SQL

Algunos de los comandos más importantes en SQL son:

- *select* – extrae información de una base de datos: “ select * from NombreDeLaTabla”
- *update* – actualiza una base de datos
- *delete* – elimina datos de una base de datos
- *insert into* – inserta nueva información en una base de datos
- *create database* – crea una nueva base de datos
- *alter database* – modifica una base de datos
- *create table* – crea una nueva tabla
- *alter table* – modifica una tabla
- *drop table* – elimina una tabla
- *create index* – crea un index
- *drop index* – elimina un index

COMANDO SELECT

Se usa para seleccionar datos/información de una base de datos:

```
SELECT CustomerName, City FROM Customers;
```

En este ejemplo, dentro de la tabla llamada “Customers” seleccionamos los siguientes datos: “CustomerName” y “City”. Para más claridad: SELECT DatosSeparadosPorComas FROM NombreDeLaTabla.

Si queremos ver todos los datos/todas las columnas de una tabla, sin especificar datos, utilizamos los siguiente:

```
SELECT * FROM Customers;
```

El * indica que queremos ver todo de la tabla.

COMANDO SELECT DISTINCT

Se utiliza para seleccionar datos/información distintos en una base de datos, es decir, que no se repitan o no estén duplicados:

```
SELECT DISTINCT Country FROM Customers;
```

En este ejemplo, queremos seleccionar todos los datos de la columna “Country” que no estén repetidos.

Si utilizamos el comando DISTINCT con una función llamada COUNT, nos devolverá el número de datos distintos; en el caso anterior, el número de países distintos:

```
SELECT COUNT(DISTINCT Country) FROM Customers;
```

COMANDO WHERE

Se utiliza para filtrar información, para extraer solo la información/datos que cumplan una determinada condición:

```
SELECT * FROM Customers
WHERE Country='Mexico';
```

En este ejemplo, seleccionamos todas las columnas de la tabla “Customers” cuyo valor de la columna “Country” sea “Mexico”.

Los datos de texto deben ir entre comillas comillas simples, mientras que los datos numéricos no llevan comillas.

Puedes utilizar otros operadores que no sean el “=”:

```
SELECT * FROM Customers
WHERE CustomerID > 80;
```

Aquí seleccionamos todos los clientes con un ID mayor de 80.

Podemos utilizar los siguientes operadores con el comando o la cláusula WHERE:

OPERADOR	DESCRIPCIÓN
=	Igual
>	mayor que
<	menor que
>=	Mayor o igual
<=	Menor o igual
<> or !=	Distinto o “no igual”
between	Entre un cierto rango
like	Buscar un patrón
in	Especificar múltiples posibles valores para una columna

COMANDO ORDER BY

Se usa para ordenar los resultados en orden ascendente o descendente:

```
SELECT * FROM Products
ORDER BY Price;
```

En este ejemplo, seleccionamos todos los datos de la tabla “Products” ordenados por su precio. De forma predefinida, ordena los datos de forma ascendente, para ordenarlos de forma descendente utilizamos DESC:

```
SELECT * FROM Products
ORDER BY Price DESC;
```

Para valores de tipo STRING, el comando los ordenará alfabéticamente. De la misma manera, para ordenarlos alfabéticamente, pero de manera inversa, utilizamos DESC.

Si utilizamos el siguiente comando:

```
SELECT * FROM Customers
ORDER BY Country, CustomerName;
```

La tabla estará ordenada según los valores de las columnas “Country” y “CustomerName”. Esto quiere decir que estará ordenada por los países, pero si varias filas tienen el mismo país, lo ordenará también por el nombre del cliente.

Si lo ordeno según varias columnas, puedo utilizar ASC y DESC al mismo tiempo, para indicar que lo ordene de forma ascendente para una columna, pero de forma descendente para otra.

OPERADOR AND

Se usa con el comando WHERE para extraer la información que cumpla más de una condición:

```
SELECT *
FROM Customers
WHERE Country = 'Spain' AND CustomerName LIKE 'G%';
```

En este caso, seleccionamos todos los clientes de España y cuyo nombre comience por la letra G. Las condiciones se separan con el operador AND.

Este operador muestra un dato si TODAS las condiciones son verdaderas.

Puedes combinar el operador AND y el operador OR en el mismo comando:

```
SELECT * FROM Customers
WHERE Country = 'Spain' AND (CustomerName LIKE 'G%' OR CustomerName LIKE
'R%');
```

En este caso, seleccionamos todos los clientes españoles cuyo nombre comience por G o por R.

Sin el paréntesis, selecciona todos los clientes españoles que empiecen por G y, aparte, todos los clientes que empiecen por R (estos últimos independientes del país).

OPERADOR OR

Se usa con el comando WHERE para extraer la información que cumpla más de una condición:

```
SELECT *
FROM Customers
WHERE Country = 'Germany' OR Country = 'Spain';
```

En este ejemplo, seleccionamos los clientes cuya región/país sea Alemania o España.

Este operador muestra un dato si UNA DE LAS DOS condiciones es verdadera.

OPERADOR NOT

Se utiliza con otros operadores para indicar el resultado opuesto o negativo:

```
SELECT * FROM Customers
WHERE NOT Country = 'Spain';
```

En este caso, seleccionamos los clientes que NO sean de España.

Se puede utilizar en combinación con otros operadores, como LIKE, BETWEEN, IN…

COMANDO INSERT INTO

Se utiliza para insertar nuevos datos en la tabla. Puedes escribir el comando de dos formas:

- Especificar los nombres de las columnas y los valores que serán insertados:

```
INSERT INTO table_name (column1, column2, column3, ...)
VALUES (value1, value2, value3, ...);
```

- Si vas a añadir datos a todas las columnas de la tabla, no necesitas especificar los nombres de las columnas. Simplemente revisa que el orden de los datos que vas a insertar sea el mismo que el de las columnas en la tabla:

```
INSERT INTO table_name
VALUES (value1, value2, value3, ...);
```

Si queremos insertar múltiples filas en un solo comando, usamos INSERT INTO, pero con muchos valores:

```
INSERT INTO Customers (CustomerName, ContactName, Address, City,
PostalCode, Country)
VALUES
('Cardinal', 'Tom B. Erichsen', 'Skagen 21', 'Stavanger', '4006',
'Norway'),
('Greasy Burger', 'Per Olsen', 'Gateveien 15', 'Sandnes', '4306',
'Norway'),
('Tasty Tee', 'Finn Egan', 'Streetroad 19B', 'Liverpool', 'L1 0AA',
'UK');
```

VALORES NULL Y NOT NULL

Para comprobar si un valor es NULL o NOT NULL utilizamos los operadores IS NULL y IS NOT NULL:

```
SELECT CustomerName, ContactName, Address
FROM Customers
WHERE Address IS NULL;
```

```
SELECT CustomerName, ContactName, Address
FROM Customers
WHERE Address IS NOT NULL;
```

Devuelve todos los nombres y contactos cuyo valor en Dirección es nulo.

COMANDO UPDATE

Se utiliza para modificar registros existentes dentro de la tabla:

```
UPDATE Customers
SET ContactName = 'Alfred Schmidt', City= 'Frankfurt'
WHERE CustomerID = 1;
```

En este caso, modificamos el primer cliente (ClienteID = 1) con un nuevo nombre de contacto y una nueva ciudad.

Para actualizar varios registros, modificamos el WHERE para que se adapte a nuestras necesidades.

COMANDO DELETE

Se utiliza para eliminar registros en una tabla:

```
DELETE FROM Customers WHERE CustomerName='Alfreds Futterkiste';
```

En este caso, eliminamos el cliente de nombre “Alfreds Futterkiste” en la tabla “Customers”.

Si no utilizamos el comando WHERE, eliminaremos todos los registros de la tabla.

Si, en vez de eliminar todos los registros o algunos, quiero eliminar la tabla en sí, utilizo DROP TABLE NombreDeLaTabla.

COMANDOS TOP, LIMIT, TETCH FIRST o ROWNUM

Se utilizan para especificar el número de registros que va a devolver:

Comando SELECT TOP

Es útil en tablas largas con miles de registros.

```
SELECT TOP 3 * FROM Customers;
```

COMANDO MIN() Y MAX()

```
SELECT MIN(Price)
FROM Products;
```

Encuentra los valores más bajos en la columna “Price” de la tabla “Products”

```
SELECT MAX(Price)
FROM Products;
```

Encuentra los valores más altos en la columna “Price” de la tabla “Products”

Si quieres darle un nuevo nombre a la columna que va a devolver, utiliza:

```
SELECT MIN(Price) AS SmallestPrice
FROM Products;
```

COMANDO COUNT()

Se utiliza para saber el número total de columnas que cumplen un determinado criterio:

```
SELECT COUNT(*)
FROM Products;
```

En este ejemplo, podremos saber el número de productos en la tabla “Products”.

Podemos añadir un WHERE para filtrar el resultado según la condición especificada:

```
SELECT COUNT(ProductID)
FROM Products
WHERE Price > 20;
```

Ahora podremos saber el número de productos cuyo precio (“Price”) sea mayor que 20.

Puedes especificar una columna en lugar del asterisco *, como en el ejemplo anterior. Si especificar una columna en lugar de poner un *, los valores NULL no se contarán.

Si utilizas DISTINCT con el COUNT, las filas con el mismo valor para una columna especificada se contarán como una sola:

```
SELECT COUNT(DISTINCT Price)
FROM Products;
```

Si quieres darle un nuevo nombre a la nueva columna, utiliza el AS:

```
SELECT COUNT(*) AS [number of records]
FROM Products;
```

FUNCIÓN SUM()

Devuelve la suma total de una columna numérica:

```
SELECT SUM(Quantity)
FROM OrderDetails;
```

En este ejemplo, devuelve la suma de todos los campos de la columna “Quantity”.

También puedes añadir un WHERE para filtrar los resultados.

También puedes darle un nuevo nombre a la nueva columna con un AS.

FUNCIÓN AVG()

Devuelve el valor promedio de una columna numérica:

```
SELECT AVG(Price)
FROM Products;
```

En este ejemplo, encontramos el precio promedio de todos los productos.

También puedes añadir un WHERE para filtrar resultados.

También puedes renombrar la columna resultante con un AS.

Para poder listar todos los registros con un valor superior al promedio:

```
SELECT * FROM Products
WHERE price > (SELECT AVG(price) FROM Products);
```

En este ejemplo, devuelve todos los productos con un precio mayor al promedio.

OPERADOR LIKE

Se utiliza con un WHERE para buscar un patrón específico en una columna:

```
SELECT * FROM Customers
WHERE CustomerName LIKE 'a%';
```

En este ejemplo, seleccionamos todos los clientes cuyo nombre comienza por “a”.

OPERADOR IN

Se utiliza como abreviatura de múltiples condiciones OR:

```
SELECT * FROM Customers
WHERE Country IN ('Germany', 'France', 'UK');
```

En este ejemplo, nos devuelve todos los clientes de Alemania, Francia o Reino Unido.

Si poner un NOT delante del IN, nos devuelve todos los registros que NO cumplen las condiciones (en el ejemplo anterior, los que NO sean de Alemania, Francia o Reino Unido).

Puedes utilizar IN con un WHERE en un subquery:

```
SELECT * FROM Customers
WHERE CustomerID IN (SELECT CustomerID FROM Orders);
```

En este ejemplo, nos devuelve todos los clientes que tienen un pedido en la tabla “Orders” (Pedidos). De la misma manera, si poner un NOT delante, nos devuelve todos los clientes que no tienen ningún pedido.

OPERADOR BETWEEN

Se utiliza para seleccionar valores dentro de un rango determinado, estos pueden ser números, textos o fechas:

```
SELECT * FROM Products
WHERE Price BETWEEN 10 AND 20;
```

En este ejemplo, selecciona todos los productos con un precio entre 10 y 20.

```
SELECT * FROM Products
WHERE Price BETWEEN 10 AND 20
AND CategoryID IN (1,2,3);
```

El NOT BETWEEN se utiliza para seleccionar valores que estén fuera del rango designado.

En este ejemplo, utilizamos el operador IN para que retorne, aparte de los productos que tengan un precio entre 10 y 20, también la CategoryID debe ser 1, 2 o 3.

```sql
SELECT * FROM Products
WHERE ProductName BETWEEN 'Carnarvon Tigers' AND 'Mozzarella di Giovanni'
ORDER BY ProductName;
```

Se puede utilizar para seleccionar entre textos.

```sql
SELECT * FROM Orders
WHERE OrderDate BETWEEN #07/01/1996# AND #07/31/1996#;
```

También se puede utilizar para buscar valores entre determinadas fechas.

OPERADOR AS

Se utiliza para darle un alias temporal a una tabla o una columna de dicha tabla.

```sql
SELECT CustomerID AS ID, CustomerName AS Customer
FROM Customers;
```

En este ejemplo, creamos dos alias, uno para la columna CustomerID y otro para la columna CustomerName.

Si quieres que el alias tenga espacios, pon el alias entre corchetes o comillas dobles.

Puedes crear un alias que combine varias columnas de una tabla:

```sql
SELECT CustomerName, (Address || ', ' || PostalCode || ' ' || City || ',
' || Country) AS Address
FROM Customers;
```

OPERADOR JOIN

Se utiliza para combinar filas de dos o más tablas, en función de una columna relacionada entre ellas.

Se utiliza INNER JOIN:

```sql
SELECT Orders.OrderID, Customers.CustomerName, Orders.OrderDate
FROM Orders
INNER JOIN Customers ON Orders.CustomerID=Customers.CustomerID;
```

En este ejemplo, se muestran los registros que tienen valores iguales en ambas tablas.

OPERADOR UNION

Se utiliza para combinar el resultado de dos o más SELECT:

```
SELECT City FROM Customers
UNION
SELECT City FROM Suppliers
ORDER BY City;
```

En este ejemplo, retorna las ciudades de ambas tablas “Customers” y “Suppliers” (solo valores distintos, no valores repetidos).

COMANDO GROUP BY

Se utiliza para agrupar filas que tienen valores iguales en nuevas filas “resumen”:

```
SELECT COUNT(CustomerID), Country
FROM Customers
GROUP BY Country
ORDER BY COUNT(CustomerID) DESC;
```

En este ejemplo, lista el número de clientes de cada país, ordenados de mayor a menor.