**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:**

**Scenario1:**

SET SERVEROUTPUT ON;

DECLARE

    v\_age NUMBER;

BEGIN

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

        -- Calculate age

        v\_age := TRUNC(MONTHS\_BETWEEN(SYSDATE, cust.DOB) / 12);

        IF v\_age > 60 THEN

            UPDATE Loans

            SET InterestRate = InterestRate - 1

            WHERE CustomerID = cust.CustomerID;

            DBMS\_OUTPUT.PUT\_LINE(' Interest rate updated for customer ' || cust.Name ||

                                 ' (ID: ' || cust.CustomerID || ', Age: ' || v\_age || ')');

        ELSE

            DBMS\_OUTPUT.PUT\_LINE(' No update: Customer ' || cust.Name ||

                                 ' is ' || v\_age || ' years old.');

        END IF;

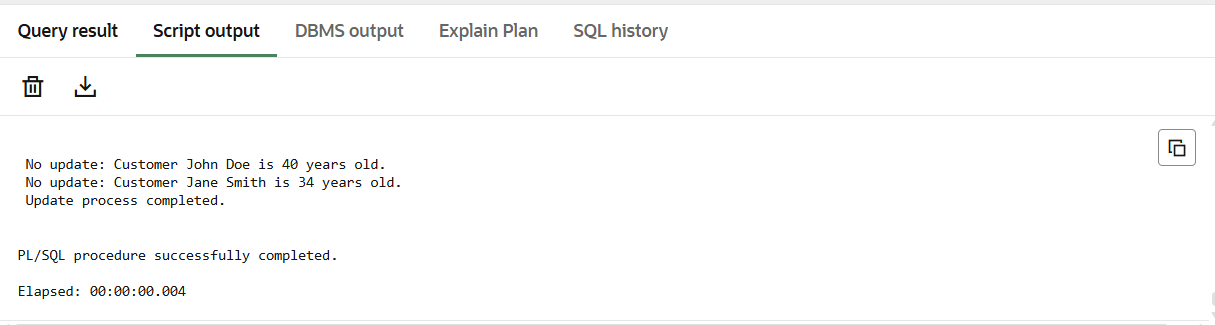
    END LOOP;

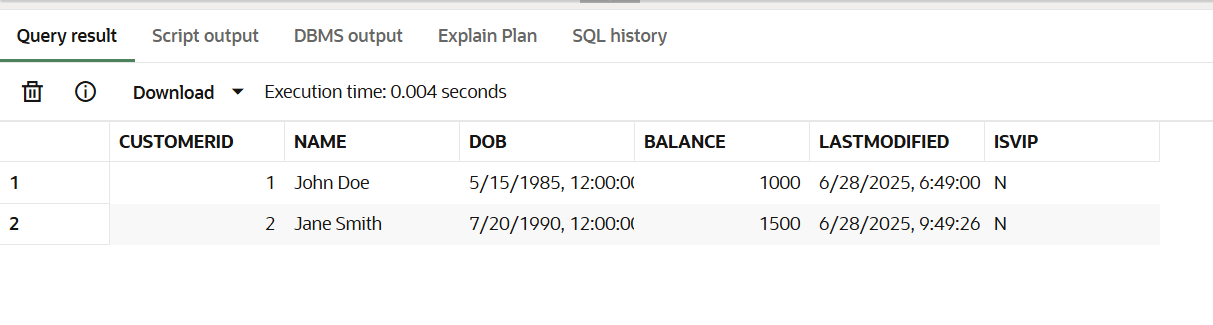
    COMMIT;

    DBMS\_OUTPUT.PUT\_LINE(' Update process completed.');

END;

/

****

****

**Scenario 2:**

ALTER TABLE Customers ADD IsVIP CHAR(1) DEFAULT 'N';

BEGIN

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

        IF cust.Balance > 10000 THEN

            UPDATE Customers

            SET IsVIP = 'Y'

            WHERE CustomerID = cust.CustomerID;

            DBMS\_OUTPUT.PUT\_LINE('VIP: ' || cust.Name || ' (ID: ' || cust.CustomerID ||

                                 ') - Balance: $' || cust.Balance);

        ELSE

            UPDATE Customers

            SET IsVIP = 'N'

            WHERE CustomerID = cust.CustomerID;

            DBMS\_OUTPUT.PUT\_LINE('Non-VIP: ' || cust.Name || ' (ID: ' || cust.CustomerID ||

                                 ') - Balance: $' || cust.Balance);

        END IF;

    END LOOP;

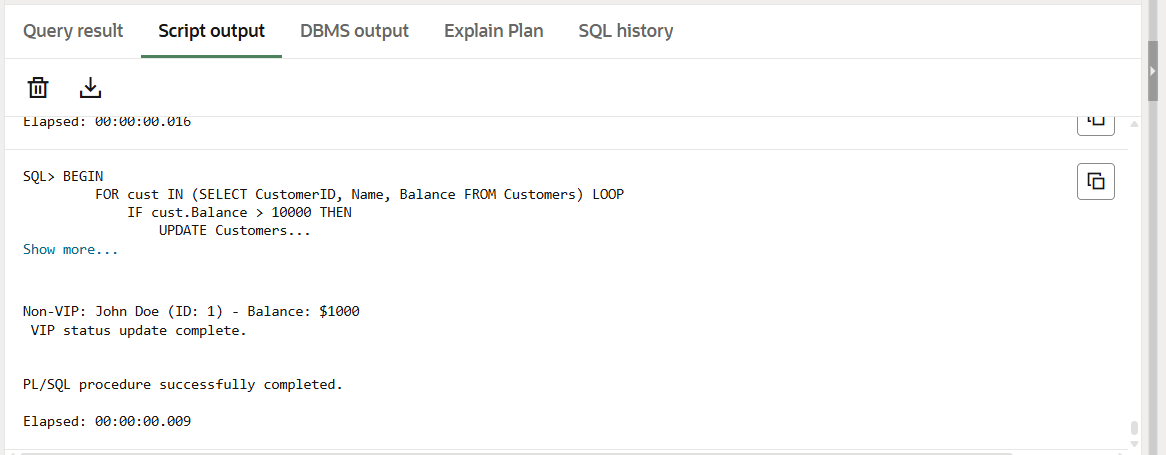
    COMMIT;

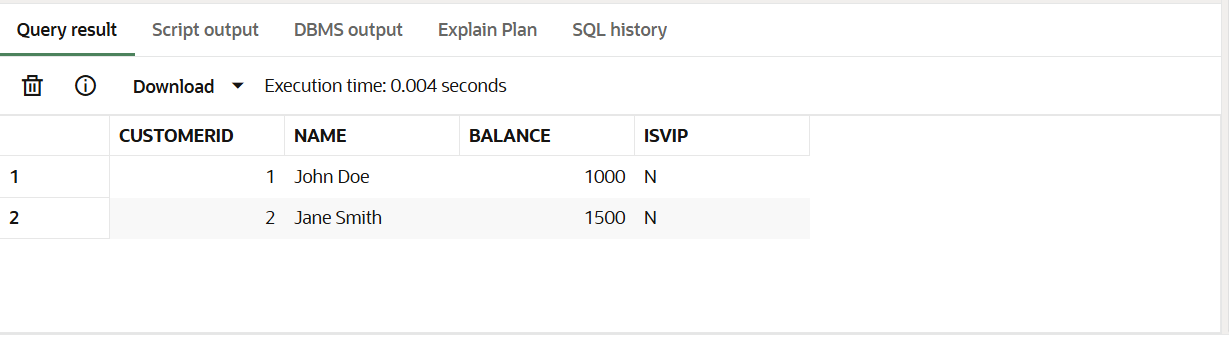
    DBMS\_OUTPUT.PUT\_LINE(' VIP status update complete.');

END;

/

SELECT CustomerID, Name, Balance, IsVIP FROM Customers;

****

****

**Scenario3:**

SET SERVEROUTPUT ON;

DECLARE

    v\_name VARCHAR2(100);

BEGIN

    FOR loan\_rec IN (

        SELECT LoanID, CustomerID, EndDate

        FROM Loans

        WHERE EndDate BETWEEN SYSDATE AND SYSDATE + 30

    ) LOOP

        -- Get customer name

        SELECT Name INTO v\_name

        FROM Customers

        WHERE CustomerID = loan\_rec.CustomerID;

        -- Print reminder

        DBMS\_OUTPUT.PUT\_LINE(' Reminder: Loan ' || loan\_rec.LoanID ||

                             ' for customer ' || v\_name ||

                             ' (CustomerID: ' || loan\_rec.CustomerID || ') is due on ' ||

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

    END LOOP;

    DBMS\_OUTPUT.PUT\_LINE(' Loan due reminder check complete.');

END;

/

SELECT

    l.LoanID,

    l.CustomerID,

    c.Name AS CustomerName,

    l.LoanAmount,

    l.InterestRate,

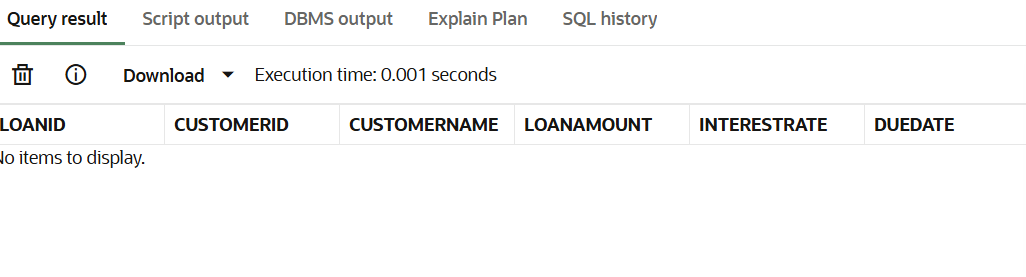
    TO\_CHAR(l.EndDate, 'DD-MON-YYYY') AS DueDate

FROM Loans l

JOIN Customers c ON l.CustomerID = c.CustomerID

WHERE l.EndDate BETWEEN SYSDATE AND SYSDATE + 30;

****

****

**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.

**Scenario 1:**

SET SERVEROUTPUT ON;

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest AS

BEGIN

    UPDATE Accounts

    SET Balance = Balance \* 1.01,

        LastModified = SYSDATE

    WHERE AccountType = 'Savings';

    DBMS\_OUTPUT.PUT\_LINE('Interest applied to savings accounts.');

END;

/

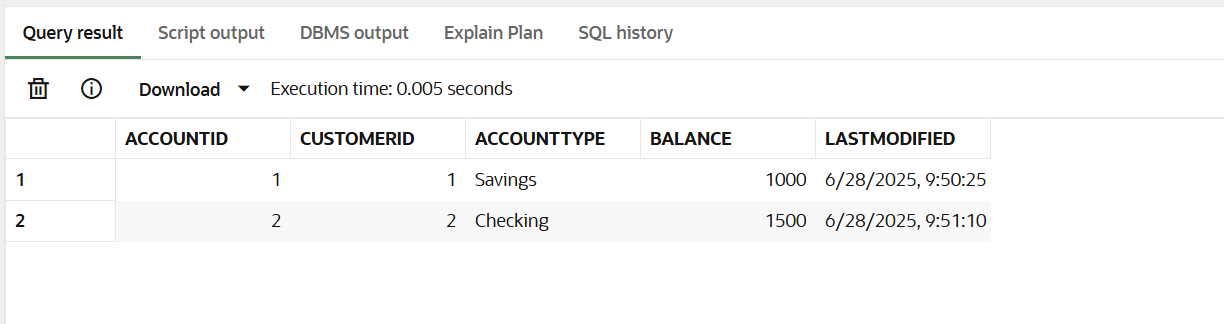
BEGIN

    ProcessMonthlyInterest;

END;

/

SELECT \* FROM Accounts;

****

**Scenario2:**

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus(

    dept IN VARCHAR2,

    bonus\_pct IN NUMBER

) AS

BEGIN

    UPDATE Employees

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

    WHERE Department = dept;

    DBMS\_OUTPUT.PUT\_LINE('Bonus applied to department: ' || dept);

END;

/

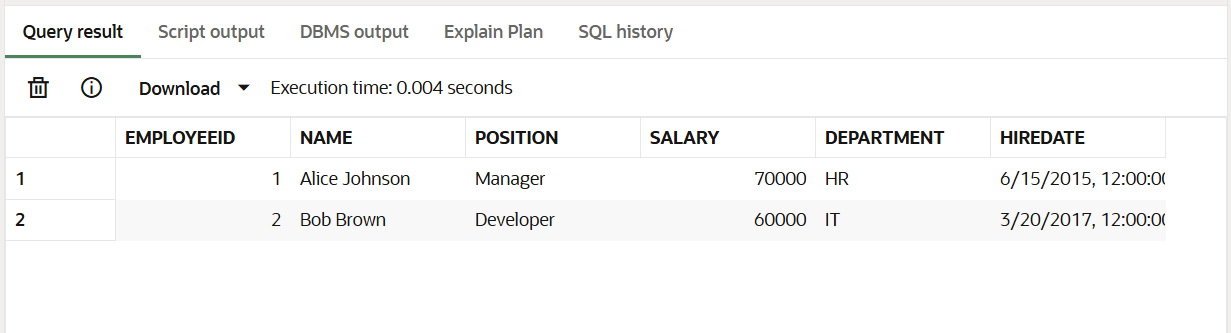
BEGIN

    UpdateEmployeeBonus('IT', 10);

END;

/

SELECT \* FROM Employees;

****

**Scenario3:**

CREATE OR REPLACE PROCEDURE TransferFunds(

fromAccID IN NUMBER,

toAccID IN NUMBER,

amount IN NUMBER

) AS

insufficient\_funds EXCEPTION;

from\_balance NUMBER;

BEGIN

SELECT Balance INTO from\_balance

FROM Accounts

WHERE AccountID = fromAccID

FOR UPDATE;

IF from\_balance < amount THEN

RAISE insufficient\_funds;

END IF;

UPDATE Accounts

SET Balance = Balance - amount,

LastModified = SYSDATE

WHERE AccountID = fromAccID;

UPDATE Accounts

SET Balance = Balance + amount,

LastModified = SYSDATE

WHERE AccountID = toAccID;

DBMS\_OUTPUT.PUT\_LINE('Transferred ' || amount || ' from account ' || fromAccID || ' to ' || toAccID);

EXCEPTION

WHEN insufficient\_funds THEN

DBMS\_OUTPUT.PUT\_LINE('Error: Insufficient funds in account ' || fromAccID);

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Unexpected error: ' || SQLERRM);

END;

/

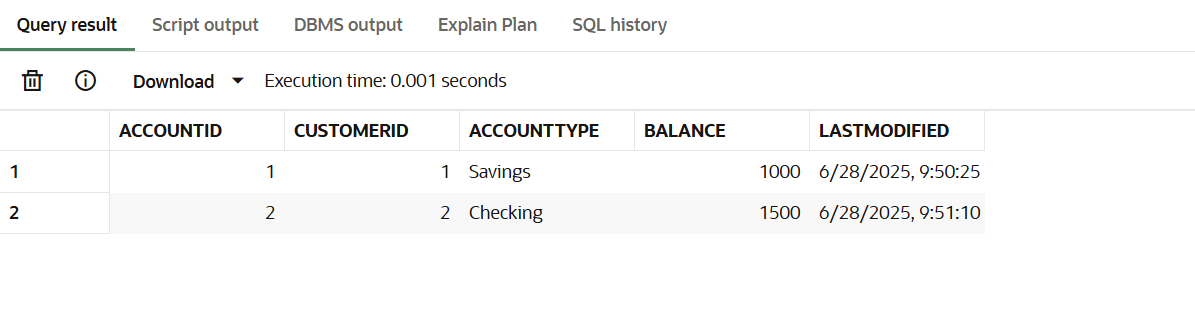
BEGIN

TransferFunds(2, 1, 200);

END;

/

SELECT \* FROM Accounts;

****

*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'));*