**PL/SQL programming:**

**QUESTION 1: Exercise 1: Control Structures:**

**Scenario 1:** The bank wants to apply a discount to loan interest rates for customers above 60 years old.

* + **Question:** Write a PL/SQL block that loops through all customers, checks their age, and if they are above 60, apply a 1% discount to their current loan interest rates.

**Scenario 2:** A customer can be promoted to VIP status based on their balance.

* + **Question:** Write a PL/SQL block that iterates through all customers and sets a flag IsVIP to TRUE for those with a balance over $10,000.

**Scenario 3:** The bank wants to send reminders to customers whose loans are due within the next 30 days.

* + **Question:** Write a PL/SQL block that fetches all loans due in the next 30 days and prints a reminder message for each customer.

**Scenario 1-SOLUTION:**

create database bankdb;

use bankdb;

CREATE TABLE customers (

customer\_id INT PRIMARY KEY,

name VARCHAR(100),

age INT,

balance DECIMAL(10, 2),

IsVIP BOOLEAN DEFAULT FALSE

);

CREATE TABLE loans (

loan\_id INT PRIMARY KEY,

customer\_id INT,

interest\_rate DECIMAL(5, 2),

due\_date DATE,

FOREIGN KEY (customer\_id) REFERENCES customers(customer\_id)

);

-- Customers

INSERT INTO customers VALUES

(1, 'John', 65, 5000.00, FALSE),

(2, 'Alice', 45, 12000.00, FALSE),

(3, 'Bob', 70, 8000.00, FALSE);

-- Loans

INSERT INTO loans VALUES

(101, 1, 9.5, '2025-07-10'),

(102, 2, 8.0, '2025-06-30'),

(103, 3, 10.0, '2025-06-28');

DELIMITER $$

CREATE PROCEDURE ApplySeniorCitizenDiscount()

BEGIN

DECLARE done INT DEFAULT FALSE;

DECLARE cust\_id INT;

DECLARE cur CURSOR FOR

SELECT customer\_id FROM customers WHERE age > 60;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

OPEN cur;

read\_loop: LOOP

FETCH cur INTO cust\_id; -- Get the next customer\_id

IF done THEN

LEAVE read\_loop; -- Exit the loop when done

END IF;

UPDATE loans

SET interest\_rate = interest\_rate - 1

WHERE customer\_id = cust\_id;

END LOOP;

CLOSE cur;

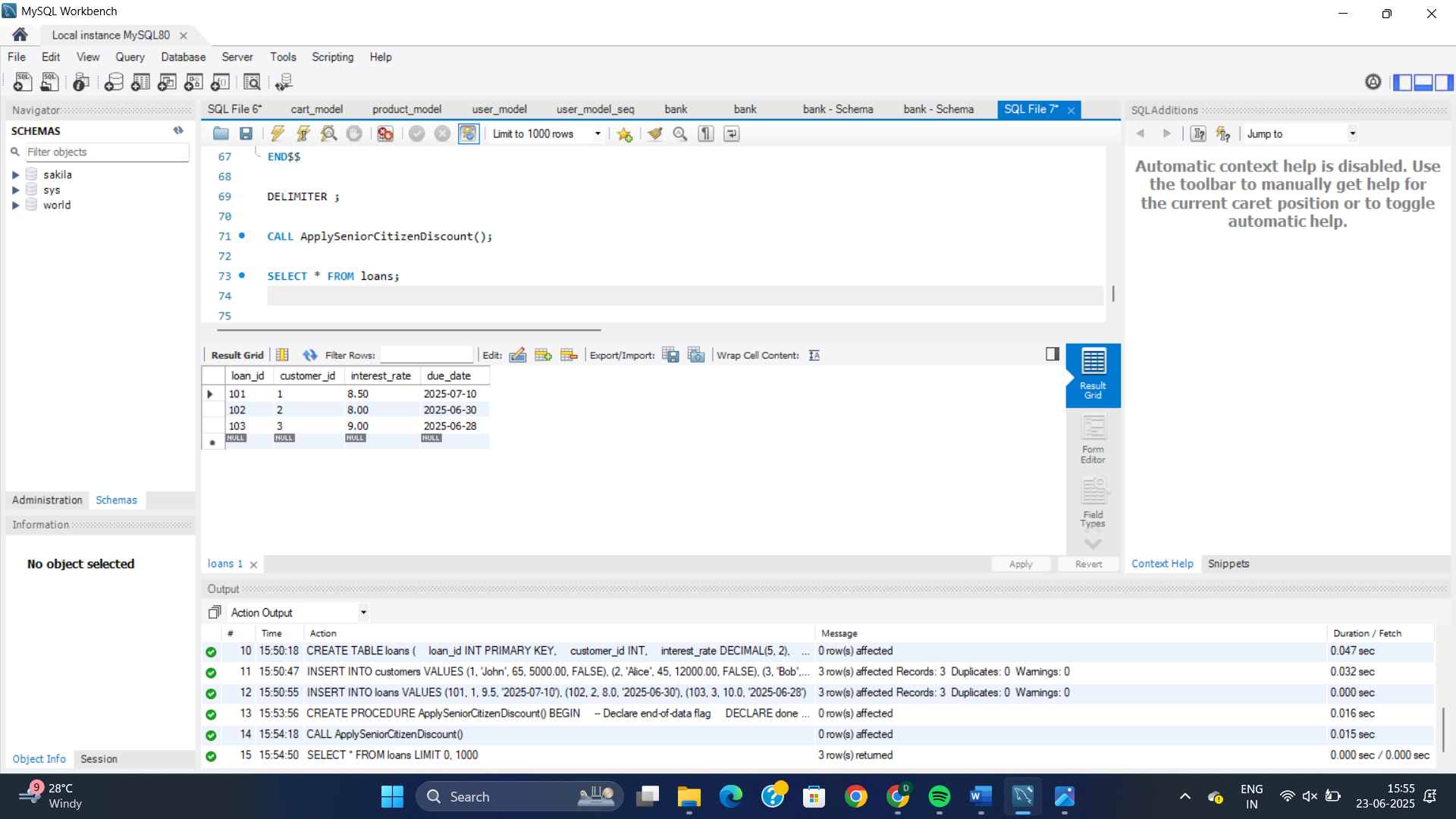
END$$

DELIMITER ;

CALL ApplySeniorCitizenDiscount();

SELECT \* FROM loans;

OUTPUT:



**Scenario 2-SOLUTION:**

DELIMITER $$

CREATE PROCEDURE PromoteToVIP()

BEGIN

UPDATE customers

SET IsVIP = TRUE

WHERE balance > 10000;

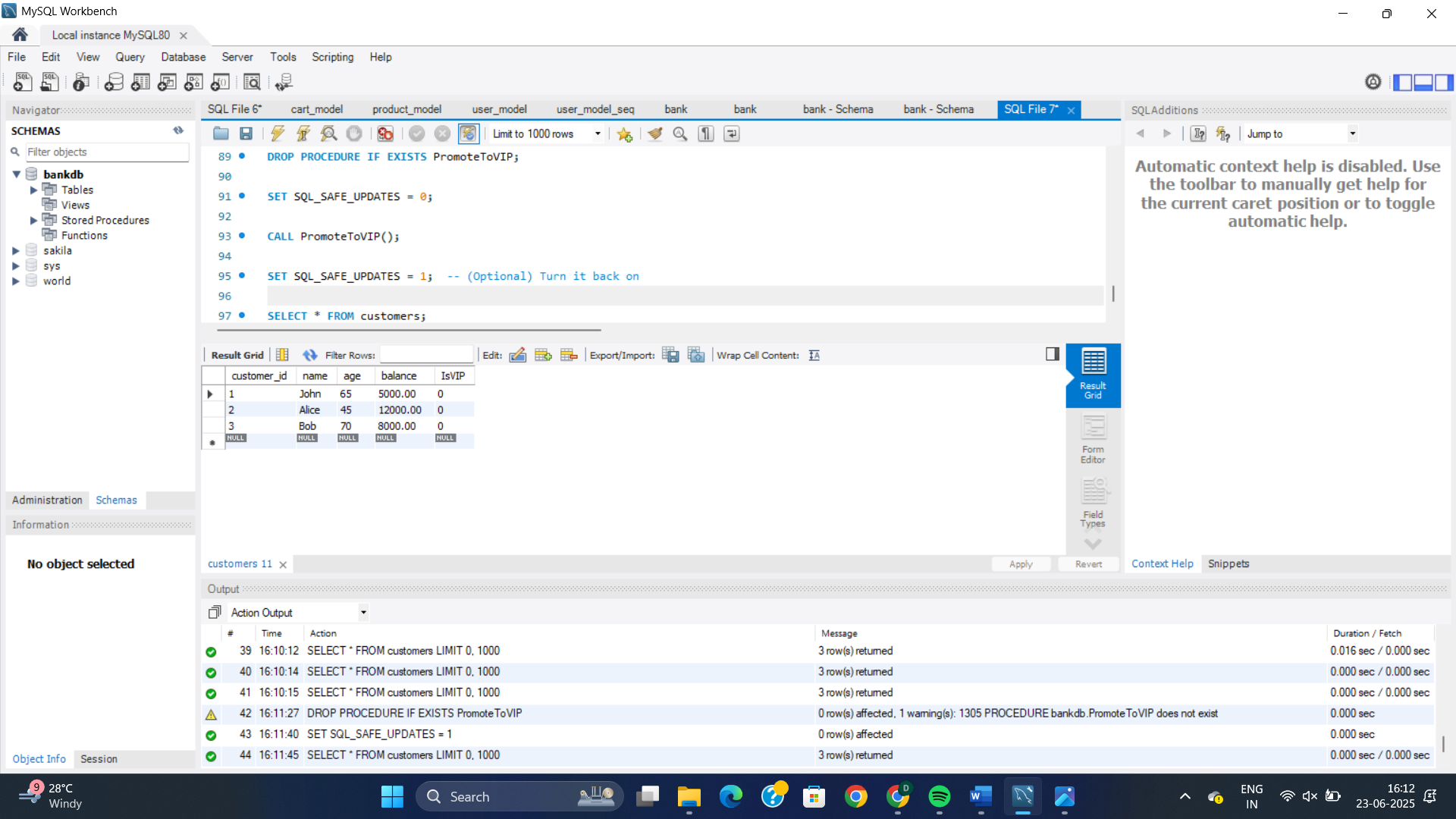
END$$

DELIMITER ;

CALL PromoteToVIP();

SELECT \* FROM customers;

OUTPUT:



**Scenario 3-SOLUTION:**

USE bankdb;

DELIMITER $$

CREATE PROCEDURE SendLoanReminders()

BEGIN

DECLARE done INT DEFAULT FALSE;

DECLARE v\_loan\_id INT;

DECLARE v\_customer\_name VARCHAR(100);

DECLARE v\_due\_date DATE;

DECLARE v\_amount\_due DECIMAL(10,2);

DECLARE loan\_cursor CURSOR FOR

SELECT

l.loan\_id,

c.name,

l.due\_date,

l.amount\_due

FROM loans l

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

WHERE l.due\_date BETWEEN CURDATE() AND CURDATE() + INTERVAL 30 DAY;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

OPEN loan\_cursor;

loan\_loop: LOOP

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

IF done THEN

LEAVE loan\_loop;

END IF;

SELECT CONCAT(

'Reminder: Dear ', v\_customer\_name,

', your loan (ID: ', v\_loan\_id,

') of amount ₹', v\_amount\_due,

' is due on ', v\_due\_date, '. Please make the payment on time.'

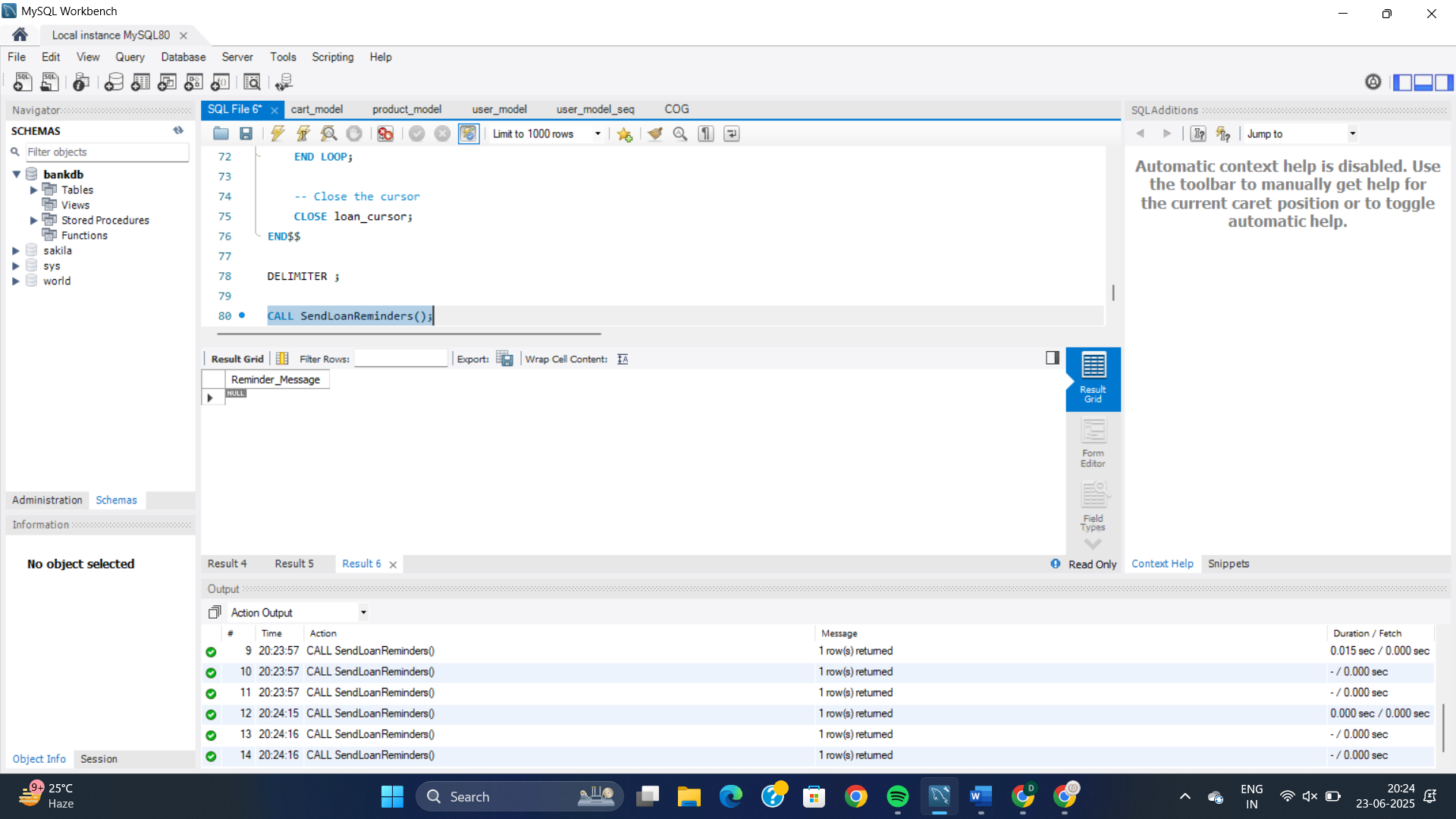
) AS Reminder\_Message;

END LOOP;

CLOSE loan\_cursor;

END$$

DELIMITER ;



**QUESTION 2:** **Exercise 3: Stored Procedures**

**Scenario 1:** The bank needs to process monthly interest for all savings accounts.

* + **Question:** Write a stored procedure **ProcessMonthlyInterest** that calculates and updates the balance of all savings accounts by applying an interest rate of 1% to the current balance.

**Scenario 2:** The bank wants to implement a bonus scheme for employees based on their performance.

* + **Question:** Write a stored procedure **UpdateEmployeeBonus** that updates the salary of employees in a given department by adding a bonus percentage passed as a parameter.

**Scenario 3:** Customers should be able to transfer funds between their accounts.

* + **Question:** Write a stored procedure **TransferFunds** that transfers a specified amount from one account to another, checking that the source account has sufficient balance before making the transfer.

**Scenario 1:SOLUTION:**

create database interest;

use interest;

CREATE TABLE accounts (

account\_id INT PRIMARY KEY,

account\_type VARCHAR(20),

balance DECIMAL(10,2)

);

INSERT INTO accounts VALUES

(101, 'savings', 10000.00),

(102, 'current', 15000.00),

(103, 'savings', 20000.00);

DELIMITER $$

CREATE PROCEDURE ProcessMonthlyInterest()

BEGIN

UPDATE accounts

SET balance = balance \* 1.01

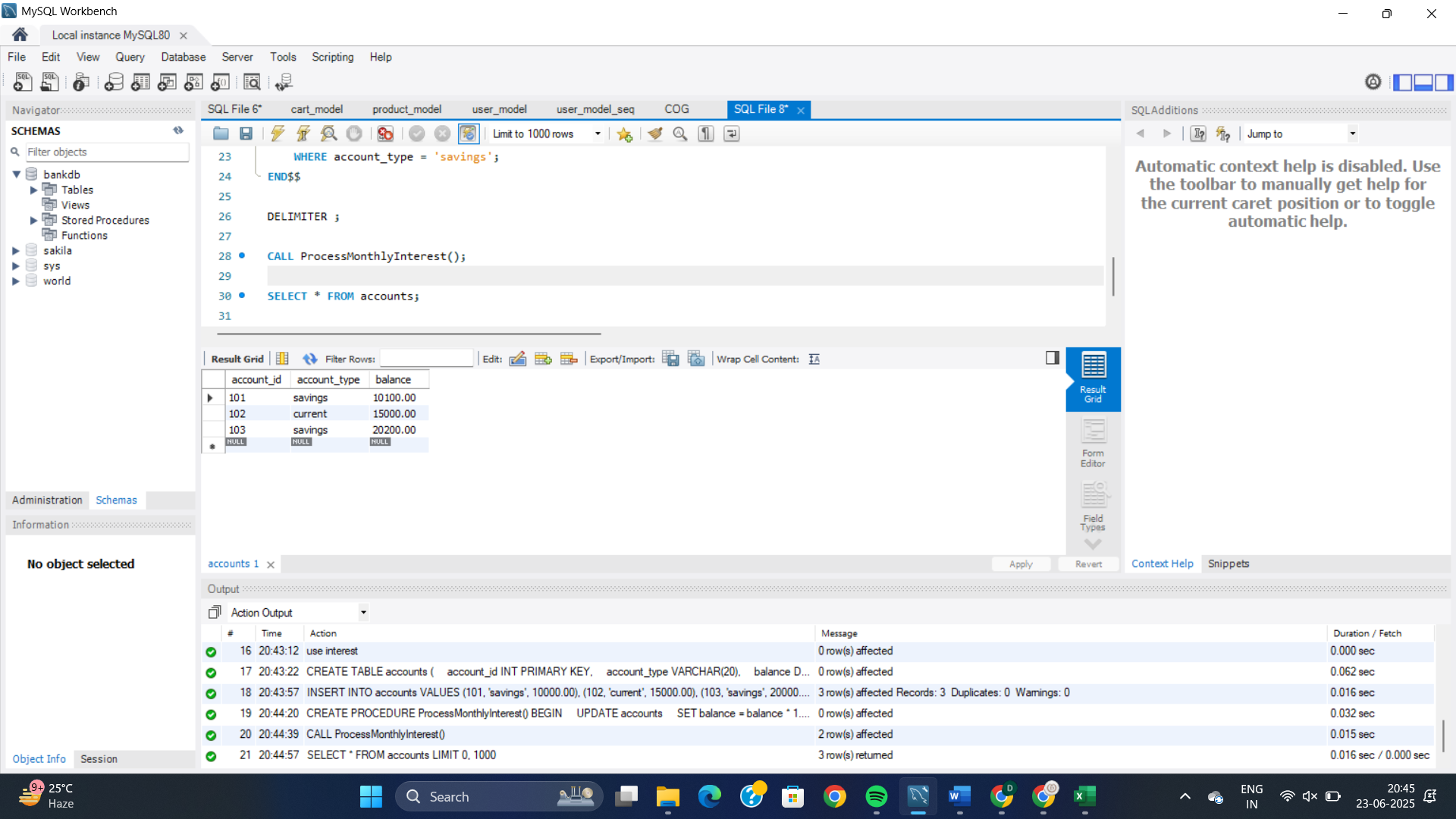
WHERE account\_type = 'savings';

END$$

DELIMITER ;

CALL ProcessMonthlyInterest();

SELECT \* FROM accounts;



**Scenario 2: SOLUTION:**

CREATE TABLE employees (

employee\_id INT PRIMARY KEY,

name VARCHAR(100),

department VARCHAR(50),

salary DECIMAL(10,2)

);

INSERT INTO employees VALUES

(1, 'Alice', 'Sales', 50000.00),

(2, 'Bob', 'IT', 60000.00),

(3, 'Charlie', 'Sales', 55000.00);

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)

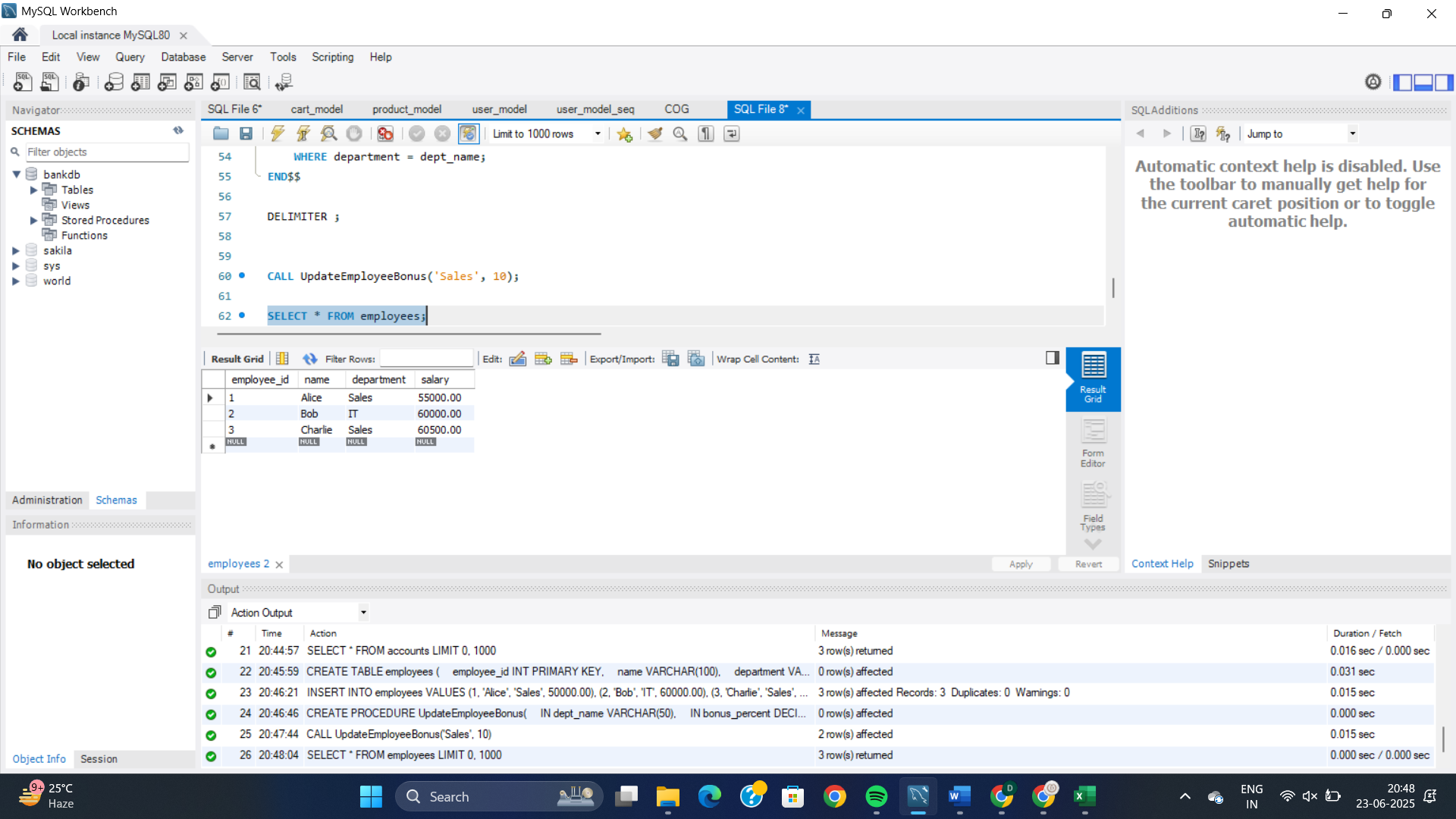
WHERE department = dept\_name;

END$$

DELIMITER ;

CALL UpdateEmployeeBonus('Sales', 10);

SELECT \* FROM employees;



**Scenario 3: SOLUTION:**

UPDATE accounts SET balance = 10000 WHERE account\_id = 101;

UPDATE accounts SET balance = 5000 WHERE account\_id = 102;

DELIMITER $$

CREATE PROCEDURE TransferFunds(

IN from\_account INT,

IN to\_account INT,

IN transfer\_amount DECIMAL(10,2)

)

BEGIN

DECLARE from\_balance DECIMAL(10,2);

-- Get current balance of source account

SELECT balance INTO from\_balance

FROM accounts

WHERE account\_id = from\_account;

-- Check if balance is enough

IF from\_balance >= transfer\_amount THEN

-- Deduct from source

UPDATE accounts

SET balance = balance - transfer\_amount

WHERE account\_id = from\_account;

-- Add to target

UPDATE accounts

SET balance = balance + transfer\_amount

WHERE account\_id = to\_account;

ELSE

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'Insufficient funds in source account.';

END IF;

END$$

DELIMITER ;

CALL TransferFunds(101, 102, 2000.00);

SELECT \* FROM accounts;

