**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 txn\_cursor IS

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

FROM Customers c

JOIN Accounts a ON c.CustomerID = a.CustomerID

JOIN Transactions t ON a.AccountID = t.AccountID

WHERE TRUNC(t.TransactionDate, 'MM') = TRUNC(SYSDATE, 'MM')

ORDER BY c.CustomerID, t.TransactionDate;

v\_customer\_id Customers.CustomerID%TYPE;

v\_name Customers.Name%TYPE;

v\_account\_id Accounts.AccountID%TYPE;

v\_txn\_date Transactions.TransactionDate%TYPE;

v\_amount Transactions.Amount%TYPE;

v\_type Transactions.TransactionType%TYPE;

BEGIN

DBMS\_OUTPUT.PUT\_LINE('Monthly Statement for Transactions in ' || TO\_CHAR(SYSDATE, 'Month YYYY'));

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

OPEN txn\_cursor;

LOOP

FETCH txn\_cursor INTO v\_customer\_id, v\_name, v\_account\_id, v\_txn\_date, v\_amount, v\_type;

EXIT WHEN txn\_cursor%NOTFOUND;

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

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

DBMS\_OUTPUT.PUT\_LINE(' Date: ' || TO\_CHAR(v\_txn\_date, 'DD-MON-YYYY') ||

', Type: ' || v\_type || ', Amount: ' || v\_amount);

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

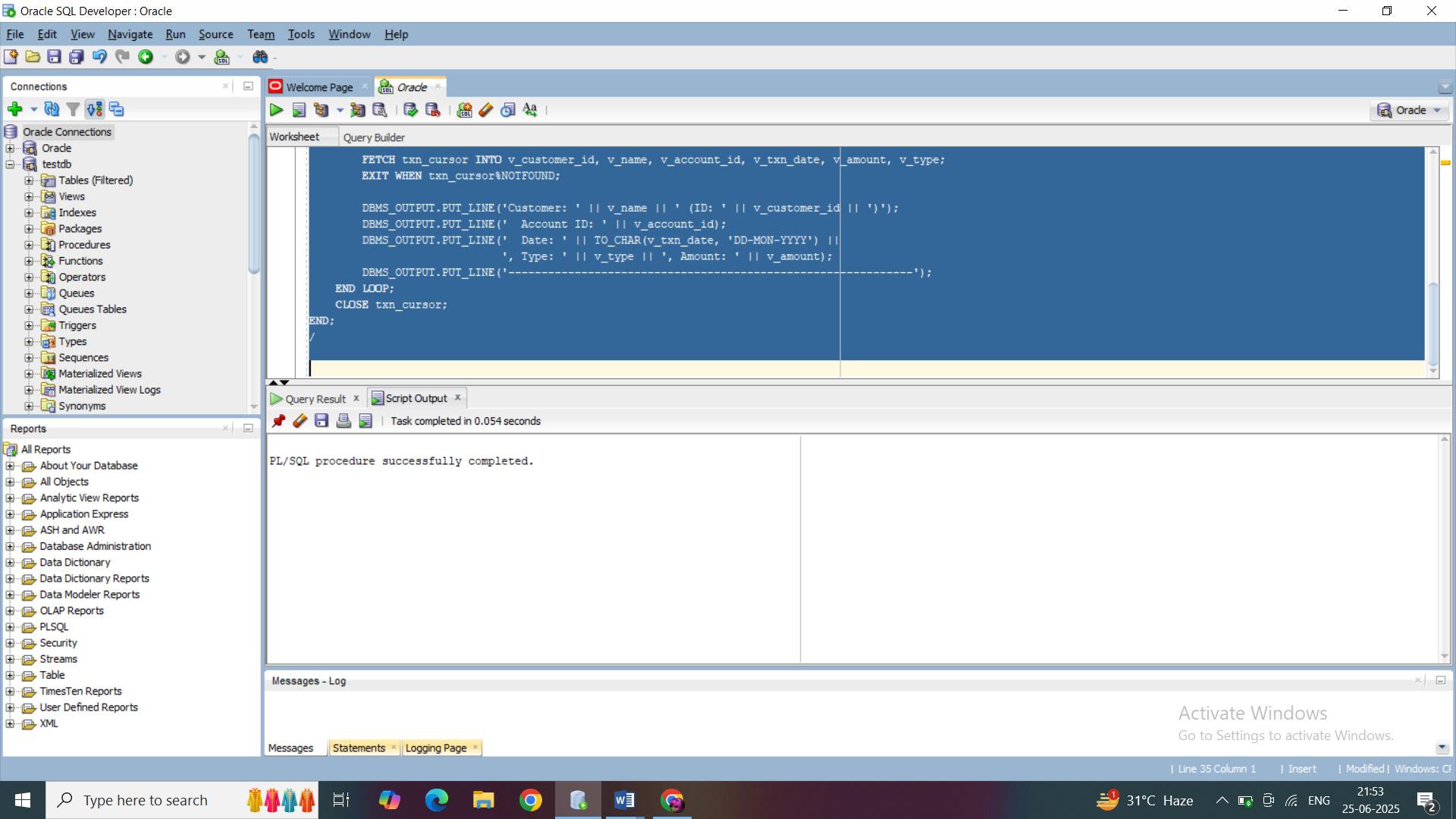
END LOOP;

CLOSE txn\_cursor;

END;

/

OUTPUT:



**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

CURSOR acc\_cursor IS

SELECT AccountID, Balance FROM Accounts FOR UPDATE;

v\_account\_id Accounts.AccountID%TYPE;

v\_balance Accounts.Balance%TYPE;

annual\_fee CONSTANT NUMBER := 100;

BEGIN

OPEN acc\_cursor;

LOOP

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

EXIT WHEN acc\_cursor%NOTFOUND;

IF v\_balance >= annual\_fee THEN

UPDATE Accounts

SET Balance = Balance - annual\_fee,

LastModified = SYSDATE

WHERE CURRENT OF acc\_cursor;

DBMS\_OUTPUT.PUT\_LINE('Fee applied to Account ID ' || v\_account\_id ||

', New Balance: ' || (v\_balance - annual\_fee));

ELSE

DBMS\_OUTPUT.PUT\_LINE('Insufficient balance in Account ID ' || v\_account\_id ||

', Fee not applied.');

END IF;

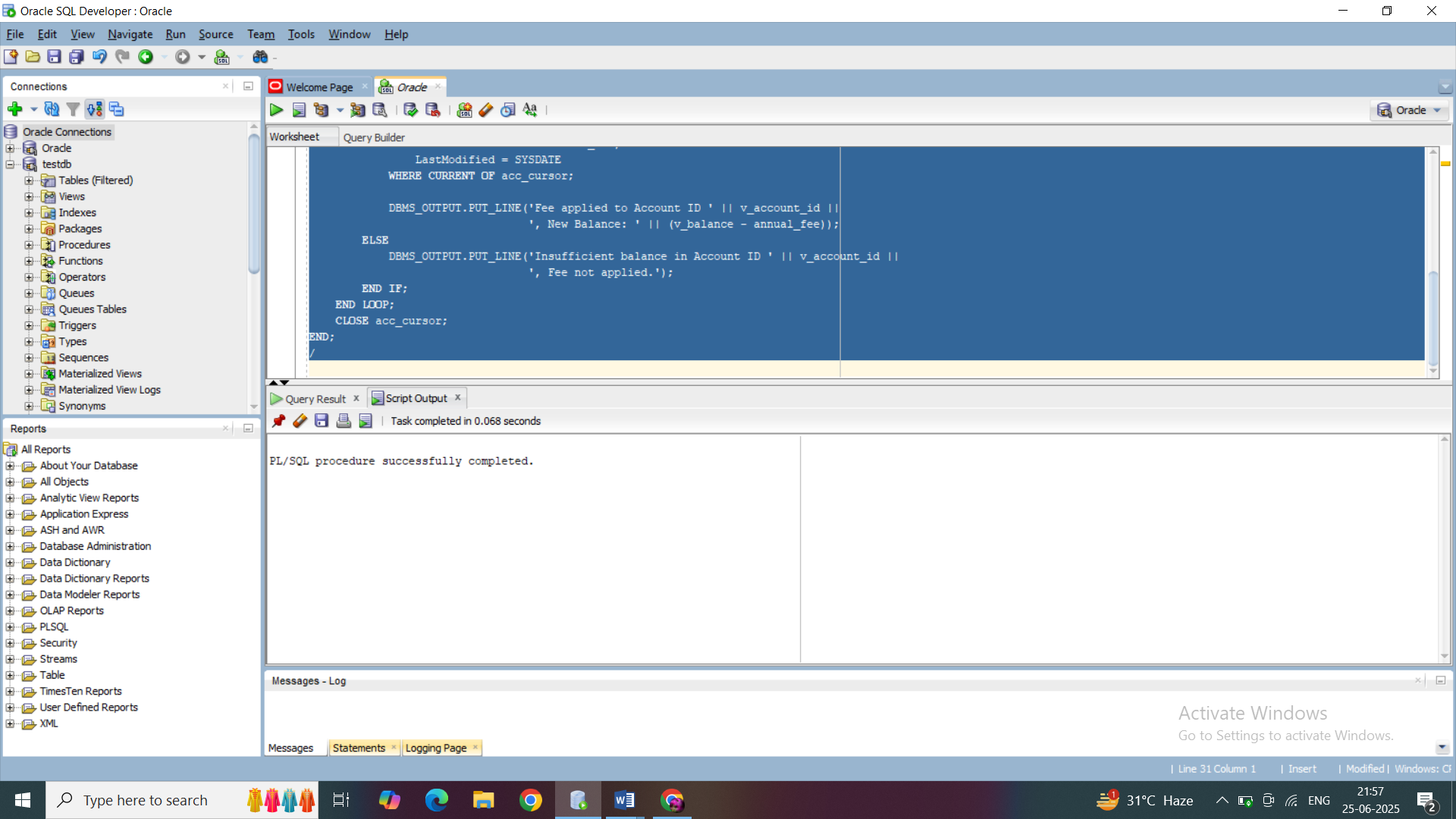
END LOOP;

CLOSE acc\_cursor;

END;

/

OUTPUT:



**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

CURSOR loan\_cursor IS

SELECT LoanID, InterestRate FROM Loans FOR UPDATE;

v\_loan\_id Loans.LoanID%TYPE;

v\_interest Loans.InterestRate%TYPE;

increment\_rate CONSTANT NUMBER := 0.5; -- Example new policy

BEGIN

OPEN loan\_cursor;

LOOP

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

EXIT WHEN loan\_cursor%NOTFOUND;

UPDATE Loans

SET InterestRate = v\_interest + increment\_rate

WHERE CURRENT OF loan\_cursor;

DBMS\_OUTPUT.PUT\_LINE('Updated Loan ID ' || v\_loan\_id ||

', Old Rate: ' || v\_interest ||

', New Rate: ' || (v\_interest + increment\_rate));

END LOOP;

CLOSE loan\_cursor;

END;

/  
  
OUTPUT:

