**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.

**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.

**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.

**Scenario 1:**

DECLARE

CURSOR loan\_cur IS

SELECT l.LoanID, l.InterestRate, c.DOB, c.Name

FROM Loans l

JOIN Customers c ON l.CustomerID = c.CustomerID;

v\_age NUMBER;

BEGIN

FOR loan\_rec IN loan\_cur LOOP

v\_age := FLOOR(MONTHS\_BETWEEN(SYSDATE, loan\_rec.DOB)/12);

IF v\_age > 60 THEN

UPDATE Loans

SET InterestRate = InterestRate - 1

WHERE LoanID = loan\_rec.LoanID;

DBMS\_OUTPUT.PUT\_LINE('Applied discount to ' || loan\_rec.Name ||

'''s loan: ' || loan\_rec.InterestRate ||

'% -> ' || (loan\_rec.InterestRate - 1) || '%');

END IF;

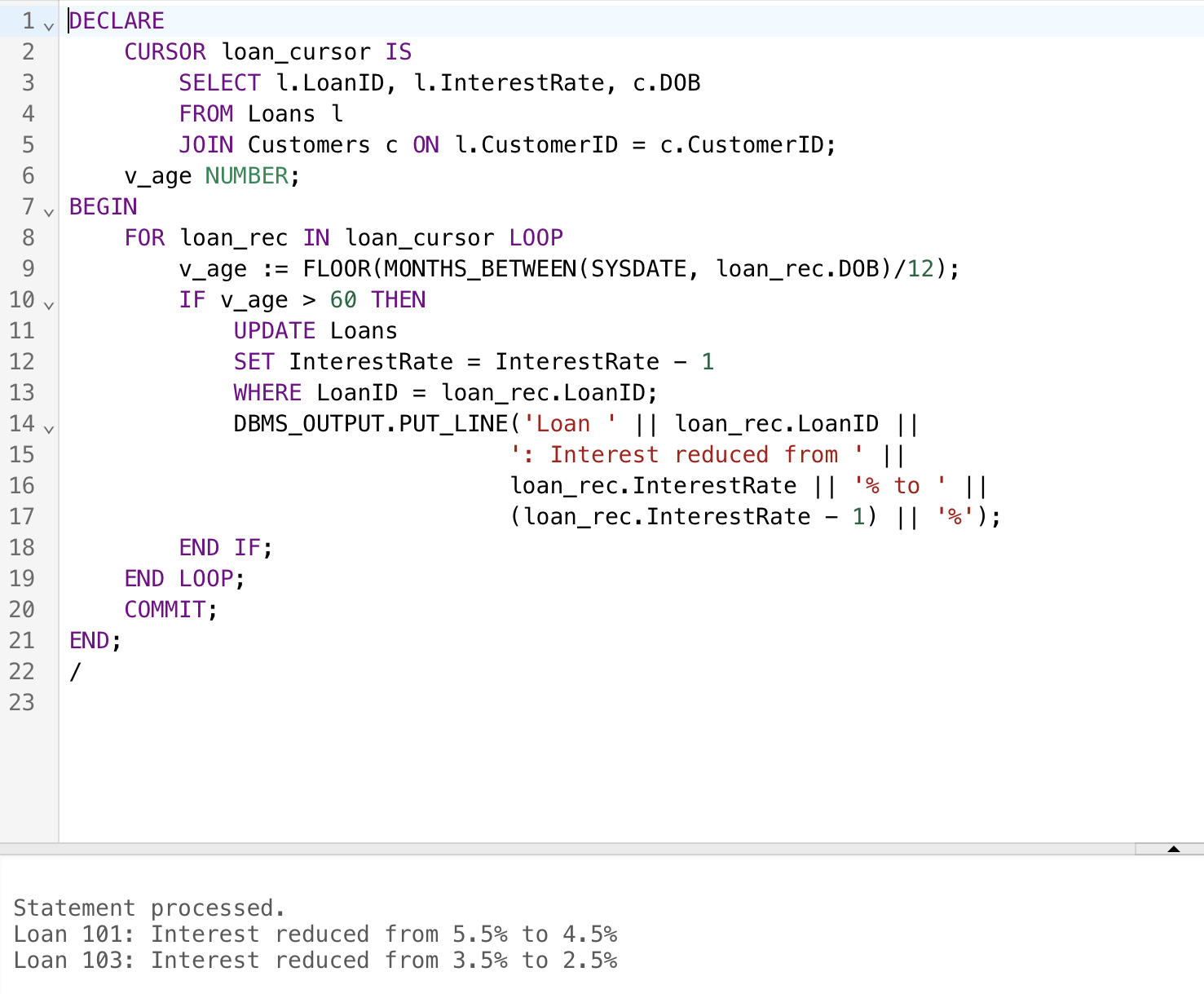
END LOOP;

COMMIT;

END;

/

OUTPUT:



**Scenario 2:**

ALTER TABLE Customers ADD IsVIP VARCHAR2(5) DEFAULT 'FALSE';

BEGIN

FOR cust\_rec IN (SELECT CustomerID, Name, Balance FROM Customers)

LOOP

IF cust\_rec.Balance > 10000 THEN

UPDATE Customers

SET IsVIP = 'TRUE'

WHERE CustomerID = cust\_rec.CustomerID;

DBMS\_OUTPUT.PUT\_LINE('VIP Status: ' || cust\_rec.Name ||

' (Balance: $' || cust\_rec.Balance || ')');

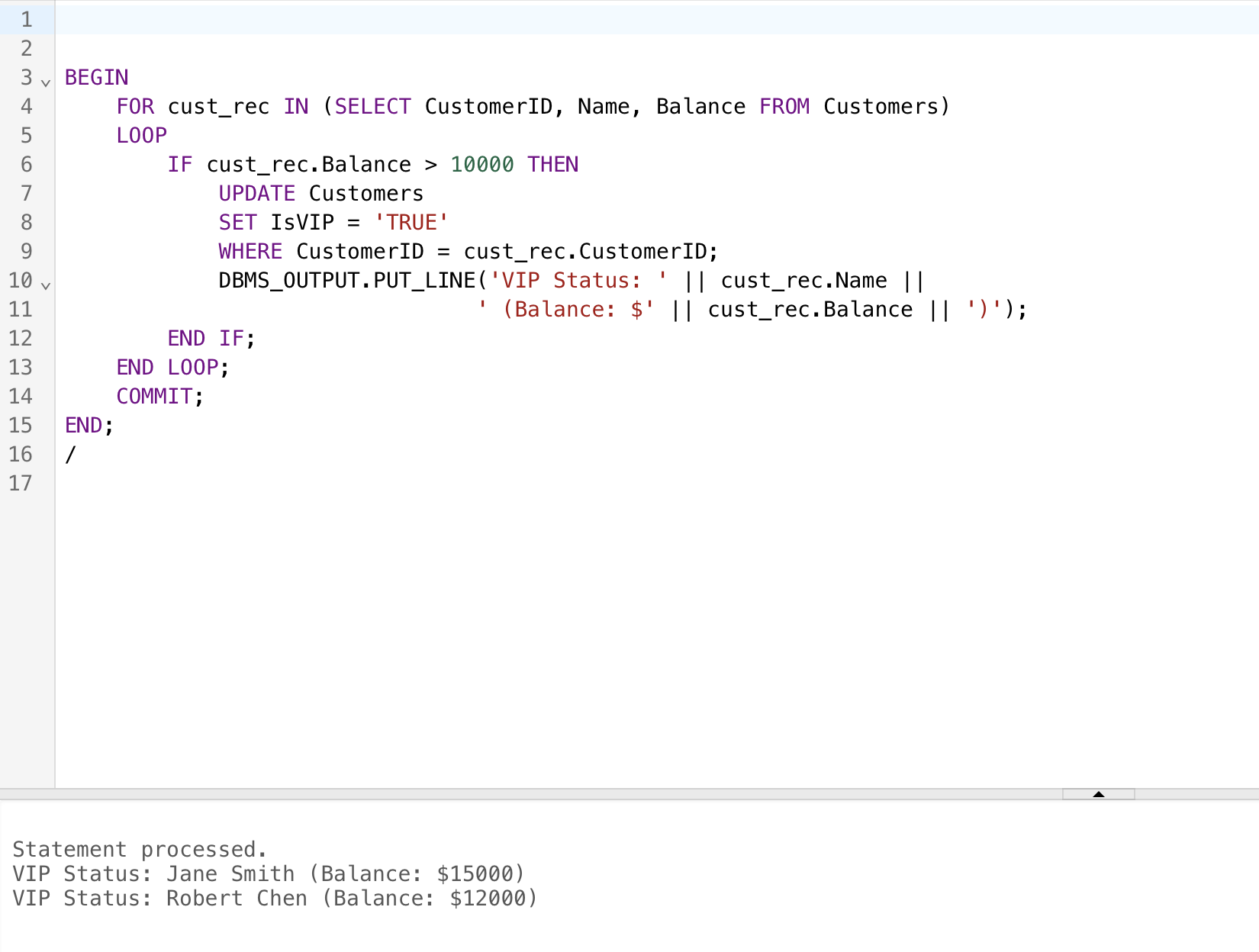
END IF;

END LOOP;

COMMIT;

END;

/



**Scenario 3:**

DECLARE

CURSOR loan\_due IS

SELECT l.LoanID, c.Name, l.EndDate

FROM Loans l

JOIN Customers c ON l.CustomerID = c.CustomerID

WHERE l.EndDate BETWEEN SYSDATE AND SYSDATE + 30;

BEGIN

FOR rec IN loan\_due LOOP

DBMS\_OUTPUT.PUT\_LINE('Reminder: ' || rec.Name ||

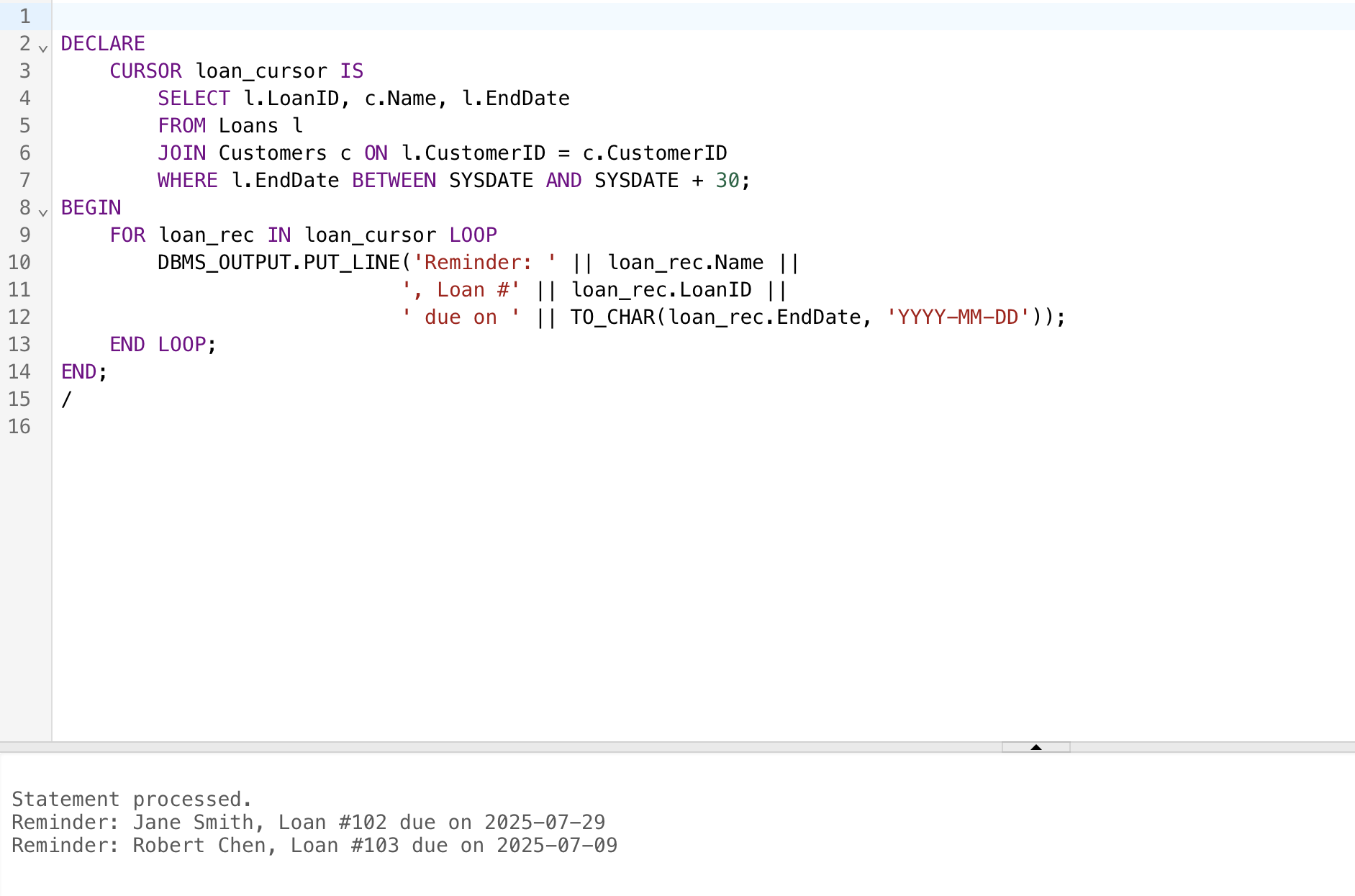
', Loan #' || rec.LoanID ||

' due on ' || TO\_CHAR(rec.EndDate, 'YYYY-MM-DD'));

END LOOP;

END;

/



**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.

**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.

**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.

*-- Exercise 3: Stored Procedures*

*-- Scenario 1: Monthly Interest*

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest AS

BEGIN

UPDATE Accounts

SET Balance = ROUND(Balance \* 1.01, 2)

WHERE AccountType = 'Savings';

DBMS\_OUTPUT.PUT\_LINE('Applied 1% interest to savings accounts');

COMMIT;

END;

/

*-- Scenario 2: Employee Bonus*

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus(

p\_dept VARCHAR2,

p\_bonus\_percent NUMBER

) AS

BEGIN

UPDATE Employees

SET Salary = Salary \* (1 + p\_bonus\_percent/100)

WHERE Department = p\_dept;

DBMS\_OUTPUT.PUT\_LINE(p\_bonus\_percent || '% bonus applied to ' || p\_dept || ' department');

COMMIT;

END;

/

*-- Scenario 3: Fund Transfer*

CREATE OR REPLACE PROCEDURE TransferFunds(

p\_from\_acc NUMBER,

p\_to\_acc NUMBER,

p\_amount NUMBER

) AS

v\_balance NUMBER;

v\_from\_exists NUMBER;

v\_to\_exists NUMBER;

BEGIN

SELECT COUNT(\*) INTO v\_from\_exists FROM Accounts WHERE AccountID = p\_from\_acc;

SELECT COUNT(\*) INTO v\_to\_exists FROM Accounts WHERE AccountID = p\_to\_acc;

IF v\_from\_exists = 0 THEN

RAISE\_APPLICATION\_ERROR(-20002, 'Source account does not exist');

ELSIF v\_to\_exists = 0 THEN

RAISE\_APPLICATION\_ERROR(-20003, 'Target account does not exist');

END IF;

SELECT Balance INTO v\_balance

FROM Accounts

WHERE AccountID = p\_from\_acc;

IF v\_balance < p\_amount THEN

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

END IF;

UPDATE Accounts SET Balance = Balance - p\_amount

WHERE AccountID = p\_from\_acc;

UPDATE Accounts SET Balance = Balance + p\_amount

WHERE AccountID = p\_to\_acc;

DBMS\_OUTPUT.PUT\_LINE('Transferred $' || p\_amount || ' from account ' ||

p\_from\_acc || ' to ' || p\_to\_acc);

COMMIT;

END;

/

*-- Test Execution*

SET SERVEROUTPUT ON;

*-- Test Exercise 1 Outputs*

BEGIN

DBMS\_OUTPUT.PUT\_LINE('--- Exercise 1 Outputs ---');

END;

/

*-- Test Exercise 3 Procedures*

BEGIN

DBMS\_OUTPUT.PUT\_LINE('--- Testing ProcessMonthlyInterest ---');

ProcessMonthlyInterest();

FOR acc IN (SELECT \* FROM Accounts WHERE AccountType = 'Savings')

LOOP

DBMS\_OUTPUT.PUT\_LINE('Account ' || acc.AccountID || ': $' || acc.Balance);

END LOOP;

DBMS\_OUTPUT.PUT\_LINE('--- Testing UpdateEmployeeBonus ---');

UpdateEmployeeBonus('HR', 10);

FOR emp IN (SELECT \* FROM Employees WHERE Department = 'HR')

LOOP

DBMS\_OUTPUT.PUT\_LINE(emp.Name || ': $' || emp.Salary);

END LOOP;

DBMS\_OUTPUT.PUT\_LINE('--- Testing TransferFunds ---');

DBMS\_OUTPUT.PUT\_LINE('Before transfer:');

DBMS\_OUTPUT.PUT\_LINE('Account 1001: $' || (SELECT Balance FROM Accounts WHERE AccountID = 1001));

DBMS\_OUTPUT.PUT\_LINE('Account 1002: $' || (SELECT Balance FROM Accounts WHERE AccountID = 1002));

TransferFunds(1001, 1002, 500);

DBMS\_OUTPUT.PUT\_LINE('After transfer:');

DBMS\_OUTPUT.PUT\_LINE('Account 1001: $' || (SELECT Balance FROM Accounts WHERE AccountID = 1001));

DBMS\_OUTPUT.PUT\_LINE('Account 1002: $' || (SELECT Balance FROM Accounts WHERE AccountID = 1002));

END;

/

OUTPUT:

