

**UNIVERSIDAD PERUANA LOS ANDES FACULTAD DE
INGENIERÍA**



Asignatura: Base de Datos II

Docente: Dr. Raúl Enrique Fernández Bejarano

Alumno: SORIANO TIMOTEO Joel Kevin

Ciclo: V

huancayo-Perú-2025

¿Qué es la Cláusula OVER en SQL?

La cláusula **OVER** permite definir una “ventana” o rango específico de filas dentro del resultado de una consulta, sobre el cual se ejecuta una **Función de Ventana** (muchas veces una función de agregación).

A diferencia de **GROUP BY**, que combina varias filas en una sola fila resumen, **OVER** calcula valores agregados **sin eliminar las filas originales**.

Esto provoca que el resultado calculado (por ejemplo, suma, promedio o máximo) aparezca **repetido en cada fila perteneciente a la misma ventana**.

Una forma sencilla de entenderlo es imaginar el cálculo del promedio de sueldos de un departamento y luego mostrar ese promedio junto al sueldo individual de cada trabajador.

¿Cómo se Utiliza? (Funciones de Ventana)

La cláusula **OVER** se usa únicamente junto con una **Función de Ventana**.

Estas funciones pueden ser de tres categorías:

1. Funciones de Agregación (en contexto de ventana)

Las funciones tradicionales como **SUM, AVG, COUNT, MIN, MAX** pasan a actuar como **funciones de ventana** cuando se les agrega la cláusula **OVER()**.

Sintaxis General:

```
FUNCION_DE_AGREGACION(columna) OVER (
    [PARTITION BY columna_para_dividir]
    [ORDER BY columna_para_ordenar]
    [Frame Specification]
)
```

2. Elementos Clave Dentro de OVER()

Componente	Función	Analogía
PARTITION BY	Divide las filas en grupos separados (ventanas). El cálculo se reinicia para cada grupo.	Clasificar empleados por Departamento .
ORDER BY	Ordena las filas dentro de cada partición. Esto es crucial para los cálculos de rango y las medias móviles.	Ordenar los empleados dentro de cada departamento por Salario .

Ejemplo 1: Promedio del Grupo (Sin colapsar filas)

Imaginemos que deseamos visualizar el sueldo de cada trabajador y, además, mostrar el promedio salarial correspondiente a su departamento, pero **sin combinar las filas**.

```
SELECT  
    Nombre,  
    Departamento,  
    Salario,  
    AVG(Salario)    OVER    (PARTITION    BY    Departamento)    AS  
PromedioDepartamento  
FROM Empleados;
```

En este caso, la función **AVG()** calcula el promedio de los sueldos por departamento, pero gracias a **OVER()**, cada fila sigue mostrando la información individual del empleado junto con el promedio de su grupo.

Nombre	Departamento	Salario	PromedioDepartamento
Ana	Ventas	40000	45000
Luis	Ventas	50000	45000
Clara	IT	70000	75000

Nombre	Departamento	Salario	PromedioDepartamento
Pedro	IT	80000	75000

Resultado:

A diferencia de usar **GROUP BY**, aquí obtenemos las cuatro filas originales, y la columna **PromedioDepartamento** muestra el mismo valor promedio en todas las filas que pertenecen al mismo departamento.

Ejemplo 2: Funciones de Ranking (La Aplicación más Útil)

Las funciones de ranking solo pueden emplearse junto con la cláusula **OVER**.

- **ROW_NUMBER()**: Genera un número secuencial único para cada fila dentro de la ventana.
- **RANK()**: Asigna un puesto; si existen empates, el siguiente número de rango se salta.
- **DENSE_RANK()**: Asigna un puesto; si hay empates, el siguiente rango **no** se salta.

Consulta: Obtener la posición de salario más alta (ranking) dentro de cada departamento

```
SELECT
    Nombre,
    Departamento,
    Salario,
    DENSE_RANK() OVER (PARTITION BY Departamento ORDER BY Salario
DESC) AS RangoSalarial
FROM Empleados;
```

Explicación:

1. **PARTITION BY Departamento:** Reinicia el ranking para cada área (por ejemplo, IT, Ventas, etc.).
2. **ORDER BY Salario DESC:** El orden de clasificación se basa en el salario, desde el monto más elevado hacia el más bajo.

El resultado indicará si un empleado ocupa la primera posición salarial (Rango 1), la segunda (Rango 2), y así sucesivamente, considerando únicamente a los trabajadores de su propio departamento.

CLÁUSULA OVER

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

21. Asignar posición por linea ordenada por precio.
22. Calcular costo por orden y su RANK (RankCosto)
23. Mostrar TotalDia y AcumuladoVentas ordenado por fecha.
24. Calcular promedio móvil para stock.
25. Mostrar PrecioAnteriorMiAmoProveedor usando LAG.
26. Añadir columna CantidadPorLinea a cada artículo.
27. Mostrar MontoProveedor y PorcentajeDelTotal.
28. Mostrar solo los 3 artículos más caros por linea.
29. Mostrar transportista y su DenseRank por TotalEnviado.
30. Mostrar por guía la suma acumulada por tienda hasta esa guía (ordenada por FechaSalida)

21. Asignar posición por línea ordenada por precio.

Enunciado: Asignar una posición (número de fila) a cada artículo, partiendo desde 1, dentro de su propia linea de producto, ordenado por el precio del proveedor de forma descendente.

Consulta SQL:

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

The screenshot shows the Microsoft SQL Server Management Studio (SSMS) interface. On the left, the Object Explorer displays the database structure for 'QchatPERU'. Under the 'Tables' node, 'Artículo' is selected. The right side of the screen shows the Query Editor with a query and its results.

Query Editor:

```
-- 21. Asignar posición por linea ordenada por precio. (ROW_NUMBER)
SELECT
    CodArticulo,
    DescripcionArticulo,
    CodLinea,
    PrecioProveedor,
    ROW_NUMBER() OVER (PARTITION BY CodLinea ORDER BY PrecioProveedor DESC) AS Posicion
FROM
```

Results Grid:

	CodArt...	DescripciónArticulo	CodLinea	PrecioProv...	Posic...
1.	84	CB0101234567890123456789	84	45,00	1
2.	85	Filtro Hepa Mod-6 Ska-B5	85	45,00	1
3.	86	Filtro Hepa Mod-7 Ska-B6	86	45,00	1
4.	87	Interruptor Servicio Mod-8 Ska...	87	45,00	1
5.	88	Filtro Hepa Mod-9 Ska-B8	88	45,00	1
6.	89	Rodamiento Z-20 Mod-10 Ska-B9	89	47,00	1
7.	90	Tornillo Titánio Mod-1 Ska-B0	90	47,00	1
8.	91	Rodamiento Z-20 Mod-2 Ska-B1	91	48,00	1
9.	92	Tornillo Titánio Mod-3 Ska-B2	92	48,00	1
10.	93	Interruptor Servicio Mod-4 Ska...	93	49,00	1
11.	94	Interruptor Servicio Mod-5 Ska...	94	49,00	1
12.	95	Cable Cat 6 Mod-6 Ska-B5	95	50,00	1
13.	96	Interruptor Servicio Mod-7 Ska...	96	50,00	1
14.	97	Sensor Óptico Mod-8 Ska-B7	97	51,00	1
15.	98	Malla Industrial Mod-9 Ska-B8	98	51,00	1
16.	99	Tornillo Titánio Mod-10 Ska-B...	99	52,00	1
17.	100	Válvula Flotador Mod-1 Ska-B00	100	52,00	1

Explicación: Se utiliza la función **ROW_NUMBER()** dentro de la cláusula OVER. El **PARTITION BY CodLinea** agrupa los artículos por línea, y el **ORDER BY PrecioProveedor DESC** asigna la posición dentro de cada grupo, dando el número 1 al artículo más caro de esa línea.

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

Enunciado: Calcular el costo total de cada orden de compra y luego asignar un rango de clasificación (RANK) a esas órdenes basado en dicho costo.

Consulta SQL:

```
-- 22. Calcular costo por orden y su RANK (RankCosto). (RANK)
WITH OrdenCosto AS (
    -- Subconsulta para calcular el costo total de cada orden
    SELECT
        NumOrden,
        SUM(PrecioCompra * CantidadSolicitada) AS CostoTotal
    FROM
        ORDEN_DETALLE
    GROUP BY
        NumOrden
)
SELECT
    NumOrden,
    CostoTotal,
    RANK() OVER (ORDER BY CostoTotal DESC) AS RankCosto
FROM
    OrdenCosto
ORDER BY
    RankCosto;
GO
```


File Edit View Help ← → ⌂ Search

CONNECTIONS AZURE + ... connected ■ SQLQuery_1.sql - (62) t..U (sat) ■ SQLQuery_1 - (53) t..U (sat) Database: ChetuPERU ↴ Estimated Plan

Run Cancel Disconnect Change Database: ChetuPERU ↴ Estimated Plan

18 ...
19
20 -- 28. Calcular costo por orden y su RANK (RankCosto), (RANK)
21 WITH OrdenCosto AS (-- Subconsulta para calcular el costo total de cada orden
22
23 SELECT
24 NumOrden,
25 SUM(PrecioCompra * CantidadSolicitada) AS CostoTotal
26 FROM
27 ORDEN_DETALLE
28 GROUP BY
29 NumOrden

Results Messages

NumOrden	CostoTotal	RankCosto
1	31500.00	1
2	30940.00	2
3	30939.00	3
4	30937.00	4
5	30936.00	5
6	30935.00	6
7	30934.00	7
8	30933.00	8
9	30932.00	9
10	30931.00	10
11	30930.00	11
12	30929.00	12
13	30928.00	13
14	30927.00	14
15	30926.00	15
16	30925.00	16
17	30924.00	17

File Edit View Help ← → ⌂ Search

CONNECTIONS AZURE + ... connected ■ SQLQuery_1.sql - (62) t..U (sat) ■ SQLQuery_1 - (53) t..U (sat) Database: ChetuPERU ↴ Estimated Plan

Run Cancel Disconnect Change Database: ChetuPERU ↴ Estimated Plan

18 ...
19
20 -- 28. Calcular costo por orden y su RANK (RankCosto), (RANK)
21 WITH OrdenCosto AS (-- Subconsulta para calcular el costo total de cada orden
22
23 SELECT
24 NumOrden,
25 SUM(PrecioCompra * CantidadSolicitada) AS CostoTotal
26 FROM
27 ORDEN_DETALLE
28 GROUP BY
29 NumOrden

Results Messages

NumOrden	CostoTotal	RankCosto
84	2835.00	84
85	2830.00	85
86	2825.00	86
87	2820.00	87
88	2815.00	88
89	2810.00	89
90	2805.00	90
91	2800.00	91
92	2815.00	92
93	2800.00	93
94	2810.00	94
95	2805.00	95
96	2805.00	96
97	2800.00	97
98	2800.00	98
99	2800.00	99
100	2805.00	100

Explicación: Primero, una Expresión de Tabla Común (CTE) calcula el costo total por NumOrden. Luego, la función **RANK()** se aplica sobre ese costo total para clasificar todas las órdenes. Los empates reciben el mismo rango, y el siguiente rango salta posiciones.

23. Mostrar TotalDia y AcumuladoVentas ordenado por fecha.

Enunciado: Calcular el valor total de ventas de cada día (TotalDia) y mostrar el valor acumulado de ventas hasta esa fecha (AcumuladoVentas).

Consulta SQL:

-- 23. Mostrar TotalDia y AcumuladoVentas ordenado por fecha.
(SUM OVER con Marco de Ventana)

SELECT

CAST(GE.FechaSalida **AS DATE**) **AS** DiaSalida,
SUM(GD.CantidadEnviada * GD.PrecioVenta) **AS** TotalDia,
-- Calcula la suma acumulada del TotalDia hasta la fecha actual
SUM(SUM(GD.CantidadEnviada * GD.PrecioVenta)) **OVER**

(ORDER BY CAST(GE.FechaSalida **AS DATE) ROWS BETWEEN
UNBOUNDED PRECEDING AND CURRENT ROW) AS**

AcumuladoVentas

FROM

GUIA_ENVIO GE

JOIN

GUIA_DETALLE GD **ON** GE.NumGuia = GD.NumGuia

GROUP BY

CAST(GE.FechaSalida **AS DATE**)

ORDER BY

DiaSalida;

GO

File Edit View Help connected SQLQuery_1.log - (62) t..U (set) SQLQuery_1 - (53) t..U (set) Database: ChuruPERU Estimated Plan

```

37 ORDER BY
38   RankCentro;
39
40
41
42
43 -- 23. Mostrar TotalDía y AcumuladoVentas ordenado por fecha. (SUM OVER con Parte de la
44 SELECT
45   ...CAST(GO.FechaSalida AS DATE) AS DiaSalida,
46   ...SUM(GO.CantidadEnvia * GO.PrecioVenta) AS TotalDía,
47   -- Calcular la suma acumulada del TotalDía hasta la fecha actual
48   ...SUM(SUM(GO.CantidadEnvia * GO.PrecioVenta)) OVER(ORDER BY CAST(GO.FechaSalida) )
49
50

```

	TotalDía	AcumuladoVentas
1	2025-09-10	210000.00
2	2025-09-11	210000.00
3	2025-09-12	121021.00
4	2025-09-13	161204.00
5	2025-09-14	443777.00
6	2025-09-15	426664.00
7	2025-09-16	489984.00
8	2025-09-17	361000.00
9	2025-09-18	377700.00
10	2025-09-19	341477.00
11	2025-09-20	346602.00
12	2025-09-21	336998.00
13	2025-09-22	316112.00
14	2025-09-23	348758.00
15	2025-09-24	287577.00
16	2025-09-25	373000.00
17	2025-09-26	342532.00

File Edit View Help connected SQLQuery_1.log - (62) t..U (set) SQLQuery_1 - (53) t..U (set) Database: ChuruPERU Estimated Plan

```

37 ORDER BY
38   RankCentro;
39
40
41
42
43 -- 23. Mostrar TotalDía y AcumuladoVentas ordenado por fecha. (SUM OVER con Parte de la
44 SELECT
45   ...CAST(GO.FechaSalida AS DATE) AS DiaSalida,
46   ...SUM(GO.CantidadEnvia * GO.PrecioVenta) AS TotalDía,
47   -- Calcular la suma acumulada del TotalDía hasta la fecha actual
48   ...SUM(SUM(GO.CantidadEnvia * GO.PrecioVenta)) OVER(ORDER BY CAST(GO.FechaSalida) )
49
50

```

	TotalDía	AcumuladoVentas
35	2025-10-14	7507.00
36	2025-10-15	6881.00
37	2025-10-16	6228.00
38	2025-10-17	5582.00
39	2025-10-18	4978.00
40	2025-10-19	4487.00
41	2025-10-20	3879.00
42	2025-10-21	3366.00
43	2025-10-22	2894.00
44	2025-10-23	2469.00
45	2025-10-24	2057.00
46	2025-10-25	1688.00
47	2025-10-26	1351.00
48	2025-10-27	1058.00
49	2025-10-28	782.00
50	2025-10-29	547.00
51	2025-10-30	196.00

Explicación: Se agrupa la venta total por día. Luego, se utiliza una segunda función **SUM()** con OVER sin PARTITION BY y con un marco de ventana (ROWS BETWEEN...) para calcular un **total acumulado** global, sumando las ventas desde el inicio hasta la fila actual (la fecha actual).

24. Calcular promedio móvil para stock.

Enunciado: Para cada artículo, calcular el promedio móvil de StockActual, promediando el stock del artículo anterior, el artículo actual y el artículo siguiente dentro de su misma línea.

Consulta SQL:

-- 24. Calcular promedio móvil para stock. (AVG OVER con Marco de Ventana)

SELECT

CodArticulo,
DescripcionArticulo,
CodLinea,
StockActual,

-- Promedio móvil: promedio del stock del artículo anterior, el actual y el siguiente dentro de la linea

CAST(AVG(StockActual) OVER (
PARTITION BY CodLinea
ORDER BY CodArticulo
ROWS BETWEEN 1 PRECEDING AND 1 FOLLOWING
) AS DECIMAL(10,2)) AS PromedioMovilStock

FROM

ARTICULO

ORDER BY

CodLinea, CodArticulo;

GO

File Edit View Help

CONNECTIONS: AZURE + ... connected

SQL databases

- cafeito (jean)
- master (jean)
- master (trabaj)
- ChatuPERU (jean)
- Tables

```
64. Calcular promedio stock para articulo. (AVG con Fuerco de Ventana)
SELECT
    CodArticulo,
    DescripcionArticulo,
    CodLinea,
    StockActual,
    -- Promedio stock: promedio del stock del articulo anterior, el actual y el siguiente
    CAST( AVG(StockActual) ) OVER (
```

	CodArticulo	DescripcionArticulo	CodLinea	StockActual
1	1	Filtro Hepa Mod-2 Sku-1	1	52
2	2	Rodamiento Z-20 Mod-3 Sku-2	2	54
3	3	Tensore Optico Mod-4 Sku-3	3	56
4	4	Rodamiento Z-20 Mod-5 Sku-4	4	58
5	5	Válvula Flotjo Mod-6 Sku-5	5	60
6	6	Malla Industrial Mod-7 Sku-6	6	62
7	7	Válvula Flotjo Mod-8 Sku-7	7	64
8	8	Filtro Hepa Mod-9 Sku-8	8	66
9	9	Válvula Flotjo Mod-10 Sku-9	9	68
10	10	Malla Industrial Mod-11 Sku-10	10	70
11	11	Tensore Optico Mod-12 Sku-11	11	72
12	12	Cable Cat-6 Mod-13 Sku-12	12	74
13	13	Malla Industrial Mod-14 Sku-13	13	76
14	14	Rodamiento Z-20 Mod-15 Sku-14	14	78
15	15	Válvula Flotjo Mod-16 Sku-15	15	80
16	16	Tornillo Tornillo Mod-17 Sku-16	16	82

File Edit View Help

CONNECTIONS: AZURE + ... connected

SQL databases

- cafeito (jean)
- master (jean)
- master (trabaj)
- ChatuPERU (jean)
- Tables

```
64. Calcular promedio stock para articulo. (AVG con Fuerco de Ventana)
SELECT
    CodArticulo,
    DescripcionArticulo,
    CodLinea,
    StockActual,
    -- Promedio stock: promedio del stock del articulo anterior, el actual y el siguiente
    CAST( AVG(StockActual) ) OVER (
```

	CodArticulo	DescripcionArticulo	CodLinea	StockActual
84	84	Cable Cat-6 Mod-17 Sku-84	84	210
85	85	Filtro Hepa Mod-8 Sku-85	85	220
86	86	Filtro Hepa Mod-7 Sku-86	86	222
87	87	Interruptor Térmico Mod-8 Sku-87	87	224
88	88	Filtro Hepa Mod-9 Sku-88	88	226
89	89	Rodamiento Z-20 Mod-10 Sku-89	89	228
90	90	Tornillo Tornillo Mod-1 Sku-90	90	230
91	91	Rodamiento Z-20 Mod-2 Sku-91	91	232
92	92	Tornillo Tornillo Mod-3 Sku-92	92	234
93	93	Interruptor Térmico Mod-4 Sku-93	93	236
94	94	Interruptor Térmico Mod-5 Sku-94	94	238
95	95	Cable Cat-6 Mod-6 Sku-95	95	240
96	96	Interruptor Térmico Mod-7 Sku-96	96	242
97	97	Tensore Optico Mod-8 Sku-97	97	244
98	98	Malla Industrial Mod-9 Sku-98	98	246
99	99	Tornillo Tornillo Mod-10 Sku-99	99	248
100	100	Válvula Flotjo Mod-1 Sku-100	100	250

Explicación: Se utiliza la función **AVG()** con OVER. El PARTITION BY CodLinea asegura que el promedio se calcule por línea. El marco de ventana **ROWS BETWEEN 1 PRECEDING AND 1 FOLLOWING** define el "móvil", que incluye un registro anterior, el actual, y un registro siguiente para el cálculo del promedio.

25. Mostrar PrecioAnteriorMismoProveedor usando LAG.

Enunciado: Mostrar el precio del proveedor (PrecioProveedor) del artículo actual junto con el precio del artículo inmediatamente anterior, siempre y cuando sea del mismo proveedor.

Consulta SQL:

-- 25. Mostrar PrecioAnteriorMismoProveedor usando LAG. (LAG)

SELECT

CodArticulo,

CodProveedor,

PrecioProveedor,

-- Obtiene el PrecioProveedor del artículo inmediatamente anterior, ordenado por código de artículo, dentro del mismo proveedor

LAG(PrecioProveedor, 1, 0) OVER (PARTITION BY

CodProveedor ORDER BY CodArticulo) AS

PrecioAnteriorMismoProveedor

FROM

ARTICULO

ORDER BY

CodProveedor, CodArticulo;

GO

File Edit View Help

CONNECTIONS AZURE + connected ■ SQLQuery_1.sqf - (52) L:U (set) ■ SQLQuery_1 - (53) L:U (set)

Alberto JEANCARLO CHIQUERA...

- Azure for Students
- SQL databases
 - catelito (jean2)
 - master (jean2)
 - master (trabaj)
 - ChatuPERU (jean2)
 - Tables
 - articulo
 - sys (jean2)
 - trabajos_05 (jean2)
- SQL servers

Estimate Actual Plan / Perse

```
--> 25. Mostrar PreciosArticulosPorProveedor usando LAG - (L:U)
SELECT
    ...
    ,CodArticulo,
    CodProveedor,
    PrecioProveedor,
    ...
    --> Obtener el PrecioProveedor del articulo inmediatamente anterior, ordenado por
    --> LAG[PrecioProveedor, 1, 0] OVER(PARTITION BY CodProveedor ORDER BY CodArticulo)
FROM
```

CodArticulo	CodProveedor	PrecioProveedor	PrecioAnteriorProveedor
1	1	10.00	0.00
2	2	11.00	0.00
3	3	12.00	0.00
4	4	13.00	0.00
5	5	14.00	0.00
6	6	15.00	0.00
7	7	16.00	0.00
8	8	17.00	0.00
9	9	18.00	0.00
10	10	19.00	0.00
11	11	20.00	0.00
12	12	21.00	0.00
13	13	22.00	0.00
14	14	23.00	0.00
15	15	24.00	0.00
16	16	25.00	0.00

File Edit View Help

CONNECTIONS AZURE + connected ■ SQLQuery_1.sqf - (52) L:U (set) ■ SQLQuery_1 - (53) L:U (set)

Alberto JEANCARLO CHIQUERA...

- Azure for Students
- SQL databases
 - catelito (jean2)
 - master (jean2)
 - master (trabaj)
 - ChatuPERU (jean2)
 - Tables
 - articulo
 - sys (jean2)
 - trabajos_05 (jean2)
- SQL servers

Estimate Actual Plan / Perse

```
--> 25. Mostrar PreciosArticulosPorProveedor usando LAG - (L:U)
SELECT
    ...
    ,CodArticulo,
    CodProveedor,
    PrecioProveedor,
    ...
    --> Obtener el PrecioProveedor del articulo inmediatamente anterior, ordenado por
    --> LAG[PrecioProveedor, 1, 0] OVER(PARTITION BY CodProveedor ORDER BY CodArticulo)
FROM
```

CodArticulo	CodProveedor	PrecioProveedor	PrecioAnteriorProveedor
64	64	46.00	0.00
65	65	47.00	0.00
66	66	48.00	0.00
67	67	49.00	0.00
68	68	49.50	0.00
69	69	50.00	0.00
70	70	47.50	0.00
71	71	48.00	0.00
72	72	48.50	0.00
73	73	49.00	0.00
74	74	49.50	0.00
75	75	50.00	0.00
76	76	50.50	0.00
77	77	51.00	0.00
78	78	51.50	0.00
79	79	52.00	0.00
80	80	52.50	0.00

Explicación: Se usa la función de ventana **LAG()** para acceder al valor de una fila anterior. El PARTITION BY CodProveedor asegura que solo se compare con artículos del mismo proveedor, mientras que el ORDER BY CodArticulo define cuál es el "anterior" lógico. El 1 indica la diferencia de 1 fila, y el 0 es el valor por defecto si no hay fila anterior.

26. Añadir columna CantidadPorLinea a cada artículo.

Enunciado: Para cada artículo, añadir una columna que muestre el conteo total de artículos que existen en su misma línea de producto.

Consulta SQL:

SQL

```
-- 26. Añadir columna CantidadPorLinea a cada artículo. (COUNT  
OVER)
```

```
SELECT
```

```
    CodArticulo,  
    DescripcionArticulo,  
    CodLinea,  
    PrecioProveedor,  
    -- Cuenta el total de artículos que existen en la misma línea  
    COUNT(CodArticulo) OVER (PARTITION BY CodLinea) AS  
    CantidadPorLinea  
FROM  
    ARTICULO  
ORDER BY  
    CodLinea;  
GO
```

File Edit View Help

CONNECTIONS: AZURE + ... connected SQLQuery_1.sql - (62) t...U (sa1) SQLQuery_1 - (53) t...U (sa1) Database: QhatuPERU Estimated Plan

Enable Actual Plan Parse

```

94  ***
95
96  -- 26. Añadir columna CantidadPorLinea a cada articulo. (COUNT OVER)
97  SELECT
98      CodArticulo,
99      DescripcionArticulo,
100     CodLinea,
101     PrecioProveedor,
102     -- Cuenta el total de articulos que existen en la misma linea
103     COUNT(CodArticulo) OVER (PARTITION BY CodLinea) AS CantidadPorLinea
104
105 FROM
106     ARTICULO

```

Results Messages

	CodArticulo	DescripcionArticulo	CodLinea	PrecioProveedor
1	1	Filtro Hepa Mod-2 Sku-1	1	3.00
2	2	Rodamiento Z-20 Mod-3 Sku-2	2	3.50
3	3	Sensor Óptico Mod-4 Sku-3	3	4.00
4	4	Rodamiento Z-20 Mod-5 Sku-4	4	4.50
5	5	Válvula Flujo Mod-6 Sku-5	5	5.00
6	6	Malla Industrial Mod-7 Sku-6	6	5.50
7	7	Válvula Flujo Mod-8 Sku-7	7	6.00
8	8	Filtro Hepa Mod-9 Sku-8	8	6.50
9	9	Válvula Flujo Mod-10 Sku-9	9	7.00
10	10	Malla Industrial Mod-1 Sku-10	10	7.50
11	11	Sensor Óptico Mod-2 Sku-11	11	8.00
12	12	Cable Cat 6 Mod-3 Sku-12	12	8.50
13	13	Malla Industrial Mod-4 Sku-13	13	9.00
14	14	Rodamiento Z-20 Mod-5 Sku-14	14	9.50
15	15	Válvula Flujo Mod-6 Sku-15	15	10.00
16	16	Tornillo Titanium Mod-7 Sku-16	16	10.50

File Edit View Help

CONNECTIONS: AZURE + ... connected SQLQuery_1.sql - (62) t...U (sa1) SQLQuery_1 - (53) t...U (sa1) Database: QhatuPERU Estimated Plan

Enable Actual Plan Parse

```

94  ***
95
96  -- 26. Añadir columna CantidadPorLinea a cada articulo. (COUNT OVER)
97  SELECT
98      CodArticulo,
99      DescripcionArticulo,
100     CodLinea,
101     PrecioProveedor,
102     -- Cuenta el total de articulos que existen en la misma linea
103     COUNT(CodArticulo) OVER (PARTITION BY CodLinea) AS CantidadPorLinea
104
105 FROM
106     ARTICULO

```

Results Messages

	CodArticulo	DescripcionArticulo	CodLinea	PrecioProveedor
84	84	Cable Cat 6 Mod-5 Sku-84	84	44.00
85	85	Filtro Hepa Mod-6 Sku-85	85	45.00
86	86	Filtro Hepa Mod-7 Sku-86	86	45.50
87	87	Interruptor Térmico Mod-8 Sk...	87	46.00
88	88	Filtro Hepa Mod-9 Sku-88	88	46.50
89	89	Rodamiento Z-20 Mod-10 Sku-89	89	47.00
90	90	Tornillo Titanium Mod-1 Sku-90	90	47.50
91	91	Rodamiento Z-20 Mod-2 Sku-91	91	48.00
92	92	Tornillo Titanium Mod-3 Sku-92	92	48.50
93	93	Interruptor Térmico Mod-4 Sk...	93	49.00
94	94	Interruptor Térmico Mod-5 Sk...	94	49.50
95	95	Cable Cat 6 Mod-6 Sku-95	95	50.00
96	96	Interruptor Térmico Mod-7 Sk...	96	50.50
97	97	Sensor Óptico Mod-8 Sku-97	97	51.00
98	98	Malla Industrial Mod-9 Sku-98	98	51.50
99	99	Tornillo Titanium Mod-10 Sku-...	99	52.00
100	100	Válvula Flujo Mod-1 Sku-100	100	52.50

Explicación: Se utiliza la función de agregación **COUNT()** con la cláusula OVER. El **PARTITION BY CodLinea** instruye a SQL a contar los artículos dentro de cada línea de forma independiente, repitiendo el resultado del conteo en cada fila de esa partición.

27. Mostrar MontoProveedor y PorcentajeDelTotal.

Enunciado: Calcular el monto total de compra por cada proveedor (MontoProveedor) y mostrar qué porcentaje representa ese monto respecto al monto total global de todas las compras.

Consulta SQL:

```
-- 27. Mostrar MontoProveedor y PorcentajeDelTotal. (SUM OVER
-- para Total Global)
WITH Montos AS (
    -- Subconsulta para calcular el monto total (costo) que representa
    -- cada proveedor
    SELECT
        P.CodProveedor,
        P.NomProveedor,
        SUM(OD.PrecioCompra * OD.CantidadSolicitada) AS
        MontoProveedor
    FROM
        PROVEEDOR P
    JOIN
        ARTICULO A ON P.CodProveedor = A.CodProveedor
    JOIN
        ORDEN_DETALLE OD ON A.CodArticulo = OD.CodArticulo
    GROUP BY
        P.CodProveedor, P.NomProveedor
)
SELECT
    CodProveedor,
    NomProveedor,
    MontoProveedor,
    -- Se divide el MontoProveedor entre la suma total de todos los
    -- montos de la tabla, usando SUM() OVER ()
    CAST(MontoProveedor / SUM(MontoProveedor) OVER () * 100
    AS DECIMAL(5,2)) AS PorcentajeDelTotal
FROM
    Montos
ORDER BY
```

MontoProveedor DESC;

GO

The screenshot shows the Microsoft SQL Server Management Studio (SSMS) interface. The left sidebar lists connections and objects under 'QhatuPERU'. The main area shows a query results grid and the query editor with T-SQL code.

Query Editor:

```
27. Mostrar MontoProveedor y PorcentajeDelTotal. (SUM OVER para Total Global)
WITH Nodos AS (
    -- Subconsulta para calcular el monto total (costo) que representa cada proveedor
    SELECT
        P.CodProveedor,
        P.NombreProveedor,
        SUM(OD.PrecioCompra * OD.CantidadSolicitada) AS MontoProveedor
    FROM dboORDEN_DETALLE OD
    JOIN dboPROVEEDOR P ON OD.CodProveedor = P.CodProveedor
    GROUP BY P.CodProveedor, P.NombreProveedor
)
SELECT
    NP.CodProveedor,
    NP.NombreProveedor,
    NP.MontoProveedor,
    NP.MontoProveedor / (SELECT SUM(MontoProveedor) FROM Nodos) AS PorcentajeDelTotal
FROM Nodos NP
```

Results Grid:

CodProveedor	NombreProveedor	MontoProveedor	PorcentajeDelTotal
87	Logistica del Norte S-A-7	2015.00	0.17
85	Logistica del Norte S-A-16	1890.00	0.15
86	Distribuidora Alfa S-A-15	1750.00	0.14
87	Suministros AQP S-A-14	1615.00	0.13
88	Distribuidora Alfa S-A-13	1485.00	0.12
89	Quimicos del Pacifico S-A-12	1360.00	0.11
90	Fabricaciones LIMA S-A-11	1240.00	0.10
91	Metalurgica Peru S-A-18	1125.00	0.09
92	Distribuidora Alfa S-A-9	1015.00	0.08
93	Logistica del Norte S-A-8	910.00	0.07
94	G&G Repres. S-A-7	810.00	0.06
95	Fabricaciones Lima S-A-6	715.00	0.05
96	Suministros AQP S-A-5	625.00	0.05
97	G&G Repres. S-A-4	540.00	0.04
98	Distribuidora Alfa S-A-3	460.00	0.03
99	Fabricaciones Lima S-A-2	385.00	0.03
100	Distribuidora Alfa S-A-1	315.00	0.02

Explicación: La CTE calcula el MontoProveedor. Luego, se usa la función **SUM(MontoProveedor) OVER ()** para obtener la suma total de todos los montos (sin partición), que es el denominador. El monto del proveedor se divide entre este total y se multiplica por 100 para obtener el porcentaje.

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

Enunciado: Mostrar el código, descripción y precio de solo los 3 artículos más caros dentro de cada línea de producto.

Consulta SQL:

```
-- 28. Mostrar solo los 3 artículos más caros por línea.  
(RANK/ROW_NUMBER con CTE y WHERE)  
WITH ArticulosRankeados AS (  
    SELECT  
        CodArticulo,  
        DescripcionArticulo,  
        CodLinea,  
        PrecioProveedor,  
        -- Asigna un rango a los artículos dentro de cada línea,  
        ordenados por precio (más caro primero)  
        ROW_NUMBER() OVER (PARTITION BY CodLinea ORDER  
        BY PrecioProveedor DESC) AS RankPrecioLinea  
    FROM  
        ARTICULO  
)  
SELECT  
    CodArticulo,  
    DescripcionArticulo,  
    CodLinea,  
    PrecioProveedor,  
    RankPrecioLinea  
FROM  
    ArticulosRankeados  
WHERE  
    RankPrecioLinea <= 3  
ORDER BY  
    CodLinea, RankPrecioLinea;  
GO
```

File Edit View Help

CONNECTIONS: AZURE + ⏪ ... sconnected SQLQuery_1.sql - (62) t...U (sa1) SQLQuery_1 - (53) t...U (sa1) Database: QhatuPERU ...

Enable Actual Plan Parse

```
-- 28. Mostrar solo los 3 artículos más caros por línea. (RANK/ROW_NUMBER con CTE y N
WITH ArticulosRankeados AS (
    SELECT
        CodArticulo,
        DescripcionArticulo,
        CodLinea,
        PrecioProveedor,
        -- Asigna un rango a los artículos dentro de cada línea, ordenados por precio
        ROW_NUMBER() OVER (PARTITION BY CodLinea ORDER BY PrecioProveedor DESC) AS Rang
)
```

Results Messages

	CodArticulo	DescripcionArticulo	CodLinea	PrecioProveedor
1	3	Filtro Hepa Mod-2 Sku-1	1	3.00
2	2	Rodamiento Z-20 Mod-3 Sku-2	2	3.50
3	3	Sensor Óptico Mod-4 Sku-3	3	4.00
4	4	Rodamiento Z-20 Mod-5 Sku-4	4	4.50
5	5	Válvula Flujo Mod-6 Sku-5	5	5.00
6	6	Malla Industrial Mod-7 Sku-6	6	5.50
7	7	Válvula Flujo Mod-8 Sku-7	7	6.00
8	8	Filtro Hepa Mod-9 Sku-8	8	6.50
9	9	Válvula Flujo Mod-10 Sku-9	9	7.00
10	10	Malla Industrial Mod-1 Sku-10	10	7.50
11	11	Sensor Óptico Mod-2 Sku-11	11	8.00
12	12	Cable Cat 6 Mod-3 Sku-12	12	8.50
13	13	Malla Industrial Mod-4 Sku-13	13	9.00
14	14	Rodamiento Z-20 Mod-5 Sku-14	14	9.50
15	15	Válvula Flujo Mod-6 Sku-15	15	10.00
16	16	Tornillo Titánio Mod-7 Sku-16	16	10.50

File Edit View Help

CONNECTIONS: AZURE + ⏪ ... sconnected SQLQuery_1.sql - (62) t...U (sa1) SQLQuery_1 - (53) t...U (sa1) Database: QhatuPERU ...

Enable Actual Plan Parse

```
-- 28. Mostrar solo los 3 artículos más caros por línea. (RANK/ROW_NUMBER con CTE y N
WITH ArticulosRankeados AS (
    SELECT
        CodArticulo,
        DescripcionArticulo,
        CodLinea,
        PrecioProveedor,
        -- Asigna un rango a los artículos dentro de cada línea, ordenados por precio
        ROW_NUMBER() OVER (PARTITION BY CodLinea ORDER BY PrecioProveedor DESC) AS Rang
)
```

Results Messages

	CodArticulo	DescripcionArticulo	CodLinea	PrecioProveedor
84	84	Cable Cat 6 Mod-3 Sku-84	84	44.50
85	85	Filtro Hepa Mod-6 Sku-85	85	45.00
86	86	Filtro Hepa Mod-7 Sku-86	86	45.50
87	87	Interruptor Térmico Mod-8 Sk...	87	46.00
88	88	Filtro Hepa Mod-9 Sku-88	88	46.50
89	89	Rodamiento Z-20 Mod-10 Sku-89	89	47.00
90	90	Tornillo Titánio Mod-1 Sku-90	90	47.50
91	91	Rodamiento Z-20 Mod-2 Sku-91	91	48.00
92	92	Tornillo Titánio Mod-3 Sku-92	92	48.50
93	93	Interruptor Térmico Mod-4 Sk...	93	49.00
94	94	Interruptor Térmico Mod-5 Sk...	94	49.50
95	95	Cable Cat 6 Mod-6 Sku-95	95	50.00
96	96	Interruptor Térmico Mod-7 Sk...	96	50.50
97	97	Sensor Óptico Mod-8 Sku-97	97	51.00
98	98	Malla Industrial Mod-9 Sku-98	98	51.50
99	99	Tornillo Titánio Mod-10 Sku-...	99	52.00
100	100	Válvula Flujo Mod-1 Sku-100	100	52.50

Explicación: Se utiliza la función **ROW_NUMBER()** con PARTITION BY CodLinea para asignar un número de fila ordenado por precio. La consulta exterior (SELECT) filtra el resultado usando la cláusula **WHERE RankPrecioLinea <= 3** para seleccionar solo los 3 artículos con la mejor posición (los más caros) en cada línea.

29. Mostrar transportista y su DenseRank por TotalEnviado.

Enunciado: Calcular la cantidad total de artículos enviados por cada transportista y asignarles un rango de clasificación denso (DENSE_RANK) basado en esa cantidad.

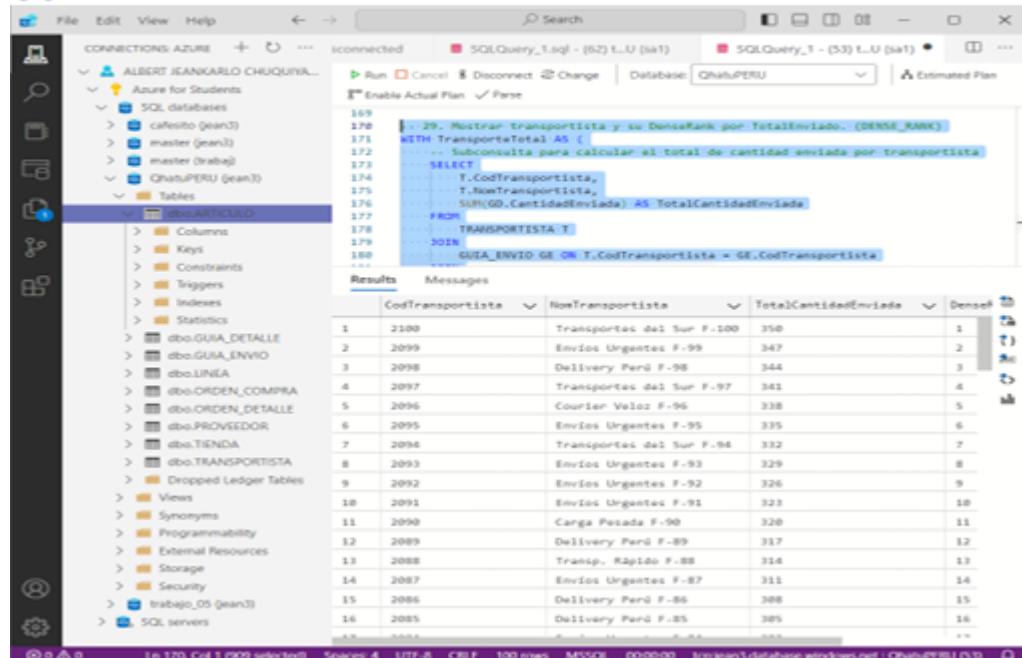
Consulta SQL:

```
-- 29. Mostrar transportista y su DenseRank por TotalEnviado.  
(DENSE_RANK)  
WITH TransporteTotal AS (  
    -- Subconsulta para calcular el total de cantidad enviada por  
    -- transportista  
    SELECT  
        T.CodTransportista,  
        T.NomTransportista,  
        SUM(GD.CantidadEnviada) AS TotalCantidadEnviada  
    FROM  
        TRANSPORTISTA T  
    JOIN  
        GUIA_ENVIO GE ON T.CodTransportista =  
        GE.CodTransportista  
    JOIN  
        GUIA_DETALLE GD ON GE.NumGuia = GD.NumGuia  
    GROUP BY  
        T.CodTransportista, T.NomTransportista  
)  
SELECT  
    CodTransportista,  
    NomTransportista,  
    TotalCantidadEnviada,  
    -- Asigna un rango denso al transportista según la cantidad total  
    -- enviada (empates tienen el mismo rango, no hay saltos)  
    DENSE_RANK() OVER (ORDER BY TotalCantidadEnviada  
DESC) AS DenseRankTotalEnviado  
FROM  
    TransporteTotal
```

ORDER BY

DenseRankTotalEnviado;

GO



File Edit View Help Search

CONNECTIONS: AZURE Run Cancel Disconnect Change Database: QhatuPERU Estimated Plan

ALBERT JEANKARLO CHUQUIYA... Enable Actual Plan Parse

Azure for Students Run Cancel Disconnect Change Database: QhatuPERU Estimated Plan

SQL databases Run Cancel Disconnect Change Database: QhatuPERU Estimated Plan

> cafenito (jean3) Run Cancel Disconnect Change Database: QhatuPERU Estimated Plan

> master (jean3) Run Cancel Disconnect Change Database: QhatuPERU Estimated Plan

> master (trabajo) Run Cancel Disconnect Change Database: QhatuPERU Estimated Plan

> QhatuPERU (jean3) Run Cancel Disconnect Change Database: QhatuPERU Estimated Plan

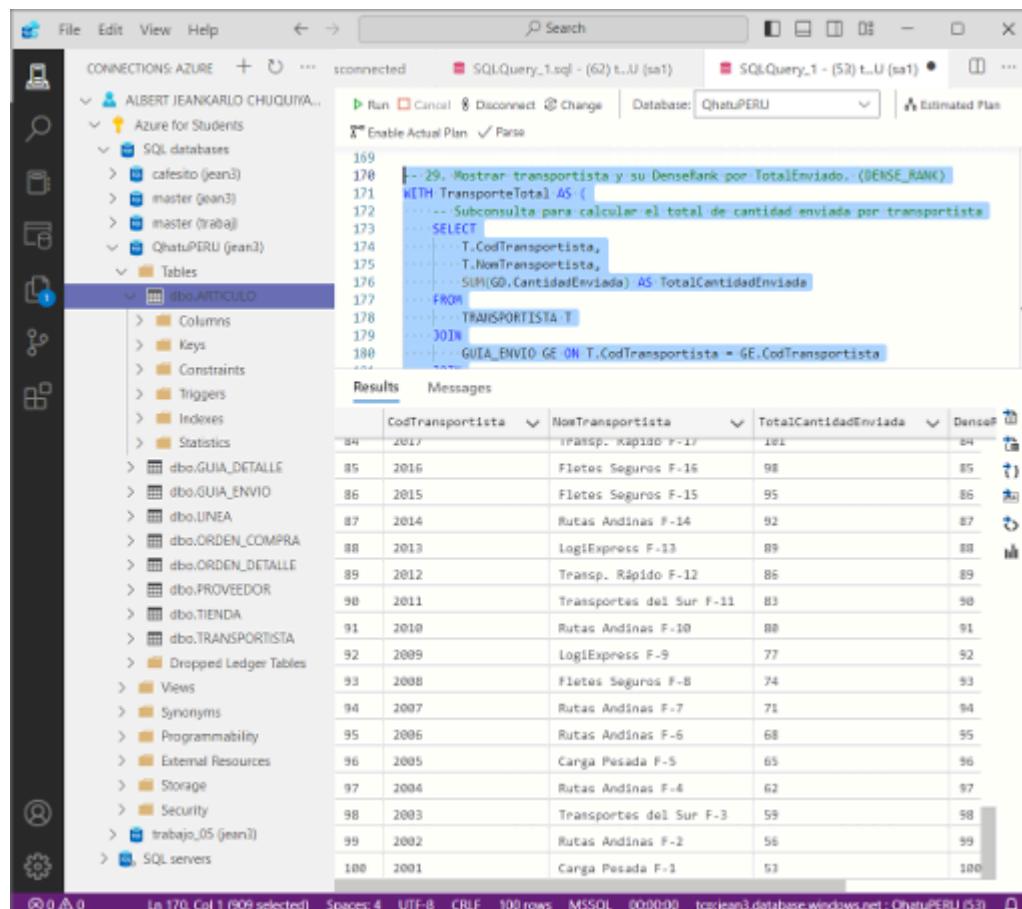
Tables Run Cancel Disconnect Change Database: QhatuPERU Estimated Plan

dbo.TRANSPORTISTA Run Cancel Disconnect Change Database: QhatuPERU Estimated Plan

Results Messages

```
-- 29. Mostrar transportista y su DenseRank por TotalEnviado. (DENSE_RANK)
WITH TransporteTotal AS (
    -- Subconsulta para calcular el total de cantidad enviada por transportista
    SELECT
        T.CodTransportista,
        T.NomTransportista,
        SUM(GD.CantidadEnviada) AS TotalCantidadEnviada
    FROM
        TRANSPORTISTA T
    JOIN
        GUIA_ENVIO GE ON T.CodTransportista = GE.CodTransportista)
```

	CodTransportista	NomTransportista	TotalCantidadEnviada	DenseR
1	2100	Transportes del Sur F-100	350	1
2	2099	Envios Urgentes F-99	347	2
3	2098	Delivery Perú F-98	344	3
4	2097	Transportes del Sur F-97	341	4
5	2096	Courier Veloz F-96	338	5
6	2095	Envios Urgentes F-95	335	6
7	2094	Transportes del Sur F-94	332	7
8	2093	Envios Urgentes F-93	329	8
9	2092	Envios Urgentes F-92	326	9
10	2091	Envios Urgentes F-91	323	10
11	2090	Carga Pesada F-90	320	11
12	2089	Delivery Perú F-89	317	12
13	2088	Transp. Rápido F-88	314	13
14	2087	Envios Urgentes F-87	311	14
15	2086	Delivery Perú F-86	308	15
16	2085	Delivery Perú F-85	305	16
...



File Edit View Help Search

CONNECTIONS: AZURE Run Cancel Disconnect Change Database: QhatuPERU Estimated Plan

ALBERT JEANKARLO CHUQUIYA... Enable Actual Plan Parse

Azure for Students Run Cancel Disconnect Change Database: QhatuPERU Estimated Plan

SQL databases Run Cancel Disconnect Change Database: QhatuPERU Estimated Plan

> cafenito (jean3) Run Cancel Disconnect Change Database: QhatuPERU Estimated Plan

> master (jean3) Run Cancel Disconnect Change Database: QhatuPERU Estimated Plan

> master (trabajo) Run Cancel Disconnect Change Database: QhatuPERU Estimated Plan

> QhatuPERU (jean3) Run Cancel Disconnect Change Database: QhatuPERU Estimated Plan

Tables Run Cancel Disconnect Change Database: QhatuPERU Estimated Plan

dbo.ARITCULO Run Cancel Disconnect Change Database: QhatuPERU Estimated Plan

Results Messages

```
-- 29. Mostrar transportista y su DenseRank por TotalEnviado. (DENSE_RANK)
WITH TransporteTotal AS (
    -- Subconsulta para calcular el total de cantidad enviada por transportista
    SELECT
        T.CodTransportista,
        T.NomTransportista,
        SUM(GD.CantidadEnviada) AS TotalCantidadEnviada
    FROM
        TRANSPORTISTA T
    JOIN
        GUIA_ENVIO GE ON T.CodTransportista = GE.CodTransportista)
```

	CodTransportista	NomTransportista	TotalCantidadEnviada	DenseR
84	2017	Transp. Rápido F-17	181	84
85	2016	Fletes Seguros F-16	98	85
86	2015	Fletes Seguros F-15	95	86
87	2014	Rutas Andinas F-14	92	87
88	2013	LogiExpress F-13	89	88
89	2012	Transp. Rápido F-12	86	89
90	2011	Transportes del Sur F-11	83	90
91	2010	Rutas Andinas F-10	80	91
92	2009	LogiExpress F-9	77	92
93	2008	Fletes Seguros F-8	74	93
94	2007	Rutas Andinas F-7	71	94
95	2006	Rutas Andinas F-6	68	95
96	2005	Carga Pesada F-5	65	96
97	2004	Rutas Andinas F-4	62	97
98	2003	Transportes del Sur F-3	59	98
99	2002	Rutas Andinas F-2	56	99
100	2001	Carga Pesada F-1	53	100

Explicación: La CTE calcula el total enviado por cada transportista. Luego, la función **DENSE_RANK()** se aplica para clasificar a los transportistas. A diferencia de RANK(), si hay empates, el siguiente rango no salta números, manteniendo una secuencia continua de clasificación.

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

Enunciado: Mostrar el valor total de cada guía de envío y calcular el valor acumulado total que esa guía ha representado para su respectiva tienda, ordenado por fecha de salida.

Consulta SQL:

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

SELECT

GE.NumGuia,

GE.CodTienda,

CAST(GE.FechaSalida AS DATE) AS FechaSalida,

-- Suma el valor total de la guía actual

SUM(GD.CantidadEnviada * GD.PrecioVenta) AS ValorGuia,

-- Suma acumulada de los valores de las guías de esa tienda,

ordenadas por fecha -- 30. Mostrar por guía la suma acumulada por tienda hasta esa guía (ordenada por FechaSalida).

SELECT

GE.NumGuia,

GE.CodTienda,

CAST(GE.FechaSalida AS DATE) AS FechaSalida,

-- Suma el valor total de la guía actual (Agregación)

SUM(GD.CantidadEnviada * GD.PrecioVenta) AS ValorGuia,

-- Suma acumulada de los valores de las guías de esa tienda,
ordenadas por fecha de salida (Función de Ventana)

CAST(SUM(SUM(GD.CantidadEnviada * GD.PrecioVenta))

OVER (

PARTITION BY GE.CodTienda

ORDER BY GE.FechaSalida

ROWS BETWEEN UNBOUNDED PRECEDING AND

CURRENT ROW

) AS MONEY) AS AcumuladoVentasTienda

```
FROM
    GUIA_ENVIO GE
JOIN
    GUIA_DETALLE GD ON GE.NumGuia = GD.NumGuia
GROUP BY
    GE.NumGuia,
    GE.CodTienda,
    GE.FechaSalida -- <--- ¡CORRECCIÓN APLICADA AQUÍ!
ORDER BY
    GE.CodTienda, FechaSalida;
GO
de salida
    CAST(SUM(SUM(GD.CantidadEnviada * GD.PrecioVenta))
OVER (
    PARTITION BY GE.CodTienda
    ORDER BY GE.FechaSalida
    ROWS BETWEEN UNBOUNDED PRECEDING AND
    CURRENT ROW
) AS MONEY) AS AcumuladoVentasTienda
FROM
    GUIA_ENVIO GE
JOIN
    GUIA_DETALLE GD ON GE.NumGuia = GD.NumGuia
GROUP BY
    GE.NumGuia, GE.CodTienda, CAST(GE.FechaSalida AS DATE)
ORDER BY
    GE.CodTienda, FechaSalida;
GO
```

File Edit View Help

CONNECTIONS: AZURE + ⏪ ... sconnected SQLQuery_1.sql - (62) t..U (sa1) SQLQuery_1 - (53) t..U (sa1) Database: QchatPERU Estimated Plan

Run Cancel Disconnect Change Database: QchatPERU Estimated Plan

198 ---
199 -- 30. Mostrar por guía la suma acumulada por tienda hasta esa guía (ordenada por FechaSalida)
200 SELECT
201 GE.NumGuia,
202 GE.CodTienda,
203 CAST(GE.FechaSalida AS DATE) AS FechaSalida,
204 -- Suma el valor total de la guía actual (Agregación)
205 SUM(GD.CantidadEnvuada * GD.PrecioVenta) AS ValorGuia,
206 -- Suma acumulada de los valores de las guías de esa tienda, ordenadas por fecha
207 -- CAST(SUM(SUM(GD.CantidadEnvuada * GD.PrecioVenta)) OVER (PARTITION BY GE.CodTienda ORDER BY GE.FechaSalida)) AS AcumuladoVentasTienda
208
209

Results	Messages					
		NumGuia	CodTienda	FechaSalida	ValorGuia	AcumuladoVentasTienda
1	4001	101	2025-10-30	196.10	196.10	
2	4002	102	2025-10-29	246.40	246.40	
3	4003	103	2025-10-29	300.90	300.90	
4	4004	104	2025-10-28	359.60	359.60	
5	4005	105	2025-10-28	422.50	422.50	
6	4006	106	2025-10-27	489.60	489.60	
7	4007	107	2025-10-27	560.90	560.90	
8	4008	108	2025-10-26	636.40	636.40	
9	4009	109	2025-10-26	716.10	716.10	
10	4010	110	2025-10-25	800.00	800.00	
11	4011	111	2025-10-25	888.10	888.10	
12	4012	112	2025-10-24	980.40	980.40	
13	4013	113	2025-10-24	1076.90	1076.90	
14	4014	114	2025-10-23	1177.60	1177.60	
15	4015	115	2025-10-23	1282.50	1282.50	
16	4016	116	2025-10-22	1391.60	1391.60	
17	4017	117	2025-10-22	1500.00	1500.00	

Ln 200, Col 1 (892 selected) Spaces: 4 UTF-8 CRLF 100 rows MSSQL 00:00:00 top:jean3.database.windows.net : QchatPERU (53)

File Edit View Help

CONNECTIONS: AZURE + ⏪ ... sconnected SQLQuery_1.sql - (62) t..U (sa1) SQLQuery_1 - (53) t..U (sa1) Database: QchatPERU Estimated Plan

Run Cancel Disconnect Change Database: QchatPERU Estimated Plan

198 ---
199 -- 30. Mostrar por guía la suma acumulada por tienda hasta esa guía (ordenada por FechaSalida)
200 SELECT
201 GE.NumGuia,
202 GE.CodTienda,
203 CAST(GE.FechaSalida AS DATE) AS FechaSalida,
204 -- Suma el valor total de la guía actual (Agregación)
205 SUM(GD.CantidadEnvuada * GD.PrecioVenta) AS ValorGuia,
206 -- Suma acumulada de los valores de las guías de esa tienda, ordenadas por fecha
207 -- CAST(SUM(SUM(GD.CantidadEnvuada * GD.PrecioVenta)) OVER (PARTITION BY GE.CodTienda ORDER BY GE.FechaSalida)) AS AcumuladoVentasTienda
208
209

Results	Messages					
		NumGuia	CodTienda	FechaSalida	ValorGuia	AcumuladoVentasTienda
84	4084	184	2025-09-18	18863.00	18863.00	
85	4085	185	2025-09-18	19062.50	19062.50	
86	4086	186	2025-09-17	19465.60	19465.60	
87	4087	187	2025-09-17	19872.90	19872.90	
88	4088	188	2025-09-17	20284.40	20284.40	
89	4089	189	2025-09-16	20700.10	20700.10	
90	4090	190	2025-09-15	21120.00	21120.00	
91	4091	191	2025-09-15	21544.10	21544.10	
92	4092	192	2025-09-14	21972.40	21972.40	
93	4093	193	2025-09-14	22404.90	22404.90	
94	4094	194	2025-09-13	22841.60	22841.60	
95	4095	195	2025-09-13	23282.50	23282.50	
96	4096	196	2025-09-12	23727.60	23727.60	
97	4097	197	2025-09-12	24176.90	24176.90	
98	4098	198	2025-09-11	24630.40	24630.40	
99	4099	199	2025-09-11	25088.10	25088.10	
100	4100	200	2025-09-10	25900.00	25900.00	

Ln 200, Col 1 (892 selected) Spaces: 4 UTF-8 CRLF 100 rows MSSQL 00:00:00 top:jean3.database.windows.net : QchatPERU (53)

Explicación: La consulta primero agrupa para obtener el ValorGuia. Luego, usa la función **SUM()** anidada con OVER. El **PARTITION BY CodTienda** reinicia el cálculo acumulado para cada tienda, y el marco de ventana **ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW** asegura que se sumen todas las guías de esa tienda desde la primera fecha hasta la fecha de la guía actual.

