**Exercise 1: Control Structures**

**CODE:**

**Table Creation:**

**Create Customers Table:**

CREATE TABLE Customers (

CustomerID NUMBER PRIMARY KEY,

CustomerName VARCHAR2(50),

Age NUMBER,

Balance NUMBER,

InterestRate NUMBER,

IsVIP VARCHAR2(5)

);

**Create Loans Table:**

CREATE TABLE Loans (

LoanID NUMBER PRIMARY KEY,

CustomerID NUMBER,

DueDate DATE,

FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)

);

**Insert Sample Data for Customers Table:**

INSERT INTO Customers VALUES (1, 'Alice', 65, 15000, 7.5, 'FALSE');

INSERT INTO Customers VALUES (2, 'Bob', 45, 9000, 8.0, 'FALSE');

INSERT INTO Customers VALUES (3, 'Charlie', 70, 12000, 6.5, 'FALSE');

**Insert Sample Data for Loans Table:**

INSERT INTO Loans VALUES (101, 1, SYSDATE + 10);

INSERT INTO Loans VALUES (102, 2, SYSDATE + 40);

INSERT INTO Loans VALUES (103, 3, SYSDATE + 25);

**Scenario 1: Apply 1 % Discount for Customers above 60**

BEGIN

FOR cust IN (SELECT CustomerID, InterestRate FROM Customers WHERE Age > 60) LOOP

UPDATE Customers

SET InterestRate = InterestRate - (InterestRate \* 0.01)

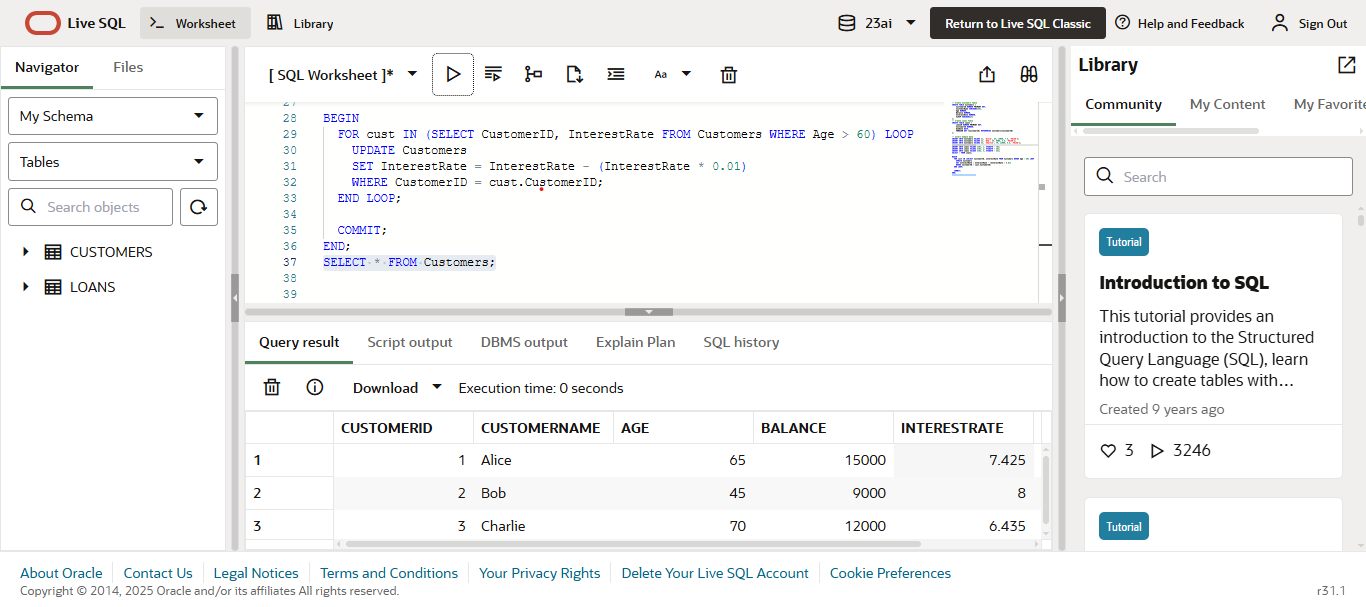
WHERE CustomerID = cust.CustomerID;

END LOOP;

COMMIT;

END;

**OUTPUT:**



**Scenario 2: Promote Customers to VIP status**

BEGIN

FOR cust IN (SELECT CustomerID FROM Customers WHERE Balance > 10000) LOOP

UPDATE Customers

SET IsVIP = 'TRUE'

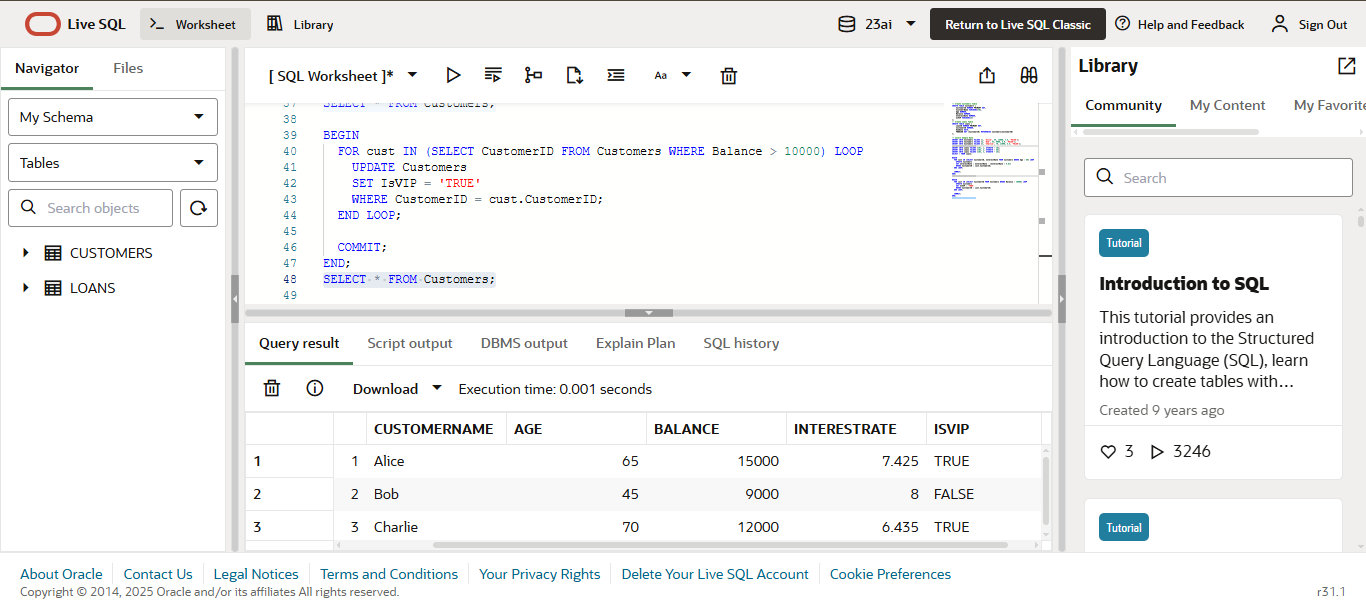
WHERE CustomerID = cust.CustomerID;

END LOOP;

COMMIT;

END;

**OUTPUT:**

****

**Scenario 3: Due Remainders**

DECLARE

v\_dueDate DATE;

v\_name VARCHAR2(100);

BEGIN

FOR loan IN (

SELECT c.CustomerName, l.DueDate

FROM Loans l

JOIN Customers c ON l.CustomerID = c.CustomerID

WHERE l.DueDate BETWEEN SYSDATE AND SYSDATE + 30

) LOOP

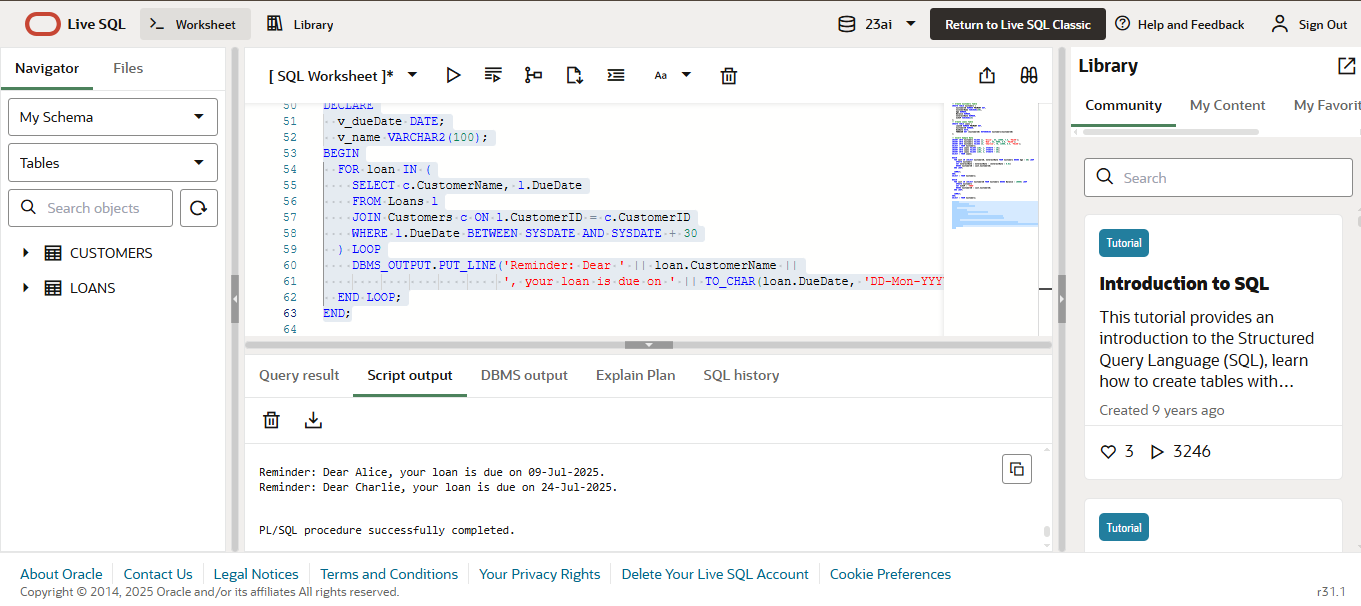
DBMS\_OUTPUT.PUT\_LINE('Reminder: Dear ' || loan.CustomerName ||

', your loan is due on ' || TO\_CHAR(loan.DueDate, 'DD-Mon-YYYY') || '.');

END LOOP;

END;

**OUTPUT:**

****

**Exercise 3: Stored Procedures**

**Scenario 1: Process Monthly Interest**

**CODE:**

-- Create Accounts Table

CREATE TABLE SavingsAccounts (

AccountID NUMBER PRIMARY KEY,

CustomerName VARCHAR2(50),

Balance NUMBER

);

-- Insert Sample Data

INSERT INTO SavingsAccounts VALUES (1, 'Alice', 10000);

INSERT INTO SavingsAccounts VALUES (2, 'Bob', 20000);

INSERT INTO SavingsAccounts VALUES (3, 'Charlie', 15000);

-- Stored Procedure to Apply 1% Interest

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest AS

BEGIN

FOR acc IN (SELECT AccountID, Balance FROM SavingsAccounts) LOOP

UPDATE SavingsAccounts

SET Balance = Balance + (Balance \* 0.01)

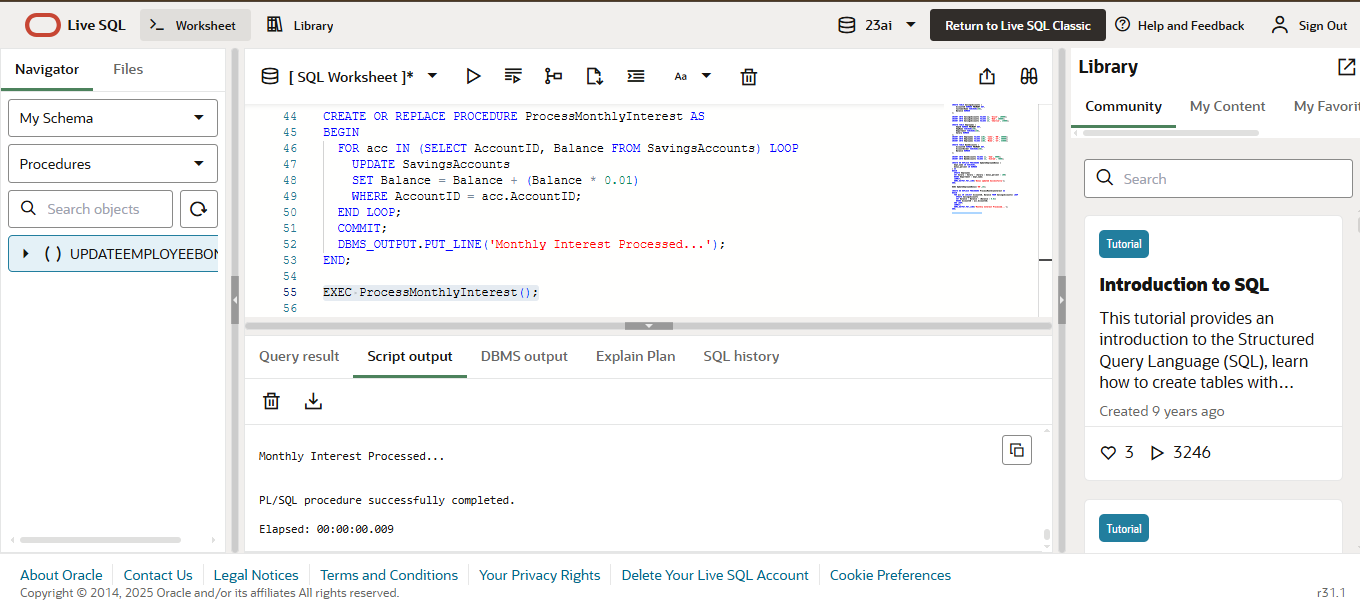
WHERE AccountID = acc.AccountID;

END LOOP;

COMMIT;

END;

**OUTPUT:**



**Scenario 2: Update Employee Bonus**

**CODE:**

-- Create Employees Table

CREATE TABLE Employees (

EmpID NUMBER PRIMARY KEY,

EmpName VARCHAR2(50),

Department VARCHAR2(30),

Salary NUMBER

);

-- Insert Sample Data

INSERT INTO Employees VALUES (101, 'John', 'HR', 30000);

INSERT INTO Employees VALUES (102, 'Jane', 'IT', 50000);

INSERT INTO Employees VALUES (103, 'Mike', 'IT', 45000);

-- Stored Procedure to Apply Bonus

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus (

dept\_name IN VARCHAR2,

bonus\_percent IN NUMBER

) AS

BEGIN

UPDATE Employees

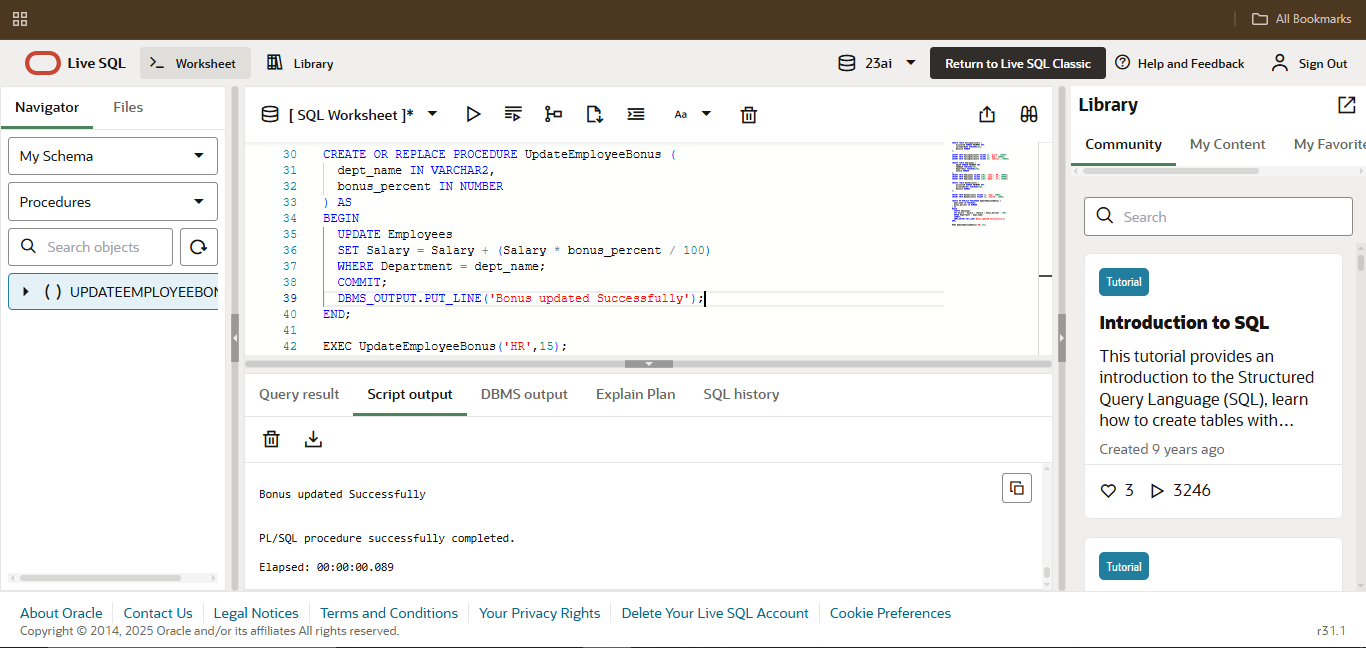
SET Salary = Salary + (Salary \* bonus\_percent / 100)

WHERE Department = dept\_name;

COMMIT;

END;

**OUTPUT:**

****

**Scenario 2: Transfer Funds**

**CODE:**

-- Create BankAccounts Table

CREATE TABLE BankAccounts (

AccountID NUMBER PRIMARY KEY,

AccountHolder VARCHAR2(50),

Balance NUMBER

);

-- Insert Sample Data

INSERT INTO BankAccounts VALUES (1, 'Sam', 5000);

INSERT INTO BankAccounts VALUES (2, 'Vikram', 7000);

-- Stored Procedure to Transfer Funds

CREATE OR REPLACE PROCEDURE TransferFunds (

from\_acc IN NUMBER,

to\_acc IN NUMBER,

amount IN NUMBER

) AS

v\_balance NUMBER;

BEGIN

-- Check balance of source account

SELECT Balance INTO v\_balance FROM BankAccounts WHERE AccountID = from\_acc;

IF v\_balance >= amount THEN

-- Deduct from source

UPDATE BankAccounts SET Balance = Balance - amount WHERE AccountID = from\_acc;

-- Add to destination

UPDATE BankAccounts SET Balance = Balance + amount WHERE AccountID = to\_acc;

COMMIT;

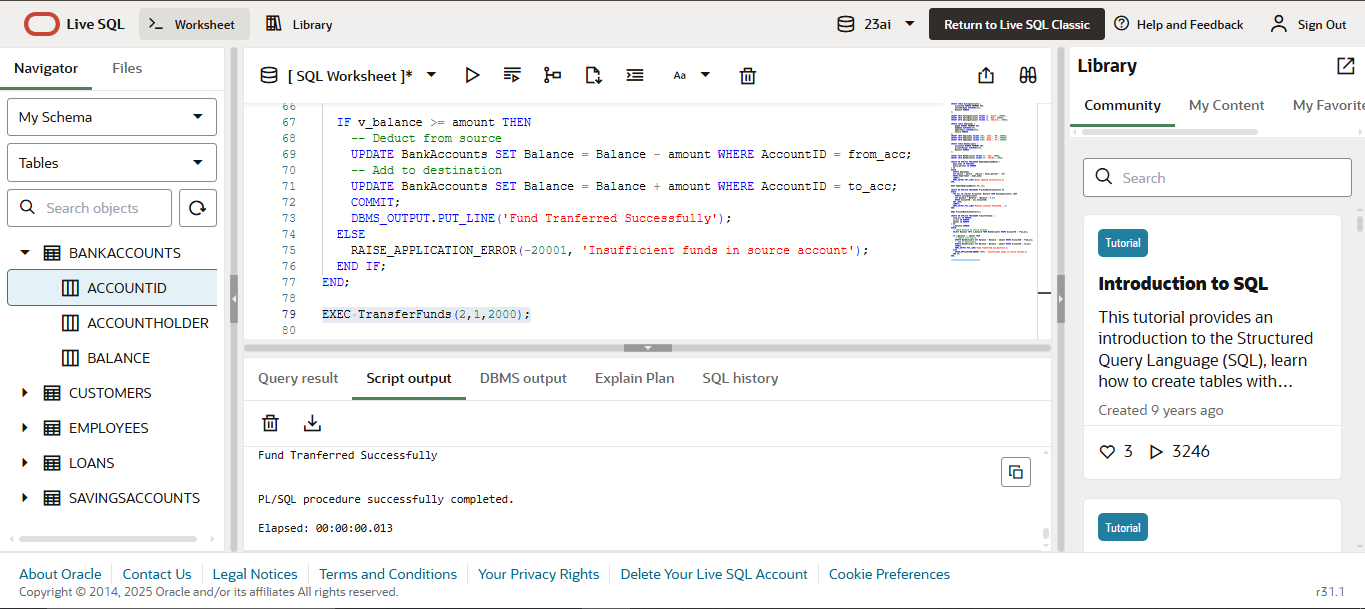
ELSE

RAISE\_APPLICATION\_ERROR(-20001, 'Insufficient funds in source account');

END IF;

END;

**OUTPUT:**

****