EECS-3421A: TEST #2

"Queries"

Electrical Engineering & Computer Science
Lassonde School of Engineering
York University

Family Name: ______________________________
Given Name: ______________________________
Student#: ______________________________
EECS Account: ______________________________

Instructor: Parke Godfrey
Exam Duration: 75 minutes
Term: Fall 2016
Instructions

- **rules**
  - The test is closed-note, closed-book. Use of a calculator is permitted.
- **answers**
  - Should you feel a question needs an assumption to be able to answer it, write the assumptions you need along with your answer.
  - If you need more room to write an answer, indicate where you are continuing the answer.
  - For multiple choice questions, choose *one* best answer for each of the following. There is no negative penalty for a wrong answer.
- **notation**
  - For schema, the underlined attributes indicate a table's primary key (and are, hence, not nullable). Attributes in *italics* are not nullable. Foreign keys are indicated by FK.
  - Assume *set* semantics for relational-algebra expressions.
- **points**
  - The number of points a given question is worth is marked.
  - There are five major parts worth 10 points each, for 50 points in total.

MARKING BOX	
1.	/10
2.	/10
3.	/10
4.	/10
5.	/10
Total	/50

1. [10pt] **Relational Algebra.** *Quadratic queries!* [SHORT ANSWER]

   For Questions 1a to 1c, use the Colours schema in Figure 1 on page 15 (as used in examples in class) to write *relational-algebra* queries for the English questions posed.

---

a. [2pt] Show customers by cust# and cname who have bought a *pink* (colour) *Lamborghini* (prod#).

---

b. [2pt] Show products by prod# and pname that are owned by at least two customers.

---

c. [2pt] For each available colour (colour), show the most expensive product by prod# and pname that comes in that colour. (In case of ties for most expensive, list all in the tie.)

R	
A	B
a	b
c	b
e	f
g	h
i	h

S	
B	C
f	b
b	d
h	a
h	c
b	e

T	
A	C
g	a
a	b
i	c
e	d
c	e

Consider the three tables **R**, **S**, & **T** above for Questions 1d & 1e.

---

d. [2pt] Show the results of $\mathbf{R} \bowtie \mathbf{S}$.

---

e. [2pt] Show the results of $\mathbf{R} \bowtie (\mathbf{S} \bowtie \mathbf{T})$.

2. (10 points) **SQL.** *Some Quidditch League!* [Exercise]

   Consider the Movie database with the schema in Figure 2 on page 15 for the questions below.

---

   a. [5pt] Write an `SQL` query for the following.

      List each actor by name, gender, role, and the minutes they appear on screen in that role in a given movie by title, studio, and year such that "Hampton Fancher" was an author of the screenplay of the movie and the movie's genre is "SciFi".

b. [5pt] For each movie, report how many male actors and how many female actresses—gender = 'M' for *male* and gender = 'F' for *female*—were *cast* in the movie in columns male and female, respectively.

Only count a given actor or actress *once* per movie; that is, that a person may have played several *characters* (*roles*) in the movie does *not* count multiple times.

For full credit, the column male or female should report '0' (zero) if there were no actors or actresses, respectively, in the movie.

3. (10 points) **Query Logic.** *Take the **L** train to **Q** Street.* [Analysis]

---

a. [2pt] State in *plain, concise English* what the following SQL query over the *Movie database* does. (See the Movie schema in Figure 2 on page 15.)

Note that you will get *zero* credit if you use database terms in your answer! (E.g., "Well, the query first *joins* two tables, taking the *projection* of..." does not count!)

```
select P.p#, P.name as actor, D.p# as d#, D.name as director
from Person P, Person D
where not exists (
            select *
            from Movie M
            where M.director = D.p#
              and not exists (
                    select *
                    from Cast C
                    where C.actor = P.p#
                       and C.title = M.title
                       and C.studio = M.studio
                       and C.year = M.year
                  )
        )
   and exists (
            select *
            from Movie M
            where M.director = D.p#
        );
```

---

b. [2pt] Is the *having* clause *logically redundant* in SQL? That is, could one always write the "same" query not using the *having* clause?

Explain briefly why or why not.

c. [2pt] Given $\mathbf{R}(\underline{A}, \underline{B})$ and $\mathbf{S}(\underline{B}, \underline{C})$, rewrite

$$(\pi_{A}(\mathbf{R}) \times \pi_{C}(\mathbf{S})) - \pi_{A,C}(\mathbf{R} \bowtie \mathbf{S})$$

as an equivalent SQL query.

---

d. [2pt] Consider relations $\mathbf{R}(\underline{A}, \underline{B})$, $\mathbf{S}(\underline{B}, \underline{C})$, and $\mathbf{T}(\underline{C}, \underline{D})$.
Can $((\mathbf{R} \bowtie \mathbf{S}) \bowtie \mathbf{T})$ and $((\mathbf{R} \bowtie \mathbf{T}) \bowtie \mathbf{S})$ evaluate to different answer sets, or must they evaluate to the same answer set?
Show how they could evaluate differently, *or argue* why they must evaluate to the same.

---

e. [2pt] Consider the relations $\mathbf{R}(\underline{A}, B)$ and $\mathbf{S}(\underline{C}, D)$, and query

```
select distinct A, B from R
where exists (select * from S where A = C);
```

Write this as a relational-algebra expression.

4. [10pt] **Normalization.** *It's the end of the world as we know it...* [Analysis]

---

a. [3pt] Consider the relation **T** with attributes A, B, C, D, and E, and with the following functional dependencies (FDs):

$$\begin{array}{ll} \mathsf{AB} \mapsto \mathsf{C} & \mathsf{C} \mapsto \mathsf{A} \\ \mathsf{AD} \mapsto \mathsf{E} & \mathsf{C} \mapsto \mathsf{D} \end{array}$$

Dr. Datta Bas claims that the decomposition step of **T** into ABC & CDE is a *lossless-join* decomposition step.

Explain convincingly *either* that he is correct *or* that he is wrong.

---

b. [2pt] Consider the relation **R** with attributes A, B, and C, and with just the one following functional dependency (FD):

$$\mathsf{A} \mapsto \mathsf{BC}$$

Consider the decomposition of **R** into AB and BC. Construct a counterexample (example tuples) that demonstrate that this decomposition is *not* lossless (that is, it is *lossy*).

For Questions 4c to 4e, consider the relation **R** with attributes A, B, C, and D, and with the following functional dependencies (FDs):

$$\mathsf{AB} \mapsto \mathsf{C} \qquad \mathsf{A} \mapsto \mathsf{D}$$
$$\mathsf{BD} \mapsto \mathsf{C}$$

---

c. [1pt] What are the keys of **R**?

---

d. [3pt] Decompose **R** losslessly into a BCNF decomposition.

---

e. [1pt] Is your decomposition in your answer to Question 4d dependency preserving? Why or why not?

5. [10pt] **General.** *...and I feel fine.* [MULTIPLE CHOICE]

Choose *one* best answer for each of the following. Each is worth one point. There is no negative penalty for a wrong answer.

In the *rare* case that you feel a clarification to your answer is needed, write a brief clarification on the side.

Let $|\mathbf{T}|$ denote the number of tuples in $\mathbf{T}$.

---

a. [1pt] Consider the schema

$\mathbf{R}(\underline{\mathsf{A}}, B)$ FK (B) refs $\mathbf{S}$

$\mathbf{S}(A, \underline{\mathsf{B}})$ FK (A) refs $\mathbf{R}$

Note that none of the attributes are nullable. Which of the following is guaranteed to produce as many as, or more, tuples than each of the others?

A. $\mathbf{R} \bowtie \mathbf{S}$
B. $\pi_{\mathsf{A}}(\mathbf{R}) \bowtie \mathbf{S}$
C. $\pi_{\mathsf{A}}(\mathbf{R}) \bowtie \pi_{\mathsf{B}}(\mathbf{S})$
D. $\mathbf{R} \bowtie \pi_{\mathsf{B}}(\mathbf{S})$
E. *There is not enough information to answer this.*

---

b. [1pt] Assume $|\mathbf{R}| > 0$ and $|\mathbf{S}| > 0$. If one natural joins tables $\mathbf{R}$ and $\mathbf{S}$, but $\mathbf{R}$ and $\mathbf{S}$ have no column names in common, then

A. it is an *error*.
B. the answer set is an empty table.
C. it is an outer join.
D. it is the same as $\mathbf{R} \cap \mathbf{S}$.
E. it is the same as $\mathbf{R} \times \mathbf{S}$.

For Questions 5c & 5d, consider the schema

$\mathbf{R}(\underline{\mathsf{A}}, B)$ FK (B) refs $\mathbf{R}$ (A)

---

c. [1pt] What is the *smallest* that $|\mathbf{R} \bowtie \pi_{\mathsf{A}\rightarrow\mathsf{B},\mathsf{B}\rightarrow\mathsf{A}}(\mathbf{R})|$ can be?

A. 0
B. $|\mathbf{R}|$
C. $\frac{1}{2}|\mathbf{R}|$
D. $2|\mathbf{R}|$
E. $|\mathbf{R}|^2$

---

d. [1pt] What is the *largest* that $|\mathbf{R} \bowtie \pi_{\mathsf{A}\rightarrow\mathsf{B},\mathsf{B}\rightarrow\mathsf{A}}(\mathbf{R})|$ can be?

A. 0
B. $|\mathbf{R}|$
C. $\frac{1}{2}|\mathbf{R}|$
D. $2|\mathbf{R}|$
E. $|\mathbf{R}|^2$

---

e. [1pt] The technique of *synthesis* for normalization always

A. works only if there will be no multi-attribute keys.
B. achieves a 3NF schema, but it may not be dependency-preserving.
C. achieves a dependency-preserving, 3NF schema.
D. achieves a BCNF schema, but it may not be dependency-preserving.
E. achieves a dependency-preserving, BCNF schema.

---

f. [1pt] Which of the following `SQL` queries is illegal?

A. `select * from T;`
B. `select count(*) from T;`
C. `select count(*) from T group by A;`
D. `select count(*), max (B) from T group by A;`
E. `select max(count(*)) from T group by A;`

---

g. [1pt] Which of the following `SQL` queries is illegal?

A. `select A from T;`
B. `select A, count(*) from T;`
C. `select A, count(*) from T group by A;`
D. `select A, count(*) from T group by A, B;`
E. `select A, B, count(*) from T group by A, B;`

---

h. [1pt] Consider the schema $\mathbf{R}(\underline{A}, B)$, $\mathbf{S}(\underline{A}, \underline{D})$, and $\mathbf{T}(\underline{D}, B)$. One of the following relational-algebra expressions is not like the others. That is, one of them may evaluate differently from the other four. Which one?

A. $\pi_{B}(\mathbf{R} \bowtie \mathbf{T} \bowtie \mathbf{S})$
B. $\pi_{B}((\mathbf{R} \bowtie \mathbf{T}) \cap (\mathbf{S} \times \pi_{B}(\mathbf{T})))$
C. $\pi_{B}(\mathbf{R} \bowtie \mathbf{S}) \cap \pi_{B}(\mathbf{S} \bowtie \mathbf{T})$
D. $\pi_{B}(\mathbf{R} \bowtie \mathbf{S} \bowtie \mathbf{S} \bowtie \mathbf{T})$
E. $\pi_{B}(\mathbf{R} \bowtie \mathbf{S} \bowtie \mathbf{T})$

---

i. [1pt] Consider table **R**(<u>A</u>, B) for which B is of type *integer* and $|\mathbf{R}| = n > 0$. How many tuples will the query

```
select A from R where B <= 5 or B > 5;
```

return?
A. 0
B. $\frac{1}{2}n$
C. n
D. n^2
E. *There is not enough information to answer this.*

---

j. [1pt] Consider the relation **Enrol** with attributes sid, cid, term, and grade which stores academic records of students. Attribute sid is a student identifier and cid is a class—a given section of a course in a given term—identifier.

Here is a query involving **Enrol**:

```
select distinct cid
    from ( select * from Enrol E1
                where not exists
                    (select *
                        from Enrol E2
                        where E2.cid = E1.cid
                              and E2.grade > E1.grade)
            ) as V
    where grade = 7;
```

Which of the following queries must return the same result as the query above?

I.
```
select distinct E.cid
    from Enrol E
    where E1.grade = 7
except
select distinct E.cid
    from Enrol E
    where E1.grade > 7
```

II.
```
select distinct cid
    from Enrol
    group by cid
    having max(grade) = 7;
```

A. **I** only.
B. **II** only.
C. Both **I** and **II**.
D. Neither **I** nor **II**.
E. There is not enough information available to determine this.

Extra Space

Extra Space

Relax. Turn in your test. Return to the wild!

Reference *(Detach this page for convenience, if you want.)*

Schema for the Colours Database.

Customer	
cust#	PK
cname	
fav_colour	
phone#	

Item	
item#	PK
prod#	FK to **Product**
cust#	FK to **Customer**
colour	
date_sold	

Product	
prod#	PK
pname	
cost	
maker	FK to **Company**

Avail_Colours	
prod#	PK, FK to **Product**
colour	PK

Figure 1: Colours Schema.

Schema for the Movie Database.

Person(<u>p#</u>, *name*, birthdate, *nationality*, *gender*)
Actor(<u>p#</u>, *aguild#*)
  FK (p#) refs **Person**
Director(<u>p#</u>, *dguild#*)
  FK (p#) refs **Person**
Writer(<u>p#</u>, *wguild#*)
  FK (p#) refs **Person**
Studio(<u>name</u>)
ScreenPlay(<u>title</u>, <u>year</u>)
Authored(<u>title</u>, <u>year</u>, <u>writer</u>)
  FK (title, year) refs **ScreenPlay**
  FK (writer) refs **Writer** (p#)
Movie(<u>title</u>, <u>studio</u>, <u>year</u>, *genre*, *director*, *length*)
  FK (studio) refs **Studio** (name)
  FK (title, year) refs **ScreenPlay**
  FK (director) refs **Director** (p#)
Cast(<u>title</u>, <u>studio</u>, <u>year</u>, <u>role</u>, <u>actor</u>, *minutes*)
  FK (title, studio, year) refs **Movie**
  FK (actor) refs **Actor** (p#)
Affiliated(<u>director</u>, <u>studio</u>)
  FK (director) refs **Director** (p#)
  FK (studio) refs **Studio** (name)

Figure 2: Movie Schema.

REFERENCE

The Normal Form Definitions.

1NF: Domain of each attribute is an *elementary* type; that is, not a *set* or a *record structure*.

2NF: Whenever $\mathcal{X} \mapsto A$ is a functional dependency that holds in relation R and $A \notin \mathcal{X}$, then either

- A is *prime*, or
- $\mathcal{X}$ is not a proper subset of any key for R.

3NF: Whenever $\mathcal{X} \mapsto A$ is a functional dependency that holds in relation R and $A \notin \mathcal{X}$, then either

- A is *prime*, or
- $\mathcal{X}$ is a key or a super-key for R.

BCNF: Whenever $\mathcal{X} \mapsto A$ is a functional dependency that holds in relation R and $A \notin \mathcal{X}$, then

- $\mathcal{X}$ is a key or a super-key for R.

An attribute A is called *prime* if A is in any of the candidate keys.

Figure 3: The Normal Forms.