Repaso Bases de Datos:

Vamos a diseñar un esquema clásico y funcional para un sistema de ventas. Usaremos tres tablas esenciales: Productos, Clientes y Ventas.

En este modelo, la tabla Ventas actuará como el puente, relacionando a quién se le vende y qué se está vendiendo.

1. El Diseño de la Base de Datos (Modelo Relacional)

Antes de tocar el código, visualicemos la estructura:

Productos: Almacena lo que vendes.

Clientes: Almacena quién compra.

Ventas: Registra la transacción, conectando las dos tablas anteriores mediante llaves foráneas (Foreign Keys).

2. Código SQL para XAMPP (phpMyAdmin)

Puedes copiar este código y pegarlo directamente en la pestaña SQL de tu phpMyAdmin:

1. Crear la base de datos

```
CREATE DATABASE SistemaVentas;
USE SistemaVentas;
```

2. Crear la tabla Productos

```
CREATE TABLE Productos (
    id_producto INT AUTO_INCREMENT PRIMARY KEY,
    nombre VARCHAR(100) NOT NULL,
    precio DECIMAL(10, 2) NOT NULL
);
```

```
-- 3. Crear la tabla Clientes
CREATE TABLE Clientes (
    id_cliente INT AUTO_INCREMENT PRIMARY KEY,
    nombre_cliente VARCHAR(100) NOT NULL,
    email VARCHAR(100) UNIQUE
);

-- 4. Crear la tabla Ventas (con Llaves Foráneas)
CREATE TABLE Ventas (
    id_venta INT AUTO_INCREMENT PRIMARY KEY,
    fecha DATE NOT NULL,
    id_cliente INT,
    id_producto INT,
    cantidad INT,
    -- Definición de Llaves Foráneas
    FOREIGN KEY (id_cliente) REFERENCES Clientes(id_cliente),
    FOREIGN KEY (id_producto) REFERENCES Productos(id_producto)
);
```

3. Inserción de Datos de Prueba

Para que veas cómo funciona la relación, insertemos algunos datos:

```
-- Insertar un producto y un cliente
INSERT INTO Productos (nombre, precio) VALUES ('Laptop Dell', 850.00);
INSERT INTO `productos`(`id_producto`, `nombre`, `precio`) VALUES ('2','lenovo','3000000');
INSERT INTO Clientes (nombre_cliente, email) VALUES ('Juan Perez', 'juan@email.com');
```

```
-- Registrar una venta (Juan compra la Laptop)
-- El id_cliente 1 y id_producto 1 deben existir previamente
INSERT INTO Ventas (fecha, id_cliente, id_producto, cantidad)
VALUES ('2026-02-11', 1, 1, 1);
```

Conceptos Clave para no olvidar:

- **Primary Key (PK):** Es el identificador único de cada fila (ej. `id_producto`). No se puede repetir.
- **Foreign Key (FK):** Es una columna en una tabla que hace referencia a la PK de otra tabla. Es lo que crea el "vínculo" real.
- **Motor de almacenamiento:** Asegúrate de que tus tablas usen **InnoDB** (es el estándar en XAMPP), ya que es el que permite el uso de llaves foráneas y mantiene la integridad de los datos.

MODIFICAR: (ACTUALIZAR)

Para modificar datos en SQL, utilizamos la instrucción **`UPDATE`**. Es una de las herramientas más potentes, pero también una de las que requiere más cuidado: **¡siempre debes usar la cláusula `WHERE`!** Sin ella, cambiarías los datos de toda la tabla de un solo golpe.

Aquí tienes los ejemplos prácticos para tu base de datos de ventas:

1. Modificar un Producto (Ej. Cambio de Precio)

Imagina que la "Laptop Dell" subió de precio. Vamos a buscarla por su `id_producto`.

```
UPDATE Productos
SET precio = 899.99
WHERE id_producto = 1;
```

ELIMINAR

Para eliminar datos en SQL utilizamos la instrucción **`DELETE`**. Al igual que con el comando de modificar, la clave absoluta aquí es la cláusula **`WHERE`**. Si la olvidas, borrarás absolutamente todos los registros de la tabla.

Aquí tienes cómo aplicarlo en tu sistema de ventas:

1. Eliminar un registro específico

Si quieres borrar un producto que ya no vas a vender (por ejemplo, el producto con ID 1):

```
DELETE FROM Productos
WHERE id_producto = 1;
```

2. El problema de la "Integridad Referencial" (Error común)

Como tus tablas están conectadas, si intentas borrar un Cliente que ya tiene una Venta registrada, MySQL te dará un error de **"Foreign Key Constraint"**.

¿Por qué sucede? Para proteger la lógica de tus datos. No puedes tener una venta de un cliente que "ya no existe". Tienes dos opciones:

1. Borrar primero la venta y luego al cliente.
2. Usar un "Borrado Lógico" (explicado abajo).

3. Borrado en Cascada (La solución automática)

Si quieres que al borrar un cliente se borren automáticamente todas sus ventas, debiste definir la llave foránea con `ON DELETE CASCADE`. Si no lo hiciste al principio, puedes modificar la tabla así:

```
-- Primero eliminamos la restricción actual
ALTER TABLE Ventas DROP FOREIGN KEY ventas_ibfk_1;

-- La volvemos a crear con la propiedad de cascada
ALTER TABLE Ventas
ADD CONSTRAINT fk_cliente_venta
FOREIGN KEY (id_cliente) REFERENCES Clientes(id_cliente)
ON DELETE CASCADE;
```

Consejo de Profesional: El "Borrado Lógico"

En la vida real, casi nunca se usa `DELETE` en tablas de ventas o clientes porque perderías el historial contable. En su lugar, se agrega una columna llamada `estado`:

1. Agregas la columna: `ALTER TABLE Productos ADD COLUMN activo TINYINT(1) DEFAULT 1;`

2. Para "borrar", solo haces un update: `UPDATE Productos SET activo = 0 WHERE id_producto = 1;`
3. En tus consultas solo buscas los que tengan `activo = 1`.

Actividad

- Insertar 5 valores más en tablas: productos, clientes y ventas.
- Modifica los 5 primeros NOMBRES de en la tabla "clientes". (UPDATE) con los siguientes nombres:

Angie Gabriela Figueroa
Lesme Josue Martinez
Emily Valentina García
Sara María Perez
luis gerardo urdaneta

- Elimine el ID 5 de la tabla "clientes". (DELETE)