#CrecimientoProfesional
#AprendeConLosPioneros

Online | DMC Perú

DMC Perú | CAPACITACIÓN PROFESIONAL

SESIÓN III

Docente: Victor Gutierrez
Data Architect

Agenda

LENGUAJE TRANSACT SQL SENTENCIAS Y FUNCIONES

- ¿Qué es Transact SQL (Structured Query Language)?
- ¿Qué es Transact SQL?
- Categorías de las sentencias SQL.
- DDL Lenguaje de base de datos. y DML Lenguaje de manipulación de
- datos.
- Consultas de manipulación de datos.
- Consultas condicionales.
- Operadores lógicos.
- Funciones de agregación.
- Ejercicios prácticos.

Lenguaje Transact SQL / Operadores SQL

Operadores Aritméticos

Los operadores aritméticos realizan operaciones matemáticas con dos expresiones de uno o más de los tipos de datos de la categoría de tipos de datos numéricos. Para obtener más información sobre las categorías de tipos de datos, vea Convenciones de sintaxis de Transact-SQL.

Operador	Significado
+ (sumar)	Suma
- (restar)	Resta
* (multiplicar)	Multiplicación
/ (dividir)	División
% (Módulo)	Devuelve el resto entero de una división. Por ejemplo, 12 % 5 = 2 porque el resto de 12 dividido entre 5 es 2.

Lenguaje Transact SQL / Operadores SQL

Operadores de asignación

El signo igual (=) es el único operador de asignación de Transact-SQL. En el siguiente ejemplo se crea la variable @MyCounter y, a continuación, el operador de asignación define @MyCounter en un valor devuelto por una expresión.

```
DECLARE @MyCounter INT;
SET @MyCounter = 1;
```

Operadores de comparación

Los operadores de comparación comprueban si dos expresiones son iguales. Se pueden utilizar en todas las expresiones excepto en las de los tipos de datos text, ntext o image. En la siguiente tabla se presentan los operadores de comparación Transact-SQL.

Operador	Significado
= (Igual a)	Igual a
> (Mayor que)	Mayor que
< (Menor que)	Menor que
>= (Mayor o igual que)	Mayor o igual que
<= (Menor o igual que)	Menor o igual que
<> (No igual a)	No es igual a
!= (No es igual a)	No es igual a (no es del estándar ISO)
!< (No menor que)	No es menor que (no es del estándar ISO)
!> (No mayor que)	No es mayor que (no es del estándar ISO)

Lenguaje Transact SQL / Operadores SQL

Operadores Lógicos

Los operadores lógicos comprueban la veracidad de alguna condición. Éstos, como los operadores de comparación, devuelven el tipo de datos Boolean con el valor TRUE, FALSE o UNKNOWN.

Operador	Significado
ALL	TRUE si el conjunto completo de comparaciones es TRUE.
AND	TRUE si ambas expresiones booleanas son TRUE.
ANY	TRUE si cualquier miembro del conjunto de comparaciones es TRUE.
BETWEEN	TRUE si el operando está dentro de un intervalo.
EXISTS	TRUE si una subconsulta contiene cualquiera de las filas.
IN	TRUE si el operando es igual a uno de la lista de expresiones.
LIKE	TRUE si el operando coincide con un patrón.
NOT	Invierte el valor de cualquier otro operador booleano.
OR	TRUE si cualquiera de las dos expresiones booleanas es TRUE.
SOME	TRUE si alguna de las comparaciones de un conjunto es TRUE.

Operadores de Cadena

SQL Server proporciona los operadores de cadena siguientes. Los operadores de concatenación de cadenas pueden combinar dos o más cadenas o columnas de caracteres o binarias, o una combinación de cadenas y nombres de columna en una expresión. Los operadores de cadena de caracteres comodín pueden coincidir con uno o más caracteres en una operación de comparación de cadenas como LIKE y PATINDEX.

+ (Concatenación de cadenas)

+= (Concatenación de cadenas)

% (Comodín - Caracteres para coincidir)

[ ] (Comodín - Caracteres para coincidir)

[^] (Comodín - Caracteres para no coincidir)

_ (Comodín - Un carácter para coincidir)

Lenguaje Transact SQL / Data Manipulation Language

Lenguaje de Manipulación de Datos (DML)

También es un lenguaje proporcionado por los sistemas gestores de bases de datos. En inglés, Data Manipulation Language (DML). Utilizando instrucciones de SQL, permite a los usuarios introducir datos para posteriormente realizar tareas de consultas o modificación de los datos que contienen las Bases de Datos.
La selección sobre una tabla consiste en elegir un subconjunto de filas que cumplan (o no) algunas condiciones determinadas. La sintaxis de una sentencia de este tipo es la siguiente:

```
SELECT [*] columna1, columna2, ....
FROM nombre-tabla
[WHERE condición]
[GROUP BY columna1, columna2.... ]
[HAVING condición-selección-grupos ]
[ORDER BY columna1 [DESC], columna2
[DESC]... ]
```

***** indica que se obtienen todas las columnas de la/s tabla/s o vista/s indicadas en la cláusula FROM. Tambien puede optar por colocar el nombre de la columna de la tabla.

Lenguaje Transact SQL / Insert

Para insertar valores a una tabla usamos el comando INSERT, se requiere:

1. Nombre de la tabla a donde se insertan los valores
2. Identificar las columnas que reciben los valores
3. Asignar los valores para las columnas que se insertan

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

```
                  1                                2
INSERT INTO ventas (tienda, fecha, producto, cliente, importe)
VALUES                                   3
('Lima','01/01/2019','Papel Bond A4','Librería Lápiz y Papel',250);
```

Observe:

- El carácter que separa a las columnas y los valores es la coma ,
- Los campos de tipo texto inician y terminan con el apostrofe '
- Los valores de tipo numérico se escriben directamente.
- La sentencia termina con el carácter punto y coma ;

Lenguaje Transact SQL / Insert

Inserta registros de una tabla

Opción #1 Especificar las columnas en los valores y la tabla

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

Opción #2 Especificar las columnas en los valores

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

Opción #3 Insertar desde una consulta

```
INSERT INTO table_name
SELECT column1, column2, ...
FROM table_name
WHERE condition;
```

Lenguaje Transact SQL / Insert

Insertar Registros

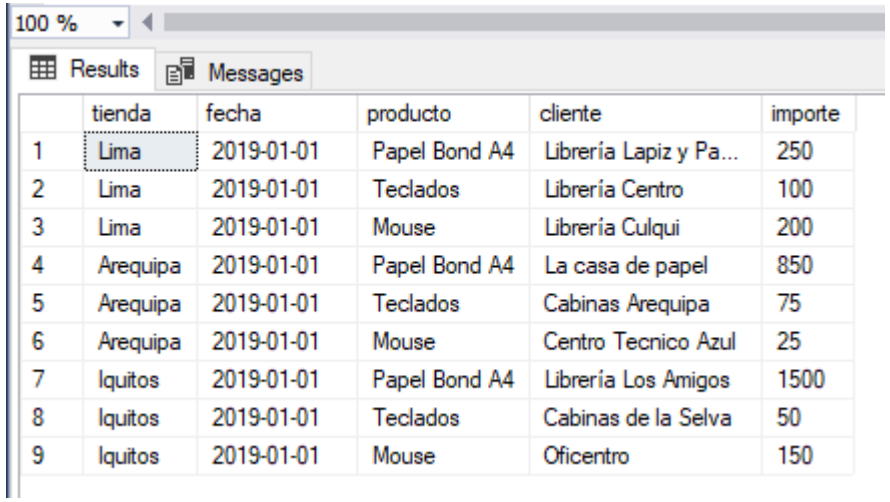
Ahora insertaremos los datos de ejemplo:

```
INSERT INTO ventas (tienda, fecha, producto, cliente, importe) VALUES ('Lima','01/01/2019','Papel Bond A4','Librería Lápiz y Papel',250);
INSERT INTO ventas (tienda, fecha, producto, cliente, importe) VALUES ('Lima','01/01/2019','Teclados','Librería Centro',100);
INSERT INTO ventas (tienda, fecha, producto, cliente, importe) VALUES ('Lima','01/01/2019','Mouse','Librería Culqui',200);
INSERT INTO ventas (tienda, fecha, producto, cliente, importe) VALUES ('Arequipa','01/01/2019','Papel Bond A4','La casa de papel',850);
INSERT INTO ventas (tienda, fecha, producto, cliente, importe) VALUES ('Arequipa','01/01/2019','Teclados','Cabinas Arequipa',75);
INSERT INTO ventas (tienda, fecha, producto, cliente, importe) VALUES ('Arequipa','01/01/2019','Mouse','Centro Técnico Azul',25);
INSERT INTO ventas (tienda, fecha, producto, cliente, importe) VALUES ('Iquitos','01/01/2019','Papel Bond A4','Librería Los Amigos',1500);
INSERT INTO ventas (tienda, fecha, producto, cliente, importe) VALUES ('Iquitos','01/01/2019','Teclados','Cabinas de la Selva',50);
INSERT INTO ventas (tienda, fecha, producto, cliente, importe) VALUES ('Iquitos','01/01/2019','Mouse','Oficentro',150);
```

Lenguaje Transact SQL / Select

Ahora utilizamos la sentencia SELECT para obtener los datos de la tabla.

```
SELECT * FROM ventas;
```



100 %

Results | Messages

	tienda	fecha	producto	cliente	importe
1	Lima	2019-01-01	Papel Bond A4	Librería Lapiz y Pa...	250
2	Lima	2019-01-01	Teclados	Librería Centro	100
3	Lima	2019-01-01	Mouse	Librería Culqui	200
4	Arequipa	2019-01-01	Papel Bond A4	La casa de papel	850
5	Arequipa	2019-01-01	Teclados	Cabinas Arequipa	75
6	Arequipa	2019-01-01	Mouse	Centro Tecnico Azul	25
7	Iquitos	2019-01-01	Papel Bond A4	Librería Los Amigos	1500
8	Iquitos	2019-01-01	Teclados	Cabinas de la Selva	50
9	Iquitos	2019-01-01	Mouse	Oficentro	150

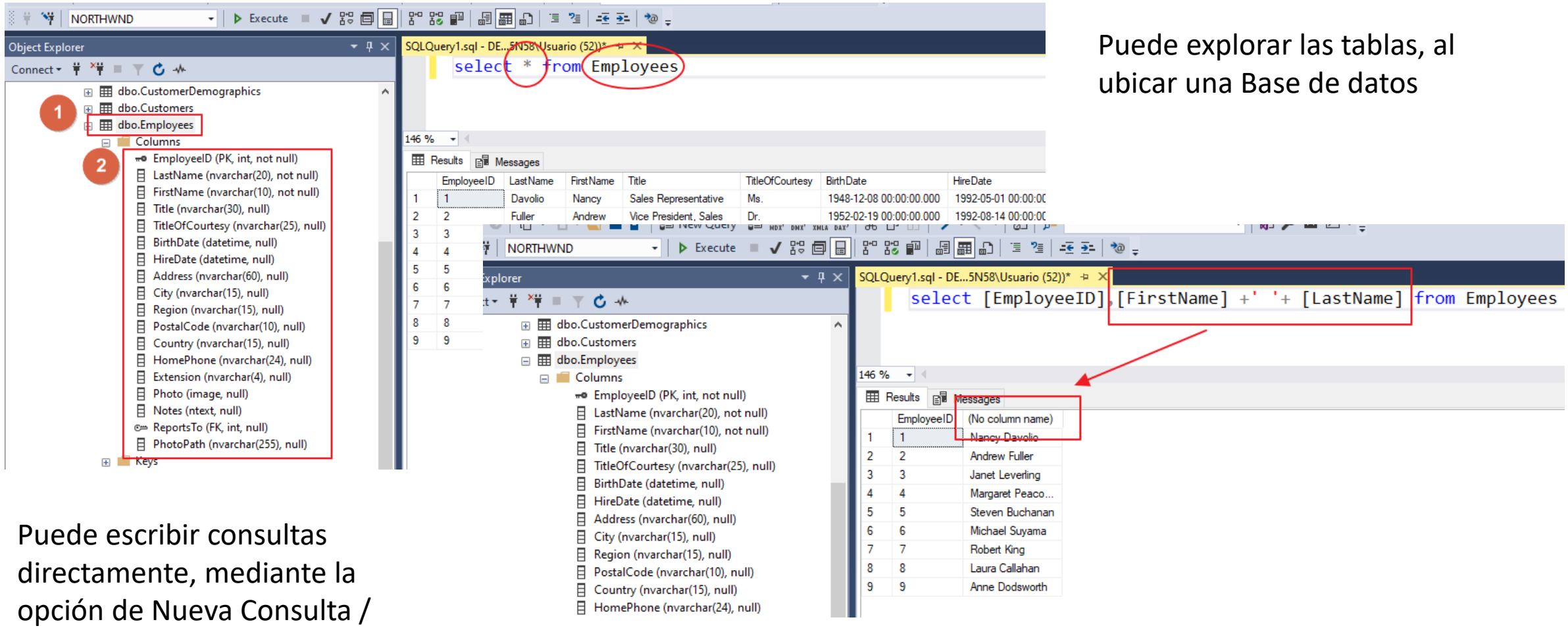
Sintaxis:

```
SELECT column1, column2, ...
FROM table_name;
WHERE condicion
```

Observe:

- El carácter asterisco reemplaza a todas las columnas de la tabla
- Se muestra el numero de registro que devuelve la consulta

Lenguaje Transact SQL / Select



Puede explorar las tablas, al ubicar una Base de datos

Puede escribir consultas directamente, mediante la opción de Nueva Consulta / New Query

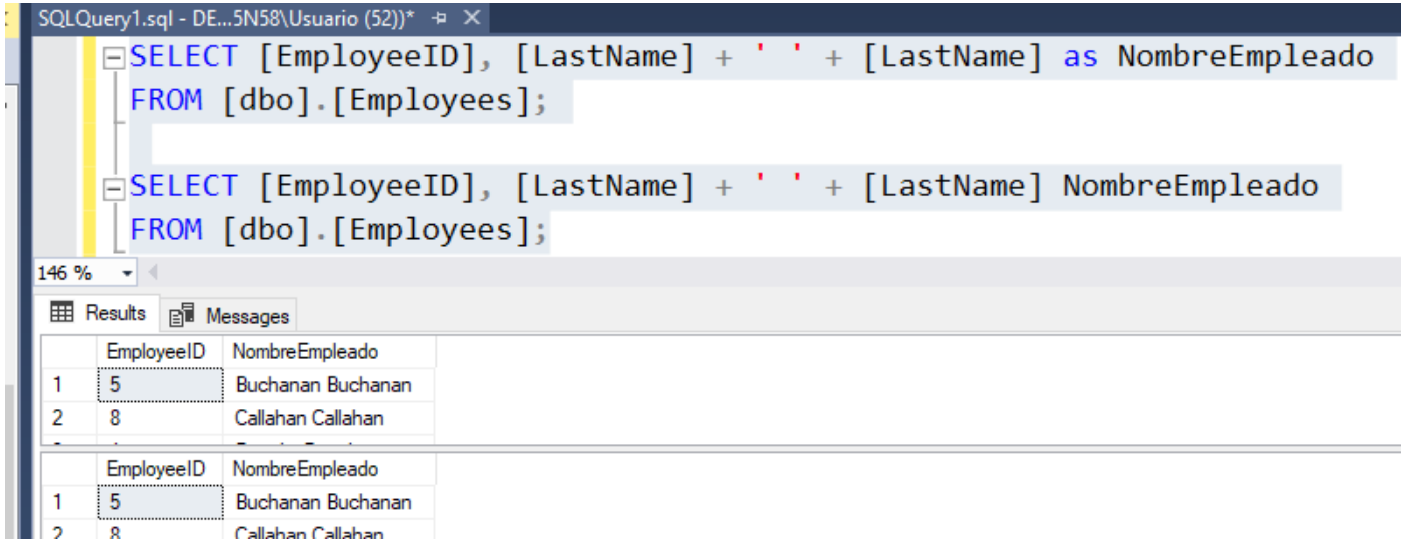
Lenguaje Transact SQL / Select

Alias de columna

Una manera de hacer más comprensible el resultado de una consulta consiste en cambiar los encabezados de las columnas para esto se usa la expresión AS o un espacio en blanco seguido de el nombre alias de columna.

```
SELECT [EmployeeID], [LastName] + ' ' + [LastName] as NombreEmpleado
FROM [dbo].[Employees];

SELECT [EmployeeID], [LastName] + ' ' + [LastName] NombreEmpleado
FROM [dbo].[Employees];
```



Lenguaje Transact SQL / Select

Columnas Calculadas

Se llama así a las columnas creadas a partir de uso de operadores y que muestran una columna con el resultado de una operación aritmética

```
SELECT [OrderID]
,[ProductID]
,[UnitPrice]
,[Quantity]
,[UnitPrice] * [Quantity]
  FROM [dbo].[Order Details]
```

Results | Messages

	OrderID	ProductID	UnitPrice	Quantity	(No column name)
1	10248	11	14,00	12	168,00
2	10248	42	9,80	10	98,00
3	10248	72	34,80	5	174,00
4	10249	14	18,60	9	167,40
5	10249	51	42,40	40	1696,00
6	10250	41	7,70	10	77,00
7	10250	51	42,40	35	1484,00
8	10250	65	16,80	15	252,00
9	10251	22	16,80	6	100,80
10	10251	57	15,60	15	234,00

Lenguaje Transact SQL / Select

Columnas Calculadas

Se llama así a las columnas creadas a partir de uso de operadores y que muestran una columna con el resultado de una operación aritmética

```
SELECT [EmployeeID]
,concat ([LastName],' ', [FirstName])
,[Title]
,[Country]
  FROM [dbo].[Employees]
```

	EmployeeID	(No column name)	Title	Country
1	1	Davolio Nancy	Sales Representative	USA
2	2	Fuller Andrew	Vice President, Sales	USA
3	3	Leverling Janet	Sales Representative	USA
4	4	Peacock Marga...	Sales Representative	USA
5	5	Buchanan Steven	Sales Manager	UK
6	6	Suyama Michael	Sales Representative	UK
7	7	King Robert	Sales Representative	UK
8	8	Callahan Laura	Inside Sales Coordina...	USA
9	9	Dodsworth Anne	Sales Representative	UK

Lenguaje Transact SQL / Select (Order by)

Hasta ahora, hemos visto cómo obtener datos de una tabla utilizando los comandos SELECT y WHERE. Con frecuencia, sin embargo, necesitamos enumerar el resultado en un orden particular. Esto podría ser en orden ascendente, en orden descendente, o podría basarse en valores numéricos o de texto. En tales casos, podemos utilizar la palabra clave ORDER BY para alcanzar nuestra meta.
La sintaxis para una instrucción ORDER BY es la siguiente:

```
SELECT nombre_columna
FROM nombre_tabla
WHERE condición
ORDER BY nombre_columna [ASC, DESC];
```

[ ] significa que la instrucción WHERE es opcional. Sin embargo, si existe una cláusula WHERE, viene antes de la cláusula ORDER BY ASC significa que los resultados se mostrarán en orden ascendente, y DESC significa que los resultados se mostrarán en orden descendente. Si no se especifica ninguno, la configuración predeterminada es ASC.
Es posible ordenar por más de una columna. En este caso, la cláusula ORDER BY anterior se convierte en:

Lenguaje Transact SQL / Update

Actualizar registros a una tabla

Opción #1 Especificar las columnas en los valores y la tabla

```
UPDATE table_name
SET column1 = value1, column2 = value2, ...
WHERE condition;
```

Opción #2 Utilizando una Sub Consulta

```
UPDATE  tempDataView
SET     marks =
        (
          SELECT marks
          FROM tempData b
          WHERE tempDataView.Name = b.Name
        )
```

Lenguaje Transact SQL / Where

Especifica la condición de filtro de las filas devueltas. Se utiliza cuando no se desea que se devuelvan todas las filas de una tabla, sino sólo las que cumplen ciertas condiciones. Lo habitual es utilizar esta cláusula en la mayoría de las consultas.

```
SELECT [EmployeeID]
,concat ([LastName],' ', [FirstName])
,[Title]
,[Country]
FROM [dbo].[Employees]
  WHERE [Country] = 'USA'

SELECT [OrderID]
,[ProductID]
,[UnitPrice]
,[Quantity]
,[UnitPrice] * [Quantity]
FROM [dbo].[Order Details]
where [ProductID] = 42
and [UnitPrice] * [Quantity] > 300
```

	EmployeeID	(No column name)	Title	Country
1	1	Davolio Nancy	Sales Representative	USA
2	2	Fuller Andrew	Vice President, Sales	USA
3	3	Leverling Janet	Sales Representative	USA
4	4	Peacock Marga...	Sales Representative	USA
5	8	Callahan Laura	Inside Sales Coordina...	USA

	OrderID	ProductID	UnitPrice	Quantity	(No column name)
1	10404	42	11,20	40	448,00
2	10463	42	11,20	50	560,00
3	10498	42	14,00	30	420,00
4	10571	42	14,00	28	392,00
5	10588	42	14,00	100	1400,00
6	10663	42	14,00	30	420,00
7	10680	42	14,00	40	560,00
8	10746	42	14,00	28	392,00

Lenguaje Transact SQL / Like / Between

LIKE:

Usada para la comparación de un modelo. Para ello utiliza los caracteres comodín especiales: “%” y “_”. Con el primero indicamos que en su lugar puede ir cualquier cadena de caracteres, y con el segundo que puede ir cualquier carácter individual (un solo carácter). Con la combinación de estos caracteres podremos obtener múltiples patrones de búsqueda. Por ejemplo:

El nombre empieza por A: Nombre LIKE ‘A%’

El nombre acaba por A: Nombre LIKE ‘%A’

El nombre contiene la letra A: Nombre LIKE ‘%A%’

El nombre empieza por A y después contiene un solo carácter cualquiera: Nombre LIKE ‘A_’

El nombre empieza una A, después cualquier carácter, luego una E y al final cualquier cadena de caracteres: Nombre LIKE ‘A_E%’

BETWEEN: para un intervalo de valores. Por ejemplo:

Clientes entre el 30 y el 100: CodCliente BETWEEN 30 AND 100

Clientes nacidos entre 1970 y 1979: FechaNac BETWEEN ‘19700101’ AND ‘19791231’

Lenguaje Transact SQL / IN

IN( )

Para especificar una relación de valores concretos.
Por ejemplo: Ventas de los Clientes 10, 15, 30 y 75:
CodCliente **IN** (10, 15, 30, 75)

Por supuesto es posible combinar varias condiciones simples de los operadores anteriores utilizando los operadores lógicos **OR**, **AND** y **NOT**, así como el uso de paréntesis para controlar la prioridad de los operadores (como en matemáticas).

Lenguaje Transact SQL / Case

La instrucción CASE pasa por condiciones y devuelve un valor cuando se cumple la primera condición. Entonces, una vez que una condición es verdadera, dejará de leer y devolverá el resultado. Si no hay condiciones verdaderas, devuelve el valor en la cláusula ELSE.

Si no hay otra parte y no hay condiciones verdaderas, devuelve NULL.

```
CASE
    WHEN condition1 THEN result1
    WHEN condition2 THEN result2
    WHEN conditionN THEN resultN
    ELSE result
END;
```

```
SELECT OrderID, Quantity,
CASE
    WHEN Quantity > 30 THEN 'The quantity is greater than 30'
    WHEN Quantity = 30 THEN 'The quantity is 30'
    ELSE 'The quantity is under 30'
END AS QuantityText
FROM [dbo].[Order Details];
```

	OrderID	Quantity	QuantityText
1	10248	12	The quantity is under 30
2	10248	10	The quantity is under 30
3	10248	5	The quantity is under 30
4	10249	9	The quantity is under 30
5	10249	40	The quantity is greater than 30

Lenguaje Transact SQL / Funciones de Agrupación

Las funciones de grupo operan en conjuntos de filas para dar un resultado por grupo

- AVG
- COUNT
- MAX
- MIN
- SUM

	DEPARTMENT_ID	SALARY
1	10	4400
2	20	13000
3	20	6000
4	110	12000
5	110	8300
6	90	24000
7	90	17000
8	90	17000
9	60	9000
10	60	6000
...		
18	80	11000
19	80	8600
20	(null)	7000

Maximum salary in EMPLOYEES table

MAX(SALARY)
24000

Lenguaje Transact SQL / Funciones de Agrupación

Syntaxis:

```
SELECT      group_function(column), ...
FROM        table
[WHERE      condition]
[ORDER BY   column];
```

Function	Description
AVG(exp)	Average value of n, ignoring null values
COUNT({*|[DISTINCT|ALL]expr })	Number of rows, where expr evaluates to something other than null (count all selected rows using *, including duplicates and rows with nulls)
MAX(expr)	Maximum value of expr, ignoring null values
MIN(expr)	Minimum value of expr, ignoring null values
SUM(n)	Sum values of n, ignoring null values

Lenguaje Transact SQL / Funciones de Agrupación

Puede usar **AVG** y **SUM** para datos numéricos.

```
SELECT  AVG(salary), MAX(salary),
        MIN(salary), SUM(salary)
FROM    employees
WHERE   job_id LIKE '%REP%';
```

	AVG(SALARY)	MAX(SALARY)	MIN(SALARY)	SUM(SALARY)
1	8150	11000	6000	32600

Puede usar **MIN** y **MAX** para los tipos de datos numéricos, de caracteres y de fecha

```
SELECT  MIN(hire_date), MAX(hire_date)
FROM        employees;
```

	MIN(HIRE_DATE)	MAX(HIRE_DATE)
1	17-JUN-87	29-JAN-00

Lenguaje Transact SQL / Funciones de Agrupación

Usando la función COUNT

- COUNT (*)
- COUNT (expr)
- COUNT (DISTINCT expr)

COUNT (*) devuelve el número de filas en una tabla que satisfacen los criterios de la declaración SELECT, incluidas las filas duplicadas y las filas que contienen valores nulos en cualquiera de las columnas. Si se incluye una cláusula WHERE devuelve el número de filas que satisfacen la condición

COUNT (expr) devuelve el número de valores no nulos que están en la columna identificada por expr.

COUNT (DISTINCT expr) devuelve el número de valores únicos no nulos que están en la columna identificada por expr.

`COUNT (*)` returns the number of rows in a table:

1

```
SELECT COUNT(*)
FROM   employees
WHERE  department_id = 50;
```

	COUNT(*)
1	5

`COUNT (expr)` returns the number of rows with non-null values for *expr*:

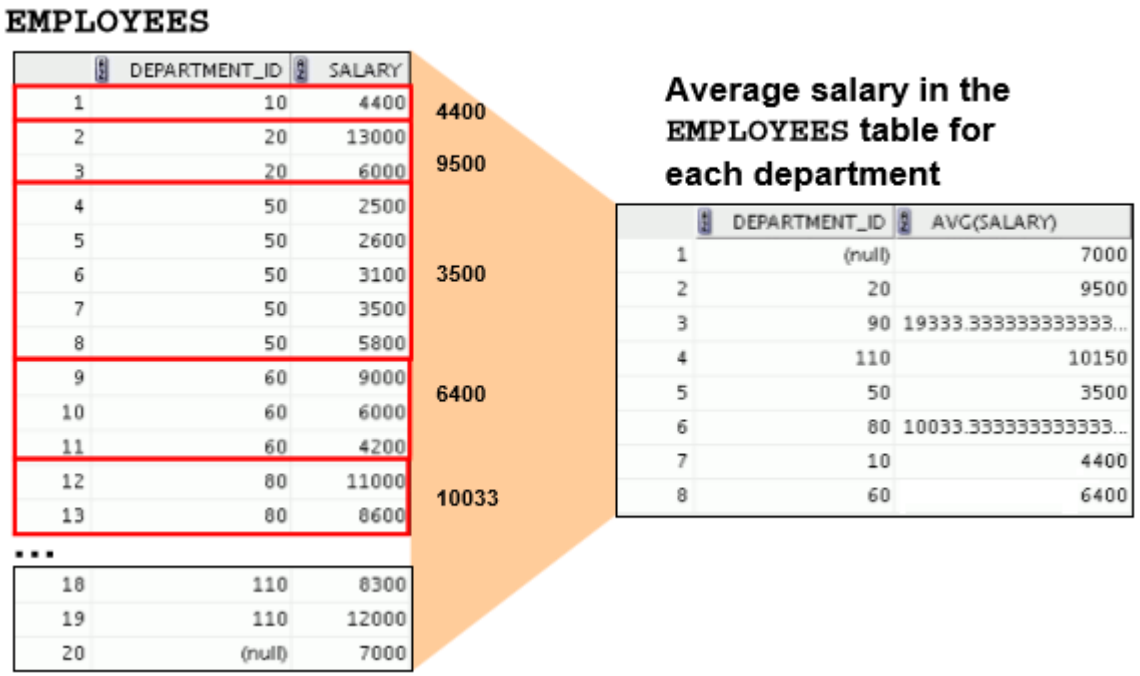
2

```
SELECT COUNT(commission_pct)
FROM   employees
WHERE  department_id = 80;
```

	COUNT(COMMISSION_PCT)
1	3

Lenguaje Transact SQL / Group by

- ✓ Puede dividir las filas de una tabla en grupos más pequeños utilizando la cláusula GROUP BY.
- ✓ Las columnas en el SELECT deben coincidir con la clausula
- ✓ La clausula WHERE excluye los registros antes de la agrupación.
- ✓ No se puede usar alias en el GROUP BY
- ✓ Solo se puede excluir las columnas del SELECT cuando aplica a todo el resultado
- ✓ Puede agrupar mas de una columna de diferentes tablas.




```
SELECT      column, group_function(column)
FROM        table
[WHERE      condition]
[GROUP BY group_by_expression]
[ORDER BY column];
```

Lenguaje Transact SQL / Having

Restringir los resultados usando la clausula Having

Cuando usa la cláusula **having**, el servidor restringe los grupos de la siguiente manera:

- Las filas están agrupadas
- Se aplica la función de grupo.
- Se muestran los grupos que coinciden con la cláusula **having**

```
SELECT     column, group_function
FROM       table
[WHERE     condition]
[GROUP BY  group_by_expression]
[HAVING    group_condition]
[ORDER BY  column];
```

```
SELECT     department_id, MAX(salary)
FROM       employees
GROUP BY   department_id
HAVING     MAX(salary)>10000 ;
```

	DEPARTMENT_ID	MAX(SALARY)
1	20	13000
2	90	24000
3	110	12000
4	80	11000

REFERENCIAS

Insert

https://docs.microsoft.com/es-es/sql/t-sql/statements/insert-transact-sql?view=sql-server-ver15

Update

https://docs.microsoft.com/es-es/sql/t-sql/queries/update-transact-sql?view=sql-server-ver15

Where

https://docs.microsoft.com/es-es/sql/t-sql/queries/where-transact-sql?view=sql-server-ver15

Funciones de Agregación

https://docs.microsoft.com/es-es/sql/t-sql/functions/aggregate-functions-transact-sql?view=sql-server-ver15

DIIC Perú | CAPACITACIÓN **PROFESIONAL**

Online | DIIC Perú