Facultatea de Matematică şi Informatică
Universitatea din Bucureşti

# Gestionarea unei săli de sport

Nistor Gheorghe

Grupa 242

Prof. coordonatori: Natalia Gabriela Ozunu
Eduard Gabriel Szmeteancă

# Exerciții

1. Prezentați pe scurt baza de date (utilitatea ei)

2. Diagrama Entitate-Relație (ERD)

3. Diagrama Conceptuală.

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

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

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

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. Apelați subprogramul

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.

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.

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

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

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

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

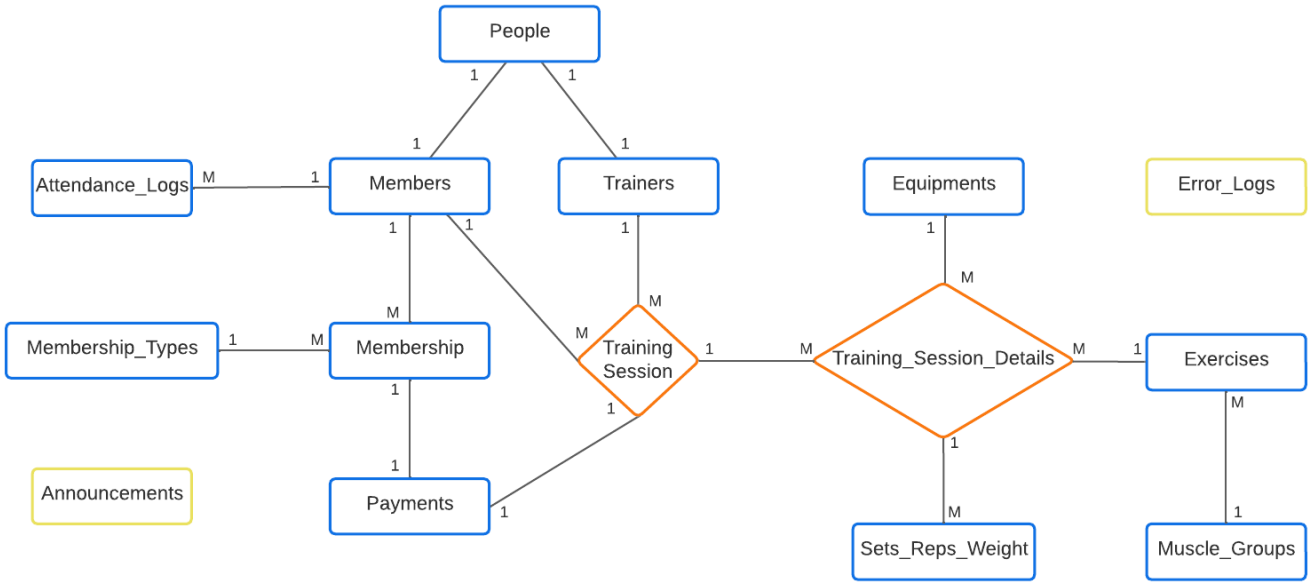
14. Definiți un pachet care să includă tipuri de date complexe și obiecte necesare unui flux de acțiuni integrate, specifice bazei de date definite (minim 2 tipuri de date, minim 2 funcții, minim 2 proceduri).

15. APEX_MAIL

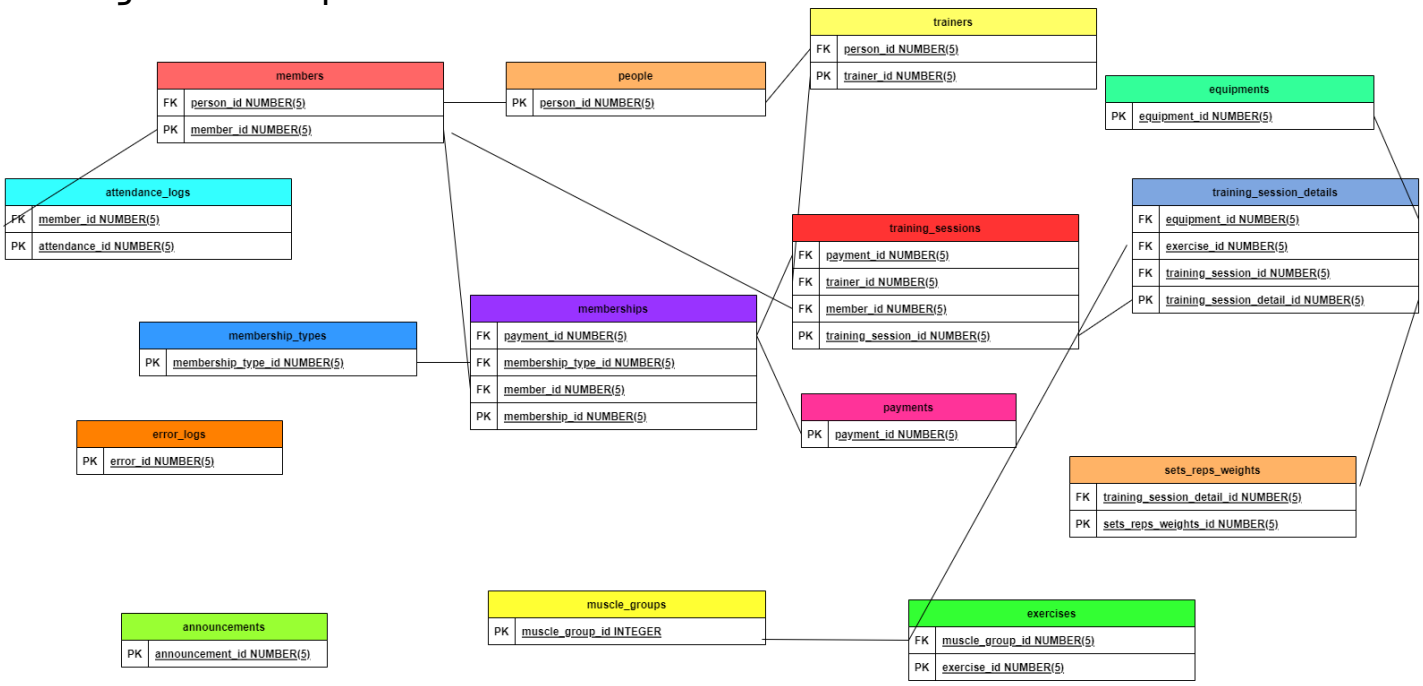
1. Prezentați pe scurt baza de date (utilitatea ei)

Baza de date va fi folosită pentru a gestiona o sală de sport independentă, respectiv o sală care nu face parte din nici o franciză și care îi aparține unui patron local.
Aceasta bază de date va fi un instrument util pentru stocarea și gestionarea informațiilor despre membrii sălii, abonamente, plăți, echipamentul din sala de fitness, instructori, sesiuni de antrenament între un instructor și un client, cât și detalii despre aceste antrenamente: exercițiile care au fost lucrate și informații despre fiecare set de lucru: număr de repetări și greutatea folosită.

2. Diagrama Entitate-Relație (ERD)



## 3. Diagrama Conceptuală.




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

```
CREATE TABLE people
  (
     person_id  NUMBER(5) PRIMARY KEY,
     first_name VARCHAR(50) NOT NULL,
     last_name  VARCHAR(50) NOT NULL,
     gender     VARCHAR(1) NOT NULL,
     birth_date DATE NOT NULL,
     email      VARCHAR(255) NOT NULL UNIQUE
  );

CREATE TABLE members
  (
     member_id       NUMBER(5) PRIMARY KEY,
     person_id       NUMBER(5),
     student         NUMBER(1) DEFAULT(0),
     starting_weight NUMBER(5, 2),
     height          NUMBER(5, 2),
     FOREIGN KEY (person_id) REFERENCES people (person_id)
  );
```

```sql
CREATE TABLE trainers
  (
     trainer_id   NUMBER(5) PRIMARY KEY,
     person_id    NUMBER(5),
     nutritionist NUMBER(1) DEFAULT(0),
     hire_date    DATE NOT NULL,
     salary       NUMBER(6, 2) NOT NULL,
     FOREIGN KEY (person_id) REFERENCES people (person_id)
  );

CREATE TABLE attendance_logs
  (
     attendance_id   NUMBER(5) PRIMARY KEY,
     member_id       NUMBER(5),
     attendance_date DATE DEFAULT(SYSDATE),
     FOREIGN KEY (member_id) REFERENCES members (member_id)
  );

CREATE TABLE memberships
  (
     membership_id      NUMBER(5) PRIMARY KEY,
     member_id          NUMBER(5),
     membership_type_id NUMBER(5),
     payment_id         NUMBER(5),
     start_date         DATE DEFAULT(SYSDATE),
     end_date           DATE NOT NULL,
     FOREIGN KEY (member_id) REFERENCES members (member_id),
     FOREIGN KEY (membership_type_id) REFERENCES membership_types (
     membership_type_id),
     FOREIGN KEY (payment_id) REFERENCES payments (payment_id)
  );

CREATE TABLE membership_types
  (
     membership_type_id NUMBER(5) PRIMARY KEY,
     membership_name    VARCHAR(50) NOT NULL,
     student            NUMBER(1) DEFAULT(0)
  );

CREATE TABLE payments
  (
     payment_id   NUMBER(5) PRIMARY KEY,
     payment_date DATE,
     amount       NUMBER(6, 2) NOT NULL,
     payment_type VARCHAR2(10)
  );
```

```sql
CREATE TABLE training_sessions
  (
     training_session_id   NUMBER(5) PRIMARY KEY,
     member_id             NUMBER(5),
     trainer_id            NUMBER(5),
     payment_id            NUMBER(5),
     training_session_date DATE NOT NULL,
     weight                NUMBER(5, 2),
     FOREIGN KEY (member_id) REFERENCES members (member_id),
     FOREIGN KEY (trainer_id) REFERENCES trainers (trainer_id),
     FOREIGN KEY (payment_id) REFERENCES payments (payment_id)
  );

CREATE TABLE training_session_details
  (
     training_session_detail_id NUMBER(5) PRIMARY KEY,
     training_session_id        NUMBER(5),
     exercise_id                NUMBER(5),
     equipment_id               NUMBER(5),
     FOREIGN KEY (training_session_id) REFERENCES training_sessions (
     training_session_id),
     FOREIGN KEY (exercise_id) REFERENCES exercises (exercise_id),
     FOREIGN KEY (equipment_id) REFERENCES equipments (equipment_id)
  );

CREATE TABLE equipments
  (
     equipment_id   NUMBER(5) PRIMARY KEY,
     equipment_name VARCHAR(50) NOT NULL
  );

CREATE TABLE sets_reps_weights
  (
     sets_reps_weights_id       NUMBER(5) PRIMARY KEY,
     training_session_detail_id NUMBER(5),
     current_set                NUMBER(1),
     reps                       NUMBER(2),
     weight                     NUMBER(3),
     FOREIGN KEY (training_session_detail_id) REFERENCES
     training_session_details (training_session_detail_id)
  );

CREATE TABLE exercises
  (
     exercise_id     NUMBER(5) PRIMARY KEY,
     muscle_group_id NUMBER(5),
     exercise_name   VARCHAR(50) NOT NULL,
     FOREIGN KEY (muscle_group_id) REFERENCES muscle_groups (muscle_group_id
)
```

```
  );

CREATE TABLE muscle_groups
  (
     muscle_group_id   INTEGER PRIMARY KEY,
     muscle_group_name VARCHAR(50) NOT NULL
  );

CREATE TABLE announcements
  (
     announcement_id NUMBER(5) PRIMARY KEY,
     title           VARCHAR(25),
     message         VARCHAR(1000),
     category        VARCHAR2(30),
     send_date       DATE,
     sent            NUMBER(1) DEFAULT(0)
  );

CREATE TABLE error_logs
  (
     id         NUMBER(5) PRIMARY KEY,
     code       INTEGER,
     message    VARCHAR2 (305),
     backtrace  CLOB,
     callstack  CLOB,
     created_on DATE,
     created_by VARCHAR2 (30)
  );
```

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

```
INSERT INTO people
VALUES(1, 'Gheorghe', 'Nistor', 'M', TO_DATE('23-04-2002', 'DD-MM-
YYYY'), 'nistorgeorge404+member@gmail.com');

INSERT INTO people
VALUES (2, 'Maria', 'Ionescu', 'F', TO_DATE('14-02-2004', 'DD-MM-
YYYY'), 'maria.ionescu@example.com');

INSERT INTO people
VALUES (3, 'Dumitru', 'Matei', 'M', TO_DATE('27-03-2010', 'DD-MM-
YYYY'), 'dumitru.matei@example.com');

INSERT INTO people
VALUES (4, 'Mihai', 'Ionescu', 'M', TO_DATE('04-05-1995', 'DD-MM-
YYYY'), 'mihai.ciocan@example.com');

INSERT INTO people
VALUES (5, 'Elena', 'Dumitru', 'F', TO_DATE('16-08-2002', 'DD-MM-
YYYY'), 'elena.dumitru@example.com');

INSERT INTO people
VALUES (6, 'Ioan', 'Petrescu', 'M', TO_DATE('22-06-2000', 'DD-MM-
YYYY'), 'nistorgeorge404+trainer@gmail.com');

INSERT INTO people
VALUES (7, 'Mihai', 'Petrescu', 'M', TO_DATE('13-07-1996', 'DD-MM-
YYYY'), 'mihai.stoian@example.com');

INSERT INTO people
VALUES (8, 'Andreea', 'Istrate', 'F', TO_DATE('08-09-1991', 'DD-MM-
YYYY'), 'andreea.istrate@example.com');

INSERT INTO people
VALUES (9, 'Radu', 'Mihailescu', 'M', TO_DATE('24-09-2010', 'DD-MM-
YYYY'), 'radu.mihailescu@example.com');

INSERT INTO people
VALUES (10, 'Diana', 'Petrache', 'F', TO_DATE('29-10-2008', 'DD-MM-
YYYY'), 'diana.petrache@example.com');

INSERT INTO members
VALUES(1, 1, 1, 57.5, 176);

INSERT INTO members
VALUES (2, 2, 0, 55.3, 165);

INTO members
VALUES (3, 3, 0, 92.3, 180);
```

```
INSERT INTO members
VALUES (4, 4, 0, 84.3, 182);

INSERT INTO members
VALUES (5, 5, 1, 61.2, 171);

INSERT INTO trainers
VALUES(1, 6, 1,  TO_DATE('01-08-2020', 'DD-MM-YYYY'), 3000);

INSERT INTO trainers
VALUES (2, 7, 0, TO_DATE('10-10-2020', 'DD-MM-YYYY'), 2500);

INSERT INTO trainers
VALUES (3, 8, 1, TO_DATE('23-04-2018', 'DD-MM-YYYY'), 4000);

INSERT INTO trainers
VALUES (4, 9, 0, TO_DATE('15-11-2022', 'DD-MM-YYYY'), 2250);

INSERT INTO trainers
VALUES (5, 10, 0, TO_DATE('05-01-2023', 'DD-MM-YYYY'), 2000);

INSERT INTO memberships
VALUES(1, 1, 3, 1, TO_DATE('03-01-2020', 'DD-MM-YYYY'), TO_DATE('03-02-
2020', 'DD-MM-YYYY'));

INSERT INTO memberships
VALUES(2, 2, 1, 2, TO_DATE('03-01-2020', 'DD-MM-YYYY'), TO_DATE('03-02-
2020', 'DD-MM-YYYY'));

INSERT INTO memberships
VALUES(3, 1, 3, 3, TO_DATE('03-01-2021', 'DD-MM-YYYY'), TO_DATE('03-02-
2021', 'DD-MM-YYYY'));

INSERT INTO memberships
VALUES(4, 2, 1, 4, TO_DATE('03-01-2021', 'DD-MM-YYYY'), TO_DATE('03-02-
2021', 'DD-MM-YYYY'));

INSERT INTO memberships
VALUES(5, 1, 3, 5, TO_DATE('03-01-2022', 'DD-MM-YYYY'), TO_DATE('03-02-
2022', 'DD-MM-YYYY'));

INSERT INTO memberships
VALUES(6, 2, 1, 6, TO_DATE('03-01-2022', 'DD-MM-YYYY'), TO_DATE('03-02-
2022', 'DD-MM-YYYY'));

INSERT INTO memberships
VALUES(7, 1, 3, 7, TO_DATE('03-01-2023', 'DD-MM-YYYY'), TO_DATE('03-02-
2023', 'DD-MM-YYYY'));

INSERT INTO memberships
VALUES(8, 2, 1, 8, TO_DATE('03-01-2023', 'DD-MM-YYYY'), TO_DATE('03-02-
2023', 'DD-MM-YYYY'));
```

```
INSERT INTO memberships
VALUES(9, 3, 2, 10, TO_DATE('03-01-2023', 'DD-MM-YYYY'), TO_DATE('03-02-
2023', 'DD-MM-YYYY'));

INSERT INTO memberships
VALUES(10, 4, 5, 11, TO_DATE('15-01-2023', 'DD-MM-YYYY'), TO_DATE('15-01-
2024', 'DD-MM-YYYY'));

INSERT INTO memberships
VALUES(11, 4, 1, 12, TO_DATE('20-01-2023', 'DD-MM-YYYY'), TO_DATE('20-02-
2024', 'DD-MM-YYYY'));

INSERT INTO membership_types
VALUES (1, 'full-time', 0);

INSERT INTO membership_types
VALUES (2, 'day-time', 0);

INSERT INTO membership_types
VALUES (3, 'full-time', 1);

INSERT INTO membership_types
VALUES (4, 'day-time', 1);

INSERT INTO membership_types
VALUES (5, '12 months', 0);

INSERT INTO attendance_logs
VALUES (1, 1, TO_DATE('03-01-2020', 'DD-MM-YYYY'));

INSERT INTO attendance_logs
VALUES (2, 1, TO_DATE('3-01-2023', 'DD-MM-YYYY'));

INSERT INTO attendance_logs
VALUES (3, 2, TO_DATE('3-01-2023', 'DD-MM-YYYY'));

INSERT INTO attendance_logs
VALUES (4, 1, TO_DATE('4-01-2023', 'DD-MM-YYYY'));

INSERT INTO attendance_logs
VALUES (5, 3, TO_DATE('4-01-2023', 'DD-MM-YYYY'));

INSERT INTO attendance_logs
VALUES (6, 2, TO_DATE('4-01-2023', 'DD-MM-YYYY'));

INSERT INTO attendance_logs
VALUES (7, 1, TO_DATE('10-01-2023', 'DD-MM-YYYY'));

INSERT INTO attendance_logs
VALUES (8, 2, TO_DATE('10-01-2023', 'DD-MM-YYYY'));

INSERT INTO attendance_logs
VALUES (9, 1, TO_DATE('11-01-2023', 'DD-MM-YYYY'));
```

```sql
INSERT INTO attendance_logs
VALUES (10, 2, TO_DATE('11-01-2023', 'DD-MM-YYYY'));

INSERT INTO attendance_logs
VALUES (11, 3, TO_DATE('11-01-2023', 'DD-MM-YYYY'));

INSERT INTO attendance_logs
VALUES (12, 1, TO_DATE('15-01-2023', 'DD-MM-YYYY'));

INSERT INTO attendance_logs
VALUES (13, 1, TO_DATE('20-01-2023', 'DD-MM-YYYY'));

INSERT INTO attendance_logs
VALUES (14, 3, TO_DATE('22-01-2023', 'DD-MM-YYYY'));

INSERT INTO attendance_logs
VALUES (15, 1, TO_DATE('25-01-2023', 'DD-MM-YYYY'));

INSERT INTO attendance_logs
VALUES (16, 2, TO_DATE('25-01-2023', 'DD-MM-YYYY'));

INSERT INTO attendance_logs
VALUES (17, 4, TO_DATE('25-01-2023', 'DD-MM-YYYY'));

INSERT INTO attendance_logs
VALUES (18, 3, TO_DATE('26-01-2023', 'DD-MM-YYYY'));

INSERT INTO attendance_logs
VALUES (19, 4, TO_DATE('27-01-2023', 'DD-MM-YYYY'));

INSERT INTO attendance_logs
VALUES (20, 4, TO_DATE('28-01-2023', 'DD-MM-YYYY'));

INSERT INTO payments
VALUES (1, TO_DATE('03-01-2020', 'DD-MM-YYYY'), 120, 'card');

INSERT INTO payments
VALUES (2, TO_DATE('03-01-2020', 'DD-MM-YYYY'), 150, 'cash');

INSERT INTO payments
VALUES (3, TO_DATE('03-01-2021', 'DD-MM-YYYY'), 120, 'card');

INSERT INTO payments
VALUES (4, TO_DATE('03-01-2021', 'DD-MM-YYYY'), 150, 'cash');

INSERT INTO payments
VALUES (5, TO_DATE('03-01-2022', 'DD-MM-YYYY'), 120, 'card');

INSERT INTO payments
VALUES (6, TO_DATE('03-01-2022', 'DD-MM-YYYY'), 150, 'card');

INSERT INTO payments
VALUES (7, TO_DATE('03-01-2023', 'DD-MM-YYYY'), 120, 'cash');

INSERT INTO payments
VALUES (8, TO_DATE('03-01-2023', 'DD-MM-YYYY'), 150, 'cash');
```

```sql
INSERT INTO payments
VALUES (9, TO_DATE('03-01-2023', 'DD-MM-YYYY'), 100, 'card');

INSERT INTO payments
VALUES (10, TO_DATE('03-01-2023', 'DD-MM-YYYY'), 115, 'cash');

INSERT INTO payments
VALUES (11, TO_DATE('15-01-2023', 'DD-MM-YYYY'), 1500, 'card');

INSERT INTO payments
VALUES (12, TO_DATE('20-01-2023', 'DD-MM-YYYY'), 150, 'card');

INSERT INTO training_sessionsS
VALUES (1, 1, 1, 9, TO_DATE('20-01-2023', 'DD-MM-YYYY'), 78);

INSERT INTO training_session_details
VALUES(1, 1, 14, 4);

INSERT INTO training_session_details
VALUES(2, 1, 15, 4);

INSERT INTO training_session_details
VALUES(3, 1, 16, 1);

INSERT INTO training_session_details
VALUES(4, 1, 17, 6);

INSERT INTO training_session_details
VALUES(5, 1, 1, null);

INSERT INTO equipments
VALUES (1, 'dumbbell');

INSERT INTO equipments
VALUES (2, 'barbell');

INSERT INTO equipments
VALUES (3, 'shoulder press machine');

INSERT INTO equipments
VALUES (4, 'squat rack');

INSERT INTO equipments
VALUES (5, 'biceps curl machine');

INSERT INTO equipments
VALUES (6, 'cable crossover machine');

INSERT INTO exercises
VALUES (1, 1, 'bench press');

INSERT INTO exercises
VALUES (2, 1, 'incline bench press');

INSERT INTO exercises
VALUES (3, 1, 'standing cable fly');
```

```
INSERT INTO exercises
VALUES (4, 2, 'shoulder press');

INSERT INTO exercises
VALUES (5, 2, 'lateral raises');

INSERT INTO exercises
VALUES (6, 2, 'face pulls');

INSERT INTO exercises
VALUES (7, 3, 'triceps pushdown');

INSERT INTO exercises
VALUES (8, 3, 'overhead triceps extension');

INSERT INTO exercises
VALUES (9, 4, 'deadlifts');

INSERT INTO exercises
VALUES (10, 4, 'bent over row');

INSERT INTO exercises
VALUES (11, 4, 'standing lat pulldown');

INSERT INTO exercises
VALUES (12, 5, 'biceps curl');

INSERT INTO exercises
VALUES (13, 5, 'hammer curl');

INSERT INTO exercises
VALUES (14, 6, 'squats');

INSERT INTO exercises
VALUES (15, 6, 'bulgarian split squat');I

NSERT INTO exercises
VALUES (16, 7, 'cable crunches');

INSERT INTO exercises
VALUES (17, 7, 'leg raises');

INSERT INTO sets_reps_weights
VALUES(1, 1, 1, 10, 100);

INSERT INTO sets_reps_weights
VALUES(2, 1, 2, 8, 110);

INSERT INTO sets_reps_weights
VALUES(3, 1, 3, 6, 115);

INSERT INTO sets_reps_weights
VALUES(4, 1, 4, 5, 120);

INSERT INTO sets_reps_weights
VALUES(5, 2, 4, 5, 100);
```

```sql
INSERT INTO muscle_groups
VALUES (1, 'chest');

INSERT INTO muscle_groups
VALUES (2, 'shoulders');

INSERT INTO muscle_groups
VALUES (3, 'triceps');

INSERT INTO muscle_groups
VALUES (4, 'back');

INSERT INTO muscle_groups
VALUES (5, 'biceps');

INSERT INTO muscle_groups
VALUES (6, 'legs');

INSERT INTO muscle_groups
VALUES (7, 'abs');
```

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

Săptămânal instructorii care au o specializare în nutriție trebuie să le trimită clienților pe care aceștia îi antrenează un mail cu macronutrienții pe care trebuie să îi consume zilnic în funcție de greutatea lor corporală așa că am automatizat tot acest proces.

Detalii pentru implementare:

- într-un vector memorez toți membrii care sunt activi

- într-un tabel imbricat memorez toți membrii activi care se antreneză cu un instructor de fitness.

```
CREATE OR replace TYPE vector AS varray(100) OF NUMBER(5);
/
CREATE OR replace PROCEDURE GET_ACTIVE_MEMBERS(active_members IN OUT VECTOR
)
IS
  -- un membru este considerat activ dacă are un abonament valabil, sau
dacă ultimul său abonament a expirat acum maxim o lună
  CURSOR c IS
    SELECT     m.member_id
    FROM       members m
    inner join memberships ms
    ON         m.member_id = ms.member_id
    WHERE      Add_months(ms.end_date, 1) > SYSDATE;
BEGIN
  OPEN c;
  FETCH c bulk collect
  INTO  active_members;
  CLOSE c;
END;
/
CREATE OR replace PROCEDURE macronutrients_message(weight IN NUMBER,
kcal IN NUMBER, email_message IN OUT VARCHAR2)
IS
  proteins      NUMBER(5);
  carbohydrates NUMBER(5);
  fat           NUMBER(5);
BEGIN
  proteins := 2*weight;
  fat := 0.3   *2.2*weight;
  carbohydrates := (kcal-(proteins*4+fat*9))/4;
  email_message := email_message
  || '<td>'
```

```
  || kcal
  ||'g</td>         <td>'
  || proteins
  ||'g</td>         <td>'
  || carbohydrates
  ||'g</td>         <td>'
  || fat
  ||'g</td></tr>';
END;
/
CREATE OR replace PROCEDURE EXERCISE_6
IS
  TYPE nested_table IS TABLE OF members%ROWTYPE;
  active_members VECTOR := VECTOR();
  trained_by_a_nutritionist NESTED_TABLE := NESTED_TABLE();
  status NUMBER(1);
  j      NUMBER(5);
  current_weight training_sessions.weight%TYPE;
  first_name people.first_name%TYPE;
  last_name people.last_name%TYPE;
  email people.email%TYPE;
  email_message VARCHAR(1000) := '';
  kcal          NUMBER(5);
BEGIN
  Get_active_members(active_members);
  -- adăgăm în tabelul trained_by_a_nutritionist toți membrii activi
  -- care sunt antrenați de către un instructor specializat în nutriție
  j := 1;
  FOR i IN active_members.first..active_members.last
  LOOP
    SELECT     Count(*)
    INTO       status
    FROM       training_sessions ts
    inner join members m
    ON         ts.member_id = m.member_id
    inner join trainers t
    ON         ts.trainer_id = t.trainer_id
    WHERE      m.member_id = Active_members(i)
    AND        t.nutritionist = 1
    AND        ROWNUM = 1;

    IF status = 1 THEN
      trained_by_a_nutritionist.extend;
      SELECT *
      INTO   Trained_by_a_nutritionist(j)
      FROM   members
      WHERE  member_id = Active_members(i);

      j := j+1;
```

```
    END IF;
  END LOOP;
  WHILE j > 1
  LOOP
    j := j-1;
    SELECT weight
    INTO   current_weight
    FROM   (
                    SELECT   *
                    FROM     training_sessions
                    WHERE    member_id = Trained_by_a_nutritionist(j).member
_id
                    ORDER BY training_session_date DESC)
    WHERE  ROWNUM = 1;

    SELECT     p.first_name,
               p.last_name,
               p.email
    INTO       first_name,
               last_name,
               email
    FROM       members m
    inner join people p
    ON         m.person_id = p.person_id
    WHERE      m.member_id = Trained_by_a_nutritionist(j).member_id;

    email_message := '<p>Salutare, '
    || first_name
    || '</p>         <p>Pentru greutatea ta actuala de <strong>'
    ||current_weight
    ||' kg</strong> iti recomand să iti consumi macronutrientii in felul urm
ator: </p>
<table>
   <tr>
     <th>Scop</th>
     <th>Kcal</th>
     <th>Proteine</th>
     <th>Carbohidrati</th>
     <th>Grasime</th>
   </tr>';
    -- deficit caloric, kcal de menținere - 500
    kcal := current_weight*2.2*15;
    kcal := kcal-500;
    email_message := email_message
    || '<tr><td>Deficit</td>';
    Macronutrients_message(current_weight, kcal, email_message);
    -- mentinere
    kcal := kcal+500;
    email_message := email_message
```

```
    || '<tr><td>Mentinere</td>';
    Macronutrients_message(current_weight, kcal, email_message);
    -- surplus caloric, kcal de menținere + 300
    kcal := kcal+300;
    email_message := email_message
    || '<tr><td>Surplus</td>';
    Macronutrients_message(current_weight, kcal, email_message);
    email_message := email_message
    || '</table><p>O zi bună cu împliniri!</p>';
    ----trimitere mail
    apex_mail.Send('nistorgeorge666@gmail.com', email, 'Tabel macronutrienti
', email_message);
    dbms_output.Put_line(first_name
    || ' '
    || last_name
    || ' a primit mail cu succes!');
  END LOOP;
END;
/

BEGIN
    Exercise_6();
END;
/
```

```
120 BEGIN
121     Exercise_6();
122 END;
123 /
124
```

Script Output | Query Result | Query Result

Task completed in 1,335 seconds

```
PL/SQL procedure successfully completed.
```

George

```
Gheorghe Nistor a primit mail cu succes!
```

# Tabel macronutrienti

Inbox ×

**nistorgeorge666@gmail.com**
to nistorgeorge404+member

Salutare, Gheorghe

Pentru greutatea ta actuala de **78 kg** iti recomand sa iti consumi macronutrientii in felul urmator:

| Scop | Kcal | Proteine | Carbohidrati | Grasime |
|---|---|---|---|---|
| Deficit | 2074g | 156g | 248g | 51g |
| Mentinere | 2574g | 156g | 373g | 51g |
| Surplus | 2874g | 156g | 448g | 51g |

O zi buna cu impliniri!

Reply | Reply all | Forward

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. Apelați subprogramul

La sfârșit de an manager-ul sălii dorește să își recompenseze cu suplimente proteice 3 membrii care au un abonament activ, au vârsta mai mare de 16 ani și se află în topul clasamentului a celor mai activi clienți din anul respectiv.

Detalii pentru implementare:

- într-un cursor salvăm toți membrii care au un abonament activ și vârsta mai mare de 16 ani.

- cursul parametrizat are ca și argument un id al unui membru și calculează numărul total de antrenamente pe care clientul le-a avut în anul respectiv.

```
CREATE OR replace PROCEDURE EXERCISE_7
IS
  CURSOR active_members_over_16 IS
    SELECT m.member_id,
           p.first_name,
           p.last_name
    FROM   members m
           inner join memberships ms
                   ON m.member_id = ms.member_id
           inner join people p
                   ON p.person_id = m.person_id
    WHERE  Add_months(ms.end_date, 1) > SYSDATE
           AND ( Months_between(Trunc(SYSDATE), p.birth_date) / 12 ) >= 16
           AND p.gender = 'M';
  CURSOR number_of_entrances(
    id members.member_id%TYPE) IS
    SELECT Count(*)
    FROM   attendance_logs
    WHERE  member_id = id
        AND Extract(year FROM attendance_date) = Extract(year FROM SYSDATE);
  TYPE m_record IS RECORD (
    member_id members.member_id%TYPE,
    first_name people.first_name%TYPE,
    last_name people.last_name%TYPE,
    nr INTEGER := 0 );
  CURRENT members.member_id%TYPE;
  first   M_RECORD;
  second  M_RECORD;
  third   M_RECORD;
```

```
BEGIN
    FOR m IN active_members_over_16 LOOP
        OPEN number_of_entrances(m.member_id);
        FETCH number_of_entrances INTO CURRENT;
        CLOSE number_of_entrances;
        IF CURRENT > first.nr THEN
          third := second;
          second := first;
          first.nr := CURRENT;
          first.member_id := m.member_id;
          first.first_name := m.first_name;
          first.last_name := m.last_name;
        ELSIF CURRENT > second.nr THEN
          third := second;
          second.nr := CURRENT;
          second.member_id := m.member_id;
          second.first_name := m.first_name;
          second.last_name := m.last_name;
        ELSIF CURRENT > third.nr THEN
          third.nr := CURRENT;
          third.member_id := m.member_id;
          third.first_name := m.first_name;
          third.last_name := m.last_name;
        END IF;
    END LOOP;
    IF first.nr > 0 THEN
      dbms_output.Put_line(first.first_name
                           || ' '
                           || first.last_name
                           || ' are '
                           || first.nr
                           || ' intrari la sala in anul '
                           || Extract(year FROM SYSDATE));
    END IF;
    IF second.nr > 0 THEN
      dbms_output.Put_line(second.first_name
                           || ' '
                           || second.last_name
                           || ' are '
                           || second.nr
                           || ' intrari la sala in anul '
                           || Extract(year FROM SYSDATE));
    END IF;
    IF third.nr > 0 THEN
      dbms_output.Put_line(third.first_name
                           || ' '
                           || third.last_name
                           || ' are '
                           || third.nr
```

```
                                || ' intrari la sala in anul '
                                || Extract(year FROM SYSDATE));
    END IF;
END;
/


BEGIN
    Exercise7();
END;
/
```

```
George
Gheorghe Nistor are 8 intrari la sala in anul 2023
Maria Ionescu are 5 intrari la sala in anul 2023
Mihai Ciocan are 3 intrari la sala in anul 2023
```

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.

La începutul fiecărei luni, manager-ul sălii de fitness dorește să plătească salariul fiecărui antrenor în parte, dar pe lângă salariul fix vrea să adauge și un procent de 10% din toți banii făcuți din ședințele de antrenament din luna precedentă.

```
CREATE TABLE error_logs
  (
     id         INTEGER UNIQUE NOT NULL,
     code       INTEGER,
     message    VARCHAR2 (305),
     backtrace  CLOB,
     callstack  CLOB,
     created_on DATE,
     created_by VARCHAR2 (30)
  );
CREATE OR replace PROCEDURE RECORD_ERROR
IS
  PRAGMA autonomous_transaction;
  id   INTEGER;
  code INTEGER := SQLCODE;
BEGIN
    SELECT Max(id)
    INTO   id
    FROM   error_logs;
    IF id IS NULL THEN
      id := 0;
    END IF;
    INSERT INTO error_log
    VALUES      (id + 1,
                 code,
                 dbms_utility.format_error_stack,
                 dbms_utility.format_error_backtrace,
                 dbms_utility.format_call_stack,
                 SYSDATE,
                 USER);
    COMMIT;
END;
/

CREATE OR REPLACE FUNCTION EXERCISE_8(t_last_name IN people.last_name%type,
                       t_first_name IN people.first_name%type DEFAULT NULL)
RETURN trainers.trainer_id%type
```

```
IS
    number_of_rows NUMBER;
    t_id trainers.trainer_id%type;
    salary trainers.salary%type;
    bonus payments.amount%type;
    no_trainer_found EXCEPTION;
    too_many_trainers_found EXCEPTION;
    CURSOR get_trainer_first_name_cursor(t_last_name people.last_name%type)
IS
        SELECT p.first_name, p.last_name
        FROM people p
        INNER JOIN trainers t
        ON p.person_id = t.person_id
        WHERE p.last_name = t_last_name;

BEGIN
  -- mă folosesc de acest query pentru a arunca excepțiile de către mine
 manual
    SELECT COUNT(*) as number_of_rows
    INTO number_of_rows
    FROM trainers t
    INNER JOIN people p
    ON t.person_id = p.person_id
    WHERE  p.last_name = t_last_name AND p.first_name =
        (CASE
            WHEN t_first_name IS NOT NULL THEN
                 t_first_name
            ELSE
                p.first_name
        END);

    IF number_of_rows= 0 THEN
        RAISE no_trainer_found;
    ELSIF number_of_rows > 1 THEN
        RAISE too_many_trainers_found;
    END IF;

    SELECT t.trainer_id
    INTO t_id
    FROM trainers t
    INNER JOIN people p
    ON t.person_id = p.person_id
    WHERE  p.last_name = t_last_name AND p.first_name =
        (CASE
            WHEN t_first_name IS NOT NULL THEN
                 t_first_name
            ELSE
                p.first_name
        END);
```

```
    SELECT salary
    INTO salary
    FROM trainers t
    WHERE t.trainer_id = t_id;

    SELECT NVL(SUM(p.amount), 0)
    INTO bonus
    FROM trainers t
    JOIN training_sessions ts
    ON t.trainer_id = ts.trainer_id
    JOIN payments p
    ON p.payment_id = ts.payment_id
    WHERE t.trainer_id = t_id AND ts.training_session_date >= TRUNC(ADD_MONT
HS(SYSDATE, -1), 'MM') AND ts.training_session_date <= trunc(sysdate, 'MM');

    IF bonus <> 0 THEN
        salary := salary + 0.1*bonus;
    END IF;
    RETURN salary;

    EXCEPTION
        WHEN no_trainer_found THEN
            DBMS_OUTPUT.PUT_LINE('Nu exista niciun antrenor cu numele de fam
ilie ' || t_last_name);
        WHEN too_many_trainers_found THEN
            DBMS_OUTPUT.PUT_LINE('Exista mai multi antrenor cu numele de fam
ilie ' || t_last_name);
            DBMS_OUTPUT.PUT_LINE('In acest caz functia trebuie apelata imrep
una cu prenume instructorului');
            FOR i IN get_trainer_first_name_cursor(t_last_name) LOOP
                DBMS_OUTPUT.PUT_LINE(i.last_name || ' ' || i.first_name);
            END LOOP;
END;
/
BEGIN
    DBMS_OUTPUT.PUT_LINE(Exercise8('Petrescu'));
END;
/
BEGIN
    DBMS_OUTPUT.PUT_LINE(Exercise8('Oprea'));
END;
/
BEGIN
    DBMS_OUTPUT.PUT_LINE(Exercise8('Petrescu', 'Ioan'));
END;
/
BEGIN
    DBMS_OUTPUT.PUT_LINE(Exercise8('Mihailescu'));
```

```
END;
/
```

George

```
Exista mai multi antrenor cu numele de familie Petrescu
In acest caz funcția trebuie apelata imrepuna cu prenume instructorului
Petrescu Ioan
Petrescu Mihai

Nu exista niciun antrenor cu numele de familie Oprea

3010

2250
```

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.

Apropriindu-se luna ianuarie, manager-ul sălii de sport dorește să angajeze un nou antrenor pentru a reușii să facă față numărului mare de membrii noi care vor avea nevoie de un antrenor personal. Acesta dorește să invite la interview chiar pe unii din membrii lui activi, dar înainte de asta dorește să afle dacă aceștia îi îndeplinesc condiția lui foarte simplă: să fie un client vechi de cel puțin 2 ani și să fie capabil să făcă 5 genoflexiuni cu 120kg la squat rack.

```
CREATE OR REPLACE PROCEDURE EXERCISE_9(m_last_name IN people.last_name%type,
 m_first_name IN people.first_name%type DEFAULT NULL)
IS
    m_id members.member_id%type;
    status NUMBER(1) := 0;
    CURSOR get_member_first_name_cursor(t_last_name people.last_name%type)
    IS
        SELECT p.first_name, p.last_name
        FROM people p
        INNER JOIN members m
        ON p.person_id = m.person_id
        WHERE p.last_name = m_last_name;
    -- am folosit 9 tabele
    CURSOR all_eligible_members_cursor IS
        SELECT m.member_id as member_id,
               MAX(p.first_name) as first_name,
               MAX(p.last_name) as last_name
        FROM members m
        INNER JOIN attendance_logs al
        ON m.member_id = al.member_id
        INNER JOIN people p
        ON m.person_id = p.person_id
        WHERE attendance_date <= add_months(trunc(sysdate,'year'), -
24) and m.member_id IN (
            SELECT active_members.member_id
            FROM training_session_details tsd
            JOIN training_sessions ts
            ON tsd.training_session_id = ts.training_session_id
            JOIN exercises ex
            ON tsd.exercise_id = ex.exercise_id
            JOIN equipments eq
            ON tsd.equipment_id = eq.equipment_id
            JOIN sets_reps_weights srw
```

```
            ON tsd.training_session_detail_id = srw.training_session_detail_
id
            JOIN (
                SELECT m.member_id
                FROM members m
                INNER JOIN memberships ms
                ON m.member_id = ms.member_id
                WHERE ADD_MONTHS(ms.END_DATE, 1) >= SYSDATE
            ) active_members
            ON ts.member_id = active_members.member_id
            WHERE ex.exercise_name = 'squats' and eq.equipment_name = 'squat
 rack' and srw.reps >= 5 and srw.weight >= 120)
        Group BY m.member_id;
BEGIN
    IF m_first_name IS NULL THEN
        SELECT m.member_id
        INTO m_id
        FROM people p
        INNER JOIN members m
        ON p.person_id = m.person_id
        WHERE p.last_name = m_last_name;
    ELSE
       SELECT m.member_id
        INTO m_id
        FROM people p
        INNER JOIN members m
        ON p.person_id = m.person_id
        WHERE p.last_name = m_last_name AND p.first_name = m_first_name;
    END IF;
-- in cursorul all_eligible_members_cursor sunt salvati toti membrii care
 sunt eligili pentru un interviu
-- cautam daca membrul dat ca parametru se afla in aceasta lista
    FOR i IN all_eligible_members_cursor LOOP
        IF i.member_id = m_id THEN
           status := 1;
        END IF;
    END LOOP;
    DBMS_OUTPUT.PUT(m_last_name || ' ');
    IF m_first_name IS NOT NULL THEN
        DBMS_OUTPUT.PUT(m_first_name || ' ');
    END IF;
    IF status = 0 THEN
        DBMS_OUTPUT.PUT('NU ');
    END IF;
    DBMS_OUTPUT.PUT('indeplinește toate criteriile pentru interview');
    DBMS_OUTPUT.NEW_LINE;
    EXCEPTION
        WHEN NO_DATA_FOUND THEN
            record_error();
```

```
            DBMS_OUTPUT.PUT_LINE('Nu exista niciun client cu numele de famil
ie ' || m_last_name);
        WHEN TOO_MANY_ROWS THEN
            record_error();
            DBMS_OUTPUT.PUT_LINE('Exista mai multi clienti cu numele de fami
lie ' || m_last_name);
            DBMS_OUTPUT.PUT_LINE('In acest caz funcția trebuie apelata imrep
una cu prenume clientului');
            FOR i IN get_member_first_name_cursor(m_last_name) LOOP
                DBMS_OUTPUT.PUT_LINE(i.last_name || ' ' || i.first_name);
            END LOOP;
END;
/

BEGIN
    Exercise9('Nistor');
END;
/

BEGIN
    Exercise9('Mihailescu');
END;
/

BEGIN
    Exercise9('Ionescu');
END;
/

BEGIN
    Exercise9('Ionescu', 'Maria');
END;
/
```

```
Nistor indeplinește toate criteriile pentru interview

Nu exista niciun client cu numele de familie Mihailescu

Exista mai multi clienti cu numele de familie Ionescu
In acest caz funcția trebuie apelata imrepuna cu prenume clientului
Ionescu Maria
Ionescu Mihai

Ionescu Maria NU indeplinește toate criteriile pentru interview
```

10. Definiți un trigger de tip LMD la nivel de comandă. Declanșați trigger-ul.
În momentul în care un membru intră în sala de fitness, dorim să se adauge o linie în Attendance_Logs doar dacă ora curentă este între 05.30 – 01.30 (acesta fiind programul sălii de sport) și dacă clientul aare un abonament activ. Să nu mai fi intrat o data in sala in ziua respectiva.

```
CREATE OR replace TRIGGER EXERCISE_10
  BEFORE INSERT OR UPDATE OR DELETE ON attendance_logs
DECLARE
    current_hour NUMBER(2);
BEGIN
    SELECT Extract(hour FROM Cast(SYSDATE AS TIMESTAMP))
    INTO   current_hour
    FROM   dual;
    IF ( ( To_char(SYSDATE, 'D') = 6 )
          OR ( To_char(SYSDATE, 'D') = 1 ) ) THEN
      IF current_hour < 8 THEN
        Raise_application_error(-20001,
        'În weekend, sala este deschisă doar în intervalul 08.00-24:00');
      END IF;
    ELSIF current_hour > 1 AND current_hour < 6 THEN
      Raise_application_error(-20002,
'În cursul săptămânii, sala este deschisă doar în intervalul 06.00-01:00')
  ;
END IF;
END;
/
```

```
24 -- vineri, ora 03:25
25 INSERT INTO attendance_logs
26 VALUES (21, 4, TO_DATE('29-01-2023', 'DD-MM-YYYY'));
```

Script Output × | Query Result ×

Task completed in 0,043 seconds

```
Error starting at line : 25 in command -
INSERT INTO attendance_logs
VALUES (21, 4, TO_DATE('29-01-2023', 'DD-MM-YYYY'))
Error report -
ORA-20002: În cursul săptămânii, sala este deschisă doar în intervalul 06.00-01:00
ORA-06512: at "GEORGE.EXERCISE_10", line 12
ORA-04088: error during execution of trigger 'GEORGE.EXERCISE_10'
```

```
23
24  -- sâmbătă, ora 00:30
25  INSERT INTO attendance_logs
26  VALUES (21, 4, TO_DATE('29-01-2023', 'DD-MM-YYYY'));
```

Script Output x | Query Result x

Task completed in 0,063 seconds

```
Error starting at line : 25 in command -
INSERT INTO attendance_logs
VALUES (21, 4, TO_DATE('29-01-2023', 'DD-MM-YYYY'))
Error report -
ORA-20001: În weekend, sala este deschisă doar în intervalul 08.00-24:00
ORA-06512: at "GEORGE.EXERCISE_10", line 9
ORA-04088: error during execution of trigger 'GEORGE.EXERCISE_10'
```

11. Definiți un trigger de tip LMD la nivel de linie. Declanșați trigger-ul.
În momentul în care un membru intră în sala de fitness, dorim să se adauge o linie în Attendance_Logs doar dacă clientul are un singur abonament activ, iar dacă este trecut de ora 16, abonamentul trebuie să fie unul full-time.

```
CREATE OR REPLACE TRIGGER EXERCISE_11
BEFORE INSERT OR UPDATE ON attendance_logs
FOR EACH ROW
DECLARE
    membership_name membership_types.membership_name%type;
BEGIN
    IF UPDATING AND (:OLD.member_id) <> (:NEW.member_id) THEN
        RAISE_APPLICATION_ERROR(-
20003, 'Este strict interzisa schimbarea id-ului unui membru');
    END IF;
    SELECT mt.membership_name
    INTO membership_name
    FROM members m
    INNER JOIN memberships ms
    ON m.member_id = ms.member_id
    INNER JOIN membership_types mt
    ON ms.membership_type_id = mt.membership_type_id
    WHERE m.member_id = (:NEW.member_id) AND ms.end_date >= (:NEW.attendance
_date);
    DBMS_OUTPUT.PUT_LINE(membership_name);
    IF membership_name LIKE '%day%' AND extract(hour from cast(sysdate as ti
mestamp)) > 16 THEN
        RAISE_APPLICATION_ERROR(-
20003, 'Accesul în sala de fitness pe baza unui abonament de tipul day-
time se face până la ora 16');
    END IF;
```

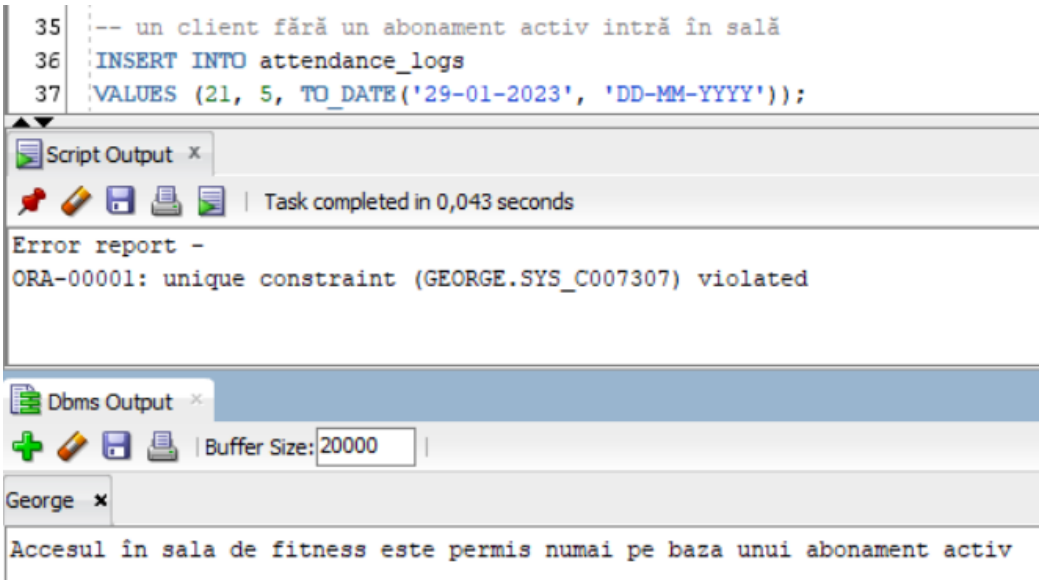
```
    EXCEPTION
        WHEN NO_DATA_FOUND THEN
            record_error();
            DBMS_OUTPUT.PUT_LINE('Accesul în sala de fitness este permis num
ai pe baza unui abonament activ');
        WHEN TOO_MANY_ROWS THEN
            record_error();
            DBMS_OUTPUT.PUT_LINE('Clientul introdus mai multe abonamente act
ive');
END;/

-- un client fără un abonament activ intră la sală
INSERT INTO attendance_logs
VALUES (21, 5, TO_DATE('29-01-2023', 'DD-MM-YYYY'));

-- un client cu un abonament de tip day-
time intră în sală dupa ora 16INSERT INTO attendance_logs
VALUES (21, 3, TO_DATE('29-01-2023', 'DD-MM-YYYY'));

-- un client are mai multe abonamente activeINSERT INTO attendance_logs
VALUES (21, 4, TO_DATE('25-01-2023', 'DD-MM-YYYY'));

-- schimbarea id-ului unui membruUPDATE attendance_logs
SET member_id = 1
WHERE attendance_id = 20;
```



```
37  -- un client cu un abonament de tip day-time intră în sală după ora 16
38  INSERT INTO attendance_logs
39  VALUES (21, 3, TO_DATE('25-01-2023', 'DD-MM-YYYY'));
40
```

Script Output × | Query Result ×

Task completed in 0,033 seconds

```
Trigger EXERCISE_11 compiled


Error starting at line : 38 in command -
INSERT INTO attendance_logs
VALUES (21, 3, TO_DATE('25-01-2023', 'DD-MM-YYYY'))
Error report -
ORA-20003: Accesul în sala de fitness pe baza unui abonament de tipul day-time se face până la ora 16
ORA-06512: at "GEORGE.EXERCISE_11", line 19
ORA-04088: error during execution of trigger 'GEORGE.EXERCISE_11'
```

```
36  -- un client are mai multe abonamente active
37  INSERT INTO attendance_logs
38  VALUES (21, 4, TO_DATE('25-01-2023', 'DD-MM-YYYY'))
39
```

Script Output × | Query Result ×

Task completed in 0,036 seconds

```
Error starting at line : 37 in command -
INSERT INTO attendance_logs
VALUES (21, 4, TO_DATE('25-01-2023', 'DD-MM-YYYY'))
Error report -
ORA-00001: unique constraint (GEORGE.SYS_C007313) violate
```

Dbms Output ×

Buffer Size: 20000

George ×

```
Clientul introdus mai multe abonamente active
```

```
47  -- schimbarea id-ului unui membru
48  UPDATE attendance_logs
49  SET member_id = 1
50  WHERE attendance_id = 20;
```

Script Output × | Query Result ×

Task completed in 0,03 seconds

```
Error starting at line : 47 in command -
UPDATE attendance_logs
SET member_id = 1
WHERE attendance_id = 20
Error report -
ORA-20003: Este strict interzisa schimbarea id-ului unui membru
ORA-06512: at "GEORGE.EXERCISE_11", line 5
ORA-04088: error during execution of trigger 'GEORGE.EXERCISE_11'
```

12. Definiți un trigger de tip LDD. Declanșați trigger-ul.
În tabelul DDL_LOGS sunt salvate automat log-urile fiecarei operație de tipul “Data definition language” care a avut loc în baza de date.

```
CREATE TABLE DDL_LOGS
  (
     id           NUMBER(5) UNIQUE NOT NULL,
     system_event VARCHAR2(50),
     object_name  VARCHAR2(50),
     object_type  VARCHAR2(50),
     created_on   DATE,
     created_by   VARCHAR2 (30)
  );

CREATE OR replace TRIGGER ddl_trigger
  AFTER CREATE OR ALTER OR DROP ON SCHEMA
DECLARE
    id NUMBER;
BEGIN
    SELECT MAX(id)
    INTO   id
    FROM   ddl_logs;

    IF id IS NULL THEN
      id := 0;
    END IF;

    INSERT INTO ddl_logs
    VALUES     (id + 1,
                sys.sysevent,
                sys.dictionary_obj_name,
                sys.dictionary_obj_type,
                SYSDATE,
                sys.login_user);
END;
/
COMMIT;
```

```
CREATE TABLE test (
    id INTEGER
);

DROP TABLE test;

SELECT * FROM ddl_logs;
```

ript Output | Query Result

SQL | All Rows Fetched: 4 in 0,005 seconds

| | ID | SYSTEM_EVENT | OBJECT_NAME | OBJECT_TYPE | CREATED_ON | CREATED_BY |
|---|---|---|---|---|---|---|
| 1 | 1 | CREATE | TEST | TABLE | 12-01-2023 | GEORGE |
| 2 | 2 | CREATE | DDL_TRIGGER | TRIGGER | 12-01-2023 | GEORGE |
| 3 | 3 | DROP | TEST | TABLE | 12-01-2023 | GEORGE |
| 4 | 4 | DROP | TEST | TABLE | 12-01-2023 | GEORGE |

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

```
CREATE OR REPLACE PACKAGE EXERCISE_13
AS
    TYPE vector IS VARRAY(100) OF NUMBER(5);
    PROCEDURE get_active_members(active_members IN OUT vector);
    PROCEDURE macronutrients_message(weight IN NUMBER, kcal IN NUMBER, email
_message IN OUT VARCHAR2);
    PROCEDURE  EXERCISE_6;
    PROCEDURE EXERCISE_7;
    FUNCTION EXERCISE_8(t_last_name IN people.last_name%type, t_first_name I
N people.first_name%type DEFAULT NULL) RETURN trainers.trainer_id%type;
    PROCEDURE Exercise9(m_last_name IN people.last_name%type, m_first_name I
N people.first_name%type DEFAULT NULL);
    PROCEDURE record_error;
END Exercise13;/CREATE OR REPLACE PACKAGE BODY  Exercise13 AS
    PROCEDURE get_active_members(active_members IN OUT vector)
    IS
        -
- un membru este considerat activ dacă are un abonament valabil, sau dacă ul
timul său abonament a expirat acum maxim o lună
        CURSOR c IS
            SELECT m.member_id
            FROM members m
            INNER JOIN memberships ms
            ON m.member_id = ms.member_id
            WHERE ADD_MONTHS(ms.END_DATE, 1) >= SYSDATE;
    BEGIN
        OPEN c;
        FETCH c BULK COLLECT INTO active_members;
        CLOSE c;
```

```
    END get_active_members;
    PROCEDURE macronutrients_message(weight IN NUMBER, kcal IN NUMBER, email
_message IN OUT VARCHAR2)
    IS
        proteins NUMBER(5);
        carbohydrates NUMBER(5);
        fat NUMBER(5);
    BEGIN
        proteins := 2*weight;
        fat := 0.3*2.2*weight;
        carbohydrates := (kcal-(proteins*4+fat*9))/4;
        email_message := email_message ||
            '<td>'||kcal||'g</td>
             <td>'||proteins||'g</td>
             <td>'||carbohydrates ||'g</td>
             <td>'||fat||'g</td></tr>';
    END macronutrients_message;
    PROCEDURE Exercise_6
    IS
        TYPE nested_table IS TABLE OF members%ROWTYPE;
        active_members vector := vector();
        trained_by_a_nutritionist nested_table := nested_table();
        status NUMBER(1);
        j      NUMBER(5);
        current_weight training_sessions.weight%type;
        first_name people.first_name%type;
        last_name people.last_name%type;
        email people.email%type;
        email_message VARCHAR(1000) := '';
        kcal   NUMBER(5);
    BEGIN
        get_active_members(active_members);
        -- adăgăm în tabelul trained_by_a_nutritionist toti membrii activi
        -- care sunt antrenati de către un instructor specializat în
           nutritie
        j := 1;
        FOR i IN active_members.FIRST..active_members.LAST LOOP
            SELECT COUNT(*)
            INTO status
            FROM training_sessions ts
            INNER JOIN members m
            ON ts.member_id = m.member_id
            INNER JOIN trainers t
            ON ts.trainer_id = t.trainer_id
            WHERE m.member_id = active_members(i) AND t.nutritionist = 1 AND
 rownum = 1;
            IF status = 1 THEN
                trained_by_a_nutritionist.extend;
                SELECT *
```

```
                INTO trained_by_a_nutritionist(j)
                FROM members
                WHERE member_id = active_members(i);
                j := j+1;
            END IF;
        END LOOP;

        WHILE j > 1 LOOP
            j := j-1;
            SELECT weight
            INTO current_weight
                FROM (
                    SELECT *
                    FROM training_sessions
                    WHERE member_id = trained_by_a_nutritionist(j).member_id
                    ORDER BY training_session_date DESC)
            WHERE rownum = 1;

            SELECT p.first_name, p.last_name, p.email
            INTO first_name, last_name, email
            FROM members m
            INNER JOIN people p
            ON m.person_id = p.person_id
            WHERE m.member_id = trained_by_a_nutritionist(j).member_id;

            email_message :=  '<p>Salutare, '|| first_name || '</p>
    <p>Pentru greutatea ta actuala de <strong>'||current_weight||'
kg</strong> iti recomand să iti consumi macronutrientii in felul urmator:
</p>
<table>
   <tr>
      <th>Scop</th>
      <th>Kcal</th>
      <th>Proteine</th>
      <th>Carbohidrati</th>
      <th>Grasime</th>
   </tr>';
            -- deficit caloric, kcal de mentinere - 500
            kcal := current_weight*2.2*15;
            kcal := kcal-500;
            email_message := email_message || '<tr><td>Deficit</td>';
            macronutrients_message(current_weight, kcal, email_message);
            -- mentinere
            kcal := kcal+500;
            email_message := email_message || '<tr><td>Mentinere</td>';
            macronutrients_message(current_weight, kcal, email_message);
            -- surplus caloric, kcal de mentinere + 300
            kcal := kcal+300;
            email_message := email_message || '<tr><td>Surplus</td>';
```

```
            macronutrients_message(current_weight, kcal, email_message);
            email_message := email_message || '</table><p>O zi bună cu impli
niri!</p>';
            ----trimitere mail
            apex_mail.send('nistorgeorge666@gmail.com', email, 'Tabel macron
utrienti', email_message);
            DBMS_OUTPUT.PUT_LINE(first_name || ' ' || last_name || ' a primi
t mail cu succes!');
        END LOOP;
    END EXERCISE_6;
    PROCEDURE EXERCISE_7
    IS
        CURSOR active_members_over_16 IS
            SELECT m.member_id, p.first_name, p.last_name
            FROM members m
            INNER JOIN memberships ms
            ON m.member_id = ms.member_id
            INNER JOIN people p
            ON p.person_id = m.person_id
            WHERE ADD_MONTHS(ms.END_DATE, 1) > SYSDATE AND (months_between(T
RUNC(sysdate), p.birth_date)/12) >= 16;

        CURSOR number_of_entrances(id members.member_id%type) IS
            SELECT COUNT(*)
            FROM attendance_logs
            WHERE member_id = id AND EXTRACT(YEAR FROM attendance_date) = EX
TRACT(YEAR FROM SYSDATE);

        TYPE m_record IS RECORD (
            member_id members.member_id%type,
            first_name people.first_name%type,
            last_name people.last_name%type,
            nr INTEGER := 0
        );
        current members.member_id%type;
        first m_record;
        second m_record;
        third m_record;
    BEGIN
        FOR m in active_members_over_16 LOOP
            OPEN number_of_entrances(m.member_id);
            FETCH number_of_entrances INTO current;
            CLOSE number_of_entrances;
            IF current > first.nr THEN
                third := second;
                second := first;

                first.nr := current;
                first.member_id := m.member_id;
```

```
                first.first_name := m.first_name;
                first.last_name := m.last_name;

            ELSIF current > second.nr THEN
                third := second;
                second.nr := current;
                second.member_id := m.member_id;
                second.first_name := m.first_name;
                second.last_name := m.last_name;

            ELSIF current > third.nr THEN
                third.nr := current;
                third.member_id := m.member_id;
                third.first_name := m.first_name;
                third.last_name := m.last_name;
            END IF;
        END LOOP;
        IF first.nr > 0 THEN
            DBMS_OUTPUT.PUT_LINE(first.first_name || ' ' || first.last_name
|| ' are ' || first.nr || ' intrari la sala in anul ' || EXTRACT(YEAR FROM S
YSDATE));
        END IF;
        IF second.nr > 0 THEN
            DBMS_OUTPUT.PUT_LINE(second.first_name || ' ' || second.last_nam
e || ' are ' || second.nr || ' intrari la sala in anul ' || EXTRACT(YEAR FRO
M SYSDATE));
        END IF;
        IF third.nr > 0 THEN
            DBMS_OUTPUT.PUT_LINE(third.first_name || ' ' || third.last_name
|| ' are ' || third.nr || ' intrari la sala in anul ' || EXTRACT(YEAR FROM S
YSDATE));
        END IF;
    END EXERCISE_7;
    FUNCTION EXERCISE_8 (t_last_name IN people.last_name%type,
                         t_first_name IN people.first_name%type DEFAULT NULL)
    RETURN trainers.trainer_id%type
    IS
        number_of_rows NUMBER;
        t_id trainers.trainer_id%type;
        salary trainers.salary%type;
        bonus payments.amount%type;
        no_trainer_found EXCEPTION;
        too_many_trainers_found EXCEPTION;
        CURSOR get_trainer_first_name_cursor(t_last_name people.last_name%ty
pe) IS
            SELECT p.first_name, p.last_name
            FROM people p
            INNER JOIN trainers t
            ON p.person_id = t.person_id
```

```
            WHERE p.last_name = t_last_name;

    BEGIN
    - mă folosesc de acest query pentru a arunca exceptiile de către
    mine manual
        SELECT COUNT(*) as number_of_rows
        INTO number_of_rows
        FROM trainers t
        INNER JOIN people p
        ON t.person_id = p.person_id
        WHERE  p.last_name = t_last_name AND p.first_name =
            (CASE
                WHEN t_first_name IS NOT NULL THEN
                     t_first_name
                ELSE
                    p.first_name
            END);

        IF number_of_rows= 0 THEN
            RAISE no_trainer_found;
        ELSIF number_of_rows > 1 THEN
            RAISE too_many_trainers_found;
        END IF;

        SELECT t.trainer_id
        INTO t_id
        FROM trainers t
        INNER JOIN people p
        ON t.person_id = p.person_id
        WHERE  p.last_name = t_last_name AND p.first_name =
            (CASE
                WHEN t_first_name IS NOT NULL THEN
                     t_first_name
                ELSE
                    p.first_name
            END);

        SELECT salary
        INTO salary
        FROM trainers t
        WHERE t.trainer_id = t_id;

        SELECT NVL(SUM(p.amount), 0)
        INTO bonus
        FROM trainers t
        JOIN training_sessions ts
        ON t.trainer_id = ts.trainer_id
        JOIN payments p
        ON p.payment_id = ts.payment_id
```

```
        WHERE t.trainer_id = t_id AND ts.training_session_date >= TRUNC(ADD_
MONTHS(SYSDATE, -
1), 'MM') AND ts.training_session_date <= trunc(sysdate, 'MM');

        IF bonus <> 0 THEN
            salary := salary + 0.1*bonus;
        END IF;
        RETURN salary;

        EXCEPTION
            WHEN no_trainer_found THEN
                DBMS_OUTPUT.PUT_LINE('Nu exista niciun antrenor cu numele de
 familie ' || t_last_name);
            WHEN too_many_trainers_found THEN
                DBMS_OUTPUT.PUT_LINE('Exista mai multi antrenor cu numele de
 familie ' || t_last_name);
                DBMS_OUTPUT.PUT_LINE('In acest caz functia trebuie apelata i
mrepuna cu prenume instructorului');
                FOR i IN get_trainer_first_name_cursor(t_last_name) LOOP
                    DBMS_OUTPUT.PUT_LINE(i.last_name || ' ' || i.first_name)
;
                END LOOP;
    END EXERCISE_8;
    PROCEDURE EXERCISE_9 (m_last_name IN people.last_name%type,
                          m_first_name IN people.first_name%type DEFAULT NULL)
    IS
        m_id members.member_id%type;
        status NUMBER(1) := 0;
        CURSOR get_member_first_name_cursor(t_last_name people.last_name%typ
e) IS
            SELECT p.first_name, p.last_name
            FROM people p
            INNER JOIN members m
            ON p.person_id = m.person_id
            WHERE p.last_name = m_last_name;
        -- am folosit 9 tabele
        CURSOR all_eligible_members_cursor IS
            SELECT m.member_id as member_id, MAX(p.first_name) as first_name
, MAX(p.last_name) as last_name
            FROM members m
            INNER JOIN attendance_logs al
            ON m.member_id = al.member_id
            INNER JOIN people p
            ON m.person_id = p.person_id
            WHERE attendance_date <= add_months(trunc(sysdate,'year'), -
24) and m.member_id IN (
                SELECT active_members.member_id
                FROM training_session_details tsd
                JOIN training_sessions ts
```

```
                ON tsd.training_session_id = ts.training_session_id
                JOIN exercises ex
                ON tsd.exercise_id = ex.exercise_id
                JOIN equipments eq
                ON tsd.equipment_id = eq.equipment_id
                JOIN sets_reps_weights srw
                ON tsd.training_session_detail_id = srw.training_session_det
ail_id
                JOIN (
                    SELECT m.member_id
                    FROM members m
                    INNER JOIN memberships ms
                    ON m.member_id = ms.member_id
                    WHERE ADD_MONTHS(ms.END_DATE, 1) >= SYSDATE
                ) active_members
                ON ts.member_id = active_members.member_id
                WHERE ex.exercise_name = 'squats' and eq.equipment_name = 's
quat rack' and srw.reps >= 5 and srw.weight >= 120)
            Group BY m.member_id;
    BEGIN
        IF m_first_name IS NULL THEN
            SELECT m.member_id
            INTO m_id
            FROM people p
            INNER JOIN members m
            ON p.person_id = m.person_id
            WHERE p.last_name = m_last_name;
        ELSE
           SELECT m.member_id
            INTO m_id
            FROM people p
            INNER JOIN members m
            ON p.person_id = m.person_id
            WHERE p.last_name = m_last_name AND p.first_name = m_first_name;
        END IF;
-- in cursorul all_eligible_members_cursor sunt salvati toti membrii
care sunt eligili pentru un interviu
-- cautam daca membrul dat ca parametru se afla in aceasta lista
        FOR i IN all_eligible_members_cursor LOOP
            IF i.member_id = m_id THEN
               status := 1;
            END IF;
        END LOOP;
        DBMS_OUTPUT.PUT(m_last_name || ' ');
        IF m_first_name IS NOT NULL THEN
            DBMS_OUTPUT.PUT(m_first_name || ' ');
        END IF;
        IF status = 0 THEN
            DBMS_OUTPUT.PUT('NU ');
```

```
        END IF;
        DBMS_OUTPUT.PUT('indeplineste toate criteriile pentru interview');
        DBMS_OUTPUT.NEW_LINE;
        EXCEPTION
            WHEN NO_DATA_FOUND THEN
                record_error();
                DBMS_OUTPUT.PUT_LINE('Nu exista niciun client cu numele de f
amilie ' || m_last_name);
            WHEN TOO_MANY_ROWS THEN
                record_error();
                DBMS_OUTPUT.PUT_LINE('Exista mai multi clienti cu numele de
familie ' || m_last_name);
                DBMS_OUTPUT.PUT_LINE('In acest caz functia trebuie apelata i
mrepuna cu prenume clientului');
                FOR i IN get_member_first_name_cursor(m_last_name) LOOP
                    DBMS_OUTPUT.PUT_LINE(i.last_name || ' ' || i.first_name)
;
                END LOOP;
    END EXERCISE_8;
    PROCEDURE RECORD_ERROR
    IS
       PRAGMA AUTONOMOUS_TRANSACTION;
       id INTEGER;
       code INTEGER := SQLCODE;
    BEGIN
        SELECT MAX(id)
                INTO id
                FROM error_logs;
                IF id IS NULL THEN
                    id := 0;
                END IF;
       INSERT INTO error_logs
       VALUES (id+1, CODE, DBMS_UTILITY.FORMAT_ERROR_STACK, DBMS_UTILITY.FOR
MAT_ERROR_BACKTRACE, DBMS_UTILITY.FORMAT_CALL_STACK, SYSDATE, USER);
       COMMIT;
    END record_error;
END Exercise13;
/
```

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).

Pachetul de mai jos are ca drept scop facilitarea trimiterii unor anunțuri atât clienților cât și instructorilor. Aceste anunțuri sunt programate să fie trimise pe email la o anumită dată.

```
CREATE TABLE announcements(
    announcement_id NUMBER(5) PRIMARY KEY,
    title VARCHAR(25),
    message VARCHAR(1000),
    category VARCHAR2(30),
    send_date DATE,
    sent NUMBER(1) DEFAULT(0)
);
/
CREATE OR REPLACE PACKAGE EXERCISE_14
    AS
    TYPE t_announcements IS TABLE OF announcements%rowtype INDEX BY PLS_INTE
GER;
    valid_announcements t_announcements;
    TYPE vector IS VARRAY(100) OF NUMBER(5);
    active_members vector;

    PROCEDURE add_message(title announcements.title%type, message announceme
nts.message%type, category announcements.category%type, send_date announceme
nts.send_date%type);
    FUNCTION get_active_members return vector;
    FUNCTION get_valid_announcements return t_announcements;
    PROCEDURE send_messages;
END Exercise14;/CREATE OR REPLACE PACKAGE BODY Exercise14
    AS
    PROCEDURE add_message(title announcements.title%type, message announceme
nts.message%type, category announcements.category%type, send_date announceme
nts.send_date%type)
    AS
    a_id announcements.announcement_id%type;
    BEGIN
        IF LENGTH(title) < 5 OR LENGTH(title) > 20 THEN
                RAISE_APPLICATION_ERROR(-
20004, 'Titlul trebuie să aiba o lungime cuprinsa intr 5 si 20 caractere');
        ELSIF LENGTH(message) <= 30 THEN
            RAISE_APPLICATION_ERROR(-
20005, 'Continutul mesajului trebuie să aibă minim 30 caractere');
        ELSIF category <> 'members' AND category <> 'trainers' AND category
<> 'all' THEN
```

```
            RAISE_APPLICATION_ERROR(-
20005, 'Categoria data este incorecta. Alege o categorie dintre: members, tr
ainers sau all');
        ELSIF send_date < sysdate THEN
            RAISE_APPLICATION_ERROR(-
20007, 'Nu poti programa un anunt pentru o data din trecut');
        END IF;

        SELECT NVL(MAX(announcement_id), 0)
        INTO a_id
        FROM announcements;

        INSERT INTO announcements
        VALUES(a_id+1, title, message, category, send_date, 0);
        DBMS_OUTPUT.PUT_LINE('Anuntul a fost adaugat cu succes');
    END add_message;

    FUNCTION get_active_members
    RETURN vector
    AS
        -
- un membru este considerat activ dacă are un abonament valabil, sau dacă ul
timul său abonament a expirat acum maxim o lună
        CURSOR c IS
            SELECT ms.member_id
            FROM members m
            INNER JOIN memberships ms
            ON m.member_id = ms.member_id
            WHERE ADD_MONTHS(ms.END_DATE, 1) >= SYSDATE
            GROUP BY ms.member_id; -
- fac group by pentru cazul in care are mai multe abonamente active
    BEGIN
        OPEN c;
        FETCH c BULK COLLECT INTO active_members;
        CLOSE c;
        return active_members;
    END get_active_members;

    FUNCTION get_valid_announcements
    RETURN t_announcements
    AS
    BEGIN
        SELECT *
        BULK COLLECT INTO valid_announcements
        FROM announcements
        WHERE TRUNC(send_date) = TRUNC(sysdate) AND sent = 0;

        return valid_announcements;
    END get_valid_announcements;
```

```
    PROCEDURE send_messages
    AS
    t_id trainers.trainer_id%type;
    email people.email%type;
    first_name people.first_name%type;
    last_name people.last_name%type;
    CURSOR trainers_cursor IS
        SELECT *
        FROM trainers t
        INNER JOIN people p
        ON p.person_id = t.person_id;
    BEGIN
        active_members := get_active_members();
        valid_announcements := get_valid_announcements();
        FOR i IN valid_announcements.FIRST..valid_announcements.LAST LOOP
            IF valid_announcements(i).category = 'trainers' OR valid_announc
ements(i).category = 'all' THEN
                DBMS_OUTPUT.PUT_LINE('Urmatoarele persoane vor primii anuntu
l cu id-ul ' || valid_announcements(i).announcement_id);
                FOR trainer IN trainers_cursor LOOP
                    apex_mail.send('nistorgeorge666@gmail.com', trainer.emai
l, valid_announcements(i).title, valid_announcements(i).message);
                    DBMS_OUTPUT.PUT_LINE(trainer.first_name || ' ' || traine
r.last_name || ': ' || trainer.email);
                END LOOP;
            ELSIF valid_announcements(i).category = 'members' OR valid_annou
ncements(i).category = 'all' THEN
                FOR j IN active_members.FIRST..active_members.LAST LOOP
                    SELECT p.first_name, p.last_name, p.email
                    INTO first_name, last_name, email
                    FROM members m
                    INNER JOIN people p
                    ON p.person_id = m.person_id
                    WHERE m.member_id = active_members(j);
                    apex_mail.send('nistorgeorge666@gmail.com', email, valid
_announcements(i).title, valid_announcements(i).message);
                    DBMS_OUTPUT.PUT_LINE(first_name || ' ' || last_name || '
: ' || email);
                END LOOP;
            END IF;
            UPDATE announcements
            SET sent = 1
            WHERE announcement_id = valid_announcements(i).announcement_id;
        END LOOP;
    END;
END EXERCISE_14;
/
```

```
BEGIN
    exercise14.add_message('La multi ani', 'Anul Nou este o ocazie perfecta
pentru a ne propune noi obiective si a ne seta noi metode de a ne indeplini
visele. Sper ca in acest an sa ne incurajam unii pe altii sa fim mai buni si
 sa ne atingem potentialul maxim. La Columbia Fitness, ne dorim sa fim parte
nerii dvs. in atingerea obiectivelor de fitness si sanatate si sa va oferim
sprijinul necesar pentru a va mentine motivatia pe parcursul anului. Haideti
 sa incepem Anul Nou cu determinare si sa ne bucuram de beneficiile exerciti
ilor fizice impreuna! Va asteptam sa ne faceti o vizita si sa incepem acest
an cu pasi mai energici si mai sanatosi!', 'all', sysdate);
END;
/

SELECT * FROM announcements;BEGIN
    exercise14.send_messages();
END;
/
```

Output tratare erori:

```
131 BEGIN
132     exercise14.add_message('-', 'Anul Nou este o ocazie perfecta pentru a
133 END;
134 /
135
136 SELECT * FROM announcements;
```

Script Output | Query Result

Task completed in 0,079 seconds

```
Error starting at line : 131 in command -
BEGIN
    exercise14.add_message('-', 'Anul Nou este o ocazie perfecta pentru a ne pro
END;
Error report -
ORA-20004: Titlul trebuie să aiba o lungime cuprinsa intr 5 si 20 caractere
ORA-06512: at "GEORGE.EXERCISE14", line 8
ORA-06512: at line 2
```

```
131 BEGIN
132     exercise14.add_message('La multi ani', '...', 'all', sysdate);
133 END;
134 /
135
136 SELECT * FROM announcements;
```

Script Output | Query Result

Task completed in 0,059 seconds

```
Error starting at line : 131 in command -
BEGIN
    exercise14.add_message('La multi ani', '...', 'all', sysdate);
END;
Error report -
ORA-20005: Continutul mesajului trebuie să aibă minim 30 caractere
ORA-06512: at "GEORGE.EXERCISE14", line 10
ORA-06512: at line 2
```

```
139 BEGIN
140     exercise14.add_message('La multi ani',
141                            'Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Aenean
142                            'test',
143                             sysdate);
144 END;
145 /
```

Script Output | Query Result

Task completed in 0,057 seconds

```
    exercise14.add_message('La multi ani',
                           'Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Aenean commoc
                           'test',
                            sysdate);
END;
Error report -
ORA-20005: Categoria data este incorecta. Alege o categorie dintre: members, trainers sau all
ORA-06512: at "GEORGE.EXERCISE14", line 12
ORA-06512: at line 2
```

```
139 BEGIN
140     exercise14.add_message('La multi ani',
141                            'Lorem ipsum dolor sit amet, consectetuer a
142                            'members',
143                             TO_DATE('05-01-2023', 'DD-MM-YYYY'));
144 END;
145 /
```

Script Output | Query Result

Task completed in 0,059 seconds

```
    exercise14.add_message('La multi ani',
                           'Lorem ipsum dolor sit amet, consectetuer adipisc
                           'members',
                            TO_DATE('05-01-2023', 'DD-MM-YYYY'));
END;
Error report -
ORA-20007: Nu poti programa un anunt pentru o data din trecut
ORA-06512: at "GEORGE.EXERCISE14", line 14
ORA-06512: at line 2
```

## Output pentru rulare cu succes:

```
121 BEGIN
122     exercise14.add_message('La mulți ani', 'Anul Nou este o ocazie perfectă pentru a ne prop
123 END;
124 /
125
126 SELECT * FROM announcements;
```

Script Output | Query Result | Query Result 1 | Query Result 2

SQL | All Rows Fetched: 1 in 0,008 seconds

| | ANNOUNCEMENT_ID | TITLE | MESSAGE | CATEGORY | SEND_DATE | SENT |
|---|---|---|---|---|---|---|
| 1 | 1 | La mulţ... | Anul Nou este o ocazie perfect... | all | 13-01-2023 | 0 |

Dbms Output | Buffer Size: 20000

George

```
Anuntul a fost adaugat cu succes
```

```
126 BEGIN
127     exercise14.send_messages();
128 END;
129 /
```

Script Output | Task completed in 6,38 seconds

```
PL/SQL procedure successfully completed.


PL/SQL procedure successfully completed.
```

Dbms Output | Buffer Size: 20000

George

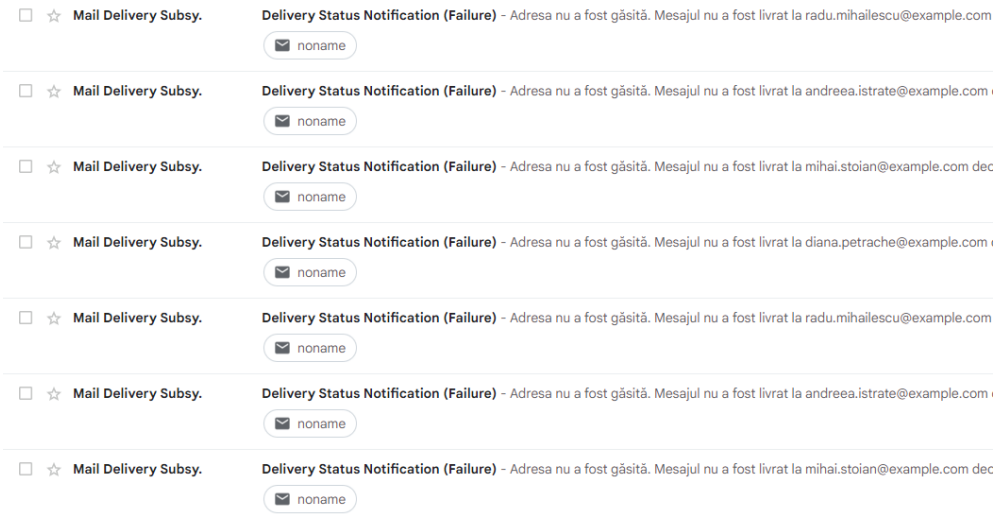
```
Anuntul a fost adaugat cu succes

Urmatoarele persoane vor primii anuntul cu id-ul 1
Ioan Petrescu: nistorgeorge404+trainer@gmail.com
Mihai Petrescu: mihai.stoian@example.com
Andreea Istrate: andreea.istrate@example.com
Radu Mihailescu: radu.mihailescu@example.com
Diana Petrache: diana.petrache@example.com
```

La multi ani Inbox

nistorgeorge666@gmail.com 9:21 PM (1 minute ago)
to nistorgeorge404+trainer

Anul Nou este o ocazie perfecta pentru a ne propune noi obiective si a ne seta noi metode de a ne indeplini visele. Sper ca in acest an sa ne incurajam unii pe altii sa fim mai buni si sa ne atingem potentialul maxim. La Columbia Fitness, ne dorim sa fim partenerii dvs. in atingerea obiectivelor de fitness si sanatate si sa va oferim sprijinul necesar pentru a va mentine motivatia pe parcursul anului. Haideti sa incepem Anul Nou cu determinare si sa ne bucuram de beneficiile exercitiilor fizice impreuna! Va asteptam sa ne faceti o vizita si sa incepem acest an cu pasi mai energici si mai sanatosi!




(email-urile din baza de date sun invalide)

15. APEX_MAIL (procedură nu îmi aparține, ea a fost preluată de aici)

```
CREATE OR replace PACKAGE apex_mail
IS
  g_smtp_host   VARCHAR2 (256) := 'localhost';
  g_smtp_port   PLS_INTEGER := 1925;
  g_smtp_domain VARCHAR2 (256) := 'gmail.com';
  g_mailer_id CONSTANT VARCHAR2 (256) := 'Mailer by Oracle UTL_SMTP';
  -- send mail using UTL_SMTP
  PROCEDURE send (
    p_sender    IN VARCHAR2,
    p_recipient IN VARCHAR2,
    p_subject   IN VARCHAR2,
    p_message   IN VARCHAR2 );
END;
/
CREATE OR replace PACKAGE BODY apex_mail
IS
  -- Write a MIME header
  PROCEDURE Write_mime_header (p_conn  IN OUT nocopy utl_smtp.connection,
                               p_name  IN VARCHAR2,
                               p_value IN VARCHAR2)
  IS
  BEGIN
      utl_smtp.Write_data (p_conn, p_name
                                   || ': '
                                   || p_value
                                   || utl_tcp.crlf);
  END;
  PROCEDURE Send (p_sender    IN VARCHAR2,
                  p_recipient IN VARCHAR2,
                  p_subject   IN VARCHAR2,
                  p_message   IN VARCHAR2)
  IS
    l_conn      utl_smtp.connection;
    nls_charset VARCHAR2(255);
  BEGIN
      -- get characterset
      SELECT value
      INTO   nls_charset
      FROM   nls_database_parameters
      WHERE  parameter = 'NLS_CHARACTERSET';
      -- establish connection and autheticate
      l_conn := utl_smtp.Open_connection (g_smtp_host, g_smtp_port);
      utl_smtp.Ehlo(l_conn, g_smtp_domain);
      utl_smtp.Command(l_conn, 'auth login');
      utl_smtp.Command(l_conn,
      utl_encode.Text_encode('user@gmail.com',
```

```
      nls_charset, 1));
      utl_smtp.Command(l_conn, utl_encode.Text_encode('parola',
                               nls_charset,
                               1));
      -- set from/recipient
      utl_smtp.Command(l_conn, 'MAIL FROM: <'
                               ||p_sender
                               ||'>');

      utl_smtp.Command(l_conn, 'RCPT TO: <'
                               ||p_recipient
                               ||'>');
      -- write mime headers
      utl_smtp.Open_data (l_conn);
      Write_mime_header (l_conn, 'From', p_sender);
      Write_mime_header (l_conn, 'To', p_recipient);
      Write_mime_header (l_conn, 'Subject', p_subject);
      Write_mime_header (l_conn, 'Content-Type', 'text/html');
      Write_mime_header (l_conn, 'X-Mailer', g_mailer_id);
      utl_smtp.Write_data (l_conn, utl_tcp.crlf);
      -- write message body
      utl_smtp.Write_data (l_conn, p_message);
      utl_smtp.Close_data (l_conn);
      -- end connection
      utl_smtp.Quit (l_conn);
  EXCEPTION
    WHEN OTHERS THEN
               BEGIN
                   utl_smtp.Quit(l_conn);
               EXCEPTION
                   WHEN OTHERS THEN
                     NULL;
               END;
               Raise_application_error(-20000,
               'Failed to send mail due to the following error: '
               || SQLERRM);
  END;
END;
/
```