UNIVERSIDAD PERUANA DE LOS ANDES

FACULTAD DE INGENIERÍA

ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS Y COMPUTACIÓN

FUNCIONES DE AGREGACION

ESTUDIANTE: CARRILLO CÁRDENAS JOSÉ ESTEBAN

ASIGNATURA: BASE DE DATOS II

DOCENTE: RAUL FERNANDEZ, BEJARANO

CICLO: V

HUANCAYO – 2025

-Explicar de manera clara y didácticas que son las funciones de agregación en SQL y como se utilizan

¿Qué son las funciones de agregación en SQL?

Las funciones de agregación son funciones especiales que realizan cálculos sobre un conjunto de filas (registros) y devuelven un solo valor como resultado.
Es decir, resumen datos.

Por ejemplo:

- ¿Cuál es el precio promedio?
- ¿Cuántos productos hay en total?
- ¿Cuál es el valor máximo o mínimo?

Estas preguntas se responden con funciones de agregación.

Principales funciones de agregación

Aquí tienes las más usadas, con ejemplos sencillos 👇

Función	Descripción	Ejemplo
COUNT()	Cuenta el número de registros.	COUNT(*) → cuenta todas las filas.
SUM()	Suma los valores de una columna numérica.	SUM(PrecioVenta) → suma los precios.
AVG()	Calcula el promedio.	AVG(PrecioVenta) → promedio del precio.
MAX()	Devuelve el valor máximo.	MAX(PrecioVenta) → precio más alto.
MIN()	Devuelve el valor mínimo.	MIN(PrecioVenta) → precio más bajo.

Ejemplo práctico

Supongamos que tenemos una tabla llamada GUIA_DETALLE:

NumGuia	CodArticulo	PrecioVenta	CantidadEnviada
2051	4	1900.00	20
2052	5	1650.00	15
2053	6	210.00	50
2054	7	35.00	80

Ejemplo 1: Sumar el total vendido

SELECT SUM(PrecioVenta) AS TotalVendido

FROM GUIA_DETALLE;

Resultado:

TotalVendido
3795.00

Explicación:

SUM() suma todos los valores de la columna PrecioVenta.

Ejemplo 2: Calcular el precio promedio

SELECT AVG(PrecioVenta) AS PrecioPromedio

FROM GUIA_DETALLE;

Resultado:

PrecioPromedio
948.75

Explicación:

AVG() calcula el promedio de todos los valores numéricos de PrecioVenta.

Ejemplo 3: Encontrar el precio más alto y más bajo

SELECT MAX(PrecioVenta) AS PrecioMaximo,

MIN(PrecioVenta) AS PrecioMinimo

FROM GUIA_DETALLE;

Resultado:

PrecioMaximo	PrecioMinimo
1900.00	35.00

Ejemplo 4: Contar cuántos artículos hay

SELECT COUNT(*) AS TotalArticulos

FROM GUIA_DETALLE;

Resultado:

TotalArticulos
4

Uso combinado con GROUP BY

Hasta ahora, todas las funciones devolvían un solo resultado general.
Pero si quieres ver resultados por grupo (por ejemplo, por artículo o por guía), usas GROUP BY.

Ejemplo 5: Total vendido por cada artículo

SELECT CodArticulo, SUM(PrecioVenta) AS TotalPorArticulo

FROM GUIA_DETALLE

GROUP BY CodArticulo;

Resultado:

CodArticulo	TotalPorArticulo
4	1900.00
5	1650.00
6	210.00
7	35.00

Explicación:

GROUP BY agrupa los registros por CodArticulo, y SUM() calcula la suma de cada grupo.

REALIZAR LAS CONSULTAS

1.Mostrar CodArticulos, DescripcionAArticulos, y ValorInventario

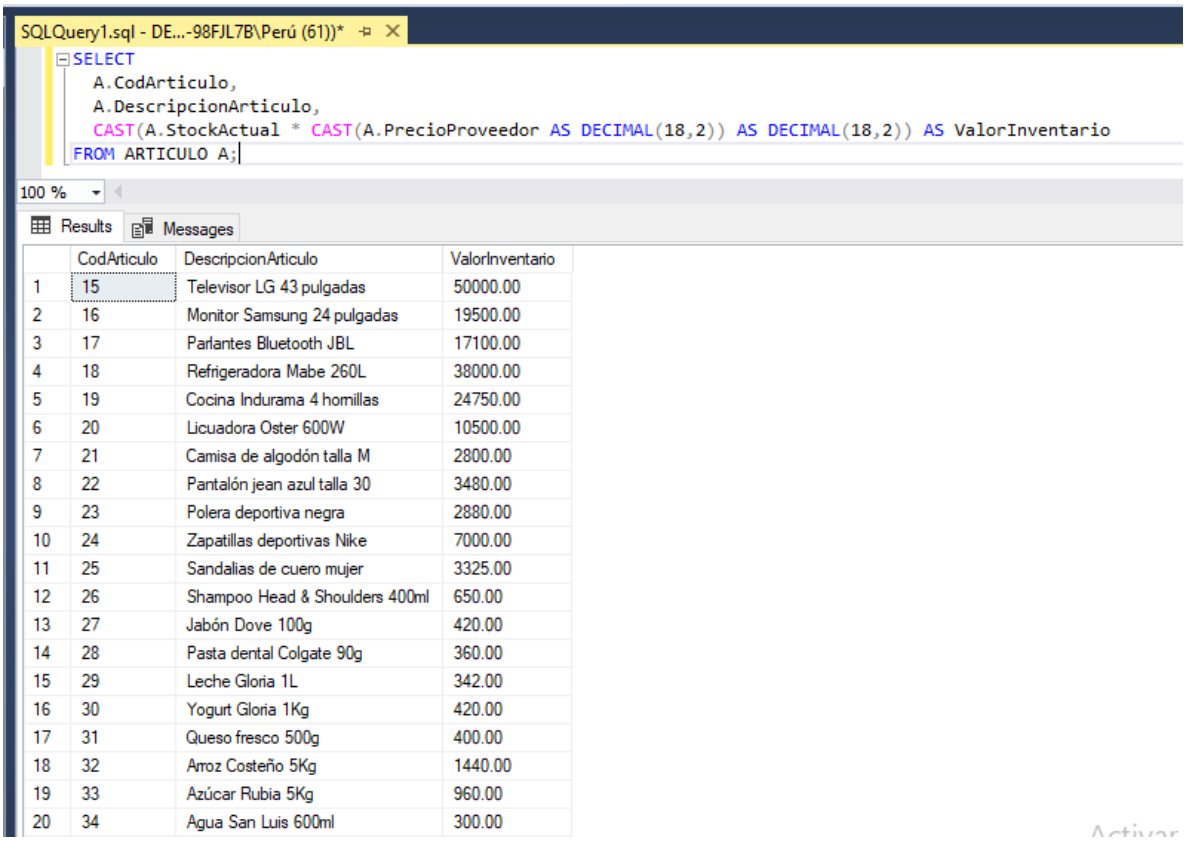
SELECT

A. CodArticulo,

A. DescripcionArticulo,

CAST (A.StockActual * CAST(A. PrecioProveedor AS DECIMAL(18,2)) AS DECIMAL(18,2)) AS ValorInventario

FROM ARTICULO A;



	CodArticulo	DescripcionArticulo	ValorInventario
1	15	Televisor LG 43 pulgadas	50000.00
2	16	Monitor Samsung 24 pulgadas	19500.00
3	17	Parlantes Bluetooth JBL	17100.00
4	18	Refrigeradora Mabe 260L	38000.00
5	19	Cocina Indurama 4 hornillas	24750.00
6	20	Licuadora Oster 600W	10500.00
7	21	Camisa de algodón talla M	2800.00
8	22	Pantalón jean azul talla 30	3480.00
9	23	Polera deportiva negra	2880.00
10	24	Zapatillas deportivas Nike	7000.00
11	25	Sandalias de cuero mujer	3325.00
12	26	Shampoo Head & Shoulders 400ml	650.00
13	27	Jabón Dove 100g	420.00
14	28	Pasta dental Colgate 90g	360.00
15	29	Leche Gloria 1L	342.00
16	30	Yogurt Gloria 1Kg	420.00
17	31	Queso fresco 500g	400.00
18	32	Arroz Costeño 5Kg	1440.00
19	33	Azúcar Rubia 5Kg	960.00
20	34	Agua San Luis 600ml	300.00

21	35	Gaseosa Inca Kola 2L	624.00
22	36	Resma papel bond A4 80gr	1320.00
23	37	Tóner HP 12A	2700.00
24	38	Detergente Ariel 1Kg	840.00
25	39	Suavizante Downy 800ml	650.00
26	40	Papel higiénico Elite 4 rollos	675.00
27	41	Toalla Nova 2 rollos	360.00
28	42	Martillo 16oz mango madera	960.00
29	43	Destornillador plano 5"	550.00
30	44	Taladro Bosch 750W	5600.00
31	45	Sierra circular Stanley	4500.00
32	46	Laptop HP Pavilion 14"	27600.00
33	47	Mouse Logitech M90	1000.00
34	48	Impresora Epson L3250	14700.00
35	49	Monitor Dell 22" LED	18000.00
36	50	Silla de oficina ergonómica	6200.00
37	51	Escritorio de melamina 1.2m	7000.00
38	52	Sofá de 3 plazas	12000.00
39	53	Mesa de centro madera	6750.00
40	54	Reloj Casio analógico	3500.00
41	55	Reloj digital deportivo	3600.00
42	56	Collar de plata 925	6300.00
43	57	Anillo de oro 18K	6000.00
44	58	Pelota de fútbol Adidas	3850.00
45	59	Bicicleta montaña aro 29	10680.00
46	60	Casco de seguridad 3M	2100.00
47	61	Guantes de látex talla M	2240.00
48	62	Lámpara LED 12W	1260.00
49	63	Bombilla ahorradora 9W	1200.00
50	64	Extensión eléctrica 3m	1000.00

Explicación

- selecciona tres columnas de la tabla **ARTICULO**:
  - CodArticulo: el código identificador del artículo.
  - DescripcionArticulo: el nombre o descripción del artículo.
  - ValorInventario: el valor total del inventario de ese artículo, que se calcula multiplicando la cantidad en stock (StockActual) por el precio del proveedor (PrecioProveedor).

Por qué usar CAST:

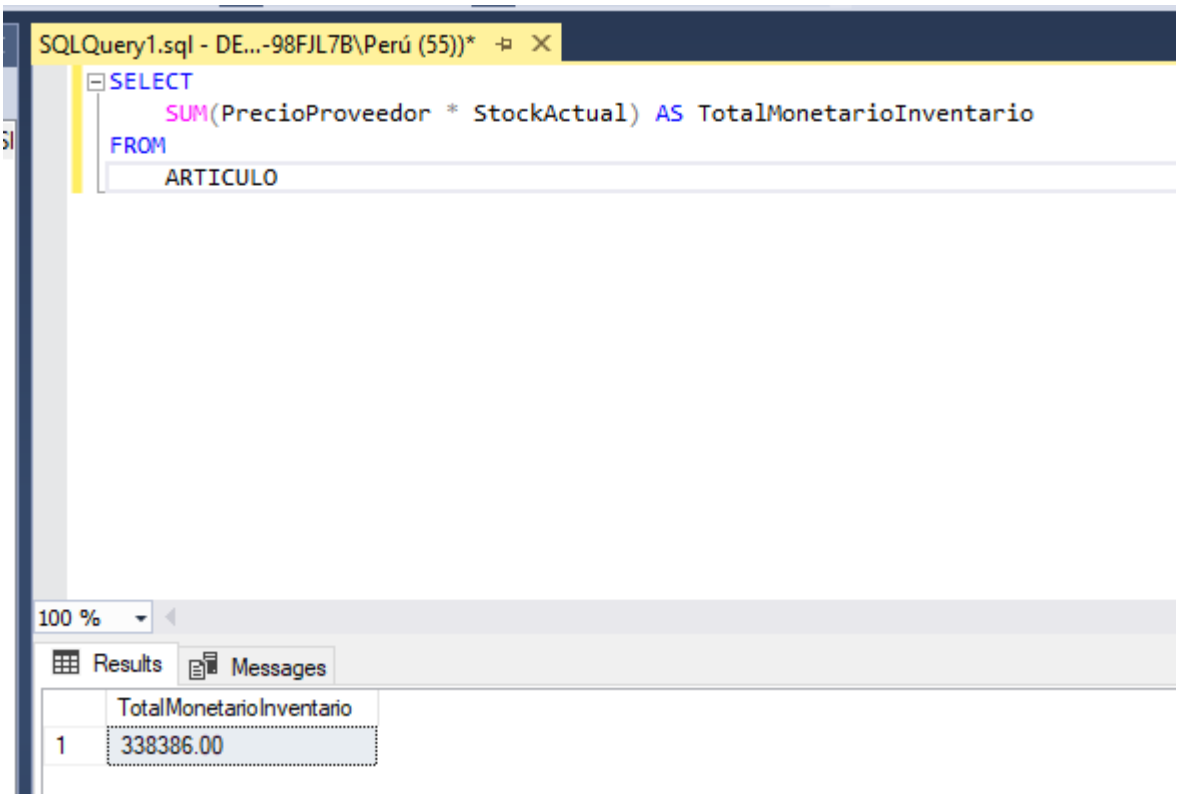
- Se usa CAST para convertir el tipo de datos de PrecioProveedor y del resultado de la multiplicación a **DECIMAL (18,2)**.
- Esto asegura que el resultado tenga un formato numérico con dos decimales, ideal para valores monetarios.

Resultado esperado:

- Te devuelve una tabla con:
  - El código del artículo,
  - La descripción del artículo,
  - Y el valor en dinero del inventario que tienes para cada artículo, calculado multiplicando stock por precio.

2.Calcular el total monetario del inventario

```
SELECT
    SUM(PrecioProveedor * StockActual) AS TotalMonetarioInventario
FROM
    ARTICULO
```

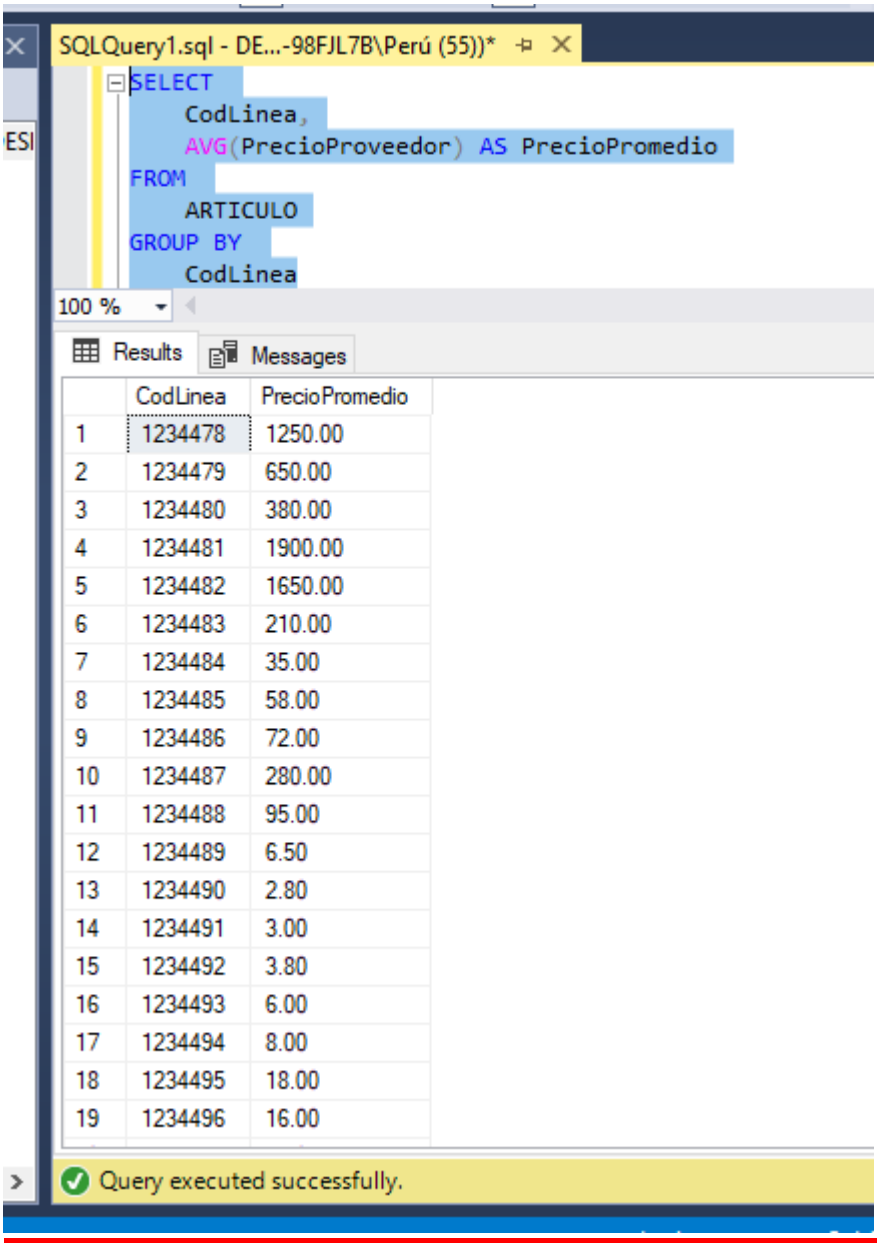


Explicación:

- `PrecioProveedor`: es el precio unitario de compra del producto.
- `StockActual`: indica cuántas unidades hay en inventario.
- `SUM(PrecioProveedor * StockActual)`: multiplica el precio por la cantidad y suma el total de todos los artículos.
- El resultado (`TotalMonetarioInventario`) es **el valor total del inventario en moneda local** (por ejemplo, soles).

3.Obtener CodLinea y PrecioProveedor promedio.

```
SELECT
    CodLinea,
    AVG(PrecioProveedor) AS PrecioPromedio
FROM
    ARTICULO
GROUP BY
    CodLinea
```


SQLQuery1.sql - DE...-98FJL7B\Perú (55))*
SELECT
CodLinea,
AVG(PrecioProveedor) AS PrecioPromedio
FROM
ARTICULO
GROUP BY
CodLinea
100 %
Results
Messages
CodLinea PrecioPromedio
1 1234478 1250.00
2 1234479 650.00
3 1234480 380.00
4 1234481 1900.00
5 1234482 1650.00
6 1234483 210.00
7 1234484 35.00
8 1234485 58.00
9 1234486 72.00
10 1234487 280.00
11 1234488 95.00
12 1234489 6.50
13 1234490 2.80
14 1234491 3.00
15 1234492 3.80
16 1234493 6.00
17 1234494 8.00
18 1234495 18.00
19 1234496 16.00
Query executed successfully.

20	1234497	1.50
21	1234498	4.80
22	1234499	22.00
23	1234500	180.00
24	1234501	7.00
25	1234502	6.50
26	1234503	4.50
27	1234504	3.00
28	1234505	12.00
29	1234506	5.50
30	1234507	280.00
31	1234508	450.00
32	1234509	2300.00
33	1234510	25.00
34	1234511	980.00
35	1234512	720.00
36	1234513	310.00
37	1234514	280.00

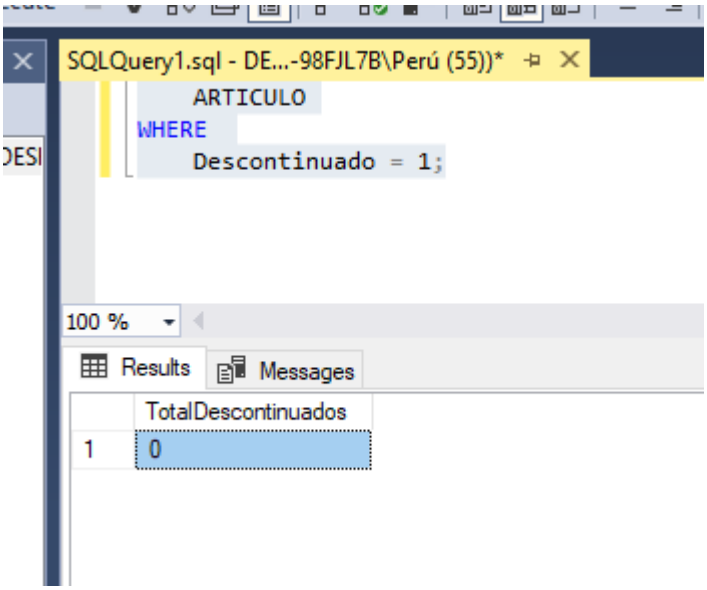
38	1234515	1200.00
39	1234516	450.00
40	1234517	140.00
41	1234518	180.00
42	1234519	210.00
43	1234520	750.00
44	1234521	110.00
45	1234522	890.00
46	1234523	35.00
47	1234524	28.00
48	1234525	18.00
49	1234526	12.00
50	1234527	25.00

Explicación:

- `CodLinea`: identifica a qué línea pertenece cada producto (por ejemplo, Electrónica, Ropa, Hogar, etc.).
- `AVG(PrecioProveedor)`: calcula el **precio promedio de los artículos** de esa línea.
- `GROUP BY CodLinea`: agrupa todos los registros que pertenecen a la misma línea para calcular su promedio.

4.Contar artículos descontinuados

```sql
SELECT
    COUNT(*) AS TotalDescontinuados
FROM
    ARTICULO
WHERE
    Descontinuado = 1;
```



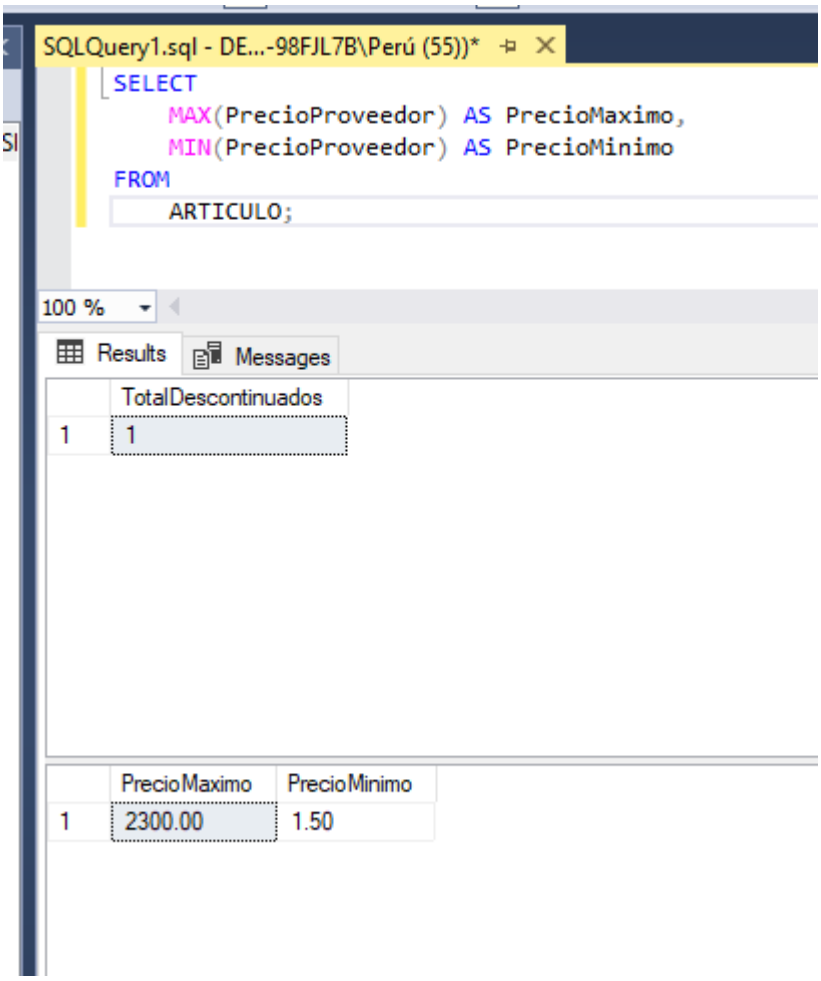
Explicación:

- `COUNT(*)`: cuenta el número total de registros que cumplen la condición.
- `WHERE Descontinuado = 1`: filtra solo los artículos marcados como **descontinuados**.
- El resultado (`TotalDescontinuados`) te mostrará **cuántos productos ya no están en venta o vigentes.**

5.Mostrar precio máximo y preciominimo del catalogo

```sql
SELECT
    MAX(PrecioProveedor) AS PrecioMaximo,
    MIN(PrecioProveedor) AS PrecioMinimo
FROM

    ARTICULO;
```

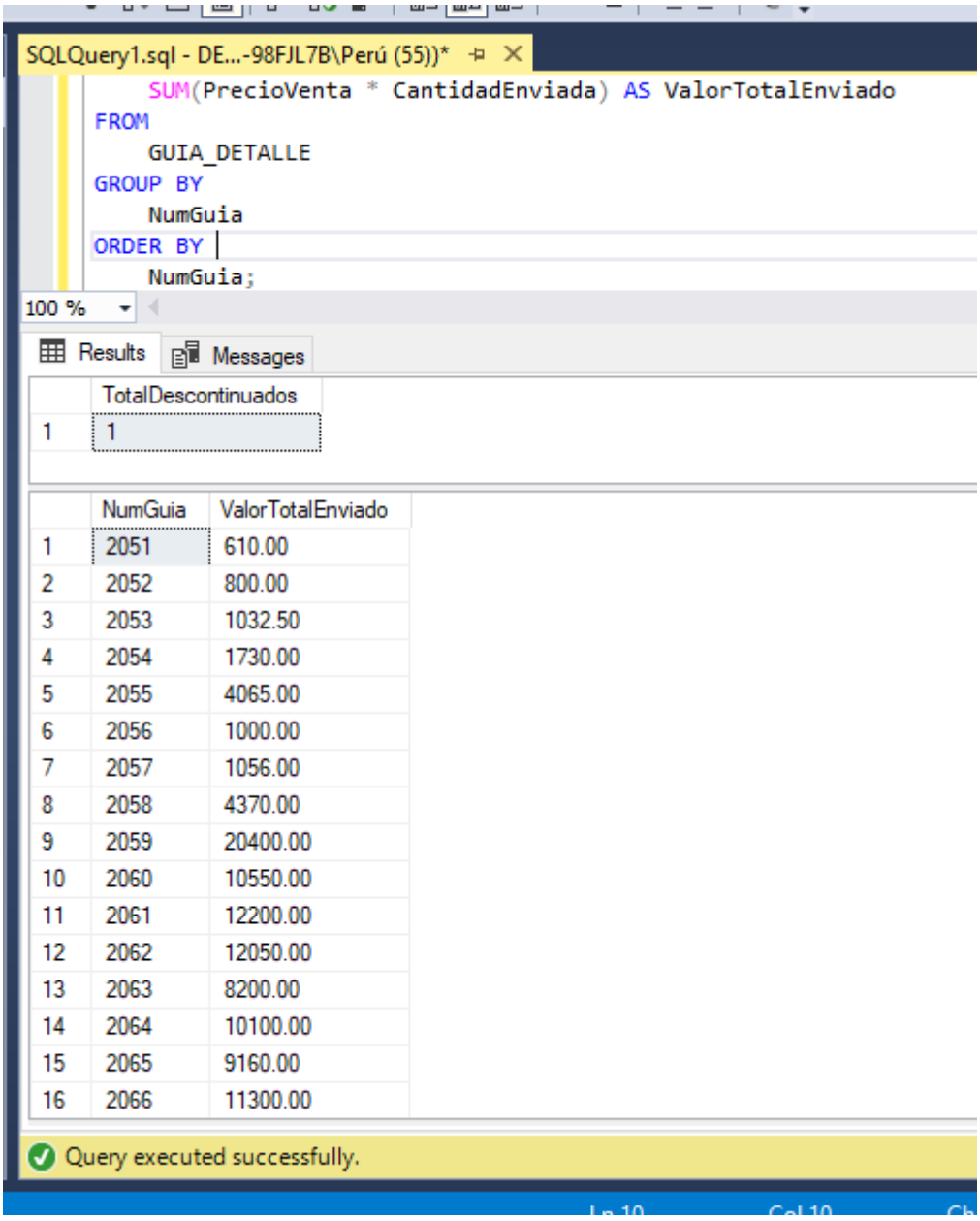


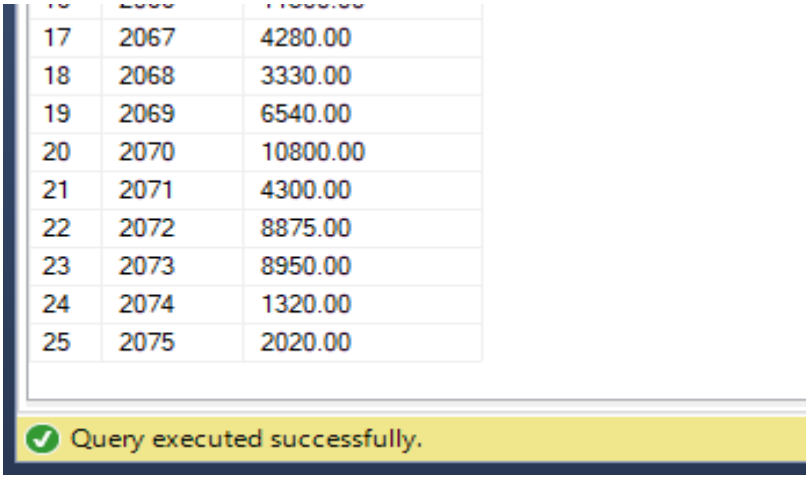
Explicación:

- `MAX(PrecioProveedor)`: devuelve el **precio más alto** registrado en la tabla `ARTICULO`.
- `MIN(PrecioProveedor)`: devuelve el **precio más bajo**.
- El resultado mostrará **un solo registro** con ambos valores.

6.Mostrar el valor total enviado por guía

```
    SUM(PrecioVenta * CantidadEnviada) AS ValorTotalEnviado
FROM
    GUIA_DETALLE
GROUP BY
    NumGuia
ORDER BY
    NumGuia;
```


SQLQuery1.sql - DE...-98FJL7B\Perú (55))*
SUM(PrecioVenta * CantidadEnviada) AS ValorTotalEnviado
FROM
GUIA_DETALLE
GROUP BY
NumGuia
ORDER BY
NumGuia;
100 %
Results
Messages
TotalDescontinuados
1
1
NumGuia ValorTotalEnviado
1 2051 610.00
2 2052 800.00
3 2053 1032.50
4 2054 1730.00
5 2055 4065.00
6 2056 1000.00
7 2057 1056.00
8 2058 4370.00
9 2059 20400.00
10 2060 10550.00
11 2061 12200.00
12 2062 12050.00
13 2063 8200.00
14 2064 10100.00
15 2065 9160.00
16 2066 11300.00
Query executed successfully.



17	2067	4280.00
18	2068	3330.00
19	2069	6540.00
20	2070	10800.00
21	2071	4300.00
22	2072	8875.00
23	2073	8950.00
24	2074	1320.00
25	2075	2020.00

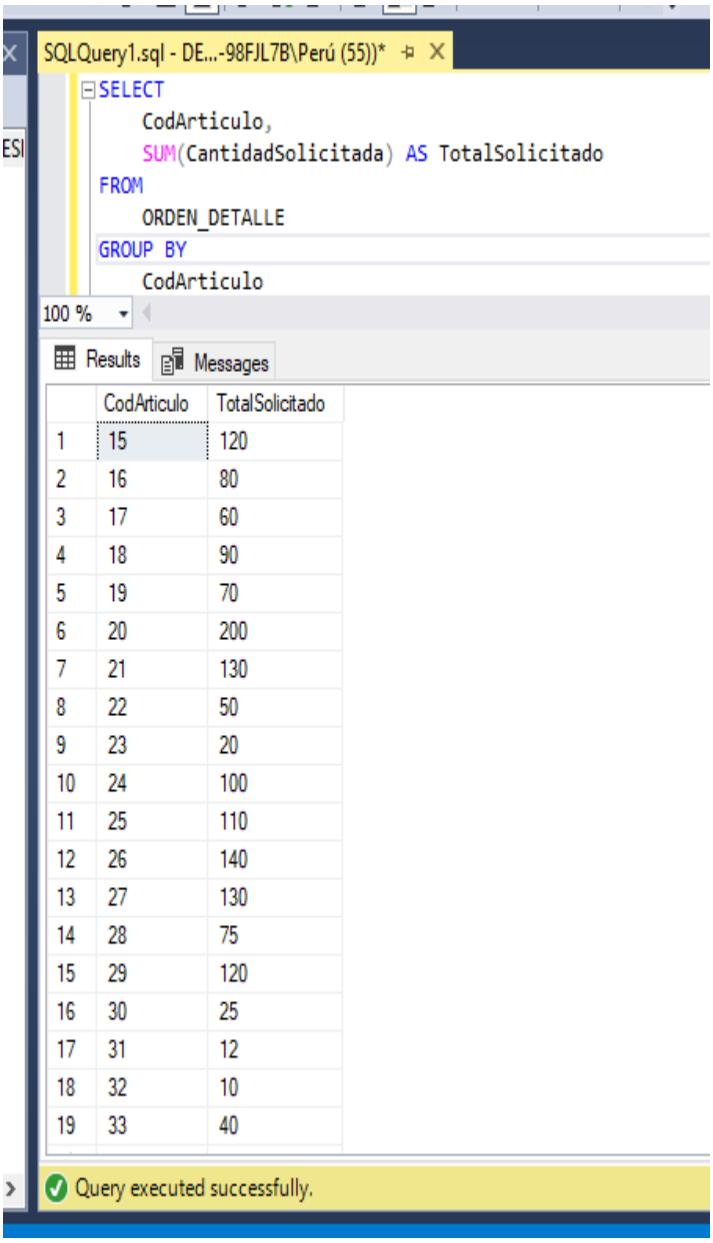
Query executed successfully.

Explicación:

- `PrecioVenta * CantidadEnviada`: calcula el **valor total de cada artículo** dentro de una guía.
- `SUM(...)`: suma todos los artículos de una misma guía.
- `GROUP BY NumGuia`: agrupa los resultados por cada guía de envío.
- `ORDER BY NumGuia`: ordena los resultados por número de guía.

7.Para cada CodArticulo mostrar TotalSolicitado

```
SELECT
    CodArticulo,
    SUM(CantidadSolicitada) AS TotalSolicitado
FROM
    ORDEN_DETALLE
GROUP BY
    CodArticulo
ORDER BY
    CodArticulo
```


SQLQuery1.sql - DE...-98FJL7B\Perú (55))*
SELECT
CodArticulo,
SUM(CantidadSolicitada) AS TotalSolicitado
FROM
ORDEN_DETALLE
GROUP BY
CodArticulo
100 %
Results
Messages
CodArticulo TotalSolicitado
1 15 120
2 16 80
3 17 60
4 18 90
5 19 70
6 20 200
7 21 130
8 22 50
9 23 20
10 24 100
11 25 110
12 26 140
13 27 130
14 28 75
15 29 120
16 30 25
17 31 12
18 32 10
19 33 40
Query executed successfully.

20	34	15
21	35	25
22	36	18
23	37	20
24	38	8
25	39	12
26	40	25
27	41	22
28	42	30
29	43	8
30	44	40
31	45	12
32	46	70
33	47	90
34	48	65
35	49	100
36	50	50
37	51	15
38	52	10
39	53	45
40	54	20
41	55	60
42	56	55
43	57	45
44	58	70
45	59	35
46	60	25
47	61	120
48	62	100
49	63	80
50	64	75

Query executed successfully.

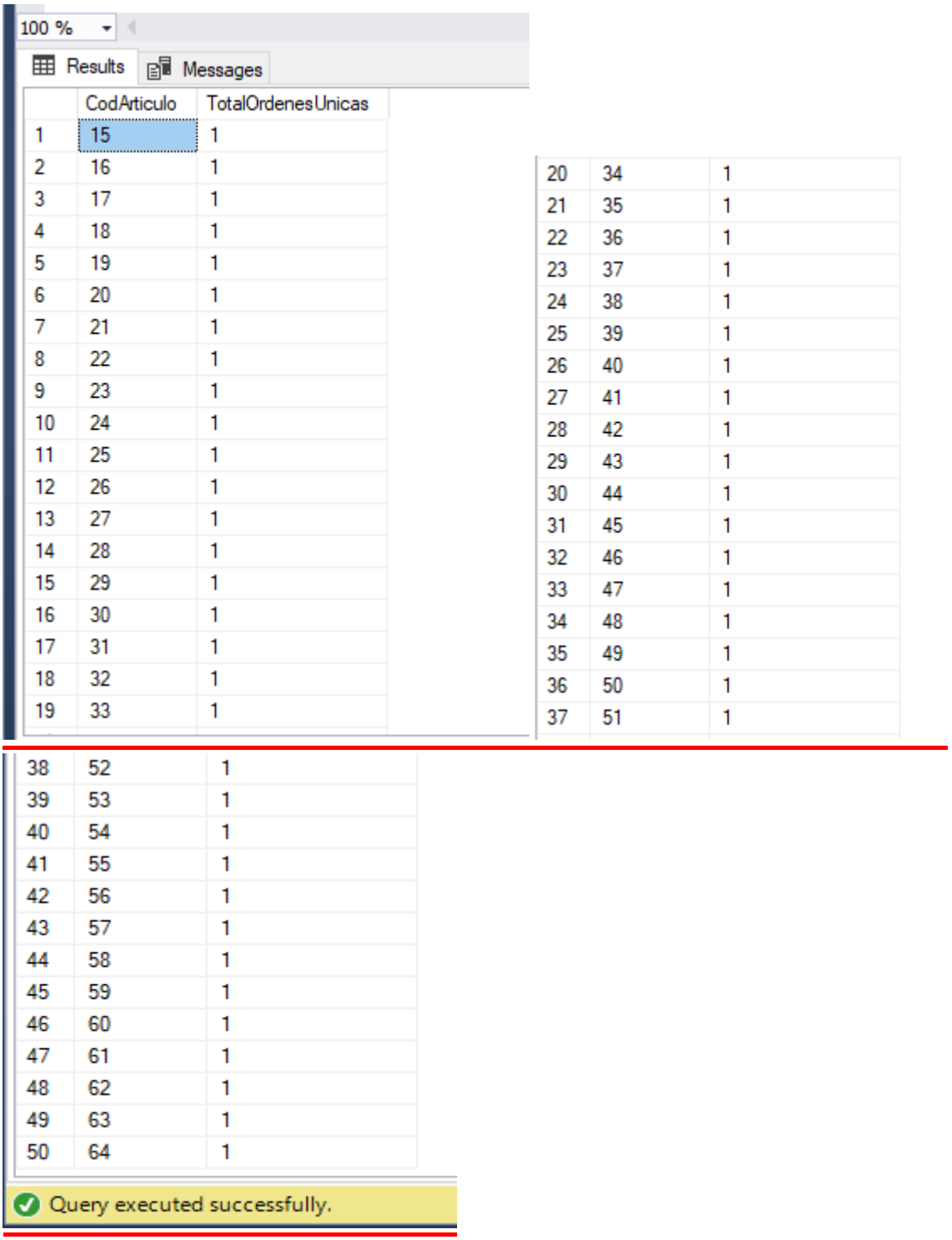
Explicación:

- `CantidadSolicitada`: representa cuántas unidades de un artículo se pidieron en una orden.
- `SUM(CantidadSolicitada)`: suma todas las solicitudes del mismo artículo.
- `GROUP BY CodArticulo`: agrupa los registros por artículo para obtener un total por cada uno.

- `ORDER BY CodArticulo`: ordena el resultado de forma ascendente por código de artículo.

8.Contar ordenes únicas que incluyen cada articulo

```
SELECT
    CodArticulo,
    COUNT(DISTINCT NumOrden) AS TotalOrdenesUnicas
FROM
    ORDEN_DETALLE
GROUP BY
    CodArticulo
ORDER BY
    CodArticulo;
```



100 %

Results | Messages

	CodArticulo	TotalOrdenesUnicas
1	15	1
2	16	1
3	17	1
4	18	1
5	19	1
6	20	1
7	21	1
8	22	1
9	23	1
10	24	1
11	25	1
12	26	1
13	27	1
14	28	1
15	29	1
16	30	1
17	31	1
18	32	1
19	33	1
20	34	1
21	35	1
22	36	1
23	37	1
24	38	1
25	39	1
26	40	1
27	41	1
28	42	1
29	43	1
30	44	1
31	45	1
32	46	1
33	47	1
34	48	1
35	49	1
36	50	1
37	51	1
38	52	1
39	53	1
40	54	1
41	55	1
42	56	1
43	57	1
44	58	1
45	59	1
46	60	1
47	61	1
48	62	1
49	63	1
50	64	1

Query executed successfully.

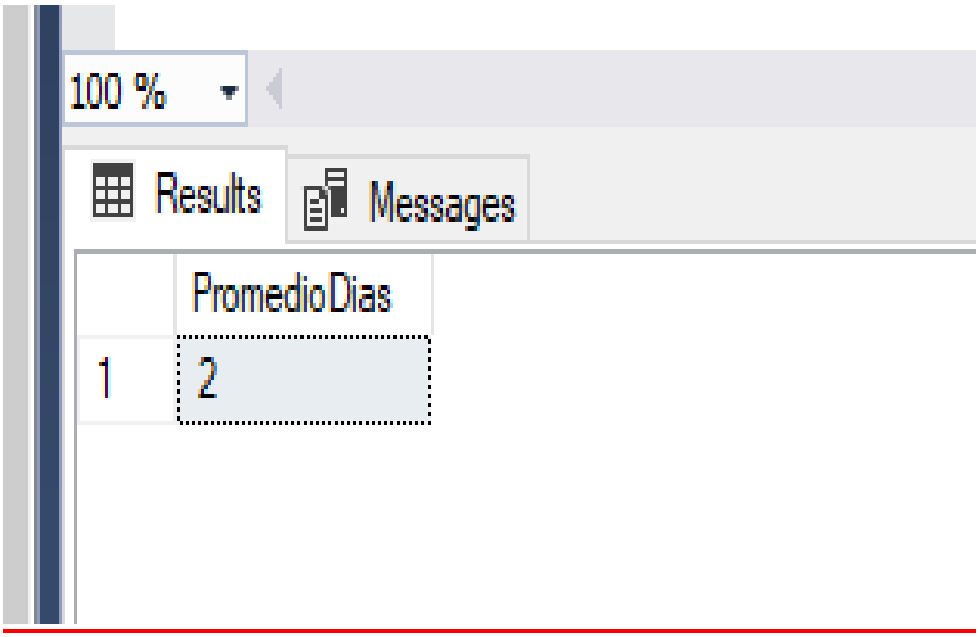
Explicación:

- `COUNT(DISTINCT NumOrden)`: cuenta **solo una vez** cada número de orden, aunque el mismo artículo aparezca varias veces en distintas órdenes.
- `GROUP BY CodArticulo`: agrupa los resultados por cada artículo.
- `ORDER BY CodArticulo`: ordena los resultados en orden ascendente por el código del artículo.

9.Calcular promedio de días por todas las ordenes con FechaIngreso

```
USE QhatuPERU;
GO

SELECT
    AVG(CAST(DATEDIFF(DAY, FechaOrden, FechaIngreso) AS FLOAT)) AS
PromedioDias
FROM
    dbo.ORDEN_COMPRA
WHERE
    FechaOrden IS NOT NULL
    AND FechaIngreso IS NOT NULL;
GO
```



DATEDIFF (DAY, FechaOrden, FechaIngreso)

🔻 Calcula la diferencia en días entre la fecha de la orden y la fecha de ingreso, para cada registro.

Por ejemplo:

FechaOrden = '2024-01-10'

FechaIngreso = '2024-01-13'

➞ Resultado: 3 días

Explicación

CAST (... AS FLOAT)

🔻 Convierte el resultado en un número decimal (por si hay fracciones en el promedio).

SQL Server normalmente devuelve enteros, y con esto garantizamos que el promedio no se redondee.

AVG(...)

🔻 Calcula el promedio general de los días de diferencia entre todas las órdenes que tienen ambas fechas registradas.

Esta consulta **calcula el promedio de días** que pasan entre la **fecha en que se realiza una orden de compra (FechaOrden)** y la **fecha en que se recibe (FechaIngreso)** en la tabla ORDEN_COMPRA.

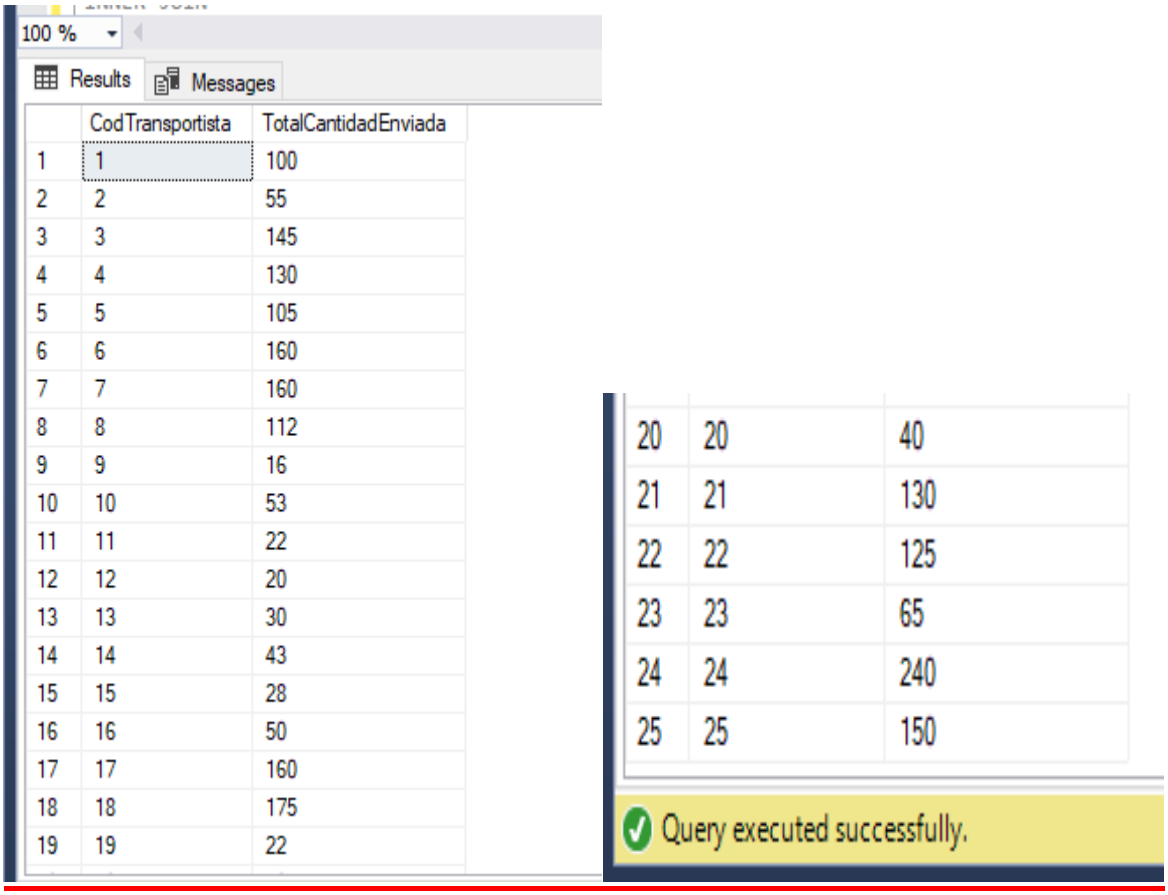
WHERE FechaOrden IS NOT NULL AND FechaIngreso IS NOT NULL

🔻 Filtra solo las órdenes que tienen ambas fechas completas (evita errores con valores vacíos o NULL).

10.Sumar CantidadEnviada por CodTransportista.

```
SELECT

    e.CodTransportista,
    SUM(d.CantidadEnviada) AS TotalCantidadEnviada
FROM
    GUIA_ENVIO e
INNER JOIN
    GUIA_DETALLE d ON e.NumGuia = d.NumGuia
GROUP BY
    e.CodTransportista
ORDER BY
    e.CodTransportista;
```



100 %

Results | Messages

	CodTransportista	TotalCantidadEnviada
1	1	100
2	2	55
3	3	145
4	4	130
5	5	105
6	6	160
7	7	160
8	8	112
9	9	16
10	10	53
11	11	22
12	12	20
13	13	30
14	14	43
15	15	28
16	16	50
17	17	160
18	18	175
19	19	22
20	20	40
21	21	130
22	22	125
23	23	65
24	24	240
25	25	150

Query executed successfully.

Explicación

- `INNER JOIN`: une ambas tablas según el número de guía (`NumGuia`).
- `SUM(d.CantidadEnviada)`: suma todas las cantidades enviadas por cada transportista.
- `GROUP BY e.CodTransportista`: agrupa los resultados por transportista.
- `ORDER BY e.CodTransportista`: ordena los resultados de forma ascendente.