PL/SQL programming

Exercise 1: Control Structures

Scenario 1: The bank wants to apply a discount to loan interest rates for customers above 60 years old.

**Code:**

DECLARE

CURSOR senior\_customers IS

SELECT c.CustomerID, l.LoanID, l.InterestRate

FROM Customers c

JOIN Loans l ON c.CustomerID = l.CustomerID

WHERE MONTHS\_BETWEEN(SYSDATE, c.DOB) / 12 > 60;

BEGIN

FOR cust IN senior\_customers LOOP

UPDATE Loans

SET InterestRate = InterestRate - 1

WHERE LoanID = cust.LoanID;

END LOOP;

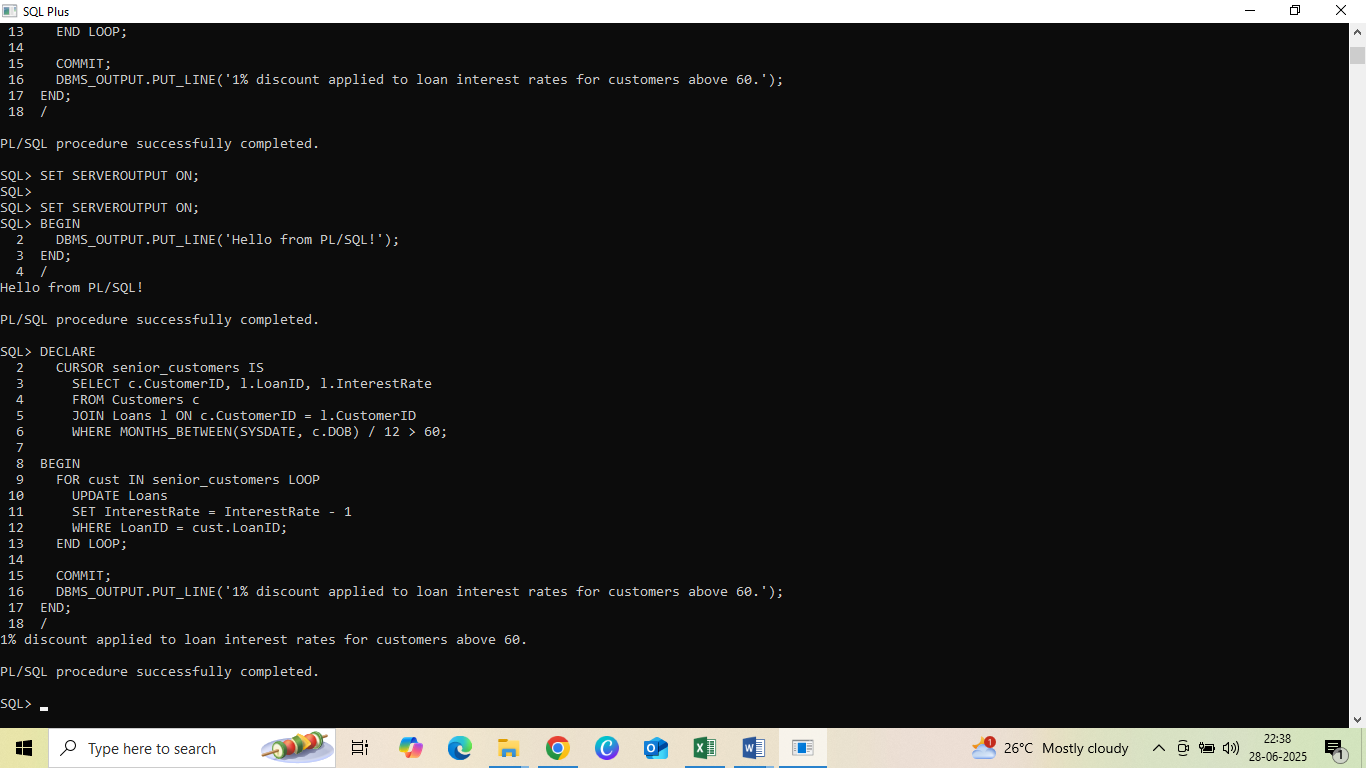
COMMIT;

DBMS\_OUTPUT.PUT\_LINE('1% discount applied to loan interest rates for customers above 60.');

END;

/

**Output:**



Scenario 2: A customer can be promoted to VIP status based on their balance.

**Code:**

ALTER TABLE Customers ADD IsVIP VARCHAR2(5);

DECLARE

CURSOR vip\_cursor IS

SELECT CustomerID FROM Customers WHERE Balance > 10000;

BEGIN

FOR cust IN vip\_cursor LOOP

UPDATE Customers

SET IsVIP = 'TRUE'

WHERE CustomerID = cust.CustomerID;

END LOOP;

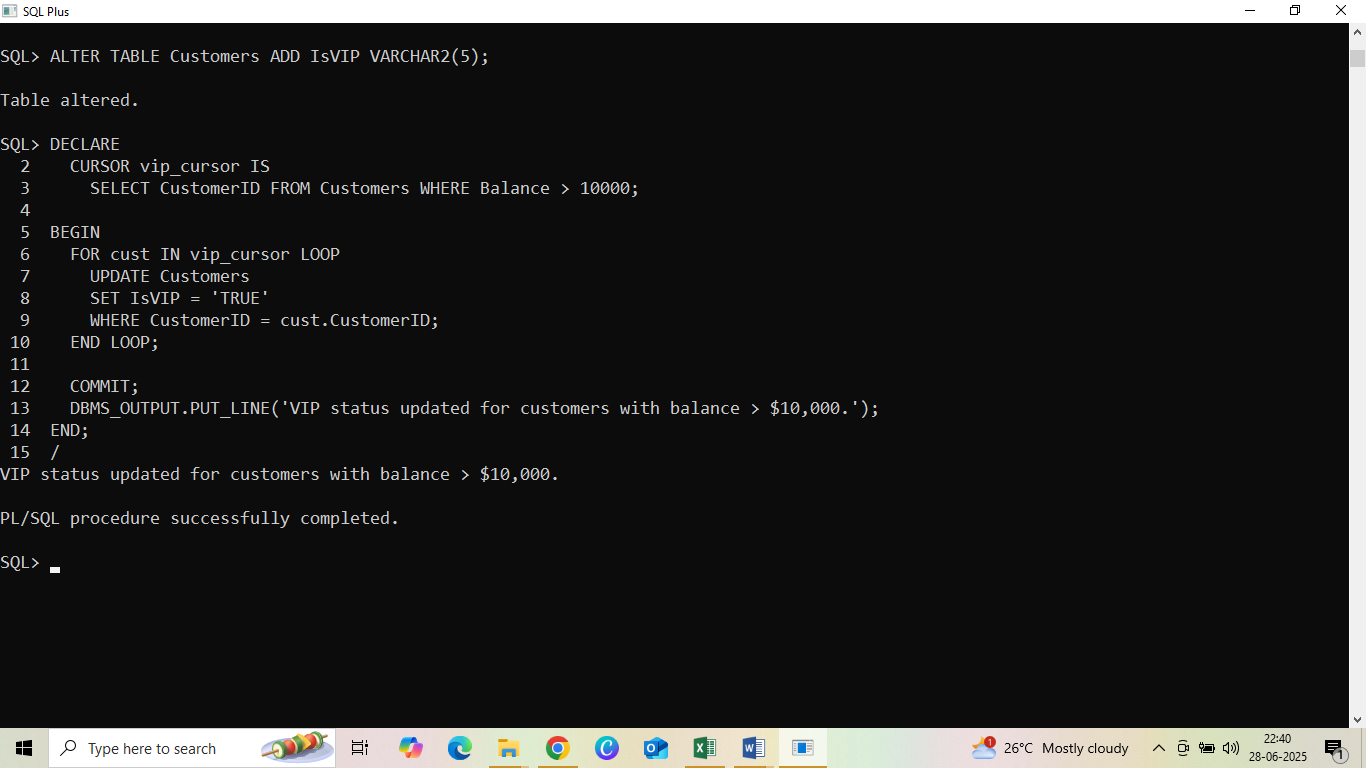
COMMIT;

DBMS\_OUTPUT.PUT\_LINE('VIP status updated for customers with balance > $10,000.');

END;

/

**Output:**



Scenario 3: The bank wants to send reminders to customers whose loans are due within the next 30 days.

**Code:**

BEGIN

FOR loan\_rec IN (

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

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\_rec.Name ||

', your loan (ID: ' || loan\_rec.LoanID ||

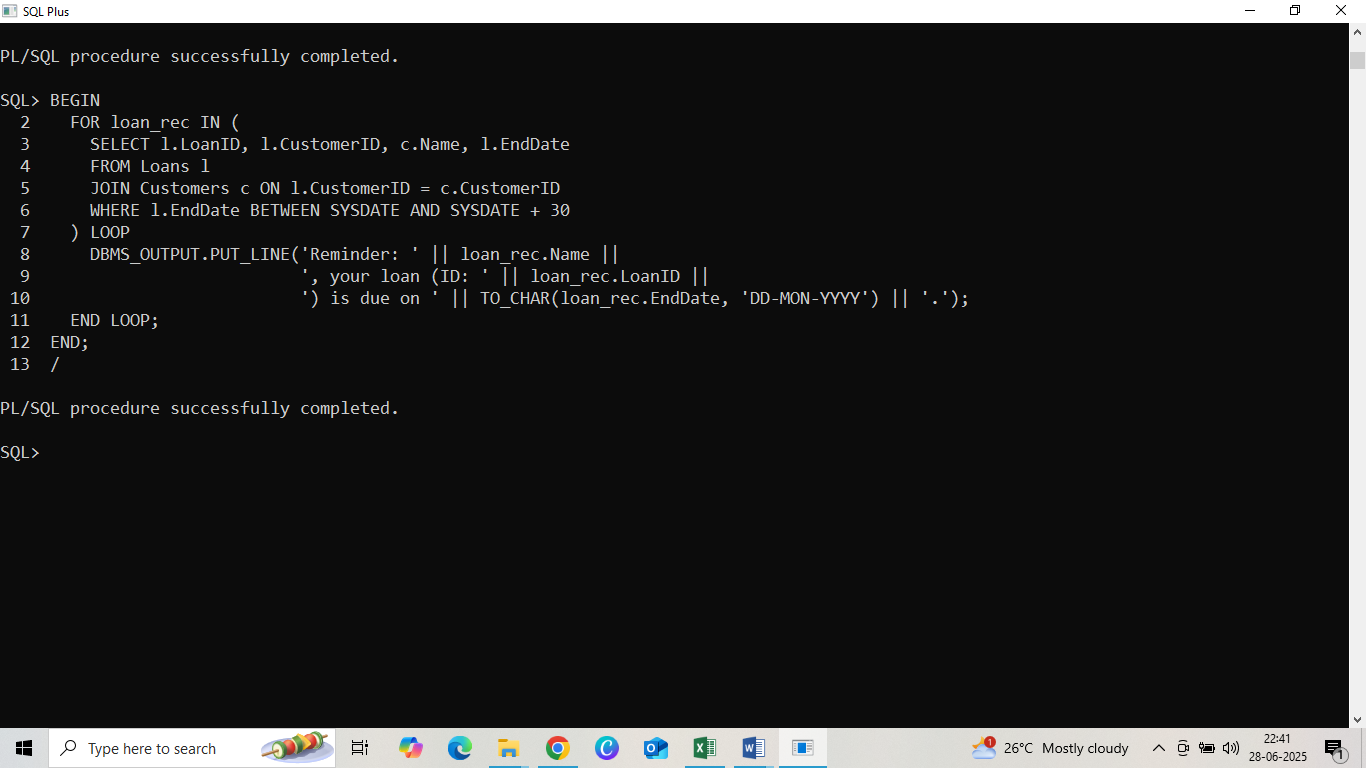
') is due on ' || TO\_CHAR(loan\_rec.EndDate, 'DD-MON-YYYY') || '.');

END LOOP;

END;

/

**Output:**



Exercise 3: Stored Procedures

Scenario 1: The bank needs to process monthly interest for all savings accounts.

**Code:**

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest IS

BEGIN

UPDATE Accounts

SET Balance = Balance + (Balance \* 0.01)

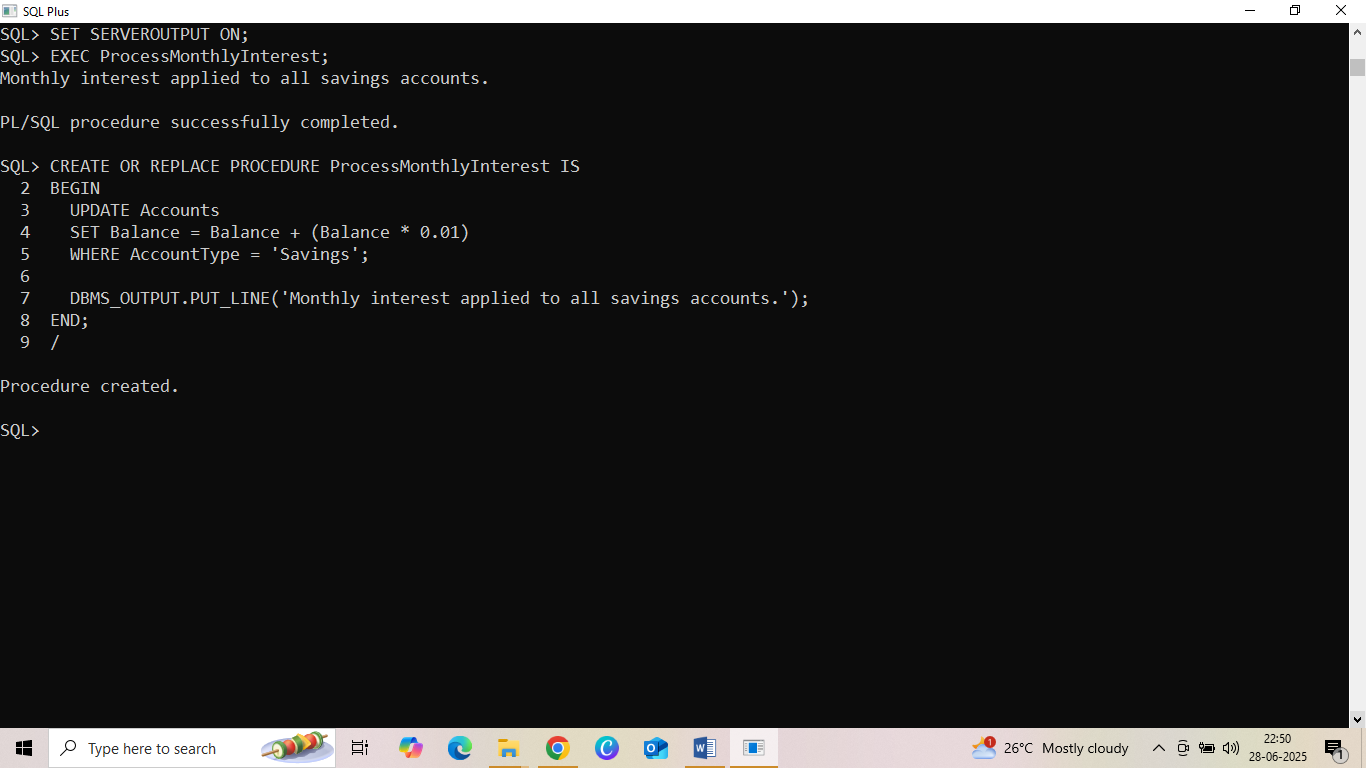
WHERE AccountType = 'Savings';

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

END;

/

**Output:**



Scenario 2: The bank wants to implement a bonus scheme for employees based on their performance.

**Code:**

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus (

p\_dept IN VARCHAR2,

p\_bonus\_percent IN NUMBER

) IS

BEGIN

UPDATE Employees

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

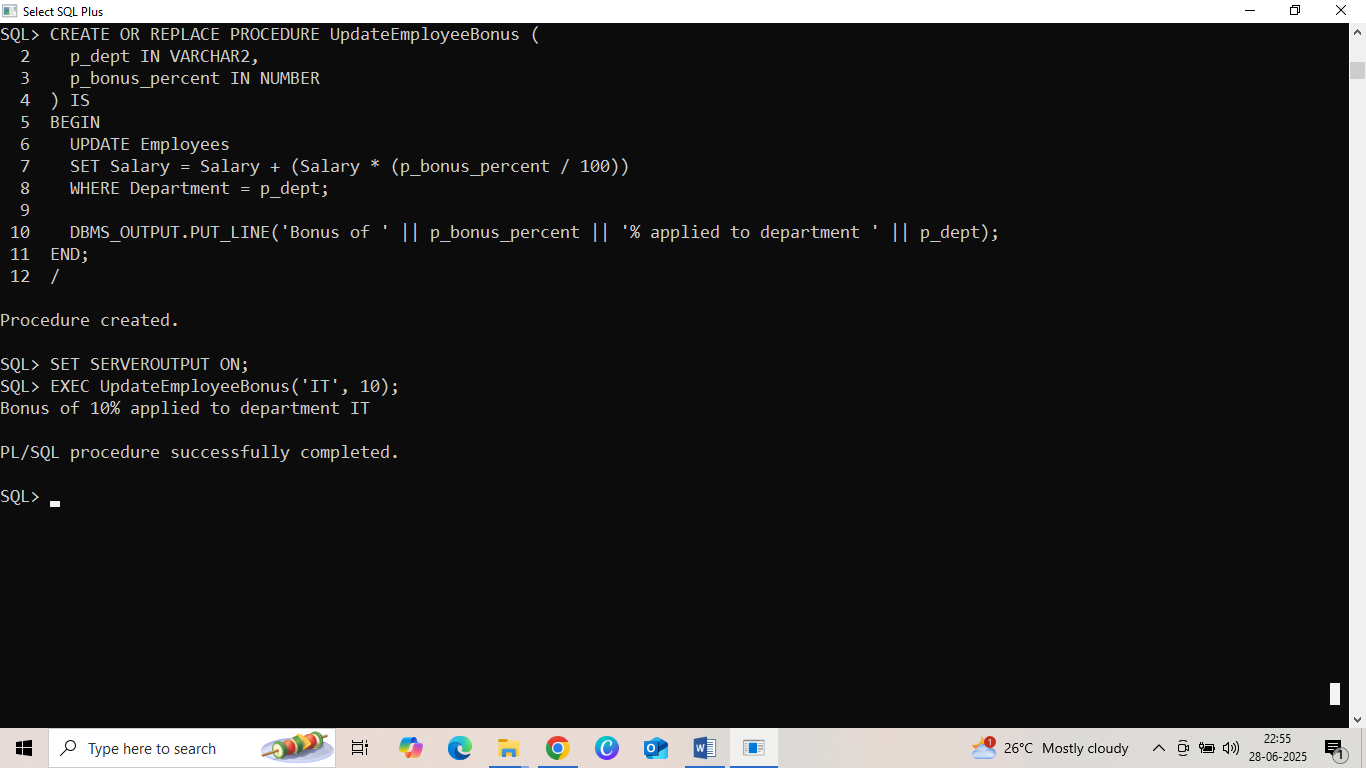
WHERE Department = p\_dept;

DBMS\_OUTPUT.PUT\_LINE('Bonus of ' || p\_bonus\_percent || '% applied to department ' || p\_dept);

END;

/

**Output:**



Scenario 3: Customers should be able to transfer funds between their accounts.

**Code:**

CREATE OR REPLACE PROCEDURE TransferFunds (

p\_from\_acc IN NUMBER,

p\_to\_acc IN NUMBER,

p\_amount IN NUMBER

) IS

v\_balance NUMBER;

BEGIN

-- Get current balance of source account

SELECT Balance INTO v\_balance

FROM Accounts

WHERE AccountID = p\_from\_acc;

-- Check for sufficient balance

IF v\_balance >= p\_amount THEN

-- Deduct from source

UPDATE Accounts

SET Balance = Balance - p\_amount

WHERE AccountID = p\_from\_acc;

-- Add to destination

UPDATE Accounts

SET Balance = Balance + p\_amount

WHERE AccountID = p\_to\_acc;

DBMS\_OUTPUT.PUT\_LINE('₹' || p\_amount || ' transferred from Account ' || p\_from\_acc || ' to Account ' || p\_to\_acc);

ELSE

DBMS\_OUTPUT.PUT\_LINE('Transfer failed: Insufficient balance in Account ' || p\_from\_acc);

END IF;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('One of the accounts does not exist.');

END;

/

**Output:**

