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_table

BEFORE DELETE ON dept_parent

 FOR EACH ROW

DECLARE

     v_count_number;

BEGIN,

     SELECT COUNT(*) INTO COUNT FROM emp_child WHERE

     dept_id = :OLD.dept.id;

  IF v_count > 0 then

     RAISE_APPLICATION_ERROR (-20001,'Cannot delete departmen
                                  - employees exist.');

  END IF;

END;

     DELETE FROM dept_parent WHERE dept_id = 10;
```

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_title
Before Insert or update on books
Declare
  Select count(*) into v_count from books.
  where title = New.title;
  if v_count > 0 then
Raise_Application_Error (-20002, 'Duplicate title not
                                  allowed :);
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 table products(
        product_id number PRIMARY key
      quantity. NUMBER
);
 CREATE or replace trigger trg_restrict_insert
before insert on products
 for each row
 Declare
     v_total_quantity number;
     v_threshold CONSTANT NUMBER := 1000;
Begin
    SELECT SUM(quantity) INTO v_total_quantity from produ
If (v_total_quantity + :New:quantity)> v_threshold Then
  Raise_application_error(-20001, 'Insertion restricted:
                                  Total quantity exceeds
                                  the threshold.');
END IF;
 END.
```


Shivakumar

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 employees (
        employee_id number primary key,
    f_name varchar2(50);
    l_name varchar2(50);
    Salary number(10,2);
);
    create table employee_audit (
        audit_id number generated by default as identity(
        employee_id number,
        old_salary number(10,2)
        change_date TIMESTAMP

BEGIN
    insert into employee_audit ( employee_id, odd_salary,
                new_salary, change_date)
    values ( :OD.employee_id, :old.salary, :New.salary,
            system TIMESTAMP)

    END;

    INSERT into employee (employee_id, f.name, l.name)
        values (101, 'John', 'Doe', 50000);
```

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 audit log (
        log_id number generated by default as identity
    user_name varchar2 (100),
    action_date Timestamp ,
    action_type varchar2(100) ;
    Primary key (log ID) .

);
  Begin
    If inserting then
        insert into audit_log (user_name, action_date,
                               action type)
      values (User, SYS TIMESTAMP - 'Insert', 'employees');
  Else if updating then
      insert into audit_log (user_name, action_date, action
                                                    type)
      values (User, systimestamp , 'UPDATE', `employees ');
END If;
  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_records (
        sales_id number primary key,
        sales_amount number (10,2),
        running_total number (10,2)
);
create or update trigger, update_running_total
    before insert or update record
  for each row
    Declare
        v_last_total number (10,2);
  Begin
      select NVL (max (running_total), 0)
      into v_last_total

    New.running_total := v_last_total + New.sales_amount;

END;
insert into sales_records (sale_id, sales, amount);
```

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 table orders (
        order_id number primary key,
        quantity number);

create table items (
        item_id number primary key
        item_name varchar2 (100);
        stock_level number);

create or replace trigger check stock_availability.
before insert on orders
  for each row
  Declare
      v_stock_level number;

Begin
      select stock_level into v_stock_level
      from items

IF : New_quantity > v_stock_level then
      Raise application_error (-20001, 'Insufficient'
                                stock for item'||
                                :New item'||':');
END;
```

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