**CS 7330**

**Homework 8.1 MLO 8.\***

1) Given the View, which of the following statements are valid?

CREATE VIEW DEPT\_SUMMARY (D, C, TOTAL\_S, AVERAGE\_S)

AS SELECT DNO, COUNT(\*), SUM(SALARY), AVG(SALARY)

FROM EMPLOYEE

GROUP BY DNO; --- which of the following is allowed?

-- A

SELECT \* FROM DEPT\_SUMMARY;

* This is allowed as select is just getting all columns from the view.

-- B

SELECT D, C FROM DEPT\_SUMMARY WHERE TOTAL\_S > 100000;

* This is allowed since columns D, C, TOTAl\_S are defined in the view and gets the department number, number of people in that department and total salary of that department.

-- C

SELECT D, AVERAGE\_S FROM DEPT\_SUMMARY WHERE C > ( SELECT C FROM DEPT\_SUMMARY WHERE D = 4);

* This is allowed since all columns exist in the view. This would the

-- D

UPDATE DEPT\_SUMMARY SET D = 3 WHERE D = 4;

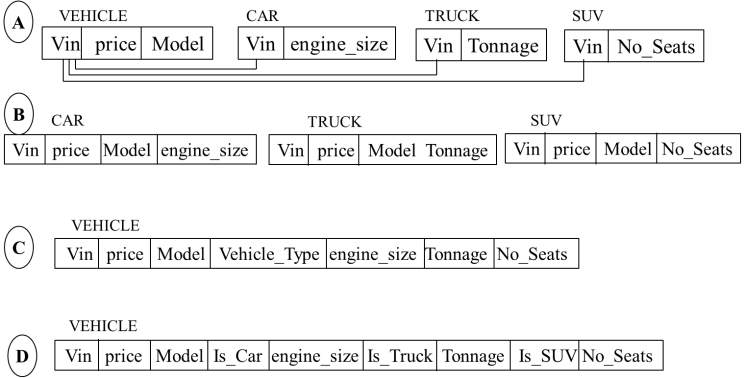
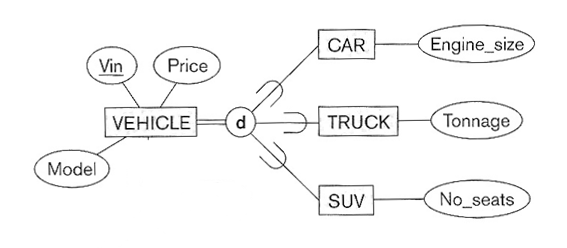
* This is allowed and changes employees who work for department 3 to department 4.

-- E

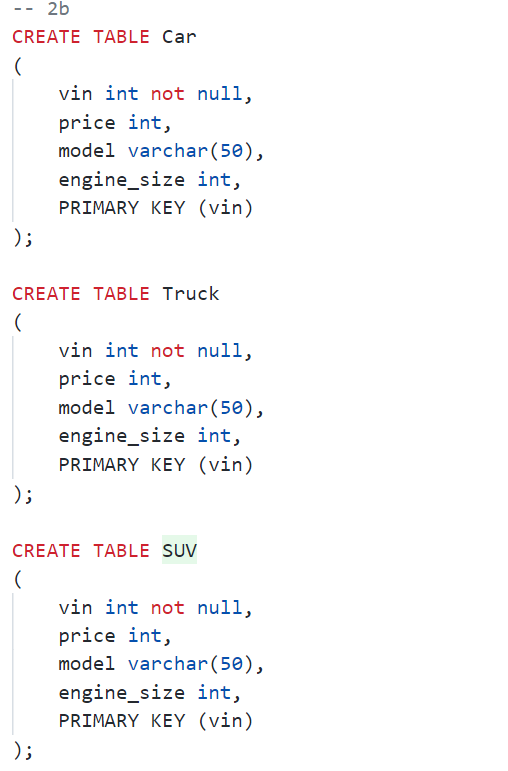
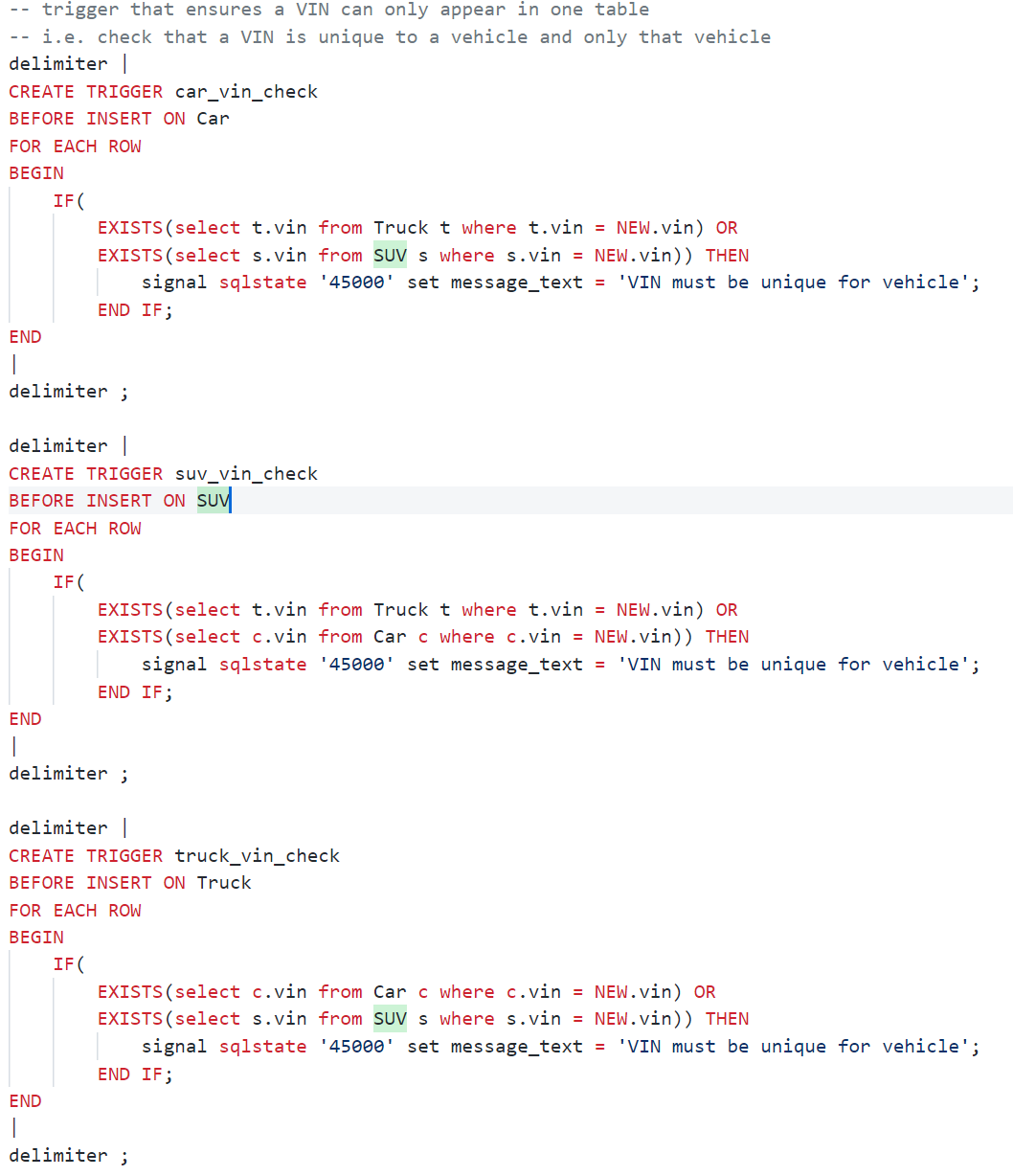
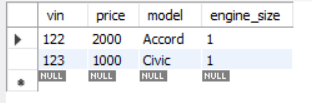
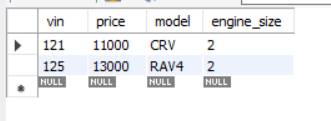
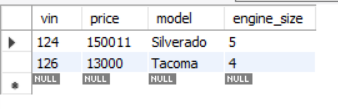
DELETE FROM DEPT\_SUMMARY WHERE C > 4;

* This is allowed and deletes the department where there are less than 4 employees.

For this HW, I am using MySql 5.5 on MySql Workbench

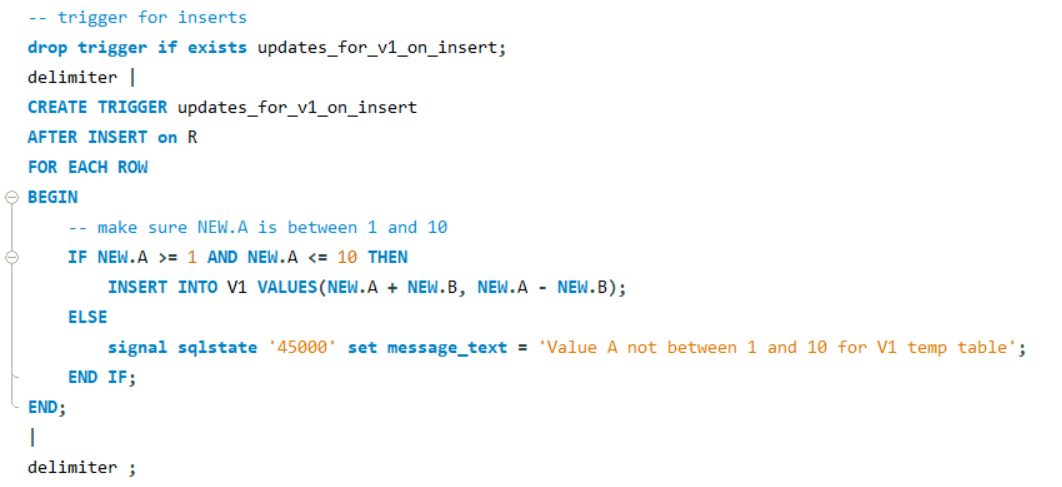
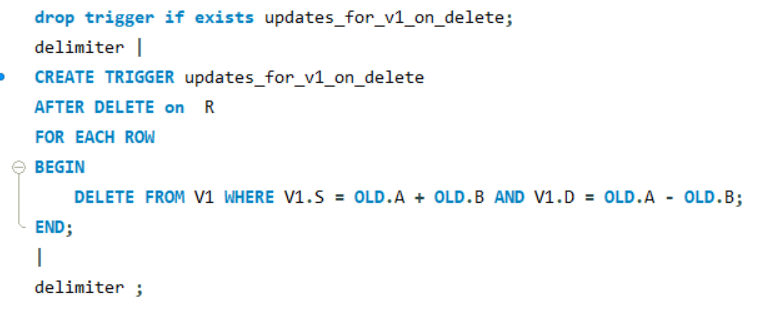
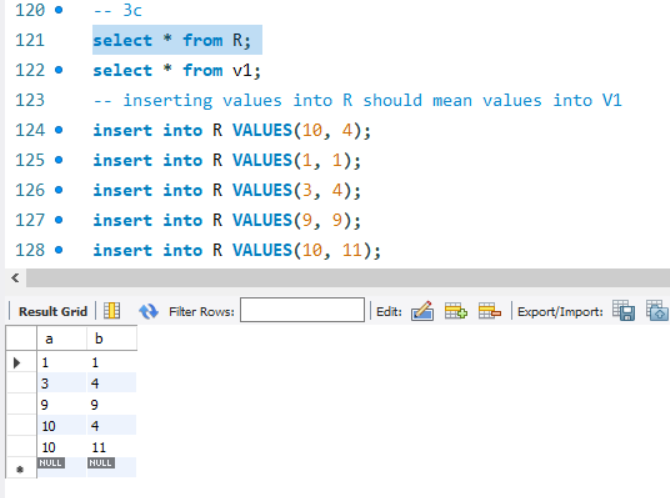
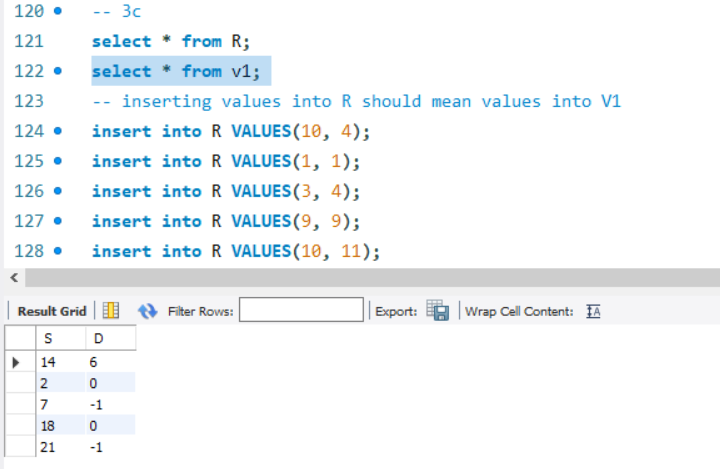
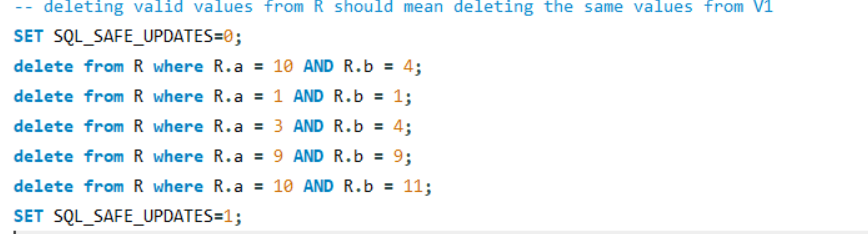
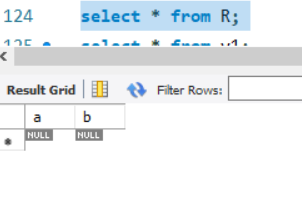
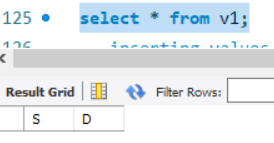
2) Consider this ER diagram and 4 possible options:

**Using option B or C:**  Implement one by showing the create table(s) statement. If you select B, write the trigger that ensures a VIN can only appear in one table. If you select C, write the trigger that requires the data associated with a particular Vehicle\_Type is not null, and sets the other data to null. Provide at least 3 sample data that demonstrates your trigger is working as expected.

* Option B
  + Creating the tables
    - 
  + The trigger
    - 
  + Data in the tables
    - Car
      * 
    - SUV
      * 
    - Truck
      * 
  + Inserting existing VIN(from an SUV entry) into Car
    - 
    - 
  + Inserting existing VIN(from a Car entry) into SUV
    - 
    - 
  + Inserting existing VIN(from an SUV) into Truck
    - 
    - 

3) One implementation technique of a view is as a temporary table. That creates the challenge for the system to keep the temporary table up-to-date as modifications are done in the base table(s). Consider this example:

create table R (a int, b int, primary key(a,b));  
create view V1 as select A+B as S, A-B as D from R where A between 1 and 10;

1. Create a temporary table for V1
   1. create temporary table V1 select A+B as S, A-B as D from R where A between 1 and 10;
2. Create the appropriate trigger(s) on R that will keep V1 current on inserts and deletes
   1. The insertion trigger
      1. 
   2. The deletion trigger
      1. 
3. Provide sufficient test cases to demonstrate the correctness of your solution.
   1. Here we insert to R
      1. 
   2. Here we observe V1 has the same entries as R
      1. 
         1. Note that the order is not kept(i.e. ascending order) but rather based on the order of the inserts
   3. Here we add values where A is outside the range stated “between 1 and 10”
      1. insert into R VALUES(11, 2);
         1. 
         2. 
      2. insert into R VALUES(0, 3);
         1. 
         2. 
   4. Here we delete the entries that were added above
      1. 
      2. We see that table R is empty
         1. 
      3. We also see that temp table V1 is also empty
         1. 

# The Queries

-- 2b

CREATE TABLE Car

(

    vin int not null,

    price int,

    model varchar(50),

    engine\_size int,

    PRIMARY KEY (vin)

);

CREATE TABLE Truck

(

    vin int not null,

    price int,

    model varchar(50),

    engine\_size int,

    PRIMARY KEY (vin)

);

CREATE TABLE SUV

(

    vin int not null,

    price int,

    model varchar(50),

    engine\_size int,

    PRIMARY KEY (vin)

);

-- trigger that ensures a VIN can only appear in one table

-- i.e. check that a VIN is unique to a vehicle and only that vehicle

delimiter |

CREATE TRIGGER car\_vin\_check

BEFORE INSERT ON Car

FOR EACH ROW

BEGIN

    IF(

        EXISTS(select t.vin from Truck t where t.vin = NEW.vin) OR

        EXISTS(select s.vin from SUV s where s.vin = NEW.vin)) THEN

            signal sqlstate '45000' set message\_text = 'VIN must be unique for vehicle';

        END IF;

END

|

delimiter ;

delimiter |

CREATE TRIGGER suv\_vin\_check

BEFORE INSERT ON SUV

FOR EACH ROW

BEGIN

    IF(

        EXISTS(select t.vin from Truck t where t.vin = NEW.vin) OR

        EXISTS(select c.vin from Car c where c.vin = NEW.vin)) THEN

            signal sqlstate '45000' set message\_text = 'VIN must be unique for vehicle';

        END IF;

END

|

delimiter ;

delimiter |

CREATE TRIGGER truck\_vin\_check

BEFORE INSERT ON Truck

FOR EACH ROW

BEGIN

    IF(

        EXISTS(select c.vin from Car c where c.vin = NEW.vin) OR

        EXISTS(select s.vin from SUV s where s.vin = NEW.vin)) THEN

            signal sqlstate '45000' set message\_text = 'VIN must be unique for vehicle';

        END IF;

END

|

delimiter ;

-- insert  3 sample data

insert into Truck values(125, 30000, 'Tundra', 3);

insert into Car values(125, 13000, 'Elantra', 3);

insert into SUV values(122, 20000, 'Sonata', 3);

-- \*\*\*\*\*\*\*\*\*PAGE BREAK\*\*\*\*\*\*\*\*\*\*

-- 3a

drop table if exists R;

create table R

(

    a int,

    b int,

    primary key(a,b)

);

-- 3b

drop temporary table if exists V1;

create temporary table V1 select A+B as S, A-B as D from R where A between 1 and 10;

-- 3c

-- trigger for inserts

drop trigger if exists updates\_for\_v1\_on\_insert;

delimiter |

CREATE TRIGGER updates\_for\_v1\_on\_insert

AFTER INSERT on R

FOR EACH ROW

BEGIN

    -- make sure NEW.A is between 1 and 10

    IF NEW.A >= 1 AND NEW.A <= 10 THEN

        INSERT INTO V1 VALUES(NEW.A + NEW.B, NEW.A - NEW.B);

    ELSE

        signal sqlstate '45000' set message\_text = 'Value A not between 1 and 10 for V1 temp table';

    END IF;

END;

|

delimiter ;

-- trigger for delete

drop trigger if exists updates\_for\_v1\_on\_delete;

delimiter |

CREATE TRIGGER updates\_for\_v1\_on\_delete

AFTER DELETE on  R

FOR EACH ROW

BEGIN

    DELETE FROM V1 WHERE V1.S = OLD.A + OLD.B AND V1.D = OLD.A - OLD.B;

END;

|

delimiter ;

-- 3c

select \* from R;

select \* from v1;

-- inserting values into R should mean values into V1

insert into R VALUES(10, 4);

insert into R VALUES(1, 1);

insert into R VALUES(3, 4);

insert into R VALUES(9, 9);

insert into R VALUES(10, 11);

-- inserting values into R outside of V1's where range

insert into R VALUES(11, 2);

insert into R VALUES(0, 3);

-- deleting valid values from R should mean deleting the same values from V1

SET SQL\_SAFE\_UPDATES=0;

delete from R where R.a = 10 AND R.b = 4;

delete from R where R.a = 1 AND R.b = 1;

delete from R where R.a = 3 AND R.b = 4;

delete from R where R.a = 9 AND R.b = 9;

delete from R where R.a = 10 AND R.b = 11;

SET SQL\_SAFE\_UPDATES=1;

(END)