Bases de datos – MySQL

## Básico:

* \sql
* \connect root@localhost (*te pedirá contraseña*)
* show databases;
* create database ‘nombre’
* drop database ‘nombre’ (elimina)
* use ‘nombre’
* show tables;

## Create table:

* créate table ‘nombre’
* créate table if not exists países (…);
* Create table ejemplo(
  + nif varchar(9) primary key,
  + codigo int AUTO\_ INCREMENT primary key,
  + nombre varchar(20) notnull;
  + int edad
  + edad int check (edad > 18) default 20);
  + fecha\_nacimiento date);

* desc clientes; (ver datos de la tabla)
* drop table `nombre` (eliminar la tabla)

## Select:

* select \* from ‘nombre’;
* select distinct(‘Lo que quieres mirar’) from ‘tabla’
  + // te muestra solo los que tienen valores distintos en esa columna.
* Select \* from televisores where (‘columna’ = ‘valor’)
  + Te busca en toda la tabla y te muestra los registros donde el valor es el que pides
* Select count(\*) from ‘tabla’;
* Select count (‘columna’) from ‘tabla’:
* Select count(distinct(‘columna’)) from tabla;
* Select sum(‘columna’) from ‘tabla’ where (col = ‘valor’)
* Select max/min…
* select concat (marca,'-',modelo) from televisores;
* select concat (marca,'-',modelo) as ‘Nombre que quieras ver’ from televisores;
* select marca, modelo, UPPER(complementos) as complementos from televisores;
  + Creo que hay que poner el upper antes del elemento a cambiar
* select marca, modelo, SUBSTRING(upper(complementos), 2,9) as complementos from televisores;
* Select \* from provincias where cod\_comunidad not in (select cod\_comunidad from comunidades);
  + Imprime todos los registros en los que el cod comunidad no esté en el cod comunidad de comunidades (En este caso es imposible porque es clave foránea).
* Select \* from televisores ORDER BY columna asc/desc;
* Select \* from televisores ORDER BY marca desc, modelo asc, precio\_unidad asc;
  + // Lo ordena por importancia según lo hayas metido
* select \* from televisores where marca LIKE ('%am%')
  + Donde está am en marca (Samsung)
* select \* from televisores where marca like ('\_g');
  + Donde está la g en la segunda posición (\_ es una letra en blanco)
* select marca, count(\*) from televisores GROUP BY marca;
  + Me saca cada marca con la cantidad de registros que hay de cada una
* select marca, sum(precio\_unidad\*numero\_unidades) from televisores group by marca;
  + Lo divide por marcas, de cada una coge cada número de unidades y lo multiplica por su precio, sacando así el precio que costaría comprar todos.
* select cod\_comunidad, count(cod\_provincia) from provincias group by cod\_comunidad HAVING cod\_comunidad='01';
  + El having es como el where pero se puede aplicar a los elementos agrupados.
* select \* from televisores LIMIT 5;
  + Te muestra solo 5 registros;
* select \* from televisores limit 3,5;
  + Te muestra desde el registro 4(3+1), 5 registros
* Select AVG(precio\_unidad) as precio\_medio\_unidad from televisores where resolucion = '4K';

Insert:

* insert into paises values ('104', 'BULGARIA', '01', 'EUROPA');
* control c ->
* \*Insertar varios registros a la vez\*
  + Insert into continentes values (‘01’,’Europa’), (‘02’,’América’)
  + Insert into continentes (cod\_continente) values (‘01’)
* Insert into ‘tabla’ values ('999','Narnia','01', NULL, LOAD\_FILE('C:\Users\Usuario DAM1\Documents\DAM\3. Base de datos\bandera.jpg'));

Alter table:

* Alter table ‘nombre tabla’ change ‘nombre nuevo’;
* Alter table países add column ‘nombre’ longblob;
* Alter table países add column ‘nombreColumna’ varchar(20) after/first ‘otraColumna’  
  //antes o detrás
* Alter table ‘tabla’drop column ‘nombre’);
* Créate temporary table ‘nombre’ (‘campos con tipo’) \*tabla temporal\*

Update:

* update televisores set complementos = 'wifi gratis' where (precio\_unidad = 1160);

Commits:

* Commits:  
  set autocommit =0/1;
* commit
* rollback

Index:

Con un index puedes guardar una cierta ordenación de datos para ser consultados más tarde

* Create index índice on tabla1(importe desc)

Join:

* SELECT \* FROM coches.coches INNER JOIN coches.propietarios t2 on (t1.nif\_propietario = t2.nif);
  + Inner join: Junta coches y propietarios donde el nif del comprador del coche y el comprador en sí sea igual
* SELECT \* FROM coches.coches LEFT JOIN coches.propietarios t2 on (t1.nif\_propietario = t2.nif);
  + Left join: Pone todos los registros de la tabla de la izquierda y si puede juntar el “nif” lo junta con el propietario y si no pues lo pone en null.
* SELECT t1.matricula, t1.marca, t1.modelo, t1.nif,t2.nif, t2.nombre, t2.apellidos FROM coches.coches t1 NATURAL JOIN coches.propietarios t2;
  + Natural join: hace un join solo si el campo de las dos tablas elegidas tiene exactamente el mismo nombre

## View:

* Create view ventas\_todo as ….
  + Te permite crear una vista o una tabla a la que se va a accede mucho, por lo que mejora el rendimiento
* Alter view
  + Tienes que poner de nuevo el prompt entero, no se borra porque queremos mantener los permisos asignados
* SELECT \* FROM baseprovincias.ventas\_todo where `nombre comunidad` = comunidad de madrid';

Grant y revoke:

* Grant Select on baseprovincias ventas to usuario1@localhost;
  + Le das el permiso a hacer select en la databse baseprovincias solo para la table ventas a usuario1
* Grant ALL PRIVILEGES on \*.\* user@localhost;

Roles:

* Create role nombre@localhost;
* Grant ‘Select on’ base01.vendedores to nombre@localhost;
* Grant role1@localhost to usuario@localhost;

Trigger:

* create trigger trigger1 before update on ventas for each row;

delimiter $$

create trigger trigger1 before update on ventas for each row

* + begin
  + if new.preciounitario**<>** old.preciounitario() or new.unidades\_vendidas<> old.unidades\_vendidas
  + then set new.importe\_total = new.unidades\_vendidas \* new.preciounitario;
  + end if;
  + end $$

delimiter ;

(Está hacienda un trigger que se ejecuta antes de que se actualice la table ventas, en el cual si el nuevo precio\_unitario o ud\_ven es mayor o menor que el antiguo precio\_unitario o ud\_ven, entonces ejecuta el comando y actualiza new importe\_total)

Procedure y loop:

\* IMPORTANTE \*

\*DDL es para la estructura(tabla) y el DML es para los datos(dentro tabla)\*

\*DCL (controlar permisos) – Grant y Revoke\*

Bases de datos – MySQL

Create databaes

* use base01 (luego simplemente inserta un archivo para crearla)

Insert

* db.empleados.insertOne({hola: "hola"})
* db.empleados.insertOne([{hola: "hola"}, {adios: "adios"}])
* db.empleados.insertMany([{hola: "hola"}, {adios: "adios"}])

Create index

* db.personas.createIndex({nif:1}, {name: "indice1"}) (El 1 ordena ascendentemente)

Create index (clave primaria)

* db.personas.createIndex({nif:1}, {name: "indice2", unique: true})
* db.personas.createIndex({nif:1}, {name: "indice2", unique: true, **\*sparse: true\*}**)
  + El sparse permite que haya documentos que no tengan nif