**Exercise 1:**

**Scenario 1:**

DELIMITER //

CREATE PROCEDURE ApplySeniorDiscount()

BEGIN

DECLARE done INT DEFAULT 0;

DECLARE vCustomerID INT;

DECLARE vDOB DATE;

DECLARE vAge INT;

DECLARE cur CURSOR FOR SELECT CustomerID, DOB FROM Customers;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;

OPEN cur;

read\_loop: LOOP

FETCH cur INTO vCustomerID, vDOB;

IF done THEN

LEAVE read\_loop;

END IF;

SET vAge = TIMESTAMPDIFF(YEAR, vDOB, CURDATE());

IF vAge > 60 THEN

UPDATE Loans

SET InterestRate = InterestRate - 1

WHERE CustomerID = vCustomerID;

END IF;

END LOOP;

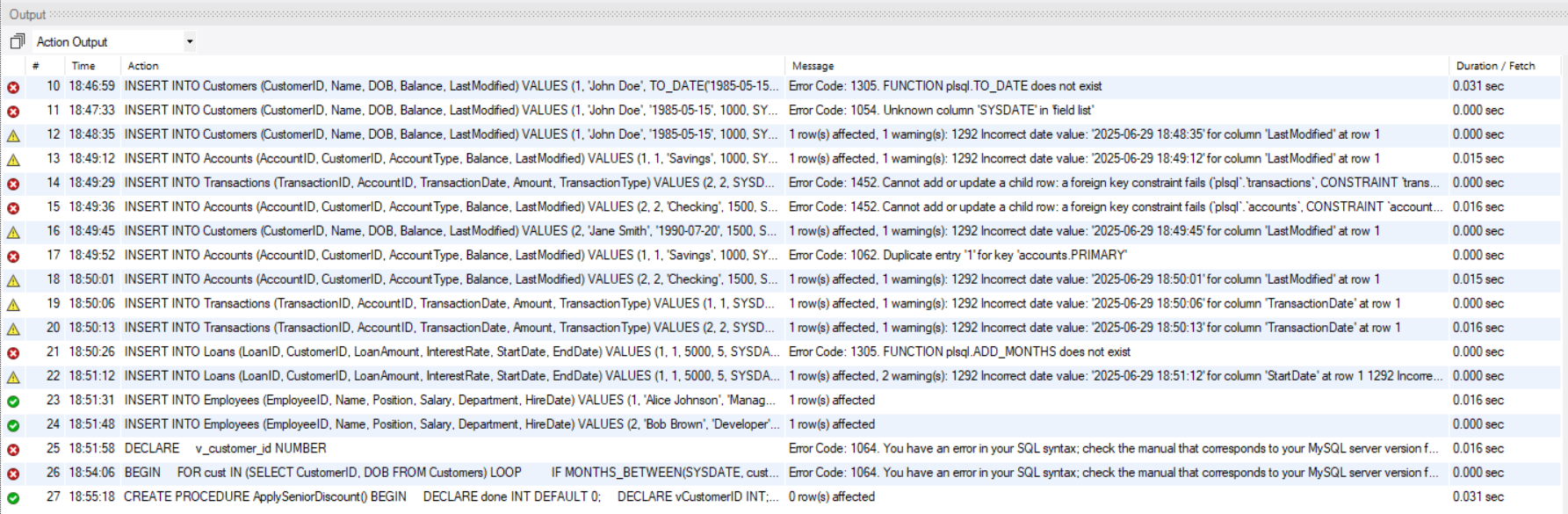
CLOSE cur;

END;

//

DELIMITER ;

**Output :**



**Scenario 2:**

DELIMITER //

CREATE PROCEDURE PromoteVIPCustomers()

BEGIN

DECLARE done INT DEFAULT 0;

DECLARE vCustomerID INT;

DECLARE vBalance DECIMAL(10,2);

DECLARE cur CURSOR FOR

SELECT CustomerID, Balance FROM Customers;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;

OPEN cur;

read\_loop: LOOP

FETCH cur INTO vCustomerID, vBalance;

IF done THEN

LEAVE read\_loop;

END IF;

IF vBalance > 10000 THEN

UPDATE Customers

SET IsVIP = TRUE

WHERE CustomerID = vCustomerID;

END IF;

END LOOP;

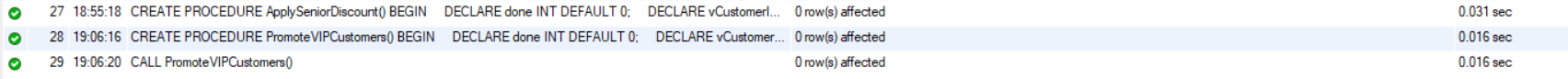
CLOSE cur;

END;

//

DELIMITER ;

CALL PromoteVIPCustomers();

**Output :**

**Scenario 3:**

DELIMITER //

CREATE PROCEDURE SendLoanReminders()

BEGIN

DECLARE done INT DEFAULT 0;

DECLARE vLoanID INT;

DECLARE vCustomerID INT;

DECLARE vEndDate DATE;

DECLARE cur CURSOR FOR

SELECT LoanID, CustomerID, EndDate

FROM Loans

WHERE EndDate BETWEEN CURDATE() AND DATE\_ADD(CURDATE(), INTERVAL 30 DAY);

DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;

OPEN cur;

read\_loop: LOOP

FETCH cur INTO vLoanID, vCustomerID, vEndDate;

IF done THEN

LEAVE read\_loop;

END IF;

-- Print message (visible in MySQL Workbench or CLI)

SELECT CONCAT('Reminder: Loan ID ', vLoanID,

' for Customer ID ', vCustomerID,

' is due on ', DATE\_FORMAT(vEndDate, '%d-%b-%Y')) AS Reminder;

END LOOP;

CLOSE cur;

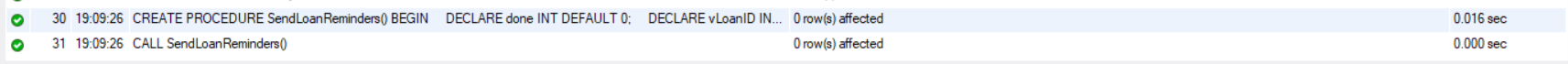
END;

//

DELIMITER ;

CALL SendLoanReminders();

**Output :**



**Exercise 3:**

**Scenario 1:**

DELIMITER //

CREATE PROCEDURE ProcessMonthlyInterest()

BEGIN

UPDATE Accounts

SET Balance = Balance \* 1.01

WHERE AccountType = 'Savings';

END;

//

DELIMITER ;

**Output :**

**Scenario 2:**

CREATE PROCEDURE UpdateEmployeeBonus(

IN dept\_name VARCHAR(100),

IN bonus\_percent DECIMAL(5,2)

)

BEGIN

UPDATE Employees

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

WHERE Department = dept\_name;

END;

//

DELIMITER ;

**Output :**

**Scenario 3:**

DELIMITER //

CREATE PROCEDURE UpdateEmployeeBonus(

IN dept\_name VARCHAR(100),

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);

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 AccountID = from\_account;

-- Check if sufficient funds

IF from\_balance >= amount THEN

-- Deduct from source

UPDATE Accounts

SET Balance = Balance - amount

WHERE AccountID = from\_account;

-- Add to target

UPDATE Accounts

SET Balance = Balance + amount

WHERE AccountID = to\_account;

ELSE

SIGNAL SQLSTATE '45000'

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

END IF;

END;

//

DELIMITER ;

**Output :**

