FUNCIONES DE AGREGACIÓN

I. Explica de manera clara y didáctica que son la Funciones de agregación en SQL y como se utilizan.

Las funciones de agregación en SQL son herramientas que permiten realizar cálculos sobre un conjunto de valores de una columna y devolver un solo resultado.

Se utilizan principalmente junto con la cláusula *GROUP BY,* que agrupa los registros de una tabla según una o más columnas.

Sirven para resumir información de los datos almacenados en una tabla. Por ejemplo:

- Contar cuántos registros hay.
- Calcular el promedio de precios.
- Obtener el valor máximo o mínimo.
- Sumar todos los importes.

Principales funciones de agregación:

Función	Descripción	Ejemplo
COUNT()	Cuenta la cantidad de filas o valores no nulos.	SELECT COUNT(*) FROM Empleados;
SUM()	Suma los valores de una columna numérica.	SELECT SUM(Sueldo) FROM Empleados;
AVG()	Calcula el promedio de una columna numérica.	SELECT AVG(Sueldo) FROM Empleados;
MAX()	Devuelve el valor máximo de una columna.	SELECT MAX(Sueldo) FROM Empleados;
MIN()	Devuelve el valor mínimo de una columna.	SELECT MIN(Sueldo) FROM Empleados;

1. **Mostrar CodArticulo, DescripcionArticulo y ValorInventario**

```sql
USE QhatuPERU;
GO
--CONSULTA 1--
SELECT
    A.CodArticulo,
    A.DescripcionArticulo,
    CAST(A.StockActual * CAST(A.PrecioProveedor AS DECIMAL(18,2)) AS DECIMAL(18,2)) AS ValorInventario
FROM ARTICULO A;
```

Results | Messages

	CodArticulo	DescripcionArticulo	ValorInventario
1	1	Leche Evaporada Gloria	700.00
2	2	Pollo Entero San Fernando	1800.00
3	3	Inca Kola 1.5L	1800.00
4	4	Pan de Molde Blanco Molitalia	550.00
5	5	Manzana Roja Delicia	1400.00
6	6	Arroz Costeño	4500.00
7	7	Detergente Ariel	2700.00
8	8	Papel Higiénico Suave	2800.00
9	9	Helado Donofrio Vainilla	1600.00
10	10	Comida para Perro DogChow	4200.00
11	11	Pisco Acholado Vargas	4500.00
12	12	Vino Blanco Sauvignon Tacama	3000.00
13	13	Cerveza Pilsen Callao	2400.00
14	14	Papas Fritas Lay's Clásicas	500.00
15	15	Galletas Soda Field	320.00
16	16	Chocolate Sublime Clásico	150.00
17	17	Caramelos de Limón Arcor	400.00
18	18	Café Altomayo Grano Molido	1250.00
19	19	Cereal Zucaritas Kellogg's	1000.00
20	20	Manjarblanco Gloria	600.00
21	21	Mayonesa Alacena	850.00
22	22	Fideos Spaguetti Don Vittorio	280.00
23	23	Lentejas Costeño	400.00
24	24	Atún Filete Florida	550.00
25	25	Duraznos en Almíbar Aconcag...	900.00
26	26	Aceite Vegetal Primor	750.00
27	27	Harina Preparada Blanca Flor	650.00
28	28	Sopa Instantánea Ajinomen	120.00
29	29	Vitamina C Redoxon	1500.00
30	30	Pañales Pampers Confort Sec ...	3500.00
31	31	Shampoo H&S Limpieza Reno...	1800.00
32	32	Jabón Dove Original	300.00
33	33	Crema Facial Nivea Q10	4000.00

Consulta 1: Calcula el valor del inventario de cada artículo multiplicando su stock actual por el precio del proveedor, mostrando el código, descripción y el valor monetario total de ese producto; así se puede conocer cuánto dinero representa cada artículo almacenado.

2. **Calcular el total monetario del inventario.**

```sql
--CONSULTA 2--
SELECT
    SUM(CAST(A.StockActual * CAST(A.PrecioProveedor AS DECIMAL(18,2)) AS DECIMAL(18,2))) AS TotalMonetarioInventario
FROM ARTICULO A;
```

Results | Messages

	TotalMonetarioInventario
1	121760.00

Consulta 2: Suma el valor monetario de todos los artículos del inventario, permitiendo conocer el total general del inventario en soles o moneda local; es decir, cuánto dinero en productos tiene la empresa en existencia.

3. **Obtener CodLinea y Precio Proveedor promedio.**

```sql
--CONSULTA 3--
SELECT
    L.CodLinea,
    AVG(CAST(A.PrecioProveedor AS DECIMAL(18,2))) AS PrecioProveedorPromedio
FROM ARTICULO A
JOIN LINEA L ON A.CodLinea = L.CodLinea
GROUP BY L.CodLinea;
```

100 %

Results | Messages

	CodLinea	PrecioProveedorPromedio
1	1	3.500000
2	2	12.000000
3	3	6.000000
4	4	5.500000
5	5	2.800000
6	6	18.000000
7	7	15.000000
8	8	7.000000
9	9	20.000000
10	10	35.000000
11	11	45.000000
12	12	30.000000
13	13	24.000000
14	14	5.000000
15	15	3.200000
16	16	1.500000
17	17	4.000000
18	18	12.500000
19	19	10.000000
20	20	6.000000
21	21	8.500000
22	22	2.800000
23	23	4.000000

Consulta 3: Calcula el precio promedio del proveedor por cada línea de artículos al agruparlos mediante GROUP BY CodLinea, lo que permite comparar los costos promedio entre diferentes categorías o familias de productos.

4. **Contar artículos descontinuados.**

```sql
--CONSULTA 4--
SELECT
    COUNT(*) AS ArticulosDescontinuados
FROM ARTICULO A
WHERE A.Descontinuado = 1;
```

100 %

Results | Messages

	ArticulosDescontinuados
1	0

Consulta 4: Cuenta cuántos artículos están marcados como descontinuados (ya no disponibles para la venta), mostrando un único valor que indica el total de productos inactivos en la base de datos.

5. **Mostrar PrecioMaximo y PrecioMinimo del catálogo.**

```
--CONSULTA 5--
SELECT
    MAX(A.PrecioProveedor) AS PrecioMaximo,
    MIN(A.PrecioProveedor) AS PrecioMinimo
FROM ARTICULO A;
```

100 %

Results | Messages

	PrecioMaximo	PrecioMinimo
1	199.00	1.20

Consulta 5: Determina el precio más alto y el más bajo de los artículos registrados, mostrando el rango de precios que existe en el inventario, útil para conocer los extremos de costos entre los productos.

6. **Mostrar el Valor total enviado por guía.**

```
--CONSULTA 6--
SELECT
    G.NumGuia,
    SUM(CAST(GD.CantidadEnviada * CAST(GD.PrecioVenta AS DECIMAL(18,2)) AS DECIMAL(18,2))) AS ValorTotalEnviado
FROM GUIA_DETALLE GD
JOIN GUIA_ENVIO G ON GD.NumGuia = G.NumGuia
GROUP BY G.NumGuia;
```

100 %

Results | Messages

	NumGuia	ValorTotalEnviado
1	1	84.00
2	2	145.00
3	3	375.00
4	4	97.50
5	5	105.00
6	6	525.00
7	7	180.00
8	8	510.00
9	9	200.00
10	10	504.00
11	11	550.00
12	12	380.00
13	13	300.00
14	14	65.00
15	15	40.00
16	16	20.00
17	17	50.00
18	18	150.00
19	19	125.00
20	20	75.00
21	21	100.00
22	22	35.00
23	23	50.00
24	24	68.00
25	25	110.00
26	26	95.00
27	27	80.00
28	28	18.00
29	29	190.00
30	30	420.00
31	31	220.00
32	32	40.00

Consulta 6: Calcula el valor total enviado por cada guía de envío, multiplicando la cantidad enviada por el precio de venta y sumando por número de guía; esto permite conocer el valor monetario de cada despacho o entrega realizada.

7. **Para cada CodArticulo, mostrar TotalSolicitado.**

```sql
--CONSULTA 7--
SELECT
    A.CodArticulo,
    SUM(OD.CantidadSolicitada) AS TotalSolicitado
FROM ORDEN_DETALLE OD
JOIN ARTICULO A ON OD.CodArticulo = A.CodArticulo
GROUP BY A.CodArticulo;
```

100 %

Results | Messages

	CodArticulo	TotalSolicitado
1	1	100
2	2	50
3	3	200
4	4	80
5	5	150
6	6	100
7	7	60
8	8	300
9	9	40
10	10	70
11	11	50
12	12	50
13	13	50
14	14	50
15	15	50
16	16	50
17	17	50
18	18	50
19	19	50
20	20	50
21	21	50
22	22	50
23	23	50
24	24	50
25	25	50
26	26	50
27	27	50
28	28	50
29	29	50
30	30	50
31	31	50
32	32	50
33	33	50

Consulta 7: Suma las cantidades solicitadas de cada artículo en todas las órdenes de compra, mostrando el total pedido por producto y ayudando a identificar cuáles son los artículos más demandados.

8. **Contar órdenes únicas que incluyen cada artículo.**

```sql
--CONSULTA 8--
SELECT
    A.CodArticulo,
    COUNT(DISTINCT OD.NumOrden) AS OrdenesUnicas
FROM ORDEN_DETALLE OD
JOIN ARTICULO A ON OD.CodArticulo = A.CodArticulo
GROUP BY A.CodArticulo;
```

100 %

Results | Messages

	CodArticulo	OrdenesUnicas
1	1	1
2	2	1
3	3	1
4	4	1
5	5	1
6	6	1
7	7	1
8	8	1
9	9	1
10	10	1
11	11	1
12	12	1
13	13	1
14	14	1
15	15	1
16	16	1
17	17	1
18	18	1
19	19	1
20	20	1
21	21	1
22	22	1
23	23	1
24	24	1
25	25	1
26	26	1
27	27	1
28	28	1
29	29	1
30	30	1
31	31	1
32	32	1
33	33	1
34	34	1

Consulta 8: Cuenta en cuántas órdenes diferentes aparece cada artículo, utilizando COUNT(DISTINCT) para no repetir órdenes; de este modo, se conoce la frecuencia con la que cada producto ha sido solicitado en pedidos distintos.

9. **Calcular promedio de días por todas las órdenes con FechaIngreso.**

```sql
--CONSULTA 9--
SELECT
    AVG(DATEDIFF(DAY, O.FechaOrden, O.FechaIngreso)) AS PromedioDiasOrden
FROM ORDEN_COMPRA O
WHERE O.FechaIngreso IS NOT NULL;
```

100 %

Results | Messages

	PromedioDiasOrden
1	4

Consulta 9: Calcula el promedio de días transcurridos entre la fecha de orden y la fecha de ingreso de las compras, midiendo la rapidez del proceso de adquisición o el tiempo promedio que demora en completarse una orden.

10. **Sumar Cantidad Enviada por CodTransportista.**

```sql
--CONSULTA 10 --
SELECT
    G.CodTransportista,
    SUM(GD.CantidadEnviada) AS TotalCantidadEnviada
FROM GUIA_DETALLE GD
JOIN GUIA_ENVIO G ON GD.NumGuia = G.NumGuia
GROUP BY G.CodTransportista;
```

100 %

Results | Messages

	CodTransportista	TotalCantidadEnviada
1	1	20
2	2	10
3	3	50
4	4	15
5	5	30
6	6	25
7	7	10
8	8	60
9	9	8
10	10	12
11	11	10
12	12	10
13	13	10
14	14	10
15	15	10
16	16	10
17	17	10
18	18	10
19	19	10
20	20	10
21	21	10
22	22	10
23	23	10
24	24	10
25	25	10
26	26	10
27	27	10
28	28	10
29	29	10
30	30	10
31	31	10
32	32	10
33	33	10
34	34	10

Consulta 10: Suma la cantidad total enviada por cada transportista agrupando por su código, con lo que se obtiene la carga total trasladada por cada uno, útil para evaluar el volumen de trabajo o eficiencia de los transportistas.

CLÁUSULA GROUP BY

II. Explica de manera clara y didáctica qué son la CLÁUSULA GROUP BY en SQL y cómo se utilizan.

La cláusula GROUP BY es una instrucción del lenguaje SQL que sirve para agrupar los registros (filas) de una tabla que tienen valores iguales en una o más columnas, con el propósito de aplicar funciones de agregación sobre cada grupo de datos.

En otras palabras, GROUP BY organiza los datos en grupos según una o varias columnas y permite obtener resultados resumidos, como totales, promedios o conteos.

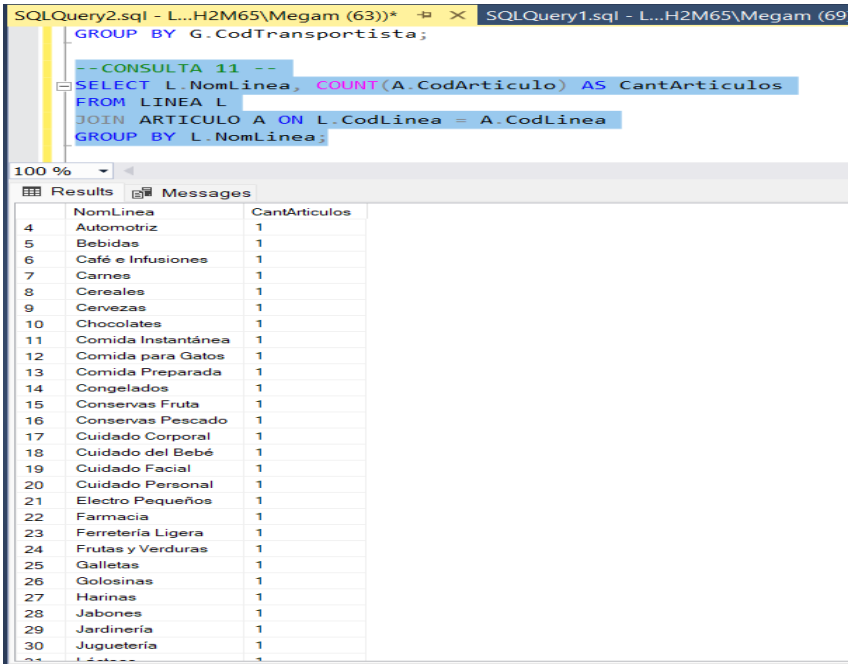
Sintaxis básica

```
SELECT columna_agrupada, función_agregada(columna)

FROM nombre_tabla

GROUP BY columna_agrupada;
```

Donde:

- columna_agrupada: es la columna por la que se quiere agrupar los datos.
- función_agregada: puede ser COUNT(), SUM(), AVG(), MAX(), o MIN().

11. **Mostrar NomLinea y CantArticulos.**



```
SQLQuery2.sql - L...H2M65\Megam (63))*   SQLQuery1.sql - L...H2M65\Megam (69
    GROUP BY G.CodTransportista;

    --CONSULTA 11 --
    SELECT L.NomLinea, COUNT(A.CodArticulo) AS CantArticulos
    FROM LINEA L
    JOIN ARTICULO A ON L.CodLinea = A.CodLinea
    GROUP BY L.NomLinea;
```

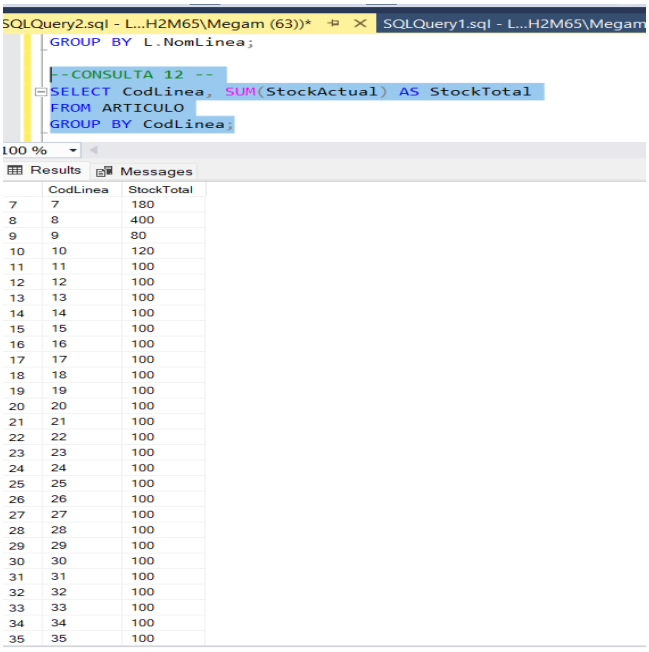
100 %

Results | Messages

	NomLinea	CantArticulos
4	Automotriz	1
5	Bebidas	1
6	Café e Infusiones	1
7	Carnes	1
8	Cereales	1
9	Cervezas	1
10	Chocolates	1
11	Comida Instantánea	1
12	Comida para Gatos	1
13	Comida Preparada	1
14	Congelados	1
15	Conservas Fruta	1
16	Conservas Pescado	1
17	Cuidado Corporal	1
18	Cuidado del Bebé	1
19	Cuidado Facial	1
20	Cuidado Personal	1
21	Electro Pequeños	1
22	Farmacia	1
23	Ferretería Ligera	1
24	Frutas y Verduras	1
25	Galletas	1
26	Golosinas	1
27	Harinas	1
28	Jabones	1
29	Jardinería	1
30	Juguetería	1

Consulta 11: Cuenta la cantidad total de artículos pertenecientes a cada línea de productos, agrupando por el nombre de línea, lo que permite conocer cuántos artículos tiene registrada cada categoría o familia de productos.

12. **Mostrar CodLinea y StockTotal.**



```
SQLQuery2.sql - L...H2M65\Megam (63))*   SQLQuery1.sql - L...H2M65\Megam
    GROUP BY L.NomLinea;

    --CONSULTA 12 --
    SELECT CodLinea, SUM(StockActual) AS StockTotal
    FROM ARTICULO
    GROUP BY CodLinea;
```

100 %

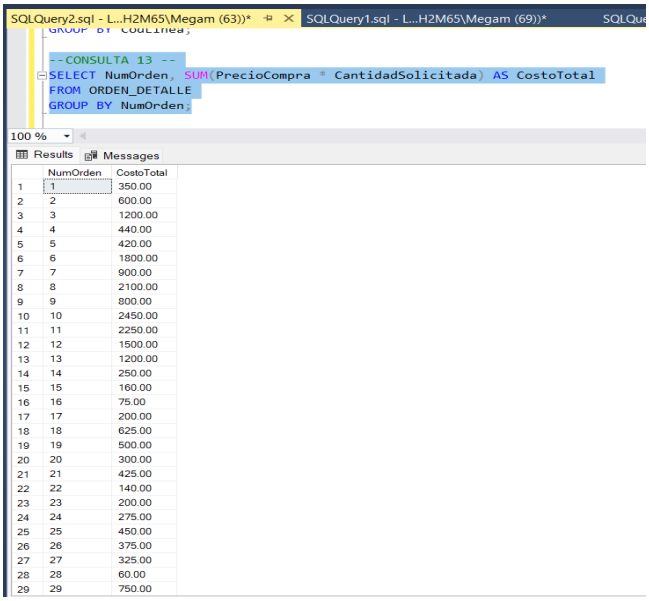
Results | Messages

	CodLinea	StockTotal
7	7	180
8	8	400
9	9	80
10	10	120
11	11	100
12	12	100
13	13	100
14	14	100
15	15	100
16	16	100
17	17	100
18	18	100
19	19	100
20	20	100
21	21	100
22	22	100
23	23	100
24	24	100
25	25	100
26	26	100
27	27	100
28	28	100
29	29	100
30	30	100
31	31	100
32	32	100
33	33	100
34	34	100
35	35	100

Consulta 12:

Suma el stock actual de todos los artículos agrupados por su código de línea, con lo que se obtiene el total de existencias disponibles por cada línea de productos, útil para controlar inventarios.

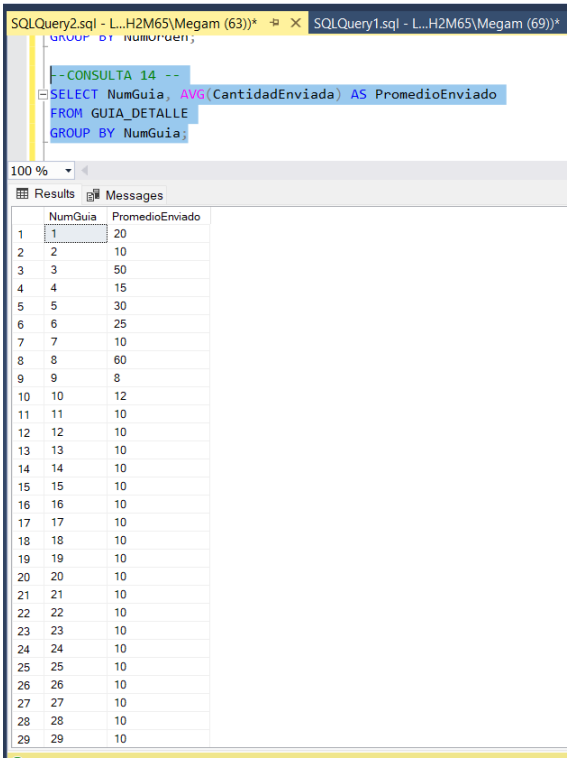
13. **Para cada NumOrden, calcular CostoTotal = SUM(PrecioCompra×Cantidad).**



	NumOrden	CostoTotal
1	1	350.00
2	2	600.00
3	3	1200.00
4	4	440.00
5	5	420.00
6	6	1800.00
7	7	900.00
8	8	2100.00
9	9	800.00
10	10	2450.00
11	11	2250.00
12	12	1500.00
13	13	1200.00
14	14	250.00
15	15	160.00
16	16	75.00
17	17	200.00
18	18	625.00
19	19	500.00
20	20	300.00
21	21	425.00
22	22	140.00
23	23	200.00
24	24	275.00
25	25	450.00
26	26	375.00
27	27	325.00
28	28	60.00
29	29	750.00

Consulta 13: Calcula el costo total de cada orden de compra multiplicando el precio de compra por la cantidad solicitada y agrupando por número de orden, lo que permite conocer el valor total de cada pedido.

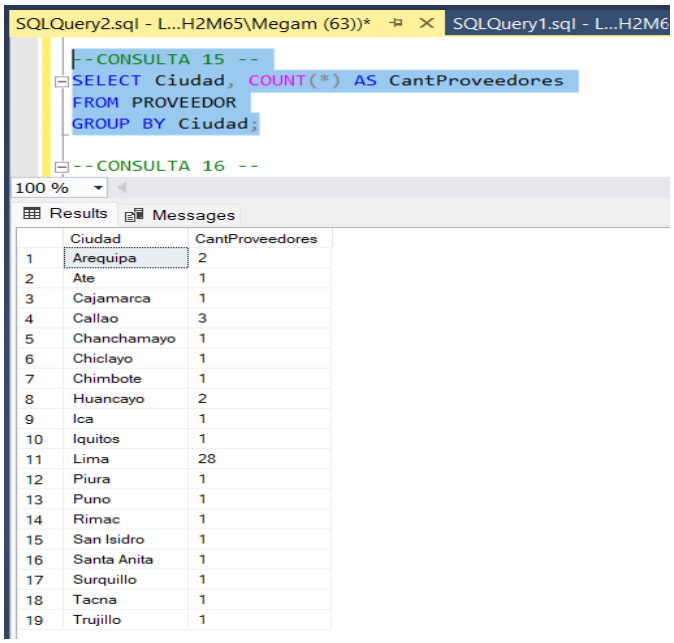
14. **Mostrar NumGuia y PromedioEnviado.**



	NumGuia	PromedioEnviado
1	1	20
2	2	10
3	3	50
4	4	15
5	5	30
6	6	25
7	7	10
8	8	60
9	9	8
10	10	12
11	11	10
12	12	10
13	13	10
14	14	10
15	15	10
16	16	10
17	17	10
18	18	10
19	19	10
20	20	10
21	21	10
22	22	10
23	23	10
24	24	10
25	25	10
26	26	10
27	27	10
28	28	10
29	29	10

Consulta 14: Obtiene el promedio de la cantidad enviada por cada guía de remisión, agrupando por número de guía, con el fin de analizar el promedio de unidades despachadas por envío.

15. **Contar proveedores agrupados por Ciudad.**



```
--CONSULTA 15 --
SELECT Ciudad, COUNT(*) AS CantProveedores
FROM PROVEEDOR
GROUP BY Ciudad;

--CONSULTA 16 --
```

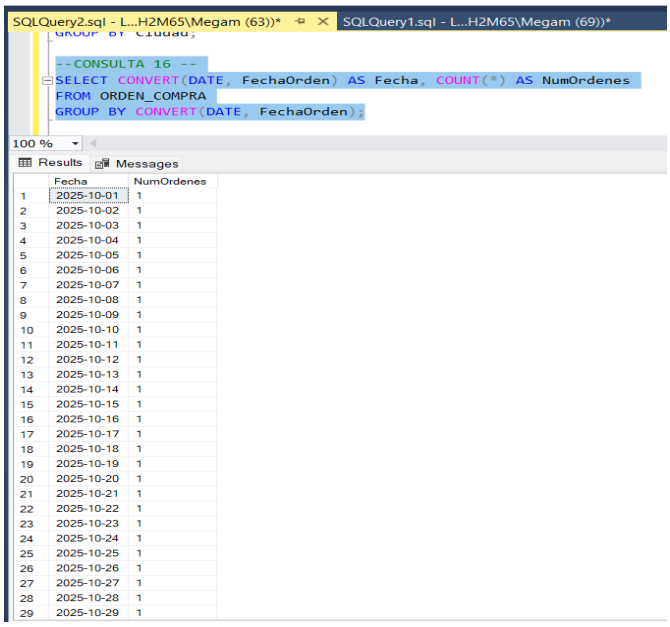
100 %

Results | Messages

	Ciudad	CantProveedores
1	Arequipa	2
2	Ate	1
3	Cajamarca	1
4	Callao	3
5	Chanchamayo	1
6	Chiclayo	1
7	Chimbote	1
8	Huancayo	2
9	Ica	1
10	Iquitos	1
11	Lima	28
12	Piura	1
13	Puno	1
14	Rimac	1
15	San Isidro	1
16	Santa Anita	1
17	Surquillo	1
18	Tacna	1
19	Trujillo	1

Consulta 15: Cuenta la cantidad de proveedores existentes en cada ciudad agrupando por el nombre de la ciudad, lo que facilita conocer la distribución geográfica de los proveedores.

16. **Mostrar el número de órdenes por día (sin hora).**



```
--CONSULTA 16 --
SELECT CONVERT(DATE, FechaOrden) AS Fecha, COUNT(*) AS NumOrdenes
FROM ORDEN_COMPRA
GROUP BY CONVERT(DATE, FechaOrden);
```

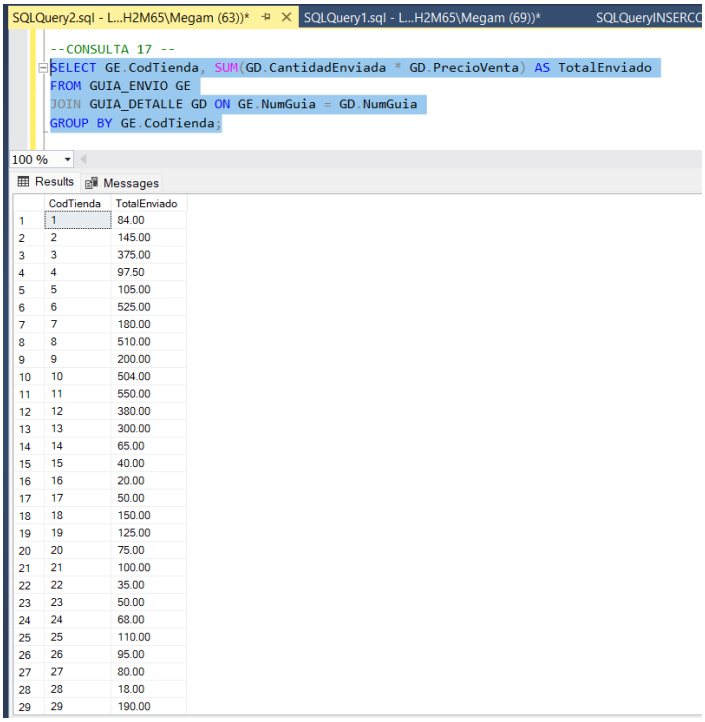
100 %

Results | Messages

	Fecha	NumOrdenes
1	2025-10-01	1
2	2025-10-02	1
3	2025-10-03	1
4	2025-10-04	1
5	2025-10-05	1
6	2025-10-06	1
7	2025-10-07	1
8	2025-10-08	1
9	2025-10-09	1
10	2025-10-10	1
11	2025-10-11	1
12	2025-10-12	1
13	2025-10-13	1
14	2025-10-14	1
15	2025-10-15	1
16	2025-10-16	1
17	2025-10-17	1
18	2025-10-18	1
19	2025-10-19	1
20	2025-10-20	1
21	2025-10-21	1
22	2025-10-22	1
23	2025-10-23	1
24	2025-10-24	1
25	2025-10-25	1
26	2025-10-26	1
27	2025-10-27	1
28	2025-10-28	1
29	2025-10-29	1

Consulta 16: Cuenta el número total de órdenes realizadas por día, agrupando las fechas de las órdenes de compra, para conocer la cantidad de pedidos gestionados en cada fecha específica.

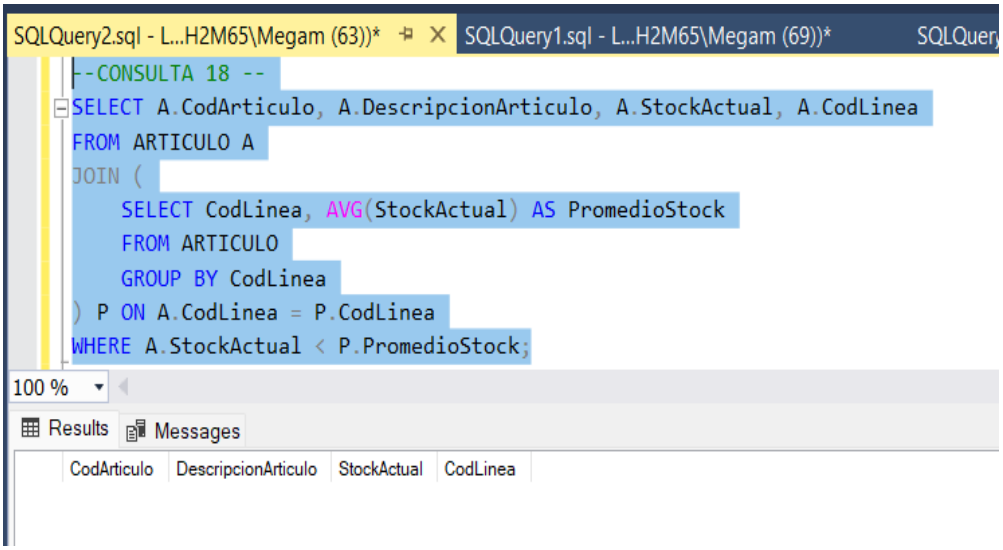
17. Sumar (Cantidad Enviada×Precio Venta) por CodTienda.



	CodTienda	TotalEnviado
1	1	84.00
2	2	145.00
3	3	375.00
4	4	97.50
5	5	105.00
6	6	525.00
7	7	180.00
8	8	510.00
9	9	200.00
10	10	504.00
11	11	550.00
12	12	380.00
13	13	300.00
14	14	65.00
15	15	40.00
16	16	20.00
17	17	50.00
18	18	150.00
19	19	125.00
20	20	75.00
21	21	100.00
22	22	35.00
23	23	50.00
24	24	68.00
25	25	110.00
26	26	95.00
27	27	80.00
28	28	18.00
29	29	190.00

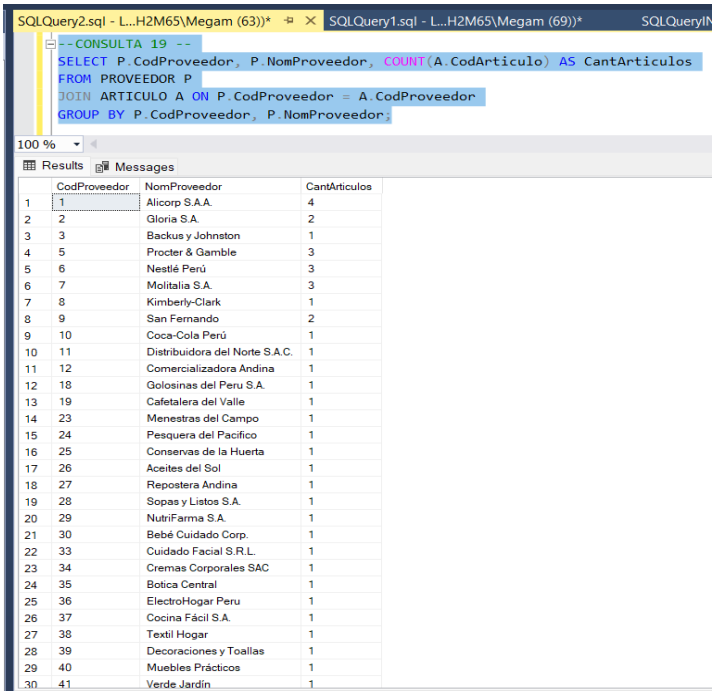
Consulta 17: Suma el total monetario de productos enviados a cada tienda multiplicando la cantidad enviada por el precio de venta y agrupando por código de tienda, con lo que se obtiene el valor total despachado a cada punto de venta.

18. Mostrar artículos cuyo StockActual < promedio de su CodLinea.



Consulta 18: Selecciona los artículos cuyo stock actual es menor que el promedio de stock de su línea, agrupando previamente por código de línea, con el propósito de identificar productos con bajo nivel de inventario.

19. Mostrar CodProveedor, Nom Proveedor y CantArticulos.



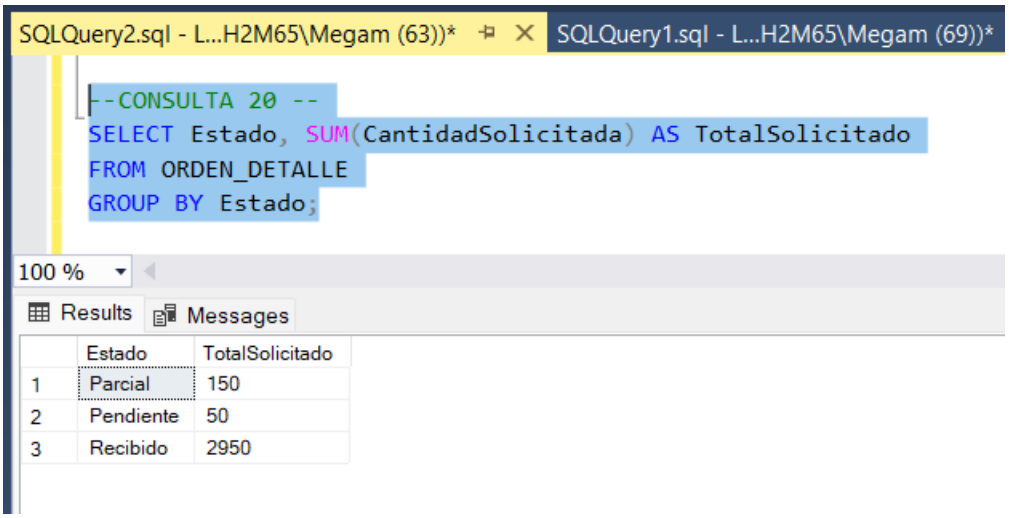
```
--CONSULTA 19 --
SELECT P.CodProveedor, P.NomProveedor, COUNT(A.CodArticulo) AS CantArticulos
FROM PROVEEDOR P
JOIN ARTICULO A ON P.CodProveedor = A.CodProveedor
GROUP BY P.CodProveedor, P.NomProveedor;
```

100 % — Results | Messages

	CodProveedor	NomProveedor	CantArticulos
1	1	Alicorp S.A.A.	4
2	2	Gloria S.A.	2
3	3	Backus y Johnston	1
4	5	Procter & Gamble	3
5	6	Nestlé Perú	3
6	7	Molitalia S.A.	3
7	8	Kimberly-Clark	1
8	9	San Fernando	2
9	10	Coca-Cola Perú	1
10	11	Distribuidora del Norte S.A.C.	1
11	12	Comercializadora Andina	1
12	18	Golosinas del Peru S.A.	1
13	19	Cafetalera del Valle	1
14	23	Menestras del Campo	1
15	24	Pesquera del Pacifico	1
16	25	Conservas de la Huerta	1
17	26	Aceites del Sol	1
18	27	Repostera Andina	1
19	28	Sopas y Listos S.A.	1
20	29	NutriFarma S.A.	1
21	30	Bebé Cuidado Corp.	1
22	33	Cuidado Facial S.R.L.	1
23	34	Cremas Corporales SAC	1
24	35	Botica Central	1
25	36	ElectroHogar Peru	1
26	37	Cocina Fácil S.A.	1
27	38	Textil Hogar	1
28	39	Decoraciones y Toallas	1
29	40	Muebles Prácticos	1
30	41	Verde Jardín	1

Consulta 19: Cuenta la cantidad total de artículos que suministra cada proveedor, agrupando por su código y nombre, lo que permite conocer la participación o aporte de cada proveedor en el catálogo de productos.

20. Mostar para cada Estado sumar CantidadSolicitada.



```
--CONSULTA 20 --
SELECT Estado, SUM(CantidadSolicitada) AS TotalSolicitado
FROM ORDEN_DETALLE
GROUP BY Estado;
```

100 % — Results | Messages

	Estado	TotalSolicitado
1	Parcial	150
2	Pendiente	50
3	Recibido	2950

Consulta 20: Suma la cantidad total de productos solicitados en las órdenes agrupando por estado, lo que permite conocer el volumen de pedidos según su situación (pendiente, atendido, etc.).

CLÁUSULA OVER

III. Explica de manera clara y didáctica qué son la CLÁUSULA OVER en SQL y cómo se utilizan.

La cláusula OVER en SQL se utiliza junto con funciones de ventana (o analíticas) para realizar cálculos avanzados sobre un conjunto de filas relacionadas, sin necesidad de agruparlas como lo hace GROUP BY.

En otras palabras, permite aplicar funciones de agregación o ranking sobre una "ventana" de datos, es decir, un grupo de filas definidas dentro de la consulta, sin perder el detalle individual de cada fila.

Sintaxis general de OVER

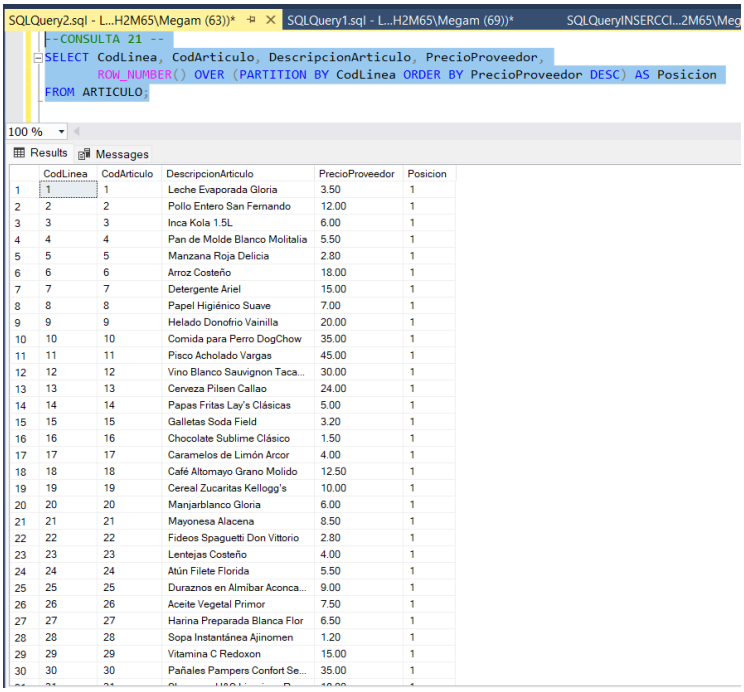
```
<función> OVER (
  [PARTITION BY columnas]
  [ORDER BY columnas]
  [ROWS or RANGE ...]
)
```

- **PARTITION BY**: Divide los datos en grupos o "particiones" donde se aplicará la función.
  (Es similar al GROUP BY, pero sin eliminar filas).
- **ORDER BY**: Indica el orden dentro de cada partición.
  Es obligatorio en funciones como ROW_NUMBER (), RANK (), LAG (), etc.
- **ROWS o RANGE**: Define un rango de filas alrededor de la actual (usado en promedios móviles o acumulados).

Principales funciones que usan OVER

1. **Funciones de clasificación (Ranking):**
   - ROW_NUMBER () → Numera las filas dentro de cada grupo.
   - RANK () → Asigna rangos, dejando saltos cuando hay empates.
   - DENSE_RANK () → Asigna rangos consecutivos, sin saltos.
2. **Funciones de agregación con OVER:**
   - SUM (), AVG (), COUNT (), MIN (), MAX ()
     Permiten calcular totales o promedios acumulados o por grupo, sin agrupar los datos.

21. Asignar posición por línea ordenada por precio.



```sql
--CONSULTA 21 --
SELECT CodLinea, CodArticulo, DescripcionArticulo, PrecioProveedor,
       ROW_NUMBER() OVER (PARTITION BY CodLinea ORDER BY PrecioProveedor DESC) AS Posicion
FROM ARTICULO;
```

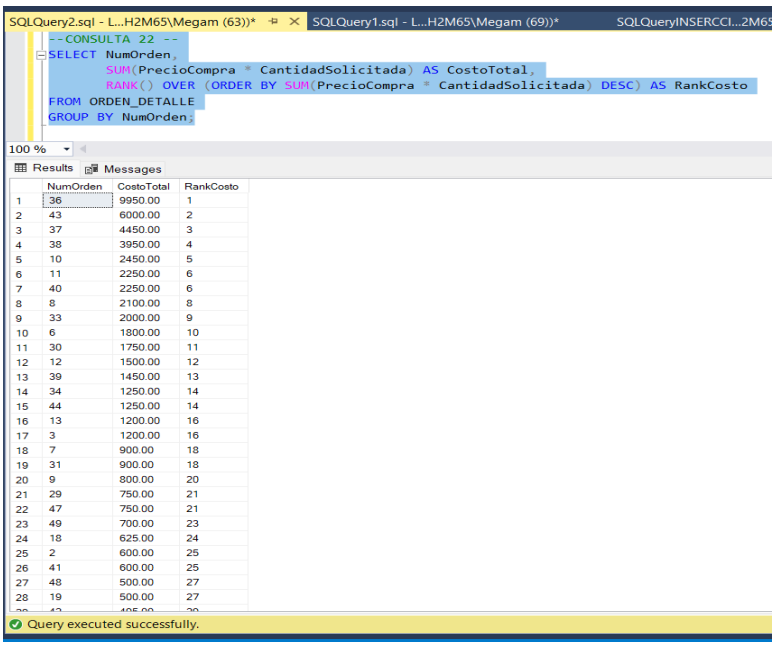
100 %

Results | Messages

	CodLinea	CodArticulo	DescripcionArticulo	PrecioProveedor	Posicion
1	1	1	Leche Evaporada Gloria	3.50	1
2	2	2	Pollo Entero San Fernando	12.00	1
3	3	3	Inca Kola 1.5L	6.00	1
4	4	4	Pan de Molde Blanco Molitalia	5.50	1
5	5	5	Manzana Roja Delicia	2.80	1
6	6	6	Arroz Costeño	18.00	1
7	7	7	Detergente Ariel	15.00	1
8	8	8	Papel Higiénico Suave	7.00	1
9	9	9	Helado Donofrio Vainilla	20.00	1
10	10	10	Comida para Perro DogChow	35.00	1
11	11	11	Pisco Acholado Vargas	45.00	1
12	12	12	Vino Blanco Sauvignon Taca...	30.00	1
13	13	13	Cerveza Pilsen Callao	24.00	1
14	14	14	Papas Fritas Lay's Clásicas	5.00	1
15	15	15	Galletas Soda Field	3.20	1
16	16	16	Chocolate Sublime Clásico	1.50	1
17	17	17	Caramelos de Limón Arcor	4.00	1
18	18	18	Café Altomayo Grano Molido	12.50	1
19	19	19	Cereal Zucaritas Kellogg's	10.00	1
20	20	20	Manjarblanco Gloria	6.00	1
21	21	21	Mayonesa Alacena	8.50	1
22	22	22	Fideos Spaguetti Don Vittorio	2.80	1
23	23	23	Lentejas Costeño	4.00	1
24	24	24	Atún Filete Florida	5.50	1
25	25	25	Duraznos en Almíbar Aconca...	9.00	1
26	26	26	Aceite Vegetal Primor	7.50	1
27	27	27	Harina Preparada Blanca Flor	6.50	1
28	28	28	Sopa Instantánea Ajinomen	1.20	1
29	29	29	Vitamina C Redoxon	15.00	1
30	30	30	Pañales Pampers Confort Se...	35.00	1

Consulta 21: Enumera los artículos dentro de cada línea asignándoles una posición según su precio de proveedor, ordenados de mayor a menor, lo que permite identificar cuál es el artículo más costoso dentro de cada línea.

22. Calcular costo por orden y su RANK (RankCosto)



```sql
--CONSULTA 22 --
SELECT NumOrden,
       SUM(PrecioCompra * CantidadSolicitada) AS CostoTotal,
       RANK() OVER (ORDER BY SUM(PrecioCompra * CantidadSolicitada) DESC) AS RankCosto
FROM ORDEN_DETALLE
GROUP BY NumOrden;
```

100 %

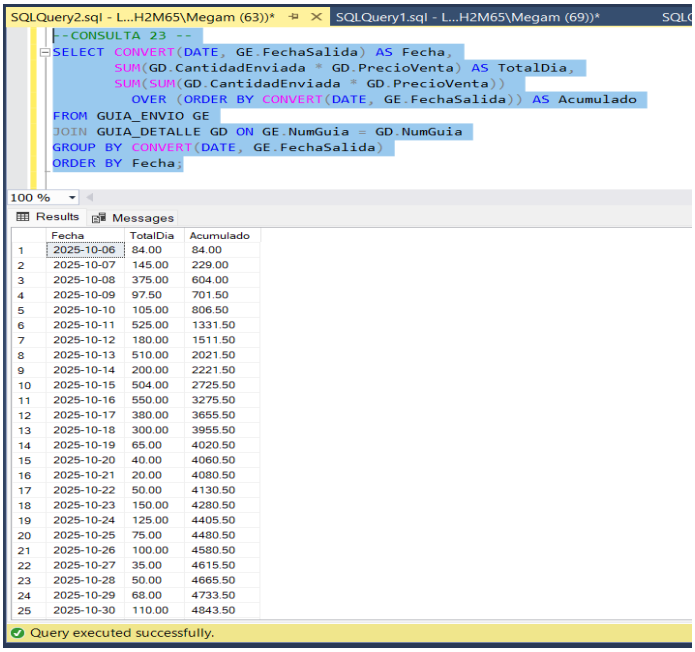
Results | Messages

	NumOrden	CostoTotal	RankCosto
1	36	9950.00	1
2	43	6000.00	2
3	37	4450.00	3
4	38	3950.00	4
5	10	2450.00	5
6	11	2250.00	6
7	40	2250.00	6
8	8	2100.00	8
9	33	2000.00	9
10	6	1800.00	10
11	30	1750.00	11
12	12	1500.00	12
13	39	1450.00	13
14	34	1250.00	14
15	44	1250.00	14
16	13	1200.00	16
17	3	1200.00	16
18	7	900.00	18
19	31	900.00	18
20	9	800.00	20
21	29	750.00	21
22	47	750.00	21
23	49	700.00	23
24	18	625.00	24
25	2	600.00	25
26	41	600.00	25
27	48	500.00	27
28	19	500.00	27

Query executed successfully.

Consulta 22: Calcula el costo total de cada orden de compra y asigna un rango de posición según el valor total, mostrando cuáles órdenes fueron más costosas, útil para analizar los pedidos de mayor inversión.

23. Mostrar TotalDia y Acumulado Ventas ordenado por fecha.



```sql
--CONSULTA 23 --
SELECT CONVERT(DATE, GE.FechaSalida) AS Fecha,
       SUM(GD.CantidadEnviada * GD.PrecioVenta) AS TotalDia,
       SUM(SUM(GD.CantidadEnviada * GD.PrecioVenta))
         OVER (ORDER BY CONVERT(DATE, GE.FechaSalida)) AS Acumulado
FROM GUIA_ENVIO GE
JOIN GUIA_DETALLE GD ON GE.NumGuia = GD.NumGuia
GROUP BY CONVERT(DATE, GE.FechaSalida)
ORDER BY Fecha;
```

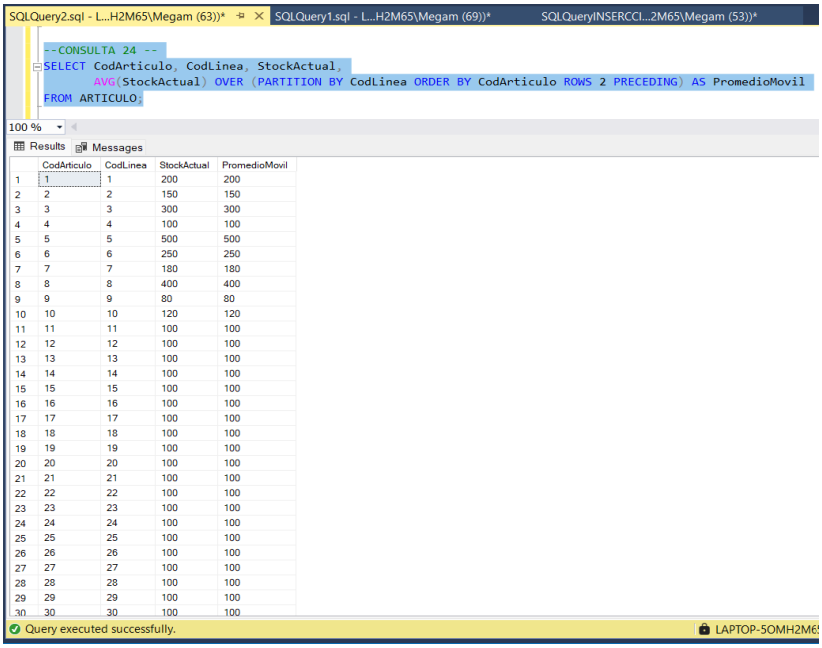
Results | Messages

	Fecha	TotalDia	Acumulado
1	2025-10-06	84.00	84.00
2	2025-10-07	145.00	229.00
3	2025-10-08	375.00	604.00
4	2025-10-09	97.50	701.50
5	2025-10-10	105.00	806.50
6	2025-10-11	525.00	1331.50
7	2025-10-12	180.00	1511.50
8	2025-10-13	510.00	2021.50
9	2025-10-14	200.00	2221.50
10	2025-10-15	504.00	2725.50
11	2025-10-16	550.00	3275.50
12	2025-10-17	380.00	3655.50
13	2025-10-18	300.00	3955.50
14	2025-10-19	65.00	4020.50
15	2025-10-20	40.00	4060.50
16	2025-10-21	20.00	4080.50
17	2025-10-22	50.00	4130.50
18	2025-10-23	150.00	4280.50
19	2025-10-24	125.00	4405.50
20	2025-10-25	75.00	4480.50
21	2025-10-26	100.00	4580.50
22	2025-10-27	35.00	4615.50
23	2025-10-28	50.00	4665.50
24	2025-10-29	68.00	4733.50
25	2025-10-30	110.00	4843.50

Query executed successfully.

Consulta 23: Suma el total vendido por día y calcula un acumulado progresivo de ventas usando la cláusula OVER, lo que permite observar la evolución diaria y el crecimiento total de los envíos.

24. Calcular promedio móvil para stock.



```sql
--CONSULTA 24 --
SELECT CodArticulo, CodLinea, StockActual,
       AVG(StockActual) OVER (PARTITION BY CodLinea ORDER BY CodArticulo ROWS 2 PRECEDING) AS PromedioMovil
FROM ARTICULO;
```

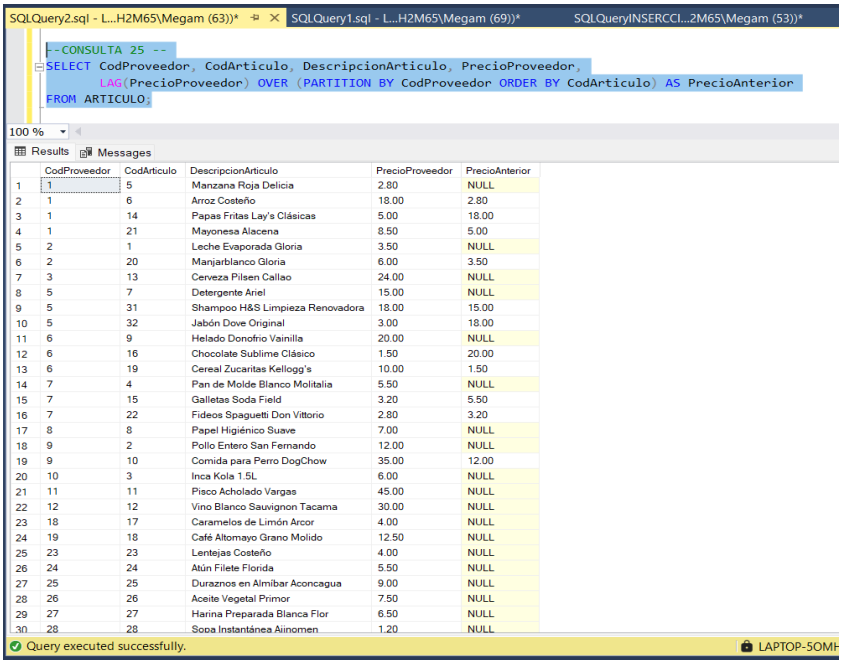
Results | Messages

	CodArticulo	CodLinea	StockActual	PromedioMovil
1	1	1	200	200
2	2	2	150	150
3	3	3	300	300
4	4	4	100	100
5	5	5	500	500
6	6	6	250	250
7	7	7	180	180
8	8	8	400	400
9	9	9	80	80
10	10	10	120	120
11	11	11	100	100
12	12	12	100	100
13	13	13	100	100
14	14	14	100	100
15	15	15	100	100
16	16	16	100	100
17	17	17	100	100
18	18	18	100	100
19	19	19	100	100
20	20	20	100	100
21	21	21	100	100
22	22	22	100	100
23	23	23	100	100
24	24	24	100	100
25	25	25	100	100
26	26	26	100	100
27	27	27	100	100
28	28	28	100	100
29	29	29	100	100
30	30	30	100	100

Query executed successfully.

Consulta 24: Obtiene un promedio móvil del stock actual por artículo dentro de cada línea considerando las dos filas anteriores, lo que permite analizar la tendencia o variación del inventario de forma continua.

25. Mostrar PrecioAnterior Mismo Proveedor usando LAG.



```
--CONSULTA 25 --
SELECT CodProveedor, CodArticulo, DescripcionArticulo, PrecioProveedor,
       LAG(PrecioProveedor) OVER (PARTITION BY CodProveedor ORDER BY CodArticulo) AS PrecioAnterior
FROM ARTICULO;
```

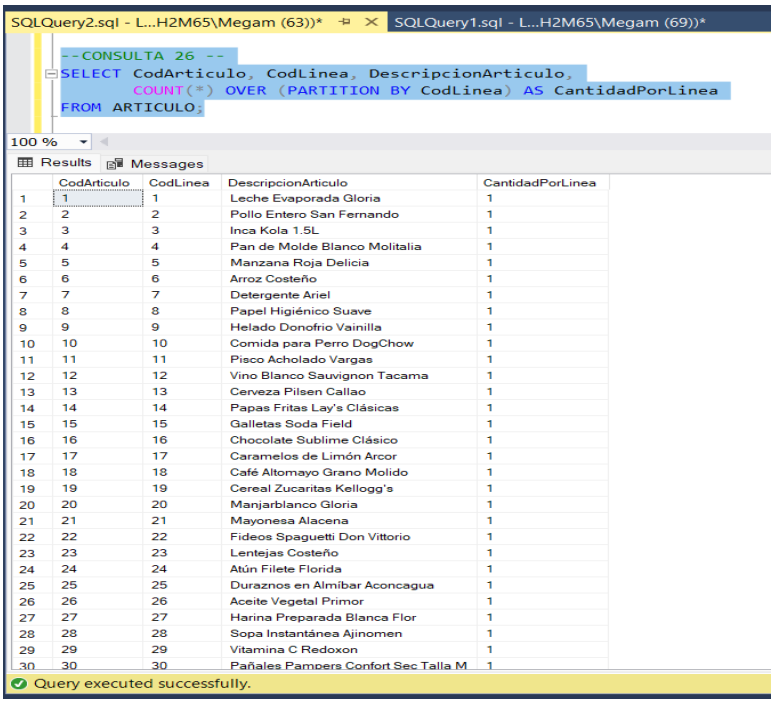
Results | Messages

	CodProveedor	CodArticulo	DescripcionArticulo	PrecioProveedor	PrecioAnterior
1	1	5	Manzana Roja Delicia	2.80	NULL
2	1	6	Arroz Costeño	18.00	2.80
3	1	14	Papas Fritas Lay's Clásicas	5.00	18.00
4	1	21	Mayonesa Alacena	8.50	5.00
5	2	1	Leche Evaporada Gloria	3.50	NULL
6	2	20	Manjarblanco Gloria	6.00	3.50
7	3	13	Cerveza Pilsen Callao	24.00	NULL
8	5	7	Detergente Ariel	15.00	NULL
9	5	31	Shampoo H&S Limpieza Renovadora	18.00	15.00
10	5	32	Jabón Dove Original	3.00	18.00
11	6	9	Helado Donofrio Vainilla	20.00	NULL
12	6	16	Chocolate Sublime Clásico	1.50	20.00
13	6	19	Cereal Zucaritas Kellogg's	10.00	1.50
14	7	4	Pan de Molde Blanco Molitalia	5.50	NULL
15	7	15	Galletas Soda Field	3.20	5.50
16	7	22	Fideos Spaguetti Don Vittorio	2.80	3.20
17	8	8	Papel Higiénico Suave	7.00	NULL
18	9	2	Pollo Entero San Fernando	12.00	NULL
19	9	10	Comida para Perro DogChow	35.00	12.00
20	10	3	Inca Kola 1.5L	6.00	NULL
21	11	11	Pisco Acholado Vargas	45.00	NULL
22	12	12	Vino Blanco Sauvignon Tacama	30.00	NULL
23	18	17	Caramelos de Limón Arcor	4.00	NULL
24	19	18	Café Altomayo Grano Molido	12.50	NULL
25	23	23	Lentejas Costeño	4.00	NULL
26	24	24	Atún Filete Florida	5.50	NULL
27	25	25	Duraznos en Almíbar Aconcagua	9.00	NULL
28	26	26	Aceite Vegetal Primor	7.50	NULL
29	27	27	Harina Preparada Blanca Flor	6.50	NULL
30	28	28	Sopa Instantánea Ajinomen	1.20	NULL

Query executed successfully.

Consulta 25: Muestra el precio del proveedor actual junto con el precio anterior del mismo proveedor utilizando la función LAG, facilitando la comparación de precios entre artículos consecutivos de un proveedor.

26. Añadir columna Cantidad PorLinea a cada artículo.



```
--CONSULTA 26 --
SELECT CodArticulo, CodLinea, DescripcionArticulo,
       COUNT(*) OVER (PARTITION BY CodLinea) AS CantidadPorLinea
FROM ARTICULO;
```

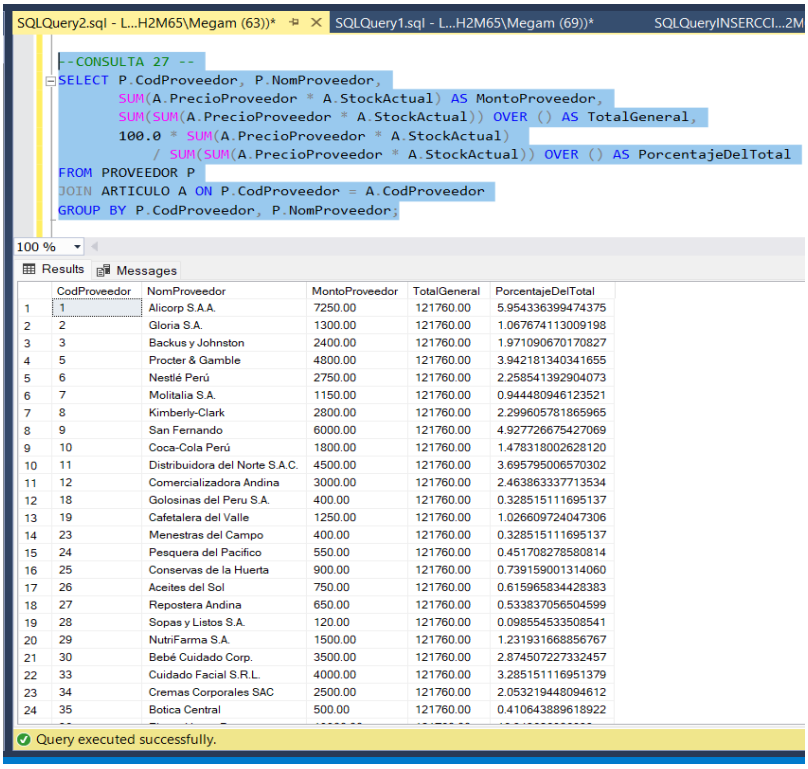
Results | Messages

	CodArticulo	CodLinea	DescripcionArticulo	CantidadPorLinea
1	1	1	Leche Evaporada Gloria	1
2	2	2	Pollo Entero San Fernando	1
3	3	3	Inca Kola 1.5L	1
4	4	4	Pan de Molde Blanco Molitalia	1
5	5	5	Manzana Roja Delicia	1
6	6	6	Arroz Costeño	1
7	7	7	Detergente Ariel	1
8	8	8	Papel Higiénico Suave	1
9	9	9	Helado Donofrio Vainilla	1
10	10	10	Comida para Perro DogChow	1
11	11	11	Pisco Acholado Vargas	1
12	12	12	Vino Blanco Sauvignon Tacama	1
13	13	13	Cerveza Pilsen Callao	1
14	14	14	Papas Fritas Lay's Clásicas	1
15	15	15	Galletas Soda Field	1
16	16	16	Chocolate Sublime Clásico	1
17	17	17	Caramelos de Limón Arcor	1
18	18	18	Café Altomayo Grano Molido	1
19	19	19	Cereal Zucaritas Kellogg's	1
20	20	20	Manjarblanco Gloria	1
21	21	21	Mayonesa Alacena	1
22	22	22	Fideos Spaguetti Don Vittorio	1
23	23	23	Lentejas Costeño	1
24	24	24	Atún Filete Florida	1
25	25	25	Duraznos en Almíbar Aconcagua	1
26	26	26	Aceite Vegetal Primor	1
27	27	27	Harina Preparada Blanca Flor	1
28	28	28	Sopa Instantánea Ajinomen	1
29	29	29	Vitamina C Redoxon	1
30	30	30	Pañales Pampers Confort Sec Talla M	1

Query executed successfully.

Consulta 26: Cuenta cuántos artículos pertenecen a cada línea sin agrupar los datos, gracias al uso de COUNT(*) OVER, lo que permite ver el detalle de cada artículo junto con la cantidad total de su línea.

27. Mostrar MontoProveedor y Porcentaje DelTotal.



```sql
--CONSULTA 27 --
SELECT P.CodProveedor, P.NomProveedor,
       SUM(A.PrecioProveedor * A.StockActual) AS MontoProveedor,
       SUM(SUM(A.PrecioProveedor * A.StockActual)) OVER () AS TotalGeneral,
       100.0 * SUM(A.PrecioProveedor * A.StockActual)
           / SUM(SUM(A.PrecioProveedor * A.StockActual)) OVER () AS PorcentajeDelTotal
FROM PROVEEDOR P
JOIN ARTICULO A ON P.CodProveedor = A.CodProveedor
GROUP BY P.CodProveedor, P.NomProveedor;
```

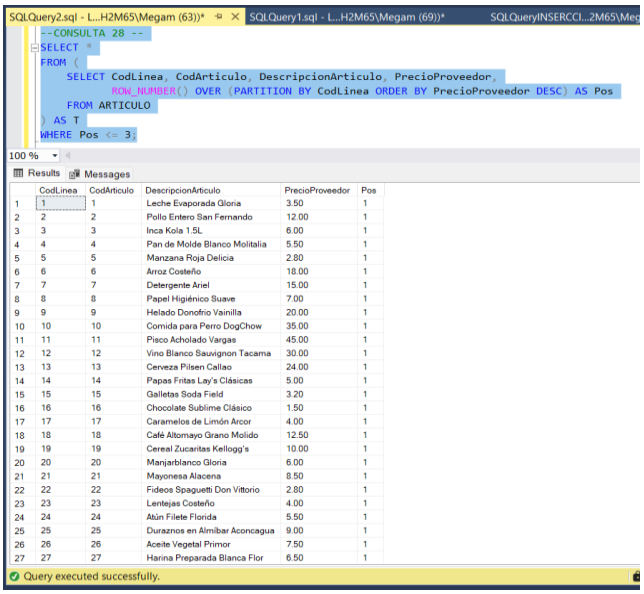
Results | Messages

	CodProveedor	NomProveedor	MontoProveedor	TotalGeneral	PorcentajeDelTotal
1	1	Alicorp S.A.A.	7250.00	121760.00	5.954336399474375
2	2	Gloria S.A.	1300.00	121760.00	1.067674113009198
3	3	Backus y Johnston	2400.00	121760.00	1.971090670170827
4	5	Procter & Gamble	4800.00	121760.00	3.942181340341655
5	6	Nestlé Perú	2750.00	121760.00	2.258541392904073
6	7	Molitalia S.A.	1150.00	121760.00	0.944480946123521
7	8	Kimberly-Clark	2800.00	121760.00	2.299605781865965
8	9	San Fernando	6000.00	121760.00	4.927726675427069
9	10	Coca-Cola Perú	1800.00	121760.00	1.478318002628120
10	11	Distribuidora del Norte S.A.C.	4500.00	121760.00	3.695795006570302
11	12	Comercializadora Andina	3000.00	121760.00	2.463863337713534
12	18	Golosinas del Peru S.A.	400.00	121760.00	0.328515111695137
13	19	Cafetalera del Valle	1250.00	121760.00	1.026609724047306
14	23	Menestras del Campo	400.00	121760.00	0.328515111695137
15	24	Pesquera del Pacifico	550.00	121760.00	0.451708278580814
16	25	Conservas de la Huerta	900.00	121760.00	0.739159001314060
17	26	Aceites del Sol	750.00	121760.00	0.615965834428383
18	27	Repostera Andina	650.00	121760.00	0.533837056504599
19	28	Sopas y Listos S.A.	120.00	121760.00	0.098554533508541
20	29	NutriFarma S.A.	1500.00	121760.00	1.231931668856767
21	30	Bebé Cuidado Corp.	3500.00	121760.00	2.874507227332457
22	33	Cuidado Facial S.R.L.	4000.00	121760.00	3.285151116951379
23	34	Cremas Corporales SAC	2500.00	121760.00	2.053219448094612
24	35	Botica Central	500.00	121760.00	0.410643889618922

Query executed successfully.

Consulta 27: Calcula el valor total de inventario por proveedor y el porcentaje que representa respecto al total general, lo que ayuda a conocer qué proveedor aporta mayor valor económico al inventario total.

28. Mostrar solo los 3 artículos más caros por línea.



```sql
--CONSULTA 28 --
SELECT *
FROM (
    SELECT CodLinea, CodArticulo, DescripcionArticulo, PrecioProveedor,
           ROW_NUMBER() OVER (PARTITION BY CodLinea ORDER BY PrecioProveedor DESC) AS Pos
    FROM ARTICULO
) AS T
WHERE Pos <= 3;
```

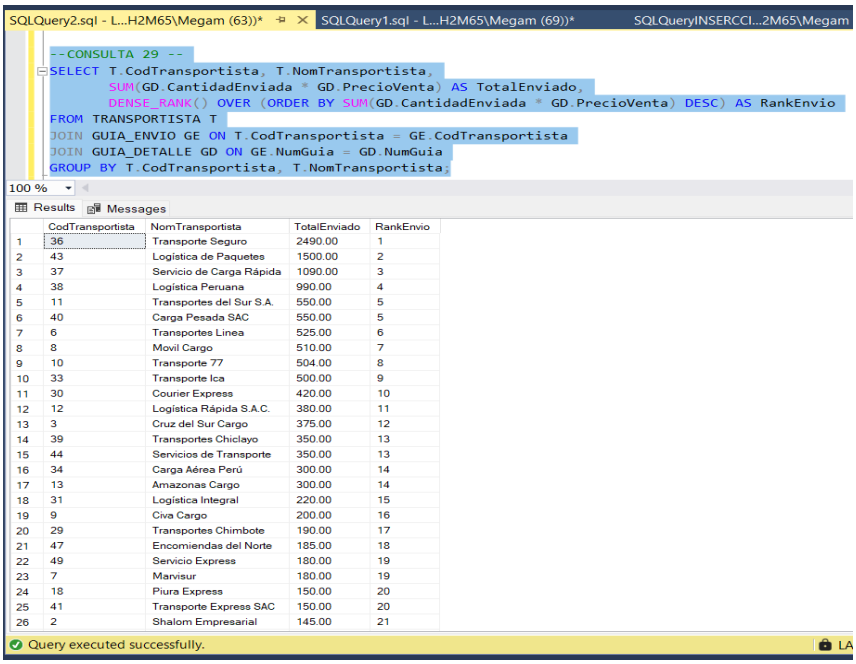
Results | Messages

	CodLinea	CodArticulo	DescripcionArticulo	PrecioProveedor	Pos
1	1	1	Leche Evaporada Gloria	3.50	1
2	2	2	Pollo Entero San Fernando	12.00	1
3	3	3	Inca Kola 1.5L	6.00	1
4	4	4	Pan de Molde Blanco Molitalia	5.50	1
5	5	5	Manzana Roja Delicia	2.80	1
6	6	6	Arroz Costeño	18.00	1
7	7	7	Detergente Ariel	15.00	1
8	8	8	Papel Higiénico Suave	7.00	1
9	9	9	Helado Donofrio Vainilla	20.00	1
10	10	10	Comida para Perro DogChow	35.00	1
11	11	11	Pisco Acholado Vargas	45.00	1
12	12	12	Vino Blanco Sauvignon Tacama	30.00	1
13	13	13	Cerveza Pilsen Callao	24.00	1
14	14	14	Papas Fritas Lay's Clásicas	5.00	1
15	15	15	Galletas Soda Field	3.20	1
16	16	16	Chocolate Sublime Clásico	1.50	1
17	17	17	Caramelos de Limón Arcor	4.00	1
18	18	18	Café Altomayo Grano Molido	12.50	1
19	19	19	Cereal Zucaritas Kellogg's	10.00	1
20	20	20	Manjarblanco Gloria	6.00	1
21	21	21	Mayonesa Alacena	8.50	1
22	22	22	Fideos Spaguetti Don Vittorio	2.80	1
23	23	23	Lentejas Costeño	4.00	1
24	24	24	Atún Filete Florida	5.50	1
25	25	25	Duraznos en Almíbar Aconcagua	9.00	1
26	26	26	Aceite Vegetal Primor	7.50	1
27	27	27	Harina Preparada Blanca Flor	6.50	1

Query executed successfully.

Consulta 28: Selecciona los tres artículos más caros dentro de cada línea utilizando ROW_NUMBER() y filtrando las primeras posiciones, con el fin de identificar los productos de mayor precio por categoría.

29. **Mostrar transportista y su DenseRank por TotalEnviado.**



```sql
--CONSULTA 29 --
SELECT T.CodTransportista, T.NomTransportista,
       SUM(GD.CantidadEnviada * GD.PrecioVenta) AS TotalEnviado,
       DENSE_RANK() OVER (ORDER BY SUM(GD.CantidadEnviada * GD.PrecioVenta) DESC) AS RankEnvio
FROM TRANSPORTISTA T
JOIN GUIA_ENVIO GE ON T.CodTransportista = GE.CodTransportista
JOIN GUIA_DETALLE GD ON GE.NumGuia = GD.NumGuia
GROUP BY T.CodTransportista, T.NomTransportista;
```

100 %

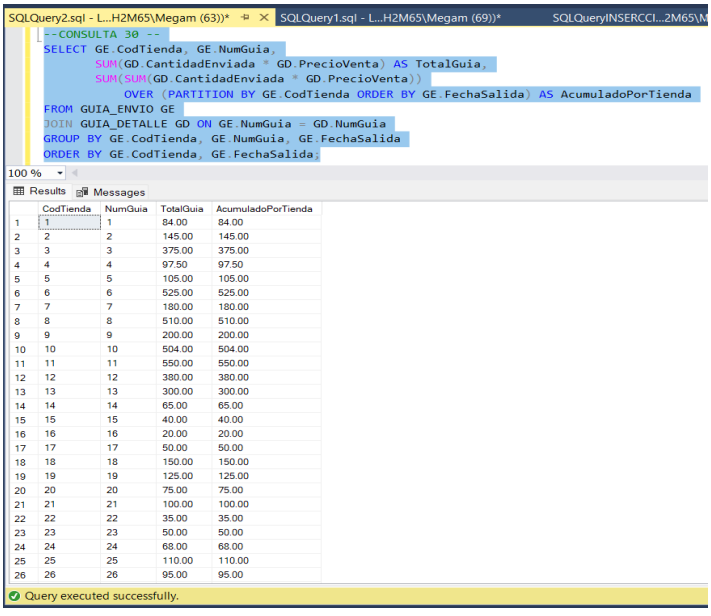
Results | Messages

	CodTransportista	NomTransportista	TotalEnviado	RankEnvio
1	36	Transporte Seguro	2490.00	1
2	43	Logística de Paquetes	1500.00	2
3	37	Servicio de Carga Rápida	1090.00	3
4	38	Logística Peruana	990.00	4
5	11	Transportes del Sur S.A.	550.00	5
6	40	Carga Pesada SAC	550.00	5
7	6	Transportes Linea	525.00	6
8	8	Movil Cargo	510.00	7
9	10	Transporte 77	504.00	8
10	33	Transporte Ica	500.00	9
11	30	Courier Express	420.00	10
12	12	Logística Rápida S.A.C.	380.00	11
13	3	Cruz del Sur Cargo	375.00	12
14	39	Transportes Chiclayo	350.00	13
15	44	Servicios de Transporte	350.00	13
16	34	Carga Aérea Perú	300.00	14
17	13	Amazonas Cargo	300.00	14
18	31	Logística Integral	220.00	15
19	9	Civa Cargo	200.00	16
20	29	Transportes Chimbote	190.00	17
21	47	Encomiendas del Norte	185.00	18
22	49	Servicio Express	180.00	19
23	7	Marvisur	180.00	19
24	18	Piura Express	150.00	20
25	41	Transporte Express SAC	150.00	20
26	2	Shalom Empresarial	145.00	21

Query executed successfully.

Consulta 29: Suma el total de productos enviados por cada transportista y les asigna una posición con DENSE_RANK() según el valor total despachado, lo que permite clasificar a los transportistas según su volumen de envíos.

30. **Mostrar por guía la suma acumulada por tienda hasta esa guía (ordenada por FechaSalida).**



```sql
--CONSULTA 30 --
SELECT GE.CodTienda, GE.NumGuia,
       SUM(GD.CantidadEnviada * GD.PrecioVenta) AS TotalGuia,
       SUM(SUM(GD.CantidadEnviada * GD.PrecioVenta))
           OVER (PARTITION BY GE.CodTienda ORDER BY GE.FechaSalida) AS AcumuladoPorTienda
FROM GUIA_ENVIO GE
JOIN GUIA_DETALLE GD ON GE.NumGuia = GD.NumGuia
GROUP BY GE.CodTienda, GE.NumGuia, GE.FechaSalida
ORDER BY GE.CodTienda, GE.FechaSalida;
```

100 %

Results | Messages

	CodTienda	NumGuia	TotalGuia	AcumuladoPorTienda
1	1	1	84.00	84.00
2	2	2	145.00	145.00
3	3	3	375.00	375.00
4	4	4	97.50	97.50
5	5	5	105.00	105.00
6	6	6	525.00	525.00
7	7	7	180.00	180.00
8	8	8	510.00	510.00
9	9	9	200.00	200.00
10	10	10	504.00	504.00
11	11	11	550.00	550.00
12	12	12	380.00	380.00
13	13	13	300.00	300.00
14	14	14	65.00	65.00
15	15	15	40.00	40.00
16	16	16	20.00	20.00
17	17	17	50.00	50.00
18	18	18	150.00	150.00
19	19	19	125.00	125.00
20	20	20	75.00	75.00
21	21	21	100.00	100.00
22	22	22	35.00	35.00
23	23	23	50.00	50.00
24	24	24	68.00	68.00
25	25	25	110.00	110.00
26	26	26	95.00	95.00

Query executed successfully.

Consulta 30: Calcula el total de cada guía de envío y un acumulado por tienda en orden de salida, mostrando cómo crece el monto total de despachos por tienda a lo largo del tiempo.

OPERADOR PIVOT

IV. Explica de manera clara y didáctica qué son la OPERADOR PIVOT en SQL y cómo se utilizan.

El operador **PIVOT** en SQL es un mecanismo de transformación tabular que permite convertir valores de una columna en encabezados de nuevas columnas, aplicando simultáneamente una función de agregación sobre un conjunto de datos. Su propósito principal es reorganizar datos orientados en filas hacia una estructura orientada en columnas, facilitando el análisis comparativo y la lectura de reportes.

Comportamiento interno

1. **Selección del eje de filas.** Se identifica la columna que conservará las filas finales del resultado.
2. **Selección del eje de columnas.** Se identifica la columna cuyos valores discretos serán transformados en encabezados de nuevas columnas.
3. **Aplicación de la función de agregación.** Dado que múltiples filas pueden coincidir en un mismo par (fila, categoría), se requiere una función agregadora para consolidar los valores. SQL Server permite SUM, AVG, MIN, MAX, COUNT y otras funciones agregadas.
4. **Construcción del conjunto rectangular.** El motor de SQL expande horizontalmente la tabla creando tantas columnas como categorías definidas en la cláusula IN, asignando a cada celda el resultado de la agregación correspondiente.
5. **Relleno con valores nulos.** Si para alguna combinación fila–columna no existe dato en la fuente, el resultado en esa celda será NULL, porque no hay información agregable.

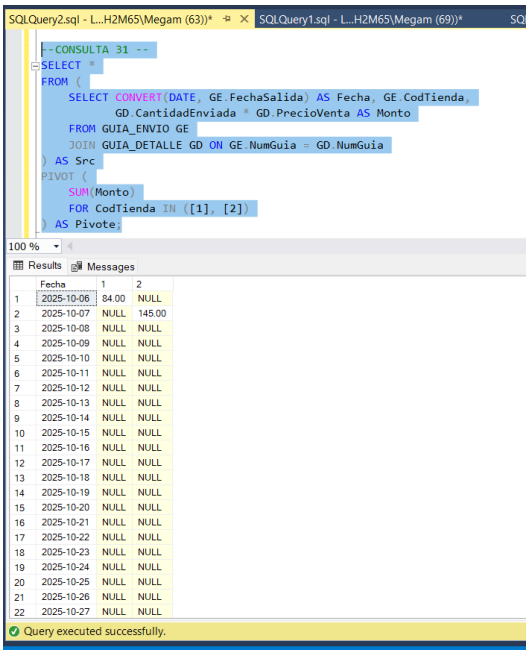
Ventajas conceptualizadas

- **Reorganización semántica:** permite visualizar categorías como atributos, facilitando la interpretación.
- **Reducción de complejidad analítica:** elimina la necesidad de múltiples agregaciones condicionadas.
- **Optimización de reportes:** genera estructuras tabulares aptas para análisis multidimensional y comparación horizontal.

Aplicación conceptual en análisis de datos

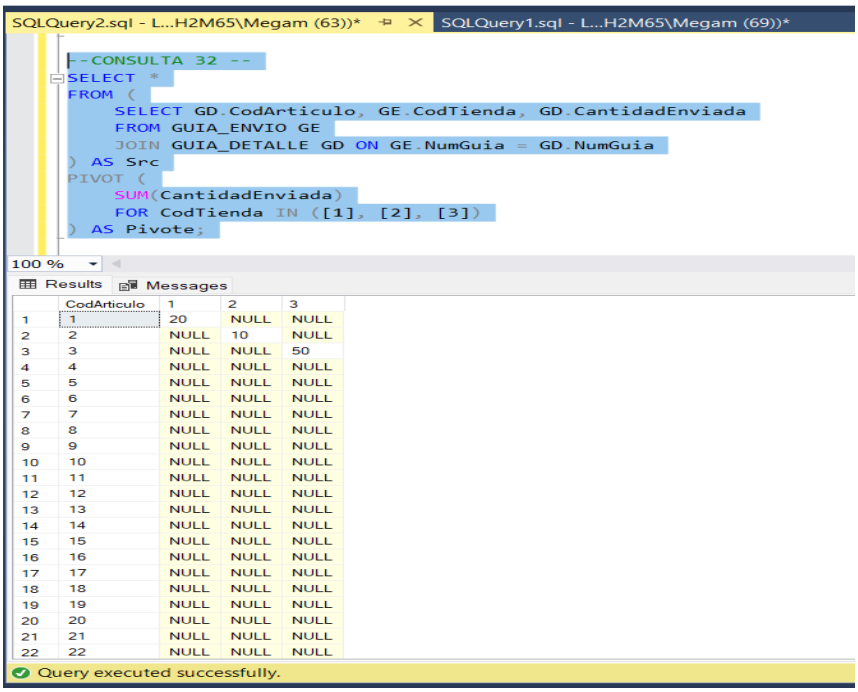
PIVOT cumple el rol de transformar datos semiestructurados en matrices analíticas. Funciona como un puente entre modelos transaccionales orientados a registro atomizado y modelos analíticos orientados a comparación categorial.

31. **Mostrar Fecha y columnas CodTienda_1, CodTienda_2, con TotalEnviado por día.**



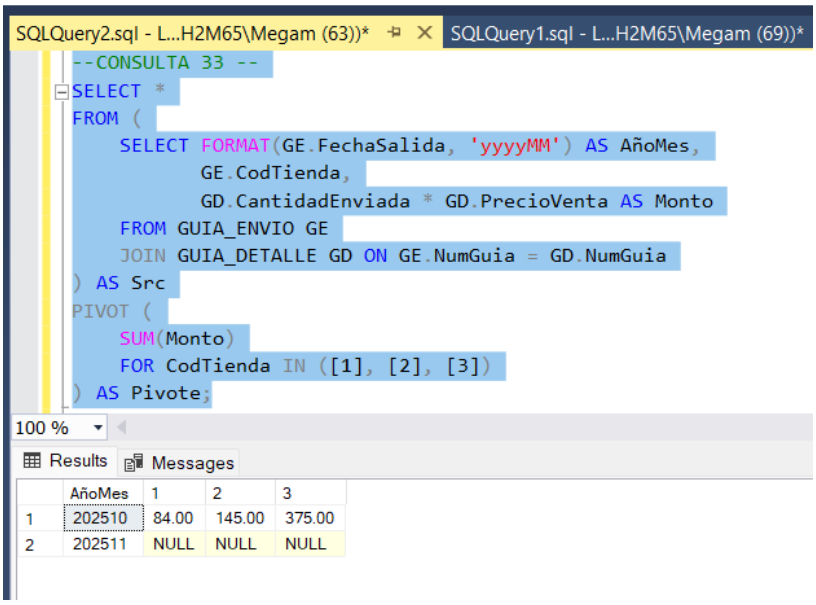
Consulta 31: Convierte las tiendas 1 y 2 en columnas y muestra cuánto monto (cantidad × precio) se envió en cada fecha. La fecha se convierte en fila y cada tienda muestra el monto total enviado ese día.

32. **Enunciado: Mostrar CodArticulo y columnas con cantidades por tienda 1..3.**



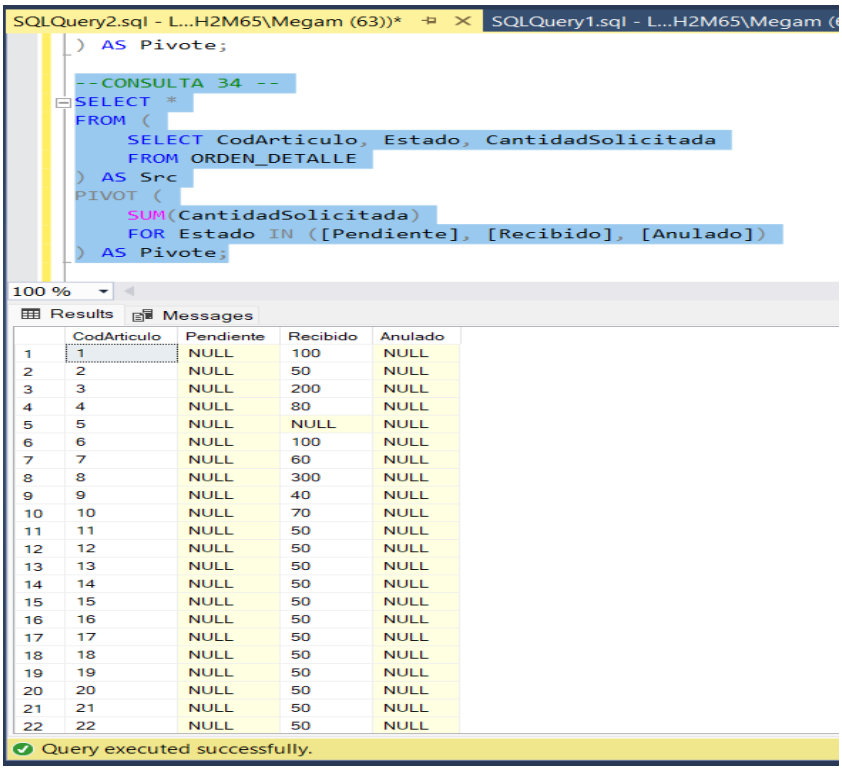
Consulta 32: Suma la cantidad enviada de cada artículo y la organiza por tienda, revelando cuántas unidades de cada producto fueron enviadas a cada tienda.

33. **Enunciado: Mostrar AñoMes y tiendas como columnas con suma PrecioVenta×Cantidad. De**



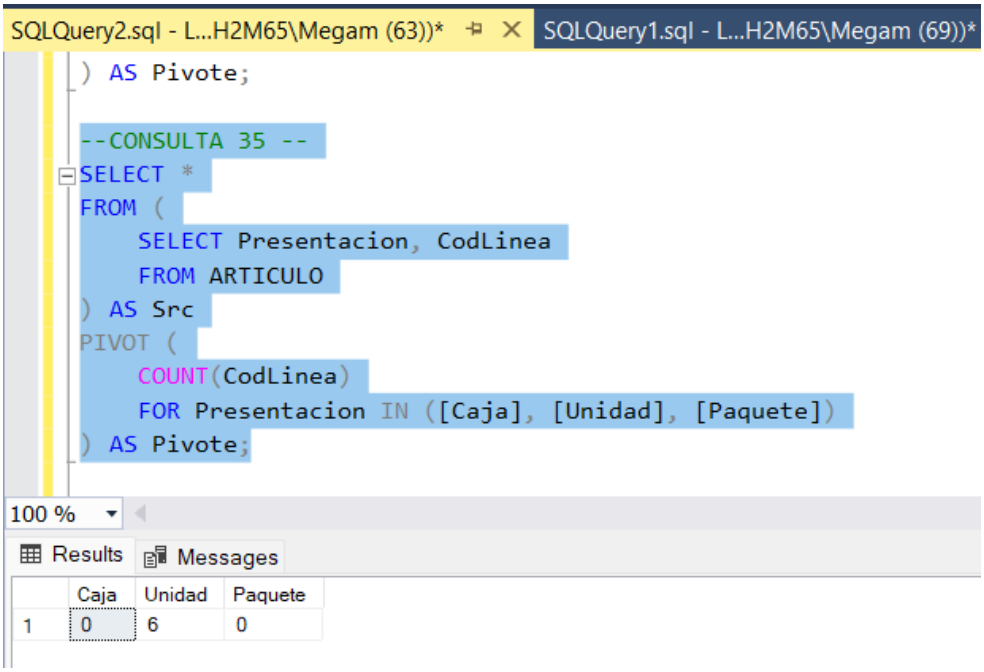
Consulta 33: Obtiene el monto mensual enviado a cada tienda y lo pivotea por código de tienda, permitiendo ver cuánto dinero recibió cada tienda en cada mes del año.

34. **Enunciado: Mostrar CodArticulo con columnas para cada Estado.**



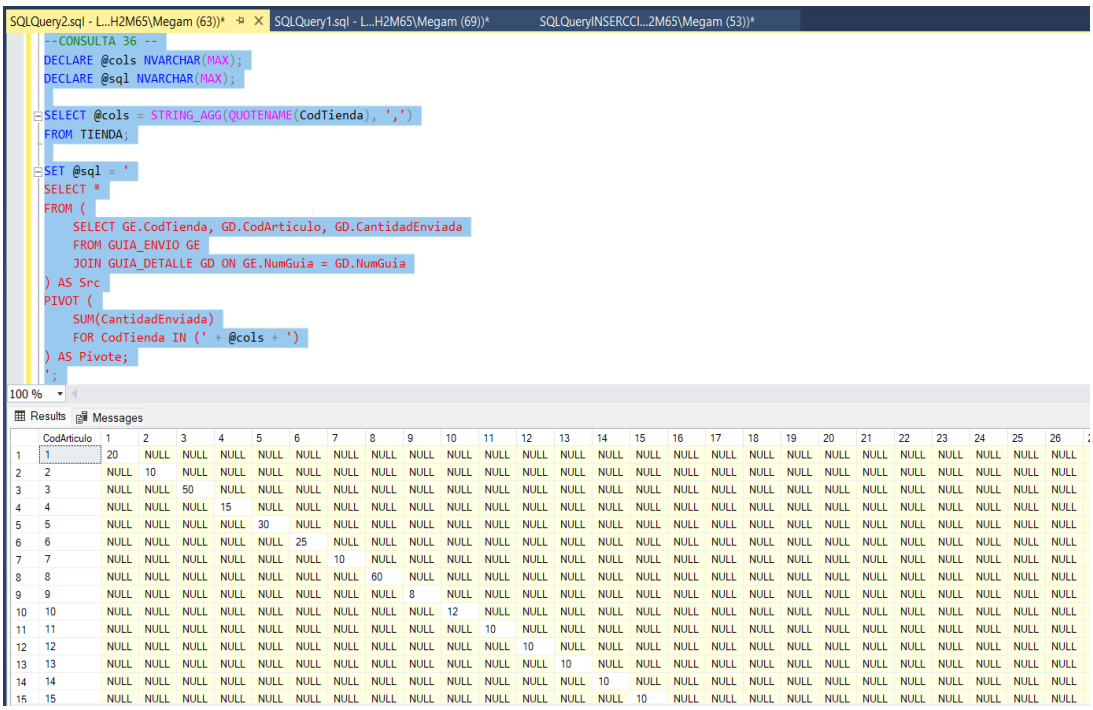
Consulta 34: Resume la cantidad solicitada por artículo según el estado de la orden, mostrando cuántas unidades están pendientes, recibidas o anuladas para cada producto.

35. Contar artículos por presentación pivotada.



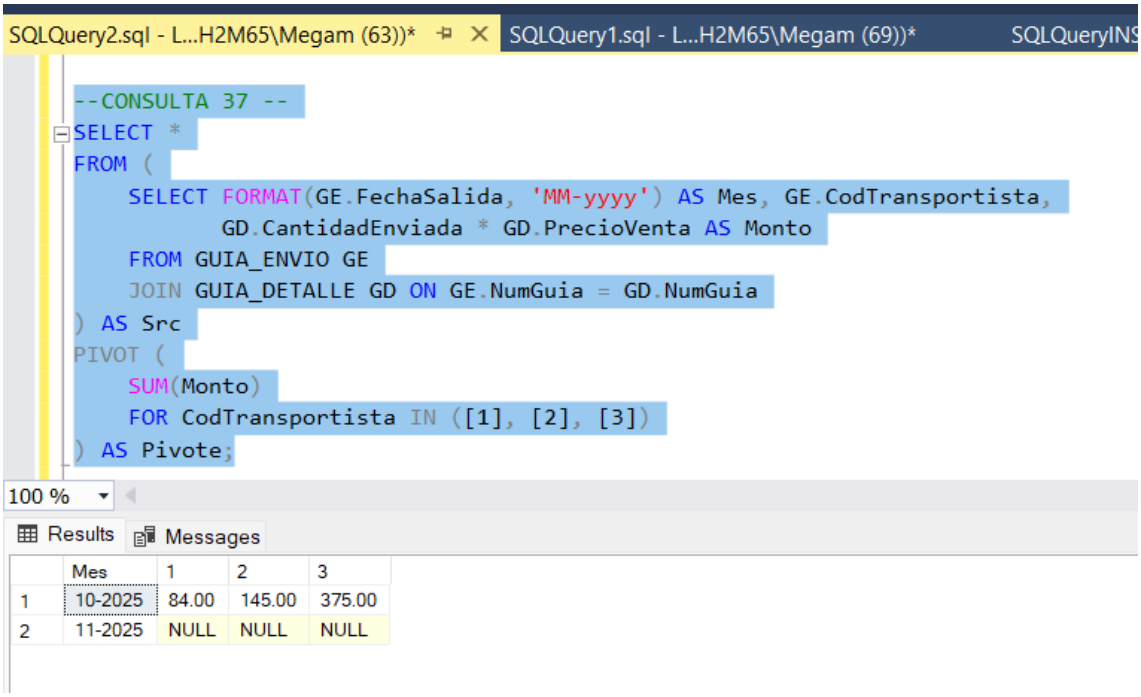
Consulta 35: Cuenta cuántos artículos por línea pertenecen a cada tipo de presentación, permitiendo ver la distribución de presentaciones como Caja, Unidad y Paquete dentro de cada línea de producto.

36. Generar PIVOT dinámico para todas las tiendas (ejemplo de patrón).



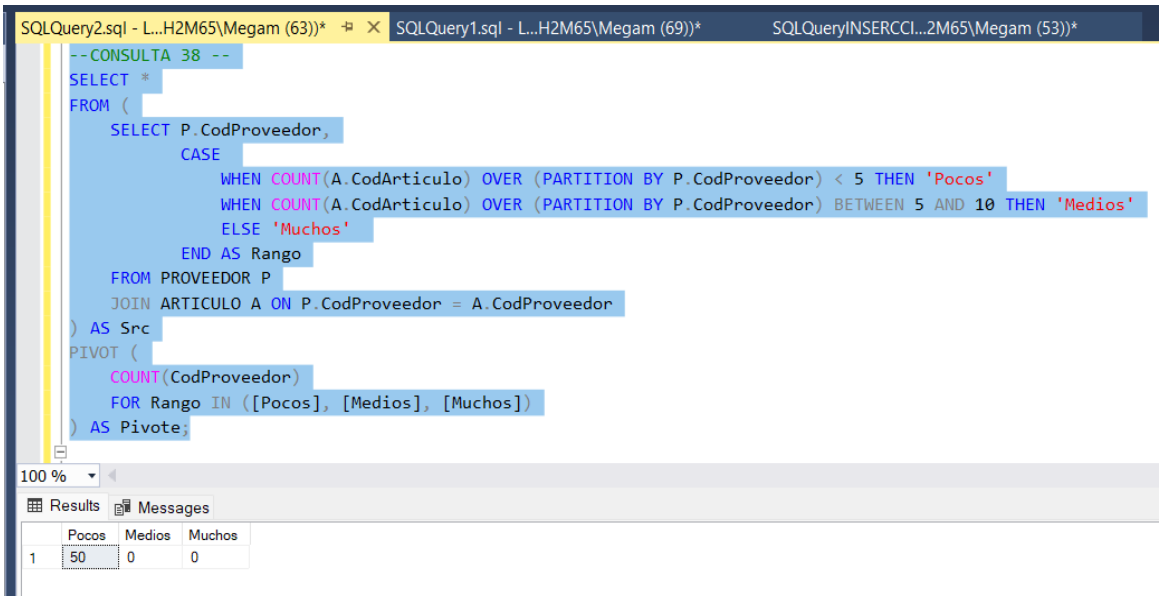
Consulta 36: Genera un pivote automático que crea columnas para todas las tiendas existentes y muestra la cantidad enviada por artículo para cada tienda sin necesidad de definir los códigos manualmente.

37. Mostrar mes y columnas por transportista con totales.



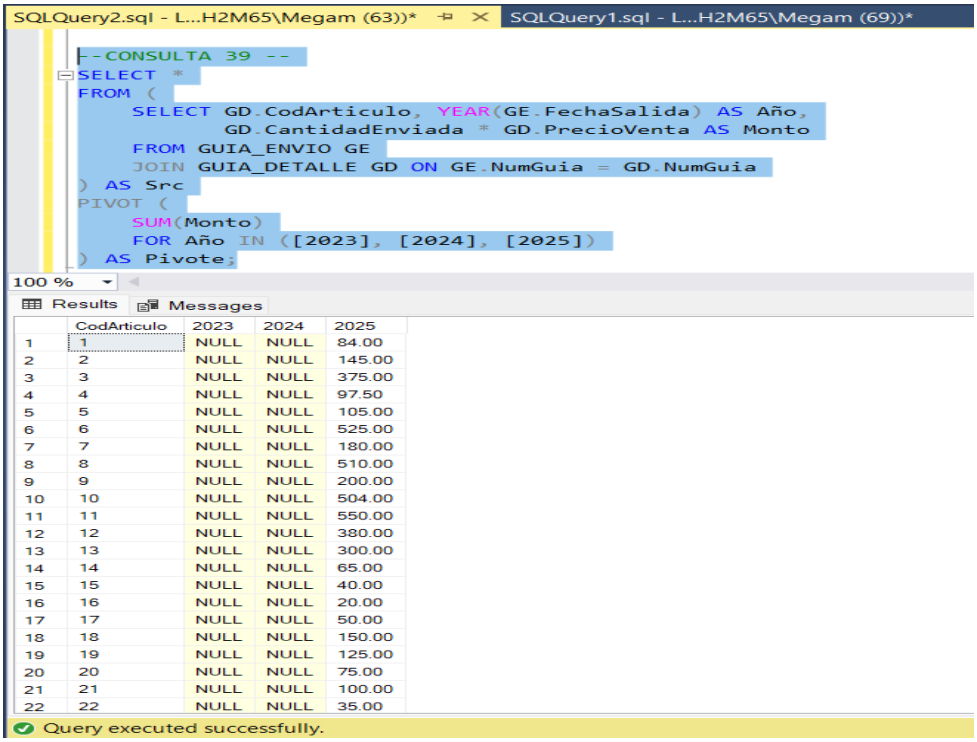
Consulta 37: Calcula el monto transportado por cada empresa de transporte en cada mes, permitiendo comparar el rendimiento mensual de los transportistas.

38. Contar proveedores por rango de variedad de artículos pivotado como columnas.



Consulta 38: Clasifica a los proveedores según la cantidad de artículos que venden (Pocos, Medios o Muchos) y luego muestra cuántos proveedores pertenecen a cada rango.

39. Mostrar CodArticulo y columnas por año con monto total vendido.



```
--CONSULTA 39 --
SELECT *
FROM (
    SELECT GD.CodArticulo, YEAR(GE.FechaSalida) AS Año,
           GD.CantidadEnviada * GD.PrecioVenta AS Monto
    FROM GUIA_ENVIO GE
    JOIN GUIA_DETALLE GD ON GE.NumGuia = GD.NumGuia
) AS Src
PIVOT (
    SUM(Monto)
    FOR Año IN ([2023], [2024], [2025])
) AS Pivote;
```

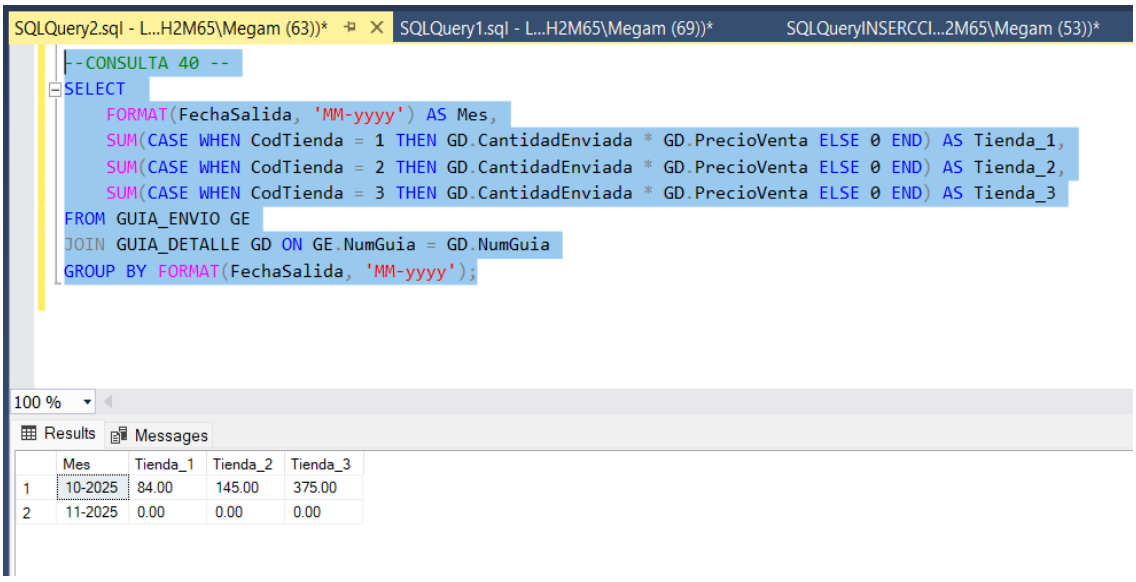
Results | Messages

	CodArticulo	2023	2024	2025
1	1	NULL	NULL	84.00
2	2	NULL	NULL	145.00
3	3	NULL	NULL	375.00
4	4	NULL	NULL	97.50
5	5	NULL	NULL	105.00
6	6	NULL	NULL	525.00
7	7	NULL	NULL	180.00
8	8	NULL	NULL	510.00
9	9	NULL	NULL	200.00
10	10	NULL	NULL	504.00
11	11	NULL	NULL	550.00
12	12	NULL	NULL	380.00
13	13	NULL	NULL	300.00
14	14	NULL	NULL	65.00
15	15	NULL	NULL	40.00
16	16	NULL	NULL	20.00
17	17	NULL	NULL	50.00
18	18	NULL	NULL	150.00
19	19	NULL	NULL	125.00
20	20	NULL	NULL	75.00
21	21	NULL	NULL	100.00
22	22	NULL	NULL	35.00

Query executed successfully.

Consulta 39: Muestra el monto anual enviado por cada artículo, revelando cómo varían las ventas de cada producto entre los años 2023, 2024 y 2025.

40. Mostrar Mes y columnas por tienda (CASE alternative).



```
--CONSULTA 40 --
SELECT
    FORMAT(FechaSalida, 'MM-yyyy') AS Mes,
    SUM(CASE WHEN CodTienda = 1 THEN GD.CantidadEnviada * GD.PrecioVenta ELSE 0 END) AS Tienda_1,
    SUM(CASE WHEN CodTienda = 2 THEN GD.CantidadEnviada * GD.PrecioVenta ELSE 0 END) AS Tienda_2,
    SUM(CASE WHEN CodTienda = 3 THEN GD.CantidadEnviada * GD.PrecioVenta ELSE 0 END) AS Tienda_3
FROM GUIA_ENVIO GE
JOIN GUIA_DETALLE GD ON GE.NumGuia = GD.NumGuia
GROUP BY FORMAT(FechaSalida, 'MM-yyyy');
```

Results | Messages

	Mes	Tienda_1	Tienda_2	Tienda_3
1	10-2025	84.00	145.00	375.00
2	11-2025	0.00	0.00	0.00

Consulta 40: Presenta el monto enviado por mes a cada tienda utilizando CASE en lugar de PIVOT, produciendo un resumen mensual de envíos por tienda.

CLÁUSULA HAVING

V. Explica de manera clara y didáctica qué son la CLÁUSULA HAVING en SQL y cómo se utilizan.

HAVING es una cláusula diseñada para restringir los resultados de una consulta agrupada.

Mientras que WHERE filtra filas individuales antes de agrupar, HAVING filtra grupos completos después del agrupamiento.

Se utiliza únicamente en consultas que incluyen funciones de agregación o GROUP BY.

HAVING siempre opera sobre grupos, no sobre filas individuales.

Cuando se utiliza

Se usa HAVING cuando:

1. Necesitas aplicar una condición a un resultado agregado.
2. Quieres mostrar solo los grupos que cumplen un requisito mínimo o máximo.
3. Debes eliminar grupos incompletos o no representativos.

Funcionamiento conceptual

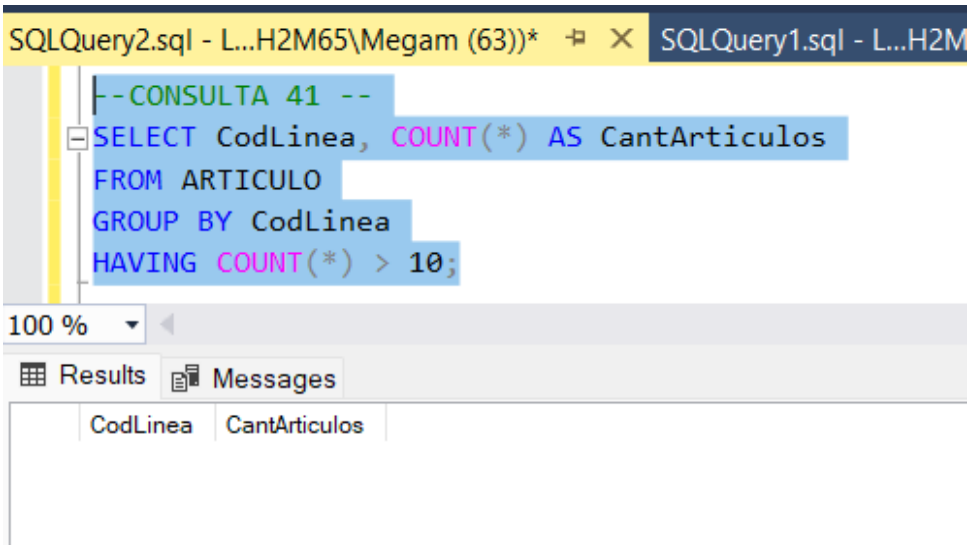
Paso 1: Se filtran filas individuales con WHERE (si existe).
Paso 2: Se agrupan los datos con GROUP BY.
Paso 3: Se calculan las funciones de agregación.
Paso 4: HAVING elimina grupos agregados que no cumplen la condición.
Paso 5: El SELECT presenta los grupos restantes.

Beneficio práctico

HAVING permite, entre otros:

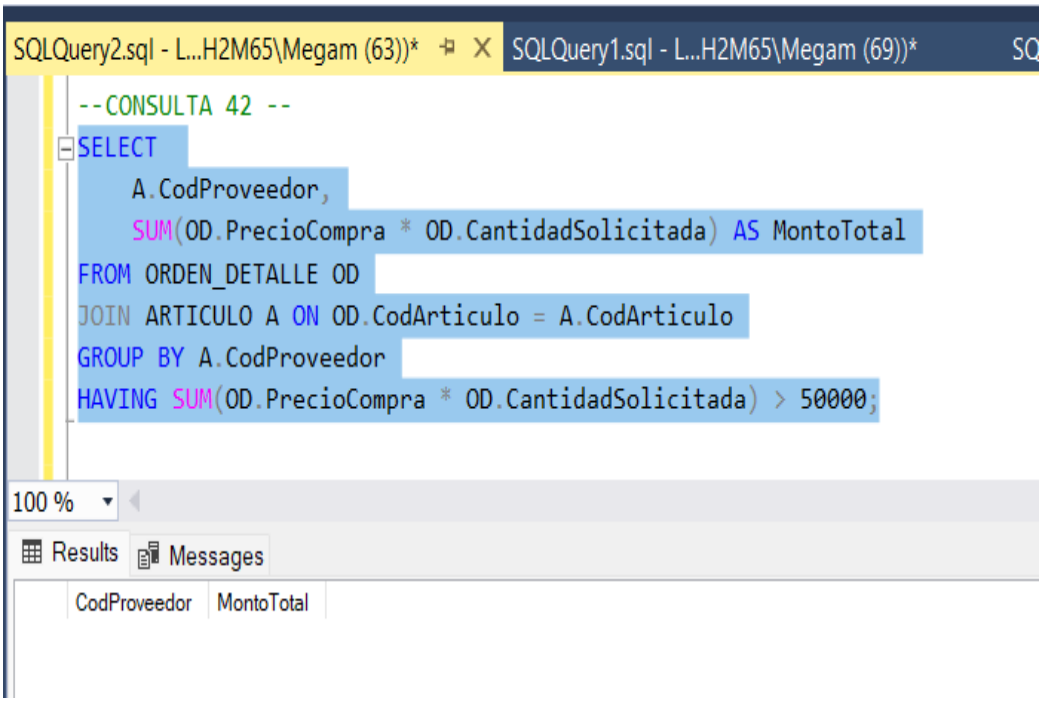
- Identificar tiendas con grandes montos de venta acumulada.
- Filtrar proveedores con poco stock.
- Seleccionar artículos con alta demanda agregada.
- Excluir grupos irrelevantes en reportes.

41. **Mostrar CodLinea y CantArticulos donde CantArticulos > 10.**



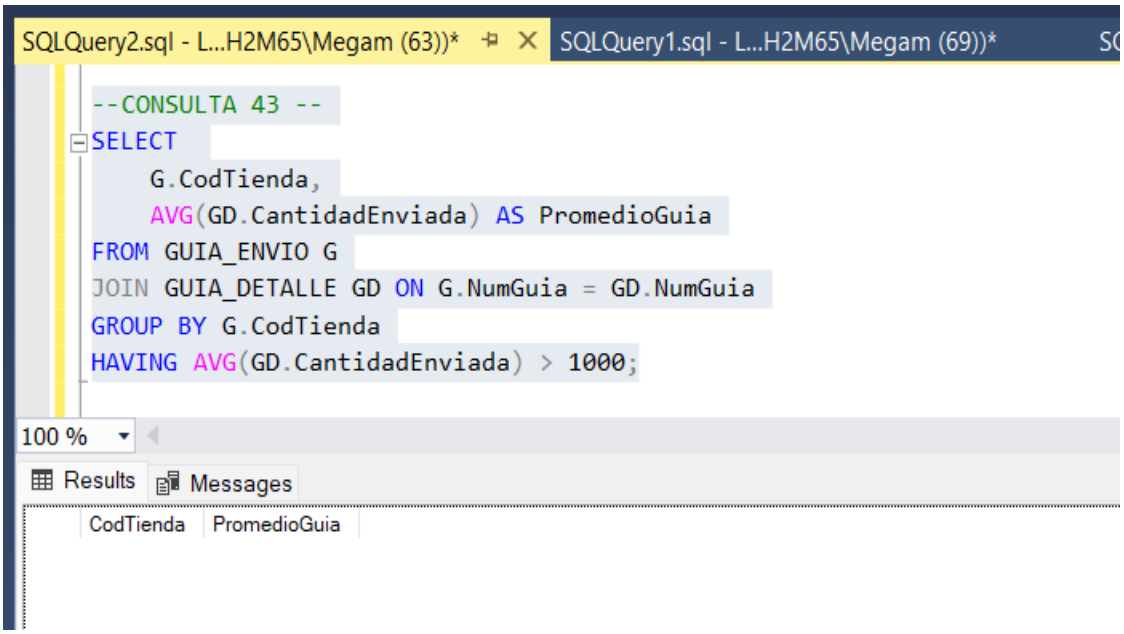
Consulta 41: Muestra solo las líneas de productos que tienen más de 10 artículos registrados, filtrando los grupos mediante HAVING.

42. **Mostrar CodProveedor y MontoTotal donde MontoTotal > 50000.**



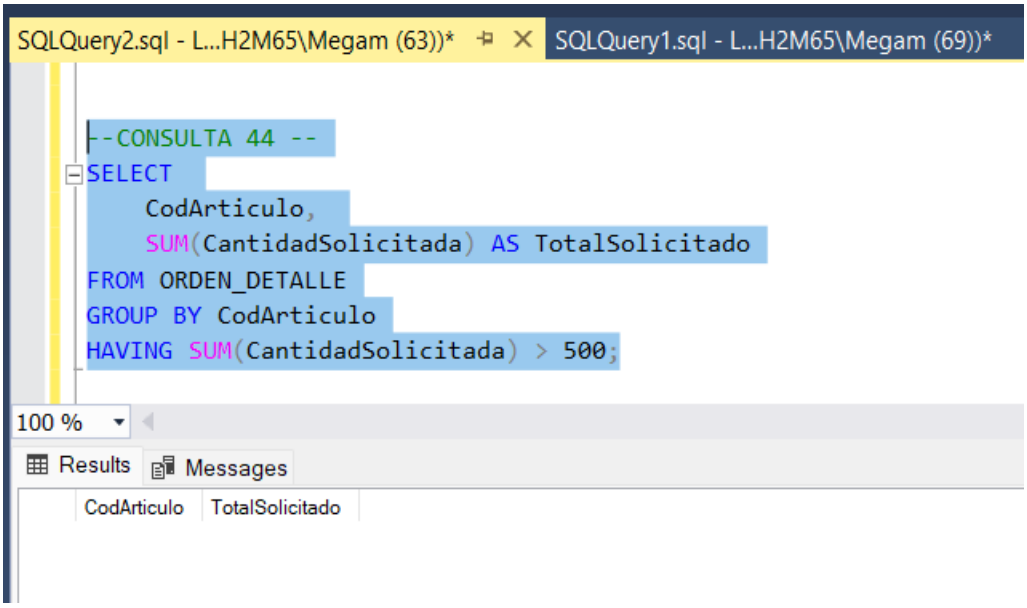
Consulta 42: Calcula el monto total comprado por proveedor y muestra únicamente a los proveedores cuya suma supera 50 000.

43. Mostrar CodTienda y PromedioGuia donde PromedioGuia>1000.



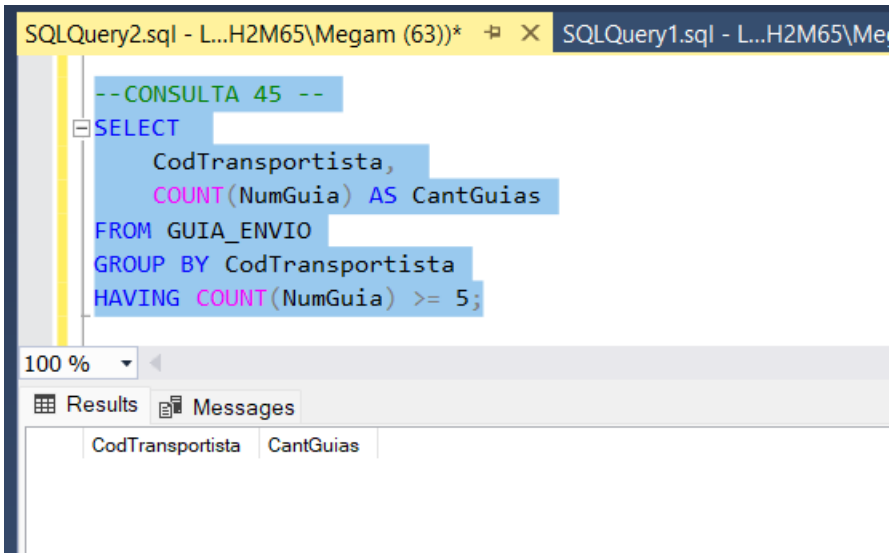
Consulta 43: Calcula el promedio de cantidad enviada por tienda y muestra únicamente las tiendas que superan un promedio de 1000 unidades enviadas por guía.

44. Mostrar CodArticulo y TotalSolicitado>500.



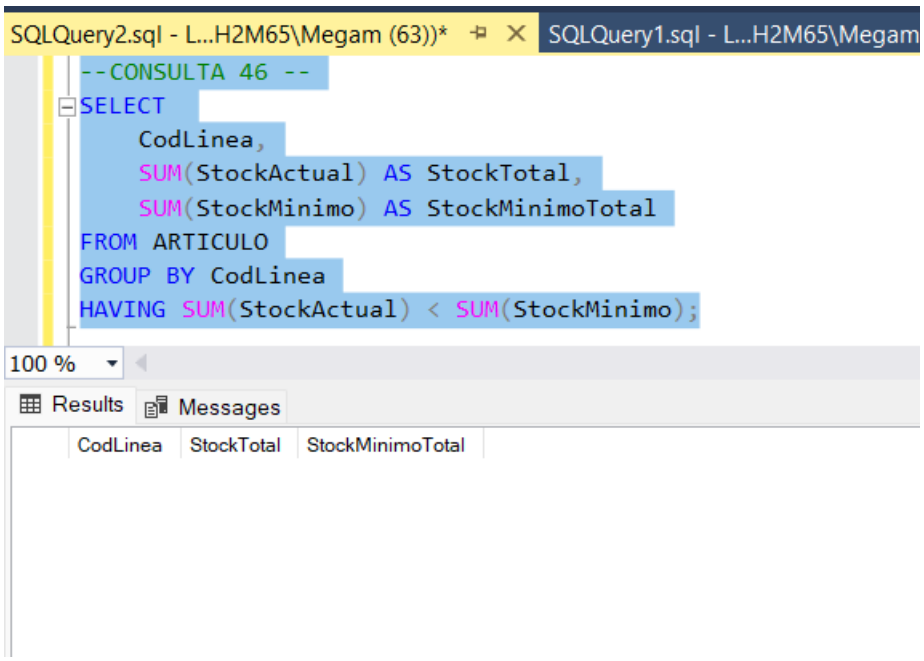
Consulta 44: Suma la cantidad solicitada por artículo y muestra solo aquellos artículos cuya solicitud total supera 500 unidades.

45. **Mostrar CodTransportista y CantGuías >= 5.**



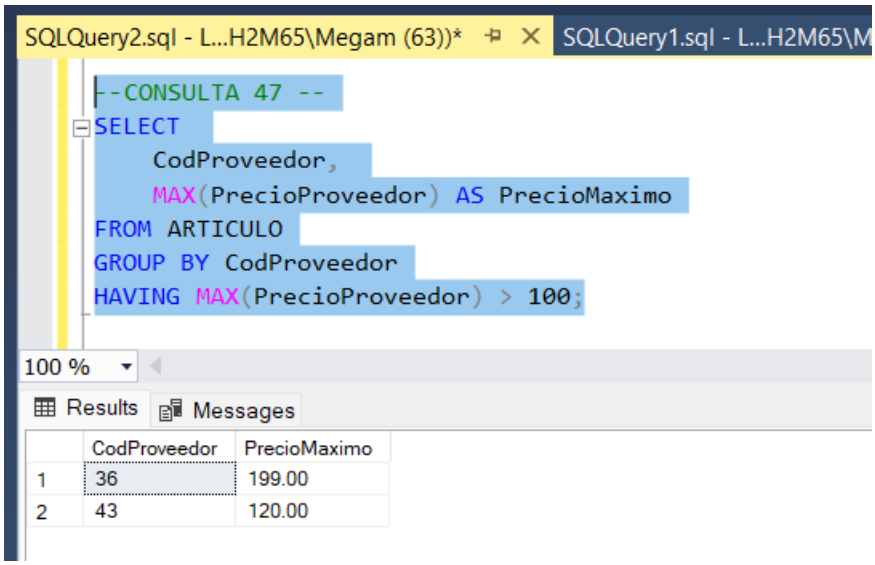
Consulta 45: Cuenta cuántas guías tiene cada transportista y muestra únicamente aquellos que han realizado 5 o más envíos.

46. **Mostrar líneas donde SUM(StockActual) < SUM(StockMinimo * NumArticulos PorLinea) ejemplo conceptual.**



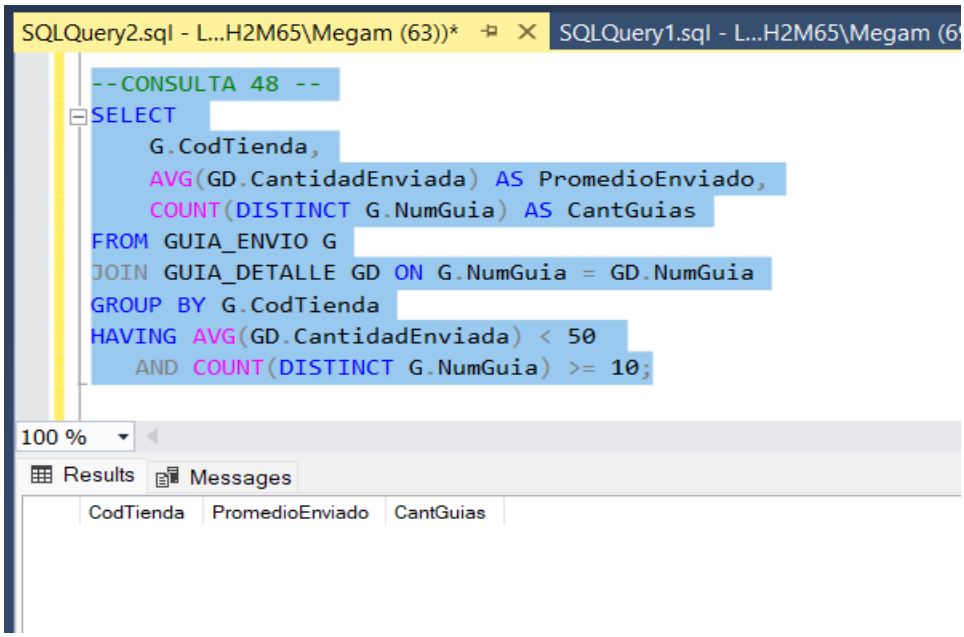
Consulta 46: Suma el stock actual y el stock mínimo por línea de productos y muestra solo las líneas donde el stock actual total es menor que el stock mínimo total, indicando riesgo de desabastecimiento.

47. **Enunciado: Mostrar proveedores donde MAX(PrecioProveedor) > 100.**



Consulta 47: Obtiene el precio máximo que maneja cada proveedor y muestra únicamente a los proveedores cuyo precio máximo supera 100.

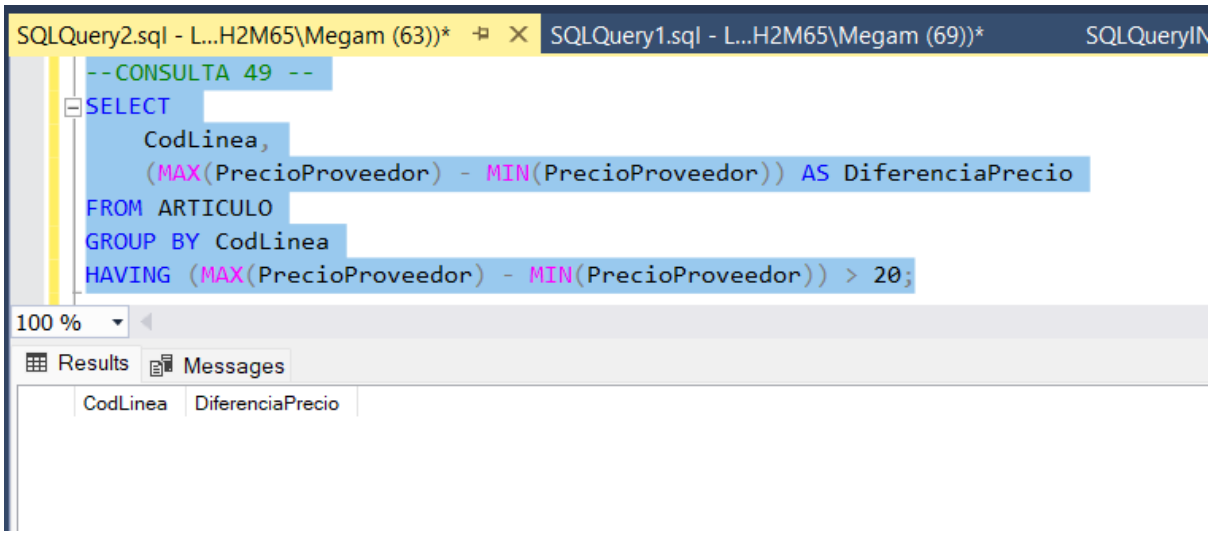
48. **Enunciado: Mostrar tiendas con AVG(CantidadEnviada) < 50 y COUNT(NumGuia) >= 10.**



Consulta 48

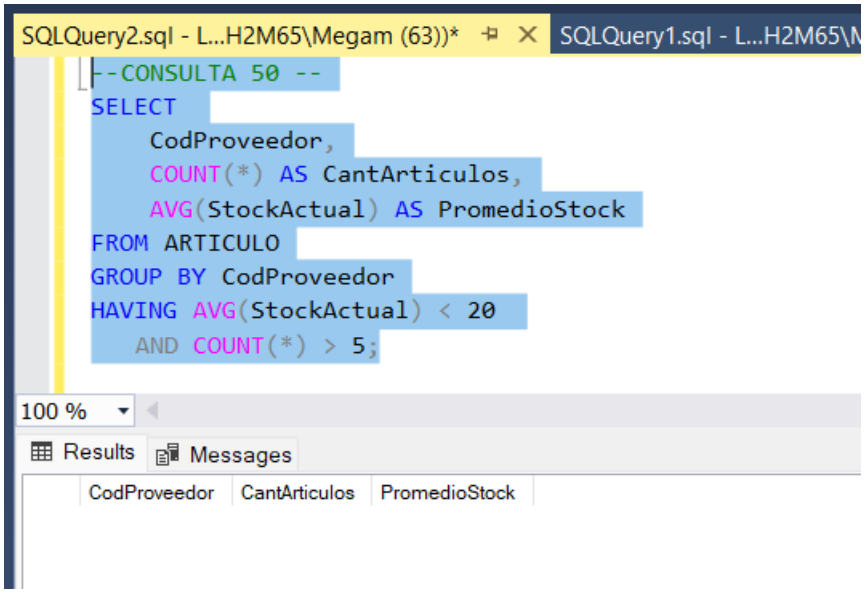
Calcula el promedio enviado y la cantidad de guías por tienda y muestra únicamente las tiendas que envían menos de 50 unidades en promedio pero que tienen al menos 10 guías registradas.

49. **Enunciado: Mostrar CodLinea donde (MAX(Precio)-MIN(Precio)) > 20.**



Consulta 49: Calcula la diferencia entre el precio máximo y mínimo dentro de cada línea y muestra solo las líneas donde esa diferencia supera 20, indicando alta variación de precios.

50. **Enunciado: Mostrar CodProveedor con COUNT() artículos donde AVG(StockActual) < 20 y COUNT() > 5.**



Consulta 50: Agrupa artículos por proveedor y muestra únicamente los proveedores que tienen más de 5 artículos y cuyo stock promedio es menor a 20 unidades, señalando bajo nivel de inventario.