Part 1: Create Tables:

```
create table SOFTWARE_PRODUCT
2 □(
         Pname varchar(20) not null,
         Pversion varchar(20) not null,
Pstatus enum('ready', 'usable', 'not-ready') default "not-ready",
primary key(Pname, Pversion)
 6
 8
         create table COMPONENT
10
     ₽(
       Cname varchar(50) not null,
Cversion varchar(3) not null,
Language enum('C','C++','C#','Java','PHP'),
Size integer not null,
Cstatus enum('ready', 'usable', 'not-ready') default "not-ready",
primary key(Cname, Cversion)
);
11
12
13
14
15
16
17
18
19 •
         create table EMPLOYEE
20
21
22
        ID integer not null,
         Ename varchar(60) not null,
Hiredate date not null,
23
24
         Mgr integer,
25
         Seniority varchar(10) default null,
26
27
28
       primary key(ID)
29 •
         create table INSPECTION
30
31
        Cname varchar(50) not null,
32
         Cversion varchar(3) not null,
33
         Date date not null,
34
         Score integer,
35
         Description varchar(4000) not null,
36
         key(date),
37
         foreign key(Cname, Cversion) references COMPONENT(Cname, Cversion) on delete cascade on update cascade
       L);
38
40 •
         create table HAVE
      ⊟(
 41
         Pname varchar(10) not null,
         Pversion varchar(10) not null,
Cname varchar(50) not null,
 43
         Cversion varchar(3) not null,
foreign key(Pname, Pversion) references SOFTWARE_PRODUCT(Pname, Pversion) on delete cascade on update cascade,
foreign key(Cname, Cversion) references COMPONENT(Cname, Cversion) on delete cascade on update cascade
 45
 46
       L);
 48
 49
 50 •
         create table OWN
 51
         Cname varchar(50) not null,
 53
54
         Cversion varchar(3) not null,
ID integer not null,
          foreign key(Cname, Cversion) references COMPONENT(Cname, Cversion) on delete cascade on update cascade, foreign key(ID) references EMPLOYEE(ID) on delete cascade on update cascade
 55
 58
 59 •
          create table INSPECTED
       旦(
 60
         Cname varchar(50) not null,
 61
         Cversion varchar(3) not null,
         Date date not null, foreign key(Cname, Cversion) references COMPONENT(Cname, Cversion) on delete cascade on update cascade,
 63
          foreign key(Date) references INSPECTION(Date) on delete cascade on update cascade
 66
 68 •
          create table CONDUCT
 69
 70
71
         ID integer not null,
         Date date not null.
          foreign key(ID) references EMPLOYEE(ID) on delete cascade on update cascade,
          foreign key(Date) references INSPECTION(Date) on delete cascade on update cascade
       L);
```

Part 2: Insert

```
Size)
Size)
Size)
Size)
Size)
Size)
                        insert into EMPLOYEE (ID, Ename, Hiredate, Mgr) values (1010), insert into EMPLOYEE (ID, Ename, Hiredate, Mgr) values (1020), insert into EMPLOYEE (ID, Ename, Hiredate, Mgr) values (1020), insert into EMPLOYEE (ID, Ename, Hiredate, Mgr) values (1020), insert into EMPLOYEE (ID, Ename, Hiredate, Mgr) values (1050), insert into EMPLOYEE (ID, Ename, Hiredate, Mgr) values (1050), insert into EMPLOYEE (ID, Ename, Hiredate, Mgr) values (1070), insert into EMPLOYEE (ID, Ename, Hiredate, Mgr) values (1070), insert into EMPLOYEE (ID, Ename, Hiredate, Mgr) values (1070), insert into EMPLOYEE (ID, Ename, Hiredate, Mgr) values (1070), insert into EMPLOYEE (ID, Ename, Hiredate, Mgr) values (1070), insert into EMPLOYEE (ID, Ename, Hiredate, Mgr) values (1070), insert into EMPLOYEE (ID, Ename, Hiredate, Mgr) values (1070), insert into EMPLOYEE (ID, Ename, Hiredate, Mgr) values (1070), insert into EMPLOYEE (ID, Ename, Hiredate, Mgr) values (1070), insert into EMPLOYEE (ID, Ename, Hiredate, Mgr) values (1070), values (1070), insert into EMPLOYEE (ID, Ename, Hiredate, Mgr) values (1070), values (1070), insert into EMPLOYEE (ID, Ename, Hiredate, Mgr) values (1070), insert into EMPLOYEE (ID, Ename, Hiredate, Mgr) values (1070), insert into EMPLOYEE (ID, Ename, Hiredate, Mgr) values (1070), insert into EMPLOYEE (ID, Ename, Hiredate, Mgr) values (1070), insert into EMPLOYEE (ID, Ename, Hiredate, Mgr) values (1070), insert into EMPLOYEE (ID, Ename, Hiredate, Mgr) values (1070), insert into EMPLOYEE (ID, Ename, Hiredate, Mgr) values (1070), insert into EMPLOYEE (ID, Ename, Hiredate, Mgr) values (1070), insert into EMPLOYEE (ID, Ename, Hiredate, Mgr) values (1070), insert into EMPLOYEE (ID, Ename, Hiredate, Mgr) values (1070), insert into EMPLOYEE (ID, Ename, Hiredate, Mgr) values (1070), insert into EMPLOYEE (ID, Ename, Hiredate, Mgr) values (1070), insert into EMPLOYEE (ID, Ename, Hiredate, Mgr) values (1070), insert into EMPLOYEE (ID, Ename, Hiredate, Mgr) values (1070), insert into EMPLOYEE (ID, Ename, Hiredate, Mgr) values (1070), 
                                                                                                                                                                                                                                                                                                       ('Keyboard Driver', 'Kil', '2010-02-14', 100, 'legacy code which is already approved');
('Touch Screen Driver', 'T00', '2017-06-01', '95, 'initial release ready for usage');
('Douse Interface', 'D00', '2010-02-22', 55, 'improved, but only handles D02' format');
('Dosse Interface', 'D00', '2010-02-22', 5, 'improved, but only handles D02' format');
('Dosse Interface', 'D00', '2010-02-22', 100, 'instribute');
('Obase Interface', 'D00', '2010-02-20, (100, 'instribute');
('Comac Interface', 'D00', '2010-02-20, (100, 'instribute');
('Comac Interface', 'D00', '2010-03-20, (100, 'instribute');
('Comac Interface', 'D00', '2011-03-20, (100, 'instribute');
('Comac Interface', 'D00', '2011-03-20, (100, 'instribute');
('Comac Interface', 'D00', '2011-03-20, (100, 'instribute');
('Comac Interface', 'D00', '2011-03-20', '80, 'instribute');
('Comac Interface', 'D00', '2011-03-20', '80, 'instribute');
('Nath unit', 'A02', '2011-03-20', '100, 'instribute');
('Nath unit', 'A02', '2011-03-20', '100, 'instribute');

('Nath unit', 'A02', '2011-03-20', '100, 'instribute');
                                                       insert into HAVE values ('Excel', '2010', 'Keyboard Driver', 'K11');
insert into HAVE values ('Excel', '2010', 'Dbase Interface', 'D00');
insert into HAVE values ('Excel', '2015', 'Keyboard Driver', 'K11');
insert into HAVE values ('Excel', '2015', 'Dbase Interface', 'D01');
insert into HAVE values ('Excel', '2015', 'Pen driver', 'P01');
insert into HAVE values ('Excel', '2018beta', 'Keyboard Driver', 'K11');
insert into HAVE values ('Excel', '2018beta', 'Touch Screen Driver', 'T00');
insert into HAVE values ('Excel', 'secret', 'Keyboard Driver', 'K11');
insert into HAVE values ('Excel', 'secret', 'Touch Screen Driver', 'T00');
insert into HAVE values ('Excel', 'secret', 'Touch Screen Driver', 'T00');
insert into HAVE values ('Excel', 'secret', 'Chart generator', 'C11');
insert into HAVE values ('Excel', 'secret', 'Chart generator', 'C11');
insert into HAVE values ('Excel', 'secret', 'Math unit', 'A02');
    38 •
     39 •
    40 •
     41 •
    42 •
    43 •
    45 0
    46 •
    47 •
    48 •
    49 •
     50
    51
                                                            insert into OWN values ('Keyboard Driver', 'K11', 10100);
     52 •
                                                         insert into OWN values ( Keyboard Driver , KII , 10100); insert into OWN values ('Touch Screen Driver', 'T00', 10100); insert into OWN values ('Dbase Interface', 'D00', 10200); insert into OWN values ('Dbase Interface', 'D01', 10300); insert into OWN values ('Chart generator', 'C11', 10200); insert into OWN values ('Pen driver', 'P01', 10700); insert into OWN values ('Math unit', 'A01', 10200); insert into OWN values ('Math unit', 'A02', 10200);
     53 •
    54 •
     55 •
     56 •
     57 •
    58 •
     59 •
    60
    61 •
                                                             insert into INSPECTED values ('Keyboard Driver', 'K11', '2010-02-14');
                                                            insert into INSPECTED values ('Touch Screen Driver', 'T00', '2017-06-01');
    62 •
                                                       insert into INSPECTED values ('Touch Screen Driver', 'T00', '2017-06-02 insert into INSPECTED values ('Dbase Interface', 'D00', '2010-02-22'); insert into INSPECTED values ('Dbase Interface', 'D00', '2010-02-24'); insert into INSPECTED values ('Dbase Interface', 'D00', '2010-02-26'); insert into INSPECTED values ('Dbase Interface', 'D00', '2010-02-28'); insert into INSPECTED values ('Dbase Interface', 'D01', '2011-05-01'); insert into INSPECTED values ('Pen driver', 'P01', '2017-07-15'); insert into INSPECTED values ('Math unit', 'A01', '2014-06-10'); insert into INSPECTED values ('Math unit', 'A02', '2014-06-15'); insert into INSPECTED values ('Math unit', 'A02', '2014-06-30'); insert into INSPECTED values ('Math unit', 'A02', '2016-11-02');
    63 •
    64 •
    65 •
    67 •
     68 •
     69 •
    70 •
     71 •
     72 •
73
74 • insert into CONDUCT values (10100, '2010-02-14');
75 • insert into CONDUCT values (10200, '2017-06-01');
76 • insert into CONDUCT values (10100, '2010-02-22');
77 • insert into CONDUCT values (10100, '2010-02-24');
78 • insert into CONDUCT values (10100, '2010-02-26');
79 • insert into CONDUCT values (10100, '2010-02-28');
80 • insert into CONDUCT values (10200, '2011-05-01');
81 • insert into CONDUCT values (10300, '2017-06-10');
82 • insert into CONDUCT values (10100, '2014-06-10');
83 • insert into CONDUCT values (10100, '2014-06-15');
84 • insert into CONDUCT values (10100, '2014-06-30');
85 • insert into CONDUCT values (10100, '2014-06-30');
86
     73
  86
```

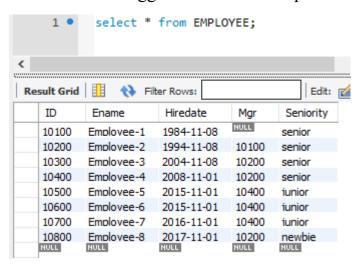
Part 3: Create Triggers

(1) Trigger for EMPLOYEE INSERT to consider the SENIORITY of

the employee:

```
delimiter //
       create trigger EMPLOYEE INS before insert on EMPLOYEE for each row
     日 begin
日 i
 3
 4
           if(exists(select * from employee where ID = new.Mgr) or new.ID = 10100) then
               if (new.hiredate > date_sub(now(), interval 1 year)) then
 5
 6
                    set new.seniority = "newbie";
 7
               else
 8
                    if (new.hiredate > date_sub(now(), interval 5 year)) then
                        set new.seniority = "junior";
9
10
                        set new.seniority = "senior";
11
12
                    end if;
13
               end if;
14
15
               SIGNAL SQLSTATE '45000'
16
               SET MESSAGE_TEXT = "Mgr data is wrong";
17
           end if;
     L<sub>end;</sub>
18
19
       //
20
       delimiter;
```

To Ensure this trigger can meet the requirement:



Using select * from and the result is correct.

It can meet the requirement.

(2) Trigger for ID in EMPLOYEE to make sure ID is 5-digit number

```
delimiter //
       create trigger id INS before insert on EMPLOYEE for each row
    □ begin if
3
           if (LENGTH(new.ID) != 5) then
4
               SIGNAL SQLSTATE '45000'
5
               SET MESSAGE TEXT = "ID should be 5 digit number";
6
7
           end if;
8
      end
     L//
9
       delimiter;
10
```

To Ensure this trigger can meet the requirement:

```
1 • insert into EMPLOYEE(ID, Ename, Hiredate, Mgr) values (100000, 'Employee-not exist', '1994-02-21', 10100);
```

Try to insert a row with length of ID not equal to 5,

```
o 400 05:14:41 insert into EMPLOYEE(ID, Ename, Hiredate, Mgr) values (100000, Employee-not exist.' 1994-02:21; 10100) Error Code: 16:44, ID should be 5 digit number
```

Meet the requirement

(3) Trigger for Score in INSPECTION to make sure score is 0-100

```
delimiter //
      create trigger score_INS before insert on INSPECTION for each row
2 •
    日 begin
日 if
3
4
          if new.Score > 100 or new.Score < 0 then
               SIGNAL SQLSTATE '45000'
5
6
               SET MESSAGE_TEXT = "Score should be a value between 0 and 100 or null";
7
           end if;
8
      end
     L//
9
10
      delimiter;
11
```

To Ensure this trigger can meet the requirement:

```
insert into INSPECTION (Cname, Cversion, Date, Score, Description) values ('Keyboard Driver', 'K11', '2010-02-14', 120, 'incorrect score');
```

Try to insert a row with score not in 0-100.

```
    401 05:18:28 insert into INSPECTION (Chame, Cversion, Date, Score, Description) values (Keyboard Driver', K11', 2010-02-14', 120, Incorrect score)
    Emor Code: 1644. Score should be a value between 0 and 100 or null
```

Meet the requirement.

(4) Trigger for Score in INSPECTION cannot be changed

```
1
       delimiter //
 2 •
       create trigger score UPD before update on INSPECTION for each row
    ⊟ begin
 3
4
           if new.Score is not null then
 5
               SIGNAL SQLSTATE '45000'
6
               SET MESSAGE_TEXT = "Score can never be changed";
7
8
      end
     L//
9
10
      delimiter;
```

To Ensure this trigger can meet the requirement:

```
update INSPECTION
set score = 99
where Cname = 'Keyboard Driver' and Cversion = 'K11';
```

Try to change the score in INSPECTION:

```
3 402 05 20:36 update INSPECTION set score = 99 where Chame = "Keyboard Driver" and Cversion = "K11" Error Code: 1644. Score can never be changed
```

Meet the requirement.

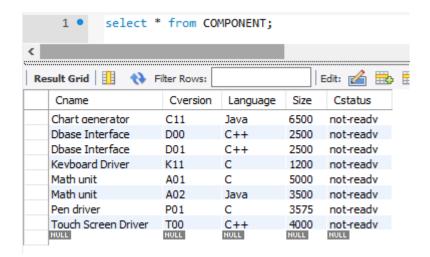
(5) Trigger for cstatus in COMPONENT to consider the status of COMPONENT, it should be only "ready", "not-ready", or "usable".

```
delimiter //
      create trigger cstatus_INS after insert on INSPECTION for each row
2
    □ begin i
3
4
          if new.score > 90 then
5
               update COMPONENT, INSPECTION
6
              set COMPONENT.Cstatus = 'ready'
7
              where COMPONENT.Cname = new.Cname and COMPONENT.Cversion = new.Cversion
    自
8
              and new.Date = (select max(Date) from INSPECTION where
9
              INSPECTION.Cname = new.Cname and INSPECTION.Cversion = new.Cversion);
10
          elseif new.score < 75 then
11
              update COMPONENT, INSPECTION
12
              set COMPONENT.Cstatus = 'not-ready'
13
              where COMPONENT.Cname = new.Cname and COMPONENT.Cversion = new.Cversion
14
              and new.Date = (select max(Date) from INSPECTION where
15
              INSPECTION.Cname = new.Cname and INSPECTION.Cversion = new.Cversion);
16
          else
              update COMPONENT, INSPECTION
17
18
              set COMPONENT.Cstatus = 'usable'
19
              where COMPONENT.Cname = new.Cname and COMPONENT.Cversion = new.Cversion
20
              and new.Date = (select max(Date) from INSPECTION where
21
              INSPECTION.Cname = new.Cname and INSPECTION.Cversion = new.Cversion);
22
           end if;
23
      end
24
      delimiter;
```

When score is larger than 90, it should be 'ready', if score is less than 75, it should be 'not-ready', otherwise, it should be 'usable'.

This trigger can choose the newest status of component because of the query "select max(Date)..." which means it can choose the latest data and take it as the newest status.

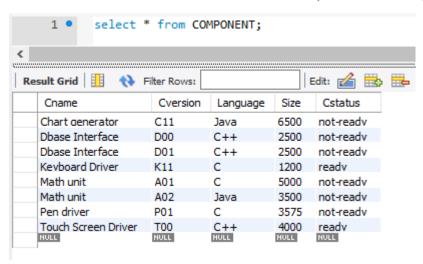
For example:



This is the initial statement.

insert into INSPECTION (Cname, Cversion, Date, Score, Description) values ('Keyboard Driver', 'K11', '2010-02-14', 100, 'legacy code which is already approved');
insert into INSPECTION (Cname, Cversion, Date, Score, Description) values ('Touch Screen Driver', 'T00', '2017-06-01', 95, 'initial release ready for usage');

INSERT some values into INSPECTION, awake the trigger.



From the capture we can see that because of the insertion of 'Keyboard Driver' and 'Touch Screen Driver'. The value in the COMPONENT also changed, which means the trigger works correctly.

It meets the requirement.

(6) Trigger for pstatus in SOFTWARE_PRODUCT to consider the status of SOFTWARE_PRODUCT, it also should be "ready", "not-ready" or "usable".

It should be judged as the worst status among its COMPONENT, so should

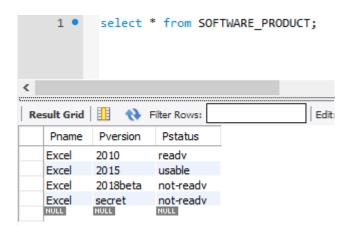
build the relation between the SOFTWARE_PRODUCT and COMPONENT, then create a trigger to consider the status of SOFTWARE PRODUCT.

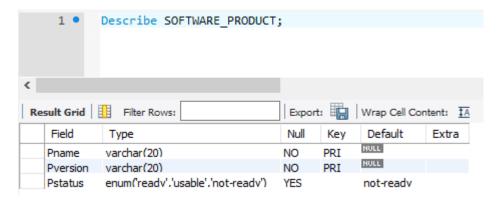
Create two triggers for Pstatus.

```
delimiter //
        create trigger pstatus_INS after update on COMPONENT for each row
 2 •
     □begin
            set @Pname = (select Pname from HAVE
     F
            where Cname = new.Cname and Cversion = new.Cversion);
 6
           set @Pversion = (select Pversion from HAVE
 7
            where Cname = new.Cname and Cversion = new.Cversion);
 8
            set @Cstatus = new.Cstatus;
 9
           set @Pstatus = (select Pstatus from SOFTWARE_PRODUCT
10
            where SOFTWARE PRODUCT.Pname = @Pname and SOFTWARE PRODUCT.Pversion = @Pversion);
11
            update SOFTWARE_PRODUCT
12
                set Pstatus = @Cstatus
13
            where (SOFTWARE PRODUCT.Pname = @Pname and SOFTWARE PRODUCT.Pversion = @Pversion);
14
15
        delimiter;
16
18
     delimiter //
       create trigger pstatus_UPD after update on SOFTWARE_PRODUCT for each row
19 •
    □begin
20
21
           declare i int default 0;
    set @Pname = (select Pname from HAVE
22
23
          where Cname = new.Cname and Cversion = new.Cversion);
24
          set @Pversion = (select Pversion from HAVE
25
          where Cname = new.Cname and Cversion = new.Cversion);
26
27
          set @loopn = (select count(*) from HAVE
          where HAVE.Pname = Pname and HAVE.Pversion = Pversion);
28
           set @Pstatus = (select Pstatus from SOFTWARE_PRODUCT
29
          where SOFTWARE_PRODUCT.Pname = @Pname and SOFTWARE_PRODUCT.Pversion = @Pversion);
          get_loop : LOOP
30
31
               if @loop = i then
32
                   LEAVE get_loop;
33
               end if;
34
          set @Cstatus = (select COMPONENT.Cstatus from COMPONENT
35
          where (COMPONENT.Cname = (select HAVE.Cname from HAVE
36
           where HAVE.Pname = @Pname and HAVE.Pversion = @Pversion limit i,1) and
37
           (select HAVE.Cversion from HAVE
38
           where HAVE.Pname = @Pname and HAVE.Pversion = @Pversion limit i,1)));
39
          if @Cstatus = 'not-ready' then
           set @Pstatus = 'not-ready';
elseif @Pstatus = 'ready' and @Cstatus = 'usable' then
10
41
42
              set @pstatus = 'usable';
43
14
               set @pstatus = 'not-ready';
45
           end if;
46
              set i = i + 1;
47
           end LOOP;
48
           update SOFTWARE PRODUCT set SOFTWARE PRODUCT.Pstatus = @pstatus
49
           where SOFTWARE_PRODUCT.Pname = @Pname and SOFTWARE_PRODUCT.Pversion = @Pversion;
50
      -end:
51
      //
     Ldelimiter;
52
```

Part 4: Select * from and Describe

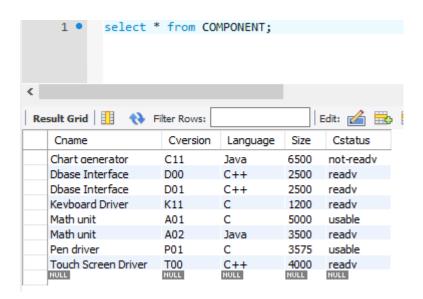
$(1) SOFTWARE_PRODUCT$



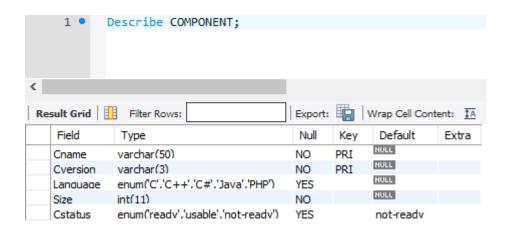


(2) COMPONENT

SELECT:

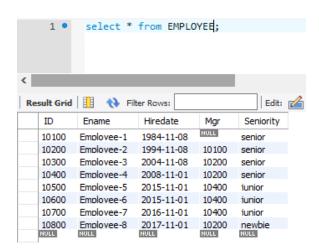


DESCRIBE:

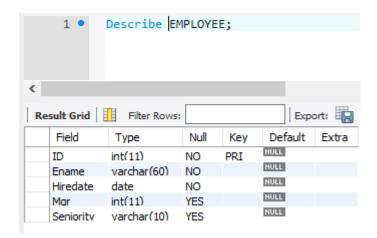


(3) EMPLOYEE

SELECT:

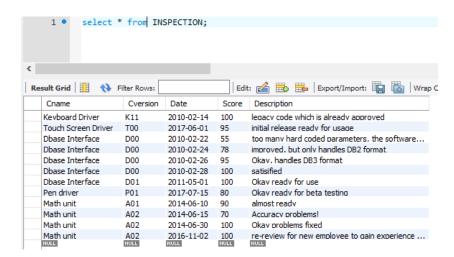


DESCRIBE:

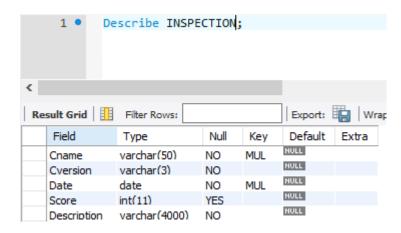


(4) INSPECTION

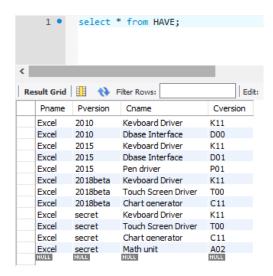
SELECT:

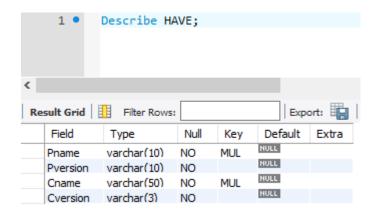


DESCRIBE:

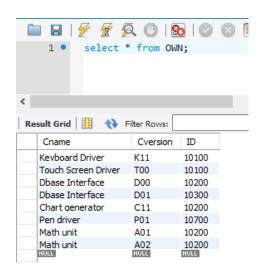


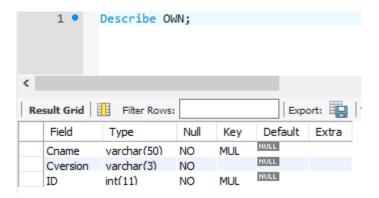
(5) HAVE





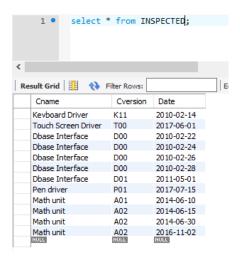
(6) **OWN**



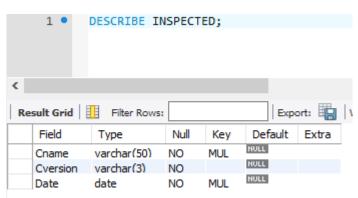


(7) INSPECTED

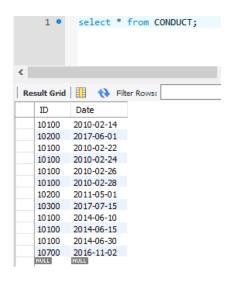
SELECT:

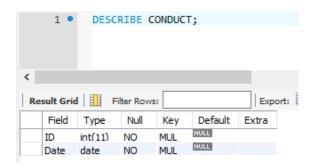


DESCRIBE:



(8) CONDUCT





Part 5: Explanation

In phase 1, I give an attribute in the relational table "HAVE" called "SOFTWARE_BUILD" while in phase 2 I choose not to use it because I found that I no longer need this attribute.

In addition, because of the values the phase 2 gave me, I have to add two attributes in table "EMPLOYEE", one is "Mgr" which store the manager id, another is "hiredate" which is the important attribute that can consider the SENIORITY of the employee.

I think I can improve or create more triggers to meet the requirement.