MySQLTUTORIAL

MySQL HAVING

Summary: in this tutorial, you will learn how to use MySQL `HAVING` clause to specify a filter condition for groups of rows or aggregates.

Introduction to MySQL HAVING clause

The `HAVING` clause is used in the `SELECT (https://www.mysqltutorial.org/mysql-select-statement-query-data.aspx)` statement to specify filter conditions for a group of rows or aggregates.

The `HAVING` clause is often used with the `GROUP BY (https://www.mysqltutorial.org/mysql-group-by.aspx)` clause to filter groups based on a specified condition. If you omit the `GROUP BY` clause, the `HAVING` clause behaves like the `WHERE (https://www.mysqltutorial.org/mysql-where/)` clause.

The following illustrates the syntax of the `HAVING` clause:

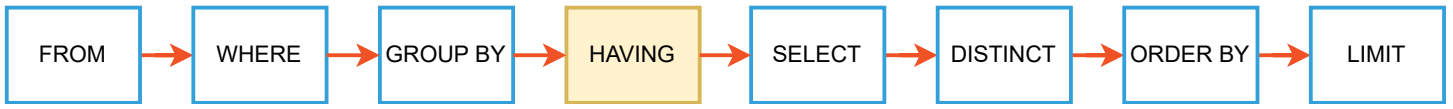
```
SELECT
    select_list
FROM
    table_name
WHERE
    search_condition
GROUP BY
    group_by_expression
HAVING
    group_condition;
```

In this syntax, you specify a condition in the `HAVING` clause.

The HAVING clause evaluates each group returned by the GROUP BY clause. If the result is true, the row is included in the result set.

Notice that the `HAVING` clause applies a filter condition to each group of rows, while the `WHERE` clause applies the filter condition to each individual row.

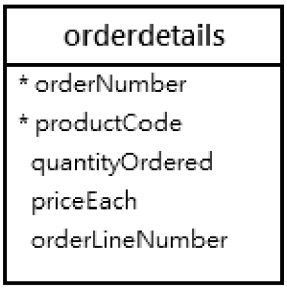
MySQL evaluates the `HAVING` clause after the `FROM`, `WHERE` (https://www.mysqltutorial.org/mysql-where/), `SELECT` and `GROUP BY` (https://www.mysqltutorial.org/mysql-group-by.aspx) clauses and before `ORDER BY` (https://www.mysqltutorial.org/mysql-order-by/), and `LIMIT` (https://www.mysqltutorial.org/mysql-limit.aspx) clauses:



Note that the SQL standard specifies that the `HAVING` is evaluated before `SELECT` clause and after `GROUP BY` clause.

MySQL HAVING clause examples

Let's take some examples of using the `HAVING` clause to see how it works. We'll use the `orderdetails` table in the sample database (https://www.mysqltutorial.org/mysql-sample-database.aspx) for the demonstration.



The following uses the `GROUP BY` clause to get order numbers, the number of items sold per order, and total sales for each from the `orderdetails` table:

```
SELECT
    ordernumber,
    SUM(quantityOrdered) AS itemsCount,
    SUM(priceeach*quantityOrdered) AS total
FROM
    orderdetails
GROUP BY ordernumber;
```

Try It Out ›

ordernumber	itemsCount	total
10100	151	10223.83
10101	142	10549.01
10102	80	5494.78
10103	541	50218.95
10104	443	40206.20
10105	545	53959.21
10106	675	52151.81
10107	229	22292.62

Now, you can find which order has total sales greater than `1000` by using the `HAVING` clause as follows:

```
SELECT
    ordernumber,
    SUM(quantityOrdered) AS itemsCount,
    SUM(priceeach*quantityOrdered) AS total
FROM
    orderdetails
GROUP BY
   ordernumber
HAVING
   total > 1000;
```

Try It Out ›

ordernumber	itemsCount	total
10100	151	10223.83
10101	142	10549.01
10102	80	5494.78
10103	541	50218.95
10104	443	40206.20
10105	545	53959.21
10106	675	52151.81
10107	229	22292.62

It's possible to form a complex condition in the `HAVING` clause using logical operators such as `OR` (https://www.mysqltutorial.org/mysql-or/) and `AND` (https://www.mysqltutorial.org/mysql-and/).

The following example uses the `HAVING` clause to find orders that have total amounts greater than `1000` and contain more than `600` items:

```
SELECT
    ordernumber,
    SUM(quantityOrdered) AS itemsCount,
    SUM(priceeach*quantityOrdered) AS total
FROM
    orderdetails
GROUP BY ordernumber
HAVING
    total > 1000 AND
    itemsCount > 600;
```

Try It Out ›

	ordernumber	itemsCount	total
▶	10106	675	52151.81
	10126	617	57131.92
	10135	607	55601.84
	10165	670	67392.85
	10168	642	50743.65
	10204	619	58793.53
	10207	615	59265.14
	10212	612	59830.55
	10222	717	56822.65

Suppose that you want to find all orders that already shipped and have a total amount greater than 1500, you can join (https://www.mysqltutorial.org/mysql-join/) the `orderdetails` table with the `orders` table using the `INNER JOIN` (https://www.mysqltutorial.org/mysql-inner-join.aspx) clause and apply a condition on `status` column and `total` aggregate as shown in the following query:

```
SELECT
    a.ordernumber,
    status,
    SUM(priceeach*quantityOrdered) total
FROM
    orderdetails a
INNER JOIN orders b
    ON b.ordernumber = a.ordernumber
GROUP BY
    ordernumber,
    status
```

```
HAVING
    status = 'Shipped' AND
    total > 1500;
```

Try It Out ›

	ordernumber	status	total
▶	10100	Shipped	10223.83
	10101	Shipped	10549.01
	10102	Shipped	5494.78
	10103	Shipped	50218.95
	10104	Shipped	40206.20
	10105	Shipped	53959.21
	10106	Shipped	52151.81

The `HAVING` clause is only useful when you use it with the `GROUP BY` clause to generate the output of the high-level reports. For example, you can use the `HAVING` clause to answer the questions like finding the number of orders this month, this quarter, or this year that have a total amount greater than 10K.

Summary

- Use the MySQL `HAVING` clause with the `GROUP BY` clause to specify a filter condition for groups of rows or aggregates.