Lecture Notes – Order Of Execution

Section 1: Lecture Summary

The lecture explains the **order of execution** for SQL clauses in a query. It starts with **FROM** to identify the table, followed by **WHERE** for row filtering, **GROUP BY** for grouping if present, **HAVING** for filtering groups, **SELECT** to choose columns, and finally **ORDER BY** for sorting.

Section 2: Key Concepts and Explanations

FROM executes first to retrieve the entire table. **WHERE** then filters rows from that data. If **GROUP BY** is used, it groups the filtered rows next, and **HAVING** filters those groups. **SELECT** follows to specify columns from the remaining data. **ORDER BY** executes last to sort the selected results. Not all clauses are required in every query, but **FROM** is mandatory when retrieving data from a table, and understanding this sequence helps predict query results.

Section 3: Example Code and Use Cases

Using eCommerceDB, consider this query with all clauses:

```
SELECT CategoryName, COUNT(*) as ProductCount, AVG(Price) as AvgPrice
FROM Products p
JOIN Categories c ON p.CategoryID = c.CategoryID
WHERE Price > 50
GROUP BY p.CategoryID, c.CategoryName
HAVING COUNT(*) > 2
ORDER BY AvgPrice DESC;
```

Execution order: **FROM/JOIN** retrieves Products and Categories; **WHERE** filters Price > 50; **GROUP BY** groups by CategoryID and CategoryName; **HAVING** keeps groups with more than 2 products; **SELECT** picks columns with aggregates; **ORDER BY** sorts by average price descending.

Section 4: Key Takeaways

SELECT executes late after filtering and grouping. Knowing the order (**FROM** → **WHERE** → **GROUP BY** → **HAVING** → **SELECT** → **ORDER BY**) enables writing effective queries and visualizing backend processing.