

Capítulo VI

Agrupamiento y Agregación de Datos

En este capítulo conoceremos cómo agrupar los datos utilizando la cláusula GROUP BY para obtener consolidados por los grupos y acumulados usando las funciones de agregación.

Uno de los requerimientos más comunes en el análisis de datos es la generación de reportes con datos consolidados: totales, subtotales, promedios, acumulados, etc. El uso de las funciones de agregación y la cláusula GROUP BY de la sentencia SELECT nos ayudan a cumplir con dicho requerimiento.

1. FUNCIONES DE AGREGACIÓN

Una función de agregación permite efectuar una operación aritmética que consolida los valores de una columna para toda la tabla o para los mismos valores agrupados según determinado criterio. La función devuelve un solo valor que es el consolidado para la tabla o para cada uno de los grupos.

El siguiente cuadro muestra las funciones de agregación más utilizadas.

Función	Descripción y Sintaxis
AVG()	Retorna el promedio de los valores de una columna ó expresión. AVG([DISTINCT] expresión_numérica)
COUNT()	Retorna la cuenta del número de valores distintos a NULL en una columna ó expresión. Devuelve un valor de tipo int . COUNT([DISTINCT] expresión) COUNT(*)
COUNT_BIG()	Funciona como COUNT(), con la diferencia de que devuelve un valor de tipo bigint . COUNT_BIG([DISTINCT] expresión) COUNT_BIG(*)
MAX()	Retorna el valor máximo de una columna ó expresión. MAX(expresión)
MIN()	Retorna el valor mínimo de una columna ó expresión.

	<i>MIN(expressión)</i>
STDEV()	Retorna la desviación estándar típica de los valores de una columna ó expresión. <i>STDEV ([DISTINCT] expresión_numérica)</i>
STDEVP()	Retorna la desviación estándar para la población de los valores de una columna ó expresión. <i>STDEVP ([DISTINCT] expresión_numérica)</i>
SUM()	Retorna la suma de los valores de una columna ó expresión. <i>SUM([DISTINCT] expresión_numérica)</i>
VAR()	Retorna la varianza de los valores de una columna ó expresión. <i>VAR ([DISTINCT] expresión_numérica)</i>
VARP()	Retorna la varianza para la población de los valores de una columna ó expresión. <i>VARP ([DISTINCT] expresión_numérica)</i>

Para todas las funciones, DISTINCT indica que debe eliminarse los valores duplicados de **expresión** ó **expresión_numérica** antes de evaluar la función.

Para consolidar los datos de una columna puede usar las funciones de agregación con la declaración SELECT; si desea consolidar los datos agrupándolos por determinado criterio añada la cláusula GROUP BY.

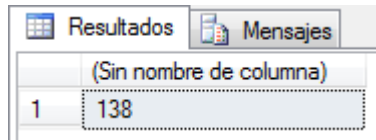
Con excepción de la función COUNT(*), todas las funciones de agregación retornan NULL si ninguna fila satisface la cláusula WHERE. La función COUNT(*) retorna un valor de cero si ninguna fila satisface la cláusula WHERE.

Ejercicio 6.1: Uso de la función COUNT()

1. Cuenta de los artículos registrados en la base de datos.

```
USE QhatuPERU
go
```

```
SELECT COUNT(*) FROM ARTICULO
go
```

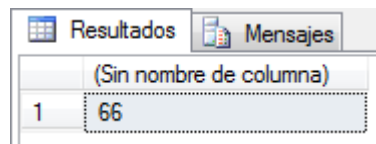


The screenshot shows a SQL Server interface with two tabs: 'Resultados' (Results) and 'Mensajes' (Messages). The 'Resultados' tab is active, displaying a single row of data. The column header is '(Sin nombre de columna)' (No column name). The row contains the value '138'.

	(Sin nombre de columna)
1	138

2. Cuenta de los artículos despachados a las diferentes tiendas de la empresa.

```
SELECT COUNT(DISTINCT CodArticulo)
FROM GUIA_DETALLE
go
```



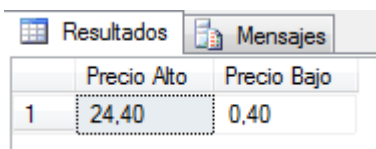
The screenshot shows a SQL Server interface with two tabs: 'Resultados' (Results) and 'Mensajes' (Messages). The 'Resultados' tab is active, displaying a single row of data. The column header is '(Sin nombre de columna)' (No column name). The row contains the value '66'.

	(Sin nombre de columna)
1	66

Ejercicio 6.2: Uso de la funciones MAX() y MIN()

1. Precio más alto y más bajo de los artículos registrados en la tabla ARTICULO.

```
SELECT MAX(precioProveedor) AS 'Precio Alto',
       MIN(precioProveedor) AS 'Precio Bajo'
FROM ARTICULO
go
```



The screenshot shows a SQL Server interface with two tabs: 'Resultados' (Results) and 'Mensajes' (Messages). The 'Resultados' tab is active, displaying two columns of data: 'Precio Alto' and 'Precio Bajo'. The row contains the values '24,40' and '0,40'.

	Precio Alto	Precio Bajo
1	24,40	0,40

2. Guía de envío más reciente y más antigua.

```
SELECT MAX(fechaSalida), MIN(fechaSalida)
FROM GUIA_ENVIO
go
```

	Resultados	Mensajes
	(Sin nombre de columna)	(Sin nombre de columna)
1	2013-04-19 20:38:06.400	2013-03-20 20:38:04.447

3. Nombre del primer artículo y del último artículo si se ordenaran en base a su descripción.

```
SELECT MIN(descripcionArticulo),
       MAX(descripcionArticulo)
FROM ARTICULO
go
```

	Resultados	Mensajes
	(Sin nombre de columna)	(Sin nombre de columna)
1	7 UP DESCARTABLE	YOGURT YOLEIT FRESA

Ejercicio 6.3: Uso de la funciones de agregación para cálculos estadísticos

El siguiente ejemplo muestra el uso de las funciones estadísticas.

```
SELECT AVG(PrecioProveedor) AS 'Promedio',
       STDEV(PrecioProveedor)
       AS 'Desviación estándar',
       STDEVP(PrecioProveedor)
       AS 'Desviación estándar población',
       VAR(PrecioProveedor) AS 'Varianza',
       VARP(PrecioProveedor)
       AS 'Varianza población'
```



```
FROM ARTICULO
go
```

Resultados		Mensajes			
	Promedio	Desviación estándar	Desviación estándar población	Varianza	Varianza población
1	5,7468	5,54546632749484	5,52533752667282	30,7521967893791	30,5293547836589

Ejercicio 6.4: Uso de la función SUM()

Monto total de los artículos enviados a las tiendas.

```
SELECT SUM(precioVenta * cantidadEnviada)
FROM GUIA_DETALLE
go
```

 Resultados		 Mensajes	
(Sin nombre de columna)			
1	378095,00		

Total de unidades despachadas del producto 27.

```
SELECT SUM(cantidadEnviada)
FROM GUIA_DETALLE
WHERE codArticulo = 27
go
```

Resultados		Mensajes	
(Sin nombre de columna)			
1	300		

2. LA CLÁUSULA GROUP BY

La cláusula GROUP BY se utiliza para agrupar los registros en base a determinado criterio, y luego ejecutar cálculos sobre las columnas para consolidar los datos para cada uno de los grupos obtenidos. Por ejemplo, puede utilizar GROUP BY para agrupar todas las guías de envío por tienda, y luego calcular el monto total enviado a cada una de ellas.

Sintáxis

```
SELECT lista_columnas,  
       función_agregación(columna),  
       función_agregación(columna), ...  
FROM nombre_tabla  
[ WHERE condición_filas ]  
GROUP BY lista_columnas  
      [ HAVING condición_grupos ]
```

- Las columnas presentes en **lista_columnas** de la cláusula GROUP BY deben necesariamente estar presentes en la **lista_columnas** de SELECT.
- Cualquier columna presente en SELECT, y que no se encuentre en la **lista_columnas** de GROUP BY debe estar afectada por una **función_agregación**.
- **condición_grupos** en la cláusula HAVING permite establecer una expresión lógica que indica que los grupos a mostrar en el resultado son aquellos para los que el valor de la expresión es verdadero.
- Una consulta GROUP BY solo entrega una fila por cada grupo generado. Esta fila muestra el resultado de la **función_agregación** aplicada sobre el grupo. No muestra el contenido del grupo.
- Cuando se utiliza GROUP BY sobre una columna que contiene valores NULL, éstos se procesan como un grupo.

Ejercicio 6.5: Uso de la cláusula GROUP BY

1. Cantidad de artículos registrados para cada línea.

```
SELECT codLinea,  
       COUNT(codArticulo) AS Artículos  
FROM ARTICULO  
GROUP BY codLinea  
go
```

	Resultados	Mensajes
	codLinea	Artículos
1	1	25
2	2	25
3	3	41
4	4	17
5	5	15
6	6	15

2. Cantidad de artículos por proveedor para las líneas 1 y 5.

```
SELECT codLinea, codProveedor,
       COUNT(codArticulo) AS Artículos
FROM ARTICULO
WHERE codLinea IN (1, 5)
GROUP BY codLinea, codProveedor
ORDER BY codLinea
go
```

Resultados		Mensajes	
	codLinea	codProveedor	Artículos
1	1	14	13
2	1	15	12
3	5	8	4
4	5	9	6
5	5	10	5

3. Monto total enviado por artículo.

```
SELECT codArticulo,
       SUM(precioVenta * cantidadEnviada)
       AS 'Monto total'
FROM GUIA_DETALLE
GROUP BY codArticulo
ORDER BY 'Monto total' DESC
go
```


Resultados		Mensajes
	codArticulo	Monto total
1	130	27000,00
2	124	22500,00
3	125	22500,00
4	129	21150,00
5	64	21000,00
6	127	20250,00
7	126	16875,00
8	132	15750,00
9	66	15000,00
10	65	15000,00

4. Artículos cuyo monto total enviado es mayor a **20,000**.

```

SELECT codArticulo,
       SUM(precioVenta * cantidadEnviada)
       AS 'Monto total'
FROM GUIA_DETALLE
GROUP BY codArticulo
HAVING SUM(precioVenta * cantidadEnviada) > 20000
ORDER BY 'Monto total' DESC
go

```

Resultados		Mensajes
	codArticulo	Monto total
1	130	27000,00
2	124	22500,00
3	125	22500,00
4	129	21150,00
5	64	21000,00
6	127	20250,00

2.1. Uso de GROUP BY con los operadores ROLLUP y CUBE

El operador ROLLUP permite resumir los valores de grupo. El operador ROLLUP se usa normalmente para producir promedios acumulados o sumas acumuladas.

Puede hacer esto aplicando la función de agregación en la lista de columnas de SELECT para cada columna en la cláusula GROUP BY moviéndose de izquierda a derecha.

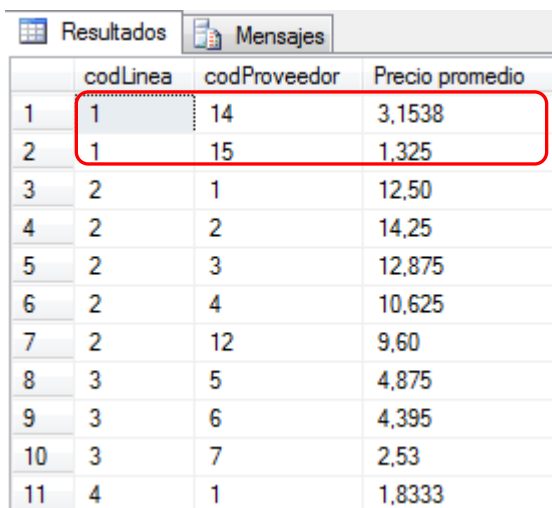
El operador CUBE permite crear y resumir todas las posibles combinaciones de grupos basadas en la cláusula GROUP BY.

El siguiente ejercicio ilustra el uso de estos operadores.

Ejercicio 6.6: Uso de los operadores ROLLUP y CUBE

1. Ejecute la siguiente consulta:

```
SELECT codLinea, codProveedor,  
       AVG(precioProveedor) AS 'Precio promedio'  
FROM ARTICULO  
GROUP BY codLinea, codProveedor  
ORDER BY codLinea, codProveedor  
go
```



	codLinea	codProveedor	Precio promedio
1	1	14	3,1538
2	1	15	1,325
3	2	1	12,50
4	2	2	14,25
5	2	3	12,875
6	2	4	10,625
7	2	12	9,60
8	3	5	4,875
9	3	6	4,395
10	3	7	2,53
11	4	1	1,8333

La consulta muestra el precio promedio para cada pareja **línea-proveedor**. En la imagen se destaca las parejas **línea1-proveedor14** con

su precio promedio **3.1538**, y **línea1-proveedor15** con su precio promedio **1.325**.

2. Ahora, ejecute la siguiente consulta usando el operador ROLLUP.

```
SELECT codLinea, codProveedor,  
       AVG(precioProveedor) AS 'Precio promedio'  
FROM ARTICULO  
GROUP BY codLinea, codProveedor  
WITH ROLLUP  
go
```

	codLinea	codProveedor	Precio promedio
1	1	14	3,1538
2	1	15	1,325
3	1	NULL	2,276
4	2	1	12,50
5	2	2	14,25
6	2	3	12,875
7	2	4	10,625
8	2	12	9,60
9	2	NULL	12,612
10	3	5	4,875
18	5	8	1,96
19	5	9	2,2066
20	5	10	16,768
21	5	NULL	6,9946
22	6	5	4,6866
23	6	6	4,576
24	6	11	13,00
25	6	NULL	6,8666
26	NULL	NULL	5,7468

La consulta muestra el precio promedio para cada pareja **línea-proveedor**. En la imagen se destaca las parejas **línea1-proveedor14** con

su precio promedio **3.1538**, y **línea1-proveedor15** con su precio promedio **1.325**.

También muestra el precio promedio de cada línea tomando en cuenta a todos los proveedores. En la imagen se destaca la **línea1** cuyo precio promedio es **2.276**. En el caso de las consultas GROUP BY, el valor NULL que se muestra en la columna **codProveedor** debe interpretarse como "todos".

En la última fila del resultado de la consulta se muestra el precio promedio de todos los productos. Observe que en la columna **codLinea** el valor es NULL ("todas" las líneas), y en la columna **codProveedor** el valor también es NULL ("todos" los proveedores). El precio promedio de todos los productos es **5.7468**.

3. Finalmente ejecute la siguiente consulta con el operador CUBE.

```
SELECT codLinea, codProveedor,  
       AVG(precioProveedor) AS 'Precio promedio'  
FROM ARTICULO  
GROUP BY codLinea, codProveedor  
WITH CUBE  
go
```

	codLinea	codProveedor	Precio promedio
1	NULL	NULL	5,7468
2	NULL	1	6,10
3	NULL	2	9,3312
4	NULL	3	12,875
5	NULL	4	10,625
6	NULL	5	4,8236
7	NULL	6	4,4312
8	NULL	7	2,53
9	NULL	8	1,96
10	NULL	9	2,2066

Si recorre el resultado de la consulta notará que la consulta con CUBE entrega los mismos resultados que la consulta con ROLLUP, pero además proporciona el precio promedio para cada proveedor. En la imagen se destaca al **proveedor2** cuyo precio promedio para "todas" las líneas es **6.10**.

Las siguientes son formas alternas de especificación de los operadores ROLLUP y CUBE.

```
SELECT codLinea, codProveedor,  
       AVG(precioProveedor) AS 'Precio promedio'  
FROM ARTICULO  
GROUP BY ROLLUP(codLinea, codProveedor)  
go
```

```
SELECT codLinea, codProveedor,  
       AVG(precioProveedor) AS 'Precio promedio'  
FROM ARTICULO  
GROUP BY CUBE(codLinea, codProveedor)  
go
```

3.LA CLÁUSULA OVER CON FUNCIONES DE AGREGACIÓN

La cláusula OVER usada con las funciones de agregación permite crear particiones de datos bajo determinado criterio, y calcular la función de agregación para cada partición con la ventaja de que permite mostrar los elementos que forman el consolidado de la partición.

Ejercicio 6.7: Uso de la cláusula OVER con la función SUM()

1. Ejecute la siguiente consulta:

```
SELECT CodLinea, DescripcionArticulo,  
       StockActual AS Unidades  
FROM ARTICULO
```

go

Resultados

Mensajes

	CodLinea	DescripcionArticulo	Unidades
1	1	CARAMELOS BASTON VIENA ARCOR	200
2	1	CARAMELOS SURTIDO DE FRUTAS	300
3	1	CARAMELOS FRUTAS SURTIDA ARCOR	250
4	1	CARAMELOS FRUTAS MASTICABLES	250
5	1	CHUPETES LOLY AMBROSOLI	150
6	1	FRUNA SURTIDA DONOFRIO	500
7	1	CHOCOLATE DOÑA PEPA FIELD	500
8	1	CHOCOLATE CUA CUA FIELD	500
9	1	MELLOWS FAMILIAR FIELD	100
10	1	WAFER CHOCOLATE FIELD	900

La consulta muestra el stock de cada artículo registrado en la tabla ARTICULO.

2. Ahora, ejecute la siguiente consulta de agregación:

```
SELECT CodLinea,  
       SUM(StockActual) AS 'Total unidades'  
FROM ARTICULO  
GROUP BY CodLinea  
go
```

Resultados		Mensajes
	CodLinea	Total unidades
1	1	7016
2	2	1990
3	3	38259
4	4	6415
5	5	4770
6	6	42556

La consulta acumula el stock de los artículos que pertenecen a la misma línea.

3. Finalmente, ejecute la siguiente consulta usando la cláusula OVER:

```
SELECT CodLinea, DescripcionArticulo,  
       StockActual AS Unidades,  
       SUM(StockActual) OVER(PARTITION BY CodLinea)  
       AS 'Total unidades'  
FROM ARTICULO  
go
```

Resultados		Mensajes			
	CodLinea	DescripcionArticulo	Unidades	Total unidades	
1	1	CARAMELOS BASTON VIENA ARCOR	200	7016	
2	1	CARAMELOS SURTIDO DE FRUTAS	300	7016	
3	1	CARAMELOS FRUTAS SURTIDA ARCOR	250	7016	
4	1	CARAMELOS FRUTAS MASTICABLES	250	7016	
5	1	CHUPETES LOLY AMBROSOLI	150	7016	
6	1	FRUNA SURTIDA DONOFRIO	500	7016	
7	1	CHOCOLATE DOÑA PEPA FIELD	500	7016	
8	1	CHOCOLATE CUA CUA FIELD	500	7016	
9	1	MELLOWS FAMILIAR FIELD	100	7016	
10	1	WAFER CHOCOLATE FIELD	900	7016	

Esta última consulta muestra la data entregada por las 2 consultas anteriores: el acumulado del stock de los artículos que son de la misma línea, y el stock de cada artículo por separado.

4. EL OPERADOR PIVOT

Con mucha frecuencia necesitamos ver los datos en base a múltiples dimensiones (ó entidades). En las versiones antiguas de SQL Server disponemos de los operadores ROLLUP y CUBE que se utilizan con la cláusula GROUP BY, y permiten ver data resumida en base a distintas dimensiones.

Las últimas versiones de SQL Server incorporan el operador PIVOT que es más fácil de entender e implementar que los operadores ROLLUP y CUBE. Por ejemplo, con el operador PIVOT podemos rotar filas y mostrarlas como columnas para obtener una vista diferente de los datos. El resultado de PIVOT es una tabla de doble entrada.

Sintáxis

```
SELECT...  
...  
PIVOT( función_agregación( columna_numérica )  
      FOR columna_dimensión  
      IN ( lista_valores_columna_dimensión ) )
```

- **columna_numérica** es la columna cuyos valores se desea mostrar en una tabla de doble entrada.
- **columna_dimensión** es la columna cuyos valores en una consulta normal se ven como filas, y que en la consulta PIVOT se desea ver como columnas.
- **lista_valores_columna_dimensión** son los valores de **columna_dimensión** a mostrar como columnas en la tabla de doble entrada.

Ejercicio 6.8: Uso del operador PIVOT

Escriba y ejecute la siguiente consulta:

```
SELECT codLinea, codProveedor,  
       AVG(precioProveedor) AS 'Precio promedio'  
FROM ARTICULO  
GROUP BY codLinea, codProveedor  
ORDER BY 1,2  
go
```


	codLinea	codProveedor	Precio promedio
1	1	14	3,1538
2	1	15	1,325
3	2	1	12,50
4	2	2	14,25
5	2	3	12,875
6	2	4	10,625
7	2	12	9,60
8	3	5	4,875
9	3	6	4,395
10	3	7	2,53

La consulta muestra el precio promedio por cada pareja línea-proveedor.

A continuación crearemos un reporte a modo de tabla de doble entrada, pero solo con el precio promedio de las parejas para los proveedores 1, 14 y 15.

```

SELECT codLinea,
       [1] AS Proveedor1, [14] AS Proveedor14,
       [15] AS Proveedor15
FROM
  (SELECT precioProveedor, codLinea, codProveedor
   FROM ARTICULO) origen
PIVOT (AVG(precioProveedor)
      FOR codProveedor
      IN ([1], [14], [15])) AS destino
go

```

	codLinea	Proveedor1	Proveedor14	Proveedor15
1	1	NULL	3,1538	1,325
2	2	12,50	NULL	NULL
3	3	NULL	NULL	NULL
4	4	1,8333	NULL	NULL
5	5	NULL	NULL	NULL
6	6	NULL	NULL	NULL

Muestra el precio promedio de todas las líneas para los proveedores 1, 14 y 15.

Finalmente, "pivotaremos" el resultado anterior para que la consulta muestre a los proveedores 1, 14 y 15 como filas, y a las líneas como columnas, pero solo las líneas 1, 2 y 4.

```
SELECT codProveedor,  
       [1] AS Linea1, [2] AS Linea2,  
       [4] AS Linea4  
FROM  
(SELECT precioProveedor, codLinea, codProveedor  
  FROM ARTICULO  
  WHERE codProveedor IN (1, 14, 15)) origen  
PIVOT (AVG(precioProveedor)  
  FOR codLinea  
  IN ([1], [2], [4])) AS destino  
go
```

	codProveedor	Linea1	Linea2	Linea4
1	1	NULL	12,50	1,8333
2	14	3,1538	NULL	NULL
3	15	1,325	NULL	NULL

5. EJERCICIOS PROPUESTOS

Para los siguientes ejercicios debe utilizar las bases de datos RH y EDUCA. Los scripts para crearlas en su servidor los encontrará en el CD que acompaña al libro.

Escriba las instrucciones SELECT para obtener:

4. (RH) Calcular el importe de la planilla del departamento de ventas. Debe incluir el sueldo y la comisión.
5. (RH) Encontrar el mayor y menor sueldo en el departamento de ventas.
6. (RH) Encontrar el salario promedio en la empresa.

7. (RH) Conocer la cantidad de empleados que hay en el departamento de ventas.
8. (RH) Conocer el importe de la planilla del departamento de ventas, con comisión y sin comisión.
9. (EDUCA) Conocer para el curso SQL Server Administración la cantidad de alumnos matriculados y a cuánto asciende el importe que se proyecta recaudar hasta el momento.
10. (EDUCA) Conocer cuál es el importe recaudado hasta el momento para el curso SQL Server Administración.
11. (RH) Encontrar el sueldo promedio por departamento.
12. (EDUCA) Encontrar el importe recaudado por curso.
13. (RH) Conocer el sueldo máximo, sueldo mínimo y el sueldo promedio por departamento.
14. (RH) Conocer cuántos empleados hay por departamento.
15. (RH) Conocer cuántos empleados han ingreso por año en cada departamento.
16. (RH) Conocer la cantidad de empleados, el importe de la planilla y el sueldo promedio.
17. (EDUCA) Conocer para cada curso la cantidad de alumnos matriculados y el importe que se tiene proyectado recaudar por los alumnos matriculados.
18. (RH) Encontrar los departamentos que tienen más de 3 trabajadores.
19. (RH) Conocer cuáles son los puestos de trabajo que tienen más de 2 empleados.
20. (EDUCA) Encontrar los ingresos por mes y los ingresos totales.
21. (RH) Encontrar el importe de la planilla por puesto de trabajo en cada departamento, el total por departamento y el total general.
22. (RH) Encontrar el importe de la planilla por cargo y departamento, encontrando resúmenes por todas las combinaciones posibles de estos datos.