### **Subquery in MySQL**

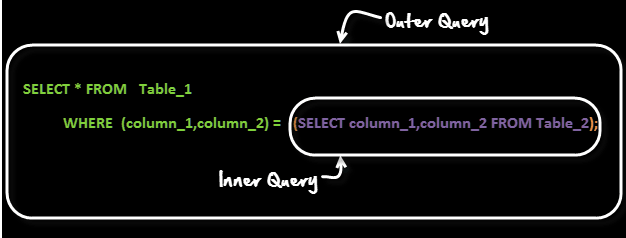
### Definition:

A subquery, also known as an inner query or nested query, is a query nested within another SQL query. It allows you to use the result of one query as a part of another query.

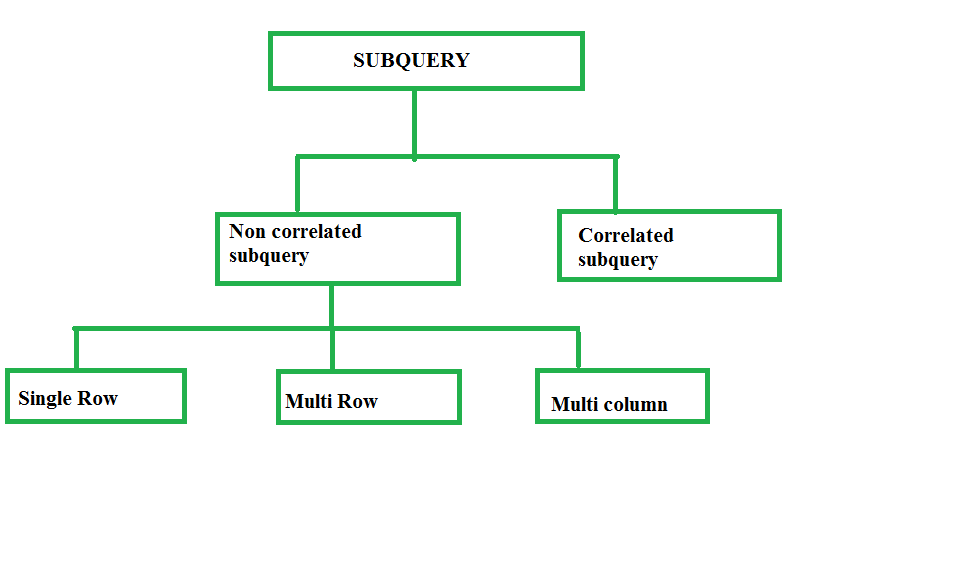
Or

A Subquery or Inner query or a Nested query allows us to create complex query on the output of another query.

Sub query syntax involves two SELECT statements



### **Types of Subqueries:**



#### **SYNTAX**

#### **Scalar Subquery:**

A scalar subquery returns a single value (scalar) and can be used wherever an expression is allowed.

SELECT column1, (SELECT MAX(column2) FROM table2) AS max\_value0

FROM table1;

#### MySQL SubQuery Tutorial with Examples

#### **Single-Row Subquery:**

A single-row subquery returns one row of data to the outer query. It's used with comparison operators that expect single values.

SELECT column1

FROM table1

WHERE column2 = (SELECT column2 FROM table2 WHERE condition);

#### **Multi-Row Subquery:**

A multi-row subquery returns multiple rows of data to the outer query. It's used with operators that expect multiple values.

SELECT column1

FROM table1

WHERE column2 IN (SELECT column2 FROM table2 WHERE condition);

#### 

#### **Correlated Subquery:**

A correlated subquery depends on the outer query and is evaluated once for each row processed by the outer query.

SELECT column1

FROM table1 t1

WHERE column2 = (SELECT MAX(column2) FROM table2 WHERE t1.column3 = table2.column3);

### 

### **Usage of Subqueries:**

#### **In SELECT Statement:**

Subqueries can be used in the SELECT statement to retrieve additional information.

SELECT column1, (SELECT MAX(column2) FROM table2) AS max\_value

FROM table1;

#### **In WHERE Clause:**

Subqueries can be used in the WHERE clause to filter rows based on certain conditions.

SELECT column1

FROM table1

WHERE column2 = (SELECT column2 FROM table2 WHERE condition);

#### 

#### **In FROM Clause:**

Subqueries can be used in the FROM clause to treat the result of the subquery as a temporary table.

SELECT \*

FROM (SELECT column1 FROM table1 WHERE condition) AS subquery\_table;

#### 

#### **In INSERT, UPDATE, DELETE Statements:**

Subqueries can be used in INSERT, UPDATE, and DELETE statements to perform operations based on the result of another query.

DELETE FROM table1

WHERE column1 IN (SELECT column1 FROM table2 WHERE condition);

### 

### **Subquery Best Practices:**

* Use subqueries when necessary, but avoid nesting too deeply as it can impact readability and performance.
* Test subqueries separately to ensure they return the expected results before incorporating them into larger queries.
* Optimize subqueries by ensuring proper indexing on columns used in subqueries' conditions.

Subqueries are powerful tools in SQL that allow you to perform complex data manipulations and retrieve specific information from your database. Understanding how to use them effectively can greatly enhance your SQL querying capabilities. Adjust the table names, column names, and conditions according to your database schema.

**Practical**

**-- Create Database**

CREATE DATABASE IF NOT EXISTS mydatabase;

USE mydatabase;

-- Create `employees` Table

CREATE TABLE employees (

employee\_id INT PRIMARY KEY,

first\_name VARCHAR(50),

last\_name VARCHAR(50),

department\_id INT,

salary DECIMAL(10, 2)

);

-- Create `departments` Table

CREATE TABLE departments (

department\_id INT PRIMARY KEY,

department\_name VARCHAR(50)

);

-- Create `projects` Table

CREATE TABLE projects (

project\_id INT PRIMARY KEY,

project\_name VARCHAR(50),

manager\_id INT

);

-- Insert Data into `employees` Table

INSERT INTO employees (employee\_id, first\_name, last\_name, department\_id, salary) VALUES

(1, 'John', 'Doe', 101, 60000.00),

(2, 'Jane', 'Smith', 102, 75000.00),

(3, 'Alice', 'Johnson', 101, 50000.00),

(4, 'Bob', 'Lee', 103, 55000.00),

(5, 'Charlie', 'Brown', 102, 70000.00);

-- Insert Data into `departments` Table

INSERT INTO departments (department\_id, department\_name) VALUES

(101, 'HR'),

(102, 'IT'),

(103, 'Finance');

-- Insert Data into `projects` Table

INSERT INTO projects (project\_id, project\_name, manager\_id) VALUES

(1001, 'Alpha', 2),

(1002, 'Beta', 3),

(1003, 'Gamma', 5);

Let's assume we have the following tables:

**Table: employees**

| **employee\_id** | **first\_name** | **last\_name** | **department\_id** | **salary** |
| --- | --- | --- | --- | --- |
| 1 | John | Doe | 101 | 60000 |
| 2 | Jane | Smith | 102 | 75000 |
| 3 | Alice | Johnson | 101 | 50000 |
| 4 | Bob | Lee | 103 | 55000 |
| 5 | Charlie | Brown | 102 | 70000 |

**Table: departments**

| **department\_id** | **department\_name** |
| --- | --- |
| 101 | HR |
| 102 | IT |
| 103 | Finance |

**Table: projects**

| **project\_id** | **project\_name** | **manager\_id** |
| --- | --- | --- |
| 1001 | Alpha | 2 |
| 1002 | Beta | 3 |
| 1003 | Gamma | 5 |

### **Types of Subqueries**

#### Subquery in oracle SQL ~ whereisstuff

#### **Scalar Subquery**

A scalar subquery returns a single value. It can be used where an expression is expected.

**Example: Get the highest salary among all employees and display it along with each employee's details.**

**SELECT   
    employee\_id,   
    first\_name,   
    last\_name,   
    salary,   
    (SELECT MAX(salary) FROM employees) AS highest\_salary  
FROM employees;**

#### **Single-Row Subquery**

A single-row subquery returns one row of data to the outer query.

**Example: Find the employee whose salary is the highest.**

**SELECT first\_name, last\_name, salary  
FROM employees  
WHERE salary = (SELECT MAX(salary) FROM employees);**

#### **Multi-Row Subquery**

A multi-row subquery returns multiple rows of data. It is used with operators that expect multiple values like IN.

**Example: Find employees who work in the IT department.**

S**ELECT first\_name, last\_name  
FROM employees  
WHERE department\_id IN (SELECT department\_id FROM departments WHERE department\_name = 'IT');**

#### **Correlated Subquery**

A correlated subquery is evaluated once for each row processed by the outer query.

**Example: Find employees whose salary is greater than the average salary of their respective departments.**

**SELECT employee\_id, first\_name, last\_name, salary  
FROM employees e  
WHERE salary > (SELECT AVG(salary)   
                FROM employees   
                WHERE department\_id = e.department\_id);**

### **Usage of Subqueries**

#### **In SELECT Statement**

**Example: Get the highest salary in each department along with employee details.**

**SELECT employee\_id, first\_name, last\_name, department\_id, salary,  
       (SELECT MAX(salary) FROM employees e2 WHERE e2.department\_id = e1.department\_id) AS highest\_salary\_in\_dept  
FROM employees e1;**

#### **In WHERE Clause**

**Example: Find employees who are managers of any project.**

**SELECT first\_name, last\_name  
FROM employees  
WHERE employee\_id IN (SELECT manager\_id FROM projects);**

#### **In FROM Clause**

**Example: Get the average salary of employees grouped by department.**

**SELECT dept\_avg.department\_id, dept\_avg.avg\_salary  
FROM (SELECT department\_id, AVG(salary) AS avg\_salary  
      FROM employees  
      GROUP BY department\_id) AS dept\_avg;**

#### **In INSERT, UPDATE, DELETE Statements**

**Example: Delete employees who do not manage any projects.**

**DELETE FROM employees  
WHERE employee\_id NOT IN (SELECT manager\_id FROM projects);**

### **Subquery Best Practices**

1. **Use subqueries when necessary**: Subqueries are useful but can become complex. Avoid excessive nesting.
2. **Test subqueries separately**: Ensure subqueries return the expected results before incorporating them.
3. **Optimize subqueries**: Use proper indexing on columns used in subqueries' conditions to improve performance.

### **Summary**

Subqueries are powerful tools in SQL for performing complex data manipulations and retrieving specific information. They can be used in various parts of SQL queries, including SELECT, WHERE, FROM, and even in INSERT, UPDATE, and DELETE statements. Understanding how to use subqueries effectively can greatly enhance your SQL querying capabilities.

Reference <https://www.mysqltutorial.org/mysql-basics/mysql-subquery/>

<https://www.guru99.com/sub-queries.html>

<https://mariadb.com/kb/en/subquery-optimizations-map/>

<https://www.scaler.com/topics/sql/correlated-subquery/>