2021-08-19T18:00:00+02:00 | Himanshu Kathuria

The SQL AVG() Function Explained With Examples

We explain the SQL AVG() function with practical examples, covering how and where you can and cannot use it.

The average is probably one of the most widely used metrics to describe some characteristics of a group. It is so versatile and useful that it can describe something about almost anything.

If you like sports, you see things like average runs per game in baseball, average assists per game or per season in basketball, and so on. Every day, you hear statements like “on average, it rains 10 days a month in this city” or “the temperature remains above 70 degrees on average during summer.”

In this article, I will cover the usage of the SQL `AVG()` function with some real-life examples to help you make good use of it in practical situations. Read through!

What Does the SQL AVG() Function Do?

The SQL `AVG()` function is used to find the average of values over records from a table.

To understand this with an example, consider the following table that contains employee records.

`employees`

employeenumber	lastname	skill_level	annual_salary	department
1056	Patterson	10	10000	Finance
1076	Firrelli	10	NULL	Marketing

employeenumber	lastname	skill_level	annual_salary	department
1088	Patterson	60	12500	Finance
1102	Bondur	21	10000	Human Resources
1143	Bow	10	5000	Sales
1165	Jennings	10	5000	Sales
1166	Thompson	10	10000	Marketing

Example 1

Let's say you want to find out the average skill level of the employees. You can use the SQL AVG() function. Here's the query:

```
SELECT AVG(skill_level)
FROM employees;
```

This is the result:

AVG(skill_level)
18.714285714285715

The `AVG()` function takes a column name as its argument (also known as the operand) and then computes the average for all values in the column. So, in this case, our query returns the average of all values in the `skill_level` column.

You may notice the result is displayed with many decimal places. Since you rarely need it that precise, you might want to round this number to the nearest integer. You can use `AVG()` nested in other SQL functions like `ROUND()`, like this:

```
SELECT ROUND(AVG(skill_level))
FROM employees;
```

The result now looks like this:

AVG(skill_level)
19

The function inside the parentheses is evaluated first. So, the query first computes the average then rounds the result to give you the value 19.

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.

Example 2

In the example above, we used `AVG()` in the `SELECT` part of the query. Let's go through some more SQL `AVG()` examples.

Say you want to get a list of the employees whose salaries are above the company average. If you are new to SQL, your first instinct may be to write something like this:

```
SELECT lastname
FROM  employees
WHERE  annual_salary > AVG(annual_salary);
```

When you run it, it outputs something like this:

```
ERROR 1111 (HY000): Invalid use of group function
```

This query throws an error, because you cannot use `AVG()` in a `WHERE` condition. Since `AVG()` is a group function, you can only use it in a `SELECT` statement or in a HAVING clause.

Instead, run the following query:

```
SELECT  lastname, annual_salary
FROM   employees
WHERE  annual_salary > (SELECT AVG(annual_salary)
                        FROM employees);
```

Here's the result:

lastname	annual_salary
Patterson	10000
Patterson	12500
Bondur	10000
Thompson	10000

The query inside a main query is called a subquery. Here, we first use a subquery (highlighted below) to get the average value of `annual_salary` from the **`employees`** table. We then compare it with each value in the column to get the result.

```
SELECT lastname, annual_salary
FROM   employees
WHERE  annual_salary > (SELECT AVG(annual_salary)
                   FROM employees);
```

Do you see anything interesting?

There is a record in our table with a NULL value in `annual_salary`, but our query didn't throw an error. This is because the SQL `AVG()` function ignores `NULL`s and simply calculates the average of the other records with numeric values. That is, it ignores the value from the row shown below.

employeenumber	lastname	skill_level	annual_salary	department
1076	Firrelli	10	NULL	Marketing

AVG() With a DISTINCT Clause

Imagine you have some duplicated data in your tables, and you want to ignore duplicate values when calculating the average.

To do this in SQL, you have to use a `DISTINCT` clause. You can see the `annual_salary` column in our **`employees`** table has values that repeat, for instance, 5000 and 10000. If we want SQL to consider these values only once, we use `AVG()` with a `DISTINCT` clause.

This is how the results differ:

Query	Result
SELECT AVG(annual_salary) FROM employees;	8750.00
SELECT AVG(DISTINCT annual_salary) FROM employees;	9166.667

In the first case, the value 10,000 was included three times, and the value 5,000 was included twice. In the second case, these values were counted only once each, hence the difference in the results.

LearnSQL.com is a great place to learn SQL. LearnSQL.com offers 30 interactive courses that range in difficulty from beginner to advanced.

You can choose from a full learning track, mini-tracks to sharpen targeted skills, and individual courses.

Aggregate Functions

`AVG()` belongs to a class of functions known as aggregate functions. An aggregate function returns a single computed result over multiple rows.

These functions are extremely important for analysis. It is often impossible to go through each record to gather insights from a table that may contain millions of rows. So, you aggregate these millions of rows into meaningful groups.

For instance, say you work at Facebook, and you want to find out which age groups spend the most time on your app. In this case, you create groups/buckets of different age groups then find the average time spent for each bucket.

Here are some of the other aggregate functions, each with an example use case:

Aggregate Function	Example Use Case
SUM()	Find the sum of salaries by department.
COUNT()	Find the number of employees in each department.
MAX()	Find the highest salary paid in each department.
MIN()	Find the lowest salary paid in each department.
AVG()	Find the average salary for each department.

If it's not very clear to you right now, don't worry. I have some good SQL `AVG()` examples next.

The SQL AVG() Function With a GROUP BY Clause

To demonstrate how AVG() works as an aggregate function in practice, let me take you through how you can use AVG() with a GROUP BY clause.

The SQL GROUP BY clause is used to group rows together. In most cases, a `GROUP BY` clause has one or more aggregate functions that calculate one or more metrics for the group.

Example 3

Imagine you work as an analyst in the compensation team of a company. You want to find out the average employee salary by department. To do so, you can use a `GROUP BY` clause with `AVG()` like this:

```
SELECT   department,
           AVG(annual_salary)
FROM     employees
GROUP BY  department;
```

Here's the result of this query:

department	AVG(annual_salary)
Finance	11250.00
Marketing	10000.00
Human Resources	10000.00
Sales	5000.00

It groups together all the records by department then computes the average `annual_salary` for each department.

Example 4

If you work in a large company with many departments, you may want to focus on the departments whose average salary is higher than a specific value. To do this, you need to filter out some groups from the result of your query.

Let's say you want to find the departments whose average salaries exceed 10000. Your query looks like this:

```
SELECT   department,
           AVG(annual_salary)
FROM     employees
GROUP BY  department
HAVING   AVG(annual_salary) > 10000;
```

Here's the result of this query:

department	AVG(annual_salary)
Finance	11250.00

Since only the Finance department has the average salary greater than 10,000, it is the only row returned.

For more on how to filter records with aggregate functions, read this article.

You don't need to have `AVG()` in the `SELECT` statement to use it in a `HAVING` clause. For example, the following query gives you just the name of the department in the result:

```
SELECT    department
FROM     employees
GROUP BY   department
HAVING    AVG(annual_salary) > 10000;
```

And the result:

department
Finance

The SQL AVG() Function With a CASE Statement

You can also use `AVG()` with a `CASE` statement. If you're not familiar with `CASE`, take a look at this article.

Example 5

Let's say you want to display "High" as the category when the average salary is greater than 7,000, and "Low" if it is equal or less. This is what the query looks like:

```
SELECT department,
       CASE WHEN AVG(annual_salary) > 7000 THEN 'High'
       ELSE 'Low' END as category
FROM employees
GROUP BY department;
```

The result of this query:

department	category
Finance	High
Marketing	High
Human Resource	High
Sales	Low

The average is computed for each department then compared with 7,000. If the average salary of a department is higher than 7,000, then `'High'` is returned as its `category`.

When Not to Use AVG()

Although useful, the average has limitations as a metric. This is especially true when the values in your data set are very skewed.

For example, say you run a YouTube channel, and you have uploaded 20 videos so far. One video has hit a million views, but the rest has yet to see any traction. While it is technically possible to compute the average, in reality, only one video contributes to the average of 200,000.

When values are very skewed, the median is often a better metric. The median is the value at the 50th percentile of a series; that is, it is the point at which half the values in the series/group are below and half are above.

Ready to Use the SQL AVG() Function in Your Queries?

By now, you should have a good understanding of the SQL `AVG()` function and the various ways to use it. If you are just starting out and want a career that deals with data, SQL is a must-learn tool.

Are you an SQL purist who acknowledges only standard SQL? Learn Standard SQL Functions with our course!

To learn more about standard functions like `AVG()`, I recommend this course by LearnSQL.com. The course is wonderfully designed and reinforces learning throughout to help you retain what you learn. If you are super enthusiastic to learn SQL end to end, then check out the SQL A to Z track.

The keys to becoming skilled at SQL are practicing and always staying updated with the developments and the new features. So, keep reading articles from LearnSQL.com for more and the latest in SQL.

Happy learning!
