Virginia Tech. **CS 4604 – Introduction to DBMS**
Computer Science **Spring 2013, Prakash**

Homework 1: Relational Algebra and SQL
(due February 8th, 2013, 9:05am, in class—hard-copy please)

Reminders:

a. Out of 100 points.
b. Rough time-estimates: 2~4 hours.
c. Please type your answers. Illegible handwriting may get no points, at the discretion of the grader. Only drawings may be hand-drawn, as long as they are neat and legible.
d. There could be more than one correct answer. We shall accept them all.
e. Whenever you are making an assumption, please state it clearly.

Q1: Movie Ratings [50 points]

This question is on a movie ratings database. Download and install SQLite3 from http://www.sqlite.org

Warm-up

Follow the documentation and load the sample database at:
http://courses.cs.vt.edu/~cs4604/Spring13/homeworks/hw1/cs4604-hw1.db

It has a table recommendation which is like the table of the *Netflix* competition: people rate movies, with ratings from 1-5 (1 for 'I hate it!', to 5, for 'I love it!'). As a sanity check that you have the correct database, running the following command at a Unix/Linux/Cygwin prompt-

```
your-machine% sqlite3 cs4604-hw1.db 'select count(*) from recommendation'
```

should return

```
14
```

We want to write SQL queries to do the following:

- Query1: Return all movies with a rating of 5 from at least one reviewer.
- Query2: Return all the reviewers who rated 'Fight Club'.

Large CSV file

A bigger raw comma separated value (csv) file is given here:
http://courses.cs.vt.edu/~cs4604/Spring13/homeworks/hw1/movie_ratings.csv

It is a subset from the Netflix-competition dataset (If you are curious, the official Netflix USD 1 Million prize dataset is at http://www.netflixprize.com//index). The dataset is anonymized, hence customers will be represented by a random, integer id. We want to write queries to do the following:

- Query3: Return the count of reviews where the rating is 5.
- Query4: Return the count of reviewers who gave a rating of 1 to 'Gone with the wind'.

Life without SQL

Finally, in your favorite language (Python/Perl/Ruby/Java/C++ etc.) write code to do both queries above (Query3 and Query4) on the csv data file directly. Notice: the end-ofline convention is the DOS one (CR LF).

Deliverables

Q1.1. (5 points) The SQL query for Query1.

Q1.2. (5 points) The SQL query for Query2.

Q1.3. (5 points) The output of running Query1 in SQLite on the sample database.

Q1.4. (5 points) The output of running Query2 in SQLite on the sample database.

Q1.5. (5 points) The SQL query for Query3.

Q1.6. (5 points) The SQL query for Query4.

Q1.7. (5 points) The output of running Query3 on the csv file after loading it in SQLite.

Q1.8. (5 points) The output of running Query4 on the csv file after loading it in SQLite.

Q1.9. (5 points) Hard copy of your python/perl/etc code for doing Query3 on the raw csv file directly.

Q1.10. (5 points) Hard copy of your python/perl/etc code for doing Query4 on the raw csv file directly.

Hints

For loading the csv file,

- Again, the end-of-line convention follows the DOS format (CR LF).
- Use the .import and .mode csv commands of sqlite3 or check the tutorial at http://my.opera.com/cookyjar/blog/2009/04/20/importing-csv-data-file
- Again as a sanity check, the command

```
your-machine% wc -l movie_ratings.csv
```

should return

```
10000 movie_ratings.csv
```

Q2: The DB-Pizza Store [50 points]

Consider the following relational design used in our friendly local DB-Pizza store in Blacksburg:

Customer (cid, name, phonenumber, ccn, neighborhood, age)
Pizza (pid, pname, size, price)
Order (cid, pid, ordertime, orderyear, ordermonth, orderday, quantity, slices)
Supplies (sname, amountleft, unitprice)
Ingredient (pid, sname, amount)

In the **Pizza** table, every pizza has a name (e.g., "the works") and particular size (e.g., 7 inches) and price and is assigned a unique pid. Note that different pizzas may have the same name, but different sizes.

The **Order** table includes the records about which customer ordered which pizza, quantity of pizzas, slices, orderyear (e.g., 2012), ordermonth (e.g., 12), orderday (e.g., 27), and the order time (e.g., "6:13pm"). Note that every customer can order one or more pizzas.

The **Customer** table maintains the personal information, such as a unique id, name, phone number, credit card number (ccn), neighborhood (e.g., Foxridge and Terrace View), and age. It is possible that two different people have the same name.

The **Supplies** table includes the information of the various groceries used by the store: the name, unit-price, and the amount-left in the store (e.g. the store might have 3kgs of mozzarella left with a unit price of 5$ per kg).

The **Ingredient** table keeps the records about the amount of ingredients used by each pizza (so "The works 7inch" might use only 10gms of mozzarella, while "Four cheese 12inch" might use 40gms).

Please answer the following questions (and write the queries in the notation indicated). **Use only the operators and SQL statements we have learnt in class. In addition, avoid the operators IN, ALL, ANY, and EXISTS.**

Q2.1. (5 points) What are good primary keys for the five tables?

Q2.2. (5 points) Relational Algebra: Find the ids of the customers with name "David".

Q2.3. (5 points) Relational Algebra: List pids of the pizzas ordered by all the customers.

Q2.4. (5 points) SQL: Find pids of all the pizzas with the price more than 15 dollars.

Q2.5. (10 points) Relational Algebra: List ids and names of the customers who ordered at least one pizza and live in Foxridge.

Q2.6. (10 points) SQL: List the ids and names of customers who spent the most on ordering pizzas during 2011. **Do not** use the MAX operator for this problem. *Hint: Get the set that cannot be the answer and then think negation.*

Q2.7. (10 points) SQL: Find all the pizzas (pid, pname, and size) where the store makes a profit of at least 10 dollars per whole pizza. (Here, we assume Profit = Pizza Price – Ingredient Cost)