Solutions to in-class exercise #1

Question 2

Here is how we can go about answering question #2, step-by-step.

"What courses meet in room CC2130 during session 20109?"

Step 0. Translate the questions into the terms used by our database.

Find course codes for all courses that meet in room with room code "CC2130" during the session with session code "20109".

Step 1. Figure out which columns in which tables have the information we need for the actual answer.

In case of question 2, we want *course codes.* This is available in the column **course_code** in the table **course**. So, our query will start with

```
select course.course_code
```

Step 2. Figure out how to express the conditions imposed in the question.

In this case, we want courses for which the **room_code** is "CC2130" and **session_code** is "20109". Those columns are in tables **room** and **session**. So, we can now write the end of our query too:

```
select course.course_code
...
where room.room_code="CC2130"
      and session.session_code="CC2130";
```

Note that we are using **and** because we want only rows that satisfy *both* of those conditions.

We will fill in the middle part of the query ("...") in the next step.

Step 3. Figure out what tables we need and how we can join them.

In our case, we know that we would *at least* need tables **course**, **room**, and **session**, because the first of those has the column we need in the end and the other two have columns that we use in the **where** clause. If we can join those three tables directly, perhaps this all we need. But it is possible that those three tables cannot be joined to each other. If so, we will need to use some other tables also.

Can we join course and room? Not really – there is no information that would allow us to join them. The rows in **room** are identified by **room_id**, but the course table does not have **room_id**. The courses are identified by **course_id**, but there is no **course_id** in room. It turns out that we have the same problem if we try to join **session** to **room**, or **course** to **session**.

So, what do we do?

It turns out that we have another table in the database: **course_instance**. This table represents a specific offering of a course. The reason we have this table is because a course like "CCT395H" is offered in different sessions (semesters). Every time it is offered, it may meet in different room. So, we do not want to record a single room number for "CCT395H" in general. We need to have a separate table for recording information that is specific to a particular offering of the course. This is our **course_instance** table. Each row in this table represents a particular course offered in a particular session, showing also the room where the course meets during this session.

We can join **course_instance** to **room** using **room_id**. After that, we can join the resulting table to **course** using **course_id**. Then we join the resulting table to **session**. Every time we do a join, we get more columns:

SQL	columns we get after this chunk of SQL	comment

`from course_instance`	`course_instance.instance_id`
	`course_instance.course_id`
	`course_instance.session_id`
	`course_instance.room_id`
	`course_instance.final_exam`
		Those are just the columns of **course_instance**.

`join room`
`  on course_instance.room_id`
`  = room.room_id`
	`course_instance.instance_id`
	`course_instance.course_id`
	`course_instance.session_id`
	`course_instance.room_id`
	`course_instance.final_exam`
	`room.room_id`
	`room.room_code`
	`room.capacity`
	`room.projector`
		Now we have all the columns of **course_instance** and all the columns of **room**. The columns shown in bold are the ones we are joining on. (That is, we will throw away all rows for which **course_instance.room_id** and **room.room_id** have different values.)

`join course`
`  on course_instance.course_id`
`  = course.course_id`
	`course_instance.instance_id`
	`course_instance.course_id`
	`course_instance.session_id`
	`course_instance.room_id`
	`course_instance.final_exam`
	`room.room_id`
	`room.room_code`
	`room.capacity`
	`room.projector`
	`course.course_id`
	`course.course_code`
	`course.course_name`
	`course.course_type`
		Now we have all the columns we got from the previous join *plus* all the columns from the **course** table. Note that in this step we are not joining **room** to **course**, but rather to the result of the previous join.

`join session`
`  on course_instance.session_id`
`  = session.session_id`
	`course_instance.instance_id`
	`course_instance.course_id`
	`course_instance.session_id`
	`course_instance.room_id`
	`course_instance.final_exam`
	`room.room_id`
	`room.room_code`
	`room.capacity`
	`room.projector`
	`course.course_id`
	`course.course_code`
	`course.course_name`
	`course.course_type`
	`session.session_id`
	`session.session_code`
	`session.start_date`
	`session.end_date`
		Notice that the table **course** that we used above (in **join course**) does not have the field **session_id**. However, we are *not* joining **session** to **course**. Rather, we are joining **session** to *the result of the previous joins*, which *does* have a column **session_id** (or, more precisely, **course_instance.session_id**).

The result of this last join is a table with 18 columns. It should have the same number of rows as our original **course_instance** table, each row representing an offering of a course in a particular session. This larger table, however, has columns not available in the original **course_instance** table. In particular, it has the two columns we need to do our selection: **room.room_id** and

session.session_id.

So, the database can now do the selection (i.e., apply the **where** clause), which will give us a table with 18 columns still, but with only those rows that represent course offered in the 20109 session that meet in room CC2130.

However, we are not interested in all the 18 columns. We are only interested in *course codes*. We achieve this with the **course.course_code** in the beginning of the query, right after **select**. Notice that we need to put list the columns we want in the beginning of the query, but this picking of specific columns ("projection") is really the *last* thing that the database will do.

So, our final query will look like this:

```
select course.course_code                                  this is what we want (this is applied last)

from course_instance                                       the first table to use

join room                                                  join it to another one
  on course_instance.room_id=room.room_id

join course                                                join the result to the third table
  on course_instance.course_id=course.course_id

join session                                               join the results to the forth
  on course_instance.session_id=session.session_id

where                                                      now pick the rows we want

  room.room_code="CC2130"                                  first, just those that have the right room code

  and session.session_code="20109"                         and from among those, the ones that have the
                                                           right session code

;                                                          a semi-column to show that we are done
```

Of course, since whitespace does not matter in SQL, we could write this query this way too:

```
select course.course_code from course_instance join room on
course_instance.room_id=room.room_id join course on
course_instance.course_id=course.course_id join session on
course_instance.session_id=session.session_id where room.room_code="CC2130" and
session.session_code="20109";
```

(It would be much harder to understand, though.)

Question 3

For question 3, we want select rows based on **building_code**, so we need to add the **building** table. We no longer have any columns from **room** in the **where** clause, but it turns out that we still need the **room** table, because there is no way of joining **building** to **course_instance** directly. So, to *list* the

codes of courses that meet in CCT in this session, we would do this:

```
select course.course_code
from course_instance
 join room on course_instance.room_id=room.room_id
 join course on course_instance.course_id=course.course_id
 join session on course_instance.session_id=session.session_id
 join building on room.building_id = building.building_id
where
  building.building_code="CCT"
  and session.session_code="20109";
```

However, the question asks us *how many* of such courses there are. So:

```
select count(course.course_code)
from course_instance
 join room on course_instance.room_id=room.room_id
 join course on course_instance.course_id=course.course_id
 join session on course_instance.session_id=session.session_id
 join building on room.building_id = building.building_id
where
  building.building_code="CCT"
  and session.session_code="20109";
```

Question 4

This question asks us what is the "maximum enrollment". However, we do not have a field called "enrollment". The reason for this is that there is no need to store this number – "enrollment" is the number of students currently registered for a course. So, let's rephrase the question:

For courses meeting in room with room code "CC2130" during the session with session code "20109", if we count the students enrolled in each course, what's the maximum count?

Now we can proceed to figuring out the beginning and end of the query:

```
select count(student.student_id)
...
where room.room_code="CC2130" and session.session_code="20109"
group by course_instance.course_id
order by count(student.student_id) desc limit 1;
```

(The end of this query is a bit more complicated, since we want to group, count, order and limit, but this is all last week's material.)

Now, what goes in the middle? We will need at least the following tables it looks like: **`student`**, **`room`**, **`session`**, **`course_instance`**. For joining **`room`**, **`session`**, and **`course_instance`** we can just copy and paste a chunk of our answer to question 2:

```
select count(student.student_id)
from course_instance
 join room on course_instance.room_id=room.room_id
```

```
 join session on course_instance.session_id=session.session_id
...
where room.room_code="CC2130" and session.session_code="20109"
group by course_instance.course_id
order by count(student.student_id) desc limit 1;
```

However, we also need the **student** table. This table cannot be joined directly to any of the tables we have joined already. (You should try to figure out why.) However, we also have another table: **enrollment**. This table creates an association with a student and a course instance. So, we can join enrollment to our collection of joined tables, and then join **student** to that.

```
select count(student.student_id)
from course_instance
 join room on course_instance.room_id=room.room_id
 join session on course_instance.session_id=session.session_id
 join enrollment on course_instance.instance_id=enrollment.instance_id
 join student on enrollment.student_id=student.student_id
where room.room_code="CC2130" and session.session_code="20109"
group by course_instance.course_id
order by count(student.student_id) desc limit 1;
```

If we want, we can simplify this somewhat by using **using** instead of the more explicit **join on**:

```
select count(student.student_id)
from course_instance
 join room using (room_id)
 join session using (session_id)
 join enrollment using (instance_id)
 join student using (student_id)
where room.room_code="CC2130" and session.session_code="20109"
group by course_instance.course_id
order by count(student.student_id) desc limit 1;
```

Question 5

Similar to question 4 but simpler, actually.

Question 6

To answer this question we would need to build on our query for question 4. We would join all seven tables, set a condition that session code is “20109”, group the rows by course instance ID *and* room capacity, then compare the counts to the value of room capacity. It would look something like this:

```
select room.room_code
from course_instance
 join room using (room_id)
 join session using (session_id)
 join enrollment using (instance_id)
 join student using (student_id)
where session.session_code="20109"
group by course_instance.course_id, room.capacity
having count(student.student_id) > room.capacity;
```