Basic filters on SQL queries

One of the most powerful features of SQL is its ability to filter.
In this video, we're going to learn how this helps us make better queries and select more specific pieces of data from a database.

Filtering is selecting data that match a certain condition.
Think of filtering as a way of only choosing the data we want.
Let's say we wanted to select apples from a fruit cart.
Filtering allows us to specify what kind of apples we want to choose.
When we go buy apples, we might explicitly say, "Choose only apples that are fresh."
This removes apples that aren't fresh from the selection.
This is a filter!
As a security analyst, you might filter a log-in attempts table to find all attempts from a specific country.
This could be done by applying a filter on the country column.
For example, you could filter to just return records containing Canada.

Before we get started, we need to focus on an important part of the syntax of SQL.
Let's learn about operators.
An operator is a symbol or keyword that represents an operation.
An example of an operator would be the equal to operator.
For example, if we wanted to find all records that have USA in the country column, we use country = 'USA'
To filter a query in SQL, we simply add an extra line to the SELECT and FROM statement we used before.
This extra line will use a WHERE clause.
In SQL, WHERE indicates the condition for a filter.
After the keyword WHERE, the specific condition is listed using operators.
So if we wanted to find all of the login attempts made in the United States, we would create this filter.
In this particular condition, we're indicating to return all records that have a value in the country column that is equal to USA.

Let's try putting it all together in SQL.
We're going to start with selecting all the columns from the log_in_attempts table. And then add the WHERE filter.
Don't forget the semicolon!
This tells us we finished the SQL statement.
Now, let's run this query! Because of our filter, only the rows where the country of the log-in attempt was USA are returned.

In the previous example, the condition for our filter was based simply on returning records that are equal to a particular value.

We can also make our conditions more complex by searching for a pattern instead of an exact word.
For example, in the employees table, we have a column for office.
We could search for records in this column that match a certain pattern.
Perhaps we might want all offices in the East building.
To search for a pattern, we used the percentage sign to act as a wildcard for unspecified characters.
If we ran a filter for 'East%', this would return all records that start with East --
for example, the offices East-120, East-290, and East-435.

When searching for patterns with the percentage sign, we cannot use the equals operator.
Instead, we use another operator, LIKE.
LIKE is an operator used with WHERE to search for a pattern in a column.

Since LIKE is an operator, similar to the equal sign, we use it instead of the equal sign.
So, when our goal is to return all values in the office column that start with the word East, LIKE would appear in a WHERE clause.

Let's go back to the example in which we wanted to filter for log-in attempts made in the United States.
Imagine that we realize that our database contains inconsistencies with how the United States is represented.
Some entries use US while others use USA.
Let's get into SQL and apply this new type of filter with LIKE.
We're going to start with the same first two lines of code because we want to select all columns from the log-in attempts table.
And we're going to add a filter with LIKE so that records will be returned if they contain a value in the country column beginning with the characters US.
This includes both US and USA.
Let's run this query to check if the output changes. This returns all the entries where the user location was in the United States.
And now we can use the LIKE clause to filter columns based on a pattern!

Wow, we've already learned how to get very precise with our database and get exactly the data we need with one single query.
I'm excited for what's next!

Revision #1
Created 2023-07-09 18:21:28 UTC by naruzkurai
Updated 2023-07-09 18:24:47 UTC by naruzkurai