Exercise 5: Triggers

Scenario 1: (Ex5-Scenario1.sql)

@InitializeData.sql

```
SET ECHO ON

SET SERVEROUTPUT ON SIZE UNLIMITED

SPOOL output-Ex5-Scenario1.txt

VARIABLE input VARCHAR2(30)

-- Trigger to automatically update the LastModified date when a customer's record is updated
CREATE OR REPLACE TRIGGER UpdateCustomerLastModified
BEFORE UPDATE ON CUSTOMERS
FOR EACH ROW
BEGIN
   :NEW.LastModified := SYSDATE;
END;
/

-- Before Updation
SELECT * FROM CUSTOMERS;

-- Test the trigger by updating a customer record
UPDATE CUSTOMERS
SET NAME = 'Richard Nomad'
WHERE CUSTOMERID = 1;
```

```
-- Verify the change
SELECT * FROM CUSTOMERS;

SPOOL OFF

@DropData.sql
```

Scenario 2: (Ex5-Scenario2.sql)

```
@InitializeData.sql

SET ECHO ON

SET SERVEROUTPUT ON SIZE UNLIMITED

SPOOL output-Ex5-Scenario2.txt

VARIABLE input VARCHAR2(30)

-- Create the AuditLog table
CREATE TABLE AuditLog (
   LogID NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
   TransactionID NUMBER,
   LogDate DATE,
   Action VARCHAR2(50)
);

-- Trigger to insert a record into the AuditLog table whenever a transaction is inserted
CREATE OR REPLACE TRIGGER LogTransaction
```

```
AFTER INSERT ON TRANSACTIONS
FOR EACH ROW
BEGIN
   INSERT INTO AuditLog (TransactionID, LogDate, Action)
   VALUES (:NEW.TransactionID, SYSDATE, 'Transaction Inserted');
END;
/

-- Test the trigger by inserting a transaction
INSERT INTO TRANSACTIONS (TransactionID, ACCOUNTID, TRANSACTIONDATE, AMOUNT,
TRANSACTIONTYPE)
VALUES (3001, 1, SYSDATE, 500, 'Deposit');

-- Verify the audit log entry
SELECT * FROM AuditLog;

SPOOL OFF

DROP TABLE AuditLog;

@DropData.sql
```

Scenario 3: (Ex5-Scenario3.sql)

```
@InitializeData.sql

SET ECHO ON

SET SERVEROUTPUT ON SIZE UNLIMITED

SPOOL output-Ex5-Scenario3.txt
```

```
VARIABLE input VARCHAR2(30)

-- Trigger to enforce business rules on deposits and withdrawals
CREATE OR REPLACE TRIGGER CheckTransactionRules
BEFORE INSERT ON TRANSACTIONS
FOR EACH ROW
DECLARE
   v_balance NUMBER;
BEGIN
   -- Get the current balance for the account
   SELECT BALANCE
   INTO v_balance
   FROM ACCOUNTS
   WHERE ACCOUNTID = :NEW.ACCOUNTID;

   -- Check for withdrawal rules
   IF :NEW.TRANSACTIONTYPE = 'Withdrawal' THEN
      IF :NEW.AMOUNT > v_balance THEN
         RAISE_APPLICATION_ERROR(-20001, 'Insufficient funds for withdrawal');
      END IF;
   END IF;

   -- Check for deposit rules
   IF :NEW.TRANSACTIONTYPE = 'Deposit' AND :NEW.AMOUNT <= 0 THEN
      RAISE_APPLICATION_ERROR(-20002, 'Deposit amount must be positive');
   END IF;
END;
/

-- Test the trigger by inserting valid and invalid transactions
```

```
BEGIN
   -- Valid Deposit
   INSERT INTO TRANSACTIONS (TRANSACTIONID, ACCOUNTID, TRANSACTIONDATE, AMOUNT,
TRANSACTIONTYPE)
   VALUES (3002, 1, SYSDATE, 500, 'Deposit');

   -- Invalid Withdrawal (Insufficient funds)
   BEGIN
      INSERT INTO TRANSACTIONS (TRANSACTIONID, ACCOUNTID, TRANSACTIONDATE, AMOUNT,
TRANSACTIONTYPE)
      VALUES (3003, 2, SYSDATE, 10000, 'Withdrawal');
   EXCEPTION
      WHEN OTHERS THEN
         DBMS_OUTPUT.PUT_LINE(SQLERRM);
   END;
END;
/

-- Verify the transactions
SELECT * FROM TRANSACTIONS;

SPOOL OFF

@DropData.sql
```