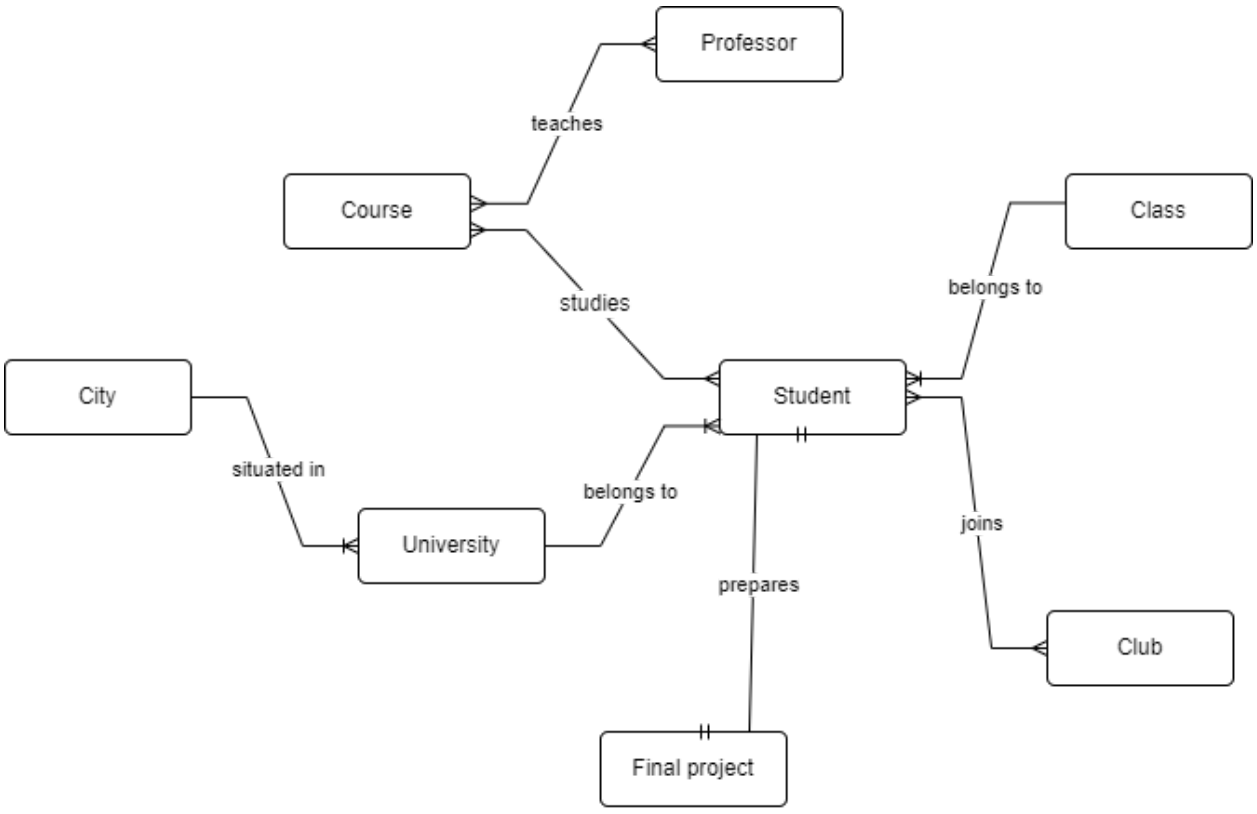
Proiect SGBD
- Trifan Robert-Gabriel 252 -

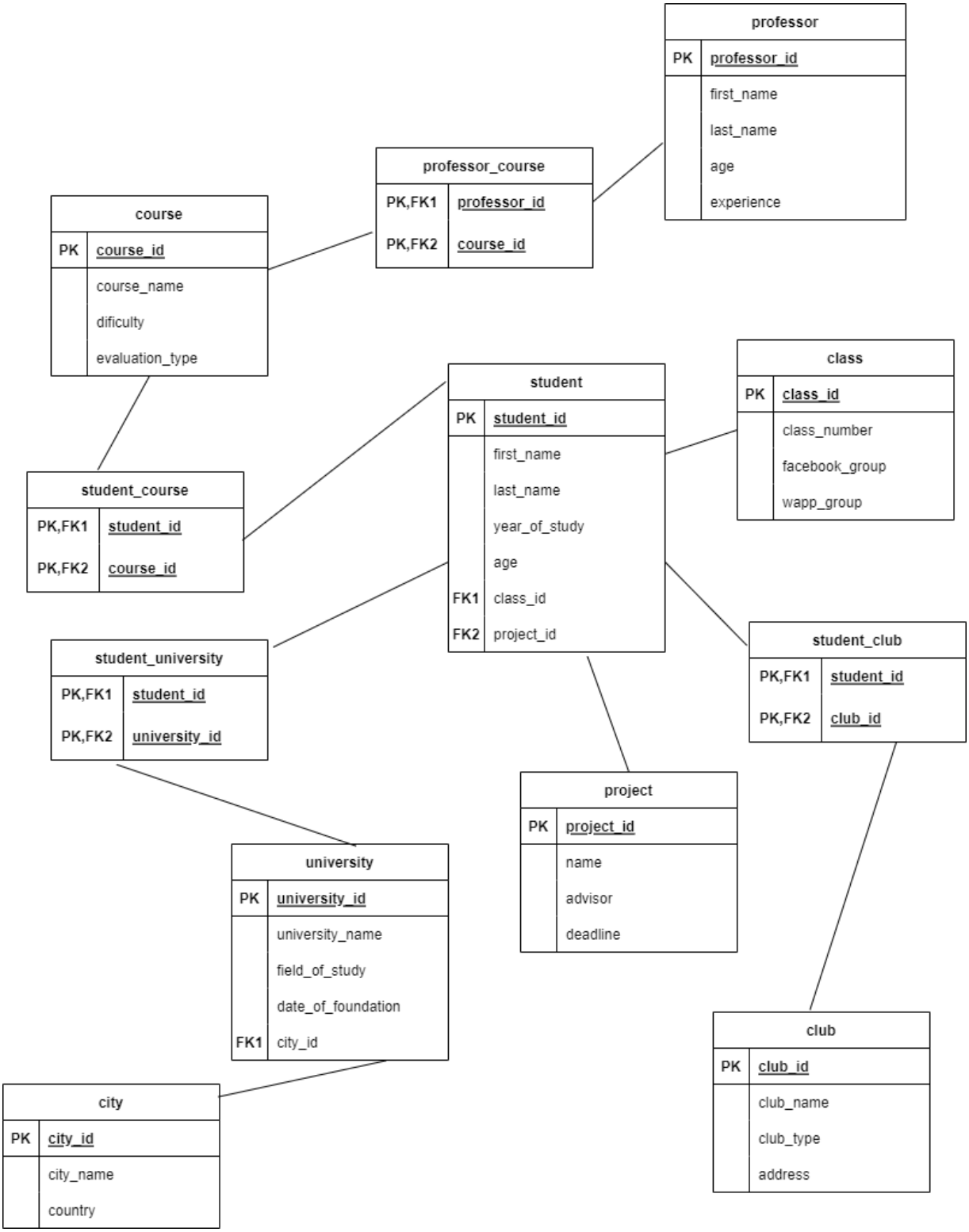
1. Prezentați pe scurt baza de date (utilitatea ei).

   În acest proiect îmi propun să implementez o bază de date care reține informații despre studenții din țară cum ar fi: universitatea și cursurile pe care le studiază precum și profesorii corespunzători. De asemenea, baza de date reține grupa și cluburile din care fac parte studenții și informații despre proiectul final.

2. Realizați diagrama entitate-relație (ERD).



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



4. Implementați în Oracle diagrama conceptuală realizată:

```
DROP TABLE student;
CREATE TABLE student(
    student_id INT NOT NULL,
    first_name VARCHAR(50) NOT NULL,
    last_name VARCHAR(50) NOT NULL,
    age INT NOT NULL,
    year_of_study INT NOT NULL,
    class_id INT NOT NULL,
    project_id INT NOT NULL,
    PRIMARY KEY(student_id),
    FOREIGN KEY(class_id) REFERENCES department(class_id) ON DELETE
CASCADE,
    FOREIGN KEY(project_id) REFERENCES final_project(project_id) ON DELETE
CASCADE
);


CREATE SEQUENCE student_sequence
  START WITH 1
  INCREMENT BY 1
  MINVALUE 1
  MAXVALUE 100
  NOCYCLE;

DROP TABLE department;
CREATE TABLE department(
    class_id INT NOT NULL,
    class_number INT NOT NULL,
    facebook_group VARCHAR(50) NOT NULL,
    wapp_group VARCHAR(50) NOT NULL,
    PRIMARY KEY(class_id)
);

DROP TABLE final_project;
CREATE TABLE final_project(
    project_id INT NOT NULL,
    project_name VARCHAR(50) NOT NULL,
    advisor VARCHAR(50) NOT NULL,
    deadline DATE NOT NULL,
    PRIMARY KEY(project_id)
```

```
);

DROP TABLE professor;
CREATE TABLE professor(
    professor_id INT NOT NULL,
    first_name VARCHAR(50) NOT NULL,
    last_name VARCHAR(50) NOT NULL,
    age INT NOT NULL,
    experience INT NOT NULL,
    PRIMARY KEY(professor_id)
);

DROP TABLE course;
CREATE TABLE course(
    course_id INT NOT NULL,
    course_name VARCHAR(50) NOT NULL,
    difficulty INT NOT NULL,
    evaluation_type VARCHAR(50) NOT NULL,
    PRIMARY KEY(course_id)
);


DROP TABLE university;
CREATE TABLE university(
    university_id INT NOT NULL,
    university_name VARCHAR(50) NOT NULL,
    field_of_study VARCHAR(50) NOT NULL,
    date_of_foundation DATE NOT NULL,
    city_id INT NOT NULL,
    PRIMARY KEY(university_id),
    FOREIGN KEY(city_id) REFERENCES city(city_id) ON DELETE CASCADE
);


DROP TABLE city;
CREATE TABLE city(
    city_id INT NOT NULL,
    city_name VARCHAR(50) NOT NULL,
    country VARCHAR(50) NOT NULL,
    PRIMARY KEY(city_id)
);

DROP TABLE club;
```

```sql
CREATE TABLE club(
    club_id INT NOT NULL,
    club_name VARCHAR(50) NOT NULL,
    club_type VARCHAR(50) NOT NULL,
    address VARCHAR(50) NOT NULL,
    PRIMARY KEY(club_id)
);

DROP TABLE student_club;
CREATE TABLE student_club(
    student_id INT NOT NULL,
    club_id INT NOT NULL,
    PRIMARY KEY(student_id, club_id),
    FOREIGN KEY(student_id) REFERENCES student(student_id) ON DELETE
CASCADE,
    FOREIGN KEY(club_id) REFERENCES club(club_id) ON DELETE CASCADE
);

DROP TABLE student_course;
CREATE TABLE student_course(
    student_id INT NOT NULL,
    course_id INT NOT NULL,
    PRIMARY KEY(student_id, course_id),
    FOREIGN KEY(student_id) REFERENCES student(student_id) ON DELETE
CASCADE,
    FOREIGN KEY(course_id) REFERENCES course(course_id) ON DELETE CASCADE
);

DROP TABLE professor_course;
CREATE TABLE professor_course(
    professor_id INT NOT NULL,
    course_id INT NOT NULL,
    PRIMARY KEY(professor_id, course_id),
    FOREIGN KEY(professor_id) REFERENCES professor(professor_id) ON DELETE
CASCADE,
    FOREIGN KEY(course_id) REFERENCES course(course_id) ON DELETE CASCADE
);

DROP TABLE student_university;
CREATE TABLE student_university(
    student_id INT NOT NULL,
    university_id INT NOT NULL,
    PRIMARY KEY(student_id, university_id),
```

```
    FOREIGN KEY(student_id) REFERENCES student(student_id) ON DELETE
CASCADE,
    FOREIGN KEY(university_id) REFERENCES university(university_id) ON
DELETE CASCADE
);
```

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

```
INSERT INTO city VALUES(1, 'București', 'Romania');
INSERT INTO city VALUES(2, 'Iasi', 'Romania');
INSERT INTO city VALUES(3, 'Cluj-Napoca', 'Romania');
INSERT INTO city VALUES(4, 'Oxford', 'UK');
INSERT INTO city VALUES(5, 'Cambridge', 'UK');
INSERT INTO city VALUES(6, 'London', 'UK');
INSERT INTO city VALUES(7, 'Boston', 'USA');
INSERT INTO city VALUES(8, 'New York', 'USA');
INSERT INTO city VALUES(9, 'Zurich', 'Switzerland');
INSERT INTO city VALUES(10, 'Paris', 'France');
INSERT INTO city VALUES(11, 'Berlin', 'Germany');

INSERT INTO club VALUES(1, 'Clubul Sportiv Fotbal', 'Fotbal', 'Strada
Universitatii, nr. 1');
INSERT INTO club VALUES(2, 'Clubul de Volei', 'Volei', 'Strada Unirii, nr.
2');
INSERT INTO club VALUES(3, 'Clubul de Tenis', 'Tenis', 'Strada Eternitatii,
nr. 3');
INSERT INTO club VALUES(4, 'Clubul de Baschet', 'Baschet', 'Strada
Libertatii, nr. 4');
INSERT INTO club VALUES(5, 'Clubul de Handbal', 'Handbal', 'Strada
Independentei, nr. 5');
INSERT INTO club VALUES(6, 'Clubul de Rugby', 'Rugby', 'Strada Unirii, nr.
6');
INSERT INTO club VALUES(7, 'Clubul de Sah', 'Sah', 'Strada Libertatii, nr.
7');
INSERT INTO club VALUES(8, 'Clubul de Istorie', 'Istorie', 'Strada
Universitatii, nr. 8');
INSERT INTO club VALUES(9, 'Clubul de Matematica', 'Matematica', 'Strada
Eternitatii, nr. 9');
INSERT INTO club VALUES(10, 'Clubul de Informatica', 'Informatica', 'Strada
Unirii, nr. 10');
```

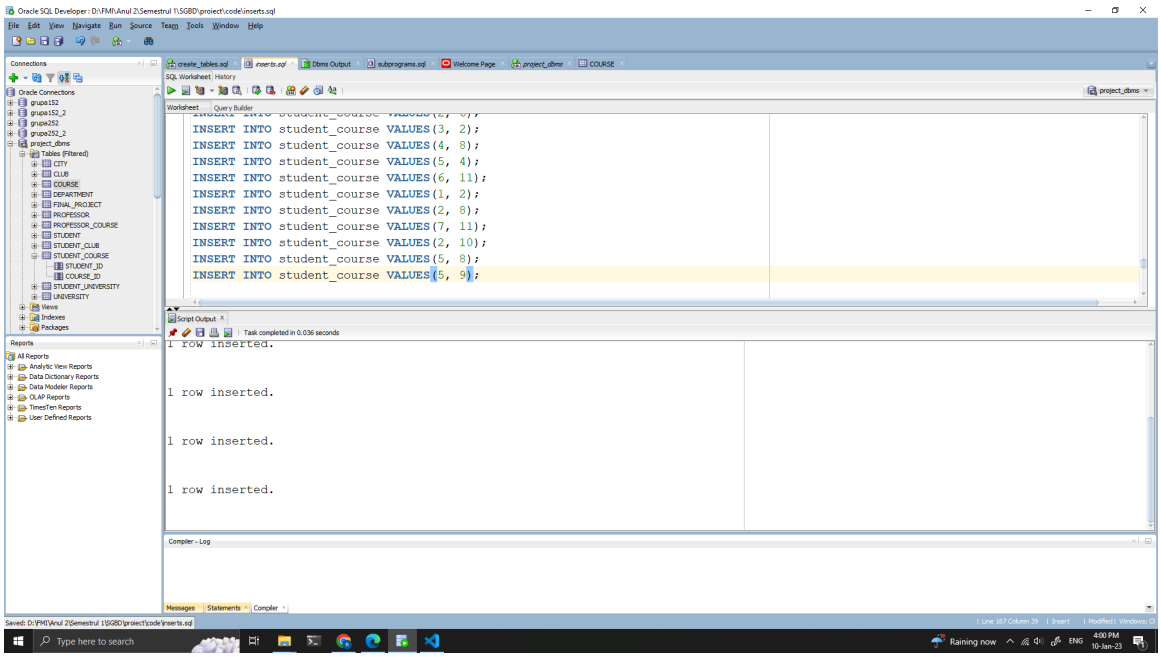
```
INSERT INTO club VALUES(11, 'Clubul de jmecheri', 'Informatica', 'Strada
Libertatii, nr. 11');
INSERT INTO club VALUES(12, 'Clubul de programatori', 'Informatica',
'Strada Universitatii, nr. 12');
INSERT INTO club VALUES(13, 'Clubul de hackathoane', 'Informatica', 'Strada
Eternitatii, nr. 13');

INSERT INTO university VALUES(1, 'Universitatea Politehnica Bucuresti',
'Engineering', TO_DATE('1920-06-10', 'YYYY-MM-DD'), 1);
INSERT INTO university VALUES(2, 'Facultatea de Matematica si Informatica',
'Computer Science', TO_DATE('1878-02-21', 'YYYY-MM-DD'), 1);
INSERT INTO university VALUES(3, 'Facultatea de Drept', 'Law',
TO_DATE('1783-04-06', 'YYYY-MM-DD'), 1);
INSERT INTO university VALUES(4, 'Universitatea Alexandru Ioan Cuza',
'Computer Science', TO_DATE('1860-01-01', 'YYYY-MM-DD'), 2);
INSERT INTO university VALUES(5, 'University of Oxford', 'Science',
TO_DATE('1096-02-19', 'YYYY-MM-DD'), 4);
INSERT INTO university VALUES(6, 'University of Cambridge', 'Science',
TO_DATE('1209-02-19', 'YYYY-MM-DD'), 5);
INSERT INTO university VALUES(7, 'Harvard University', 'Science',
TO_DATE('1636-02-19', 'YYYY-MM-DD'), 7);
INSERT INTO university VALUES(8, 'Massachusetts Institute of Technology',
'Science', TO_DATE('1861-02-19', 'YYYY-MM-DD'), 7);
INSERT INTO university VALUES(9, 'University of Zurich', 'Science',
TO_DATE('1833-02-19', 'YYYY-MM-DD'), 9);
INSERT INTO university VALUES(10, 'University of Paris', 'Science',
TO_DATE('1150-02-19', 'YYYY-MM-DD'), 10);
INSERT INTO university VALUES(11, 'University of Berlin', 'Law',
TO_DATE('1809-02-19', 'YYYY-MM-DD'), 11);
INSERT INTO university VALUES(12, 'University of London', 'Medicine',
TO_DATE('1836-02-19', 'YYYY-MM-DD'), 6);
INSERT INTO university VALUES(13, 'University of New York', 'Medicine',
TO_DATE('1831-02-19', 'YYYY-MM-DD'), 8);
INSERT INTO university VALUES(14, 'Carol Davila', 'Medicine and Pharmacy',
TO_DATE('1636-02-19', 'YYYY-MM-DD'), 1);

INSERT INTO department VALUES(1, '131', 'fb_group_131', 'wapp_group_131');
INSERT INTO department VALUES(2, '132', 'fb_group_132', 'wapp_group_132');
INSERT INTO department VALUES(3, '133', 'fb_group_133', 'wapp_group_133');
INSERT INTO department VALUES(4, '134', 'fb_group_134', 'wapp_group_134');
INSERT INTO department VALUES(5, '141', 'fb_group_141', 'wapp_group_141');
INSERT INTO department VALUES(6, '142', 'fb_group_142', 'wapp_group_142');
INSERT INTO department VALUES(7, '143', 'fb_group_143', 'wapp_group_143');
```

```
INSERT INTO department VALUES(8, '144', 'fb_group_144', 'wapp_group_144');
INSERT INTO department VALUES(9, '151', 'fb_group_151', 'wapp_group_151');
INSERT INTO department VALUES(10, '152', 'fb_group_152', 'wapp_group_152');
INSERT INTO department VALUES(11, '231', 'fb_group_231', 'wapp_group_231');
INSERT INTO department VALUES(12, '232', 'fb_group_232', 'wapp_group_232');
INSERT INTO department VALUES(13, '233', 'fb_group_233', 'wapp_group_233');
INSERT INTO department VALUES(14, '234', 'fb_group_234', 'wapp_group_234');
INSERT INTO department VALUES(15, '241', 'fb_group_241', 'wapp_group_241');
INSERT INTO department VALUES(16, '242', 'fb_group_242', 'wapp_group_242');
INSERT INTO department VALUES(17, '243', 'fb_group_243', 'wapp_group_243');
INSERT INTO department VALUES(18, '244', 'fb_group_244', 'wapp_group_244');
INSERT INTO department VALUES(19, '251', 'fb_group_251', 'wapp_group_251');
INSERT INTO department VALUES(20, '252', 'fb_group_252', 'wapp_group_252');

INSERT INTO course VALUES(1, 'Algebra', 5, 'Examen');
INSERT INTO course VALUES(2, 'Analiza', 2, 'Examen');
INSERT INTO course VALUES(3, 'Geometrie', 3, 'Examen');
INSERT INTO course VALUES(4, 'Logica', 4, 'Examen');
INSERT INTO course VALUES(5, 'Baze de date', 3, 'Proiect');
INSERT INTO course VALUES(6, 'Gandire critica', 1, 'Proiect');
INSERT INTO course VALUES(7, 'LFA', 5, 'Examen');
INSERT INTO course VALUES(8, 'OOP', 4, 'Examen');
INSERT INTO course VALUES(9, 'DAW', 2, 'Proiect');
INSERT INTO course VALUES(10, 'SGBD', 3, 'Proiect');
INSERT INTO course VALUES(11, 'TW', 4, 'Proiect');
INSERT INTO course VALUES(12, 'GAL', 3, 'Examen');
INSERT INTO course VALUES(13, 'PAO', 4, 'Examen');

INSERT INTO final_project VALUES(1, 'Crowd knowledge contribution', 'Alexe
Bogdan', TO_DATE('2018-06-10', 'YYYY-MM-DD'));
INSERT INTO final_project VALUES(2, 'Digital school groups', 'Boriga Radu',
TO_DATE('2019-06-10', 'YYYY-MM-DD'));
INSERT INTO final_project VALUES(3, 'Micro-social platform', 'Paun Andrei',
TO_DATE('2022-06-10', 'YYYY-MM-DD'));
INSERT INTO final_project VALUES(4, 'Open discussion', 'Dobrovat Anca',
TO_DATE('2018-06-10', 'YYYY-MM-DD'));
INSERT INTO final_project VALUES(5, 'Social bookmarking', 'Ionescu Radu',
TO_DATE('2019-06-10', 'YYYY-MM-DD'));
INSERT INTO final_project VALUES(6, 'Online shop', 'Cimpean Iulian',
TO_DATE('2022-06-10', 'YYYY-MM-DD'));
INSERT INTO final_project VALUES(7, 'Task management', 'Alexe Bogdan',
TO_DATE('2022-06-10', 'YYYY-MM-DD'));
INSERT INTO final_project VALUES(8, 'Music platform', 'Bill Gates',
```

```
TO_DATE('2022-03-23', 'YYYY-MM-DD'));
INSERT INTO final_project VALUES(9, 'Video platform', 'Klaus Iohannis',
TO_DATE('2023-02-18', 'YYYY-MM-DD'));
INSERT INTO final_project VALUES(10, 'Audio platform', 'Joe Biden',
TO_DATE('2023-01-02', 'YYYY-MM-DD'));

INSERT INTO student VALUES(1, 'Robert', 'Trifan', 20, 2, 20, 3);
INSERT INTO student VALUES(2, 'Alexandru', 'Pascu', 20, 2, 20, 7);
INSERT INTO student VALUES(3, 'Anna', 'Pecheanu', 18, 1, 10, 4);
INSERT INTO student VALUES(4, 'Albert', 'Balauta', 20, 1, 9, 2);
INSERT INTO student VALUES(5, 'Radu', 'Nedelcu', 19, 2, 19, 1);
INSERT INTO student VALUES(6, 'Tudor', 'Haulica', 19, 2, 18, 5);
INSERT INTO student VALUES(7, 'Andrei', 'Murica', 18, 1, 2, 6);
INSERT INTO student VALUES(8, 'Radu', 'Pop', 21, 2, 20, 8);
INSERT INTO student VALUES(9, 'Cornel', 'Frunza', 20, 1, 19, 9);
INSERT INTO student VALUES(10, 'Gigel', 'Gheorghe', 19, 2, 18, 10);

INSERT INTO student_club VALUES(1, 11);
INSERT INTO student_club VALUES(2, 1);
INSERT INTO student_club VALUES(3, 3);
INSERT INTO student_club VALUES(4, 1);
INSERT INTO student_club VALUES(5, 6);
INSERT INTO student_club VALUES(6, 4);
INSERT INTO student_club VALUES(1, 1);
INSERT INTO student_club VALUES(2, 9);
INSERT INTO student_club VALUES(3, 9);
INSERT INTO student_club VALUES(4, 13);
INSERT INTO student_club VALUES(5, 12);
INSERT INTO student_club VALUES(6, 8);
INSERT INTO student_club VALUES(1, 7);
INSERT INTO student_club VALUES(2, 5);
INSERT INTO student_club VALUES(3, 2);
INSERT INTO student_club VALUES(4, 10);
INSERT INTO student_club VALUES(5, 5);
INSERT INTO student_club VALUES(6, 3);

INSERT INTO professor VALUES(1, 'Bogdan', 'Alexe', 30, 10);
INSERT INTO professor VALUES(2, 'Radu', 'Boriga', 45, 20);
INSERT INTO professor VALUES(3, 'Andrei', 'Paun', 40, 7);
INSERT INTO professor VALUES(4, 'Anca', 'Dobrovat', 35, 5);
INSERT INTO professor VALUES(5, 'Radu', 'Ionescu', 35, 12);
INSERT INTO professor VALUES(6, 'Iulian', 'Cimpean', 30, 8);
INSERT INTO professor VALUES(7, 'Laurentiu', 'Leustean', 60, 30);
```

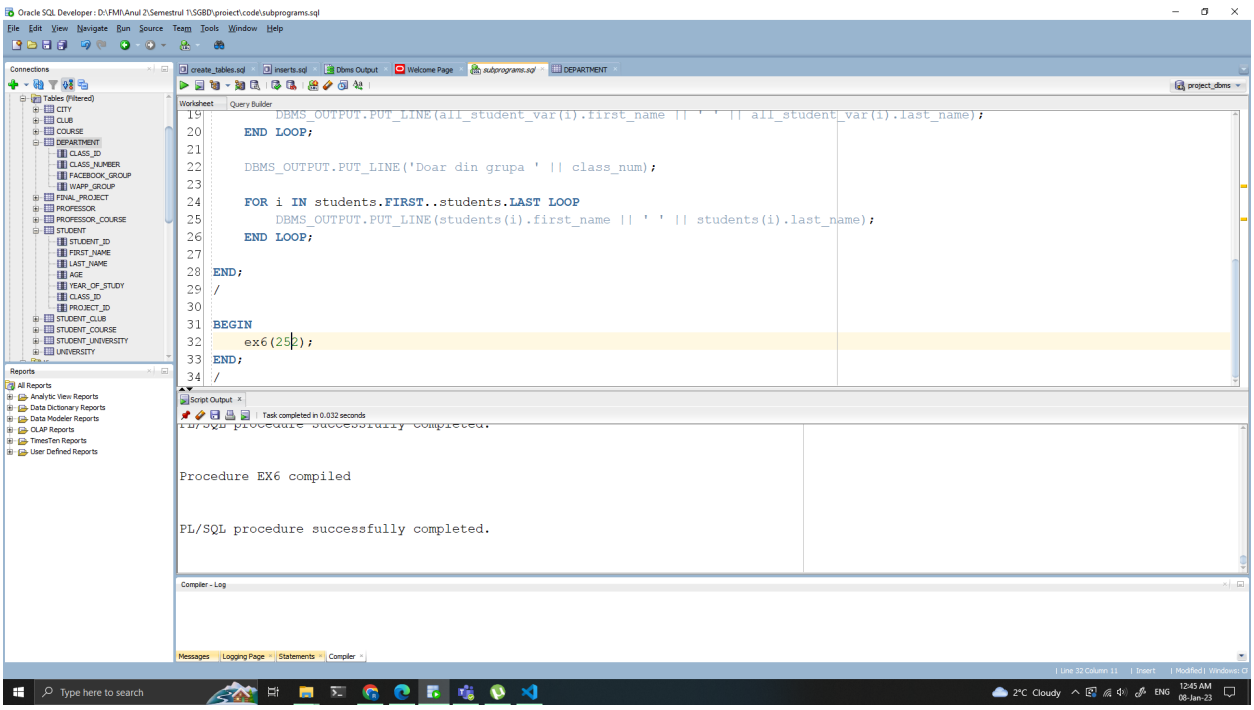
```
INSERT INTO professor VALUES(8, 'Iulia', 'Hirica', 55, 35);
INSERT INTO professor VALUES(9, 'Cezara', 'Benegui', 27, 3);
INSERT INTO professor VALUES(10, 'Marius', 'Dumitran', 37, 12);

INSERT INTO professor_course VALUES(1, 4);
INSERT INTO professor_course VALUES(2, 7);
INSERT INTO professor_course VALUES(3, 2);
INSERT INTO professor_course VALUES(4, 1);
INSERT INTO professor_course VALUES(5, 3);
INSERT INTO professor_course VALUES(6, 10);
INSERT INTO professor_course VALUES(7, 11);
INSERT INTO professor_course VALUES(8, 5);
INSERT INTO professor_course VALUES(9, 6);
INSERT INTO professor_course VALUES(10, 8);
INSERT INTO professor_course VALUES(1, 2);
INSERT INTO professor_course VALUES(2, 10);
INSERT INTO professor_course VALUES(3, 6);
INSERT INTO professor_course VALUES(4, 7);
INSERT INTO professor_course VALUES(5, 3);
INSERT INTO professor_course VALUES(6, 2);
INSERT INTO professor_course VALUES(7, 1);
INSERT INTO professor_course VALUES(8, 4);
INSERT INTO professor_course VALUES(9, 5);
INSERT INTO professor_course VALUES(10, 5);

INSERT INTO student_course VALUES(1, 5);
INSERT INTO student_course VALUES(2, 9);
INSERT INTO student_course VALUES(3, 11);
INSERT INTO student_course VALUES(4, 1);
INSERT INTO student_course VALUES(5, 7);
INSERT INTO student_course VALUES(6, 3);
INSERT INTO student_course VALUES(1, 10);
INSERT INTO student_course VALUES(2, 6);
INSERT INTO student_course VALUES(3, 2);
INSERT INTO student_course VALUES(4, 8);
INSERT INTO student_course VALUES(5, 4);
INSERT INTO student_course VALUES(6, 11);
INSERT INTO student_course VALUES(1, 2);
INSERT INTO student_course VALUES(2, 8);
INSERT INTO student_course VALUES(7, 11);
INSERT INTO student_course VALUES(2, 10);
INSERT INTO student_course VALUES(5, 8);
INSERT INTO student_course VALUES(5, 9);
```

```
INSERT INTO student_course VALUES(10, 12);

INSERT INTO student_university VALUES(1, 8);
INSERT INTO student_university VALUES(2, 3);
INSERT INTO student_university VALUES(3, 5);
INSERT INTO student_university VALUES(4, 14);
INSERT INTO student_university VALUES(5, 7);
INSERT INTO student_university VALUES(6, 9);
INSERT INTO student_university VALUES(1, 9);
INSERT INTO student_university VALUES(2, 5);
INSERT INTO student_university VALUES(3, 13);
INSERT INTO student_university VALUES(4, 12);
INSERT INTO student_university VALUES(5, 2);
INSERT INTO student_university VALUES(6, 10);
INSERT INTO student_university VALUES(7, 11);
INSERT INTO student_university VALUES(1, 1);
INSERT INTO student_university VALUES(2, 4);
INSERT INTO student_university VALUES(3, 6);
INSERT INTO student_university VALUES(4, 7);
INSERT INTO student_university VALUES(5, 8);
INSERT INTO student_university VALUES(6, 11);
INSERT INTO student_university VALUES(7, 12);

COMMIT;
ROLLBACK;
```



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

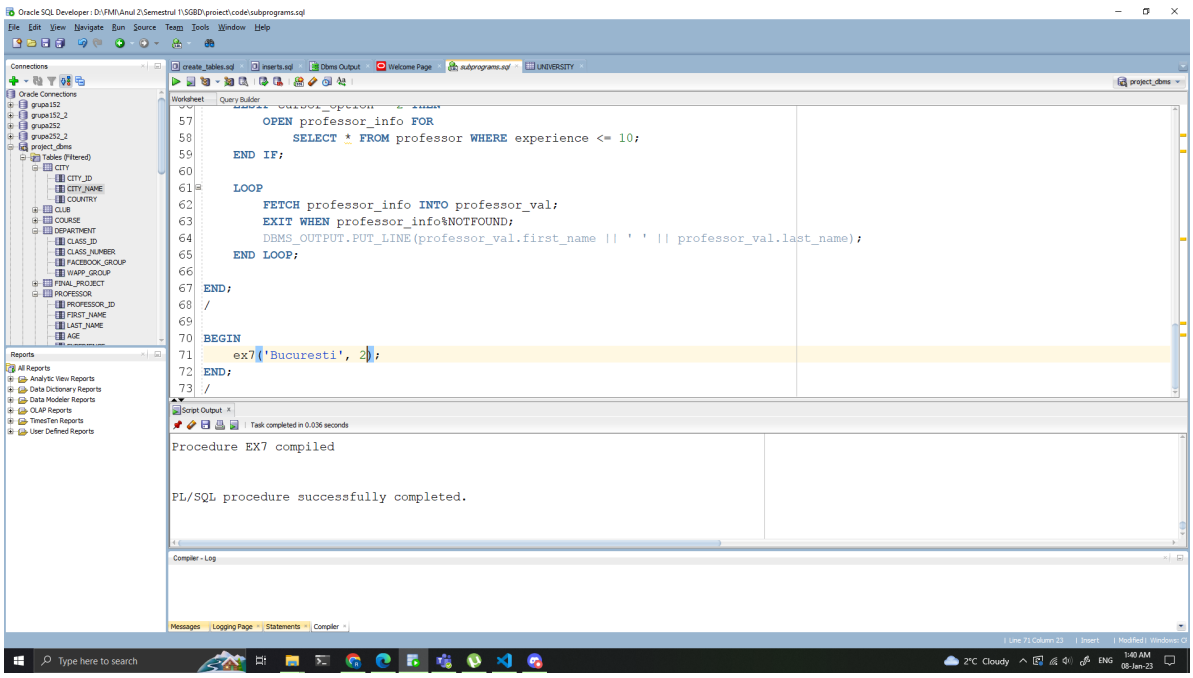
```
-- Print all students and all students from a given class.

CREATE OR REPLACE PROCEDURE print_students_of_dept (class_num
department.class_number%TYPE) IS
    TYPE dept_students IS TABLE OF student%ROWTYPE INDEX BY PLS_INTEGER;
    TYPE all_students IS VARRAY(100) OF student%ROWTYPE;
    all_student_var all_students;
    students dept_students;
    curr_class_id department.class_id%TYPE;
BEGIN
    SELECT * BULK COLLECT INTO all_student_var FROM student;

    SELECT class_id INTO curr_class_id FROM department WHERE class_number =
class_num;

    SELECT * BULK COLLECT INTO students FROM student WHERE class_id =
curr_class_id;

    FOR i IN students.FIRST..students.LAST LOOP
        DBMS_OUTPUT.PUT_LINE(students(i).first_name);
    END LOOP;

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

    FOR i IN all_student_var.FIRST..all_student_var.LAST LOOP
        DBMS_OUTPUT.PUT_LINE(all_student_var(i).first_name || ' ' ||
all_student_var(i).last_name);
    END LOOP;
END;
/

BEGIN
    print_students_of_dept(251);
END;
/
```



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

```
-- cursor parametrizat pentru universitățile dintr-un oraș dat ca parametru
-- cursor dinamic pentru profesorii cu experiență mai mare sau mai mica de
10 ani

CREATE OR REPLACE PROCEDURE ex7 (city_name_par city.city_name%TYPE,
cursor_option NUMBER) IS
    CURSOR univ_from_city (city_id_par city.city_id%TYPE) IS
        SELECT * FROM university WHERE city_id = city_id_par;
    TYPE tip_cursor IS REF CURSOR RETURN professor%ROWTYPE;
    professor_info tip_cursor;
    city_id_var city.city_id%TYPE;
    val university%ROWTYPE;
    professor_val professor%ROWTYPE;
BEGIN
    SELECT city_id INTO city_id_var FROM city WHERE city_name =
city_name_par;
    OPEN univ_from_city(city_id_var);
```

```
    LOOP
        FETCH univ_from_city INTO val;
        EXIT WHEN univ_from_city%NOTFOUND;
        DBMS_OUTPUT.PUT_LINE(val.university_name);
    END LOOP;
    CLOSE univ_from_city;

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

    IF cursor_option = 1 THEN
        -- all professors that have more than 10 years of experience
        OPEN professor_info FOR
            SELECT * FROM professor WHERE experience > 10;
    ELSIF cursor_option = 2 THEN
        OPEN professor_info FOR
            SELECT * FROM professor WHERE experience <= 10;
    END IF;

    LOOP
        FETCH professor_info INTO professor_val;
        EXIT WHEN professor_info%NOTFOUND;
        DBMS_OUTPUT.PUT_LINE(professor_val.first_name || ' ' ||
professor_val.last_name);
    END LOOP;

END;
/

BEGIN
    ex7('București', 2);
END;
/
```

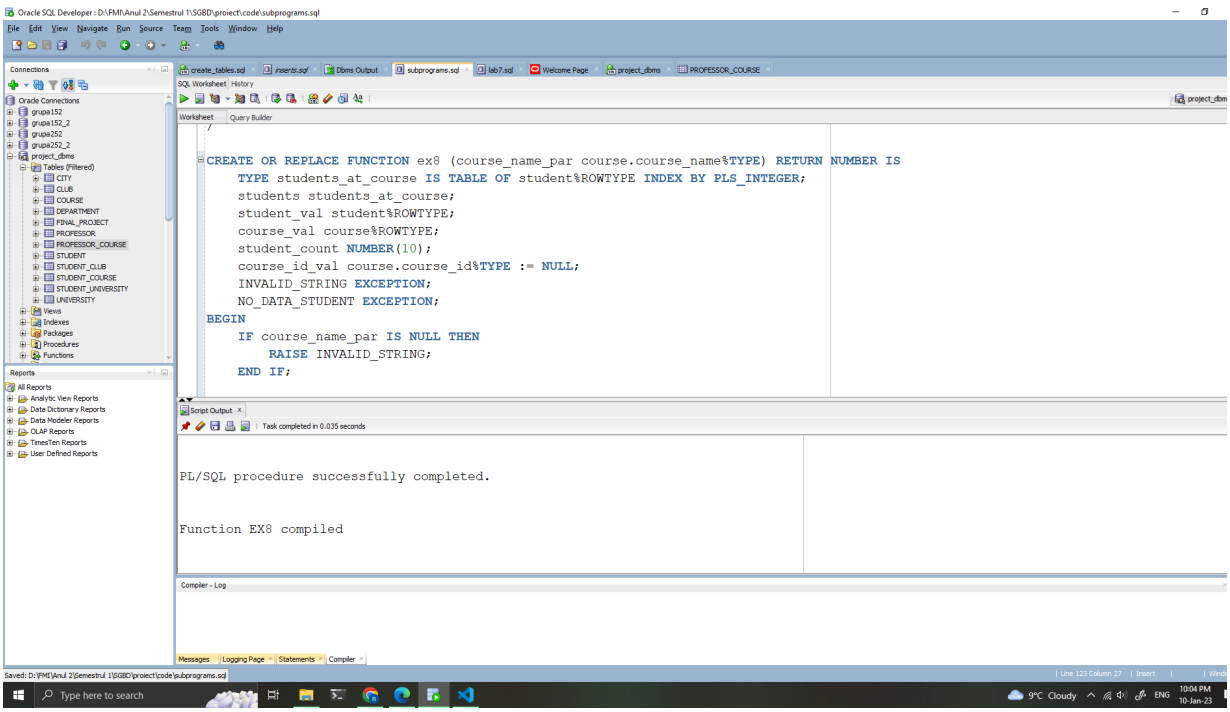


8. Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat independent de tip funcție care să utilizeze într-o singură comandă SQL 3 dintre tabelele definite. Definiți minim 2 excepții. Apelați subprogramul astfel încât să evidențiați toate cazurile tratate.

```
-- INVALID_STRING exceptie definita pentru parametru dat gresit
-- NO_DATA_STUDENT exceptie definita pentru cazul in care nu sunt studenti
la o anumita materie
-- functie care primeste ca parametru numele unei materii si returneaza
numarul de studenti la acea materie
-- cele 3 tabele implicate sunt: student, student_course, course

CREATE OR REPLACE FUNCTION ex8 (course_name_par course.course_name%TYPE)
RETURN NUMBER IS
    TYPE students_at_course IS TABLE OF student%ROWTYPE INDEX BY
PLS_INTEGER;
    students students_at_course;
    student_val student%ROWTYPE;
    course_val course%ROWTYPE;
    student_count NUMBER(10);
    course_id_val course.course_id%TYPE := NULL;
    INVALID_STRING EXCEPTION;
    NO_DATA_STUDENT EXCEPTION;
```

```
BEGIN
    IF course_name_par IS NULL THEN
        RAISE INVALID_STRING;
    END IF;

    SELECT course_id INTO course_id_val FROM course WHERE course_name =
course_name_par;

    SELECT s.* BULK COLLECT INTO students FROM student s
    JOIN student_course sc ON sc.student_id = s.student_id
    JOIN course c ON c.course_id = sc.course_id
    WHERE c.course_id = course_id_val;

    IF students.COUNT = 0 THEN
        RAISE NO_DATA_STUDENT;
    END IF;

    student_count := students.COUNT;

    FOR i IN students.FIRST..students.LAST LOOP
        student_val := students(i);
        DBMS_OUTPUT.PUT_LINE(student_val.first_name || ' ' ||
student_val.last_name);
    END LOOP;

    RETURN student_count;

    EXCEPTION
        WHEN INVALID_STRING THEN
            DBMS_OUTPUT.PUT_LINE('Invalid parameter');
            RETURN -1;
        WHEN TOO_MANY_ROWS THEN
            DBMS_OUTPUT.PUT_LINE('Multiple courses with the same name');
            RETURN -1;
        WHEN NO_DATA_STUDENT THEN
            DBMS_OUTPUT.PUT_LINE('No students for course with id ' ||
course_id_val);
            RETURN -1;
        WHEN NO_DATA_FOUND THEN
            DBMS_OUTPUT.PUT_LINE('No course with name ' || course_name_par
|| ' found');
            RETURN -1;
        WHEN OTHERS THEN
```

```
            DBMS_OUTPUT.PUT_LINE('Other exception');
            RETURN -1;
END;
/

BEGIN
    DBMS_OUTPUT.PUT_LINE(ex8(NULL));
    DBMS_OUTPUT.PUT_LINE(ex8('TW'));
    DBMS_OUTPUT.PUT_LINE(ex8('GAL'));
    DBMS_OUTPUT.PUT_LINE(ex8('PF'));
END;
/
```

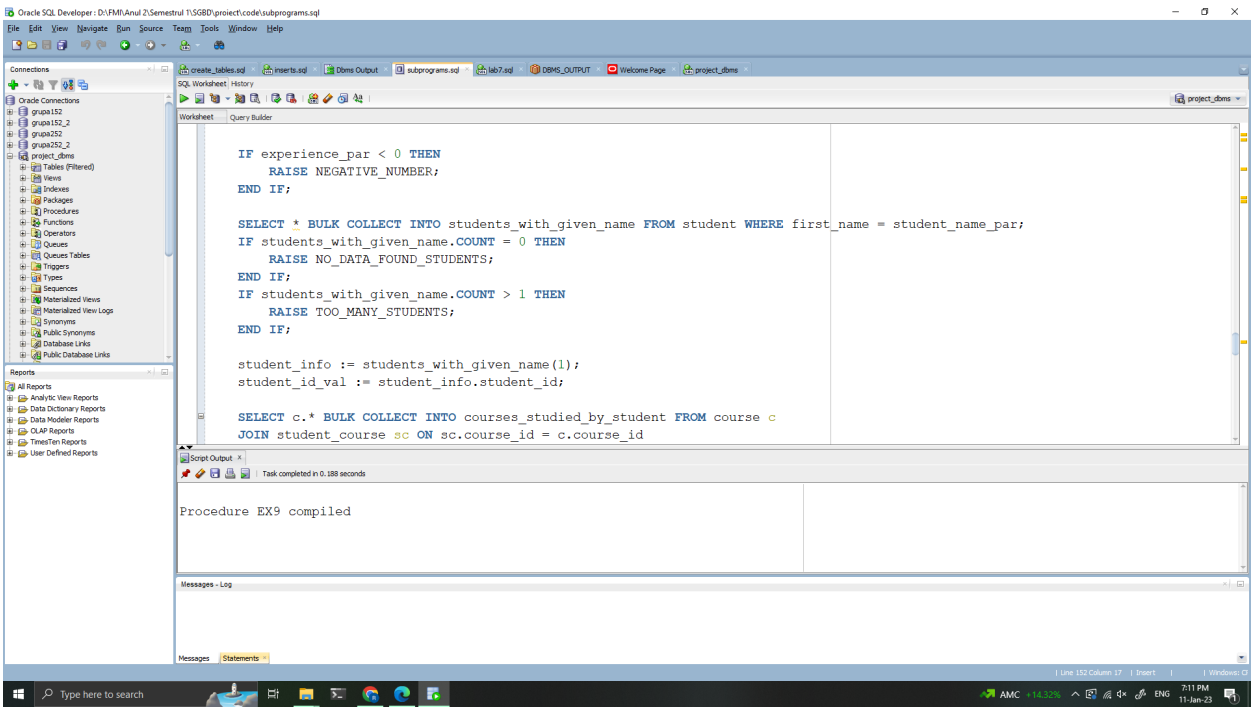


9. Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat independent de tip procedură care să utilizeze într-o singură comandă SQL 5 dintre tabelele definite. Tratați toate excepțiile care pot apărea, incluzând excepțiile NO_DATA_FOUND și TOO_MANY_ROWS. Apelați subprogramul astfel încât să evidențiați toate cazurile tratate.

```
-- INVALID_PARAMETER exceptie definita pentru parametrii dati gresit
-- NEGATIVE_NUMBER exceptie definita pentru cazul in care numarul de ani de
experienta este negativ
-- NO_DATA_FOUND_STUDENTS exceptie definita pentru cazul in care nu exista
studenti cu numele dat ca parametru
-- TOO_MANY_STUDENTS exceptie definita pentru cazul in care exista mai
multi studenti cu numele dat ca parametru
-- NO_DATA_COURSES exceptie definita pentru cazul in care nu exista cursuri
la care sa participe studentul cu numele dat ca parametru
-- NO_DATA_PROFESSORS exceptie definita pentru cazul in care nu exista
profesori care sa predea la cursurile la care participa studentul cu numele
dat ca parametru
-- OTHERS exceptia generala
-- procedura care primeste ca parametrii numele unui student si numarul de
ani de experienta si returneaza numele si prenumele profesorilor care au
mai mult de numarul
-- de ani de experienta dat ca parametru si care preda la cursurile la care
participa studentul cu numele dat ca parametru
-- cele 5 tabele implicate sunt: professor, professor_course, course,
student_course, student

CREATE OR REPLACE PROCEDURE ex9 (student_name_par student.first_name%TYPE,
experience_par NUMBER) IS
    TYPE student_type IS TABLE OF student%ROWTYPE INDEX BY PLS_INTEGER;
    TYPE course_type IS TABLE OF course%ROWTYPE INDEX BY PLS_INTEGER;
    TYPE professor_type IS TABLE OF professor%ROWTYPE INDEX BY PLS_INTEGER;
    courses_studied_by_student course_type;
    students_with_given_name student_type;
    student_id_val student.student_id%TYPE;
    student_info student%ROWTYPE;
    professor_info professor_type;
    answer professor_type;
    professor_val professor%ROWTYPE;
    INVALID_PARAMETER EXCEPTION;
    NO_DATA_FOUND_STUDENTS EXCEPTION;
    TOO_MANY_STUDENTS EXCEPTION;
    NO_DATA_COURSES EXCEPTION;
    NO_DATA_PROFESSORS EXCEPTION;
    NEGATIVE_NUMBER EXCEPTION;
BEGIN
    IF student_name_par IS NULL OR student_name_par = '' OR experience_par
IS NULL THEN
        RAISE INVALID_PARAMETER;
```

```
    END IF;

    IF experience_par < 0 THEN
        RAISE NEGATIVE_NUMBER;
    END IF;

    SELECT * BULK COLLECT INTO students_with_given_name FROM student WHERE
first_name = student_name_par;
    IF students_with_given_name.COUNT = 0 THEN
        RAISE NO_DATA_FOUND_STUDENTS;
    END IF;
    IF students_with_given_name.COUNT > 1 THEN
        RAISE TOO_MANY_STUDENTS;
    END IF;

    student_info := students_with_given_name(1);
    student_id_val := student_info.student_id;

    SELECT c.* BULK COLLECT INTO courses_studied_by_student FROM course c
    JOIN student_course sc ON sc.course_id = c.course_id
    JOIN student s ON s.student_id = sc.student_id
    WHERE s.student_id = student_id_val;

    IF courses_studied_by_student.COUNT = 0 THEN
        RAISE NO_DATA_COURSES;
    END IF;

    SELECT p.* BULK COLLECT INTO professor_info FROM professor p
    JOIN professor_course pc ON pc.professor_id = p.professor_id
    JOIN course c ON c.course_id = pc.course_id
    JOIN student_course sc ON sc.course_id = c.course_id
    JOIN student s ON s.student_id = sc.student_id
    WHERE s.student_id = student_id_val;

    IF professor_info.COUNT = 0 THEN
        RAISE NO_DATA_PROFESSORS;
    END IF;

    FOR i IN professor_info.FIRST..professor_info.LAST LOOP
        professor_val := professor_info(i);
        IF professor_val.experience >= experience_par THEN
            answer(i) := professor_val;
        END IF;
```

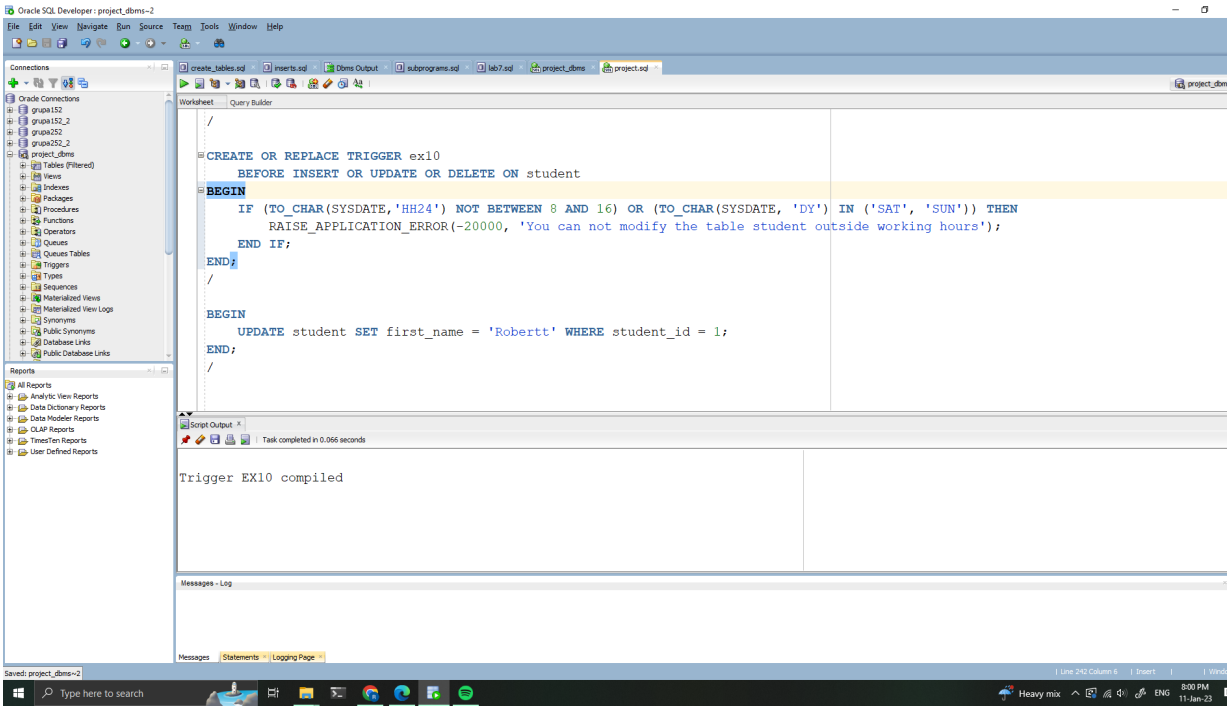
```
    END LOOP;


    FOR i IN answer.FIRST..answer.LAST LOOP
        professor_val := answer(i);
        DBMS_OUTPUT.PUT_LINE(professor_val.first_name || ' ' ||
professor_val.last_name);
    END LOOP;


    EXCEPTION
        WHEN INVALID_PARAMETER THEN
            DBMS_OUTPUT.PUT_LINE('Invalid parameter');
        WHEN NEGATIVE_NUMBER THEN
            DBMS_OUTPUT.PUT_LINE('Negative experience number');
        WHEN TOO_MANY_STUDENTS THEN
            DBMS_OUTPUT.PUT_LINE('Multiple students with the same name');
        WHEN NO_DATA_FOUND_STUDENTS THEN
            DBMS_OUTPUT.PUT_LINE('No students with name ' ||
student_name_par || ' found');
        WHEN NO_DATA_COURSES THEN
            DBMS_OUTPUT.PUT_LINE('No courses for student with id ' ||
student_id_val);
        WHEN NO_DATA_PROFESSORS THEN
            DBMS_OUTPUT.PUT_LINE('No professors for student with id ' ||
student_id_val);
        WHEN OTHERS THEN
            DBMS_OUTPUT.PUT_LINE('code error ' || SQLCODE);
            DBMS_OUTPUT.PUT_LINE('message error ' || SQLERRM);

END;
/

BEGIN
    ex9('Radu', 10); -- TOO_MANY_STUDENTS
    ex9('John', 10); -- NO_DATA_FOUND_STUDENTS
    ex9('Robert', -10); -- NEGATIVE_NUMBER
    ex9('', NULL); -- INVALID_PARAMETER
    ex9('Cornel', 10); -- NO_DATA_COURSES
    ex9('Gigel', 10); -- NO_DATA_PROFESSORS
    ex9('Robert', 10);
END;
/
```



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

```
-- trigger de tip LMD la nivel de comanda
-- Create a trigger that will not allow any modification of the table
student outside working hours (8-16) and on weekends.

CREATE OR REPLACE TRIGGER ex10
    BEFORE INSERT OR UPDATE OR DELETE ON student
BEGIN
    IF (TO_CHAR(SYSDATE,'HH24') NOT BETWEEN 8 AND 16) OR (TO_CHAR(SYSDATE,
'DY') IN ('SAT', 'SUN')) THEN
        -- RAISE_APPLICATION_ERROR(-20000, 'You can not modify the table
student outside working hours');
        IF INSERTING THEN
            RAISE_APPLICATION_ERROR(-20001,'Inserarea in tabel este permisa
doar in timpul programului de lucru!');
        ELSIF DELETING THEN
            RAISE_APPLICATION_ERROR(-20002,'Stergerea este permisa doar in
timpul programului de lucru!');
        ELSE
            RAISE_APPLICATION_ERROR(-20003,'Actualizarile sunt permise doar
in timpul programului de lucru!');
        END IF;
```

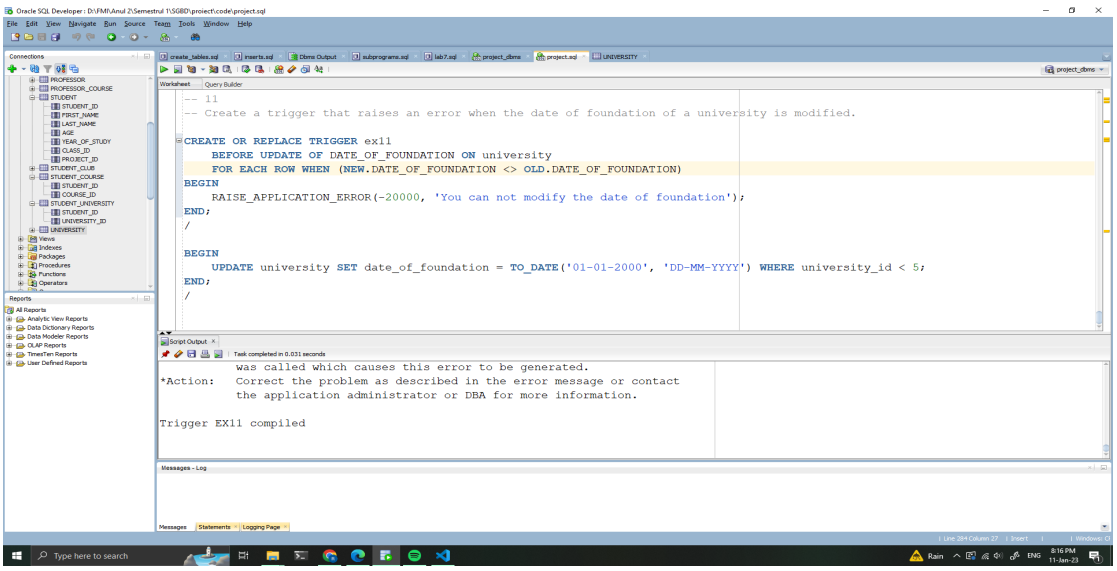
```
    END IF;
END;
/

BEGIN
    UPDATE student SET first_name = 'Robert' WHERE student_id = 1;
    -- change working hours to see the error
END;
/
```

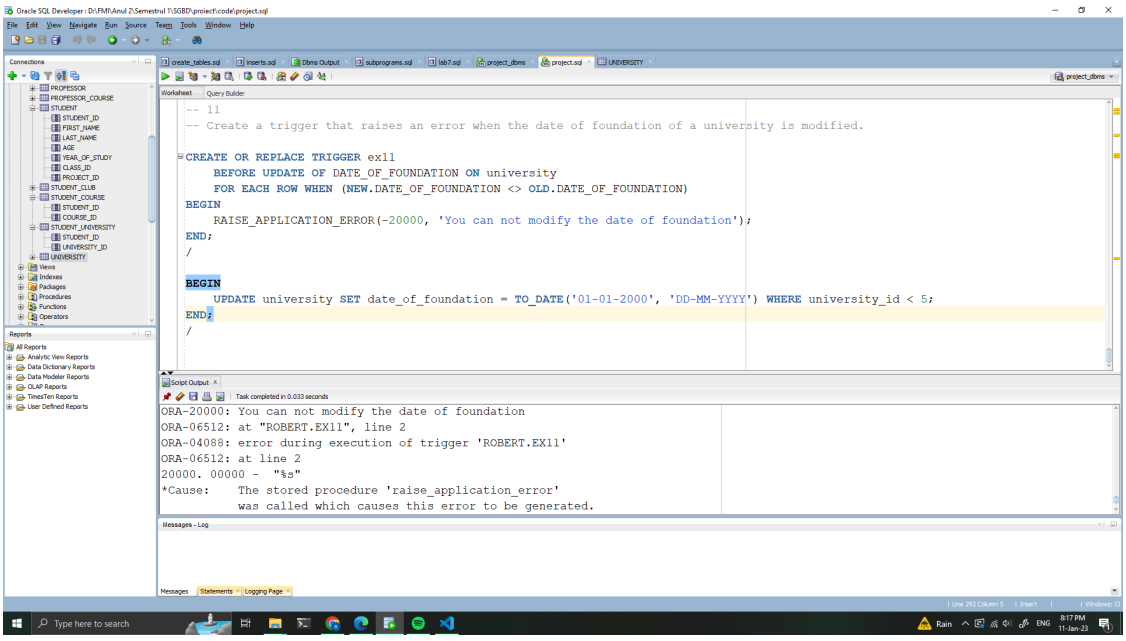


11. Definiți un trigger de tip LMD la nivel de linie. Declanșați trigger-ul.

```
-- trigger de tip LMD la nivel de linie
-- Create a trigger that raises an error when the date of foundation of a
university is modified.

CREATE OR REPLACE TRIGGER ex11
    BEFORE UPDATE OF DATE_OF_FOUNDATION ON university
    FOR EACH ROW WHEN (NEW.DATE_OF_FOUNDATION <> OLD.DATE_OF_FOUNDATION)
BEGIN
    RAISE_APPLICATION_ERROR(-20000, 'You can not modify the date of
```

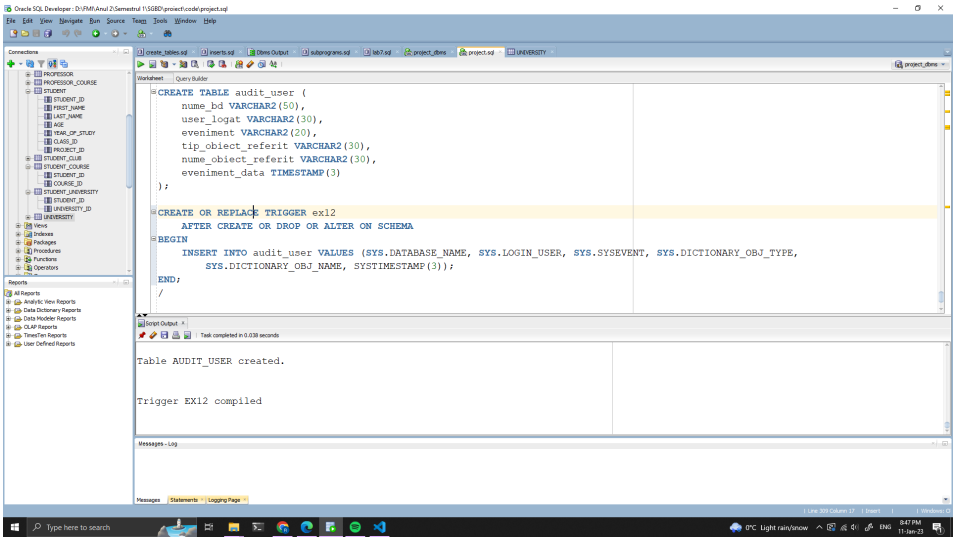
```
foundation');
END;
/

BEGIN
    UPDATE university SET date_of_foundation = TO_DATE('01-01-2000',
'DD-MM-YYYY') WHERE university_id < 5;
END;
/
```

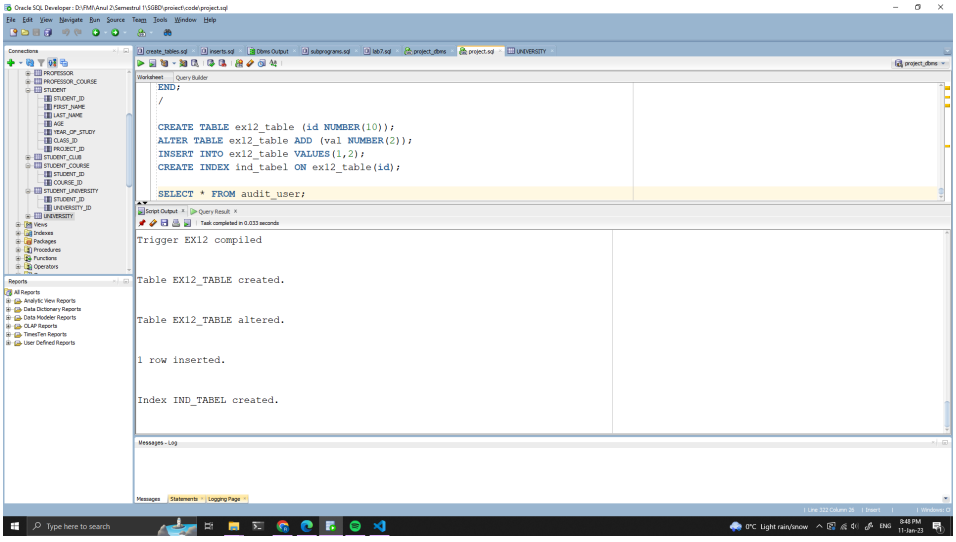


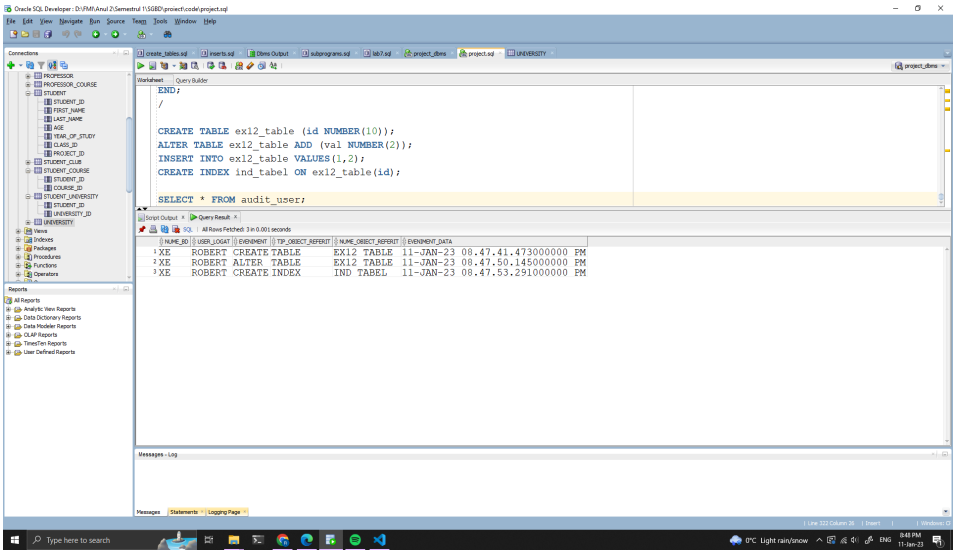


12. Definiți un trigger de tip LDD. Declanșați trigger-ul.

```
-- trigger de tip LDD
-- Create a trigger that records user actions on the database in a table.

CREATE TABLE audit_user (
    nume_bd VARCHAR2(50),
    user_logat VARCHAR2(30),
    eveniment VARCHAR2(20),
    tip_obiect_referit VARCHAR2(30),
    nume_obiect_referit VARCHAR2(30),
    eveniment_data TIMESTAMP(3)
);

CREATE OR REPLACE TRIGGER ex12
    AFTER CREATE OR DROP OR ALTER ON SCHEMA
BEGIN
    INSERT INTO audit_user VALUES (SYS.DATABASE_NAME, SYS.LOGIN_USER,
SYS.SYSEVENT, SYS.DICTIONARY_OBJ_TYPE,
        SYS.DICTIONARY_OBJ_NAME, SYSTIMESTAMP(3));
END;
/

CREATE TABLE ex12_table (id NUMBER(10));
ALTER TABLE ex12_table ADD (val NUMBER(2));
INSERT INTO ex12_table VALUES(1,2);
CREATE INDEX ind_tabel ON ex12_table(id);

SELECT * FROM audit_user;
```


CREATE TABLE audit_user (
nume_bd VARCHAR2(50),
user_logat VARCHAR2(30),
eveniment VARCHAR2(20),
tip_obiect_referit VARCHAR2(30),
nume_obiect_referit VARCHAR2(30),
eveniment_data TIMESTAMP(3)
);
CREATE OR REPLACE TRIGGER ex12
AFTER CREATE OR DROP OR ALTER ON SCHEMA
BEGIN
INSERT INTO audit_user VALUES (SYS.DATABASE_NAME, SYS.LOGIN_USER, SYS.SYSEVENT, SYS.DICTIONARY_OBJ_TYPE,
SYS.DICTIONARY_OBJ_NAME, SYSTIMESTAMP(3));
END;
/
Table AUDIT_USER created.
Trigger EX12 compiled


END;
/
CREATE TABLE ex12_table (id NUMBER(10));
ALTER TABLE ex12_table ADD (val NUMBER(2));
INSERT INTO ex12_table VALUES(1,2);
CREATE INDEX ind_tabel ON ex12_table(id);
SELECT * FROM audit_user;
Trigger EX12 compiled
Table EX12_TABLE created.
Table EX12_TABLE altered.
1 row inserted.
Index IND_TABEL created.


END;
/
CREATE TABLE ex12_table (id NUMBER(10));
ALTER TABLE ex12_table ADD (val NUMBER(2));
INSERT INTO ex12_table VALUES(1,2);
CREATE INDEX ind_tabel ON ex12_table(id);
SELECT * FROM audit_user;
NUME_BD USER_LOGAT EVENIMENT TIP_OBIECT_REFERIT NUME_OBIECT_REFERIT EVENIMENT_DATA
1 XE ROBERT CREATE TABLE EX12 TABLE 11-JAN-23 08.47.41.473000000 PM
2 XE ROBERT ALTER TABLE EX12 TABLE 11-JAN-23 08.47.50.145000000 PM
3 XE ROBERT CREATE INDEX IND TABEL 11-JAN-23 08.47.53.291000000 PM

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

```
-- Create a package that contains all the procedures and functions from the
previous exercises.
CREATE OR REPLACE PACKAGE ex13 AS
    PROCEDURE ex6 (class_num department.class_number%TYPE); -- ex6
    PROCEDURE ex7 (city_name_par city.city_name%TYPE, cursor_option
NUMBER); -- ex7
    FUNCTION ex8 (course_name_par course.course_name%TYPE) RETURN NUMBER;
-- ex8
    PROCEDURE ex9 (student_name_par student.first_name%TYPE, experience_par
NUMBER); -- ex9
END ex13;
/

CREATE OR REPLACE PACKAGE BODY ex13 AS
    -- ex6
    PROCEDURE ex6 (class_num department.class_number%TYPE) AS
        TYPE dept_students IS TABLE OF student%ROWTYPE INDEX BY
PLS_INTEGER;
        TYPE all_students IS VARRAY(100) OF student%ROWTYPE;
        all_student_var all_students;
        students dept_students;
        curr_class_id department.class_id%TYPE;
    BEGIN
        SELECT * BULK COLLECT INTO all_student_var FROM student;

        SELECT class_id INTO curr_class_id FROM department WHERE
class_number = class_num;

        SELECT * BULK COLLECT INTO students FROM student WHERE class_id =
curr_class_id;

        FOR i IN students.FIRST..students.LAST LOOP
            DBMS_OUTPUT.PUT_LINE(students(i).first_name);
        END LOOP;

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

        FOR i IN all_student_var.FIRST..all_student_var.LAST LOOP
            DBMS_OUTPUT.PUT_LINE(all_student_var(i).first_name || ' ' ||
```

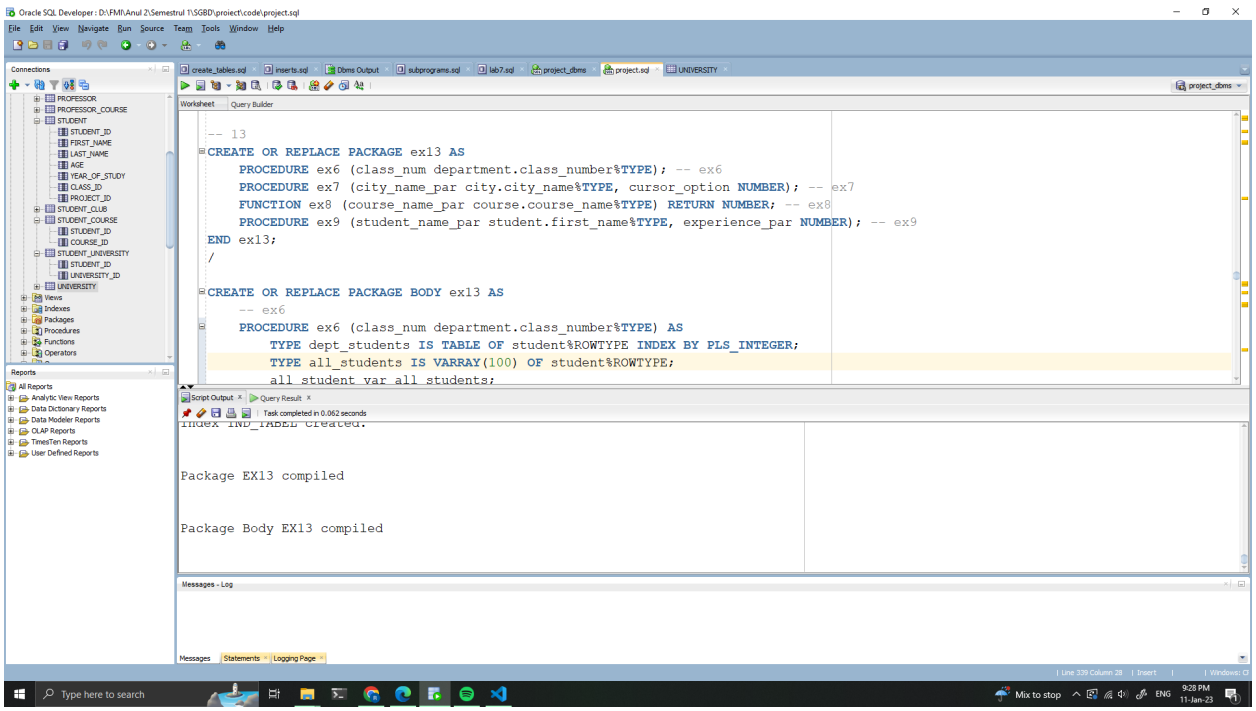
```
all_student_var(i).last_name);
        END LOOP;
    END;

    -- ex7
    PROCEDURE ex7 (city_name_par city.city_name%TYPE, cursor_option NUMBER)
AS
        CURSOR univ_from_city (city_id_par city.city_id%TYPE) IS
        SELECT * FROM university WHERE city_id = city_id_par;
        TYPE tip_cursor IS REF CURSOR RETURN professor%ROWTYPE;
        professor_info tip_cursor;
        city_id_var city.city_id%TYPE;
        val university%ROWTYPE;
        professor_val professor%ROWTYPE;
    BEGIN
        SELECT city_id INTO city_id_var FROM city WHERE city_name =
city_name_par;

        OPEN univ_from_city(city_id_var);
        LOOP
            FETCH univ_from_city INTO val;
            EXIT WHEN univ_from_city%NOTFOUND;
            DBMS_OUTPUT.PUT_LINE(val.university_name);
        END LOOP;
        CLOSE univ_from_city;

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

        IF cursor_option = 1 THEN
            -- all professors that have more than 10 years of experience
            OPEN professor_info FOR
                SELECT * FROM professor WHERE experience > 10;
        ELSIF cursor_option = 2 THEN
            OPEN professor_info FOR
                SELECT * FROM professor WHERE experience <= 10;
        END IF;

        LOOP
            FETCH professor_info INTO professor_val;
            EXIT WHEN professor_info%NOTFOUND;
            DBMS_OUTPUT.PUT_LINE(professor_val.first_name || ' ' ||
professor_val.last_name);
        END LOOP;
```

```
    END;

     -- ex8
    FUNCTION ex8 (course_name_par course.course_name%TYPE) RETURN NUMBER AS
        TYPE students_at_course IS TABLE OF student%ROWTYPE INDEX BY
PLS_INTEGER;
        students students_at_course;
        student_val student%ROWTYPE;
        course_val course%ROWTYPE;
        student_count NUMBER(10);
        course_id_val course.course_id%TYPE := NULL;
        INVALID_STRING EXCEPTION;
        NO_DATA_STUDENT EXCEPTION;
    BEGIN
        IF course_name_par IS NULL THEN
            RAISE INVALID_STRING;
        END IF;

        SELECT course_id INTO course_id_val FROM course WHERE course_name =
course_name_par;

        SELECT s.* BULK COLLECT INTO students FROM student s
        JOIN student_course sc ON sc.student_id = s.student_id
        JOIN course c ON c.course_id = sc.course_id
        WHERE c.course_id = course_id_val;

        IF students.COUNT = 0 THEN
            RAISE NO_DATA_STUDENT;
        END IF;

        student_count := students.COUNT;

        FOR i IN students.FIRST..students.LAST LOOP
            student_val := students(i);
            DBMS_OUTPUT.PUT_LINE(student_val.first_name || ' ' ||
student_val.last_name);
        END LOOP;

        RETURN student_count;

        EXCEPTION
            WHEN INVALID_STRING THEN
                DBMS_OUTPUT.PUT_LINE('Invalid parameter');
```

```
                RETURN -1;
            WHEN TOO_MANY_ROWS THEN
                DBMS_OUTPUT.PUT_LINE('Multiple courses with the same
name');
                RETURN -1;
            WHEN NO_DATA_STUDENT THEN
                DBMS_OUTPUT.PUT_LINE('No students for course with id ' ||
course_id_val);
                RETURN -1;
            WHEN NO_DATA_FOUND THEN
                DBMS_OUTPUT.PUT_LINE('No course with name ' ||
course_name_par || ' found');
                RETURN -1;
            WHEN OTHERS THEN
                DBMS_OUTPUT.PUT_LINE('Other exception');
                RETURN -1;
    END;
    -- ex9
    PROCEDURE ex9 (student_name_par student.first_name%TYPE, experience_par
NUMBER) AS
        TYPE student_type IS TABLE OF student%ROWTYPE INDEX BY PLS_INTEGER;
        TYPE course_type IS TABLE OF course%ROWTYPE INDEX BY PLS_INTEGER;
        TYPE professor_type IS TABLE OF professor%ROWTYPE INDEX BY
PLS_INTEGER;
        courses_studied_by_student course_type;
        students_with_given_name student_type;
        student_id_val student.student_id%TYPE;
        student_info student%ROWTYPE;
        professor_info professor_type;
        answer professor_type;
        professor_val professor%ROWTYPE;
        INVALID_PARAMETER EXCEPTION;
        NO_DATA_FOUND_STUDENTS EXCEPTION;
        TOO_MANY_STUDENTS EXCEPTION;
        NO_DATA_COURSES EXCEPTION;
        NO_DATA_PROFESSORS EXCEPTION;
        NEGATIVE_NUMBER EXCEPTION;
    BEGIN
        IF student_name_par IS NULL OR student_name_par = '' OR
experience_par IS NULL THEN
            RAISE INVALID_PARAMETER;
        END IF;
```

```
        IF experience_par < 0 THEN
            RAISE NEGATIVE_NUMBER;
        END IF;

        SELECT * BULK COLLECT INTO students_with_given_name FROM student
WHERE first_name = student_name_par;
        IF students_with_given_name.COUNT = 0 THEN
            RAISE NO_DATA_FOUND_STUDENTS;
        END IF;
        IF students_with_given_name.COUNT > 1 THEN
            RAISE TOO_MANY_STUDENTS;
        END IF;

        student_info := students_with_given_name(1);
        student_id_val := student_info.student_id;

        SELECT c.* BULK COLLECT INTO courses_studied_by_student FROM course
c
        JOIN student_course sc ON sc.course_id = c.course_id
        JOIN student s ON s.student_id = sc.student_id
        WHERE s.student_id = student_id_val;

        IF courses_studied_by_student.COUNT = 0 THEN
            RAISE NO_DATA_COURSES;
        END IF;

        SELECT p.* BULK COLLECT INTO professor_info FROM professor p
        JOIN professor_course pc ON pc.professor_id = p.professor_id
        JOIN course c ON c.course_id = pc.course_id
        JOIN student_course sc ON sc.course_id = c.course_id
        JOIN student s ON s.student_id = sc.student_id
        WHERE s.student_id = student_id_val;

        IF professor_info.COUNT = 0 THEN
            RAISE NO_DATA_PROFESSORS;
        END IF;

        FOR i IN professor_info.FIRST..professor_info.LAST LOOP
            professor_val := professor_info(i);
            IF professor_val.experience >= experience_par THEN
                answer(i) := professor_val;
            END IF;
        END LOOP;
```

```
        FOR i IN answer.FIRST..answer.LAST LOOP
            professor_val := answer(i);
            DBMS_OUTPUT.PUT_LINE(professor_val.first_name || ' ' ||
professor_val.last_name);
        END LOOP;


        EXCEPTION
            WHEN INVALID_PARAMETER THEN
                DBMS_OUTPUT.PUT_LINE('Invalid parameter');
            WHEN NEGATIVE_NUMBER THEN
                DBMS_OUTPUT.PUT_LINE('Negative experience number');
            WHEN TOO_MANY_STUDENTS THEN
                DBMS_OUTPUT.PUT_LINE('Multiple students with the same
name');
            WHEN NO_DATA_FOUND_STUDENTS THEN
                DBMS_OUTPUT.PUT_LINE('No students with name ' ||
student_name_par || ' found');
            WHEN NO_DATA_COURSES THEN
                DBMS_OUTPUT.PUT_LINE('No courses for student with id ' ||
student_id_val);
            WHEN NO_DATA_PROFESSORS THEN
                DBMS_OUTPUT.PUT_LINE('No professors for student with id '
|| student_id_val);
            WHEN NO_DATA_FOUND THEN
                null;
            WHEN OTHERS THEN
                DBMS_OUTPUT.PUT_LINE('code error ' || SQLCODE);
                DBMS_OUTPUT.PUT_LINE('message error ' || SQLERRM);
    END;
END ex13;
/
```



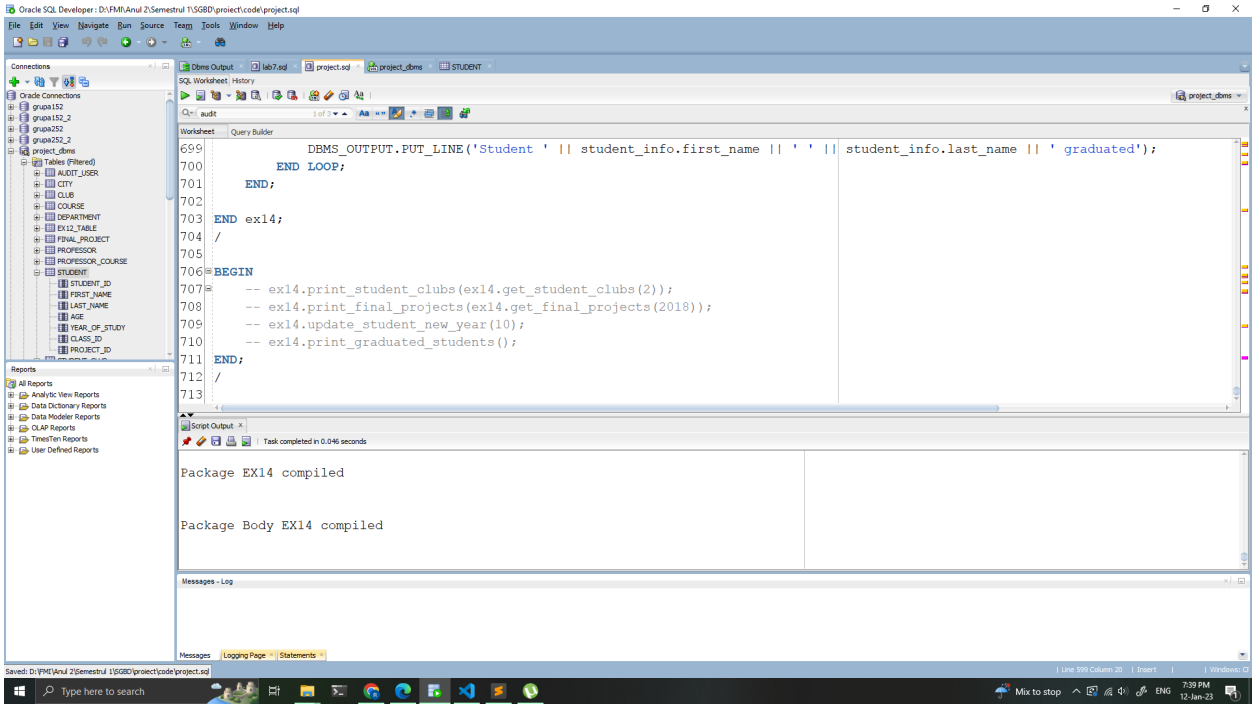
14. Definiți un pachet care să includă tipuri de date complexe și obiecte necesare unui flux de acțiuni integrate, specifice bazei de date definite (minim 2 tipuri de date, minim 2 funcții, minim 2 proceduri)

```
CREATE OR REPLACE PACKAGE ex14 AS
    TYPE club_type IS TABLE OF club%ROWTYPE INDEX BY PLS_INTEGER;
    TYPE final_project_type IS TABLE OF final_project%ROWTYPE;
    TYPE student_type IS VARRAY(100) OF student%ROWTYPE;
    FUNCTION get_student_clubs (student_id_par student.student_id%TYPE)
RETURN club_type;
    PROCEDURE print_student_clubs(student_clubs club_type);
    FUNCTION get_final_projects(year_par NUMBER) RETURN final_project_type;
    PROCEDURE print_final_projects(final_projects final_project_type);
    PROCEDURE update_student_new_year(student_id_par
student.student_id%TYPE);
    PROCEDURE print_graduated_students;
END ex14;
/

CREATE OR REPLACE PACKAGE BODY ex14 AS
    FUNCTION get_student_clubs(student_id_par student.student_id%TYPE)
```

```
RETURN club_type IS
        answer club_type;
    BEGIN
        SELECT c.* BULK COLLECT INTO answer FROM student s
        JOIN student_club sc ON sc.student_id = s.student_id
        JOIN club c ON c.club_id = sc.club_id
        WHERE s.student_id = student_id_par;

        RETURN answer;

        EXCEPTION
            WHEN NO_DATA_FOUND THEN
                RETURN club_type();
            WHEN OTHERS THEN
                DBMS_OUTPUT.PUT_LINE('code error ' || SQLCODE);
                DBMS_OUTPUT.PUT_LINE('message error ' || SQLERRM);
                RETURN club_type();
    END;

    PROCEDURE print_student_clubs(student_clubs club_type) IS
        club_info club%ROWTYPE;
    BEGIN
        FOR i IN student_clubs.FIRST..student_clubs.LAST LOOP
            club_info := student_clubs(i);
            DBMS_OUTPUT.PUT_LINE(i || '.');
            DBMS_OUTPUT.PUT_LINE('club name: ' || club_info.club_name);
            DBMS_OUTPUT.PUT_LINE('club type: ' || club_info.club_type);
            DBMS_OUTPUT.PUT_LINE('club address: ' || club_info.address);
        END LOOP;
    END;

    FUNCTION get_final_projects(year_par NUMBER) RETURN final_project_type
IS
        answer final_project_type;
        INVALID_PARAMETER EXCEPTION;
    BEGIN
        IF year_par IS NULL OR year_par < 2000 THEN
            RAISE INVALID_PARAMETER;
        END IF;

        SELECT fp.* BULK COLLECT INTO answer FROM final_project fp
        WHERE EXTRACT(YEAR FROM fp.deadline) = year_par;
```

```
        RETURN answer;
        EXCEPTION
            WHEN INVALID_PARAMETER THEN
                DBMS_OUTPUT.PUT_LINE('Invalid parameter');
                RETURN final_project_type();
            WHEN NO_DATA_FOUND THEN
                RETURN final_project_type();
            WHEN OTHERS THEN
                DBMS_OUTPUT.PUT_LINE('code error ' || SQLCODE);
                DBMS_OUTPUT.PUT_LINE('message error ' || SQLERRM);
                RETURN final_project_type();
    END;
    PROCEDURE print_final_projects(final_projects final_project_type) IS
        final_project_info final_project%ROWTYPE;
        student_for_project student_type;
        student_info student%ROWTYPE;
        NO_DATA_STUDENT EXCEPTION;
    BEGIN
        student_for_project := student_type();
        FOR i IN final_projects.FIRST..final_projects.LAST LOOP
            student_for_project.extend();
            SELECT * INTO student_info FROM student s WHERE s.project_id =
final_projects(i).project_id;
            IF SQL%NOTFOUND THEN
                RAISE NO_DATA_STUDENT;
            END IF;
            student_for_project(student_for_project.COUNT) := student_info;
        END LOOP;

        FOR i IN final_projects.FIRST..final_projects.LAST LOOP
            final_project_info := final_projects(i);
            DBMS_OUTPUT.PUT_LINE(i || '.');
            DBMS_OUTPUT.PUT_LINE('Student name: ' ||
student_for_project(i).first_name || ' ' ||
student_for_project(i).last_name);
            DBMS_OUTPUT.PUT_LINE('Project name: ' ||
final_project_info.project_name);
            DBMS_OUTPUT.PUT_LINE('Deadline: ' ||
final_project_info.deadline);
        END LOOP;

        EXCEPTION
            WHEN NO_DATA_STUDENT THEN
```

```
                DBMS_OUTPUT.PUT_LINE('No student for project');
                RETURN;
            WHEN NO_DATA_FOUND THEN
                RETURN;
            WHEN OTHERS THEN
                DBMS_OUTPUT.PUT_LINE('code error ' || SQLCODE);
                DBMS_OUTPUT.PUT_LINE('message error ' || SQLERRM);
                RETURN;
    END;
    PROCEDURE update_student_new_year(student_id_par
student.student_id%TYPE) IS
        student_info student%ROWTYPE;
    BEGIN
        UPDATE student SET year_of_study = year_of_study + 1 WHERE
student_id = student_id_par;
        SELECT * INTO student_info FROM student WHERE student_id =
student_id_par;
        IF student_info.year_of_study > 4 THEN
            -- DELETE FROM student WHERE student_id = student_id_par;
            DBMS_OUTPUT.PUT_LINE('Student ' || student_info.first_name || '
' || student_info.last_name || ' graduated');
        END IF;
    END;

    PROCEDURE print_graduated_students IS
        student_info student%ROWTYPE;
        CURSOR c IS SELECT * FROM student WHERE year_of_study > 4;
    BEGIN
        FOR student_info IN c LOOP
            DBMS_OUTPUT.PUT_LINE('Student ' || student_info.first_name || '
' || student_info.last_name || ' graduated');
        END LOOP;
    END;
END ex14;
/

BEGIN
    -- ex14.print_student_clubs(ex14.get_student_clubs(2));
    -- ex14.print_final_projects(ex14.get_final_projects(2018));
    -- ex14.update_student_new_year(10);
    -- ex14.print_graduated_students();
END;
/
```


Oracle SQL Developer : D:\FMI\Anul 2\Semestrul 1\SGBD\proiect\code\project.sql
File Edit View Navigate Run Source Team Tools Window Help
Connections
Oracle Connections
grupa152
grupa152_2
grupa252
grupa252_2
project_dbms
Tables (Filtered)
AUDIT_USER
CITY
CLUB
COURSE
DEPARTMENT
EX12_TABLE
FINAL_PROJECT
PROFESSOR
PROFESSOR_COURSE
STUDENT
STUDENT_ID
FIRST_NAME
LAST_NAME
AGE
YEAR_OF_STUDY
CLASS_ID
PROJECT_ID
Reports
All Reports
Analytic View Reports
Data Dictionary Reports
Data Modeler Reports
OLAP Reports
TimesTen Reports
User Defined Reports
Dbms Output
lab7.sql
project.sql
project_dbms
STUDENT
SQL Worksheet History
Worksheet
Query Builder
699 DBMS_OUTPUT.PUT_LINE('Student ' || student_info.first_name || ' ' || student_info.last_name || ' graduated');
700 END LOOP;
701 END;
702
703 END ex14;
704 /
705
706 BEGIN
707 -- ex14.print_student_clubs(ex14.get_student_clubs(2));
708 -- ex14.print_final_projects(ex14.get_final_projects(2018));
709 -- ex14.update_student_new_year(10);
710 -- ex14.print_graduated_students();
711 END;
712 /
713
Script Output
Task completed in 0.046 seconds
Package EX14 compiled
Package Body EX14 compiled
Messages - Log
Messages
Logging Page
Statements
Type here to search
Mix to stop
ENG
7:39 PM
12-Jan-23