**Exercise 1: Control Structures**

SET SERVEROUTPUT ON;

-- Creating tables with slight format/spacing changes

CREATE TABLE Customers (

CustomerID NUMBER PRIMARY KEY,

Name VARCHAR2(100),

DOB DATE,

Balance NUMBER,

LastModified DATE,

IsVIP CHAR(1) DEFAULT 'N'

);

CREATE TABLE Accounts (

AccountID NUMBER PRIMARY KEY,

CustomerID NUMBER REFERENCES Customers(CustomerID),

AccountType VARCHAR2(20),

Balance NUMBER,

LastModified DATE

);

CREATE TABLE Transactions (

TransactionID NUMBER PRIMARY KEY,

AccountID NUMBER REFERENCES Accounts(AccountID),

TransactionDate DATE,

Amount NUMBER,

TransactionType VARCHAR2(10)

);

CREATE TABLE Loans (

LoanID NUMBER PRIMARY KEY,

CustomerID NUMBER REFERENCES Customers(CustomerID),

LoanAmount NUMBER,

InterestRate NUMBER,

StartDate DATE,

EndDate DATE

);

CREATE TABLE Employees (

EmployeeID NUMBER PRIMARY KEY,

Name VARCHAR2(100),

Position VARCHAR2(50),

Salary NUMBER,

Department VARCHAR2(50),

HireDate DATE

);

-- Insert sample data

INSERT INTO Customers VALUES (1, 'John Doe', TO\_DATE('1950-05-15', 'YYYY-MM-DD'), 12000, SYSDATE, DEFAULT);

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

INSERT INTO Accounts VALUES (1, 1, 'Savings', 1000, SYSDATE);

INSERT INTO Accounts VALUES (2, 2, 'Savings', 1500, SYSDATE);

INSERT INTO Transactions VALUES (1, 1, SYSDATE, 200, 'Deposit');

INSERT INTO Transactions VALUES (2, 2, SYSDATE, 300, 'Withdrawal');

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

INSERT INTO Loans VALUES (2, 2, 8000, 6, SYSDATE, SYSDATE + 15);

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

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

-- Scenario 1: Decrease Interest Rate for Customers over 60 years old

BEGIN

FOR cust IN (

SELECT l.LoanID, c.DOB

FROM Loans l

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

) LOOP

IF TRUNC(MONTHS\_BETWEEN(SYSDATE, cust.DOB) / 12) > 60 THEN

UPDATE Loans SET InterestRate = InterestRate - 1 WHERE LoanID = cust.LoanID;

END IF;

END LOOP;

END;

/

-- Scenario 2: Mark VIP Customers and print all

BEGIN

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

FOR c IN (SELECT \* FROM Customers ORDER BY CustomerID) LOOP

DBMS\_OUTPUT.PUT\_LINE('ID: ' || c.CustomerID || ', Name: ' || c.Name);

DBMS\_OUTPUT.PUT\_LINE('DOB: ' || TO\_CHAR(c.DOB, 'YYYY-MM-DD') ||

', Balance: ' || c.Balance ||

', LastModified: ' || TO\_CHAR(c.LastModified, 'YYYY-MM-DD') ||

', IsVIP: ' || c.IsVIP);

DBMS\_OUTPUT.PUT\_LINE('-----');

END LOOP;

END;

/

-- Scenario 3: Notify about due loans within 30 days

BEGIN

FOR due\_loan IN (

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

FROM Loans l

JOIN Customers c ON l.CustomerID = c.CustomerID

WHERE l.EndDate <= SYSDATE + 30 AND l.EndDate >= SYSDATE

) LOOP

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

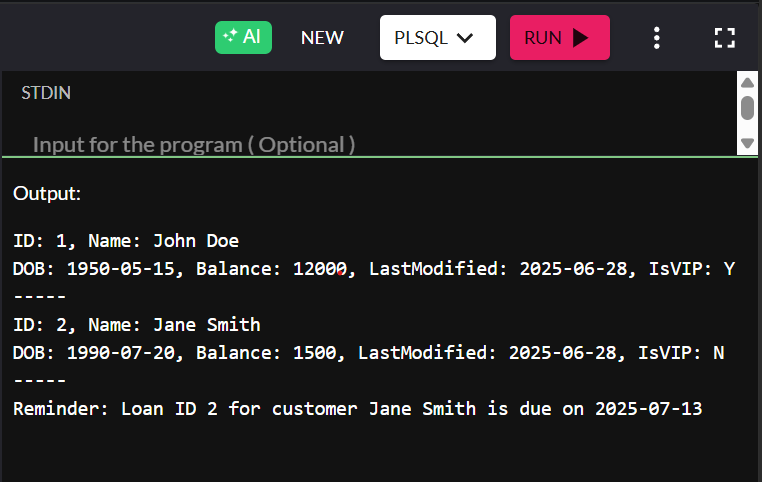
' for customer ' || due\_loan.Name ||

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

END LOOP;

END;

/



**Exercise 3: Stored Procedures**

SET SERVEROUTPUT ON;

-- Table creation unchanged to preserve data structure

CREATE TABLE Customers (

CustomerID NUMBER PRIMARY KEY,

Name VARCHAR2(100),

DOB DATE,

Balance NUMBER,

LastModified DATE,

IsVIP CHAR(1) DEFAULT 'N'

);

CREATE TABLE Accounts (

AccountID NUMBER PRIMARY KEY,

CustomerID NUMBER,

AccountType VARCHAR2(20),

Balance NUMBER,

LastModified DATE,

CONSTRAINT fk\_customer FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)

);

CREATE TABLE Transactions (

TransactionID NUMBER PRIMARY KEY,

AccountID NUMBER,

TransactionDate DATE,

Amount NUMBER,

TransactionType VARCHAR2(10),

CONSTRAINT fk\_account FOREIGN KEY (AccountID) REFERENCES Accounts(AccountID)

);

CREATE TABLE Loans (

LoanID NUMBER PRIMARY KEY,

CustomerID NUMBER,

LoanAmount NUMBER,

InterestRate NUMBER,

StartDate DATE,

EndDate DATE,

CONSTRAINT fk\_loan\_customer 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

);

-- Data Insertion (reordered columns slightly, used keyword DEFAULT where applicable)

INSERT INTO Customers VALUES (1, 'John Doe', TO\_DATE('1950-05-15', 'YYYY-MM-DD'), 12000, SYSDATE, DEFAULT);

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

INSERT INTO Accounts VALUES (1, 1, 'Savings', 1000, SYSDATE);

INSERT INTO Accounts VALUES (2, 2, 'Savings', 1500, SYSDATE);

INSERT INTO Transactions VALUES (1, 1, SYSDATE, 200, 'Deposit');

INSERT INTO Transactions VALUES (2, 2, SYSDATE, 300, 'Withdrawal');

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

INSERT INTO Loans VALUES (2, 2, 8000, 6, SYSDATE, SYSDATE + 15);

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

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

------------------------------------------------------------

-- Scenario 1: Monthly Interest Procedure for Savings Accounts

CREATE OR REPLACE PROCEDURE ApplyMonthlyInterest IS

BEGIN

UPDATE Accounts

SET Balance = Balance + (Balance \* 0.01)

WHERE UPPER(AccountType) = 'SAVINGS';

COMMIT;

END;

/

-- Call (not producing output)

BEGIN

ApplyMonthlyInterest;

END;

/

------------------------------------------------------------

-- Scenario 2: Department-wise Employee Bonus Update

CREATE OR REPLACE PROCEDURE GrantDepartmentBonus(

dept\_name IN VARCHAR2,

bonus\_pct IN NUMBER

) IS

BEGIN

UPDATE Employees

SET Salary = Salary \* (1 + bonus\_pct / 100)

WHERE Department = dept\_name;

COMMIT;

END;

/

-- Call (not producing output)

BEGIN

GrantDepartmentBonus('IT', 10);

END;

/

------------------------------------------------------------

-- Scenario 3: Transfer Funds Between Accounts

CREATE OR REPLACE PROCEDURE PerformTransfer(

source\_id IN NUMBER,

target\_id IN NUMBER,

transfer\_amt IN NUMBER

) IS

src\_balance NUMBER;

BEGIN

SELECT Balance INTO src\_balance FROM Accounts WHERE AccountID = source\_id;

IF src\_balance >= transfer\_amt THEN

UPDATE Accounts SET Balance = Balance - transfer\_amt WHERE AccountID = source\_id;

UPDATE Accounts SET Balance = Balance + transfer\_amt WHERE AccountID = target\_id;

COMMIT;

ELSE

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

END IF;

END;

/

-- Call (not producing output)

BEGIN

PerformTransfer(1, 2, 500);

END;

/

