Guía de estudio N°2- Consultas Agrupadas

¡Hola! Te damos la bienvenida a esta nueva guía de estudio.

¿En qué consiste esta guía?

En la siguiente guía podrás trabajar los siguientes aprendizajes:

- Contar registros en base a condiciones especificadas.
- Usar distinct y contar elementos distintos en base a un criterio.
- Seleccionar sobre campos agrupados.
- Aplicar funciones de agregado sobre campos no agrupados.
- Having vs Where.
- Filtrar resultados agrupados utilizando having.

¡Vamos con todo!

Tabla de contenidos

¡Comencemos!

Motivación

Parte del día a día del trabajo con SQL es generar algún tipo de reporte, hay personas que incluso se especializan en la generación de estos. En este capítulo, aprenderemos sobre cómo generar las consultas más frecuentes en estos reportes, como contar datos, contar datos distintos, promediar datos o agrupar datos por cierta categoría para luego contar cuántos elementos hay de cada categoría.

Funciones de agregado

Las funciones de agregado nos permiten extraer información valiosa de una o más tablas.

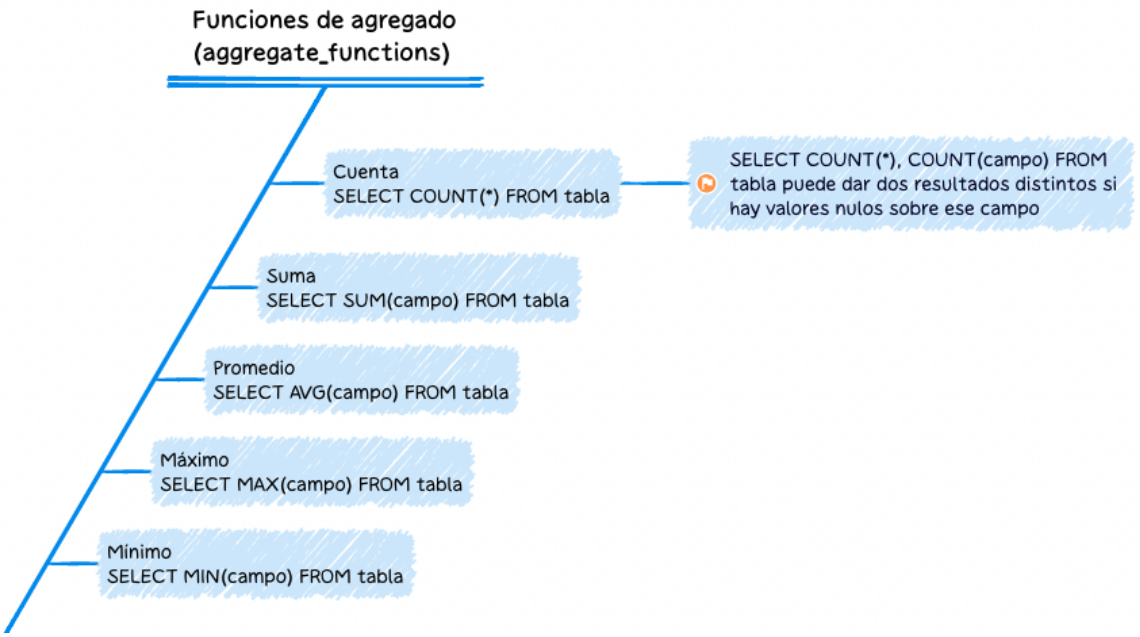

Imagen 1. Resumen funciones de agregado
Fuente: Desafío Latam.

Actividad N° 1

Crea el archivo desafio_actividades.sql donde deberás guardar todas las consultas pedidas a continuación.

- Crea la base de datos actividades_agrupadas

- Dentro de la base de datos ejecuta el siguiente SQL. (cópialo en la actividad para que puedas tener toda la información junta)
- En la tabla cada registro representa un depósito, bitcoinAdress guarda el valor de la dirección donde se hizo el depósito

Bloques de código

```
--Datos actividad 1
create table bitcoins (
      email VARCHAR(50),
      bitcoinAdress VARCHAR(50),
      monto DECIMAL(3,2)
);

insert into bitcoins (email, bitcoinAdress, monto) values
('jaime@email.com', '1HB8RTKNzFAQZ5LtLFRL3R7rbaLGuJt5Un', 0.32);
insert into bitcoins (email, bitcoinAdress, monto) values
('diego@email.com', '15z6eAp7GGforurLkgntSLocJvTafMPThp', 0.08);
insert into bitcoins (email, bitcoinAdress, monto) values
('francisca@email.com', '1NumuVDAuyYGy7b5G19X47dpvYRzRCCc4a', 0.04);
insert into bitcoins (email, bitcoinAdress, monto) values
('francisca@email.com', '1NQpZnNzJL2ntadzXj2PbN9nGgEj8zeHVP', 0.28);
insert into bitcoins (email, bitcoinAdress, monto) values
('jaime@email.com', '1NtsgLEjo3wXNH8ZSfQrF6CY3WRau3ic5Y', 0.53);
insert into bitcoins (email, bitcoinAdress, monto) values
('camila@email.com', '1HbvwhvxXqSUB5FZGZjjrJzi7Y9SpUy4LW', 0.65);
insert into bitcoins (email, bitcoinAdress, monto) values
('francisca@email.com', '14uX7dNXuU657AbNz3fh5K17UZYfmETidb', 0.38);
insert into bitcoins (email, bitcoinAdress, monto) values
('diego@email.com', '18RfzSJsJJej9mzwFNoRs8hpg5tR8bmKim', 0.73);
insert into bitcoins (email, bitcoinAdress, monto) values
('camila@email.com', '115A5LUmNVsjREzCHKbpuec2XueBViJG86', 0.19);
insert into bitcoins (email, bitcoinAdress, monto) values
('jaime@email.com', '1DVP9ATi1H3QUcp7PQcSsTEJdMab1ZM78Q', 0.93);
insert into bitcoins (email, bitcoinAdress, monto) values
('jaime@email.com', '1LcB8bsqyqRmJV8BSFdXMxFwnCWmKj3P7B', 0.83);
insert into bitcoins (email, bitcoinAdress, monto) values
('diego@email.com', '1CNXt7BL8Cm5o8zd7jriC2bUoHmyVUENF7', 0.89);
insert into bitcoins (email, bitcoinAdress, monto) values
('francisca@email.com', '17o3MhRdGL2e5uBUTCiH5mLbVxQ7JHbuck', 0.82);
insert into bitcoins (email, bitcoinAdress, monto) values
('francisca@email.com', '1HtutwGyd573w1a8MB99yU3qZY2uUmvb5V', 0.18);
```

```
insert into bitcoins (email, bitcoinAdress, monto) values
('camila@email.com', '12ZDMxk73Ej6zJZJ2XwnfmGUr4jBUDHmk9', 0.96);
insert into bitcoins (email, bitcoinAdress, monto) values
('jaime@email.com', '13L3hqtQbAzVWbu5zhnMM7v6y7kEM4wW3K', 0.2);
insert into bitcoins (email, bitcoinAdress, monto) values
('jaime@email.com', '14obJfGmgqFQeVeqBtpCWc7YE1r138djQt', 0.55);
insert into bitcoins (email, bitcoinAdress, monto) values
('camila@email.com', '1MggBaGyQGDgSUcb63g4Pqb3FTUg3VQawy', 0.21);
insert into bitcoins (email, bitcoinAdress, monto) values
('camila@email.com', '1L3281ktysdWfdkfGhq5SrmLkrwvTMdozS', 0.07);
insert into bitcoins (email, bitcoinAdress, monto) values
('francisca@email.com', '19CLwEhWsVW2oHUC6Dy66abetBbqB4qTiQ', 0.53);
insert into bitcoins (email, bitcoinAdress, monto) values
('jaime@email.com', '1A5oc25Q2Kj36aVT5JN3FBkXaHHrcKAR81', 0.43);
insert into bitcoins (email, bitcoinAdress, monto) values
('francisca@email.com', '14qXaaDbKMiTbovAd8772uZ6YDy2bWeGZm', 0.04);
insert into bitcoins (email, bitcoinAdress, monto) values
('jaime@email.com', '1M3dMYYngXbfbbjympmZAP57CaVDcx8Ffx', 0.95);
insert into bitcoins (email, bitcoinAdress, monto) values
('diego@email.com', '16WDsrHZ375PKBCdYLhhxrGFtguDQtJDSq', 0.61);
insert into bitcoins (email, bitcoinAdress, monto) values
('jaime@email.com', '1Jq2d8qk6a5p35Y3eqnrsgzDMdzTTCWUXv', 0.41);
insert into bitcoins (email, bitcoinAdress, monto) values
('diego@email.com', '15CBFjbUw46xH2md5eEeFpzaLHRZgATuT7', 0.72);
insert into bitcoins (email, bitcoinAdress, monto) values
('jaime@email.com', '1AFXgLKNCwPVTxVuHedTQrQY65yrAmyst5', 0.98);
insert into bitcoins (email, bitcoinAdress, monto) values
('camila@email.com', '19iMdr6JK6HgarHsebHmMMVjdxnNbybG6L', 0.8);
insert into bitcoins (email, bitcoinAdress, monto) values
('jaime@email.com', '1EL98fxNT1LyS6C8ckFBqocTho9NsEQATS', 0.96);
insert into bitcoins (email, bitcoinAdress, monto) values
('camila@email.com', '1B8xZAcBbU3mTfb6bNJvGtLdPanyAK9kN6', 0.79);
```

- Cuenta la cantidad de registros.
- Selecciona el monto de transacción más alto.
- Selecciona el monto de transacción más bajo.
- Selecciona el promedio de transacciones.
- Selecciona la suma total de transacciones.

Distinct

La cláusula distinct nos permite seleccionar todos los campos distintos.

```
SELECT DISTINCT(campo) From tabla
```

Por ejemplo, si queremos seleccionar todos los emails distintos podemos hacer:

```
SELECT DISTINCT(email) From bitcoins
```

Es posible combinar distinct con funciones de agrupados.

```
SELECT COUNT(DISTINCT(campo)) From tabla
```

Nos devolvería todos los registros cuyo campo es distinto.

Actividad N° 2

Utilizando lo aprendido agrega al archivo desafio_actividades.sql de la actividad anterior queries que conteste las siguientes preguntas:

- ¿Cuántos emails únicos hay? (Selecciona la cantidad de emails únicos)
- ¿Cuántos montos de transacción únicos hay?

Actividad N° 3: Agrupando datos con Group by

Hasta ahora solo hemos utilizado funciones de agregado con todos los datos, pero también podemos agrupar los datos en base a algún criterio.

```
--Datos actividad 3
CREATE TABLE ventas (
 categoria VARCHAR(50),
 monto INT
);

INSERT INTO ventas (categoria, monto) VALUES ('Books', 214);
INSERT INTO ventas (categoria, monto) VALUES ('Games', 293);
INSERT INTO ventas (categoria, monto) VALUES ('Baby', 241);
INSERT INTO ventas (categoria, monto) VALUES ('Tools', 719);
INSERT INTO ventas (categoria, monto) VALUES ('Tools', 385);
INSERT INTO ventas (categoria, monto) VALUES ('Movies', 882);
```

```
INSERT INTO ventas (categoria, monto) VALUES ('Outdoors', 654);
INSERT INTO ventas (categoria, monto) VALUES ('Baby', 332);
INSERT INTO ventas (categoria, monto) VALUES ('Grocery', 332);
INSERT INTO ventas (categoria, monto) VALUES ('Toys', 952);
INSERT INTO ventas (categoria, monto) VALUES ('Games', 682);
INSERT INTO ventas (categoria, monto) VALUES ('Books', 527);
INSERT INTO ventas (categoria, monto) VALUES ('Kids', 980);
INSERT INTO ventas (categoria, monto) VALUES ('Grocery', 300);
```

Podemos agrupar por categoría y mostrarlas

```
SELECT categoría FROM ventas GROUP by categoria;
```

Lo cual es equivalente a seleccionar todas las categorías distintas

```
SELECT distinct(categoria) FROM ventas;
```

Salvo una excepción importante. Al agrupar, podemos utilizar funciones de agregado

```
SELECT categoria, count(categoria) FROM ventas GROUP by categoria;
```

```
SELECT categoria, sum(monto) FROM ventas GROUP by categoria;
```

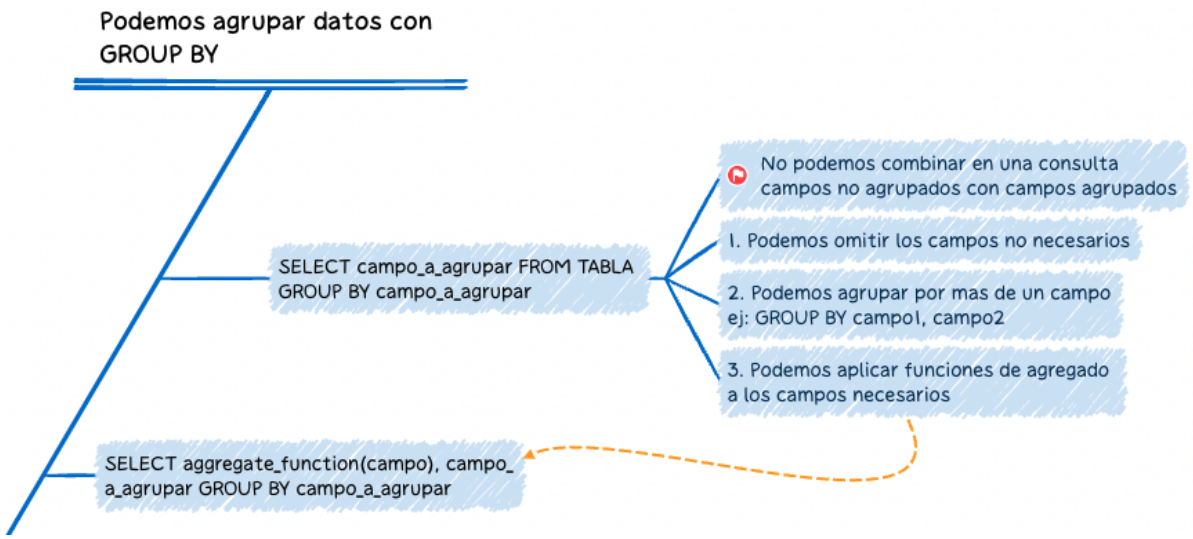

Imagen 2. Resumen Cláusula GROUP BY
Fuente: Desafío Latam.

Actividad N° 4

Agrega las siguientes consultas al archivo de desafio_actividades.sql utilizando los datos de la actividad 3, funciones de agregado y group by

- Seleccionar el monto total de bitcoins de cada usuario
- Seleccionar la cantidad de transacciones de cada categoría
- Calcular el promedio de venta de cada categoría
- Seleccionar el mínimo vendido de cada categoría
- Seleccionar el máximo vendido de cada categoría

Utilizando los datos de la actividad 1

- Seleccionar la cantidad total de depósitos recibidos por cada usuario
- Seleccionar el monto total de bitcoins de cada usuario
- Seleccionar el menor depósito recibido por cada usuario
- Seleccionar el mayor depósito recibido por cada usuario

Having

La cláusula having nos permite filtrar por el resultado de funciones de agregado. Por ejemplo, usando los datos de la tabla de ventas podemos filtrar los resultados por aquellos que tienen 2 o más ventas:

```
SELECT categoria, COUNT(categoria) FROM ventas GROUP BY categoria HAVING
COUNT(categoria) >= 2;
```

Podemos combinar la consulta de distintas formas, por ejemplo, podemos sumar sólo si la cuenta de elementos es mayor a 3:

```
SELECT categoria, SUM(monto) FROM ventas GROUP BY categoria HAVING
COUNT(categoria) > 3;
```

O podemos contar sólo si la suma total es mayor que cierto valor.

```
SELECT categoria, COUNT(categoria) FROM ventas GROUP BY categoria HAVING
SUM(monto) > 500;
```

Actividad N° 5

Agrega las siguientes consultas al archivo de desafio_actividades.sql utilizando los datos de la actividad 1

- Listar todos los correos de los usuarios que hayan recibido un solo depósito.
- Listar todos los correos de los usuarios que hayan recibido un total de depósitos mayor a 1.5 bitcoins.

Where Vs Having

Una confusión típica es saber cuándo se debe ocupar **where** y cuándo **having**. Having se ocupa cuando la condición es sobre una función de agregado.

Por ejemplo, si queremos filtrar por un correo, o por montos inferiores a cierto valor entonces ocuparemos **where**, si queremos filtrar por la suma o conteo de los valores deberemos ocupar **having**.

Actividad N° 6

Agrega las siguientes consultas al archivo de desafio_actividades.sql utilizando los datos de la actividad 1

- Listar todas las transacciones de la tabla bitcoin que sean mayores o iguales a 0.9 btc.
- Listar todas las transacciones de la tabla bitcoin exceptuando aquellos que de monto superior a 0.5 bitcoins.

Subconsultas

Una subconsulta consiste en hacer una consulta interior dentro de una consulta exterior. Esto nos permite seleccionar registros en función a los resultados de otra consulta.

Imagen 3. Subconsultas
Fuente: Desafío Latam.

Por ejemplo, podemos seleccionar todos los registros que sean mayor al valor promedio, para eso necesitaremos una consulta exterior que seleccione todos los valores que cumplen un valor seleccionado por una subconsulta.

Utilizando los datos de la tabla ventas de la actividad 3, podemos probar la siguiente consulta.

```
SELECT * FROM ventas WHERE monto > (SELECT AVG(monto) FROM ventas);
```

Este tipo de subconsultas recibe el nombre de **single row subqueries** o subconsultas de una sola fila, esto debido a que devuelven una fila o ninguna de resultado.

Actividad N° 7

- Listar todas las transacciones de la tabla bitcoin que sean mayores a el monto promedio de las transacciones
- Listar todas las transacciones de la tabla bitcoin que sean mayores a el monto de la primera transacción en la tabla

Existen formas de hacer las subconsultas con múltiples filas, esto no lo cubriremos en la guía, pero si quieres saber más al respecto puedes buscar sobre la instrucción en el siguiente link

¡Continúa aprendiendo y practicando!