
🔹 **¿Qué es una subquery en SQL?**

Una **subquery** es una consulta dentro de otra consulta. Sirve para obtener datos que después usa la consulta principal. Es como hacer una pregunta dentro de otra pregunta.

🔹 **Ejemplo sencillo para entenderlo**

Imagina que quieres saber qué empleados ganan más que el salario promedio de la empresa. Primero necesitas calcular el salario promedio y luego compararlo con los salarios individuales.

Sin subquery, tendrías que hacer dos pasos:
1. **Sacar el salario promedio:**

```
SELECT AVG(salario) FROM empleados;
```

2. **Usar ese valor en otra consulta:**

```
SELECT nombre, salario FROM empleados WHERE salario > [salario_promedio];
```

Pero eso es complicado. En vez de eso, podemos usar una subquery:

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

📌 Aquí, la subquery (SELECT AVG(salario) FROM empleados) se ejecuta primero, y el resultado se usa en la consulta principal.

🔹 **Tipos de Subqueries con ejemplos**

📌 **1. Subquery en WHERE (para filtrar datos)**

👉 **Ejemplo:** Encuentra clientes que han hecho pedidos.

```
SELECT nombre
FROM clientes
WHERE ClienteID IN (SELECT ClienteID FROM pedidos);
```

📌 La subquery devuelve los ClienteID de quienes han hecho pedidos, y la consulta principal selecciona sus nombres.

📌 **2. Subquery en SELECT (para calcular valores)**

👉 **Ejemplo:** Comparar el precio de cada producto con el precio promedio.

```
SELECT nombre, precio,
       (SELECT AVG(precio) FROM productos) AS precio_promedio
FROM productos;
```

📌 Aquí la subquery obtiene el precio promedio y lo muestra en cada fila.

📌 3. Subquery en FROM (como tabla temporal)

👉 **Ejemplo:** Filtrar clientes con un gasto promedio mayor a 500.

```
SELECT *
FROM (SELECT ClienteID, AVG(precio) AS gasto_promedio
      FROM pedidos GROUP BY ClienteID) AS subconsulta
WHERE gasto_promedio > 500;
```

📌 La subquery calcula el promedio de gasto por cliente, y la consulta principal filtra los que gastan más de 500.

🔹 Ejercicios (con pistas)

➖ **Ejercicio 1:** Encuentra clientes que no han hecho pedidos.
Pista: Usa NOT IN o NOT EXISTS.

```
SELECT nombre FROM clientes
WHERE ClienteID NOT IN (SELECT ClienteID FROM pedidos);
```

➖ **Ejercicio 2:** Encuentra productos con un precio mayor al promedio.
Pista: Usa AVG() dentro de una subquery.

```
SELECT nombre FROM productos
WHERE precio > (SELECT AVG(precio) FROM productos);
```

➖ **Ejercicio 3:** Encuentra empleados con el segundo salario más alto.
Pista: Usa MAX() dentro de una subquery.

```
SELECT nombre FROM empleados
WHERE salario = (SELECT MAX(salario)
                 FROM empleados
                 WHERE salario < (SELECT MAX(salario) FROM empleados));
```

➖ **Ejercicio 4:** Muestra clientes con la cantidad de pedidos que hicieron.
Pista: Usa COUNT() dentro de SELECT.

```sql
SELECT nombre, (SELECT COUNT(*) FROM pedidos WHERE pedidos.ClienteID = clientes.ClienteID) AS total_pedidos
FROM clientes;
```

➖ **Ejercicio 5:** Encuentra los empleados mejor pagados en cada departamento.
Pista: Usa MAX() con WHERE.

```sql
SELECT nombre, departamento, salario
FROM empleados e
WHERE salario = (SELECT MAX(salario) FROM empleados WHERE departamento = e.departamento);
```

🔹 **¿Cuándo usar subqueries?**

✅ Cuando necesitas hacer cálculos antes de ejecutar la consulta principal.
✅ Cuando quieres comparar datos dinámicamente.
✅ Cuando necesitas filtrar datos en base a otra consulta.

💡 **Consejo:** Si las subqueries se vuelven muy complejas, usa JOIN, que a veces es más eficiente.

📌 ¿Cuál de los ejercicios te gustaría resolver primero? 🚀