Lecture Notes – COUNT( )

Section 1: Lecture Summary

The lecture covers using **aggregate window functions** count, max, and min for challenging queries. It demonstrates department salary insights showing employee details with department-wise employee count and average salary using a natural join between Employees and Departments tables. Another example shows product details with category-wise product count from a products table. Results display aggregates alongside individual row details for analysis, with notes on ordering via ORDER BY.

Section 2: Key Concepts and Explanations

Aggregate window functions like **count**, **avg**, **max**, and **min** compute values over partitions without collapsing rows. Window functions use **OVER (PARTITION BY column)** to group calculations, such as counting employees or averaging salaries per department. **Natural join** combines tables on common columns like DeptID. Individual details (e.g., employee ID, salary) appear with aggregates for each row in the partition, enabling per-row comparisons like salary vs. department average. Results may not be sorted by partition without **ORDER BY**, as join order affects output.

Section 3: Example Code and Use Cases

Department salary insights query using companyDB:

```
SELECT
    d.DeptName,
    e.EmpID,
    e.Salary,
    COUNT(e.EmpID) OVER (PARTITION BY d.DeptName) AS TotalEmployees,
    AVG(e.Salary) OVER (PARTITION BY d.DeptName) AS AvgSalary
FROM Employees e
NATURAL JOIN Departments d;
```

This returns DeptName, EmpID, Salary for each employee, plus total employees and average salary in their department (e.g., 5 employees in Finance with their avg salary repeated per row).

Product count per category query (adapted to companyDB Projects table for department-wise project count):

```
SELECT
    ProjectID,
    ProjectName,
    DeptID,
    COUNT(ProjectID) OVER (PARTITION BY DeptID) AS TotalProjects
FROM Projects
ORDER BY DeptID, ProjectID;
```

This returns ProjectID, ProjectName, DeptID for each project, plus total projects in that DeptID (e.g., repeated count for all projects in same department).

Section 4: Key Takeaways

Window functions with **OVER (PARTITION BY)** provide aggregates per group alongside row details. Use joins like **natural join** for related data (e.g., DeptName via DeptID). Add **ORDER BY** for consistent sorting. Enables analysis like comparing individual salary to department average or project count per department. Next covers **max** for highest values per group.