October 3rd Class Participation Exercise

- Gary Miller

Discussion Questions

- By opening the database file in the db browser program the main tab displays all of the tables that exist within the database. These tables include course, department, instructor, student, takes, and teaches.
- Under the same database structure tab that displayed that tables within the database, it is possible to see the schema of each of the tables in this database. Alternatively the .schema command can be entered by running sqlite3 in the terminal and it will output the schema of each of the table that can be easily copy and pasted into another document. The schema of the table were of the the following:

  CREATE TABLE instructor (ID char(5), name varchar(20) not null, student varchar(20) not null, deptName varchar(20), salary numeric(8,2), primary key (ID), foreign key (deptName) references department); CREATE TABLE student (ID varchar(5), name varchar(20) not null, deptName varchar(20), totCred numeric(3,0), primary key (ID), foreign key (deptName) references department); CREATE TABLE course (courseId varchar(8) primary key, title varchar(50), deptName varchar(20), credits numeric(2,0), foreign key (deptName) references department); CREATE TABLE department (courseId varchar(8) primary key, courseType varchar(8), deptName varchar(8)); CREATE TABLE takes (ID char(5) NOT NULL, courseId varchar(8) NOT NULL, secId varchar(8) NOT NULL, semester varchar(8) NOT NULL, year int NOT NULL, grade text NOT NULL); CREATE TABLE teaches (ID char(5) primary key, courseId varchar(8), secId varchar(8), semester varchar(8), year varchar(8));
- By entering following the query:

  ```
  SELECT courseID FROM course INTERSECT SELECT courseID FROM department;
  ```

  The output gives the courseID for each course that is common to both the course table and the department table. The output of this query shows that CS100, CS200, and CS601 are common to both tables.
- In order to list all of the instructors and the students with whom are working with one another the following query was entered:

  ```
  select instructor.name, student.name from instructor
  inner join student on student.ID = student;
  ```

The output of this query returned the names of all the instructors and students that are working together and gave the following list of student instructor pairs:

```
"Miller"    "Michaels"
"Johnson"   "Michaels"
"Charleson" "Peterson"
"Thompson"  "Peterson"
"Mauler"    "Mullen"
"Jackson"   "Mullen"
"Chesterfield"  "Mullen"
"Jenkins"   "Scotts"
"William"   "Scotts"
"Wu"        "Michaels"
```

- In order to get a similar query as the previous one except the department name of the instructor should be included as well, the query was changed to:

  ```
  select instructor.name, instructor.deptName, student.name from instructor
  inner join student on student.ID = student;
  ```

The output of the new query was as follows:

```
"Miller"    "CompSci"   "Michaels"
"Johnson"   "CompSci"   "Michaels"
"Charleson" "CompSci"   "Peterson"
"Thompson"  "CompSci"   "Peterson"
"Mauler"    "Math"      "Mullen"
"Jackson"   "CompSci"   "Mullen"
"Chesterfield"  "CompBio"   "Mullen"
"Jenkins"   "CompBio"   "Scotts"
"William"   "Math"      "Scotts"
"Wu"        "Math"      "Michaels"
```

- In order to create a view table composed of instructors, their department name, and the name of the students that they are working with, the following query was entered:

  ```
  create view iSViewTable as select instructor.name, instructor.deptName, student.name
  from instructor inner join student on student.ID = student;
  ```

- In order to drop the view table that was created in the previous query, the following query was entered:

  ```
  drop table iSViewTable;
  ```

- In order to find all of the CS courses that exist in the teaches table the following query was used:

  ```
  select * from teaches where courseID like 'CS%';
  ```

The underscore allows for wild card matches therefore any course id that matches CS with any four characters after will yield a match. the output of the query was as follows:

```
"123461"    "CS-101"    "1" "Fall"      "2014"
"123462"    "CS-101"    "1" "Fall"      "2014"
"12347"     "CS-201"    "1" "Spring"    "2014"
"123481"    "CS-104"    "2" "Spring"    "2016"
"1234821"   "CS-105"    "2" "Summer"    "2016"
"1234821a"  "CS-106"    "2" "Summer"    "2016"
"1234821b"  "CS-107"    "2" "Summer"    "2016"
"1234822"   "CS-108"    "2" "Summer"    "2016"
"1234823"   "CS-109"    "2" "Summer"    "2016"
"12349"     "CS-401"    "1" "Fall"      "2009"
```

- In order to find the courses that are only associated with the English department in the teaches table, the following query was entered:

```
select * from teaches
where courseID like 'Eng%';
```

The output of this query returned the following data:

```
"12362" "Eng-101"   "1" "Spring"    "2016"
"12363" "Eng-102"   "1" "Spring"    "2016"
"12364" "Eng-201"   "1" "Spring"    "2016"
"12365" "Eng-202"   "1" "Spring"    "2016"
```

- In order to find only the years for the courses associated with the math department in the teaches table, the following query was entered:

```
select teaches.year from teaches
where courseID like 'Math%';
```

The output from this query returned the following data:

```
"2016"
"2016"
"2016"
"2016"
```

- In order to find the sections, semesters, and years of all of the 100-level CS courses in the teaches table, the following query was entered:

```
select teaches.secId, teaches.semester, teaches.year
from teaches where courseID like 'CS_1__';
```

The output form this query returned the following data:

```
"1" "Fall"      "2014"
"1" "Fall"      "2014"
"2" "Spring"    "2016"
"2" "Summer"    "2016"
"2" "Summer"    "2016"
"2" "Summer"    "2016"
"2" "Summer"    "2016"
```

```
"2" "Summer"    "2016"
"1" "Fall"      "2009"
```

- In order to find the distinct semesters for all 200-level CS classes, the following query was entered:

```
select distinct(teaches.semester) from teaches
where courseID like 'CS-2__';
```

The output of this query showed that 200-level CS classes are only offered in the spring semester.

- In order to determine the department that Professor Watson is teaching as well as her salary, the following query was entered:

```
select instructor.deptName, instructor.salary
from instructor where instructor.name == "Watson";
```

The output of this query showed that Professor Watson teaches in the CompSci department and their salary is $100,500.

- In order to determine the instructors who have a salary greater than $100,000, the following query was entered:

```
select * from instructor where salary > 100000;
```

The output of this query returned the following data:

```
"10108" "Jenkins"   "S4"    "CompBio"   "102000"
"10110" "Watson"    "S4"    "CompSci"   "100500"
"10111" "Nelson"    "S5"    "CompBio"   "103000"
"10112" "Farber"    "S5"    "CompBio"   "101000"
```

- In order to return the name of each department and the average salary of the instructors in each department in only a single query, the following query was entered:

```
select distinct(instructor.deptName), avg(instructor.salary)
from instructor
group by instructor.deptName;
```

It was found that the 'group by' command must be added to this query in order to list out each of the department names and the average salary. Without this command it would simply list out the department name of the last entry and the average salary of that particular department. The output from this query was:

```
"Biology"   "82500.0"
"CompBio"   "100750.0"
"CompSci"   "97800.0"
"Math"      "90000.0"
```

- In order to select the name, department and salary of instructors that make more than their department average, the following list queries were

entered:

```
select name, deptName, salary
from instructor where salary > (select avg(salary)
from instructor where deptName == "Biology") and deptName == "Biology";

select name, deptName, salary from instructor where salary > (select avg(salary)
from instructor where deptName == "CompBio") and deptName == "CompBio";

select name, deptName, salary from instructor where salary > (select avg(salary)
from instructor where deptName =="CompSci") and deptName == "CompSci";

select name, deptName, salary from instructor where salary > (select avg(salary)
from instructor where deptName == "Math") and deptName == "Math";
```

The output of these several queries were:

```
"Maximillian"   "Biology"   "86000"
"Jenkins"   "CompBio"   "102000"
"Nelson"    "CompBio"   "103000"
"Farber"    "CompBio"   "101000"
"Thompson"  "CompSci"   "100000"
"Jackson"   "CompSci"   "99900"
"Watson"    "CompSci"   "100500"
"Mauler"    "Math"      "99000"
```

- In order to update the student ID of Beuller to "XS5", the following query was entered:

```
update student set ID = "XS5" where name == "Beuller";
```

The DB browser interface output that the query executed successfully and by looking into the browse data tab, we can see the ID for Beuller reflects this change.

- In order to reflect Professor Maximillian's new promotion such that his new salary is $100,000, his name is now MaxiMillion, and his department is Psychology the following query was entered:

```
update instructor set salary = "100000", name = "MaxiMillion",  deptName = "Psychology"
where name == "Maximillian";
```

The DB browser interface output that the query executed successfully and by looking into the browse data tab, we can see the ID for Beuller reflects this change.

- In order to reflect Professor Jenkins"s new promotion such that his salary is now $103,000 the following query was entered:

```
update instructor set salary = "103000"
where name == "Jenkins";
```

The DB browser interface output that the query executed successfully and by looking into the browse data tab, we can see the ID for Beuller reflects this change.

- In order to show that the records for Professor MaxiMillion and Professor Jenkins have been updated successfully, the following query was entered:

```
select * from instructor where name == "MaxiMillion" or name == "Jenkins";
```

The output of the query was:

```
"10108" "Jenkins"   "S4"    "CompBio"   "103000"
"10114" "MaxiMillion"   "S5"    "Psychology"    "100000"
```

Challenges Faced In the Assignment

Some of the key challenges faced in this assignment were associated with queries that were related to joining multiple tables. The most difficult aspect of this was simply familiarizing myself with some of these concepts due to my absence during the lectures that covered this material. After a little reading and implementation practice, I feel I have over come these particular challenges now and have a much better understanding of some of these concepts after this assignment. Another thing that I found particularly helpful for some of the more difficult queries was writing them down on a piece of paper so I focused more on each individual command and field in a particular query. Also when querying for multiple fields or things that needed particular values or met certain conditions, I found it quite helpful to break it down and consider each part on its own before combing them together. For example, when listing out each department in the instructor table with the average salary made by instructors in that department, I found wrote a query to list out the distinct department names, then wrote a query that listed out the average salaries, and finally combined them together to get the data formatted the way I desired.