###### 1. Guía creación de la base de datos con SQL

###### Comentarios en SQL

**--comentario corto**

**/\* comentario largo**

**y que salta de línea \*/**

###### Crear base de datos, usarla, y añadirle una tabla

**create database BaseDeEjemplo**

**use BaseDeEjemplo**

**create table TablaDeEjemplo**

**(**

**NombreDelCampo varchar(50),**

**NombreDelCampo char(6) identity primary key,**

**NombreDelCampo datetime,**

**NombreDelCampo int**

**);**

###### Eliminar Tabla

**drop table TablaDeEjemplo**

**IF OBJECT\_ID('TablaDeEjemplo')is not null**

**drop table TablaDeEjemplo**

###### Añadir registros

**insert into TablaDeEjemplo**

**(NombreCampo1, NombreCampo2, ... , nombrecampoN)**

**values (valorcampo1, valorcampo2, ..., valorcampoN);**

###### Borrar registros

**delete from NombreTabla**

**where CONDICIÓN**

###### Actualizar registros

**update NOMBRETABLA**

**set CAMPO=nuevovalor;**

**where CONDICIÓN**

###### Sacar información de una tabla

**select (lista de campos separados por comas)**

**from NombreTabla**

**where Campo1 = x**

###### Operadores relacionales

**= igual**

**<> distinto**

**> mayor**

**< menor**

**>= mayor o igual**

**<= menor o igual**

###### Null

**null** significa "dato desconocido" o "valor inexistente". No es lo mismo que un valor "0", una

cadena vacía o una cadena literal "null".

Podemos asignarlo directamente, colocando NULL en la lista de valores de un insert o no asignándole valor al no añadir el campo en la lista de campos del insert.

A los campos de una tabla se les puede añadir una especificación que indica si se admiten o no valores null.

**create table NOMBRETABLA**

**(**

**...**

**campoA tipo null,**

**campoB tipo not null,**

**...**

**)**

El campoA puede no asignarse ( o asignarse a NULL) y el campoB es obligatorio, siempre hay que darle valor y nunca puede ser NULL.

No es lo mismo NULL que vacío:

*Ejemplo:*

**select nombrecliente**

**from FAC\_T\_Cliente**

**where DatosCliente =''**

###### Default

Podemos establecer un valor que se usará en el caso que no

especifiquemos valor al campo en el insert. Se denominará valor por

defecto.

La cláusula en la creación de la tabla será:

**create table nombretabla**

**(**

**campo tipo default ‘valorpordefecto’**

**);**

###### Clave Primaria

**create table NOMBRETABLA**

**(**

**CAMPO TIPO identity primary key,**

**CAMPO TIPO,**

**...**

**);**

Para permitir ingresar un valor en un campo de identidad se debe activar la opción " **identity\_insert**":

**set identity\_insert NombreTabla on;**

Es decir, podemos ingresar valor en un campo "identity" cambiando la opción " **identity\_insert**" en " **on**". Cuando " **identity\_insert**" está en  **on**, las instrucciones "**insert**" deben especificar un valor .

###### Concatenar cadenas y Alias

**--apellidos y nombre juntos**

**select DNICliente,Apellidos+', '+Nombre**

**from ALQ\_Cliente;**

**--nombre, apellidos y datos juntos**

**select DNICliente,**

**Nombre +' '+Apellidos+' con datos: '+DatosCliente**

**from ALQ\_Cliente;**

**select OperaciónEntreCampos as NombreDeLaColumna ...**

###### Fecha y Hora

Para almacenar valores fecha y hora usamos la sintaxis: **mdy** y todas las variantes posibles.

*Ejemplo*: **convert (varchar,campoFechaHora,103) ⇢** Nos lo muestra en formato dd/mm/aaaa

create table NombreTabla

(

CodArticulo varchar (50) default ‘ValorPorDefecto’

NombreArticulo varchar(50),

);

insert TipoPlato2

(TipoPlato, Agrupa)

values ('Ensalada','Plato')

###### Funciones Fecha y Hora

**select getdate();**

**--retorna la fecha y hora actuales**

**select datepart(month,getdate());**

**--Retorna el número de mes actual**

**select datepart(day,getdate());**

**--Retorna el día actual**

**select datepart(hour,getdate());**

**--Retorna la hora actual**

**select datename(month,getdate());**

**--Retorna el nombre del mes actual**

**select datename(dw,getdate());**

**--Retorna el nombre del día de la semana de la fecha actual**

El resto de funciones están en a partir de la página 52 en el siguiente [PDF](https://drive.google.com/file/d/1nlpQaGgajAjecxFRxLREgbuGpZwrOdSS/view?usp=share_link).

###### Funciones de cadenas de caracteres y matemáticas

**select substring('Buenas tardes',8,6);**

**--extraer una subcadena dando inicio y longitud**

**--retorna "tardes".**

**select str(-123.456,7,3);**

**--convierte el valor numérico "123.456" a cadena,**

**--especificando 7 de longitud y 3 decimales**

**--retorna '-123.46';**

Hay muchas funciones, están a partir de la página 37 en el siguiente [PDF](https://drive.google.com/file/d/1nlpQaGgajAjecxFRxLREgbuGpZwrOdSS/view?usp=share_link).

###### Ordenar registros

Podemos ordenar el resultado de un "**select**" para que los registros se muestren ordenados por algún

campo, para ello usamos la cláusula "**order by**".

La sintaxis básica es la siguiente:

**select lista de campos from NOMBRETABLA**

**order by lista de campos que definen el orden;**

Aparecen los registros ordenados alfabéticamente por el campo especificado o numéricamente de menor a mayor.

###### 

###### Operadores Lógicos

**and**

**or**

**not**

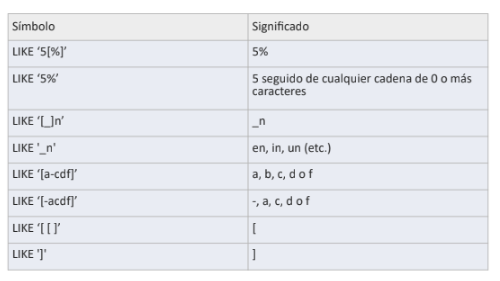
**between**

###### Búsqueda de Patrones LIKE

Se utiliza "like" para averiguar si el valor de un campo coincide con un patrón establecido.

Formato como operador condicional (para los where, por ejemplo):

**campo LIKE patrón**



###### (count) Funciones agrupadoras

En SQL tendremos funciones que nos realizan operaciones con conjuntos de registros.

La más básica es la que nos permite contar el número de registros.

**select count(\*) from tabla;**

**count(distinct campo)**

###### (sum) Funciones agrupadoras

Otra función agrupadora es la que nos permite sumar los valores de un campo numérico

Su formato es:

**select sum(campo)**

**from tabla**

El resto de funciones agrupadoras son: **max, min, avg, stdev y var.**

Están en la página 19 del siguiente [PDF](https://drive.google.com/file/d/17vShZ72p4HFXnjSs4XG3gCwhWbLJBNWC/view?usp=share_link).

###### Cláusula Top

La cláusula TOP se utiliza al inicio de la lista de campos del select para indicar el número de

registros que queremos sacar.

**select TOP nº lista de campos...**

Se puede especificar en vez de un número exacto, un porcentaje de registros.

**select TOP nº percent lista de campos...**

Podremos tener en cuenta los empates. Para ello tenemos la cláusula:

**TOP nº WITH TIES**

Nos mostrará el número de registros especificado, añadiendo los que empaten con el último en criterio de ordenación.

###### Group By

Podemos agrupar registros de nuestra tabla, de manera que obtengamos un único registro por cada grupo.

Para establecer los grupos lo haremos especificando los campos que lo generan.

Aparecerá un único resultado para cada valor diferente de los campos del grupo.

**select ....**

**from ....**

**group by campo1, campo2,...**

###### Having

Relacionado con el group by tenemos la posibilidad de realizar filtros de grupos.

Es decir, decidir qué grupos salen en la consulta y cuáles no salen.

La sintaxis es como la del where, pero afectando a campos agrupadores o a funciones agrupadoras.

Es importante el orden de cada cláusula.

**1. select**

**2. from**

**3. where**

**4. group by**

**5. having**

**6. order by**

