

Taller SQL operadores

Inner Join:

es una operación en SQL que se utiliza para combinar filas de dos o más tablas basadas en una condición relacionada entre ellas. Solo devuelve las filas donde hay una coincidencia en ambas tablas involucradas. Si no hay coincidencias entre las tablas, esas filas no se incluirán en el resultado.

Ejemplo:

Supongamos que tienes dos tablas: Clientes y Pedidos.

Clientes:	
CienteID	Nombre
1	Juan
2	María
3	Carlos

Pedidos:		
PedidoID	CienteID	Producto
101	1	Laptop
102	2	Teléfono
103	1	Tablet

```
SELECT Clientes.Nombre, Pedidos.Producto
```

```
FROM Clientes
```

```
INNER JOIN Pedidos
```

```
ON Clientes.CienteID = Pedidos.CienteID;
```

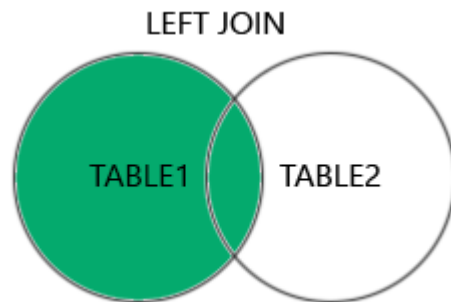
Si hacemos un INNER JOIN entre Clientes y Pedidos basándonos en la columna CienteID, solo se devolverán las filas donde exista una coincidencia en ambas tablas:

Resultado:	
Nombre	Producto
Juan	Laptop
María	Teléfono
Juan	Tablet

la consulta muestra los clientes que tienen pedidos, pero omite a aquellos que no tienen coincidencias (en este caso, Carlos no tiene pedidos y no aparece en el resultado).

Left Join:

LEFT JOIN (o LEFT OUTER JOIN) es una operación en SQL que devuelve todas las filas de la tabla de la izquierda (la primera tabla mencionada en la consulta) y las filas coincidentes de la tabla de la derecha (la segunda tabla mencionada). Si no hay coincidencias en la tabla de la derecha, aún se devuelven todas las filas de la tabla de la izquierda, pero las columnas de la tabla de la derecha tendrán valores NULL.



```
SELECT column_name(s)
FROM table1
LEFT JOIN table2
ON table1.column_name = table2.column_name;
```

Ejemplo:

Sigamos con las mismas tablas: Clientes y Pedidos.

Cientes:

CienteID	Nombre
1	Juan
2	María
3	Carlos

Pedidos:

PedidoID	CienteID	Producto
101	1	Laptop
102	2	Teléfono
103	1	Tablet

Si realizamos un LEFT JOIN entre Cientes y Pedidos:

SELECT Cientes.Nombre, Pedidos.Producto

FROM Cientes

LEFT JOIN Pedidos

ON Cientes.CienteID = Pedidos.CienteID;

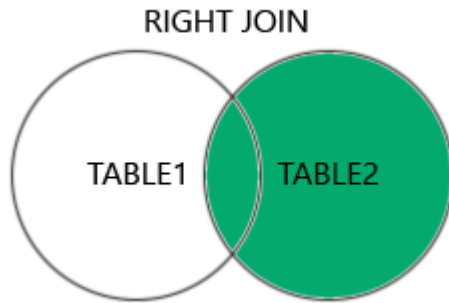
Resultado:

Nombre	Producto
Juan	Laptop
Juan	Tablet
María	Teléfono
Carlos	NULL

todas las filas de la tabla Cientes se devuelven. Si hay una coincidencia en Pedidos, se muestran los datos del producto. Si no hay coincidencia (como en el caso de Carlos), la columna correspondiente del producto tendrá un valor NULL.

Right Join:

RIGHT JOIN (o RIGHT OUTER JOIN) es una operación en SQL que devuelve todas las filas de la tabla de la derecha (la segunda tabla mencionada en la consulta) y las filas coincidentes de la tabla de la izquierda (la primera tabla mencionada). Si no hay coincidencias en la tabla de la izquierda, las columnas correspondientes de la tabla de la izquierda se rellenarán con valores NULL.



```
SELECT column_name(s)
FROM table1
RIGHT JOIN table2
ON table1.column_name = table2.column_name;
```

Ejemplo:

Cientes:

CienteID	Nombre
1	Juan
2	María
3	Carlos

Pedidos:

PedidoID	CienteID	Producto
101	1	Laptop
102	2	Teléfono
103	1	Tablet
104	4	Monitor

```
SELECT Cientes.Nombre, Pedidos.Producto
FROM Cientes
RIGHT JOIN Pedidos
ON Cientes.CienteID = Pedidos.CienteID;
```

Resultado:

Nombre	Producto
Juan	Laptop
María	Teléfono
Juan	Tablet
NULL	Monitor

En este caso, todas las filas de la tabla Pedidos se devuelven. Si hay una coincidencia en la tabla Clientes, se muestra el nombre del cliente. Si no hay coincidencia (como en el caso del pedido con el monitor), la columna correspondiente de la tabla Clientes tiene un valor NULL.

Like:

En MySQL, la cláusula LIKE se usa en una consulta SELECT para buscar un patrón específico dentro de una columna de tipo texto. Es muy útil cuando deseas buscar registros que coincidan parcialmente con un valor determinado, permitiendo búsquedas flexibles.

Sintaxis básica:

```
SELECT column_name(s)
```

```
FROM table_name
```

```
WHERE column_name LIKE pattern;
```

Caracteres comodines:

%: Representa cero o más caracteres. Puedes usarlo para buscar cualquier secuencia de caracteres antes o después de un texto específico.

Ejemplo: 'a%' coincidirá con cualquier texto que comience con la letra "a".

_: Representa un solo carácter. Se utiliza para buscar coincidencias con un carácter en una posición específica.

Ejemplo: 'a_' coincidirá con cualquier cadena que comience con "a" y tenga un carácter adicional.

Ejemplo:

Ejemplo 1: Buscar todos los nombres que comiencen con "J".

```
SELECT * FROM Clientes
```

```
WHERE Nombre LIKE 'J%';
```

Este ejemplo devolverá todos los clientes cuyos nombres comienzan con la letra "J", como "Juan" o "Julia".

Ejemplo 2: Buscar todos los nombres que terminen con "a".

```
SELECT * FROM Clientes  
  
WHERE Nombre LIKE '%a';
```

Este ejemplo devolverá todos los clientes cuyos nombres terminan con la letra "a", como "María" o "Laura".

Ejemplo 3: Buscar todos los nombres que contengan la letra "l".

```
SELECT * FROM Clientes  
  
WHERE Nombre LIKE '%l%';
```

Este ejemplo devolverá todos los clientes cuyos nombres contengan la letra "l" en cualquier posición, como "Carlos" o "Luis".

Ejemplo 4: Buscar nombres con exactamente 5 caracteres.

```
SELECT * FROM Clientes  
  
WHERE Nombre LIKE '_____';
```

En este caso, los cinco guiones bajos () indican que el nombre debe tener exactamente 5 caracteres.

Por defecto, en MySQL, las comparaciones con LIKE no son sensibles a mayúsculas y minúsculas cuando se usa el cotejamiento predeterminado (latin1_swedish_ci), lo que significa que 'a%' y 'A%' devolverán los mismos resultados. Sin embargo, esto depende de la configuración del cotejamiento de la base de datos.

Count:

En SQL, la función COUNT() se utiliza para contar el número de filas que cumplen con una determinada condición en una consulta. Esta función es comúnmente usada para calcular cuántos registros existen en una tabla o cuántos registros cumplen con un criterio específico.

Sintaxis básica:

```
SELECT COUNT(column_name)  
  
FROM table_name  
  
WHERE condition;
```

Ejemplo:

Ejemplo 1: Contar todas las filas de una tabla

Si deseas contar el número total de filas en una tabla (independientemente de si tienen valores NULL o no), puedes usar:

```
SELECT COUNT(*)
```

```
FROM Clientes;
```

Este ejemplo devolverá el número total de registros en la tabla Clientes.

Ejemplo 2: Contar filas que no tengan valores NULL en una columna específica

Si deseas contar cuántas filas tienen valores no nulos en una columna específica, puedes usar:

```
SELECT COUNT(Nombre)
```

```
FROM Clientes;
```

Este ejemplo devolverá el número de registros en la columna Nombre donde los valores no son NULL.

Ejemplo 3: Contar filas que cumplen una condición

Si deseas contar el número de filas que cumplen con una condición específica:

```
SELECT COUNT(*)
```

```
FROM Clientes
```

```
WHERE Ciudad = 'Madrid';
```

Diferencia entre COUNT(*) y COUNT(column_name):

COUNT(*): Cuenta todas las filas de la tabla, incluidas las que tienen valores NULL en cualquier columna.

COUNT(column_name): Cuenta solo las filas donde la columna especificada no es NULL.

Max():

La función MAX() en SQL se utiliza para obtener el valor máximo de una columna en un conjunto de datos. Es útil cuando quieres encontrar el valor más alto en una columna numérica, de fecha o incluso de texto (en este caso, devuelve el valor alfabéticamente mayor).

Sintaxis básica:

```
SELECT MAX(column_name)
```

```
FROM table_name
```

```
WHERE condition;
```

Ejemplos:

Ejemplo 1: Encontrar el valor máximo en una columna numérica

Si tienes una tabla llamada Empleados con una columna Salario y deseas encontrar el salario más alto:

```
SELECT MAX(Salario)
```

```
FROM Empleados;
```

Este ejemplo devolverá el salario más alto de todos los empleados.

Ejemplo 2: Encontrar la fecha más reciente

Si tienes una tabla llamada Pedidos con una columna FechaPedido, y deseas encontrar la fecha más reciente en que se realizó un pedido:

```
SELECT MAX(FechaPedido)
```

```
FROM Pedidos;
```

Este ejemplo devolverá la fecha más reciente en la que se hizo un pedido.

Ejemplo 3: Encontrar el valor máximo en una columna de texto

Aunque es menos común, también puedes utilizar MAX() en columnas de texto. En este caso, devolverá el valor más grande según el orden lexicográfico (alfabético). Por ejemplo, si tienes una columna Nombre en la tabla Clientes:

```
SELECT MAX(Nombre)
```

```
FROM Clientes;
```

Este ejemplo devolverá el nombre que es alfabéticamente mayor, como "Zulema", si está presente en los datos.

Min():

La función MIN() en SQL se utiliza para obtener el valor mínimo de una columna en un conjunto de datos. Al igual que MAX(), es útil para encontrar el valor más bajo en una columna numérica, de fecha o de texto (en este caso, devuelve el valor alfabéticamente menor).

Sintaxis básica:

```
SELECT MIN(column_name)
```

```
FROM table_name
```

```
WHERE condition;
```

Ejemplos:

Ejemplo 1: Encontrar el valor mínimo en una columna numérica

Si tienes una tabla llamada Empleados con una columna Salario y deseas encontrar el salario más bajo:


```
SELECT MIN(Salario)
```

```
FROM Empleados;
```

Este ejemplo devolverá el salario más bajo de todos los empleados.

Ejemplo 2: Encontrar la fecha más antigua

Si tienes una tabla llamada Pedidos con una columna FechaPedido, y deseas encontrar la fecha más antigua en que se realizó un pedido:

```
SELECT MIN(FechaPedido)
```

```
FROM Pedidos;
```

Este ejemplo devolverá la fecha más antigua en la que se hizo un pedido.

Ejemplo 3: Encontrar el valor mínimo en una columna de texto

Puedes usar MIN() en columnas de texto para obtener el valor alfabéticamente menor. Por ejemplo, si tienes una columna Nombre en la tabla Clientes:

```
SELECT MIN(Nombre)
```

```
FROM Clientes;
```

Este ejemplo devolverá el nombre que es alfabéticamente menor, como "Aaron" o "Ana".

AVG():

La función AVG() en SQL se utiliza para calcular el promedio de los valores en una columna numérica. Ignora los valores NULL y suma los valores no nulos, dividiéndolos por el número de registros no nulos.

Sintaxis básica:

```
SELECT AVG(column_name)
```

```
FROM table_name
```

```
WHERE condition;
```

Ejemplos:

Ejemplo 1: Calcular el promedio de una columna numérica

Si tienes una tabla llamada Empleados con una columna Salario y deseas calcular el salario promedio de todos los empleados:

```
SELECT AVG(Salario)
```

```
FROM Empleados;
```

Este ejemplo devolverá el salario promedio de todos los empleados.

Ejemplo 2: Calcular el promedio de ventas

Si tienes una tabla llamada Ventas con una columna MontoVenta y deseas calcular el promedio de las ventas realizadas:

```
SELECT AVG(MontoVenta)
FROM Ventas;
```

Este ejemplo devolverá el monto promedio de las ventas.

SUM():

La función SUM() en SQL se utiliza para sumar los valores de una columna numérica. Ignora los valores NULL y calcula la suma de todos los valores no nulos.

Sintaxis básica:

```
SELECT SUM(column_name)
FROM table_name
WHERE condition;
```

Ejemplos:

Ejemplo 1: Sumar los valores de una columna numérica

Si tienes una tabla llamada Ventas con una columna MontoVenta y deseas calcular el total de todas las ventas realizadas:

```
SELECT SUM(MontoVenta)
FROM Ventas;
```

Este ejemplo devolverá la suma de todas las ventas registradas en la tabla.

Ejemplo 2: Sumar los salarios de todos los empleados

Si tienes una tabla llamada Empleados con una columna Salario y deseas sumar los salarios de todos los empleados:

```
SELECT SUM(Salario)
FROM Empleados;
```

Este ejemplo devolverá la suma total de los salarios.

ORDER BY:

ORDER BY es una cláusula en SQL que se utiliza para ordenar el resultado de una consulta en orden ascendente o descendente, basado en una o más columnas. Puedes ordenar tanto por columnas numéricas, de texto, como de fechas.

Sintaxis básica:

```
SELECT column_name(s)
FROM table_name
ORDER BY column_name [ASC|DESC];
```

ASC: Ordena los resultados en orden ascendente (de menor a mayor). Es el comportamiento predeterminado si no se especifica.

DESC: Ordena los resultados en orden descendente (de mayor a menor).

Ejemplos:

Ejemplo 1: Ordenar por una columna en orden ascendente

Si tienes una tabla llamada Clientes y deseas ordenar los resultados por el nombre del cliente en orden alfabético:

```
SELECT Nombre, Ciudad
FROM Clientes
ORDER BY Nombre ASC;
```

Este ejemplo devolverá los nombres de los clientes en orden ascendente (A-Z).

Ejemplo 2: Ordenar por una columna en orden descendente

Si deseas ordenar a los clientes por la columna Edad en orden descendente:

```
SELECT Nombre, Edad
FROM Clientes
ORDER BY Edad DESC;
```

Este ejemplo devolverá los clientes ordenados de mayor a menor edad.

Ejemplo 3: Ordenar por múltiples columnas

Si deseas ordenar por varias columnas, por ejemplo, primero por Ciudad en orden ascendente y luego por Nombre en orden descendente:

```
SELECT Nombre, Ciudad
FROM Clientes
```

ORDER BY Ciudad ASC, Nombre DESC;

Este ejemplo devolverá los clientes ordenados primero por ciudad (A-Z) y luego, dentro de cada ciudad, los nombres se ordenarán en orden descendente (Z-A).

GROUP BY:

La cláusula GROUP BY en SQL se utiliza para agrupar filas que tienen los mismos valores en una o más columnas en un solo registro. Esta cláusula se usa frecuentemente junto con funciones de agregación (como COUNT(), SUM(), AVG(), MIN(), MAX()) para realizar cálculos en cada grupo de datos.

Sintaxis básica:

```
SELECT column_name(s), aggregate_function(column_name)
FROM table_name
GROUP BY column_name(s);
```

Ejemplos:

Ejemplo 1: Contar el número de registros por grupo

Si tienes una tabla llamada Ventas y deseas contar el número de ventas realizadas por cada vendedor:

```
SELECT VendedorID, COUNT(*)
FROM Ventas
GROUP BY VendedorID;
```

Este ejemplo devolverá el número total de ventas realizadas por cada vendedor.

Ejemplo 2: Calcular la suma de ventas por producto

Si tienes una tabla llamada Ventas y deseas calcular la suma total de ventas para cada producto:

```
SELECT ProductoID, SUM(MontoVenta)
FROM Ventas
GROUP BY ProductoID;
```

Este ejemplo devolverá el monto total de ventas para cada producto.

Ejemplo 3: Calcular el salario promedio por departamento

Si tienes una tabla llamada Empleados y deseas calcular el salario promedio de los empleados en cada departamento:

```
SELECT Departamento, AVG(Salario)
```

```
FROM Empleados
```

```
GROUP BY Departamento;
```

Este ejemplo devolverá el salario promedio de los empleados en cada departamento.

Ejemplo 4: Encontrar el salario más alto en cada departamento

Si deseas encontrar el salario más alto en cada departamento:

```
SELECT Departamento, MAX(Salario)
```

```
FROM Empleados
```

```
GROUP BY Departamento;
```

Este ejemplo devolverá el salario más alto en cada departamento.

LIMIT:

La cláusula LIMIT en SQL se utiliza para especificar el número máximo de registros que se deben devolver en una consulta. Es especialmente útil cuando deseas obtener solo una parte del conjunto de resultados, por ejemplo, los primeros 10 registros, o para paginar resultados en una aplicación.

Sintaxis básica:

```
SELECT column_name(s)
```

```
FROM table_name
```

```
LIMIT number_of_records;
```

Sintaxis con desplazamiento:

Algunos sistemas de gestión de bases de datos, como MySQL, también permiten especificar un desplazamiento para omitir un número determinado de registros antes de comenzar a devolver resultados.

```
SELECT column_name(s)
```

```
FROM table_name
```

```
LIMIT offset, number_of_records;
```

number_of_records: Número máximo de registros a devolver.

offset: Número de registros a omitir antes de comenzar a devolver resultados.

Ejemplos:

Ejemplo 1: Obtener los primeros 10 registros

Si deseas obtener los primeros 10 registros de una tabla llamada Clientes:

```
SELECT *  
  
FROM Clientes  
  
LIMIT 10;
```

Este ejemplo devolverá los primeros 10 registros de la tabla Clientes.

Ejemplo 2: Obtener los registros a partir del número 11

Si deseas obtener los registros 11 a 20 de una tabla Clientes:

```
SELECT *  
  
FROM Clientes  
  
LIMIT 10 OFFSET 10;
```

Este ejemplo omite los primeros 10 registros y devuelve los siguientes 10 registros (es decir, registros 11 a 20).

Ejemplo 3: Obtener los primeros 5 registros de una tabla ordenada

Si deseas obtener los primeros 5 registros después de ordenar la tabla Ventas por el monto de venta en orden descendente:

```
SELECT *  
  
FROM Ventas  
  
ORDER BY MontoVenta DESC  
  
LIMIT 5;
```

Este ejemplo devuelve las 5 ventas con los montos más altos.

BETWEEN:

La cláusula BETWEEN en SQL se utiliza para filtrar los resultados de una consulta para que solo incluya registros cuyo valor en una columna esté dentro de un rango especificado. Este rango puede ser definido por dos valores, el límite inferior y el límite superior, y BETWEEN incluye ambos extremos en el resultado.

Sintaxis básica:

```
SELECT column_name(s)  
  
FROM table_name
```

WHERE column_name BETWEEN value1 AND value2;

Ejemplos:

Ejemplo 1: Filtrar por un rango de fechas

Si tienes una tabla llamada Pedidos con una columna FechaPedido y deseas encontrar todos los pedidos realizados entre el 1 de enero de 2024 y el 31 de diciembre de 2024:

```
SELECT *
```

```
FROM Pedidos
```

```
WHERE FechaPedido BETWEEN '2024-01-01' AND '2024-12-31';
```

Este ejemplo devolverá todos los pedidos realizados dentro del año 2024.

Ejemplo 2: Filtrar por un rango de valores numéricos

Si tienes una tabla llamada Productos con una columna Precio y deseas encontrar todos los productos cuyo precio está entre 50 y 100:

```
SELECT *
```

```
FROM Productos
```

```
WHERE Precio BETWEEN 50 AND 100;
```

Este ejemplo devolverá todos los productos cuyo precio está entre 50 y 100, incluidos ambos valores.

Ejemplo 3: Filtrar por un rango de edades

Si tienes una tabla llamada Empleados con una columna Edad y deseas encontrar todos los empleados cuya edad está entre 25 y 40 años:

```
SELECT *
```

```
FROM Empleados
```

```
WHERE Edad BETWEEN 25 AND 40;
```

Este ejemplo devolverá todos los empleados cuya edad esté en el rango de 25 a 40 años, incluidos los extremos.

Ejemplo con texto:

Si tienes una columna Nombre en una tabla Clientes y deseas encontrar los clientes cuyos nombres estén entre "Carlos" y "Marta" en orden alfabético:

```
SELECT *
```

```
FROM Clientes
```

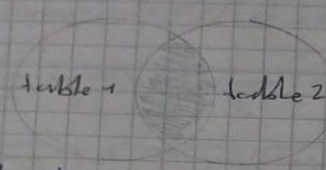
WHERE Nombre BETWEEN 'Carlos' AND 'Marta';

Este ejemplo devolverá todos los clientes cuyos nombres estén en el rango alfabético de "Carlos" a "Marta".

Evidencia de cuaderno:

Trabaja con las siguientes sentencias

Inner Join: Es una operación en SQL que se utiliza para combinar filas de dos o más tablas basadas en una condición relacionada.



Ejemplo: Clientes

Clientid	Nombre
1	Juan
2	Marta
3	Carlos

Pedidos

PedidoId	Clientid	Producto
101	1	Laptop
102	2	telefono
103	1	tableta

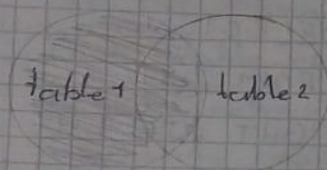
Sentencias

```
SELECT Clientes.Nombre, Pedidos.Producto
FROM Clientes
INNER JOIN Pedidos
ON Clientes.Clientid = Pedidos.Clientid;
```

Resultados:

Nombre	Producto
Juan	Laptop
Marta	telefono
Juan	tableta

Left Join: Es una operación en SQL que devuelve todos los filas de la tabla de la izquierda (la primera tabla mencionada en la consulta) y los filas coincidentes de la tablas de la derecha (la segunda tabla mencionada).



Sentencias

Select column_name(s)

FROM table1

LEFT JOIN table2

ON table1.column_name = table2.column_name;

Ejemplo

Clientes

Clientid	Nombre
1	Juan
2	Maria
3	Carlos

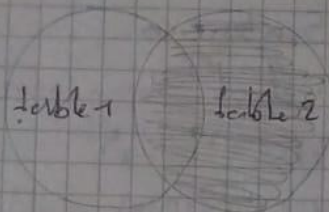
Pedidos

Pedidoid	Clientid	Products
101	1	Laptop
102	2	Telefono
103	1	tablet

Resultados:

Nombre	Products
Juan	Laptop
Juan	tablet
Maria	Telefono
Carlos	Null

Right Join: Es una operación en SQL que devuelve todos los filas de la tabla de la derecha (la segunda tabla mencionada en la consulta) y las filas coincidentes de la tabla de la izquierda.



Sintaxis

SELECT column-name(s)

FROM table 1

RIGHT JOIN table 2

ON table1.column-name=table2.column-name

Ejemplo Clientes

Clientid	Nombre
1	Juan
2	Maria
3	Carlos

Pedidos

Pedidoid	Clientid	Producto
101	1	Laptop
102	2	telefono
103	1	tablet
104	4	Monitor

Resultados

Nombre	Producto
Juan	Laptop
Maria	telefono
Juan	tablet
Null	Monitor

Like: La cláusula like se usa en una consulta SELECT para buscar un patrón específico dentro de una columna de tipo texto. Muy útil cuando deseas buscar registros que coincidan parcialmente con un valor determinado, permitiendo búsquedas flexibles.

Sintaxis SELECT column_name(s)
FROM table_name
WHERE column_name LIKE pattern;

Ejemplo % Representa cero o más caracteres. Puedes usarlo para buscar cualquier secuencia de caracteres antes o después del texto

SELECT Nombre
FROM Productos
WHERE Nombre LIKE '%Camisa%'

Nombre		Nombre
Camisa de Algodón		Camisa de Algodón
Camisa de mezclilla	→	Camisa de mezclilla
Pantalones		Camisa camuflaje
Chaqueta camuflaje		

Count: Se utiliza para contar el número de filas que cumplen con una determinada condición en una consulta

Sintaxis SELECT COUNT(column_name)
FROM table_name
WHERE condition;

Ejemplo

Id	Nombre	Departamento
1	Juan	Ventas
2	Maria	Marketing
3	Carlos	Ventas
4	Ana	Recursos Humanos

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

Count
4

Max(): Se utiliza para obtener el valor máximo de una columna en un conjunto de datos.

Sintaxis

```
SELECT MAX(column-name)
FROM table_name
WHERE condition;
```

Ejemplos

```
SELECT MAX(Precio) AS PrecioMaximo
FROM Productos;
```

Producto	Precio
Camisa	20.000
Pantalones	35.000
Chaqueta	50.000
Zapatos	45.000

→	PrecioMaximo
	50.000

Min (): Se utiliza para obtener el valor mínimo de una columna en un conjunto de datos

Sintaxis Básica:

```
SELECT MIN(column-name)
FROM table-name
WHERE condition;
```

Ejemplo

```
SELECT MIN(Precio) AS PrecioMinimo
FROM Productos;
```

Producto	Precio
Camiseta	20.000
Pantalones	35.000
Chaquetas	50.000
Zapatos	40.000

PrecioMinimo
20.000

AVG (): Se utiliza para calcular el promedio de los valores en una columna numérica.

Sintaxis:

```
SELECT AVG(column-name)
FROM table-name
WHERE condition;
```

Ejemplo `SELECT AVG(Precio) AS Promedio PrecioPromedio`
`FROM Productos;`

Producto	Precio
Camiseta	10.000
Pantalones	35.000
Chaqueta	80.000
Zapatos	40.000

Precio Promedio
36.25

`SUM()`: Se utiliza para sumar los valores de una columna numérica.

Sintaxis

`SELECT SUM(column_name)`
`FROM table_name`
`WHERE condition;`

Ejemplos

`SELECT SUM(MontoVenta) AS TotalVentas`
`FROM Ventas;`

VentaId	MontoVenta
1	100.000
2	150.000
3	200.000
4	250.000

TotalVentas
700.000

Order By: Se utiliza para ordenar el resultado de una consulta en orden ascendente o descendente, basado en una o más columnas.

Sintaxis

```
SELECT column_name(s)
FROM table_name
ORDER BY column_name [ASC|DESC];
```

Ejemplo:

```
SELECT Nombre, Edad
FROM Empleados
ORDER BY Edad ASC;
```

Nombre	Edad	⇒	Nombre	Edad
Ana	22		Ana	22
Juan	30		Maria	25
Maria	25		Juan	30

Group By: Se utiliza para agrupar filas que tienen los mismos valores en una o más columnas en un solo registro.

Sintaxis

```
SELECT column_name(s), aggregate_function(column_name)
FROM table_name
GROUP BY column_name(s)
```

Ejemplo

```
SELECT Departamento, COUNT(*) AS Numero Empleados  
FROM Empleados  
GROUP BY Departamento;
```

nombre	Departamento		Departamento	Numero Empleados
Ana	Ventas		Ventas	2
Juan	Marketing	⇒	Marketing	1
Maria	Ventas		Finanzas	7
Pedro	Finanzas			

Limit: Se utiliza para especificar el número máximo de registros que se deben devolver en una consulta.

Sintaxis:

```
SELECT column_name(s)  
FROM table_name  
LIMIT number_of_records;
```

Ejemplo:

```
SELECT *  
FROM Productos  
LIMIT 5;
```


ProductoId	Nombre	Precio
1	Camiseta	20.000
2	Pantalones	35.000
3	Chaqueta	50.000
4	Zapatos	40.000
5	Sombrero	15.000
6	Bufanda	25.000

=>

ProductoId	Nombre	Precio
1	Camiseta	20.000
2	Pantalones	35.000
3	Chaqueta	50.000
4	Zapatos	40.000
5	Sombrero	15.000

Between: Se utiliza para filtrar los resultados de una consulta para que solo incluya registros cuyo valor en una columna este dentro de un rango especificado.

Sintaxis

SELECT column_name(s)

FROM table_name

WHERE column_name BETWEEN value1 AND value2;

Ejemplo:

SELECT *

FROM Productos

WHERE Precio BETWEEN 20 AND 50;

ProductoId	Nombre	Precio
1	Camiseta	20.000
2	Pantalones	35.000
3	Chaqueta	50.000
4	Zapatos	40.000
5	Sombrero	15.000
6	Bufanda	25.000

=>

ProductoId	Nombre	Precio
1	Camiseta	20.000
2	Pantalones	35.000
3	Chaqueta	50.000
4	Zapatos	40.000
6	Bufanda	25.000