## Exercise 1: Control Structures - Scenario 1

DECLARE

CURSOR cust\_cursor IS SELECT CustomerID, DOB FROM Customers;

v\_age NUMBER;

BEGIN

FOR cust\_rec IN cust\_cursor LOOP

v\_age := FLOOR(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;

END;

/

## Exercise 1: Control Structures - Scenario 2

ALTER TABLE Customers ADD IsVIP CHAR(1);

BEGIN

UPDATE Customers SET IsVIP = 'Y' WHERE Balance > 10000;

END;

/

## Exercise 1: Control Structures - Scenario 3

DECLARE

CURSOR due\_loans IS

SELECT l.CustomerID, 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 loan\_rec IN due\_loans LOOP

DBMS\_OUTPUT.PUT\_LINE('Reminder: ' || loan\_rec.Name || ' - Your loan is due on ' || loan\_rec.EndDate);

END LOOP;

END;

/

## Exercise 2: Error Handling - Scenario 1

CREATE OR REPLACE PROCEDURE SafeTransferFunds(p\_from NUMBER, p\_to NUMBER, p\_amount NUMBER) IS

v\_balance NUMBER;

BEGIN

SELECT Balance INTO v\_balance FROM Accounts WHERE AccountID = p\_from;

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;

UPDATE Accounts SET Balance = Balance + p\_amount WHERE AccountID = p\_to;

COMMIT;

EXCEPTION

WHEN OTHERS THEN

ROLLBACK;

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

END;

/

## Exercise 2: Error Handling - Scenario 2

CREATE OR REPLACE PROCEDURE UpdateSalary(p\_empid NUMBER, p\_percent NUMBER) IS

BEGIN

UPDATE Employees

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

WHERE EmployeeID = p\_empid;

IF SQL%ROWCOUNT = 0 THEN

RAISE\_APPLICATION\_ERROR(-20002, 'Employee ID not found');

END IF;

COMMIT;

EXCEPTION

WHEN OTHERS THEN

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

END;

/

## Exercise 2: Error Handling - Scenario 3

CREATE OR REPLACE PROCEDURE AddNewCustomer(p\_id NUMBER, p\_name VARCHAR2, p\_dob DATE, p\_balance NUMBER) IS

BEGIN

INSERT INTO Customers(CustomerID, Name, DOB, Balance, LastModified)

VALUES(p\_id, p\_name, p\_dob, p\_balance, SYSDATE);

COMMIT;

EXCEPTION

WHEN DUP\_VAL\_ON\_INDEX THEN

DBMS\_OUTPUT.PUT\_LINE('Error: Customer ID already exists');

WHEN OTHERS THEN

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

END;

/

## Exercise 3: Stored Procedures - Scenario 1

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest IS

BEGIN

UPDATE Accounts

SET Balance = Balance + (Balance \* 0.01),

LastModified = SYSDATE

WHERE AccountType = 'Savings';

COMMIT;

END;

/

## Exercise 3: Stored Procedures - Scenario 2

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus(p\_dept VARCHAR2, p\_bonus\_percent NUMBER) IS

BEGIN

UPDATE Employees

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

WHERE Department = p\_dept;

COMMIT;

END;

/

## Exercise 3: Stored Procedures - Scenario 3

CREATE OR REPLACE PROCEDURE TransferFunds(p\_from NUMBER, p\_to NUMBER, p\_amount NUMBER) IS

v\_balance NUMBER;

BEGIN

SELECT Balance INTO v\_balance FROM Accounts WHERE AccountID = p\_from;

IF v\_balance < p\_amount THEN

RAISE\_APPLICATION\_ERROR(-20003, 'Insufficient balance');

END IF;

UPDATE Accounts SET Balance = Balance - p\_amount WHERE AccountID = p\_from;

UPDATE Accounts SET Balance = Balance + p\_amount WHERE AccountID = p\_to;

COMMIT;

END;

/