Coms W4111.001–Introduction to Databases
Spring 2016
Homework Assignment 2
Solutions

1. (1.5 pts.) Consider relation $T_1$, with attributes $A$, $B$, and $C$:

$$\begin{array}{ccc}
A & B & C \\
1 & a & 2 \\
2 & b & 2 \\
3 & c & 3 \\
\end{array}$$

and

relation $T_2$, with attributes $A$, $B$, and $D$:

$$\begin{array}{ccc}
A & B & D \\
1 & b & 8 \\
2 & b & 8 \\
2 & b & 9 \\
\end{array}$$

Show the results of the following operations:

(a) $\Pi_D(T_2) = \begin{array}{c} D \\
8 \\
9 \end{array}$

(b) $T_1 \bowtie_{T_1.C=T_2.A} T_2 = \begin{array}{cccc}
A & (B) & C & (B) & D \\
1 & a & 2 & b & 8 \\
1 & a & 2 & b & 9 \\
2 & b & 2 & b & 8 \\
2 & b & 2 & b & 9 \\
\end{array}$

(c) $T_1 \bowtie T_2 = \begin{array}{cccc}
A & B & C & D \\
2 & b & 2 & 8 \\
2 & b & 2 & 9 \\
\end{array}$

(d) $T_1 - T_2 = \begin{array}{cc}
A & B \\
1 & a \\
2 & b \\
3 & c \\
\end{array}$

(e) $(T_1 \bowtie T_2) \bowtie (T_2 \bowtie T_2) = \begin{array}{cccc}
A & B & C & D \\
2 & b & 2 & 8 \\
2 & b & 2 & 9 \\
\end{array}$
2. Consider the following relational database schema:

- **Movies**(title, year, length, studioName)
  
  A tuple \((t, y, \ell, s)\) indicates that a movie with title \(t\) was produced by studio \(s\) in year \(y\), and the length (in minutes) of the movie was \(\ell\).

- **Stars**(starName, title, year)
  
  A tuple \((s, t, y)\) indicates that the star (i.e., actor or actress) with name \(s\) acted in a movie with title \(t\) in year \(y\).

Express each of the following queries in relational algebra (not in SQL):

(a) **(1 pt.)** Find the title and year of every pair of movies that have at least one star in common.

\[
\rho(T_1, Stars) \\
\rho(T_2, Stars) \\
\Pi_{T_1.title, T_1.year, T_2.title, T_2.year}(T_1 \bowtie(T_1.title \neq T_2.title \lor T_1.year \neq T_2.year \land T_1.starName = T_2.starName) T_2)
\]

(b) **(1.25 pts.)** Find the name of each star who never acted in more than one movie in any single year.

\[
\rho(T_1(2 \rightarrow title1, 3 \rightarrow year1, 4 \rightarrow title2, 5 \rightarrow year2), Stars \bowtie_{starName} Stars) \\
\Pi_{starName}(Stars) - \Pi_{starName}(\sigma_{year1 = year2 \land title1 \neq title2}(T_1))
\]

(c) **(1.25 pts.)** Find the name of each star who acted in every movie produced by the “Paramount” studio between 2000 and 2015.

\[
Stars / \Pi_{title, year}(\sigma_{year \geq 2000 \land year \leq 2015 \land studioName='Paramount'}(Movies))
\]

3. This problem uses a PostgreSQL database with data about a university. The schema of the database is provided below. Keys are underlined, field types are omitted; assume that appropriate foreign key constraints (for example, from prof into dept) are present in the full version of the schema:

- **student**(sid, sname, sex, age, year, gpa)
- **dept**(dname, numphds)
- **prof**(pname, dname)
- **course**(cno, cname, dname)
- **major**(dname, sid)
- **section**(dname, cno, sectno, pname)
- **enroll**(sid, grade, dname, cno, sectno)

We ask that you write exactly one SQL query for each question below.
1. Print the name and department of each professor who has taught a course not from his/her department at least once.

```
SELECT DISTINCT P.pname, P.dname
FROM section S, prof P
WHERE S.pname=P.pname AND S.dname!=P.dname
```

<table>
<thead>
<tr>
<th>name</th>
<th>department</th>
</tr>
</thead>
<tbody>
<tr>
<td>Smith, S.</td>
<td>Industrial Engineering</td>
</tr>
<tr>
<td>Clark, E.</td>
<td>Civil Engineering</td>
</tr>
</tbody>
</table>

2. Print the sid and name of each student who has earned a 3.5 or higher grade in at least two different courses.

```
SELECT DISTINCT E1.sid, S.sname
FROM enroll E1, enroll E2, student S
WHERE E1.sid=E2.sid AND S.sid=E1.sid AND
  E1.grade>=3.5 AND E2.grade>=3.5 AND
  (E1.dname!=E2.dname OR E1.cno!=E2.cno OR E1.sectno!=E2.sectno)
```

<table>
<thead>
<tr>
<th>sid</th>
<th>name</th>
</tr>
</thead>
<tbody>
<tr>
<td>49</td>
<td>Villa-lobos, M.</td>
</tr>
<tr>
<td>90</td>
<td>Zappa, F.</td>
</tr>
<tr>
<td>67</td>
<td>Altenhaus, Gloria</td>
</tr>
<tr>
<td>64</td>
<td>Fred, Edwin B.</td>
</tr>
</tbody>
</table>

3. Print the name and age of the student(s) with the highest GPA in their exact age group (i.e., the name and age of the student(s) with the highest GPA among all 15 year olds, the name and age of the student(s) with the highest GPA among all 16 year olds, and so on), for ages less than or equal to 18.

```
SELECT S1.sname, S1.age
FROM student S1
WHERE S1.age<=18 AND S1.gpa=(SELECT MAX(S2.gpa)
                        FROM student S2
                        WHERE S2.age=S1.age)
```

<table>
<thead>
<tr>
<th>name</th>
<th>age</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baker, C.</td>
<td>18</td>
</tr>
<tr>
<td>News, Nightly</td>
<td>15</td>
</tr>
<tr>
<td>Zappa, F.</td>
<td>16</td>
</tr>
<tr>
<td>Morgan, D.</td>
<td>18</td>
</tr>
</tbody>
</table>

4. For each department that both (a) has the substring "Engineering" in its name (e.g., "Electrical Engineering") and (b) has at least 2 students majoring in the department, print the name of the department and the average GPA of the students who major in the department.

```
SELECT M.dname, AVG(S.gpa)
FROM major M, student S
WHERE M.dname LIKE '%Engineering%' AND M.sid=S.sid
GROUP BY M.dname
HAVING COUNT(*)>=2
```
5. Some courses are popular among students just because students enrolled in those courses usually get good grades. Print the department name, course number, and course enrollment of each course C such that the following two conditions hold: (1) course C's enrollment is at least 3% larger than the average enrollment of the courses offered by C's department, and (2) course C's average grade is at least 3% larger than the average grade obtained by students in the courses offered by C's department. Assume that the enrollment of a course is the sum of the enrollment of all its sections. **You can completely ignore any course that has no students enrolled in it.**

```
SELECT E.dname, E.cno, COUNT(*) as enrollment
FROM enroll E
GROUP BY E.dname, E.cno
HAVING COUNT(*)>=1.03*(SELECT COUNT(*)/COUNT(DISTINCT E1.cno)
FROM enroll E1
WHERE E1.dname = E.dname) AND
AVG(E.grade)>=1.03*(SELECT AVG(E2.grade)
FROM enroll E2
WHERE E2.dname = E.dname)
```