HOMEWORK 1
CS480 (Spring, 2016)
DUE: Wednesday 2/10/2016, 11.59pm

Do the following exercises from chapter 3 of the textbook.

Exercise number	- Points
3.11	- 16
3.12	- 24
3.14	- 8
3.15	- 12
3.16	- 20
3.17	- 12
3.18	- 4
3.19	- 4
Total	100

These exercises are based on schemas written in the book and also below for your convenience. You must write your solutions in standard SQL, but are encouraged to run the solutions also on MySQL to get acquainted with MySQL. In case MySQL does not support the constructs in your solution, you are encouraged to find other ways of running the queries. Submit only the SQL solutions.

Submit your solutions on Blackboard under Homework 1. No late submissions will be accepted. The link for the submission will be removed automatically at midnight of the deadline day.

Exercise 3.11

Write the following queries in SQL, using the **University** schema.

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

b. Find the IDs and names of all students who have not taken any course offering before Spring 2009.

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

d. Find the lowest, across all departments, of the per-department maximum

salary computed by the preceding query.

Exercise 3.12
Write the following queries in SQL, using the **University** schema.
a. Create a new course "CS-001", titled "Weekly Seminar", with 0 credits.
b. Create a section of this course in Autumn 2009, with *section id* of 1.
c. Enroll every student in the Comp. Sci. department in the above section.
d. Delete enrollments in the above section where the student's name is Chavez.
e. Delete the course CS-001. What will happen if you run this delete statement without first deleting offerings (sections) of this course.
f. 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 the title.

Exercise 3.14
Consider the **Insurance** database, where the primary keys are underlined. Construct the following SQL queries for this relational database.
a. Find the number of accidents in which the cars belonging to "John Smith" were involved.
b. Update the damage amount for the car with license number "AABB2000" in the accident with report number "AR2197" to $3000.

Exercise 3.15
Consider the **Bank** database, where the primary keys are underlined. Construct the following SQL queries for this relational database.
a. Find all customers who have an account at *all* the branches located in "Brooklyn".
b. Find out the total sum of all loan amounts in the bank.
c. Find the names of all branches that have assets greater than those of at least one branch located in "Brooklyn".

Exercise 3.16
Consider the **Employee** database, where the primary keys are

underlined. Give an expression in SQL for each of the following queries.
a. Find the names of all employees who work for First Bank Corporation.
b. Find all employees in the database who live in the same cities as the companies for which they work.
c. Find all employees in the database who live in the same cities and on the same streets as do their managers.
d. Find all employees who earn more than the average salary of all employees of their company.
e. Find the company that has the smallest payroll.

Exercise 3.17
Consider the **Employee** relational database. Give an expression in SQL for each of the following queries.
a. Give all employees of First Bank Corporation a 10 percent raise.
b. Give all managers of First Bank Corporation a 10 percent raise.
c. Delete all tuples in the *works* relation for employees of Small Bank Corporation.

Exercise 3.18
List two reasons why null values might be introduced into the database.

Exercise 3.19
Show that, in SQL, <> **all** is identical to **not in**.

Insurance schema

person (driver_id, name, address)
car (license, model, year)
accident (report_number, date, location)
owns (driver_id, license)
participated (report_number, license, driver_id, damage amount)

Bank schema

branch(branch_name, branch city, assets)
customer (customer name, customer street, customer city)
loan (loan number, branch name, amount)
borrower (customer name, loan number)

account (account number, branch name, balance)
depositor (customer name, account number)

Employee schema

employee (employee name, street, city)
works (employee name, company name, salary)
company (company name, city)
manages (employee name, manager name)

University schema

classroom (building, room_number, capacity)
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_hr, start_min, end_hr, end_min)
prereq(course_id, prereq_id)

where the tables have been created in the following way:

```
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),
       primary key (dept_name)
      );

create table course
      (course_id      varchar(8),
       title          varchar(50),
       dept_name      varchar(20),
       credits        numeric(2,0),
```

```
       primary key (course_id),
       foreign key (dept_name) references department(dept_name)
      );

create table instructor
      (ID             varchar(5),
       name           varchar(20) not null,
       dept_name      varchar(20),
       salary         numeric(8,2),
       primary key (ID),
       foreign key (dept_name) references department(dept_name)
      );

create table section
      (course_id      varchar(8),
       sec_id         varchar(8),
       semester       varchar(6),
       year           numeric(4,0),
       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),
       foreign key (building, room_number) references
       classroom(building, room_number)
      );

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),
       foreign key (ID) references instructor(ID)
      );

create table student
      (ID             varchar(5),
       name           varchar(20) not null,
       dept_name      varchar(20),
       tot_cred       numeric(3,0),
       primary key (ID),
       foreign key (dept_name) references department(dept_name)
      );

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),
       foreign key (ID) references student(ID)
      );

create table advisor
      (s_ID            varchar(5),
       i_ID            varchar(5),
       primary key (s_ID),
       foreign key (i_ID) references instructor (ID),
       foreign key (s_ID) references student (ID)
      );

create table time_slot
      (time_slot_id    varchar(4),
       day             varchar(1),
       start_hr        numeric(2),
       start_min       numeric(2),
       end_hr          numeric(2),
       end_min         numeric(2),
       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),
       foreign key (prereq_id) references course(course_id)
      );
```