**LO BÁSICO https://sqlzoo.net/wiki/SQL\_Tutorial**

¿Qué consulta ejecutarías para obtener **todos los usuarios**? SELECT \* FROM users;

¿Qué consulta ejecutarías para obtener **solo los nombres de todos los usuarios**? SELECT first\_name FROM users;

¿Qué consulta ejecutarías para obtener **solo los nombres y apellidos de todos los usuarios**? SELECT first\_name, last\_name FROM users;

¿Qué consulta ejecutarías para obtener **solo los nombres** de los usuarios con id de 2? SELECT first\_name FROM users WHERE id = 2;

¿Qué consulta ejecutarías para obtener **solo los apellidos** de los usuarios con id de 2 y 3? SELECT last\_name FROM users WHERE id = 2 OR id = 3;

SELECT last\_name FROM users WHERE id IN (1,2,3);

¿Qué consulta ejecutarías para obtener **todos los usuarios con id mayor que 2**? SELECT \* FROM users WHERE id > 2;

¿Qué consulta ejecutarías para obtener **todos los usuarios con id menor o igual a 3**? SELECT \* FROM users WHERE <= 3;

¿Qué consulta ejecutarías para obtener **todos los usuarios con nombres que terminen en e**? SELECT \* FROM users WHERE first\_name LIKE "%e";

EXTRA ¿Qué consulta ejecutarías para obtener **todos los países con nombres que terminen en a y l**? SELECT name FROM world WHERE name LIKE '%a' OR name LIKE '%l';

EXTRA ¿Qué consulta ejecutarías para **obtener el nombre y la población de todos los países que empiecen con Al**? SELECT name, population FROM world WHERE name LIKE "Al%"

¿Qué consulta ejecutarías para obtener **todos los usuarios con nombres que no comiencen en K**? SELECT \* FROM users WHERE first\_name NOT LIKE "K%";

EXTRA ¿Qué consulta ejecutarías para obtener **el nombre de todos los países que tengan una x**? SELECT name FROM world WHERE name LIKE '%X%'

EXTRA Encuentra el pais cuyo nombre empieza con C y termina con IA SELECT name FROM world WHERE name LIKE 'C%'AND name LIKE '%IA'

EXTRA Encuentra el país cuyo nombre tiene 3 a: SELECT name FROM world WHERE name LIKE '%a%a%a%'; INFO This query will retrieve the names of countries that have three or more occurrences of the letter 'a' in their names. The "%" wildcard is used to match any characters (including none) before, between, and after the 'a' letters.

EXTRA India and Angola have an n as the second character. You can use the underscore as a single character wildcard. SELECT name FROM world WHERE name LIKE '\_n%' ORDER BY name

EXTRA Find the countries that have two "o" characters separated by two others SELECT name FROM world WHERE name LIKE '\_%o\_\_o%';

EXTRA Find the countries that have exactly four characters

SELECT name FROM world WHERE name LIKE '\_\_\_\_'

**El % representa la guía: si pongo la letra antes del % significa que “comience con” y si pongo después del % significa “que termine con”**

¿Qué consulta ejecutarías para obtener **todos los usuarios, con los usuarios más jóvenes en la parte superior de la tabla**? SELECT \* FROM users ORDER BY birthday DESC;

¿Qué consulta ejecutarías para obtener todos los usuarios, con los usuarios de mayor edad en la parte superior de la tabla? SELECT \* FROM users ORDER BY birthday ASC;

¿Qué consulta ejecutarías para obtener **todos los usuarios cuyo nombre termine en “e”, con los usuarios más jóvenes en la parte superior de la tabla?** SELECT \* FROM users WHERE first\_name LIKE "%e" ORDER BY birthday DESC;

¿Qué consulta ejecutarías para obtener **solo los nombres de todos los usuarios en orden alfabético?**

SELECT first\_name FROM users ORDER BY first\_name;

**El valor predeterminado para ORDER BY es ASC, por lo que podemos omitir esa parte si queremos que el ordenamiento sea ascendente.**

¿Qué consulta ejecutarías para obtener **los primeros 3 usuarios?** SELECT \* FROM users LIMIT 3;

¿Qué consulta ejecutarías para obtener **los registros de usuario 3-5?** SELECT \* FROM users LIMIT 3 OFFSET 2;

Es decir, que quite los 3 primeros pero que solo muestre los dos siguientes

También se pueden combinar el limit y offset así: SELECT \* FROM users LIMIT 2,3;

También se puede usar **BETWEEN** como parámetro SELECT \* FROM users WHERE id BETWEEN 2 AND 3;

EXTRA One or the other (but not both). Exclusive OR (XOR). Show the countries that are big by area (more than 3 million) or big by population (more than 250 million) but not both. Show name, population and area.

SELECT name, population, area FROM world WHERE (area > 3000000 OR population > 250000000) AND NOT (area > 3000000 AND population > 250000000);

EXTRA Show the name and population in millions and the GDP in billions for the countries of the continent 'South America'. Use the ROUND function to show the values to two decimal places. For South America show population in millions and GDP in billions both to 2 decimal places. Millions and billions.

SELECT name, ROUND(population / 1000000, 2) AS population\_in\_millions, ROUND(gdp / 1000000000, 2) AS gdp\_in\_billions FROM world WHERE continent = 'South America';

Edición de contenido

**INSERT**

INSERT INTO nombre\_de\_la\_tabla (nombre\_columna1, nombre\_columna2) VALUES('valor\_columna1', 'valor\_columna2');

**UPDATE**

UPDATE nombre\_de\_la\_tabla SET nombre\_columna1 = 'un\_valor', nombre\_columna2 = 'otro\_valor', ... WHERE condición;

**IMPORTANTE: si la condición WHERE no se agrega a la instrucción UPDATE, los cambios se aplicarán a todos los registros de la tabla.**

UPDATE users SET first\_name = 'MARIA', last\_name = 'IVERSON' WHERE id = 3;

También haciendo click derecho puede aparecer las opciones para el update, debo recordar escribir bien y después indicar el comando para que me muestre todos los datos, pero actualizados.

DELETE

Si recibes un error con respecto a las ACTUALIZACIONES DE SEGURIDAD SQL, ejecuta el siguiente comando para que MySQL Workbench sepa que sabes lo que estás haciendo y quieres BORRAR cosas de la base de datos.

SET SQL\_SAFE\_UPDATES = 0; escribir y apretar el rayito

El patrón de comando SQL para eliminar/remover registros es el siguiente:

DELETE FROM nombre\_de\_la\_tabla WHERE condición;

**IMPORTANTE: si la condición WHERE no se agrega a la instrucción DELETE, se borrarán todos los registros de la tabla.**

CONSULTAS

|  |  |
| --- | --- |
| Tipos de datos para funciones de texto | (VARCHAR, TEXT, CHAR, etc.) |
| Tipos de datos para funciones numéricas | (INT, BIGINT, FLOAT, etc.) |
| Tipos de datos para funciones de fecha y hora | (DATETIME) |
| **SELECT FUNCIÓN (columna) FROM nombre\_tabla** | |

A screenshot of a computer

Description automatically generated

A table with text and numbers

Description automatically generated with medium confidence

Para ver los diferentes tipos de formato para TIME\_FORMAT() y DATE\_FORMAT(), mira aquí: <https://www.w3schools.com/sql/func_mysql_date_format.asp>

UNIR TABLAS

**Unión: Uno a uno**

SELECT \* FROM customers JOIN addresses ON addresses.id = customers.address\_id;

**Unión: uno a muchos**

SELECT \* FROM orders JOIN customers ON customers.id = orders.customer\_id;

**Unión: muchos a muchos**

SELECT \* FROM orders JOIN items\_orders ON orders.id = items\_orders.order\_id JOIN items ON items.id = items\_orders.item\_id;

**Agrupar resultados**

Con GROUP BY, agruparemos varias filas, realizando una función para combinar los valores de esas filas. Debido a que esto da como resultado un único resultado para el grupo, combinará esas filas agrupadas en una sola fila resultante

SELECT sites.id, COUNT(clients.id) AS client\_count FROM sites JOIN clients ON sites.id = clients.id GROUP BY sites.id;

|  |  |
| --- | --- |
| SELECT sites.id: | This specifies the column you want to retrieve from the "sites" table. |
| COUNT(clients.id) AS client\_count: | This uses the COUNT aggregate function to count the number of clients for each site. The result of this calculation is given an alias "client\_count" for readability. |
| FROM sites: | This indicates that you're selecting data from the "sites" table. |
| JOIN clients ON sites.id = clients.id: | This performs an inner join between the "sites" and "clients" tables based on the matching "id" columns. |
| GROUP BY sites.id: | This groups the results by the "id" column from the "sites" table, allowing you to aggregate the count of clients for each site separately. |

**Funciones de agregación (los más usados):**

A table with black text

Description automatically generated

LEFT JOIN

Es una función que define la prioridad de la unión. Left join indica que existe PRIORIDAD en la unión de la tabla de la izquierda que la de la derecha. En el caso de la tabla de tuits y users del que habla el profe, los usuarios con tuits aparecen en la tabla, pero el nuevo usuario que el creó, no tiene tuits entonces no se consideró en esa agrupación. Si aun así yo quiero saber quienes son los usuarios con 0 tuits, pongo left join para que le dé prioridad a la tabla con el nombre de los usuarios. La sintaxis es así: SELECT \* FROM users ***u*** LEFT JOIN tweets ***t*** on u.id = t.user\_id

***u y t representan alias que le doy al término previo, funciona así directamente CON LA FUENTE DE TW***

1. ¿Qué consulta ejecutarías para obtener todos los tweets del ID de usuario 1?

SELECT \* FROM users LEFT JOIN tweets ON users.id = tweets.user\_id WHERE users.id = 1;

Puedes simplemente tomar los tweets de la siguiente manera:

SELECT tweets.tweet FROM users LEFT JOIN tweets ON users.id = tweets.user\_id WHERE users.id = 1;

O cambia el nombre de la columna de salida que quieres como kobe\_tweets modificando la sentencia para que tenga el siguiente aspecto:

SELECT tweets.tweet as kobe\_tweets FROM users LEFT JOIN tweets ON users.id = tweets.user\_id WHERE users.id = 1;

2. ¿Qué consulta devolvería todos los tweets que el usuario con id 2 ha marcado como favoritos?

SELECT \* FROM users LEFT JOIN faves ON users.id = faves.user\_id LEFT JOIN tweets ON faves.tweet\_id = tweets.id WHERE users.id = 2

3. ¿Qué consulta ejecutarías para obtener todos los tweets que ese usuario con id 2, o usuario con id 1, ha marcado como favoritos?

SELECT tweets.\* FROM users LEFT JOIN faves ON users.id = faves.user\_id LEFT JOIN tweets ON faves.tweet\_id = tweets.id WHERE users.id IN (2,3);

4. ¿Qué consulta ejecutarías para obtener todos los usuarios que siguen al usuario con id 1?

**autounión**. relación de muchos a muchos, usando la misma tabla dos veces. Un usuario puede: tener muchos seguidores y seguir a muchos, entonces es una relación n a n pero las dos filas que se relacionan son del mismo tipo. La palabra clave para recordar es la palabra "AS" porque nos permite unir la misma tabla dos veces al proporcionarle a SQL otra variable para hacer referencia a la tabla que se está uniendo nuevamente.

SELECT users.first\_name as followed, users2.first\_name as follower FROM users LEFT JOIN follows ON users.id = follows.followed\_id LEFT JOIN users as users2 ON users2.id = follows.follower\_id WHERE users.id = 1;

|  |  |
| --- | --- |
| LEFT JOIN follows ON users.id = follows.followed\_id: | Esto realiza una operación LEFT JOIN entre las tablas users y follows, uniéndolas mediante el ID del usuario (id) y el ID del usuario seguido (followed\_id). |
| LEFT JOIN users as users2 ON users2.id = follows.follower\_id: | Esto realiza otra operación LEFT JOIN, pero esta vez entre la tabla follows y la tabla users2, usando el ID del seguidor (follower\_id) y el ID del usuario (id). |
| WHERE users.id = 1;: | Esto filtra los resultados para incluir solo datos para el usuario con ID 1. |

5. ¿Qué consulta ejecutarías para obtener todos los usuarios que no siguen al usuario con id 2?

SELECT \* FROM users WHERE users.id NOT IN (SELECT follower\_id FROM follows WHERE followed\_id = 2) AND users.id != 2;

|  |  |
| --- | --- |
| WHERE users.id NOT IN (SELECT follower\_id FROM follows WHERE followed\_id = 2): | Aquí se realiza una condición para filtrar los resultados. La subconsulta (SELECT follower\_id FROM follows WHERE followed\_id = 2) obtiene los IDs de los seguidores para el usuario con ID 2. La cláusula NOT IN filtra los resultados para excluir aquellos registros cuyos IDs de usuario estén presentes en la subconsulta. |
| AND users.id != 2;: | Esta parte de la condición filtra aún más los resultados para excluir aquellos registros cuyo ID de usuario sea igual a 2. |

Podemos ejecutar funciones en columnas específicas y, a menudo, se empareja con la sentencia GROUP BY. Este sería para saber cuantos usuarios le siguen.

SELECT users.first\_name as user, COUNT(users2.first\_name) as follower\_count

FROM users

LEFT JOIN follows ON users.id = follows.followed\_id

LEFT JOIN users as users2 ON users2.id = follows.follower\_id

WHERE users.id = 1

GROUP BY users.id;

|  |  |
| --- | --- |
| SELECT users.first\_name as user, COUNT(users2.first\_name) as follower\_count: | This part of the query specifies that you want to select two columns: the first name of the user (renamed as "user") and the count of the first names of users who are followers (renamed as "follower\_count"). |

¿Qué consulta ejecutarías para obtener todos los tweets de cada usuario, aunque no hayan publicado tuits?

A screenshot of a message

Description automatically generated

Para resumir, JOIN solo incluirá la intersección de las dos tablas (o sea solo los users que hayan publicado tuits y va a ignorar los que no publicaron), mientras que LEFT JOIN incluirá todos los registros de la primera tabla, más los registros de la segunda tabla que correspondan. Esta es la razón por la que JOIN a veces se denomina INNER JOIN, mientras que todas las demás uniones (incluida LEFT JOIN) se denominan OUTER JOIN.