

# Project Examples with Visualizations

Barbara Ericson

Based on slides from Charles Severance

From Python for Everybody

[www.py4e.com](http://www.py4e.com)

# Grading Rubric for Final Project is on Canvas

- ▶ Two+ files (two+ functions)
  - ▶ Function to save data to the database
    - ▶ Save a max of 25 rows on each execution in one table per API/website
    - ▶ Must have two tables with a shared integer key for one API/website
      - ▶ Don't duplicate string data and don't break one table into two
  - ▶ Function to read data from the database, calculate something, write a report, and create visualizations from what you calculated
  - ▶ Don't store any data from an API/website in a file - it should be stored in the database!

# Breaking One Table Into Two

- ▶ If the data belongs to the same entity
- ▶ If both tables have the same number of rows
- ▶ If putting the data in one table would not result in any duplicate string data

Duplicate string data

The screenshot shows a database interface with two tables displayed side-by-side.

**Table: movies\_basic**

id	name	genre	cast
1	The Shawshank Redemption	Drama	Tim Robbins, Morgan Freeman
2	The Godfather	Crime,Drama	Marlon Brando, Al Pacino, James Caan
3	The Godfather Part II	Crime,Drama	Al Pacino, Robert De Niro, Robert Duvall
4	The Dark Knight	Action,Crime,Drama	Christian Bale, Heath Ledger, Aaron Eckhart
5	12 Angry Men	Crime,Drama	Henry Fonda, Lee J. Cobb, Martin Balsam
6	Schindler's List	Biography,Drama,History	Liam Neeson, Ralph Fiennes, Ben Kingsley
7	The Lord of the Rings: The Return of the King	Action,Adventure,Drama	Elijah Wood, Viggo Mortensen, Ian McKellen
8	Pulp Fiction	Crime,Drama	John Travolta, Uma Thurman, Samuel L. Jackson
9	The Good, the Bad and the Ugly	Adventure,Western	Clint Eastwood, Eli Wallach, Lee Van Cleef
10	Fight Club	Drama	Brad Pitt, Edward Norton, Meagan Shetterly
11	Forrest Gump	Drama,Romance	Tom Hanks, Robin Wright, Gary Sinise
12	Inception	Action,Adventure,Sci-Fi	Leonardo DiCaprio, Joseph Gordon-Levitt, Ellen Page
13	The Lord of the Rings: The Fellowship of the Ring	Action,Adventure,Drama	Elijah Wood, Ian McKellen, Orlando Bloom
14	Star Wars: Episode V - The Empire Strikes Back	Action,Adventure,Fantasy	Mark Hamill, Harrison Ford, Carrie Fisher
15	The Lord of the Rings: The Two Towers	Action,Adventure,Drama	Elijah Wood, Ian McKellen, Viggo Mortensen
16	Interstellar	Adventure,Drama,Sci-Fi	Matthew McConaughey, Anne Hathaway, Isla Fisher
17	City of God	Crime,Drama	Alexandre Rodrigues, Leandro Hassum, Caio Blat
18	Spirited Away	Animation,Adventure,Family	Daveigh Chase, Suzanne Pleshette, Kikuko Inoue
19	Saving Private Ryan	Drama,War	Tom Hanks, Matt Damon, Tom Sizemore
20	The Green Mile	Crime,Drama,Fantasy	Tom Hanks, Michael Clarke Duncan, Tim Robbins
21	Life Is Beautiful	Comedy,Drama,Romance	Roberto Benigni, Nicoletta Braschi
22	Se7en	Crime,Drama,Mystery	Morgan Freeman, Brad Pitt, Kevin Spacey
23	The Silence of the Lambs	Crime,Drama,Thriller	Jodie Foster, Anthony Hopkins
24	It's a Wonderful Life	Drama,Family,Fantasy	James Stewart, Donna Reed, Lionel Barrymore

**Table: movies\_ratings**

movie_id	imdb_rating	rotten_tomatoes_rating	meta_critic_rating
1	9.3	91%	82/100
2	9.2	97%	100/100
3	9.0	96%	90/100
4	9.0	94%	84/100
5	9.0	100%	97/100
6	9.0	98%	95/100
7	9.0	94%	94/100
8	8.9	92%	95/100
9	8.8	97%	90/100
10	8.8	79%	67/100
11	8.8	71%	82/100
12	8.8	87%	74/100
13	8.8	91%	92/100
14	8.7	95%	82/100
15	8.8	95%	87/100
16	8.7	73%	74/100
17	8.6	91%	79/100
18	8.6	96%	96/100
19	8.6	94%	91/100
20	8.6	79%	61/100
21	8.6	81%	59/100
22	8.6	83%	65/100
23	8.6	95%	86/100
24	8.6	94%	89/100

# Duplicate String Data

- ▶ Do not store duplicate string data in a database!
- ▶ In the same table
- ▶ In two different tables
- ▶ Watch out for dates if they are strings

The image shows two database tables side-by-side. The left table is titled "Players" and has columns: PLAYER\_ID, NAME, TEAM\_ID, and AGE. The right table is titled "Salaries" and has columns: player\_id, name, and salary. Both tables contain numerous rows of data, including names like LeBron James, Udonis Haslem, Andre Iguodala, Chris Paul, Rudy Gay, Kyle Lowry, P.J. Tucker, Kevin Durant, Al Horford, Mike Conley, Jeff Green, Thaddeus Young, Derrick Rose, Russell Westbrook, Kevin Love, Eric Gordon, Brook Lopez, Robin Lopez, JaVale McGee, Serge Ibaka, Nicolas Batum, George Hill, DeAndre Jordan, Goran Dragic, Lonzo Ball, Eric Bledsoe, Derrick Favors, Chet Holmgren, Aron Baynes, Danilo Gallinari, David Nwaba, Dante Exum, Lou Williams, Maurice Harkless, Chandler Hutchison, Gabriel Deck, Sekou Doumbouya, Trey Burke, Marc Gasol, Anthony Tolliver, Marquese Chriss, Nik Stauskas, Jahlil Okafor, Abdel Nader, Jontay Porter, Denzel Valentine, Killian Tillie, and Wes Iwundu. The "NAME" column in the Players table and the "name" column in the Salaries table both contain many duplicates.

Table: Players				
	PLAYER_ID	NAME	TEAM_ID	AGE
1	2544	LeBron James	1610612747	38
2	2617	Udonis Haslem	1610612748	43
3	2738	Andre Iguodala	1610612744	39
4	101108	Chris Paul	1610612756	38
5	200752	Rudy Gay	1610612762	36
6	200768	Kyle Lowry	1610612748	37
7	200782	P.J.Tucker	1610612755	38
8	201142	Kevin Durant	1610612756	34
9	201143	Al Horford	1610612738	37
10	201144	Mike Conley	1610612750	35
11	201145	Jeff Green	1610612743	36
12	201152	Thaddeus Young	1610612761	35
13	201565	Derrick Rose	1610612752	34
14	201566	Russell Westbrook	1610612746	34
15	201567	Kevin Love	1610612748	34
16	201569	Eric Gordon	1610612746	34
17	201572	Brook Lopez	1610612749	35
18	201577	Robin Lopez	1610612739	35
19	201580	JaVale McGee	1610612742	35
20	201586	Serge Ibaka	1610612749	33
21	201587	Nicolas Batum	1610612746	34
22	201588	George Hill	1610612754	37
23	201599	DeAndre Jordan	1610612743	34
24	201609	Goran Dragic	1610612749	37

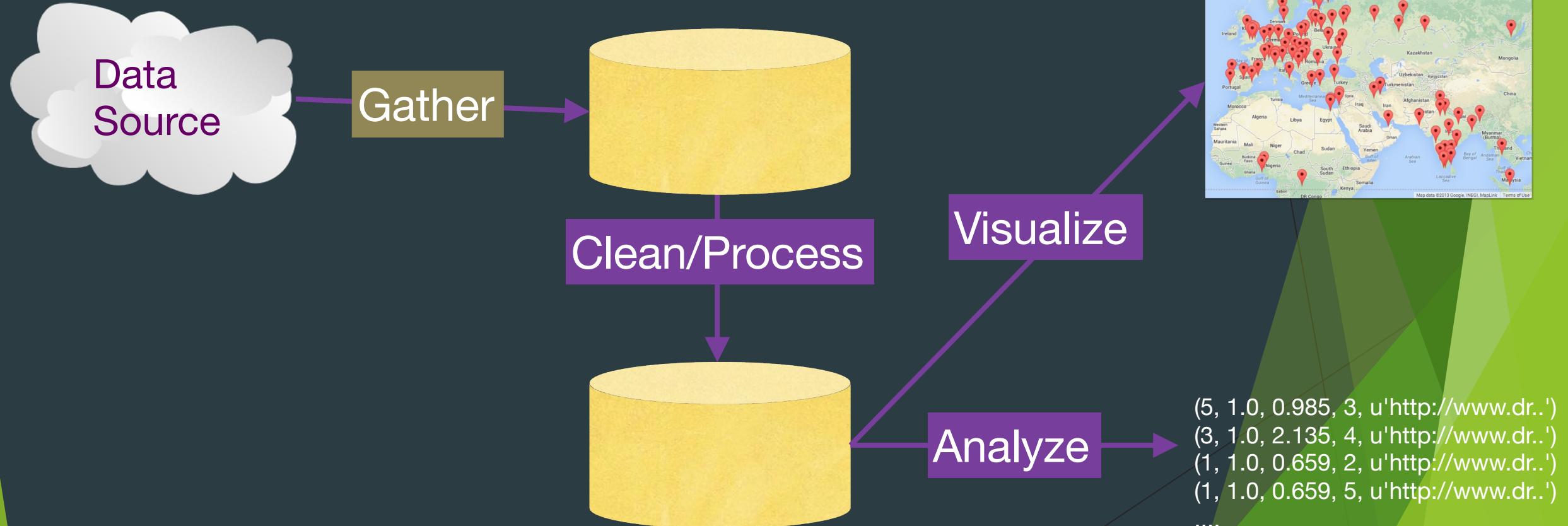
  

Table: Salaries			
	player_id	name	salary
1	1	Lonzo Ball	19534884
2	2	Eric Bledsoe	19375000
3	3	Derrick Favors	10183800
4	4	Chet Holmgren	9891240
5	5	Aron Baynes	7350000
6	6	Danilo Gallinari	6479000
7	7	David Nwaba	5022000
8	8	Dante Exum	5000000
9	9	Lou Williams	5000000
10	10	Maurice Harkless	4564980
11	11	Chandler Hutchison	4019459
12	12	Gabriel Deck	3676852
13	13	Sekou Doumbouya	3613680
14	14	Trey Burke	3243750
15	15	Marc Gasol	2692991
16	16	Anthony Tolliver	2692991
17	17	Marquese Chriss	2193920
18	18	Nik Stauskas	2193920
19	19	Jahlil Okafor	2130023
20	20	Abdel Nader	2000000
21	21	Jontay Porter	1950000
22	22	Denzel Valentine	1939350
23	23	Killian Tillie	1901625
24	24	Wes Iwundu	1824003

# More Useful SQL Commands

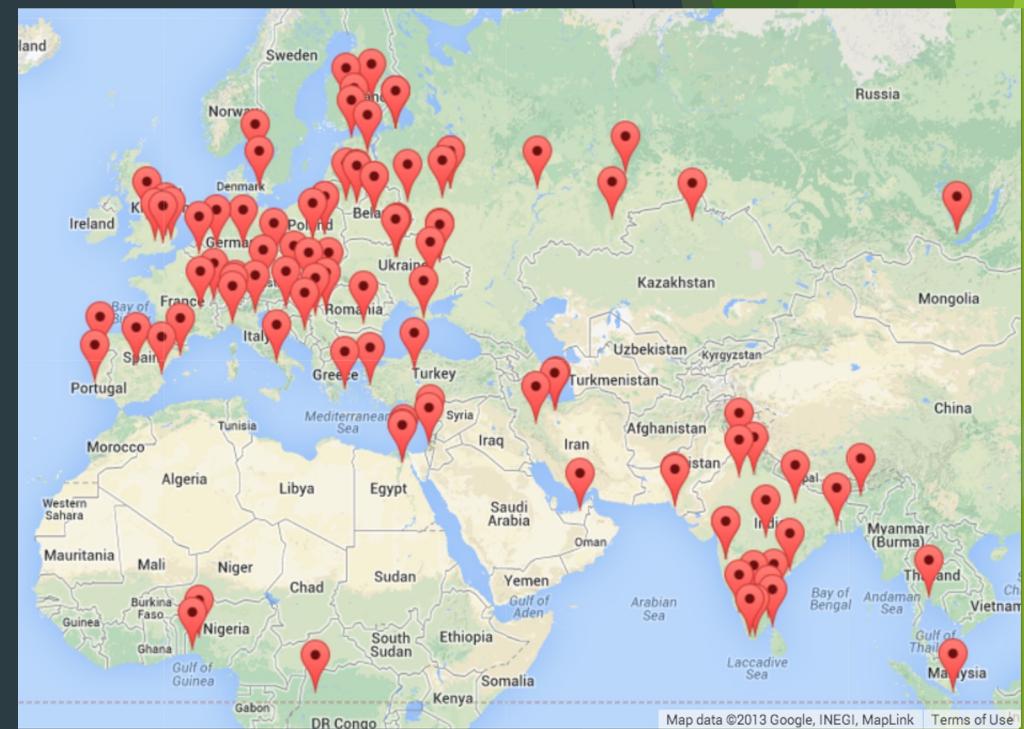
- ▶ `SELECT MAX(col) FROM table`
- ▶ `SELECT COUNT(col) FROM table`
- ▶ `SELECT AVG(col) FROM table`
- ▶ `SELECT SUM(col) FROM table`

# Multi-Step Data Analysis

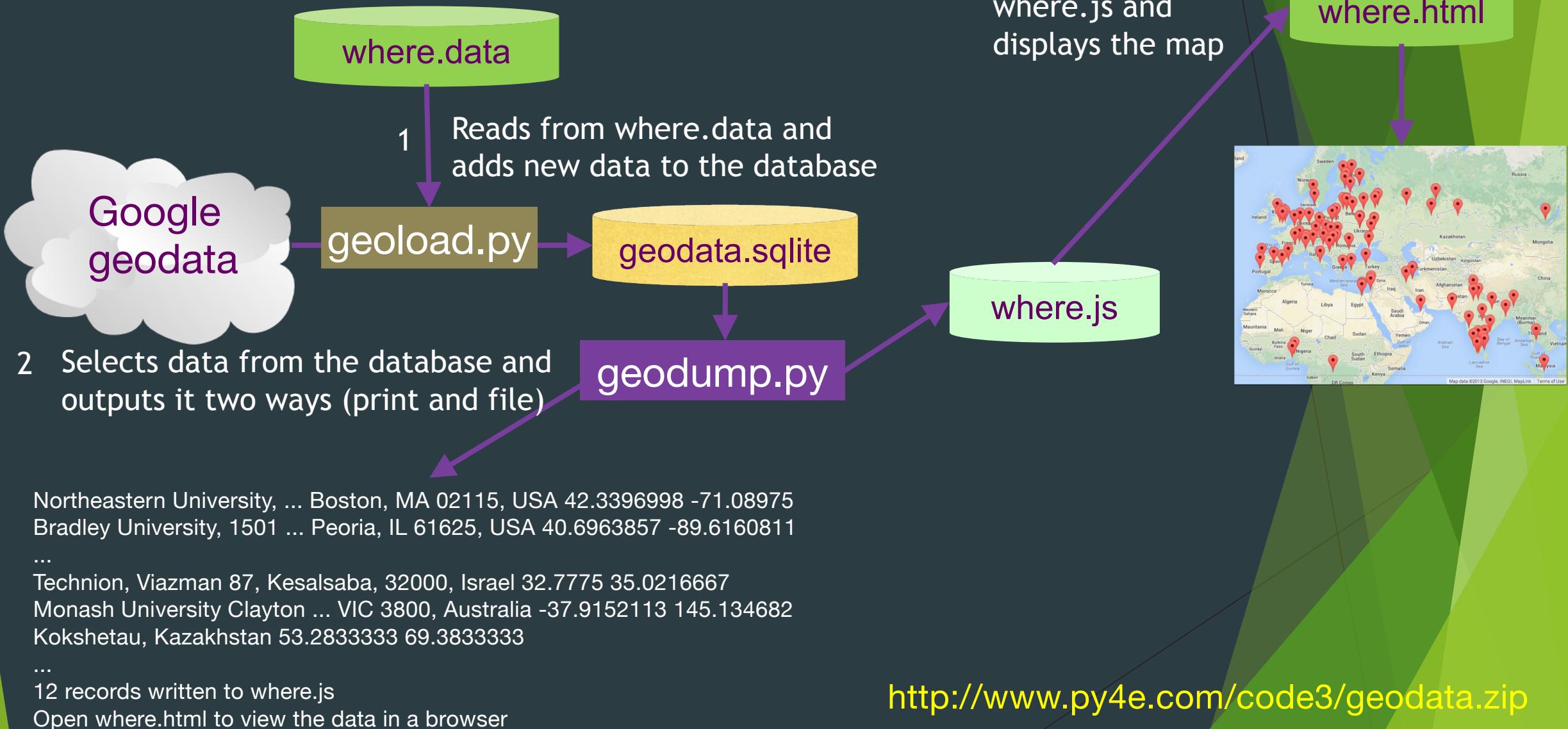


# GeoData

- ▶ Makes a Google Map from user entered location data from a MOOC
- ▶ Uses the Google Geodata API to get latitude and longitude data
- ▶ Stores data in a database to avoid rate limiting and allow restarting
- ▶ Visualized in a browser using the Google Maps API

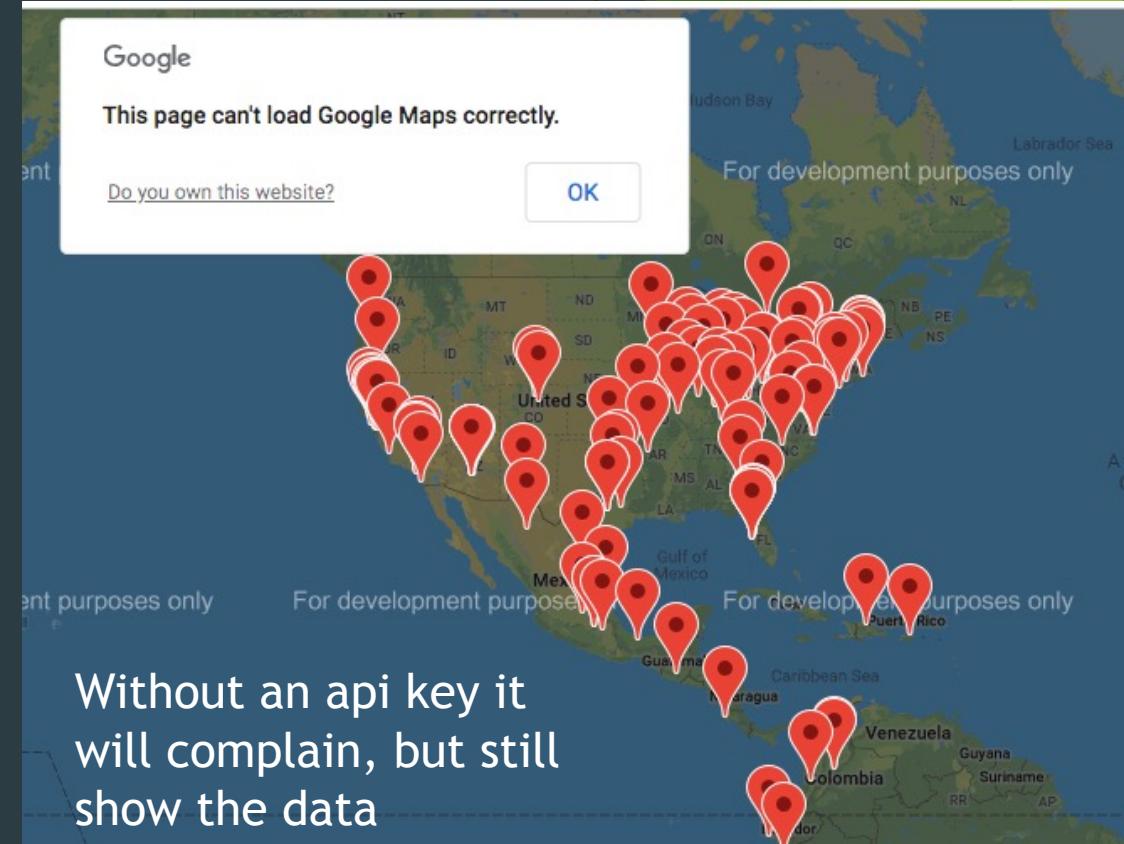


<http://www.py4e.com/code3/geodata.zip>



# Try It

- ▶ Download the zip file from Canvas
- ▶ Open *where.html* in a browser to see the resulting visualization
  - ▶ Hover over the markers to see information



# Run the Code!

- ▶ cd to the current directory
  - ▶ The code doesn't add the path to the current file
- ▶ python geoload.py
  - ▶ read from where.data and add to the database
- ▶ Browse the database - geodata.sqlite
- ▶ python geodump.py
  - ▶ Reads from database, prints output, and stores in where.js
- ▶ Click on where.html
  - ▶ Displays the data in where.js

# How does the visualization work?

- ▶ Reads a javascript list of lists from where.js

```
1 myData = [                                     where.js
2 [42.340075,-71.0895367, 'Northeastern, Boston, MA 02115, USA'],
3 [32.778949,35.019648, 'Technion/ Sports Building, Haifa'],
4 [33.1561058,131.826132, 'Japan, 〒875-0002 Ōita-ken, Usuki-shi, Shitanoe, 1232-2 UMD'],
5 [42.4036847,-71.120482, 'South Hall Tufts University, 30 Lower Campus Rd, Somerville, MA 02144, USA'],
6 [-37.914517,145.1303881, 'Monash College, Wellington Rd, Clayton VIC 3168, Australia'],
7 [53.2948229,69.4047872, 'Kokshetau 020000, Kazakhstan'],

1 <html>                                         where.html
2   <head>
3     <meta name="viewport" content="initial-scale=1.0, user-scalable=no">
4     <meta charset="utf-8">
5     <title>A Map of Information</title>
6     <link href="https://google-developers.appspot.com/maps/documentation/javascript/examples/default.css" rel="stylesheet">
7
8   <!-- If you are in China, you may need to use theis site for the Google Maps code
9   <script src="https://maps.google.cn/maps/api/js" type="text/javascript"></script> -->
10
11  <script src="https://maps.googleapis.com/maps/api/js"></script>
12  <script src="where.js"></script>
```

# Getting the Data

- ▶ Reads from *where.data*
  - ▶ Gets the location from a file
  - ▶ Checks if in the database
    - ▶ If yes, continue
    - ▶ Else, get the data

```
1 AGH University of Science and Technology
2 Academy of Fine Arts Warsaw Poland
3 American University in Cairo
4 Arizona State University
5 Athens Information Technology
6 BITS Pilani
7 Babcock University
8 Banaras Hindu University
9 Bangalore University
```

where.data

```
33 fh = open("where.data")
34 count = 0
35 for line in fh:
36     if count > 200 :
37         print('Retrieved 200 locations, restart to retrieve more')
38         break Stop the current loop
39
40 address = line.strip() Get the current address
41 print('')
42 cur.execute("SELECT geodata FROM Locations WHERE address= ?",
43             (memoryview(address.encode()), ))
44
45 try: Is in database?
46     data = cur.fetchone()[0]
47     print("Found in database ",address)
48     continue
49 except: If yes, continue (skip rest of this)
50     pass Otherwise keep going
```

geoload.data

# `continue`, `break`, and `exit` in Python

- ▶ `continue`
  - ▶ Returns control to the beginning of the loop
    - ▶ Won't execute any statements in the loop after it
- ▶ `break`
  - ▶ Terminates the loop
    - ▶ Execution continues after the loop
- ▶ `sys.exit()`
  - ▶ Ends the program

# Get the Data from the web

```
52     parms = dict()
53     parms["address"] = address
54     if api_key is not False: parms['key'] = api_key
55     url = serviceurl + urllib.parse.urlencode(parms)
56
57     print('Retrieving', url)
58     uh = urllib.request.urlopen(url, context=ctx)
59     data = uh.read().decode()
60     print('Retrieved', len(data), 'characters', data[:20].replace('\n', ' '))
61     count = count + 1  Increment the count
62
```

Code uses urllib rather than requests

Replace new line with space

# If got the JSON data, save it in the database

```
63 v     try:
64         js = json.loads(data)
65 v     except:
66         print(data) # We print in case unicode causes an error
67         continue
68
69 v     if 'status' not in js or (js['status'] != 'OK' and js['status'] != 'ZERO_RESULTS') :
70         print('==== Failure To Retrieve ====')
71         print(data)
72         break
73
74 v     cur.execute('''INSERT INTO Locations (address, geodata)
75                 VALUES ( ?, ? )''', (memoryview(address.encode()), memoryview(data.encode())))
76     conn.commit()
77 v     if count % 10 == 0 :           Insert into the database
78         print('Pausing for a bit...')
79         time.sleep(5)               Commit the changes
80
81 v
82 v
```

Commit the changes  
Sleep for 5 seconds every 10<sup>th</sup> time

# geodump.py

- ▶ Select from the database
- ▶ Loop through the results
- ▶ Convert from JSON to Python
- ▶ Check result was okay

```
5 conn = sqlite3.connect('geodata.sqlite')
6 cur = conn.cursor()
7
8 cur.execute('SELECT * FROM Locations')
9 fhand = codecs.open('where.js', 'w', "utf-8")
10 fhand.write("myData = [\n")
11 count = 0
12 for row in cur :
13     data = str(row[1].decode())
14     try: js = json.loads(str(data))
15     except: continue
16
17     if not('status' in js and js['status'] == 'OK') : continue
18     count = count + 1
19     if count % 1000 == 0:
20         print(count)
```

# geodump.py (continued)

- ▶ Print the data
- ▶ Write the data to
  - ▶ where.js

```
19     lat = js["results"][0]["geometry"]["location"]["lat"]
20     lng = js["results"][0]["geometry"]["location"]["lng"]
21     if lat == 0 or lng == 0 : continue
22     where = js['results'][0]['formatted_address']
23     where = where.replace("", "")
24     try :
25         print(where, lat, lng)    Print the data
26
27         count = count + 1
28         if count > 1 : fhand.write(",\n")
29         output = "["+str(lat)+","+str(lng)+", '"+where+"']"
30         fhand.write(output)      Write the data
31     except:
32         continue
33
34     fhand.write("\n];\n")
35     cur.close()
36     fhand.close()
37     print(count, "records written to where.js")
38     print("Open where.html to view the data in a browser")
```

# Peer Instruction #1

Q-1: Which of the following is how geoload.py limits the data that it gets from the API?

- A. Only asks the API for 200 items at a time
- B. Quits the program after retrieving 200 new items from the API
- C. Quits the program after reading 200 items from where.data
- D. Only reads 200 items from the database

**Check Me**

**Compare me**

# Restart and Limit Process for geoload.py

- ▶ Create the table if it doesn't exist
  - ▶ Keep the table around
- ▶ Read locations from a file (where.data)
- ▶ Keep a count of the number retrieved from the API and stop at 200
- ▶ Check if the current location is already in the database
  - ▶ and if so move on to the next
  - ▶ Otherwise, get the data for the current location and add it to the database
  - ▶ Increment the count of items retrieved
- ▶ Sleep every 10<sup>th</sup> retrieved item
  - ▶ Some systems limit how many items you can get in a time period

# Mailing Lists - Gmane

- ▶ Crawl the archive of a mailing list
  - ▶ Do some analysis / cleanup
  - ▶ Visualize the data as word cloud and lines



<http://www.py4e.com/code3/gmane.zip>

# Warning: This Dataset is > 1GB

- ▶ Do not just point this application at [gmane.org](http://gmane.org) and let it run
- ▶ There is no rate limit - these are cool folks

Use this for your testing:

<http://mbox.dr-chuck.net/sakai.devel/4/5>

mbox.dr-chuck.net

1. add data to the database

gmane.py

content.sqlite

mapping.sqlite

gmodel.py

2. clean the data

index.sqlite

gbasic.py

3. output data

How many to dump? 5

Loaded messages= 51330 subjects= 25033 senders= 1584

Top 5 Email list participants

steve.swinsburg@gmail.com 2657

azeckoski@unicon.net 1742

ieb@tfd.co.uk 1591

csev@umich.edu 1304

david.horwitz@uct.ac.za 1184

...

<http://www.py4e.com/code3/gmane.zip>

index.sqlite

gline.py

4. create word cloud

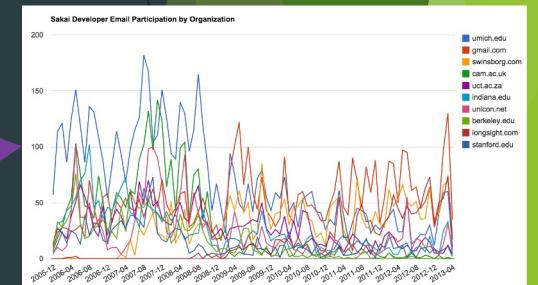
gline.js

5. output line chart

gword.htm  
d3.js

gword.js

gword.py



gline.htm  
d3.js

# D3.js - Data-Driven Documents

- ▶ JavaScript library for modifying an HTML document to display data
  - ▶ Very popular
  - ▶ Used to create data visualizations in web browsers
    - ▶ Can be interactive
  - ▶ See <https://d3js.org/>

# Examples:

<https://github.com/d3/d3/wiki/Gallery>

## Gallery

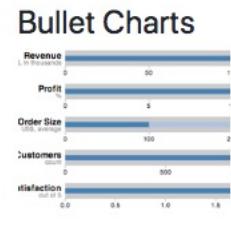
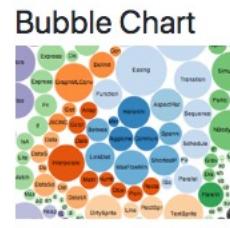
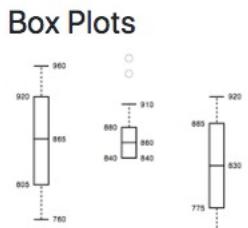
[Jump to bottom](#)

Mike Bostock edited this page on Oct 27 · 1287 revisions

[Wiki](#) ▶ [Gallery](#)

Welcome to the D3 gallery! More examples are available for forking on [Observable](#); see [my profile](#) and the [visualization collection](#). Feel free to publish and share your own!

## Visual Index



# Try it!

- ▶ Download

<http://www.py4e.com/code3/gmane.zip>

- ▶ Can also download

<https://online.dr-chuck.com/files/sakai/email/content.sqlite>

# Run gmane.py

- ▶ It asks how many messages to get and then gets that number (adds to the database)

```
How many messages:5
http://mbox.dr-chuck.net/sakai.devel/32/33 2910
    ggolden@umich.edu 2005-12-12T20:02:00-05:00 sakai 2.1 conversion – an update
http://mbox.dr-chuck.net/sakai.devel/33/34 3512
    ys2n@virginia.edu 2005-12-12T21:15:22+00:00 worksite taxonomy
http://mbox.dr-chuck.net/sakai.devel/34/35 5362
    lfernandez@weber.edu 2005-12-13T01:08:01-07:00 re: memory error with 2.1
http://mbox.dr-chuck.net/sakai.devel/35/36 2270
    lfernandez@weber.edu 2005-12-13T01:29:49-07:00 audio recordings of austin presentations
http://mbox.dr-chuck.net/sakai.devel/36/37 6958
    ys2n@virginia.edu 2005-12-13T02:46:51+00:00 re: worksite taxonomy
How many messages:8
http://mbox.dr-chuck.net/sakai.devel/37/38 8425
    jleasia@umich.edu 2005-12-13T08:02:47-05:00 re: worksite taxonomy
http://mbox.dr-chuck.net/sakai.devel/38/39 3014
    csev@umich.edu 2005-12-13T08:12:08-05:00 re: audio recordings of austin presentations
http://mbox.dr-chuck.net/sakai.devel/39/40 9459
```

# Peer Instruction #2

Q-1: What technique does gmane.py use to restart?

- A. Read items to be processed from a file and only gather data that isn't already in the database
- B. Keeps a count of the number of items retrieved and exits after a limit is reached
- C. Limits the number of items retrieved from the API on the API call
- D. Uses the primary key to know where to start when restarted

**Check Me**

**Compare me**

Problem: 3 -- Activity: 1 Multiple Choice (db-example-gmane)

# Run gmodel.py

- ▶ Cleans and summarizes the data
  - ▶ Reads from content.sqlite and mapping.sqlite
  - ▶ Writes to index.sqlite

```
[0587395416:gmane-orig barbarer$ python gmodel.py
Loaded allsenders 1771 and mapping 29 dns mapping 1
1 2005-12-08T23:34:30-06:00 ggolden22@mac.com
251 2005-12-22T10:03:20-08:00 tpamsler@ucdavis.edu
501 2006-01-12T11:17:34-05:00 lance@indiana.edu
751 2006-01-24T11:13:28-08:00 vrajgopalan@ucmerced.edu
1001 2006-02-02T08:27:30-07:00 john.ellis@rsmart.com
1251 2006-02-15T19:13:17-05:00 ggolden22@mac.com
1501 2006-02-23T16:49:00-05:00 csev@umich.edu
1751 2006-03-13T13:48:36-05:00 csev@umich.edu
```

# Run gbasic.py

- ▶ Reports on the data
  - ▶ Top email list participants
  - ▶ Top email list organizations

```
0587395416:gmane-orig barbarer$ python gbasic.py
How many to dump? 5
Loaded messages= 58970 subjects= 29065 senders= 1765

Top 5 Email list participants
steve.swinsburg@swinsborg.com 3303
azeckoski@unicon.net 1907
ian@cam.ac.uk 1591
csev@umich.edu 1471
david.horwitz@uct.ac.za 1221

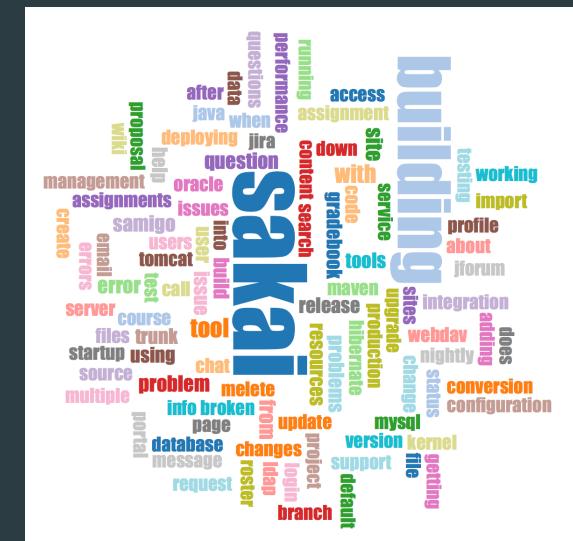
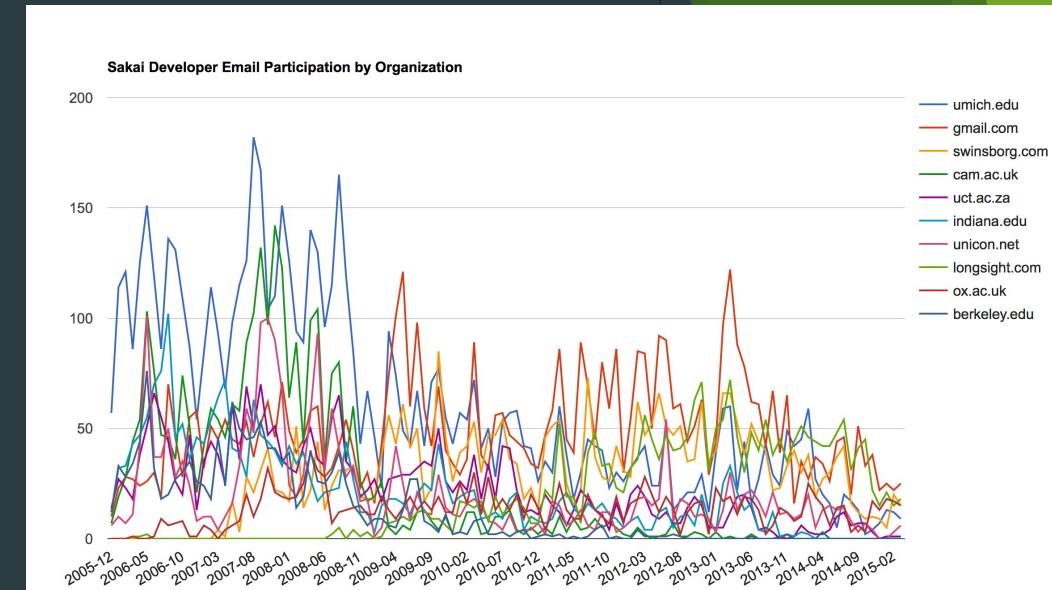
Top 5 Email list organizations
umich.edu 6785
gmail.com 5588
swinsborg.com 3303
cam.ac.uk 2626
uct.ac.za 2577
```

# Create the visualizations

- ▶ Run `gline.py`
  - ▶ Reads from `index.sqlite`
  - ▶ Writes to `gline.js`
- ▶ Run `gword.py`
  - ▶ Reads from `index.sqlite`
  - ▶ Writes `gword.js`

# View the visualizations

- ▶ Open gline.htm in a browser
    - ▶ Line chart using d3.js
  - ▶ Open gword.htm in a browser



# How to Limit the Rows Stored to 25

- ▶ If your API has way to specify a limit, use it
- ▶ Keep a count of the number of new rows added to the database and break when you reach 25
  - ▶ Check that the data isn't in the database before you add it
  - ▶ Can use `SELECT MAX(column)` to get the highest number on a primary key
- ▶ Have the user input something like a city name and add 25 rows for that item
  - ▶ Or read data from a file (like city names)