Lecture Notes – MIN()

Section 1: Lecture Summary

The lecture demonstrates using the **MIN** aggregate function as a window function to analyze customer spending patterns in the eCommerceDB. It covers two main queries: one showing customer orders with their average and minimum **TotalAmount** per customer from the **Orders** table, and another displaying payments with the minimum payment amount per customer by joining **Payments**, **Orders**, and **Customers** tables.

Section 2: Key Concepts and Explanations

MIN as a window function calculates the minimum value within a specified partition for each row, allowing comparison of individual records against customer-level aggregates without grouping.

In the first query, **AVG(TotalAmount)** and **MIN(TotalAmount)** are computed over partitions by **CustomerID** from **Orders**, showing per-order details alongside customer-specific averages and minimums.

The second query uses **MIN(Amount)** over partitions by **CustomerID** from joined **Payments P**, **Orders O**, and **Customers C** tables, revealing each payment alongside the customer's lowest payment amount.

Window functions require **OVER (PARTITION BY CustomerID)** to scope aggregates customer-wise, ensuring results include all rows with added computed columns.

Section 3: Example Code and Use Cases

First query on **eCommerceDB** for customer average and minimum spending:

```
SELECT
    CustomerID,
    OrderID,
    TotalAmount,
    AVG(TotalAmount) OVER (PARTITION BY CustomerID) AS AvgSpending,
    MIN(TotalAmount) OVER (PARTITION BY CustomerID) AS MinSpending
FROM Orders;
```

This returns all orders with **CustomerID**, **OrderID**, **TotalAmount**, and the same **AvgSpending** and **MinSpending** repeated for each customer's orders.

Second query on **eCommerceDB** for payments with minimum per customer:

```
SELECT
    C.CustomerID,
    C.FirstName,
    P.PaymentID,
    P.Amount,
    MIN(P.Amount) OVER (PARTITION BY C.CustomerID) AS MinPaymentByCustomer
FROM Payments P
NATURAL JOIN Orders O
NATURAL JOIN Customers C;
```

This shows customer details, payment details, and the minimum **Amount** for each customer across their payments.

Section 4: Key Takeaways

MIN window functions enable row-level analysis with partition-level aggregates, using **OVER (PARTITION BY CustomerID)** for customer-specific minimums.

Combine with **AVG** for spending insights directly in result sets.

Joins across **Payments**, **Orders**, and **Customers** link transactions to customer names while applying windows on **CustomerID**.

Practice by modifying partitions or aggregates to observe row-level repetition of computed values.