**Week 2- PL/SQL HANDS ON**

**Name: Adi Jain**

**Superset ID: 6390400**

**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.

**Code**

BEGIN

  for res in (

    select c.name, c.customerID, l.loanId, l.interestRate

    from loans l

    join customers c on l.customerID = c.customerID

    where trunc(Months\_between(sysdate, c.dob)/12)>60

  ) loop

    update loans

    set interestRate = res.interestRate-1

    where loanid = res.loanid;

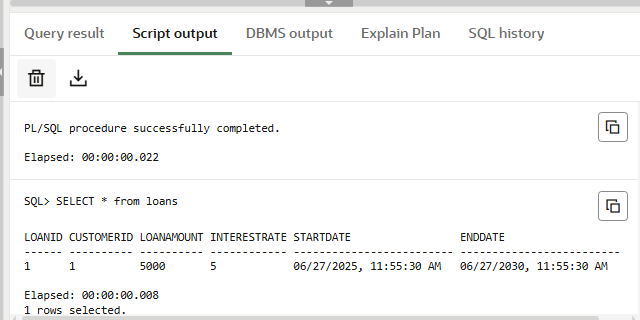
  end loop;

END;

/

SELECT \* from loans;

**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.

**Code**

ALTER TABLE Customers

ADD IsVIP CHAR(1) DEFAULT 'N' CHECK (IsVIP IN ('Y', 'N'));

BEGIN

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

        IF cust.Balance > 10000 THEN

            UPDATE Customers

            SET IsVIP = 'Y'

            WHERE CustomerID = cust.CustomerID;

        ELSE

            UPDATE Customers

            SET IsVIP = 'N'

            WHERE CustomerID = cust.CustomerID;

        END IF;

    END LOOP;

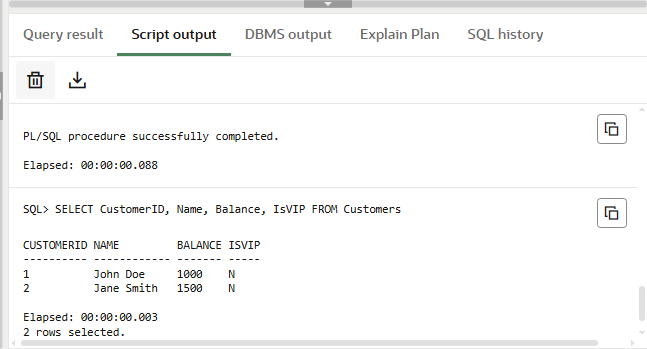
    COMMIT;

END;

/

SELECT CustomerID, Name, Balance, IsVIP FROM Customers;

**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.

**Code**

DECLARE

    cnt number:=0;

    BEGIN

    FOR loan\_rec IN (

        SELECT l.LoanID, l.CustomerID, l.EndDate, c.Name

        FROM Loans l

        JOIN Customers c ON l.CustomerID = c.CustomerID

        WHERE l.EndDate BETWEEN SYSDATE AND SYSDATE + 30

    ) LOOP

        DBMS\_OUTPUT.PUT\_LINE(

            'Reminder: Loan ID ' || loan\_rec.LoanID ||

            ' for customer "' || loan\_rec.Name ||

            '" (Customer ID: ' || loan\_rec.CustomerID ||

            ') is due on ' || TO\_CHAR(loan\_rec.EndDate, 'DD-MON-YYYY')

        );

    END LOOP;

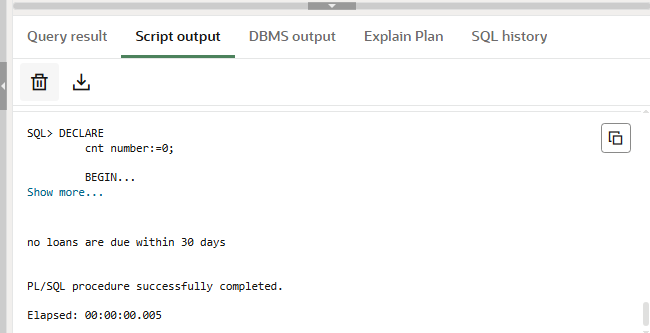
    IF cnt=0 then dbms\_output.PUT\_LINE ('no loans are due within 30 days');

    END IF;

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.

**Code**

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest IS

BEGIN

    UPDATE Accounts

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

        LastModified = SYSDATE

    WHERE AccountType = 'Savings';

    COMMIT;

END;

/

BEGIN

    ProcessMonthlyInterest;

END;

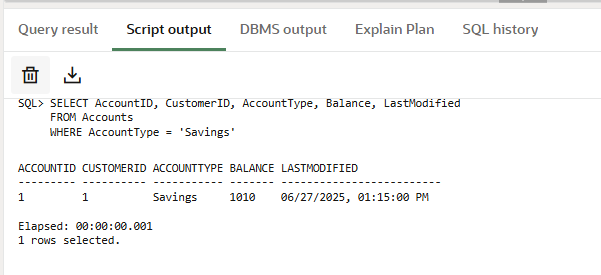
/

SELECT AccountID, CustomerID, AccountType, Balance, LastModified

FROM Accounts

WHERE AccountType = 'Savings';

**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.

**Code**

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus (

    p\_department IN VARCHAR2,

    p\_bonus\_percent IN NUMBER

) AS

BEGIN

    UPDATE Employees

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

    WHERE Department = p\_department;

    COMMIT;

END;

/

BEGIN

    UpdateEmployeeBonus('IT', 10);

END;

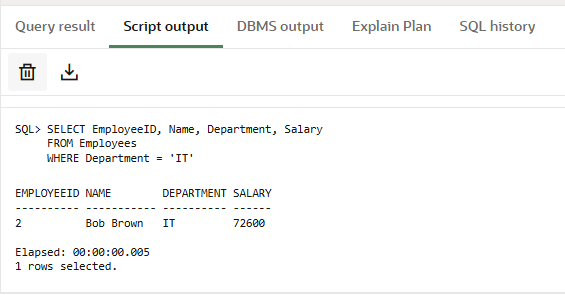
/

SELECT EmployeeID, Name, Department, Salary

FROM Employees

WHERE Department = 'IT';

**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.

**Code**

set SERVEROUTPUT on;

CREATE or REPLACE procedure fundtransfer(

    from\_account IN number,

    to\_account In number,

    amount in number

)is

v\_balance number;

begin

   select balance into v\_balance

   from accounts

   where accountid= from\_account

   for update;

   if v\_balance<amount then

   raise\_application\_error(-20001, 'Insufficient Balance');

   end if;

   update accounts

   set balance= balance- amount, lastmodified= sysdate

   where accountid= from\_account;

   update accounts

   set balance= balance+ amount, lastmodified= sysdate

   where accountid= to\_account;

end;

/

SELECT \* from ACCOUNTS;

**Output**

