# Exercise 1: Control Structures

**-- Scenario 1: Apply interest discount for seniors**BEGIN  
 FOR customer\_rec IN (SELECT \* FROM Customers) LOOP  
 IF MONTHS\_BETWEEN(SYSDATE, customer\_rec.DOB) / 12 > 60 THEN  
 UPDATE Loans  
 SET InterestRate = InterestRate - 1  
 WHERE CustomerID = customer\_rec.CustomerID;  
 END IF;  
 END LOOP;  
END;  
/

**-- Scenario 2: Promote to VIP based on balance**  
BEGIN  
 FOR customer\_rec IN (SELECT \* FROM Customers) LOOP  
 IF customer\_rec.Balance > 10000 THEN  
 UPDATE Customers  
 SET IsVIP = 'TRUE'  
 WHERE CustomerID = customer\_rec.CustomerID;  
 END IF;  
 END LOOP;  
END;  
/

**-- Scenario 3: Loan due reminders**  
BEGIN  
 FOR loan\_rec IN (  
 SELECT \* FROM Loans  
 WHERE EndDate <= SYSDATE + 30  
 ) LOOP  
 DBMS\_OUTPUT.PUT\_LINE('Reminder: Loan ' || loan\_rec.LoanID ||  
 ' for customer ' || loan\_rec.CustomerID ||  
 ' is due on ' || TO\_CHAR(loan\_rec.EndDate, 'YYYY-MM-DD'));  
 END LOOP;  
END;  
/

# Exercise 2: Error Handling

**-- Scenario 1: Safe fund transfer**  
CREATE OR REPLACE PROCEDURE SafeTransferFunds(  
 p\_fromAccount NUMBER,  
 p\_toAccount NUMBER,  
 p\_amount NUMBER  
)  
IS  
 v\_balance NUMBER;  
BEGIN  
 SELECT Balance INTO v\_balance FROM Accounts WHERE AccountID = p\_fromAccount;  
  
 IF v\_balance < p\_amount THEN  
 RAISE\_APPLICATION\_ERROR(-20001, 'Insufficient funds');  
 END IF;  
  
 UPDATE Accounts SET Balance = Balance - p\_amount WHERE AccountID = p\_fromAccount;  
 UPDATE Accounts SET Balance = Balance + p\_amount WHERE AccountID = p\_toAccount;  
  
 COMMIT;  
EXCEPTION  
 WHEN OTHERS THEN  
 ROLLBACK;  
 DBMS\_OUTPUT.PUT\_LINE('Error during fund transfer: ' || SQLERRM);  
END;  
/

**-- Scenario 2: Update salary**  
CREATE OR REPLACE PROCEDURE UpdateSalary(  
 p\_empID NUMBER,  
 p\_percentage NUMBER  
)  
IS  
BEGIN  
 UPDATE Employees  
 SET Salary = Salary + (Salary \* p\_percentage / 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 updating salary: ' || SQLERRM);  
 ROLLBACK;  
END;  
/

**-- Scenario 3: Add new customer**  
CREATE OR REPLACE PROCEDURE AddNewCustomer(  
 p\_customerID NUMBER,  
 p\_name VARCHAR2,  
 p\_dob DATE,  
 p\_balance NUMBER  
)  
IS  
BEGIN  
 INSERT INTO Customers (CustomerID, Name, DOB, Balance, LastModified)  
 VALUES (p\_customerID, p\_name, p\_dob, p\_balance, SYSDATE);  
  
 COMMIT;  
EXCEPTION  
 WHEN DUP\_VAL\_ON\_INDEX THEN  
 DBMS\_OUTPUT.PUT\_LINE('Customer already exists with ID ' || p\_customerID);  
 ROLLBACK;  
 WHEN OTHERS THEN  
 DBMS\_OUTPUT.PUT\_LINE('Error adding customer: ' || SQLERRM);  
 ROLLBACK;  
END;  
/

# Exercise 3: Stored Procedures

**-- Scenario 1: Process monthly interest**  
CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest  
IS  
BEGIN  
 UPDATE Accounts  
 SET Balance = Balance + (Balance \* 0.01)  
 WHERE AccountType = 'Savings';  
  
 COMMIT;  
END;  
/

**-- Scenario 2: Update employee bonus**CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus(  
 p\_department VARCHAR2,  
 p\_bonus\_percent NUMBER  
)  
IS  
BEGIN  
 UPDATE Employees  
 SET Salary = Salary + (Salary \* p\_bonus\_percent / 100)  
 WHERE Department = p\_department;  
  
 COMMIT;  
END;  
/

**-- Scenario 3: Customer fund transfer**  
CREATE OR REPLACE PROCEDURE TransferFunds(  
 p\_fromAccount NUMBER,  
 p\_toAccount NUMBER,  
 p\_amount NUMBER  
)  
IS  
 v\_balance NUMBER;  
BEGIN  
 SELECT Balance INTO v\_balance FROM Accounts WHERE AccountID = p\_fromAccount;  
  
 IF v\_balance < p\_amount THEN  
 RAISE\_APPLICATION\_ERROR(-20003, 'Insufficient balance');  
 END IF;  
  
 UPDATE Accounts SET Balance = Balance - p\_amount WHERE AccountID = p\_fromAccount;  
 UPDATE Accounts SET Balance = Balance + p\_amount WHERE AccountID = p\_toAccount;  
  
 COMMIT;  
EXCEPTION  
 WHEN OTHERS THEN  
 DBMS\_OUTPUT.PUT\_LINE('Transfer error: ' || SQLERRM);  
 ROLLBACK;  
END;  
/