Information and Database Management Systems I (CIS 4301)  
(Fall 2016)  
Instructor: Dr. Markus Schneider  
TA: Yang Chen  

Exam 2 Part 2 Solutions

<table>
<thead>
<tr>
<th>Name:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>UFID:</td>
<td></td>
</tr>
<tr>
<td>Email Address:</td>
<td></td>
</tr>
</tbody>
</table>

Pledge (Must be signed according to UF Honor Code)

On my honor, I have neither given nor received unauthorized aid in doing this assignment.

_______________________________________________  
Signature

For scoring use only:

<table>
<thead>
<tr>
<th>Question 3</th>
<th>Maximum</th>
<th>Received</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>50</td>
<td></td>
</tr>
</tbody>
</table>
**Question 3 (SQL) [50 points]**

Consider the Yelp reviews collected from multiple cities around the world. The schema is given below.

Write SQL queries for each of the following questions.

1. Find every business in Florida that has the word “Coffee” in its name but is not classified as a “Coffee” place (i.e. not in the “Coffee” category). List the business’s id and name in ascending bid order. [5 points]

   ```sql
   SELECT bid, name
   FROM Business
   WHERE name LIKE '%Coffee%' AND state = 'FL'
   AND bid NOT IN (
      SELECT bid
      FROM BusinessCategory
      WHERE category = 'Coffee')
   ORDER BY bid ASC;
   ```

2. We define the “usefulness count” of a user as the sum of the “useful” votes his reviews attract. Find the user with the highest such count. Print the user's uid, name, and usefulness count. [5 points]

   ```sql
   SELECT uid, name, cnt AS usefulness
   FROM (SELECT U.uid, U.name, SUM(r.votes_useful) AS cnt
      FROM User U, Review R
      WHERE U.uid = R.uid
      GROUP BY U.uid, U.name) V
   WHERE cnt = (SELECT MAX(cnt) FROM V)
   ORDER BY uid ASC;
   ```
3. Find Florida businesses that have more than 10 reviews and all of them are 5 stars. For each business, print bid, name and number of reviews. [10 points]

```sql
SELECT B.bid, B.name, COUNT(*) AS numReviews
FROM Business B, Review R
WHERE B.bid = R.bid
AND B.bid NOT IN (
    SELECT B.bid
    FROM Business B1, Review R1
    WHERE B1.bid = R1.bid AND B1.state = 'FL'
    AND R1.stars < 5)
AND B.state = 'FL'
GROUP BY B.bid, B.name
HAVING COUNT(*) > 10;
```

4. Consider the reviews by frequent travelers (users who have been to more than 5 states). We define the score as the average number of stars received from frequent travelers. List the top 5 “Burgers” category restaurants in Gainesville, Florida. For each restaurant, print its bid, name, score, and number of frequent traveler reviews received. [10 points]

```sql
CREATE VIEW Traveler AS
SELECT DISTINCT U.uid
FROM User U, Review R, Business B
GROUP BY U.uid
HAVING COUNT(DISTINCT B.state) > 5;

SELECT * FROM (  
    SELECT B.bid, B.name, AVG(R.stars) AS score, COUNT(*) AS numReviews
    FROM Traveler T, Review R, Business B, BusinessCategory BC
    AND B.city = 'Gainesville' AND B.state = 'FL'
    AND BC.category = 'Burgers'
    GROUP BY B.bid, B.name
    ORDER BY score DESC)
WHERE ROWNUM <= 5;
```

5. Find the businesses whose average rating was raised by more than one star from May 2011 to June 2011. For each business, print its bid, name, and change of average rating. Order your results by the change (largest first).

Please use `EXTRACT(MONTH FROM date)` and `EXTRACT(YEAR FROM date)` to get the month and year from a date. [10 points]

```sql
CREATE VIEW MayStar AS
SELECT bid, AVG(stars) AS avgStars
FROM Review
WHERE EXTRACT(MONTH FROM date) = 5 AND EXTRACT(YEAR FROM date) = 2011
GROUP BY bid;
```
CREATE VIEW JuneStar AS
SELECT bid, AVG(stars) AS avgStars
FROM   Review
WHERE  EXTRACT(MONTH FROM date) = 6 AND EXTRACT(YEAR FROM date) = 2011
GROUP  BY bid;

SELECT B.bid, B.name, (J.avgStars - M.avgStars) AS change
FROM   Business B, MayStar M, JuneStar J
AND   J.avgStars - M.avgStars > 1
ORDER  BY change DESC;

6. Amongst users with at least 50 reviews, find the 2 “most similar” users, where the similarity of
two users is defined as the fraction of shared businesses they have reviewed. Return the users' id
and the similarity.

Specifically, if A and B are the sets of businesses the two users have reviewed, the similarity is
|A∩B| / |A∪B|. [10 points]

[Hint: you may use the equation |A∪B| = |A| + |B| - |A∩B|.

CREATE VIEW U AS
SELECT uid, COUNT(*) AS cnt
FROM   Review
GROUP  BY uid
HAVING COUNT(*) >= 50;

CREATE VIEW SharedReviews AS
SELECT R1.uid AS u1, R2.uid AS u2, COUNT(*) AS count
FROM   Review R1, Review R2
WHERE  R1.bid = R2.bid
GROUP  BY R1.uid, R2.uid;

SELECT u1, u2,
    1.0 * R.count / (U1.cnt + U2.cnt - R.count) AS similarity
FROM   SharedReviews R, U U1, U U2
WHERE  R.u1 = U1.uid AND R.u2 = U2.uid
AND    U1.cnt >= 50 AND U2.cnt >= 50
ORDER  BY similarity DESC
LIMIT 2;