SQL for Citizen Data Scientists

SQL AND, OR, NOT Operators

SETScholars & WACAMLDS

Often, you'll need to specify multiple conditions in a single WHERE clause, for example, to retrieve rows based on the values in multiple columns. You can use the AND and OR operators to combine two or more conditions into a compound condition.

AND, OR, and a third operator, NOT, are logical operators (often referred to as boolean operators). These logical operators help you create powerful and sophisticated search conditions.

Sample Table

To help you better understand the examples, and enable you to follow along with the tutorial, we are going to use the following sample table.

This table is part of an 'Employee Management System' that contains basic information about employees.

ID	Name	Age	City	Job	Salary
1	Bob	28	New York	Manager	60000
2	Eve	24	New York	Developer	32000
3	Max	26	New York	Janitor	9000
4	Kim	25	Chicago	Manager	55000
5	Joe	23	Chicago	Developer	30000
6	Sam	27	Chicago	Janitor	10000

The AND Operator

The AND logical operator allows you to specify more than one condition in a WHERE statement. It instructs the database to return only those rows that meet all of the specified conditions.

Here's how you use the AND operator to append conditions to your WHERE clause:

```
SELECT column_name(s)
FROM table_name
WHERE condition1 AND condition2 ... ;
```

For example, if you wanted to select employees who are not 'Developers' and who live in 'New York' city, you could run this query:

```
SELECT *
FROM Employees
WHERE Job <> 'Developer' AND City = 'New York';
```

If an employee is not a developer but lives in a different city other than New York, then he is not included in the result set. Similarly, an employee who lives in the New York city but is not a developer by profession, is also not included in the result set. The output generated by this SQL statement is as follows:

ID	Name	Age	City	Job	Salary
1	Bob	28	New York	Manager	60000
3	Max	26	New York	Janitor	9000

In this example, only a single AND clause was used and was thus made up of two filter conditions. But you can specify as many filter conditions as you want, just separate them by an AND keyword.

Here's an extension of the previous query that includes a third condition stating that only those employees whose salary is more than $40,000 should be included:

```
SELECT *
FROM Employees
WHERE Job <> 'Developer' AND City = 'New York' AND Salary > 40000;
```

ID	Name	Age	City	Job	Salary
1	Bob	28	New York	Manager	60000

The OR Operator

The OR operator is exactly the opposite of AND. It instructs the database to return only those rows that match either condition.

Here's how you use the OR operator in your WHERE clause:

```
SELECT column_name(s)
FROM table_name
WHERE condition1 OR condition2 ... ;
```

For example, here is a query that selects only those employees who are either 'Managers' or 'Developers'

```
SELECT *
FROM Employees
WHERE Job = 'Manager' OR Job = 'Developer';
```

ID	Name	Age	City	Job	Salary
1	Bob	28	New York	Manager	60000
2	Eve	24	New York	Developer	32000
4	Kim	25	Chicago	Manager	55000
5	Joe	23	Chicago	Developer	30000

Did you know?

If the first condition is met in the OR WHERE clause, the row is retrieved regardless of the second condition. That is why most DBMSs do not even evaluate the second condition if the first condition is already met.

The NOT Operator

NOT operator has one and only one function: negating whatever condition comes next. It retrieves the rows for which the specified condition is FALSE (NOT TRUE).

The NOT operator is never used by itself; It is always used in conjunction with other operators such as BETWEEN, ANY, AND, OR, or LIKE. That's why its syntax is a little bit different than other operators. The NOT keyword is used before the column name, and not after it, like this:

```
SELECT column_name(s)
FROM table_name
WHERE NOT condition;
```

For example, if you wanted to select all employees who are not from 'Chicago', you could write a query like:

```
SELECT *
FROM Employees
WHERE NOT City = 'Chicago';
```

ID	Name	Age	City	Job	Salary
1	Bob	28	New York	Manager	60000
2	Eve	24	New York	Developer	32000
3	Max	26	New York	Janitor	9000

Order of Evaluation

The WHERE clause can contain any number of AND, OR and NOT operators. Combining these enables you to have a greater degree of filter control.

But you should be aware that these logical operators have different priorities for evaluation: the NOT operator has the highest priority, AND is evaluated next, and the OR operator has the lowest priority. If you do not pay attention to these different priorities, you will get unexpected results.

To demonstrate this, look at an example. Suppose you wanted a list of all employees who are either 'Managers' or 'Developers' and live in 'New York' city, and you wrote a query like this:

```
SELECT *
FROM Employees
WHERE Job = 'Manager' OR Job = 'Developer' AND City = 'New York';
```

ID	Name	Age	City	Job	Salary
1	Bob	28	New York	Manager	60000
2	Eve	24	New York	Developer	32000
4	Kim	25	Chicago	Manager	55000

Look at the result above. The last employee is not from the 'New York' city – so, obviously, that row was not filtered as intended. Why did this happen? The answer is the order of evaluation.

As parentheses have a higher order of evaluation than either of these three operators, the solution to this problem is to explicitly group related operators using parentheses.

Let's modify the above query:

```
SELECT *
FROM Employees
WHERE (Job = 'Manager' OR Job = 'Developer') AND  City = 'New York';
```

ID	Name	Age	City	Job	Salary
1	Bob	28	New York	Manager	60000
2	Eve	24	New York	Developer	32000

As you can see, the DBMS first filtered the OR condition within those parentheses and then the AND condition, which is exactly what we want.

Logical Operations with NULL

When working on logical operators it is important to know about the behavior of NULL values. The following truth tables specify their behavior.

In these tables, the top row and the first column represent the values of the expressions on which the operator works. Follow them and see where they intersect; that value is the resulting value.

Truth table for AND operator

	True	NULL	False
True	True	NULL	False
NULL	NULL	NULL	False
False	False	False	False

Truth table for OR operator

	True	NULL	False
True	True	True	True
NULL	True	NULL	NULL
False	True	NULL	False

Truth table for NOT operator

True	False
NULL	NULL
False	True