GROUP BY to Combine Data

by Sophia Tutorial

WHAT'S COVERED

This tutorial explores the GROUP BY clause in a SELECT statement to divide rows into groups in three parts:

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

1. Using the GROUP BY Clause

The GROUP BY clause in the SELECT statement can help to divide the rows that are being returned from the SELECT statement into specific groups. We can then apply an aggregate function to each group. 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.

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

SELECT <column1>..., <aggregate function>
FROM <tablename>
GROUP BY <column1>...;
We have the columns that we want to group in both the SELECT clause and the GROUP BY clause.

2. 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

It is important that the column is listed in both the SELECT and GROUP BY clause. If we do not include it in the GROUP BY clause:

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

We will get an error:

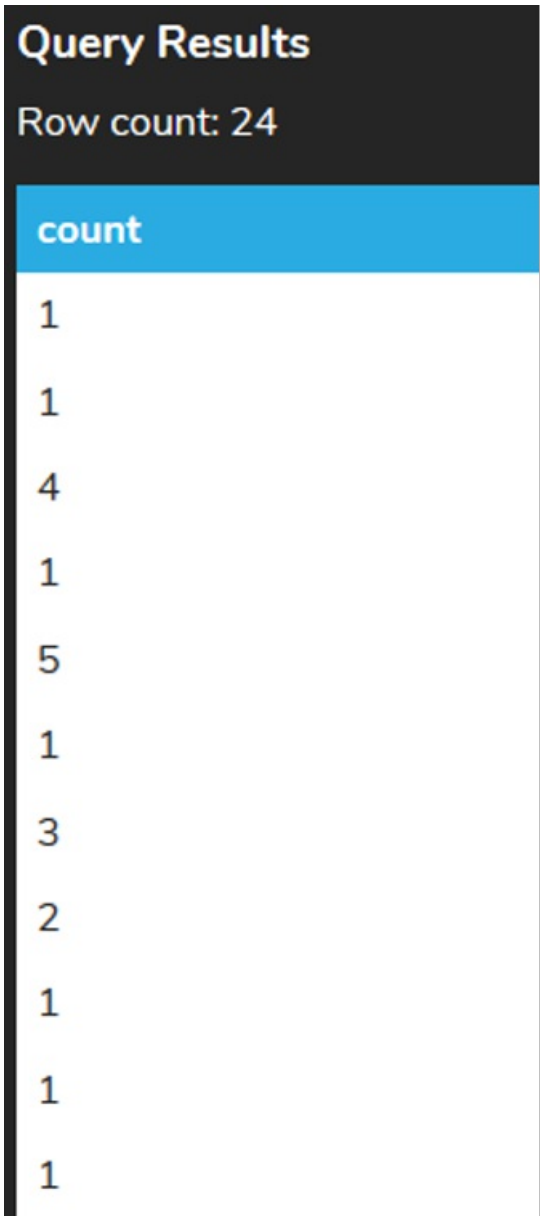
Query Results

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

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 in relation to:



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 do not know which country each of the counts 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.

Video Transcription

[MUSIC PLAYING] The group by clause within a select statement helps us to be able to divide different rows into various groups and then do some aggregate functions associated with those different items. So let's go ahead and take a look at an example in this case here.

So if we want to take a look at the number of customers within different countries, one approach that we could take is looking at the country and then identifying those. However, if we want to be able to break things down a little bit further, what we want to do is utilize the group by clause. So the group by clause allows us to be able to filter based on specific columns. So in this case here, being that we want to utilize the country as a choice, we want to include country.

One thing to note, though, is that when we are having anything that's in the group by clause, it should also appear within select. And only items that are in the group by can appear in the select. Otherwise, it'll generate an error. For example, we'd run it as is, it'll identify that the customer ID. In this case here, is the first column, has to appear in the group by clause. So if we choose country in this case here, it would

be very similar to running a unique clause in this case to display every single one of different countries only one time.

However, the power of the GROUP BY clause is when we're able to utilize different aggregate functions associated with these components. So for example, in this case here, if you want to take a look at the number of customers that are within each country, we can go ahead and make a change with the count in this case here.

So we have the country and then the count of the number of customers within each one. So as we kind of slow down, we'll see that USA has 13, Brazil has five and so forth. We can utilize any option in this case here with as part of the aggregates as long as there are specific items associated with it. As a reminder when it comes to group by clause, you can only have items that are in the group by clause within the select statement. Otherwise everything else has to be aggregate functions.

[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 GROUP BY clause allows us to divide rows into groups and then apply aggregate functions to each of the individual groups.

Source: Authored by Vincent Tran