**PL/SQL PROGRAMMING**

**Schema to be Created:**

*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*

*);*

**Example Scripts for 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'));*

**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.

**PL/SQL code:**

DECLARE

DAYS NUMBER;

BEGIN

    for rec in (SELECT LOANID,C.CUSTOMERID,DOB FROM LOANS L,CUSTOMERS C WHERE  C.CUSTOMERID=L.CUSTOMERID) loop

        DAYS:=SYSDATE-rec.DOB;

        IF (DAYS/365) > 60 THEN

        UPDATE LOANS

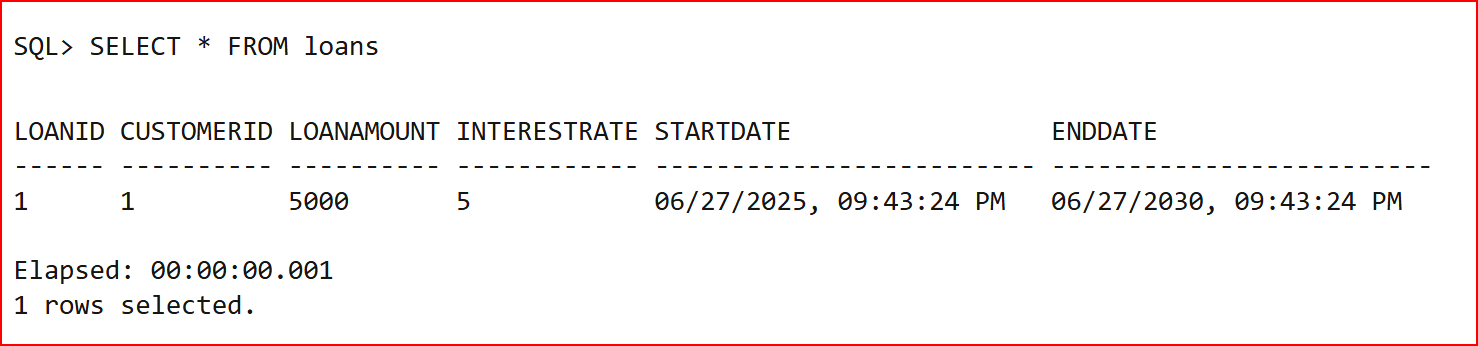
        SET INTERESTRATE=0.99\*INTERESTRATE

        WHERE LOANID=rec.LOANID;

        END IF;

    END LOOP;

END;

******OUTPUT**:

**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.

**PL/SQL code:**

BEGIN

    UPDATE CUSTOMERS

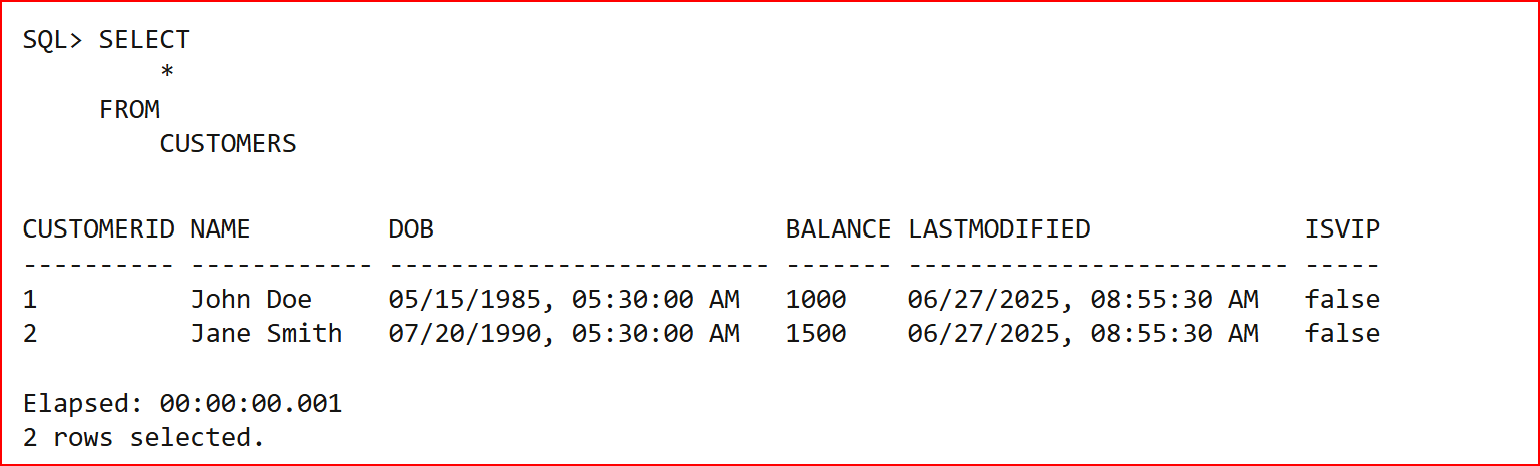
    SET ISVIP=CASE WHEN BALANCE>10000 THEN TRUE

    ELSE FALSE

    END;

END;

**OUTPUT:**



**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.

**PL/SQL code:**

DECLARE NAME VARCHAR(100);

BEGIN

    FOR REC IN (SELECT \* FROM LOANS) LOOP

    SELECT C.NAME INTO NAME FROM CUSTOMERS C WHERE C.CUSTOMERID=REC.CUSTOMERID;

    IF(REC.ENDDATE-SYSDATE < 30) THEN

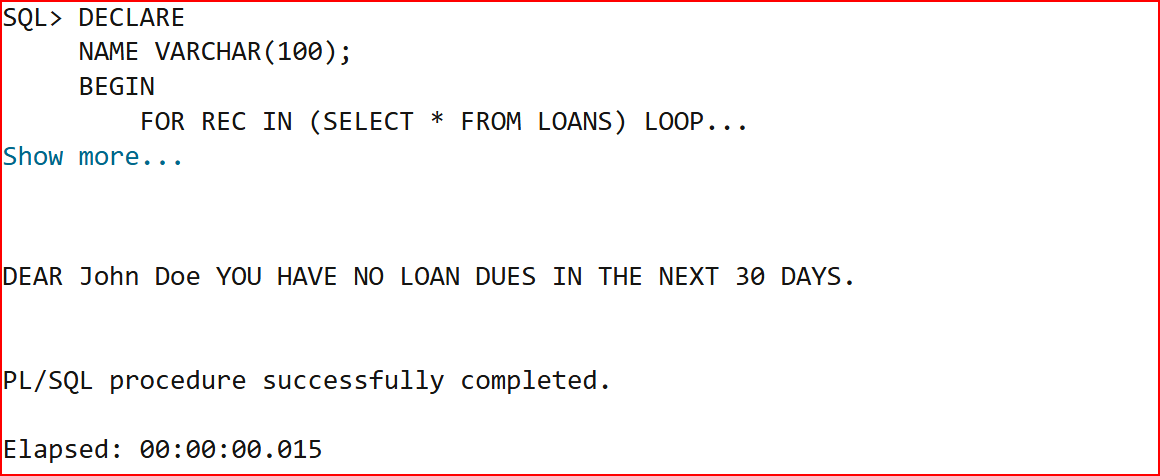
    DBMS\_OUTPUT.PUT\_LINE('DEAR '||NAME||' YOU HAVE LOAN DUE IN '||CEIL(REC.ENDDATE-SYSDATE)|| ' DAYS');

    ELSE  DBMS\_OUTPUT.PUT\_LINE('DEAR '||NAME||' YOU HAVE NO LOAN DUES IN THE NEXT 30 DAYS.');

     END IF;

     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.

**PL/SQL code:**

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest IS

BEGIN

    UPDATE ACCOUNTS

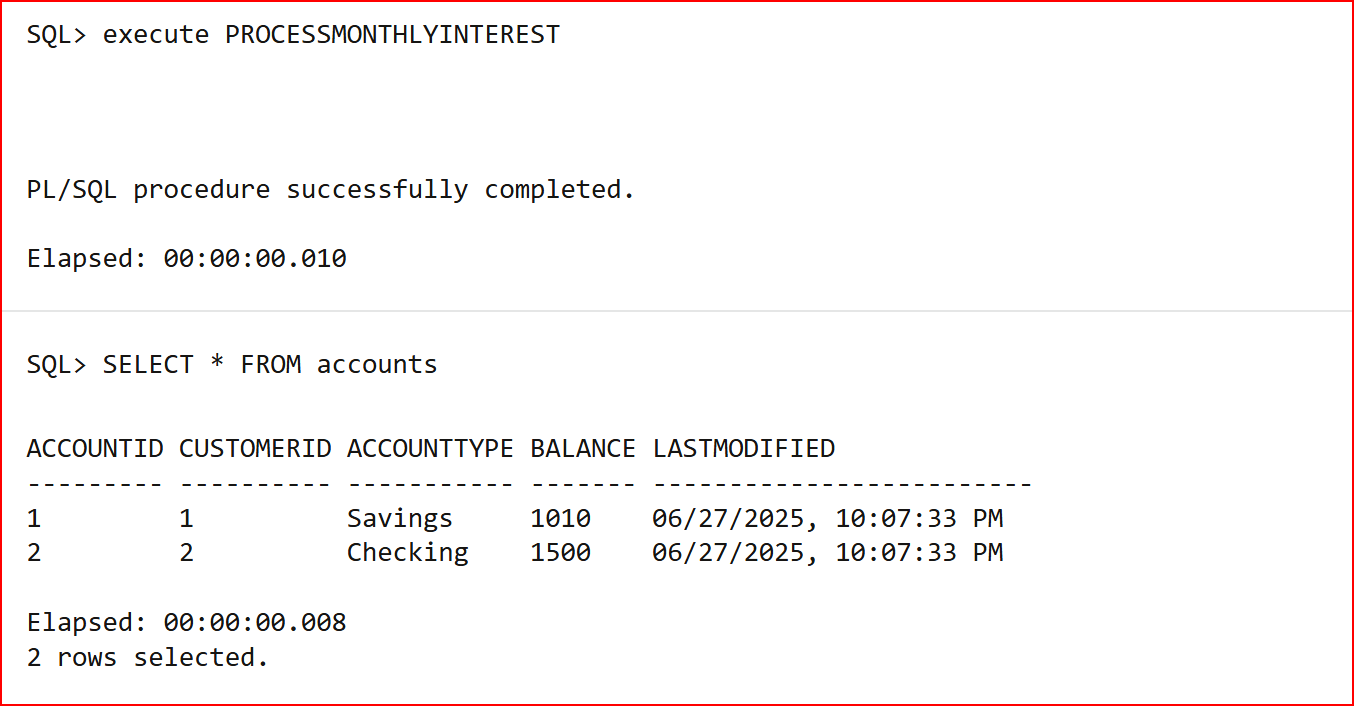
    SET BALANCE=1.01\*BALANCE

    WHERE ACCOUNTTYPE='Savings';

END ProcessMonthlyInterest;

**Executing procedure:**

execute PROCESSMONTHLYINTEREST;

**OUTPUT:**

**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.

**PL/SQL Code:**

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus(PERCENT IN NUMBER, DEPT IN VARCHAR) IS

BEGIN

    UPDATE EMPLOYEES

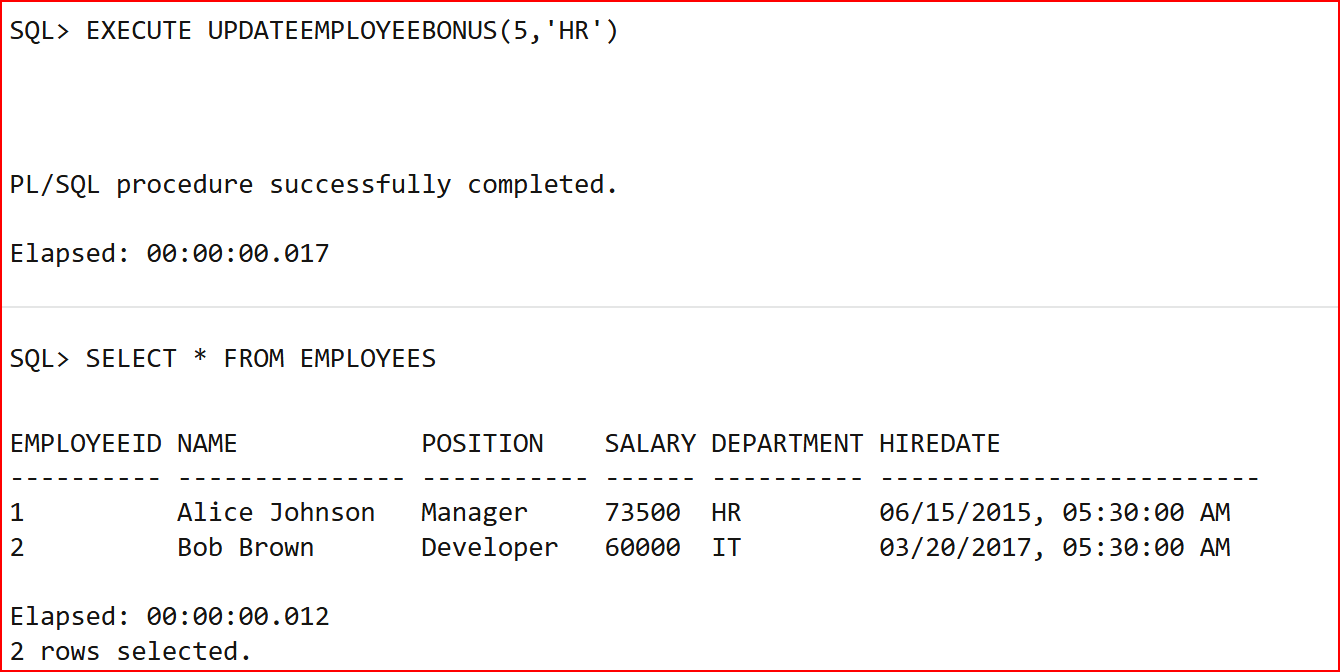
    SET SALARY=SALARY+SALARY\*PERCENT/100

    WHERE DEPARTMENT=DEPT;

END UpdateEmployeeBonus;

**Executing procedure:**

EXECUTE UPDATEEMPLOYEEBONUS(5,'HR');

**OUTPUT:**

**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.

**PL/SQL Code:**

CREATE OR REPLACE PROCEDURE TransferFunds(CUST1 IN NUMBER,CUST2 IN NUMBER,AMOUNT IN NUMBER) IS

BAL NUMBER;

CNT NUMBER;

BEGIN

    SELECT COUNT(\*) INTO CNT FROM ACCOUNTS WHERE ACCOUNTID=CUST1;

    IF CNT<>1 THEN

    DBMS\_OUTPUT.PUT\_LINE(CUST1||' ACCOUNT NOT FOUND');

    RETURN;

    END IF;

    SELECT COUNT(\*) INTO CNT FROM ACCOUNTS WHERE ACCOUNTID=CUST2;

    IF CNT<>1 THEN

    DBMS\_OUTPUT.PUT\_LINE(CUST2||' ACCOUNT NOT FOUND');

    RETURN;

    END IF;

    SELECT BALANCE INTO BAL FROM ACCOUNTS WHERE ACCOUNTID=CUST1;

    IF(BAL<AMOUNT) THEN

    DBMS\_OUTPUT.PUT\_LINE('INSUFFICIENT BALANCE TO INITIATE TRANSFER');

    RETURN;

    END IF;

    UPDATE ACCOUNTS

    SET BALANCE=BALANCE-AMOUNT

    WHERE ACCOUNTID=CUST1;

    UPDATE ACCOUNTS

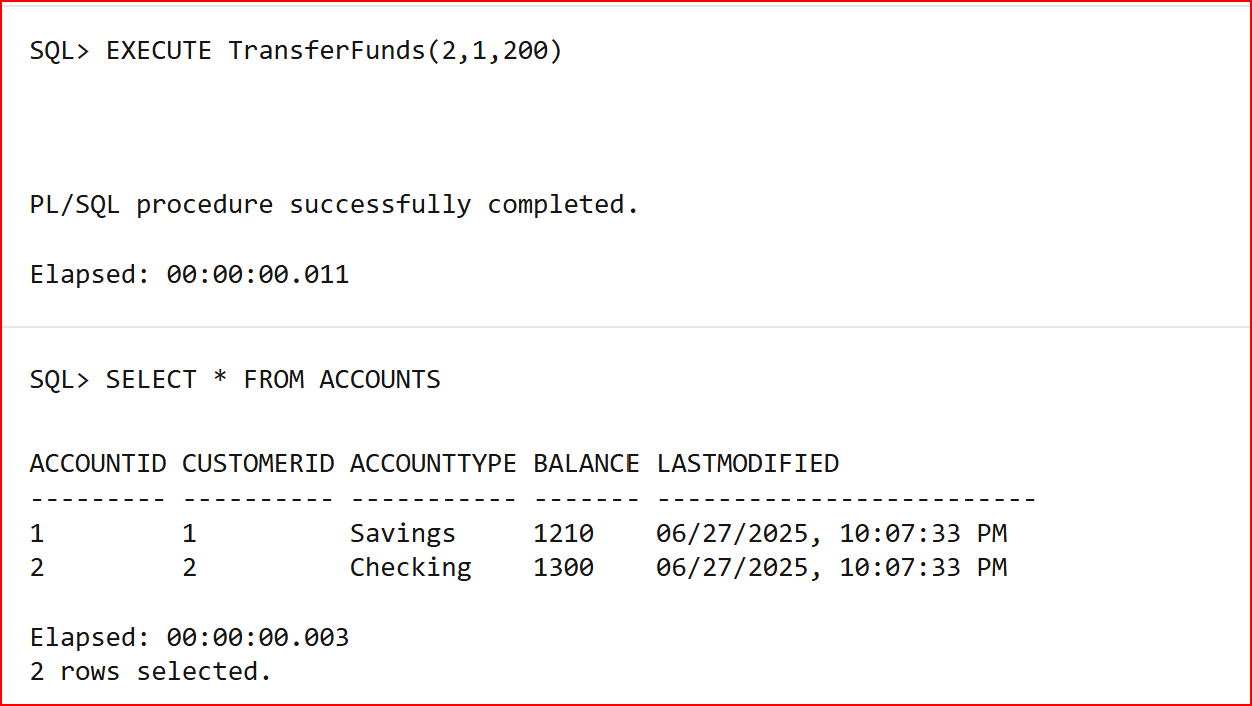
    SET BALANCE=BALANCE+AMOUNT

    WHERE ACCOUNTID=CUST2;

END TransferFunds;

**EXECUTING PROCEDURE:**

EXECUTE TransferFunds(2,1,200);

**OUTPUT:**