**SQL Nested Queries**

A nested query (also called a subquery) is a query written inside another SQL query.

* The inner query runs first and provides results for the outer query.
* Subqueries can be placed inside SELECT, FROM, or WHERE clauses.
* They are useful for breaking down complex problems into smaller, manageable steps.

Nested queries are highly useful when:

* We need to retrieve data based on the results of another query.
* We want to perform filtering or aggregation without using joins.
* We need to break down complex operations into smaller steps.

**Sample Tables for Examples**

We will use the following sample tables to demonstrate nested queries:

**STUDENT Table**

The STUDENT table stores information about students, including their unique ID, name, address, phone number, and age.

STUDENT TABLE

**COURSE Table**

The STUDENT\_COURSE table maps students to the courses they have enrolled in. It uses the student and course IDs as foreign keys.

COURSE TABLE

**STUDENT\_COURSE Table**

This table maps students to the courses they have enrolled in, with columns for student ID (S\_ID) and course ID (C\_ID):

STUDENT COURSE TABLE

**Types of Nested Queries in SQL**

There are two primary types of nested queries in SQL, Independent Nested Queries and Correlated Nested Queries. Each type has its own use case and benefits depending on the complexity of the task at hand.

**Independent Nested Queries**

In an independent nested query, the execution of the inner query is independent of the outer query. The inner query runs first and its result is used directly by the outer query. Operators like IN, NOT IN, ANY and ALL are commonly used with independent nested query.

**Example 1: Using IN**

In this Example we will find the S\_IDs of students who are enrolled in the courses ‘DSA’ or ‘DBMS’. We can break the query into two parts:

**Step 1: Find the C\_IDs of the courses:**

This query retrieves the IDs of the courses named 'DSA' or 'DBMS' from the **COURSE** table.

SELECT C\_ID FROM COURSE WHERE C\_NAME IN ('DSA', 'DBMS');

**Output**

| **C\_ID** |
| --- |
| C1 |
| C3 |

**Step 2: Use the result of Step 1 to find the corresponding S\_IDs:**

The inner query finds the course IDs, and the outer query retrieves the student IDs associated with those courses from the STUDENT\_COURSE table

SELECT S\_ID FROM STUDENT\_COURSE

WHERE C\_ID IN (

SELECT C\_ID FROM COURSE WHERE C\_NAME IN ('DSA', 'DBMS')

);

**Output**

| **S\_ID** |
| --- |
| S1 |
| S2 |
| S4 |

**Explanation:** In this example, the inner query retrieves the C\_IDs of the courses 'DSA' and 'DBMS', and the outer query retrieves the student IDs (S\_IDs) enrolled in those courses.

**Correlated Nested Queries**

In correlatednested queries, the inner query depends on the outer query for its execution. For each row processed by the outer query, the inner query is executed. This means the inner query references columns from the outer query. The EXISTS keyword is often used with correlated queries.

**Example 2: Using EXISTS**

In this Example, we will find the names of students who are enrolled in the course with C\_ID = 'C1':

SELECT S\_NAME FROM STUDENT S

WHERE EXISTS (

SELECT 1 FROM STUDENT\_COURSE SC

WHERE S.S\_ID = SC.S\_ID AND SC.C\_ID = 'C1'

);

**Output**

| **S\_NAME** |
| --- |
| RAM |
| RAMESH |

**Explanation:**

For each student in the STUDENT table, the inner query checks if an entry exists in the STUDENT\_COURSE table with the same S\_ID and the specified C\_ID. If such a record exists, the student’s name is included in the output.

**Common SQL Operators for Nested Queries**

SQL provides several operators that can be used with nested queries to filter, compare, and perform conditional checks.

**1. IN Operator**

The IN operator is used to check whether a column value matches any value in a list of values returned by a subquery. This operator simplifies queries by avoiding the need for multiple OR conditions.

**Example: Retrieve student names who enrolled in ‘DSA’ or ‘DBMS’:**

This query filters the students enrolled in the specified courses by chaining multiple nested queries.

SELECT S\_NAME FROM STUDENT

WHERE S\_ID IN (

SELECT S\_ID FROM STUDENT\_COURSE

WHERE C\_ID IN (

SELECT C\_ID FROM COURSE WHERE C\_NAME IN ('DSA', 'DBMS')

)

);

**2. NOT IN Operator**

The NOT IN operator excludes rows based on a set of values from a subquery. It is particularly useful for filtering out unwanted results. This operator helps identify records that do not match the conditions defined in the subquery.

**Example: Retrieve student IDs not enrolled in ‘DSA’ or ‘DBMS’:**

This query excludes students who are enrolled in the courses 'DSA' or 'DBMS'.

SELECT S\_ID FROM STUDENT

WHERE S\_ID NOT IN (

SELECT S\_ID FROM STUDENT\_COURSE

WHERE C\_ID IN (

SELECT C\_ID FROM COURSE WHERE C\_NAME IN ('DSA', 'DBMS')

)

);

**Output**

| **S\_ID** |
| --- |
| S3 |

**3. EXISTS Operator**

The EXISTS operator checks for the existence of rows in a subquery. It returns true if the subquery produces any rows, making it efficient for conditional checks. This operator is often used to test for relationships between tables.

**Example: Find student names enrolled in ‘DSA'**

The inner query checks for matching records in the STUDENT\_COURSE table, and the outer query returns the corresponding student names.

SELECT S\_NAME FROM STUDENT S  
WHERE EXISTS (  
 SELECT 1 FROM STUDENT\_COURSE SC  
 WHERE S.S\_ID = SC.S\_ID AND SC.C\_ID = 'C1'  
);

**4. ANY & ALL Operators**

* ANY: Compares a value with any value returned by the subquery.
* ALL: Compares a value with all values returned by the subquery.

**Example using ANY:**

*SELECT S\_NAME FROM STUDENT  
WHERE S\_AGE > ANY (  
SELECT S\_AGE FROM STUDENT WHERE S\_ADDRESS = 'DELHI'  
);*

**Example using ALL:**

*SELECT S\_NAME FROM STUDENT  
WHERE S\_AGE > ALL (  
SELECT S\_AGE FROM STUDENT WHERE S\_ADDRESS = 'DELHI'  
);*