Subconsultas

Las subconsultas, también conocidas como subconsultas anidadas o consultas internas, son consultas SQL que se incluyen dentro de otra consulta. Estas permiten realizar consultas complejas y obtener resultados específicos basados en los resultados de otra consulta. Las subconsultas pueden ser muy útiles para una variedad de tareas, como filtrar resultados, realizar cálculos y combinar datos de múltiples tablas.

```
CREATE TABLE departamentos (
    id INT PRIMARY KEY AUTO_INCREMENT,  -- Identificador único para cada
departamento
    nombre VARCHAR(100) NOT NULL,       -- Nombre del departamento
    ubicacion VARCHAR(255)              -- Ubicación física del departamento
);
```

```
CREATE TABLE empleados (
    id INT PRIMARY KEY AUTO_INCREMENT, -- Identificador único para cada empleado
    nombre VARCHAR(50) NOT NULL,        -- Nombre del empleado
    apellido VARCHAR(50) NOT NULL,      -- Apellido del empleado
    fecha_nacimiento DATE,              -- Fecha de nacimiento del empleado
    fecha_contratacion DATE,            -- Fecha de contratación del empleado
    salario DECIMAL(10, 2) NOT NULL,    -- Salario del empleado
    departamento_id INT,                -- Identificador del departamento al que
pertenece el empleado
    email VARCHAR(100) UNIQUE,          -- Correo electrónico del empleado
    telefono VARCHAR(15),               -- Número de teléfono del empleado
    direccion VARCHAR(255),             -- Dirección del empleado
    ciudad VARCHAR(100),                -- Ciudad del empleado
    estado VARCHAR(100),                -- Estado del empleado
    pais VARCHAR(100),                  -- País del empleado
    codigo_postal VARCHAR(10),          -- Código postal del empleado
    FOREIGN KEY (departamento_id) REFERENCES departamentos(id) -- Clave foránea
que referencia la tabla de departamentos
);
```

```
-- Insertar algunos departamentos
INSERT INTO departamentos (nombre, ubicacion) VALUES
('Ventas', 'Edificio A, Planta 1'),
('Recursos Humanos', 'Edificio B, Planta 2'),
('Tecnología', 'Edificio C, Planta 3'),
('Contabilidad', 'Edificio A, Planta 2');
```

```sql
-- Insertar algunos empleados
INSERT INTO empleados (nombre, apellido, fecha_nacimiento, fecha_contratacion,
salario, departamento_id, email, telefono, direccion, ciudad, estado, pais,
codigo_postal) VALUES
('Juan', 'García', '1990-05-15', '2015-03-20', 50000, 1,
'juan.garcia@example.com', '123-456-7890', 'Calle Principal 123', 'Ciudad A',
'Estado A', 'País A', '12345'),
('María', 'Martínez', '1985-08-25', '2010-06-10', 60000, 2,
'maria.martinez@example.com', '987-654-3210', 'Avenida Central 456', 'Ciudad B',
'Estado B', 'País B', '54321'),
('Carlos', 'López', '1992-11-30', '2018-09-05', 55000, 3,
'carlos.lopez@example.com', '555-555-5555', 'Paseo Peatonal 789', 'Ciudad C',
'Estado C', 'País C', '67890'),
('Ana', 'González', '1988-03-10', '2012-11-15', 70000, 4,
'ana.gonzalez@example.com', '111-222-3333', 'Carrera Residencial 321', 'Ciudad
D', 'Estado D', 'País D', '98765');
```

Tipos de Subconsultas

1. **Subconsultas Escalares**: Estas subconsultas devuelven un solo valor y se pueden utilizar en cualquier lugar donde se pueda usar una expresión. Por ejemplo:

```sql
SELECT
    nombre,
    salario
FROM
    empleados
WHERE
    salario > (SELECT AVG(salario) FROM empleados);
```

```
mysql> SELECT
    ->     nombre,
    ->     salario
    -> FROM
    ->     empleados
    -> WHERE
    ->     salario > (SELECT AVG(salario) FROM empleados);
+--------+----------+
| nombre | salario  |
+--------+----------+
| María  | 60000.00 |
| Ana    | 70000.00 |
| Sofía  | 58000.00 |
| David  | 65000.00 |
+--------+----------+
4 rows in set (0.00 sec)
```

Subconsultas de Columna Única: Estas subconsultas devuelven una columna de resultados y se pueden usar con operadores de comparación como `IN`, `ANY`, `ALL`. Por ejemplo:

```sql
SELECT
    nombre
FROM
    empleados
WHERE
    departamento_id IN (SELECT departamento_id FROM departamentos WHERE
departamento_id = 1);
```

```
mysql> SELECT
    ->     nombre
    -> FROM
    ->     empleados
    -> WHERE
    ->     departamento_id IN (SELECT departamento_id FROM departamentos WHERE departamento_id = 1);
+--------+
| nombre |
+--------+
| Juan   |
| Laura  |
+--------+
2 rows in set (0.00 sec)
```

Subconsultas de Varias Columnas: Estas subconsultas devuelven múltiples columnas y se pueden utilizar en cláusulas como `EXISTS` o en la lista de selección. Por ejemplo:

```sql
SELECT
    nombre, salario
FROM
    empleados
WHERE
    (departamento_id, salario) IN (SELECT departamento_id, MAX(salario) FROM
empleados GROUP BY departamento_id);
```

```
mysql> SELECT
    ->     nombre, salario
    -> FROM
    ->     empleados
    -> WHERE
    ->     (departamento_id, salario) IN (SELECT departamento_id, MAX(salario) FROM empleados GROUP BY departamento_id);

+--------+----------+
| nombre | salario  |
+--------+----------+
| Juan   | 50000.00 |
| María  | 60000.00 |
| Ana    | 70000.00 |
| Sofía  | 58000.00 |
+--------+----------+
4 rows in set (0.00 sec)
```

Subconsultas Correlacionadas: Estas subconsultas hacen referencia a la tabla externa en la subconsulta y se evalúan una vez por cada fila procesada por la consulta externa. Por ejemplo:

```sql
SELECT
    e1.nombre, e1.salario
FROM
    empleados e1
WHERE
    salario > (SELECT AVG(salario) FROM empleados e2 WHERE e2.departamento_id =
e1.departamento_id);
```

```
mysql> SELECT
    ->     e1.nombre, e1.salario
    -> FROM
    ->     empleados e1
    -> WHERE
    ->     salario > (SELECT AVG(salario) FROM empleados e2 WHERE e2.departamento_id = e1.departamento_id);
+--------+----------+
| nombre | salario  |
+--------+----------+
| Juan   | 50000.00 |
| María  | 60000.00 |
| Ana    | 70000.00 |
| Sofía  | 58000.00 |
+--------+----------+
4 rows in set (0.00 sec)
```

Uso de Subconsultas en Cláusulas

En la cláusula `WHERE`:

```sql
SELECT
    nombre
FROM
    empleados
WHERE
    salario > (SELECT AVG(salario) FROM empleados);
```

```
mysql> SELECT
    ->     nombre
    -> FROM
    ->     empleados
    -> WHERE
    ->     salario > (SELECT AVG(salario) FROM empleados);
+--------+
| nombre |
+--------+
| María  |
| Ana    |
| Sofía  |
| David  |
+--------+
4 rows in set (0.00 sec)
```

En la cláusula `FROM`:

```sql
SELECT
    depto_promedio.departamento_id, depto_promedio.salario_promedio
FROM
    (SELECT
        departamento_id,
        AVG(salario) AS salario_promedio
     FROM
        empleados
     GROUP BY
        departamento_id) AS depto_promedio;
```

```
mysql> SELECT
    ->     depto_promedio.departamento_id, depto_promedio.salario_promedio
    -> FROM
    ->     (SELECT
    ->         departamento_id,
    ->         AVG(salario) AS salario_promedio
    ->      FROM
    ->         empleados
    ->      GROUP BY
    ->         departamento_id) AS depto_promedio;
+-----------------+------------------+
| departamento_id | salario_promedio |
+-----------------+------------------+
|               1 |     49000.000000 |
|               2 |     56000.000000 |
|               3 |     56500.000000 |
|               4 |     67500.000000 |
+-----------------+------------------+
4 rows in set (0.00 sec)
```

En la cláusula `SELECT`:

```sql
SELECT
    nombre,
    (SELECT AVG(salario) FROM empleados) AS salario_promedio
FROM
    empleados;
```

```
mysql> SELECT
    ->     nombre,
    ->     (SELECT AVG(salario) FROM empleados) AS salario_promedio
    -> FROM
    ->     empleados;
+--------+------------------+
| nombre | salario_promedio |
+--------+------------------+
| Juan   |     57250.000000 |
| María  |     57250.000000 |
| Carlos |     57250.000000 |
| Ana    |     57250.000000 |
| Laura  |     57250.000000 |
| Pedro  |     57250.000000 |
| Sofía  |     57250.000000 |
| David  |     57250.000000 |
+--------+------------------+
8 rows in set (0.00 sec)
```

En la cláusula `HAVING`:

La cláusula `HAVING` en SQL se utiliza para filtrar los resultados de una consulta agregada. Es similar a la cláusula `WHERE`, pero a diferencia de `WHERE`, que filtra las filas antes de que se realicen las funciones de agregación (como `COUNT`, `AVG`, `SUM`, etc.), `HAVING` filtra las filas después de que se han calculado los valores agregados.

cómo funciona `HAVING`:

1. **Agregación de Datos**: Primero, se agrupan los datos utilizando la cláusula `GROUP BY`.
2. **Aplicación de Funciones de Agregación**: Se aplican funciones de agregación (por ejemplo, `SUM`, `AVG`, `COUNT`, `MAX`, `MIN`) a cada grupo de datos.

3. **Filtrado de Resultados**: La cláusula `HAVING` se utiliza para filtrar los grupos de datos en función de los resultados de las funciones de agregación. Solo los grupos que cumplen con la condición especificada en `HAVING` se incluirán en el resultado final.

Supongamos que tienes una tabla `ventas` con las siguientes columnas: `vendedor_id`, `producto`, y `monto`.

id	vendedor_id	producto	monto
1	1	A	100
2	2	B	200
3	1	C	150
4	3	A	300
5	2	B	250

Ahora, se requiere encontrar los vendedores que han generado ventas totales superiores a 300. Se puedes usar `HAVING` de la siguiente manera:

```
SELECT
    vendedor_id,
    SUM(monto) AS total_ventas
FROM
    ventas
GROUP BY
    vendedor_id
HAVING
    SUM(monto) > 300;
```

Proceso de Ejecución:

1. **GROUP BY vendedor_id**:
   Los datos se agrupan por `vendedor_id`.
2. **SUM(monto)**:
   Se calcula la suma de `monto` para cada grupo de `vendedor_id`.
3. **HAVING SUM(monto) > 300**:
   Se filtran los grupos para incluir solo aquellos donde la suma de `monto` es mayor que 300.

Resultado:

vendedor_id	total_ventas
1	250
2	450
3	300

En este ejemplo, solo el `vendedor_id` 2 tendría un `total_ventas` superior a 300, por lo que la salida final sería:

vendedor_id	total_ventas
2	450

La cláusula `HAVING` es esencial cuando necesitas filtrar los resultados de una consulta basada en funciones de agregación y no puede ser reemplazada por `WHERE` en estos casos, ya que `WHERE` no puede filtrar resultados después de la agregación.

```
SELECT
    departamento_id,
    AVG(salario)
FROM
    empleados
GROUP BY
    departamento_id
HAVING
    AVG(salario) > (SELECT AVG(salario) FROM empleados);
```

```
mysql> SELECT
    ->     departamento_id,
    ->     AVG(salario)
    -> FROM
    ->     empleados
    -> GROUP BY
    ->     departamento_id
    -> HAVING
    ->     AVG(salario) > (SELECT AVG(salario) FROM empleados);
+-----------------+--------------+
| departamento_id | AVG(salario) |
+-----------------+--------------+
|               4 | 67500.000000 |
+-----------------+--------------+
1 row in set (0.00 sec)
```