Program 1

Write a code in PL/SQL to develop a trigger that enforces referential integrity by preventing the deletion of a parent record if child records exist.

```
CREATE TABLE dept( deptno NUMBER PRIMARY KEY,
d_name VARCHAR2(20));

CREATE TABLE emp (
        empno   NUMBER      PRIMARY KEY,
        ename   VARCHAR2(20),
        deptno  NUMBER      REFERENCES
   dept(deptno)
  );

CREATE OR REPLACE TRIGGER
Prevent_delete
BEFORE DELETE ON dept
FOR EACH ROW
DECLARE
        v_Count NUMBER;
 BEGIN
        SELECT COUNT (*) INTO v_Count FROM
  emp   WHERE deptno = :OLD.deptno;
        If v_Count>0 THEN
              RAISE_APPLICATION_ERROR (-20001,
 'child records exist. Cannot delete parent.');
  END IF;
END;
```

Program 2

Write a code in PL/SQL to create a trigger that checks for duplicate values in a specific column and raises an exception if found.

```
CREATE OR REPLACE TRIGGER
check duplicate_id
BEFORE INSERT OR UPDATE ON employee
FOR EACH ROW
DECLARE
        v_Count NUMBER;
BEGIN
        SELECT COUNT(*) INTO v_Count
        FROM employee
        WHERE emp_id = : NEW.emp_id

        If v_Count >0 THEN
RAISE_APPLICATION_ERROR (-20001, 'Duplicate emp id
found !');
        END If ;
END;
/
```

Program 3

Write a code in PL/SQL to create a trigger that restricts the insertion of new rows if the total of a column's values exceeds a certain threshold.

```
CREATE OR REPLACE TRIGGER
check_total_salary
 BEFORE INSERT ON employee
 FOR EACH ROW
 DECLARE
        v_total NUMBER;
 BEGIN
        SELECT SUM(Salary) INTO v_total
 FROM employee;
        IF v_total + :NEW.Salary > 100000
 THEN
 RAISE_APPLICATION_ERROR (-20002, 'Total Salary limit
     exceeded:');
          END IF;
 END;
 /
```

Program 4

Write a code in PL/SQL to design a trigger that captures changes made to specific columns and logs them in an audit table.

```
CREATE  TABLE   emp_audit (
        emp_id        NUMBER,
        old_Salary    NUMBER,
        new_Salary    NUMBER,
        changed_on    DATE
    );

    CREATE   OR   REPLACE   TRIGGER
    log_Salary_Changes
    AFTER   UPDATE   OF   Salary   ON   employee
    FOR   EACH   ROW
    BEGIN
            INSERT   INTO   emp_audit (emp_id, old_Salary,
    new_Salary,  changed_on)
            VALUES
    ( :OLD. emp_id,  :OLD. Salary,  :NEW. Salary,
      SYSDATE );

    END;
    /
```

14/4

Program 5

Write a code in PL/SQL to implement a trigger that records user activity (inserts, updates, deletes) in an audit log for a given set of tables.

```
CREATE TABLE ~~as~~ audit_log (
    username     VARCHAR2(30),
    operation    VARCHAR2(10),
    table_name   VARCHAR2(30),
    action_time  DATE );


CREATE   OR   REPLACE   TRIGGER
record_user_activity
AFTER  INSERT  OR  UPDATE  OR  DELETE  ON
employee
 BEGIN
      INSERT   INTO   audit_log (username,  operation,
      table_name,   action_time)
      VALUES (USER,   ORA_SYSEVENT,
      'EMPLOYEE' ,  SYSDATE);

   END;
   /
```

Program 7

Write a code in PL/SQL to implement a trigger that automatically calculates and updates a running total column for a table whenever new rows are inserted.

```
CREATE   TABLE   Sales(
        Sale_id     NUMBER,
        amount      NUMBER
        running-total   NUMBER);

CREATE   OR   REPLACE   TRIGGER
update_running - total
AFTER   INSERT   ON   Sales
FOR   EACH   ROW
BEGIN
        UPDATE   Sales
        SET      running-total  =(SELECT  SUM(amount)
                 FROM  Sales);
    END;
```

Program 8

Write a code in PL/SQL to create a trigger that validates the availability of items before allowing an order to be placed, considering stock levels and pending orders.

```
CREATE OR REPLACE TRIGGER
trg_check_item_availability
BEFORE INSERT ON orders
FOR EACH ROW
DECLARE
        v_availability_qty NUMBER;
        v_pending_qty NUMBER;
BEGIN
        SELECT stock_qty
        INTO v_available_qty
        FROM items
        WHERE items_id = :NEW.item_id;

        SELECT NVL (SUM (quantity), 0)
        INTO v_pending_qty
        FROM orders
        WHERE items_id = :NEW.item_id
            AND status = 'PENDING';
    IF (:NEW.quantity + v_pending_qty > v_available_qty THEN
RAISE_APPLICATION_ERROR (-20001, 'Insufficient stock available
for item ID: ' || :NEW.item_id || '. Available: ' ||
    (v_available_qty - v_pending_qty)); END IF; END;
```

Evaluation Procedure	Marks awarded
PL/SQL Procedure(5)	5
Program/Execution (5)	5
Viva(5)	5
Total (15)	15
Faculty Signature	[illegible]