2021-12-30T18:00:00+02:00
Tihomir Babic

How to Include Zero in a COUNT() Aggregate

Explaining how to include zero (0) counts in your SQL query result.

Here's the problem: you want to count something that doesn't exist, and you want to show your result as zero. How do you do that in SQL?

Using the `COUNT()` aggregate function is a reasonable first step. It will count all the data it finds and return the number of occurrences. But what if there are no occurrences of certain data? It will not show up in the result. However, suppose you want to create a report that will also show data that has zero occurrences. How do you achieve that?

Does this problem seem too vague? I'll show you a concrete example and a solution so you can see what I mean by including zero counts in SQL results. First, you should understand how aggregate functions work; if you're not familiar with them, check out our Beginner's Guide to SQL Aggregate Functions and Overview of SQL Aggregate Functions before continuing.

Example Tables

There are two tables I'll use for my example: `car_buyers` and `service_appointment`.

The table `car_buyers` contains this data:

id	first_name	last_name	nin
1	Steve	Rich	15499863215
2	Susan	Santana	56412846987
3	Mike	Rubens	36545888453
4	Dolores	Raich	63549884422

id	first_name	last_name	nin
5	Ralf	Connery	32145844412

It's a simple list of car buyers for a fictional car sale and repair shop.

The `service_appointment` table contains this data on various car service appointments:

id	appointment_date	description	car_buyer_id
1	2021-01-03	Regular checkup	3
2	2021-01-14	Oil change	3
3	2021-01-14	Regular checkup	5
4	2021-01-15	Regular checkup	1
5	2021-03-08	Lights change	1
6	2021-03-12	Battery replacement	5
7	2021-03-12	AC repair	3
8	2021-03-12	Windshield repair	1
9	2021-06-22	Clutch repair	1
10	2021-08-16	Gearbox change	3
11	2021-11-12	Regular checkup	1

What Do I Want?

Using these tables, I want to get a list of my car buyers along with the number of service appointments they had up until now. I also want that list to include those car buyers who haven't had a service appointment yet, and I want to see a zero by their names. Those with zero appointments could be buyers that just bought a new car, so there hasn't been enough time to need a service appointment.

If I do that, it means I've succeeded in including zero (0) in the `COUNT()` aggregate.

Not Much of a Solution

Intuitively, I might write this code in an attempt to solve the problem:

```
SELECT cb.id,
       cb.first_name,
       cb.last_name,
       COUNT(sa.id) AS no_of_appointments
FROM car_buyers cb
JOIN service_appointment sa
ON cb.id = sa.car_buyer_id
GROUP BY cb.id, cb.first_name, cb.last_name;
```

What did I do here? I've included `id`, `first_name`, and `last_name` from the table **`car_buyers`** in the `SELECT` list and in `GROUP BY`. To count the number of appointments, I've used the `COUNT()` function on the column id from the table **`service_appointment`**.

Do you want to see what result this code returns? I'm sure you do. Take a look:

id	first_name	last_name	no_of_appointments
1	Steve	Rich	5
5	Ralf	Connery	2
3	Mike	Rubens	4

The result shows three buyers with a number of appointments. OK, that's good. However, scroll up a little and you'll see there are five records in the table **`car_buyers`**. I can conclude that the two buyers missing are those who have had zero service appointments up until now. However, I want my code to include them in this result table and I want to see explicitly that they've made zero appointments.

Here's how a little tweaking of the above code can help.

The Real Solution - LEFT JOIN or RIGHT JOIN

Here's the solution that will include zero counts in the result:

```
SELECT cb.id,
       cb.first_name,
       cb.last_name,
       COUNT(sa.id) AS no_of_appointments
FROM car_buyers cb
LEFT JOIN service_appointment sa
ON cb.id = sa.car_buyer_id
GROUP BY cb.id, cb.first_name, cb.last_name;
```

This is the same code as the previous one, but this time I've joined tables using the LEFT JOIN. Here's what it returns:

id	first_name	last_name	no_of_appointments
1	Steve	Rich	5
5	Ralf	Connery	2
2	Susan	Santana	0
4	Dolores	Raich	0
3	Mike	Rubens	4

That's a nice surprise! There are all five car buyers, and the table also shows that Susan Santana and Dolores Raich haven't had any car service appointments.

How does this work? It's all about the JOIN type. Using the suitable JOIN is crucial when you want to include zeros in the `COUNT()` aggregate.

If you know how the LEFT JOIN works, it's easy for you to understand why this code returns the result with zeros. `LEFT JOIN` will return all the buyers from the table `car_buyers`. For those who can be found in that table but couldn't be found in the table `service_appointments` (i.e. they had zero appointments) there will be NULL values in the result. For example, here's how it looks if you just join tables using the `LEFT JOIN` but don't count the number of appointments:

id	first_name	last_name	appointment_date	description
3	Mike	Rubens	2021-01-03	Regular checkup
3	Mike	Rubens	2021-01-14	Oil change
5	Ralf	Connery	2021-01-14	Regular checkup
1	Steve	Rich	2021-01-15	Regular checkup
1	Steve	Rich	2021-03-08	Lights change
5	Ralf	Connery	2021-03-12	Battery replacement
3	Mike	Rubens	2021-03-12	AC repair
1	Steve	Rich	2021-03-12	Windshield repair
1	Steve	Rich	2021-06-22	Clutch repair
3	Mike	Rubens	2021-08-16	Gearbox change
1	Steve	Rich	2021-11-12	Regular checkup
2	Susan	Santana	NULL	NULL

id	first_name	last_name	appointment_date	description
4	Dolores	Raich	NULL	NULL

If you now use the aggregate function `COUNT()`, like in the code above, it will not count the NULL values and the result will be zero. So, it's also important that you know how COUNT() works in various circumstances.

You can get the same result using the `RIGHT JOIN`, too. It's very like the `LEFT JOIN` query shown above, only the table order is reversed:

```
SELECT cb.id,
       cb.first_name,
       cb.last_name,
       COUNT(sa.id) AS no_of_appointments
FROM service_appointment sa
RIGHT JOIN car_buyers cb
ON cb.id = sa.car_buyer_id
GROUP BY cb.id, cb.first_name, cb.last_name;
```

Everything else can stay the same. If you're not familiar with using `JOINs`, I recommend this article explaining all the SQL JOIN types.

Now You Know How to Get Zeros in Your SQL Result

You see that the main point here is not so much using the aggregate function `COUNT()`. The game changer is the kind of `JOIN` you use when you write the query. A simple `JOIN` will not return the desired result; it will show only those buyers that have one or more service appointments. To include zeros resulting from `COUNT()`, you'll have to use `LEFT JOIN` or `RIGHT JOIN`.

Level up your SQL with our interactive SQL JOINs course. Its 93 hands-on exercises let you learn by doing.

You can learn all about JOINs and their differences in our SQL JOINs course, which is a part of the SQL from A to Z track. This will not only allow you to show zero counts, but also help you to become a master at joining tables in SQL.
