Problem Set 1 Solutions

1 Questions

Questions 1-3 and 5 are worth one point. Questions 4 and 6-8 are worth two points.

1. Write a query (using the `SELECT` statement) that will find the complete names of all papers that contain the string "Beer". You may find the "˜" operator useful. Show both the query and the result.

```
SELECT name
FROM papers
WHERE name˜'Beer';

                                    name
------------------------------------------------------------------------------
 Champagne Training on a Beer Budget.
 Fireworks, Beer and Old Halfpennies - The Risks of Assumption.
 Beers Modell Lebensf\uffff\uffffhiger Systeme als Instrument der
       Wissenskoordination in ERP-Projekten in Mittelbetrieben.
 Computers Play the Beer Game - Can Artificial Agents Manage Supply Chains?
 Predicting the Speed of Beer Fermentation in Laboratory and Industrial Scale.
(5 rows)
```

2. Write a query that finds the venues in which those papers were published. You will need to write a "join query". Show the query and the result.

```
SELECT venue.name
FROM papers,venue
WHERE  papers.name˜'Beer'
AND  papers.venue = venue.id;

        name
--------------------
 Commun. ACM
 Computers&Security
 Wissensmanagement
 HICSS
 IWANN (2)
(5 rows)
```

In SQL, a subquery is a query over the results of another query. You can use subqueries in the `SELECT` list, the `FROM` list, or as a part of the `WHERE` clause. For example, suppose we want to find the name of the paper with the smallest id. To find the smallest id, we would write the query `SELECT min(id) FROM papers`, but SQL doesn't provide a way to get the other attributes of that minimum-id paper without using a nested query. We can do this either with nesting in the `FROM` list or in the `WHERE` clause. Using the nesting `FROM` list, we would write:

```
SELECT name
FROM papers,
     (SELECT min(id) AS minid
      FROM papers) AS nested
WHERE papers.id =nested.minid;
```

Using nesting in the `WHERE` clause, we would write:

```
SELECT name
FROM papers
WHERE papers.id = (SELECT min(id) FROM papers);
```

Hence, there are usually several possible ways to write a given query, and some of those ways may provide different performance (despite the best efforts of database system designers to build optimizers that yield query performance that is independent of the query's formulation.)

Note that if you were interested in finding papers that matched a list of ids, you could replace the "=" sign in the query above with the `IN` keyword; e.g.:

(1)

```
SELECT name
FROM papers
WHERE papers.venue IN (SELECT id  FROM venue WHERE ... );
```

It is usually the case that when confronted with a subquery of this form, it is possible to un-nest the query by rewriting it as a join.

3. Show what the `SELECT ...  WHERE ...  IN` query labeled (1) above would look like when re-written as a join.

```
SELECT name
FROM papers, venue
WHERE papers.venue = venue.id
AND venue. ...
```

4. Write a query that finds the total number of papers published in each of the conferences identified in Problem 2. Try to avoid actually typing the names of the conferences in your query.

   It is possible to answer this question using either a join or a subquery. Show both versions, and the results of running the query.

   Here, the subquery finds the id's of the venues that published papers with beer in the title, and the outer query finds the names of those conferences and the number of papers they published. Note that some of you adding a clause that verified that the venue was a conference; this is perhaps a more strict interpretation of the question but either approach was given full credit. We show the version without the test for conference-dom here.

```
SELECT venue.name,count(*)
FROM papers,venue
WHERE papers.venue IN (
        SELECT venue.id
        FROM papers,venue
        WHERE papers.name~'Beer'
        AND papers.venue = venue.id
)
AND papers.id=venue.id
GROUP BY venue.name;
```

   In this version, we filter the papers table twice: once to find the papers about beer and once to find the papers in conferences that published papers about beer.

```
SELECT venue.name,count(*)
FROM papers AS p1, papers AS p2,venue
WHERE p1.name~'Beer'
AND p1.venue = p2.venue
AND p2.venue = venue.id
GROUP BY venue.name;

        name        | count
--------------------+-------
 HICSS              |   528
 Commun. ACM        |    11
 Wissensmanagement  |   110
 IWANN (2)          |    92
 Computers&Security |    27
(5 rows)
```

5. Write a query that finds the authors of all of the papers whose titles contain the phrase "Beer". Show your query and the results.

   A simple three-way join.

```
SELECT authors.name
FROM authors, papers, paperauths
WHERE paperauths.paperid = papers.id
AND paperauths.authid = authors.id
AND papers.name~'Beer';

         name
----------------------
 Fang Zhong
 Tapio Elomaa
 Robert J. Aarts
 Juho Rousu
 Dong-Jun Wu
 Aileen Cater-Steel
 Bernd St\uffff\uffffckert
 Dirk Kahlert
 Edmond P. Fitzgerald
 Steven O. Kimbrough
 Stephen Hinde
```

6. In the database community, the two most important conferences are SIGMOD and VLDB. Write a query that finds the top 10 most published authors in these two conferences (you should search for the strings 'SIGMOD Conference' and 'VLDB'; don't use the '~' operator – instead, use the '=' operator). Compute a single top 10 list ordered by the sum of the papers published in either conference. Show your query and the top 10 results.

   A three way join with an aggregate and a group by. The "LIMIT 10" command provides the top 10 results.

```
SELECT a.name, count(*)
FROM authors AS a, papers AS p, paperauths AS pa, venue AS v
WHERE (v.name='VLDB' OR v.name='SIGMOD Conference')
AND p.venue=v.id
AND pa.paperid=p.id
AND  pa.authid=a.id
GROUP BY a.name
ORDER BY count(*) DESC
LIMIT 10;

            name            | count
----------------------------+-------
 David J. DeWitt            |    60
 Michael J. Carey           |    58
 Hector Garcia-Molina       |    52
 Michael Stonebraker        |    50
 H. V. Jagadish             |    50
 Rakesh Agrawal             |    49
 Jeffrey F. Naughton        |    48
 Surajit Chaudhuri          |    48
 Divesh Srivastava          |    46
 Raghu Ramakrishnan         |    44
```

7. The previous query shows people with the highest publication count, but doesn't normalize for the fact that some of these authors are really OLD. Arguably, a more interesting metric would be to divide the number of publications by the number of years since the person published his or her first paper (not in SIGMOD or VLDB, but in any conference.) Write a query that computes the new top 10 list according to this metric, and show your query and your top ten list (as well as the score associated with each person in the list). Note that this is a long query that will take quite a while to run...

   Here, we use a nested query in the SELECT list to find the number of years since a particular author published his or her first paper, and divide the total paper count by this value to obtain a score for each author. We then order the results by this score and select the top ten highest scoring authors using the ORDER BY and LIMIT clauses.

```
SELECT a.name,a.id, count(*), count(*)::float/ (
        SELECT 2006-min(year)::float
        FROM papers AS p2, venue AS v2, paperauths AS pa2, authors AS a2
        WHERE pa2.authid=a.id
        AND a2.name=a.name
        AND pa2.paperid=p2.id
        AND p2.venue=v2.id) AS score
FROM authors AS a, papers AS p, paperauths AS pa, venue AS v
WHERE (v.name='VLDB' OR v.name='SIGMOD Conference')
AND p.venue=v.id
AND pa.paperid=p.id
AND pa.authid=a.id
GROUP BY a.name,a.id
ORDER BY score DESC
LIMIT 10;
```

```
           name            |   id   | count |        score
---------------------------+--------+-------+---------------------
 Divesh Srivastava         |  11692 |    46 |               2.875
 Alon Y. Halevy            |  19469 |    16 |    2.66666666666667
 Yufei Tao                 | 101595 |    13 |                 2.6
 Surajit Chaudhuri         |  15715 |    48 |    2.52631578947368
 H. V. Jagadish            |  15711 |    50 |                 2.5
 Nick Koudas               |  17036 |    25 |                 2.5
 Michael J. Carey          |  15716 |    58 |    2.41666666666667
 Jeffrey F. Naughton       |  46633 |    48 |                 2.4
 Joseph M. Hellerstein     | 100848 |    33 |    2.35714285714286
 Jian Pei                  | 115992 |    14 |    2.33333333333333
```

8. Write a query that finds people who have published a paper in SIGMOD for each of the past five years (e.g., in SIGMOD 2001 - 2005). See if you can do this without simply combining the results of five different queries (one for each year.) Show your query and the list of names.

   The trick here is to realize that grouping the list of recent papers published in SIGMOD by author name and year of publication will produce a list that has one entry for each year that a given author published in SIGMOD. Hence, if a given author appears five times in this aggregated list, he or she must have published a paper for each of the five previous years.

   There are other ways to compute this answer, but this is by far the most succinct and efficient.

```
SELECT name
FROM (SELECT a.name AS name, v.year AS year, count(*)
      FROM authors AS a, papers AS p, paperauths AS pa, venue AS v
      WHERE v.name='SIGMOD Conference'
      AND v.year>2000
      AND a.id = pa.authid
      AND p.id = pa.paperid
      AND v.id = p.venue
      GROUP BY a.name, v.year ) AS rec_papers
GROUP BY name
HAVING count(*) = 5;

         name          | count
----------------------+-------
 Surajit Chaudhuri    |     5
 Peter J. Haas        |     5
 Divesh Srivastava    |     5
 Minos N. Garofalakis |     5
 Per-Ake Larson       |     5
 Philip Bohannon      |     5
 Dennis Shasha        |     5
 H. V. Jagadish       |     5
 Johannes Gehrke      |     5
 Dimitris Papadias    |     5
 Philip S. Yu         |     5
 Guy M. Lohman        |     5
(12 rows)
```