

SUBQUERIES

MySQL

What is a subquery?

A Subquery or Inner query or a Nested query is a query within another SQL query (main query) and embedded within the WHERE clause.

A subquery is used to return data that will be used in the main query as a condition to further restrict the data to be retrieved.

Subqueries can be used with the SELECT, INSERT, UPDATE, and DELETE statements along with the operators like, =, <, >, >=, <=, IN, BETWEEN, etc.

Subqueries rules

- **Subqueries must be enclosed within parentheses.**
- **A subquery can have only one column in the SELECT clause, unless multiple columns are in the main query for the subquery to compare its selected columns.**
- **An ORDER BY command cannot be used in a subquery, although the main query can use an ORDER BY. The GROUP BY command can be used to perform the same function as the ORDER BY in a subquery.**
- **Subqueries that return more than one row can only be used with multiple value operators such as the IN operator.**
- **The SELECT list cannot include any references to values that evaluate to a BLOB, ARRAY, CLOB, or NCLOB.**
- **A subquery cannot be immediately enclosed in a set function.**
- **The BETWEEN operator cannot be used with a subquery. However, the BETWEEN operator can be used within the subquery.**
- **The inner query executes first before its parent query so that the results of the inner query can be passed to the outer query.**
- **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.**

Subqueries with SELECT statement

```
SELECT column_name [, column_name ]  
FROM table1 [, table2 ]  
WHERE column_name OPERATOR  
(  
    SELECT column_name  
    FROM table1 [, table2 ]  
    [WHERE]  
)
```

Consider the CUSTOMERS table having the following records –

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	35	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	MP	4500.00
7	Muffy	24	Indore	10000.00

Now, let us check the following subquery with a SELECT statement.

```
SQL> SELECT *  
      FROM CUSTOMERS  
      WHERE ID IN (SELECT ID  
                  FROM CUSTOMERS  
                  WHERE SALARY > 4500) ;
```

This would produce the following result.

ID	NAME	AGE	ADDRESS	SALARY
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
7	Muffy	24	Indore	10000.00

Subqueries in the SELECT list

Subqueries can occur in the select list of the containing query. The results from the following statement are ordered by the first column (customer_name). You could also write ORDER BY 2 and specify that the results be ordered by the select-list subquery.

```
=> SELECT c.customer_name, (SELECT AVG(annual_income) FROM customer_dimension  
    WHERE deal_size = c.deal_size) AVG_SAL_DEAL FROM customer_dimension c  
    ORDER BY 1;
```

customer_name	AVG_SAL_DEAL
Goldstar	603429
Metatech	628086
Metadata	666728
Foodstar	695962
Verihope	715683
Veridata	868252
Bettercare	879156
Foodgen	958954
Virtacom	991551
Inicorp	1098835
...	

Subqueries with INSERT statement

```
INSERT INTO table_name [ (column1[, column2 ]) ]  
SELECT [ *|column1 [, column2 ]  
FROM table1 [, table2 ]  
[ WHERE VALUE OPERATOR ]
```

```
SQL> INSERT INTO CUSTOMERS_BKP  
      SELECT * FROM CUSTOMERS  
      WHERE ID IN (SELECT ID  
                  FROM CUSTOMERS) ;
```

Subqueries with UPDATE statement

UPDATE table

SET column_name = new_value

[WHERE OPERATOR [VALUE]

**(
SELECT COLUMN_NAME
FROM TABLE_NAME**

**)
[WHERE)]**

Assuming, we have CUSTOMERS_BKP table available which is backup of CUSTOMERS table. The following example updates SALARY by 0.25 times in the CUSTOMERS table for all the customers whose AGE is greater than or equal to 27.

```
SQL> UPDATE CUSTOMERS  
      SET SALARY = SALARY * 0.25  
      WHERE AGE IN (SELECT AGE FROM CUSTOMERS_BKP  
                    WHERE AGE >= 27 );
```

This would impact two rows and finally CUSTOMERS table would have the following records.

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	35	Ahmedabad	125.00
2	Khilan	25	Delhi	1500.00
3	kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	2125.00
6	Komal	22	MP	4500.00
7	Muffy	24	Indore	10000.00

Subqueries with DELETE statement

```
DELETE FROM TABLE_NAME
[WHERE OPERATOR [ VALUE ]
(
    SELECT COLUMN_NAME
    FROM TABLE_NAME
)
[ WHERE) ]
```

```
SQL> DELETE FROM CUSTOMERS
      WHERE AGE IN (SELECT AGE FROM CUSTOMERS_BKP
                    WHERE AGE >= 27 );
```

This would impact two rows and finally the CUSTOMERS table would have the following records.

ID	NAME	AGE	ADDRESS	SALARY
2	Khilan	25	Delhi	1500.00
3	kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
6	Komal	22	MP	4500.00
7	Muffy	24	Indore	10000.00

Subqueries as a 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 is 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

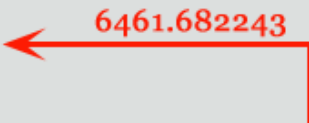
```
mysql> SELECT employee_id, last_name,  
(CASE WHEN department_id=(  
SELECT department_id from departments WHERE location_id=2500)  
THEN 'Canada' ELSE 'USA' END)  
location FROM employees;
```

employee_id	last_name	location
100	King	USA
101	Kochhar	USA
102	De Haan	USA
103	Hunold	USA
104	Ernst	USA
105	Austin	USA

<https://www.w3resource.com/mysql/subqueries/index.php>

Subqueries using Comparisons

```
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);
```

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.

Subqueries with ALL

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.

```
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);
```

<https://www.w3resource.com/mysql/subqueries/index.php>

Subqueries with ANY

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.

```
mysql> SELECT first_name, last_name, department_id  
FROM employees WHERE department_id= ANY  
(SELECT DEPARTMENT_ID FROM departments WHERE location_id=1800);
```

Subqueries with IN & SOME

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:

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

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

Code:

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

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:

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

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.

```
mysql> SELECT first_name
FROM employees
WHERE ROW(department_id, manager_id)
= (SELECT department_id, manager_id
FROM departments WHERE location_id = 1)
```

<https://www.w3resource.com/mysql/subqueries/index.php>

Subqueries with EXISTS & 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:

**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);**

<https://www.w3resource.com/mysql/subqueries/index.php>

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.

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

<https://www.w3resource.com/mysql/subqueries/index.php>

Subqueries in the FROM clause

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.

Table tb1

+-----+-----+-----+			
c1	c2	c3	
+-----+-----+-----+			
1	1	1	
2	2	2	
3	3	3	
+-----+-----+-----+			

```
SELECT sc1, sc2, sc3  
FROM  
(
```

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

RESULT

+-----+-----+-----+			
sc1	sc2	sc3	
+-----+-----+-----+			
2	2	6	
3	3	9	
+-----+-----+-----+			