Lecture Notes – ParameterizedStoredProcedures

Section 1: Lecture Summary

Parameterized stored procedures accept parameters for flexible execution with different inputs. The lecture covers syntax for **IN**, **OUT**, and **INOUT** parameters, focusing on **IN** parameters with an example retrieving customer details by **CustomerID** from the **Customers** table in eCommerceDB. Multiple parameters are possible, with **OUT** and **INOUT** examples deferred to later lectures.

Section 2: Key Concepts and Explanations

Stored procedures use parameters prefixed by **IN**, **OUT**, or **INOUT** followed by parameter name and data type. **IN** parameters receive input values directly when calling the procedure via **CALL procedure_name(value)**. **OUT** parameters return results to a user-defined variable prefixed with **@variable**, declared and used in **SELECT @variable**. **INOUT** parameters pass input via **SET @variable = value** and receive output similarly. Delimiter must be set to **//** before **CREATE PROCEDURE**. Same procedure reuses with varying parameter values for different results.

Section 3: Example Code and Use Cases

```
DELIMITER //
CREATE PROCEDURE GetCustomerById(IN custID INT)
BEGIN
    SELECT * FROM Customers WHERE CustomerID = custID;
END //
DELIMITER ;
```

Execution:

```
CALL GetCustomerById(1);
```

Returns all columns (**CustomerID**, **FirstName**, **LastName**, **Gender**, **Email**, **City**, **JoinDate**) for **CustomerID = 1**.

```
CALL GetCustomerById(15);
```

Returns details for **CustomerID = 15**. Reusable for any **CustomerID** without rewriting SQL.

Section 4: Key Takeaways

Stored procedures with **IN** parameters enable dynamic queries using single or multiple inputs. **CALL** passes values directly for **IN**; **@variables** required for **OUT** and **INOUT**. Avoid embedding SQL in applications by calling procedures instead.