

**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**

## Manual de implementación de consultas

# SEGUNDA PARTE DE LAS CONSULTAS:

## ¿Qué es la Cláusula GROUP BY en SQL?

La cláusula **GROUP BY** se emplea en SQL para **organizar** filas que comparten los mismos valores en una o más columnas definidas.

Su función principal es permitir que las **Funciones de Agregación** (COUNT, SUM, AVG, etc.) trabajen sobre **grupos específicos de registros**, en vez de evaluar toda la tabla completa.

Una forma simple de entenderlo es imaginar que ordenas tu ropa: tomas todas tus prendas (todas las filas) y las divides por colores (la columna Color). Luego puedes contar cuántas prendas hay en cada grupo por separado.

## ¿Para qué Sirve y Cómo se Utiliza?

**GROUP BY** es fundamental cuando necesitas generar **resúmenes, cálculos, o indicadores por categoría**.

### 1. Sintaxis Básica

La cláusula **GROUP BY** siempre se coloca después de **WHERE** (si lo hay) y antes de **ORDER BY**.

```
SELECT columna_de_agrupacion,  
       FUNCION_DE_AGREGACION(otra_columna)  
FROM nombre_tabla  
WHERE [condiciones de filtrado (opcional)]  
GROUP BY columna_de_agrupacion;
```

### 2. La Regla de Oro 📜

Toda columna que aparezca en el **SELECT** y que **no** forme parte de una función de agregación **debe** estar incluida en la cláusula **GROUP BY**.

Si no, la consulta generará error.

### 3. Ejemplo Práctico

Supongamos que tienes una tabla de ventas llamada **Pedidos** que contiene las columnas **Region** y **MontoVenta**. Tu objetivo es conocer el **Monto Total vendido por cada región**.

Region	MontoVenta
Norte	100
Sur	200
Norte	150
Sur	300

Consulta SQL:  
SQL

```
SELECT Region, SUM(MontoVenta) AS TotalVentas
FROM Pedidos
GROUP BY Region;
```

Resultado:

Region	TotalVentas
Norte	250
Sur	500

**Explicación:** SQL agrupa todas las filas con Region = 'Norte' y les aplica la función SUM(). Luego hace lo mismo para las filas con Region = 'Sur', devolviendo un resumen por cada grupo.

### Complemento: La Cláusula HAVING

Una vez que has agrupado los datos con GROUP BY, a menudo querrás filtrar esos grupos basándote en el resultado de la agregación (el SUM, el AVG, etc.). Para esto se usa la cláusula HAVING.

- **WHERE:** Filtra **filas individuales** *antes* de la agrupación.
- **HAVING:** Filtra **grupos** *después* de la agrupación y agregación.

## Ejemplo con HAVING:

Muestra solo las regiones donde el total de ventas haya superado los \$400:

SQL

```
SELECT Region, SUM(MontoVenta) AS TotalVentas
FROM Pedidos
GROUP BY Region
HAVING SUM(MontoVenta) > 400; -- Filtra los grupos
```

**Resultado:**

Sur, 500

### CLÁUSULA GROUP BY

II. Explica de manera clara y didáctica qué son la **CLÁUSULA GROUP BY** en SQL y cómo se utilizan.

11. Mostrar NomLinea y CantArticulos.
12. Mostrar CodLinea y StockTotal.
13. Para cada NumOrden, calcular CostoTotal = SUM(PrecioCompra\*Cantidad).
14. Mostrar NumGuia y PromedioEnviado.
15. Contar proveedores agrupados por Ciudad.
16. Mostrar el número de órdenes por día (sin hora).
17. Sumar (CantidadEnviada\*PrecioVenta) por CodTienda.
18. Mostrar artículos cuyo StockActual < promedio de su CodLinea.
19. Mostrar CodProveedor, NomProveedor y CantArticulos.
20. Mostrar para cada Estado sumar CantidadSolicitada.

## Explicación: Cláusula GROUP BY en SQL

La cláusula GROUP BY es una herramienta fundamental en SQL que se utiliza en combinación con funciones de agregación (como SUM, AVG, COUNT, MAX, MIN).

**¿Qué es y cómo funciona?**

- **Propósito:** GROUP BY divide el conjunto de resultados de una consulta en grupos de filas, basándose en los valores de una o más columnas.
- **Función:** Una vez agrupadas las filas, las funciones de agregación aplicadas en la cláusula SELECT operan sobre cada grupo de forma independiente, en lugar de hacerlo sobre todo el conjunto de resultados.
- **Analogía Didáctica:** Imagina que tienes una hoja de cálculo con todas las ventas de la semana (Artículo, Cantidad, Vendedor). Si quieres saber cuántas unidades vendió **cada vendedor**, debes *agrupar* los datos por el nombre del Vendedor y luego *sumar* las cantidades dentro de cada grupo. El GROUP BY hace esta "agrupación por vendedor".

**Regla Crucial:** Cualquier columna que esté en la cláusula SELECT y **no** sea una función de agregación (como SUM(), COUNT(), etc.), **debe** aparecer obligatoriamente en la cláusula GROUP BY.

## 11. Mostrar NomLinea y CantArticulos.

**Enunciado:** Mostrar NomLinea y CantArticulos.

**Código SQL:**

```
-- 11. Mostrar NomLinea y CantArticulos.
SELECT L.NomLinea, COUNT(A.CodArticulo) AS CantArticulos
FROM LINEA L
JOIN ARTICULO A ON L.CodLinea = A.CodLinea
GROUP BY L.NomLinea
ORDER BY CantArticulos DESC;
GO
```

CONNECTIONS: AZURE + [66] t..U (sa1) consulta\_01.sql - (54) t..U (sa1) SQLQuery\_1 - (53) t..U (sa1)

ALBERT JEANKARLO CHUQUIYA...

Azure for Students

SQL databases

cafesito (jean3)

master (jean3)

master (trabajo)

QhatuPERU (jean3)

Tables

dbo.ARTICULO

Columns

Keys

Constraints

Triggers

Indexes

Statistics

dbo.GUIA\_DETALLE

dbo.GUIA\_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

Enable Actual Plan Parse

```

1 -- 11. Mostrar NomLinea y CantArticulos.
2 SELECT L.NomLinea, COUNT(A.CodArticulo) AS CantArticulos
3 FROM LINEA L
4 JOIN ARTICULO A ON L.CodLinea = A.CodLinea
5 GROUP BY L.NomLinea
6 ORDER BY CantArticulos DESC;
7 GO
8
9 ---
10
11 -- 12. Mostrar CodLinea y StockTotal.
12 SELECT CodLinea, SUM(StockActual) AS StockTotal

```

Results Messages

	NomLinea	CantArticulos
1	LN-ACAB13	1
2	LN-ACAB18	1
3	LN-ACAB23	1
4	LN-ACAB28	1
5	LN-ACAB3	1
6	LN-ACAB33	1
7	LN-ACAB38	1
8	LN-ACAB43	1
9	LN-ACAB48	1
10	LN-ACAB53	1
11	LN-ACAB58	1
12	LN-ACAB63	1
13	LN-ACAB68	1
14	LN-ACAB73	1
15	LN-ACAB78	1
16	LN-ACAB8	1
17	LN-ACAB83	1

**Explicación:** Une las tablas LINEA y ARTICULO. Agrupa los resultados por el nombre de la línea (NomLinea) y cuenta cuántos artículos diferentes pertenecen a cada línea de producto (COUNT(A.CodArticulo)).

## 12. Mostrar CodLinea y StockTotal.

**Enunciado:** Mostrar CodLinea y StockTotal.

**Código SQL:**

```

-- 12. Mostrar CodLinea y StockTotal.
SELECT CodLinea, SUM(StockActual) AS StockTotal
FROM ARTICULO
GROUP BY CodLinea
ORDER BY StockTotal DESC;
GO

```

File Edit View Help

CONNECTIONS: AZURE + ... [66] t..U (sa1) consulta\_01.sql - (54) t..U (sa1) SQLQuery\_1 - (53) t..U (sa1)

Run Cancel Disconnect Change Database: QhutuPERU Estimated Plan

Enable Actual Plan Parse

```

7 60
8
9 ---
10
11 -- 12. Mostrar CodLinea y StockTotal.
12 SELECT CodLinea, SUM(StockActual) AS StockTotal
13 FROM ARTICULO
14 GROUP BY CodLinea
15 ORDER BY StockTotal DESC;
16 60
17
18 ---
19

```

Results Messages

	CodLinea	StockTotal
1	100	250
2	99	248
3	98	246
4	97	244
5	96	242
6	95	240
7	94	238
8	93	236
9	92	234
10	91	232
11	90	230
12	89	228
13	88	226
14	87	224
15	86	222
16	85	220

Left sidebar: SQL databases, cafesito (jean3), master (jean3), master (trabaj), QhutuPERU (jean3), Tables, dbo.ARTICULO, Columns, Keys, Constraints, Triggers, Indexes, Statistics, dbo.GUIA\_DETALLE, dbo.GUIA\_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.

File Edit View Help

CONNECTIONS: AZURE + ... [66] t..U (sa1) consulta\_01.sql - (54) t..U (sa1) SQLQuery\_1 - (53) t..U (sa1)

Run Cancel Disconnect Change Database: QhutuPERU Estimated Plan

Enable Actual Plan Parse

```

7 60
8
9 ---
10
11 -- 12. Mostrar CodLinea y StockTotal.
12 SELECT CodLinea, SUM(StockActual) AS StockTotal
13 FROM ARTICULO
14 GROUP BY CodLinea
15 ORDER BY StockTotal DESC;
16 60
17
18 ---
19

```

Results Messages

	CodLinea	StockTotal
84	17	84
85	16	82
86	15	80
87	14	78
88	13	76
89	12	74
90	11	72
91	10	70
92	9	68
93	8	66
94	7	64
95	6	62
96	5	60
97	4	58
98	3	56
99	2	54
100	1	52

Left sidebar: SQL databases, cafesito (jean3), master (jean3), master (trabaj), QhutuPERU (jean3), Tables, dbo.ARTICULO, Columns, Keys, Constraints, Triggers, Indexes, Statistics, dbo.GUIA\_DETALLE, dbo.GUIA\_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.

**Explicación:** Agrupa todos los registros de la tabla ARTICULO por el CodLinea y utiliza la función SUM(StockActual) para calcular el inventario total (suma de unidades) para cada línea de productos.

**13. Para cada NumOrden, calcular CostoTotal = SUM([PrecioCompra \* Cantidad]).**

**Enunciado:** Para cada NumOrden, calcular CostoTotal = SUM([PrecioCompra \* Cantidad]).

**Código SQL:**

```
-- 13. Para cada NumOrden, calcular CostoTotal =  
SUM([PrecioCompra * Cantidad]).  
SELECT NumOrden, CAST(SUM(PrecioCompra *  
CantidadSolicitada) AS MONEY) AS CostoTotalOrden  
FROM ORDEN_DETALLE  
GROUP BY NumOrden  
ORDER BY CostoTotalOrden DESC;  
GO
```



File Edit View Help Search

CONNECTIONS: AZURE + [66] t...U (sa1) consulta\_01.sql - (54) t...U (sa1) SQLQuery\_1 - (53) t...U (sa1)

ALBERT JEANKARLO CHUQUIVA...  
Azure for Students  
SQL databases  
cafeito (jean3)  
master (jean3)  
master (trabaj)  
QhatuPERU (jean3)  
Tables  
dbo.ARTICULO  
Columns  
Keys  
Constraints  
Triggers  
Indexes  
Statistics  
dbo.GUIA\_DETALLE  
dbo.GUIA\_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  
Enable Actual Plan Parse

```
17  
18 ---  
19  
20 -- 13. Para cada NumOrden, calcular CostoTotal = SUM([PrecioCompra * Cantidad]).  
21 SELECT NumOrden, CAST(SUM(PrecioCompra * CantidadSolicitada) AS MONEY) AS CostoTotal0  
22 FROM ORDEN_DETALLE  
23 GROUP BY NumOrden  
24 ORDER BY CostoTotalOrden DESC;  
25 GO  
26  
27 ---  
28
```

Results Messages

	NumOrden	CostoTotalOrden
1	3100	31500.00
2	3099	30940.00
3	3098	30385.00
4	3097	29835.00
5	3096	29290.00
6	3095	28750.00
7	3094	28215.00
8	3093	27685.00
9	3092	27160.00
10	3091	26640.00
11	3090	26125.00
12	3089	25615.00
13	3088	25110.00
14	3087	24610.00
15	3086	24115.00
16	3085	23625.00
17	3084	23140.00

The screenshot shows the SQL Server Enterprise Manager interface. The left pane displays the database structure, including the 'QhutuPERU' database and its tables. The right pane shows the execution plan and the results of a query. The query is as follows:

```

-- 13. Para cada NumOrden, calcular CostoTotal = SUM([PrecioCompra * Cantidad]).
SELECT NumOrden, CAST(SUM(PrecioCompra * CantidadSolicitada) AS MONEY) AS CostoTotalOrden
FROM ORDEN_DETALLE
GROUP BY NumOrden
ORDER BY CostoTotalOrden DESC;

```

The results table shows the following data:

NumOrden	CostoTotalOrden
84	3017
85	3016
86	3015
87	3014
88	3013
89	3012
90	3011
91	3010
92	3009
93	3008
94	3007
95	3006
96	3005
97	3004
98	3003
99	3002
100	3001

**Explicación:** Agrupa los detalles de las compras por el NumOrden. Para cada orden, calcula el costo de cada detalle (PrecioCompra \* CantidadSolicitada) y luego suma todos esos costos (SUM()) para obtener el costo total de la orden completa.

#### 14. Mostrar NumGuia y PromedioEnviado.

**Enunciado:** Mostrar NumGuia y PromedioEnviado.

**Código SQL:**

```

-- 14. Mostrar NumGuia y PromedioEnviado.
SELECT NumGuia, CAST(AVG(CantidadEnviada) AS
DECIMAL(10,2)) AS PromedioArticulosEnviados
FROM GUIA_DETALLE
GROUP BY NumGuia;
GO

```

File Edit View Help Search

CONNECTIONS: AZURE + ... [66] t..U (sa1) consulta\_01.sql - (54) t..U (sa1) SQLQuery\_1 - (53) t..U (sa1)

Run Cancel Disconnect Change Database: QhatuPERU Estimated Plan

Enable Actual Plan Parse

```

26
27 ---
28
29 -- 14. Mostrar NumGuia y PromedioEnviado.
30 SELECT NumGuia, CAST(AVG(CantidadEnviada) AS DECIMAL(10,2)) AS PromedioArticulosEnviado
31 FROM GUIA_DETALLE
32 GROUP BY NumGuia;
33 GO
34
35 ---
36
37 -- 15. Contar proveedores agrupados por Ciudad.

```

Results Messages

	NumGuia	PromedioArticulosEnviados
1	4001	53.00
2	4002	56.00
3	4003	59.00
4	4004	62.00
5	4005	65.00
6	4006	68.00
7	4007	71.00
8	4008	74.00
9	4009	77.00
10	4010	80.00
11	4011	83.00
12	4012	86.00
13	4013	89.00
14	4014	92.00
15	4015	95.00
16	4016	98.00
17	4017	101.00

Left sidebar: SERVERS, DATABASES, TABLES, COLUMNS, KEYS, CONSTRAINTS, TRIGGERS, INDEXES, STATISTICS, VIEWS, SYNONYMS, PROGRAMMABILITY, EXTERNAL RESOURCES, STORAGE, SECURITY, QhatuPERU (jean3), SQL servers.

File Edit View Help Search

CONNECTIONS: AZURE + ... [66] t..U (sa1) consulta\_01.sql - (54) t..U (sa1) SQLQuery\_1 - (53) t..U (sa1)

Run Cancel Disconnect Change Database: QhatuPERU Estimated Plan

Enable Actual Plan Parse

```

26
27 ---
28
29 -- 14. Mostrar NumGuia y PromedioEnviado.
30 SELECT NumGuia, CAST(AVG(CantidadEnviada) AS DECIMAL(10,2)) AS PromedioArticulosEnviado
31 FROM GUIA_DETALLE
32 GROUP BY NumGuia;
33 GO
34
35 ---
36
37 -- 15. Contar proveedores agrupados por Ciudad.

```

Results Messages

	NumGuia	PromedioArticulosEnviados
84	4084	302.00
85	4085	305.00
86	4086	308.00
87	4087	311.00
88	4088	314.00
89	4089	317.00
90	4090	320.00
91	4091	323.00
92	4092	326.00
93	4093	329.00
94	4094	332.00
95	4095	335.00
96	4096	338.00
97	4097	341.00
98	4098	344.00
99	4099	347.00
100	4100	350.00

Left sidebar: SERVERS, DATABASES, TABLES, COLUMNS, KEYS, CONSTRAINTS, TRIGGERS, INDEXES, STATISTICS, VIEWS, SYNONYMS, PROGRAMMABILITY, EXTERNAL RESOURCES, STORAGE, SECURITY, QhatuPERU (jean3), SQL servers.

Status bar: Ln 29, Col 1 (175 selected) Spaces: 4 UTF-8 CRLF 100 rows MSSQL 00:00:00 tcp:jean3.database.windows.net: QhatuPERU (53)

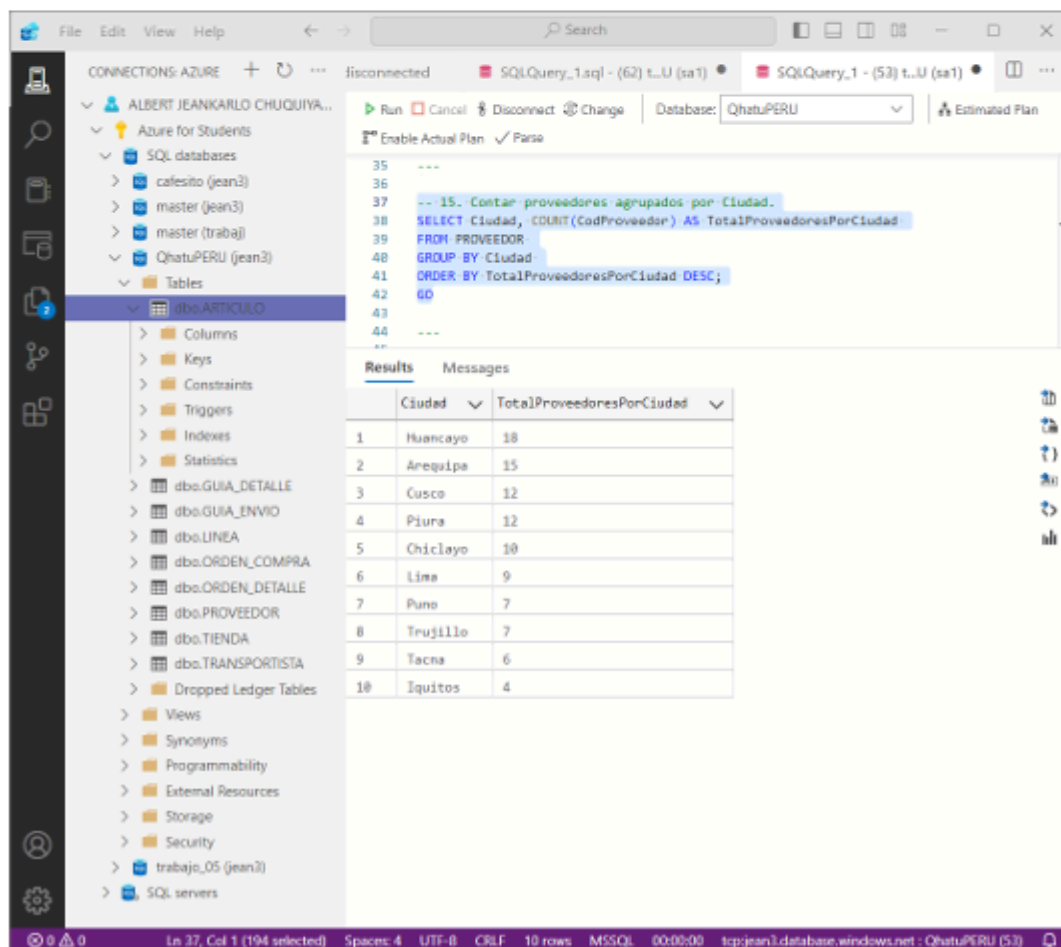
**Explicación:** Agrupa los registros de envío por el NumGuia y utiliza la función AVG(CantidadEnviada) para obtener la cantidad promedio de artículos que se enviaron por cada línea de detalle dentro de esa guía.

## 15. Contar proveedores agrupados por Ciudad.

**Enunciado:** Contar proveedores agrupados por Ciudad.

**Código SQL:**

```
-- 15. Contar proveedores agrupados por Ciudad.  
SELECT Ciudad, COUNT(CodProveedor) AS  
TotalProveedoresPorCiudad  
FROM PROVEEDOR  
GROUP BY Ciudad  
ORDER BY TotalProveedoresPorCiudad DESC;  
GO
```



The screenshot displays the SQL Server Enterprise Manager interface. The left pane shows the database structure, with the 'dbo' schema selected. The right pane shows the execution of a SQL query. The query is as follows:

```
-- 15. Contar proveedores agrupados por Ciudad.  
SELECT Ciudad, COUNT(CodProveedor) AS TotalProveedoresPorCiudad  
FROM PROVEEDOR  
GROUP BY Ciudad  
ORDER BY TotalProveedoresPorCiudad DESC;  
GO
```

The results are shown in a table with two columns: 'Ciudad' and 'TotalProveedoresPorCiudad'. The data is as follows:

Ciudad	TotalProveedoresPorCiudad
Huancayo	18
Arequipa	15
Cusco	12
Piura	12
Chiclayo	10
Lima	9
Puno	7
Trujillo	7
Tacna	6
Iquitos	4

**Explicación:** Agrupa la tabla PROVEEDOR basándose en la columna Ciudad y cuenta cuántos proveedores únicos (COUNT(CodProveedor)) tienen su ubicación registrada en cada una de esas ciudades.

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

**Enunciado:** Mostrar el número de órdenes por día (sin hora).

### Código SQL:

```
-- 16. Mostrar el número de órdenes por día (sin hora).  
SELECT CAST(FechaOrden AS DATE) AS DiaOrden,  
COUNT(NumOrden) AS TotalOrdenesDelDia  
FROM ORDEN_COMPRA  
GROUP BY CAST(FechaOrden AS DATE)  
ORDER BY DiaOrden DESC;  
GO
```

The screenshot shows the SQL Server Enterprise Manager interface. The left pane displays the database structure for 'QhutuPERU', with the 'dbo' schema expanded. The center pane shows the SQL query from the previous block. The right pane displays the results of the query, which is a table with two columns: 'DiaOrden' and 'TotalOrdenesDelDia'. The results show 17 rows of data, representing the number of orders per day from 2025-10-13 to 2025-10-29.

	DiaOrden	TotalOrdenesDelDia
1	2025-10-29	1
2	2025-10-28	1
3	2025-10-27	1
4	2025-10-26	1
5	2025-10-25	1
6	2025-10-24	1
7	2025-10-23	1
8	2025-10-22	1
9	2025-10-21	1
10	2025-10-20	1
11	2025-10-19	1
12	2025-10-18	1
13	2025-10-17	1
14	2025-10-16	1
15	2025-10-15	1
16	2025-10-14	1
17	2025-10-13	1

The screenshot shows the SQL Server Enterprise Manager interface. On the left, the 'Tables' folder under 'QhutuPERU (jean3)' is expanded, showing various tables like 'dbo.GUIA\_DETALLE', 'dbo.GUIA\_ENVIO', 'dbo.LINEA', 'dbo.ORDEN\_COMPRA', etc. The central pane displays the execution of a SQL query. The query text is:   
 -- 16. Mostrar el número de órdenes por día (sin hora).  
 SELECT CAST(FechaOrden AS DATE) AS DiaOrden, COUNT(NumOrden) AS TotalOrdenesDeDia  
 FROM ORDEN\_COMPRA  
 GROUP BY CAST(FechaOrden AS DATE)  
 ORDER BY DiaOrden DESC;  
 The results pane shows a table with two columns: 'DiaOrden' and 'TotalOrdenesDeDia'. The data rows show dates from 2025-08-07 down to 2025-07-22, each with a count of 1.

**Explicación:** Utiliza CAST(FechaOrden AS DATE) para ignorar el componente de tiempo y agrupar solo por la fecha. COUNT(NumOrden) suma cuántas órdenes se registraron en cada día único.

## 17. Sumar (CantidadEnviada \* PrecioVenta) por CodTienda.

**Enunciado:** Sumar (CantidadEnviada \* PrecioVenta) por CodTienda.

### Código SQL:

```
-- 17. Sumar (CantidadEnviada * PrecioVenta) por CodTienda.
SELECT GE.CodTienda, CAST(SUM(GD.CantidadEnviada *
GD.PrecioVenta) AS MONEY) AS ValorTotalVentas
FROM GUIA_ENVIO GE
JOIN GUIA_DETALLE GD ON GE.NumGuia = GD.NumGuia
GROUP BY GE.CodTienda
ORDER BY ValorTotalVentas DESC;
GO
```



File Edit View Help [66] t..U (sa1) consulta\_01.sql - (54) t..U (sa1) SQLQuery\_1 - (53) t..U (sa1)

CONNECTIONS: AZURE + [66] t..U (sa1) consulta\_01.sql - (54) t..U (sa1) SQLQuery\_1 - (53) t..U (sa1)

ALBERT JEANKARLO CHUQUIVA... Azure for Students

SQL databases

- cafeito (jean3)
- master (jean3)
- master (trabaj)
- QhatuPERU (jean3)
  - Tables
    - dbo.ARTICULO
    - Columns
    - Keys
    - Constraints
    - Triggers
    - Indexes
    - Statistics
    - dbo.GUIA\_DETALLE
    - dbo.GUIA\_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

Enable Actual Plan Parse

```

54
55 -- 17. Sumar (CantidadEnviada * PrecioVenta) por CodTienda.
56 SELECT GE.CodTienda, CAST(SUM(GD.CantidadEnviada * GD.PrecioVenta) AS MONEY) AS ValorT
57 FROM GUIA_ENVIO GE
58 JOIN GUIA_DETALLE GD ON GE.NumGuia = GD.NumGuia
59 GROUP BY GE.CodTienda
60 ORDER BY ValorTotalVentas DESC;
61
62
63 ---
64
65 -- 18. Mostrar articulos cuyo StockActual < promedio de su CodLinea.

```

Results Messages

	CodTienda	ValorTotalVentas
1	200	25900.00
2	199	25888.10
3	198	24630.40
4	197	24176.90
5	196	23727.60
6	195	23282.50
7	194	22841.60
8	193	22484.90
9	192	21972.40
10	191	21544.10
11	190	21120.00
12	189	20700.10
13	188	20284.40
14	187	19872.90
15	186	19465.60
16	185	19062.50
17	184	18663.60

Ln 55, Col 1 (292 selected) Spaces: 4 UTF-8 CRLF 100 rows MSSQL 00:00:00 tcp:jean3.database.windows.net : QhatuPERU (53)

File Edit View Help [66] t..U (sa1) consulta\_01.sql - (54) t..U (sa1) SQLQuery\_1 - (53) t..U (sa1)

CONNECTIONS: AZURE + [66] t..U (sa1) consulta\_01.sql - (54) t..U (sa1) SQLQuery\_1 - (53) t..U (sa1)

ALBERT JEANKARLO CHUQUIVA... Azure for Students

SQL databases

- cafeito (jean3)
- master (jean3)
- master (trabaj)
- QhatuPERU (jean3)
  - Tables
    - dbo.ARTICULO
    - Columns
    - Keys
    - Constraints
    - Triggers
    - Indexes
    - Statistics
    - dbo.GUIA\_DETALLE
    - dbo.GUIA\_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

Enable Actual Plan Parse

```

54
55 -- 17. Sumar (CantidadEnviada * PrecioVenta) por CodTienda.
56 SELECT GE.CodTienda, CAST(SUM(GD.CantidadEnviada * GD.PrecioVenta) AS MONEY) AS ValorT
57 FROM GUIA_ENVIO GE
58 JOIN GUIA_DETALLE GD ON GE.NumGuia = GD.NumGuia
59 GROUP BY GE.CodTienda
60 ORDER BY ValorTotalVentas DESC;
61
62
63 ---
64
65 -- 18. Mostrar articulos cuyo StockActual < promedio de su CodLinea.

```

Results Messages

	CodTienda	ValorTotalVentas
84	117	1504.90
85	116	1391.60
86	115	1282.50
87	114	1177.60
88	113	1076.90
89	112	980.40
90	111	888.10
91	110	880.00
92	109	716.10
93	108	616.40
94	107	560.90
95	106	489.60
96	105	422.50
97	104	359.60
98	103	300.90
99	102	246.40
100	101	196.10

Ln 55, Col 1 (292 selected) Spaces: 4 UTF-8 CRLF 100 rows MSSQL 00:00:00 tcp:jean3.database.windows.net : QhatuPERU (53)

**Explicación:** Une las guías de envío (GUIA\_ENVIO) con los detalles (GUIA\_DETALLE). Agrupa por CodTienda y suma el valor de venta de todos los productos despachados a esa tienda (CantidadEnviada \* PrecioVenta).

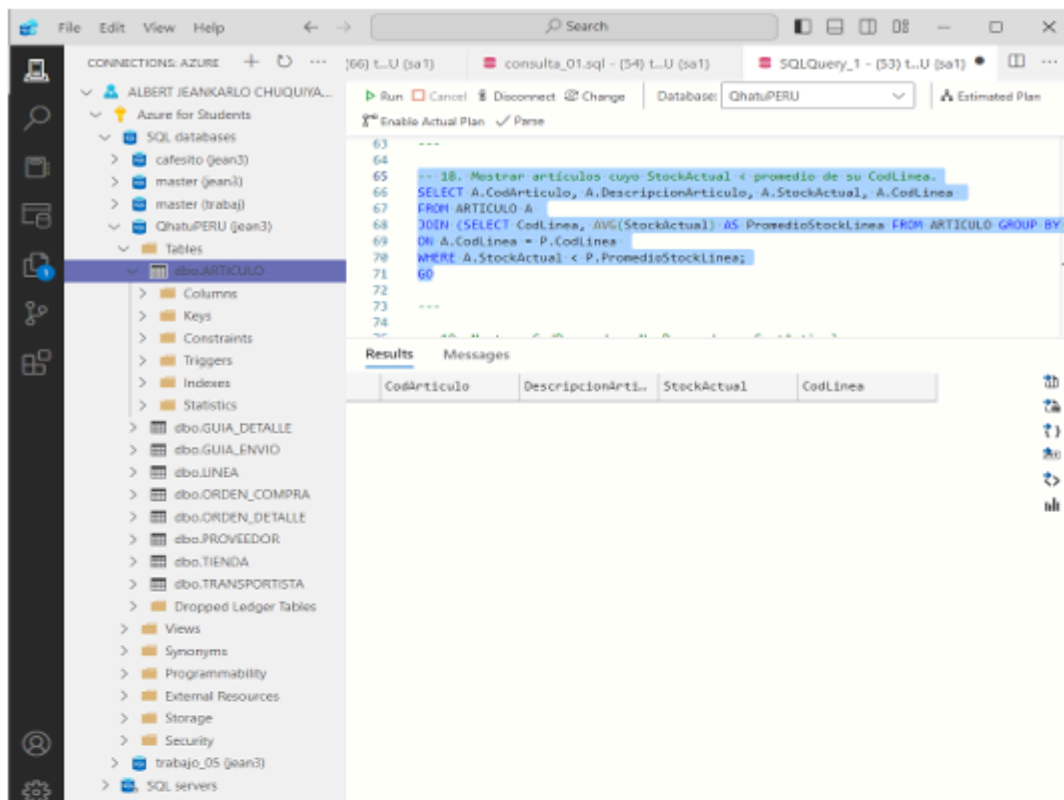
#### **18. Mostrar artículos cuyo StockActual < promedio de su CodLinea.**

**Enunciado:** Mostrar artículos cuyo StockActual < promedio de su CodLinea.

#### **Código SQL:**

```
-- 18. Mostrar artículos cuyo StockActual < promedio de su  
CodLinea.  
SELECT A.CodArticulo, A.DescripcionArticulo, A.StockActual,  
A.CodLinea  
FROM ARTICULO A  
JOIN (SELECT CodLinea, AVG(StockActual) AS  
PromedioStockLinea FROM ARTICULO GROUP BY CodLinea) AS  
P  
ON A.CodLinea = P.CodLinea  
WHERE A.StockActual < P.PromedioStockLinea;  
GO
```





**Explicación:** Esta consulta utiliza una **subconsulta** (P) que calcula el stock promedio para cada línea. Luego, el resultado se une con la tabla principal (ARTICULO A) para filtrar (WHERE) y mostrar solo aquellos artículos cuyo StockActual es menor que el PromedioStockLinea calculado para su categoría.

## 19. Mostrar CodProveedor, NomProveedor y CantArticulos.

**Enunciado:** Mostrar CodProveedor, NomProveedor y CantArticulos.

**Código SQL:**

```

-- 19. Mostrar CodProveedor, NomProveedor y CantArticulos.
SELECT P.CodProveedor, P.NomProveedor,
COUNT(A.CodArtículo) AS CantArticulosSuministrados
FROM PROVEEDOR P
JOIN ARTICULO A ON P.CodProveedor = A.CodProveedor
GROUP BY P.CodProveedor, P.NomProveedor
ORDER BY CantArticulosSuministrados DESC;
GO

```

File Edit View Help [66] T.U (sa1) consulta\_01.sql - (54) T.U (sa1) SQLQuery\_1 - (53) T.U (sa1)

Run Cancel Disconnect Change Database: QhatuPERU Estimated Plan

Enable Actual Plan Parse

```

74 ---
75 -- 19. Mostrar CodProveedor, NonProveedor y CantArticulos.
76 SELECT P.CodProveedor, P.NonProveedor, COUNT(A.CodArticulo) AS CantArticulosSuministrados
77 FROM PROVEEDOR P
78 JOIN ARTICULO A ON P.CodProveedor = A.CodProveedor
79 GROUP BY P.CodProveedor, P.NonProveedor
80 ORDER BY CantArticulosSuministrados DESC;
81
82
83 ---
84

```

Results Messages

	CodProveedor	NonProveedor	CantArticulosSuministrados
1	1	Fabricaciones Lima S-A-1	1
2	2	G&G Repres. S-A-2	1
3	3	Metalúrgica Perú S-A-3	1
4	4	G&G Repres. S-A-4	1
5	5	Logística del Norte S-A-5	1
6	6	Logística del Norte S-A-6	1
7	7	Insumos Global S-A-7	1
8	8	Distribuidora Alfa S-A-8	1
9	9	Químicos del Pacífico S-A-9	1
10	10	Suministros AQP S-A-10	1
11	11	G&G Repres. S-A-11	1
12	12	Suministros AQP S-A-12	1
13	13	TecnoIndustrial S-A-13	1
14	14	Insumos Global S-A-14	1
15	15	Fabricaciones Lima S-A-15	1
16	16	Insumos Global S-A-16	1

File Edit View Help [66] T.U (sa1) consulta\_01.sql - (54) T.U (sa1) SQLQuery\_1 - (53) T.U (sa1)

Run Cancel Disconnect Change Database: QhatuPERU Estimated Plan

Enable Actual Plan Parse

```

74 ---
75 -- 19. Mostrar CodProveedor, NonProveedor y CantArticulos.
76 SELECT P.CodProveedor, P.NonProveedor, COUNT(A.CodArticulo) AS CantArticulosSuministrados
77 FROM PROVEEDOR P
78 JOIN ARTICULO A ON P.CodProveedor = A.CodProveedor
79 GROUP BY P.CodProveedor, P.NonProveedor
80 ORDER BY CantArticulosSuministrados DESC;
81
82
83 ---
84

```

Results Messages

	CodProveedor	NonProveedor	CantArticulosSuministrados
84	84	Insumos Global S-A-84	1
85	85	Químicos del Pacífico S-A-...	1
86	86	Químicos del Pacífico S-A-...	1
87	87	Distribuidora Alfa S-A-87	1
88	88	TecnoIndustrial S-A-88	1
89	89	Químicos del Pacífico S-A-...	1
90	90	Logística del Norte S-A-90	1
91	91	Metalúrgica Perú S-A-91	1
92	92	Químicos del Pacífico S-A-...	1
93	93	Químicos del Pacífico S-A-...	1
94	94	Distribuidora Alfa S-A-94	1
95	95	Logística del Norte S-A-95	1
96	96	Metalúrgica Perú S-A-96	1
97	97	Suministros AQP S-A-97	1
98	98	G&G Repres. S-A-98	1
99	99	Fabricaciones Lima S-A-99	1
100	100	Químicos del Pacífico S-A-...	1

Ln 75, Col 1 (311 selected) Spaces: 4 UTF-8 CRLF 100 rows MSSQL 00:00:00 tcp:jean3.database.windows.net : QhatuPERU (53)

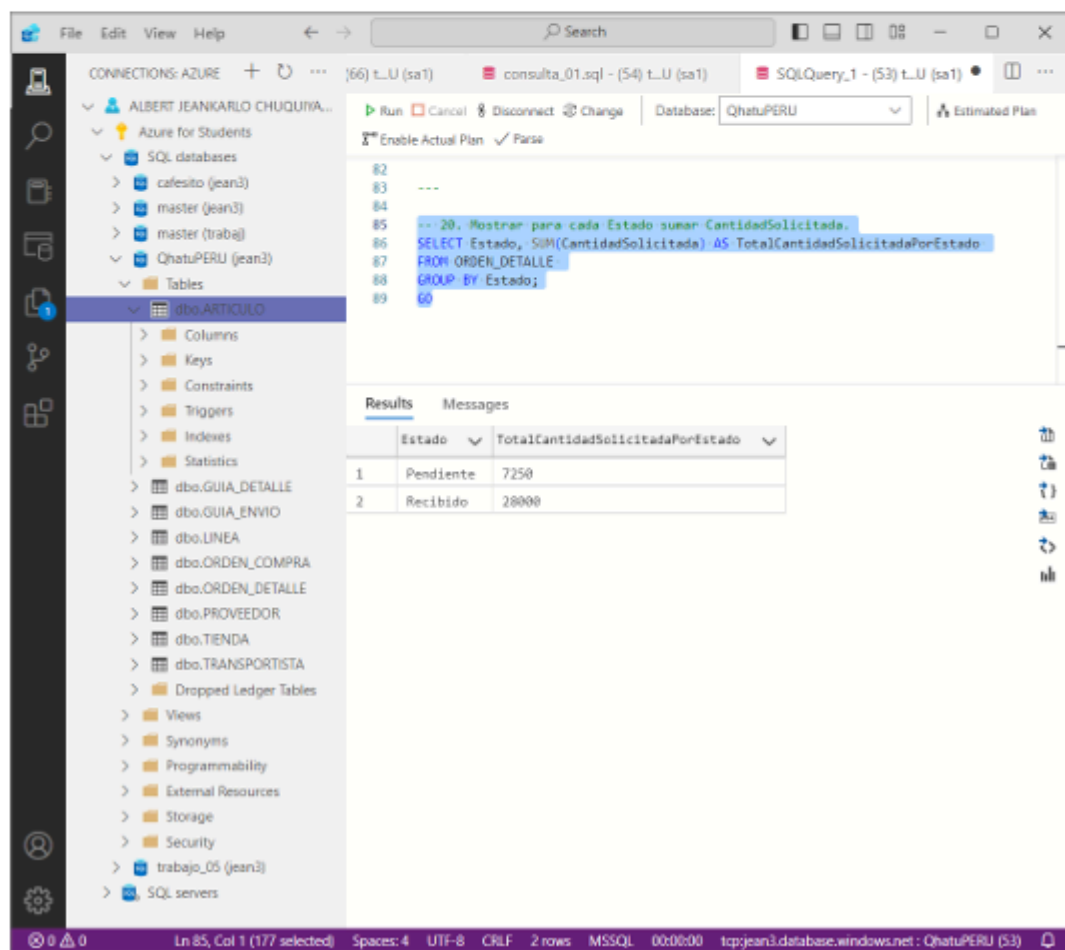
**Explicación:** Une las tablas PROVEEDOR y ARTICULO. Agrupa por el código y nombre del proveedor y utiliza COUNT(A.CodArticulo) para determinar cuántos productos diferentes suministra cada proveedor.

## 20. Mostrar para cada Estado sumar CantidadSolicitada.

**Enunciado:** Mostrar para cada Estado sumar CantidadSolicitada.

### Código SQL:

```
-- 20. Mostrar para cada Estado sumar CantidadSolicitada.  
SELECT Estado, SUM(CantidadSolicitada) AS  
TotalCantidadSolicitadaPorEstado  
FROM ORDEN_DETALLE  
GROUP BY Estado;  
GO
```



The screenshot shows the SQL Server Enterprise Manager interface. The left pane displays the database structure, including the 'ORDEN\_DETALLE' table. The right pane shows the execution of the SQL query. The query is: `SELECT Estado, SUM(CantidadSolicitada) AS TotalCantidadSolicitadaPorEstado FROM ORDEN_DETALLE GROUP BY Estado;` The results are displayed in a table with two columns: 'Estado' and 'TotalCantidadSolicitadaPorEstado'.

Estado	TotalCantidadSolicitadaPorEstado
Pendiente	7250
Recibido	28000

**Explicación:** Agrupa los registros de órdenes por el campo Estado (e.g., 'Recibido', 'Pendiente') y utiliza SUM(CantidadSolicitada) para totalizar la cantidad de productos que se encuentran en cada estado de la orden.