**EXERCISE 5: TRIGGERS**

**Scenario 1: UpdateCustomerLastModified - Automatically Update Last Modified Date**

CREATE OR REPLACE TRIGGER UpdateCustomerLastModified

AFTER UPDATE ON Customers

FOR EACH ROW

BEGIN

-- Update the LastModified date to the current date

:NEW.LastModified := SYSDATE;

-- Log the update for verification

DBMS\_OUTPUT.PUT\_LINE('Updated LastModified for Customer ID ' || :NEW.CustomerID);

END UpdateCustomerLastModified;

/

**Scenario 2: LogTransaction - Maintain an Audit Log for Transactions**

CREATE TABLE AuditLog (

AuditID NUMBER PRIMARY KEY,

TransactionID NUMBER,

AccountID NUMBER,

TransactionDate DATE,

Amount NUMBER,

TransactionType VARCHAR2(10),

LogDate DATE

);

CREATE OR REPLACE TRIGGER LogTransaction

AFTER INSERT ON Transactions

FOR EACH ROW

BEGIN

-- Insert a record into the AuditLog table

INSERT INTO AuditLog (

AuditID,

TransactionID,

AccountID,

TransactionDate,

Amount,

TransactionType,

LogDate

) VALUES (

AuditLog\_seq.NEXTVAL, -- Assuming a sequence named AuditLog\_seq is used to generate AuditID

:NEW.TransactionID,

:NEW.AccountID,

:NEW.TransactionDate,

:NEW.Amount,

:NEW.TransactionType,

SYSDATE -- Current date as the log date

);

-- Log the audit for verification

DBMS\_OUTPUT.PUT\_LINE('Logged transaction ID ' || :NEW.TransactionID || ' to AuditLog');

END LogTransaction;

/

CREATE SEQUENCE AuditLog\_seq

START WITH 1

INCREMENT BY 1;

**Scenario 3: CheckTransactionRules - Enforce Business Rules on Transactions**

CREATE OR REPLACE TRIGGER CheckTransactionRules

BEFORE INSERT ON Transactions

FOR EACH ROW

DECLARE

v\_Balance Accounts.Balance%TYPE;

BEGIN

-- Fetch the current balance of the account

SELECT Balance INTO v\_Balance

FROM Accounts

WHERE AccountID = :NEW.AccountID;

-- Check if the transaction is a withdrawal and exceeds the current balance

IF :NEW.TransactionType = 'Withdrawal' AND :NEW.Amount > v\_Balance THEN

RAISE\_APPLICATION\_ERROR(-20002, 'Withdrawal exceeds account balance.');

END IF;

-- Check if the transaction is a deposit and the amount is non-positive

IF :NEW.TransactionType = 'Deposit' AND :NEW.Amount <= 0 THEN

RAISE\_APPLICATION\_ERROR(-20003, 'Deposit amount must be positive.');

END IF;

-- Log successful validation for verification

DBMS\_OUTPUT.PUT\_LINE('Transaction validation passed for account ID ' || :NEW.AccountID);

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

RAISE\_APPLICATION\_ERROR(-20004, 'Account not found.');

END CheckTransactionRules;

/