Lecture Notes – Date & Time-Difference

Section 1: Lecture Summary

The lecture covers **DATEDIFF** and **TIMESTAMPDIFF** functions for calculating date-time differences in MySQL. **DATEDIFF** returns the difference between two dates in days only. **TIMESTAMPDIFF** allows specifying units like day, month, year, hour, minute, or second for more flexible comparisons. Examples demonstrate usage with dates like '2025-01-01' and '2024-01-01', accounting for leap years (366 days). A practical query analyzes customer engagement by measuring days between **OrderDate** and **ReviewDate** in the eCommerceDB, joining Reviews, Customers, Products, and Orders tables. The initial query produces incorrect results due to multiple orders per customer; it needs OrderItems for accurate matching of orders to reviewed products.

Section 2: Key Concepts and Explanations

DATEDIFF(date1, date2) computes days between two dates, with date1 minus date2; order matters as it can yield positive or negative values. **TIMESTAMPDIFF(unit, date1, date2)** calculates difference in specified unit (e.g., DAY, MONTH, YEAR, HOUR, MINUTE, SECOND), handling both dates and timestamps. Parameter order is critical: earlier date first for positive gaps. In eCommerceDB analysis, join Reviews on CustomerID and ProductID, Orders on CustomerID and OrderDate, Products on ProductID and CategoryID; incomplete joins cause mismatches from multiple customer orders. Customer engagement measures speed of post-order reviews using ReviewDate - OrderDate.

Section 3: Example Code and Use Cases

Basic function demos:

```
SELECT DATEDIFF('2025-01-01', '2024-01-01');  -- Returns 366 (leap year)
SELECT TIMESTAMPDIFF(DAY, '2025-01-01', '2026-01-01');  -- Returns 365
```

Customer engagement query (initial version, shows negative gaps due to multiple orders):

```
SELECT
    c.FirstName,
    p.ProductName,
    o.OrderDate,
    r.ReviewDate,
    TIMESTAMPDIFF(DAY, o.OrderDate, r.ReviewDate) AS days_after_order
```

```
FROM Reviews r
NATURAL JOIN Customers c
NATURAL JOIN Products p
NATURAL JOIN Orders o
ORDER BY days_after_order;
```

Improved query requires OrderItems for precise order-product matching:

```
SELECT
    c.FirstName,
    p.ProductName,
    o.OrderDate,
    r.ReviewDate,
    TIMESTAMPDIFF(DAY, o.OrderDate, r.ReviewDate) AS days_after_order
FROM Reviews r
JOIN Customers c ON r.CustomerID = c.CustomerID
JOIN Products p ON r.ProductID = p.ProductID
JOIN OrderItems oi ON p.ProductID = oi.ProductID
JOIN Orders o ON oi.OrderID = o.OrderID AND o.CustomerID = c.CustomerID
ORDER BY days_after_order;
```

Section 4: Key Takeaways

Use **DATEDIFF** for simple day differences; prefer **TIMESTAMPDIFF** for custom units. Always verify join conditions to avoid cross-matching (e.g., include **OrderItems** for order-product-review links). Parameter order determines positive/negative results. Functions support leap year calculations accurately. Analyze engagement via time gaps between business events like orders and reviews.