Lecture Notes – Date & Time-UTC

Section 1: Lecture Summary

UTC date and time functions provide the current date and time based on **Coordinated Universal Time** (UTC), which is the standard time at the Prime Meridian (time zone zero) in Greenwich, England. The world is divided into time zones, with India at UTC+5:30, and using UTC avoids conflicts in global applications like eCommerce by recording consistent timestamps regardless of user location. Functions like **UTC_TIME()** and **CURRENT_TIME()** retrieve UTC time versus local time, and data such as order dates should be inserted using UTC for standardization, with local adjustments handled in the UI.

Section 2: Key Concepts and Explanations

UTC (Coordinated Universal Time) serves as a global standard based on the Prime Meridian, dividing the world into approximately 24 time zones offset by hours (e.g., India is 5 hours 30 minutes ahead). In databases for global eCommerce, recording dates and times in UTC prevents discrepancies from varying user time zones, server locations, or local machine times. Key functions include **UTC_TIME()** for UTC time, **CURRENT_TIME()** for local time, **UTC_DATE()** for UTC date, and **UTC_TIMESTAMP()** for both. When inserting records like orders, use UTC values (e.g., UTC_DATE() for OrderDate); retrieval shows UTC, but display can convert to user local time by adding their time zone offset outside SQL.

Section 3: Example Code and Use Cases

```
USE eCommerceDB;


-- Compare local time vs UTC time

SELECT CURRENT_TIME() AS local_time, UTC_TIME() AS utc_time;


-- Insert order using UTC date

INSERT INTO Orders (OrderID, CustomerID, OrderDate, TotalAmount, OrderStatus)

VALUES (1001, 500, UTC_DATE(), 299.99, 'Pending');
```

-- Retrieve orders with UTC timestamps (e.g., for global reporting)

SELECT OrderID, CustomerID, OrderDate, TotalAmount

FROM Orders

WHERE OrderDate >= UTC_DATE() - INTERVAL 7 DAY;

Section 4: Key Takeaways

Record all timestamps in databases using **UTC** functions like **UTC_TIME()**, **UTC_DATE()**, and **UTC_TIMESTAMP()** for global consistency. Use UTC in INSERT statements for tables like **Orders** to standardize data across time zones. Local time display is managed in application UI by applying user time zone offsets, not in SQL. This approach eliminates time zone conflicts in international eCommerce applications.