

# MySQL

## 04\_ Operador y función



# MySQL\_ Operador y función.

## Contenido

|     |   |    |
|-----|---|----|
| 1.  | Funciones con cadenas                       | 4  |
| 2.  | Funciones matemáticas                       | 12 |
| 3.  | Funciones fecha/hora                        | 16 |
| 4.  | Order by                                    | 20 |
| 5.  | Operadores relacionales I                   | 23 |
| 6.  | Operadores relacionales II                  | 25 |
| 7.  | Operadores Lógicos (and - or - not)         | 28 |
| 8.  | Funciones de control de flujo (if)          | 32 |
| 9.  | Operadores de flujo CASE                    | 36 |
| 10. | Búsqueda de patrones (like y not like)      | 41 |
| 11. | Búsqueda de patrones (regexp)               | 44 |
| 12. | Contar registros (count)                    | 48 |
| 13. | Funciones de agrupamiento agregado          | 51 |
| 14. | Agrupar registros (group by)                | 54 |
| 15. | Selección de un grupo de registros (having) | 59 |
| 16. | Registros duplicados (distinct)             | 64 |
| 17. | Alias                                       | 68 |

# MySQL\_ Operador y función.

## Introducción

Los operadores y funciones son elementos importantes en una base de datos en MySQL que permiten la manipulación y análisis de los datos almacenados en las tablas.

Los operadores en MySQL se utilizan para realizar operaciones matemáticas, comparaciones y combinación de expresiones. Algunos de los operadores comunes en MySQL incluyen:

**Operadores aritméticos:** permiten realizar operaciones matemáticas como suma (+), resta (-), multiplicación (\*), división (/), entre otros.

**Operadores de comparación:** permiten comparar dos valores y devolver un valor de verdadero o falso. Algunos ejemplos son el operador igual (=), el operador mayor que (>), el operador menor que (<), entre otros.

**Operadores lógicos:** permiten combinar expresiones de comparación para evaluar una condición más compleja. Algunos ejemplos son el operador AND, el operador OR, el operador NOT, entre otros.

Las funciones en MySQL son herramientas que permiten realizar operaciones más complejas sobre los datos almacenados en las tablas. Las funciones pueden ser de varios tipos, como funciones de agregación (SUM, COUNT, AVG), funciones de fecha y hora (NOW, DATE\_FORMAT), funciones matemáticas (SIN, COS, LOG), entre otras. Las funciones se utilizan comúnmente en las consultas SQL para filtrar y analizar los datos en una tabla.

En resumen, los operadores y funciones son herramientas fundamentales en una base de datos en MySQL que permiten la manipulación y análisis de los datos almacenados en las tablas.



# MySQL\_ Operador y función.

## 1. Funciones para el manejo de cadenas

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 o campo que tenga el mismo nombre de una función.

MySQL tiene algunas funciones para trabajar con cadenas de caracteres. Estas son algunas:

-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 enviados como 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". El primer 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".

-find\_in\_set(cadena,lista de cadenas): devuelve un valor entre de 0 a n (correspondiente a la posición), si la cadena enviada como primer argumento está presente en la lista de cadenas enviadas como segundo argumento. La lista de cadenas enviada como segundo argumento es una cadena formada por subcadenas separadas por comas. Devuelve 0 si no encuentra "cadena" en la "lista de cadenas".

## MySQL\_ Operador y función.

Ejemplo:

```
select find_in_set('hola','como esta,hola,buen dia');
```

retorna 2, porque la cadena "hola" se encuentra en la lista de cadenas, en la posición 2.

-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 en la cadena.

Ejemplo:

```
select locale('o','como le va');
```

retorna 2.

- position(subcadena in cadena): funciona como "locate()". Devuelve "0" si la subcadena no se encuentra en la cadena. Ejemplo:

```
select position('o' in 'como le va');
```

retorna 2.

- locate(subcadena,cadena,posicioninicial): retorna la posición de la primera ocurrencia de la subcadena enviada como primer argumentos en la cadena enviada como segundo argumento, empezando en la posición enviada como tercer argumento. Devuelve "0" si la subcadena no se encuentra en la cadena. Ejemplos:

```
select locate('ar','Margarita',1);
```

retorna 1.

```
select locate('ar','Margarita',3);
```

retorna 5.

- instr(cadena,subcadena): retorna la posición de la primera ocurrencia de la subcadena enviada como segundo argumento en la cadena enviada como primer argumento. Ejemplo:

```
select instr('como le va','om');
```

devuelve 2.

- lpad(cadena,longitud,cadenarelleno): retorna la cadena enviada como primer argumento, rellena por la izquierda con la cadena enviada como tercer argumento hasta que 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 la derecha.

## MySQL\_ Operador y función.

- left(cadena,longitud): retorna la cantidad (longitud) de caracteres de la cadena comenzando 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 cadena comenzando desde la derecha, último caracter. Ejemplo:

```
select right('buenos dias',8);
```

retorna "nos dias".

- substring(cadena,posicion,longitud): retorna una subcadena de tantos caracteres de longitud como especifica en tercer argumento, de la cadena enviada como primer argumento, empezando desde la posición especificada en el segundo argumento. Ejemplo:

```
select substring('Buenas tardes',3,5);
```

retorna "enas".

- substring(cadena from posicion for longitud): variante de "substring(cadena,posicion,longitud)". Ejemplo:

```
select substring('Buenas tardes' from 3 for 5);
```

- mid(cadena,posicion,longitud): igual que "substring(cadena,posicion,longitud)". Ejemplo:

```
select mid('Buenas tardes' from 3 for 5);
```

retorna "enas".

- substring(cadena,posicion): retorna la subcadena de la cadena enviada como argumento, empezando desde la posición indicada por el segundo argumento. Ejemplo:

```
select substring('Margarita',4);
```

retorna "garita".

-substring(cadena from posicion): variante de "substring(cadena,posicion)". Ejemplo:

```
select substring('Margarita' from 4);
```

retorna "garita".

-substring\_index(cadena,delimitador,ocurrencia): retorna la subcadena de la cadena enviada como argumento antes o después de la "ocurrencia" de la cadena enviada como delimitador. Si "ocurrencia" es positiva, retorna la subcadena anterior al delimitador (comienza desde la izquierda); si "ocurrencia" es negativa, retorna la subcadena posterior al delimitador (comienza desde la derecha). Ejemplo:

```
select substring_index( 'margarita','ar',2);
```

retorna "marg", todo lo anterior a la segunda ocurrencia de "ar".

```
select substring_index( 'margarita','ar',-2);
```

retorna "garita", todo lo posterior a la segunda ocurrencia de "ar".

## MySQL\_ Operador y función.

-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".

```
select trim ('  hola  ');
```

retorna "hola".

-replace(cadena,cadenareemplazo,cadenareemplazar): retorna la cadena con todas las ocurrencias de la subcadena reemplazo por la subcadena a reemplazar. Ejemplo:

```
select replace('xxx.mysql.com','x','w');
```

retorna "www.mysql.com".

-repeat(cadena,cantidad): devuelve una cadena consistente en la cadena repetida la cantidad de veces especificada. Si "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 order de los caracteres. Ejemplo:

```
select reverse('Hola');
```

## MySQL\_ Operador y función.

retorna "aloH".

-insert(cadena,posicion,longitud,nuevacadena): retorna la cadena con la nueva cadena colocándola en la posición indicada por "posicion" y elimina la cantidad de caracteres indicados por "longitud". Ejemplo:

```
select insert('buenas tardes',2,6,'xx');
```

retorna "'bxxtardes".

-lcase(cadena) y lower(cadena): retornan la cadena con todos los caracteres en minúsculas. Ejemplo:

```
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 en mayúsculas. Ejemplo:

```
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 menor que la segunda y 1 si la primera es mayor que la segunda. Ejemplo:

```
select strcmp('Hola','Chau');
```

retorna 1.

Un listado completo de todas las funciones para el manejo de cadenas de caracteres de las diferentes versiones de MySQL las podemos consultar en la [documentación oficial](#).

Veamos ejemplos trabajando con campos

```
create table alumnos (  
    nombre varchar(30),  
    apellidos varchar(30),  
    clave varchar(10)  
);  
insert into alumnos (nombre,apellidos,clave)  
values ('luis','gracia','fumador');  
insert into alumnos (nombre,apellidos,clave)  
values ('ana','perez','elfa');  
insert into alumnos (nombre,apellidos,clave)  
values ('victoria','cuevas','gata');  
insert into alumnos (nombre,apellidos,clave)  
values ('margarita','revuelta','flor');  
insert into alumnos (nombre,apellidos,clave)
```



## MySQL\_ Operador y función.

```
values (' pedro  ','alvarez','conejo');
insert into alumnos (nombre,apellidos,clave)
values ('XXXXandresXXXX','vazquez','perro');
insert into alumnos (nombre,apellidos,clave)
values ('FELIX','PEREZ','FOCA');
select concat(nombre,' ',apellidos) from alumnos;
select concat_ws("-",nombre,apellidos) from alumnos;
select length(apellidos) from alumnos where nombre="ana";
select instr(apellidos,'vas') from alumnos where nombre="victoria";
select lpad(apellidos,10,'X') from alumnos where nombre="victoria";
select rpad(apellidos,10,'X') from alumnos where nombre="victoria";
select left(apellidos,3) from alumnos where nombre="victoria";
select right(apellidos,3) from alumnos where nombre="victoria";
select substring(apellidos,3,5) from alumnos where nombre="victoria";
select substring(apellidos from 3 for 5) from alumnos where nombre="victoria";
select substring(apellidos ,3) from alumnos where nombre="victoria";
select substring(apellidos from 3) from alumnos where nombre="victoria";
select mid(apellidos from 3 for 5) from alumnos where nombre="victoria";
select substring_index( nombre,'ar',2) from alumnos where apellidos="revuelta";
select substring_index( nombre,'ar',-2) from alumnos where apellidos="revuelta";
select nombre from alumnos where apellidos="alvarez";
select ltrim(nombre) from alumnos where apellidos="alvarez";
select rtrim(nombre) from alumnos where apellidos="alvarez";
select trim(nombre) from alumnos where apellidos="alvarez";
select nombre from alumnos where apellidos="suarez";
select trim(leading 'X' from nombre) from alumnos where apellidos="vazquez";
select trim(trailing 'X' from nombre) from alumnos where apellidos="vazquez";
select trim(both 'X' from nombre) from alumnos where apellidos="vazquez";
select trim('X' from nombre) from alumnos where apellidos="vazquez";
select replace(apellidos,'z','s') from alumnos ;
select repeat(apellidos,5) from alumnos ;
select insert(apellidos,2,1,"xxx") from alumnos ;
select reverse(apellidos) from alumnos;
select lower(apellidos) from alumnos ;
select lcase(apellidos) from alumnos ;
select upper(apellidos) from alumnos ;
select ucase(apellidos) from alumnos ;
select replace(replace(ucase(apellidos),'Z','S'),'A',"B") from alumnos ;
```

## MySQL\_ Operador y función.

### Servidor de MySQL instalado en forma local.

Ingrese al programa "Workbench" y ejecutemos el siguiente bloque de instrucciones SQL llamando a funciones para el manejo de cadenas en MySQL:

```
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 alehp','Borges','Paidos',33.4);
insert into libros (titulo,autor,editorial,precio)
  values('Alicia en el pais de las maravillas','L. Carroll','Planeta',16);

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

select left(titulo,15)
  from libros;

select titulo,insert(editorial,1,0,'edit. ')
  from libros;

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

Que nos genera una salida similar a esta:

## MySQL\_ Operador y función.

The screenshot shows a MySQL Query Editor window with a query editor at the top and a result grid at the bottom. The query editor contains several SQL statements, with the last one highlighted in a red box. The result grid shows the output of the highlighted query, also highlighted in a red box.

```
13 values('El alehp', 'Borges', 'Paidos', 35.4);
14 insert into libros (titulo, autor, editorial, precio)
15 values('Alicia en el pais de las maravillas', 'L. Carroll', 'Planeta', 16);
16
17 select concat_ws('-', titulo, autor)
18 from libros;
19
20 select left(titulo, 15)
21 from libros;
22
23 select titulo, insert(editorial, 1, 8, 'edit. ')
24 from libros;
25
26 select lower(titulo), upper(editorial)
27 from libros;
```

The result grid shows the output of the highlighted query (Result 4):

| lower(titulo)                       | upper(editorial) |
|-------------------------------------|------------------|
| el alehp                            | PAIDOS           |
| alicia en el pais de las maravillas | PLANETA          |

# MySQL\_ Operador y función.

## 2. Funciones matemáticas

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

```
select 5/0;
```

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 o campo 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),
```

retorna 13.

-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.

-dividendo%divisor: devuelve el resto de una división. Ejemplos:

```
select 10%3;
```

retorna 1.

```
select 10%2;
```

retorna 0.

-power(x,y): retorna el valor de "x" elevado a la "y" potencia. Ejemplo:

```
select power(2,3);
```

## MySQL\_ Operador y función.

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 raíz cuadrada del valor enviado como argumento.

-truncate(x,d): retorna el número "x", truncado a "d" decimales. Si "d" es 0, el resultado no tendrá 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.

Un listado completo de todas las funciones matemáticas de las diferentes versiones de MySQL las podemos consultar en la [documentación oficial](#).

# MySQL\_ Operador y función.

## Servidor de MySQL instalado en forma local.

Ingresemos al programa "Workbench" y ejecutemos el siguiente bloque de instrucciones SQL para probar algunas funciones matemáticas de MySQL:

```
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 alehp','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;
```

## MySQL\_ Operador y función.

Que nos genera una salida similar a esta:

The screenshot displays a MySQL query editor window titled "Query 1". The SQL code is as follows:

```
8      precio decimal(5,2) unsigned,  
9      primary key(codigo)  
10     );  
11  
12     insert into libros (titulo,autor,editorial,precio)  
13     values('El alehp','Borges','Paidós',33.45);  
14     insert into libros (titulo,autor,editorial,precio)  
15     values('Alicia en el país de las maravillas','L. Carroll','Planeta',16.31);  
16     insert into libros (titulo,autor,editorial,precio)  
17     values('Alicia a través del espejo','L. Carroll','Planeta',18.89);  
18  
19     select titulo, ceiling(precio), floor(precio)  
20     from libros;  
21  
22     select titulo, round(precio)  
23     from libros;  
24  
25     select titulo, truncate(precio,1)  
26     from libros;  
27  
28
```

The results grid shows the output of the last query (Result 3). The columns are "titulo" and "truncate(precio,1)". The data is as follows:

| titulo                              | truncate(precio,1) |
|-------------------------------------|--------------------|
| El alehp                            | 33.4               |
| Alicia en el país de las maravillas | 16.3               |
| Alicia a través del espejo          | 18.8               |

# MySQL\_ Operador y función.

## 3. Funciones para uso Fecha y Hora

MySQL tiene algunas funciones para trabajar con fechas y horas. Estas son algunas:

-`adddate(fecha, interval expresion)`: retorna la fecha agregándole el intervalo especificado. Ejemplos: `adddate('2006-10-10',interval 25 day)` retorna "2006-11-04". `adddate('2006-10-10',interval 5 month)` retorna "2007-03-10".

-`adddate(fecha, dias)`: retorna la fecha agregándole a fecha "dias". Ejemplo: `adddate('2006-10-10',25)`, retorna "2006-11-04".

-`addtime(expresion1,expresion2)`: agrega expresion2 a expresion1 y retorna el resultado.

-`current_date`: retorna la fecha de hoy con formato "YYYY-MM-DD" o "YYYYMMDD".

-`current_time`: retorna la hora actual con formato "HH:MM:SS" o "HHMMSS".

-`date_add(fecha,interval expresion tipo)` y `date_sub(fecha,interval expresion tipo)`: el argumento "fecha" es un valor "date" o "datetime", "expresion" especifica el valor de intervalo a ser añadido o sustraído de la fecha indicada (puede empezar con "-", para intervalos negativos), "tipo" indica la medida de adición o sustracción. Ejemplo: `date_add('2006-08-10', interval 1 month)` retorna "2006-09-10"; `date_add('2006-08-10', interval -1 day)` retorna "2006-09-09"; `date_sub('2006-08-10 18:55:44', interval 2 minute)` retorna "2006-08-10 18:53:44"; `date_sub('2006-08-10 18:55:44', interval '2:3' minute_second)` retorna "2006-08-10 18:52:41". Los valores para "tipo" pueden ser: second, minute, hour, day, month, year, minute\_second (minutos y segundos), hour\_minute (horas y minutos), day\_hour (días y horas), year\_month (año y mes), hour\_second (hora, minuto y segundo), day\_minute (dias, horas y minutos), day\_second(dias a segundos).

-`datediff(fecha1,fecha2)`: retorna la cantidad de días entre fecha1 y fecha2.

-`dayname(fecha)`: retorna el nombre del día de la semana de la fecha. Ejemplo: `dayname('2006-08-10')` retorna "thursday".

-`dayofmonth(fecha)`: retorna el día del mes para la fecha dada, dentro del rango 1 a 31. Ejemplo: `dayofmonth('2006-08-10')` retorna 10.

-`dayofweek(fecha)`: retorna el índice del día de semana para la fecha pasada como argumento. Los valores de los índices son: 1=domingo, 2=lunes,... 7=sábado). Ejemplo: `dayofweek('2006-08-10')` retorna 5, o sea jueves.

-`dayofyear(fecha)`: retorna el día del año para la fecha dada, dentro del rango 1 a 366. Ejemplo: `dayofyear('2006-08-10')` retorna 222.

-`extract(tipo from fecha)`: extrae partes de una fecha.



## MySQL\_ Operador y función.

Ejemplos:

```
extract(year from '2006-10-10'), retorna "2006".  
extract(year_month from '2006-10-10 10:15:25') retorna "200610".  
extract(day_minute from '2006-10-10 10:15:25') retorna "101015";
```

Los valores para tipo pueden ser: second, minute, hour, day, month, year, minute\_second, hour\_minute, day\_hour, year\_month, hour\_second (horas, minutos y segundos), day\_minute (días, horas y minutos), day\_second (días a segundos).

-hour(hora): retorna la hora para el dato dado, en el rango de 0 a 23.

Ejemplo: hour('18:25:09') retorna "18";

-minute(hora): retorna los minutos de la hora dada, en el rango de 0 a 59.

-monthname(fecha): retorna el nombre del mes de la fecha dada.

Ejemplo: monthname('2006-08-10') retorna "August".

-month(fecha): retorna el mes de la fecha dada, en el rango de 1 a 12.

-now() y sysdate(): retornan la fecha y hora actuales.

-period\_add(p,n): agrega "n" meses al periodo "p", en el formato "YYMM" o "YYYYMM"; retorna un valor en el formato "YYYYMM". El argumento "p" no es una fecha, sino un año y un mes. Ejemplo: period\_add('200608',2) retorna "200610".

-period\_diff(p1,p2): retorna el número de meses entre los períodos "p1" y "p2", en el formato "YYMM" o "YYYYMM". Los argumentos de período no son fechas sino un año y un mes. Ejemplo: period\_diff('200608','200602') retorna 6.

-second(hora): retorna los segundos para la hora dada, en el rango de 0 a 59.

-sec\_to\_time(segundos): retorna el argumento "segundos" convertido a horas, minutos y segundos. Ejemplo: sec\_to\_time(90) retorna "1:30".

-timediff(hora1,hora2): retorna la cantidad de horas, minutos y segundos entre hora1 y hora2.

-time\_to\_sec(hora): retorna el argumento "hora" convertido en segundos.

-to\_days(fecha): retorna el número de día (el número de día desde el año 0).

-weekday(fecha): retorna el índice del día de la semana para la fecha pasada como argumento. Los índices son: 0=lunes, 1=martes,... 6=domingo). Ejemplo: weekday('2006-08-10') retorna 3, o sea jueves.

-year(fecha): retorna el año de la fecha dada, en el rango de 1000 a 9999. Ejemplo: year('06-08-10') retorna "2006".

Un listado completo de todas las funciones relacionadas a fechas y horas las podemos consultar en la [documentación oficial](#).

## MySQL\_ Operador y función.

### Servidor de MySQL instalado en forma local.

Ingresemos al programa "Workbench" y ejecutemos el siguiente bloque de instrucciones SQL empleando algunas funciones para el manejo de fechas y horas:

```
drop table if exists prestamos;

create table prestamos(
  titulo varchar(40) not null,
  documento char(8) not null,
  fechaprestamo date not null,
  fechadevolucion date,
  devuelto char(1) default 'n'
);

insert into prestamos (titulo,documento,fechaPrestamo,fechaDevolucion)
values ('Manual de 1 grado','23456789','2006-08-10',date_add('2006-08-10', interval 5 day));

select * from prestamos;

insert into prestamos (titulo,documento,fechaPrestamo,fechaDevolucion)
values ('Alicia en el pais de las maravillas','23456789',
'2006-08-12',date_add('2006-08-12', interval 5 day));
insert into prestamos (titulo,documento,fechaPrestamo,fechaDevolucion)
values ('El aleph','22543987','2006-08-15',date_add('2006-08-15', interval 5 day));
insert into prestamos (titulo,documento,fechaPrestamo,fechaDevolucion)
values ('Manual de geografia 5 grado','25555666','2006-08-30',
date_add('2006-08-30', interval 5 day));

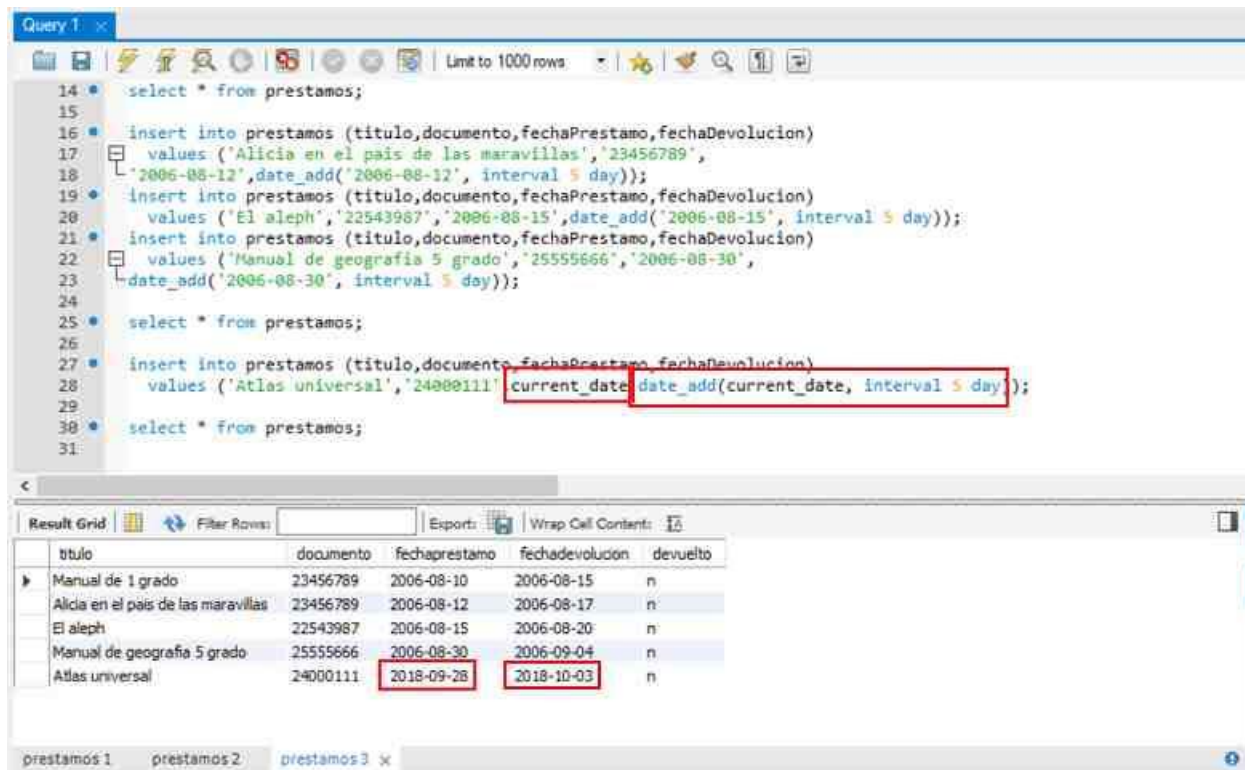
select * from prestamos;

insert into prestamos (titulo,documento,fechaPrestamo,fechaDevolucion)
values ('Atlas universal','24000111',current_date,date_add(current_date, interval 5 day));

select * from prestamos;
```

Que nos genera una salida similar a esta:

## MySQL\_ Operador y función.



The screenshot displays a MySQL query editor window titled "Query 1". The SQL code is as follows:

```
14 select * from prestamos;
15
16 insert into prestamos (titulo,documento,fechaPrestamo,fechaDevolucion)
17 values ('Alicia en el país de las maravillas','23456789',
18 '2006-08-12',date_add('2006-08-12', interval 5 day));
19 insert into prestamos (titulo,documento,fechaPrestamo,fechaDevolucion)
20 values ('El aleph','22543987','2006-08-15',date_add('2006-08-15', interval 5 day));
21 insert into prestamos (titulo,documento,fechaPrestamo,fechaDevolucion)
22 values ('Manual de geografía 5 grado','25555666','2006-08-30',
23 date_add('2006-08-30', interval 5 day));
24
25 select * from prestamos;
26
27 insert into prestamos (titulo,documento,fechaPrestamo,fechaDevolucion)
28 values ('Atlas universal','24000111',current_date,date_add(current_date, interval 5 day));
29
30 select * from prestamos;
31
```

The results are shown in a "Result Grid" table with the following data:

| titulo                              | documento | fechaprestamo | fechadevolucion | devuelto |
|-------------------------------------|-----------|---------------|-----------------|----------|
| Manual de 1 grado                   | 23456789  | 2006-08-10    | 2006-08-15      | n        |
| Alicia en el país de las maravillas | 23456789  | 2006-08-12    | 2006-08-17      | n        |
| El aleph                            | 22543987  | 2006-08-15    | 2006-08-20      | n        |
| Manual de geografía 5 grado         | 25555666  | 2006-08-30    | 2006-09-04      | n        |
| Atlas universal                     | 24000111  | 2018-09-28    | 2018-10-03      | n        |

# MySQL\_ Operador y función.

## 4. Cláusula order by del select.

Podemos ordenar el resultado de un "select" para que los registros se muestren ordenados 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.

También podemos colocar el número de orden del campo por el que queremos que se ordene en lugar de su nombre. Por ejemplo, queremos el resultado del "select" ordenado por "precio":

```
select codigo,titulo,autor,editorial,precio from libros order by 5;
```

Por defecto, si no aclaramos en la sentencia, los ordena de manera ascendente (de menor a 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;
```

También podemos ordenar por varios campos, por ejemplo, por "titulo" y "editorial":

```
select codigo,titulo,autor,editorial,precio from libros order by titulo, editorial;
```

Incluso, podemos ordenar en distintos sentidos, por ejemplo, por "titulo" en sentido ascendente y "editorial" en sentido descendente:

```
select codigo,titulo,autor,editorial,precio  
from libros order by titulo asc, editorial desc;
```

Debe aclararse al lado de cada campo, pues estas palabras claves afectan al campo inmediatamente anterior.

# MySQL\_ Operador y función.

## Servidor de MySQL instalado en forma local.

Ingresemos al programa "Workbench" y ejecutemos el siguiente bloque de instrucciones SQL:

```
drop table if exists libros;

create table libros(
  codigo int unsigned auto_increment,
  titulo varchar(40),
  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',15.50);
insert into libros (titulo,autor,editorial,precio)
  values('Martin Fierro','Jose Hernandez','Emece',22.90);
insert into libros (titulo,autor,editorial,precio)
  values('Martin Fierro','Jose Hernandez','Planeta',39);
insert into libros (titulo,autor,editorial,precio)
  values('Aprenda PHP','Mario Molina','Emece',19.50);
insert into libros (titulo,autor,editorial,precio)
  values('Cervantes y el quijote','Borges','Paidos',35.40);
insert into libros (titulo,autor,editorial,precio)
  values('Matematica estas ahi', 'Paenza', 'Paidos',19);

select codigo,titulo,autor,editorial,precio
  from libros
  order by titulo;

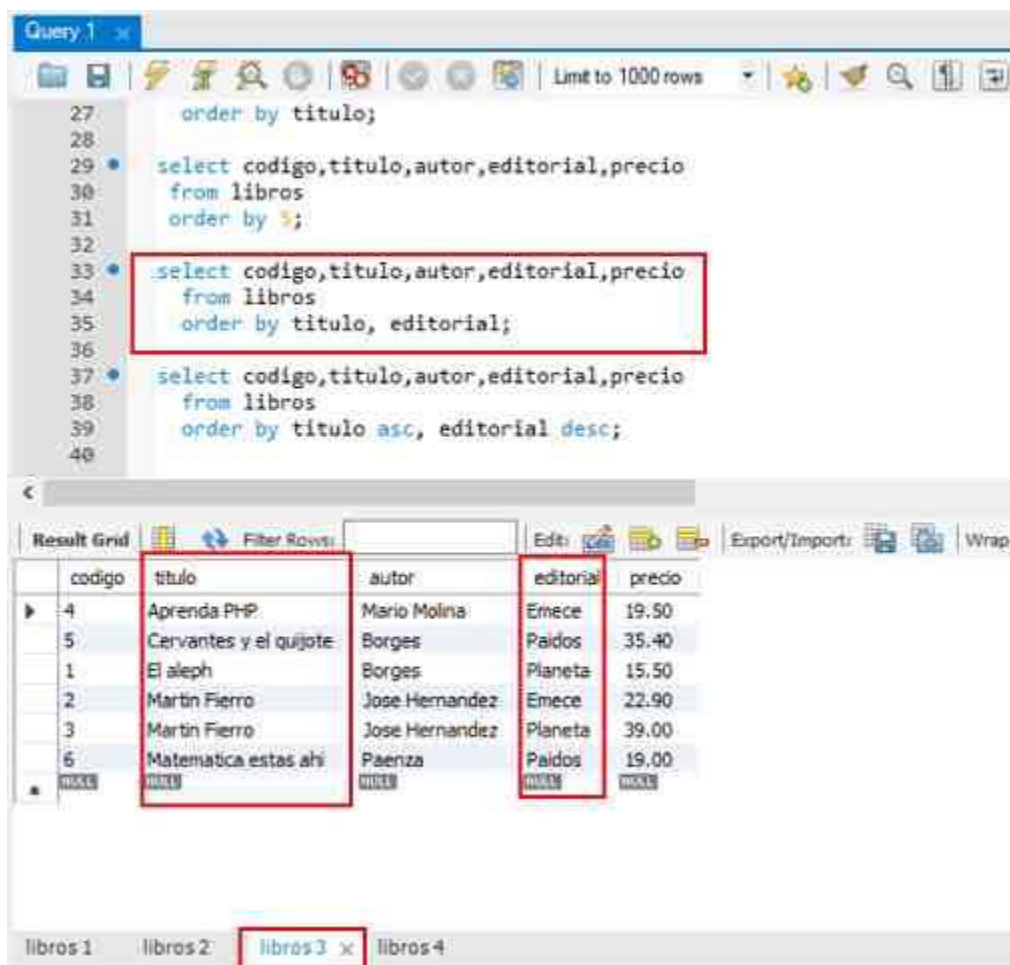
select codigo,titulo,autor,editorial,precio
  from libros
  order by 5;

select codigo,titulo,autor,editorial,precio
  from libros
  order by titulo, editorial;

select codigo,titulo,autor,editorial,precio
  from libros
  order by titulo asc, editorial desc;
```

## MySQL\_ Operador y función.

Que nos genera una salida similar a esta:



The screenshot shows the MySQL Query Editor interface. The top pane displays a SQL query: `select codigo,titulo,autor,editorial,precio from libros order by titulo, editorial;`. The bottom pane shows the result grid with 6 rows of data. The columns are: codigo, titulo, autor, editorial, and precio. The data is sorted by titulo and then by editorial.

|   | codigo | titulo                 | autor          | editorial | precio |
|---|--------|------------------------|----------------|-----------|--------|
| 4 | 4      | Aprenda PHP            | Mario Molina   | Emece     | 19.50  |
| 5 | 5      | Cervantes y el quijote | Borges         | Paidós    | 35.40  |
| 1 | 1      | El aleph               | Borges         | Planeta   | 15.50  |
| 2 | 2      | Martin Fierro          | Jose Hernandez | Emece     | 22.90  |
| 3 | 3      | Martin Fierro          | Jose Hernandez | Planeta   | 39.00  |
| 6 | 6      | Matematica estas ahi   | Paenza         | Paidós    | 19.00  |

# MySQL\_ Operador y función.

## 5. Operadores Relacionales = <> < <= > >=

Hemos aprendido a especificar condiciones de igualdad para seleccionar registros de una tabla; por ejemplo:

```
select titulo,autor,editorial from libros where autor='Borges';
```

Utilizamos el operador relacional de igualdad.

Los operadores relacionales vinculan un campo con un valor para que MySQL compare cada 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 la condición:

```
select titulo,autor,editorial from libros where autor<>'Borges';
```

Podemos comparar valores numéricos. Por ejemplo, queremos mostrar los libros cuyos precios 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;
```

### Servidor de MySQL instalado en forma local.

Ingresemos al programa "Workbench" y ejecutamos los siguientes comandos SQL donde utilizamos los operadores relacionales:

```
drop table if exists libros;
```

```
create table libros(
  titulo varchar(20),
  autor varchar(30),
  editorial varchar(15),
  precio float
);
```

## MySQL\_ Operador y función.

```
insert into libros (titulo,autor,editorial,precio) values ('El aleph','Borges','Planeta',12.50);
insert into libros (titulo,autor,editorial,precio) values ('Martin Fierro','Jose
Hernandez','Emece',16.00);
insert into libros (titulo,autor,editorial,precio) values ('Aprenda PHP','Mario
Molina','Emece',35.40);
insert into libros (titulo,autor,editorial,precio) values ('Cervantes','Borges','Paidos',50.90);
```

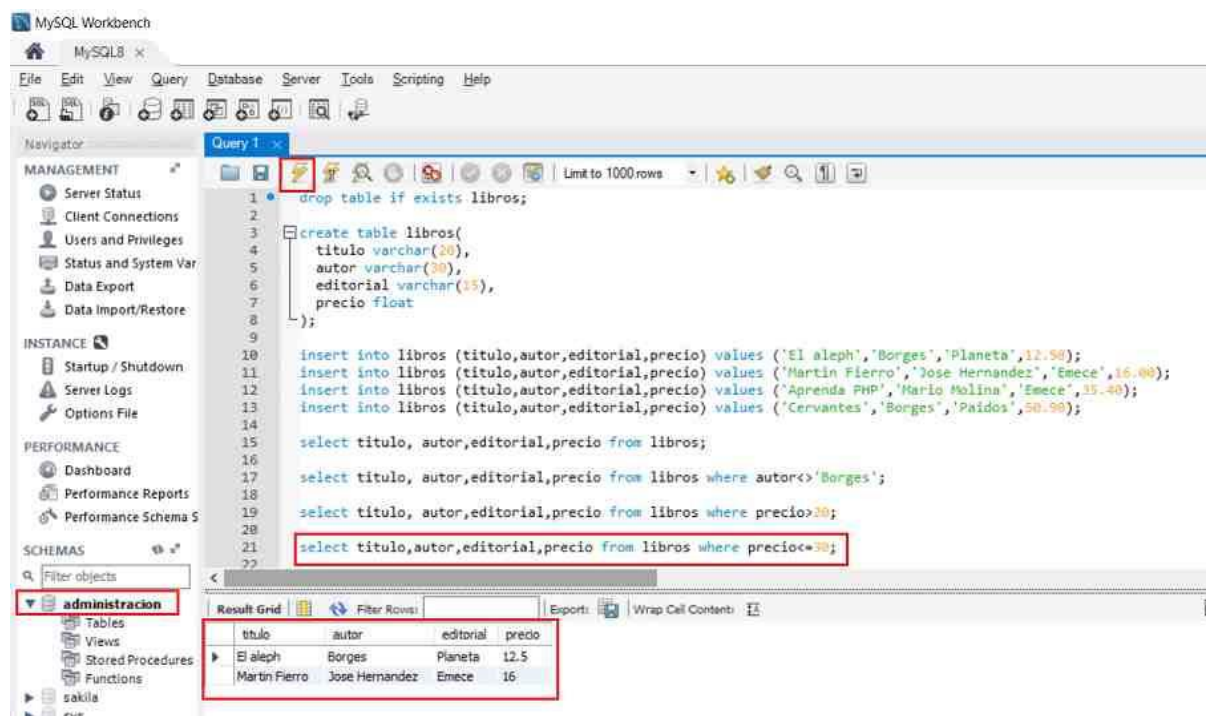
```
select titulo, autor,editorial,precio from libros;
```

```
select titulo, autor,editorial,precio from libros where autor<>'Borges';
```

```
select titulo, autor,editorial,precio from libros where precio>20;
```

```
select titulo,autor,editorial,precio from libros where precio<=30;
```

Tenemos como resultado:



The screenshot shows the MySQL Workbench interface. The 'Query' tab is active, displaying a SQL script. The script includes a 'drop table if exists libros;' statement, followed by a 'create table libros' statement with columns: titulo (varchar(20)), autor (varchar(30)), editorial (varchar(15)), and precio (float). Below the create statement are four 'insert into libros' statements with the following values: ('El aleph', 'Borges', 'Planeta', 12.50), ('Martin Fierro', 'Jose Hernandez', 'Emece', 16.00), ('Aprenda PHP', 'Mario Molina', 'Emece', 35.40), and ('Cervantes', 'Borges', 'Paidos', 50.90). The script concludes with three 'select' statements: 'select titulo, autor, editorial, precio from libros;', 'select titulo, autor, editorial, precio from libros where autor <> 'Borges';', and 'select titulo, autor, editorial, precio from libros where precio <= 30;'. The last 'select' statement is highlighted with a red box. The 'Result Grid' at the bottom shows the output of the last query, displaying two rows of data. The first row is 'El aleph' by 'Borges' from 'Planeta' with a price of 12.5. The second row is 'Martin Fierro' by 'Jose Hernandez' from 'Emece' with a price of 16. The 'Result Grid' is also highlighted with a red box.

| titulo        | autor          | editorial | precio |
|---------------|----------------|-----------|--------|
| El aleph      | Borges         | Planeta   | 12.5   |
| Martin Fierro | Jose Hernandez | Emece     | 16     |



# MySQL\_ Operador y función.

## 6. Operadores relacionales : between - in

Hemos visto los operadores relacionales:

= (igual), <> (distinto), > (mayor), < (menor), >= (mayor o igual), <= (menor o igual), is null/is not null (si un valor es NULL o no).

Existen otros que simplifican algunas consultas:

Para recuperar de nuestra tabla "libros" los registros que tienen precio mayor o igual a 20 y menor 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 entre los 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 de valores 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.

# MySQL\_ Operador y función.

## Servidor de MySQL instalado en forma local.

Ingresemos al programa "Workbench" y ejecutemos el siguiente bloque de instrucciones SQL para utilizar los operadores relacionales 'between' y 'in':

```
drop table if exists libros;

create table libros(
  codigo int unsigned auto_increment,
  titulo varchar(40),
  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',15.50);
insert into libros (titulo,autor,editorial,precio)
  values('Martin Fierro','Jose Hernandez','Emece',22.90);
insert into libros (titulo,autor,editorial,precio)
  values('Martin Fierro','Jose Hernandez','Planeta',39);
insert into libros (titulo,autor,editorial,precio)
  values('Aprenda PHP','Mario Molina','Emece',19.50);
insert into libros (titulo,autor,editorial,precio)
  values('Cervantes y el quijote','Borges','Paidos',35.40);
insert into libros (titulo,autor,editorial,precio)
  values('Matematica estas ahi', 'Paenza', 'Paidos',19);

select * from libros
  where precio>=20 and
  precio<=40;

select * from libros
  where precio between 20 and 40;

select * from libros
  where autor='Borges' or
  autor='Paenza';

select * from libros
  where autor in('Borges','Paenza');

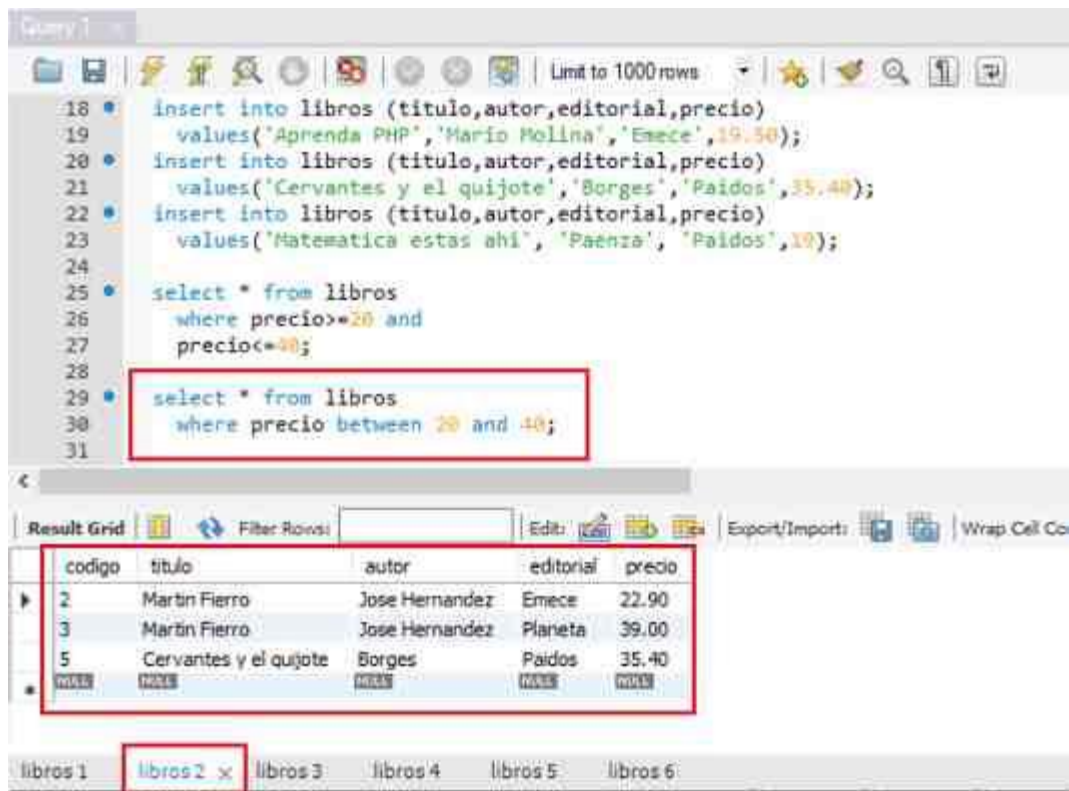
select * from libros
  where autor<>'Borges' and
  autor<>'Paenza';

select * from libros
```

## MySQL\_ Operador y función.

where autor not in ('Borges','Paenza');

Que nos genera una salida similar a esta:



The screenshot shows a MySQL query editor with a query window and a result grid. The query window contains SQL code for inserting data into a 'libros' table and selecting data based on price. The result grid shows the output of the query, displaying columns: codigo, titulo, autor, editorial, and precio. The results are as follows:

| codigo | titulo                 | autor          | editorial | precio |
|--------|------------------------|----------------|-----------|--------|
| 2      | Martin Fierro          | Jose Hernandez | Emece     | 22.90  |
| 3      | Martin Fierro          | Jose Hernandez | Planeta   | 39.00  |
| 5      | Cervantes y el quijote | Borges         | Paidos    | 35.40  |
| NULL   | NULL                   | NULL           | NULL      | NULL   |

The query editor shows the following SQL code:

```
18 insert into libros (titulo,autor,editorial,precio)
19 values('Aprenda PHP','Mario Molina','Emece',19.50);
20 insert into libros (titulo,autor,editorial,precio)
21 values('Cervantes y el quijote','Borges','Paidos',35.40);
22 insert into libros (titulo,autor,editorial,precio)
23 values('Matematica estas ahi','Paenza','Paidos',19);
24
25 select * from libros
26 where precio >= 20 and
27 precio <= 40;
28
29 select * from libros
30 where precio between 20 and 40;
31
```

# MySQL\_ Operador y función.

## 7. Operadores Lógicos (and - or - not)

Hasta el momento, hemos aprendido a establecer una condición con "where" utilizando operadores 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 "y/o",
- xor, significa "o",
- not, significa "no", invierte el resultado
- (), paréntesis

Los operadores lógicos se usan para combinar condiciones.

Queremos recuperar todos los registros cuyo autor sea igual a "Borges" y cuyo precio no supere 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';
```

En la sentencia anterior usamos el operador "or", indicamos que recupere los libros en los cuales el valor del campo "autor" sea "Borges" y/o el valor del campo "editorial" sea "Planeta", es decir, seleccionará los registros que cumplan con la primera condición, con la segunda condición o con ambas condiciones.

Los registros recuperados con una sentencia que une 2 condiciones con el operador "or", cumplen 1 de las condiciones o ambas.

Queremos ver los libros cuyo autor sea "Borges" o cuya editorial sea "Planeta":

```
select * from libros
where (autor='Borges') xor
(editorial='Planeta');
```

## MySQL\_ Operador y función.

En la sentencia anterior usamos el operador "xor", indicamos que recupere los libros en los cuales el valor del campo "autor" sea "Borges" o el valor del campo "editorial" sea "Planeta", es decir, seleccionará los registros que cumplan con la primera condición o con la segunda condición pero no los que cumplan con ambas condiciones. Los registros recuperados con una sentencia que une 2 condiciones con el operador "xor", cumplen 1 de las condiciones, no ambas.

Queremos recuperar los libros que no cumplan la condición dada, por ejemplo, aquellos cuya editorial NO sea "Planeta":

```
select * from libros
where not (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", no cumplen con la condición a la cual afecta el "NO".

Los paréntesis se usan para encerrar condiciones, para que se evalúen como una sola expresión.

Cuando explicitamos varias condiciones con diferentes operadores lógicos (combinamos "and", "or") permite establecer el orden de prioridad de la evaluación; además permite diferenciar las expresiones más claramente.

Por ejemplo, las siguientes expresiones devuelven un resultado diferente:

```
select * from libros
where (autor='Borges') or
(editorial='Paidos' and precio<20);

select*from libros
where (autor='Borges' or editorial='Paidos') and
(precio<20);
```

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

El orden de prioridad de los operadores lógicos es el siguiente: "not" se aplica antes que "and" y "and" antes que "or", si no se especifica un orden de evaluación mediante el uso de paréntesis.

El orden en el que se evalúan los operadores con igual nivel de precedencia es indefinido, por ello se recomienda usar los paréntesis.

### Servidor de MySQL instalado en forma local.

Ingresemos al programa "Workbench" y ejecutemos el siguiente bloque de instrucciones SQL donde probamos los operadores lógicos and, or, not y xor:

```
drop table if exists libros;

create table libros(
codigo int unsigned auto_increment,
titulo varchar(40),
autor varchar(30),
```

## MySQL\_ Operador y función.

```
editorial varchar(15),
precio decimal(5,2),
primary key(codigo)
);

insert into libros (titulo,autor,editorial,precio)
values('El aleph','Borges','Planeta',15.50);
insert into libros (titulo,autor,editorial,precio)
values('Martin Fierro','Jose Hernandez','Emece',22.90);
insert into libros (titulo,autor,editorial,precio)
values('Martin Fierro','Jose Hernandez','Planeta',39);
insert into libros (titulo,autor,editorial,precio)
values('Aprenda PHP','Mario Molina','Emece',19.50);
insert into libros (titulo,autor,editorial,precio)
values('Cervantes y el quijote','Borges','Paidos',35.40);
insert into libros (titulo,autor,editorial,precio)
values('Matematica estas ahi', 'Paenza', 'Paidos',19);

select * from libros
where autor='Borges' and
precio<=20;

select * from libros
where autor='Paenza' or
editorial='Planeta';

select * from libros
where (autor='Borges') xor
(editorial='Planeta');

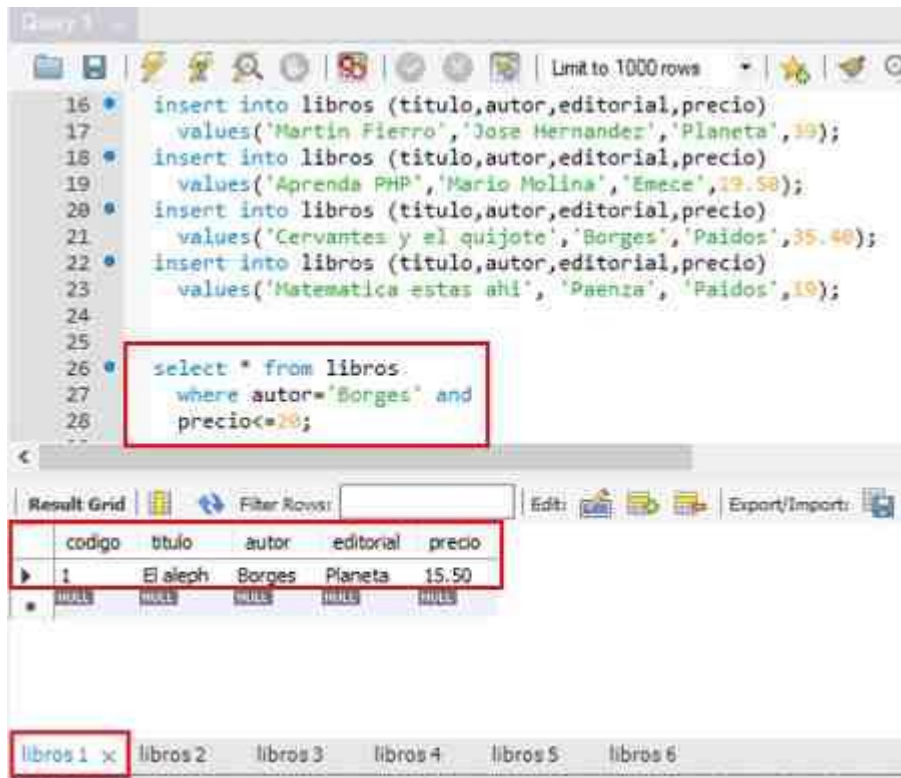
select * from libros
where not (editorial='Planeta');

select * from libros
where (autor='Borges') or
(editorial='Paidos' and precio<20);

select * from libros
where (autor='Borges' or editorial='Paidos')
and (precio<20);
```

## MySQL\_ Operador y función.

Que nos genera una salida similar a esta:



The screenshot shows a MySQL query editor with a toolbar at the top. The SQL editor contains several INSERT statements followed by a SELECT statement. The SELECT statement is highlighted with a red box. Below the editor, the 'Result Grid' shows the results of the SELECT query, also highlighted with a red box. The results table has columns: codigo, titulo, autor, editorial, and precio. The first row shows a book with codigo 1, titulo 'El aleph', autor 'Borges', editorial 'Planeta', and precio 15.50. At the bottom, there are tabs for 'libros 1' through 'libros 6', with 'libros 1' selected and highlighted with a red box.

```
16 • insert into libros (titulo,autor,editorial,precio)
17   values('Martin Fierro','Jose Hernandez','Planeta',39);
18 • insert into libros (titulo,autor,editorial,precio)
19   values('Aprenda PHP','Mario Molina','Emece',29.58);
20 • insert into libros (titulo,autor,editorial,precio)
21   values('Cervantes y el quijote','Borges','Paidós',35.40);
22 • insert into libros (titulo,autor,editorial,precio)
23   values('Matematica estas ahi','Paenza','Paidós',19);
24
25
26 • select * from libros
27   where autor='Borges' and
28   precio<=20;
```

|   | codigo | titulo   | autor  | editorial | precio |
|---|--------|----------|--------|-----------|--------|
| ▶ | 1      | El aleph | Borges | Planeta   | 15.50  |
| • | NULL   | NULL     | NULL   | NULL      | NULL   |

libros 1 x libros 2 libros 3 libros 4 libros 5 libros 6

# MySQL\_ Operador y función.

## 8. Funciones de control de flujo (if)

Trabajamos con las tablas "libros" de una librería.

No nos interesa el precio exacto de cada libro, sino si el precio es menor o mayor a 50 €. Podemos utilizar estas sentencias:

```
select titulo from libros
where precio<50;
```

```
select titulo from libros
where precio >=50;
```

En la primera sentencia mostramos los libros con precio menor a 50 y en la segunda los demás.

También podemos usar la función "if".

"if" es una función a la cual se le envían 3 argumentos: el segundo y tercer argumento corresponden a los valores que retornará en caso que el primer argumento (una expresión de comparación) sea "verdadero" o "falso"; es decir, si el primer argumento es verdadero, retorna el segundo argumento, sino retorna el tercero.

Veamos el ejemplo:

```
select titulo,
if (precio>50,'caro','economico')
from libros;
```

Si el precio del libro es mayor a 50 (primer argumento del "if"), coloca "caro" (segundo argumento del "if"), en caso contrario coloca "economico" (tercer argumento del "if").

Veamos otros ejemplos.

Queremos mostrar los nombres de los autores y la cantidad de libros de cada uno de ellos; para ello especificamos el nombre del campo a mostrar ("autor"), contamos los libros con "autor" conocido con la función "count()" y agrupamos por nombre de autor:

```
select autor, count(*)
from libros
group by autor;
```

El resultado nos muestra cada autor y la cantidad de libros de cada uno de ellos. Si solamente queremos mostrar los autores que tienen más de 1 libro, es decir, la cantidad mayor a 1, podemos usar esta sentencia:

```
select autor, count(*)
from libros
group by autor
having count(*)>1;
```



## MySQL\_ Operador y función.

Pero si no queremos la cantidad exacta sino solamente saber si cada autor tiene más de 1 libro, podemos usar "if":

```
select autor,  
if (count(*)>1,'Más de 1','1')  
from libros  
group by autor;
```

Si la cantidad de libros de cada autor es mayor a 1 (primer argumento del "if"), coloca "Más de 1" (segundo argumento del "if"), en caso contrario coloca "1" (tercer argumento del "if").

Queremos saber si la cantidad de libros por editorial supera los 4 o no:

```
select editorial,  
if (count(*)>4,'5 o más','menos de 5') as cantidad  
from libros  
group by editorial  
order by cantidad;
```

Si la cantidad de libros de cada editorial es mayor a 4 (primer argumento del "if"), coloca "5 o más" (segundo argumento del "if"), en caso contrario coloca "menos de 5" (tercer argumento del "if").

### Servidor de MySQL instalado en forma local.

Ingresemos al programa "Workbench" y ejecutemos el siguiente bloque de instrucciones SQL:

```
drop table if exists libros;  
  
create table libros(  
codigo int unsigned auto_increment,  
titulo varchar(40) not null,  
autor varchar(30),  
editorial varchar(30),  
precio decimal(5,2) unsigned,  
primary key (codigo)  
);  
  
insert into libros (titulo, autor, editorial, precio)  
values('Alicia en el pais de las maravillas', 'Lewis Carroll', 'Paidos', 50.5);  
insert into libros (titulo, autor, editorial, precio)  
values('Alicia a traves del espejo', 'Lewis Carroll', 'Emece', 25);  
insert into libros (titulo, autor, editorial, precio)  
values('El aleph', 'Borges', 'Paidos', 15);  
insert into libros (titulo, autor, editorial, precio)  
values('Matemática estas ahí', 'Paenza', 'Paidos', 10);  
insert into libros (titulo, autor, editorial)  
values('Antología', 'Borges', 'Paidos');  
insert into libros (titulo, editorial)  
values('El gato con botas', 'Paidos');  
insert into libros (titulo, autor, editorial, precio)
```

## MySQL\_ Operador y función.

```
values('Martin Fierro','Jose Hernandez','Emece',90);

select titulo from libros
where precio<50;

select titulo from libros
where precio >=50;

select titulo,
if(precio>50,'caro','economico')
from libros;

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

select autor, count(*)
from libros
group by autor
having count(*)>1;

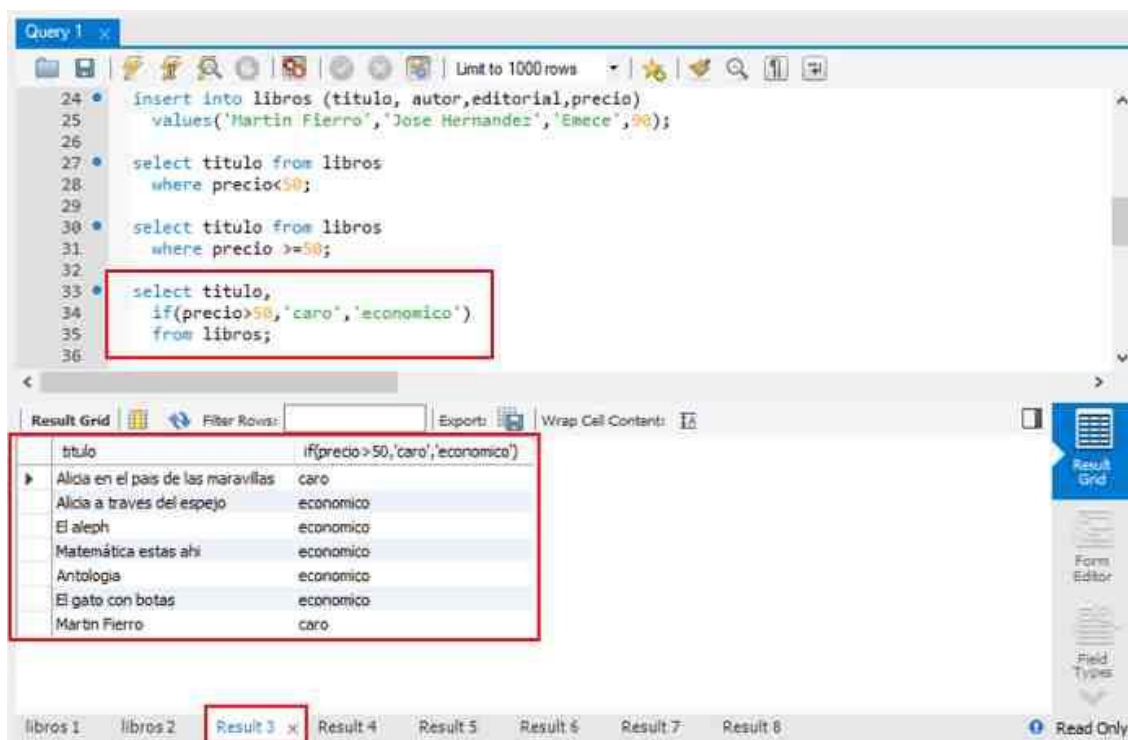
-- mostrar cada autor y un mensaje si tiene 1 o más de un libro
select autor,
if (count(*)>1,'Más de 1','1')
from libros
group by autor;

select autor,
if (count(*)>1,'Más de 1','1') as cantidad
from libros
group by autor
order by cantidad;

select editorial,
if (count(*)>4,'5 o más','menos de 5') as cantidad
from libros
group by editorial
order by cantidad;
```

# MySQL\_ Operador y función.

Genera una salida similar a esta:



The screenshot shows the MySQL Workbench interface. The top pane displays a SQL query with four statements. The third statement, which uses an IF function, is highlighted with a red box. The bottom pane shows the 'Result Grid' for 'Result 3', also highlighted with a red box. The grid contains two columns: 'titulo' and 'if(precio > 50, 'caro', 'economico')'. The data rows are as follows:

| titulo                              | if(precio > 50, 'caro', 'economico') |
|-------------------------------------|--------------------------------------|
| Alicia en el país de las maravillas | caro                                 |
| Alicia a través del espejo          | economico                            |
| El aleph                            | economico                            |
| Matemática estas ahí                | economico                            |
| Antología                           | economico                            |
| El gato con botas                   | economico                            |
| Martin Fierro                       | caro                                 |

# MySQL\_ Operador y función.

## 9. Funciones de control de flujo (case)

La función "case" es similar a la función "if", sólo que se pueden establecer varias condiciones a cumplir.

Trabajemos con la tabla "libros" de una librería.

Queremos saber si la cantidad de libros de cada editorial es menor o mayor a 1, tipeamos:

```
select editorial,  
if (count(*)>1,'Mas de 2','1') as 'cantidad'  
from libros  
group by editorial;
```

vemos los nombres de las editoriales y una columna "cantidad" que especifica si hay más o menos de uno. Podemos obtener la misma salida usando un "case":

```
select editorial,  
case count(*)  
  when 1 then 1  
  else 'mas de 1' end as 'cantidad'  
from libros  
group by editorial;
```

Por cada valor hay un "when" y un "then"; si encuentra un valor coincidente en algún "where" ejecuta el "then" correspondiente a ese "where", si no encuentra ninguna coincidencia, se ejecuta el "else", si no hay parte "else" retorna "null". Finalmente se coloca "end" para indicar que el "case" ha finalizado.

Entonces, la sintaxis es:

```
case  
  when then  
  ...  
  else end
```

Se puede obviar la parte "else":

```
select editorial,  
case count(*)  
  when 1 then 1  
  end as 'cantidad'  
from libros  
group by editorial;
```

## MySQL\_ Operador y función.

Con el "if" solamente podemos obtener dos salidas, cuando la condición resulta verdadera y cuando es falsa, si queremos más opciones podemos usar "case". Vamos a extender el "case" anterior para mostrar distintos mensajes:

```
select editorial,
case count(*)
  when 1 then 1
  when 2 then 2
  when 3 then 3
  else 'Más de 3' end as 'cantidad'
from libros
group by editorial;
```

Incluso podemos agregar una cláusula "order by" y ordenar la salida por la columna "cantidad":

```
select editorial,
case count(*)
  when 1 then 1
  when 2 then 2
  when 3 then 3
  else 'Más de 3' end as 'cantidad'
from libros
group by editorial
order by cantidad;
```

La diferencia con "if" es que el "case" toma valores puntuales, no expresiones. La siguiente sentencia provocará un error:

```
select editorial,
case count(*)
  when 1 then 1
  when >1 then 'mas de 1'
end as 'cantidad'
from libros
group by editorial;
```

Pero existe otra sintaxis de "case" que permite condiciones:

```
case
when then
...
else
end
```

## MySQL\_ Operador y función.

Veamos un ejemplo:

```
select editorial,
case
  when count(*)=1 then 1
  else 'mas de uno'
end as cantidad
from libros
group by editorial;
```

### Servidor de MySQL instalado en forma local.

Ingresemos al programa "Workbench" y ejecutemos el siguiente bloque de instrucciones SQL:

```
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,
  cantidad smallint unsigned,
  primary key(codigo)
);

insert into libros (titulo,autor,editorial,precio,cantidad)
values('El aleph','Borges','Planeta',34.5,100);
insert into libros (titulo,autor,editorial,precio,cantidad)
values('Alicia en el pais de las maravillas','Carroll L.','Paidos',20.7,50);
insert into libros (titulo,autor,editorial,precio,cantidad)
values('harry Potter y la camara secreta',null,'Emece',35,500);
insert into libros (titulo,autor,editorial,precio,cantidad)
values('Aprenda PHP','Molina Mario','Planeta',54,100);
insert into libros (titulo,autor,editorial,precio,cantidad)
values('Harry Potter y la piedra filosofal',null,'Emece',38,500);
insert into libros (titulo,autor,editorial,precio,cantidad)
values('Aprenda Java','Molina Mario','Planeta',55,100);
insert into libros (titulo,autor,editorial,precio,cantidad)
values('Aprenda JavaScript','Molina Mario','Planeta',58,150);

select editorial,
case count(*)
  when 1 then 1
  else 'mas de 1' end as 'cantidad'
from libros
group by editorial;

select editorial,
```

## MySQL\_ Operador y función.

```
case count(*)
  when 1 then 1
  end as 'cantidad'
from libros
group by editorial;
```

```
select editorial,
  case count(*)
    when 1 then 1
    when 2 then 2
    when 3 then 3
  else 'Más de 3' end as 'cantidad'
from libros
group by editorial;
```

```
select editorial,
  case count(*)
    when 1 then 1
    when 2 then 2
    when 3 then 3
  else 'Más de 3' end as 'cantidad'
from libros
group by editorial
order by cantidad;
```

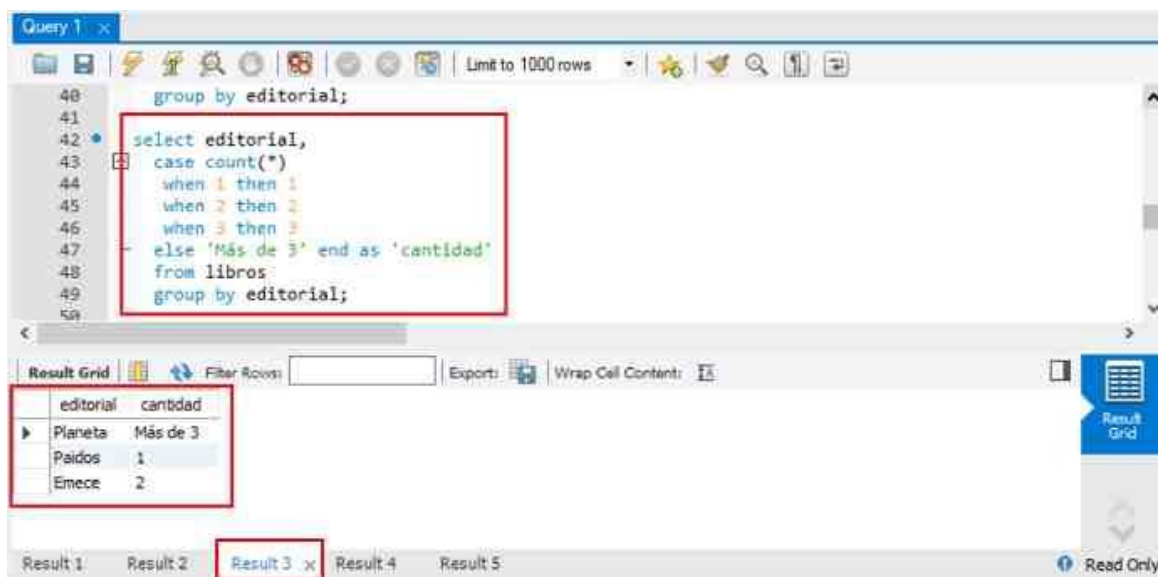
-- la estructura del case es incorrecto

```
select editorial,
  case count(*)
    when 1 then 1
    when >1 then 'mas de 1'
  end as 'cantidad'
from libros
group by editorial;
```

```
select editorial,
  case when count(*)=1 then 1
    else 'mas de 1'
  end as 'cantidad'
from libros
group by editorial;
```

## MySQL\_ Operador y función.

Genera una salida similar a esta:



The screenshot shows the MySQL Workbench interface. The top pane displays a SQL query for 'Query 1'. The query is as follows:

```
group by editorial;

select editorial,
case count(*)
when 1 then 1
when 2 then 2
when 3 then 3
else 'Más de 3' end as 'cantidad'
from libros
group by editorial;
```

The bottom pane shows the 'Result Grid' for 'Result 3'. The grid contains the following data:

| editorial | cantidad |
|-----------|----------|
| Planeta   | Más de 3 |
| Paidós    | 1        |
| Emece     | 2        |



# MySQL\_ Operador y función.

## 10. Búsqueda de patrones (like y not 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 cadenas completas.

Imaginemos que tenemos registrados estos 2 libros:

```
El Aleph de Borges;
Antología poetica de J.L. Borges;
```

Si queremos recuperar todos los libros cuyo autor sea "Borges", y especificamos la siguiente 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) comparan cadenas 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 ningún carácter). Es un carácter 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 tener como 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%';
```

## MySQL\_ Operador y función.

Así como "%" reemplaza cualquier cantidad de caracteres, el guión bajo "\_" reemplaza un caracter, 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 "%\".

### Servidor de MySQL instalado en forma local.

Ingresemos al programa "Workbench" y ejecutemos el siguiente bloque de instrucciones SQL:

```
drop table if exists libros;

create table libros(
  codigo int unsigned auto_increment,
  titulo varchar(40),
  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',15.50);
insert into libros (titulo,autor,editorial,precio)
values('Martin Fierro','Jose Hernandez','Emece',22.90);
insert into libros (titulo,autor,editorial,precio)
values('Antologia poetica','J.L. Borges','Planeta',39);
insert into libros (titulo,autor,editorial,precio)
values('Aprenda PHP','Mario Molina','Emece',19.50);
insert into libros (titulo,autor,editorial,precio)
values('Cervantes y el quijote','Bioy Casare- J.L. Borges','Paidos',35.40);
insert into libros (titulo,autor,editorial,precio)
values('Manual de PHP','J.C. Paez','Paidos',19);
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 camara secreta','J.K. Rowling','Paidos',46.00);
insert into libros (titulo,autor,editorial,precio)
```

## MySQL\_ Operador y función.

```
values('Alicia en el pais de las maravillas','Lewis Carroll','Paidos',36.00);
```

```
select * from libros
where autor='Borges';
```

```
select * from libros
where autor like '%Borges%';
```

```
select * from libros
where titulo like 'A%';
```

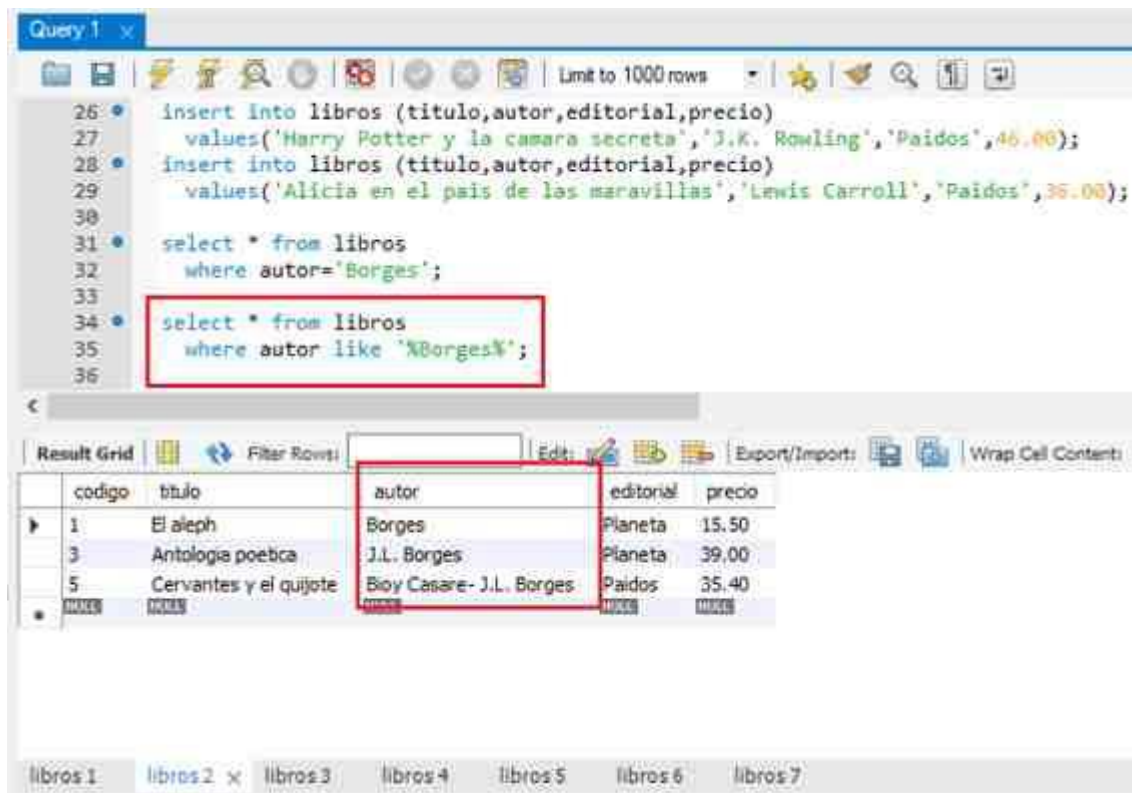
```
select * from libros
where titulo not like 'A%';
```

```
select * from libros
where autor like '%Carrol_';
```

```
select * from libros
where titulo like '%Harry Potter%';
```

```
select * from libros
where titulo like '%PHP%';
```

Que nos genera una salida similar a esta:



The screenshot shows a MySQL query editor window titled 'Query 1'. The SQL code is as follows:

```
26 • insert into libros (titulo,autor,editorial,precio)
27     values('Harry Potter y la camara secreta','J.K. Rowling','Paidos',46.00);
28 • insert into libros (titulo,autor,editorial,precio)
29     values('Alicia en el pais de las maravillas','Lewis Carroll','Paidos',36.00);
30
31 • select * from libros
32     where autor='Borges';
33
34 • select * from libros
35     where autor like '%Borges%';
36
```

The results are displayed in a table with the following columns: codigo, titulo, autor, editorial, and precio. The table contains three rows of data:

| codigo | titulo                 | autor                    | editorial | precio |
|--------|------------------------|--------------------------|-----------|--------|
| 1      | El aleph               | Borges                   | Planeta   | 15.50  |
| 3      | Antologia poetica      | J.L. Borges              | Planeta   | 39.00  |
| 5      | Cervantes y el quijote | Bioy Casare- J.L. Borges | Paidos    | 35.40  |

# MySQL\_ Operador y función.

## 11. Búsqueda de patrones (regexp)

Los operadores "regexp" y "not regexp" busca patrones de modo similar a "like" y "not like".

Para buscar libros que contengan la cadena "Ma" usamos:

```
select titulo from libros
where titulo regexp 'Ma';
```

Para buscar los autores que tienen al menos una "h" o una "k" o una "w" tipeamos:

```
select autor from libros
where autor regexp '[hkw]';
```

Para buscar los autores que no tienen ni "h" o una "k" o una "w" tipeamos:

```
select autor from libros
where autor not regexp '[hkw]';
```

Para buscar los autores que tienen por lo menos una de las letras de la "a" hasta la "d", es decir, "a,b,c,d", usamos:

```
select autor from libros
where autor regexp '[a-d]';
```

Para ver los títulos que comienzan con "A" tipeamos:

```
select titulo from libros
where titulo regexp '^A';
```

Para ver los títulos que terminan en "HP" usamos:

```
select titulo from libros
where titulo regexp 'HP$';
```

Para buscar títulos que contengan una "a" luego un caracter cualquiera y luego una "e" utilizamos la siguiente sentencia:

```
select titulo from libros
where titulo regexp 'a.e';
```

El punto (.) identifica cualquier caracter.

Podemos mostrar los títulos que contienen una "a" seguida de 2 caracteres y luego una "e":

```
select titulo from libros
where titulo regexp 'a..e';
```

Para buscar autores que tengan 6 caracteres exactamente usamos:

```
select autor from libros
where autor regexp '^.....$';
```

Para buscar autores que tengan al menos 6 caracteres usamos:

## MySQL\_ Operador y función.

```
select autor from libros
where autor regexp '.....';
```

Para buscar títulos que contengan 2 letras "a" usamos:

```
select titulo from libros
where titulo regexp 'a.*a';
```

El asterisco indica que busque el caracter inmediatamente anterior, en este caso cualquiera porque hay un punto.

### Servidor de MySQL instalado en forma local.

Ingresemos al programa "Workbench" y ejecutemos el siguiente bloque de instrucciones SQL para utilizar patrones de búsqueda mediante "regexp":

```
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,
  primary key(codigo)
);

insert into libros (titulo,autor,editorial,precio)
values('El aleph','Borges','Planeta',15.50);
insert into libros (titulo,autor,editorial,precio)
values('Martin Fierro','Jose Hernandez','Emece',22.90);
insert into libros (titulo,autor,editorial,precio)
values('Antologia poetica','J.L. Borges','Planeta',39);
insert into libros (titulo,autor,editorial,precio)
values('Aprenda PHP','Mario Molina','Emece',19.50);
insert into libros (titulo,autor,editorial,precio)
values('Cervantes y el quijote','Bioy Casare- J.L. Borges','Paidos',35.40);
insert into libros (titulo,autor,editorial,precio)
values('Manual de PHP','J.C. Paez','Paidos',19);
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 camara secreta','J.K. Rowling','Paidos',46.00);
insert into libros (titulo,autor,editorial,precio)
values('Alicia en el pais de las maravillas','Lewis Carroll','Paidos',36.00);

-- Para buscar libros cuyos títulos contengan la cadena "Ma" usamos:
select titulo from libros
where titulo regexp 'Ma';
```

## MySQL\_ Operador y función.

```
-- Para buscar los registros cuyos autores tienen al menos una "h" o una "k" o una "w"
tipeamos:
select titulo,autor from libros
  where autor regexp '[hkw]';

-- Para buscar los libros cuyos autores no tienen ni "h" o una "k" o una "w" tipeamos:
select titulo,autor from libros
  where autor not regexp '[hkw]';

-- Para buscar los autores que tienen por lo menos una de las letras de la "a"
-- hasta la "d", es decir, "a,b,c,d", usamos:
select autor from libros
  where autor regexp '[a-d]';

-- Para ver los títulos que comienzan con "A" tipeamos:
select titulo from libros
  where titulo regexp '^A';

-- Para ver los títulos que terminan en "HP" usamos:
select titulo from libros
  where titulo regexp 'HP$';

-- Para buscar títulos que contengan una "a" luego un caracter cualquiera y luego una "e":
select titulo from libros
  where titulo regexp 'a.e';

-- Podemos mostrar los títulos que contienen una "a" seguida de 2 caracteres y luego una "e":
select titulo from libros
  where titulo regexp 'a..e';

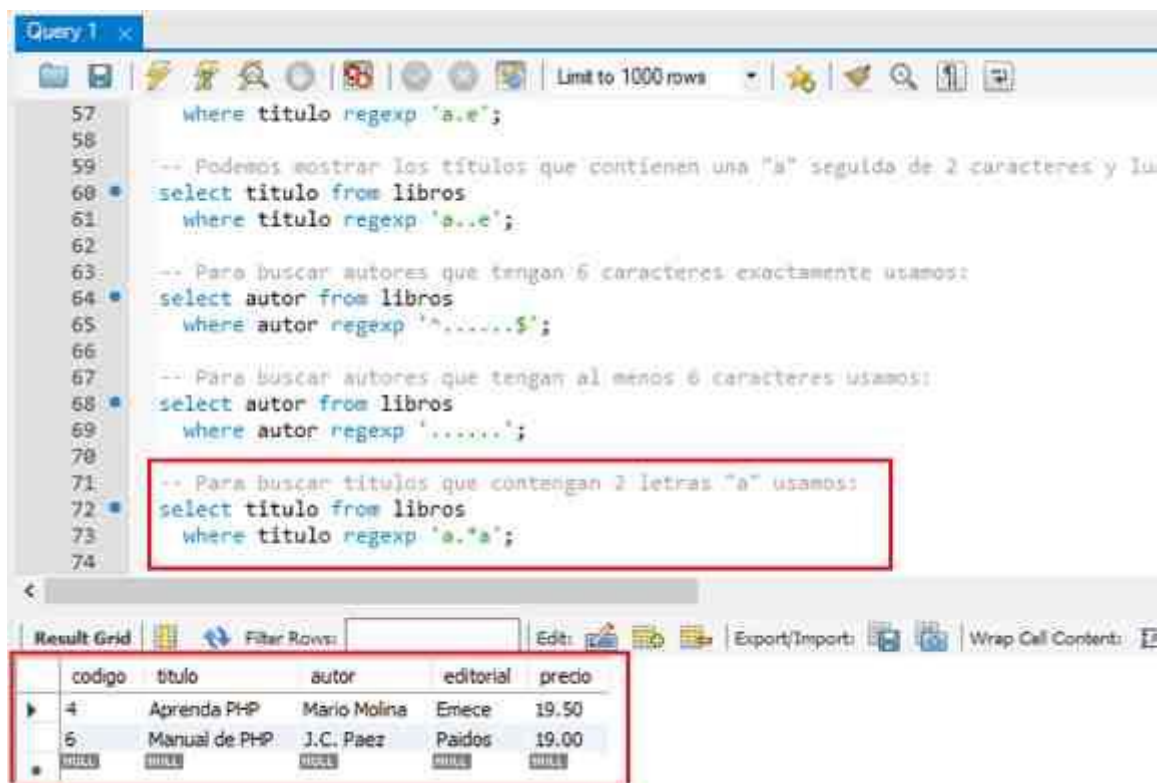
-- Para buscar autores que tengan 6 caracteres exactamente usamos:
select autor from libros
  where autor regexp '^.....$';

-- Para buscar autores que tengan al menos 6 caracteres usamos:
select autor from libros
  where autor regexp '.....';

-- Para buscar títulos que contengan 2 letras "a" usamos:
select titulo from libros
  where titulo regexp 'a.*a';
```

## MySQL\_ Operador y función.

Que nos genera una salida similar a esta:



The screenshot shows a MySQL query editor window titled "Query 1". The SQL code is as follows:

```
57 where titulo regexp 'a.e';
58
59 -- Podemos mostrar los títulos que contienen una "a" seguida de 2 caracteres y la
60 select titulo from libros
61 where titulo regexp 'a..e';
62
63 -- Para buscar autores que tengan 6 caracteres exactamente usamos:
64 select autor from libros
65 where autor regexp '^.....$';
66
67 -- Para buscar autores que tengan al menos 6 caracteres usamos:
68 select autor from libros
69 where autor regexp '.....';
70
71 -- Para buscar títulos que contengan 2 letras "a" usamos:
72 select titulo from libros
73 where titulo regexp 'a.*a';
74
```

The results are displayed in a table with the following columns: codigo, titulo, autor, editorial, and precio. The table contains two rows of data:

| codigo | titulo        | autor        | editorial | precio |
|--------|---------------|--------------|-----------|--------|
| 4      | Aprenda PHP   | Mario Molina | Emece     | 19.50  |
| 5      | Manual de PHP | J.C. Paez    | Paidós    | 19.00  |

# MySQL\_ Operador y función.

## 12. Contar registros (count)

Existen en MySQL funciones que nos permiten contar registros, calcular sumas, promedios, obtener valores máximos y mínimos. Veamos algunas de ellas.

Imaginemos que nuestra tabla "libros" contiene muchos registros. Para averiguar la cantidad sin necesidad de contarlos manualmente usamos la función "count()":

```
select count(*) from libros;
```

La función "count()" cuenta la cantidad de registros de una tabla, incluyendo los que tienen valor nulo.

Para saber la cantidad de libros de la editorial "Planeta" tipeamos:

```
select count(*) from libros  
where editorial='Planeta';
```

También podemos utilizar esta función junto con la cláusula "where" para una consulta más específica. Por ejemplo, solicitamos la cantidad de libros que contienen la cadena "Borges":

```
select count(*) from libros  
where autor like '%Borges%';
```

Para contar los registros que tienen precio (sin tener en cuenta los que tienen valor nulo), usamos la función "count()" y en los paréntesis colocamos el nombre del campo que necesitamos contar:

```
select count(precio) from libros;
```

Note que "count(\*)" retorna la cantidad de registros de una tabla (incluyendo los que tienen valor "null") mientras que "count(precio)" retorna la cantidad de registros en los cuales el campo "precio" no es nulo. No es lo mismo. "count(\*)" cuenta registros, si en lugar de un asterisco colocamos como argumento el nombre de un campo, se contabilizan los registros cuyo valor en ese campo no es nulo.

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.

### Servidor de MySQL instalado en forma local.

Ingresemos al programa "Workbench" y ejecutemos el siguiente bloque de instrucciones SQL para probar la función 'count':



## MySQL\_ Operador y función.

```
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 mediumint 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 poetica','J.L. 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','Bioy Casares- J.L. Borges','Paidos',36.40,100);
insert into libros (titulo,autor,editorial,precio,cantidad)
  values('Manual de PHP', 'J.C. Paez', 'Paidos',30.80,120);
insert into libros (titulo,autor,editorial,precio,cantidad)
  values('Harry Potter y la piedra filosofal','J.K. Rowling','Paidos',45.00,50);
insert into libros (titulo,autor,editorial,precio,cantidad)
  values('Harry Potter y la camara secreta','J.K. Rowling','Paidos',46.00,100);
insert into libros (titulo,autor,editorial,precio,cantidad)
  values('Alicia en el pais de las maravillas','Lewis Carroll','Paidos',null,200);

select count(*) from libros;

select count(*) from libros
  where editorial='Planeta';

select count(*) from libros where autor like '%Borges%';

select count(precio) from libros;
```

Que nos genera una salida similar a esta:

## MySQL\_ Operador y función.

Query 1 x

```
1 drop table if exists visitantes;
2
3 create table visitantes(
4     nombre varchar(30),
5     edad integer unsigned,
6     sexo char(1),
7     domicilio varchar(30),
8     ciudad varchar(20),
9     telefono varchar(11),
10    montocompra float unsigned
11 );
12
13 describe visitantes;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

|   | Field       | Type             | Null | Key | Default | Extra |
|---|-------------|------------------|------|-----|---------|-------|
| ► | nombre      | varchar(30)      | YES  |     | NULL    |       |
|   | edad        | int(10) unsigned | YES  |     | NULL    |       |
|   | sexo        | char(1)          | YES  |     | NULL    |       |
|   | domicilio   | varchar(30)      | YES  |     | NULL    |       |
|   | ciudad      | varchar(20)      | YES  |     | NULL    |       |
|   | telefono    | varchar(11)      | YES  |     | NULL    |       |
|   | montocompra | float unsigned   | YES  |     | NULL    |       |

Result 7 x

## MySQL\_ Operador y función.

### 13. Funciones de agrupamiento

#### (count - max - min - sum - avg)

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. Por ejemplo, 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 libros tenemos 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 los libros:

```
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. Por ejemplo, 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 conjuntos de 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.

#### Servidor de MySQL instalado en forma local.

Ingresemos al programa "Workbench" y ejecutemos el siguiente bloque de instrucciones SQL para probar varias funciones de agrupamiento:

## MySQL\_ Operador y función.

```
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 mediumint 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 poetica','J.L. 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','Bioy Casares- J.L. Borges','Paidos',36.40,100);
insert into libros (titulo,autor,editorial,precio,cantidad)
  values('Manual de PHP', 'J.C. Paez', 'Paidos',30.80,120);
insert into libros (titulo,autor,editorial,precio,cantidad)
  values('Harry Potter y la piedra filosofal','J.K. Rowling','Paidos',45.00,50);
insert into libros (titulo,autor,editorial,precio,cantidad)
  values('Harry Potter y la camara secreta','J.K. Rowling','Paidos',46.00,100);
insert into libros (titulo,autor,editorial,precio,cantidad)
  values('Alicia en el pais de las maravillas','Lewis Carroll','Paidos',null,200);

select sum(cantidad) from libros;

select sum(cantidad) from libros
  where editorial ='Planeta';

select max(precio) from libros;

select * from libros
  order by precio desc;

select min(precio) from libros
  where autor like '%Rowling%';

select * from libros
  where autor like '%Rowling%'
  order by 5;
```

## MySQL\_ Operador y función.

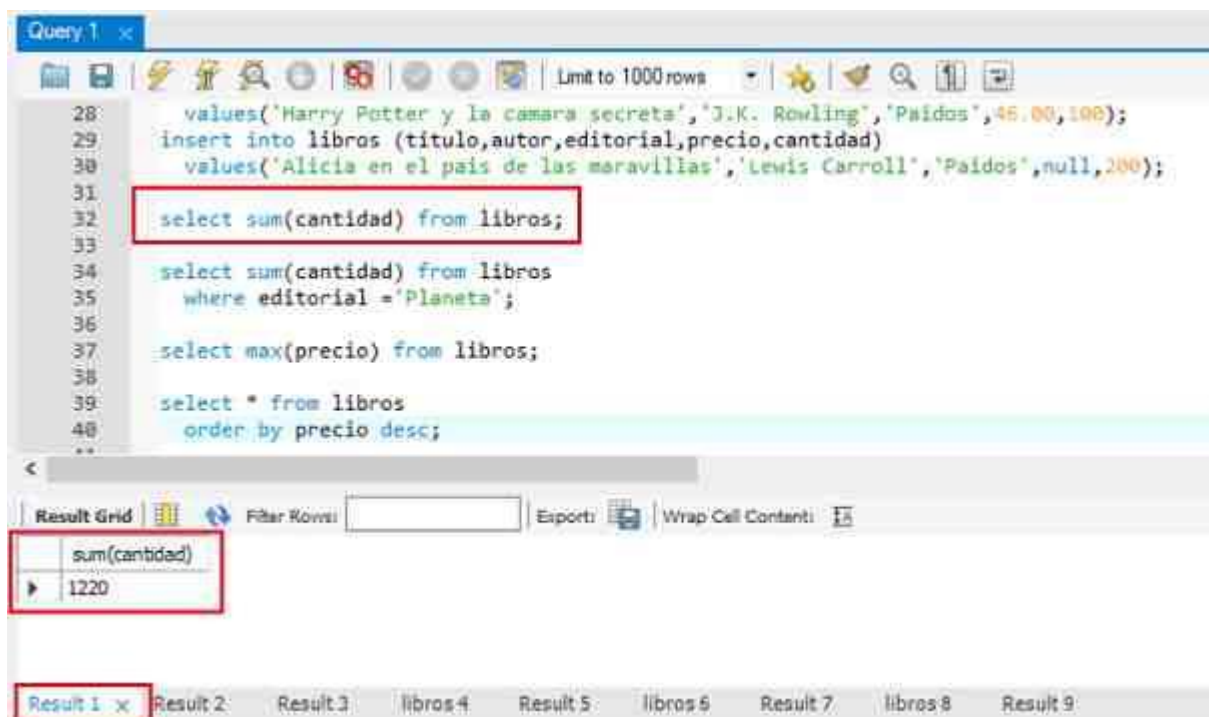
```
select avg(precio) from libros  
where titulo like '%PHP%';
```

```
select * from libros  
where titulo like '%PHP%';
```

```
select count(*) from libros;
```

```
select count (*) from libros;
```

Que nos genera una salida similar a esta:



The screenshot shows a MySQL query editor window titled "Query 1". The query is as follows:

```
28: values('Harry Potter y la cámara secreta','J.K. Rowling','Paidós',46.00,100);  
29: insert into libros (titulo,autor,editorial,precio,cantidad)  
30: values('Alicia en el país de las maravillas','Lewis Carroll','Paidós',null,200);  
31:   
32: select sum(cantidad) from libros;  
33:   
34: select sum(cantidad) from libros  
35: where editorial = 'Planeta';  
36:   
37: select max(precio) from libros;  
38:   
39: select * from libros  
40: order by precio desc;  
41: **
```

The query is executed, and the result is displayed in the "Result Grid" tab. The result is a single row with the value 1220.

| sum(cantidad) |
|---------------|
| 1220          |

The "Result Grid" tab is selected, and the result is displayed. The "Result 1" tab is also visible in the bottom tab bar.

# MySQL\_ Operador y función.

## 14. Agrupar registros (group by)

Hemos aprendido que las funciones de agrupamiento permiten contar registros, calcular sumas y promedios, obtener valores máximos y mínimos. También dijimos que dichas funciones operan sobre conjuntos de registros, no con datos individuales.

Generalmente estas funciones se combinan con la sentencia "group by", que agrupa registros para consultas detalladas.

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

```
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 el agrupamiento, 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 de las ciudades y la cantidad de registros para cada valor del campo.

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

## MySQL\_ Operador y función.

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

Se pueden simplificar las 2 sentencias anteriores en una sola sentencia, ya que usan el mismo "group by":

```
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;
```

Podemos usar las palabras claves "asc" y "desc" para una salida ordenada:

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

# MySQL\_ Operador y función.

## Servidor de MySQL instalado en forma local.

Ingrese al programa "Workbench" y ejecutemos el siguiente bloque de instrucciones SQL para analizar la cláusula 'group by':

```
drop table if exists visitantes;

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;
```



## MySQL\_ Operador y función.

```
-- 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 visitantes
group by sexo;

-- Se pueden simplificar las 2 sentencias anteriores en una sola sentencia, ya que usan el mismo
"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;

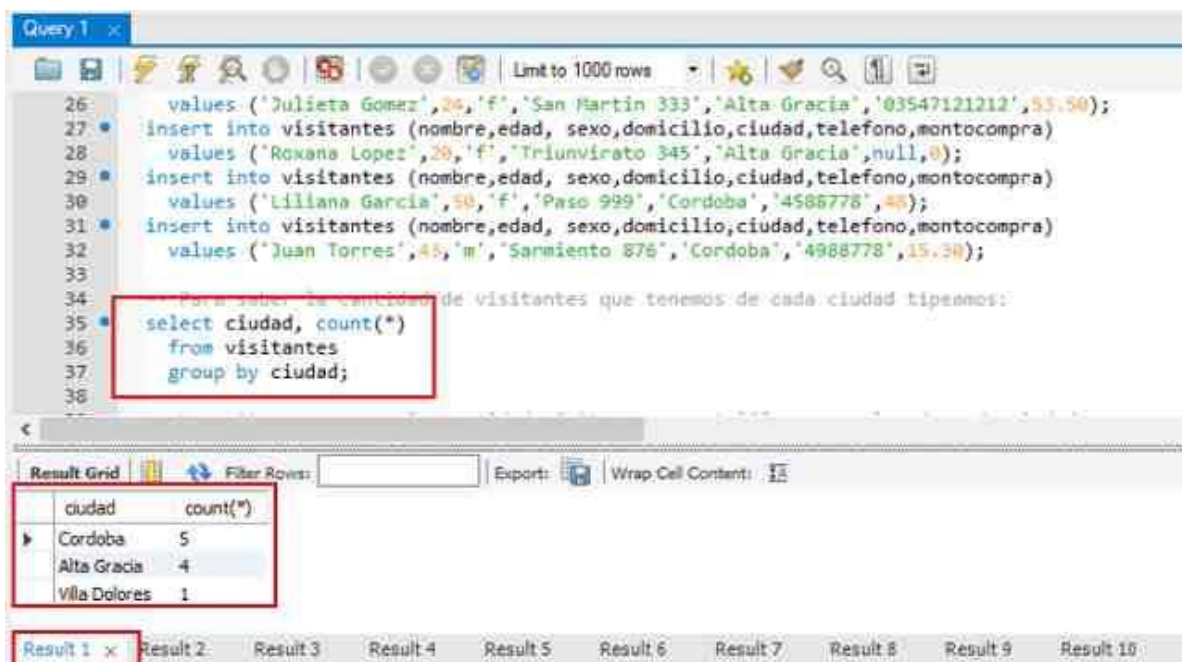
-- Contamos los registros y agrupamos por 2 campos, "ciudad" y "sexo":
select ciudad, sexo, count(*) from visitantes
group by ciudad,sexo;

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

-- Usando la palabra clave "desc" obtenemos la salida ordenada en forma descendente:
select ciudad, count(*) from visitantes
group by ciudad desc;
```

## MySQL\_ Operador y función.

Que nos genera una salida similar a esta:



The screenshot shows a MySQL query editor window titled "Query 1". The SQL code is as follows:

```
26 values ('Julietta Gomez',24,'f','San Martin 333','Alta Gracia','03547121212',55.50);
27 insert into visitantes (nombre,edad, sexo,domicilio,ciudad,telefono,montocompra)
28 values ('Roxana Lopez',28,'f','Triunvirato 345','Alta Gracia',null,0);
29 insert into visitantes (nombre,edad, sexo,domicilio,ciudad,telefono,montocompra)
30 values ('Lilliana Garcia',50,'f','Paso 999','Cordoba','4586778',48);
31 insert into visitantes (nombre,edad, sexo,domicilio,ciudad,telefono,montocompra)
32 values ('Juan Torres',45,'m','Sarmiento 876','Cordoba','4988778',15.30);
33
34 Para saber la cantidad de visitantes que tenemos de cada ciudad tipeamos:
35 select ciudad, count(*)
36 from visitantes
37 group by ciudad;
38
```

The results are displayed in a "Result Grid" below the query editor. The grid shows the following data:

| ciudad        | count(*) |
|---------------|----------|
| Cordoba       | 5        |
| Alta Gracia   | 4        |
| Villa Dolores | 1        |

The "Result 1" tab is selected at the bottom of the window.

## MySQL\_ Operador y función.

### 15. Selección de un grupo de registros (having)

Así como la cláusula "where" permite seleccionar (o rechazar) registros individuales; la cláusula "having" permite seleccionar (o rechazar) un grupo de registros.

Si queremos saber la cantidad de libros agrupados por editorial usamos la siguiente instrucción 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 siguiente instrucción:

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

Se utiliza "having", seguido de la condición de búsqueda, para seleccionar ciertas filas retornadas por la cláusula "group by".

Veamos otros ejemplos. Queremos el promedio de los precios de los libros agrupados por editorial:

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

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

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

En algunos casos es posible confundir las cláusulas "where" y "having". Queremos contar los 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 contarlos y 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 condiciones para la selección de registros de una salida "group by".

## MySQL\_ Operador y función.

Veamos otros ejemplos combinando "where" y "having".

Queremos la cantidad de libros, sin considerar los que tienen precio nulo, agrupados por editorial, sin considerar la editorial "Planeta":

```
select editorial, count(*) from libros
where 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 cumplan con la condición dada en el "having".

Generalmente se usa la cláusula "having" con funciones de agrupamiento, esto no puede hacerlo 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 libros
group by editorial
having count(*) > 2;
```

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

```
select editorial, max(precio) from libros
group by editorial
having max(precio)>=30;
```

Esta misma sentencia puede usarse empleando 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;
```

# MySQL\_ Operador y función.

## Servidor de MySQL instalado en forma local.

Ingrese al programa "Workbench" y ejecutemos el siguiente bloque de instrucciones SQL para analizar la cláusula having:

```
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('Antologia poetica','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 camara secreta','J.K. Rowling','Paidos',46.00);
insert into libros (titulo,autor,editorial,precio)
  values('Alicia en el pais 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 2
select 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
```

## MySQL\_ Operador y función.

```
group by editorial;

-- sólo queremos aquellos cuyo promedio supere los 25 euros:
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 libros
  where editorial<>'Planeta'
  group by editorial;
select editorial, count(*) from libros
  group 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 null
  group by editorial
  having editorial<>'Planeta';

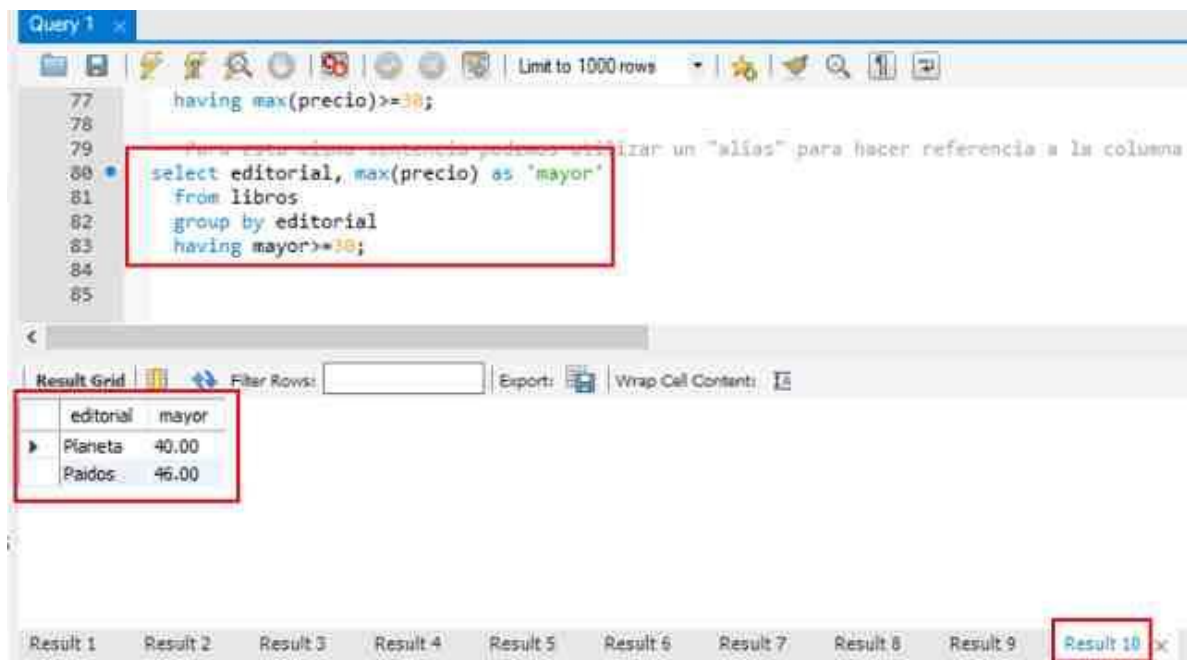
-- promedio de los precios agrupados por editorial, de aquellas editoriales
-- que tienen más de 2 libros
select editorial, avg(precio) from libros
  group by editorial
  having count(*) > 2;

-- mayor valor de los libros agrupados por editorial y luego seleccionar las filas
-- que tengan un valor mayor o igual a 30
select 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;
```

## MySQL\_ Operador y función.

Que nos genera una salida similar a esta:



The screenshot shows a MySQL query editor window titled "Query 1". The query is as follows:

```
77 having max(precio)>=38;  
78  
79 Para esta consulta podemos utilizar un "alias" para hacer referencia a la columna  
80 select editorial, max(precio) as 'mayor'  
81 from libros  
82 group by editorial  
83 having mayor>=38;  
84  
85
```

The query results are displayed in a table with two columns: "editorial" and "mayor". The results are:

| editorial | mayor |
|-----------|-------|
| Planeta   | 40.00 |
| Paidós    | 45.00 |

The "Result 10" tab is selected at the bottom of the window.

# MySQL\_ Operador y función.

## 16. Registros duplicados (distinct)

Con la cláusula "distinct" se especifica que los registros con ciertos datos duplicados sean obviadas en el resultado. Por ejemplo, queremos conocer todos los autores de los cuales tenemos libros, si utilizamos esta sentencia:

```
select autor from libros;
```

Aparecen repetidos. Para obtener la lista de autores sin repetición usamos:

```
select distinct autor from libros;
```

También podemos tipear:

```
select autor from libros  
group by autor;
```

Note que en los tres casos anteriores aparece "null" como un valor para "autor". Si sólo queremos la lista de autores conocidos, es decir, no queremos incluir "null" en la lista, podemos utilizar la sentencia siguiente:

```
select distinct autor from libros  
where autor is not null;
```

Para contar los distintos autores, sin considerar el valor "null" usamos:

```
select count(distinct autor)  
from libros;
```

Note que si contamos los autores sin "distinct", no incluirá los valores "null" pero si los repetidos:

```
select count(autor)  
from libros;
```

Esta sentencia cuenta los registros que tienen autor.

Para obtener los nombres de las editoriales usamos:

```
select editoriales from libros;
```

Para una consulta en la cual los nombres no se repitan tipeamos:

```
select distinct editorial from libros;
```

Podemos saber la cantidad de editoriales distintas usamos:

```
select count(distinct editoriales) from libros;
```

Podemos combinarla con "where". Por ejemplo, queremos conocer los distintos autores de la editorial "Planeta":

```
select distinct autor from libros  
where editorial='Planeta';
```

También puede utilizarse con "group by":



## MySQL\_ Operador y función.

```
select editorial, count(distinct autor)
from libros
group by editorial;
```

Para mostrar los títulos de los libros sin repetir títulos, usamos:

```
select distinct titulo from libros
order by titulo;
```

La cláusula "distinct" afecta a todos los campos presentados. Para mostrar los títulos y editoriales de los libros sin repetir títulos ni editoriales, usamos:

```
select distinct titulo,editorial
from libros
order by titulo;
```

Note que los registros no están duplicados, aparecen títulos iguales pero con editorial diferente, cada registro es diferente.

### Servidor de MySQL instalado en forma local.

Ingresemos al programa "Workbench" y ejecutemos el siguiente bloque de instrucciones SQL:

```
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,
  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('Martin Fierro','Jose Hernandez','Planeta',42.20);
insert into libros (titulo,autor,editorial,precio)
values('Antologia poetica','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','Bioy Casares- Borges','Paidos',36.40);
insert into libros (titulo,autor,editorial,precio)
values('Manual de PHP', null, 'Paidos',30.80);
insert into libros (titulo,autor,editorial,precio)
values('Harry Potter y la piedra filosofal','J.K. Rowling','Planeta',45.00);
insert into libros (titulo,autor,editorial,precio)
```

## MySQL\_ Operador y función.

```
values('Harry Potter y la camara secreta','J.K. Rowling','Planeta',46.00);
insert into libros (titulo,autor,editorial,precio)
values('Alicia en el pais de las maravillas','Lewis Carroll','Paidos',null);
insert into libros (titulo,autor,editorial,precio)
values('Alicia en el pais de las maravillas','Lewis Carroll','Emece',12.10);

-- Para obtener la lista de autores sin repetición usamos "distinct"
select distinct autor
from libros;

-- Si sólo queremos la lista de autores conocidos, es decir,
-- no queremos incluir "null" en la lista
select distinct autor
from libros
where autor is not null;

-- Para contar los distintos autores, sin considerar el valor "null"
select count(distinct autor)
from libros;

-- contamos los autores sin "distinct", no incluirá los valores "null"
-- pero si los repetidos:
select count(autor)
from libros;

-- los nombres de las editoriales
select editorial
from libros;

-- Nombres de las editoriales sin repetición
select distinct editorial
from libros;

-- cantidad de editoriales distintas
select count(distinct editorial)
from libros;

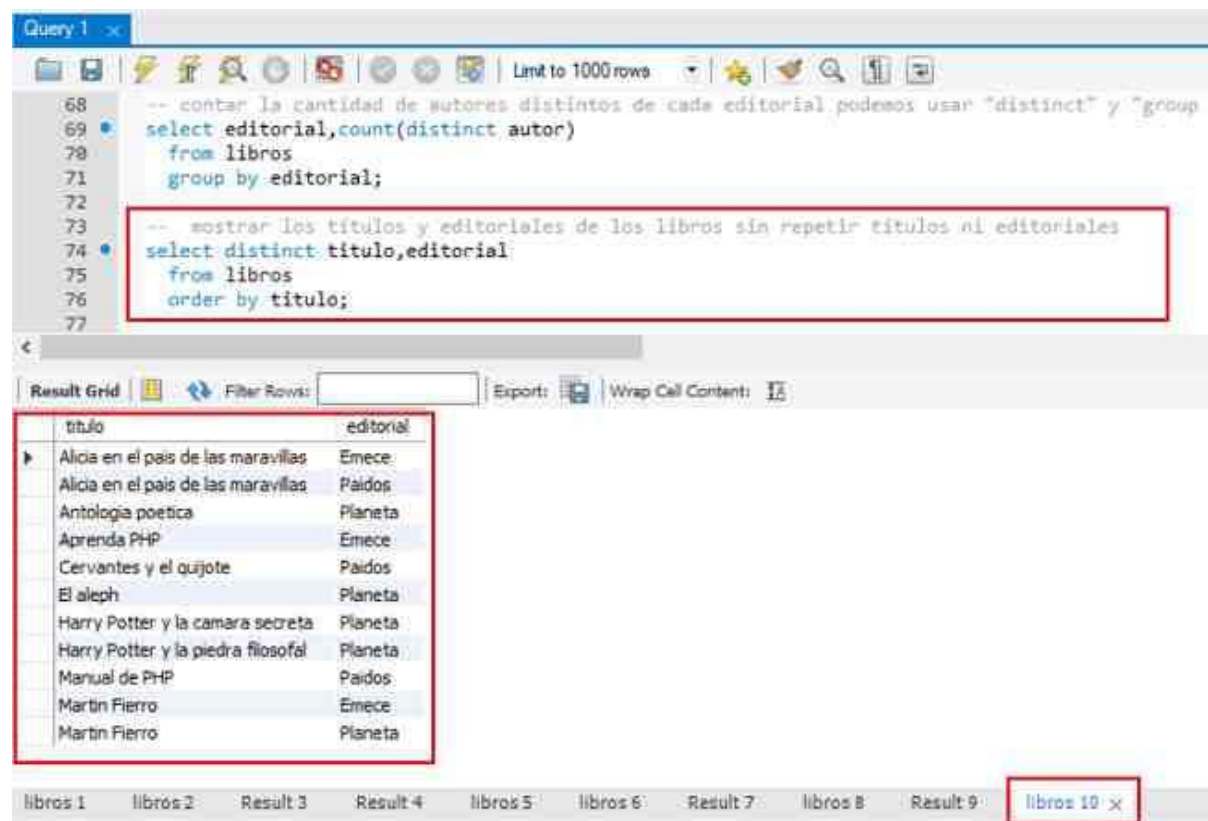
-- distintos autores de la editorial "Planeta"
select distinct autor from libros
where editorial='Planeta';

-- contar la cantidad de autores distintos de cada editorial
-- podemos usar "distinct" y "group by"
select editorial,count(distinct autor)
from libros
group by editorial;
```

## MySQL\_ Operador y función.

```
-- mostrar los títulos y editoriales de los libros sin repetir
-- títulos ni editoriales
select distinct titulo,editorial
from libros
order by titulo;
```

Genera una salida similar a esta:



The screenshot shows a MySQL query editor window titled "Query 1". The query is as follows:

```
-- contar la cantidad de autores distintos de cada editorial podemos usar "distinct" y "group
68
69 • select editorial,count(distinct autor)
70     from libros
71     group by editorial;
72
73 -- mostrar los títulos y editoriales de los libros sin repetir títulos ni editoriales
74 • select distinct titulo,editorial
75     from libros
76     order by titulo;
77
```

The results are displayed in a "Result Grid" below the query. The grid has two columns: "titulo" and "editorial". The results are as follows:

| titulo                              | editorial |
|-------------------------------------|-----------|
| Alicia en el país de las maravillas | Emece     |
| Alicia en el país de las maravillas | Paidós    |
| Antología poética                   | Planeta   |
| Aprenda PHP                         | Emece     |
| Cervantes y el quijote              | Paidós    |
| El aleph                            | Planeta   |
| Harry Potter y la cámara secreta    | Planeta   |
| Harry Potter y la piedra filosofal  | Planeta   |
| Manual de PHP                       | Paidós    |
| Martin Fierro                       | Emece     |
| Martin Fierro                       | Planeta   |

The "Result Grid" tab is highlighted in the bottom tab bar, which also includes tabs for "libros 1", "libros 2", "Result 3", "Result 4", "libros 5", "libros 6", "Result 7", "libros 8", "Result 9", and "libros 10".

# MySQL\_ Operador y función.

## 17. Alias

Un "alias" se usa como nombre de un campo o de una expresión o para referenciar una tabla cuando se utilizan más de una tabla (tema que veremos más adelante).

Cuando usamos una función de agrupamiento, por ejemplo:

```
select count(*)  
from libros  
where autor like '%Borges%';
```

la columna en la salida tiene como encabezado "count(\*)", para que el resultado sea más claro podemos utilizar un alias:

```
select count(*) as librosdeborges  
from libros  
where autor like '%Borges%';
```

La columna de la salida ahora tiene como encabezado el alias, lo que hace más comprensible el resultado.

Un alias puede tener hasta 255 caracteres, acepta todos los caracteres. La palabra clave "as" es opcional en algunos casos, pero es conveniente usarla. Si el alias consta de una sola cadena las comillas no son necesarias, pero si contiene más de una palabra, es necesario colocarla entre comillas.

Se pueden utilizar alias en las cláusulas "group by", "order by", "having". Por ejemplo:

```
select editorial as 'Nombre de editorial'  
from libros  
group by 'Nombre de editorial';  
select editorial, count(*) as cantidad  
from libros  
group by editorial  
order by cantidad;  
select editorial, count(*) as cantidad  
from libros  
group by editorial  
having cantidad>2;
```

No está permitido utilizar alias de campos en las cláusulas "where".

Los alias serán de suma importancia cuando rescate datos desde el lenguaje PHP

### Servidor de MySQL instalado en forma local.

Ingresemos al programa "Workbench" y ejecutemos el siguiente bloque de instrucciones SQL:

## MySQL\_ Operador y función.

```
drop table if exists libros;

create table libros(
  codigo int unsigned auto_increment,
  titulo varchar(50) not null,
  autor varchar(30),
  editorial varchar(15),
  precio decimal(5,2),
  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('Antologia poetica','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 camara secreta','J.K. Rowling','Paidos',46.00);
insert into libros (titulo,autor,editorial,precio)
  values('Alicia en el pais de las maravillas','Lewis Carroll','Paidos',null);

select count(*) as 'Libros de Borges'
  from libros
  where autor like '%Borges%';

select editorial as 'Nombre de editorial'
  from libros
  group by 'Nombre de editorial';

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

select editorial, count(*) as cantidad
  from libros
  group by editorial
```

## MySQL\_ Operador y función.

having cantidad>2;

Genera una salida similar a esta:

The screenshot shows a MySQL query editor window titled "Query 1". The query is as follows:

```
38 group by 'Nombre de editorial';
39
40 • select editorial, count(*) as cantidad
41   from libros
42   group by editorial
43   order by cantidad;
44
45 • select editorial, count(*) as cantidad
46   from libros
47   group by editorial
48   having cantidad>2;
49
```

The query is executed, and the result is displayed in a table with the following columns: editorial, cantidad. The result is:

| editorial | cantidad |
|-----------|----------|
| Paidos    | 5        |

The result is displayed in a window titled "Result 4 x".