Solutions for exercises for the SQL tutorial

Fall 2019

1 Queries on a single table

1. Select all employees in the alphabetical order.

```
SELECT *
FROM employee
ORDER BY last_name, first_name
```

2. Select the year of birth, the salary and the note of Karoline Lekve.

```
SELECT year_of_birth, hour_salary, note
FROM employee
WHERE first_name = "Karoline" AND last_name = "Lekve"
```

3. Select the names of all departments.

```
SELECT title
FROM department
```

4. Select all employees whose last name starts with L and the first name is not Victor.

```
SELECT *
FROM employee
WHERE last_name LIKE "L%"
  AND first_name <> "Victor"
```

5. Select all distinct salaries, from the highest to the lowest.

```
SELECT DISTINCT hour_salary
FROM employee
ORDER BY hour_salary DESC
```

6. Select all employees born between 1970 and 1980.

```
SELECT *
FROM employee
WHERE year_of_birth BETWEEN 1970 AND 1980
```

or

```
SELECT *
FROM employee
WHERE year_of_birth >= 1970 AND year_of_birth <= 1980
```

7. Select the name and the salary of the supervisors (i.e., employees without a supervisor) of departments 1 and 2.

```
SELECT first_name, last_name, hour_salary
FROM employee
WHERE supervisor_id IS NULL
  AND department_id IN (1, 2)
```

or

```
SELECT first_name, last_name, hour_salary
FROM employee
WHERE supervisor_id IS NULL
  AND (department_id = 1 OR department_id = 2)
```

(Note: the parentheses are needed in this case.)

8. Select all employees ordered by their salaries. In the case of a tie, order by their last name.

```
SELECT *
FROM employee
ORDER BY hour_salary, last_name
```

2 Queries involving several tables

1. Select all employees, together with the name of their department.

```
SELECT employee.*, department.title
FROM employee, department
WHERE employee.department_id = department.id
```

or

```
SELECT employee.*, department.title
FROM employee
JOIN department ON employee.department_id = department.id
```

2. Select all employees working for the ”Planning” department.

```
SELECT employee.*
FROM employee, department
WHERE employee.department_id = department.id
  AND department.title = "Planning"
```

or

```
SELECT employee.*
FROM employee
JOIN department ON employee.department_id = department.id
WHERE department.title = "Planning"
```

3. Select distinct salaries in each department, including the department name.

```
SELECT DISTINCT department.title, employee.hour_salary
FROM employee, department
WHERE employee.department_id = department.id
```

or

```
SELECT DISTINCT department.title, employee.hour_salary
FROM employee
JOIN department ON employee.department_id = department.id
```

4. Select the first and last name of employees, the titles of projects they are working on, and the time they have spent working on these projects.

```
SELECT employee.last_name, employee.first_name, project.title,
   employee_project.hours_spent
FROM employee, project, employee_project
WHERE employee_project.employee_id = employee.id
  AND employee_project.project_id = project.id
```

or

```
SELECT employee.last_name, employee.first_name, project.title,
   employee_project.hours_spent
FROM employee_project
JOIN project ON project.id = employee_project.project_id
JOIN employee ON employee.id = employee_project.employee_id
```

5. Select the last name of all employees that have a supervisor, together with the name of their supervisor.

```
SELECT e1.last_name, e2.last_name AS supervisor_last_name
FROM employee AS e1, employee AS e2
WHERE e1.supervisor_id = e2.id
```

or

```
SELECT e1.last_name, e2.last_name AS supervisor_last_name
FROM employee AS e1
JOIN employee AS e2 ON e1.supervisor_id = e2.id
```

6. Produce all possible pairs of employees (include both first and last names). Every pair of people must be listed only once.

```
SELECT e1.last_name, e1.first_name,
   e2.last_name AS last_name_2, e2.first_name AS first_name_2
FROM employee AS e1, employee AS e2
WHERE e1.id<e2.id
```

or

```
SELECT e1.last_name, e1.first_name,
   e2.last_name AS last_name_2, e2.first_name AS first_name_2
FROM employee AS e1
JOIN employee AS e2
WHERE e1.id<e2.id
```

3 Queries involving set-based operations

1. Make a single-column list of the project and department names.

```
SELECT title FROM project
UNION
SELECT title FROM department
```

2. Make a single-column list of all projects where every project is listed twice.

```
SELECT title FROM project
UNION ALL
SELECT title FROM project
ORDER BY title
```

3. Select the id of employees working on both project 2 and project 3.

```
SELECT employee_id FROM employee_project WHERE project_id = 2
INTERSECT
SELECT employee_id FROM employee_project WHERE project_id = 3
```

4 Queries involving grouping and aggregates

1. Count the number of departments.

```
SELECT count(*)
FROM department
```

2. Count the number of employees in the ”Production A” department.

```
SELECT count(*)
FROM employee, department
WHERE employee.department_id = department.id
  AND department.title = "Production A"
```

3. How many hours in total has been spent on each project? Include the project name in the result.

```
SELECT project.title, sum(hours_spent) AS total_hours_spent
FROM employee_project, project
WHERE employee_project.project_id = project.id
GROUP BY project.id, project.title
```

4. Count the number of employees, the minimum and the maximum salary in each department, including its name.

```
SELECT department.title, count(*), min(hour_salary), max(hour_salary)
FROM employee, department
WHERE employee.department_id = department.id
GROUP BY department.id, department.title
```

5. What is the average salary at departments with at least 5 employees. Include the name of the departments.

```
SELECT department.title, avg(hour_salary)
FROM employee, department
WHERE employee.department_id = department.id
GROUP BY department.id, department.title
HAVING count(*) >= 5
```

6. Count how many employees each supervisor manages.

```
SELECT m.first_name, m.last_name, count(*)
FROM employee e, employee m
WHERE e.supervisor_id = m.id
GROUP BY m.id, m.first_name, m.last_name
```

7. (*) Which projects did not blow the budget (i.e., have estimated cost higher than the real cost)? Include the project name, the estimated and the real cost.

```
SELECT p.title, p.cost_est, sum(hours_spent*hour_salary) AS cost
FROM employee_project ep, employee e, project p
WHERE ep.employee_id = e.id
  AND ep.project_id = p.id
GROUP BY p.id
HAVING p.cost_est >= cost
```

5 Additional queries (nesting and outer joins)

1. Select employees with the highest salary.

   ```
   SELECT *
   FROM employee
   WHERE hour_salary = (SELECT max(hour_salary) FROM employee)
   ```

2. Select employees not working on any project.

   ```
   SELECT *
   FROM employee
   WHERE id NOT IN (SELECT employee_id FROM employee_project)
   ```

3. Select the name of all employees, together with the name of their supervisor.

   ```
   SELECT e1.last_name, e1.first_name,
      e2.last_name AS supervisor_last_name,
      e2.first_name AS supervisor_first_name
   FROM employee e1
   LEFT JOIN employee e2 ON e1.supervisor_id = e2.id
   ```

4. Select employees which have the highest salaries in their respective departments

   ```
   SELECT *
   FROM employee e1
   WHERE e1.hour_salary = (
      SELECT max(e2.hour_salary)
      FROM employee e2
      WHERE e1.department_id = e2.department_id
   )
   ```

5. (*) Select the department with the highest average salary.

   ```
   SELECT title, avg(hour_salary) AS avg_hour_salary
   FROM employee, department
   WHERE department_id = department.id
   GROUP BY department_id
   HAVING avg_hour_salary = (
      SELECT max(avg_hour_salary) FROM (
         SELECT avg(hour_salary) AS avg_hour_salary
         FROM employee
         GROUP BY department_id
      )
   )
   ```

   If this is too complicated for you, try to solve it using views. (Note however that the automatic assessment in the SQL Explorer will not be able to provide any feedback in this case.)