Lecture Notes – SUM() & AVG()

Section 1: Lecture Summary

The lecture demonstrates using **SUM()** and **AVG()** aggregate functions with **GROUP BY** and **HAVING** clauses on the eCommerceDB. Examples include calculating total order value by customer for delivered orders, average rating for customers with more than one review, and categories where sum of sales exceeds 100K with average product price above 20K.

Section 2: Key Concepts and Explanations

Aggregate functions **SUM()** and **AVG()** compute totals and averages on grouped data. **GROUP BY** groups rows by a column like **CustomerID** or **CategoryName** before applying aggregates. **WHERE** filters rows before grouping, such as **OrderStatus = 'delivered'**. **HAVING** filters groups after aggregation, such as **COUNT(ReviewID) > 1** or **SUM(Subtotal) > 100000**. Conditions on raw data use **WHERE**; conditions on aggregates use **HAVING**. Multi-table queries require joins via primary-foreign key relationships, such as **Categories** to **Products** on **CategoryID** and **Products** to **OrderItems** on **ProductID**.

Section 3: Example Code and Use Cases

Total order value by customer for delivered orders:

```
SELECT CustomerID, SUM(TotalAmount) AS total_spent
FROM Orders
WHERE OrderStatus = 'delivered'
GROUP BY CustomerID;
```

Shows customers 1, 2, 3, 5, 7, 8, 9, 10 with their total spends; excludes customer 4 (shipped/pending) and 6.

Customers with more than one review and average rating:

```
SELECT CustomerID, COUNT(ReviewID) AS reviews_given, AVG(Rating) AS
average_rating
FROM Reviews
GROUP BY CustomerID
HAVING COUNT(ReviewID) > 1;
```

Returns customers 1 and 2 with 2 reviews each, averages 4.5 and 4.

Categories with sum of sales > 100K and average product price > 20K:

```
SELECT c.CategoryName, SUM(oi.Subtotal) AS total_sales, AVG(p.Price) AS
avg_price
FROM Categories c
NATURAL JOIN Products p
NATURAL JOIN OrderItems oi
GROUP BY c.CategoryName
HAVING SUM(oi.Subtotal) > 100000 AND AVG(p.Price) > 20000;
```

Returns Electronics with total_sales 732000 and avg_price 36105.

Section 4: Key Takeaways

Use **SUM()** for totals and **AVG()** for averages with **GROUP BY** for customer-wise or category-wise summaries. Filter pre-group with **WHERE**, post-group with **HAVING**. Join tables on foreign keys for multi-table aggregates. Understand data distribution to interpret results accurately.