Welcome Raghavendra HM from Using Databases with Python   ✏   Exit

> Your current grade on this assignment is: 100%

To get credit for this assignment, perform the instructions below and upload your SQLite3 database here:

Choose File │ No file chosen

(Must have a .sqlite suffix)

Hint: The top organizational count is 536.

Submit

You do not need to export or convert the database - simply upload the **.sqlite** file that your program creates. See the example code for the use of the **connect()** statement.

Counting Organizations

This application will read the mailbox data (mbox.txt) count up the number email messages per organization (i.e. domain name of the email address) using a database with the following schema to maintain the counts.

```
CREATE TABLE Counts (org TEXT, count INTEGER)
```

When you have run the program on **mbox.txt** upload the resulting database file above for grading.

If you run the program multiple times in testing or with dfferent files, make sure to empty out the data before each run.

You can use this code as a starting point for your application: http://www.pythonlearn.com/code/emaildb.py (http://www.pythonlearn.com/code/emaildb.py).

The data file for this application is the same as in previous assignments: http://www.pythonlearn.com/code/mbox.txt (http://www.pythonlearn.com/code/mbox.txt).

Because the sample code is using an **UPDATE** statement and committing the results to the database as each record is read in the loop, it might take as long as a few minutes to process all the data. The commit insists on completely writing all the data to disk every time it is called.

The program can be speeded up greatly by moving the commit operation outside of the loop. In any database program, there is a balance between the number of operations you execute between commits and the importance of not losing the results of operations that have not yet been committed.