SQL 2 Laboratory

Author: Piotr Copek

Date: 09.04.2025

Tasks were checked by laboratories supervisor.

Group B

Task 1. For each student with name starting with B or M letter display the number of all subjects for which they have the grades. Note. We are interested in the number of subjects not the number of grades.

```sql
SELECT st.student_id, st.student_name, COUNT(DISTINCT g.subject_id)
FROM students st
JOIN grades g ON st.student_id = g.student_id
WHERE st.student_name LIKE 'B%' OR st.student_name LIKE 'M%'
GROUP BY st.student_id, st.student_name;
```

Task 2. Display the IDs, names and the average salaries of employees who have payment in project No 1. Note. We are interested in the average of all salaries not just the payment in the given project.

```sql
SELECT e.employee_id, e.emp_name, AVG(s1.amount)
FROM employees e
JOIN salaries s1 ON e.employee_id = s1.employee_id
WHERE e.employee_id IN (
    SELECT s2.employee_id FROM salaries s2 WHERE s2.project_id = 1
)
GROUP BY e.employee_id, e.emp_name;
```

Task 3. Display the IDs and the names of projects in which the most employees received salaries. Note. The same number of employees may have received salaries in the multiple projects.

```
SELECT s.project_id, p.project_name, COUNT(DISTINCT s.employee_id) AS num
FROM salaries s
JOIN projects p ON s.project_id = p.project_id
GROUP BY s.project_id, p.project_name
HAVING COUNT(DISTINCT s.employee_id) = (
    SELECT MAX(emp_count) FROM (
        SELECT COUNT(DISTINCT s2.employee_id) AS emp_count
        FROM salaries s2
        GROUP BY s2.project_id
    ) AS counts
);
```

Task 4. Display the IDs and the names of students from the 1st major, who have a single grade lower than any grade given in COMPUTER ARCHITECTURE subject.

```
SELECT DISTINCT s.student_id, s.student_name
FROM students s
JOIN grades g ON s.student_id = g.student_id
WHERE s.major_id = 1
AND EXISTS (
    SELECT 1
    FROM grades g3
    JOIN subjects sub ON g3.subject_id = sub.subject_id
    WHERE sub.subject_name = 'COMPUTER ARCHITECTURE'
    AND g3.grade > g.grade
);
```

Task used to be implemented with `MIN`, but after consultations it was corrected.

Task 5. Display the IDs and the names of the majors for which the youngest student is older than student GREGG.

```
SELECT m.major_id, m.major_name
FROM majors m
JOIN students s ON m.major_id = s.major_id
GROUP BY m.major_id, m.major_name
HAVING MAX(s.date_of_birth) < (
    SELECT date_of_birth FROM students WHERE student_name = 'GREGG'
);
```

We assume that there is only one GREGG in database which is not best practice.

Task 6. Display the IDs and the names of subjects where no female students have graded. The results should not be repeated.

```
SELECT DISTINCT sub.subject_id, sub.subject_name
FROM subjects sub
WHERE sub.subject_id NOT IN (
    SELECT g.subject_id
    FROM grades g
    JOIN students st ON g.student_id = st.student_id
    WHERE st.gender = 'F'
);
```

Task 7. Display the number of the employees who got salaries in the projects managed by employees with IDs less than 40 and greater than 36. We are interested in all projects!

```
SELECT p.project_id, p.project_name, COUNT(DISTINCT s.employee_id)
FROM projects p
LEFT JOIN salaries s ON p.project_id = s.project_id
WHERE p.manager_id > 36 AND p.manager_id < 40
GROUP BY p.project_id, p.project_name;
```

Task 8. Display the average number of students in majors. One number!

```sql
SELECT AVG(num) AS avg
FROM (
    SELECT COUNT(*) AS num FROM students GROUP BY major_id
) AS sub;
```

Task 9. Display the names of the oldest students in each major.

```sql
SELECT s.student_id, s.student_name, s.date_of_birth
FROM students s
WHERE s.date_of_birth = (
    SELECT MIN(s1.date_of_birth)
    FROM students s1
    WHERE s.major_id = s1.major_id
);
```

Task 10. Display the employees with the names starting with letter M who do not have any classes/lessons in any of the subjects taught by employee JOHNNY. Note. Each employee should be displayed only once.

```sql
SELECT DISTINCT e.employee_id, e.emp_name
FROM employees e
WHERE e.emp_name LIKE 'M%'
  AND e.employee_id NOT IN (
      SELECT DISTINCT s1.employee_id
      FROM schedules s1
      WHERE s1.subject_id IN (
          SELECT DISTINCT subject_id
          FROM schedules s2
          WHERE s2.employee_id = (
              SELECT employee_id FROM employees WHERE emp_name = 'JOHNNY'
          )
      )
  );
```

Task 11. Display the names of the employees who have salaries in all projects managed by the WOLF.

```sql
SELECT DISTINCT e.employee_id, e.emp_name
FROM employees e
JOIN salaries s ON e.employee_id = s.employee_id
WHERE s.project_id IN (
    SELECT project_id FROM projects WHERE manager_id = (
        SELECT employee_id FROM employees WHERE emp_name = 'WOLF'
    )
);
```