MySQL SUM

Introduction to the MySQL SUM() function

The SUM() function is an [aggregate function](https://www.mysqltutorial.org/mysql-aggregate-functions.aspx) that allows you to calculate the sum of values in a set. The syntax of the SUM() function is as follows:

SUM(DISTINCT expression)

Code language: SQL (Structured Query Language) (sql)

Here is how the SUM() function works:

* If you use the SUM() function in a [SELECT](https://www.mysqltutorial.org/mysql-select-statement-query-data.aspx) statement that returns no row, the SUM() function returns NULL, not zero.
* The DISTINCT option instructs the SUM() function to calculate the sum of only distinct values in a set.
* The SUM() function ignores the NULL values in the calculation.

MySQL SUM() function illustration

First, [create a new table](https://www.mysqltutorial.org/perl-mysql/perl-mysql-create-table/) named sum\_demo:

**CREATE** **TABLE** sum\_demo (

n INT

);

Code language: SQL (Structured Query Language) (sql)

Then, [insert some rows](https://www.mysqltutorial.org/mysql-insert-statement.aspx) into the sum\_demo table:

**INSERT** **INTO** sum\_demo(n)

**VALUES**(1),(1),(2),(NULL),(3);

Code language: SQL (Structured Query Language) (sql)

Third, use the SUM() function to calculate the total values in the n column:

**SELECT**

**SUM**(n)

**FROM**

sum\_demo;

Code language: SQL (Structured Query Language) (sql)

https://sp.mysqltutorial.org/wp-content/uploads/2019/08/mysql-sum-all-rows-demo.png

As you can see, the SUM() function calculates the total of 1, 1, 2, and 3. And it ignores NULL.

Finally, use the SUM() with the DISTINCT option to calculate the total values in the n column:

**SELECT**

**SUM**(**DISTINCT** n)

**FROM**

sum\_demo;

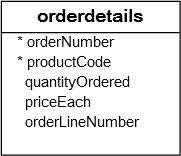
Code language: SQL (Structured Query Language) (sql)

https://sp.mysqltutorial.org/wp-content/uploads/2019/08/mysql-sum-distinct-example.png

In this case, the SUM() with the DISTINCT option only calculates the sum of distinct values which are 1, 2 and 3.

MySQL SUM() function examples

Let’s take a look at the table orderdetails in the [sample database](https://www.mysqltutorial.org/mysql-sample-database.aspx).



1) Simple MySQL SUM() function example

This example uses the SUM() function to get the total number of items of the order details:

**SELECT**

**SUM**(quantityOrdered) SalesQuantity

**FROM**

orderdetails;

Code language: SQL (Structured Query Language) (sql)

https://sp.mysqltutorial.org/wp-content/uploads/2019/08/mysql-sum-example.png

2) MySQL SUM() function with expression example

The following shows the order line items of the order number 10110:

**SELECT**

orderNumber,

quantityOrdered,

priceEach

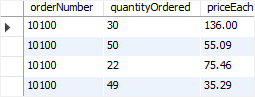
**FROM**

orderdetails

**WHERE**

orderNumber = 10100;

Code language: SQL (Structured Query Language) (sql)



To calculate the total for the order number 10110, you use the SUM() function as follows:

**SELECT**

**SUM**(quantityOrdered \* priceEach) orderTotal

**FROM**

orderdetails

**WHERE**

orderNumber = 10100;

Code language: SQL (Structured Query Language) (sql)

https://sp.mysqltutorial.org/wp-content/uploads/2019/08/mysql-sum-order-total.png

In this tutorial, the SUM() function calculates the total of the following expression of all order line items of the order number 10110:

quantityOrdered \* priceEach

Code language: SQL (Structured Query Language) (sql)

3) MySQL SUM() with the GROUP BY clause example

The SUM() function is often used with the [GROUP BY](https://www.mysqltutorial.org/mysql-group-by.aspx) clause to calculate the sum for each group.

For example, you can calculate the total amount of each order by using the SUM() function with the GROUP BY clause as shown in the following query:

**SELECT**

orderNumber,

**SUM**(quantityOrdered \* priceEach) orderTotal

**FROM**

orderdetails

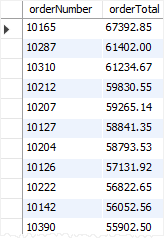
**GROUP** **BY**

orderNumber

**ORDER** **BY**

orderTotal **DESC**;

Code language: SQL (Structured Query Language) (sql)



In this example:

* The GROUP BY clause divides order details into groups grouped by the order number.
* The SUM() function calculates the total of each amount of each order.

4) MySQL SUM() with HAVING clause example

You can use the SUM() function in the [HAVING](https://www.mysqltutorial.org/mysql-having.aspx) clause to filter the group. This example illustrates how to select orders whose order amounts are greater than 60,000.

**SELECT**

orderNumber,

**SUM**(quantityOrdered \* priceEach) orderTotal

**FROM**

orderdetails

**GROUP** **BY**

orderNumber

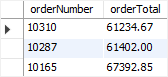
**HAVING**

**SUM**(quantityOrdered \* priceEach) > 60000

**ORDER** **BY**

orderTotal;

Code language: SQL (Structured Query Language) (sql)



5) MySQL SUM() with NULL example

The SUM() function returns NULL if the result set is empty. Sometimes, you may want the SUM() function to return zero instead of NULL.

In this case, you can use the [COALESCE()](https://www.mysqltutorial.org/mysql-coalesce/) function. The COALESCE function accepts two arguments and returns the second argument if the first argument is NULL; otherwise, it returns the first argument.

See the following query:

**SELECT**

**COALESCE**(**SUM**(quantityOrdered \* priceEach), 0) **result**

**FROM**

orderdetails

**WHERE**

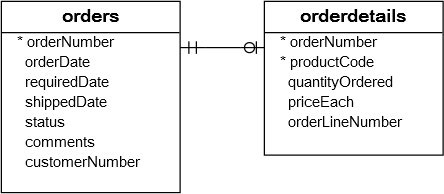
productCode = 'S1\_20';

Code language: SQL (Structured Query Language) (sql)

https://sp.mysqltutorial.org/wp-content/uploads/2019/08/mysql-sum-coalesce.png

6) MySQL SUM() with join example

See the following orders and orderdetails tables:



You can use the SUM() function in a SELECT with [JOIN](https://www.mysqltutorial.org/mysql-join/) clause to calculate the sum of values in a table based on a condition specified by the values in another table.

This statement uses the SUM() function to calculate the total amounts of the canceled orders:

**SELECT**

**SUM**(quantityOrdered \* priceEach) cancelled\_amount

**FROM**

orderdetails

**INNER** **JOIN** orders **USING** (orderNumber)

**WHERE**

**status** = 'Cancelled';

Code language: SQL (Structured Query Language) (sql)

7) MySQL SUM IF example

The following statement uses the SUM() function to calculate the number of items sold for each order status:

**SELECT**

**status**,

**SUM**(quantityOrdered)

**FROM**

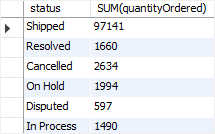
orderdetails

**INNER** **JOIN**

orders **USING** (orderNumber)

**GROUP** **BY** **status**;

Code language: SQL (Structured Query Language) (sql)



If you want to rotate rows to columns, you can use the SUM() fnction with [CASE](https://www.mysqltutorial.org/mysql-case-function/) expression. It is kind of SUMIF logic:

**SELECT**

**SUM**(**CASE**

**WHEN** **status** = 'Shipped' **THEN** quantityOrdered

**END**) qty\_shipped,

**SUM**(**CASE**

**WHEN** **status** = 'Resolved' **THEN** quantityOrdered

**END**) qty\_resolved,

**SUM**(**CASE**

**WHEN** **status** = 'Cancelled' **THEN** quantityOrdered

**END**) qty\_cancelled,

**SUM**(**CASE**

**WHEN** **status** = 'On Hold' **THEN** quantityOrdered

**END**) qty\_on\_hold,

**SUM**(**CASE**

**WHEN** **status** = 'Disputed' **THEN** quantityOrdered

**END**) qty\_on\_disputed,

**SUM**(**CASE**

**WHEN** **status** = 'In Process' **THEN** quantityOrdered

**END**) qty\_in\_process

**FROM**

orderdetails

**INNER** **JOIN**

orders **USING** (orderNumber);

Code language: SQL (Structured Query Language) (sql)

https://sp.mysqltutorial.org/wp-content/uploads/2019/08/mysql-sumif-example.png