Homework: Chapter 3

3.1 Consider the following employee database, where the primary keys are underlined. Given an expression in SQL for each of the following queries.

*employee(employee\_name, street, city)*

*works(employee\_name, company\_name, salary)*

*company(company\_name, city)*

*manages(employee\_name, manager\_name)*

1. Find the names and cities of residence of all employees who work for “First Bank Corporation”.

**select** *e.employee name*, *city* **from** *employee e*, *works w* **where** *w.company name =* ’First Bank Corporation’ **and** *w.employee name = e.employee name*

1. Find the names, street address, and cities of residence of all employees who work for “First Bank Corporation” and earn more than $10,000.

**select** \* **from** *employee* **where** *employee name* **in** (**select** *employee name* **from** *works*

**where** *company name* = ’First Bank Corporation’ **and** *salary > 10000*)

1. Find the names of all employees in database who work for “First Bank Corporation”.’

**select** *employee\_name* **from** *works* **where** *company name* = ’First Bank Corporation’

1. Find the employees in database who do not work for “First Bank Corporation”.’

**select** *employee\_name* **from** *works* **where** *company name !*= ’First Bank Corporation’

1. Find all employees in the database who earn more than each employee of “Small Bank Corporation”.

**select** *employee name* **from** *works* **where** *salary* > (**select max(***salary)* **from** *works*

**where** *company name* = ’Small Bank Corporation’)

1. Assume that the companies may be located in several cities. Find all companies located in every city in which “Small Bank Corporation” is located.

**select** *S.company name* **from** *company S* **where not exists** ((**select** *city* **from** *company*

**where** *company name* = ’Small Bank Corporation’)

**except**

(**select** *city* **from** *company T* **where** *S.company name = T.company name*))

1. Find the company that has the most employees.

**select** *company\_name* **from** *works* **group by** *company\_name*

**having count(***distinct employee\_name***) >= all** (**select count(***distinct employee\_name***)**

**from** *works* **group by** *company\_name*)

1. Find those companies whose employees earn a higher salary, on average, than the average salary at “First Bank Company”.

**select** *company\_name* **from** *works* **group by** *company\_name* **having avg(***salary***) >**

(**select avg(***salary***) from** *works* **where** *company\_name* **= ‘***First Bank Corporation***’**)

1. Find all employees in the database who live in the same cities as the companies for which they work.

**select** *employee.employee\_name* **from** *employee, works, company*

**where** *employee.employee\_name* **=** *works.employee\_name* **and** *works.company\_name* **=** *company.company\_name* **and** *employee.city* **=** *company.city*

1. Find all employees in the database who live in the same cities and on the same streets as do their managers.

**select** *E.employee\_name* **from** *employee E, employee M, manages*

**where** *E.employee\_name* **=** *manages.employee\_name* **and** *M.employee\_name* **=** *manages.manager\_name* **and** *E.street* **=** *M.street* **and** *E.city* **=** *M.city*

1. Find all employees who earn more than the average salary of all employees of their company.

**with** *salary\_avg(company\_name,avg\_salary)* **as** (**select** *company\_name, avg(salary)*

**from** *works* **group by** *company\_name*)

**select** *W.employee\_name* **from** *works W, salary\_avg*

**where** *W.salary* **>** *avg\_salary* **and** *W.company\_name* **=** *salary\_avg.company\_name*

1. Find the company that has the smallest payroll.

**select** *company\_name* **from** *works* **group by** *company\_name*

**having** **sum***(salary)* **<= all** (**select sum***(salary)* **from** *works* **group by** *company\_name*)

1. Modify the database so that “Jones” now lives in “Newtown”.

**update** *employee* **set** *city* **=** ‘*Newtown*’ **where** *employee\_name* **=** ‘*Jones*’

1. Give all employees of “First Bank Corporation” a 10 percent raise.

**update** *works* **set** *salary* **=** *salary\*1.1* **where** *company\_name* **=** ‘*First Bank Corporation’*

1. Give all managers of “First Bank Corporation” a 10 percent raise.

**update** *works* **set** *salary* **=** *salary\*1.1* **where** *employee\_name* **in** (**select** *manager\_names* **from** *manages*) **and** *company\_name* **=** ‘*First Bank Corporation*’

1. Give all managers of “First Bank Corporation” a 10 percent raise unless the salary becomes greater than $100,000; in such case, give only a 3 percent raise.

**update** *works* **set** *salary* **=** **case** **when** *salary* >= *10000* **then** *salary\*1.03* **else** s*alary\*1.1*

**where** *employee\_name* **in** (**select** *manager\_names* **from** *manages*) **and** *company\_name* **=** ‘*First Bank Corporation*’

1. Delete all tuples in the works relation for employees of “Small Bank Corporation”.

**delete from** *works* **where** *company\_name* **=** ‘*Small Bank Corporation*’

3.2 Consider the following university schema. Write the following queries in SQL.

*department(dept\_name,building, budget)*

*course (course\_id, title, dept\_name, credits)*

*instructor (ID, name, dept\_name, salary)*

*section (course\_id, sec\_id, semester, year, building, room\_number, time, slot\_id)*

*teaches (ID, course\_id, sec\_id, semester, year)*

*student(ID, name, dept\_name, tot\_cred)*

*takes(ID, course\_id, sec\_id, semester, year, grade)*

*advisor(s\_ID, i\_ID)*

*time\_slot(time\_slot\_id, day, start\_time, end\_time)*

*prereq(course\_id, prereq\_id)*

*classroom (building, room\_number, capacity)*

1. Find the *title*s of courses in the Comp.Sci. department that have 3 credits.

**select** *title* **from** *course* **where** *dept name* = ’Comp. Sci.’ **and** *credits* = 3

1. Find the *ID*s of all students who were taught by an instructor named Einstein; make sure there are no duplicates result.

**select** **distinct** *student.ID* **from** s*tudent*, *takes,instructor, teaches* **where** *student.ID* **=** *takes.ID* **and** *instructor.ID* **=** *teaches.ID* **and** *takes.course\_id* **=** *teaches.course\_id* **and** *takes.sec\_id* **=** *teaches.sec\_id* **and** *takes.semester* **=** *teaches.semester* **and** *takes.year* **=** *teaches.year* **and** *instructor.name* **=** ’*Einstein*’

1. Find the highest salary of any instructor.

**select** max(salary) **from** instructor

1. Find all instructors earning the highest salary (there may be more than one with the same salary).

**select** ID, name **from** instructor where salary = (select max(salary) from instructor)

1. Find the enrollment of each section that was offered in Fall 2009.

**select** *course id,* *sec id*, **count***(ID)* **from** *section* **natural join** *takes*

**where** *semester* **=** *’Fall’* **and** *year* = *2009* **group by** *course id,* *sec id*

1. Find the maximum enrollment, across all sections, in Fall 2009.

**select max**(*enrollment*) **from** (**select count**(*ID*) **as** *enrollment*

**from** *section* **natural join** *takes*

**where** *semester* = ’*Fall*’ **and** *year* = 2009

**group by** *course id*, *sec id*)

1. Find the *section*s that had the maximum enrollment in Fall 2009.

**with** *sec\_enrollment* **as** (

**select** *course\_ id*, *sec\_id*, **count**(*ID*) **as** *enrollment*

**from** *section* **natural join** *takes*

**where** *semester* = ’Autumn’

**and** *year* = 2009

**group by** *course id*, *sec id*)

**select** *course id*, *sec id*

**from** *sec enrollment*

**where** *enrollment* = (**select max**(*enrollment*) **from** *sec enrollment*)

1. Find the *name*s of all students who have taken at least one Comp.Sci. course; make sure there are no duplicate names in the result.

**select** **distinct** *name* **from** *student* **natural join** *takes* **natural join** *course* **where** *course.dept\_name* **=** ‘*Comp.Sci*’

1. Find the *ID*s and *name*s of all students who have not taken any course offering before Spring 2009.

**select** *id, name* **from** *student*

**except**

**select** *id*, *name* **from** *student* **natural join** *takes* **where** *year* **<** *2009*

1. For each department, find the maximum salary of instructors in that department. You may assume that every department has at least one instructor.

**select** *dept\_name*, max(*salary*) **from** *instructor* **group by** *dept\_name*

1. Find the lowest, across all departments, of the per-department maximum salary computed by the preceding query.

**select** **min**(*max\_salary*) **from** (**Select** **max**(*salary*) **as** *max\_salary* **from** *instructor* **group by** *dept\_name*)

1. Create a new course “CS-001”, titled “Weekly Seminar”, with 0 credits.

**iInsert into** *course* **values**(‘*CS-001*’, ‘*Weekly Seminar*’, 0)

1. Create a section of this course in Fall 2009, with *sec\_id* of 1.

**insert into** *section* **values**(‘*CS-001*’, ’*1*’, ‘*Autumn*’, *2009*, *null, null, null, null, null)*

1. Enroll every student in the Comp.Sci. department in the above section.

**insert into** *takes* **select** *id*, ‘*CS-001*’, ‘*1*’, ‘*Autumn*’, *2009,* *null* **from** *student* **where** *dept\_name* **=** ‘*Comp\_Sci*’

1. Delete all enrollments in the above section where the student’s name is Chavez.

**delete** **from** *takes* **where** *course\_id* **=** ‘*CS-001*’ **and** *section\_id* **=** ‘*1*’ **and** *year* **=** *2009* **and** *semester* **=** ‘*Autumn*’ **and** *id* **in** (**select** *id* **from** *student* **where** *name* **=** ‘*Chavez*’)

1. Delete the course CS-001. What will happen if you run this delete statement without first deleting offerings (sections) of this course.

**delete from** *takes* **where** *course\_id* **=** ‘*CS-001*’

**delete from** *section* where *course\_id* **=** ‘*CS-001*’

**delete from** *course* where *course\_id* **=** ‘*CS-001*’

如果没有删除section和takes中的相应课程，则不能删除course中的相应课程，因为破坏了外键约束，删除非法。

1. Delete all *takes* tuples corresponding to any section of any course with the word “database” as a part of the title; ignore case when matching the word with title.

**delete from** *takes* **where** *course\_id* **in** (**select** *course\_id* **from** *course* **where** **lower**(*title*) **like** ‘%*database*%’)

3.3 Consider the query:

Select course\_id, semester, year, sec\_id, avg(tot\_cred)

From takes natural join student

Where year = 2009

Group by course\_id, semester, year, sec\_id

Having count(ID) >= 2

Explain why joining section as well in the from clause would not change the result.

因为section的主码在takes中完全包含，所以连接后不会产生任何新的元组，加不加入不影响结果。