SQL Joins and Subqueries

In SQL, joins and subqueries are powerful tools that help you combine data from multiple tables and filter data in more complex ways. Let's break them down into simple concepts.

1. SQL Joins

Joins are used to combine data from two or more tables based on a related column. When you join tables, you can retrieve data that is spread across multiple tables and link them using common values.

INNER JOIN

- The INNER JOIN returns only the rows that have matching values in both tables.
- It excludes rows where there is no match in both tables.

Example:

```
SELECT employees.name, departments.department_name
FROM employees
INNER JOIN departments ON employees.department_id = departments.department_id;
```

This query retrieves the names of employees and their respective department names, but only for employees who are assigned to a department. If an employee does not have a department, they will not appear in the result.

LEFT JOIN (or LEFT OUTER JOIN)

- The LEFT JOIN returns all rows from the left table (the table before the `JOIN` keyword) and the matching rows from the right table (the table after the `JOIN` keyword).
- If there is no match, NULL values are returned for columns from the right table.

Example:

```
SELECT employees.name, departments.department_name
FROM employees
LEFT JOIN departments ON employees.department_id = departments.department_id;
```

This query retrieves all employees, along with their department names. Employees who don't belong to any department will have `NULL` in the `department_name` column.

 # RIGHT JOIN (or RIGHT OUTER JOIN)

- The RIGHT JOIN is the opposite of the LEFT JOIN. It returns all rows from the right table and the matching rows from the left table.
- If there is no match, NULL values are returned for columns from the left table.

Example:

```
SELECT employees.name, departments.department_name
FROM employees
RIGHT JOIN departments ON employees.department_id = departments.department_id;
```

This query retrieves all departments, along with the names of employees working in them. If a department has no employees, the employee name will be `NULL`.

 # FULL JOIN (or FULL OUTER JOIN)

- The FULL JOIN returns all rows when there is a match in either the left or the right table.
- It includes rows that do not have a match in both tables, and places `NULL` in columns where data is missing.

Example:

```
SELECT employees.name, departments.department_name
FROM employees
FULL JOIN departments ON employees.department_id = departments.department_id;
```

This query retrieves all employees and all departments, matching where possible. Employees without a department and departments without employees will still appear in the results with `NULL` in the columns that don't have matching data.

2. Subqueries

A subquery is a query within another query. It allows you to filter or perform calculations based on the results of another query.

 # Subquery in the WHERE Clause

- A subquery can be used in the `WHERE` clause to filter results based on a condition.
- The subquery returns a value (or a set of values) that is used by the main query to narrow down the results.

Example:

```
SELECT name, salary
FROM employees
WHERE department_id = (SELECT department_id FROM departments WHERE
department_name = 'Sales');
```

In this query, the subquery retrieves the `department_id` for the 'Sales' department, and the main query returns the names and salaries of employees who belong to that department.

 # Subquery with IN

- The `IN` keyword is used to match multiple values returned by a subquery.

Example:

```
SELECT name, salary
FROM employees
WHERE department_id IN (SELECT department_id FROM departments WHERE
department_name IN ('Sales', 'Marketing'));
```

Here, the subquery returns the `department_id` values for the 'Sales' and 'Marketing' departments, and the main query retrieves the names and salaries of employees who work in those departments.

 # Subquery in the HAVING Clause

- A subquery can also be used in the `HAVING` clause to filter groups based on aggregated data.

Example:

```
SELECT department_id, AVG(salary) AS avg_salary
FROM employees
GROUP BY department_id
HAVING AVG(salary) > (SELECT AVG(salary) FROM employees);
```

This query calculates the average salary per department and only returns departments where the average salary is greater than the overall average salary across all employees.

 # Subquery in the SELECT Clause

- A subquery can also be used in the `SELECT` clause to perform calculations or retrieve additional information.

Example:

```
SELECT name,
      (SELECT department_name FROM departments WHERE department_id =
employees.department_id) AS department
FROM employees;
```

In this query, the subquery retrieves the department name for each employee based on their `department_id`.

In Summary

- SQL Joins are used to combine data from multiple tables:
  - INNER JOIN: Only returns matching rows from both tables.
  - LEFT JOIN: Returns all rows from the left table and matching rows from the right table (with `NULL` where no match is found).
  - RIGHT JOIN: Returns all rows from the right table and matching rows from the left table (with `NULL` where no match is found).
  - FULL JOIN: Returns all rows from both tables, with `NULL` where there's no match.

- Subqueries are queries within another query, used to filter or calculate values:
  - WHERE: Filter data based on results from another query.
  - IN: Match multiple values returned by a subquery.
  - HAVING: Filter grouped data based on results from a subquery.
  - SELECT: Retrieve additional information using a subquery.

Joins and subqueries allow you to handle complex data retrieval and analysis from multiple tables in a structured way.