Reglas para escribir sentencias SQL.

Trataremos algunas de las **reglas** a seguir cuando se escriben sentencias SQL, dentro de MySQL:

✓ **Valores literales: números, cadenas y de fecha y hora.**

Una **cadena** es un conjunto de caracteres encerrado entre comillas simples o dobles. Dentro de una cadena se puede utilizar un carácter de escape para señalar que los caracteres que vienen detrás tienen un significado especial. Estas secuencias empiezan con una barra invertida o contrabarra (****)

Ejemplo: Si las cadenas normalmente deben escribirse entre comillas, cuando las comillas forman parte de la cadena que quiero mostrar, deberé indicarlo. Si ejecutamos estas consultas SQL:

SELECT "Santander";	lo que veremos en pantalla será Santander
SELECT "\"Santander\"";	lo que veremos en pantalla será "Santander"

En cuanto a los **números** pueden ser: enteros, decimales y en coma flotante. los enteros se representan como secuencias de dígitos. Los decimales utilizan '**.**' como separador decimal y los flotantes se especifican usando la notación **E** ó **e**, que significa *10 elevado a*. Cada tipo de número puede estar precedido con '**-**' para indicar un valor negativo.

Ejemplo:

Enteros:			**Decimales**:			**Coma flotante:**			
23	-18	3000	23.56	-18.00	0.003	4.6E2	4.6e2	-4.6E2	4.6E-2

De **fecha y hora**: se expresan com cadenas encerradas entre comillas o como enteros. Representan la fecha, la hora o una combinación de ambas.

Ejemplo:

Fecha:	**Hora**:	**Fecha y Hora:**
"20101106"	"17:43:55"	"20101106174355"
"2010-11-06"	'17:43:55'	'2010-11-06 17:43:55'
20101106	174355	20101106174355

✓ **Valores NULL**

El valor Null significa "no hay dato" pero NO es lo mismo que una cadena vacía para columnas de tipo cadena, o un 0 para las columnas de tipo numérico

Ejemplo: No es lo mismo insertar en la fila de un cliente NULL en su número de teléfono, que insertar **" "**. En el primer caso interpretaremos que el cliente no tiene teléfono y en el segundo que lo desconocemos.

✓ **Identificadores**

Son los nombres de bases de datos, tablas, índices, columnas y alias .

- Un nombre es un conjunto de caracteres alfanuméricos cuya **longitud máxima** es de 64 bits, excepto en el alias que es de 255.
- Con relación a los ***caracteres válidos***, serán válidos todos los caracteres, excepto en el caso de bases de datos y tablas que serán válidos los caracteres permitidos en los nombres de directorios y ficheros, excepto: `**\**` , `**/**` y `**.**`. Pueden empezar por cualquier carácter pero no pueden consistir solo en números.
- En cuanto a la ***distinción entre mayúsculas y minúsculas,*** en el caso de los nombres de bases de datos y de tablas: ambos objetos corresponden a directorios y ficheros a la hora de almacenarse por tanto la distinción entre mayúsculas y minúsculas depende del sistema operativo. En columnas e índices el uso de mayúsculas o minúsculas es indiferente. En los alias deberán especificarse tal y como se hayan definido.

Ejemplo: Son nombres válidos:

CLIENTES	$CARGO	1TOTAL	TOTAL_

✓ **Palabras reservadas.**

La mayoría de ellas están prohibidas por el estándar SQL para ser empleadas como nombres de columnas y/o tablas Una palabra reservada puede emplearse como identificador si se la delimita con comillas. .Las palabras reservadas NO distinguen entre mayúsculas y minúsculas.

Ejemplo: Son palabras reservadas:
CREATE integer WHERE COMMIT

✓ **Comentarios.**

De una sola línea: Pueden empezar con **#** o con **–** . De varias líneas: desde **/*** hasta ***/**

Ejemplo:
```
/* Este texto sería considerado un comentario
por MySQL*/
```

✓ **Operadores.**

MySQL maneja los operadores habituales en cualquier lenguaje de programación: aritméticos, lógicos y de comparación.

Operadores aritméticos: sirven para realizar operaciones aritméticas formando parte de expresiones en las que intervienen valores de columnas, constantes, funciones, etc.

Se pueden combinar entre sí y se pueden utilizar paréntesis para cambiar el orden de prioridad de los operadores.

Operadores de comparación: se utilizan para comparar un valor con otro obteniendo como resultado de la comparación verdadero (TRUE=1) o falso (FALSE=0) o nulo (NULL)

Operadores lógicos: se utilizan para combinar y comparar expresiones. Devuelven verdadero (TRUE=1) si la expresión es verdadera y falso (FALSE=0) si la expresión es falsa. Si una de las expresiones es nula devuelve NULL.

Tipos de operadores

Operadores aritméticos	Operadores de comparación	Operadores lógicos
+ suma	= igual a	NOT devuelve TRUE cuando la expresión es falsa.
- resta	> mayor que	AND devuelve TRUE cuando las dos expresiones que enlaza son verdaderas
* multiplicación	< menor que	OR devuelve TRUE cuando una de las dos expresiones es verdadera.
/ división	>= mayor o igual que	
% módulo: devuelve el resto de una división	<= menor o igual que	
	!= < > distinto de	
	< = > igual a y además NULL.. *Es similar a = pero devuelve 1 en lugar de NULL si los valores son nulos y 0 si uno es nulo*.	

Prioridad de los operadores: Las operaciones se evalúan de izquierda a derecha, según el orden de prioridad de los operadores, pero es conveniente usar paréntesis paa forzar el orden en que se deben evaluar. La prioridad por defecto es: **OR, AND, NOT, =, < = >, >=, >, <=, <, <>, !=, -, +, *, /, %**

Ejemplos:
SELECT (23+56)*2; Da como resultado 158
SELECT 10 < 20; Devuelve 1 porque es verdadero.
SELECT NOT 0; Devuelve 1 porque la expresión es 0 (falsa)
SELECT 30>20 AND 10=10 Devuelve 1 porque las dos expresiones son verdaderas
SELECT 30>20 OR 10<10 Devuelve 1 porque una de las dos expresiones es verdadera