MySQL CheatSheet - Marcos Zabala - DAM

Primeros pasos:

(Aviso: Todos los comandos usados pueden ser escritos en mayúscula como en minúscula, aunque se recomienda hacerlo en mayúscula por organización.)

Después de iniciar el acceso a terminal escribiremos:

mysql -u root -p /Enter/

Acto seguido introduciremos la contraseña (contraseña que asignaste en el puerto con anterioridad) en caso de no tener, tan solo pulsa de nuevo /Enter/.

En caso de querer abandonar el monitor, usa el comando quit.

El comando show databases permite visualizar las bases de datos actualmente activas.

show databases;

Para tener acceso a las databases usaremos el comando use.

use (nombre de la tabla)

(No es necesario el uso de ; para este ni para el comando quit).

En caso de no tener tablas o que desees crear una nueva, usarás el comando

create database (Nombre de la tabla);

lo cual creará una tabla vacía a la que podrás acceder con use.
(Es recomendable que antes de crear una tabla nueva se use el comando

drop database if exists (nombre de la tabla);

para borrarla en caso de que ya existiera.)

Una vez creada nuestra base de datos necesitamos tablas en la que ingresar los datos. Para hacer esto usaremos el comando:

create table (nombre de la table) ([Aquí insertamos los tipo de valor de la tabla y sus respectivos nombres]);

Ejem:

```
CREATE TABLE providers (
id INT AUTO_INCREMENT PRIMARY KEY,
prov_name CHAR(200) UNIQUE,
category_id INT UNSIGNED,
total DOUBLE(10,2) UNSIGNED,
date_of_register DATETIME,
description TEXT);
```

Para ver las tablas de nuestra database usamos el comando:

SHOW TABLES;

En caso de querer borrar una tabla introducimos:

DROP (nombre de la tabla);

Bien, ahora veremos como alterar nuestra tabla en casa de querer agregar o eliminar columnas.

Para agregar usaremos:

ALTER TABLE (Nombre de la tabla) ADD COLUMN (Datos a introducir);

En caso de eliminar una tabla:

ALTER TABLE (Nombre de la tabla) DROP COLUMN (Nombre de la columna);

Una vez hecho esto tendremos las columnas deseadas en nuestra tabla, para revisarlas usaremos:

DESCRIBE (nombre de la columna);

Ahora tendremos que insertar datos en las columnas, con lo que usaremos:

INSERT INTO (nombredetabla) [(nombrecol1, nombrecol2,...,nombrecoln)] VALUES(valorcol1,valorcol2,..,valorcoln),..,(valorcol1,valorcol2,...,valorcoln);

En caso de insertar el nombre de las columnas, las columnas no insertadas no variarán mientras que si no se declarará ningún nombre, se tendrán en cuenta todas las columnas en el orden de introducción.

También podremos usar el valor ***DEFAULT*** para usar el valor por defecto de las variables o ***NULL*** en caso de querer dejarlas nulas.

En caso de querer actualizar los datos de la columna usamos el comando ***UPDATE:***

UPDATE nombre_de_la_tabla
SET columna_1 = valor_1, .. , columna_n = valor_n;

Podemos darle especificaciones al comando para variar valores que cumplan una condición equis.

UPDATE nombre_de_la_tabla SET columna_1 = valor_1, .. , columna_n = valor_n WHERE (Condicion lógica);

En caso de querer variar un número equis de valores usamos el comando ***LIMIT:***

UPDATE nombre_de_la_tabla SET columna_1 = valor_1, .., columna_n = valor_n WHERE condición_lógica ORDER BY columna (ASC|DESC) LIMIT n;

Y en caso de querer mostrarlos en cierto orden, usamos el comando ***ORDER BY*** como ya vimos en el ejemplo de arriba.

Para borrar datos usaremos el comando ***DELETE***, al cuál se le pueden aplicar todos los comandos anteriores:

DELETE FROM nombre_de_tabla [WHERE condición_lógica] [ORDER BY columna] [LIMIT n] ;

Ahora veremos la consulta de datos en nuestras tablas, para seleccionar todos los datos de una columna usaremos:

SELECT * FROM (nombre de la columna);
[El * representa todos los datos de la columna]

Al igual que en los ***UPDATE/DELETE*** podemos usar los comandos ***ORDER BY / WHERE / LIMIT*** de forma similar:

SELECT * FROM (nombre de la columna) LIMIT 3;

SELECT * FROM (nombre de la columna) ORDER BY (nombre del dato);

SELECT * FROM (nombre de la columna) WHERE (condición lógica);

Mezclandolos todos:

SELECT * FROM (nombre de la columna) WHERE (condición lógica) ORDER BY (nombre del dato) DESC LIMIT 3;

Propiedades no mencionadas de ***LIMIT/ORDER BY:***

En ***LIMIT*** si queremos mostrar 5 filas después del décimo registro, el comando sería algo así:

SELECT * FROM (nombre de la columna) LIMIT 10, 5;

En el caso de ***ORDER BY*** se puede colocar el comando ***[ASC]/[DESC]*** para alterar el orden en que se muestran los datos.

Para el comando ***WHERE*** tenemos todos los operadores lógicos como [y,o,<,>,=,!=,<=,>=, entre, y las negaciones de estos con ***NOT***].

Otro comando de ***WHERE*** a tener en cuenta es el comando ***LIKE*** el cual se expresa:

<expresión> LIKE <patrón> [ESCAPE 'carácter_escape']

Al comando ***WHERE*** también se le pueden aplicar operaciones matemáticas.

La cláusula GROUP BY se usa para agrupar filas que tienen los mismos valores. Se usa en la sentencia SELECT. Opcionalmente se usa en conjunto con funciones para producir resúmenes de datos de la base de datos.

Las consultas que contienen la cláusula GROUP BY se llaman consultas agrupadas y sólo devuelve una fila por cada ítem agrupado.

SELECT statements GROUP BY column_name1[,column_name2,...] [HAVING condition];

Ejemplos:

Supongamos que queremos saber todos los release_year para la category_id 8.

SELECT * FROM `movies` GROUP BY `category_id`,`year_released` HAVING `category_id`= 8;

Supongamos que queremos el total de males y females en nuestra base de datos. Podríamos utilizar la siguiente consulta.

SELECT `gender`,COUNT(`membership_number`) FROM `members` GROUP BY `gender`;

Una subconsulta consiste en una instrucción ***SELECT*** anidada dentro de otra instrucción (pueden ser ***HAVING*** o ***WHERE***). Estas retornan un valor único.

La estructura de una subconsulta es la siguiente:

SELECT lista_de_columnas FROM lista_de_tablas WHERE <expresión> <condición> (SELECT nombre_columna FROM lista_de_tablas WHERE condición);

En el caso de que busquemos encontrar un dato específico, pero para los conjuntos de datos usaremos:

SELECT lista_de_columnas FROM lista_de_tablas WHERE <expresión> IN (SELECT nombre_columna FROM lista_de_tablas WHERE condición);

A su vez, podemos hacer subconsultas que aniden otras subconsultas o usar los comandos ***EXISTS/ANY/ALL*** que devuelven si la variable existe, algunos valores que concuerden con la condición o todos los valores.

Los ***JOIN*** son usados en una sentencia SQL para recuperar datos de varias tablas al mismo tiempo. Estas tablas tienen que estar relacionadas de alguna forma.

Pueden ser agrupados en 6 tipos distintos:

LEFT JOIN: Coge todos los resultados de la tabla izquierda, incluyendo los que coinciden con la otra tabla.

SELECT usuarios.username, juegos.juegoname FROM usuarios
LEFT JOIN juegousuario ON usuarios.ID = juegousuario.ID_usuario
LEFT JOIN juegos ON juegousuario.ID_juego = juegos.ID

RIGHT JOIN: Coge todos los resultados de la tabla derecha, incluyendo los que coinciden con la otra tabla.

SELECT usuarios.username, juegos.juegoname FROM usuarios
RIGHT JOIN juegousuario ON usuarios.ID = juegousuario.ID_usuario
RIGHT JOIN juegos ON juegousuario.ID_juego = juegos.ID

INNER JOIN: Coge solo los resultados que coinciden en ambas tablas.

SELECT usuarios.username, juegos.juegoname FROM usuarios
INNER JOIN juegousuario ON usuarios.ID = juegousuario.ID_usuario
INNER JOIN juegos ON juegousuario.ID_juego = juegos.ID

FULL OUTER JOIN: Coge todos los resultados de ambas tablas.

SELECT usuarios.username, juegos.juegoname FROM usuarios
LEFT JOIN juegousuario ON usuarios.ID = juegousuario.ID_usuario
LEFT JOIN juegos ON juegousuario.ID_juego = juegos.ID
UNION SELECT usuarios.username, juegos.juegoname FROM usuarios
RIGHT JOIN juegousuario ON usuarios.ID = juegousuario.ID_usuario
RIGHT JOIN juegos ON juegousuario.ID_juego = juegos.ID

LEFT JOIN (IF NULL): Coge todos los resultados de la tabla izquierda, sin incluir los que coinciden con la otra tabla.

RIGHT JOIN (IF NULL): Coge todos los resultados de la tabla derecha, sin incluir los que coinciden con la otra tabla.

Los ***IF NULL*** son la negación de los right/left join.