**DIGITAL NURTURE 4.0 JavaFSE**

**WEEK 2**

**PL/SQL**

**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.

CREATE TABLE Customers (

CustomerID NUMBER PRIMARY KEY,

Name VARCHAR2(100),

Age NUMBER,

Balance NUMBER(10,2),

IsVIP CHAR(1) DEFAULT 'N' -- 'Y' or 'N'

);

INSERT INTO Customers VALUES (1, 'John Doe', 65, 12000.00, 'N');

INSERT INTO Customers VALUES (2, 'Alice Smith', 45, 8000.00, 'N');

INSERT INTO Customers VALUES (3, 'Bob Johnson', 70, 15000.00, 'N');

INSERT INTO Customers VALUES (4, 'Sara Lee', 30, 3000.00, 'N');

BEGIN

FOR rec IN (

SELECT c.CustomerID, l.LoanID

FROM Customers c

JOIN Loans l ON c.CustomerID = l.CustomerID

WHERE c.Age > 60

) LOOP

UPDATE Loans

SET InterestRate = InterestRate - 1

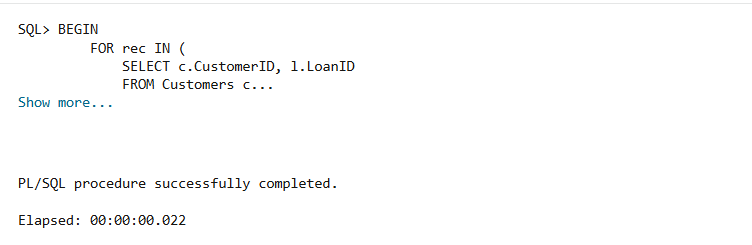
WHERE LoanID = rec.LoanID;

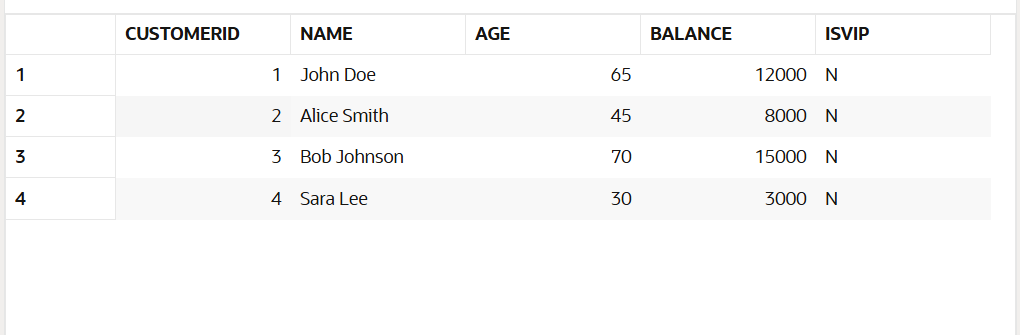
END LOOP;

COMMIT;

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.

CREATE TABLE Loans (

LoanID NUMBER PRIMARY KEY,

CustomerID NUMBER,

InterestRate NUMBER(5,2), -- in percent

DueDate DATE,

FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)

);

INSERT INTO Loans VALUES (101, 1, 8.5, SYSDATE + 10);

INSERT INTO Loans VALUES (102, 2, 9.0, SYSDATE + 40);

INSERT INTO Loans VALUES (103, 3, 7.5, SYSDATE + 5);

INSERT INTO Loans VALUES (104, 4, 10.0, SYSDATE + 15);

BEGIN

FOR rec IN (SELECT CustomerID FROM Customers WHERE Balance > 10000) LOOP

UPDATE Customers

SET IsVIP = 'Y'

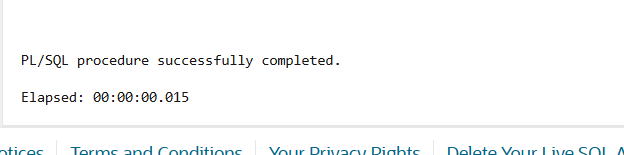
WHERE CustomerID = rec.CustomerID;

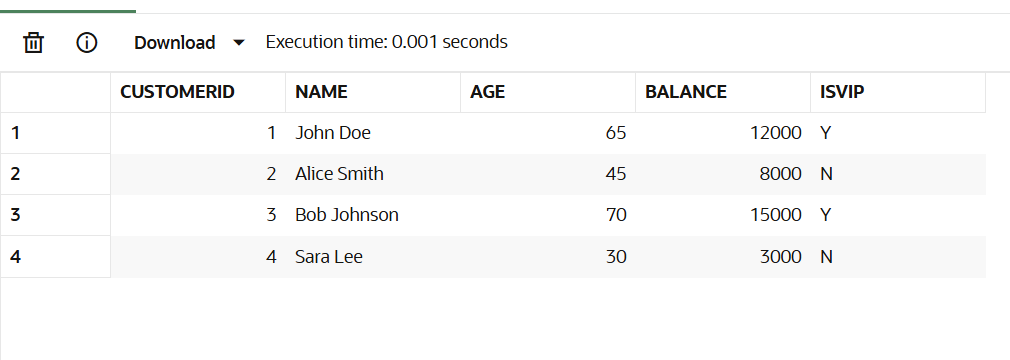
END LOOP;

COMMIT;

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.

BEGIN

FOR rec IN (

SELECT c.Name, l.LoanID, l.DueDate

FROM Loans l

JOIN Customers c ON l.CustomerID = c.CustomerID

WHERE l.DueDate BETWEEN SYSDATE AND SYSDATE + 30

) LOOP

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

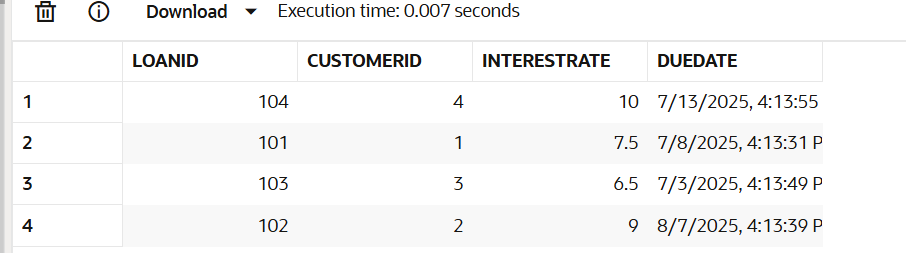
' for customer ' || rec.Name ||

' is due on ' || TO\_CHAR(rec.DueDate, 'DD-MON-YYYY'));

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.

**Table Creation:**

CREATE TABLE SavingsAccounts (

AccountID NUMBER PRIMARY KEY,

CustomerID NUMBER,

Balance NUMBER(10,2)

);

INSERT INTO SavingsAccounts (AccountID, CustomerID, Balance)

VALUES (101, 1, 5000.00);

INSERT INTO SavingsAccounts (AccountID, CustomerID, Balance)

VALUES (102, 2, 10000.00);

INSERT INTO SavingsAccounts (AccountID, CustomerID, Balance)

VALUES (103, 3, 2000.00);

**PL/SQL Query:**

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest IS

BEGIN

    FOR acc IN (SELECT AccountID, Balance FROM SavingsAccounts) LOOP

        UPDATE SavingsAccounts

        SET Balance = Balance + (Balance \* 0.01)

        WHERE AccountID = acc.AccountID;

    END LOOP;

COMMIT;

END;

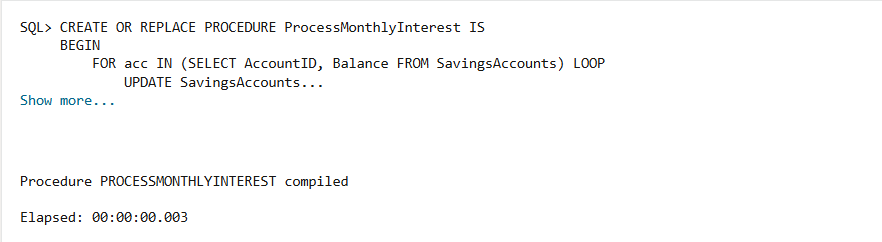
BEGIN

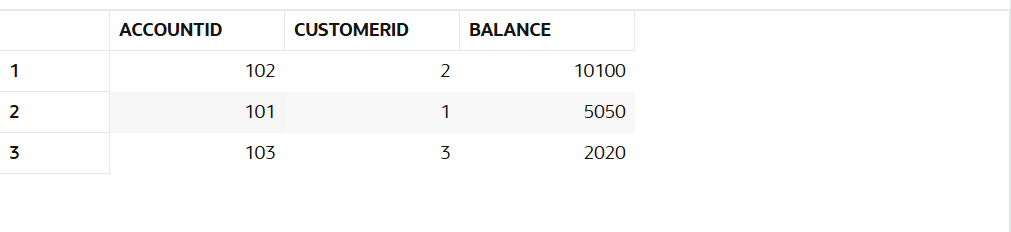
    ProcessMonthlyInterest;

END;

SELECT \* FROM SavingsAccounts;

**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.

CREATE TABLE Employees {

    EmpID        NUMBER PRIMARY KEY,

    Name         VARCHAR2(100),

    Department   VARCHAR2(50),

    Salary       NUMBER(10,2)

);

INSERT INTO Employees (EmpID, Name, Department, Salary)

VALUES (201, 'Alice', 'HR', 40000);

INSERT INTO Employees (EmpID, Name, Department, Salary)

VALUES (202, 'Bob', 'Finance', 45000);

INSERT INTO Employees (EmpID, Name, Department, Salary)

VALUES (203, 'Charlie', 'HR', 50000);

**Query:**

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus (

    p\_department IN VARCHAR2,

    p\_bonus\_percent IN NUMBER

) IS

BEGIN

    UPDATE Employees

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

    WHERE Department = p\_department;

    COMMIT;

END;

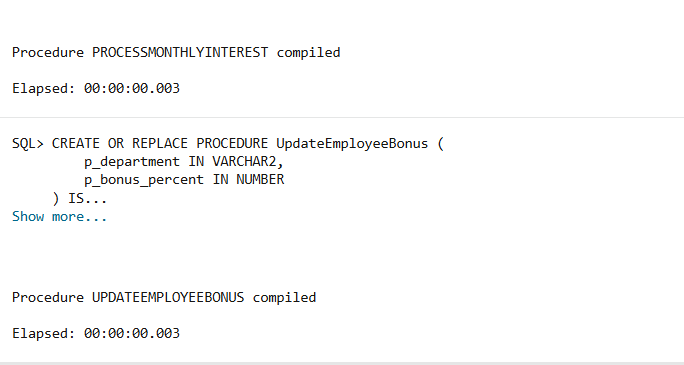
BEGIN

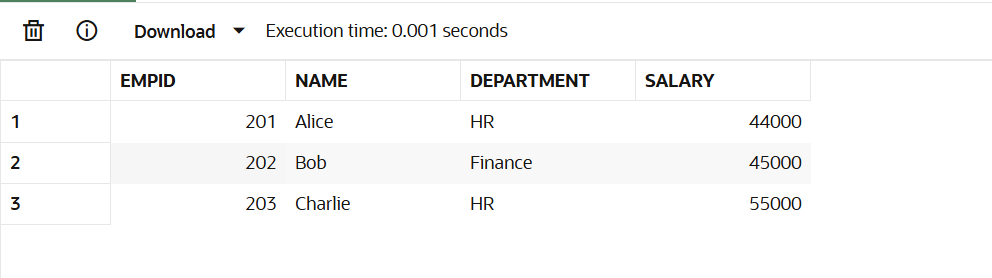
    UpdateEmployeeBonus('HR', 10);

END;

SELECT \* FROM Employees;

**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.

CREATE TABLE BankAccounts (

    AccountID    NUMBER PRIMARY KEY,

    CustomerID   NUMBER,

    Balance      NUMBER(10,2)

);

INSERT INTO BankAccounts (AccountID, CustomerID, Balance)

VALUES (301, 1, 8000.00);

INSERT INTO BankAccounts (AccountID, CustomerID, Balance)

VALUES (302, 2, 3000.00);

**QUERY:**

CREATE OR REPLACE PROCEDURE TransferFunds (

    p\_source\_account IN NUMBER,

    p\_target\_account IN NUMBER,

    p\_amount IN NUMBER

) IS

    v\_balance NUMBER;

BEGIN

    SELECT Balance INTO v\_balance

    FROM BankAccounts

    WHERE AccountID = p\_source\_account;

    IF v\_balance < p\_amount THEN

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

    END IF;

    UPDATE BankAccounts

    SET Balance = Balance - p\_amount

    WHERE AccountID = p\_source\_account;

    UPDATE BankAccounts

    SET Balance = Balance + p\_amount

    WHERE AccountID = p\_target\_account;

    COMMIT;

END;

BEGIN

    TransferFunds(301, 302, 2000);

END;

SELECT \* FROM BankAccounts;

**OUTPUT:**

