**Exercise 6: Cursors**

**Scenario 1: Generate monthly statements for all customers.**

* + Question: 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.

**Scenario 2: Apply annual fee to all accounts.**

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

**Scenario 3: Update the interest rate for all loans based on a new policy.**

* + Question: Write a PL/SQL block using an explicit cursor UpdateLoanInterestRates that fetches all loans and updates their interest rates based on the new policy.

**Scenario 1: Generate Monthly Statements for All Customers**

DECLARE

CURSOR transaction\_cursor IS

SELECT t.TransactionID, t.AccountID, t.TransactionDate, t.Amount, t.TransactionType, a.CustomerID, c.Name

FROM Transactions t

JOIN Accounts a ON t.AccountID = a.AccountID

JOIN Customers c ON a.CustomerID = c.CustomerID

WHERE TO\_CHAR(t.TransactionDate, 'YYYY-MM') = TO\_CHAR(SYSDATE, 'YYYY-MM');

v\_transaction\_id Transactions.TransactionID%TYPE;

v\_account\_id Transactions.AccountID%TYPE;

v\_transaction\_date Transactions.TransactionDate%TYPE;

v\_amount Transactions.Amount%TYPE;

v\_transaction\_type Transactions.TransactionType%TYPE;

v\_customer\_id Accounts.CustomerID%TYPE;

v\_customer\_name Customers.Name%TYPE;

BEGIN

OPEN transaction\_cursor;

LOOP

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

EXIT WHEN transaction\_cursor%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE('Customer: ' || v\_customer\_name);

DBMS\_OUTPUT.PUT\_LINE('Account ID: ' || v\_account\_id);

DBMS\_OUTPUT.PUT\_LINE('Transaction Date: ' || TO\_CHAR(v\_transaction\_date, 'YYYY-MM-DD'));

DBMS\_OUTPUT.PUT\_LINE('Transaction Type: ' || v\_transaction\_type);

DBMS\_OUTPUT.PUT\_LINE('Amount: ' || v\_amount);

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

END LOOP;

CLOSE transaction\_cursor;

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('An error occurred: ' || SQLERRM);

END;

/

**Scenario 2: Apply Annual Fee to All Accounts**

DECLARE

CURSOR account\_cursor IS

SELECT AccountID, Balance

FROM Accounts;

v\_account\_id Accounts.AccountID%TYPE;

v\_balance Accounts.Balance%TYPE;

ANNUAL\_FEE CONSTANT NUMBER := 50;

BEGIN

OPEN account\_cursor;

LOOP

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

EXIT WHEN account\_cursor%NOTFOUND;

v\_balance := v\_balance - ANNUAL\_FEE;

UPDATE Accounts

SET Balance = v\_balance

WHERE AccountID = v\_account\_id;

END LOOP;

CLOSE account\_cursor;

COMMIT;

EXCEPTION

WHEN OTHERS THEN

ROLLBACK;

DBMS\_OUTPUT.PUT\_LINE('An error occurred: ' || SQLERRM);

END;

/

**Scenario 3: Update the Interest Rate for All Loans Based on a New Policy**

DECLARE

CURSOR loan\_cursor IS

SELECT LoanID, InterestRate

FROM Loans;

v\_loan\_id Loans.LoanID%TYPE;

v\_interest\_rate Loans.InterestRate%TYPE;

INTEREST\_INCREMENT CONSTANT NUMBER := 0.5;

BEGIN

OPEN loan\_cursor;

LOOP

FETCH loan\_cursor INTO v\_loan\_id, v\_interest\_rate;

EXIT WHEN loan\_cursor%NOTFOUND;

v\_interest\_rate := v\_interest\_rate + INTEREST\_INCREMENT;

UPDATE Loans

SET InterestRate = v\_interest\_rate

WHERE LoanID = v\_loan\_id;

END LOOP;

CLOSE loan\_cursor;

COMMIT;

EXCEPTION

WHEN OTHERS THEN

ROLLBACK;

DBMS\_OUTPUT.PUT\_LINE('An error occurred: ' || SQLERRM);

END;

/