Semana 2 – CRUD completo

Objetivos de la semana

- Consolidar el manejo de SQL como lenguaje para trabajar con datos de manera profesional.
- Aprender a realizar las cuatro operaciones básicas del CRUD: **Crear, Leer, Actualizar y Borrar**.
- Utilizar filtros avanzados para consultas más precisas.
- Comprender cómo se calculan resúmenes estadísticos a través de funciones de agregación.
- Entender el rol del GROUP BY para agrupar información y obtener resultados significativos.

1. Repaso de la semana anterior

En la primera semana aprendimos a **crear tablas** con CREATE TABLE, insertar datos con INSERT y consultarlos de forma básica con SELECT.

Ese fue el primer paso: *poner en marcha la base de datos y comenzar a usarla*.

Pero en la vida real, una base de datos está en constante cambio. Se agregan registros, otros se modifican, algunos dejan de tener sentido y deben eliminarse.

Además, muchas veces no necesitamos mirar cada registro individualmente, sino obtener un **resumen** de lo que está pasando (ej. cuántos libros tiene cada autor, o cuál es el promedio de ventas).

De todo eso se ocupa el contenido de esta semana.

2. Consultas más elaboradas con SELECT

El comando SELECT es mucho más que "traer todo de una tabla". Nos permite seleccionar con precisión qué registros queremos.

- **Operadores lógicos (AND, OR):** sirven para combinar condiciones.

```
SELECT * FROM Libros
WHERE Precio > 500 AND AutorID = 2;
```

Este ejemplo devuelve los libros de un autor específico que superen cierto precio.

- **Listas con IN:** práctico cuando tenemos varias opciones posibles.

```
SELECT * FROM Libros
WHERE AutorID IN (1, 3, 5);
```

Se lee como "el AutorID debe ser 1, 3 o 5".

- **Búsqueda por patrón con LIKE:** permite trabajar con coincidencias parciales.

```
SELECT * FROM Clientes
WHERE Email LIKE '%gmail.com';
```

El símbolo % funciona como "comodín" que representa cualquier cantidad de caracteres.

3. Operaciones CRUD

SQL nos da comandos para modificar datos ya existentes, no solo para insertarlos. Aquí es donde aparecen **errores comunes** si no se tiene cuidado.

a) INSERT múltiple

Hasta ahora insertamos un registro por vez. También se pueden agregar varios juntos:

```
INSERT INTO Autores (AutorID, Nombre)
VALUES
(2, 'Julio Cortázar'),
(3, 'Gabriel García Márquez');
```

Esto ahorra tiempo y reduce la cantidad de sentencias ejecutadas.

b) UPDATE

Sirve para modificar información.

```
UPDATE Libros
SET Precio = 600
WHERE LibroID = 10;
```

Advertencia: si olvidamos el WHERE, se actualizarán **todos** los registros de la tabla.

c) DELETE

Elimina registros de forma permanente.

```
DELETE FROM Libros
WHERE LibroID = 10;
```

Igual que en UPDATE, nunca debe ejecutarse sin WHERE, salvo que queramos vaciar toda la tabla (y eso casi nunca es el caso).

4. Funciones de agregación

En lugar de mirar cada registro uno por uno, muchas veces queremos ver un **resumen numérico** de los datos. Ahí aparecen las funciones de agregación:

- COUNT(*): cuántos registros hay.
- SUM(Precio): suma de valores de una columna.
- AVG(Precio): promedio de los valores de una columna.
- MAX(Precio), MIN(Precio): el mayor y el menor de una columna.

Ejemplo:

```
SELECT COUNT(*) AS CantidadLibros FROM Libros;
```

Esto devuelve un solo número: la cantidad de libros en la tabla.

5. Agrupamiento con GROUP BY

Aquí llegamos a una herramienta muy poderosa.

¿Para qué sirve?

El GROUP BY permite **agrupar registros que comparten un mismo valor en una columna** y calcular un resumen para cada grupo.

Ejemplo típico: cuántos libros tiene cada autor.

Caso práctico

Supongamos la tabla **Libros**:

LibroID	Titulo	Precio	AutorID
1	La Casa de los Espíritus	500	1
2	Paula	400	1
3	Rayuela	600	2
4	Bestiario	350	2
5	Cien Años de Soledad	700	3

Consulta:

```
SELECT AutorID, COUNT(*) AS CantidadLibros
FROM Libros
```

```
GROUP BY AutorID;
```

Resultado:

AutorID	CantidadLibros
1	2
2	2
3	1

Ahora cada autor aparece **una sola vez** con el total de libros asociados.

Otro ejemplo con promedio de precios:

```
SELECT AutorID, AVG(Precio) AS PrecioPromedio
FROM Libros
GROUP BY AutorID;
```

Resultado:

AutorID	PrecioPromedio
1	450
2	475
3	700

Esto muestra cuánto cuestan en promedio los libros de cada autor.

¿Qué pasa si no usamos GROUP BY?

Si ejecutáramos:

```
SELECT AutorID, COUNT(*) FROM Libros;
```

Obtenemos un error: *"Columna AutorID no está en GROUP BY ni en una función de agregación"*.

Esto pasa porque **cuando hay agregaciones, todas las demás columnas deben agruparse explícitamente**.

6. Buenas prácticas

- **Probar antes:** si vas a hacer un DELETE o UPDATE, primero ejecutá un SELECT con la misma condición para verificar que afecta a los registros correctos.

- **Siempre con WHERE:** salvo en casos muy específicos, nunca ejecutar UPDATE o DELETE sin condición.
- **Nombrar resultados:** usar alias (AS) para que las columnas agregadas tengan nombres más claros.
- **Insertar de manera explícita:** indicar siempre las columnas en INSERT.

7. Actividad práctica

1. Insertar tres clientes nuevos en la tabla **Clientes**.
2. Consultar los clientes cuyo nombre empiece con "A".
3. Actualizar el email de un cliente.
4. Eliminar un cliente específico con DELETE.
5. Insertar varios libros para distintos autores y luego:
    - Mostrar cuántos libros tiene cada autor (COUNT + GROUP BY).
    - Calcular el precio promedio de los libros por autor (AVG + GROUP BY).
    - Mostrar el autor que tiene el libro más caro (MAX + GROUP BY).