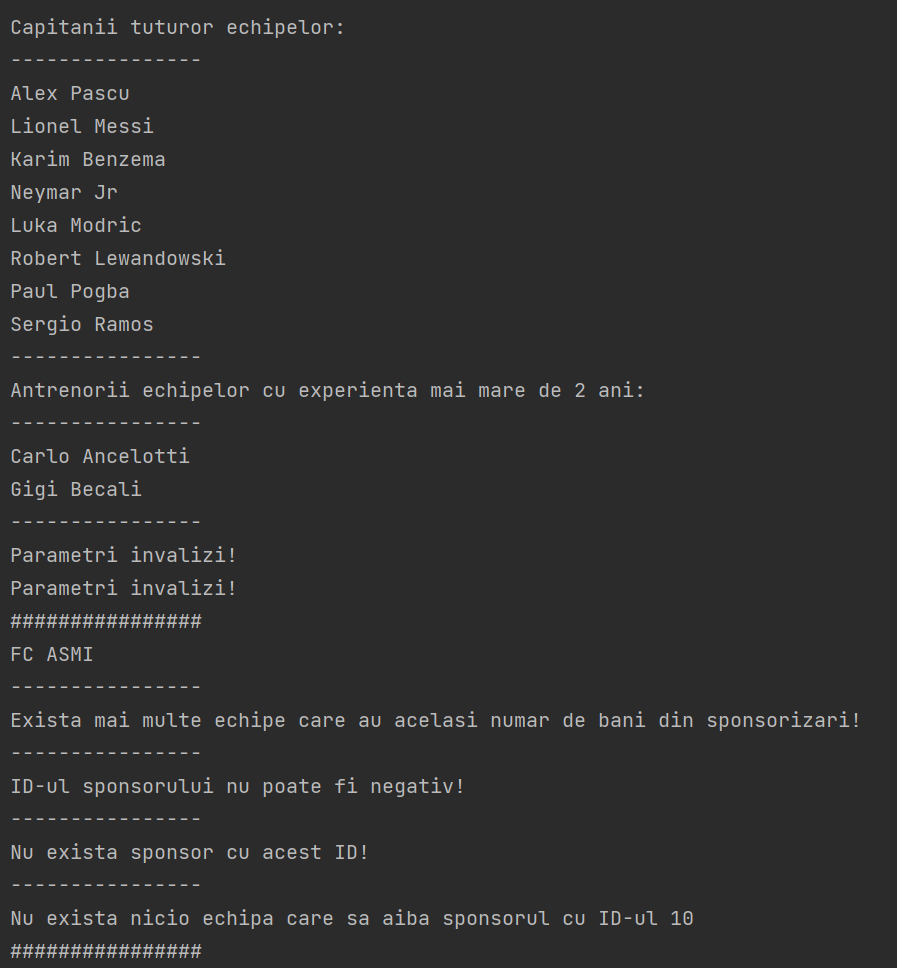
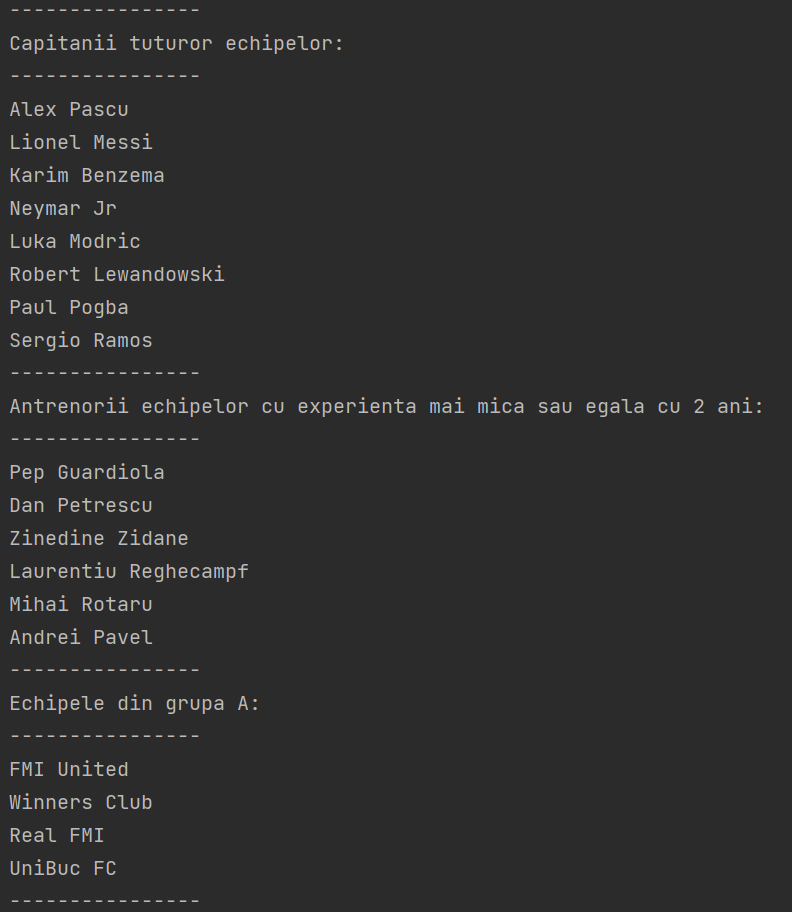
****

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

-- exercitiul 13  
-- Definiti un pachet care sa contina toate obiectele definite in cadrul proiectului  
CREATE OR REPLACE PACKAGE proiect\_AXP AS  
 PROCEDURE *get\_players\_from\_team*(id\_echipa IN NUMBER);  
 PROCEDURE *get\_teams\_captains\_coaches*(p\_grupa IN GRUPA.NUME%type, p\_cursor IN NUMBER);  
 FUNCTION *get\_team\_with\_most\_money*(p\_sponsor\_id IN NUMBER) RETURN VARCHAR2;  
 PROCEDURE *get\_field\_with\_most\_capacity*(p\_nume\_jucator IN VARCHAR2);  
END proiect\_AXP;  
CREATE OR REPLACE PACKAGE BODY proiect\_AXP AS  
 -- exercitiul 6  
 -- Afisati toti jucatorii care participa la turneu, apoi doar jucatorii care  
 -- fac parte din echipa cu un id dat  
 PROCEDURE *get\_players\_from\_team*(id\_echipa IN NUMBER)  
IS  
 TYPE jucatori is varray(100) of jucator%rowtype;  
 v\_jucatori jucatori;  
 v\_jucator jucator%rowtype;  
 TYPE toti\_jucatorii is table of jucator%rowtype;  
 v\_toti\_jucatorii toti\_jucatorii;  
 ECHIPA\_NEGASITA EXCEPTION;  
BEGIN  
 SELECT \* BULK COLLECT INTO v\_toti\_jucatorii from JUCATOR;  
 DBMS\_OUTPUT.*PUT\_LINE*('Toti jucatorii:');  
 DBMS\_OUTPUT.*PUT\_LINE*('----------------');  
 FOR i in 1..v\_toti\_jucatorii.COUNT loop  
 v\_jucator := v\_toti\_jucatorii(i);  
 DBMS\_OUTPUT.*PUT\_LINE*(v\_jucator.NUME\_PRENUME);  
 end loop;  
 SELECT \* bulk collect into v\_jucatori FROM JUCATOR WHERE TEAM\_ID = id\_echipa;  
 IF v\_jucatori.COUNT = 0 THEN  
 RAISE ECHIPA\_NEGASITA;  
 end if;  
 DBMS\_OUTPUT.*PUT\_LINE*('----------------');  
 DBMS\_OUTPUT.*PUT\_LINE*('Jucatorii din echipa cu id-ul ' || id\_echipa || ':');  
 DBMS\_OUTPUT.*PUT\_LINE*('----------------');  
 FOR i IN 1 .. v\_jucatori.COUNT LOOP  
 v\_jucator := v\_jucatori(i);  
 DBMS\_OUTPUT.*PUT\_LINE*(v\_jucator.NUME\_PRENUME);  
 END LOOP;  
 DBMS\_OUTPUT.*PUT\_LINE*('----------------');  
  
 EXCEPTION  
 WHEN ECHIPA\_NEGASITA THEN  
 DBMS\_OUTPUT.*PUT\_LINE*('Nu exista echipa cu id-ul ' || id\_echipa);  
  
END;  
  
 -- exercitiul 7  
 -- Afisati echipele dintr-o grupa data, apoi capitanii tuturor echipelor  
 -- apoi afisati antrenorii echipelor din grupa respectiva cu mai mult/mai putin de 2 ani experienta  
 PROCEDURE *get\_teams\_captains\_coaches*(p\_grupa IN GRUPA.NUME%type, p\_cursor IN NUMBER)  
AS  
 cursor c\_teams (p\_grupa IN GRUPA.NUME%type) is  
 select \* from echipa e  
 join grupa g on e.GROUP\_ID = g.GROUP\_ID  
 where g.nume = p\_grupa;  
 cursor c\_captains is  
 select \* from jucator j  
 join capitan c on c.PLAYER\_ID = j.PLAYER\_ID;  
  
 v\_echipa c\_teams%rowtype;  
 v\_capitan c\_captains%rowtype;  
 TYPE tip\_cursor IS REF CURSOR RETURN antrenor%ROWTYPE;  
 v\_coach\_cursor tip\_cursor;  
  
 v\_coach ANTRENOR%rowtype;  
 INVALID\_INPUT EXCEPTION;  
BEGIN  
 IF p\_cursor > 2 OR p\_cursor < 1 THEN  
 RAISE INVALID\_INPUT;  
 end if;  
 DBMS\_OUTPUT.*PUT\_LINE*('Echipele din grupa ' || p\_grupa || ':');  
 DBMS\_OUTPUT.*PUT\_LINE*('----------------');  
 OPEN c\_teams(p\_grupa);  
 LOOP  
 FETCH c\_teams INTO v\_echipa;  
 EXIT WHEN c\_teams%NOTFOUND;  
 DBMS\_OUTPUT.*PUT\_LINE*(v\_echipa.NUME\_ECHIPA);  
 END LOOP;  
 CLOSE c\_teams;  
 DBMS\_OUTPUT.*PUT\_LINE*('----------------');  
 DBMS\_OUTPUT.*PUT\_LINE*('Capitanii tuturor echipelor:');  
 DBMS\_OUTPUT.*PUT\_LINE*('----------------');  
 OPEN c\_captains;  
 LOOP  
 FETCH c\_captains INTO v\_capitan;  
 EXIT WHEN c\_captains%NOTFOUND;  
 DBMS\_OUTPUT.*PUT\_LINE*(v\_capitan.nume\_prenume);  
 END LOOP;  
 CLOSE c\_captains;  
  
 IF p\_cursor = 1 THEN  
 OPEN v\_coach\_cursor FOR  
 SELECT \* FROM ANTRENOR WHERE EXPERIENTA > 2;  
 ELSIF p\_cursor = 2 THEN  
 OPEN v\_coach\_cursor FOR  
 SELECT \* FROM ANTRENOR WHERE EXPERIENTA <= 2;  
 END IF;  
  
 DBMS\_OUTPUT.*PUT\_LINE*('----------------');  
 if p\_cursor = 1 then  
 DBMS\_OUTPUT.*PUT\_LINE*('Antrenorii echipelor cu experienta mai mare de 2 ani:');  
 else  
 DBMS\_OUTPUT.*PUT\_LINE*('Antrenorii echipelor cu experienta mai mica sau egala cu 2 ani:');  
 end if;  
 DBMS\_OUTPUT.*PUT\_LINE*('----------------');  
 LOOP  
 FETCH v\_coach\_cursor INTO v\_coach;  
 EXIT WHEN v\_coach\_cursor%NOTFOUND;  
 DBMS\_OUTPUT.*PUT\_LINE*(v\_coach.nume);  
 END LOOP;  
 DBMS\_OUTPUT.*PUT\_LINE*('----------------');  
  
 EXCEPTION  
 WHEN INVALID\_INPUT THEN  
 DBMS\_OUTPUT.*PUT\_LINE*('Parametri invalizi!');  
  
END;  
  
 -- exercitiul 8  
 -- Afisati echipa care primeste cei mai multi bani din sponsorizari  
 -- avand un sponsor dat  
 FUNCTION *get\_team\_with\_most\_money*(p\_sponsor\_id IN NUMBER) RETURN VARCHAR2 AS  
 v\_nume\_echipa ECHIPA.nume\_echipa%type;  
 v\_suma NUMBER;  
 v\_s NUMBER;  
 v\_cnt NUMBER;  
 v\_sol ECHIPA.nume\_echipa%type;  
 v\_sponsor\_id NUMBER;  
 v\_cursor SYS\_REFCURSOR;  
 NUMAR\_NEGATIV EXCEPTION;  
 FARA\_ECHIPE EXCEPTION;  
 PREA\_MULTE\_ECHIPE EXCEPTION;  
BEGIN  
 IF p\_sponsor\_id < 0 THEN  
 RAISE NUMAR\_NEGATIV;  
 END IF;  
 SELECT SPONSOR\_ID  
 INTO v\_sponsor\_id  
 FROM SPONSOR  
 WHERE SPONSOR\_ID = p\_sponsor\_id;  
 OPEN v\_cursor FOR  
 SELECT e.NUME\_ECHIPA  
 FROM ECHIPA e  
 JOIN SPONSORIZEAZA ON e.TEAM\_ID = SPONSORIZEAZA.TEAM\_ID  
 JOIN SPONSOR s ON SPONSORIZEAZA.SPONSOR\_ID = s.SPONSOR\_ID  
 WHERE s.SPONSOR\_ID = p\_sponsor\_id;  
 v\_s := 0;  
 v\_suma := 0;  
 v\_cnt := 0;  
 LOOP  
 FETCH v\_cursor INTO v\_nume\_echipa;  
 EXIT WHEN v\_cursor%NOTFOUND;  
 SELECT *SUM*(SUMA\_SPONSORIZATA)  
 INTO v\_s  
 FROM SPONSORIZEAZA  
 JOIN SPONSOR s ON SPONSORIZEAZA.SPONSOR\_ID = s.SPONSOR\_ID  
 WHERE TEAM\_ID = (SELECT TEAM\_ID FROM ECHIPA WHERE NUME\_ECHIPA = v\_nume\_echipa);  
 IF v\_s > v\_suma THEN  
 v\_suma := v\_s;  
 v\_sol := v\_nume\_echipa;  
 v\_cnt := 1;  
 ELSIF v\_s = v\_suma THEN  
 v\_cnt := v\_cnt + 1;  
 END IF;  
 END LOOP;  
 close v\_cursor;  
 IF v\_sol IS NULL THEN  
 RAISE FARA\_ECHIPE;  
 END IF;  
 IF v\_cnt > 1 THEN  
 RAISE PREA\_MULTE\_ECHIPE;  
 end if;  
 RETURN v\_sol;  
  
 EXCEPTION  
 WHEN NUMAR\_NEGATIV THEN  
 DBMS\_OUTPUT.*PUT\_LINE*('ID-ul sponsorului nu poate fi negativ!');  
 RETURN NULL;  
 WHEN FARA\_ECHIPE THEN  
 DBMS\_OUTPUT.*PUT\_LINE*('Nu exista nicio echipa care sa aiba sponsorul cu ID-ul ' || p\_sponsor\_id);  
 RETURN NULL;  
 WHEN PREA\_MULTE\_ECHIPE THEN  
 DBMS\_OUTPUT.*PUT\_LINE*('Exista mai multe echipe care au acelasi numar de bani din sponsorizari!');  
 RETURN NULL;  
 WHEN NO\_DATA\_FOUND THEN  
 DBMS\_OUTPUT.*PUT\_LINE*('Nu exista sponsor cu acest ID!');  
 RETURN NULL;  
  
END;  
  
 -- exercitiul 9  
 -- Pentru un jucator al carui prenume este dat, afisati terenul cu cea mai mare capacitate  
 -- in care a jucat  
 PROCEDURE *get\_field\_with\_most\_capacity*(p\_nume\_jucator IN VARCHAR2) AS  
 v\_nume\_teren TEREN.NUME%type;  
 v\_capacitate TEREN.capacitate%type;  
 aux\_capacitate TEREN.capacitate%type;  
 aux\_nume\_teren TEREN.NUME%type;  
 v\_playerid JUCATOR.player\_id%type;  
 v\_init NUMBER := -1;  
 CURSOR c\_teren IS  
 SELECT t.CAPACITATE, t.NUME, JUCATOR.PLAYER\_ID  
 FROM teren t  
 JOIN meci m on m.FIELD\_ID = t.FIELD\_ID  
 JOIN joaca on joaca.MATCH\_ID = m.MATCH\_ID  
 JOIN ECHIPA on ECHIPA.TEAM\_ID = joaca.TEAM\_ID  
 join JUCATOR on JUCATOR.TEAM\_ID = ECHIPA.TEAM\_ID  
 WHERE *UPPER*(JUCATOR.NUME\_PRENUME) LIKE '%' || *UPPER*(p\_nume\_jucator) || '%';  
 INVALID\_INPUT EXCEPTION;  
 TYPE\_MISMATCH EXCEPTION;  
BEGIN  
 IF p\_nume\_jucator IS NULL THEN  
 RAISE INVALID\_INPUT;  
 END IF;  
 IF *regexp\_like*(p\_nume\_jucator, '[0-9]') THEN  
 RAISE TYPE\_MISMATCH;  
 END IF;  
 OPEN c\_teren;  
 v\_capacitate := -1;  
 LOOP  
 FETCH c\_teren INTO aux\_capacitate, aux\_nume\_teren, v\_playerid;  
 EXIT WHEN c\_teren%NOTFOUND;  
 if v\_playerid != v\_init and v\_init != -1 then  
 raise too\_many\_rows;  
 end if;  
 if aux\_capacitate > v\_capacitate then  
 v\_capacitate := aux\_capacitate;  
 v\_nume\_teren := aux\_nume\_teren;  
 end if;  
 v\_init := v\_playerid;  
 end loop;  
 CLOSE c\_teren;  
 IF v\_capacitate = -1 THEN  
 raise no\_data\_found;  
 end if;  
 DBMS\_OUTPUT.*PUT\_LINE*('Terenul cu cea mai mare capacitate in care a jucat ' || p\_nume\_jucator || ' este ' || v\_nume\_teren || ' cu o capacitate de ' || v\_capacitate);  
  
 EXCEPTION  
 WHEN NO\_DATA\_FOUND THEN  
 DBMS\_OUTPUT.*PUT\_LINE*('Nu exista niciun jucator cu numele ' || p\_nume\_jucator);  
 RETURN;  
 WHEN TOO\_MANY\_ROWS THEN  
 DBMS\_OUTPUT.*PUT\_LINE*('Exista mai mult de un jucator cu numele ' || p\_nume\_jucator);  
 RETURN;  
 WHEN TYPE\_MISMATCH THEN  
 DBMS\_OUTPUT.*PUT\_LINE*('Numele jucatorului trebuie sa fie un sir de caractere fara cifre!');  
 RETURN;  
 WHEN INVALID\_INPUT THEN  
 DBMS\_OUTPUT.*PUT\_LINE*('Numele jucatorului nu poate fi NULL!');  
 RETURN;  
 WHEN OTHERS THEN  
 DBMS\_OUTPUT.*PUT\_LINE*('Alta eroare!');  
  
END;  
END proiect\_AXP;  
  
-- testare pachet  
BEGIN  
 proiect\_axp.*get\_players\_from\_team*(1);  
 proiect\_axp.*get\_players\_from\_team*(152);  
 DBMS\_OUTPUT.*PUT\_LINE*('###############');  
 proiect\_axp.*get\_teams\_captains\_coaches*('B', 2);  
 proiect\_axp.*get\_teams\_captains\_coaches*('A', 1);  
 proiect\_axp.*get\_teams\_captains\_coaches*('A', 3);  
 proiect\_axp.*get\_teams\_captains\_coaches*('C', -3);  
 DBMS\_OUTPUT.*PUT\_LINE*('################');  
 DBMS\_OUTPUT.*PUT\_LINE*(proiect\_axp.*get\_team\_with\_most\_money*(1));  
 DBMS\_OUTPUT.*PUT\_LINE*('----------------');  
 DBMS\_OUTPUT.*PUT\_LINE*(proiect\_axp.*get\_team\_with\_most\_money*(2));  
 DBMS\_OUTPUT.*PUT\_LINE*('----------------');  
 DBMS\_OUTPUT.*PUT\_LINE*(proiect\_axp.*get\_team\_with\_most\_money*(-3));  
 DBMS\_OUTPUT.*PUT\_LINE*('----------------');  
 DBMS\_OUTPUT.*PUT\_LINE*(proiect\_axp.*get\_team\_with\_most\_money*(123));  
 DBMS\_OUTPUT.*PUT\_LINE*('----------------');  
 DBMS\_OUTPUT.*PUT\_LINE*(proiect\_axp.*get\_team\_with\_most\_money*(10));  
 DBMS\_OUTPUT.*PUT\_LINE*('################');  
 proiect\_axp.*get\_field\_with\_most\_capacity*('Alex');  
 proiect\_axp.*get\_field\_with\_most\_capacity*('Andrei');  
 proiect\_axp.*get\_field\_with\_most\_capacity*('Obama');  
 proiect\_axp.*get\_field\_with\_most\_capacity*(6);  
 proiect\_axp.*get\_field\_with\_most\_capacity*('ale32');  
 proiect\_axp.*get\_field\_with\_most\_capacity*('');  
end;

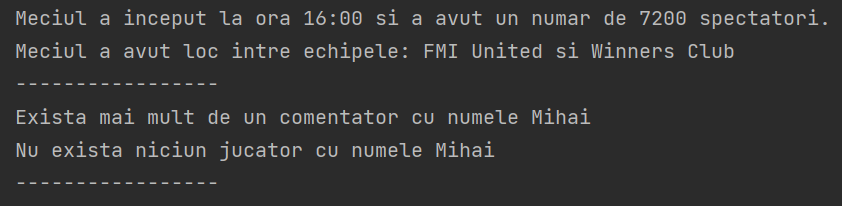
****

****

****

**14. Definiți un pachet care să includă tipuri de date complexe și obiecte necesare unui flux de acțiuni integrate, specifice bazei de date definite (minim 2 tipuri de date, minim 2 funcții, minim 2 proceduri).**

-- exercitiul 14  
-- Definiti un pachet care sa includa tipuri de date complexe si obiecte necesare  
-- pentru actiuni integrate  
-- Pachetul contine functii si proceduri care permit, fiind date ca parametri  
-- numele unui arbitru, comentator si jucator, afisarea tuturor meciurilor in care  
-- au participat, respectiv au comentat, respectiv au jucat persoanele respective  
-- si a echipelor care au jucat in acele meciuri  
CREATE OR REPLACE PACKAGE proiect\_ex14 AS  
 TYPE tip\_meci IS TABLE OF MECI.match\_id%TYPE;  
 TYPE tip\_echipa is varray(2) of ECHIPA%ROWTYPE;  
  
 FUNCTION *get\_referee*(p\_nume\_arbitru IN VARCHAR2) RETURN ARBITRU.referee\_id%type;  
 FUNCTION *get\_commentator*(p\_nume\_comentator IN VARCHAR2) RETURN COMENTATOR.commentator\_id%type;  
 FUNCTION *get\_player*(p\_nume\_jucator IN VARCHAR2) RETURN JUCATOR.player\_id%type;  
 PROCEDURE *get\_matches*(p\_nume\_arbitru IN VARCHAR2, p\_nume\_comentator IN VARCHAR2, p\_nume\_jucator IN VARCHAR2);  
 PROCEDURE *print\_match\_and\_teams*(p\_match\_id IN NUMBER, v\_echipe IN tip\_echipa);  
END proiect\_ex14;  
CREATE OR REPLACE PACKAGE BODY proiect\_ex14 AS  
 PROCEDURE *print\_match\_and\_teams*(p\_match\_id IN NUMBER, v\_echipe IN tip\_echipa) AS  
 v\_meci MECI%rowtype;  
 BEGIN  
 SELECT \* INTO v\_meci FROM MECI WHERE MATCH\_ID = p\_match\_id;  
  
 DBMS\_OUTPUT.*PUT\_LINE*('Meciul a inceput la ora ' || v\_meci.ORA\_INCEPUT || ' si a avut un numar de ' || v\_meci.SPECTATORI || ' spectatori.');  
  
 DBMS\_OUTPUT.*PUT\_LINE*('Meciul a avut loc intre echipele: ' || v\_echipe(1).NUME\_ECHIPA || ' si ' || v\_echipe(2).NUME\_ECHIPA);  
 END *print\_match\_and\_teams*;  
  
 FUNCTION *get\_referee*(p\_nume\_arbitru IN VARCHAR2) RETURN ARBITRU.referee\_id%type AS  
 v\_id ARBITRU.referee\_id%type;  
 cnt NUMBER;  
 BEGIN  
 SELECT *COUNT*(REFEREE\_ID) INTO cnt FROM ARBITRU WHERE *UPPER*(NUME) LIKE '%' || *UPPER*(p\_nume\_arbitru) || '%';  
 IF cnt = 0 THEN  
 raise no\_data\_found;  
 ELSIF cnt > 1 THEN  
 raise too\_many\_rows;  
 ELSE  
 SELECT REFEREE\_ID INTO v\_id FROM ARBITRU WHERE *UPPER*(NUME) LIKE '%' || *UPPER*(p\_nume\_arbitru) || '%';  
 END IF;  
  
 RETURN v\_id;  
  
 EXCEPTION  
 WHEN NO\_DATA\_FOUND THEN  
 DBMS\_OUTPUT.*PUT\_LINE*('Nu exista niciun arbitru cu numele ' || p\_nume\_arbitru);  
 RETURN -1;  
 WHEN TOO\_MANY\_ROWS THEN  
 DBMS\_OUTPUT.*PUT\_LINE*('Exista mai mult de un arbitru cu numele ' || p\_nume\_arbitru);  
 RETURN -1;  
 WHEN OTHERS THEN  
 DBMS\_OUTPUT.*PUT\_LINE*('Alta eroare!');  
 RETURN -1;  
  
 end *get\_referee*;  
  
 FUNCTION *get\_commentator*(p\_nume\_comentator IN VARCHAR2) RETURN COMENTATOR.commentator\_id%type AS  
 v\_id COMENTATOR.commentator\_id%type;  
 cnt NUMBER;  
 BEGIN  
 SELECT *COUNT*(COMMENTATOR\_ID) INTO cnt FROM COMENTATOR WHERE *UPPER*(NUME) LIKE '%' || *UPPER*(p\_nume\_comentator) || '%';  
 IF cnt = 0 THEN  
 raise no\_data\_found;  
 ELSIF cnt > 1 THEN  
 raise too\_many\_rows;  
 ELSE  
 SELECT COMMENTATOR\_ID INTO v\_id FROM COMENTATOR WHERE *UPPER*(NUME) LIKE '%' || *UPPER*(p\_nume\_comentator) || '%';  
 END IF;  
  
 RETURN v\_id;  
  
 EXCEPTION  
 WHEN NO\_DATA\_FOUND THEN  
 DBMS\_OUTPUT.*PUT\_LINE*('Nu exista niciun comentator cu numele ' || p\_nume\_comentator);  
 RETURN -1;  
 WHEN TOO\_MANY\_ROWS THEN  
 DBMS\_OUTPUT.*PUT\_LINE*('Exista mai mult de un comentator cu numele ' || p\_nume\_comentator);  
 RETURN -1;  
 WHEN OTHERS THEN  
 DBMS\_OUTPUT.*PUT\_LINE*('Alta eroare!');  
 RETURN -1;  
 end *get\_commentator*;  
  
 FUNCTION *get\_player*(p\_nume\_jucator IN VARCHAR2) RETURN JUCATOR.player\_id%type AS  
 v\_id JUCATOR.player\_id%type;  
 cnt NUMBER;  
 BEGIN  
 SELECT *COUNT*(PLAYER\_ID) INTO cnt FROM JUCATOR WHERE *UPPER*(NUME\_PRENUME) LIKE '%' || *UPPER*(p\_nume\_jucator) || '%';  
 IF cnt = 0 THEN  
 raise no\_data\_found;  
 ELSIF cnt > 1 THEN  
 raise too\_many\_rows;  
 ELSE  
 SELECT PLAYER\_ID INTO v\_id FROM JUCATOR WHERE *UPPER*(NUME\_PRENUME) LIKE '%' || *UPPER*(p\_nume\_jucator) || '%';  
 END IF;  
  
 RETURN v\_id;  
  
 EXCEPTION  
 WHEN NO\_DATA\_FOUND THEN  
 DBMS\_OUTPUT.*PUT\_LINE*('Nu exista niciun jucator cu numele ' || p\_nume\_jucator);  
 RETURN -1;  
 WHEN TOO\_MANY\_ROWS THEN  
 DBMS\_OUTPUT.*PUT\_LINE*('Exista mai mult de un jucator cu numele ' || p\_nume\_jucator);  
 RETURN -1;  
 WHEN OTHERS THEN  
 DBMS\_OUTPUT.*PUT\_LINE*('Alta eroare!');  
 RETURN -1;  
 end *get\_player*;  
  
 PROCEDURE *get\_matches*(p\_nume\_arbitru IN VARCHAR2, p\_nume\_comentator IN VARCHAR2, p\_nume\_jucator IN VARCHAR2) AS  
 v\_id\_arbitru ARBITRU.referee\_id%type;  
 v\_id\_comentator COMENTATOR.commentator\_id%type;  
 v\_id\_jucator JUCATOR.player\_id%type;  
 v\_id\_echipa ECHIPA.team\_id%type;  
 v\_meciuri tip\_meci;  
 v\_echipe tip\_echipa;  
 i NUMBER;  
 BEGIN  
 v\_id\_arbitru := *get\_referee*(p\_nume\_arbitru);  
 v\_id\_comentator := *get\_commentator*(p\_nume\_comentator);  
 v\_id\_jucator := *get\_player*(p\_nume\_jucator);  
  
 IF v\_id\_arbitru = -1 OR v\_id\_comentator = -1 OR v\_id\_jucator = -1 THEN  
 RETURN;  
 END IF;  
  
 SELECT ECHIPA.TEAM\_ID INTO v\_id\_echipa  
 FROM ECHIPA  
 JOIN JUCATOR ON ECHIPA.TEAM\_ID = JUCATOR.TEAM\_ID  
 WHERE JUCATOR.PLAYER\_ID = v\_id\_jucator;  
  
 IF v\_id\_echipa IS NULL THEN  
 DBMS\_OUTPUT.*PUT\_LINE*('Jucatorul ' || p\_nume\_jucator || ' nu face parte din nici o echipa!');  
 RETURN;  
 END IF;  
  
 SELECT J.MATCH\_ID  
 BULK COLLECT INTO v\_meciuri  
 FROM JOACA J  
 JOIN MECI on MECI.MATCH\_ID = J.MATCH\_ID  
 WHERE REFEREE\_ID = v\_id\_arbitru AND COMMENTATOR\_ID = v\_id\_comentator  
 AND J.TEAM\_ID = v\_id\_echipa;  
  
 FOR i IN 1..v\_meciuri.COUNT LOOP  
 SELECT E.\*  
 BULK COLLECT INTO v\_echipe  
 FROM JOACA J  
 JOIN ECHIPA E ON E.TEAM\_ID = J.TEAM\_ID  
 WHERE J.MATCH\_ID = v\_meciuri(i);  
  
 *print\_match\_and\_teams*(v\_meciuri(i), v\_echipe);  
  
 END LOOP;  
 END *get\_matches*;  
  
END;  
/  
-- testare pachet  
BEGIN  
 proiect\_ex14.*get\_matches*('Cristian','Alexandru','Alex');  
 DBMS\_OUTPUT.*PUT\_LINE*('-----------------');  
 proiect\_ex14.*get\_matches*('Mihai', 'Mihai', 'Mihai');  
 DBMS\_OUTPUT.*PUT\_LINE*('-----------------');  
END;

****