**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.

**Query:**

BEGIN

FOR CUS IN (SELECT Customers.CustomerID FROM Customers JOIN Loans ON Customers.CustomerID=Loans.CustomerID

WHERE TRUNC(MONTHS\_BETWEEN(SYSDATE, DOB)/12)>60)LOOP

UPDATE Loans

SET InterestRate=InterestRate-1

WHERE Loans.CustomerID=CUS.CustomerID;

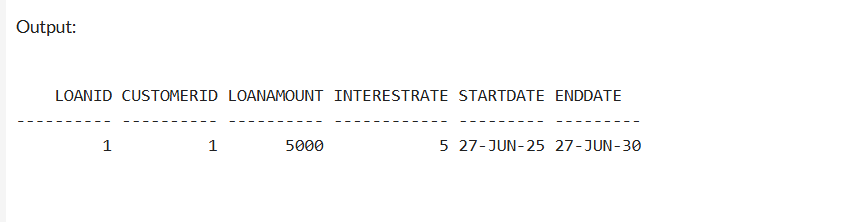
END LOOP;

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.

**Query:**

ALTER TABLE Customers

ADD IsVip NUMBER(1) DEFAULT 0;

BEGIN

FOR CUS IN(SELECT CustomerID FROM Customers W

WHERE Balance>10000)

LOOP

UPDATE Customers

SET IsVip=1

WHERE CustomerID=CUS.CustomerID;

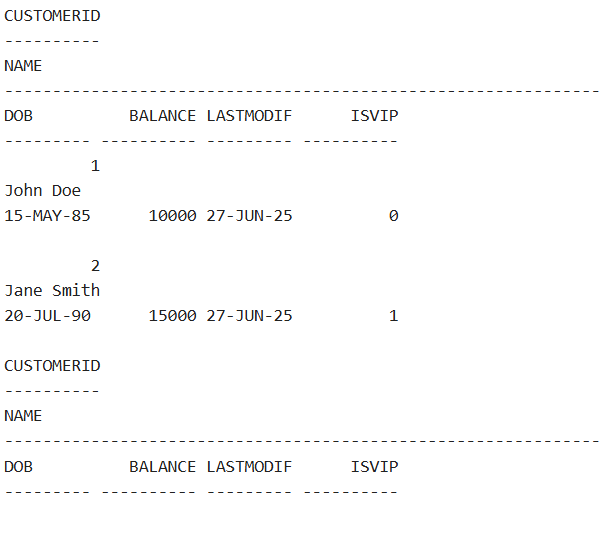
END LOOP;

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.

BEGIN

FOR CUS IN(SELECT Customers.Name,Loans.EndDate FROM

Loans JOIN Customers ON Loans.CustomerID=Customers.CustomerID WHERE (Loans.EndDate-SYSDATE<30))

LOOP

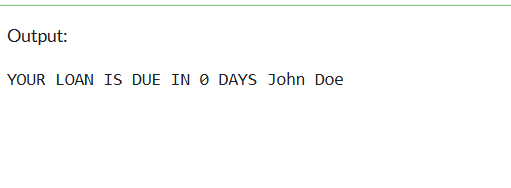
DBMS\_OUTPUT.PUT\_LINE('YOUR LOAN IS DUE ' || (CUS.EndDate-SYSDATE) || ' DAYS' || CUS.Name);

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.

**Query:**

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest

AS

BEGIN

FOR CUS IN(SELECT CustomerID FROM Accounts

WHERE AccountType='Savings')

LOOP

UPDATE Accounts

SET Balance=Balance+Balance\*0.01

WHERE CustomerID=CUS.CustomerID;

END LOOP;

END;

/

BEGIN

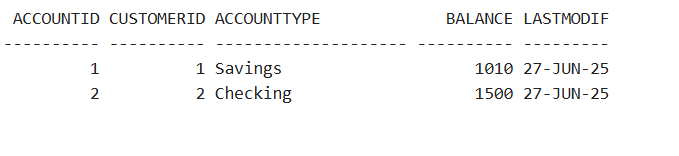
ProcessMonthlyInterest;

END;

/

SELECT\* FROM Accounts;

**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.

**Query:**

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus(

PERCENTAGE IN NUMBER

)

AS

BEGIN

FOR EMP IN(SELECT EmployeeID FROM Employees)

LOOP

UPDATE Employees

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

WHERE EmployeeID=EMP.EmployeeID;

END LOOP;

END;

/

BEGIN

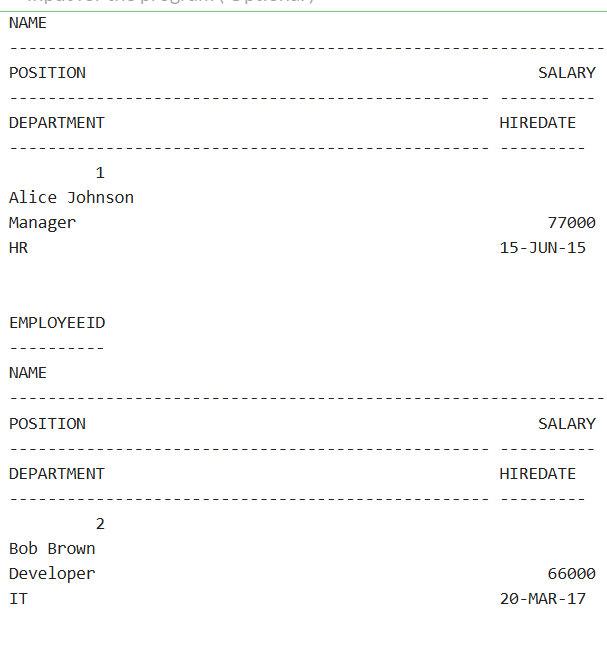
UpdateEmployeeBonus(10);

END;

/

SELECT\* FROM Employees;

**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.

**QUERY:**

CREATE OR REPLACE PROCEDURE TransferFunds (

ACC\_TO IN NUMBER,

ACC\_FROM IN NUMBER,

AMT IN NUMBER

)

AS

SRC\_BALANCE NUMBER;

BEGIN

SELECT Balance INTO SRC\_BALANCE

FROM Accounts

WHERE AccountID = ACC\_FROM;

IF(AMT>SRC\_BALANCE) THEN

RAISE\_APPLICATION\_ERROR(-20101,'INSUFFICIENT BALANCE');

END IF;

UPDATE Accounts

SET Balance=Balance-AMT

WHERE AccountID=ACC\_FROM;

UPDATE Accounts

SET Balance=Balance+AMT

WHERE AccountID=ACC\_TO;

COMMIT;

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE(SQLERRM);

END;

/

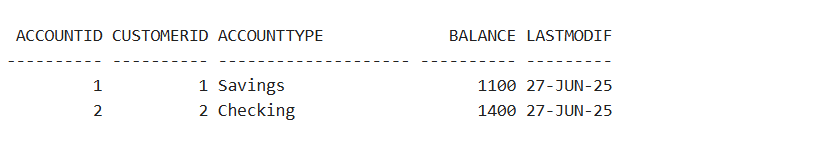
BEGIN

TransferFunds(1,2,10000);

END;

/

**OUTPUT:**

****