Managementul scolilor de soferi din Romania

Dumitrescu Delia Ioana

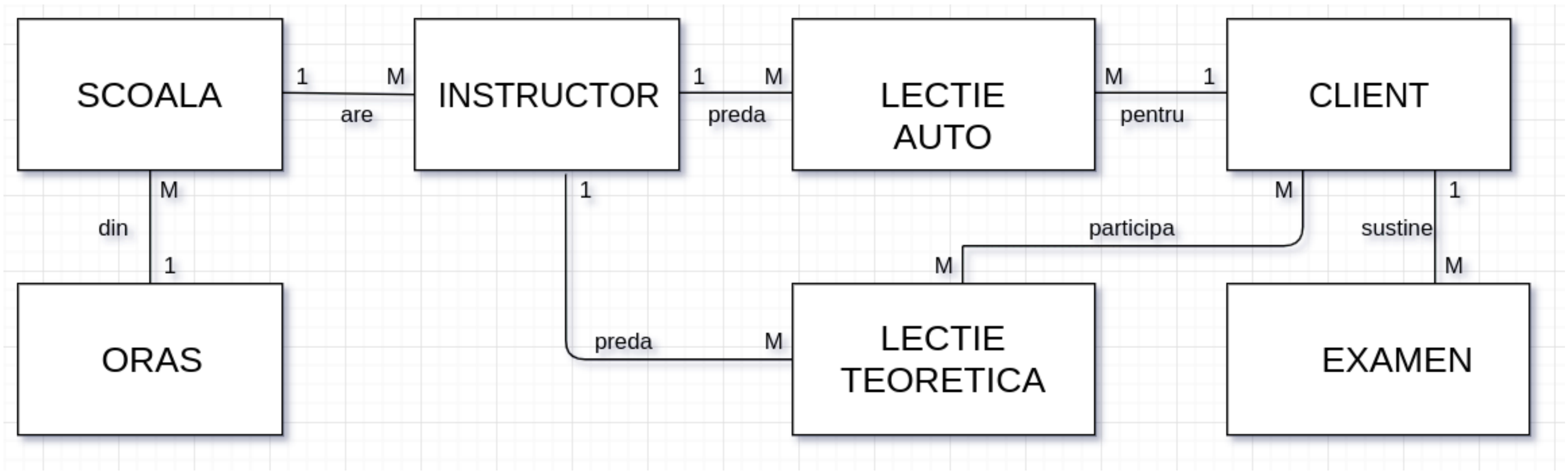
Proiect - Sisteme de Gestiune a Bazelor de Date

Exercitiul 1

Scolile de soferi din Romania tind sa fie, uneori, usor neorganizate. Deoarece nu exista un sistem bine pus la punct pentru managementul acestor institutii, apar probleme de comunicare. Instructorii pot pierde din vedere numarul de sedinte efectuat de fiecare client in parte, clientii pot uita de lectiile teoretice la care trebuie sa participe si exemplele continua. Consider ca, in acest caz, o baza de date ar fi ideala pentru a tine evidenta tuturor datelor. Astfel, ar exista o desfasurare mult mai eficienta a lucrurilor si o experienta mai placuta per total, atat pentru clienti, cat si pentru instructori.

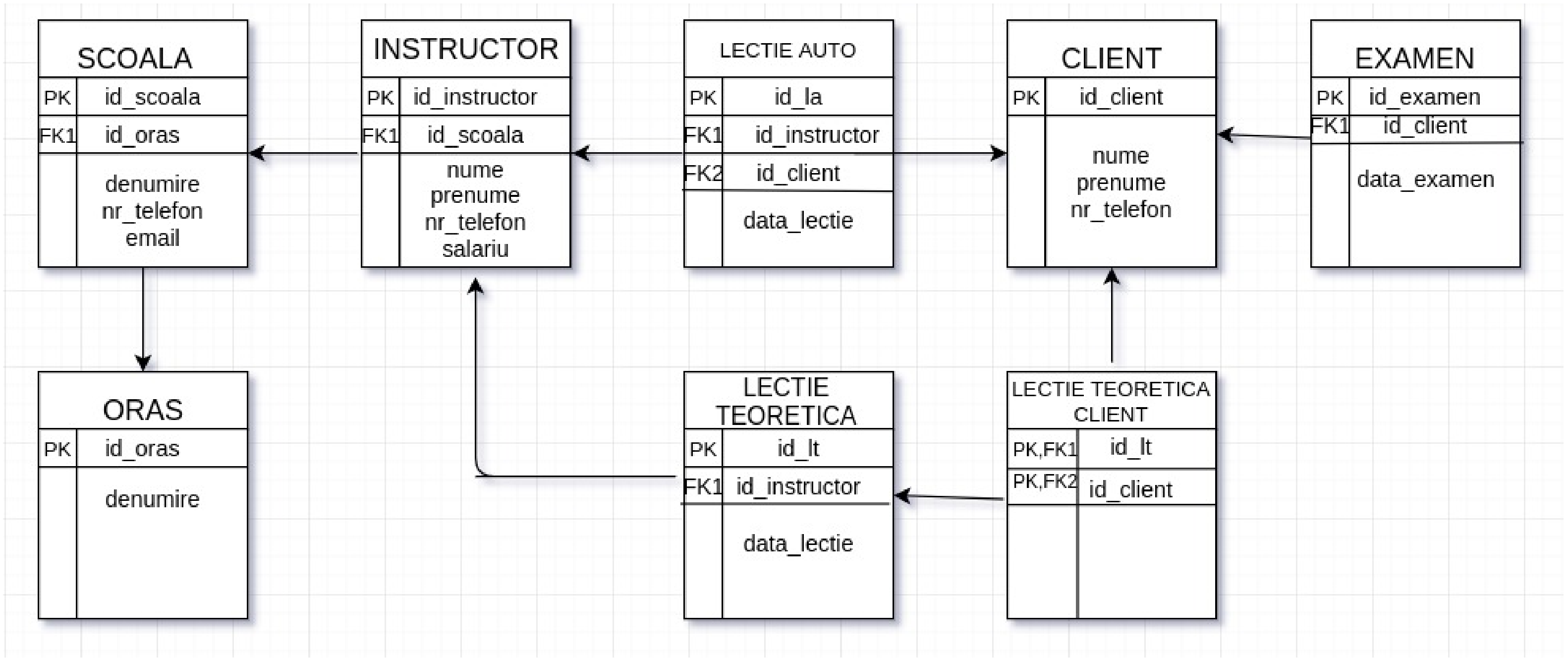
Exercitiul 2

Diagrama entitate-relatie(ERD):



Exercitiul 3

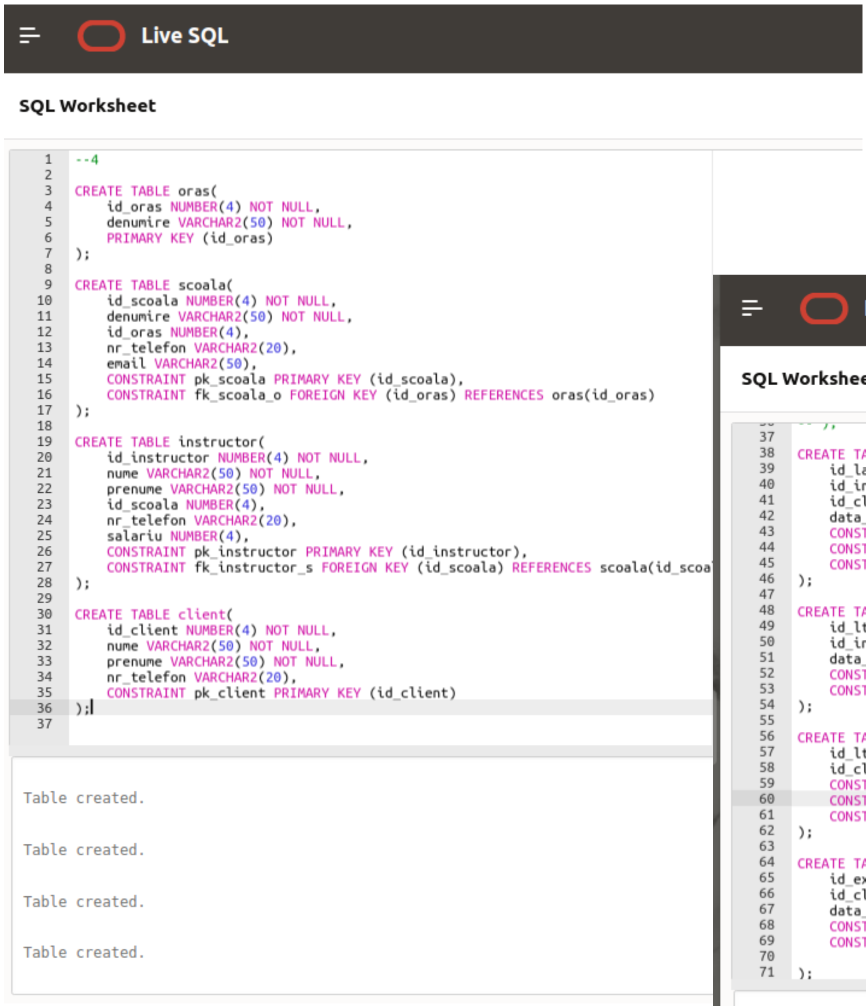
Diagrama conceptuala:



Exercitiul 4

Crearea tabelelor:

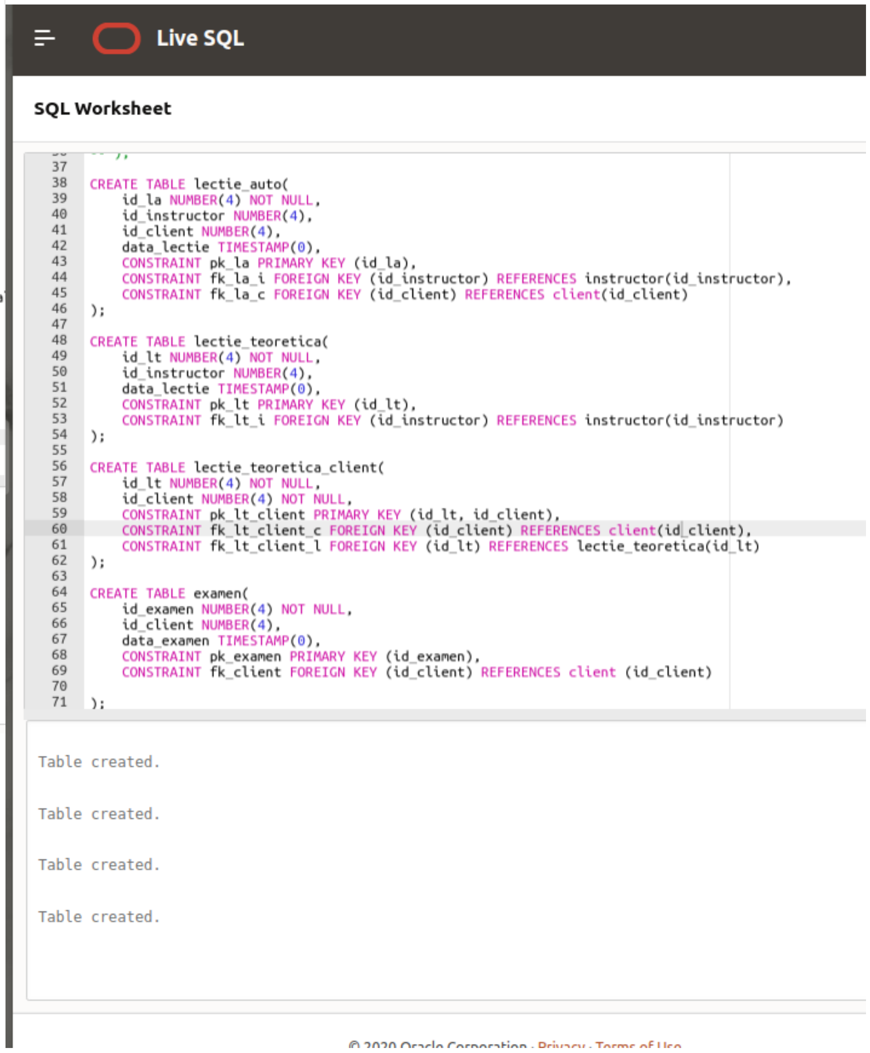
```
CREATE TABLE oras(
    id_oras NUMBER(4) NOT NULL,
    denumire VARCHAR2(50) NOT NULL,
    PRIMARY KEY (id_oras)
);

CREATE TABLE scoala(
    id_scoala NUMBER(4) NOT NULL,
    denumire VARCHAR2(50) NOT NULL,
    id_oras NUMBER(4),
    nr_telefon VARCHAR2(20),
    email VARCHAR2(50),
    CONSTRAINT pk_scoala PRIMARY KEY (id_scoala),
    CONSTRAINT fk_scoala_o FOREIGN KEY (id_oras) REFERENCES oras(id_oras)
);

CREATE TABLE instructor(
    id_instructor NUMBER(4) NOT NULL,
    nume VARCHAR2(50) NOT NULL,
    prenume VARCHAR2(50) NOT NULL,
    id_scoala NUMBER(4),
    nr_telefon VARCHAR2(20),
    salariu NUMBER(4),
    CONSTRAINT pk_instructor PRIMARY KEY (id_instructor),
    CONSTRAINT fk_instructor_s FOREIGN KEY (id_scoala) REFERENCES scoala(id_scoala)
);

CREATE TABLE client(
    id_client NUMBER(4) NOT NULL,
    nume VARCHAR2(50) NOT NULL,
    prenume VARCHAR2(50) NOT NULL,
    nr_telefon VARCHAR2(20),
    CONSTRAINT pk_client PRIMARY KEY (id_client)
);

CREATE TABLE lectie_auto(
    id_la NUMBER(4) NOT NULL,
    id_instructor NUMBER(4),
    id_client NUMBER(4),
    data_lectie TIMESTAMP(0),
    CONSTRAINT pk_la PRIMARY KEY (id_la),
    CONSTRAINT fk_la_i FOREIGN KEY (id_instructor) REFERENCES instructor(id_instructor),
    CONSTRAINT fk_la_c FOREIGN KEY (id_client) REFERENCES client(id_client)
);

CREATE TABLE lectie_teoretica(
    id_lt NUMBER(4) NOT NULL,
    id_instructor NUMBER(4),
    data_lectie TIMESTAMP(0),
    CONSTRAINT pk_lt PRIMARY KEY (id_lt),
    CONSTRAINT fk_lt_i FOREIGN KEY (id_instructor) REFERENCES instructor(id_instructor)
);

CREATE TABLE lectie_teoretica_client(
    id_lt NUMBER(4) NOT NULL,
    id_client NUMBER(4) NOT NULL,
    CONSTRAINT pk_lt_client PRIMARY KEY (id_lt, id_client),
    CONSTRAINT fk_lt_client_c FOREIGN KEY (id_client) REFERENCES client(id_client),
    CONSTRAINT fk_lt_client_l FOREIGN KEY (id_lt) REFERENCES lectie_teoretica(id_lt)
);

CREATE TABLE examen(
    id_examen NUMBER(4) NOT NULL,
    id_client NUMBER(4),
    data_examen TIMESTAMP(0),
    CONSTRAINT pk_examen PRIMARY KEY (id_examen),
    CONSTRAINT fk_client FOREIGN KEY (id_client) REFERENCES client (id_client)
);
```



SQL Worksheet

```
--4

CREATE TABLE oras(
    id_oras NUMBER(4) NOT NULL,
    denumire VARCHAR2(50) NOT NULL,
    PRIMARY KEY (id_oras)
);

CREATE TABLE scoala(
    id_scoala NUMBER(4) NOT NULL,
    denumire VARCHAR2(50) NOT NULL,
    id_oras NUMBER(4),
    nr_telefon VARCHAR2(20),
    email VARCHAR2(50),
    CONSTRAINT pk_scoala PRIMARY KEY (id_scoala),
    CONSTRAINT fk_scoala_o FOREIGN KEY (id_oras) REFERENCES oras(id_oras)
);

CREATE TABLE instructor(
    id_instructor NUMBER(4) NOT NULL,
    nume VARCHAR2(50) NOT NULL,
    prenume VARCHAR2(50) NOT NULL,
    id_scoala NUMBER(4),
    nr_telefon VARCHAR2(20),
    salariu NUMBER(4),
    CONSTRAINT pk_instructor PRIMARY KEY (id_instructor),
    CONSTRAINT fk_instructor_s FOREIGN KEY (id_scoala) REFERENCES scoala(id_scoa
);

CREATE TABLE client(
    id_client NUMBER(4) NOT NULL,
    nume VARCHAR2(50) NOT NULL,
    prenume VARCHAR2(50) NOT NULL,
    nr_telefon VARCHAR2(20),
    CONSTRAINT pk_client PRIMARY KEY (id_client)
);

```

```
Table created.

Table created.

Table created.

Table created.
```



SQL Worksheet

```

CREATE TABLE lectie_auto(
    id_la NUMBER(4) NOT NULL,
    id_instructor NUMBER(4),
    id_client NUMBER(4),
    data_lectie TIMESTAMP(0),
    CONSTRAINT pk_la PRIMARY KEY (id_la),
    CONSTRAINT fk_la_i FOREIGN KEY (id_instructor) REFERENCES instructor(id_instructor),
    CONSTRAINT fk_la_c FOREIGN KEY (id_client) REFERENCES client(id_client)
);

CREATE TABLE lectie_teoretica(
    id_lt NUMBER(4) NOT NULL,
    id_instructor NUMBER(4),
    data_lectie TIMESTAMP(0),
    CONSTRAINT pk_lt PRIMARY KEY (id_lt),
    CONSTRAINT fk_lt_i FOREIGN KEY (id_instructor) REFERENCES instructor(id_instructor)
);

CREATE TABLE lectie_teoretica_client(
    id_lt NUMBER(4) NOT NULL,
    id_client NUMBER(4) NOT NULL,
    CONSTRAINT pk_lt_client PRIMARY KEY (id_lt, id_client),
    CONSTRAINT fk_lt_client_c FOREIGN KEY (id_client) REFERENCES client(id_client),
    CONSTRAINT fk_lt_client_l FOREIGN KEY (id_lt) REFERENCES lectie_teoretica(id_lt)
);

CREATE TABLE examen(
    id_examen NUMBER(4) NOT NULL,
    id_client NUMBER(4),
    data_examen TIMESTAMP(0),
    CONSTRAINT pk_examen PRIMARY KEY (id_examen),
    CONSTRAINT fk_client FOREIGN KEY (id_client) REFERENCES client (id_client)

);
```

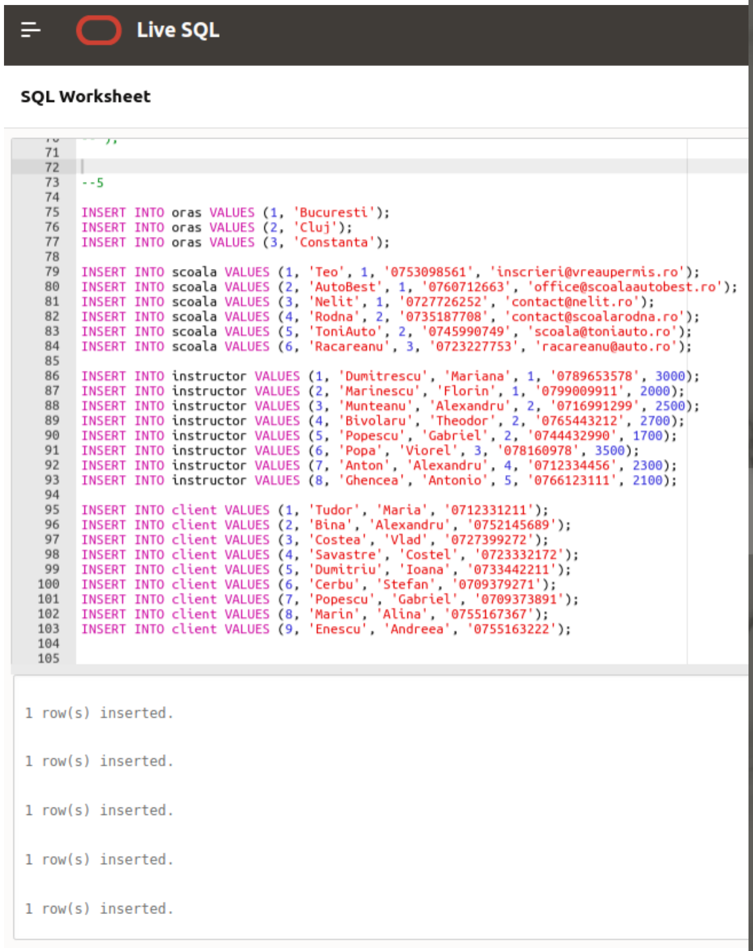
```
Table created.

Table created.

Table created.

Table created.
```

© 2020 Oracle Corporation · Privacy · Terms of Use

Exercitiul 5

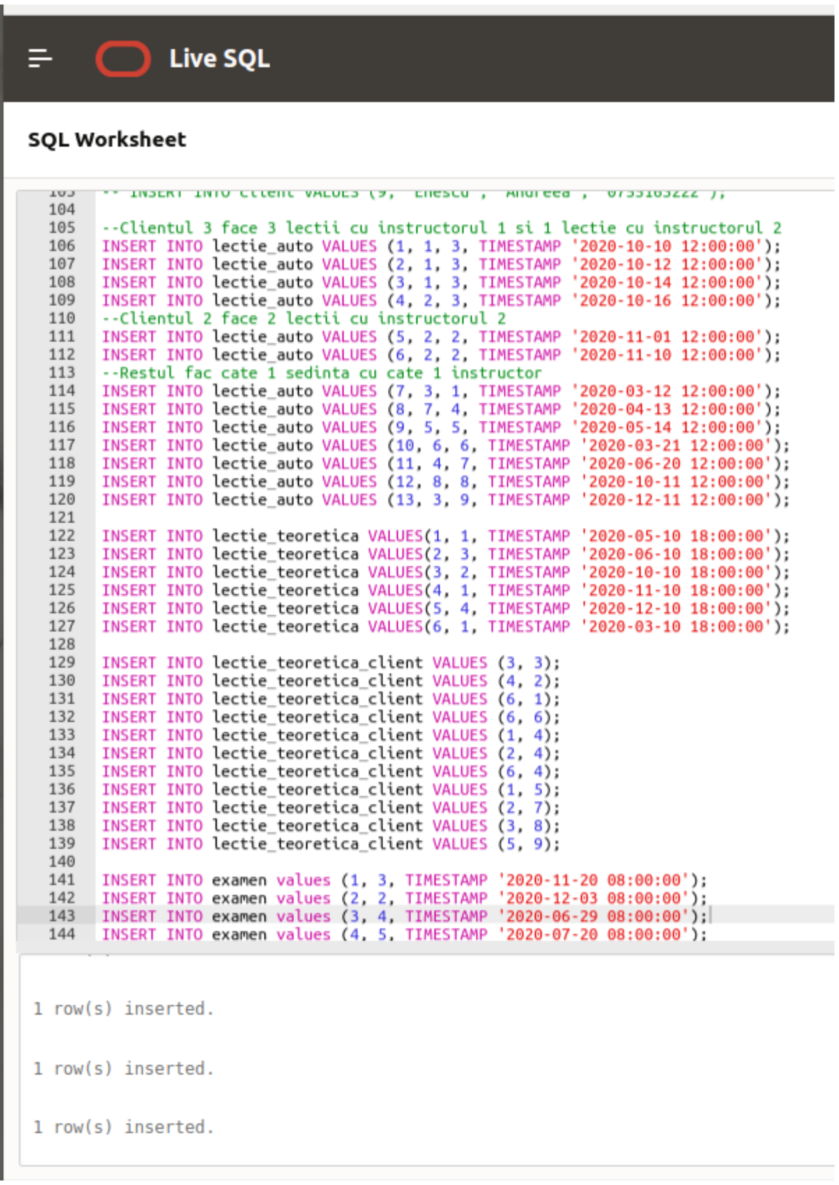
Popularea tabelelor:

```sql
INSERT INTO oras VALUES (1, 'Bucuresti');
INSERT INTO oras VALUES (2, 'Cluj');
INSERT INTO oras VALUES (3, 'Constanta');

INSERT INTO scoala VALUES (1, 'Teo', 1, '0753098561', 'inscrieri@vreaupermis.ro');
INSERT INTO scoala VALUES (2, 'AutoBest', 1, '0760712663', 'office@scoalaautobest.ro');
INSERT INTO scoala VALUES (3, 'Nelit', 1, '0727726252', 'contact@nelit.ro');
INSERT INTO scoala VALUES (4, 'Rodna', 2, '0735187708', 'contact@scoalarodna.ro');
INSERT INTO scoala VALUES (5, 'ToniAuto', 2, '0745990749', 'scoala@toniauto.ro');
INSERT INTO scoala VALUES (6, 'Racareanu', 3, '0723227753', 'racareanu@auto.ro');

INSERT INTO instructor VALUES (1, 'Dumitrescu', 'Mariana', 1, '0789653578', 3000);
INSERT INTO instructor VALUES (2, 'Marinescu', 'Florin', 1, '0799009911', 2000);
INSERT INTO instructor VALUES (3, 'Munteanu', 'Alexandru', 2, '0716991299', 2500);
INSERT INTO instructor VALUES (4, 'Bivolaru', 'Theodor', 2, '0765443212', 2700);
INSERT INTO instructor VALUES (5, 'Popescu', 'Gabriel', 2, '0744432990', 1700);
INSERT INTO instructor VALUES (6, 'Popa', 'Viorel', 3, '078160978', 3500);
INSERT INTO instructor VALUES (7, 'Anton', 'Alexandru', 4, '0712334456', 2300);
INSERT INTO instructor VALUES (8, 'Ghencea', 'Antonio', 5, '0766123111', 2100);

INSERT INTO client VALUES (1, 'Tudor', 'Maria', '0712331211');
INSERT INTO client VALUES (2, 'Bina', 'Alexandru', '0752145689');
INSERT INTO client VALUES (3, 'Costea', 'Vlad', '0727399272');
INSERT INTO client VALUES (4, 'Savastre', 'Costel', '0723332172');
INSERT INTO client VALUES (5, 'Dumitriu', 'Ioana', '0733442211');
INSERT INTO client VALUES (6, 'Cerbu', 'Stefan', '0709379271');
INSERT INTO client VALUES (7, 'Popescu', 'Gabriel', '0709373891');
INSERT INTO client VALUES (8, 'Marin', 'Alina', '0755167367');
INSERT INTO client VALUES (9, 'Enescu', 'Andreea', '0755163222');

--Clientul 3 face 3 lectii cu instructorul 1 si 1 lectie cu instructorul 2
INSERT INTO lectie_auto VALUES (1, 1, 3, TIMESTAMP '2020-10-10 12:00:00');
INSERT INTO lectie_auto VALUES (2, 1, 3, TIMESTAMP '2020-10-12 12:00:00');
INSERT INTO lectie_auto VALUES (3, 1, 3, TIMESTAMP '2020-10-14 12:00:00');
INSERT INTO lectie_auto VALUES (4, 2, 3, TIMESTAMP '2020-10-16 12:00:00');
--Clientul 2 face 2 lectii cu instructorul 2
INSERT INTO lectie_auto VALUES (5, 2, 2, TIMESTAMP '2020-11-01 12:00:00');
INSERT INTO lectie_auto VALUES (6, 2, 2, TIMESTAMP '2020-11-10 12:00:00');
--Restul fac cate 1 sedinta cu cate 1 instructor
INSERT INTO lectie_auto VALUES (7, 3, 1, TIMESTAMP '2020-03-12 12:00:00');
INSERT INTO lectie_auto VALUES (8, 7, 4, TIMESTAMP '2020-04-13 12:00:00');
INSERT INTO lectie_auto VALUES (9, 5, 5, TIMESTAMP '2020-05-14 12:00:00');
INSERT INTO lectie_auto VALUES (10, 6, 6, TIMESTAMP '2020-03-21 12:00:00');
INSERT INTO lectie_auto VALUES (11, 4, 7, TIMESTAMP '2020-06-20 12:00:00');
INSERT INTO lectie_auto VALUES (12, 8, 8, TIMESTAMP '2020-10-11 12:00:00');
INSERT INTO lectie_auto VALUES (13, 3, 9, TIMESTAMP '2020-12-11 12:00:00');

INSERT INTO lectie_teoretica VALUES(1, 1, TIMESTAMP '2020-05-10 18:00:00');
INSERT INTO lectie_teoretica VALUES(2, 3, TIMESTAMP '2020-06-10 18:00:00');
INSERT INTO lectie_teoretica VALUES(3, 2, TIMESTAMP '2020-10-10 18:00:00');
INSERT INTO lectie_teoretica VALUES(4, 1, TIMESTAMP '2020-11-10 18:00:00');
INSERT INTO lectie_teoretica VALUES(5, 4, TIMESTAMP '2020-12-10 18:00:00');
INSERT INTO lectie_teoretica VALUES(6, 1, TIMESTAMP '2020-03-10 18:00:00');

INSERT INTO lectie_teoretica_client VALUES (3, 3);
INSERT INTO lectie_teoretica_client VALUES (4, 2);
INSERT INTO lectie_teoretica_client VALUES (6, 1);
INSERT INTO lectie_teoretica_client VALUES (6, 6);
INSERT INTO lectie_teoretica_client VALUES (1, 4);
INSERT INTO lectie_teoretica_client VALUES (2, 4);
INSERT INTO lectie_teoretica_client VALUES (6, 4);
INSERT INTO lectie_teoretica_client VALUES (1, 5);
INSERT INTO lectie_teoretica_client VALUES (2, 7);
INSERT INTO lectie_teoretica_client VALUES (3, 8);
INSERT INTO lectie_teoretica_client VALUES (5, 9);

INSERT INTO examen values (1, 3, TIMESTAMP '2020-11-20 08:00:00');
INSERT INTO examen values (2, 2, TIMESTAMP '2020-12-03 08:00:00');
INSERT INTO examen values (3, 4, TIMESTAMP '2020-06-29 08:00:00');
INSERT INTO examen values (4, 5, TIMESTAMP '2020-07-20 08:00:00');
```



```sql
--5

INSERT INTO oras VALUES (1, 'Bucuresti');
INSERT INTO oras VALUES (2, 'Cluj');
INSERT INTO oras VALUES (3, 'Constanta');

INSERT INTO scoala VALUES (1, 'Teo', 1, '0753098561', 'inscrieri@vreaupermis.ro');
INSERT INTO scoala VALUES (2, 'AutoBest', 1, '0760712663', 'office@scoalaautobest.ro');
INSERT INTO scoala VALUES (3, 'Nelit', 1, '0727726252', 'contact@nelit.ro');
INSERT INTO scoala VALUES (4, 'Rodna', 2, '0735187708', 'contact@scoalarodna.ro');
INSERT INTO scoala VALUES (5, 'ToniAuto', 2, '0745990749', 'scoala@toniauto.ro');
INSERT INTO scoala VALUES (6, 'Racareanu', 3, '0723227753', 'racareanu@auto.ro');

INSERT INTO instructor VALUES (1, 'Dumitrescu', 'Mariana', 1, '0789653578', 3000);
INSERT INTO instructor VALUES (2, 'Marinescu', 'Florin', 1, '0799009911', 2000);
INSERT INTO instructor VALUES (3, 'Munteanu', 'Alexandru', 2, '0716991299', 2500);
INSERT INTO instructor VALUES (4, 'Bivolaru', 'Theodor', 2, '0765443212', 2700);
INSERT INTO instructor VALUES (5, 'Popescu', 'Gabriel', 2, '0744432990', 1700);
INSERT INTO instructor VALUES (6, 'Popa', 'Viorel', 3, '078160978', 3500);
INSERT INTO instructor VALUES (7, 'Anton', 'Alexandru', 4, '0712334456', 2300);
INSERT INTO instructor VALUES (8, 'Ghencea', 'Antonio', 5, '0766123111', 2100);

INSERT INTO client VALUES (1, 'Tudor', 'Maria', '0712331211');
INSERT INTO client VALUES (2, 'Bina', 'Alexandru', '0752145689');
INSERT INTO client VALUES (3, 'Costea', 'Vlad', '0727399272');
INSERT INTO client VALUES (4, 'Savastre', 'Costel', '0723332172');
INSERT INTO client VALUES (5, 'Dumitriu', 'Ioana', '0733442211');
INSERT INTO client VALUES (6, 'Cerbu', 'Stefan', '0709379271');
INSERT INTO client VALUES (7, 'Popescu', 'Gabriel', '0709373891');
INSERT INTO client VALUES (8, 'Marin', 'Alina', '0755167367');
INSERT INTO client VALUES (9, 'Enescu', 'Andreea', '0755163222');
```

```
1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.
```



```sql

--Clientul 3 face 3 lectii cu instructorul 1 si 1 lectie cu instructorul 2
INSERT INTO lectie_auto VALUES (1, 1, 3, TIMESTAMP '2020-10-10 12:00:00');
INSERT INTO lectie_auto VALUES (2, 1, 3, TIMESTAMP '2020-10-12 12:00:00');
INSERT INTO lectie_auto VALUES (3, 1, 3, TIMESTAMP '2020-10-14 12:00:00');
INSERT INTO lectie_auto VALUES (4, 2, 3, TIMESTAMP '2020-10-16 12:00:00');
--Clientul 2 face 2 lectii cu instructorul 2
INSERT INTO lectie_auto VALUES (5, 2, 2, TIMESTAMP '2020-11-01 12:00:00');
INSERT INTO lectie_auto VALUES (6, 2, 2, TIMESTAMP '2020-11-10 12:00:00');
--Restul fac cate 1 sedinta cu cate 1 instructor
INSERT INTO lectie_auto VALUES (7, 3, 1, TIMESTAMP '2020-03-12 12:00:00');
INSERT INTO lectie_auto VALUES (8, 7, 4, TIMESTAMP '2020-04-13 12:00:00');
INSERT INTO lectie_auto VALUES (9, 5, 5, TIMESTAMP '2020-05-14 12:00:00');
INSERT INTO lectie_auto VALUES (10, 6, 6, TIMESTAMP '2020-03-21 12:00:00');
INSERT INTO lectie_auto VALUES (11, 4, 7, TIMESTAMP '2020-06-20 12:00:00');
INSERT INTO lectie_auto VALUES (12, 8, 8, TIMESTAMP '2020-10-11 12:00:00');
INSERT INTO lectie_auto VALUES (13, 3, 9, TIMESTAMP '2020-12-11 12:00:00');

INSERT INTO lectie_teoretica VALUES(1, 1, TIMESTAMP '2020-05-10 18:00:00');
INSERT INTO lectie_teoretica VALUES(2, 3, TIMESTAMP '2020-06-10 18:00:00');
INSERT INTO lectie_teoretica VALUES(3, 2, TIMESTAMP '2020-10-10 18:00:00');
INSERT INTO lectie_teoretica VALUES(4, 1, TIMESTAMP '2020-11-10 18:00:00');
INSERT INTO lectie_teoretica VALUES(5, 4, TIMESTAMP '2020-12-10 18:00:00');
INSERT INTO lectie_teoretica VALUES(6, 1, TIMESTAMP '2020-03-10 18:00:00');

INSERT INTO lectie_teoretica_client VALUES (3, 3);
INSERT INTO lectie_teoretica_client VALUES (4, 2);
INSERT INTO lectie_teoretica_client VALUES (6, 1);
INSERT INTO lectie_teoretica_client VALUES (6, 6);
INSERT INTO lectie_teoretica_client VALUES (1, 4);
INSERT INTO lectie_teoretica_client VALUES (2, 4);
INSERT INTO lectie_teoretica_client VALUES (6, 4);
INSERT INTO lectie_teoretica_client VALUES (1, 5);
INSERT INTO lectie_teoretica_client VALUES (2, 7);
INSERT INTO lectie_teoretica_client VALUES (3, 8);
INSERT INTO lectie_teoretica_client VALUES (5, 9);

INSERT INTO examen values (1, 3, TIMESTAMP '2020-11-20 08:00:00');
INSERT INTO examen values (2, 2, TIMESTAMP '2020-12-03 08:00:00');
INSERT INTO examen values (3, 4, TIMESTAMP '2020-06-29 08:00:00');
INSERT INTO examen values (4, 5, TIMESTAMP '2020-07-20 08:00:00');
```

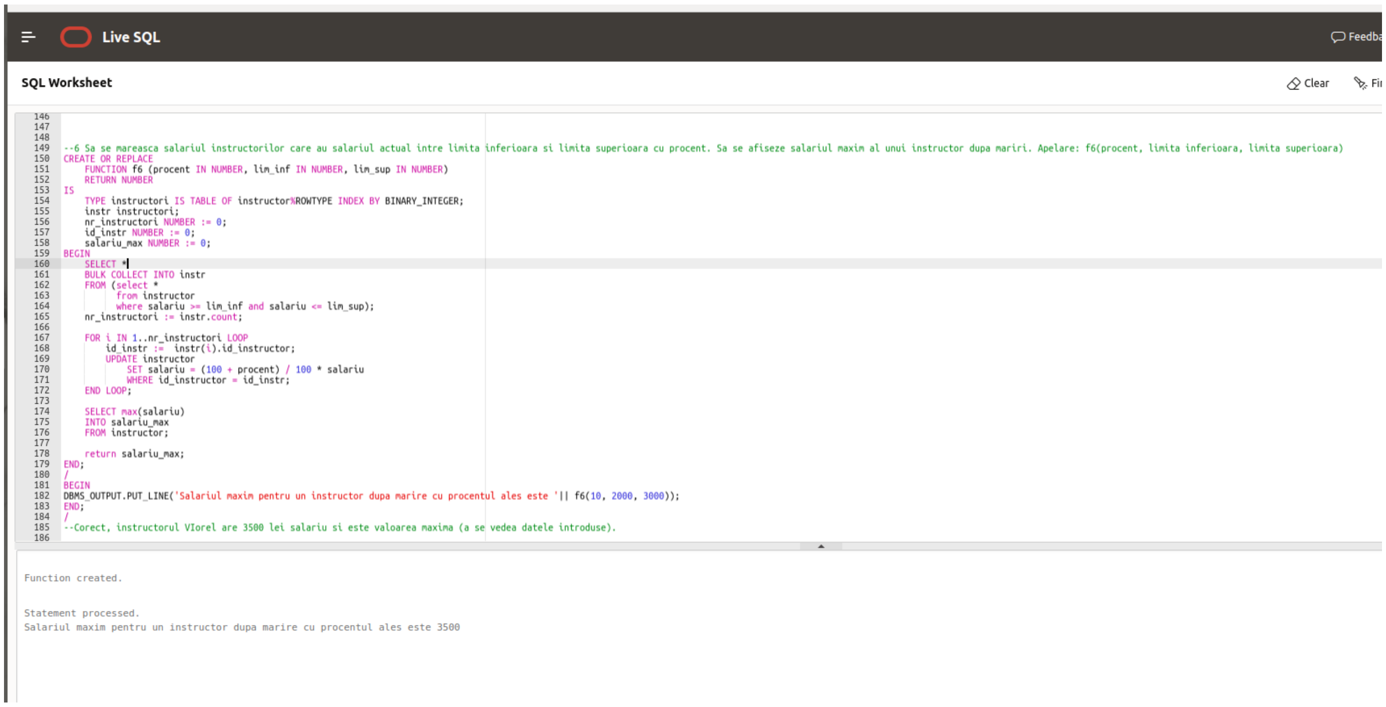
```
1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.
```

Exercitiul 6

Un subprogram stocat care sa utilizeze un tip de colectie studiat:

Sa se mareasca salariul instructorilor care au salariul actual intre o limita inferioara si o limita superioara cu un procent. Sa se afiseze salariul maxim al unui instructor dupa mariri.

```sql
CREATE OR REPLACE
    FUNCTION f6 (procent IN NUMBER, lim_inf IN NUMBER, lim_sup IN NUMBER)
    RETURN NUMBER
IS
    TYPE instructori IS TABLE OF instructor%ROWTYPE INDEX BY BINARY_INTEGER;
    instr instructori;
    nr_instructori NUMBER := 0;
    id_instr NUMBER := 0;
    salariu_max NUMBER := 0;
BEGIN
    SELECT *
    BULK COLLECT INTO instr
    FROM (select *
          from instructor
          where salariu >= lim_inf and salariu <= lim_sup);
    nr_instructori := instr.count;

    FOR i IN 1..nr_instructori LOOP
        id_instr :=  instr(i).id_instructor;
        UPDATE instructor
            SET salariu = (100 + procent) / 100 * salariu
            WHERE id_instructor = id_instr;
    END LOOP;

    SELECT max(salariu)
    INTO salariu_max
    FROM instructor;

    return salariu_max;
END;
/
BEGIN
DBMS_OUTPUT.PUT_LINE('Salariul maxim pentru un instructor dupa marire cu procentul
ales este '|| f6(10, 2000, 3000));
END;
/
```



```
--6 Sa se mareasca salariul instructorilor care au salariul actual intre limita inferioara si limita superioara cu procent. Sa se afiseze salariul maxim al unui instructor dupa mariri. Apelare: f6(procent, limita inferioara, limita superioara)
CREATE OR REPLACE
    FUNCTION f6 (procent IN NUMBER, lim_inf IN NUMBER, lim_sup IN NUMBER)
    RETURN NUMBER
IS
    TYPE instructori IS TABLE OF instructor%ROWTYPE INDEX BY BINARY_INTEGER;
    instr instructori;
    nr_instructori NUMBER := 0;
    id_instr NUMBER := 0;
    salariu_max NUMBER := 0;
BEGIN
    SELECT *
    BULK COLLECT INTO instr
    FROM (select *
            from instructor
            where salariu >= lim_inf and salariu <= lim_sup);
    nr_instructori := instr.count;

    FOR i IN 1..nr_instructori LOOP
        id_instr :=  instr(i).id_instructor;
        UPDATE instructor
            SET salariu = (100 + procent) / 100 * salariu
            WHERE id_instructor = id_instr;
    END LOOP;

    SELECT max(salariu)
    INTO salariu_max
    FROM instructor;

    return salariu_max;
END;
/
BEGIN
DBMS_OUTPUT.PUT_LINE('Salariul maxim pentru un instructor dupa marire cu procentul ales este '|| f6(10, 2000, 3000));
END;
/
--Corect, instructorul VIorel are 3500 lei salariu si este valoarea maxima (a se vedea datele introduse).
```

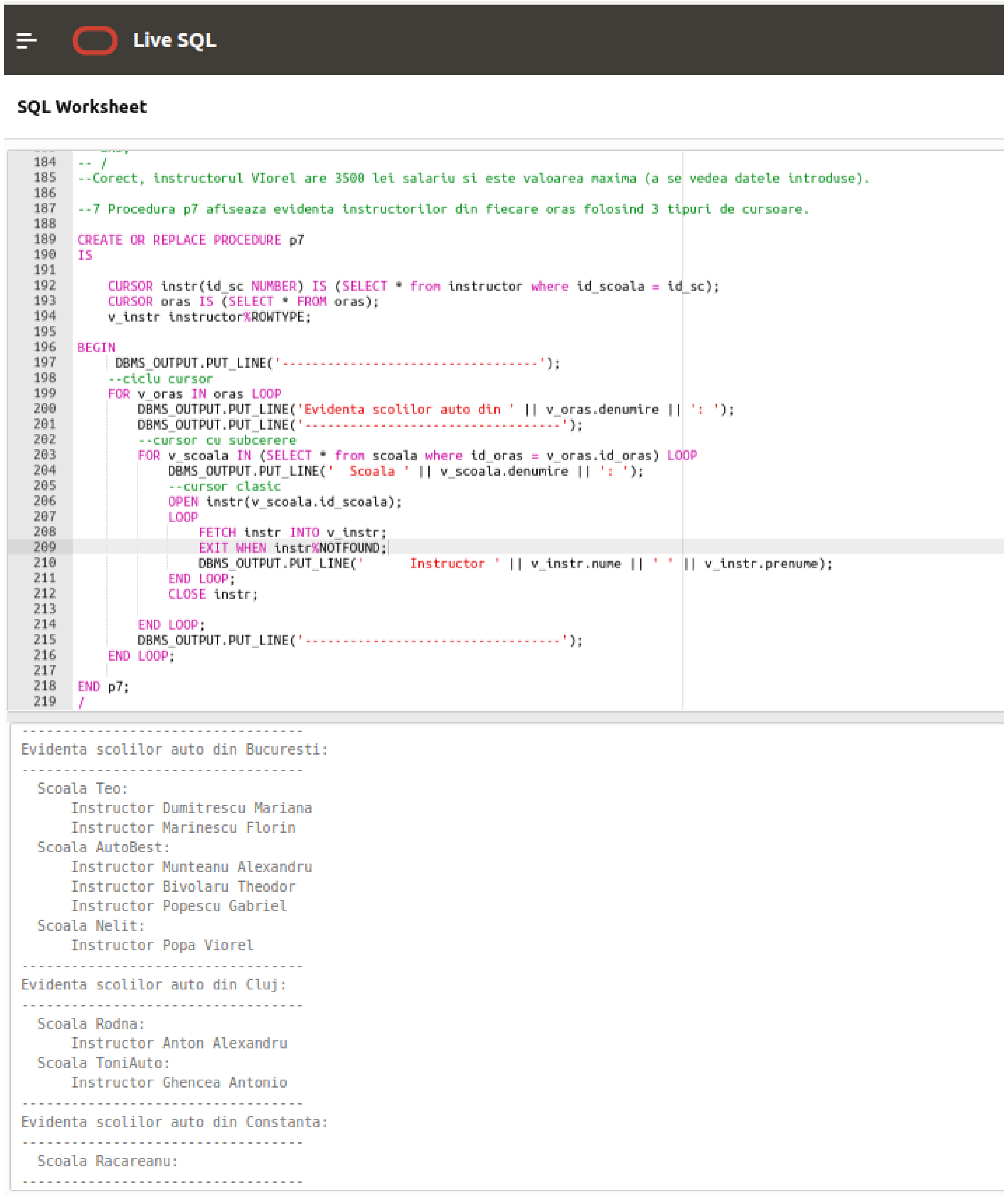
```
Function created.

Statement processed.
Salariul maxim pentru un instructor dupa marire cu procentul ales este 3500
```

Exercitiul 7

Un subprogram stocat care utilizeaza 3 tipuri de cursoare:

Sa se afiseze evidenta instructorilor din fiecare oras.

```
CREATE OR REPLACE PROCEDURE p7
IS

    CURSOR instr(id_sc NUMBER) IS (SELECT * from instructor where id_scoala = id_sc);
    CURSOR oras IS (SELECT * FROM oras);
    v_instr instructor%ROWTYPE;

BEGIN
     DBMS_OUTPUT.PUT_LINE('-----------------------------------');
    --ciclu cursor
    FOR v_oras IN oras LOOP
        DBMS_OUTPUT.PUT_LINE('Evidenta scolilor auto din ' || v_oras.denumire || ': ');
        DBMS_OUTPUT.PUT_LINE('-----------------------------------');
        --cursor cu subcerere
        FOR v_scoala IN (SELECT * from scoala where id_oras = v_oras.id_oras) LOOP
            DBMS_OUTPUT.PUT_LINE('  Scoala ' || v_scoala.denumire || ': ');
            --cursor clasic
            OPEN instr(v_scoala.id_scoala);
            LOOP
                FETCH instr INTO v_instr;
                EXIT WHEN instr%NOTFOUND;
                DBMS_OUTPUT.PUT_LINE('      Instructor ' || v_instr.nume || ' '
                || v_instr.prenume);
            END LOOP;
            CLOSE instr;
        END LOOP;
        DBMS_OUTPUT.PUT_LINE('---------------------------------');
    END LOOP;

END p7;
/

BEGIN
    p7();
END;
/
```



```
--   /
--Corect, instructorul VIorel are 3500 lei salariu si este valoarea maxima (a se vedea datele introduse).

--7 Procedura p7 afiseaza evidenta instructorilor din fiecare oras folosind 3 tipuri de cursoare.

CREATE OR REPLACE PROCEDURE p7
IS

    CURSOR instr(id_sc NUMBER) IS (SELECT * from instructor where id_scoala = id_sc);
    CURSOR oras IS (SELECT * FROM oras);
    v_instr instructor%ROWTYPE;

BEGIN
     DBMS_OUTPUT.PUT_LINE('---------------------------------');
    --ciclu cursor
    FOR v_oras IN oras LOOP
        DBMS_OUTPUT.PUT_LINE('Evidenta scolilor auto din ' || v_oras.denumire || ': ');
        DBMS_OUTPUT.PUT_LINE('---------------------------------');
        --cursor cu subcerere
        FOR v_scoala IN (SELECT * from scoala where id_oras = v_oras.id_oras) LOOP
            DBMS_OUTPUT.PUT_LINE('  Scoala ' || v_scoala.denumire || ': ');
            --cursor clasic
            OPEN instr(v_scoala.id_scoala);
            LOOP
                FETCH instr INTO v_instr;
                EXIT WHEN instr%NOTFOUND;
                DBMS_OUTPUT.PUT_LINE('      Instructor ' || v_instr.nume || ' ' || v_instr.prenume);
            END LOOP;
            CLOSE instr;

        END LOOP;
        DBMS_OUTPUT.PUT_LINE('---------------------------------');
    END LOOP;

END p7;
/
```

```
---------------------------------
Evidenta scolilor auto din Bucuresti:
---------------------------------
  Scoala Teo:
      Instructor Dumitrescu Mariana
      Instructor Marinescu Florin
  Scoala AutoBest:
      Instructor Munteanu Alexandru
      Instructor Bivolaru Theodor
      Instructor Popescu Gabriel
  Scoala Nelit:
      Instructor Popa Viorel
---------------------------------
Evidenta scolilor auto din Cluj:
---------------------------------
  Scoala Rodna:
      Instructor Anton Alexandru
  Scoala ToniAuto:
      Instructor Ghencea Antonio
---------------------------------
Evidenta scolilor auto din Constanta:
---------------------------------
  Scoala Racareanu:
---------------------------------
```

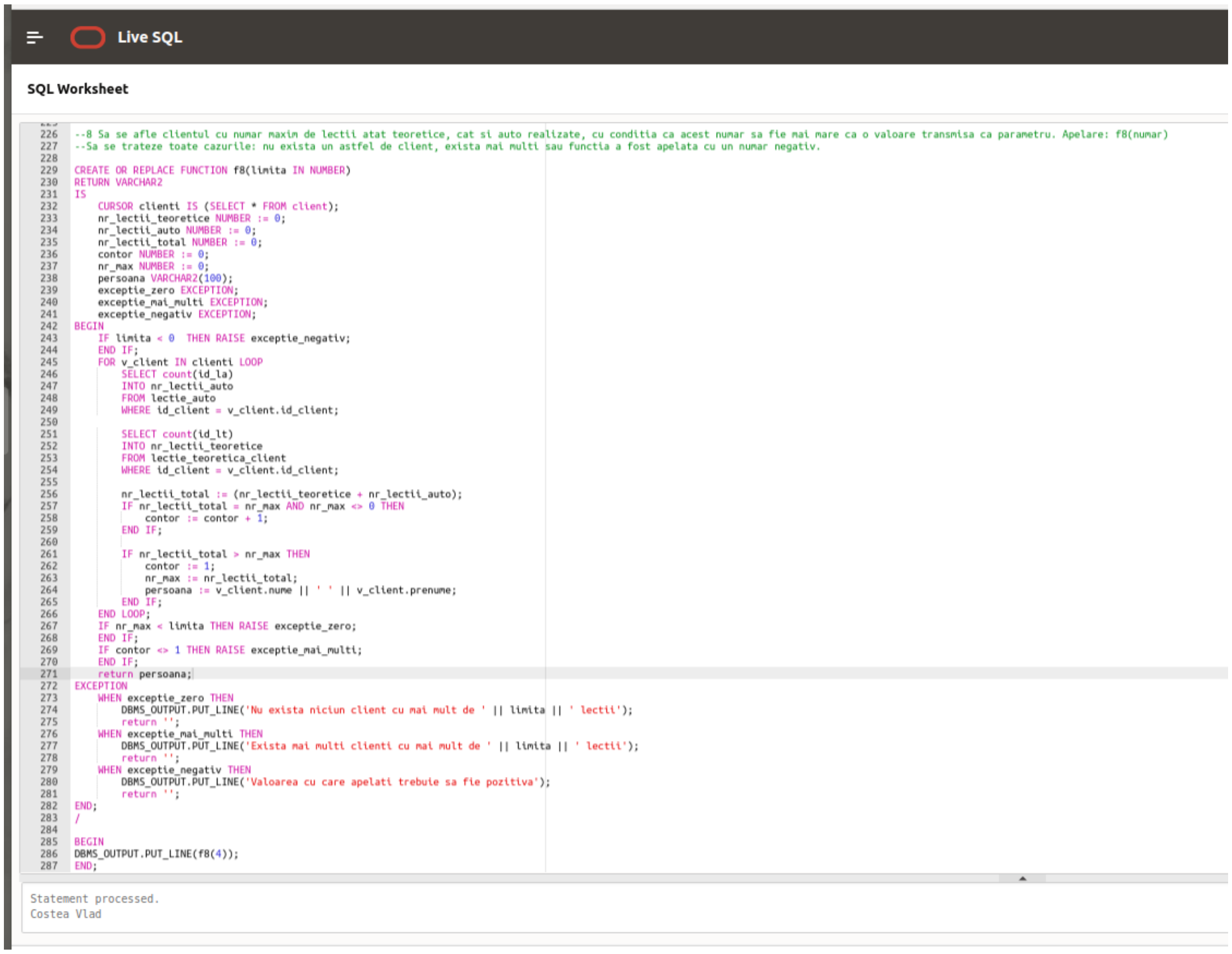
Exercitiul 8

Definiti un subprogram stocat de tip functie care sa utilizeze 3 dintre tabelele definite. Tratati toate exceptiile care pot aparea.

Sa se afle clientul cu numar maxim de lectii atat teoretice, cat si auto realizate, cu conditia ca acest numar sa fie mai mare ca o valoare transmisa ca parametru. Sa se trateze toate cazurile: nu exista un astfel de client, exista mai multi sau functia a fost apelata cu un numar negativ.

```
CREATE OR REPLACE FUNCTION f8(limita IN NUMBER)
RETURN VARCHAR2
IS
    CURSOR clienti IS (SELECT * FROM client);
    nr_lectii_teoretice NUMBER := 0;
    nr_lectii_auto NUMBER := 0;
    nr_lectii_total NUMBER := 0;
    contor NUMBER := 0;
    nr_max NUMBER := 0;
    persoana VARCHAR2(100);
    exceptie_zero EXCEPTION;
    exceptie_mai_multi EXCEPTION;
    exceptie_negativ EXCEPTION;
BEGIN
    IF limita < 0  THEN RAISE exceptie_negativ;
    END IF;
    FOR v_client IN clienti LOOP
        SELECT count(id_la)
        INTO nr_lectii_auto
        FROM lectie_auto
        WHERE id_client = v_client.id_client;

        SELECT count(id_lt)
        INTO nr_lectii_teoretice
        FROM lectie_teoretica_client
        WHERE id_client = v_client.id_client;

        nr_lectii_total := (nr_lectii_teoretice + nr_lectii_auto);
        IF nr_lectii_total = nr_max AND nr_max <> 0 THEN
            contor := contor + 1;
        END IF;
        IF nr_lectii_total > nr_max THEN
            contor := 1;
            nr_max := nr_lectii_total;
            persoana := v_client.nume || ' ' || v_client.prenume;
        END IF;
    END LOOP;
    IF nr_max < limita THEN RAISE exceptie_zero;
    END IF;
    IF contor <> 1 THEN RAISE exceptie_mai_multi;
    END IF;
    return persoana;
EXCEPTION
    WHEN exceptie_zero THEN
        DBMS_OUTPUT.PUT_LINE('Nu exista niciun client cu mai mult de '
        || limita || ' lectii');
        return '';
    WHEN exceptie_mai_multi THEN
        DBMS_OUTPUT.PUT_LINE('Exista mai multi clienti cu mai mult de '
        || limita || ' lectii');
        return '';
    WHEN exceptie_negativ THEN
        DBMS_OUTPUT.PUT_LINE('Valoarea cu care apelati trebuie sa fie pozitiva');
        return '';
END;
/

BEGIN
DBMS_OUTPUT.PUT_LINE(f8(4));
END;
/
BEGIN
DBMS_OUTPUT.PUT_LINE(f8(15));
END;
/

BEGIN
DBMS_OUTPUT.PUT_LINE(f8(-1));
END;
/
--INSERT INTO lectie_teoretica_client VALUES (5, 2) ;
--INSERT INTO lectie_teoretica_client VALUES (6, 2) ;
BEGIN
DBMS_OUTPUT.PUT_LINE(f8(2));
END;
/
```

Exista un singur client:



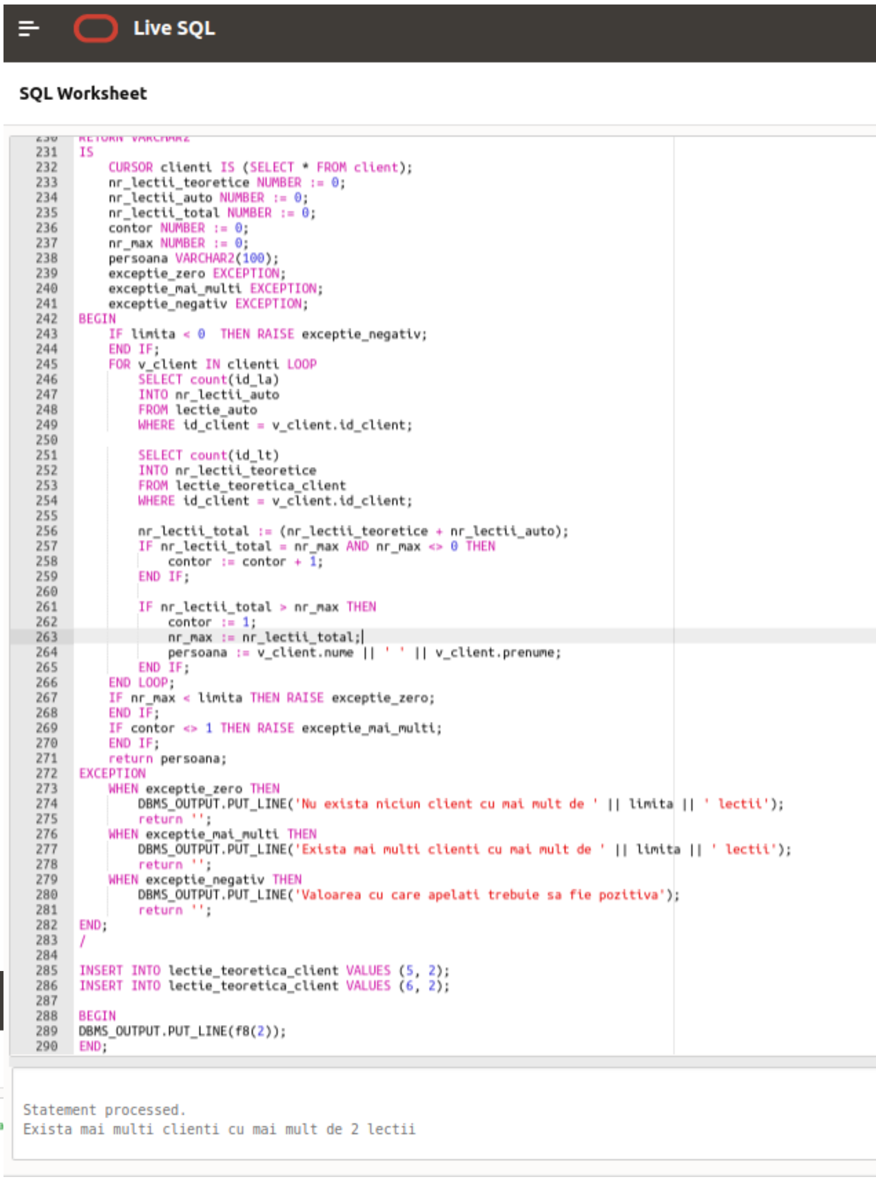
Live SQL

SQL Worksheet

```
--8 Sa se afle clientul cu numar maxim de lectii atat teoretice, cat si auto realizate, cu conditia ca acest numar sa fie mai mare ca o valoare transmisa ca parametru. Apelare: f8(numar)
--Sa se trateze toate cazurile: nu exista un astfel de client, exista mai multi sau functia a fost apelata cu un numar negativ.

CREATE OR REPLACE FUNCTION f8(limita IN NUMBER)
RETURN VARCHAR2
IS
    CURSOR clienti IS (SELECT * FROM client);
    nr_lectii_teoretice NUMBER := 0;
    nr_lectii_auto NUMBER := 0;
    nr_lectii_total NUMBER := 0;
    contor NUMBER := 0;
    nr_max NUMBER := 0;
    persoana VARCHAR2(100);
    exceptie_zero EXCEPTION;
    exceptie_mai_multi EXCEPTION;
    exceptie_negativ EXCEPTION;
BEGIN
    IF limita < 0  THEN RAISE exceptie_negativ;
    END IF;
    FOR v_client IN clienti LOOP
        SELECT count(id_la)
        INTO nr_lectii_auto
        FROM lectie_auto
        WHERE id_client = v_client.id_client;

        SELECT count(id_lt)
        INTO nr_lectii_teoretice
        FROM lectie_teoretica_client
        WHERE id_client = v_client.id_client;

        nr_lectii_total := (nr_lectii_teoretice + nr_lectii_auto);
        IF nr_lectii_total = nr_max AND nr_max <> 0 THEN
            contor := contor + 1;
        END IF;

        IF nr_lectii_total > nr_max THEN
            contor := 1;
            nr_max := nr_lectii_total;
            persoana := v_client.nume || ' ' || v_client.prenume;
        END IF;
    END LOOP;
    IF nr_max < limita THEN RAISE exceptie_zero;
    END IF;
    IF contor <> 1 THEN RAISE exceptie_mai_multi;
    END IF;
    return persoana;
EXCEPTION
    WHEN exceptie_zero THEN
        DBMS_OUTPUT.PUT_LINE('Nu exista niciun client cu mai mult de ' || limita || ' lectii');
        return '';
    WHEN exceptie_mai_multi THEN
        DBMS_OUTPUT.PUT_LINE('Exista mai multi clienti cu mai mult de ' || limita || ' lectii');
        return '';
    WHEN exceptie_negativ THEN
        DBMS_OUTPUT.PUT_LINE('Valoarea cu care apelati trebuie sa fie pozitiva');
        return '';
END;
/

BEGIN
DBMS_OUTPUT.PUT_LINE(f8(4));
END;
```

```
Statement processed.
Costea Vlad
```

Exista mai multi clienti:

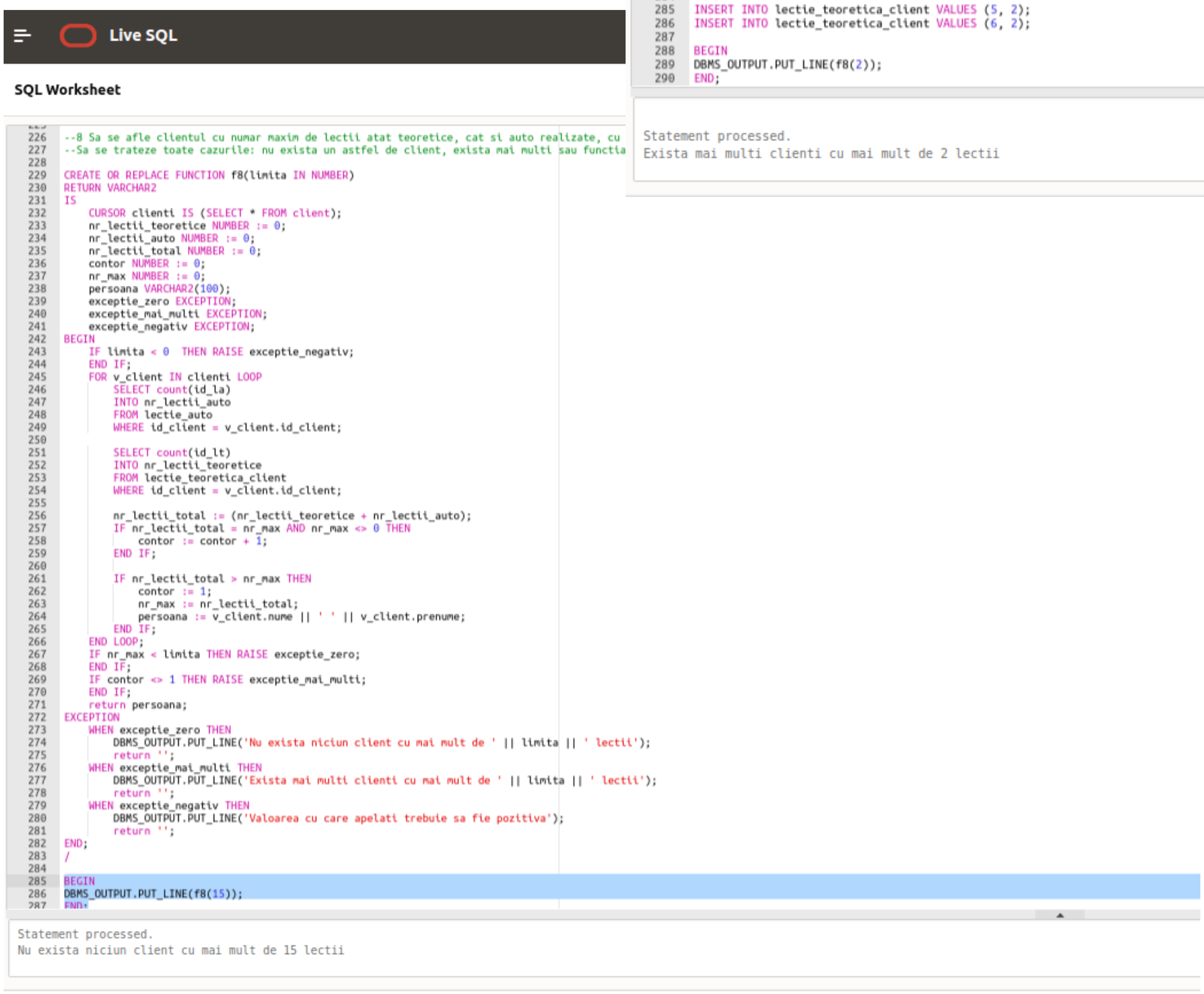


SQL Worksheet

```
IS
    CURSOR clienti IS (SELECT * FROM client);
    nr_lectii_teoretice NUMBER := 0;
    nr_lectii_auto NUMBER := 0;
    nr_lectii_total NUMBER := 0;
    contor NUMBER := 0;
    nr_max NUMBER := 0;
    persoana VARCHAR2(100);
    exceptie_zero EXCEPTION;
    exceptie_mai_multi EXCEPTION;
    exceptie_negativ EXCEPTION;
BEGIN
    IF limita < 0  THEN RAISE exceptie_negativ;
    END IF;
    FOR v_client IN clienti LOOP
        SELECT count(id_la)
        INTO nr_lectii_auto
        FROM lectie_auto
        WHERE id_client = v_client.id_client;

        SELECT count(id_lt)
        INTO nr_lectii_teoretice
        FROM lectie_teoretica_client
        WHERE id_client = v_client.id_client;

        nr_lectii_total := (nr_lectii_teoretice + nr_lectii_auto);
        IF nr_lectii_total = nr_max AND nr_max <> 0 THEN
            contor := contor + 1;
        END IF;

        IF nr_lectii_total > nr_max THEN
            contor := 1;
            nr_max := nr_lectii_total;
            persoana := v_client.nume || ' ' || v_client.prenume;
        END IF;
    END LOOP;
    IF nr_max < limita THEN RAISE exceptie_zero;
    END IF;
    IF contor <> 1 THEN RAISE exceptie_mai_multi;
    END IF;
    return persoana;
EXCEPTION
    WHEN exceptie_zero THEN
        DBMS_OUTPUT.PUT_LINE('Nu exista niciun client cu mai mult de ' || limita || ' lectii');
        return '';
    WHEN exceptie_mai_multi THEN
        DBMS_OUTPUT.PUT_LINE('Exista mai multi clienti cu mai mult de ' || limita || ' lectii');
        return '';
    WHEN exceptie_negativ THEN
        DBMS_OUTPUT.PUT_LINE('Valoarea cu care apelati trebuie sa fie pozitiva');
        return '';
END;
/

INSERT INTO lectie_teoretica_client VALUES (5, 2);
INSERT INTO lectie_teoretica_client VALUES (6, 2);

BEGIN
DBMS_OUTPUT.PUT_LINE(f8(2));
END;
```

```
Statement processed.
Exista mai multi clienti cu mai mult de 2 lectii
```

Nu exista niciun client :



SQL Worksheet

```
--8 Sa se afle clientul cu numar maxim de lectii atat teoretice, cat si auto realizate, cu
--Sa se trateze toate cazurile: nu exista un astfel de client, exista mai multi sau functia

CREATE OR REPLACE FUNCTION f8(limita IN NUMBER)
RETURN VARCHAR2
IS
    CURSOR clienti IS (SELECT * FROM client);
    nr_lectii_teoretice NUMBER := 0;
    nr_lectii_auto NUMBER := 0;
    nr_lectii_total NUMBER := 0;
    contor NUMBER := 0;
    nr_max NUMBER := 0;
    persoana VARCHAR2(100);
    exceptie_zero EXCEPTION;
    exceptie_mai_multi EXCEPTION;
    exceptie_negativ EXCEPTION;
BEGIN
    IF limita < 0  THEN RAISE exceptie_negativ;
    END IF;
    FOR v_client IN clienti LOOP
        SELECT count(id_la)
        INTO nr_lectii_auto
        FROM lectie_auto
        WHERE id_client = v_client.id_client;

        SELECT count(id_lt)
        INTO nr_lectii_teoretice
        FROM lectie_teoretica_client
        WHERE id_client = v_client.id_client;

        nr_lectii_total := (nr_lectii_teoretice + nr_lectii_auto);
        IF nr_lectii_total = nr_max AND nr_max <> 0 THEN
            contor := contor + 1;
        END IF;

        IF nr_lectii_total > nr_max THEN
            contor := 1;
            nr_max := nr_lectii_total;
            persoana := v_client.nume || ' ' || v_client.prenume;
        END IF;
    END LOOP;
    IF nr_max < limita THEN RAISE exceptie_zero;
    END IF;
    IF contor <> 1 THEN RAISE exceptie_mai_multi;
    END IF;
    return persoana;
EXCEPTION
    WHEN exceptie_zero THEN
        DBMS_OUTPUT.PUT_LINE('Nu exista niciun client cu mai mult de ' || limita || ' lectii');
        return '';
    WHEN exceptie_mai_multi THEN
        DBMS_OUTPUT.PUT_LINE('Exista mai multi clienti cu mai mult de ' || limita || ' lectii');
        return '';
    WHEN exceptie_negativ THEN
        DBMS_OUTPUT.PUT_LINE('Valoarea cu care apelati trebuie sa fie pozitiva');
        return '';
END;
/

BEGIN
DBMS_OUTPUT.PUT_LINE(f8(15));
END;
```

```
Statement processed.
Nu exista niciun client cu mai mult de 15 lectii
```

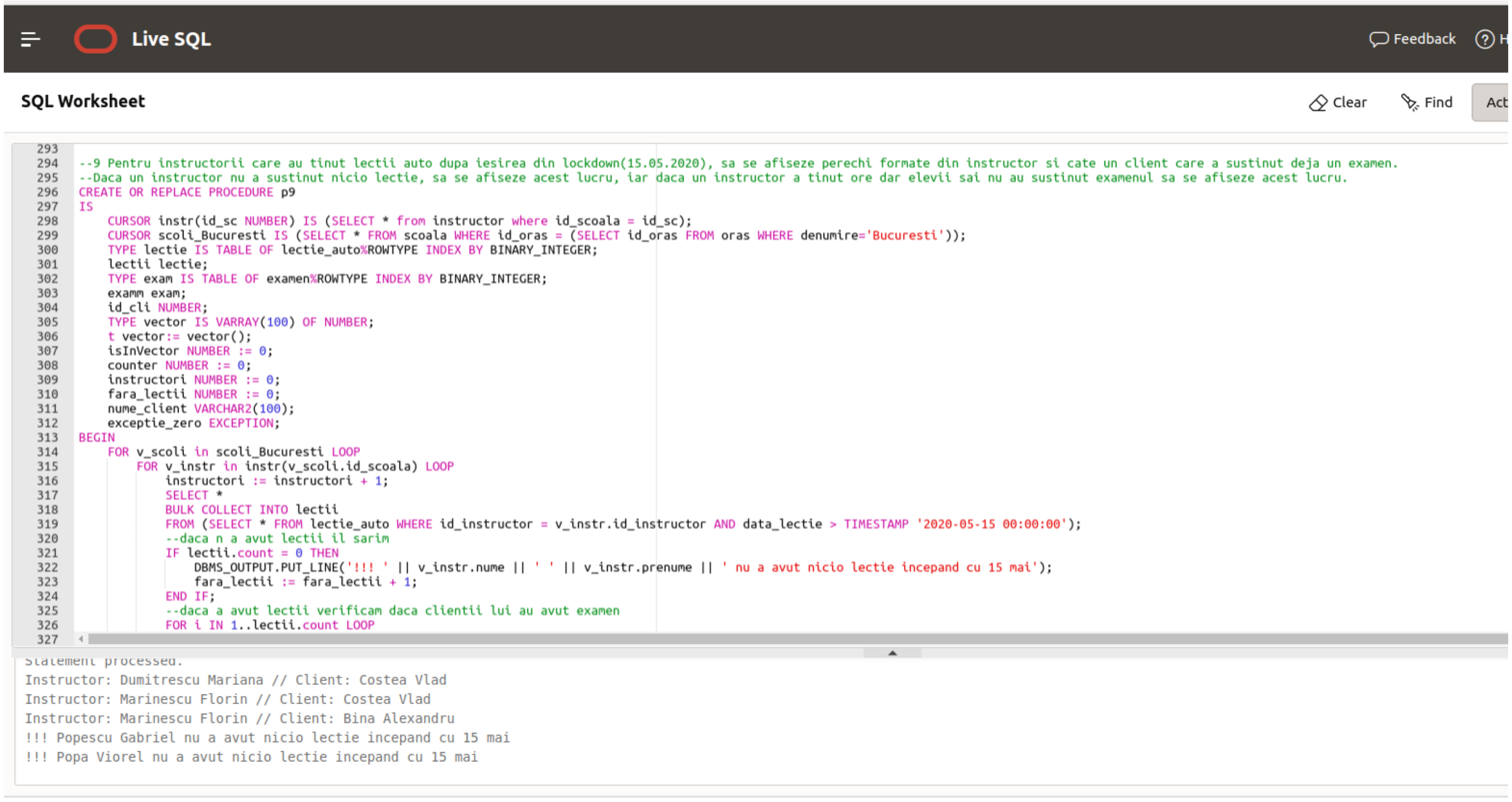
Exercitiul 9

Un subprogram stocat de tip procedura care sa foloseasca 5 tabele (instructor, scoala, oras, lectie auto, examen):

Pentru instructorii care au tinut lectii auto dupa iesirea din lockdown(15.05.2020), sa se afiseze perechi formate din instructor si cate un client care a sustinut deja un examen. Daca un instructor nu a sustinut nicio lectie, sa se afiseze acest lucru, iar daca un instructor a tinut ore dar elevii sai nu au sustinut examenul sa se afiseze acest lucru.

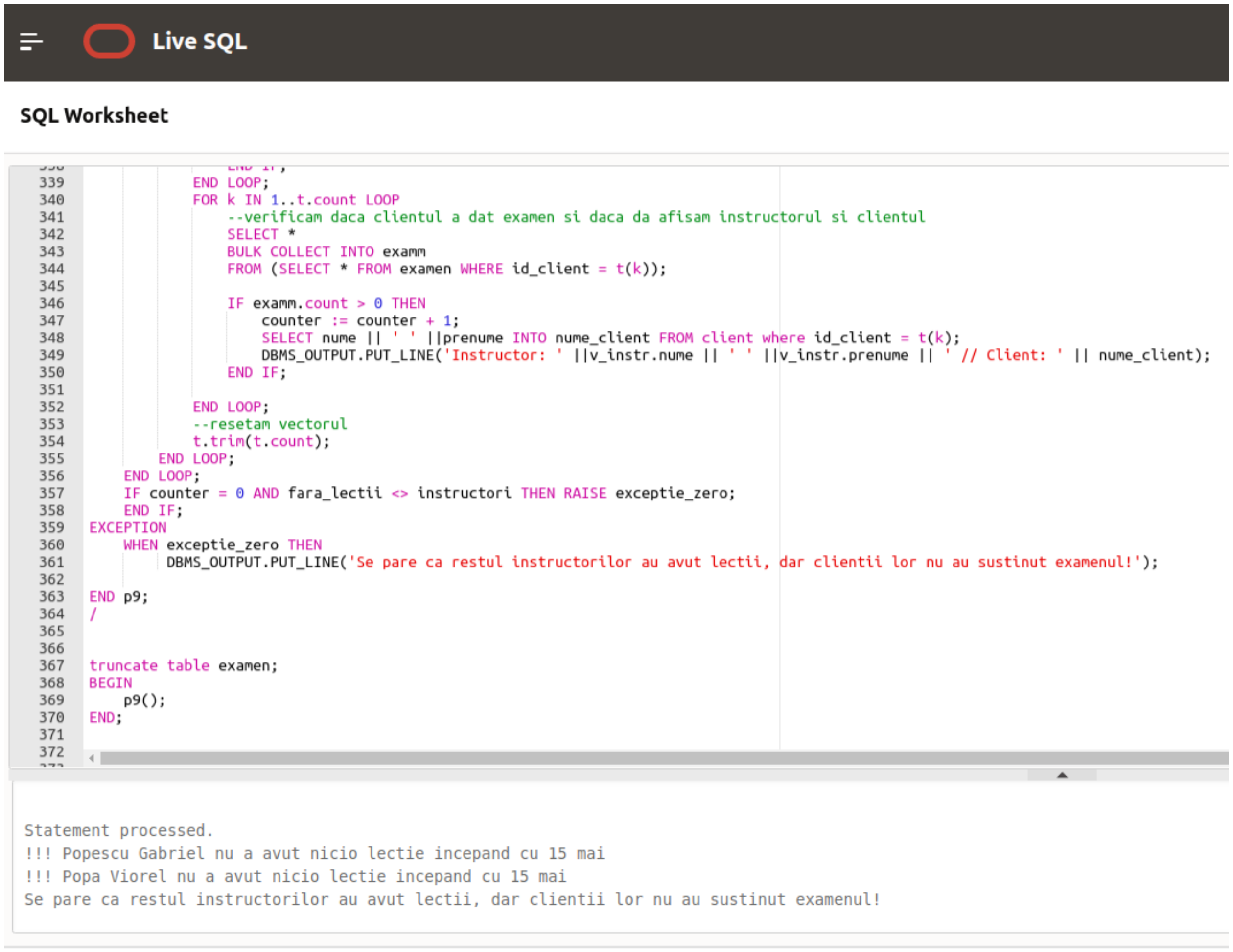
```
CREATE OR REPLACE PROCEDURE p9
IS
    CURSOR instr(id_sc NUMBER) IS (SELECT * from instructor where id_scoala = id_sc);
    CURSOR scoli_Bucuresti IS (SELECT * FROM scoala WHERE id_oras =
        (SELECT id_oras FROM oras WHERE denumire='Bucuresti'));
    TYPE lectie IS TABLE OF lectie_auto%ROWTYPE INDEX BY BINARY_INTEGER;
    lectii lectie;
    TYPE exam IS TABLE OF examen%ROWTYPE INDEX BY BINARY_INTEGER;
    examm exam;
    id_cli NUMBER;
    TYPE vector IS VARRAY(100) OF NUMBER;
    t vector:= vector();
    isInVector NUMBER := 0;
    counter NUMBER := 0;
    instructori NUMBER := 0;
    fara_lectii NUMBER := 0;
    nume_client VARCHAR2(100);
    exceptie_zero EXCEPTION;
BEGIN
    FOR v_scoli in scoli_Bucuresti LOOP
        FOR v_instr in instr(v_scoli.id_scoala) LOOP
            instructori := instructori + 1;
            SELECT *
            BULK COLLECT INTO lectii
            FROM (SELECT * FROM lectie_auto WHERE id_instructor = v_instr.id_instructor
                  AND data_lectie > TIMESTAMP '2020-05-15 00:00:00');
            --daca n a avut lectii il sarim
            IF lectii.count = 0 THEN
                DBMS_OUTPUT.PUT_LINE('!!! ' || v_instr.nume || ' ' || v_instr.prenume
                || ' nu a avut nicio lectie incepand cu 15 mai');
                fara_lectii := fara_lectii + 1;
            END IF;
            --daca a avut lectii verificam daca clientii lui au avut examen
            FOR i IN 1..lectii.count LOOP
                isInVector := 0;
                id_cli := lectii(i).id_client;
                --verificam daca am procesat deja clientul
                FOR j IN 1..t.count LOOP
                    IF t(j) = id_cli THEN isInVector := 1;
                    END IF;
                END LOOP;
                --daca nu, il adaugam in vector
                IF isInVector = 0 THEN
                    t.extend();
                    t(t.count) := id_cli;
                END IF;
            END LOOP;
            FOR k IN 1..t.count LOOP
                --verificam daca clientul a dat examen si daca da afisam
                --instructorul si clientul
                SELECT *
                BULK COLLECT INTO examm
                FROM (SELECT * FROM examen WHERE id_client = t(k));

                IF examm.count > 0 THEN
                    counter := counter + 1;
                    SELECT nume || ' ' ||prenume
                    INTO nume_client
                    FROM client
                   WHERE id_client = t(k);
                    DBMS_OUTPUT.PUT_LINE('Instructor: ' ||v_instr.nume
                        || ' ' ||v_instr.prenume || ' // Client: '
                        || nume_client);
                END IF;

            END LOOP;
            --resetam vectorul
            t.trim(t.count);
        END LOOP;
    END LOOP;
    IF counter = 0 AND fara_lectii <> instructori THEN RAISE exceptie_zero;
    END IF;
EXCEPTION
    WHEN exceptie_zero THEN
         DBMS_OUTPUT.PUT_LINE('Se pare ca restul instructorilor au avut lectii,
        dar clientii lor nu au sustinut examenul!');

END p9;
/

BEGIN
```

```
    p9();
END;
/
```



```
--9 Pentru instructorii care au tinut lectii auto dupa iesirea din lockdown(15.05.2020), sa se afiseze perechi formate din instructor si cate un client care a sustinut deja un examen.
--Daca un instructor nu a sustinut nicio lectie, sa se afiseze acest lucru, iar daca un instructor a tinut ore dar elevii sai nu au sustinut examenul sa se afiseze acest lucru.
CREATE OR REPLACE PROCEDURE p9
IS
    CURSOR instr(id_sc NUMBER) IS (SELECT * from instructor where id_scoala = id_sc);
    CURSOR scoli_Bucuresti IS (SELECT * FROM scoala WHERE id_oras = (SELECT id_oras FROM oras WHERE denumire='Bucuresti'));
    TYPE lectie IS TABLE OF lectie_auto%ROWTYPE INDEX BY BINARY_INTEGER;
    lectii lectie;
    TYPE exam IS TABLE OF examen%ROWTYPE INDEX BY BINARY_INTEGER;
    examm exam;
    id_cli NUMBER;
    TYPE vector IS VARRAY(100) OF NUMBER;
    t vector:= vector();
    isInVector NUMBER := 0;
    counter NUMBER := 0;
    instructori NUMBER := 0;
    fara_lectii NUMBER := 0;
    nume_client VARCHAR2(100);
    exceptie_zero EXCEPTION;
BEGIN
    FOR v_scoli in scoli_Bucuresti LOOP
        FOR v_instr in instr(v_scoli.id_scoala) LOOP
            instructori := instructori + 1;
            SELECT *
            BULK COLLECT INTO lectii
            FROM (SELECT * FROM lectie_auto WHERE id_instructor = v_instr.id_instructor AND data_lectie > TIMESTAMP '2020-05-15 00:00:00');
            --daca n a avut lectii il sarim
            IF lectii.count = 0 THEN
                DBMS_OUTPUT.PUT_LINE('!!! ' || v_instr.nume || ' ' || v_instr.prenume || ' nu a avut nicio lectie incepand cu 15 mai');
                fara_lectii := fara_lectii + 1;
            END IF;
            --daca a avut lectii verificam daca clientii lui au avut examen
            FOR i IN 1..lectii.count LOOP
```

```
Statement processed.
Instructor: Dumitrescu Mariana // Client: Costea Vlad
Instructor: Marinescu Florin // Client: Costea Vlad
Instructor: Marinescu Florin // Client: Bina Alexandru
!!! Popescu Gabriel nu a avut nicio lectie incepand cu 15 mai
!!! Popa Viorel nu a avut nicio lectie incepand cu 15 mai
```



```
            END LOOP;
            FOR k IN 1..t.count LOOP
                --verificam daca clientul a dat examen si daca da afisam instructorul si clientul
                SELECT *
                BULK COLLECT INTO examm
                FROM (SELECT * FROM examen WHERE id_client = t(k));

                IF examm.count > 0 THEN
                    counter := counter + 1;
                    SELECT nume || ' ' ||prenume INTO nume_client FROM client where id_client = t(k);
                    DBMS_OUTPUT.PUT_LINE('Instructor: ' ||v_instr.nume || ' ' ||v_instr.prenume || ' // Client: ' || nume_client);
                END IF;

            END LOOP;
            --resetam vectorul
            t.trim(t.count);
        END LOOP;
    END LOOP;
    IF counter = 0 AND fara_lectii <> instructori THEN RAISE exceptie_zero;
    END IF;
EXCEPTION
    WHEN exceptie_zero THEN
        DBMS_OUTPUT.PUT_LINE('Se pare ca restul instructorilor au avut lectii, dar clientii lor nu au sustinut examenul!');

END p9;
/


truncate table examen;
BEGIN
    p9();
END;
```

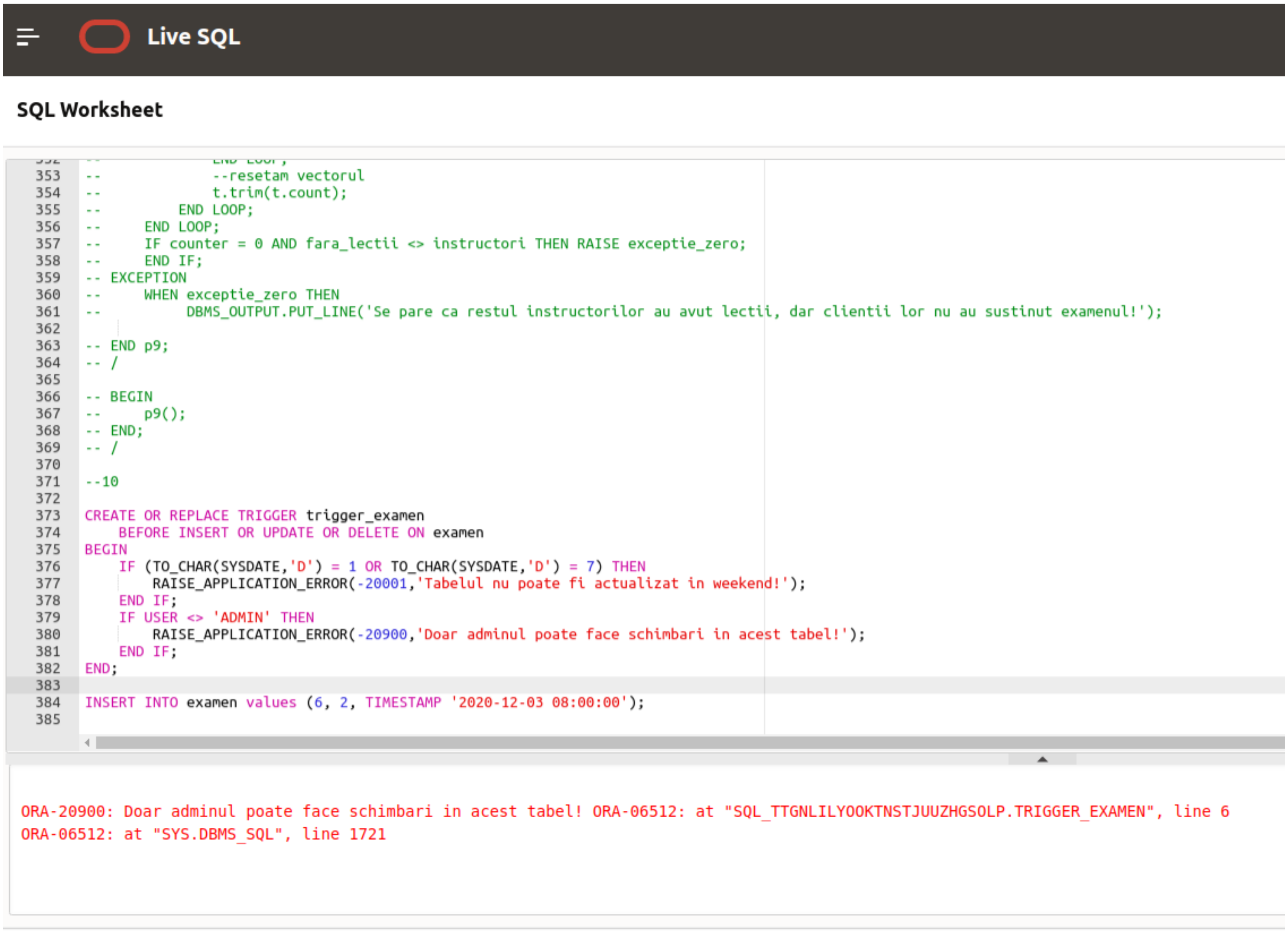
```
Statement processed.
!!! Popescu Gabriel nu a avut nicio lectie incepand cu 15 mai
!!! Popa Viorel nu a avut nicio lectie incepand cu 15 mai
Se pare ca restul instructorilor au avut lectii, dar clientii lor nu au sustinut examenul!
```

Exercitiul 10

Un trigger de tip LMD la nivel de comanda:

Sa se creeze un trigger care sa nu permita manipularea tabelului examen in weekend sau de catre altcineva inafara de admin.

```
CREATE OR REPLACE TRIGGER trigger_examen
    BEFORE INSERT OR UPDATE OR DELETE ON examen
BEGIN
    IF (TO_CHAR(SYSDATE,'D') = 1 OR TO_CHAR(SYSDATE,'D') = 7) THEN
        RAISE_APPLICATION_ERROR(-20001,'Tabelul nu poate fi actualizat in weekend!');
    END IF;
    IF USER <> 'ADMIN' THEN
        RAISE_APPLICATION_ERROR(-20900,'Doar adminul poate face schimbari in acest tabel!');
    END IF;
END;
/
INSERT INTO examen values (6, 2, TIMESTAMP '2020-12-03 08:00:00');
/
```



```
--            --resetam vectorul
--            t.trim(t.count);
--        END LOOP;
--    END LOOP;
--    IF counter = 0 AND fara_lectii <> instructori THEN RAISE exceptie_zero;
--    END IF;
-- EXCEPTION
--    WHEN exceptie_zero THEN
--        DBMS_OUTPUT.PUT_LINE('Se pare ca restul instructorilor au avut lectii, dar clientii lor nu au sustinut examenul!');

-- END p9;
-- /

-- BEGIN
--    p9();
-- END;
-- /

--10

CREATE OR REPLACE TRIGGER trigger_examen
    BEFORE INSERT OR UPDATE OR DELETE ON examen
BEGIN
    IF (TO_CHAR(SYSDATE,'D') = 1 OR TO_CHAR(SYSDATE,'D') = 7) THEN
        RAISE_APPLICATION_ERROR(-20001,'Tabelul nu poate fi actualizat in weekend!');
    END IF;
    IF USER <> 'ADMIN' THEN
        RAISE_APPLICATION_ERROR(-20900,'Doar adminul poate face schimbari in acest tabel!');
    END IF;
END;

INSERT INTO examen values (6, 2, TIMESTAMP '2020-12-03 08:00:00');
```

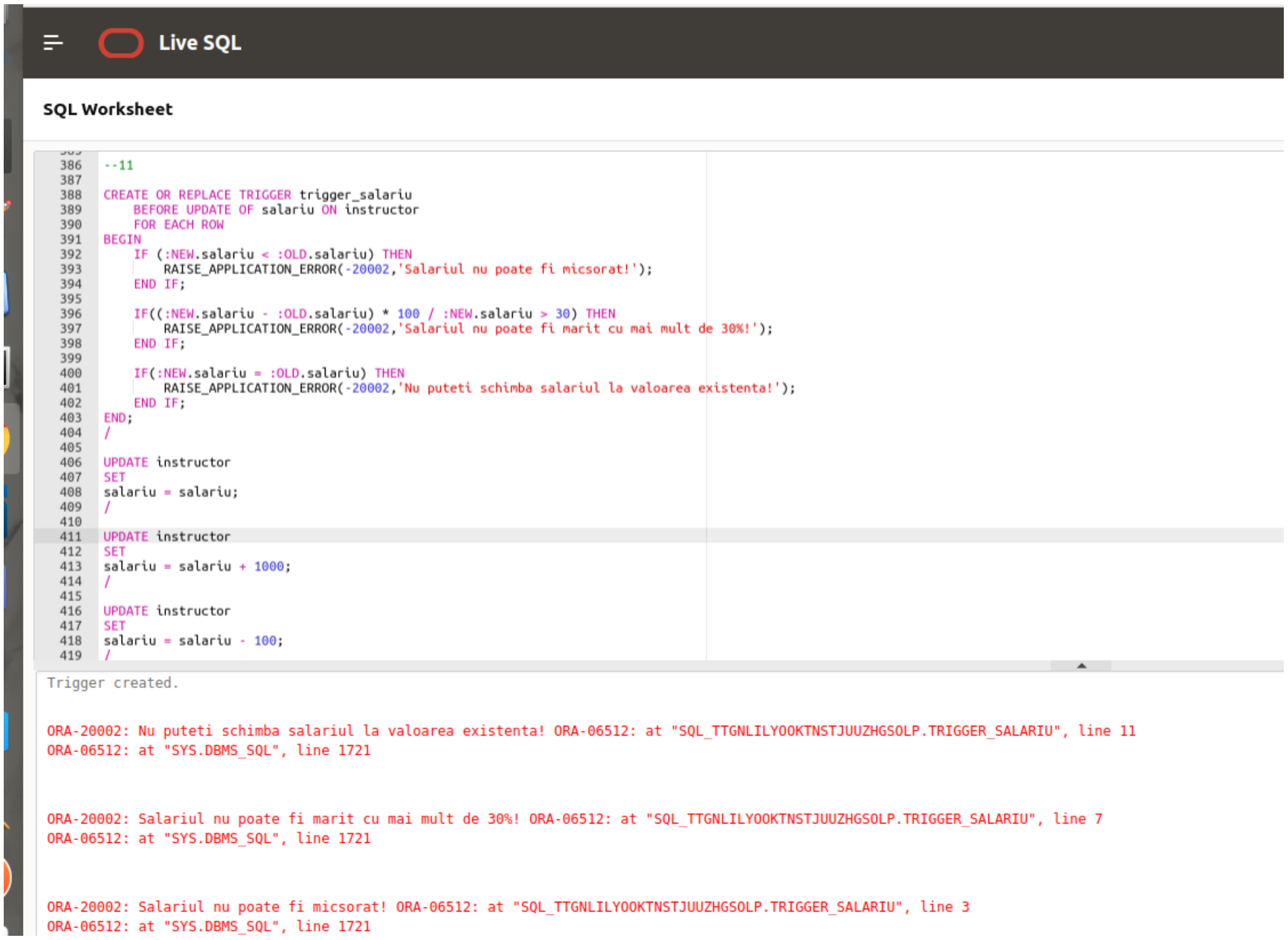
```
ORA-20900: Doar adminul poate face schimbari in acest tabel! ORA-06512: at "SQL_TTGNLILYOOKTNSTJUUZHGSOLP.TRIGGER_EXAMEN", line 6
ORA-06512: at "SYS.DBMS_SQL", line 1721
```

Exercitiul 11

Un trigger de tip LMD la nivel de linie:

Sa se creeze un trigger pentru editarea salariului instructorilor.

```
CREATE OR REPLACE TRIGGER trigger_salariu
    BEFORE UPDATE OF salariu ON instructor
    FOR EACH ROW
BEGIN
    IF (:NEW.salariu < :OLD.salariu) THEN
        RAISE_APPLICATION_ERROR(-20002,'Salariul nu poate fi micsorat!');
    END IF;

    IF((:NEW.salariu - :OLD.salariu) * 100 / :NEW.salariu > 30) THEN
        RAISE_APPLICATION_ERROR(-20002,'Salariul nu poate fi marit cu mai mult de 30%!');
    END IF;
    IF(:NEW.salariu = :OLD.salariu) THEN
        RAISE_APPLICATION_ERROR(-20002,'Nu puteti schimba salariul la valoarea existenta!')
    END IF;
END;
/

UPDATE instructor
SET
salariu = salariu;
/

UPDATE instructor
SET
salariu = salariu + 1000;
/

UPDATE instructor
SET
salariu = salariu - 100;
/
```



Live SQL

SQL Worksheet

```
--11

CREATE OR REPLACE TRIGGER trigger_salariu
    BEFORE UPDATE OF salariu ON instructor
    FOR EACH ROW
BEGIN
    IF (:NEW.salariu < :OLD.salariu) THEN
        RAISE_APPLICATION_ERROR(-20002,'Salariul nu poate fi micsorat!');
    END IF;

    IF((:NEW.salariu - :OLD.salariu) * 100 / :NEW.salariu > 30) THEN
        RAISE_APPLICATION_ERROR(-20002,'Salariul nu poate fi marit cu mai mult de 30%!');
    END IF;

    IF(:NEW.salariu = :OLD.salariu) THEN
        RAISE_APPLICATION_ERROR(-20002,'Nu puteti schimba salariul la valoarea existenta!');
    END IF;
END;
/

UPDATE instructor
SET
salariu = salariu;
/

UPDATE instructor
SET
salariu = salariu + 1000;
/

UPDATE instructor
SET
salariu = salariu - 100;
/
```

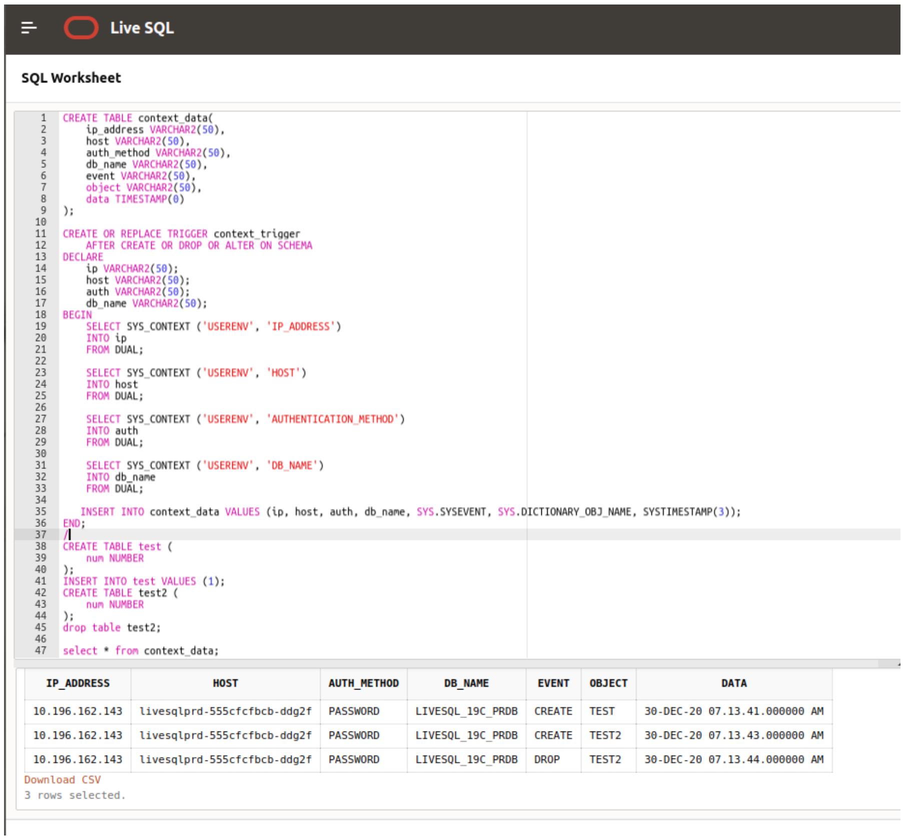
```
Trigger created.

ORA-20002: Nu puteti schimba salariul la valoarea existenta! ORA-06512: at "SQL_TTGNLILYOOKTNSTJUUZHGSOLP.TRIGGER_SALARIU", line 11
ORA-06512: at "SYS.DBMS_SQL", line 1721


ORA-20002: Salariul nu poate fi marit cu mai mult de 30%! ORA-06512: at "SQL_TTGNLILYOOKTNSTJUUZHGSOLP.TRIGGER_SALARIU", line 7
ORA-06512: at "SYS.DBMS_SQL", line 1721


ORA-20002: Salariul nu poate fi micsorat! ORA-06512: at "SQL_TTGNLILYOOKTNSTJUUZHGSOLP.TRIGGER_SALARIU", line 3
ORA-06512: at "SYS.DBMS_SQL", line 1721
```

Exercitiul 12

Un trigger de tip LDD:

Sa se stocheze date despre diferite evenimente.

```
CREATE TABLE context_data(
    ip_address VARCHAR2(50),
    host VARCHAR2(50),
    auth_method VARCHAR2(50),
    db_name VARCHAR2(50),
    event VARCHAR2(50),
    object VARCHAR2(50),
    data TIMESTAMP(0)
);

CREATE OR REPLACE TRIGGER context_trigger
    AFTER CREATE OR DROP OR ALTER ON SCHEMA
DECLARE
    ip VARCHAR2(50);
    host VARCHAR2(50);
    auth VARCHAR2(50);
    db_name VARCHAR2(50);
BEGIN
    SELECT SYS_CONTEXT ('USERENV', 'IP_ADDRESS')
    INTO ip
    FROM DUAL;

    SELECT SYS_CONTEXT ('USERENV', 'HOST')
    INTO host
    FROM DUAL;

    SELECT SYS_CONTEXT ('USERENV', 'AUTHENTICATION_METHOD')
    INTO auth
    FROM DUAL;

    SELECT SYS_CONTEXT ('USERENV', 'DB_NAME')
    INTO db_name
    FROM DUAL;

   INSERT INTO context_data VALUES (ip, host, auth, db_name,
        SYS.SYSEVENT, SYS.DICTIONARY_OBJ_NAME, SYSTIMESTAMP(3));
END;
/
CREATE TABLE test (
    num NUMBER
);
INSERT INTO test VALUES (1);
CREATE TABLE test2 (
    num NUMBER
);
drop table test2;

select * from context_data;
```



Live SQL

SQL Worksheet

```
CREATE TABLE context_data(
    ip_address VARCHAR2(50),
    host VARCHAR2(50),
    auth_method VARCHAR2(50),
    db_name VARCHAR2(50),
    event VARCHAR2(50),
    object VARCHAR2(50),
    data TIMESTAMP(0)
);

CREATE OR REPLACE TRIGGER context_trigger
    AFTER CREATE OR DROP OR ALTER ON SCHEMA
DECLARE
    ip VARCHAR2(50);
    host VARCHAR2(50);
    auth VARCHAR2(50);
    db_name VARCHAR2(50);
BEGIN
    SELECT SYS_CONTEXT ('USERENV', 'IP_ADDRESS')
    INTO ip
    FROM DUAL;

    SELECT SYS_CONTEXT ('USERENV', 'HOST')
    INTO host
    FROM DUAL;

    SELECT SYS_CONTEXT ('USERENV', 'AUTHENTICATION_METHOD')
    INTO auth
    FROM DUAL;

    SELECT SYS_CONTEXT ('USERENV', 'DB_NAME')
    INTO db_name
    FROM DUAL;

   INSERT INTO context_data VALUES (ip, host, auth, db_name, SYS.SYSEVENT, SYS.DICTIONARY_OBJ_NAME, SYSTIMESTAMP(3));
END;
/
CREATE TABLE test (
    num NUMBER
);
INSERT INTO test VALUES (1);
CREATE TABLE test2 (
    num NUMBER
);
drop table test2;

select * from context_data;
```

IP_ADDRESS	HOST	AUTH_METHOD	DB_NAME	EVENT	OBJECT	DATA
10.196.162.143	livesqlprd-555cfcfbcb-ddg2f	PASSWORD	LIVESQL_19C_PRDB	CREATE	TEST	30-DEC-20 07.13.41.000000 AM
10.196.162.143	livesqlprd-555cfcfbcb-ddg2f	PASSWORD	LIVESQL_19C_PRDB	CREATE	TEST2	30-DEC-20 07.13.43.000000 AM
10.196.162.143	livesqlprd-555cfcfbcb-ddg2f	PASSWORD	LIVESQL_19C_PRDB	DROP	TEST2	30-DEC-20 07.13.44.000000 AM

Download CSV

3 rows selected.

Exercitiul 13 si Exercitiul 14

Un pachet care sa contina toate obiectele definite in cadrul proiectului, tipuri de date complexe si obiecte necesare pentru actiuni integrate.

Adaugati in pachet o functie cu actiuni integrate pentru resetarea datelor din tabele la prima versiune.

```
CREATE OR REPLACE PACKAGE pachet AS
    --6
    FUNCTION f6 (procent IN NUMBER, lim_inf IN NUMBER, lim_sup IN NUMBER)
        RETURN NUMBER;
    -- --7
    PROCEDURE p7;
    -- --8
    FUNCTION f8(limita IN NUMBER)
        RETURN VARCHAR2;
    -- --9
    PROCEDURE p9;
    --14
    PROCEDURE insert_default_table_data;
    PROCEDURE delete_table_data;
    PROCEDURE reset_table;

END pachet;
/

CREATE OR REPLACE PACKAGE BODY pachet AS
----6
FUNCTION f6 (procent IN NUMBER, lim_inf IN NUMBER, lim_sup IN NUMBER)
RETURN  NUMBER
IS
    TYPE ins IS TABLE OF instructor%ROWTYPE INDEX BY BINARY_INTEGER;
    instructs ins;
    nr_instructori NUMBER := 0;
    id_instr NUMBER := 0;
    salariu_max NUMBER := 0;
BEGIN
    salariu_max := 0;
    SELECT *
    BULK COLLECT INTO instructs
    FROM (select *
          from instructor
          where salariu >= lim_inf and salariu <= lim_sup);
    nr_instructori := instructs.count;

    FOR i IN 1..nr_instructori LOOP
        id_instr :=  instructs(i).id_instructor;
        UPDATE instructor
            SET salariu = (100 + procent) / 100 * salariu
            WHERE id_instructor = id_instr;
    END LOOP;

    SELECT max(salariu)
    INTO salariu_max
    FROM instructor;

    return salariu_max;
END f6;

--7
PROCEDURE p7
IS
    CURSOR instr(id_sc NUMBER) RETURN instructor%ROWTYPE IS (SELECT * from instructor
where id_scoala = id_sc);
    CURSOR orass RETURN oras%ROWTYPE IS SELECT * FROM oras;
    v_instr instructor%ROWTYPE;
BEGIN
     DBMS_OUTPUT.PUT_LINE('-----------------------------------');
    --ciclu cursor
    FOR v_oras IN orass LOOP
        DBMS_OUTPUT.PUT_LINE('Evidenta scolilor auto din ' || v_oras.denumire || ': ');
        DBMS_OUTPUT.PUT_LINE('-----------------------------------');
        --cursor cu subcerere
        FOR v_scoala IN (SELECT * from scoala where id_oras = v_oras.id_oras) LOOP
            DBMS_OUTPUT.PUT_LINE('  Scoala ' || v_scoala.denumire || ': ');
            --cursor clasic
            OPEN instr(v_scoala.id_scoala);
            LOOP
                FETCH instr INTO v_instr;
                EXIT WHEN instr%NOTFOUND;
                DBMS_OUTPUT.PUT_LINE('      Instructor '
|| v_instr.nume || ' ' || v_instr.prenume);
            END LOOP;
            CLOSE instr;

        END LOOP;
        DBMS_OUTPUT.PUT_LINE('-----------------------------------');
    END LOOP;
END p7;

--8
FUNCTION f8(limita IN NUMBER)
RETURN VARCHAR2
IS
    persoana VARCHAR2(100);
    CURSOR clienti RETURN client%ROWTYPE IS (SELECT * FROM client);
    nr_lectii_teoretice NUMBER := 0;
    nr_lectii_auto NUMBER := 0;
    nr_lectii_total NUMBER := 0;
    contor NUMBER := 0;
    nr_max NUMBER := 0;
    exceptie_zero EXCEPTION;
    exceptie_mai_multi EXCEPTION;
    exceptie_negativ EXCEPTION;
BEGIN
    IF limita < 0  THEN RAISE exceptie_negativ;
    END IF;
    FOR v_client IN clienti LOOP
        SELECT count(id_la)
        INTO nr_lectii_auto
        FROM lectie_auto
        WHERE id_client = v_client.id_client;

        SELECT count(id_lt)
        INTO nr_lectii_teoretice
        FROM lectie_teoretica_client
        WHERE id_client = v_client.id_client;

        nr_lectii_total := (nr_lectii_teoretice + nr_lectii_auto);
        IF nr_lectii_total = nr_max AND nr_max <> 0 THEN
            contor := contor + 1;
        END IF;

        IF nr_lectii_total > nr_max THEN
            contor := 1;
            nr_max := nr_lectii_total;
            persoana := v_client.nume || ' ' || v_client.prenume;
        END IF;
    END LOOP;
    IF nr_max < limita THEN RAISE exceptie_zero;
    END IF;
    IF contor <> 1 THEN RAISE exceptie_mai_multi;
    END IF;
    return persoana;
    EXCEPTION
        WHEN exceptie_zero THEN
            DBMS_OUTPUT.PUT_LINE('Nu exista niciun client cu mai mult de ' || limita
|| ' lectii');
            return '';
        WHEN exceptie_mai_multi THEN
            DBMS_OUTPUT.PUT_LINE('Exista mai multi clienti cu mai mult de ' || limita
|| ' lectii');
            return '';
        WHEN exceptie_negativ THEN
            DBMS_OUTPUT.PUT_LINE('Valoarea cu care apelati trebuie sa fie pozitiva');
            return '';
END f8;

---9
PROCEDURE p9
IS
    CURSOR instr(id_sc NUMBER) RETURN instructor%ROWTYPE IS
 (SELECT * from instructor where id_scoala = id_sc);
    CURSOR scoli_Bucuresti RETURN scoala%ROWTYPE IS
(SELECT * FROM scoala WHERE id_oras = (SELECT id_oras FROM oras WHERE denumire='Bucuresti'))
    TYPE lectie IS TABLE OF lectie_auto%ROWTYPE INDEX BY BINARY_INTEGER;
    lectii lectie;
    TYPE exam IS TABLE OF examen%ROWTYPE INDEX BY BINARY_INTEGER;
    examm exam;
    id_cli NUMBER;
    TYPE vector IS VARRAY(100) OF NUMBER;
    t vector:= vector();
    isInVector NUMBER := 0;
    counter NUMBER := 0;
    instructori NUMBER := 0;
    fara_lectii NUMBER := 0;
    nume_client VARCHAR2(100);
    exceptie_zero EXCEPTION;
BEGIN
    FOR v_scoli in scoli_Bucuresti LOOP
        FOR v_instr in instr(v_scoli.id_scoala) LOOP
            instructori := instructori + 1;
            SELECT *
            BULK COLLECT INTO lectii
            FROM (SELECT * FROM lectie_auto WHERE id_instructor = v_instr.id_instructor
 AND data_lectie > TIMESTAMP '2020-05-15 00:00:00');
            --daca n a avut lectii il sarim
            IF lectii.count = 0 THEN
                DBMS_OUTPUT.PUT_LINE('!!! ' || v_instr.nume || ' '
|| v_instr.prenume || ' nu a avut nicio lectie incepand cu 15 mai');
                fara_lectii := fara_lectii + 1;
            END IF;
            --daca a avut lectii verificam daca clientii lui au avut examen
            FOR i IN 1..lectii.count LOOP
                isInVector := 0;
                id_cli := lectii(i).id_client;
                --verificam daca am procesat deja clientul
                FOR j IN 1..t.count LOOP
                    IF t(j) = id_cli THEN isInVector := 1;
                    END IF;
                END LOOP;
                --daca nu, il adaugam in vector
                IF isInVector = 0 THEN
                    t.extend();
                    t(t.count) := id_cli;
                END IF;
            END LOOP;
            FOR k IN 1..t.count LOOP
                --verificam daca clientul a dat examen si daca da afisam
--instructorul si clientul
                SELECT *
                BULK COLLECT INTO examm
                FROM (SELECT * FROM examen WHERE id_client = t(k));

                IF examm.count > 0 THEN
                    counter := counter + 1;
                    SELECT nume || ' ' ||prenume INTO nume_client
                        FROM client where id_client = t(k);
                    DBMS_OUTPUT.PUT_LINE('Instructor: ' ||v_instr.nume || ' '
 ||v_instr.prenume || ' // Client: ' || nume_client);
                END IF;

            END LOOP;
            --resetam vectorul
            t.trim(t.count);
        END LOOP;
    END LOOP;
    IF counter = 0 AND fara_lectii <> instructori THEN RAISE exceptie_zero;
    END IF;
    EXCEPTION
        WHEN exceptie_zero THEN
            DBMS_OUTPUT.PUT_LINE('Se pare ca restul instructorilor
au avut lectii, dar clientii lor nu au sustinut examenul!');
END p9;

PROCEDURE delete_table_data IS
BEGIN
     EXECUTE IMMEDIATE 'TRUNCATE TABLE examen';
     EXECUTE IMMEDIATE 'TRUNCATE TABLE lectie_auto';
     EXECUTE IMMEDIATE 'TRUNCATE TABLE lectie_teoretica_client';
     EXECUTE IMMEDIATE 'TRUNCATE TABLE lectie_teoretica';
     EXECUTE IMMEDIATE 'TRUNCATE TABLE instructor';
     EXECUTE IMMEDIATE 'TRUNCATE TABLE client';
     EXECUTE IMMEDIATE 'TRUNCATE TABLE scoala';
     EXECUTE IMMEDIATE 'TRUNCATE TABLE oras';
END delete_table_data;
```

```
PROCEDURE insert_default_table_data IS
BEGIN
    INSERT INTO oras VALUES (1, 'Bucuresti');
    INSERT INTO oras VALUES (2, 'Cluj');
    INSERT INTO oras VALUES (3, 'Constanta');

    INSERT INTO scoala VALUES (1, 'Teo', 1, '0753098561', 'inscrieri@vreaupermis.ro');
    INSERT INTO scoala VALUES (2, 'AutoBest', 1, '0760712663', 'office@scoalaautobest.ro');
    INSERT INTO scoala VALUES (3, 'Nelit', 1, '0727726252', 'contact@nelit.ro');
    INSERT INTO scoala VALUES (4, 'Rodna', 2, '0735187708', 'contact@scoalarodna.ro');
    INSERT INTO scoala VALUES (5, 'ToniAuto', 2, '0745990749', 'scoala@toniauto.ro');
    INSERT INTO scoala VALUES (6, 'Racareanu', 3, '0723227753', 'racareanu@auto.ro');

    INSERT INTO instructor VALUES (1, 'Dumitrescu', 'Mariana', 1, '0789653578', 3000);
    INSERT INTO instructor VALUES (2, 'Marinescu', 'Florin', 1, '0799009911', 2000);
    INSERT INTO instructor VALUES (3, 'Munteanu', 'Alexandru', 2, '0716991299', 2500);
    INSERT INTO instructor VALUES (4, 'Bivolaru', 'Theodor', 2, '0765443212', 2700);
    INSERT INTO instructor VALUES (5, 'Popescu', 'Gabriel', 2, '0744432990', 1700);
    INSERT INTO instructor VALUES (6, 'Popa', 'Viorel', 3, '078160978', 3500);
    INSERT INTO instructor VALUES (7, 'Anton', 'Alexandru', 4, '0712334456', 2300);
    INSERT INTO instructor VALUES (8, 'Ghencea', 'Antonio', 5, '0766123111', 2100);

    INSERT INTO client VALUES (1, 'Tudor', 'Maria', '0712331211');
    INSERT INTO client VALUES (2, 'Bina', 'Alexandru', '0752145689');
    INSERT INTO client VALUES (3, 'Costea', 'Vlad', '0727399272');
    INSERT INTO client VALUES (4, 'Savastre', 'Costel', '0723332172');
    INSERT INTO client VALUES (5, 'Dumitriu', 'Ioana', '0733442211');
    INSERT INTO client VALUES (6, 'Cerbu', 'Stefan', '0709379271');
    INSERT INTO client VALUES (7, 'Popescu', 'Gabriel', '0709373891');
    INSERT INTO client VALUES (8, 'Marin', 'Alina', '0755167367');
    INSERT INTO client VALUES (9, 'Enescu', 'Andreea', '0755163222');

    --Clientul 3 face 3 lectii cu instructorul 1 si 1 lectie cu instructorul 2
    INSERT INTO lectie_auto VALUES (1, 1, 3, TIMESTAMP '2020-10-10 12:00:00');
    INSERT INTO lectie_auto VALUES (2, 1, 3, TIMESTAMP '2020-10-12 12:00:00');
    INSERT INTO lectie_auto VALUES (3, 1, 3, TIMESTAMP '2020-10-14 12:00:00');
    INSERT INTO lectie_auto VALUES (4, 2, 3, TIMESTAMP '2020-10-16 12:00:00');
    --Clientul 2 face 2 lectii cu instructorul 2
    INSERT INTO lectie_auto VALUES (5, 2, 2, TIMESTAMP '2020-11-01 12:00:00');
    INSERT INTO lectie_auto VALUES (6, 2, 2, TIMESTAMP '2020-11-10 12:00:00');
    --Restul fac cate 1 sedinta cu cate 1 instructor
    INSERT INTO lectie_auto VALUES (7, 3, 1, TIMESTAMP '2020-03-12 12:00:00');
    INSERT INTO lectie_auto VALUES (8, 7, 4, TIMESTAMP '2020-04-13 12:00:00');
    INSERT INTO lectie_auto VALUES (9, 5, 5, TIMESTAMP '2020-05-14 12:00:00');
    INSERT INTO lectie_auto VALUES (10, 6, 6, TIMESTAMP '2020-03-21 12:00:00');
    INSERT INTO lectie_auto VALUES (11, 4, 7, TIMESTAMP '2020-06-20 12:00:00');
    INSERT INTO lectie_auto VALUES (12, 8, 8, TIMESTAMP '2020-10-11 12:00:00');
    INSERT INTO lectie_auto VALUES (13, 3, 9, TIMESTAMP '2020-12-11 12:00:00');

    INSERT INTO lectie_teoretica VALUES(1, 1, TIMESTAMP '2020-05-10 18:00:00');
    INSERT INTO lectie_teoretica VALUES(2, 3, TIMESTAMP '2020-06-10 18:00:00');
    INSERT INTO lectie_teoretica VALUES(3, 2, TIMESTAMP '2020-10-10 18:00:00');
    INSERT INTO lectie_teoretica VALUES(4, 1, TIMESTAMP '2020-11-10 18:00:00');
    INSERT INTO lectie_teoretica VALUES(5, 4, TIMESTAMP '2020-12-10 18:00:00');
    INSERT INTO lectie_teoretica VALUES(6, 1, TIMESTAMP '2020-03-10 18:00:00');

    INSERT INTO lectie_teoretica_client VALUES (3, 3);
    INSERT INTO lectie_teoretica_client VALUES (4, 2);
    INSERT INTO lectie_teoretica_client VALUES (6, 1);
    INSERT INTO lectie_teoretica_client VALUES (6, 6);
    INSERT INTO lectie_teoretica_client VALUES (1, 4);
    INSERT INTO lectie_teoretica_client VALUES (2, 4);
    INSERT INTO lectie_teoretica_client VALUES (6, 4);
    INSERT INTO lectie_teoretica_client VALUES (1, 5);
    INSERT INTO lectie_teoretica_client VALUES (2, 7);
    INSERT INTO lectie_teoretica_client VALUES (3, 8);
    INSERT INTO lectie_teoretica_client VALUES (5, 9);

    INSERT INTO examen values (1, 3, TIMESTAMP '2020-11-20 08:00:00');
    INSERT INTO examen values (2, 2, TIMESTAMP '2020-12-03 08:00:00');
    INSERT INTO examen values (3, 4, TIMESTAMP '2020-06-29 08:00:00');
    INSERT INTO examen values (4, 5, TIMESTAMP '2020-07-20 08:00:00');

END insert_default_table_data;

PROCEDURE reset_table IS
BEGIN
    pachet.delete_table_data();
    pachet.insert_default_table_data();
    DBMS_OUTPUT.PUT_LINE('Table was reset');
END reset_table;

END pachet;
/

BEGIN
    DBMS_OUTPUT.PUT_LINE('Salariul maxim pentru un instructor dupa marire cu
procentul ales este '|| pachet.f6(10, 2000, 3000));
END;
/

BEGIN
```

```
  pachet.p7();
END;
/

BEGIN
DBMS_OUTPUT.PUT_LINE(pachet.f8(4));
END;
/
BEGIN
DBMS_OUTPUT.PUT_LINE(pachet.f8(15));
END;
/

BEGIN
DBMS_OUTPUT.PUT_LINE(pachet.f8(-1));
END;
/
-- INSERT INTO lectie_teoretica_client VALUES (5, 2) ;
-- INSERT INTO lectie_teoretica_client VALUES (6, 2) ;
BEGIN
DBMS_OUTPUT.PUT_LINE(pachet.f8(2));
END;
/

BEGIN
    pachet.p9();
END;
/

BEGIN
    pachet.reset_table();
END;
/
```