“Año De La Recuperación Y
Consolidación De La Economía Peruana”

UNIVERSIDAD PERUANA LOS ANDES

“FACULTAD DE INGENIERÍA”

ESCUELA PROFESIONAL “SISTEMAS Y COMPUTACIÓN”

Ejercicios Propuestos Semana 9

CÁTEDRA: Base de Datos II

CATEDRÁTICO: Ing. Fernandez Bejarano Raul Enrique

ESTUDIANTE: Quispe Segama Franklin Noe

CICLO: V

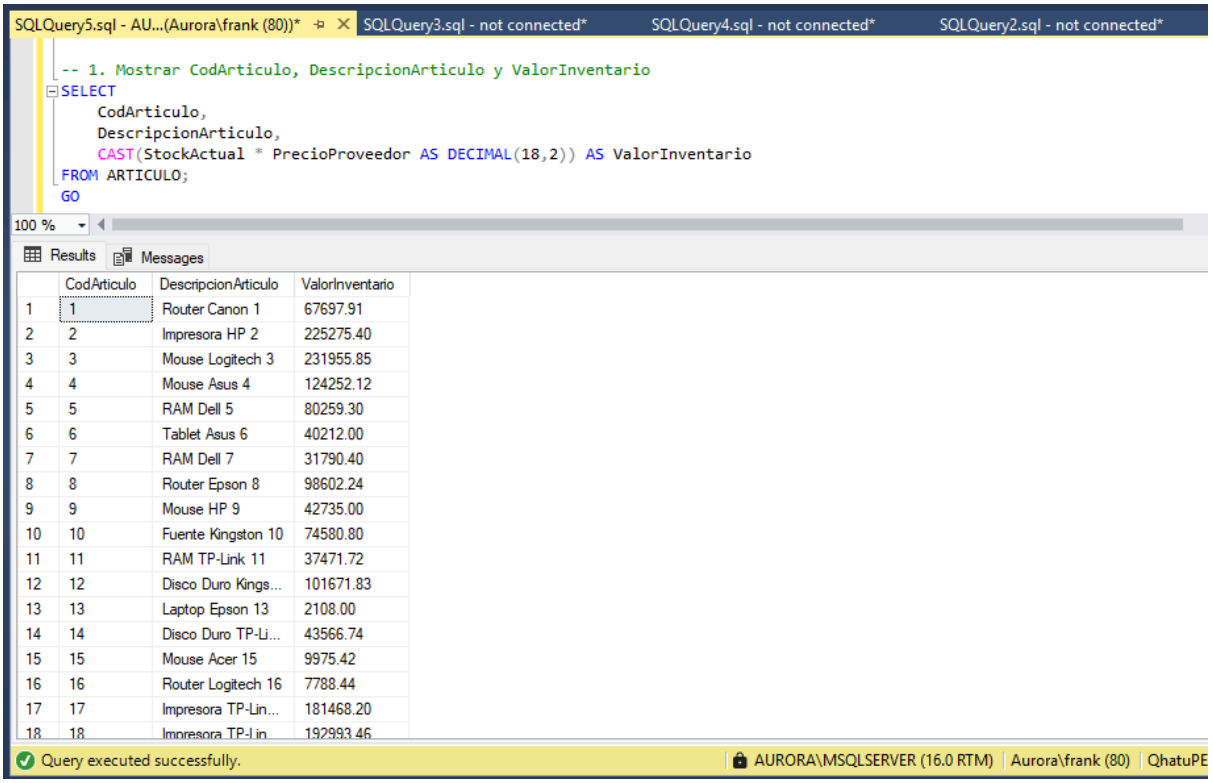
SECCIÓN: A1

HUANCAYO PERÚ

2025

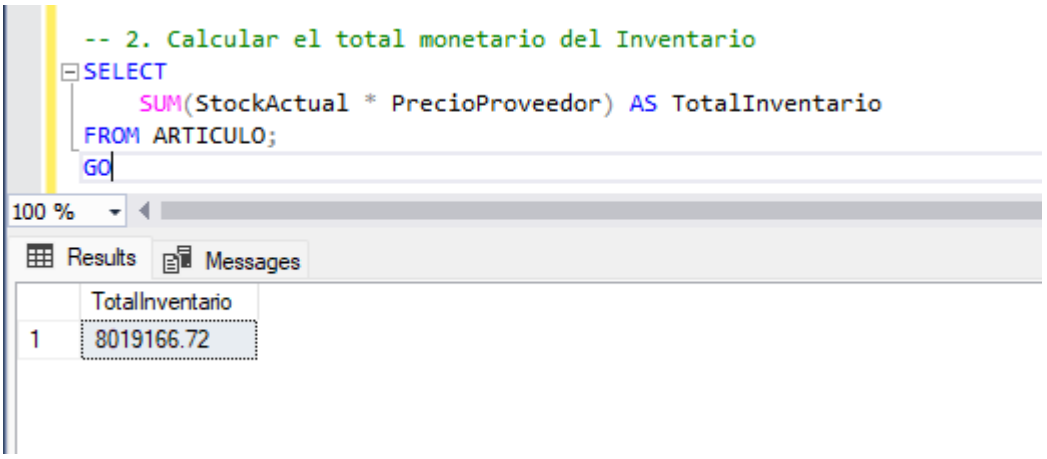
FUNCIONES DE AGREGACIÓN

1. Enunciado: Mostrar CodArticulo, DescripcionArticulo y ValorInventario.
Consulta SQL:



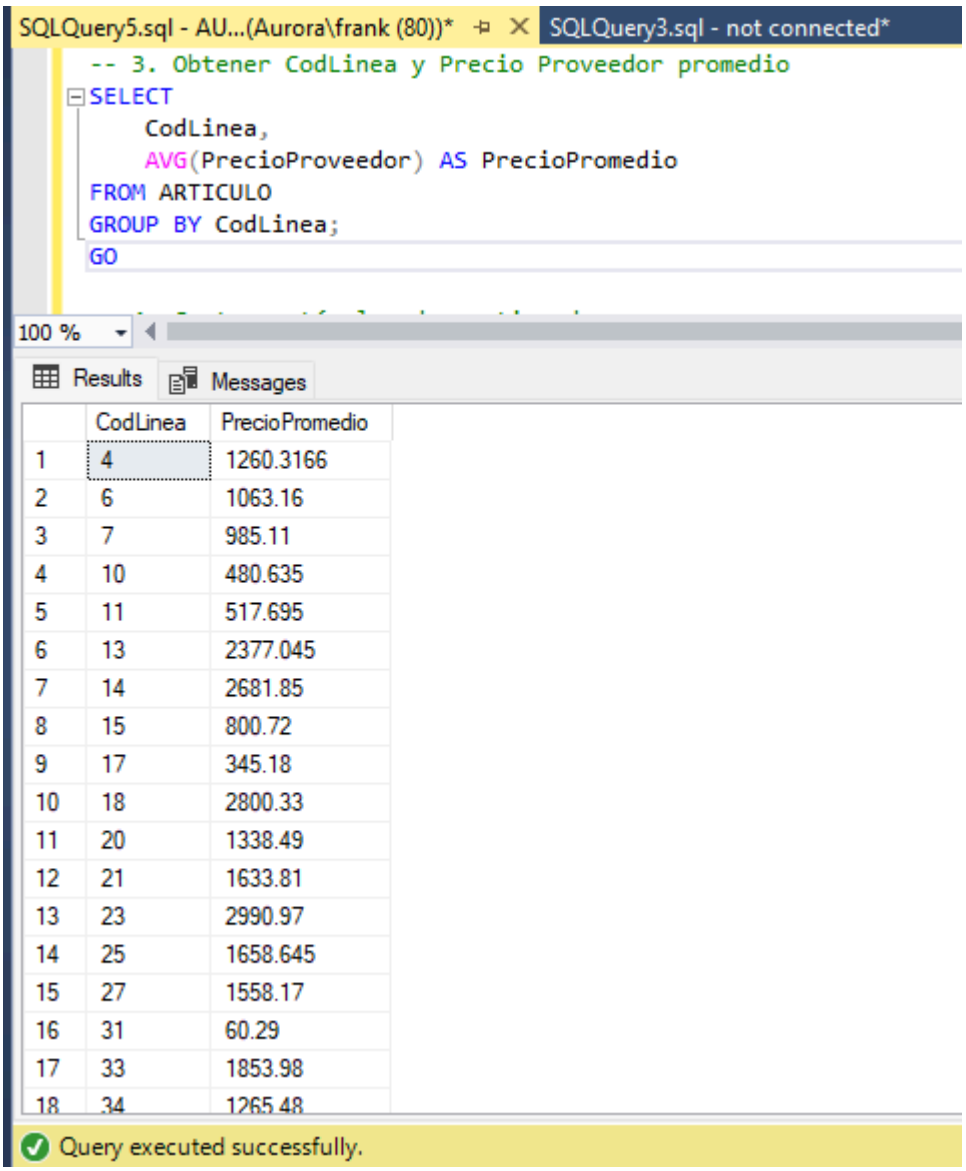
Explicación: Multiplica el stock actual por el precio del proveedor para calcular el valor monetario del inventario por artículo.

2. Enunciado: Calcular el total monetario del Inventario.
Consulta SQL:



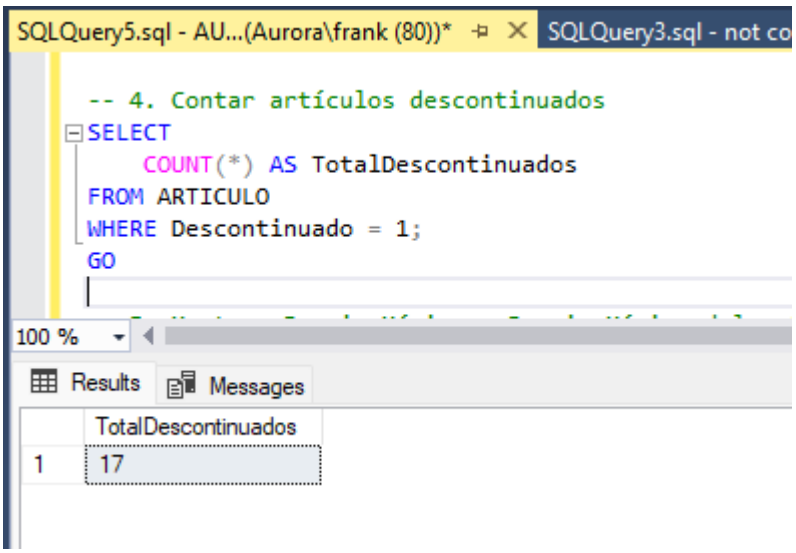
Explicación: Usa SUM para obtener el valor total del inventario acumulando stock * precio.

3. Enunciado: Obtener CodLinea y Precio Proveedor promedio.
Consulta SQL:



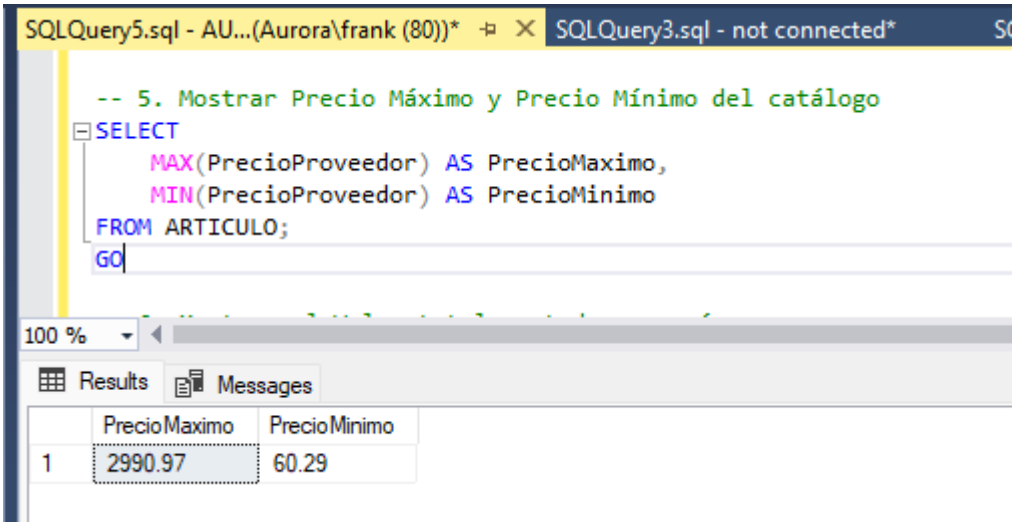
Explicación: Calcula el precio promedio de los artículos agrupados por línea.

4. Enunciado: Contar artículos descontinuados.
Consulta SQL:



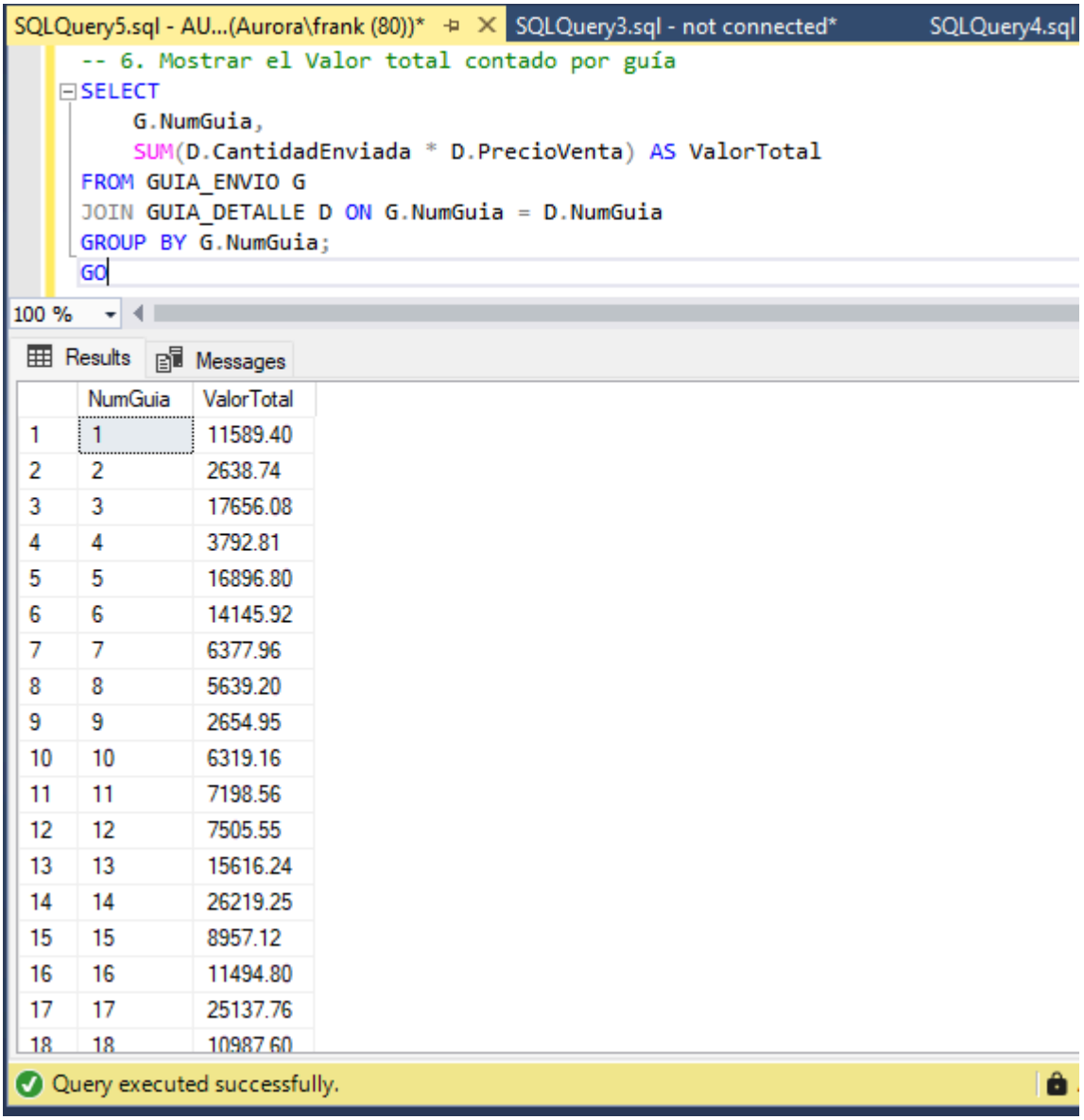
Explicación: Cuenta todos los registros marcados como descontinuados.

5. Enunciado: Mostrar Precio Máximo y Precio Mínimo del catálogo.
Consulta SQL:



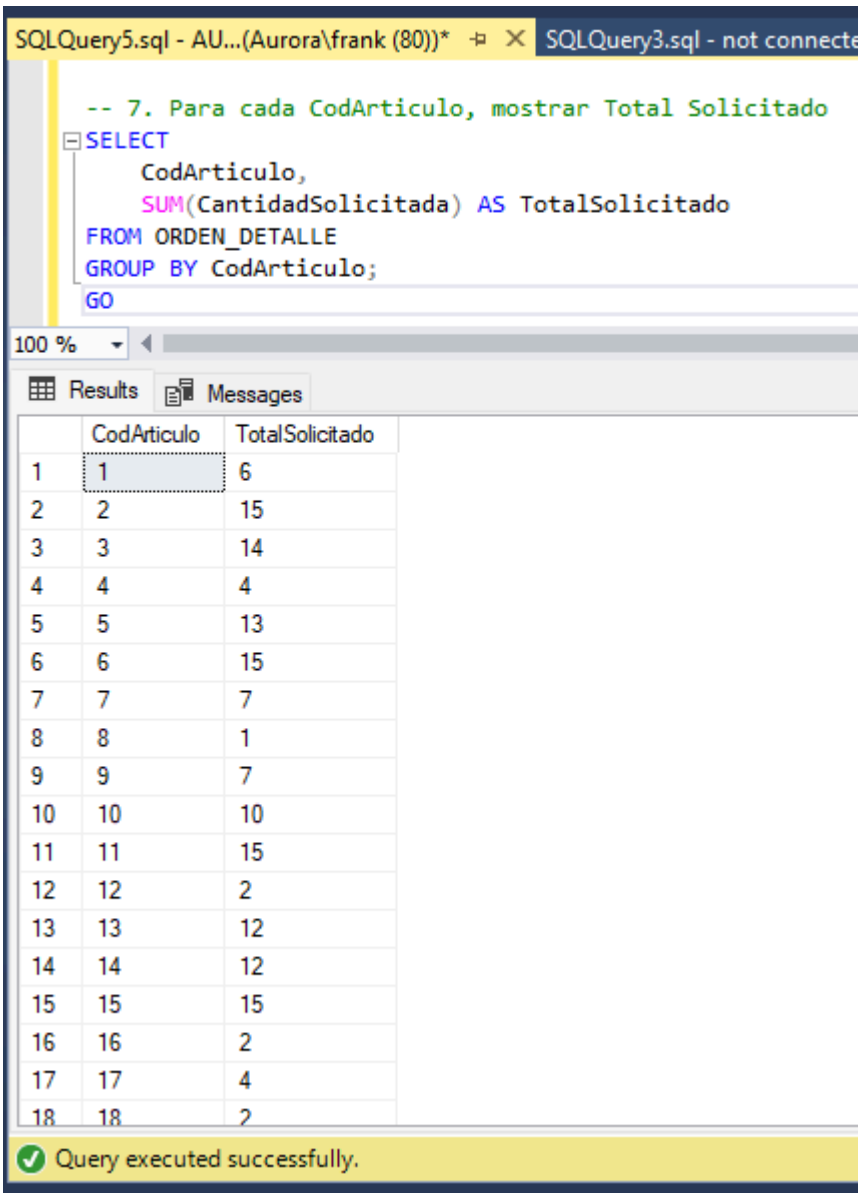
Explicación: Usa funciones MAX y MIN para obtener el precio más alto y más bajo del catálogo.

6. Enunciado: Mostrar el Valor total contado por guía.
Consulta SQL:



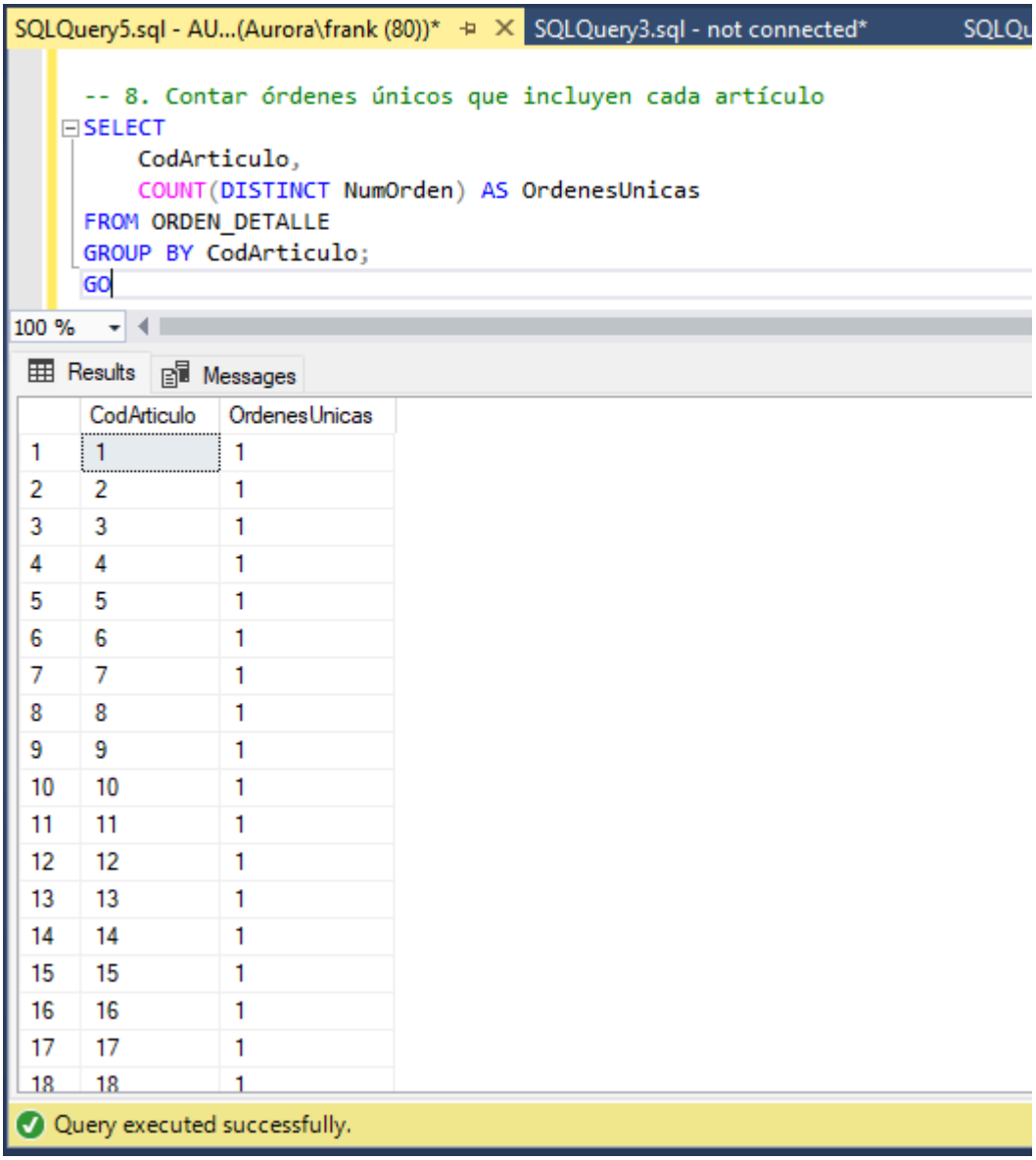
Explicación: Calcula el valor total por guía multiplicando cantidad por precio.

7. Enunciado: Para cada CodArticulo, mostrar Total Solicitado.
Consulta SQL:



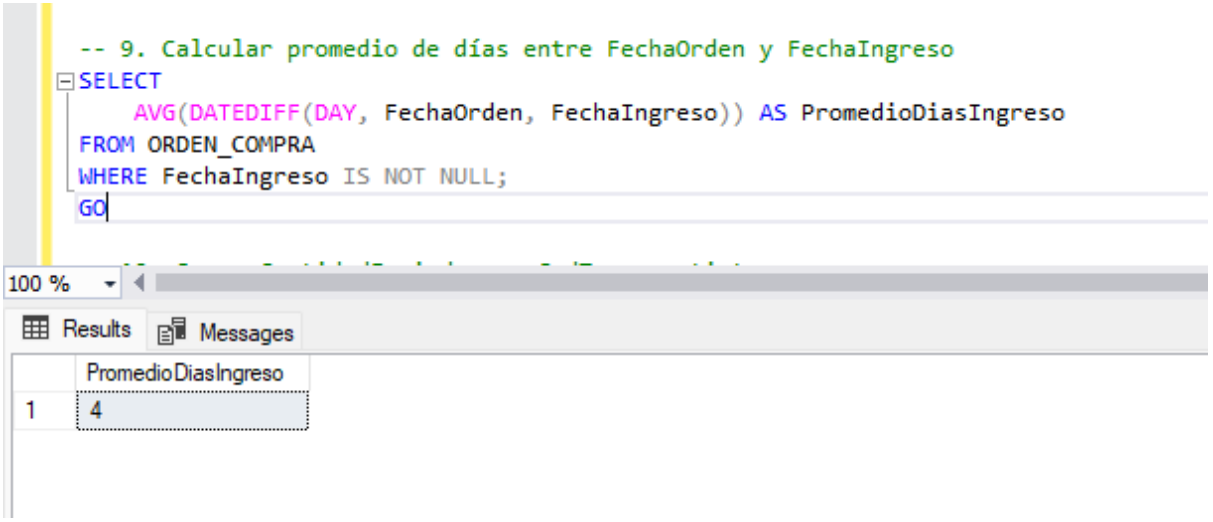
Explicación: Suma todas las cantidades solicitadas agrupadas por artículo.

8. Enunciado: Contar órdenes únicos que incluyen cada artículo.
Consulta SQL:



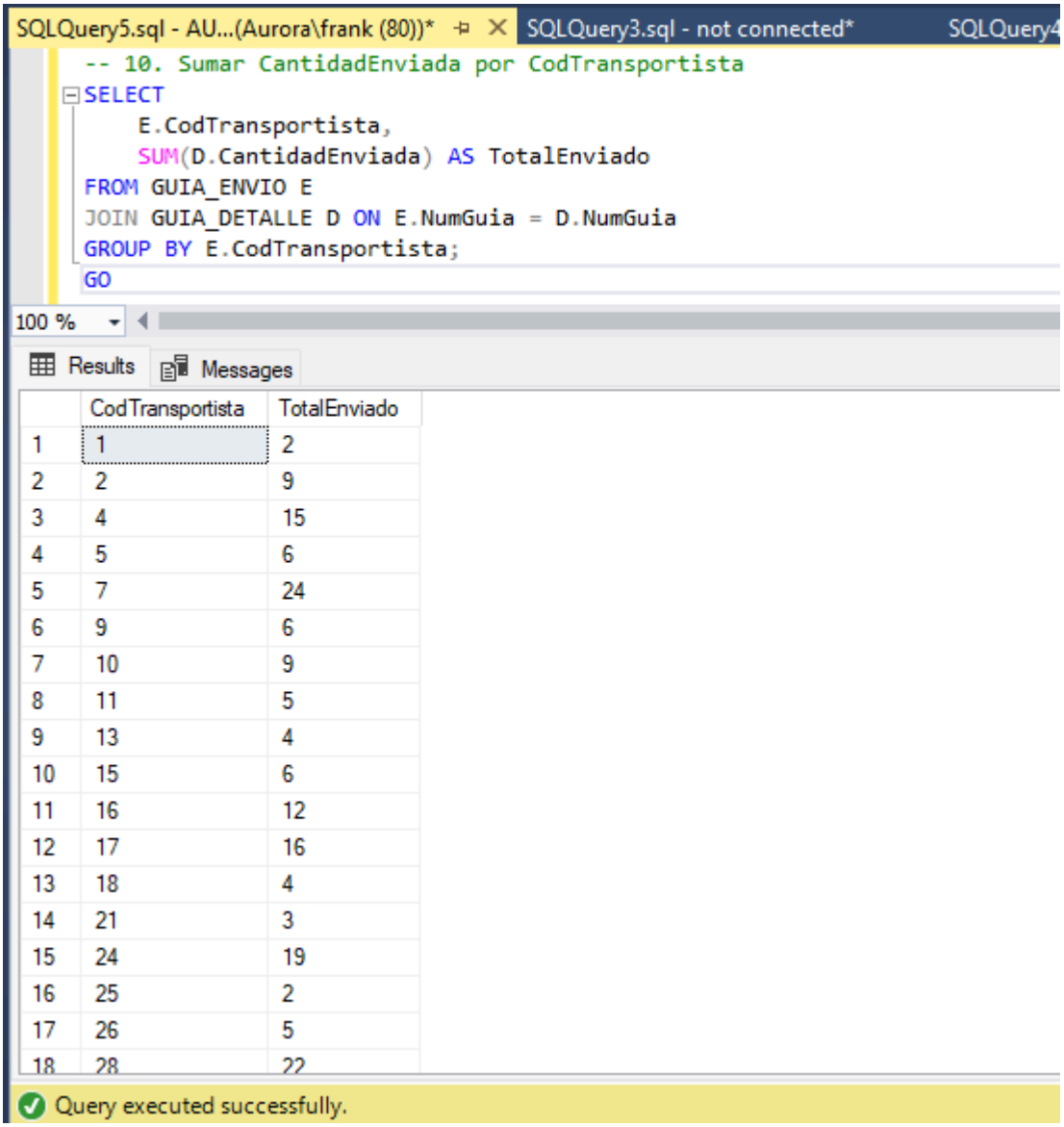
Explicación: Cuenta cuántas órdenes distintas incluyen un artículo.

9. Enunciado: Calcular promedio de días entre FechaOrden y FechaIngreso.
Consulta SQL:



Explicación: Calcula el tiempo promedio de ingreso de órdenes en días.

10. Enunciado: Sumar CantidadEnviada por CodTransportista.
Consulta SQL:



Explicación: Obtiene el total de productos enviados por transportista.

CLÁUSULA GROUP BY

11. Enunciado: Mostrar NomLinea y CantArticulos.
Consulta SQL:

SQLQuery5.sql - AU...(Aurora\frank (80))* | SQLQuery3.sql - not connec

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

100 %

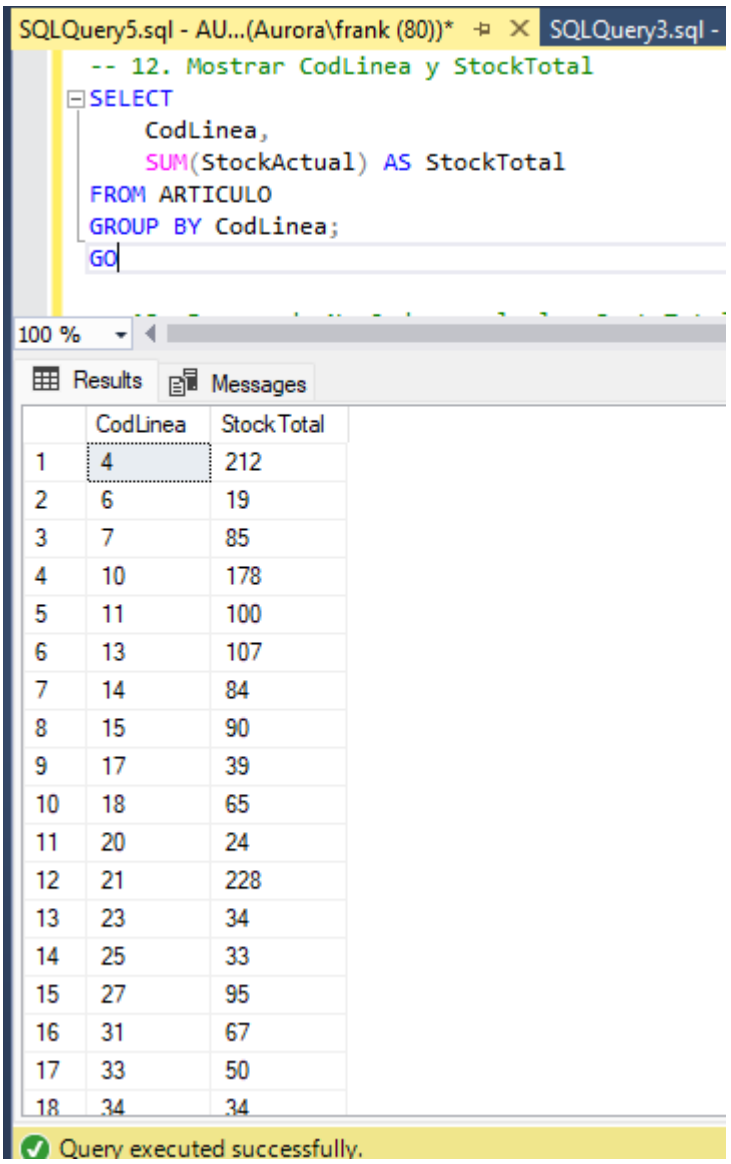
Results | Messages

	NomLinea	CantArticulos
1	Mouses	3
2	Memorias	1
3	Procesadores	1
4	Cables	2
5	Auriculares	2
6	Accesorios	2
7	Redes	1
8	Servidores	1
9	UPS	1
10	Proyectores	1
11	Sillas Gamer	1
12	Parlantes	3
13	Smartwatch	1
14	Cases	2
15	Fuentes de Poder	1
16	Lectoras	1
17	Controles	2
18	Pantallas Táctiles	1

Explicación: Agrupa por línea y cuenta cuántos artículos tiene cada una.

12. Enunciado: Mostrar CodLinea y StockTotal.

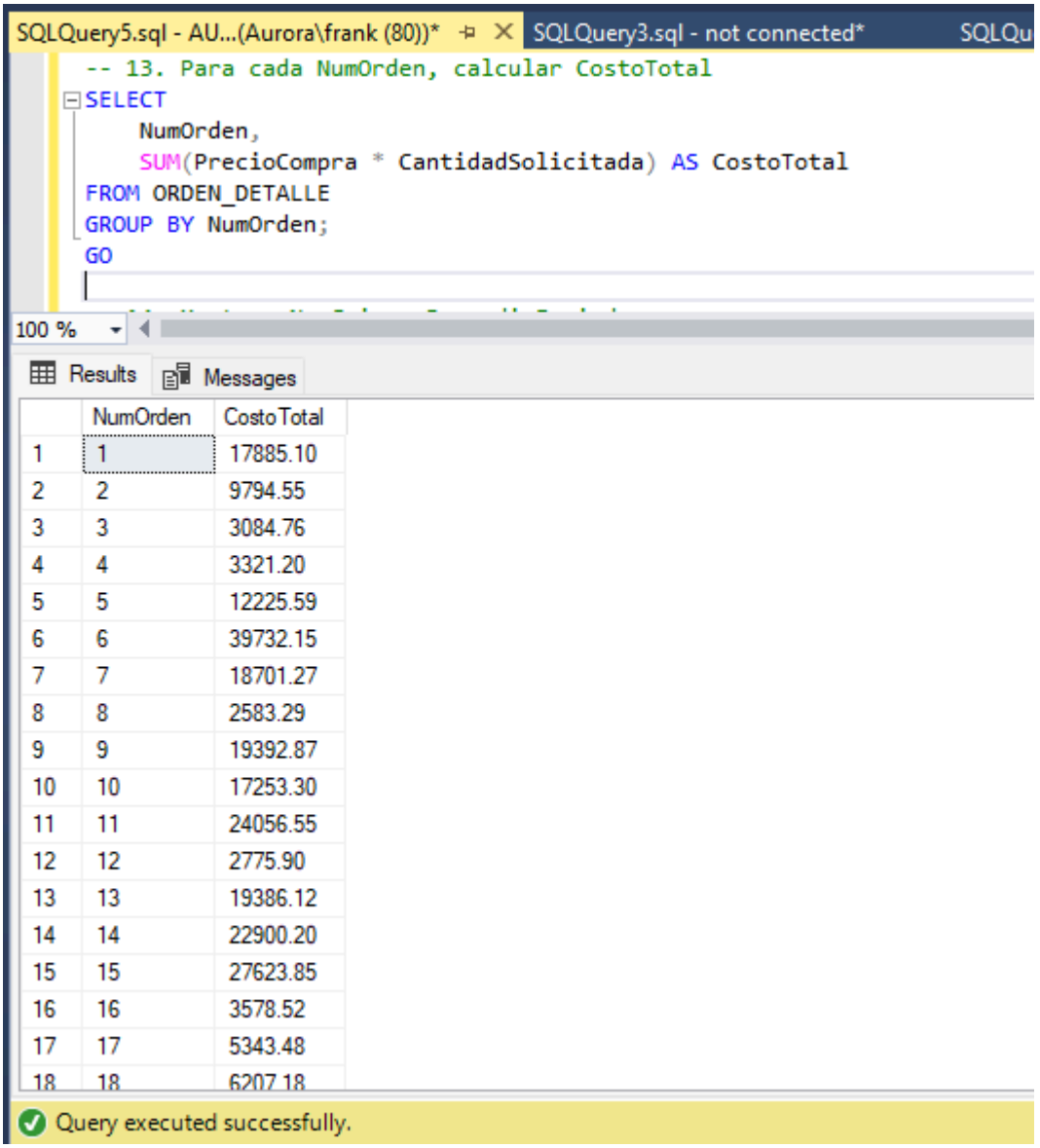
Consulta SQL:



Explicación: Suma el stock total agrupado por línea.

13. Enunciado: Para cada NumOrden, calcular CostoTotal.

Consulta SQL:



Explicación: Calcula el costo total de cada orden.

14. Enunciado: Mostrar NumGuia y PromedioEnviado.

Consulta SQL:

```sql
-- 14. Mostrar NumGuia y PromedioEnviado
SELECT
    NumGuia,
    AVG(CantidadEnviada) AS PromedioEnviado
FROM GUIA_DETALLE
GROUP BY NumGuia;
GO
```

100 %

Results | Messages

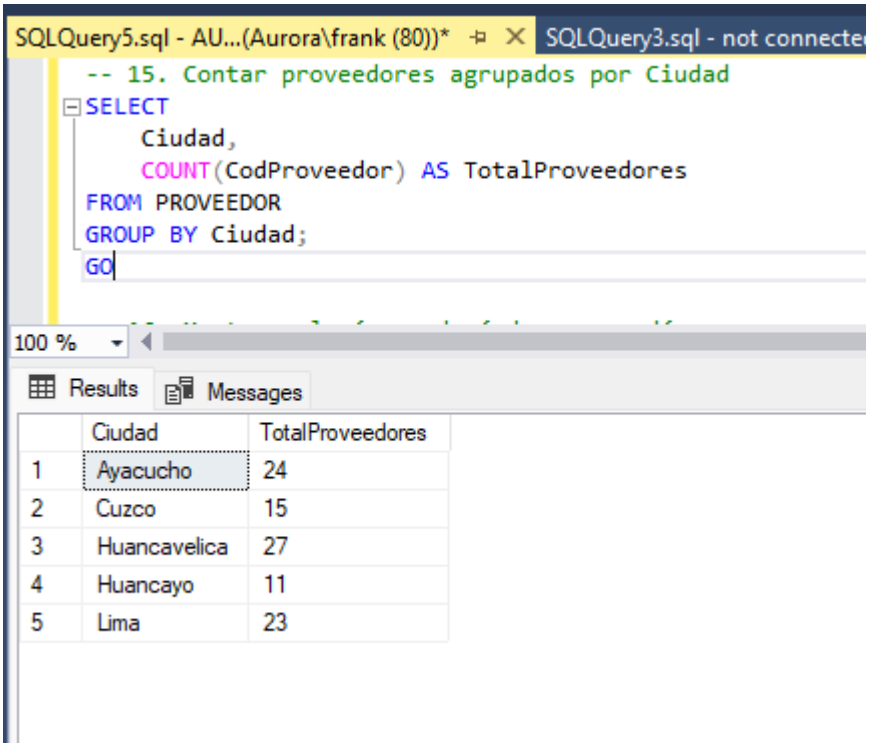
	NumGuia	PromedioEnviado
1	1	4
2	2	3
3	3	6
4	4	7
5	5	5
6	6	8
7	7	2
8	8	4
9	9	5
10	10	4
11	11	6
12	12	5
13	13	8
14	14	9
15	15	4
16	16	9
17	17	8
18	18	10

Query executed successfully.

Explicación: Calcula el promedio de unidades enviadas por guía.

15. Enunciado: Contar proveedores agrupados por Ciudad.

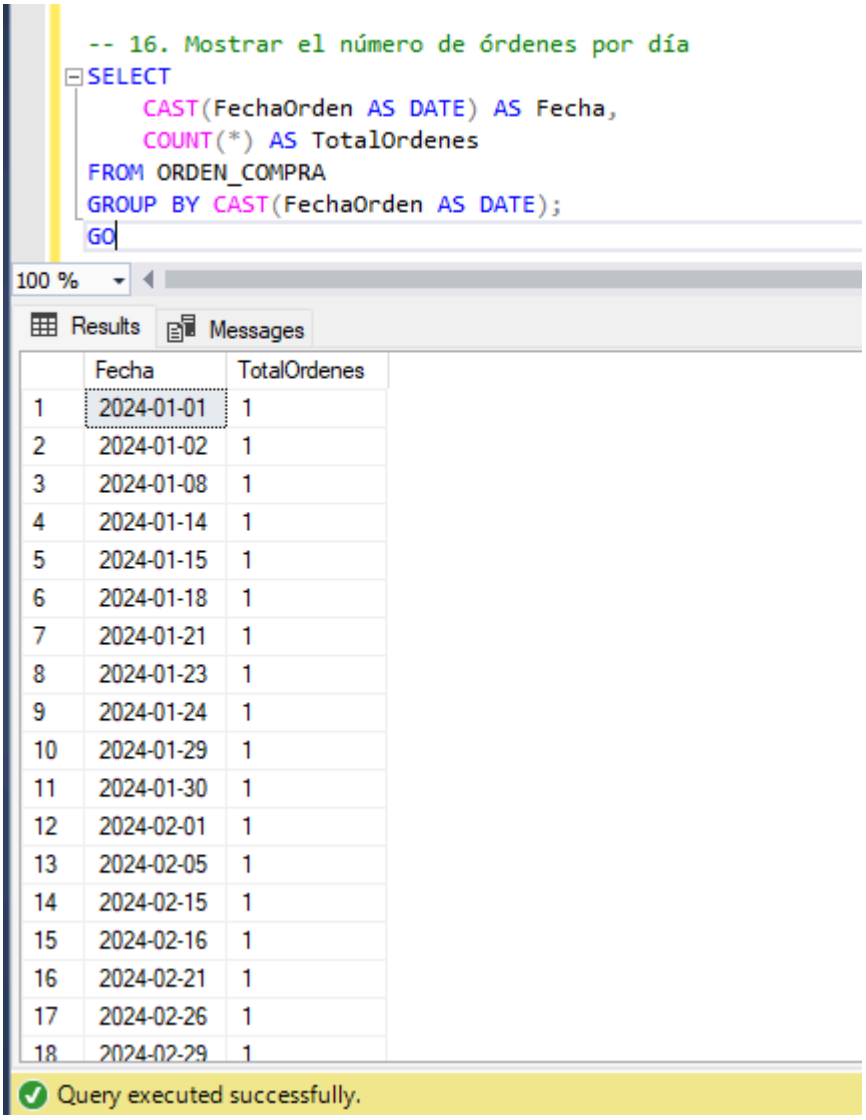
Consulta SQL:



Explicación: Cuenta proveedores según su ciudad.

16. Enunciado: Mostrar el número de órdenes por día.

Consulta SQL:



Explicación: Agrupa las órdenes por fecha sin considerar la hora.

17. Enunciado: Sumar (CantidadEnviada * PrecioVenta) por CodTienda.

Consulta SQL:

```sql
-- 17. Sumar (CantidadEnviada * PrecioVenta) por CodTienda
SELECT
    G.CodTienda,
    SUM(D.CantidadEnviada * D.PrecioVenta) AS TotalVentas
FROM GUIA_ENVIO G
JOIN GUIA_DETALLE D ON G.NumGuia = D.NumGuia
GROUP BY G.CodTienda;
GO
```

100 %

Results | Messages

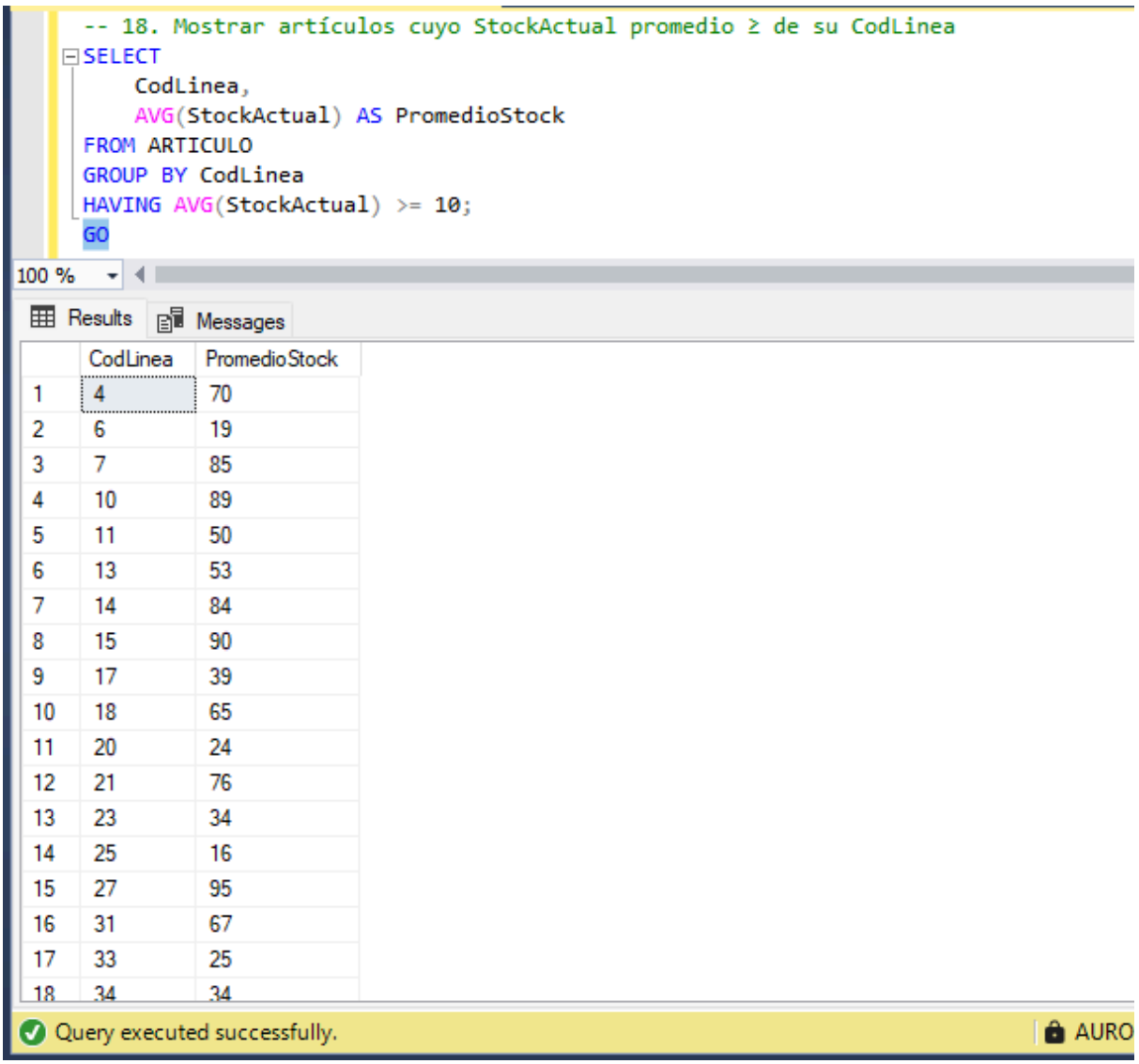
	CodTienda	TotalVentas
1	1	7799.11
2	3	45371.76
3	5	30198.18
4	9	6039.42
5	12	11830.92
6	14	28951.48
7	15	2287.70
8	16	318.04
9	17	17247.72
10	18	11494.80
11	19	10309.32
12	20	27843.77
13	22	25456.32
14	23	7096.67
15	25	22487.42
16	26	30229.35
17	27	20111.77
18	28	31754.28

Query executed successfully.

Explicación: Calcula el total vendido por tienda.

18. Enunciado: Mostrar artículos cuyo StockActual promedio ≥ 10.

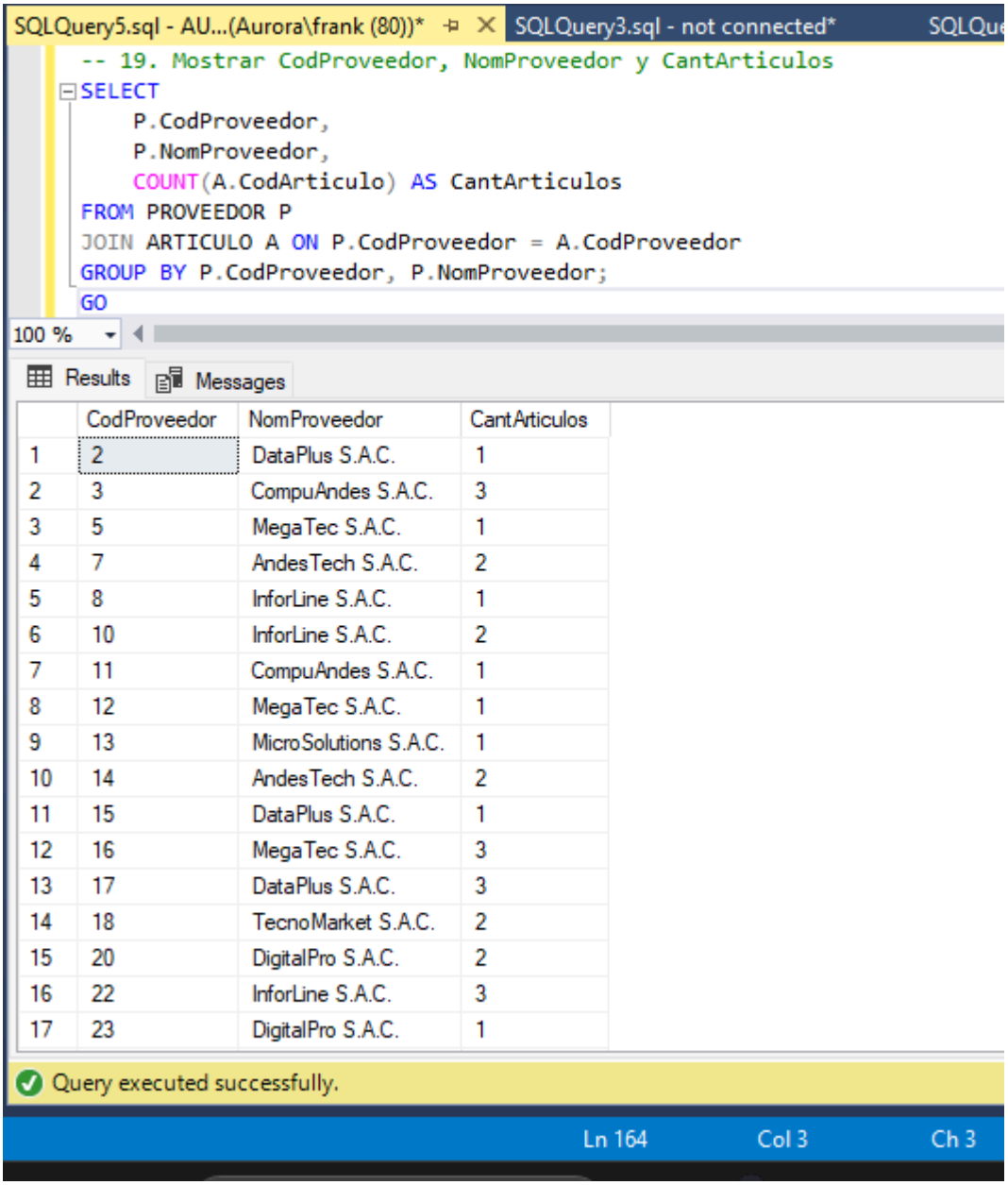
Consulta SQL:



Explicación: Filtra líneas cuyo stock promedio es mayor o igual a 10.

19. Enunciado: Mostrar CodProveedor, NomProveedor y CantArticulos.

Consulta SQL:



Explicación: Muestra cuántos artículos tiene cada proveedor.

20. Enunciado: Mostrar para cada Estado la suma de CantidadSolicitada.

Consulta SQL:

```sql
-- 20. Mostrar para cada Estado la suma de CantidadSolicitada
SELECT
    Estado,
    SUM(CantidadSolicitada) AS TotalSolicitado
FROM ORDEN_DETALLE
GROUP BY Estado;
GO
```

100 %

Results | Messages

	Estado	TotalSolicitado
1	Recibido	830

Explicación: Muestra el total solicitado según estado.

CLÁUSULA OVER

21. Enunciado: Asignar posición por línea ordenada por precio.

Consulta SQL:

```sql
-- 21. Asignar posición por línea ordenada por precio
SELECT
    CodLinea,
    CodArticulo,
    PrecioProveedor,
    ROW_NUMBER() OVER (PARTITION BY CodLinea ORDER BY PrecioProveedor DESC) AS Posicion
FROM ARTICULO;
GO
```

100 %

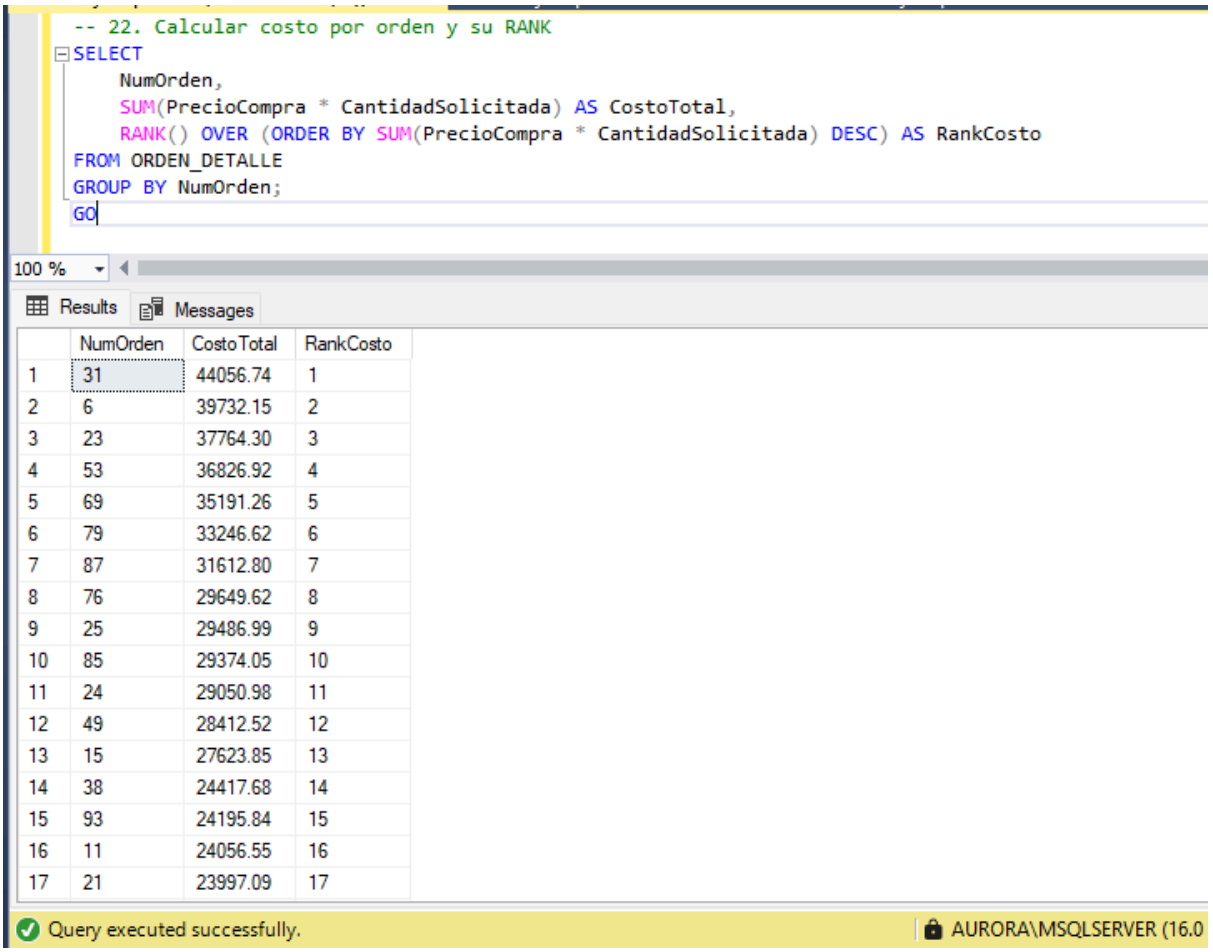
Results | Messages

	CodLinea	CodArticulo	PrecioProveedor	Posicion
1	4	63	1696.05	1
2	4	12	1517.49	2
3	4	53	567.41	3
4	6	19	1063.16	1
5	7	98	985.11	1
6	10	87	632.24	1
7	10	81	329.03	2
8	11	55	777.42	1
9	11	78	257.97	2
10	13	43	2610.54	1
11	13	69	2143.55	2
12	14	2	2681.85	1
13	15	45	800.72	1
14	17	84	345.18	1
15	18	74	2800.33	1
16	20	44	1338.49	1
17	21	22	2309.26	1

Explicación: Numera los artículos dentro de cada línea, ordenados por precio de mayor a menor.

22. Enunciado: Calcular costo por orden y su RANK.

Consulta SQL:



```sql
-- 22. Calcular costo por orden y su RANK
SELECT
    NumOrden,
    SUM(PrecioCompra * CantidadSolicitada) AS CostoTotal,
    RANK() OVER (ORDER BY SUM(PrecioCompra * CantidadSolicitada) DESC) AS RankCosto
FROM ORDEN_DETALLE
GROUP BY NumOrden;
GO
```

100 %

Results | Messages

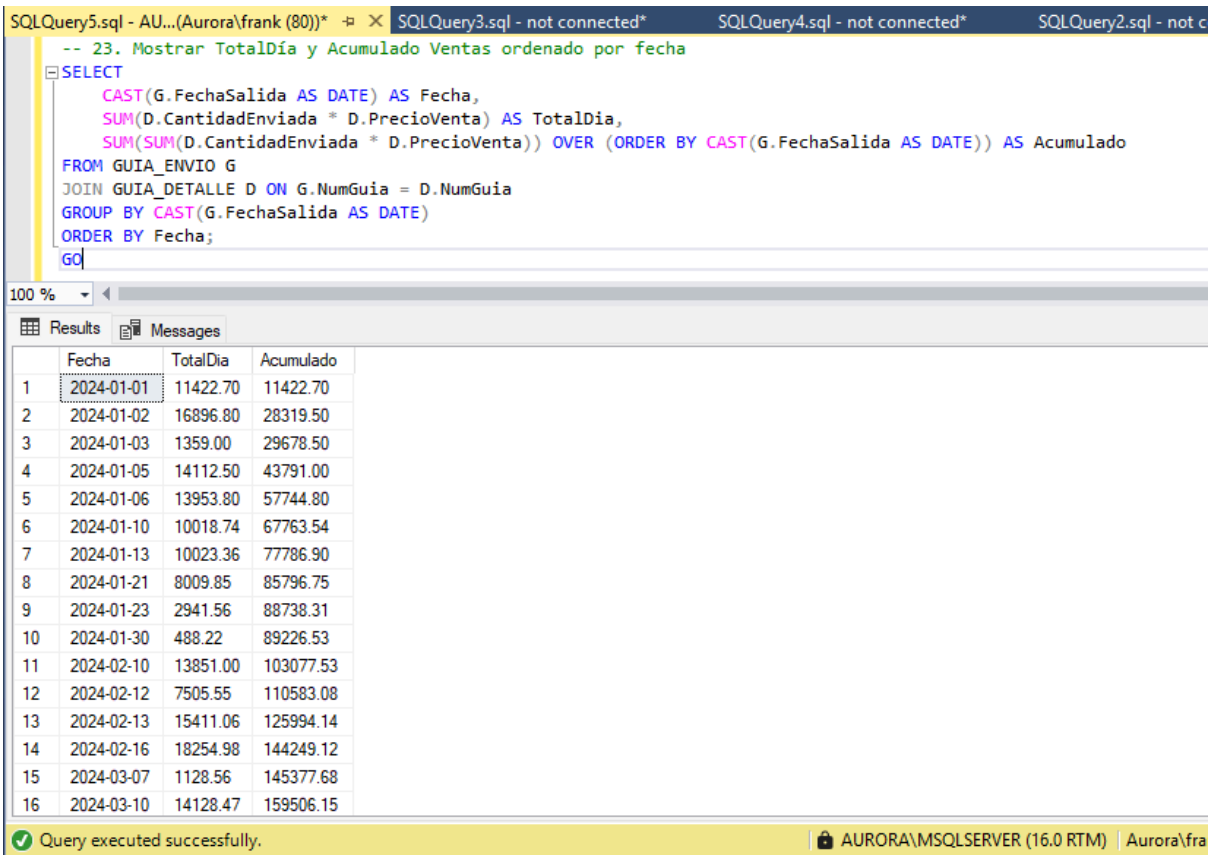
	NumOrden	CostoTotal	RankCosto
1	31	44056.74	1
2	6	39732.15	2
3	23	37764.30	3
4	53	36826.92	4
5	69	35191.26	5
6	79	33246.62	6
7	87	31612.80	7
8	76	29649.62	8
9	25	29486.99	9
10	85	29374.05	10
11	24	29050.98	11
12	49	28412.52	12
13	15	27623.85	13
14	38	24417.68	14
15	93	24195.84	15
16	11	24056.55	16
17	21	23997.09	17

Query executed successfully. AURORA\MSQLSERVER (16.0

Explicación: Calcula el costo total por orden y asigna un rango según su valor.

23. Enunciado: Mostrar TotalDía y Acumulado Ventas ordenado por fecha.

Consulta SQL:



```sql
-- 23. Mostrar TotalDía y Acumulado Ventas ordenado por fecha
SELECT
    CAST(G.FechaSalida AS DATE) AS Fecha,
    SUM(D.CantidadEnviada * D.PrecioVenta) AS TotalDia,
    SUM(SUM(D.CantidadEnviada * D.PrecioVenta)) OVER (ORDER BY CAST(G.FechaSalida AS DATE)) AS Acumulado
FROM GUIA_ENVIO G
JOIN GUIA_DETALLE D ON G.NumGuia = D.NumGuia
GROUP BY CAST(G.FechaSalida AS DATE)
ORDER BY Fecha;
GO
```

Results | Messages

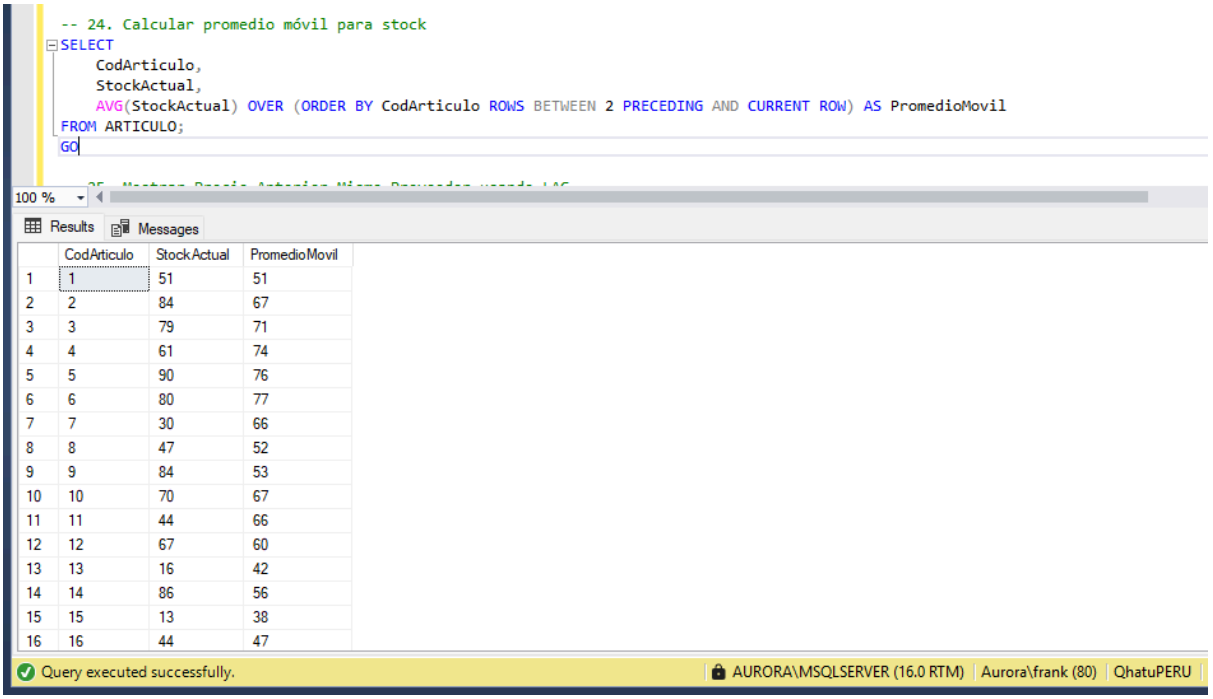
	Fecha	TotalDia	Acumulado
1	2024-01-01	11422.70	11422.70
2	2024-01-02	16896.80	28319.50
3	2024-01-03	1359.00	29678.50
4	2024-01-05	14112.50	43791.00
5	2024-01-06	13953.80	57744.80
6	2024-01-10	10018.74	67763.54
7	2024-01-13	10023.36	77786.90
8	2024-01-21	8009.85	85796.75
9	2024-01-23	2941.56	88738.31
10	2024-01-30	488.22	89226.53
11	2024-02-10	13851.00	103077.53
12	2024-02-12	7505.55	110583.08
13	2024-02-13	15411.06	125994.14
14	2024-02-16	18254.98	144249.12
15	2024-03-07	1128.56	145377.68
16	2024-03-10	14128.47	159506.15

Query executed successfully. AURORA\MSQLSERVER (16.0 RTM) Aurora\fra

Explicación: Calcula las ventas diarias y su acumulado a lo largo del tiempo.

24. Enunciado: Calcular promedio móvil para stock.

Consulta SQL:



```sql
-- 24. Calcular promedio móvil para stock
SELECT
    CodArticulo,
    StockActual,
    AVG(StockActual) OVER (ORDER BY CodArticulo ROWS BETWEEN 2 PRECEDING AND CURRENT ROW) AS PromedioMovil
FROM ARTICULO;
GO
```

Results | Messages

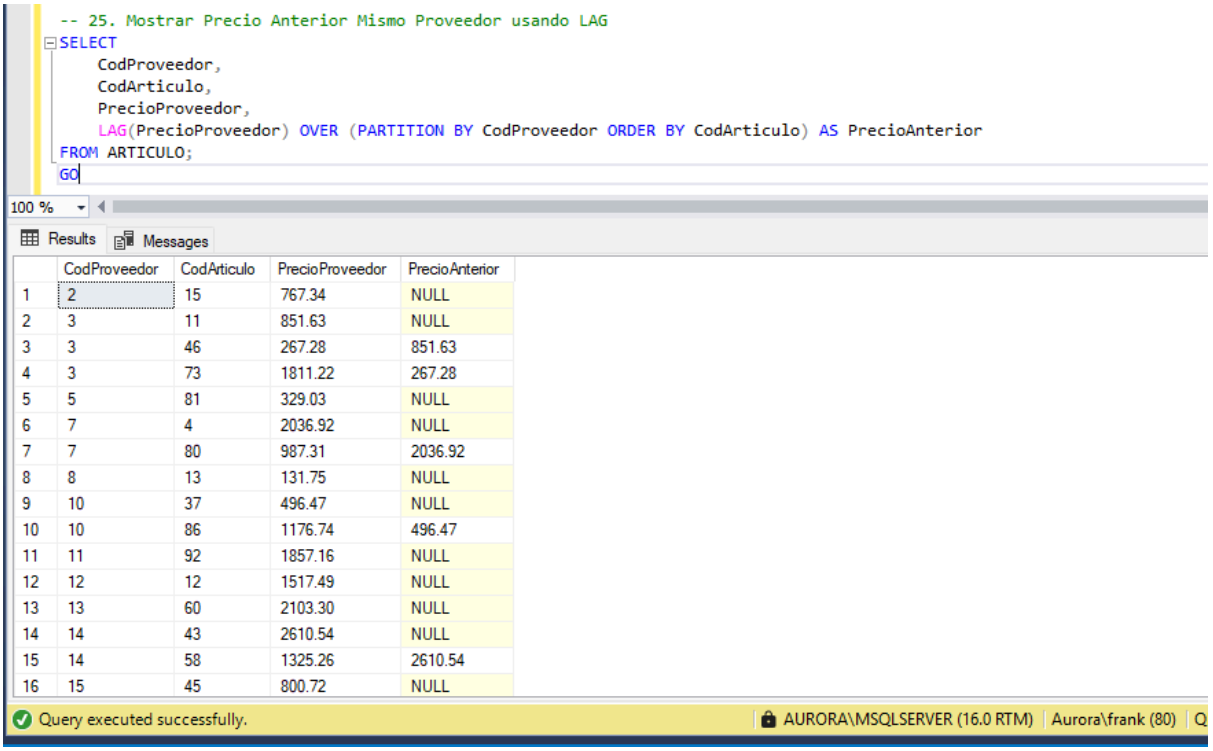
	CodArticulo	StockActual	PromedioMovil
1	1	51	51
2	2	84	67
3	3	79	71
4	4	61	74
5	5	90	76
6	6	80	77
7	7	30	66
8	8	47	52
9	9	84	53
10	10	70	67
11	11	44	66
12	12	67	60
13	13	16	42
14	14	86	56
15	15	13	38
16	16	44	47

Query executed successfully. AURORA\MSQLSERVER (16.0 RTM) Aurora\frank (80) QhatuPERU

Explicación: Calcula un promedio móvil considerando el artículo actual y los dos anteriores.

25. Enunciado: Mostrar Precio Anterior Mismo Proveedor usando LAG.

Consulta SQL:



```sql
-- 25. Mostrar Precio Anterior Mismo Proveedor usando LAG
SELECT
    CodProveedor,
    CodArticulo,
    PrecioProveedor,
    LAG(PrecioProveedor) OVER (PARTITION BY CodProveedor ORDER BY CodArticulo) AS PrecioAnterior
FROM ARTICULO;
GO
```

100 %

Results | Messages

	CodProveedor	CodArticulo	PrecioProveedor	PrecioAnterior
1	2	15	767.34	NULL
2	3	11	851.63	NULL
3	3	46	267.28	851.63
4	3	73	1811.22	267.28
5	5	81	329.03	NULL
6	7	4	2036.92	NULL
7	7	80	987.31	2036.92
8	8	13	131.75	NULL
9	10	37	496.47	NULL
10	10	86	1176.74	496.47
11	11	92	1857.16	NULL
12	12	12	1517.49	NULL
13	13	60	2103.30	NULL
14	14	43	2610.54	NULL
15	14	58	1325.26	2610.54
16	15	45	800.72	NULL

Query executed successfully. | AURORA\MSQLSERVER (16.0 RTM) | Aurora\frank (80) | Q

Explicación: Usa LAG para comparar el precio actual con el anterior del mismo proveedor.

26. Enunciado: Añadir columna Porcentaje de Línea a cada artículo.

Consulta SQL:

```sql
-- 26. Añadir columna Cantidad Porc.Linea a cada artículo
SELECT
    CodLinea,
    CodArticulo,
    StockActual,
    CAST(StockActual * 100.0 / SUM(StockActual) OVER (PARTITION BY CodLinea) AS DECIMAL(5,2)) AS PorcentajeLinea
FROM ARTICULO;
GO
```

100 %

Results | Messages

	CodLinea	CodArticulo	StockActual	PorcentajeLinea
1	4	12	67	31.60
2	4	53	59	27.83
3	4	63	86	40.57
4	6	19	19	100.00
5	7	98	85	100.00
6	10	81	92	51.69
7	10	87	86	48.31
8	11	78	56	56.00
9	11	55	44	44.00
10	13	43	87	81.31
11	13	69	20	18.69
12	14	2	84	100.00
13	15	45	90	100.00
14	17	84	39	100.00
15	18	74	65	100.00
16	20	44	24	100.00

Query executed successfully. AURORA\MSQLSERVER (16.0 RTM) | Aurora\frank (80) | QhatuP

Explicación: Calcula el porcentaje que representa el stock del artículo dentro de su línea.

27. Enunciado: Mostrar Monto Proveedor y Porcentaje del Total.

Consulta SQL:

```sql
-- 27. Mostrar Monto Proveedor y Porcentaje Del Total
SELECT
    CodProveedor,
    SUM(PrecioProveedor * StockActual) AS MontoProveedor,
    CAST(SUM(PrecioProveedor * StockActual) * 100.0 / SUM(SUM(PrecioProveedor * StockActual)) OVER() AS DECIMAL(5,2)) AS PorcentajeTotal
FROM ARTICULO
GROUP BY CodProveedor;
GO
```

100 %

Results | Messages

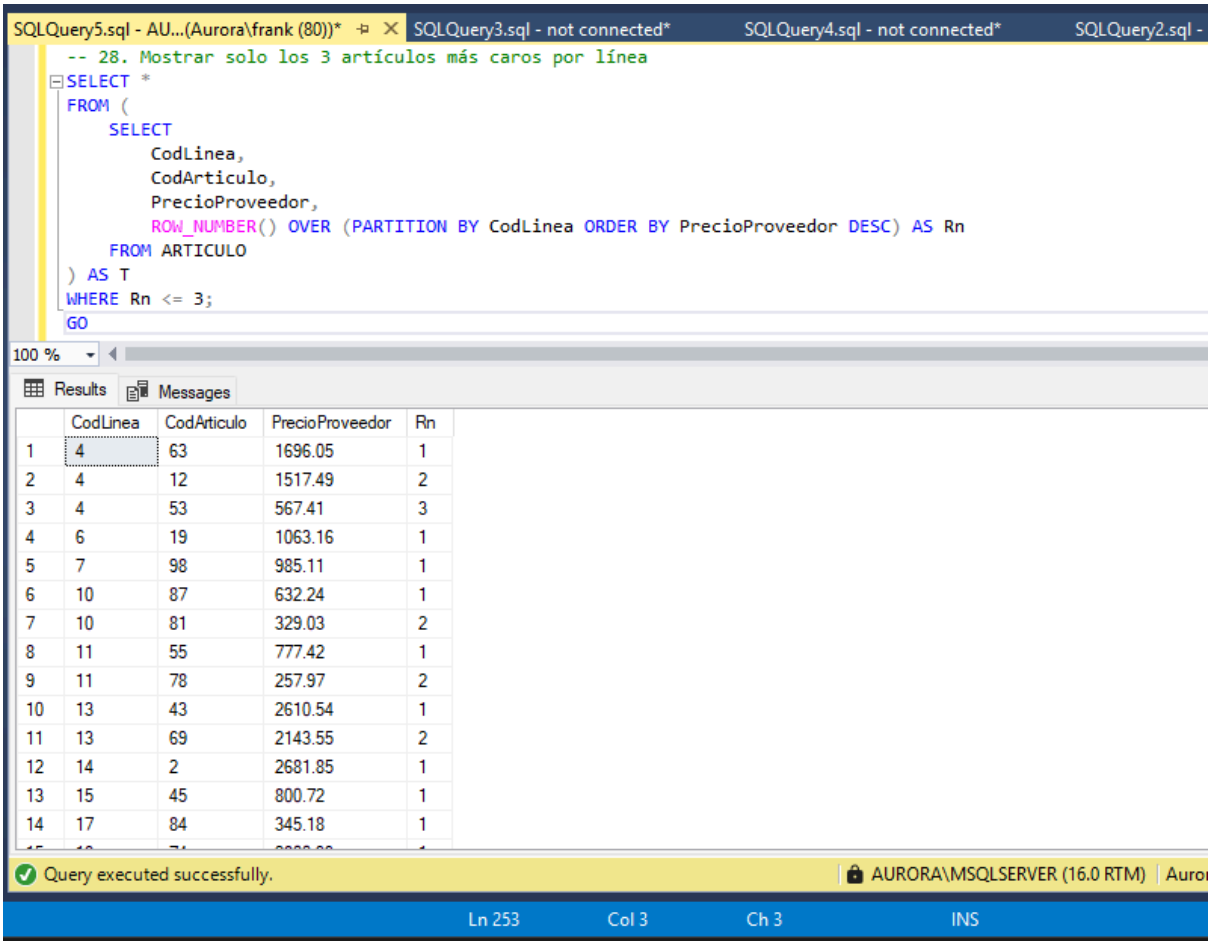
	CodProveedor	MontoProveedor	PorcentajeTotal
1	2	9975.42	0.12
2	3	232375.90	2.90
3	5	30270.76	0.38
4	7	146960.25	1.83
5	8	2108.00	0.03
6	10	65863.62	0.82
7	11	79857.88	1.00
8	12	101671.83	1.27
9	13	151437.60	1.89
10	14	258923.22	3.23
11	15	72064.80	0.90
12	16	290344.60	3.62
13	17	95822.42	1.19
14	18	76941.48	0.96
15	20	205575.68	2.56
16	22	227058.15	2.83

Query executed successfully. AURORA\MSQLSERVER (16.0 RTM) | Aurora\frank (80) | QhatuPERU | 00:00:00 | 66 rows

Explicación: Muestra cuánto aporta cada proveedor al valor total del inventario.

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

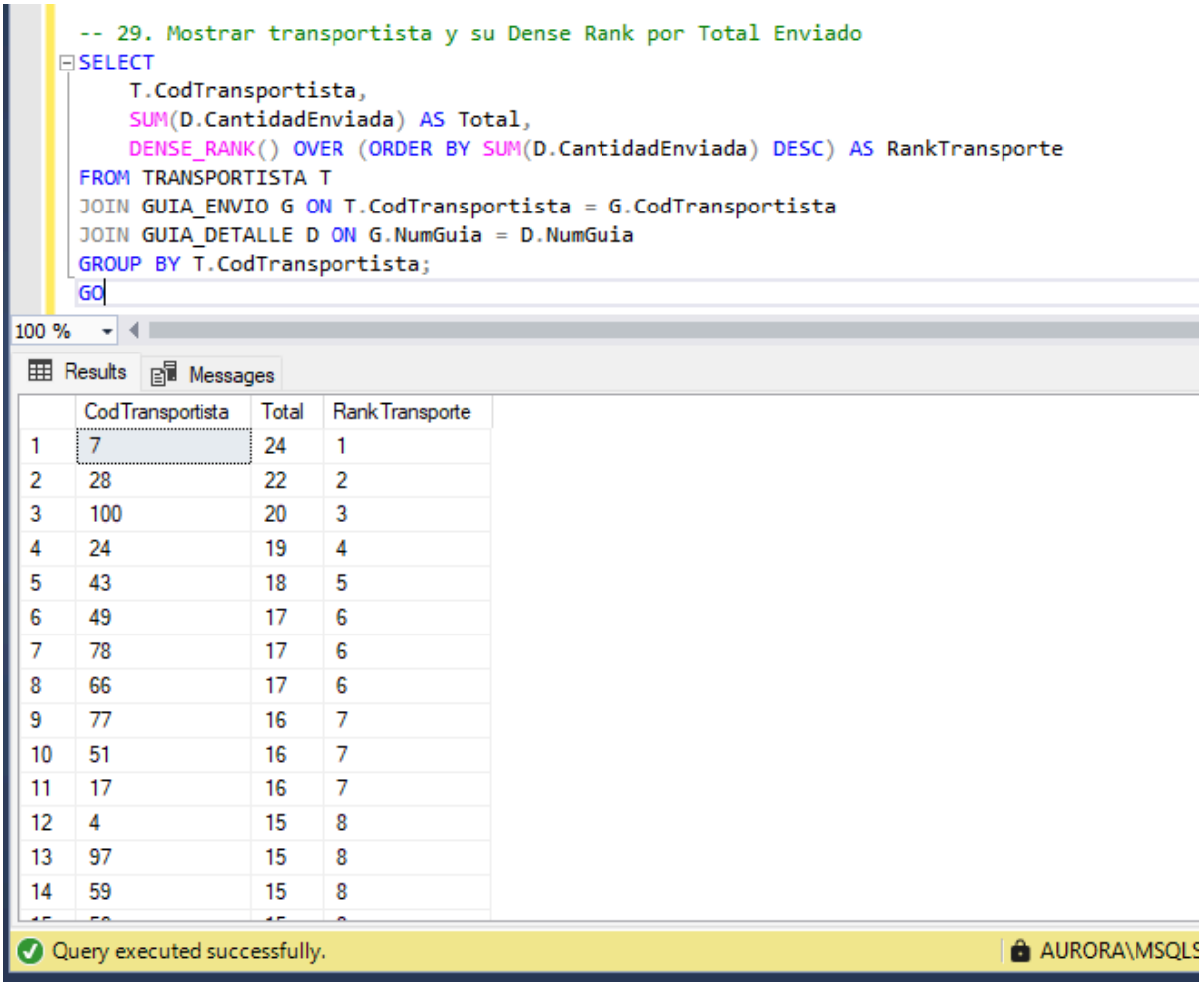
Consulta SQL:



Explicación: Usa ROW_NUMBER para filtrar los 3 artículos más caros por línea.

29. Enunciado: Mostrar transportista y su Dense Rank por Total Enviado.

Consulta SQL:



```sql
-- 29. Mostrar transportista y su Dense Rank por Total Enviado
SELECT
    T.CodTransportista,
    SUM(D.CantidadEnviada) AS Total,
    DENSE_RANK() OVER (ORDER BY SUM(D.CantidadEnviada) DESC) AS RankTransporte
FROM TRANSPORTISTA T
JOIN GUIA_ENVIO G ON T.CodTransportista = G.CodTransportista
JOIN GUIA_DETALLE D ON G.NumGuia = D.NumGuia
GROUP BY T.CodTransportista;
GO
```

100 %

Results | Messages

	CodTransportista	Total	RankTransporte
1	7	24	1
2	28	22	2
3	100	20	3
4	24	19	4
5	43	18	5
6	49	17	6
7	78	17	6
8	66	17	6
9	77	16	7
10	51	16	7
11	17	16	7
12	4	15	8
13	97	15	8
14	59	15	8

Query executed successfully. AURORA\MSQLS

Explicación: Asigna un ranking de transporte según la cantidad total enviada.

30. Enunciado: Mostrar por guía la suma acumulada por tienda.

Consulta SQL:

```sql
-- 30. Mostrar por guía la suma acumulada por tienda hasta esa guía
SELECT
    G.CodTienda,
    G.NumGuia,
    SUM(D.CantidadEnviada * D.PrecioVenta) AS TotalGuia,
    SUM(SUM(D.CantidadEnviada * D.PrecioVenta)) OVER (PARTITION BY G.CodTienda ORDER BY G.FechaSalida) AS AcumuladoTienda
FROM GUIA_ENVIO G
JOIN GUIA_DETALLE D ON G.NumGuia = D.NumGuia
GROUP BY G.CodTienda, G.NumGuia, G.FechaSalida;
GO
```

100 %

Results | Messages

	CodTienda	NumGuia	TotalGuia	AcumuladoTienda
1	1	25	2941.56	2941.56
2	1	29	4857.55	7799.11
3	3	12	7505.55	7505.55
4	3	88	19282.68	26788.23
5	3	65	18583.53	45371.76
6	5	77	10023.36	10023.36
7	5	52	20174.82	30198.18
8	9	90	6039.42	6039.42
9	12	42	11830.92	11830.92
10	14	7	6377.96	6377.96
11	14	22	3454.56	9832.52
12	14	55	19118.96	28951.48
13	15	60	2287.70	2287.70
14	16	91	318.04	318.04

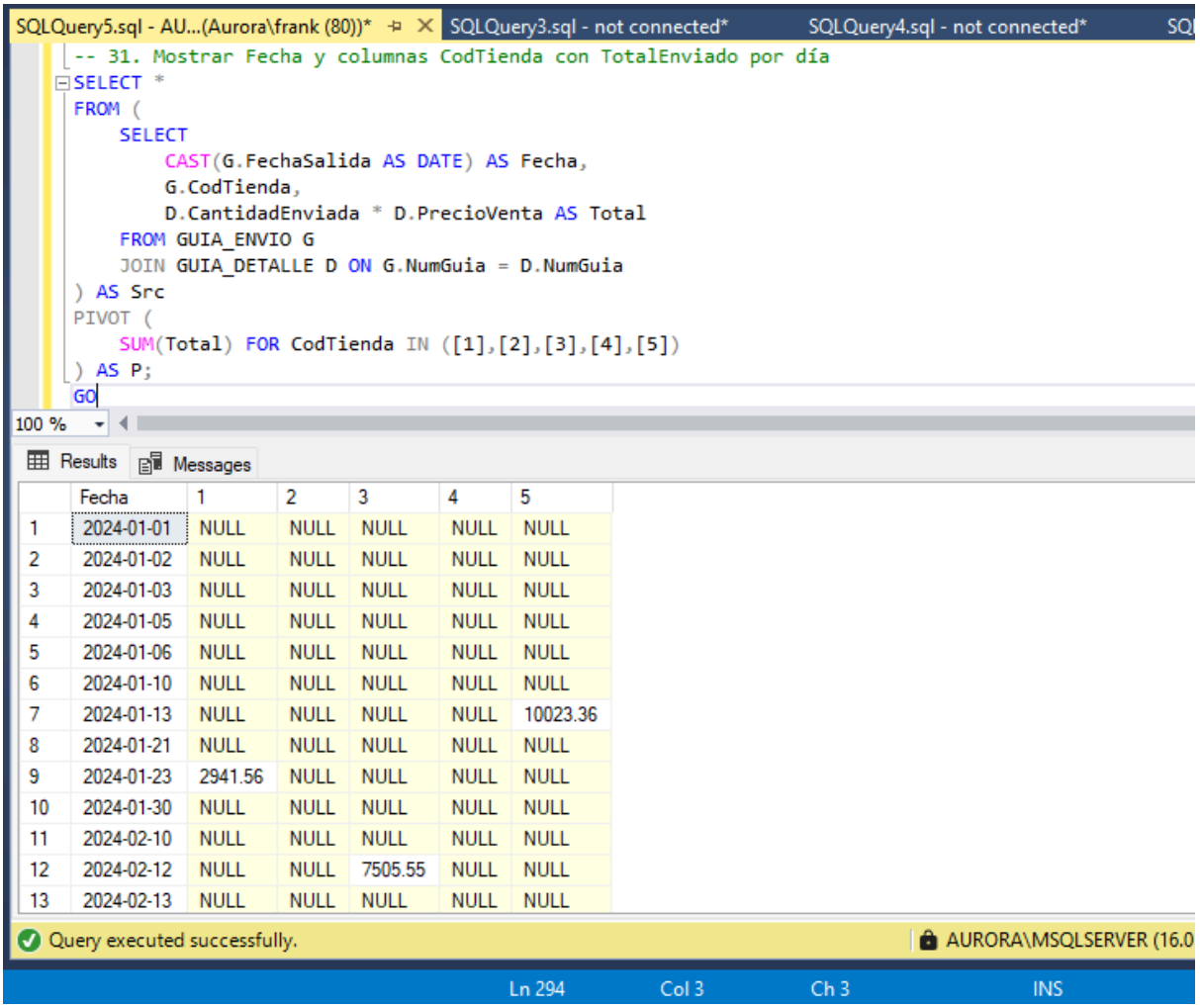
Query executed successfully. AURORA\MSQLSERVER (16.0 RTM) | Aurora\frank (80) | QhatuPERU

Explicación: Muestra el total por guía y su acumulado dentro de la misma tienda.

OPERADOR PIVOT

31. Enunciado: Mostrar Fecha y columnas CodTienda con TotalEnviado por día.

Consulta SQL:

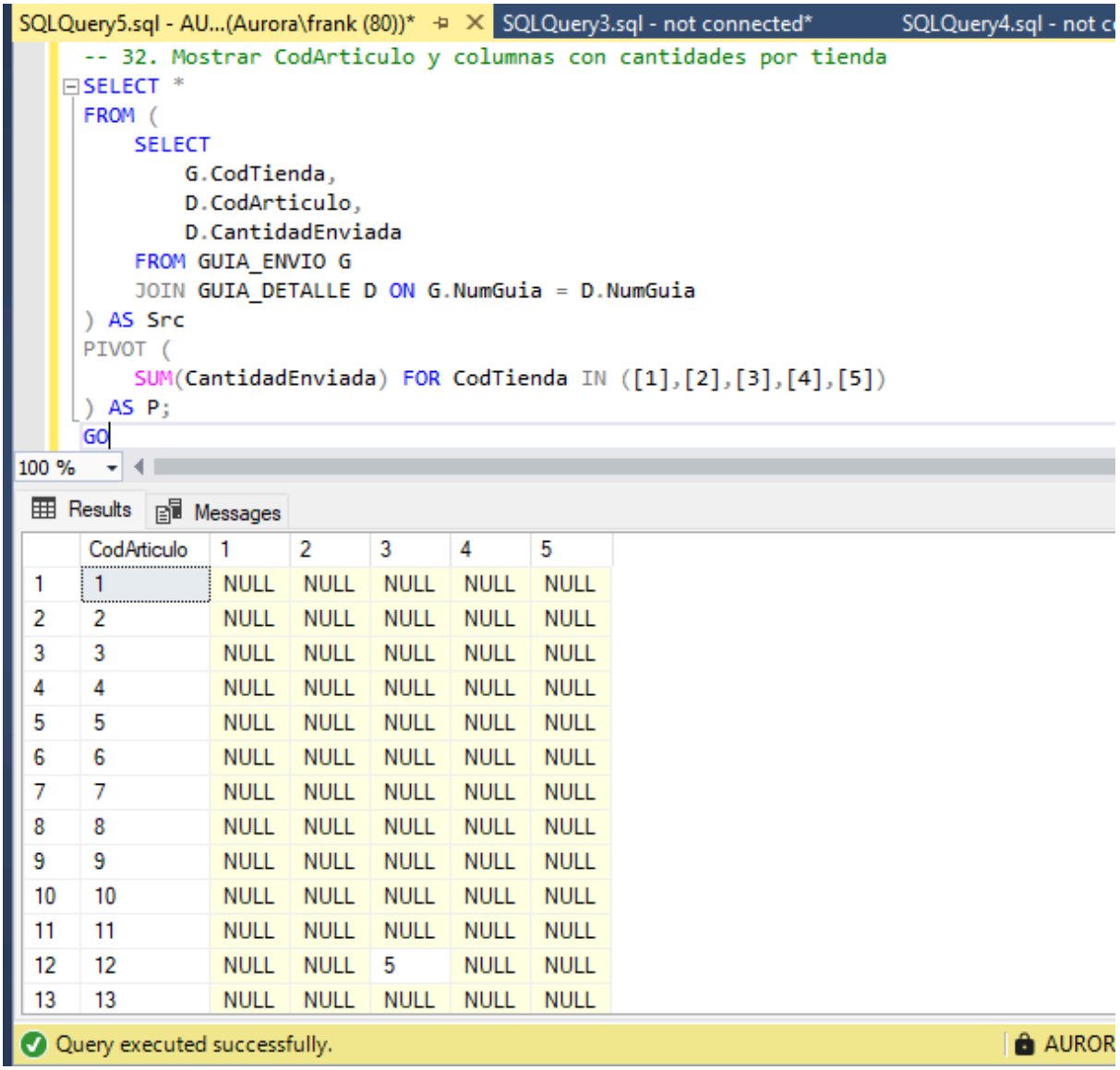


	Fecha	1	2	3	4	5
1	2024-01-01	NULL	NULL	NULL	NULL	NULL
2	2024-01-02	NULL	NULL	NULL	NULL	NULL
3	2024-01-03	NULL	NULL	NULL	NULL	NULL
4	2024-01-05	NULL	NULL	NULL	NULL	NULL
5	2024-01-06	NULL	NULL	NULL	NULL	NULL
6	2024-01-10	NULL	NULL	NULL	NULL	NULL
7	2024-01-13	NULL	NULL	NULL	NULL	10023.36
8	2024-01-21	NULL	NULL	NULL	NULL	NULL
9	2024-01-23	2941.56	NULL	NULL	NULL	NULL
10	2024-01-30	NULL	NULL	NULL	NULL	NULL
11	2024-02-10	NULL	NULL	NULL	NULL	NULL
12	2024-02-12	NULL	NULL	7505.55	NULL	NULL
13	2024-02-13	NULL	NULL	NULL	NULL	NULL

Explicación: Crea columnas dinámicas con los totales enviados por tienda por día.

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

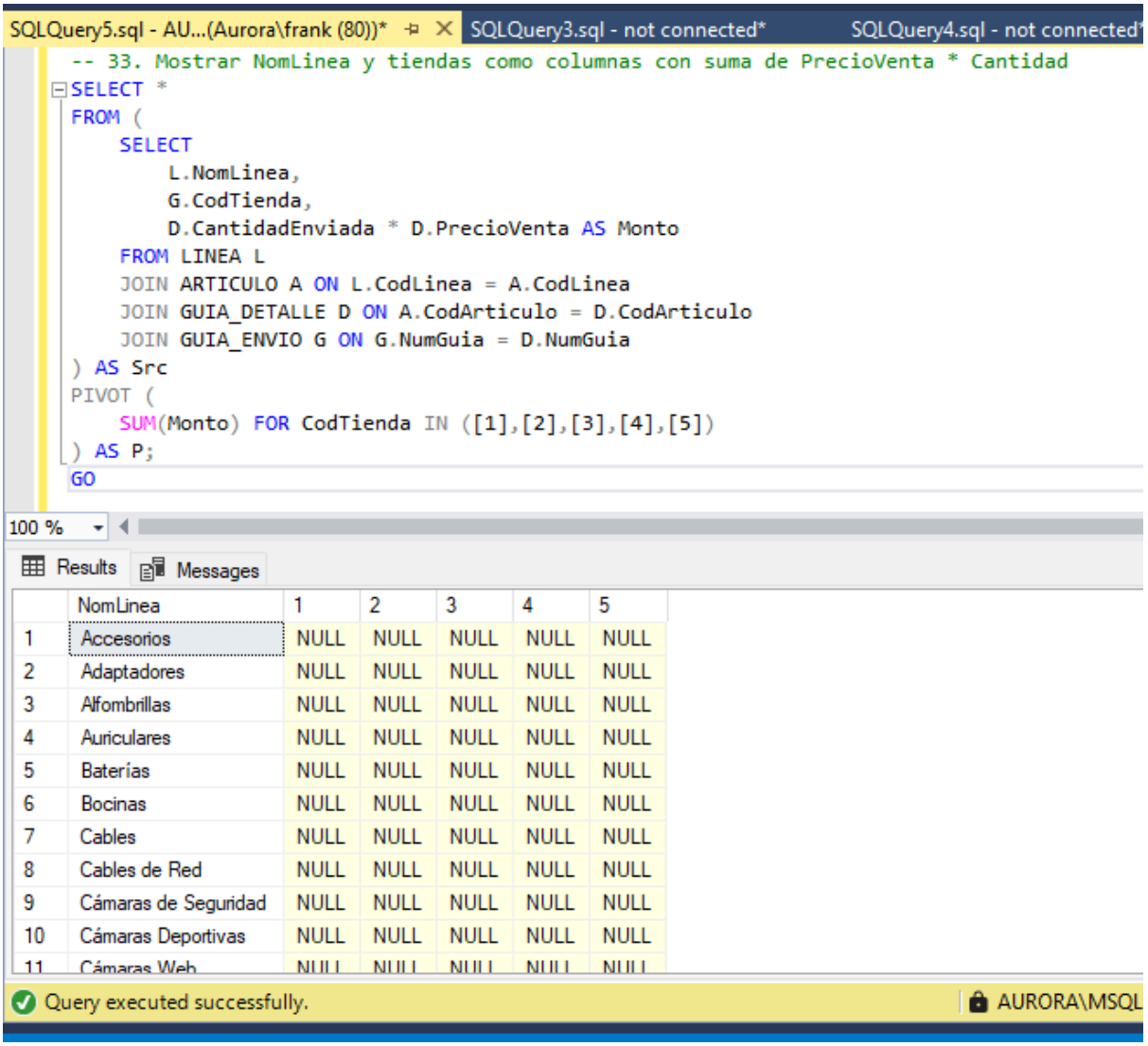
Consulta SQL:



Explicación: Transforma filas de cantidad enviada en columnas por tienda.

33. Enunciado: Mostrar NomLinea y tiendas como columnas con suma de PrecioVenta * Cantidad.

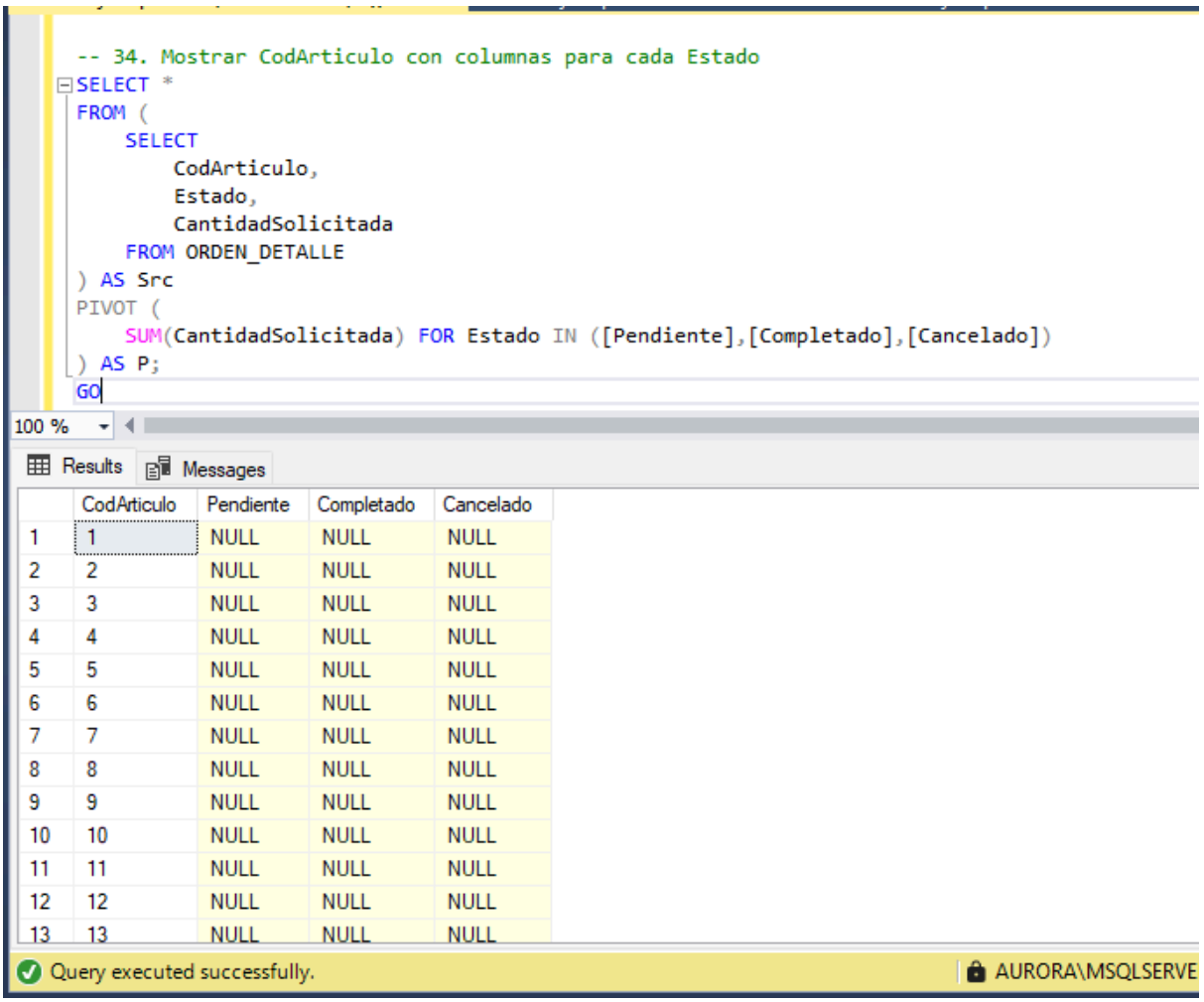
Consulta SQL:



Explicación: Resume las ventas totales por línea y tienda.

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

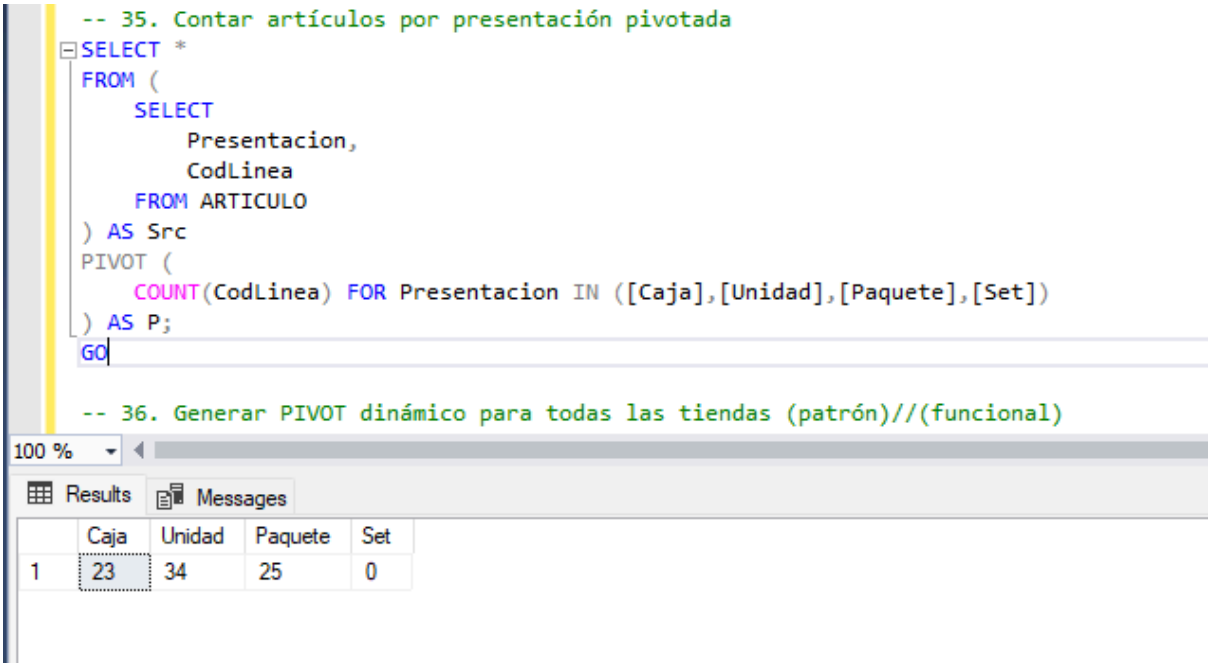
Consulta SQL:



Explicación: Muestra los artículos con cantidades separadas por estado.

35. Enunciado: Contar artículos por presentación pivotada.

Consulta SQL:



```sql
-- 35. Contar artículos por presentación pivotada
SELECT *
FROM (
    SELECT
        Presentacion,
        CodLinea
    FROM ARTICULO
) AS Src
PIVOT (
    COUNT(CodLinea) FOR Presentacion IN ([Caja],[Unidad],[Paquete],[Set])
) AS P;
GO

-- 36. Generar PIVOT dinámico para todas las tiendas (patrón)//(funcional)
```

100 %

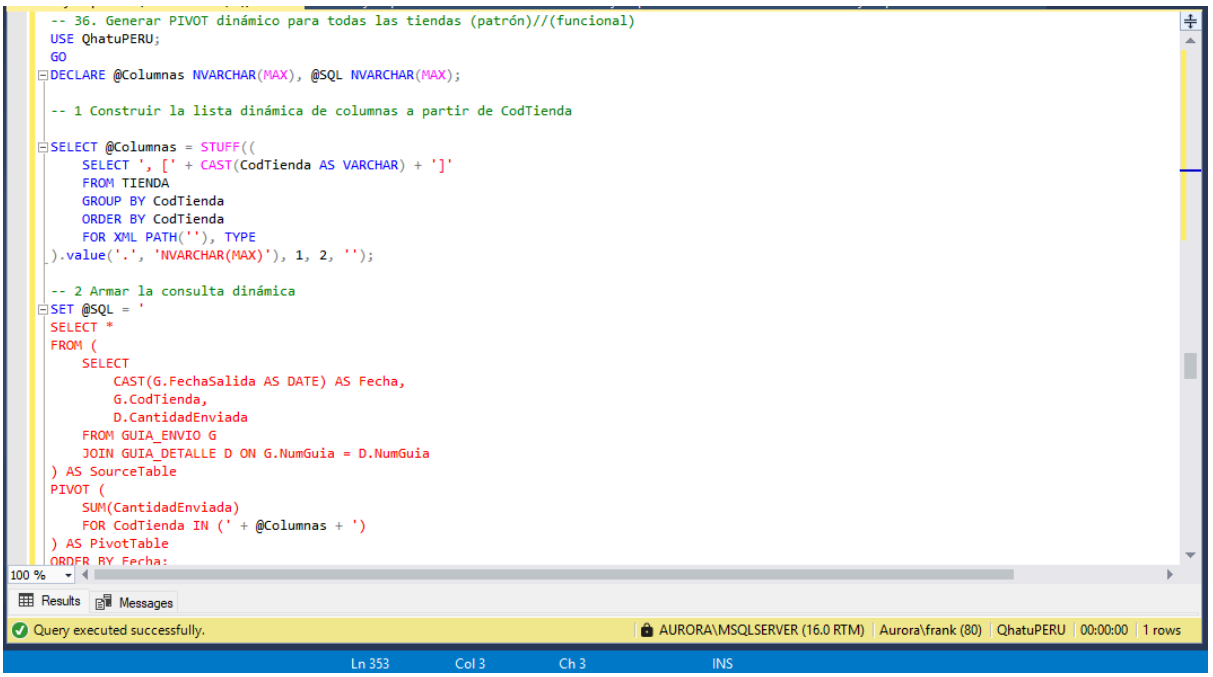
Results | Messages

	Caja	Unidad	Paquete	Set
1	23	34	25	0

Explicación: Cuenta los artículos por tipo de presentación.

36. Enunciado: Generar PIVOT dinámico para todas las tiendas.

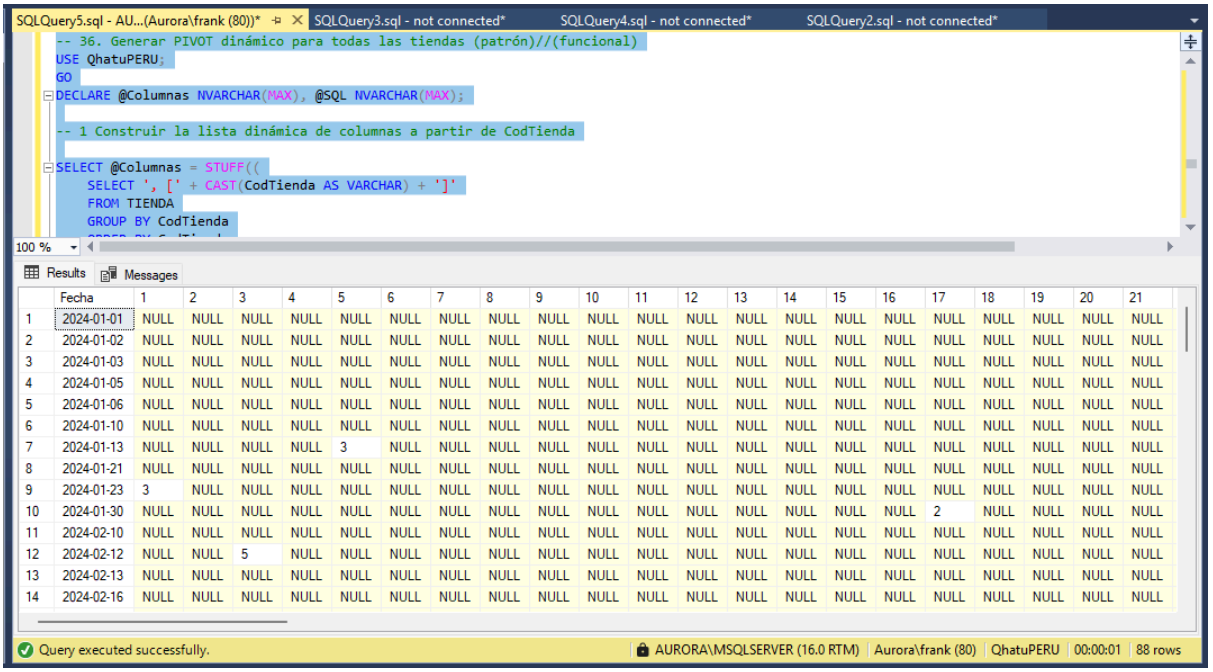
Consulta SQL:



```sql
-- 36. Generar PIVOT dinámico para todas las tiendas (patrón)///(funcional)
USE QhatuPERU;
GO
DECLARE @Columnas NVARCHAR(MAX), @SQL NVARCHAR(MAX);

-- 1 Construir la lista dinámica de columnas a partir de CodTienda

SELECT @Columnas = STUFF((
    SELECT ', [' + CAST(CodTienda AS VARCHAR) + ']'
    FROM TIENDA
    GROUP BY CodTienda
    ORDER BY CodTienda
    FOR XML PATH(''), TYPE
).value('.', 'NVARCHAR(MAX)'), 1, 2, '');

-- 2 Armar la consulta dinámica
SET @SQL = '
SELECT *
FROM (
    SELECT
        CAST(G.FechaSalida AS DATE) AS Fecha,
        G.CodTienda,
        D.CantidadEnviada
    FROM GUIA_ENVIO G
    JOIN GUIA_DETALLE D ON G.NumGuia = D.NumGuia
) AS SourceTable
PIVOT (
    SUM(CantidadEnviada)
    FOR CodTienda IN (' + @Columnas + ')
) AS PivotTable
ORDER BY Fecha;
';

-- 3 Ejecutar la consulta generada
PRINT @SQL;  -- opcional, para ver el SQL generado
EXEC sp_executesql @SQL;
GO
```

Query executed successfully. | AURORA\MSQLSERVER (16.0 RTM) | Aurora\frank (80) | QhatuPERU | 00:00:00 | 1 rows

Ln 353 Col 3 Ch 3 INS



SQLQuery5.sql - AU...(Aurora\frank (80))* | SQLQuery3.sql - not connected* | SQLQuery4.sql - not connected* | SQLQuery2.sql - not connected*

Results | Messages

	Fecha	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
1	2024-01-01	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
2	2024-01-02	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
3	2024-01-03	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
4	2024-01-05	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
5	2024-01-06	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
6	2024-01-10	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
7	2024-01-13	NULL	NULL	NULL	NULL	3	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
8	2024-01-21	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
9	2024-01-23	3	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
10	2024-01-30	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	2	NULL	NULL	NULL	NULL
11	2024-02-10	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
12	2024-02-12	NULL	NULL	5	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
13	2024-02-13	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
14	2024-02-16	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Query executed successfully. | AURORA\MSQLSERVER (16.0 RTM) | Aurora\frank (80) | QhatuPERU | 00:00:01 | 88 rows

Explicación: Crea un PIVOT adaptable al número de tiendas existentes.

37. Enunciado: Mostrar mes y columnas por transportista con totales.

Consulta SQL:

```sql
-- 37. Mostrar mes y columnas por transportista con totales
SELECT *
FROM (
    SELECT
        MONTH(G.FechaSalida) AS Mes,
        G.CodTransportista,
        D.CantidadEnviada * D.PrecioVenta AS Total
    FROM GUIA_ENVIO G
    JOIN GUIA_DETALLE D ON G.NumGuia = D.NumGuia
) AS Src
PIVOT (
    SUM(Total) FOR CodTransportista IN ([1],[2],[3],[4],[5])
) AS P;
GO

-- 38. Contar proveedores por rango de variedad de artículos
```

100 %

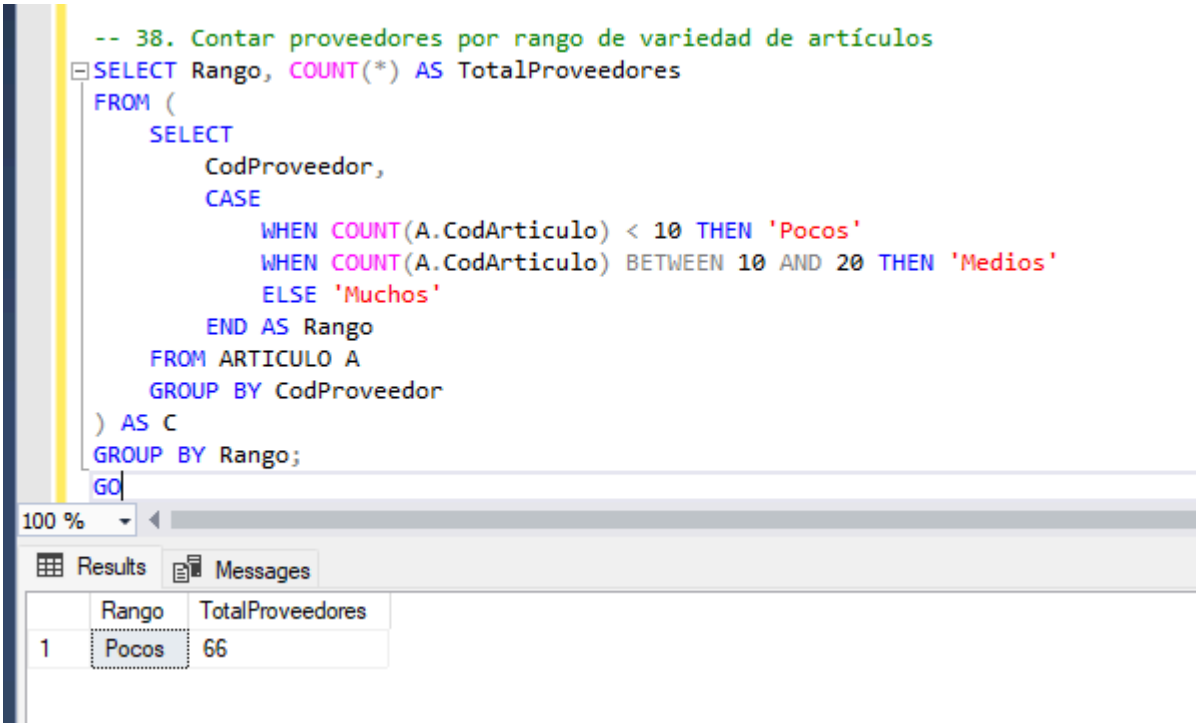
Results | Messages

	Mes	1	2	3	4	5
1	1	NULL	NULL	NULL	NULL	NULL
2	2	NULL	15411.06	NULL	NULL	NULL
3	3	NULL	NULL	NULL	NULL	NULL
4	4	NULL	NULL	NULL	NULL	NULL
5	5	NULL	NULL	NULL	18685.38	15867.96
6	6	NULL	NULL	NULL	NULL	NULL
7	7	NULL	NULL	NULL	NULL	NULL
8	8	NULL	NULL	NULL	NULL	NULL
9	9	6377.96	NULL	NULL	NULL	NULL
10	10	NULL	NULL	NULL	16926.88	NULL
11	11	NULL	NULL	NULL	NULL	NULL

Explicación: Agrupa las ventas mensuales por transportista.

38. Enunciado: Contar proveedores por rango de variedad de artículos.

Consulta SQL:



Explicación: Clasifica a los proveedores según la cantidad de artículos que manejan.

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

Consulta SQL:

SQLQuery5.sql - AU...(Aurora\frank (80))* | SQLQuery3.sql - not connected* | SQLQue

```sql
-- 39. Mostrar CodArticulo y columnas por año con monto total vendido
SELECT *
FROM (
    SELECT
        D.CodArticulo,
        YEAR(G.FechaSalida) AS Anio,
        D.CantidadEnviada * D.PrecioVenta AS Total
    FROM GUIA_DETALLE D
    JOIN GUIA_ENVIO G ON G.NumGuia = D.NumGuia
) AS Src
PIVOT (
    SUM(Total) FOR Anio IN ([2023],[2024],[2025])
) AS P;
GO

-- 40. Mostrar Mes y columnas por tienda (CASE alternativo)
```

100 %

Results | Messages

	CodArticulo	2023	2024	2025
1	1	NULL	11589.40	NULL
2	2	NULL	2638.74	NULL
3	3	NULL	17656.08	NULL
4	4	NULL	3792.81	NULL
5	5	NULL	16896.80	NULL
6	6	NULL	14145.92	NULL
7	7	NULL	6377.96	NULL
8	8	NULL	5639.20	NULL
9	9	NULL	2654.95	NULL
10	10	NULL	6319.16	NULL
11	11	NULL	7198.56	NULL

Explicación: Muestra ventas por artículo divididas por año.

40. Enunciado: Mostrar Mes y columnas por tienda (CASE alternativo).

Consulta SQL:

```sql
-- 40. Mostrar Mes y columnas por tienda (CASE alternativo)
SELECT
    MONTH(G.FechaSalida) AS Mes,
    SUM(CASE WHEN G.CodTienda = 1 THEN D.CantidadEnviada ELSE 0 END) AS Tienda1,
    SUM(CASE WHEN G.CodTienda = 2 THEN D.CantidadEnviada ELSE 0 END) AS Tienda2,
    SUM(CASE WHEN G.CodTienda = 3 THEN D.CantidadEnviada ELSE 0 END) AS Tienda3
FROM GUIA_ENVIO G
JOIN GUIA_DETALLE D ON G.NumGuia = D.NumGuia
GROUP BY MONTH(G.FechaSalida);
GO
```

100 %

Results | Messages

	Mes	Tienda1	Tienda2	Tienda3
1	1	3	0	0
2	2	0	0	5
3	3	0	0	0
4	4	0	0	9
5	5	0	0	7
6	6	0	0	0
7	7	0	0	0
8	8	0	0	0
9	9	0	0	0
10	10	5	0	0
11	11	0	0	0

Query executed successfully. AURORA\MSOLSE

Explicación: Usa CASE para pivotar manualmente las tiendas sin PIVOT.

CLÁUSULA HAVING

41. Enunciado: Mostrar CodLinea y CantArticulos donde CantArticulos ≥ 10.

Consulta SQL:

```sql
-- 41. Mostrar CodLinea y CantArticulos donde CantArticulos ≥ 10
SELECT
    CodLinea,
    COUNT(CodArticulo) AS CantArticulos
FROM ARTICULO
GROUP BY CodLinea
HAVING COUNT(CodArticulo) >= 10;
GO
```

100 %

Results | Messages

CodLinea	CantArticulos

Explicación: Filtra solo las líneas con al menos 10 artículos.

42. Enunciado: Mostrar CodProveedor y MontoTotal donde MontoTotal > 50000.

Consulta SQL:

```sql
-- 42. Mostrar CodProveedor y MontoTotal donde MontoTotal > 50000
SELECT
    CodProveedor,
    SUM(PrecioProveedor * StockActual) AS MontoTotal
FROM ARTICULO
GROUP BY CodProveedor
HAVING SUM(PrecioProveedor * StockActual) > 50000;
GO
```

100 %

Results | Messages

	CodProveedor	MontoTotal
1	3	232375.90
2	7	146960.25
3	10	65863.62
4	11	79857.88
5	12	101671.83
6	13	151437.60
7	14	258923.22
8	15	72064.80
9	16	290344.60
10	17	95822.42
11	18	76941.48
12	20	205575.68
13	22	227058.15
14	23	181468.20
15	25	231955.85
16	26	274842.96

Query executed successfully.

Explicación: Muestra proveedores con inventario valorizado superior a 50,000.

43. Enunciado: Mostrar CodTienda y PromedioGuía ≥ 1000.

Consulta SQL:

```sql
-- 43. Mostrar CodTienda y PromedioGuía donde PromedioGuia ≥ 1000
SELECT
    CodTienda,
    AVG(D.CantidadEnviada * D.PrecioVenta) AS PromedioGuia
FROM GUIA_ENVIO G
JOIN GUIA_DETALLE D ON G.NumGuia = D.NumGuia
GROUP BY CodTienda
HAVING AVG(D.CantidadEnviada * D.PrecioVenta) >= 1000;
GO
```

100 %

Results | Messages

	CodTienda	PromedioGuia
1	1	3899.555
2	3	15123.92
3	5	15099.09
4	9	6039.42
5	12	11830.92
6	14	9650.4933
7	15	2287.70
8	17	8623.86
9	18	11494.80
10	19	10309.32
11	20	9281.2566
12	22	25456.32
13	23	7096.67
14	25	11243.71
15	26	15114.675
16	27	20111.77

Query executed successfully.

Explicación: Calcula tiendas cuyo promedio de guía supera los 1000 soles.

44. Enunciado: Mostrar CodArticulo y TotalSolicitado ≤ 500.

Consulta SQL:

```sql
-- 44. Mostrar CodArticulo y TotalSolicitado ≤ 500
SELECT
    CodArticulo,
    SUM(CantidadSolicitada) AS TotalSolicitado
FROM ORDEN_DETALLE
GROUP BY CodArticulo
HAVING SUM(CantidadSolicitada) <= 500;
GO
```

100 %

Results | Messages

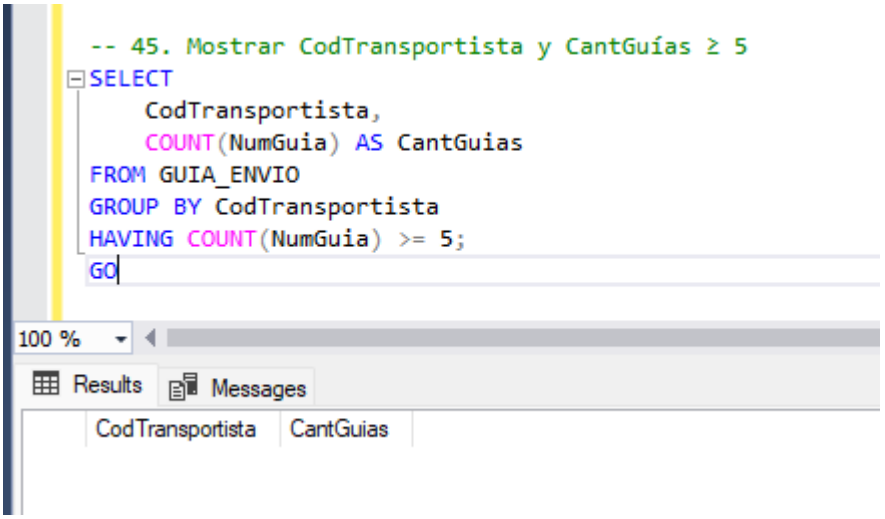
	CodArticulo	TotalSolicitado
1	1	6
2	2	15
3	3	14
4	4	4
5	5	13
6	6	15
7	7	7
8	8	1
9	9	7
10	10	10
11	11	15
12	12	2
13	13	12
14	14	12
15	15	15
16	16	2

Query executed successfully.

Explicación: Filtra artículos con pocas solicitudes.

45. Enunciado: Mostrar CodTransportista y CantGuías ≥ 5.

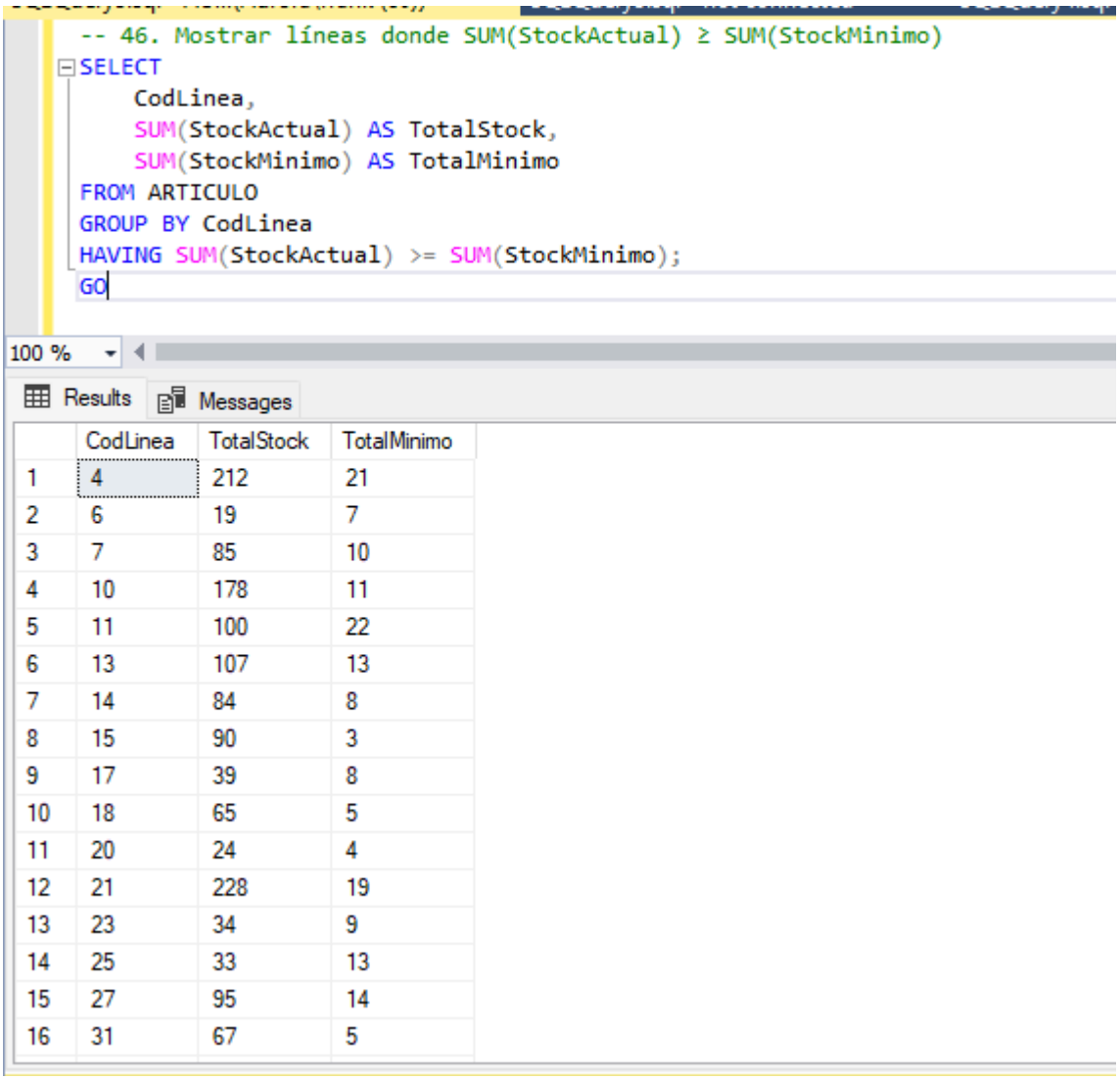
Consulta SQL:



Explicación: Selecciona transportistas con al menos 5 guías emitidas.

46. Enunciado: Mostrar líneas donde SUM(StockActual) ≥ SUM(StockMinimo).

Consulta SQL:



```sql
-- 46. Mostrar líneas donde SUM(StockActual) ≥ SUM(StockMinimo)
SELECT
    CodLinea,
    SUM(StockActual) AS TotalStock,
    SUM(StockMinimo) AS TotalMinimo
FROM ARTICULO
GROUP BY CodLinea
HAVING SUM(StockActual) >= SUM(StockMinimo);
GO
```

100 %

Results | Messages

	CodLinea	TotalStock	TotalMinimo
1	4	212	21
2	6	19	7
3	7	85	10
4	10	178	11
5	11	100	22
6	13	107	13
7	14	84	8
8	15	90	3
9	17	39	8
10	18	65	5
11	20	24	4
12	21	228	19
13	23	34	9
14	25	33	13
15	27	95	14
16	31	67	5

Explicación: Comprueba si las líneas cumplen su stock mínimo.

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

Consulta SQL:

```sql
-- 47. Mostrar proveedores donde MAX(PrecioProveedor) > 100
SELECT
    CodProveedor,
    MAX(PrecioProveedor) AS MaxPrecio
FROM ARTICULO
GROUP BY CodProveedor
HAVING MAX(PrecioProveedor) > 100;
GO
```

100 %

Results | Messages

	CodProveedor	MaxPrecio
1	2	767.34
2	3	1811.22
3	5	329.03
4	7	2036.92
5	8	131.75
6	10	1176.74
7	11	1857.16
8	12	1517.49
9	13	2103.30
10	14	2610.54
11	15	800.72
12	16	2783.23
13	17	2588.25
14	18	777.42
15	20	1912.71
16	22	2309.26

Explicación: Muestra los proveedores con al menos un artículo costoso.

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

Consulta SQL:

```sql
-- 48. Mostrar tiendas con AVG(CantidadEnviada) < 50 y COUNT(NumGuia) ≥ 10
SELECT
    CodTienda,
    AVG(D.CantidadEnviada) AS Promedio,
    COUNT(G.NumGuia) AS TotalGuias
FROM GUIA_ENVIO G
JOIN GUIA_DETALLE D ON G.NumGuia = D.NumGuia
GROUP BY CodTienda
HAVING AVG(D.CantidadEnviada) < 50 AND COUNT(G.NumGuia) >= 10;
GO
```

100 %

Results | Messages

CodTienda	Promedio	TotalGuias

Explicación: Muestra tiendas con bajo promedio de envío pero muchas guías.

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

Consulta SQL:

```sql
-- 49. Mostrar CodLínea donde (MAX(Precio) - MIN(Precio)) > 20
SELECT
    CodLinea,
    (MAX(PrecioProveedor) - MIN(PrecioProveedor)) AS Diferencia
FROM ARTICULO
GROUP BY CodLinea
HAVING (MAX(PrecioProveedor) - MIN(PrecioProveedor)) > 20;
GO

-- 50. Mostrar CodProveedor con COUNT(artículos) donde AVG(StockActual) ≥ 20 y COUNT ≥ 5
```

100 %

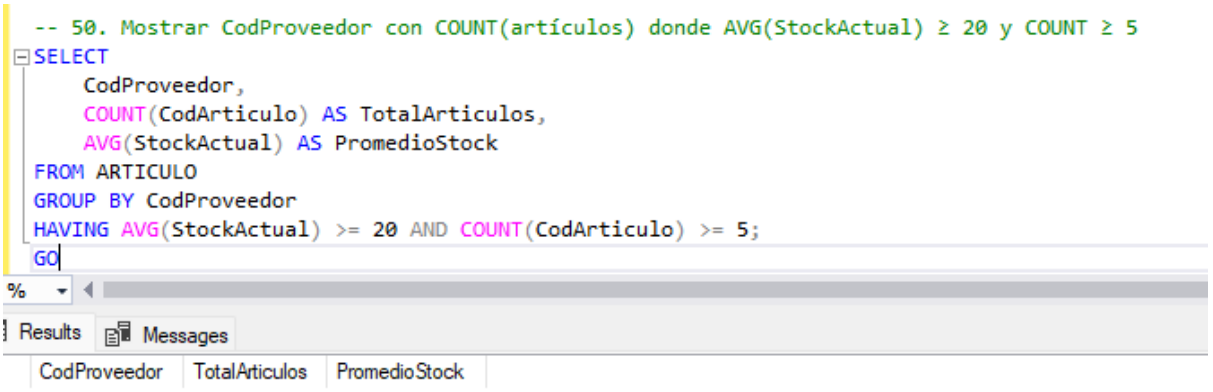
Results | Messages

	CodLinea	Diferencia
1	4	1128.64
2	10	303.21
3	11	519.45
4	13	466.99
5	21	1772.00
6	25	223.69
7	33	63.62
8	35	2166.67
9	37	462.15
10	38	2238.66
11	42	560.13
12	44	2439.68
13	56	755.94
14	59	650.49
15	62	1829.19
16	64	796.26

Explicación: Evalúa la diferencia de precios dentro de cada línea.

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

Consulta SQL:



```sql
-- 50. Mostrar CodProveedor con COUNT(artículos) donde AVG(StockActual) ≥ 20 y COUNT ≥ 5
SELECT
    CodProveedor,
    COUNT(CodArticulo) AS TotalArticulos,
    AVG(StockActual) AS PromedioStock
FROM ARTICULO
GROUP BY CodProveedor
HAVING AVG(StockActual) >= 20 AND COUNT(CodArticulo) >= 5;
GO
```


Explicación: Selecciona proveedores con inventario promedio alto y buena variedad.