Lecture Notes – IntroductionToWindowFunctions

Section 1: Lecture Summary

Window functions in SQL provide analytical results similar to GROUP BY or subqueries but shorten query length and return details for every row. They use aggregate functions, ranking functions, value access functions, and distribution functions with the OVER clause to perform calculations over a defined window of data.

Section 2: Key Concepts and Explanations

Window functions combine a function (like **AVG** or **SUM**) with OVER() to compute values across rows without collapsing results like GROUP BY. OVER() without parameters treats the entire result set as the window. Inside OVER(), use **PARTITION BY** to group data like GROUP BY, **ORDER BY** for running totals (cumulative sums), and **ROWS BETWEEN** for framing specific row ranges. Framing options include **UNBOUNDED PRECEDING** (from start), **n PRECEDING** (previous rows), **CURRENT ROW**, **n FOLLOWING** (next rows), or **UNBOUNDED FOLLOWING** (to end). Combine clauses for department-wise running totals sorted by salary.

Section 3: Example Code and Use Cases

List all employee details with company-wide average salary:

```
SELECT *, AVG(Salary) OVER() AS AvgSalary
FROM Employees;
```

Each row shows employee details plus the same average salary value.

Sum salaries partitioned by department:

```
SELECT *, SUM(Salary) OVER(PARTITION BY DeptID) AS DeptTotalSalary
FROM Employees;
```

Repeats department total for each employee in that department.

Running total of salaries across all employees:

```
SELECT *, SUM(Salary) OVER(ORDER BY Salary) AS RunningTotal
FROM Employees;
```

Cumulative sum increases with each row in salary order.

Department-wise running total:

```
SELECT *, SUM(Salary) OVER(PARTITION BY DeptID ORDER BY Salary) AS
DeptRunningTotal
FROM Employees;
```

Resets per department, accumulates within each.

Running total over previous row and current:

```
SELECT *, SUM(Salary) OVER(ORDER BY Salary ROWS BETWEEN 1 PRECEDING AND
CURRENT ROW) AS PrevPlusCurrent
FROM Employees;
```

Sums only the prior row's salary plus current for each.

Full running total using unbounded preceding:

```
SELECT *, SUM(Salary) OVER(ORDER BY Salary ROWS BETWEEN UNBOUNDED
PRECEDING AND CURRENT ROW) AS FullRunningTotal
FROM Employees;
```

Accumulates from first row to current in order.

Section 4: Key Takeaways

Window functions enable row-level analytics without losing detail. OVER() defines the window; PARTITION BY groups, ORDER BY enables running totals, ROWS BETWEEN limits frame. Use aggregates like SUM or AVG for comparisons like salary vs. average or department totals. Practice combining clauses for complex analysis.