

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 prevent-parent-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, 'Record
not exist')

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-rollno

Before insert or update on Student

for each row

Declare

v_count number;

Begin

Select Count(*) into v_count from Student where
rollno = : New.rollno;

if v_count > 1 then

raise Application Error (-20002, 'Duplicate Roll
Number 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 account (  
    ac-no NUMBER PRIMARY KEY,  
    cust-name VARCHAR2(30),  
    balance NUMBER  
);
```

Create or replace Trigger

check-total-balance

Before INSERT OR UPDATE ON account
for each row

Declare

```
v-total NUMBER;  
v-threshold CONSTANT NUMBER := 100000;
```

Begin

```
Select NVL(SUM(balance), 0) INTO v-total FROM account;
```

```
v-total := v-total + :NEW.balance;
```

```
IF v-total > v-threshold THEN
```

```
    RAISE_APPLICATION_ERROR(-20003, 'Total balance  
exceeds allowed limit of 100000');
```

```
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(  
    emp-id NUMBER PRIMARY KEY,  
    emp-name VARCHAR2(30),  
    Salary NUMBER);
```

```
Create TABLE audit-log(  
    log-id NUMBER GENERATED ALWAYS AS IDENTITY  
    PRIMARY KEY,  
    emp-id NUMBER,  
    old-Salary NUMBER,  
    new-Salary NUMBER,  
    changed-on DATE,  
    changed-by VARCHAR2(30) );
```

Create an replace trigger Salary.audit-trigger After
update OF Salary on employee
for each row

```
Begin  
    INSERT INTO audit-log 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 or replace trigger emp-audit-trigger After insert
or update or delete on employee

```
BEGIN
    INSERT INTO auditlog values ('EMPLOYEE', ORA_SYSEVENT,
USER, SYSDATE );
END;
/
```

Create or replace trigger dept-audit-trigger
After insert or update or delete on department

```
BEGIN
    INSERT INTO audit-log values ('DEPARTMENT',
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 PRIMARY KEY,
 Sale-amount NUMBER,
 running-total NUMBER);

Create or replace trigger update-running-total before

INSERT ON Sales

for each row

declare

 v-total NUMBER;

BEGIN

 select NVL (Sum (Sale-amount), 0) INTO v-total from
 Sales;

 :NEW.running-total := v-total + :NEW.Sale-amount;

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 Validate - Stock - Before - order
before INSERT ON orders for each row

declare

v_available - qty NUMBER;

BEGIN

Select stock_qty into v_available_qty from items
where item_id = :New.item_id;

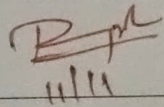
IF :New.order_qty > v_available_qty ~~from items~~ then

~~RAISE~~ RAISE - APPLICATION - ERROR (-20005, 'Insufficient Stock!
Cannot place order.1');

END IF;

UPDATE items SET Stock_qty = Stock_qty - :New.order_qty
where item_id = :New.item_id;

END;

Evaluation Procedure	Marks awarded
PL/SQL Procedure(5)	5
Program/Execution (5)	5
Viva(5)	5
Total (15)	15
Faculty Signature	 11/11