Introduction to SQL: 6th lesson – Joining Data

You have the tools to obtain data from a single table in whatever format you want it. But what if the data you want is spread across multiple tables?

That's where JOIN comes in! JOIN is incredibly important in practical SQL workflows. So let's get started.

JOIN function:

Using JOIN, we can write a query to create a table with just two columns: the name of the pet and the name of the owner.

query = “””

SELECT p.Name AS Pet\_Name, o.Name AS Owner\_Name

FROM `bigquery-public-data.pet\_records.pets` AS p

INNER JOIN `bigquery-public-data.pet\_records.owners` AS o

ON p.ID = o.Pet\_ID

“””

---------------------- -----------------

Pet\_Name Owner\_Name

---------------------- -----------------

Dr. Harris Bonkers Aubrey Little

Moon Magnus Burnsides

Ripley Chett Crawfish

Tom Jules Spinner

We combine information from both tables by matching rows where the ID column in the pets table matches the Pet\_ID column in the owners table.

In the query, ON determines which column in each table to use to combine the tables. Notice that since the ID column exists in both tables, we have to clarify which one to use. We use p.ID to refer to the ID column from the pets table, and o.Pet\_ID refers to the Pet\_ID column from the owners table. In general, when you're joining tables, it's a good habit to specify which table each of your columns comes from. That way, you don't have to pull up the schema every time you go back to read the query.

The type of JOIN we're using today is called an INNER JOIN. That means that a row will only be put in the final output table if the value in the columns you're using to combine them shows up in both the tables you're joining. For example, if Tom's ID number of 4 didn't exist in the pets table, we would only get 3 rows back from this query. There are other types of JOIN, but an INNER JOIN is very widely used, so it's a good one to start with.