**Lab Exercise 1: Creating a Simple Stored Procedure**

**Task:** Create a stored procedure GetAllEmployees that retrieves all records from the employees table.

**Solution:**

DELIMITER //

CREATE PROCEDURE GetAllEmployees()

BEGIN

SELECT \* FROM employees;

END //

DELIMITER ;

**Lab Exercise 2: Creating a Stored Procedure with Input Parameter**

**Task:** Create a stored procedure GetEmployeeByID that takes an emp\_id as input and retrieves the employee details.

**Solution:**

DELIMITER //

CREATE PROCEDURE GetEmployeeByID(IN emp\_id INT)

BEGIN

SELECT \* FROM employees WHERE id = emp\_id;

END //

DELIMITER ;

**Lab Exercise 3: Creating a Stored Procedure with Multiple Input Parameters**

**Task:** Create a stored procedure GetEmployeesByDepartmentAndSalary that takes dept\_name and min\_salary as input.

**Solution:**

DELIMITER //

CREATE PROCEDURE GetEmployeesByDepartmentAndSalary(IN dept\_name VARCHAR(50), IN min\_salary DECIMAL(10,2))

BEGIN

SELECT \* FROM employees WHERE department = dept\_name AND salary >= min\_salary;

END //

DELIMITER ;

**Lab Exercise 4: Creating a Stored Procedure with Output Parameter**

**Task:** Create a stored procedure GetEmployeeCount that returns the total number of employees.

**Solution:**

DELIMITER //

CREATE PROCEDURE GetEmployeeCount(OUT emp\_count INT)

BEGIN

SELECT COUNT(\*) INTO emp\_count FROM employees;

END //

DELIMITER ;

**Lab Exercise 5: Using Variables in Stored Procedure**

**Task:** Create a stored procedure CalculateTotalSalary that calculates the total salary of all employees.

**Solution:**

DELIMITER //

CREATE PROCEDURE CalculateTotalSalary(OUT total\_salary DECIMAL(10,2))

BEGIN

SELECT SUM(salary) INTO total\_salary FROM employees;

END //

DELIMITER ;

**Lab Exercise 6: Stored Procedure with IF Statement**

**Task:** Create a stored procedure CheckSalary that checks if an employee’s salary is greater than a given value.

**Solution:**

DELIMITER //

CREATE PROCEDURE CheckSalary(IN emp\_id INT, IN min\_salary DECIMAL(10,2), OUT result VARCHAR(50))

BEGIN

DECLARE emp\_salary DECIMAL(10,2);

SELECT salary INTO emp\_salary FROM employees WHERE id = emp\_id;

IF emp\_salary > min\_salary THEN

SET result = 'Above Minimum';

ELSE

SET result = 'Below Minimum';

END IF;

END //

DELIMITER ;

**Lab Exercise 7: Stored Procedure with LOOP**

**Task:** Create a stored procedure PrintNumbers that prints numbers from 1 to a given number.

**Solution:**

DELIMITER //

CREATE PROCEDURE PrintNumbers(IN num INT)

BEGIN

DECLARE counter INT DEFAULT 1;

loop\_label: LOOP

IF counter > num THEN

LEAVE loop\_label;

END IF;

SELECT counter;

SET counter = counter + 1;

END LOOP loop\_label;

END //

DELIMITER ;

**Lab Exercise 8: Using WHILE Loop**

**Task:** Create a stored procedure SumNumbers that calculates the sum of numbers from 1 to N.

**Solution:**

DELIMITER //

CREATE PROCEDURE SumNumbers(IN num INT, OUT total\_sum INT)

BEGIN

DECLARE counter INT DEFAULT 1;

SET total\_sum = 0;

WHILE counter <= num DO

SET total\_sum = total\_sum + counter;

SET counter = counter + 1;

END WHILE;

END //

DELIMITER ;

**Lab Exercise 9: Using REPEAT Loop**

**Task:** Create a stored procedure CountEvenNumbers that counts the number of even numbers up to N.

**Solution:**

DELIMITER //

CREATE PROCEDURE CountEvenNumbers(IN num INT, OUT even\_count INT)

BEGIN

DECLARE counter INT DEFAULT 1;

SET even\_count = 0;

REPEAT

IF counter MOD 2 = 0 THEN

SET even\_count = even\_count + 1;

END IF;

SET counter = counter + 1;

UNTIL counter > num END REPEAT;

END //

DELIMITER ;

**Lab Exercise 10: Handling Errors with DECLARE HANDLER**

**Task:** Create a stored procedure DivideNumbers that handles division by zero.

**Solution:**

DELIMITER //

CREATE PROCEDURE DivideNumbers(IN num1 INT, IN num2 INT, OUT result DECIMAL(10,2))

BEGIN

DECLARE CONTINUE HANDLER FOR SQLEXCEPTION SET result = NULL;

IF num2 = 0 THEN

SET result = NULL;

ELSE

SET result = num1 / num2;

END IF;

END //

DELIMITER ;

**Lab Exercise 11: Updating Data with Stored Procedure**

**Task:** Create a stored procedure UpdateSalary to increase an employee’s salary by a given percentage.

**Solution:**

DELIMITER //

CREATE PROCEDURE UpdateSalary(IN emp\_id INT, IN percent\_increase DECIMAL(5,2))

BEGIN

UPDATE employees

SET salary = salary + (salary \* percent\_increase / 100)

WHERE id = emp\_id;

END //

DELIMITER ;

**Lab Exercise 12: Deleting Data Using Stored Procedure**

**Task:** Create a stored procedure DeleteEmployee that deletes an employee by ID.

**Solution:**

DELIMITER //

CREATE PROCEDURE DeleteEmployee(IN emp\_id INT)

BEGIN

DELETE FROM employees WHERE id = emp\_id;

END //

DELIMITER ;

**Lab Exercise 13: Using Transactions in Stored Procedure**

**Task:** Create a stored procedure TransferSalary to transfer salary from one employee to another.

**Solution:**

DELIMITER //

CREATE PROCEDURE TransferSalary(IN from\_emp INT, IN to\_emp INT, IN amount DECIMAL(10,2))

BEGIN

START TRANSACTION;

UPDATE employees SET salary = salary - amount WHERE id = from\_emp;

UPDATE employees SET salary = salary + amount WHERE id = to\_emp;

COMMIT;

END //

DELIMITER ;

**Lab Exercise 14: Using Cursor in Stored Procedure**

**Task:** Create a stored procedure ListEmployeeNames that retrieves all employee names.

**Solution:**

DELIMITER //

CREATE PROCEDURE ListEmployeeNames()

BEGIN

DECLARE emp\_name VARCHAR(100);

DECLARE done INT DEFAULT FALSE;

DECLARE cur CURSOR FOR SELECT name FROM employees;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

OPEN cur;

read\_loop: LOOP

FETCH cur INTO emp\_name;

IF done THEN

LEAVE read\_loop;

END IF;

SELECT emp\_name;

END LOOP;

CLOSE cur;

END //

DELIMITER ;