Lecture Notes – QueriesOnAVG( )

Section 1: Lecture Summary

The lecture demonstrates using the **average** aggregate function with **window functions** for two business queries in the eCommerceDB. First query calculates **average inventory value per category** using Products and Categories tables to show total inventory value per product and category average for warehouse planning. Second query computes **customer profitability score** from Orders table by grouping customer total spending, comparing to overall average, and deriving a score as total divided by average.

Section 2: Key Concepts and Explanations

Average as window function: Computes average of an expression (like total inventory value) partitioned by a column (e.g., **CategoryName**) and repeats the value for every row in that partition, enabling per-row comparison to group average.

Inventory value: Calculated as **Price * Stock** for each product; average computed category-wise for reordering and sales focus decisions.

Customer profitability score: Total spending per customer (**SUM(TotalAmount)** grouped by **CustomerID**) divided by average of those totals across all customers; higher score indicates more profitable customers. Uses aggregate inside aggregate with **GROUP BY**; no partitioning needed for overall average after grouping.

ROUND function: Limits decimal places (e.g., ROUND(expression, 2)) for score readability.

Query structure differences: Join tables for multi-table average with window; single table with GROUP BY for customer totals and nested aggregates.

Section 3: Example Code and Use Cases

Query 1: Average inventory value per category (warehouse planning)

```
SELECT
    c.CategoryName,
    p.ProductName,
    (p.Price * p.Stock) AS inventory_value,
    AVG(p.Price * p.Stock) OVER (PARTITION BY c.CategoryName) AS
average_inventory_value_per_category
FROM Products p
NATURAL JOIN Categories c;
```

Shows **CategoryName**, **ProductName**, per-product inventory (**Price * Stock**), and repeated category average (e.g., Accessories average 3,35,000 across its products) for deciding reorder focus.

Query 2: Customer profitability score

```
SELECT
    CustomerID,
    SUM(TotalAmount) AS customer_total_spend,
    AVG(SUM(TotalAmount)) OVER () AS average_platform_spend,
    ROUND(SUM(TotalAmount) / AVG(SUM(TotalAmount)) OVER (), 2) AS
profitability_score
FROM Orders
GROUP BY CustomerID;
```

Groups by **CustomerID** for total spend per customer, computes average of those totals (same value repeated), and derives score (e.g., 1.91 highest) to rank customer value.

Section 4: Key Takeaways

Use **AVG** with **OVER (PARTITION BY)** for category-wise averages in joined tables. Combine **GROUP BY** with nested aggregates and **OVER()** for overall averages post-grouping. Multiply **Price * Stock** for inventory value. Divide customer total by platform average for **profitability score**; apply **ROUND** for precision. Practice rewriting both queries independently.