LAB - LIKE Operator

In this lab, you will learn how to use the SQL Server `LIKE` to check whether a character string matches a specified pattern.

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` clause of the `SELECT`, `UPDATE`, and `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]
```

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.

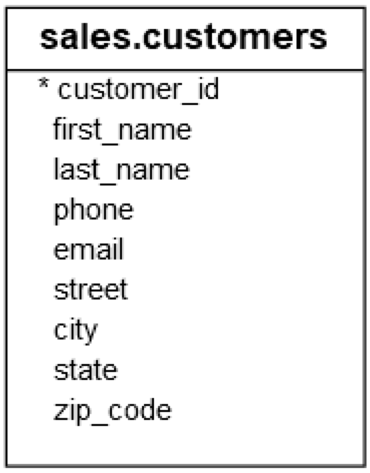
The `LIKE` operator returns `TRUE` if the column or expression matches the specified pattern.

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

```
column | expression NOT LIKE pattern [ESCAPE escape_character]
```

Examples

See the following `customers` table

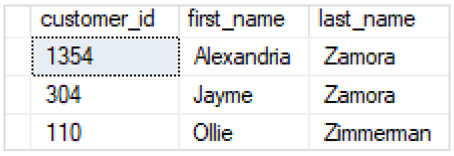


sales.customers
* customer_id
first_name
last_name
phone
email
street
city
state
zip_code

The % (percent) wildcard examples

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

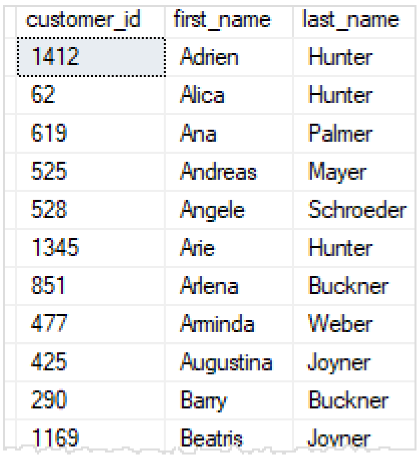
```
SELECT
    customer_id,
    first_name,
    last_name
FROM
    sales.customers
WHERE
    last_name LIKE 'z%'
ORDER BY
    first_name;
```



customer_id	first_name	last_name
1354	Alexandria	Zamora
304	Jayme	Zamora
110	Ollie	Zimmerman

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

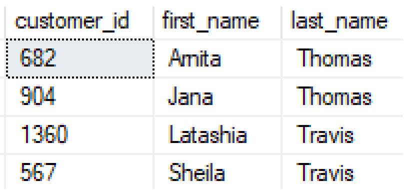
```
SELECT
    customer_id,
    first_name,
    last_name
FROM
    sales.customers
WHERE
    last_name LIKE '%er'
ORDER BY
    first_name;
```



customer_id	first_name	last_name
1412	Adrien	Hunter
62	Alica	Hunter
619	Ana	Palmer
525	Andreas	Mayer
528	Angele	Schroeder
1345	Arie	Hunter
851	Arlena	Buckner
477	Arminda	Weber
425	Augustina	Joyner
290	Barry	Buckner
1169	Beatris	Joyner

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;
```

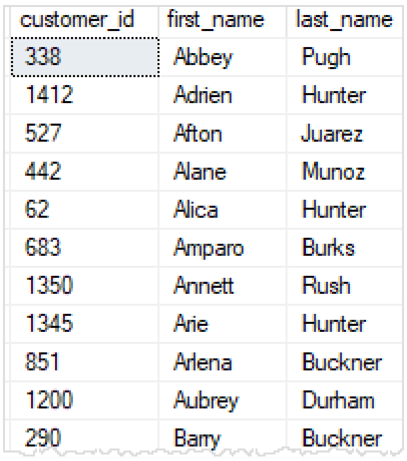


customer_id	first_name	last_name
682	Amita	Thomas
904	Jana	Thomas
1360	Latashia	Travis
567	Sheila	Travis

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;
```



customer_id	first_name	last_name
338	Abbey	Pugh
1412	Adrien	Hunter
527	Afton	Juarez
442	Alane	Munoz
62	Alica	Hunter
683	Amparo	Burks
1350	Annett	Rush
1345	Arie	Hunter
851	Arlena	Buckner
1200	Aubrey	Durham
290	Barry	Buckner

The pattern `_u%`

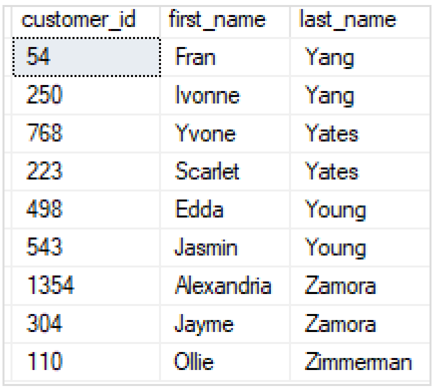
- The first underscore character (`_`) matches any single character.
- The second letter `u` matches the letter u exactly
- The third character `%` matches any sequence of characters

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;
```



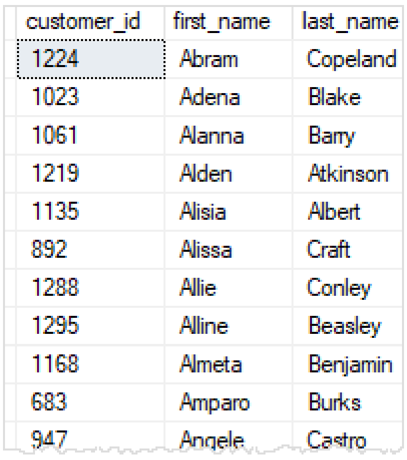
customer_id	first_name	last_name
54	Fran	Yang
250	Ivonne	Yang
768	Yvone	Yates
223	Scarlet	Yates
498	Edda	Young
543	Jasmin	Young
1354	Alexandria	Zamora
304	Jayme	Zamora
110	Ollie	Zimmerman

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;
```



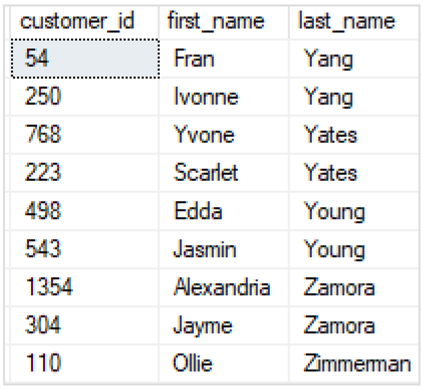
customer_id	first_name	last_name
1224	Abram	Copeland
1023	Adena	Blake
1061	Alanna	Barry
1219	Alden	Atkinson
1135	Alisia	Albert
892	Alissa	Craft
1288	Allie	Conley
1295	Alline	Beasley
1168	Almeta	Benjamin
683	Amparo	Burks
947	Angele	Castro

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;
```

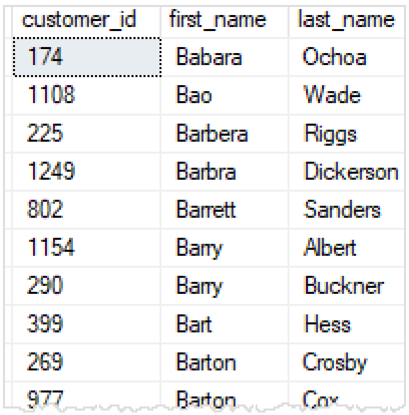


customer_id	first_name	last_name
54	Fran	Yang
250	Ivonne	Yang
768	Yvone	Yates
223	Scarlet	Yates
498	Edda	Young
543	Jasmin	Young
1354	Alexandria	Zamora
304	Jayme	Zamora
110	Ollie	Zimmerman

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;
```



customer_id	first_name	last_name
174	Babara	Ochoa
1108	Bao	Wade
225	Barbera	Riggs
1249	Barbra	Dickerson
802	Barrett	Sanders
1154	Barry	Albert
290	Barry	Buckner
399	Bart	Hess
269	Barton	Crosby
977	Barton	Cox

`LIKE` with `ESCAPE` example

First, create a new table for the demonstration:

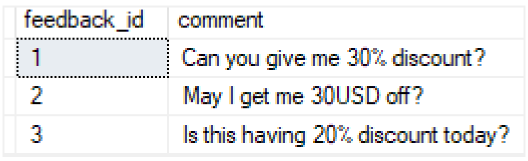
```
CREATE TABLE sales.feedbacks (
   feedback_id INT IDENTITY(1, 1) PRIMARY KEY,
    comment     VARCHAR(255) NOT NULL
);
```

Second, insert some rows 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?');
```

Third, query data from the `sales.feedbacks` table:

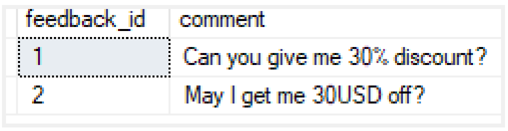
```
SELECT * FROM sales.feedbacks;Code language: SQL (Structured Query Language) (sql)
```



feedback_id	comment
1	Can you give me 30% discount?
2	May I get me 30USD off?
3	Is this having 20% discount today?

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%';
```

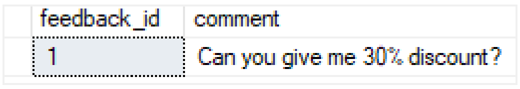


feedback_id	comment
1	Can you give me 30% discount?
2	May I get me 30USD off?

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 '!';
```



	feedback_id	comment
	1	Can you give me 30% discount?

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.

In this tutorial, you have learned how to use the SQL Server `LIKE` operator to check if a character string matches a specified pattern.