**Exercise 1: Control Structures**

1.1

BEGIN  
FOR c IN (SELECT LoanID, InterestRate, DOB FROM Loans JOIN Customers USING(CustomerID))  
 LOOP  
 IF MONTHS\_BETWEEN(SYSDATE, c.DOB) / 12 > 60 THEN  
 UPDATE Loans SET InterestRate = InterestRate - 1 WHERE LoanID = c.LoanID;  
 END IF;  
 END LOOP;  
END;

1.2

ALTER TABLE Customers ADD IsVIP VARCHAR2(5);  
BEGIN  
FOR c IN (SELECT CustomerID, Balance FROM Customers)  
 LOOP  
 IF c.Balance > 10000 THEN  
 UPDATE Customers SET IsVIP = 'TRUE' WHERE CustomerID = c.CustomerID;  
 END IF;  
 END LOOP;  
END;

1.3

BEGIN  
 FOR rec IN (  
 SELECT Name, EndDate FROM Loans JOIN Customers USING(CustomerID)  
 WHERE EndDate <= SYSDATE + 30  
 )  
 LOOP  
 DBMS\_OUTPUT.PUT\_LINE('Reminder: ' || rec.Name || ', your loan is due on ' || TO\_CHAR(rec.EndDate, 'DD-Mon-YYYY'));  
 END LOOP;  
END;

**Exercise 2: Error Handling**

2.1

CREATE OR REPLACE PROCEDURE SafeTransferFunds(p\_from NUMBER, p\_to NUMBER, p\_amount NUMBER) IS  
 insufficient\_funds EXCEPTION;  
BEGIN  
 DECLARE  
 v\_balance NUMBER;  
 BEGIN  
 SELECT Balance INTO v\_balance FROM Accounts WHERE AccountID = p\_from;  
 IF v\_balance < p\_amount THEN  
 RAISE 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 insufficient\_funds THEN  
 DBMS\_OUTPUT.PUT\_LINE('Error: Insufficient funds.');  
 ROLLBACK;  
 WHEN OTHERS THEN  
 DBMS\_OUTPUT.PUT\_LINE('Transfer failed: ' || SQLERRM);  
 ROLLBACK;  
 END;  
END;

2.2

Update salary with error handling:

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%NOTFOUND THEN  
 RAISE\_APPLICATION\_ERROR(-20001, 'Employee ID not found.');  
 END IF;  
  
 COMMIT;  
EXCEPTION  
 WHEN OTHERS THEN  
 DBMS\_OUTPUT.PUT\_LINE('Error updating salary: ' || SQLERRM);  
 ROLLBACK;  
END;

2.3

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('Customer with ID ' || p\_id || ' already exists.');  
 ROLLBACK;  
 WHEN OTHERS THEN  
 DBMS\_OUTPUT.PUT\_LINE('Error: ' || SQLERRM);  
 ROLLBACK;  
END;

**Exercise 3: Stored Procedures**

3.1

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest IS  
BEGIN  
 UPDATE Accounts  
 SET Balance = Balance + (Balance \* 0.01)  
 WHERE AccountType = 'Savings';  
 COMMIT;  
END;

3.2

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus(p\_dept VARCHAR2, p\_bonus\_pct NUMBER) IS  
BEGIN  
 UPDATE Employees  
 SET Salary = Salary + (Salary \* p\_bonus\_pct / 100)  
 WHERE Department = p\_dept;  
 COMMIT;  
END;

3.3

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  
 UPDATE Accounts SET Balance = Balance - p\_amount WHERE AccountID = p\_from;  
 UPDATE Accounts SET Balance = Balance + p\_amount WHERE AccountID = p\_to;  
 COMMIT;  
 ELSE  
 RAISE\_APPLICATION\_ERROR(-20002, 'Insufficient balance');  
 END IF;  
END;

**Exercise 4: Functions**

4.1

CREATE OR REPLACE FUNCTION CalculateAge(p\_dob DATE) RETURN NUMBER IS  
BEGIN  
 RETURN TRUNC(MONTHS\_BETWEEN(SYSDATE, p\_dob)/12);  
END;

4,2

CREATE OR REPLACE FUNCTION CalculateMonthlyInstallment(p\_amount NUMBER, p\_rate NUMBER, p\_years NUMBER)  
RETURN NUMBER IS  
 r NUMBER := p\_rate / (12 \* 100);  
 n NUMBER := p\_years \* 12;  
BEGIN  
 RETURN ROUND(p\_amount \* r / (1 - POWER(1 + r, -n)), 2);  
END;

4.3   
CREATE OR REPLACE FUNCTION HasSufficientBalance(p\_accountid NUMBER, p\_amount NUMBER) RETURN BOOLEAN IS  
 v\_balance NUMBER;  
BEGIN  
 SELECT Balance INTO v\_balance FROM Accounts WHERE AccountID = p\_accountid;  
 RETURN v\_balance >= p\_amount;  
EXCEPTION  
 WHEN NO\_DATA\_FOUND THEN  
 RETURN FALSE;  
END;