**Exercise 1: Control Structures**

**Scenario 1**

-- Use your database

USE BankDB;

-- Drop procedure if already exists

DROP PROCEDURE IF EXISTS ApplyInterestDiscount;

-- Create procedure

DELIMITER $$

CREATE PROCEDURE ApplyInterestDiscount()

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;

interest\_loop: LOOP

FETCH cur INTO cust\_id;

IF done THEN

LEAVE interest\_loop;

END IF;

UPDATE Loans

SET interest\_rate = interest\_rate - (interest\_rate \* 0.01)

WHERE customer\_id = cust\_id;

END LOOP;

CLOSE cur;

END $$

DELIMITER ;

-- Turn off safe update mode (required in some systems)

SET SQL\_SAFE\_UPDATES = 0;

-- Call the procedure

CALL ApplyInterestDiscount();

-- View updated loan interest rates

SELECT \* FROM Loans;

**Scenario 2**

-- Use your database

USE BankDB;

-- Drop procedure if already exists

DROP PROCEDURE IF EXISTS PromoteToVIP;

-- Create procedure

DELIMITER $$

CREATE PROCEDURE PromoteToVIP()

BEGIN

UPDATE Customers

SET is\_vip = TRUE

WHERE balance > 10000;

END $$

DELIMITER ;

-- Turn off safe update mode (required in some systems)

SET SQL\_SAFE\_UPDATES = 0;

-- Call the procedure

CALL PromoteToVIP();

-- View updated customer VIP status

SELECT \* FROM Customers;

**Scenario 3**

-- Use your database

USE BankDB;

-- Step 1: Drop old procedure if it exists

DROP PROCEDURE IF EXISTS SendLoanReminders;

-- Step 2: Create the procedure

DELIMITER $$

CREATE PROCEDURE SendLoanReminders()

BEGIN

DECLARE done INT DEFAULT FALSE;

DECLARE cust\_name VARCHAR(100);

DECLARE due\_date DATE;

-- Cursor to find loans due in next 30 days

DECLARE cur CURSOR FOR

SELECT c.name, l.due\_date

FROM Customers c

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

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

DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

OPEN cur;

reminder\_loop: LOOP

FETCH cur INTO cust\_name, due\_date;

IF done THEN

LEAVE reminder\_loop;

END IF;

-- Simulated output using SELECT

SELECT CONCAT('Reminder: ', cust\_name, ', your loan is due on ', due\_date) AS Reminder\_Message;

END LOOP;

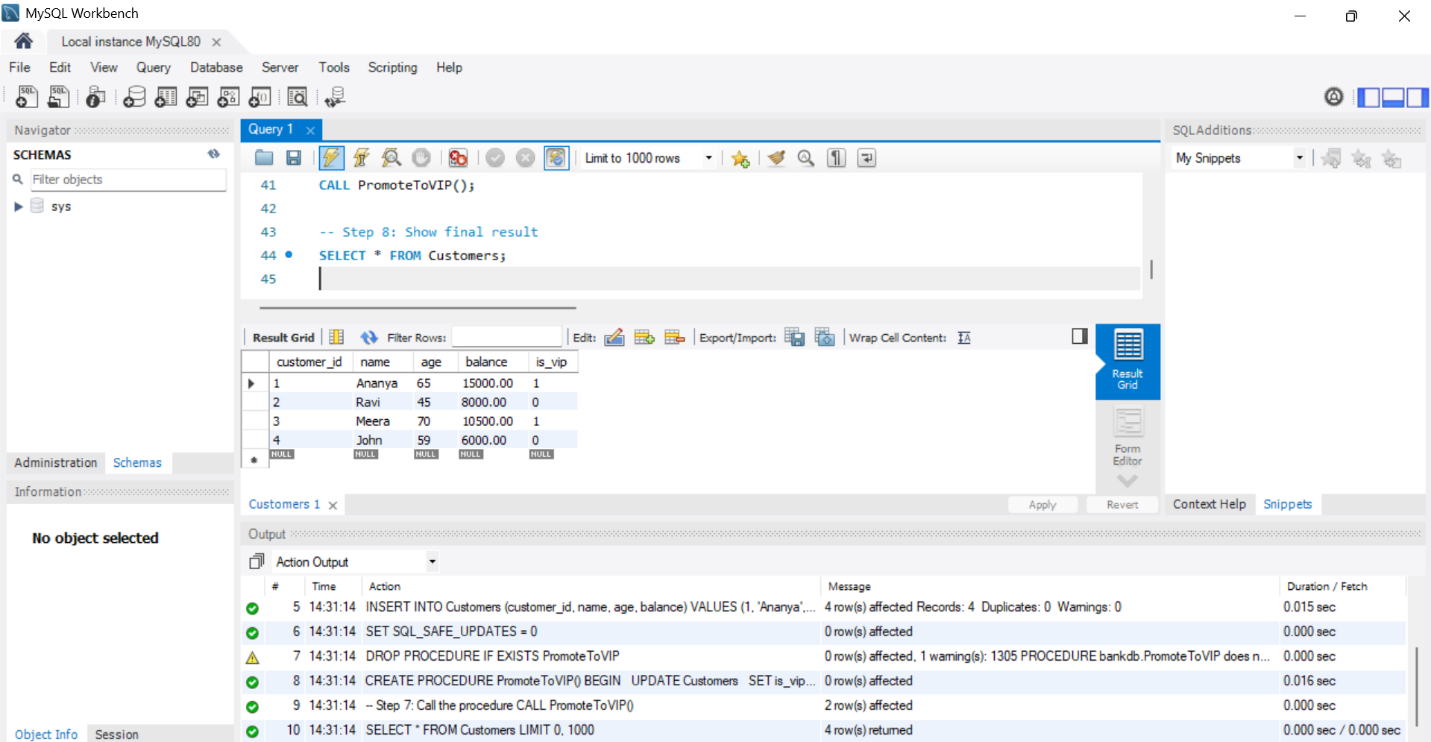
CLOSE cur;

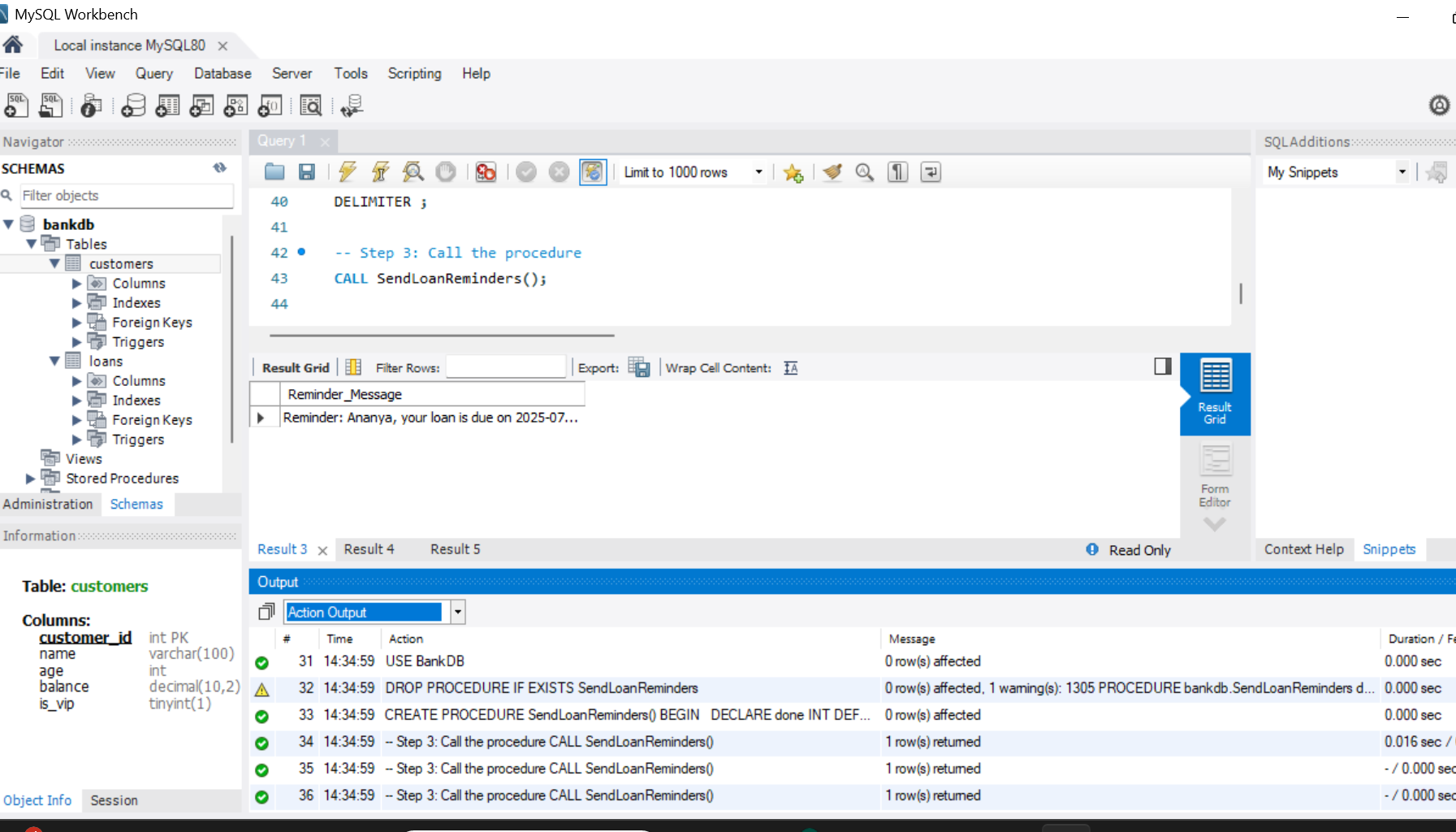
END $$

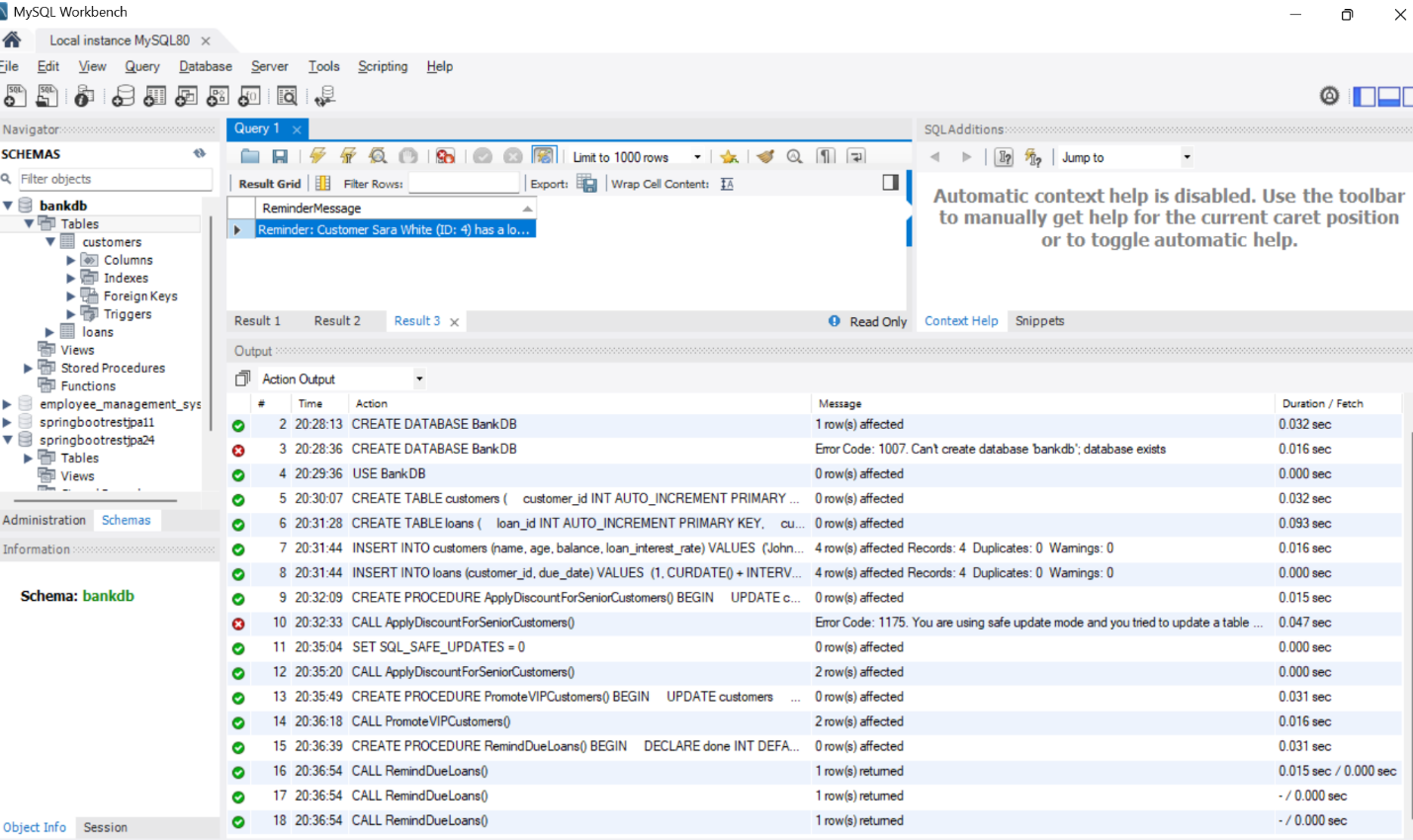
DELIMITER ;

-- Step 3: Call the procedure

CALL SendLoanReminders();**Output:**

****

****

****

**Exercise 3: Stored Procedures**

**Scenario 1:**

USE BankDB;

DROP PROCEDURE IF EXISTS ProcessMonthlyInterest;

DELIMITER $$

CREATE PROCEDURE ProcessMonthlyInterest()

BEGIN

UPDATE Accounts

SET balance = balance + (balance \* 0.01)

WHERE account\_type = 'savings';

END $$

DELIMITER ;

CALL ProcessMonthlyInterest();

SELECT \* FROM Accounts;

**Scenario 2:**

USE BankDB;

DROP PROCEDURE IF EXISTS UpdateEmployeeBonus;

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:**

**USE BankDB;**

**DROP PROCEDURE IF EXISTS TransferFunds;**

**DELIMITER $$**

**CREATE PROCEDURE TransferFunds(**

**IN from\_account INT,**

**IN to\_account INT,**

**IN amount DECIMAL(10,2)**

**)**

**BEGIN**

**DECLARE from\_balance DECIMAL(10,2);**

**SELECT balance INTO from\_balance**

**FROM Accounts**

**WHERE account\_id = from\_account;**

**IF from\_balance >= amount THEN**

**UPDATE Accounts**

**SET balance = balance - amount**

**WHERE account\_id = from\_account;**

**UPDATE Accounts**

**SET balance = balance + amount**

**WHERE account\_id = to\_account;**

**ELSE**

**SIGNAL SQLSTATE '45000'**

**SET MESSAGE\_TEXT = 'Insufficient balance in source account';**

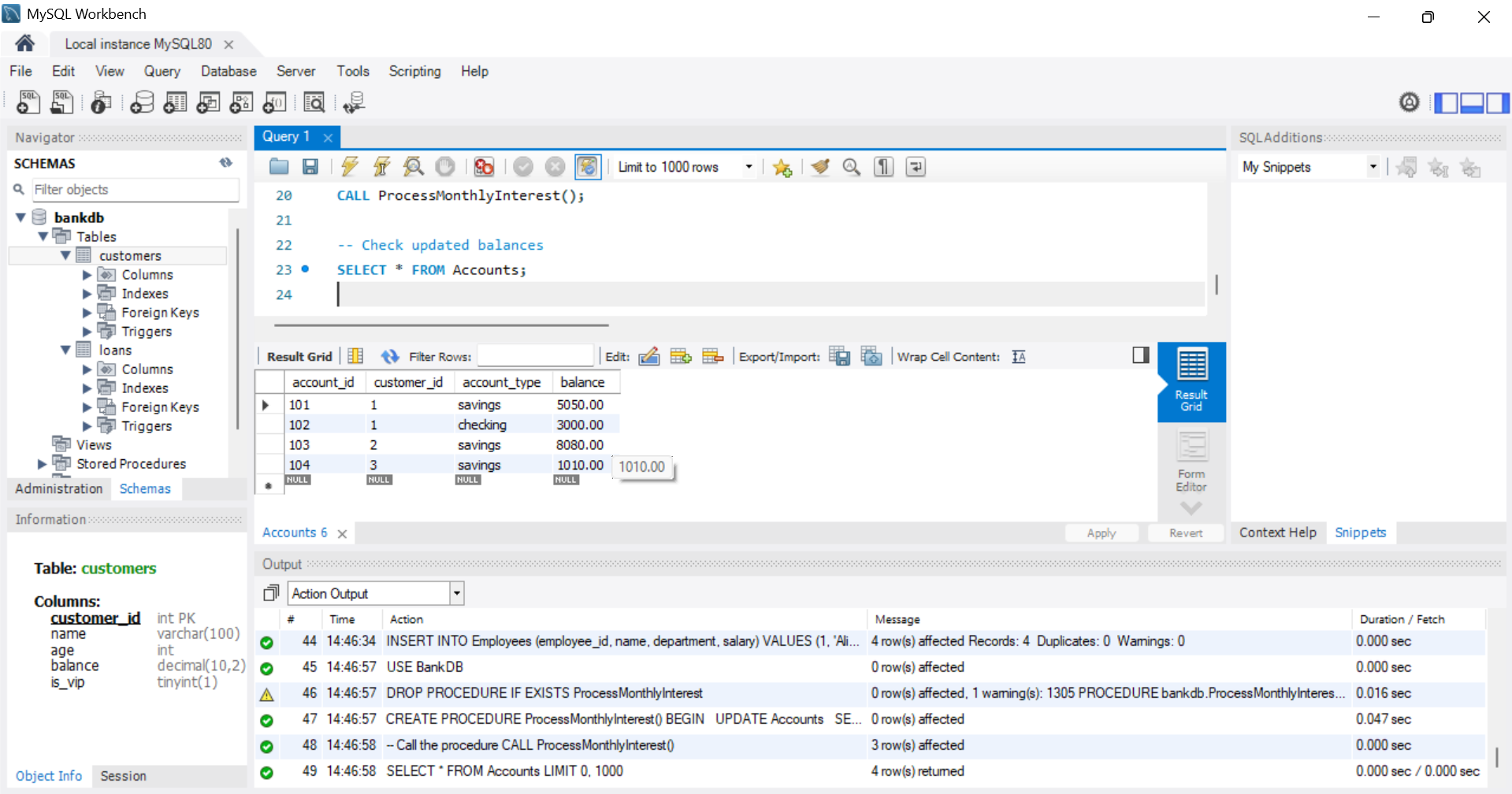
**END IF;**

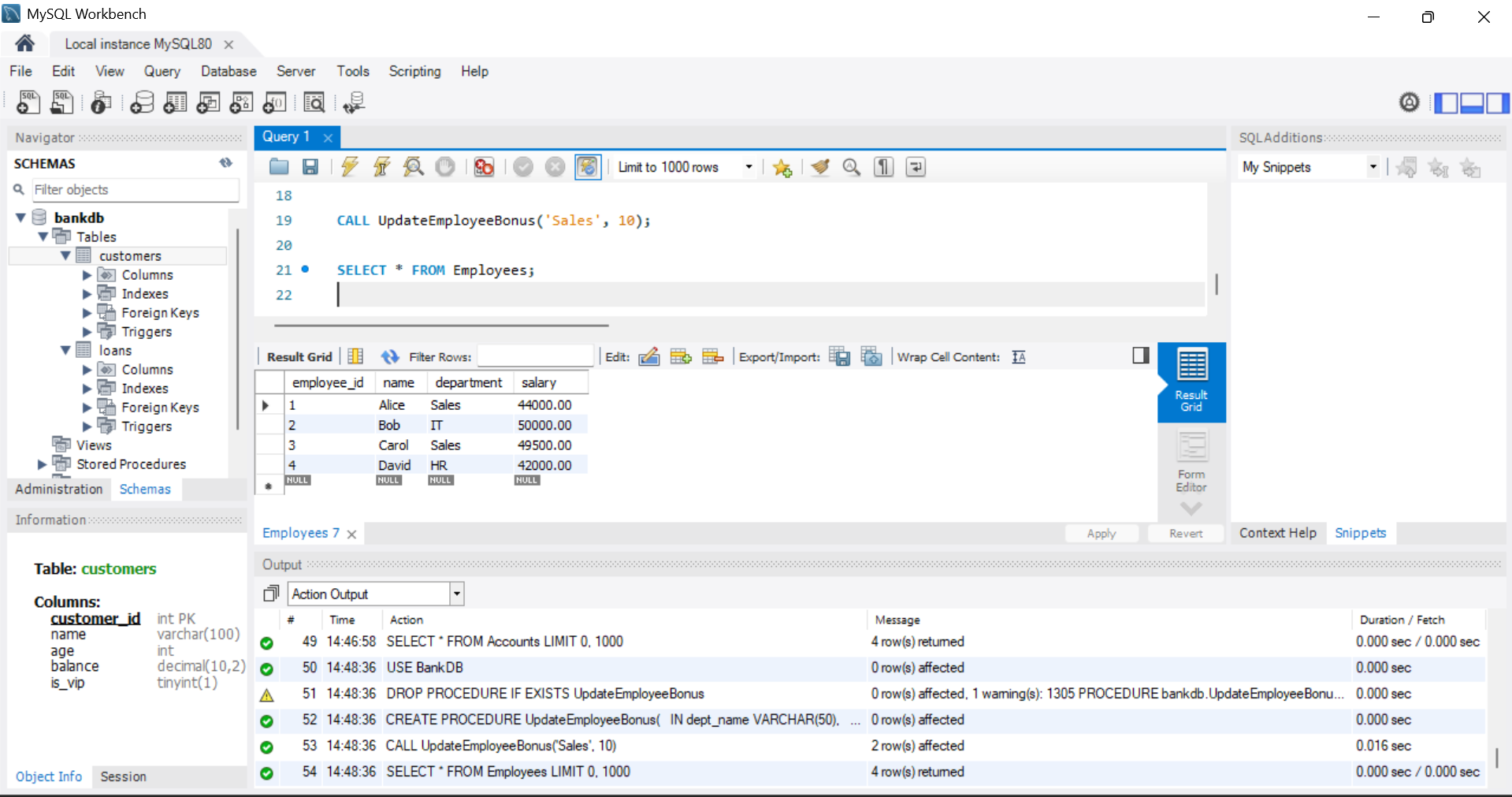
**END $$**

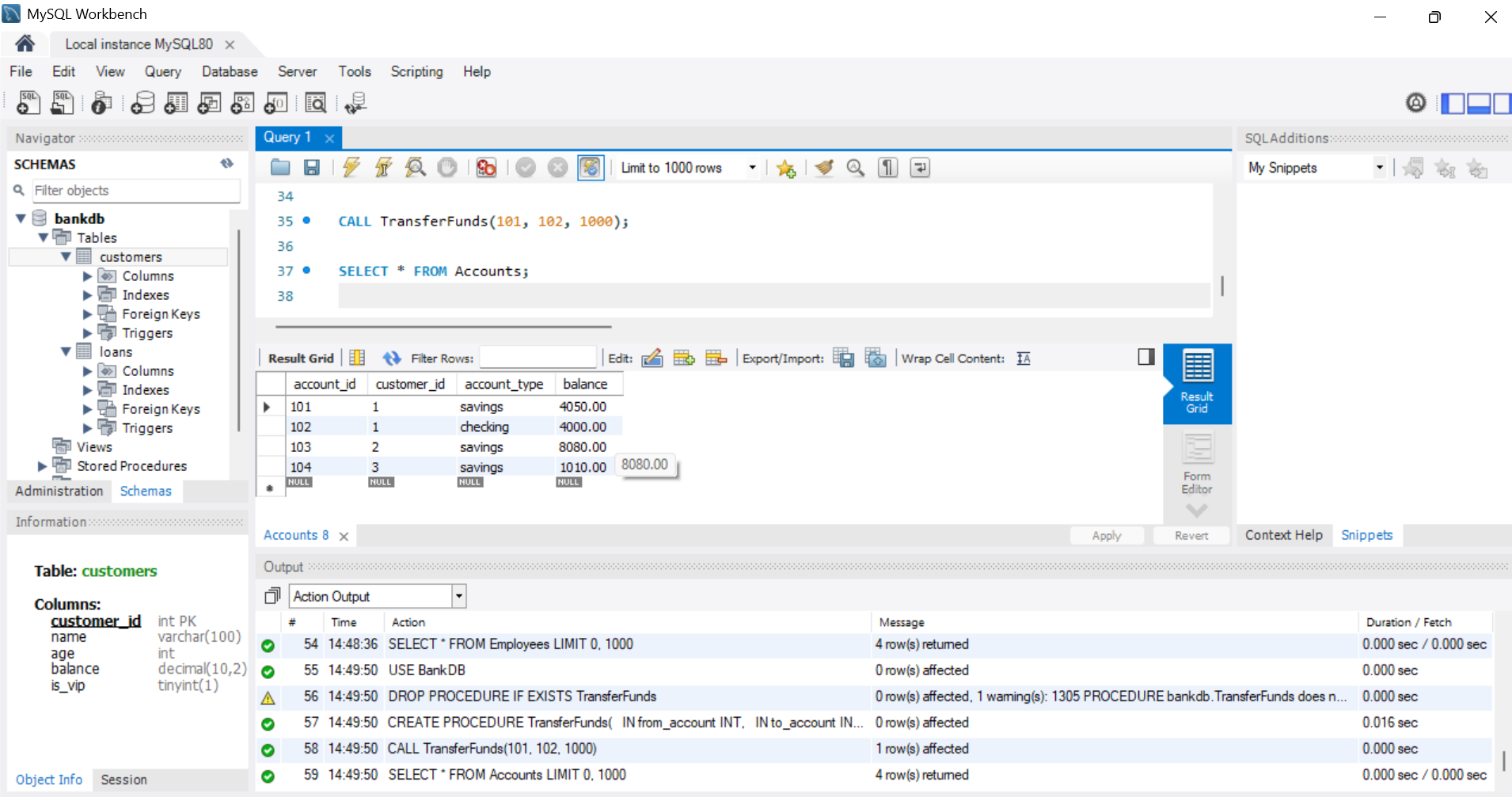
**DELIMITER ;**

**CALL TransferFunds(101, 102, 1000);**

**SELECT \* FROM Accounts;Output:**

****

****

****