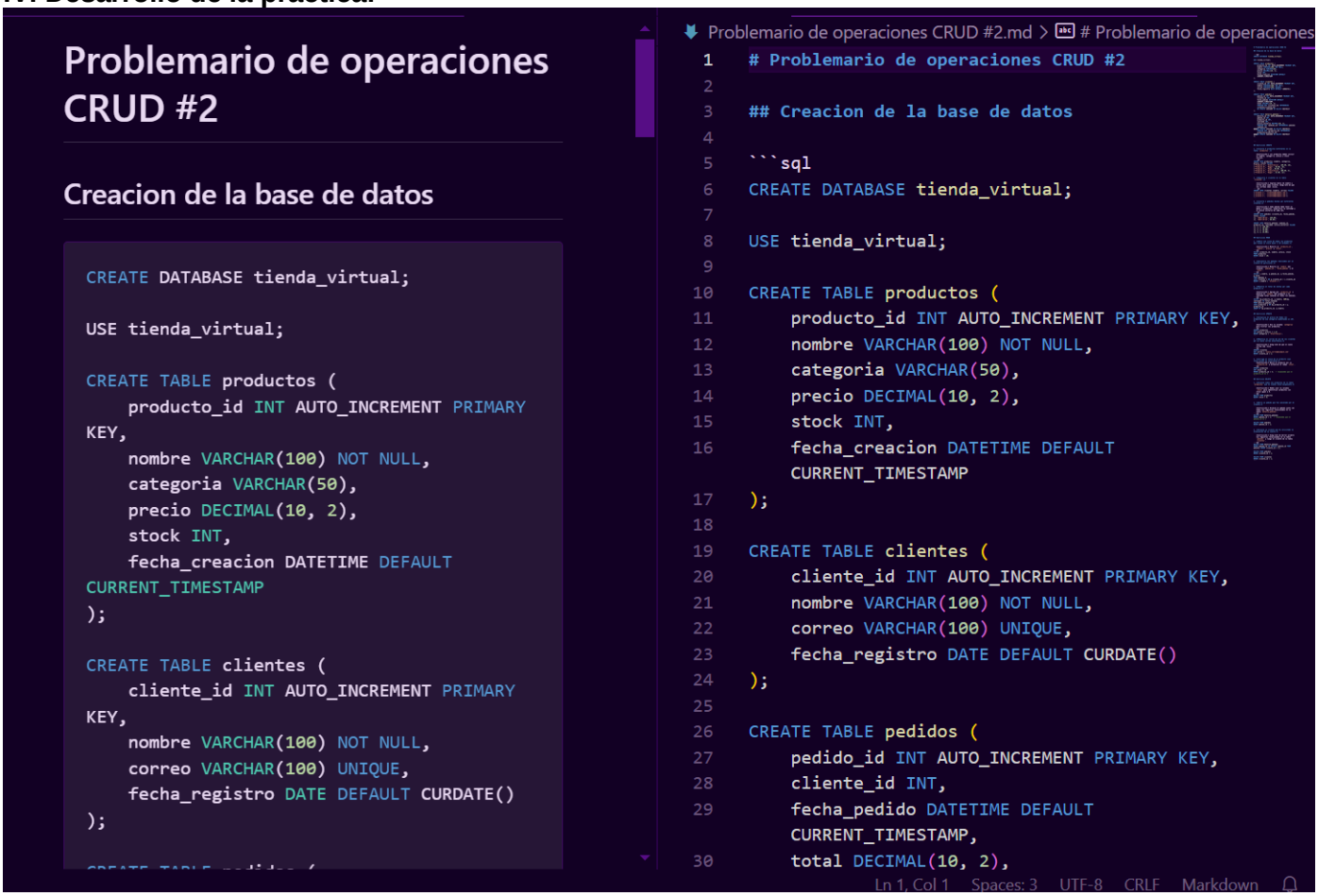
Nombre de la práctica	Problemario 2			No.	5
Asignatura:	Taller de Bases de Datos	Carrera:	ISIC	Duración de la práctica (Hrs)	2hrs

I. Competencia(s) específica(s):

II. Lugar de realización de la práctica (laboratorio, taller, aula u otro): Aula

III. Material empleado:
Computadora

IV. Desarrollo de la práctica:



Problemario de operaciones CRUD #2

Creacion de la base de datos

```sql
CREATE DATABASE tienda_virtual;

USE tienda_virtual;

CREATE TABLE productos (
    producto_id INT AUTO_INCREMENT PRIMARY
KEY,
    nombre VARCHAR(100) NOT NULL,
    categoria VARCHAR(50),
    precio DECIMAL(10, 2),
    stock INT,
    fecha_creacion DATETIME DEFAULT
CURRENT_TIMESTAMP
);

CREATE TABLE clientes (
    cliente_id INT AUTO_INCREMENT PRIMARY
KEY,
    nombre VARCHAR(100) NOT NULL,
    correo VARCHAR(100) UNIQUE,
    fecha_registro DATE DEFAULT CURDATE()
);
```

Problemario de operaciones CRUD #2.md > # Problemario de operaciones

```
1  # Problemario de operaciones CRUD #2
2
3  ## Creacion de la base de datos
4
5  ```sql
6  CREATE DATABASE tienda_virtual;
7
8  USE tienda_virtual;
9
10 CREATE TABLE productos (
11     producto_id INT AUTO_INCREMENT PRIMARY KEY,
12     nombre VARCHAR(100) NOT NULL,
13     categoria VARCHAR(50),
14     precio DECIMAL(10, 2),
15     stock INT,
16     fecha_creacion DATETIME DEFAULT
       CURRENT_TIMESTAMP
17 );
18
19 CREATE TABLE clientes (
20     cliente_id INT AUTO_INCREMENT PRIMARY KEY,
21     nombre VARCHAR(100) NOT NULL,
22     correo VARCHAR(100) UNIQUE,
23     fecha_registro DATE DEFAULT CURDATE()
24 );
25
26 CREATE TABLE pedidos (
27     pedido_id INT AUTO_INCREMENT PRIMARY KEY,
28     cliente_id INT,
29     fecha_pedido DATETIME DEFAULT
       CURRENT_TIMESTAMP,
30     total DECIMAL(10, 2),
```

Ln 1, Col 1 Spaces: 3 UTF-8 CRLF Markdown

```sql
CREATE TABLE pedidos (
    pedido_id INT AUTO_INCREMENT PRIMARY
KEY,
    cliente_id INT,
    fecha_pedido DATETIME DEFAULT
CURRENT_TIMESTAMP,
    total DECIMAL(10, 2),
    FOREIGN KEY (cliente_id) REFERENCES
clientes(cliente_id)
    ON UPDATE CASCADE ON DELETE RESTRICT
);

CREATE TABLE detalle_pedidos (
    detalle_id INT AUTO_INCREMENT PRIMARY
KEY,
    pedido_id INT,
    producto_id INT,
    cantidad INT,
    precio_unitario DECIMAL(10, 2),
    FOREIGN KEY (pedido_id) REFERENCES
pedidos(pedido_id)
    ON UPDATE CASCADE ON DELETE RESTRICT,
    FOREIGN KEY (producto_id) REFERENCES
productos(producto_id)
    ON UPDATE CASCADE ON DELETE RESTRICT
);
```

Ejercicios CREATE

blemario de operaciones CRUD #2.md > # Problemario de operaciones CRUD #2

```
1   # Problemario de operaciones CRUD #2
3   ## Creacion de la base de datos
25
26  CREATE TABLE pedidos (
27      pedido_id INT AUTO_INCREMENT PRIMARY KEY,
28      cliente_id INT,
29      fecha_pedido DATETIME DEFAULT
        CURRENT_TIMESTAMP,
30      total DECIMAL(10, 2),
31      FOREIGN KEY (cliente_id) REFERENCES
        clientes(cliente_id)
32      ON UPDATE CASCADE ON DELETE RESTRICT
33  );
34
35  CREATE TABLE detalle_pedidos (
36      detalle_id INT AUTO_INCREMENT PRIMARY KEY,
37      pedido_id INT,
38      producto_id INT,
39      cantidad INT,
40      precio_unitario DECIMAL(10, 2),
41      FOREIGN KEY (pedido_id) REFERENCES pedidos
        (pedido_id)
42      ON UPDATE CASCADE ON DELETE RESTRICT,
43      FOREIGN KEY (producto_id) REFERENCES
        productos(producto_id)
44      ON UPDATE CASCADE ON DELETE RESTRICT
45  );
46
47
48  ```
49
50  ## Ejercicios CREATE
```

Ejercicios CREATE

1. **Inserta 5 productos diferentes en la tabla `productos`.**

   Instrucción: Los productos deben incluir un nombre, categoría, precio y stock inicial.

```sql
INSERT INTO productos (nombre, categoria,
precio, stock) VALUES
('Producto A', 'Electrónica', 100.00, 20),
('Producto B', 'Hogar', 50.00, 15),
('Producto C', 'Ropa', 30.00, 5),
('Producto D', 'Electrónica', 200.00, 8),
('Producto E', 'Hogar', 25.00, 12);
```

2. **Registra 3 clientes en la tabla `clientes`.**

   Instrucción: Ingresa datos de nombre y correo para cada cliente. Asegúrate de que los correos sean únicos.

```sql
INSERT INTO clientes (nombre, correo)
VALUES
('Cliente 1', 'cliente1@example.com'),
('Cliente 2', 'cliente2@example.com'),
('Cliente 3', 'cliente3@example.com');
```

blemario de operaciones CRUD #2.md > # Problemario de operaciones CRUD

```
1   # Problemario de operaciones CRUD #2
49
50  ## Ejercicios CREATE
51
52  1. **Inserta 5 productos diferentes en la
    tabla `productos`.**
53
54     *Instrucción:* Los productos deben incluir
       un nombre, categoría, precio y stock
       inicial.
55  ``` sql
56  INSERT INTO productos (nombre, categoria,
    precio, stock) VALUES
57  ('Producto A', 'Electrónica', 100.00, 20),
58  ('Producto B', 'Hogar', 50.00, 15),
59  ('Producto C', 'Ropa', 30.00, 5),
60  ('Producto D', 'Electrónica', 200.00, 8),
61  ('Producto E', 'Hogar', 25.00, 12);
62  ```
63
64  2. **Registra 3 clientes en la tabla
    `clientes`.**
65
66     *Instrucción:* Ingresa datos de nombre y
       correo para cada cliente. Asegúrate de que
       los correos sean únicos.
67  ``` sql
68  INSERT INTO clientes (nombre, correo) VALUES
69  ('Cliente 1', 'cliente1@example.com'),
70  ('Cliente 2', 'cliente2@example.com'),
71  ('Cliente 3', 'cliente3@example.com');
72  ```
```

3. **Inserta 2 pedidos hechos por diferentes clientes.**

Instrucción: Cada pedido debe tener al menos 2 productos, especifica la cantidad y el precio unitario de cada uno.

```sql
INSERT INTO pedidos (cliente_id, fecha_pedido, total) VALUES
(1, '2024-10-22', 250.00),
(2, '2024-10-23', 80.00);

INSERT INTO detalle_pedidos (pedido_id, producto_id, cantidad, precio_unitario)
VALUES
(1, 1, 2, 100.00),
(1, 2, 1, 50.00),
(2, 3, 2, 30.00),
(2, 5, 2, 25.00);
```

Ejercicios READ

1. **Obtén una lista de todos los productos que tienen un stock mayor a 10 unidades.**

Instrucción: Muestra el `producto_id`, `nombre`, `precio` y `stock`.

```sql
SELECT producto_id, nombre, precio, stock
FROM productos
WHERE stock > 10;
```

2. **Encuentra los pedidos realizados por un cliente en particular.**

Instrucción: Muestra el `nombre` del cliente, `pedido_id`, `fecha_pedido` y el `total`.

```sql
SELECT c.nombre, p.pedido_id, p.fecha_pedido, p.total
FROM pedidos p
JOIN clientes c ON p.cliente_id = c.cliente_id
WHERE c.nombre = 'Cliente 1';
```

3. **Muestra el total de ventas por cada producto.**

Instrucción: Agrupa por `producto_id` y muestra el `nombre` del producto y la cantidad total vendida en todos los pedidos

```
# Problemario de operaciones CRUD #2
## Ejercicios CREATE

3. **Inserta 2 pedidos hechos por diferentes clientes.**

   *Instrucción:* Cada pedido debe tener al menos 2 productos, especifica la cantidad y el precio unitario de cada uno.
```sql
INSERT INTO pedidos (cliente_id, fecha_pedido, total) VALUES
(1, '2024-10-22', 250.00),
(2, '2024-10-23', 80.00);

INSERT INTO detalle_pedidos (pedido_id, producto_id, cantidad, precio_unitario) VALUES
(1, 1, 2, 100.00),
(1, 2, 1, 50.00),
(2, 3, 2, 30.00),
(2, 5, 2, 25.00);
```

Ejercicios READ

1. **Obtén una lista de todos los productos que tienen un stock mayor a 10 unidades.**

   Instrucción: Muestra el `producto_id`, `nombre`, `precio` y `stock`.
```sql
SELECT producto_id, nombre, precio, stock
```
```

3. **Muestra el total de ventas por cada producto.**

   Instrucción: Agrupa por `producto_id` y muestra el `nombre` del producto y la cantidad total vendida en todos los pedidos.

```sql
SELECT dp.producto_id, p.nombre,
SUM(dp.cantidad) AS total_vendido
FROM detalle_pedidos dp
JOIN productos p ON dp.producto_id =
p.producto_id
GROUP BY dp.producto_id, p.nombre;
```

Ejercicios UPDATE

1. **Actualiza el precio de todos los productos de una categoria aumentando un 15%.**

   Instrucción: Usa la columna `categoria` para filtrar los productos.

```sql
UPDATE productos
SET precio = precio * 1.15
WHERE categoria = 'Electrónica';
```

2. **Modifica el correo de uno de los clientes por un nuevo correo electrónico**

2. **Modifica el correo de uno de los clientes por un nuevo correo electrónico.**

   Instrucción: Asegúrate de que el nuevo correo sea único.

```sql
UPDATE clientes
SET correo = 'nuevo_correo@example.com'
WHERE cliente_id = 1;
```

3. **Corrige el stock de un producto cuyo stock actual es incorrecto.** *Instrucción:* Busca el producto por su `producto_id` y actualiza el campo `stock`.

```sql
UPDATE productos
SET stock = 10
WHERE producto_id = 3;  -- Asumiendo que el
producto_id es 3
```

Ejercicos DELETE

1. **Elimina todos los productos de la tabla `productos` que no tienen stock disponible.**

   Instrucción: Debes usar la columna `stock` para identificar productos con stock igual a 0.

```
## Ejercicios READ

3. **Muestra el total de ventas por cada producto.**

   *Instrucción:* Agrupa por `producto_id` y muestra el `nombre` del producto y la cantidad total vendida en todos los pedidos.
```sql
SELECT dp.producto_id, p.nombre, SUM(dp.cantidad) AS total_vendido
FROM detalle_pedidos dp
JOIN productos p ON dp.producto_id = p.producto_id
GROUP BY dp.producto_id, p.nombre;
```

Ejercicios UPDATE

1. **Actualiza el precio de todos los productos de una categoria aumentando un 15%.**

   Instrucción: Usa la columna `categoria` para filtrar los productos.
```sql
UPDATE productos
SET precio = precio * 1.15
WHERE categoria = 'Electrónica';
```

Problemario de operaciones CRUD #2

Ejercicios UPDATE

2. **Modifica el correo de uno de los clientes por un nuevo correo electrónico.**

   Instrucción: Asegúrate de que el nuevo correo sea único.
```sql
UPDATE clientes
SET correo = 'nuevo_correo@example.com'
WHERE cliente_id = 1;
```

3. **Corrige el stock de un producto cuyo stock actual es incorrecto.**
   Instrucción: Busca el producto por su `producto_id` y actualiza el campo `stock`.
```sql
UPDATE productos
SET stock = 10
WHERE producto_id = 3;  -- Asumiendo que el producto_id es 3
```

Ejercicos DELETE

1. **Elimina todos los productos de la tabla `productos` que no tienen stock disponible.**

   Instrucción: Debes usar la columna `stock` para identificar productos con stock igual a 0.
```

Ejercicos DELETE

1. **Elimina todos los productos de la tabla `productos` que no tienen stock disponible.**

   Instrucción: Debes usar la columna `stock` para identificar productos con stock igual a 0.

```sql
DELETE FROM productos
WHERE stock = 0;
```

2. **Borra un pedido que fue cancelado por el cliente.**

   Instrucción: Elimina el pedido junto con todos los registros relacionados en la tabla `detalle_pedidos`.

```sql
DELETE FROM detalle_pedidos
WHERE pedido_id = 1;  -- Asumiendo que el pedido_id es 1

DELETE FROM pedidos
WHERE pedido_id = 1;
```

3. **Elimina un cliente que ha solicitado la eliminación de su cuenta.**

   Instrucción: Asegúrate de borrar primero los

3. **Elimina un cliente que ha solicitado la eliminación de su cuenta.**

   Instrucción: Asegúrate de borrar primero los registros relacionados en la tabla `pedidos` y luego el cliente de la tabla `clientes`.

```sql
DELETE FROM detalle_pedidos
WHERE pedido_id IN (SELECT pedido_id FROM pedidos WHERE cliente_id = 1);

DELETE FROM pedidos
WHERE cliente_id = 1;

DELETE FROM clientes
WHERE cliente_id = 1;
```

```
1    # Problemario de operaciones CRUD #2
147
148  ## Ejercicos DELETE
149
150  1. **Elimina todos los productos de la tabla
     `productos` que no tienen stock disponible.**
151
152     *Instrucción:* Debes usar la columna
        `stock` para identificar productos con
        stock igual a 0.
153  ```sql
154  DELETE FROM productos
155  WHERE stock = 0;
156  ```
157
158  2. **Borra un pedido que fue cancelado por el
     cliente.**
159
160     *Instrucción:* Elimina el pedido junto con
        todos los registros relacionados en la
        tabla `detalle_pedidos`.
161  ```sql
162  DELETE FROM detalle_pedidos
163  WHERE pedido_id = 1;  -- Asumiendo que el
     pedido_id es 1
164
165  DELETE FROM pedidos
166  WHERE pedido_id = 1;
167  ```
168
169  3. **Elimina un cliente que ha solicitado la
     eliminación de su cuenta.**
148  ## Ejercicos DELETE
168
169  3. **Elimina un cliente que ha solicitado la
     eliminación de su cuenta.**
170
171     *Instrucción:* Asegúrate de borrar primero
        los registros relacionados en la tabla
        `pedidos` y luego el cliente de la tabla
        `clientes`.
172  ```sql
173  DELETE FROM detalle_pedidos
174  WHERE pedido_id IN (SELECT pedido_id FROM
     pedidos WHERE cliente_id = 1);
175
176  DELETE FROM pedidos
177  WHERE cliente_id = 1;
178
179  DELETE FROM clientes
180  WHERE cliente_id = 1;
181  ```
```

V. Conclusiones:

Entre mas ejercicios desarrollas sobre las operaciones CRUD mas te das cuenta de que su utilidad es eficaz para una buena consulta y una mejor organización de la información así evitando cometer algún error, pero recordando que lo que se elimina ya no se puede recuperar, por eso también hay que tener mucho cuidado con el uso que le damos a estas operaciones, ya que contienen una gran función en las bases de datos.