PLSQL programming

Exercise 1. Control Structures  
 **Scenario : 1  
  
Code :   
  
*-- Create Customers table***

CREATE TABLE Customers (

CustomerID NUMBER PRIMARY KEY,

CustomerName VARCHAR2(50),

Age NUMBER,

Balance NUMBER,

IsVIP VARCHAR2(5)

);

***-- Create Loans table***

CREATE TABLE Loans (

LoanID NUMBER PRIMARY KEY,

CustomerID NUMBER REFERENCES Customers(CustomerID),

InterestRate NUMBER,

DueDate DATE

);

***-- Insert sample customers***

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

INSERT INTO Customers VALUES (2, 'Bob', 45, 5000, 'FALSE');

INSERT INTO Customers VALUES (3, 'Charlie', 70, 8000, 'FALSE');

INSERT INTO Customers VALUES (4, 'Diana', 55, 12000, 'FALSE');

***-- Insert sample loans***

INSERT INTO Loans VALUES (101, 1, 7.5, SYSDATE + 10);

INSERT INTO Loans VALUES (102, 2, 6.0, SYSDATE + 40);

INSERT INTO Loans VALUES (103, 3, 8.0, SYSDATE + 5);

INSERT INTO Loans VALUES (104, 4, 7.0, SYSDATE + 25);

**Apply 1% Discount for Customers Over 60  
  
Code :**BEGIN

FOR rec IN (

SELECT l.LoanID

FROM Loans l

JOIN Customers c ON l.CustomerID = c.CustomerID

WHERE c.Age > 60

) LOOP

UPDATE Loans

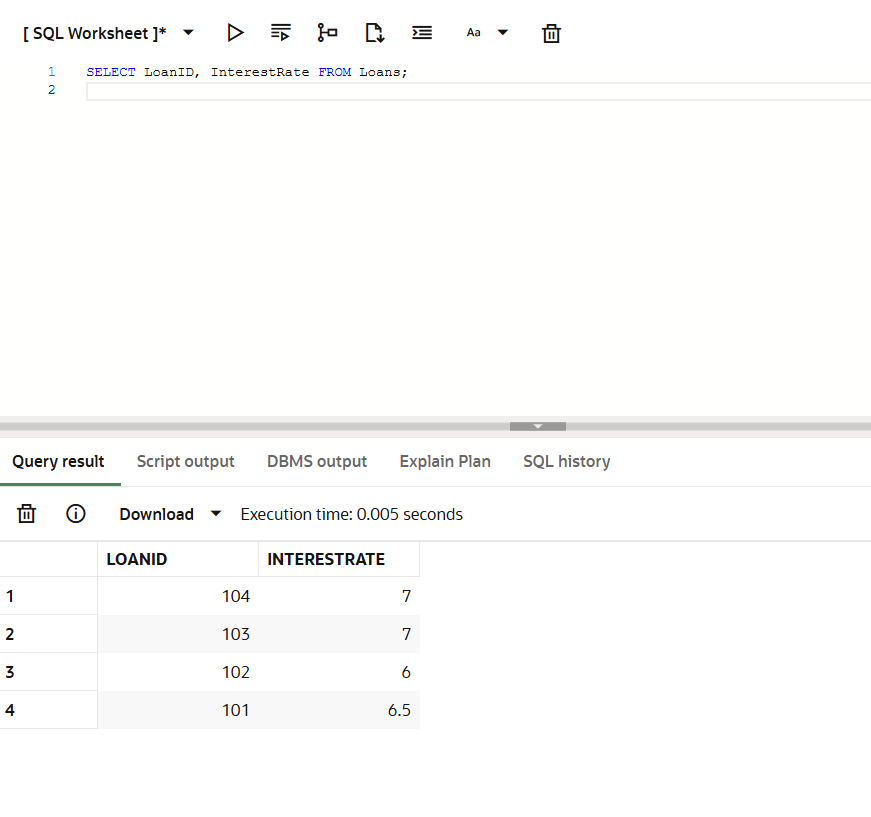
SET InterestRate = InterestRate - 1

WHERE LoanID = rec.LoanID;

END LOOP;

COMMIT;

END**;**  
**RESULT :**SELECT LoanID, InterestRate FROM Loans;

**Output :  
  
  
**

**Scenario : 2  
  
Promote Customers to VIP  
  
Code :**BEGIN

FOR rec IN (

SELECT CustomerID

FROM Customers

WHERE Balance > 10000

) LOOP

UPDATE Customers

SET IsVIP = 'TRUE'

WHERE CustomerID = rec.CustomerID;

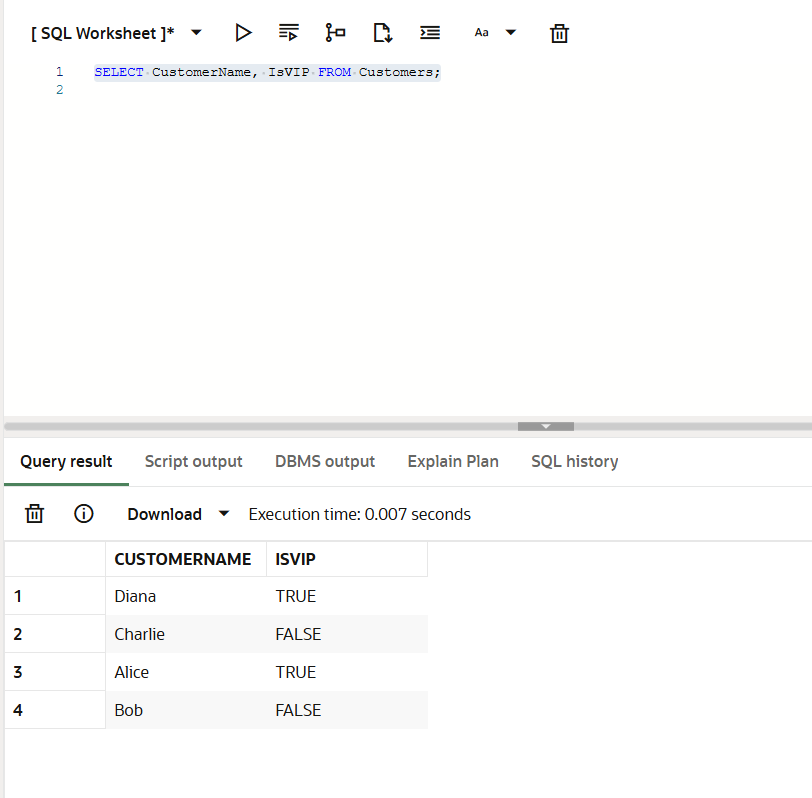
END LOOP;

COMMIT;

END;

/

**RESULT :**SELECT CustomerName, IsVIP FROM Customers;

**Output :**  
  


**Scenario : 3  
  
Code :**BEGIN

FOR rec IN (

SELECT l.LoanID, l.DueDate, c.CustomerName

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: Loan ID ' || rec.LoanID ||

' for customer ' || rec.CustomerName ||

' is due on ' || TO\_CHAR(rec.DueDate, 'DD-MON-YYYY')

);

END LOOP;

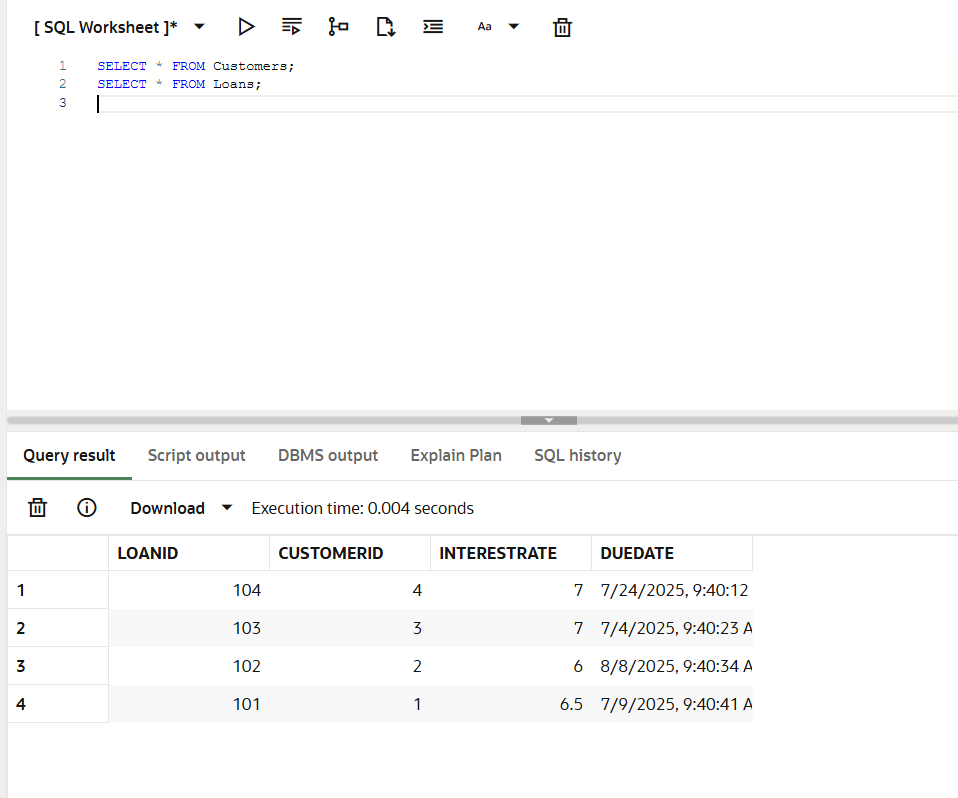
END;

/

**RESULT :**SELECT \* FROM Customers;

SELECT \* FROM Loans;

**Output :**



Exercise 3. Stored Procedures  
 **Scenario : 1  
  
Code :**  
***-- Create SavingsAccounts table***

CREATE TABLE SavingsAccounts (

AccountID NUMBER PRIMARY KEY,

Balance NUMBER

);

***-- Insert sample data***

INSERT INTO SavingsAccounts VALUES (1, 1000);

INSERT INTO SavingsAccounts VALUES (2, 2500);

INSERT INTO SavingsAccounts VALUES (3, 5000);

**Stored Procedure: ProcessMonthlyInterest**

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest IS

BEGIN

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

UPDATE SavingsAccounts

SET Balance = Balance + (Balance \* 0.01)

WHERE AccountID = rec.AccountID;

END LOOP;

COMMIT;

END;

/

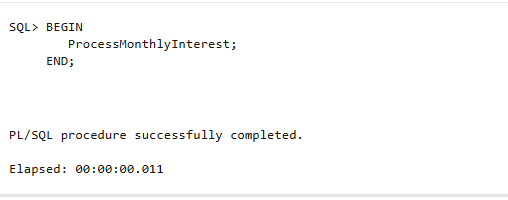
BEGIN

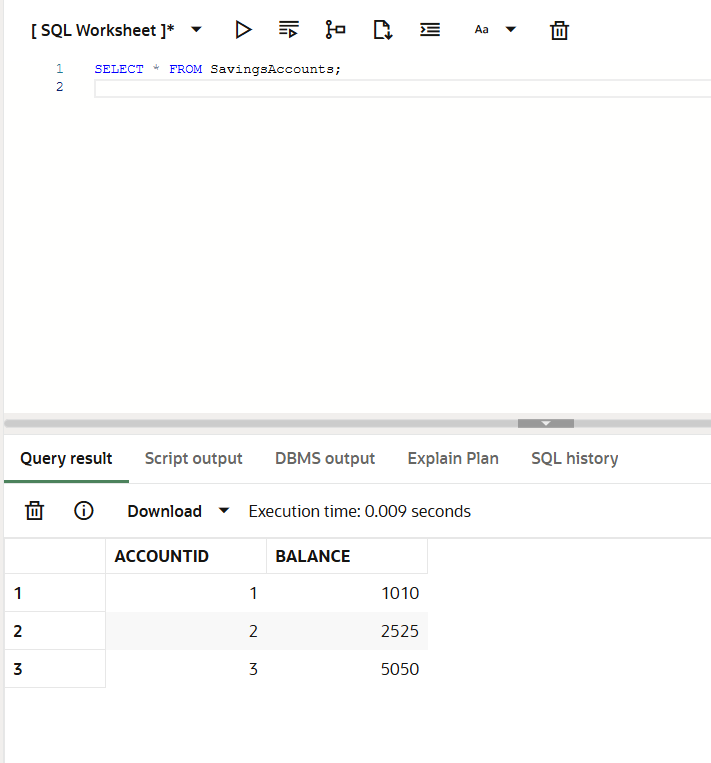
ProcessMonthlyInterest;

END;

/  
  
**RESULT :**SELECT \* FROM SavingsAccounts;

**Output :**

****

**  
Scenario : 2  
  
Code :**

***-- Create Employees table***

CREATE TABLE Employees (

EmployeeID NUMBER PRIMARY KEY,

DepartmentID NUMBER,

Salary NUMBER

);

***-- Insert sample data***

INSERT INTO Employees VALUES (1, 10, 5000);

INSERT INTO Employees VALUES (2, 20, 6000);

INSERT INTO Employees VALUES (3, 10, 7000);

INSERT INTO Employees VALUES (4, 30, 8000);

**Stored Procedure: UpdateEmployeeBonus**  
  
CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus(

p\_DepartmentID IN NUMBER,

p\_BonusPercent IN NUMBER

) IS

BEGIN

UPDATE Employees

SET Salary = Salary + (Salary \* p\_BonusPercent / 100)

WHERE DepartmentID = p\_DepartmentID;

COMMIT;

END;

/

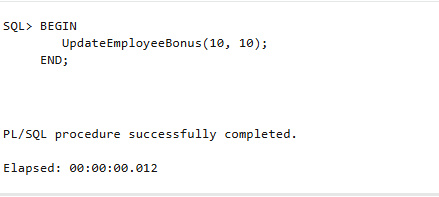
BEGIN

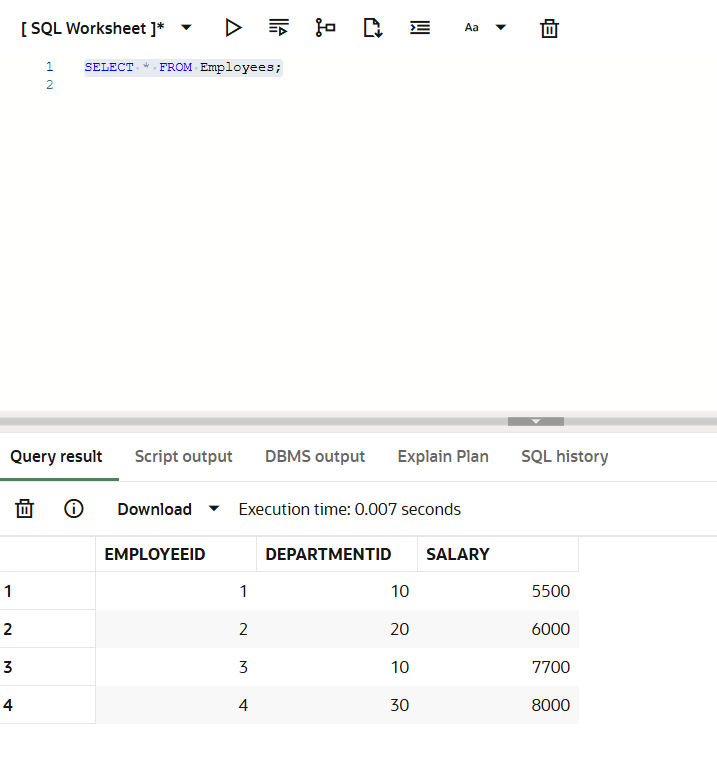
UpdateEmployeeBonus(10, 10);

END;

/  
 **RESULT :**SELECT \* FROM Employees;

**Output :**

****

****

i **Scenario : 2  
  
Code :**

***-- Create Employees table***CREATE TABLE Accounts (

AccountID NUMBER PRIMARY KEY,

Balance NUMBER

);

***-- Insert sample data***

INSERT INTO Accounts VALUES (1, 5000);

INSERT INTO Accounts VALUES (2, 3000);

INSERT INTO Accounts VALUES (3, 2000);

**Stored Procedure: TransferFunds**

CREATE OR REPLACE PROCEDURE TransferFunds(

p\_FromAccountID IN NUMBER,

p\_ToAccountID IN NUMBER,

p\_Amount IN NUMBER

) IS

v\_FromBalance NUMBER;

BEGIN

-- Check balance

SELECT Balance INTO v\_FromBalance

FROM Accounts

WHERE AccountID = p\_FromAccountID;

IF v\_FromBalance < p\_Amount THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Insufficient funds.');

END IF;

-- Deduct from source

UPDATE Accounts

SET Balance = Balance - p\_Amount

WHERE AccountID = p\_FromAccountID;

-- Add to destination

UPDATE Accounts

SET Balance = Balance + p\_Amount

WHERE AccountID = p\_ToAccountID;

COMMIT;

END;

/  
  
  
BEGIN

TransferFunds(1, 2, 1000);

END;

/

**RESULT :**SELECT \* FROM Accounts;

**Output :**

