

# Assignment - 7

## Database Management Systems (DBMS)

**Name:** Kamithkar Vinod

**Course:** PG DAC August 2025

**PRN:** 250850320040

**Form No:** 250500480

**Date:** 30-10-2025

### Problem 1: Exception Handling

**Task:** Create a stored procedure `add_department` to insert a department. If the department name already exists, handle error 1062 using an `EXIT` handler and display the message "Department already exists".

**Code:** —

```
drop procedure if exists add_department;

delimiter //
create procedure add_department(in dname varchar(50))
begin
    declare exit handler for 1062
    begin
        select 'Department already exists' as message;
    end;

    insert into departments(dept_name) values (dname);
    select 'Department added successfully' as message;
end //
delimiter ;

call add_department('IT');
```

**Output:** —

```
mysql> call add_department('IT');
ERROR 1062 (23000): Duplicate entry 'IT' for key 'departments.dept_name'
mysql> call add_department('IT');
+-----+
| message                               |
+-----+
| Department already exists             |
+-----+
```

## Problem 2: Exception Handling

**Task:** Create a stored procedure `add_employee` to insert an employee. Handle duplicate employee names using an EXIT handler for error 1062.

**Code:** —

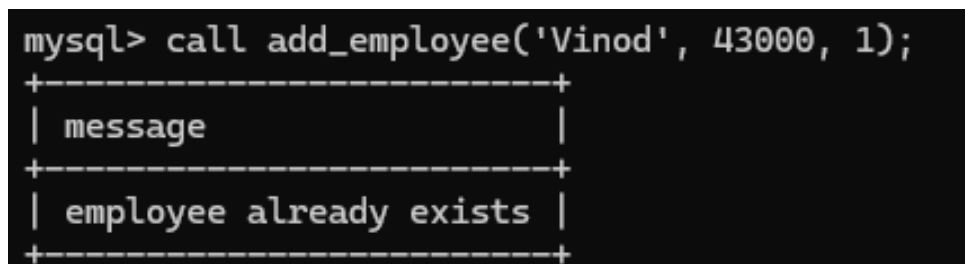
```
drop procedure if exists add_employee;

delimiter //
create procedure add_employee(in name varchar(50), in sal decimal
(10, 2), in d_id int)
begin
    declare exit handler for 1062
    begin
        select 'employee already exists' as message;
    end;
    insert into employees(emp_name, salary, dept_id)
    value (name, sal, d_id);
    select 'employee added successfully' as message;
end //

delimiter ;

call add_employee('Vinod', 43000, 1);
```

**Output:** —



```
mysql> call add_employee('Vinod', 43000, 1);
+-----+
| message                               |
+-----+
| employee already exists              |
+-----+
```

## Problem 3: Exception Handling

**Task:** Create a stored procedure `bulk_insert_employees` to insert 5 employees in a loop using a CONTINUE handler to skip duplicate names.

**Code:** —

```
drop procedure if exists bulk_insert_employees;

delimiter //
create procedure bulk_insert_employees()
begin
    declare i int default 1;
    declare continue handler for 1062
    begin
```

```
        select concat('Duplicate record found at ', i) as
            warning;
    end;

    while i <= 5 do
        insert into employees(emp_name, salary, dept_id)
        values (concat('Emp', i), 3000*i, 1);
        set i = i + 1;
    end while;
end //
delimiter ;

call bulk_insert_employees();
```

Output: —

```
+-----+
| warning |
+-----+
| Duplicate record found at 4 |
+-----+
1 row in set (0.02 sec)

+-----+
| warning |
+-----+
| Duplicate record found at 5 |
+-----+
```

## Problem 4: Exception Handling

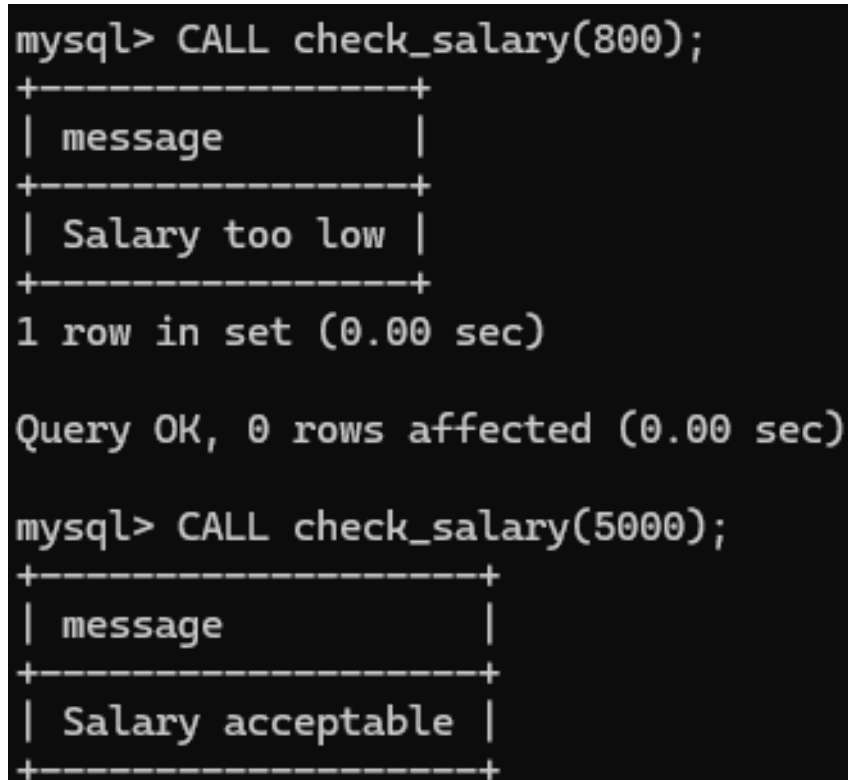
**Task:** Create a stored procedure `check_salary` that checks if a given salary is less than 1000. Use `SIGNAL` to raise a custom error and an `EXIT` handler to catch it.

Code: —

```
DELIMITER //
CREATE PROCEDURE check_salary(IN sal DECIMAL(10,2))
BEGIN
    DECLARE low_salary CONDITION FOR SQLSTATE '45000';
    DECLARE EXIT HANDLER FOR low_salary
    BEGIN
        SELECT 'Salary too low' AS message;
    
```

```
END;  
  
IF sal < 1000 THEN  
    SIGNAL low_salary;  
ELSE  
    SELECT 'Salary acceptable' AS message;  
END IF;  
END//  
DELIMITER ;  
  
CALL check_salary(800);  
CALL check_salary(5000);
```

Output: —



```
mysql> CALL check_salary(800);  
+-----+  
| message |  
+-----+  
| Salary too low |  
+-----+  
1 row in set (0.00 sec)  
  
Query OK, 0 rows affected (0.00 sec)  
  
mysql> CALL check_salary(5000);  
+-----+  
| message |  
+-----+  
| Salary acceptable |  
+-----+
```

## Problem 5: Exception Handling

**Task:** Create a function `annual_salary` that returns annual salary as  $(12 \times \text{monthly salary})$ .

**Code:** —

```
DELIMITER //  
CREATE FUNCTION annual_salary(monthly DECIMAL(10,2))  
RETURNS DECIMAL(10,2)  
DETERMINISTIC  
BEGIN  
    RETURN monthly * 12;
```

```

END//
DELIMITER ;

SELECT emp_name, salary, annual_salary(salary) AS yearly FROM
employees;

```

Output: —

emp_name	salary	yearly
Vinod	43000.00	516000.00
Emp1	3000.00	36000.00
Emp2	6000.00	72000.00
Emp3	9000.00	108000.00
Emp4	12000.00	144000.00
Emp5	15000.00	180000.00

## Problem 6: Exception Handling

**Task:** Create a stored procedure `count_employees` to count employees with non-null department IDs, using a `CONTINUE` handler to skip null values.

Code: —

```

DELIMITER //
CREATE PROCEDURE count_employees()
BEGIN
    DECLARE done INT DEFAULT 0;
    DECLARE total INT DEFAULT 0;
    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;

    SELECT COUNT(*) INTO total FROM employees WHERE dept_id IS NOT
        NULL;
    SELECT CONCAT('Total employees (with dept): ', total) AS
        message;
END//
DELIMITER ;

CALL count_employees();

```

Output: —

message
▶ Total employees (with dept): 6

## Problem 7: Exception Handling

**Task:** Create a stored procedure `delete_employee` that deletes an employee based on `emp_id`. If the employee is not found, display "No such employee".

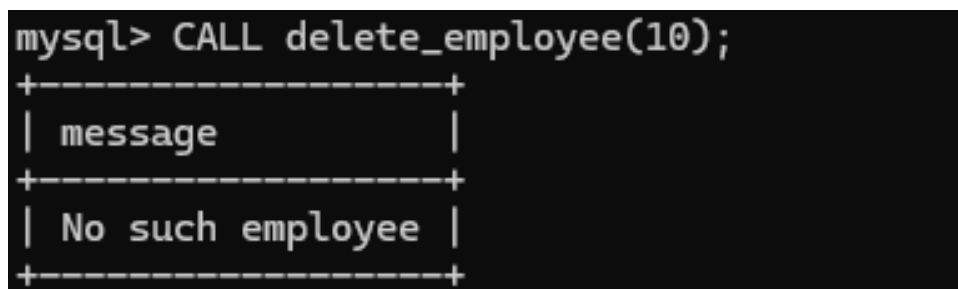
**Code:** —

```
DELIMITER //
CREATE PROCEDURE delete_employee(IN eid INT)
BEGIN
    DECLARE EXIT HANDLER FOR SQLEXCEPTION
    BEGIN
        SELECT 'Error deleting employee' AS message;
    END;

    IF (SELECT COUNT(*) FROM employees WHERE emp_id = eid) = 0
    THEN
        SELECT 'No such employee' AS message;
    ELSE
        DELETE FROM employees WHERE emp_id = eid;
        SELECT 'Employee deleted' AS message;
    END IF;
END//
DELIMITER ;

CALL delete_employee(10);
```

**Output:** —



```
mysql> CALL delete_employee(10);
+-----+
| message |
+-----+
| No such employee |
+-----+
```

## Problem 8: Exception Handling

**Task:** Create a stored procedure `safe_add_employee` that adds an employee only if the given department ID exists; otherwise, use `SIGNAL` to raise an error "Invalid Department ID".

**Code:** —

```
DELIMITER //
CREATE PROCEDURE safe_add_employee(IN name VARCHAR(100), IN sal
    DECIMAL(10,2), IN did INT)
BEGIN
    DECLARE invalid_dept CONDITION FOR SQLSTATE '45000';
    DECLARE EXIT HANDLER FOR invalid_dept
```

```

BEGIN
    SELECT 'Invalid Department ID' AS message;
END;

IF (SELECT COUNT(*) FROM departments WHERE dept_id = did) = 0
    THEN
        SIGNAL invalid_dept;
    ELSE
        INSERT INTO employees(emp_name, salary, dept_id)
        VALUES (name, sal, did);
        SELECT 'Employee added safely' AS message;
    END IF;
END//
DELIMITER ;

CALL safe_add_employee('Bob', 7000, 1);
CALL safe_add_employee('Charlie', 6000, 99);

```

Output: —

```

mysql> CALL safe_add_employee('Bob', 7000, 1);
+-----+
| message |
+-----+
| Employee added safely |
+-----+
1 row in set (0.01 sec)

Query OK, 0 rows affected (0.01 sec)

mysql> CALL safe_add_employee('Charlie', 6000, 99);
+-----+
| message |
+-----+
| Invalid Department ID |
+-----+

```

## Problem 9: Exception Handling

**Task:** Create a stored procedure `increase_salary` that increases salary by 10% for all employees; skip any employees with NULL salary using a `CONTINUE` handler.

**Code:** —

```

DELIMITER //
CREATE PROCEDURE increase_salary()
BEGIN
    DECLARE done INT DEFAULT 0;
    DECLARE eid INT;
    DECLARE sal DECIMAL(10,2);

```

```
DECLARE cur CURSOR FOR SELECT emp_id, salary FROM employees;
DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;

OPEN cur;
loop1: LOOP
    FETCH cur INTO eid, sal;
    IF done THEN
        LEAVE loop1;
    END IF;

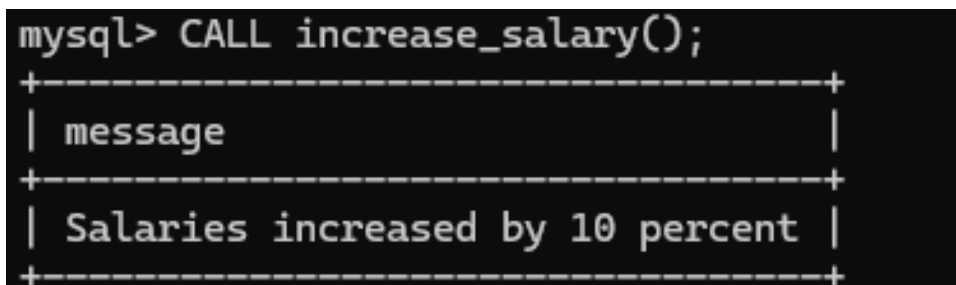
    IF sal IS NULL THEN
        ITERATE loop1;
    END IF;

    UPDATE employees SET salary = salary * 1.10 WHERE emp_id =
        eid;
END LOOP;
CLOSE cur;

SELECT 'Salaries increased by 10 percent' AS message;
END //
DELIMITER ;

CALL increase_salary();
```

Output: —



```
mysql> CALL increase_salary();
+-----+
| message |
+-----+
| Salaries increased by 10 percent |
+-----+
```

## Problem 10: Exception Handling

**Task:** Create a stored procedure `list_department_employees` that lists all employees of a given department using a `CURSOR` and handles the `NOT FOUND` condition gracefully with a `CONTINUE` handler.

**Code:** —

```
DELIMITER //
CREATE PROCEDURE list_department_employees(IN did INT)
BEGIN
    DECLARE done INT DEFAULT 0;
    DECLARE ename VARCHAR(100);
    DECLARE cur CURSOR FOR SELECT emp_name FROM employees WHERE
        dept_id = did;
```

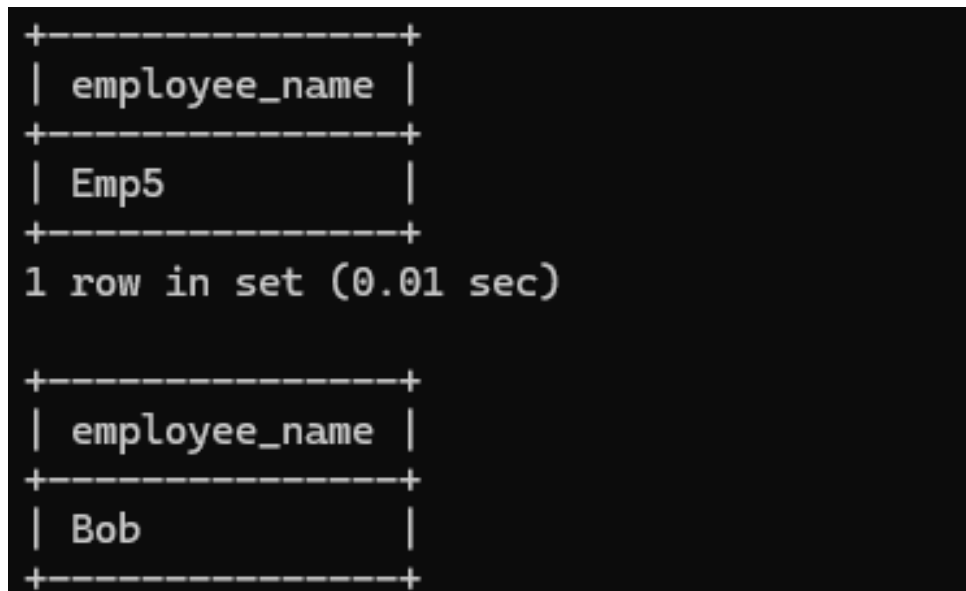


```
DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;

OPEN cur;
loop2: LOOP
    FETCH cur INTO ename;
    IF done THEN
        LEAVE loop2;
    END IF;
    SELECT ename AS employee_name;
END LOOP;
CLOSE cur;
END//
DELIMITER ;

CALL list_department_employees(1);
```

Output: —



The screenshot displays two SQL query results. The first result shows a table with one row containing the name 'Emp5'. The second result shows a table with one row containing the name 'Bob'. Both tables have a single column labeled 'employee\_name'.

employee_name
Emp5

1 row in set (0.01 sec)

employee_name
Bob