Lecture Notes – Sub Queries-SELECT Clause

Section 1: Lecture Summary

Subqueries in the **select clause** allow embedding a query as a column value in the outer query, where the subquery must return a single scalar value for each row of the outer query. These are **nested queries** (correlated subqueries) because the subquery references columns from the outer query using aliases, executing the subquery for each outer row. Examples demonstrate showing departments with employee counts, customers with total spend, and categories with product counts, all using **where** conditions to match IDs between outer and inner tables.

Section 2: Key Concepts and Explanations

Subqueries in the **select clause** follow the syntax: SELECT column1, column2, (SELECT ... FROM inner_table WHERE inner_id = outer_alias.outer_id) AS alias FROM outer_table outer_alias. The subquery returns one value per outer row, computed by filtering inner table rows to match the current outer row's ID. Execution order starts with the outer query, running the subquery repeatedly for each outer row due to the correlation via aliases. This approach avoids **GROUP BY** or **JOIN** for simple aggregations like counts or sums tied to outer rows. All examples are correlated, using **WHERE** to link tables by ID for department-wise employee counts, customer-wise order totals, or category-wise product counts.

Section 3: Example Code and Use Cases

Using eCommerceDB schemas:

Show each **category** with number of **products**:

```
SELECT CategoryID, CategoryName,
       (SELECT COUNT(*) FROM Products P WHERE P.CategoryID = C.CategoryID)
AS NumProducts
FROM Categories C;
```

This returns category details with product count per category, correlating via CategoryID.

Show each **customer** with their **total amount spent**:

```
SELECT CustomerID, FirstName, LastName,
       (SELECT SUM(TotalAmount) FROM Orders O WHERE O.CustomerID =
C.CustomerID) AS TotalSpend
FROM Customers C;
```

This returns customer details with summed TotalAmount from their orders, correlating via CustomerID; returns NULL for customers with no orders.

Section 4: Key Takeaways

Subqueries in **select clause** must return scalar values and work as nested (correlated) queries via outer table aliases in **WHERE**. Use for per-row aggregations like counts or sums without **JOIN** or **GROUP BY**. Filter inner queries by matching IDs to outer row values for accurate row-specific results. Practice adapting to tables like Categories-Products or Customers-Orders.