DBMS - SQL Queries

LIKE / NOT LIKE

3. Show those employee details whose first name ends with the letter 'a'.

```
SELECT *
FROM employees
WHERE first_name NOT LIKE "%a";
```

4. Show those employee details whose first name contains 'da' as substring.

```
SELECT *
FROM employees
WHERE first_name LIKE "%da%";
```

7. Show those employee details whose first name contains the letter 'o' and 'a'.

```
SELECT *
FROM employees
WHERE first_name LIKE "%o%a%";
```

BETWEEN … AND… / NOT BETWEEN … AND … / AND, &&

1. Show those department details whose location_id is within the range 1000 to 2000 inclusive.

```
SELECT *
FROM employees
WHERE salary BETWEEN 10000 AND 20000;
```

3. Show those employee details whose hire_date is within the range ‘1987-01-01’ to ‘1987-06-30’ inclusive.

```
SELECT *
FROM employees
WHERE hire_date BETWEEN '1987-01-01' AND '1987-06-30';
```

Numerical and String Functions

1. Show all the employees employee id and their short name in lowercase format.
   Short name format: first 3 letters from the first name followed by an underscore and then followed by the first 3 letters of the last name.

```
SELECT employee_id, LOWER(CONCAT(LEFT(first_name, 3), '_', LEFT(last_name,
3))) AS 'short name'
FROM employees;
```

3. Show all the employees employee id and email (i.e. add '@gmail.com' at the end of each email).

```
SELECT employee_id, CONCAT(email, '@gmail.com')
FROM employees;
```

6. Show those location details from locations table whose postal code consists at most 5 characters and the first two digits of its postal code is between 50 to 99 inclusive.

```
SELECT *
FROM locations
WHERE LENGTH(postal_code) <= 5 AND LEFT(postal_code, 2) BETWEEN 50 AND 99;
```

8. For each job, show the job id, job title and how much greater the max_salary from its min_salary in percentage format.
   Note: Show the output in 2 decimal points.
   $\%greater = (max\_salary - min\_salary) * 100/min\_salary$

```
SELECT job_id, job_title, ROUND((max_salary-min_salary) *
(100/min_salary), 2) AS '%greater'
FROM jobs;
```

Date and Time Functions

2. Show all the employees' email, hire date in "Jan 1987, 04" format.

```
SELECT email, DATE_FORMAT(hire_date, "%b %Y, %d") AS "hire date"
FROM employees;
```

6. Show those employees first name, email, phone number who is hired before the date "1st June 1987 11:01 PM".

```
SELECT first_name, email, phone_number
FROM employees
WHERE hire_date < '1987-06-30 23:01:00';
```

8. Show all the employees employee id, email and his current job experience in "10 years, 06 months and 15 days" format.
   Note: show the months in two digits format if necessary left pad with 0.

```
SELECT employee_id, email, CONCAT(
                                DATEDIFF(CURRENT_DATE, hire_date) DIV 365,
' years, ',
                                LPAD(DATEDIFF(CURRENT_DATE, hire_date)%365
DIV 30, 2, '0'), ' months and ',
                                DATEDIFF(CURRENT_DATE, hire_date)%30, '
days'
                                ) AS 'job experience'
FROM employees;
```

10. For each job_history, show how many days an employee has served during his last month of retirement.

```
SELECT employee_id, end_date, DAY(end_date) AS 'last month days'
FROM `job_history`;
```

12. Show all those employee details who have been hired on the leap day(29th Feb) of any leap year.

```
SELECT *
FROM `employees`
WHERE DAY(hire_date) = 29 AND MONTH(hire_date) = 2;
```

15. Show all the employees first name, department id, manager id and his updated join date that is one week before the real join date.

```
SELECT first_name, department_id, manager_id, hire_date,
DATE_SUB(hire_date, INTERVAL 7 DAY) AS 'updated join date'
FROM `employees`;
```

ORDER BY clause

2. Show all the employees employee id and their join date in such a way that the senior most employee comes first.
   If multiple employees have the same join date then also sort them based on the descending order of their department id.

```
SELECT employee_id, department_id, hire_date
FROM `employees`
ORDER BY hire_date, department_id DESC;
```

4. Show all the employees employee id, email, hire year (only the year portion) and hire month (show the full month name). Show the output from most recent hired employee to old employees.

```
SELECT employee_id, YEAR(hire_date) AS 'hire year', DATE_FORMAT(hire_date,
"%M") AS 'hire month'
FROM `employees`
ORDER BY hire_date DESC;
```

LIMIT clause

4. Show the top 3 lowest salary holder employee details from department number 60.

```
SELECT *
FROM `employees`
WHERE department_id = 60
ORDER BY salary
LIMIT 3;
```

Aggregate Operations (GROUP BY, HAVING clauses)

3. Show the maximum experienced employee hire date working in department number 50 from employees table.

Solution-1:

```
SELECT department_id, hire_date 'maximum experienced emp hire date'
FROM employees
WHERE department_id = 50
ORDER BY hire_date
LIMIT 0, 1;
```

Solution-2:

```
SELECT department_id, MIN(hire_date) AS 'maximum experienced emp hire
date'
FROM employees
GROUP BY department_id
HAVING department_id = 50;
```

7. Count the number of employees managed by manager id 114 from employees table.

```
SELECT manager_id, COUNT(*) 'total emp under manager 114'
FROM employees
WHERE manager_id = 114;
```

11. Show the maximum and minimum salary range value (i.e. salary range = max_salary - min_salary) from jobs table.

```
SELECT MAX(max_salary)-MIN(min_salary) AS 'Salary range'
FROM `jobs`;
```

---

1. Show each region_id and corresponding no of countries in that region from countries table.

```
SELECT region_id, COUNT(*)
FROM countries
GROUP BY region_id;
```

6. For each department_id, show the no of managers from that department using employees table.

```
SELECT department_id, COUNT(manager_id) AS 'Number of managers'
FROM `employees`
GROUP BY department_id;
```

7. Count the total number of employees joined in the even month and total number of employees joined in the odd number months from the employees table.

```
SELECT MONTH(hire_date)%2, COUNT(*)
FROM `employees`
GROUP BY MONTH(hire_date)%2;
```

14. For each job_id and each department, show the total no of employees in that group from the employees table.

```
SELECT job_id, department_id, COUNT(*)
FROM `employees`
GROUP BY job_id, department_id;
```

---

1. Show only those manager_ids who handle more than 5 employees.

```
SELECT manager_id, COUNT(*)
FROM employees
GROUP BY manager_id
HAVING COUNT(*) > 5;
```

3. Count the total no of employees for each department. Don’t consider employees of job_id "AD_PRESS" and also consider only those departments where total no of employees is greater than 5.

```
SELECT department_id, COUNT(*)
FROM `employees`
```

```
WHERE job_id <> 'AD_PRESS'
GROUP BY department_id
HAVING COUNT(*) > 5;
```

Table Join Operations (JOIN, LEFT JOIN clauses)

1. Show the region_name and corresponding country_name.

```
SELECT  countries.country_name, regions.region_name
FROM    countries
        JOIN regions
        ON countries.region_id = regions.region_id;
```

3. Show the employee_name and his job place country_name.

```
SELECT  CONCAT(e.first_name, ' ', e.last_name) AS 'Emp Name',
        c.country_name AS 'Country Name'
FROM    employees AS e
        JOIN departments AS d
        ON e.department_id = d.department_id

        JOIN locations AS l
        ON d.location_id = l.location_id

        JOIN countries AS c
        ON l.country_id = c.country_id;
```

5. Show the employee_name and his manager_name.

```
SELECT  CONCAT(emp.first_name, ' ', emp.last_name) AS 'Emp. Name',
        CONCAT(man.first_name, ' ', man.last_name) AS 'Man. Name'
FROM    employees AS emp
        JOIN employees AS man
        ON emp.manager_id = man.employee_id;
```

7. Show the employee_id, his salary, his manager_id, his manager_name, his manager_salary.

```
SELECT  emp.employee_id,
        emp.salary,

        man.employee_id AS "Manger's ID",
        CONCAT(man.first_name, ' ', man.last_name) AS 'Manager Name',
        man.salary
FROM    employees AS emp
```

```
        JOIN employees AS man
        ON emp.manager_id = man.employee_id;
```

10. Show the employee name and his manager name only for those employees who have joined after his manager.

```
SELECT  CONCAT(emp.first_name, ' ', emp.last_name) AS 'Employee Name',
        emp.hire_date,
        CONCAT(man.first_name, ' ', man.last_name) AS 'Manager Name',
        man.hire_date
FROM    employees AS emp
        JOIN employees AS man
        ON emp.hire_date > man.hire_date
```

11. Show the employees name and other employees name who receives higher salary than him.

```
SELECT  CONCAT(emp.first_name, ' ', emp.last_name) AS 'Employee Name',
        emp.salary,
        CONCAT(other.first_name, ' ', other.last_name) AS "Other
employee's Name",
        other.salary
FROM    employees AS emp
        JOIN employees AS other
        ON emp.salary < other.salary;
```