**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.

DECLARE

-- Cursor to retrieve all transactions for the current month

CURSOR cur\_Transactions IS

SELECT

t.TransactionID,

t.AccountID,

a.CustomerID,

t.TransactionDate,

t.Amount,

t.TransactionType

FROM Transactions t

JOIN Accounts a ON t.AccountID = a.AccountID

WHERE EXTRACT(MONTH FROM t.TransactionDate) = EXTRACT(MONTH FROM SYSDATE)

AND EXTRACT(YEAR FROM t.TransactionDate) = EXTRACT(YEAR FROM SYSDATE)

ORDER BY a.CustomerID, t.TransactionDate;

-- Record type to hold the data fetched by the cursor

rec\_Transaction cur\_Transactions%ROWTYPE;

-- Variable to hold the current customer ID for statement generation

v\_CurrentCustomerID Accounts.CustomerID%TYPE := NULL;

BEGIN

-- Open the cursor

OPEN cur\_Transactions;

-- Loop through each transaction

LOOP

FETCH cur\_Transactions INTO rec\_Transaction;

EXIT WHEN cur\_Transactions%NOTFOUND;

-- Check if the customer ID has changed (indicating a new statement)

IF v\_CurrentCustomerID IS NULL OR v\_CurrentCustomerID != rec\_Transaction.CustomerID THEN

-- Print statement header for the new customer

IF v\_CurrentCustomerID IS NOT NULL THEN

DBMS\_OUTPUT.PUT\_LINE('End of Statement for Customer ID ' || v\_CurrentCustomerID);

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

END IF;

v\_CurrentCustomerID := rec\_Transaction.CustomerID;

DBMS\_OUTPUT.PUT\_LINE('Statement for Customer ID ' || v\_CurrentCustomerID);

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

END IF;

-- Print transaction details

DBMS\_OUTPUT.PUT\_LINE('Transaction ID: ' || rec\_Transaction.TransactionID);

DBMS\_OUTPUT.PUT\_LINE('Account ID: ' || rec\_Transaction.AccountID);

DBMS\_OUTPUT.PUT\_LINE('Date: ' || rec\_Transaction.TransactionDate);

DBMS\_OUTPUT.PUT\_LINE('Amount: ' || rec\_Transaction.Amount);

DBMS\_OUTPUT.PUT\_LINE('Type: ' || rec\_Transaction.TransactionType);

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

END LOOP;

-- Print the end of the last customer’s statement

IF v\_CurrentCustomerID IS NOT NULL THEN

DBMS\_OUTPUT.PUT\_LINE('End of Statement for Customer ID ' || v\_CurrentCustomerID);

END IF;

-- Close the cursor

CLOSE cur\_Transactions;

END;

/

**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.

DECLARE

-- Define the annual fee amount

v\_AnnualFee NUMBER := 50; -- Adjust the fee amount as needed

-- Cursor to retrieve all accounts

CURSOR cur\_Accounts IS

SELECT AccountID, Balance

FROM Accounts

FOR UPDATE;

-- Record type to hold data fetched by the cursor

rec\_Account cur\_Accounts%ROWTYPE;

BEGIN

-- Open the cursor

OPEN cur\_Accounts;

-- Loop through each account

LOOP

FETCH cur\_Accounts INTO rec\_Account;

EXIT WHEN cur\_Accounts%NOTFOUND;

-- Deduct the annual fee from the account balance

UPDATE Accounts

SET Balance = Balance - v\_AnnualFee

WHERE AccountID = rec\_Account.AccountID;

-- Optionally, print the update for verification

DBMS\_OUTPUT.PUT\_LINE('Account ID: ' || rec\_Account.AccountID || ' Balance updated to: ' || (rec\_Account.Balance - v\_AnnualFee));

END LOOP;

-- Close the cursor

CLOSE cur\_Accounts;

-- Commit the transaction

COMMIT;

-- Optionally, print completion message

DBMS\_OUTPUT.PUT\_LINE('Annual fee applied to all accounts.');

EXCEPTION

WHEN OTHERS THEN

-- Rollback in case of error

ROLLBACK;

-- Print error message

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

END;

/

**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.

DECLARE

-- Define the percentage increase in interest rate

v\_InterestRateIncrease NUMBER := 0.02; -- 2% increase, adjust as needed

-- Cursor to retrieve all loans

CURSOR cur\_Loans IS

SELECT LoanID, InterestRate

FROM Loans

FOR UPDATE;

-- Record type to hold data fetched by the cursor

rec\_Loan cur\_Loans%ROWTYPE;

BEGIN

-- Open the cursor

OPEN cur\_Loans;

-- Loop through each loan

LOOP

FETCH cur\_Loans INTO rec\_Loan;

EXIT WHEN cur\_Loans%NOTFOUND;

-- Calculate the new interest rate

-- Ensure the new rate does not exceed a maximum allowable rate if necessary

UPDATE Loans

SET InterestRate = InterestRate + (InterestRate \* v\_InterestRateIncrease)

WHERE CURRENT OF cur\_Loans;

-- Optionally, print the update for verification

DBMS\_OUTPUT.PUT\_LINE('Loan ID: ' || rec\_Loan.LoanID ||

' Interest Rate updated to: ' ||

(rec\_Loan.InterestRate + (rec\_Loan.InterestRate \* v\_InterestRateIncrease)));

END LOOP;

-- Close the cursor

CLOSE cur\_Loans;

-- Commit the transaction

COMMIT;

-- Optionally, print completion message

DBMS\_OUTPUT.PUT\_LINE('Interest rates updated for all loans based on new policy.');

EXCEPTION

WHEN OTHERS THEN

-- Rollback in case of error

ROLLBACK;

-- Print error message

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

END;

/