



UPLA
UNIVERSIDAD PERUANA LOS ANDES



“Año De La Recuperación Y
Consolidación De La Economía Peruana”

UNIVERSIDAD PERUANA LOS ANDES

“FACULTAD DE INGENIERÍA”

ESCUELA PROFESIONAL “SISTEMAS Y
COMPUTACIÓN”

CÁTEDRA: Base de Datos II

CATEDRÁTICO: Ing. Fernandez Bejarano Raul Enrique

ESTUDIANTE: Miranda Lévano Sebastián Gabriel

CICLO: V

SECCIÓN: B1

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 utiliza en SQL para **agrupar filas que tienen los mismos valores en una o más columnas especificadas**. Su propósito es permitir que las **Funciones de Agregación** (como COUNT, SUM, AVG, etc.) operen sobre **subconjuntos de filas** en lugar de sobre toda la tabla.

Piensa en ella como clasificar tu ropa: Tomas toda tu ropa (todas las filas) y la agrupas por color (la columna Color). Luego puedes contar cuántas prendas tiene cada montón por separado.

¿Para qué Sirve y Cómo se Utiliza?

El GROUP BY es indispensable cuando necesitas obtener resúmenes, estadísticas o métricas **por categoría**.

1. Sintaxis Básica

La cláusula GROUP BY siempre va después de la cláusula WHERE (si existe) y antes de ORDER BY.

SQL

```
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 📜

Cualquier columna que aparezca en la cláusula SELECT y que NO esté dentro de una Función de Agregación, DEBE estar listada en la cláusula GROUP BY.

3. Ejemplo Práctico

Imagina que tienes una tabla de pedidos (Pedidos) con las columnas Region y MontoVenta. Quieres saber el **Monto Total de Ventas 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 NumLinea y CantArticulos.
12. Mostrar CodLinea y StockTotal.
13. Para cada NumOrden, calcular CostoTotal = SUM(PrecioCompra×Cantidad).
14. Mostrar NumGuia y PromedioEnvío.
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
```

```

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
13  FROM ARTICULO

```

NomLinea	CantArticulos
LN-ACAB13	1
LN-ACAB18	1
LN-ACAB23	1
LN-ACAB28	1
LN-ACAB3	1
LN-ACAB33	1
LN-ACAB38	1
LN-ACAB43	1
LN-ACAB48	1
LN-ACAB53	1
LN-ACAB58	1
LN-ACAB63	1
LN-ACAB68	1
LN-ACAB73	1
LN-ACAB78	1
LN-ACAB8	1
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 ← → Search

CONNECTIONS: AZURE + ⌂ ... [66] t..U (sa1) consulta_01.sql - (54) t..U (sa1) SQLQuery_1 - (53) t..U (sa1) ...

ALBERT JEANKARLO CHUQUIAYA... Azure for Students SQL databases cafelito (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

```

7 GO
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 GO
17
18 ===
19

```

Results Messages

	CodLinea	StockTotal
1	188	258
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

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 CHUQUIAYA... Azure for Students SQL databases cafelito (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

```

7 GO
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 GO
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	3	52

In 11. Col 1 (154 selected) Spaces: 4 UTF-8 CR/LF 100 rows MSSQL 00:00:00 https://qhatuperu.database.windows.net:1433 QhatuPERU (53)

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) Database: QhatuPERU Estimated Plan

Run Cancel Disconnect Change Enable Actual Plan Parse

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

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 SSMS interface with the following details:

- Connections:** AZURE
- Database:** QhatuPERU
- Query:**

```

-- 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

```
- Results Grid:**

	NumOrden	CostoTotalOrden
84	3017	2035.00
85	3016	1890.00
86	3015	1750.00
87	3014	1615.00
88	3013	1485.00
89	3012	1360.00
90	3011	1240.00
91	3010	1125.00
92	3009	1015.00
93	3008	910.00
94	3007	810.00
95	3006	715.00
96	3005	625.00
97	3004	540.00
98	3003	460.00
99	3002	385.00
100	3001	315.00

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) 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 PromedioArticulosEnvia
31 FROM GUIA_DETALLE
32 GROUP BY NumGuia;
33
34
35
36
37 -- 15. Contar proveedores agrupados por Ciudad.

```

Results Messages

NumGuia	PromedioArticulosEnvia
1	4001
2	4002
3	4003
4	4004
5	4005
6	4006
7	4007
8	4008
9	4009
10	4010
11	4011
12	4012
13	4013
14	4014
15	4015
16	4016
17	4017

File Edit View Help ← → Search

CONNECTIONS: AZURE + ⏪ ⏴ ... (66) t...U (sa1) consulta_01.sql - (54) t...U (sa1) SQLQuery_1 - (53) t...U (sa1) 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 PromedioArticulosEnvia
31 FROM GUIA_DETALLE
32 GROUP BY NumGuia;
33
34
35
36
37 -- 15. Contar proveedores agrupados por Ciudad.

```

Results Messages

NumGuia	PromedioArticulosEnvia
84	4084
85	4085
86	4086
87	4087
88	4088
89	4089
90	4090
91	4091
92	4092
93	4093
94	4094
95	4095
96	4096
97	4097
98	4098
99	4099
100	4100

Ln 29, Col 1 (175 selected) Spaces: 4 UTF-8 CRLF 100 rows MSSQL 00:00:00 tqjean3.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 shows the SSMS interface with the following details:

- File Bar:** File, Edit, View, Help, Search.
- Toolbars:** Standard toolbar with icons for New Query, Run, Stop, Disconnect, Change, Parse, and Estimated Plan.
- Connections:** AZURE, ALBERT JEANKARLO CHUQUIAYA..., Azure for Students, SQL databases, master (jean3), master (trabaj), QhatuPERU (jean3), Tables, dbo.ARTICULO.
- Query Editor:** Contains the SQL code for question 15. The code is highlighted in blue and green.

```
-- 15. Contar proveedores agrupados por Ciudad.
SELECT Ciudad, COUNT(CodProveedor) AS TotalProveedoresPorCiudad
FROM PROVEEDOR
GROUP BY Ciudad
ORDER BY TotalProveedoresPorCiudad DESC;
GO
```
- Results Grid:** Shows the results of the query. The columns are Ciudad and TotalProveedoresPorCiudad. The data is as follows:

	Ciudad	TotalProveedoresPorCiudad
1	Huancayo	18
2	Arequipa	15
3	Cusco	12
4	Piura	12
5	Chiclayo	10
6	Lima	9
7	Puno	7
8	Trujillo	7
9	Tacna	6
10	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 SSMS interface with the following details:

- File Explorer:** Shows the database structure under "AZURE". The "Tables" node is expanded, showing "dbo.ARTICULO" which is selected. Other tables like "dbo.GUIA_DETALLE", "dbo.PROVEEDOR", etc., are also listed.
- Object Explorer:** Shows the same database structure, with "dbo.ARTICULO" selected.
- SQL Editor:** Contains the SQL query for question 16. The results of the query are displayed below it.
- Results Grid:** Displays the output of the query:

	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

```

-- 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

```

DiaOrden	TotalOrdenesDelDia
2025-08-07	1
2025-08-06	1
2025-08-05	1
2025-08-04	1
2025-08-03	1
2025-08-02	1
2025-08-01	1
2025-07-31	1
2025-07-30	1
2025-07-29	1
2025-07-28	1
2025-07-27	1
2025-07-26	1
2025-07-25	1
2025-07-24	1
2025-07-23	1
2025-07-22	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 ← → Search

CONNECTIONS: AZURE + ⏪ ⏴ ... (66) t...U (sa1) consulta_01.sql - (54) t...U (sa1) SQLQuery_1 - (53) t...U (sa1) ● Database: QhatuPERU Estimated Plan

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 GO
62
63 ---
64
65 -- 18. Mostrar artículos cuyo StockActual < promedio de su CodLinea.
66 SELECT AL.CodLinea, AL.Articulo, AVG(AL.StockActual) AS PromedioStock

```

Results Messages

CodTienda	ValorTotalVentas
1	200
2	199
3	198
4	197
5	196
6	195
7	194
8	193
9	192
10	191
11	190
12	189
13	188
14	187
15	186
16	185
17	184

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 ← → Search

CONNECTIONS: AZURE + ⏪ ⏴ ... (66) t...U (sa1) consulta_01.sql - (54) t...U (sa1) SQLQuery_1 - (53) t...U (sa1) ● Database: QhatuPERU Estimated Plan

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 GO
62
63 ---
64
65 -- 18. Mostrar artículos cuyo StockActual < promedio de su CodLinea.
66 SELECT AL.CodLinea, AL.Articulo, AVG(AL.StockActual) AS PromedioStock

```

Results Messages

CodTienda	ValorTotalVentas
84	117
85	116
86	115
87	114
88	113
89	112
90	111
91	110
92	109
93	108
94	107
95	106
96	105
97	104
98	103
99	102
100	101

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
```

```

File Edit View Help ← → Search
CONNECTIONS: AZURE + ⌂ ... (66) t...U (sa1) consulta_01.sql - (54) t...U (sa1) SQLQuery_1 - (53) t...U (sa1) Database: QhatuPERU Estimated Plan
Run Cancel Disconnect Change Database: QhatuPERU
Enable Actual Plan Parse
63 ---
64
65 -- 18. Mostrar artículos cuyo StockActual < promedio de su CodLinea.
66 SELECT A.CodArticulo, A.DescripcionArticulo, A.StockActual, A.CodLinea
67 FROM ARTICULO A
68 JOIN (SELECT CodLinea, AVG(StockActual) AS PromedioStockLinea FROM ARTICULO GROUP BY CodLinea)
69 ON A.CodLinea = P.CodLinea
70 WHERE A.StockActual < P.PromedioStockLinea;
71 GO
72
73 ---
74

```

Results

CodArticulo	DescripcionArti...	StockActual	CodLinea

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.CodArticulo) 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 ← → Search

CONNECTIONS: AZURE + ⏪ ⏴ ...

ALBERT JEANKARLO CHUQUIYA...
Azure for Students
SQL databases
cafesito (jean3)
master (jean3)
master (trabaj)
QhatuPERU (jean3)
Tables
dbo.ARTICULOS
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

(66) t..U (sa1) consulta_01.sql - (54) t..U (sa1) SQLQuery_1 - (53) t..U (sa1) Database: QhatuPERU Estimated Plan

Run Cancel Disconnect Change Database: QhatuPERU Estimated Plan

19. Mostrar CodProveedor, NombreProveedor y CantArticulos.

```
76 SELECT P.CodProveedor, P.NombreProveedor, COUNT(A.CodArticulo) AS CantArticulosSuministrados
77 FROM PROVEEDOR P
78 JOIN ARTICULO A ON P.CodProveedor = A.CodProveedor
79 GROUP BY P.CodProveedor, P.NombreProveedor
80 ORDER BY CantArticulosSuministrados DESC;
```

Results Messages

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

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.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

(66) t..U (sa1) consulta_01.sql - (54) t..U (sa1) SQLQuery_1 - (53) t..U (sa1) Database: QhatuPERU Estimated Plan

Run Cancel Disconnect Change Database: QhatuPERU Estimated Plan

19. Mostrar CodProveedor, NombreProveedor y CantArticulos.

```
76 SELECT P.CodProveedor, P.NombreProveedor, COUNT(A.CodArticulo) AS CantArticulosSuministrados
77 FROM PROVEEDOR P
78 JOIN ARTICULO A ON P.CodProveedor = A.CodProveedor
79 GROUP BY P.CodProveedor, P.NombreProveedor
80 ORDER BY CantArticulosSuministrados DESC;
```

Results Messages

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

- Left pane (Object Explorer):** Shows the database structure for 'QhatuPERU'. It includes a connection to 'ALBERT JEANKARLO CHUQUIYA...', several Azure databases, and the local database 'QhatuPERU'. Under 'QhatuPERU', there are tables like 'dbo.ARTICULO', 'dbo.GUIA_DETALLE', etc.
- Center pane (Query Editor):** Displays the SQL query for question 20. The code is:

```
-- 20. Mostrar para cada Estado sumar CantidadSolicitada.  
SELECT Estado, SUM(CantidadSolicitada) AS TotalCantidadSolicitadaPorEstado  
FROM ORDEN_DETALLE  
GROUP BY Estado;  
GO
```
- Right pane (Results):** Shows the execution results in a table format. The table has two columns: 'Estado' and 'TotalCantidadSolicitadaPorEstado'. The data is:

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.