Program 1

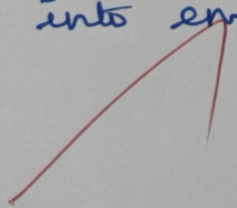
Write a code in PL/SQL to develop a trigger that enforces referential integrity by preventing the deletion of a parent record if child records exist.

```
create table departments (
        department_id NUMBER PRIMARY KEY,
        department_name VARCHAR2(100)
);
Create table employees(
        employee_id NUMBER PRIMARY KEY,
        employee_name VARCHAR2(100),
        department_id NUMBER,
        CONSTRAINT fk_dept FOREIGN KEY (department_id) REFERENCES
                                        departments(department_
);
create or replace trigger prevent_department_deletion
before delete on departments
FOR EACH ROW
Declare
    child_count NUMBER;
BEGIN
    Select count(*) into child_count
    from employees
    where department_id =:OLD.department_id;
    if child_count>0 Then
        Raise_application_error(-20001,'Cannot delete department
                                        with existing employees.');
        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 table employees(
    employee_id NUMBER PRIMARY KEY,
    employee_name VARCHAR2(100)
);
create or replace trigger check_duplicate_employee_id
before insert on employees
for each row
Declare
    v_count NUMBER;
Begin
    Select COUNT(*)
    INTO v_count
    FROM employees
    WHERE employee_id = :NEW.employee_id;
    IF v_count > 0 THEN
        Raise_application_error (-20001, 'Duplicate employee id
                                  found. Cannot insert.');
    END IF;
END;
Insert into employees (employeeid, employee_name) values
                                  (1, 'John Doe');
```



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 try_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 products;
    If (v_total_quantity + :NEW.quantity) > v_threshold THEN
        Raise_application_error (-20001, 'Insertion restricted:
                                    Total quantity exceeds
                                    the 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 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),
    new_salary number(10,2),
    change_date TIMESTAMP
);
create or replace trigger audit_employee_salary
after update of salary ON employees
FOR each ROW
When(OLD.salary IS DISTINCT FROM NEW.salary)
Begin
    insert into employee_audit(employee_id, old_salary,
                new_salary, change_date)
    values(:OLD.employee_id, :OLD.salary, :New.salary,
                - SYSTIMESTAMP);
END;
Insert into employees(employee_id, f_name, l_name,
                salary)
    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(10),
        table_name varchar2(100),
        PRIMARY KEY (log_id)
    );

create or replace trigger try_employees_audit
after insert or update or delete on employees
for each row
Begin
    If inserting then
        insert into audit_log(user_name, action_date,
                              action_type, table_name)
        values (USER, SYSTIMESTAMP, 'INSERT', 'employees');
    Elseif updating then
        insert into audit_log(user_name, action_date, action_type,
                              table_name)
        values (USER, SYSTIMESTAMP, 'UPDATE', 'employees');
    Elseif deleting then
        insert into audit_log(user_name, action_date, action_type,
                              table_name)
        values (USER, SYSTIMESTAMP, 'DELETE', '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,
        sale_amount number (10,2),
        running_total number (10,2)
);
create or replace trigger update_running_total
before insert on sales_records
for each row
Declare
    v_last_total number(10,2);
Begin
    select NVL(max(running_total),0)
    into v_last_total
    from sales_records;
    :New.running_total := v_last_total + :NEW.sale_amount;
END;
Insert into sales_records(sale_id, sales amount) values (1,100);
```

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 primarykey,
    item_id NUMBER,
    quantity NUMBER );
create table items (
    item_id number primarykey,
    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
    where item_id = :NEW.item_id;
    IF :NEW.quantity > v_stock_level THEN
        Raise application_error(-20001,'Insufficient
                                stock for item' ||
                                :NEW.item_id ||'.');
    END IF;
END;
```

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

20D