**PL/SQL Hands-on**

**Name : Udit Bhargava**

**Superset Id : 6386801**

**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 :**

SELECT \* FROM loans;

begin

for c in ( select customerid, trunc(MONTHS\_BETWEEN(sysdate, DOB)/12) as age from customers) loop

if c.age>60 then

update loans

set interestrate = interestrate-1

where customerid = c.customerid;

end if;

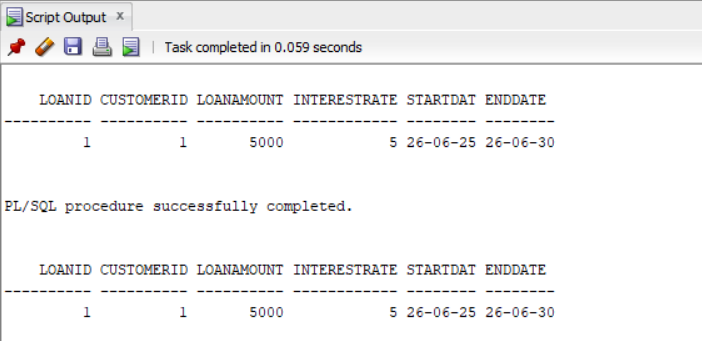
end loop;

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 VARCHAR2(5);

SELECT \* FROM customers;

BEGIN

FOR c in ( SELECT customerid, balance FROM customers) LOOP

IF c.balance>10000 THEN

UPDATE customers

SET IsVIP = 'TRUE'

WHERE customerid = c.customerid;

ELSE

UPDATE customers

SET IsVIP = 'FALSE'

WHERE customerid = c.customerid;

END IF;

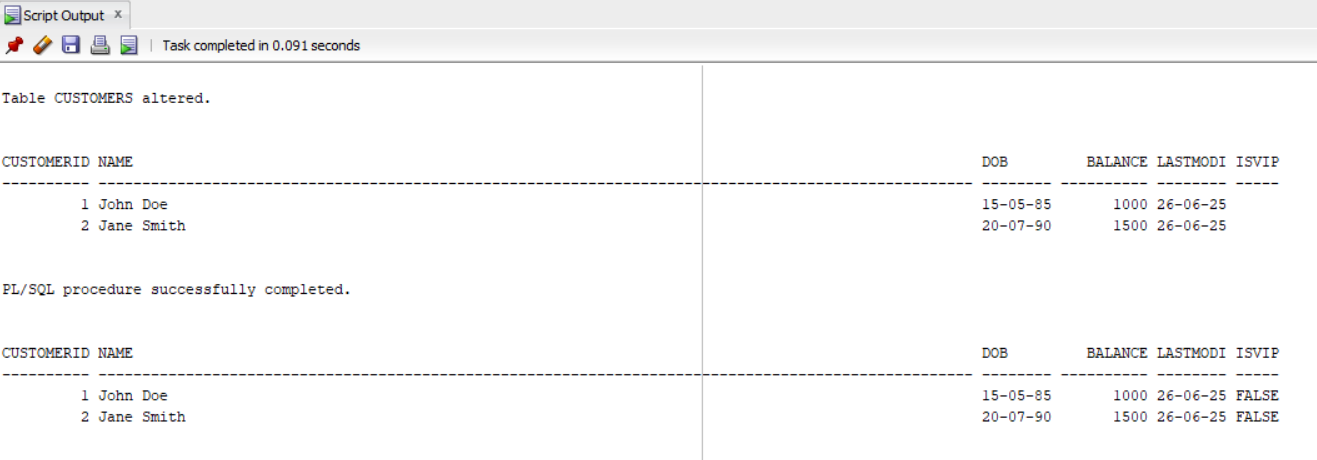
END LOOP;

END;

/

SELECT \* 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 :**

SET SERVEROUTPUT ON;

SELECT \* FROM loans;

BEGIN

FOR l 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: Dear ' || l.Name ||

', your loan (LoanID: ' || l.LoanID ||

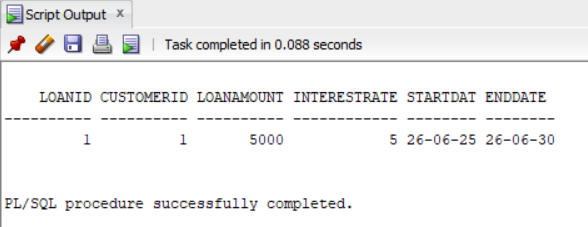
') is due on ' || TO\_CHAR(l.EndDate, 'DD-MON-YYYY')

);

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.

**CODE :**

**Procedure ProcessMonthlyInterest :**

create or replace PROCEDURE ProcessMonthlyInterest IS

BEGIN

UPDATE Accounts

SET Balance = Balance \* 1.01

WHERE AccountType = 'Savings';

DBMS\_OUTPUT.PUT\_LINE('Monthly interest applied to all savings accounts.');

END;

**Execution Code :**

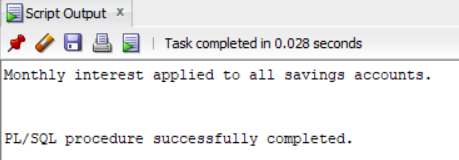
SET SERVEROUTPUT ON;

BEGIN

ProcessMonthlyInterest;

END;

**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 :**

**Procedure UpdateEmployeeBonus :**

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus(

dept IN VARCHAR2,

bonus IN NUMBER

) IS

BEGIN

UPDATE Employees

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

WHERE Department = dept;

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

END;

**Execution Code :**

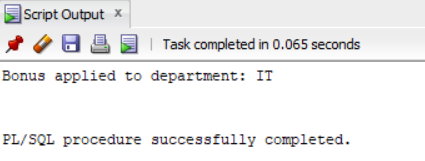
SET SERVEROUTPUT ON;

BEGIN

UpdateEmployeeBonus('IT', 10);

END;

**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 :**

**Procedure TransferFunds :**

CREATE OR REPLACE PROCEDURE TransferFunds(

FromAccount IN NUMBER,

ToAccount IN NUMBER,

Amt IN NUMBER

) IS

v\_Balance NUMBER;

BEGIN

SELECT Balance INTO v\_Balance

FROM Accounts

WHERE AccountID = FromAccount;

IF v\_Balance >= Amt THEN

UPDATE Accounts

SET Balance = Balance - Amt

WHERE AccountID = FromAccount;

UPDATE Accounts

SET Balance = Balance + Amt

WHERE AccountID = ToAccount;

DBMS\_OUTPUT.PUT\_LINE('Transfer of ' || Amt || ' from Account ' ||

FromAccount || ' to Account ' || ToAccount ||

' completed successfully.');

ELSE

DBMS\_OUTPUT.PUT\_LINE('Insufficient funds in Account ' || FromAccount);

END IF;

END;

**Execution Code :**

SET SERVEROUTPUT ON;

SELECT AccountID, CustomerID, Balance FROM Accounts WHERE AccountID IN (1, 2);

BEGIN

TransferFunds(1, 2, 200);

END;

/

SELECT AccountID, CustomerID, Balance FROM Accounts WHERE AccountID IN (1, 2);

**Output :**

