**Exercise 1: Control Structures**

**Scenario 1:**

SQL> DECLARE

2 CURSOR c1 IS

3 SELECT c.CustomerID, l.LoanID, l.InterestRate, c.DOB

4 FROM Customers c

5 JOIN Loans l ON c.CustomerID = l.CustomerID;

6

7 v\_age NUMBER;

8 BEGIN

9 FOR rec IN c1 LOOP

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

11 IF v\_age > 60 THEN

12 UPDATE Loans

13 SET InterestRate = InterestRate - 1

14 WHERE LoanID = rec.LoanID;

15 END IF;

16 END LOOP;

17

18 COMMIT;

19 END;

20 /

PL/SQL procedure successfully completed.

**Scenario 2:**

SQL> ALTER TABLE Customers ADD (IsVIP VARCHAR2(5));

Table altered.

SQL> BEGIN

2 FOR rec IN (SELECT CustomerID, Balance FROM Customers) LOOP

3 IF rec.Balance > 10000 THEN

4 UPDATE Customers

5 SET IsVIP = 'TRUE'

6 WHERE CustomerID = rec.CustomerID;

7 ELSE

8 UPDATE Customers

9 SET IsVIP = 'FALSE'

10 WHERE CustomerID = rec.CustomerID;

11 END IF;

12 END LOOP;

13

14 COMMIT;

15 END;

16 /

PL/SQL procedure successfully completed.

**Scenario 3:**

SQL> SET SERVEROUTPUT ON;

SQL>

SQL> DECLARE

2 CURSOR c2 IS

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

4 FROM Loans l

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

6 WHERE l.EndDate BETWEEN SYSDATE AND SYSDATE + 30;

7 BEGIN

8 FOR rec IN c2 LOOP

9 DBMS\_OUTPUT.PUT\_LINE('Reminder: Loan ' || rec.LoanID || ' for customer ' || rec.Name || ' is due on ' || TO\_CHAR(rec.EndDate, 'YYYY-MM-DD'));

10 END LOOP;

11 END;

12 /

PL/SQL procedure successfully completed.

**Exercise 3: Stored Procedures**

**Scenario 1:**

SQL> CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest

2 IS

3 BEGIN

4 UPDATE Accounts

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

6 WHERE AccountType = 'Savings';

7 COMMIT;

8 END;

9 /

Procedure created.

**Scenario 2:**

SQL> CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus (

2 p\_department IN VARCHAR2,

3 p\_bonus\_percent IN NUMBER

4 )

5 IS

6 BEGIN

7 UPDATE Employees

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

9 WHERE Department = p\_department;

10 COMMIT;

11 END;

12 /

Procedure created.

SQL> BEGIN

2 UpdateEmployeeBonus('IT', 10);

3 END;

4 /

PL/SQL procedure successfully completed.

**Scenario 3:**

SQL> CREATE OR REPLACE PROCEDURE TransferFunds (

2 p\_from\_account IN NUMBER,

3 p\_to\_account IN NUMBER,

4 p\_amount IN NUMBER

5 )

6 IS

7 v\_balance NUMBER;

8 BEGIN

9 SELECT Balance INTO v\_balance

10 FROM Accounts

11 WHERE AccountID = p\_from\_account;

12

13 IF v\_balance >= p\_amount THEN

14 UPDATE Accounts

15 SET Balance = Balance - p\_amount

16 WHERE AccountID = p\_from\_account;

17 UPDATE Accounts

18 SET Balance = Balance + p\_amount

19 WHERE AccountID = p\_to\_account;

20 COMMIT;

21 ELSE

22 RAISE\_APPLICATION\_ERROR(-20001, 'Insufficient funds in source account');

23 END IF;

24 END;

25 /

Procedure created.

SQL> SET SERVEROUTPUT ON;

SQL> DECLARE

2 v\_from\_balance NUMBER;

3 v\_to\_balance NUMBER;

4 BEGIN

5 TransferFunds(1,2,200);

6 SELECT Balance INTO v\_from\_balance FROM Accounts WHERE AccountID=1;

7 SELECT Balance INTO v\_to\_balance FROM Accounts WHERE AccountID=2;

8 DBMS\_OUTPUT.PUT\_LINE('Account 1 Balance: '||v\_from\_balance);

9 DBMS\_OUTPUT.PUT\_LINE('Account 2 Balance: '||v\_to\_balance);

10 END;

11 /

Account 1 Balance: 600

Account 2 Balance: 1900

PL/SQL procedure successfully completed.