Let’s learn this **step by step** with **clear examples** using our familiar CollegeDB (with Student and Department tables).

**🌟 1️⃣ What is a Subquery?**

A **subquery** (or nested query) is a query **inside another query**.  
It provides **intermediate results** that the outer query uses.

👉 Syntax:

SELECT column\_name(s)

FROM table\_name

WHERE column\_name operator (SELECT column\_name FROM another\_table WHERE condition);

✅ The **inner query** runs first, and its result is passed to the **outer query**.

**🌟 2️⃣ Example Setup (Recap)**

Let’s make sure we have some sample data to work with 👇

CREATE DATABASE CollegeDB;

USE CollegeDB;

CREATE TABLE Department (

DeptID INT PRIMARY KEY AUTO\_INCREMENT,

DeptName VARCHAR(50),

Location VARCHAR(50)

);

CREATE TABLE Student (

StudentID INT PRIMARY KEY AUTO\_INCREMENT,

Name VARCHAR(50),

Gender CHAR(1),

Marks INT,

DeptID INT,

FOREIGN KEY (DeptID) REFERENCES Department(DeptID)

);

INSERT INTO Department (DeptName, Location)

VALUES ('Computer Science','Block A'),

('Mechanical','Block B'),

('Electronics','Block C');

INSERT INTO Student (Name, Gender, Marks, DeptID)

VALUES ('Mayuri','F',85,1),

('Ravi','M',72,2),

('Anil','M',65,3),

('Sita','F',90,1),

('Deepak','M',58,2),

('Priya','F',77,3);

**🌟 3️⃣ Types of Subqueries**

Subqueries can be of several types:

| **Type** | **Description** |
| --- | --- |
| **Single-row subquery** | Returns one value |
| **Multi-row subquery** | Returns multiple values |
| **Multi-column subquery** | Returns multiple columns |
| **Correlated subquery** | Depends on outer query |
| **Nested subquery in SELECT / FROM / HAVING** | Used in different clauses |

Let’s see examples for each 👇

**✅ A. Single-Row Subquery**

Used when inner query returns **one value**.

-- Find students who scored more than the average marks

SELECT \* FROM Student

WHERE Marks > (SELECT AVG(Marks) FROM Student);

🧠 Explanation:

* Inner query: (SELECT AVG(Marks) FROM Student) gives one value (average marks).
* Outer query: selects students whose marks are greater than that average.

**✅ B. Multi-Row Subquery**

Used when inner query returns **multiple rows** — usually with IN, ANY, ALL.

-- Get students who belong to departments located in 'Block A' or 'Block C'

SELECT \* FROM Student

WHERE DeptID IN (SELECT DeptID FROM Department WHERE Location IN ('Block A', 'Block C'));

🧠 Explanation:

* Inner query returns multiple DeptIDs (say 1 and 3).
* Outer query filters students with those DeptIDs.

**✅ C. Multi-Column Subquery**

Used to compare multiple columns at once.

-- Find (DeptID, Marks) combinations that exist for female students

SELECT DeptID, Marks FROM Student

WHERE (DeptID, Marks) IN (

SELECT DeptID, Marks FROM Student WHERE Gender = 'F'

);

🧠 Explanation:

* Inner query returns pairs of (DeptID, Marks) for female students.
* Outer query checks if other students have the same combinations.

**✅ D. Correlated Subquery**

Subquery that **depends on the outer query** — it runs **once per row**.

-- Students whose marks are greater than the average marks of their own department

SELECT Name, DeptID, Marks

FROM Student s

WHERE Marks > (

SELECT AVG(Marks)

FROM Student

WHERE DeptID = s.DeptID

);

🧠 Explanation:

* Inner query uses s.DeptID from outer query.
* For each department, it calculates average marks and compares each student’s marks.

**✅ E. Subquery in FROM (Inline View)**

Used as a **temporary table** in the FROM clause.

-- Show average marks per department and filter only departments with avg > 75

SELECT DeptID, AvgMarks

FROM (

SELECT DeptID, AVG(Marks) AS AvgMarks

FROM Student

GROUP BY DeptID

) AS DeptAvg

WHERE AvgMarks > 75;

🧠 Explanation:

* Inner query (DeptAvg) calculates average marks by department.
* Outer query filters only those with average > 75.

**✅ F. Subquery in SELECT Clause**

Used to compute a **derived value** for each row.

-- Show each student's marks and the overall average marks (in the same row)

SELECT Name, Marks,

(SELECT AVG(Marks) FROM Student) AS OverallAvg

FROM Student;

🧠 Explanation:

* Subquery in SELECT computes the same value for every row (the overall average).

**✅ G. Subquery in HAVING Clause**

Used to filter groups after aggregation.

-- Show departments whose average marks are higher than the overall average

SELECT DeptID, AVG(Marks) AS DeptAvg

FROM Student

GROUP BY DeptID

HAVING AVG(Marks) > (SELECT AVG(Marks) FROM Student);

🧠 Explanation:

* Inner query gives overall average marks.
* HAVING filters departments based on that.

**🌟 4️⃣ Real-Life Example (Combining Concepts)**

Let’s say you want:

All female students whose marks are **above the average marks of their department**,  
and their department is located in **Block A**.

SELECT s.Name, s.Marks, d.DeptName, d.Location

FROM Student s

JOIN Department d ON s.DeptID = d.DeptID

WHERE s.Gender = 'F'

AND d.Location = 'Block A'

AND s.Marks > (

SELECT AVG(Marks)

FROM Student

WHERE DeptID = s.DeptID

);

🧠 Explanation:

* Joins Student with Department for location.
* Filters females only.
* Subquery ensures marks > average of their department.

**🌟 5️⃣ Summary Table**

| **Type of Subquery** | **Returns** | **Used With** | **Example** |
| --- | --- | --- | --- |
| **Single-row** | 1 value | =, <, > | Marks > (SELECT AVG(Marks)...) |
| **Multi-row** | Multiple rows | IN, ANY, ALL | DeptID IN (SELECT DeptID...) |
| **Multi-column** | Multiple columns | (col1, col2) | (DeptID, Marks) IN (...) |
| **Correlated** | Row-by-row | Uses outer query value | WHERE Marks > (SELECT AVG(Marks) WHERE DeptID=s.DeptID) |
| **In FROM** | Temporary table | Used as dataset | FROM (SELECT ...) AS alias |
| **In SELECT** | Derived column | Computes per-row value | SELECT Name, (SELECT AVG(Marks)) |
| **In HAVING** | Filter aggregates | Compare groups | HAVING AVG(Marks) > (SELECT AVG(Marks)) |