SQL for Citizen Data Scientists

SQL WHERE Clause - Learn By Example

SETScholars & WACAMLDS

Database tables usually contain large amounts of data, and you rarely need to retrieve all the rows in a table. More often than not, you will want to retrieve the subset of the table's data as needed for specific operations or reports. Retrieving the desired data involves specifying search criteria, also known as search condition.

The data is filtered by specifying the search condition in the WHERE clause within the SELECT statement. If the condition is true for a row, then that row is included in the result set.

Syntax

```
SELECT column_name(s)
FROM table_name
WHERE condition;
```

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	*NULL*	Manager	55000
5	Joe	23	Chicago	Developer	30000
6	Sam	*NULL*	Chicago	Janitor	10000

Using the WHERE Clause

The WHERE clause allows you to retrieve only rows you are interested in.

For example, perhaps you are interested in retrieving data from the 'Employees' table, but only those employees whose salary is $30,000. The following query employs the WHERE clause to retrieve only one employee record:

```
SELECT *
FROM Employees
WHERE Salary = 30000;
```

ID	Name	Age	City	Job	Salary
5	Joe	23	Chicago	Developer	30000

The rules for qualifying text fields follow the same structure as number fields. However, when using text, you must wrap literals (or text values you specify) in single quotes.

For example, if you wanted to view employees from 'New York', you could run this query:

```
SELECT *
FROM Employees
WHERE City = 'New York';
```

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

The WHERE Clause Operators

The above examples test only for equality – determining if a column contains a specific value. The WHERE clause doesn't always have to test for equality. You can use other comparison operators to filter down records.

SQL supports a whole range of conditional operators.

SQL conditional operators

Operator	Description	Example
=	Equal	WHERE x = 1
!=, <>	Not equal	WHERE x <> 'a'
>	Greater than	WHERE x > 1
<	Less than	WHERE x < 1
>=	Greater than or equal	WHERE x >= 1
<=	Less than or equal	WHERE x <= 1
BETWEEN	Checks if a value lies within a range	WHERE x BETWEEN 1 AND 10
EXISTS	Checks if rows exist matching conditions that you specify	WHERE EXISTS(subquery)
IN	Checks if a value is contained in a set of specified values	WHERE x IN (5, 10, 15, 20)
IS [NOT] NULL	Tests for nullity	WHERE x IS NULL
LIKE	Checks if a value matches a pattern	WHERE x LIKE '%abc_'

You have already seen an example of testing for equality. Let's take a look at some examples that demonstrate the use of other operators.

The first example lists all employees whose salary is more than $40,000

```
SELECT *
FROM Employees
WHERE Salary > 40000;
```

ID	Name	Age	City	Job	Salary
1	Bob	28	New York	Manager	60000
4	Kim	25	*NULL*	Manager	55000

The next example lists all employees who are not developers.

```
SELECT *
FROM Employees
WHERE Job <> 'Developer';
```

ID	Name	Age	City	Job	Salary
1	Bob	28	New York	Manager	60000
3	Max	26	New York	Janitor	9000
4	Kim	25	*NULL*	Manager	55000
6	Sam	*NULL*	Chicago	Janitor	10000

Combining Conditions (with AND and OR operator)

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.

The AND Operator

If two conditions are connected by the AND operator, rows are retrieved for which both conditions are true.

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';
```

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

The first condition `Job <> 'Developer'` filtered out 4 of 6 employee rows, and the second condition `City = 'New York'` filtered out an additional 2 rows, leaving 2 rows in the final result set.

The OR Operator

If two conditions are connected by the OR operator, all rows of a table are retrieved in which either the first or the second condition (or both) is true.

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	*NULL*	Manager	55000
5	Joe	23	Chicago	Developer	30000

Negating a Condition (with 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 the NOT keyword is used before the column name, and not after it.

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

Range Conditions

Along with checking against a single value, you can build conditions that check for a range of values. This type of condition is used when working with numerical or temporal data (data related to date and time).

To check whether an expression falls within a certain range, you can use the BETWEEN operator. Its syntax differs slightly from other WHERE clause operators because it requires two values – the beginning and the end of the range.

The BETWEEN operator can be used, for example, to find all employees whose salaries range between $30,000 to $40,000.

```
SELECT *
FROM Employees
WHERE Salary BETWEEN 30000 and 40000;
```

ID	Name	Age	City	Job	Salary
2	Eve	24	New York	Developer	32000
5	Joe	23	Chicago	Developer	30000

Read more about the BETWEEN operator here.

Membership Conditions

In some cases you might want to check for an expression, not against a single value or range of values, but against a set of values. For example, you might want to find all employees who are either 23, 26 or 28 years old.

While this can be done by OR'ing three conditions together which is not very tedious, imagine if the set of expressions contained 10 or 20 values. For these situations, you can use the IN operator instead:

```
SELECT *
FROM Employees
WHERE Age IN(23, 26, 28);
```

ID	Name	Age	City	Job	Salary
1	Bob	28	New York	Manager	60000
3	Max	26	New York	Janitor	9000
5	Joe	23	Chicago	Developer	30000

If you want to see whether the expression does not exist within a set of expressions, you can use the NOT IN operator.

For example, the following query lists all employees who are neither 23, 26 nor 28 years old.

```
SELECT *
FROM Employees
WHERE Age NOT IN(23, 26, 28);
```

ID	Name	Age	City	Job	Salary
2	Eve	24	New York	Developer	32000
4	Kim	25	*NULL*	Manager	55000

Read more about the IN operator here.

Matching Conditions

By now, you have been introduced to conditions that identify an exact string or a set of strings; the final condition type deals with partial string matches. For example, you may want to find all employees whose second letter of name is 'o'.

When searching for such partial string matches, the LIKE operator is used. There are two wildcards often used in conjunction with the LIKE operator:

% The percent sign matches zero or more characters

_ The underscore character matches just a single character

As an example let's find all employees whose second letter of name is 'o'.

```
SELECT *
FROM Employees
WHERE Name LIKE '_o%';
```

ID	Name	Age	City	Job	Salary
1	Bob	28	New York	Manager	60000
5	Joe	23	Chicago	Developer	30000

Read more about the LIKE operator here.

Checking for NULL Values

You may have noticed that some columns, such as 'Age' and 'City', have NULL values. A NULL is a value that has no value. It is nothing but an absence of value.

NULL values cannot be determined with an '= NULL'. You need to use the IS NULL or IS NOT NULL clauses to identify them. So, to get a list of employees with no recorded city, you could run this query:

```
SELECT *
FROM Employees
WHERE City IS NULL;
```

ID	Name	Age	City	Job	Salary
4	Kim	25	*NULL*	Manager	55000