Universidad Peruana Los Andes

Facultad de Ingeniería

Escuela Profesional de Ingeniería de Sistemas y Computación

Escuela Profesional de Ingeniería de Sistemas y Computación

Curso: Base de datos II

Docente: Raul Fernandez Bejarano

Estudiante: Sarmiento Mosquera Yeims Abraham

V Ciclo - Código: s03807f

Huancayo – 2025

Desarrollo de los enunciados de la semana 9

- ENUNCIADO 1

```
-- EJERCICIO 1
-- ENUNCIADO: MOSTRAR CODARTICULO, DESCRIPCIONARTICULO Y VALORINVENTARIO
-- CONSULTA SQL:

SELECT
A.CodArticulo,
A.DescripcionArticulo,
CAST(A.StockActual * CAST(A.PrecioProveedor AS DECIMAL(18,2)) AS DECIMAL (18,2)) AS
ValorInventario
FROM ARTICULO A

-- EXPLICACION: MULTIPLICA STOCK POR PRECIO POR FILA; SE USA EL CAST PARA PRECISION
```

RESULTADO

110 % — No se encontraron problemas.

Resultados | Mensajes

	CodArticulo	DescripcionArticulo	ValorInventario
1	1	Transistor NPN BC547	250.00
2	2	Resistencia 220 Ohm 1/4W	100.00
3	3	Condensador Electrolítico 10uF	160.00
4	4	Microcontrolador ATmega328P	500.00
5	5	Sensor de temperatura LM35	360.00
6	6	Diodo 1N4007	135.00
7	7	LED rojo 5mm	75.00
8	8	Potenciómetro 10K	320.00
9	9	Display LCD 16x2	350.00
10	10	Relé 5V 10A	250.00
11	11	Pulsador táctil	180.00
12	12	Cristal 16MHz	175.00
13	13	Zócalo DIP 28 pines	140.00
14	14	Puente rectificador KBPC5010	240.00
15	15	Sensor de humedad DHT11	270.00
16	16	Regulador de voltaje LM7805	300.00
17	17	Motor DC 6V	198.00
18	18	Driver L298N	330.00
19	19	Placa Arduino UNO	375.00
20	20	Módulo Bluetooth HC-05	323.00
21	21	Sensor infrarrojo	242.00
22	22	Buzzer activo 5V	210.00
23	23	Módulo WiFi ESP8266	380.00
24	24	Sensor de gas MQ-2	364.00
25	25	Módulo RFID RC522	320.00
26	26	Sensor de luz LDR	140.00

- ENUNCIADO 2

```sql
-- EJERCICIO 2
-- ENUNCIADO : CALCULAR EL TOTAL MONETARIO DEL INVENTARIO
--CONSULTA SQL :

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

/* EXPLICACION: SE RECORRE TODA LA TABLA ARTICULO, SE MULTIPLICA EL PRECIOPROVEEDOR
 POR EL STOCK ACTUAL Y TODO ELLO SE SUMA PARA DEVOLVERLO EN EL TOTALINVENTARIOMONETARIO
*/

```

RESULTADO

110 % No se encontraron problemas.

Resultados Mensajes

	TotalInventarioMonetario
1	13066,00

- ENUNCIADO 3

```sql
-- EJERCICIO 3

SELECT CodLinea, AVG(PrecioProveedor) AS PrecioPromedio
FROM ARTICULO
GROUP BY CodLinea;
/*
EXPLICACION: se obtiene el precio promedio de los articulos
*/
```

110 % No se encontraron problemas.

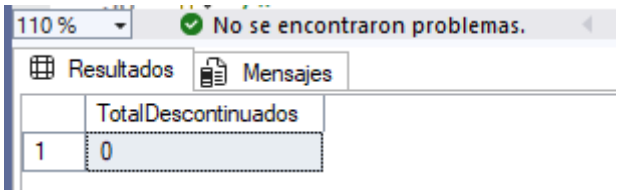
Resultados Mensajes

	CodLinea	PrecioPromedio
1	1	0,50
2	2	0,10
3	3	0,20
4	4	2,50
5	5	1,20
6	6	0,15
7	7	0,05
8	8	0,80
9	9	3,50
10	10	1,00
11	11	0,30
12	12	0,25
13	13	0,40
14	14	2,00
15	15	1,50
16	16	0,60
17	17	2,20
18	18	3,00
19	19	5,00
20	20	3,80

- ENUNCIADO 4

```
41  -- EJERCICIO 4
42
43  SELECT COUNT(*) AS TotalDescontinuados
44  FROM ARTICULO
45  WHERE Descontinuado = 1;
46
47  /*
48  EXPLICACION: se cuentan los articulos descontinuados
49  */
```

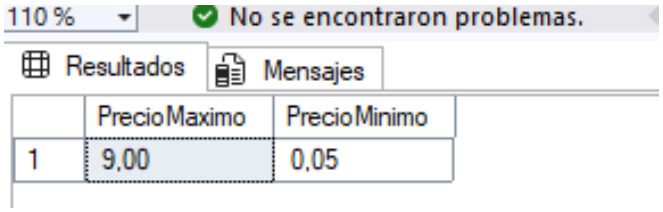
RESULTADO



- ENUNCIADO 5

```
51  -- EJERCICIO 5
52
53  SELECT MAX(PrecioProveedor) AS PrecioMaximo,
54         MIN(PrecioProveedor) AS PrecioMinimo
55  FROM ARTICULO;
56  /*
57  EXPLICACION: Muestra precio maximo y minimo
58  */
```

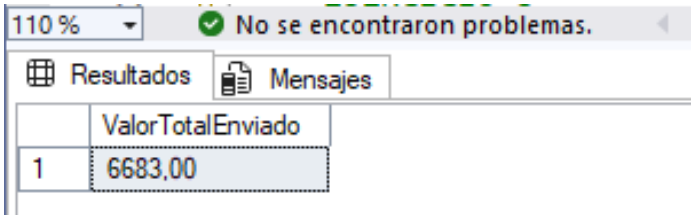
RESULTADO



- ENUNCIADO 6

```
60  -- EJERCICIO 6
61
62  SELECT SUM(PrecioVenta * CantidadEnviada) AS ValorTotalEnviado
63  FROM GUIA_DETALLE;
64  /*
65  EXPLICACION: muestra el valor multiplicar y sumar los datos de las guia
66  */
```

RESULTADO



- ENUNCIADO 7

```sql
--EJERCICIO 7

SELECT CodArticulo, SUM(CantidadSolicitada) AS TotalSolicitado
FROM ORDEN_DETALLE
GROUP BY CodArticulo;
/*
EXPLICACION: muestra el total solicitado de cada articulo
*/
```

RESUTADO

110 % — No se encontraron problemas.

Resultados | Mensajes

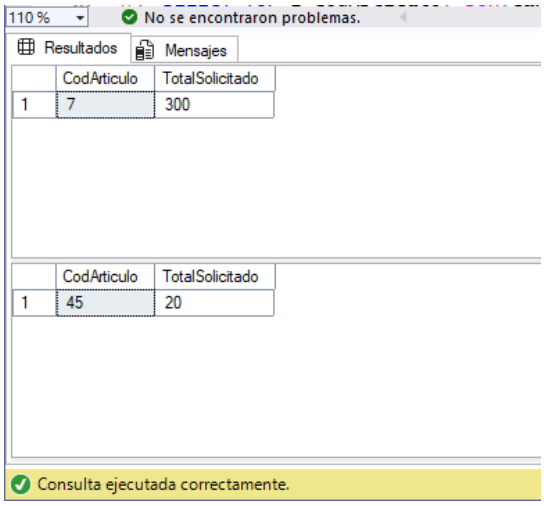
	CodArticulo	TotalSolicitado
1	1	100
2	2	200
3	3	150
4	4	80
5	5	120
6	6	180
7	7	300
8	8	90
9	9	60
10	10	110
11	11	130
12	12	140
13	13	70
14	14	50

Consulta ejecutada correctamente.

- ENUNCIADO 8

```sql
-- EJERCICIO 8

SELECT TOP 1 CodArticulo, SUM(CantidadSolicitada) AS TotalSolicitado
FROM ORDEN_DETALLE
GROUP BY CodArticulo
ORDER BY TotalSolicitado DESC;

SELECT TOP 1 CodArticulo, SUM(CantidadSolicitada) AS TotalSolicitado
FROM ORDEN_DETALLE
GROUP BY CodArticulo
ORDER BY TotalSolicitado ASC;

/*
EXPLICACION: cuenta ordenes unicas de cada producto
*/
```

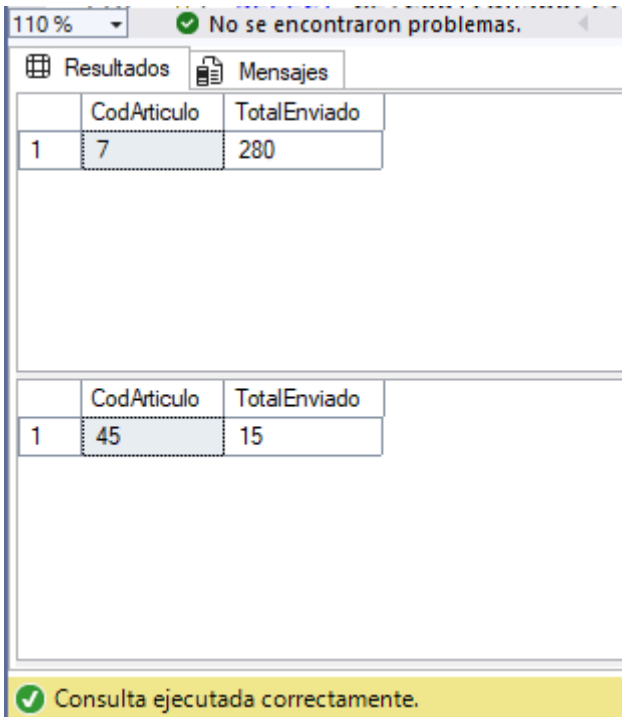
RESULTADO



- ENUNCIADO 9

```sql
-- EJERCICIO 9

SELECT TOP 1 CodArticulo, SUM(CantidadEnviada) AS TotalEnviado
FROM GUIA_DETALLE
GROUP BY CodArticulo
ORDER BY TotalEnviado DESC;

SELECT TOP 1 CodArticulo, SUM(CantidadEnviada) AS TotalEnviado
FROM GUIA_DETALLE
GROUP BY CodArticulo
ORDER BY TotalEnviado ASC;
/*
EXPLICACION: sunma cantidades enviadas por todas las ordenes de la fecha de ingreso
*/
```

RESULTADO



- ENUNCIADO 10

```sql
-- EJERCICIO 10

SELECT GE.CodTransportista, SUM(GD.CantidadEnviada) AS TotalEnviado
FROM GUIA_ENVIO GE
JOIN GUIA_DETALLE GD ON GE.NumGuia = GD.NumGuia
GROUP BY GE.CodTransportista;
/*
EXPLICACION: suma las cantidades enviadas por transportista
*/
```

RESULTADO

110 % — No se encontraron problemas.

Resultados | Mensajes

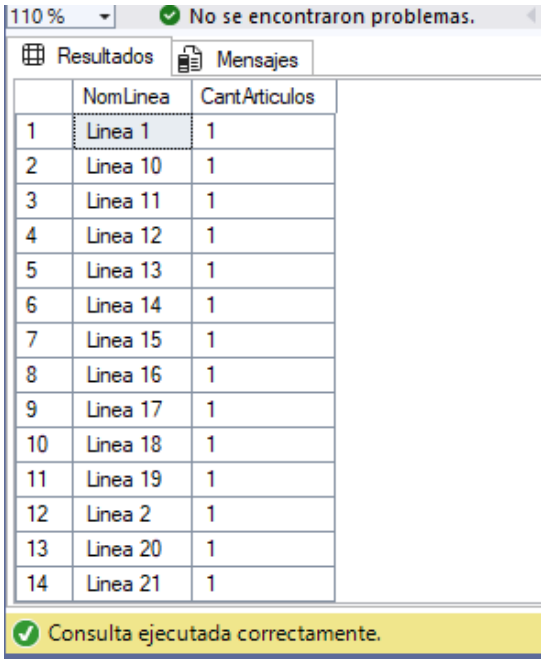
	CodTransportista	TotalEnviado
1	1	80
2	2	180
3	3	140
4	4	60
5	5	100
6	6	160
7	7	280
8	8	70
9	9	50
10	10	90
11	11	120
12	12	130
13	13	60
14	14	45

Consulta ejecutada correctamente.

- ENUNCIADO 11

```sql
-- EJERCICIO 11

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

/*
EXPLICACION: Cuenta cuántos artículos hay por cada línea de productos.
*/
```

RESULTADO


110 % No se encontraron problemas.

Resultados Mensajes

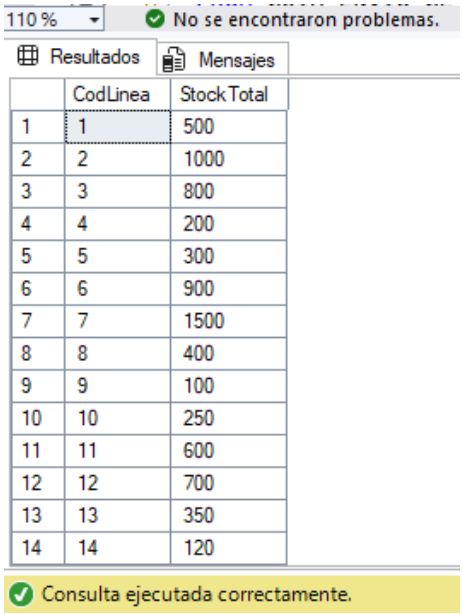
	NomLinea	CantArticulos
1	Linea 1	1
2	Linea 10	1
3	Linea 11	1
4	Linea 12	1
5	Linea 13	1
6	Linea 14	1
7	Linea 15	1
8	Linea 16	1
9	Linea 17	1
10	Linea 18	1
11	Linea 19	1
12	Linea 2	1
13	Linea 20	1
14	Linea 21	1

Consulta ejecutada correctamente.

- ENUNCIADO 12

```
130   -- EJERCICIO 12
131
132   SELECT CodLinea, SUM(StockActual) AS StockTotal
133   FROM ARTICULO
134   GROUP BY CodLinea;
135   /*
136   EXPLICACION: Suma el stock actual por cada línea de producto.
137   */
```

RESULTADO


110 % No se encontraron problemas.

Resultados Mensajes

	CodLinea	StockTotal
1	1	500
2	2	1000
3	3	800
4	4	200
5	5	300
6	6	900
7	7	1500
8	8	400
9	9	100
10	10	250
11	11	600
12	12	700
13	13	350
14	14	120

Consulta ejecutada correctamente.

- ENUNCIADO 13

```sql
-- EJERCICIO 13

SELECT GE.NumGuia, SUM(A.PrecioProveedor * GD.CantidadEnviada) AS CostoTotal
FROM GUIA_ENVIO GE
JOIN GUIA_DETALLE GD ON GE.NumGuia = GD.NumGuia
JOIN ARTICULO A ON GD.CodArticulo = A.CodArticulo
GROUP BY GE.NumGuia;
/*
EXPLICACION: Calcula el costo total por orden de compra.
*/
```

RESULTADO

110 % — No se encontraron problemas.

Resultados | Mensajes

	NumGuia	CostoTotal
1	1	40,00
2	2	18,00
3	3	28,00
4	4	150,00
5	5	120,00
6	6	24,00
7	7	14,00
8	8	56,00
9	9	175,00
10	10	90,00
11	11	36,00
12	12	32,50
13	13	24,00
14	14	90,00

Consulta ejecutada correctamente.

- ENUNCIADO 14

```sql
-- EJERCICIO 14

SELECT NumGuia, AVG(CantidadEnviada) AS PromedioEnviado
FROM GUIA_DETALLE
GROUP BY NumGuia;
/*
EXPLICACION: Promedio de unidades enviadas por cada guía.
*/
```

RESULTADO

110 % — No se encontraron problemas.

Resultados | Mensajes

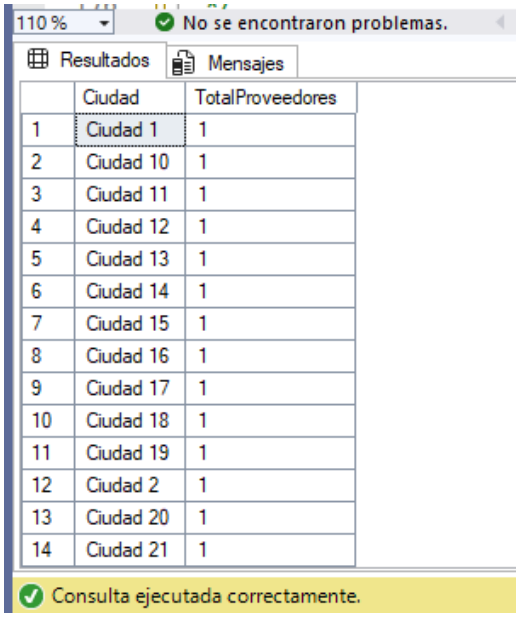
	NumGuia	PromedioEnviado
1	1	80
2	2	180
3	3	140
4	4	60
5	5	100
6	6	160
7	7	280
8	8	70
9	9	50
10	10	90
11	11	120
12	12	130
13	13	60
14	14	45

Consulta ejecutada correctamente.

- ENUNCIADO 15

```sql
-- EJERCICIO 15

SELECT Ciudad, COUNT(CodProveedor) AS TotalProveedores
FROM PROVEEDOR
GROUP BY Ciudad;
/*
EXPLICACION: Cuenta cuántos proveedores hay en cada ciudad.
*/
```

RESULTADO



110 % — No se encontraron problemas.

Resultados | Mensajes

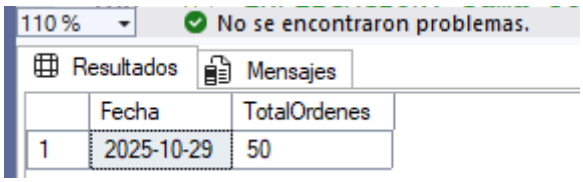
	Ciudad	TotalProveedores
1	Ciudad 1	1
2	Ciudad 10	1
3	Ciudad 11	1
4	Ciudad 12	1
5	Ciudad 13	1
6	Ciudad 14	1
7	Ciudad 15	1
8	Ciudad 16	1
9	Ciudad 17	1
10	Ciudad 18	1
11	Ciudad 19	1
12	Ciudad 2	1
13	Ciudad 20	1
14	Ciudad 21	1

Consulta ejecutada correctamente.

- ENUNCIADO 16

```sql
-- EJERCICIO 16

SELECT CONVERT(DATE, FechaOrden) AS Fecha, COUNT(NumOrden) AS TotalOrdenes
FROM ORDEN_COMPRA
GROUP BY CONVERT(DATE, FechaOrden);
/*
EXPLICACION: Cuenta las órdenes por fecha, ignorando la hora.
*/
```

RESULTADO



110 % — No se encontraron problemas.

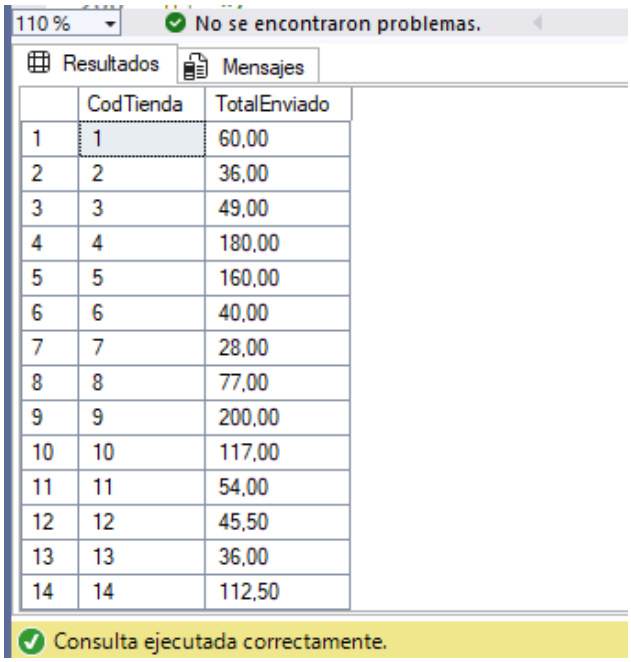
Resultados | Mensajes

	Fecha	TotalOrdenes
1	2025-10-29	50

- ENUNCIADO 17

```sql
178  -- EJERCICIO 17
179
180  SELECT GE.CodTienda, SUM(GD.CantidadEnviada * GD.PrecioVenta) AS TotalEnviado
181  FROM GUIA_ENVIO GE
182  JOIN GUIA_DETALLE GD ON GE.NumGuia = GD.NumGuia
183  GROUP BY GE.CodTienda;
184  /*
185  EXPLICACION: Suma el valor total enviado por cada tienda.
186  */
```

RESULTADO


110 %
No se encontraron problemas.
Resultados
Mensajes

	CodTienda	TotalEnviado
1	1	60,00
2	2	36,00
3	3	49,00
4	4	180,00
5	5	160,00
6	6	40,00
7	7	28,00
8	8	77,00
9	9	200,00
10	10	117,00
11	11	54,00
12	12	45,50
13	13	36,00
14	14	112,50

Consulta ejecutada correctamente.

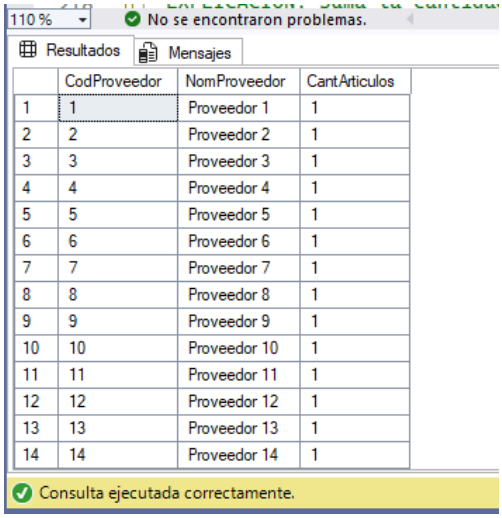
- ENUNCIADO 18

```sql
-- EJERCICIO 18

SELECT A.CodArticulo, A.CodLinea, A.StockActual
FROM ARTICULO A
JOIN (
    SELECT CodLinea, AVG(StockActual) AS PromedioStock
    FROM ARTICULO
    GROUP BY CodLinea
) AS PromedioPorLinea ON A.CodLinea = PromedioPorLinea.CodLinea
WHERE A.StockActual < PromedioPorLinea.PromedioStock;
/*
EXPLICACION: Filtra artículos con stock menor al promedio de su línea.
*/
```

- ENUNCIADO 19

```sql
-- EJERCICIO 19

SELECT P.CodProveedor, P.NomProveedor, COUNT(A.CodArticulo) AS CantArticulos
FROM PROVEEDOR P
JOIN ARTICULO A ON P.CodProveedor = A.CodProveedor
GROUP BY P.CodProveedor, P.NomProveedor;
/*
EXPLICACION: Muestra cuántos artículos tiene cada proveedor.
*/
```

RESULTADO

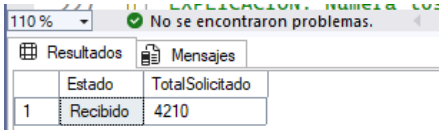


	CodProveedor	NomProveedor	CantArticulos
1	1	Proveedor 1	1
2	2	Proveedor 2	1
3	3	Proveedor 3	1
4	4	Proveedor 4	1
5	5	Proveedor 5	1
6	6	Proveedor 6	1
7	7	Proveedor 7	1
8	8	Proveedor 8	1
9	9	Proveedor 9	1
10	10	Proveedor 10	1
11	11	Proveedor 11	1
12	12	Proveedor 12	1
13	13	Proveedor 13	1
14	14	Proveedor 14	1

- ENUNCIADO 20

```sql
-- EJERCICIO 20

SELECT Estado, SUM(CantidadSolicitada) AS TotalSolicitado
FROM ORDEN_DETALLE
GROUP BY Estado;
/*
EXPLICACION: Suma la cantidad solicitada agrupada por estado de orden.
*/
```

RESULTADO

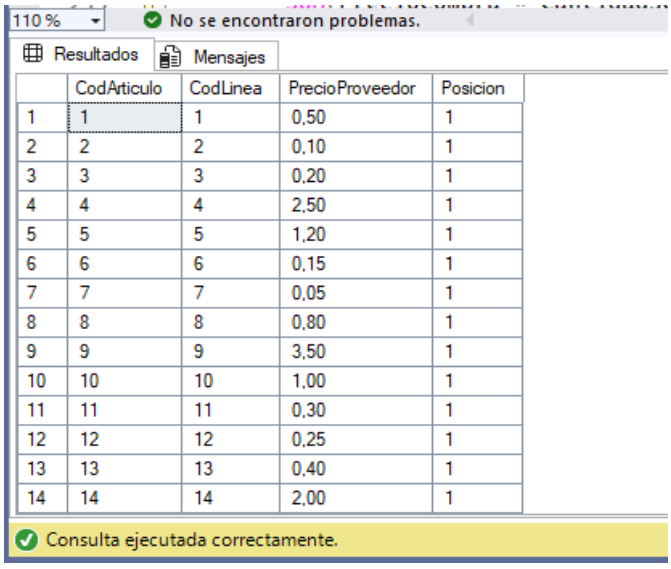


	Estado	TotalSolicitado
1	Recibido	4210

- ENUNCIADO 21

```sql
-- EJERCICIO 21

SELECT CodArticulo, CodLinea, PrecioProveedor,
       ROW_NUMBER() OVER (PARTITION BY CodLinea ORDER BY PrecioProveedor DESC) AS Posicion
FROM ARTICULO;
/*
EXPLICACION: Numera los artículos dentro de cada línea según su precio (mayor primero).
*/
```

RESULTADO



	CodArticulo	CodLinea	PrecioProveedor	Posicion
1	1	1	0,50	1
2	2	2	0,10	1
3	3	3	0,20	1
4	4	4	2,50	1
5	5	5	1,20	1
6	6	6	0,15	1
7	7	7	0,05	1
8	8	8	0,80	1
9	9	9	3,50	1
10	10	10	1,00	1
11	11	11	0,30	1
12	12	12	0,25	1
13	13	13	0,40	1
14	14	14	2,00	1

- ENUNCIADO 22

```sql
-- EJERCICIO 22

SELECT NumOrden,
       SUM(PrecioCompra * CantidadSolicitada) AS CostoTotal,
       RANK() OVER (ORDER BY SUM(PrecioCompra * CantidadSolicitada) DESC) AS RankCosto
FROM ORDEN_DETALLE
GROUP BY NumOrden;
/*
EXPLICACION: Calcula el costo total por orden y le asigna un ranking según el monto.
*/
```

RESULTADO

Resultados | Mensajes

	NumOrden	CostoTotal	RankCosto
1	9	216,00	1
2	4	208,00	2
3	23	205,00	3
4	44	200,00	4
5	25	198,00	5
6	28	198,00	5
7	49	190,00	7
8	24	188,50	8
9	48	183,00	9
10	45	182,00	10
11	31	180,00	11
12	29	178,50	12
13	19	178,50	12
14	43	176,00	14

Consulta ejecutada correctamente.

- ENUNCIADO 23

```sql
-- EJERCICIO 23

SELECT Fecha, TotalDia,
       SUM(TotalDia) OVER (ORDER BY Fecha) AS AcumuladoVentas
FROM (
    SELECT CAST(FechaSalida AS DATE) AS Fecha,
           SUM(PrecioVenta * CantidadEnviada) AS TotalDia
    FROM GUIA_ENVIO GE
    JOIN GUIA_DETALLE GD ON GE.NumGuia = GD.NumGuia
    GROUP BY CAST(FechaSalida AS DATE)
) AS VentasPorDia;
/*
EXPLICACION: Muestra el total de ventas por día y su acumulado cronológico.
*/
```

RESULTADO

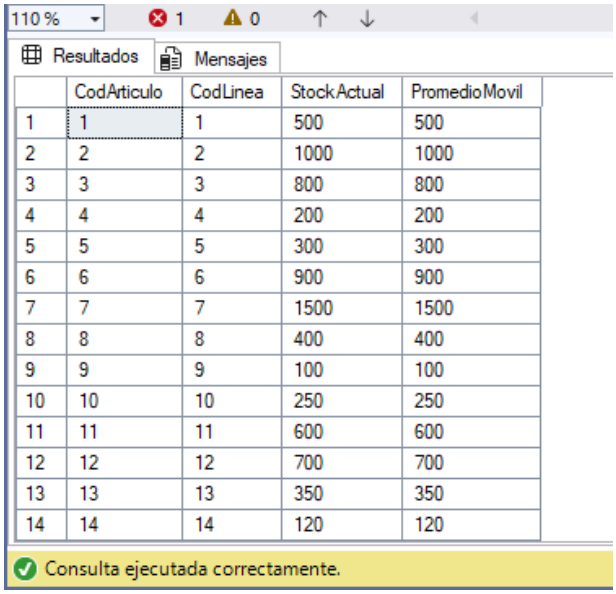
Resultados | Mensajes

	Fecha	TotalDia	AcumuladoVentas
1	2025-10-29	6683,00	6683,00

- ENUNCIADO 24

```sql
-- EJERCICIO 24

SELECT CodArticulo, CodLinea, StockActual,
       AVG(StockActual) OVER (PARTITION BY CodLinea ORDER BY CodArticulo ROWS BETWEEN 1 PRECEDING AND 1 FOLLOWING) AS PromedioMovil
FROM ARTICULO;
/*
EXPLICACION: Promedio móvil de stock por línea, considerando artículo anterior y siguiente.
*/
```

RESULTADO



110 % 1 0

Resultados Mensajes

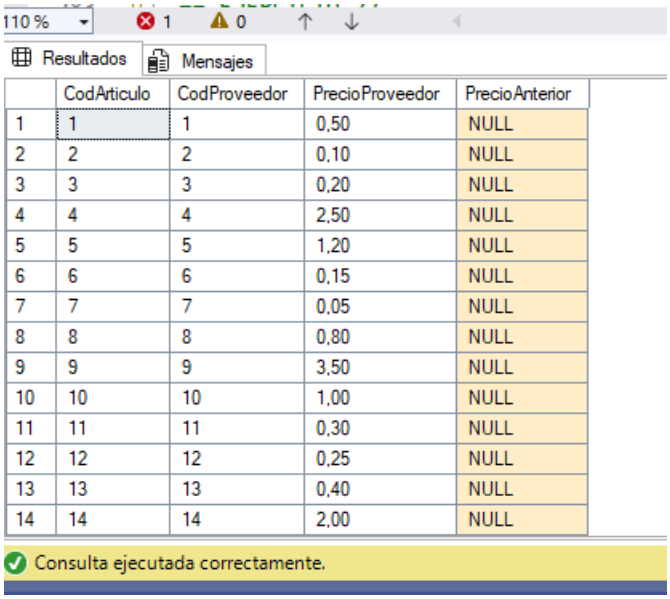
	CodArticulo	CodLinea	StockActual	PromedioMovil
1	1	1	500	500
2	2	2	1000	1000
3	3	3	800	800
4	4	4	200	200
5	5	5	300	300
6	6	6	900	900
7	7	7	1500	1500
8	8	8	400	400
9	9	9	100	100
10	10	10	250	250
11	11	11	600	600
12	12	12	700	700
13	13	13	350	350
14	14	14	120	120

Consulta ejecutada correctamente.

- RESULTADO 25

```sql
-- EJERCICIO 25

SELECT CodArticulo, CodProveedor, PrecioProveedor,
       LAG(PrecioProveedor) OVER (PARTITION BY CodProveedor ORDER BY CodArticulo) AS PrecioAnterior
FROM ARTICULO;
/*
EXPLICACION: Muestra el precio anterior de cada artículo del mismo proveedor.
*/
```

RESULTADO



110 % 1 0

Resultados Mensajes

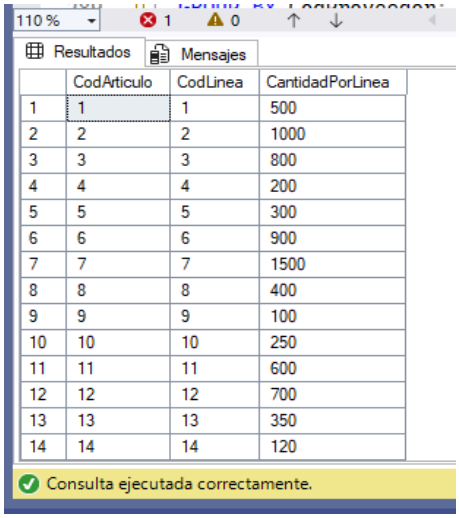
	CodArticulo	CodProveedor	PrecioProveedor	PrecioAnterior
1	1	1	0,50	NULL
2	2	2	0,10	NULL
3	3	3	0,20	NULL
4	4	4	2,50	NULL
5	5	5	1,20	NULL
6	6	6	0,15	NULL
7	7	7	0,05	NULL
8	8	8	0,80	NULL
9	9	9	3,50	NULL
10	10	10	1,00	NULL
11	11	11	0,30	NULL
12	12	12	0,25	NULL
13	13	13	0,40	NULL
14	14	14	2,00	NULL

Consulta ejecutada correctamente.

- ENUNCIADO 26

```sql
-- EJERCICIO 26

SELECT CodArticulo, CodLinea,
       SUM(StockActual) OVER (PARTITION BY CodLinea) AS CantidadPorLinea
FROM ARTICULO;
/*
EXPLICACION: Agrega el total de stock por línea a cada artículo.
*/
```

RESULTADO

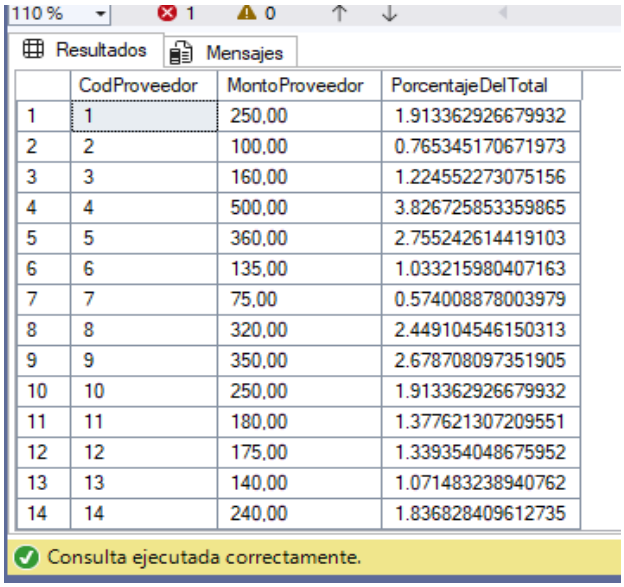


	CodArticulo	CodLinea	CantidadPorLinea
1	1	1	500
2	2	2	1000
3	3	3	800
4	4	4	200
5	5	5	300
6	6	6	900
7	7	7	1500
8	8	8	400
9	9	9	100
10	10	10	250
11	11	11	600
12	12	12	700
13	13	13	350
14	14	14	120

- ENUNCIADO 27

```sql
-- EJERCICIO 27

SELECT CodProveedor,
       SUM(PrecioProveedor * StockActual) AS MontoProveedor,
       SUM(PrecioProveedor * StockActual) * 100.0 / SUM(SUM(PrecioProveedor * StockActual)) OVER () AS PorcentajeDelTotal
FROM ARTICULO
GROUP BY CodProveedor;
/*
EXPLICACION: Calcula el porcentaje que representa cada proveedor del inventario total.
*/
```

RESULTADO

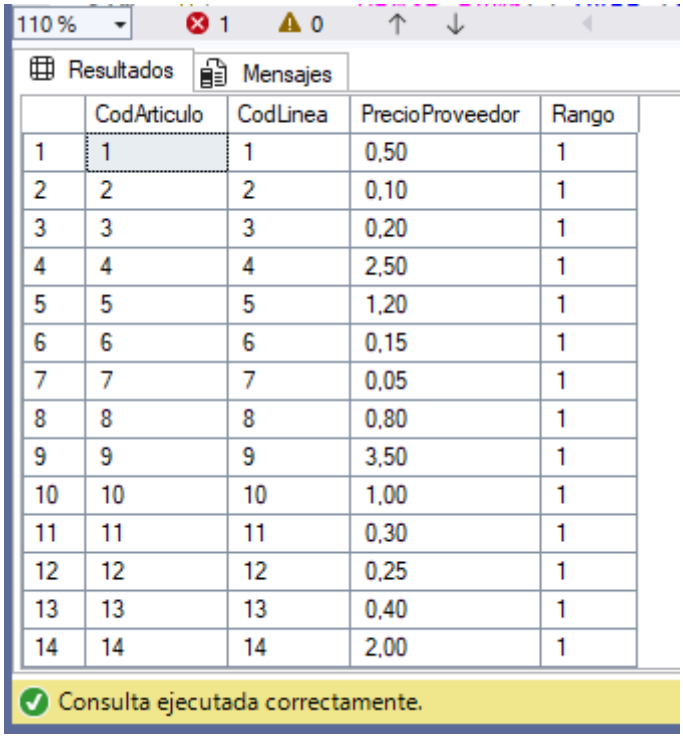


	CodProveedor	MontoProveedor	PorcentajeDelTotal
1	1	250,00	1.913362926679932
2	2	100,00	0.765345170671973
3	3	160,00	1.224552273075156
4	4	500,00	3.826725853359865
5	5	360,00	2.755242614419103
6	6	135,00	1.033215980407163
7	7	75,00	0.574008878003979
8	8	320,00	2.449104546150313
9	9	350,00	2.678708097351905
10	10	250,00	1.913362926679932
11	11	180,00	1.377621307209551
12	12	175,00	1.339354048675952
13	13	140,00	1.071483238940762
14	14	240,00	1.836828409612735

- ENUNCIADO 28

```sql
-- EJERCICIO 28

SELECT *
FROM (
    SELECT CodArticulo, CodLinea, PrecioProveedor,
           DENSE_RANK() OVER (PARTITION BY CodLinea ORDER BY PrecioProveedor DESC) AS Rango
    FROM ARTICULO
) AS Sub
WHERE Rango <= 3;
/*
EXPLICACION: Filtra los tres artículos más caros por cada línea.
*/
```

RESULTADO



110 % 1 0

Resultados Mensajes

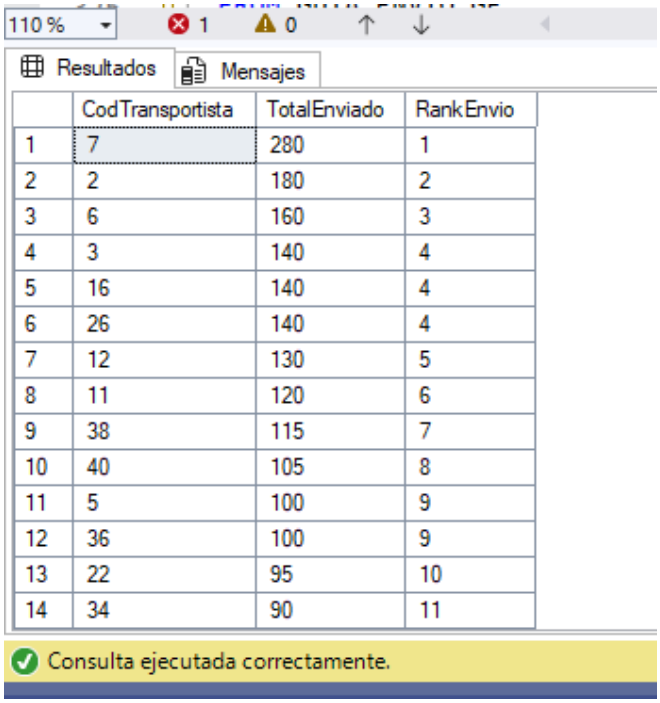
	CodArticulo	CodLinea	PrecioProveedor	Rango
1	1	1	0,50	1
2	2	2	0,10	1
3	3	3	0,20	1
4	4	4	2,50	1
5	5	5	1,20	1
6	6	6	0,15	1
7	7	7	0,05	1
8	8	8	0,80	1
9	9	9	3,50	1
10	10	10	1,00	1
11	11	11	0,30	1
12	12	12	0,25	1
13	13	13	0,40	1
14	14	14	2,00	1

Consulta ejecutada correctamente.

- ENUNCIADO 29

```sql
-- EJERCICIO 29

SELECT CodTransportista, TotalEnviado,
       DENSE_RANK() OVER (ORDER BY TotalEnviado DESC) AS RankEnvio
FROM (
    SELECT GE.CodTransportista, SUM(GD.CantidadEnviada) AS TotalEnviado
    FROM GUIA_ENVIO GE
    JOIN GUIA_DETALLE GD ON GE.NumGuia = GD.NumGuia
    GROUP BY GE.CodTransportista
) AS Totales;
/*
EXPLICACION: Asigna ranking por cantidad enviada a cada transportista.
*/
```

RESULTADO



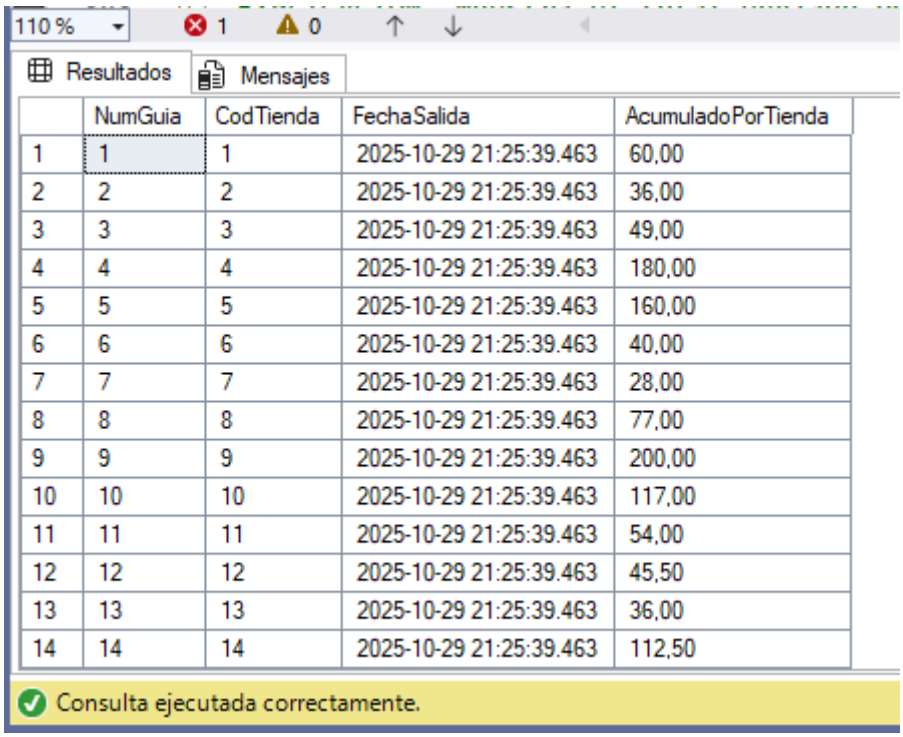
	CodTransportista	TotalEnviado	RankEnvio
1	7	280	1
2	2	180	2
3	6	160	3
4	3	140	4
5	16	140	4
6	26	140	4
7	12	130	5
8	11	120	6
9	38	115	7
10	40	105	8
11	5	100	9
12	36	100	9
13	22	95	10
14	34	90	11

Consulta ejecutada correctamente.

- ENUNCIADO 30

```sql
-- EJERCICIO 30

SELECT GE.NumGuia, CodTienda, FechaSalida,
       SUM(PrecioVenta * CantidadEnviada) OVER (PARTITION BY CodTienda ORDER BY FechaSalida) AS AcumuladoPorTienda
FROM GUIA_ENVIO GE
JOIN GUIA_DETALLE GD ON GE.NumGuia = GD.NumGuia;
/*
EXPLICACION: Calcula el total acumulado enviado por tienda hasta cada guía.
*/
```

RESULTADO



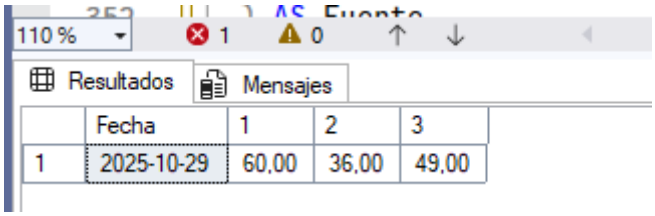
	NumGuia	CodTienda	FechaSalida	AcumuladoPorTienda
1	1	1	2025-10-29 21:25:39.463	60,00
2	2	2	2025-10-29 21:25:39.463	36,00
3	3	3	2025-10-29 21:25:39.463	49,00
4	4	4	2025-10-29 21:25:39.463	180,00
5	5	5	2025-10-29 21:25:39.463	160,00
6	6	6	2025-10-29 21:25:39.463	40,00
7	7	7	2025-10-29 21:25:39.463	28,00
8	8	8	2025-10-29 21:25:39.463	77,00
9	9	9	2025-10-29 21:25:39.463	200,00
10	10	10	2025-10-29 21:25:39.463	117,00
11	11	11	2025-10-29 21:25:39.463	54,00
12	12	12	2025-10-29 21:25:39.463	45,50
13	13	13	2025-10-29 21:25:39.463	36,00
14	14	14	2025-10-29 21:25:39.463	112,50

Consulta ejecutada correctamente.

- ENUNCIADO 31

```sql
-- EJERCICIO 31
SELECT *
FROM (
    SELECT CAST(FechaSalida AS DATE) AS Fecha, CodTienda, PrecioVenta * CantidadEnviada AS Total
    FROM GUIA_ENVIO GE
    JOIN GUIA_DETALLE GD ON GE.NumGuia = GD.NumGuia
) AS Fuente
PIVOT (
    SUM(Total) FOR CodTienda IN ([1], [2], [3])
) AS Resultado;

/*
EXPLICACION: Muestra el total enviado por tienda en columnas, agrupado por fecha.
*/
```

RESULTADO



110 % 1 0

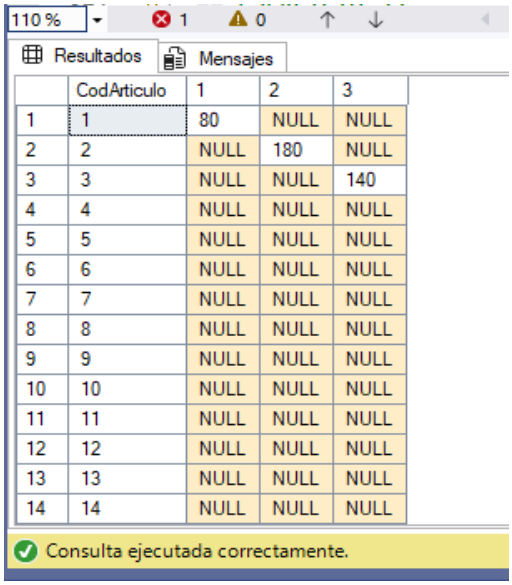
Resultados Mensajes

	Fecha	1	2	3
1	2025-10-29	60,00	36,00	49,00

- ENUNCIADO 32

```sql
-- EJERCICIO 32
SELECT *
FROM (
    SELECT GD.CodArticulo, GE.CodTienda, GD.CantidadEnviada
    FROM GUIA_ENVIO GE
    JOIN GUIA_DETALLE GD ON GE.NumGuia = GD.NumGuia
) AS Fuente
PIVOT (
    SUM(CantidadEnviada) FOR CodTienda IN ([1], [2], [3])
) AS Resultado;

/*
EXPLICACION: Muestra cada artículo con la cantidad enviada por tienda.
*/
```

RESULTADO



110 % 1 0

Resultados Mensajes

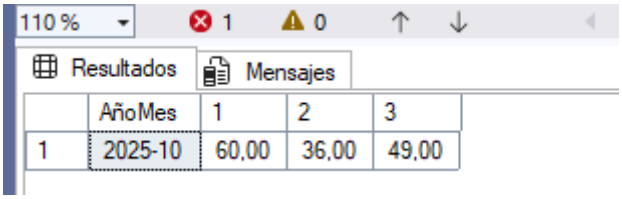
	CodArticulo	1	2	3
1	1	80	NULL	NULL
2	2	NULL	180	NULL
3	3	NULL	NULL	140
4	4	NULL	NULL	NULL
5	5	NULL	NULL	NULL
6	6	NULL	NULL	NULL
7	7	NULL	NULL	NULL
8	8	NULL	NULL	NULL
9	9	NULL	NULL	NULL
10	10	NULL	NULL	NULL
11	11	NULL	NULL	NULL
12	12	NULL	NULL	NULL
13	13	NULL	NULL	NULL
14	14	NULL	NULL	NULL

Consulta ejecutada correctamente.

- ENUNCIADO 33

```sql
-- EJERCICIO 33
SELECT *
FROM (
    SELECT FORMAT(FechaSalida, 'yyyy-MM') AS AñoMes, CodTienda, PrecioVenta * CantidadEnviada AS Total
    FROM GUIA_ENVIO GE
    JOIN GUIA_DETALLE GD ON GE.NumGuia = GD.NumGuia
) AS Fuente
PIVOT (
    SUM(Total) FOR CodTienda IN ([1], [2], [3])
) AS Resultado;

/*
EXPLICACION: Agrupa por mes y muestra el total enviado por tienda.
*/
```

RESULTADO



110 % 1 0

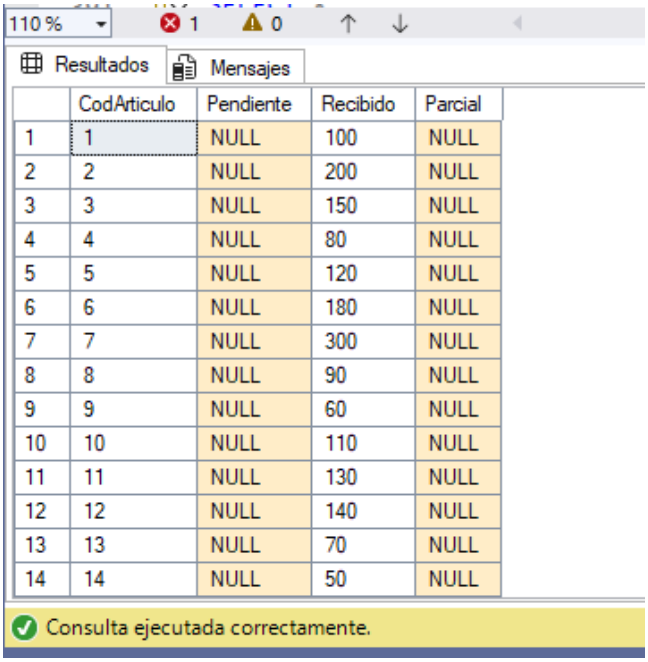
Resultados Mensajes

	AñoMes	1	2	3
1	2025-10	60,00	36,00	49,00

- ENUNCIADO 34

```sql
-- EJERCICIO 34
SELECT *
FROM (
    SELECT CodArticulo, Estado, CantidadSolicitada
    FROM ORDEN_DETALLE
) AS Fuente
PIVOT (
    SUM(CantidadSolicitada) FOR Estado IN ([Pendiente], [Recibido], [Parcial])
) AS Resultado;

/*
EXPLICACION: Muestra cada artículo con la cantidad solicitada por estado.
*/
```

RESULTADO



110 % 1 0

Resultados Mensajes

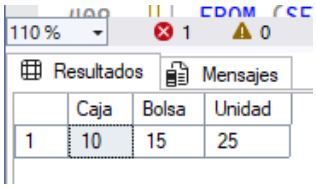
	CodArticulo	Pendiente	Recibido	Parcial
1	1	NULL	100	NULL
2	2	NULL	200	NULL
3	3	NULL	150	NULL
4	4	NULL	80	NULL
5	5	NULL	120	NULL
6	6	NULL	180	NULL
7	7	NULL	300	NULL
8	8	NULL	90	NULL
9	9	NULL	60	NULL
10	10	NULL	110	NULL
11	11	NULL	130	NULL
12	12	NULL	140	NULL
13	13	NULL	70	NULL
14	14	NULL	50	NULL

Consulta ejecutada correctamente.

- ENUNCIADO 35

```sql
-- EJERCICIO 35
SELECT *
FROM (
    SELECT Presentacion, CodArticulo
    FROM ARTICULO
) AS Fuente
PIVOT (
    COUNT(CodArticulo) FOR Presentacion IN ([Caja], [Bolsa], [Unidad])
) AS Resultado;

/*
EXPLICACION: Cuenta cuántos artículos hay por tipo de presentación.
*/
```

RESULTADO



110 % 1 0

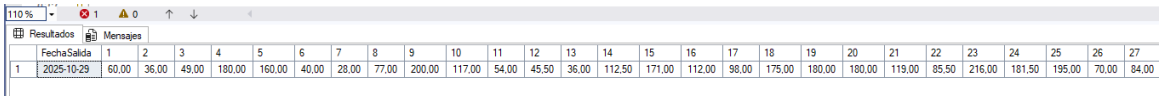
Resultados Mensajes

	Caja	Bolsa	Unidad
1	10	15	25

- ENUNCIADO 36

```sql
-- EJERCICIO 36
DECLARE @sql NVARCHAR(MAX);
DECLARE @cols NVARCHAR(MAX);

SELECT @cols = STRING_AGG(QUOTENAME(CodTienda), ',')
FROM (SELECT DISTINCT CodTienda FROM GUIA_ENVIO) AS T;

SET @sql = '
SELECT FechaSalida, ' + @cols + '
FROM (
    SELECT CAST(FechaSalida AS DATE) AS FechaSalida, CodTienda, PrecioVenta * CantidadEnviada AS Total
    FROM GUIA_ENVIO GE
    JOIN GUIA_DETALLE GD ON GE.NumGuia = GD.NumGuia
) AS Fuente
PIVOT (
    SUM(Total) FOR CodTienda IN (' + @cols + ')
) AS Resultado;';

EXEC sp_executesql @sql;

/*
EXPLICACION: Genera dinámicamente columnas por tienda según los datos existentes.
*/
```

RESULTADO



110 % 1 0

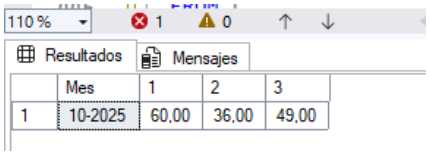
Resultados Mensajes

	FechaSalida	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27
1	2025-10-29	60,00	36,00	49,00	180,00	160,00	40,00	28,00	77,00	200,00	117,00	54,00	45,50	36,00	112,50	171,00	112,00	98,00	175,00	180,00	180,00	119,00	85,50	216,00	181,50	195,00	70,00	84,00

- ENUNCIADO 37

```sql
-- EJERCICIO 37
SELECT *
FROM (
    SELECT FORMAT(FechaSalida, 'MM-yyyy') AS Mes, CodTransportista, PrecioVenta * CantidadEnviada AS Total
    FROM GUIA_ENVIO GE
    JOIN GUIA_DETALLE GD ON GE.NumGuia = GD.NumGuia
) AS Fuente
PIVOT (
    SUM(Total) FOR CodTransportista IN ([1], [2], [3])
) AS Resultado;

/*
EXPLICACION: Muestra el total enviado por transportista en cada mes.
*/
```

RESULTADO

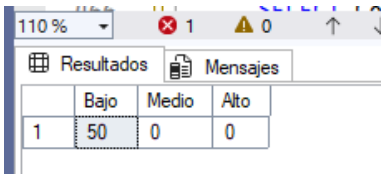


	Mes	1	2	3
1	10-2025	60,00	36,00	49,00

- ENUNCIADO 38

```sql
-- EJERCICIO 38
SELECT *
FROM (
    SELECT CodProveedor,
           CASE
               WHEN COUNT(CodArticulo) <= 5 THEN 'Bajo'
               WHEN COUNT(CodArticulo) <= 10 THEN 'Medio'
               ELSE 'Alto'
           END AS Rango
    FROM ARTICULO
    GROUP BY CodProveedor
) AS Fuente
PIVOT (
    COUNT(CodProveedor) FOR Rango IN ([Bajo], [Medio], [Alto])
) AS Resultado;

/*
EXPLICACION: Clasifica proveedores por variedad de artículos y los cuenta por rango.
*/
```

RESULTADO

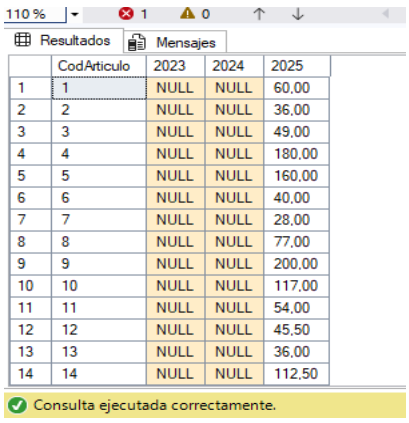


	Bajo	Medio	Alto
1	50	0	0

- ENUNCIADO 39

```sql
-- EJERCICIO 39
SELECT *
FROM (
    SELECT CodArticulo, YEAR(FechaSalida) AS Año, PrecioVenta * CantidadEnviada AS Total
    FROM GUIA_ENVIO GE
    JOIN GUIA_DETALLE GD ON GE.NumGuia = GD.NumGuia
) AS Fuente
PIVOT (
    SUM(Total) FOR Año IN ([2023], [2024], [2025])
) AS Resultado;

/*
EXPLICACION: Muestra el total vendido por artículo en cada año.
*/
```

RESULTADO



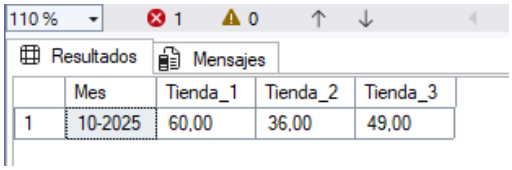
	CodArticulo	2023	2024	2025
1	1	NULL	NULL	60,00
2	2	NULL	NULL	36,00
3	3	NULL	NULL	49,00
4	4	NULL	NULL	180,00
5	5	NULL	NULL	160,00
6	6	NULL	NULL	40,00
7	7	NULL	NULL	28,00
8	8	NULL	NULL	77,00
9	9	NULL	NULL	200,00
10	10	NULL	NULL	117,00
11	11	NULL	NULL	54,00
12	12	NULL	NULL	45,50
13	13	NULL	NULL	36,00
14	14	NULL	NULL	112,50

Consulta ejecutada correctamente.

- ENUNCIADO 40

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

/*
EXPLICACION: Alternativa al PIVOT usando CASE para mostrar columnas por tienda.
*/
```

RESULTADO



	Mes	Tienda_1	Tienda_2	Tienda_3
1	10-2025	60,00	36,00	49,00

- ENUNCIADO 41

```sql
-- EJERCICIO 41
SELECT CodLinea, COUNT(*) AS CantArticulos
FROM ARTICULO
GROUP BY CodLinea
HAVING COUNT(*) > 10;

/*
EXPLICACION: Filtra líneas que tienen más de 10 artículos registrados.
*/
```

- ENUNCIADO 42

```sql
-- EJERCICIO 42
SELECT CodProveedor, SUM(PrecioProveedor * StockActual) AS MontoTotal
FROM ARTICULO
GROUP BY CodProveedor
HAVING SUM(PrecioProveedor * StockActual) > 50000;

/*
EXPLICACION: Muestra proveedores cuyo inventario supera los 50,000 en valor.
*/
```

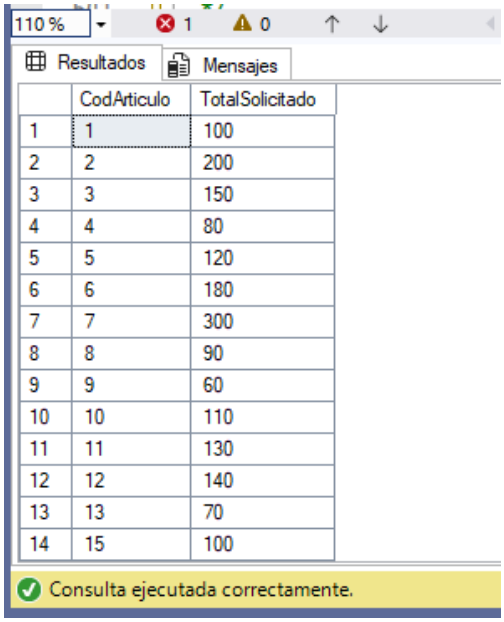
- ENUNCIADO 43

```sql
-- EJERCICIO 43
SELECT CodTienda, AVG(PrecioVenta * CantidadEnviada) AS PromedioGuia
FROM GUIA_ENVIO GE
JOIN GUIA_DETALLE GD ON GE.NumGuia = GD.NumGuia
GROUP BY CodTienda
HAVING AVG(PrecioVenta * CantidadEnviada) > 1000;

/*
EXPLICACION: Filtra tiendas cuyo promedio por guía supera los 1000.
*/
```

- ENUNCIADO 44

```sql
-- EJERCICIO 44
SELECT CodArticulo, SUM(CantidadSolicitada) AS TotalSolicitado
FROM ORDEN_DETALLE
GROUP BY CodArticulo
HAVING SUM(CantidadSolicitada) > 50;

/*
EXPLICACION: Muestra artículos que han sido solicitados más de 50 veces.
*/
```

RESULTADO



110 % 1 0

Resultados Mensajes

	CodArticulo	TotalSolicitado
1	1	100
2	2	200
3	3	150
4	4	80
5	5	120
6	6	180
7	7	300
8	8	90
9	9	60
10	10	110
11	11	130
12	12	140
13	13	70
14	15	100

Consulta ejecutada correctamente.

- ENUNCIADO 45

```sql
-- EJERCICIO 45
SELECT A.CodProveedor, COUNT(DISTINCT GE.NumGuia) AS CantGuias
FROM ARTICULO A
JOIN GUIA_DETALLE GD ON A.CodArticulo = GD.CodArticulo
JOIN GUIA_ENVIO GE ON GD.NumGuia = GE.NumGuia
GROUP BY A.CodProveedor
HAVING COUNT(DISTINCT GE.NumGuia) >= 5;

/*
EXPLICACION: Proveedores cuyos artículos han sido enviados en al menos 5 guías.
*/
```

- ENUNCIADO 46

```sql
-- EJERCICIO 46
SELECT CodLinea,
       SUM(StockActual) AS TotalStock,
       SUM(StockMinimo) * COUNT(*) AS LimiteTeorico
FROM ARTICULO
GROUP BY CodLinea
HAVING SUM(StockActual) < SUM(StockMinimo) * COUNT(*);

/*
EXPLICACION: Filtra líneas donde el stock total es menor al mínimo esperado por cantidad de artículos.
*/
```

- ENUNCIADO 47

```sql
-- EJERCICIO 47
SELECT CodProveedor, MAX(PrecioProveedor) AS PrecioMaximo
FROM ARTICULO
GROUP BY CodProveedor
HAVING MAX(PrecioProveedor) > 1000;

/*
EXPLICACION: Proveedores que ofrecen al menos un artículo con precio mayor a 1000.
*/
```

- ENUNCIADO 48

```sql
-- EJERCICIO 48
SELECT CodTienda,
       AVG(CantidadEnviada) AS PromedioCantidad,
       COUNT(DISTINCT GE.NumGuia) AS TotalGuias
FROM GUIA_ENVIO GE
JOIN GUIA_DETALLE GD ON GE.NumGuia = GD.NumGuia
GROUP BY CodTienda
HAVING AVG(CantidadEnviada) < 50 AND COUNT(DISTINCT GE.NumGuia) >= 10;

/*
EXPLICACION: Tiendas con bajo promedio de envío pero alta frecuencia de guías.
*/
```

- ENUNCIADO 49

```sql
-- EJERCICIO 49
SELECT CodLinea,
       MAX(PrecioProveedor) - MIN(PrecioProveedor) AS RangoPrecio
FROM ARTICULO
GROUP BY CodLinea
HAVING MAX(PrecioProveedor) - MIN(PrecioProveedor) > 20;

/*
EXPLICACION: Líneas con diferencia de precios entre artículos mayor a 20.
*/
```

- ENUNCIADO 50

```sql
-- EJERCICIO 50
SELECT CodProveedor,
       COUNT(*) AS TotalArticulos,
       AVG(StockActual) AS PromedioStock
FROM ARTICULO
GROUP BY CodProveedor
HAVING AVG(StockActual) < 20 AND COUNT(*) > 5;

/*
EXPLICACION: Proveedores con más de 5 artículos y promedio de stock bajo.
*/
```