**WEEK 2**

**PL/SQL programming**

**Exercise 1: Control Structures**

DELIMITER //

CREATE PROCEDURE check\_number(IN num INT)

BEGIN

IF num > 0 THEN

SELECT 'The number is positive' AS result;

ELSEIF num < 0 THEN

SELECT 'The number is negative' AS result;

ELSE

SELECT 'The number is zero' AS result;

END IF;

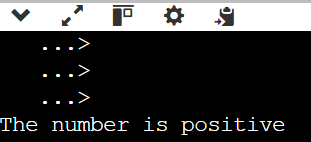
END;

//

DELIMITER ;

CALL check\_number(5);

**OUTPUT**:



**Exercise 3: Stored Procedures**

CREATE TABLE employees (

emp\_id INT PRIMARY KEY,

emp\_name VARCHAR(50),

salary INT

);

INSERT INTO employees VALUES (101, 'ilakeyah', 30000);

INSERT INTO employees VALUES (102, 'pranaya', 40000);

DELIMITER //

CREATE PROCEDURE get\_employee\_details(IN empId INT)

BEGIN

SELECT emp\_name, salary

FROM employees

WHERE emp\_id = empId;

END;

//

DELIMITER ;

CALL get\_employee\_details(101);

**OUTPUT:**



**TDD using JUnit5 and Mockito**

**Exercise 1: Setting Up Junit**

Calculator.java

public class Calculator {

public int add(int a, int b) {

return a + b;

}

}

CalculatorTest.java

import static org.junit.jupiter.api.Assertions.\*;

import org.junit.jupiter.api.Test;

public class CalculatorTest {

@Test

public void testAdd() {

Calculator calc = new Calculator();

assertEquals(5, calc.add(2, 3));

}

}

**Exercise 3: Assertions in Junit**

@Test

public void testAssertions() {

String expected = "Hello";

String actual = "Hello";

assertEquals(expected, actual);

assertNotNull(actual);

assertTrue(actual.startsWith("H"));

assertFalse(actual.isEmpty());

}