

## Homework 2

### SQL

Total points: 40

1. (10 pts.) Database creation and querying exercise.

advisor table:

```

database — sqlite3 — 80x24
Error: no such table: advisor
sqlite> .schema advisor
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
);
sqlite> select * from advisor;
s_ID      i_ID
-----
00128     45565
12345     10101
23121     76543
44553     22222
45678     22222
76543     45565
76653     98345
98765     98345
98988     76766
sqlite>

```

classroom table:

```

database — sqlite3 — 80x24
Packard    101      500
Painter    514       10
Taylor     3128     70
Watson     100       30
Watson     120       50
sqlite> .tables
advisor    course      instructor  section    takes      time_slot
classroom  department  prereq     student    teaches
sqlite> .schema classroom
CREATE TABLE classroom
(
  building      varchar(15),
  room_number   varchar(7),
  capacity      numeric(4,0),
  primary key (building, room_number)
);
sqlite> select * from classroom;
building    room_number  capacity
-----
Packard     101         500
Painter     514         10
Taylor      3128        70
Watson      100         30
Watson      120         50
sqlite>

```

course table:

```
database — sqlite3 — 80x27
sqlite> .schema course
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
                        on delete set null
);
sqlite> select * from course;
course_id  title                dept_name  credits
-----
BIO-101    Intro. to Biology    Biology    4
BIO-301    Genetics             Biology    4
BIO-399    Computational Bio     Biology    3
CS-101     Intro. to Compute    Comp. Sci. 4
CS-190     Game Design          Comp. Sci. 4
CS-315     Robotics             Comp. Sci. 3
CS-319     Image Processing      Comp. Sci. 3
CS-347     Database System C     Comp. Sci. 3
EE-181     Intro. to Digital    Elec. Eng. 3
FIN-201    Investment Bankin     Finance    3
HIS-351    World History         History    3
MU-199     Music Video Produ     Music      3
PHY-101    Physical Principl     Physics    4
sqlite>
```

department table:

```
database — sqlite3 — 80x24
HIS-351    World History         History    3
MU-199     Music Video Produ     Music      3
PHY-101    Physical Principl     Physics    4
sqlite> .tables
advisor    course        instructor    section    takes        time_slot
classroom  department    prereq        student    teaches
sqlite> .schema department
CREATE TABLE department
(
  dept_name          varchar(20),
  building           varchar(15),
  budget             numeric(12,2) check (budget > 0),
  primary key (dept_name)
);
sqlite> select * from department;
dept_name  building  budget
-----
Biology    Watson   90000
Comp. Sci. Taylor    100000
Elec. Eng. Taylor    85000
Finance    Painter   120000
History    Painter   50000
Music      Packard   80000
Physics    Watson    70000
sqlite>
```

instructor table:

```
database — sqlite3 — 80x26
sqlite> .schema instructor
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
    on delete set null
);
sqlite> select * from instructor;
ID          name        dept_name   salary
-----
10101       Srinivasan   Comp. Sci.  65000
12121       Wu           Finance     90000
15151       Mozart       Music       40000
22222       Einstein     Physics     95000
32343       El Said      History     60000
33456       Gold         Physics     87000
45565       Katz          Comp. Sci.  75000
58583       Califieri    History     62000
76543       Singh        Finance     80000
76766       Crick        Biology     72000
83821       Brandt       Comp. Sci.  92000
98345       Kim          Elec. Eng.  80000
sqlite>
```

prereq table:

```
database — sqlite3 — 80x24
98345       Kim          Elec. Eng.  80000
sqlite> .tables
adviser      course      instructor  section     takes       time_slot
classroom    department  prereq      student     teaches
sqlite> .schema prereq
CREATE TABLE prereq
(
  course_id   varchar(8),
  prereq_id   varchar(8),
  primary key (course_id, prereq_id),
  foreign key (course_id) references course
    on delete cascade,
  foreign key (prereq_id) references course
);
sqlite> select * from prereq;
course_id   prereq_id
-----
BI0-301     BI0-101
BI0-399     BI0-101
CS-190      CS-101
CS-315      CS-101
CS-319      CS-101
CS-347      CS-101
EE-181      PHY-101
sqlite>
```



section table:

```
sqlite> .schema section
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
    on delete cascade,
  foreign key (building, room_number) references classroom
    on delete set null
);

sqlite> select * from section;
course_id  sec_id  semester  year  building  room_number  time_slot_id
-----
BIO-101    1       Summer   2009   Painter   514          B
BIO-301    1       Summer   2010   Painter   514          A
CS-101     1       Fall     2009   Packard   101          H
CS-101     1       Spring   2010   Packard   101          F
CS-190     1       Spring   2009   Taylor    3128         E
CS-190     2       Spring   2009   Taylor    3128         A
CS-315     1       Spring   2010   Watson    120          D
CS-319     1       Spring   2010   Watson    100          B
CS-319     2       Spring   2010   Taylor    3128         C
CS-347     1       Fall     2009   Taylor    3128         A
EE-181     1       Spring   2009   Taylor    3128         C
FIN-201    1       Spring   2010   Packard   101          B
HIS-351    1       Spring   2010   Painter   514          C
MU-199     1       Spring   2010   Packard   101          D
PHY-101    1       Fall     2009   Watson    100          A
sqlite>
```

student table:

```
sqlite> .schema student
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
    on delete set null
);

sqlite> select * from student;
ID      name      dept_name  tot_cred
-----
00128   Zhang     Comp. Sci. 102
12345   Shankar   Comp. Sci. 32
19991   Brandt    History    80
23121   Chavez    Finance    110
44553   Peltier   Physics     56
45678   Levy      Physics     46
54321   Williams  Comp. Sci. 54
55739   Sanchez   Music       38
70557   Snow      Physics     0
76543   Brown     Comp. Sci. 58
76653   Aoi       Elec. Eng. 60
98765   Bourikas  Elec. Eng. 98
98988   Tanaka    Biology     120
sqlite>
```

takes table:

```
sqlite> .schema takes
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
    on delete cascade,
  foreign key (ID) references student
    on delete cascade
);

sqlite> select * from takes;
ID          course_id   sec_id      semester    year        grade
-----
00128      CS-101      1           Fall        2009        A
00128      CS-347      1           Fall        2009        A-
12345      CS-101      1           Fall        2009        C
12345      CS-190      2           Spring      2009        A
12345      CS-315      1           Spring      2010        A
12345      CS-347      1           Fall        2009        A
19991      HIS-351      1           Spring      2010        B
23121      FIN-201      1           Spring      2010        C+
44553      PHY-101      1           Fall        2009        B-
45678      CS-101      1           Fall        2009        F
45678      CS-101      1           Spring      2010        B+
45678      CS-319      1           Spring      2010        B
54321      CS-101      1           Fall        2009        A-
54321      CS-190      2           Spring      2009        B+
55739      MU-199      1           Spring      2010        A-
76543      CS-101      1           Fall        2009        A
76543      CS-319      2           Spring      2010        A
76653      EE-181      1           Spring      2009        C
98765      CS-101      1           Fall        2009        C-
98765      CS-315      1           Spring      2010        B
98988      BIO-101      1           Summer      2009        A
98988      BIO-301      1           Summer      2010        NULL
sqlite>
```

teaches table:

```
sqlite> .schema teaches
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
    on delete cascade,
  foreign key (ID) references instructor
    on delete cascade
);

sqlite> select * from teaches;
ID          course_id   sec_id      semester    year
-----
10101      CS-101      1           Fall        2009
10101      CS-315      1           Spring      2010
10101      CS-347      1           Fall        2009
12121      FIN-201      1           Spring      2010
15151      MU-199      1           Spring      2010
22222      PHY-101      1           Fall        2009
32343      HIS-351      1           Spring      2010
45565      CS-101      1           Spring      2010
45565      CS-319      1           Spring      2010
76766      BIO-101      1           Summer      2009
76766      BIO-301      1           Summer      2010
83821      CS-190      1           Spring      2009
83821      CS-190      2           Spring      2009
83821      CS-319      2           Spring      2010
98345      EE-181      1           Spring      2009
sqlite>
```

time\_slot table:

2.  
(10

```
sqlite> .schema time_slot
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)
);

sqlite> select * from time_slot;
time_slot_id  day  start_hr  start_min  end_hr  end_min
-----
A             M      8         0         8       50
A             W      8         0         8       50
A             F      8         0         8       50
B             M      9         0         9       50
B             W      9         0         9       50
B             F      9         0         9       50
C             M     11         0        11       50
C             W     11         0        11       50
C             F     11         0        11       50
D             M     13         0        13       50
D             W     13         0        13       50
D             F     13         0        13       50
E             T     10        30        11       45
E             R     10        30        11       45
F             T     14        30        15       45
F             R     14        30        15       45
G             M     16         0        16       50
G             W     16         0        16       50
G             F     16         0        16       50
H             W     10         0        12       30
sqlite>
```

pts.) Run the following queries on the university database you just created.

a. List all students.

Response:

```
select *
from student;
```

ID	name	dept_name	tot_cred
00128	Zhang	Comp. Sci.	102
12345	Shankar	Comp. Sci.	32
19991	Brandt	History	80
23121	Chavez	Finance	110
44553	Peltier	Physics	56
45678	Levy	Physics	46
54321	Williams	Comp. Sci.	54
55739	Sanchez	Music	38
70557	Snow	Physics	0
76543	Brown	Comp. Sci.	58

76653	Aoi	Elec. Eng.	60
98765	Bourikas	Elec. Eng.	98
98988	Tanaka	Biology	120

b. List only course\_ids of courses that are offered in Spring 2009.

Response:

```
select course_id
from section
where year=2009 and semester='Spring';
```

course_id
-----
CS-190
CS-190
EE-181

c. List only student names and how many more credits they have to take to complete their degree. Assume that the degree completion requirement is 124 credits for all students.

Response:

```
select      name,
            (124 - tot_cred) as credit_remaining
from student;
```

name	credit_remaining
-----	-----
Zhang	22
Shankar	92
Brandt	44
Chavez	14
Peltier	68
Levy	78
Williams	70
Sanchez	86
Snow	124
Brown	66
Aoi	64
Bourikas	26
Tanaka	4

d. Find the total number of instructors and their average salary.

Response:

```
select      count(ID) as num_instructors,
            avg(salary) as average_salary
from instructor;
```



num_instructors	average_salary
12	74833.3333333333

e. List only course\_ids of all courses that are either offered in Spring or Summer.

Response:

```
select course_id
from section
where semester='Spring' or semester='Summer';
```

course_id
BI0-101
BI0-301
CS-101
CS-190
CS-190
CS-315
CS-319
CS-319
EE-181
FIN-201
HIS-351
MU-199

f. List all rooms that have a capacity of at least 50 and utmost 100.

Response:

```
select *
from classroom
where capacity>=50 and capacity<=100;
```

building	room_number	capacity
Taylor	3128	70
Watson	120	50

g. List all instructors who have a name that begins with K.

Response:

```
select *
from instructor
where name like 'K%';
```

ID	name	dept_name	salary
45565	Katz	Comp. Sci.	75000



98345

Kim

Elec. Eng. 80000

h. List only student\_ids of students who have received a grade of A, A-, or B+ in any course.

Response:

```
select ID as student_id
from takes
where grade like 'A%' or grade='B+';
```

student\_id

-----

00128

00128

12345

12345

12345

45678

54321

54321

55739

76543

76543

98988

3. (8 pts.) Run these queries on the university database you have created.

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.

Response:

```
select distinct name
from student, takes
where course_id like 'CS%';
```

name

-----

Zhang

Shankar

Brandt

Chavez

Peltier

Levy

Williams

Sanchez

Snow

Brown

Aoi

Bourikas

Tanaka

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

Response:

```
select distinct student.ID, name
from student, takes
where takes.ID = student.ID
      and (year>2009 or (year=2009 and semester<>'Winter'));
```

ID	name
-----	-----
00128	Zhang
12345	Shankar
19991	Brandt
23121	Chavez
44553	Peltier
45678	Levy
54321	Williams
55739	Sanchez
70557	Snow
76543	Brown
76653	Aoi
98765	Bourikas
98988	Tanaka

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

Response:

```
select instructor.dept_name, max(salary)
from instructor, department
group by instructor.dept_name;
```

dept_name	max(salary)
-----	-----
Biology	72000
Comp. Sci.	92000
Elec. Eng.	80000
Finance	90000
History	62000
Music	40000
Physics	95000

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

Response:

```
select dept_name, min(max_salary)
from (
    select instructor.dept_name, max(salary) as max_salary
    from instructor, department
    group by instructor.dept_name);
```

dept_name	min(max_salary)
Music	40000

4. (4 pts.) Exercise 3.15 – parts b and c only.

Consider the bank database of Figure 3.19, where the primary keys are underlined. Construct the following SQL queries for this relational database:

- Find all customers who have an account at all the branches located in “Brooklyn”.
- Find out the total sum of all loan amounts in the bank.
- Find the names of all branches that have assets greater than those of at least one branch located in “Brooklyn”.

Response:

- ```
select sum(amount)
from loan;
```
- ```
select distinct branch_name
from branch
where asset > (
    select min(assets)
    from branch
    where branch_name='Brooklyn');
```

5. (4 pts.) Exercise 3.21 – parts a and c only.

Consider the library database of Figure 3.21. Write the following queries in SQL:

- Print the names of members who have borrowed any book published by “McGraw-Hill”.
- Print the names of members who have borrowed all books published by “McGraw-Hill”.
- For each publisher, print the names of members who have borrowed more than five books of that publisher.
- Print the average number of books borrowed per member. Take into account that if a member does not borrow any books, then that member does not appear in the borrowed relation at all.

Response:

- ```
select distinct member.name
from member, borrowed, book
where book.publisher='McGraw-Hill';
```
- ```
select member.name
from member, (
    select      publisher,
               count(borrowed.isbn),
```

```
        borrowed.memb_no as borrow_count
    from book, borrowed
    group by publisher) as publisher_count
where borrow_count > 5;
```

6. (2 pts.) Exercise 4.14

Show how to define a view tot\_credits (year, num\_credits), giving the total number of credits taken by students in each year.

Response:

```
create view tot_credits as
select year, sum(credits)
from course, section
group by year;
```

7. (2 pts.) Exercise 5.12

Consider the following relations for a company database:

- emp(ename, dname, salary)
- mgr(ename, mname) and the Java code in Figure 5.26, which uses the JDBC API. Assume that the userid, password, machine name, etc. are all okay. Describe in concise English what the Java program does. (That is, produce an English sentence like “It finds the manager of the toy department,” not a line-by-line description of what each Java statement does.)

Response: This program travels up the employee hierarchy, displaying a managers name, then the name of who manages them, then who manages them, until an employee without a manager is reached.