**WEEK 2**

**PLSQL\_Exercises**

**Exercise 1: Control Structures**

**Scenario 1**

CREATE DATABASE IF NOT EXISTS bankdb;

USE bankdb;

CREATE TABLE IF NOT EXISTS customers (

customer\_id INT PRIMARY KEY,

name VARCHAR(100),

age INT,

loan\_interest\_rate DECIMAL(5,2)

);

INSERT INTO customers (customer\_id, name, age, loan\_interest\_rate)

VALUES

(1, 'John Smith', 65, 10.5),

(2, 'Alice Brown', 55, 9.5),

(3, 'Robert Green', 70, 11.0)

AS new\_vals(customer\_id, name, age, loan\_interest\_rate)

ON DUPLICATE KEY UPDATE

name = new\_vals.name,

age = new\_vals.age,

loan\_interest\_rate = new\_vals.loan\_interest\_rate;

DROP PROCEDURE IF EXISTS ApplyLoanDiscount;

DELIMITER $$

CREATE PROCEDURE ApplyLoanDiscount()

BEGIN

DECLARE done INT DEFAULT FALSE;

DECLARE v\_customer\_id INT;

DECLARE v\_age INT;

DECLARE v\_interest\_rate DECIMAL(5,2);

DECLARE cur CURSOR FOR

SELECT customer\_id, age, loan\_interest\_rate FROM customers;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

OPEN cur;

read\_loop: LOOP

FETCH cur INTO v\_customer\_id, v\_age, v\_interest\_rate;

IF done THEN

LEAVE read\_loop;

END IF;

IF v\_age > 60 THEN

UPDATE customers

SET loan\_interest\_rate = loan\_interest\_rate - 1

WHERE customer\_id = v\_customer\_id;

END IF;

END LOOP;

CLOSE cur;

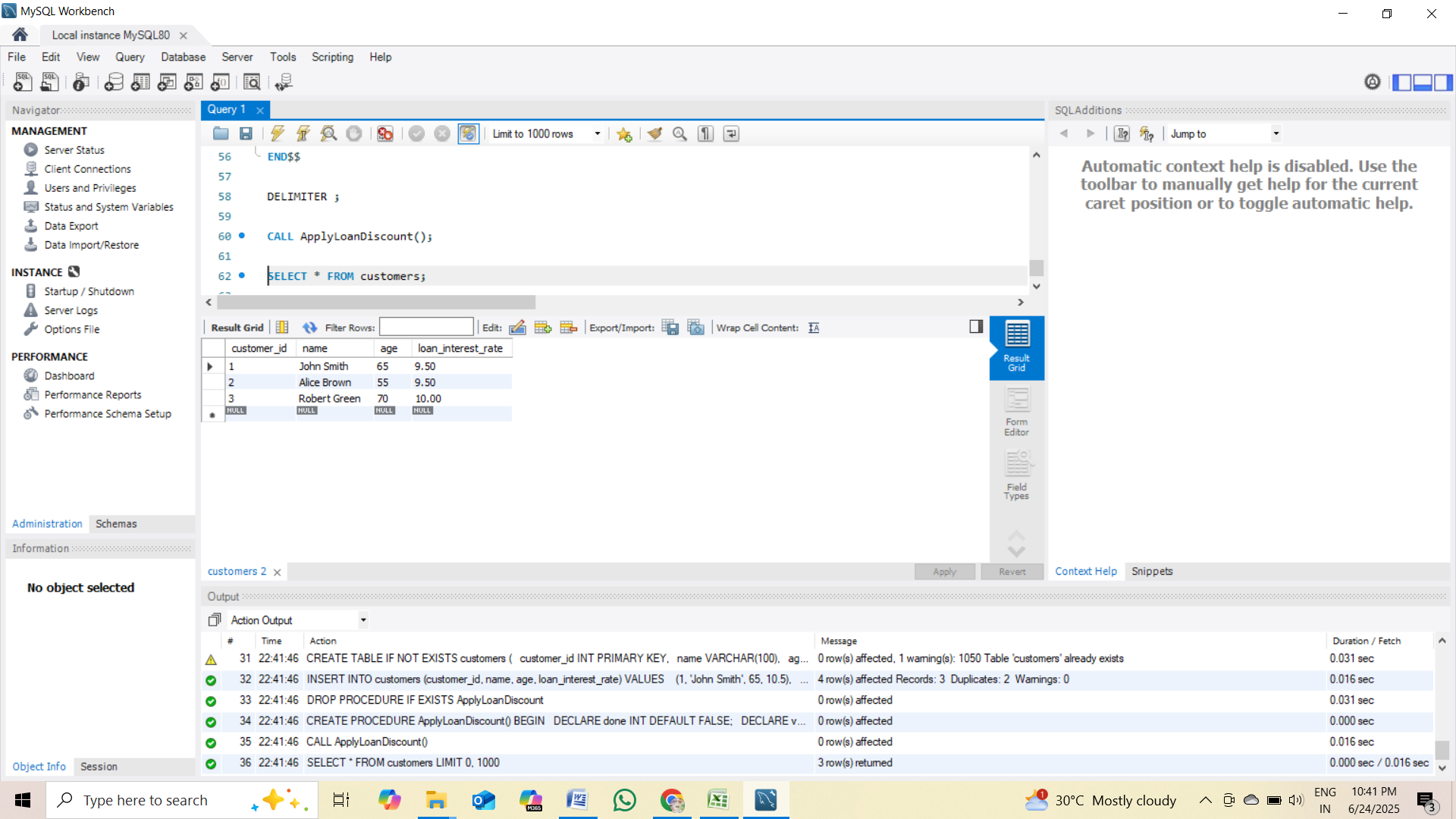
END$$

DELIMITER ;

CALL ApplyLoanDiscount();

SELECT \* FROM customers;

**OUTPUT**

****

**Scenario 2**

ALTER TABLE customers

ADD COLUMN balance DECIMAL(10,2) DEFAULT 0,

ADD COLUMN is\_vip BOOLEAN DEFAULT FALSE;

UPDATE customers SET balance = 12000 WHERE customer\_id = 1; -- Should be VIP

UPDATE customers SET balance = 8000 WHERE customer\_id = 2; -- Not VIP

UPDATE customers SET balance = 15000 WHERE customer\_id = 3; -- Should be VIP

DROP PROCEDURE IF EXISTS PromoteVIPCustomers;

DELIMITER $$

CREATE PROCEDURE PromoteVIPCustomers()

BEGIN

DECLARE done INT DEFAULT FALSE;

DECLARE v\_customer\_id INT;

DECLARE v\_balance DECIMAL(10,2);

DECLARE cur CURSOR FOR

SELECT customer\_id, balance FROM customers;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

OPEN cur;

read\_loop: LOOP

FETCH cur INTO v\_customer\_id, v\_balance;

IF done THEN

LEAVE read\_loop;

END IF;

IF v\_balance > 10000 THEN

UPDATE customers

SET is\_vip = TRUE

WHERE customer\_id = v\_customer\_id;

END IF;

END LOOP;

CLOSE cur;

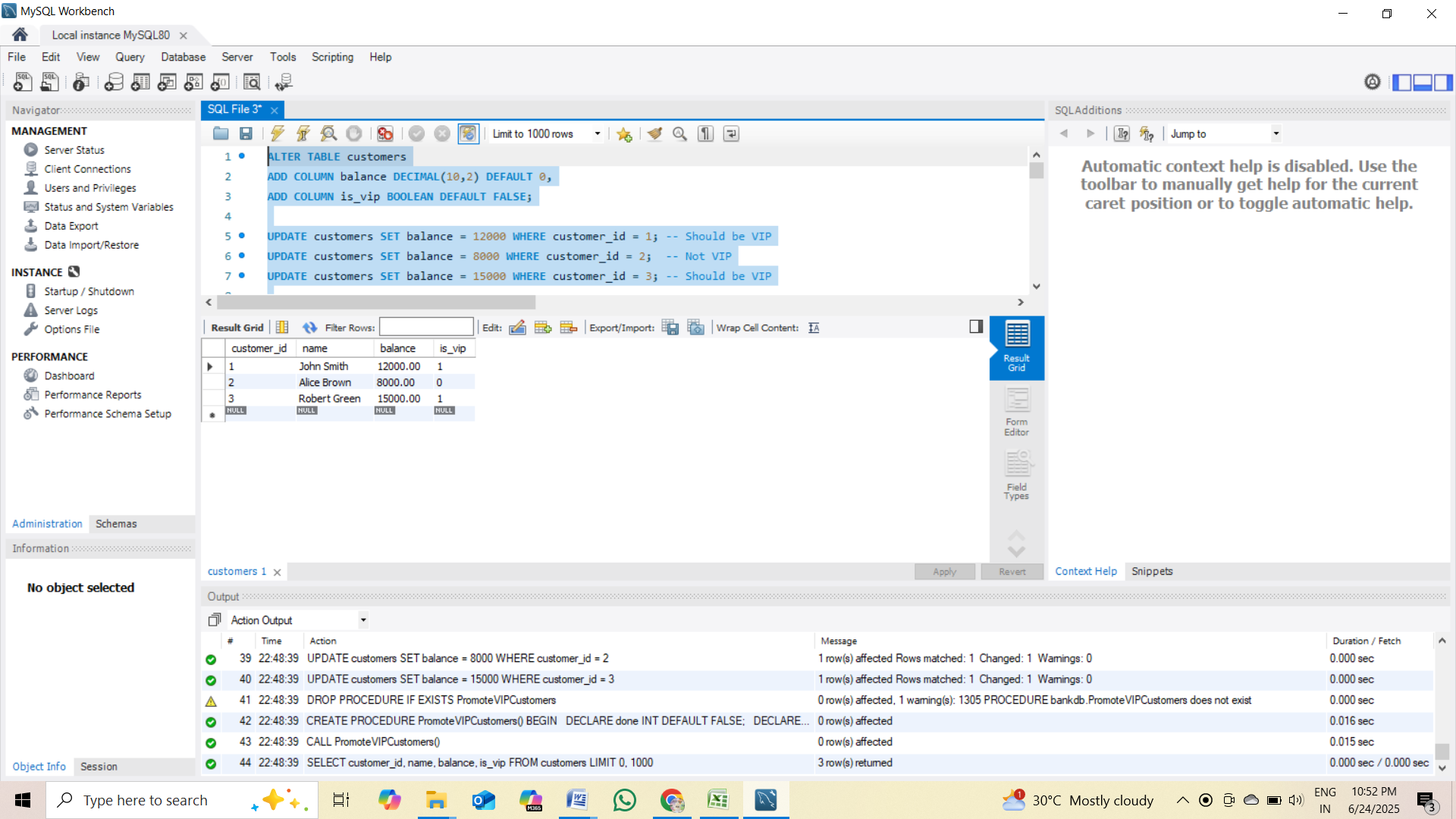
END$$

DELIMITER ;

CALL PromoteVIPCustomers();

SELECT customer\_id, name, balance, is\_vip FROM customers;

**OUTPUT**



**Scenario 3**

DROP PROCEDURE IF EXISTS SendLoanReminders;

DROP TEMPORARY TABLE IF EXISTS temp\_reminders;

DELIMITER $$

CREATE PROCEDURE SendLoanReminders()

BEGIN

DECLARE done INT DEFAULT 0;

DECLARE v\_customer\_id INT;

DECLARE v\_name VARCHAR(100);

DECLARE v\_due\_date DATE;

DECLARE v\_amount DECIMAL(10,2);

-- Declare cursor and handler BEFORE any executable statements

DECLARE loan\_cursor CURSOR FOR

SELECT c.customer\_id, c.name, l.due\_date, l.amount

FROM loans l

JOIN customers c ON l.customer\_id = c.customer\_id

WHERE l.due\_date BETWEEN CURDATE() AND DATE\_ADD(CURDATE(), INTERVAL 30 DAY);

DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;

-- Create temporary table AFTER declarations

CREATE TEMPORARY TABLE IF NOT EXISTS temp\_reminders (

Reminder\_Message VARCHAR(255)

);

-- Open cursor and loop

OPEN loan\_cursor;

loan\_loop: LOOP

FETCH loan\_cursor INTO v\_customer\_id, v\_name, v\_due\_date, v\_amount;

IF done THEN

LEAVE loan\_loop;

END IF;

-- Insert reminder message into the temp table

INSERT INTO temp\_reminders (Reminder\_Message)

VALUES (

CONCAT('Reminder: Dear ', v\_name,

', your loan of ₹', v\_amount,

' is due on ', DATE\_FORMAT(v\_due\_date, '%d-%m-%Y'),

'. Please make the payment on time.')

);

END LOOP;

CLOSE loan\_cursor;

-- Display all messages in one result set

SELECT \* FROM temp\_reminders;

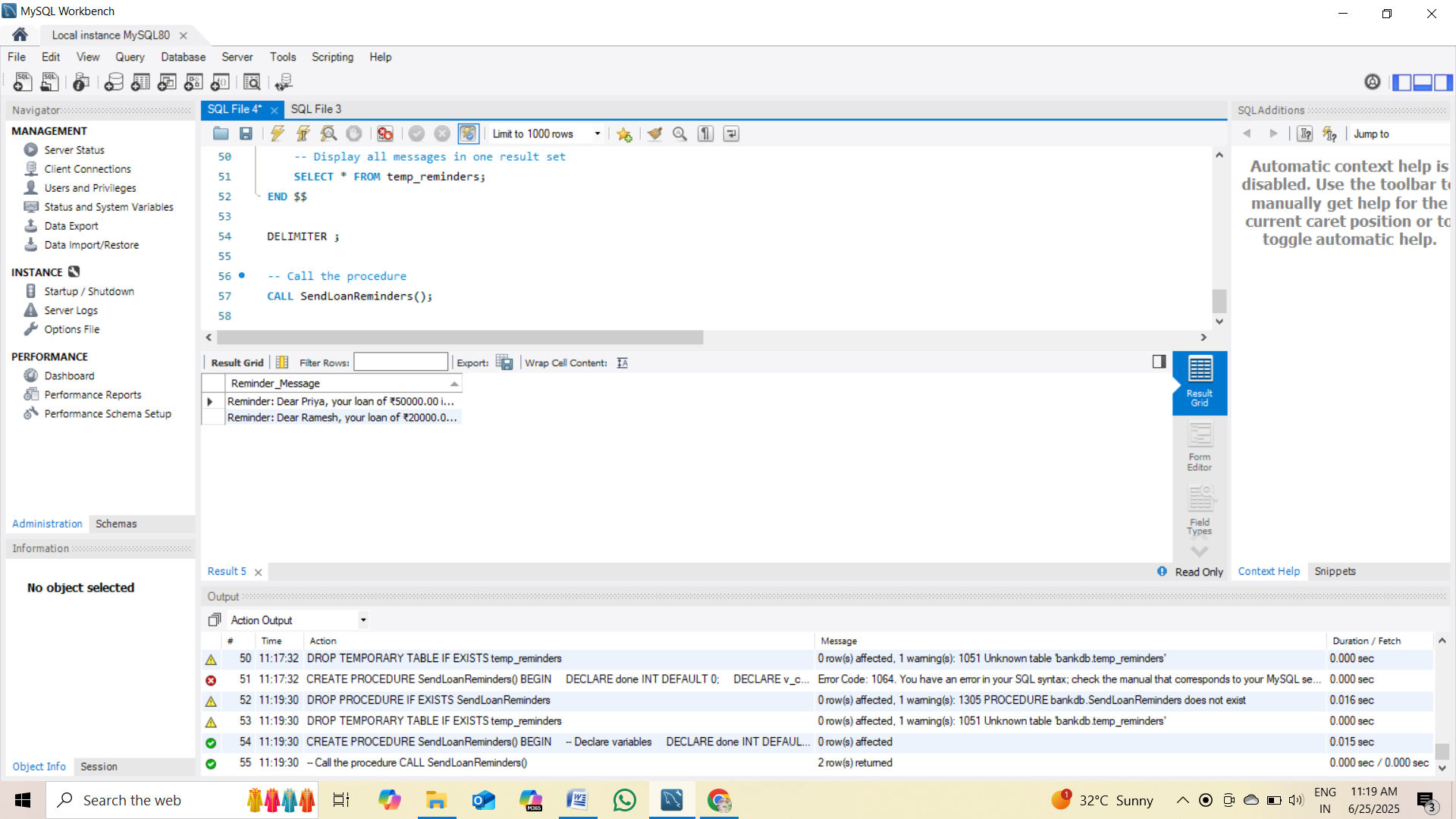
END $$

DELIMITER ;

-- Call the procedure

CALL SendLoanReminders();

**OUTPUT**



**Exercise 3: Stored Procedures**

**Scenario 1**

-- Step 1: Create and Use the Database

CREATE DATABASE IF NOT EXISTS bankdb;

USE bankdb;

-- Step 2: Drop and Recreate Table (Fixes column error)

DROP TABLE IF EXISTS savings\_accounts;

CREATE TABLE savings\_accounts (

account\_id INT PRIMARY KEY,

customer\_name VARCHAR(100),

balance DECIMAL(10,2)

);

-- Step 3: Insert Sample Data

INSERT INTO savings\_accounts (account\_id, customer\_name, balance)

VALUES

(1, 'Alice', 1000.00),

(2, 'Bob', 2000.00);

-- Step 4: Drop Procedure if It Exists

DROP PROCEDURE IF EXISTS ProcessMonthlyInterest;

-- Step 5: Create the Procedure

DELIMITER $$

CREATE PROCEDURE ProcessMonthlyInterest()

BEGIN

-- Add 1% interest to each savings account

UPDATE savings\_accounts

SET balance = balance + (balance \* 0.01);

END $$

DELIMITER ;

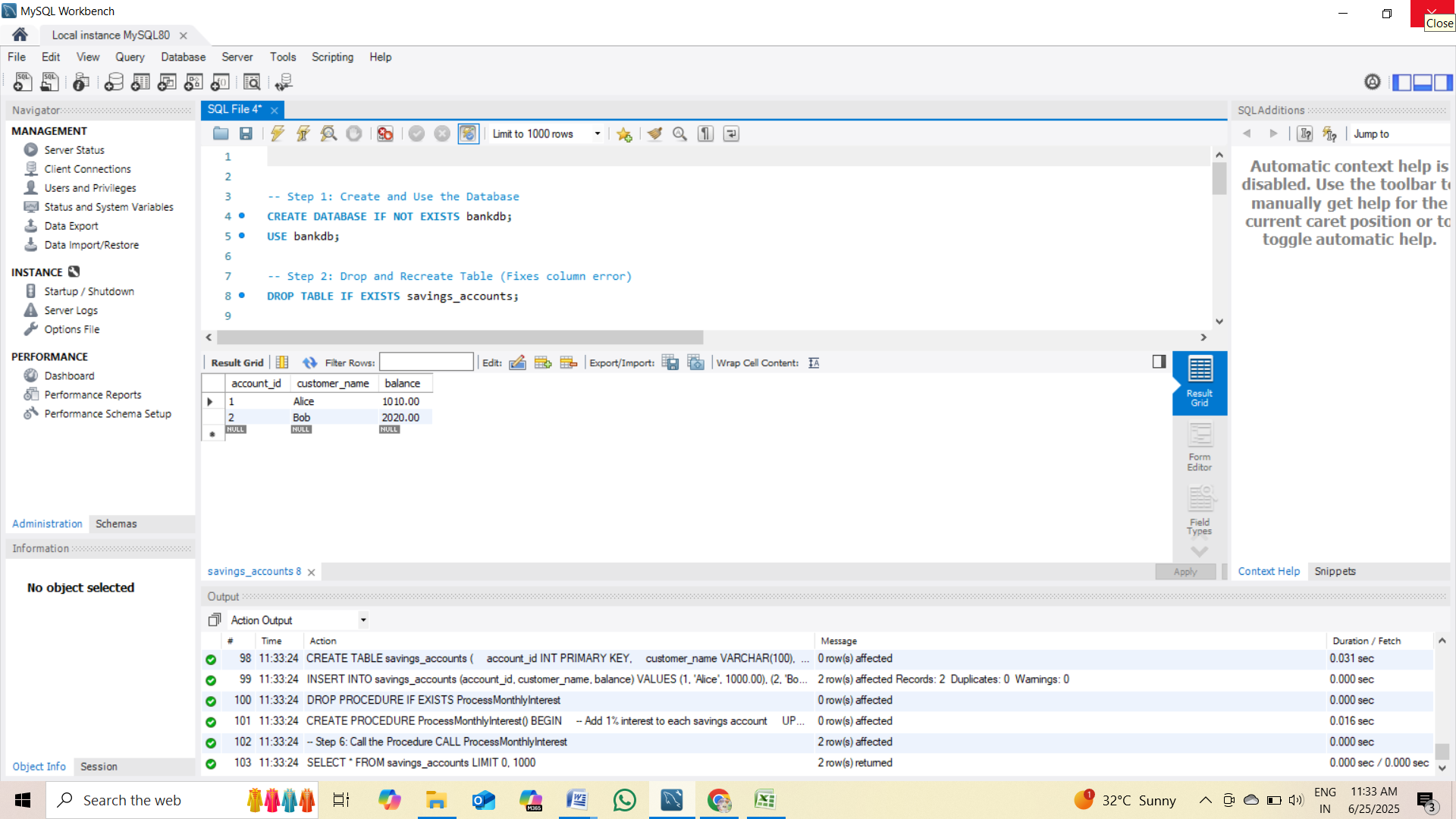
-- Step 6: Call the Procedure

CALL ProcessMonthlyInterest;

-- Step 7: View Results

SELECT \* FROM savings\_accounts;

**OUTPUT**

****

**Scenario 2**

-- Step 1: Create and use the database

CREATE DATABASE IF NOT EXISTS bankdb;

USE bankdb;

-- Step 2: Drop and recreate the employees table

DROP TABLE IF EXISTS employees;

CREATE TABLE employees (

emp\_id INT PRIMARY KEY,

emp\_name VARCHAR(100),

department VARCHAR(50),

salary DECIMAL(10,2)

);

-- Step 3: Insert sample data

INSERT INTO employees (emp\_id, emp\_name, department, salary)

VALUES

(1, 'Alice', 'HR', 50000),

(2, 'Bob', 'Finance', 60000),

(3, 'Charlie', 'HR', 55000),

(4, 'David', 'IT', 70000);

-- Step 4: Drop procedure if it exists

DROP PROCEDURE IF EXISTS UpdateEmployeeBonus;

-- Step 5: Create the stored procedure

DELIMITER $$

CREATE PROCEDURE UpdateEmployeeBonus(

IN dept\_name VARCHAR(50),

IN bonus\_percent DECIMAL(5,2)

)

BEGIN

UPDATE employees

SET salary = salary + (salary \* (bonus\_percent / 100))

WHERE department = dept\_name;

END $$

DELIMITER ;

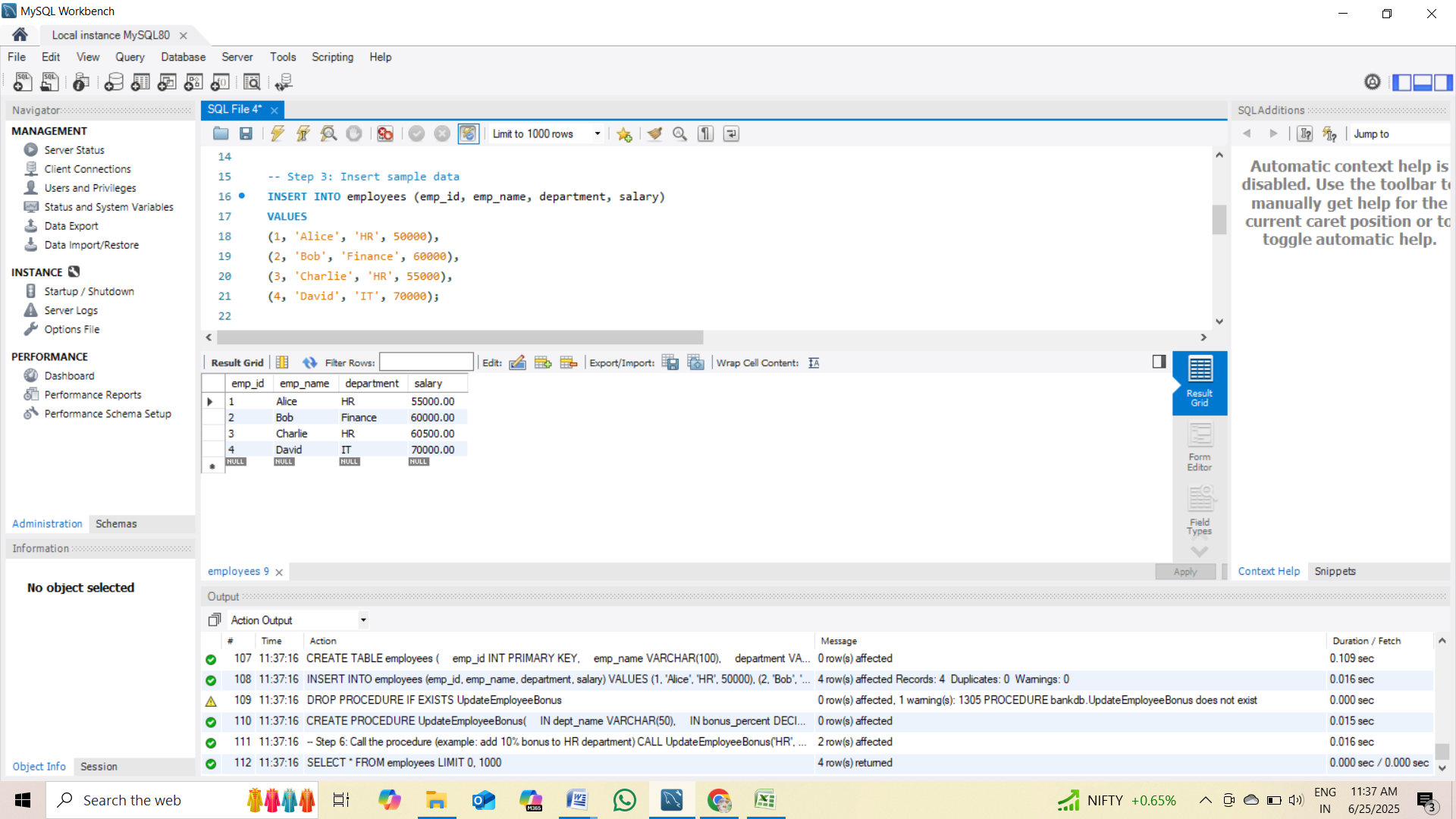
-- Step 6: Call the procedure (example: add 10% bonus to HR department)

CALL UpdateEmployeeBonus('HR', 10);

-- Step 7: View updated salaries

SELECT \* FROM employees;

**OUTPUT**



**Scenario 3**

-- Step 1: Use or create the database

CREATE DATABASE IF NOT EXISTS bankdb;

USE bankdb;

-- Step 2: Drop and recreate the accounts table

DROP TABLE IF EXISTS accounts;

CREATE TABLE accounts (

account\_id INT PRIMARY KEY,

customer\_name VARCHAR(100),

balance DECIMAL(10,2)

);

-- Step 3: Insert sample data

INSERT INTO accounts (account\_id, customer\_name, balance)

VALUES

(1, 'Alice', 5000.00),

(2, 'Bob', 3000.00);

-- Step 4: Drop the procedure if it exists

DROP PROCEDURE IF EXISTS TransferFunds;

-- Step 5: Create the stored procedure

DELIMITER $$

CREATE PROCEDURE TransferFunds(

IN from\_account INT,

IN to\_account INT,

IN amount DECIMAL(10,2)

)

BEGIN

DECLARE from\_balance DECIMAL(10,2);

-- Get balance of source account

SELECT balance INTO from\_balance

FROM accounts

WHERE account\_id = from\_account;

-- Check for sufficient balance

IF from\_balance IS NULL THEN

SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'Source account does not exist';

ELSEIF from\_balance < amount THEN

SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'Insufficient funds in source account';

ELSE

-- Subtract from source account

UPDATE accounts

SET balance = balance - amount

WHERE account\_id = from\_account;

-- Add to destination account

UPDATE accounts

SET balance = balance + amount

WHERE account\_id = to\_account;

END IF;

END $$

DELIMITER ;

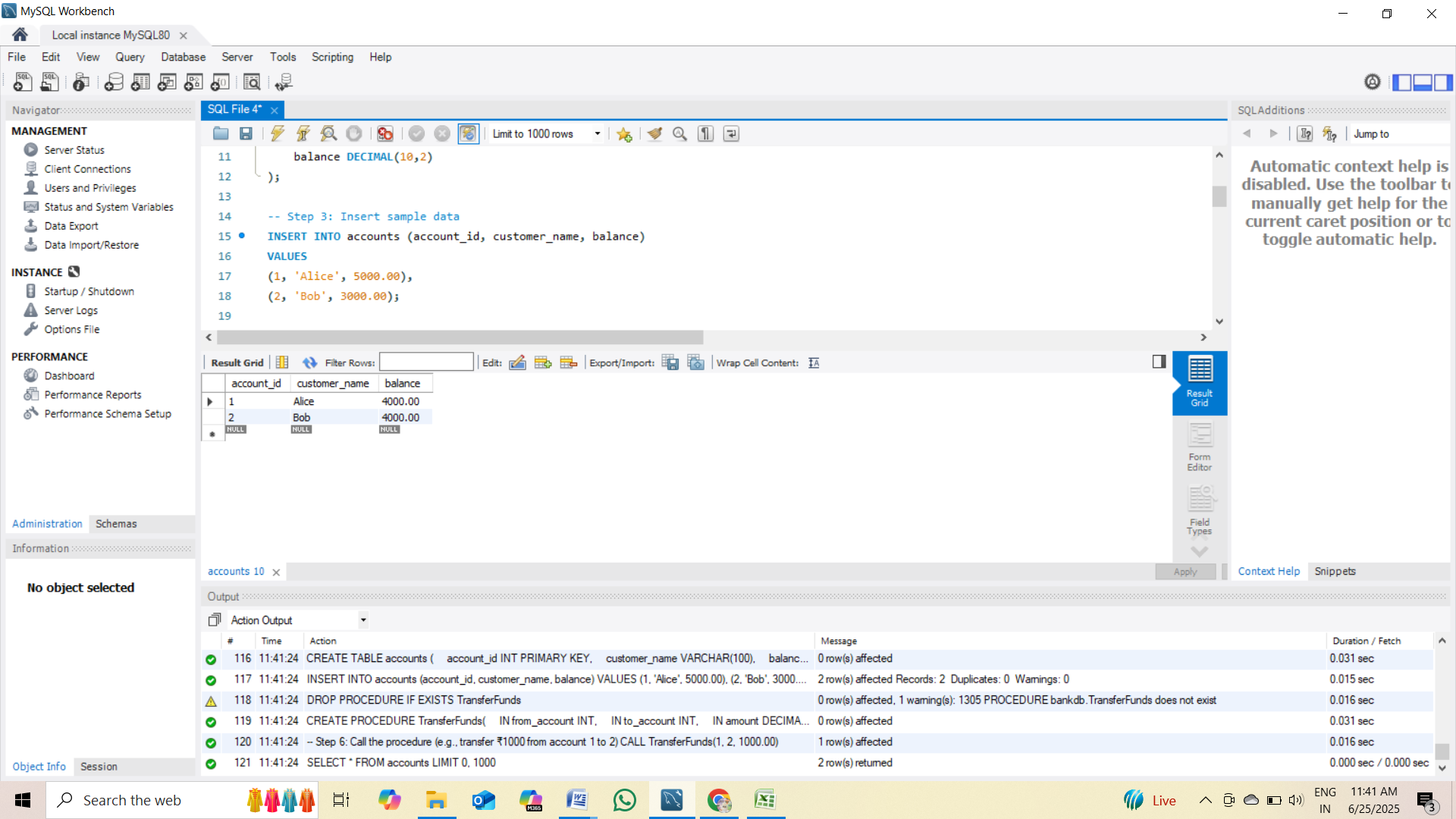
-- Step 6: Call the procedure (e.g., transfer ₹1000 from account 1 to 2)

CALL TransferFunds(1, 2, 1000.00);

-- Step 7: View updated balances

SELECT \* FROM accounts;

**OUTPUT**

****