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 OR REPLACE TRIGGER
trg_prevent_parent_delete
BEFORE DELETE ON department
FOR EACH ROW
DECLARE
   v_count NUMBER;
BEGIN
   SELECT COUNT(*) INTO v_count FROM employee WHERE
   dept_id = :OLD.dept_id;
   IF v_count > 0 THEN
    RAISE_APPLICATION_ERROR (-20001, 'Cannot delete Parent Record.
Child records exists in EMPLOYEE table.');

    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
trg_check_duplicate_email
BEFORE INSERT OR UPDATE ON students
FOR EACH ROW
DECLARE
    v_count NUMBER;
BEGIN
    SELECT COUNT(*) INTO v_count FROM students
WHERE email = :NEW.email;
    IF v_count >0 THEN
    RAISE_APPLICATION_ERROR (-20002, 'Duplicate email detected.
    Each email must be uniqe.');
    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
trg_limit_total_salary
BEFORE INSERT ON employee
FOR EACH ROW
DECLARE
   v_total NUMBER;
   v_threshold CONSTANT NUMBER := 100000;
BEGIN
   SELECT NVL(SUM(salary), 0) INTO v_total FROM employee;
  IF (v_total + :NEW.salary) > v_threshold THEN
     RAISE_APPLICATION_ERROR(-20003, 'Total salary exceeds the
  allowed threshold.');
   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 employee_audit (
  emp_id NUMBER,
  old_salary NUMBER,
  new_salary NUMBER,
  Change_date DATE,
  Changed_by VARCHAR2(30));

CREATE OR REPLACE TRIGGER
trg_audit_salary_change
AFTER UPDATE OF salary ON employee
FOR EACH ROW
BEGIN
  INSERT INTO employee_audit (emp_id, old_salary, new_salary
      Change_date, changed_by)
  VALUES (:OLD.emp_id, :OLD.salary, :NEW.salary,
  SYSDATE, USER);
END;
```

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 activity_log (
    table_name VARCHAR2(50),
    operation_type VARCHAR2(20),
    user_name VARCHAR2(30),
    activity_date DATE
);

CREATE OR REPLACE TRIGGER trg_user activity
AFTER INSERT OR UPDATE OR DELETE ON employee
BEGIN
    INSERT INTO activity_log (table_name, operation_type, user_name,
    activity_date)
        VALUES ('EMPLOYEE', ORA_SYSEVENT, USER, 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
 trg_update_running_total
 AFTER INSERT ON sales
 FOR EACH ROW
 DECLARE
    v_total NUMBER;
 BEGIN
   SELECT NVL (SUM (amount), 0) INTO v_total FROM sales ;

   UPDATE sales SET running_total = v_total WHERE
       sale_id = :NEW.sale_id;
 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_stock_availability
BEFORE INSERT ON orders
FOR EACH ROW
DECLARE
 v_stock NUMBER;
BEGIN
    SELECT quantity_in_stock INTO v_stock FROM inventory
WHERE item_id = :NEW.item_id;
    IF v_stock < :NEW.order_quantity THEN
    RAISE_APPLICATION_ERROR (-20004, 'Insufficient stock
available for the requested item.');
    END IF;
END;
```

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