GROUP BY to Combine Data

by Sophia

WHAT'S COVERED

In this lesson, you will explore the GROUP BY clause in a SELECT statement to divide rows into groups, in three parts. Specifically, this lesson will cover:

1. Using the GROUP BY Clause
2. GROUP BY Examples
3. Multiple GROUP BY Quantities

1. Using the GROUP BY Clause

To group comparable data, the SQL **GROUP BY** clause can be used along with the SELECT statement. SELECT statements use this GROUP BY clause after the WHERE clause and before the ORDER BY or HAVING clauses. It is often used with aggregate functions, including SUM, COUNT, AVG, MAX, and MIN, to calculate the grouped data.

Imagine you have a table of sales data that includes information about the date, the product, and the amount. To calculate an organization's total sales in a year, the GROUP BY clause can be used to group sales by product. The total sales for each product on each day can also be calculated by grouping the data by product and date.

So far, we have worked with aggregate functions that return a single aggregate value across the result set. Using the GROUP BY clause, we can calculate aggregates within a group of rows. This is much more flexible for data consumers when wanting to compare large data sets looking for similarities in groups of data sets.

The structure of a statement that uses the GROUP BY clause looks like this:

```
SELECT <column1>..., <aggregate function>
FROM <tablename>
GROUP BY <column1>...;
```

The columns to be grouped appear in both the SELECT clause and the GROUP BY clause.

TERM TO KNOW

GROUP BY

A clause that calculates aggregates within a group of rows, with groups created based on values in one or more specified columns.

---

2. GROUP BY Examples

For example, if we wanted to know the number of customers that we have in each country, we can use the GROUP BY clause like this:

```
SELECT country, COUNT(*)
FROM customer
GROUP BY country;
```

Query Results

Row count: 24

country	count
Austria	1
Australia	1
Germany	4
Poland	1
France	5
Argentina	1
United Kingdom	3
Czech Republic	2
Italy	1
Sweden	1
Spain	1
Brazil	5
Ireland	1
India	2
Denmark	1
Chile	1
Hungary	1
Belgium	1
USA	13
Portugal	2

The column must be listed in both the SELECT and GROUP BY clause. Consider if we do not include it in the GROUP BY clause:

```
SELECT country, COUNT(*)
FROM customer;
```

We will get an error, as follows:

Query Results

Query failed because of: error: column "customer.country" must appear in the GROUP BY clause or be used in an aggregate function

Now, consider if we did not have the column in the SELECT clause:

```
SELECT COUNT(*)
FROM customer
GROUP BY country;
```

There will not be an error. But the result set isn't useful at all because it doesn't indicate what the aggregate function is about:

Query Results

Row count: 24

count
1
1
4
1
5
1
3
2
1
1
1

As you see in the result set above, even though we grouped based on country, we need to know which country each count is for.

We can run a similar query to find the total amount of orders by country from the invoice table:

```
SELECT billing_country, SUM(total)
FROM invoice
GROUP BY billing_country;
```

Query Results

Row count: 24

billing_country	sum
Netherlands	41
Australia	38
Argentina	38
Brazil	192
Hungary	46
Spain	38
Ireland	46
Austria	43
Poland	38
Sweden	39
Italy	38
India	76

3. Multiple GROUP BY Quantities

We can also add more complexity by grouping by multiple columns and returning more than one aggregate value. For example, if we wanted to find the total and average of each invoice based on the state and country, we can do the following:

```
SELECT billing_country, billing_state, SUM(total), ROUND(AVG(total),2)
FROM invoice
GROUP BY billing_country, billing_state
ORDER BY billing_country, billing_state;
```

Notice above, we have added the ORDER BY clause so that the results are sorted in an order that makes logical sense:

Query Results

Row count: 42

billing_country	billing_state	sum	round
Argentina		38	5.43
Australia	NSW	38	5.43
Austria		43	6.14
Belgium		38	5.43
Brazil	DF	38	5.43
Brazil	RJ	38	5.43
Brazil	SP	116	5.52
Canada	AB	38	5.43
Canada	BC	39	5.57
Canada	MB	38	5.43
Canada	NS	38	5.43
Canada	NT	38	5.43
Canada	ON	76	5.43
Canada	QC	40	5.71

The possibilities are endless for us to group the data and run aggregate functions on those subsets of data.

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

In this lesson, you learned that in PostgreSQL, the **GROUP BY clause** groups rows based on one or more columns. By aggregating data and calculating groups of rows, you are able to perform calculations on data. You learned about specific **GROUP BY examples** which included partitioning the data using the GROUP BY clause based on unique combinations of values in the specified columns. Finally, you learned about using **multiple GROUP BY quantities** to manipulate the data to achieve your desired partitioning result.

Source: THIS TUTORIAL WAS AUTHORED BY DR. VINCENT TRAN, PHD (2020) AND Faithe Wempen (2024) FOR SOPHIA LEARNING. PLEASE SEE OUR TERMS OF USE.

TERMS TO KNOW

GROUP BY

A clause that calculates aggregates within a group of rows, with groups created based on values in one or more specified columns.