Exercise 5:

-- Drop tables if they already exist

BEGIN

EXECUTE IMMEDIATE 'DROP TABLE AuditLog';

EXECUTE IMMEDIATE 'DROP TABLE Transactions';

EXECUTE IMMEDIATE 'DROP TABLE Accounts';

EXECUTE IMMEDIATE 'DROP TABLE Customers';

EXCEPTION

WHEN OTHERS THEN NULL;

END;

/

-- Customers table

CREATE TABLE Customers (

CustomerID NUMBER PRIMARY KEY,

Name VARCHAR2(100),

DOB DATE,

Balance NUMBER,

LastModified DATE

);

-- Accounts table

CREATE TABLE Accounts (

AccountID NUMBER PRIMARY KEY,

CustomerID NUMBER,

AccountType VARCHAR2(20),

Balance NUMBER,

LastModified DATE,

FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)

);

-- Transactions table

CREATE TABLE Transactions (

TransactionID NUMBER PRIMARY KEY,

AccountID NUMBER,

TransactionDate DATE,

Amount NUMBER,

TransactionType VARCHAR2(10),

FOREIGN KEY (AccountID) REFERENCES Accounts(AccountID)

);

-- AuditLog table

CREATE TABLE AuditLog (

LogID NUMBER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,

ActionDate DATE,

AccountID NUMBER,

TransactionType VARCHAR2(10),

Amount NUMBER

);

-- Insert sample customer and account

INSERT INTO Customers VALUES (1, 'John Doe', TO\_DATE('1985-01-01', 'YYYY-MM-DD'), 1000, SYSDATE);

INSERT INTO Accounts VALUES (101, 1, 'Savings', 5000, SYSDATE);

CREATE OR REPLACE TRIGGER UpdateCustomerLastModified

BEFORE UPDATE ON Customers

FOR EACH ROW

BEGIN

:NEW.LastModified := SYSDATE;

END;

/

UPDATE Customers SET Balance = Balance + 200 WHERE CustomerID = 1;

SELECT \* FROM Customers;

CREATE OR REPLACE TRIGGER LogTransaction

AFTER INSERT ON Transactions

FOR EACH ROW

BEGIN

INSERT INTO AuditLog (ActionDate, AccountID, TransactionType, Amount)

VALUES (SYSDATE, :NEW.AccountID, :NEW.TransactionType, :NEW.Amount);

END;

/

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

VALUES (1, 101, SYSDATE, 1000, 'Deposit');

SELECT \* FROM AuditLog;

CREATE OR REPLACE TRIGGER CheckTransactionRules

BEFORE INSERT ON Transactions

FOR EACH ROW

DECLARE

v\_balance NUMBER;

BEGIN

-- Check for Withdrawal

IF :NEW.TransactionType = 'Withdrawal' THEN

SELECT Balance INTO v\_balance

FROM Accounts

WHERE AccountID = :NEW.AccountID;

IF :NEW.Amount > v\_balance THEN

DBMS\_OUTPUT.PUT\_LINE('Transaction failed: Insufficient balance');

:NEW.Amount := NULL; -- Block insert silently

END IF;

ELSIF :NEW.TransactionType = 'Deposit' THEN

IF :NEW.Amount <= 0 THEN

DBMS\_OUTPUT.PUT\_LINE('Transaction failed: Deposit must be positive');

:NEW.Amount := NULL;

END IF;

END IF;

END;

/

-- Valid Deposit

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

VALUES (2, 101, SYSDATE, 500, 'Deposit');

-- Invalid Withdrawal (exceeds balance)

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

VALUES (3, 101, SYSDATE, 999999, 'Withdrawal');

-- Valid Withdrawal

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

VALUES (4, 101, SYSDATE, 1000, 'Withdrawal');

S