SQL Join: SELF JOIN

- A self join is a join that joins a table to itself. This can be useful for comparing data within the same table, or for finding relationships between rows in the same table.
- This type of joining of records with other records in the same table when a condition is matched is also called as a Unary relationship.
- For example, let's say we have a table called `employees` that has columns for the employee's name, department, and manager. We can use a self join to find all of the employees who have the same manager.

```
SELECT e1.name, e2.name
FROM employees e1
JOIN employees e2 ON e1.manager_id = e2.id;

-- OUTPUT
|     name      | manager_name |
|---------------|--------------|
| John Smith    | Tom Lanon    |
| Jane Anderson | Tom Lanon    |
| Tom Lanon     | Anne Connor  |
| Jeremy York   | John Smith   |
```

Self joins can be a powerful tool for analyzing data within a single table. They can be used to find relationships between rows, to compare data, and to identify patterns. If you are working with a table that has a lot of data, a self join can be a great way to get a better understanding of the data.

Here are some other case studies of self join:

- Find all of the employees who have the same manager.
- Find all of the customers who have purchased the same product.
- Find all of the orders that have been shipped to the same address.
- Find all of the products that have been sold by the same salesperson.

EXAMPLE

Let's say we have a table called "Employees" with the following columns: EmployeeID, FirstName, LastName, and ManagerID. The ManagerID column contains the ID of the manager for each employee.

To perform a self join on the "Employees" table, we can retrieve the names of employees and their corresponding managers. The query would look like this:

```
SELECT e.FirstName AS EmployeeFirstName, e.LastName AS EmployeeLastName,
       m.FirstName AS ManagerFirstName, m.LastName AS ManagerLastName
FROM Employees e
JOIN Employees m ON e.ManagerID = m.EmployeeID;
```

This query will join the "Employees" table with itself based on the ManagerID and EmployeeID columns. It retrieves the first and last names of both the employee and their manager. The result will include rows where an employee is joined with their corresponding manager.

EmployeeFirstName	EmployeeLastName	ManagerFirstName	ManagerLastName
John	Smith	Sarah	Johnson
Alice	Brown	Sarah	Johnson
Mark	Davis	Michael	Anderson
Sarah	Johnson	NULL	NULL
Michael	Anderson	NULL	NULL

Let's consider a table called "Products" with the following columns: ProductID, ProductName, CategoryID, and SupplierID. We want to retrieve pairs of products from the same category but different suppliers. The query would look like this:

```
SELECT p1.ProductName AS Product1, p2.ProductName AS Product2,
       p1.SupplierID AS Supplier1, p2.SupplierID AS Supplier2
FROM Products p1
JOIN Products p2 ON p1.CategoryID = p2.CategoryID
WHERE p1.SupplierID <> p2.SupplierID;
```

In this query, we join the "Products" table with itself based on the CategoryID column. The WHERE clause includes the condition "p1.SupplierID <> p2.SupplierID," which ensures that the suppliers of the two products are not equal.

Product1	Product2	Supplier1	Supplier2
Chair	Table	101	102
Desk	Cabinet	103	104
Mouse	Keyboard	201	202

SELF JOIN USING LEFT JOIN

- The use of a left join with a self join becomes more significant when you want to ensure that all rows from the left side are included, even if there are no matching rows on the right side.
- Let's consider an adjusted example:
    - Suppose we have a table called "Departments" with columns like DepartmentID and DepartmentName. We want to retrieve a list of all departments along with the names of their managers, if available.

    - If we use a left join with self join, it ensures that all departments are included in the result, regardless of whether they have a manager assigned or not. The left join ensures that all rows from the left side (departments) are included, even if there are no matching rows on the right side (managers).

- The query would look like this:

```
SELECT d.DepartmentName, m.FirstName AS ManagerFirstName, m.LastName AS ManagerLas
tName
FROM Departments d
LEFT JOIN Employees m ON d.DepartmentID = m.DepartmentID;
```

- In this case, a regular self join would only return departments that have a manager assigned to them. However, by using a left join with a self join, we

ensure that all departments are included in the result, even if they don't have a matching manager.

Therefore, the use of a left join with a self join becomes meaningful when you specifically want to include all rows from the left side, even if there are no matching rows on the right side. It provides a way to retrieve a complete set of data, including cases where the join condition is not met for some rows on the right side.

Don’t use `<>` not equal to in self join because if we consider a table with self join having `<>` it means A —> B & B —> A, is right but it is a duplicate, as A is related to B and B is related to a so not equal to will give this type of duplicate value in self join better use `>` and `<`