```
/*************************************************************************
CSE532 -- Project 2
File name: DbDesign
Author(s): Shachee Mishra (109915951)
           Phani Krishna Penumarthi (109951184)

We pledge our honor that all parts of this project were done by us alone and without
collaboration with anybody else.


*************************************************************************/
/*

DATABASE DESIGN:

Tables:

Type: PersonType.
Table: Person - Typed table of PersonType.
Table: Contestants inherits from Person.
        Primary Key: id.
Judges inherits from Person.
        Primary Key: id.

Type: PieceType
Table: Pieces - Typed table of PieceType
        Primary Key: Pid.

Table: Showdet - For saving ShowId, ShowName and date of Shows.
        Primary Key: Sid.

Table: ShowJ - Show and Judge pair. Each row represents a show and judge combination. This is
done so that it is easier to add new judges to a show, and also to list down all the judges
belonging to a show.
        Primary Key: sjId.
        Foreign keys: 1. Reference to judge in judges table -    ON DELETE CASCADE.
                      2. Reference to Show in ShowDet table -    ON DELETE CASCADE.
                ON DELETE CASCADE: All the rows with a judge or show get deleted if the
                corresponding one is deleted from main table, so references are present in
                ShowJ table.

Table: Scores - Stores marks received by an artist in a show for a piece performed.
        Primary Key: sjId, contestant, piece .
            Assumption: A contestant performs a piece only once in a given show.

        Foreign Key: sjId: Show-Judge pair - reference to ShowJ - ON DELETE CASCADE.
                     Contestant ID: The contestant who performed - reference to Contestants - ON
                     DELETE CASCADE.
                     Piece ID: Piece performed - reference to Pieces - ON DELETE CASCADE.
                     marks: points received. Check - should be more than 0.
            ON DELETE CASCADE: This constraint is applied so that, for rmeoval of an entity
```

```
            from parent table, its corresponding references are removed from this table too.

*/

/*

Work Division:
The installation of PostGreSQL was done by Shachee.
The installation of Apache server, and setting eclipse environment was done by Phani.
This was done on two different machines initially.
Design decisions were taken by both of us.
We strictly followed the policy: "I will work only if the partner is available".
And all the work is performed with strong collaboration by both of us.
Finally, we integrated both queries and the front-end, to make a working demo.
Both of us helped each other in installing the software, and the demo can be shown on both
machines now.

*/
/*

To install:

Run SQL statements till all the insert commands.
This will create the database and populate values.

To Run queries:

Run each query one by one to get the results.

*/

CREATE DATABASE cse532
  WITH OWNER = postgres
       ENCODING = 'UTF8'
       TABLESPACE = pg_default
       LC_COLLATE = 'English_United States.1252'
       LC_CTYPE = 'English_United States.1252'
       CONNECTION LIMIT = -1;




CREATE  TYPE PersonType AS (id integer,name char(20));

create type PieceType as (
    Pid integer,
    PName char(20)
);

create table Person of PersonType;

create table Pieces of PieceType(
    PRIMARY KEY(Pid)
```

```
);


CREATE TABLE IF NOT EXISTS Contestants (PRIMARY KEY(id)) INHERITS (Person) ;

CREATE TABLE IF NOT EXISTS Judges (PRIMARY KEY(id)) INHERITS (Person) ;

create TABLE IF NOT EXISTS ShowDet(
 Sid integer PRIMARY KEY,
 Sname char(20),
 Sdate date
);

create Table if not exists ShowJ(
    sjId integer PRIMARY KEY,
    judgeId integer REFERENCES Judges ON DELETE CASCADE ,
    showId INTEGER REFERENCES ShowDet  ON DELETE CASCADE
);


create Table if not exists Scores(
    sjId integer REFERENCES ShowJ ON DELETE CASCADE,
    Contestant integer REFERENCES Contestants (id) ON DELETE CASCADE,
    Piece integer REFERENCES Pieces(Pid) ON DELETE CASCADE,
    marks integer,
    PRIMARY KEY(sjId,Contestant, Piece),
    CHECK (marks>0)
);

Insert into Judges values(1,'Judy');
Insert into Judges values(2,'Lucy');
Insert into Judges values(3,'Irving');
Insert into Judges values(4,'Oscar');
Insert into Judges values(5,'Phil');


Insert into Contestants values(10,'Joe');
Insert into Contestants values(11,'Mary');
Insert into Contestants values(12,'Bess');
Insert into Contestants values(13,'Don');
Insert into Contestants values(14,'Ann');
Insert into Contestants values(15,'Bob');
Insert into Contestants values(16,'Tom');

Insert into ShowDet values(20,'Show1','20140202');
Insert into ShowDet values(21,'Show2','20140402');
Insert into ShowDet values(22,'Show3','20140602');
Insert into ShowDet values(23,'Show4','20140802');
Insert into ShowDet values(24,'Show5','20141005');

Insert into Pieces values(50,'Barcarolle');
Insert into Pieces values(51,'Giselle');
Insert into Pieces values(52,'Besame Mucho');
```

```
Insert into Pieces values(53,'Swan Lake');
Insert into Pieces values(54,'Habanera');
Insert into Pieces values(55,'The Tramp');

/*  Show1: Date=20140202, Judges=Judy, Lucy, Irving */

INSERT INTO ShowJ Values(30, 1, 20);
INSERT INTO ShowJ Values(31, 2, 20 );
INSERT INTO ShowJ Values(32, 3, 20 );

/*  Show2: Date=20140402, Judges=Judy, Phil */

INSERT INTO ShowJ Values(33, 1, 21 );
INSERT INTO ShowJ Values(34, 5, 21 );

/* Show3: Date=20140602, Judges=Irving, Phil, Oscar */
INSERT INTO ShowJ Values(35, 3 , 22 );
INSERT INTO ShowJ Values(36, 5 , 22 );
INSERT INTO ShowJ Values(37, 4 , 22 );

/* Show4: Date=20140802, Judges=Lucy, Oscar */
INSERT INTO ShowJ Values(38, 2 , 23 );
INSERT INTO ShowJ Values(39, 4 , 23 );

/* Show5: Date=20141005, Judges=Lucy, Irving, Phil */
INSERT INTO ShowJ Values(40, 2 , 24 );
INSERT INTO ShowJ Values(41, 3 , 24 );
INSERT INTO ShowJ Values(42, 5 , 24 );

/*
  Show1: Date=20140202, Judges=Judy, Lucy, Irving

INSERT INTO ShowJ Values(30, 1, 20);
INSERT INTO ShowJ Values(31, 2, 20 );
INSERT INTO ShowJ Values(32, 3, 20 );

*/

Insert into Scores values(30, 10,50, 7);
Insert into Scores values(31, 10,50, 8);
Insert into Scores values(32, 10,50, 6);
Insert into Scores values(30, 11,50, 5);
Insert into Scores values(31, 11,50, 6);
Insert into Scores values(32, 11,50, 6);
Insert into Scores values(30, 11,51, 9);
Insert into Scores values(31, 11,51, 6);
Insert into Scores values(32, 11,51, 8);
Insert into Scores values(30, 12,52, 4);
Insert into Scores values(31, 12,52, 5);
Insert into Scores values(32, 12,52, 6);
Insert into Scores values(30, 13,52, 9);
Insert into Scores values(31, 13,52, 9);
Insert into Scores values(32, 13,52, 7);
```

```
Insert into Scores values(30, 13,53, 7);
Insert into Scores values(31, 13,53, 7);
Insert into Scores values(32, 13,53, 10);

Insert into Scores values(33, 15,53, 8);
Insert into Scores values(34, 15,53, 6);
Insert into Scores values(33, 11,54, 3);
Insert into Scores values(34, 11,54, 5);
Insert into Scores values(33, 11,52, 9);
Insert into Scores values(34, 11,52, 10);
Insert into Scores values(33, 14,54, 7);
Insert into Scores values(34, 14,54, 6);
Insert into Scores values(33, 12,51, 8);
Insert into Scores values(34, 12,51, 7);
Insert into Scores values(33, 16,52, 5);
Insert into Scores values(34, 16,52, 5);
Insert into Scores values(33, 16,55, 7);
Insert into Scores values(34, 16,55, 6);

Insert into Scores values(35, 15,55, 8);
Insert into Scores values(36, 15,55, 7);
Insert into Scores values(37, 15,55, 9);
Insert into Scores values(35, 14,52, 7);
Insert into Scores values(36, 14,52, 6);
Insert into Scores values(37, 14,52, 4);
Insert into Scores values(35, 12,53, 6);
Insert into Scores values(36, 12,53, 8);
Insert into Scores values(37, 12,53, 7);
Insert into Scores values(35, 16,50, 9);
Insert into Scores values(36, 16,50, 7);
Insert into Scores values(37, 16,50, 6);
Insert into Scores values(35, 13,51, 8);
Insert into Scores values(36, 13,51, 6);
Insert into Scores values(37, 13,51, 9);

Insert into Scores values(38, 10,55, 7);
Insert into Scores values(39, 10,55, 6);
Insert into Scores values(38, 15,50, 6);
Insert into Scores values(39, 15,50, 8);
Insert into Scores values(38, 14,52, 9);
Insert into Scores values(39, 14,52, 8);
Insert into Scores values(38, 16,55, 8);
Insert into Scores values(39, 16,55, 10);
Insert into Scores values(38, 16,50, 5);
Insert into Scores values(39, 16,50, 5);

Insert into Scores values(40, 15,53, 3);
Insert into Scores values(41, 15,53, 5);
Insert into Scores values(42, 15,53, 6);
Insert into Scores values(40, 15,51, 4);
Insert into Scores values(41, 15,51, 6);
Insert into Scores values(42, 15,51, 8);
Insert into Scores values(40, 11,50, 5);
```

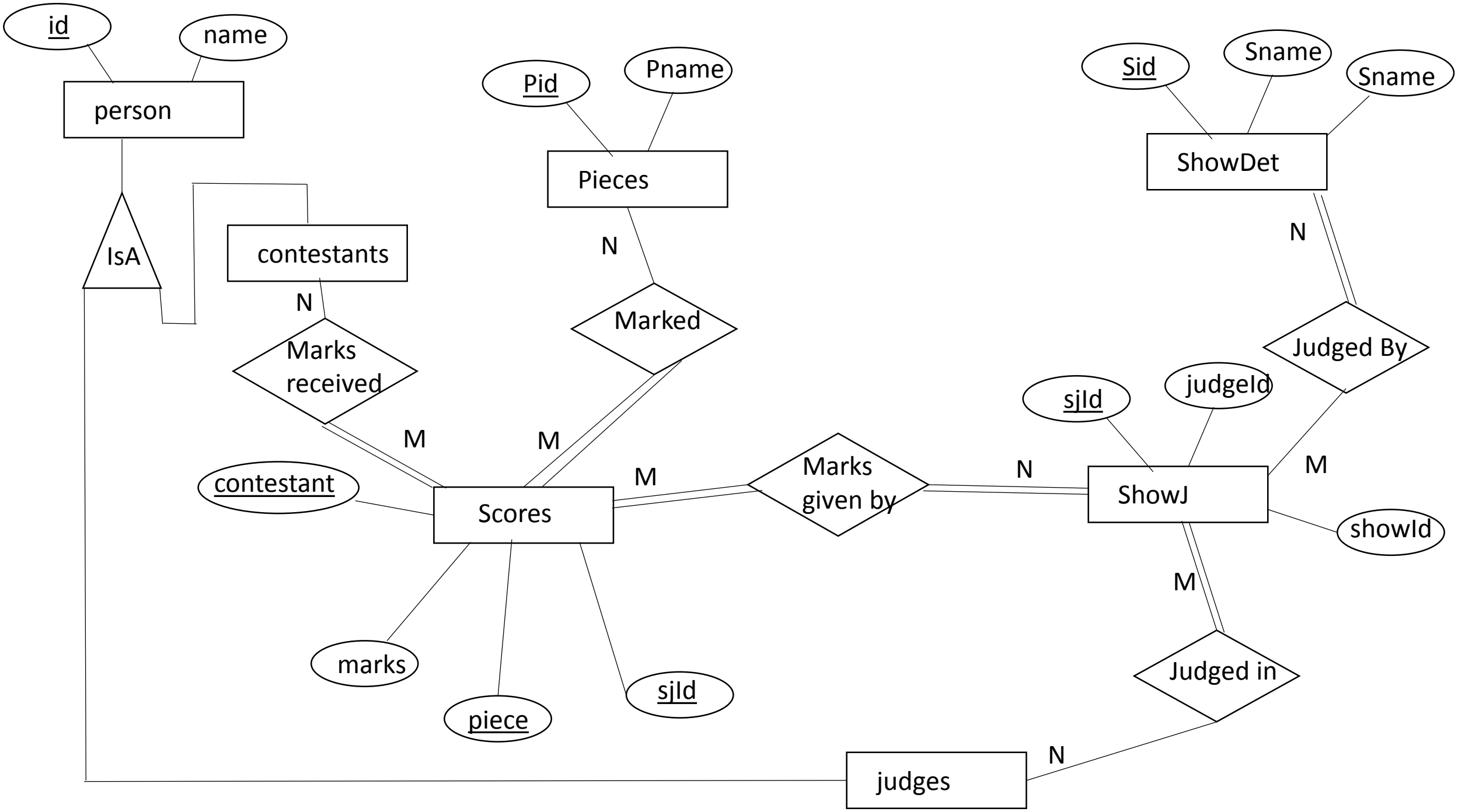
```
Insert into Scores values(41, 11,50, 7);
Insert into Scores values(42, 11,50, 10);
Insert into Scores values(40, 11,52, 6);
Insert into Scores values(41, 11,52, 8);
Insert into Scores values(42, 11,52, 7);
Insert into Scores values(40, 14,52, 7);
Insert into Scores values(41, 14,52, 9);
Insert into Scores values(42, 14,52, 6);
Insert into Scores values(40, 12,52, 9);
Insert into Scores values(41, 12,52, 8);
Insert into Scores values(42, 12,52, 8);
Insert into Scores values(40, 12,54, 8);
Insert into Scores values(41, 12,54, 7);
Insert into Scores values(42, 12,54, 10);
Insert into Scores values(40, 16,55, 6);
Insert into Scores values(41, 16,55, 6);
Insert into Scores values(42, 16,55, 8);
Insert into Scores values(40, 13,55, 5);
Insert into Scores values(41, 13,55, 8);
Insert into Scores values(42, 13,55, 7);




/*DB Creation ends here*/

/*SQL Queries*/



/*Query 1 - Find all pairs of contestants who happened to audition the same piece during the
same show
and got the same score from at least one judge */

Select distinct C1.name , C2.name
from Scores Sc, Scores Sc1 , Contestants C1, Contestants C2
where Sc.Contestant > Sc1.Contestant
    and     Sc.Piece = Sc1.Piece
    and     Sc.marks = Sc1.marks
    and
    (   select judgeId from ShowJ SJ where SJ.sjId = Sc.sjId )
        =
        (   select judgeId from ShowJ SJ where SJ.sjId = Sc1.sjId)
        and
        (select showId from ShowJ SJ where SJ.sjId = Sc.sjId )
        =
        (select showId from ShowJ SJ where SJ.sjId = Sc1.sjId)
    and Sc.Contestant = C1.id and Sc1.Contestant = C2.id
```

```
/*Query2 - Find all pairs of contestants who happened to audition the same piece (in possibly
different
shows) and got the same average score for that piece.*/


SELECT s1.cont, s2.cont
        FROM
        (SELECT avg(marks) as av, p.PName as artPiece, c.name as cont
            FROM Scores Sc3, Pieces p, Contestants c, ShowJ sj, ShowDet dt
            WHERE p.Pid = Sc3.Piece
            AND
                Sc3.Contestant = c.id
            AND
                Sc3.sjId = sj.sjId
            AND
                sj.showId = dt.Sid
            Group By p.PName, c.name, dt.Sid
            ORDER BY dt.Sid ) s1,
        (SELECT avg(marks) as av, p.PName as artPiece, c.name as cont
            FROM Scores Sc3, Pieces p, Contestants c, ShowJ sj, ShowDet dt
            WHERE p.Pid = Sc3.Piece
            AND
                Sc3.Contestant = c.id
            AND
                Sc3.sjId = sj.sjId
            AND
                sj.showId = dt.Sid
            Group By p.PName, c.name, dt.Sid
            ORDER BY dt.Sid ) s2
WHERE s1.av = s2.av
    AND
        s1.artPiece = s2.artPiece
    AND
        s1.cont < s2.cont
ORDER BY s2.cont


/*Query3 - Find all pairs of contestants who auditioned the same piece in (possibly different)
shows that
had at least 3 judges and the two contestants got the same highest score */

SELECT s1.cont, s2.cont
    FROM
        (SELECT max(marks) as max_score, p.PName as artPiece, c.name as cont
            FROM Scores s, Pieces p, Contestants c, ShowJ sj, ShowDet dt
            WHERE p.Pid = s.Piece
            AND
                s.Contestant = c.id
            AND
                s.sjId = sj.sjId
            AND
                sj.showId = dt.Sid
```

```
            Group By p.PName, c.name, dt.Sid
            HAVING COUNT(sj.judgeId) >= 3
            ORDER BY dt.Sid, cont ) s1,
        (SELECT max(marks) as max_score, p.PName as artPiece, c.name as cont
            FROM Scores s, Pieces p, Contestants c, ShowJ sj, ShowDet dt
            WHERE p.Pid = s.Piece
            AND
                s.Contestant = c.id
            AND
                s.sjId = sj.sjId
            AND
            sj.showId = dt.Sid
            Group By p.PName, c.name, dt.Sid
            HAVING COUNT(sj.judgeId) >= 3
            ORDER BY dt.Sid, cont ) s2
    WHERE s1.max_score = s2.max_score
    AND
        s1.artPiece = s2.artPiece
    AND
        s1.cont < s2.cont
    ORDER BY s2.cont



/*Query4 - Find all pairs of contestants such that the first contestants has performed all the
pieces of the
second contestant (possibly in different shows) */

SELECT DISTINCT record1.cont, record2.cont
FROM
    (SELECT DISTINCT c.name as cont, p.PName as artPiece
        FROM Scores s, Pieces p, Contestants c, ShowJ sj, ShowDet dt
        WHERE p.Pid = s.Piece
        AND
            s.Contestant = c.id
        AND
            s.sjId = sj.sjId
        AND
            sj.showId = dt.Sid
        ORDER BY cont
    ) record1,
    (SELECT DISTINCT c.name as cont, p.PName as artPiece
        FROM Scores s, Pieces p, Contestants c, ShowJ sj, ShowDet dt
        WHERE p.Pid = s.Piece
        AND
            s.Contestant = c.id
        AND
            s.sjId = sj.sjId
        AND
            sj.showId = dt.Sid
        ORDER BY cont
    ) record2
WHERE record1.cont != record2.cont
```

```
AND
(
    (SELECT COUNT(*) as c_intersect
        FROM(
            (SELECT DISTINCT artPiece from (SELECT DISTINCT c.name as cont, p.PName as artPiece
                FROM Scores s, Pieces p, Contestants c, ShowJ sj, ShowDet dt
                WHERE p.Pid = s.Piece
                AND
                    s.Contestant = c.id
                AND
                    s.sjId = sj.sjId
                AND
                    sj.showId = dt.Sid
                ORDER BY cont
            ) r1
        where r1.cont = record1.cont)
        INTERSECT
            (SELECT DISTINCT artPiece from (SELECT DISTINCT c.name as cont, p.PName as artPiece
                FROM Scores s, Pieces p, Contestants c, ShowJ sj, ShowDet dt
                WHERE p.Pid = s.Piece
                AND
                    s.Contestant = c.id
                AND
                    s.sjId = sj.sjId
                AND
                    sj.showId = dt.Sid
                ORDER BY cont
            ) r2
        where r2.cont = record2.cont)
    ) r)
>=
(SELECT count(*) as c_r2
    FROM (SELECT DISTINCT artPiece from (SELECT DISTINCT c.name as cont, p.PName as artPiece
        FROM Scores s, Pieces p, Contestants c, ShowJ sj, ShowDet dt
        WHERE p.Pid = s.Piece
        AND
            s.Contestant = c.id
        AND
            s.sjId = sj.sjId
        AND
            sj.showId = dt.Sid
        ORDER BY cont
    ) r2
where r2.cont = record2.cont) r)
)
ORDER BY record1.cont



/*Query5 - Find all chained co-auditions. A chained co-auditions is the transitive closure of
the following
binary relation: X and Y (directly) co-auditioned iff they both performed the same piece in the
same show and got the same score from at least one (same) judge. Thus, a chained co-audition
```

```
can be either a direct or an indirect co-audition.*/

create or replace recursive view IndirectChain(Cont1,Cont2) As
        Select distinct C1.name , C2.name
            from Scores Sc, Scores Sc1 , Contestants C1, Contestants C2
            where Sc.Contestant > Sc1.Contestant
                and
                Sc.Piece = Sc1.Piece and Sc.marks = Sc1.marks
                and
                (select judgeId from ShowJ SJ where SJ.sjId = Sc.sjId ) = (select judgeId from
                ShowJ SJ where SJ.sjId = Sc1.sjId)
                and
                (select showId from ShowJ SJ where SJ.sjId = Sc.sjId ) = (select showId from
                ShowJ SJ where SJ.sjId = Sc1.sjId)
                and
                Sc.Contestant = C1.id and Sc1.Contestant = C2.id
Union
        Select D.C1name, I.Cont2
            from (Select distinct C1.name as C1name, C2.name as C2name
                    from Scores Sc, Scores Sc1 , Contestants C1, Contestants C2
                    where Sc.Contestant > Sc1.Contestant
                        and
                          Sc.Piece = Sc1.Piece
                        and
                          Sc.marks = Sc1.marks
                        and
                        (select judgeId from ShowJ SJ where SJ.sjId = Sc.sjId ) = (select
                        judgeId from ShowJ SJ where SJ.sjId = Sc1.sjId)
                        and
                        select showId from ShowJ SJ where SJ.sjId = Sc.sjId ) = (select showId
                        from ShowJ SJ where SJ.sjId = Sc1.sjId)
                        and
                        Sc.Contestant = C1.id
                        and
                        Sc1.Contestant = C2.id
            ) D, IndirectChain I
            where D.C2name = I.Cont1

Select * from IndirectChain;
```


id
name
person
IsA
contestants
Pid
Pname
Pieces
N
Marked
N
Marks
received
M
M
contestant
Scores
M
Marks
given by
N
ShowJ
marks
piece
sjId
Sid
Sname
Sname
ShowDet
N
Judged By
sjId
judgeId
M
showId
M
Judged in
N
judges