Version 1.1

December 7, 2015

Presented by:

DEBAJYOTI DIBYARANJAN NAYAK

Sagar Choudhary

Rahul Bhattad

**noSQL database Design**

MS - Data science (Indian university)



Table of Contents

[Project Proposal for 2](#_Toc437264400)

[“Design of a noSQL database using MONGODB AND 2](#_Toc437264401)

[Cassandra for Big Data” 2](#_Toc437264402)

[Team: 2](#_Toc437264403)

[Email: 2](#_Toc437264404)

[Description: 2](#_Toc437264405)

[Section 1: Data Modeling using Cassandra 2](#_Toc437264406)

[Section 2: Data Modeling using MongoDB 2](#_Toc437264407)

[Environment: 2](#_Toc437264408)

[Deliverable: 3](#_Toc437264409)

[Project Work for 4](#_Toc437264410)

[“Design of a noSQL database using MONGODB 4](#_Toc437264411)

[AND Cassandra for Big Data” 4](#_Toc437264412)

[Overview 4](#_Toc437264413)

[Section 1: Database design using Cassandra 4](#_Toc437264414)

[Project Environment: 4](#_Toc437264415)

[Installation of Cassandra. 4](#_Toc437264416)

[Entity-Relationship diagram 6](#_Toc437264417)

[Database Schema 6](#_Toc437264418)

[Import data: 19](#_Toc437264419)

[Screens shots 20](#_Toc437264420)

[Delivered: 21](#_Toc437264421)

[Section 2: Database design using MongoDB 22](#_Toc437264422)

[Project Environment: 22](#_Toc437264423)

[Installation of Mongo DB. 22](#_Toc437264424)

[Sample Raw data 23](#_Toc437264425)

[Data Cleanup (Reformat) 23](#_Toc437264426)

[Query and Update 24](#_Toc437264427)

[Screenshots 26](#_Toc437264428)

[Delivered: 29](#_Toc437264429)

# Project Proposal for

# “Design of a noSQL database using MONGODB AND

# Cassandra for Big Data”

# Team:

* Rahul Bhattad,
* Sagar Choudhary,
* Debajyoti Nayak

# Email:

1. Rahul Bhattad ([bhattadr2007@gmail.com](mailto:bhattadr2007@gmail.com) )
2. Sagar Choudhary([sagar.choudhary@gmail.com](mailto:sagar.choudhary@gmail.com) )
3. Debajyoti Nayak ([debajyoti.nayak@gmail.com](mailto:debajyoti.nayak@gmail.com) )

# Description:

This project is divided into 2 sections:

# Section 1: Data Modeling using Cassandra

The objective of Section 1 is to design a data model leveraging Cassandra for Ratting data of Videos/actors, users etc. , additionally ingest and query a large dataset using Cassandra. We will also provide some scripts to cleanup and import the data.

# Section 2: Data Modeling using MongoDB

The objective of Section 2 is to design a data model leveraging MongoDB for location data, additionally ingest and query a large dataset using MongoDB. We will also provide some scripts to cleanup and import the data.

# Environment:

* Oracle Enterprise Linux.
* MongoDB
* Google Map
* Shell Scripts
* JAVA
* Cassandra

# Deliverable:

* Instructions
* MongoDB/Cassandra Schema design
* Sample raw data and cleaned Dataset.
* Shell Scripts
* MongoDB/Cassandra scripts
* MongoDB/Cassandra files

# Project Work for

# “Design of a noSQL database using MONGODB

# AND Cassandra for Big Data”

# Overview

This project is divided into 2 sections.

Section 1 describes the Database design using Cassandra and Section 2 describes the Database design using MongoDB.

What is Cassandra?

Apache **Cassandra** is an open source distributed **database** management system designed to handle large amounts of data across many commodity servers, providing high availability with no single point of failure. It is a noSQL database with SQL interface for the schema design and playing with data.

What is MongoDB?

**MongoDB** is an open source, document-oriented database designed with both scalability and developer agility in mind. Instead of storing your data in tables and rows as you would with a relational database, in **MongoDB** you store JSON-like documents with dynamic schemas.

# Section 1: Database design using Cassandra

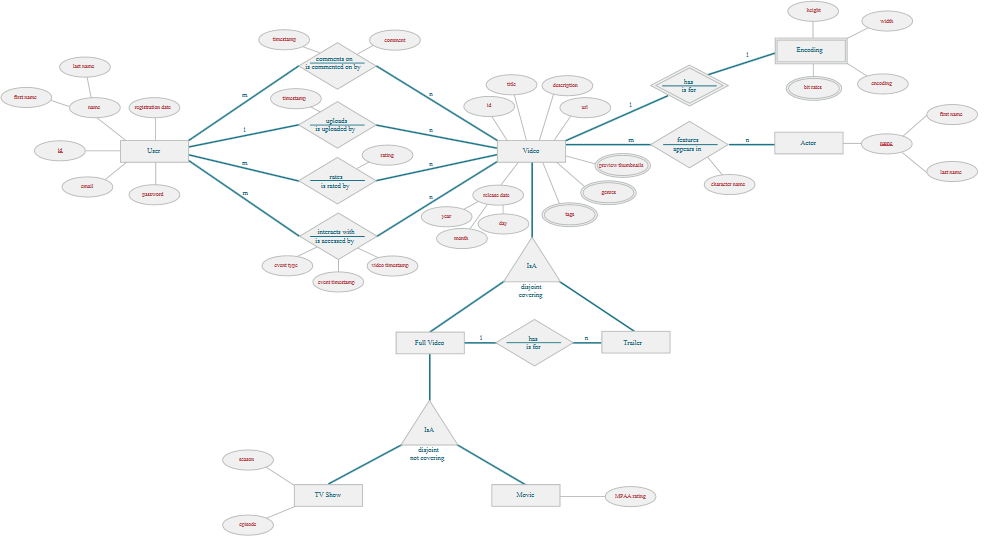
# Project Environment:

1. Red Hat Enterprise Linux Server release 5.9 (Tikanga)
2. Cassandra
3. OpenSSL

# Installation of Cassandra.

1. Install JAVA
   1. tar -zxf jdk-8u65-linux-x64.tar.gz
2. setup environment
   1. JAVA\_HOME=/usr/local/java/jdk1.8.0\_65
   2. PATH=$PATH:$HOME/bin:$JAVA\_HOME/bin
   3. JRE\_HOME=/usr/local/java/jdk1.8.0\_65
   4. PATH=$PATH:$HOME/bin:$JRE\_HOME/bin
   5. export JAVA\_HOME
   6. export JRE\_HOME
   7. export PATH
3. Install Cassandra
   1. tar -zxf apache-cassandra-2.2.3-bin.tar.gz
   2. mv apache-cassandra-2.2.3 ~/cassandra
4. Set environment Variable
5. sudo mkdir /var/lib/cassandra
6. sudo mkdir /var/log/cassandra
7. sudo chown -R $USER:$GROUP /var/lib/cassandra
8. sudo chown -R $USER:$GROUP /var/log/cassandra
9. export CASSANDRA\_HOME=~/cassandra
10. export PATH=$PATH:$CASSANDRA\_HOME/bin
11. start database Cassandra
12. sudo sh ~/cassandra/bin/cassandra
13. Start cqlsh
14. sudo sh ~/cassandra/bin/cqlsh

# Entity-Relationship diagram



# Database Schema

Here we have created following tables: Table Schemas are separated by color coding

* 1. encoding\_type
  2. trailers\_by\_video
  3. users
  4. videos
  5. users\_by\_email
  6. latest\_videos
  7. comments\_by\_user
  8. actors\_by\_video
  9. video\_interactions\_by\_user\_video
  10. videos\_by\_user
  11. ratings\_by\_video
  12. comments\_by\_video

cqlsh:killr\_video> describe keyspace killr\_video;

CREATE KEYSPACE killr\_video WITH replication = {'class': 'SimpleStrategy', 'replication\_factor': '1'} AND durable\_writes = true;

CREATE TYPE killr\_video.encoding\_type (

encoding text,

height int,

width int,

bit\_rates frozen<set<text>>

);

CREATE TABLE killr\_video.trailers\_by\_video (

video\_id timeuuid,

title text,

trailer\_id timeuuid,

preview\_thumbnails map<int, blob>,

tags set<text>,

type text,

PRIMARY KEY (video\_id, title, trailer\_id)

) WITH CLUSTERING ORDER BY (title ASC, trailer\_id ASC)

AND bloom\_filter\_fp\_chance = 0.01

AND caching = '{"keys":"ALL", "rows\_per\_partition":"NONE"}'

AND comment = ''

AND compaction = {'class': 'org.apache.cassandra.db.compaction.SizeTieredCompactionStrategy'}

AND compression = {'sstable\_compression': 'org.apache.cassandra.io.compress.LZ4Compressor'}

AND dclocal\_read\_repair\_chance = 0.1

AND default\_time\_to\_live = 0

AND gc\_grace\_seconds = 864000

AND max\_index\_interval = 2048

AND memtable\_flush\_period\_in\_ms = 0

AND min\_index\_interval = 128

AND read\_repair\_chance = 0.0

AND speculative\_retry = '99.0PERCENTILE';

CREATE TABLE killr\_video.users (

user\_id uuid PRIMARY KEY,

email text,

first\_name text,

last\_name text,

registration\_date timestamp

) WITH bloom\_filter\_fp\_chance = 0.01

AND caching = '{"keys":"ALL", "rows\_per\_partition":"NONE"}'

AND comment = ''

AND compaction = {'class': 'org.apache.cassandra.db.compaction.SizeTieredCompactionStrategy'}

AND compression = {'sstable\_compression': 'org.apache.cassandra.io.compress.LZ4Compressor'}

AND dclocal\_read\_repair\_chance = 0.1

AND default\_time\_to\_live = 0

AND gc\_grace\_seconds = 864000

AND max\_index\_interval = 2048

AND memtable\_flush\_period\_in\_ms = 0

AND min\_index\_interval = 128

AND read\_repair\_chance = 0.0

AND speculative\_retry = '99.0PERCENTILE';

CREATE TABLE killr\_video.videos (

video\_id timeuuid PRIMARY KEY,

avg\_rating float,

description text,

encoding frozen<encoding\_type>,

genres set<text>,

mpaa\_rating text,

preview\_thumbnails map<int, blob>,

release\_date timestamp,

tags set<text>,

title text,

type text,

url text,

user\_id uuid

) WITH bloom\_filter\_fp\_chance = 0.01

AND caching = '{"keys":"ALL", "rows\_per\_partition":"NONE"}'

AND comment = ''

AND compaction = {'class': 'org.apache.cassandra.db.compaction.SizeTieredCompactionStrategy'}

AND compression = {'sstable\_compression': 'org.apache.cassandra.io.compress.LZ4Compressor'}

AND dclocal\_read\_repair\_chance = 0.1

AND default\_time\_to\_live = 0

AND gc\_grace\_seconds = 864000

AND max\_index\_interval = 2048

AND memtable\_flush\_period\_in\_ms = 0

AND min\_index\_interval = 128

AND read\_repair\_chance = 0.0

AND speculative\_retry = '99.0PERCENTILE';

CREATE TABLE killr\_video.users\_by\_email (

email text PRIMARY KEY,

password text,

user\_id uuid

) WITH bloom\_filter\_fp\_chance = 0.01

AND caching = '{"keys":"ALL", "rows\_per\_partition":"NONE"}'

AND comment = ''

AND compaction = {'class': 'org.apache.cassandra.db.compaction.SizeTieredCompactionStrategy'}

AND compression = {'sstable\_compression': 'org.apache.cassandra.io.compress.LZ4Compressor'}

AND dclocal\_read\_repair\_chance = 0.1

AND default\_time\_to\_live = 0

AND gc\_grace\_seconds = 864000

AND max\_index\_interval = 2048

AND memtable\_flush\_period\_in\_ms = 0

AND min\_index\_interval = 128

AND read\_repair\_chance = 0.0

AND speculative\_retry = '99.0PERCENTILE';

CREATE TABLE killr\_video.latest\_videos (

video\_bucket int,

video\_id timeuuid,

preview\_thumbnails map<int, blob>,

tags set<text>,

title text,

type text,

PRIMARY KEY (video\_bucket, video\_id)

) WITH CLUSTERING ORDER BY (video\_id DESC)

AND bloom\_filter\_fp\_chance = 0.01

AND caching = '{"keys":"ALL", "rows\_per\_partition":"NONE"}'

AND comment = ''

AND compaction = {'class': 'org.apache.cassandra.db.compaction.SizeTieredCompactionStrategy'}

AND compression = {'sstable\_compression': 'org.apache.cassandra.io.compress.LZ4Compressor'}

AND dclocal\_read\_repair\_chance = 0.1

AND default\_time\_to\_live = 0

AND gc\_grace\_seconds = 864000

AND max\_index\_interval = 2048

AND memtable\_flush\_period\_in\_ms = 0

AND min\_index\_interval = 128

AND read\_repair\_chance = 0.0

AND speculative\_retry = '99.0PERCENTILE';

CREATE TABLE killr\_video.comments\_by\_user (

user\_id uuid,

posted\_timestamp timestamp,

video\_id timeuuid,

comment text,

preview\_thumbnails map<int, blob>,

tags set<text>,

title text,

type text,

PRIMARY KEY (user\_id, posted\_timestamp, video\_id)

) WITH CLUSTERING ORDER BY (posted\_timestamp DESC, video\_id ASC)

AND bloom\_filter\_fp\_chance = 0.01

AND caching = '{"keys":"ALL", "rows\_per\_partition":"NONE"}'

AND comment = ''

AND compaction = {'class': 'org.apache.cassandra.db.compaction.SizeTieredCompactionStrategy'}

AND compression = {'sstable\_compression': 'org.apache.cassandra.io.compress.LZ4Compressor'}

AND dclocal\_read\_repair\_chance = 0.1

AND default\_time\_to\_live = 0

AND gc\_grace\_seconds = 864000

AND max\_index\_interval = 2048

AND memtable\_flush\_period\_in\_ms = 0

AND min\_index\_interval = 128

AND read\_repair\_chance = 0.0

AND speculative\_retry = '99.0PERCENTILE';

CREATE TABLE killr\_video.actors\_by\_video (

video\_id timeuuid,

actor\_name text,

character\_name text,

PRIMARY KEY (video\_id, actor\_name, character\_name)

) WITH CLUSTERING ORDER BY (actor\_name ASC, character\_name ASC)

AND bloom\_filter\_fp\_chance = 0.01

AND caching = '{"keys":"ALL", "rows\_per\_partition":"NONE"}'

AND comment = ''

AND compaction = {'class': 'org.apache.cassandra.db.compaction.SizeTieredCompactionStrategy'}

AND compression = {'sstable\_compression': 'org.apache.cassandra.io.compress.LZ4Compressor'}

AND dclocal\_read\_repair\_chance = 0.1

AND default\_time\_to\_live = 0

AND gc\_grace\_seconds = 864000

AND max\_index\_interval = 2048

AND memtable\_flush\_period\_in\_ms = 0

AND min\_index\_interval = 128

AND read\_repair\_chance = 0.0

AND speculative\_retry = '99.0PERCENTILE';

CREATE TABLE killr\_video.video\_interactions\_by\_user\_video (

user\_id uuid,

video\_id timeuuid,

event\_timestamp timestamp,

event\_type text,

video\_timestamp timestamp,

PRIMARY KEY ((user\_id, video\_id), event\_timestamp)

) WITH CLUSTERING ORDER BY (event\_timestamp DESC)

AND bloom\_filter\_fp\_chance = 0.01

AND caching = '{"keys":"ALL", "rows\_per\_partition":"NONE"}'

AND comment = ''

AND compaction = {'class': 'org.apache.cassandra.db.compaction.SizeTieredCompactionStrategy'}

AND compression = {'sstable\_compression': 'org.apache.cassandra.io.compress.LZ4Compressor'}

AND dclocal\_read\_repair\_chance = 0.1

AND default\_time\_to\_live = 0

AND gc\_grace\_seconds = 864000

AND max\_index\_interval = 2048

AND memtable\_flush\_period\_in\_ms = 0

AND min\_index\_interval = 128

AND read\_repair\_chance = 0.0

AND speculative\_retry = '99.0PERCENTILE';

CREATE TABLE killr\_video.videos\_by\_user (

user\_id uuid,

video\_id timeuuid,

preview\_thumbnails map<int, blob>,

tags set<text>,

title text,

type text,

PRIMARY KEY (user\_id, video\_id)

) WITH CLUSTERING ORDER BY (video\_id DESC)

AND bloom\_filter\_fp\_chance = 0.01

AND caching = '{"keys":"ALL", "rows\_per\_partition":"NONE"}'

AND comment = ''

AND compaction = {'class': 'org.apache.cassandra.db.compaction.SizeTieredCompactionStrategy'}

AND compression = {'sstable\_compression': 'org.apache.cassandra.io.compress.LZ4Compressor'}

AND dclocal\_read\_repair\_chance = 0.1

AND default\_time\_to\_live = 0

AND gc\_grace\_seconds = 864000

AND max\_index\_interval = 2048

AND memtable\_flush\_period\_in\_ms = 0

AND min\_index\_interval = 128

AND read\_repair\_chance = 0.0

AND speculative\_retry = '99.0PERCENTILE';

CREATE TABLE killr\_video.ratings\_by\_video (

video\_id timeuuid PRIMARY KEY,

num\_ratings counter,

sum\_ratings counter

) WITH bloom\_filter\_fp\_chance = 0.01

AND caching = '{"keys":"ALL", "rows\_per\_partition":"NONE"}'

AND comment = ''

AND compaction = {'class': 'org.apache.cassandra.db.compaction.SizeTieredCompactionStrategy'}

AND compression = {'sstable\_compression': 'org.apache.cassandra.io.compress.LZ4Compressor'}

AND dclocal\_read\_repair\_chance = 0.1

AND default\_time\_to\_live = 0

AND gc\_grace\_seconds = 864000

AND max\_index\_interval = 2048

AND memtable\_flush\_period\_in\_ms = 0

AND min\_index\_interval = 128

AND read\_repair\_chance = 0.0

AND speculative\_retry = '99.0PERCENTILE';

CREATE TABLE killr\_video.comments\_by\_video (

video\_id timeuuid,

posted\_timestamp timestamp,

user\_id uuid,

comment text,

preview\_thumbnails map<int, blob> static,

tags set<text> static,

title text static,

type text static,

PRIMARY KEY (video\_id, posted\_timestamp, user\_id)

) WITH CLUSTERING ORDER BY (posted\_timestamp DESC, user\_id ASC)

AND bloom\_filter\_fp\_chance = 0.01

AND caching = '{"keys":"ALL", "rows\_per\_partition":"NONE"}'

AND comment = ''

AND compaction = {'class': 'org.apache.cassandra.db.compaction.SizeTieredCompactionStrategy'}

AND compression = {'sstable\_compression': 'org.apache.cassandra.io.compress.LZ4Compressor'}

AND dclocal\_read\_repair\_chance = 0.1

AND default\_time\_to\_live = 0

AND gc\_grace\_seconds = 864000

AND max\_index\_interval = 2048

AND memtable\_flush\_period\_in\_ms = 0

AND min\_index\_interval = 128

AND read\_repair\_chance = 0.0

AND speculative\_retry = '99.0PERCENTILE';

# Import data:

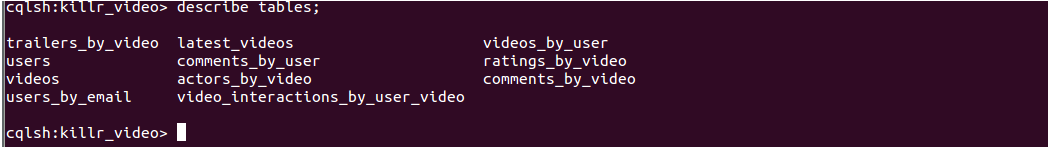
COPY videos FROM 'videos.csv' WITH HEADER=true;

COPY latest\_videos FROM 'latest\_videos.csv' WITH HEADER=true;

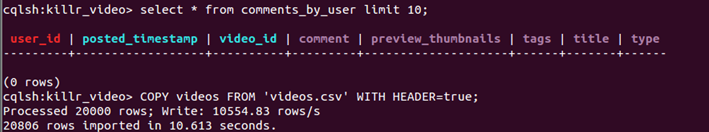
COPY trailers\_by\_video FROM 'trailers\_by\_video.csv' WITH HEADER=true;

COPY actors\_by\_video FROM 'actors\_by\_video.csv' WITH HEADER=true;

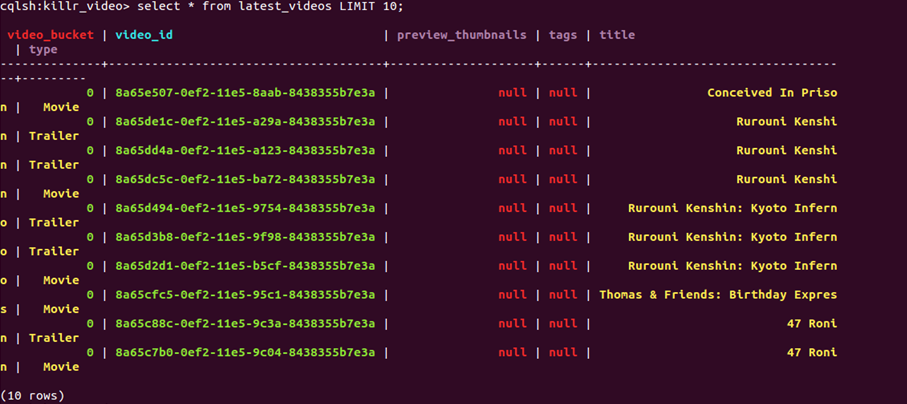
# Screens shots











# Delivered:

File Name : Cassandra.zip

Input file : Inputdata.zip

Output/DB Dump: killr\_video.tar.gz

Database Schema: schema\_describe

# Section 2: Database design using MongoDB

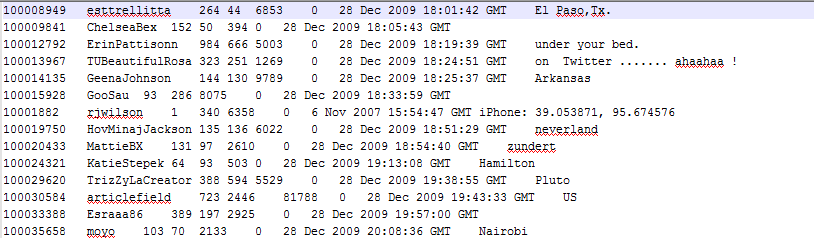
# Project Environment:

1. Red Hat Enterprise Linux Server release 5.9 (Tikanga)
2. mongodb-linux-x86\_64-rhel62-3.0.7
3. OpenSSL

# Installation of Mongo DB.

1. Install OpenSSL
   1. $ ./config
   2. $ make
   3. $ make test
   4. $ make install
2. Intall MongoDB
   1. tar –xvf mongodb-linux-x86\_64-rhel62-3.0.7

# Sample Raw data



# Data Cleanup (Reformat)

The raw txt file of user profiles is encoded in ISO-8859-1 format. This is a format that the MongoDB NoSQL store does not accept, a common problem. So you will need to convert the txt file into the UTF-8 format that MongoDB accepts. You need to do this before you can store the Twitter user profiles into the MongoDB database. Reformat the user profile twitter dataset from ISO-8859-1 to UTF-8 format by running the following reformatting script that is in your bin directory. Name the output file

reformat.sh <input file> <output file>



Double click on the file you created to bring the file up in the “gedit” editor. Add the following line as the first line to the newly reformatted Twitter data file (it becomes the “headerline”,something MongoDB understands). Be sure that you use tabs to split the fields.

user\_id user\_name friend\_count follower\_count status\_count favorite\_count

account\_age user\_location

**Create database, table and import data.**

The tab-separated values (tsv) file can be imported directly into MongoDB, however, proper header lines (fields) must be defined so that MongoDB can give structure to the data when converting it to its internal format. Next you will import the data into MongoDB. To do so, run the following script which exists in your bin directory. Note that <import file type> is tsv. Choose your own <db name> and <collection name>.

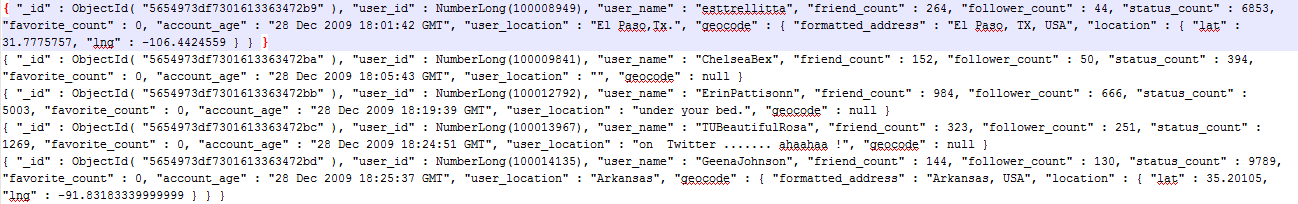
./bin/import\_mangodb.sh <db name> <collection name> <import file type>

<import file>



The above script contains mongoDB script “mongoimport --db $1 --collection $2 --type $3 --headerline --file $4“

Here is the sample JSON data from the exported database



# Query and Update

**Query and Update the User Profile Collection: QueryAndUpdate.sh**

The Twitter user profile permits a Twitter user to input arbitrary text as their location meaning user locations may be fabricated. Through the QueryAndUpdate.sh tool you will access the Google geocoding API to validate user locations and extract valid Latitude/Longitude of the user locations. If you are interested in what the Google geocoding API does, take a look here <https://developers.google.com/maps/documentation/geocoding>

Simple but workable software for doing the geocoding is QueryAndUpdate.sh. It’s a script, but you should peek at the Java code that the script invokes to see how it works. The Java code is at src/google/GeoCodingClient.java (see tree structure above). QueryAndUpdate.sh allows you to specify an authentication option in the configuration file that you can find in the config directory (see tree structure above). While Google provides three authentication options, you will use the anonymous user option:

Anonymous user: limited to making 2500 geocoding queries per day. To use this option, leave all authentication configuration parameters blank. This means you will need to run your tool 4 times over 4 days to finish geocoding all 10,000 user profiles. This workaround is simpler than the other authentication options.

QueryAndUpdate.sh <configuration file> <db name> <collection name> <query criteria file> <log file>





A sample of the geocode information that is added by the Google geocoding service is given below

{

"geocode" : {

"formatted\_address" : "Noel N5, Kitimat-Stikine D, BC V0J, Canada",

"location" : { "lat" : 57.4755555, "lng" : -132.3597222 } }}

**Other Mongodb queries:**

1. Access MongoDB

$ mongo

> use twittercollection

switched to db twittercollection

1. Find a user

>db. twittercollection.find({user\_id:100008949})

Result

{ "\_id" : ObjectId("5415fc01d77bc408f1397df5"), "user\_id" : NumberLong(100008949),"user\_name" : "esttrellitta", "friend\_count" : 264, "follower\_count" : 44, "status\_count" : 6853, "favorite\_count" : 0, "account\_age" : "28 Dec 2009 18:01:42 GMT", "user\_location" : "El Paso,Tx." }

1. Update user

db.twittercollection.update({user\_id:100008949},{$set: {geolocation :{formatted\_address: "El Paso, TX, USA", location:{lat: 31.7775757, lng:-106.6359219}}}})

1. Verify update

db.twittercollection.find({user\_id:100008949})

Result

{ "\_id" : ObjectId("5415fc01d77bc408f1397df5"), "user\_id" : NumberLong(100008949), "user\_name" : "esttrellitta", "friend\_count" : 264, "follower\_count" : 44, "status\_count" : 6853, "favorite\_count" : 0, "account\_age" : "28 Dec 2009 18:01:42 GMT", "user\_location" : "El Paso,Tx.", "geolocation" : { "formatted\_address" : "El Paso, TX, USA", "location" : { "lat" :31.7775757, "lng" : -106.6359219 } } }

1. Find user who has no Geocode

{"geocode": {"$exists": false}}

1. Find the user whose geocode is null

db.twittercollection.find({"geocode": {"$eq": Null}}).count()

1. Find the user whose geocode is null

db.twittercollection.find({"geocode": {"$ne": Null}}).count()

User detail who has geocode is null

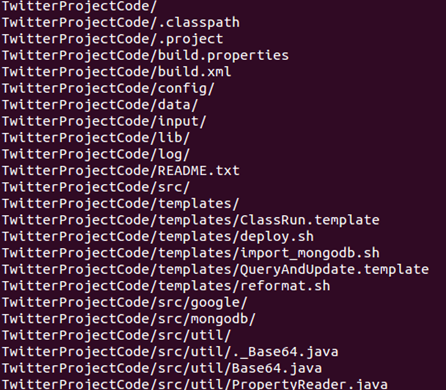
db.twittercollection.find({"geocode": {"$eq": Null}}).Pretty()

Additional reference for the query criteria is here:

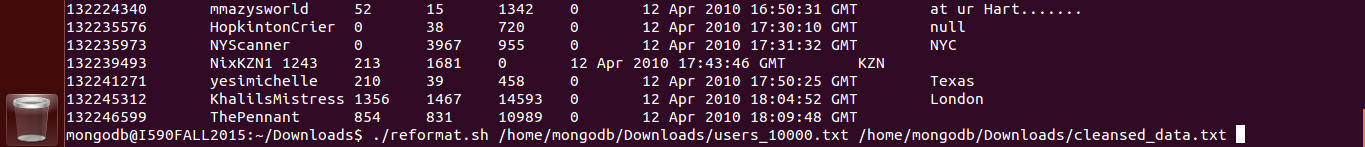
<http://docs.MongoDB.org/manual/core/crud-introduction/#query>

# Screenshots

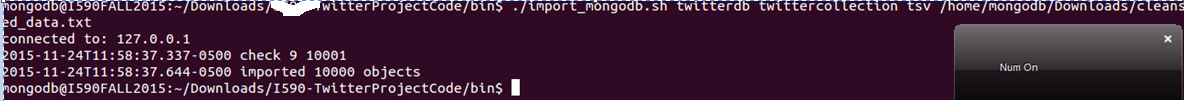
Code and configuration file structure

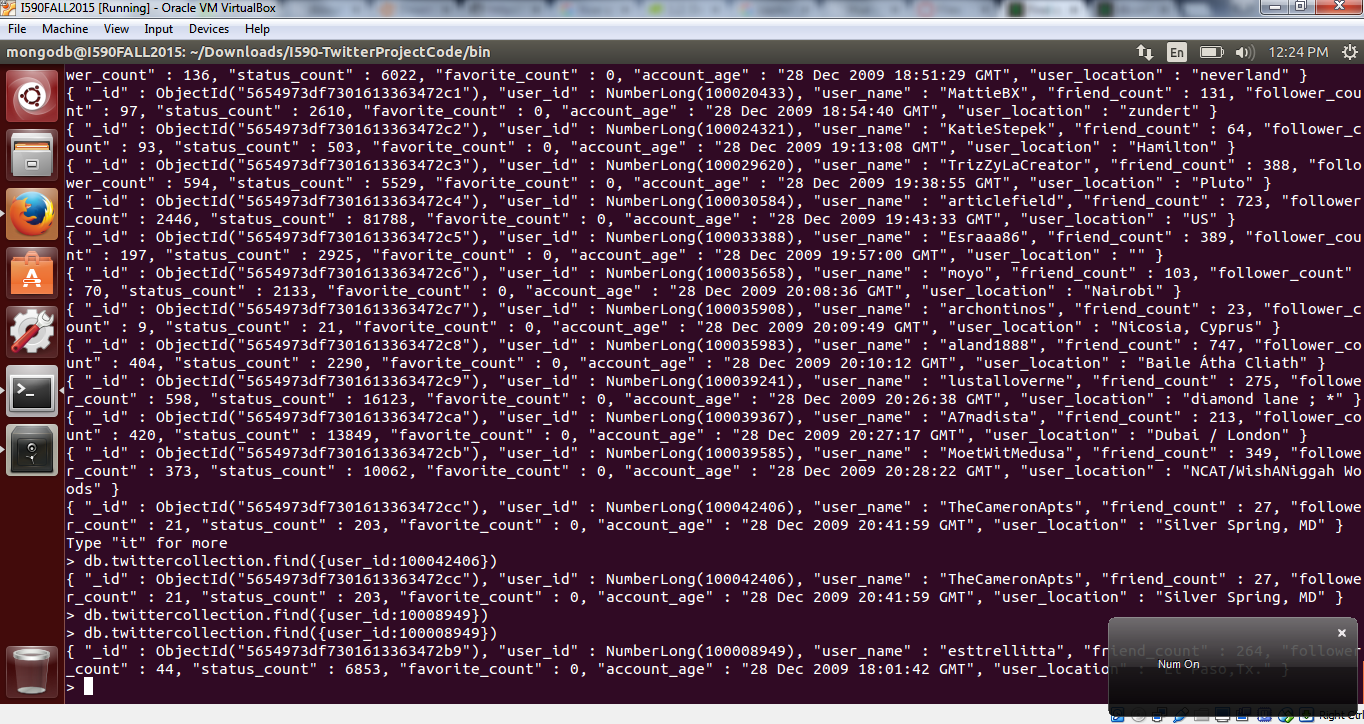


Data clean-up using script.

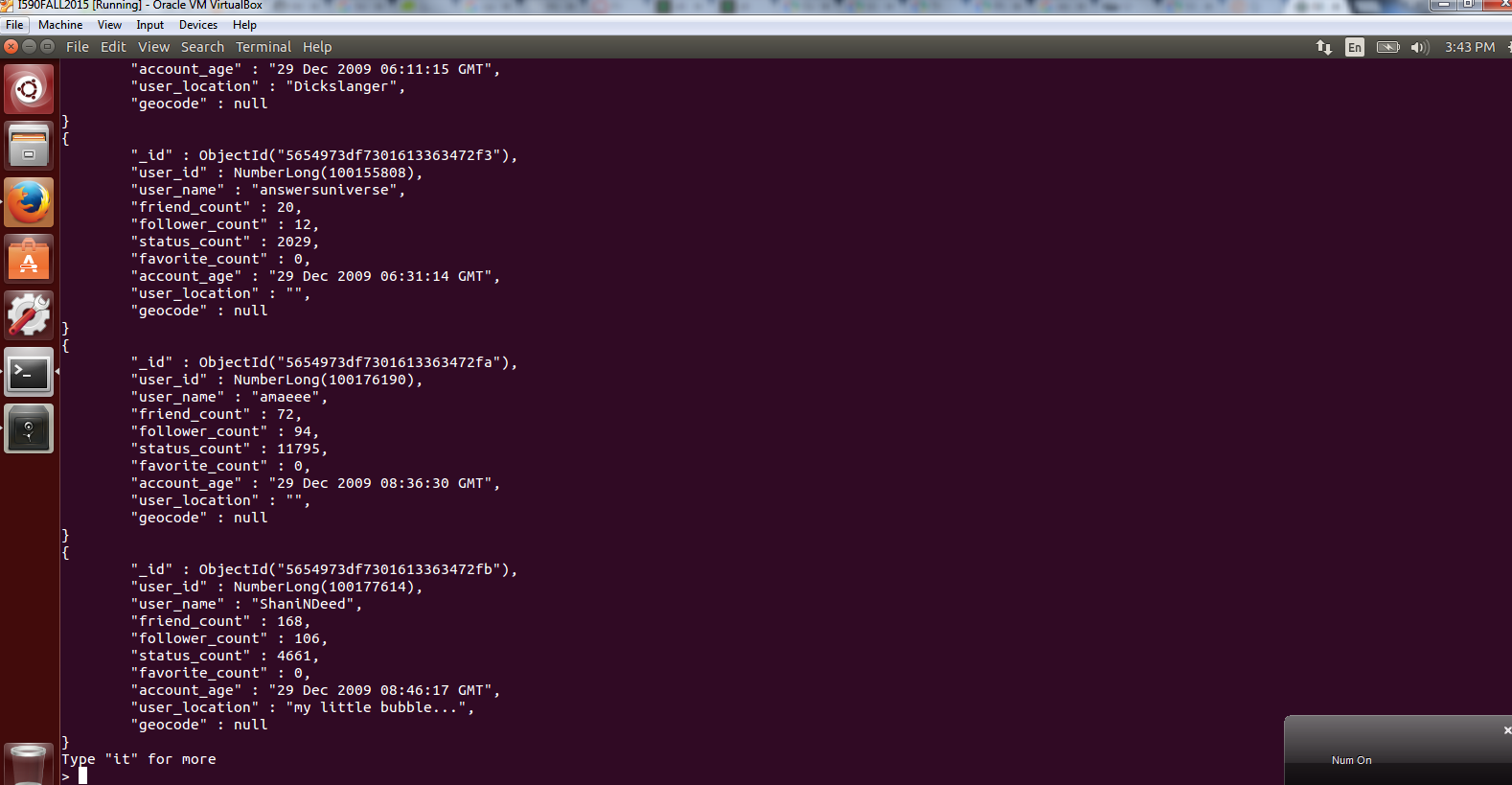


Importing of data into Mongo db

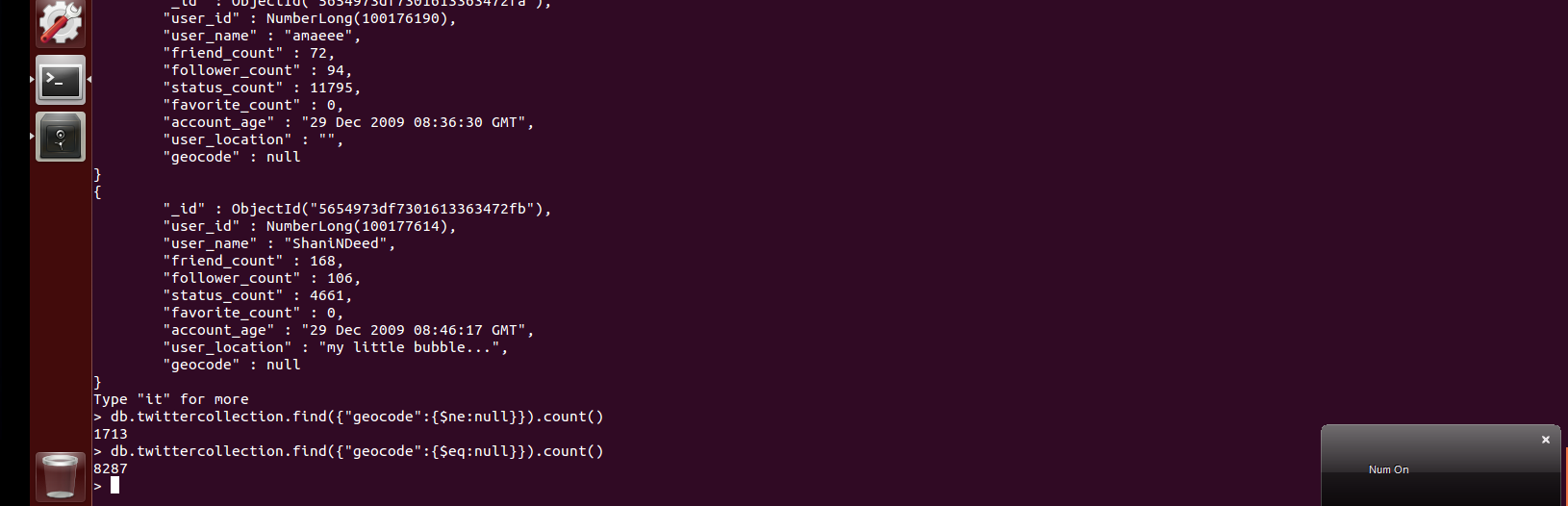


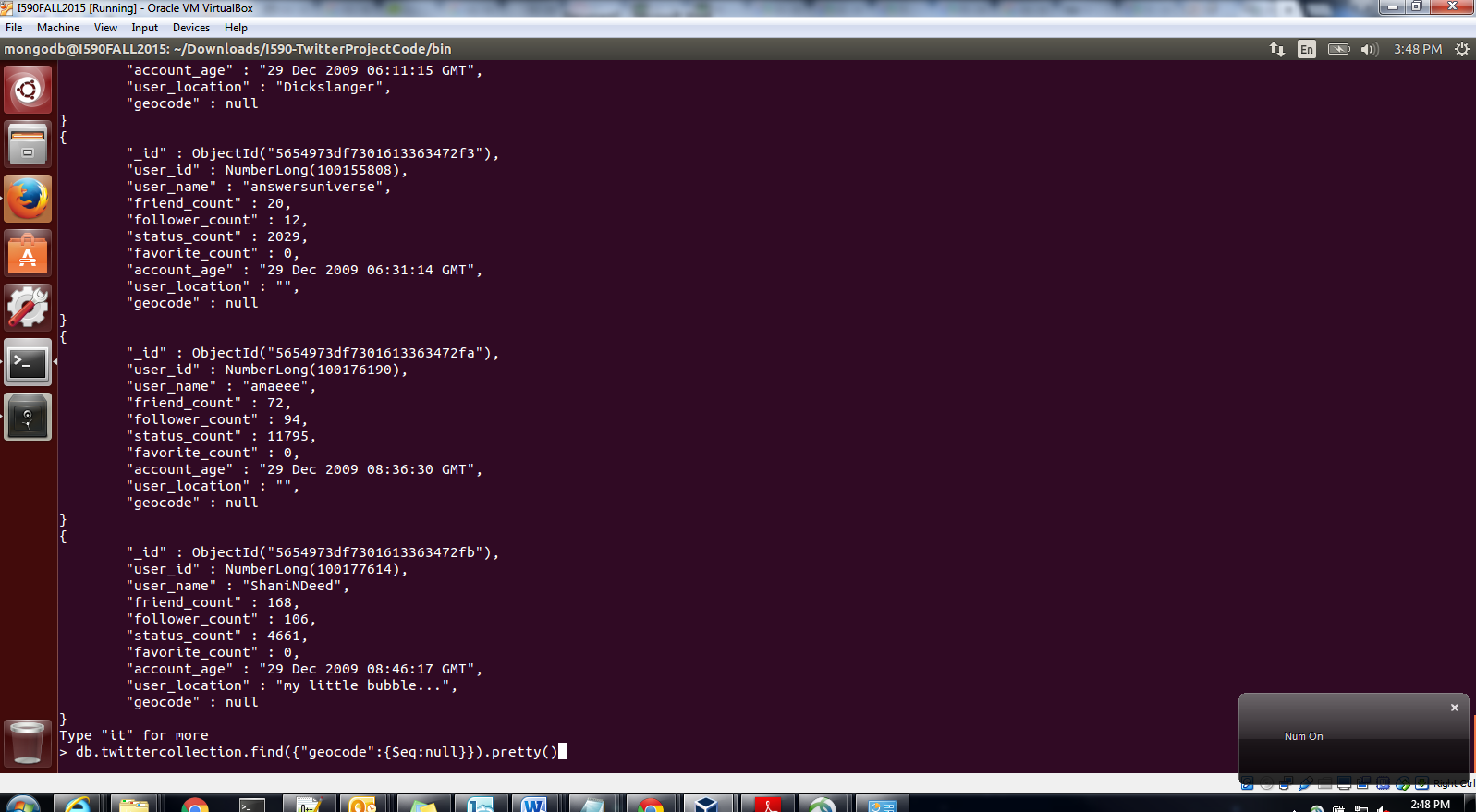


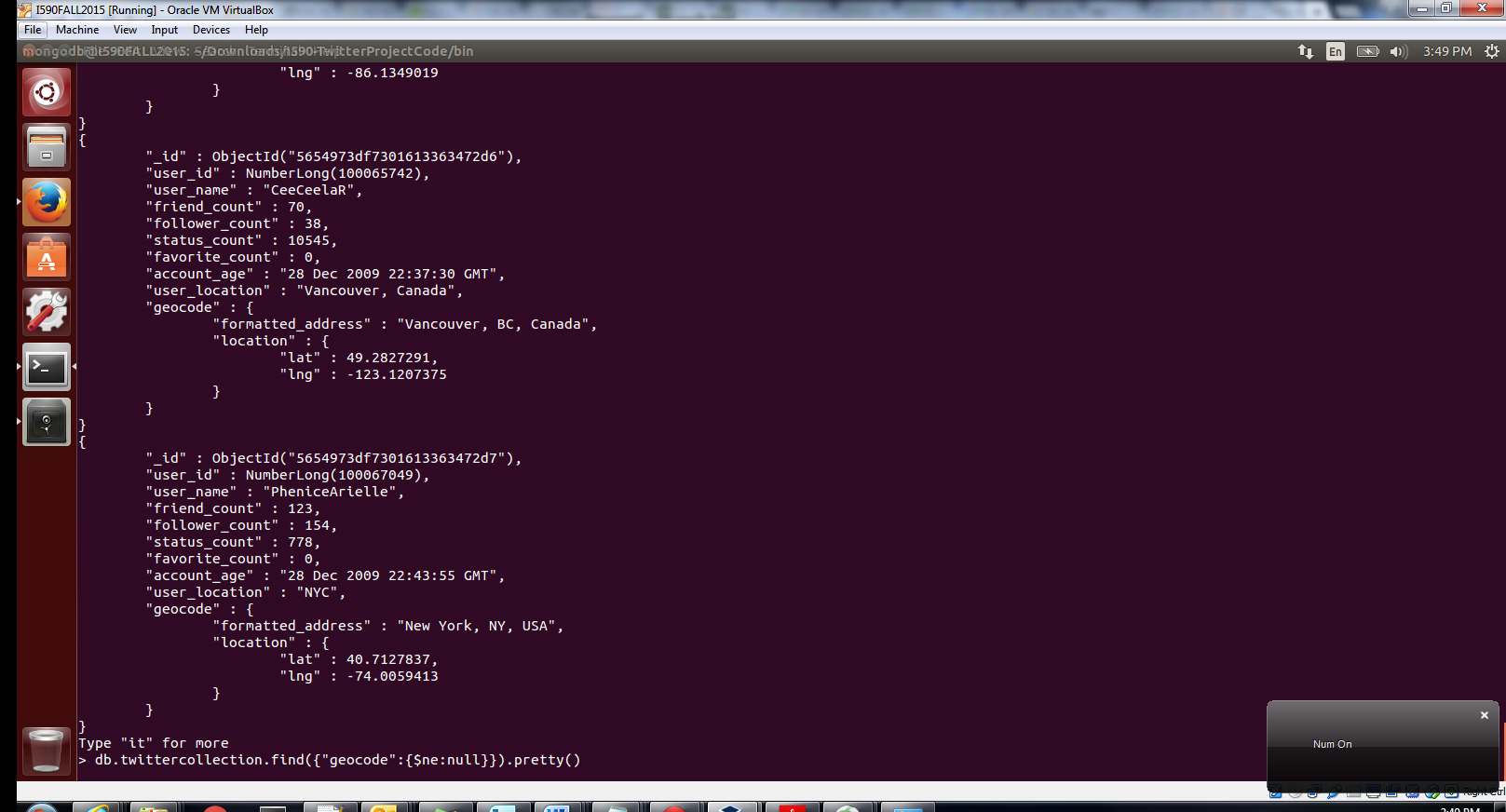
Users with no location



Users with location and no location count







# Delivered:

File Name: mongodb.zip

1. Shell script
2. Mogodb Scripts
3. Input Data
4. Output data/DB dump
5. Java code, library files, config files