Assignment 3
CDS-302 Section:2 – Fall 2021
Due: 11:59pm Wednesday September 29th, 2021
(Worth (estimated) 10% of your overall course assignment points)

Your Name: Isabel Angela Aldaba

Instructions

Turn in a MS Word or PDF document including both the questions and the answers. You can use a copy of this Microsoft Word document to paste in 1) your SQL Statements, and 2) the resulting output when you run the query in SQLite DB Browser under each question.

To receive partial credit you must include something for each SQL query. We will use a version of the university database from the lectures but the data values in the database are different from those in the lecture. Your answers must include 1) your SQL statement, and 2) the result from your query when you run it. Do not rearrange or edit the output of the query. Unless otherwise indicated, each question is to be answered with a **single SQL statement**.

To import the data you will first create a database in DB Browser and then use the file named **uni-v4.sql** to create the tables and import the data. Be sure to review this file and understand what it is doing. Understanding this file will also help with the bonus question.

1 More SQL [100 points]

Write SQL queries to answer the following questions against this database. For each question: provide your SQL code, and also paste in the results from the DB Browser. Special note: Sometimes a question posed in a natural language may be ambigious when mapped onto a programming language. So as hard as we may try to write them clearly, if you find a question to be ambigious, make an interpretation and describe it in a comment in your answer. Note that some questions also ask you to provide a written answer. Pleas just add your text under each question.

10 Questions

Example format for your answers:

Q0. (0 points) What are all the departments within the University?

SQL:

```
select dept_name, building, budget
from department
order by dept_name;
```

Output:

dept_name	building	budget
Biology	Exploratory Hall	85000
CDS	Research Hall	100000
Computer Science	Engineering Center	200000
Geography	Exploratory Hall	60000
History	Arts Center	50000
Math	Exploratory Hall	55000
Music	Arts Center	70000
Physics	Innovation Hall	90000

Questions

Q1. (10 points) Display all the students. Include all columns from the student table without specifying each column individually. Sort the output by department and total credits (within each department).

SQL:

```
select *
from student
group by dept_name, tot_cred
```

Output:

ID	name	dept_name	tot_cred
00011	Paul	*NULL*	*NULL*
00008	Mary	Biology	44
00003	John	Biology	56
00001	Bob	CDS	32
00006	Lucy	CDS	86
00007	Rick	Math	12
00002	Alice	Math	80
00010	Daisy	Music	12
00005	Bill	Music	110
00009	Jimmy	Physics	39
00004	Hellen	Physics	110

Q2. (10 points) Note that one student has a null value entry for department and total credits. Modify your a query in the prior problem (Q1) such that it displays only students with a null value for total credits. Display all student attributes, but replace any null values (dept_name, tot_cred) using the 'ifnull' function. For a null value in total credits indicate a value of 0, and for a null department indicate a value of "Undeclared". Sort by department name and ID as in the problem above.

SQL:
```
select ID, name, ifnull(dept_name, 'Undeclared') as dept_name,
ifnull(tot_cred, 0) as tot_credit
from student
where dept_name is null
```

Output:

ID	name	dept_name	tot_credit
00011	Paul	Undeclared	0

Q3. (10 points) Modify the prior query (Q2) by renaming the 4 columns of the result set to be simply c1, c2, c3, c4 rather than their given attribute names.

Hint: In your list of selected attributes alias the result by putting the new name after the attribute (e.g. ID) or function

(e.g. ifnull()):

select <col0> c1, <col1> c1, etc...

where <colN> are either the column names or some function.

SQL:
```
select id c1, name c2,  ifnull(dept_name, 'Undeclared') c3,
ifnull(tot_cred, 0) c4
from student
where dept_name is null
```

Output:

c1	c2	c3	c4
00011	Paul	Undeclared	0

Q4. (10 points) Display the lowest, highest and average number of credits for students by department include the department in the results of the query and order the results alphabetically by department. Hint (you need a **group by** clause)

Special note: One student, a new student, has a null values for tot_cred - filter that student from the result.

SQL:
```
select dept_name, min(tot_cred), max(tot_cred),  avg(tot_cred)
from student
where tot_cred is not null
group by dept_name
```

Output:

dept_name	min(tot_cred)	max(tot_cred)	avg(tot_cred)
Biology	44	56	50.0
CDS	32	86	59.0
Math	12	80	46.0
Music	12	110	61.0
Physics	39	110	74.5

Q5. (10 points) Modify the query completed in Q4, however this time filter out any department whose student's average total credits are less than 60 hours. Again, display the lowest, highest and average number of credits for the students by department include the

department in the results of the query and order the results alphabetically by department. Again this should be done in one SQL query statement. Hint (you need a **having** clause)

SQL:

```
select dept_name, min(tot_cred), max(tot_cred),  avg(tot_cred)
from student
where tot_cred is not null
group by dept_name
having avg(tot_cred) > 60
```

Output:

dept_name	min(tot_cred)	max(tot_cred)	avg(tot_cred)
Music	12	110	61.0
Physics	39	110	74.5

Q6. (10 points) Review the schema for the University Database. In particular, review the primary keys and foreign keys for student, advisor and instructor. Based on your understanding for the schema, answer these questions:

a) Does this model allow students to have zero, one or many advisors?

The model allows students to have one advisor

b) Does this model allow advisors to advise zero, one or many students?

The model allows advisors to advise many students

Explain your answers. You can review the detailed constraints in the uni-v4.sql file and/or the graphical schema model for the University database.

Answer:

When looking at the table "advisor", the column s_ID (representing the students' IDs) are all unique; however, IDs 333333 and 54321 occur twice in the i_ID column. This means that students can only have one advisor, but advisors can be assigned to multiple students, as the instructors with the IDs 333333 and 54321 each advise 2 students.

Q7. (10 points) This question asks you to first form the cartesian product between two tables and then refine the result by joining the two tables together. The goal is to get you to think about what is

happening at each step, so I would like you to explain in your own words what is happening.

a) write a query that results in the cartesian product, showing all attributes for both tables between the tables student and advisor. Write the SQL statement, run it, and record the number of rows returned (**you do not have to record the actual output as it should be quite long.**

SQL:

```
select *
from student, advisor
```

Output:

ID	name	dept_name	tot_cred	s_ID	i_ID
00001	Bob	CDS	32	00001	11112
00001	Bob	CDS	32	00002	44444
00001	Bob	CDS	32	00003	54321
00001	Bob	CDS	32	00004	33333
00001	Bob	CDS	32	00005	12121
00001	Bob	CDS	32	00006	11111
00001	Bob	CDS	32	00007	33333
00001	Bob	CDS	32	00008	54321
00001	Bob	CDS	32	00009	33333
00002	Alice	Math	80	00001	11112
00002	Alice	Math	80	00002	44444
00002	Alice	Math	80	00003	54321
00002	Alice	Math	80	00004	33333
00002	Alice	Math	80	00005	12121

ecution finished without errors.
sult: 99 rows returned in 12ms

b) Now, explain why so many rows are returned. What is happening?

The code returned so many rows because I did not specify a relationship between the two tables when making my query. Thus, each row in the table "student" was paired with each row in "advisor". Because there are 11 advisors and 9 students, the query resulted in a table with 11 x 9 rows (99 in total).

c) Now, modify this query such that each student is correctly paired with the id (i_id) of the students advisor. Explain the modification you made and why it results in so many fewer tuples being returned. Hint: You will need to use a join between the tables. See discussion on joins from week 3.(Next week we will look at joins in more detail). Show all attributes from each table. Order the result by student id.

SQL:

```
select *
from student join advisor on student.ID = advisor.s_ID
order by student.ID
```

Output:

ID	name	dept_name	tot_cred	s_ID	i_ID
00001	Bob	CDS	32	00001	11112
00002	Alice	Math	80	00002	44444
00003	John	Biology	56	00003	54321
00004	Hellen	Physics	110	00004	33333
00005	Bill	Music	110	00005	12121
00006	Lucy	CDS	86	00006	11111
00007	Rick	Math	12	00007	33333
00008	Mary	Biology	44	00008	54321
00009	Jimmy	Physics	39	00009	33333

Q8. (10 points) This query builds on the query from the prior problem. There is no reason a query has to be limited to joining only two tables and the hint for this query is that you will need 3 tables (student, advisor, and instructor). In the from clause, multiple tables can be joined in succession...there is actually multiple ways of doing this, but for now use this syntax in the from clause:

from t0 join t1 on <condition0> join t2 on <condition1>

Use this idea to write a query that lists the id and name of each student along with the id and name of the student's advisor. You may find it convienient to rename(i.e. alias)the name of each table in this query. Your results will make more sense if you label each column with a meaningful name. If you are unable to solve this problem, be sure to include your best attempt so that you recieve partial credit.

SQL:

```
select *
from student join advisor on student.ID = advisor.s_ID  join
instructor on advisor.i_ID = instructor.ID
```

Output:

ID	name	dept_name	tot_cred	s_ID	i_ID	ID	name	dept_name	salary
00001	Bob	CDS	32	00001	11112	11112	Widom	CDS	100000
00002	Alice	Math	80	00002	44444	44444	Euler	Math	77000
00003	John	Biology	56	00003	54321	54321	Crick	Biology	88000
00004	Hellen	Physics	110	00004	33333	33333	Newton	Physics	80000
00005	Bill	Music	110	00005	12121	12121	Mozart	Music	75000
00006	Lucy	CDS	86	00006	11111	11111	Turing	Computer Science	95000
00007	Rick	Math	12	00007	33333	33333	Newton	Physics	80000
00008	Mary	Biology	44	00008	54321	54321	Crick	Biology	88000
00009	Jimmy	Physics	39	00009	33333	33333	Newton	Physics	80000

Q9 (10 points) Display the a list of students (all attributes) that are either Math or Physics students using two different methods. In the first method use a predicate in the where clause to select these students. In the second method use a set operation to accomplish the same result.

SQL:

```
select *
from student
where dept_name = "Physics" or dept_name = "Math"
```

Output:

ID	name	dept_name	tot_cred
00002	Alice	Math	80
00004	Hellen	Physics	110
00007	Rick	Math	12
00009	Jimmy	Physics	39

SQL:

```
select *
from student
where dept_name = "Physics"
UNION
select *
from student
where dept_name = "Math"
```

Output:

ID	name	dept_name	tot_cred
00002	Alice	Math	80
00004	Hellen	Physics	110
00007	Rick	Math	12
00009	Jimmy	Physics	39

Q10. (10 points) 4 parts in total.

Part a) List all attributes for students whose name starts with an 'J'.

SQL:

```
select *
from student
where name like 'J%'
```

Output:

ID	name	dept_name	tot_cred
00003	John	Biology	56
00009	Jimmy	Physics	39

Part b) List all attributes for all instructors whose name ends with 's'.

SQL:
```
select *
from instructor
where name like '%s'
```

Output:

ID	name	dept_name	salary
22222	Heraclitus	History	65000
12345	Eratosthenes	Geography	55000

Part c) List all attributes for all students whose name is exactly 4 characters long and ends with 'y'.

SQL:
```
select *
from student
where name like '%y' and length(name) = 4
```

Output:

ID	name	dept_name	tot_cred
00006	Lucy	CDS	86
00008	Mary	Biology	44

Part d) Think about a situation where you might need to use the string functions lower, upper and trim. Briefly describe such a situation in your own words.

I may need to use the string functions upper, lower, and trim while cleaning data. Data that comes from a survey may come in different formats – For example, some people may input their names in all lowercase letters while others capitalize. To make the data entries appear uniform, it would be useful to use upper,lower, and trim.