CSE 344 Midterm

Wednesday, February 19, 2014, 14:30-15:20

Name: ______________________

Question	Points	Score
1	30	
2	50	
3	12	
4	8	
Total:	100	

- This exam is open book and open notes but NO laptops or other portable devices.
- You have 50 minutes; budget time carefully.
- Please read all questions carefully before answering them.
- Some questions are easier, others harder. Plan to answer all questions, do not get stuck on one question. If you have no idea how to answer a question, write your thoughts about the question for partial credit.
- Good luck!

Reference for SQL Syntax

Outer Joins

```
-- left outer join with two selections:
SELECT * FROM R LEFT OUTER JOIN S ON R.x = 55 AND R.y = S.z AND S.u = 99
```

The CREATE TABLE Statement:

```
CREATE TABLE R (
       attrA VARCHAR(30) PRIMARY KEY,
       attrB int REFERENCES S(anotherAttr),
       attrC CHAR(20),
       attrD TEXT,
       -- PRIMARY KEY (attrA),     (equivalently)
       FOREIGN KEY (attrC, attrD) REFERENCES  T(anAttrC, anAttrD))
```

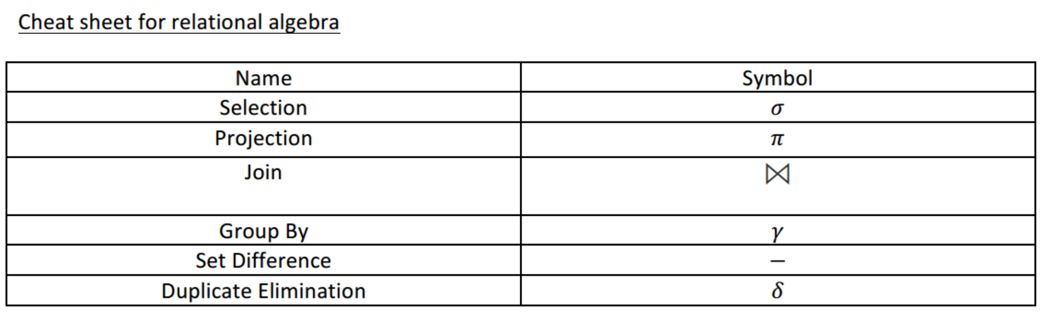
The CASE Statement:

```
SELECT R.name, (CASE WHEN R.rating=1 THEN 'like it'
                     when R.rating=0 THEN 'do not like it'
                     when R.rating IS NULL THEN 'do not know'
                     ELSE 'unknown' END) AS my_rating
FROM R;
```

The WITH Statement

```
WITH T AS (SELECT * FROM R WHERE R.K > 10),
     S AS (SELECT * FROM R WHERE R.a > 50)
SELECT * FROM T, S WHERE T.K<20 AND S.a < 20
```

Reference for the Relational Algebra



Cheat sheet for relational algebra

Name	Symbol
Selection	σ
Projection	π
Join	$\bowtie$
Group By	γ
Set Difference	$-$
Duplicate Elimination	δ

1 SQL

1. (30 points) There are many websites nowadays that offer *Massive Open Online Courses (MOOC)*, where professors from top-class universities teach online courses. Students across the world have free online access to enroll in these courses and can learn different topics this way. The director of such an organization *www.GreatMooc.org*, Dr. Alice VeryStrict, wanted to analyze the performance of the participating students and professors, and the quality of the offered courses to ensure that everything is going well. Since you are a student of CSE 344, can you help her write the following queries to do this data analysis? The information about students, instructors, and courses is stored in the following relations.

   Course(cid, name, year, duration)
   Student(sid, name, univ)
   Instructor(tid, name, univ)
   Enrollment(sid, cid) — sid, cid reference $Student$ and $Course$ respectively
   Teaches(tid, cid) – tid, cid reference $Instructor$ and $Course$ respectively.

   Assume that

   - Duration is in month, so takes value between 1 to 12.
   - Each course is entirely contained within the same *year*, i.e. does not span across two or more years.
   - Feel free to abbreviate the relation names as C, S, I, E, T.
   - The schema is shown on the top of the pages where you need it.

   Recall that *variance* of n numbers $x_1, \cdots, x_n$ is defined as

   $$\frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2 \quad = \quad \frac{1}{n}\sum_{i=1}^{n} x_i^2 - \mu^2$$

   where $\mu = \frac{1}{n}\sum_{i=1}^{n} x_i$ denotes the average of $x_1, \cdots, x_n$.

Course(cid, name, year, duration),	Student(sid, name, univ)

Instructor(tid, name, univ),	Enrollment(sid, cid),	Teaches(tid, cid)

(a) (9 points) First, Dr. VeryStrict wanted to examine the duration of courses offered by the professors from different universities in different years.

So, write an SQL query to output
the average duration (as *avg_duration*), maximum duration (as *max_duration*) and the variance of duration (as *var_duration*) of the courses offered in the same year by professors from the same universities. Further, the answers should be sorted in **decreasing** order of the years (universities can be output in any order).

e.g. The answer to the query should be of the form

year	univ	avg_duration	max_duration	var_duration
2013	'MIT'	2.8	5.2	0.001
2013	'UW'	3.5	4.4	0.09
2012	'UW'	1.9	2.7	0.02

...

The first answer tuple $(2013, \text{'}MIT', 2.8, 5.2, 0.001)$ has been computed using all courses offered in year = 2013 that were taught by some professor from MIT.

Solution:

```
SELECT C.year,
       I.univ,
       avg(C.duration) as avg_duration,
       max(C.duration) as max_duration,
       ((sum(C.duration * C.duration) * 1.0)/count(*) -
        avg(C.duration) * avg(C.duration)) as var_duration
FROM Course C, Teaches T, Instructor I
WHERE C.cid = T.cid
  AND T.tid = I.tid
GROUP BY C.year, I.univ
ORDER BY C.year DESC
```

====================================

Grading strategy: Points are divided as below: Average duration: 1 point; Max duration: 1 point; Variance calculation: 3 points; where clause: 1 point; group by: 1 point; order by: 1 point; select query elements/syntax: 1 point; If the variance calculation is incomplete or incorrect, points are deducted (between 0.5 to 2 points) based on the error.

Course(cid, name, year, duration),	Student(sid, name, univ)

Instructor(tid, name, univ),	Enrollment(sid, cid),	Teaches(tid, cid)

(b) (6 points) How would you modify the above answer if we change the requirements in the following three ways? **DO NOT write the query once again. Just say which clause(s) you will add/modify and how**.

We want to output *avg, max, and variance of durations* of the courses for the *year, univ* pairs such that ...

i. ... *all* courses offered in that *year* from that *univ* had duration ≥ 3 months.

Solution: Add

```
HAVING min(C.duration) >= 3
```

ii. ...*some* courses offered in that *year* from that *univ* had duration ≥ 3 months.

Solution: Add

```
HAVING max(C.duration) >= 3
```

iii. ... the avg, max, and variance are computed (for each *univ*) over only those courses which had duration ≥ 3 months.

Solution: Add to the WHERE clause

```
AND C.duration >= 3
```

==================================

Grading strategy: Attempt: 1 point; 0.5 or 1 point deducted if explanation/query is incomplete

Course(cid, name, year, duration), Student(sid, name, univ)

Instructor(tid, name, univ), Enrollment(sid, cid), Teaches(tid, cid)

(c) (15 points) Dr. VeryStrict wanted to find the instructors who were unpopular in the year 2012, but gained popularity rapidly in the year 2013.

Write an SQL query to output the *name* and *univ* of the instructors and the total number of (distinct) courses ever taught by him/her (as *num_courses*) who has taught at most 50 distinct students (in all of his/her courses) in the year 2012 (so 0 students enrolled or no courses taught in 2012 should be included), but at least 300 distinct students in 2013.

Solution:

Solution 1: (an elegant solution from some of you without using a WITH or LEFT OUTER JOIN)

```
SELECT I.name,
       I.univ,
       count(distinct T.cid) as num_courses
FROM  Instructor I, Teaches T
WHERE I.tid = T.tid and
        T.tid NOT IN (
            SELECT T1.tid
            FROM Course C1, Teaches T1, Enrollment E1
            WHERE T1.cid = C1.cid and C1.cid = E1.cid and C1.year = 2012
            GROUP BY T1.tid
            HAVING count(distinct E1.sid) > 50) and
        T.tid IN (
            SELECT T2.tid
            FROM Course C2, Teaches T2, Enrollment E2
            WHERE T2.cid = C2.cid and C2.cid = E2.cid and C2.year = 2013
            GROUP BY T2.tid
            HAVING count(distinct E2.sid) >= 300)
  GROUP BY I.tid, I.name, I.univ
```

NOTE:

1. 'NOT IN' and '> 50' is important in the above solution, 'IN' '$\leq$ 50' will not include instructors who did not teach in 2012 or for whom no students were enrolled in 2012.

2. Another way to write the same query (also from some of you), is to use the following correlated nested subquery instead of uncorrelated subquery using 'NOT IN' and '> 50' above (similarly for the second subquery).

```
 WHERE ..... and
   50 < (SELECT count(distinct E1.sid)
         FROM Course C1, Teaches T1, Enrollment E1
         WHERE T.tid = T1.tid and ------ CORRELATION WITH OUTER T
               T1.cid = C1.cid and C1.cid = E1.cid and C1.year = 2012)
and ....
```

<u>**Solution 2: Using WITH and LEFT OUTER JOIN**</u>

```
WITH CandidateTeachers2012 AS
         (SELECT T.tid
            FROM Course C, Teaches T, Enrollment E
            WHERE T.cid = C.cid and C.cid = E.cid and C.year = 2012
            GROUP BY T.tid
            HAVING count(distinct E.sid) <= 50),
     CandidateTeachers2013 AS
            (SELECT T.tid
             FROM Course C, Teaches T, Enrollment E
             WHERE T.cid = C.cid and C.cid = E.cid and C.year = 2013
             GROUP BY T.tid
             HAVING count(distinct E.sid) >= 300)
SELECT I.name,
       I.univ,
       count(distinct T.cid) as num_courses
FROM  Instructor I, Teaches T
WHERE I.tid = T.tid and
        T.tid in (SELECT CT1.tid
                  FROM CandidateTeachers2013 CT1 left outer join
                       CandidateTeachers2012 CT2 on
                       CT1.tid = CT2.tid)
  GROUP BY I.tid, I.name, I.univ
```

Note that

1. left outer join is important to include instructors who did not teach any

course in 2012 or no students enrolled in his/her course in 2012.

2. For both the solutions, *GROUP BY I.tid, I.name, I.univ* is important (instead of *GROUP BY I.name, I.univ*) so that distinct instructors from the same university with the same name are considered separately.

(An almost correct) Solution 3:
Simply join T with two copies of (C, T, E): (C1, T1, E1) and (C2, T2, E2), and another copy of C, C3, to output total number of distinct courses taught by an instructor. C1 should be courses taught in 2012, C2 for 2013. Group by T.tid, T.name, T.univ. HAVING clause should ensure that the number of distinct E1.sid in 2012 is ≤ 50 and distinct E2.sid in 2013 is ≥ 300.
Note that it will miss instructors who did not teach any course in 2012 or no students enrolled in his/her course in 2012, therefore it is not fully correct!

==================================
Grading strategy: I gave up to 10 points for logic (even if the syntax is incorrect), and deducted 0.5 to 5 points for wrong syntax. In particular, for missing those special instructors (if you did not use a LEFT OUTER JOIN or something equivalent), then I took 0.5 off. I did not deduct any point for missing *tid* in GROUP BY. Some of you interpreted the question that *every* course taught by the instructor should have ≤ 50 students. This interpretation was also accepted.

2 Datalog, Relational Calculus, Relational Algebra

2. (50 points)

   In this problem you will continue to help Dr. VeryStrict to analyze the online courses in *www.GreatMooc.org*. (**You do not need to look at Problem 1 or your answers to Problem 1 for this problem**).

   The information about students, instructors, and courses is stored in the following relations.

   Course(<u>cid</u>, name, year, duration)
   Student(<u>sid</u>, name, univ)
   Instructor(<u>tid</u>, name, univ)
   Enrollment(<u>sid, cid</u>) — sid, cid reference $Student$ and $Course$ respectively
   Teaches(<u>tid, cid</u>) – tid, cid reference $Instructor$ and $Course$ respectively.

   Assume that

   - Duration is in month, so takes value between 1 to 12.
   - Each course is entirely contained within the same *year*, i.e. does not span across two or more years.
   - Feel free to abbreviate the relation names as C, S, I, E, T.
   - The schema is shown on the top of the pages where you need it.

   (a) (30 points) We want to output the name of the students who in the year 2012 took only courses taught by professors from univ = 'UW',
   Note: your answer should include students who did not take any course in 2012.

   i. Write a Relational CALCULUS expression for this query.

   Solution:

$$\begin{aligned} & Ans(n) \\ = \; & \exists s\ \exists v\ S(s,n,v) \land \forall c\ \forall t\ \forall m\ \forall u\ \forall n\ \forall d \\ & \quad ((E(s,c)\ \land\ T(t,c)\ \land I(t,m,u) \land C(c,n,2012,d)) \Rightarrow (u =' UW')) \end{aligned}$$

Course(cid, name, year, duration),	Student(sid, name, univ)

Instructor(tid, name, univ),	Enrollment(sid, cid),	Teaches(tid, cid)

ii. Write the query in SQL.

Solution: It is easier to start with a datalog+negation program: (i) find NonAns student ids, who took a course in 2012 taught by an instructor not from 'UW', (ii) ignore all these NonAns students to output the answer.

```
NonAns(s) :- E(s, c), T(t, c), I(t, _, u),
                        C(c, _, 2012, _), u != 'UW'
Ans(n) :- S(s, n, _), NOT NonAns(s)
```

Then we transform it to SQL:

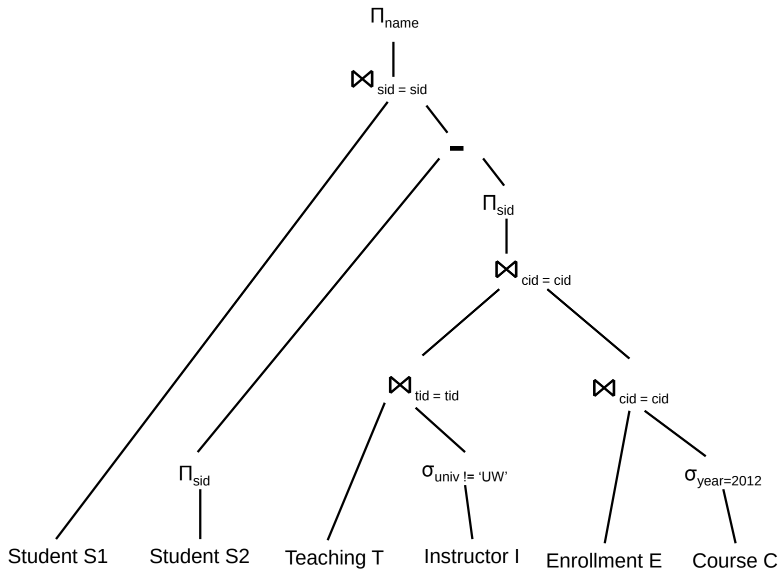
```
SELECT S.name
FROM Student S
WHERE S.sid NOT IN
      (SELECT E.sid
         FROM Enrollment E, Teaches T, Instructor I, Course C
         WHERE E.cid = T.cid
           AND T.tid = I.tid
           AND E.cid = C.cid
           AND C.year = 2012
           AND I.univ <> 'UW')
```

Course(<u>cid</u>, name, year, duration),	Student(<u>sid</u>, name, univ)

Instructor(<u>tid</u>, name, univ),	Enrollment(<u>sid, cid</u>),	Teaches(<u>tid, cid</u>)

iii. Write a Relational ALGEBRA expression (or a logical query plan as a tree) for this query.

Solution:
It is quite straightforward to transform the nested sub-query to an RA expression. However, for set difference at the top, the two relations involved in the operation must have the same schema. The sub-query only outputs *sid*, whereas we need *name* as final output. So we would need an additional join with students relation S before or after the difference operation. Here is a logical query plan, and several other plans are possible.



====================================

Grading strategy: 10 points are allocated for each of i, ii, iii. For each of these questions, 4 points are for valid syntax, and 6 points are for correct semantics.

Course(<u>cid</u>, name, year, duration),	Student(<u>sid</u>, name, univ)

Instructor(<u>tid</u>, name, univ),	Enrollment(<u>sid, cid</u>),	Teaches(<u>tid, cid</u>)

(b) (20 points) We want to output the name of the courses where all enrolled students are from the same university (but students in two different courses can be from two different universities).

Note: your answer should include courses where no students were enrolled.

i. Write a Relational CALCULUS expression for this query.

Solution:

$$\begin{aligned} & Ans(n) \\ = \; & \exists c\ \exists y\ \exists d\ \ C(c,n,y,d)\ \wedge \\ & (\forall s1\ \forall s2\ \forall n1\ \forall n2\ \forall u1\ \forall u2 \\ & \ (S(s1,n1,u1)\ \wedge\ E(s1,c)\ \wedge\ S(s2,n2,u2)\ \wedge E(s2,c))\ \Rightarrow (u1 = u2)) \end{aligned}$$

=====================================

Grading strategy: -2 If use only one E, S; -1 The relations are in the wrong place; -1 Use of '_'; -1 Did not quantify all the variables; -2 Did not have an implies (only uses and).

ii. Write a non-recursive datalog + negation expression for this query.

Solution: You may want to convert the RC expression to datalog + negation systematically, but it is not necessary for this query (you can directly find the NonAns set). Here is the conversion from the above RC expression

$$\begin{aligned} & Ans(n) \\ = \ & \exists c\ \exists y\ \exists d\ \ C(c,n,y,d)\ \wedge\ (\forall s1\ \forall s2\ \forall n1\ \forall n2\ \forall u1\ \forall u2 \\ & (\neg(S(s1,n1,u1)\ \wedge\ E(s1,c)\ \wedge\ S(s2,n2,u2)\ \wedge E(s2,c))) \vee (u1 = u2)) \\ = \ & \exists c\ \exists y\ \exists d\ \ C(c,n,y,d)\ \wedge\ \ \neg(\exists s1\ \exists s2\ \exists n1\ \exists n2\ \exists u1\ \exists u2 \\ & ((S(s1,n1,u1)\ \wedge\ E(s1,c)\ \wedge\ S(s2,n2,u2)\ \wedge E(s2,c))) \wedge (u1! = u2)) \end{aligned}$$

Then convert it to datalog + negation.
You can directly find $NonAnsCourse(c)$ as well which says find all courses where two students are enrolled who are not from the same university.

```
NonAnsCourse(c) :-  S(s1, _, u1), E(s1, c), S(s2, _, u2),
                                        E(s2, c), u1 != u2
Ans(n) :- C(c, n, _, _), NOT NonAnsCourse(c)
========================================
```

Grading strategy: -1 Extra relation; -1 The relations are in the wrong place; -1 Forget the negation

3 XML, XPath, XQuery

3. (12 points)

   Consider the following XML document *midterm.xml* (the header has been omitted).

```
<a m="1">
  <b n="1" o="2">
    <c p="3">3</c>
    <d/>
  </b>
  <b n="1" o="2">
    <c p="3">3</c>
    <f s="1"/>
    <d q="3">
      <e r="2">2</e>
    </d>
  </b>
</a>
```

   (a) (6 points) Consider the following XPath expressions and write **true/false** for the following claims (**if false, write the correct value, but no explanations are needed**). **Note:** The "count()" method returns the number of nodes in a node-set.

   i. `count(/*//*) = 9`

   Solution: False.
   The correct answer is 8.
   Here we want to count *all descendant elements of all child elements of the document node*. The document node has one child element (root element): a. The element "a" has the following descendant elements: b, c, d, b, c, f, d, e. Also note that although the names of some of the elements appear to be duplicated, they are distinct from each other.

   ii. `count(/*/*/@*) = 4`

   Solution: True.
   Here we want to count *all attributes of all child elements of all child elements of the document node*. The document node has one child element: a. The

element “a” has two children “b”. Each of these “b” elements have two attributes each, “n” and “o”.

iii. `count(/*/*/*) = 8`

Solution: False.
The correct answer is 5.
Here we want to count *all child elements of all child elements of all child elements of the document node*, and there are five of them: c, d, c, f, d.

====================================

Grading strategy: 2 points for TRUE answer; 1 point for FALSE answer and 1 point for correct value.

(b) (2 points) Does midterm.xml match with the following DTD? (write **YES/NO**). **If your answer is NO**, point to the line in the question where it does not match (**just draw arrow(s) next to the line(s) in the DTD, and add a one-line explanation**).

```
<?xml version="1.0"?>
  <!DOCTYPE a [
    <!ELEMENT a (b+)>
    <!ELEMENT b (c, d, f*)>
    <!ELEMENT c (#PCDATA)>
    <!ELEMENT d (e?)>
    <!ELEMENT f (#PCDATA)>
   ]>
```

Solution: NO.
The order of the elements is violated for

`<!ELEMENT b (c, d, f*)>`

=====================================

Grading strategy:
Yes/No: 1 point; Identifying the incorrect line in the schema correctly with reason: 1 point

(c) (4 points) Suppose we want to write an XQuery to find all "c" elements with value 3. Do the following queries work? Write **YES/NO. If your answer is NO, add a one-line explanation**.

i.
```
let $t := doc("midterm.xml")/a/b/c
where $t = 3
  return $t
```

Solution: No.
Here t is the sequence of all c elements, so the where-clause is true if any of these c elements has value 3.

ii.
```
for $t in doc("midterm.xml")/a/b/c
  where $t = 3
  return <c>{$t}</c>
```

Solution: No.
There will be two $< c >$ ... $< /c >$ tags.

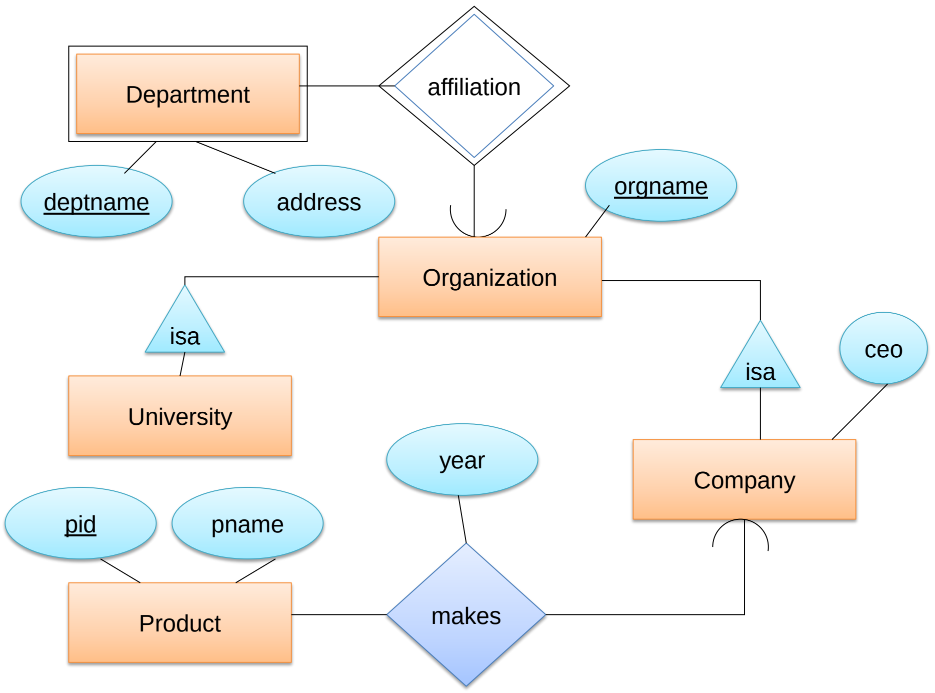
==================================

Grading strategy:
Yes/No: 1 point;   Reason: 1 point

4 E/R Diagram

4. (8 points)

Write down the CREATE TABLE statements to create the *Department* and *Product* relations from the E/R diagram below. Note that you have to declare the primary keys (using PRIMARY KEY) and foreign keys (using REFERENCES, assume the same relation name as the entities) to get full credit. Assume all the attributes are of type VARCHAR(20).



(a) (4 points) Write CREATE TABLE statement for Department.

Solution:

```
CREATE TABLE Department (deptname VARCHAR(20),
                         address VARCHAR(20),
                         orgname VARCHAR(20) REFERENCES Organization,
                         PRIMARY KEY(deptname, orgname))
```

(b) (4 points) Write CREATE TABLE statement for Product.

Solution:

```
CREATE TABLE Product (pid VARCHAR(20) PRIMARY KEY,
                      pname VARCHAR(20),
                      year VARCHAR(20),
                      orgname VARCHAR(20) REFERENCES Company)
```

Note that Company table will have orgname as the primary key which will also have a reference to the Organization table.

====================================

Grading Strategy: -1 Missing attribute; -1 Wrong Primary Key; -1 Missing Foreign Key; -1 Referencing the wrong attribute

For Rough Use

For Rough Use