Ryan Young

CSE 4701

2-25-2020

Project 1 Part 2

1. *For each of the following query in English, provide a SQL statement, and show its evaluation result. No view is allowed to use and the query must be a single statement.*
   a.) *The names of students who have failed in Physics(SCORE < 60).*
   SQL Statements:
   select distinct SNAME
   ->from student, course, result
   ->where result.score < 60 and result.CNO = 15 and result.sno = student.sno;
   Result:

```
+---------+
| SNAME   |
+---------+
| J.Brown |
+---------+
```

   b.) *The average score in Physics.*
   SQL Statements:
   select AVG(score) from course, result where cname = 'physcis and course.cno = result.cno;
   Result:

```
+------------+
| AVG(score) |
+------------+
|    73.0000 |
+------------+
```

   c.) *The names of students who performed lower than average in Physics.*
   SQL Statements:
   select distinct SNAME
   -> from student, course, result
   -> where course.cname = 'phsyics and result.sno = student.sno and course.cno = result.cno and result.score < (select avg(score) from course, result where course.cname = 'physics' and course.cno = result.cno);
   Result:

```
+-------------+
| SNAME       |
+-------------+
| T.Smith     |
| J.Brown     |
| D. E. Knuth |
+-------------+
```

   d.) *The names of students who performed lower than average in each course. Show both student name (SNAME) and course number (CNO) so that in which course(s) the student*

performed lower than the average is (are) clear. Do no treat Null in any special way. Explain how the system takes care of NULL in the average computation.

SQL Statement:

```
select SNAME, result.CNO
   -> from result as r, student as s join(select avg(score) as course_avg, cno from result group by cno) as avgs on r.cno = avgs.cno
   ->  where s.sno = r.sno and r.score < avgs.course.avg;
```

Result:

SNAME	CNO
M.Roberts	2
D. E. Knuth	2
I.Blake	2
D.Davisson	2
D. E. Knuth	3
D. E. Knuth	12
D.Davisson	12
J.Brown	13
T.Smith	15
J.Brown	15
D. E. Knuth	15

The system takes care of NULL in the average computation by ignoring it.

e.) *The numbers (CNO) and names (CNAME) of course taken by J. Brown, sorted on CNO in descending order.*

SQL Statement:

```
select CNAME, course.CNO
   -> from student, course, result
   -> where student.SNAME = 'J.Brown' and student.sno = result.sno and result.cno = course.cno
      -> order by cno desc
```

Result:

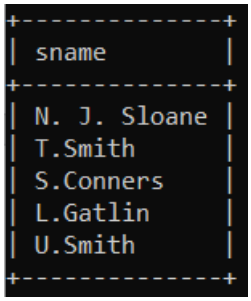
CNAME	CNO
Physics	15
Mathematics III	13

f.) *The names of female students in any course taken by J. Brown. Do it using a nested query with "IN".*

SQL Statement:

```
select distinct student.sname
   -> from student, course, result
   -> where course.cno in(select result.cno from result, student where student.sname ='J.Brown' and result.sno = student.sno)and student.sex = 'F';
```

Result:



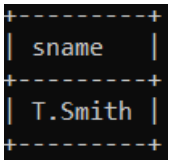
sname
N. J. Sloane
T.Smith
S.Conners
L.Gatlin
U.Smith

g.) *The names of female students who take every course taken by J. Brown and no other course.*

SQL Statement:

```
select distinct s1.sname
   -> from student s1, student s2, result r1, result r2
   -> where s1.sno = r1.sno
   -> and s2.sno = r2.sno
   -> and r1.cno = r2.cno
   -> and s1.sex ='f'
   -> and s2.sname = 'J.Brown'
   -> and s1.sno not in (select sno from result where cno not in(select cno from student,
result where sname ='J.Brown' and student.sno = result.sno));
```

Result:



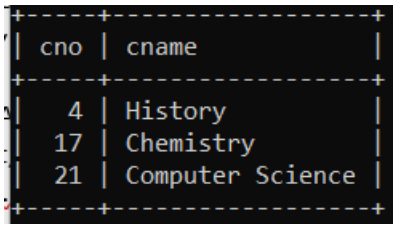
sname
T.Smith

h.) *The course for which there are no grades as no one took them.*

SQL Statement:

```
Select distinct cno, cname
-> from course
-> where course.cno not in(select distinct result.cno from result);
```

Result:



cno	cname
4	History
17	Chemistry
21	Computer Science

i.) *The numbers (SNO) and names (SNAME) of students having no scores at all (NULL score value is considered as no score in addition to the case having no entry in RESULT).*

SQL Statement:

```
select distinct student.sno, student.sname
   -> from student, result
   -> where score is NULL and student.sno = result.sno or student.sno not in(select result.sno
from result);
```

Result:

sno	sname
1	N. J. Sloane
22	S.Conners
11	N.Park

j.) *The possible names of Miss U. Smith's boyfriend. We know that he has taken every course that was taken by Miss Smith.*

SQL Statement:

```
select distinct s1.sname
  -> from student s1, student s2, result r1, result r2
  -> where s1.sno = r1.sno
  -> and s2.sno = r2.sno
  -> and r1.cno = r2.cno
  -> and s1.sex = 'M'
  -> and s2.sname = 'U.Smith'
  -> and s1.sno not in (select sno from result where cno not in (select cno from student, result where sname = 'U.Smith' and student.sno = result.sno))
  -> and s2.sno no in (seect sno form result where cno not in (select cno form student, result where student.sno = s1.sno and student.sno = result.sno));
```

Result:

sname
D.Davisson

k.) *The average score for each course in ascending order by CNO.*

SQL Statement:

```
select course.cno, avg(score)
  -> from course, result
  -> where course.cno = result.cno
  -> group by course.cno
  -> order by course.cno asc;
```

Result:

cno	avg(score)
2	75.1667
3	69.0000
12	70.8000
13	73.0000
15	73.0000

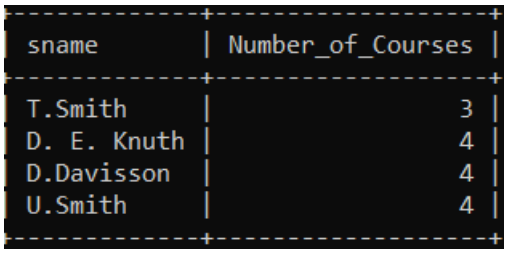
l.) *For each student who took more than two course, list the name of the student and the number of course the student took.*

SQL Statement:

```
select student.sname, count(result.sno) as Number_of_Courses
```

-> from student, course, result
-> where student.sno = result.sno and course.cno = result.cno
-> group by result.sno
-> having count(result.sno) > 2;

Result:



sname	Number_of_Courses
T.Smith	3
D. E. Knuth	4
D.Davisson	4
U.Smith	4

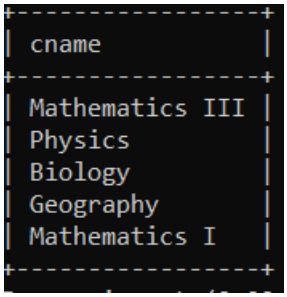
2. *Given the query, "the course names which are taken by the students living at Whitney,"*

a.) Write an SQL statement.

SQL Statement:

select distinct cname
-> from student, result, course
-> where student.sno = result.sno and course.cno = result.cno and student.address = 'Whitney';

Result:



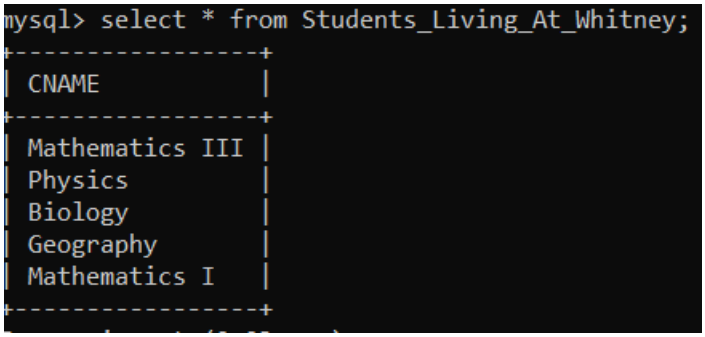
cname
Mathematics III
Physics
Biology
Geography
Mathematics I

b.) Define a view for "the student living at Whitney," and rephrase (a) by using the view.

SQL Statement:

create view Students_Living_At_Whitney as
-> select distinct CNAME
-> from course, result, student
-> where student.sno = result.sno and course.cno = result.cno and student.address = 'Whitney';

Result:



```
mysql> select * from Students_Living_At_Whitney;
+-----------------+
| CNAME           |
+-----------------+
| Mathematics III |
| Physics         |
| Biology         |
| Geography       |
| Mathematics I   |
+-----------------+
```

c.) Show how the systems catalog stores the view definition.

```
mysql> select * from Students_Living_At_Whitney;
+-----------------+
| CNAME           |
+-----------------+
| Mathematics III |
| Physics         |
| Biology         |
| Geography       |
| Mathematics I   |
+-----------------+
```

3. *The following points should be observed and discussed. Do this with SQL.*
   a.) *The STUDENT relation is to be modified by adding an additional column SEMESTER. Investigate how it can be done in SQL and discuss what you have discovered on this. You may try to make an update on this new column, for example, by adding semester values for 3 or 4 entries of the table and see how it turns out. Does the system change the table when modifying the schema, or when updating the table?*
   You can add new columns to tables in general by using the Alter Table, Add [Column] query. I also discovered that there can be multiple ways to achieve different tasks in MySQL and SQL in general. The system changes the table when the schema is modified.

   b.) *Do the following two problems:(1) Create an index with UNIQUE option on SNO of STUDENT. Show that you successfully created the index. (2) This time try to create an index with UNIQUE option on ADDRESS of STUDENT. Discuss your result of this attempt.*
   Creating an index with unique option on sno of student: create unique index stu_indx on student(sno asc);. This will create a unique index named stu_indx on the sno column of student with the key sno will be in ascending order. Creating an index with unique option on address of student: create unique index stu_indx on student(address). This will create a unique option on the address column of the student table and the key will be address. This query when run will throw a duplicated key error. This is because some students live at the same address. We can fix this error by: deleting or updating the key values, changing the key column to one that does not have duplicates, and we could temporarily combine the column with another column so that it will be a unique key and not have duplicates.

   c.) *How does the system treat updates on views? For example, create a view for STUDENT (SEX, ADDRESS) and do the followings: insert (F, E. Quad);delete (M, Whitney); and update (M, E. Quad) to (F, E. Quad). Can you do these?*
   You cannot insert (F, E.Quad):

```
mysql> insert into student_view values('F', 'E.Quad');
ERROR 1423 (HY000): Field of view 'project_1.student_view
' underlying_table doesn't have a default value
```

   You cannot delete (M, Whitney):

```
mysql> delete from student_view where sex = 'M' and addre
ss = 'Whitney';
ERROR 1451 (23000): Cannot delete or update a parent row:
 a foreign key constraint fails (`project_1`.`result`, CO
NSTRAINT `result_ibfk_2` FOREIGN KEY (`sno`) REFERENCES `
student` (`sno`))
```

You can update (M, E.Quad) to (F, E.Quad):

```
mysql> update student_view
    -> set sex = 'F'
    -> where sex = 'M' and address = 'E.Quad';
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```