**What is a Subquery?**

A **subquery** (or inner query or nested query) is a query within another SQL query. It helps retrieve data to be used in the main (outer) query.

**Types of Subqueries**

| **Type** | **Description** |
| --- | --- |
| **Single-row** | Returns only one row |
| **Multi-row** | Returns multiple rows |
| **Correlated** | Depends on the outer query; runs once for each row in the outer query |
| **Nested Subquery** | Subquery within another subquery |
| **In SELECT clause** | Subquery used as a computed column |
| **In FROM clause** | Treats subquery as a temporary table (derived table) |

1. **Single-row Subquery**

| **name** | **salary** |
| --- | --- |
| **Alice** | **50000** |
| **Bob** | **70000** |
| **Charlie** | **70000** |
| **Diana** | **60000** |

**Ques:** **fetches the name and salary of employees who have the highest salary.**

SELECT name, salary

FROM employees

WHERE salary = (SELECT MAX(salary) FROM employees);

1. **Multi-row Subquery**

| **department\_id** | **department\_name** | **Location** |
| --- | --- | --- |
| **1** | **HR** | **Bangalore** |
| **2** | **IT** | **Chennai** |
| **3** | **Finance** | **Bangalore** |

| **name** | **department\_id** |
| --- | --- |
| **Alice** | **1** |
| **Bob** | **2** |
| **Charlie** | **3** |
| **Diana** | **3** |

SELECT name

FROM employees

WHERE department\_id IN (SELECT department\_id FROM departments WHERE location = 'Bangalore');

1. **Subquery in FROM Clause**

| **name** | **department\_id** | **salary** |
| --- | --- | --- |
| **Alice** | **1** | **50000** |
| **Bob** | **1** | **70000** |
| **Charlie** | **2** | **60000** |
| **Diana** | **2** | **80000** |
| **Eve** | **3** | **55000** |

SELECT department\_id, AVG(salary) AS avg\_salary

FROM (

SELECT department\_id, salary

FROM employees

) AS emp\_sub

GROUP BY department\_id;

But if you want to:

* Filter rows
* Join with other tables
* Compute temporary fields

**4. Subquery in SELECT Clause**

SELECT name,

(SELECT department\_name FROM departments WHERE departments.department\_id = employees.department\_id) AS dept\_name

FROM employees;

You're using a **subquery in the SELECT clause**, and it's a good example of how to retrieve related data from another table **without using a JOIN**.

* For **each row** in the employees table:
  + It retrieves the name of the employee.
  + It executes a **scalar subquery** to find the corresponding department\_name from the departments table, **where the IDs match**.

This works similarly to a JOIN, but may be **less efficient** on large datasets because the subquery runs once per row (like a **correlated subquery**).

**Important Notes:**

* It works fine if **each department\_id is unique** in the departments table.
* If there are **multiple departments with the same department\_id**, it will raise an error like:  
  Error Code: 1242. Subquery returns more than 1 row

SELECT e.name, d.department\_name AS dept\_name

FROM employees e

JOIN departments d ON e.department\_id = d.department\_id;

| **name** | **department\_id** |
| --- | --- |
| Alice | 1 |
| Bob | 2 |

| **department\_id** | **department\_name** |
| --- | --- |
| 1 | HR |
| 2 | IT |

**5. Correlated Subquery**

SELECT name, salary

FROM employees e1

WHERE salary > (

SELECT AVG(salary)

FROM employees e2

WHERE e1.department\_id = e2.department\_id

);

Here, the inner query refers to the outer query's alias (e1.department\_id) — that makes it **correlated**.

**correlated subquery**, which is a powerful SQL concept used to compare each row against a **computed value** specific to that row.

| **name** | **department\_id** | **salary** |
| --- | --- | --- |
| Alice | 1 | 50000 |
| Bob | 1 | 70000 |
| Charlie | 2 | 60000 |
| Diana | 2 | 80000 |
| Eve | 3 | 55000 |

* For **each employee (e1)**, the subquery:

SELECT AVG(salary)

FROM employees e2

WHERE e1.department\_id = e2.department\_id

calculates the **average salary of the employee's department**.

* Then, the **outer query** selects only those employees **whose salary is greater than their department's average salary**.

SELECT e1.name, e1.salary

FROM employees e1

JOIN (

SELECT department\_id, AVG(salary) AS avg\_salary

FROM employees

GROUP BY department\_id

) AS dept\_avg

ON e1.department\_id = dept\_avg.department\_id

WHERE e1.salary > dept\_avg.avg\_salary;

6. **Using EXISTS**

SELECT name

FROM employees e

WHERE EXISTS (

SELECT 1 FROM departments d WHERE d.department\_id = e.department\_id AND d.location = 'Chennai'

);

Your query uses the **EXISTS** operator, which is ideal when you simply want to **check for the existence of rows** meeting certain conditions.

* For each employee e, the subquery checks:
  + Is there **at least one** department in the departments table
  + Where d.department\_id = e.department\_id
  + And the location is 'Chennai'?
* If such a row **exists**, the employee's name is included in the result.

**Why Use EXISTS?**

* It returns **TRUE/FALSE** for each row.
* It **stops searching as soon as it finds a match**, making it **faster** than IN or = when dealing with **large or non-unique datasets**.

| **Name** | **department\_id** |
| --- | --- |
| Alice | 1 |
| Bob | 2 |
| Charlie | 3 |

| **department\_id** | **department\_name** | **location** |
| --- | --- | --- |
| 1 | HR | Chennai |
| 2 | IT | Bangalore |
| 3 | Finance | Chennai |

**EXISTS vs IN (Comparison):**

* EXISTS: stops at the **first match** → faster
* IN: compares all values → may be slower with many values
* Use EXISTS when working with **correlated subqueries** or large datasets with duplicates.