Lecture Notes – Self JOIN

Section 1: Lecture Summary

Self-join involves joining a table with itself to relate records within the same table, such as finding employees and their managers. The lecture demonstrates using the **Employees** table twice, aliasing one as E for employees and M for managers, with a **LEFT JOIN** on manager ID matching employee ID to retrieve employee ID, employee name, and manager name.

Section 2: Key Concepts and Explanations

Self-join treats the same table as two separate instances by using table aliases, allowing comparison of records within the table. The join condition links E.managerID = M.EmpID, where the left side (E) represents subordinates and the right side (M) represents managers. **LEFT JOIN** is preferred to include employees without managers (showing NULL for manager name). No specific "SELF JOIN" keyword exists; standard join types like **JOIN** or **LEFT JOIN** are used with aliases. Results show employee details with matched manager names based on the ID relationship.

Section 3: Example Code and Use Cases

```
SELECT E.EmpID, E.FirstName, E.LastName, M.FirstName AS ManagerFirstName,
M.LastName AS ManagerLastName
FROM Employees E
LEFT JOIN Employees M ON E.managerID = M.EmpID;
```

This query retrieves employee IDs, names, and manager names from the **Employees** table using self-join. It includes all employees (left side) and matches managers where managerID exists, returning NULL for manager names if no match.

Sample result structure (based on lecture example):

- EmpID 101, Raghav (no manager)

- EmpID 102, Tanvi (manager Raghav)

- EmpID 103, Neeraj (manager Tanvi)

- EmpID 104, Meera (manager Raghav)

Section 4: Key Takeaways

Self-join enables intra-table relationships using aliases and standard join syntax. Use **LEFT JOIN** for complete employee lists including those without managers. Commonly applied for hierarchical data like employee-manager pairs in the **Employees** table.