SQL Subqueries: Complete Guide for MS SQL

1️⃣ What is a Subquery?

A **subquery** (also called an inner query or nested query) is a query inside another query. It allows you to:

- Filter data dynamically.
- Aggregate and pass results to outer queries.
- Handle complex logic in steps instead of a single large query.

Basic Syntax:

```sql
SELECT column1, column2
FROM Table1
WHERE columnX = (SELECT columnY
                 FROM Table2
                 WHERE condition);
```

💡 **Key Points:**

- Subqueries are enclosed in parentheses `()`.
- They can return **single values**, **multiple values**, or even **entire tables**.
- Subqueries can be in `SELECT`, `FROM`, `WHERE`, `HAVING` clauses.

2️⃣ Types of Subqueries

A. Single-row Subquery

- Returns **one row, one column**.
- Typically used with `=, <, >, <=, >=`.

Example: Find products that match the highest price in their category

```sql
SELECT Name, ListPrice
FROM Production.Product
WHERE ListPrice = (
    SELECT MAX(ListPrice)
    FROM Production.Product
    WHERE ProductSubcategoryID = 1
);
```

☑ Real-world scenario: Identify the most expensive product in each category for reporting or alerting.

B. Multiple-row Subquery

- Returns **multiple rows, single column**.
- Use operators: `IN, NOT IN, ANY, ALL`.

Example: List employees in departments with more than 5 employees

```
SELECT FirstName, LastName, DepartmentID
FROM HumanResources.Employee
WHERE DepartmentID IN (
    SELECT DepartmentID
    FROM HumanResources.Employee
    GROUP BY DepartmentID
    HAVING COUNT(EmployeeID) > 5
);
```

C. Multiple-column Subquery

- Returns **more than one column**.
- Often used with tuples `(col1, col2)` or `EXISTS`.

Example: Find products whose price and weight match specific criteria in another table

```
SELECT p.Name, p.ListPrice, p.Weight
FROM Production.Product p
WHERE EXISTS (
    SELECT 1
    FROM Production.ProductInventory pi
    WHERE pi.ProductID = p.ProductID
      AND pi.Quantity > 50
);
```

D. Correlated Subquery

- The inner query **depends on the outer query**.
- Runs **once per outer row**.

Example: Find employees whose salary is above the average salary of their department

```
SELECT e.BusinessEntityID, e.JobTitle, e.SalariedFlag
FROM HumanResources.Employee e
WHERE e.BusinessEntityID IN (
    SELECT e2.BusinessEntityID
    FROM HumanResources.Employee e2
    WHERE e2.Salary > (
        SELECT AVG(Salary)
        FROM HumanResources.Employee
        WHERE DepartmentID = e.DepartmentID
    )
);
```

☑ Real-world scenario: Calculate **dynamic thresholds per group** for reports or data pipelines.

---

3 Subqueries in Different Clauses

A. In SELECT Clause

Example: Show each product and the number of orders it has

```
SELECT p.Name,
       (SELECT COUNT(*)
        FROM Sales.SalesOrderDetail d
        WHERE d.ProductID = p.ProductID) AS TotalOrders
FROM Production.Product p;
```

B. In FROM Clause

Example: Top 5 most sold products

```
SELECT t.ProductID, t.TotalQuantity
FROM (
    SELECT ProductID, SUM(OrderQty) AS TotalQuantity
    FROM Sales.SalesOrderDetail
    GROUP BY ProductID
) t
ORDER BY t.TotalQuantity DESC
OFFSET 0 ROWS FETCH NEXT 5 ROWS ONLY;
```

C. In WHERE Clause

Example: Products with price higher than average price

```
SELECT Name, ListPrice
FROM Production.Product
WHERE ListPrice > (SELECT AVG(ListPrice) FROM Production.Product);
```

D. In HAVING Clause

Example: Departments with avg salary above company avg

```
SELECT DepartmentID, AVG(Salary) AS AvgDeptSalary
FROM HumanResources.Employee
GROUP BY DepartmentID
HAVING AVG(Salary) > (SELECT AVG(Salary) FROM HumanResources.Employee);
```

4️⃣ Advanced Subquery Techniques for Data Engineering

A. Using `EXISTS` / `NOT EXISTS`

Example: Employees who have at least one sales order

```
SELECT e.BusinessEntityID, e.JobTitle
FROM HumanResources.Employee e
WHERE EXISTS (
    SELECT 1
    FROM Sales.SalesOrderHeader s
    WHERE s.SalesPersonID = e.BusinessEntityID
);
```

B. Nested Subqueries (Multiple levels)

Example: Products with sales greater than average sales of the most selling category

```sql
SELECT Name
FROM Production.Product p
WHERE ProductID IN (
    SELECT ProductID
    FROM Sales.SalesOrderDetail
    WHERE OrderQty > (
        SELECT AVG(OrderQty)
        FROM Sales.SalesOrderDetail
        WHERE ProductID IN (
            SELECT ProductID
            FROM Production.Product
            WHERE ProductSubcategoryID = 1
        )
    )
);
```

C. Subqueries for Data Transformation

Example: Only load active customers with at least 2 orders

```sql
SELECT CustomerID, CompanyName
FROM Sales.Customer
WHERE CustomerID IN (
    SELECT CustomerID
    FROM Sales.SalesOrderHeader
    GROUP BY CustomerID
    HAVING COUNT(SalesOrderID) >= 2
);
```

5️⃣ Best Practices for Subqueries

1. Prefer `JOINs` for large datasets.
2. Use `EXISTS` instead of `IN` for big tables.
3. Use derived tables for complex aggregations.
4. Avoid `SELECT *` inside subqueries.
5. Test nested subqueries individually before combining.

6️⃣ Key Real-World Scenarios for Data Engineering

Scenario	Subquery Type	Example Purpose
Find top customers by sales	Single-row or multi-row	Generate monthly top customer report
Filter data for ETL load	Multi-row subquery	Only load active products with orders > threshold
Compute metrics per group	Correlated	Calculate sales variance per region
Data validation	EXISTS	Ensure foreign key references exist before insert
KPI pipelines	Subquery in SELECT	Add dynamic columns like average sales per category

---

7️⃣ Summary

- **Subquery** = query inside a query.
- Types: **Single-row, Multi-row, Multi-column, Correlated**.
- Can be used in `SELECT`, `FROM`, `WHERE`, `HAVING`.
- Real-world importance: **ETL filtering, KPI calculation, dynamic reporting**.
- Optimization: Use `JOINs` and `EXISTS` carefully for performance.