**Exercise 3: Stored Procedures**

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

**Question:** Write a stored procedure **ProcessMonthlyInterest** that calculates and updates the balance of all savings accounts by applying an interest rate of 1% to the current balance.

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest IS

CURSOR savings\_accounts IS

SELECT AccountID, Balance FROM Accounts WHERE AccountType = 'Savings' FOR UPDATE;

v\_new\_balance NUMBER;

BEGIN

FOR acc IN savings\_accounts LOOP

v\_new\_balance := acc.Balance + (acc.Balance \* 0.01);

UPDATE Accounts

SET Balance = v\_new\_balance,

LastModified = SYSDATE

WHERE AccountID = acc.AccountID;

DBMS\_OUTPUT.PUT\_LINE('Interest applied: AccountID=' || acc.AccountID ||

', Old Balance=' || acc.Balance ||

', New Balance=' || v\_new\_balance);

END LOOP;

COMMIT;

EXCEPTION

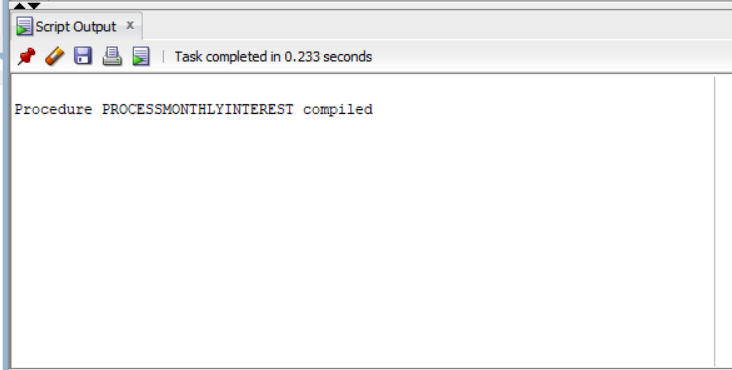
WHEN OTHERS THEN

ROLLBACK;

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

END;

/  
  
OUTPUT:



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

**Question:** Write a stored procedure **UpdateEmployeeBonus** that updates the salary of employees in a given department by adding a bonus percentage passed as a parameter.

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus(

p\_Department IN VARCHAR2,

p\_BonusPercent IN NUMBER

) AS

CURSOR emp\_cursor IS

SELECT EmployeeID, Name, Salary FROM Employees WHERE Department = p\_Department FOR UPDATE;

v\_new\_salary NUMBER;

BEGIN

FOR emp IN emp\_cursor LOOP

v\_new\_salary := emp.Salary + (emp.Salary \* p\_BonusPercent / 100);

UPDATE Employees

SET Salary = v\_new\_salary

WHERE EmployeeID = emp.EmployeeID;

DBMS\_OUTPUT.PUT\_LINE('Bonus applied: EmployeeID=' || emp.EmployeeID ||

', Name=' || emp.Name ||

', Old Salary=' || emp.Salary ||

', New Salary=' || v\_new\_salary);

END LOOP;

IF emp\_cursor%ROWCOUNT = 0 THEN

DBMS\_OUTPUT.PUT\_LINE('No employees found in department: ' || p\_Department);

END IF;

COMMIT;

EXCEPTION

WHEN OTHERS THEN

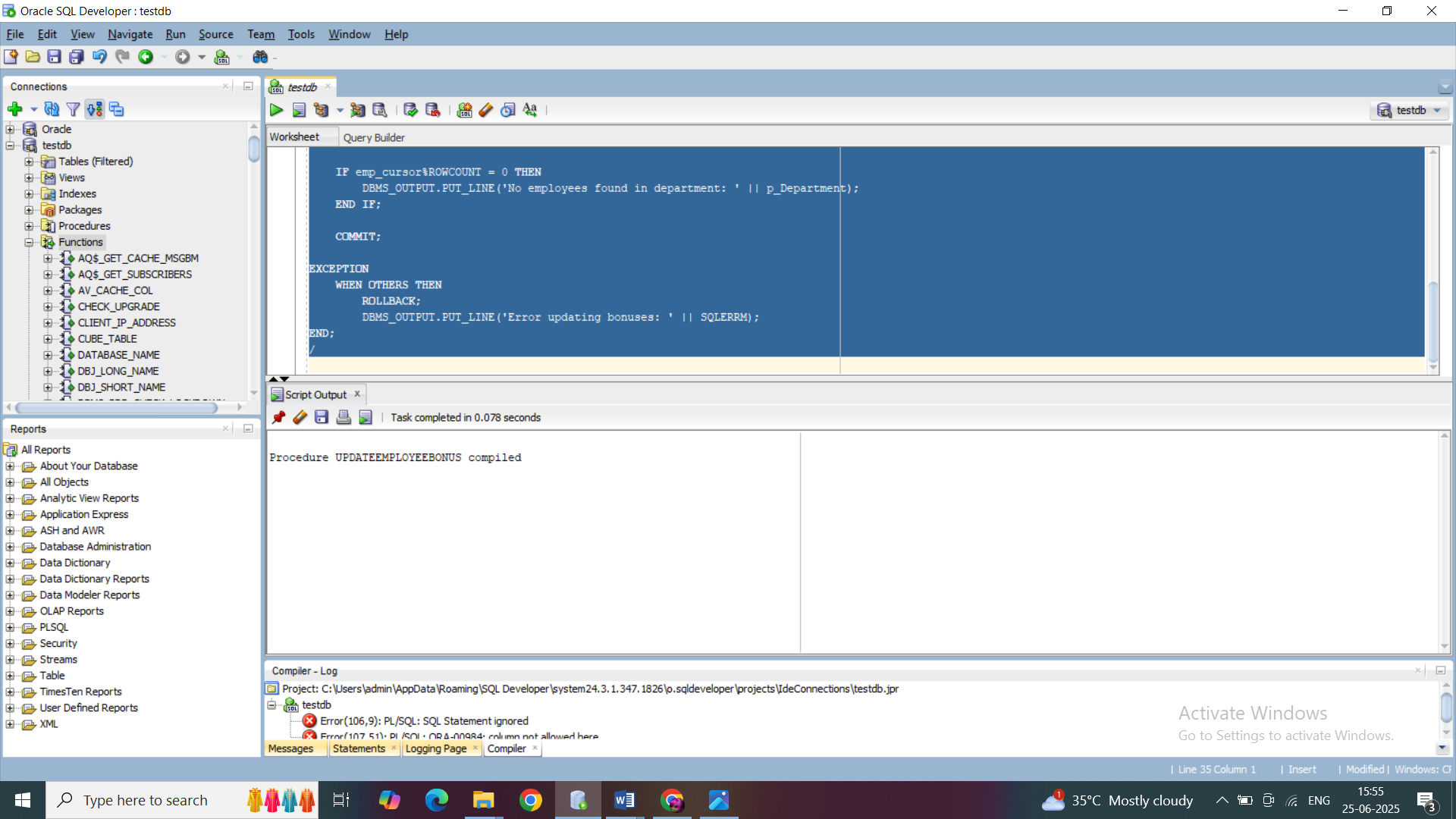
ROLLBACK;

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

END;

/

OUTPUT:



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

**Question:** Write a stored procedure **TransferFunds** that transfers a specified amount from one account to another, checking that the source account has sufficient balance before making the transfer.

CREATE OR REPLACE PROCEDURE TransferFunds(

p\_SourceAccountID IN NUMBER,

p\_TargetAccountID IN NUMBER,

p\_Amount IN NUMBER

) AS

v\_SourceBalance NUMBER;

v\_TargetBalance NUMBER;

BEGIN

-- Lock and get source balance

SELECT Balance INTO v\_SourceBalance FROM Accounts

WHERE AccountID = p\_SourceAccountID FOR UPDATE;

IF v\_SourceBalance < p\_Amount THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Insufficient balance in source account.');

END IF;

-- Lock and get target balance

SELECT Balance INTO v\_TargetBalance FROM Accounts

WHERE AccountID = p\_TargetAccountID FOR UPDATE;

-- Update source

UPDATE Accounts

SET Balance = v\_SourceBalance - p\_Amount,

LastModified = SYSDATE

WHERE AccountID = p\_SourceAccountID;

-- Update target

UPDATE Accounts

SET Balance = v\_TargetBalance + p\_Amount,

LastModified = SYSDATE

WHERE AccountID = p\_TargetAccountID;

-- Transactions

INSERT INTO Transactions (TransactionID, AccountID, TransactionDate, Amount, TransactionType)

VALUES (Transactions\_SEQ.NEXTVAL, p\_SourceAccountID, SYSDATE, p\_Amount, 'TransferOut');

INSERT INTO Transactions (TransactionID, AccountID, TransactionDate, Amount, TransactionType)

VALUES (Transactions\_SEQ.NEXTVAL, p\_TargetAccountID, SYSDATE, p\_Amount, 'TransferIn');

DBMS\_OUTPUT.PUT\_LINE('Transfer Successful!');

DBMS\_OUTPUT.PUT\_LINE('From Account: ' || p\_SourceAccountID || ', Deducted: ' || p\_Amount ||

', New Balance: ' || (v\_SourceBalance - p\_Amount));

DBMS\_OUTPUT.PUT\_LINE('To Account: ' || p\_TargetAccountID || ', Added: ' || p\_Amount ||

', New Balance: ' || (v\_TargetBalance + p\_Amount));

COMMIT;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

ROLLBACK;

DBMS\_OUTPUT.PUT\_LINE('Invalid source or target account ID.');

WHEN OTHERS THEN

ROLLBACK;

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

END;

/

OUTPUT:

