**Exercise 1: Control Statements**

**Scenario 1:**

Code:

DECLARE

v\_age NUMBER;

BEGIN

FOR cust\_rec IN (SELECT CustomerID, DOB FROM Customers) LOOP

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

IF v\_age > 60 THEN

UPDATE Loans

SET InterestRate = InterestRate - 1

WHERE CustomerID = cust\_rec.CustomerID;

END IF;

END LOOP;

COMMIT;

END;

**Scenario 2:**

Code:

*Add IsVIP Column if Not Exists*

ALTER TABLE Customers

ADD IsVIP VARCHAR2(5);

*PL/SQL Block with DECLARE*

DECLARE

v\_isVIP VARCHAR2(5);

BEGIN

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

IF cust\_rec.Balance > 10000 THEN

v\_isVIP := 'TRUE';

ELSE

v\_isVIP := 'FALSE';

END IF;

UPDATE Customers

SET IsVIP = v\_isVIP

WHERE CustomerID = cust\_rec.CustomerID;

END LOOP;

COMMIT;

END;

**Scenario 3:**

Code:

DECLARE

v\_due\_date DATE;

BEGIN

FOR loan\_rec IN (

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

FROM Loans l

JOIN Customers c ON l.CustomerID = c.CustomerID

WHERE l.EndDate BETWEEN SYSDATE AND SYSDATE + 30

) LOOP

v\_due\_date := loan\_rec.EndDate;

DBMS\_OUTPUT.PUT\_LINE('Reminder: Loan ID ' || loan\_rec.LoanID ||

' for customer ' || loan\_rec.Name ||

' is due on ' || TO\_CHAR(v\_due\_date, 'DD-MON-YYYY'));

END LOOP;

END;

**Exercise 3: Stored Procedures**

**Scenario 1:**

Code:

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest AS

BEGIN

FOR acc\_rec IN (SELECT AccountID, Balance FROM Accounts WHERE AccountType = 'Savings') LOOP

UPDATE Accounts

SET Balance = Balance + (Balance \* 0.01)

WHERE AccountID = acc\_rec.AccountID;

END LOOP;

COMMIT;

END;

**Scenario 2:**

Code:

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus (

p\_Department IN VARCHAR2,

p\_BonusPercent IN NUMBER

) AS

BEGIN

UPDATE Employees

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

WHERE Department = p\_Department;

COMMIT;

END;

**Scenario 3:**

Code:

CREATE OR REPLACE PROCEDURE TransferFunds (

p\_SourceAccountID IN NUMBER,

p\_DestinationAccountID IN NUMBER,

p\_Amount IN NUMBER

) AS

v\_SourceBalance NUMBER;

BEGIN

-- Fetch current balance of source account

SELECT Balance INTO v\_SourceBalance

FROM Accounts

WHERE AccountID = p\_SourceAccountID

FOR UPDATE;

-- Check for sufficient funds

IF v\_SourceBalance >= p\_Amount THEN

-- Deduct from source

UPDATE Accounts

SET Balance = Balance - p\_Amount

WHERE AccountID = p\_SourceAccountID;

-- Add to destination

UPDATE Accounts

SET Balance = Balance + p\_Amount

WHERE AccountID = p\_DestinationAccountID;

COMMIT;

DBMS\_OUTPUT.PUT\_LINE('Transfer successful: ' || p\_Amount || ' transferred.');

ELSE

DBMS\_OUTPUT.PUT\_LINE('Transfer failed: Insufficient balance.');

END IF;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Invalid Account ID provided.');

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Unexpected error: ' || SQLERRM);

END;