**PL/SQL programming Mandatory Hands-On**

**Exercise 1: Control Structures**

**Query:**

**Scenario 1:**

DELIMITER //

CREATE PROCEDURE ApplySeniorInterestDiscount()

BEGIN

UPDATE Loans

JOIN Customers ON Loans.CustomerID = Customers.CustomerID SET Loans.InterestRate Loans.InterestRate – 1

WHERE TIMESTAMPDIFF(YEAR, Customers.DOB, CURDATE()) > 60;

END;

//

DELIMITER ;

CALL ApplySeniorInterestDiscount();

**Scenario: 2**

DELIMITER //

CREATE PROCEDURE PromoteVIPCustomers()

BEGIN

UPDATE Customers

SET IsVIP = 'Y'

WHERE Balance > 10000;

END;

//

DELIMITER ;

CALL PromoteVIPCustomers();

**Scenario: 3**

DELIMITER //

CREATE PROCEDURE ShowLoanReminders()

BEGIN

SELECT Customers.Name AS CustomerName,Loans.LoanID,Loans.EndDate,

CONCAT('Reminder: Loan ', Loans.LoanID, ' for ', Customers.Name,

' is due on ', DATE\_FORMAT(Loans.EndDate, '%Y-%m-%d')) AS Reminder

FROM Loans

JOIN Customers ON Loans.CustomerID = Customers.CustomerID

WHERE Loans.EndDate <= CURDATE() + INTERVAL 30 DAY;

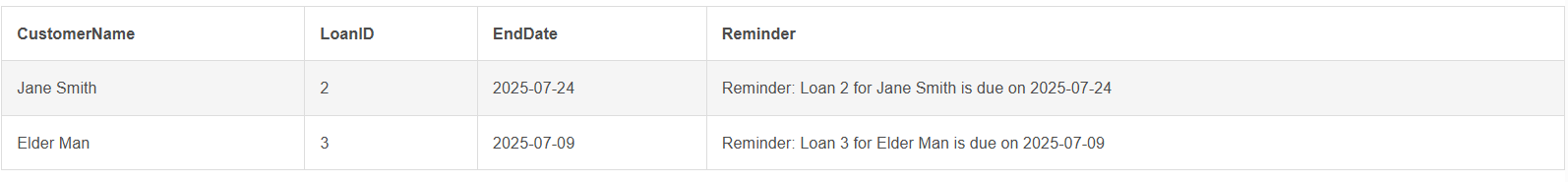
END;

//

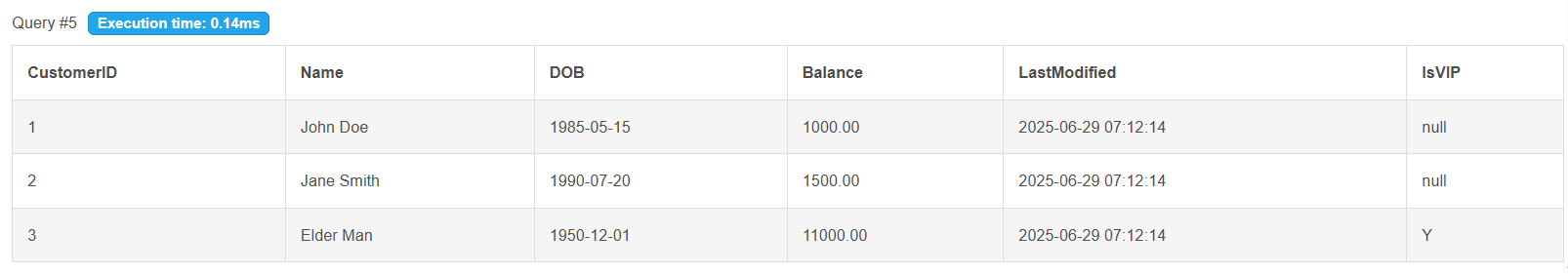
DELIMITER ;

CALL ShowLoanReminders();

**Final DB:**







**Exercise 3: Stored Procedures**

**Query:**

**Scenario: 1**

DROP PROCEDURE IF EXISTS ProcessMonthlyInterest;

CREATE PROCEDURE ProcessMonthlyInterest()

BEGIN

UPDATE Accounts

SET Balance = Balance + (Balance \* 0.01)

WHERE AccountType = 'Savings';

END;

CALL ProcessMonthlyInterest();

**Scenario: 2**

DROP PROCEDURE IF EXISTS UpdateEmployeeBonus;

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;

CALL UpdateEmployeeBonus('IT', 10);

**Scenario: 3**

DROP PROCEDURE IF EXISTS TransferFunds;

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 AccountID = from\_account;

IF from\_balance >= amount THEN

UPDATE Accounts SET Balance = Balance – amount WHERE AccountID = from\_account;

UPDATE Accounts SET Balance = Balance + amount WHERE AccountID = to\_account;

ELSE

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'Insufficient balance to transfer funds.';

END IF;

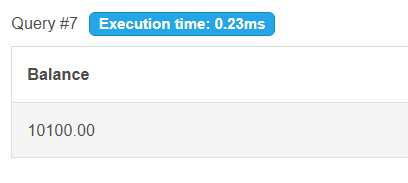
END;

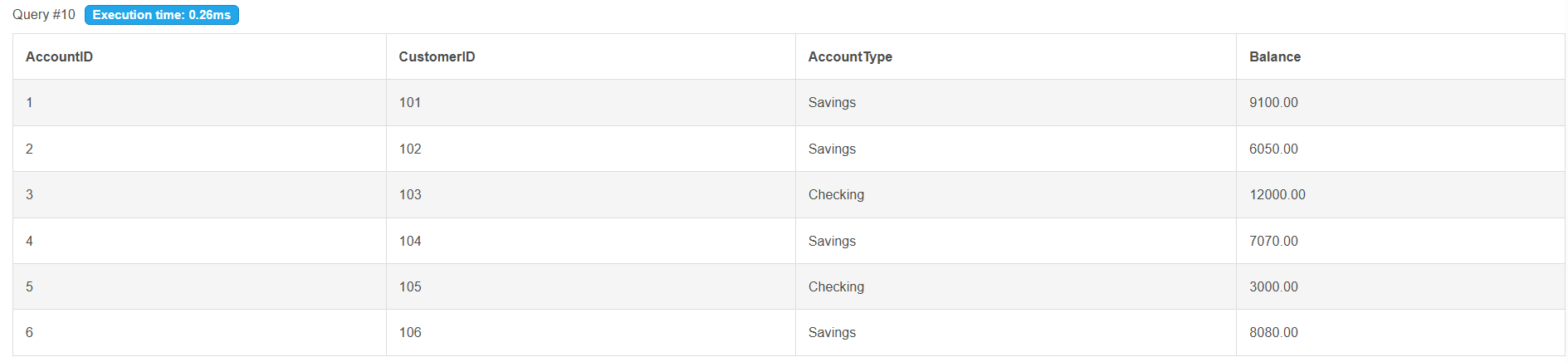
CALL TransferFunds(1, 2, 1000);

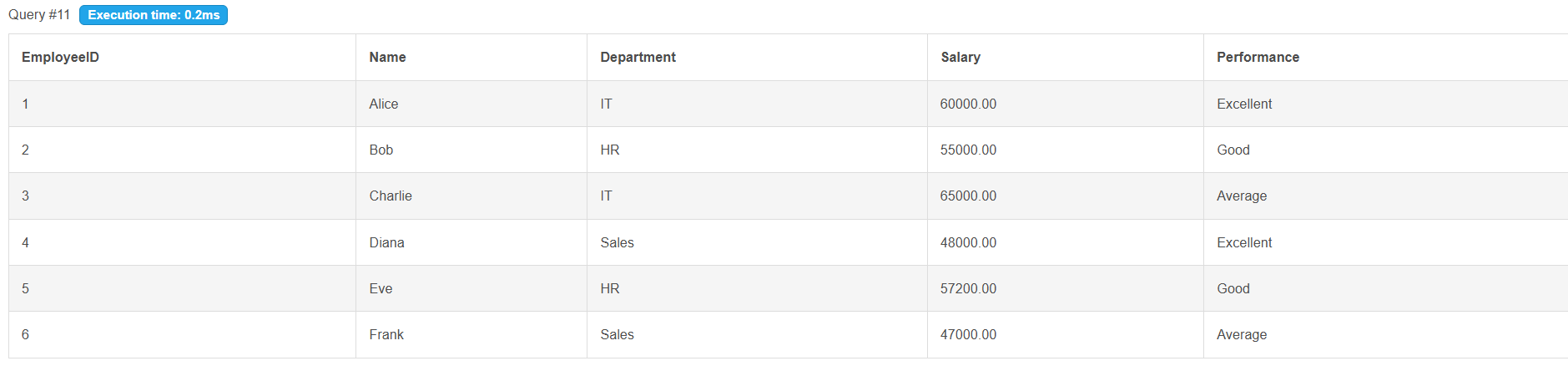
SELECT \* FROM Accounts;

SELECT \* FROM Employees;

**Output:**

****

****

****