Guía Rápida de Comandos SQL

**1. CREACIÓN Y GESTIÓN DE BASES DE DATOS**

* - CREATE DATABASE [IF NOT EXISTS] nombre\_bbdd [CHARACTER SET charset] [COLLATE collate] - Crea una nueva base de datos.
* - DROP DATABASE [IF EXISTS] nombre\_bbdd - Elimina una base de datos existente.
* - ALTER DATABASE nombre\_bbdd ... - Modifica las propiedades de una base de datos.
* - USE nombre\_bbdd - Selecciona una base de datos para trabajar con ella.

**Ejemplo:**

drop database if exists 03\_alquiler\_coches;

create database 03\_alquiler\_coches;

use 03\_alquiler\_coches;

**2. GESTIÓN DE TABLAS**

* - CREATE TABLE nombre\_tabla (definición columnas y restricciones) - Crea una nueva tabla.

**La estructura básica para crear una tabla es:**

**create table** marcas(

id\_marca int AUTO\_INCREMENT,

marca varchar(30) not null unique,

primary key (id\_marca)

);

**create table** modelos(

id\_modelo int AUTO\_INCREMENT,

modelo varchar(30) not null unique,

fk\_marca int not null,

primary key(id\_modelo),

foreign key (fk\_marca) references marcas(id\_marca)

);

**create table** coches\_por\_reservas(

fk\_coche int,

fk\_reserva int,

litros int not null,

km\_inicio int not null,

km\_fin int,

primary key (fk\_coche, fk\_reserva),

foreign key (fk\_coche) references coches(id\_coche),

foreign key (fk\_reserva) references reservas(id\_reserva)

);

**2.1 Claves Primarias (PRIMARY KEY)**

La clave primaria permite identificar de forma única cada fila de la tabla.

**Ejemplo de clave primaria compuesta:**

create table coches\_por\_reservas(

fk\_coche int,

fk\_reserva int,

litros int not null,

km\_inicio int not null,

km\_fin int,

**primary key (fk\_coche, fk\_reserva)**, (ejemplo de una primary key combinada)

**constraint**(opcional si no el sistema lo pone por defecto) cpr\_coches foreign key (fk\_coche) references coches(id\_coche),

**foreign key** (nombre de la fk)(fk\_reserva**) references**(tabla referenciada)reservas(id\_reserva));

Ejemplo:

create table coches\_por\_reservas(

fk\_coche int,

fk\_reserva int,

litros int not null,

km\_inicio int not null,

km\_fin int,

primary key (fk\_coche, fk\_reserva),

**constraint** cpr\_coches **foreign key** (fk\_coche) **references** coches(id\_coche),

**constraint** cpr\_reservas **foreign key** (fk\_reserva) **references** reservas(id\_reserva)

);

**2.2. AUTO\_INCREMENT**

Se usa para generar automáticamente un valor único incremental para una columna, normalmente una clave primaria.  
  
Ejemplo:  
 CREATE TABLE curso (  
 codigo INT **AUTO\_INCREMENT**,  
 nombre VARCHAR(30),  
 PRIMARY KEY (codigo)  
 );

**2.3 MODIFICACION TABLAS Y ELIMINACION**

* - DROP TABLE [IF EXISTS] nombre\_tabla - Elimina una o varias tablas.
* - ALTER TABLE nombre\_tabla ADD columna TIPO DE DATO(INT,VARCHAR)- Añade una nueva columna.
* - ALTER TABLE nombre\_tabla DROP columna - Elimina una columna.
* - ALTER TABLE nombre\_tabla MODIFY columna NUEVO TIPO DE DATO - Modifica tipo de dato o restricciones.
* - ALTER TABLE nombre\_tabla CHANGE antigua nueva TIPO DE DATO o NUEVO TIPO DE DATO - Renombra y modifica una columna.
* - SHOW TABLES - Muestra todas las tablas de la base de datos.
* - DESCRIBE nombre\_tabla - Muestra la estructura de la tabla.
* - SHOW CREATE TABLE nombre\_tabla - Muestra el comando SQL que crea la tabla.
* ejemplos:

- Agregar columna:  
 ALTER TABLE persona ADD direccion VARCHAR(50);  
  
- Modificar tipo de columna:  
 ALTER TABLE persona MODIFY direccion TEXT;  
  
- Cambiar nombre de columna:  
 ALTER TABLE persona CHANGE direccion direccion\_residencia TEXT;  
  
- Eliminar columna:  
 ALTER TABLE persona DROP direccion\_residencia;

**3. CONSULTAS BÁSICAS (SELECT)**

* - SELECT columnas FROM tabla [WHERE condición] - Consulta datos.
* - SELECT \* FROM tabla - Selecciona todos los campos.
* - SELECT DISTINCT columna FROM tabla - Evita duplicados.
* - ORDER BY columna [ASC|DESC] - Ordena los resultados.
* - LIMIT n OFFSET m - Limita el número de resultados.

**4. FUNCIONES EN CONSULTAS**

* - CONCAT, LOWER, UPPER, SUBSTR - Manipulación de cadenas.

Ejemplos:

**CONCAT**(cadena1, cadena2, ...)/Une cadenas de texto.

SELECT CONCAT('Hola', ' ', 'Mundo'); -- Resultado: 'Hola Mundo'

**LOWER**(cadena)/ Convierte todo el texto a minúsculas.

SELECT LOWER('CHATGPT'); -- Resultado: 'chatgpt'

**UPPER**(cadena)/Convierte todo el texto a mayúsculas.

SELECT UPPER('chatgpt');-- Resultado: 'CHATGPT'

**SUBSTR**(cadena, inicio, longitud)/Devuelve una parte de la cadena.

inicio empieza en 1.

SELECT SUBSTR('Programación', 1, 7); -- Resultado: 'Program'

* - ABS, POW, SQRT, ROUND - Funciones matemáticas.

Ejemplos :

**ABS**(valor)/Valor absoluto (sin signo negativo).

SELECT ABS(-25); -- Resultado: 25

**POW**(base, exponente) o **POWER**(base, exponente)/Potencia de un número.

SELECT POW(2, 3);-- Resultado: 8

**SQRT**(valor)/Raíz cuadrada.

SELECT SQRT(16);-- Resultado: 4

**ROUND**(valor, decimales)/Redondea un número.

SELECT ROUND(3.14159, 2);-- Resultado: 3.14

* - NOW, DATEDIFF, YEAR, MONTH - Funciones de fecha y hora.

Ejemplos :

**NOW**()/Devuelve la fecha y hora actual del sistema.

SELECT NOW(); -- Resultado: '2025-06-21 14:00:00' (ejemplo)

**DATEDIFF**(fecha\_fin, fecha\_inicio)/Diferencia en días entre dos fechas.

SELECT DATEDIFF('2025-07-01', '2025-06-21'); -- Resultado: 10

**YEAR**(fecha)/Extrae el año de una fecha.

SELECT YEAR('1982-12-08'); -- Resultado: 1982

**MONTH**(fecha)/Extrae el número del mes.

SELECT MONTH('1982-12-08'); -- Resultado: 12

* - COUNT, SUM, AVG, MAX, MIN - Funciones de agregación porque trabajan sobre grupos de filas para devolverte un único valor como resultado.Estas funciones se suelen usar con GROUP BY cuando quieres ver los resultados por categoría o grupo.

Ejemplos:

🔢 **COUNT**(columna)/Cuenta cuántas filas tienen un valor en esa columna (no cuenta los NULL).

SELECT COUNT(\*) FROM clientes; -- Cuenta el total de filas en la tabla `clientes`.

➕ **SUM**(columna)/Suma todos los valores de una columna numérica.

SELECT SUM(total\_pedido) FROM pedidos; -- Suma todos los importes de los pedidos.

📈 **AVG**(columna)/Devuelve el promedio (media) de los valores numéricos.

SELECT AVG(precio) FROM productos; -- Devuelve el precio medio de los productos.

🔝 **MAX**(columna)/Devuelve el valor más alto de la columna.

SELECT MAX(salario) FROM empleados; -- Devuelve el salario más alto.

🔽 **MIN**(columna)/Devuelve el valor más bajo de la columna.

SELECT MIN(salario) FROM empleados; -- Devuelve el salario más bajo.

**5. CONSULTAS AVANZADAS**

* - **GROUP BY** columna Agrupa los resultados por valores iguales en una columna.

Ejemplo:

SELECT departamento, AVG(salario) FROM empleados GROUP BY departamento;

Calcula el salario medio por cada departamento.

* - **HAVING** condición - Filtra los resultados después del GROUP BY. No se puede usar WHERE para filtrar agregados, para eso es HAVING.

Ejemplo:

SELECT departamento, COUNT(\*) AS num\_empleados FROM empleados

GROUP BY departamento

HAVING COUNT(\*) > 5;

* - **INNER JOIN (**es lo mismo que **JOIN )**Devuelve solo las coincidencias (cuando hay datos en ambas tablas).

Ejemplo:

SELECT \* FROM empleados JOIN departamentos

ON empleados.depto\_id = departamentos.id;

Solo muestra empleados que están asignados a un departamento.

* **LEFT JOIN** -Devuelve todos los registros de la tabla izquierda y los que coinciden de la derecha. Si no hay coincidencia, pone NULL.

Ejemplo :

SELECT \* FROM empleados LEFT JOIN departamentos

ON empleados.depto\_id = departamentos.id;

Muestra todos los empleados, incluso los que no tienen departamento.

* **RIGHT JOIN** - Devuelve todos los registros de la tabla derecha y los que coinciden de la izquierda.

Ejemplo:

SELECT \* FROM empleados RIGHT JOIN departamentos

ON empleados.depto\_id = departamentos.id;

Muestra todos los departamentos, aunque no tengan empleados asignados.

**6. SUBCONSULTAS**

* - Subconsulta en WHERE - Filtra según otra consulta.
* - Subconsulta en FROM - Crea tablas temporales.
* - Subconsulta correlacionada - Usa datos de la consulta externa.
* - Operadores: IN, EXISTS, ANY, ALL - Comparaciones avanzadas.

**7. MODIFICACIÓN DE DATOS**

* - INSERT INTO tabla (campos) VALUES (valores) - Inserta datos.
* - UPDATE tabla SET campo=valor WHERE condición - Modifica datos.
* - DELETE FROM tabla WHERE condición - Elimina registros.

**modificacion datos con @autocommit= 0**

Cuando esto ocurre debemos grabar con COMMIT para que los cambios sean guardados.

Si hay que desacer la ultima opcion de usa ROLLBACK

Ejemplo:

select @@autocommit;

**set autocommit = 0 ;**

select \*

from productos where idproducto = 23;

update productos set prod\_precio = 10 where idproducto = 23;

**rollback;**

update productos set prod\_precio = 25 where idproducto = 23;

**commit;**

**Insertar Datos en Tablas**

Para agregar datos a una tabla usamos:  
 INSERT INTO nombre\_tabla VALUES (valor1, valor2, ..., valorN);  
  
Ejemplo:  
 INSERT INTO persona VALUES (123456, 'Pedro', 'Pérez', '042496738', 19);  
  
Si no se insertan todos los campos, se debe especificar el nombre de las columnas:  
 INSERT INTO persona (cedula, nombre, apellido) VALUES (123457, 'Ana', 'Gómez');

**8. COMANDOS VARIOS**

* - mysql -u usuario -p - Conectar al servidor MySQL.
* - source ruta\archivo.sql - Ejecutar script SQL.
* - SHOW DATABASES - Ver todas las bases de datos.

**9. VISTAS**

* CREATE VIEW (nombre) AS (y a continuacion la consulta) me crea una tabla con los datos de mi consulta
* CREATE OR REPLACE VIEW (nombre) AS (modifica las vistas si necesito añadir algo)