Verónica Lara de la Torre

Comandos mysql.

(Las partes en gris son modificables en los comandos)

mysql -u [root] **-p** → acceder a mysql con contraseña.

show databases; → muestra una lista con todas las bases de datos disponibles.

use [nombre database]**;** → accedes a una base de datos en concreto.
ej.: *use empleado;*

create database [nombre]**;** → creas una base de datos.
ej.: *create database prueba;*

drop database if exists [nombre]**;** → compruebas si una base de datos ya existe.
ej.: *drop database if exists prueba;*

drop database [nombre]**;** → eliminas una base de datos en concreto.
ej.: *drop database prueba;*

select database; → te muestra la base de datos en la que trabajas.

create table [nombre] **(** id [NULL/NOT NULL] [AUTO_INCREMENT] [PRIMARY KEY] [UNIQUE] [DEFAULT] , other_column type_ data … **) ENGINE INNODB;** → estructura para crear una tabla con sus atributos.
ej.:

```
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) ENGINE INNODB;
```

SELECT * FROM [nombre] → muestra el contenido de una tabla.
ej.: *select * from departamentos;*

INSERT INTO [nombre] VALUES ([datos]); → inserta datos en una tabla en concreto.

ej.:

INSERT INTO citas_enero VALUES(1,Una cita','2008-06-24','21:00');

INSERT INTO enteros VALUES(1);

*Se puede insertar registros múltiples:

ej.:

INSERT INTO employees (full_name, identity_doc) VALUES ('Roger Marcano','20707888'), ('Pedro González', '18235998'), ('Juan Magdaleno', '24588225');

SHOW TABLES; → muestra la tabla en forma.

DESCRIBE [nombre]; → muestra la estructura de la tabla.

ej.: *describe empleados;*

DROP TABLE; → elimina una tabla;

DROP TABLE [nombre]; → elimina una tabla, sus datos e índices asociados;

ej.: *drop table películas;*

ALTER TABLE [nombre] ADD COLUMN [nombre columna] VARCHAR (255); → agrega una nueva columna a una tabla.

ej.: *ALTER TABLE país ADD COLUMN ciudades VARCHAR (255);*

* Se puede insertar un campo con ADD COLUMN y una opción de posición. Hay 3 opciones FIRST, AFTER o insertar al final de la tabla que es la predeterminada.

ej.: ALTER TABLE país ADD COLUMN pueblos VARCHAR (255) AFTER ciudades;

ALTER TABLE [nombre] DROP COLUMN [nombre]; → elimina una columna.

ej.: *ALTER TABLE país DROP COLUMN ciudades;*

ALTER TABLE [nombre] CHANGE COLUMN [nombre antiguo] [nombre nuevo] CHAR (200); → renombrar una tabla.

ej.: *ALTER TABLE país CHANGE COLUMN ciudades ciudad CHAR(200);*

UPDATE [nombre] SET [nombre] = [...]; → actualizar datos de una tabla;

ej.: *UPDATE producto SET precio = 1000;*

*Para evitar que se modifique toda una tabla al completo se usa la clausula WHERE.

ej.: *UPDATE producto SET precio = 1000 WHERE [condición lógica];*

CONSULTAS.

La sintaxis básica es SELECT (contenido separado por comas) FROM nombre de las tablas.

SELECT * FROM [nombre]; → muestra una tabla.

SELECT * FROM [nombre] LIMIT [número]; → muestra los datos de la tabla con un límite determinado.

ej.: *SELECT * FROM empleados LIMIT 5;*

SELECT [campo], [campo] FROM [nombre]; → muestra los datos de dos campos de una tabla determinada.

ej.: *SELECT nombre, apellido FROM empleados;*

SELECT * FROM [nombre] ORDER BY [campo]; → muestra los datos de la tabla ordenados por el campo seleccionado. Se puede ordenar por orden ascendente o descendente añadiendo [ASC|DESC].

ej.: *SELECT * FROM empleado ORDER BY nombre;*

* La cláusula ORDER BY va después de WHERE o GROUP BY, si ninguna de éstas son necesarias se coloca después de FROM.

SELECT * FROM [nombre] WHERE [condición]; → muestra los datos que cumplan la condición.

ej.: SELECT * FROM artículos WHERE precio < 10;

Operadores matemáticos:

suma +; resta -; multiplicación *; división /; igual =;

Operadores de magnitud:

Operador	Descripción
<=	Menor o igual
<	Menor
>	Mayor
>=	Mayor o igual

SELECT [campo] **FROM** [nombre] **WHERE** [campo] **BETWEEN** [minimo] **AND** [máximo]; → comprueba si los datos están entre dos valores, se puede hacer el caso contrario usando NOT BETWEEN.

ej.: *SELECT precio FROM producto WHERE precio BETWEEN 10 AND 100;*

SELECT * FROM [nombre] WHERE [campo] LIKE [condición]; → hace comparaciones entre cadenas. Con LIKe se pueden usar 3 operadores '_' (representa un carácter), 'a' (representa una letra que definamos) o '%' (representa un conjunto de caracteres). También podemos usar en el caso contrario la cláusula NOT LIKE

ej.: *SELECT * FROM producto WHERE nombre LIKE '%n';*

SELECT [campo] **FROM** [nombre] **GROUP BY** [campo]; → muestra los datos agrupados por filas que tienen los mismos valores. Sólo devuelve una fila por cada item agrupado.

ej.: *SELECT gender FROM members GROUP BY gender;*

SELECT [campo], **COUNT(*) FROM** [nombre] **GROUP BY** [campo]; → cuenta el total de los valores en concreto de una tabla.

ej.: *SELECT gender, COUNT(membership_number) FROM members GROUP BY gender;*

SELECT * FROM [nombre] **GROUP BY** [campo] **HAVING** [condición]; → restringe los datos de la consulta.

ej.: *SELECT * FROM películas GROUP BY categoría, año, HAVING categoría =5;*

SUBCONSULTAS.

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

SELECT [campo] **FROM** [nombre] **WHERE** [expresión] [condición] **(SELECT** [campo] **FROM** [nombre] **WHERE** [condición]**);** → estructura básica de una subconsulta.

ej.: SELECT dep_nom FROM carga_f WHERE eci = (SELECT ci FROM empleado WHERE nombre= 'Humberto' AND apellido = 'Pons');

SELECT [campo] **FROM [nombre] WHERE** [expresión] **IN (SELECT** [campo] **FROM** [nombre] **WHERE** [condición]**;** → recupera sólo los registros de la consulta principal con un valor igual. Se usa NOT IN para realizar la función contraria.

ej.: *SELECT nombre, apellido FROM empleado WHERE ci IN (SELECT eci FROM carga_f WHERE YEAR (fecha_n) BETWEEN '1980' AND '1999');*

SELECT nombre, apellido FROM empleado WHERE ci NOT IN (SELECT eci FROM carga_f WHERE YEAR(fecha_n) BETWEEN '1980' AND '1990');

SELECT [campo] **FROM** [nombre] **WHERE** [expresión] [condición] **ANY (SELECT** [campo] **FROM** [nombre] **WHERE** [condición]**;** → recupera registros de la consulta que cumplan con la comparación de algunos registros de la subconsulta.

ej.: *SELECT dnombre FROM departamento WHERE dnumero = ANY (SELECT dno FROM empleado WHERE salario = '2500');*

SELECT [campo] **FROM** [nombre] **WHERE** [expresión] **IN (SELECT** [campo] **FROM** [nombre] **WHERE** [expresión] **IN (SELECT** [campo] **FROM** [nombre] **WHERE** [condición]**));** → estructura de una subconsulta anidada, que son las que tienen más de dos niveles de consulta.

ej.: *SELECT nombre, apellido FROM empleado WHERE ci IN(SELECT eci FROM trabaja_en WHERE pno IN (SELECT pnumero FROM proyecto WHERE nombre = 'Beneficios'));*

Join.

Se usan para recuperar datos de varias tablas al mismo tiempo , las tablas deben estar relacionada de algún modo.

SELECT [campo] **FROM** [nombre] **INNER JOIN** [nombre] **ON** [campo] **=** `[campo] **WHERE** [condición]**;** → devuelve sólo los datos que estén disponibles en todas las tablas a la vez. Son los estándar

ej.: *SELECT dep_nom, relacion FROM carga_f INNER JOIN empleado ON cargas_f.eci = empleado.ci where nombre = 'Elena' AND apellido = 'Tapia';*

SELECT * FROM [nombre] **LEFT JOIN** [nombre] **ON** [campo] **=** [campo]**;** → devuelve los resultados que coincidan en la primera tabla con los datos que tenga de la segunda.

ej.: *SELECT * FROM empleados.a LEFT JOIN departamentos.b ON a.id_departamento = b.id;*

SELECT * FROM [nombre] **RIGHT JOIN** [nombre] **ON** [campo] **=** [campo] **WHERE** [condición]**;** → devuelve todos los datos de la tabla con la que se relaciona la anterior.

ej.: SELECT * FROM empleados.a RIGHT JOIN departamentos.b ON a.id_ departamento = b.id WHERE a.hipoteca = 5;

*Éstos son los más utilizados, los tipos de join son: inner join, left join, right join, full outer join que se cambia por union.