SQL Joins and Subqueries

1. INNER JOIN

Purpose:

- `INNER JOIN` combines rows from two or more tables based on a related column between them. It returns only the rows that have matching values in both tables.

Example Scenario: Suppose you have two tables: `Orders` and `Customers`.

- `Customers` Table:

CustomerID	Name
1	Alice
2	Bob
3	Charlie

- `Orders` Table:

OrderID	CustomerID	Amount
101	1	150
102	2	200
103	1	250
104	3	300

INNER JOIN Query:

sql

```
SELECT Customers.Name, Orders.OrderID, Orders.Amount
FROM Customers
INNER JOIN Orders ON Customers.CustomerID = Orders.CustomerID;
```

Result:

Name	OrderID	Amount
Alice	101	150
Alice	103	250
Bob	102	200
Charlie	104	300

Explanation:

- The `INNER JOIN` returns rows where there is a match between `Customers.CustomerID` and `Orders.CustomerID`.

2. LEFT JOIN (or LEFT OUTER JOIN)

Purpose:

- `LEFT JOIN` returns all rows from the left table and the matched rows from the right table. If no match is found, NULL values are returned for columns from the right table.

Example Scenario: Let's use the same `Customers` and `Orders` tables.

LEFT JOIN Query:

sql

```
SELECT Customers.Name, Orders.OrderID, Orders.Amount
FROM Customers
LEFT JOIN Orders ON Customers.CustomerID = Orders.CustomerID;
```

Result:

Name	OrderID	Amount
Alice	101	150
Alice	103	250
Bob	102	200
Charlie	104	300
David	NULL	NULL

Explanation:

- `LEFT JOIN` includes all rows from `Customers`, even if there is no matching row in `Orders`. In this example, if a `CustomerID` does not have a corresponding order, the `OrderID` and `Amount` columns will show `NULL`.

3. Subqueries

Purpose:

- A subquery is a query nested within another SQL query. It can be used in various clauses such as `SELECT`, `FROM`, `WHERE`, or `HAVING` to refine the results of the outer query.

Example Scenario: Suppose you want to find all customers who have placed at least one order.

Subquery in `WHERE` Clause:

sql

```
SELECT Name
FROM Customers
WHERE CustomerID IN (SELECT CustomerID FROM Orders);
```

Result:

Name
Alice
Bob
Charlie

Explanation:

- The subquery `(SELECT CustomerID FROM Orders)` retrieves all `CustomerID`s from the `Orders` table. The outer query then selects names from `Customers` where the `CustomerID` matches one in the list returned by the subquery.

Correlated Subquery:

- A correlated subquery refers to columns from the outer query, making it more dynamic and powerful.

Example: Find customers who have placed more than one order:

sql

```
SELECT Name
FROM Customers c
WHERE (SELECT COUNT(*) FROM Orders o WHERE o.CustomerID =
c.CustomerID) > 1;
```

Explanation:

- This query checks each customer (`c.CustomerID`) against the `Orders` table, counting how many orders they've made. If the count is greater than 1, the customer's name is included in the results.

Summary

- **INNER JOIN**: Combines rows from multiple tables where the join condition is met, returning only matching rows.
- **LEFT JOIN**: Returns all rows from the left table and matching rows from the right table, with `NULL` for non-matching rows.
- **Subqueries**: Allow you to perform complex queries by nesting one query inside another. They can be used in various SQL clauses and can be correlated or non-correlated.