**Exercise 1: Control Structures**

**Scenario 1:** The bank wants to apply a discount to loan interest rates for customers above 60 years old.

* + **Question:** Write a PL/SQL block that loops through all customers, checks their age, and if they are above 60, apply a 1% discount to their current loan interest rates.

**Scenario 2:** A customer can be promoted to VIP status based on their balance.

* + **Question:** Write a PL/SQL block that iterates through all customers and sets a flag IsVIP to TRUE for those with a balance over $10,000.

**Scenario 3:** The bank wants to send reminders to customers whose loans are due within the next 30 days.

* + **Question:** Write a PL/SQL block that fetches all loans due in the next 30 days and prints a reminder message for each customer.

**SOLUTION:**

SET SERVEROUTPUT ON;

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,

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('1950-05-15', 'YYYY-MM-DD'), 12000, 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, 'Savings', 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 Loans (LoanID, CustomerID, LoanAmount, InterestRate, StartDate, EndDate)

VALUES (2, 2, 8000, 6, SYSDATE, SYSDATE + 15); -- Due within 30 days

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'));

**-- 1st scenario**

**BEGIN**

**FOR Rec IN (**

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

**FROM Customers c**

**JOIN Loans l ON c.CustomerID = l.CustomerID**

**) LOOP**

**IF FLOOR(MONTHS\_BETWEEN(SYSDATE, Rec.DOB) / 12) > 60 THEN**

**UPDATE Loans**

**SET InterestRate = InterestRate - 1**

**WHERE LoanID = Rec.LoanID;**

**END IF;**

**END LOOP;**

**END;**

**/**

**-- 2nd scenario**

**BEGIN**

**UPDATE Customers**

**SET IsVIP = 'Y'**

**WHERE Balance > 10000;**

**FOR Rec IN (SELECT \* FROM Customers) LOOP**

**DBMS\_OUTPUT.PUT\_LINE(**

**'ID: ' || Rec.CustomerID ||**

**', Name: ' || Rec.Name ||**

**', DOB: ' || TO\_CHAR(Rec.DOB, 'YYYY-MM-DD') ||**

**', Balance: ' || Rec.Balance ||**

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

**', IsVIP: ' || Rec.IsVIP**

**);**

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

**END LOOP;**

**END;**

**/**

**-- 3rd scenario**

**BEGIN**

**FOR Rec IN (**

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

**FROM Loans l**

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

**WHERE l.EndDate BETWEEN SYSDATE AND SYSDATE + 30**

**) LOOP**

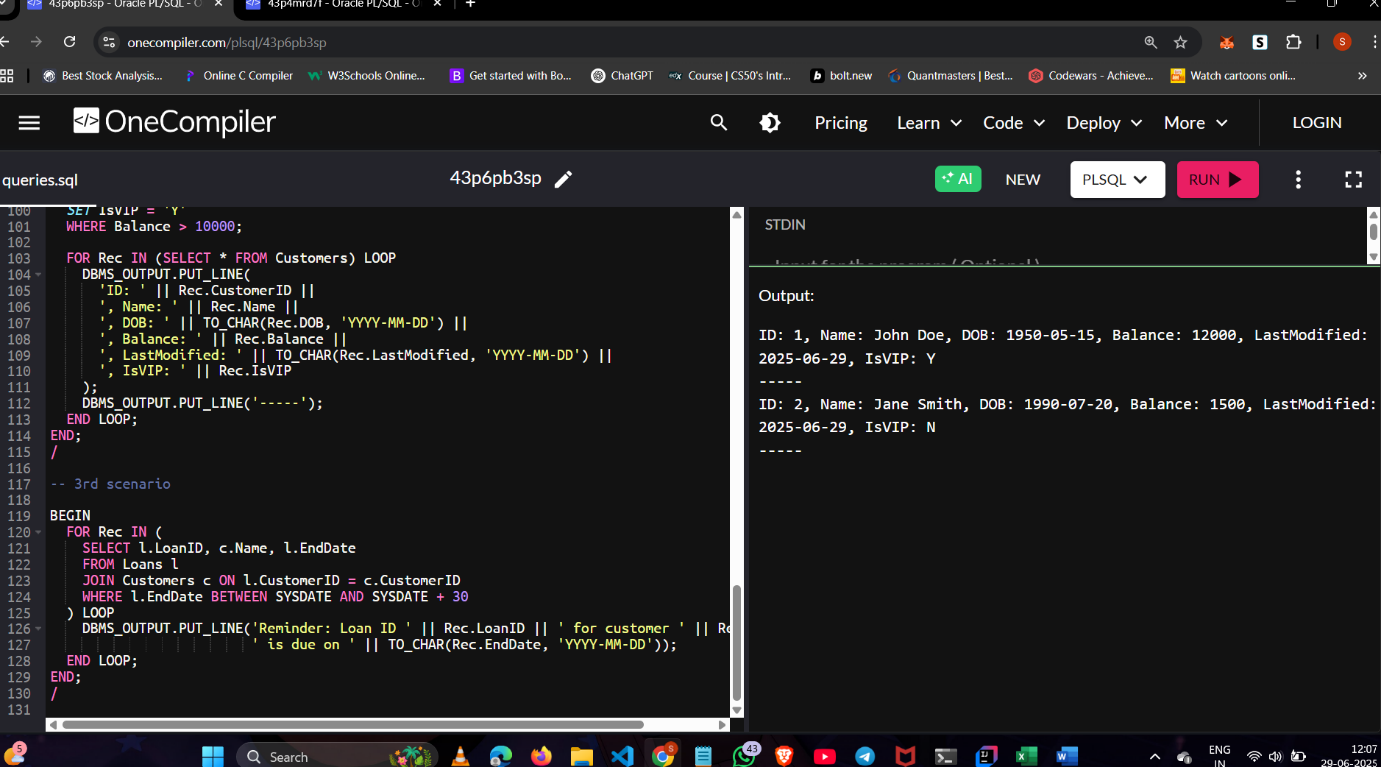
**DBMS\_OUTPUT.PUT\_LINE('Reminder: Loan ID ' || Rec.LoanID || ' for customer ' || Rec.Name ||' is due on ' || TO\_CHAR(Rec.EndDate, 'YYYY-MM-DD'));**

**END LOOP;**

**END;**

**/**

**OUTPUT:**



**Exercise 3: Stored Procedures**

**Scenario 1: The bank needs to process monthly interest for all savings accounts.**

* + **Question: Write a stored procedure ProcessMonthlyInterest that calculates and updates the balance of all savings accounts by applying an interest rate of 1% to the current balance.**

**Scenario 2: The bank wants to implement a bonus scheme for employees based on their performance.**

* + **Question: Write a stored procedure UpdateEmployeeBonus that updates the salary of employees in a given department by adding a bonus percentage passed as a parameter.**

**Scenario 3: Customers should be able to transfer funds between their accounts.**

* + **Question: Write a stored procedure TransferFunds that transfers a specified amount from one account to another, checking that the source account has sufficient balance before making the transfer.**

**SOLUTION:**

SET SERVEROUTPUT ON;

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,

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('1950-05-15', 'YYYY-MM-DD'), 12000, 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, 'Savings', 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 Loans (LoanID, CustomerID, LoanAmount, InterestRate, StartDate, EndDate)

VALUES (2, 2, 8000, 6, SYSDATE, SYSDATE + 15); -- Due within 30 days

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'));

**-- first scenario**

**CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest AS**

**BEGIN**

**UPDATE Accounts**

**SET Balance = Balance \* 1.01**

**WHERE AccountType = 'Savings';**

**COMMIT;**

**END;**

**/**

**-- 2nd scenario**

**CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus(**

**p\_department IN VARCHAR2,**

**p\_bonus\_percent IN NUMBER**

**) AS**

**BEGIN**

**UPDATE Employees**

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

**WHERE Department = p\_department;**

**COMMIT;**

**END;**

**/**

**BEGIN**

**UpdateEmployeeBonus('IT', 10);**

**END;**

**/**

**-- 3rd scenario**

**CREATE OR REPLACE PROCEDURE TransferFunds(**

**p\_from\_account IN NUMBER,**

**p\_to\_account IN NUMBER,**

**p\_amount IN NUMBER**

**) AS**

**v\_balance NUMBER;**

**BEGIN**

**SELECT Balance INTO v\_balance**

**FROM Accounts**

**WHERE AccountID = p\_from\_account;**

**IF v\_balance < p\_amount THEN**

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

**END IF;**

**UPDATE Accounts**

**SET Balance = Balance - p\_amount**

**WHERE AccountID = p\_from\_account;**

**UPDATE Accounts**

**SET Balance = Balance + p\_amount**

**WHERE AccountID = p\_to\_account;**

**COMMIT;**

**END;**

**/**

**BEGIN**

**TransferFunds(1, 2, 500);**

**END;**

**/**

**OUTPUT:**

