**Scenario 1:** Generate monthly statements for all customers. Write a PL/SQL block using an explicit cursor **GenerateMonthlyStatements** that retrieves all transactions for the current month and prints a statement for each customer.

**Ans :**

DECLARE

CURSOR customer\_cursor IS

SELECT DISTINCT c.CustomerID, c.Name

FROM Customers c

JOIN Accounts a ON c.CustomerID = a.CustomerID;

CURSOR transaction\_cursor (p\_customer\_id NUMBER) IS

SELECT TransactionID, TransactionDate, Amount, TransactionType

FROM Transactions

WHERE AccountID IN (SELECT AccountID FROM Accounts WHERE CustomerID = p\_customer\_id)

AND TransactionDate >= TRUNC(SYSDATE, 'MM')

AND TransactionDate < TRUNC(ADD\_MONTHS(SYSDATE, 1), 'MM');

v\_customer\_id NUMBER;

v\_customer\_name VARCHAR2(100);

v\_transaction\_id NUMBER;

v\_transaction\_date DATE;

v\_amount NUMBER;

v\_transaction\_type VARCHAR2(10);

v\_balance NUMBER := 0;

BEGIN

OPEN customer\_cursor;

LOOP

FETCH customer\_cursor INTO v\_customer\_id, v\_customer\_name;

EXIT WHEN customer\_cursor%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE('Statement for ' || v\_customer\_name);

DBMS\_OUTPUT.PUT\_LINE('---------------------');

OPEN transaction\_cursor(v\_customer\_id);

LOOP

FETCH transaction\_cursor INTO v\_transaction\_id, v\_transaction\_date, v\_amount, v\_transaction\_type;

EXIT WHEN transaction\_cursor%NOTFOUND;

v\_balance := v\_balance + v\_amount;

DBMS\_OUTPUT.PUT\_LINE(v\_transaction\_date || ' ' || v\_transaction\_type || ' ' || v\_amount || ' Balance: ' || v\_balance);

END LOOP;

CLOSE transaction\_cursor;

DBMS\_OUTPUT.PUT\_LINE('Total Balance: ' || v\_balance);

DBMS\_OUTPUT.PUT\_LINE('---------------------');

END LOOP;

CLOSE customer\_cursor;

END;

**Scenario 2 :** Apply annual fee to all accounts. Write a PL/SQL block using an explicit cursor **ApplyAnnualFee** that deducts an annual maintenance fee from the balance of all accounts.

**Ans :**

CREATE OR REPLACE PROCEDURE ApplyAnnualFee

AS

CURSOR accounts\_cursor IS

SELECT AccountID, Balance

FROM Accounts;

v\_account\_id NUMBER;

v\_balance NUMBER;

v\_annual\_fee NUMBER := 100; -- Replace with desired annual fee

BEGIN

OPEN accounts\_cursor;

LOOP

FETCH accounts\_cursor INTO v\_account\_id, v\_balance;

EXIT WHEN accounts\_cursor%NOTFOUND;

IF v\_balance >= v\_annual\_fee THEN

UPDATE Accounts

SET Balance = Balance - v\_annual\_fee

WHERE AccountID = v\_account\_id;

ELSE

-- Handle insufficient balance (e.g., log an error, disable account, etc.)

DBMS\_OUTPUT.PUT\_LINE('Insufficient balance for account ' || v\_account\_id);

END IF;

END LOOP;

CLOSE accounts\_cursor;

END;

**Scenario 3 :** Update the interest rate for all loans based on a new policy. a PL/SQL block using an explicit cursor **UpdateLoanInterestRates** that fetches all loans and updates their interest rates based on the new policy.

**Ans :**

DECLARE

CURSOR loans\_cursor IS

SELECT LoanID, InterestRate

FROM Loans

FOR UPDATE;

v\_loanID Loans.LoanID%TYPE;

v\_interestRate Loans.InterestRate%TYPE;

v\_new\_interestRate Loans.InterestRate%TYPE;

v\_errorMessage VARCHAR2(4000);

BEGIN

OPEN loans\_cursor;

LOOP

FETCH loans\_cursor INTO v\_loanID, v\_interestRate;

EXIT WHEN loans\_cursor%NOTFOUND;

-- New Rate increase policy (let increase by 0.5%)

v\_new\_interestRate := v\_interestRate + 0.5;

UPDATE Loans

SET InterestRate = v\_new\_interestRate

WHERE CURRENT OF loans\_cursor;

END LOOP;

COMMIT;

CLOSE loans\_cursor;

EXCEPTION

WHEN OTHERS THEN

v\_errorMessage := 'Error: ' || SQLERRM || ' while updating loan interest rates.';

INSERT INTO ErrorLog (ErrorDate, ErrorMessage) VALUES (SYSDATE, v\_errorMessage);

ROLLBACK;

END;

select \* from loans;