**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.

**CODE:**

CREATE TABLE Customers (

CustomerID INT,

Name VARCHAR(50),

Age INT,

InterestRate DECIMAL(5,2)

);

INSERT INTO Customers VALUES (1, 'geetha', 65, 10.0);

INSERT INTO Customers VALUES (2, 'hari', 45, 12.0);

INSERT INTO Customers VALUES (3, 'Charmy', 70, 11.5);

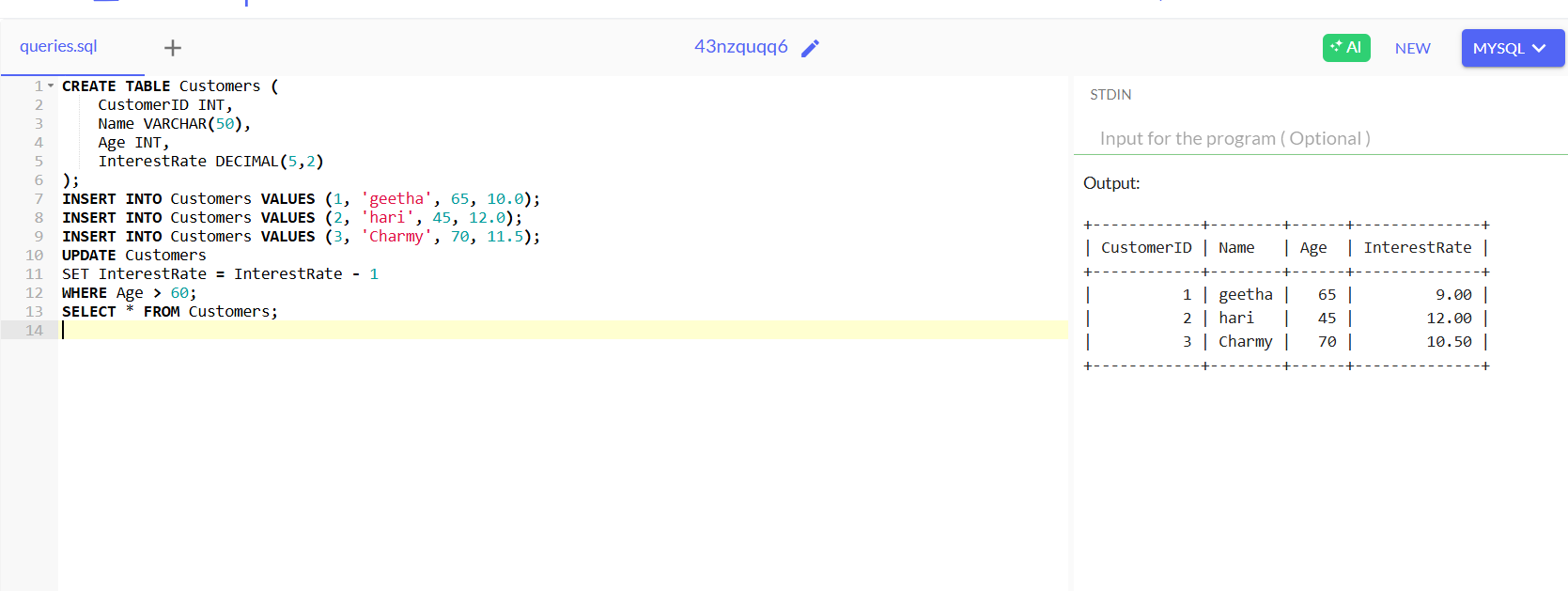
UPDATE Customers

SET InterestRate = InterestRate - 1

WHERE Age > 60;

SELECT \* FROM Customers;

**OUTPUT:**

****

**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.

**CODE:**

CREATE TABLE Customers (

CustomerID INT,

Name VARCHAR(50),

Balance DECIMAL(10,2),

IsVIP BOOLEAN

);

INSERT INTO Customers VALUES (1, 'Geetha', 12000.00, FALSE);

INSERT INTO Customers VALUES (2, 'Tharun', 9000.00, FALSE);

INSERT INTO Customers VALUES (3, 'Charmy', 15000.00, FALSE);

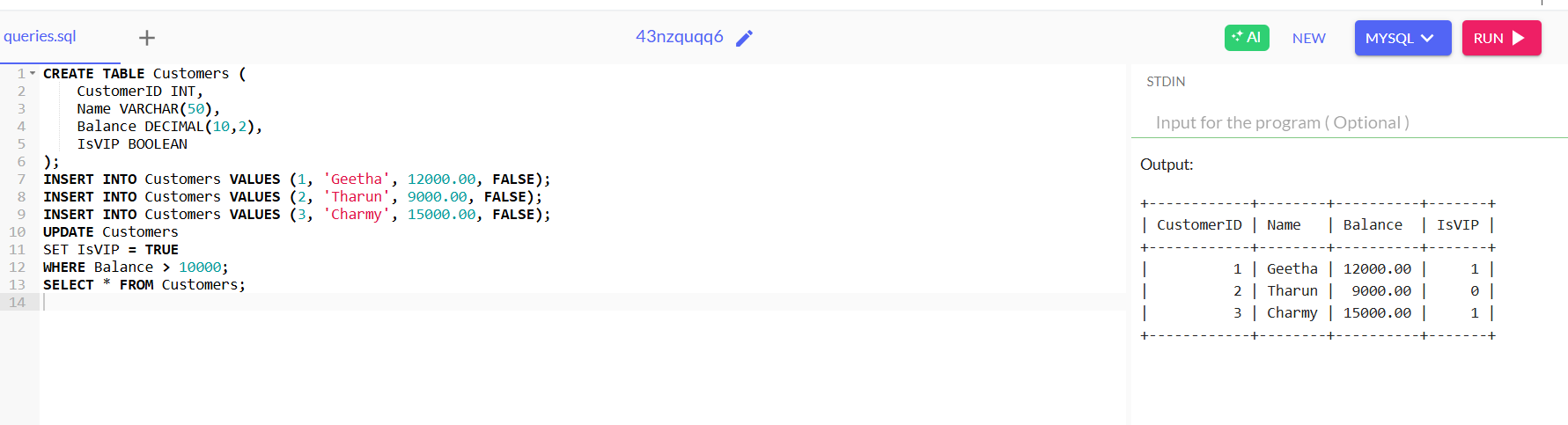
UPDATE Customers

SET IsVIP = TRUE

WHERE Balance > 10000;

SELECT \* FROM Customers;

**OUTPUT:**

****

**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.

**CODE:**

CREATE TABLE Customers(CustomerID INT,Name VARCHAR(50));

CREATE TABLE Loans(LoanID INT,CustomerID INT,DueDate DATE);

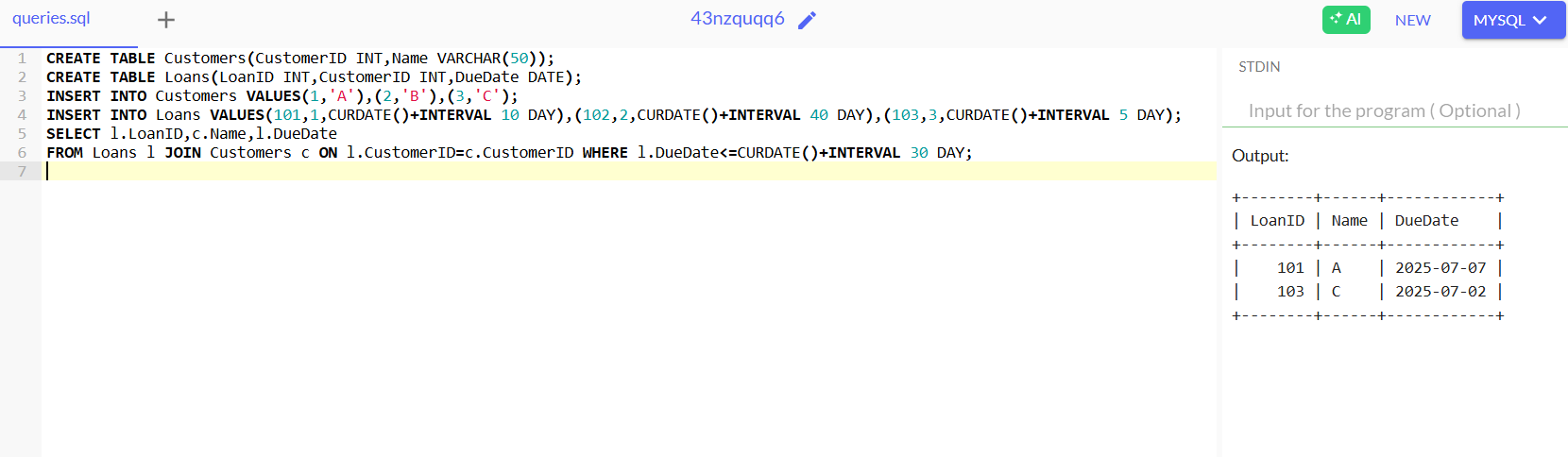
INSERT INTO Customers VALUES(1,'A'),(2,'B'),(3,'C');

INSERT INTO Loans VALUES(101,1,CURDATE()+INTERVAL 10 DAY),(102,2,CURDATE()+INTERVAL 40 DAY),(103,3,CURDATE()+INTERVAL 5 DAY);

SELECT l.LoanID,c.Name,l.DueDate

FROM Loans l JOIN Customers c ON l.CustomerID=c.CustomerID WHERE l.DueDate<=CURDATE()+INTERVAL 30 DAY;

**OUTPUT:**

****

**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.

**CODE:**

CREATE TABLE SavingsAccounts(AccountID INT,Balance DECIMAL(10,2));

INSERT INTO SavingsAccounts VALUES(1,1000.00),(2,2500.00),(3,500.00);

DELIMITER //

CREATE PROCEDURE ProcessMonthlyInterest()

BEGIN

UPDATE SavingsAccounts SET Balance=Balance\*1.01;

END;

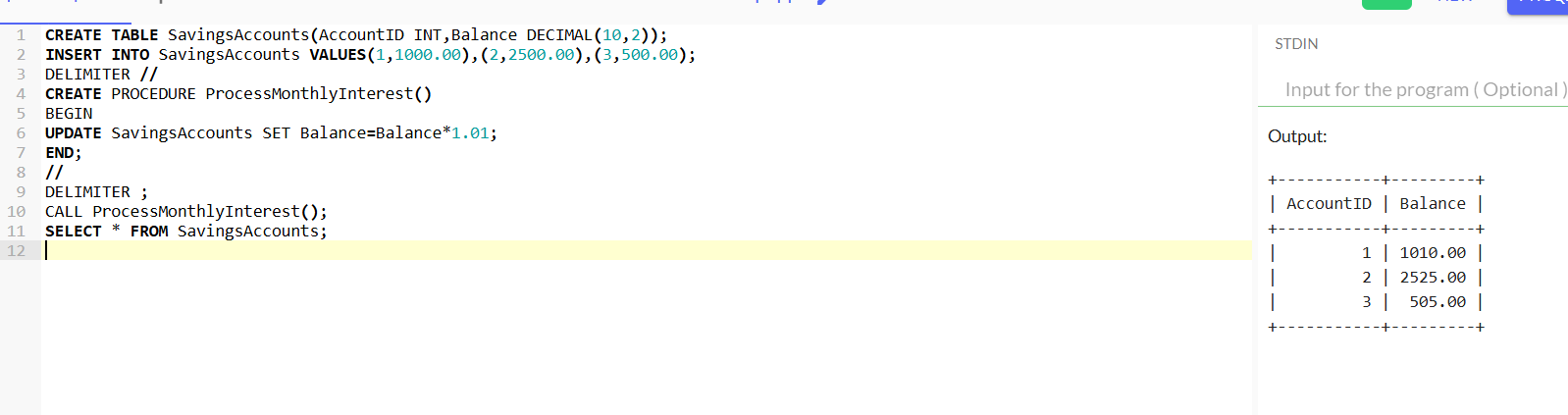
//

DELIMITER ;

CALL ProcessMonthlyInterest();

SELECT \* FROM SavingsAccounts;

**OUTPUT:**

****

**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.

**CODE:**

CREATE TABLE Employees(EmpID INT,Name VARCHAR(50),Department VARCHAR(50),Salary DECIMAL(10,2));

INSERT INTO Employees VALUES(1,'A','HR',50000),(2,'B','IT',60000),(3,'C','IT',55000);

DELIMITER //

CREATE PROCEDURE UpdateEmployeeBonus(IN dept VARCHAR(50),IN bonus DECIMAL(5,2))

BEGIN

UPDATE Employees SET Salary=Salary+(Salary\*bonus/100) WHERE Department=dept;

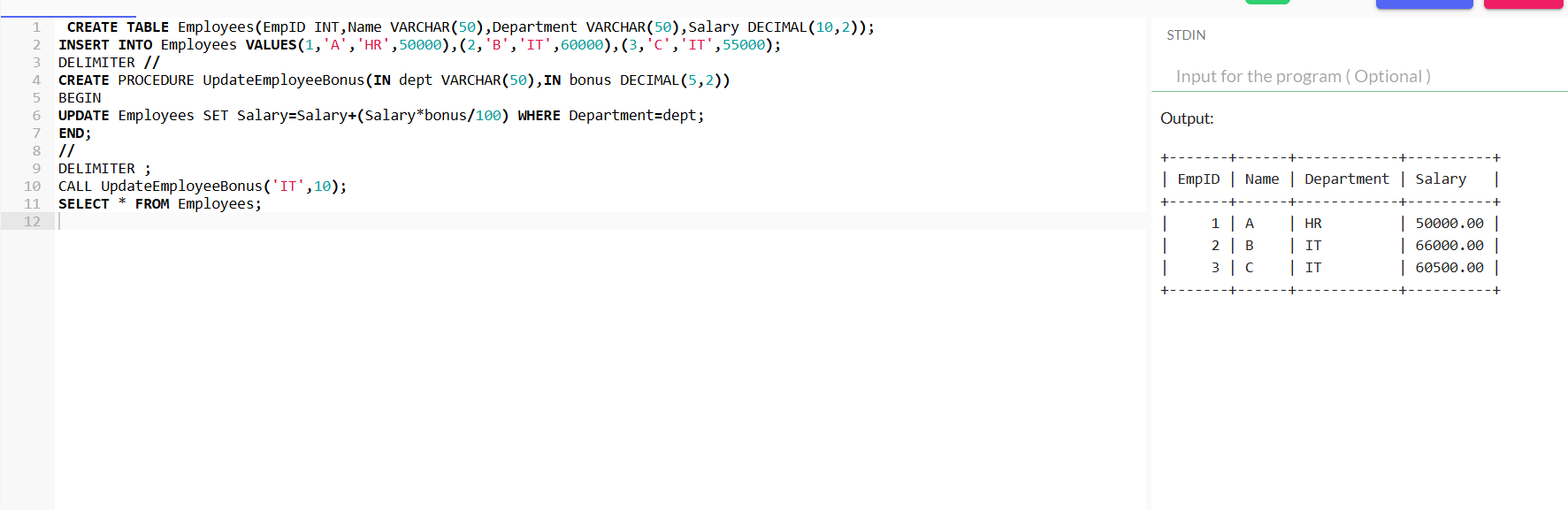
END;//

DELIMITER ;

CALL UpdateEmployeeBonus('IT',10);

SELECT \* FROM Employees;

**OUTPUT:**

****

**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.

**CODE:**

CREATE TABLE Accounts(AccountID INT,Balance DECIMAL(10,2));

INSERT INTO Accounts VALUES(1,1000.00),(2,500.00);

DELIMITER //

CREATE PROCEDURE TransferFunds(IN fromAcc INT,IN toAcc INT,IN amt DECIMAL(10,2))

BEGIN

DECLARE fromBal DECIMAL(10,2);

SELECT Balance INTO fromBal FROM Accounts WHERE AccountID=fromAcc;

IF fromBal>=amt THEN

UPDATE Accounts SET Balance=Balance-amt WHERE AccountID=fromAcc;

UPDATE Accounts SET Balance=Balance+amt WHERE AccountID=toAcc;

END IF;

END;

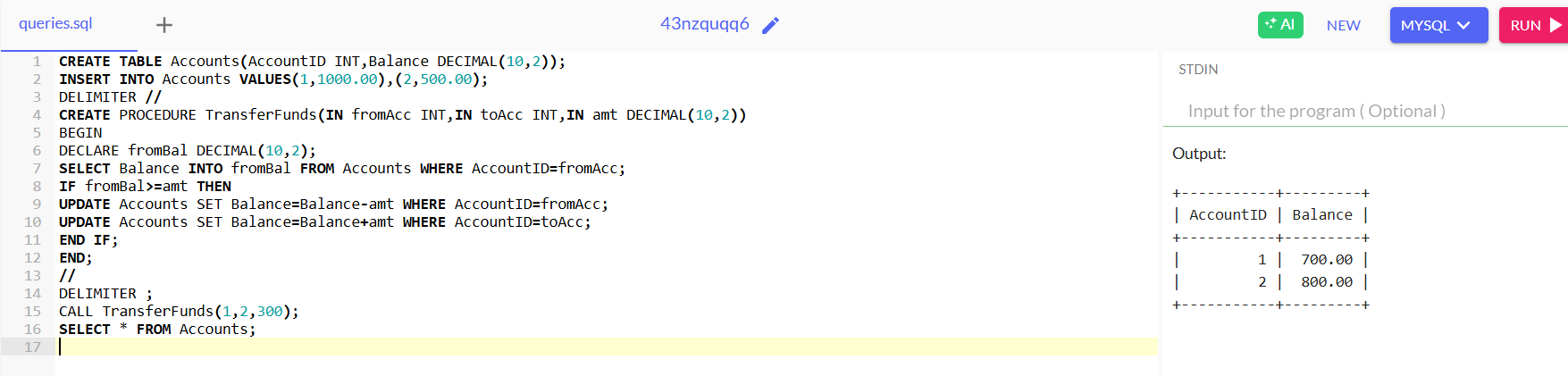
//

DELIMITER ;

CALL TransferFunds(1,2,300);

SELECT \* FROM Accounts;

**OUTPUT:**

****