**TASK 7: Triggers, Views, and Exception Handling**

**AIM**: To understand and implement Triggers, Views, and Exception Handling for managing CRUD operations in an Oracle database**.**

**Part 0: Setup (Base Tables)**

**Before starting, create base tables to work with.**

DROP TABLE students CASCADE CONSTRAINTS;

DROP TABLE departments CASCADE CONSTRAINTS;

-- Drop tables if they already exist

DROP TABLE student\_log CASCADE CONSTRAINTS;

-- Create departments table

CREATE TABLE departments (

dept\_id NUMBER PRIMARY KEY,

dept\_name VARCHAR2(50),

hod\_name VARCHAR2(50)

);

-- Create students table

CREATE TABLE students (

student\_id NUMBER PRIMARY KEY,

student\_name VARCHAR2(50),

age NUMBER,

dept\_id NUMBER,

marks NUMBER(5,2),

CONSTRAINT fk\_dept FOREIGN KEY (dept\_id) REFERENCES departments(dept\_id)

);

**Part 1: Implementing Triggers**

**1. Prevent Insertion of Underage Students**

Create a trigger that prevents inserting students under 18 years old.

CREATE OR REPLACE TRIGGER trg\_prevent\_underage\_students

BEFORE INSERT ON students

FOR EACH ROW

BEGIN

IF :NEW.age < 18 THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Student must be at least 18 years old.');

END IF;

END;

/

✅ Test:

INSERT INTO students VALUES (1, 'Alex', 17, NULL, 85);

-- Should raise error: Student must be at least 18 years old

**2. Create a Log Table**

**This table and trigger will log insert, update, and delete operations on the students table.**

-- Log table

CREATE TABLE student\_log (

log\_id NUMBER GENERATED ALWAYS AS IDENTITY,

operation VARCHAR2(10),

student\_id NUMBER,

log\_date DATE,

old\_marks NUMBER(5,2),

new\_marks NUMBER(5,2)

);

-- Trigger for logging

CREATE OR REPLACE TRIGGER trg\_student\_log

AFTER INSERT OR UPDATE OR DELETE ON students

FOR EACH ROW

BEGIN

IF INSERTING THEN

INSERT INTO student\_log (operation, student\_id, log\_date)

VALUES ('INSERT', :NEW.student\_id, SYSDATE);

ELSIF UPDATING THEN

INSERT INTO student\_log (operation, student\_id, log\_date, old\_marks, new\_marks)

VALUES ('UPDATE', :OLD.student\_id, SYSDATE, :OLD.marks, :NEW.marks);

ELSIF DELETING THEN

INSERT INTO student\_log (operation, student\_id, log\_date, old\_marks)

VALUES ('DELETE', :OLD.student\_id, SYSDATE, :OLD.marks);

END IF;

END;

**/**

**✅ Test:**

INSERT INTO students VALUES (2, 'John', 20, NULL, 88);

UPDATE students SET marks = 90 WHERE student\_id = 2;

DELETE FROM students WHERE student\_id = 2;

SELECT \* FROM student\_log**;**

**Part 2: Creating Views**

**1. View for Top Students**

Create a view showing students with marks greater than or equal to 85.

CREATE OR REPLACE VIEW vw\_top\_students AS

SELECT student\_id, student\_name, marks

FROM students

WHERE marks >= 85;

✅ Query:

SELECT \* FROM vw\_top\_students**;**

**2. View for Department Summary**

Show department name, total students, and average marks.

CREATE OR REPLACE VIEW vw\_department\_summary AS

SELECT d.dept\_name,

COUNT(s.student\_id) AS total\_students,

AVG(s.marks) AS avg\_marks

FROM departments d

LEFT JOIN students s ON d.dept\_id = s.dept\_id

GROUP BY d.dept\_name**;**

**✅ Query:**

SELECT \* FROM vw\_department\_summary**;**

**Part 3: Exception Handling**

**1. Stored Procedure with Exception Handling for Inserting Student Records**

Create a procedure that inserts a student and handles possible errors (like foreign key or age violation).

CREATE OR REPLACE PROCEDURE add\_student(

p\_id NUMBER,

p\_name VARCHAR2,

p\_age NUMBER,

p\_dept NUMBER,

p\_marks NUMBER

)

IS

BEGIN

INSERT INTO students (student\_id, student\_name, age, dept\_id, marks)

VALUES (p\_id, p\_name, p\_age, p\_dept, p\_marks);

DBMS\_OUTPUT.PUT\_LINE('Student inserted successfully.');

EXCEPTION

WHEN DUP\_VAL\_ON\_INDEX THEN

DBMS\_OUTPUT.PUT\_LINE('Error: Student ID already exists.');

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Error: ' || SQLERRM);

END;

**/**

**✅ Test:**

EXEC add\_student(3, 'Maria', 19, 1, 92);

EXEC add\_student(3, 'Duplicate', 22, 1, 80); -- should raise duplicate ID error

**2. Function to Fetch Student Details with Error Handling**

**Return student name and marks based on student ID.**

CREATE OR REPLACE FUNCTION get\_student\_details(p\_id NUMBER)

RETURN VARCHAR2

IS

v\_name students.student\_name%TYPE;

v\_marks students.marks%TYPE;

BEGIN

SELECT student\_name, marks INTO v\_name, v\_marks

FROM students

WHERE student\_id = p\_id;

RETURN 'Name: ' || v\_name || ', Marks: ' || v\_marks;

**EXCEPTION**

WHEN NO\_DATA\_FOUND THEN

RETURN 'Error: Student not found.';

WHEN OTHERS THEN

RETURN 'Error: ' || SQLERRM;

END;

/

**✅ Test:**

SELECT get\_student\_details(3) FROM dual;

SELECT get\_student\_details(99) FROM dual; -- should return error message

**✅ Summary of Deliverables**

| **Part** | **Task** | **Object Created** |
| --- | --- | --- |
| 1.1 | Prevent underage students | TRIGGER trg\_prevent\_underage\_students |
| 1.2 | Log operations | TABLE student\_log, TRIGGER trg\_student\_log |
| 2.1 | Top students view | VIEW vw\_top\_students |
| 2.2 | Department summary view | VIEW vw\_department\_summary |
| 3.1 | Procedure with exception handling | PROCEDURE add\_student |
| 3.2 | Function with error handling | FUNCTION get\_student\_details |

**RESULT:**

To understand and implement Triggers, Views, and Exception Handling for managing CRUD operations in an Oracle database.