SELECT (Seleccionar registros)

Para ver los registros de una tabla usamos "**SELECT**":

SELECT nombre,clave **FROM** usuarios;

El comando "**SELECT**" recupera los registros de una tabla. Luego del comando SELECTindicamos los nombres de los campos a rescatar.

Seleccionar todos los campos

SELECT * **FROM** libros;

El comando "SELECT" recupera los registros de una tabla. Con el asterisco (*****) indicamosque seleccione todos los campos de la tabla que nombramos. Podemos especificar el nombre de los campos que queremos ver separándolos por comas:

SELECT titulo,autor,editorial **FROM** libros;

En la sentencia anterior la consulta mostrará sólo los campos "titulo", "autor" y "editorial". Enla siguiente sentencia, veremos los campos correspondientes al título y precio de todos los libros:

SELECT titulo,precio **FROM** libros;

Ejemplo:

```
DROP TABLE IF EXISTS libros;

CREATE TABLE libros(titulo varchar(100), autor varchar(30), editorial
varchar(15), precio float,
cantidad integer
);

INSERT INTO libros (titulo,autor,editorial,precio,cantidad) VALUES('El
aleph','Borges','Emece',45.50,100),
('Alicia en el pais de las maravillas','Lewis Carroll','Planeta',25,200),
('Matematica estas ahi','Paenza','Planeta',15.8,200);

SELECT titulo,precio FROM libros; SELECT editorial,cantidad FROM libros;
SELECT * FROM libros;
```

Seleccionar con WHERE

Existe una cláusula, "WHERE" que es opcional, con ella podemos especificar

condicionespara la consulta "SELECT". Es decir, podemos recuperar algunos registros, sólo los que cumplan con ciertas condiciones indicadas con la cláusula "WHERE". Por ejemplo, queremos ver el usuario cuyo nombre es "MarioPerez", para ello utilizamos "WHERE" y luego de ella, la condición:

SELECT nombre, clave FROM usuarios WHERE nombre='MarioPerez';

Para las condiciones se utilizan operadores relacionales. El signo igual(=) es un operador relacional. Para la siguiente selección de registros especificamos una condición que solicitalos usuarios cuya clave es igual a 'bocajuniors':

SELECT nombre, clave FROM usuarios WHERE clave='bocajuniors';

Si ningún registro cumple la condición establecida con el "WHERE", no aparecerá ningúnregistro.

Fuente: https://www.tutorialesprogramacionya.com

INSERT INTO (Insertar registro)

Un registro es una fila de la tabla que contiene los datos propiamente dichos. Cada registrotiene un dato por cada columna.

CREATE TABLE usuarios (nombre varchar(30),clave varchar(10)
);

Al ingresar los datos de cada registro debe tenerse en cuenta la cantidad y el orden de loscampos.

Ahora vamos a agregar un registro a la tabla:

INSERT INTO usuarios (nombre, clave) VALUES ('MarioPerez','Marito');

Usamos "**INSERT INTO**". Especificamos los nombres de los campos entre paréntesis y separados por comas y luego los valores para cada campo, también entre paréntesis yseparados por comas.

La tabla usuarios ahora la podemos graficar de la siguiente forma:

nombre	clave
Mario Pérez	Marito

Es importante ingresar los valores en el mismo orden en que se nombran los campos, si ingresamos los datos en otro orden, no aparece un mensaje de error y los datos se guardande modo incorrecto.

Note que los datos ingresados, como corresponden a campos de cadenas de caracteres, secolocan entre comillas. Las comillas son **OBLIGATORIAS**.

Fuente: https://www.tutorialesprogramacionya.com

Operadores relacionales

Los operadores relacionales vinculan un campo con un valor para que MySQL comparecada registro (el campo especificado) con el valor dado.
Los operadores relacionales son los siguientes:

=	igual
<>	distinto
>	mayor
<	menor
>=	mayor o igual
<=	menor o igual

Podemos seleccionar los registros cuyo autor sea diferente de 'Borges', para ello usamos lacondición:

SELECT titulo,autor,editorial FROM libros WHERE autor<>'Borges';

Podemos comparar valores numéricos. Por ejemplo, queremos mostrar los libros cuyosprecios sean mayores a 20 pesos:

SELECT titulo,autor,editorial,precio FROM libros WHERE precio>20;

También, los libros cuyo precio sea menor o igual a 30:

SELECT titulo,autor,editorial,precio FROM libros WHERE precio<=30;
Operadores lógicos
Hasta el momento, hemos aprendido a establecer una condición con "WHERE" utilizandooperadores relacionales. Podemos establecer más de una condición con la cláusula "WHERE", para ello aprenderemos los operadores lógicos.
Son los siguientes:

- AND, significa "y",
- OR, significa "o",
- NOT, significa "no", invierte el resultado

Los operadores lógicos se usan para combinar condiciones.
Queremos recuperar todos los registros cuyo autor sea igual a "Borges" y cuyo precio nosupere los 20 pesos, para ello necesitamos 2 condiciones:

SELECT * FROM libros WHERE (autor='Borges') AND(precio<=20);

Los registros recuperados en una sentencia que une 2 condiciones con el operador "AND",cumplen con las 2 condiciones.
Queremos ver los libros cuyo autor sea "Borges" y/o cuya editorial sea "Planeta":

SELECT * FROM libros WHERE autor='Borges' OR editorial='Planeta';

El operador "NOT" invierte el resultado de la condición a la cual antecede.
Los registros recuperados en una sentencia en la cual aparece el operador "NOT", nocumplen con la condición a la cual afecta el "NO".
Los paréntesis se usan para encerrar condiciones, para que se evalúen como una solaexpresión.

Cuando explicitamos varias condiciones con diferentes operadores lógicos (combinamos "AND", "OR") permite establecer el orden de prioridad de la evaluación; además permitediferenciar las expresiones más claramente.
Por ejemplo, las siguientes expresiones devuelven un resultado diferente:

SELECT * FROM libros WHERE (autor='Borges') OR (editorial='Paidos' ANDprecio<20);

SELECT * FROM libros WHERE (autor='Borges' OR editorial='Paidos') ANDprecio<20);

Si bien los paréntesis no son obligatorios en todos los casos, se recomienda utilizarlos paraevitar confusiones.

Between / In

Existen otros que simplifican algunas consultas:
Para recuperar de nuestra tabla "libros" los registros que tienen precio mayor o igual a 20 ymenor o igual a 40, usamos 2 condiciones unidas por el operador lógico "and":

SELECT * FROM libros WHERE precio>=20 AND precio<=40;

Podemos usar "between":

SELECT * FROM libros WHERE precio BETWEEN 20 AND 40;

"between" significa "entre". Averiguamos si el valor de un campo dado (precio) está entrelos valores mínimo y máximo especificados (20 y 40 respectivamente).
Si agregamos el operador "not" antes de "between" el resultado se invierte.

Para recuperar los libros cuyo autor sea 'Paenza' o 'Borges' usamos 2 condiciones:

SELECT * FROM libros WHERE autor='Borges' OR autor='Paenza';

Podemos usar "in":

SELECT * FROM libros WHERE autor IN('Borges','Paenza');

Con "in" averiguamos si el valor de un campo dado (autor) está incluido en la lista devalores especificada (en este caso, 2 cadenas).
Para recuperar los libros cuyo autor no sea 'Paenza' ni 'Borges' usamos:

SELECT * FROM libros WHERE autor<>'Borges' AND autor<>'Paenza';

También podemos usar "in" :

SELECT * FROM libros WHERE autor NOT IN('Borges','Paenza');

Con "IN" averiguamos si el valor del campo está incluido en la lista, con "NOT" antecediendo la condición, invertimos el resultado.

Selecciones ordenadas

Podemos ordenar el resultado de un "SELECT" para que los registros se muestrenordenados por algún campo, para ello usamos la cláusula "**ORDER BY**".
Por ejemplo, recuperamos los registros de la tabla "libros" ordenados por el título:

SELECT codigo,titulo,autor,editorial,precio FROM libros ORDER BY titulo;

Aparecen los registros ordenados alfabéticamente por el campo especificado.

Por defecto, si no aclaramos en la sentencia, los ordena de manera ascendente (de menora mayor). Podemos ordenarlos de mayor a menor, para ello agregamos la palabra clave "**DESC**":

SELECT codigo,titulo,autor,editorial,precio FROM libros ORDER BY editorial desc;

Fuente: https://www.tutorialesprogramacionya.com

UPDATE (Borrar registro)

Si queremos eliminar uno o varios registros debemos indicar cuál o cuáles, para ello utilizamos el comando "DELETE" junto con la cláusula "WHERE" con la cual establecemos la condición que deben cumplir los registros a borrar. Por ejemplo, queremos eliminar aquelregistro cuyo nombre de usuario es 'Leonardo':

DELETE FROM usuarios WHERE nombre='Leonardo';

Si solicitamos el borrado de un registro que no existe, es decir, ningún registro cumple con la condición especificada, no se borrarán registros, pues no encontró registros con ese dato.Hay que tener mucho cuidado en su uso, una vez eliminado un registro no hay forma de recuperarlo. Si por ejemplo ejecutamos el comando:

DELETE FROM usuarios;

Si la tabla tiene 1000000 de filas, **todas ellas serán eliminadas**.

Fuente: https://www.tutorialesprogramacionya.com

Índices

Para facilitar la obtención de información de una tabla se utilizan índices.

El índice de una tabla desempeña la misma función que el índice de un libro: permite encontrar datos **rápidamente**; en el caso de las tablas, localiza registros.
Una tabla se indexa por un campo (o varios).

El índice es un tipo de archivo con 2 entradas: un dato (un valor de algún campo de la tabla)y un puntero.

Un índice posibilita el acceso directo y rápido haciendo más eficiente las búsquedas. Siníndice, se debe recorrer secuencialmente toda la tabla para encontrar un registro.
El objetivo de un índice es acelerar la recuperación de información.La desventaja es que consume espacio en el disco.

La indexación es una técnica que optimiza el acceso a los datos, mejora el rendimiento acelerando las consultas y otras operaciones. Es útil cuando la tabla contiene miles deregistros.

Los índices se usan para varias operaciones:
- para buscar registros rápidamente.
- para recuperar registros de otras tablas empleando "JOIN".

Es importante identificar el o los campos por los que sería útil crear un índice, aquellos campos por los cuales se realizan operaciones de búsqueda con frecuencia.

Hay distintos tipos de índices, a saber:
1) "primary key": es el que definimos como **clave primaria**. Los valores indexados deben ser **únicos** y además **no pueden ser nulos**. MySQL le da el nombre "PRIMARY". Una tablasolamente puede tener una clave primaria.
2) "index": crea un índice común, los valores no necesariamente son únicos y aceptan valores "null". Podemos darle un nombre, si no se lo damos, se coloca uno por defecto."key" es sinónimo de "index". Puede haber varios por tabla.
3) "unique": crea un índice para los cuales los valores deben ser **únicos y diferentes**, aparece un mensaje de error si intentamos agregar un registro con un valor ya existente.

Una tabla puede tener hasta 64 índices. Los nombres de índices aceptan todos los caracteres y pueden tener una longitud máxima de 64 caracteres. Pueden

comenzar con undígito, pero no pueden tener sólo dígitos.

Una tabla puede ser indexada por campos de tipo numérico o de tipo caracter. También sepuede indexar por un campo que contenga valores NULL, excepto los PRIMARY.

"SHOW index" muestra información sobre los índices de una tabla. Por ejemplo:

SHOW index FROM libros;

El índice llamado **PRIMARY** se crea automáticamente cuando establecemos un campo como **clave primaria**, no podemos crearlo directamente. El campo por el cual se indexapuede ser de tipo numérico o de tipo caracter.

Los valores indexados deben ser **únicos** y además **no pueden ser nulos**. Una tablasolamente puede tener una clave primaria por lo tanto, solamente tiene un índice **PRIMARY**.

Una clave primaria es un campo (o varios) que identifica 1 solo registro (fila) en una tabla.

Para un valor del campo clave existe solamente 1 registro. Los valores no se repiten nipueden ser nulos.

Veamos un ejemplo, si tenemos una tabla con datos de personas, el número de documentopuede establecerse como clave primaria, es un valor que no se repite; puede haber personas con igual apellido y nombre, incluso el mismo domicilio (padre e hijo por ejemplo),pero su documento será siempre distinto.

Si tenemos la tabla "usuarios", el nombre de cada usuario puede establecerse como claveprimaria, es un valor que no se repite; puede haber usuarios con igual clave, pero su nombre de usuario será siempre distinto.
Veamos un ejemplo definiendo la tabla "libros" con una clave primaria:CREATE TABLE libros(
codigo int unsigned auto_increment,
titulo varchar(40) not null,autor varchar(30), editorial varchar(15), primary key(codigo)
);

Podemos ver la estructura de los índices de una tabla con "SHOW index". Por ejemplo:

SHOW index FROM libros;

Aparece el índice PRIMARY creado automáticamente al definir el campo "codigo" comoclave primaria.

Ejemplo:

```
DROP TABLE if exists libros;CREATE TABLE libros(
codigo int unsigned auto_increment,titulo varchar(40) not null,
autor varchar(30), editorial varchar(15),primary key(codigo)
);

show index FROM libros;
```

Establecemos que un campo sea clave primaria al momento de creación de la tabla:

```
CREATE TABLE usuarios (
nombre varchar(20),clave varchar(10),
primary key(nombre)
);
```

Para definir un campo como clave primaria agregamos "primary key" luego de la definiciónde todos los campos y entre paréntesis colocamos el nombre del campo que queremos como clave.

Si visualizamos la estructura de la tabla con "DESCRIBE" vemos que el campo "nombre" esclave primaria y no acepta valores nulos(más adelante explicaremos esto detalladamente).

Ingresamos algunos registros:

```
INSERT INTO usuarios (nombre, clave)VALUES ('Leonardo','hola');
INSERT INTO usuarios (nombre, clave)VALUES ('MarioPerez','Marito');
INSERT INTO usuarios (nombre, clave)VALUES ('Marcelo','River');
INSERT INTO usuarios (nombre, clave)VALUES ('Gustavo','River');
```

Si intentamos ingresar un valor para el campo clave que ya existe, aparece un mensaje de error indicando que el registro no se cargó pues el dato clave existe. Esto sucede porque loscampos definidos como clave primaria no pueden repetirse.

Ingresamos un registro con un nombre de usuario repetido, por ejemplo:

```
INSERT INTO usuarios (nombre, clave) VALUES ('Gustavo','Boca');
```

Una tabla sólo puede tener una clave primaria. Cualquier campo (de cualquier tipo) puedeser clave primaria, debe cumplir como requisito, que sus valores no se repitan.

Al establecer una clave primaria estamos indexando la tabla, es decir, creando un índicepara dicha tabla; a este tema lo veremos más adelante.

```
DROP TABLE IF EXISTS usuarios;

CREATE TABLE usuarios (nombre  varchar(20), clave varchar(10),
primary key (nombre)
);

DESCRIBE usuarios;

INSERT INTO usuarios (nombre, clave) VALUES ('Leonardo', 'hola'); INSERT
INTO usuarios (nombre, clave) VALUES ('MarioPerez','Marito');INSERT INTO
usuarios (nombre, clave) VALUES ('Marcelo','River'); INSERT INTO usuarios
(nombre, clave) VALUES ('Gustavo','River');

INSERT INTO usuarios (nombre, clave) VALUES ('Gustavo','Boca');
```

Index

Dijimos que hay 3 tipos de índices. Hasta ahora solamente conocemos la clave primaria quedefinimos al momento de crear una tabla.

Vamos a ver el otro tipo de índice, común. Un índice común se crea con "index", los valoresno necesariamente son únicos y aceptan valores "null". Puede haber varios por tabla.
Vamos a trabajar con nuestra tabla "libros".

Un campo por el cual realizamos consultas frecuentemente es "editorial", indexar la tablapor ese campo sería útil.
Creamos un índice al momento de crear la tabla:

```
CREATE TABLE libros(
codigo int unsigned auto_increment,titulo varchar(40) not null,
autor varchar(30), editorial varchar(15), primary key(codigo), index i_editorial
(editorial)
);
```

Luego de la definición de los campos colocamos "index" seguido del nombre que le damos yentre paréntesis el o los campos por los cuales se indexará dicho índice.
"SHOW index" muestra la estructura de los índices:

SHOW index FROM libros;

Si no le asignamos un nombre a un índice, por defecto tomará el nombre del

primer campoque forma parte del índice, con un sufijo opcional (_2,_3,...) para que sea único.

Unique Index

Veamos el otro tipo de índice, único. Un índice único se crea con "**unique**", los valores deben ser únicos y diferentes, aparece un mensaje de error si intentamos agregar un registro con un valor ya existente. Permite valores nulos y pueden definirse varios por tabla.Podemos darle un nombre, si no se lo damos, se coloca uno por defecto.
Vamos a trabajar con nuestra tabla "libros".

```
CREATE TABLE libros(
codigo int unsigned auto_increment,titulo varchar(40) not null,
autor varchar(30), editorial varchar(15), unique i_codigo(codigo)
);
```

Luego de la definición de los campos colocamos "unique" seguido del nombre que le damosy entre paréntesis el o los campos por los cuales se indexará dicho índice.

Eliminar índice

Para eliminar un índice usamos "DROP index". Ejemplo:

```
DROP index i_editorial ON libros;
```

Se elimina el índice con "DROP index" seguido de su nombre y "ON" seguido del nombre dela tabla a la cual pertenece.

Podemos eliminar los índices creados con "index" y con "unique" pero no el que se crea aldefinir una clave primaria. Un índice PRIMARY se elimina automáticamente al eliminar la clave primaria.

Agregar índice a tabla existente

Podemos agregar un índice a una tabla existente.
Para agregar un índice común a la tabla "libros" tipeamos:

```
CREATE index i_editorial ON libros (editorial);
```

Entonces, para agregar un índice común a una tabla existente usamos "CREATE index", indicamos el nombre, sobre qué tabla y el o los campos por

los cuales se indexará, entreparéntesis.
Para agregar un índice único a la tabla "libros" tipeamos:

CREATE unique index i_tituloeditorial ON libros (editorial);

Para agregar un índice único a una tabla existente usamos "CREATE unique index", indicamos el nombre, sobre qué tabla y entre paréntesis, el o los campos por los cuales seindexará.
Un índice PRIMARY no puede agregarse, se crea automáticamente al definir una claveprimaria.

Campos autoincrementables

Un campo de tipo entero puede tener otro atributo extra 'auto_increment'. Los valores de uncampo 'auto_increment', se inician en 1 y se incrementan en 1 automáticamente.

Se utiliza generalmente en campos correspondientes a códigos de identificación paragenerar valores únicos para cada nuevo registro que se inserta.

Sólo puede haber un campo "auto_increment" y debe ser clave primaria (o estar indexado). Para establecer que un campo autoincremente sus valores automáticamente, éste debe serentero (integer) y debe ser clave primaria:

CREATE TABLE libros(codigo int auto_increment,titulo varchar(50),
autor varchar(50), editorial varchar(25),primary key (codigo)
);

Para definir un campo autoincrementable colocamos "auto_increment" luego de la definicióndel campo al crear la tabla.
Hasta ahora, al ingresar registros, colocamos el nombre de todos los campos antes de los valores; es posible ingresar valores para algunos de los campos de la tabla, pero recuerdeque al ingresar los valores debemos tener en cuenta los campos que detallamos y el ordenen que lo hacemos.
Cuando un campo tiene el atributo "auto_increment" no es necesario ingresar valor para él,porque se inserta automáticamente tomando el último valor como referencia, o 1 si es el primero.

Para ingresar registros omitimos el campo definido como "auto_increment", por ejemplo:

INSERT INTO libros (titulo,autor,editorial) VALUES('El aleph','Borges','Planeta');

Este primer registro ingresado guardará el valor 1 en el campo correspondiente al código.Si continuamos ingresando registros, el código (dato que no ingresamos) se cargará automáticamente siguiendo la secuencia de autoincremento.
Un campo "auto_increment" funciona correctamente sólo cuando contiene únicamentevalores positivos.

Está permitido ingresar el valor correspondiente al campo "auto_increment", por ejemplo:INSERT INTO libros (codigo,titulo,autor,editorial)
VALUES(6,'Martin Fierro','Jose Hernandez','Paidós');

Pero debemos tener cuidado con la inserción de un dato en campos "auto_increment".Debemos tener en cuenta que:
- si el valor está repetido aparecerá un mensaje de error y el registro no se ingresará.
- si el valor dado saltea la secuencia, lo toma igualmente y en las siguientes inserciones, continuará la secuencia tomando el valor más alto.
- si el valor ingresado es 0, no lo toma y guarda el registro continuando la secuencia.

Fuente: https://www.tutorialesprogramacionya.com

UPDATE (Actualizar registro)

Para modificar uno o varios datos de uno o varios registros utilizamos "UPDATE"(actualizar).
Por ejemplo, en nuestra tabla "usuarios", queremos cambiar los valores de todas las claves,por "RealMadrid":

UPDATE usuarios SET clave='RealMadrid';

Utilizamos "UPDATE" junto al nombre de la tabla y "SET" junto con el campo a modificar ysu nuevo valor.
El cambio afectará a todos los registros.
Podemos modificar algunos registros, para ello debemos establecer condiciones deselección con "WHERE".

Por ejemplo, queremos cambiar el valor correspondiente a la clave de nuestro usuariollamado 'MarioPerez', queremos como nueva clave 'Boca', necesitamos una condición"WHERE" que afecte solamente a este registro:

UPDATE usuarios set clave='Boca' WHERE nombre='MarioPerez';

Si no encuentra registros que cumplan con la condición del "WHERE", ningún registro esafectado.
Las condiciones no son obligatorias, pero si omitimos la cláusula "WHERE", la actualizaciónafectará a todos los registros.
También se puede actualizar varios campos en una sola instrucción:

UPDATE usuarios set nombre='MarceloDuarte', clave='Marce' WHERE nombre='Marcelo';

Para ello colocamos "UPDATE", el nombre de la tabla, "SET" junto al nombre del campo yel nuevo valor y separado por coma, el otro nombre del campo con su nuevo valor.

Fuente: https://www.tutorialesprogramacionya.com

LIKE

Hemos realizado consultas utilizando operadores relacionales para comparar cadenas. Por ejemplo, sabemos recuperar los libros cuyo autor sea igual a la cadena "Borges":

SELECT * FROM libros WHERE autor='Borges';

Los operadores relacionales nos permiten comparar valores numéricos y cadenas de caracteres. Pero al realizar la comparación de cadenas, busca coincidencias de cadenascompletas.
Imaginemos que tenemos registrados estos 2 libros:
-El Aleph de Borges;
-Antología poética de J.L. Borges;

Si queremos recuperar todos los libros cuyo autor sea "Borges", y especificamos lasiguiente condición:

SELECT * FROM libros WHERE autor='Borges';

sólo aparecerá el primer registro, ya que la cadena "Borges" no es igual a la cadena "J.L.Borges".
Esto sucede porque el operador "=" (igual), también el operador "<>" (distinto) comparancadenas de caracteres completas. Para comparar porciones de cadenas utilizamos los operadores "LIKE" y "NOT LIKE".

Entonces, podemos comparar trozos de cadenas de caracteres para realizar consultas. Para recuperar todos los registros cuyo autor contenga la cadena "Borges" debemos tipear:

SELECT * FROM libros WHERE autor LIKE "%Borges%";

El símbolo "%" (porcentaje) reemplaza cualquier cantidad de caracteres (incluyendo ningúncaracter). Es un caracter comodín. "LIKE" y "NOT LIKE" son operadores de comparación que señalan igualdad o diferencia.

Para seleccionar todos los libros que comiencen con "A":

SELECT * FROM libros WHERE titulo LIKE 'A%';

Note que el símbolo "%" ya no está al comienzo, con esto indicamos que el título debe tenercomo primera letra la "A" y luego, cualquier cantidad de caracteres.

Para seleccionar todos los libros que no comiencen con "A":

SELECT * FROM libros WHERE titulo NOT LIKE 'A%';

Así como "%" reemplaza cualquier cantidad de caracteres, el guión bajo "_" reemplaza uncaracter, es el otro caracter comodín. Por ejemplo, queremos ver los libros de "Lewis Carroll" pero no recordamos si se escribe "Carroll" o "Carrolt", entonces tipeamos esta condición:

SELECT * FROM libros WHERE autor LIKE "%Carrol_";

Si necesitamos buscar un patrón en el que aparezcan los caracteres comodines, por ejemplo, queremos ver todos los registros que comiencen con un guión bajo, si utilizamos

'_%', mostrará todos los registros porque lo interpreta como "patrón que comienza con un caracter cualquiera y sigue con cualquier cantidad de caracteres". Debemos utilizar "_%",esto se interpreta como 'patrón que comienza con guión bajo y continúa con cualquier cantidad de caracteres".

Es decir, si queremos incluir en una búsqueda de patrones los caracteres comodines, debemos anteponer al caracter comodín, la barra invertida "\", así lo tomará como caracter de búsqueda literal y no como comodín para la búsqueda. Para buscar el caracter literal "%"se debe colocar "\%".

Fuente: https://www.tutorialesprogramacionya.com

Funciones matemáticas

Existen en MySQL funciones que nos permiten contar registros, calcular sumas, promedios,obtener valores máximos y mínimos. Ya hemos aprendido "count()", veamos otras.
La función "sum()" retorna la suma de los valores que contiene el campo especificado. Porejemplo, queremos saber la cantidad de libros que tenemos disponibles para la venta:

SELECT sum(cantidad) FROM libros;

También podemos combinarla con "WHERE". Por ejemplo, queremos saber cuántos librostenemos de la editorial "Planeta":

SELECT sum(cantidad) FROM libros WHERE editorial ='Planeta';

Para averiguar el valor máximo o mínimo de un campo usamos las funciones "max()" y "min()" respectivamente. Ejemplo, queremos saber cuál es el mayor precio de todos loslibros:

SELECT max(precio) FROM libros;

Queremos saber cuál es el valor mínimo de los libros de "Rowling":

SELECT min(precio) FROM libros WHERE autor LIKE '%Rowling%';

La función avg() retorna el valor promedio de los valores del campo especificado. Porejemplo, queremos saber el promedio del precio de los libros referentes a "PHP":

SELECT avg(precio) FROM libros WHERE titulo LIKE '%PHP%';

Estas funciones se denominan "funciones de agrupamiento" porque operan sobre conjuntosde registros, no con datos individuales.
Tenga en cuenta que no debe haber espacio entre el nombre de la función y el paréntesis,porque puede confundirse con una referencia a una tabla o campo.
Las siguientes sentencias son distintas:
SELECT count(*) FROM libros;SELECT count (*) FROM libros;

La primera es correcta, la segunda incorrecta.

Fuente: https://www.tutorialesprogramacionya.com

Primary key y Foreign Key

Las **claves primarias (Primary Keys)** son valores que identifican de manera única a cada fila o registro de una tabla, esto quiere decir que no se puede repetir. Por ejemplo: un DNI, un código de producto, etc.
Una **clave foránea (Foreign Key)** es un campo de una tabla “X” que sirve para enlazar o relacionar entre sí con otra tabla “Y” en la cual el campo de esta tabla es una llave primaria (Primary Key). Para que sea una clave foránea un campo, esta tiene que ser una llave primaria en otra tabla.
Por ejemplo, en la tabla clientes el dni es una primary key, pero en una tabla “pedidos” representa a quién pertenece ese determinado pedido.
En los siguientes apartados veremos más detalles sobre estos dos tipos de claves.

Clave primaria

Una clave primaria es un campo (o varios) que identifica 1 solo registro (fila) en una tabla.Para un valor del campo clave existe solamente 1 registro. Los valores no se repiten ni pueden ser nulos.
Veamos un ejemplo, si tenemos una tabla con datos de personas, el número de documentopuede establecerse como clave primaria, es un valor que no se repite; puede haber personas con igual apellido y nombre, incluso el mismo domicilio (padre e hijo por ejemplo),pero su documento será siempre distinto.
Si tenemos la tabla "usuarios", el nombre de cada usuario puede establecerse como claveprimaria, es un valor que no se repite; puede haber usuarios con igual clave, pero su nombre de usuario será siempre distinto.
Establecemos que un campo sea clave primaria al momento de creación de la tabla:

```
CREATE TABLE usuarios (nombre  varchar(20), clave varchar(10),
primary key(nombre)
);
```

Para definir un campo como clave primaria agregamos "primary key" luego de la definiciónde todos los campos y entre paréntesis colocamos el nombre del campo que queremos como clave.
Si visualizamos la estructura de la tabla con "describe" vemos que el campo "nombre" es clave primaria y no acepta valores nulos(más adelante explicaremos esto detalladamente).Ingresamos algunos registros:

```
INSERT INTO usuarios (nombre, clave)  VALUES ('Leonardo',Leo'),
('MarioPerez','Marito'),
('Marcelo','River'),
('Gustavo','River');
```

Si intentamos ingresar un valor para el campo clave que ya existe, aparece un mensaje de error indicando que el registro no se cargó pues el dato clave existe. Esto sucede porque loscampos definidos como clave primaria no pueden repetirse.
Ingresamos un registro con un nombre de usuario repetido, por ejemplo:

```
INSERT INTO usuarios (nombre, clave) VALUES ('Gustavo','Boca');
```

Una tabla sólo puede tener una clave primaria. Cualquier campo (de cualquier tipo) puedeser clave primaria, debe cumplir como requisito, que sus valores no se repitan.
Al establecer una clave primaria estamos indexando la tabla, es decir, creando

un índicepara dicha tabla; a este tema lo veremos más adelante.

Ejemplo:

```
DROP TABLE IF EXISTS usuarios;

CREATE TABLE usuarios (nombre  varchar(20), clave varchar(10), PRIMARY
KEY(nombre)
);

DESCRIBE usuarios;

INSERT INTO usuarios (nombre, clave)  VALUES  ('Leonardo','Leo'),
('MarioPerez','Marito'),
('Marcelo','River'),
('Gustavo','River'),
('Gustavo','Boca');
```

Fuente: https://www.tutorialesprogramacionya.com

Clave foránea

Un campo que se usa para establecer un "JOIN" con otra tabla en la cual es clave primaria,se denomina "clave ajena o foránea".
En el ejemplo de la librería en que utilizamos las tablas "libros" y "editoriales" con loscampos:
libros: codigo (clave primaria), titulo, autor, codigo_editorial, precio, cantidad y **editoriales**: codigo (clave primaria), nombre.

el campo "codigo_editorial" de "libros" es una clave foránea, se emplea para enlazar la tabla"libros" con "editoriales" y es clave primaria en "editoriales" con el nombre "codigo".
Cuando alteramos una tabla, debemos tener cuidado con las claves foráneas. Si modificamos el tipo, longitud o atributos de una clave foránea, ésta puede quedarinhabilitada para hacer los enlaces.
Las claves foráneas y las claves primarias deben ser del mismo tipo para poder enlazarse.Si modificamos una, debemos modificar la otra para que los valores se correspondan.

Fuente: https://www.tutorialesprogramacionya.com

JOIN

Los JOINs en SQL sirven para combinar filas de dos o más tablas basándose en un campocomún entre ellas, devolviendo por tanto datos de diferentes tablas. Un JOIN se produce cuando dos o más tablas se juntan en una sentencia SQL.
Los más importantes son los siguientes:

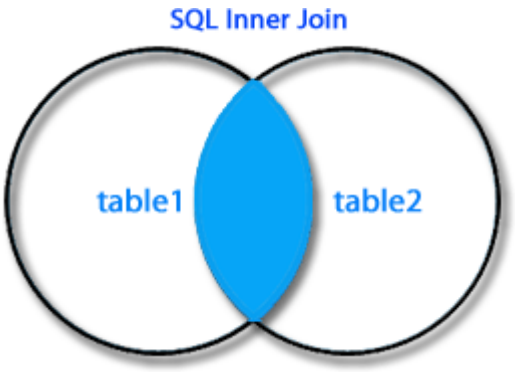
1. **INNER JOIN**: Devuelve todas las filas cuando hay al menos una coincidencia enambas tablas.
2. **LEFT JOIN:** Devuelve todas las filas de la tabla de la izquierda, y las filascoincidentes de la tabla de la derecha.
3. **RIGHT JOIN**: Devuelve todas las filas de la tabla de la derecha, y las filascoincidentes de la tabla de la izquierda.
4. **OUTER JOIN**: Devuelve todas las filas de las dos tablas, la izquierda y la derecha.También se llama FULL OUTER JOIN.

1. INNER JOIN

INNER JOIN selecciona todas las filas de las dos columnas siempre y cuando haya unacoincidencia entre las columnas en ambas tablas. Es el tipo de JOIN más común.

```
SELECT nombreColumna(s)
FROM tabla1
INNER JOIN tabla2
ON tabla1.nombreColumna=table2.nombreColumna;
```

Se ve más claro utilizando una imagen:



Vamos a verlo también con un ejemplo, mediante las tablas clientes y pedidos:
clientes:

id_cliente	nombre	contacto

1	Marco Lambert	456443552
2	Lydia Roderic	445332221
3	Ebbe Therese	488982635
4	Sofie Mariona	412436773

pedidos:

id_pedido	id_cliente	factura
233	4	160
234	2	48
235	3	64
236	4	92

Creación de Base de Datos de prueba:

```
CREATE TABLE clientes(
id_cliente int unsigned AUTO_INCREMENT,nombre varchar(40) not null,
contacto int unsigned not null,PRIMARY KEY (id_cliente)
);

CREATE TABLE pedidos(
id_pedido int unsigned AUTO_INCREMENT,id_cliente int unsigned,
factura int unsigned not null,PRIMARY KEY(id_pedido)
);

ALTER TABLE pedidos AUTO_INCREMENT=233;

INSERT INTO clientes (nombre,contacto) VALUES ('Marco Lambert',456443552),
('Lydia Roderic',445332221), ('Ebbe Therese',488982635), ('Sofie Mariona',412436773);
```

INSERT INTO pedidos (id_cliente,factura) VALUES (4,160),
(2,48),
(3,64),
(4,92),
(10,550);

SET FOREIGN_KEY_CHECKS = 0;
ALTER TABLE pedidos ADD FOREIGN KEY(id_cliente) REFERENCES clientes(id_cliente);SET FOREIGN_KEY_CHECKS = 1;

La siguiente sentencia SQL devolverá todos los clientes con pedidos:
SELECT c.nombre,p.id_pedido
FROM clientes c
INNER JOIN pedidos p
ON c.id_cliente=p.id_cliente;

Si hay filas en clientes que no tienen coincidencias en pedidos, los clientes no se mostrarán.La sentencia anterior mostrará el siguiente resultado:

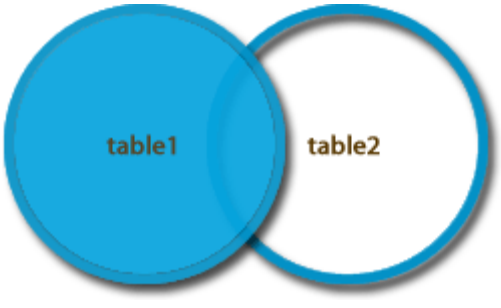
nombre	id_pedido
Ebbe Therese	235
Lydia Roderic	234
Sofie Mariona	233
Sofie Mariona	236

Sofie Mariona aparece dos veces ya que ha realizado dos pedidos. No aparece MarcoLambert, pues no ha realizado ningún pedido.

2. LEFT JOIN

LEFT JOIN mantiene todas las filas de la tabla izquierda (la tabla1). Las filas de la tabla derecha se mostrarán si hay una coincidencia con las de la izquierda. Si existen valores enla tabla izquierda pero no en la tabla derecha, ésta mostrará null.

La representación de **LEFT JOIN** en una imagen es:



Tomando de nuevo las tablas de clientes y pedidos, ahora queremos mostrar todos losclientes, y cualquier pedido que pudieran haber encargado:

```
SELECT c.id_cliente,c.nombre,p.id_pedido
FROM clientes c
LEFT JOIN pedidos p
ON c.id_cliente=p.id_cliente;
```

La sentencia anterior devolverá lo siguiente:

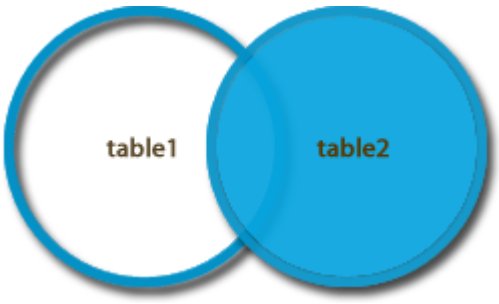
nombre	id_pedido
Ebbe Therese	235
Lydia Roderic	234
Marco Lambert	(null)
Sofie Mariona	233
Sofie Mariona	236

Ahora vemos que se muestran todas las filas de la tabla Clientes, que es la tabla de la izquierda, tantas veces como haya coincidencias con el lado derecho. Marco Lambert no harealizado ningún pedido, por lo que se muestra null.

3. RIGHT JOIN

Es igual que LEFT JOIN pero al revés. Ahora se mantienen todas las filas de la tabla derecha (tabla2). Las filas de la tabla izquierda se mostrarán si hay una coincidencia con lasde la derecha. Si existen valores en la tabla derecha pero no en la tabla izquierda, ésta se mostrará null.

La imagen que representa a RIGHT JOIN es:



De nuevo tomamos el ejemplo de Clientes y Pedidos, y vamos a hacer el mismo ejemploanterior, pero cambiando LEFT por RIGHT:

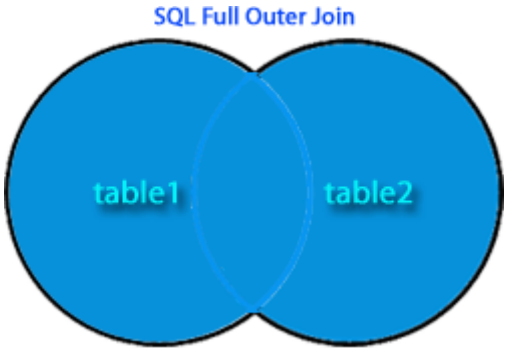
SELECT p.id_pedido, c.nombre
FROM clientes c **RIGHT JOIN** pedidos p
ON c.id_cliente=p.id_cliente;

Ahora van a aparecer todos los pedidos, y los nombres de los clientes que han realizado unpedido. Nótese que también se ha cambiado el orden, y se han ordenado los datos por id_pedido.

PedidoID	NombreCliente
233	Sofie Mariona
234	Lydia Roderic
235	Ebbe Therese
236	Sofie Mariona

4. OUTER JOIN

OUTER JOIN o FULL OUTER JOIN devuelve todas las filas de la tabla izquierda (tabla1) yde la tabla derecha (tabla2). Combina el resultado de los joins LEFT y RIGHT. Aparecerá null en cada una de las tablas alternativamente cuando no haya una coincidencia.
La imagen que representa el OUTER JOIN es la siguiente:



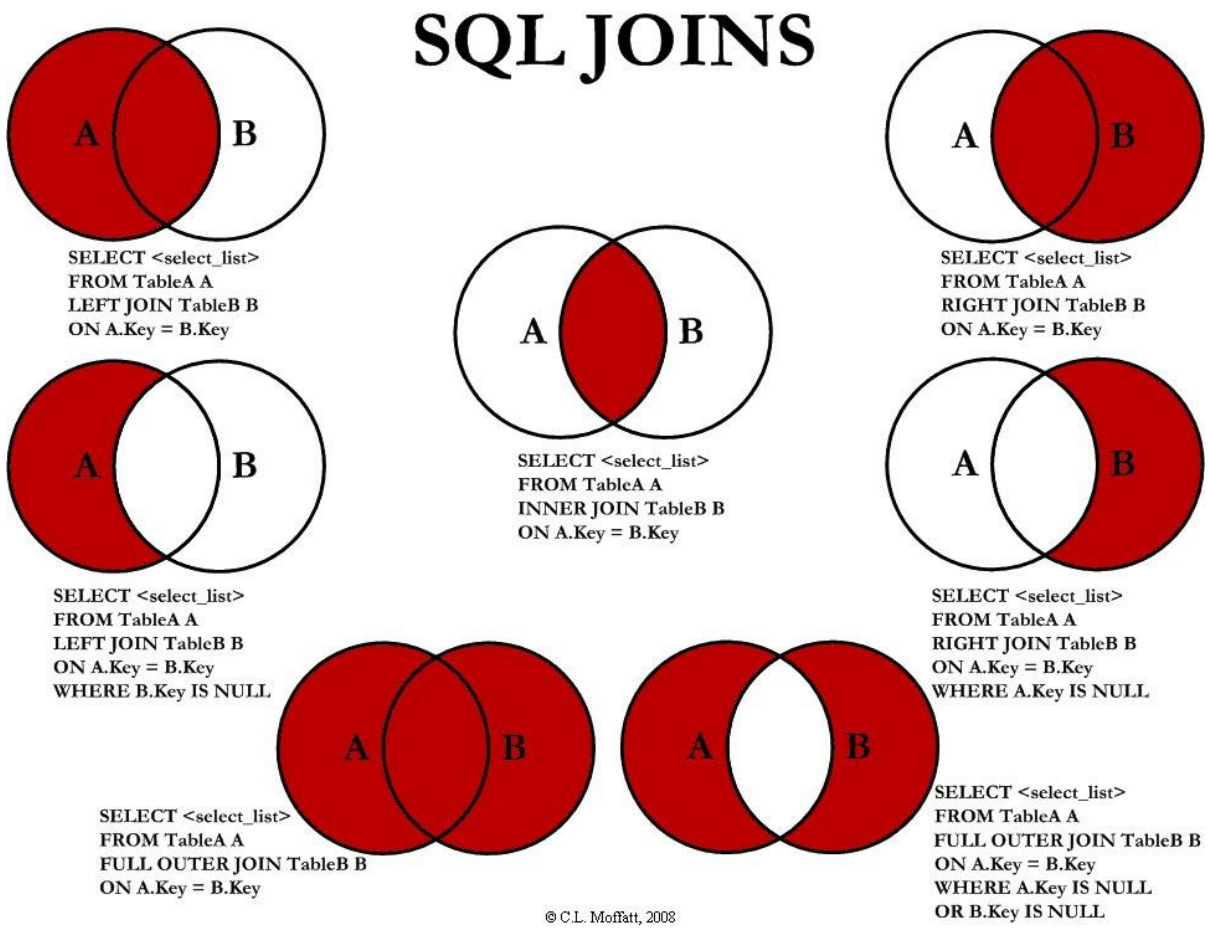
Vamos a obtener todas las filas de las tablas clientes y pedidos:
La sentencia devolverá todos los Clientes y todos los Pedidos, si un cliente no tiene pedidos mostrará null en id_pedido, y si un pedido no tuviera un cliente mostraría null en nombre (en este ejemplo no sería lógico que un Pedido no tuviera un cliente).

La sintaxis de OUTER JOIN o FULL OUTER JOIN no existen en MySQL, pero se puedeconseguir el mismo resultado de diferentes formas, esta es una:

SELECT c.id_cliente,p.id_pedido
FROM clientes c
LEFT JOIN pedidos p
ON c.id_cliente=p.id_cliente

UNION

SELECT c.id_cliente,p.id_pedido
FROM clientes c
RIGHT JOIN pedidos p
ON c.id_cliente=p.id_cliente;



Fuente: https://www.tutorialesprogramacionya.com

ALTER TABLE

Para modificar la estructura de una tabla existente, usamos "alter table"."alter table" se usa para:

- agregar nuevos campos,
- eliminar campos existentes,
- modificar el tipo de dato de un campo,
- agregar o quitar modificadores como "null", "unsigned", "auto_increment",
- cambiar el nombre de un campo,
- agregar o eliminar la clave primaria,
- agregar y eliminar índices,
- renombrar una tabla.

"alter table" hace una copia temporal de la tabla original, realiza los cambios en la copia,luego borra la tabla original y renombra la copia.
Aprenderemos a agregar campos a una tabla.
Para ello utilizamos nuestra tabla "libros", definida con la siguiente estructura:

- código, int unsigned auto_increment, clave primaria,
- titulo, varchar(40) not null,
- autor, varchar(30),
- editorial, varchar (20),
- precio, decimal(5,2) unsigned.

Necesitamos agregar el campo "cantidad", de tipo smallint unsigned not null, tipeamos:

ALTER TABLE libros ADD cantidad smallint unsigned not null;

Usamos "alter table" seguido del nombre de la tabla y "add" seguido del nombre del nuevocampo con su tipo y los modificadores.
Agreguemos otro campo a la tabla:

ALTER TABLE libros ADD edicion date;

Si intentamos agregar un campo con un nombre existente, aparece un mensaje de error indicando que el campo ya existe y la sentencia no se ejecuta.
Cuando se agrega un campo, si no especificamos, lo coloca al final, después de todos los campos existentes; podemos indicar su posición (luego de qué campo debe aparecer) con"after":

ALTER TABLE libros ADD cantidad tinyint unsigned AFTER autor;

Alter table para eliminar un campo de una tabla

"alter table" nos permite alterar la estructura de la tabla, podemos usarla para eliminar uncampo.
Continuamos con nuestra tabla "libros". Para eliminar el campo "edicion" tipeamos:

ALTER TABLE libros DROP edicion;

Entonces, para borrar un campo de una tabla usamos "alter table" junto con "drop" y elnombre del campo a eliminar.
Si intentamos borrar un campo inexistente aparece un mensaje de error y la acción no serealiza.
Podemos eliminar 2 campos en una misma sentencia:

ALTER TABLE libros DROP editorial, DROP cantidad;

Si se borra un campo de una tabla que es parte de un índice, también se borra el índice.Si una tabla tiene sólo un campo, este no puede ser borrado.
Hay que tener cuidado al eliminar un campo, este puede ser clave primaria. Es posibleeliminar un campo que es clave primaria, no aparece ningún mensaje:

ALTER TABLE libros DROP codigo;

Si eliminamos un campo clave, la clave también se elimina.

Alter table para modificar un campo de una tabla

Con "alter table" podemos modificar el tipo de algún campo incluidos sus atributos.Continuamos con nuestra tabla "libros", definida con la siguiente estructura:

- código, int unsigned,
- titulo, varchar(30) not null,
- autor, varchar(30),
- editorial, varchar (20),
- precio, decimal(5,2) unsigned,
- cantidad int unsigned.

Queremos modificar el tipo del campo "cantidad", como guardaremos valores que nosuperarán los 50000 usaremos smallint unsigned, tipeamos:

ALTER TABLE libros MODIFY cantidad smallint unsigned;

Usamos "alter table" seguido del nombre de la tabla y "modify" seguido del

nombre delnuevo campo con su tipo y los modificadores.
Queremos modificar el tipo del campo "titulo" para poder almacenar una longitud de 40caracteres y que no permita valores nulos, tipeamos:

ALTER TABLE libros MODIFY titulo varchar(40) not null;

Hay que tener cuidado al alterar los tipos de los campos de una tabla que ya tiene registroscargados. Si tenemos un campo de texto de longitud 50 y lo cambiamos a 30 de longitud, los registros cargados en ese campo que superen los 30 caracteres, se cortarán (en versiones nuevas de MySQL 8.x genera un error y no modifica la estructura de la tabla) Igualmente, si un campo fue definido permitiendo valores nulos, se cargaron registros con valores nulos y luego se lo define "not null", todos los registros con valor nulo para ese campo cambiarán al valor por defecto según el tipo (cadena vacía para tipo texto y 0 para numéricos), ya que "null" se convierte en un valor inválido.
Si definimos un campo de tipo decimal(5,2) y tenemos un registro con el valor "900.00" y luego modificamos el campo a "decimal(4,2)", el valor "900.00" se convierte en un valor inválido para el tipo, entonces guarda en su lugar, el valor límite más cercano, "99.99" (enversiones nuevas de MySQL genera un error y no modifica la estructura de la tabla).
Si intentamos definir "auto_increment" un campo que no es clave primaria, aparece unmensaje de error indicando que el campo debe ser clave primaria.
Por ejemplo:

ALTER TABLE libros MODIFY codigo int unsigned auto_increment;

Fuente: https://www.tutorialesprogramacionya.com

Subconsultas

El uso de subconsultas es una técnica que permite utilizar el resultado de una tabla **SELECT** en otra consulta **SELECT**. Permite solucionar consultas complejas mediante el uso de resultados previos conseguidos a través de otra consulta.
El **SELECT** que se coloca en el interior de otro **SELECT** se conoce con el término de **SUBCONSULTAS**. Esa **SUBCONSULTA** se puede colocar dentro de las cláusulas **WHERE**, **HAVING**, **FROM** o **JOIN**.

Subconsultas simples

Las subconsultas simples son aquellas que devuelven una única fila. Si además devuelven una única columna, se las llama subconsultas escalares, ya que devuelven un único valor. La sintaxis es:

```
SELECT
listaExpresiones
FROM tabla
WHERE expresión OPERADOR
                    (SELECT
                    listaExpresiones
                    FROM tabla);
```

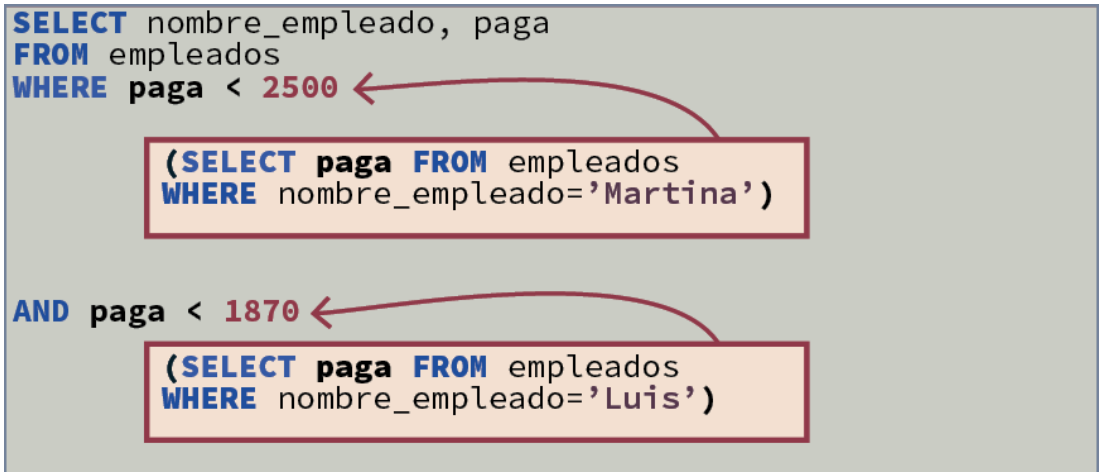
El operador puede ser >,<,>=,<=,!=, = o IN. Ejemplo:

```
SELECT nombre_empleado, paga FROM empleados
WHERE paga <
(SELECT paga FROM empleados WHERE nombre_empleado='Martina');
```

Esta consulta muestra el nombre y paga de los empleados cuya paga es menor que la de la empleada Martina. Para que funcione esta consulta, la subconsulta sólo puede devolver un valor (solo puede haber una empleada que se llame Martina.

Se pueden usar subconsultas las veces que haga falta:

```
SELECT nombre_empleado, paga FROM empleados
WHERE paga <
(SELECT paga FROM empleados WHERE nombre_empleado='Martina')
AND paga >
(SELECT paga FROM empleado WHERE nombre_empleado='Luis');
```



En realidad lo primero que hace la base de datos es calcular el resultado de la subconsulta:
La última consulta obtiene los empleados cuyas pagas estén entre lo que gana Luís (1870 euros) y lo que gana Martina (2500) .
Las subconsultas siempre se deben encerrar entre paréntesis y se deberían (aunque no es obligatorio, sí altamente recomendable) colocar a la derecha del operador relacional.
Una subconsulta que utilice los valores >,<,>=,... tiene que devolver un único valor, de otro modo ocurre un error.

Además tienen que devolver el mismo tipo y número de datos para relacionar la subconsulta con la consulta que la utiliza (no puede ocurrir que la subconsulta tenga dos columnas y ese resultado se compara usando una sola columna en la consulta general).

Fuente: https://www.tutorialesprogramacionya.com

Funciones para Strings

NO debe haber espacios entre un nombre de función y los paréntesis porque MySQL puedeconfundir una llamada a una función con una referencia a una tabla o campo que tenga el mismo nombre de una función.
MySQL tiene algunas funciones para trabajar con cadenas de caracteres. Estas sonalgunas:

-ord(caracter): Retorna el código ASCII para el caracter enviado como argumento.Ejemplo: select ord('A'); retorna 65.

-char(x,..): retorna una cadena con los caracteres en código ASCII de los enteros enviadoscomo argumentos. Ejemplo: select char(65,66,67); retorna "ABC".

-concat(cadena1,cadena2,...): devuelve la cadena resultado de concatenar los argumentos. Ejemplo: select concat('Hola,',' ','como esta?'); retorna "Hola, como esta?".

-concat_ws(separador,cadena1,cadena2,...): "ws" son las iniciales de "with separator". Elprimer argumento especifica el separador que utiliza para los demás argumentos; el separador se agrega entre las cadenas a concatenar. Ejemplo: select concat_ws('-','Juan','Pedro','Luis'); retorna "Juan-Pedro-Luis".

-length(cadena): retorna la longitud de la cadena enviada como argumento. Ejemplo:select length('Hola'); devuelve 4.

-locate(subcadena,cadena): retorna la posición de la primera ocurrencia de la subcadena en la cadena enviadas como argumentos. Devuelve "0" si la subcadena no se encuentra enla cadena. Ejemplo: select locate('o','como le va'); retorna 2.

- **lpad(cadena,longitud,cadenarelleno):** retorna la cadena enviada como primer argumento, rellenada por la izquierda con la cadena enviada como tercer argumento hastaque la cadena retornada tenga la longitud especificada como segundo argumento. Si la cadena es más larga, la corta. Ejemplo: select lpad('hola',10,'0'); retorna "000000hola".

- **rpad(cadena,longitud,cadenarelleno):** igual que "lpad" excepto que rellena por laderecha.

- **left(cadena,longitud)**: retorna la cantidad (longitud) de caracteres de la cadenacomenzando desde la izquierda, primer caracter. Ejemplo:

select left('buenos dias',8); retorna "buenos d".

- **right(cadena,longitud):** retorna la cantidad (longitud) de caracteres de la cadenacomenzando desde la derecha, último caracter. Ejemplo:
select right('buenos dias',8); retorna "nos dias".

-ltrim(cadena): retorna la cadena con los espacios de la izquierda eliminados. Ejemplo:

select ltrim(' Hola '); retorna "Hola "

- **rtrim(cadena):** retorna la cadena con los espacios de la derecha eliminados. Ejemplo:select rtrim(' Hola '); retorna " Hola"

-trim([[both|leading|trailing] [subcadena] from] cadena): retorna una cadena igual a la enviada pero eliminando la subcadena prefijo y/o sufijo. Si no se indica ningún especificador(both, leading o trailing) se asume "both" (ambos). Si no se especifica prefijos o sufijos elimina los espacios. Ejemplos:

select trim(' Hola '); retorna 'Hola'.
select trim (leading '0' from '00hola00'); retorna "hola00".select trim (trailing '0' from '00hola00'); retorna "00hola". select trim (both '0' from '00hola00'); retorna "hola". select trim ('0' from '00hola00'); retorna "hola".

-replace(cadena,cadenareemplazo,cadenareemplazar): retorna la cadena con todas lasocurrencias de la subcadena reemplazo por la subcadena a reemplazar.
Ejemplo: select replace('yyy.mysql.com','y','w'); retorna "www.mysql.com'.

-repeat(cadena,cantidad): devuelve una cadena consistente en la cadena repetida lacantidad de veces especificada. Si la "cantidad" es menor o igual a cero, retorna una cadena vacía. Ejemplo: select repeat('hola',3); retorna "holaholahola".

-reverse(cadena): devuelve la cadena invirtiendo el orden de los caracteres. Ejemplo:select reverse('Hola'); retorna "aloH".

-lcase(cadena) y lower(cadena): retornan la cadena con todos los caracteres enminúsculas. Ejemplos:

select lower('HOLA ESTUDIAnte'); retorna "hola estudiante".select lcase('HOLA ESTUDIAnte'); retorna "hola estudiante".

-ucase(cadena) y upper(cadena): retornan la cadena con todos los caracteres

enmayúsculas. Ejemplos:

select upper('HOLA ESTUDIAnte'); retorna "HOLA ESTUDIANTE".select ucase('HOLA ESTUDIAnte'); retorna "HOLA ESTUDIANTE".

-strcmp(cadena1,cadena2): retorna 0 si las cadenas son iguales, -1 si la primera es menorque la segunda y 1 si la primera es mayor que la segunda. Ejemplo:
select strcmp('Hola','Chau'); retorna 1.

Ejemplo práctico:

```
create table libros(
codigo int unsigned auto_increment,titulo varchar(40) not null,
autor varchar(30), editorial varchar (20),
precio decimal(5,2) unsigned,primary key(codigo)
);

insert into libros (titulo,autor,editorial,precio)values('El
Aleph','Borges','Paidos',33.4); insert into libros (titulo,autor,editorial,precio)
values('Alicia en el País de las Maravillas','L. Carroll','Planeta',16);

select concat_ws('-',titulo,autor)from libros;

select left(titulo,15)from libros;

select lower(titulo), upper(editorial)from libros;
```

Los operadores aritméticos son "+","-","*" y "/". Todas las operaciones matemáticas retornan"null" en caso de error. Ejemplo:
select 5/0;

Fuente: http://www.tutorialesprogramacionya.com

Funciones para valores numéricos

MySQL tiene algunas funciones para trabajar con números. Aquí presentamos algunas. RECUERDE que NO debe haber espacios entre un nombre de función y los paréntesis porque MySQL puede confundir una llamada a una función con una referencia a una tabla ocampo que tenga el mismo nombre de una función.

-abs(x): retorna el valor absoluto del argumento "x". Ejemplo: select abs(-20); retorna 20.

-ceiling(x): redondea hacia arriba el argumento "x". Ejemplo: select ceiling(12.34),retorna13.

-floor(x): redondea hacia abajo el argumento "x". Ejemplo: select floor(12.34); retorna 12.

-greatest(x,y,..): retorna el argumento de máximo valor.

-least(x,y,...): con dos o más argumentos, retorna el argumento más pequeño.

-mod(n,m): significa "módulo aritmético"; retorna el resto de "n" dividido en "m". Ejemplos:select mod(10,3); retorna 1.
select mod(10,2); retorna 0.

-power(x,y): retorna el valor de "x" elevado a la "y" potencia. Ejemplo: select power(2,3);retorna 8.

-rand(): retorna un valor de coma flotante aleatorio dentro del rango 0 a 1.0.

-round(x): retorna el argumento "x" redondeado al entero más cercano. Ejemplos:select round(12.34); retorna 12.
select round(12.64); retorna 13.

-sqrt(): devuelve la raiz cuadrada del valor enviado como argumento.

-truncate(x,d): retorna el número "x", truncado a "d" decimales. Si "d" es 0, el resultado notendrá parte fraccionaria. Ejemplos:
select truncate(123.4567,2); retorna 123.45;select truncate (123.4567,0); retorna 123.

Todas retornan null en caso de error.

Ejemplo práctico:

```
drop table if exists libros;create table libros(
codigo int unsigned auto_increment,titulo varchar(40) not null,
autor varchar(30), editorial varchar (20),
precio decimal(5,2) unsigned,primary key(codigo)
);

insert into libros (titulo,autor,editorial,precio)values('El
aleph','Borges','Paidos',33.46); insert into libros (titulo,autor,editorial,precio)
values('Alicia en el pais de las maravillas','L. Carroll','Planeta',16.31);insert into
libros (titulo,autor,editorial,precio)
values('Alicia a traves del espejo','L. Carroll','Planeta',18.89);

select titulo, ceiling(precio),floor(precio)from libros;

select titulo, round(precio) from libros;

select titulo,truncate(precio,1)from libros;
```

Fuente: http://www.tutorialesprogramacionya.com

Columnas calculadas

Es posible obtener salidas en las cuales una columna sea el resultado de un cálculo y no uncampo de una tabla.
Si queremos ver los títulos, precio y cantidad de cada libro escribimos la siguientesentencia:
select titulo,precio,cantidadfrom libros;

Si queremos saber el monto total en dinero de un título podemos multiplicar el precio por lacantidad por cada título, pero también podemos hacer que MySQL realice el cálculo y lo incluya en una columna extra en la salida:

select titulo, precio,cantidad,precio*cantidadfrom libros;

Si queremos saber el precio de cada libro con un 10% de descuento podemos incluir en lasentencia los siguientes cálculos:
select titulo, precio,precio*0.1,precio-(precio*0.1)from libros;

Ejemplo práctico:

```
drop table if exists libros;create table libros(
codigo int unsigned auto_increment,
titulo varchar(40) not null,autor varchar(30), editorial varchar(15),
precio decimal(5,2) unsigned,cantidad smallint unsigned, primary key (codigo)
);

insert into libros (titulo,autor,editorial,precio,cantidad)values('El
Aleph','Borges','Planeta',15,100);
insert into libros (titulo,autor,editorial,precio,cantidad) values('Martin
Fierro','Jose Hernandez','Emece',22.20,200); insert into libros
(titulo,autor,editorial,precio,cantidad) values('Antologia
Poética','Borges','Planeta',40,150);
insert into libros (titulo,autor,editorial,precio,cantidad) values('Aprenda
PHP','Mario Molina','Emece',18.20,200); insert into libros
(titulo,autor,editorial,precio,cantidad) values('Cervantes y el
Quijote','Borges','Paidos',36.40,100);
insert into libros (titulo,autor,editorial,precio,cantidad) values('Manual de PHP',
'J.C. Paez', 'Paidos',30.80,100); insert into libros
(titulo,autor,editorial,precio,cantidad)
values('Harry Potter y la Piedra Filosofal','J.K. Rowling','Paidos',45.00,500);
insert into libros (titulo,autor,editorial,precio,cantidad)
values('Harry Potter y la Cámara Secreta','J.K. Rowling','Paidos',46.00,300);
insert into libros (titulo,autor,editorial,precio,cantidad)
values('Alicia en el País de las Maravillas','Lewis Carroll','Paidos',null,50);
```

select titulo, precio,cantidad,precio*cantidadfrom libros;

select titulo, precio,precio*0.1,precio-(precio*0.1)from libros;

Fuente: http://www.tutorialesprogramacionya.com

GROUP BY

Es posible obtener salidas en las cuales una columna sea el resultado de un cálculo y no uncampo de una tabla.
Si queremos ver los títulos, precio y cantidad de cada libro escribimos la siguientesentencia:

select titulo,precio,cantidad from libros;

Si queremos saber el monto total en dinero de un título podemos multiplicar el precio por lacantidad por cada título, pero también podemos hacer que MySQL realice el cálculo y lo incluya en una columna extra en la salida:

select titulo, precio,cantidad,precio*cantidad from libros;

Si queremos saber el precio de cada libro con un 10% de descuento podemos incluir en lasentencia los siguientes cálculos:

select titulo, precio,precio*0.1,precio-(precio*0.1) from libros;

Queremos saber la cantidad de visitantes de cada ciudad, podemos tipear la siguientesentencia:

select count(*) from visitantes where ciudad='Cordoba';

y repetirla con cada valor de "ciudad":

select count(*) from visitantes where ciudad='Alta Gracia'; select count(*) from visitantes where ciudad='Villa Dolores';

Pero hay otra manera, utilizando la cláusula "group by":

select ciudad, count(*) from visitantes group by ciudad;

Entonces, para saber la cantidad de visitantes que tenemos en cada ciudad utilizamos la función "count()", agregamos "group by" y el campo por el que deseamos que se realice elagrupamiento, también colocamos el nombre del campo a recuperar.
La instrucción anterior solicita que muestre el nombre de la ciudad y cuente la cantidad agrupando los registros por el campo "ciudad". Como resultado aparecen los nombres delas ciudades y la cantidad de registros para cada valor del campo.

Para obtener la cantidad de visitantes con teléfono no nulo, de cada ciudad utilizamos la función "count()" enviándole como argumento el campo "telefono",

agregamos "group by" yel campo por el que deseamos que se realice el agrupamiento (ciudad):

select ciudad, count(telefono) from visitantes group by ciudad;

Como resultado aparecen los nombres de las ciudades y la cantidad de registros de cada una, sin contar los que tienen teléfono nulo. Recuerde la diferencia de los valores que retorna la función "count()" cuando enviamos como argumento un asterisco o el nombre de un campo: en el primer caso cuenta todos los registros incluyendo los que tienen valor nulo,en el segundo, los registros en los cuales el campo especificado es no nulo.
Para conocer el total de las compras agrupadas por sexo:

select sexo, sum(montocompra) from visitantes group by sexo;

Para saber el máximo y mínimo valor de compra agrupados por sexo:

select sexo, max(montocompra), min(montocompra) from visitantes group by sexo;

Para calcular el promedio del valor de compra agrupados por ciudad:

select ciudad, avg(montocompra) from visitantes group by ciudad;

Podemos agrupar por más de un campo, por ejemplo, vamos a hacerlo por "ciudad" y"sexo":

select ciudad, sexo, count(*) from visitantes group by ciudad,sexo;

También es posible limitar la consulta con "where".
Vamos a contar y agrupar por ciudad sin tener en cuenta "Cordoba":

select ciudad, count(*) from visitantes where ciudad<>'Cordoba' group by ciudad;Ejemplo Práctico:

```
create table visitantes( nombre varchar(30), edad tinyint unsigned,sexo char(1),
domicilio varchar(30),ciudad varchar(20),
telefono varchar(11),
montocompra decimal (6,2) unsigned
);

insert into visitantes (nombre,edad,
sexo,domicilio,ciudad,telefono,montocompra)values ('Susana Molina',
28,'f','Colon 123','Cordoba',null,45.50);
insert into visitantes (nombre,edad,
```

```
sexo,domicilio,ciudad,telefono,montocompra) values ('Marcela
Mercado',36,'f','Avellaneda 345','Cordoba','4545454',0);
insert into visitantes (nombre,edad,
sexo,domicilio,ciudad,telefono,montocompra)values ('Alberto
Garcia',35,'m','Gral. Paz 123','Alta Gracia','03547123456',25); insert into
visitantes (nombre,edad, sexo,domicilio,ciudad,telefono,montocompra)values
('Teresa Garcia',33,'f','Gral. Paz 123','Alta Gracia','03547123456',0);
insert into visitantes (nombre,edad,
sexo,domicilio,ciudad,telefono,montocompra)values ('Roberto
Perez',45,'m','Urquiza 335','Cordoba','4123456',33.20);
insert into visitantes (nombre,edad,
sexo,domicilio,ciudad,telefono,montocompra)values ('Marina
Torres',22,'f','Colon 222','Villa Dolores','03544112233',25);
insert into visitantes (nombre,edad,
sexo,domicilio,ciudad,telefono,montocompra) values ('Julieta Gomez',24,'f','San
Martin 333','Alta Gracia','03547121212',53.50); insert into visitantes
(nombre,edad, sexo,domicilio,ciudad,telefono,montocompra) values ('Roxana
Lopez',20,'f','Triunvirato 345','Alta Gracia',null,0);
insert into visitantes (nombre,edad,
sexo,domicilio,ciudad,telefono,montocompra)values ('Liliana Garcia',50,'f','Paso
999','Cordoba','4588778',48);
insert into visitantes (nombre,edad,
sexo,domicilio,ciudad,telefono,montocompra)values ('Juan
Torres',43,'m','Sarmiento 876','Cordoba','4988778',15.30);

-- Para saber la cantidad de visitantes que tenemos de cada ciudad tipeamos:
select ciudad, count(*)
from visitantes group by ciudad;

-- Necesitamos conocer la cantidad visitantes con teléfono no nulo, de cada
ciudad:select ciudad, count(telefono)
from visitantes group by ciudad;

-- Queremos conocer el total de las compras agrupadas por sexo:select sexo,
sum(montocompra) from visitantes
group by sexo;

-- Para obtener el máximo y mínimo valor de compra agrupados por sexo:
select sexo, max(montocompra) from visitantes
group by sexo;
select sexo, min(montocompra) from visitantesgroup by sexo;

-- Se pueden simplificar las 2 sentencias anteriores en una sola sentencia, ya
que usan elmismo "group by":
```

```
select sexo, max(montocompra)
min(montocompra)from visitantes group by sexo;

-- Queremos saber el promedio del valor de compra agrupados por ciudad:
select ciudad, avg(montocompra) from visitantes
group by ciudad;

-- Contamos los registros y agrupamos por 2 campos, "ciudad" y "sexo":select
ciudad, sexo, count(*) from visitantes
group by ciudad,sexo;

-- Limitamos la consulta, no incluimos los visitantes de "Cordoba", contamos y
agrupar porciudad:
select ciudad, count(*) from visitanteswhere ciudad<>'Cordoba'
group by ciudad;
```

Fuente: http://www.tutorialesprogramacionya.com

HAVING

Así como la cláusula "where" permite seleccionar (o rechazar) registros individuales; lacláusula "having" permite seleccionar (o rechazar) un grupo de registros.
Si queremos saber la cantidad de libros agrupados por editorial usamos la siguienteinstrucción ya aprendida:

select editorial, count(*) from libros group by editorial;

Si queremos saber la cantidad de libros agrupados por editorial pero considerando sólo algunos grupos, por ejemplo, los que devuelvan un valor mayor a 2, usamos la siguienteinstrucción:

select editorial, count(*) from libros group by editorial having count(*)>2;

Se utiliza "having", seguido de la condición de búsqueda, para seleccionar ciertas filasretornadas por la cláusula "group by".
Veamos otros ejemplos. Queremos el promedio de los precios de los libros agrupados poreditorial:

select editorial, avg(precio) from libros group by editorial;

Ahora, sólo queremos aquellos cuyo promedio supere los 25 pesos:

select editorial, avg(precio) from libros group by editorial having avg(precio)>25;

Entendiendo las cláusulas HAVING y WHERE

En algunos casos es posible confundir las cláusulas "where" y "having".
Queremos contarlos registros agrupados por editorial sin tener en cuenta a la editorial "Planeta".
Analicemos las siguientes sentencias:

select editorial, count(*) from libros where editorial<>'Planeta' group by editorial; select editorial, count(*) from libros group by editorial having editorial<>'Planeta';

Ambas devuelven el mismo resultado, pero son diferentes.
La primera, selecciona todos los registros rechazando los de editorial "Planeta" y luego los agrupa para contarlos. La segunda, selecciona todos los registros, los agrupa para contarlosy finalmente rechaza la cuenta correspondiente a la

editorial "Planeta".

No debemos confundir la cláusula "where" con la cláusula "having"; la primera establece condiciones para la selección de registros de un "select"; la segunda establece condicionespara la selección de registros de una salida "group by".
Veamos otros ejemplos combinando "where" y "having".
Queremos la cantidad de libros, sin considerar los que tienen precio nulo, agrupados poreditorial, sin considerar la editorial "Planeta":

select editorial, count(*) from libroswhere precio is not null
group by editorial having editorial<>'Planeta';

Aquí, selecciona los registros rechazando los que no cumplan con la condición dada en "where", luego los agrupa por "editorial" y finalmente rechaza los grupos que no cumplancon la condición dada en el "having".
Generalmente se usa la cláusula "having" con funciones de agrupamiento, esto no puedehacerlo la cláusula "where". Por ejemplo queremos el promedio de los precios agrupados por editorial, de aquellas editoriales que tienen más de 2 libros:

select editorial, avg(precio) from librosgroup by editorial
having count(*) > 2;

Podemos encontrar el mayor valor de los libros agrupados por editorial y luego seleccionarlas filas que tengan un valor mayor o igual a 30:

select editorial, max(precio) from librosgroup by editorial
having max(precio)>=30;

Esta misma sentencia puede usarse empleando un "alias", para hacer referencia a lacolumna de la expresión:

select editorial, max(precio) as 'mayor' from librosgroup by editorial
having mayor>=30;

Ejemplo:

```
drop table if exists libros;create table libros(
codigo int unsigned auto_increment,titulo varchar(60) not null,
autor varchar(30), editorial varchar(15),
precio decimal(5,2) unsigned,primary key (codigo)
);

insert into libros (titulo,autor,editorial,precio)values('El
Aleph','Borges','Planeta',15); insert into libros (titulo,autor,editorial,precio)
```

```
values('Martin Fierro','Jose Hernandez','Emece',22.20);insert into libros
(titulo,autor,editorial,precio) values('Antología poética','Borges','Planeta',40);
insert into libros (titulo,autor,editorial,precio) values('Aprenda PHP','Mario
Molina','Emece',18.20); insert into libros (titulo,autor,editorial,precio)
values('Cervantes y el Quijote','Borges','Paidos',36.40); insert into libros
(titulo,autor,editorial,precio) values('Manual de PHP', 'J.C. Paez',
'Paidos',30.80); insert into libros (titulo,autor,editorial,precio)
values('Harry Potter y la Piedra Filosofal','J.K. Rowling','Paidos',45.00);insert
into libros (titulo,autor,editorial,precio)
values('Harry Potter y la Cámara Secreta','J.K. Rowling','Paidos',46.00);insert
into libros (titulo,autor,editorial,precio)
values('Alicia en el País de las Maravillas','Lewis Carroll','Paidos',null);

-- Queremos averiguar la cantidad de libros agrupados por editorial:select
editorial, count(*) from libros
group by editorial;

-- Queremos conocer la cantidad de libros agrupados por editorial pero
considerando
-- sólo los que devuelvan un valor mayor a 2select editorial, count(*) from libros
group by editorial having count(*)>2;

-- Necesitamos el promedio de los precios de los libros agrupados por editorial:
select editorial, avg(precio)
from libros
group by editorial;

-- sólo queremos aquellos cuyo promedio supere los 25 pesos:select editorial,
avg(precio)
from libros
group by editorial having avg(precio)>25;

-- Queremos contar los registros agrupados por editorial sin tener en cuenta
-- a la editorial "Planeta"
select editorial, count(*) from libroswhere editorial<>'Planeta'
group by editorial;
select editorial, count(*) from librosgroup by editorial
having editorial<>'Planeta';

-- Queremos la cantidad de libros, sin tener en cuenta los que tienen precio
nulo,
-- agrupados por editorial, rechazando los de editorial "Planeta"select editorial,
count(*) from libros
where precio is not nullgroup by editorial
```

```
having editorial<>'Planeta';

-- promedio de los precios agrupados por editorial, de aquellas editoriales
-- que tienen más de 2 libros
select editorial, avg(precio) from librosgroup by editorial
having count(*) > 2;

-- mayor valor de los libros agrupados por editorial y luego seleccionar las filas
-- que tengan un valor mayor o igual a 30select editorial, max(precio)
from libros
group by editorial
having max(precio)>=30;

-- Para esta misma sentencia podemos utilizar un "alias" para hacer referencia
a la
-- columna de la expresión
select editorial, max(precio) as 'mayor'from libros
group by editorial having mayor>=30;
```

Fuente: http://www.tutorialesprogramacionya.com

Procedimientos almacenados

Los procedimientos almacenados se crean en la base de datos seleccionada. En primer lugar se deben tipear y probar las instrucciones que se incluyen en el procedimiento almacenado, luego, si se obtiene el resultado esperado, se crea elprocedimiento.
Los procedimientos almacenados pueden hacer referencia a tablas, vistas y otrosprocedimientos almacenados.
Un procedimiento almacenado pueden incluir cualquier cantidad y tipo de instrucciones. Para crear un procedimiento almacenado empleamos la instrucción "create procedure".La sintaxis básica parcial es:

create procedure NOMBREPROCEDIMIENTO()begin
INSTRUCCIONES;
end

Con las siguientes instrucciones creamos un procedimiento almacenado llamado "pa_libros_limite_stock" que retorna todos los libros de los cuales hay menos de 10disponibles:

create procedure pa_libros_limite_stock()begin
select * from libroswhere stock<=10; end

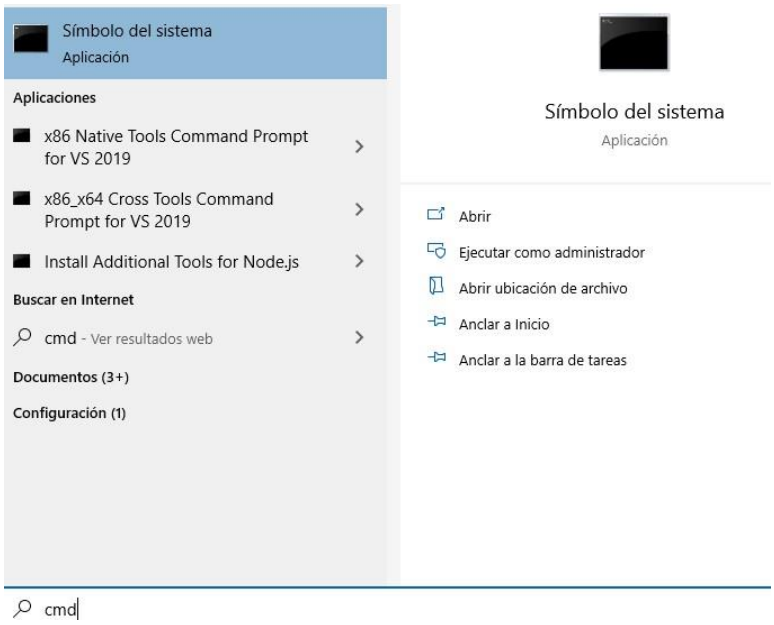
Para llamar luego al procedimiento almacenado debemos utilizar la cláusula 'call' yseguidamente el nombre del procedimiento almacenado:
call pa_libros_limite_stock();create procedure pa_libros_mayor_precio()
begin
select * from libroswhere precio>=15; end

call pa_libros_mayor_precio();

Fuente: http://www.tutorialesprogramacionya.com

Backup con mysqldump

1) Acceder a la ubicación del archivo mysqldump.exe con cmd (en modo administrador)



Si no se cambió la ruta de instalación, escribir:

cd C:\Program Files\MySQL\MySQL Server 8.0\bin

```
C:\WINDOWS\system32>cd C:\Program Files\MySQL\MySQL Server 8.0\bin
```

2) Hacer el backup

Escribir mysqldump -u *nombre de usuario* -p *nombreBD* > back.sqlPedirá el password del usuario root.

```
C:\Program Files\MySQL\MySQL Server 8.0\bin>mysqldump -u root -p ejemplo > back.sql
Enter password: ****
```

3) Restaurar base de datos

Escribir mysql -u *nombre de usuario* -p *nombreBD* < back.sql
Pedirá el password del usuario root. Debe estar creada la base de datos *nombreBD* en VS Code, que estará en primera instancia vacía hasta hacer el restore.

```
C:\Program Files\MySQL\MySQL Server 8.0\bin>mysql -u root -p ejemplo < back.sql
Enter password: ****
```

Fuente: https://www.tutorialesprogramacionya.com