**SQL con Python: JOIN**

Hasta ahora hemos hecho queries sobre una única tabla, pero **¿y si queremos juntar datos de varias tablas?** Para eso están los JOINs. Para ello **necesitas tener uno o varios campos comunes entre ambas tablas, que se denominan CLAVES**. [como ya vimos en la lejana sesión inicial dedicada a bases de datos relacionales]

**¿Cuándo usarlos?** Por ejemplo, si tenemos una tabla con un conjunto de clientes y necesitamos añadirles campos nuevos, tendremos que acudir a otras tablas donde esté ese identificador de cliente y aplicar un JOIN. Es lo que se conoce como *pegar campos* a otra tabla.

O imagina que tienes una tabla con todos tus pedidos, con muchos campos (ciudad, dirección, cliente...) y en otra tabla únicamente los números de pedido que no se llegaron a entregar. Si quieres filtrar dentro de tu tabla total de pedidos los que no se llegaron a entregar, podrías aplicar lo que se llama un INNER JOIN de manera que te quedes con lo común en ambas tablas, siendo tu clave el identificativo del pedido...

Veamos la teoría sobre los JOIN que existe, pero no te agobies con los nombres, no necesitas saber si estás haciendo un inner, un outter o un left inner outter simpatetic join, lo que necesitas es saber cómo quieres juntar tus tablas y luego aplicar el concepto adecuado.

**Tipos de JOINS**

Vamos a verlo primero con un ejemplo sencillo de dos tablas:

**Tabla 1: Empleados**



(El -1 quiere decir por ejemplo que todavía no tiene departamento asignado)

**Tabla 2: Departamentos**

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Como puedes ver tienen dos columnas que comparten valores y que (en este caso) se llaman igual: id\_departamento

Supongamos ahora que queremos unirlas usando esa columna común y que llamamos tabla de la izquierda a la tabla 1 y tabla de la derecha a la tabla\_2, cuántas formas hay de hacerlo...:

**Left (Outer) Join**

1. Podemos hacerlo considerando únicamente la tabla\_1 como directora es decir unimos a cada empleado los datos del departamento que corresponda con el id\_departamento. El resultado sería:

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Hemos pegado el nombre del departamento correspondiente y a Luis un NaN o Null porque su departamento no existe en la tabla 2.

Bien a este JOIN en el que manda la tabla de la IZQUIERDA o primera tabla mencionada se le llama LEFT JOIN (que curioso verdad) y también LEFT OUTER JOIN.

Su sintaxis en SLQ sería algo como:

**SELECT** A.id\_empleado, A.nombre, A.id\_departamento, B.nombre\_departamento

**FROM** Empleados **AS** A

**LEFT** **JOIN** Departamentos **AS** B

**ON** A.id\_departamento **=** B.id\_departamento

Lo primero que te llamará la atención son los alias es decir eso que aparece como as A o as B detrás del nombre de las tablas. Sí podemos ponerle alias a las tablas para luego referirnos a sus campos con la **notación <alias>.<campo>**. Ojo podemos prescindir de los alias y **usar <nombre\_tabla>.<campo>** pero cuando los nombres son largos se suelen utilizar alias.

Superado el tema del alias, ves que la tabla de la izquierda es la primera que aparece (la que está detrás del FROM) y la de la derecha la que aparece tras la palabra JOIN.

Finalmente se usa ON para anticipar la parte de las claves que deben coincidir usando la notación <alias>.<campo> (o <nombre tabla>.<campo>)

**Right (outer) join**

1. De igual manera, aunque quizás más contraintuitivo, podemos pegar a cada departamento los datos de los empleados que tengan igual id\_departamento y NaN si no hay empleados en ese departamento. Quedaría así:

Interfaz de usuario gráfica, Tabla

Descripción generada automáticamente con confianza media

Antes aparecían todos los elementos de la izquierda y no necesariamente todos los elementos de la derecha y ahora es al contrario aparecen todos los elementos/filas de la tabla de la derecha y no todos los de la izquierda (no está Luis). Las tablas en ambos casos se completan con NaN/Nulls para los valores faltantes.

Es decir, aquí manda la tabla de la derecha. Es por eso que se denomina Right (outter) join.

Su sintaxis.

**SELECT** A.id\_empleado, A.nombre, A.id\_departamento, B.nombre\_departamento

**FROM** Empleados **AS** A

**RIGHT** **JOIN** Departamentos **AS** B

**ON** A.id\_departamento **=** B.id\_departamento

Y las secciones y alias tienen la misma explicación que en el caso anterior.

**Inner Join**

1. Un tercer caso, quizá más intuitivo que los dos anteriores, es quedarme solo con los datos de una y otra tabla que tienen id en las dos. Es decir la intersección por id\_departamento en este caso, y quedaría algo así como:

Interfaz de usuario gráfica, Tabla

Descripción generada automáticamente

Fíjate que Luis que no tiene departamento o su id\_departamento no aparece en Departamentos y Recursos Humanos el departamento que no aparece en los id\_departamento de los empleados, se quedan fuera de la tabla resultante.

A esta intersección se le llama INNER JOIN y su sintaxis es:

**SELECT** A.id\_empleado,A.nombre,A.id\_departamento, B.nombre\_departamento

**FROM** Empleados **AS** A

**INNER** **JOIN** Departamentos **AS** B

**ON** A.id\_departamento **=** B.id\_departamento

**Full Outer Join**

1. Igual que consideramos sólo las coincidencias podemos considerar unas y otras y hacer la unión de las salidas del left join y del right join:

Interfaz de usuario gráfica

Descripción generada automáticamente

En esta tabla ya aparecen los empleados sin departamento y los departamentos sin empleados eso sí con sus correspondientes campos a NaN

La sintaxis es similar a las anteriores, pero ahora usan FULL OUTTER JOIN:

**SELECT** A.id\_empleado,A.nombre,A.id\_departamento, B.nombre\_departamento

**FROM** Empleados **AS** A

**FULL** **OUTER** **JOIN** Departamentos **AS** B

**ON** A.id\_departamento **=** B.id\_departamento

**Otros Joins**

Existen otros joins como el producto cartesiano y los joins que se combinan con cláusulas where pero no los vamos a ver [aquí](https://www.dataquest.io/blog/sql-joins-tutorial/) y [aquí](https://www.freecodecamp.org/news/sql-joins-tutorial/) tienes un par de tutoriales donde se amplía lo que hemos visto.

**Lo importante**

Lo importante es lo que uno quiera hacer con las tablas:

* Que tengo una tabla que "manda" y quiero completarla en lo posible y con nulos si no hay datos en las otras tablas de las que quiero sacar información: Necesitas un LEFT o un RIGHT JOIN dependiendo del orden en el que pongas la tabla "directora" (la que quieres completar), si la pones en el FROM, entonces LEFT JOIN si no, RIGHT JOIN
* Que quieres quedarte solo con las filas que no vayan a tener nulos: INNER JOIN (sólo te quedaras con las filas o información de una y otra tabla que coincidan, las que producen nulos se quitan)
* Que quieres mezclar las dos tablas al completo independientemente de donde se generen Nulos: FULL JOIN.