**SOLUTIONS WEEK 2**

**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.

CODE

*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*

*);*

**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 Customers (CustomerID, Name, DOB, Balance, LastModified)*

*VALUES (3, 'John Das', TO\_DATE('1940-07-20', '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 Accounts (AccountID, CustomerID, AccountType, Balance, LastModified)*

*VALUES (3, 3, 'Savings', 1000, 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 Transactions (TransactionID, AccountID, TransactionDate, Amount, TransactionType)*

*VALUES (3, 3, SYSDATE, 200, 'Deposit');*

*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, 3, 6000, 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'));*

Scenario 1:

SET SERVEROUTPUT ON

DECLARE

CURSOR cus IS

SELECT CUSTOMERID,DOB FROM CUSTOMERS;

BEGIN

    FOR ind IN cus

    LOOP

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

        UPDATE LOANS

        SET INTERESTRATE=INTERESTRATE\*0.99

        WHERE CUSTOMERID=ind.CUSTOMERID;

        DBMS\_OUTPUT.PUT\_LINE('LOANS TABLE  IS UPDATED');

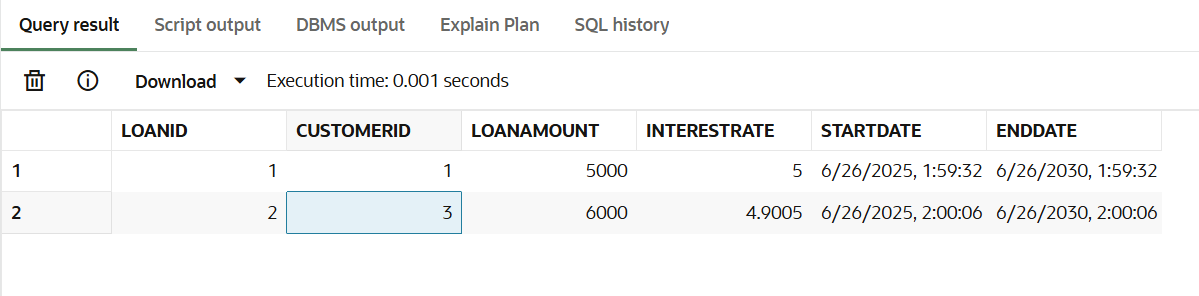
        END IF;

        END LOOP;

        END;

SELECT \* FROM LOANS;

OUTPUT



Scenario 2:

A screenshot of a computer

AI-generated content may be incorrect.

Scenario 2:

ALTER TABLE Customers ADD IsVIP CHAR(1);

SELECT \* FROM CUSTOMERS;

SET SERVEROUTPUT ON

DECLARE

CURSOR vip IS

SELECT CUSTOMERID,BALANCE FROM CUSTOMERS;

BEGIN

    FOR ind IN vip

    LOOP

        IF ind.BALANCE>10000 THEN

        UPDATE CUSTOMERS

        SET ISVIP='T'

        WHERE CUSTOMERID=ind.CUSTOMERID;

        ELSE

        UPDATE CUSTOMERS

        SET ISVIP='F'

        WHERE CUSTOMERID=ind.CUSTOMERID;

        DBMS\_OUTPUT.PUT\_LINE('CUSTOMER TABLE  IS UPDATED');

        END IF;

        END LOOP;

        END;

SELECT \* FROM CUSTOMERS;

Output

BEFORE CURSOR FOR LOOP

A screenshot of a computer

AI-generated content may be incorrect.

AFTER CURSOR FOR LOOP

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

Scenario 3:

INSERT INTO Loans (LoanID, CustomerID, LoanAmount, InterestRate, StartDate, EndDate)

VALUES (3, 2, 5000, 5, SYSDATE, SYSDATE+30)

SET SERVEROUTPUT ON

DECLARE

CURSOR rem IS

SELECT CUSTOMERID,STARTDATE,ENDDATE FROM LOANS;

BEGIN

    FOR ind IN rem

    LOOP

        IF ind.ENDDATE=ind.STARTDATE+30 THEN

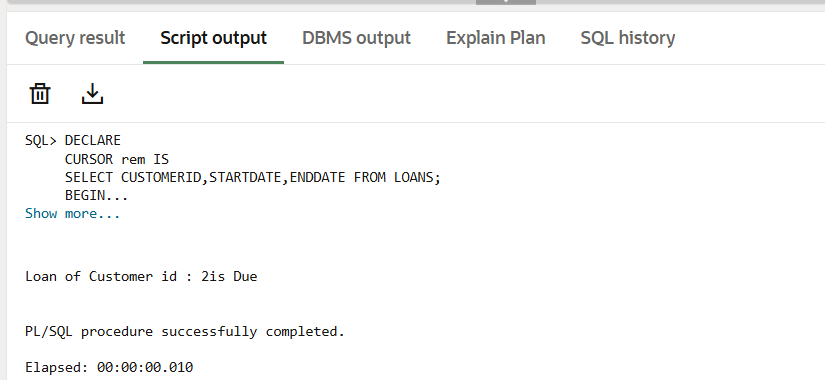
        DBMS\_OUTPUT.PUT\_LINE('Loan of Customer id : '||ind.CUSTOMERID||'is Due');

        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.

**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.

CODE

SCENARIO 1:

CREATE OR REPLACE  PROCEDURE ProcessMonthlyInterest IS

BEGIN

    UPDATE ACCOUNTS

    SET BALANCE=BALANCE+(BALANCE\*0.01)

    WHERE ACCOUNTTYPE='Savings';

    DBMS\_OUTPUT.PUT\_LINE('BALANCE UPDATED');

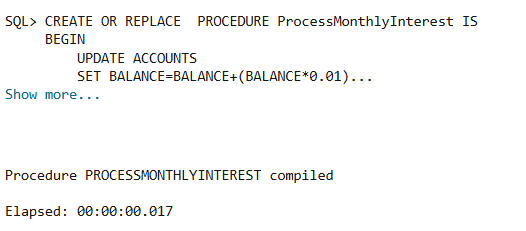
END ProcessMonthlyInterest ;

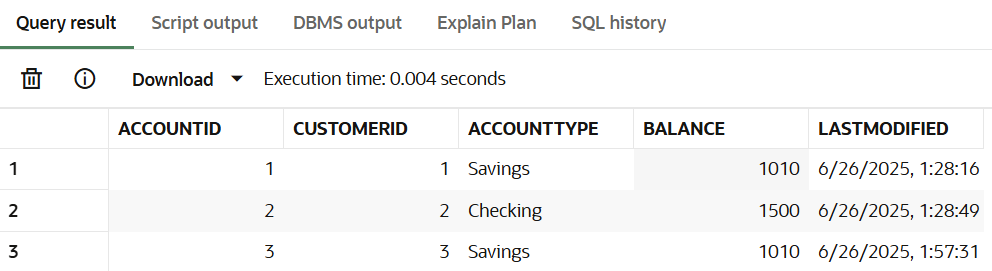
/

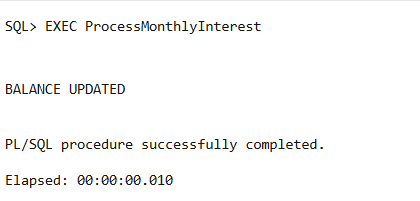
EXEC ProcessMonthlyInterest;

select \* from ACCOUNTS;

OUTPUT





****

SCENARIO 2:

CREATE OR REPLACE  PROCEDURE UpdateEmployeeBonus(dep IN VARCHAR2 ,bonus DECIMAL) IS

BEGIN

    UPDATE EMPLOYEES

    SET SALARY=SALARY+(SALARY\*(bonus)/100)

    WHERE DEPARTMENT=dep;

    DBMS\_OUTPUT.PUT\_LINE('SALARY UPDATED');

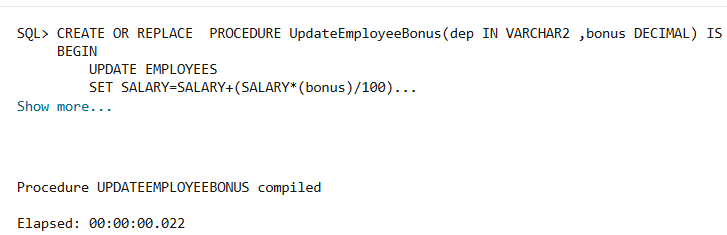
END UpdateEmployeeBonus ;

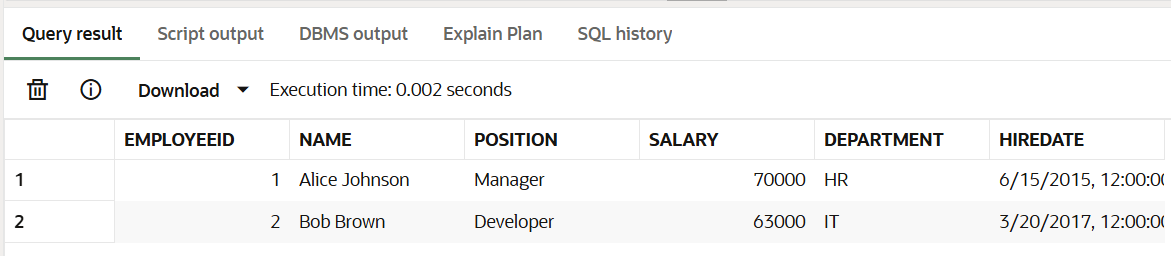
/

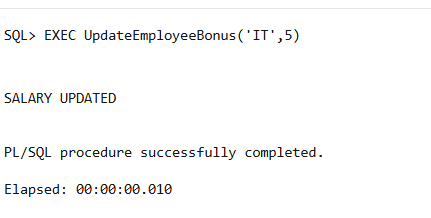
EXEC UpdateEmployeeBonus('IT',5);

select \* from EMPLOYEES;

OUTPUT







Scenario 3:

CREATE OR REPLACE  PROCEDURE TransferFunds(fromAcc NUMBER,toAcc NUMBER,amt NUMBER) IS

frombal NUMBER;

BEGIN

    SELECT BALANCE INTO frombal FROM ACCOUNTS WHERE ACCOUNTID=fromAcc;

    IF frombal>=amt THEN

    UPDATE ACCOUNTS

    SET BALANCE=BALANCE+amt

    WHERE ACCOUNTID=toAcc;

    UPDATE ACCOUNTS

    SET BALANCE=BALANCE-amt

    WHERE ACCOUNTID=fromAcc;

    DBMS\_OUTPUT.PUT\_LINE('TRANSACTION DONE');

    ELSE

    DBMS\_OUTPUT.PUT\_LINE('INSUFFICIENT AMOUNT');

    END IF;

END TransferFunds ;

/

EXEC TransferFunds(2,3,500);

select \* from ACCOUNTS;

OUTPUT

BEFORE TRANSACTION

A screenshot of a computer

AI-generated content may be incorrect.

AFTER TRANSACTION

