

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;
```

The screenshot shows the SQL Server Management Studio interface. In the top query editor window, a query is written to calculate the total monetary value of the inventory:

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

In the bottom results pane, there is one row of data:

	TotalInventarioMonetario
1	952590.00

2. Obtener CodLinea y PrecioProveedor promedio.

SQL

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

```
SQLQuery3.sql - LA...PC20\USER 17 (66)*  X LAB04-PC20\MSSQL...PERU - dbo.LINEA      3..sql - LAB04-PC2...-PC20\USER 17 (90))      SQLQuery2.sql - LA...PC20\USER 1
SELECT CodLinea, AVG(PrecioProveedor) AS PrecioPromedio
FROM ARTICULO
GROUP BY CodLinea;
```

100 %

Results Messages

	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;
```

Results	
	Messages
ArticulosDescontinuados	50

4. Mostrar PrecioMaximo y PrecioMinimo del catálogo.

SQL

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

SQLQuery3.sql - LA...PC20\USER 17 (66)*		LAB04-PC20\MSSQL...PERU - dbo.LINEA	3..sql - LAB04-PC2...-PC20\USER 17 (90)	SQLQuery2.sql -				
		SELECT MAX(PrecioProveedor) AS PrecioMaximo, MIN(PrecioProveedor) AS PrecioMinimo						
		FROM ARTICULO;						
100 %								
Results								
<table border="1"><thead><tr><th>PrecioMaximo</th><th>PrecioMinimo</th></tr></thead><tbody><tr><td>8700.00</td><td>25.00</td></tr></tbody></table>					PrecioMaximo	PrecioMinimo	8700.00	25.00
PrecioMaximo	PrecioMinimo							
8700.00	25.00							

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;
```

The screenshot shows the SQL Server Management Studio interface. In the top pane, there are four tabs: 'SQLQuery3.sql - LA...PC20\USER 17 (66)*' (highlighted), 'LAB04-PC20\MSSQL...PERU - dbo.LINEA', '3..sql - LAB04-PC2...-PC20\USER 17 (90)', 'SQLQuery2.sql - LA...PC20\USER 17 (54)*', and 'SQLQuery1.sql - LA...PC20\USER 17 (54)*'. The query in the active tab is:

```
SELECT GD.NumGuia, SUM(GD.CantidadEnviada * GD.PrecioVenta) AS TotalEnviado  
FROM GUIA_DETALLE AS GD  
GROUP BY GD.NumGuia;
```

In the bottom pane, the 'Results' tab is selected, displaying the following table:

	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;
```

SQLQuery3.sql - LA...PC20\USER 17 (66)* => X LAB04-PC20\MSSQL...PERU - dbo.LINEA 3..sql - LAB04-PC2...-PC20\USER 17 (90) SQLQuery2.sql - LA...PC20\USER 17 (54)* SQLQuery1.sql - LA...PC20\USER 17 (53)*

```
SELECT OD.CodArticulo, SUM(OD.CantidadSolicitada) AS TotalSolicitado
FROM ORDEN_DETALLE AS OD
GROUP BY OD.CodArticulo;
```

100 %

Results Messages

	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
20	50	0

7. Contar órdenes únicas que incluyan cada artículo.

SQL

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

```
SQLQuery3.sql - LA...PC20\USER 17 (66)*  X  LAB04-PC20\MSSQL...PERU - dbo.LINEA      3..sql - LAB04-PC2...-PC20\USER 17 (90)  SQLQuery2.sql - LA...PC20\USER 17 (54)*  SQLQuery1.sql - LA...PC20\

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

100 %

Results Messages

CodArticulo	NumOrdenesUnicas
1	1
2	1
3	1
4	1
5	1
6	1
7	1
8	1
9	1
10	1
11	1
12	1
13	1
14	1
15	1
16	1
17	1
18	1
19	1
20	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;
```

```
!LQuery3.sql - LA...PC20\USER 17 (66)* + X LAB04-PC20\MSSQL...PERU - dbo.LINEA      3..sql - LAB04-PC2...PC20\USER 17 (90)) SC
SELECT AVG(DATEDIFF(DAY, FechaOrden, FechaIngreso)) AS PromedioDiasIngreso
FROM ORDEN_COMPRA
WHERE FechaIngreso IS NOT NULL;
```

0 %

Results Messages

PromedioDiasIngreso
4

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;
```

```
SQLQuery3.sql - LA...PC20\USER 17 (66)*  ▶ X LAB04-PC20\MSSQL...PERU - dbo.LINEA      3..sql - LAB04-PC2...-PC20\USER 17 (90)      SQLQuery2.sql - LA
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;
```

10 %

Results Messages

	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)

```

SQLQuery3.sql - LA...PC20\USER 17 (66)* -> LAB04-PC20\MSSQL...PERU - dbo.LINEA      3..sql - LAB04-PC2...-PC20\USER 17 (90))      SQLQuery2.sql - LA...PC20\USER 17 (54)*      SQLQuery1.sql - LA...PC20\US
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;

```

10 %

Results Messages

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

```

```

SQLQuery3.sql - LA...PC20\USER 17 (66)* -> LAB04-PC20\MSSQL...PERU - dbo.LINEA      3..sql - LAB04-PC2...-PC20\USER 17 (90))      SQLQuery2.sql - LA...PC20\USER 17 (54)*      SQLQuery1.sql - LA...PC20\USER 17 (53)*
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;

```

100 %

Results Messages

NomLinea	CantArticulos
Access Points	1
Adaptadores	1
AllInOne	1
Antivirus	1
Auriculares	1
Cableado	1
Cables	1
Cámaras IP	1
Chromebooks	1
Conectores	1
Discos Duros	1
Discos Externos	1
Enfriamiento	1
Estabilizadores	1
Fuentes Poder	1
Gabinetes	1
Gaming	1
Impresoras	1
Laptops	1

12. Mostrar CodLinea y StockTotal.

SQL

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

The screenshot shows the SQL Server Management Studio interface. In the top pane, there are four tabs: 'SQLQuery3.sql - LA...PC20\USER 17 (66)*', 'LAB04-PC20\MSSQL_PERU - dbo.LINEA 3.sql - LAB04-PC2...PC20\USER 17 (90)', 'SQLQuery2.sql - LA...PC20\USER 17 (54)*', and 'SQLQuery1.sql - LA...PC20\USER 17 (53)*'. The second tab contains the query code. The bottom pane shows the 'Results' tab with the query results:

	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;
```

SQL Server

```
SQLQuery3.sql - LA...PC20\USER 17 (66)*  LAB04-PC20\MSSQL...PERU - dbo.LINEA  3..sql - LAB04-PC2...-PC20\USER 17 (90)  SQLQuery2.sql - LA...PC20\USER 17 (54)*  SQLQue
SELECT NumOrden, SUM(PrecioCompra * CantidadRecibida) AS CostoTotal
FROM ORDEN_DETALLE
WHERE CantidadRecibida IS NOT NULL
GROUP BY NumOrden;
```

RA
E

Ales

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

14. Mostrar NumGuia y PromedioEnviado (Cantidad).

SQL

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

SQL Server

```
SQLQuery3.sql - LA...PC20\USER 17 (66)*  LAB04-PC20\MSSQL...PERU - dbo.LINEA  3..sql - LAB04-PC2...-PC20\USER 17 (90)  SQLQuery2.sql - LA...PC20\USER 17 (54)*  SQLQuery1.sql - LA...PC20\U
SELECT NumGuia, AVG(CantidadEnviada * 1.0) AS PromedioEnviado
FROM GUIA_DETALLE
GROUP BY NumGuia;
```

RA
E

ts

	NumGuia	PromedioEnviado
1	5001	4.000000
2	5002	18.333333
3	5003	9.333333
4	5004	5.000000
5	5005	3.000000
6	5006	4.000000
7	5007	2.000000
8	5008	4.500000
9	5009	2.000000
10	5010	4.000000

15. Contar proveedores agrupados por Ciudad.

SQL

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

The screenshot shows a SQL Server Management Studio window with four tabs at the top: 'SQLQuery3.sql - LA...PC20\USER 17 (66)*' (selected), 'LAB04-PC20\MSSQL...PERU - dbo.LINEA', '3..sql - LAB04-PC2...-PC20\USER 17 (90)', and 'SQLQuery2.sql - LA...PC20\USER 17 (54)*' and 'SQLQuery1.sql - LA...PC20\USER 17 (53)*'. The query in the selected tab is:

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

The results pane displays a table with 17 rows, showing the number of providers per city:

	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);
```

```
SQLQuery3.sql - LA...PC20\USER 17 (66)*  X LAB04-PC20\MSSQL...PERU - dbo.LINEA      3..sql - LAB04-PC2...-PC20\USER 17 (90))      SQLQuery2.sql - LA...PC20\USER 1
SELECT CAST(FechaOrden AS DATE) AS Dia, COUNT(NumOrden) AS NumOrdenes
FROM ORDEN_COMPRA
GROUP BY CAST(FechaOrden AS DATE);
```

100 %

Results Messages

	Dia	NumOrdenes
1	2025-10-01	1
2	2025-10-03	1
3	2025-10-05	1
4	2025-10-07	1
5	2025-10-10	1
6	2025-10-12	1
7	2025-10-15	1
8	2025-10-17	1
9	2025-10-20	1
10	2025-10-22	1

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;
```

```

SELECT A.CodLinea, SUM(GD.CantidadEnvilada * 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	10500.00
2	8700.00
3	6200.00
4	2000.00
5	1400.00
6	1500.00
7	1260.00
8	2200.00
9	4200.00
10	3500.00
11	4500.00
12	4800.00
13	2560.00
14	6300.00
15	1250.00
16	900.00
17	4600.00
18	1300.00
19	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;

```

```

SQLQuery3.sql - LA...PC20\USER.17 (66)* X LAB04-PC20\MSQL..PERU - dbo.LINEA 3..sql - LAB04-PC20\USER.17 (90) SQLQuery2.sql - LA...PC20\USER.17 (54)* SQLQuery1.sql - LA...PC20\USER.17 (53)*
SELECT A.CodArticulo, A.DescripcionArticulo, A.StockActual
FROM ARTICULO AS A
INNER JOIN (
    SELECT CodLinea, AVG(StockActual) = $j.0) AS PromedioStock
    FROM ARTICULO
    GROUP BY CodLinea
) AS T ON A.CodLinea = T.CodLinea
WHERE A.StockActual < T.PromedioStock;

```

100 %

	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	Tecaldo Logitech K120	100
12	12	Mouse Logitech M90	100
13	13	Auriculares Redragon Zeus	45
14	14	Parlantes Genius 2.0	80
15	15	Microfono 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;

```

```

SQLQuery4.sql - LA...PC13\USER 17 (67)* 3.sql - LAB04-PC1...-PC13\USER 17 (99) SQLQuery3.sql - LA...PC13\USER 17 (71) SQLQuery2.sql - LA...PC13\USER 17 (70)* SQLQuery1.sql - LA...PC13\USER 17 (68)*
SELECT P.CodProveedor, P.NomProveedor, COUNT(A.CodArticulo) AS CantidadArticulos
FROM PROVEEDOR AS P
INNER JOIN ARTICULO AS A ON P.CodProveedor = A.CodProveedor
GROUP BY P.CodProveedor, P.NomProveedor;

```

Results

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

20. Mostrar para cada estado sumar CantidadSolicitada.

SQL

```

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

```

```

SQLQuery4.sql - LA...PC13\USER 17 (67)* 3.sql - LAB04-PC1...-PC13\USER 17 (99) SQLQuery3.sql - LA...PC13\USER 17 (71) SQLQuery2.sql - LA...PC13\USER 17 (70)* SQLQuery1.sql - LA...PC13\USER 17 (68)*
SELECT Estado, SUM(CantidadSolicitada) AS TotalCantidadSolicitada
FROM ORDEN_DETALLE
GROUP BY Estado;

```

Results

Estado	TotalCantidadSolicitada
Pendiente	84
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;
```

The screenshot shows the SQL Server Management Studio interface. In the top bar, there are four tabs: 'SQLQuery4.sql - LA...PC13\USER 17 (67)*', '3.sql - LAB04-PC1...-PC13\USER 17 (99)', 'SQLQuery3.sql - LA...PC13\USER 17 (71)', 'SQLQuery2.sql - LA...PC13\USER 17 (70)*', and 'SQLQuery1.sql - LA...PC13\USER 17 (68)*'. The main window displays a query editor with the following code:

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

Below the query editor is a results grid showing the output of the query. The columns are 'CodArticulo', 'CodLinea', 'PrecioProveedor', and 'PosicionPorPrecio'. The data is as follows:

	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;

```

The screenshot shows a SQL Server Management Studio interface. In the top bar, there are tabs for 'SQLQuery4.sql - LA...PC13\USER 17 (67)*' (selected), '3..sql - LAB04-PC1...-PC13\USER 17 (99)', 'SQLQuery3.sql - LA...PC13\USER 17 (71)', 'SQLQuery2.sql - LA...PC13\USER 17 (70)*', and 'SQLQu'. The main area contains a query window with the following code:

```

SELECT
    NumOrden,
    SUM(PrecioCompra * CantidadRecibida) AS CostoOrden,
    RANK() OVER (ORDER BY SUM(PrecioCompra * CantidadRecibida) DESC) AS RankCosto
FROM ORDEN_DETALLE
GROUP BY NumOrden;

```

Below the code, the 'Results' tab is selected, showing a table with 10 rows of data:

	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;

```

The screenshot shows the SQL Server Management Studio interface with four tabs at the top: 'SQLQuery4.sql - LA...PC13\USER 17 (67)*', '3..sql - LAB04-PC1...-PC13\USER 17 (99)', 'SQLQuery3.sql - LA...PC13\USER 17 (71)', 'SQLQuery2.sql - LA...PC13\USER 17 (70)*', and 'SQLQuery1.sql - LA...PC13\USER 17 (68)*'. The main window displays the following T-SQL code:

```

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;

```

Below the code, the 'Results' tab is selected, showing the following data:

	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;
```

The screenshot shows a SQL Server Management Studio window. At the top, there are five tabs: 'SQLQuery4.sql - LA...PC13\USER 17 (67)*' (selected), '3..sql - LAB04-PC1...-PC13\USER 17 (99))', 'SQLQuery3.sql - LA...PC13\USER 17 (71))', 'SQLQuery2.sql - LA...PC13\USER 17 (70)*', and 'SQLQuery1.sql - LA...PC13\USER 17 (68))'. The main area contains the following SQL code:

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

Below the code, the results pane shows a table with 14 rows of data:

	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;
```

```
SQLQuery4.sql - LA...PC13\USER 17 (67)* 3..sql - LAB04-PC1...-PC13\USER 17 (99)) SQLQuery3.sql - LA...PC13\USER 17 (71)) SQLQuery2.sql - LA...PC13\USER 17 (70)) SQLQuery1.sql - LA...PC13\USER 17 (68))  
SELECT  
    CodProveedor,  
    PrecioProveedor,  
    LAG(PrecioProveedor, 1, 0) OVER (PARTITION BY CodProveedor ORDER BY CodArticulo) AS PrecioAnterior  
FROM ARTICULO  
ORDER BY CodProveedor, CodArticulo;
```

Results

	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;
```

```

SQLQuery4.sql - LA...PC13\USER 17 (67)* 3.sql - LAB04-PC1...PC13\USER 17 (99) SQLQuery3.sql - LA...PC13\USER 17 (71)) SQLQuery2.sql - LA...PC13\USER 17 (70)* SQLQuery1
SELECT
    CodArticulo,
    DescripcionArticulo,
    CodLinea,
    StockActual,
    SUM(StockActual) OVER (PARTITION BY CodLinea) AS CantidadTotalPorLinea
FROM ARTICULO;

```

100 % <

Results Messages

	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	Tecclado 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;
```

SQLQuery4.sql - LA...PC13\USER 17 (67)* 3..sql - LAB04-PC1...-PC13\USER 17 (99)) SQLQuery3.sql - LA...PC13\USER 17 (71)) SQLQuery2.sql - LA...PC13\USER 17 (70)*

```

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;

```

100 %

Results Messages

	CodProveedor	MontoProveedor	PorcentajeDelTotal
1	1	81000.00	8.503133562183100
2	2	62500.00	6.561059847363503
3	3	48000.00	5.038893962775170
4	4	48000.00	5.038893962775170
5	5	84000.00	8.818064434856548
6	6	46400.00	4.870930830682665
7	7	34500.00	3.621705035744654
8	8	32000.00	3.359262641850113
9	9	43500.00	4.566497653764998
10	10	33000.00	3.464239599407929
11	11	5000.00	0.524884787789080
12	12	4000.00	0.419907830231264
13	13	9900.00	1.039271879822378
14	14	6000.00	0.629861745346896

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;

```

```

SQLQuery4.sql - LA...PC13\USER 17 (67)* 3..sql - LAB04-PC1...PC13\USER 17 (99) SQLQuery3.sql - LA...PC13\USER 17 (71) SQLQuery2.sql - LA...PC13\USER 17 (70)* SQLQuery1.sql - LA...P
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;

```

SQLQuery4.sql - LA...PC13\USER 17 (67)* 3..sql - LAB04-PC1...-PC13\USER 17 (99)) SQLQuery3.sql - LA...PC13\USER 17 (71)) SQLQuery2.sql - LA...PC13\USER 17 (70)* SQLQuery1.sql - LA...

```

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;

```

Results Messages

	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,
        VentasAgregadas.VentaGuia
    FROM GUIA_ENVIO AS GE
    JOIN GUIA_DETALLE AS GD ON GE.NumGuia = GD.NumGuia
    JOIN TIENDA AS T ON GE.CodTienda = T.CodTienda
    JOIN VENTASAGREGADAS AS VA ON GE.NumGuia = VA.NumGuia
)

```

```

        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;

```

The screenshot shows the SQL Server Management Studio interface. The query is written in the top pane, and the results are displayed in the bottom pane.

```

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;

```

The results grid displays the following data:

	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;
```

```

SQLQuery4.sql - LA...PC13\USER 17 (67)* 3.sql - LAB04-PC1...PC13\USER 17 (99) SQLQuery3.sql - LA...PC13\USER 17 (71) SQLQuery2.sql - LA...PC13\USER 17 (70)* 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;

```

100 %

	Dia	CodTienda_1	CodTienda_2
1	2025-10-05	8	NULL
2	2025-10-07	NULL	55
3	2025-10-09	NULL	NULL
4	2025-10-10	NULL	NULL
5	2025-10-12	NULL	NULL
6	2025-10-14	NULL	NULL
7	2025-10-15	NULL	NULL
8	2025-10-17	NULL	NULL
9	2025-10-18	NULL	NULL
10	2025-10-20	NULL	NULL

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;

```

SQLQuery4.sql - LA...PC13\USER 17 (67)* 3..sql - LAB04-PC1...PC13\USER 17 (99)) SQLQuery3.sql - LA...PC13\USER 17 (71)) SQLQuery2.sql - LA...PC13\USER 17 (70))*

```

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;

```

00 %

Results Messages

	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 AnoMes 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;

```

The screenshot shows the SQL Server Management Studio interface with the following details:

- Toolbar:** Standard SSMS toolbar with icons for Execute, Save, Copy, Paste, and other database operations.
- Object Explorer:** Shows the current connection path: SQLQuery4.sql - LA...PC13\USER 17 (67)*.
- Results Grid:** Displays the query results in a tabular format. The columns are AnoMes, Tienda_1, and Tienda_2. The data row shows AnoMes as 2025-10, Tienda_1 as 19200.00, and Tienda_2 as 9600.00.
- Status Bar:** Shows the zoom level as 100%.

AnoMes	Tienda_1	Tienda_2	
1	2025-10	19200.00	9600.00

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;
```

```
SQLQuery4.sql - LA...PC13\USER 17 (67)* 3..sql - LAB04-PC1...-PC13\USER 17 (99) SQLQuery3.sql - LA...PC13\USER 17 (71) SQLQuery2.sql - LA...PC13\USER 17 (70)*
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;
```

100 % < >

Results Messages

	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;

```

The screenshot shows the SQL Server Management Studio interface. The top pane displays the T-SQL code for generating a dynamic pivot table. The bottom pane shows the results of the query execution, which is a table with 16 rows, each containing a product description and two columns: Botella and Caja, both of which are currently null.

	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;

```

```

SQLQuery4.sql - LA...PC13\USER 17 (67)* <--> 3.sql - LAB04-PC1...PC13\USER 17 (99)      SQLQuery3.sql - LA...PC13\USER 17 (71)      SQLQuery2.sql - LA...PC13\USER 17 (70)*      SQLQuery1.sql - LA...PC13\USER 17 (68)*
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;

```

100% <--> 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	10	NULL																														
2	2	3	NULL	10	NULL																														
3	10	NULL	20	NULL																															
4	11	NULL	25	NULL																															
5	12	NULL	30	NULL																															
6	13	NULL	5	NULL	35	NULL																													
7	15	NULL	3	NULL	40	NULL																													
8	17	NULL	45	NULL																															
9	20	NULL	50	NULL																															
10	21	NULL	55	NULL																															
11	24	NULL	60	NULL																															
12	25	NULL	65	NULL	2																														
13	26	NULL	70	NULL	8																														
14	40	NULL	75	NULL																															
15	41	NULL	80	NULL																															
16	42	NULL	85	NULL																															
17	47	NULL	90	NULL																															
18	48	NULL	95	NULL																															

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						
<table border="1"> <thead> <tr> <th>Mes</th> <th>Transp_1</th> <th>Transp_2</th> </tr> </thead> <tbody> <tr> <td>2025-10</td> <td>19200.00</td> <td>9600.00</td> </tr> </tbody> </table>	Mes	Transp_1	Transp_2	2025-10	19200.00	9600.00	
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;

```

SQLQuery4.sql - LA...PC13\USER 17 (67)* 3..sql - LAB04-PC1...-PC13\USER 17 (99)) SQLQuery3.sql - LA...PC13\USER 17 (71)) SQLQuery2.sql - LA...PC13\USER 17 (70)* SQLQ

```

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;

```

100 %

Results Messages

	Poca	Media	Mucha
1	50	0	0

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;

```

The screenshot shows the SQL Server Management Studio interface with four tabs at the top: 'SQLQuery4.sql - LA...PC13\USER 17 (67)*' (active), '3..sql - LAB04-PC1...-PC13\USER 17 (99)', 'SQLQuery3.sql - LA...PC13\USER 17 (71)', and 'SQLQuery2.sql - LA...PC13\USER 17 (70)*'. The query window contains the T-SQL code provided above. Below the tabs is a progress bar at 0%. The results window displays a table with columns 'CodArticulo', '2024', and '2025'. The data is as follows:

CodArticulo	2024	2025
1	NULL	10500.00
2	NULL	8700.00
10	NULL	6200.00
11	NULL	2000.00
12	NULL	1400.00
13	NULL	1500.00
15	NULL	1260.00
17	NULL	2200.00
20	NULL	4200.00
21	NULL	3500.00
24	NULL	4500.00
25	NULL	4800.00
26	NULL	2560.00
40	NULL	6300.00
41	NULL	1250.00
42	NULL	900.00
47	NULL	4600.00
48	NULL	1300.00
49	NULL	1100.00

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;
```

The screenshot shows the SQL Server Management Studio interface. In the top tab bar, there are five tabs: 'SQLQuery4.sql - LA...PC13\USER 17 (67)*' (selected), '3.sql - LAB04-PC1...-PC13\USER 17 (99)', 'SQLQuery3.sql - LA...PC13\USER 17 (71)', 'SQLQuery2.sql - LA...PC13\USER 17 (70)*', and 'SQLQuery1.sql - LA...PC13\US'. The query in the editor is:

```
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;
```

In the results pane below, the output is displayed in a table:

Mes	Tienda_1	Tienda_2
1 2025-10	19200.00	9600.00

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;
```

```
SQLQuery4.sql - LA...PC13\USER 17 (67)* -> X 3..sql - LAB04-PC1...-PC13\USER 17 (99) SQLQuery3.sql - LA...PC13\USER 17 (71) SQLQuery2.sql - LA...PC13\USER 17 (70)* SQLQuery1.sql - LA...PC13\USER 17 (70)*
SELECT CodLinea, COUNT(CodArticulo) AS CantArticulos
FROM ARTICULO
GROUP BY CodLinea
HAVING COUNT(CodArticulo) > 0;
```

100 %

Results Messages

CodLinea	CantArticulos
1	1
2	1
3	1
4	1
5	1
6	1
7	1
8	1
9	1
10	1
11	1
12	1
13	1
14	1
15	1

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;
```

The screenshot shows a SQL Server Management Studio window with multiple tabs at the top. The active tab is 'SQLQuery4.sql - LA...PC13\USER 17 (67)*'. The query in the editor is:

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

The results grid below shows the output:

	CodProveedor	MontoTotal
1	1	81000.00
2	2	62500.00
3	5	84000.00

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;
```

```

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 %

	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

```

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

```

The screenshot shows the SSMS interface. In the top query window, a T-SQL SELECT statement is displayed:

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

In the bottom results pane, the tabs "Results" and "Messages" are visible. The "Results" tab is selected, showing an empty table structure with columns "CodArticulo" and "TotalSolicitado".

45. Mostrar CodTransportista y CantGuías >= 5.

SQL

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

The screenshot shows a SQL Server Management Studio window. In the top bar, there are tabs for 'Execute' and several other queries like 'SQLQuery3.sql', 'SQLQuery2.sql', and 'SQLQuery'. The main area contains a query window with the following T-SQL code:

```
SELECT CodTransportista, COUNT(NumGuia) AS CantGuias
FROM GUIA_ENVIO
GROUP BY CodTransportista
HAVING COUNT(NumGuia) >= 0;
```

Below the query window is a results grid titled 'Results'. It has two columns: 'CodTransportista' and 'CantGuias'. The data is as follows:

	CodTransportista	CantGuias
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

46. Mostrar líneas donde $\text{SUM}(\text{StockActual}) < \text{SUM}(\text{StockMinimo}) * \text{NumArticulosPorLinea}$ – 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;
```

A screenshot of the SQL Server Management Studio interface. The top bar shows multiple tabs: 'Execute' (selected), 'SQLQuery4.sql - LA...PC13(USER 17 (67))', '3..sql - LABD4-PC1...PC13(USER 17 (99))', 'SQLQuery3.sql - LA...PC13(USER 17 (71))', 'SQLQuery2.sql - LA...PC13(USER 17 (70))', and 'SQLQuery1.sql - LA...PC13(USER 17 (68))'. The main area contains a query window with the following T-SQL code:

```
-- 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;
```

The results grid shows the following data:

	CodLinea	Stock Total
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

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

SQL

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

A screenshot of the SQL Server Management Studio interface. The top bar shows several tabs: 'New Query', 'Execute', 'SQLQuery4.sql - LA...PC13\USER 17 (67)*', '3.sql - LAB04-PC1...-PC13\USER 17 (99)', 'SQLQuery3.sql - LA...PC13\USER 17 (71)*', 'SQLQuery2.sql - LA...PC13\USER 17 (70)*', and 'SQLQuery1.sql - LA...PC13\USER 17 (68)*'. The main area contains a query window with the following SQL code:

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

The results grid shows the following data:

CodProveedor	PrecioMaximo
1	1800.00
2	2500.00
3	1600.00
4	1200.00
5	4200.00
6	5800.00
7	2300.00
8	1600.00

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;
```

The screenshot shows a SQL Server Management Studio interface. At the top, there are several tabs labeled: 'SQLQuery4.sql - LA...PC13\USER 17 (67)*' (active), 'SQLQuery3.sql - LA...PC13\USER 17 (71)', 'SQLQuery2.sql - LA...PC13\USER 17 (70)*', and 'SQLQuery1.sql - LA...PC13\USER 17 (68)'. The active query window contains the following SQL code:

```
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;
```

The results pane shows a table with three columns: 'CodTienda', 'PromedioEnviado', and 'CantidadGuias'. There is one row of data.

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;
```

The screenshot shows a SQL Server Management Studio interface. In the top query window, a SELECT statement is written to find the maximum and minimum prices for each product line, filtering for a price range greater than 20:

```
SELECT CodLinea, MAX(PrecioProveedor) AS PrecioMaximo, MIN(PrecioProveedor) AS PrecioMinimo
FROM ARTICULO
GROUP BY CodLinea
HAVING (MAX(PrecioProveedor) - MIN(PrecioProveedor)) > 20;
```

The results pane below shows the output of the query:

CodLinea	PrecioMaximo	PrecioMinimo

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;
```

Execute ✓ SQLQuery4.sql - LA...PC13\USER 17 (67)* 3.sql - LAB04-PC1...PC13\USER 17 (99) SQLQuery3.sql - LA...PC13\USER 17 (71) SQLQuery2.sql - LA...PC13\USER 17 (70)* SQLQuery1.sql - LA...PC13\USER 17 (70)

```
SELECT CodProveedor, COUNT(CodArticulo) AS CantArticulos, AVG(StockActual) AS PromedioStock
FROM ARTICULO
GROUP BY CodProveedor
HAVING AVG(StockActual) < 10 AND COUNT(CodArticulo) > 0;
```

100 %

Results Messages

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