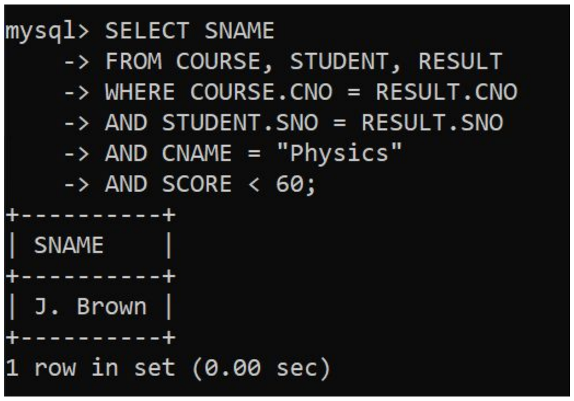
CSE4701 - Databases
Project 1 Part 2 - 2020-02-25

1. For each of the following query in English, provide a SQL statement, and show its evaluated 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).

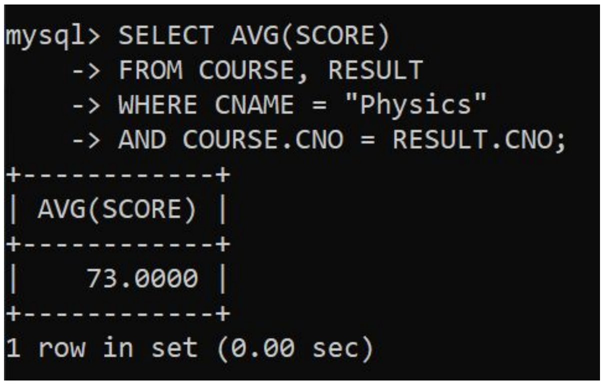
SQL:	Machine-generated solution:

SELECT SNAME
FROM COURSE, STUDENT, RESULT
WHERE COURSE.CNO = RESULT.CNO
AND STUDENT.SNO = RESULT.SNO
AND CNAME = “Physics”
AND SCORE < 60;
	

```
mysql> SELECT SNAME
    -> FROM COURSE, STUDENT, RESULT
    -> WHERE COURSE.CNO = RESULT.CNO
    -> AND STUDENT.SNO = RESULT.SNO
    -> AND CNAME = "Physics"
    -> AND SCORE < 60;
+----------+
| SNAME    |
+----------+
| J. Brown |
+----------+
1 row in set (0.00 sec)
```

(b) The average score in Physics.

SQL:	Machine-generated solution:

SELECT AVG(SCORE)
FROM COURSE, RESULT
WHERE CNAME = “Physics”
AND COURSE.CNO = RESULT.CNO;
	

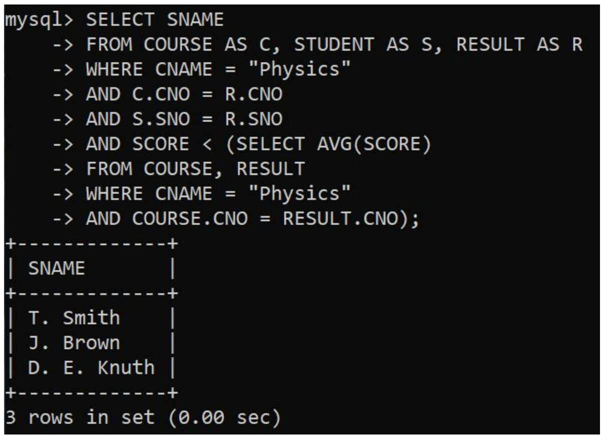
```
mysql> SELECT AVG(SCORE)
    -> FROM COURSE, RESULT
    -> WHERE CNAME = "Physics"
    -> AND COURSE.CNO = RESULT.CNO;
+------------+
| AVG(SCORE) |
+------------+
|    73.0000 |
+------------+
1 row in set (0.00 sec)
```

(c) The names of students who performed lower than average in Physics.

SQL:

```
SELECT SNAME
FROM COURSE AS C, STUDENT AS S, RESULT AS R
WHERE CNAME = “Physics”
AND C.CNO = R.CNO
AND S.SNO = R.SNO
AND SCORE < (SELECT AVG(SCORE)
FROM COURSE, RESULT
WHERE CNAME = “Physics”
AND COURSE.CNO = RESULT.CNO);
```

Machine-generated solution:



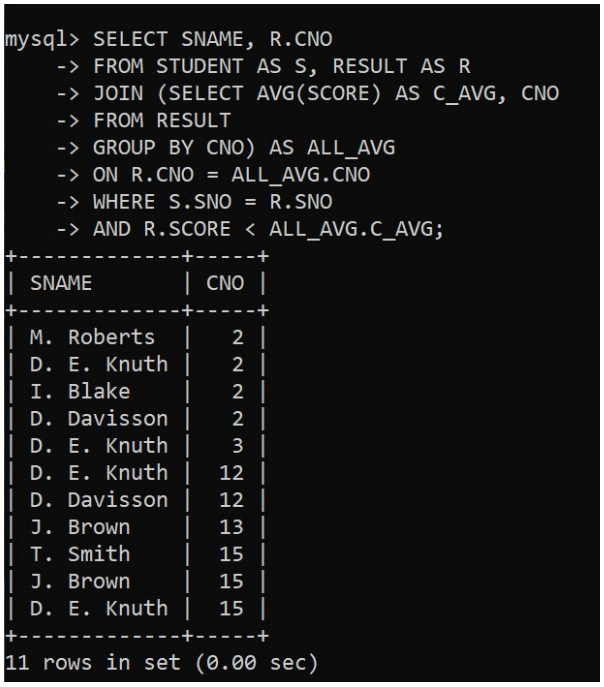
```
mysql> SELECT SNAME
    -> FROM COURSE AS C, STUDENT AS S, RESULT AS R
    -> WHERE CNAME = "Physics"
    -> AND C.CNO = R.CNO
    -> AND S.SNO = R.SNO
    -> AND SCORE < (SELECT AVG(SCORE)
    -> FROM COURSE, RESULT
    -> WHERE CNAME = "Physics"
    -> AND COURSE.CNO = RESULT.CNO);
+-------------+
| SNAME       |
+-------------+
| T. Smith    |
| J. Brown    |
| D. E. Knuth |
+-------------+
3 rows in set (0.00 sec)
```

(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 course average is (are) clear. Do not treat Null in any special way. Explain how the system takes care of Null in the average computation.

SQL:

```
SELECT SNAME, R.CNO
FROM STUDENT AS S, RESULT AS R
JOIN (SELECT AVG(SCORE) AS C_AVG, CNO
        FROM RESULT
        GROUP BY CNO) AS ALL_AVG
ON R.CNO = ALL_AVG.CNO
WHERE S.SNO = R.SNO
AND R.SCORE < ALL_AVG.C_AVG;
```

Machine-generated solution:

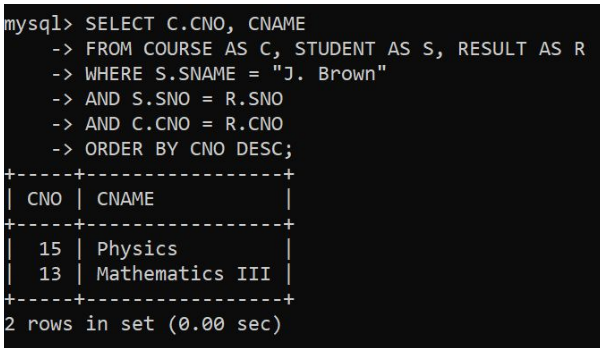


```
mysql> SELECT SNAME, R.CNO
    -> FROM STUDENT AS S, RESULT AS R
    -> JOIN (SELECT AVG(SCORE) AS C_AVG, CNO
    -> FROM RESULT
    -> GROUP BY CNO) AS ALL_AVG
    -> ON R.CNO = ALL_AVG.CNO
    -> WHERE S.SNO = R.SNO
    -> AND R.SCORE < ALL_AVG.C_AVG;
+-------------+-----+
| SNAME       | CNO |
+-------------+-----+
| 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 |
+-------------+-----+
11 rows in set (0.00 sec)
```

The NULL is “ignored”, thus the SCORE value is not considered in any way, and the student is not considered as an additional count in the denominator of the average. This way of taking care of the NULL essential takes that the student didn't take the exam, or it can be interpreted that the student didn't take the course in the first place.

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

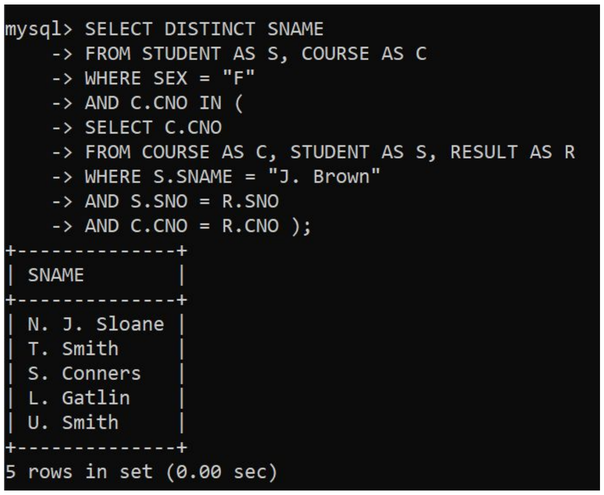
SQL:	**Machine-generated solution:**

SELECT C.CNO, CNAME
FROM COURSE AS C, STUDENT AS S, RESULT AS R
WHERE S.SNAME = "J. Brown"
AND S.SNO = R.SNO
AND C.CNO = R.CNO
ORDER BY CNO DESC;
	

```
mysql> SELECT C.CNO, CNAME
    -> FROM COURSE AS C, STUDENT AS S, RESULT AS R
    -> WHERE S.SNAME = "J. Brown"
    -> AND S.SNO = R.SNO
    -> AND C.CNO = R.CNO
    -> ORDER BY CNO DESC;
+-----+-----------------+
| CNO | CNAME           |
+-----+-----------------+
|  15 | Physics         |
|  13 | Mathematics III |
+-----+-----------------+
2 rows in set (0.00 sec)
```

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

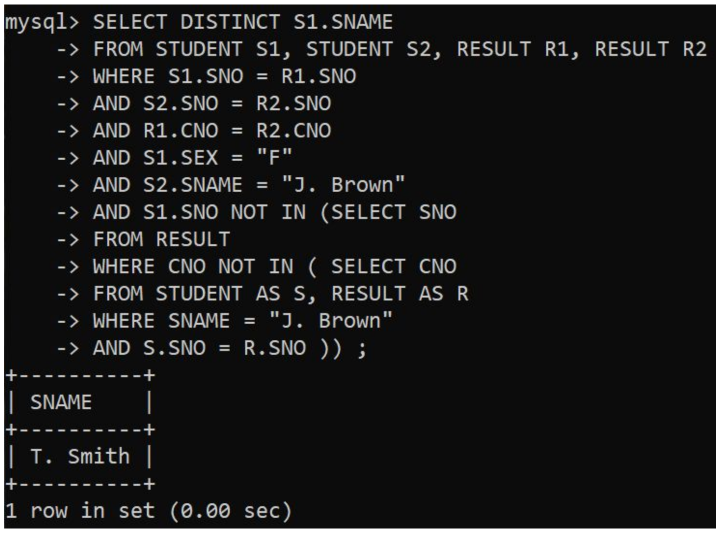
SQL:	**Machine-generated solution:**

SELECT DISTINCT SNAME, CNAME
FROM STUDENT AS S, COURSE AS C
WHERE SEX = "F"
AND C.CNO IN (
    SELECT SNAME, CNO
    FROM STUDENT AS S, RESULT AS R
    WHERE SNAME = "J. Brown"
    AND S.SNO = R.SNO)
	

```
mysql> SELECT DISTINCT SNAME
    -> FROM STUDENT AS S, COURSE AS C
    -> WHERE SEX = "F"
    -> AND C.CNO IN (
    -> SELECT C.CNO
    -> FROM COURSE AS C, STUDENT AS S, RESULT AS R
    -> WHERE S.SNAME = "J. Brown"
    -> AND S.SNO = R.SNO
    -> AND C.CNO = R.CNO );
+--------------+
| SNAME        |
+--------------+
| N. J. Sloane |
| T. Smith     |
| S. Conners   |
| L. Gatlin    |
| U. Smith     |
+--------------+
5 rows in set (0.00 sec)
```

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

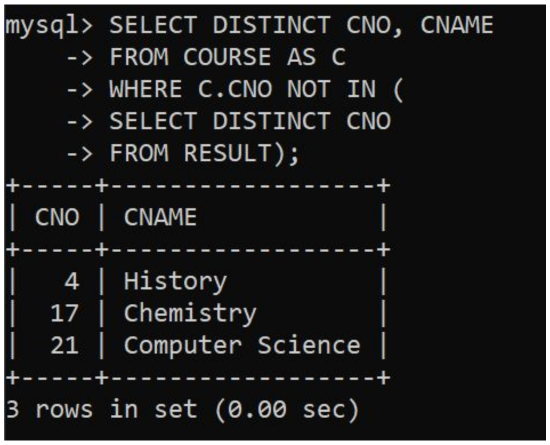
SQL:	Machine-generated solution:

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 AS S, RESULT AS R
                WHERE SNAME = “J. Brown”
                AND S.SNO = R.SNO)) ;
	

```
mysql> 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 AS S, RESULT AS R
    -> WHERE SNAME = "J. Brown"
    -> AND S.SNO = R.SNO )) ;
+----------+
| SNAME    |
+----------+
| T. Smith |
+----------+
1 row in set (0.00 sec)
```

(h) The courses for which there are no grades as no one took them.

SQL:	Machine-generated solution:

SELECT DISTINCT CNO, CNAME
FROM COURSE AS C
WHERE C.CNO NOT IN (
        SELECT DISTINCT CNO
        FROM RESULT);
	

```
mysql> SELECT DISTINCT CNO, CNAME
    -> FROM COURSE AS C
    -> WHERE C.CNO NOT IN (
    -> SELECT DISTINCT CNO
    -> FROM RESULT);
+-----+------------------+
| CNO | CNAME            |
+-----+------------------+
|   4 | History          |
|  17 | Chemistry        |
|  21 | Computer Science |
+-----+------------------+
3 rows in set (0.00 sec)
```

CSE4701 - Databases

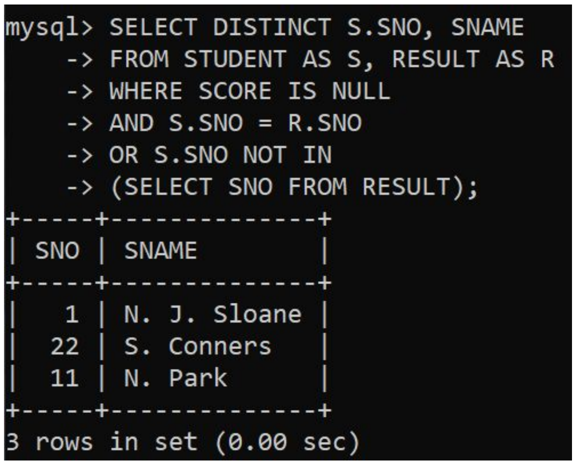
Project 1 Part 2 - 2020-02-25

(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:

```
SELECT DISTINCT S.SNO, SNAME
FROM STUDENT AS S, RESULT AS R
WHERE SCORE IS NULL
AND S.SNO = R.SNO
OR S.SNO NOT IN
(SELECT SNO FROM RESULT);
```

Machine-generated solution:



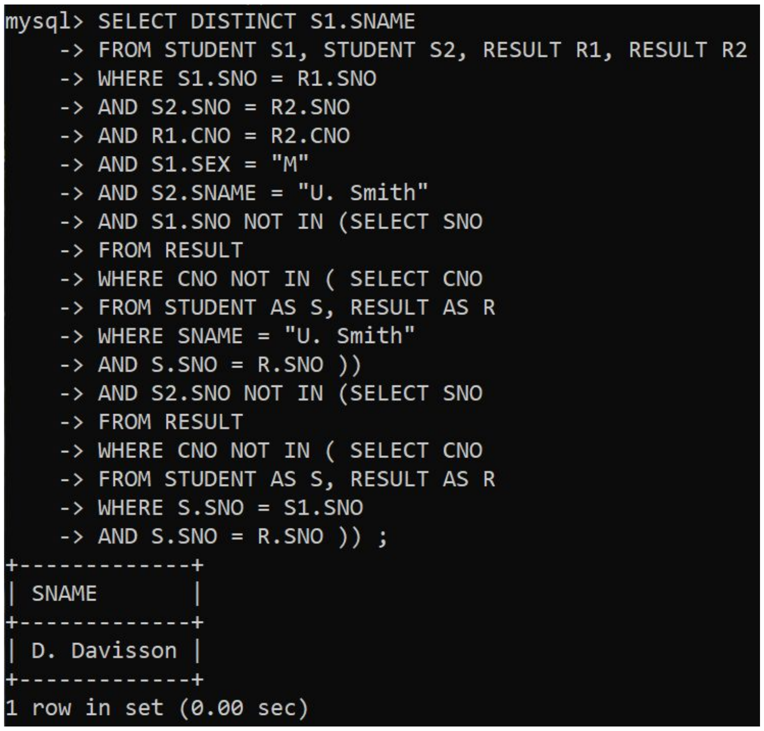
```
mysql> SELECT DISTINCT S.SNO, SNAME
    -> FROM STUDENT AS S, RESULT AS R
    -> WHERE SCORE IS NULL
    -> AND S.SNO = R.SNO
    -> OR S.SNO NOT IN
    -> (SELECT SNO FROM RESULT);
+-----+--------------+
| SNO | SNAME        |
+-----+--------------+
|   1 | N. J. Sloane |
|  22 | S. Conners   |
|  11 | N. Park      |
+-----+--------------+
3 rows in set (0.00 sec)
```

(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:

```
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 AS S, RESULT AS R
                WHERE SNAME = “U. Smith”
                AND S.SNO = R.SNO ))
AND S2.SNO NOT IN (SELECT SNO
        FROM RESULT
        WHERE CNO NOT IN ( SELECT CNO
                FROM STUDENT AS S, RESULT AS R
                WHERE S.SNO = S1.SNO
                AND S.SNO = R.SNO )) ;
```

Machine-generated solution:

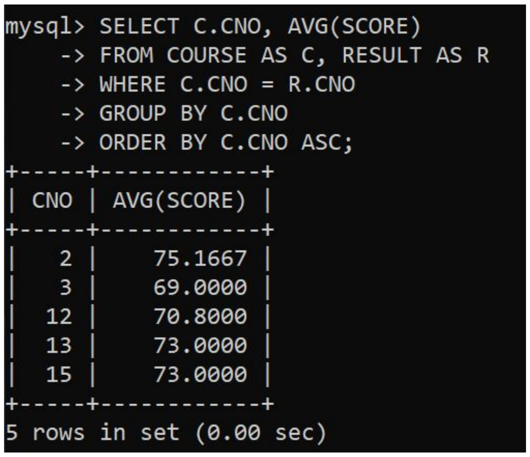


```
mysql> SELECT DISTINCT S1.SNAME
    -> FROM STUDENT S1, STUDENT S2, RESULT R1, RESULT R2
    -> WHERE S1.SNO = R1.SNO
    -> AND S2.SNO = R2.SNO
    -> AND R1.CNO = R2.CNO
    -> AND S1.SEX = "M"
    -> AND S2.SNAME = "U. Smith"
    -> AND S1.SNO NOT IN (SELECT SNO
    -> FROM RESULT
    -> WHERE CNO NOT IN ( SELECT CNO
    -> FROM STUDENT AS S, RESULT AS R
    -> WHERE SNAME = "U. Smith"
    -> AND S.SNO = R.SNO ))
    -> AND S2.SNO NOT IN (SELECT SNO
    -> FROM RESULT
    -> WHERE CNO NOT IN ( SELECT CNO
    -> FROM STUDENT AS S, RESULT AS R
    -> WHERE S.SNO = S1.SNO
    -> AND S.SNO = R.SNO )) ;
+-------------+
| SNAME       |
+-------------+
| D. Davisson |
+-------------+
1 row in set (0.00 sec)
```

CSE4701 - Databases

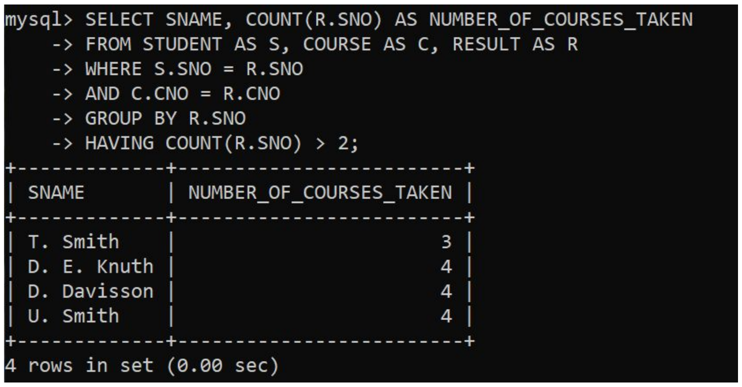
Project 1 Part 2 - 2020-02-25

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

SQL:	Machine-generated solution:
SELECT C.CNO, AVG(SCORE) FROM COURSE AS C, RESULT AS R WHERE C.CNO = R.CNO GROUP BY C.CNO ORDER BY C.CNO ASC;	

```
mysql> SELECT C.CNO, AVG(SCORE)
    -> FROM COURSE AS C, RESULT AS R
    -> WHERE C.CNO = R.CNO
    -> GROUP BY C.CNO
    -> ORDER BY C.CNO ASC;
+-----+------------+
| CNO | AVG(SCORE) |
+-----+------------+
|   2 |    75.1667 |
|   3 |    69.0000 |
|  12 |    70.8000 |
|  13 |    73.0000 |
|  15 |    73.0000 |
+-----+------------+
5 rows in set (0.00 sec)
```

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

SQL:	Machine-generated solution:
SELECT SNAME, COUNT(R.SNO) AS NUMBER_OF_COURSES_TAKEN FROM STUDENT AS S, COURSE AS C, RESULT AS R WHERE S.SNO = R.SNO AND C.CNO = R.CNO GROUP BY R.SNO HAVING COUNT(R.SNO) > 2;	

```
mysql> SELECT SNAME, COUNT(R.SNO) AS NUMBER_OF_COURSES_TAKEN
    -> FROM STUDENT AS S, COURSE AS C, RESULT AS R
    -> WHERE S.SNO = R.SNO
    -> AND C.CNO = R.CNO
    -> GROUP BY R.SNO
    -> HAVING COUNT(R.SNO) > 2;
+------------+-------------------------+
| SNAME      | NUMBER_OF_COURSES_TAKEN |
+------------+-------------------------+
| T. Smith   |                       3 |
| D. E. Knuth|                       4 |
| D. Davisson|                       4 |
| U. Smith   |                       4 |
+------------+-------------------------+
4 rows in set (0.00 sec)
```

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

(a) Write an SQL statement.

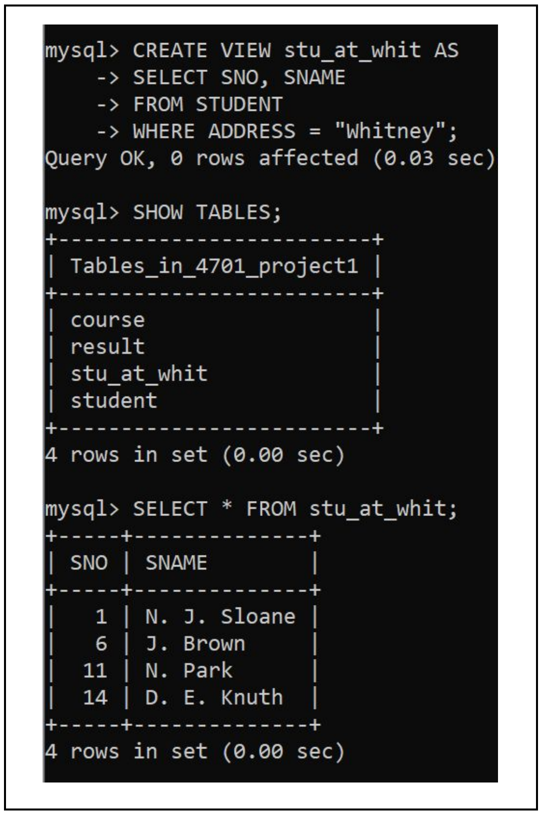
```
SELECT CNAME, SNAME
FROM STUDENT AS S, COURSE AS C, RESULT AS R
WHERE ADDRESS = “Whitney”
AND S.SNO = R.SNO
AND C.CNO = R.CNO;
```

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

```
CREATE VIEW stu_at_whit AS
SELECT SNO, SNAME
FROM STUDENT
WHERE ADDRESS = “Whitney”;

SELECT CNAME, SNAME
FROM stu_at_whit AS S, COURSE AS C, RESULT AS R
WHERE S.SNO = R.SNO
AND C.CNO = R.CNO;
```

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



```
mysql> CREATE VIEW stu_at_whit AS
    -> SELECT SNO, SNAME
    -> FROM STUDENT
    -> WHERE ADDRESS = "Whitney";
Query OK, 0 rows affected (0.03 sec)

mysql> SHOW TABLES;
+------------------------+
| Tables_in_4701_project1 |
+------------------------+
| course                 |
| result                 |
| stu_at_whit            |
| student                |
+------------------------+
4 rows in set (0.00 sec)

mysql> SELECT * FROM stu_at_whit;
+-----+-------------+
| SNO | SNAME       |
+-----+-------------+
|   1 | N. J. Sloane |
|   6 | J. Brown    |
|  11 | N. Park     |
|  14 | D. E. Knuth  |
+-----+-------------+
4 rows in set (0.00 sec)
```

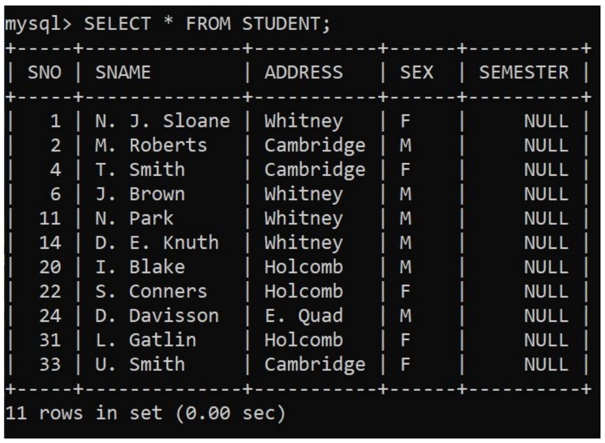
CSE4701 - Databases

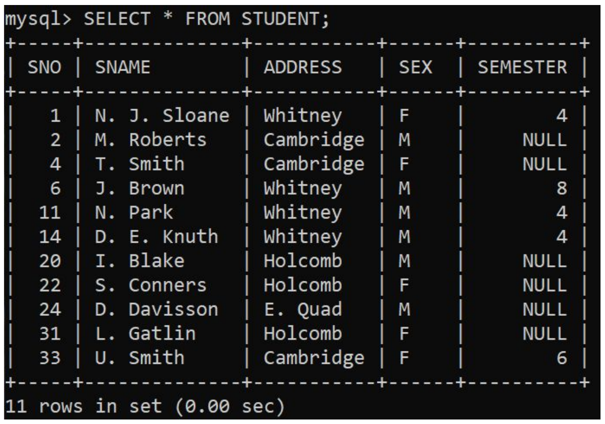
Project 1 Part 2 - 2020-02-25

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?

Assuming that the “SEMESTER” column represents a student’s semester as how long they have been in the school, and representable by an integer (semester 1 through 8, for example).

SQL:	MySQL Output

ALTER TABLE STUDENT
ADD SEMESTER int;
 	
SELECT * FROM STUDENT;
	

UPDATE STUDENT
SET SEMESTER = 4
WHERE ADDRESS = “Whitney”;
 	
UPDATE STUDENT
SET SEMESTER = 8
WHERE SNAME = “J. Brown”;
 	
UPDATE STUDENT
SET SEMESTER = 6
WHERE SNAME = “U. Smith”;
 	
SELECT * FROM STUDENT;
	

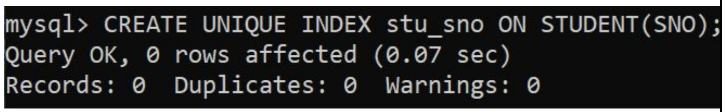
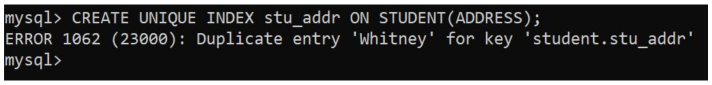
```
mysql> SELECT * FROM STUDENT;
+-----+-------------+-----------+------+----------+
| SNO | SNAME       | ADDRESS   | SEX  | SEMESTER |
+-----+-------------+-----------+------+----------+
|   1 | N. J. Sloane | Whitney   | F    |     NULL |
|   2 | M. Roberts  | Cambridge | M    |     NULL |
|   4 | T. Smith    | Cambridge | F    |     NULL |
|   6 | J. Brown    | Whitney   | M    |     NULL |
|  11 | N. Park     | Whitney   | M    |     NULL |
|  14 | D. E. Knuth | Whitney   | M    |     NULL |
|  20 | I. Blake    | Holcomb   | M    |     NULL |
|  22 | S. Conners  | Holcomb   | F    |     NULL |
|  24 | D. Davisson | E. Quad   | M    |     NULL |
|  31 | L. Gatlin   | Holcomb   | F    |     NULL |
|  33 | U. Smith    | Cambridge | F    |     NULL |
+-----+-------------+-----------+------+----------+
11 rows in set (0.00 sec)
```

```
mysql> SELECT * FROM STUDENT;
+-----+-------------+-----------+------+----------+
| SNO | SNAME       | ADDRESS   | SEX  | SEMESTER |
+-----+-------------+-----------+------+----------+
|   1 | N. J. Sloane | Whitney   | F    |        4 |
|   2 | M. Roberts  | Cambridge | M    |     NULL |
|   4 | T. Smith    | Cambridge | F    |     NULL |
|   6 | J. Brown    | Whitney   | M    |        8 |
|  11 | N. Park     | Whitney   | M    |        4 |
|  14 | D. E. Knuth | Whitney   | M    |        4 |
|  20 | I. Blake    | Holcomb   | M    |     NULL |
|  22 | S. Conners  | Holcomb   | F    |     NULL |
|  24 | D. Davisson | E. Quad   | M    |     NULL |
|  31 | L. Gatlin   | Holcomb   | F    |     NULL |
|  33 | U. Smith    | Cambridge | F    |        6 |
+-----+-------------+-----------+------+----------+
11 rows in set (0.00 sec)
```

It seems as if the system changes the table when modifying the schema and creating the new column “SEMESTER”. This is seen in the first table upon ALTERing the table.

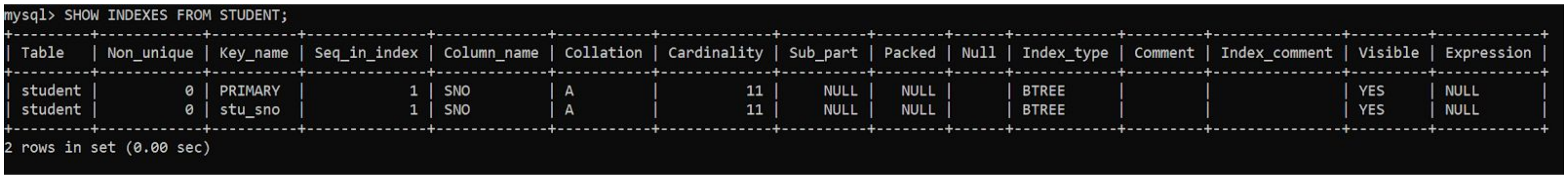
CSE4701 - Databases
Project 1 Part 2 - 2020-02-25

(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.

SQL:	Machine-generated solution:
CREATE UNIQUE INDEX stu_sno ON STUDENT(SNO);	
CREATE UNIQUE INDEX stu_addr ON STUDENT(ADDRESS);	
SHOW INDEXES FROM STUDENT;	see below

```
mysql> CREATE UNIQUE INDEX stu_sno ON STUDENT(SNO);
Query OK, 0 rows affected (0.07 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

```
mysql> CREATE UNIQUE INDEX stu_addr ON STUDENT(ADDRESS);
ERROR 1062 (23000): Duplicate entry 'Whitney' for key 'student.stu_addr'
mysql>
```

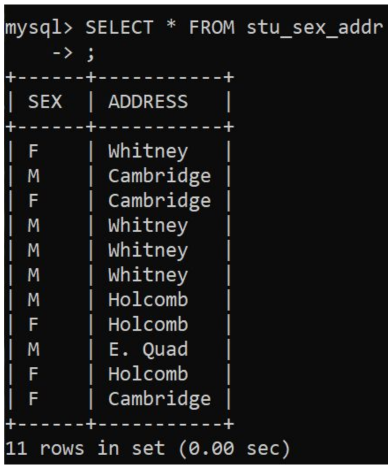


```
mysql> SHOW INDEXES FROM STUDENT;
+---------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+---------+------------+
| Table   | Non_unique | Key_name | Seq_in_index | Column_name | Collation | Cardinality | Sub_part | Packed | Null | Index_type | Comment | Index_comment | Visible | Expression |
+---------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+---------+------------+
| student |          0 | PRIMARY  |            1 | SNO         | A         |          11 |     NULL |   NULL |      | BTREE      |         |               | YES     | NULL       |
| student |          0 | stu_sno  |            1 | SNO         | A         |          11 |     NULL |   NULL |      | BTREE      |         |               | YES     | NULL       |
+---------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+---------+------------+
2 rows in set (0.00 sec)
```

Indexes can be created from existing keys or from attributes that are even non keys, by performing the show indexes query, you are able to see all the indexes that are created. However, if there are duplicate values in the attribute, you cannot create a unique index.

(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?

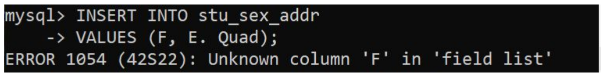
SQL:	Machine-generated solution:

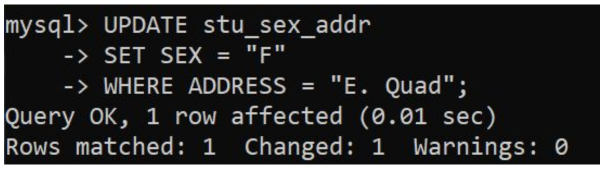
CREATE VIEW stu_sex_addr AS
SELECT SEX, ADDRESS
FROM STUDENT;
 	
SELECT * FROM stu_sex_addr;
	

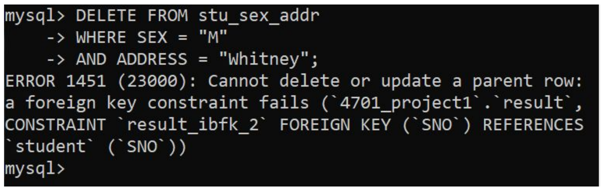
```
mysql> SELECT * FROM stu_sex_addr
    -> ;
+------+-----------+
| SEX  | ADDRESS   |
+------+-----------+
| F    | Whitney   |
| M    | Cambridge |
| F    | Cambridge |
| M    | Whitney   |
| M    | Whitney   |
| M    | Whitney   |
| M    | Holcomb   |
| F    | Holcomb   |
| M    | E. Quad   |
| F    | Holcomb   |
| F    | Cambridge |
+------+-----------+
11 rows in set (0.00 sec)
```

CSE4701 - Databases
Project 1 Part 2 - 2020-02-25

SQL:	Machine-generated solution:

INSERT INTO stu_sex_addr
VALUES (F, E. Quad);
	
	`mysql> INSERT INTO stu_sex_addr`
	`    -> VALUES (F, E. Quad);`
	`ERROR 1054 (42S22): Unknown column 'F' in 'field list'`

UPDATE stu_sex_addr
SET SEX = “F”
WHERE ADDRESS = “E. Quad”;
	
	`mysql> UPDATE stu_sex_addr`
	`    -> SET SEX = "F"`
	`    -> WHERE ADDRESS = "E. Quad";`
	`Query OK, 1 row affected (0.01 sec)`
	`Rows matched: 1  Changed: 1  Warnings: 0`

DELETE FROM stu_sex_addr
WHERE SEX = “M”
AND ADDRESS = “Whitney”;
	
	`mysql> DELETE FROM stu_sex_addr`
	`    -> WHERE SEX = "M"`
	`    -> AND ADDRESS = "Whitney";`
	``ERROR 1451 (23000): Cannot delete or update a parent row: a foreign key constraint fails (`4701_project1`.`result`, CONSTRAINT `result_ibfk_2` FOREIGN KEY (`SNO`) REFERENCES `student` (`SNO`))``
	`mysql>`

As seen from the above output per each output, insert and delete on the view do not work, views do not allow for those operations. The update operation however does work, in this case, given the unique address name.