Consultas agrupadas

Objetivos

- Comprender para que son los reportes y análisis de datos .
- Contar registros.
- Contar registros sin repeticiones.
- Agrupar registros.
- Sumar datos.
- Promediar registros
- Subconsultas

Consultas de reportería

La reportería, se refiere a la creación de informes y análisis de datos a partir de información almacenada en bases de datos. La capacidad de crear consultas de reportería utilizando bases de datos es fundamental en el mundo empresarial y en el análisis de datos, ya que permite extraer, transformar y presentar datos de manera efectiva para la toma de decisiones informadas.

Algunos de los usos de reporteria son los siguientes:

Realizar Análisis: SQL permite realizar análisis de datos complejos, como la agregación de valores, la clasificación de registros y la agrupación de datos. Esto es esencial para identificar patrones, tendencias y relaciones en los datos.

Crear Informes y Dashboards: Los resultados de las consultas SQL se pueden utilizar para generar informes visuales y tablas de datos. Estos informes pueden ser presentados de manera que sean fáciles de entender y ayuden en la toma de decisiones.

Contando registros

Contar registros en una base de datos es una operación esencial en la gestión de datos y tiene múltiples aplicaciones útiles en diversos contextos.

La cantidad de registros en una base de datos, a menudo referida como el "recuento de registros", es una métrica fundamental para comprender y administrar los datos almacenados. Esto es relevante en una amplia variedad de situaciones, desde el seguimiento de la salud de una base de datos hasta la toma de decisiones basadas en datos.

La función agregada COUNT

La función COUNT en SQL se utiliza para contar el número de registros en una tabla. Puedes aplicar COUNT a toda la tabla o limitarlo a un subconjunto de registros que cumplan con ciertas condiciones.

Esta función es especialmente útil en la gestión de bases de datos y en la creación de consultas que requieren estadísticas sobre la cantidad de datos disponibles.

Sintaxis Básica: La sintaxis básica de COUNT se ve así: COUNT(columna) o COUNT(*), donde columna es opcional y representa la columna que deseas contar. Si usas *, se contarán todas las filas en la tabla.

```sql
SELECT COUNT(*) FROM mi_tabla;
```

También puedes usar WHERE para realizar operaciones sobre los registros que cumplan con la condición dada.

```sql
SELECT COUNT(*) FROM mi_tabla WHERE columna = 'valor';
```

La cláusula DISTINC

La cláusula DISTINCT se usa para asegurar que los resultados de una consulta sean únicos. Por ejemplo, si tienes una tabla de empleados y deseas saber cuántos departamentos únicos existen en la empresa, puedes usar DISTINCT junto con COUNT de la siguiente manera:

```sql
SELECT COUNT(DISTINCT departamento) FROM empleados;
```

En este caso se espera un conteo de los departamentos, pero antes usamos DISTINC para dejar fuera de la consulta los departamentos de la empresa con nombres repetidos, de esta forma obtenemos un dato real del conteo.

La cláusula GROUP BY

La cláusula GROUP BY se utiliza para dividir los datos en grupos o conjuntos basados en los valores de una o más columnas específicas en una tabla. Luego, se pueden aplicar funciones de agregación a cada grupo para realizar cálculos y resúmenes sobre esos datos agrupados.

Ejemplo de Uso:

Supongamos que tienes una tabla de ventas que registra cada transacción, incluyendo el producto vendido, la fecha de venta y la cantidad vendida. Si deseas calcular la suma de las ventas por producto, puedes utilizar GROUP BY de la siguiente manera:

```sql
SELECT producto, SUM(cantidad) as total_ventas FROM ventas GROUP BY producto;
```

En este ejemplo, **GROUP BY** producto agrupa las filas de ventas por productos, y luego **SUM(cantidad)** calcula la suma de la cantidad vendida para cada grupo de productos.

Consideraciones Importantes:

Al utilizar GROUP BY, es importante tener en cuenta que solo se pueden seleccionar columnas que estén incluidas en la cláusula GROUP BY o en una función de agregación. También, debes estar seguro de que comprendes cómo se agruparán los datos para obtener resultados precisos.

La función SUM

La función SUM es una herramienta esencial en SQL para realizar cálculos de suma en datos numéricos. Permite agregar los valores de una columna específica en un conjunto de registros, proporcionando un resultado que representa la suma total de esos valores.

Sintaxis:

La sintaxis de la función SUM es la siguiente:

```sql
SELECT SUM(columna) FROM tabla;
```

Ejemplos de Uso:

Suma de Ventas: Supongamos que tienes una tabla de ventas y deseas calcular la suma total de todas las ventas realizadas:

```sql
SELECT SUM(venta) FROM ventas;
```

Suma agrupando por Categoría: Si tienes una tabla de productos con una columna que indica la categoría de cada producto y deseas calcular la suma de los precios por categoría:

```
SELECT categoria, SUM(precio) as total_por_categoria FROM productos GROUP BY categoria;
```

La función AVG

La función AVG es una herramienta esencial en SQL para calcular el promedio de valores numéricos en un conjunto de registros. Permite obtener el valor promedio de una columna específica, lo que es útil para realizar análisis estadísticos y comprender la tendencia general de los datos numéricos.

Sintaxis:

La sintaxis de la función AVG es la siguiente:

```
1    SELECT AVG(columna) FROM tabla;
```

Ejemplos de Uso:

Cálculo del Promedio de Calificaciones: Supongamos que tienes una tabla de calificaciones de estudiantes y deseas calcular el promedio de calificaciones:

```
SELECT AVG(calificacion) FROM calificaciones;
```

Cálculo de promedio usando group by:

```
SELECT estudiante, AVG(calificacion) as promedio_calificaciones FROM calificaciones GROUP BY estudiante;
```

En esta consulta, estamos agrupando los datos por la columna "estudiante". Luego, calculamos el promedio de las calificaciones para cada estudiante. El resultado será algo como:

```
| estudiante   | promedio_calificaciones |
|--------------|-------------------------|
| Estudiante1  | 88.5                    |
| Estudiante2  | 83.0                    |
| Estudiante3  | 93.5                    |
```

Agrupando sumando y promediando

Supongamos que tenemos una tabla de ventas con información sobre ventas realizadas por diferentes vendedores la tabla se vería así:

vendedor	producto	monto
Vendedor1	ProductoA	100
Vendedor2	ProductoB	150
Vendedor1	ProductoA	75
Vendedor2	ProductoA	50
Vendedor1	ProductoB	200

Ahora queremos sumar y promediar las ventas de cada vendedor nuestra consulta se vería así:

```
SELECT vendedor, SUM(monto) as monto_total, AVG(monto) as promedio_monto FROM ventas GROUP BY vendedor, producto;
```

En esta consulta, estamos agrupando los datos por "vendedor" y "producto". Luego, calculamos el monto total de ventas para cada vendedor y el promedio de montos por producto. El resultado será algo como:

vendedor	monto_total	promedio_monto
Vendedor1	375	125
Vendedor2	200	100

La clausula HAVING

La cláusula HAVING es una extensión de SQL que se utiliza en combinación con la cláusula GROUP BY para filtrar filas resultantes de una consulta de agregación basándose en una

condición específica. En otras palabras, HAVING se utiliza para aplicar condiciones a grupos de filas que se crean mediante la cláusula GROUP BY.

Un uso común de HAVING es para restringir los resultados basados en el resultado de una función de agregación, como SUM, COUNT, AVG, etc. Por ejemplo, puedes utilizar HAVING para seleccionar grupos que cumplan con ciertos criterios de suma o promedio.

Supongamos que tenemos una tabla de ventas y queremos encontrar las categorías de productos cuyo total de ventas es mayor que 1000:

```
SELECT categoria, SUM(venta_total) AS total_ventas FROM ventas GROUP BY categoria HAVING SUM(venta_total) > 1000;
```

Diferencia entre WHERE y HAVING

La diferencia principal entre WHERE y HAVING es que WHERE se utiliza para filtrar filas antes de que se realice la agregación (por ejemplo, antes de la cláusula GROUP BY), mientras que HAVING se utiliza para filtrar grupos de filas después de la agregación.

Subconsultas

Las subconsultas, también conocidas como subqueries, son consultas SQL anidadas dentro de otras consultas SQL. Estas subconsultas se utilizan para extraer datos de una o más tablas y luego usar esos datos en una consulta principal para realizar operaciones más complejas o para filtrar los resultados.

Las subconsultas permiten realizar tareas avanzadas de consulta y recuperación de datos en una base de datos. Se pueden utilizar en diversas partes de una consulta SQL, como en cláusulas SELECT, FROM, WHERE, HAVING, JOIN, entre otras. Las subconsultas pueden ser correlacionadas o no correlacionadas:

Subconsulta no correlacionada: Una subconsulta no correlacionada es independiente de la consulta principal y se ejecuta solo una vez antes de que comience la consulta principal. Su resultado es tratado como un valor constante y se utiliza en la consulta principal.

Ejemplo:

Supongamos que tienes una base de datos de una tienda de libros y deseas encontrar todos los libros que tienen un precio mayor al precio promedio de todos los libros en la tienda.

Nuestra búsqueda quedaría así:

```
SELECT title, price FROM books WHERE price > (SELECT AVG(price) FROM books);
```

¿Por qué se considera no correlacionada?

- La subconsulta (SELECT AVG(price) FROM books) se ejecuta solo una vez antes de la consulta principal. Calcula el precio promedio de todos los libros en la tabla "books" y devuelve un solo valor, que es un número que representa ese promedio.

- en la consulta principal, comparamos el precio de cada libro con ese valor único (el promedio calculado anteriormente). No importa cuántos libros haya en la tabla "books" ni cuántos registros hay en total; la subconsulta se ejecuta una sola vez y devuelve un solo valor constante, que luego se compara con el precio de cada libro.

Subconsulta correlacionada: Una subconsulta correlacionada depende de la consulta principal y se ejecuta una vez para cada fila en la consulta principal. La subconsulta puede hacer referencia a valores de la consulta principal, lo que la hace más flexible pero potencialmente más costosa en términos de rendimiento.

Ejemplo:

Supongamos que tienes una tabla "empleados" y deseas encontrar todos los empleados que tienen un salario mayor que el promedio de su departamento.

Nuestra búsqueda quedaría así:

```
SELECT e.nombre, e.salario, e.departamento FROM empleados AS e
WHERE e.salario > (
    SELECT AVG(salario)
    FROM empleados
    WHERE departamento = e.departamento
);
```

¿Por qué se considera correlacionada?

- La subconsulta es la parte que calcula ese promedio de salario, pero aquí está el truco: en la subconsulta, decimos "Quiero calcular el promedio de los salarios de todos los empleados en el mismo departamento que el empleado actual". Esto significa que la subconsulta es diferente para cada empleado porque cada empleado puede estar en un departamento diferente.

- Entonces, para cada empleado, la subconsulta se ejecuta y calcula el promedio de los salarios de los empleados en su departamento específico. Luego, compara el salario del empleado actual con ese promedio.

Ahora que ya sabemos hacer agrupaciones sumas etc. vamos a realizar unos ejercicios para practicar