# PL/SQL Exercise

Exercise 1- Control Structures

Code

BEGIN

EXECUTE IMMEDIATE 'DROP TABLE Transactions';

EXECUTE IMMEDIATE 'DROP TABLE Accounts';

EXECUTE IMMEDIATE 'DROP TABLE Loans';

EXECUTE IMMEDIATE 'DROP TABLE Employees';

EXECUTE IMMEDIATE 'DROP TABLE Customers';

EXCEPTION

WHEN OTHERS THEN NULL;

END;

/

-- ========================================

-- TABLE CREATION

-- ========================================

CREATE TABLE Customers (

CustomerID NUMBER PRIMARY KEY,

Name VARCHAR2(100),

DOB DATE,

Balance NUMBER,

LastModified DATE

);

CREATE TABLE Accounts (

AccountID NUMBER PRIMARY KEY,

CustomerID NUMBER,

AccountType VARCHAR2(20),

Balance NUMBER,

LastModified DATE,

FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)

);

CREATE TABLE Transactions (

TransactionID NUMBER PRIMARY KEY,

AccountID NUMBER,

TransactionDate DATE,

Amount NUMBER,

TransactionType VARCHAR2(10),

FOREIGN KEY (AccountID) REFERENCES Accounts(AccountID)

);

CREATE TABLE Loans (

LoanID NUMBER PRIMARY KEY,

CustomerID NUMBER,

LoanAmount NUMBER,

InterestRate NUMBER,

StartDate DATE,

EndDate DATE,

FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)

);

CREATE TABLE Employees (

EmployeeID NUMBER PRIMARY KEY,

Name VARCHAR2(100),

Position VARCHAR2(50),

Salary NUMBER,

Department VARCHAR2(50),

HireDate DATE

);

-- ========================================

-- SAMPLE DATA INSERTION

-- ========================================

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

VALUES (1, 'John Doe', TO\_DATE('1985-05-15', 'YYYY-MM-DD'), 1000, SYSDATE);

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

VALUES (2, 'Jane Smith', TO\_DATE('1990-07-20', 'YYYY-MM-DD'), 1500, SYSDATE);

INSERT INTO Accounts (AccountID, CustomerID, AccountType, Balance, LastModified)

VALUES (1, 1, 'Savings', 1000, SYSDATE);

INSERT INTO Accounts (AccountID, CustomerID, AccountType, Balance, LastModified)

VALUES (2, 2, 'Checking', 1500, SYSDATE);

INSERT INTO Transactions (TransactionID, AccountID, TransactionDate, Amount, TransactionType)

VALUES (1, 1, SYSDATE, 200, 'Deposit');

INSERT INTO Transactions (TransactionID, AccountID, TransactionDate, Amount, TransactionType)

VALUES (2, 2, SYSDATE, 300, 'Withdrawal');

INSERT INTO Loans (LoanID, CustomerID, LoanAmount, InterestRate, StartDate, EndDate)

VALUES (1, 1, 5000, 5, SYSDATE, ADD\_MONTHS(SYSDATE, 60));

INSERT INTO Employees (EmployeeID, Name, Position, Salary, Department, HireDate)

VALUES (1, 'Alice Johnson', 'Manager', 70000, 'HR', TO\_DATE('2015-06-15', 'YYYY-MM-DD'));

INSERT INTO Employees (EmployeeID, Name, Position, Salary, Department, HireDate)

VALUES (2, 'Bob Brown', 'Developer', 60000, 'IT', TO\_DATE('2017-03-20', 'YYYY-MM-DD'));

COMMIT;

--SCENARIO 1: Discount 1% on Loan Interest if age > 60

CREATE OR REPLACE PROCEDURE ApplySeniorInterestDiscount IS

BEGIN

FOR rec IN (

SELECT c.CustomerID, l.LoanID

FROM Customers c

JOIN Loans l ON c.CustomerID = l.CustomerID

WHERE MONTHS\_BETWEEN(SYSDATE, c.DOB) / 12 > 60)

LOOP

UPDATE Loans

SET InterestRate = InterestRate - 1

WHERE LoanID = rec.LoanID;

DBMS\_OUTPUT.PUT\_LINE('1% interest discount applied for Customer ID: ' || rec.CustomerID);

END LOOP;

END;

/

-- SCENARIO 2: Promote to VIP if Balance > $10,000

ALTER TABLE Customers ADD IsVIP VARCHAR2(5);

CREATE OR REPLACE PROCEDURE PromoteVIPCustomers IS

BEGIN

FOR rec IN (

SELECT CustomerID, Balance FROM Customers WHERE Balance > 10000

) LOOP

UPDATE Customers

SET IsVIP = 'TRUE'

WHERE CustomerID = rec.CustomerID;

DBMS\_OUTPUT.PUT\_LINE('Customer ID ' || rec.CustomerID || ' promoted to VIP.');

END LOOP;

END;

/

-- SCENARIO 3: Remind Customers if Loan Due in Next 30 Days

CREATE OR REPLACE PROCEDURE SendLoanDueReminders IS

BEGIN

FOR rec IN (

SELECT LoanID, CustomerID, EndDate

FROM Loans

WHERE EndDate BETWEEN SYSDATE AND SYSDATE + 30

) LOOP

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

' for Customer ID ' || rec.CustomerID ||

' is due on ' || TO\_CHAR(rec.EndDate, 'DD-MON-YYYY'));

END LOOP;

END;

/

-- ========================================

-- PROCEDURE CALLS

-- ========================================

-- 1. Apply 1% interest discount to senior customers

BEGIN

ApplySeniorInterestDiscount;

END;

/

-- 2. Promote customers with balance > 10000 to VIP

BEGIN

PromoteVIPCustomers;

END;

/

-- 3. Send reminders for loans due within 30 days

BEGIN

SendLoanDueReminders;

END;

/

-- ========================================

-- VIEW RESULTS

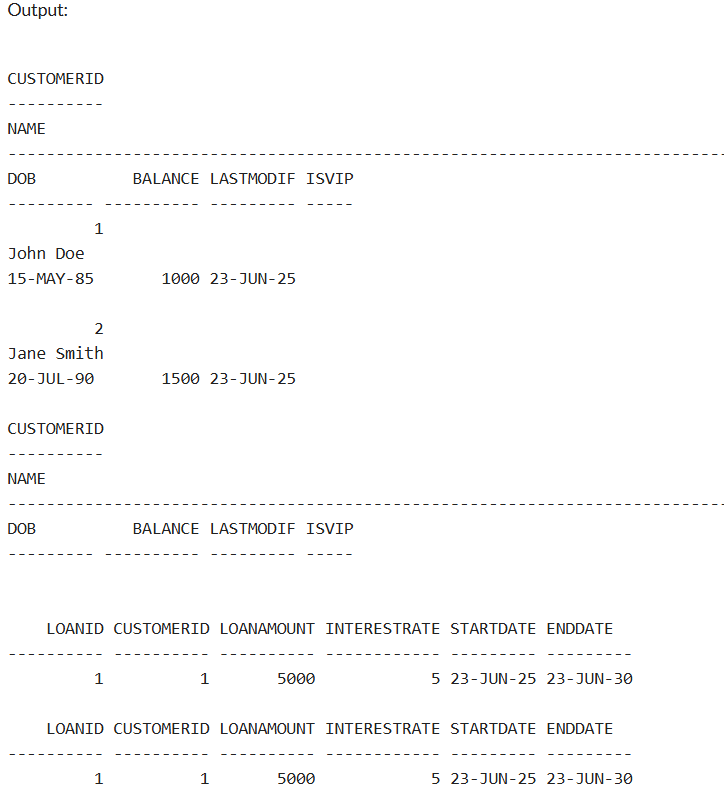
-- ========================================

-- View updated Customers table (to check VIP)

SELECT \* FROM Customers;

-- View updated Loans table (to check interest rate changes)

SELECT \* FROM Loans;



Exercise 3- Stored Procedures  
Code

CREATE TABLE Customers (

CustomerID NUMBER PRIMARY KEY,

Name VARCHAR2(100),

DOB DATE,

Balance NUMBER,

LastModified DATE

);

CREATE TABLE Accounts (

AccountID NUMBER PRIMARY KEY,

CustomerID NUMBER,

AccountType VARCHAR2(20),

Balance NUMBER,

LastModified DATE,

FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)

);

CREATE TABLE Transactions (

TransactionID NUMBER PRIMARY KEY,

AccountID NUMBER,

TransactionDate DATE,

Amount NUMBER,

TransactionType VARCHAR2(10),

FOREIGN KEY (AccountID) REFERENCES Accounts(AccountID)

);

CREATE TABLE Loans (

LoanID NUMBER PRIMARY KEY,

CustomerID NUMBER,

LoanAmount NUMBER,

InterestRate NUMBER,

StartDate DATE,

EndDate DATE,

FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)

);

CREATE TABLE Employees (

EmployeeID NUMBER PRIMARY KEY,

Name VARCHAR2(100),

Position VARCHAR2(50),

Salary NUMBER,

Department VARCHAR2(50),

HireDate DATE

);

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

VALUES (1, 'John Doe', TO\_DATE('1985-05-15', 'YYYY-MM-DD'), 1000, SYSDATE);

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

VALUES (2, 'Jane Smith', TO\_DATE('1990-07-20', 'YYYY-MM-DD'), 1500, SYSDATE);

INSERT INTO Accounts (AccountID, CustomerID, AccountType, Balance, LastModified)

VALUES (1, 1, 'Savings', 1000, SYSDATE);

INSERT INTO Accounts (AccountID, CustomerID, AccountType, Balance, LastModified)

VALUES (2, 2, 'Checking', 1500, SYSDATE);

INSERT INTO Transactions (TransactionID, AccountID, TransactionDate, Amount, TransactionType)

VALUES (1, 1, SYSDATE, 200, 'Deposit');

INSERT INTO Transactions (TransactionID, AccountID, TransactionDate, Amount, TransactionType)

VALUES (2, 2, SYSDATE, 300, 'Withdrawal');

INSERT INTO Loans (LoanID, CustomerID, LoanAmount, InterestRate, StartDate, EndDate)

VALUES (1, 1, 5000, 5, SYSDATE, ADD\_MONTHS(SYSDATE, 60));

INSERT INTO Employees (EmployeeID, Name, Position, Salary, Department, HireDate)

VALUES (1, 'Alice Johnson', 'Manager', 70000, 'HR', TO\_DATE('2015-06-15', 'YYYY-MM-DD'));

INSERT INTO Employees (EmployeeID, Name, Position, Salary, Department, HireDate)

VALUES (2, 'Bob Brown', 'Developer', 60000, 'IT', TO\_DATE('2017-03-20', 'YYYY-MM-DD'));

-- ============================================================

-- SCENARIO 1: Process Monthly Interest (1% for Savings Accounts)

-- ============================================================

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest IS

BEGIN

FOR rec IN (

SELECT AccountID, Balance

FROM Accounts

WHERE AccountType = 'Savings'

) LOOP

UPDATE Accounts

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

LastModified = SYSDATE

WHERE AccountID = rec.AccountID;

DBMS\_OUTPUT.PUT\_LINE('Interest applied to Account ID: ' || rec.AccountID);

END LOOP;

END;

/

-- ============================================================

-- SCENARIO 2: Update Employee Bonus (Add bonus % to a department)

-- ============================================================

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus (

dept IN VARCHAR2,

bonus\_percent IN NUMBER

) IS

BEGIN

FOR rec IN (

SELECT EmployeeID, Salary

FROM Employees

WHERE Department = dept

) LOOP

UPDATE Employees

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

WHERE EmployeeID = rec.EmployeeID;

DBMS\_OUTPUT.PUT\_LINE('Bonus applied to Employee ID: ' || rec.EmployeeID);

END LOOP;

END;

/

-- ============================================================

-- SCENARIO 3: Transfer Funds Between Accounts

-- ============================================================

CREATE OR REPLACE PROCEDURE TransferFunds (

from\_account IN NUMBER,

to\_account IN NUMBER,

amount IN NUMBER

) IS

v\_balance NUMBER;

BEGIN

-- Get source account balance

SELECT Balance INTO v\_balance

FROM Accounts

WHERE AccountID = from\_account;

-- Check if sufficient balance

IF v\_balance < amount THEN

DBMS\_OUTPUT.PUT\_LINE('Transfer failed: Insufficient funds in Account ID ' || from\_account);

ELSE

-- Deduct from source account

UPDATE Accounts

SET Balance = Balance - amount,

LastModified = SYSDATE

WHERE AccountID = from\_account;

-- Credit to destination account

UPDATE Accounts

SET Balance = Balance + amount,

LastModified = SYSDATE

WHERE AccountID = to\_account;

DBMS\_OUTPUT.PUT\_LINE('Transferred ' || amount ||

' from Account ' || from\_account ||

' to Account ' || to\_account);

END IF;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Transfer failed: Account not found.');

END;

/

-- ============================================================

-- PROCEDURE CALLS

-- ============================================================

-- Scenario 1: Apply monthly interest

BEGIN

ProcessMonthlyInterest;

END;

/

-- Scenario 2: Apply 10% bonus to 'IT' department

BEGIN

UpdateEmployeeBonus('IT', 10);

END;

/

-- Scenario 3: Transfer 300 from Account 1 to Account 2

BEGIN

TransferFunds(1, 2, 300);

END;

/

-- ============================================================

-- CHECK UPDATED DATA

-- ============================================================

-- Accounts table after interest and transfer

SELECT \* FROM Accounts;

-- Employees table after bonus

SELECT \* FROM Employees;

