BETWEEN to Filter Data

by Sophia Tutorial

WHAT'S COVERED

This tutorial explores using the BETWEEN operator within a SELECT statement to filter data based on ranges in three parts:

1. Introduction
2. Using BETWEEN on Dates
3. Adding NOT

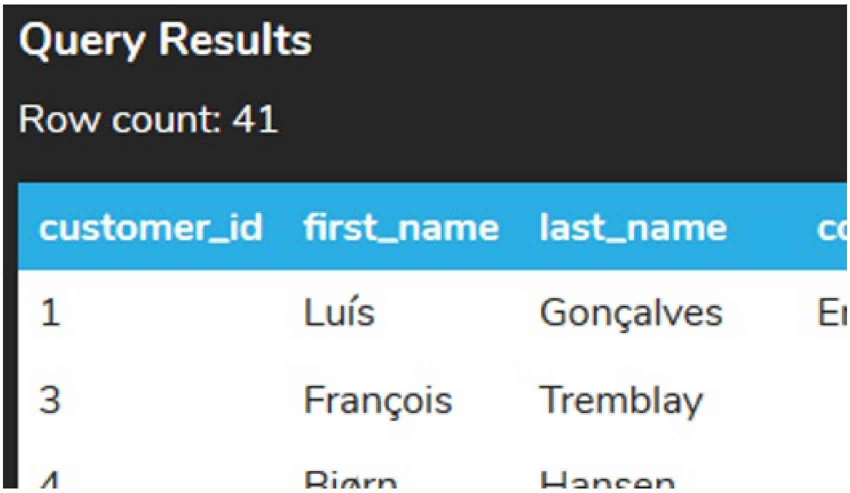
1. Introduction

The BETWEEN is an operator that allows us to check if an attribute is within a range of values. It is similar to using two separate conditions using the logical operators, but helps to simplify things further. The values defined in the BETWEEN operator are inclusive of what is searched for.

It is important to note that we always need to specify the smaller value first. For example, if we have the following statement:

```
SELECT *
FROM customer
WHERE support_rep_id BETWEEN 1 AND 4;
```

This should return a result set with 41 rows.



Query Results

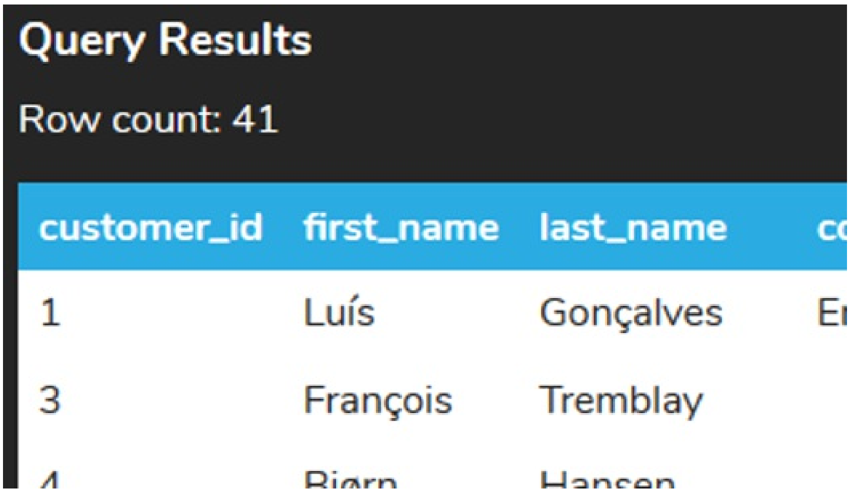
Row count: 41

customer_id	first_name	last_name	co
1	Luís	Gonçalves	Er
3	François	Tremblay	
4	Bjørn	Hansen	

This command can be interpreted as follows:

```
SELECT *
FROM customer
WHERE support_rep_id >= 1
AND support_rep_id <= 4;
```

This command would also return 41 rows.



Query Results

Row count: 41

customer_id	first_name	last_name	co
1	Luís	Gonçalves	E
3	François	Tremblay	
4	Bjørn	Hansen	

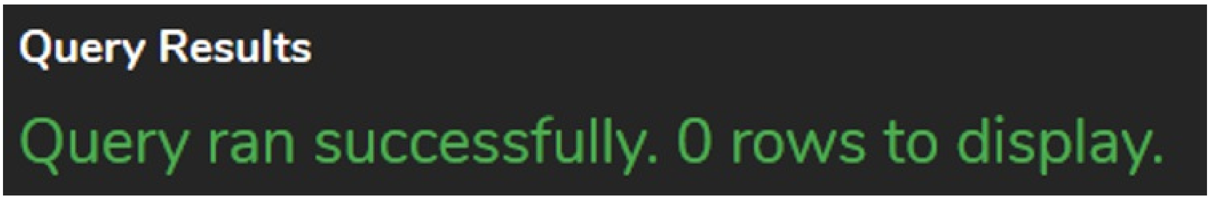
However, if we have the larger number first:

```
SELECT *
FROM customer
WHERE support_rep_id BETWEEN 4 AND 1;
```

It would try to run it as the following:

```
SELECT *
FROM customer
WHERE support_rep_id >= 4
AND support_rep_id <= 1;
```

Of course, this would not work. The support_rep_id could not be greater than or equal to 4 at the same time as the support_rep_id is less than or equal to 1. Therefore, no rows could match that criteria, which is why we would get the following result:



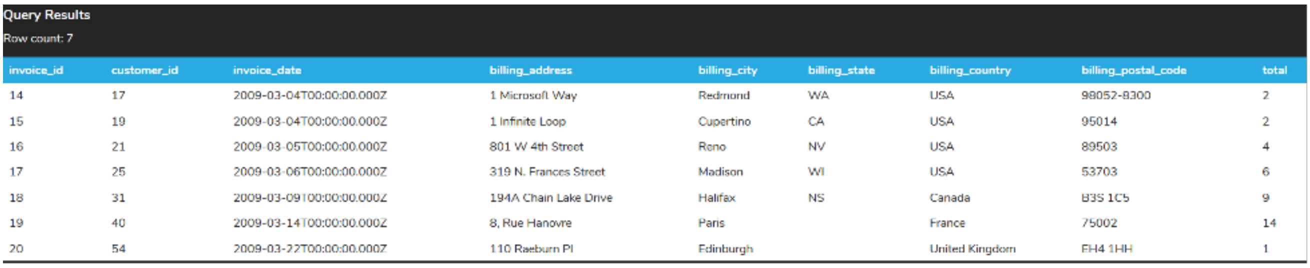
Query Results

Query ran successfully. 0 rows to display.

2. Using BETWEEN on Dates

You can also use BETWEEN for dates when comparing to a range of dates. For example, if we wanted to search for invoices that had the invoice_date in the month of March in 2009, we could do the following:

```
SELECT *
FROM invoice
WHERE invoice_date BETWEEN '2009-03-01' AND '2009-03-31';
```



Query Results

Row count: 7

invoice_id	customer_id	invoice_date	billing_address	billing_city	billing_state	billing_country	billing_postal_code	total
14	17	2009-03-04T00:00:00.000Z	1 Microsoft Way	Redmond	WA	USA	98052-8300	2
15	19	2009-03-04T00:00:00.000Z	1 Infinite Loop	Cupertino	CA	USA	95014	2
16	21	2009-03-05T00:00:00.000Z	801 W 4th Street	Reno	NV	USA	89503	4
17	25	2009-03-06T00:00:00.000Z	319 N. Frances Street	Madison	WI	USA	53703	6
18	31	2009-03-09T00:00:00.000Z	194A Chain Lake Drive	Halifax	NS	Canada	B3S 1C5	9
19	40	2009-03-14T00:00:00.000Z	8, Rue Hanovre	Paris		France	75002	14
20	54	2009-03-22T00:00:00.000Z	110 Raeburn Pl	Edinburgh		United Kingdom	EH4 1HH	1

This would be simpler than having two separate parts:

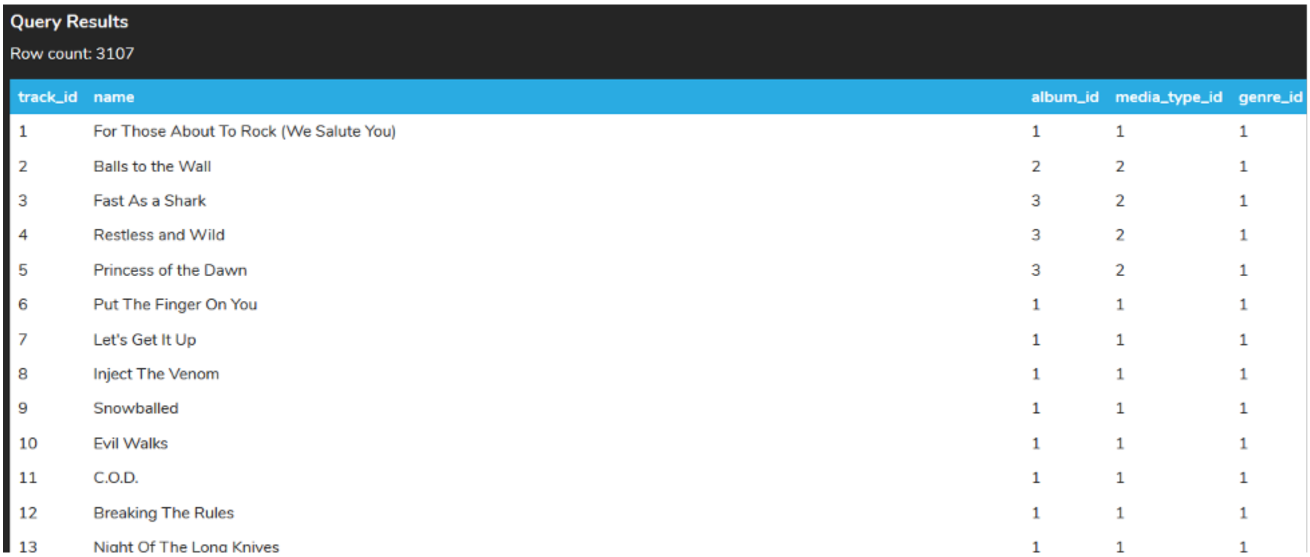
```
SELECT *
FROM invoice
WHERE invoice_date >= '2009-03-01'
AND invoice_date <= '2009-3-31';
```

3. Adding NOT

You can also use the NOT operator to return the opposite result set. Using the example from the prior tutorial, if we wanted to get the tracks that had the genre_id NOT being between 10-20, we could write our query like this:

```
SELECT *
FROM track
WHERE genre_id NOT BETWEEN 10 AND 20;
```

This would include all of the tracks between 1-9 and 21-25 (along with any other genre_id that may be added to the table beyond 25).



Query Results

Row count: 3107

track_id	name	album_id	media_type_id	genre_id
1	For Those About To Rock (We Salute You)	1	1	1
2	Balls to the Wall	2	2	1
3	Fast As a Shark	3	2	1
4	Restless and Wild	3	2	1
5	Princess of the Dawn	3	2	1
6	Put The Finger On You	1	1	1
7	Let's Get It Up	1	1	1
8	Inject The Venom	1	1	1
9	Snowballed	1	1	1
10	Evil Walks	1	1	1
11	C.O.D.	1	1	1
12	Breaking The Rules	1	1	1
13	Night Of The Long Knives	1	1	1

We could do the same thing for dates, to query invoices that were not between 2010-01-01 and 2010-12-31.

```
SELECT *
FROM invoice
WHERE invoice_date NOT BETWEEN '2010-01-01' AND '2010-12-31';
```

This could also be written as:

```
SELECT *
FROM invoice
WHERE invoice_date < '2010-01-01'
OR invoice_date > '2010-12-31';
```

Notice that unlike the BETWEEN statement on its own, this example is exclusive of the values due to the NOT.

Video Transcription

[MUSIC PLAYING] Here, we'll be covering the BETWEEN operator. The BETWEEN operator allows us to search the area based on specific ranges. So for example here, we're looking at the invoice table, looking for the customer ID of the value greater or equal to 2, and then the customer ID of less than or equal to 5. This gives us a range of 2 to 5 for all the customers. So in this case here, all those items will be in the place.

However, this can get a little bit lengthy. You can simplify this by utilizing the BETWEEN operator by changing this out. Say between 2 and 5. Between those two it would work exactly the same way. It simplifies a query. You can make the changes to this without having to worry about those details. One thing to note is that you do want to make sure the value that is the smallest appears first, the value appears last should be the larger value.

If we made a switch between the two, it will return no data in this case here. Note that we can also utilize this for dates. So for example, if we have here for the invoice date, it's greater than a certain date. And then also less than a certain date finding all those invoices in this case here returns eight invoices. We can do the same thing as well, utilizing the between. We can remove everything there. If we run that, it return the same values as well.

[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 BETWEEN operator allows us to check whether a value is in a range of specified values.

Source: Authored by Vincent Tran