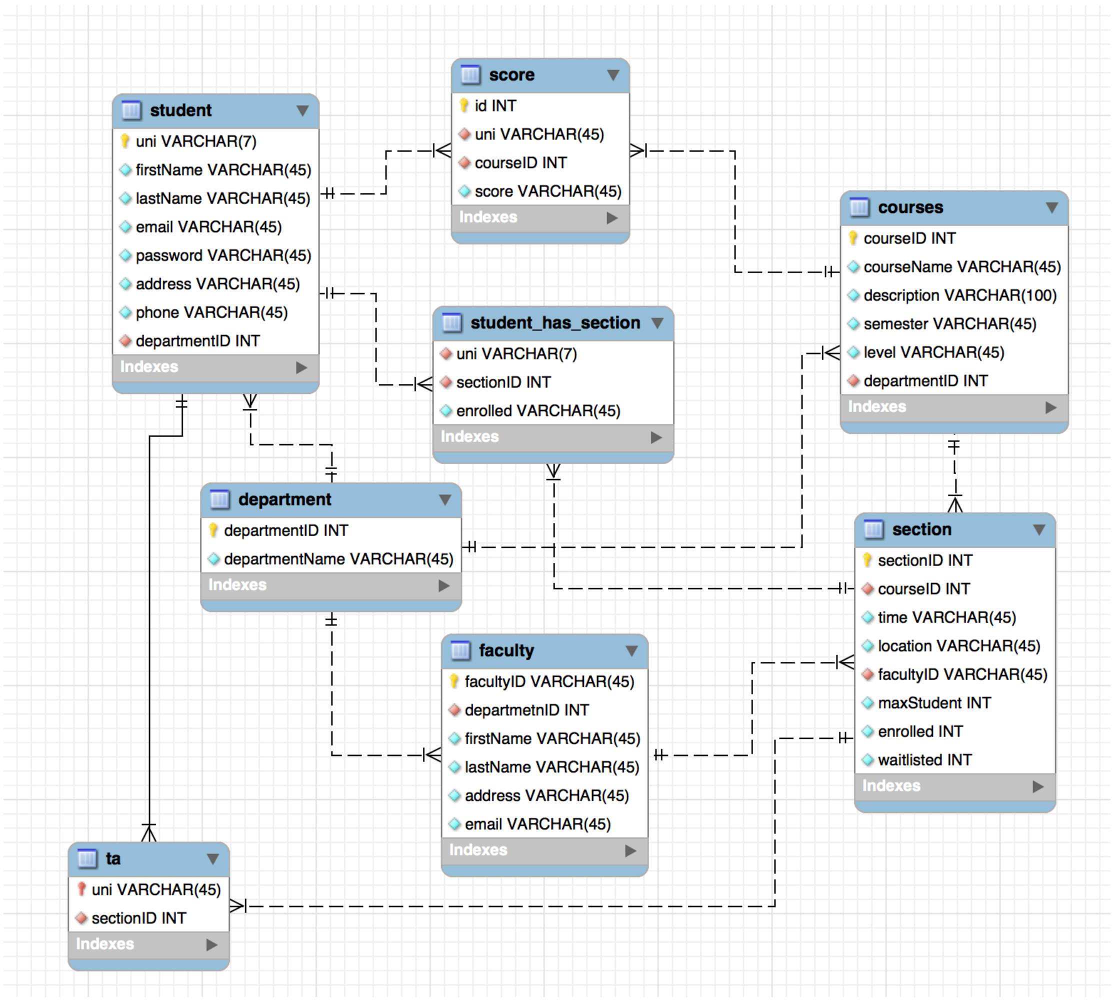
COMS 4111-Introduction to Databases Assignment 2

UNI: jc4833 Jiahe Chen

Data Model diagrams:



Explanation of the model design:

Each **department** must have a unique departmentID (primary key), and a departmentName.
Each **student** must have a unique UNI(primary key) and a profile. The profile must include first name, last name, email, password, address, phone and departmentID. departmentID is the foreign key which will link to department's departmentID. Its existence depends is only defined relative to department entity, so it is a weak entity.
Each **faculty** must have a unique facultyID(primary key), and a profile. The

profile must include departmendID, first name, last name, address and email. The departmentID is the foreign key which will link to department's departmentID. Its existence depends is only defined relative to department entity, so it is a weak entity.

Each **courses** must have a unique courseID(primary key), courseName, description, semester, level and departmendID. The departmendID is the foreign key which will link to department's departmentID. Its existence depends is only defined relative to department entity, so it is a weak entity.

Each **section** must have a unique sectionID(primary key), courseID which a foreign key which linked with courses' courseID to help identify the course information of this section, time, location, facultyID which is a foreign key which linked with faculty's facultyID to identify which professor teaches this section, maxStudent, enrolled(the number of enrolled student), waitlisted(the number of waitlisted student). And the value of enrolled should not be greater than maxStudent. The value of maxStudent, enrolled, and waitlisted should not be a negative number. Its existence depends in only defined relative to courses entity, so it is a weak entity.

Each **ta** must have a UNI and sectionID. UNI is the foreign key which will be linked to student entity's uni to identify this student's information and sectionID is also a foreign key which will be linked to section entity's sectionID to identify the section's information. Its existence depends is defined relative to student and section entity, so it is a weak entity.

Each **score** must have a UNI, courseID, and score. The score cannot be a negative number. UNI is the foreign key which will be linked with student entity's UNI, and courseID is also a foreign key which will be linked with courses entity' courseID. Its existence depends is defined relative to student entity and courses, so it is a weak entity.

For **student_has_section(associative entity),** it represents a relationship between student and sections, and it is used for many-to-many relationship. UNI is a foreign key which is linked with student entity's UNI, and sectionID is a foreign key which is linked with section entity's sectionID.

- Every student can have multiple score records, and choose many courses.
- A student cannot have multiple relationship with a course, for example, enrolled or waitlisted and TA, because if he is a TA, he will not be able to choose that course. And if he is a student of that class that means he has not learned that course, so he will not be able to be the TA.
- A student can only be a course's TA, that means if he has already been a TA of a section, he will no longer be able to be a TA of another section.
- A course can have many sections, but a section can only belong to one course. And a course can have multiple score records.
- A section must have an instructor.
- A faculty can teach many sections, but a section can only have a faculty.
- A department can have many students and faculties, but a student and a faculty can only belong to a faculty.

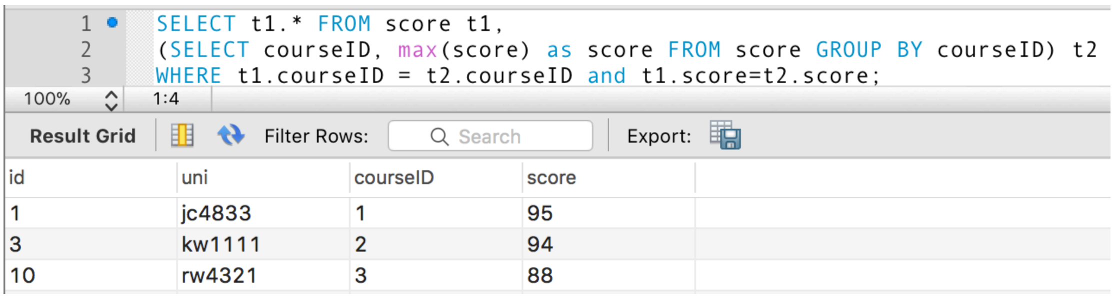
DDL:

The code is in the **createTable.sql** file.

User stories and screenshot with test data:

The code can be found in **userStory.md** file. And the test data can be found in **testdata.sql** file.

1. As a professor, I want to know who is the top student (with the highest mark) of the courses.



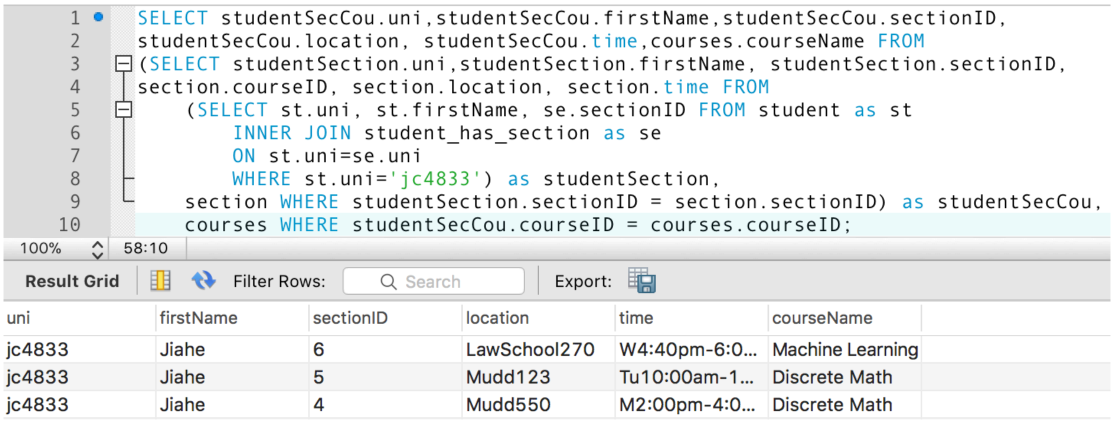
```
SELECT t1.* FROM score t1,
(SELECT courseID, max(score) as score FROM score GROUP BY courseID) t2
WHERE t1.courseID = t2.courseID and t1.score=t2.score;
```

100% 1:4

Result Grid Filter Rows: Search Export:

id	uni	courseID	score
1	jc4833	1	95
3	kw1111	2	94
10	rw4321	3	88

2. As a student whose UNI is jc4833, I want to know the courses I have selected. (Select specific course information (sectionID, section location, section time, and courName) and the student's firstName and uni from a student whose uni is jc4833.)



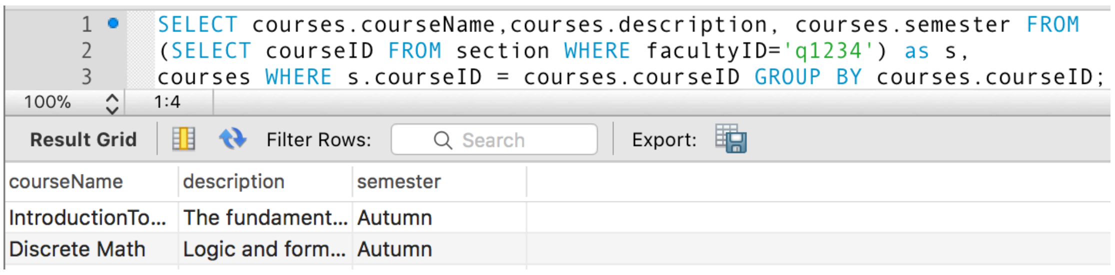
```
SELECT studentSecCou.uni,studentSecCou.firstName,studentSecCou.sectionID,
studentSecCou.location, studentSecCou.time,courses.courseName FROM
(SELECT studentSection.uni,studentSection.firstName, studentSection.sectionID,
section.courseID, section.location, section.time FROM
    (SELECT st.uni, st.firstName, se.sectionID FROM student as st
        INNER JOIN student_has_section as se
        ON st.uni=se.uni
        WHERE st.uni='jc4833') as studentSection,
    section WHERE studentSection.sectionID = section.sectionID) as studentSecCou,
    courses WHERE studentSecCou.courseID = courses.courseID;
```

100% 58:10

Result Grid Filter Rows: Search Export:

uni	firstName	sectionID	location	time	courseName
jc4833	Jiahe	6	LawSchool270	W4:40pm-6:0...	Machine Learning
jc4833	Jiahe	5	Mudd123	Tu10:00am-1...	Discrete Math
jc4833	Jiahe	4	Mudd550	M2:00pm-4:0...	Discrete Math

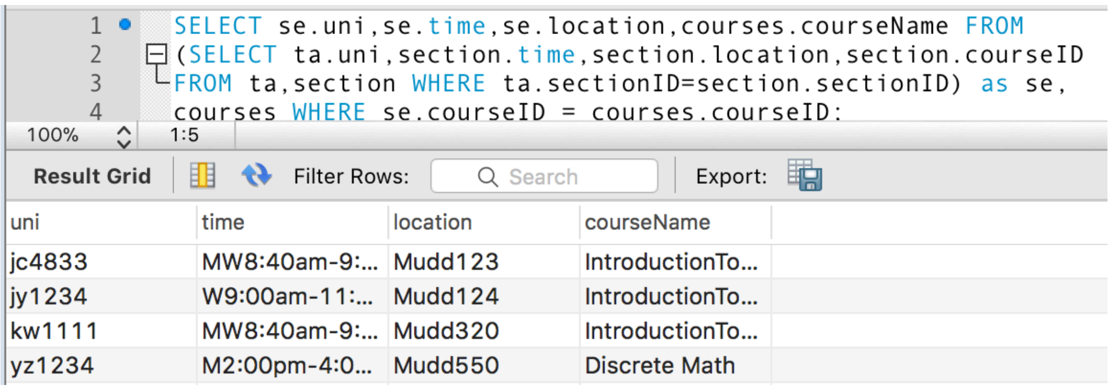
3. As a professor whose facultyID is ‘q1234’, I want to know the information of courses which I will teach. (Select the course information(courseName, description, semester) which will be taught by professor whose facultyID is 'q1234'.)



```sql
SELECT courses.courseName,courses.description, courses.semester FROM
(SELECT courseID FROM section WHERE facultyID='q1234') as s,
courses WHERE s.courseID = courses.courseID GROUP BY courses.courseID;
```

courseName	description	semester
IntroductionTo...	The fundament...	Autumn
Discrete Math	Logic and form...	Autumn

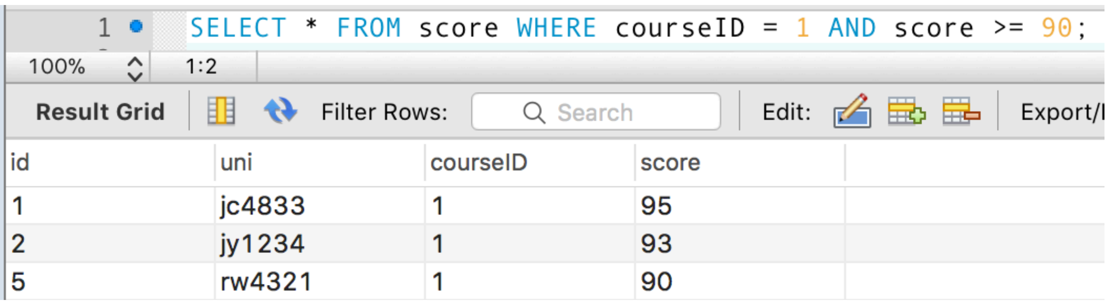
4. As an administrator, I want to see all TA's information. (Select all TA's information, their uni, course name, and session location, session address.)



```sql
SELECT se.uni,se.time,se.location,courses.courseName FROM
(SELECT ta.uni,section.time,section.location,section.courseID
FROM ta,section WHERE ta.sectionID=section.sectionID) as se,
courses WHERE se.courseID = courses.courseID;
```

uni	time	location	courseName
jc4833	MW8:40am-9:...	Mudd123	IntroductionTo...
jy1234	W9:00am-11:...	Mudd124	IntroductionTo...
kw1111	MW8:40am-9:...	Mudd320	IntroductionTo...
yz1234	M2:00pm-4:0...	Mudd550	Discrete Math

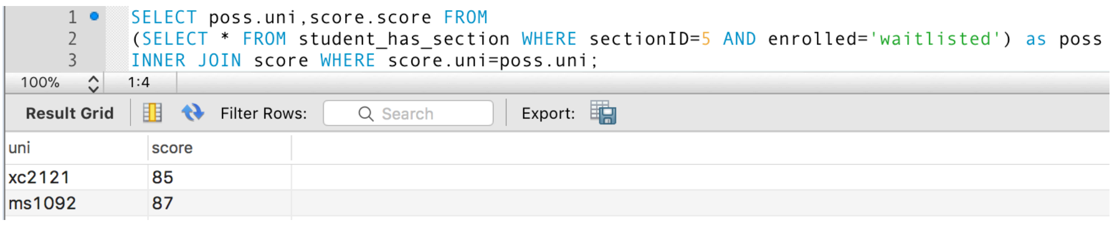
5. As a professor, I am looking for some students as TA, and the requirements are the student must have already taken course which courseID is 1 and the score must be >= 90. (Select all possible students who has already taken course(courseID=1) and score is >= 90 as TA.)



```sql
SELECT * FROM score WHERE courseID = 1 AND score >= 90;
```

id	uni	courseID	score
1	jc4833	1	95
2	jy1234	1	93
5	rw4321	1	90

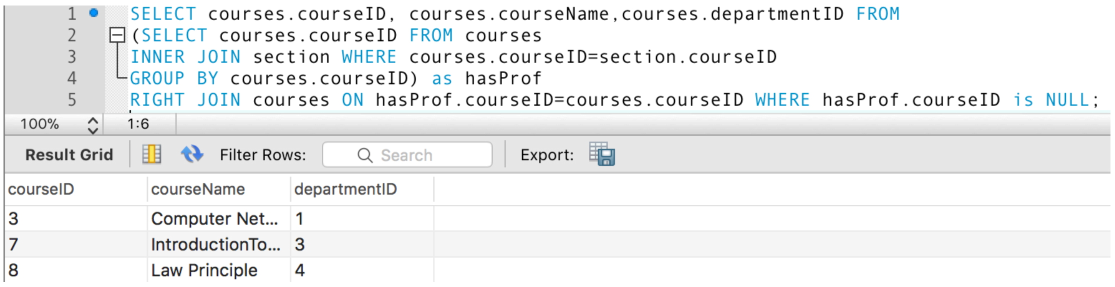
6. As a professor, I want to find some students who want to take 'Advanced Database' and have already taken 'Introduction to Databases'. And I want to see theses students' introduction to Databases's score. Maybe I will let some students with high mark to enroll in Advanced Databases.



```sql
SELECT poss.uni,score.score FROM
(SELECT * FROM student_has_section WHERE sectionID=5 AND enrolled='waitlisted') as poss
INNER JOIN score WHERE score.uni=poss.uni;
```

uni	score
xc2121	85
ms1092	87

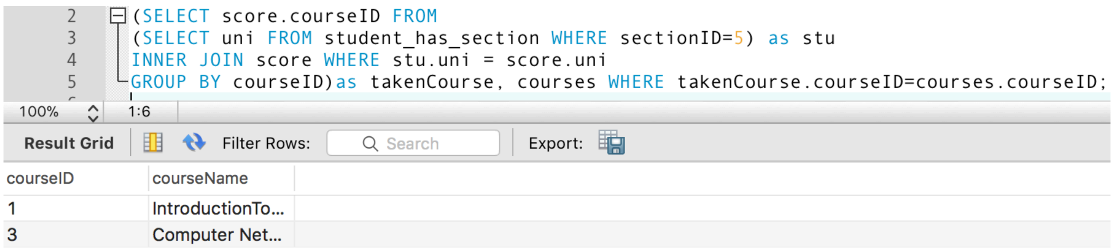
7. As an administrator, I want to find the courses which do not have any professor to teach.



```sql
SELECT courses.courseID, courses.courseName,courses.departmentID FROM
(SELECT courses.courseID FROM courses
INNER JOIN section WHERE courses.courseID=section.courseID
GROUP BY courses.courseID) as hasProf
RIGHT JOIN courses ON hasProf.courseID=courses.courseID WHERE hasProf.courseID is NULL;
```

```
courseID  courseName        departmentID
3         Computer Net...   1
7         IntroductionTo... 3
8         Law Principle     4
```

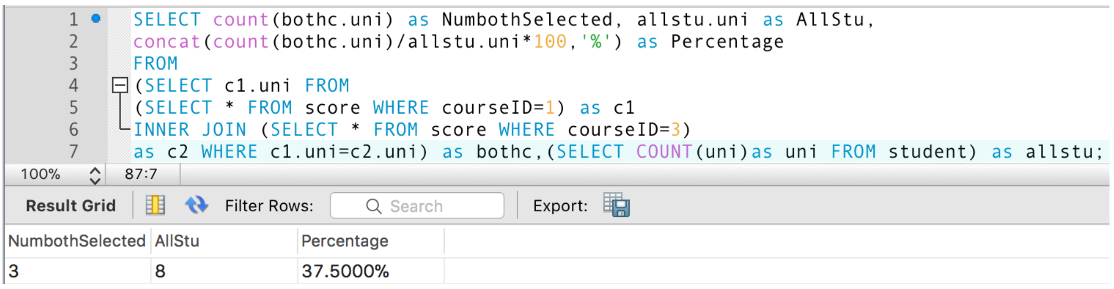
8. As a professor, I want to know which courses student in his class have taken.



```sql
(SELECT score.courseID FROM
(SELECT uni FROM student_has_section WHERE sectionID=5) as stu
INNER JOIN score WHERE stu.uni = score.uni
GROUP BY courseID)as takenCourse, courses WHERE takenCourse.courseID=courses.courseID;
```

```
courseID  courseName
1         IntroductionTo...
3         Computer Net...
```

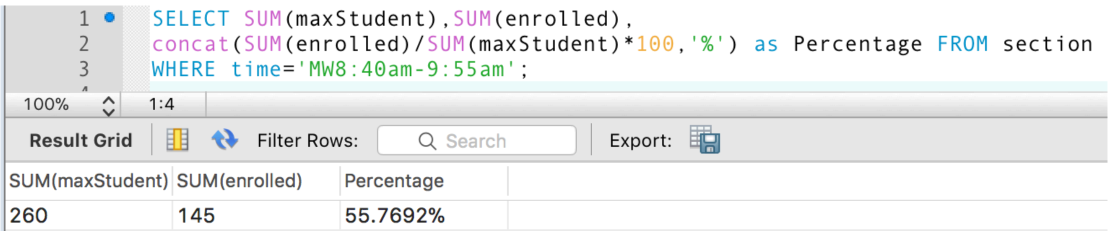
9. As an administrator, I want to know what percentage of student take course1 and course3.



```sql
SELECT count(bothc.uni) as NumbothSelected, allstu.uni as AllStu,
concat(count(bothc.uni)/allstu.uni*100,'%') as Percentage
FROM
(SELECT c1.uni FROM
(SELECT * FROM score WHERE courseID=1) as c1
INNER JOIN (SELECT * FROM score WHERE courseID=3)
as c2 WHERE c1.uni=c2.uni) as bothc,(SELECT COUNT(uni)as uni FROM student) as allstu;
```

```
NumbothSelected  AllStu  Percentage
3                8       37.5000%
```

10. As an administrator, I want to know the section enrollment based on course time of 'MW8:40am-9:55am'



```sql
SELECT SUM(maxStudent),SUM(enrolled),
concat(SUM(enrolled)/SUM(maxStudent)*100,'%') as Percentage FROM section
WHERE time='MW8:40am-9:55am';
```

```
SUM(maxStudent)  SUM(enrolled)  Percentage
260              145            55.7692%
```