MySQLTUTORIAL

MySQL Alias

Summary: in this tutorial, you will learn how to use **MySQL alias** to improve the readability of the queries.

MySQL supports two kinds of aliases: column alias and table alias.

MySQL alias for columns

Sometimes, column names are so technical that make the query's output very difficult to understand. To give a column a descriptive name, you can use a column alias.

The following statement illustrates how to use the column alias:

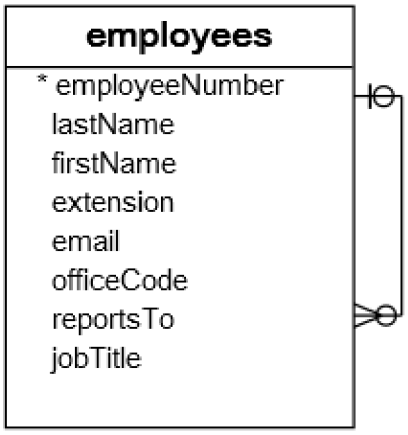
```
SELECT
   [column_1 | expression] AS descriptive_name
FROM table_name;
```

To assign an alias to a column, you use the `AS` keyword followed by the alias. If the alias contains spaces, you must quote it as the following:

```
SELECT
   [column_1 | expression] AS `descriptive name`
FROM
   table_name;
```

Because the `AS` keyword is optional, you can omit it in the statement. Note that you can also give an expression an alias.

Let's look at the `employees` table in the sample database. (https://www.mysqltutorial.org/mysql-sample-database.aspx)



The following query selects the first names and last names of employees. It uses the `CONCAT_WS()` function to concatenate (https://www.mysqltutorial.org/sql-concat-in-mysql.aspx) first name and last name into full name.

```
SELECT
    CONCAT_WS(', ', lastName, firstname)
FROM
    employees;
```

Try It Out ›

The column heading is quite difficult to read. To solve this, you can assign the column heading of the output a column alias as shown in the following query:

```
SELECT
   CONCAT_WS(', ', lastName, firstname) AS `Full name`
```

```
FROM
    employees;
```

Try It Out ›

In MySQL, you can use the column alias in the `ORDER BY (https://www.mysqltutorial.org/mysql-order-by/)`, `GROUP BY (https://www.mysqltutorial.org/mysql-group-by.aspx)` and `HAVING (https://www.mysqltutorial.org/mysql-having.aspx)` clauses to refer to the column.

The following query uses the column alias in the `ORDER BY` clause to sort the employee's full names alphabetically:

```
SELECT
        CONCAT_WS(', ', lastName, firstname) `Full name`
FROM
        employees
ORDER BY
        `Full name`;
```

Try It Out ›

The following statement selects the orders whose total amount is greater than 60000. It uses column aliases in `GROUP BY` and `HAVING` clauses.

```
SELECT
        orderNumber `Order no.`,
        SUM(priceEach * quantityOrdered) total
FROM
        orderDetails
GROUP BY
        `Order no.`
HAVING
        total > 60000;
```

Try It Out ›

> Notice that you cannot use a column alias in the `WHERE (https://www.mysqltutorial.org/mysql-where/)` clause. The reason is that when MySQL evaluates the `WHERE` clause, the values of columns specified in the `SELECT (https://www.mysqltutorial.org/mysql-select-statement-query-data.aspx)` clause are not be evaluated yet.

MySQL alias for tables

You can use an alias to give a table a different name. You assign a table an alias by using the `AS` keyword as the following syntax:

```
table_name AS table_alias
```

The alias for a table is called a table alias. Like the column alias, the `AS` keyword is optional so you can omit it.

This query shows how to assign the `employees` table alias as e:

```
SELECT * FROM employees e;
```

Once a table is assigned an alias, you can refer to the table columns using the following syntax:

```
table_alias.column_name
```

For example:

```
SELECT
    e.firstName,
    e.lastName
FROM
    employees e
ORDER BY e.firstName;
```

The table aliases are often used in the statement that contains `INNER JOIN` (https://www.mysqltutorial.org/mysql-inner-join.aspx) , `LEFT JOIN` (https://www.mysqltutorial.org/mysql-left-join.aspx) , `RIGHT JOIN` (https://www.mysqltutorial.org/mysql-right-join/) clauses and in subqueries (https://www.mysqltutorial.org/mysql-subquery/) .

Let's look at the `customers` and `orders` tables:

Both tables have the same column name: `customerNumber`.Without using the table alias to qualify the `customerNumber` column, you will get an error message like:

```
Error Code: 1052. Column 'customerNumber' in on clause is ambiguous
```

To avoid this error, you use a table alias to qualify the `customerNumber` column:

```
SELECT
        customerName,
        COUNT(o.orderNumber) total
FROM
        customers c
INNER JOIN orders o ON c.customerNumber = o.customerNumber
GROUP BY
        customerName
ORDER BY
        total DESC;
```

Try It Out ›

The query above selects the customer name and the number of orders from the `customers` and `orders` tables. It uses `c` as a table alias for the `customers` table and `o` as a table alias for the `orders` table. The columns in the `customers` and `orders` tables are referred to via the table aliases.

If you do not use the alias in the query above, you have to use the table name to refer to its columns, which makes the query lengthy and less readable as the following:

```
SELECT
        customers.customerName,
        COUNT(orders.orderNumber) total
FROM
        customers
INNER JOIN orders ON customers.customerNumber = orders.customerNumber
GROUP BY
        customerName
ORDER BY
        total DESC
```

Try It Out ›

In this tutorial, you have learned how to use MySQL aliases including column and table aliases.