Guia base de datos

Wuke Zhang

https://

www.w3schools.c

om/sql/default.asp

SHOW DATABASES;  DESCRIBE TABLE ….. ;

```sql
CREATE DATABASE ICSEHotelDB;

 USE ICSEHotelDB;

CREATE TABLE customer(

ID INT(15) PRIMARY KEY AUTO_INCREMENT,
NAME VARCHAR(15) NOT NULL,
SURNAME VARCHAR(15) NOT NULL,
BIRTHDATE DATE NOT NULL,
GENDER VARCHAR(15) NOT NULL,
PHONE_NUMBER VARCHAR(15) NOT NULL,
DNI VARCHAR(15) NOT NULL,
ADDRESS VARCHAR(30) NOT NULL,
POSTAL_CODE VARCHAR(15) NOT NULL,
PARKING_REQUIRED ENUM("YES","NO") NOT NULL,
SUGGESTIONS VARCHAR(30),
ROOM_TYPE ENUM("individual","double","family") NOT NULL,
ROOM_NUMBER ENUM("1.1","1.2","1.3","2.1","2.2","2.3") NOT NULL,
PAYMENT_METHOD ENUM("cash","card") NOT NULL,
CHECK_IN_DATE DATE NOT NULL,
CHECK_OUT_DATE DATE NOT NULL,
RESERVATION_NUMBER INT(15) NOT NULL UNIQUE

);

INSERT INTO customer
(NAME,SURNAME,BIRTHDATE,GENDER,PHONE_NUMBER,DNI,ADDRESS,POSTAL_CODE,PARKING_REQUIRED,SUGGESTIONS,ROOM_TYPE,ROOM_NUMBER,PAYMENT_METHOD,CHECK_IN_DATE,CHECK_OUT_DATE,RESERVATION_NUMBER)
VALUES
("JHON","DILO","1994-11-14","MALE",994555241,"V42124578I","Bravo Murillo",35100,"YES","NOTHING SPECIAL","individual","1.1","cash","2000-10-6","2006-7-11",62),
("JHON","DILO","1994-11-14","MALE",994555241,"V42124578I","Bravo Murillo",35100,"YES","NOTHING SPECIAL","individual","1.2","cash","2000-10-6","2006-7-11",63);
```

http

sakjdakjldjajkajdkl

://adsa dadas localhost dasd / ddad phpmyadmin /

https://desarrolloweb.com/manuales/tutorial-sql.html

Relación 1 a 1 (uno a uno)

Ejemplo: Cada persona tiene un único pasaporte y cada pasaporte pertenece a una única persona. Código de Ejemplo:

```
CREATE TABLE Persona (
    id_persona INT PRIMARY KEY,
    nombre VARCHAR(100)
);

CREATE TABLE Pasaporte (
    id_pasaporte INT PRIMARY KEY,
    numero_pasaporte VARCHAR(50),
    id_persona INT UNIQUE,
    FOREIGN KEY (id_persona) REFERENCES Persona(id_persona)
);
```

Inserción de valores en la relación 1 a 1

```
INSERT INTO Persona (id_persona, nombre) VALUES
(1, 'Alejandro García'),
(2, 'Carmen López'),
(3, 'Roberto Fernández'),
(4, 'Patricia Martínez'),
(5, 'Luis González'),
(6, 'Sandra Pérez'),
(7, 'Manuel Gutiérrez'),
(8, 'Lucía Rodríguez'),
(9, 'Francisco Sánchez'),
(10, 'Ana Torres');

INSERT INTO Pasaporte (id_pasaporte, numero_pasaporte, id_persona) VALUES
(1, 'PAS123456', 1),
(2, 'PAS234567', 2),
(3, 'PAS345678', 3),
(4, 'PAS456789', 4),
(5, 'PAS567890', 5),
(6, 'PAS678901', 6),
(7, 'PAS789012', 7),
(8, 'PAS890123', 8),
(9, 'PAS901234', 9),
(10, 'PAS012345', 10);
```

## Consulta JOIN a una relación 1a1 de las tablas creadas en la clase anterior

```
SELECT *
FROM Persona p
JOIN Pasaporte pa ON p.id_persona = pa.id_persona;
```

## Explicación del Uso de JOIN en SQL

### Uso del JOIN en la Consulta de Persona y Pasaporte

En las tablas Persona y Pasaporte, que tienen una relación 1-1, utilizamos JOIN (por defecto, un INNER JOIN) para combinar filas de estas dos tablas basándonos en su columna de relación, id_persona. Este JOIN opera de la siguiente manera:

- Examina cada fila de la tabla Persona.

- Busca en Pasaporte una fila donde id_persona coincida.
- Combina la información de las filas coincidentes en una única fila.
- El resultado incluye columnas de ambas tablas.

### Concepto General de JOIN

El JOIN en SQL es una operación para combinar filas de dos o más tablas, basándose en una columna común (clave de unión). Los tipos más comunes de JOIN son:

- **INNER JOIN:** Devuelve filas cuando hay una coincidencia en ambas tablas.
- **LEFT JOIN (o LEFT OUTER JOIN):** Devuelve todas las filas de la tabla izquierda y las filas coincidentes de la tabla derecha. Si no hay coincidencia, el resultado es NULL en el lado derecho.
- **RIGHT JOIN (o RIGHT OUTER JOIN):** Es lo opuesto al LEFT JOIN. Devuelve todas las filas de la tabla derecha y las coincidentes de la izquierda. Si no hay coincidencia, el resultado es NULL en el lado izquierdo.
- **FULL JOIN (o FULL OUTER JOIN):** Combina los resultados de LEFT JOIN y RIGHT JOIN. Devuelve filas cuando hay una coincidencia en una de las tablas.

En el contexto de clases sobre relaciones 1-1 en bases de datos y el ejemplo específico de Persona y Pasaporte, el enfoque se centra en el uso del INNER JOIN. Los otros tipos de JOIN (como LEFT JOIN y RIGHT JOIN) también son importantes en SQL, pero pueden ser explorados más adelante en contextos donde las relaciones entre las tablas son diferentes o se requieren consultas más complejas.

- Creación de una Relación 1 a N entre Libro y Autor

### Creación de Tablas

Primero, definimos las tablas Autor y Libro. Cada autor puede escribir varios libros, pero cada libro tiene un único autor. Esto establece una relación de 1 a N (uno a muchos) entre Autor y Libro.

```
CREATE TABLE Autor (
    id_autor INT PRIMARY KEY,
    nombre VARCHAR(100)
);

CREATE TABLE Libro (
    id_libro INT PRIMARY KEY,
    titulo VARCHAR(100),
    id_autor INT,
    FOREIGN KEY (id_autor) REFERENCES Autor(id_autor)
);
```

## Inserción de Datos en las Tablas Autor y Libro

### Inserción de Autores

Insertamos 20 autores en la tabla Autor.

```
INSERT INTO Autor (id_autor, nombre) VALUES
(1, 'Gabriel García Márquez'),
(2, 'J.K. Rowling'),
(3, 'George Orwell'),
(4, 'Isabel Allende'),
(5, 'Stephen King'),
(6, 'Haruki Murakami'),
(7, 'Jane Austen'),
(8, 'Charles Dickens'),
(9, 'Leo Tolstoy'),
(10, 'Mark Twain'),
(11, 'Virginia Woolf'),
(12, 'Agatha Christie'),
(13, 'F. Scott Fitzgerald'),
(14, 'Ernest Hemingway'),
(15, 'William Shakespeare'),
(16, 'Miguel de Cervantes'),
(17, 'Herman Melville'),
(18, 'Franz Kafka'),
(19, 'J.R.R. Tolkien'),
(20, 'H.P. Lovecraft')
```

```
INSERT INTO Libro (id_libro, titulo, id_autor) VALUES
(1, 'Cien años de soledad', 1),
(2, 'El amor en los tiempos del cólera', 1),
(3, 'Harry Potter y la piedra filosofal', 2),
(4, 'Harry Potter y la cámara secreta', 2),
(5, '1984', 3),
(6, 'Rebelión en la granja', 3),
(7, 'La casa de los espíritus', 4),
(8, 'Eva Luna', 4),
(9, 'It', 5),
(10, 'El resplandor', 5),
(11, 'Kafka en la orilla', 6),
(12, 'Norwegian Wood', 6),
(13, 'Orgullo y prejuicio', 7),
(14, 'Sentido y sensibilidad', 7),
(15, 'Oliver Twist', 8),
(16, 'Historia de dos ciudades', 8),
(17, 'Guerra y Paz', 9),
(18, 'Anna Karenina', 9),
(19, 'Las aventuras de Tom Sawyer', 10),
(20, 'Las aventuras de Huckleberry Finn', 10);
```

Sure, I can help you with those queries in SQL. Here are the queries for each of your requirements:

1. Consultar Todos los Autores (Retrieve All Authors):

```sql
SELECT * FROM Autor;
```

2. Consultar Todos los Libros de un Autor Específico (Retrieve All Books by a Specific Author):

```sql
-- Replace 'AuthorName' with the actual name of the author
SELECT Libro.* FROM Libro
JOIN Autor ON Libro.id_autor = Autor.id_autor
WHERE Autor.nombre = 'AuthorName';
```

3. Contar el Número Total de Libros (Count the Total Number of Books):

```sql
SELECT COUNT(*) AS total_libros FROM Libro;
```

4. Encontrar Autores que no Han Escrito Libros (Find Authors Who Have Not Written Books):

```sql
SELECT Autor.* FROM Autor
LEFT JOIN Libro ON Autor.id_autor = Libro.id_autor
WHERE Libro.id_libro IS NULL;
```

5. Consultar Libros y sus Autores (Retrieve Books and their Authors):

```sql
SELECT Libro.*, Autor.nombre AS nombre_autor FROM Libro
JOIN Autor ON Libro.id_autor = Autor.id_autor;
```

6. Encontrar el Autor con Más Libros Publicados (Find the Author with the Most Published Books):

```sql
SELECT Autor.id_autor, Autor.nombre, COUNT(Libro.id_libro) AS total_libros
FROM Autor
LEFT JOIN Libro ON Autor.id_autor = Libro.id_autor
GROUP BY Autor.id_autor, Autor.nombre
ORDER BY total_libros DESC
LIMIT 1;
```

7. Actualizar el Nombre de un Autor (Update the Name of an Author):

```sql
-- Replace 'AuthorID' with the actual ID of the author and 'NewName' with the new name
UPDATE Autor SET nombre = 'NewName' WHERE id_autor = AuthorID;
```

8. Eliminar un Libro (Delete a Book):

```sql
-- Replace 'BookID' with the actual ID of the book
DELETE FROM Libro WHERE id_libro = BookID;
```

Make sure to replace the placeholders (e.g., 'AuthorName', 'AuthorID', 'NewName', 'BookID') with actual values when executing these queries.

Esquema de Base de Datos

1. **Biblioteca**: `id_biblioteca`, `nombre`, `ubicacion`
2. **Libro**: `id_libro`, `titulo`, `autor`, `anio_publicacion`, `id_biblioteca`
3. **Miembro**: `id_miembro`, `nombre`, `email`
4. **Prestamo**: `id_prestamo`, `id_libro`, `id_miembro`, `fecha_prestamo`, `fecha_devolucion`

## Tareas

1. **Creación de Tablas**: Crea las tablas según el esquema proporcionado.
2. **Inserción de Datos**: Inserta al menos 5 bibliotecas, 15 libros (distribuidos entre las bibliotecas), y 10 miembros.
3. **Consulta de Libros**: Haz una consulta para mostrar todos los libros de una biblioteca específica.
4. **Prestar un Libro**: Inserta un registro en `Prestamo` para simular que un miembro toma prestado un libro.
5. **Libros Prestados**: Haz una consulta para mostrar todos los libros que están actualmente prestados.
6. **Actualizar Datos**: Cambia la ubicación de una biblioteca.
7. **Eliminar Datos**: Elimina un libro que ya no está disponible.
8. **Consulta Avanzada**: Lista todos los miembros que han tomado prestado un libro publicado después del año 2000.

```sql
CREATE TABLE Library(
    id_library INT PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(15) NOT NULL,
    ubication VARCHAR(15) NOT NULL
);

CREATE TABLE Books(
    id_book INT PRIMARY KEY AUTO_INCREMENT,
    title VARCHAR(15) NOT NULL,
    author VARCHAR(15) NOT NULL,
    release_year DATE NOT NULL,
    Library_id INT NOT NULL,
    CONSTRAINT FK_Books_Library FOREIGN KEY (Library_id)
    REFERENCES Library(id_library)
```

```
);

CREATE TABLE Member(
   id_member INT PRIMARY KEY AUTO_INCREMENT,
   name VARCHAR(15) NOT NULL,
   email VARCHAR(30) NOT NULL
);

CREATE TABLE Loan(
   id_loan INT PRIMARY KEY AUTO_INCREMENT,
   Books_id INT,
   Member_id INT,
   CONSTRAINT FK_Loan_Books FOREIGN KEY (Books_id)
   REFERENCES Books(id_book),
   CONSTRAINT FK_Loan_Member FOREIGN KEY (Member_id)
   REFERENCES Member(id_member),
   loan_date DATE NOT NULL,
   refund_date DATE
);

INSERT INTO Library(name,ubication)
VALUES("santz","barcelona"),("sol","madrid"),("santa","canarias"),("cohone","venezuela"),("el buru","canarias");

INSERT INTO Books(title,author,release_year,Library_id)
VALUES('El Cazador de la Bruja','J.K. Rowling','2023-12-04',3),('Harry Potter y el prisionero de Azkaban','J.K.
Rowling','2023-12-03',4),
('The Lord of the Rings','J.R.R Tolkien','2004-09-05',1),("Uzumaki","Junji Ito","1998-1-19",5),("Las aventuras de
Juancholooooooooo","Me","2023-12-24",3),
("Sayonara Eri","Tatsuki Fujimoto","2022-04-11",1),("Juan Piz Red edition","Goda Sensei","2011-07-01",2),('El Cazador de
la Bruja 2','J.K. Rowling','2023-12-04',3),
('El Cazador de la Bruja 3','J.K. Rowling','2023-12-04',3),('El Cazador de la Bruja 4','J.K. Rowling','2023-12-04',3),
("Juan Piz Blue edition","Goda Sensei","2011-07-01",2),
('The Hobbit','J.R.R Tolkien',"2006-02-03",1),("Bernarda Alba","pepe","1990-03-06",2),("Las aventuras de
Juancholooo","Me","2023-12-31",3),("The last punch","Hadriel","2023-11-24",3);

INSERT INTO Member(name,email)
VALUES("Me","sydfs@jsfdjf.com"),("You","sydf@jsfdjf.com"),("pepa","syds@jsfdjf.com"),("pepi","syfs@jsfdjf.com"),
("pepe","sdfs@jsfdjf.com"),
("pimpe","ydfs@jsfdjf.com"),("kimerico","sydfs@jsfdj.com"),("Taniansu","sydfs@jsfdf.com"),("chonay","sydfs@jsfjf.com"),
("Kento Yoshioka","japanese@super_rat.com");

INSERT INTO Loan(Books_id,Member_id,loan_date,refund_date)
VALUES(3,4,"2010-01-10","2011-10-01"),(1,1,"2010-01-10",NULL),(4,2,"2010-08-15",NULL);

SELECT * FROM Books WHERE Library_id = 1;

SELECT * FROM Books
JOIN Loan ON Books.id_book=Loan.Books_id
WHERE Loan.refund_date IS NULL;

UPDATE Library
SET ubication="las palmas"
WHERE id_library=5;

DELETE FROM Books WHERE title="Bernarda Alba";

SELECT * FROM Member
JOIN Loan ON Member.id_member = Loan.Member_id
WHERE (Loan.loan_date) > 2000;
```

```
DROP DATABASE databasename;
```

**Note:** Be careful before dropping a database. Deleting a database will result in loss of complete information stored in the database!

## DROP DATABASE Example

The following SQL statement drops the existing database "testDB":

### Example

Get your own SQL Server

```
DROP DATABASE testDB;
```

Activar Wi

### Example

Get your own SQL Server

Return data from the Customers table:

```
SELECT CustomerName, City FROM Customers;
```

Try it Yourself »

## Syntax

```
SELECT column1, column2, ...
FROM table_name;
```

### Example

Return all the columns from the Customers table:

```
SELECT * FROM Customers;
```

Try it Yourself »

## Example

Get your own SQL Server

Select all customers from Mexico:

```
SELECT * FROM Customers
WHERE Country='Mexico';
```

Try it Yourself »

## Syntax

```
SELECT column1, column2, ...
FROM table_name
WHERE condition;
```

### Example

```
SELECT * FROM Customers
WHERE CustomerID=1;
```

Try it Yourself »

## Operators in The WHERE Clause

You can use other operators than the `=` operator to filter the search.

### Example

Select all customers with a CustomerID greater than 80:

```
SELECT * FROM Customers
WHERE CustomerID > 80;
```

Try it Yourself »

### Example

Remove the Customers table:

```
DROP TABLE Customers;
```

It is possible to delete all rows in a table without deleting the table. This means that the table structure, attributes, and indexes will be intact:

```
DELETE FROM table_name;
```

The following SQL statement deletes all rows in the "Customers" table, without deleting the table:

### Example

```
DELETE FROM Customers;
```

### Example

```
DELETE FROM Customers WHERE CustomerName='Alfreds Futterkiste';
```

# SQL FOREIGN KEY on ALTER TABLE

To create a `FOREIGN KEY` constraint on the "PersonID" column when the "Orders" table is already created, use the following SQL:

**MySQL / SQL Server / Oracle / MS Access:**

```
ALTER TABLE Orders
ADD FOREIGN KEY (PersonID) REFERENCES Persons(PersonID);
```

To allow naming of a `FOREIGN KEY` constraint, and for defining a `FOREIGN KEY` constraint on multiple columns, use the following SQL syntax:

**MySQL / SQL Server / Oracle / MS Access:**

```
ALTER TABLE Orders
ADD CONSTRAINT FK_PersonOrder
FOREIGN KEY (PersonID) REFERENCES Persons(PersonID);
```

# DROP a FOREIGN KEY Constraint

To drop a `FOREIGN KEY` constraint, use the following SQL:

**MySQL:**

```
ALTER TABLE Orders
DROP FOREIGN KEY FK_PersonOrder;
```

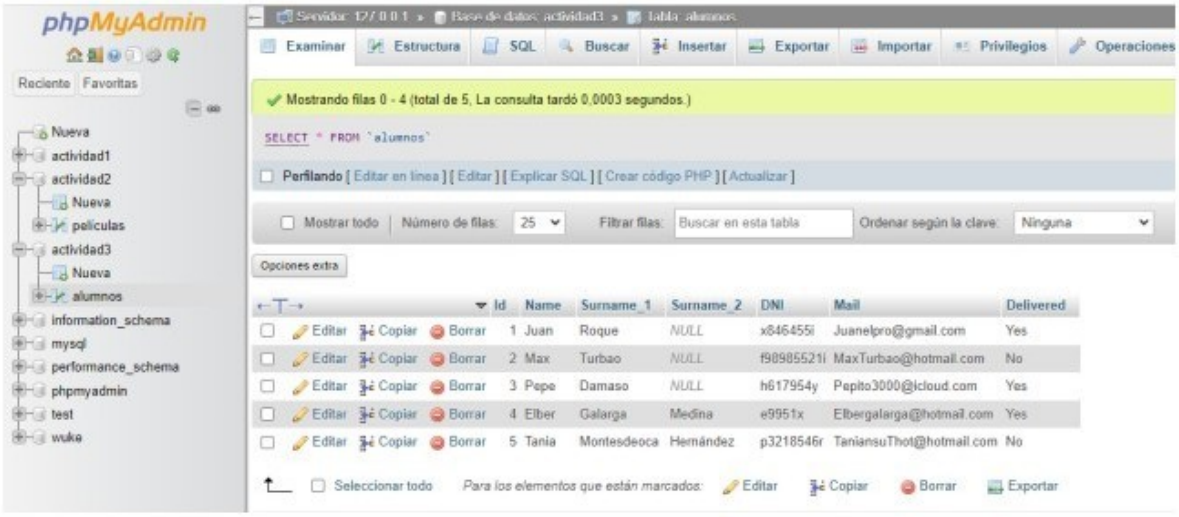
**SQL Server / Oracle / MS Access:**

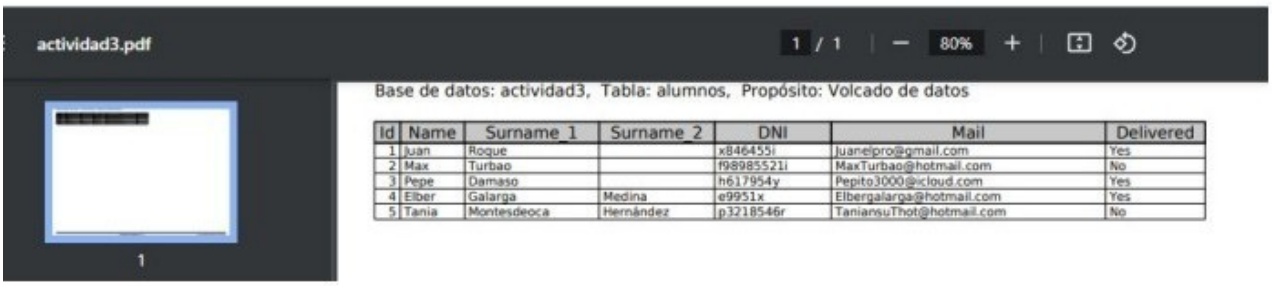
```
ALTER TABLE Orders
DROP CONSTRAINT FK_PersonOrder;
```

ON DELETE CASCADE
ON UPDATE
CASCADE;

ON DELETE RESTRICT

```sql
CREATE DATABASE ACTIVIDAD5;

USE ACTIVIDAD5

CREATE TABLE COUNTRY(
ID INT(15) PRIMARY KEY AUTO_INCREMENT,
NAME VARCHAR(15) NOT NULL UNIQUE,
CONTINENT_ID INT(15) NOT NULL,
CODE INT(15) NOT NULL,
CONSTRAINT FK_COUNTRY_CONTINENT FOREIGN KEY (CONTINENT_ID)
REFERENCES CONTINENT(ID)
);

CREATE TABLE CONTINENT(
ID INT(15) PRIMARY KEY AUTO_INCREMENT,
NAME VARCHAR(15) NOT NULL UNIQUE
);

INSERT INTO CONTINENT (NAME)
VALUES ('ASIA'),
       ('EUROPE'),
       ('AFRICA'),
       ('NORTH_AMERICA'),
       ('SOUTH_AMERICA');


INSERT INTO COUNTRY (CODE, NAME, CONTINENT_ID)
VALUES (108, 'SPAIN', 2),
       (115, 'ITALY', 2),
       (202, 'ANGOLA', 3),
       (227, 'MALI', 3),
       (407, 'CHINA', 1),
       (415, 'JAPAN', 1),
       (302, 'USA', 4),
       (301, 'CANADA', 4),
       (340, 'ARGENTINA', 5),
       (342, 'BRAZIL', 5);

      DELETE FROM COUNTRY
      WHERE ID = 1;

      DELETE FROM CONTINENT
      WHERE ID = 3;
```




Paso 1: Acceder a phpmyadmin mediante xampp o el buscador.

Paso 2: Ir al apartado de SQL.

Paso 3: Crear la tabla con CREATE TABLE “nombre” (……..);

Paso 4: Insertar los valores con INSERT INTO() y abajo Values().

Activa
Ve a Co

```
CREATE TABLE actividad1(

Id INT(15) PRIMARY KEY AUTO_INCREMENT,
Title VARCHAR(50) NOT NULL,
Release_Year INT(30) NOT NULL ,
Producer VARCHAR(35) NOT NULL,
Sypnosis VARCHAR(500) NOT NULL,
Rate ENUM("1 star", "2 star", "3 star") NOT NULL,
Seen_Or_Not_Seen ENUM("Yes", "No") NOT NULL,
MONETIZATION INT(255) NOT NULL

);

INSERT INTO actividad1 (Title,Release_Year,Producer,Sypnosis,Rate,Seen_Or_Not_Seen,Monetization)
VALUES ("Gladiador",2018,"Me","Sypnosis","3 star","Yes",20000),("Blancanieves",2000,"Fran","Synopsis","2 star","Yes",1000000),("Doraemon",2006,"Nobita","Sypnosis","2 star","Yes",80000),("Miércoles",2022,"Alvaro",
```

```sql
CREATE TABLE sudaderas(

ID INT(15) PRIMARY KEY AUTO_INCREMENT,
NAME VARCHAR(15) NOT NULL,
SURNAME_1 VARCHAR(15) NOT NULL,
SURNAME_2 VARCHAR(15),
DNI VARCHAR(15) NOT NULL UNIQUE,
MAIL VARCHAR(15) NOT NULL,
SIZE ENUM("S","M","L","XL") NOT NULL DEFAULT 'L'

);

INSERT INTO sudaderas (NAME,SURNAME_1,SURNAME_2,DNI,MAIL,SIZE)
VALUES("Alvaro","Artiles","Hernández","c66587687i","alvarom@icloud.com","M"),
("Miguel","Medina","Castellano","i565654c","Mompi@gmail.com","S"),
("Iván","Hernández","Dominguero","n65492o","Ivanazo@hotmail.com","L");


INSERT INTO sudaderas (NAME,SURNAME_1,DNI,MAIL,SIZE)
VALUES("Wuke","Zhang","i987456p","ukelele@icloud.com","L");

INSERT INTO sudaderas (NAME,SURNAME_1,DNI,MAIL)
VALUES("Rubén","Jonay","o85125e","Rubiales@outlook.com");
```

# Crear una base de datos II

Wuke Zhang

1-ASIR

Primero entre en phpmyadmin mediante el siguiente link en mi buscador [redacted], otra opción es entrar desde xampp en el apartado de mysql darle a admin y ahi fui al apartado de base de datos y le di a crear, tras eso cree la tabla, primero con la estructura y luego en el apartado de insertar, inserte los valores.
Luego vuelves a la base de datos porque sino se te queda seleccionado la tabla y lo que nosotros queremos es seleccionar la base de datos y lo hacemos seleccionandolo en la parte izquierda para exportarlo con el formato que quieras.