Guía rápida de sintaxis de SQL en MySQL

1. Tipos de datos en MySQL

Númericos:

- **INT**: Entero con rango de -2,147,483,648 a 2,147,483,647.
- **TINYINT**: Entero pequeño (-128 a 127).
- **SMALLINT**: Entero mediano (-32,768 a 32,767).
- **BIGINT**: Entero grande (-9,223,372,036,854,775,808 a 9,223,372,036,854,775,807).
- **FLOAT**(m,d): Número decimal de precisión simple.
- **DOUBLE**(m,d): Número decimal de doble precisión.

Cadenas:

- **CHAR(n)**: Cadena de longitud fija, de 0 a 255 caracteres.
- **VARCHAR(n)**: Cadena de longitud variable, de 0 a 65,535 caracteres.
- **TEXT**: Cadena de longitud larga, hasta 65,535 caracteres.

Fechas y Tiempos:

- **DATE**: Fecha (AAAA-MM-DD).
- **TIME**: Hora (HH:MM).
- **DATETIME**: Fecha y hora (AAAA-MM-DD HH:MM).

---

2. CREAR y ELIMINAR Base de Datos y Tablas

Crear una base de datos:

```
CREATE DATABASE nombre_base_datos;
```

Eliminar una base de datos:

```
DROP DATABASE nombre_base_datos;
```

Crear una tabla:

```
CREATE TABLE nombre_tabla (
    columna1 INT,
    columna2 VARCHAR(255),
    columna3 DATE
);
```

Eliminar una tabla:

```
DROP TABLE nombre_tabla;
```

---

3. Crear Usuario y Asignación de Permisos

Crear usuario:

```
CREATE USER 'nombre_usuario'@'localhost' IDENTIFIED BY 'contraseña';
```

Asignación de permisos:

```
GRANT ALL PRIVILEGES ON nombre_base_datos.* TO
'nombre_usuario'@'localhost';
```

Revocar permisos:

```
REVOKE ALL PRIVILEGES ON nombre_base_datos.* FROM
                        'nombre_usuario'@'localhost';
```

---

4. CONSULTAS SQL: SELECT

Selecciona datos de una o varias tablas.

```
SELECT columna1, columna2 FROM nombre_tabla;
SELECT * FROM nombre_tabla WHERE columna1 = valor;
```

Condicionales

- **=**: Igual a.
- **<>**: Distinto de.
- **LIKE**: Coincidencia con un patrón.
- **IS NULL**: Es nulo.
- **IS NOT NULL**: No es nulo.
- **IN**: Dentro de un conjunto de valores.
- **NOT IN**: No está dentro de un conjunto.
- **AND** / **OR**: Operadores lógicos para combinar condiciones.

Ejemplo:

```
SELECT * FROM clientes WHERE ciudad = 'Madrid' AND edad >= 18;
SELECT * FROM productos WHERE precio BETWEEN 10 AND 50;
```

Comodines

- **LIKE '%cadena%'**: Cualquier cadena que contenga 'cadena'.
- **LIKE 'cadena%'**: Cualquier cadena que empiece con 'cadena'.
- **_**: Sustituye un carácter.

Ejemplo:

```
SELECT * FROM empleados WHERE nombre LIKE 'A%';
```

---

5. INSERT, UPDATE y DELETE

- **INSERT:** Inserta una nueva fila en una tabla.

```
INSERT INTO nombre_tabla (columna1, columna2) VALUES (valor1, valor2);
```

- **UPDATE:** Actualiza datos existentes en una tabla.

```
UPDATE nombre_tabla SET columna1 = nuevo_valor WHERE columna2 = criterio;
```

- **DELETE:** Elimina datos de una tabla.

```
DELETE FROM nombre_tabla WHERE columna = criterio;
```

---

6. Subconsultas

Permiten hacer consultas dentro de otras consultas.

```
SELECT nombre FROM clientes WHERE id IN
          (SELECT id_cliente FROM pedidos WHERE total > 100);
```

---

7. JOINs (Uniones de Tablas)

- **INNER JOIN:** Une filas de ambas tablas donde hay coincidencias.

```
SELECT * FROM tabla1 INNER JOIN tabla2 ON tabla1.columna = tabla2.columna;
```

- **LEFT JOIN:** Devuelve todas las filas de la primera tabla, y las coincidencias de la segunda (si existen).

```
SELECT * FROM tabla1 LEFT JOIN tabla2 ON tabla1.columna = tabla2.columna;
```

- **RIGHT JOIN:** Devuelve todas las filas de la segunda tabla y las coincidencias de la primera (si existen).

```
SELECT * FROM tabla1 RIGHT JOIN tabla2 ON tabla1.columna = tabla2.columna;
```

- **FULL JOIN:** Devuelve todas las filas cuando hay coincidencia en una tabla u otra. No está soportado en MySQL directamente, pero se puede simular con `UNION`.

```
SELECT * FROM tabla1 LEFT JOIN tabla2 ON tabla1.columna = tabla2.columna
UNION
SELECT * FROM tabla1 RIGHT JOIN tabla2 ON tabla1.columna = tabla2.columna;
```

---

8. Ejemplos Adicionales

Crear una tabla con claves primarias y ajenas:

```
CREATE TABLE empleados (
    id INT PRIMARY KEY,
    nombre VARCHAR(50),
    id_departamento INT,
    FOREIGN KEY (id_departamento) REFERENCES departamentos(id)
);
```

Funciones Agregadas

- **COUNT**: Cuenta el número de filas.
- **SUM**: Suma los valores de una columna.
- **AVG**: Calcula el promedio.
- **MAX**: Encuentra el valor máximo.
- **MIN**: Encuentra el valor mínimo.

Ejemplo:

```
SELECT COUNT(*) FROM empleados;
SELECT AVG(salario) FROM empleados;
```

Agrupamiento con GROUP BY y HAVING

```
SELECT departamento, COUNT(*) FROM empleados GROUP BY departamento;
SELECT departamento, AVG(salario) FROM empleados GROUP BY departamento
HAVING AVG(salario) > 30000;
```

Sintaxis y características específicas de MySQL:

1. **Tipos de datos específicos de MySQL**:
    - MySQL tiene algunos tipos de datos con características específicas, aunque muchos son comunes a otras bases de datos.
    - Ejemplo: `TINYINT`, `TEXT`, y el uso de `VARCHAR(n)` hasta 65,535 caracteres son particulares de MySQL (aunque `VARCHAR` es común en SQL, otros SGBD pueden tener diferentes limitaciones en longitud).
2. **`LIMIT`** para restringir el número de resultados**:
    - MySQL utiliza `LIMIT` para especificar el número de filas devueltas en una consulta, algo común también en PostgreSQL. Otras bases de datos utilizan `FETCH FIRST` o `TOP`.
3. **No existe `FULL JOIN` nativo en MySQL**:
    - MySQL no soporta `FULL JOIN` directamente, como se mencionó en el documento. En MySQL, se simula un `FULL JOIN` usando la combinación de `LEFT JOIN` y `RIGHT JOIN` con `UNION`.
4. **Declaración de variables de sesión (`@variable`)**:
    - En MySQL, es común ver el uso de variables de sesión utilizando `@`, lo cual no es compatible con otros sistemas de bases de datos SQL.
5. **Funciones de texto, fecha y agregación que son específicas de MySQL**:
    - Algunas funciones de MySQL como `NOW()`, `CURDATE()`, `CONCAT()`, `IFNULL()` y `GROUP_CONCAT()` son específicas o tienen comportamiento único en MySQL.