Query Primer:
find ( ) and aggregate ( )

Daily stand up

- Name – Company
- Something about Mongo admin/coding you know that no one else here knows
- Something about Mongo you wish you knew
- Your next major Mongo project

Something I know

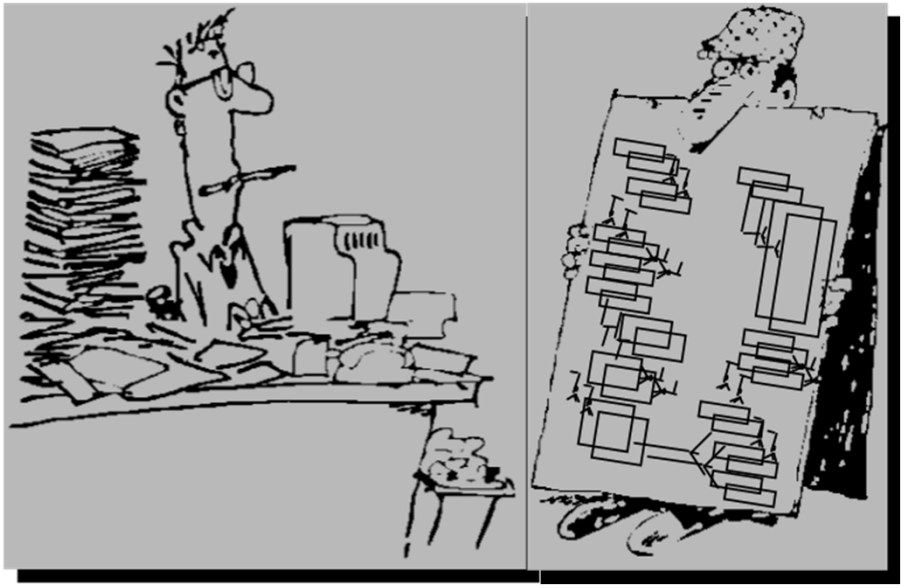
http://blog.mongodb.org/post/95839709598/how-to-perform-fuzzy-matching-with-mongo-connector

https://github.com/mongodb-labs/mongo-connector/wiki/Usage-with-ElasticSearch
https://github.com/mongodb-labs/mongo-connector/wiki/FAQ

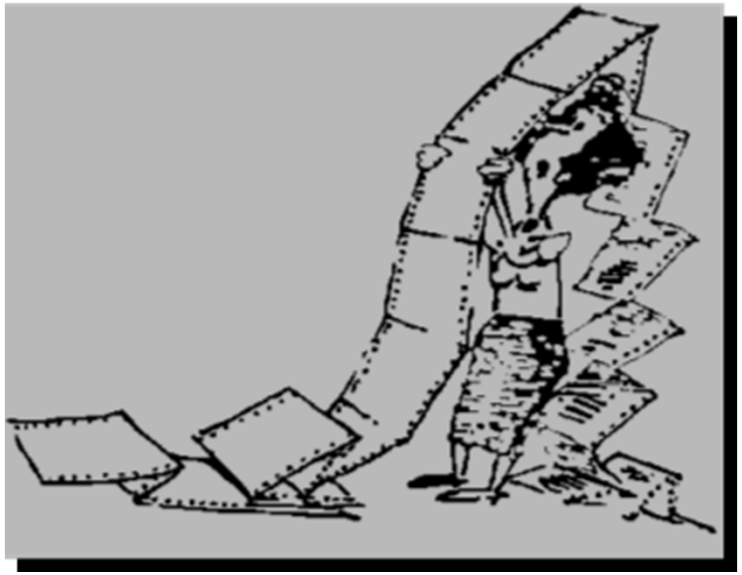
https://blog.jixee.me/how-to-use-mongo-connector-with-elasticsearch/

My team and I have been tasked with learning MongoDB, writing queries and getting a new project out of the door in 3 weeks.
I know how to write most queries in SQL, and have been doing that for years.

Can you detail for me when to use MongoDB find ( ), versus aggregate ( ); pretty much give me a primer on the whole subject area?



Rules of Engagement



Covered-
- TPC-C
- TPC-H
- (Favorite past DB)
- Zips dataset

- find ( )
- aggregate ( )
- count ( ), distinct ( )

- 20+ queries

Not covered-
- Update ( )
- Remove ( )

- Query tuning (last month)

Final exam ?!

And first'ish:

Y/N I have more than 24 months experience with SQL

Y/N I have more than 6 months experience with MongoDB

Y/N I have written every query in MongoDB I ever needed, and was happy

Y/N Puppies scare me

When to use ..

find ( )

- 32 MB
- No overflow

- sort ( )
- limit ( )
- pretty ( )

- Less than SELECT

- (Nothing else)

aggregate ( )

- 100 MB *per stage*
- Yes overflow

- More than SELECT

- (Everything else)

count ( )

distinct ( )

zips.json

https://university.mongodb.com/

http://media.mongodb.org/zips.json

```
db.zips.findOne( )
   {
   "_id" : ObjectId("57042dfb4395c1c26641c1f2"),
   "city" : "ACMAR",
   "zip" : "35004",
   "loc" : {
         "y" : 33.584132,
         "x" : 86.51557
         },
   "pop" : 6055,
   "state" : "AL"
   }
```

Queries: 101 - 106

Example 101: db.zips.find({ “state” : “CO” })

Example 102: db.zips.find({ “state” : “CO” }, { “_id” : 0 })

Example 103: db.zips.find({ "state" : "CO" }).count( )

query document
projection document
cursor methods

Example 104: db.zips.find({ "state" : "CO" }).sort({ "pop" : 1 })

pipelined (sort)

Example 105: db.zips.find({ "state" : "CO" }).
sort({ "pop" : -1 }).limit(5)

Example 106: db.zips.find({ "state" : "CO" }, { "_id" : 0 }).
sort({ "pop" : -1 }).limit(5).explain("executionStats")

Trivia # 01:

mongod .. --notablescan

db.zips.find({ "pop" : 7442 })

Trivia # 01: --notablescan, answer

```
db.zips.find( { "pop" : 7442 } )
```

```
Error: error: {
        "waitedMS" : NumberLong(0),
        "ok" : 0,
        "errmsg" : "error processing query:
ns=test_db1.zipsTree: pop == 7442.0\nSort: {}\nProj:
{}\n No query solutions",
        "code" : 2
}
```

```
db.zips.aggregate(
  [
  { "$match" :
    {
    "pop" : 7442
    }
  } ] )
```

Queries: 107, first aggregate

```
db.zips.aggregate(
   [
   { "$group" :
      {
      "_id" : "$state",
         //
      "totalPop" : { "$sum" : "$pop" },
      "cityCount" : { "$sum" : 1 }
      }
   },
   { "$sort" :
      { "_id" : 1 }
   }
   ],
   {
   "allowDiskUse" : true
   } )
```

aggregation framework stages

group accumulator operators
(other) operators

$pop ?
$sum (1)
$sort -> _id ?

allow disk use

$sum works in project too (3.2+)

Queries: 108, match stage

```
db.zips.aggregate(
   [
   { "$match" :
      {
      "state" : "CO"
      }
   }
   ] )
```

Queries: 109, sort stage

```
db.zips.aggregate(
  [
  { "$match" :
    {
    "state" : "CO"
    }
  },
  { "$sort" :
    {
    "city" : 1,
    "pop" : -1
    }
  }
  ] )
```

Queries: 110, group stage

```
db.zips.aggregate( [
  { "$group" :
    {
    "_id"       : "$state",
      //
    "totalPop"  : { "$sum" : "$pop" },
    "cityCount" : { "$sum" : 1 }
    }
  } ,
  { "$sort" :
    {
    "_id" : 1
    }
  }
  ] )
```

Queries: 111, limit stage

```
db.zips.aggregate( [
  { "$group" :
    {
    "_id"       : "$state",
    "totalPop"  : { "$sum" : "$pop" }
    }
  } ,
  { "$sort" :
    {
    "totalPop" : -1
    }
  },
  { "$limit" : 5
  }
  ] )
```

This gives us a FIRST (N)

Is there a FIRST (N) LAST (N)

Queries: 112, out stage

```
db.zips.aggregate([
  { "$match" :
    {
    "state"  : "CO"
    }
  } ,
  { "$out" : "zipsCO"
  }
  ] )
```

Not sharded, nor capped
Preserves indexes

Transactional'ish

Queries: 113, sample stage

```
db.zips.aggregate([
  { "$sample" :
    { "size" : 5 }
  }
  ] )
```

Complex

https://docs.mongodb.com/manual/reference/operator/aggregation/sample/#pipe._S_sample

```
db.states.insert( { "abbr" : "WI", "name" : "Wisconsin" } )
db.zips.aggregate( [
  {
  "$match" :  { "state" : "WI" } },
  {
  "$lookup" :
    {
    "from"         : "states",
    "localField"   : "state",
    "foreignField" : "abbr",
    "as"           : "state"
    }
  },
  {
  "$project" :
    {
    "_id"        : 0,
    "city"       : 1,
    "state"      : "$state.name"
    }
  } ] )
```

Queries: 114, lookup stage

Left outer

Version 3.3.8, $graphLookup

project ?

Queries: 115, indexStats stage

```
db.zips.aggregate( [ { "$indexStats" : {} } ] ).pretty( )
```

Trivia # 02: Speaking of indexes, (and memory) ..

match { }
sort { }
group { }

(other)

Trivia # 02: Speaking of indexes, (and memory) ..

https://docs.mongodb.com/manual/core/aggregation-pipeline/#aggregation-pipeline-behavior

Not thus yet covered ..

geoNear
unwind
(wind)
project

TPC-C: Queries 201 – 203, ands and ors

Example 201: db.zips.find({ "state" : "CO", "city" : "BUENA VISTA" })

Example 202: db.zips.find(

```
{
"$or" : [ { "state" : "CO" }, { "city" : "BUENA VISTA" } ]
}
)
```

Example 203: db.zips.find(

```
{
"$or" :
  [
  { "state" : "CO" , "city" : "BUENA VISTA" },
  { "pop" : { "$lte" : 4 } }
  ]
}
)
```

TPC-C: Query 204, column operator/expression

```
Example 204: db.zips.aggregate(
  [
  { "$match" :
    {
    "pop" : 4
    }
  },
  { "$project" :
    {
    "_id"        : 0,
    "state"      : 1,
    "city"       : 1,
    "pop"        : { "$add" : [ "$pop" , 2 ] }
    }
  }
  ]
  )
```

Aggregation arithmetic operator

TPC-C: Query 205, count ( )

Example 205: db.zips.find({ "state" : "CO" }).count( )

Collection operator

TPC-C: Query 206, distinct ( )

Example 206:

```
db.zips.distinct( "city" )
db.zips.distinct( "city" , { "pop" : { "$lt" : 100 } } )
```

Different than SQL, one key only

Multiple keys ?, $group

TPC-C: Query 207, group ( )

```
//  This query fails
db.zips.aggregate(
   [
   { "$group" :
      {
      "_id" : { "state" : "$state" , "city" : "$city" },
      "pop" : "$pop"
      }
   }
   ]
   )
// This query works, but ..
db.zips.aggregate(
   [
   { "$group" :
      {
      "_id" : { "state" : "$state" , "city" : "$city" }
      }
   }
   ]
   ) )
```

Why ?

And why but ?

Query 208: More coming ..

```
db.zips.find( { "city" : "BUENA VISTA" }, { "_id" : 0, "city" : 1, "state" : 1 } )

# { "city" : "BUENA VISTA", "state" : "CO" }
# { "city" : "BUENA VISTA", "state" : "PA" }
# { "city" : "BUENA VISTA", "state" : "TN" }
# { "city" : "BUENA VISTA", "state" : "VA" }


# {
# '_id': {'city': 'BUENA VISTA'},
# 'countOf': 4,
# {'memberStates': [{'state': 'CO'},
#    {'state': 'PA'}, {'state': 'TN'}, {'state': 'VA'}]
# }
```

TPC-C: Query 209, in list

```
db.zips.find( { "state" : { "$in" : [ "CO" , "WI" ] } } )
```

Not in the TPC-C: Query 210, Nested documents

```
# {'city': 'BUENA VISTA', 'state': 'CO', 'zip': '81211', 'pop': 5220, 'loc': [-106.147121, 38.838003]}

db.zips.update( { "zip" : "81211" }, { "$set" : { "thingsToDo.water" : "raft", "thingsToDo.food" :
"Pizza Works" } } )

db.zips.find( { "thingsToDo.water" : "raft" }, { "_id" : 0 } )

# { "city" : "BUENA VISTA", "loc" : [ -106.147121, 38.838003 ],
#    "pop" : 5220, "state" : "CO", "zip" : "81211",
#    "thingsToDo" : { "water" : "raft", "food" : "Pizza Works" } }

db.zips.update( { "zip" : "81211" }, { "$unset" : { "thingsToDo" : "" } } )
```

Not in the TPC-C: Query 211, Positional querying

```
db.zips.update( { "zip" : "81211" }, { "$set" : { "thingsToDo" : [ "raft", "Pizza Works" ] } } )

# { "city" : "BUENA VISTA", "loc" : [ -106.147121, 38.838003 ],
#    "pop" : 5220, "state" : "CO", "zip" : "81211", "thingsToDo" : [ "raft", "Pizza Works" ] }

db.zips.find( { "thingsToDo" : "Pizza Works" }, { "_id" : 0 } )
db.zips.find( { "thingsToDo.0" : "Pizza Works" }, { "_id" : 0 } )
db.zips.find( { "thingsToDo.1" : "Pizza Works" }, { "_id" : 0 } )

db.zips.update( { "zip" : "81211" }, { "$set" : { "thingsToDo.1" : "Pizza Works 3000" } } )

db.zips.update( { "zip" : "81211" }, { "$unset" : { "thingsToDo" : "" } } )
```

TPC-H: Not that much new, just more

```
select
        l_returnflag,
        l_linestatus,
        sum(l_quantity) as sum_qty,
        sum(l_extendedprice) as sum_base_price,
        sum(l_extendedprice * (1 - l_discount)) as sum_disc_price,
        sum(l_extendedprice * (1 - l_discount) * (1 + l_tax)) as sum_charge,
        avg(l_quantity) as avg_qty,
        avg(l_extendedprice) as avg_price,
        avg(l_discount) as avg_disc,
        count(*) as count_order
from
        lineitem
where
        l_shipdate <= date ('1998-12-01') - :1 day
group by l_returnflag, l_linestatus
order by l_returnflag, l_linestatus;
```

TPC-H: Query 212, CASE statement

```
db.zips.aggregate(
  [
  { "$match" :
    { "state" : { "$in" : [ "CO" , "WI", "TX", "NV" ] } }
  },
  { "$group" :
    { "_id" : "$state" }
  },
```

```
{ "$project" :
  {
  "state"         : 1,
  "stateCoolness" :
    {
    "$switch" :
      {
        "branches" :
        [
        {
        "case" :
          { "$or" : [
            { "$eq" : [ "$_id" , "CO" ] },
            { "$eq" : [ "$_id" , "WI" ] }
          ] } ,
        "then" : "Very Cool"
        },
        ],
        "default" : "Not Cool"

        } } } } ] )   //  Sorry about the indents
```

Version 3.3.5 or higher

TPC-H: Query 212, CASE statement results

{ "_id" : "TX", "stateCoolness" : "Not Cool" }
{ "_id" : "WI", "stateCoolness" : "Very Cool" }
{ "_id" : "NV", "stateCoolness" : "Not Cool" }
{ "_id" : "CO", "stateCoolness" : "Very Cool"

Favorite past database: Queries 213 - 214

Example 213: db.zips.find({ "zip" : { "$in" : [/^53/, /^80/] } })

Example 214: db.zips.aggregate(

```
  [
  { "$group" :
    { "_id" : "$state" }
  },
  { "$match" :
    { "_id" : { "$in" : [ "CO" , "WI", "TX", "NV" ] } }
  },
  ] )
```

LIKE

HAVING Clause

Final Exam ?!

Final exam: Query 215

```
db.zips.findOne( )
  {
  "_id" : ObjectId("57042dfb4
         395c1c26641c1f2"),
  "city" : "ACMAR",
  "zip" : "35004",
  "loc" : {
        "y" : 33.584132,
        "x" : 86.51557
        },
  "pop" : 6055,
  "state" : "AL"
  }
```

$match { }
$sort { }
$group { }
$limit { }
$sample { }
$lookup { }
$geoNear { }
$out { }
$indexStats { }
$unwind { }
$project { }

Which city name is most common across US states ?

Final exam: Query 215, resultant data

{countOf': 24, _id': u'CLINTON'}
{countOf': 24, _id': u'FRANKLIN'}
{countOf': 23, _id': u'MADISON'}
{countOf': 22, _id': u'GREENVILLE'}
{countOf': 22, _id': u'ARLINGTON'}
{countOf': 21, _id': u'SALEM'}
{countOf': 21, _id': u'CHESTER'}
{countOf': 20, _id': u'SPRINGFIELD'}
{countOf': 19, _id': u'PRINCETON'}
{countOf': 19, _id': u'TROY'}

Final exam: Query 215, answer

```
db.zips.aggregate(
  [  { "$group" :
       {
       "_id"     : { "city"   : "$city" , "state" : "$state“ }
       }
     } ,
     { "$group" :
       {
       "_id"     : "$_id.city",
       "countOf" : { "$sum" : 1 }
       }
     } ,
     {
     "$sort" : { "countOf" : -1 }
     },
     { "$limit" : 10 }
  ] )
```

Final exam: But what if you need this ?

{countOf': 24, _id': u'CLINTON‘}

```
#  [u'MS', u'IA', u'SC', u'NC', u'LA', u'ME', u'AR',
#     u'OK', u'OH', u'PA', u'WI', u'TN', u'MN',
#     u'WA', u'MI', u'IL', u'NY', u'IN', u'CT',
#     u'NJ', u'MT', u'KY', u'MD', u'MA'],
#     u'countOf': 24, u'_id': u'CLINTON'}
```

Final exam: Query 216

```
db.zips.aggregate(
  [ { "$group" :
       { "_id"       : { "city"    : "$city" , "state"  : "$state“ } }
    } ,
    {
    "$group" :
      {
      "_id"       : "$_id.city",
      "countOf"   : { "$sum" : 1 },
      "stateArr"  : { "$addToSet" : "$_id.state" }
      }
    } ,
    {
    "$sort" : { "countOf" : -1 }
    },
    { "$limit" : 10 }
  ] )
```

Which 1 line is different ?

Final exam: Query 217, alternate answer

```
db.zips.aggregate(
  [ { "$group" :
        { "_id"       : { "city"    : "$city" , "state"  : "$state” }
        }
    } ,
    { "$group" :
      {
      "_id"       : "$_id.city",
      "stateArr"  : { "$addToSet" : "$_id.state" }
      }
    } ,
    { "$project" :
      {
      "_id"       : "$_id",
      "countOf"   : { "$size"  : "$stateArr“  },
      "stateArr"  : "$stateArr"
      }
    },
    { "$sort" : { "countOf" : -1 }
    },
    { "$limit" : 10 } ] )
```

Final exam: But what if you need this ?

{countOf': 24, _id': u'CLINTON‘}

```
#  [u'MS', u'IA', u'SC', u'NC', u'LA', u'ME', u'AR',
#     u'OK', u'OH', u'PA', u'WI', u'TN', u'MN',
#     u'WA', u'MI', u'IL', u'NY', u'IN', u'CT',
#     u'NJ', u'MT', u'KY', u'MD', u'MA'],
#     u'countOf': 24, u'_id': u'CLINTON'}

# {'stateArr': 'MS', 'countOf': 24, '_id': 'CLINTON'}
# {'stateArr': 'IA', 'countOf': 24, '_id': 'CLINTON'}
# {'stateArr': 'SC', 'countOf': 24, '_id': 'CLINTON'}
```

Final exam: Query 218, $unwind

```
db.zips.aggregate(
   [ { "$group" :
         { "_id"       : { "city"    : "$city" , "state"  : "$state” }
         }
      } ,
           ... ...  lines deleted

      { "$sort" : { "countOf" : -1 }
      },
      { "$limit" : 10 },
      {
      "$unwind" : "$stateArr"
      } ] )
```

Final exam: Query 219

```
db.zips.findOne( )
  {
  "_id" : ObjectId("57042dfb4
        395c1c26641c1f2"),
  "city" : "ACMAR",
  "zip" : "35004",
  "loc" : {
       "y" : 33.584132,
       "x" : 86.51557
       },
  "pop" : 6055,
  "state" : "AL"
  }
```

$match { }
$sort { }
$group { }
$limit { }
$sample { }
$lookup { }
$geoNear { }
$out { }
$indexStats { }
$unwind { }
$project { }

Which US state has the highest (or lowest) number of unique city names ?

Final exam: Query 219, resultant data

{ "_id" : ["PA"], "countOf" : 856 }
{ "_id" : ["NY"], "countOf" : 785 }
{ "_id" : ["CA"], "countOf" : 730 }
{ "_id" : ["TX"], "countOf" : 683 }

... ... lines deleted

{ "_id" : ["NV"], "countOf" : 35 }
{ "_id" : ["RI"], "countOf" : 24 }
{ "_id" : ["DE"], "countOf" : 15 }
{ "_id" : ["DC"], "countOf" : 1 }

Final exam: Query 219

```
db.zips.aggregate(
  [ { "$group" :
        {
        "_id"     : { "city"   : "$city" , "state" : "$state" }
        }
    } ,
    { "$group" :
      { "_id"     : "$_id.city",
      "countOf" : { "$sum" : 1 } ,
      "memberStates" : { "$push" : { "stateName" : "$_id.state" } }
      }
    } ,
    { "$match" : { "countOf" : 1 } },
    { "$group" :
      { "_id"     : "$memberStates.stateName",
      "countOf" : { "$sum" : 1 }
      }
    },
    { "$sort" : { "countOf" : -1 } }
  ] )
```

Final exam: Query 301a, grand total

```
db.zips.aggregate(
  [ { "$group" :
      { "_id"           : { "state"      : "$state" },
      "groupTotal"    : { "$sum" : "$pop" }
      }
    },
    { "$project" :
      { "_id"           : 0,
      "state"         : "$_id.state",
      "groupTotal"    : 1
      }
    },
    { "$group" :
      { "_id"           : { "_id2"       : { "$literal" : 1 } },
      "grandTotal"    : { "$sum" : "$groupTotal" }
      }
    }
  ] )
```

Final exam: Query 301b, simpler

```
db.zips.aggregate(
  [
    {
    "$group" :
      {
      "_id"           :
        {
        "_id2"       : { "$literal" : 1 }
        },
      "grandTotal"    : { "$sum" : "$pop" }
      }
    }
  ] )
```

Final exam: Query 302

```
db.zips.findOne( )
  {
  "_id" : ObjectId("57042dfb4
        395c1c26641c1f2"),
  "city" : "ACMAR",
  "zip" : "35004",
  "loc" : {
       "y" : 33.584132,
       "x" : 86.51557
       },
  "pop" : 6055,
  "state" : "AL"
  }
```

$match { }
$sort { }
$group { }
$limit { }
$sample { }
$lookup { }
$geoNear { }
$out { }
$indexStats { }
$unwind { }
$project { }

There is a FIRST N (or LAST N), but not a built in FIRST AND LAST.

Still: How could you deliver this ?

Say $sum of pop, top 5 and bottom 5-

Final exam: Query 302, resultant data

{ "state" : "AK", "pop" : 550043 }
{ "state" : "DC", "pop" : 606900 }
{ "state" : "WY", "pop" : 453588 }
{ "state" : "ND", "pop" : 638800 }
{ "state" : "VT", "pop" : 562758 }

{ "state" : "PA", "pop" : 11881643 }
{ "state" : "FL", "pop" : 12937926 }
{ "state" : "TX", "pop" : 16986510 }
{ "state" : "NY", "pop" : 17990455 }
{ "state" : "CA", "pop" : 29760021 }

Final exam: Query 302, top and bottom

```
db.zips.aggregate(
  [ { "$group" : { "_id" : { "state" : "$state" },
      "pop"   : { "$sum" : "$pop" } } },
    { "$project"                  :
      {
      "_id"         : 0,
      "state"       : "$_id.state",
      "pop"         : 1,
      "nextGroupId" : { "$literal"  : 1 }
      }
    },
    { "$sort"  : { "pop"  : -1 } },
    { "$group"  : { "_id"  : { "id2"  : "$nextGroupId" },
      "stateArr"  : { "$push"  :
        { "state"  : "$state", "pop"  : "$pop"  } } }
    },
        ... ...  continued
```

Final exam: Query 302, top and bottom

```
    ... ...  continued

  { "$project"  :
    { "_id"  : 0,
    "stateArrTop"     : { "$slice" : [ "$stateArr"  , 5 ] },
    "stateArrBottom"  : { "$slice" : [ "$stateArr" , -5 ] },
    }
  },
  { "$project"  : { "allStates"  :
       { "$setUnion"  : [ "$stateArrTop", "$stateArrBottom" ] } }
  },
  { "$unwind"  : "$allStates" },
  {
  "$project"  :
    { "state" : "$allStates.state", "pop" : "$allStates.pop", }
  }
] )
```

How fun was that ?

Resources:

- The parent to this preso,
  https://github.com/farrell0/MongoDB-Developers-Notebook
- University.MongoDB.com

- zips.json
  http://media.mongodb.org/zips.json

- Call Dave Lutz, at home, ... On Sunday (early)
  (512)555/1212