



Funciones de agregación y agrupamiento de datos

En este capítulo veremos las funciones de agregación que nos permiten consolidar datos numéricos. También veremos el uso de la cláusula GROUP BY para obtener reportes con data consolidada según distintos criterios.

Esta página se ha dejado en blanco intencionalmente.

Capítulo 9

Funciones de agregación y agrupamiento de datos

Contenido

- ❑ *Funciones de agregación*
 - ✓ **Ejercicio 80:** *Uso de la función AVG()*
 - ✓ **Ejercicio 81:** *Uso de la función COUNT()*
 - ✓ **Ejercicio 82:** *Uso de las funciones MAX() y MIN()*
 - ✓ **Ejercicio 83:** *Uso de la función SUM()*
- ❑ *La cláusula GROUP BY*
 - ✓ **Ejercicio 84:** *Uso de la cláusula GROUP BY*
 - ✓ *Uso de GROUP BY con el operador ROLLUP*
 - ✓ **Ejercicio 85:** *Uso del operador ROLLUP*
 - ✓ *Uso de GROUP BY con el operador CUBE*
 - ✓ **Ejercicio 86:** *Uso del operador CUBE*
- ❑ *Resúmenes detallados – Uso de COMPUTE...BY*
 - ✓ **Ejercicio 87:** *Uso de COMPUTE...BY*

Esta página se ha dejado en blanco intencionalmente.

Funciones de agregación y agrupamiento de datos

Uno de los requerimientos más comunes cuando se trabaja con bases de datos es la obtención de resultados numéricos consolidados (totales, subtotales, promedios, etc.), y la obtención de reportes con datos consolidados. SQL nos permite cumplir con estos requerimientos mediante el uso de las funciones de agregación y la cláusula GROUP BY de la instrucción SELECT.

Funciones de agregación

Son funciones que permiten efectuar una operación aritmética que resume los valores de una columna de toda la tabla, o que resume los valores de la columna agrupados según determinado criterio. La función produce un solo valor que es el resumen de la tabla, o de cada uno de los grupos.

El siguiente cuadro presenta las funciones de agregación más utilizadas.

Función	Descripción
AVG()	Retorna el promedio de los valores de una columna ó expresión.
COUNT()	Retorna la cuenta del número de filas de una consulta.
MAX()	Retorna el valor máximo de una columna ó expresión.
MIN()	Retorna el valor mínimo de una columna ó expresión.
SUM()	Retorna la suma de los valores de una columna ó expresión.

Puede usar las funciones de agregación con la declaración SELECT o en combinación con la cláusula GROUP BY.

Con excepción de la función COUNT(*), todas las funciones de resumen 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 80: Uso de la función AVG()

Sintaxis

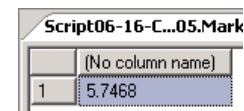
AVG([DISTINCT] *expresión_numérica*)

- **DISTINCT** indica que debe eliminarse los valores duplicados de *expresión_numérica* antes de evaluar la función.

Ejercicio a)

Obtener el precio unitario promedio de todos los productos en la tabla **Producto** de la base de datos **MarketPERU**.

```
SELECT AVG(precioProveedor)
FROM Producto
go
```



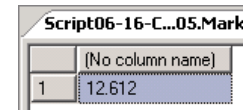
Script06-16-C...05.Mark

	(No column name)
1	5.7468

Ejercicio b)

Precio unitario promedio de los productos de la categoría **2**.

```
SELECT AVG(precioProveedor)
FROM Producto
WHERE idCategoria = 2
go
```



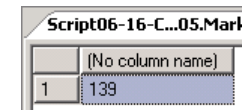
Script06-16-C...05.Mark

	(No column name)
1	12.612

Ejercicio c)

Stock promedio de los productos cuyo precio unitario es mayor a **10.00**.

```
SELECT AVG(stockActual)
FROM Producto
WHERE precioProveedor > 10
go
```



Script06-16-C...05.Mark

	(No column name)
1	139

Ejercicio 81: Uso de la función COUNT()

Sintaxis

```
COUNT( [ DISTINCT ] expresión )  
COUNT( * )
```

- **DISTINCT** indica que debe eliminarse los valores duplicados de expresión antes de evaluar la función.
- **COUNT(*expresión*)** ignora los valores NULL de expresión; COUNT(*) se utiliza para contar filas, por lo que no ignora los valores NULL.

Ejercicio a)

Cuenta de los productos registrados en la base de datos.

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

Script06-16-C...05.Mark	
	(No column name)
1	138

Ejercicio b)

Cuenta de los productos despachados a los diferentes locales de la empresa.

```
SELECT COUNT(DISTINCT idProducto)  
FROM Guia_detalle  
go
```

Script06-16-C...05.Mark	
	(No column name)
1	66

Ejercicio 82: Uso de las funciones MAX() y MIN()

Sintaxis

```
MAX( expresión )
MIN( expresión )
```

- **expresión** indica los valores para los que se debe hallar el máximo y el mínimo. Si expresión es de tipo fecha, MAX retorna la fecha final del rango de valores fecha ordenados, y MIN retorna la fecha inicial.

Ejercicio a)

Precio más alto y más bajo de los productos registrados en la tabla **Producto**.

```
SELECT MAX(precioProveedor),
       MIN(precioProveedor)
FROM Producto
go
```

Script06-16-C...05.MarketPERU*			Summary
	(No column name)	(No column name)	
1	24.40	0.40	

Ejercicio b)

Guía de remisión más reciente y más antigua.

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

Script06-16-C...05.MarketPERU*			Summary
	(No column name)	(No column name)	
1	2005-02-02 11:38:54.683	2005-01-23 11:38:54.263	

Ejercicio c)

Nombre del primer producto y del último producto si se ordenaran en base al nombre.

```
SELECT MIN(nombre),
       MAX(nombre)
FROM Producto
go
```

Script06-16-C...05.MarketPERU*			Summary
	(No column name)	(No column name)	
1	7 UP DESCARTABLE	YOGURT YOLEIT FRESA	

Ejercicio 83: Uso de la función SUM()

Sintaxis

SUM([DISTINCT] *expresión*)

- **DISTINCT** indica que debe eliminarse los valores duplicados de expresión antes de evaluar la función.

Ejercicio a)

Monto total de los productos salidos del almacén.

```
SELECT SUM(precioVenta * cantidad)
FROM Guia_detalle
go
```

Script06-16-C...05.Mark	
	(No column name)
1	378095.00

Ejercicio b)

Total de unidades despachadas del producto 7.

```
SELECT SUM(cantidad)
FROM Guia_detalle
WHERE idProducto = 7
go
```

Script06-16-C...05.Mark	
	(No column name)
1	500

La cláusula GROUP BY

La cláusula GROUP BY se utiliza para agrupar las filas en base a determinado criterio, y luego ejecutar una operación que resume un atributo para cada uno de los grupos así formados. Por ejemplo, puede utilizar GROUP BY para agrupar todas las facturas por cliente, y luego calcular el monto total facturado de cada cliente.

Sintaxis

```
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 84: Uso de la cláusula GROUP BY

Ejercicio a)

Cantidad de productos registrados para cada categoría.

```
SELECT idCategoria,
       COUNT(idProducto) AS Productos
FROM Producto
GROUP BY idCategoria
go
```

	idCategoria	Productos
1	1	25
2	2	25
3	3	41
4	4	17
5	5	15
6	6	15

Ejercicio b)

Cantidad de productos por proveedor para las categorías 2 y 4

```
SELECT idCategoria, idProveedor,
       COUNT(idProducto) AS Productos
FROM Producto
WHERE idCategoria IN (2, 4)
GROUP BY idCategoria, idProveedor
ORDER BY idCategoria
go
```

	idCategoria	idProveedor	Productos
1	2	1	4
2	2	2	10
3	2	3	4
4	2	4	4
5	2	12	3
6	4	1	6
7	4	2	6
8	4	13	5

Ejercicio c)

Monto total despachado por producto.

```
SELECT idProducto,
       SUM(precioVenta * cantidad)
       AS 'Monto total'
FROM Guia_detalle
GROUP BY idProducto
ORDER BY 'Monto total' DESC
go
```

	idProducto	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
11	30	10200.00
12	68	8400.00
13	74	6600.00
14	113	6300.00
15	29	6000.00
16	109	5775.00
17	110	5775.00

Ejercicio d)

Productos cuyo monto total despachado es mayor a **20,000**.

```
SELECT idProducto,
       SUM(precioVenta * cantidad) AS 'Monto total'
FROM Guia_detalle
GROUP BY idProducto
HAVING SUM(precioVenta * cantidad) > 20000
ORDER BY 'Monto total' DESC
go
```

	idProducto	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

Uso de GROUP BY con el operador ROLLUP

Use la cláusula GROUP BY con el operador ROLLUP para 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.

Ejercicio 85: Uso del operador ROLLUP

En este ejemplo, GROUP BY agrupa las filas de la tabla **Producto** en base a la combinación de los valores de las columnas **idCategoria** e **idProveedor**, y para cada una de las combinaciones obtenidas calcula el promedio de columna **precioProveedor**. Luego, el operador ROLLUP agrupa en base a los valores de la primera columna en GROUP BY (**idCategoria**) sin tomar en cuenta los valores de la segunda columna en GROUP BY, y calcula el promedio de **precioProveedor** para los grupos así obtenidos. Finalmente calcula el promedio de la columna **precioProveedor** para toda la tabla.

```
SELECT idCategoria, idProveedor,
       AVG(precioProveedor) AS 'Precio promedio'
FROM Producto
GROUP BY idCategoria, idProveedor
WITH ROLLUP
go
```

Script07-02-U...5.MarketPERU*			
			Summary
	idCategoria	idProveedor	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
11	3	6	4.395
12	3	7	2.53
13	3	NULL	4.3548

12	3	7	2.53
13	3	NULL	4.3548
14	4	1	1.8333
15	4	2	1.1333
16	4	13	3.32
17	4	NULL	2.0235
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

EditorResultsMessages

El resultado de la consulta muestra la siguiente información:

- **Precio promedio de los productos de cada proveedor para cada categoría:** GROUP BY ha agrupado en base a idCategoria e idProveedor obteniendo las combinaciones categoría 1/proveedor 14, y categoría 1/proveedor 15. Para cada una de ellas calculo el promedio de la columna precioProveedor obteniendo 3.1538 y 1.325 respectivamente.
- **Precio promedio de todos los productos de cada categoría:** El operador ROLLUP agrupó en base a la columna idCategoria sin considerar la columna idProveedor obteniendo la combinación categoría 1/NULL, y para este grupo calculó el promedio de la columna precioProveedor obteniendo el valor 2.276.
- **Precio promedio de todos los productos:** Finalmente, calculó el promedio de la columna precioProveedor para toda la consulta obteniendo la combinación NULL/NULL con el resultado 5.7468.

Uso de GROUP BY con el operador CUBE

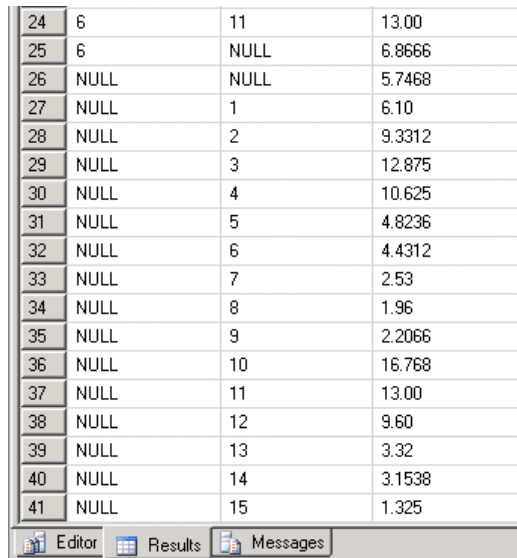
Use la cláusula GROUP BY con el operador CUBE para crear y resumir todas las posibles combinaciones de grupos basadas en la cláusula GROUP BY.

Ejercicio 86: Uso del operador CUBE

En este ejemplo, el operador CUBE entrega las mismas filas que el operador ROLLUP, pero añade las combinaciones generadas al agrupar en base a la segunda columna en GROUP BY (**idProveedor**) sin tomar en cuenta los valores de la primera columna (**idCategoria**).

```
SELECT idCategoria, idProveedor,  
       AVG(precioProveedor) AS 'Precio promedio'  
FROM Producto  
GROUP BY idCategoria, idProveedor  
        WITH CUBE  
go
```

24	6	11	13.00
25	6	NULL	6.0666
26	NULL	NULL	5.7468
27	NULL	1	6.10
28	NULL	2	9.3312
29	NULL	3	12.875
30	NULL	4	10.625
31	NULL	5	4.8236
32	NULL	6	4.4312
33	NULL	7	2.53
34	NULL	8	1.96
35	NULL	9	2.2066
36	NULL	10	16.768
37	NULL	11	13.00
38	NULL	12	9.60
39	NULL	13	3.32
40	NULL	14	3.1538
41	NULL	15	1.325



El resultado de la consulta muestra la siguiente información:

- **Precio promedio de los productos de cada proveedor para cada categoría:** GROUP BY ha agrupado en base a idCategoría e idProveedor obteniendo las combinaciones categoría 1/proveedor 14, y categoría 1/proveedor 15. Para cada una de ellas calculo el promedio de la columna precioProveedor obteniendo 3.1538 y 1.325 respectivamente.
- **Precio promedio de todos los productos de cada categoría:** El operador ROLLUP agrupó en base a la columna idCategoría sin considerar la columna idProveedor obteniendo la combinación categoría 1/NULL, y para este grupo calculó el promedio de la columna precioProveedor obteniendo el valor 2.276.
- **Precio promedio de todos los productos:** Finalmente, calculó el promedio de la columna precioProveedor para toda la consulta obteniendo la combinación NULL/NULL con el resultado 5.7468.
- **Precio promedio de todos los productos de cada proveedor:** El operador CUBE ha generado todos los grupos que generó ROLLUP, y ha añadido los grupos generados en base a la columna idProveedor sin tener en cuenta los valores de la columna idCategoría obteniendo las combinaciones NULL/proveedor 1, NULL/proveedor 2, y así sucesivamente.

Resúmenes detallados – Uso de COMPUTE...BY

La cláusula COMPUTE...BY permite agrupar las filas en base a los valores de una ó mas columnas, y luego efectuar alguna operación de resumen sobre cada grupo así generado.

La diferencia con GROUP BY es que permite mostrar información mas detallada de cada grupo.

Sintaxis

```
SELECT lista_columnas
FROM nombre_tabla
[ WHERE condición_filas ]
ORDER BY columnaX, columnaP, ...
COMPUTE función_agregación( columnaM )
      BY columnaX, columnaP, ...
```

- **columnaX**, **columnaP** son las columnas en base a cuyos valores se formarán los grupos.
- **columnaM** es la columna en la que se desea ejecutar el cálculo indicado por la *función_agregación*.
- COMPUTE...BY requiere que la consulta se ordene en base a las columnas utilizadas para la generación de los grupos.

Ejercicio 87: Uso de COMPUTE...BY

Ejercicio a)

La siguiente consulta agrupa las filas de la tabla **Producto** en base al valor de la columna **idCategoria**, y luego, calcula el precio promedio de los productos de cada categoría. Muestra el código de la categoría, el código del producto, el nombre del producto, la unidad de medida, y el precio unitario para cada producto contenido en cada uno de los grupos.

```
SELECT idCategoria, idProducto, nombre,
       unidadMedida, precioProveedor
FROM Producto
ORDER BY idCategoria
COMPUTE AVG(precioProveedor) BY idCategoria
go
```

SQLQuery1.sql...5.MarketPERU* Summary					
	idCategoria	idProducto	nombre	unidadMedida	precioProveec
1	1	1	CARAMELOS BASTON VIE...	PAQUETE 454 GR	1.50
2	1	2	CARAMELOS SURTIDO DE...	PAQUETE 450 GR	1.00
3	1	3	CARAMELOS FRUTAS SU...	PAQUETE 520 GR	1.50
4	1	4	CARAMELOS FRUTAS MA...	PAQUETE 454 GR	1.30
5	1	5	CHUPETES LOLY AMBROS...	KILOGRAMO	1.20
6	1	6	FRUNA SURTIDA DONOFR...	PAQUETE X 24 UN...	1.80
7	1	7	CHOCOLATE DONIA BEPA	PAQUETE X 6 UNID	2.20
	avg				
1					2.276
	idCategoria	idProducto	nombre	unidadMedida	precioProveec
1	2	26	JAMONADA LAIVE	KILOGRAMO	12.50
2	2	27	JAMONADA ESPECIAL LA S...	KILOGRAMO	10.50
3	2	28	JAMONADA POLACA OTTO...	KILOGRAMO	12.50
4	2	29	JAMONADA DE POLLO SAN...	KILOGRAMO	10.00
5	2	30	JAMONADA ESPECIAL OTT...	KILOGRAMO	17.00

Ejercicio b)

```
SELECT idProveedor, idCategoria, idProducto,
       nombre, precioProveedor
FROM Producto
ORDER BY idProveedor, idCategoria
COMPUTE COUNT(idProducto), MAX(precioProveedor)
       BY idProveedor, idCategoria
go
```

Esta consulta agrupa las filas de la tabla **Producto** en base a la columna **idProveedor**; luego, crea subgrupos en base a la columna **idCategoria**. Para cada subgrupo cuenta cuántos productos existen, y determina el precio más alto.




SQLQuery1.sql...5.MarketPERU* Summary

	idProveedor	idCategoria	idProducto	nombre	precioProveedor
1	1	2	26	JAMONADA LAIVE	12.50
2	1	2	32	JAMON INGLES LAIVE	20.50
3	1	2	37	HOT DOG LAIVE PELADO	5.50
4	1	2	44	CHORIZO PARRILLERO LAIVE	11.50

	cnt	max
1	4	20.50

	idProveedor	idCategoria	idProducto	nombre	precioProveedor
1	1	4	92	CREMA DE LECHE LAIVE	2.00
2	1	4	97	YOGURT LAIVE FRESA	2.00
3	1	4	98	CREAM CHESSE LAIVE	3.00
4	1	4	99	CREMA DE QUESO LAIVE	3.00
5	1	4	101	MANTEQUILLA LAIVE C/SAL	0.50
6	1	4	102	MANTEQUILLA LAIVE	0.50

	cnt	max
1	6	3.00

 Editor  Results  Messages

Esta página se ha dejado en blanco intencionalmente.