Ex.No.: 13		WORKING WITH TRIGGER **TRIGGER**
Date:		

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_deletion
BEFORE DELETE ON departments
FOR EACH ROW
DECLARE
   v_count NUMBER;
BEGIN
   -- Check if there are any associated child records in 'employees'
   SELECT COUNT(*) INTO v_count FROM employees WHERE department_id =
:OLD.department_id;

   -- If child records exist, raise an error
   IF v_count > 0 THEN
      RAISE_APPLICATION_ERROR(-20001, 'Cannot delete department with associated 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 products (
product_id NUMBER PRIMARY KEY,
product_nameVARCHAR2(50)
);
-- Create a trigger to check for duplicate values
CREATE OR REPLACE TRIGGER prevent_duplicates
BEFORE INSERT ON products
FOR EACH ROW
DECLARE
v_count NUMBER;
BEGIN
   -- Check if the new product_name already exists
   SELECT COUNT(*) INTO v_count FROM products WHERE product_name = :NEW.product_name;
      -- If duplicate value found, raise an error
   IF v_count> 0 THEN
      RAISE_APPLICATION_ERROR(-20001, 'Product name already exists.');
   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 orders (
    order_id NUMBER PRIMARY KEY,
    customer_id NUMBER,
    order_amount NUMBER
);

-- Create a trigger to restrict the total order amount for a customer
CREATE OR REPLACE TRIGGER check_order_amount
BEFORE INSERT ON orders
FOR EACH ROW
DECLARE
    total_amount NUMBER;
    max_threshold NUMBER := 10000; -- Change this to your desired threshold
BEGIN
    -- Calculate the current total order amount for the customer
    SELECT NVL(SUM(order_amount), 0) INTO total_amount
    FROM orders
    WHERE customer_id = :NEW.customer_id;

    -- Check if inserting the new row will exceed the threshold
    IF total_amount + :NEW.order_amount > max_threshold THEN
        RAISE_APPLICATION_ERROR(-20001, 'Total order amount 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 OR REPLACE TRIGGER salary_change_audit
AFTER UPDATE ON employees
FOR EACH ROW
WHEN (NEW.salary<>OLD.salary) -- Only capture changes in the salary column
DECLARE
v_audit_id NUMBER;
BEGIN
    -- Generate a unique audit ID
    SELECT seq_salary_audit.NEXTVAL INTO v_audit_id FROM DUAL;

    -- Insert the change details into the audit table
    INSERT INTO salary_audit (audit_id, employee_id, old_salary, new_salary,
change_date)
    VALUES (v_audit_id, :OLD.employee_id, :OLD.salary, :NEW.salary,
SYSTIMESTAMP);
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 Employee table
CREATE TABLE Employee (
emp_id NUMBER PRIMARY KEY,
emp_nameVARCHAR2(100),
emp_salary NUMBER
);
-- Create Audit_Log table
CREATE TABLE Audit_Log (
log_id NUMBER PRIMARY KEY,
table_nameVARCHAR2(100),
activity_typeVARCHAR2(20),
activity_date TIMESTAMP,
user_idVARCHAR2(50)
);
CREATE SEQUENCE Audit_Log_Seq START WITH 1 INCREMENT BY 1;
CREATE OR REPLACE TRIGGER Employee_Audit_Trigger
AFTER INSERT OR UPDATE OR DELETE ON Employee
FOR EACH ROW
DECLARE
v_activity_typeVARCHAR2(20);
BEGIN
   IF INSERTING THEN
v_activity_type := 'INSERT';
   ELSIF UPDATING THEN
v_activity_type := 'UPDATE';
   ELSIF DELETING THEN
v_activity_type := 'DELETE';
   END IF;
   INSERT INTO Audit_Log (log_id, table_name, activity_type, activity_date, user_id)
   VALUES (Audit_Log_Seq.NEXTVAL, 'Employee', v_activity_type, SYSTIMESTAMP, USER);
END;
-- Insert a new employee
INSERT INTO Employee (emp_id, emp_name, emp_salary)
VALUES (1, 'John Doe', 50000);
-- Update an employee's salary
UPDATE Employee SET emp_salary = 55000 WHERE emp_id = 1;
-- Delete an employee
DELETE FROM Employee WHERE emp_id = 1;
SELECT * FROM Audit_Log;
```

Program 6

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 Sales table
CREATE TABLE Sales (
sale_id NUMBER PRIMARY KEY,
sale_date DATE,
amount NUMBER,
running_total NUMBER
);

-- Create Trigger
CREATE OR REPLACE TRIGGER Update_Running_Total
BEFORE INSERT ON Sales
FOR EACH ROW
BEGIN
IF :NEW.running_total IS NULL THEN
      SELECT NVL(MAX(running_total), 0) + :NEW.amount
INTO :NEW.running_total
      FROM Sales;
   ELSE
:NEW.running_total := :NEW.running_total + :NEW.amount;
   END IF;
END;
/
```

Program 7

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 Products table
CREATE TABLE Products (
    product_id NUMBER PRIMARY KEY,
    product_name VARCHAR2(100), -- Added space
    stock_quantity NUMBER
);



-- Create a new Orders table
CREATE TABLE Orders (
    order_id NUMBER PRIMARY KEY,
    product_id NUMBER,
    order_quantity NUMBER
);


-- Create Trigger to validate availability before placing an order
CREATE OR REPLACE TRIGGER Validate_Order_Availability
BEFORE INSERT ON Orders
FOR EACH ROW
DECLARE
    v_current_stock NUMBER;
    v_pending_orders NUMBER;
BEGIN
    -- Get current stock for the product
    SELECT stock_quantity INTO v_current_stock
    FROM Products
    WHERE product_id = :NEW.product_id;

    -- Get total quantity of pending orders for the product
    SELECT NVL(SUM(order_quantity), 0) INTO v_pending_orders
    FROM Orders
    WHERE product_id = :NEW.product_id;

    -- Calculate total available quantity (stock - pending orders)
    IF v_current_stock - v_pending_orders - :NEW.order_quantity < 0 THEN
        RAISE_APPLICATION_ERROR(-20001, 'Insufficient stock for the order');
    END IF;
END;
/
```