1)¿QUÉ ES SQL? Y MYSQL?

SQL, Lenguaje de Consulta Estructurado (Structured Query Language) es un lenguaje declarativo de acceso a bases de datos relacionales que permite especificar diversos tipos de operaciones sobre las mismas.
El Lenguaje Estructurado de Consulta (SQL) es un lenguaje de base de datos normalizado, utilizado por los diferentes motores de bases de datos para realizar determinadas operaciones sobre los datos, o sobre la estructura de los mismos.

2)TRANSACT-SQL

SQL Server tiene incorporado un lenguaje denominado Transact-SQL que contiene los comandos y sentencias necesarios para administrar una instancia de SQL Server, crear y administrar sus objetos, y consultar, insertar, modificar, y eliminar datos de las tablas.

3)Tipos de Sentencias

Las distintas sentencias que se pueden ejecutar son:

- (**DDL** Data Definition Language) **De Definición de Datos**: se utilizan para crear, modificar y eliminar objetos de una base de datos.

Algunas son: CREATE
TABLE, ALTER TABLE, DROP TABLE, CREATE INDEX etc.

- (**DML** Data Manipulation Language) **De Manipulación de Datos**:

se utilizan para recuperar, agregar, modificar o eliminar datos de una base de datos. Por ejemplo: SELECT, INSERT, UPDATE, DELETE.

- (**DCL** Data Control Language) **De Control de Acceso**:

conceden o suprimen privilegios a usuarios. Por ejemplo: GRANT y REVOKE.

- (**TCL** Transaction Contro Languaje) **De Control de Transacciones**:

finalizan o abortan la transacción actual: COMMIT, ROLLBACK.

- **De Programación**: DECLARE, OPEN, EXECUTE, DESCRIBE, etc.

LO PRIMERO: CREAR LA BASE DE DATOS:

CREATE DATABASE……nombre de la base de datos:
Ejemplo: **CREATE DATABASE** LIBRERÍA

creación de tabla:

!

CREATE TABLE nombre_tabla
(nom_col1 tipo_dato **[PRIMARY KEY] [AUTOINCREMENT][NULL|NOT NULL]**,
nom_col2 tipo_dato **[NULL|NOT NULL]**,
nom_col_n tipo_dato **[NULL|NOT NULL]**
)

QUÉ ES AUTOINCREMENT: es un incremento automático
permite generar un número único cuando insertamos un nuevo registro en la tabla. Se utiliza para tener una clave primaria de una tabla mediante la generación automática de un número secuencial único en la tabla.

OJO CON DNI!!!

NULL / NOT NULL

Lo que está entre **[ ]** es opcional incorporarlo o no en la sentencia.
Lo que se separa por la barra vertical "**|**" es que debe elegirse uno o el otro.
Las palabras en mayúsculas son palabras reservadas del lenguaje (EL XAMPP no es "case sensitivity" por lo tanto LAS TOMA EN MINÚSCULAS)
Los tipos de datos más comunes que se pueden utilizar en MySQL:

- **INTEGER/INT** almacena números enteros como -2567, 0 , 1 y 67.
- **VARCHAR** almacena datos de texto. Se puede especificar el número máximo de caracteres así *VARCHAR(n)*. Por ejemplo, varchar(4) puede almacenar los textos "Lisa", "Os" o "Gary".
- **CHAR** es similar, pero en lugar de establecer la longitud máxima, se establece la longitud absoluta. Por ejemplo, *CHAR(4)* almacenará 'Os' como 'Os '. Tenga en cuenta los dos espacios adicionales, que hacen que la longitud sea de 4.
- **FLOAT** almacena números de punto flotante, como -23,789, 23,5 y 78,0.
- **DECIMAL** es similar a float en el sentido de que almacena números de punto decimal. Sin embargo, puede indicar el número total máximo de dígitos*(*p) y el número de dígitos después del decimal*(s*) de la siguiente manera *DECIMAL(p,s)*. En otras palabras, decimal(4,2) puede almacenar 12.56, 1.56, 70. A diferencia de FLOAT, DECIMAL ofrece cálculos precisos.

Utilízalo para almacenar valores monetarios.

- **TIME** almacena datos de tiempo, por ejemplo, '12:34:50'.
- **DATE** almacena datos de fecha, por ejemplo '2020-09-09'.
- **DATETIME** almacena datos de fecha y hora en un solo campo, por ejemplo "2020-03-30 12:34:50".

modificar la estructura o definición de un objeto, agregar o eliminar componentes del mismo

```
ALTER TABLE nombre_tabla
ADD COLUMN nombre_columna tipo_dato
```

Ejemplo: se quiere agregar la contextura física en tabla Personas

```
ALTER TABLE Personas
ADD contextura_fisica varchar(20)
```

Para eliminar una columna:

```
ALTER TABLE nombre_tabla
DROP COLUMN nombre_columna
```

Ejemplo: se quiere eliminar la columna o el campo contextura_fisica de la tabla Personas

```
ALTER TABLE Personas
```

DROP COLUMN contextura_fisica

Para modificar el tipo de dato:

ALTER TABLE nombre_tabla
ALTER COLUMN nombre_columna tipo_dato
Ejemplo: se quiere modificar el tamaño de la columna o el campo nro_doc de la tabla Personas
ALTER TABLE Personas
ALTER COLUMN nro_doc varchar(10)

Para agregar una clave primaria:

ALTER TABLE nombre_tabla
ADD CONSTRAINT nombre_clave **PRIMARY KEY (**columna_PK**)**
Ejemplo: se quiere agregar la clave primaria a la tabla Títulos que no se determinó cuando se
creó la tabla
Alter table Titulos
add constraint pk_titulos primary key (id_titulo)

Para eliminar una tabla y todo su contenido(ojo con las relaciones con otras tablas):

DROP TABLE nombre_table
Ejemplo: se quiere eliminar la tabla Personas
drop table Personas
Con las Sentencias de Manipulación de Datos se puede recuperar datos de las tablas de
una base de datos, insertar nuevos registros, modificar o eliminar **registros** existentes.
Las sentencias son:
SELECT : Para recuperar los datos de una o más tablas.
INSERT : Para insertar o cargar nuevos datos
UPDATE : Para modificar datos existentes.
DELETE : Para eliminar datos existentes.

Sentencia INSERT

Se utiliza la sentencia INSERT para agregar nuevos registros de datos a una tabla de una base
de datos.
INSERT INTO Nombre_Tabla **(**lista_columnas**)**
VALUES (lista_valores**)**
Sentencia abreviada:
INSERT INTO Nombre_Tabla
VALUES (lista_valores**)**
Los valores deben coincidir con el tipo de dato de la columna correspondiente, y deben existir tantos valores como columnas se especifiquen en la lista. Si una columna no tiene valor se debe introducir un valor NULL. Los datos string y date van con comillas y entre los valores van comas.

Ejemplo:
INSERT INTO Personas (legajo,apellido,peso ,fecha_nac)
VALUES (2154,'López',89.5,1,'10/05/1997')
OJO: SI PUSIMOS AUTOINCREMENT A LEGAJO VA NULLSENTENCIA ABREVIADA:
INSERT INTO Personas
VALUES (2154,'López',89.5,1,'10/05/1997')

Sentencia UPDATE

Esta sentencia nos permite modificar los datos de las columnas de las tablas.
UPDATE Nombre_Tabla
SET columna1 = Nuevo_Valor1
[WHERE condición**]**
Ejemplo: actualizar el peso de la persona ingresada en el ejemplo anterior con cuyo legajo es 2154
UPDATE Personas
SET peso = 85.5
WHERE legajo = 2154

Si no le agregamos una condicion con el where le cambiará a todas las personas el peso.

Sentencia DELETE

La instrucción DELETE elimina una o más filas de una tabla.
DELETE Nombre_tabla
[WHERE Condición**]**
Ojo: Tener en cuenta que, si no se especifica la cláusula WHERE, se borran todas las filas de una tabla.
Por ejemplo:
eliminar el registro de la tabla personas cuyo legajo es 3210
DELETE personas
WHERE legajo = 3210

Sentencia SELECT

Es para recuperar datos, son las llamadas **"Consultas".** Una Consulta es un conjunto de instrucciones, que permiten ver o interactuar con los datos contenidos en las tablas de una base de datos. Se recurre a ellas cuando, por ejemplo, se quiere obtener salidas de información de tablas de una base de datos obteniendo un subconjunto de registros que pertenecen a una o varias tablas.
No se trata únicamente de obtener los datos deseados, sino de hacerlo además en un tiempo razonable.
La sentencia SELECT recupera datos de una base de datos y los devuelve en forma de tablas que **no** quedan guardadas en la base de datos
La sintaxis es la siguiente
SELECT lista_columnas

FROM lista_tablas

Teniendo las tablas del Personas se quiere listar legajo y apellido y nombre y fecha de nacimiento desde la tabla personas

SELECT legajo, apellido y nombre y fech_nacim
FROM Personas
select *
from **Personas**

Concatenar caracteres utilizando el signo más: **'+'**

SELECT legajo, **CONCAT**(apellido **+ ' , ' +** nombre), fecha_nac
FROM Personas

Alias: AS
Se puede agregar un alias a los campos que aparecerán como encabezados de columnas: los alias van entre comillas si son dos o más palabras…si es una no hacen falta las comillas

SELECT legajo,
CONCAT(apellido **' , '** nombre) **AS** 'Nombre completo', fecha_nac
FROM Personas

Ordenar la lista de los clientes por el apellido
SELECT legajo, apellido y nombre y fech_nacim
FROM Personas
ORDER BY **apellido (ASC O DESC)**
OJO: la condición para ordenar SIEMPRE debe estar en el select Se puede enumerar: ejemplo: ordenar de manera descendente por el apellido: ORDER BY 2 (asc o desc)

Supongamos que necesito los dos libros más baratos:
SELECT nombre_libro,precio
FROM libros
ORDER BY **precio** **ASC**
LIMIT 2
