/***

CSE532 -- Project 2 Report

Author(s): Minghui Lin (109557872)

Yishuo Wang (108533945)

Report Documentation

I (or We, if working with a partner) pledge my (or our) honor that all parts of this project were done by me (or us) alone and without collaboration with anybody else.

***/

The division of work：

Yishuo Wang:

1 Discuss the ER diagram and design it.

2 Insert all the data

3 Change the default schema to the object relational schema.

4.Query 1,2,4 and Query1.jsp, Query2.jsp, Query4.jsp. Query5 direct part.

5.Write the report

Minghui Lin:

1 Discuss the ER diagram and design it..

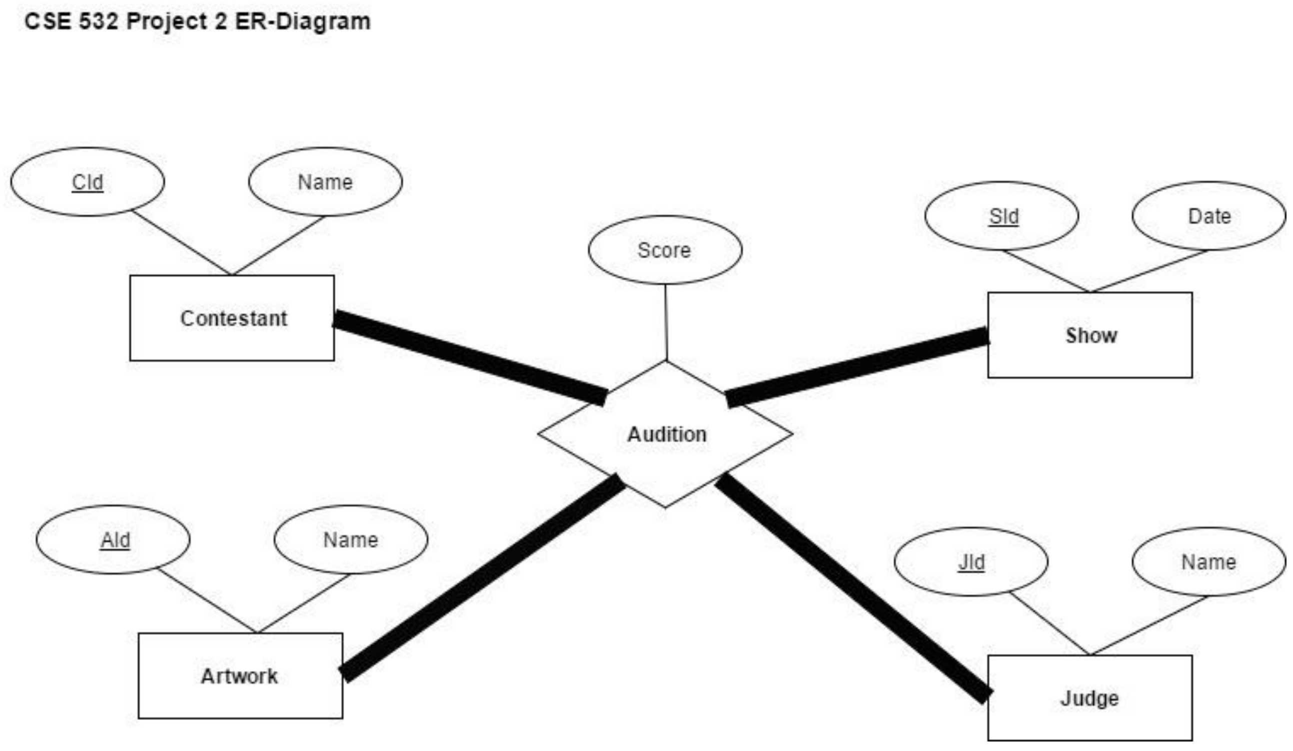
2 Create the default schema.

3 Setup the jsp, jdbc, tomcat, Netbeans to run successfully.

4.Query 3, 5 and Query3.jsp, Query5.jsp

5.Write the report

Entity-Relationship (ER) diagram:



A clear description of the database scheme:

1) There are 4 entity types and 1 relationship type in our database scheme.

2) The 4 entity types are: Contestant, Show, Artwork and Judge.

Attributes of Contestant are: CId and Name

Primary Key of Contestant: CId.

Attributes of Show are: SId and Date

Primary Key of Show: SId

Attributes of Artwork are: AId, Name

Primary Key of Artwork: AId

Attributes of Judge are: JId and Name

Primary Key of Judge: JId

The Audition is relationship type which connects to the entity types of Contestant, Show, Artwork and Judge.

Audition has its own attribute which is Score.

Audition has 4 foreign keys which are combined as its foreign key as well. They are:

CId, SId, AId and JId.

This database scheme is very simply, we can query information from Audition table and connect to the other entity type to get extra information if we needed.

Description of integrity constraints, including referential integrity and CHECK-constraints whenever applicable:

Contestant:

CId is Primary Key.

Show:

SId is Primary Key

Artwork:

AId is Primary Key

Judge:

JId is Primary Key

Audition:

CId, SId, AId and JId are foreign keys to CId in Contestant, SId in Show, AId in Artwork and JId in Judge.

CId, SId, AId and JId are Primary key of Audition.

CHECK-constraints:

We check Attribute of Score must be between 0 and 10.

All SQL CREATE TABLE/VIEW/TYPE commands used to build the database. These statements must include all the applicable FOREIGN KEY and CHECK statements.

Create Types:

```
CREATE TYPE public."Artwork_Type" AS
(
        "AId" character(30),
        "Name" character(30)
);


CREATE TYPE public."Contestant_Type" AS
(
        "CId" character(30),
        "Name" character(30)
);


CREATE TYPE public."Judge_Type" AS
(
        "JId" character(30),
        "Name" character(30)
);
CREATE TYPE public."Show_Type" AS
(
        "SId" character(30),
        "Date" date
);


CREATE TYPE public."Audition_Type" AS
```

```
(
        "Score" integer,
        "CId" character(30),
        "AId" character(30),
        "SId" character(30),
        "JId" character(30)
);
```

Create Tables:

```
CREATE TABLE public."Artwork"
   OF "Artwork_Type"
(
   CONSTRAINT "Artwork_pkey" PRIMARY KEY ("AId")
)
WITH (
   OIDS = TRUE
);

CREATE TABLE public."Contestant"
   OF "Contestant_Type"
(
   CONSTRAINT "Contestant_pkey" PRIMARY KEY ("CId")
)
WITH (
   OIDS = TRUE
);

CREATE TABLE public."Judge"
```

```
    OF "Judge_Type"
(
    CONSTRAINT "Judge_pkey" PRIMARY KEY ("JId")
)
WITH (
    OIDS = TRUE
);

CREATE TABLE public."Show"
    OF "Show_Type"
(
    CONSTRAINT "Show_pkey" PRIMARY KEY ("SId")
)
WITH (
    OIDS = TRUE
);

CREATE TABLE public."Audition"
    OF "Audition_Type"
(
    CONSTRAINT "Audition_pkey" PRIMARY KEY ("CId", "AId", "SId", "JId"),
    CONSTRAINT "AId" FOREIGN KEY ("AId")
        REFERENCES public."Artwork" ("AId") MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION,
    CONSTRAINT "CId" FOREIGN KEY ("CId")
        REFERENCES public."Contestant" ("CId") MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION,
    CONSTRAINT "JId" FOREIGN KEY ("JId")
        REFERENCES public."Judge" ("JId") MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION,
    CONSTRAINT "SId" FOREIGN KEY ("SId")
        REFERENCES public."Show" ("SId") MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION,
    CONSTRAINT "Audition_Score_check" CHECK ("Score" >= 0 AND "Score" <= 10)
)
WITH (
    OIDS = TRUE
);
```

Create Views:

We will create views based on different query, please see each query for different views we have created.

For each query mentioned in Section 3, the report must supply the appropriate SQL statement.

Query 1:

```
SELECT DISTINCT c1."Name", c2."Name"
FROM public."Contestant" c1, public."Contestant" c2,
    public."Audition" a1, public."Audition" a2,
    public."Show" s1, public."Show" s2
WHERE c1."CId" != c2."CId" AND c1."Name" < c2."Name"
    AND a1."CId" = c1."CId" AND a2."CId" = c2."CId" AND a1."AId" = a2."AId" AND
a1."Score" = a2."Score"
```

```
   AND a1."SId" = s1."SId" AND a2."SId" = s2."SId" AND s1."Date" = s2."Date"
```

Query 2:

```
SELECT R.name1, R.name2 FROM
(
        SELECT DISTINCT c1."Name" name1, c2."Name" name2,
                MAX(a1."Score") OVER (PARTITION BY a1."CId", a1."SId", a1."AId")
max1,
                MAX(a2."Score") OVER (PARTITION BY a2."CId", a2."SId", a2."AId")
max2,
                MIN(a1."Score") OVER (PARTITION BY a1."CId", a1."SId", a1."AId")
min1,
                MIN(a2."Score") OVER (PARTITION BY a2."CId", a2."SId", a2."AId")
min2
        FROM "Contestant" c1, "Contestant" c2,
                "Audition" a1, "Audition" a2
        WHERE c1."CId" != c2."CId" AND c1."Name" < c2."Name"
        AND a1."CId" = c1."CId" AND a2."CId" = c2."CId" AND a1."AId" = a2."AId"
        GROUP BY name1, name2,
                a1."Score", a2."Score",
                a1."CId", a2."CId",
                a1."SId", a2."SId",
                        a1."AId", a2."AId"
) R

WHERE R.max1 = R.max2 AND R.min1 = R.min2
ORDER BY R.name1
```

Query 3:

```
CREATE OR REPLACE VIEW "SAME_ARTWORK" AS
        SELECT a1."CId", a1."AId", count(a1."JId") "JUDGE_NUM", avg(a1."Score")
"AVG_SCORE"
        FROM "Audition" a1
        GROUP BY a1."CId", a1."AId", a1."SId"
        ORDER BY a1."CId";


SELECT DISTINCT C1."Name" name1, C2."Name" name2
        FROM "SAME_ARTWORK" S1, "SAME_ARTWORK" S2, "Contestant" C1,
"Contestant" C2
        WHERE
        S1."CId" = C1."CId" AND
                S2."CId" = C2."CId" AND
        S1."CId" != S2."CId" AND
                C1."Name" < C2."Name" AND
                S1."AId" = S2."AId" AND
                S1."JUDGE_NUM" = S2."JUDGE_NUM" AND
                S1."AVG_SCORE" = S2."AVG_SCORE"
        ORDER BY C1."Name";
```

Query 4:

```
CREATE OR REPLACE VIEW "Show_Contestants" AS
        SELECT a1."CId", a1."SId"
        FROM "Audition" a1
        GROUP BY a1."CId", a1."SId"
        ORDER BY a1."CId";
```

```
SELECT DISTINCT contestant1."Name", contestant2."Name"
FROM "Contestant" contestant1, "Contestant" contestant2
WHERE NOT EXISTS
        (
        SELECT DISTINCT c1."Name", c2."Name"
  FROM
        "Contestant" c1,
        "Contestant" c2,
                "Show_Contestants" sc1,
        "Show_Contestants" sc2
        WHERE
        contestant1."CId" = c1."CId"
        AND contestant2."CId" = c2."CId"
        AND c1."CId" != c2."CId"
                AND sc1."CId" = c1."CId"
        AND sc2."CId" = c2."CId"
                AND sc2."SId" NOT IN (SELECT sc3."SId"
                     FROM "Show_Contestants" sc3
                     WHERE sc3."CId" = sc1."CId"
                     )
  )
        AND contestant1."CId" != contestant2."CId"
-----------------------------------------------------
```

Query 5:

```
-------------------
--  Recusive part(This is Recursive View) --
-------------------
```

```
CREATE OR REPLACE RECURSIVE VIEW "CLOSRURE" (cid1, cid2) AS
    SELECT DISTINCT view1."CId" cid1, view2."CId" cid2
    FROM "AVG_VIEW" view1, "AVG_VIEW" view2,
        "Audition" A1, "Audition" A2
    WHERE
            view1."CId" != view2."CId" AND
        view1."CId" = A1."CId" AND
        view2."CId" = A2."CId" AND
        A1."AId" = A2."AId" AND
        A1."SId" = A2."SId" AND
        abs(view1."AVG_SCORE" - view2."AVG_SCORE") <= 0.2
UNION
    SELECT DISTINCT CL."cid1" cid2, view2."CId" cid1
    FROM "AVG_VIEW" view1, "AVG_VIEW" view2,
        "Audition" A1, "Audition" A2,
        "CLOSRURE" CL
    WHERE
        view1."CId" != view2."CId" AND
        view1."CId" = A1."CId" AND
        view2."CId" = A2."CId" AND
        A1."AId" = A2."AId" AND
        A1."SId" = A2."SId" AND
        abs(view1."AVG_SCORE" - view2."AVG_SCORE") <= 0.2 AND
        view1."CId" = CL."cid2";


--------------------------------
--  Display final result part --
--------------------------------
```

```
SELECT C1."Name", C2."Name"
FROM "CLOSRURE" CL, "Contestant" C1, "Contestant" C2
WHERE CL."cid1" = C1."CId" AND
CL."cid2" = C2."CId" AND
C1."Name" < C2."Name"
ORDER BY C1."Name",C2."Name";
```

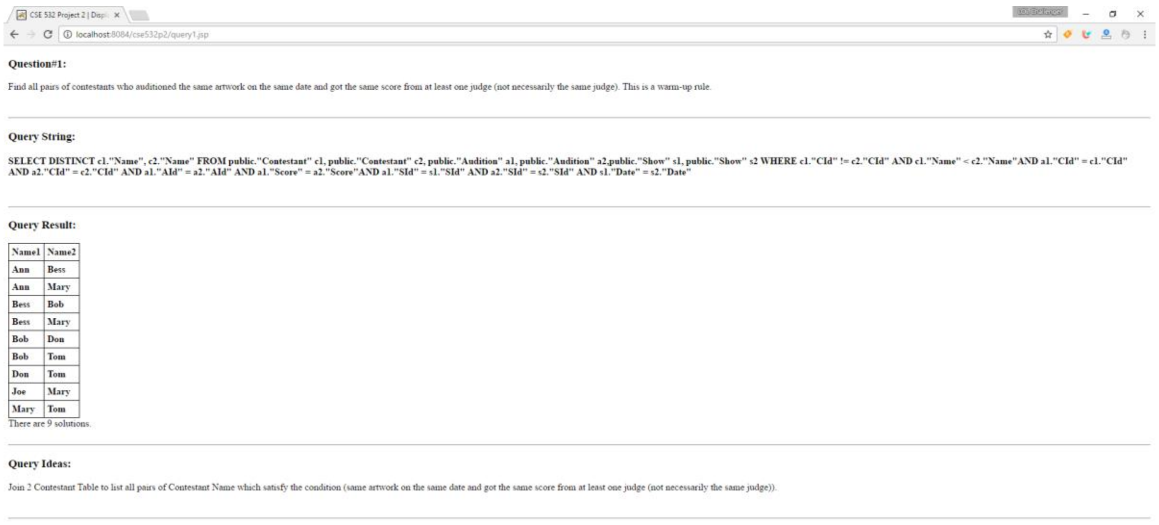
A brief user guide. Explain how to install and run your program.

To Run our program is very simple.

Please open Netbean IDE and open our project under “Coding” directory and Press Run button in Netbean to run the program.

Please make sure you have installed Tomcat with NetBeans configuration.

The following screenshot are from our running program:



Question#1:

Find all pairs of contestants who auditioned the same artwork on the same date and got the same score from at least one judge (not necessarily the same judge). This is a warm-up rule.

Query String:

```
SELECT DISTINCT c1."Name", c2."Name" FROM public."Contestant" c1, public."Contestant" c2, public."Audition" a1, public."Audition" a2,public."Show" s1, public."Show" s2 WHERE c1."CId" != c2."CId" AND c1."Name" < c2."Name"AND a1."CId" = c1."CId"
AND a2."CId" = c2."CId" AND a1."AId" = a2."AId" AND a1."Score" = a2."Score"AND a1."SId" = s1."SId" AND a2."SId" = s2."SId" AND s1."Date" = s2."Date"
```

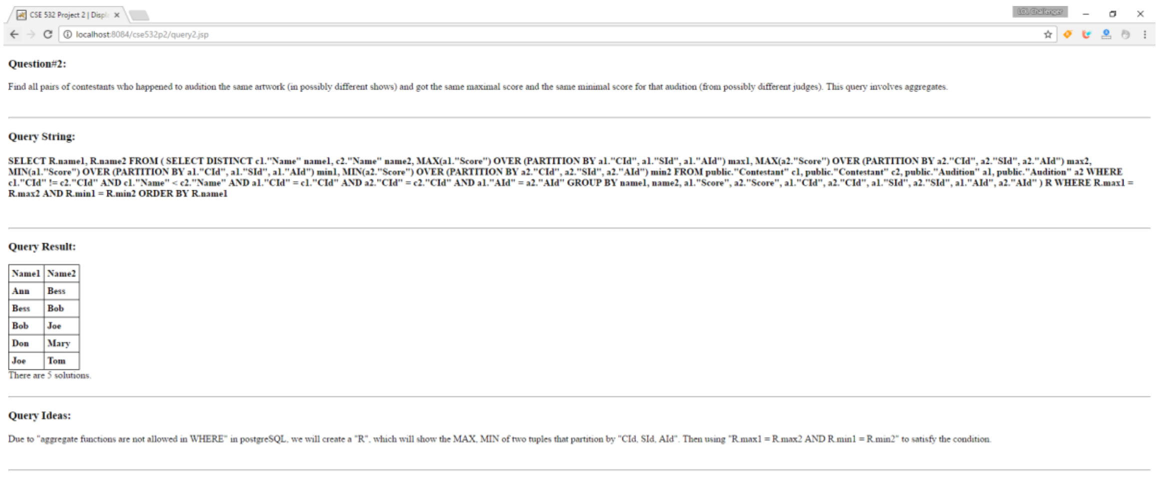
Query Result:

Name1	Name2
Ann	Bess
Ann	Mary
Bess	Bob
Bess	Mary
Bob	Don
Bob	Tom
Don	Tom
Joe	Mary
Mary	Tom

There are 9 solutions.

Query Ideas:

Join 2 Contestant Table to list all pairs of Contestant Name which satisfy the condition (same artwork on the same date and got the same score from at least one judge (not necessarily the same judge)).



Question#2:

Find all pairs of contestants who happened to audition the same artwork (in possibly different shows) and got the same maximal score and the same minimal score for that audition (from possibly different judges). This query involves aggregates.

Query String:

```
SELECT R.name1, R.name2 FROM ( SELECT DISTINCT c1."Name" name1, c2."Name" name2, MAX(a1."Score") OVER (PARTITION BY a1."CId", a1."SId", a1."AId") max1, MAX(a2."Score") OVER (PARTITION BY a2."CId", a2."SId", a2."AId") max2,
MIN(a1."Score") OVER (PARTITION BY a1."CId", a1."SId", a1."AId") min1, MIN(a2."Score") OVER (PARTITION BY a2."CId", a2."SId", a2."AId") min2 FROM public."Contestant" c1, public."Contestant" c2, public."Audition" a1, public."Audition" a2 WHERE
c1."CId" != c2."CId" AND c1."Name" < c2."Name" AND a1."CId" = c1."CId" AND a2."CId" = c2."CId" AND a1."AId" = a2."AId" GROUP BY name1, name2, a1."Score", a2."Score", a1."CId", a2."CId", a1."SId", a2."SId", a1."AId", a2."AId" ) R WHERE R.max1 =
R.max2 AND R.min1 = R.min2 ORDER BY R.name1
```

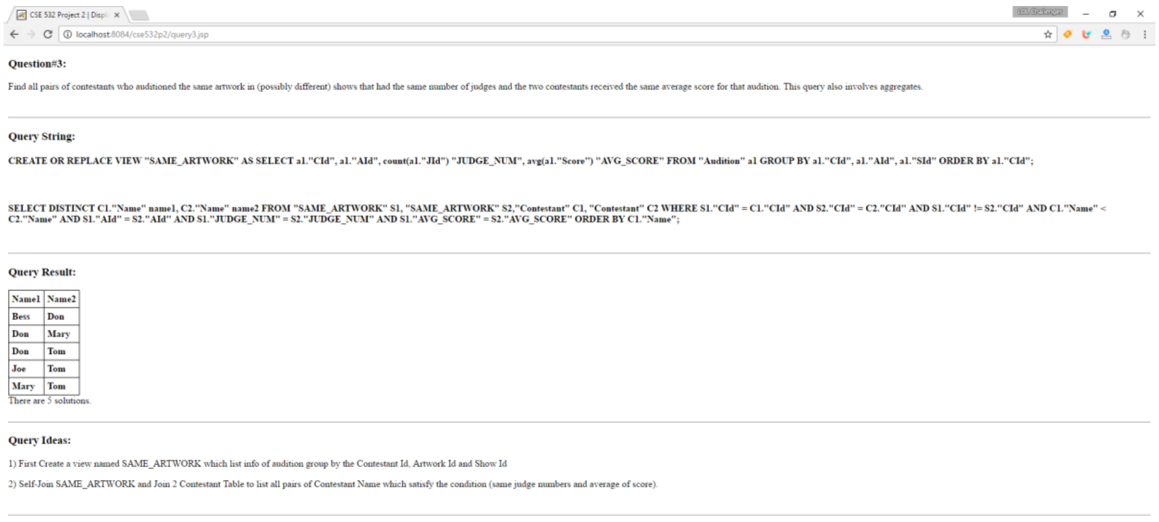
Query Result:

Name1	Name2
Ann	Bess
Bess	Bob
Bob	Joe
Don	Mary
Joe	Tom

There are 5 solutions.

Query Ideas:

Due to "aggregate functions are not allowed in WHERE" in postgreSQL, we will create a "R", which will show the MAX, MIN of two tuples that partition by "CId, SId, AId". Then using "R.max1 = R.max2 AND R.min1 = R.min2" to satisfy the condition.



Question#3:

Find all pairs of contestants who auditioned the same artwork in (possibly different) shows that had the same number of judges and the two contestants received the same average score for that audition. This query also involves aggregates.

Query String:

```
CREATE OR REPLACE VIEW "SAME_ARTWORK" AS SELECT a1."CId", a1."AId", count(a1."JId") "JUDGE_NUM", avg(a1."Score") "AVG_SCORE" FROM "Audition" a1 GROUP BY a1."CId", a1."AId", a1."SId" ORDER BY a1."CId";


SELECT DISTINCT C1."Name" name1, C2."Name" name2 FROM "SAME_ARTWORK" S1, "SAME_ARTWORK" S2,"Contestant" C1, "Contestant" C2 WHERE S1."CId" = C1."CId" AND S2."CId" = C2."CId" AND S1."CId" != S2."CId" AND C1."Name" <
C2."Name" AND S1."AId" = S2."AId" AND S1."JUDGE_NUM" = S2."JUDGE_NUM" AND S1."AVG_SCORE" = S2."AVG_SCORE" ORDER BY C1."Name";
```

Query Result:

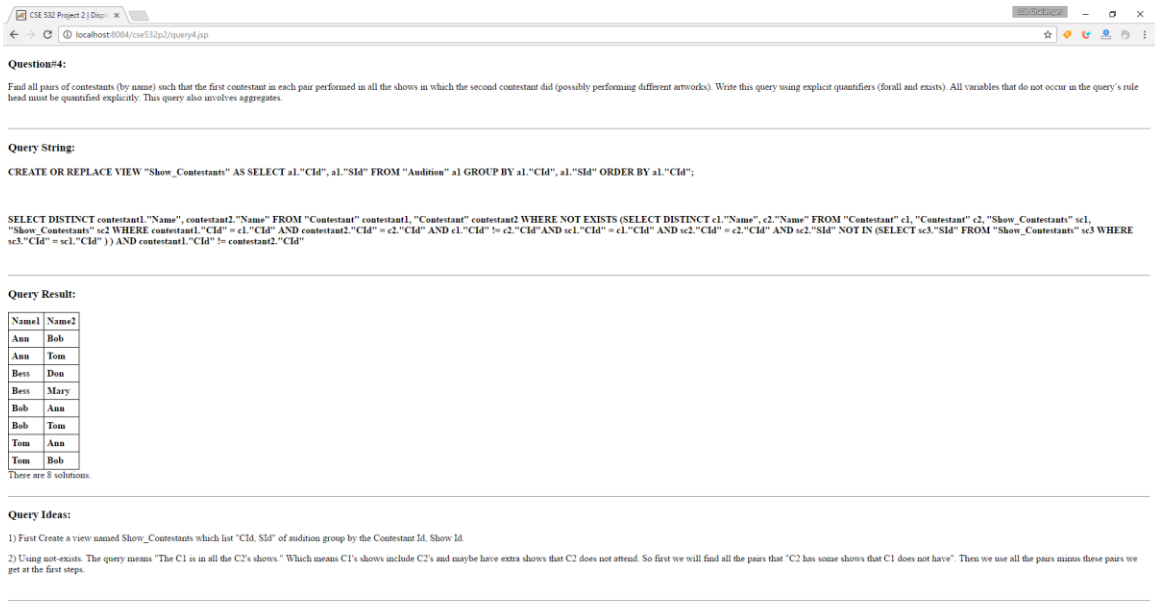
Name1	Name2
Bess	Don
Don	Mary
Don	Tom
Joe	Tom
Mary	Tom

There are 5 solutions.

Query Ideas:

1) First Create a view named SAME_ARTWORK which list info of audition group by the Contestant Id, Artwork Id and Show Id

2) Self-Join SAME_ARTWORK and Join 2 Contestant Table to list all pairs of Contestant Name which satisfy the condition (same judge numbers and average of score).



CSE 532 Project 2 | Displ

localhost:8084/cse532p2/query4.jsp

Question#4:

Find all pairs of contestants (by name) such that the first contestant in each pair performed in all the shows in which the second contestant did (possibly performing different artworks). Write this query using explicit quantifiers (forall and exists). All variables that do not occur in the query's rule head must be quantified explicitly. This query also involves aggregates.

Query String:

```
CREATE OR REPLACE VIEW "Show_Contestants" AS SELECT a1."CId", a1."SId" FROM "Audition" a1 GROUP BY a1."CId", a1."SId" ORDER BY a1."CId";
```

```
SELECT DISTINCT contestant1."Name", contestant2."Name" FROM "Contestant" contestant1, "Contestant" contestant2 WHERE NOT EXISTS (SELECT DISTINCT c1."Name", c2."Name" FROM "Contestant" c1, "Contestant" c2, "Show_Contestants" sc1, "Show_Contestants" sc2 WHERE contestant1."CId" = c1."CId" AND contestant2."CId" = c2."CId" AND c1."CId" != c2."CId"AND sc1."CId" = c1."CId" AND sc2."CId" = c2."CId" AND sc2."SId" NOT IN (SELECT sc3."SId" FROM "Show_Contestants" sc3 WHERE sc3."CId" = sc1."CId" ) ) AND contestant1."CId" != contestant2."CId"
```

Query Result:

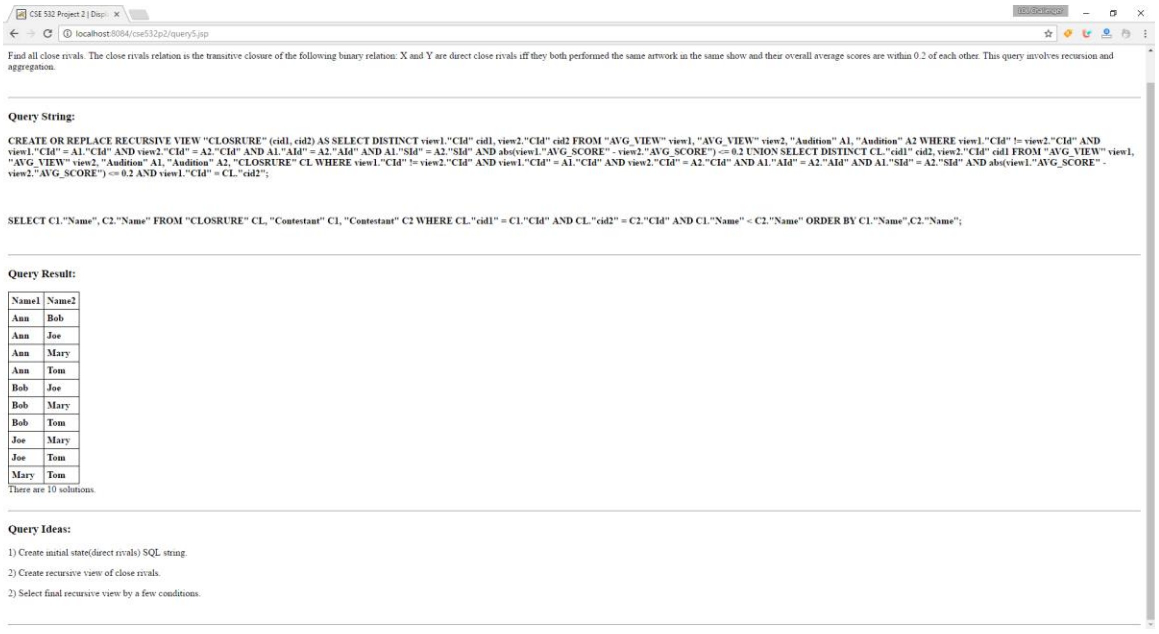
Name1	Name2
Ann	Bob
Ann	Tom
Bess	Don
Bess	Mary
Bob	Ann
Bob	Tom
Tom	Ann
Tom	Bob

There are 8 solutions.

Query Ideas:

1) First Create a view named Show_Contestants which list "CId, SId" of audition group by the Contestant Id, Show Id.

2) Using not-exists. The query means "The C1 is in all the C2's shows." Which means C1's shows include C2's and maybe have extra shows that C2 does not attend. So first we will find all the pairs that "C2 has some shows that C1 does not have". Then we use all the pairs minus these pairs we get at the first steps.



CSE 532 Project 2 | Displ

localhost:8084/cse532p2/query5.jsp

Find all close rivals. The close rivals relation is the transitive closure of the following binary relation: X and Y are direct close rivals iff they both performed the same artwork in the same show and their overall average scores are within 0.2 of each other. This query involves recursion and aggregation.

Query String:

```
CREATE OR REPLACE RECURSIVE VIEW "CLOSRURE" (cid1, cid2) AS SELECT DISTINCT view1."CId" cid1, view2."CId" cid2 FROM "AVG_VIEW" view1, "AVG_VIEW" view2, "Audition" A1, "Audition" A2 WHERE view1."CId" != view2."CId" AND view1."CId" = A1."CId" AND view2."CId" = A2."CId" AND A1."AId" = A2."AId" AND A1."SId" = A2."SId" AND abs(view1."AVG_SCORE" - view2."AVG_SCORE") <= 0.2 UNION SELECT DISTINCT CL."cid1" cid2, view2."CId" cid1 FROM "AVG_VIEW" view1, "AVG_VIEW" view2, "Audition" A1, "Audition" A2, "CLOSRURE" CL WHERE view1."CId" != view2."CId" AND view1."CId" = A1."CId" AND view2."CId" = A2."CId" AND A1."AId" = A2."AId" AND A1."SId" = A2."SId" AND abs(view1."AVG_SCORE" - view2."AVG_SCORE") <= 0.2 AND view1."CId" = CL."cid2";
```

```
SELECT C1."Name", C2."Name" FROM "CLOSRURE" CL, "Contestant" C1, "Contestant" C2 WHERE CL."cid1" = C1."CId" AND CL."cid2" = C2."CId" AND C1."Name" < C2."Name" ORDER BY C1."Name",C2."Name";
```

Query Result:

Name1	Name2
Ann	Bob
Ann	Joe
Ann	Mary
Ann	Tom
Bob	Joe
Bob	Mary
Bob	Tom
Joe	Mary
Joe	Tom
Mary	Tom

There are 10 solutions.

Query Ideas:

1) Create initial state(direct rivals) SQL string.

2) Create recursive view of close rivals.

2) Select final recursive view by a few conditions.