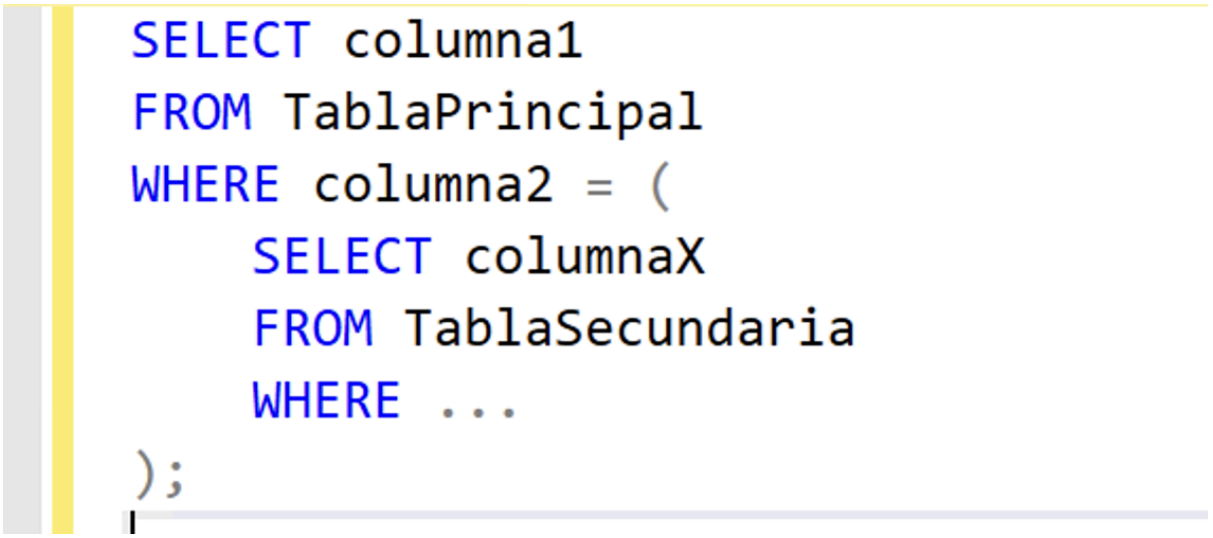
¿Qué es una subconsulta?

Una subconsulta es una consulta completa (una sentencia SELECT) que está anidada dentro de otra sentencia SQL. Por ejemplo, puedes tener algo como:



```sql
SELECT columna1
FROM TablaPrincipal
WHERE columna2 = (
    SELECT columnaX
    FROM TablaSecundaria
    WHERE ...
);
```

El resultado de esa subconsulta (la parte en paréntesis) se utiliza en la consulta principal para comparar, filtrar, calcular, etc.

¿Por qué usar subconsultas?

1. Reutilizar lógica: En lugar de escribir varias consultas separadas o usar variables, integras la lógica dentro de una sola instrucción.

2. Evitar código redundante: Puedes “encapsular” partes de la lógica en subconsultas, manteniendo la consulta principal más limpia.

3. Descomponer problemas: Si la consulta principal requiere datos de otra parte, la subconsulta hace esa extracción de datos en el mismo contexto.

4. Flexibilidad: Las subconsultas se pueden anidar en distintas secciones de una consulta: **SELECT**, **FROM**, **WHERE**, **HAVING**, etc.

¿Dónde se pueden ubicar las subconsultas?

1. En la cláusula WHERE

    - Ejemplo: filtrar filas de la consulta principal usando el resultado de otra consulta.

Uso típico:

```sql
SELECT EmpleadoID, Nombre
FROM Empleados
WHERE Salario > (
    SELECT AVG(Salario)
    FROM Empleados
);
```

    - *Explicación*: Calcula el promedio de salarios de la tabla Empleados en la subconsulta, y filtra los que estén por encima de ese promedio.

2. En la cláusula FROM (tabla derivada o *derived table*)

    - Permite tratar el resultado de la subconsulta como una tabla temporal dentro de la misma consulta.

Uso típico:

```sql
SELECT dt.DepartamentoID, dt.NumEmpleados
FROM (
    SELECT DepartamentoID, COUNT(*) AS NumEmpleados
    FROM Empleados
    GROUP BY DepartamentoID
) AS dt
WHERE dt.NumEmpleados > 5;
```

- *Explicación*: La subconsulta agrupa empleados por departamento y calcula COUNT(*). En la parte externa (la consulta principal), se filtra a los departamentos con más de 5 empleados.

3. En la cláusula SELECT

   - Sirve para calcular valores en tiempo de ejecución y mostrarlos como columna adicional.

Uso típico:

```
SELECT
    E.EmpleadoID,
    E.Nombre,
    (SELECT COUNT(*)
     FROM Ventas V
     WHERE V.EmpleadoID = E.EmpleadoID
    ) AS CantVentas
FROM Empleados E;
```

   - *Explicación*: Cada fila de Empleados se acompaña de otra consulta que cuenta cuántas ventas hizo ese empleado.

4. En la cláusula HAVING

   - Permite filtrar grupos (después de GROUP BY) basados en el resultado de una subconsulta.

Uso típico:

```sql
SELECT DepartamentoID, SUM(Salario) AS TotalSalarios
FROM Empleados
GROUP BY DepartamentoID
HAVING SUM(Salario) > (
    SELECT AVG(Suma)
    FROM (
        SELECT DepartamentoID, SUM(Salario) AS Suma
        FROM Empleados
        GROUP BY DepartamentoID
    ) AS T
);
```

- *Explicación*: Primero agrupa empleados por departamento, calcula la suma de salarios, y luego filtra (HAVING) aquellos departamentos que superan el promedio de las sumas de todos los departamentos.

Proceso de evaluación de una subconsulta

Cuando el motor de base de datos se encuentra una subconsulta, la evalúa (o la integra) de acuerdo con la cláusula en la que se encuentre y el tipo de subconsulta:

1. Subconsulta no correlacionada

    - No depende de valores de la consulta externa; se ejecuta una sola vez o se optimiza como si fuera una consulta independiente.

Ejemplo:

```sql
WHERE Salario > (SELECT AVG(Salario) FROM Empleados)
```

- *No usa columnas de la consulta externa*, por lo que puede calcularse sin mirar las filas de la consulta principal.

2. Subconsulta correlacionada

    - Hace referencia a columnas de la consulta principal, por lo que puede ejecutarse para cada fila de la consulta externa.

Ejemplo:

```sql
SELECT P.ProductoID, P.Precio
FROM Productos P
WHERE P.Precio > (
    SELECT AVG(P2.Precio)
    FROM Productos P2
    WHERE P2.CategoriaID = P.CategoriaID
);
```

- *Explicación*: P.CategoriaID (consulta externa) se compara en la subconsulta (WHERE P2.CategoriaID = P.CategoriaID). El motor debe evaluar la subconsulta para cada producto, calculando el promedio de la categoría de ese producto en particular.

Orden lógico de evaluación (simplificado)

Para entender por qué no puedes usar alias definidos en SELECT en tu WHERE, o cómo se combina todo:

1. FROM (incluyendo subconsultas en FROM, *derived tables* o JOINs)
2. WHERE (usa condiciones de filtrado; si hay subconsultas en WHERE, se ejecutan en este paso)
3. GROUP BY (si existe)
4. HAVING (si existe, filtra los grupos)

5. SELECT (se calculan las columnas seleccionadas, incluidas subconsultas en SELECT)

6. ORDER BY (por último, se ordenan los resultados)

Ejemplo ilustrativo (paso a paso)

Imaginemos esta sentencia:

```sql
SELECT
    E.EmpleadoID,
    E.Nombre,
    (SELECT COUNT(*)
     FROM Ventas V
     WHERE V.EmpleadoID = E.EmpleadoID
    ) AS CantidadVentas
FROM Empleados E
WHERE E.DepartamentoID IN (
    SELECT D.DepartamentoID
    FROM Departamentos D
    WHERE D.NombreDepto = 'Ventas'
);
```

¿Cómo se procesa internamente?

1. Se va a FROM Empleados E: se toma la tabla de empleados.

2. WHERE E.DepartamentoID IN (subconsulta): se ejecuta la subconsulta SELECT D.DepartamentoID FROM Departamentos D WHERE D.NombreDepto = 'Ventas'.

- Esto devuelve un listado de ID de departamentos que se llamen “Ventas”.
- El motor filtra empleados que tengan su DepartamentoID en ese conjunto.

3. (No hay GROUP BY, HAVING, etc.)
4. SELECT: de cada fila resultante (todos los Empleados cuyo DepartamentoID esté en “Ventas”), se ejecuta la subconsulta del SELECT para contar cuántas ventas tienen en la tabla Ventas.
    - Esa subconsulta (SELECT COUNT(*) FROM Ventas V WHERE V.EmpleadoID = E.EmpleadoID) se hace para cada Empleado, usando su EmpleadoID.
    - Al final, se proyectan (se muestran) las columnas: EmpleadoID, Nombre y CantidadVentas.

---

7. Buenas prácticas y consideraciones

1. Optimización:
    - Las subconsultas simples (no correlacionadas) suelen ser optimizadas por el motor, a veces transformándolas en joins internos.
    - Las subconsultas correlacionadas pueden ser más costosas si se ejecutan “fila por fila”.
    - Asegúrate de tener los índices adecuados para las columnas que se comparan en la subconsulta.
2. Repetición de lógica:
    - Si tu subconsulta es compleja y la usas varias veces, podrías considerar usar una CTE (Common Table Expression) o una tabla derivada en FROM para no repetir código.

3. Evitar confusiones con alias:
    - No puedes usar un alias de columna definido en el SELECT dentro del mismo WHERE o HAVING (porque el alias “existe” después de que se evalúan esos pasos).
    - Usa subconsultas directamente en WHERE, o bien tabla derivada/CTE con el alias para luego filtrar en una consulta externa.
4. Cuando reemplazar una subconsulta por un JOIN:
    - A veces, un JOIN es más claro y eficiente.
    - Ejemplo: si tienes una subconsulta que obtiene valores de otra tabla y la comparas con la tabla principal, evaluar si un INNER JOIN o LEFT JOIN con las debidas condiciones sería más simple y rápido.