CSE 5330/7330 Fall 2017
Phase 3 Functional Requirements

Using your database populated with the data provided.

Everyone: Write Queries (and show the results) to answer the following questions:

1. List all software product names and versions and current product status.

```
select * from SOFTWARE_PRODUCT;
```

Result Grid | Filter Rows: | Edit:

Pname	Pversion	Pstatus
Excel	2010	readv
Excel	2015	usable
Excel	2018beta	not-readv
Excel	secret	not-readv
NULL	NULL	NULL

2. List the owner name, component name & version of all "not ready" components.

```
select Ename, Cname, Cversion from
(select COMPONENT.Cname, COMPONENT.Cversion, COMPONENT.Cstatus, EMPLOYEE.ID, EMPLOYEE.Ename from
(COMPONENT inner join OWN on COMPONENT.Cname = OWN.Cname and COMPONENT.Cversion = OWN.Cversion)
inner join EMPLOYEE on OWN.ID = EMPLOYEE.ID
where COMPONENT.Cstatus = 'not-ready')as A;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Ename	Cname	Cversion
Employee-2	Chart generator	C11

3. List all component names and versions that have not been inspected.

```
select COMPONENT.Cname, COMPONENT.Cversion from COMPONENT left join INSPECTION
on COMPONENT.Cname = INSPECTION.Cname and COMPONENT.Cversion = INSPECTION.Cversion
where score is null;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Cname	Cversion
Chart generator	C11

4. What is the average number of components owned per person?

```
select AVG(count_own)as result_of_average from
(select EMPLOYEE.ID,count(OWN.ID) as count_own from COMPONENT right join OWN on COMPONENT.Cname = OWN.Cname and COMPONENT.Cversion = OWN.Cversion
right join EMPLOYEE on EMPLOYEE.ID = OWN.ID group by EMPLOYEE.ID)as A;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

result_of_average
1.0000

5. What is the average score of all inspections for Excel secret?

```
select AVG(score) from
(select score from
(INSPECTION inner join HAVE on INSPECTION.Cname = HAVE.Cname and INSPECTION.Cversion = HAVE.Cversion)
inner join SOFTWARE_PRODUCT on SOFTWARE_PRODUCT.Pname = HAVE.Pname and SOFTWARE_PRODUCT.Pversion = HAVE.Pversion
where SOFTWARE_PRODUCT.Pname = 'Excel' and SOFTWARE_PRODUCT.Pversion = 'secret')as A;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

AVG(score)
93.0000

6. List all employees by name, seniority, count of components assigned to them, count of inspections performed by them and their average inspection score.

```
create view COUNT_OF_INSPECTION as
select Ename, Seniority, count(CEID)as count_of_inspections_performed, ifnull(AVG(score), 0) as average_inspection_score from
(select CONDUCT.ID as CEID, EMPLOYEE.ID as EID, Ename, Seniority, Score from
EMPLOYEE left join CONDUCT on EMPLOYEE.ID = CONDUCT.ID
left join INSPECTION on INSPECTION.Date = CONDUCT.Date) as A group by EID;

create view COUNT_OF_COMPONENT as
select Ename, Seniority, count(OID) as count_of_components_assigned from
(select OWN.ID as OID, EMPLOYEE.ID as EID, Ename, Seniority, COMPONENT.Cname, COMPONENT.Cversion from
EMPLOYEE left join OWN on EMPLOYEE.ID = OWN.ID
left join COMPONENT on COMPONENT.Cname = OWN.Cname and COMPONENT.Cversion = OWN.Cversion) as A group by EID;

select COUNT_OF_COMPONENT.Ename, COUNT_OF_COMPONENT.Seniority, count_of_inspections_performed, count_of_components_assigned, average_inspection_score from
COUNT_OF_INSPECTION inner join COUNT_OF_COMPONENT on COUNT_OF_INSPECTION.Ename = COUNT_OF_COMPONENT.Ename;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Ename	Seniority	count_of_inspections_performed	count_of_components_assigned	average_inspection_score
Emplovee-1	senior	8	2	86.0000
Emplovee-2	senior	2	4	97.5000
Emplovee-3	senior	1	1	80.0000
Emplovee-4	senior	0	0	0.0000
Emplovee-5	iunior	0	0	0.0000
Emplovee-6	iunior	0	0	0.0000
Emplovee-7	iunior	1	1	100.0000
Emplovee-8	newbie	0	0	0.0000

7. Assume an inspection that results in a “ready” status costs $200, and all other inspections cost $100 each. How much did *OSF* in 2010 for inspections conducted by each seniority level?

First, the period is in the year 2010, so firstly we should select inspections and seniorities of people who conducted those inspections and create a view as follows:

```
drop view if exists ForCOST;
create view ForCOST
as select Hiredate, score, INSPECTION.Date from
(INSPECTION inner join CONDUCT on INSPECTION.Date = CONDUCT.Date)
inner join EMPLOYEE on CONDUCT.ID = EMPLOYEE.ID
where INSPECTION.Date between '2010-01-01' and '2010-12-31';
```

Second, create 3 views containing the cost of ‘ready’ inspections and various seniorities of people who conducted this inspection, it should be noticed that the seniority is based on the date the inspection occurred.

```
drop view if exists newbie_Readycost;
create view newbie_Readycost as
select Hiredate, ifnull(count(score) * 200, 0) as new_cost_of_ready from ForCOST
where score > 90 and Hiredate > date_sub(Date, interval 1 year);
```

```
drop view if exists junior_Readycost;
create view junior_Readycost as
select Hiredate, ifnull(count(score) * 200, 0) as jun_cost_of_ready from ForCOST
where score > 90 and Hiredate > date_sub(Date, interval 5 year)
                 and Hiredate <= date_sub(Date, interval 1 year);
```

```
drop view if exists senior_Readycost;
create view senior_Readycost as
select Hiredate, ifnull(count(score) * 200, 0) as sen_cost_of_ready from ForCOST
where score > 90 and Hiredate <= date_sub(Date, interval 5 year);
```

Then, also create 3 views containing the cost of other inspections and various seniorities of people who conducted this inspection, it should be noticed that the seniority is based on the date the inspection occurred.

```
drop view if exists newbie_Otherscost;
create view newbie_Otherscost as
select Hiredate, ifnull(count(score) * 100, 0) as new_cost_of_others from ForCOST
where score <= 90 and Hiredate > date_sub(Date, interval 1 year);
```

```
drop view if exists junior_Otherscost;
create view junior_Otherscost as
select Hiredate, ifnull(count(score) * 100, 0) as jun_cost_of_others from ForCOST
where score <= 90 and Hiredate > date_sub(Date, interval 5 year)
                  and Hiredate <= date_sub(Date, interval 1 year);
```

```
drop view if exists senior_Otherscost;
create view senior_Otherscost as
select Hiredate, ifnull(count(score) * 100, 0) as sen_cost_of_others from ForCOST
where score <= 90 and Hiredate <= date_sub(Date, interval 5 year);
```

Finally, add the cost of 'ready' and others together in different seniorities and we can finally get the answer:

```
select (new_cost_of_ready + new_cost_of_others)as newbie_cost, (jun_cost_of_ready + jun_cost_of_others)as junior_cost, (sen_cost_of_ready + sen_cost_of_others)as senior_cost from
newbie_Readycost, newbie_Otherscost, junior_Readycost, junior_Otherscost, senior_Readycost, senior_Otherscost;
```

	newbie_cost	junior_cost	senior_cost
	0	0	800

The cost in senior level is 800, and each cost of other levels like junior and newbie is 0.

Everyone: Demonstrate ≡ show the SQL command(s) and result

8. Demonstrate the adding of a new inspection by employee 10400 on Pen driver - P01 held on 8/15/2017 with the score of 60 and description of "needs rework, introduced new errors".

Insert a new inspection, according to my own database, I need to write 3 INSERT clauses to complete the adding process.

First, we need to put the information into INSPECTION tables, which may influence the status of the component - 'Pen driver', 'P01'.

Then, when this insertion complete, the relational table CONDUCT which represents the relationship between the employee and the date when his/her conducted

because date can be the key. INSPECTED will also be influenced, is shows the relationship between the component and its date of inspection.

```
insert into INSPECTION values ('Pen driver', 'P01', '2017-08-15', 60, 'needs rework, introduced new errors');
insert into CONDUCT values (10400, '2017-08-15');
insert into INSPECTED values ('Pen driver', 'P01', '2017-08-15');
```

```
select * from INSPECTION;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Cname	Cversion	Date	Score	Description
Keyboard Driver	K11	2010-02-14	100	legacy code which is already approved
Touch Screen Driver	T00	2017-06-01	95	initial release ready for usage
Dbase Interface	D00	2010-02-22	55	too many hard coded parameters. the software...
Dbase Interface	D00	2010-02-24	78	improved. but only handles DB2 format
Dbase Interface	D00	2010-02-26	95	Okay. handles DB3 format
Dbase Interface	D00	2010-02-28	100	satisified
Dbase Interface	D01	2011-05-01	100	Okay ready for use
Pen driver	P01	2017-07-15	80	Okay ready for beta testing
Math unit	A01	2014-06-10	90	almost ready
Math unit	A02	2014-06-15	70	Accuracy problems!
Math unit	A02	2014-06-30	100	Okay problems fixed
Math unit	A02	2016-11-02	100	re-review for new employee to gain experience ...
Pen driver	P01	2017-08-15	60	needs rework, introduced new errors

```
select * from CONDUCT;
```

Result Grid | Filter Rows: | Expo

ID	Date
10100	2010-02-14
10200	2017-06-01
10100	2010-02-22
10100	2010-02-24
10100	2010-02-26
10100	2010-02-28
10200	2011-05-01
10300	2017-07-15
10100	2014-06-10
10100	2014-06-15
10100	2014-06-30
10700	2016-11-02
10400	2017-08-15

```
8 •  select * from INSPECTED;
```

Result Grid | Filter Rows:

Cname	Cversion	Date
Kevboard Driver	K11	2010-02-14
Touch Screen Driver	T00	2017-06-01
Dbase Interface	D00	2010-02-22
Dbase Interface	D00	2010-02-24
Dbase Interface	D00	2010-02-26
Dbase Interface	D00	2010-02-28
Dbase Interface	D01	2011-05-01
Pen driver	P01	2017-07-15
Math unit	A01	2014-06-10
Math unit	A02	2014-06-15
Math unit	A02	2014-06-30
Math unit	A02	2016-11-02
Pen driver	P01	2017-08-15

```
9 •  select * from COMPONENT;
```

Result Grid | Filter Rows: | Export: | Wrap

Cname	Cversion	Language	Size	Cstatus
Chart generator	C11	Java	6500	not-readv
Dbase Interface	D00	C++	2500	readv
Dbase Interface	D01	C++	2500	readv
Kevboard Driver	K11	C	1200	readv
Math unit	A01	C	5000	usable
Math unit	A02	Java	3500	readv
Pen driver	P01	C	3575	not-readv
Touch Screen Driver	T00	C++	4000	readv

Specially, when we use SELECT * FROM to the table COMPONENT. It is obvious that the status of 'Pen driver', 'P01' changes to 'not-ready'. Because the new inspection of this component is 'not-ready' status.

In addition, we know that 'Pen driver', 'P01', the component changed its status, is one of the component in product 'Excel', '2015'. Its project status is 'usable' at first. However, because of the change of component status, it will also be influenced. Because its component status becomes 'not-ready', the project status of 'Excel', '2015' becomes 'not-ready'. To ensure the result, we use SELECT * FROM as below:

```
10 •  select * from SOFTWARE_PRODUCT;
```

Result Grid | Filter Rows: | Export:

Pname	Pversion	Pstatus
Excel	2010	readv
Excel	2015	not-readv
Excel	2018beta	not-readv
Excel	secret	not-readv

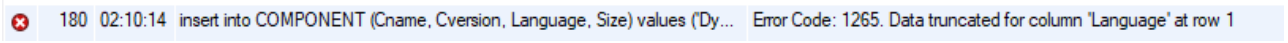
9. A) Demonstrate adding a new component to Excel 2018beta. This new component is named “Dynamic Table Interface”, version D01, and was written in javascript by person 10400, size = 775.

Insert a new component, according to my own database, 3 tables will be influenced including COMPONENT, HAVE, OWN.

First, we need to put the information into COMPONENT tables which contains the primary keys that HAVE and OWN uses as a foreign key. Thus, the table COMPONENT should be inserted first.

Then, the insertion of HAVE and OWN may be the next. HAVE table represents the relationship between the product and its components. OWN table represents the relationship between the component and its owner.

***** I find there is a problem that because I have defined that the Language should only be ‘C’, ‘C++’, ‘C#’, ‘Java’, or ‘PHP’, but ‘Javascript’ is not in the list, so it cannot be inserted successfully.**



To insert this component with ‘Javascript’ successfully, I modify the type of Language column from enum('C','C++','C#','Java','PHP') to enum('C','C++','C#','Java','PHP', 'Javascript').

```
1  alter table COMPONENT modify Language enum('C','C++','C#','Java','PHP', 'Javascript');
```

After this change, the insertion can be success.

```
1  insert into COMPONENT (Cname, Cversion, Language, Size) values ('Dynamic Table Interface', 'D01', 'Javascript', 775);
2  insert into HAVE values ('Excel', '2018beta', 'Dynamic Table Interface', 'D01');
3  insert into OWN values ('Dynamic Table Interface', 'D01', 10400);
```

```
18  select * from COMPONENT;
```

Result Grid | Filter Rows: | Export: | Wrap Ce

Cname	Cversion	Language	Size	Cstatus
Chart generator	C11	Java	6500	not-readv
Dbase Interface	D00	C++	2500	readv
Dbase Interface	D01	C++	2500	readv
Dynamic Table Interface	D01	Javascript	755	not-readv
Keyboard Driver	K11	C	1200	readv
Math unit	A01	C	5000	usable
Math unit	A02	Java	3500	readv
Pen driver	P01	C	3575	not-readv
Touch Screen Driver	T00	C++	4000	readv

In the table COMPONENT, because this new component has not been inspected, so its status is ‘not-ready’.

```
19 • select * from HAVE;
```

Result Grid | Filter Rows: | Expor

Pname	Pversion	Cname	Cversion
Excel	2010	Kevboard Driver	K11
Excel	2010	Dbase Interface	D00
Excel	2015	Kevboard Driver	K11
Excel	2015	Dbase Interface	D01
Excel	2015	Pen driver	P01
Excel	2018beta	Kevboard Driver	K11
Excel	2018beta	Touch Screen Driver	T00
Excel	2018beta	Chart oenerator	C11
Excel	secret	Kevboard Driver	K11
Excel	secret	Touch Screen Driver	T00
Excel	secret	Chart oenerator	C11
Excel	secret	Math unit	A02
Excel	2018beta	Dvnamic table Inter...	D01

```
20 • select * from OWN;
```

Result Grid | Filter Rows:

Cname	Cversion	ID
Kevboard Driver	K11	10100
Touch Screen Driver	T00	10100
Dbase Interface	D00	10200
Dbase Interface	D01	10300
Chart oenerator	C11	10200
Pen driver	P01	10700
Math unit	A01	10200
Math unit	A02	10200
Dvnamic Table Interface	D01	10400

B) What is the Excel 2018beta product status?

According to A), the new component will influence the column Pstatus which represents the status of Product, because Excel 2018beta has a new component which has not been inspected yet. It means the status of new component is 'not-ready', so the status of Excel 2018beta should also be 'not-ready'.

```
32 • select Pstatus from SOFTWARE_PRODUCT where Pname = 'Excel' and Pversion = '2018beta';
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Pstatus
not-readv

From the capture, we can see that Excel 2018beta changed its status to 'not-ready'.

10. A) Demonstrate the adding of an inspection on the component you just added. This inspection occurred on 11/20/2017 by inspector 10500, with a score of 80, and note of "minor fixes needed".

Insert a new inspection, according to my own database, I need to write 3 INSERT clauses to complete the adding process.

First, we need to put the information into INSPECTION tables, which may influence the status of the component - 'Dynamic Table Interface', 'D01'.

Then, when this insertion complete, the relational table CONDUCT which represents the relationship between the employee and the date when his/her conducted because date can be the key. INSPECTED will also be influenced, is shows the relationship between the component and its date of inspection.

```
insert into INSPECTION values ('Dynamic Table Interface', 'D01', '2017-11-20', 80, 'minor fixes needed');
insert into CONDUCT values (10500, '2017-11-20');
insert into INSPECTED values ('Dynamic Table Interface', 'D01', '2017-11-20');
```

After the insertion, the status of the new component 'Dynamic Table Interface' 'D01' will be changed. Before the insertion, because this component had not been inspected, its status is 'not-ready'. From the information we have, the score of this inspection is 80, which is equal to the status 'usable'. Thus, the component status (Cstatus) of this new component shown in table COMPONENT will change its status to 'usable'. Using SELECT * FROM COMPONENT to ensure the result.

```
select * from COMPONENT;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Cor

Cname	Cversion	Language	Size	Cstatus
Chart generator	C11	Java	6500	not-ready
Dbase Interface	D00	C++	2500	ready
Dbase Interface	D01	C++	2500	ready
Dynamic Table Interface	D01	Javascript	755	usable
Keyboard Driver	K11	C	1200	ready
Math unit	A01	C	5000	usable
Math unit	A02	Java	3500	ready
Pen driver	P01	C	3575	not-ready
Touch Screen Driver	T00	C++	4000	ready

The change of other tables shows below:

```
select * from INSPECTION;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Cname	Cversion	Date	Score	Description
Keyboard Driver	K11	2010-02-14	100	legacy code which is already approved
Touch Screen Driver	T00	2017-06-01	95	initial release ready for usage
Dbase Interface	D00	2010-02-22	55	too many hard coded parameters. the software...
Dbase Interface	D00	2010-02-24	78	improved. but only handles DB2 format
Dbase Interface	D00	2010-02-26	95	Okay. handles DB3 format
Dbase Interface	D00	2010-02-28	100	satisified
Dbase Interface	D01	2011-05-01	100	Okay ready for use
Pen driver	P01	2017-07-15	80	Okay ready for beta testing
Math unit	A01	2014-06-10	90	almost ready
Math unit	A02	2014-06-15	70	Accuracy problems!
Math unit	A02	2014-06-30	100	Okay problems fixed
Math unit	A02	2016-11-02	100	re-review for new employee to gain experience ...
Pen driver	P01	2017-08-15	60	needs rework. introduced new errors
Dynamic Table Inte...	D01	2017-11-20	80	minor fixes needed

```
30 • select * from CONDUCT;
```

Result Grid | Filter Rows:

ID	Date
10100	2010-02-14
10200	2017-06-01
10100	2010-02-22
10100	2010-02-24
10100	2010-02-26
10100	2010-02-28
10200	2011-05-01
10300	2017-07-15
10100	2014-06-10
10100	2014-06-15
10100	2014-06-30
10700	2016-11-02
10400	2017-08-15
10500	2017-11-20

```
31 • select * from INSPECTED;
```

Result Grid | Filter Rows:

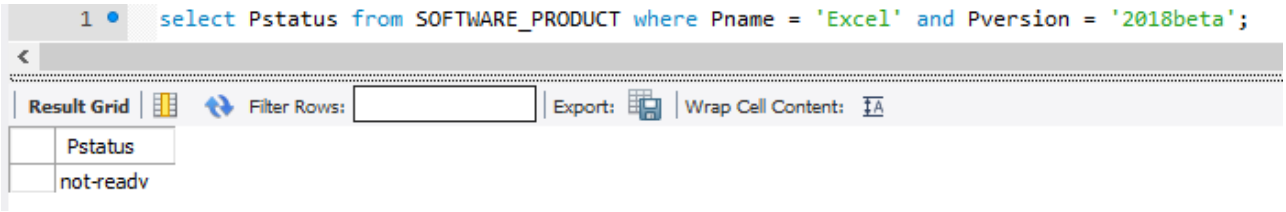
Cname	Cversion	Date
Kevboard Driver	K11	2010-02-14
Touch Screen Driver	T00	2017-06-01
Dbase Interface	D00	2010-02-22
Dbase Interface	D00	2010-02-24
Dbase Interface	D00	2010-02-26
Dbase Interface	D00	2010-02-28
Dbase Interface	D01	2011-05-01
Pen driver	P01	2017-07-15
Math unit	A01	2014-06-10
Math unit	A02	2014-06-15
Math unit	A02	2014-06-30
Math unit	A02	2016-11-02
Pen driver	P01	2017-08-15
Dynamic Table Inte...	D01	2017-11-20

B) What is the Excel 2018beta product status?

According to the A), a new inspection of this new component which has been added to the Product Excel 2018beta changed its own status. Because the status of the product is influenced by components this product has, the Excel 2018beta may also change its product status.

Before the adding of this new inspection, the product status of Excel 2018beta is 'not-ready', which means when this inspection let one of the component status change its status from 'not-ready' to 'usable', it may change the status of the Excel 2018beta from 'not-ready' to 'usable' if there is no other component which is in Excel 2018beta with the status 'not-ready'. However, from the initial information, one of the component – 'Chart generator' in Excel 2018beta is still in 'not-ready' status, which means Excel 2018beta cannot change to other status though 'Dynamic Table Interface' changes its status to 'usable'.

Thus, when we use SELECT clause, the Excel 2018beta is still in 'not-ready' status.



GRADUATE:

11. Person 10700 has decided to leave *OSF* for other employment. Implement a solution for this situation.

When Person 10700 leaves OSF, the component and its inspection which related to 10700 should also be influenced.

First, about the component relates to 10700. In my opinion, the component the 10700 owns should be changed because Person 10700 is no longer in the OSF. I suggest that because each component needs an owner, and each owner may have a person who manage him/her (except 10100), the component which is owned by Person 10700 can change its owner to the person who manages 10700 (From the table Employee, we know the manager of 10700 is 10400). It can be the solution for the changing of owner. The component which will be influenced is 'Pen driver', 'P01'.

Then, about the inspection relates to 10700. In my opinion, although Person 10700 leaves OSF, the record of this person's inspection should also be reserved in the database. However, although Person 10700 leaves OSF, his ID isn't change, which means other person who will be hired in the OSF will not occupy the ID 10700. Thus, the row of Person 10700 in table CONDUCT will not be removed or modified. I decide to create a new table called LEAVEOSF which represents the person who leaves OSF, the value can only be inserted into this table when the value has been deleted in the table EMPLOYEE. I can use a trigger to make this process execute successfully.

There is a trigger called "Move_to_LEAVE". It triggers after the deletion of the row of 10700 in the table EMPLOYEE, then the value contains "ID" (for this question ID = 10700), "hire date", "leave date" into the table LEAVEOSF automatically by this trigger. However, a foreign key "ID" in table CONDUCT has been defined, which may cause the deletion of row of ID = 10700 in the table EMPLOYEE fails because CONDUCT still contain the row of ID = 10700. To solve this problem, I modify the table CONDUCT that let the column ID no longer be a foreign key references EMPLOYEE. To ensure the ID in CONDUCT is correctly, I remove the foreign key constraint and create other triggers have the similar function. It is called "CONDUCT_INS" for insertion of CONDUCT, "CONDUCT_DEL" for insertion, and "CONDUCT_UPD" for update of CONDUCT.

From the explanation above, here is the process of remove the information of 10700:

Step 1: Change the owner of 'Pen driver', 'P01' which used to belong to 10700 to 10400.

```
update OWN
set ID = 10400
where Cname = 'Pen driver' and Cversion = 'P01';
```

Result of Step 1:

```
select * from OWN;
```

Result Grid | Filter Rows:

Cname	Cversion	ID
Kevboard Driver	K11	10100
Touch Screen Driver	T00	10100
Dbase Interface	D00	10200
Dbase Interface	D01	10300
Chart generator	C11	10200
Pen driver	P01	10400
Math unit	A01	10200
Math unit	A02	10200
Dynamic Table Interface	D01	10400
NULL	NULL	NULL

Step 2: Remove the foreign key "ID" in table CONDUCT. Because table CONDUCT can be dropped directly and recreated.

```
create table CONDUCT
(
ID integer not null,
Date date not null,
foreign key(Date) references INSPECTION(Date)
);
```

```
insert into CONDUCT values (10100, '2010-02-14');
insert into CONDUCT values (10200, '2017-06-01');
insert into CONDUCT values (10100, '2010-02-22');
insert into CONDUCT values (10100, '2010-02-24');
insert into CONDUCT values (10100, '2010-02-26');
insert into CONDUCT values (10100, '2010-02-28');
insert into CONDUCT values (10200, '2011-05-01');
insert into CONDUCT values (10300, '2017-07-15');
insert into CONDUCT values (10100, '2014-06-10');
insert into CONDUCT values (10100, '2014-06-15');
insert into CONDUCT values (10100, '2014-06-30');
insert into CONDUCT values (10700, '2016-11-02');
```

Then recreate the table CONDUCT without the foreign key ID.

However, the column of ID should still have constraint because it is related to the table EMPLOYEE and new table LEAVEOSF.

Step 3: Create a new table LEAVEOSF to store the information of person who decides to leave OSF.

```
create table LEAVEOSF
(
ID integer not null,
Hiredate date not null,
Leavedate date not null
);
```

Step 4: Because the insertion of LEAVEOSF should after the deletion of EMPLOYEE, create a trigger "Move_to_LEAVE".

```
delimiter //
create trigger Move_to_LEAVE after delete on EMPLOYEE for each row
begin
    set @ID = old.ID;
    set @Hiredate = old.Hiredate;
    set @Leavedate = now();
    insert into LEAVEOSF values (@ID, @Hiredate, @Leavedate);
end
//
delimiter ;
```

Result of Step 4:

If delete the row of ID = 10700 in table EMPLOYEE, the result shows below:

```
delete from EMPLOYEE where ID = 10700;
```

```
select * from EMPLOYEE;
```

Result Grid | Filter Rows: | Edit:

ID	Ename	Hiredate	Mgr	Seniority
10100	Employee-1	1984-11-08	NULL	senior
10200	Employee-2	1994-11-08	10100	senior
10300	Employee-3	2004-11-08	10200	senior
10400	Employee-4	2008-11-01	10200	senior
10500	Employee-5	2015-11-01	10400	junior
10600	Employee-6	2015-11-01	10400	junior
10800	Employee-8	2017-11-01	10200	newbie
NULL	NULL	NULL	NULL	NULL

ID = 10700 is no longer in table EMPLOYEE;

```
select * from LEAVEOSF;
```

Result Grid | Filter Rows: | Export:

ID	Hiredate	Leavedate
10700	2016-11-01	2017-12-04

ID = 10700 automatically added into new table EMPLOYEE by trigger.

Check the table CONDUCT which should not be changed.

```
select * from CONDUCT;
```

Result Grid | Filter Rows:

ID	Date
10100	2010-02-14
10200	2017-06-01
10100	2010-02-22
10100	2010-02-24
10100	2010-02-26
10100	2010-02-28
10200	2011-05-01
10300	2017-07-15
10100	2014-06-10
10100	2014-06-15
10100	2014-06-30
10700	2016-11-02
NULL	NULL

Step 5: Create 3 triggers in table CONDUCT to replace the function of foreign key constraint.

Trigger CONDUCT_INS:

```
delimiter //
create trigger CONDUCT_INS before insert on CONDUCT for each row
begin
    if new.ID not in (select ID from EMPLOYEE) and new.ID not in (select ID from LEAVEOSF) then
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = "Incorrect ID insert, ID not in EMPLOYEE or LEAVEOSF";
    end if;
end
//
delimiter ;
```

Check:

```
insert into CONDUCT values (11111, '2010-02-14');
```

```
136  21:19:14  insert into CONDUCT values (11111, '2010-02-14')    Error Code: 1644. Incorrect ID insert, ID not in EMPLOYEE or LEAVEOSF
```

Trigger CONDUCT_UPD:

```
delimiter //
create trigger CONDUCT_UPD before update on CONDUCT for each row
begin
    if new.ID not in (select ID from EMPLOYEE) and new.ID not in (select ID from LEAVEOSF) then
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = "Incorrect ID insert, ID not in EMPLOYEE or LEAVEOSF";
    end if;
end
//
delimiter ;
```

Check:

```
update CONDUCT
set ID = 11111
where date = '2010-02-14';
```

```
137  21:20:22  update CONDUCT set ID = 11111 where date = '2010-02-14'    Error Code: 1644. Incorrect ID insert, ID not in EMPLOYEE or LEAVEOSF
```

Trigger CONDUCT_DEL:

```
delimiter //
create trigger CONDUCT_DEL before delete on CONDUCT for each row
begin
    SIGNAL SQLSTATE '45000'
    SET MESSAGE_TEXT = "INSPECTION information cannot be deleted";
end
//
delimiter ;
```

Check:

```
delete from CONDUCT where ID = 10100;
```

```
140  21:24:31  delete from CONDUCT where ID = 10100    Error Code: 1644. INSPECTION information cannot be deleted
```

Now finish the process of the situation when 10700 decides to leave OSF. The component owned by 10700 changes its owner to the person who manages 10700 (10400). The inspection conducted by 10700 is still reserved in the table INSPECTION and CONDUCT. In addition, when Person 10700 leaves 10700, the information of 10700 will be moved from table EMPLOYEE to a new table LEAVEOSF for leaving people. The movement works by trigger.

DBMS Choosing: MySQL
Version: 5.7
(END)