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) $\Pi_D(T_2)$

(b) $T_1 \bowtie_{T_1.C=T_2.A} T_2$

(c) $T_1 \bowtie T_2$

(d) $T_1 - T_2$ (Assume that $T_1$ and $T_2$ are union-compatible.)

(e) $(T_1 \bowtie T_2) \bowtie (T_2 \bowtie T_2)$

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.

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

(c) (1.25 pts.) Find the name of each star who acted in every movie produced by the “Paramount” studio between 2000 and 2017.
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. Run your SQL queries on one of our PostgreSQL servers (choose any one, they are identical) using the web interface that we have set up at http://w4111p1.cs.columbia.edu/ or http://w4111p2.cs.columbia.edu/. The SQL interpreter in PostgreSQL is not identical to the one described in the textbook. If the query you write is not accepted by the server, try different equivalent ways until you get one that works with PostgreSQL, see http://www.postgresql.org/docs/9.3/interactive/index.html. We do not care if your answer contains duplicate rows or not. The answer to each question is worth 1 point. For each question, you should submit both (a) the SQL query and (b) the results of the execution of the query on one of the PostgreSQL servers above, which you should cut and paste from the web interface.

(a) Print the name and department of each professor who has taught a course not from his/her department at least once.

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

(c) 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.

(d) 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.

(e) 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.