Step 1:

Problem Statement – Advanced Database Programming and Auditing

In a biometric-based ticketing and payment system like **PalmPayment**, which manages transactions, authentication, and user records, **data integrity and operational control are critical**. The system handles sensitive operations such as financial transactions and biometric authentications, which—if manipulated maliciously or accidentally during business hours—could lead to **fraud, inconsistency, or system failure**.

The Problem:

Our system is vulnerable to unauthorized or unintended **INSERT**, **UPDATE**, or **DELETE** operations—especially by employees during **weekday business hours** or on **designated public holidays**. Such actions, if not properly restricted, could compromise transaction accuracy, tamper with audit trails, and impact user trust.

✅ Why Advanced Programming Is Needed:

To mitigate these risks, we must implement **advanced PL/SQL techniques** that introduce automation, control, and accountability into our system:

- **Triggers** are required to enforce restrictions and prevent DML operations during:
  - Weekdays (Monday–Friday)
  - Upcoming month's public holidays (stored in a reference table)
- **Packages and functions** will modularize reusable logic such as:
  - Checking whether a date is restricted (weekday or holiday)
  - Centralizing validation and security logic
- **Auditing mechanisms** are necessary to:
  - Track and log every DML attempt on sensitive tables
  - Capture metadata such as the user who attempted the change, timestamp, type of operation, and whether it was **allowed or denied**
  - Enable accountability and review of user activity

Restriction Rule Summary:

To enforce secure and policy-compliant operations:

- **Employees are restricted** from making changes (INSERT/UPDATE/DELETE) on:
  - Weekdays (Monday to Friday)
  - Public holidays in the upcoming month (stored in a static reference table)

- **All operations will be monitored** using:
  - **Triggers** to block DML
  - **Audit table** to record allowed/denied actions
  - **Packages/functions** to manage reusable auditing logic

STEP 2: TRIGGER IMPLEMENTATION

A. Creating and inserting data in the Reference Table for Holidays

```
CREATE TABLE Holidays (
   HolidayDate DATE PRIMARY KEY,
   Description VARCHAR2(100)
);

INSERT INTO Holidays (HolidayDate, Description) VALUES (TO_DATE('2025-06-01', 'YYYY-MM-DD'),
'Independence Day');
INSERT INTO Holidays (HolidayDate, Description) VALUES (TO_DATE('2025-06-07', 'YYYY-MM-DD'),
'National Heroes Day');

COMMIT;
```

B. SIMPLE TRIGGER – Prevent INSERT, UPDATE, DELETE on Weekdays & Holidays

Here's a **BEFORE INSERT OR UPDATE OR DELETE** trigger on a sensitive table like **Transaction:**

SYNTAX

```
CREATE OR REPLACE TRIGGER trg_prevent_dml_weekday_holiday
BEFORE INSERT OR UPDATE OR DELETE ON Transaction
FOR EACH ROW
DECLARE
   v_today DATE := SYSDATE;
   v_day   VARCHAR2(10);
   v_holiday_count INT;
BEGIN
   v_day := TO_CHAR(v_today, 'DY', 'NLS_DATE_LANGUAGE=ENGLISH');

   SELECT COUNT(*) INTO v_holiday_count
   FROM Holidays
   WHERE HolidayDate = TRUNC(v_today)
     AND HolidayDate BETWEEN TRUNC(SYSDATE)
                         AND TRUNC(ADD_MONTHS(SYSDATE, 1));
```

```
  IF v_day IN ('MON', 'TUE', 'WED', 'THU', 'FRI') OR v_holiday_count > 0 THEN
    RAISE_APPLICATION_ERROR(-20001, '❌ DML operations are blocked on weekdays or holidays within the upcoming month.');
  END IF;
END;
/
```

1. Testing the Simple Trigger: trg_prevent_dml_weekday_holiday

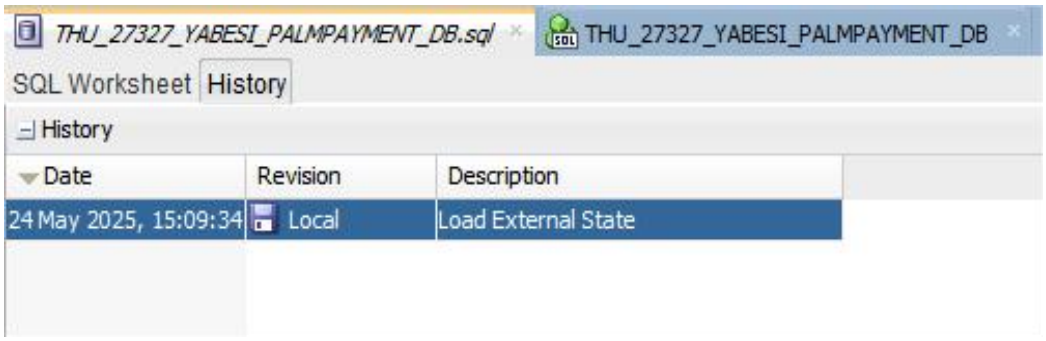
Purpose:

Block **INSERT, UPDATE, or DELETE** on the **Transaction** table:

- On weekdays (Monday–Friday)
- On any date found in the **Holidays** table

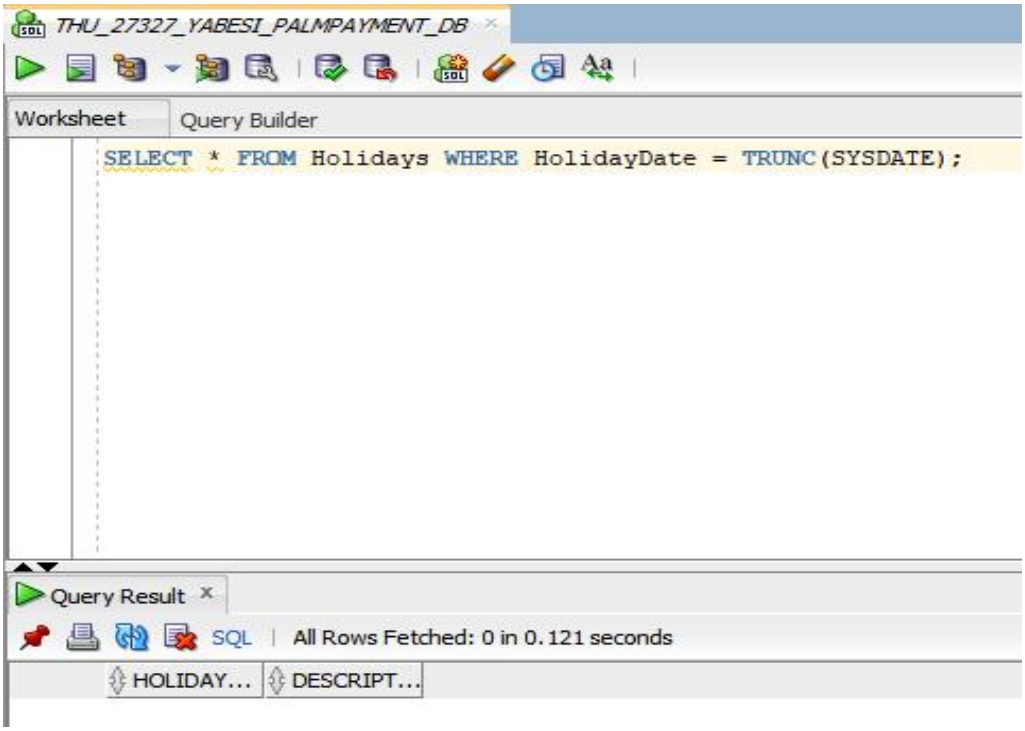
TEST 1 :Check Today's Date and Day

SELECT TO_CHAR(SYSDATE, 'DAY', 'NLS_DATE_LANGUAGE=ENGLISH') AS Today FROM DUAL;



TEST 2 :CHECK IF TODAY IS A HOLIDAY

SELECT * FROM Holidays WHERE HolidayDate = TRUNC(SYSDATE);



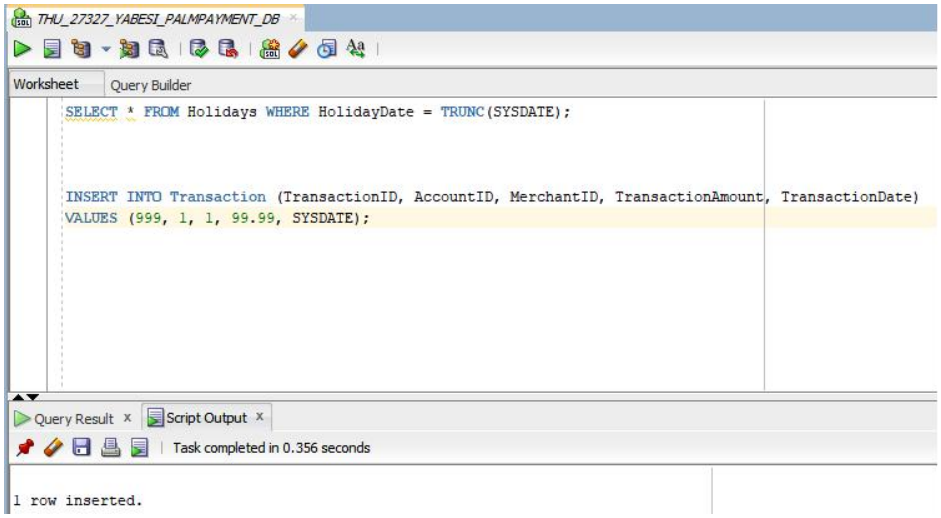
There is no rows fetched because the day 24/05/2025 is no a holiday in the next month

TEST 3: Try a Blocked Operation (INSERT)

If today is a weekend and not in the Holidays table

SYNTAX

INSERT INTO Transaction (TransactionID, AccountID, MerchantID, TransactionAmount, TransactionDate)
VALUES (1000, 2, 1, 49.99, SYSDATE);



Result: the test succedded because today is a weekend and not in the Holidays table

C. COMPOUND TRIGGER

SYNTAX

```
CREATE OR REPLACE TRIGGER trg_compound_dml_blocker
FOR INSERT OR UPDATE OR DELETE ON Transaction
COMPOUND TRIGGER

  -- Shared variables across all trigger sections
  v_day      VARCHAR2(10);
  v_holiday  INT;
  v_status   VARCHAR2(10) := 'ALLOWED';
  v_user     VARCHAR2(50);
  v_table    VARCHAR2(50) := 'Transaction';
  v_block    BOOLEAN := FALSE;
  v_reason   VARCHAR2(200);

BEFORE STATEMENT IS
BEGIN
```

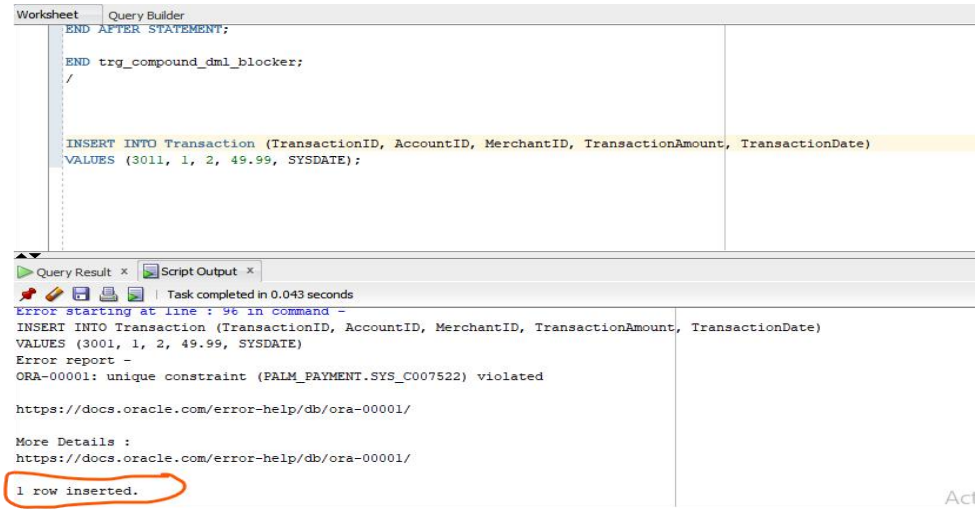
```
    v_user := SYS_CONTEXT('USERENV', 'SESSION_USER');

    v_day := TO_CHAR(SYSDATE, 'DY', 'NLS_DATE_LANGUAGE=ENGLISH');

    SELECT COUNT(*) INTO v_holiday
    FROM Holidays
    WHERE HolidayDate = TRUNC(SYSDATE)
      AND HolidayDate BETWEEN TRUNC(SYSDATE) AND TRUNC(ADD_MONTHS(SYSDATE, 1));

    IF v_day IN ('MON', 'TUE', 'WED', 'THU', 'FRI') OR v_holiday > 0 THEN
        v_block := TRUE;
        v_status := 'DENIED';
        v_reason := 'DML blocked on weekday or holiday';
    END IF;
END BEFORE STATEMENT;

BEFORE EACH ROW IS
BEGIN
    IF v_block THEN
        RAISE_APPLICATION_ERROR(-20002, '❌ DML blocked on weekday or holiday');
    END IF;
END BEFORE EACH ROW;

AFTER STATEMENT IS
BEGIN
    -- Log user attempt
    INSERT INTO Audit_Log (Username, Action, ActionTime, Status, TableName)
    VALUES (
        v_user,
        ORA_SYSEVENT,
        SYSTIMESTAMP,
        v_status,
        v_table
    );

END AFTER STATEMENT;

END trg_compound_dml_blocker;
/
```

TESTS

TEST 1: Try a blocked operation (INSERT)

SYNTAX

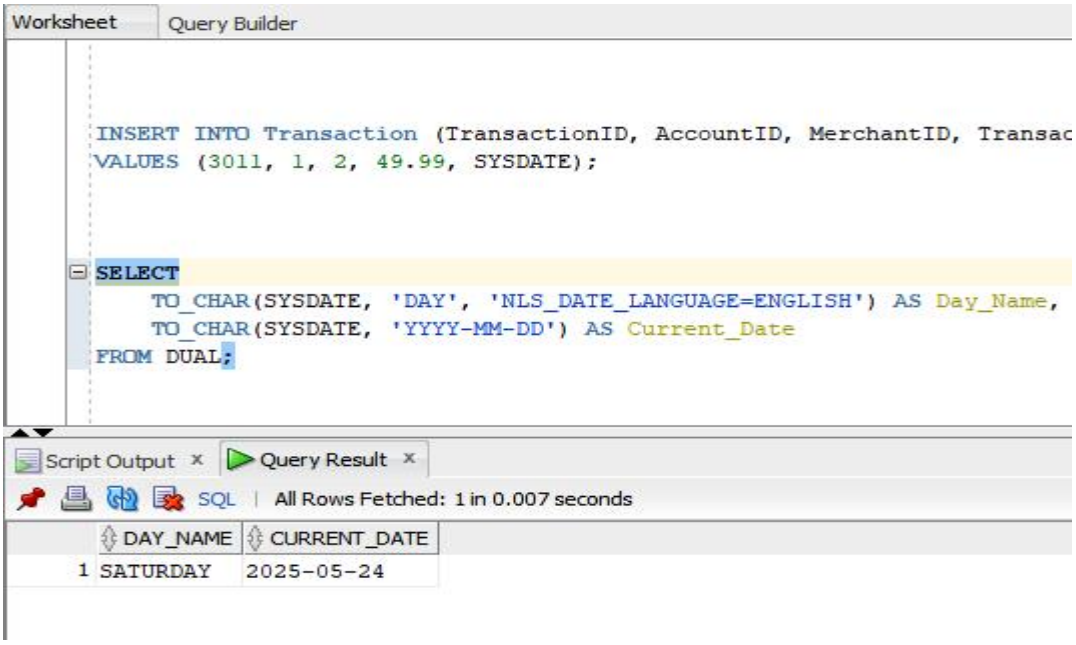
INSERT INTO Transaction (TransactionID, AccountID, MerchantID, TransactionAmount, TransactionDate)
VALUES (3011, 1, 2, 49.99, SYSDATE);



RESULT : THE row was inserted because today (24/05/2025) is a weekend not a week day.

TEST 2 : CHECKING THE CURRENT DAY OF THE WEEK

SELECT
  TO_CHAR(SYSDATE, 'DAY', 'NLS_DATE_LANGUAGE=ENGLISH') AS Day_Name,
  TO_CHAR(SYSDATE, 'YYYY-MM-DD') AS Current_Date
FROM DUAL;

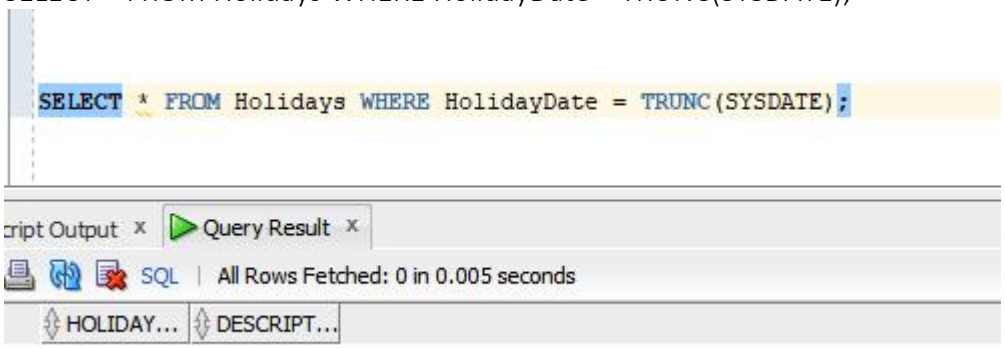


TEST 3 : REFERING TO THE HOLIDAY TABLE

Check If Today Is Already in the Holidays Table

SYNTAX

SELECT * FROM Holidays WHERE HolidayDate = TRUNC(SYSDATE);



Auditing with Restrictions and Tracking

STEP 1: Create an Audit Table

Purpose: This table will store details about every DML attempt — whether allowed or blocked.

```
CREATE TABLE Audit_Log (
   AuditID      NUMBER GENERATED BY DEFAULT ON NULL AS IDENTITY PRIMARY KEY,
   UserID       VARCHAR2(50),      -- Who attempted it
   Action       VARCHAR2(20),      -- INSERT, UPDATE, DELETE
   ActionTime   TIMESTAMP,         -- When it happened
   Status       VARCHAR2(10),      -- ALLOWED or DENIED
   TableName    VARCHAR2(50)       -- Which table was targeted
);
```

STEP 2: CREATE THE AUDIT_PKG PACKAGE

Goal: Provide a reusable, centralized procedure to log user actions.

```
CREATE OR REPLACE PACKAGE Audit_Pkg AS
   PROCEDURE Log_Audit(
      p_user    VARCHAR2,
      p_action  VARCHAR2,
      p_status  VARCHAR2,
      p_table   VARCHAR2
   );

   FUNCTION Is_Authorized(p_user VARCHAR2) RETURN BOOLEAN;
END Audit_Pkg;
/
```

Package Body

```
CREATE OR REPLACE PACKAGE BODY Audit_Pkg AS

   PROCEDURE Log_Audit(
      p_user    VARCHAR2,
      p_action  VARCHAR2,
      p_status  VARCHAR2,
      p_table   VARCHAR2
   ) IS
   BEGIN
      INSERT INTO Audit_Log (UserID, Action, ActionTime, Status, TableName)
      VALUES (p_user, p_action, SYSTIMESTAMP, p_status, p_table);
   END;

   FUNCTION Is_Authorized(p_user VARCHAR2) RETURN BOOLEAN IS
   BEGIN
      RETURN UPPER(p_user) IN ('SYS', 'ADMIN');
   END;

END Audit_Pkg;
/
```

Using Triggers to block unauthorized access or manipulation and Functions and packages to automate audit tracking

SYNTAX

```
CREATE OR REPLACE TRIGGER trg_secure_transaction_dml
FOR INSERT OR UPDATE OR DELETE ON Transaction
COMPOUND TRIGGER

   -- Shared variables
   v_day      VARCHAR2(10);
   v_holiday   INT;
   v_status    VARCHAR2(10) := 'ALLOWED';
   v_user      VARCHAR2(50);
   v_table     VARCHAR2(50) := 'Transaction';
   v_block     BOOLEAN := FALSE;

BEFORE STATEMENT IS
BEGIN
   v_user := SYS_CONTEXT('USERENV', 'SESSION_USER');

   v_day := TO_CHAR(SYSDATE, 'DY', 'NLS_DATE_LANGUAGE=ENGLISH');
```

```
   SELECT COUNT(*) INTO v_holiday
   FROM Holidays
   WHERE HolidayDate = TRUNC(SYSDATE)
     AND HolidayDate BETWEEN TRUNC(SYSDATE) AND TRUNC(ADD_MONTHS(SYSDATE, 1));

   IF v_day IN ('MON', 'TUE', 'WED', 'THU', 'FRI') OR v_holiday > 0 THEN
      v_block := TRUE;
      v_status := 'DENIED';
   END IF;
END BEFORE STATEMENT;

BEFORE EACH ROW IS
BEGIN
   IF NOT Audit_Pkg.Is_Authorized(v_user) THEN
      v_status := 'DENIED';
      RAISE_APPLICATION_ERROR(-20003, '❌ Access Denied: Unauthorized user.');
   END IF;

   IF v_block THEN
      RAISE_APPLICATION_ERROR(-20002, '❌ DML blocked on weekday or holiday.');
   END IF;
END BEFORE EACH ROW;

AFTER STATEMENT IS
BEGIN
   Audit_Pkg.Log_Audit(
      p_user   => v_user,
      p_action => ORA_SYSEVENT,
      p_status => v_status,
      p_table  => v_table
   );
END AFTER STATEMENT;

END;
/
```

TESTING

TEST : CAN THE TRIGGER BLOCK UNAUTHORIZED USERS FROM MANIPULATING DATA?

SYNTAX EXECUTED

```
INSERT INTO Transaction (TransactionID, AccountID, MerchantID, TransactionAmount,
TransactionDate)
VALUES (6001, 1, 1, 250.00, SYSDATE);
```

OUTPUT

```
INSERT INTO Transaction (TransactionID, AccountID, MerchantID, TransactionAmount, TransactionDate)
VALUES (6001, 1, 1, 250.00, SYSDATE);
```

Script Output | Query Result

Task completed in 0.294 seconds

```
Error at Command Line : 374 Column : 13
Error report -
SQL Error: ORA-20003: ✕ Access Denied: Unauthorized user.
ORA-06512: at "PALM_PAYMENT.TRG_SECURE_TRANSACTION_DML", line 32
ORA-04088: error during execution of trigger 'PALM_PAYMENT.TRG_SECURE_TRANSACTION_DML'
```

AUDIT_LOG RESULTS

SELECT * FROM Audit_Log ORDER BY ActionTime DESC;

```
SELECT * FROM Audit_Log ORDER BY ActionTime DESC;
```

Script Output | Query Result

SQL | All Rows Fetched: 1 in 0.007 seconds

	AUDITID	USERID	ACTION	ACTIONTIME	STATUS	TABLENAME
1	1	PALM_PAYMENT	(null)	25-MAY-25 12.52.23.304000000 AM	DENIED	Transaction