Lecture Notes – ROW_NUMBER()

Section 1: Lecture Summary

The lecture introduces the **row_number()** window function for assigning sequential numbers to records in analytical queries using the companyDB Employees table. It demonstrates basic usage with SELECT *, then adds **PARTITION BY DeptID** for department-wise numbering, **ORDER BY HireDate** for hire date sequencing across all employees, and combines partitioning with ordering for department-specific seniority analysis based on hire dates.

Section 2: Key Concepts and Explanations

Window functions include ranking functions like **row_number()**, **rank()**, **dense_rank()**, and **ntile()**, distinct from aggregate functions such as AVG, COUNT, MAX, MIN. The **row_number()** function assigns unique sequential numbers (1, 2, 3...) to each row in the result set. The **OVER()** clause defines the window:

- Without partitioning or ordering, it numbers all rows sequentially.

- **PARTITION BY DeptID** restarts numbering within each department group.

- **ORDER BY HireDate** sorts rows by hire date before numbering, enabling identification of hiring order and employee seniority.

- Combining **PARTITION BY DeptID ORDER BY HireDate** partitions by department, then orders and numbers within each partition to analyze department-wise seniority.

Section 3: Example Code and Use Cases

```
-- Row numbers for all employees
SELECT *, ROW_NUMBER() OVER() AS num FROM Employees;
```

Numbers all 2020 employee rows sequentially from 1.

```
-- Department-wise row numbers
SELECT *, ROW_NUMBER() OVER(PARTITION BY DeptID) AS num FROM Employees;
```

Restarts numbering per DeptID (e.g., Dept 1: 1-4; Dept 2: 1-6; Dept 3: 1-5).

```
-- Row numbers ordered by HireDate (all employees)
SELECT *, ROW_NUMBER() OVER(ORDER BY HireDate) AS num FROM Employees;
```

Sorts by HireDate (e.g., 2016 first), assigns numbers 1-2020 to show organization-wide hiring position.

```
-- Department-wise row numbers ordered by HireDate
SELECT *, ROW_NUMBER() OVER(PARTITION BY DeptID ORDER BY HireDate) AS num
FROM Employees;
```

Partitions by DeptID, orders each by HireDate (e.g., Dept 1 sorted ascending HireDate with numbers 1-4), identifies seniors/juniors per department.

Section 4: Key Takeaways

ROW_NUMBER() OVER() sequentially numbers rows for analysis. Use **PARTITION BY** to group and restart numbers; **ORDER BY** to sequence within windows. Combine for partitioned, ordered numbering to analyze hiring order and count employees per department using companyDB Employees table.