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 dept_parent
FOR EACH ROW
DECLARE
    v_count NUMBER;
BEGIN
    SELECT COUNT(*) INTO v_count FROM emp_child WHERE
                                    dept_id = :OLD.dept_id;
    IF v_count > 0 THEN
        RAISE_APPLICATION_ERROR (-20001, 'Cannot delete department -
                                    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 trg_check_duplicate_email
BEFORE INSERT OR UPDATE ON customers
FOR EACH ROW
DECLARE
    v_count NUMBER;
BEGIN
    SELECT COUNT(*) INTO v_count
    FROM customers
    WHERE email = :NEW.email;
    IF v_count > 0 THEN
        RAISE_APPLICATION_ERROR(-20002, 'Duplicate email detected!');
    END IF;
END;
/
INSERT INTO customers VALUES (2, 'abc@gmail.com');
```

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_salary_limit
BEFORE INSERT ON emp_budget
FOR EACH ROW
DECLARE
    v_total NUMBER;
BEGIN
    SELECT sum(salary) INTO v_total FROM emp_budget;
    IF NVL(v_total,0) + :NEW.salary >100000 THEN
        RAISE_APPLICATION_ERROR(-20003, 'Salary total exceeds 100000
                                        limit. Cannot insert.');
    END IF;
END;
/

INSERT INTO emp_budget VALUES (3, 'Mike', 40000);
```

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 trg_audit_salary_changes
AFTER UPDATE OF salary ON emp_salary
FOR EACH ROW
BEGIN
    INSERT INTO emp_audit (emp_id, old_salary, new_salary, change_date) VALUES (:OLD.emp_id, :OLD.salary, :NEW.salary, SYSDATE);
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 trg_emp_activity
AFTER INSERT OR UPDATE OR DELETE ON emp_salary
FOR EACH ROW
BEGIN
    IF INSERTING THEN
        INSERT INTO emp_activity_log (username, operation, emp_id,
        action_date) VALUES (USER, 'INSERT', :NEW.emp_id, SYSDATE);
    ELSEIF UPDATING THEN
        INSERT INTO emp_activity_log (username, operation, emp_id,
        action_date) VALUES (USER, 'UPDATE', :OLD.emp_id, SYSDATE);
    ELSEIF DELETING THEN
        INSERT INTO emp_activity_log (username, operation, emp_id
        action_date) VALUES (USER, 'DELETE', :OLD.emp_id, SYSDATE);
    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 OR REPLACE TRIGGER trg_update_total_sales
AFTER INSERT ON sales_data
FOR EACH ROW
BEGIN
    UPDATE sales_data
    SET total_sales = (SELECT sum(amount) FROM sales_data);
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_item_stock
BEFORE INSERT ON orders
FOR EACH ROW
DECLARE
    v_available NUMBER;
BEGIN
    SELECT available_qty INTO v_available
    FROM stock_items
    WHERE item_id = :NEW.item_id;
    IF v_available < :NEW.order_qty THEN
        RAISE_APPLICATION_ERROR(-20000, 'Not enough stock to
                                         fulfill this order.');
    END IF;
END;
/
```

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