Filtering data with conditions

In the previous lesson, you learned about the comparison operators that filter data in SQL. In this lesson, you'll learn how to filter data using conditional operators. The concept is similar; while comparison operators filter data based on just one condition, conditional operators are used to filter data based on multiple conditions.

To see how you might use these conditional operators, think about when you're shopping online. You typically want to filter items by prices as well as other criteria, like item type, size, color, or rating. In the same way that you filter your online shopping, you can filter SQL data. If you're looking at a database of items, you can add one or multiple filters to narrow your search down to the exact criteria that you want.

AND

The AND operator is paired with WHERE to create multiple conditions. Records that meet all conditions are retrieved. A statement using WHERE and AND follows this structure:

```
SELECT column1, column2, …
FROM table_name
WHERE condition1 AND condition2 AND condition3,...;
```

Whenever you start working with a new database, execute a simple SELECT query on your new tables to get a sense of what the data contains.

Try creating a query on the medicalcosts table to pull all data for male smokers.

That query can be written like this:

```
SELECT *
FROM medicalcosts
WHERE sex = 'male' AND smoker = 'yes'
```

OR

Like the AND operator, the OR operator is paired with WHERE to create multiple conditions. However, with OR, records that meet at least one of the conditions are retrieved. A statement using WHERE and OR follows this structure:

```
SELECT column1, column2, …
FROM table_name
WHERE condition1 OR condition2 OR condition3,...;
```

Say that you now want to view medical costs for people with one, two, or three children. How would you use pgAdmin and the medicalcosts table to write this query?

Your query should look like this:

```
SELECT *
FROM medicalcosts
WHERE children = 1 OR children = 2 OR children = 3;
```

NOT

The NOT operator is paired with WHERE to retrieve records that don't meet the condition specified.

The structure of this statement is as follows:

```
SELECT column1, column2, …
FROM table_name
WHERE NOT condition;
```

Using the medicalcosts table again, create a query in pgAdmin that pulls all medical cost records for people who don't have zero children. Your SQL query can be written as follows:

```
SELECT *
FROM medicalcosts
WHERE NOT children = 0;
```

Combining AND, OR, and NOT

You can use AND, OR, and NOT together to create different combinations of conditions that must be met. For example, how might you write a SQL query to view all records in the medicalcosts table for females who have either zero children or one child?

That query can be written as follows:

```
SELECT *
FROM medicalcosts
WHERE sex = 'female' AND (children = 0 OR children = 1);
```

Or, if you want to view all records for people who don't have one, two, or three children, your SQL query can be written as follows:

```
SELECT *
FROM medicalcosts
WHERE NOT children = 1 AND NOT children = 2 AND NOT children = 3;
```

IN

When you want to create a query using multiple OR clauses, it can get tedious to write out OR condition1 OR condition2 … for each condition. Fortunately, there's a shortcut for writing out multiple OR clauses: the IN clause. Here's an example of the structure for the IN clause:

```
SELECT column1, column2, …
FROM table_name
WHERE column_name IN (value1, value2, …);
```

Now, revisit the example from the OR section above: imagine that you want to view all medical costs for people with one, two, or three children. In pgAdmin, use the IN shortcut to come up with a query that will retrieve these records. Your SQL query should look like this:

```
SELECT *
FROM medicalcosts
WHERE children IN (1, 2, 3);
```

Compare this query to the query in the earlier OR section. As you can see, using the IN operator shortens the query and makes it easier to write.

BETWEEN

The BETWEEN operator, when paired with WHERE, returns records whose values fall between a certain range. The structure is this:

```
SELECT column1, column2, …
FROM table_name
WHERE column_name BETWEEN value1 AND value2;
```

In pgAdmin, use the BETWEEN clause structure above to create a query that pulls all records with a medical charge between $1,000 and $2,000. That query can be written as follows:

```
SELECT *
FROM medicalcosts
WHERE charges BETWEEN 1000 AND 2000;
```

Execute this query and look through the Data Output. You won't see any records with charges less than $1,000 or greater than $2,000.

LIKE

The LIKE operator is paired with WHERE to find and retrieve data that has a certain pattern. For example, you might be looking for a specific product but can't fully remember the product name—you just know that it starts with the letter a. So, you pull all product names that start with the letter a. The structure of a LIKE clause is shown below.

```
SELECT column1, column2, …
FROM table_name
WHERE column_name LIKE pattern;
```

The LIKE operator is typically used with wildcard characters. Wildcard characters are symbols used to represent a certain number of characters. The two wildcard characters used with LIKE are % and _.

- The % wildcard represents zero, one, or more characters.
- The _ wildcard represents one single character.

To see this in action, say that you're looking for records in the medicalcosts table for the southeast or southwest regions. You can't remember if the data for the words southwest and southeast are abbreviated, split into two words, or are written as just one word in the data—but you do know that they start with the letter s. You can pull all records with a region that begins with the letter s with the following SQL query.

```
SELECT *
FROM medicalcosts
WHERE region LIKE 's%';
```

Keep in mind that capitalization doesn't matter for SQL syntax, but it does matter if the item that you're looking for is capitalized. In this example, regions are written in all lowercase letters, so use lowercase s in your query.

Execute this query. In your Data Output, you'll only see regions starting with s.

Similarly, if you want to pull records with regions that end with the letter t, you can use the following SQL query.

```
SELECT *
FROM medicalcosts
WHERE region LIKE '%t';
```

Maybe you suddenly remember that the region you're looking for has east somewhere in it. You can use the following SQL query to pull all products that contain east.

```
SELECT *
FROM medicalcosts
WHERE region LIKE '%east%';
```

Execute this query. Your output will only list regions that contain east somewhere in the name.

The % wildcard represents some number of characters, so the placement of the wildcard determines where in the value the pattern is searched for.

The _ wildcard represents a single character, and its position also determines where in the value the pattern is searched for. To see the wildcard in action, say that you're searching for records by region. All you know is that it has the letter u as the third letter. You can use the following query to pull all records with regions that have u as the third letter:

```
SELECT *
FROM medicalcosts
WHERE region LIKE '_ _ u%';
```

Execute this query. You'll notice that all regions in your output have u as the third letter.

Similarly, if you want to pull all regions that have n as the first character and you know that the region names are at least three characters in length, you can use the following query:

```
SELECT *
FROM medicalcosts
WHERE region LIKE 'n_%';
```