Assignment 4
CDS-302 Section:2 – Fall 2021 Joe Boone
Due: 11:59pm, Tuesday, October 5th, 2021
(Worth (estimated) 10% of your overall course assignment points)

Your Name: Isabel Aldaba

Instructions

In this homework you will turn in a Word document covering Part 1 (More SQL (Joins/Subqueries) and a PDF covering Part 2 (Relational Algebra). The Word document you will turn in for section 1 is similar format to the Word documents used in assignment 2 and 3 where you paste your query and output after each question.

1 More SQL (Joins/Subqueries) [80 points]

For this section, to receive partial credit you must include something for each SQL query. Special note: Sometimes a question posed in a natural language may be ambiguous when mapped onto a programming language. So as hard as we may try to write them clearly, if you find a question to be ambiguous, make an interpretation and describe it in a comment in your answer. Note that some questions also ask you to provide a written answer. Please just add your text under each question.

1a) Using the University Database

Use the university database to answer these queries. As before you can use the include uni-v4.sql file to build the database or if you have save the database from prior weeks you can use that.

Example format for your answers for part 1a) and 1b):

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. List the student names, the course ids that they have taken, and the grades they received in each course. The names of students who have not taken any courses yet should also be listed in the same query with null values for the course and grade. Hint: Consider the available join types (and associated syntax) and select the most appropriate given the query.

Note: when the result set in DB Browser contains a NULL value, when you paste this into the Word document – these

will appear as empty – this is fine for this assignment. Alternatively, you may substitute a value for these null values using the 'ifnull' function as we have seen in lectures and prior assignments.

SQL:

```
select name, course_id, grade
from student natural left outer join takes
```

Output:

name	course_id	grade
Bob	CDS-101	A
Bob	CDS-130	B+
Bob	CDS-302	A+
Alice	CDS-302	A+
Alice	MAT-114	B
John	BIO-101	C
Hellen	PHY-403	B-
Bill	MUS-299	D
Lucy	CDS-101	A
Lucy	CDS-130	B-
Rick	BIO-101	C
Rick	CDS-302	A+
Rick	MAT-114	B
Mary	CDS-302	A
Jimmy	CDS-302	A
Daisy	*NULL*	*NULL*
Paul	*NULL*	*NULL*

Q2. List all departments along with their average instructor salaries, for those departments that have an average salary

above the average salary of all instructors. Hint: You can use either a subquery or a with clause to answer this query.

SQL:

```
select dept_name, avg_salary
from ( select dept_name, avg (salary) as avg_salary
from instructor
group by dept_name)
where avg_salary >
(select avg(salary)
from instructor);
```

Output:

dept_name	avg_salary
Biology	88000.0
CDS	100000.0
Computer Science	95000.0
Music	97500.0

Q3. List every unique building and room_number that Bob (a student) has ever had a class in. Make sure each unique building and room_number only appears one time in the list and that the list is sorted. Consider the join types used – determine if you can use a Natural Join or not.

SQL:

Output:

Q4. List the id, name, and department name of **every student that has an advisor that has more than one student they advise.** Hint: A subquery can be used to first find the instructors that have more than one student that they advise. Try writing this subquery first. Then use this subquery, to answer the top level question.

SQL:

```
select id, name, dept_name
from advisor join student on advisor.s_id = student.ID
```

```
where i_ID = (select i_id
from advisor
group by i_id
having count(i_id) > 1)
```

Output:

ID	name	dept_name
00004	Hellen	Physics
00007	Rick	Math
00009	Jimmy	Physics

Q5. Explain why the following statement returns the value 64.

```
select count(*)
from department d1, department d2;
```

Answer:
There are 8 rows associated with department d1 and 8 rows associated with department d2. The query multiplies both numbers of rows to return 64.

1b) Using the Movies Database

Use the movies.db SQLite database distributed with the homework. You can directly attach to this database using DB Browser for SQLite. This section will have fewer hints, work through each query and be sure to include your best attempt to get partial credit.

The schema of the movies database is the following:

```
actors(aid, firstname, lastname);
country(cid, country);
langs(lid, language);
movies(mid, title, year, grade);
incountry(cndx, mid, cid);
inlang(lndx, mid, lid);
isin(id, mid, aid);
```

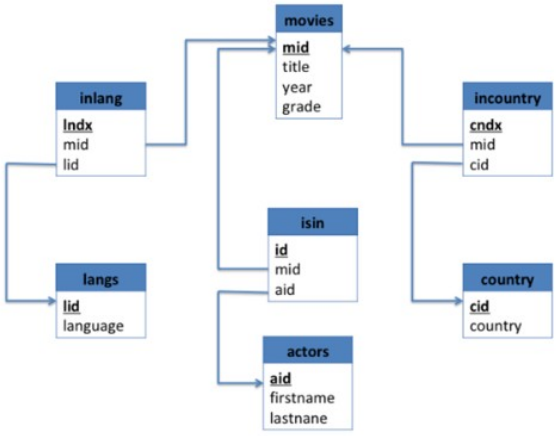

Figure 1: Movies database schema.

Q6. The movies database is incomplete in the following sense: Each movie may not have information for each of the categories (where it was filmed, what actors are in it, what languages it is available in). Think about the keys that make up each table in the database. Write a small set of short queries that you could use to verify what I've just told you is true. You do not have to record the output of these queries, just include the queries themselves.

SQL:

```
select *
from movies natural left outer join isin
```

```
select *
from movies natural left outer join incountry

select *
from movies natural left outer join  inlan
```

Q7. What is the average grade for movies produced in the year 2000 or later, per language - (You need to show each language and the corresponding average grade). Display this list, sorted, with the highest average grades first.

SQL:

```
select avg(grade)
from movies join inlang on movies.mid = inlang.mid
join langs on inlang.lid = langs.lid
group by language
having year >= 2000
order by avg(grade) desc
```

Output:

language	avg(grade)
Greek	10.0
Bulgarian	10.0
Czechoslovakian	9.33333333333333
Ukranian	9.0
Finnish	9.0
Dutch	8.5
Italian	8.25
German	8.23076923076923
Yiddish	8.0
Tibetan	8.0
Nepali	8.0
Mandarin	8.0
Korean	8.0
Kazakh	8.0

Irani	8.0
Inuktitut	8.0
Hindi	8.0
Hebrew	8.0
Estonian	8.0
Russian	7.75
Arabic	7.66666666666667
Norwegian	7.5
Xhosa	7.0
Japanese	7.0
French	6.89285714285714
Mongolian	6.5
Hungarian	6.0
Australian	6.0
Danish	5.25
Sindarin	4.5
Navajo	4.0
Portugese	2.0

Q8. For this query, consider only movies released in a particular year – let's say 1963. List the movie titles that were filmed in the United States (country=`USA') or are in English (language = `English'). Display the list of titles in ascending order.

SQL:

```
select title
from movies join inlang on movies.mid = inlang.mid
join langs on inlang.lid = langs.lid
where year = '1963' and language = 'English'
```

Output:

Title
The Great Escape
It's a Mad, Mad, Mad, Mad World

The Birds
The Pink Panther

Q9. Display the number of movies for each country for the Eighties (1980 thru 1989 inclusive). For movies that do not have country information provided (i.e. the movie mid does not appear in the table incountry, count those counties as 'Unknown' . (The count for 'Unknown' is much greater than the counts for known countries).

SQL:

```
select count(*) as 'known'
from movies
where year like '198%'
```

Output:

Known
128

Q10. Determine how many movies have an actor in them with a last name beginning with the letters 'Ca' and display a count of those movies by the country they were filmed in. Your output should be a sorted list of countries followed by the count that meet the criteria specified. For movies that do not have country information indicate them as 'Unknown' and give the count for them.

SQL:

```
select country, count(*)
from movies join isin on movies.mid = isin.mid
join actors on isin.aid = actors.aid
join incountry on movies.mid = incountry.mid
join country on incountry.cid = country.cid
where lastname like 'Ca%'
group by country
```

Output:

country	count(*)
UK	3
USA	4

2 Relational Algebra [20 points]

Convert each of the following SQL statements into a relational algebra expression. Use LaTeX to encode the expressions. Each question is worth 5 points. For this section – turn in a PDF covering the 4 questions below.

Q11. Convert the following SQL expression to an equivalent relational algebra expression and use natural language to describe what the statement represents.

SQL:

```
select id, name, tot_cred
from student
where tot_cred > 100 and dept_name = “CDS”;
```

Q12. Convert the following SQL expresion to an equivalent relational algebra expression and use natural language to describe what the statement represents.

SQL:

```
select dept_name, avg(salary) avg_sal
from instructor
group by dept_name
having avg_sal >= 50000
```

Q13. Convert the following SQL expresion to an equivalent relational algebra expression and use natural language to describe what the statement represents.

SQL:

```
select distinct course_id, title, dept_name
from course natural join department
where dept_name = 'Biology' and
```

```
building in ('Research Hall','Exploratory Hall');
```

Q14. Convert your answer for Q3 (above) to an equivalent relational algebra expression. You do not need to describe this in natural language since it should match the description in Q3.