In this lesson we are looking at the LIKE command and the Wildcards. The Wildcards are useful because it lets us select data only based off a part of a field instead of an entire field.

For example, we are running a promotion on our customers that we want to give them a discount if they have the letter ‘t’ in their last name.

We know how to select everything from a table, using an asterisk (\*) on the table name.

*SELECT \* FROM customers;*

|  |  |  |
| --- | --- | --- |
| first\_name  character varying (100) | id  integer | last\_name  character varying (255) |
| Rolf | 1 | Smith |
| Jose | 2 | Salvatierra |
| Anne | 2 | Watson |
| Craig | 4 | Scott |
| Michael | 5 | Adam |

Now to select only the customers who has the letter ‘t’ in their last name we have to use the LIKE command. This allows us to compare a field with a pattern. We need to write our SELECT statement as before, but this time use the WHERE clause after it and then use the LIKE command and then write a pattern, this pattern can use a couple of wildcards, one of them is the ‘%’ wildcard, which if we run, we get everything as we got from the simple SELECT statement.

*SELECT \* FROM customers WHERE last\_name LIKE ‘%’;*

|  |  |  |
| --- | --- | --- |
| first\_name  character varying (100) | id  integer | last\_name  character varying (255) |
| Rolf | 1 | Smith |
| Jose | 2 | Salvatierra |
| Anne | 2 | Watson |
| Craig | 4 | Scott |
| Michael | 5 | Adam |

We can also use the underscore ‘\_’ wildcard, and if we use four of them, we will only get Adam as the last name.

*SELECT \* FROM customers WHERE last\_name LIKE ‘\_\_\_\_’;*

|  |  |  |
| --- | --- | --- |
| first\_name  character varying (100) | id  integer | last\_name  character varying (255) |
| Michael | 5 | Adam |

So, remember four underscores get us Adam, and one % symbol gets us all customers. The % symbol is just like the asterisk (\*) symbol. It means anything, any character, the underscore symbol (\_) means a single character. That is why when we used underscore, we got Adam, because he is the only customer that has the four-letter in his last name.

Now we will see how we can use % to get the customers who have the letter ‘t’ in their last names.

*SELECT \* FROM customers WHERE last\_name LIKE ‘%t%’;*

|  |  |  |
| --- | --- | --- |
| first\_name  character varying (100) | id  integer | last\_name  character varying (255) |
| Rolf | 1 | Smith |
| Jose | 2 | Salvatierra |
| Anne | 2 | Watson |
| Craig | 4 | Scott |

As we can see that we got all the customers that have the letter ‘t’ in their last names. In the pattern *%t%,* it means that any character before the letter ‘t’ and any character after the letter ‘t’.

Now let’s see how we can use underscore (\_) and ‘%’ symbol in our pattern to get an output.

*SELECT \* FROM customers WHERE last\_name LIKE ‘%t\_’;*

|  |  |  |
| --- | --- | --- |
| first\_name  character varying (100) | id  integer | last\_name  character varying (255) |
| Rolf | 1 | Smith |
| Craig | 4 | Scott |

Here, we only got the customer that has a single letter after the letter ‘t’, that is because we only used one underscore after ‘t’ thus letting Postgres know that we are only looking for customers that have only one letter after the letter ‘t’.

This is how we can do a LIKE filtering, which is not the fastest thing to do, so unless we require this sort of filtering, we do not need to use it. It is always faster to filter by specific things, because it decreases a bit of the power of the server. However, this exists, and we can use it when we need it. This is how we can filter based on a pattern. Remember the % symbol means any character at any number of times and an underscore (\_) symbol means a single character.