MySQLTUTORIAL

An Essential Guide to MySQL Derived Tables

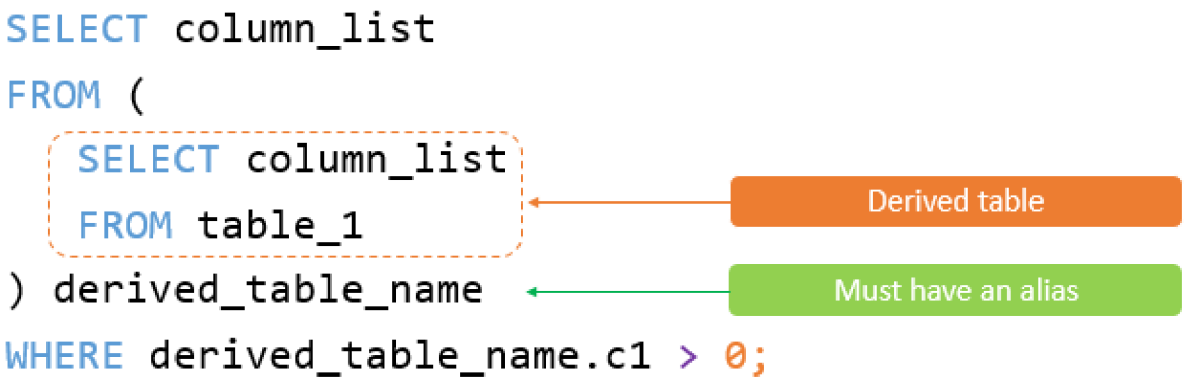
Summary: in this tutorial, you will learn about the MySQL derived tables and how to use them to simplify complex queries.

Introduction to MySQL derived tables

A derived table is a virtual table returned from a `SELECT` (https://www.mysqltutorial.org/mysql-select-statement-query-data.aspx) statement. A derived table is similar to a temporary table (https://www.mysqltutorial.org/mysql-temporary-table/) , but using a derived table in the `SELECT` statement is much simpler than a temporary table because it does not require creating the temporary table.

The term derived table and subquery (https://www.mysqltutorial.org/mysql-subquery/) is often used interchangeably. When a stand-alone subquery is used in the `FROM` clause of a `SELECT` statement, it is also called a derived table.

The following illustrates a query that uses a derived table:



Note that a stand-alone subquery is a subquery that can execute independently of the outer query.

Unlike a subquery, a derived table must have an alias (https://www.mysqltutorial.org/mysql-alias/) so that you can reference its name later in the query. If a derived table does not have an alias, MySQL will issue the following error:

```
Every derived table must have its own alias.
```

The following illustrates the syntax of a query that uses a derived table:

```
SELECT
    select_list
FROM
    (SELECT
        select_list
    FROM
        table_1) derived_table_name
WHERE
    derived_table_name.c1 > 0;
```

A simple MySQL derived table example

The following query gets the top five products by sales revenue in 2003 from the `orders` and `orderdetails` tables in the sample database (https://www.mysqltutorial.org/mysql-sample-database.aspx) :

```
SELECT
    productCode,
    ROUND(SUM(quantityOrdered * priceEach)) sales
FROM
    orderdetails
        INNER JOIN
    orders USING (orderNumber)
WHERE
    YEAR(shippedDate) = 2003
```

```
GROUP BY productCode
ORDER BY sales DESC
LIMIT 5;
```

You can use the result of this query as a derived table and join it with the `products` table as follows:

```
SELECT
    productName, sales
FROM
    (SELECT
        productCode,
        ROUND(SUM(quantityOrdered * priceEach)) sales
    FROM
        orderdetails
    INNER JOIN orders USING (orderNumber)
    WHERE
        YEAR(shippedDate) = 2003
    GROUP BY productCode
    ORDER BY sales DESC
    LIMIT 5) top5products2003
INNER JOIN
    products USING (productCode);
```

The following shows the output of the query above:

In this example:

1. First, the subquery is executed to create a result set or derived table.
2. Then, the outer query is executed that joined the `top5product2003` derived table with the `products` table using the `productCode` column.

A more complex MySQL derived table example

Suppose you have to classify the customers who bought products in 2003 into 3 groups: `platinum`, `gold`, and `silver`. And you need to know the number of customers in each group with the following conditions:

- Platinum customers who have orders with the volume greater than 100K.
- Gold customers who have orders with the volume between 10K and 100K.
- Silver customers who have orders with the volume less than 10K.

To form this query, you first need to put each customer into the respective group using `CASE` (https://www.mysqltutorial.org/mysql-case-function/) expression and `GROUP BY` (https://www.mysqltutorial.org/mysql-group-by.aspx) clause as follows:

```
SELECT
    customerNumber,
    ROUND(SUM(quantityOrdered * priceEach)) sales,
    (CASE
        WHEN SUM(quantityOrdered * priceEach) < 10000 THEN 'Silver'
        WHEN SUM(quantityOrdered * priceEach) BETWEEN 10000 AND 100000 THEN 'Gold'
        WHEN SUM(quantityOrdered * priceEach) > 100000 THEN 'Platinum'
    END) customerGroup
FROM
```

```
    orderdetails
        INNER JOIN
    orders USING (orderNumber)
WHERE
    YEAR(shippedDate) = 2003
GROUP BY customerNumber;
```

The following is the output of the query:

Then, you can use this query as the derived table and perform grouping as follows:

```
SELECT
    customerGroup,
    COUNT(cg.customerGroup) AS groupCount
FROM
    (SELECT
        customerNumber,
            ROUND(SUM(quantityOrdered * priceEach)) sales,
            (CASE
                WHEN SUM(quantityOrdered * priceEach) < 10000 THEN 'Silver'
                WHEN SUM(quantityOrdered * priceEach) BETWEEN 10000 AND 100000 THEN 'Gold'
                WHEN SUM(quantityOrdered * priceEach) > 100000 THEN 'Platinum'
            END) customerGroup
    FROM
        orderdetails
    INNER JOIN orders USING (orderNumber)
    WHERE
        YEAR(shippedDate) = 2003
```

```
    GROUP BY customerNumber) cg
GROUP BY cg.customerGroup;
```

The query returns the customer groups and the number of customers in each.

In this tutorial, you have learned how to use the MySQL derived tables which are subqueries in the `FROM` clause to simplify complex queries.