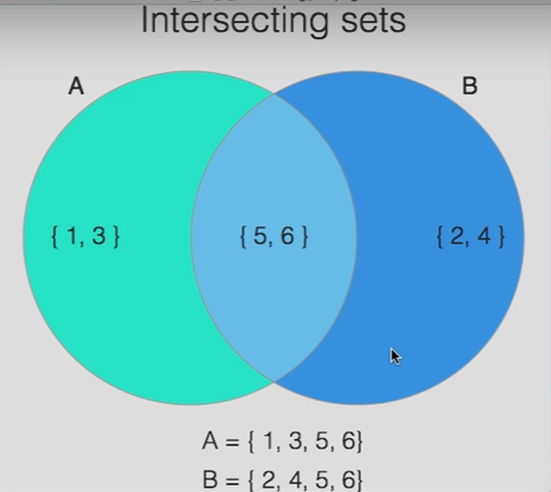
In this lesson we are looking at JOINS.

*JOINS*

* *JOINs are one of the key elements of relational databases*
* *They allow us to retrieve data from multiple tables at once*
* *Let’s go over the different types of JOIN*
* *JOINs are like Sets*
* *We have looked at Sets in Python before:* ***unordered groups of unique elements***
* *JOINs treat rows of data as if they were Sets*
* *We can perform Set operations on the Tables*
* *Let’s see how a set operation can be translated to a JOIN and how they are similar*
* *Remember the intersect operation between sets in Venn Diagram*
* *Set Intersection is the elements those are common between two sets*

**

* *We have set A which has elements 1, 3, 5 and 6*
* *We have set B which has elements 2, 4, 5 and 6*
* *As you can see 5 and 6 are common to both sets*
* *So, if we were to calculate the intersection between A and B, then we get the elements 5 and 6*
* *So, 5 and 6 are elements common to both sets*
* *The first and most common type of JOIN we are going to be looking at is similar to the INTERSECT*
* *And that is the Inner Join*

*INNER JOIN*

* *The SQL INNER JOIN is akin (similar) to Set Intersection*
* *INNER JOIN selects rows from table1 and table2 where they match the selecting column*

Let’s say we have a Customers’ table, where each customer has an ID and a name. Then we also have an Orders’ table, this Orders’ table has an Order ID column containing ID for each order the customers made, and a customer ID column containing the IDs of each customer corresponding with the Order ID with each order they made. Then we also have a Product column containing the names of the product that each of those customers bought through their order.



Now we can see that Orders’ table only contains the Order ID, the Customer ID and the names of the products those were bought through the orders, but we do not have the names of customers that made the orders in the Orders’ table, we only have their IDs in the Orders’ table.

To get to know the name of customer that made each order we need to go the customers’ table and match each of their IDs with the customer IDs in the Orders’ table to see who made which order. We need to do this matching manually for all the orders made, which is not ideal.

Postgres lets us query our database in a way that gives us the customers names according to the orders that they have done. In that way we do not have to go looking for customers names in a different table.

The way to do that is by using a JOIN called INNER JOIN.

Let’s see how we can write the syntax of joining these two tables by using an INNER JOIN between the Customer ID and the Order ID to get the names of the customers according to the orders.

*SELECT \* FROM Customers*

*INNER JOIN Orders*

*ON Customers.ID = Orders.Customers\_ID*

According to this syntax, we are selecting all customers, but we are selecting it using an inner join to connect the Customer ID column from the customers’ table with Customer ID column in the Orders’ table.

This query will give us all the entries from the Customers’ table and the Orders’ table from the rows where the Customers’ ID matched in the Customers’ table matched with the Customers’ ID in the Orders’ table.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Customers.ID | Customers.Name | Orders.ID | Orders.Customers\_ID | Orders.Product |
| 1 | Jose | 1 | 1 | Chair |
| 1 | Jose | 2 | 1 | Pen |
| 1 | Jose | 3 | 1 | Monitor |
| 3 | Anna | 4 | 3 | Headphones |

As Jose is the customer with Customer ID-1 who made the first three orders in the Orders’ table, that is why as a result we get all the entries from the orders’ table for Jose in our resultant table showing the Order IDs for the orders that he made, his customer ID for each of those orders and the products he bought through each of those orders.

Then we get Anna as our next customer as she made the next order after Jose and the resultant table shows her ID, the ID of the order she made, her customer ID and the product she bought through her order.

Notice that this is the INNER JOIN, it gets data from the table on the left and from the table on the right, if they match the selecting columns, so here the resultant table that we got only has customers with IDs 1 and 3 because they are the only customers, those are present in the Orders’ table.

The resultant table is what we are selecting from, so we are selecting from the customers’ inner join orders on Customer.ID equals Orders.Customers\_ID.

*OUTER JOIN*

* *This selects all rows from the table1, on the left, and the rows from the table2, on the right, if they match*
* *If they don’t match, the data for the right table is blank*



We are using the same table for OUTER JOIN or LEFT JOIN and we can see that Customers with IDs 2 and 4 has no orders so let’s see what happens for them in our resultant table.

*SELECT \* FROM Customers*

*LEFT JOIN Orders*

*ON Customers.ID = Orders.Customer\_ID*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Customers.ID | Customers.Name | Orders.ID | Orders.Customer\_ID | Orders.Product |
| 1 | Jose | 1 | 1 | Chair |
| 1 | Jose | 2 | 1 | Pen |
| 1 | Jose | 3 | 1 | Monitor |
| 3 | Anna | 4 | 3 | Headphones |
| 2 | Rolf |  |  |  |
| 4 | Robert |  |  |  |

This is an OUTER JOIN or a LEFT JOIN. We have got all the data from the left table and we have got data from the right table where it matches and if it does not match then it will remain blank.

*RIGHT JOIN*

* *Opposite to LEFT JOIN*
* *This selects all the rows from the table on the right, and then the rows from the table on the left if they match*
* *If they do not match, then the data for the table on the left is blank*



Here is an example of RIGHT JOIN and notice how we have changed the Orders’ table. Now, Order with ID number 2 has no customers associated with it. We can assume that the item was a part of a promotion and nobody bought this item, so this was given away for free.

Let’s see what happens when we execute this query on this two table.

*SELECT \* FROM Customers*

*RIGHT JOIN Orders*

*ON Customers.ID = Orders.Customer\_ID*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Customers.ID | Customers.Name | Orders.ID | Orders.Customers\_ID | Orders.Product |
| 1 | Jose | 1 | 1 | Chair |
| 1 | Jose | 3 | 1 | Monitor |
|  |  | 2 |  | Pen |
| 3 | Anna | 4 | 3 | Headphones |

In our resultant table, we get customer with ID 1, with his first order, then we get customer with ID 1 with his other order which is order number 3 and then we get the second order from the orders’ table with no customer, so we are getting the data from the right but in this case, order number 2 does not have a customer, so it is left blank. Then we get customer with ID number 3 with her other order.

What we are showing here is that data from the right table and then we are putting in data from the left table if it exists, and if it does not exist then it is blank. But we still show the data from the right table which is the Orders’ table.

*FULL JOIN*

* *This selects all rows from both tables, matching them if there is a match and then it displays it on the resultant table but if there is no match then it leaves those entries blank*
* *Think of it both as a LEFT and a RIGHT join*



It is the same table as before, this time we are doing a FULL JOIN.

*SELECT \* FROM Customers*

*FULL JOIN Orders*

*ON Customers.ID =Orders.Customers\_ID*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Customers.ID | Customers.Name | Orders.ID | Orders.Customer\_ID | Orders.Product |
| 1 | Jose | 1 | 1 | Chair |
| 1 | Jose | 3 | 1 | Monitor |
| 2 | Rolf |  |  |  |
|  |  | 2 |  | Pen |
| 3 | Anna | 4 | 3 | Headphones |
| 4 | Robert |  |  |  |

The first row is the first customer with his order, then the same customer with his order id number 3, then we have the second customer with no orders. Then we have got the order with no customer, so the customer field is blank. Then we have our third customer with her order and then we have our fourth customer with no order.

As we can see that we have got all the data in our table and we are matching the data where possible, but if a data cannot be matched then it is left blank for the part that is missing.

This is particularly not useful for an application because there are too many data missing essentially, but it can be used, and it is still used. It is mainly useful when we are doing data analysis and things like that. Nevertheless, we are going to find these four JOINS towards our journey in SQL. That is why it is important to know what they all are.

*JOINS*

* *Are essential to relational data, as it lets us get data from various tables*
* *Its much quicker to extract data using JOINS than getting the data from the tables separately and matching the data in our application*
* *The most common JOIN by far is the INNER JOIN, followed by the OUTER JOIN (LEFT JOIN)*