--Display all trigger name---

SELECT name FROM sqlite_master

WHERE type = 'trigger';

Syntax:

```
create trigger <trigger name>
before/after/instead of
insert/update/delete [/update of column name]
on <table name / view name>
[for each row]
[when condition]
begin
    ...
end;
```

Que) Write a trigger to restrict the user from using the emp table on sunday.

Ans)

```
drop table emp;

create table emp
(
ID int,
NAME varchar(20),
SALARY int
);

insert into emp values(1,'viral',5000);
insert into emp values(2,'Ramesh',6000);
insert into emp values(3,'Mahesh',7000);
insert into emp values(4,'Syam',2000);
```

```
drop trigger tr1;

create trigger tr1
before insert on emp
begin
select case
when (select strftime('%w','now') = '6') then
   RAISE(ABORT,'sorrry...today is Saturday...You cannot insert data.')
end;
end;

drop trigger tr2;

create trigger tr2
after delete on emp
begin
select case
when (select strftime('%w','now') = '5')
then RAISE(ABORT,'today is thursday...not allow to delete')
end;
end;

drop trigger tr3;

create trigger tr3
before update on emp
begin
select case
when (select strftime('%w','now') = '5') then
  RAISE(ABORT,'today is Thursday..not allow to update')
end;
end;
```

```
insert into emp values(5,'Tejash',8000);
delete from emp where id=1;
update emp set name='sss' where id=2;

delete from emp;

--we can also use ROLLBACK/FAIL at ABORT
--When we drop table, All triggers on table drop automatically.
--At this time, SQLite supports only FOR EACH ROW triggers, not FOR EACH STATEMENT triggers.
--Hence, explicitly specifying FOR EACH ROW is optional.
```

Que) Write a trigger to insert old record of product price in product_history table before update product price in product table.

Ans)

```
drop table product_history;
drop table product;

create table product_history
(
product_id varchar(3),
unit_price number,
time_of_update text
);

create table product
(
product_id varchar(3),
unit_price number
);

insert into product values('P1',1000);
```

```
insert into product values('P2',2000);
insert into product values('P3',3000);

drop trigger tr4;

CREATE TRIGGER tr4
BEFORE UPDATE OF unit_price
ON product
BEGIN
INSERT INTO product_history VALUES
(old.product_id,old.unit_price,datetime('now','localtime'));
END;

select * from product;
select * from product_history;

update product set unit_price=500 where product_id='P1';
update product set unit_price=2500 where product_id='P2';
update product set unit_price=1500 where product_id='P3';

select * from product;
select * from product_history;

update product set unit_price=unit_price+100;

select * from product;
select * from product_history;
```

---

insert trigger allow only new

update trigger allow new and old both

delete trigger allow only old

Que) Generate primary key using MAX Function

Ans)

```
Drop table emp;

create table emp
(
eno varchar(10),
ename varchar(30),
esalary number
);

drop trigger tr5;

create trigger tr5 after insert on emp
BEGIN
update emp set eno=
(
 select
 case
 when ((select count(eno) from emp) = 0) then 'E1'
 else
 (select 'E'||(cast( substr(max(eno),2) as integer)+1) from emp)
 end
) where eno IS NULL;
END;

select * from emp;

insert into emp(ename,esalary) values('Viral',5000);
insert into emp(ename,esalary) values('Sanket',6000);
insert into emp(ename,esalary) values('Jinesh',7000);
```

select * from emp;

Que)

1.create trigger to open account with minimum 500 balance.

2.Create trigger to maintain minimum 500 balance in account at the time of withdraw

Ans)

```
drop table ACCOUNT;

CREATE TABLE ACCOUNT
(
ano INTEGER PRIMARY KEY,
balance REAL
);

drop trigger tr6;

CREATE TRIGGER tr6 BEFORE INSERT
ON ACCOUNT
BEGIN
 SELECT
 CASE
 WHEN NEW.balance < 500 THEN
 RAISE(ABORT,'To open account minimun 500 balance required.')
 END;
END;

insert into ACCOUNT(balance) values(200);
insert into ACCOUNT(balance) values(500);
insert into ACCOUNT(balance) values(1000);
insert into ACCOUNT(balance) values(2000);
```

```
select * from ACCOUNT;

drop trigger tr7;

CREATE TRIGGER tr7 AFTER UPDATE
ON ACCOUNT
BEGIN
 SELECT
 CASE
 WHEN NEW.balance < 500 THEN
 RAISE(ABORT,'Cannot withdraw as minimum 500 balance required')
 END;
END;

update ACCOUNT set balance=balance-200 where ano=2;
select * from ACCOUNT;
update ACCOUNT set balance=balance-200 where ano=2;
select * from ACCOUNT;
update ACCOUNT set balance=balance-200 where ano=2;
select * from ACCOUNT;

update ACCOUNT set balance=balance-100;

----another way using when condition with trigger------

drop trigger tr7;
CREATE TRIGGER tr7 AFTER UPDATE
ON ACCOUNT
WHEN NEW.balance < 500
BEGIN
 select RAISE(ABORT,'Cannot withdraw as minimum 500 balance required');
END;
```

Que) create trigger if account balance is less than 500
then close account by delete all data

```
drop table ACCOUNT;

CREATE TABLE ACCOUNT
(
ano INTEGER PRIMARY KEY,
balance REAL
);

insert into ACCOUNT(balance) values(1000);
insert into ACCOUNT(balance) values(2000);

drop trigger tr7;

CREATE TRIGGER tr7 after update
ON ACCOUNT
WHEN NEW.balance < 500
BEGIN
select
case
when new.balance < 0 then
  raise(abort,'cannot withdraw as withdraw amount is less than balance')
end;
delete from ACCOUNT where ano=old.ano;
END;

update ACCOUNT set balance=balance-300 where ano=1;
select * from ACCOUNT;
update ACCOUNT set balance=balance-300 where ano=1;
select * from ACCOUNT;
```

```
update ACCOUNT set balance=balance-300 where ano=1;
select * from ACCOUNT;

update ACCOUNT set balance=balance-3000 where ano=2
select * from ACCOUNT;

update ACCOUNT set balance=balance-300;
```

----UPDATE OF column-------

Que) create trigger for cannot change password of any user

Ans)

```
drop table ONLINE_ACCOUNT;

create table ONLINE_ACCOUNT
(
  ano integer PRIMARY KEY,
  username text,
  password text
);

insert into ONLINE_ACCOUNT(username,password) values('Admin','1234');
insert into ONLINE_ACCOUNT(username,password) values('user1','1234');
insert into ONLINE_ACCOUNT(username,password) values('user2','1234');

select * from ONLINE_ACCOUNT;

drop trigger tr8;

CREATE TRIGGER tr8 BEFORE UPDATE of password
ON ONLINE_ACCOUNT
BEGIN
  select RAISE(ABORT,'Cannot change password');
```

```
END;

update ONLINE_ACCOUNT set password='abcd' where ano=1;
update ONLINE_ACCOUNT set username='USER0' where ano=1;
```

----------insted of trigger----------

Que) create triggers for write/modify view 'STUDENT_RESULT'.

OR

Que) create triggers to perform insert,update and delete operations on view 'STUDENT_RESULT'.

OR

Que) create triggers for view 'STUDENT_RESULT' as

1) when we try to insert data in view,
   it will insert data in base tables STUDENT and RESULT.
2) when we try to update data in view,
   it will update data in base tables STUDENT and RESULT.
3) when we try to delete data in view,
   it will delete data from base tables STUDENT and RESULT.

```
drop table RESULT;
drop table STUDENT;

create table STUDENT
(
  roll_no integer primary key,
  name text
);
create table RESULT
(
   rno integer references STUDENT(roll_no),
   s1 int,
   s2 int,
```

```
   s3 int,
   res text
);
insert into STUDENT values(1,'ABC');
insert into STUDENT values(2,'PQR');
insert into STUDENT values(3,'XYZ');

insert into RESULT values(1,60,70,80,'PASS');
insert into RESULT values(2,10,20,40,'FAIL');
insert into RESULT values(3,80,90,75,'PASS');

select * from STUDENT;
select * from RESULT;
-------
create view STU_RES_VW as
select rno,name,s1,s2,s3,res from STUDENT inner join RESULT on rno=roll_no;

select * from STU_RES_VW;
-------
drop trigger t1;

create trigger t1
INSTEAD OF insert on STU_RES_VW
begin
  insert into STUDENT values(new.rno,new.name);
  insert into RESULT values(new.rno,new.s1,new.s2,new.s3,new.res);
end;

insert into STU_RES_VW values(4,'AAA',70,70,70,'PASS');

select * from STUDENT;
select * from RESULT;
select * from STU_RES_VW;
```

```
insert into STU_RES_VW
values(5,'BBB',85,70,60,'PASS'),(6,'CCC',65,72,82,'PASS'),(7,'DDD',77,71,73,'PASS');

select * from STUDENT;
select * from RESULT;
select * from STU_RES_VW;
---------
drop trigger t2;

create trigger t2
INSTEAD OF update on STU_RES_VW

begin
  update RESULT set s1=new.s1,s2=new.s2,s3=new.s3,res=new.res where rno=old.rno;
  update STUDENT set name=new.name where roll_no=old.rno;
end;

update STU_RES_VW set name='FFF',s1=10,s2=10,s3=10 where rno=2;

select * from STUDENT;
select * from RESULT;
select * from STU_RES_VW;

update STU_RES_VW set s1=50;

select * from STUDENT;
select * from RESULT;
select * from STU_RES_VW;
-------
drop trigger t3;

create trigger t3
```

```
INSTEAD OF delete on STU_RES_VW
begin
  delete from RESULT where rno = old.rno;
  delete from STUDENT where roll_no = old.rno;
end;

delete from STU_RES_VW where rno=1;

select * from STUDENT;
select * from RESULT;
select * from STU_RES_VW;

delete from STU_RES_VW;

select * from STUDENT;
select * from RESULT;
select * from STU_RES_VW;
```