CSE 414 Midterm

Friday, May 3, 2019, 1:30-2:20

Name (please print): _____

Student number: _____

Question	Points	Score
1	40	
2	15	
3	30	
4	5	
5	10	
Total:	100	

- This exam is CLOSED book and CLOSED devices.
- You are allowed ONE letter-size page with notes (both sides).
- You have 50 minutes;
- Answer the easy questions before you spend too much time on the more difficult ones.
- If there is a line to write your answer (e.g True/False, a/b/c/d, etc), you must write your answer. Circling the option is not enough.
- Good luck!

1 SQL

1. (40 points)

   The city of Nullville maintains a database of all cars in the city, and records their license plates whenever they drive downtown:

   ```
   Car(lp, make, owner)
   Downtown(lp, day)
   ```

   - The relation `Car` stores information about all the cars registered in the city. All its attributes are of type `text`; `lp` it is the primary key in `Car`, `make` represents the make of that car (Honda, Ford, etc) and `owner` is the name of the registered owner. Notice that an owner may have multiple cars.
   - The relation `Downtown` represents days when a particular car drove downtown. The attribute `lp` is a foreign key to `Car`, and `day` is an integer that represents the day number, counting from the moment when the database became operational (assume this was about ten years ago). There is at most one record for each car and day: if a car drives downtown multiple times during one day, then we only record that once.

   (a) (5 points) Write the sequence of SQL statements necessary to create the tables above. Include all keys or foreign keys declarations.

   Solution:

   ```
   DROP TABLE IF EXISTS Downtown;
   DROP TABLE IF EXISTS Car;

   CREATE TABLE Car(lp text primary key, make text, owner text);
   CREATE TABLE Downtown(lp text references Car, day int);

   -- for instructor's testing purposes only
   insert into car values ('XY52Z','Honda','Alice');
   insert into car values ('ZY23X','Bentley','Bob');
   insert into car values ('MWM92','Bentley','Alice');
   insert into downtown values ('MWM92', '2000');
   insert into downtown values ('MWM92', '3000');
   insert into downtown values ('ZY23X', '4000');
   insert into downtown values ('MWM92', '5000');
   ```

```
Car(lp, make, owner)
Downtown(lp, day)
```

(b) (5 points) Write a SQL query that computes, for each owner, how many cars they own. Your query should return a a set of `owner`, `count` pairs, sorted in decreasing order of the `count`.

Solution:

```
select x.owner, count(*) as Count
from Car x
group by x.owner
order by count(*) desc;
```

```
Car(lp, make, owner)
Downtown(lp, day)
```

(c) (5 points) Write a SQL query that returns, for each car `make` the total number of cars of that type that entered downtown on or after day 1000. Your query should count every day when a car passed through downtown, in other words, if the same car passes through downtown on days 2200, 2250, and 3100, then you count that as three. Your answer should consists of a set of `make`, `count` pairs, sorted in decreasing order of the `count`, like this:

Make	Count
Honda	4201
Fiat	3477
Ford	2010
...	
Bentley	0

Your query should include the makes of the cars that were never driven downtown, for example Bentley above.

Solution:

```
select x.make, count(y.lp) as Count
from Car x left outer join Downtown y
  on x.lp = y.lp and y.day >= 1000
group by x.make
order by count(*) desc;
```

```
Car(lp, make, owner)
Downtown(lp, day)
```

(d) (10 points) Find all owners who have not driven to downtown on or after day 1000. Notice that some owners may own multiple cars; you need to return them only if none of their cars drove downtown on or after day 1000. Your query should return a set of owners; include each owner only once.

Solution:

```
select distinct u.owner
from car u
where not exists (select *
                  from car x, downtown y
                  where x.lp = y.lp
                    and y.day >= 1000
                    and x.owner = u.owner);
```

```
Car(lp, make, owner)
Downtown(lp, day)
```

(e) Consider the following query:

```
--  Q:
select distinct x.owner
from car x, downtown y
where x.make = 'Honda'
  and x.lp = y.lp
  and y.day >= 1000;
```

For each query below, indicate whether return the same answer or not. You only need to answer Y or N.

i. (2 points) Is this query equivalent to Q?

```
--  Q1:
select distinct x.owner
from car x, downtown y, downtown v
where x.make = 'Honda'
  and x.lp = y.lp
  and y.day >= 5000
  and x.lp = v.lp
  and v.day >= 1000;
```

i. **N**

Yes or No?

ii. (2 points) Is this query equivalent to Q?

```
--  Q2:
select distinct x.owner
from car x, downtown y, downtown v
where x.make = 'Honda'
  and x.lp = y.lp
  and y.day >= 200
  and x.lp = v.lp
  and v.day >= 1000;
```

ii. **Y**

Yes or No?

iii. (2 points) Is this query equivalent to Q?

```
-- Q3:
select distinct x.owner
from car x, downtown y, car u, downtown v
where x.make = 'Honda'
  and x.lp = y.lp
  and y.day >= 1000
  and x.owner = u.owner
  and u.lp = v.lp;
```

iii. **Y**

Yes or No?

iv. (2 points) Is this query equivalent to Q?

```
-- Q4:
select distinct x.owner
from car x, downtown y, car u, downtown v
where x.make = 'Honda'
  and x.lp = y.lp
  and x.owner = u.owner
  and u.lp = v.lp
  and v.day >= 1000;
```

iv. **N**

Yes or No?

v. (2 points) Is this query equivalent to Q?

```
-- Q5:
select distinct x.owner
from car x, downtown y, car u, downtown v
where x.lp = y.lp
  and y.day >= 1000
  and x.owner = u.owner
  and u.make = 'Honda'
  and u.lp = v.lp
  and v.day = y.day;
```

v. **y**

Yes or No?

(f) Consider the following table:

R :

A	B
3	5
3	NULL
NULL	5

Answer the following questions:

i. (1 point) What does the following query return?

```
select * from R where (A=3) and not(B=3);
```

Solution:

A	B
3	5

ii. (1 point) What does the following query return?

```
select * from R where (A=3) or not(B=3);
```

Solution:

A	B
3	5
3	NULL
NULL	5

iii. (1 point) What does the following query return?

```
select * from R where (A=3) or (A!=3);
```

Solution:

A	B
3	5
3	NULL

iv. (1 point) What does the following query return?

```
select * from R where A+B != 8;
```

Solution: empty

v. (1 point) What does the following query return?

```
select * from R where A is NULL or B is NULL;
```

Solution:

A	B
NULL	5
3	NULL

2 Relational Algebra

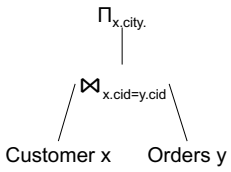
2. (15 points)

Consider the same relational schema as before:

```
Car(lp, make, owner)
Downtown(lp, day)
```

(a) (5 points) Write a Relational Algebra expression in the form of a logical query plan (i.e., draw a tree) that is equivalent to the SQL query below. Your query plan does not have to be necessarily "optimal": however, points will be taken off for overly complex solutions.

Hint: to avoid renaming, use aliases in the query plan, like this

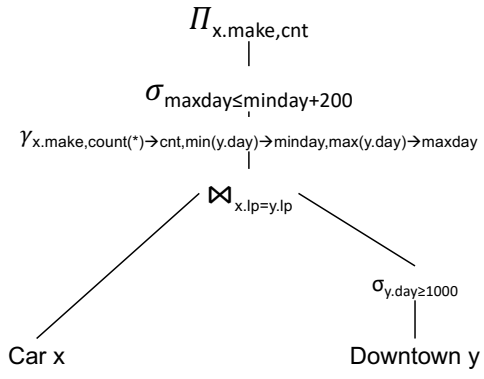



```
select x.make, count(*) as cnt
from Car x, Downtown y
where x.lp = y.lp
  and y.day >= 1000
group by x.make
having max(y.day) <= min(y.day) + 200;
```

Write the Relational Query expression below:

Solution:

$\Pi_{\texttt{x.make,cnt}}\,(\sigma_{\texttt{maxday}\leq\texttt{minday}+200}\,(\gamma_{\texttt{x.make,count(*)}\rightarrow\texttt{cnt,min(y.day)}\rightarrow\texttt{minday,max(y.day)}\rightarrow\texttt{maxday}}\,(\texttt{Car x}\bowtie_{\texttt{x.lp=y.lp}}\sigma_{\texttt{y.day}\geq 1000}(\texttt{Downtown y}))))$



```
Car(lp, make, owner)
Downtown(lp, day)
```

(b) i. (2 points) Which of the following is the most accurate English interpretation of the SQL query below?

```
select distinct u.owner
from Car u
where not exists
       (select *
        from Car x, Downtown y
        where u.owner = x.owner
          and x.make = 'Honda'
          and x.lp = y.lp
          and y.day >= 1000);
```

Returns all owners that ...

(A) Don't own a Honda and never drove downtown after day 1000.

(B) Own a Honda but never drove downtown after day 1000.

(C) Never drove downtown with a Honda after day 1000.

(D) Never drove downtown with a Honda before day 1000.

(E) All their trips downtown before day 1000 were in a Honda.

(F) All their trips downtown after day 1000 were in a Honda.

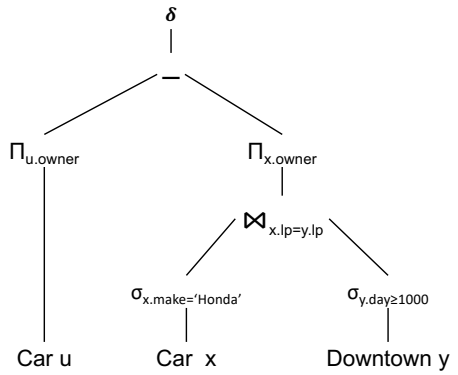
i. **C**

A/B/C/D/E/F:

ii. (8 points) Write a Relational Algebra expression in the form of a logical query plan (i.e., draw a tree) that is equivalent to the SQL query above. Your query plan does not have to be necessarily "optimal": however, points will be taken off for overly complex solutions.

Solution:

$\delta(\Pi_{\texttt{u.owner}}(\texttt{Car u}) - \Pi_{\texttt{x.owner}}(\sigma_{\texttt{x.make='Honda'}}(\texttt{Car x}) \bowtie_{\texttt{x.lp=y.lp}} \sigma_{\texttt{y.day}\geq 1000}(\texttt{Downtown y})))$



3 Datalog

3. (30 points)

Consider the same schema as before:

```
Car(lp, make, owner)
Downtown(lp, day)
```

Answer the questions below.

(a) (5 points) Write a datalog program that returns all owners who own both a Honda and a Ford.

Solution:

```
Q(x) :- Car(_, 'Honda',x), Car(_, 'Ford',x)
```

```
Car(lp, make, owner)
Downtown(lp, day)
```

(b) (5 points) Write a datalog program that returns all owners who own a Honda and do not own a Ford.

Solution:

```
HasFord(x) :- Car(_, 'Ford', x)
Q(x)       :- Car(_, 'Honda', x), !HasFord(x)
```

```
Car(lp, make, owner)
Downtown(lp, day)
```

(c) (10 points) A spy ring has infiltrated Nullville, and you are a counter intelligence officer charged with catching them. After lots of hard work you have found one suspected spy: Alice. To find more suspects, you make the following judgment. You know that some days the spies have secret meetings downtown, and when they meet downtown then they always drive Bentleys. You reason that, whenever you have found a suspect, if she/he drives a Bentley downtown one day, then every other person who drives a Bentley downtown the same day automatically becomes a suspect too. Write a datalog program to compute all suspects. (Hint: your first rule should be this single fact: `Suspect('Alice').`)

Solution:

```
Suspect('Alice').
Suspect(Y) :- Suspect(X),
              Car(lpx,'Bentley',X), Car(lpy,'Bentley',Y),
              Downtown(lpx,d), Downtown(lpy,d)
```

(d) Answer the questions below:

i. (2 points) Is this datalog rule safe?

```
Q(u) :- Car(x,'Honda',u), !Downtown(x,2000)
```

i. **Safe**

Safe or Unsafe?

ii. (2 points) Is this datalog rule safe?

```
Q(u) :- Car(x,'Honda',u), !Downtown(x,y)
```

ii. **Unsafe**

Safe or Unsafe?

iii. (2 points) Is this datalog rule safe?

```
Q(u) :- Car(x,'Honda',u), Downtown(x,d), !Car(y,'Ford',u), Downtown(y,d)
```

iii. **Safe**

Safe or Unsafe?

iv. (2 points) Is this datalog rule safe?

```
Q(u) :- Car(x,'Honda',u), Downtown(x,d), Car(y,'Ford',u), !Downtown(y,d)
```

iv. **Safe**

Safe or Unsafe?

v. (2 points) Is this datalog rule safe?

```
Q(u) :- Car(x,'Honda',u), !Downtown(x,d), Car(y,'Ford',u), Downtown(y,d)
```

v. **Safe**

Safe or Unsafe?

4 JSON and SQL++

4. (5 points)

   (a) (5 points) Consider the relational database instance below:

Car:

lp	make	owner
XY52Z	Honda	Alice
ZY23X	Bentley	Bob
MWM92	Bentley	Alice

Downtown:

lp	day
MWM92	2000
MWM92	3000
ZY23X	4000
MWM92	5000

Write a Json file that represents the same data.

(This page is intentionally left blank)

Solution:

```
{"Database":
   [{"lp": "XY52Z",
     "make": "Honda",
     "owner": "Alice",
     "downtown":[]},
    {"lp": "ZY23X",
     "make": "Bentley",
     "owner": "Bob",
     "downtown":[4000]},
    {"lp": "MWM92",
     "make": "Bentley",
     "owner": "Alice",
     "downtown": [2000,3000,5000]}]}
```

We also accepted solutions grouped by owner:

```
{"Database":
   [{"owner": "Alice",
     "cars": [ {"lp": "XY52Z",
                "make": "Honda",
                "downtown":[]},
               {"lp": "MWM92",
                "make": "Bentley",
                "downtown": [2000,3000,5000]}]},
    {"owner": "Bob",
     "cars": [ {"lp": "ZY23X",
                "make": "Bentley",
                "downtown":[4000]} ] } ] }
```

5 Miscellaneous

5. (10 points)

For each statement below, indicate whether it is true or false:

(a) (1 point) *Physical data independence* means that the data is compressed.

(a) **False**

True/False:

(b) (1 point) *First Normal Form* means that an attribute of a relation cannot be another relation.

(b) **True**

True or false?

(c) (1 point) A relational database is always in First Normal Form.

(c) **True**

True or false?

(d) (1 point) JSON data is always in First Normal Form.

(d) **False**

True or false?

(e) (1 point) If an attribute is a foreign key, then no two tuples may have the same value of that attribute.

(e) **False**

True or false?

(f) (1 point) It is possible for a relation to have three different primary keys, for example each of A, B, and C is a primary key in $R(A, B, C, D)$.

(f) **False**

True or false?

(g) (1 point) It is possible for a relation to have three different foreign keys, for example each of A, B, and C in $R(A, B, C, D)$ is a foreign key.

(g) **True**

True or false?

(h) (1 point) Alice writes a SQL query that ends in `...group by A`.
but her query returns 1000 answers, too many to see on the screen. She makes a single change, replace the last line by `...group by A,B`.
She hopes to get fewer than 1000 answers. Will the new query return at least 1000 answer, or at most 1000 answers?

(h) ≥ 1000

Answer "≥ 1000" or "≤ 1000" or "unknown":

(i) (1 point) Suppose the attribute K is a key in R, and suppose the attribute FK in S is foreign key to R. Then the size of the join $R \bowtie_{R.K=S.FK} S$ is always less than or equal to the size of R.

(i) **False**

True or false?

(j) (1 point) Suppose the attribute K is a key in R, and suppose the attribute FK in S is foreign key to R. Then the size of the join $R \bowtie_{R.K=S.FK} S$ is always less than or equal to the size of S.

(j) **True**

True or false?