**SHOW DATABASES**

lista las bases de datos administradas por el servidor

**SHOW TABLES**

Lista todas las tablas de la base de datos seleccionada.

**SHOW COLUMNS** **FROM** customers

Lista todas las columnas de una tabla determinada.

**SELECT** FirstName **FROM** customers

La declaracion SELECT selecciona columnas de una determinada tabla.

**SELECT** FirstName **FROM** customers;

**SELECT** City **FROM** customers;

Tambien se puede generar una multiple consulta.

**SELECT** \* **FROM** customers;

El asterisco genera que se seleccionen todas las columnas de la tabla determinada.

**SELECT DISTINCT** City **FROM** customers;

La declaracion DISTINCT retorna todos los valores de una columna de una tabla seleccionada que no esten repetidos.

**SELECT** ID, FirstName, LastName, City

**FROM** customers **LIMIT** 5;

La declaracion LIMIT hace que se seleccione una determina cantidad de datos especificada por un numero.

**SELECT** ID, FirstName, LastName, City

**FROM** customers **LIMIT** 3, 4;

Tambien se pueden delimiter de que índice traer y hasta donde.

LOS INDICES EMPIEZAN POR EL 0.  
  
**SELECT** \* **FROM** customers

**ORDER BY** LastName, Age;

ORDER BY hacer que se ordenen los registros traídos pero de manera ASCENDENTE.  
Tambien se pueden ordenar por prioridad de criterios. En el ejemplo esta ordenando por LastName, si hay dos registros iguales , va a ordenar por Age.  
  
**SELECT** \* **FROM** customers

**WHERE** ID = 7;

La declaración WHERE hace que según su condición muestre datos.  
En el ejemplo anterior muestra todas las columnas de la tabla customers donde los ID sean igual a 7.

**OPERADORES**

Imagen que contiene captura de pantalla

Descripción generada automáticamente

**SELECT** \* **FROM** customers

**WHERE** ID != 5;

Aca estamos usando el operador “distino” con lo cual en el ejemplo anterior traemos todos los datos de las columnas de la tabla customers donde el ID sea distinto de 5.

**SELECT** \* **FROM** customers

**WHERE** ID **BETWEEN** 3 **AND** 7;

El operador BETWEEN selecciona desde el ID 3 especificado hasta el ID 7 inclusive.

**WHERE**

**SELECT** ID, FirstName, LastName, City

**FROM** customers

**WHERE** City = 'New York';

Con la condicion WHERE podemos seleccionar valores de texto .  
en el ejemplo anterior estamos seleccionando unos determinados campos de la tabla customers donde city sea igual a new york.  
  
**OPERADORES LOGICOS**

Imagen que contiene captura de pantalla

Descripción generada automáticamente

**AND**

**SELECT** ID, FirstName, LastName, Age

**FROM** customers

**WHERE** Age >= 30 **AND** Age <= 40;

Al utilizar un operador lógico como AND se fija en la condición , con lo cual esta consulta traerá solamente los datos en los que la edad esten entre 30 y 40 años inclusive.

**OR**

Imagen que contiene captura de pantalla

Descripción generada automáticamente

**SELECT** \* **FROM** customers

**WHERE** City = 'New York' **OR** City = 'Chicago';

En esta consulta me va atraer los datos cuando la ciudad sea new york o chicago.

**COMBINANDO OPERADORES**  
**SELECT** \* **FROM** customers

**WHERE** City = 'New York'

**AND** (Age=30 **OR** Age=35);

En esta consulta se va a fijar en los registros donde city sea igual a new york y donde la edad sea igual a 30 o igual a 35.  
  
**IN**

El operador IN es utilizado cuando quieres comparar una columna con más de un valor.

**SELECT** \* **FROM** customers

**WHERE** City = 'New York'

**OR** City = 'Los Angeles'

**OR** City = 'Chicago';

**SELECT** \* **FROM** customers

**WHERE** City **IN** ('New York', 'Los Angeles', 'Chicago');

**NOT IT**

te permite excluir una lista de valores específicos del conjunto de resultados.

**SELECT** \* **FROM** customers

**WHERE** City **NOT** **IN** ('New York', 'Los Angeles', 'Chicago');

En esta consulta traera resultados donde city sea diferente a new york , los ángeles y chicago.

**CONCAT**

**SELECT** **CONCAT**(FirstName, ', ' , City) **FROM** customers;

La seleccion CONCAT sirve para concatenar dos o mas datos de las columnas.

**AS**

Los resultados de una concatenación resultan en una nueva columna. El nombre predeterminado de la columna será el de la función CONCAT.

Puedes asignar un nombre personalizado a la columna resultante utilizando la palabra clave AS:

**SELECT** **CONCAT**(FirstName,', ', City) **AS** new\_column

**FROM** customers;

**Operadores aritméticos**

Los operadores aritméticos incluyen adición (+), sustracción (-), multiplicación (\*) y división (/).

**SELECT** ID, FirstName, LastName, Salary**+**500 **AS** Salary

**FROM** employees;

En esta consulta a la columna salary le suma a todos 500 y la renombra.

**UPPER**

**SELECT** FirstName, **UPPER**(LastName) **AS** LastName

**FROM** employees;

Esta función hace que los datos de LastName se pongan en mayuscula.

**SQRT y AVG**

**SELECT** Salary, **SQRT**(Salary)

**FROM** employees;

La function SQRT realiza la raíz cuadrada de salary.

**SELECT** **AVG**(Salary) **FROM** employees;

La function AVG realiza el promedio de todos los salary.  
  
**SUM**

**SELECT** **SUM**(Salary) **FROM** employees;

Esta function suma todos los datos de salarios que hay en la tabla employees.

**SUBCONSULTA**

**SELECT** FirstName, Salary **FROM** employees

**WHERE** Salary > (**SELECT** **AVG**(Salary) **FROM** employees)

**ORDER** **BY** Salary **DESC**;

En esta consulta lo que hace es fijarse en todos los salarios que son mayores al promedio de todos los salarios de employees y los ordena de manera descendiente.  
  
**LIKE**

**SELECT** \* **FROM** employees

**WHERE** FirstName **LIKE** 'A%';

La function like en este ejemplo muestra todos los datos de la columna FirstName de la tabla employees que empiecen con la letra A.  
  
**SELECT** \* **FROM** employees

**WHERE** LastName LIKE '%s';

En esta consulta muestra todos los datos de la columna LastName de la tabla employees que terminen con la letra s.

**MIN**

**SELECT** **MIN**(Salary) **AS** Salary **FROM** employees;

En esta consulta la función min se fija en todos los datos de la columna salarios de la tabla employees y muestra el salario minimo.

**HAVING**

SELECT column\_name(s)  
FROM table\_name  
WHERE condition  
GROUP BY column\_name(s)HAVING conditionORDER BY column\_name(s);

La cláusula HAVING se agregó a SQL porque la palabra clave WHERE no se pudo usar con funciones agregadas.

LLAVE PRIMARIA

Una llave primaria es un campo único , irrepetible que sirve para identificar los registros de una tabla como distintos, además de que sirve para hacer relaciones ( uno a uno, uno a muchos, muchos a muchos entre tablas, sin las llaves primaria las bases de dato relacionales no tendrían sentido y siempre existiría información repetida.

LLAVE FORANEA

Es uno o mas campos de un tabla que hacen referencia al campo o campos de llave principal de otra tabla, una llave foranea indica como esta relacionadas las tablas. Los datos en los campos de ambas deben coincidir, aunque los nombres de los campos no sean los mismos.