Nombre de la práctica	Problemario 1			No.	4
Asignatura:	Taller de Base 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 para operaciones CRUD

## CREACION DE LA BASE DE DATOS

```sql
CREATE TABLE clientes (
    id_cliente INT PRIMARY KEY AUTO_INCREMENT,
    nombre VARCHAR(100) NOT NULL,
    email VARCHAR(100) UNIQUE NOT NULL CHECK
    (email LIKE '%_@_%._%'),
    telefono VARCHAR(15) CHECK (LENGTH
    (telefono) >= 10),
    direccion VARCHAR(255) NOT NULL
);
```

Ejercicios CREATE

1. Inserta un cliente válido en la tabla.
```sql
INSERT INTO clientes (nombre, correo,
telefono) VALUES ('Juan Pérez', 'juan@example.
com', '5551234567');
```
2. Inserta un cliente sin especificar el campo
`id_cliente`.
``` sql
INSERT INTO clientes (nombre, correo,
telefono) VALUES ('Ana García', 'ana@example.
com', '5559876543');
```
```

Problemario para operaciones CRUD

CREACION DE LA BASE DE DATOS

```sql
CREATE TABLE clientes (
    id_cliente INT PRIMARY KEY
AUTO_INCREMENT,
    nombre VARCHAR(100) NOT NULL,
    email VARCHAR(100) UNIQUE NOT NULL
CHECK (email LIKE '%_@_%._%'),
    telefono VARCHAR(15) CHECK
(LENGTH(telefono) >= 10),
    direccion VARCHAR(255) NOT NULL
);
```

Ejercicios CREATE

1. Inserta un cliente válido en la tabla.

```sql
INSERT INTO clientes (nombre, correo,
telefono) VALUES ('Juan Pérez',
'juan@example.com', '5551234567');
```

```
## Ejercicios CREATE
com', '5551234567');
```

2. Inserta un cliente sin especificar el campo `id_cliente`.

```sql
INSERT INTO clientes (nombre, correo,
telefono) VALUES ('Ana García', 'ana@example.
com', '5559876543');
```

3. Intenta insertar un cliente con un formato de correo incorrecto (debería fallar).

```sql
INSERT INTO clientes (nombre, correo,
telefono) VALUES ('Luis Torres', 'luis@com',
'5557654321');
```

4. Inserta múltiples clientes en una sola consulta.

```sql
INSERT INTO clientes (nombre, correo,
telefono) VALUES
('Carlos Fernández', 'carlos@example.com',
'5556543210'),
('María López', 'maria@example.com',
'5555432109');
```

5. Inserta un cliente con un número de teléfono de menos de 10 caracteres (debería fallar).

```sql
('María López', 'maria@example.com',
'5555432109');
```

5. Inserta un cliente con un número de teléfono de menos de 10 caracteres (debería fallar).

```sql
INSERT INTO clientes (nombre, correo,
telefono) VALUES ('Pedro Ruiz', 'pedro@example.
com', '555123');
```

Ejercicios READ

1. Consulta todos los registros de la tabla `clientes`.

```sql
SELECT * FROM clientes;
```

2. Consulta el `nombre` y `email` de todos los clientes.

```sql
SELECT nombre, correo FROM clientes;
```

3. Consulta los clientes cuyo número de teléfono empiece con "555".

```sql
SELECT * FROM clientes WHERE telefono LIKE
```

````
SELECT nombre, correo FROM clientes;
```

3. Consulta los clientes cuyo número de teléfono empiece con "555".
```sql
SELECT * FROM clientes WHERE telefono LIKE '555%';
```

4. Consulta los clientes cuyo nombre contenga "López".
```sql
SELECT * FROM clientes WHERE nombre LIKE '%López%';
```

5. Consulta los clientes ordenados por `nombre` en orden ascendente.
```sql
SELECT * FROM clientes ORDER BY nombre ASC;
```

6. Consulta el `email` de los clientes cuyo id sea par.
```sql
SELECT correo FROM clientes WHERE id_cliente % 2 = 0;
```
````

3. Consulta los clientes cuyo número de teléfono empiece con "555".

```sql
SELECT * FROM clientes WHERE telefono LIKE '555%';
```

4. Consulta los clientes cuyo nombre contenga "López".

```sql
SELECT * FROM clientes WHERE nombre LIKE '%López%';
```

5. Consulta los clientes ordenados por `nombre` en orden ascendente.

```sql
SELECT * FROM clientes ORDER BY nombre ASC;
```

6. Consulta el `email` de los clientes cuyo id sea par.

```sql
SELECT correo FROM clientes WHERE id_cliente % 2 = 0;
```

````
## Ejercicios READ

7. Consulta los clientes con direcciones que contengan más de 10 caracteres.
```sql
SELECT * FROM clientes WHERE LENGTH(direccion) > 10;
```
````

7. Consulta los clientes con direcciones que contengan más de 10 caracteres.

```sql
SELECT * FROM clientes WHERE LENGTH(direccion) > 10;
```

Ejercicios UPDATE

````
## Ejercicios UPDATE

1. Actualiza el número de teléfono de un cliente específico.
```sql
UPDATE clientes SET telefono = '5550000000' WHERE id_cliente = 1;
```

2. Cambia el `email` de un cliente con un `id_cliente` específico.
```sql
UPDATE clientes SET correo = 'nuevo_email@example.com' WHERE id_cliente = 2;
```

3. Intenta actualizar el correo de un cliente a un email que ya existe (debería fallar).
```sql
UPDATE clientes SET correo = 'juan@example.com' WHERE id_cliente = 3;  -- Asumiendo que
````

1. Actualiza el número de teléfono de un cliente específico.

```sql
UPDATE clientes SET telefono = '5550000000' WHERE id_cliente = 1;
```

2. Cambia el `email` de un cliente con un `id_cliente` específico.

```sql
UPDATE clientes SET correo = 'nuevo_email@example.com' WHERE id_cliente = 2;
```

3. Intenta actualizar el correo de un cliente a un email que ya existe (debería fallar).

```sql
UPDATE clientes SET correo =
```

```sql
3. Intenta actualizar el correo de un cliente a un email que ya existe (debería fallar).
```sql
UPDATE clientes SET correo = 'juan@example.com' WHERE id_cliente = 3;  -- Asumiendo que el correo ya existe
```

4. Actualiza la dirección de todos los clientes cuyos nombres contengan "López".
```sql
UPDATE clientes SET direccion = 'Nueva Dirección' WHERE nombre LIKE '%López%';
```

5. Incrementa los `id_cliente` de todos los clientes en 10 (esto es solo un ejercicio teórico).
``` sql
UPDATE clientes SET id_cliente = id_cliente + 10;  -- Esto no es recomendable en la práctica
```

Ejercicios DELETE

1. Elimina un cliente específico con un `id_cliente` dado.
```sql
```

3. Intenta actualizar el correo de un cliente a un email que ya existe (debería fallar).

```
UPDATE clientes SET correo = 'juan@example.com' WHERE id_cliente = 3;  -- Asumiendo que el correo ya existe
```

4. Actualiza la dirección de todos los clientes cuyos nombres contengan "López".

```
UPDATE clientes SET direccion = 'Nueva Dirección' WHERE nombre LIKE '%López%';
```

5. Incrementa los `id_cliente` de todos los clientes en 10 (esto es solo un ejercicio teórico).

```
UPDATE clientes SET id_cliente = id_cliente + 10;  -- Esto no es recomendable en la práctica
```

Ejercicios DELETE

1. Elimina un cliente específico con un `id_cliente` dado.

```sql
## Ejercicios DELETE

1. Elimina un cliente específico con un `id_cliente` dado.
```sql
DELETE FROM clientes WHERE id_cliente = 1;
```

2. Elimina todos los clientes que tengan un número de teléfono que empiece con "555".
```sql
DELETE FROM clientes WHERE telefono LIKE '555%';
```

3. Elimina todos los clientes cuyo nombre contenga "Gómez".
``` sql
DELETE FROM clientes WHERE nombre LIKE '%Gómez%';
```

4. Elimina todos los clientes con direcciones que contengan menos de 10 caracteres.
```sql
DELETE FROM clientes WHERE LENGTH(direccion) < 10;
```
```

## Ejercicios DELETE

1. Elimina un cliente específico con un `id_cliente` dado.

```
DELETE FROM clientes WHERE id_cliente = 1;
```

2. Elimina todos los clientes que tengan un número de teléfono que empiece con "555".

```
DELETE FROM clientes WHERE telefono LIKE '555%';
```

3. Elimina todos los clientes cuyo nombre contenga "Gómez".

```
DELETE FROM clientes WHERE nombre LIKE '%Gómez%';
```

4. Elimina todos los clientes con direcciones que contengan menos de 10 caracteres.

```
DELETE FROM clientes WHERE LENGTH(direccion) < 10;
```
```

V. Conclusiones:

El uso de la operaciones CRUD en las bases de datos nos facilita la filtración de los dato o el llenado de algunos campos específicos sin hacer tantas operaciones o una sintaxis tan grande, de igual manera nos ayudan a eliminar los datos de una manera mas eficaz y especifica, ya que tenemos que escribir y detallar de donde queremos borrar ese dato, especificando la tabla donde este se encuentra.