**The Problem**

SF Messaging is an aggregation company researching data to determine if they can improve internal processing. Each user has a registered email account (Users.txt files that contain a unique identifier, name, & email address respectively). The company also has gained access to a subset of info (mail.txt files that contain mail to: email address, mail from: email address, and timestamp respectively) from all emails sent on the internet. With such a huge dataset, you have been asked to analyze a random subset of the data from 2014 that has been chunked into separate files on 4 different server nodes located in src/test/resources. Your job is to implement the methods defined in the QueryService interface to obtain some high level stats about users and their email habits. Below are some examples:

List all users from a specific Domain. (Example: google)

List of emails in a specific date range.

Find the most popular hour within the day that emails are sent.

User “x” most active time, least active time, most active day, least active day.

User who emails themselves the most.

Most popular email address.

Most active sender, receiver.

Given a person, show ‘friends of friends’.

Sender who sends to the most number of people.

A *README* file is provided outlining the language parameters in addition to JUnits.  JUnit tests are the highest authority related to requirements and can be found by following the path *src/test/java*.

\*\*\*\*\*\*\*\*\* **Do not change anything in the JUnit tests!** \*\*\*\*\*\*\*\*\*\*

**First Actions:**

* Import the problem statement into your IDE.
* We have provided Maven dependency for JUnit 4. If you are not set up with the recommended IDE (Spring Tool Suite), you may need to add JUnit 4.
* Run your Junit tests, code, and repeat.
  + - For questions on how to run JUnits, please see document in Workspace: “How to run JUnits”

**When you are done:**

* Update the feedback.txt file and include the following information:
  + - Your team – name of each individual participating.
    - How many JUnits you were able to execute successfully.
    - Document and describe the additional “nice to have” features included, to help the judges properly grade your submission and explain how to properly execute new enhancements.
* Export your IDE project (The judges will need your entire project to run the JUnits) as a zip file and email zip file to [codingcompetition@statefarm.com](mailto:codingcompetition@statefarm.com) with subject line: State Farm Coding Competition – Name 1/Name 2

**Rules**

* Contestants cannot seek help from individuals outside their team.
* Teams are expected to have the necessary Tools and JARs preloaded on their machines **prior** to the competition.
* If you believe this document and the JUnit tests conflict, the JUnit tests are the highest authority.

**How you will be Graded**

Components of submission will be weighted as follows.

* 100% core requirements met, including:
  + Number of JUnits that pass using correct functionality in the program
  + Maintaining Object Oriented Programming principles
  + Code documentation
  + Code must compile and execute
* Do not complete any Bonus unless you have all the JUnit tests completed
  + Bonus credit awarded for any extra features added (up to 10%)

In the event of a tie, we will further judge your solution based on: code cleanliness, maintainability, and adherence to object-orientated principles.

Bonus “Nice to have” features:

* Bonus Credit – Do not complete any bonus features unless you have completed all the required functionality and all JUnits pass.
  + Create a Graphical User Interface (GUI), command line interface, or web UI that runs data queries from QueryService interface.
  + If you can think of any other useful features to add, we appreciate ingenuity and will gladly accept any useful enhancements. Be sure to document any bonus features in the feedback text file.