What is a Subquery?

A **subquery**, also known as an inner query or nested query, is a query embedded within another SQL query. It allows you to use the result of one query as a part of another query.

Advantages of Subqueries:

1. **Isolation of Parts**: Subqueries allow you to break down complex SQL statements into simpler parts, making it easier to understand and manage each component of the query.

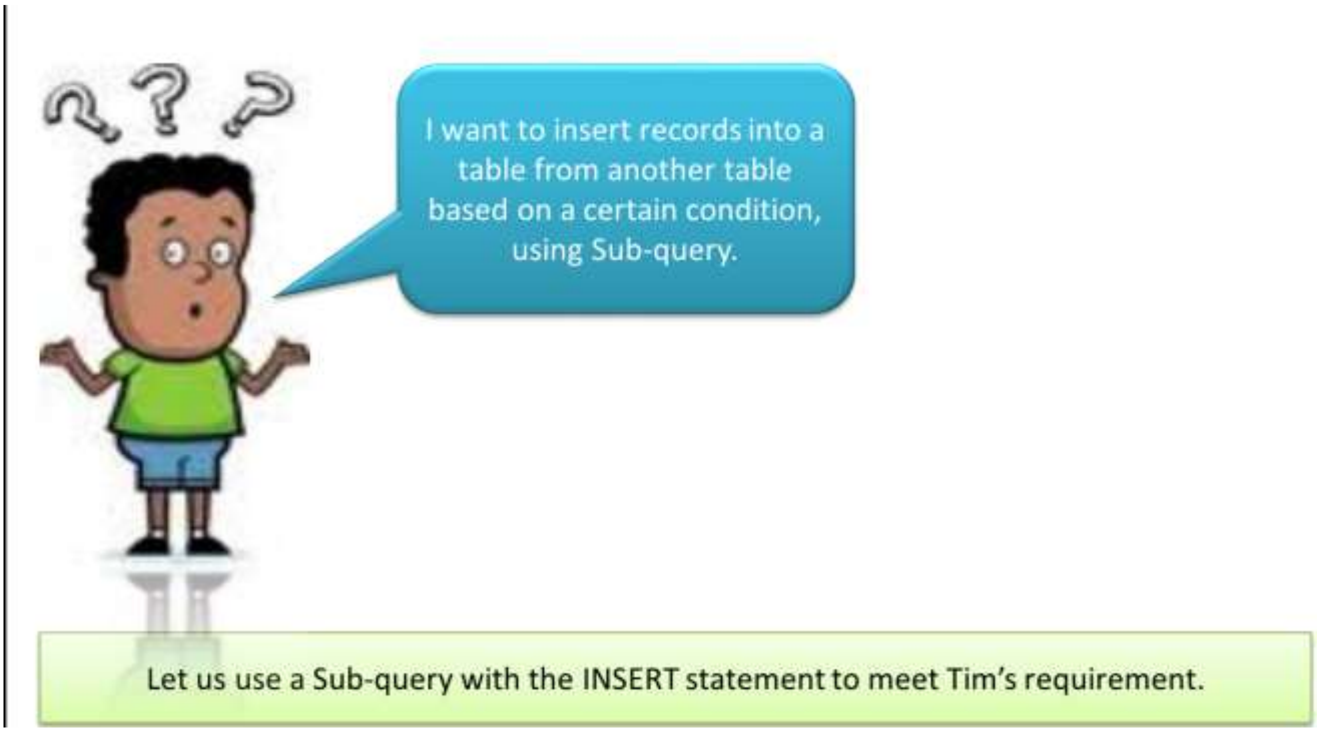
2. **Alternative Operations**: Subqueries provide alternative methods to perform operations that might otherwise require more complex joins or unions. They can simplify the SQL syntax for certain types of queries

3. **Readability**: Many developers find subqueries more readable compared to complex joins or unions, especially when dealing with scenarios involving **correlated data or aggregations**.

Rules for Subqueries

1. **Enclosed in Parentheses**: Subqueries must be enclosed within parentheses.
2. **Single Column in SELECT Clause**: A subquery can have only one column in the SELECT clause unless it is used for comparison with multiple columns in the main query.
3. **No ORDER BY in Subquery**: An ORDER BY clause cannot be used in a subquery, though the main query can have an ORDER BY clause. However, GROUP BY can be used to achieve similar functionality in a subquery.

4. **Multiple Rows with Multi-Value Operators**: Subqueries that return more than one row can only be used with multiple value operators, such as IN, ANY, ALL.
5. **No Direct Set Function Enclosure**: A subquery cannot be directly enclosed in a set function.
6. **BETWEEN Usage**: The BETWEEN operator cannot be used with a subquery directly, but it can be used within the subquery.


I want to insert records into a table from another table based on a certain condition, using Sub-query.
Let us use a Sub-query with the INSERT statement to meet Tim's requirement.

```
CREATE TABLE USA_Offices (
  officeCode VARCHAR(10) NOT NULL,
  city VARCHAR(50) NOT NULL,
  phone VARCHAR(50) NOT NULL,
  addressLine1 VARCHAR(50) NOT NULL,
  addressLine2 VARCHAR(50) NULL,
  state VARCHAR(50) NULL,
  country VARCHAR(50) NOT NULL,
  postalCode VARCHAR(15) NOT NULL,
  territory VARCHAR(10) NOT NULL,
  PRIMARY KEY (officeCode)
);
```

- Let us see how we can copy records having country as USA, from Offices table into USA_Offices table, using Subquery with INSERT statement.

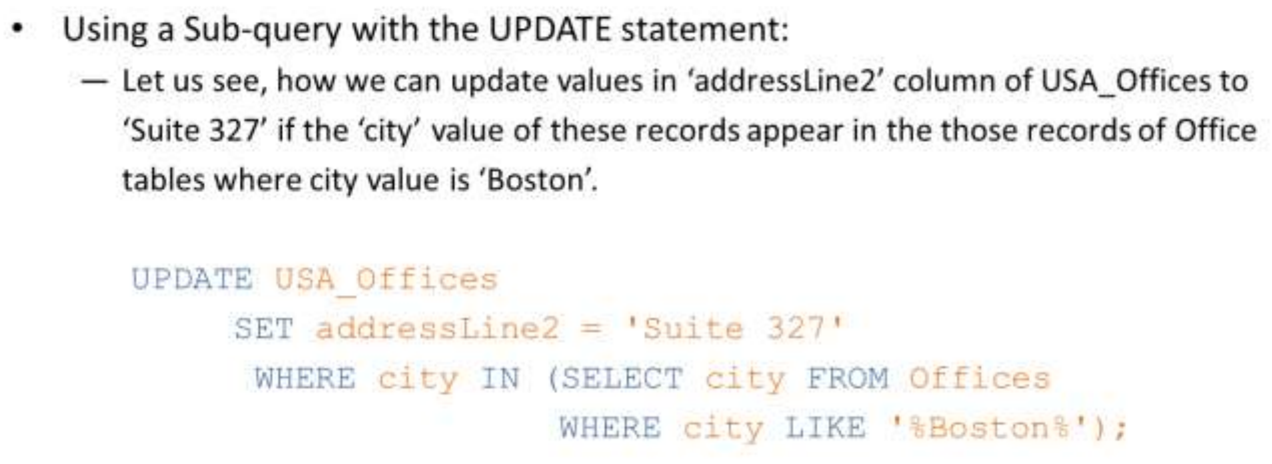
```
  INSERT INTO USA_Offices
  SELECT * FROM Offices
  WHERE country IN (SELECT country FROM offices
                                WHERE country = 'USA');
```

I want to update values in 'addressLine2' column of USA_Offices to 'Suite 327'. This can be done only to the records which has the 'city' value equals to 'Boston' in Office tables.

Let us use a Sub-query with the UPDATE statement to meet Tim's requirement.

```
CREATE TABLE USA_Offices (
 officeCode VARCHAR(10) PRIMARY KEY,
 city VARCHAR(50) ,
 phone VARCHAR(50) ,
 addressLine1 VARCHAR(50) ,
 addressLine2 VARCHAR(50) ,
 state VARCHAR(50) ,
 country VARCHAR(50) ,
 postalCode VARCHAR(15) ,
 territory VARCHAR(10)
);

CREATE TABLE offices (
 officeCode VARCHAR(10) PRIMARY KEY,
 city VARCHAR(50),
 phone VARCHAR(50),
 addressLine1 VARCHAR(50),
 addressLine2 VARCHAR(50),
 state VARCHAR(50),
 country VARCHAR(50),
  postalCode VARCHAR(15),

  territory VARCHAR(10)

);

INSERT INTO offices (officeCode, city, phone, addressLine1,
addressLine2, state, country, postalCode, territory) VALUES

('O6', 'New York', '123-456-7890', '123 Main St', '', 'NY', 'USA',
'10001', 'NA');
```

```
INSERT INTO USA_Offices

SELECT * FROM Offices

WHERE country IN (SELECT country FROM offices

                                    WHERE country = 'USA');
```



- Using a Sub-query with the UPDATE statement:
  - Let us see, how we can update values in 'addressLine2' column of USA_Offices to 'Suite 327' if the 'city' value of these records appear in the those records of Office tables where city value is 'Boston'.

```
UPDATE USA_Offices
       SET addressLine2 = 'Suite 327'
        WHERE city IN (SELECT city FROM Offices
                            WHERE city LIKE '%Boston%');
```

SET SQL_SAFE_UPDATES = 0;

UPDATE USA_Offices SET addressLine2 = 'Suite 327'

WHERE city IN (SELECT city FROM Offices WHERE city LIKE '%NY%');

I want to delete records from USA_Offices where the values in city column of USA_Offices appear in the values in city column of Offices for 'NY' state.

Let us use a Sub-query with the DELETE statement to meet Tim's requirement.

- Using a Subquery with the DELETE statement:
  - Let us see, how we can delete records from USA_Offices where the values in 'city' column of USA_Offices appear in the values in city column of Offices for 'NY' state.

```
DELETE FROM USA_Offices
    WHERE city IN (SELECT city FROM Offices
                                WHERE state LIKE '%NY%');
```

DELETE FROM USA_Offices WHERE city IN (SELECT city FROM Offices

WHERE state LIKE '%NY%')

- Scalar Sub-query:
  - — A scalar Sub-query returns a variable like a number, date, or string.
  - — A scalar Sub-query returns only one column for a single row and is also known as an SQL expression. You can use a scalar Sub-query in:
    - the WHERE clause of a SELECT
    - the VALUES clause of an INSERT statement
    - the SET or WHERE clauses of an UPDATE
    - the WHERE clause of a DELETE statement

- Example:

```
SELECT customerNumber, checkNumber, amount
FROM payments
WHERE amount > (SELECT AVG(amount) FROM payments);
```

- Single Row Sub-query:
  - — A single row Sub-query returns all columns for a single row.
  - — You can use a single row Sub-query in:
    - the INSERT statement when it provides all required values for a single row insertion
    - the SET or WHERE clauses of an UPDATE
    - the WHERE clause of a DELETE statement
- Legal operators for row Sub-query comparisons are: **> < >= <> != <=>**

Step 1: Create the Employees Table

CREATE TABLE Employees (

```
    EmployeeID INT PRIMARY KEY,
    FirstName VARCHAR(50),
    LastName VARCHAR(50),
    Age INT,
    Salary DECIMAL(10, 2),
    Department VARCHAR(50)
);
```

Step 2: Insert Records into the Employees Table

```
INSERT INTO Employees (EmployeeID, FirstName, LastName, Age,
Salary, Department)
VALUES
(1, 'Rajesh', 'Kumar', 30, 60000.00, 'IT'),
(2, 'Anita', 'Sharma', 28, 55000.00, 'HR'),
(3, 'Vikram', 'Singh', 35, 75000.00, 'Finance'),
(4, 'Pooja', 'Reddy', 26, 50000.00, 'IT'),
(5, 'Amit', 'Verma', 32, 70000.00, 'Operations');
```

Step 3: Apply a Single-Row Subquery

Let's say we want to find the details of the employee who has the highest salary. We can achieve this using a single-row subquery.

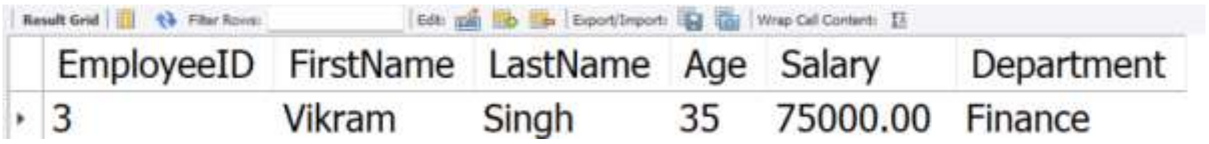
```
SELECT *
FROM Employees
WHERE Salary = (SELECT MAX(Salary) FROM Employees);
```

Explanation

1. **Subquery**: (SELECT MAX(Salary) FROM Employees) - This subquery finds the highest salary in the Employees table.
2. **Main Query**: SELECT * FROM Employees WHERE Salary = ... - This main query retrieves the details of the employee(s) whose salary matches the highest salary found by the subquery.

Expected Result

The result of the above query will return the details of the employee with the highest salary.



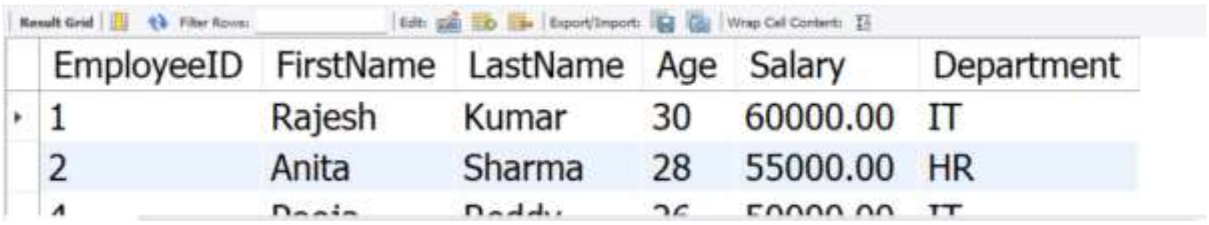
EmployeeID	FirstName	LastName	Age	Salary	Department
3	Vikram	Singh	35	75000.00	Finance

In this example, Vikram Singh has the highest salary of 75,000.00, so his details are returned by the query.

SELECT *

FROM Employees

WHERE Salary < (SELECT MAX(Salary) FROM Employees);

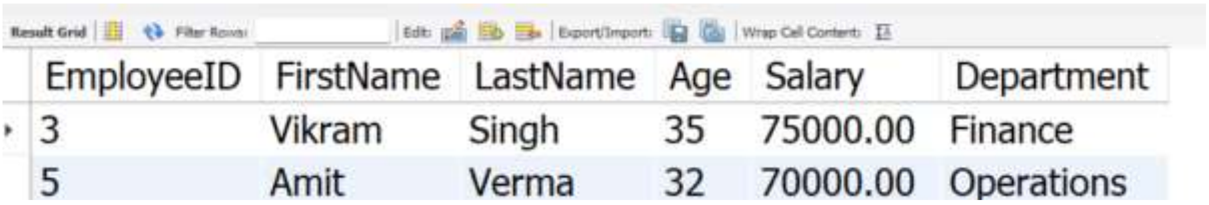


EmployeeID	FirstName	LastName	Age	Salary	Department
1	Rajesh	Kumar	30	60000.00	IT
2	Anita	Sharma	28	55000.00	HR
4	Pooja	Reddy	26	50000.00	IT

SELECT *

FROM Employees5

WHERE Salary > (SELECT AVG(Salary) FROM Employees5);



EmployeeID	FirstName	LastName	Age	Salary	Department
3	Vikram	Singh	35	75000.00	Finance
5	Amit	Verma	32	70000.00	Operations

A multi-row subquery returns multiple rows and is often used with operators like IN, ANY, ALL, EXISTS, etc.

Example: Find Employees in Departments with More Than One Employee

We'll use a multi-row subquery to find departments that have more than one employee and then retrieve the details of employees in those departments.

Step 3: Apply a Multi-Row Subquery

```
SELECT *

FROM Employees

WHERE Department IN (SELECT Department

                     FROM Employees

                     GROUP BY Department

                     HAVING COUNT(*) > 2);
```

Any

```
SELECT *

FROM Employees5

WHERE Salary > ANY (SELECT Salary

                   FROM Employees5

                   WHERE Department = 'HR');
```

All

```sql
SELECT *

FROM Employees

WHERE Salary > ALL (SELECT Salary

                    FROM Employees

                    WHERE Department = 'HR');
```