Join tables in SQL

You've already learned a lot about SQL queries and filters. Nice work!
The last concept we're introducing in this section is joining tables when querying a database.
This is helpful when you need information from two different tables in a database.
Let's say we have two tables: one that tells us about security vulnerabilities of different operating systems, and one about different machines in our company, including their operating systems.
Having the ability to combine them gives us a list of vulnerable machines.
That's pretty cool, right?

First, let's start talking about the syntax of joins.
Since we're working with two tables now, we need a way to tell SQL what table we're picking columns from.
In our example database, we have an employee_id column in both the employees table and the machines table.
In SQL statements that contain two columns, SQL needs to know which column we're referring to.
The way to resolve this is by writing the name of the table first, then a period, and then the name of a column.
So, we would have employees followed by a period, followed by the column name.
This is the employee_id column for the employees table.
Similarly, this is the employee_id column for the machines table.
Now that we understand this syntax, let's apply it to a join!

Imagine that we want to get a deeper understanding of the employees accessing the machines in our company.
By joining the employees and the machines tables, we can do this!
We first need to identify the shared column that we'll use to connect the two tables.
In this case, we'll use a primary key and one table to connect to another table where it's a foreign key.
The primary key of the employees table is employee_id, which is a foreign key in the machines table.
employee_id is a primary key in the employees table because it has a unique value for every row in the employees table, and no empty values.
We don't have a guarantee that the employee_id column in the machines table follows the same criteria since it's
a foreign key and not a primary key.

Next, we'll use a type of join called an INNER JOIN.
An INNER JOIN returns rows matching on a specified column that exists in more than one table.
Tables usually contain many more rows, but to further explain what we mean by INNER JOIN, let's focus on just four rows from the employees table and four rows from the machines table.
We'll also look at just a few columns of each table for this example.
Let's say we choose employee_id in both tables to perform an INNER JOIN.
Let's look at the two rows where there is a match.

Both tables have 1188 and 1189 in their respective employee_id columns, so they are considered a match.
The results of the join is the two rows that have 1188 and 1189 and all columns from both tables.
Before we move on to the queries, we have to talk about the NULL values in the tables.
In SQL, NULL represents a missing value due to any reason.
In this case, this might be machines that are not assigned to any employee.
Now, let's bring this into SQL and do an INNER JOIN on the full tables.
Let's imagine we want to join these tables in order to get a list of users and their office location that also shows
what operating system they use on their machines.
employee_id is a common column between these tables, and we can use this to join them.
But we won't need to show this column in the results.
First, let's start with a basic query that indicates we want to select the username, office, and operating_system columns.
We want employees to be our first or left table, so we'll use that in our FROM statement.
Now, we write the part of the query that tells SQL to join the machines table with the employees table.

Let's break down this query.
INNER JOIN tells SQL to perform the INNER JOIN.
Then, we name the second table we want to combine with the first.
This is called the right table.
In this case, we want to join machines with the employees table that was already identified after FROM.
Lastly, we tell SQL what column to base the join on.
In our case, we're using the employee_id column.
Since we're using two tables, we have to identify the table and follow that with the column name.
So, we have employees.employee_id. And machines.employee_id.

Let's review the output.
Perfect! We have now joined two tables.
The results of our query displays the records that match on the employee_id column.
Notice that these records contain columns from both tables, but only the ones we've indicated through our SELECT statement.
There are other types of joins that don't require a match to join two tables, and we're going to discuss those in
the next video. I'll meet you there!

Revision #1
Created 2023-07-10 13:14:05 UTC by naruzkurai
Updated 2023-07-10 13:15:56 UTC by naruzkurai