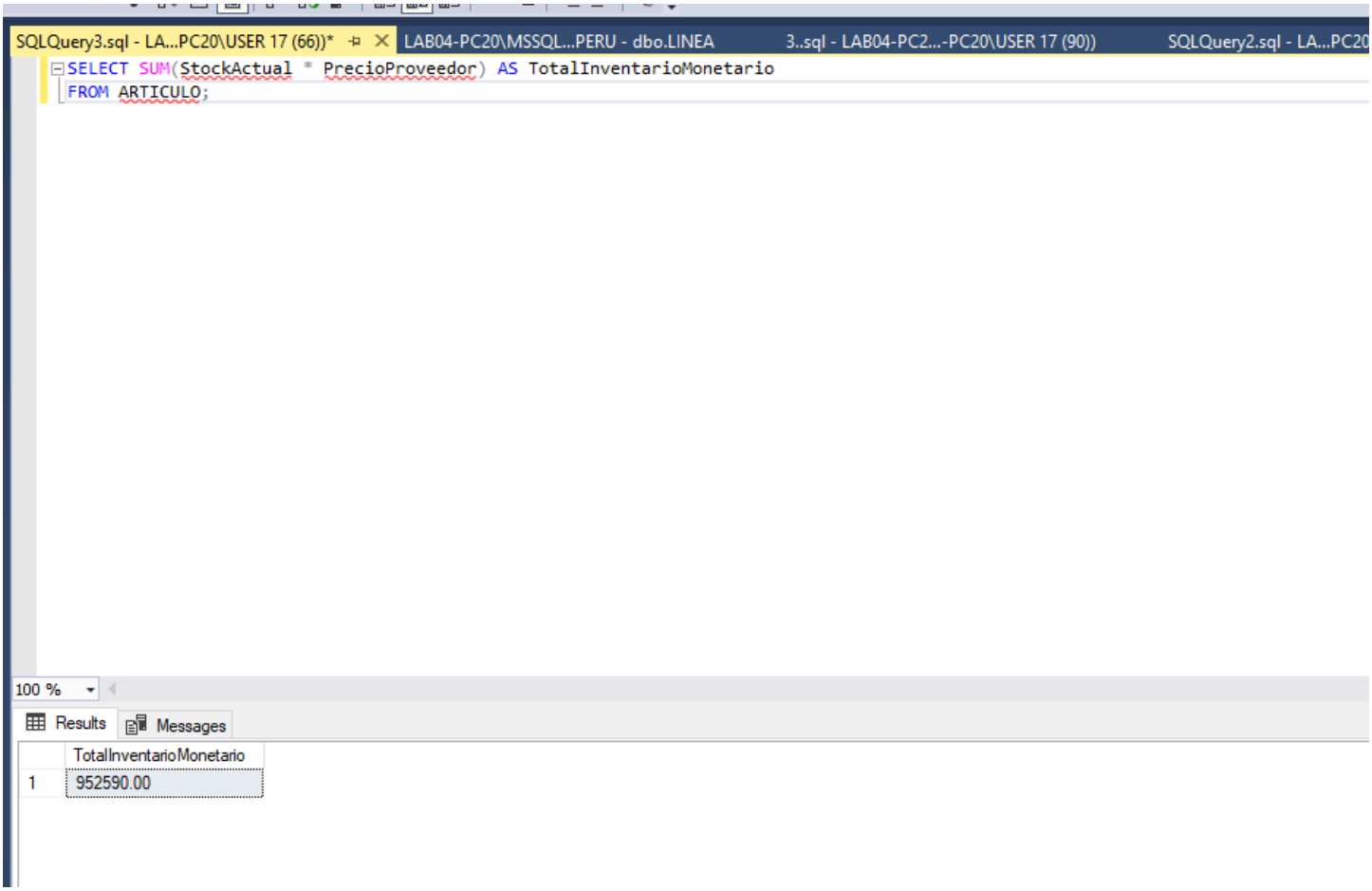
Contenido de Consultas SQL para QhatuPERU

Ejercicios 1 - 10 (Consultas Simples)

1. Calcular el total monetario del Inventario.

SQL

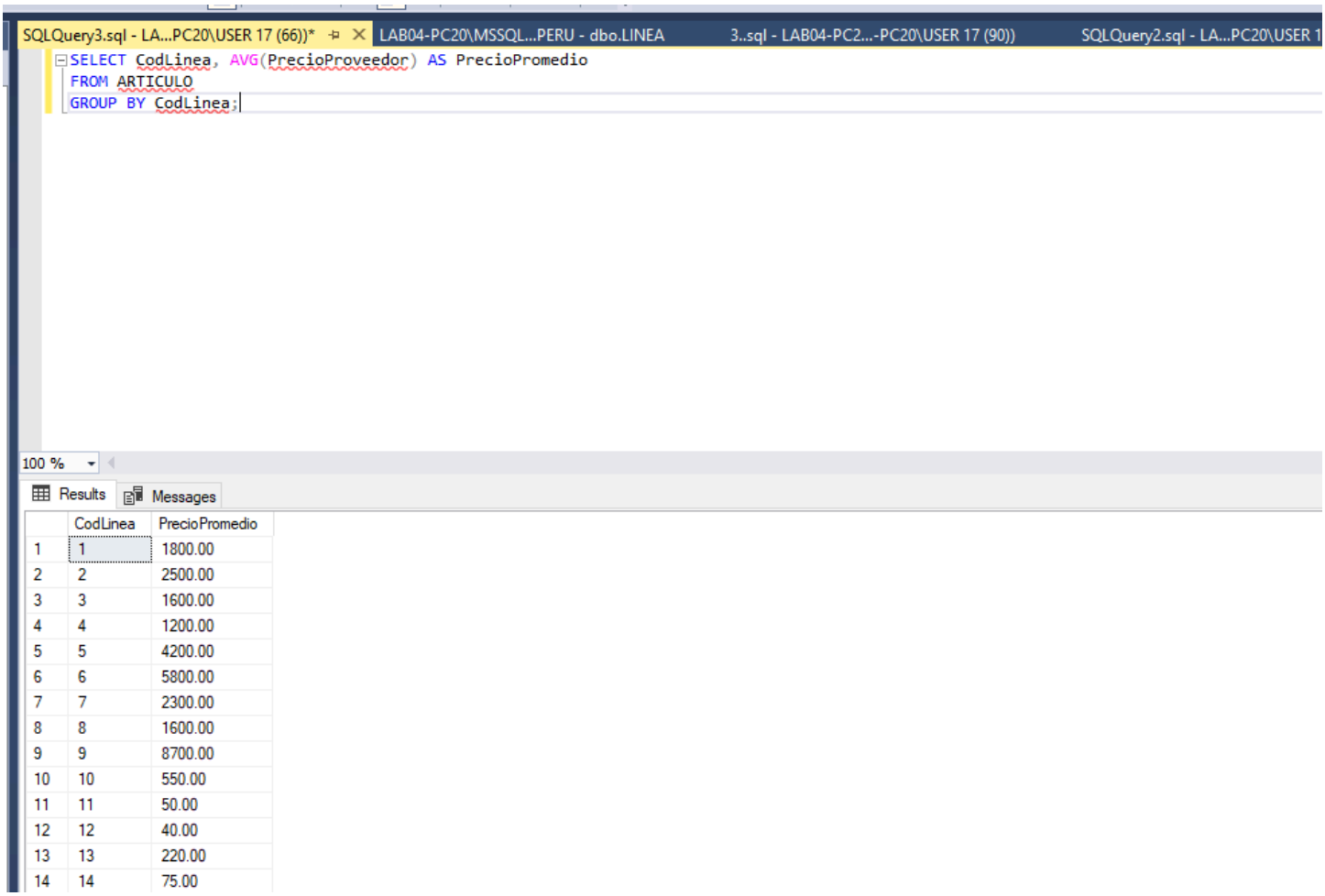
```
SELECT SUM(StockActual * PrecioProveedor) AS TotalInventarioMonetario
FROM ARTICULO;
```



2. Obtener CodLinea y PrecioProveedor promedio.

SQL

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



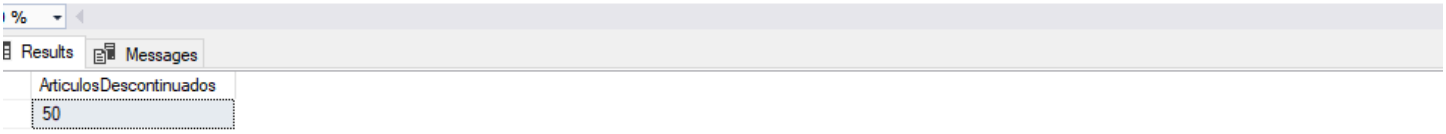
	CodLinea	PrecioPromedio
1	1	1800.00
2	2	2500.00
3	3	1600.00
4	4	1200.00
5	5	4200.00
6	6	5800.00
7	7	2300.00
8	8	1600.00
9	9	8700.00
10	10	550.00
11	11	50.00
12	12	40.00
13	13	220.00
14	14	75.00

3. Contar artículos descontinuados.

SQL

```
SELECT COUNT(CodArticulo) AS ArticulosDescontinuados

FROM ARTICULO

WHERE Descontinuado = 1;
```

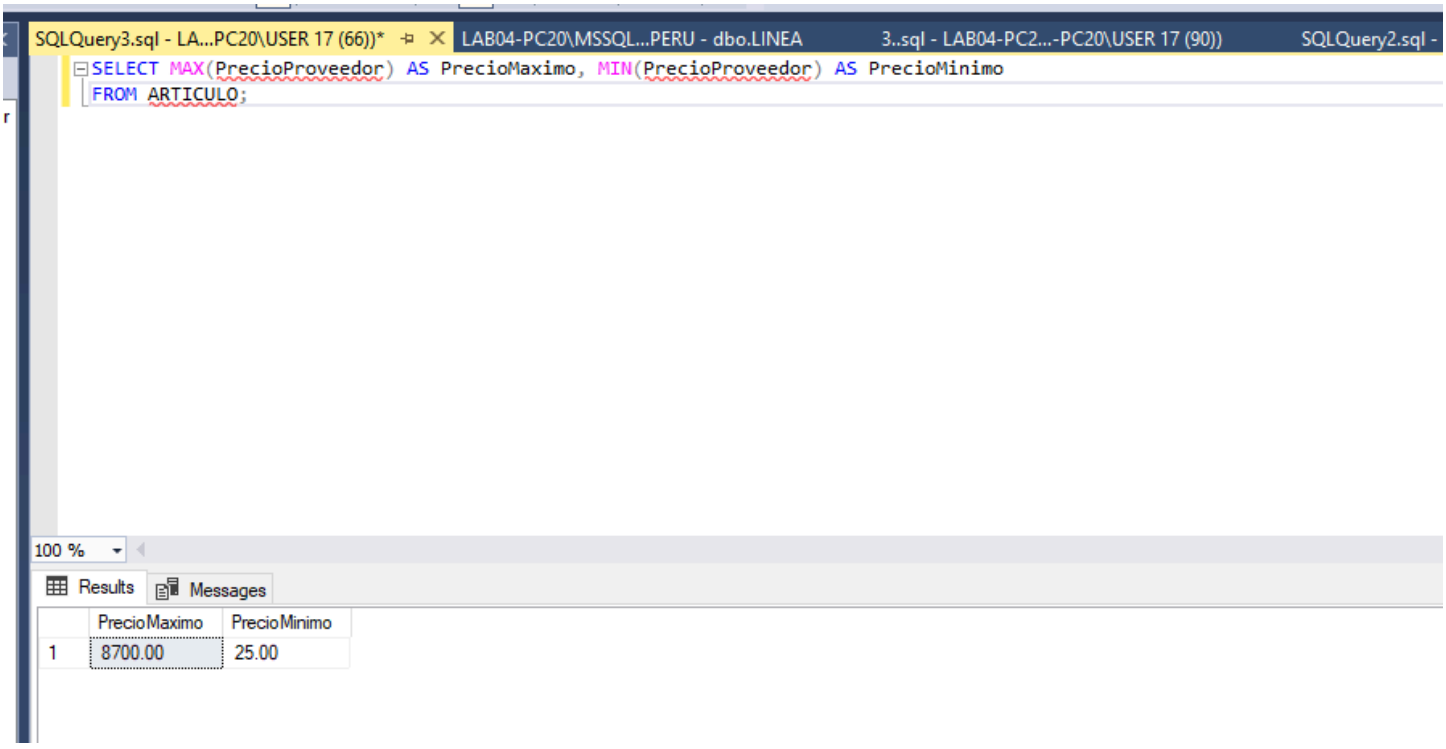
```
SELECT COUNT(CodArticulo) AS ArticulosDescontinuados
FROM ARTICULO
WHERE Descontinuado = 0;
```



4. Mostrar PrecioMaximo y PrecioMinimo del catálogo.

SQL

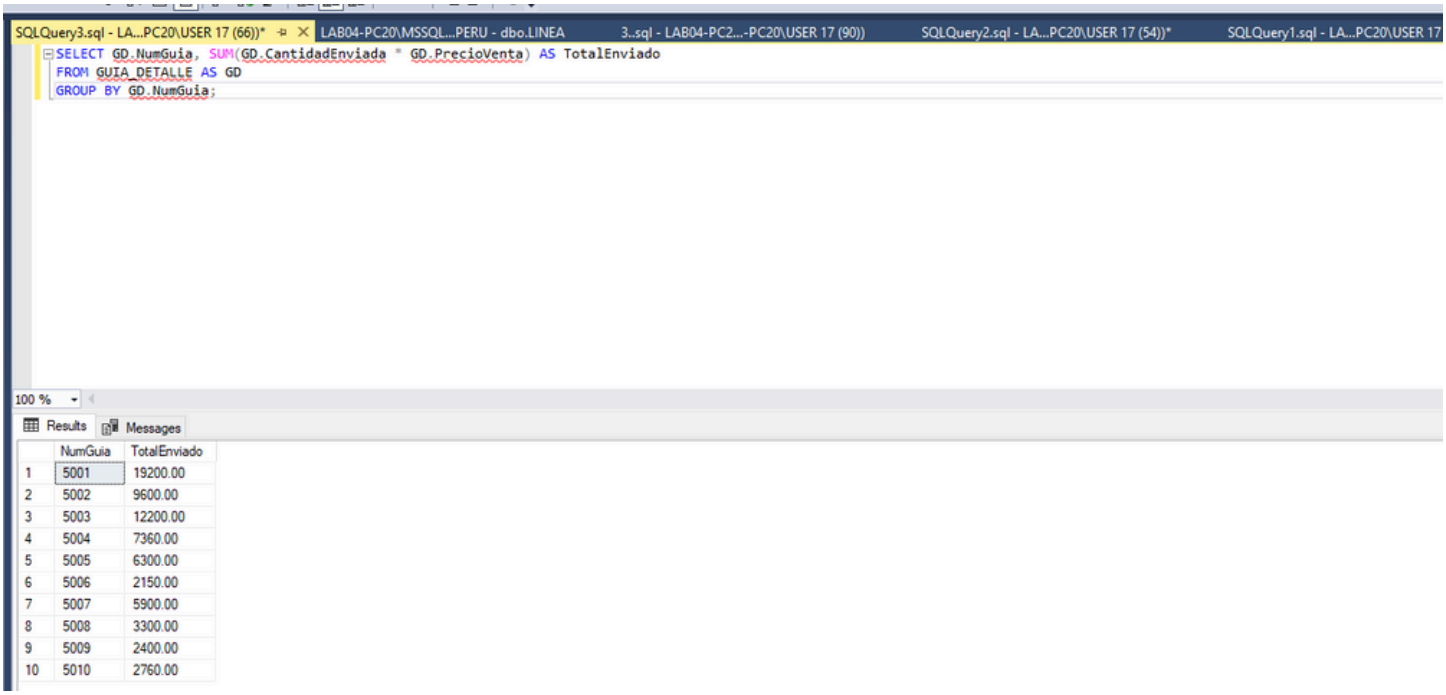
```
SELECT MAX(PrecioProveedor) AS PrecioMaximo, MIN(PrecioProveedor) AS PrecioMinimo

FROM ARTICULO;
```



5. Mostrar el valor Total enviado por guía.

SQL

```
SELECT GD.NumGuia, SUM(GD.CantidadEnviada * GD.PrecioVenta) AS
TotalEnviado

FROM GUIA_DETALLE AS GD

GROUP BY GD.NumGuia;
```



	NumGuia	TotalEnviado
1	5001	19200.00
2	5002	9600.00
3	5003	12200.00
4	5004	7360.00
5	5005	6300.00
6	5006	2150.00
7	5007	5900.00
8	5008	3300.00
9	5009	2400.00
10	5010	2760.00

6. Para cada CodArticulo, mostrar TotalSolicitado.

SQL

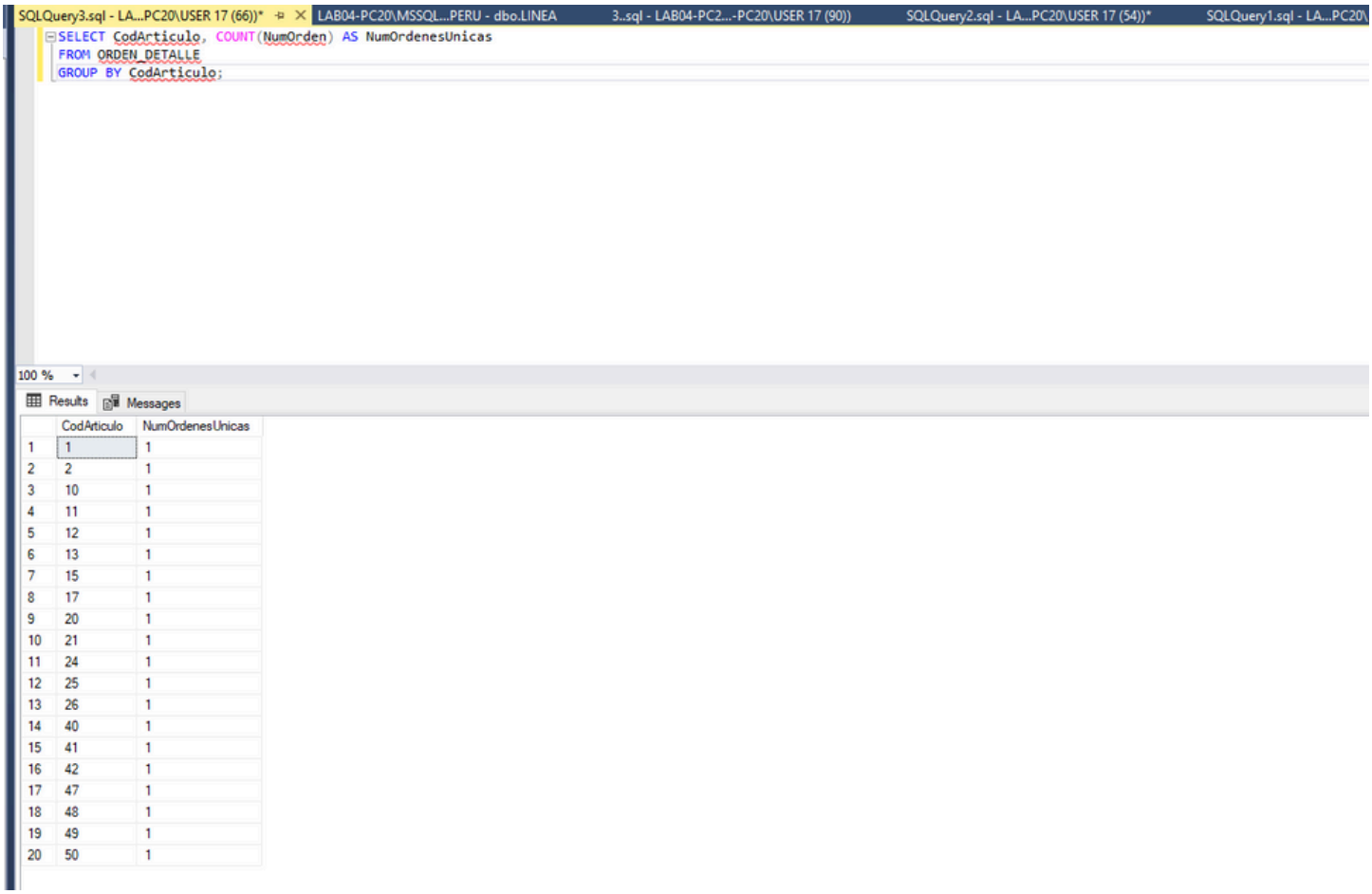
```
SELECT OD.CodArticulo, SUM(OD.CantidadSolicitada) AS TotalSolicitado

FROM ORDEN_DETALLE AS OD

GROUP BY OD.CodArticulo;
```

	CodArticulo	TotalSolicitado
1	1	10
2	2	5
3	10	20
4	11	50
5	12	50
6	13	15
7	15	10
8	17	8
9	20	30
10	21	20
11	24	5
12	25	8
13	26	15
14	40	5
15	41	10
16	42	8
17	47	3
18	48	4
19	49	10

7. Contar órdenes únicas que incluyan cada artículo.

SQL

```
SELECT CodArticulo, COUNT(NumOrden) AS NumOrdenesUnicas
FROM ORDEN_DETALLE
GROUP BY CodArticulo;
```

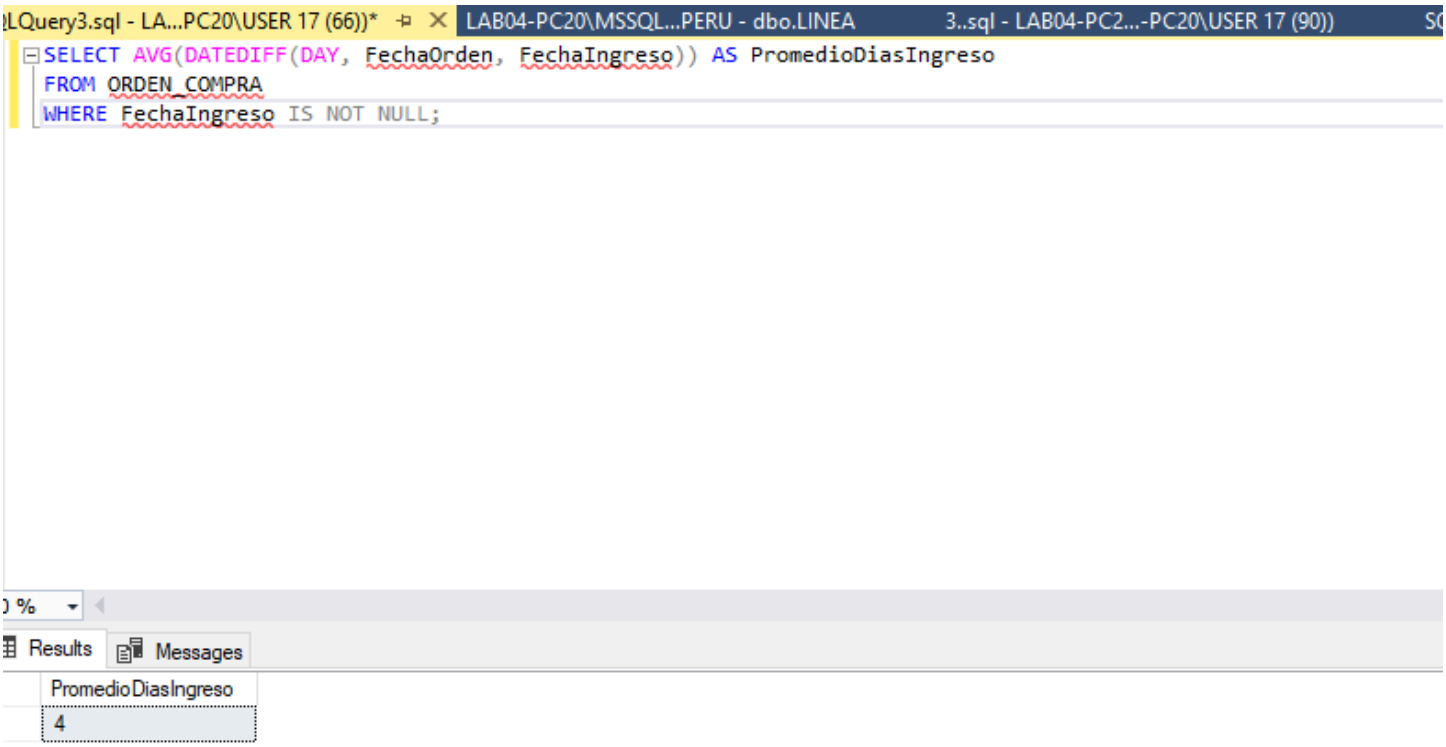


	CodArticulo	NumOrdenesUnicas
1	1	1
2	2	1
3	10	1
4	11	1
5	12	1
6	13	1
7	15	1
8	17	1
9	20	1
10	21	1
11	24	1
12	25	1
13	26	1
14	40	1
15	41	1
16	42	1
17	47	1
18	48	1
19	49	1
20	50	1

8. Calcular promedio de días por todas las órdenes con FechaIngreso.

SQL

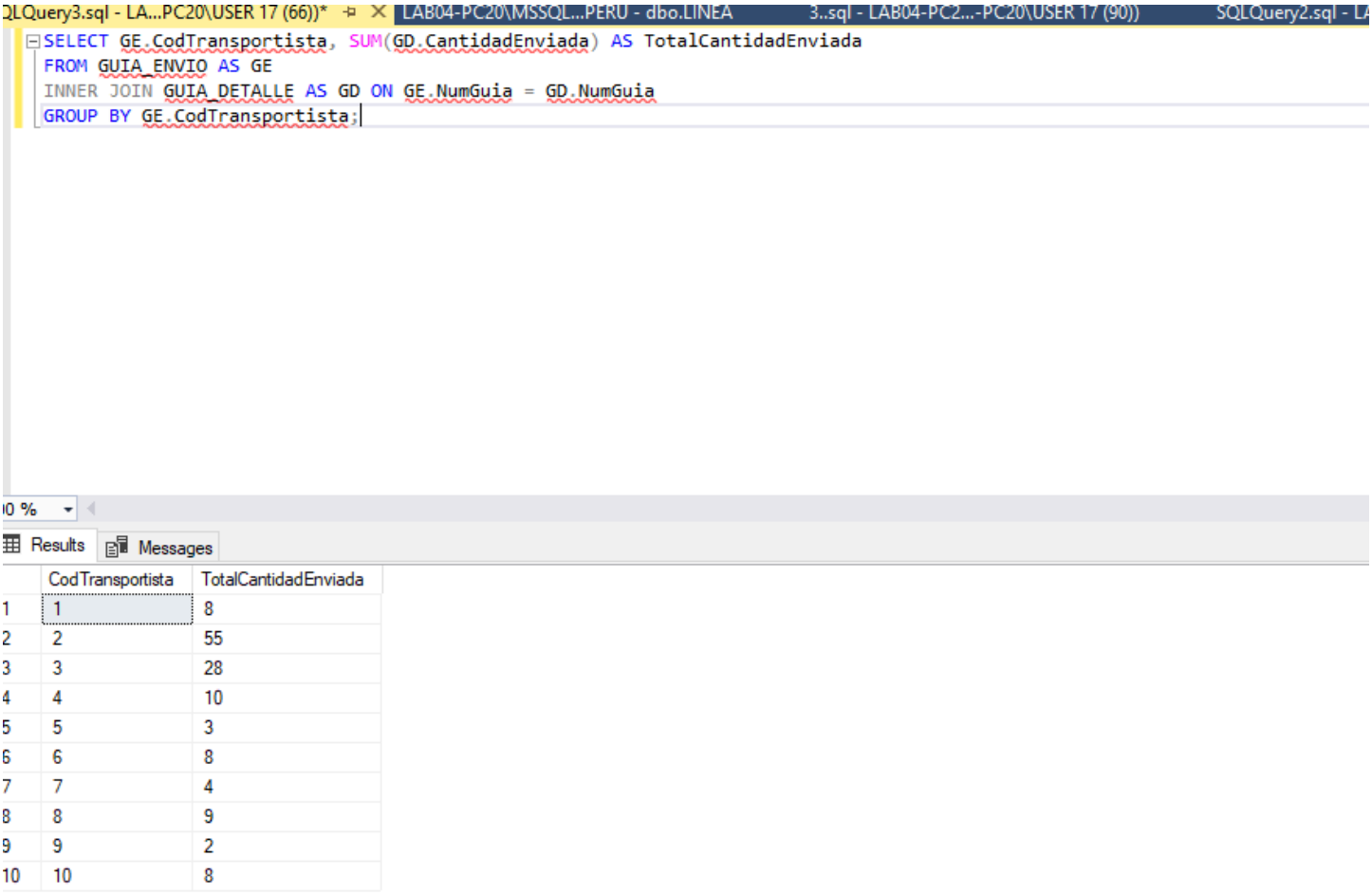
```
SELECT AVG(DATEDIFF(DAY, FechaOrden, FechaIngreso)) AS PromedioDiasIngreso

FROM ORDEN_COMPRA

WHERE FechaIngreso IS NOT NULL;
```



9. Mostrar CantidadEnviada por CodTransportista.

SQL

```
SELECT GE.CodTransportista, SUM(GD.CantidadEnviada) AS
TotalCantidadEnviada

FROM GUIA_ENVIO AS GE

INNER JOIN GUIA_DETALLE AS GD ON GE.NumGuia = GD.NumGuia

GROUP BY GE.CodTransportista;
```



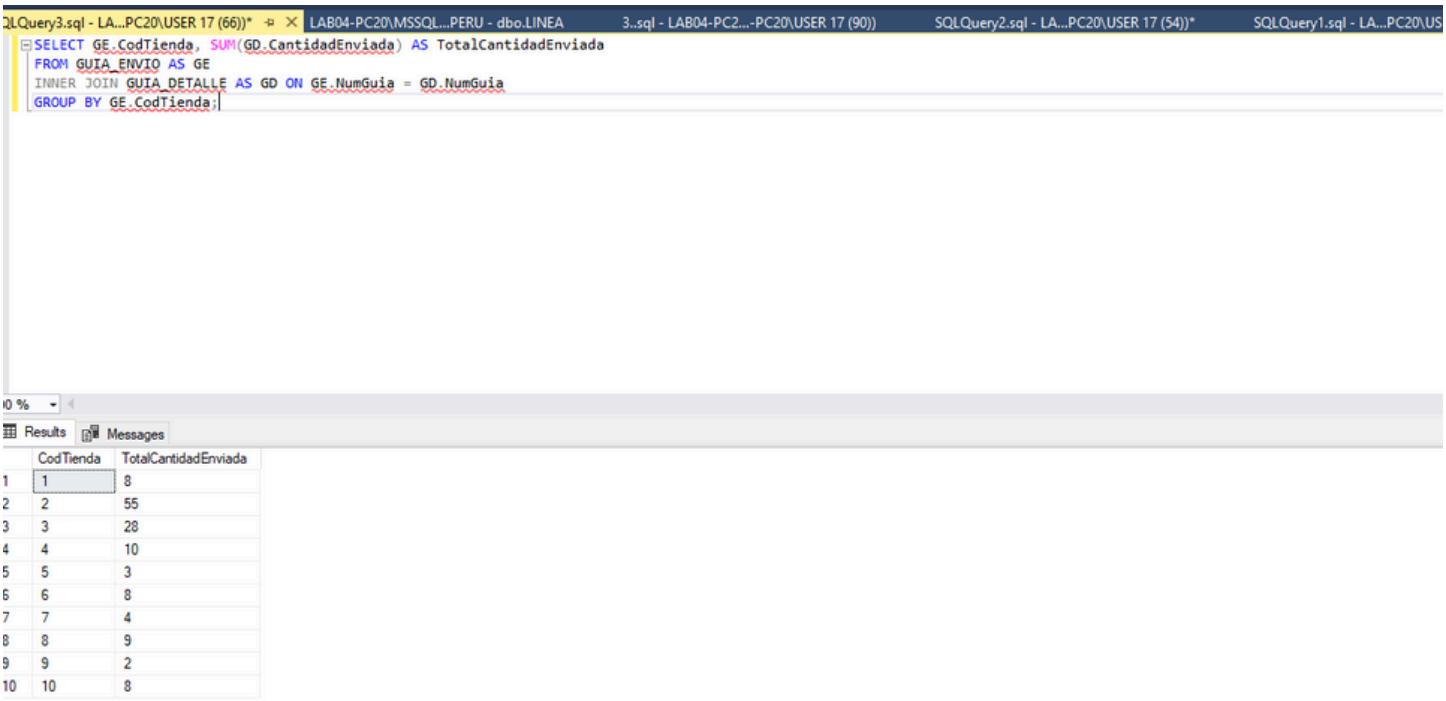
	CodTransportista	TotalCantidadEnviada
1	1	8
2	2	55
3	3	28
4	4	10
5	5	3
6	6	8
7	7	4
8	8	9
9	9	2
10	10	8

10. Sumar CantidadEnviada por Tienda.

SQL

```
SELECT GE.CodTienda, SUM(GD.CantidadEnviada) AS TotalCantidadEnviada
FROM GUIA_ENVIO AS GE
INNER JOIN GUIA_DETALLE AS GD ON GE.NumGuia = GD.NumGuia
GROUP BY GE.CodTienda;
```

Ejercicios 11 - 20 (Consultas con GROUP BY)



```
SELECT GE.CodTienda, SUM(GD.CantidadEnviada) AS TotalCantidadEnviada
FROM GUIA_ENVIO AS GE
INNER JOIN GUIA_DETALLE AS GD ON GE.NumGuia = GD.NumGuia
GROUP BY GE.CodTienda;
```

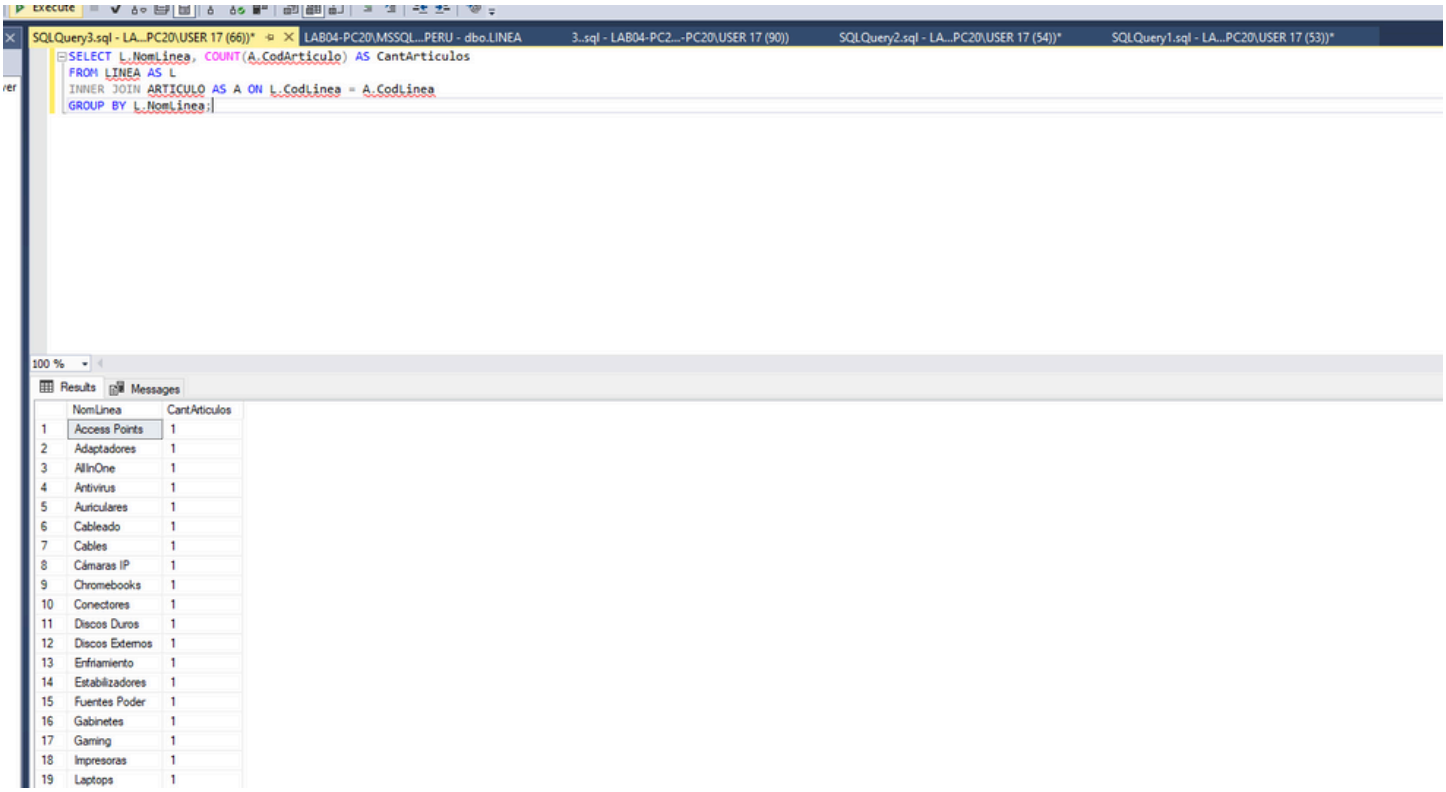
Results | Messages

	CodTienda	TotalCantidadEnviada
1	1	8
2	2	55
3	3	28
4	4	10
5	5	3
6	6	8
7	7	4
8	8	9
9	9	2
10	10	8

11. Mostrar NomLinea y CantArticulos.

SQL

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



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

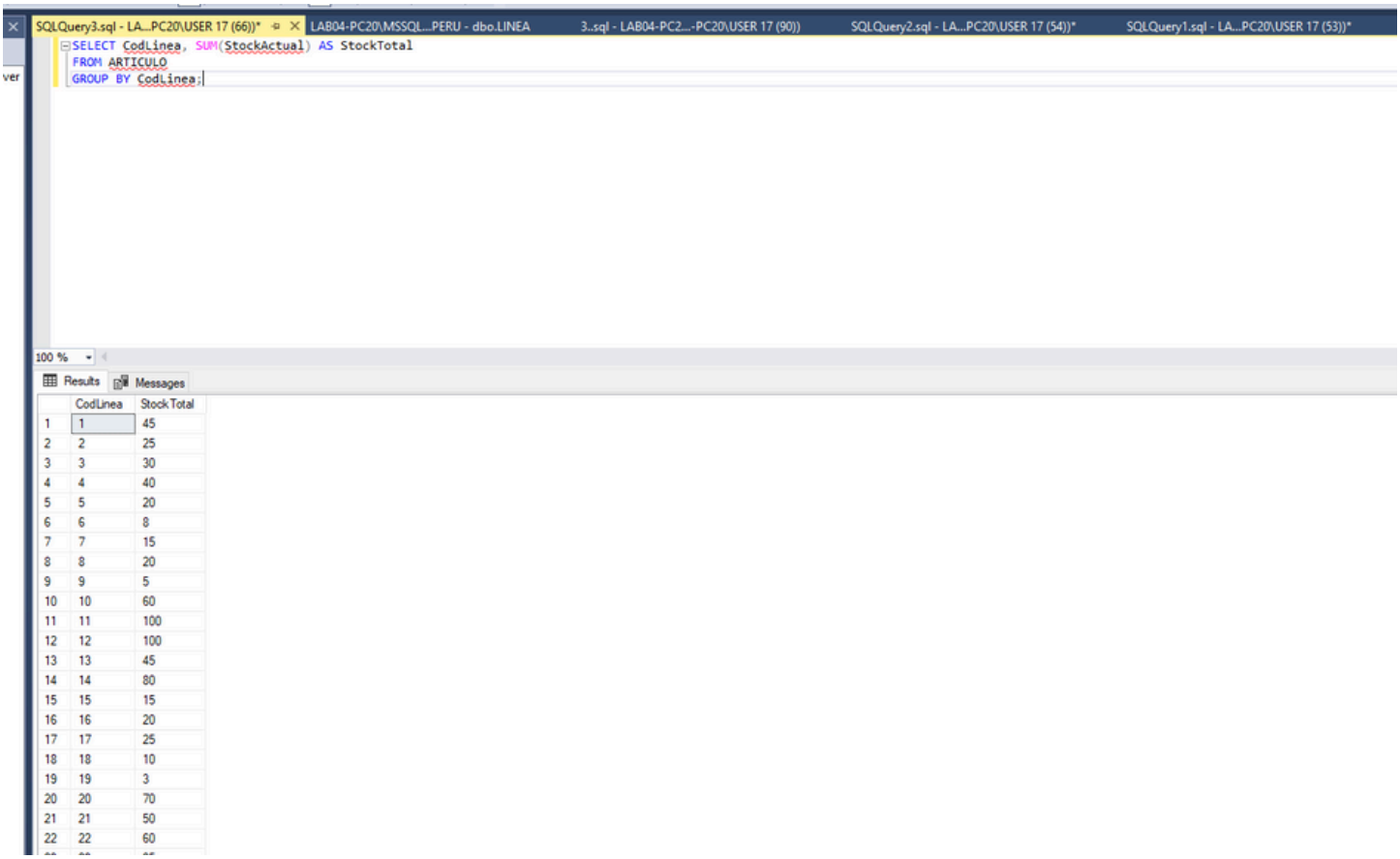
Results | Messages

	NomLinea	CantArticulos
1	Access Points	1
2	Adaptadores	1
3	AllInOne	1
4	Antivirus	1
5	Auriculares	1
6	Cableado	1
7	Cables	1
8	Cámaras IP	1
9	Chromebooks	1
10	Conectores	1
11	Discos Duros	1
12	Discos Externos	1
13	Enfriamiento	1
14	Estabilizadores	1
15	Fuentes Poder	1
16	Gabinetes	1
17	Gaming	1
18	Impresoras	1
19	Laptops	1

12. Mostrar CodLinea y StockTotal.

SQL

```
SELECT CodLinea, SUM(StockActual) AS StockTotal

FROM ARTICULO

GROUP BY CodLinea;
```

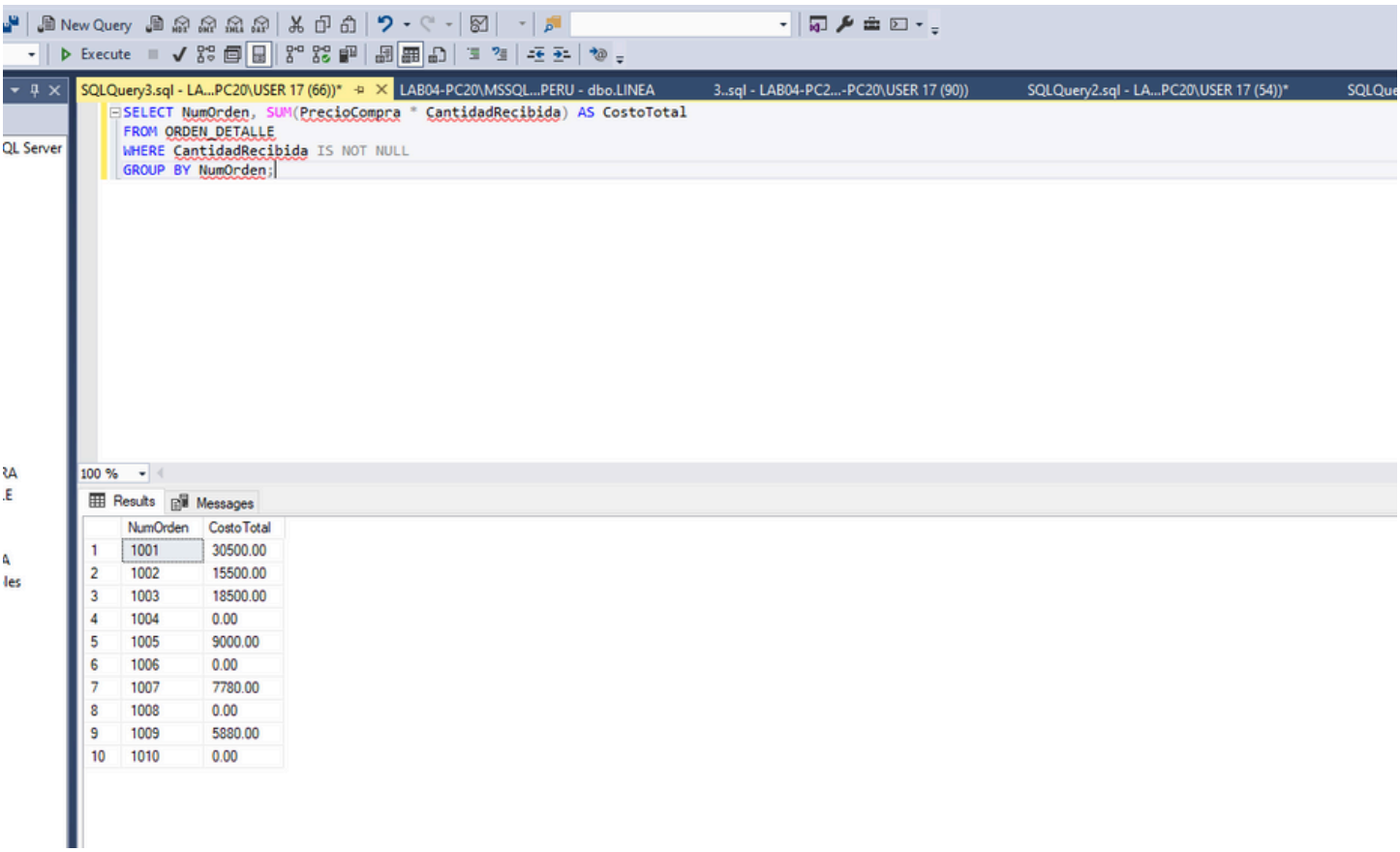


	CodLinea	StockTotal
1	1	45
2	2	25
3	3	30
4	4	40
5	5	20
6	6	8
7	7	15
8	8	20
9	9	5
10	10	60
11	11	100
12	12	100
13	13	45
14	14	80
15	15	15
16	16	20
17	17	25
18	18	10
19	19	3
20	20	70
21	21	50
22	22	60

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

SQL

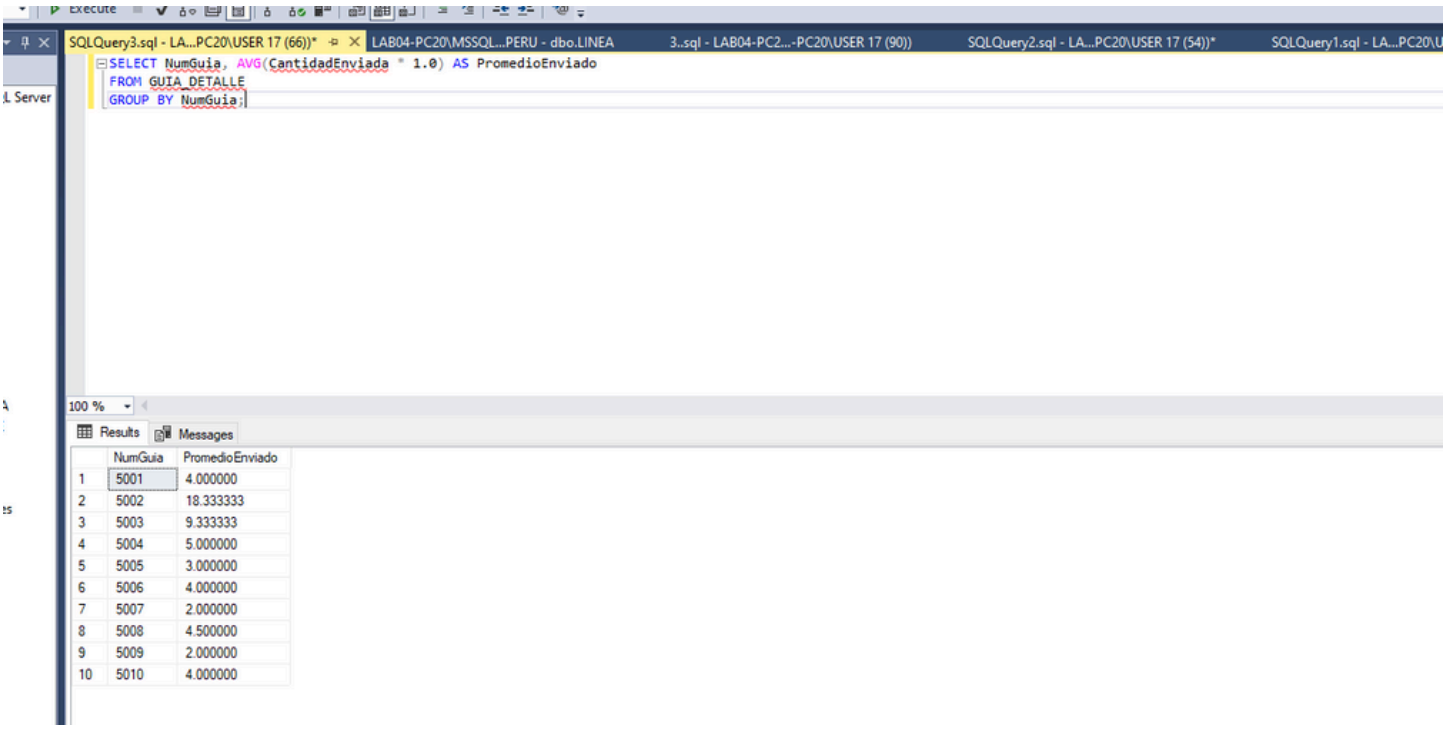
```
SELECT NumOrden, SUM(PrecioCompra * CantidadRecibida) AS CostoTotal

FROM ORDEN_DETALLE

WHERE CantidadRecibida IS NOT NULL

GROUP BY NumOrden;
```



14. Mostrar NumGuia y PromedioEnviado (Cantidad).

SQL

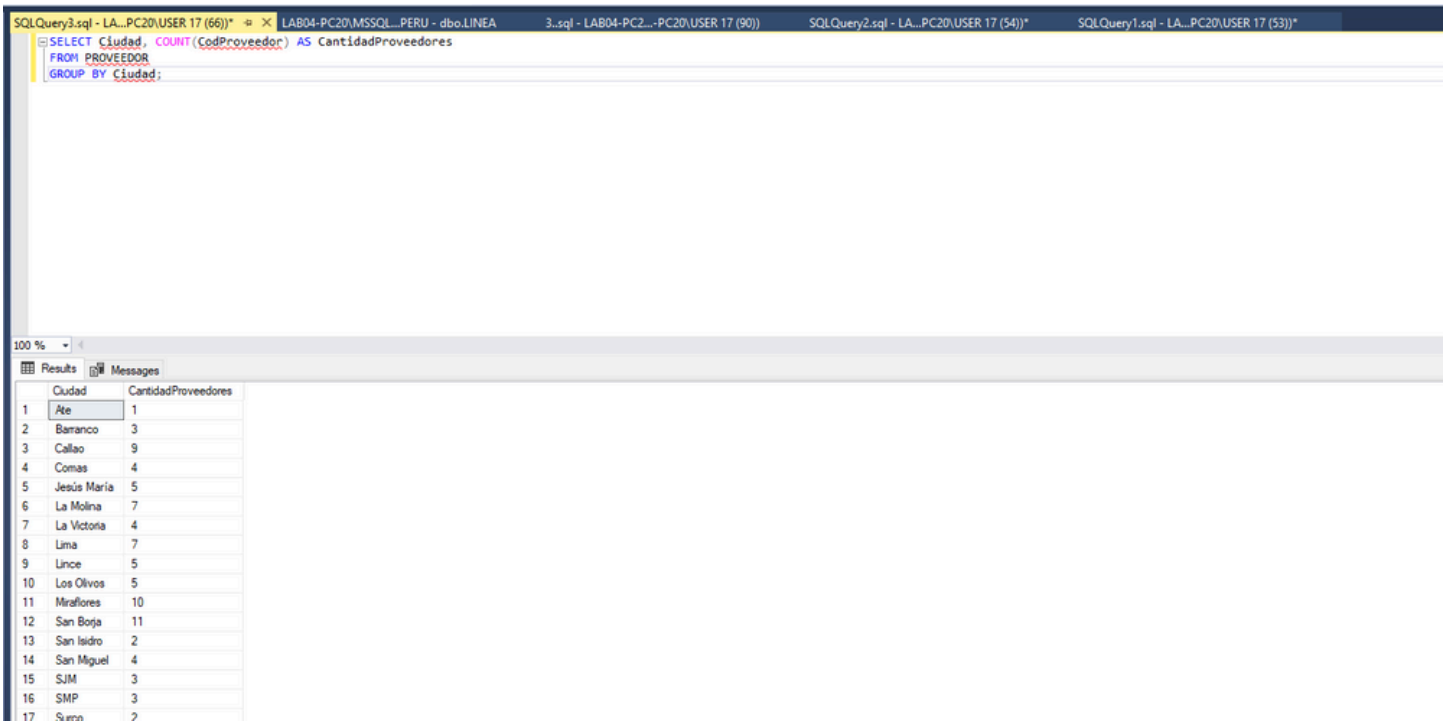
```
SELECT NumGuia, AVG(CantidadEnviada * 1.0) AS PromedioEnviado
FROM GUIA_DETALLE
GROUP BY NumGuia;
```



15. Contar proveedores agrupados por Ciudad.

SQL

```
SELECT Ciudad, COUNT(CodProveedor) AS CantidadProveedores
FROM PROVEEDOR
GROUP BY Ciudad;
```

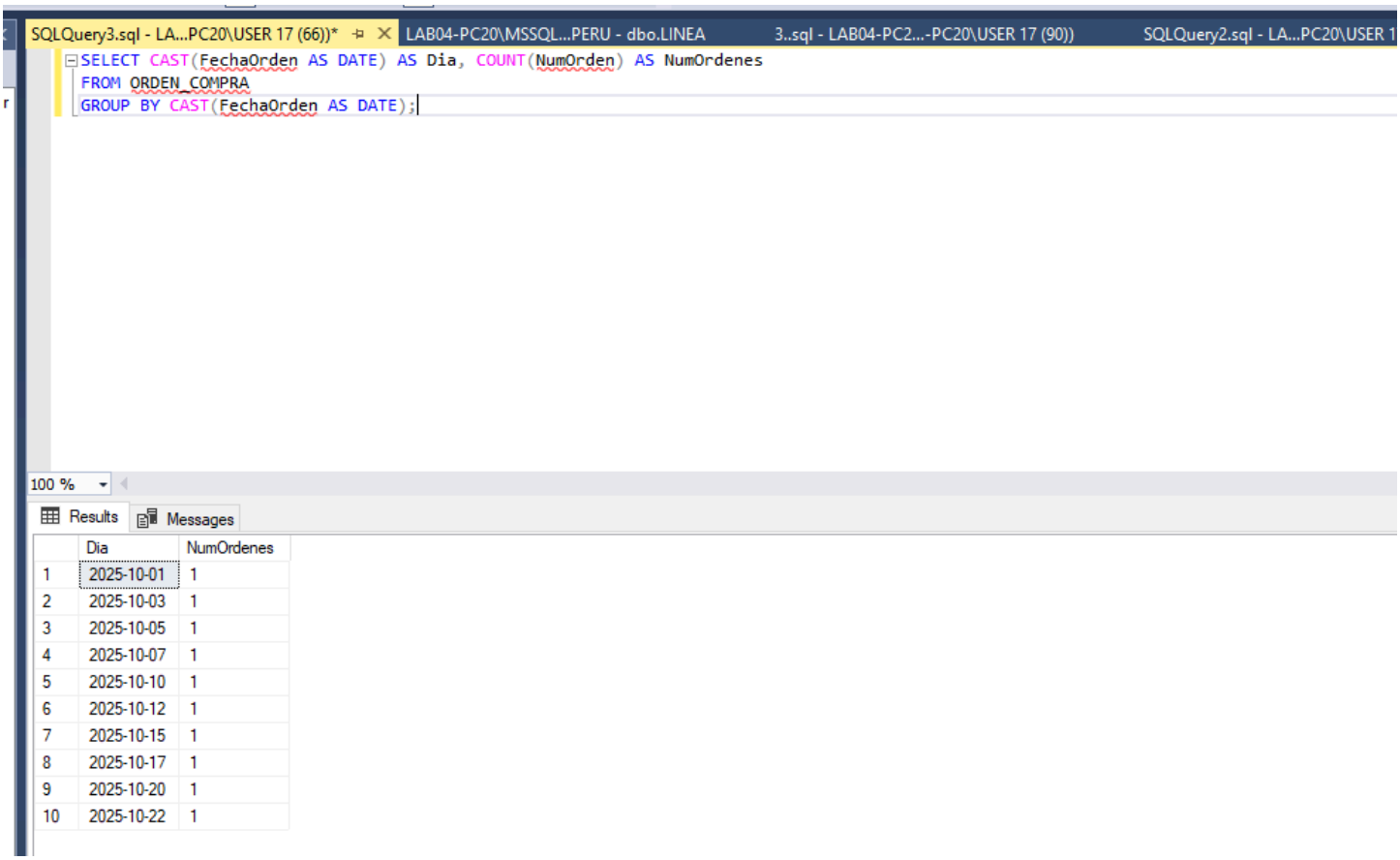


	Ciudad	CantidadProveedores
1	Ate	1
2	Barranco	3
3	Callao	9
4	Comas	4
5	Jesús María	5
6	La Molina	7
7	La Victoria	4
8	Lima	7
9	Lince	5
10	Los Olivos	5
11	Miraflores	10
12	San Borja	11
13	San Isidro	2
14	San Miguel	4
15	SJM	3
16	SMP	3
17	Surco	2

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

SQL

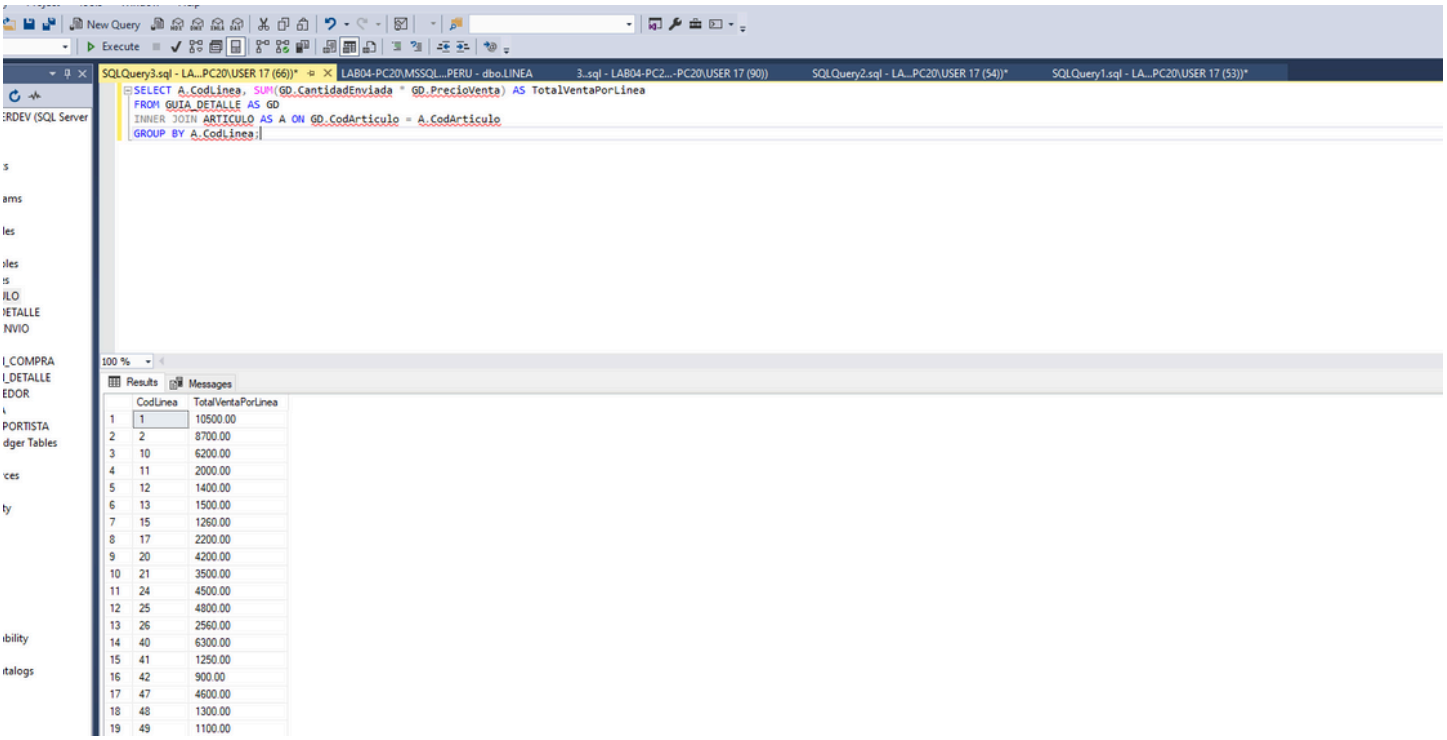
```
SELECT CAST(FechaOrden AS DATE) AS Dia, COUNT(NumOrden) AS NumOrdenes
FROM ORDEN_COMPRA
GROUP BY CAST(FechaOrden AS DATE);
```



17. Sumar (CantidadEnviada * PrecioVenta) por CodLinea.

SQL

```
SELECT A.CodLinea, SUM(GD.CantidadEnviada * GD.PrecioVenta) AS
TotalVentaPorLinea

FROM GUIA_DETALLE AS GD

INNER JOIN ARTICULO AS A ON GD.CodArticulo = A.CodArticulo

GROUP BY A.CodLinea;
```

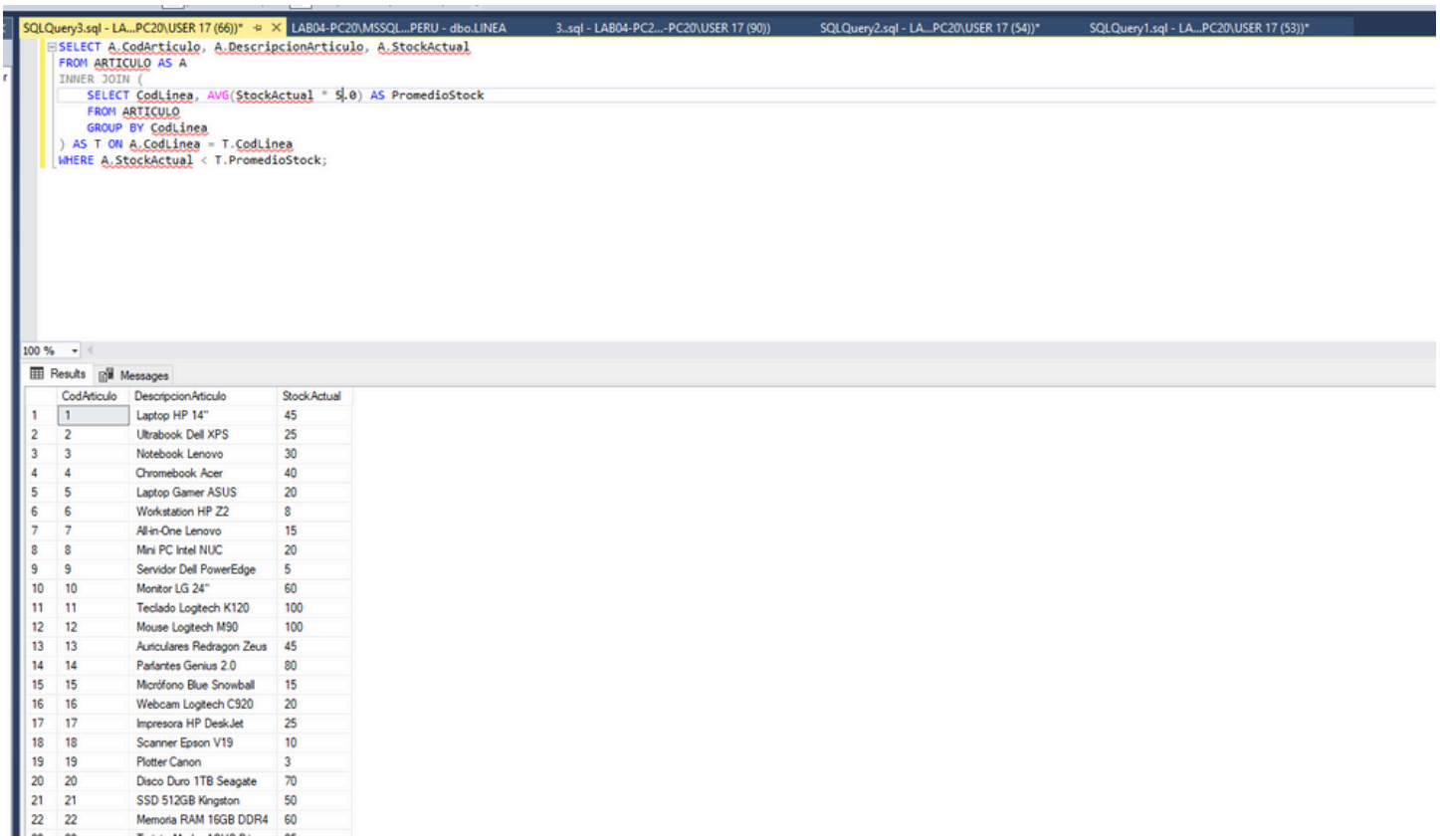


	CodLinea	TotalVentaPorLinea
1	1	10500.00
2	2	8700.00
3	10	6200.00
4	11	2000.00
5	12	1400.00
6	13	1500.00
7	15	1260.00
8	17	2200.00
9	20	4200.00
10	21	3500.00
11	24	4500.00
12	25	4800.00
13	26	2560.00
14	40	6300.00
15	41	1250.00
16	42	900.00
17	47	4600.00
18	48	1300.00
19	49	1100.00

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

SQL

```
SELECT A.CodArticulo, A.DescripcionArticulo, A.StockActual
FROM ARTICULO AS A
INNER JOIN (
    SELECT CodLinea, AVG(StockActual * 1.0) AS PromedioStock
    FROM ARTICULO
    GROUP BY CodLinea
) AS T ON A.CodLinea = T.CodLinea
WHERE A.StockActual < T.PromedioStock;
```



```
SELECT A.CodArticulo, A.DescripcionArticulo, A.StockActual
FROM ARTICULO AS A
INNER JOIN (
    SELECT CodLinea, AVG(StockActual * 5.0) AS PromedioStock
    FROM ARTICULO
    GROUP BY CodLinea
) AS T ON A.CodLinea = T.CodLinea
WHERE A.StockActual < T.PromedioStock;
```

100 %

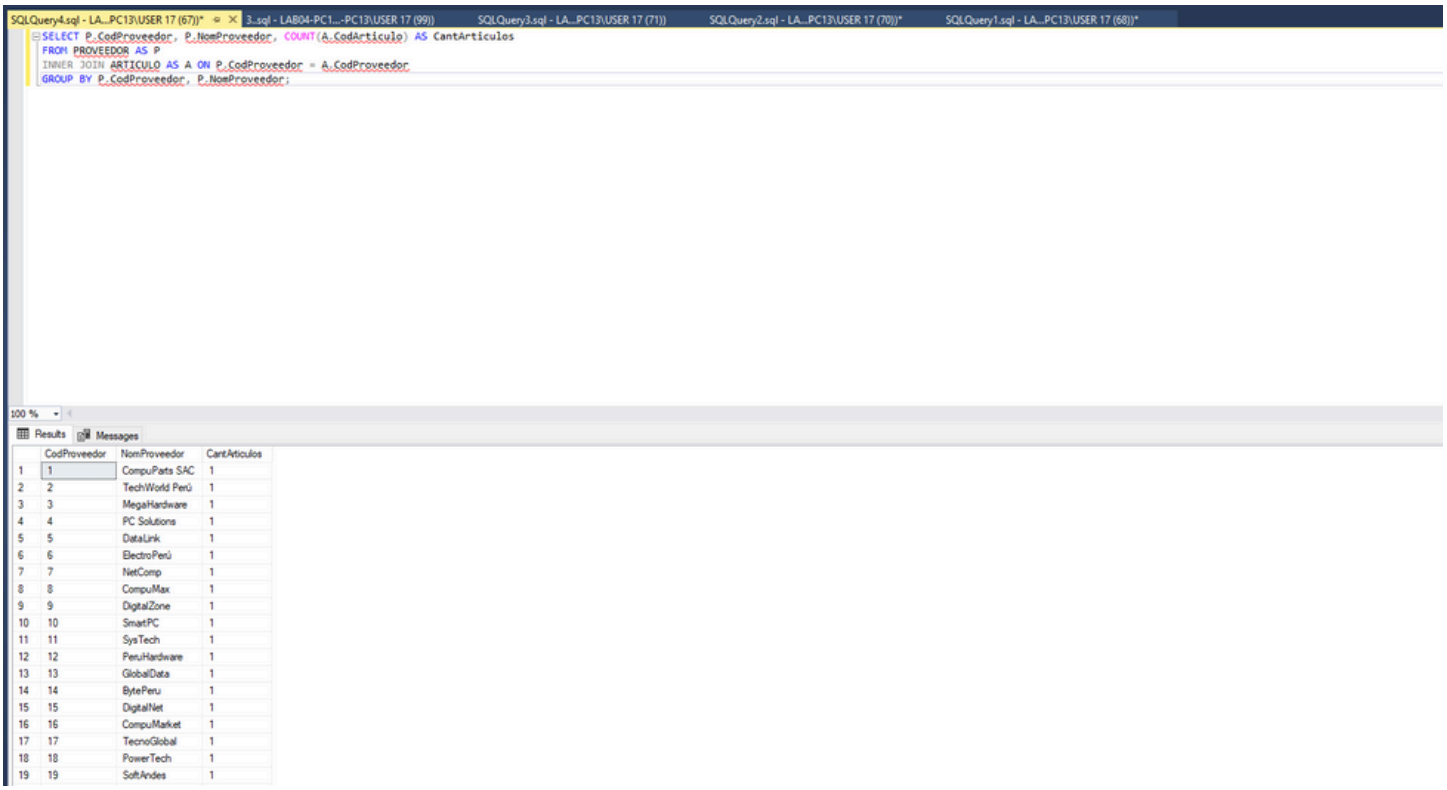
Results | Messages

	CodArticulo	DescripcionArticulo	StockActual
1	1	Laptop HP 14"	45
2	2	Ultrabook Dell XPS	25
3	3	Notebook Lenovo	30
4	4	Chromebook Acer	40
5	5	Laptop Gamer ASUS	20
6	6	Workstation HP Z2	8
7	7	All-in-One Lenovo	15
8	8	Mini PC Intel NUC	20
9	9	Servidor Dell PowerEdge	5
10	10	Monitor LG 24"	60
11	11	Teclado Logitech K120	100
12	12	Mouse Logitech M90	100
13	13	Auriculares Redragon Zeus	45
14	14	Parlantes Genius 2.0	80
15	15	Micrófono Blue Snowball	15
16	16	Webcam Logitech C920	20
17	17	Impresora HP DeskJet	25
18	18	Scanner Epson V19	10
19	19	Plotter Canon	3
20	20	Disco Duro 1TB Seagate	70
21	21	SSD 512GB Kingston	50
22	22	Memoria RAM 16GB DDR4	60

19. Mostrar CodProveedor, NomProveedor y CantArticulos.

SQL

```
SELECT P.CodProveedor, P.NomProveedor, COUNT(A.CodArticulo) AS
CantArticulos

FROM PROVEEDOR AS P

INNER JOIN ARTICULO AS A ON P.CodProveedor = A.CodProveedor

GROUP BY P.CodProveedor, P.NomProveedor;
```



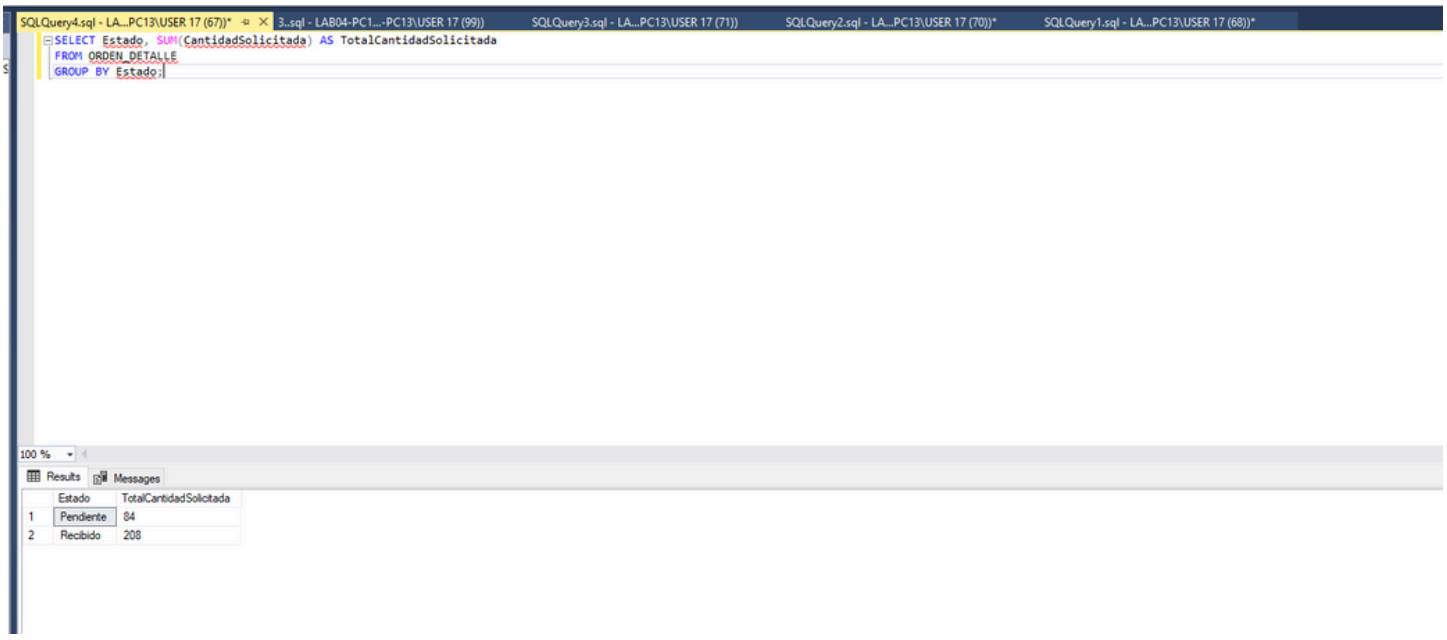
```
SELECT P.CodProveedor, P.NomProveedor, COUNT(A.CodArticulo) AS CantArticulos
FROM PROVEEDOR AS P
INNER JOIN ARTICULO AS A ON P.CodProveedor = A.CodProveedor
GROUP BY P.CodProveedor, P.NomProveedor;
```

	CodProveedor	NomProveedor	CantArticulos
1	1	CompuParts SAC	1
2	2	TechWorld Perú	1
3	3	MegaHardware	1
4	4	PC Solutions	1
5	5	DataLink	1
6	6	ElectroPerú	1
7	7	NetComp	1
8	8	CompuMax	1
9	9	DigitalZone	1
10	10	SmartPC	1
11	11	SysTech	1
12	12	PeruHardware	1
13	13	GlobalData	1
14	14	BytePeru	1
15	15	DigitalNet	1
16	16	CompuMarket	1
17	17	TecnoGlobal	1
18	18	PowerTech	1
19	19	SoftAndes	1

20. Mostrar para cada estado sumar CantidadSolicitada.

SQL

```
SELECT Estado, SUM(CantidadSolicitada) AS TotalCantidadSolicitada
FROM ORDEN_DETALLE
GROUP BY Estado;
```



```
SELECT Estado, SUM(CantidadSolicitada) AS TotalCantidadSolicitada
FROM ORDEN_DETALLE
GROUP BY Estado;
```

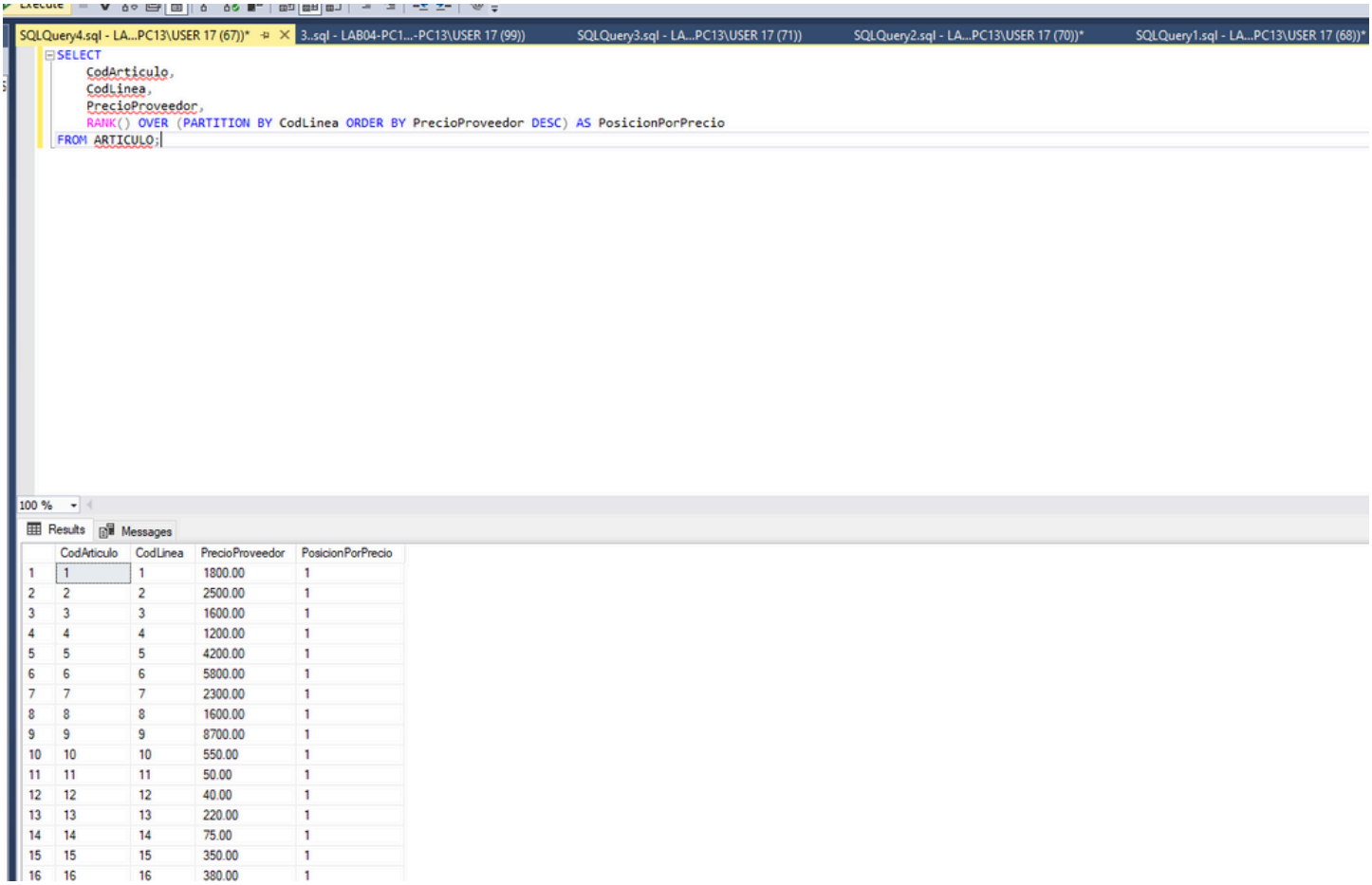
	Estado	TotalCantidadSolicitada
1	Pendiente	84
2	Recibido	208

Ejercicios 21 - 30 (Consultas con OVER / Funciones de Ventana)

21. Asignar posición por línea ordenada por precio.

SQL

```
SELECT
    CodArticulo,
    CodLinea,
    PrecioProveedor,
    RANK() OVER (PARTITION BY CodLinea ORDER BY PrecioProveedor DESC) AS PosicionPorPrecio
FROM ARTICULO;
```



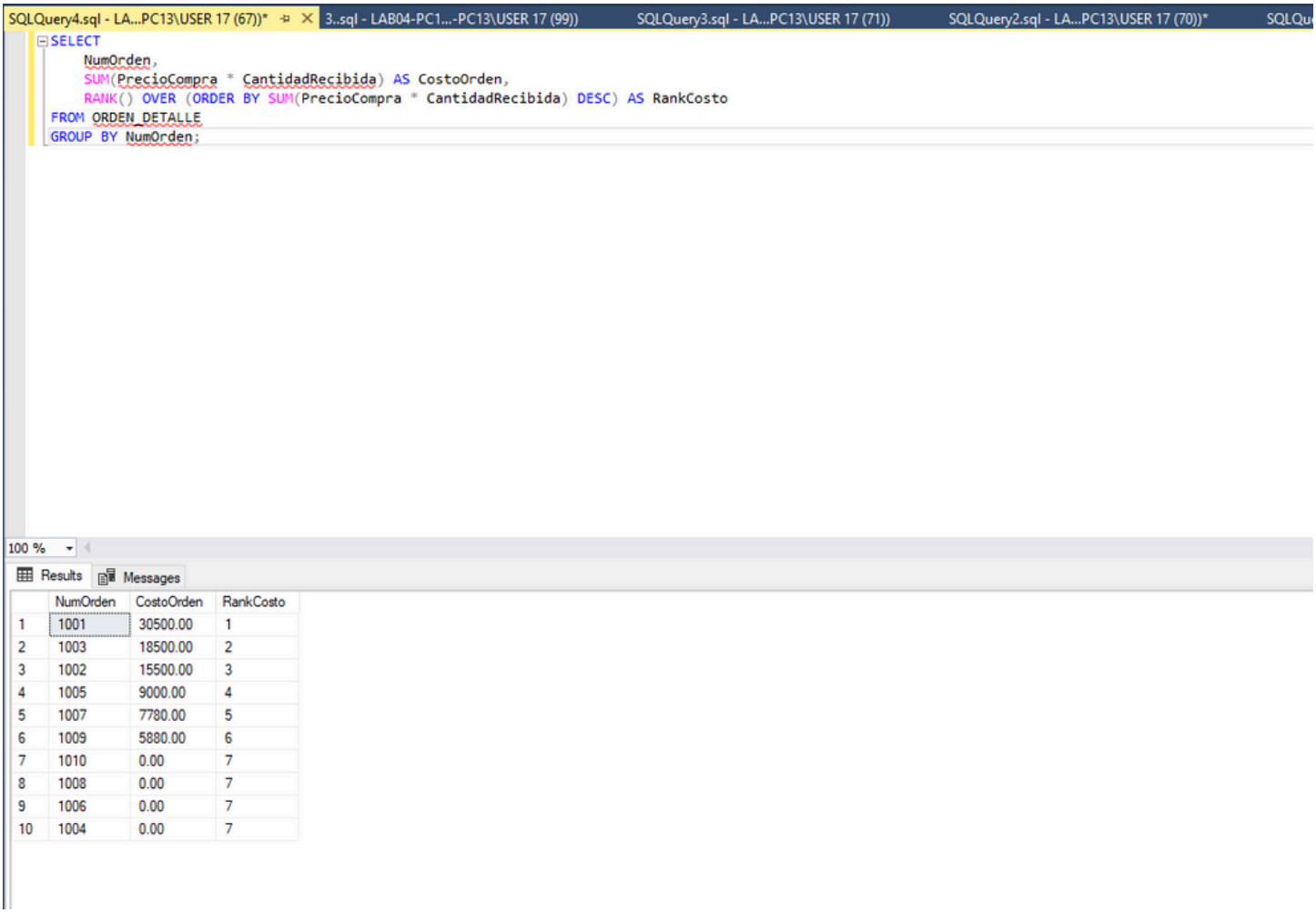
	CodArticulo	CodLinea	PrecioProveedor	PosicionPorPrecio
1	1	1	1800.00	1
2	2	2	2500.00	1
3	3	3	1600.00	1
4	4	4	1200.00	1
5	5	5	4200.00	1
6	6	6	5800.00	1
7	7	7	2300.00	1
8	8	8	1600.00	1
9	9	9	8700.00	1
10	10	10	550.00	1
11	11	11	50.00	1
12	12	12	40.00	1
13	13	13	220.00	1
14	14	14	75.00	1
15	15	15	350.00	1
16	16	16	380.00	1

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

SQL

```
SELECT
```

```
    NumOrden,

    SUM(PrecioCompra * CantidadRecibida) AS CostoOrden,

    RANK() OVER (ORDER BY SUM(PrecioCompra * CantidadRecibida) DESC) AS
RankCosto

FROM ORDEN_DETALLE

GROUP BY NumOrden;
```



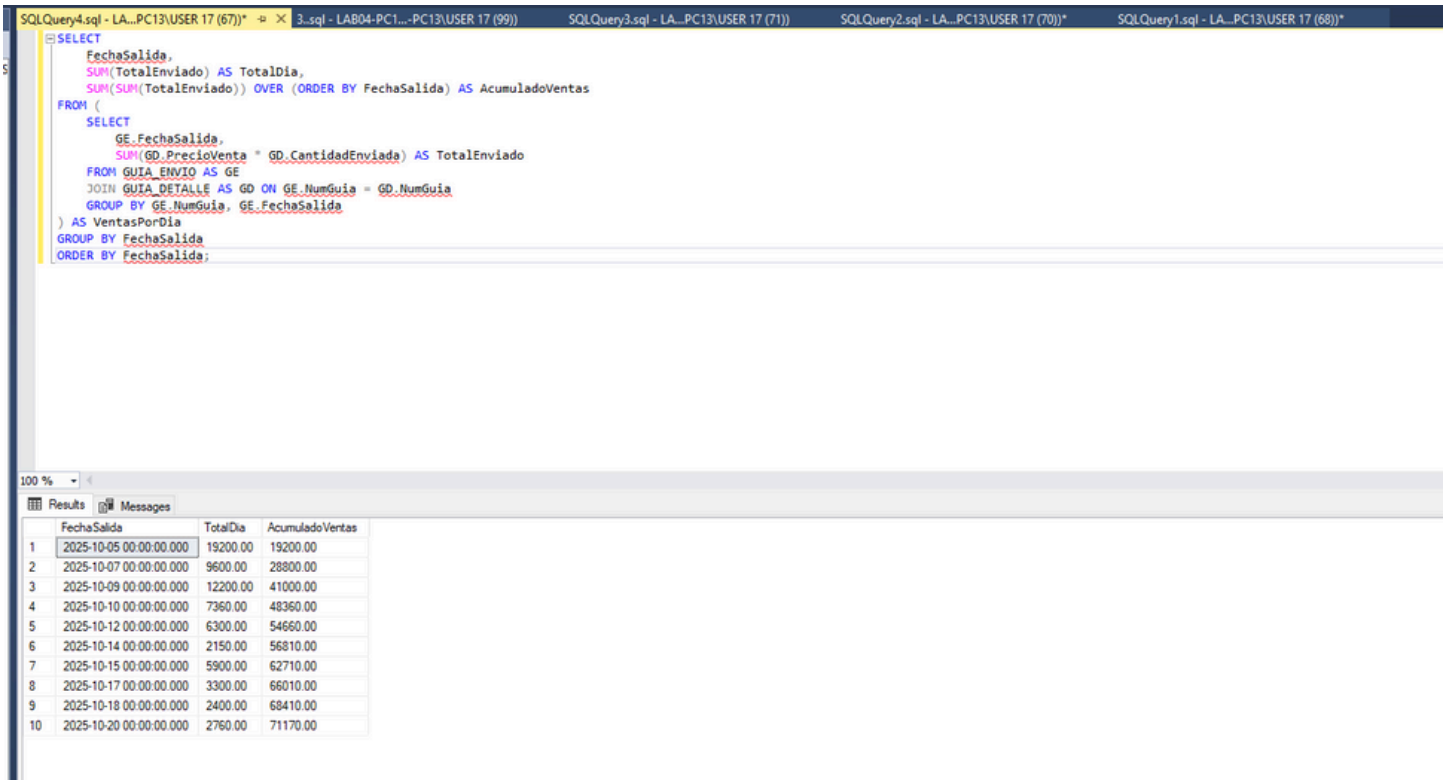
	NumOrden	CostoOrden	RankCosto
1	1001	30500.00	1
2	1003	18500.00	2
3	1002	15500.00	3
4	1005	9000.00	4
5	1007	7780.00	5
6	1009	5880.00	6
7	1010	0.00	7
8	1008	0.00	7
9	1006	0.00	7
10	1004	0.00	7

23. Mostrar TotalDia y AcumuladoVentas ordenado por fecha.

SQL

```
SELECT

    FechaSalida,

    SUM(TotalEnviado) AS TotalDia,

    SUM(SUM(TotalEnviado)) OVER (ORDER BY FechaSalida) AS AcumuladoVentas

FROM (
```

```
    SELECT
        GE.FechaSalida,
        SUM(GD.PrecioVenta * GD.CantidadEnviada) AS TotalEnviado
    FROM GUIA_ENVIO AS GE
    JOIN GUIA_DETALLE AS GD ON GE.NumGuia = GD.NumGuia
    GROUP BY GE.NumGuia, GE.FechaSalida
) AS VentasPorDia
GROUP BY FechaSalida
ORDER BY FechaSalida;
```



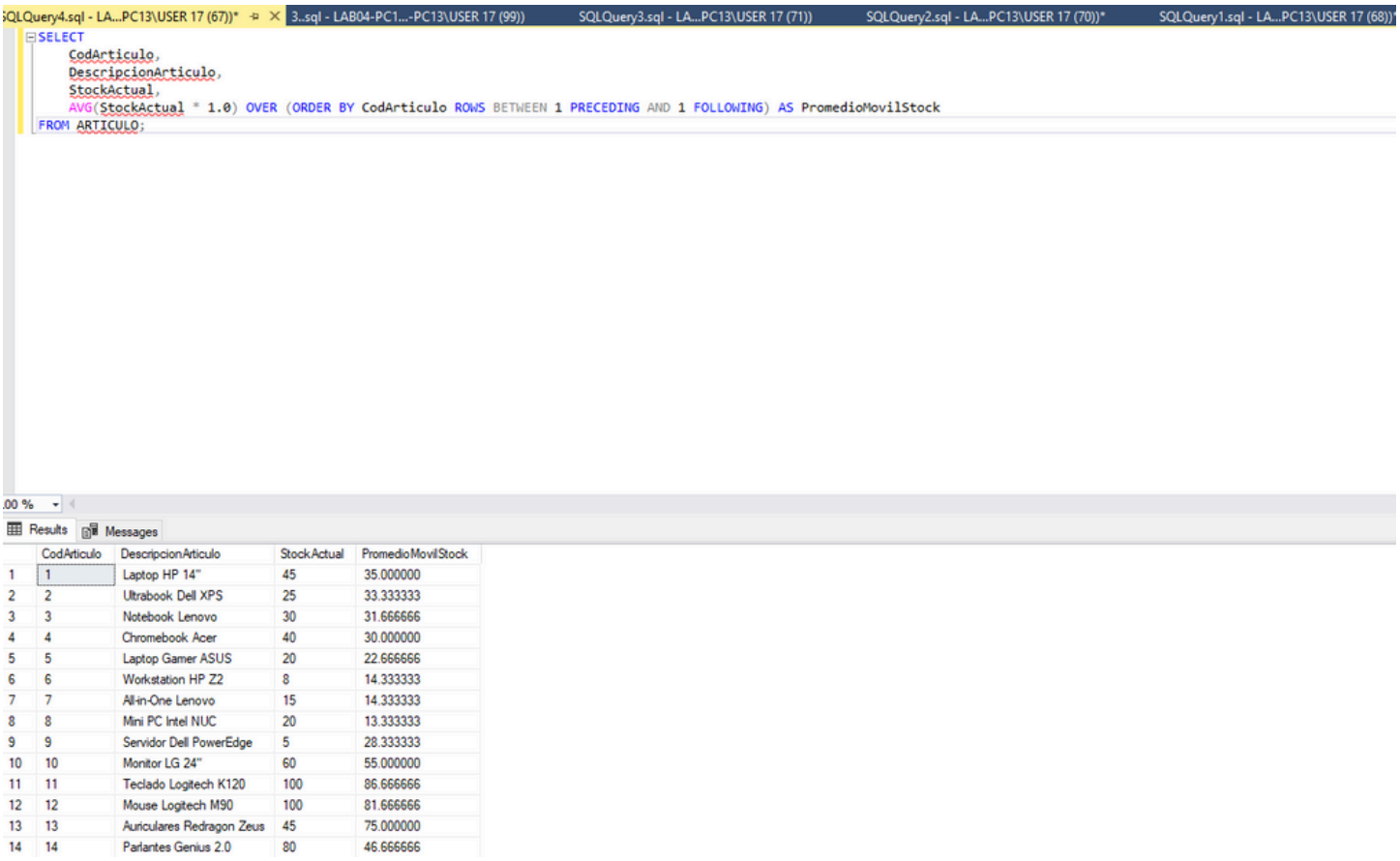
	FechaSalida	TotalDia	AcumuladoVentas
1	2025-10-05 00:00:00.000	19200.00	19200.00
2	2025-10-07 00:00:00.000	9600.00	28800.00
3	2025-10-09 00:00:00.000	12200.00	41000.00
4	2025-10-10 00:00:00.000	7360.00	48360.00
5	2025-10-12 00:00:00.000	6300.00	54660.00
6	2025-10-14 00:00:00.000	2150.00	56810.00
7	2025-10-15 00:00:00.000	5900.00	62710.00
8	2025-10-17 00:00:00.000	3300.00	66010.00
9	2025-10-18 00:00:00.000	2400.00	68410.00
10	2025-10-20 00:00:00.000	2760.00	71170.00

24. Calcular promedio móvil para stock.

SQL

```
SELECT
    CodArticulo,
    DescripcionArticulo,
    StockActual,
```

```
    AVG(StockActual * 1.0) OVER (ORDER BY CodArticulo ROWS BETWEEN 1
PRECEDING AND 1 FOLLOWING) AS PromedioMovilStock

FROM ARTICULO;
```



```
SELECT
    CodArticulo,
    DescripcionArticulo,
    StockActual,
    AVG(StockActual * 1.0) OVER (ORDER BY CodArticulo ROWS BETWEEN 1 PRECEDING AND 1 FOLLOWING) AS PromedioMovilStock
FROM ARTICULO;
```

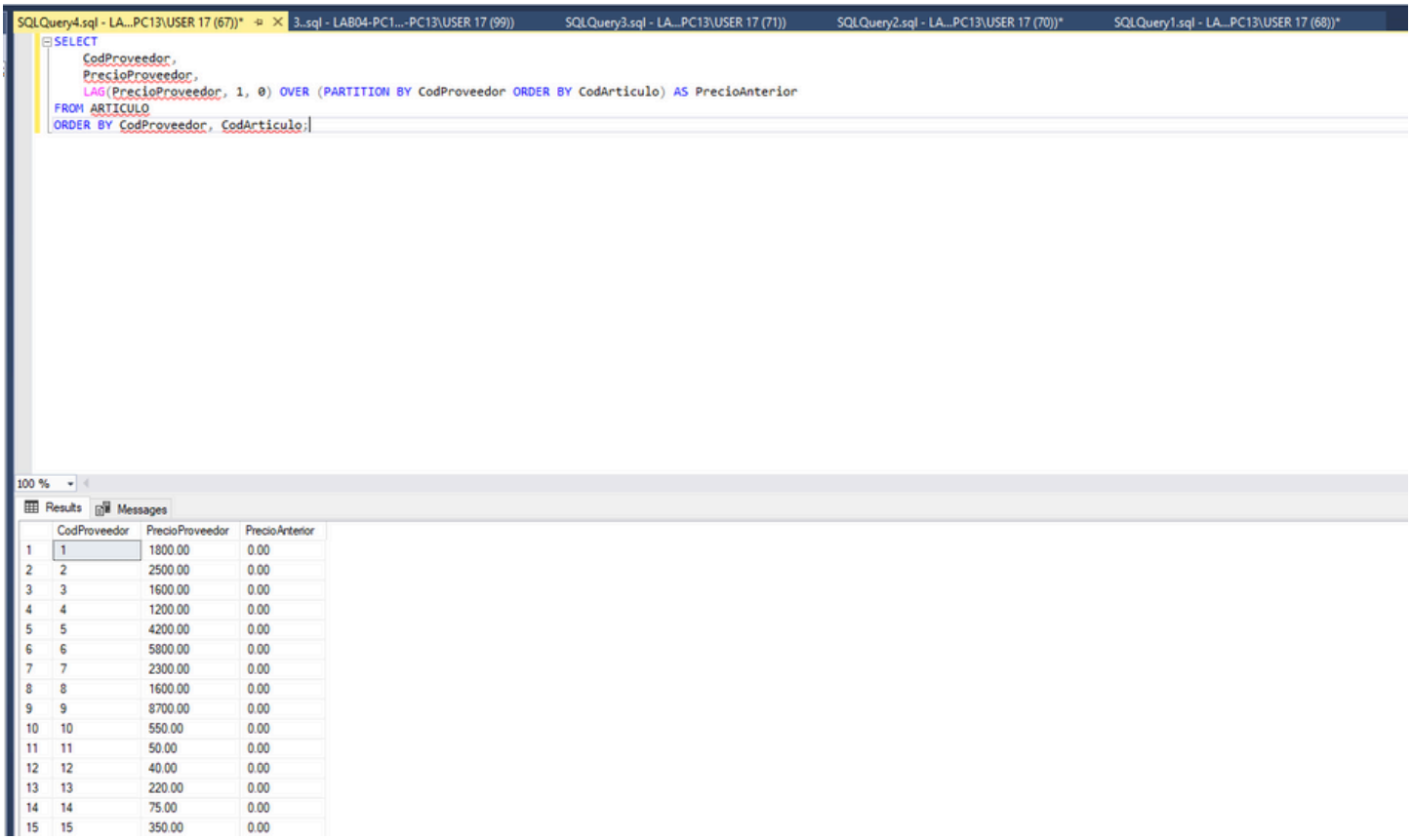
Results | Messages

	CodArticulo	DescripcionArticulo	StockActual	PromedioMovilStock
1	1	Laptop HP 14"	45	35.000000
2	2	Ultrabook Dell XPS	25	33.333333
3	3	Notebook Lenovo	30	31.666666
4	4	Chromebook Acer	40	30.000000
5	5	Laptop Gamer ASUS	20	22.666666
6	6	Workstation HP Z2	8	14.333333
7	7	All-in-One Lenovo	15	14.333333
8	8	Mini PC Intel NUC	20	13.333333
9	9	Servidor Dell PowerEdge	5	28.333333
10	10	Monitor LG 24"	60	55.000000
11	11	Teclado Logitech K120	100	86.666666
12	12	Mouse Logitech M90	100	81.666666
13	13	Auriculares Redragon Zeus	45	75.000000
14	14	Parlantes Genius 2.0	80	46.666666

25. Mostrar PrecioAnteriorMaximoProveedor usando LAG.

SQL

```
SELECT

    CodProveedor,

    PrecioProveedor,

    LAG(PrecioProveedor, 1, 0) OVER (PARTITION BY CodProveedor ORDER BY
CodArticulo) AS PrecioAnterior

FROM ARTICULO

ORDER BY CodProveedor, CodArticulo;
```

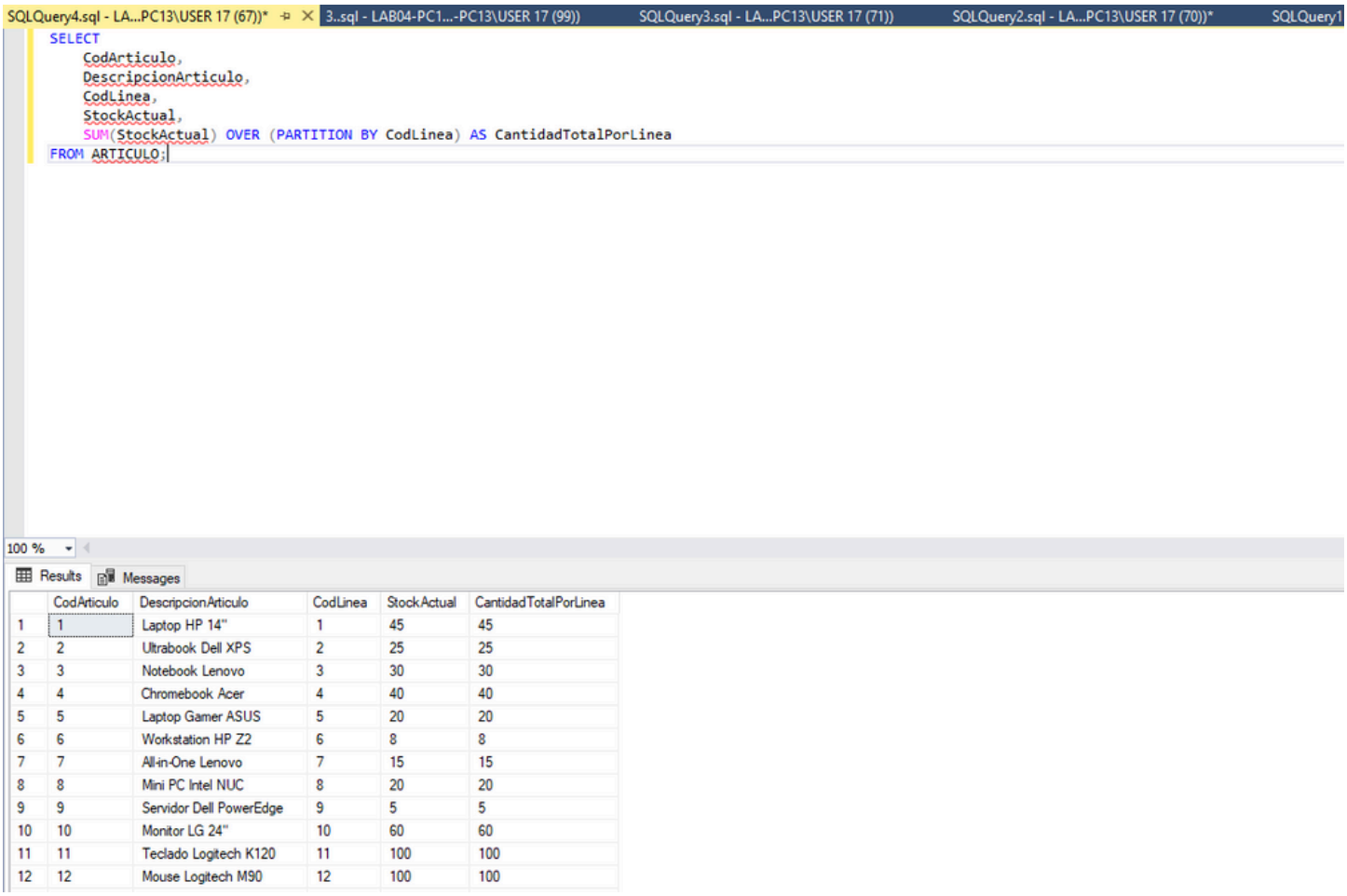


	CodProveedor	PrecioProveedor	PrecioAnterior
1	1	1800.00	0.00
2	2	2500.00	0.00
3	3	1600.00	0.00
4	4	1200.00	0.00
5	5	4200.00	0.00
6	6	5800.00	0.00
7	7	2300.00	0.00
8	8	1600.00	0.00
9	9	8700.00	0.00
10	10	550.00	0.00
11	11	50.00	0.00
12	12	40.00	0.00
13	13	220.00	0.00
14	14	75.00	0.00
15	15	350.00	0.00

26. Añadir columna CantidadPorLinea a cada artículo.

SQL

```
SELECT
    CodArticulo,
    DescripcionArticulo,
    CodLinea,
    StockActual,
    SUM(StockActual) OVER (PARTITION BY CodLinea) AS CantidadTotalPorLinea
FROM ARTICULO;
```

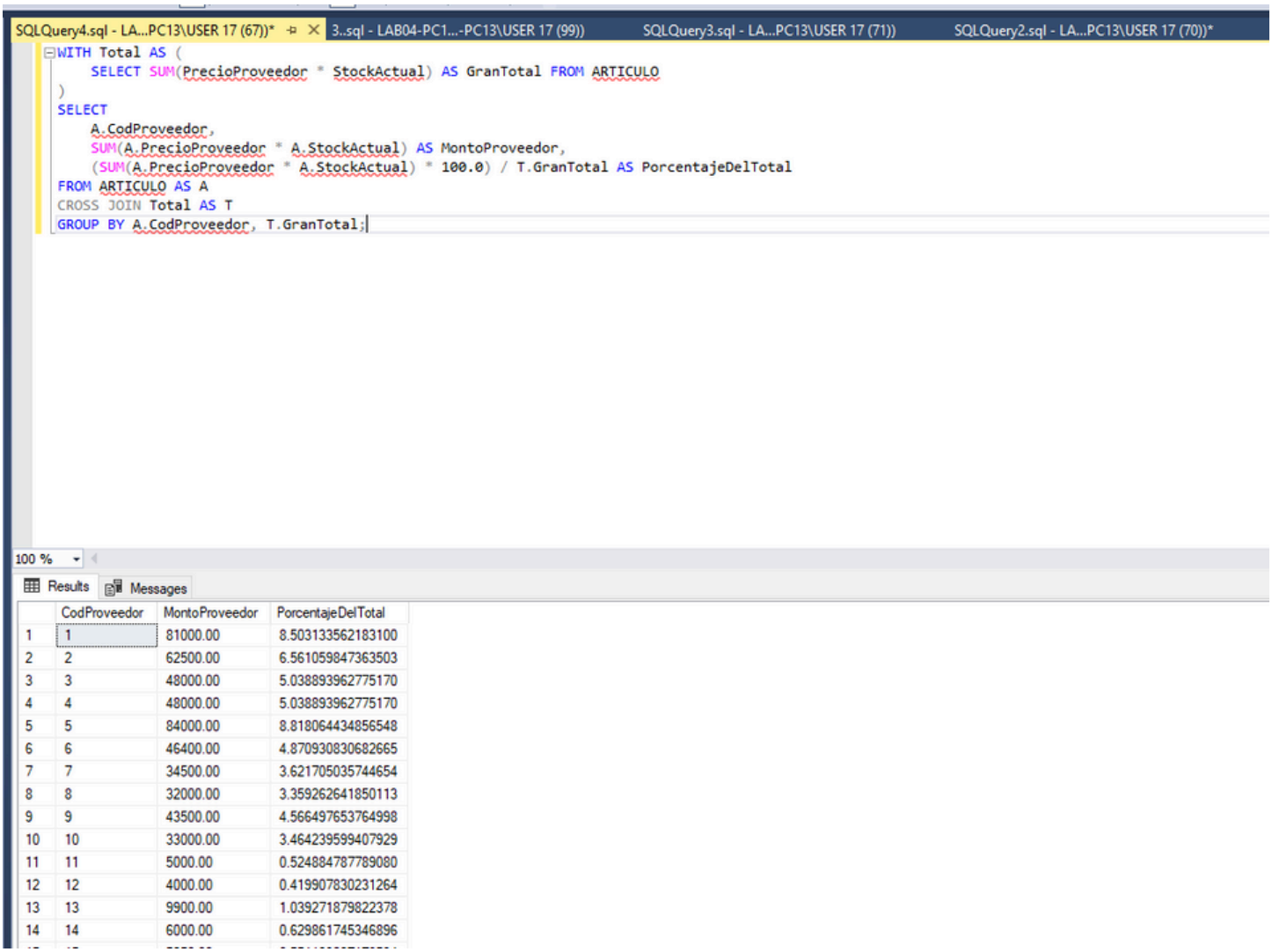


	CodArticulo	DescripcionArticulo	CodLinea	StockActual	CantidadTotalPorLinea
1	1	Laptop HP 14"	1	45	45
2	2	Ultrabook Dell XPS	2	25	25
3	3	Notebook Lenovo	3	30	30
4	4	Chromebook Acer	4	40	40
5	5	Laptop Gamer ASUS	5	20	20
6	6	Workstation HP Z2	6	8	8
7	7	All-in-One Lenovo	7	15	15
8	8	Mini PC Intel NUC	8	20	20
9	9	Servidor Dell PowerEdge	9	5	5
10	10	Monitor LG 24"	10	60	60
11	11	Teclado Logitech K120	11	100	100
12	12	Mouse Logitech M90	12	100	100

27. Mostrar MontoProveedor y PorcentajeDelTotal.

SQL

```
WITH Total AS (
    SELECT SUM(PrecioProveedor * StockActual) AS GranTotal FROM ARTICULO
)
SELECT
    A.CodProveedor,
    SUM(A.PrecioProveedor * A.StockActual) AS MontoProveedor,
    (SUM(A.PrecioProveedor * A.StockActual) * 100.0) / T.GranTotal AS
PorcentajeDelTotal
FROM ARTICULO AS A
CROSS JOIN Total AS T
GROUP BY A.CodProveedor, T.GranTotal;
```

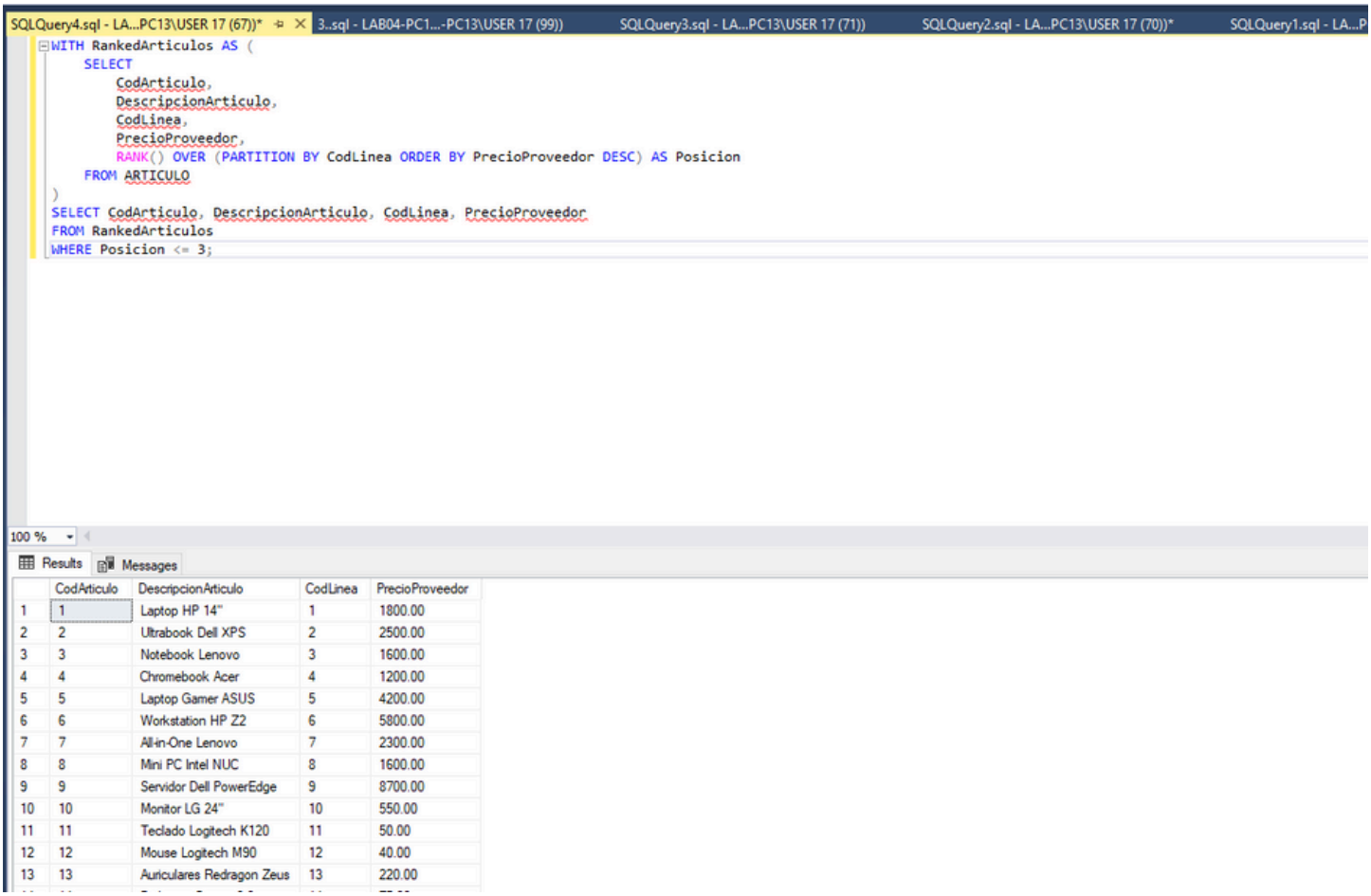


28. Mostrar solo los 3 artículos más caros por línea.

SQL

```
WITH RankedArticulos AS (
    SELECT
        CodArticulo,
        DescripcionArticulo,
        CodLinea,
        PrecioProveedor,
        RANK() OVER (PARTITION BY CodLinea ORDER BY PrecioProveedor DESC)
AS Posicion
    FROM ARTICULO
)
```

```
SELECT CodArticulo, DescripcionArticulo, CodLinea, PrecioProveedor
FROM RankedArticulos
WHERE Posicion <= 3;
```

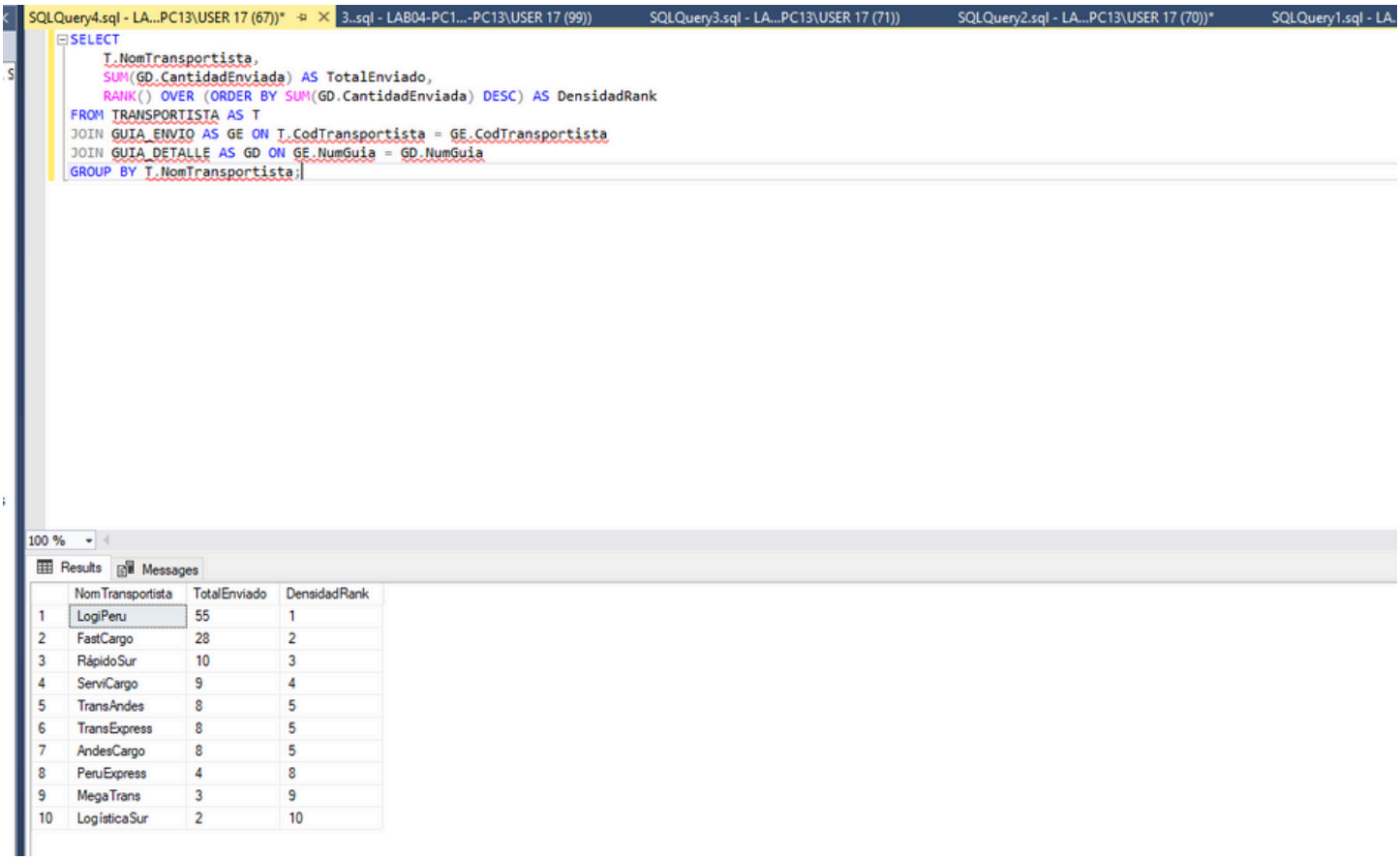


	CodArticulo	DescripcionArticulo	CodLinea	PrecioProveedor
1	1	Laptop HP 14"	1	1800.00
2	2	Ultrabook Dell XPS	2	2500.00
3	3	Notebook Lenovo	3	1600.00
4	4	Chromebook Acer	4	1200.00
5	5	Laptop Gamer ASUS	5	4200.00
6	6	Workstation HP Z2	6	5800.00
7	7	All-in-One Lenovo	7	2300.00
8	8	Mini PC Intel NUC	8	1600.00
9	9	Servidor Dell PowerEdge	9	8700.00
10	10	Monitor LG 24"	10	550.00
11	11	Teclado Logitech K120	11	50.00
12	12	Mouse Logitech M90	12	40.00
13	13	Auriculares Redragon Zeus	13	220.00

29. Mostrar transportista y su DensidadRank por TotalEnviado.

SQL

```
SELECT
    T.NomTransportista,
    SUM(GD.CantidadEnviada) AS TotalEnviado,
    RANK() OVER (ORDER BY SUM(GD.CantidadEnviada) DESC) AS DensidadRank
FROM TRANSPORTISTA AS T
JOIN GUIA_ENVIO AS GE ON T.CodTransportista = GE.CodTransportista
JOIN GUIA_DETALLE AS GD ON GE.NumGuia = GD.NumGuia
GROUP BY T.NomTransportista;
```



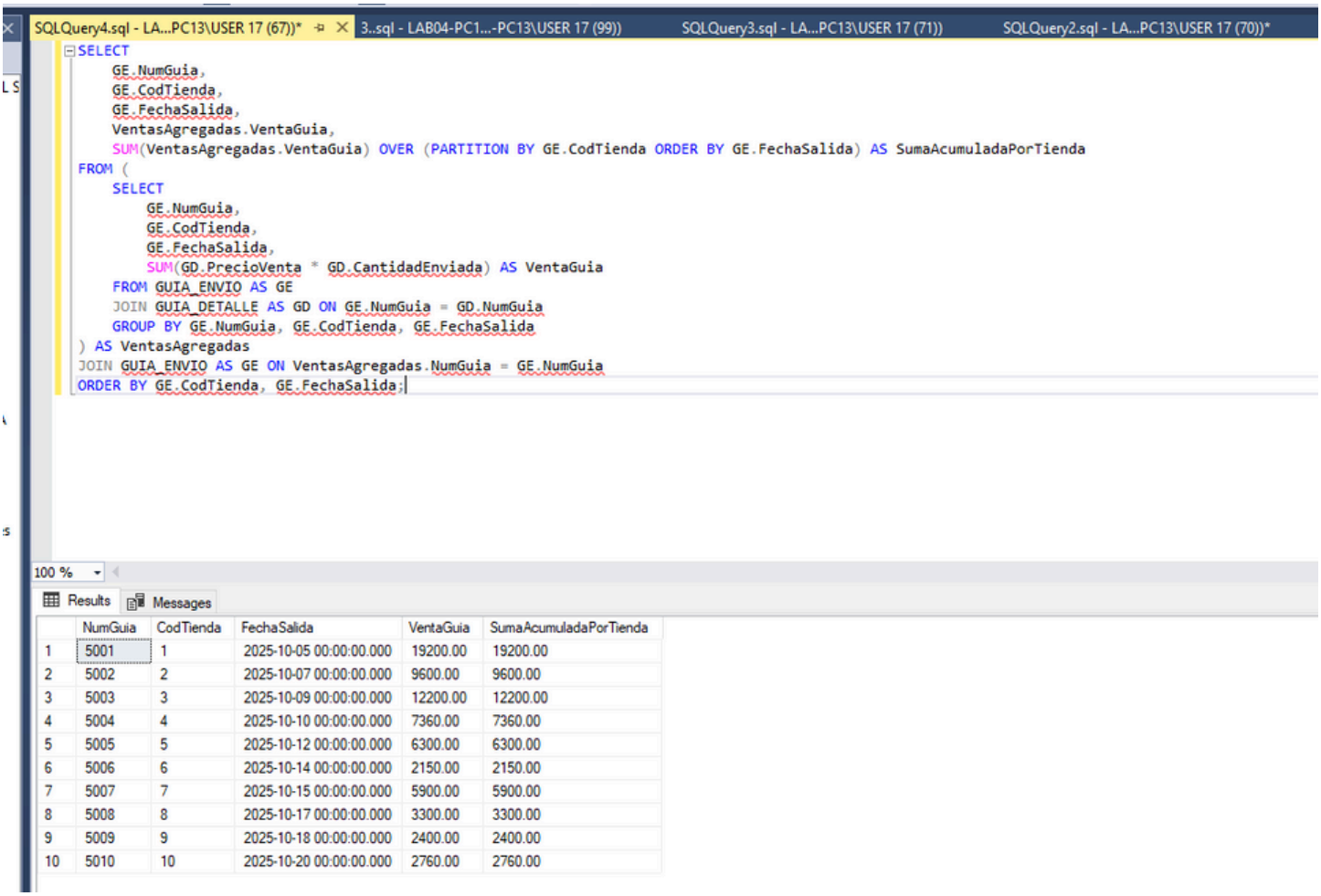
	NomTransportista	TotalEnviado	DensidadRank
1	LogiPeru	55	1
2	FastCargo	28	2
3	RápidoSur	10	3
4	ServiCargo	9	4
5	TransAndes	8	5
6	TransExpress	8	5
7	AndesCargo	8	5
8	PeruExpress	4	8
9	MegaTrans	3	9
10	LogísticaSur	2	10

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

SQL

```
SELECT

    GE.NumGuia,

    GE.CodTienda,

    GE.FechaSalida,

    VentasAgregadas.VentaGuia,

    SUM(VentasAgregadas.VentaGuia) OVER (PARTITION BY GE.CodTienda ORDER BY GE.FechaSalida) AS SumaAcumuladaPorTienda

FROM (

    SELECT

        GE.NumGuia,

        GE.CodTienda,

        GE.FechaSalida,
```

```
        SUM(GD.PrecioVenta * GD.CantidadEnviada) AS VentaGuia
    FROM GUIA_ENVIO AS GE
    JOIN GUIA_DETALLE AS GD ON GE.NumGuia = GD.NumGuia
    GROUP BY GE.NumGuia, GE.CodTienda, GE.FechaSalida
) AS VentasAgregadas
JOIN GUIA_ENVIO AS GE ON VentasAgregadas.NumGuia = GE.NumGuia
ORDER BY GE.CodTienda, GE.FechaSalida;
```



	NumGuia	CodTienda	FechaSalida	VentaGuia	SumaAcumuladaPorTienda
1	5001	1	2025-10-05 00:00:00.000	19200.00	19200.00
2	5002	2	2025-10-07 00:00:00.000	9600.00	9600.00
3	5003	3	2025-10-09 00:00:00.000	12200.00	12200.00
4	5004	4	2025-10-10 00:00:00.000	7360.00	7360.00
5	5005	5	2025-10-12 00:00:00.000	6300.00	6300.00
6	5006	6	2025-10-14 00:00:00.000	2150.00	2150.00
7	5007	7	2025-10-15 00:00:00.000	5900.00	5900.00
8	5008	8	2025-10-17 00:00:00.000	3300.00	3300.00
9	5009	9	2025-10-18 00:00:00.000	2400.00	2400.00
10	5010	10	2025-10-20 00:00:00.000	2760.00	2760.00

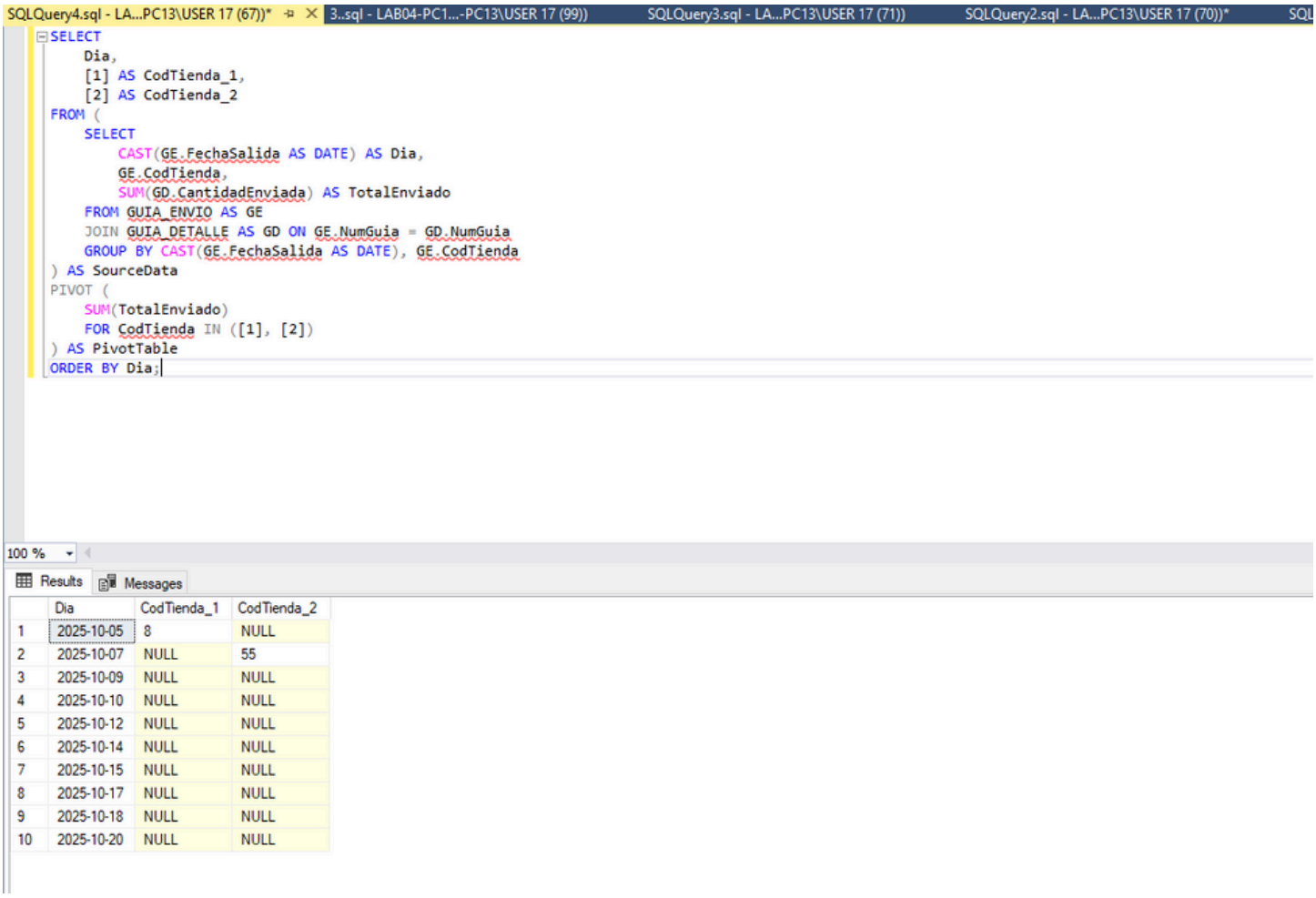
Ejercicios 31 - 40 (Consultas con PIVOT y CASE)

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

SQL

```
SELECT
    Dia,
```

```
    [1] AS CodTienda_1,
    [2] AS CodTienda_2
FROM (
    SELECT
        CAST(GE.FechaSalida AS DATE) AS Dia,
        GE.CodTienda,
        SUM(GD.CantidadEnviada) AS TotalEnviado
    FROM GUIA_ENVIO AS GE
    JOIN GUIA_DETALLE AS GD ON GE.NumGuia = GD.NumGuia
    GROUP BY CAST(GE.FechaSalida AS DATE), GE.CodTienda
) AS SourceData
PIVOT (
    SUM(TotalEnviado)
    FOR CodTienda IN ([1], [2])
) AS PivotTable
ORDER BY Dia;
```

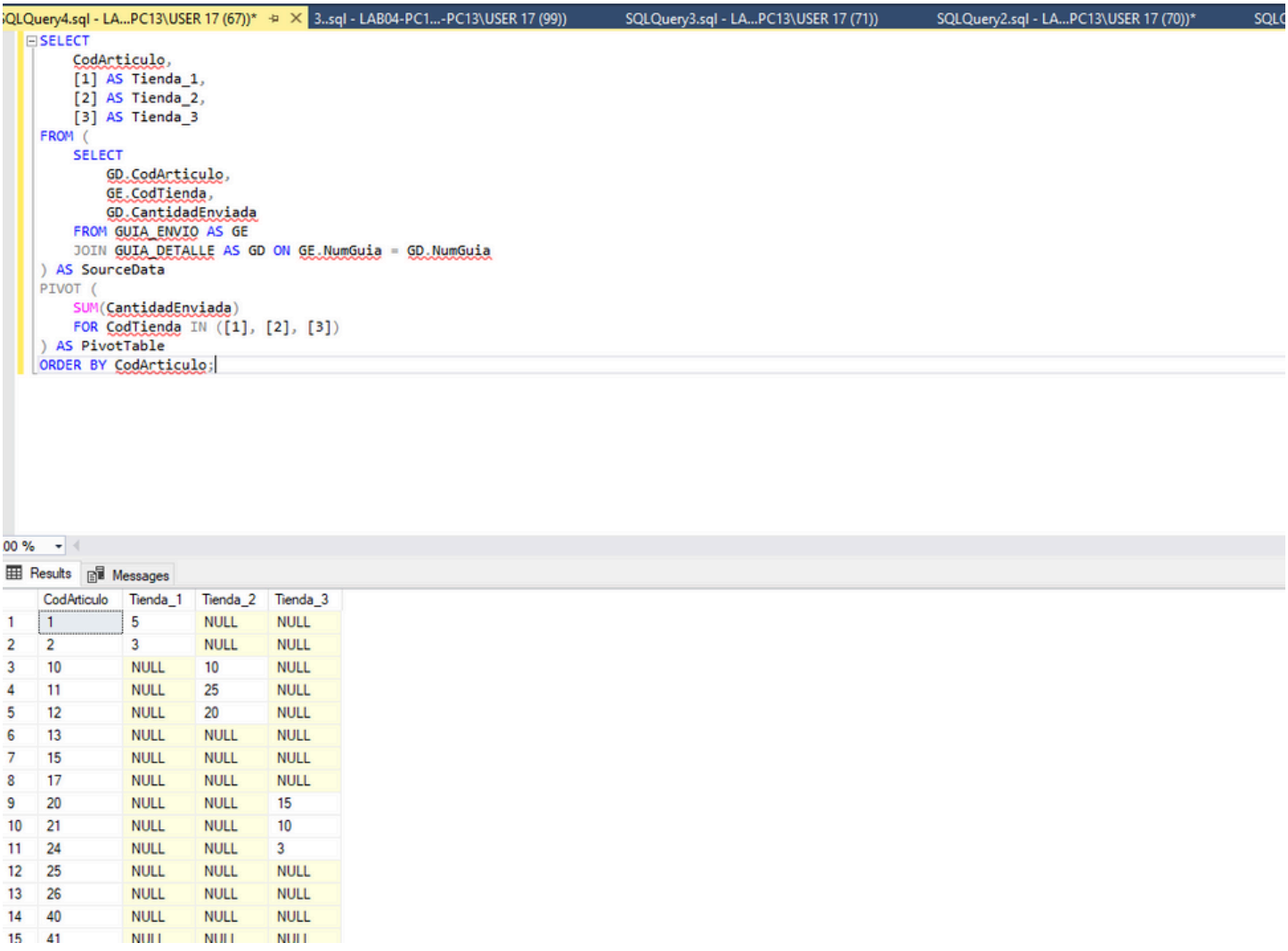


32. Mostrar CodArticulo y columnas con cantidades por tienda 1..3.

SQL

```
SELECT
    CodArticulo,
    [1] AS Tienda_1,
    [2] AS Tienda_2,
    [3] AS Tienda_3
FROM (
    SELECT
        GD.CodArticulo,
        GE.CodTienda,
        GD.CantidadEnviada
    FROM GUIA_ENVIO AS GE
```

```
    JOIN GUIA_DETALLE AS GD ON GE.NumGuia = GD.NumGuia
) AS SourceData
PIVOT (
    SUM(CantidadEnviada)
    FOR CodTienda IN ([1], [2], [3])
) AS PivotTable
ORDER BY CodArticulo;
```



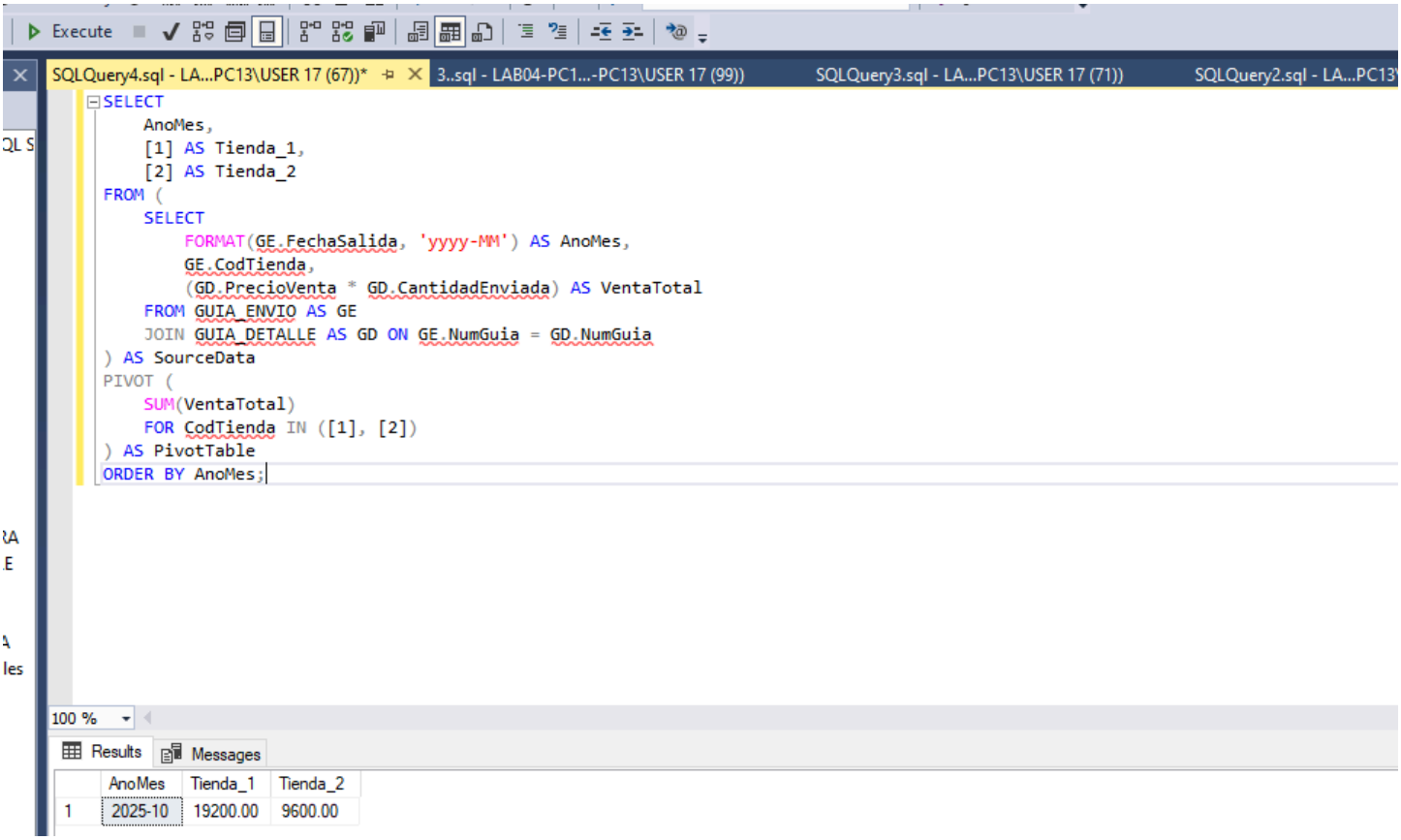
	CodArticulo	Tienda_1	Tienda_2	Tienda_3
1	1	5	NULL	NULL
2	2	3	NULL	NULL
3	10	NULL	10	NULL
4	11	NULL	25	NULL
5	12	NULL	20	NULL
6	13	NULL	NULL	NULL
7	15	NULL	NULL	NULL
8	17	NULL	NULL	NULL
9	20	NULL	NULL	15
10	21	NULL	NULL	10
11	24	NULL	NULL	3
12	25	NULL	NULL	NULL
13	26	NULL	NULL	NULL
14	40	NULL	NULL	NULL
15	41	NULL	NULL	NULL

33. Mostrar AñoMes y tiendas como columnas con suma de PrecioVenta*Cantidad.

SQL

```
SELECT
    AnoMes,
    [1] AS Tienda_1,
```

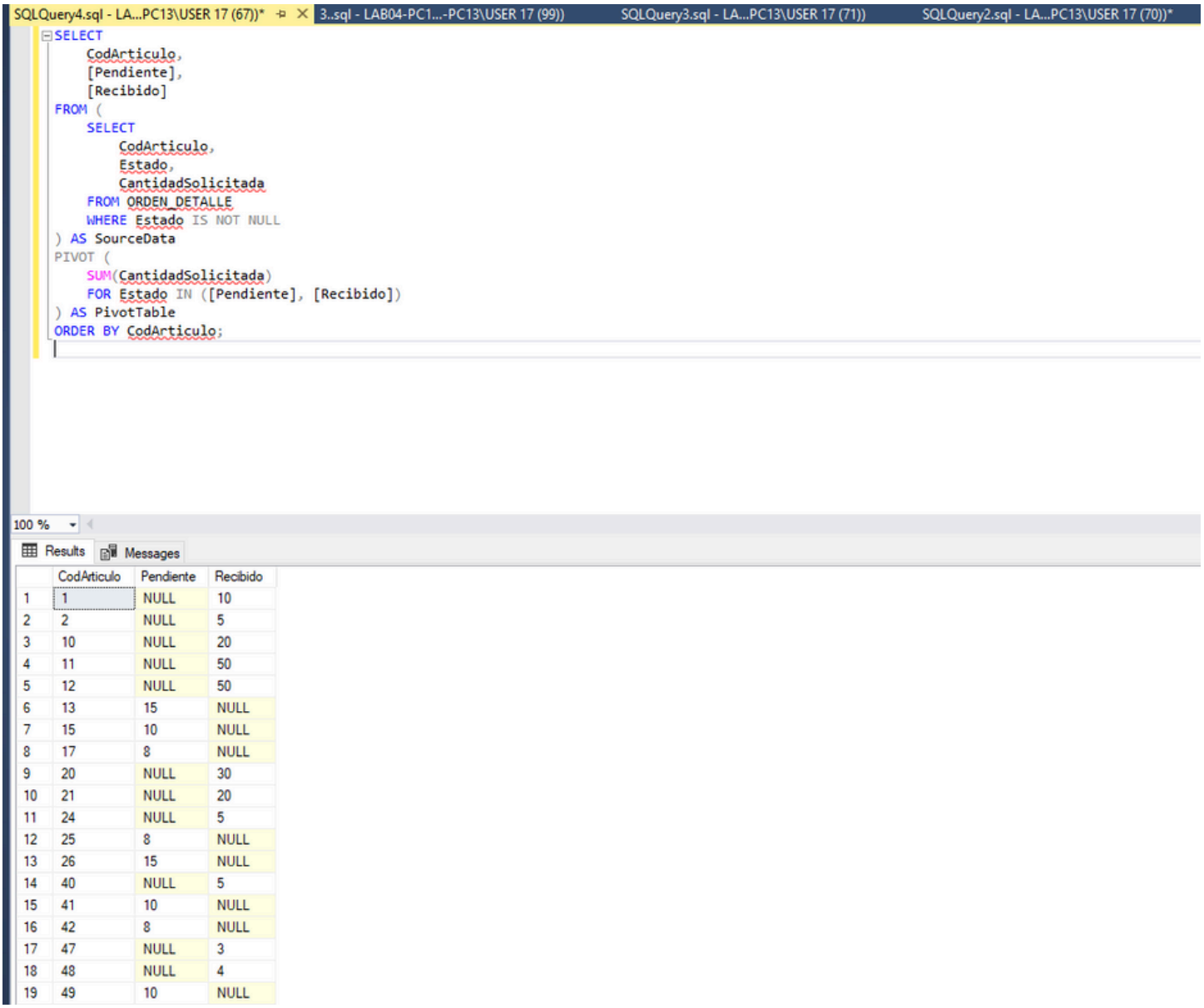
```
    [2] AS Tienda_2
FROM (
    SELECT
        FORMAT(GE.FechaSalida, 'yyyy-MM') AS AnoMes,
        GE.CodTienda,
        (GD.PrecioVenta * GD.CantidadEnviada) AS VentaTotal
    FROM GUIA_ENVIO AS GE
    JOIN GUIA_DETALLE AS GD ON GE.NumGuia = GD.NumGuia
) AS SourceData
PIVOT (
    SUM(VentaTotal)
    FOR CodTienda IN ([1], [2])
) AS PivotTable
ORDER BY AnoMes;
```



34. Mostrar CodArticulo con columnas para cada Estado (de orden de compra).

SQL

```
SELECT
    CodArticulo,
    [Pendiente],
    [Recibido]
FROM (
    SELECT
        CodArticulo,
        Estado,
        CantidadSolicitada
    FROM ORDEN_DETALLE
    WHERE Estado IS NOT NULL
) AS SourceData
PIVOT (
    SUM(CantidadSolicitada)
    FOR Estado IN ([Pendiente], [Recibido])
) AS PivotTable
ORDER BY CodArticulo;
```



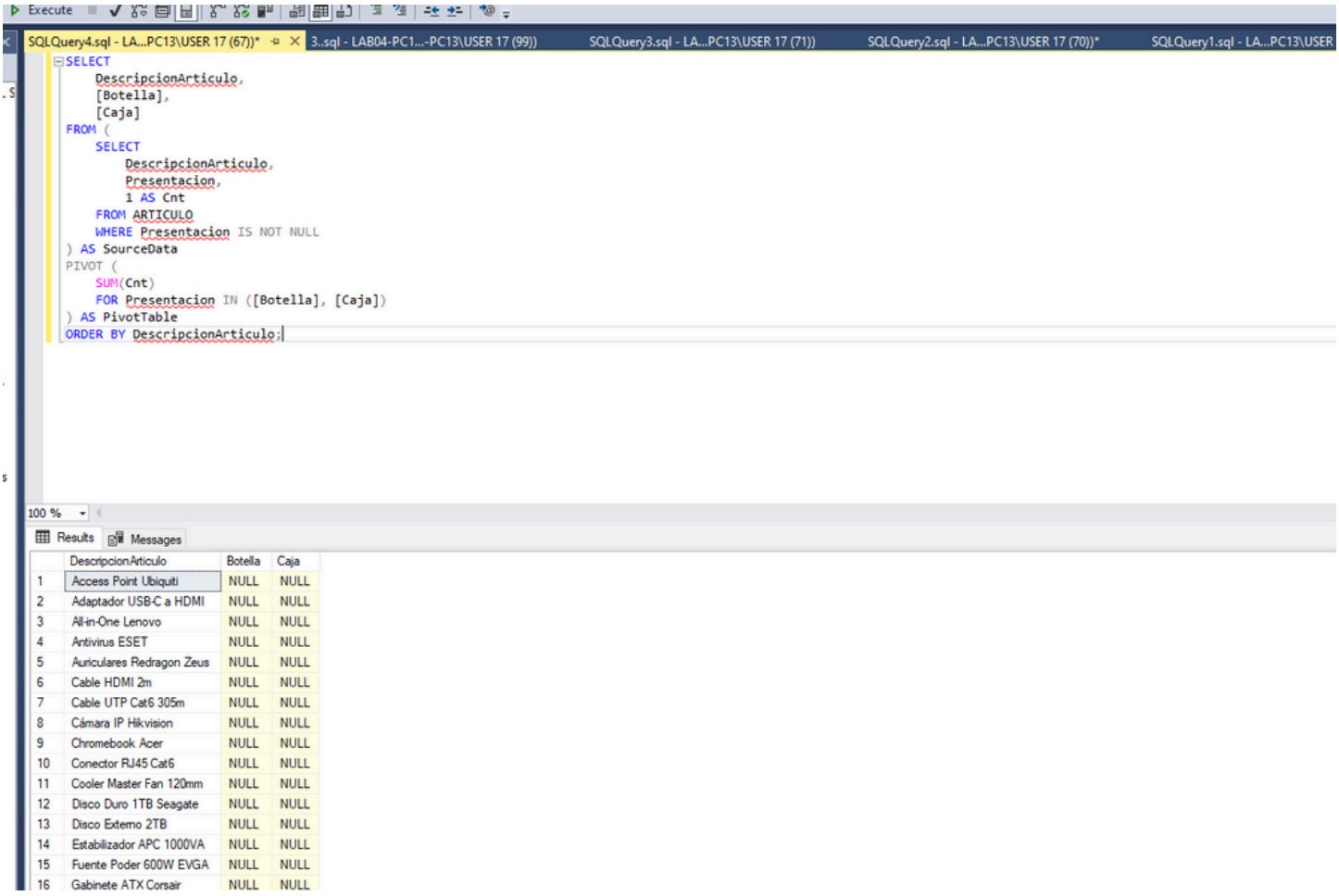
	CodArticulo	Pendiente	Recibido
1	1	NULL	10
2	2	NULL	5
3	10	NULL	20
4	11	NULL	50
5	12	NULL	50
6	13	15	NULL
7	15	10	NULL
8	17	8	NULL
9	20	NULL	30
10	21	NULL	20
11	24	NULL	5
12	25	8	NULL
13	26	15	NULL
14	40	NULL	5
15	41	10	NULL
16	42	8	NULL
17	47	NULL	3
18	48	NULL	4
19	49	10	NULL

35. Contar artículos por presentación pivoteada.

SQL

```
SELECT
    DescripcionArticulo,
    [Botella],
    [Caja]
FROM (
    SELECT
        DescripcionArticulo,
        Presentacion,
```

```
        1 AS Cnt
    FROM ARTICULO
    WHERE Presentacion IS NOT NULL
) AS SourceData
PIVOT (
    SUM(Cnt)
    FOR Presentacion IN ([Botella], [Caja])
) AS PivotTable
ORDER BY DescripcionArticulo;
```



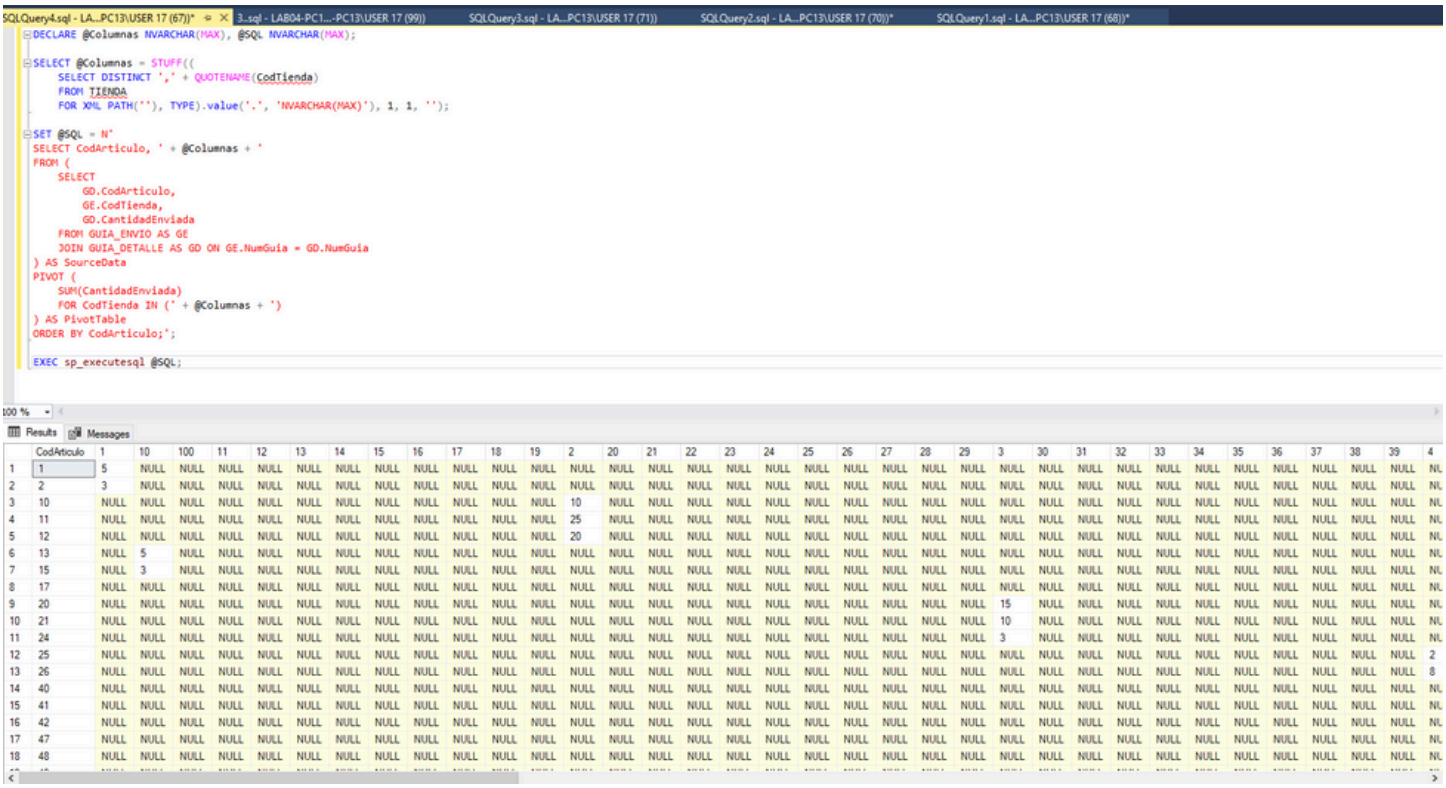
	DescripcionArticulo	Botella	Caja
1	Access Point Ubiquiti	NULL	NULL
2	Adaptador USB-C a HDMI	NULL	NULL
3	All-in-One Lenovo	NULL	NULL
4	Antivirus ESET	NULL	NULL
5	Auriculares Redragon Zeus	NULL	NULL
6	Cable HDMI 2m	NULL	NULL
7	Cable UTP Cat6 305m	NULL	NULL
8	Cámara IP Hikvision	NULL	NULL
9	Chromebook Acer	NULL	NULL
10	Conector RJ45 Cat6	NULL	NULL
11	Cooler Master Fan 120mm	NULL	NULL
12	Disco Duro 1TB Seagate	NULL	NULL
13	Disco Externo 2TB	NULL	NULL
14	Estabilizador APC 1000VA	NULL	NULL
15	Fuente Poder 600W EVGA	NULL	NULL
16	Gabinete ATX Corsair	NULL	NULL

36. Generar PIVOT dinámico para todas las tiendas (Ejemplo de patrón T-SQL).

SQL

```
DECLARE @Columnas NVARCHAR(MAX), @SQL NVARCHAR(MAX);
```

```
SELECT @Columnas = STUFF((
    SELECT DISTINCT ',' + QUOTENAME(CodTienda)
    FROM TIENDA
    FOR XML PATH(''), TYPE).value('.', 'NVARCHAR(MAX)'), 1, 1, '');

SET @SQL = N'
SELECT CodArticulo, ' + @Columnas + '
FROM (
    SELECT
        GD.CodArticulo,
        GE.CodTienda,
        GD.CantidadEnviada
    FROM GUIA_ENVIO AS GE
    JOIN GUIA_DETALLE AS GD ON GE.NumGuia = GD.NumGuia
) AS SourceData
PIVOT (
    SUM(CantidadEnviada)
    FOR CodTienda IN (' + @Columnas + ')
) AS PivotTable
ORDER BY CodArticulo;';

EXEC sp_executesql @SQL;
```



```
DECLARE @Columnas NVARCHAR(MAX), @SQL NVARCHAR(MAX);

SELECT @Columnas = STUFF((
    SELECT DISTINCT ',' + QUOTENAME(CodTienda)
    FROM TIENDA
    FOR XML PATH(''), TYPE).value('.', 'NVARCHAR(MAX)'), 1, 1, '');

SET @SQL = N'
SELECT CodArticulo, ' + @Columnas + '
FROM (
    SELECT
        GD.CodArticulo,
        GE.CodTienda,
        GD.CantidadEnviada
    FROM GUIA_ENVIO AS GE
    JOIN GUIA_DETALLE AS GD ON GE.NumGuia = GD.NumGuia
) AS SourceData
PIVOT (
    SUM(CantidadEnviada)
    FOR CodTienda IN (' + @Columnas + ')
) AS PivotTable
ORDER BY CodArticulo;';

EXEC sp_executesql @SQL;
```

Results | Messages

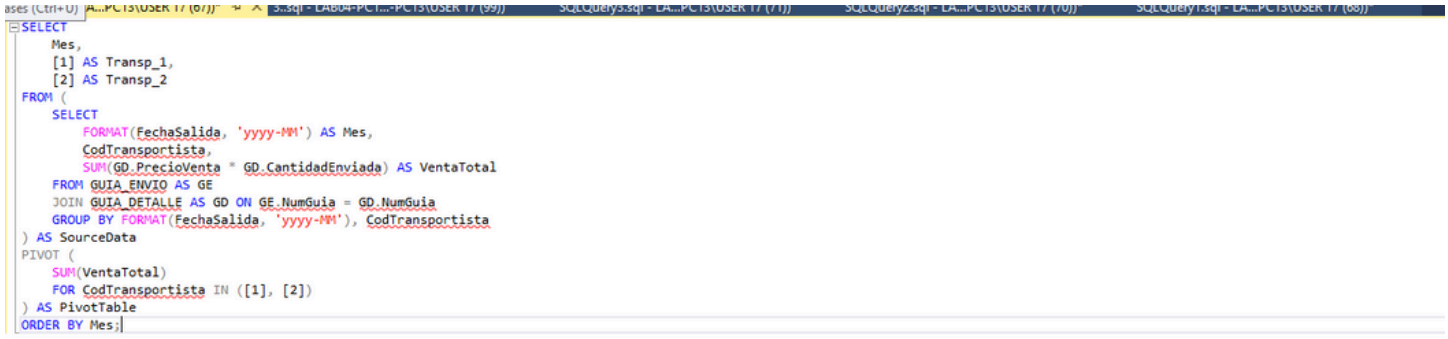
	CodArticulo	1	10	100	11	12	13	14	15	16	17	18	19	2	20	21	22	23	24	25	26	27	28	29	3	30	31	32	33	34	35	36	37	38	39	4
1	1	5	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NU
2	2	3	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NU
3	10	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	10	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NU
4	11	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	25	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NU
5	12	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	20	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NU
6	13	NULL	5	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NU
7	15	NULL	3	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NU
8	17	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NU
9	20	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	15	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NU
10	21	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	10	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NU
11	24	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	3	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NU
12	25	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	2	
13	26	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	8	
14	40	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NU
15	41	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NU
16	42	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NU
17	47	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NU
18	48	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NU

37. Mostrar Mes y columnas por transportista con totales.

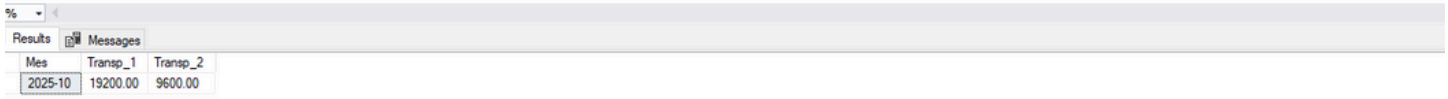
SQL

```
SELECT
    Mes,
    [1] AS Transp_1,
    [2] AS Transp_2
FROM (
    SELECT
        FORMAT(FechaSalida, 'yyyy-MM') AS Mes,
        CodTransportista,
        SUM(GD.PrecioVenta * GD.CantidadEnviada) AS VentaTotal
    FROM GUIA_ENVIO AS GE
    JOIN GUIA_DETALLE AS GD ON GE.NumGuia = GD.NumGuia
    GROUP BY FORMAT(FechaSalida, 'yyyy-MM'), CodTransportista
) AS SourceData
PIVOT (
```

```
    SUM(VentaTotal)
    FOR CodTransportista IN ([1], [2])
) AS PivotTable
ORDER BY Mes;
```



```
SELECT
    Mes,
    [1] AS Transp_1,
    [2] AS Transp_2
FROM (
    SELECT
        FORMAT(FechaSalida, 'yyyy-MM') AS Mes,
        CodTransportista,
        SUM(GD.PrecioVenta * GD.CantidadEnviada) AS VentaTotal
    FROM GUIA_ENVIO AS GE
    JOIN GUIA_DETALLE AS GD ON GE.NumGuia = GD.NumGuia
    GROUP BY FORMAT(FechaSalida, 'yyyy-MM'), CodTransportista
) AS SourceData
PIVOT (
    SUM(VentaTotal)
    FOR CodTransportista IN ([1], [2])
) AS PivotTable
ORDER BY Mes;
```



Results | Messages

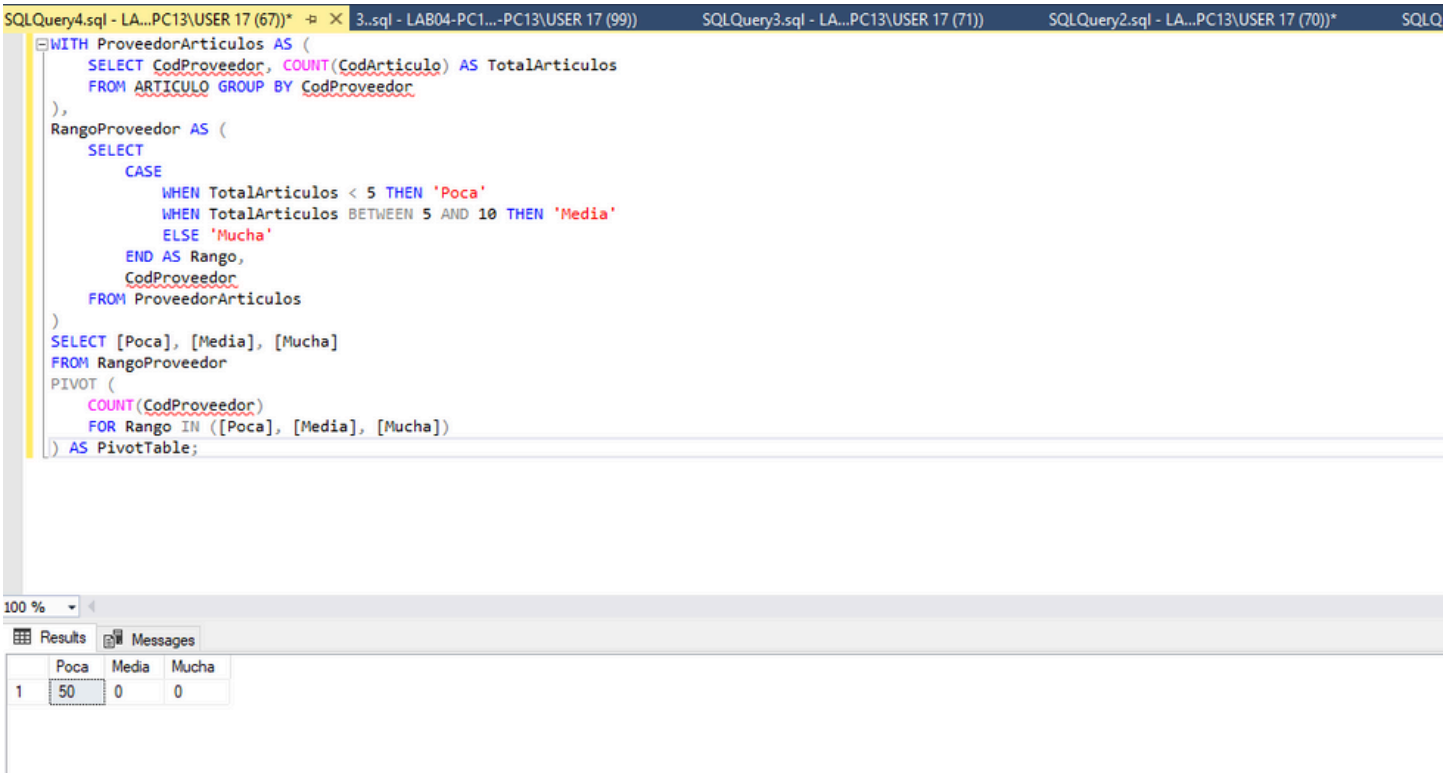
Mes	Transp_1	Transp_2
2025-10	19200.00	9600.00

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

SQL

```
WITH ProveedorArticulos AS (
    SELECT CodProveedor, COUNT(CodArticulo) AS TotalArticulos
    FROM ARTICULO GROUP BY CodProveedor
),
RangoProveedor AS (
    SELECT
        CASE
            WHEN TotalArticulos < 5 THEN 'Poca'
            WHEN TotalArticulos BETWEEN 5 AND 10 THEN 'Media'
            ELSE 'Mucha'
        END AS Rango,
        CodProveedor
```

```
    FROM ProveedorArticulos
)
SELECT [Poca], [Media], [Mucha]
FROM RangoProveedor
PIVOT (
    COUNT(CodProveedor)
    FOR Rango IN ([Poca], [Media], [Mucha])
) AS PivotTable;
```

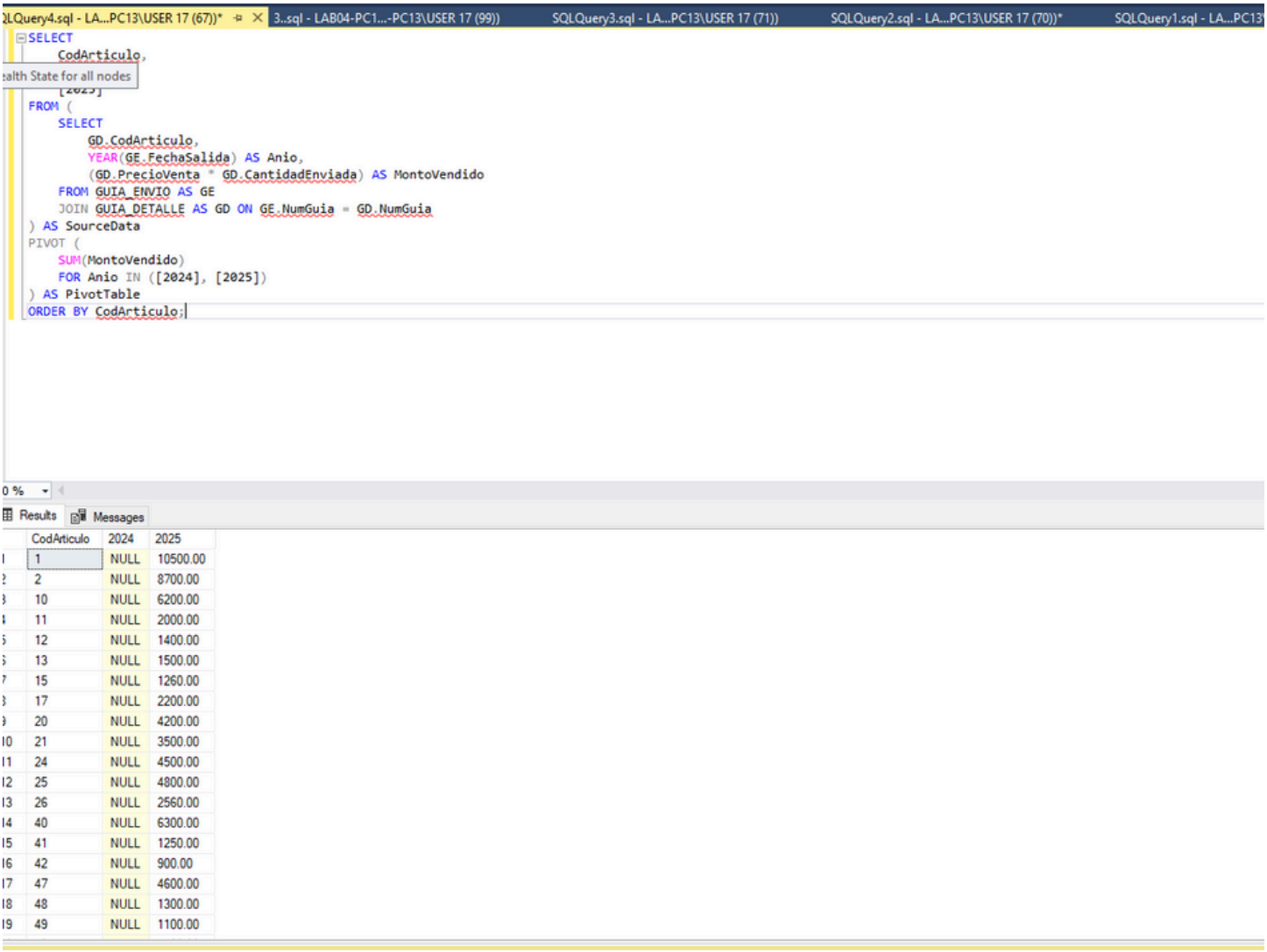


39. Mostrar CodArticulo y columnas por año con monto total vendido.

SQL

```
SELECT
    CodArticulo,
    [2024],
    [2025]
FROM (
    SELECT
```

```
        GD.CodArticulo,
        YEAR(GE.FechaSalida) AS Anio,
        (GD.PrecioVenta * GD.CantidadEnviada) AS MontoVendido
    FROM GUIA_ENVIO AS GE
    JOIN GUIA_DETALLE AS GD ON GE.NumGuia = GD.NumGuia
) AS SourceData
PIVOT (
    SUM(MontoVendido)
    FOR Anio IN ([2024], [2025])
) AS PivotTable
ORDER BY CodArticulo;
```

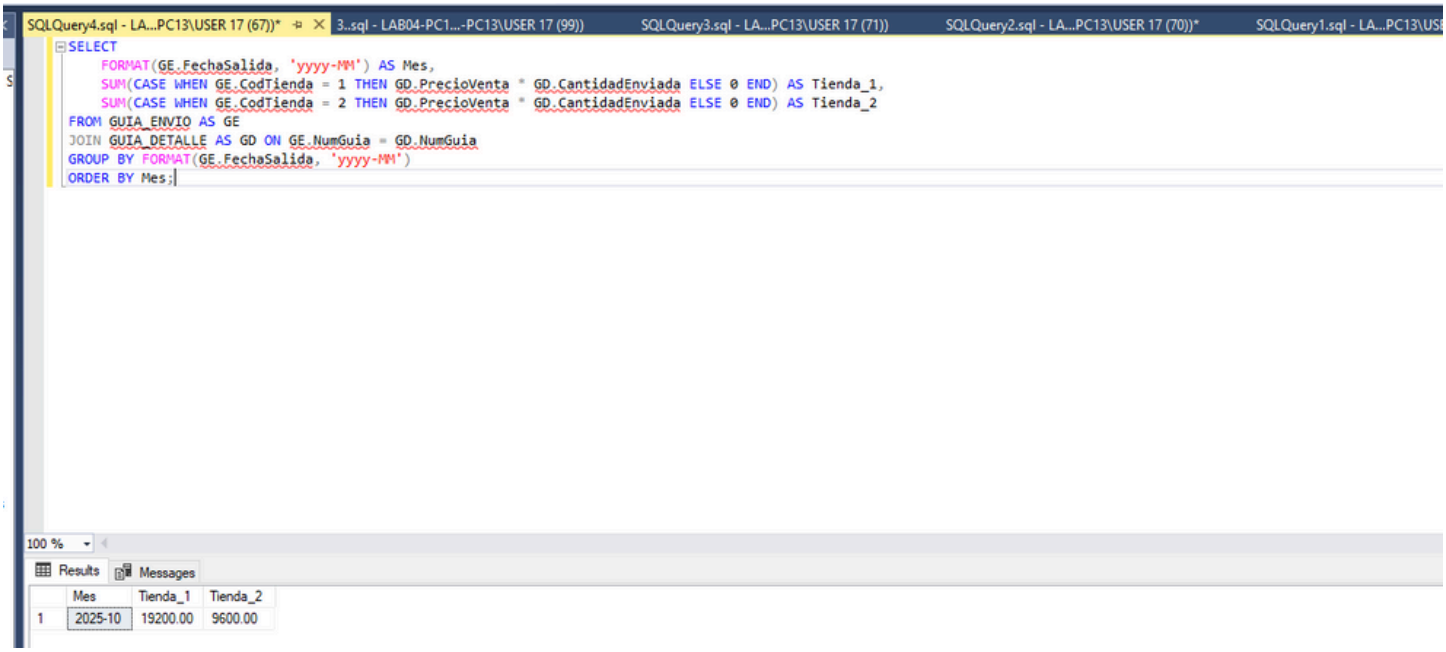


	CodArticulo	2024	2025
1	1	NULL	10500.00
2	2	NULL	8700.00
3	10	NULL	6200.00
4	11	NULL	2000.00
5	12	NULL	1400.00
6	13	NULL	1500.00
7	15	NULL	1260.00
8	17	NULL	2200.00
9	20	NULL	4200.00
10	21	NULL	3500.00
11	24	NULL	4500.00
12	25	NULL	4800.00
13	26	NULL	2560.00
14	40	NULL	6300.00
15	41	NULL	1250.00
16	42	NULL	900.00
17	47	NULL	4600.00
18	48	NULL	1300.00
19	49	NULL	1100.00

40. Mostrar Mes y columnas por tienda (Alternativa con CASE).

SQL

```
SELECT

    FORMAT(GE.FechaSalida, 'yyyy-MM') AS Mes,

    SUM(CASE WHEN GE.CodTienda = 1 THEN GD.PrecioVenta *
GD.CantidadEnviada ELSE 0 END) AS Tienda_1,

    SUM(CASE WHEN GE.CodTienda = 2 THEN GD.PrecioVenta *
GD.CantidadEnviada ELSE 0 END) AS Tienda_2

FROM GUIA_ENVIO AS GE

JOIN GUIA_DETALLE AS GD ON GE.NumGuia = GD.NumGuia

GROUP BY FORMAT(GE.FechaSalida, 'yyyy-MM')

ORDER BY Mes;
```

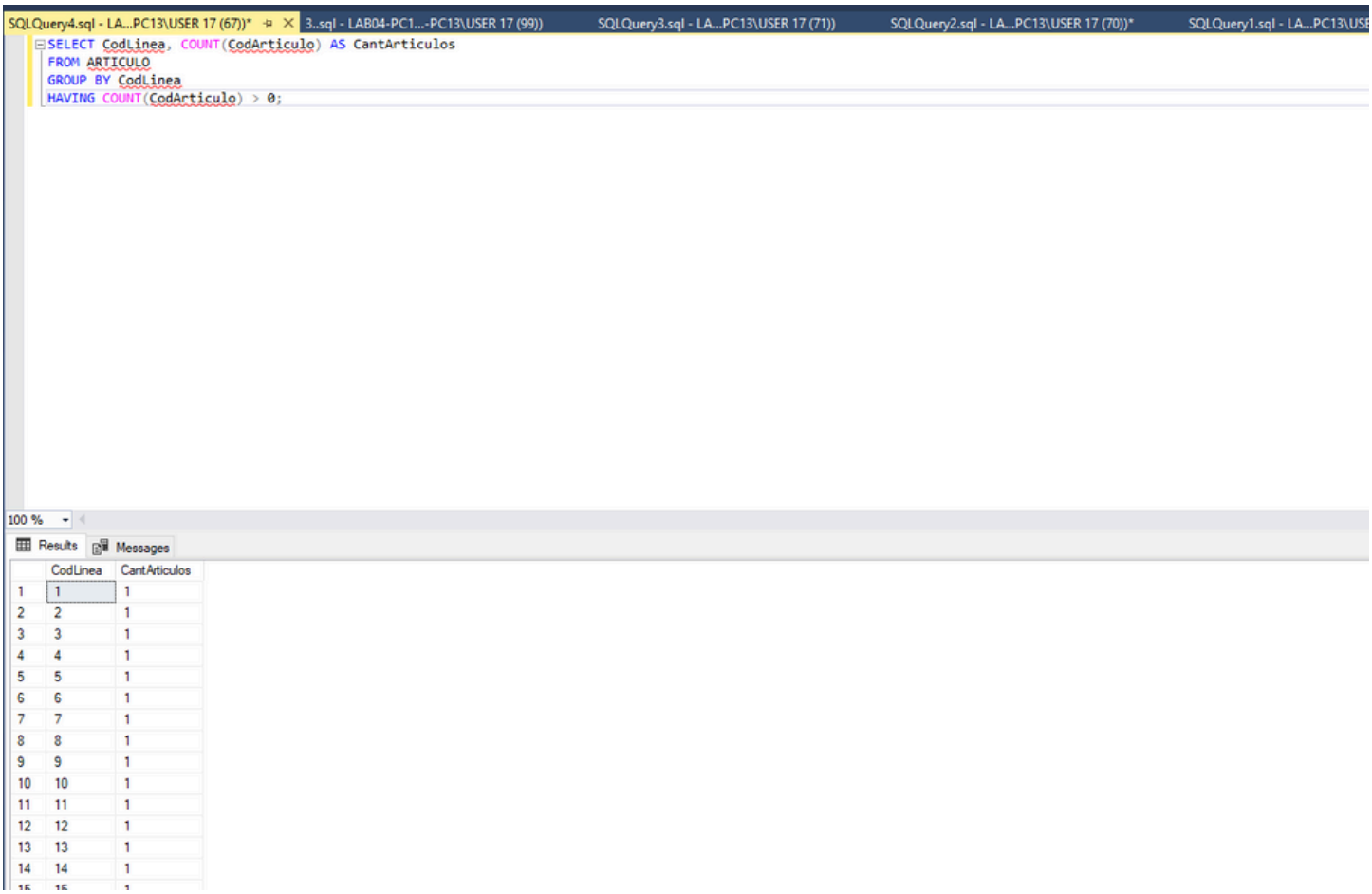


Ejercicios 41 - 50 (Consultas con HAVING)

41. Mostrar CodLinea y CantArticulos donde CantArticulos > 0.

SQL

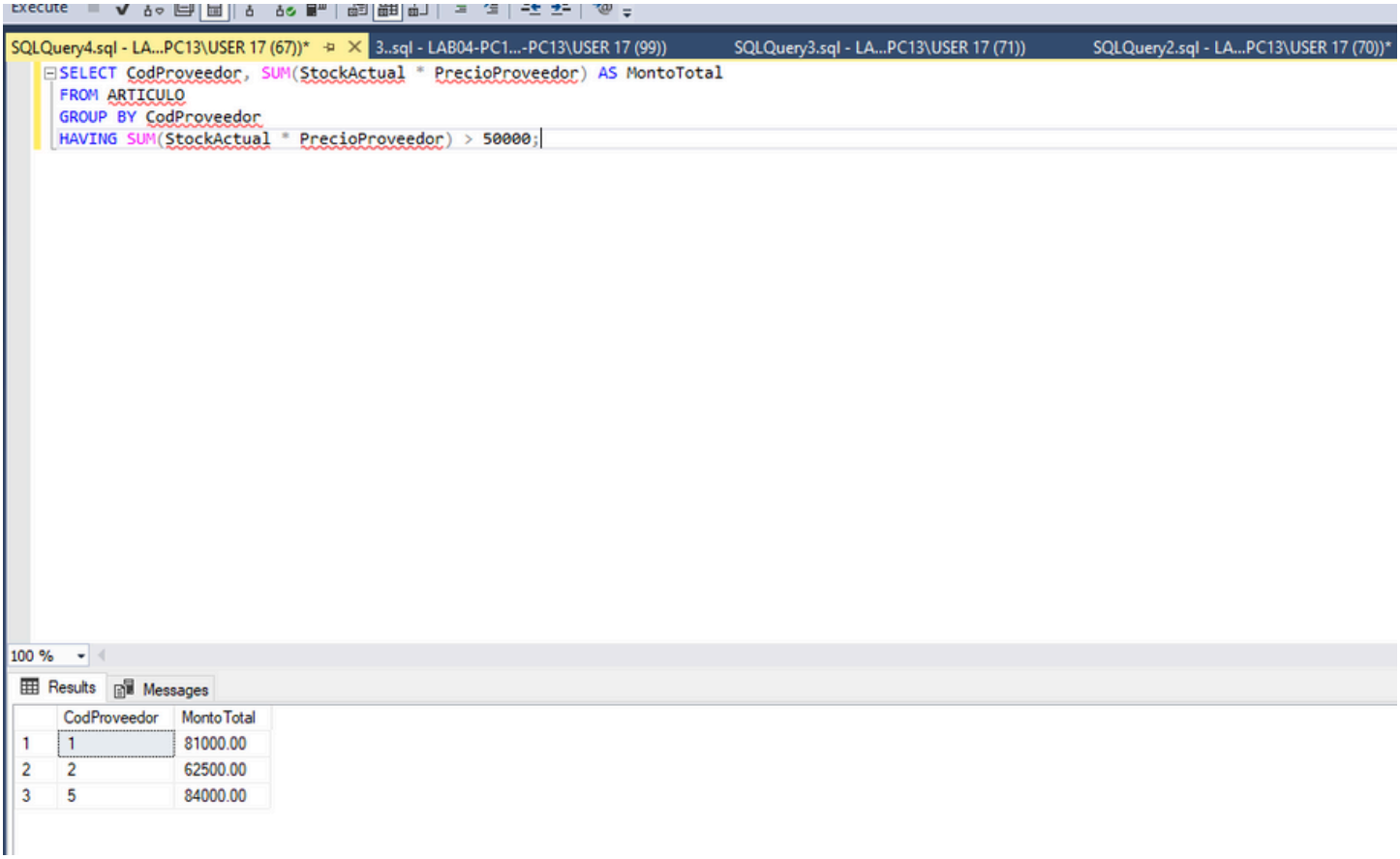
```
SELECT CodLinea, COUNT(CodArticulo) AS CantArticulos

FROM ARTICULO

GROUP BY CodLinea

HAVING COUNT(CodArticulo) > 10;
```



42. Mostrar CodProveedor y MontoTotal donde MontoTotal > 50000.

SQL

```
SELECT CodProveedor, SUM(StockActual * PrecioProveedor) AS MontoTotal
FROM ARTICULO
GROUP BY CodProveedor
HAVING SUM(StockActual * PrecioProveedor) > 50000;
```



43. Mostrar CodTienda y PromedioGuia donde PromedioGuia > 1000.

SQL

```
SELECT GE.CodTienda, AVG(GD.CantidadEnviada * GD.PrecioVenta) AS
PromedioGuia

FROM GUIA_ENVIO AS GE

INNER JOIN GUIA_DETALLE AS GD ON GE.NumGuia = GD.NumGuia

GROUP BY GE.CodTienda

HAVING AVG(GD.CantidadEnviada * GD.PrecioVenta) > 1000;
```

```sql
SELECT GE.CodTienda, AVG(GD.CantidadEnviada * GD.PrecioVenta) AS PromedioGuia
FROM GUIA_ENVIO AS GE
INNER JOIN GUIA_DETALLE AS GD ON GE.NumGuia = GD.NumGuia
GROUP BY GE.CodTienda
HAVING AVG(GD.CantidadEnviada * GD.PrecioVenta) > 1000;
```

0 %

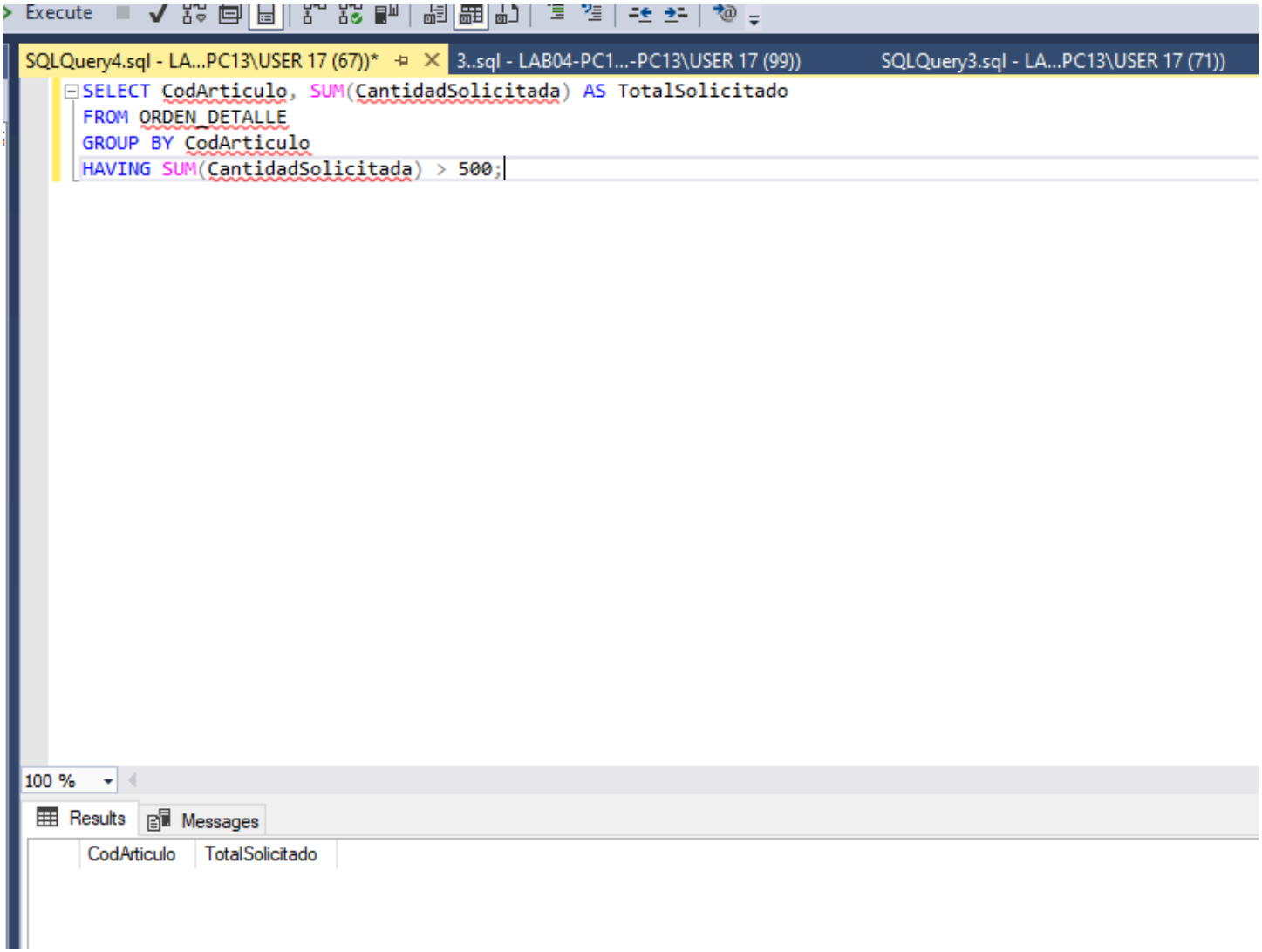
Results | Messages

	CodTienda	PromedioGuia
1	1	9600.00
2	2	3200.00
3	3	4066.6666
4	4	3680.00
5	5	6300.00
6	6	1075.00
7	7	2950.00
8	8	1650.00
9	9	2400.00
10	10	1380.00

44. Mostrar CodArticulo y TotalSolicitado > 500.

SQL

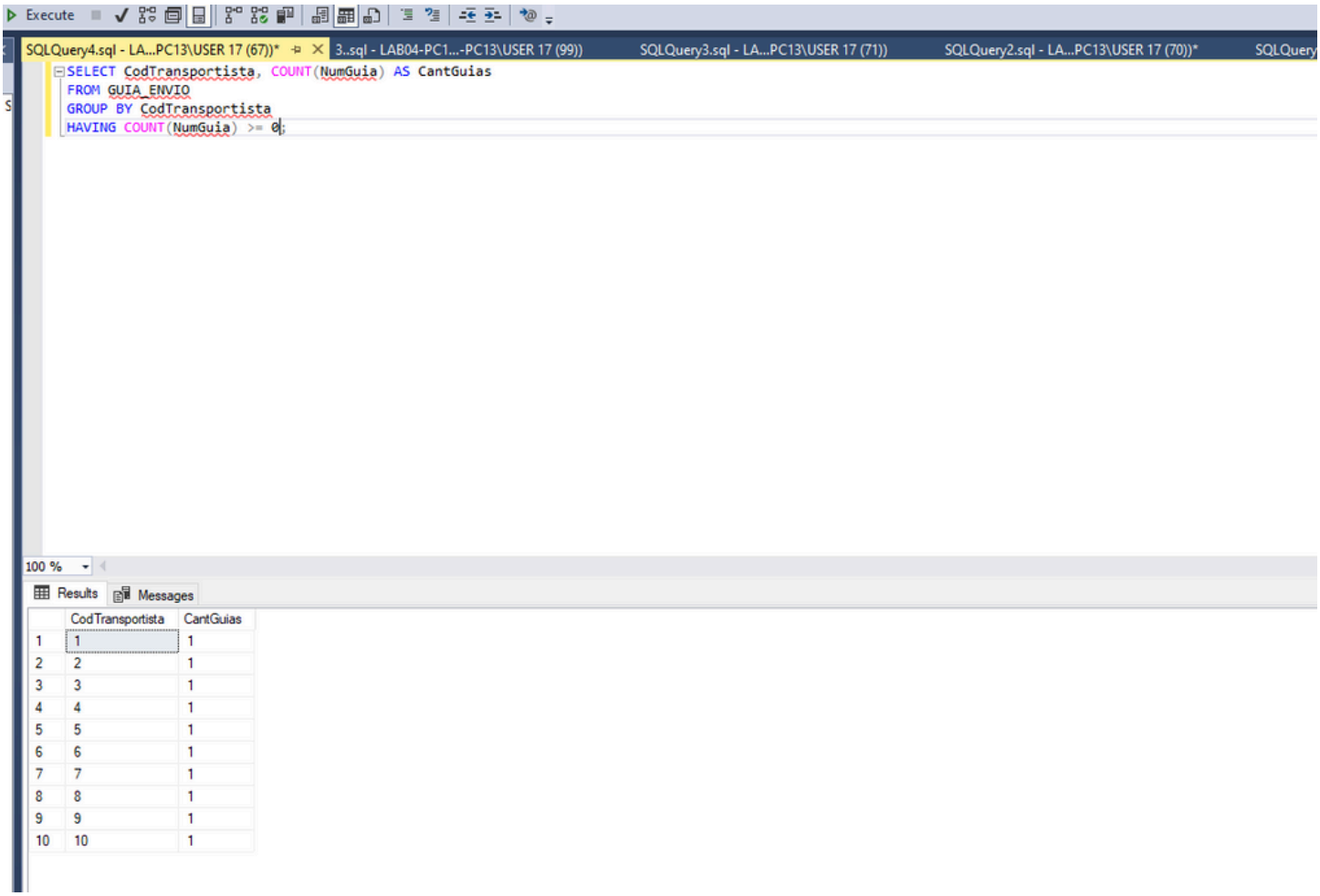
```sql
SELECT CodArticulo, SUM(CantidadSolicitada) AS TotalSolicitado
FROM ORDEN_DETALLE
GROUP BY CodArticulo
HAVING SUM(CantidadSolicitada) > 500;
```



45. Mostrar CodTransportista y CantGuías >= 5.

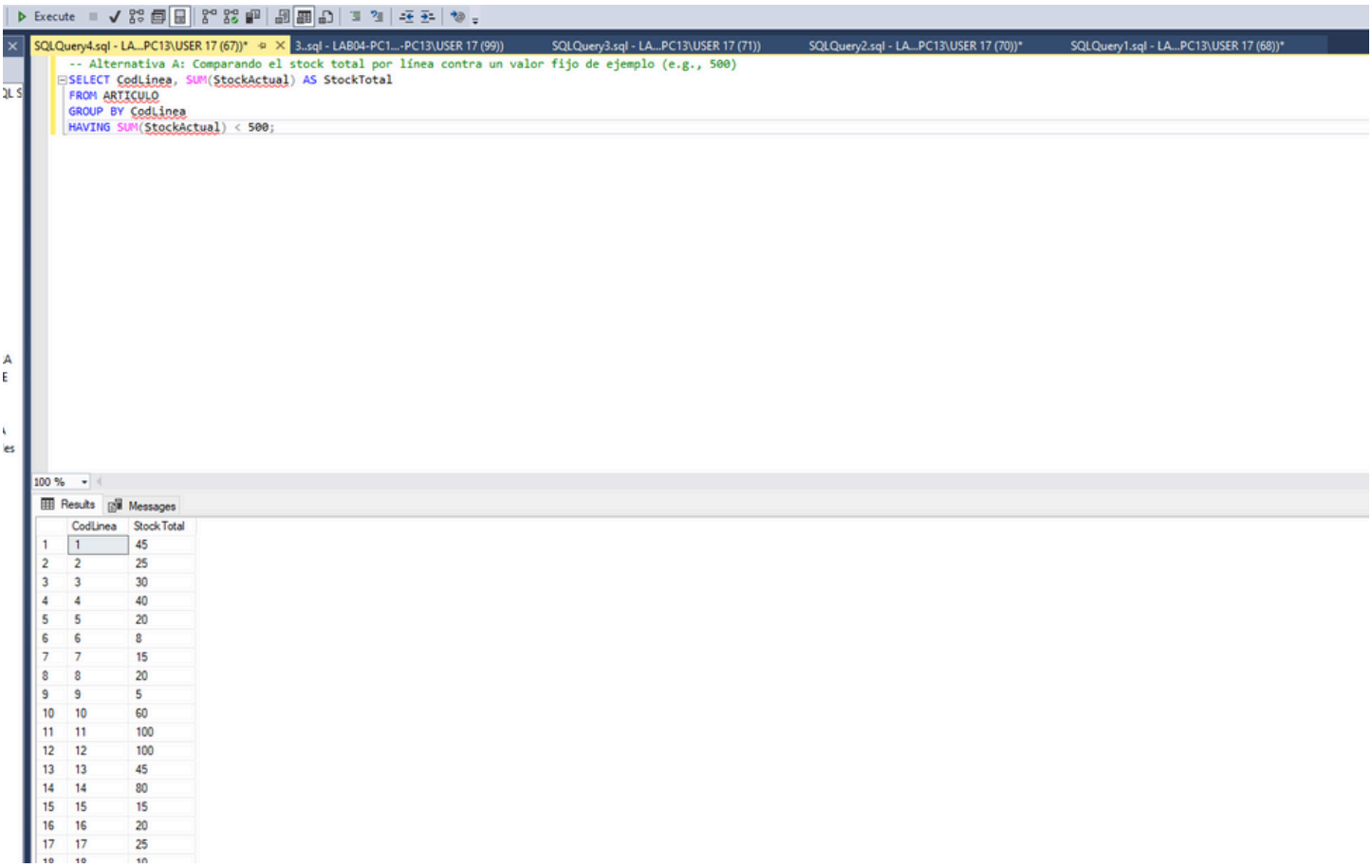
SQL

```
SELECT CodTransportista, COUNT(NumGuia) AS CantGuias
FROM GUIA_ENVIO
GROUP BY CodTransportista
HAVING COUNT(NumGuia) >= 5;
```



46. Mostrar líneas donde SUM(StockActual) < SUM(StockMinimo * NumArticulosPorLinea) — ejemplo conceptual.

```
-- Alternativa A: Comparando el stock total por línea contra un valor fijo de ejemplo (e.g., 500)

SELECT CodLinea, SUM(StockActual) AS StockTotal

FROM ARTICULO

GROUP BY CodLinea

HAVING SUM(StockActual) < 500;
```

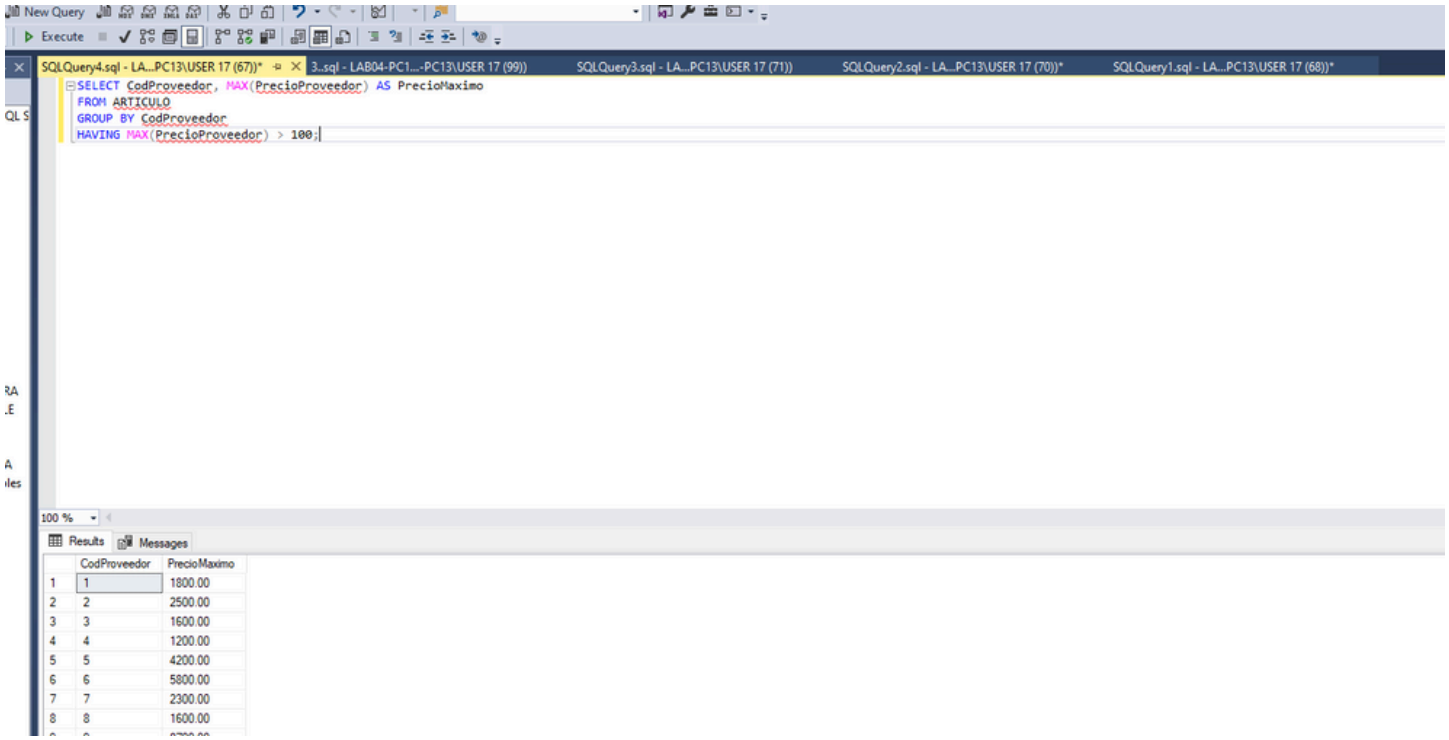


	CodLinea	StockTotal
1	1	45
2	2	25
3	3	30
4	4	40
5	5	20
6	6	8
7	7	15
8	8	20
9	9	5
10	10	60
11	11	100
12	12	100
13	13	45
14	14	80
15	15	15
16	16	20
17	17	25

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

SQL

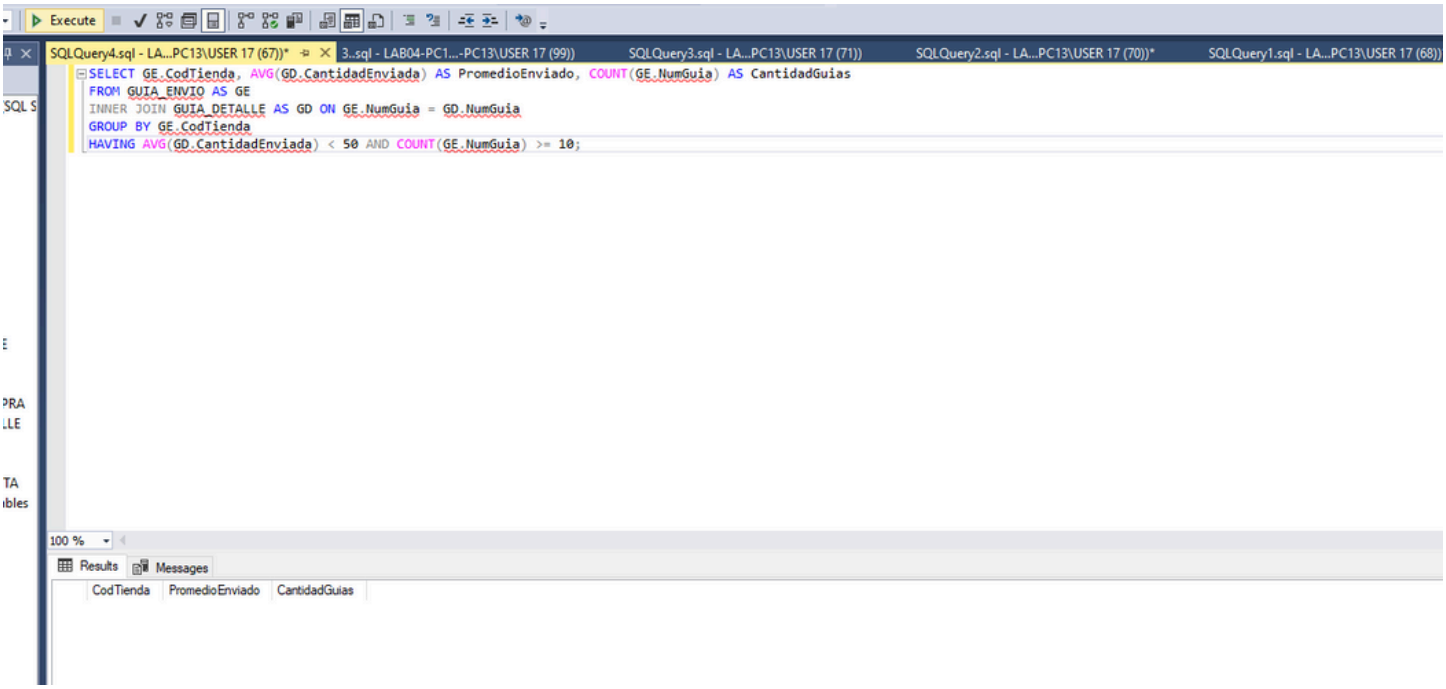
```
SELECT CodProveedor, MAX(PrecioProveedor) AS PrecioMaximo
FROM ARTICULO
GROUP BY CodProveedor
HAVING MAX(PrecioProveedor) > 100;
```



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

SQL

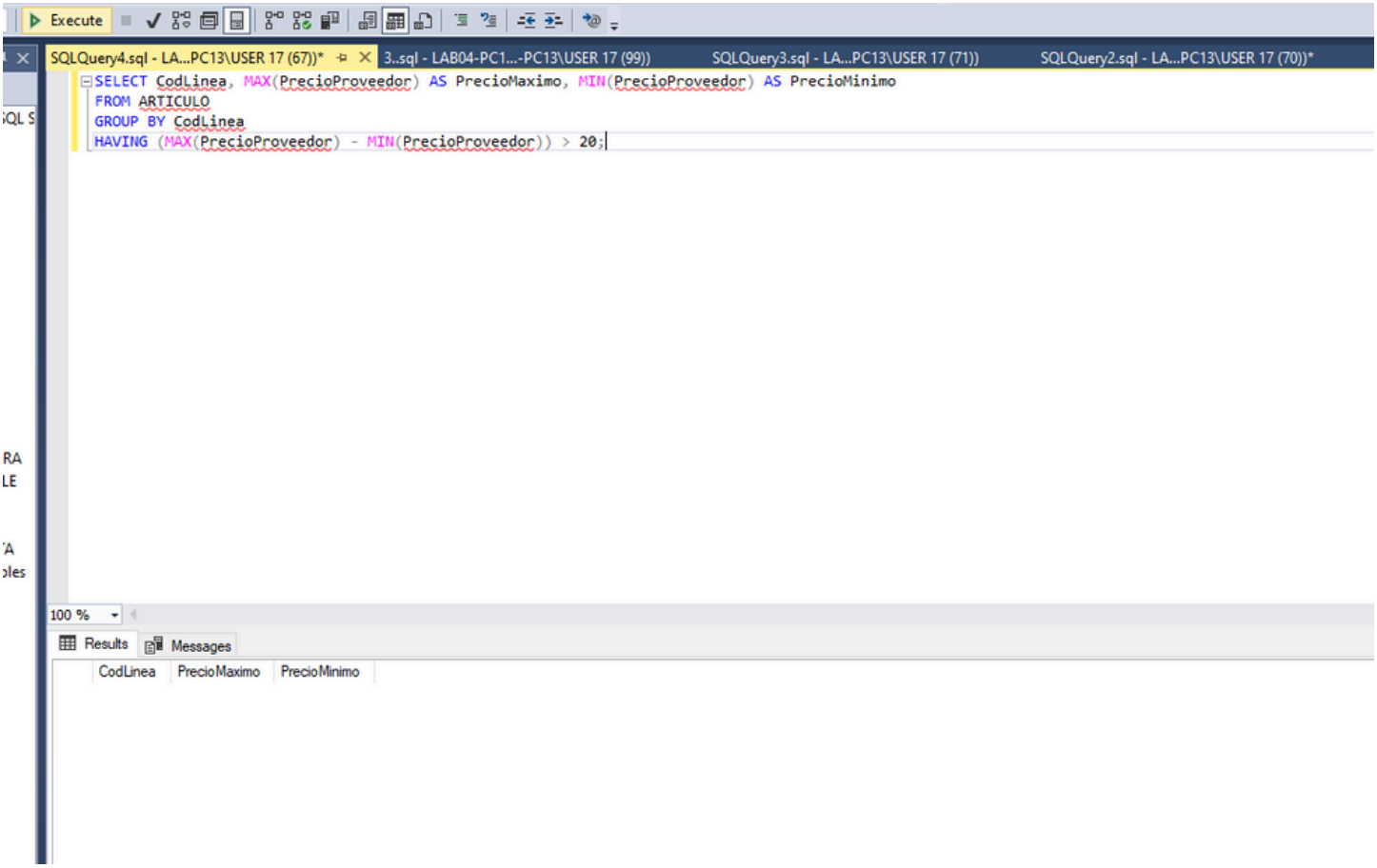
```
SELECT GE.CodTienda, AVG(GD.CantidadEnviada) AS PromedioEnviado,
COUNT(GE.NumGuia) AS CantidadGuias

FROM GUIA_ENVIO AS GE

INNER JOIN GUIA_DETALLE AS GD ON GE.NumGuia = GD.NumGuia

GROUP BY GE.CodTienda

HAVING AVG(GD.CantidadEnviada) < 50 AND COUNT(GE.NumGuia) >= 10;
```



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

SQL

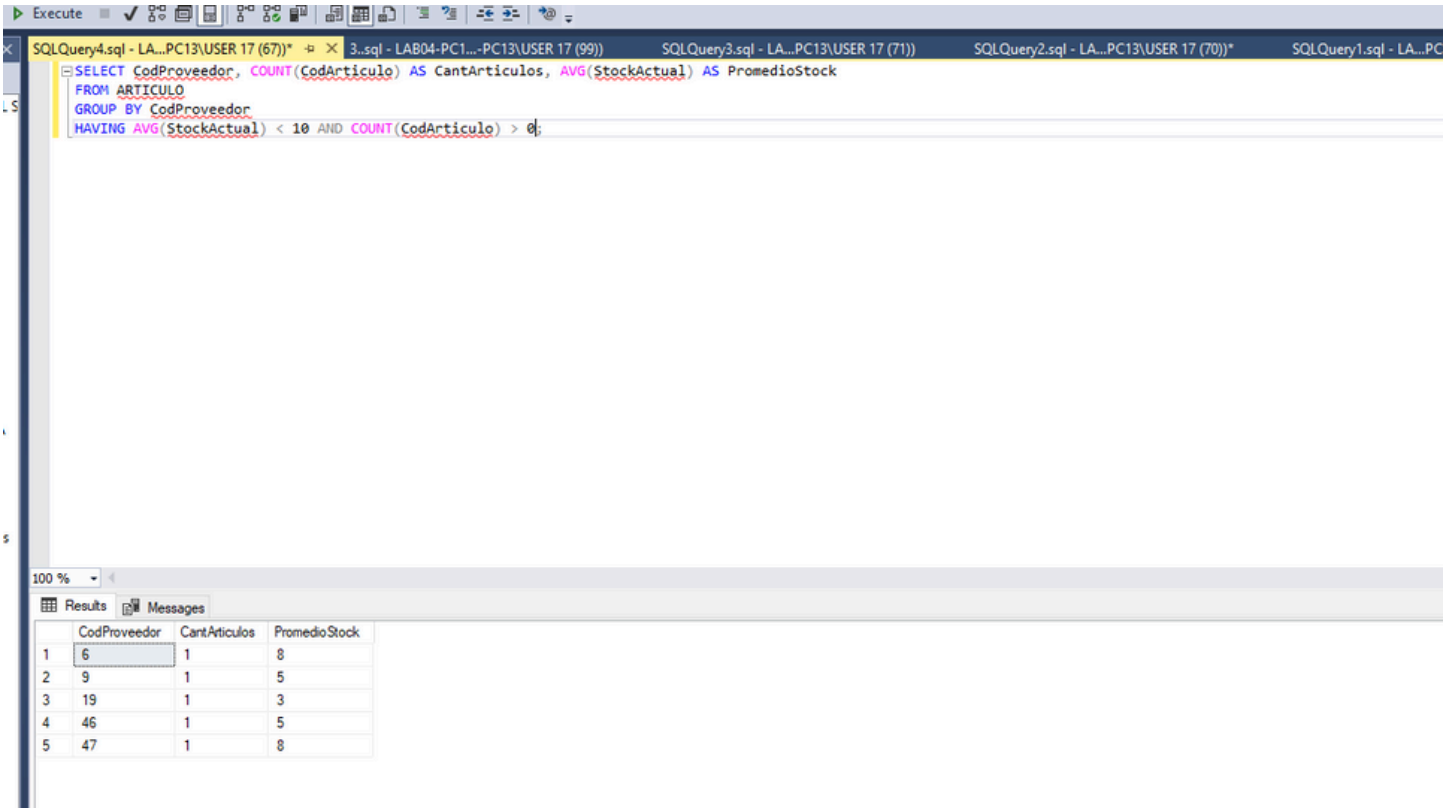
```
SELECT CodLinea, MAX(PrecioProveedor) AS PrecioMaximo,
MIN(PrecioProveedor) AS PrecioMinimo

FROM ARTICULO

GROUP BY CodLinea

HAVING (MAX(PrecioProveedor) - MIN(PrecioProveedor)) > 20;
```



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

SQL

```
SELECT CodProveedor, COUNT(CodArticulo) AS CantArticulos, AVG(StockActual) AS PromedioStock

FROM ARTICULO

GROUP BY CodProveedor

HAVING AVG(StockActual) < 20 AND COUNT(CodArticulo) > 5;
```



	CodProveedor	CantArticulos	PromedioStock
1	6	1	8
2	9	1	5
3	19	1	3
4	46	1	5
5	47	1	8