**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 cust IN (SELECT CustomerID, DOB FROM Customers) LOOP

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

UPDATE Loans

SET InterestRate = InterestRate - 1

WHERE CustomerID = cust.CustomerID;

END IF;

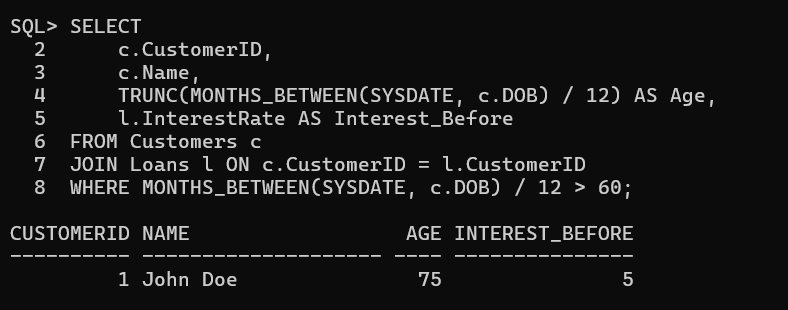
END LOOP;

COMMIT;

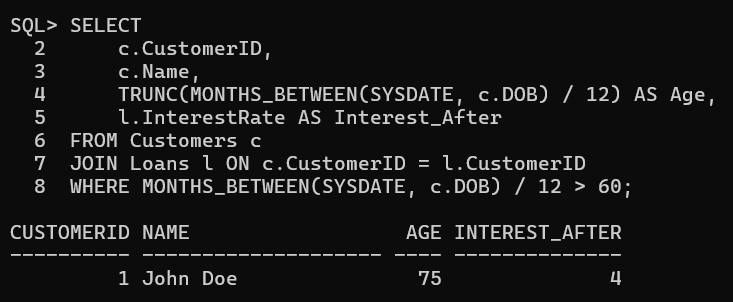
END;

/

**Output before applying changes:**



**Output after applying changes:**



**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:**

BEGIN

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

IF cust.Balance > 10000 THEN

UPDATE Customers

SET IsVIP = 'TRUE'

WHERE CustomerID = cust.CustomerID;

END IF;

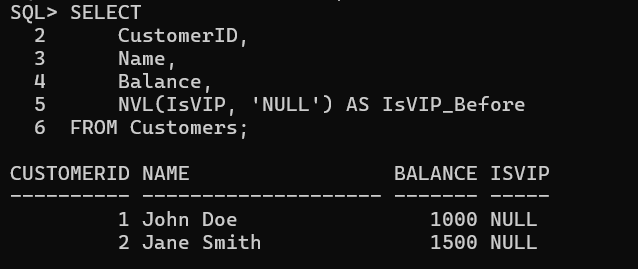
END LOOP;

COMMIT;

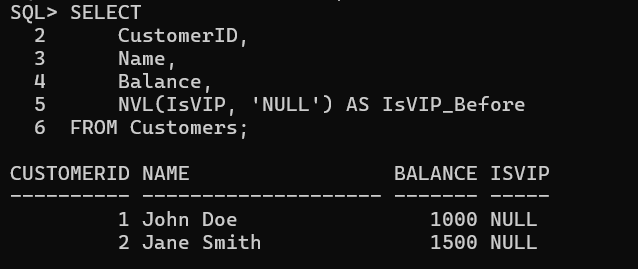
END;

/

**Output before applying changes:**



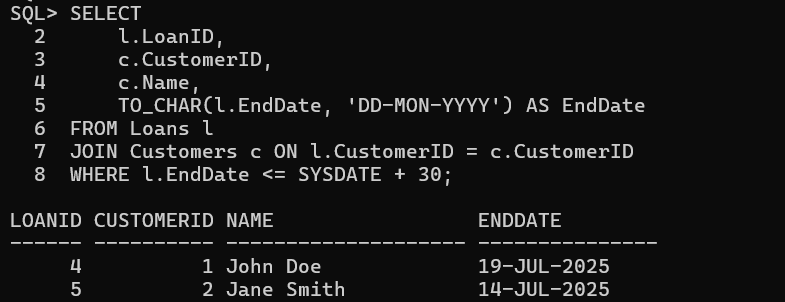
**After changes output:**



**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.

**Before:**



**Query:**

BEGIN

FOR loan IN (

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

FROM Loans l

JOIN Customers c ON l.CustomerID = c.CustomerID

WHERE l.EndDate <= SYSDATE + 30

) LOOP

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

' for customer ' || loan.Name ||

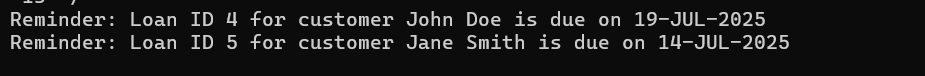
' is due on ' || TO\_CHAR(loan.EndDate, 'DD-MON-YYYY'));

END LOOP;

END;

/

**After Output:**

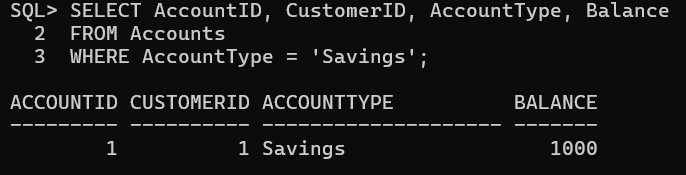


**Exercise 3: Stored Procedures**

**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.

**Before:**



**Query:**

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest IS

BEGIN

FOR acc IN (

SELECT AccountID FROM Accounts WHERE AccountType = 'Savings'

) LOOP

UPDATE Accounts

SET Balance = Balance + (Balance \* 0.01)

WHERE AccountID = acc.AccountID;

END LOOP;

COMMIT;

END;

/

**Execute Procedure:**

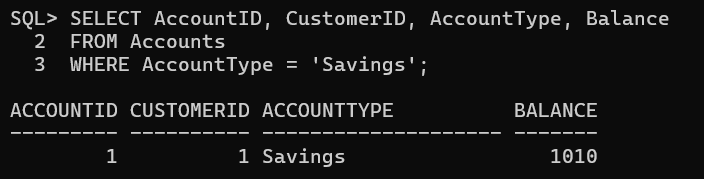
BEGIN

ProcessMonthlyInterest;

END;

/

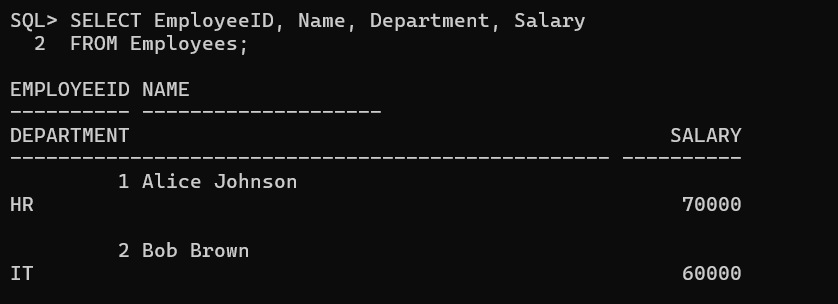
**After:**



**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.

**Before:**



**Query:**

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus (

dept\_name IN VARCHAR2,

bonus\_pct IN NUMBER

) IS

BEGIN

UPDATE Employees

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

WHERE Department = dept\_name;

COMMIT;

END;

/

**Execute Procedure:**

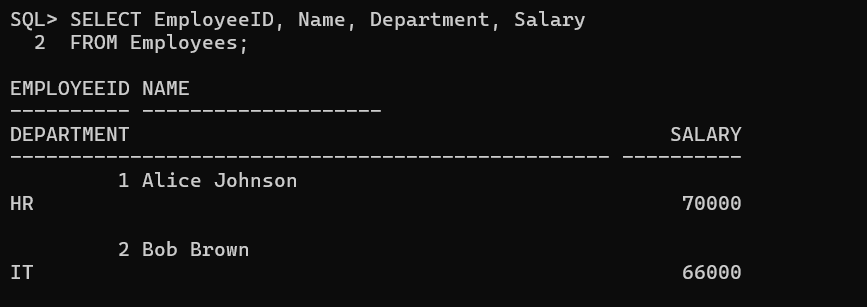
BEGIN

UpdateEmployeeBonus('IT', 10); -- Example: 10% bonus to IT dept

END;

/

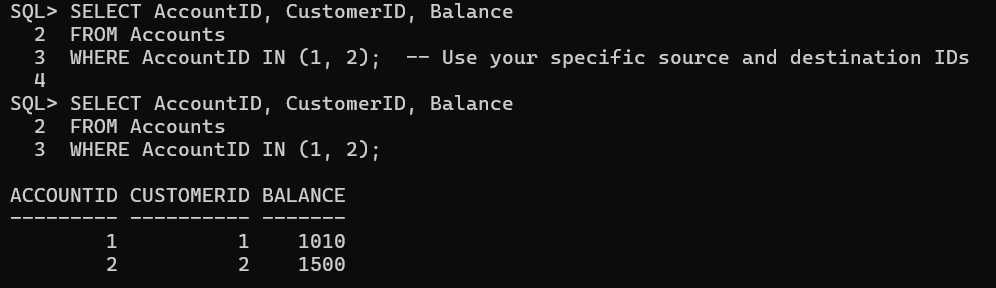
**After Output:**



**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.

**Before:**



**Query:**

Creating Procedure:  
CREATE OR REPLACE PROCEDURE TransferFunds (

from\_acc IN NUMBER,

to\_acc IN NUMBER,

amount IN NUMBER

) IS

insufficient\_balance EXCEPTION;

curr\_balance NUMBER;

BEGIN

-- Get the balance of source account

SELECT Balance INTO curr\_balance FROM Accounts WHERE AccountID = from\_acc;

IF curr\_balance < amount THEN

RAISE insufficient\_balance;

END IF;

-- Deduct from source

UPDATE Accounts

SET Balance = Balance - amount

WHERE AccountID = from\_acc;

-- Add to destination

UPDATE Accounts

SET Balance = Balance + amount

WHERE AccountID = to\_acc;

COMMIT;

EXCEPTION

WHEN insufficient\_balance THEN

DBMS\_OUTPUT.PUT\_LINE('Transfer failed: Insufficient balance.');

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Transfer failed: ' || SQLERRM);

END;

/

Execute Procedure:

SET SERVEROUTPUT ON;

BEGIN

TransferFunds(1, 2, 200); -- You can change amount if needed

END;

/

**Output:**

