Task 7

Task 7: Triggers, Views, and Exceptions

Objective:

To understand and implement **Triggers**, **Views**, and **Exception Handling** for performing and managing **CRUD (Create, Read, Update, Delete)** operations in an Oracle database.

Part 1: Implementing Triggers

1. Prevent Insertion of Underage Students

```
CREATE TABLE Students (
    StudentID NUMBER PRIMARY KEY,
    Name VARCHAR2(50),
    Age NUMBER,
    Department VARCHAR2(50),
    Marks NUMBER
);
```

```
CREATE OR REPLACE TRIGGER prevent_underage_students
BEFORE INSERT ON Students
FOR EACH ROW
BEGIN
    IF :NEW.Age < 18 THEN
        RAISE_APPLICATION_ERROR(-20001, 'Age must be 18 or above');
    END IF;
END;
/
```

2. Create a Log Table

```
CREATE TABLE StudentLog (
    LogID NUMBER PRIMARY KEY,
    StudentID NUMBER,
    ActionType VARCHAR2(20),
    ActionDate TIMESTAMP DEFAULT SYSTIMESTAMP
);
```

Part 2: Creating Views

1. View for Top Students

```
CREATE OR REPLACE VIEW View_TopStudents AS
SELECT StudentID, Name, Marks
FROM Students
WHERE Marks > 80;
```

2. View for Department Summary

```
CREATE OR REPLACE VIEW View_DepartmentSummary AS
SELECT Department, COUNT(*) AS TotalStudents, ROUND(AVG(Marks),2) AS AverageMarks
FROM Students
GROUP BY Department;
```

Part 3: Exception Handling

1. Stored Procedure with Exception Handling for Inserting Student Records

```
CREATE OR REPLACE PROCEDURE InsertStudent (
    p_StudentID IN NUMBER,
    p_Name IN VARCHAR2,
    p_Age IN NUMBER,
    p_Department IN VARCHAR2,
    p_Marks IN NUMBER
)
IS
BEGIN
    INSERT INTO Students (StudentID, Name, Age, Department, Marks)
    VALUES (p_StudentID, p_Name, p_Age, p_Department, p_Marks);
    DBMS_OUTPUT.PUT_LINE('Record Inserted Successfully');
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Error: ' || SQLERRM);
END;
/
```

2. Function to Fetch Student Details with Error Handling

```sql
CREATE OR REPLACE FUNCTION GetStudentDetails (p_StudentID IN NUMBER)
RETURN VARCHAR2
IS
    student_info VARCHAR2(255);
BEGIN
    SELECT 'Name: ' || Name || ', Age: ' || Age || ', Department: ' || Department || ', Marks: ' || Marks
    INTO student_info
    FROM Students
    WHERE StudentID = p_StudentID;
    RETURN student_info;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RETURN 'Student Not Found';
    WHEN OTHERS THEN
        RETURN 'Error: ' || SQLERRM;
END;
/
```