# Lecture Notes – More JOINS

Section 1: Lecture Summary

The lecture explains **left join**, **right join**, and **full join** using custom employees and departments tables without a foreign key relationship. **Left join** includes all rows from the left table (employees) and matching rows from the right table (departments), filling non-matches with null. **Right join** includes all rows from the right table and matching rows from the left table, filling non-matches with null. **Full join** includes all rows from both tables, duplicating matches and filling non-matches with null. Self join is mentioned as the topic for the next lecture.

Section 2: Key Concepts and Explanations

**Left join** takes the full left table (employees as E) and matches on the **ON** condition (E.DeptID = D.DeptID), adding department data where available or null otherwise. For example, employee DeptID 1 matches DeptID 1 (Admin), DeptID 2 matches DeptID 2 (IT), DeptID 3 matches multiple DeptID 3 (Finance), while null and 5 get nulls.

**Right join** takes the full right table (departments as D) and matches from the left table, so all department rows appear with employee data where matching or null otherwise; DeptID 4 gets nulls for employees. These are also called **left outer join** and **right outer join**, where "outer" means including non-matching rows.

**Full join** combines all rows from both tables: all employees and all departments, with matches duplicated (e.g., multiple DeptID 3) and non-matches filled with null (e.g., employees 6 and 7, DeptID 4). Joins work regardless of foreign key constraints, based on the **ON** condition. Table order matters: left side of JOIN determines left table.

Section 3: Example Code and Use Cases

Using companyDB schemas to demonstrate equivalent joins on **Employees** and **Departments** (matching on **DeptID**):

```
-- Left join: All Employees, matching Departments or null
SELECT * FROM Employees E
LEFT JOIN Departments D ON E.DeptID = D.DeptID;
```

All **Employees** rows appear; unmatched **DeptID** get null **DeptName** and **Location**.

```
-- Right join: All Departments, matching Employees or null
SELECT * FROM Employees E
RIGHT JOIN Departments D ON E.DeptID = D.DeptID;
```

All **Departments** rows appear; unmatched **DeptID** get null **EmpID**, **FirstName**, etc.

```
-- Full join equivalent (MySQL uses UNION of left and right)
SELECT E.EmpID, E.FirstName, E.LastName, E.JobTitle, E.DeptID, D.DeptName,
D.Location
FROM Employees E
LEFT JOIN Departments D ON E.DeptID = D.DeptID
UNION
SELECT E.EmpID, E.FirstName, E.LastName, E.JobTitle, E.DeptID, D.DeptName,
D.Location
FROM Employees E
RIGHT JOIN Departments D ON E.DeptID = D.DeptID;
```

All rows from both tables, matches duplicated, non-matches null-filled.

Section 4: Key Takeaways

- **Left join**: Full left table + matches from right (nulls for no match).

- **Right join**: Full right table + matches from left (nulls for no match).

- **Full join**: Full both tables (MySQL simulates via UNION; matches duplicate).

- Joins rely on **ON** condition, not requiring foreign keys.

- Table position (left/right of JOIN) determines which gets all rows.