Lecture Notes – Self Join Queries

Section 1: Lecture Summary

Self-join involves joining a table with itself using different aliases to compare rows within the same table. The lecture demonstrates writing queries to list products with matching prices, employees with the same salary, and customers in the same city, using aliases like P1/P2, E1/E2, and ensuring comparisons only go forward with a **WHERE** condition on ID.

Section 2: Key Concepts and Explanations

Self-join treats the same table as two separate instances with distinct aliases (e.g., products P1, products P2) to enable row-to-row comparisons. The **JOIN** clause uses **ON** to match conditions like equal prices (P1.Price = P2.Price), while **WHERE P1.ProductID < P2.ProductID** prevents self-matching and duplicate pairs by checking only subsequent rows. This produces pairs like matching products or customers without reverse duplicates.

Section 3: Example Code and Use Cases

Query to list products with the same price using **eCommerceDB**:

```
SELECT P1.ProductName AS ProductA,
       P2.ProductName AS ProductB,
       P1.Price
FROM Products P1
JOIN Products P2
  ON P1.Price = P2.Price
WHERE P1.ProductID < P2.ProductID;
```

This returns pairs like wireless keyboard with badminton racket at price 1500.

Equivalent for customers in the same city using **eCommerceDB** **Customers** table:

```
SELECT C1.FirstName AS CustomerA,
       C1.LastName AS CustomerA_Last,
       C2.FirstName AS CustomerB,
       C2.LastName AS CustomerB_Last,
       C1.City
FROM Customers C1
JOIN Customers C2
  ON C1.City = C2.City
WHERE C1.CustomerID < C2.CustomerID;
```

This pairs customers like those in Mumbai (e.g., Rajesh with Manish).

Section 4: Key Takeaways

Use table aliases for self-joins to compare rows within one table. Match on shared attributes with **ON**, and use **WHERE** with ID comparison (<) to avoid self-pairs and duplicates. Practice on **Products**, **Customers** for price/city matches.