SQL Server IN operator overview

The IN operator is a logical operator that allows you to test whether a specified value matches any value in a list.

The following shows the syntax of the SQL Server IN operator:

column | expression IN ( v1, v2, v3, ...)

Code language: SQL (Structured Query Language) (sql)

In this syntax:

* First, specify the column or expression to test.
* Second, specify a list of values to test. All the values must have the same type as the type of the column or expression.

If a value in the column or the expression is equal to any value in the list, the result of the IN operator is TRUE.

The IN operator is equivalent to multiple [OR](https://www.sqlservertutorial.net/sql-server-basics/sql-server-or/) operators, therefore, the following predicates are equivalent:

column IN (v1, v2, v3)

column = v1 OR column = v2 OR column = v3

Code language: SQL (Structured Query Language) (sql)

To negate the IN operator, you use the NOT IN operator as follows:

column | expression NOT IN ( v1, v2, v3, ...)

Code language: SQL (Structured Query Language) (sql)

The result the NOT IN operator is TRUE if the column or expression does not equal to any value in the list.

In addition to a list of values, you can use a [subquery](https://www.sqlservertutorial.net/sql-server-basics/sql-server-subquery/) that returns a list of values with the IN operator as shown below:

column | expression IN (subquery)

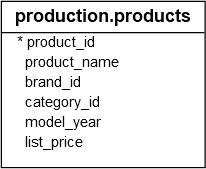
Code language: SQL (Structured Query Language) (sql)

In this syntax, the subquery is a [SELECT](https://www.sqlservertutorial.net/sql-server-basics/sql-server-select/) statement that returns a list of values of a single column.

Note that if a list contains [NULL](https://www.sqlservertutorial.net/sql-server-basics/sql-server-null/), the result of IN or NOT IN will be UNKNOWN.

SQL Server IN operator examples

See the following production.roducts table from the [sample database](https://www.sqlservertutorial.net/sql-server-sample-database/).



A) Using SQL Server IN with a list of values example

The following statement finds the products whose list price is one of the following values: 89.99, 109.99, and 159.99:

SELECT

product\_name,

list\_price

FROM

production.products

WHERE

list\_price IN (89.99, 109.99, 159.99)

ORDER BY

list\_price;

Code language: SQL (Structured Query Language) (sql)



The query above is equivalent to the following query that uses the [OR](https://www.sqlservertutorial.net/sql-server-basics/sql-server-or/) operator instead:

SELECT

product\_name,

list\_price

FROM

production.products

WHERE

list\_price = 89.99 OR list\_price = 109.99 OR list\_price = 159.99

ORDER BY

list\_price;

Code language: SQL (Structured Query Language) (sql)

To find the products whose list prices are not one of the prices above, you use the NOT IN operator as shown in the following query:

SELECT

product\_name,

list\_price

FROM

production.products

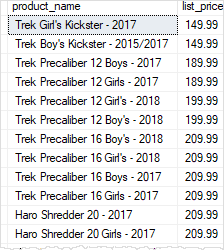
WHERE

list\_price NOT IN (89.99, 109.99, 159.99)

ORDER BY

list\_price;

Code language: SQL (Structured Query Language) (sql)



B) Using SQL Server IN operator with a subquery example

The following query returns a list of product identification numbers of the products located in the store id one and has the quantity greater than or equal to 30:

SELECT

product\_id

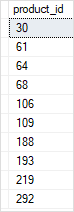
FROM

production.stocks

WHERE

store\_id = 1 AND quantity >= 30;

Code language: SQL (Structured Query Language) (sql)



You can use the query above as a [subquery](https://www.sqlservertutorial.net/sql-server-basics/sql-server-subquery/) in as shown in the following query:

SELECT

product\_name,

list\_price

FROM

production.products

WHERE

product\_id IN (

SELECT

product\_id

FROM

production.stocks

WHERE

store\_id = 1 AND quantity >= 30

)

ORDER BY

product\_name;

Code language: SQL (Structured Query Language) (sql)



In this example:

* First, the [subquery](https://www.sqlservertutorial.net/sql-server-basics/sql-server-subquery/) returned a list of product id.
* Second, the outer query retrieved the product names and list prices of the products whose product id matches any value returned by the subquery.

Overview of the SQL Server BETWEEN operator

The BETWEEN operator is a logical operator that allows you to specify a range to test.

The following illustrates the syntax of the BETWEEN operator:

column | expression BETWEEN start\_expression AND end\_expression

Code language: SQL (Structured Query Language) (sql)

In this syntax:

* First, specify the column or expression to test.
* Second, place the  start\_expression and end\_expression between the BETWEEN and the AND keywords. The start\_expression, end\_expression and the expression to test must have the same data type.

The BETWEEN operator returns TRUE if the expression to test is greater than or equal to the value of the start\_expression and less than or equal to the value of the end\_expression.

You can use the greater than or equal to (>=) and less than or equal to (<=) to substitute the BETWEEN operator as follows:

column | expression <= end\_expression AND column | expression >= start\_expression

Code language: SQL (Structured Query Language) (sql)

The condition that uses the BETWEEN operator is much more readable the one that uses the comparison operators >=, <= and the logical operator [AND](https://www.sqlservertutorial.net/sql-server-basics/sql-server-and/).

To negate the result of the BETWEEN operator, you use NOT BETWEEN operator as follows:

column | expression NOT BETWEEN start\_expression AND end\_expresion

Code language: SQL (Structured Query Language) (sql)

The NOT BETWEEN returns TRUE if the value in the column or expression is less than the value of the  start\_expression and greater than the value of the end\_expression. It is equivalent to the following condition:

column | expression < start\_expression AND column | expression > end\_expression

Code language: SQL (Structured Query Language) (sql)

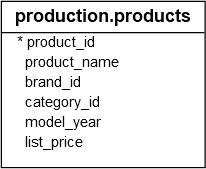
Note that if any input to the BETWEEN or NOT BETWEEN is NULL, then the result is UNKNOWN.

SQL Server BETWEEN examples

Let’s take some examples of using the BETWEEN operator to understand how it works.

A) Using SQL Server BETWEEN with numbers example

See the following products table from the [sample database](https://www.sqlservertutorial.net/sql-server-sample-database/):



The following query finds the products whose list prices are between 149.99 and 199.99:

SELECT

product\_id,

product\_name,

list\_price

FROM

production.products

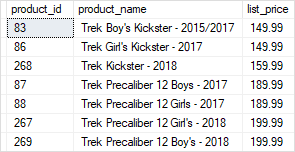
WHERE

list\_price BETWEEN 149.99 AND 199.99

ORDER BY

list\_price;

Code language: SQL (Structured Query Language) (sql)



To get the products whose list prices are in the range 149.99 and 199.99, you use the NOT BETWEEN operator as follows:

SELECT

product\_id,

product\_name,

list\_price

FROM

production.products

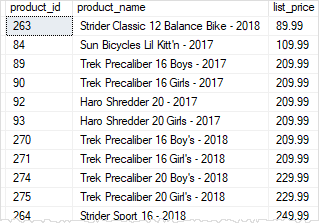
WHERE

list\_price NOT BETWEEN 149.99 AND 199.99

ORDER BY

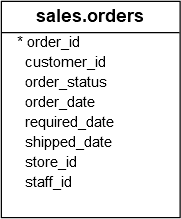
list\_price;

Code language: SQL (Structured Query Language) (sql)



B) Using SQL Server BETWEEN with dates example

Consider the following orders table:



The following query finds the orders that customers placed between January 15, 2017 and January 17, 2017:

SELECT

order\_id,

customer\_id,

order\_date,

order\_status

FROM

sales.orders

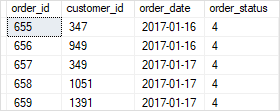
WHERE

order\_date BETWEEN '20170115' AND '20170117'

ORDER BY

order\_date;

Code language: SQL (Structured Query Language) (sql)



Notice that to specify a date constant, you use the format ‘YYYYMMDD‘ where YYYY is 4-digits year e.g., 2017, MM is 2-digits month e.g., 01 and DD is 2-digits day e.g., 15.

SQL Server LIKE operator overview

The SQL Server LIKE is a logical operator that determines if a character string matches a specified pattern. A pattern may include regular characters and wildcard characters. The LIKE operator is used in the [WHERE](https://www.sqlservertutorial.net/sql-server-basics/sql-server-where/) clause of the [SELECT](https://www.sqlservertutorial.net/sql-server-basics/sql-server-select/), [UPDATE](https://www.sqlservertutorial.net/sql-server-basics/sql-server-update/), and [DELETE](https://www.sqlservertutorial.net/sql-server-basics/sql-server-delete/) statements to filter rows based on pattern matching.

The following illustrates the syntax of the SQL Server LIKE operator:

column | expression LIKE pattern [ESCAPE escape\_character]

Code language: SQL (Structured Query Language) (sql)

Pattern

The pattern is a sequence of characters to search for in the column or expression. It can include the following valid wildcard characters:

* The percent wildcard (%): any string of zero or more characters.
* The underscore (\_) wildcard: any single character.
* The [list of characters] wildcard: any single character within the specified set.
* The [character-character]: any single character within the specified range.
* The [^]: any single character not within a list or a range.

The wildcard characters makes the LIKE operator more flexible than the equal (=) and not equal (!=) string comparison operators.

Escape character

The escape character instructs the LIKE operator to treat the wildcard characters as the regular characters. The escape character has no default value and must be evaluated to only one character.

The LIKE operator returns TRUE if the column or expression matches the specified pattern.

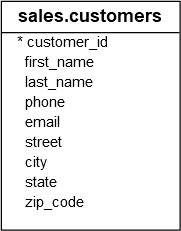
To negate the result of the LIKE operator, you use the NOT operator as follows:

column | expression NOT LIKE pattern [ESCAPE escape\_character]

Code language: SQL (Structured Query Language) (sql)

SQL Server LIKE examples

See the following customers table from the [sample database](https://www.sqlservertutorial.net/sql-server-sample-database/):



The % (percent) wildcard examples

The following example finds the customers whose last name starts with the letter z:

SELECT

customer\_id,

first\_name,

last\_name

FROM

sales.customers

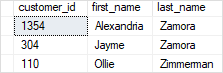
WHERE

last\_name LIKE 'z%'

ORDER BY

first\_name;

Code language: SQL (Structured Query Language) (sql)



The following example returns the customers whose last name ends with the string er:

SELECT

customer\_id,

first\_name,

last\_name

FROM

sales.customers

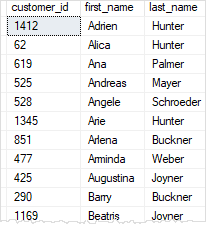
WHERE

last\_name LIKE '%er'

ORDER BY

first\_name;

Code language: SQL (Structured Query Language) (sql)



The following statement retrieves the customers whose last name starts with the letter t and ends with the letter s:

SELECT

customer\_id,

first\_name,

last\_name

FROM

sales.customers

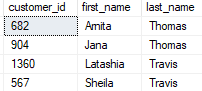
WHERE

last\_name LIKE 't%s'

ORDER BY

first\_name;

Code language: SQL (Structured Query Language) (sql)



The \_ (underscore) wildcard example

The underscore represents a single character. For example, the following statement returns the customers where the second character is the letter u:

SELECT

customer\_id,

first\_name,

last\_name

FROM

sales.customers

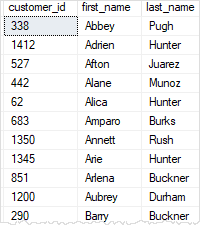
WHERE

last\_name LIKE '\_u%'

ORDER BY

first\_name;

Code language: SQL (Structured Query Language) (sql)



The pattern \_u%

* The first underscore character ( \_) matches any single character.
* The second letter u matches the letter u exactly
* The third character % matches any sequence of characters

The [list of characters] wildcard example

The square brackets with a list of characters e.g., [ABC] represents a single character that must be one of the characters specified in the list.

For example, the following query returns the customers where the first character in the last name is Y or Z:

SELECT

customer\_id,

first\_name,

last\_name

FROM

sales.customers

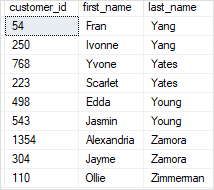
WHERE

last\_name LIKE '[YZ]%'

ORDER BY

last\_name;

Code language: SQL (Structured Query Language) (sql)



The [character-character] wildcard example

The square brackets with a character range e.g., [A-C] represent a single character that must be within a specified range.

For example, the following query finds the customers where the first character in the last name is the letter in the range A through C:

SELECT

customer\_id,

first\_name,

last\_name

FROM

sales.customers

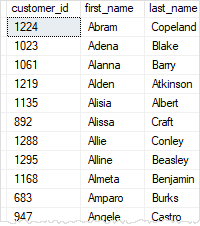
WHERE

last\_name LIKE '[A-C]%'

ORDER BY

first\_name;

Code language: SQL (Structured Query Language) (sql)



The [^Character List or Range] wildcard example

The square brackets with a caret sign (^) followed by a range e.g., [^A-C] or character list e.g., [ABC] represent a single character that is not in the specified range or character list.

For example, the following query returns the customers where the first character in the last name is not the letter in the range A through X:

SELECT

customer\_id,

first\_name,

last\_name

FROM

sales.customers

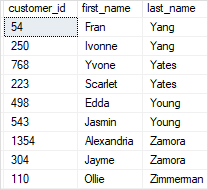
WHERE

last\_name LIKE '[^A-X]%'

ORDER BY

last\_name;

Code language: SQL (Structured Query Language) (sql)



The NOT LIKE operator example

The following example uses the NOT LIKE operator to find customers where the first character in the first name is not the letter A:

SELECT

customer\_id,

first\_name,

last\_name

FROM

sales.customers

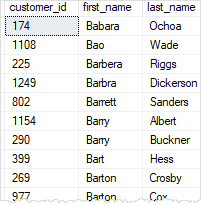
WHERE

first\_name NOT LIKE 'A%'

ORDER BY

first\_name;

Code language: SQL (Structured Query Language) (sql)



SQL Server LIKE with ESCAPE example

First, [create a new table](https://www.sqlservertutorial.net/sql-server-basics/sql-server-create-table/) for the demonstration:

CREATE TABLE sales.feedbacks (

feedback\_id INT IDENTITY(1, 1) PRIMARY KEY,

comment VARCHAR(255) NOT NULL

);

Code language: SQL (Structured Query Language) (sql)

Second, [insert some rows](https://www.sqlservertutorial.net/sql-server-basics/sql-server-insert/) into the sales.feedbacks table:

INSERT INTO sales.feedbacks(comment)

VALUES('Can you give me 30% discount?'),

('May I get me 30USD off?'),

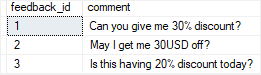
('Is this having 20% discount today?');

Code language: SQL (Structured Query Language) (sql)

Third, [query data](https://www.sqlservertutorial.net/sql-server-basics/sql-server-select/) from the sales.feedbacks table:

SELECT \* FROM sales.feedbacks;

Code language: SQL (Structured Query Language) (sql)



If you want to search for 30% in the comment column, you may come up with a query like this:

SELECT

feedback\_id,

comment

FROM

sales.feedbacks

WHERE

comment LIKE '%30%';

Code language: SQL (Structured Query Language) (sql)

SQL Server LIKE without ESCAPE clause

The query returns the comments that contain 30% and 30USD, which is not what we expected.

To solve this issue, you need to use the ESCAPE clause:

SELECT

feedback\_id,

comment

FROM

sales.feedbacks

WHERE

comment LIKE '%30!%%' ESCAPE '!';

Code language: SQL (Structured Query Language) (sql)

SQL Server LIKE with ESCAPE clause

In this query the  ESCAPE clause specified that the character ! is the escape character. It instructs the LIKE operator to treat the % character as a literal string instead of a wildcard. Note that without the ESCAPE clause, the query would return an empty result set.