Arquitectura y Administración de bases de datos con SQL 2021

Héctor Manuel Garduño Castañeda

Diciembre, 2021

scidata

Contenido

+ La columna en azul la podemos pensar como una "columna de discriminación" pues con esta voy a discriminar los diferentes grupos/categorias:

la columna en azul puede tener varias CATEGORIAS, y estoy interesado en conocer algo de cada una de esas categorias (se dice que la columan tiene datos de tipo categóricos, como si fuera una particion)

El comando **GROUP BY** se utiliza con **SELECT** para agrupar el conjunto de resultados por una o mas columnas.

SELECT ⟨⟨nombre de la columna⟩⟩, ⟨⟨tipo de función⟩⟩**(**⟨⟨nombre de otra columna⟩⟩**)**
FROM ⟨⟨nombre de la tabla⟩⟩
(**WHERE** condiciones extra)
GROUP BY ⟨⟨nombre de la columna⟩⟩**;**

↳ Alguna función agregada

+ Esta ha sido una de las sintaxis más complejas, queda mejor explicada con los ejemplos

+ De las sentencias más importantes

Práctica

Contar todos los clientes de cada región y mostrar los resultados como "Total de clientes"; ¿cuántos productos de cada tipo se vendieron? Ordena los resultados del mayor al menor; muestra por región y estado el total de clientes y su edad promedio.

SELECT region **AS** "Región", **COUNT**(customer_id) **AS** "Total de clientes"
FROM customer **GROUP BY** region;

- **SELECT** product_id **AS** "Producto", **SUM**(quantity) **AS** Total_vendidos
FROM sales **GROUP BY** product_id
ORDER BY Total_vendidos **DESC;**

Importante esa coma

- **SELECT** region **AS** "Región", state **AS** "Estado",
COUNT(customer_id) **AS** "Total de clientes", **AVG**(age) **AS** "Edad promedio"
FROM customer **GROUP BY** region, state;

Más completo

Importante: GROUP BY se puede hacer con variables categóricas

Práctica

Preferible llamarla sentencia SELECT-GROUP BY

Para cada cliente, calcula el mínimo, el máximo, el promedio y el total de ingresos que ha realizado para la compañía y muestra los 5 clientes mas redituables.

```
SELECT customer_Id,
MIN(sales) AS min_ventas,
MAX(sales) AS max_ventas,
AVG(sales) AS prom_ventas,
SUM(sales) AS total_ventas
FROM sales
GROUP BY customer_Id
ORDER BY prom_ventas DESC
LIMIT 5;
```

+ Con este ejemplo se puede entender mejor la sentencia GROUP BY:

1. Definimos en la sentencia SELECT quienes serán los categóricos (la partición), al final ponemos ","

2. Definimos los datos que queremos comparar de esos datos categóricos

3. FROM

4.GROUP BY

El comando **HAVING** se utiliza en combinación con **GROUP BY** para restringir los grupos que retorna el conjunto de resultados y nos devuelva solo aquellas filas que cumplan una condición.

Función de agregación

SELECT ⟨⟨nombre de la columna⟩⟩, ⟨⟨tipo de función⟩⟩(⟨⟨nombre de otra columna⟩⟩)
FROM ⟨⟨nombre de la tabla⟩⟩
(**WHERE** condiciones extra)
GROUP BY ⟨⟨nombre de la columna⟩⟩
HAVING ⟨⟨condiciones en las filas⟩⟩**;**

Columna de la función de agregación

Nota. La diferencia entre filtrar con **WHERE** y **HAVING** es que el **HAVING** se aplica a la función de agregación y el **WHERE** no.

+ El HAVING pone una condición que debe cumplir los registros de la columna de la función de agregación para ser mostrados

Práctica

Mostrar las regiones que tengan mas de 200 clientes; Mostrar las regiones que tengan entre 15 y 20 clientes, inclusive, cuyos nombres inicien con A;

esto es un WHERE

SELECT region, **COUNT**(customer_id) **AS** "Total de clientes"
FROM customer
GROUP BY region
HAVING COUNT(customer_id)>200;

SELECT region, **COUNT**(customer_id) **AS** "Total de clientes"
FROM customer
WHERE customer_name **LIKE** 'A%'
GROUP BY region
HAVING COUNT(customer_id) **BETWEEN** 15 **AND** 20;

+ El de 15 a 20 clientes es un HAVING, porque para saber el no. de clientes por cada región necesito un GROUP BY

scidata