# Clase 06. Lenguaje SQL

## Clausula DISTINCT

Si sabe que la instrucción va a seleccionar datos redundantes y en su lugar preferiría ver solo valores diferentes, puede usar la palabra clave DISTINCT en la cláusula SELECT. Por ejemplo, supongamos que los clientes representan distintos intereses, algunos de los cuales usan el mismo número de teléfono. Si quiere asegurarse de que cada número de teléfono solo se vea una vez, la cláusula SELECT sería la siguiente:

**SELECT DISTINCT txtCustomerPhone**

## Clausula WHERE

En una instrucción SQL, la cláusula WHERE especifica criterios que tienen que cumplir los valores de campo para que los registros que contienen los valores se incluyan en los resultados de la consulta.

Se pueden usar criterios de consulta en la cláusula WHERE de una instrucción SELECT.

Una cláusula WHERE tiene la siguiente sintaxis básica:

**WHERE campo = criterio**

Por ejemplo, para obtener el número de teléfono de un cliente, pero solo se tiene el apellido del cliente es Valladares. En lugar de buscar en todos los números de teléfono de la base de datos, se puede usar una cláusula WHERE para limitar los resultados y que resulte más sencillo encontrar el número de teléfono. Si se da por hecho que los apellidos se almacenan en un campo denominado Apellidos, la cláusula WHERE sería similar a lo siguiente:

**Select NumTelefono From Clientes**
**WHERE Apellido = ‘Valladares’**

# Operadores

Permiten complementar las condiciones generadas con el Where para segmentar la información consultada. Así como también realizar operaciones aritméticas con campos.

Un operador es un símbolo que especifica una acción que se realiza en una o más expresiones.

## Operadores aritméticos

Los operadores aritméticos ejecutan operaciones matemáticas con dos expresiones de uno o varios tipos de datos. Se ejecutan en la categoría de tipos de datos numéricos.

| Operador | Operación | Ejemplo |
|---|---|---|
| + | Permite **sumar** los valores de dos columnas, una columna y un valor, o dos valores. | **Select campo1 + campo2 as campo3 ...** |
| - | Permite **restar**. | **Select campo1 - campo2 as campo3 ...** |
| * | Permite **multiplicar**. | **Select campo1 * campo2 as campo3** |
| / | Permite **dividir**. | **Select campo1 / campo2 as campo3** |

Sintaxis:

Por ejemplo, si queremos saber cuanto es el total de las compras realizadas por un cliente, y nuestra tabla de información sólo contiene el detalle por producto en la factura:

```
Select
Cliente,
Precio * Cantidad as TOTAL
From
Compras
```

## Operadores Relacionales

Los operadores relacionales o de comparación comprueban si dos expresiones tienen relación de igualdad o diferencia. Se pueden usar en todas las expresiones.

| Operador | Condición | Ejemplo |
|---|---|---|
| = | Verdadero si es igual | **Where** campo **=** "valor" |
| != | Verdadero si es diferente | **Where** campo **!=** "valor" |
| > | Verdadero si es mayor que | **Where** campo **>** "valor" |
| < | Verdadero si es menor que | **Where** campo **<** "valor" |
| >= | Verdadero si es mayor e igual que | **Where** campo **>=**"valor" |
| <= | Verdadero si es menor e igual que | **Where** campo **<=**"valor" |

Sintaxis:

Por ejemplo, si queremos saber cuanto es el total de las compras realizadas por los clientes que compraron un producto específico y nuestra tabla de información sólo contiene el detalle por producto en la factura:

```
Select
Cliente,
Producto,
Precio * Cantidad as TOTAL
From
Compras
Where Producto = 'Camisa'
```

## Operadores Lógicos

Los operadores lógicos comprueban la veracidad de alguna condición. Al igual que los operadores de comparación, devuelven el tipo de datos binario con el valor verdadero o falso, de esta forma permite que el resultado de nuestra consulta sea un conjunto de datos.

| Operador | Condición | Ejemplo |
|---|---|---|
| AND | Verdadero si se cumplen las dos condiciones | 1000 |
| OR | Verdadero si se cumple una de las dos condiciones | 10,51 |
| NOT | Verdadero si no se cumple la condición | Argentina |
| LIKE | Verdadero si cumple un patrón | L0332154 |
| IN | Verdadero si encuentra alguno de los valores otorgados | 01/10/2019 |
| BETWEEN | Verdadero si el valor se encuentra en el rango | 01/10/2019 12:15:00 |

Sintaxis
Por ejemplo, si queremos saber cuanto es el total de las compras realizadas por los clientes con apellido Lopez y Martinez, y nuestra tabla de información sólo contiene el detalle por producto en la factura:

```
Select
Cliente,
Producto,
Precio * Cantidad as TOTAL
From
Compras
Where Apellido in ('Lopez', 'Martinez')
```

# Clausula AS

Permite usar nombres de sustitutos para campos o expresiones: la palabra clave AS.

Se puede cambiar la etiqueta que se muestra para un campo en la vista de resultados mediante la palabra clave AS y un alias de campo en la cláusula SELECT. Un alias de campo es un nombre que se asigna a un campo de una consulta para facilitar la lectura de los resultados. Por ejemplo, si quiere seleccionar los datos de un campo denominado txtCustPhone y el campo contiene números de teléfono de clientes, puede mejorar la legibilidad de los resultados mediante un alias de campo en la instrucción SELECT.

Es importante tener en cuenta que este alias no afecta el nombre original del campo en la tabla de origen.

Por ejemplo:
**SELECT txtCustPhone AS Telefono**

# Clausula ORDER BY

Esta cláusula permite, ordenar el conjunto de resultados de una consulta por la lista de columnas especificada y, opcionalmente.
El orden en que se devuelven las filas en un conjunto de resultados no se puede garantizar, a menos que se especifique una cláusula ORDER BY.

Una cláusula ORDER BY tiene la siguiente sintaxis básica:

Por ejemplo, la sintaxis de la consulta para obtener todos los nombres y apellidos de los clientes, ordenado en orden alfabética de A hasta Z:

**Select Nombre, Apellido From Clientes**
**ORDER BY Nombre ASC**

# Clausula GROUP BY y Funciones

## Funcion

Una función definida por el usuario es un objeto de SQL que acepta parámetros, realiza una acción, como un cálculo complejo, y devuelve el resultado de esa acción como un valor. El valor devuelto puede ser un valor escalar (único) o una tabla.

Diferencias entre funciones y store procedures.
1. El procedimiento se denomina un objeto que realiza una o más tareas y la función se denomina objeto que realiza una acción específica.

2. El procedimiento está precompilado y la función se compila siempre en tiempo de ejecución

3. El procedimiento puede o no devolver un valor, pero la función debe devolver un valor siempre.

4. Las funciones definidas por el usuario son más fáciles de invocar que los procedimientos almacenados. Sin embargo, los procedimientos almacenados ofrecen un mejor rendimiento.

5. Podemos llamar a un procedimiento almacenado dentro de Procedimiento almacenado, Función dentro de la función, Stored Procedure dentro de la función pero no podemos llamar a la función dentro del procedimiento almacenado.

6. Podemos llamar a la función dentro de la instrucción select. Pero no podemos llamar un procedure ya que el procedure se llama solo con las sentencias EXEC y EXECUTE

7. En los procedures podemos utilizar cualquier sentencia del Tipo Select, Insert, Update, Delete mientras que las funciones solo podemos utilizar la sentencia Select para consultar datos o utilizarlas para procesar datos.

Las funciones de agregado realizan un cálculo en un conjunto de valores y devuelven un valor único. Se pueden usar en la lista de selección o en la cláusula HAVING de una instrucción SELECT.
Puede usar una agregación en combinación con la cláusula GROUP BY para calcular la agregación en las categorías de f

| Función | Operación | Ejemplo |
|---|---|---|
| **AVG** | Calcula el promedio | AVG(Campo1) as Promedio |
| **SUM** | Calcula la sumatoria | SUM(Campo2) as Sumatoria |
| **MAX** | Devuelve el valor máximo | MAX(Campo3) as Último |
| **MIN** | Devuelve el valor mínimo | MIN(Campo4) as Primero |
| **COUNT** | Calcula la cantidad de registros | COUNT(Campo5) as Conteo |
| **GROUP BY** | Permite que todas las funciones se puedan ejecutar | From Tabla Group By ... |

# Clausula HAVING

En una instrucción SQL, la cláusula HAVING especifica criterios que tienen que cumplir los valores de un campo nuevo generado en la consulta, este campo debe estar compuesto por alguna operación especial de agrupación como por ejemplo SUM, MAX, AVG, etc.

De esta forma los registros que apliquen al criterio o condición podrán ser filtrados y obtenidos como resultado.

Se pueden usar criterios de consulta en la cláusula HAVING de una instrucción SELECT, tales como los usados en la consulta WHERE.

Una cláusula HAVING tiene la siguiente sintaxis básica:

Por ejemplo, la sintaxis de la consulta para obtener los nombres de los clientes que tuvieron compras superiores a 3000, en el total de la tabla:

**Select Nombre, Sum(Compra) as TOTAL From Compras**
**HAVING Sum(Compras) > 3000**