2021-08-31T18:00:00+02:00 | Dorota Wdzięczna

SQL MIN and MAX Functions Explained in 6 Examples

What are the SQL MIN() and MAX() functions? When should you use them as aggregate functions, and when should you use them with window functions? We explain using practical examples.

It's difficult to imagine SQL without the MIN() and MAX() aggregate functions. They are very useful among many data professionals because they allow us to resolve a lot of business problems. In this article, I will explain what each function does and discuss several use cases.

Let's start.

What Is the MIN() Function in SQL?

The SQL standard is supported by most relational database engines and includes several aggregate functions. These aggregate functions compute calculations on numerical data. A Beginner's Guide to SQL Aggregate Functions discusses all of them in detail; this article focuses on only two of these functions: `MIN()` and `MAX()`.

First, let's talk about the `MIN()` function. It returns the smallest value in a set of values. The values can come from a numeric column or as the result of an expression returning a numeric value. (Note: The column can come from a table or a view.) It is a scalar function that returns one numeric value.

The syntax of the `MIN()` function is presented below:

```
MIN(column_or_expression)
```

As you see, this function takes one argument – a column or an expression with numeric values. It's important that the values are numeric, because the function finds the minimum value among them. For example, from this set of values ...

```
4, 5, 8, 2, 14
```

... MIN() returns 2.

What Is the MAX() Function in SQL?

Like `MIN()`, `MAX()` is an aggregate function that returns a numeric value from a set. The difference is that it returns the largest (maximum) value. The values can come from a column or as the result of an expression that returns a numeric value or values. It is a scalar function that returns one value.

Here is the syntax of the `MAX()` function:

```
MAX(column_or_expression)
```

From this set of values ...

```
4, 5, 8, 2, 14
```

... `MAX()` returns 14.

The Sample Data

Before we will start to analyze `MIN()` and `MAX()` use cases, let's look at the data we'll be using in the table **`cosmetics`**:

id	name	category	delivered_year	price
1	shampoo	hair	2020	11
2	night cream	face	2020	12
3	cleanser	face	2020	5
4	brush	hair	2020	3
5	mascara	eye	2021	8
6	eyebrow	eye	2021	15
7	conditioner	hair	2021	21
8	spray	hair	2021	13

id	name	category	delivered_year	price
9	eyeliner	eye	2021	20
10	tonic	face	2020	21
11	face mask	face	2021	10
12	eye cream	eye	2021	22
13	hair mask	hair	2020	3

This table stores information about cosmetic products in three categories (the column `category`): hair, face, and eye. The names of the products are in the column name. The last two columns present data about when the product was delivered (`delivered_year`) and its `price`. There are only two years (2020 and 2021) in the column `delivered_year`.

If you need help with the SQL statements and functions we use in this article, try the SQL Basics Cheat Sheet.

LearnSQL.com is a platform that lets you go through all the SQL topics and pick the right path for you with the guarantee of being able to change your mind at any time without any consequences.

Examples of MIN() and MAX() Use Cases

1. Using `MIN()` or `MAX()` as a Single Column

Aggregate functions like `MIN()` can be used as a single column in the `SELECT` query. For example:

```
SELECT MIN(price) FROM cosmetics;
```

Here is the result:

min
3

After the `SELECT` keyword, we put `MIN()` and the column name (in this case, `price`). Next is the keyword `FROM` and the table name (**`cosmetics`**). The result is the smallest value in the column `price`, which is 3.

You can use `MAX()` in exactly the same way to find the highest product price:

```
SELECT MAX(price) FROM cosmetics;
```

And here is the result:

max
22

The maximum price in the table **`cosmetics`** is 22 (the price of the eye cream).

The article How to Find Minimum Values in Columns gives you more examples about how to find the minimum value.

2. Using MIN() and MAX() with Other Columns

A more advanced use of the `MIN()` or `MAX()` functions is to use them on groups of rows. Here is another query:

```
SELECT category,
  MAX(price) AS max_price
FROM cosmetics
GROUP BY category;
```

It returns the data:

category	max_price
hair	21
face	21
eye	22

First is the `SELECT` keyword with the column name (`category`), then the function `MAX()` with the column `price` as the argument. Then we have `max_price`, the alias of the new column created by `MAX()`.

Next is the keyword `FROM` and the table name. At the end, we have the `GROUP BY` clause with the column `category`. This determines the groups (i.e. products are grouped by category). For each group, we get its largest value – in this case, the highest price among products in that category. In the "eye" category, the highest price is $22; in the "face" and "hair" categories, it's $21.

The article How to Find Maximum Values in Rows gives you more examples of this function.

3. Using MIN() and MAX() in the Same Query

You can use both the `MIN` and `MAX` functions in one `SELECT`. If you use only these functions without any columns, you don't need a `GROUP BY` clause.

Below we have a query that implements both functions:

```
SELECT MIN(price) AS min_price,
  MAX(price) AS max_price
FROM cosmetics;
```

This query returns the data:

min_price	max_price
3	22

In the `SELECT`, we have the `MIN()` function with the `price` column argument followed by `MAX()` with the same argument and their respective aliases. This returns the minimum price ($3) and maximum price ($22) for all products in the table.

Of course, you can do the same by calculating the minimum and maximum price for each category. However, here you'll need to use GROUP BY:

```
SELECT category,
  MIN(price) AS min_price,
  MAX(price) AS max_price
FROM cosmetics
GROUP BY category;
```

Here is the result set:

category	min_price	max_price
hair	3	21
face	5	21

category	min_price	max_price
eye	8	22

In this case, the minimum price in the "hair" category is $3 and the maximum price is $21, while in the "eye" category the lowest value is $8 and the highest is $22.

You can learn more about finding minimum and maximum values in the articles How to Find the Minimum Value of a Column in SQL and How to Find the Maximum Value of a Numeric Column in SQL.

Want to learn SQL? Check out our SQL From A to Z track!

4. Using MIN() or MAX() In the HAVING Clause

Another way to use `MIN()` or `MAX()` is filtering rows according to the value returned by this function – i.e. in the `HAVING` clause.

The query below implements `MIN()` in `HAVING`:

```
SELECT category,
  MAX(price) AS max_price
FROM cosmetics
GROUP BY category
HAVING MIN(price)>4;
```

The result set:

category	max_price
face	21
eye	22

In the `SELECT`, we have the column `category`. Next is the `MAX()` function with `price` as the argument. For each category, we will calculate the maximum price among products belonging to this category. After `FROM` **`cosmetics`** comes the `GROUP BY` clause with the column `category`.

At the end of the query is the `HAVING` clause with the `MIN()` function. This will find the lowest price among products in each category; if that price is less than 4, it will not be included in the results. Because the minimum price in the “hair” category is $3, it was not displayed in the result set.

Notice that using either function in `HAVING` doesn’t require using it (or any aggregate function) in `SELECT`.

5. Using MIN() or MAX() In a Simple Window Function

The most advanced use of `MIN()` or `MAX()` is in a window function. If you aren’t familiar with window functions, read the article SQL Window Function Examples with Explanations.

Here is a simple window function that uses the `MIN()` function:

```
SELECT name, price, category, delivered_year,
  MIN(price) OVER (ORDER BY category DESC) AS min_price
FROM cosmetics;
```

The first thing you can notice is the list of `SELECT` columns: `name`, `price`, `category`, `delivered_year`. Next is the `MIN(price)` aggregate function, which finds the lowest value in the `price` column. `OVER` is what makes this a window function; it defines the window. or the set of rows within the query result set. This allows us to calculate an aggregate value for each row in the window. Here, `OVER` is paired with `ORDER BY` category `DESC` (i.e. in descending order); thus, the minimum price is always $3, because the lowest price in the “hair” category is $3, which is lower than the next category minimum of $5.

Here is the result:

name	price	category	delivered_year	min_price
hair mask	3	hair	2020	3
brush	3	hair	2020	3
conditioner	21	hair	2021	3
spray	13	hair	2021	3
shampoo	11	hair	2020	3
night cream	12	face	2020	3
cleanser	5	face	2020	3
tonic	21	face	2020	3
face mask	10	face	2021	3

name	price	category	delivered_year	min_price
mascara	8	eye	2021	3
eye cream	22	eye	2021	3
eyeliner	20	eye	2021	3
eyebrow	15	eye	2021	3

6. Using MIN/MAX In a Partitioned Window Function

Next, let's use `MIN()` or `MAX()` in a window function that has the `PARTITION BY` clause. (If you need a refresher on `PARTITION BY`, take a look at the SQL Window Functions Cheat Sheet.) This is the query:

```
SELECT name, price, category, delivered_year,
  MIN(price) OVER (PARTITION BY delivered_year
              ORDER BY category DESC)
  AS min_price
FROM cosmetics;
```

This query calculates the minimum price for each partition based on the `delivered_year` column and sorts rows by the category.

It returns the following result set:

name	price	category	delivered_year	min_price
shampoo	11	hair	2020	3
brush	3	hair	2020	3
hair mask	3	hair	2020	3
tonic	21	face	2020	3
night cream	12	face	2020	3
cleanser	5	face	2020	3
spray	13	hair	2021	13
conditioner	21	hair	2021	13
face mask	10	face	2021	10
eyeliner	20	eye	2021	8
mascara	8	eye	2021	8
eyebrow	15	eye	2021	8
eye cream	22	eye	2021	8

These results contain the list of cosmetics, including their name, price, category and the delivery year. Each row also contains the minimum price for that delivery year (which is the partition). In each partition, rows are sorted in descending order by category.

In this case, the `OVER` clause contains the `ORDER BY` and `PARTITION BY` clauses. `PARTITION BY delivered_year` denotes that the minimum price is calculated separately for each year. The records in each partition (the year the products were delivered) are sorted according to category in descending order (`ORDER BY` category `DESC`).

Learn SQL by writing SQL! Build your SQL skills (and confidence) by completing the 129 interactive exercises in our SQL Basics course.

What We've Learned About SQL's MIN() and MAX() Functions

The SQL `MIN()` and `MAX()` aggregate functions are very popular. In this article, I've explained how to use them in various cases. The main difference between them is that `MIN()` finds the minimum value in a set of values, while `MAX()` returns the maximum value.

Now that you know how to use MIN() and MAX(), you're ready to:

- Find the largest/smallest values among all given rows or given groups of rows.
- Compare minimum and maximum values with other rows.
- Use these functions on partitions and on simple window functions.

For more information on the differences between aggregate values and window functions, try our article SQL Window Functions vs. SQL Aggregate Functions: Similarities and Differences.

If you're a beginner and you'd like to expand your knowledge of SQL, I recommend LearnSQL.com's SQL Basics course. But if you're already familiar with basic SQL, you might want to take our Window Functions course to level up your skills.