

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 línea 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 PrecioAnteriorMismoProveedor usando LAG.
26. Añadir columna CantidadPorLinea a cada artículo.
27. Mostrar MontoProveedor y PorcentajeDelTotal.
28. Mostrar solo los 3 artículos más caros por línea.
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 línea 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

File Edit View Help Search

CONNECTIONS: AZURE + Refresh ...

connected SQLQuery_1.sql - (62) L.U (sa1) SQLQuery_1 - (53) L.U (sa1)

Run Cancel Disconnect Change Database: ChutuPERU Estimated Plan

Enable Actual Plan Pass

```

1  -- .....
2  -- 9. CONSULTAS AVANZADAS (Funciones de Ventana - Puntos 21 al 30)
3  -- .....
4
5  -- 21. Asignar posición por línea ordenada por precio. (ROW_NUMBER)
6  SELECT
7      CodArtículo,
8      DescripciónArtículo,
9      CodLínea,
10     PrecioProveedor,
11     ROW_NUMBER() OVER (PARTITION BY CodLínea ORDER BY PrecioProveedor DESC) AS Posición
12 FROM .....

```

Results Messages

	CodAr...	DescripciónArtículo	CodLínea	PrecioProv...	Posi...
1	1	Filtro Hepa Rod-2 Sku-1	1	3.00	1
2	2	Rodamiento 2-20 Rod-3 Sku-2	2	3.50	1
3	3	Sensor Óptico Rod-4 Sku-3	3	4.00	1
4	4	Rodamiento 2-20 Rod-5 Sku-4	4	4.50	1
5	5	Válvula Fluido Rod-6 Sku-5	5	5.00	1
6	6	Halla Industrial Rod-7 Sku-6	6	5.50	1
7	7	Válvula Fluido Rod-8 Sku-7	7	6.00	1
8	8	Filtro Hepa Rod-9 Sku-8	8	6.50	1
9	9	Válvula Fluido Rod-10 Sku-9	9	7.00	1
10	10	Halla Industrial Rod-1 Sku-10	10	7.50	1
11	11	Sensor Óptico Rod-2 Sku-11	11	8.00	1
12	12	Cable Cat 6 Rod-3 Sku-12	12	8.50	1
13	13	Halla Industrial Rod-4 Sku-13	13	9.00	1
14	14	Rodamiento 2-20 Rod-5 Sku-14	14	9.50	1
15	15	Válvula Fluido Rod-6 Sku-15	15	10.00	1
16	16	Tornillo Titaneo Rod-7 Sku-16	16	10.50	1

In 5, Col 1 (322 selected) Spaces: 4 UTF-8 CR/LF 100 rows MSSQL 000000 tcpipen3.database.windows.net - ChutuPERU (53)

File Edit View Help Search

CONNECTIONS: AZURE + Refresh ...

connected SQLQuery_1.sql - (62) L.U (sa1) SQLQuery_1 - (53) L.U (sa1)

Run Cancel Disconnect Change Database: ChutuPERU Estimated Plan

Enable Actual Plan Pass

```

1  -- .....
2  -- 9. CONSULTAS AVANZADAS (Funciones de Ventana - Puntos 21 al 30)
3  -- .....
4
5  -- 21. Asignar posición por línea ordenada por precio. (ROW_NUMBER)
6  SELECT
7      CodArtículo,
8      DescripciónArtículo,
9      CodLínea,
10     PrecioProveedor,
11     ROW_NUMBER() OVER (PARTITION BY CodLínea ORDER BY PrecioProveedor DESC) AS Posición
12 FROM .....

```

Results Messages

	CodAr...	DescripciónArtículo	CodLínea	PrecioProv...	Posi...
84	84	Cable Cat 6 Rod-9 Sku-84	84	44.00	1
85	85	Filtro Hepa Rod-6 Sku-85	85	45.00	1
86	86	Filtro Hepa Rod-7 Sku-86	86	45.50	1
87	87	Interruptor Termico Rod-8 Sku...	87	46.00	1
88	88	Filtro Hepa Rod-9 Sku-88	88	46.50	1
89	89	Rodamiento 2-20 Rod-10 Sku-89	89	47.00	1
90	90	Tornillo Titaneo Rod-1 Sku-90	90	47.50	1
91	91	Rodamiento 2-20 Rod-2 Sku-91	91	48.00	1
92	92	Tornillo Titaneo Rod-3 Sku-92	92	48.50	1
93	93	Interruptor Termico Rod-4 Sku...	93	49.00	1
94	94	Interruptor Termico Rod-5 Sku...	94	49.50	1
95	95	Cable Cat 6 Rod-6 Sku-95	95	50.00	1
96	96	Interruptor Termico Rod-7 Sku...	96	50.50	1
97	97	Sensor Óptico Rod-8 Sku-97	97	51.00	1
98	98	Halla Industrial Rod-9 Sku-98	98	51.50	1
99	99	Tornillo Titaneo Rod-10 Sku...	99	52.00	1
100	100	Válvula Fluido Rod-1 Sku-100	100	52.50	1

In 5, Col 1 (322 selected) Spaces: 4 UTF-8 CR/LF 100 rows MSSQL 000000 tcpipen3.database.windows.net - ChutuPERU (53)

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

connected SQLQuery_1.sql - (52) T.U (part) SQLQuery_1 - (53) T.U (part)

Database: QhentuPERU

Run Cancel Disconnect Change Estimated Plan

Enable Actual Plan Parse

```

18 ---
19
20 -- 22. Calcular costo por orden y su RANK (RankCosto)...(RANK)
21 WITH OrdenCosto AS (
22     -- Subconsulta para calcular el costo total de cada orden
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	3099	31500.00	1
2	3099	28940.00	2
3	3098	28885.00	3
4	3097	29435.00	4
5	3096	29290.00	5
6	3095	28750.00	6
7	3094	28215.00	7
8	3093	27685.00	8
9	3092	27160.00	9
10	3091	26640.00	10
11	3090	26125.00	11
12	3089	25615.00	12
13	3088	25110.00	13
14	3087	24610.00	14
15	3086	24115.00	15
16	3085	23625.00	16
17	3084	23140.00	17

Ln 20, Col 1 (459 selected) Spaces: 4 UTF-8 CRLF 100 rows MSSQL 00:00:00 top(jean3).database.windows.net : QhentuPERU (53)

File Edit View Help Search

connected SQLQuery_1.sql - (52) T.U (part) SQLQuery_1 - (53) T.U (part)

Database: QhentuPERU

Run Cancel Disconnect Change Estimated Plan

Enable Actual Plan Parse

```

18 ---
19
20 -- 22. Calcular costo por orden y su RANK (RankCosto)...(RANK)
21 WITH OrdenCosto AS (
22     -- Subconsulta para calcular el costo total de cada orden
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	3017	2895.00	84
85	3016	1890.00	85
86	3015	1750.00	86
87	3014	1615.00	87
88	3013	1485.00	88
89	3012	1360.00	89
90	3011	1240.00	90
91	3010	1125.00	91
92	3009	1015.00	92
93	3008	910.00	93
94	3007	810.00	94
95	3006	715.00	95
96	3005	625.00	96
97	3004	540.00	97
98	3003	460.00	98
99	3002	385.00	99
100	3001	315.00	100

Ln 20, Col 1 (459 selected) Spaces: 4 UTF-8 CRLF 100 rows MSSQL 00:00:00 top(jean3).database.windows.net : QhentuPERU (53)

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 Search

CONNECTIONS: AZURE + ... connected SQLQuery_1.sql - (52) L.U (sa7) SQLQuery_1 - (52) L.U (sa7)

Run Cancel Disconnect Change Database ChatusPERU Estimated Plan

Enable Actual Plan ✓ Force

```

37 ORDER BY
38 RankConto;
39 GO
40
41 ...
42
43 -- 25. Mostrar TotalDía y AcumuladoVentas ordenado por fecha. (SUM OVER con Rango de 9
44 SELECT
45 CAST(GET.FechaSalida AS DATE) AS DiaSalida,
46 SUM(SD.CantidadEnviada * SD.PrecioVenta) AS TotalDía,
47 -- Calcule la suma acumulada del TotalDía hasta la fecha actual
48 SUM(SUM(SD.CantidadEnviada * SD.PrecioVenta) OVER (ORDER BY CAST(GET.FechaSalida
49 AS DATE)

```

Results Messages

	DiaSalida	TotalDía	AcumuladoVentas
1	2025-09-10	25000.00	25000.00
2	2025-09-11	49750.50	75000.50
3	2025-09-12	47900.50	122923.00
4	2025-09-13	44324.50	169647.50
5	2025-09-14	44377.50	214024.00
6	2025-09-15	42644.50	256668.50
7	2025-09-16	40904.50	297573.00
8	2025-09-17	39338.50	337011.50
9	2025-09-18	37726.50	374737.00
10	2025-09-19	36147.50	410884.50
11	2025-09-20	34602.50	445487.00
12	2025-09-21	33090.50	478577.50
13	2025-09-22	31612.50	510190.00
14	2025-09-23	30168.50	540358.50
15	2025-09-24	28757.50	569115.00
16	2025-09-25	27380.50	596495.50
17	2025-09-26	26030.50	622526.00

File Edit View Help Search

CONNECTIONS: AZURE + ... connected SQLQuery_1.sql - (52) L.U (sa7) SQLQuery_1 - (52) L.U (sa7)

Run Cancel Disconnect Change Database ChatusPERU Estimated Plan

Enable Actual Plan ✓ Force

```

37 ORDER BY
38 RankConto;
39 GO
40
41 ...
42
43 -- 25. Mostrar TotalDía y AcumuladoVentas ordenado por fecha. (SUM OVER con Rango de 9
44 SELECT
45 CAST(GET.FechaSalida AS DATE) AS DiaSalida,
46 SUM(SD.CantidadEnviada * SD.PrecioVenta) AS TotalDía,
47 -- Calcule la suma acumulada del TotalDía hasta la fecha actual
48 SUM(SUM(SD.CantidadEnviada * SD.PrecioVenta) OVER (ORDER BY CAST(GET.FechaSalida
49 AS DATE)

```

Results Messages

	DiaSalida	TotalDía	AcumuladoVentas
35	2025-10-14	7597.50	899737.00
36	2025-10-15	6892.50	968629.50
37	2025-10-16	6220.50	932850.00
38	2025-10-17	5582.50	938432.50
39	2025-10-18	4978.50	923410.00
40	2025-10-19	4487.50	927817.50
41	2025-10-20	3870.50	931688.00
42	2025-10-21	3366.50	935054.50
43	2025-10-22	2894.50	937951.00
44	2025-10-23	2460.50	940411.50
45	2025-10-24	2057.50	942468.00
46	2025-10-25	1688.50	944156.50
47	2025-10-26	1351.50	945508.00
48	2025-10-27	1050.50	946559.50
49	2025-10-28	782.50	947341.00
50	2025-10-29	547.50	947888.50
51	2025-10-30	196.50	948085.00

La 43, Col 1 (594 selected) Spaces: 4 UTF-8 CRLF 51 rows MSSQL 00:00:00 http://jean3.database.windows.net/ ChatusPERU (52)

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 línea
    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 | SQLQuery_1.sql - (52) L.U (sa1) | SQLQuery_1 - (52) L.U (sa1) | Estimated Plan

Run Cancel Disconnect Change Database: ChutuPERU

Enable Actual Plan | Parse

```

57 60
58
59 ...
60
61 -- 34. Calcular promedio móvil para stock. (AVG OVER con Marco de Ventana)
62 SELECT
63     CodArtículo,
64     DescripciónArtículo,
65     CodLinea,
66     StockActual,
67     ... Promedio móvil: promedio del stock del artículo anterior, el actual y el siguiente
68     CAST(AVG(StockActual) OVER (

```

Results Messages

	CodArtículo	DescripciónArtículo	CodLinea	StockActual
1	1	Filtro Heka Mod-2 Sku-1	1	52
2	2	Rodamiento 2-20 Mod-3 Sku-2	2	54
3	3	Sensor Óptico Mod-4 Sku-3	3	56
4	4	Rodamiento 2-20 Mod-5 Sku-4	4	58
5	5	Valvula Flujo Mod-6 Sku-5	5	60
6	6	Malla Industrial Mod-7 Sku-6	6	62
7	7	Valvula Flujo Mod-8 Sku-7	7	64
8	8	Filtro Heka Mod-9 Sku-8	8	66
9	9	Valvula Flujo Mod-10 Sku-9	9	68
10	10	Malla Industrial Mod-1 Sku-10	10	70
11	11	Sensor Óptico Mod-2 Sku-11	11	72
12	12	Cable Cat 6 Mod-3 Sku-12	12	74
13	13	Malla Industrial Mod-4 Sku-13	13	76
14	14	Rodamiento 2-20 Mod-5 Sku-14	14	78
15	15	Valvula Flujo Mod-6 Sku-15	15	80
16	16	Tornillo Titania Mod-7 Sku-16	16	82

Ln 61, Col 1 (525 selected) | Space: 4 | UTF-8 | CR/LF | 100 rows | MSSQL | 00:00:00 | tcp:gran1.database.windows.net : ChutuPERU (5.3)

File Edit View Help | SQLQuery_1.sql - (52) L.U (sa1) | SQLQuery_1 - (53) L.U (sa1) | Estimated Plan

Run Cancel Disconnect Change Database: ChutuPERU

Enable Actual Plan | Parse

```

57 60
58
59 ...
60
61 -- 34. Calcular promedio móvil para stock. (AVG OVER con Marco de Ventana)
62 SELECT
63     CodArtículo,
64     DescripciónArtículo,
65     CodLinea,
66     StockActual,
67     ... Promedio móvil: promedio del stock del artículo anterior, el actual y el siguiente
68     CAST(AVG(StockActual) OVER (

```

Results Messages

	CodArtículo	DescripciónArtículo	CodLinea	StockActual
84	84	Cable Cat 6 Mod-3 Sku-84	84	218
85	85	Filtro Heka Mod-6 Sku-85	85	220
86	86	Filtro Heka Mod-7 Sku-86	86	222
87	87	Interrupcion Termico Mod-8 Sku...	87	224
88	88	Filtro Heka Mod-9 Sku-88	88	226
89	89	Rodamiento 2-20 Mod-10 Sku-89	89	228
90	90	Tornillo Titania Mod-1 Sku-90	90	230
91	91	Rodamiento 2-20 Mod-2 Sku-91	91	232
92	92	Tornillo Titania Mod-3 Sku-92	92	234
93	93	Interrupcion Termico Mod-4 Sku...	93	236
94	94	Interrupcion Termico Mod-5 Sku...	94	238
95	95	Cable Cat 6 Mod-6 Sku-95	95	240
96	96	Interrupcion Termico Mod-7 Sku...	96	242
97	97	Sensor Óptico Mod-8 Sku-97	97	244
98	98	Malla Industrial Mod-9 Sku-98	98	246
99	99	Tornillo Titania Mod-10 Sku...	99	248
100	100	Valvula Flujo Mod-1 Sku-100	100	250

Ln 61, Col 1 (525 selected) | Space: 4 | UTF-8 | CR/LF | 100 rows | MSSQL | 00:00:00 | tcp:gran1.database.windows.net : ChutuPERU (5.3)

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 Search

CONNECTIONS: AZURE + ... connected SQLQuery_1.sql - (62) L.U (sat) SQLQuery_1 - (52) L.U (sat)

Run Cancel Disconnect Change Database: QhutuPERU Estimated Plan

Enable Actual Plan / Para

```

76  -- 25. Mostrar PrecioAnteriorPorProveedor usando LAG. (LAG)
77  SELECT
78  ...
79  ...
80  ...
81  -- 25. Mostrar PrecioAnteriorPorProveedor usando LAG. (LAG)
82  SELECT
83  ...
84  ...
85  ...
86  -- Obtenga el PrecioProveedor del artículo inmediatamente anterior, ordenado por
87  LAG(PrecioProveedor, 1, 0) OVER (PARTITION BY CodProveedor ORDER BY CodArticulo)
88  FROM

```

Results Messages

	CodArticulo	CodProveedor	PrecioProveedor	PrecioAnteriorPorProveedor
1	1	1	3.00	0.00
2	2	2	3.50	0.00
3	3	3	4.00	0.00
4	4	4	4.50	0.00
5	5	5	5.00	0.00
6	6	6	5.50	0.00
7	7	7	6.00	0.00
8	8	8	6.50	0.00
9	9	9	7.00	0.00
10	10	10	7.50	0.00
11	11	11	8.00	0.00
12	12	12	8.50	0.00
13	13	13	9.00	0.00
14	14	14	9.50	0.00
15	15	15	10.00	0.00
16	16	16	10.50	0.00
17	17	17	11.00	0.00

Ln 81, Col 1 (445) selected Spans: 4 UTF-8 CRLF 100 rows MSSQL 00:00:00 top.jeanb.database.windows.net : QhutuPERU (5.3)

File Edit View Help Search

CONNECTIONS: AZURE + ... connected SQLQuery_1.sql - (62) L.U (sat) SQLQuery_1 - (53) L.U (sat)

Run Cancel Disconnect Change Database: QhutuPERU Estimated Plan

Enable Actual Plan / Para

```

76  -- 25. Mostrar PrecioAnteriorPorProveedor usando LAG. (LAG)
77  SELECT
78  ...
79  ...
80  ...
81  -- 25. Mostrar PrecioAnteriorPorProveedor usando LAG. (LAG)
82  SELECT
83  ...
84  ...
85  ...
86  -- Obtenga el PrecioProveedor del artículo inmediatamente anterior, ordenado por
87  LAG(PrecioProveedor, 1, 0) OVER (PARTITION BY CodProveedor ORDER BY CodArticulo)
88  FROM

```

Results Messages

	CodArticulo	CodProveedor	PrecioProveedor	PrecioAnteriorPorProveedor
84	84	84	44.50	0.00
85	85	85	45.00	0.00
86	86	86	45.50	0.00
87	87	87	46.00	0.00
88	88	88	46.50	0.00
89	89	89	47.00	0.00
90	90	90	47.50	0.00
91	91	91	48.00	0.00
92	92	92	48.50	0.00
93	93	93	49.00	0.00
94	94	94	49.50	0.00
95	95	95	50.00	0.00
96	96	96	50.50	0.00
97	97	97	51.00	0.00
98	98	98	51.50	0.00
99	99	99	52.00	0.00
100	100	100	52.50	0.00

Ln 81, Col 1 (445) selected Spans: 4 UTF-8 CRLF 100 rows MSSQL 00:00:00 top.jeanb.database.windows.net : QhutuPERU (5.3)

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 Search

CONNECTIONS: AZURE + ...

ALBERT JEANKARLO CHUQUIYA...

Azure for Students

SQL databases

cafesito (jean3)

master (jean3)

master (trabaj)

QhatuPERU (jean3)

Tables

dbo.ARTICULO

Columns

Keys

Constraints

Triggers

Indexes

Statistics

dbo.GUÍA_DETALLE

dbo.GUÍA_ENVIO

dbo.LINEA

dbo.ORDEN_COMPRA

dbo.ORDEN_DETALLE

dbo.PROVEEDOR

dbo.TIENDA

dbo.TRANSPORTISTA

Dropped Ledger Tables

Views

Synonyms

Programmability

External Resources

Storage

Security

trabajo_05 (jean3)

SQL servers

Run Cancel Disconnect Change Database: QhatuPERU Estimated Plan

SQLQuery_1.sql - (62) L.U (sa1)

SQLQuery_1 - (53) L.U (sa1)

Enable Actual Plan Parse

```

26. Añadir columna CantidadPorLinea a cada artículo. (COUNT OVER)
SELECT
  CodArtículo,
  DescripciónArtículo,
  CodLinea,
  PrecioProveedor,
  ... Cuenta el total de artículos que existen en la misma línea
  COUNT(CodArtículo) OVER (PARTITION BY CodLinea) AS CantidadPorLinea
FROM
  ARTICULO

```

Results Messages

	CodArtículo	DescripciónArtículo	CodLinea	PrecioProveedor
1	1	Filtro Heka Mod-2 Sku-1	1	3.00
2	2	Rodamiento 2-20 Mod-3 Sku-2	2	3.50
3	3	Sensor Óptico Mod-4 Sku-3	3	4.00
4	4	Rodamiento 2-20 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 Heka 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 2-20 Mod-5 Sku-14	14	9.50
15	15	Válvula Flujo Mod-6 Sku-15	15	10.00
16	16	Tornillo Titanio Mod-7 Sku-16	16	10.50

Ln 96, Col 1 (348 selected) Spaces: 4 UTF-8 CRLF 100 rows MSSQL 00:00:00 tcp:jean3.database.windows.net : QhatuPERU (53)

File Edit View Help Search

CONNECTIONS: AZURE + ...

ALBERT JEANKARLO CHUQUIYA...

Azure for Students

SQL databases

cafesito (jean3)

master (jean3)

master (trabaj)

QhatuPERU (jean3)

Tables

dbo.ARTICULO

Columns

Keys

Constraints

Triggers

Indexes

Statistics

dbo.GUÍA_DETALLE

dbo.GUÍA_ENVIO

dbo.LINEA

dbo.ORDEN_COMPRA

dbo.ORDEN_DETALLE

dbo.PROVEEDOR

dbo.TIENDA

dbo.TRANSPORTISTA

Dropped Ledger Tables

Views

Synonyms

Programmability

External Resources

Storage

Security

trabajo_05 (jean3)

SQL servers

Run Cancel Disconnect Change Database: QhatuPERU Estimated Plan

SQLQuery_1.sql - (62) L.U (sa1)

SQLQuery_1 - (53) L.U (sa1)

Enable Actual Plan Parse

```

26. Añadir columna CantidadPorLinea a cada artículo. (COUNT OVER)
SELECT
  CodArtículo,
  DescripciónArtículo,
  CodLinea,
  PrecioProveedor,
  ... Cuenta el total de artículos que existen en la misma línea
  COUNT(CodArtículo) OVER (PARTITION BY CodLinea) AS CantidadPorLinea
FROM
  ARTICULO

```

Results Messages

	CodArtículo	DescripciónArtículo	CodLinea	PrecioProveedor
84	84	Cable Cat 6 Mod-3 Sku-84	84	44.50
85	85	Filtro Heka Mod-6 Sku-85	85	45.00
86	86	Filtro Heka Mod-7 Sku-86	86	45.50
87	87	Interruptor Termico Mod-8 Sku...	87	46.00
88	88	Filtro Heka Mod-9 Sku-88	88	46.50
89	89	Rodamiento 2-20 Mod-10 Sku-89	89	47.00
90	90	Tornillo Titanio Mod-1 Sku-90	90	47.50
91	91	Rodamiento 2-20 Mod-2 Sku-91	91	48.00
92	92	Tornillo Titanio Mod-3 Sku-92	92	48.50
93	93	Interruptor Termico Mod-4 Sku...	93	49.00
94	94	Interruptor Termico Mod-5 Sku...	94	49.50
95	95	Cable Cat 6 Mod-6 Sku-95	95	50.00
96	96	Interruptor Termico Mod-7 Sku...	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 Titanio Mod-10 Sku-...	99	52.00
100	100	Válvula Flujo Mod-1 Sku-100	100	52.50

Ln 96, Col 1 (348 selected) Spaces: 4 UTF-8 CRLF 100 rows MSSQL 00:00:00 tcp:jean3.database.windows.net : QhatuPERU (53)

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 SQL Server Enterprise Manager interface. The left pane displays the database structure for 'QhutuPERU'. The center pane shows the execution plan for the query '27. Mostrar MontoProveedor y PorcentajeDelTotal. (SUM OVER para Total Global)'. The right pane displays the results of the query, which is a table with 5 columns: 'CodProveedor', 'NonProveedor', 'MontoProveedor', and 'PorcentajeDe'. The results are sorted by 'MontoProveedor' in descending order.

CodProveedor	NonProveedor	MontoProveedor	PorcentajeDe	
1	100	Químicos del Pacífico S-A-100	31500.00	2.65
2	99	TecnoIndustrial S-A-99	30940.00	2.60
3	98	G&G Repres. S-A-98	30385.00	2.56
4	97	TecnoIndustrial S-A-97	29835.00	2.51
5	96	Logística del Norte S-A-96	29290.00	2.46
6	95	Fabricaciones Lima S-A-95	28750.00	2.42
7	94	Distribuidora Alfa S-A-94	28215.00	2.37
8	93	Distribuidora Alfa S-A-93	27685.00	2.33
9	92	Metalúrgica Perú S-A-92	27160.00	2.28
10	91	Químicos del Pacífico S-A-91	26640.00	2.24
11	90	Suministros AQP S-A-90	26125.00	2.20
12	89	Insumos Global S-A-89	25615.00	2.15
13	88	Fabricaciones Lima S-A-88	25110.00	2.11
14	87	Logística del Norte S-A-87	24610.00	2.07
15	86	Metalúrgica Perú S-A-86	24115.00	2.03
16	85	G&G Repres. S-A-85	23625.00	1.99

The screenshot shows the SQL Server Enterprise Manager interface. The left pane displays the database structure for 'QhutuPERU'. The center pane shows the execution plan for the query '27. Mostrar MontoProveedor y PorcentajeDelTotal. (SUM OVER para Total Global)'. The right pane displays the results of the query, which is a table with 5 columns: 'CodProveedor', 'NonProveedor', 'MontoProveedor', and 'PorcentajeDe'. The results are sorted by 'MontoProveedor' in descending order.

CodProveedor	NonProveedor	MontoProveedor	PorcentajeDe	
84	17	Químicos del Pacífico S-A-17	2035.00	0.27
85	16	Logística del Norte S-A-16	1890.00	0.25
86	15	Distribuidora Alfa S-A-15	1750.00	0.24
87	14	Suministros AQP S-A-14	1615.00	0.22
88	13	Distribuidora Alfa S-A-13	1485.00	0.21
89	12	Químicos del Pacífico S-A-12	1360.00	0.19
90	11	Fabricaciones Lima S-A-11	1240.00	0.18
91	10	Metalúrgica Perú S-A-10	1125.00	0.16
92	9	Distribuidora Alfa S-A-9	1015.00	0.14
93	8	Logística del Norte S-A-8	910.00	0.13
94	7	G&G Repres. S-A-7	810.00	0.11
95	6	Fabricaciones Lima S-A-6	715.00	0.10
96	5	Suministros AQP S-A-5	625.00	0.09
97	4	G&G Repres. S-A-4	540.00	0.08
98	3	Distribuidora Alfa S-A-3	460.00	0.07
99	2	Fabricaciones Lima S-A-2	385.00	0.06
100	1	Distribuidora Alfa S-A-1	315.00	0.05

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 + ...

Albert JEANKARLO CHUQUIYA...

Azure for Students

SQL databases

cafeito (jean3)

master (jean3)

master (trabaj)

QhatuPERU (jean3)

Tables

dbo.ARTICULO

Columns

Keys

Constraints

Triggers

Indexes

Statistics

dbo.GUÍA_DETALLE

dbo.GUÍA_ENVIO

dbo.LINEA

dbo.ORDEN_COMPRA

dbo.ORDEN_DETALLE

dbo.PROVEEDOR

dbo.TIENDA

dbo.TRANSPORTISTA

Dropped Ledger Tables

Views

Synonyms

Programmability

External Resources

Storage

Security

trabajo_05 (jean3)

SQL servers

SQLQuery_1.sql - (62) T.U (sa1)

Run Cancel Disconnect Change Database: QhatuPERU Estimated Plan

Enable Actual Plan Parse

```

139
140
141
142 -- 28. Mostrar solo los 3 artículos más caros por línea. (RANK/ROW_NUMBER con CTE y
143 WITH ArticulosRankados AS (
144     SELECT
145         CodArticulo,
146         DescripcionArticulo,
147         CodLinea,
148         PrecioProveedor,
149         -- Asigna un rango a los artículos dentro de cada línea, ordenados por precio
150         ROW_NUMBER() OVER (PARTITION BY CodLinea ORDER BY PrecioProveedor DESC) AS Ra

```

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 Titánio Mod-7 Sku-16	16	10.50

Ln 142, Col 1 (687 selected) Spaces: 4 UTF-8 CRLF 100 rows MSSQL 00:00:00 tcp:jean3.database.windows.net : QhatuPERU (53)

File Edit View Help

CONNECTIONS: AZURE + ...

Albert JEANKARLO CHUQUIYA...

Azure for Students

SQL databases

cafeito (jean3)

master (jean3)

master (trabaj)

QhatuPERU (jean3)

Tables

dbo.ARTICULO

Columns

Keys

Constraints

Triggers

Indexes

Statistics

dbo.GUÍA_DETALLE

dbo.GUÍA_ENVIO

dbo.LINEA

dbo.ORDEN_COMPRA

dbo.ORDEN_DETALLE

dbo.PROVEEDOR

dbo.TIENDA

dbo.TRANSPORTISTA

Dropped Ledger Tables

Views

Synonyms

Programmability

External Resources

Storage

Security

trabajo_05 (jean3)

SQL servers

SQLQuery_1.sql - (62) T.U (sa1)

Run Cancel Disconnect Change Database: QhatuPERU Estimated Plan

Enable Actual Plan Parse

```

139
140
141
142 -- 28. Mostrar solo los 3 artículos más caros por línea. (RANK/ROW_NUMBER con CTE y
143 WITH ArticulosRankados AS (
144     SELECT
145         CodArticulo,
146         DescripcionArticulo,
147         CodLinea,
148         PrecioProveedor,
149         -- Asigna un rango a los artículos dentro de cada línea, ordenados por precio
150         ROW_NUMBER() OVER (PARTITION BY CodLinea ORDER BY PrecioProveedor DESC) AS Ra

```

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	Interrupción Térmico Mod-8 Sku-87	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	Interrupción Térmico Mod-4 Sku-93	93	49.00
94	94	Interrupción Térmico Mod-5 Sku-94	94	49.50
95	95	Cable Cat 6 Mod-6 Sku-95	95	50.00
96	96	Interrupción Térmico Mod-7 Sku-96	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	99	52.00
100	100	Válvula Flujo Mod-1 Sku-100	100	52.50

Ln 142, Col 1 (687 selected) Spaces: 4 UTF-8 CRLF 100 rows MSSQL 00:00:00 tcp:jean3.database.windows.net : QhatuPERU (53)

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

The screenshot shows the SQL Server Enterprise Manager interface. The left pane displays the database structure, including tables like `dbo.GUIA_DETALLE`, `dbo.GUIA_ENVIO`, `dbo.LINEA`, `dbo.ORDEN_COMPRA`, `dbo.ORDEN_DETALLE`, `dbo.PROVEEDOR`, `dbo.TIENDA`, `dbo.TRANSPORTISTA`, and `dbo.ARTICULO`. The right pane shows the execution plan for a query. The query is as follows:

```

-- 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.NonTransportista,
        SUM(GO.CantidadEnviada) AS TotalCantidadEnviada
    FROM
        TRANSPORTISTA T
    JOIN
        GUIA_ENVIO GE ON T.CodTransportista = GE.CodTransportista
)

```

The results table shows the following data:

CodTransportista	NonTransportista	TotalCantidadEnviada	DenseRank	
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

The screenshot shows the SQL Server Enterprise Manager interface. The left pane displays the database structure, including tables like `dbo.GUIA_DETALLE`, `dbo.GUIA_ENVIO`, `dbo.LINEA`, `dbo.ORDEN_COMPRA`, `dbo.ORDEN_DETALLE`, `dbo.PROVEEDOR`, `dbo.TIENDA`, `dbo.TRANSPORTISTA`, and `dbo.ARTICULO`. The right pane shows the execution plan for a query. The query is as follows:

```

-- 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.NonTransportista,
        SUM(GO.CantidadEnviada) AS TotalCantidadEnviada
    FROM
        TRANSPORTISTA T
    JOIN
        GUIA_ENVIO GE ON T.CodTransportista = GE.CodTransportista
)

```

The results table shows the following data:

CodTransportista	NonTransportista	TotalCantidadEnviada	DenseRank	
84	2017	Transp. Rápido F-17	101	84
85	2016	Fleets Seguros F-16	98	85
86	2015	Fleets 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	Fleets 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 Search

CONNECTIONS: AZURE + ... sconnected SQLQuery_1.sql - (62) T.U (sa1) SQLQuery_1 - (53) T.U (sa1)

Run Cancel Disconnect Change Database: QhatuPERU Estimated Plan

Enable Actual Plan Parse

```

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

```

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

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

File Edit View Help Search

CONNECTIONS: AZURE + ... sconnected SQLQuery_1.sql - (62) T.U (sa1) SQLQuery_1 - (53) T.U (sa1)

Run Cancel Disconnect Change Database: QhatuPERU Estimated Plan

Enable Actual Plan Parse

```

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

```

Results Messages

	NumGuia	CodTienda	FechaSalida	ValorGuia	AcumuladoVentasTienda
84	4084	184	2025-09-18	18063.00	18063.00
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	25500.00	25500.00

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

