Eddie Gurnee

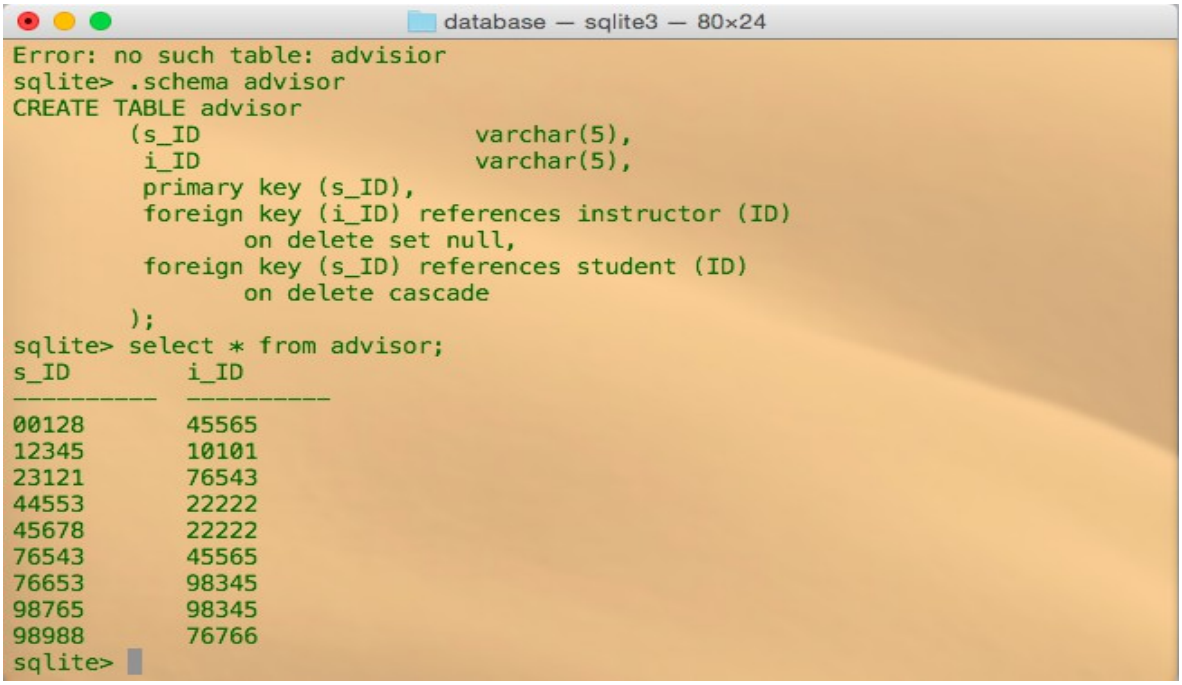
# Homework 2

## SQL

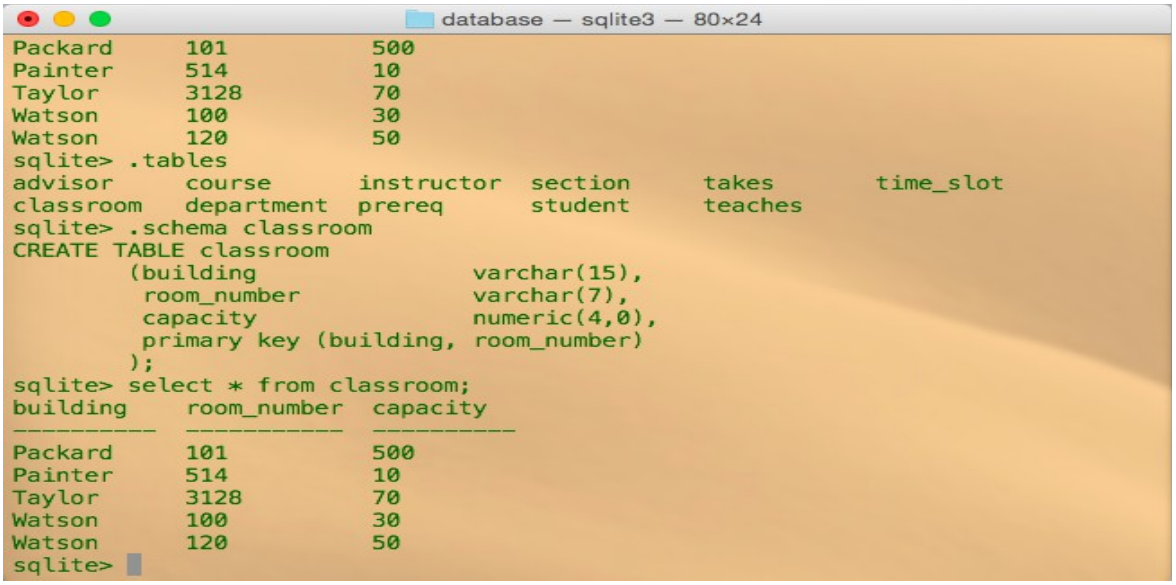
Total points: 40

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

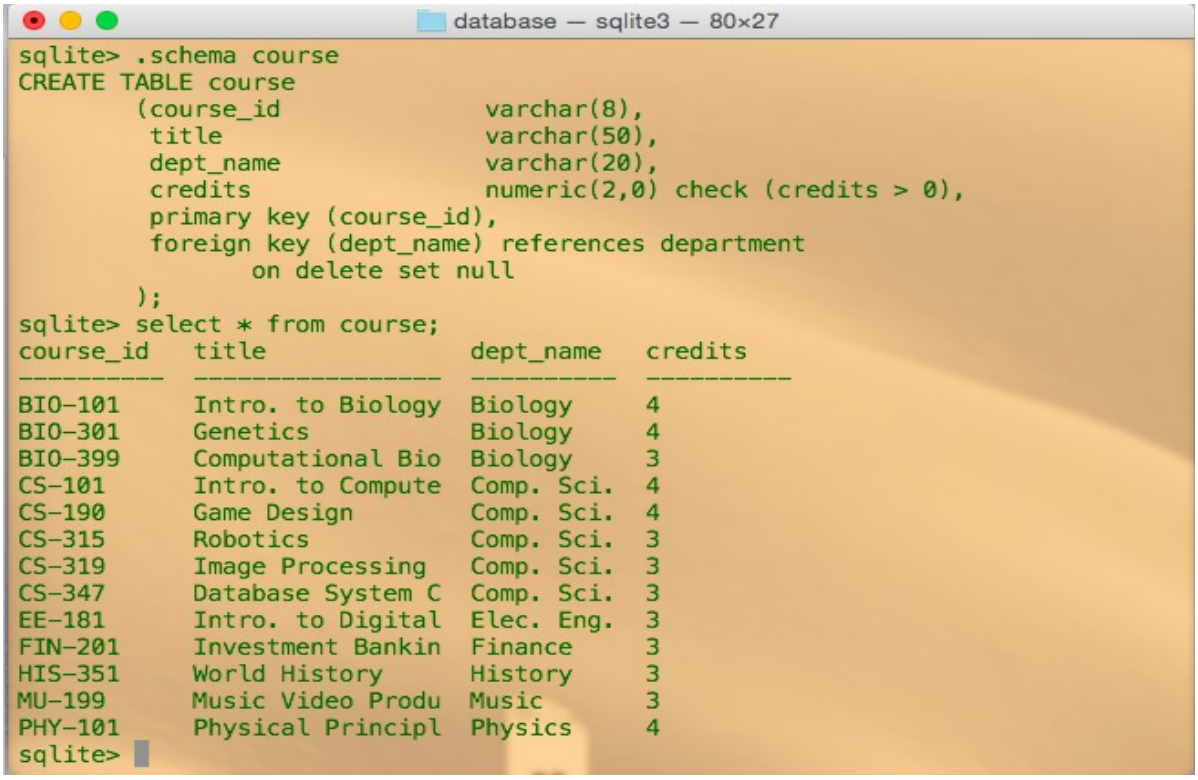
advisor table:



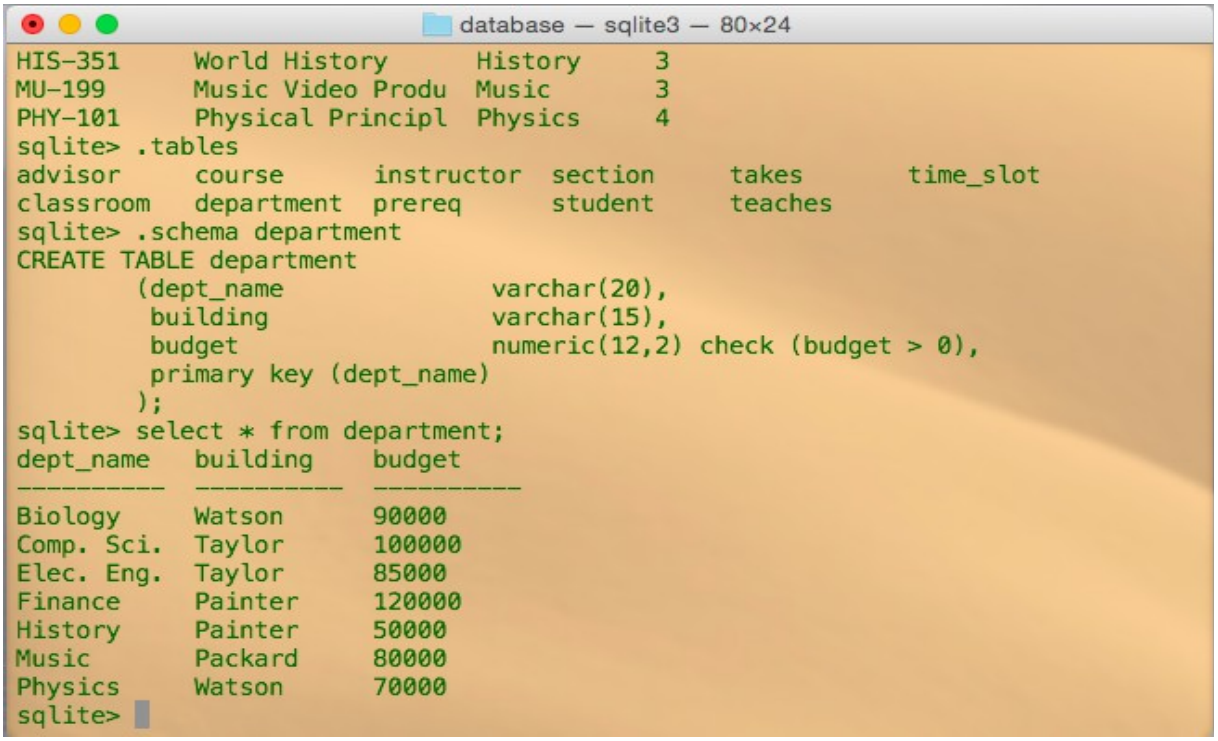

classroom table:



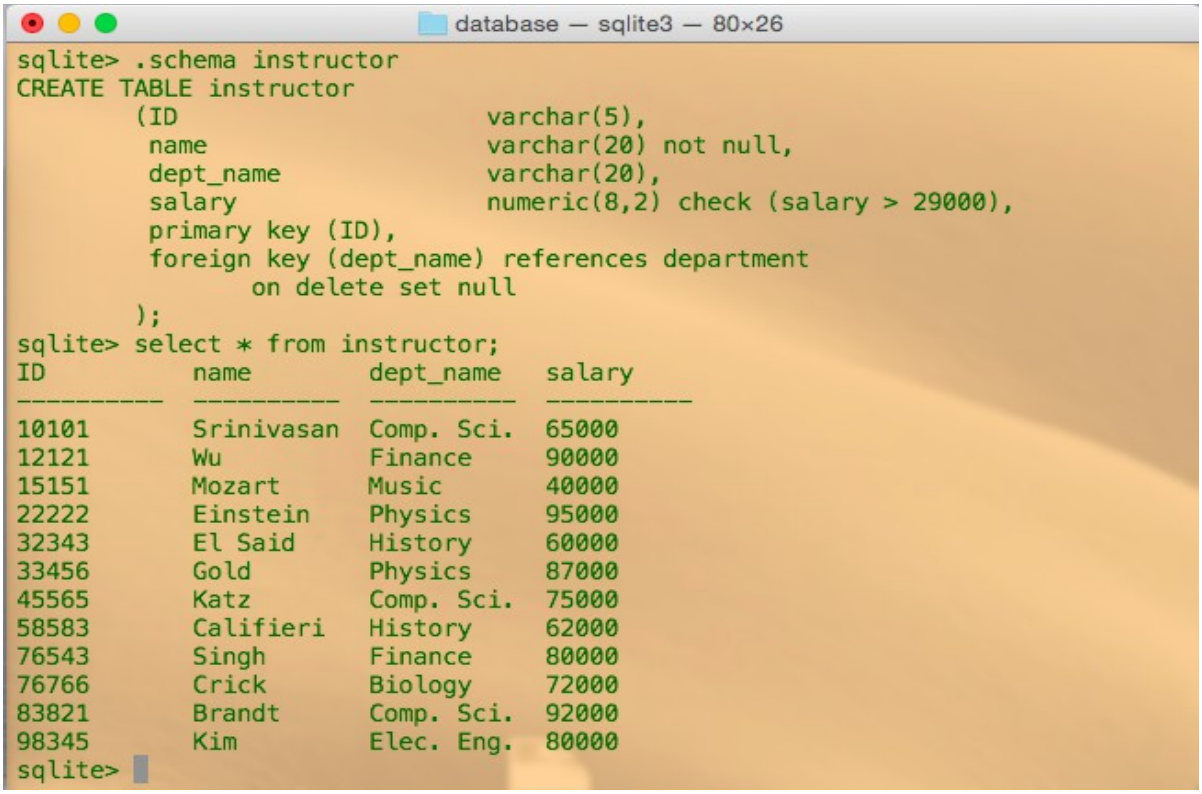
course table:



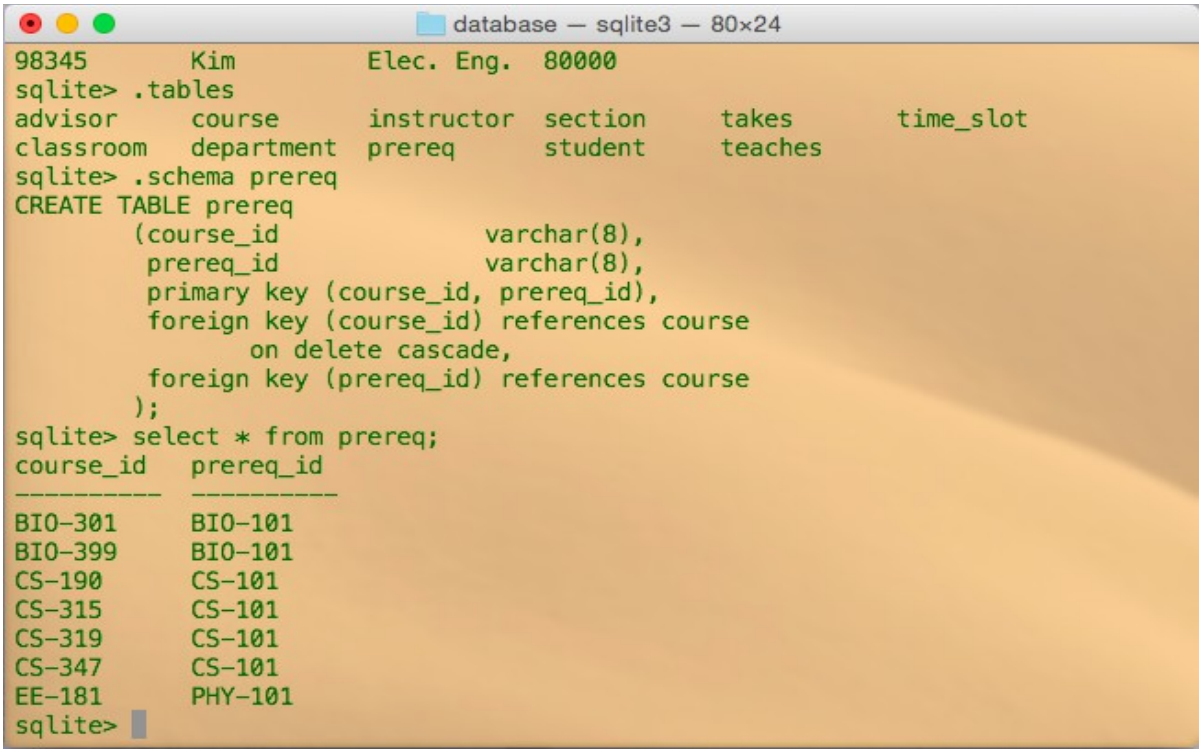

department table:



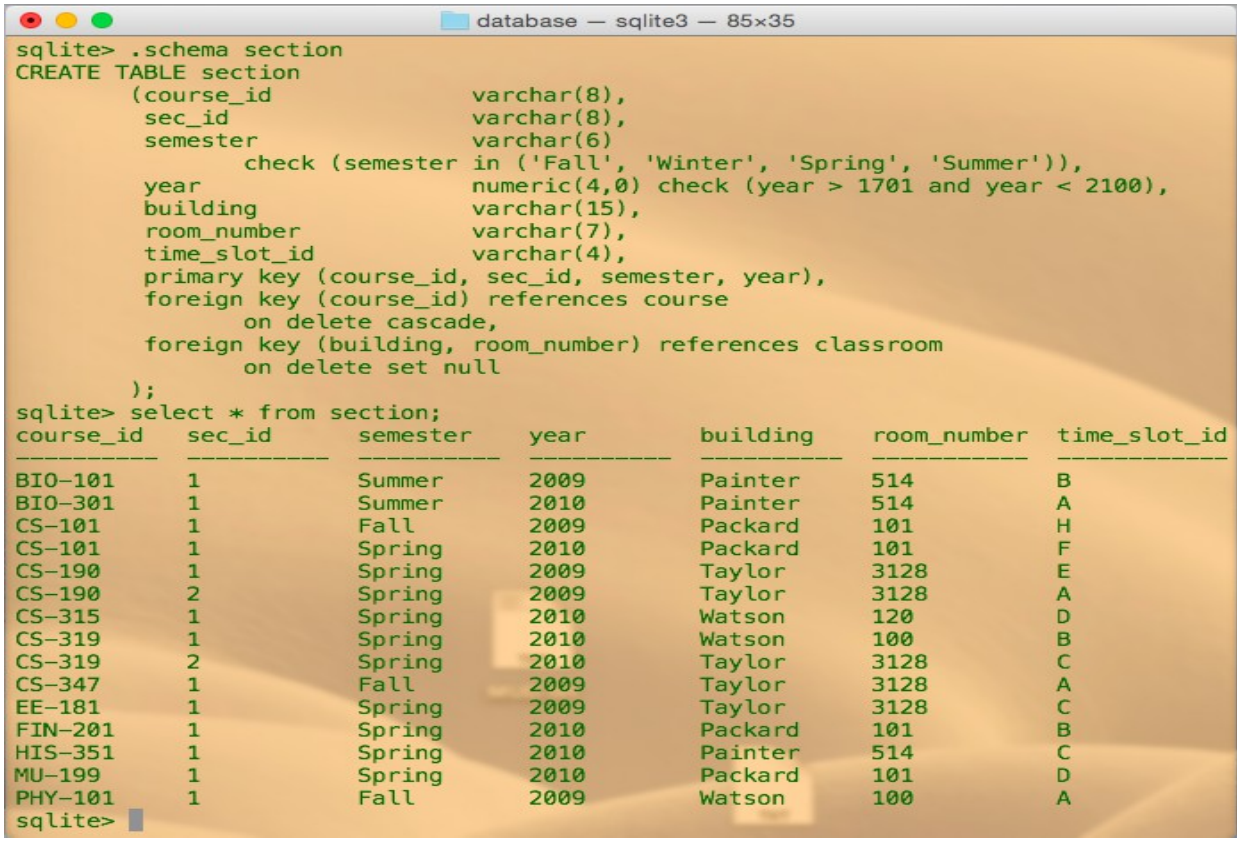
instructor table:




prereq table:

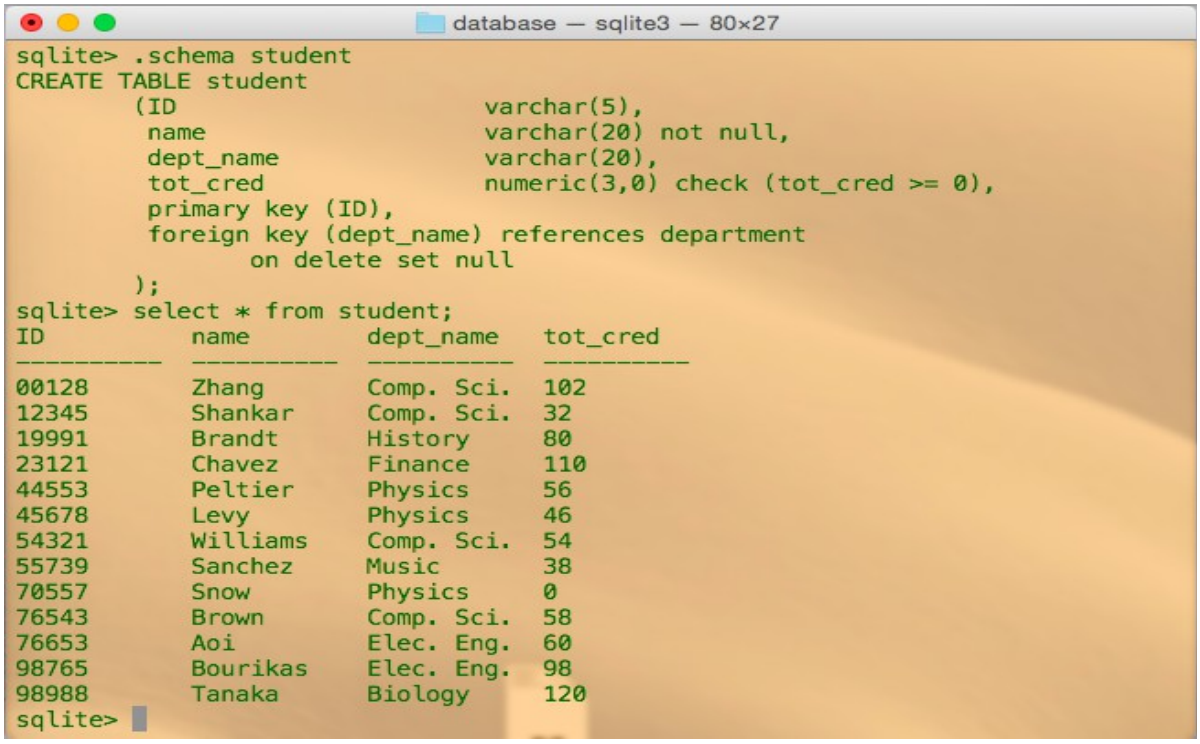


section table:



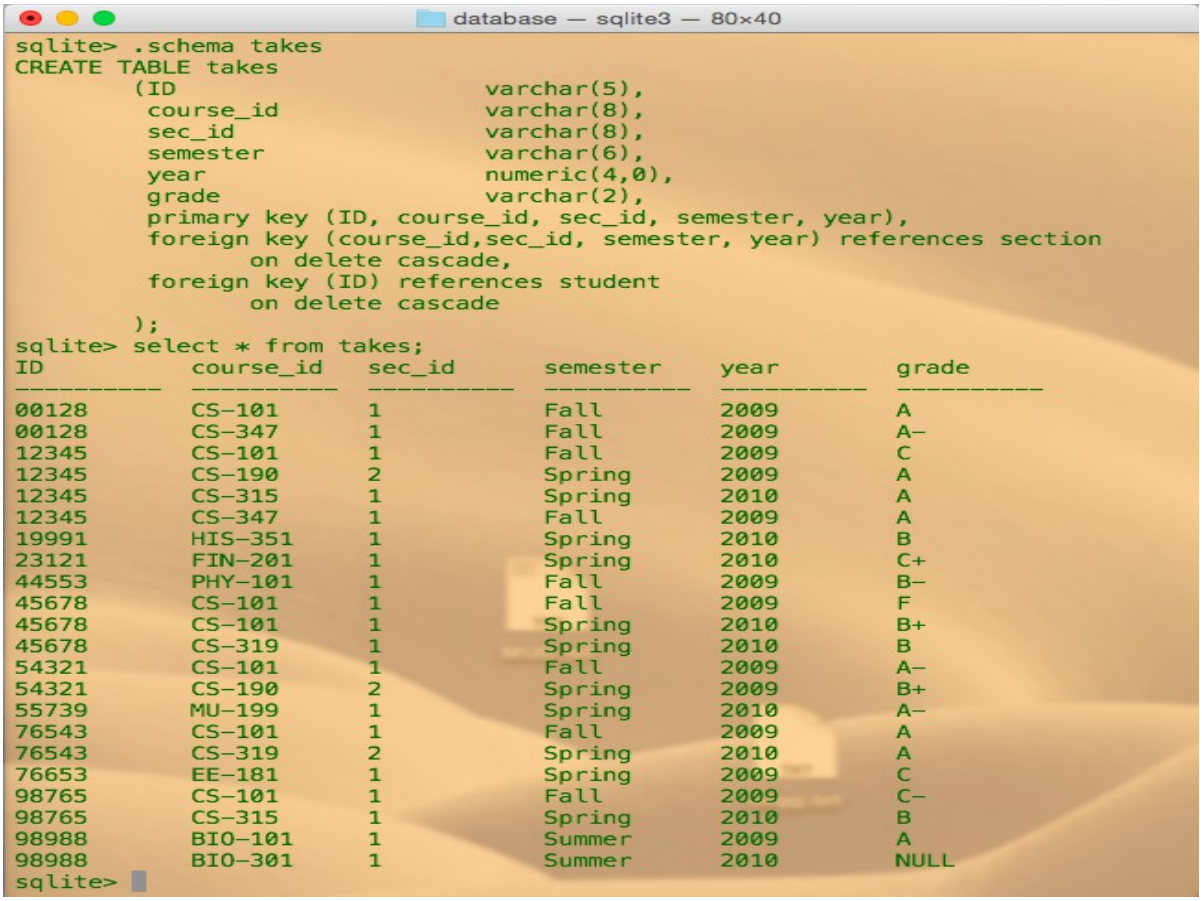
```
database — sqlite3 — 85×35
sqlite> .schema section
CREATE TABLE section
        (course_id              varchar(8),
         sec_id                 varchar(8),
         semester               varchar(6)
                check (semester in ('Fall', 'Winter', 'Spring', 'Summer')),
         year                   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:

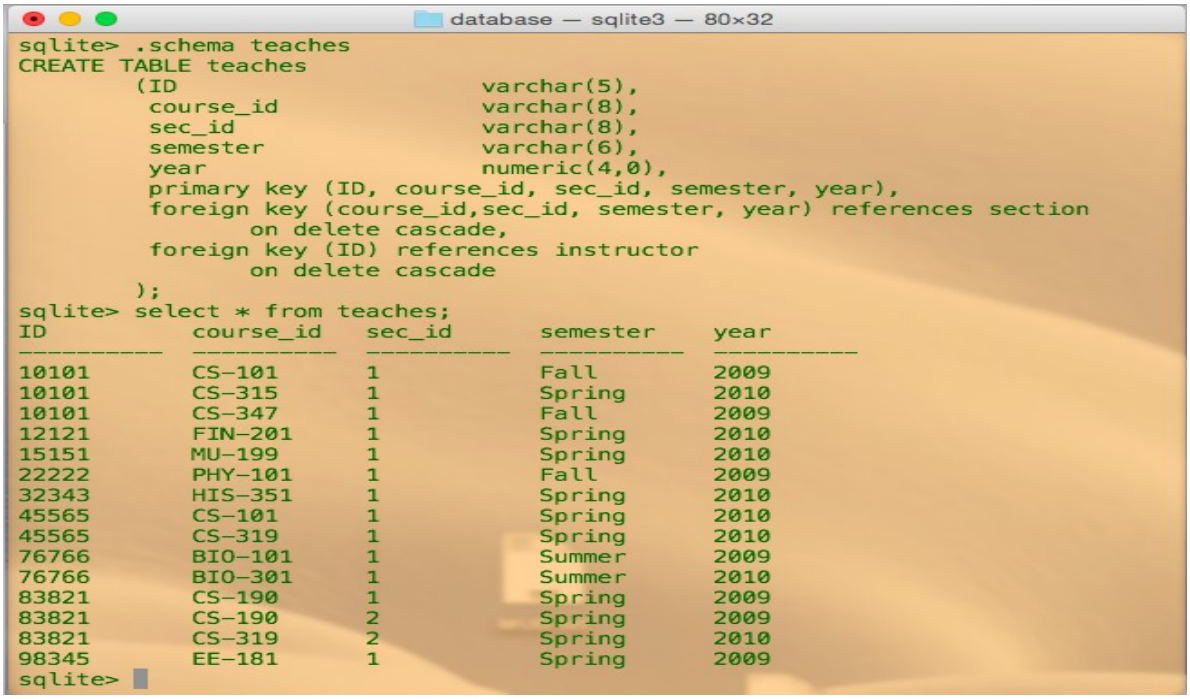


```
database — sqlite3 — 80×27
sqlite> .schema student
CREATE TABLE student
        (ID                     varchar(5),
         name                   varchar(20) not null,
         dept_name              varchar(20),
         tot_cred               numeric(3,0) check (tot_cred >= 0),
         primary 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:

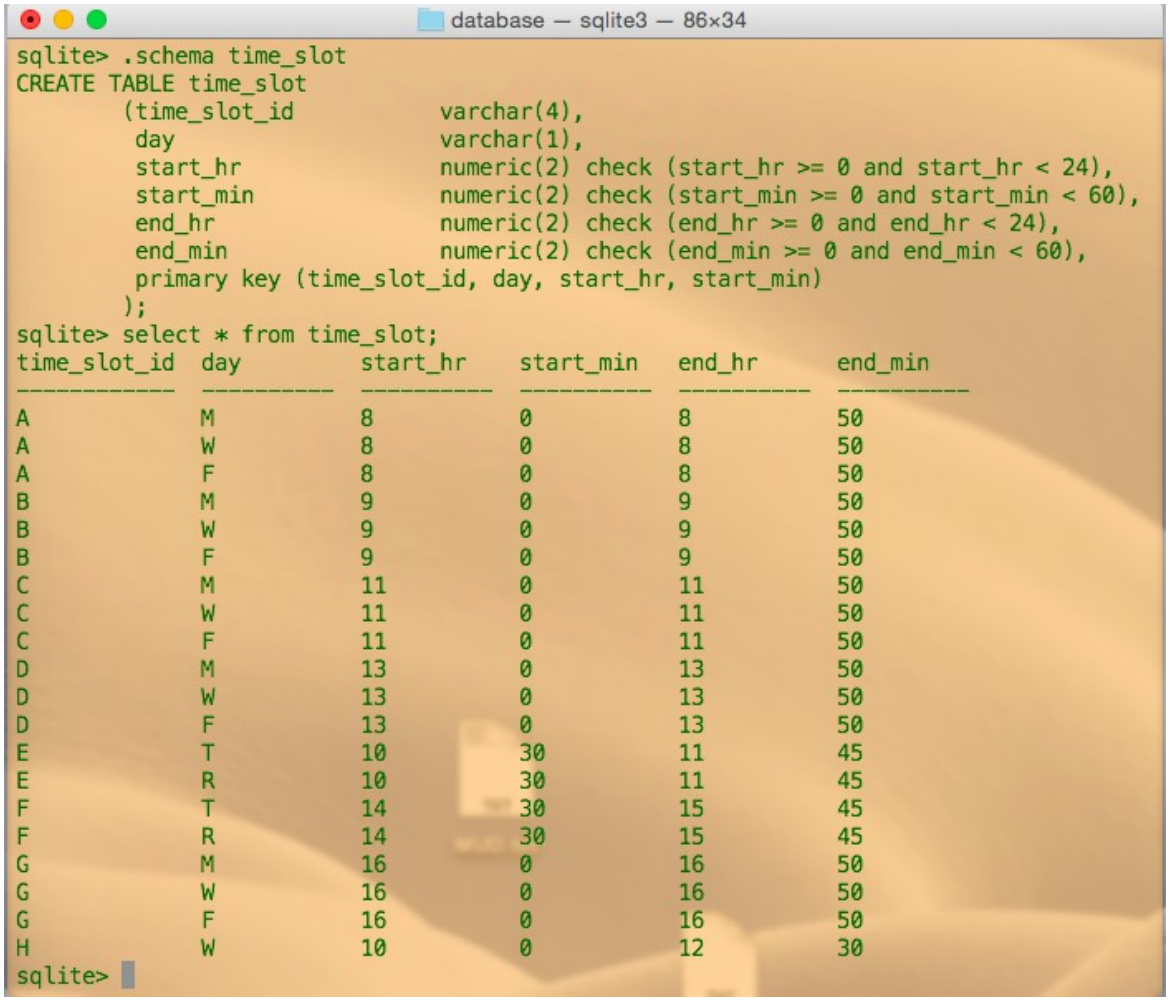



teaches table:



time_slot table:

2. (10




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
----------
BIO-101
BIO-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:

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”.

Response:

```
b.    select sum(amount)
      from loan;
c.    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:

a. Print the names of members who have borrowed any book published by “McGraw-Hill”.

b. Print the names of members who have borrowed all books published by “McGraw-Hill”.

c. For each publisher, print the names of members who have borrowed more than five books of that publisher.

d. 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:

```
a.    select distinct member.name
      from member, borrowed, book
      where book.publisher='McGraw-Hill';
c.    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.