```sql
drop table Account;
create table Account(id int primary key, name varchar(20), bal int);
drop trigger trigger2;

CREATE  TRIGGER trigger2
BEFORE INSERT ON Account
FOR EACH ROW
DECLARE
 min_bal integer := 0;
BEGIN
IF (:new.bal < min_bal ) THEN
    dbms_output.put_line('Error..! Balance cannot be less than zero');
    raise_application_error(-20000, 'Balance cannot be less than zero');
END IF;
END;

insert into Account values(100, 'Alice', 200);
insert into Account values(104, 'Bob', 500);

select * from Account;
insert into Account values(102, 'Cindy', -500);

/*
Test 2:
Write a PL/SQL program to create a trigger before insert
for each row and not allowing transaction on weekends.
*/

drop table Student;
create table Student(Rollno int primary key, name varchar(20), address varchar(50));
insert into Student values(104, 'Bob', 'Address1');

drop trigger trigger3;
CREATE OR REPLACE TRIGGER trigger3
BEFORE INSERT ON Student
REFERENCING NEW AS NEW OLD AS OLD
FOR EACH ROW
BEGIN
    IF TO_CHAR(SYSDATE, 'D') <> '7' THEN
        dbms_output.put_line('Error..! Cannot insert record on weekdays');
        RAISE_APPLICATION_ERROR(-20000, 'Cannot insert record on weekdays');
    END IF;
END;

insert into Student values(103, 'Alice', 'Address2');


/*
Sapple 3
    Book_avail (bookid, title, no_of _copies, price)
    Student (st_id,name,class,fine)
    Issue_tab (st_id, book_id, issuedate, returndate)

    Create a database trigger to calculate the fine based on the rules given below.
        After 1 month 5% of price
        After 2 month 10% of price
        After 3 month 20% of price.
*/
drop table Book_avail;
drop table Student;
drop table Issue_tab;
create table Book_avail(bookid int primary key, title varchar(20), no_of_copies int,
price int);
create table Student(st_id int primary key, name varchar(20), class varchar(10), fine int
);
create table Issue_tab(st_id int, book_id int, issue_date date , return_date date,
primary key(st_id, book_id));

insert into Student values(100, 'Alice', 'CSE', 0);
```

```
insert into Student values(101, 'Bob', 'CSE', 0);
insert into Book_avail values(1, 'Data Structure', 10, 1000);
insert into Book_avail values(2, 'Java - Complete ref', 10, 1000);

--TO_DATE('2003/05/03 21:02:44', 'yyyy/mm/dd hh24:mi:ss')
insert into Issue_tab values(100, 1, TO_DATE('2022/01/01','%yyyy-%mm-%dd'), TO_DATE(
'2022/02/01','yyyy-%mm-%dd'));
insert into Issue_tab values(101, 2, TO_DATE('2022/01/01','%yyyy-%mm-%dd'), TO_DATE(
'2022/03/01','yyyy-%mm-%dd'));
select * from Issue_tab;
select * from Student;
select * from Book_avail;
drop trigger trigger4;

CREATE OR REPLACE TRIGGER trigger4
BEFORE UPDATE ON Issue_tab
REFERENCING NEW AS NEW OLD AS OLD
FOR EACH ROW
DECLARE
    bprice int;
    months int;
    latefine int;
BEGIN
    select price into bprice from Book_avail where bookid=1;
    months:=months_between(:new.return_date,:old.issue_date);
    dbms_output.put_line('months = ' || months);

    -- Calculate fine while update issue_tab
    if months>=1 and months<2 then
        latefine := bprice*0.05;
    else if months>=2 and months<3 then
        latefine := bprice*0.01;
    else if months>=3 then
        latefine := bprice*0.2;
    end if;
    end if;
    end if;

    -- Update fine into Student table while update issue_tab
    dbms_output.put_line('latefine = ' || latefine);
    update Student set fine=latefine where st_id=:old.st_id;
END;

/* Update statement to test trigger */
update Issue_tab set return_date=TO_DATE('2022/03/02','%yyyy-%mm-%dd') where st_id=100;
select * from Student;

/*
Sapple 4
    Create a Table
    Customer(id int primary key, name varchar(20),
                age int, address varchar(30), sal int);

    Create a trigger to display the salary difference between new and old values
    before - insert, delete or update of values on the table
*/
drop table Customer;
create table Customer(id int primary key,
                      name varchar(20), age int, address varchar(30), sal int);

insert into Customer values(1, 'Alice', 20, 'Address1', 10000);
insert into Customer values(2, 'Bob', 30, 'Address2', 20000);
insert into Customer values(3, 'Cindy', 40, 'Address3', 30000);
insert into Customer values(4, 'Sam', 50, 'Address4', 40000);
insert into Customer values(5, 'Eric', 60, 'Address5', 50000);

select * from  Customer;

drop trigger trigger5;
```

```
CREATE OR REPLACE TRIGGER trigger5
BEFORE DELETE OR INSERT OR UPDATE ON Customer
REFERENCING NEW AS NEW OLD AS OLD
FOR EACH ROW
DECLARE
    sal_diff int;
BEGIN
    dbms_output.put_line('TRIGGER 5');
    if (:new.id <= 0) then
        dbms_output.put_line('Error ...! : Invaid ID');
        RAISE_APPLICATION_ERROR(-20000, 'Error ...! : Invaid ID ');
    else
        sal_diff := :new.sal - :old.sal;
        dbms_output.put_line('Old Salary = ' || :old.sal);
        dbms_output.put_line('New Salary = ' || :new.sal);
        dbms_output.put_line('Salary diff = ' || sal_diff);
    end if;
END;

/* Update statement to test trigger */
update Customer set sal=30000 where id=1;
select * from Customer;

/* You can also write a sample program to test this trigger */
DECLARE
    new_sal int;
    cust_id int;
BEGIN
    cust_id := &cust_id;
    new_sal := &new_sal;
    dbms_output.put_line('new_sal = ' || new_sal);
    if (cust_id <= 0) then
            RAISE_APPLICATION_ERROR(-20000, 'Error ...! : Invaid ID ');
    else
        update Customer set sal=new_sal where id=cust_id;
    end if;
END;
/
```