MySQL Exercises

Exercise 1:

Find the average salary of male and female employees in each department.

```
USE employees;

SELECT
    d.dept_name, e.gender, AVG(salary) as average_salary
FROM
    salaries s
        JOIN
    employees e ON s.emp_no = e.emp_no
        JOIN
    dept_emp de ON e.emp_no = de.emp_no
        JOIN
    departments d ON d.dept_no = de.dept_no
GROUP BY de.dept_no , e.gender
ORDER BY de.dept_no;
```

Query Result:

dept_name	gender	average_salary
Marketing	M	72198.19
Marketing	F	71464.48
Finance	M	70327.03
Finance	F	69914.92
Human Resources	M	55196.55
Human Resources	F	55596.37
Production	M	59596.36
Production	F	59456.00
Development	M	59576.33
Development	F	59391.95
Quality Managem...	M	57206.90
Quality Managem...	F	57423.31
Sales	M	80879.76
Sales	F	80626.56
Research	M	59965.77
Research	F	59712.78

Exercise 2:

Find the lowest department number encountered in the 'dept_emp' table. Then, find the highest department number.

2.1)

```
SELECT MIN(dept_no)
FROM dept_emp;
```

Query Result:

	MIN(dept_no)
▶	d001

2.2)

```
SELECT MAX(dept_no)
FROM dept_emp;
```

Query Result:

	MAX(dept_no)
▶	d009

Exercise 3:

Obtain a table containing the following three fields for all individuals whose employee number is not greater than 10040:

- employee number
- the lowest department number among the departments where the employee has worked in (Hint: use a subquery to retrieve this value from the 'dept_emp' table)
- assign '110022' as 'manager' to all individuals whose employee number is lower than or equal to 10020, and '110039' to those whose number is between 10021 and 10040 inclusive.

Use a CASE statement to create the third field.

If you've worked correctly, you should obtain an output containing 40 rows.
Here's the top part of the output. Does it remind you of an output you've obtained earlier in the course?

	emp_no	dept_no	manager
▶	10001	d005	110022
	10002	d007	110022
	10003	d004	110022
	10004	d004	110022
	10005	d003	110022
	10006	d005	110022

```
SELECT
    emp_no,
    (SELECT
            MIN(dept_no)
        FROM
            dept_emp de
        WHERE
            e.emp_no = de.emp_no) dept_no,
    CASE
        WHEN emp_no <= 10020 THEN '110022'
        ELSE '110039'
    END AS manager
FROM
    employees e
WHERE
    emp_no <= 10040;
```

Query Result:

emp_no	dept_no	manager
10001	d005	110022
10002	d007	110022
10003	d004	110022
10004	d004	110022
10005	d003	110022
10006	d005	110022
10007	d008	110022
10008	d005	110022
10009	d006	110022
10010	d004	110022
10011	d009	110022
10012	d005	110022
10013	d003	110022
10014	d005	110022
10015	d008	110022
10016	d007	110022
10017	d001	110022
10018	d004	110022
10019	d008	110022
10020	d004	110022
10021	d005	110039

Result 1

Exercise 4:

Retrieve a list with all employees that have been hired in the year 2000.

```
SELECT *
FROM employees
WHERE YEAR(hire_date) = 2000;
```

Query Result:

emp_no	birth_date	first_name	last_name	gender	hire_date
47291	1960-09-09	Ulf	Flexer	M	2000-01-12
60134	1964-04-21	Seshu	Rathonyi	F	2000-01-02
72329	1953-02-09	Randi	Luit	F	2000-01-02
108201	1955-04-14	Mariangiola	Boreale	M	2000-01-01
205048	1960-09-12	Ennio	Alblas	F	2000-01-06
222965	1959-08-07	Volkmar	Perko	F	2000-01-13
226633	1958-06-10	Xuejun	Benzmuller	F	2000-01-04
227544	1954-11-17	Shahab	Demeyer	M	2000-01-08
422990	1953-04-09	Jaana	Verspoor	F	2000-01-11
424445	1953-04-27	Jeong	Boreale	M	2000-01-03
428377	1957-05-09	Yucai	Gerlach	M	2000-01-23
463807	1964-06-12	Bikash	Covnot	M	2000-01-28
499553	1954-05-06	Hideyuki	Delgrande	F	2000-01-22
NULL	NULL	NULL	NULL	NULL	NULL

Exercise 5:

Retrieve a list with all employees from the 'titles' table who are engineers.

Repeat the exercise, this time retrieving a list with all employees from the 'titles' table who are senior engineers.

5.1)

```
SELECT *
FROM titles
WHERE title LIKE ('%engineer%');
```

Query Result:

emp_no	title	from_date	to_date
10001	Senior Engineer	1986-06-26	9999-01-01
10003	Senior Engineer	1995-12-03	9999-01-01
10004	Engineer	1986-12-01	1995-12-01
10004	Senior Engineer	1995-12-01	9999-01-01
10006	Senior Engineer	1990-08-05	9999-01-01
10008	Assistant Engineer	1998-03-11	2000-07-31
10009	Assistant Engineer	1985-02-18	1990-02-18
10009	Engineer	1990-02-18	1995-02-18
10009	Senior Engineer	1995-02-18	9999-01-01
10010	Engineer	1996-11-24	9999-01-01
10012	Engineer	1992-12-18	2000-12-18
10012	Senior Engineer	2000-12-18	9999-01-01
10014	Engineer	1993-12-29	9999-01-01
10018	Engineer	1987-04-03	1995-04-03
10018	Senior Engineer	1995-04-03	9999-01-01
10020	Engineer	1997-12-30	9999-01-01
10022	Engineer	1999-09-03	9999-01-01
10023	Engineer	1999-09-27	9999-01-01
10024	Assistant Engineer	1998-06-14	9999-01-01
10026	Engineer	1995-03-20	2001-03-19
10026	Senior Engineer	2001-03-19	9999-01-01

titles 3

5.2)

```
SELECT *
FROM titles
WHERE title LIKE ('%senior engineer%');
```

Query Result:

emp_no	title	from_date	to_date
10001	Senior Engineer	1986-06-26	9999-01-01
10003	Senior Engineer	1995-12-03	9999-01-01
10004	Senior Engineer	1995-12-01	9999-01-01
10006	Senior Engineer	1990-08-05	9999-01-01
10009	Senior Engineer	1995-02-18	9999-01-01
10012	Senior Engineer	2000-12-18	9999-01-01
10018	Senior Engineer	1995-04-03	9999-01-01
10026	Senior Engineer	2001-03-19	9999-01-01
10027	Senior Engineer	2001-04-01	9999-01-01
10029	Senior Engineer	2000-09-17	9999-01-01
10030	Senior Engineer	2001-02-17	9999-01-01
10031	Senior Engineer	1998-09-01	9999-01-01
10032	Senior Engineer	1995-06-20	9999-01-01
10035	Senior Engineer	1996-09-05	9999-01-01
10037	Senior Engineer	1995-12-05	9999-01-01
10040	Senior Engineer	1999-02-14	9999-01-01
10043	Senior Engineer	1997-10-20	9999-01-01
10047	Senior Engineer	1998-03-31	9999-01-01
10051	Senior Engineer	1998-10-15	9999-01-01
10056	Senior Engineer	1999-02-01	9999-01-01
10057	Senior Engineer	2000-01-15	9999-01-01

titles 4

Exercise 6:

- Create a procedure that asks you to insert an employee number to obtain an output containing the same number, as well as the number and name of the last department the employee has worked for.
- Finally, call the procedure for employee number 10010.

  If you've worked correctly, you should see that employee number 10010 has worked for department number 6 - "Quality Management".

```
DROP procedure IF EXISTS last_department;

DELIMITER $$
CREATE PROCEDURE last_department (in p_emp_no integer)
BEGIN
SELECT
    e.emp_no, d.dept_no, d.dept_name
FROM
    employees e
        JOIN
    dept_emp de ON e.emp_no = de.emp_no
        JOIN
    departments d ON de.dept_no = d.dept_no
WHERE
    e.emp_no = p_emp_no
        AND de.from_date = (SELECT
            MAX(from_date)
        FROM
            dept_emp
        WHERE
            emp_no = p_emp_no);
END$$
DELIMITER ;

call employees.last_department(10010);
```

Query Result:

	emp_no	dept_no	dept_name
▶	10010	d006	Quality Management

Exercise 7:

How many contracts have been registered in the 'salaries' table with duration of more than one year and of value higher than or equal to $100,000?

Hint: You may wish to compare the difference between the start and end date of the salaries contracts.

```
SELECT COUNT(*) as num_contact
FROM salaries
WHERE salary >= 100000 AND DATEDIFF(to_date, from_date) > 365;
```

Query Result:

	num_contact
▶	5970

Exercise 8:

- Define a function that retrieves the largest contract salary value of an employee. Apply it to employee number 11356.
- In addition, what is the lowest contract salary value of the same employee? You may want to create a new function that to obtain the result.

8.1)

```
DROP FUNCTION IF EXISTS f_highest_salary;

DELIMITER $$
CREATE FUNCTION f_highest_salary (p_emp_no INTEGER) RETURNS DECIMAL(10,2)
DETERMINISTIC
BEGIN

DECLARE v_highest_salary DECIMAL(10,2);

SELECT
    MAX(s.salary)
INTO v_highest_salary FROM
    employees e
        JOIN
    salaries s ON e.emp_no = s.emp_no
WHERE
    e.emp_no = p_emp_no;

RETURN v_highest_salary;
END$$

DELIMITER ;


SELECT f_highest_salary(11356);
```

Query Result:

	f_highest_salary(11356)
▶	83067.00

8.2)

```
DROP FUNCTION IF EXISTS f_lowest_salary;

DELIMITER $$
CREATE FUNCTION f_lowest_salary (p_emp_no INTEGER) RETURNS DECIMAL(10,2)
DETERMINISTIC
BEGIN

DECLARE v_lowest_salary DECIMAL(10,2);

SELECT
    MIN(s.salary)
INTO v_lowest_salary FROM
    employees e
        JOIN
    salaries s ON e.emp_no = s.emp_no
WHERE
    e.emp_no = p_emp_no;
```

```
RETURN v_lowest_salary;
END$$

DELIMITER ;


SELECT f_lowest_salary(10356);
```

Query Result:

	f_lowest_salary(10356)
▶	44901.00

Exercise 9:

- Based on the previous example, you can now try to create a function that accepts also a second parameter which would be a character sequence. Evaluate if its value is 'min' or 'max' and based on that retrieve either the lowest or the highest salary (using the same logic and code from Exercise 8). If this value is a string value *different* from 'min' or 'max', then the output of the function should return the difference between the highest and the lowest salary.

```
DROP FUNCTION IF EXISTS f_salary;

DELIMITER $$
CREATE FUNCTION f_salary (p_emp_no INTEGER, p_min_or_max varchar(10)) RETURNS
DECIMAL(10,2)
DETERMINISTIC
BEGIN

DECLARE v_salary_info DECIMAL(10,2);

SELECT
    CASE
        WHEN p_min_or_max = 'max' THEN MAX(s.salary)
        WHEN p_min_or_max = 'min' THEN MIN(s.salary)
        ELSE MAX(s.salary) - MIN(s.salary)
    END AS salary_info
INTO v_salary_info FROM
    employees e
        JOIN
    salaries s ON e.emp_no = s.emp_no
WHERE
    e.emp_no = p_emp_no;

RETURN v_salary_info;
END$$

DELIMITER ;

select employees.f_salary(11356, 'min');
```

	employees.f_salary(11356, 'min')
▶	53112.00

```
select employees.f_salary(11356, 'max');
```

	employees.f_salary(11356, 'max')
▶	83067.00

```
select employees.f_salary(11356, 'maxxx');
```

	employees.f_salary(11356, 'maxxx')
▶	29955.00