## Slide 1: Title Slide

**Advanced MySQL Concepts**

Procedures, Functions, Triggers, and Cursors

With Practical Examples and Schema Design

**Author:** Anantha Krishnan PTA

**Institution:** ITvedant Institute

## Slide 2: Introduction

**What we'll cover:**

- **Definitions** of Procedures, Functions, Triggers, and Cursors
- **Practical Examples** with SQL code
- **Database Schema Design**
- **DDL, DML, and DQL Queries**

Why these concepts?

- Automate repetitive tasks
- Ensure data integrity
- Simplify complex operations

## Slide 3: Database Design

### Schema: Employee Management System

**Tables:**

1. **Departments**: Department details
   - Columns: `dept_id`, `dept_name`
2. **Employees**: Employee details
   - Columns: `emp_id`, `emp_name`, `dept_id`
3. **Salaries**: Salary history
   - Columns: `salary_id`, `emp_id`, `salary`, `date`

**Relationships:**

- **Employees** belong to **Departments**
- **Salaries** are tied to **Employees**

## Definition:

Precompiled SQL statements used to encapsulate logic.

## Syntax:

```
DELIMITER //                              --Changes the delimiter of sql to '//' instead of ';'
   CREATE PROCEDURE proc_name(parameters) --Declares the procedure
      BEGIN                               --Begins procedure definition
          SQL statements;
      END//                               --End of procedure definition
DELIMITER ;                               --Changes the delimited back to ';'
```

## Example: Calculate average salary by department

```
DELIMITER //
CREATE PROCEDURE GetAvgSalaryByDept()
BEGIN
    SELECT dept_id, AVG(salary) AS avg_salary
    FROM Salaries
    GROUP BY dept_id;
END //
```

## Definition:

Reusable SQL routines that return a single value.

## Syntax:

```
CREATE FUNCTION func_name(parameters)
RETURNS data_type
BEGIN
    RETURN value;
END;
```

## Example: Get total salary of an employee

```
DELIMITER //
CREATE FUNCTION GetTotalSalary(emp_id INT)
RETURNS DECIMAL(10,2)
BEGIN
    DECLARE total_salary DECIMAL(10,2);
    SELECT SUM(salary) INTO total_salary
    FROM Salaries
    WHERE employee_id = emp_id;
```

## Syntax:

```
CREATE TRIGGER trigger_name
AFTER | BEFORE [INSERT | UPDATE | DELETE]
ON table_name
FOR EACH ROW
BEGIN
    SQL statements;
END;
```

## Example: Audit salary updates

```
-- Create AuditLog Table
CREATE TABLE AuditLog (
    audit_id INT PRIMARY KEY AUTO_INCREMENT,
    employee_id INT NOT NULL,
    old_salary DECIMAL(10, 2),
    new_salary DECIMAL(10, 2),
    change_date DATETIME DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (emp_id) REFERENCES Employees(emp_id)
);
```

```
DELIMITER //
```

Allow row-by-row processing of query results.

## Syntax:

```
DECLARE cursor_name CURSOR FOR query;
OPEN cursor_name;
FETCH cursor_name INTO variables;
CLOSE cursor_name;
```

## Example: Process employee bonuses

```
DELIMITER //
CREATE PROCEDURE ProcessBonuses()
BEGIN
    DECLARE emp_id INT;
    DECLARE done INT DEFAULT FALSE;
    DECLARE bonus_cursor CURSOR FOR SELECT employee_id FROM Employees;
    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

    OPEN bonus_cursor;
    read_loop: LOOP
        FETCH bonus_cursor INTO emp_id;
        IF done THEN LEAVE read_loop;
```

## Slide 8: DDL, DML, and DQL Queries

### DDL (Schema Creation):

```
CREATE TABLE Employees (
    emp_id INT PRIMARY KEY,
    emp_name VARCHAR(100),
    dept_id INT
);
```

### DML (Inserting Data):

```
INSERT INTO Salaries (emp_id, salary, date)
VALUES (1, 50000, '2024-11-01');
```

### DQL (Retrieving Data):

```
SELECT * FROM Salaries WHERE emp_id = 1;
```

## Slide 9: Summary

- **Procedures:** Encapsulate SQL logic for reuse.
- **Functions:** Provide reusable calculations.
- **Triggers:** Automate responses to database events.
- **Cursors:** Enable row-by-row processing.

Key takeaway: Use these tools to optimize and maintain complex databases effectively.

## Slide 10: References

- MySQL Official Documentation
- ITvedant Tutorials
- Practical SQL Guides