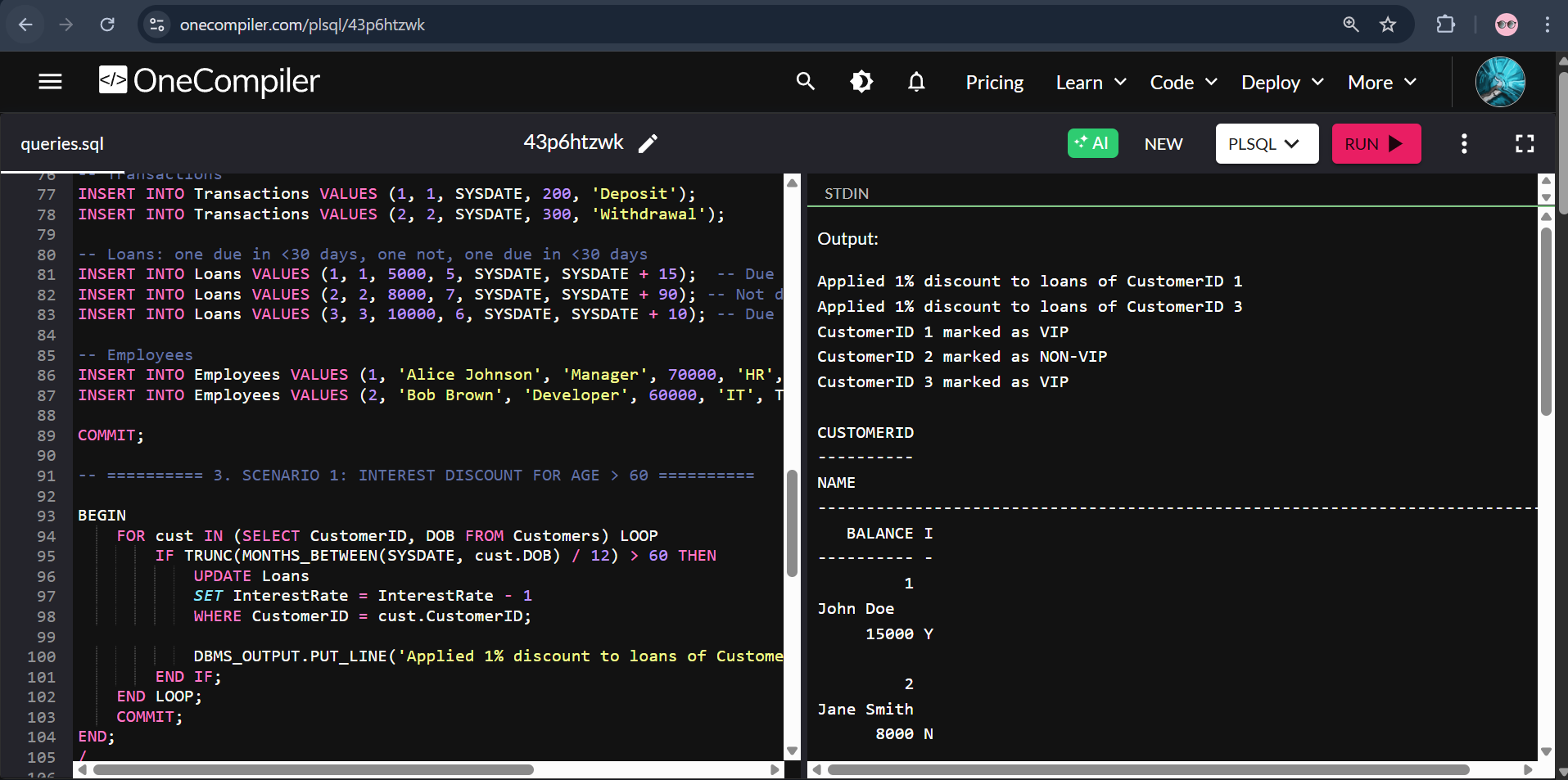
PL/SQL  
  
Exercise 1  
  
Output  


Code  
  
-- Enable output

SET SERVEROUTPUT ON;

-- Drop tables if they exist (safe for rerun)

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;

/

-- ========== 1. TABLE CREATION ==========

CREATE TABLE Customers (

    CustomerID NUMBER PRIMARY KEY,

    Name VARCHAR2(100),

    DOB DATE,

    Balance NUMBER,

    LastModified DATE,

    IsVIP CHAR(1)

);

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

);

-- ========== 2. SAMPLE DATA ==========

-- Elderly and young customers, some VIP candidates

INSERT INTO Customers VALUES (1, 'John Doe', TO\_DATE('1950-05-15', 'YYYY-MM-DD'), 15000, SYSDATE, NULL);

INSERT INTO Customers VALUES (2, 'Jane Smith', TO\_DATE('1990-07-20', 'YYYY-MM-DD'), 8000, SYSDATE, NULL);

INSERT INTO Customers VALUES (3, 'Elder Joe', TO\_DATE('1940-03-01', 'YYYY-MM-DD'), 12000, SYSDATE, NULL);

-- Accounts

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

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

INSERT INTO Accounts VALUES (3, 3, 'Savings', 2000, SYSDATE);

-- Transactions

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

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

-- Loans: 2 due in <30 days, 1 not

INSERT INTO Loans VALUES (1, 1, 5000, 5, SYSDATE, SYSDATE + 15);  -- Due soon

INSERT INTO Loans VALUES (2, 2, 8000, 7, SYSDATE, SYSDATE + 90); -- Not due soon

INSERT INTO Loans VALUES (3, 3, 10000, 6, SYSDATE, SYSDATE + 10); -- Due soon

-- Employees

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'));

COMMIT;

-- ========== 3. SCENARIO 1: INTEREST DISCOUNT FOR AGE > 60 ==========

BEGIN

    FOR cust IN (SELECT CustomerID, DOB FROM Customers) LOOP

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

            UPDATE Loans

            SET InterestRate = InterestRate - 1

            WHERE CustomerID = cust.CustomerID;

            DBMS\_OUTPUT.PUT\_LINE('✅ Applied 1% interest discount for CustomerID ' || cust.CustomerID);

        END IF;

    END LOOP;

    COMMIT;

END;

/

-- ========== 4. SCENARIO 2: SET IsVIP = 'Y' IF BALANCE > 10000 ==========

BEGIN

    FOR cust IN (SELECT CustomerID, Balance FROM Customers) LOOP

        IF cust.Balance > 10000 THEN

            UPDATE Customers SET IsVIP = 'Y' WHERE CustomerID = cust.CustomerID;

            DBMS\_OUTPUT.PUT\_LINE('🌟 CustomerID ' || cust.CustomerID || ' marked as VIP');

        ELSE

            UPDATE Customers SET IsVIP = 'N' WHERE CustomerID = cust.CustomerID;

            DBMS\_OUTPUT.PUT\_LINE('ℹ️ CustomerID ' || cust.CustomerID || ' marked as Non-VIP');

        END IF;

    END LOOP;

    COMMIT;

END;

/

-- ========== 5. SCENARIO 3: LOAN DUE REMINDERS (WITHIN 30 DAYS) ==========

BEGIN

    FOR loan IN (

        SELECT l.CustomerID, l.EndDate

        FROM Loans l

        WHERE l.EndDate <= SYSDATE + 30

    ) LOOP

        DECLARE

            v\_name Customers.Name%TYPE;

        BEGIN

            SELECT Name INTO v\_name FROM Customers WHERE CustomerID = loan.CustomerID;

            DBMS\_OUTPUT.PUT\_LINE('📢 Reminder: Loan for ' || v\_name || ' is due on ' ||

                                 TO\_CHAR(loan.EndDate, 'YYYY-MM-DD'));

        END;

    END LOOP;

END;

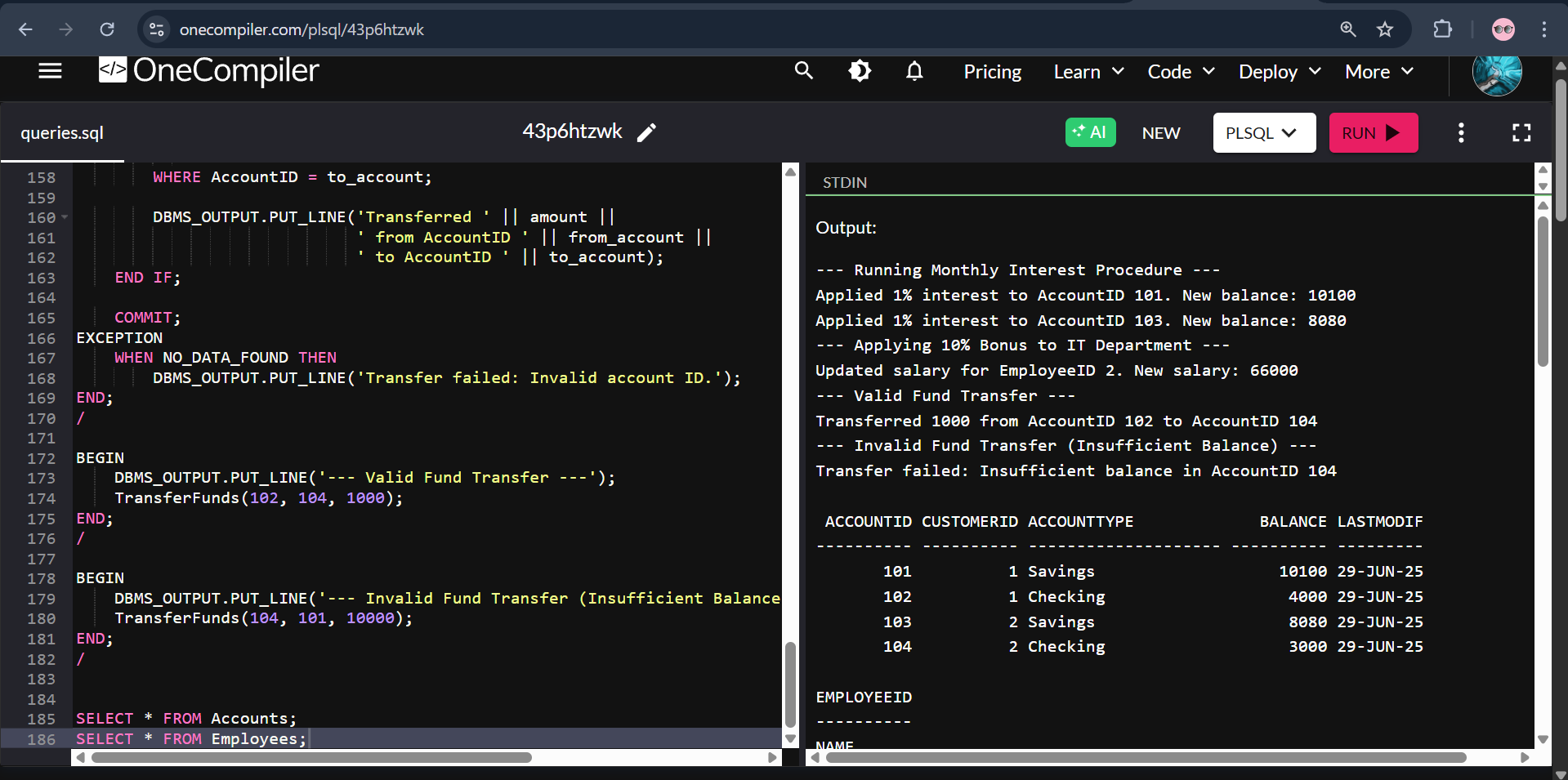
/

-- ========== 6. OPTIONAL: FINAL STATE CHECK (USE MANUALLY) ==========

-- SELECT \* FROM Loans;

-- SELECT CustomerID, Name, Balance, IsVIP FROM Customers;

Exercise 3  
  
Output

  
  
  
Code  
  
-- Enable DBMS Output

SET SERVEROUTPUT ON;

-- Drop existing tables if they exist

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;

/

-- ========== 1. TABLE CREATION ==========

CREATE TABLE Customers (

CustomerID NUMBER PRIMARY KEY,

Name VARCHAR2(100),

DOB DATE,

Balance NUMBER,

LastModified DATE,

IsVIP CHAR(1)

);

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

);

-- ========== 2. SAMPLE DATA ==========

-- Customers

INSERT INTO Customers VALUES (1, 'John Doe', TO\_DATE('1985-05-15', 'YYYY-MM-DD'), 0, SYSDATE, NULL);

INSERT INTO Customers VALUES (2, 'Jane Smith', TO\_DATE('1990-07-20', 'YYYY-MM-DD'), 0, SYSDATE, NULL);

-- Accounts

INSERT INTO Accounts VALUES (101, 1, 'Savings', 10000, SYSDATE);

INSERT INTO Accounts VALUES (102, 1, 'Checking', 5000, SYSDATE);

INSERT INTO Accounts VALUES (103, 2, 'Savings', 8000, SYSDATE);

INSERT INTO Accounts VALUES (104, 2, 'Checking', 2000, SYSDATE);

-- Employees

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'));

COMMIT;

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

-- ========== SCENARIO 1: Interest ==========

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

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest AS

BEGIN

FOR acc IN (SELECT AccountID, Balance FROM Accounts WHERE AccountType = 'Savings') LOOP

UPDATE Accounts

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

LastModified = SYSDATE

WHERE AccountID = acc.AccountID;

DBMS\_OUTPUT.PUT\_LINE('Applied 1% interest to AccountID ' || acc.AccountID ||

'. New balance: ' || TO\_CHAR(acc.Balance \* 1.01));

END LOOP;

COMMIT;

END;

/

BEGIN

DBMS\_OUTPUT.PUT\_LINE('--- Running Monthly Interest Procedure ---');

ProcessMonthlyInterest;

END;

/

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

-- ========== SCENARIO 2: Bonus Update ======

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

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus (

dept\_name IN VARCHAR2,

bonus\_pct IN NUMBER

) AS

BEGIN

FOR emp IN (SELECT EmployeeID, Salary FROM Employees WHERE Department = dept\_name) LOOP

UPDATE Employees

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

WHERE EmployeeID = emp.EmployeeID;

DBMS\_OUTPUT.PUT\_LINE('Updated salary for EmployeeID ' || emp.EmployeeID ||

'. New salary: ' || TO\_CHAR(emp.Salary \* (1 + bonus\_pct / 100)));

END LOOP;

COMMIT;

END;

/

BEGIN

DBMS\_OUTPUT.PUT\_LINE('--- Applying 10% Bonus to IT Department ---');

UpdateEmployeeBonus('IT', 10);

END;

/

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

-- ========== SCENARIO 3: Transfer Funds ====

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

CREATE OR REPLACE PROCEDURE TransferFunds (

from\_account IN NUMBER,

to\_account IN NUMBER,

amount IN NUMBER

) AS

v\_balance NUMBER;

BEGIN

SELECT Balance INTO v\_balance FROM Accounts WHERE AccountID = from\_account FOR UPDATE;

IF v\_balance < amount THEN

DBMS\_OUTPUT.PUT\_LINE('Transfer failed: Insufficient balance in AccountID ' || from\_account);

ELSE

UPDATE Accounts

SET Balance = Balance - amount,

LastModified = SYSDATE

WHERE AccountID = from\_account;

UPDATE Accounts

SET Balance = Balance + amount,

LastModified = SYSDATE

WHERE AccountID = to\_account;

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

' from AccountID ' || from\_account ||

' to AccountID ' || to\_account);

END IF;

COMMIT;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Transfer failed: Invalid account ID.');

END;

/

BEGIN

DBMS\_OUTPUT.PUT\_LINE('--- Valid Fund Transfer ---');

TransferFunds(102, 104, 1000);

END;

/

BEGIN

DBMS\_OUTPUT.PUT\_LINE('--- Invalid Fund Transfer (Insufficient Balance) ---');

TransferFunds(104, 101, 10000);

END;

/

SELECT \* FROM Accounts;

SELECT \* FROM Employees;