AVG to Average Values

by Sophia Tutorial

WHAT'S COVERED

This tutorial explores using the AVG function to average values of a given column within a table in two parts:

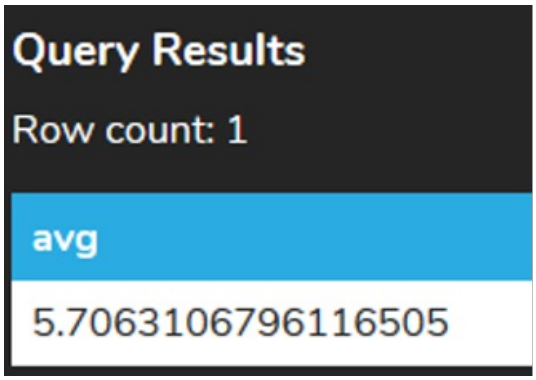
1. Introduction to the AVG Function
2. Dealing with 0 and NULL

1. Introduction to the AVG Function

The AVG function is used to calculate the average of a set of values. This is one of the most commonly used aggregate functions. If we wanted to find the average total from the invoice table, we could run something like this:

```
SELECT AVG(total)
FROM invoice;
```

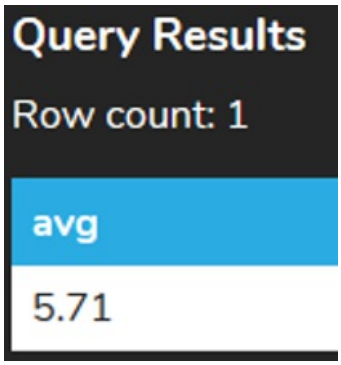
This would return a value like:



Since the value we're interested in is a price, it may make more sense to limit the result to two decimal places. Unique to PostgreSQL, we can cast the result as follows:

```
SELECT AVG(total)::numeric(10,2)
FROM invoice;
```

This will cast it to 10 digits, with two digits after the decimal:



In other databases, you may see the use of functions like ROUND, to round the value, or TRUNC, to truncate the value to a certain position without rounding.

There are times when you may want to find averages of values with criteria being applied. For example, if we were interested in all of the orders that were set by customer_id equal to 2:

```
SELECT *
FROM invoice
WHERE customer_id = 2;
```

We should find seven rows with a total value in each:

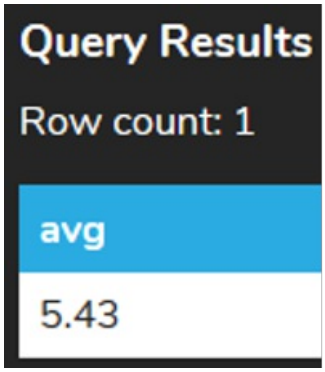
Query Results

Row count: 7

invoice_id	customer_id	invoice_date	billing_address	billing_city	billing_state	billing_country	billing_postal_code	total
1	2	2009-01-01T00:00:00.000Z	Theodor-Heuss-Straße 34	Stuttgart		Germany	70174	2
12	2	2009-02-11T00:00:00.000Z	Theodor-Heuss-Straße 34	Stuttgart		Germany	70174	14
67	2	2009-10-12T00:00:00.000Z	Theodor-Heuss-Straße 34	Stuttgart		Germany	70174	9
196	2	2011-05-19T00:00:00.000Z	Theodor-Heuss-Straße 34	Stuttgart		Germany	70174	2
219	2	2011-08-21T00:00:00.000Z	Theodor-Heuss-Straße 34	Stuttgart		Germany	70174	4
241	2	2011-11-23T00:00:00.000Z	Theodor-Heuss-Straße 34	Stuttgart		Germany	70174	6
293	2	2012-07-13T00:00:00.000Z	Theodor-Heuss-Straße 34	Stuttgart		Germany	70174	1

We can calculate the average to two decimal places, like this:

```
SELECT AVG(total)::numeric(10,2)
FROM invoice
WHERE customer_id = 2;
```



2. Dealing with 0 and NULL

It is important to note that the average only consists of values that are not NULL. For example, consider what

happens if we set one of the seven invoices to have the total as NULL:

```
UPDATE invoice
SET total = NULL
WHERE invoice_id = 1;
```

If we queried for the customer_id = 2, we would see that the total has no value. This is different than having a value of 0.

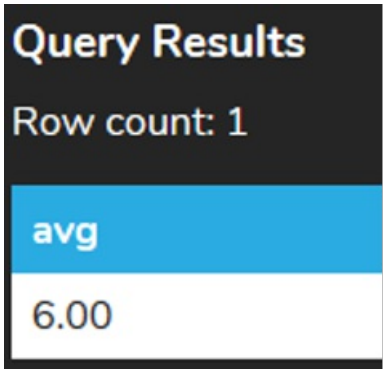
Query Results

Row count: 7

invoice_id	customer_id	invoice_date	billing_address	billing_city	billing_state	billing_country	billing_postal_code	total
12	2	2009-02-11T00:00:00.000Z	Theodor-Heuss-Straße 34	Stuttgart		Germany	70174	14
67	2	2009-10-12T00:00:00.000Z	Theodor-Heuss-Straße 34	Stuttgart		Germany	70174	9
196	2	2011-05-19T00:00:00.000Z	Theodor-Heuss-Straße 34	Stuttgart		Germany	70174	2
219	2	2011-08-21T00:00:00.000Z	Theodor-Heuss-Straße 34	Stuttgart		Germany	70174	4
241	2	2011-11-23T00:00:00.000Z	Theodor-Heuss-Straße 34	Stuttgart		Germany	70174	6
293	2	2012-07-13T00:00:00.000Z	Theodor-Heuss-Straße 34	Stuttgart		Germany	70174	1
1	2	2009-01-01T00:00:00.000Z	Theodor-Heuss-Straße 34	Stuttgart		Germany	70174	

Calculating the average would result in the following:

```
SELECT AVG(total)::numeric(10,2)
FROM invoice
WHERE customer_id = 2;
```

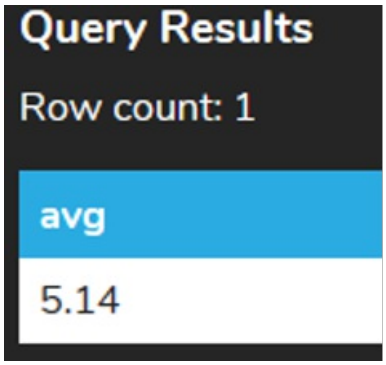


However, if we updated that same invoice to set the total to 0, like this:

```
UPDATE invoice
SET total = 0
WHERE invoice_id = 1;
```

Running the same calculation:

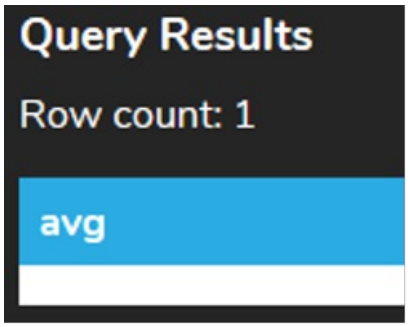
```
SELECT AVG(total)::numeric(10,2)
FROM invoice
WHERE customer_id = 2;
```



In the first case, even though there are seven rows, the null value does not count. The query runs the following calculation: (14 + 9 + 2 + 4 + 6 + 1)/6 = 6.00. In the second case, since the 0 is added instead of the null value, it counts it as part of the calculation: (14 + 9 + 2 + 4 + 6 + 1 + 0)/7 = 5.14

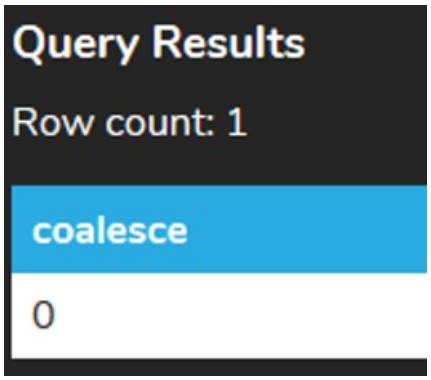
Similar to the SUM function, if we only have NULL values, using the AVG would return NULL.

```
SELECT AVG(total)
FROM invoice
WHERE invoice_id < 1;
```



To have a 0 returned, you would need to use the COALESCE function that would return the second argument if the first argument was NULL.

```
SELECT COALESCE(AVG(total),0)
FROM invoice
WHERE invoice_id < 1;
```



Video Transcription

[MUSIC PLAYING] The aggregate function is probably one of the most useful aggregate functions that's out there. When it comes to utilizing that, we would actually calculate the average of a certain column

based on certain parameters. Let's take a look at an example here looking at invoice, where the customer ID equals two, in which we have seven rows that are actually returned.

If we're trying to calculate the average in this case here, we can make a modification to this to only display the average of the total. One thing you might notice when it calculates the average is that it has a very lengthy number of values after the decimal place. What they can do in this case here, especially when it comes to the average, is to cast it to a different data type. So in this case here, we're going to do is actually cast it to a numeric value, which it currently is already. But what we're going to do is that we're going to set it to have 10 values in which two of them are going to be after decimal place. So when we have that, it'll show as 5.43 instead of a very lengthy value. It is important to note when it comes to the average function that it only calculates it based on actual value. So if there's a null in place, it won't be adding that in place.

So let's go ahead and take a look at that. What we're going to do is we're going to actually set the Invoice ID of 1 to null for the total. And then once we do that, if we try to select it in this case here, what we'll see is that that first invoice is actually going to be set to null. If we try to make a sum of it just to see what value it currently is set to as you'll remember the sum function. It's set to 36. Now note that we have actually seven different rows associated with it, so if it did include that value end with sum, if we try to calculate the average, it should return a different value versus a zero.

However let's see what value we're get in this case here. We're calculating the average it'll show six. So it's only taking 36 divided by six, and then returning that value in this case here. Now let's go ahead and take a look at what the case is if it's actually set to 0. It's going to go ahead update that first one again for the invoice ID with a total of 0 versus a null value. Now, if you calculate the average, it should take 36 and then divided by 7 to return 5.14.

[MUSIC PLAYING]

TRY IT

Your turn! Open the SQL tool by clicking on the LAUNCH DATABASE button below. Then enter in one of the examples above and see how it works. Next, try your own choices for which columns you want the query to provide.

SUMMARY

The AVG function allows us to calculate the average of a set of values.

Source: Authored by Vincent Tran