IN to Filter Data

by Sophia

WHAT'S COVERED

In this lesson, you will learn about using the IN clause within a SELECT statement to filter specific values, in two parts. Specifically, this lesson will cover:

1. Using IN
2. Using NOT IN

1. Using IN

The IN operator allows you to search using a variety of values for a single column. This can simplify queries by avoiding writing a separate condition using the OR operator. Instead, you can have a list of values enclosed in parentheses separated by commas within the IN operator.

For example, if we wanted to find customers that live in either Brazil, Belgium, Norway, or Austria using the OR operator that we learned in the prior lesson, we would do the following:

```
SELECT *
FROM customer
WHERE country = 'Brazil'
OR country = 'Belgium'
OR country = 'Norway'
OR country = 'Austria';
```

Query Results

Row count: 8

customer_id	first_name	last_name	company	address	city	state	country
1	Luís	Gonçalves	Embraer - Empresa Brasileira de Aeronáutica S.A.	Av. Brigadeiro Faria Lima, 2170	São José dos Campos	SP	Brazil
4	Bjørn	Hansen		Ullevålsveien 14	Oslo		Norway
7	Astrid	Gruber		Rotenturmstraße 4, 1010 Innere Stadt	Vienne		Austria
8	Daan	Peeters		Grétrystraat 63	Brussels		Belgium
10	Eduardo	Martins	Woodstock Discos	Rua Dr. Falcão Filho, 155	São Paulo	SP	Brazil
11	Alexandre	Rocha	Banco do Brasil S.A.	Av. Paulista, 2022	São Paulo	SP	Brazil
12	Roberto	Almeida	Riotur	Praça Pio X, 119	Rio de Janeiro	RJ	Brazil
13	Fernanda	Ramos		Qe 7 Bloco G	Brasília	DF	Brazil

This can get lengthy and increase the chance of the SQL programmer making an error if there are other conditions.

By using the IN operator, the query can be simplified as follows:

```
SELECT *
FROM customer
WHERE country IN ('Brazil', 'Belgium', 'Norway', 'Austria');
```

Query Results

Row count: 8

customer_id	first_name	last_name	company	address	city	state	country
1	Luís	Gonçalves	Embraer - Empresa Brasileira de Aeronáutica S.A.	Av. Brigadeiro Faria Lima, 2170	São José dos Campos	SP	Brazil
4	Bjørn	Hansen		Ullevålsveien 14	Oslo		Norway
7	Astrid	Gruber		Rotenturmstraße 4, 1010 Innere Stadt	Vienne		Austria
8	Daan	Peeters		Grétrystraat 63	Brussels		Belgium
10	Eduardo	Martins	Woodstock Discos	Rua Dr. Falcão Filho, 155	São Paulo	SP	Brazil
11	Alexandre	Rocha	Banco do Brasil S.A.	Av. Paulista, 2022	São Paulo	SP	Brazil
12	Roberto	Almeida	Riotur	Praça Pio X, 119	Rio de Janeiro	RJ	Brazil
13	Fernanda	Ramos		Qe 7 Bloco G	Brasília	DF	Brazil

To add other countries, we don't have to add another comparison; we simply add it to the list of values. Notice that we still use single quotes around the strings that we are comparing.

If we wanted the customers that had the support_rep_id set to 1, 2, 3, or 4, then we would use the following statement:

```
SELECT *
FROM customer
WHERE support_rep_id IN (1,2,3,4);
```

HINT

When we were looking for country, we used 'Brazil' with the quotes because the column is VARCHAR, but when we looked for support_re_id, we just used 1, without the quotes, because it is a number, and the

column data type is INT. Remember when working with character strings, we always use quotes around what we are looking for.

The order of the values does not matter. The results would be the same if we ran the following statement:

```
SELECT *
FROM customer
WHERE support_rep_id IN (4,3,2,1);
```

Similar to the OR operator, the result set is combined based on each comparison.

The IN operator is beneficial when used in conjunction with subqueries, which you will learn about in a later lesson.

---

2. Using NOT IN

You can also use the NOT IN operator to negate the conditional expression. Since all conditional expressions evaluate to true or false, the NOT operator will get the rows that do not match a certain condition. For the NOT IN operator, it would return data that does not fall under that particular criteria. For example, if we wanted to look for customers that are not in Brazil, Belgium, Norway, or Austria, we would need to run the following:

```
SELECT *
FROM customer
WHERE country NOT IN ('Brazil','Belgium','Norway','Austria');
```

This would return all of the customers in any country (or not having any country at all), but excluding the four aforementioned countries:

Query Results

Row count: 51

customer_id	first_name	last_name	company	address	city	state	country	postal_code
2	Leonie	Köhler		Theodor-Heuss-Straße 34	Stuttgart		Germany	70174
3	François	Tremblay		1498 rue Bélanger	Montréal	QC	Canada	H2G 1A7
5	František	Wichterlová	JetBrains s.r.o.	Klanova 9/506	Prague		Czech Republic	14700
6	Helena	Holý		Rilská 3174/6	Prague		Czech Republic	14300
9	Kara	Nielsen		Sønder Boulevard 51	Copenhagen		Denmark	1720
14	Mark	Philips	Telus	8210 111 ST NW	Edmonton	AB	Canada	T6G 2C7
15	Jennifer	Peterson	Rogers Canada	700 W Pender Street	Vancouver	BC	Canada	V6C 1G8
16	Frank	Harris	Google Inc.	1600 Amphitheatre Parkway	Mountain View	CA	USA	94043-1351
17	Jack	Smith	Microsoft Corporation	1 Microsoft Way	Redmond	WA	USA	98052-8300
18	Michelle	Brooks		627 Broadway	New York	NY	USA	10012-2612
19	Tim	Gover	Apple Inc.	1 Infinite Loop	Cupertino	CA	USA	95014

This can be useful with the IN operator if the list of values to compare to is smaller than the opposite set of values. For example, if we wanted to get the tracks that had the genre_id with the values of 1–20, we could use the IN operator like this:

```
SELECT *
FROM track
WHERE genre_id IN (1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20);
```

But since we know there are 25 genre_id values from the genre table, we could simplify the query by using the NOT IN option instead:

```
SELECT *
FROM track
WHERE genre_id NOT IN (21,22,23,24,25);
```

In this case, both would return the same result set of 3,307 rows.

Query Results

Row count: 3307

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

WATCH

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

In this lesson, you learned that the **IN operator** is a useful tool for filtering query results according to predefined criteria in PostgreSQL. The operator checks if any of the specified values match a given column value, enabling you to specify multiple values within parentheses. You also learned that by replacing multiple OR statements with the IN operator or **NOT IN operator**, complex conditions are simplified. Your queries will be more readable and maintainable with this efficient way to filter rows based on a predefined set of values.

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