Lecture Notes – MIn() & Max()

Section 1: Lecture Summary

The lecture covers the practical application of **MIN()** and **MAX()** aggregate functions within GROUP BY and HAVING clauses. Three progressive queries demonstrate how to find minimum and maximum values across different grouping levels: customer-wise order amounts, category-wise product prices with conditional filtering, and order-wise item prices with subtotal thresholds. The final discussion emphasizes query optimization by identifying unnecessary table joins.

Section 2: Key Concepts and Explanations

Aggregate Functions with GROUP BY

The **MIN()** and **MAX()** functions calculate the smallest and largest values within grouped datasets. When combined with GROUP BY, these functions operate on each group independently rather than the entire table, enabling comparative analysis across categories.

WHERE vs HAVING Clauses

A critical distinction exists in filter placement:

- **WHERE clause** filters individual rows before grouping occurs and cannot reference aggregate functions

- **HAVING clause** filters groups after aggregation and must be used when conditions involve aggregate functions like COUNT(), SUM(), or when checking group-level thresholds

Query Optimization

Unnecessary table joins increase computational overhead. When all required columns exist in a single table, joining additional tables wastes processing time. Identifying which table contains all necessary data prevents redundant operations.

Section 3: Example Code and Use Cases

Query 1: Minimum and Maximum Order Value per Customer

```
SELECT
    CustomerID,
    MIN(TotalAmount) AS MinOrderAmount,
    MAX(TotalAmount) AS MaxOrderAmount
FROM Orders
GROUP BY CustomerID;
```

This query groups all orders by CustomerID and calculates the smallest and largest order amounts each customer has placed. Every customer with multiple orders will show different minimum and maximum values.

Query 2: Maximum Product Price per Category with Filtering

```
SELECT
    CategoryID,
    MAX(Price) AS MaxProductPrice
FROM Products
GROUP BY CategoryID
HAVING COUNT(*) > 2;
```

This query finds the maximum product price in each category but only displays categories containing more than two products. The COUNT(*) > 2 condition must use HAVING because it references an aggregate function and filters groups, not individual rows.

Query 3: Maximum and Minimum Unit Prices for Orders with High Subtotals

```
SELECT
    OrderID,
    SUM(Subtotal) AS OrderTotal,
    MAX(UnitPrice) AS HighestUnitPrice,
    MIN(UnitPrice) AS LowestUnitPrice
FROM OrderItems
GROUP BY OrderID
HAVING SUM(Subtotal) > 40000;
```

This query operates exclusively on the OrderItems table, grouping by OrderID to find the highest and lowest unit prices within each order while filtering for orders with subtotals exceeding 40,000. No join is necessary since OrderItems contains all required data (OrderID, Subtotal, and UnitPrice).

Section 4: Key Takeaways

- **MIN() and MAX()** functions within GROUP BY clauses enable comparative analysis across grouped data

- **HAVING clauses must filter aggregate conditions**, while WHERE clauses filter row-level data

- **Query optimization requires examining which table contains all necessary columns** before adding joins, as unnecessary joins degrade performance

- **Single-table queries are preferable to multi-table joins** when all required data exists in one table