6.830
Database Systems
2007/09/12

Dan Ports
drkp@mit.edu

11/10

Problem 1

```
SELECT title FROM movies
WHERE movies.title ~* '.*godmother.*'
AND movies.title !~* '.*fairy.*';
```

title
Godmother
Godmother
Godmother 2, The
Godmother's Present
Godmother, The
Godmother, The
Godmother, The
Godmother, The
Godmothers, The
Godmothers, The
Scary Godmother Halloween Spooktakular
Scary Godmother: The Revenge of Jimmy

(12 rows)

6.830 **Dan Ports**

Database Systems drkp@mit.edu

2007/09/12

Problem 2

```
SELECT movies.title, directors.first, directors.last
FROM movies, directors, directed
WHERE directed.movieid = movies.id
AND directed.directorid = directors.id
AND movies.title ~* '.*godmother.*'
AND movies.title !~* '.*fairy.*';
```

title	first	last
Godmother	Vinay	Shukla
Godmother 2, The	John	Stagliano
Godmother's Present	Ekaterina	Mikhaylova
Godmother, The	Ralph	Ince
Godmother, The	John	Stagliano
Godmother, The	George	Kuchar
Godmothers, The	William (I)	Watson
Godmothers, The	William	Grefe
Scary Godmother Halloween Spooktakular	Ezekiel	Norton
Scary Godmother: The Revenge of Jimmy	Ezekiel	Norton

(10 rows)

Problem 3

```
SELECT actors.first, actors.last
FROM actors, acted, movies
WHERE actors.id = acted.actorid
AND acted.movieid = movies.id
AND ...
```

Problem 4

```
SELECT movies.year, max(castsizes.castsize)
FROM movies,
     (SELECT movies.id AS movieid,
             count(acted.actorid) AS castsize
      FROM movies, ratings, acted
      WHERE movies.type = 0
      AND movies.id = ratings.id
      AND ratings.votes > 100000
      AND acted.movieid = movies.id
      GROUP BY movies.id) AS castsizes
WHERE movies.id = castsizes.movieid
GROUP BY movies.year
ORDER BY movies.year DESC;
```

year	max
2006	95
2005	137
2004	19
2003	81
2002	122
2001	78
2000	40
1999	125
1998	89
1997	136
1996	40
1995	80
1994	146
1993	146
1992	25
1991	62
1990	148
1988	62
1986	30
1985	49
1983	138
1982	29
1981	42
1980	61
1979	56
1977	105
1975	42
⋮	

1974	81
1972	63
1971	47
1964	19
1942	81
(32 rows)	

6.830 **Dan Ports**

Database Systems drkp@mit.edu

2007/09/12

Problem 5

```
SELECT (sum(castsizes.castsize * ratings.score) -
        sum(castsizes.castsize) * sum(ratings.score) / count(*)) /
       sqrt((sum(castsizes.castsize^2) -
            (sum(castsizes.castsize))^2/count(*)) *
           (sum(ratings.score^2) -
            (sum(ratings.score))^2/count(*))) AS correlation
FROM movies, ratings,
     (SELECT movies.id AS movieid,
             count(acted.actorid) AS castsize
      FROM movies, ratings, acted
      WHERE movies.id = ratings.id
      AND ratings.votes > 100000
      AND acted.movieid = movies.id
      GROUP BY movies.id) AS castsizes
WHERE movies.id = castsizes.movieid
AND movies.id = ratings.id;
```

```
    correlation
-------------------
 -0.270535198964407
(1 row)
```

Dan Ports
drkp@mit.edu

Problem 6

```
CREATE TEMPORARY TABLE average_yearly_score
AS SELECT actors.id AS actorid,
          movies.year AS year,
          avg(ratings.score) AS score
FROM actors, acted, movies, ratings
WHERE movies.id = acted.movieid
AND actors.id = acted.actorid
AND ratings.id = movies.id
AND ratings.votes >= 10000
AND movies.type = 0
GROUP BY actors.id, movies.year;

SELECT actors.first, actors.last, a1.year, a1.score, a2.year, a2.score
FROM actors, average_yearly_score a1, average_yearly_score a2
WHERE a1.actorid = a2.actorid
AND a1.year = a2.year + 1
AND a1.score <= a2.score - 4
AND actors.id = a1.actorid
ORDER BY (a2.score - a1.score) DESC;

DROP TABLE average_yearly_score;
```

first	last	year	score	year	score
Jim	Piddock	2007	2.3	2006	8.4
Barry	Pepper	2000	2.3	1999	8.2
Kevin (I)	Allen	1997	2.8	1996	8.1
Michael (I)	Caine	1987	2.5	1986	7.8
Forest	Whitaker	2000	2.3	1999	7.4
Jayma	Mays	2007	2.3	2006	7.3
Claire	Rushbrook	1997	2.8	1996	7.8
Thora	Birch	2000	3.6	1999	8.5
Stephen	Tobolowsky	2001	3.7	2000	8.6
Richard	Briers	1997	2.8	1996	7.6
Perdita	Weeks	1997	2.8	1996	7.6
Peter	Wingfield	2004	3.2	2003	7.9
Kal	Penn	2007	2.3	2006	6.9
Dant	McCarthy	1995	3.8	1994	8.4
Anwar	Burton	2007	2.3	2006	6.7
Marc John	Jefferies	2005	3.4	2004	7.8
Lainie	Kazan	2003	2.3	2002	6.7
Becca	Sweitzer	2003	1.8	2002	6.2
David Jean	Thomas	2000	4.3	1999	8.7
⋮					

Kimberly	Flynn	1996	3.7	1995	8.1
Allison	Kyler	2006	2.7	2005	7.1
Diego	Diablo Del Mar	2004	3.2	2003	7.5
Lea (I)	Thompson	1986	3.9	1985	8.2
Clint	Howard	2003	2	2002	6.2
Gregory	Jbara	2007	2.3	2006	6.5
Pete	Antico	1991	3.4	1990	7.6
Mark	Acheson	2005	2.2	2004	6.3
Frank	Welker	1993	3.6	1992	7.7
Thandie	Newton	2007	3.6	2006	7.7
Dan (I)	Duran	2005	3.4	2004	7.4
Christian	Slater	2005	2.2	2004	6.2
Alex	Borstein	2004	3.2	2003	7.2
Mary Ellen	Trainor	1995	4.4	1994	8.4
(33 rows)					

Problem 7

```
(SELECT count(*)
FROM
      (SELECT DISTINCT acted1.actorid                    -- (*)
       FROM acted acted1, acted acted2, movies           -- (*)
       WHERE acted1.movieid = movies.id                  -- (*)
       AND acted2.movieid = movies.id                    -- (*)
       AND acted2.actorid                                -- (*)
       IN                                                -- (*)
           (SELECT DISTINCT acted1.actorid
           FROM acted acted1, acted acted2, movies
           WHERE acted1.movieid = movies.id
           AND acted2.movieid = movies.id
           AND acted2.actorid
           IN (SELECT DISTINCT id
               FROM actors
               WHERE actors.first = 'Kevin' AND actors.last =
               'Bacon')))
AS actorids);
```

count
427771
(1 row)

To generalize to a higher Bacon number, we can simply repeatedly apply the subquery to the list of actors with Bacon number 2, by repeating the lines marked with a (*).

Problem 8

```
SELECT directors.first, directors.last, t.topcount
FROM directors,
     (SELECT directorid, count(*) AS topcount
      FROM directed
      WHERE movieid IN
          (SELECT m1.id
          FROM movies m1, movies m2, ratings r1, ratings r2
          WHERE m1.year = m2.year
          AND m1.type = 0 AND m2.type = 0
          AND r1.id = m1.id and r2.id = m2.id
          AND r1.votes >= 10000 AND r2.votes >= 10000
          AND (r1.score < r2.score
               OR (r1.score = r2.score AND m1.id >= m2.id))
          GROUP BY m1.id, m1.year
          HAVING count(*) <= 10)
      GROUP BY directorid
      HAVING count(*) >= 4) AS t
WHERE t.directorid=directors.id
ORDER by t.topcount DESC;
```

You need an outer join because this will not include the top-rated movie

first	last	topcount
Alfred (I)	Hitchcock	14
Stanley	Kubrick	10
Steven (I)	Spielberg	10
Martin	Scorsese	7
Quentin	Tarantino	6
Rob	Reiner	6
Joel	Coen	6
Ethan	Coen	6
Terry	Gilliam	6
Francis Ford	Coppola	5
Billy	Wilder	5
Richard	Donner	5
Clint	Eastwood	5
David (I)	Lynch	5
Hayao	Miyazaki	5
Sergio (I)	Leone	5
Peter (I)	Jackson	5
Charles	Chaplin	4
Akira	Kurosawa	4
Tim (I)	Burton	4
Sidney	Lumet	4

⋮

John (I)	Ford	4
Brian	De Palma	4
Woody	Allen	4
John (I)	Landis	4
Frank	Capra	4
Howard	Hawks	4
Ridley	Scott	4
Hamilton	Luske	4
John (I)	Huston	4
Guy (I)	Hamilton	4
(31 rows)		