2021-07-23T18:00:00+02:00 | Dorota Wdzięczna

SQL SUM() Function Explained with 5 Practical Examples

Aggregate functions are an important part of SQL knowledge – and there's no better place to start learning them than with the SUM() function. In this article, you can expand or refresh your SQL with 5 practical examples of SUM().

SQL allows us to do more than select values or expressions from tables. Most operations on relational databases use aggregate functions like `SUM()` to do computations on data.

Aggregate functions allow you to calculate various statistics for a group of rows. You can find more info about them in the article A Beginner's Guide to SQL Aggregate Functions.

The `SUM()` function sums up all the values in a given column or the values returned by an expression (which could be made up of numbers, column values, or both). It's a good introduction to SQL's aggregate functions, so let's dive right in!

Improve your SQL skills with our SQL Practice track. Five interactive SQL courses with over 600 exercises – and we keep adding more!

Getting Started: SUM Function Syntax

Here is the syntax of the SUM function:

```
SUM([DISTINCT] column_or_expression)
```

As you see, this function requires only one argument: a column name or an expression. The values provided in the argument are summed up and returned by the `SUM()` function. The `DISTINCT` keyword is optional; when it's used, `SUM()` only adds non-repeating values. I will explain this issue more in the next section.

First, let's get to know the data on which we will build our SQL `SUM()` queries.

The **`product`** table stores data we'll use to calculate the sum of given values. Here's a sample of its data:

id	name	price	quantity	category
1	dress	120	1	clothing
2	T-shirt	55	2	clothing
3	jacket	600	2	clothing
4	netbook	2100	NULL	IT
5	mouse	24	NULL	IT
6	bricks	68	1	toys
7	wood mobile	22	1	toys
8	teddy bear	70	NULL	toys

Example 1: Using SUM() with One Column

If you want to sum values stored in one column, use `SUM()` with that column's name as the argument. Look at the example below:

```
SELECT
  SUM(quantity) AS sum_quantity
FROM product;
```

In this query, we use `SUM()` alone in the SELECT statement. The `SUM()` function adds all values from the `quantity` column and returns the total as the result of the function. The name of the new result column (i.e. the alias) is `sum_quantity`.

Here's the result:

sum_quantity
7

As you see, the sum of values in the `quantity` column in the table **`product`** is 7. If one of the rows has a `NULL` in this column, `SUM()` omits this row from the calculation. Note that there is only one row in the result: all the individual rows were “collapsed” by `SUM()` – their details are not available in the result.

Example 2: Using SUM() with an Expression

Next, we’ll consider an example that uses an expression as the `SUM()` argument. This is the query:

```
SELECT
  SUM(quantity*price) AS total_value
FROM product;
```

And the result returned by it is:

total_value
1520

In this case, the argument in the `SUM()` function is an expression that calculates the total value for each product: the quantity of the product (the value in the column `quantity`) multiplied by its price (the value in the column price). The total values calculated for each product are summed and the grand total of their values is returned; the `total_sum` is 1520, as you can see in the result.

Example 3: Using SUM() with GROUP BY

Usually, you use the SUM function with the `GROUP BY` clause. With `GROUP BY`, the summed values are computed for a group of rows. If you’re not familiar with `GROUP BY`, I suggest reading Using GROUP BY in SQL or How Does SQL GROUP BY Work? before proceeding with this example.

In the query below, you can see how many products are in each category:

```
SELECT
  category,
  SUM(quantity) AS total_quantity
FROM product
GROUP BY category;
```

And the result returned by this query is:

category	total_quantity
clothing	5
IT	NULL
toys	2

In this case, the category column is in the `SELECT` because we want to see the category for which the sum is calculated. Next is the `SUM()` function, which sums up the quantity values. Adding the `GROUP BY` clause means that products with the same value in the category column are put into one group; the sum is calculated for each group separately. Finally, remember that the `GROUP BY` clause must always come after `FROM`.

Notice that for the IT category the calculated sum is `NULL`. This is because all the rows that have an 'IT' value in the category column have a NULL in the `quantity` column. The 'toys' category has one product with a NULL in the quantity column, so the remaining values in this category are summed.

Of course, you can also group records and calculate sums on more than one column.

Example 4: Using SUM() with DISTINCT

The SQL `SUM()` function can also sum up unique (i.e. non-repeating) values only. To do this, use the `SUM` function with the `DISTINCT` keyword in the argument. For example:

```
SELECT
  category,
  SUM(DISTINCT quantity)
FROM product
GROUP BY category;
```

And the result returned by it:

category	total_quantity
clothing	3
IT	NULL

category	total_quantity
toys	2

If you look at the database, you'll see that there are five items of clothing in stock:

name	quantity	category
dress	1	clothing
T-shirt	2	clothing
jacket	2	clothing

However, the query returns a 3 for the total quantity of clothing in stock. Why? Because '2' appears twice in the quantity column for items in the clothing category. When you use `DISTINCT`, any repeated values in the column are ignored.

Note that the `DISTINCT` keyword comes first in the function argument, followed by the column name or the expression.

Example 5: Using SUM() with HAVING

You can use `SUM()` in the `HAVING` clause as part of the filter criteria. For example, in the query below we only want rows with above a certain quantity to appear in the result:

```
SELECT
  category,
  SUM(quantity) AS total_quantity
FROM product
GROUP BY category
HAVING SUM(quantity)>2;
```

And the result:

category	total_quantity
clothing	5

Only one category was returned by the query: clothing. Why? Because the sum of the quantity of products must be greater than 2 to be included in the result. The clothing column (with a `total_quantity` of 5) meets this criteria. The other categories are not displayed because their total quantities are less than 2.

Already have a good idea of the basics? LearnSQL.com has a fantastic SQL Practice track that will get you up to speed with key SQL coding skills. There's nothing more effective than learning while doing, and this course will help you get to that point quickly and painlessly.

Want to Test Your SQL SUM() Function Skill?

In this article, you've learned how the SQL function `SUM()` works. You can use it to add all the values in one column across all rows in a table, to total the results of an expression that uses more than one column, and to sum up values for a group of rows. You can also use SUM() inside the HAVING clause to filter data according to the summed values.

Do you want to learn more about `SUM()`? The article How to Use CASE WHEN with SUM() in SQL explains how to use `SUM()` with the `CASE WHEN` expression. Expand your knowledge! And if you want to test your SQL skills, our SQL Practice Set is a good option.

Keep learning and keep practicing SQL!

Viewed using Just Read