HOME   VIEWS   TRIGGERS   FUNCTIONS ⌄   SQL CHEAT SHEET   SQL IN ACTION
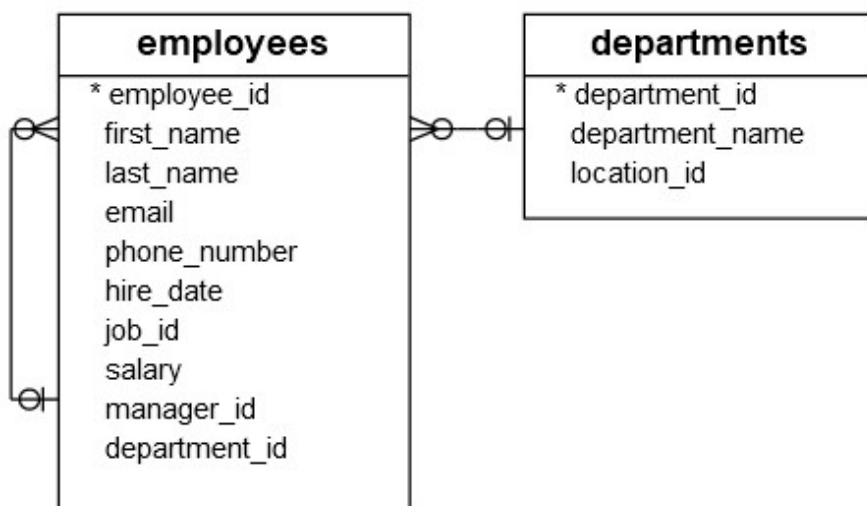
SQL Subquery

Summary: in this tutorial, you will learn about the SQL subquery and how to use the subqueries to form flexible SQL statements.

SQL subquery basic

Consider the following `employees` and `departments` tables from the sample database:

```
1   SELECT
2       *
3   FROM
4       departments
5   WHERE
6       location_id = 1700;
```

department_id	department_name	location_id
1	Administration	1700
3	Purchasing	1700
9	Executive	1700
10	Finance	1700
11	Accounting	1700

Second, find all employees that belong to the location 1700 by using the department id list of the previous query:

```
1   SELECT
2       employee_id, first_name, last_name
3   FROM
4       employees
5   WHERE
6       department_id IN (1 , 3, 8, 10, 11)
7   ORDER BY first_name , last_name;
```

employee_id	first_name	last_name
115	Alexander	Khoo
179	Charles	Johnson
109	Daniel	Faviet
114	Den	Raphaely
118	Guy	Himuro
111	Ismael	Sciarra
177	Jack	Livingston
200	Jennifer	Whalen
110	John	Chen
145	John	Russell

This solution has two problems. To start with, you have looked at the `departments` table to check

Another problem was that you have to revise the queries whenever you want to find employees who locate in a different location.

A much better solution to this problem is to use a subquery. By definition, a subquery is a query nested inside another query such as `SELECT`, `INSERT`, `UPDATE`, or `DELETE` statement. In this tutorial, we are focusing on the subquery used with the `SELECT` statement.

In this example, you can rewrite combine the two queries above as follows:

```
 1  SELECT
 2      employee_id, first_name, last_name
 3  FROM
 4      employees
 5  WHERE
 6      department_id IN (SELECT
 7              department_id
 8          FROM
 9              departments
10          WHERE
11              location_id = 1700)
12  ORDER BY first_name , last_name;
```

The query placed within the parentheses is called a subquery. It is also known as an inner query or inner select. The query that contains the subquery is called an outer query or an outer select.

To execute the query, first, the database system has to execute the subquery and substitute the subquery between the parentheses with its result – a number of department id located at the location 1700 – and then executes the outer query.

You can use a subquery in many places such as:

With the `IN` or `NOT IN` operator

With comparison operators

With the `EXISTS` or `NOT EXISTS` operator

HOME   VIEWS   TRIGGERS   FUNCTIONS ⌄   SQL CHEAT SHEET   SQL IN ACTION

SQL subquery examples

Let's take some examples of using the subqueries to understand how they work.

SQL subquery with the IN or NOT IN operator

In the previous example, you have seen how the subquery was used with the `IN` operator. The following example uses a subquery with the `NOT IN` operator to find all employees who do not locate at the location 1700:

```
 1  SELECT
 2      employee_id, first_name, last_name
 3  FROM
 4      employees
 5  WHERE
 6      department_id NOT IN (SELECT
 7              department_id
 8          FROM
 9              departments
10          WHERE
11              location_id = 1700)
12  ORDER BY first_name , last_name;
```

employee_id	first_name	last_name
121	Adam	Fripp
103	Alexander	Hunold
193	Britney	Everett
104	Bruce	Ernst
179	Charles	Johnson
105	David	Austin
107	Diana	Lorentz
204	Hermann	Baer
126	Irene	Mikkilineni
177	Jack	Livingston
145	John	Russell
176	Jonathon	Taylor

SQL subquery with the comparison operator

The following syntax illustrates how a subquery is used with a comparison operator:

Greater than (>)

Less than (<)

Greater than or equal (>=)

Less than or equal (<=)

Not equal (!=) or (<>)

The following example finds the employees who have the highest salary:

```
1   SELECT
2       employee_id, first_name, last_name, salary
3   FROM
4       employees
5   WHERE
6       salary = (SELECT
7               MAX(salary)
8           FROM
9               employees)
10  ORDER BY first_name , last_name;
```

employee_id	first_name	last_name	salary
100	Steven	King	24000.00

In this example, the subquery returns the highest salary of all employees and the outer query finds the employees whose salary is equal to the highest one.

The following statement finds all employees who salaries are greater than the average salary of all employees:

```
1   SELECT
2       employee_id, first_name, last_name, salary
3   FROM
4       employees
5   WHERE
6       salary > (SELECT
7               AVG(salary)
```

109	Daniel	Faviet	9000.00
114	Den	Raphaely	11000.00
204	Hermann	Baer	10000.00
177	Jack	Livingston	8400.00
110	John	Chen	8200.00
145	John	Russell	14000.00
176	Jonathon	Taylor	8600.00
146	Karen	Partners	13500.00
102	Lex	De Haan	17000.00
201	Michael	Hartstein	13000.00

In this example, first, the subquery returns the average salary of all employees. Then, the outer query uses the greater than operator to find all employees whose salaries are greater than the average.

SQL subquery with the EXISTS or NOT EXISTS operator

The `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 syntax of the `EXISTS` operator is as follows:

```
1  EXISTS (subquery )
```

The `NOT EXISTS` operator is opposite to the `EXISTS` operator.

```
1  NOT EXISTS (subquery)
```

The following example finds all departments which have at least one employee with the salary is greater than 10,000:

```
1  SELECT
2      department_name
3  FROM
4      departments d
5  WHERE
6      EXISTS( SELECT
7              1
```

HOME   VIEWS   TRIGGERS   FUNCTIONS ⌄   SQL CHEAT SHEET   SQL IN ACTION

department_name
▶ Accounting
Executive
Finance
Marketing
Purchasing
Sales

Similarly, the following statement finds all departments that do not have any employee with the salary greater than 10,000:

```
 1  SELECT
 2      department_name
 3  FROM
 4      departments d
 5  WHERE
 6      NOT EXISTS( SELECT
 7              1
 8          FROM
 9              employees e
10          WHERE
11              salary > 10000
12                  AND e.department_id = d.department_id)
13  ORDER BY department_name;
```

department_name
▶ Administration
Human Resources
IT
Public Relations
Shipping

SQL subquery with the ALL operator

The syntax of the subquery when it is used with the `ALL` operator is as follows:

```
 1  comparison_operator ALL (subquery)
```

The following condition evaluates to true if `x` is greater than every value returned by the subquery.

```
1  x > ALL (1,2,3)
```

The following query uses the `GROUP BY` clause and `MIN()` function to find the lowest salary by department:

```
1  SELECT
2      MIN(salary)
3  FROM
4      employees
5  GROUP BY department_id
6  ORDER BY MIN(salary) DESC;
```

MIN(salary)
17000.00
10000.00
8300.00
6900.00
6500.00
6200.00
6000.00
4400.00
4200.00
2700.00
2500.00

The following example finds all employees whose salaries are greater than the lowest salary of every department:

```
1  SELECT
2      employee_id, first_name, last_name, salary
3  FROM
4      employees
5  WHERE
6      salary >= ALL (SELECT
7              MIN(salary)
8          FROM
9              employees
10         GROUP BY department_id)
11 ORDER BY first_name , last_name;
```

HOME   VIEWS   TRIGGERS   FUNCTIONS ⌄   SQL CHEAT SHEET   SQL IN ACTION

SQL subquery with the ANY operator

The following shows the syntax of a subquery with the `ANY` operator:

```
1  comparison_operator ANY (subquery)
```

For example, the following condition evaluates to true if x is greater than any value returned by the subquery. So the condition `x > SOME (1,2,3)` evaluates to true if x is greater than 1.

```
1  x > ANY (subquery)
```

Note that the `SOME` operator is a synonym for the `ANY` operator so you can use them interchangeably.

The following query finds all employees whose salaries are greater than or equal to the highest salary of every department.

```
 1  SELECT
 2      employee_id, first_name, last_name, salary
 3  FROM
 4      employees
 5  WHERE
 6      salary >= SOME (SELECT
 7              MAX(salary)
 8          FROM
 9              employees
10          GROUP BY department_id);
```

HOME   VIEWS   TRIGGERS   FUNCTIONS ⌄   SQL CHEAT SHEET   SQL IN ACTION

114	Den	Raphaely	11000.00
204	Hermann	Baer	10000.00
111	Ismael	Sciarra	7700.00
177	Jack	Livingston	8400.00
200	Jennifer	Whalen	4400.00

In this example, the subquery finds the highest salary of employees in each department. The outer query looks at these values and determines which employee's salaries are greater than or equal to any highest salary by department.

SQL subquery in the FROM clause

You can use a subquery in the FROM clause of the SELECT statement as follows:

```
1  SELECT
2      *
3  FROM
4      (subquery) AS table_name
```

In this syntax, the table alias is mandatory because all tables in the FROM clause must have a name.

Note that the subquery specified in the FROM clause is called a derived table in MySQL or inline view in Oracle.

The following statement returns the average salary of every department:

```
1  SELECT
2      AVG(salary) average_salary
3  FROM
4      employees
5  GROUP BY department_id;
```

↑

	10000.000000
	9616.666667
	19333.333333
	8600.000000
	10150.000000

You can use this query as a subquery in the `FROM` clause to calculate the average of average salary of departments as follows:

```
1  SELECT
2      ROUND(AVG(average_salary), 0)
3  FROM
4      (SELECT
5          AVG(salary) average_salary
6      FROM
7          employees
8      GROUP BY department_id) department_salary;
```

ROUND(AVG(average_salary), 0)
8536

SQL Subquery in the SELECT clause

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

```
1  SELECT
2      employee_id,
3      first_name,
4      last_name,
5      salary,
6      (SELECT
7              ROUND(AVG(salary), 0)
8          FROM
9              employees) average_salary,
10     salary - (SELECT
11             ROUND(AVG(salary), 0)
```

employee_id	first_name	last_name	salary	average_salary	difference
121	Adam	Fripp	8200.00	8060	140.00
103	Alexander	Hunold	9000.00	8060	940.00
115	Alexander	Khoo	3100.00	8060	-4960.00
193	Britney	Everett	3900.00	8060	-4160.00
104	Bruce	Ernst	6000.00	8060	-2060.00
179	Charles	Johnson	6200.00	8060	-1860.00
109	Daniel	Faviet	9000.00	8060	940.00
105	David	Austin	4800.00	8060	-3260.00
114	Den	Raphaely	11000.00	8060	2940.00
107	Diana	Lorentz	4200.00	8060	-3860.00
118	Guy	Himuro	2600.00	8060	-5460.00
204	Hermann	Baer	10000.00	8060	1940.00

Now you should understand what an SQL subquery is and how to use subqueries to form flexible SQL statements.

Was this tutorial helpful ?   👍 Yes   👎 No

HOME   VIEWS   TRIGGERS   FUNCTIONS ⌄   SQL CHEAT SHEET   SQL IN ACTION

GETTING STARTED

What Is SQL

SQL Syntax

SQL Sample Database

SQL TUTORIAL

SQL SELECT

SQL ORDER BY

SQL DISTINCT

SQL LIMIT

SQL FETCH

SQL WHERE

SQL Comparison Operators

SQL Logical Operators

SQL AND

HOME   VIEWS   TRIGGERS   FUNCTIONS ⌄   SQL CHEAT SHEET   SQL IN ACTION

SQL NOT

SQL IS NULL

SQL Alias

SQL INNER JOIN

SQL LEFT JOIN

SQL SELF JOIN

SQL FULL OUTER JOIN

SQL CROSS JOIN

SQL GROUP BY

SQL GROUPING SETS

SQL ROLLUP

SQL CUBE

SQL HAVING

SQL Subquery

SQL Correlated Subquery

SQL ALL

SQL ANY

SQL EXISTS

SQL UNION

SQL INTERSECT

SQL CASE

SQL MINUS

SQL INSERT

HOME   VIEWS   TRIGGERS   FUNCTIONS ∨   SQL CHEAT SHEET   SQL IN ACTION

SQL AVG

SQL COUNT

SQL MAX

SQL MIN

SQL SUM

MANAGING DATABASE OBJECTS

SQL Data Types

SQL CREATE TABLE

SQL Identity

SQL Auto Increment

SQL ALTER TABLE

SQL ADD COLUMN

SQL DROP COLUMN

SQL DROP TABLE

HOME   VIEWS   TRIGGERS   FUNCTIONS ⌄   SQL CHEAT SHEET   SQL IN ACTION

SQL Foreign Key

SQL UNIQUE Constraint

SQL CHECK Constraint

SQL NOT NULL Constraint
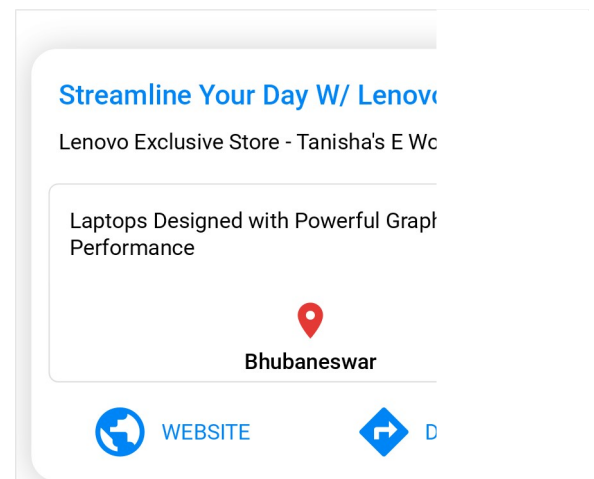
The SQLTutorial.org is created to help you master the SQL language fast by using simple but practical examples and easy-to-understand explanations.

Search this website

RECENT TUTORIALS

SQL DROP COLUMN

SQL Identity

SQL Auto Increment

SQL ADD COLUMN

SQL PARTITION BY

SITE LINKS

Home

⬆

HOME   VIEWS   TRIGGERS   FUNCTIONS ⌄   SQL CHEAT SHEET   SQL IN ACTION