**MySQL Subquery**

* A subquery in MySQL is a query, which is nested into another SQL query and embedded with SELECT, INSERT, UPDATE or DELETE statement along with the various operators.
* A subquery is known as the **inner query**, and the query that contains subquery is known as the **outer query**.
* The inner query executed first gives the result to the outer query, and then the main/outer query will be performed which means the inner query executes first before its parent query so that the results of the inner query can be passed to the outer query.
* You can use a subquery in many places such as:
  + With the [IN](https://www.sqltutorial.org/sql-in/) or NOT IN operator
  + With [comparison operators](https://www.sqltutorial.org/sql-comparison-operators/)
  + With the [EXISTS](https://www.sqltutorial.org/sql-exists/) or [NOT EXISTS](https://www.sqltutorial.org/sql-exists/) operator
  + With the [ANY](https://www.sqltutorial.org/sql-any/) or [ALL](https://www.sqltutorial.org/sql-all/) operator
  + In the FROM clause
  + In the [SELECT](https://www.sqltutorial.org/sql-select/) clause
* MySQL Subquery Syntax

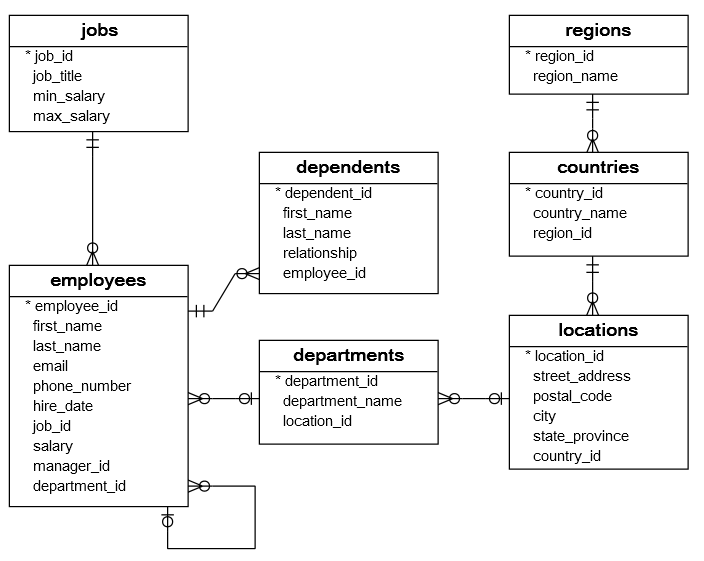
SELECT column\_list (s) FROM  table\_name

WHERE  column\_name OPERATOR

(SELECT column\_list (s)  FROM table\_name [WHERE])

* MySQL Subquery Example

**The following database diagram illustrates the HR sample database:**



**The HR sample database has seven tables:**

1. The employees table stores the data of employees.
2. The jobs table stores the job data including job title and salary range.
3. The departments table stores department data.
4. The dependents table stores the employee’s dependents.
5. The locations table stores the location of the departments of the company.
6. The countries table stores the data of countries where the company is doing business.
7. The regions table stores the data of regions such as Asia, Europe, America, and the Middle East and Africa. The countries are grouped into regions.

**Consider the following employees and departments tables from the**[**sample database**](https://www.sqltutorial.org/sql-sample-database/)**:**

### **SQL subquery with the IN or NOT IN operator**

* 1. **find all employees who locate in the location with the id 1700.**

SELECT

employee\_id, first\_name, last\_name

FROM

employees

WHERE

department\_id IN (SELECT

department\_id

FROM

departments

WHERE

location\_id = 1700)

ORDER BY first\_name , last\_name;

* 1. **find all employees who do not locate at the location 1700:**

SELECT

employee\_id, first\_name, last\_name

FROM

employees

WHERE

department\_id NOT IN (SELECT

department\_id

FROM

departments

WHERE

location\_id = 1700)

ORDER BY first\_name , last\_name;

### **SQL subquery with the comparison operator**

* + [Comparison operator](https://www.sqltutorial.org/sql-comparison-operators/) is one of these operators:
* Equal (=)
* Greater than (>)
* Less than (<)
* Greater than or equal ( >=)
* Less than or equal (<=)
* Not equal ( !=) or (<>)
  1. finds the employees who have the highest salary:

SELECT

employee\_id, first\_name, last\_name, salary

FROM

employees

WHERE

salary = (SELECT

MAX(salary)

FROM

employees)

ORDER BY first\_name , last\_name;

* 1. finds all employees who salaries are greater than the average salary of all employees:

SELECT

employee\_id, first\_name, last\_name, salary

FROM

employees

WHERE

salary > (SELECT

AVG(salary)

FROM

employees);

### **SQL subquery with the EXISTS or NOT EXISTS operator**

* + The [EXISTS](https://www.sqltutorial.org/sql-exists/) operator checks for the existence of rows returned from the subquery. It returns true if the subquery contains any rows. Otherwise, it returns false.
  + The NOT EXISTS operator is opposite to the EXISTS operator.
  1. finds all departments which have at least one employee with the salary is greater than 10,000:

SELECT department\_name

FROM departments WHERE EXISTS( SELECT 1

FROM

employees e

WHERE

salary > 10000

AND e.department\_id = d.department\_id)

ORDER BY department\_name;

* 1. finds all departments that do not have any employee with the salary greater than 10,000:

SELECT

department\_name

FROM

departments d

WHERE

NOT EXISTS( SELECT

1

FROM

employees e

WHERE

salary > 10000

AND e.department\_id = d.department\_id)

ORDER BY department\_name;

* 1. finds all employees whose salaries are greater than the lowest salary of every department:

SELECT

employee\_id, first\_name, last\_name, salary

FROM

employees

WHERE

salary >= ALL (SELECT

MIN(salary)

FROM

employees

GROUP BY department\_id)

ORDER BY first\_name , last\_name;

* 1. Return the average salary of every department:

SELECT

AVG(salary) average\_salary

FROM

employees

GROUP BY department\_id;

### **SQL Subquery in the SELECT clause**

A subquery can be used anywhere an expression can be used in the SELECT clause.

* 1. The following example finds the salaries of all employees, their average salary, and the difference between the salary of each employee and the average salary.

SELECT

employee\_id,

first\_name,

last\_name,

salary,

(SELECT

ROUND(AVG(salary), 0)

FROM

employees) average\_salary,

salary - (SELECT

ROUND(AVG(salary), 0)

FROM

employees) difference

FROM

employees

ORDER BY first\_name , last\_name;

## SQL UNION operator

1) The UNION operator combines result sets of two or more [SELECT](https://www.sqltutorial.org/sql-select/) statements into a single result set. The following statement illustrates how to use the UNION operator to combine result sets of two queries:

SELECT

column1, column2

FROM

table1

UNION [ALL]

SELECT

column3, column4

FROM

table2;

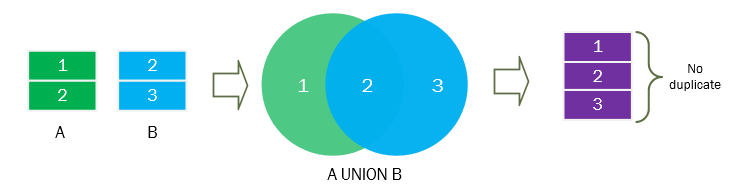
2) To use the UNION operator, you write the individual [SELECT statements](https://www.sqltutorial.org/sql-select/) and join them by the keyword UNION.

3) The columns returned by the SELECT statements must have the same or convertible data type, size, and be the same order.

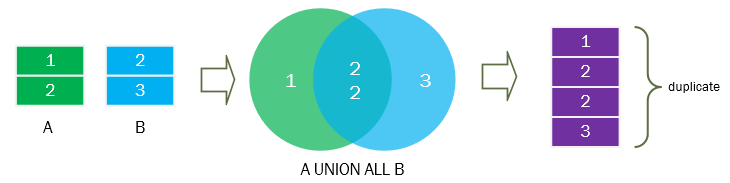
4) The database system processes the query by executing two SELECT statements first. Then, it combines two individual result sets into one and eliminates duplicate rows. To eliminate the duplicate rows, the database system sorts the combined result set by every column and scans it for the matching rows located next to one another.

5) To retain the duplicate rows in the result set, you use the UNION ALL operator.

6) Suppose, we have two result sets A(1,2) and B(2,3). The following picture illustrates A UNION B:



And the following picture illustrates A UNION ALL B



7) The union is different from the [join](https://www.sqltutorial.org/sql-inner-join/) that the join combines *columns* of multiple tables while the union combines *rows* of the tables.

## The SQL UNION examples

To get the data from the A table, you use the following SELECT statement:

Select id from a;



To retrieve the data from the B table, you use the following statement:

Select id from b;



To combine result sets of these two queries, you use the UNION operator as follows:

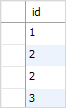
Select id from a UNION Select id from b;



The result set includes only 3 rows because the UNION operator removes one duplicate row.

### **SQL UNION ALL example**

To retain the duplicate row, you use the UNION ALL operator as follows:



The database system performs the following steps:

* + First, execute each SELECT statement individually.
  + Second, combine result sets and remove duplicate rows to create the combined result set.
  + Third, sort the combined result set by the column specified in the ORDER BY clause.

**Ques- To combine the first name and last name of employees and dependents.**



SELECT first\_name, last\_name FROM employees UNION select first\_name, last\_name from dependents order by last\_name;

