database server - mysql, mongo
make sure the server is running
(if it's not running, it doesn't know how to talk)

database sits inside databases server
cannot directly access database without dbms
go to your services settings or type in cmd line to have it started

NoSQL Data Processing

MySQL = learning to parse upfront, then scrub it when you put it into the table easily

W3 schools is a good resource for Mongo

hey these slides kinda suck

Outline

- Introduction to NoSQL
- Getting started with MongoDB
- Python NoSQL data processing
- Data aggregation

NoSQL databases

- NoSQL, Not only SQL, is a term used in several technologies where the nature of data does not require a relational model.
  - Hugh quantity of data
  - Higher availability
  - Scalability
  - Performance

```
data = x.json()
firstname = data['name']
address = data['address']
```

parsing later on
need to know how to parse them based on your code
spits back out into a dictionary format

NoSQL databases

- Types of NoSQL stores:
  - **Document** store: data are organized as a collection of documents, e.g. MongoDB, CouchDB;
  - **Key-value** store: data are stored as key-value pairs without predefined schema, e.g. Apache Cassandra, Dynamo, Hbase, Amazon SimpleDB;
  - **Graph**-based store: data are stored in graph structures with nodes, edges, and properties, e.g. Neo4j, InfoGrid, Horton.

http://nosql-database.org/

MongoDB

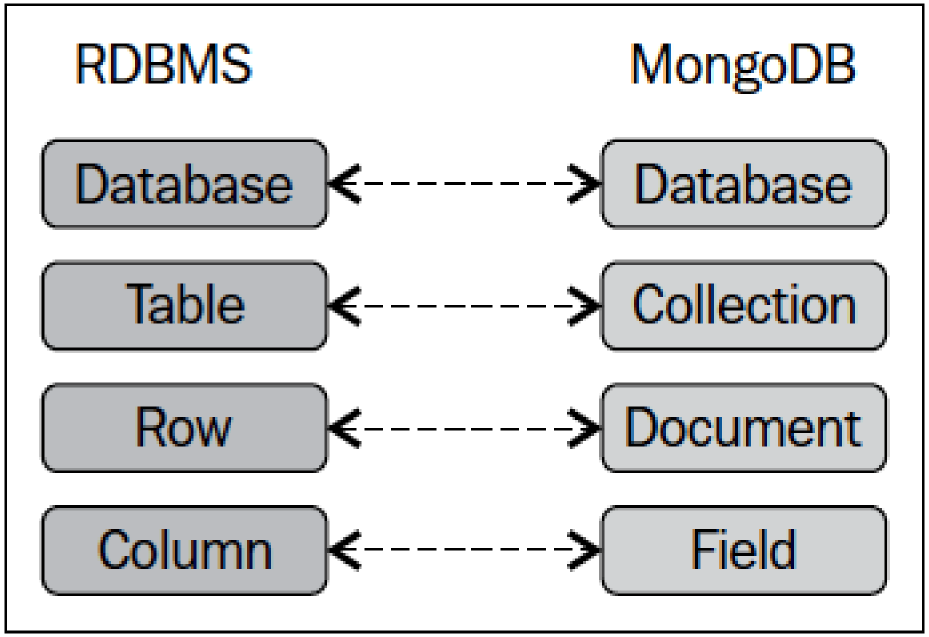
- MongoDB
  - NoSQL database
  - Document-oriented database
  - High performance for storage and retrieval
  - BSON (binary JSON) storage
  - Supports ad hoc queries, replication, load balancing, aggregation, map-reduce, etc.

features that allows you to run cluster, load balance data across multiple mongo databases, map-reduce = pointer pointing from one location to another

MongoDB

- MongoDB compared to relational databases:

relational database
like mySQL

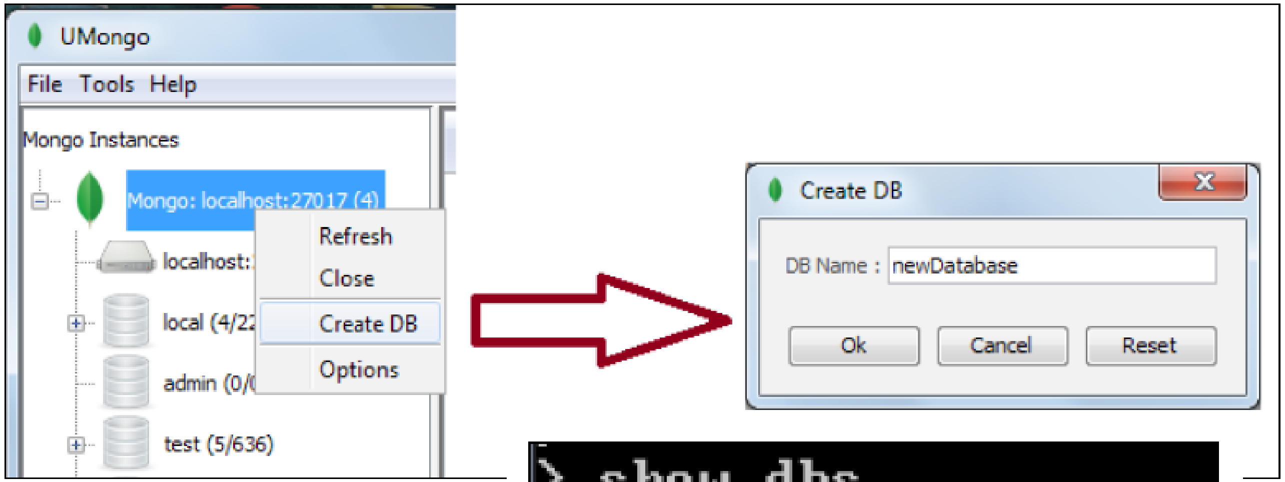


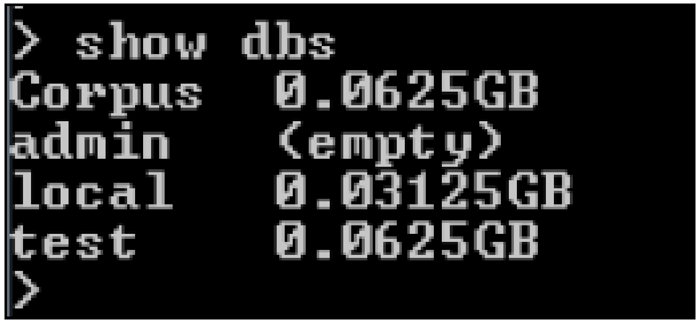
RDBMS	MongoDB
Database	Database
Table	Collection
Row	Document
Column	Field

MongoDB

compass: similar to workbench

- MongoDB database
  - A physical container for document collections;

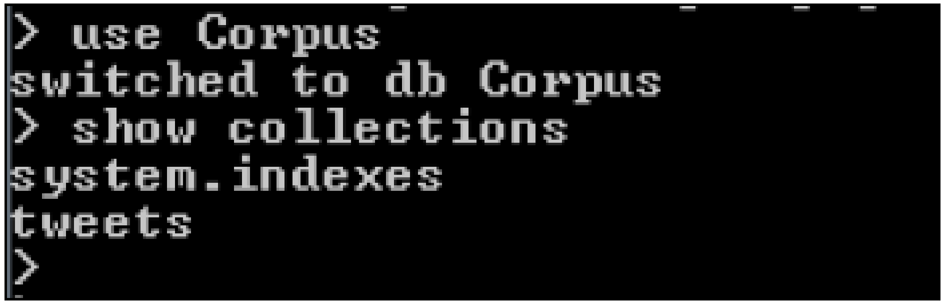




```
> show dbs
Corpus   0.0625GB
admin    (empty)
local    0.03125GB
test     0.0625GB
>
```

MongoDB

- MongoDB collection, a group of documents



```
> use Corpus
switched to db Corpus
> show collections
system.indexes
tweets
>
```

  - Collections can be split to distribute documents across MongoDB instances (shards)
  - Sharding for horizontal scaling

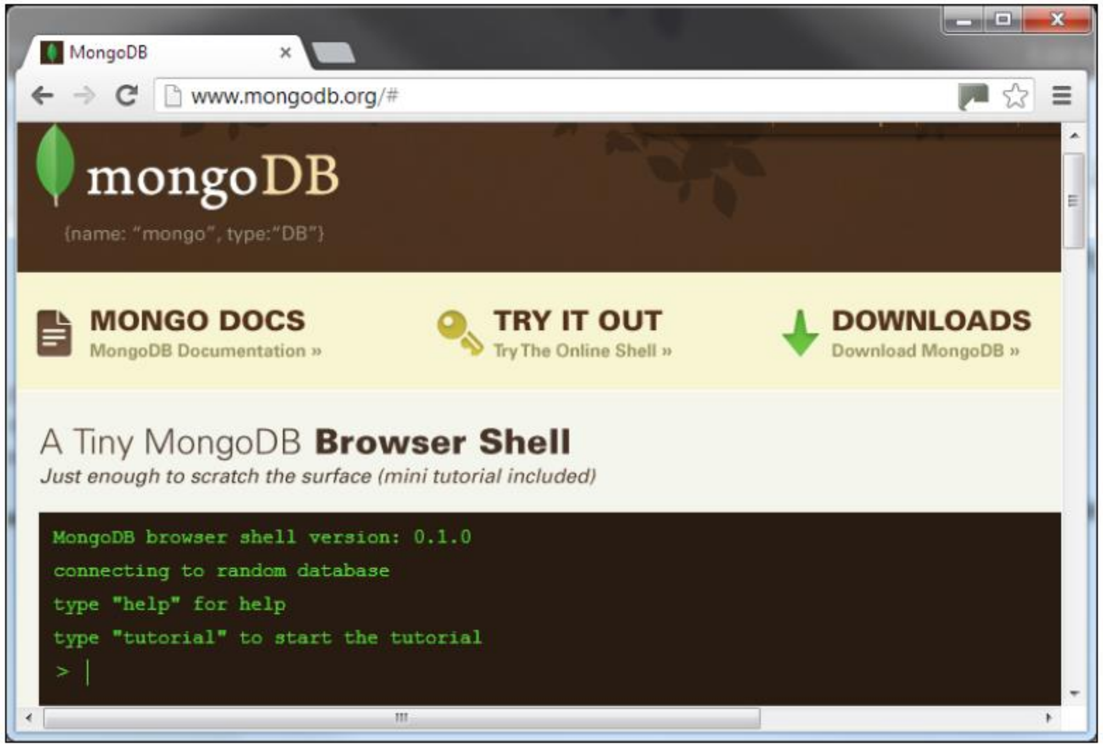
MongoDB

- MongoDB document, is a record and implements a schema-less model
  - Documents do not have to have same fields
  - Format similar to JSON (Javascript Object Notation)
  - Binary representation with BSON

no need to create fields and stuff (benefit of having mongo), it doesn't care, it's dynamically expandable

Mongo shell

- Mongo shell is an interactive console
  - Comes as a standard feature with MongoDB
  - Online / in-browser shell



A Tiny MongoDB **Browser Shell**

Just enough to scratch the surface (mini tutorial included)

```
MongoDB browser shell version: 0.1.0
connecting to random database
type "help" for help
type "tutorial" to start the tutorial
> |
```

Mongo shell

$ something has a specific meaning or reserved keyword

- Insert / Update / Delete

specify which database you're referring to

Insert method in MongoDB:

```
db.collection.insert({ name: { first: 'Jan', last: 'Smith' } )
```

Update method in MongoDB:

update specific parameters into it

```
db.collection.update(
  { 'name.first': 'Jan' },
  { $set: { 'name.first': 'Joan' } }
)
```

w3 schools thing here (update)

$set = noSQL cmd to update a field and sets it; can have an entire set of data in it, $ = insert entire set OR retrieves something to change; the key is upsert() true

Delete method in MongoDB:

```
db.collection.remove( { 'name.first' : 'Jan' }, safe=True )
```

http://docs.mongodb.org/manual/crud/

Mongo shell

- Queries to retrieve data

  Selecting all elements from the collection in MongoDB:

  ```
  db.collection.find()
  ```

  Getting the number of documents retrieved by a query with MongoDB:

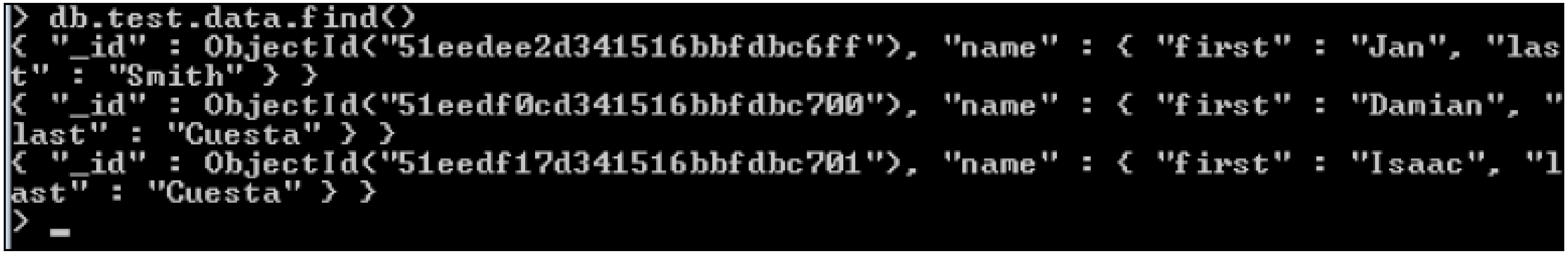
  ```
  db.collection.find().count()
  ```

  Query with a specific criteria with MongoDB:

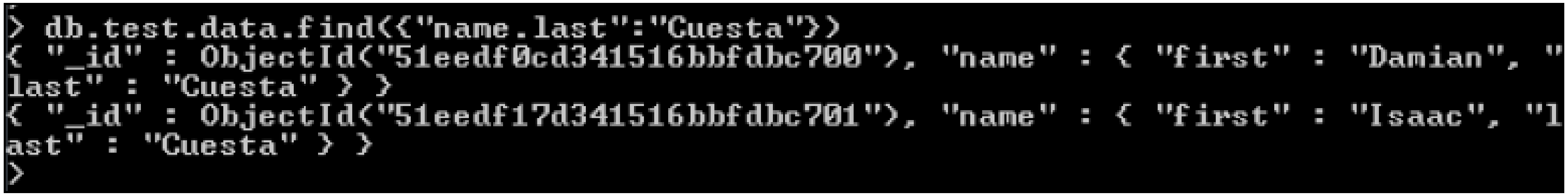
  ```
  db.collection.find({"name.last":"Cuesta"})
  ```

http://docs.mongodb.org/manual/crud/

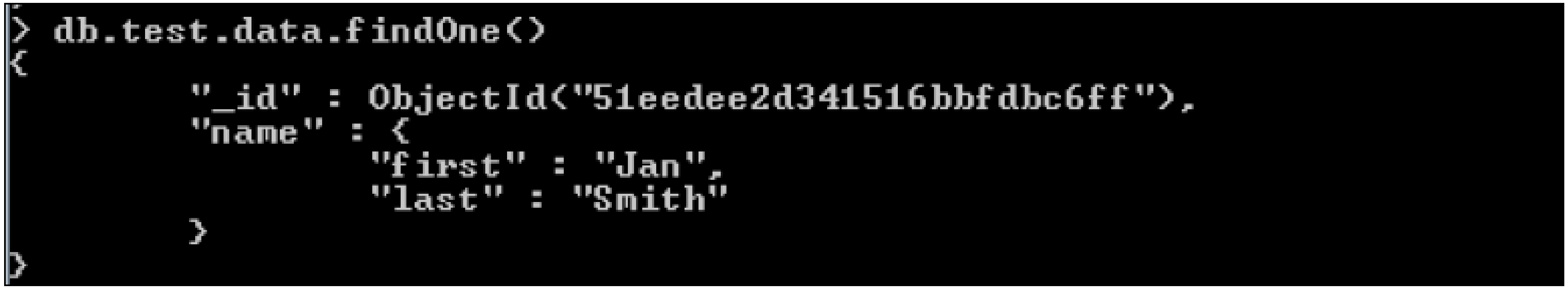
Mongo shell



```
> db.test.data.find()
{ "_id" : ObjectId("51eedee2d341516bbfdbc6ff"), "name" : { "first" : "Jan", "las
t" : "Smith" } }
{ "_id" : ObjectId("51eedf0cd341516bbfdbc700"), "name" : { "first" : "Damian", "
last" : "Cuesta" } }
{ "_id" : ObjectId("51eedf17d341516bbfdbc701"), "name" : { "first" : "Isaac", "l
ast" : "Cuesta" } }
> _
```



```
> db.test.data.find({"name.last":"Cuesta"})
{ "_id" : ObjectId("51eedf0cd341516bbfdbc700"), "name" : { "first" : "Damian", "
last" : "Cuesta" } }
{ "_id" : ObjectId("51eedf17d341516bbfdbc701"), "name" : { "first" : "Isaac", "l
ast" : "Cuesta" } }
>
```



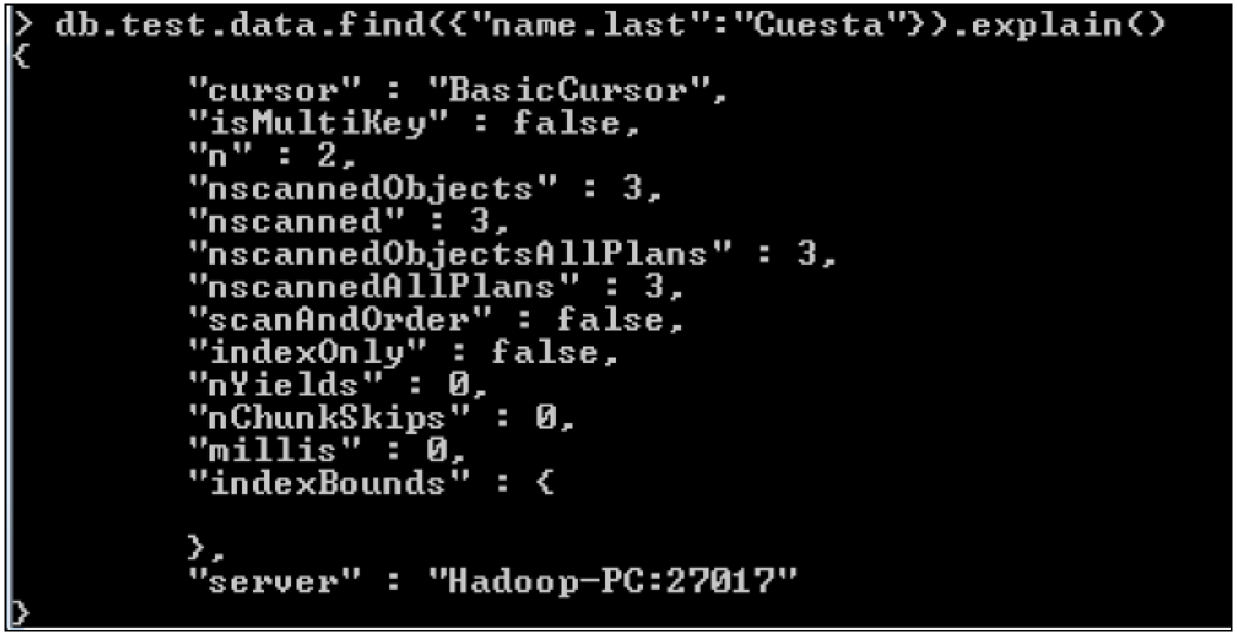
```
> db.test.data.findOne()
{
        "_id" : ObjectId("51eedee2d341516bbfdbc6ff"),
        "name" : {
                "first" : "Jan",
                "last" : "Smith"
        }
}
```

http://docs.mongodb.org/manual/crud/

Mongo shell

- The explain method to test query operation, timing, etc.

  db.collection.find({"name.last":"Cuesta"}).explain()



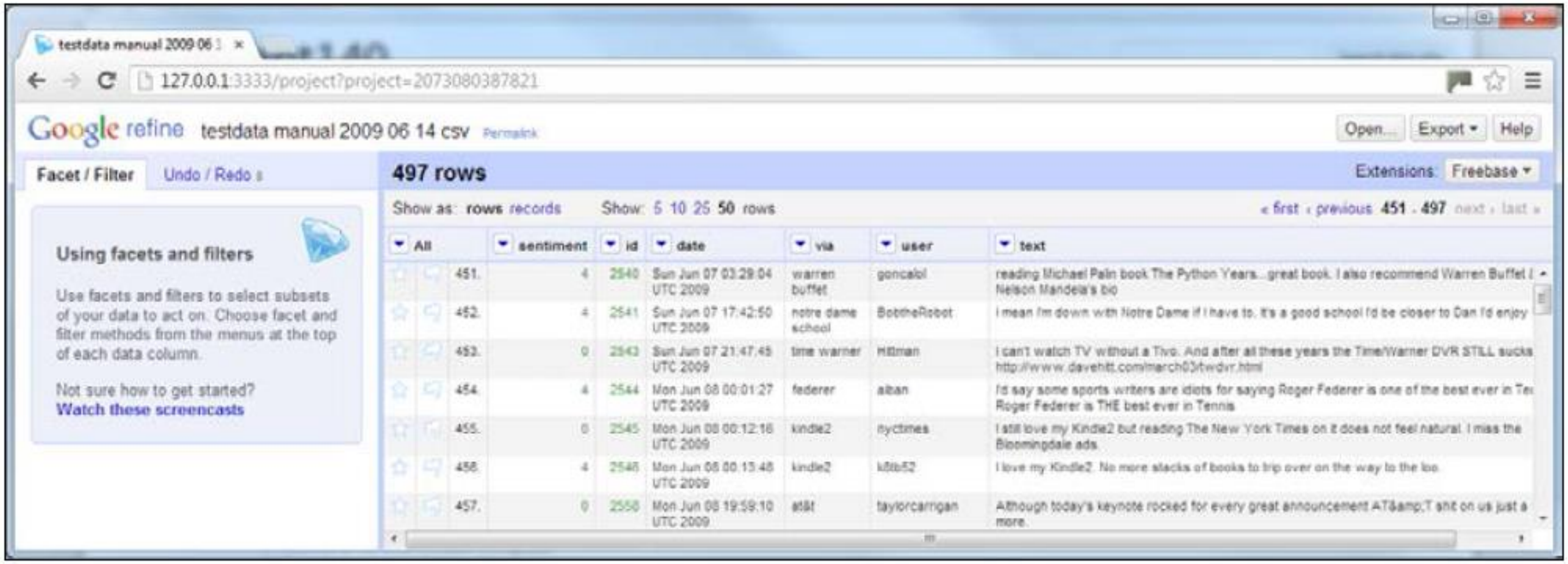
```
> db.test.data.find({"name.last":"Cuesta"}).explain()
{
        "cursor" : "BasicCursor",
        "isMultiKey" : false,
        "n" : 2,
        "nscannedObjects" : 3,
        "nscanned" : 3,
        "nscannedObjectsAllPlans" : 3,
        "nscannedAllPlans" : 3,
        "scanAndOrder" : false,
        "indexOnly" : false,
        "nYields" : 0,
        "nChunkSkips" : 0,
        "millis" : 0,
        "indexBounds" : {

        },
        "server" : "Hadoop-PC:27017"
}
```

Data Transformation With OpenRefine

OpenRefine

- Import a csv file with OpenRefine:



testdata manual 2009 06 1

127.0.0.1:3333/project?project=2073080387821

Google refine testdata manual 2009 06 14 csv Permalink

Open... Export ▾ Help

Facet / Filter Undo / Redo

Using facets and filters

Use facets and filters to select subsets of your data to act on. Choose facet and filter methods from the menus at the top of each data column.

Not sure how to get started?
Watch these screencasts

497 rows

Extensions: Freebase ▾

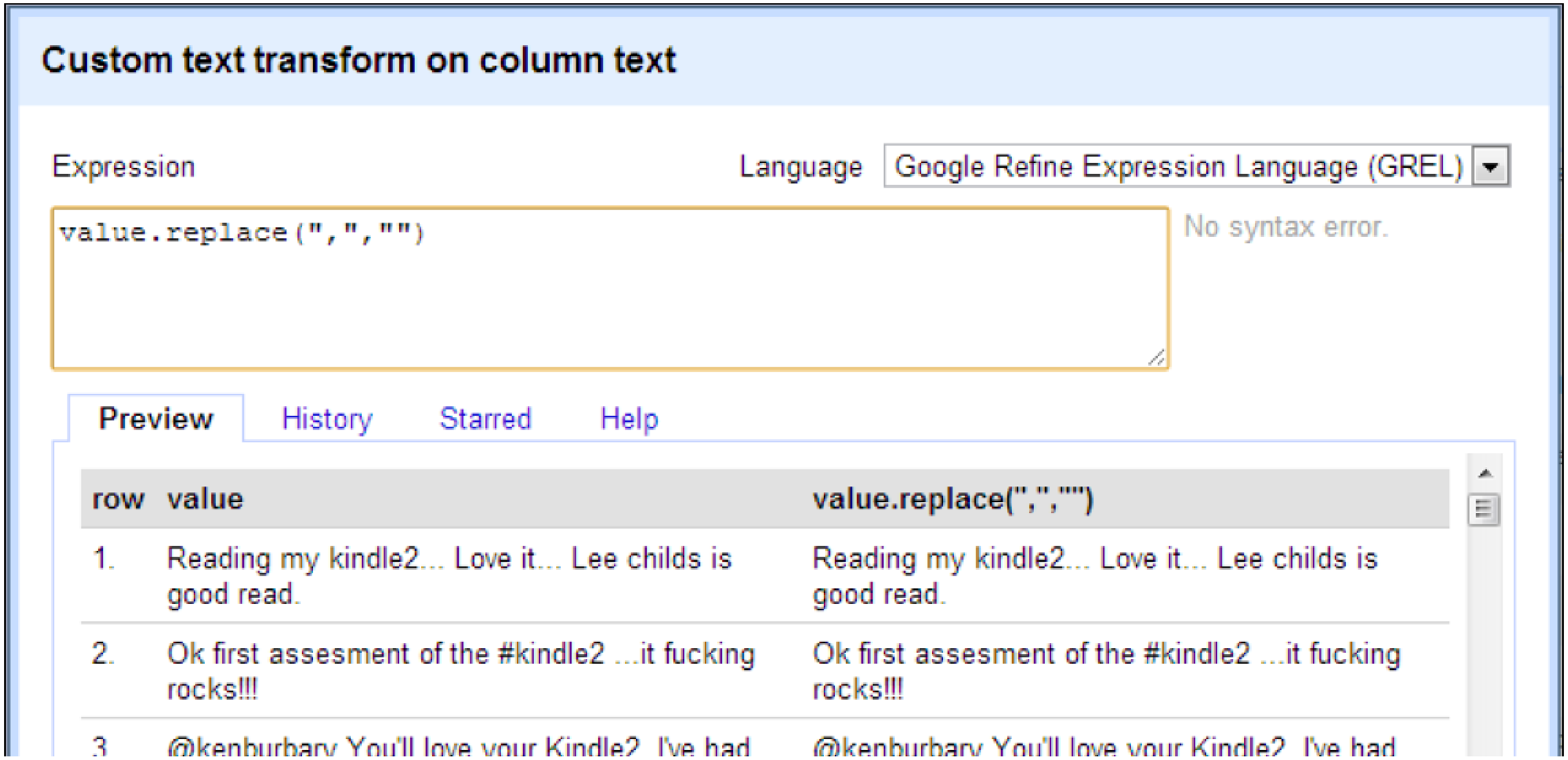
Show as: **rows** records Show: 5 10 25 **50** rows « first ‹ previous **451 - 497** next › last »

All	sentiment	id	date	via	user	text
451.	4	2540	Sun Jun 07 03:29:04 UTC 2009	warren buffet	goncalol	reading Michael Palin book The Python Years...great book. I also recommend Warren Buffet & Nelson Mandela's bio
452.	4	2541	Sun Jun 07 17:42:50 UTC 2009	notre dame school	BobtheRobot	I mean I'm down with Notre Dame if I have to. It's a good school I'd be closer to Dan I'd enjoy
453.	0	2543	Sun Jun 07 21:47:45 UTC 2009	time warner	Hillman	I can't watch TV without a Tivo. And after all these years the Time/Warner DVR STILL sucks http://www.davehitt.com/march03/twdvr.html
454.	4	2544	Mon Jun 08 00:01:27 UTC 2009	federer	alban	I'd say some sports writers are idiots for saying Roger Federer is one of the best ever in Te Roger Federer is THE best ever in Tennis
455.	0	2545	Mon Jun 08 00:12:16 UTC 2009	kindle2	nyctimes	I still love my Kindle2 but reading The New York Times on it does not feel natural. I miss the Bloomingdale ads.
456.	4	2546	Mon Jun 08 00:13:48 UTC 2009	kindle2	k8tb52	I love my Kindle2. No more stacks of books to trip over on the way to the loo.
457.	0	2558	Mon Jun 08 19:59:10 UTC 2009	at&t	taylorcarrigan	Although today's keynote rocked for every great announcement AT&amp;T shit on us just a more.

OpenRefine

is it the proper JSON format from any sources?
it depends. may need to scrub the data before you use it, this program is one of them where you can replace all or pecific spots before getting rid of the slash (every dict has a slash before/after it?)

- Replace values:



Custom text transform on column text

Expression   Language: Google Refine Expression Language (GREL)

```
value.replace(",","")
```

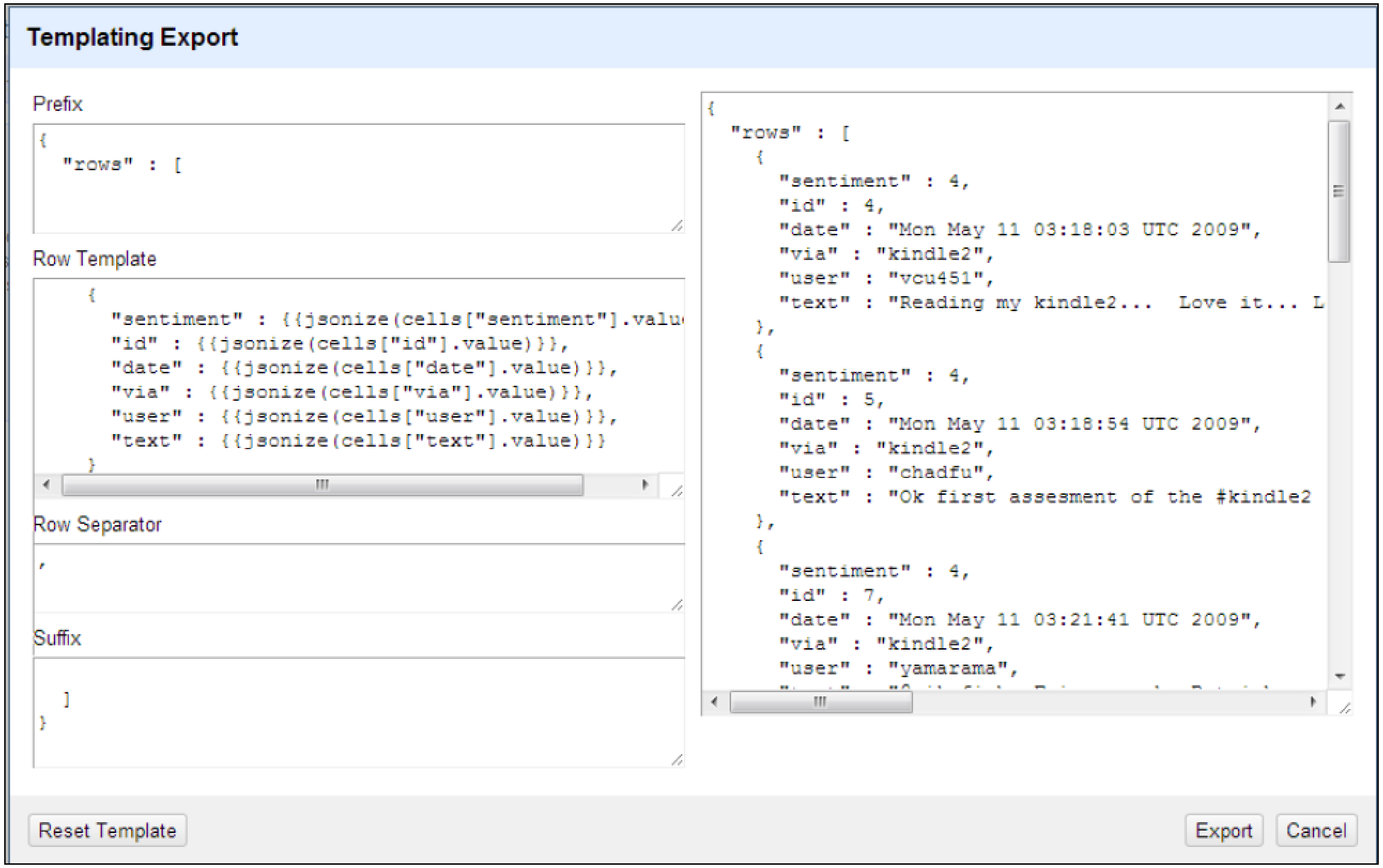
No syntax error.

Preview | History | Starred | Help

row	value	value.replace(",","")
1.	Reading my kindle2... Love it... Lee childs is good read.	Reading my kindle2... Love it... Lee childs is good read.
2.	Ok first assesment of the #kindle2 ...it fucking rocks!!!	Ok first assesment of the #kindle2 ...it fucking rocks!!!
3	@kenburbary You'll love your Kindle2. I've had	@kenburbary You'll love your Kindle2. I've had

OpenRefine

- Export to a JSON format:



```
{
  "rows" : [
```

```
    {
      "sentiment" : {{jsonize(cells["sentiment"].valu
      "id" : {{jsonize(cells["id"].value)}},
      "date" : {{jsonize(cells["date"].value)}},
      "via" : {{jsonize(cells["via"].value)}},
      "user" : {{jsonize(cells["user"].value)}},
      "text" : {{jsonize(cells["text"].value)}}
    }
```

```
,
```

```
  ]
}
```

```
{
  "rows" : [
    {
      "sentiment" : 4,
      "id" : 4,
      "date" : "Mon May 11 03:18:03 UTC 2009",
      "via" : "kindle2",
      "user" : "vcu451",
      "text" : "Reading my kindle2...  Love it... L
    },
    {
      "sentiment" : 4,
      "id" : 5,
      "date" : "Mon May 11 03:18:54 UTC 2009",
      "via" : "kindle2",
      "user" : "chadfu",
      "text" : "Ok first assesment of the #kindle2
    },
    {
      "sentiment" : 4,
      "id" : 7,
      "date" : "Mon May 11 03:21:41 UTC 2009",
      "via" : "kindle2",
      "user" : "yamarama",
```

Python MongoDB Processing

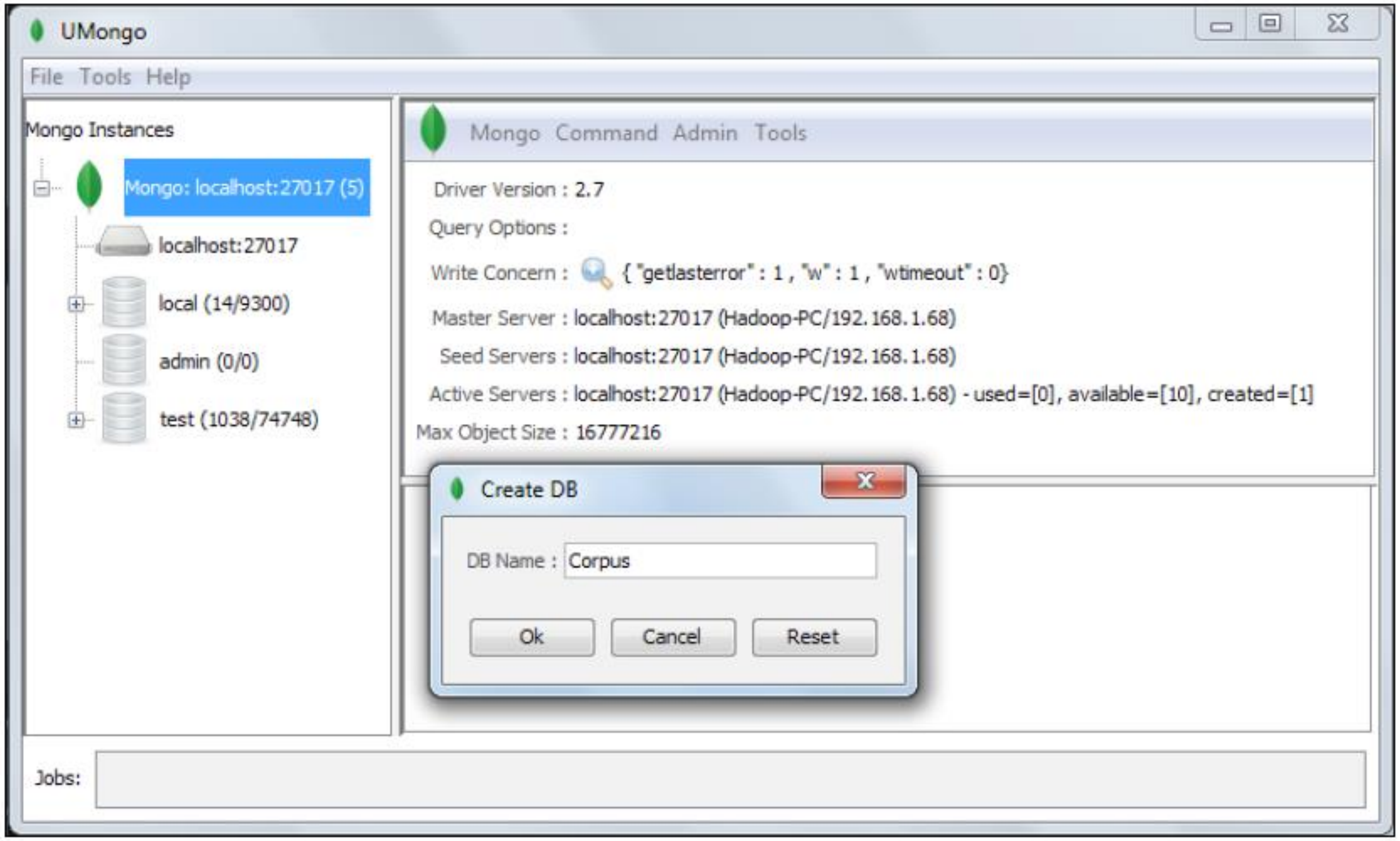
PyMongo

- Umongo as GUI client for MongoDB
- With Python:
  - PyMongo module to access MongoDB
  - json module to process data

```
import json
from pymongo import MongoClient
```

PyMongo

- Create DB with Umongo: named *Corpus*



Driver Version : 2.7
Query Options :
Write Concern : { "getlasterror" : 1 , "w" : 1 , "wtimeout" : 0}
Master Server : localhost:27017 (Hadoop-PC/192.168.1.68)
Seed Servers : localhost:27017 (Hadoop-PC/192.168.1.68)
Active Servers : localhost:27017 (Hadoop-PC/192.168.1.68) - used=[0], available=[10], created=[1]
Max Object Size : 16777216

DB Name : Corpus

PyMongo

- Connect to MongoDB with PyMongo:
  - Connect to MongoDB
  - Select the Corpus database
  - Select the tweets collection

```
con = MongoClient()
db = con.Corpus
tweets = db.tweets
```

PyMongo

- To load and insert data from file:
  - Load data from text file (prev tweet data in JSON)

```
with open("test.txt") as f:
    data = json.loads(f.read())
```

  - Iterate the rows and insert them into collection

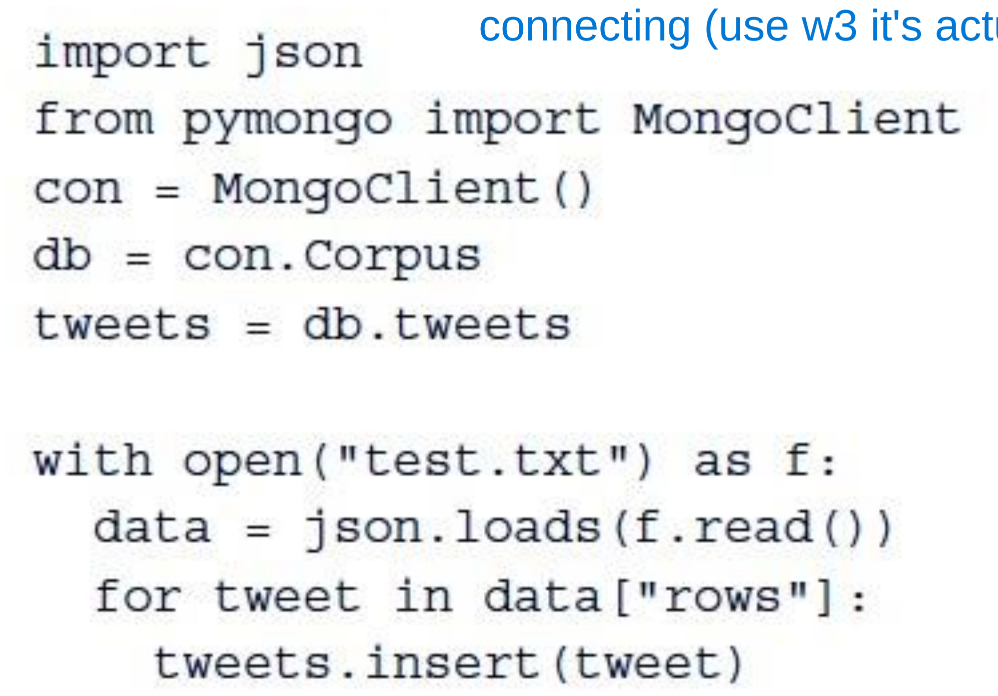
```
for tweet in data["rows"]:
    tweets.insert(tweet)
```

PyMongo

translates your statements into something actual, still has that skeleton that we're looking for

- Complete code:

connecting (use w3 it's actually easier than this)

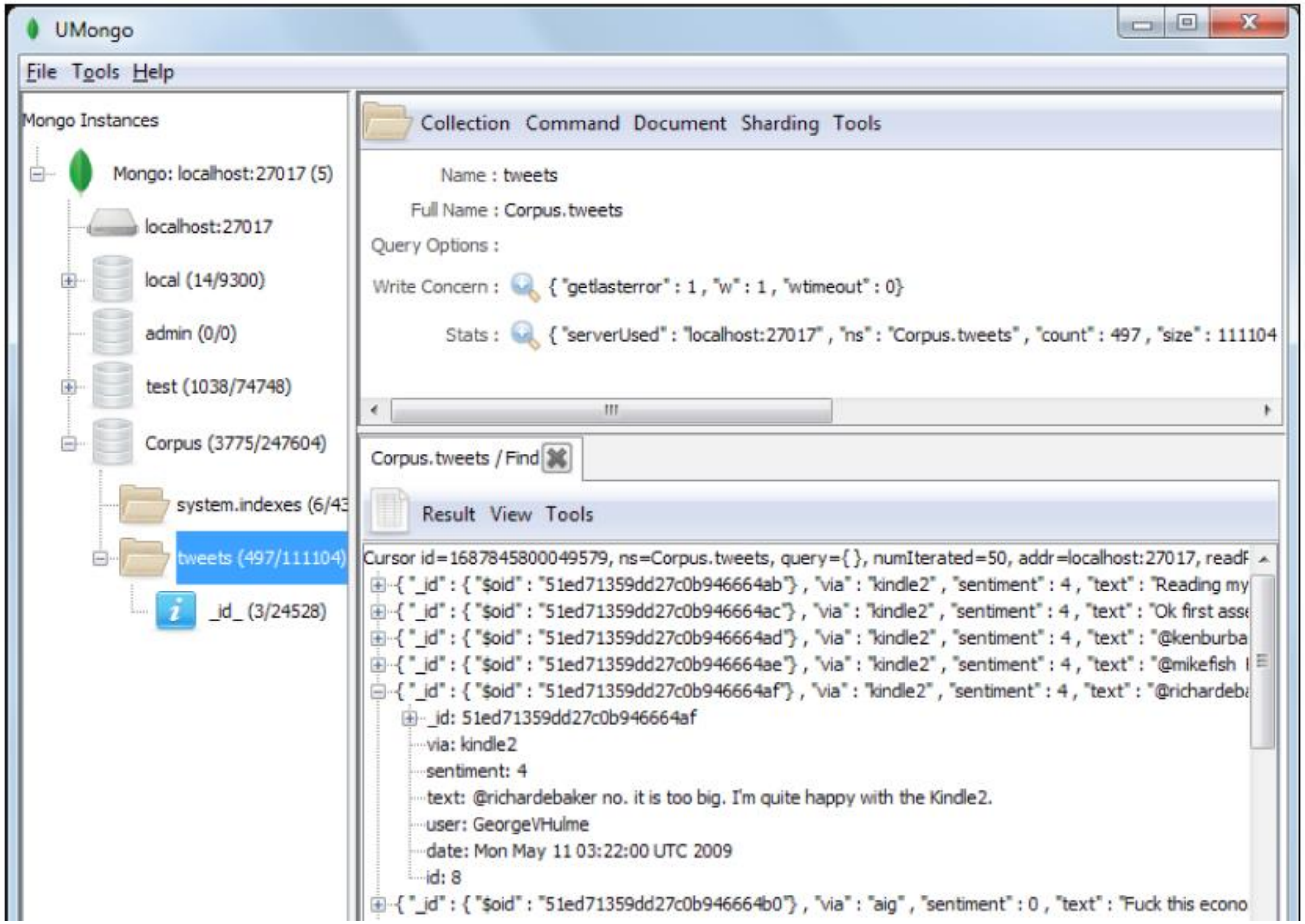


```
import json
from pymongo import MongoClient
con = MongoClient()
db = con.Corpus
tweets = db.tweets

with open("test.txt") as f:
  data = json.loads(f.read())
  for tweet in data["rows"]:
    tweets.insert(tweet)
```

PyMongo

- View results in DB with UMongo



```
Mongo Instances
  Mongo: localhost:27017 (5)
    localhost:27017
    local (14/9300)
    admin (0/0)
    test (1038/74748)
    Corpus (3775/247604)
      system.indexes (6/43
      tweets (497/111104)
        _id_ (3/24528)
```

```
        Name : tweets
   Full Name : Corpus.tweets
Query Options :
Write Concern : { "getlasterror" : 1 , "w" : 1 , "wtimeout" : 0}
        Stats : { "serverUsed" : "localhost:27017" , "ns" : "Corpus.tweets" , "count" : 497 , "size" : 111104
```

```
Cursor id=1687845800049579, ns=Corpus.tweets, query={}, numIterated=50, addr=localhost:27017, readP
{ "_id" : { "$oid" : "51ed71359dd27c0b946664ab"} , "via" : "kindle2" , "sentiment" : 4 , "text" : "Reading my
{ "_id" : { "$oid" : "51ed71359dd27c0b946664ac"} , "via" : "kindle2" , "sentiment" : 4 , "text" : "Ok first asse
{ "_id" : { "$oid" : "51ed71359dd27c0b946664ad"} , "via" : "kindle2" , "sentiment" : 4 , "text" : "@kenburba
{ "_id" : { "$oid" : "51ed71359dd27c0b946664ae"} , "via" : "kindle2" , "sentiment" : 4 , "text" : "@mikefish
{ "_id" : { "$oid" : "51ed71359dd27c0b946664af"} , "via" : "kindle2" , "sentiment" : 4 , "text" : "@richardeba
    _id: 51ed71359dd27c0b946664af
    via: kindle2
    sentiment: 4
    text: @richardebaker no. it is too big. I'm quite happy with the Kindle2.
    user: GeorgeVHulme
    date: Mon May 11 03:22:00 UTC 2009
    id: 8
{ "_id" : { "$oid" : "51ed71359dd27c0b946664b0"} , "via" : "aig" , "sentiment" : 0 , "text" : "Fuck this econo
```

Aggregation

- Compare NoSQL to SQL, example:
  - SQL

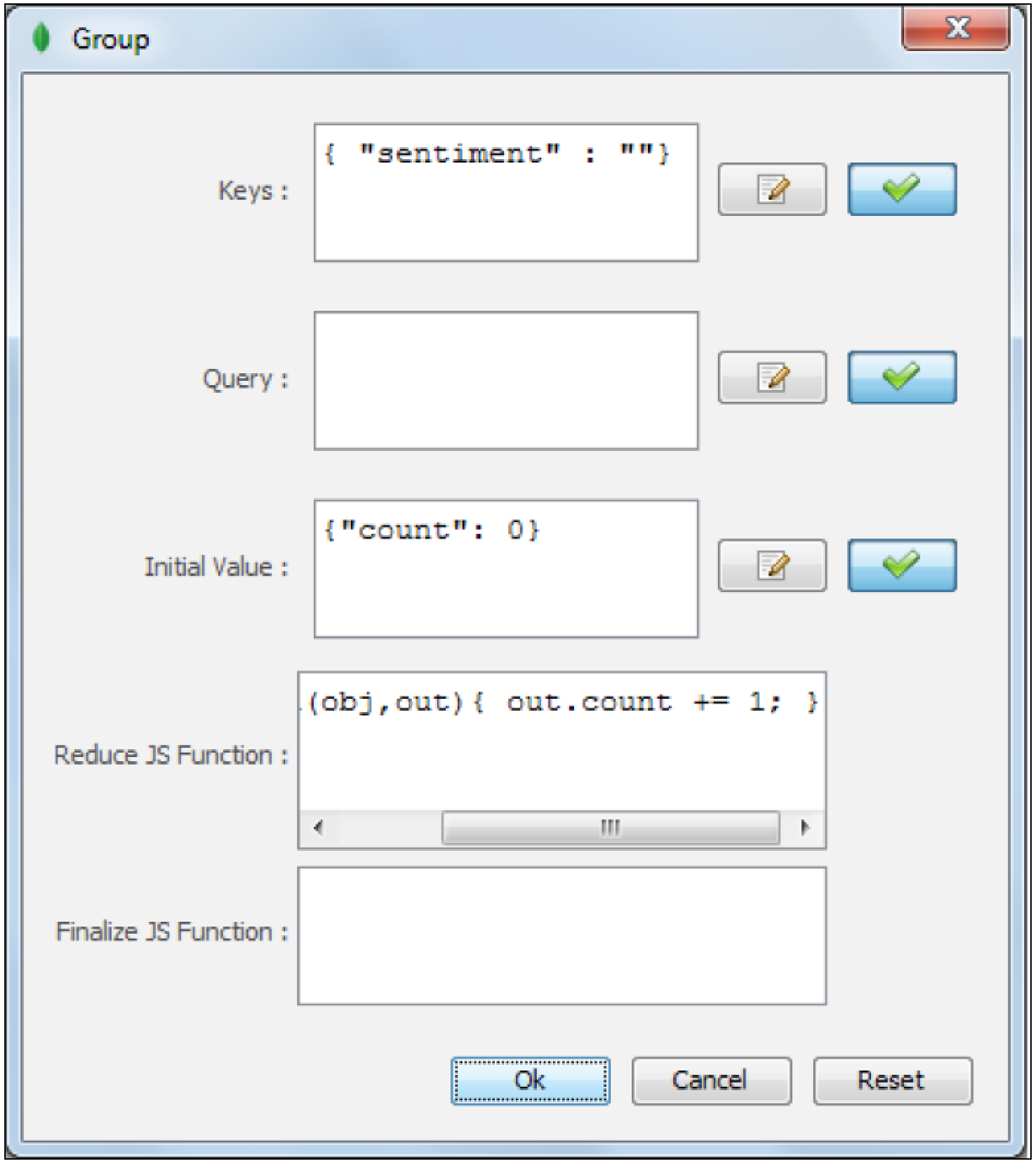
```
SELECT sentiment, COUNT(*)
FROM Tweets
GROUP BY sentiment
```

  - NoSQL with MongoDB:

```
db.collection.group({
  key:{sentiment:true},
  reduce: function(obj,prev{prev. sentimentsum += obj.c}),
  initial: {sentimentsum: 0}
});
```

Aggregation

- Via UMongo:
  - Statement
  - Results next slide



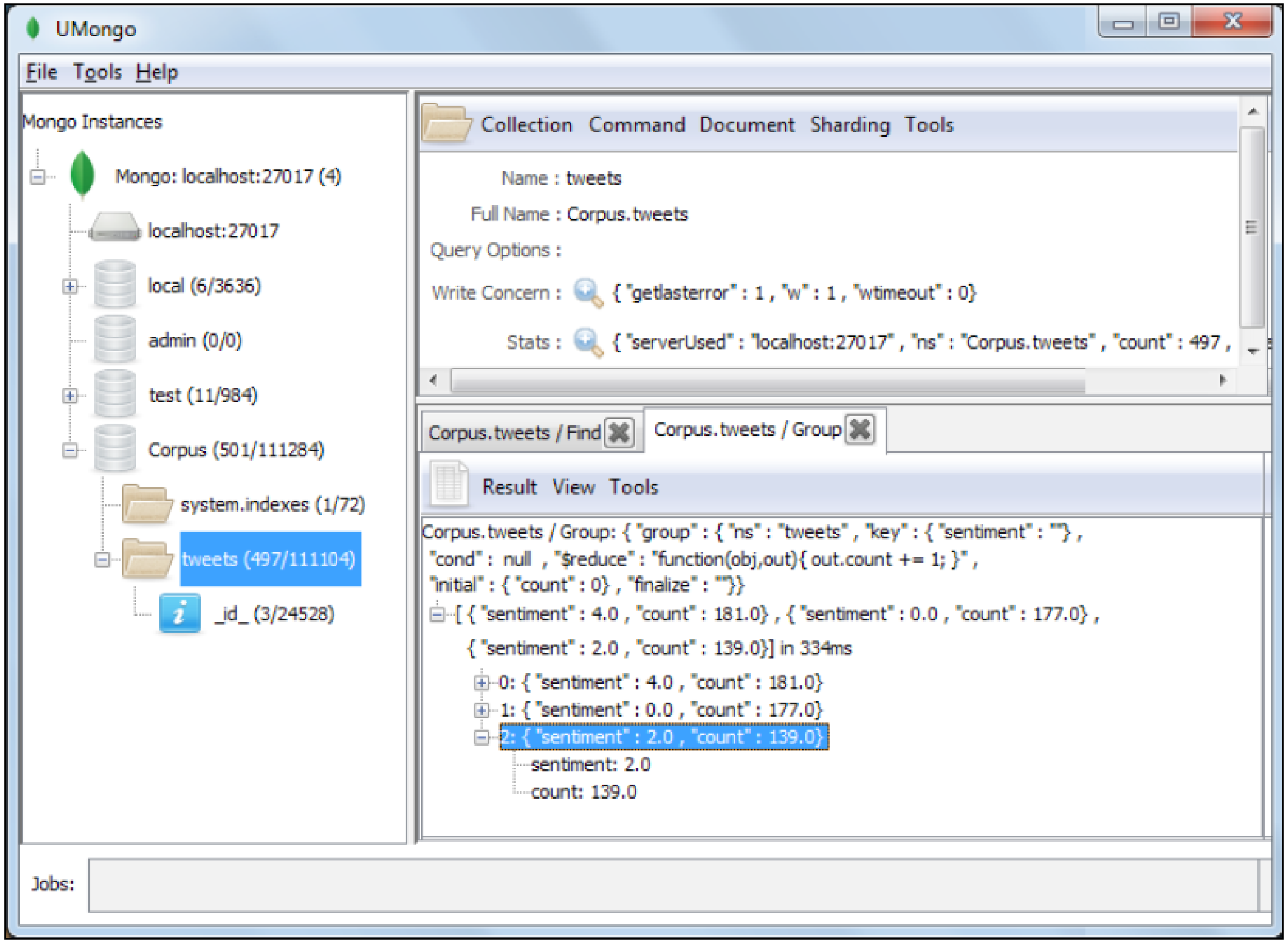
```
{ "sentiment" : ""}
```

```
{"count": 0}
```

```
(obj,out){ out.count += 1; }
```

Aggregation

- Via UMongo:
  - Results



```
Name : tweets
Full Name : Corpus.tweets
Query Options :
Write Concern : { "getlasterror" : 1 , "w" : 1 , "wtimeout" : 0}
Stats : { "serverUsed" : "localhost:27017" , "ns" : "Corpus.tweets" , "count" : 497 ,
```

```
Corpus.tweets / Group: { "group" : { "ns" : "tweets" , "key" : { "sentiment" : ""} ,
 "cond" : null , "$reduce" : "function(obj,out){ out.count += 1; }" ,
 "initial" : { "count" : 0} , "finalize" : ""}}
[ { "sentiment" : 4.0 , "count" : 181.0} , { "sentiment" : 0.0 , "count" : 177.0} ,
  { "sentiment" : 2.0 , "count" : 139.0}] in 334ms
    0: { "sentiment" : 4.0 , "count" : 181.0}
    1: { "sentiment" : 0.0 , "count" : 177.0}
    2: { "sentiment" : 2.0 , "count" : 139.0}
        sentiment: 2.0
        count: 139.0
```

Aggregation

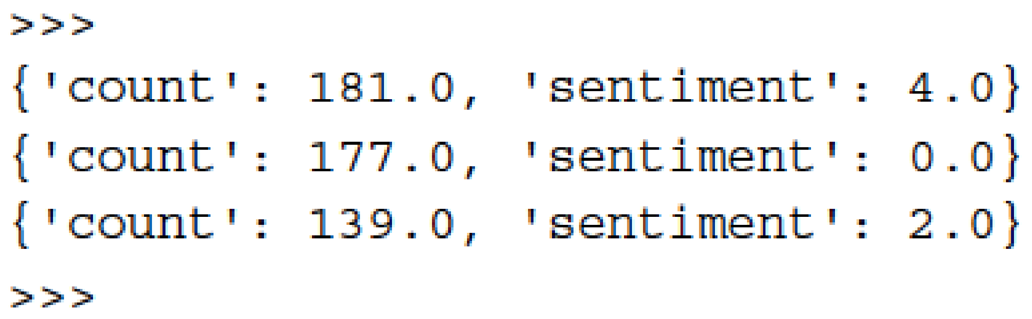
- Complete code for aggregation with PyMongo:

```
from pymongo import MongoClient
con = MongoClient()
db = con.Corpus
tweets = db.tweets

categories = tweets.group(key={"sentiment":1},
  condition={},
    initial={"count": 0},
      reduce="function(obj,   prev)
        {prev.count++;}")
for doc in categories:
  print(doc)
```

Aggregation

- Results from the previous slide’s code:



```
>>>
{'count': 181.0, 'sentiment': 4.0}
{'count': 177.0, 'sentiment': 0.0}
{'count': 139.0, 'sentiment': 2.0}
>>>
```

The Aggregation Framework

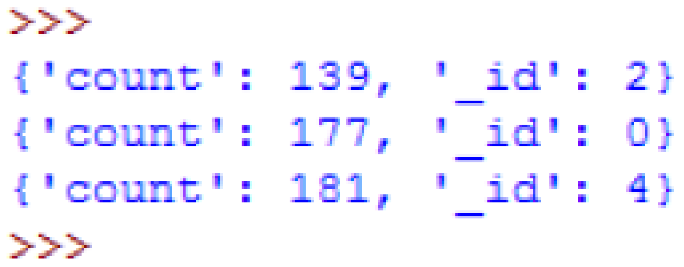
- General aggregate method:

```
db.collection.aggregate( [<pipeline>] )
```

- Example to count:

```
results = tweets.aggregate([
  {"$group": {"_id": "$sentiment", "count": {"$sum": 1}}}
  ])

for doc in results["result"]:
  print(doc)
```



```
>>>
{'count': 139, '_id': 2}
{'count': 177, '_id': 0}
{'count': 181, '_id': 4}
>>>
```

http://docs.mongodb.org/manual/reference/aggregation/

The Aggregation Framework

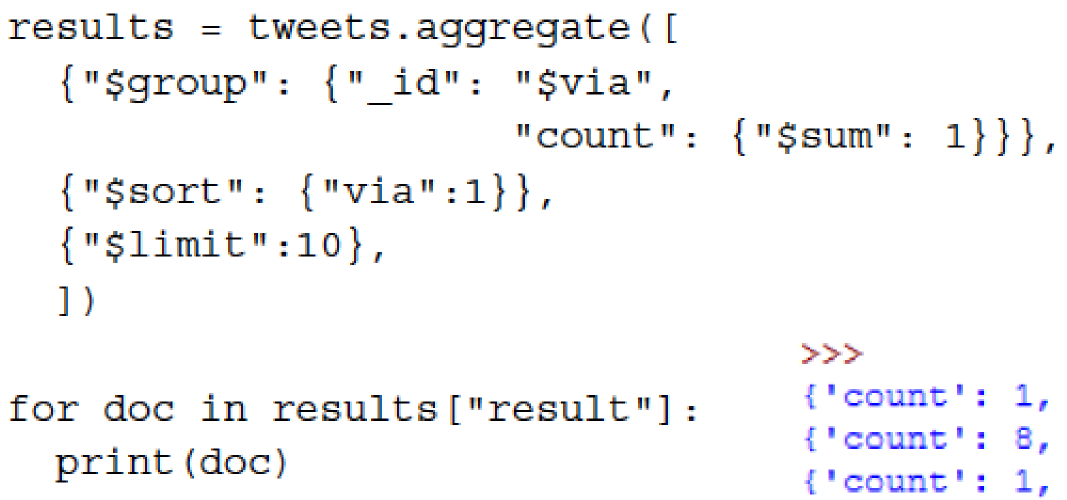
- Pipeline, a series of operators the filter or transform data:
  - $match: filters documents
  - $group: groups documents by an id and can use all the computational expressions such as $max, $min…
  - $unwind: operates on an array field, yield documents for each array value, and also complements $match and $group
  - $sort: sorts documents by one or more fields
  - $skip: skips over documents in the pipeline
  - $limit: restricts the number of documents in an aggregation pipeline

```
db.posts.aggregate(
 $e
}
```

http://docs.mongodb.org/manual/reference/aggregation/

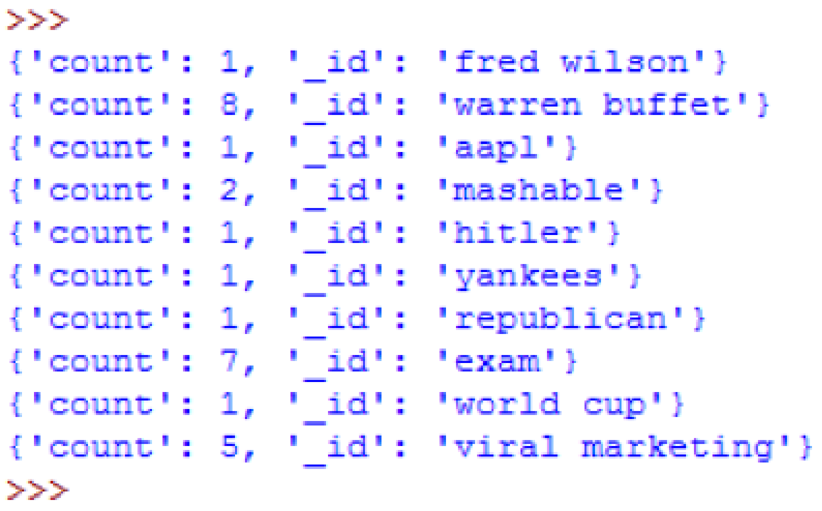
The Aggregation Framework

- Pipelines example:



```
results = tweets.aggregate([
  {"$group": {"_id": "$via",
                      "count": {"$sum": 1}}},
  {"$sort": {"via":1}},
  {"$limit":10},
  ])

for doc in results["result"]:
  print(doc)
```



```
>>>
{'count': 1, '_id': 'fred wilson'}
{'count': 8, '_id': 'warren buffet'}
{'count': 1, '_id': 'aapl'}
{'count': 2, '_id': 'mashable'}
{'count': 1, '_id': 'hitler'}
{'count': 1, '_id': 'yankees'}
{'count': 1, '_id': 'republican'}
{'count': 7, '_id': 'exam'}
{'count': 1, '_id': 'world cup'}
{'count': 5, '_id': 'viral marketing'}
>>>
```

http://docs.mongodb.org/manual/reference/aggregation/

The Aggregation Framework

- Expressions produce documents based on calculations performed on input documents:
  - $max: returns the highest value in the group
  - $min: returns the lowest value in the group
  - $avg: returns the average of all the group values
  - $sum: returns the sum of all values in the group
  - $addToSet: returns an array of all the distinct values for a certain field in each document in that group

http://docs.mongodb.org/manual/reference/aggregation/

The Aggregation Framework

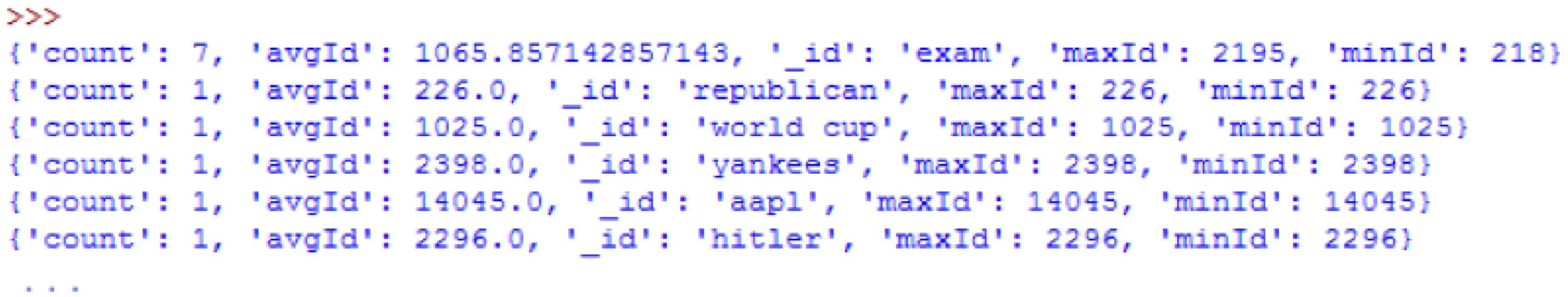
- Operators in expressions:
  - Boolean: $and, $or, and $not
  - Arithmetic: $add, $divide, $mod, $multiply, and $substract
  - String: $concat, $substr, $toUpper, $toLower, and $strcasecmp
  - Conditional: $cond and $ifNull

http://docs.mongodb.org/manual/reference/aggregation/

The Aggregation Framework

- Expressions example:

```
results = tweets.aggregate([
  {"$group": {"_id": "$via",
    "avgId": {"$avg": "$id"} ,
    "maxId": {"$max": "$id"} ,
    "minId": {"$min": "$id"} ,
    "count": {"$sum": 1}}}
  ])
for doc in results["result"]:
  print(doc)
```



```
>>>
{'count': 7, 'avgId': 1065.857142857143, '_id': 'exam', 'maxId': 2195, 'minId': 218}
{'count': 1, 'avgId': 226.0, '_id': 'republican', 'maxId': 226, 'minId': 226}
{'count': 1, 'avgId': 1025.0, '_id': 'world cup', 'maxId': 1025, 'minId': 1025}
{'count': 1, 'avgId': 2398.0, '_id': 'yankees', 'maxId': 2398, 'minId': 2398}
{'count': 1, 'avgId': 14045.0, '_id': 'aapl', 'maxId': 14045, 'minId': 14045}
{'count': 1, 'avgId': 2296.0, '_id': 'hitler', 'maxId': 2296, 'minId': 2296}
 ...
```

http://docs.mongodb.org/manual/reference/aggregation/

References

- Chapter 12 Data Processing and Aggregation with MongoDB, of Hector Cuesta (2013). Practical Data Analysis. https://ebookcentral-proquest-com.ezproxy2.library.drexel.edu/lib/drexel-ebooks/detail.action?docID=1507840