Laboratorinis darbas nr. 4

Užduotis atlikta naudojant → <https://sqliteonline.com/> mariadb serverį

**Komandos sukurti lenteles:**

CREATE TABLE students (

id INT(10) NOT NULL AUTO\_INCREMENT,

name VARCHAR(50),

email VARCHAR(50),

CONSTRAINT students\_pk PRIMARY KEY (id)

);

CREATE TABLE courses (

id INT(10) NOT NULL AUTO\_INCREMENT,

name VARCHAR(50),

enrolled\_students INT,

CONSTRAINT courses\_pk PRIMARY KEY (id)

);

CREATE TABLE exam\_results (

id INT(10) NOT NULL AUTO\_INCREMENT,

student\_id INT(10) REFERENCES students(id),

course\_id INT(10) REFERENCES courses(id),

exam\_date DATE,

score INT,

CONSTRAINT exam\_results\_pk PRIMARY KEY(id)

);

CREATE TABLE enrollments (

id INT(10) NOT NULL AUTO\_INCREMENT,

student\_id INT(10) REFERENCES students(id),

course\_id INT(10) REFERENCES courses(id),

CONSTRAINT enrollments\_pk PRIMARY KEY(id)

);

**Komandos duomenų užpildymui:**

INSERT INTO students VALUES (1, 'John Smith', ['john.smith@example.com](mailto:'john.smith@example.com)'),

(2, 'Jane Doe', ['jane.doe@example.com](mailto:'jane.doe@example.com)'),

(3, 'Sam Bravo', ['sam.bravo@example.com](mailto:'sam.bravo@example.com)'),

(4, 'Mia Williams', ['Mia Williams@example.com](mailto:'sam.bravo@example.com)');

INSERT INTO courses VALUES (1, 'Introduction to Programming', 1),

(2, 'Database Systems', 3),

(3, 'Operating systems', 2);

INSERT INTO exam\_results VALUES (1, 1, 1, STR\_TO\_DATE('2023-01-01', '%Y-%m-%d'), 80),

(2, 1, 2, STR\_TO\_DATE('2023-01-15', '%Y-%m-%d'), 90),

(3, 2, 1, STR\_TO\_DATE('2023-02-01', '%Y-%m-%d'), 85),

(4, 2, 2, STR\_TO\_DATE('2023-02-15', '%Y-%m-%d'), 95),

(5, 4, 3, STR\_TO\_DATE('2023-03-16', '%Y-%m-%d'), 100),

(6, 3, 3, STR\_TO\_DATE('2022-02-15', '%Y-%m-%d'), 78),

(7, 3, 2, STR\_TO\_DATE('2022-02-15', '%Y-%m-%d'), 95);

INSERT INTO enrollments VALUES (1, 1, 1),

(2, 1, 2),

(3, 2, 1),

(4, 2, 2),

(5, 3, 3),

(6, 4, 3),

(7, 3, 2);

**Sukurti funkciją, kuri apskaičiuoja studento pažymių vidurkį:**  
  
CREATE FUNCTION calculate\_avg\_score(stud\_id INT) RETURNS DECIMAL(10,2)

BEGIN

DECLARE avg\_score DECIMAL(10,2);

SELECT AVG(exam\_results.score) INTO avg\_score

FROM exam\_results

WHERE student\_id = stud\_id;

IF avg\_score IS NULL THEN

SET avg\_score = 0;

END IF;

RETURN avg\_score;

END;

**Sukurti funkciją skirtą gauti kurso pavadinimą:**

CREATE FUNCTION get\_course\_name(course\_id INT)

RETURNS VARCHAR(50)

BEGIN

DECLARE course\_name VARCHAR(50);

SELECT name INTO course\_name

FROM courses

WHERE id = course\_id;

IF course\_name IS NULL THEN

SET course\_name = '';

END IF;

RETURN course\_name;

END;

**Sukurti funkciją, kuri patikrina ar egzistuoja studentas pagal id:**

CREATE FUNCTION student\_exists(stud\_id INT)

RETURNs DECIMAL(10,0)

BEGIN

DECLARE s\_exists DECIMAL(10,0);

DECLARE counter DECIMAL(10,0);

SELECT COUNT(id) INTO counter FROM students WHERE id = stud\_id;

IF counter = 0 THEN

SET s\_exists = 0;

else

SET s\_exists = 1;

END IF;

RETURN s\_exists;

END;

**Sukurti procedūrą skirtą pridėti studentą į modulį:**CREATE PROCEDURE enroll\_student(IN student\_id INT, IN course\_id INT)

BEGIN

DECLARE s\_exists DECIMAL(10,0);

SELECT student\_exists(student\_id) INTO s\_exists;

IF s\_exists = 0 THEN

SIGNAL SQLSTATE '45000' SET

MESSAGE\_TEXT = 'Student not found';

END IF;

INSERT INTO enrollments (student\_id, course\_id)

VALUES (student\_id, course\_id);

COMMIT;

END;

**Sukurti procedūra ištrinti studentą su kursioriumi:**CREATE PROCEDURE remove\_student(IN student\_id INT)

BEGIN

DECLARE found\_student DECIMAL(10,0);

DECLARE num\_students DECIMAL(10,0);

DECLARE course INT;

DECLARE id INT;

DECLARE name VARCHAR(255);

DECLARE email VARCHAR(255);

DECLARE done INT DEFAULT FALSE;

DECLARE cur\_student CURSOR FOR SELECT \* FROM students;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

SET found\_student = 0;

OPEN cur\_student;

FETCH cur\_student INTO id, name, email;

students\_loop:

LOOP

IF id = student\_id THEN

SET found\_student = id;

LEAVE students\_loop;

END IF;

FETCH cur\_student INTO id, name, email;

IF done THEN

LEAVE students\_loop;

END IF;

end loop

students\_loop;

CLOSE cur\_student;

IF found\_student = 0 THEN

SIGNAL SQLSTATE '45000' SET

MESSAGE\_TEXT = 'Student not found';

END IF;

DELETE FROM exam\_results WHERE exam\_results.student\_id = found\_student;

DELETE FROM enrollments WHERE enrollments.student\_id = found\_student;

DELETE FROM students WHERE students.id = found\_student;

COMMIT;

END;

**Studento pažymių vidurkių funkcijos panaudojimas:**

SELECT calculate\_avg\_score(1) AS avg\_score;

**Kurso gavimo funkcijos panaudojimas:**

SELECT get\_course\_name(1) AS course;

**Patikrinti ar studentas egzistuoja funkcijos panaudojimas:**

SELECT student\_exists(1) AS student;

**Iškviesti enroll\_student procedūrą:**

Iškviesti be klaidų:

CALL enroll\_student(1,1);

Iškviesti su klaida:  
 CALL enroll\_student(5,1);

**Iškviesti studento pašalinimo funkciją:**

Iškviesti be klaidų:

CALL remove\_student(2)

Iškviesti su klaida:

CALL remove\_student(10)