

Unknown date  
Unknown author

# MySQL Subqueries

## MySQL Subqueries

Last update on November 09 2019 06:55:41 (UTC/GMT +8 hours)

### Subqueries

A subquery is a SQL query nested inside a larger query.

- A subquery may occur in:
  - - A SELECT clause
  - - A FROM clause
  - - A WHERE clause
- In MySQL subquery can be nested inside a SELECT, INSERT, UPDATE, DELETE, SET, or DO statement or inside another subquery.
- A subquery is usually added within the WHERE Clause of another SQL SELECT statement.
- You can use the comparison operators, such as >, <, or =. The comparison operator can also be a multiple-row operator, such as IN, ANY, SOME, or ALL.
- A subquery can be treated as an inner query, which is a SQL query placed as a part of another query called as outer query.
- The inner query executes first before its parent query so that the results of the inner query can be passed to the outer query.

#### Contents:

- [Subquery Syntax](#)
- [MySQL Subquery Example](#)
- [Subqueries : Guidelines and Types of Subqueries](#)
- [MySQL Subquery as Scalar Operand](#)
- [MySQL Subqueries : Using Comparisons](#)
- [MySQL Subqueries with ALL, ANY, IN, or SOME](#)
- [MySQL Row Subqueries](#)

[MySQL Subqueries with EXISTS or NOT EXISTS](#)

[MySQL Correlated Subqueries](#)

[MySQL Subqueries in the FROM Clause](#)

#### Subquery Syntax :

- The subquery (inner query) executes once before the main query (outer query) executes.
- The main query (outer query) use the subquery result.

#### Subquery syntax as specified by the SQL standard and supported in MySQL

```
DELETE FROM t1
WHERE s11 > ANY
(SELECT COUNT(*) /* no hint */ FROM t2
WHERE NOT EXISTS
(SELECT * FROM t3
WHERE ROW(5*t2.s1,77)=
(SELECT 50,11*s1 FROM t4 UNION SELECT 50,77 FROM
(SELECT * FROM t5) AS t5)));
```

A subquery can return a scalar (a single value), a single row, a single column, or a table (one or more rows of one or more columns). These are called scalar, column, row, and table subqueries.

#### MySQL Subquery Example:

Using a subquery, list the name of the employees, paid more than 'Alexander' from emp\_details .

EMPLOYEE_ID	FIRST_NAME	LAST_N
100	Steven	King
101	Neena	Kochhar
102	Lex	De Haan
103	Alexander	Hunold
104	Bruce	Ernst
105	David	Austin

```
mysql> SELECT first_name,last_name, salary FROM emp_details
WHERE salary >(SELECT salary FROM emp_details
WHERE first_name='Alexander');
+-----+-----+-----+
| first_name | last_name | salary |
+-----+-----+-----+
| Steven | King | 24000.00 |
| Neena | Kochhar | 17000.00 |
| Lex | De Haan | 17000.00 |
| RABI | CHANDRA | 15000.00 |
| Ana | King | 17000.00 |
+-----+-----+-----+
5 rows in set (0.00 sec)
```

### Subqueries: Guidelines

There are some guidelines to consider when using subqueries :

- A subquery must be enclosed in parentheses.
- Use single-row operators with single-row subqueries, and use multiple-row operators with multiple-row subqueries.
- If a subquery (inner query) returns a null value to the outer query, the outer query will not return any rows when using certain comparison operators in a WHERE clause.

### Types of Subqueries

- The Subquery as Scalar Operand
- Comparisons using Subqueries
- Subqueries with ALL, ANY, IN, or SOME
- Row Subqueries
- Subqueries with EXISTS or NOT EXISTS
- Correlated Subqueries
- Subqueries in the FROM Clause

### MySQL Subquery as Scalar Operand

A scalar subquery is a subquery that returns exactly one column value from one row. A scalar subquery is a simple operand, and you can use it almost anywhere a single column value or literal is legal. If the subquery returns 0 rows then the value of scalar subquery expression in NULL and if the subquery returns more than one row then MySQL returns an error.

There is some situation where a scalar subquery cannot be used. If a statement permits only a literal value, you cannot use a subquery. For example, LIMIT requires literal integer arguments, and LOAD DATA INFILE requires a literal string file name. You cannot use subqueries to supply these values.

### Example: MySQL Subquery as Scalar Operand

```
mysql> SELECT employee_id, last_name,
(CASE WHEN department_id=(
SELECT department_id from departments WHERE location_id=2500)
THEN 'Canada' ELSE 'USA' END)
location FROM employees;
+-----+-----+-----+
| employee_id | last_name | location |
+-----+-----+-----+
| 100 | King | USA |
| 101 | Kochhar | USA |
| 102 | De Haan | USA |
| 103 | Hunold | USA |
| 104 | Ernst | USA |
| 105 | Austin | USA |
| -----|
| -----|
107 rows in set (0.00 sec)
```

### MySQL Subqueries: Using Comparisons

A subquery can be used before or after any of the comparison operators. The subquery can return at most one value. The value can be the result of an arithmetic expression or a column function. SQL then compares the value that results from the subquery with the value on the other side of the comparison operator. You can use the following comparison operators:

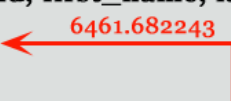
Operator	Description
=	Equal to
>	Greater than
>=	Greater than or equal to
<	Less than
<=	Less than or equal to
!=	Not equal to
<>	Not equal to
<=>	NULL-safe equal to operator

For example, suppose you want to find the employee id, first\_name, last\_name, and salaries for employees whose average salary is higher than the average salary throughout the company.

```

Select  employee_id, first_name, last_name, salary
From    employees
Where   salary >
        ( Select  AVG ( salary )
          From    employees );

```



```

mysql> SELECT employee_id,first_name,last_name,salary
FROM employees WHERE salary >
(SELECT AVG(SALARY) FROM employees);
+-----+-----+-----+-----+
| employee_id | first_name | last_name | salary |
+-----+-----+-----+-----+
| 100 | Steven | King | 24000.00 |
| 101 | Neena | Kochhar | 17000.00 |
| 102 | Lex | De Haan | 17000.00 |
| 103 | Alexander | Hunold | 9000.00 |
| 108 | Nancy | Greenberg | 12000.00 |
| 109 | Daniel | Faviet | 9000.00 |
| 120 | Matthew | Weiss | 8000.00 |
| 121 | Adam | Fripp | 8200.00 |
| 122 | Payam | Kaufling | 7900.00 |
+-----+-----+-----+-----+
51 rows in set (0.00 sec)

```

### MySQL Subqueries with ALL, ANY, IN, or SOME

You can use a subquery after a comparison operator, followed by the keyword ALL, ANY, or SOME.

The ALL operator compares value to every value returned by the subquery. Therefore ALL operator (which must follow a comparison operator) returns TRUE if the comparison is TRUE for ALL of the values in the column that the subquery returns.

#### Syntax:

operand comparison\_operator ALL (subquery)

NOT IN is an alias for <> ALL. Thus, these two statements are the same:

Code:

```

SELECT c1 FROM t1 WHERE c1 <> ALL (SELECT c1 FROM t2);
SELECT c1 FROM t1 WHERE c1 NOT IN (SELECT c1 FROM t2);

```

Copy

Example: MySQL Subquery, ALL operator

The following query selects the department with the highest average salary. The subquery finds the average salary for each department, and then the main query selects the department with the highest average salary.

EMPLOYEE_ID	FIRST_NAME	LAST_NAME
100	Steven	King
101	Neena	Kochhar
102	Lex	De Haan
103	Alexander	Hunold
104	Bruce	Ernst
105	Daniel	Faviet

```

mysql> SELECT department_id, AVG(SALARY)
FROM EMPLOYEES GROUP BY department_id
HAVING AVG(SALARY)>=ALL
(SELECT AVG(SALARY) FROM EMPLOYEES GROUP BY department_id);
+-----+-----+
| department_id | AVG(SALARY) |
+-----+-----+
| 90 | 19333.333333 |
+-----+-----+
1 row in set (0.00 sec)

```

Note: Here we have used ALL keyword for this subquery as the department selected by the query must have an average salary greater than or equal to all the average salaries of the other departments.

The ANY operator compares the value to each value returned by the subquery. Therefore ANY keyword (which must follow a comparison operator) returns TRUE if the comparison is TRUE for ANY of the values in the column that the subquery returns.

#### Syntax:

operand comparison\_operator ANY (subquery)

Example: MySQL Subquery, ANY operator

The following query selects any employee who works in the location 1800. The subquery finds the department id in the 1800 location, and then the main query selects the employees who work in any of these departments.

**employees table :**

EMPLOYEE_ID	FIRST_NAME	LAST_N
100	Steven	King
101	Neena	Kochhar
102	Lex	De Haan
103	Alexander	Hunold
104	Bruce	Ernst
105	Daniel	Alien

**departments table:**

DEPARTMENT_ID	DEPARTMENT_NAME
10	Administration
20	Marketing
30	Purchasing
40	Human Resources
50	Shipping

```
mysql> SELECT first_name, last_name, department_id
FROM employees WHERE department_id= ANY
(SELECT DEPARTMENT_ID FROM departments WHERE location_id=1800);
+-----+-----+-----+
| first_name | last_name | department_id |
+-----+-----+-----+
| Michael | Hartstein | 20 |
| Pat | Fay | 20 |
+-----+-----+-----+
2 rows in set (0.00 sec)
```

Note: We have used ANY keyword in this query because it is likely that the subquery will find more than one departments in 1800 location. If you use the ALL keyword instead of the ANY keyword, no data is selected because no employee works in all departments of 1800 location

When used with a subquery, the word IN (equal to any member of the list) is an alias for = ANY. Thus, the following two statements are the same:

Code:

```
SELECT c1 FROM t1 WHERE c1 = ANY (SELECT c1 FROM t2);
SELECT c1 FROM t1 WHERE c1 IN (SELECT c1 FROM t2);
```

Copy

The word SOME is an alias for ANY. Thus, these two statements are the same:

Code:

```
SELECT c1 FROM t1 WHERE c1 <> ANY (SELECT c1 FROM t2);
SELECT c1 FROM t1 WHERE c1 <> SOME (SELECT c1 FROM t2);
```

Copy

#### MySQL Row Subqueries

A row subquery is a subquery that returns a single row and more than one column value. You can use =, >, <, >=, <=, <>, !=, <=> [comparison operators](#). See the following examples:

Code:

```
SELECT * FROM table1 WHERE (col1,col2) = (SELECT col3, col4 FROM table2 WHERE id = 10);
SELECT * FROM table1 WHERE ROW(col1,col2) = (SELECT col3, col4 FROM table2 WHERE id = 10);
```

Copy

For both queries,

- if the table table2 contains a single row with id = 10, the subquery returns a single row. If this row has col3 and col4 values equal to the col1 and col2 values of any rows in table1, the WHERE expression is TRUE and each query returns those table1 rows.
- If the table2 row col3 and col4 values are not equal the col1 and col2 values of any table1 row, the expression is FALSE and the query returns an empty result set. The expression is unknown (that is, NULL) if the subquery produces no rows.
- An error occurs if the subquery produces multiple rows because a row subquery can return at most one row.

### Example: MySQL Row Subqueries

In the following examples, queries shows different result according to above conditions :

**departments table:**

DEPARTMENT_ID	DEPARTMENT_NAME
10	Administration
20	Marketing
30	Purchasing
40	Human Resources
50	Shipping

**employees table:**

EMPLOYEE_ID	FIRST_NAME	LAST_NAME
100	Steven	King
101	Neena	Kochhar
102	Lex	De Haan
103	Alexander	Hunold
104	Bruce	Ernst
105	David	Austin

```
mysql> SELECT first_name
FROM employees
WHERE ROW(department_id, manager_id) = (SELECT department_id, manager_id FROM
departments WHERE location_id = 1800);
+-----+
| first_name |
+-----+
| Pat |
+-----+
1 row in set (0.00 sec)
```

Code:

```
mysql>SELECT first_name
FROM employees
WHERE ROW(department_id, manager_id) = (SELECT department_id, manager_id FROM departments WHERE location_
Empty set (0.00 sec)
```

Copy

Code:

```
mysql>SELECT first_name
FROM employees
WHERE ROW(department_id, manager_id) = (SELECT department_id, manager_id FROM departments WHERE location_
ERROR 1242 (21000): Subquery returns more than 1 row
```

Copy

### MySQL Subqueries with EXISTS or NOT EXISTS

The EXISTS operator tests for the existence of rows in the results set of the subquery. If a subquery row value is found, EXISTS subquery is TRUE and in this case NOT EXISTS subquery is FALSE.

**Syntax:**

```
SELECT column1 FROM table1 WHERE EXISTS (SELECT * FROM table2);
```

In the above statement, if table2 contains any rows, even rows with NULL values, the EXISTS condition is TRUE. Generally, an EXISTS subquery starts with SELECT \*, but it could begin with SELECT 'X', SELECT 5, or SELECT column1 or anything at all. MySQL ignores the SELECT list in such a subquery, so it makes no difference.

Example: MySQL Subqueries with EXISTS

From the following tables (employees) find employees (employee\_id, first\_name, last\_name, job\_id, department\_id) who have at least one person reporting to them.

**employees table:**

EMPLOYEE_ID	FIRST_NAME	LAST_NAME
100	Steven	King
101	Neena	Kochhar
102	Lex	De Haan
103	Alexander	Hunold
104	Bruce	Ernst
105	David	Austin
106	Valli	Databalla

```
SELECT employee_id, first_name, last_name, job_id, department_id
FROM employees E
WHERE EXISTS (SELECT * FROM employees WHERE manager_id = E.employee_id);
```

employee_id	first_name	last_name	job_id	department_id
100	Steven	King	AD_PRES	90
101	Neena	Kochhar	AD_VP	90
102	Lex	De Haan	AD_VP	90
103	Alexander	Hunold	IT_PROG	60
108	Nancy	Greenberg	FI_MGR	100
114	Den	Raphaely	PU_MAN	30
120	Matthew	Weiss	ST_MAN	50
121	Adam	Fripp	ST_MAN	50

18 rows in set (0.02 sec)

Example: MySQL Subqueries with NOT EXISTS

NOT EXISTS subquery almost always contains correlations. Here is an example :  
From the following table (departments and employees) find all departments (department\_id, department\_name) that do not have any employees.

**departments table:**

DEPARTMENT_ID	DEPARTMENT_NAME
10	Administration
20	Marketing
30	Purchasing
40	Human Resources
50	Shipping
60	IT

**employees table:**

EMPLOYEE_ID	FIRST_NAME	LAST_NAME
100	Steven	King
101	Neena	Kochhar
102	Lex	De Haan
103	Alexander	Hunold
104	Bruce	Ernst
105	David	Austin
106	Valli	Databalla

```
mysql> SELECT department_id, department_name
FROM departments d
WHERE NOT EXISTS (SELECT * FROM employees WHERE department_id = d.department_id);
```

department_id	department_name
120	Treasury
130	Corporate Tax
140	Control And Credit
150	Shareholder Services
160	Benefits
170	Manufacturing
180	Construction
190	Contracting
200	Operations

16 rows in set (0.00 sec)

### MySQL Correlated Subqueries

A correlated subquery is a subquery that contains a reference to a table (in the parent query) that also appears in the outer query. MySQL evaluates from inside to outside.

#### Correlated subquery syntax:

Example - 1: MySQL Correlated Subqueries

Following query find all employees who earn more than the average salary in their department.

employees table:

EMPLOYEE_ID	FIRST_NAME	LAST_NAME
100	Steven	King
101	Neena	Kochhar
102	Lex	De Haan
103	Alexander	Hunold
104	Bruce	Ernst
105	David	Austin
106	Valli	Databalla

```
mysql> SELECT last_name, salary, department_id
FROM employees outerr
WHERE salary > (SELECT AVG(salary) FROM employees WHERE department_id = outerr.department_id);
```

last_name	salary	department_id
King	24000.00	90
Hunold	9000.00	60
Ernst	6000.00	60
Greenberg	12000.00	100
Faviet	9000.00	100
Raphaely	11000.00	30
Weiss	8000.00	50
Frapp	8200.00	50

38 rows in set (0.02 sec)

Example - 2: MySQL Correlated Subqueries

From the employees and job\_history tables display details of those employees who have changed jobs at least once.

employees table:

EMPLOYEE_ID	FIRST_NAME	LAST_NAME
100	Steven	King
101	Neena	Kochhar
102	Lex	De Haan
103	Alexander	Hunold
104	Bruce	Ernst
105	David	Austin
106	Valli	Databalla

job\_history table:

EMPLOYEE_ID	START_DATE	END_DATE
102	13-Jan-93	24-Jul-93
101	21-Sep-89	27-Oct-93
101	28-Oct-93	15-Mar-94
201	17-Feb-96	19-Dec-97
114	24-Mar-98	31-Dec-99
122	1-Jan-99	31-Dec-99

```
mysql> SELECT first_name, last_name, employee_id, job_id
FROM employees E
WHERE 1 <= (SELECT COUNT(*) FROM Job_history WHERE employee_id = E.employee_id);
```

first_name	last_name	employee_id	job_id
Neena	Kochhar	101	AD_VP
Lex	De Haan	102	AD_VP
Den	Raphaely	114	PU_MAN
Payam	Kaufling	122	ST_MAN
Jonathon	Taylor	176	SA_REP
Jennifer	Whalen	200	AD_ASST
Michael	Hartstein	201	MK_MAN

7 rows in set (0.00 sec)

### MySQL Subqueries in the FROM Clause

Subqueries work in a SELECT statement's FROM clause. The syntax is :

```
SELECT ... FROM (subquery) [AS] name ...
```

Every table in a FROM clause must have a name, therefore the [AS] name clause is mandatory. Any columns in the subquery select list must have unique names.

Example: MySQL Subqueries in the FROM Clause

We have the following table tb1.

```
mysql> CREATE TABLE tb1 (c1 INT, c2 CHAR(5), c3 FLOAT); Query OK, 0 rows affected (0.73 sec)
```

Let insert some values into tb1.

```
mysql> INSERT INTO tb1 VALUES (1, '1', 1.0);  
Query OK, 1 row affected (0.11 sec)
```

```
mysql> INSERT INTO tb1 VALUES (2, '2', 2.0);  
Query OK, 1 row affected (0.07 sec)
```

```
mysql> INSERT INTO tb1 VALUES (3, '3', 3.0);  
Query OK, 1 row affected (0.03 sec)
```

```
mysql> select * from tb1;
```

c1	c2	c3
1	1	1
2	2	2
3	3	3

3 rows in set (0.00 sec)

Here is how to use a subquery in the FROM clause, using the example table (tb1) :

```
mysql> SELECT sc1, sc2, sc3  
FROM (SELECT c1 AS sc1, c2 AS sc2, c3*3 AS sc3 FROM tb1) AS sb  
WHERE sc1 > 1;
```

sc1	sc2	sc3
2	2	6
3	3	9

2 rows in set (0.02 sec)

Previous: [NATURAL JOIN](#)

Next: [MySQL Procedure](#)