Lab 1

The goal of this lab is to familiarize yourself with SQL statements. Please make sure to sanity check your work. Just because you get a response from SQL, does it make sense? Is it returning thousands of rows, when it should be returning only a handful?

Completing the assignment

Following set up below. Questions should be emailed to me with raw SQL statements in the body of the email. The subject of the email should be “Spring 2013: Lab 1”

Setup:

Set up your machine to use RVM (this assumes you have root access to your machine):

From the a shell in a terminal window, run:

```
$ \curl -L https://get.rvm.io | bash -s stable --ruby
```

And follow the instructions. (For more details, check here: https://rvm.io/rvm/install/)

Install the most recent ruby version (1.9.3, or later). (More info here: https://rvm.io/rvm/install/):

```
$ rvm install 1.9.3
```

RVM will now download and build ruby. This may take a few, go get a beverage or better yet, read up on Ruby.

When ruby has finished building, checkout out the SQLite file from git hub:

```
$ git clone https://github.com/BJK/funWithSQL.git
```

Change to the directory:

```
$ cd funWithSQL
```

Create an RVM gemset (and use it):

```
$ rvm use ruby-1.9.3@funwithsql --create
```

Install the SQLite3 gem:

```
$ gem install sqlite3
```

Run the sqlite3 console:

```
$ sqlite3 tweet_db
```

You are now in the SQL lite console.

Helpful Commands

```
sqlite> .quit
sqlite> .tables ## lists all the tables in the DB
sqlite> PRAGMA table_info(NAME_OF_TABLE) ## shows you columns in
table.
sqlite> select * from users ; ## returns all rows and all
columns from the users table.
```

Hints

For a shortcut, you can 'nickname' a table by specifying a name after the table name in the FROM clause. For example, if I want to nickname the users table as 'u', I can do it like so in the following query.

```
SELECT u.name, s.status FROM users u INNER JOIN statuses s ON
s.user_id = u.id ;
```

You can use the LIKE operator in a where clause to look for text patterns. The % character is used as a wild card.

```
WHERE name LIKE '%ST'
```

Will find rows whose name column starts with the characters "ST".

```
WHERE name LIKE '%ST%'
```

Will find rows whose name column has the characters "ST" anywhere within the phrase.

```
WHERE name = "STEVE JOBS"
```

Will find only the rows whose name column EXACLY matches "STEVE JOBS".

If you get stuck, google around, and make sure to checkout the SQLite command reference. http://www.sqlite.org/lang.html

Questions

Remember to turn in the SQL queries that return the answers, I am not actually interested in the answers themselves, but valid SQL.

1. Select all the user names from the users table.
2. Select all the user names whose name starts with the letter "b".
3. How many users are there?
4. How many posts by a certain user?
5. Return the user name column, and all the status by that user.

6. Return all users who have at least 1 status that has a geocode. (Hint, you'll want somewhere your WHERE clause "geocode IS NOT NULL").
7. Return all user names who have written more than one status.
8. Return all user names who have written more than one status with the tag "food."
9. Return user names, their statuses, and their tags in one big join.
10. What is the most recent status?
11. What is the most popular tag? (Hint, count how many statuses their are by tag).
12. What is the most popular user?
13. Give me ALL statuses by all users, with their tags, even the ones that don't necessarily have a tag.

Extra Credit

13. What is the most popular word? (you may need the help of ruby)
14. What are some other kinds of queries that are useful?