Left Join vs Right Join

LEFT JOIN and **RIGHT JOIN** are both types of outer joins in SQL used to retrieve rows from two tables based on a related column. The primary difference between them lies in which table's rows are included in the result when there is no match found in the other table.

LEFT JOIN (or LEFT OUTER JOIN)

The `LEFT JOIN` (or `LEFT OUTER JOIN`) returns all rows from the left table and the matched rows from the right table. If there is no match, the result is NULL on the side of the right table.

RIGHT JOIN (or RIGHT OUTER JOIN)

The `RIGHT JOIN` (or `RIGHT OUTER JOIN`) returns all rows from the right table and the matched rows from the left table. If there is no match, the result is NULL on the side of the left table.

On clause Vs using clause

In SQL, both the `ON` clause and the `USING` clause are used to specify the condition for joining tables, but they are used in slightly different contexts and have different syntax. Here's a comparison of the two:

`ON` Clause

- **Usage:** The `ON` clause is used in `JOIN` operations to specify the condition for the join explicitly. It allows you to define complex join conditions, including multiple columns or different operators.
- **Flexibility:** You can use `ON` to join tables on columns with different names or even create conditions that are not based solely on column equality.

Syntax:

```
SELECT columns
FROM table1
JOIN table2
ON table1.column = table2.column;
```

Example:

```
SELECT e.first_name, d.dept_name
FROM employees e
INNER JOIN departments d
ON e.dept_id = d.id;
```

In this example, `ON` specifies that the `dept_id` column from the `employees` table should match the `id` column from the `departments` table.

`USING` Clause

- **Usage:** The `USING` clause is a shorthand for specifying join conditions when the columns involved in the join have the same name in both tables. It simplifies the syntax by eliminating the need to write the table names and column names twice.
- **Limitation:** It can only be used when joining on columns with the same name and cannot handle more complex conditions.

Syntax:

```
SELECT columns
FROM table1
JOIN table2
USING (common_column);
```

Example:

```
SELECT e.first_name, d.dept_name
FROM employees e
INNER JOIN departments d
USING (dept_id);
```

In this example, `USING` assumes that `dept_id` is a common column in both `employees` and `departments`.

Natural Join vs Inner Join

NATURAL JOIN

- **Definition:** A `NATURAL JOIN` automatically joins tables based on columns with the same name in both tables. It matches columns with the same names and automatically includes them in the result.

- **Usage:** Simpler syntax for joining on columns with the same name, but it can be less explicit.
- **Limitation:** Only works when column names match and can lead to unintended results if columns with the same name are not meant to be joined.

Example:

```
SELECT *
FROM employees
NATURAL JOIN departments;
```

This joins `employees` and `departments` on columns with the same name in both tables (e.g., `dept_id`).

INNER JOIN

- **Definition:** An `INNER JOIN` explicitly specifies the join condition using the `ON` clause. It returns rows where there is a match in both tables based on the condition provided.
- **Usage:** More flexible and explicit, allowing you to specify join conditions on columns with different names or complex conditions.
- **Flexibility:** Can join on columns with different names and supports complex join conditions.

Example:

```
SELECT e.first_name, d.dept_name
FROM employees e
INNER JOIN departments d
ON e.dept_id = d.id;
```

This joins `employees` and `departments` explicitly on `dept_id` from `employees` and `id` from `departments`.

Inner Join vs Left Join

INNER JOIN

- **Definition:** Returns only the rows where there is a match in both tables based on the specified join condition.
- **Usage:** Use when you want to include only the rows that have corresponding matches in both tables.
- **Result:** Excludes rows from both tables if there is no match.

Example:

```
SELECT e.first_name, d.dept_name
FROM employees e
INNER JOIN departments d
ON e.dept_id = d.id;
```

Result: Rows with matching `dept_id` in both `employees` and `departments`.

LEFT JOIN (or LEFT OUTER JOIN)

- **Definition:** Returns all rows from the left table and the matched rows from the right table. If there is no match, the result is NULL for columns from the right table.
- **Usage:** Use when you want to include all rows from the left table regardless of whether there is a match in the right table.
- **Result:** Includes unmatched rows from the left table with NULLs for columns from the right table.

Example:

```
SELECT e.first_name, d.dept_name
FROM employees e
LEFT JOIN departments d
ON e.dept_id = d.id;
```

Result: All rows from `employees`, and matched rows from `departments`. Unmatched rows from `departments` result in NULLs.