Lecture Notes – MoreQueriesOnRANK()_DENSERANK()

Section 1: Lecture Summary

The lecture demonstrates advanced queries using **RANK()** and **DENSE_RANK()** window functions on the eCommerceDB to rank customers and cities by spending metrics. It starts with ranking customers by total spending using **Customers** and **Orders** tables, then adapts the query to rank cities by total revenue, and finally ranks customers by average order value using only the **Orders** table.

Section 2: Key Concepts and Explanations

Queries require joining **Customers** and **Orders** tables on **CustomerID** for customer-level summaries, with **GROUP BY** on **CustomerID** (or concatenated name) and **SUM(TotalAmount)** for total spending. For city ranking, **GROUP BY City** replaces customer grouping, using **DENSE_RANK()** over **SUM(TotalAmount) DESC**. Average order value uses **SUM(TotalAmount) / COUNT(*)** per customer with **DENSE_RANK()** over that metric descending. Window functions like **RANK() OVER (ORDER BY aggregate DESC)** assign ranks after grouping and aggregation, handling ties differently (**DENSE_RANK()** skips no ranks for ties). Single-table queries simplify when names are omitted, as **Orders** provides **CustomerID** directly.

Section 3: Example Code and Use Cases

```
-- Rank customers by total spending (full name included)
SELECT
    C.CustomerID,
    CONCAT(C.FirstName, ' ', C.LastName) AS full_name,
    SUM(O.TotalAmount) AS total_spend,
    RANK() OVER (ORDER BY SUM(O.TotalAmount) DESC) AS spend_rank
FROM Customers C
JOIN Orders O ON C.CustomerID = O.CustomerID
GROUP BY C.CustomerID, full_name;
```

```
-- Rank cities by total revenue
SELECT
    C.City AS city,
    SUM(O.TotalAmount) AS city_revenue,
    DENSE_RANK() OVER (ORDER BY SUM(O.TotalAmount) DESC) AS revenue_rank
FROM Customers C
```

```
JOIN Orders O ON C.CustomerID = O.CustomerID
GROUP BY C.City;
```

```
-- Rank customers by average order value (Orders table only)
SELECT
    CustomerID,
    SUM(TotalAmount) / COUNT(*) AS avg_order_value,
    DENSE_RANK() OVER (ORDER BY SUM(TotalAmount) / COUNT(*) DESC) AS
avg_rank
FROM Orders
GROUP BY CustomerID;
```

Section 4: Key Takeaways

Adapt base query structure (**JOIN**, **GROUP BY**, aggregate, **RANK OVER (ORDER BY aggregate DESC)**) for variations like total vs. average or customer vs. city. Use **CONCAT** for names in **GROUP BY**; omit **Customers** table when only **CustomerID** suffices. **DENSE_RANK()** handles ties without gaps, unlike **RANK()**. Practice modifications yield different rankings, e.g., total spend favors high-volume customers while average favors per-order value.