|  |  |
| --- | --- |
|  | TOSHKENT SHAHRIDAGI INHA UNIVERSITETI  INHA UNIVERSITY IN TASHKENT |

**Database (SOC3020), Fall 2017**

**Homework 2, Group 004**

|  |  |  |
| --- | --- | --- |
| Maftuna Sharabbaeva  U1510067 | Akhror Gaibnazarov  U1510005 | Azizbek  Tursunov  U1510009 |

**11/9/2017**

**Answer for question #1:**

**(a)**I used phpMyAdmin which is in my Xamp to build a database which has name as “dhomework”. I created it by uploading ‘DDL.sql’ and ‘smallRelationsInsertFile.sql’ filles. I corrected all errors, previous homework and , now I am going to use it. Here tables:

create table classroom

(building varchar(15), room\_number varchar(7),

capacity numeric(4,0),

primary key (building, room\_number)

);

create table department

(dept\_name varchar(20), building varchar(15),

budget numeric(12,2) check (budget > 0),

primary key (dept\_name)

);

create table course

(course\_id varchar(8), title varchar(50), dept\_name varchar(20),

credits numeric(2,0) check (credits > 0), primary key (course\_id),

foreign key (dept\_name) references department **(dept\_name)**

on delete set null

);

create table instructor

|  |  |  |  |
| --- | --- | --- | --- |
|  | (ID |  | varchar(5), |
|  | name |  | varchar(20) not null, |
|  | dept\_name |  | varchar(20), |
|  | salary |  | numeric(8,2) check (salary > 29000), |

primary key (ID),

foreign key (dept\_name) references department **(dept\_name)**

on delete set null

);

create table section

(course\_id varchar(8), sec\_id varchar(8), semester varchar(6)

check (semester in ('Fall', 'Winter', 'Spring', 'Summer')),

year numeric(4,0) check (year > 1701 and year < 2100), building varchar(15), room\_number varchar(7), time\_slot\_id varchar(4),

primary key (course\_id, sec\_id, semester, year),

foreign key (course\_id) references course **(course\_id)**

on delete cascade,

foreign key (building, room\_number) references classroom **(building, room\_number)**

on delete set null

);

create table teaches

|  |  |  |  |
| --- | --- | --- | --- |
|  | (ID |  | varchar(5), |
|  | course\_id |  | varchar(8), |
|  | sec\_id |  | varchar(8), |
|  | semester |  | varchar(6), |
|  | year |  | numeric(4,0), |

primary key (ID, course\_id, sec\_id, semester, year),

foreign key (course\_id,sec\_id, semester, year) references section **(course\_id,sec\_id, semester, year)**

on delete cascade,

foreign key (ID) references instructor **(ID)**

on delete cascade

);

create table student

|  |  |  |  |
| --- | --- | --- | --- |
|  | (ID |  | varchar(5), |
|  | name |  | varchar(20) not null, |
|  | dept\_name |  | varchar(20), |
|  | tot\_cred |  | numeric(3,0) check (tot\_cred >= 0), |

primary key (ID),

foreign key (dept\_name) references department **(dept\_name)** on delete set null

);

create table takes

(ID varchar(5),

course\_id varchar(8),

|  |  |  |
| --- | --- | --- |
|  | sec\_id | varchar(8), |
|  | semester | varchar(6), |
|  | year | numeric(4,0), |
|  | grade | varchar(2), |

primary key (ID, course\_id, sec\_id, semester, year),

foreign key (course\_id,sec\_id, semester, year) references section **(course\_id,sec\_id, semester, year)**

on delete cascade,

foreign key (ID) references student **(ID)**

on delete cascade

);

create table advisor

(s\_ID varchar(5), i\_ID varchar(5),

primary key (s\_ID),

foreign key (i\_ID) references instructor **(ID)**

on delete set null,

foreign key (s\_ID) references student **(ID)**

on delete cascade

);

create table time\_slot

|  |  |  |  |
| --- | --- | --- | --- |
|  | (time\_slot\_id |  | varchar(4), |
|  | day |  | varchar(1), |
|  | start\_hr |  | numeric(2) check (start\_hr >= 0 and start\_hr < 24), |
|  | start\_min |  | numeric(2) check (start\_min >= 0 and start\_min < 60), |
|  | end\_hr |  | numeric(2) check (end\_hr >= 0 and end\_hr < 24), |
|  | end\_min |  | numeric(2) check (end\_min >= 0 and end\_min < 60), |

primary key (time\_slot\_id, day, start\_hr, start\_min)

);

create table prereq

(course\_id varchar(8), prereq\_id varchar(8),

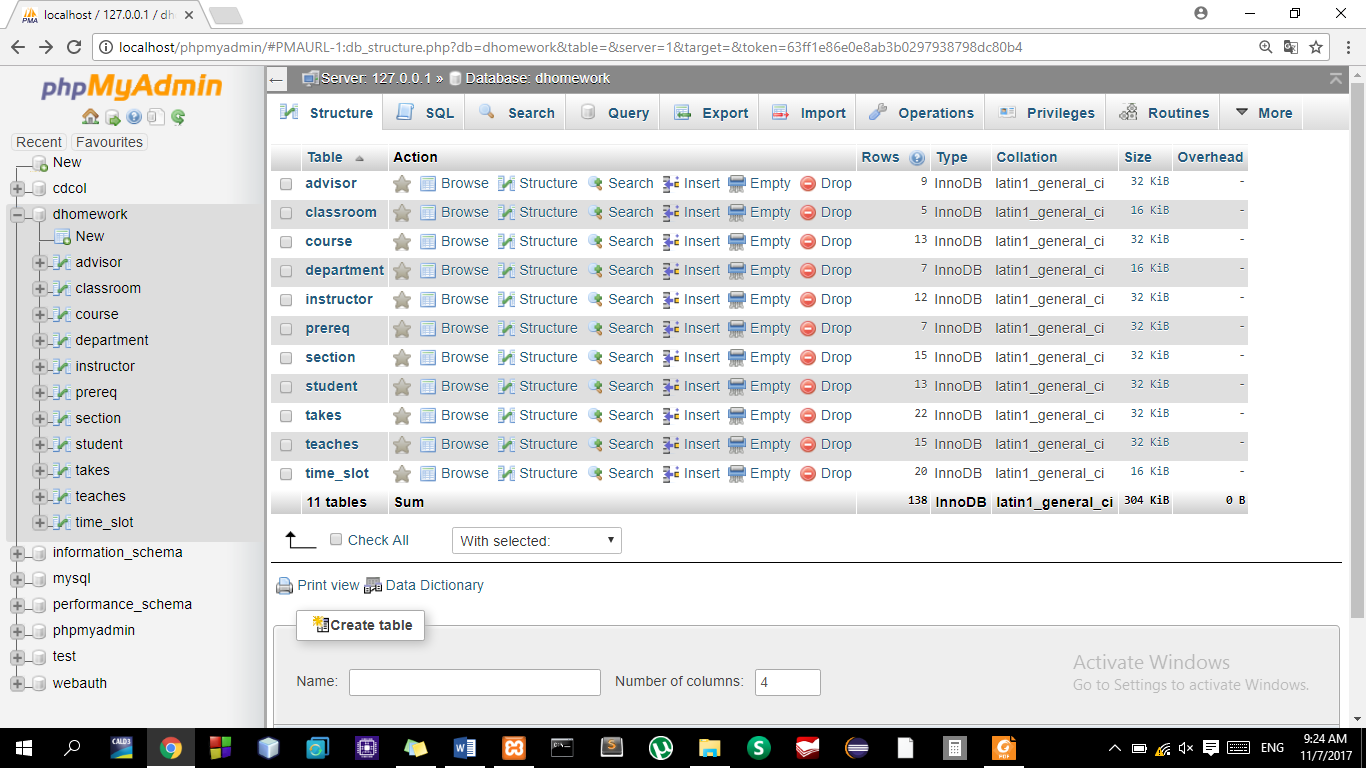
primary key (course\_id, prereq\_id),

foreign key (course\_id) references course **(course\_id)** on delete cascade,

foreign key (prereq\_id) references course **(course\_id)**

);

After correcting them result is:

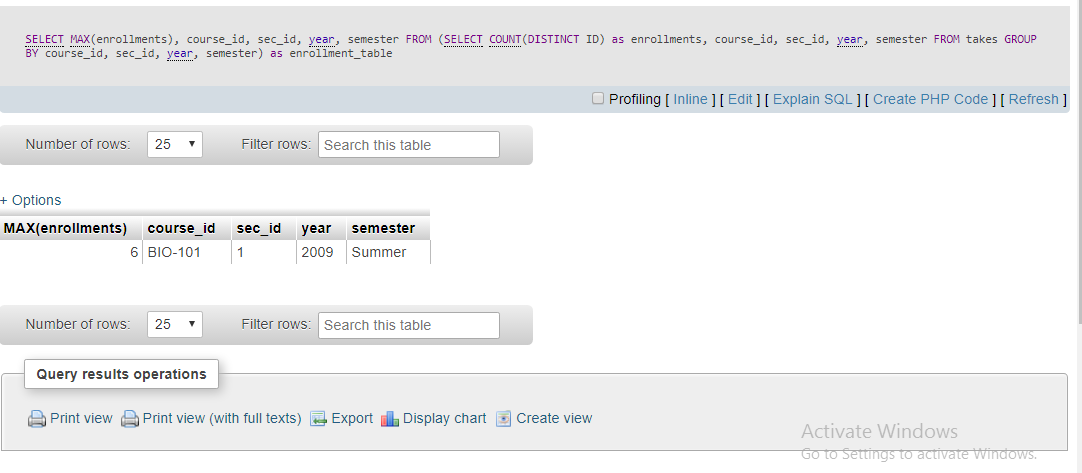


**(b) 1.1**  SELECT MAX(enrollments), course\_id, sec\_id, year, semester

FROM (SELECT COUNT(DISTINCT ID) as enrollments, course\_id, sec\_id, year,semester

FROM takes

GROUP BY course\_id, sec\_id, year, semester) as enrollment\_table



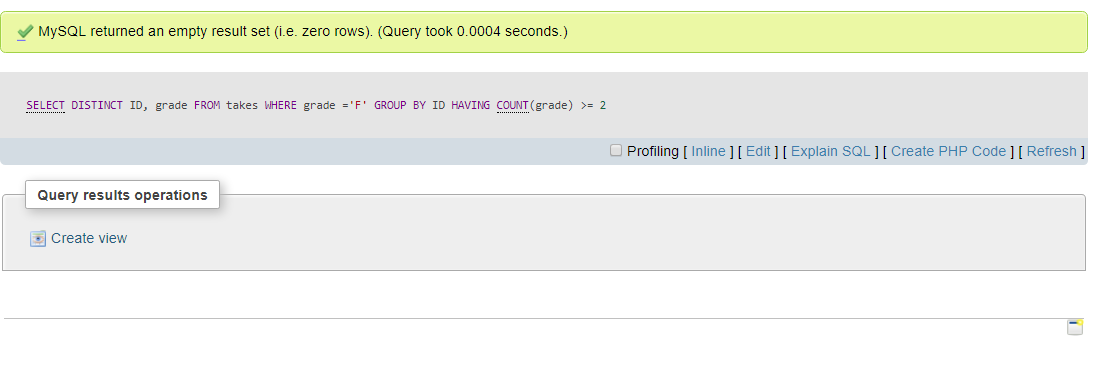
**1.2**  SELECT DISTINCT ID, grade

FROM takes

WHERE grade ='F'

GROUP BY ID

HAVING COUNT(grade) >= 2



Here I got empty result , because there is not any student who got F in tables .

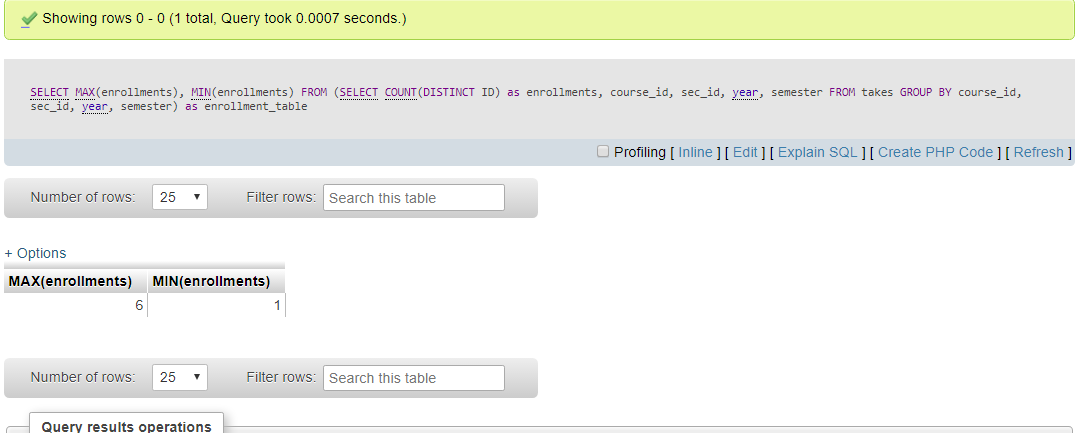
**1.4**

SELECT MAX(enrollments), MIN(enrollments)

FROM (SELECT COUNT(DISTINCT ID) as enrollments, course\_id, sec\_id, year, semester

FROM takes

GROUP BY course\_id, sec\_id, year, semester) as enrollment\_table



**1.6 a**

(SELECT course\_id, sec\_id, year, semester, COUNT(ID)

FROM takes as

GROUP BY (course\_id, sec\_id, year, semester,0)

UNION

(SELECT course\_id, sec\_id, year, semester, 0)

FROM section

WHERE (course\_id, sec\_id, year, semester, 0) not in (SELECT course\_id, sec\_id, year, semester, 0 FROM takes))

**1.6 b**

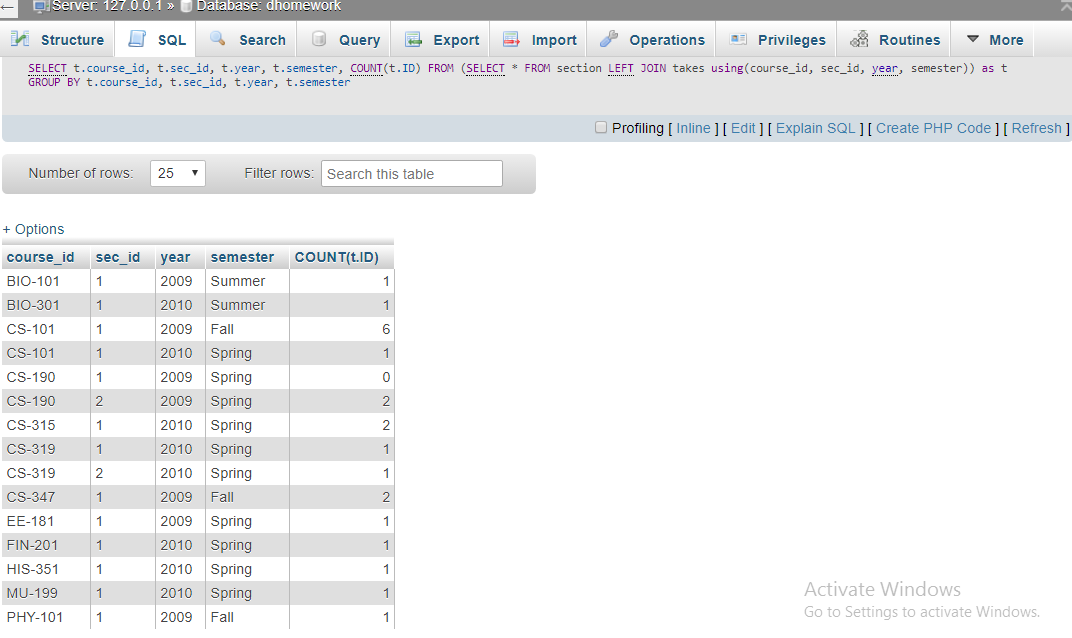
SELECT t.course\_id, t.sec\_id, t.year, t.semester, COUNT(t.ID)

FROM (SELECT \*

FROM section LEFT JOIN takes

using(course\_id, sec\_id, year, semester)) as t

GROUP BY t.course\_id, t.sec\_id, t.year, t.semester



1.7

SELECT course\_id, title

FROM course

WHERE course\_id LIKE 'CS-%'



1.8

SELECT \*

FROM teaches as AllTeachers

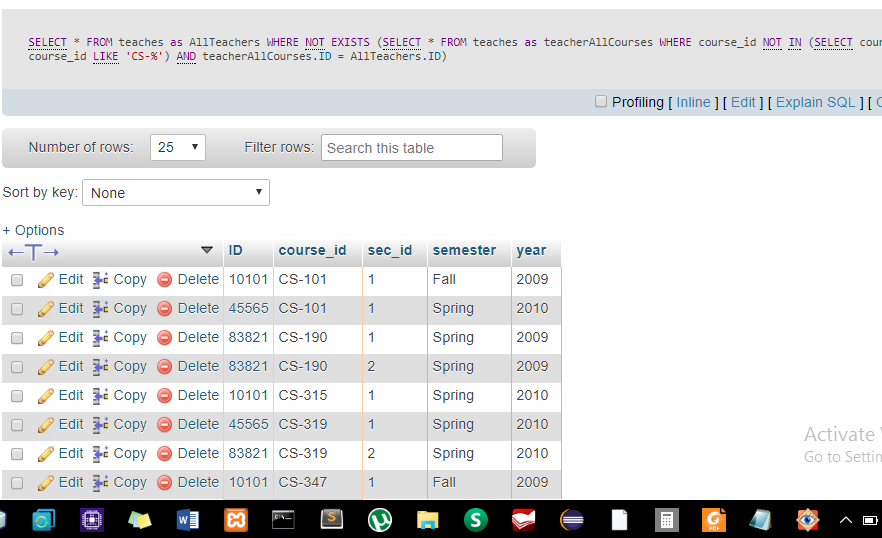
WHERE NOT EXISTS (SELECT \*

FROM teaches as teacherAllCourses

WHERE course\_id NOT IN (SELECT course\_id

FROM course WHERE course\_id LIKE 'CS-%')

AND teacherAllCourses.ID = AllTeachers.ID)



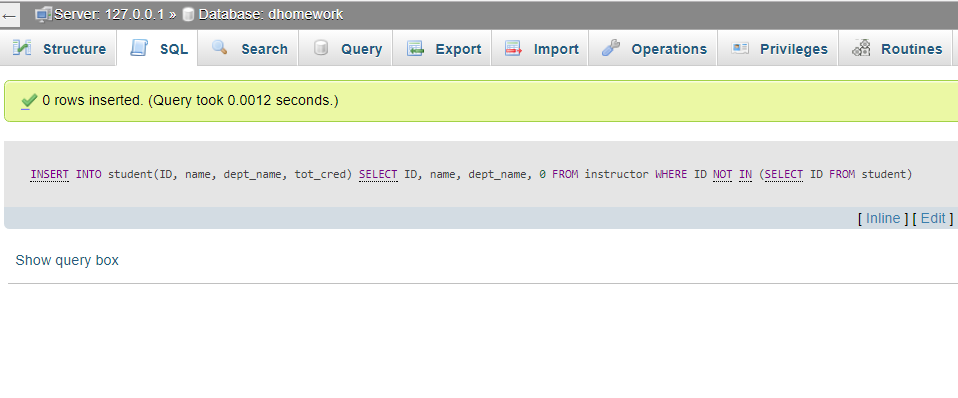
1.9

INSERT INTO student(ID, name, dept\_name, tot\_cred)

SELECT ID, name, dept\_name, 0

FROM instructor

WHERE ID NOT IN (SELECT ID FROM student)

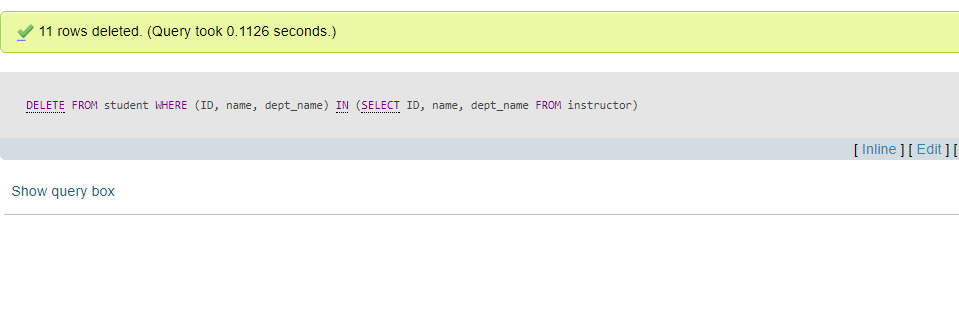


1.10

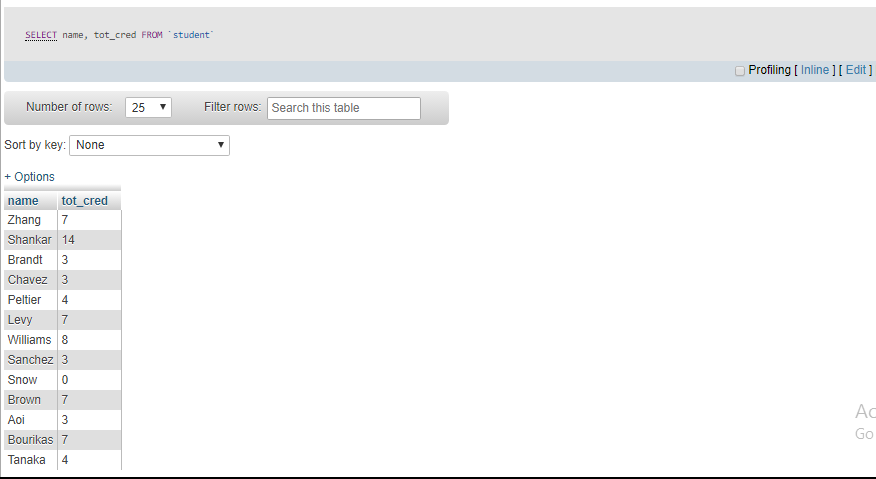
DELETE FROM student

WHERE (ID, name, dept\_name)

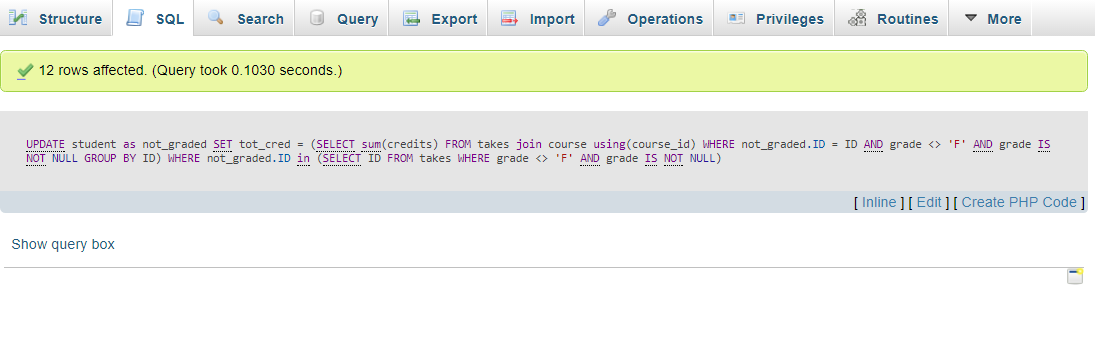
IN (SELECT ID, name, dept\_name FROM instructor)



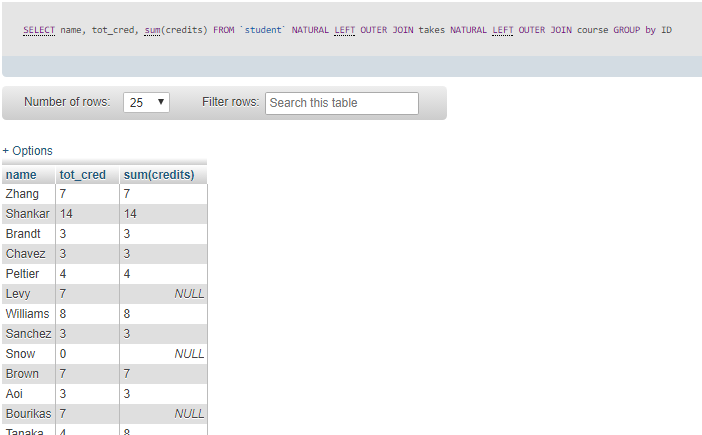
2 a) As I mentioned before, I already corrected both .sql files

2.1 SELECT name, tot\_cred FROM `student` 

WHERE grade <> 'F' AND grade IS NOT NULL)



SELECT name, tot\_cred, sum(credits) FROM `student` NATURAL LEFT OUTER JOIN takes NATURAL LEFT OUTER JOIN course GROUP by ID



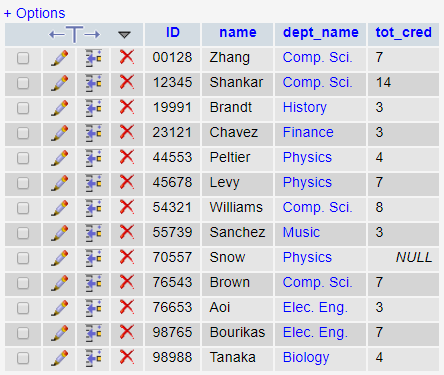
2.2

UPDATE student as new set tot\_cred= (SELECT SUM(credits) FROM

takes NATURAL LEFT OUTER JOIN course

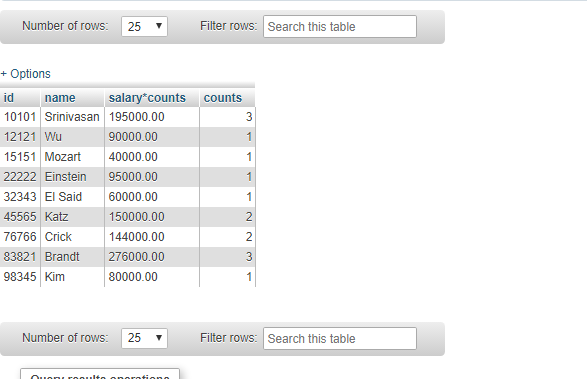
WHERE new.ID= takes.ID AND takes.grade<> 'F' AND takes.grade is NOT null

)



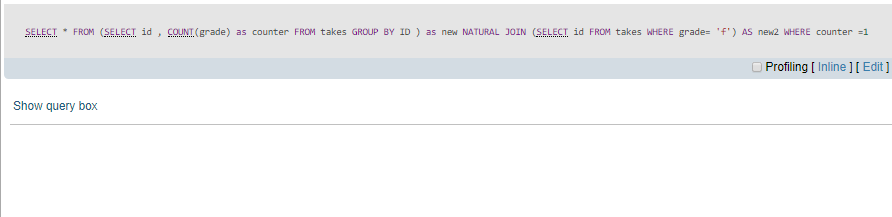
2.3

SELECT id , name , salary\*counts ,counts FROM (SELECT id , name , COUNT(sec\_id) as counts, salary FROM instructor NATURAL JOIN teaches group by ID) as new

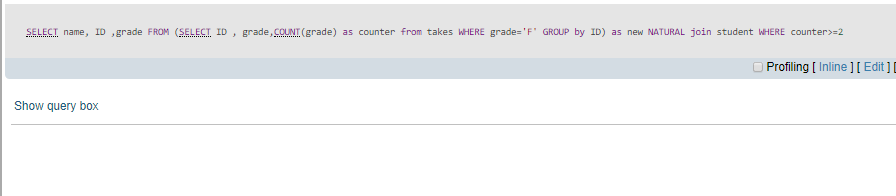


2.5

SELECT \* FROM (SELECT id , COUNT(grade) as counter FROM takes GROUP BY ID ) as new NATURAL JOIN (SELECT id FROM takes WHERE grade= 'f') AS new2 WHERE counter =1

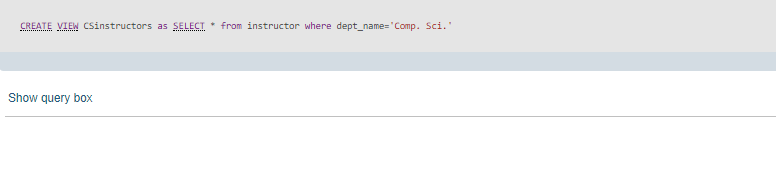


2.6 SELECT name, ID ,grade FROM (SELECT ID , grade,COUNT(grade) as counter from takes WHERE grade='F' GROUP by ID) as new NATURAL join student WHERE counter>=2



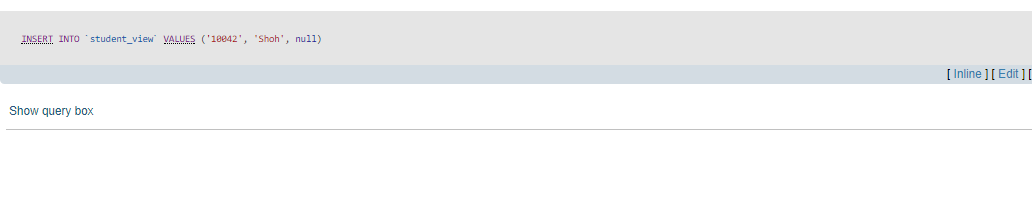
2.7 CREATE VIEW student\_view as SELECT ID, name, dept\_name FROM student



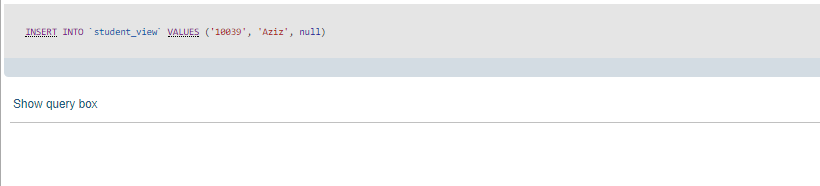
2.8 CREATE VIEW CSinstructors as SELECT \* from instructor where dept\_name='Comp. Sci.'

2.9

INSERT INTO `student\_view` VALUES ('10042', 'Shoh', null)

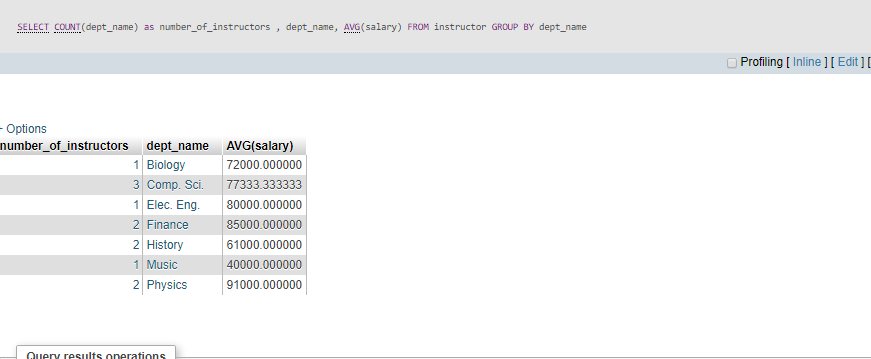


2.9 INSERT INTO `student\_view` VALUES ('10039', 'Aziz', null)



2.10

SELECT COUNT(dept\_name) as number\_of\_instructors , dept\_name, AVG(salary) FROM instructor GROUP BY dept\_name



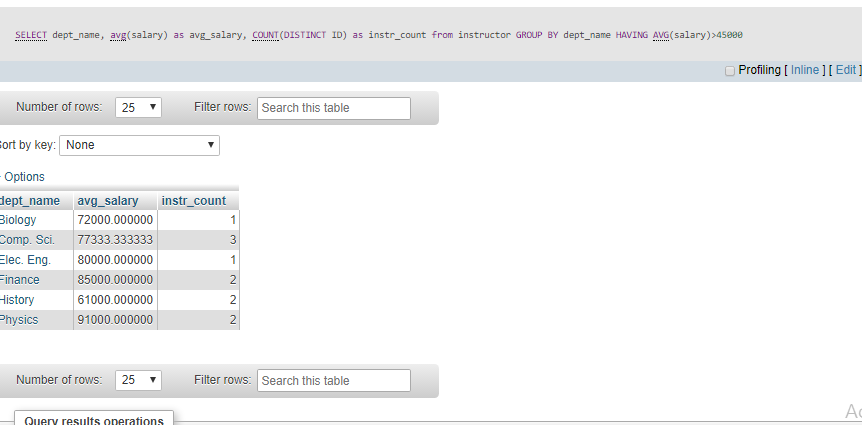
**3**

SELECT dept\_name, avg(salary) as avg\_salary, COUNT(DISTINCT ID) as instr\_count

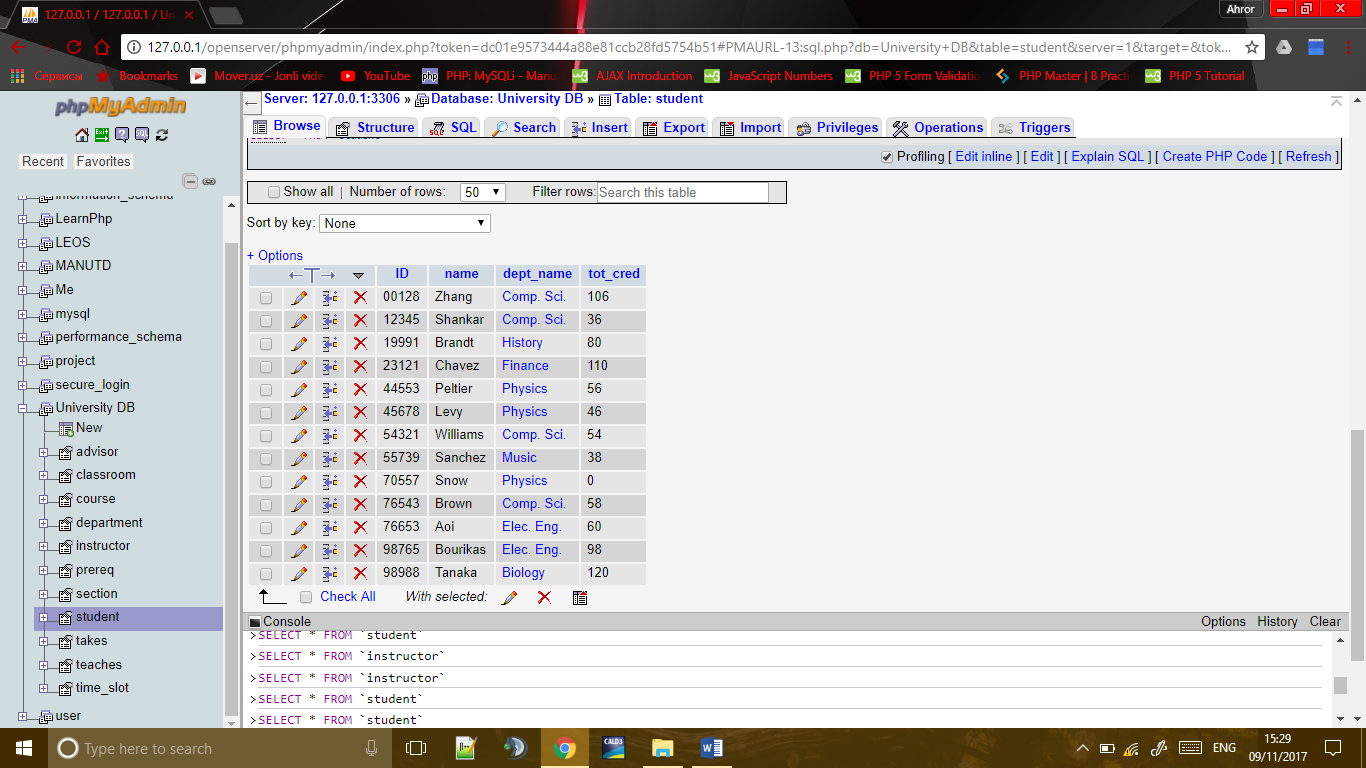
from instructor

GROUP BY dept\_name

HAVING AVG(salary)>45000



4 a) INSERT INTO instructor SELECT ID, name, dept\_name, 20000 FROM student WHERE dept\_name=’Music’ AND tot\_cred>30



INSERT INTO instructor SELECT ID, name, dept\_name, 20000 FROM student WHERE dept\_name=’Finance’ AND tot\_cred>80

b) In the first example , we took data from instructor table and inserted in student table in Comp.Sci department , and in the second example in Financial department

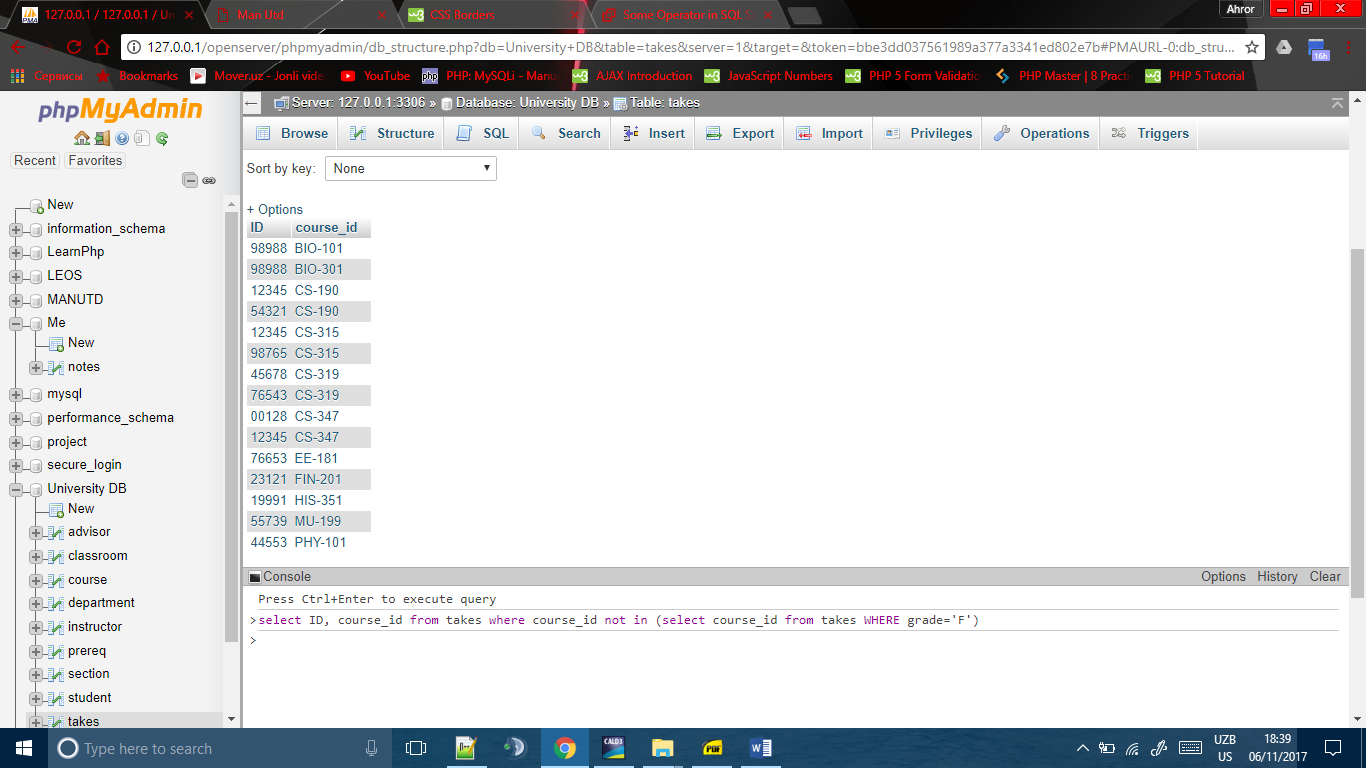
**Exercise 5**

1. select ID,course\_id from takes where course\_id <> all (select course\_id from takes where grade='F')
2. Select id, course\_id from takes where course\_id not in (select course\_id from takes where grade ='F')
3. Both queries are taking input from exactly same relation: **takes**;
4. Yes, done. Mentioned in below.

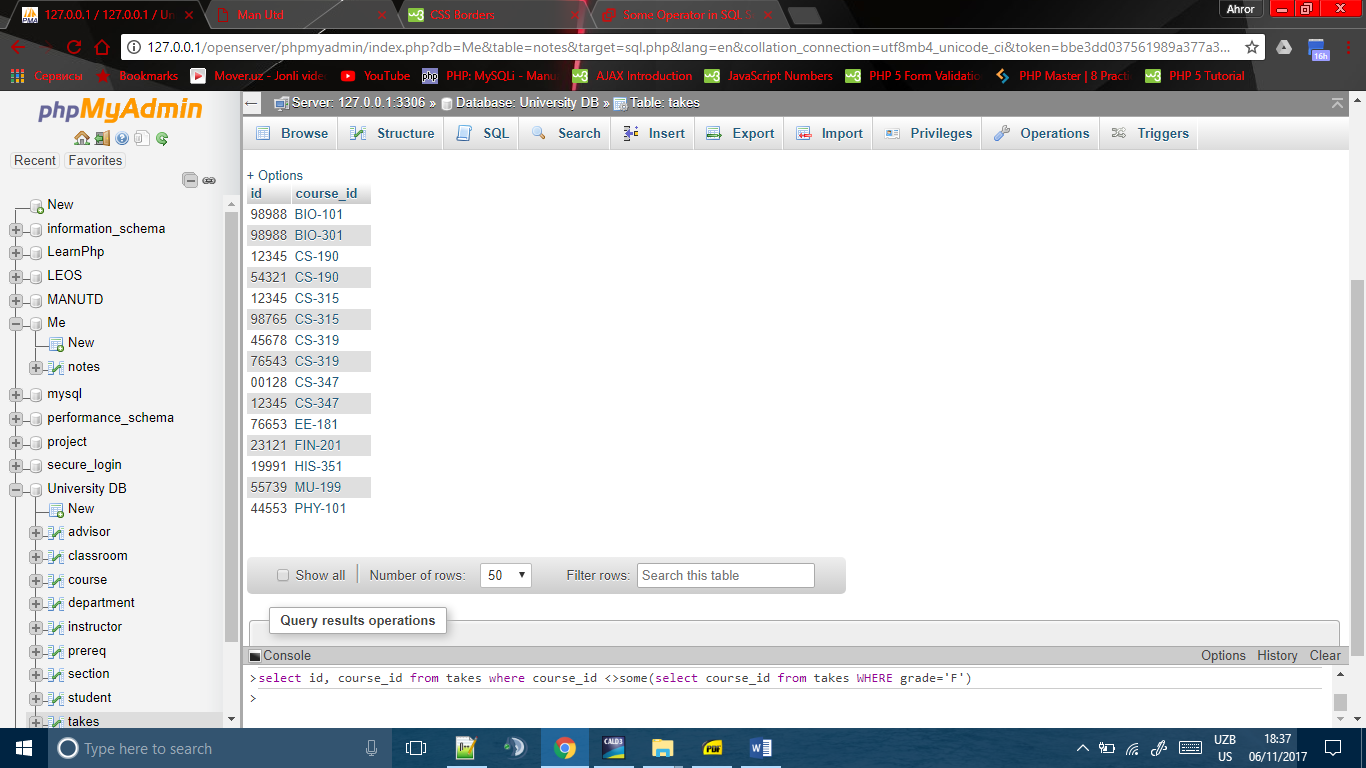
**Then:**

1. **Q1**: find the IDs and course\_ids of all students who did not study with some students who took ‘F’ in one course. **Q2**: find the IDs and course\_ids of all students who did not study with students who took ‘F’ in one course.
2. **1. Q1:** select ID,course\_id from takes where course\_id <> all (select course\_id from takes where grade='F')

**Q2:** Select id, course\_id from takes where course\_id not in (select course\_id from takes where grade ='F')

**2.Q1:**

**Q2:**



**3.** Q1 and Q2 does not give same result in all cases. In our case, they are giving same results, because student who took ‘F’ is one. If we take, who took grade ‘C’ it will be very different result between Q1 and Q2. Because, there are two students who took ‘C’ and using **some** gives we can compare our variable to not all result of subquery. In Q2 we used **not** **in**. It means variable will be compared to all results and we take if our variable has not any same result in results. It means, **not in** is equal to **<>all** and not equal to **<>some**.

**Exercise 6**

1. select ID,course\_id from takes where course\_id = all (select course\_id from takes where grade='F')

**b)** Select id, course\_id from takes where course\_id in (select course\_id from takes where grade ='F')

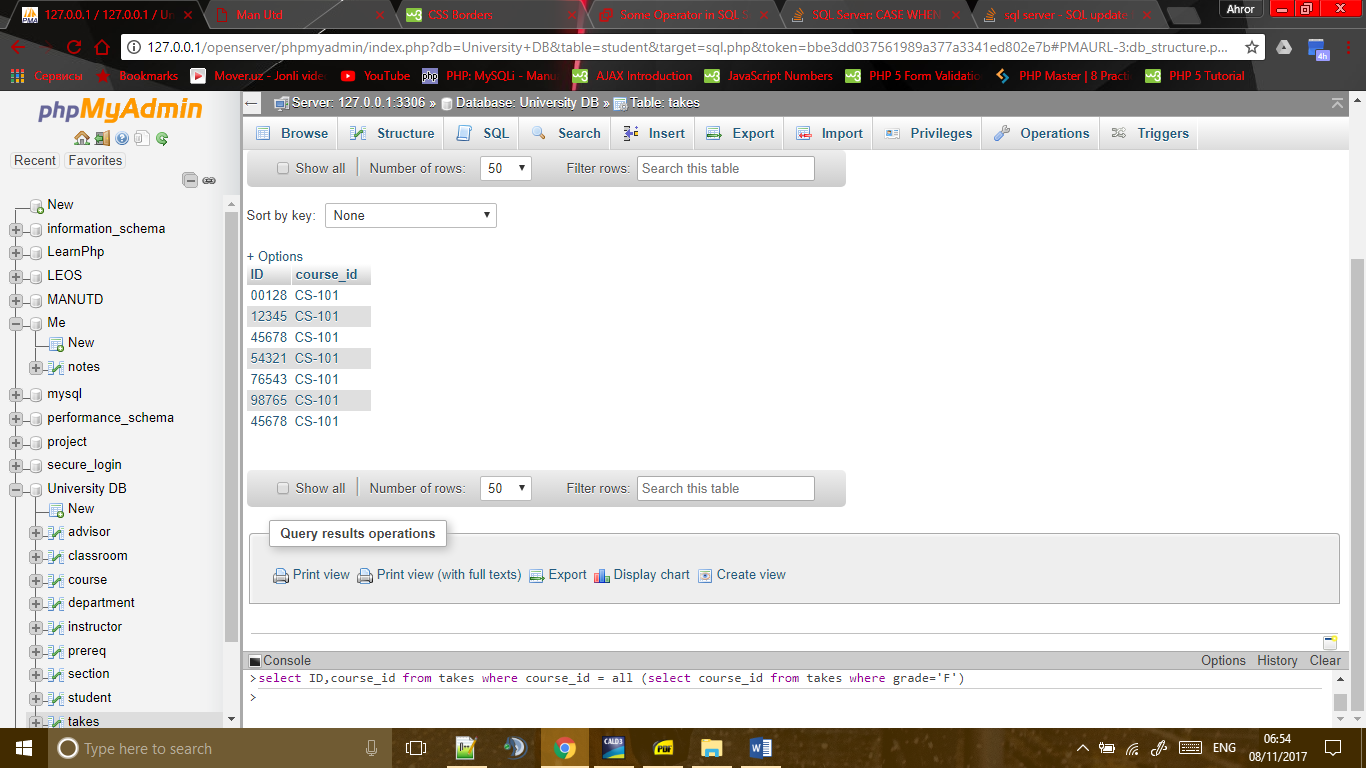
**c)** Both queries are taking input from exactly same relation: takes;

**d)** Yes, done. Mentioned in below.

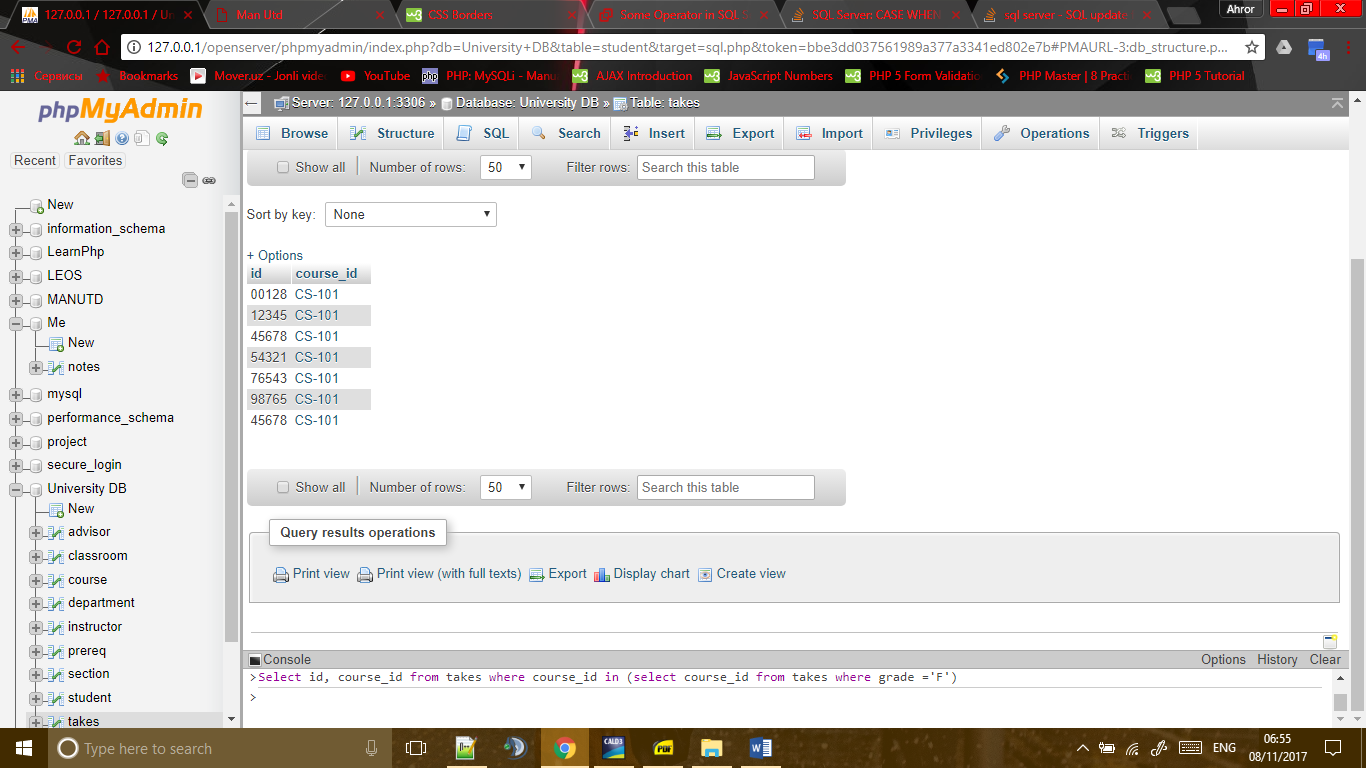
**Then:**

1. **Q1**: find the IDs and course\_ids of all students who did not study with all students who took ‘F’ in one course. **Q2**: find the IDs and course\_ids of all students who did not study with students who took ‘F’ in one course.
2. **1. Q1:** select ID,course\_id from takes where course\_id = all (select course\_id from takes where grade='F')

**Q2:** Select id, course\_id from takes where course\_id in (select course\_id from takes where grade ='F')

**2.** **Q1:**

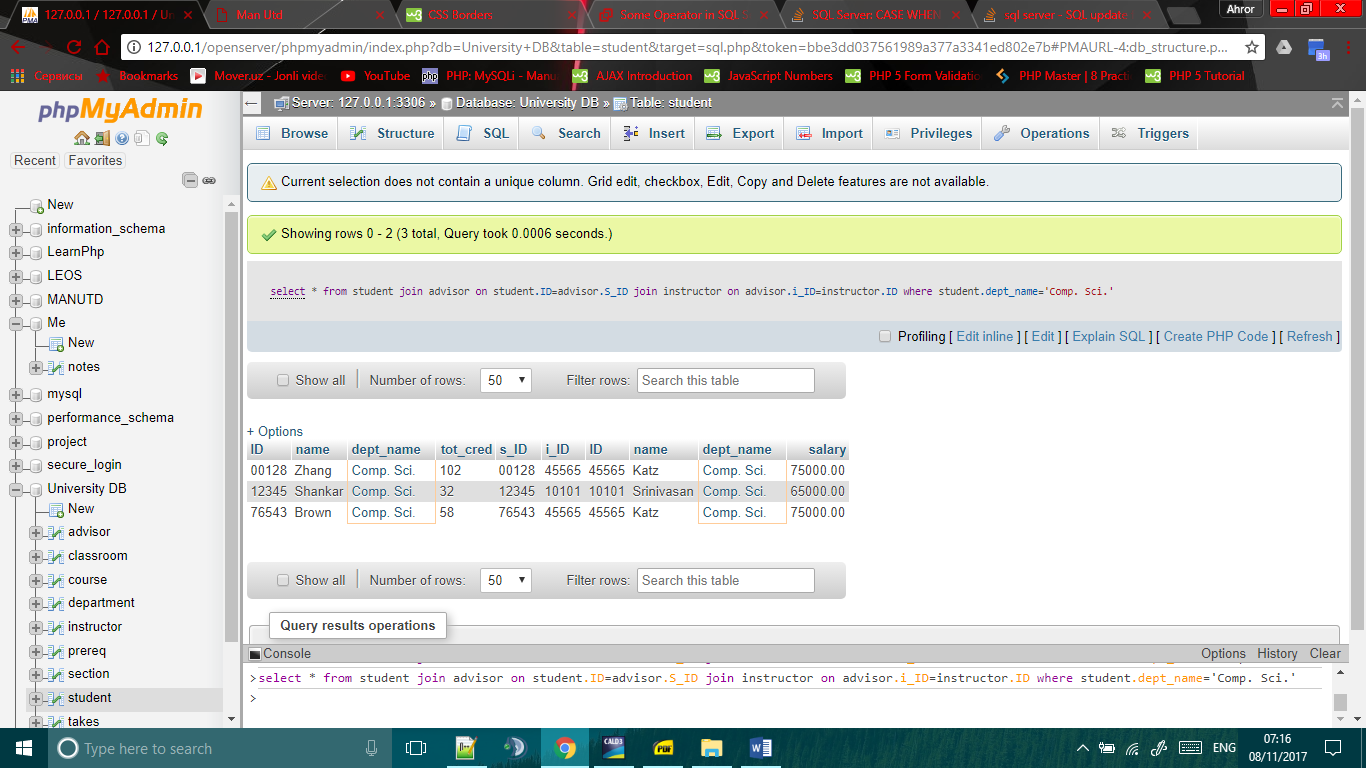
**Q2:**



**3.** Q1 and Q2 does not give same result in all cases. In our case, they are giving same results, because student who took ‘F’ is one. If we take, who took grade ‘C’ it will be very different result between Q1 and Q2. Because, there are two students who took ‘C’ and using **all** we can compare our variable to all results of subquery and comparable variable should be equal to all result, which means very rare. Therefore, I chose students who took ‘F’, and we have some result! If, I chose another grades it would had big chance to not print any result! In Q2 we used **in**. It means variable will be compared to all results. Moreover, when using **in** we compare our variable to all results and get if at least one will be in the results. It means, **in** is equal to **=some** and not equal to **=all**.

**Exercise 7**

**Query:** select \* from student join advisor on student.ID=advisor.S\_ID join instructor on advisor.i\_ID=instructor.ID where student.dept\_name= 'Comp. Sci.'



**Exercise 8**

**Query:** update student set student.tot\_cred=case

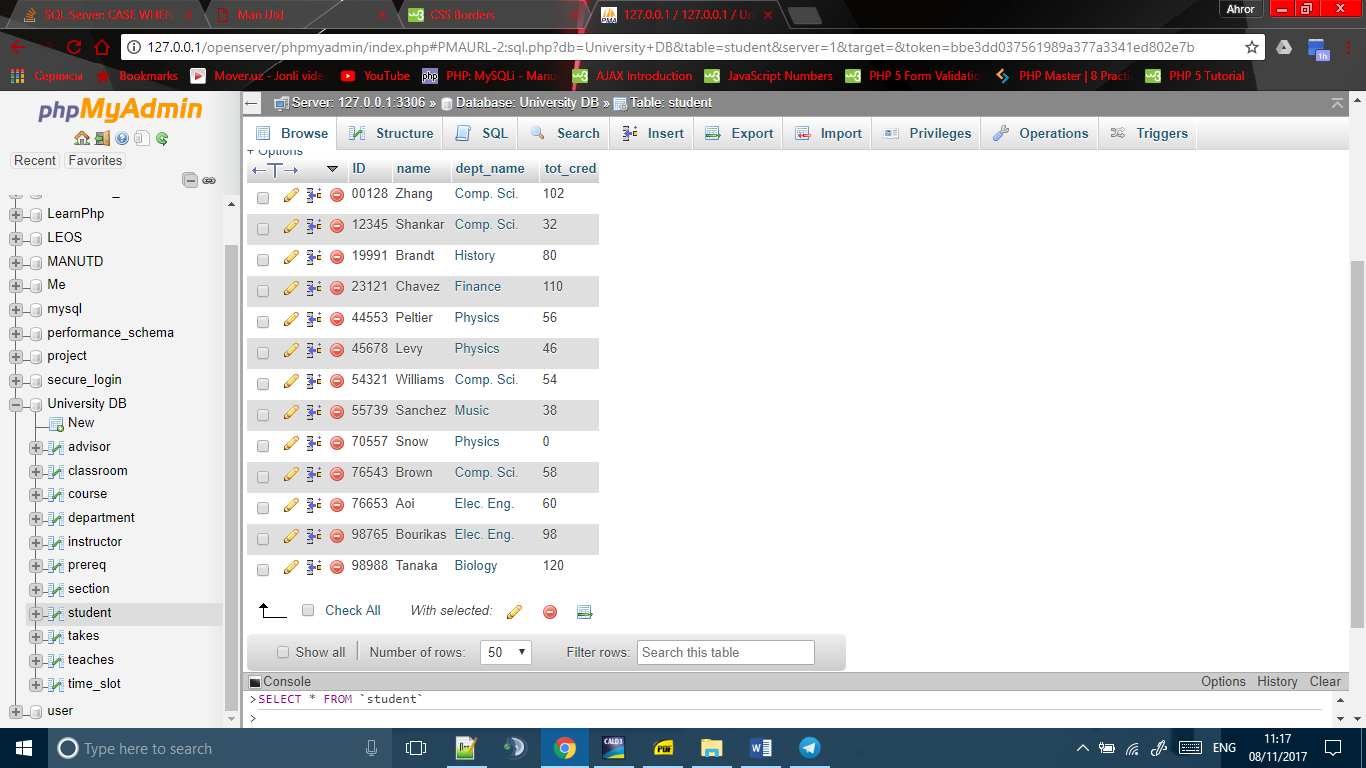
when student.dept\_name='Comp. Sci.' and student.ID in (Select id from takes where takes.course\_id='CS-347' and takes.grade in ('A','A-','A+','B+')) then student.tot\_cred+4

when student.dept\_name='Comp. Sci.' and student.ID in (Select id from takes where takes.course\_id='CS-347' and takes.grade in ('C+','B-','B')) then student.tot\_cred+3

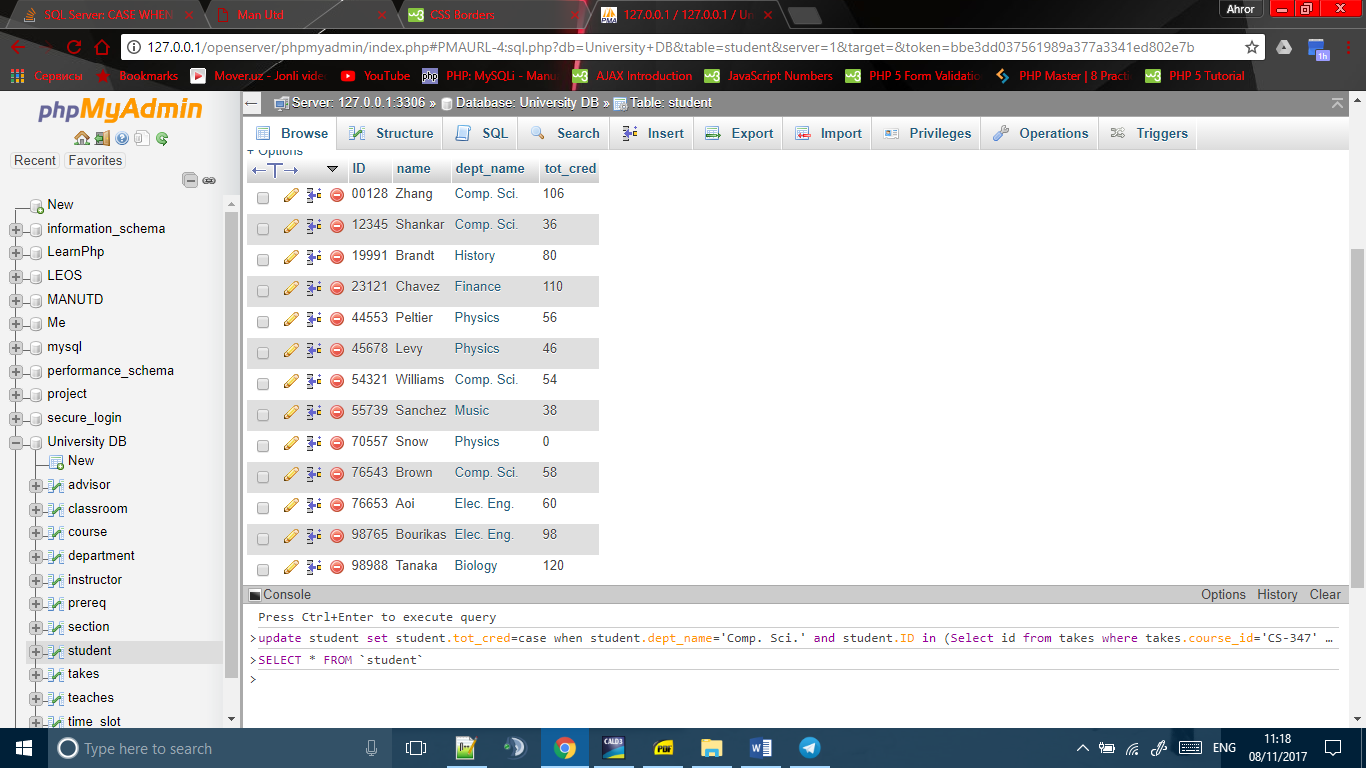
when student.dept\_name='Comp. Sci.' and student.ID in (Select id from takes where takes.course\_id='CS-347' and takes.grade in ('F')) then student.tot\_cred-3

else student.tot\_cred

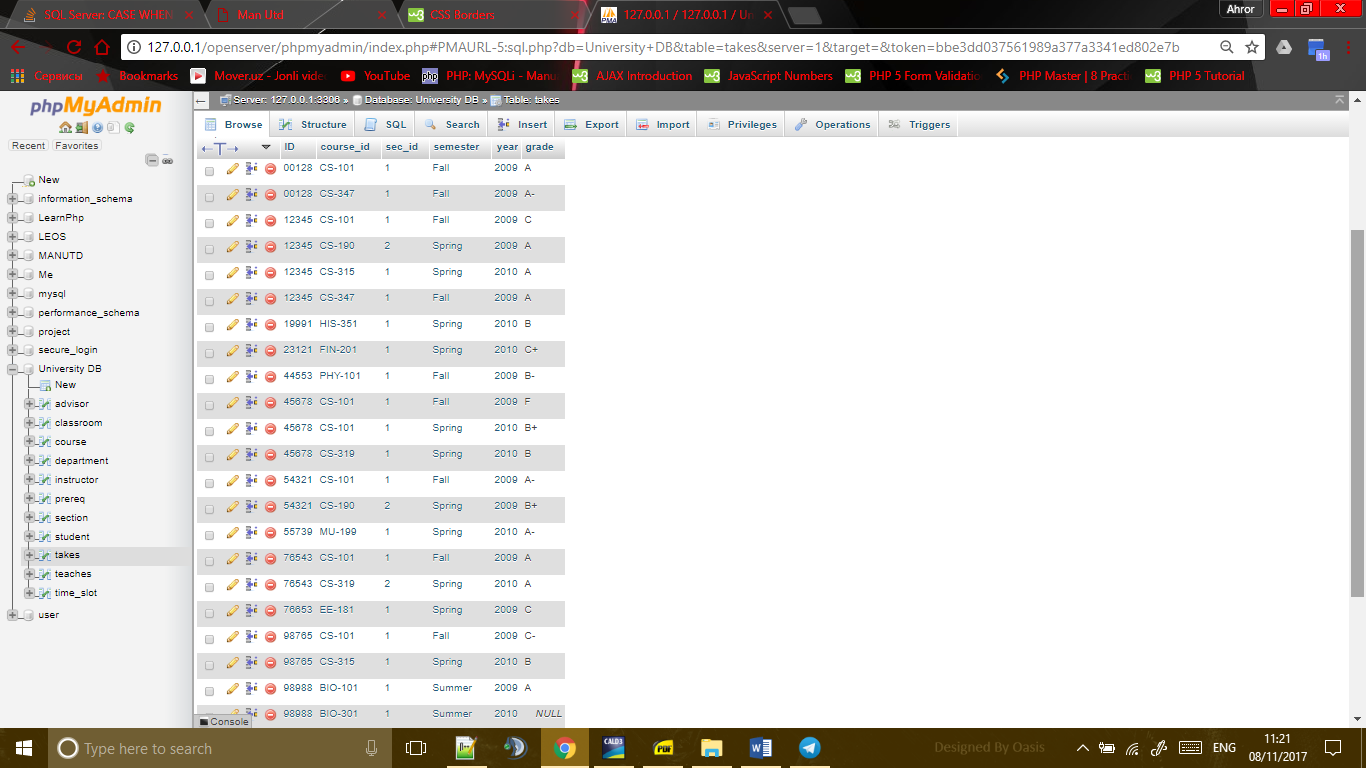
end

**Before update:** 

**After update:**



We can see from takes table that only student with id 00128 and 12345’s course\_id is CS-347, so only they changed. 4 added to their grade because they A- and A grade respectively.

**Takes(Table):** 

**Answers of 9:**

**(a):**

**create table** *transaction\_journal* (

*ID*  **varchar(20),**

*note\_of\_transaction* **varchar(20),**

*SAaccount* **numeric(17,2),**

*ODaccount* **numeric(17,2),**

*amount*  **numeric(17,2),**

**primary key** (*ID*)

);

**create table** *OD\_accounts* (

*ID* **varchar(20),**

*person\_name* **varchar(15),**

*ODaccount\_number* **numeric(8,0),**

*balance*  **numeric(18,2),**

**primary key** (*ID, person\_name*),

**foreign key** (*ID*) **references** *transaction\_journal*

);

**create table** *SA\_accounts*

(*ID*  **varchar(20),**

*person\_name* **varchar(15),**

*SAaccount\_number* **numeric(8,0),**

*balance*  **numeric(18,2),**

**primary key** (*ID, person\_name*),

**foreign key** (*ID, person\_name*) **references** *OD\_accounts*,

**foreign key** (ID) **references** *transaction\_journal*);

**(b)**

**insert into** *SA\_account*

**values** ('U1510067', 'Maftuna Sharabbaeva', '789456', '258921.12');

**insert into** *SA\_account*

**values** ('U1510066', 'Sharaf Muqiova', '999898', '888921.12');

**insert into** *OD\_account*

**values** ('U1510067', 'Maftuna Sharabbaeva', '456456', '0.00');

**insert into** *OD\_account*

**values** ('U1510066', 'Sharaf Muqiova', '789789', '0.00');

**insert into** *transaction\_journal*

**values** ('U1510067', 'transacted well', '789456', '456456', '0.00');

**insert into** *transaction\_journal*

**values** ('U1510066', 'transacted well', '999898', '789789', '0.00');

**(c)**

**start transaction**;

**update** *SA\_accounts*

**set** *balance*=*balance*-256.06

**where** *SA\_accounts.ID*='U1510066'

**and** *SA\_accounts.person\_name*='Sharaf Muqiova';

**update** *OD\_accounts*

**set** *balance*=*balance*+256.06

**where** *OD\_accounts.ID*='U1510066'

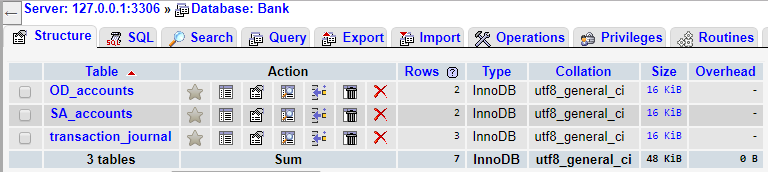
**and** *OD\_accounts.person\_name*='Sharaf Muqiova';

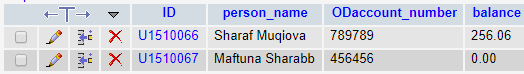
**INSERT INTO** *transaction\_journal*

**values**('1', 'transacted well', '999898', '789789', '256.06');

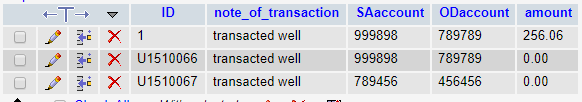
**commit;**

**sample result of above work at the end:**









**Answer of 10:**

**(a):**

**i.** for section relation we check in following way:

*first check is :*

check (building, room\_no

in (select building, room\_no

from classroom));

*second check is:*

check (time\_slot\_id

in (select time\_slot\_id

from time\_slot));

ii. for instructor relation, we should check following way:

check (dept\_name

in (select dept\_name

from department));

iii. for advisor relation we should check in following way:

*first check is:*

check (s\_id

in (select ID

from student));

*second check is:*

check (i\_id

in (select ID

from instructor));

**(b):**

**i:**

**section** -> unprotected because *course\_id, sec\_id,* *semester* and *year* are primary keys and referenced by **takes** relation. if we insert or update or delete any data from them, takes relation will have problems because it may be using that data. we can use the same concept with takes relation with these attributes. In addition, from logical side if my mom wants to insert any **section**that **section** should be taught by at least one **instructor**. so we should check here. As well as we should check building and *room\_no* whether in building and *room\_no* of **classroom** relation because if we want to insert any data that data should be at **classroom** relation too. As well as we should check whether *time\_slot \_id* is in *time\_slot* relation for consistency.

**teaches** -> unprotected because *ID* is primary key and foreign key for **instructor** relation. if we want to insert any **instructor**'s *ID* at **teaches** that **instructor** should be exist. so we should check before inserting

**takes** -> unprotected because *ID* is primary key and foreign key for **student** relation. if we want to insert any **student**'s *ID* at takes that **student** should be exist. so we should check before inserting

**course** -> unprotected because *course\_id* is primary key and referenced by *course\_id* attribute of **section** and *course\_id* as well as *prereq\_id* attributes of **prereq** relation. if we insert or update or delete any data from it, **prereq** and **section** relations will have problems because they may be using that data we also check whether *dept\_name* is in *dept\_name* of **department** relation for consistency.

**student** -> unprotected because *ID* is primary key and referenced by *s\_id* attribute of **advisor** relation. if we insert or update or delete any data from it, **advisor** relation will have problems because it may be using that data. In addition, from logical side if my mom wants to insert any student that student should take any advisor according to rule of University otherwise my mom will build problems in **advisor** relation. In addition, we should check *dept\_name* because student should be in exist department. My mom should not build inconsistency.

**instructor** -> unprotected because *dept\_name* should be checked whether it is in **department** relation or not because instructor should take exist department.

**advisor**-> unprotected. Because for example when my mom wants to insert advisor of student that student should be exist. *s\_id* should be in ID of student relation. The same is with instructor *i\_id* should be in *ID* attribute of **instructor** relation for consistency.

**list of them are:**

1. section

2. teaches

3. takes

4. course

5. student

6. advisor

7. instructor

**ii:**

check() for section table:

*first check() is*:

check ((course\_id, sec\_id, semester, year)

in (select course\_id, sec\_id, semester, year

from section));

*second check() is :*

check (time\_slot\_id

in (select time\_slot\_id

from time\_slot));

*third check() is:*

check ((building, room\_no)

in (select building, room\_no

from classroom));

*fourth check() is:*

check ((course\_id, sec\_id, semester, year)

in (select course\_id, sec\_id, semester, year

from takes));

check() for takes relation:

check (ID

in (select ID

from student));

check() for teaches relation:

check (ID

in (select ID

from instructor));

check() for course relation:

*first check() is:*

check (course\_id

in (select course\_id, prereq\_id

from prereq));

*second check() is:*

check (dept\_name

in (select dept\_name

from department));

check() for student relation:

*first check:*

check (ID

in (select s\_id

from advisor));

*second check() is:*

check (dept\_name

in (select dept\_name

from department));

check() for instructor relation:

*first check is:*

check (ID

in (select ID

from teaches));

*second check() is:*

check (dept\_name

in (select dept\_name

from department));

check() for advisor relation:

check(i\_id

in (select ID

from instructor));