

```
/*  
 * @Author: Sidharth Mishra  
 * @Date: 2017-02-24 13:21:30  
 * @Last Modified by: Sidharth Mishra  
 * @Last Modified time: 2017-02-25 00:25:07  
 */
```

// 1. Download the zips.json file to your VM.

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

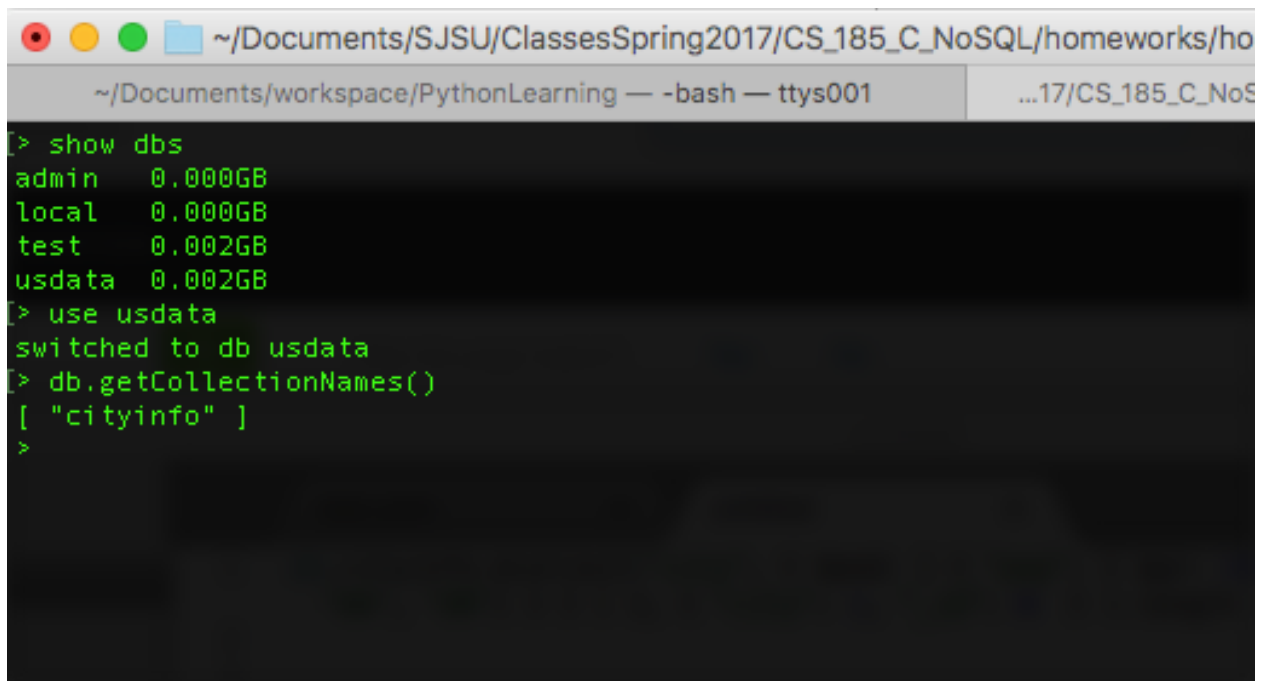
// 2. Import zips.json into MongoDB using mongoimport command.

// Import the data into a collection called cityinfo in a database called usdata.

```
mongoimport --db usdata --collection cityinfo --file zips.json
```

// 3. Get the screenshot (screen 1) that shows all collections of the database usdata.

```
show dbs  
use usdata  
db.getCollectionNames()
```

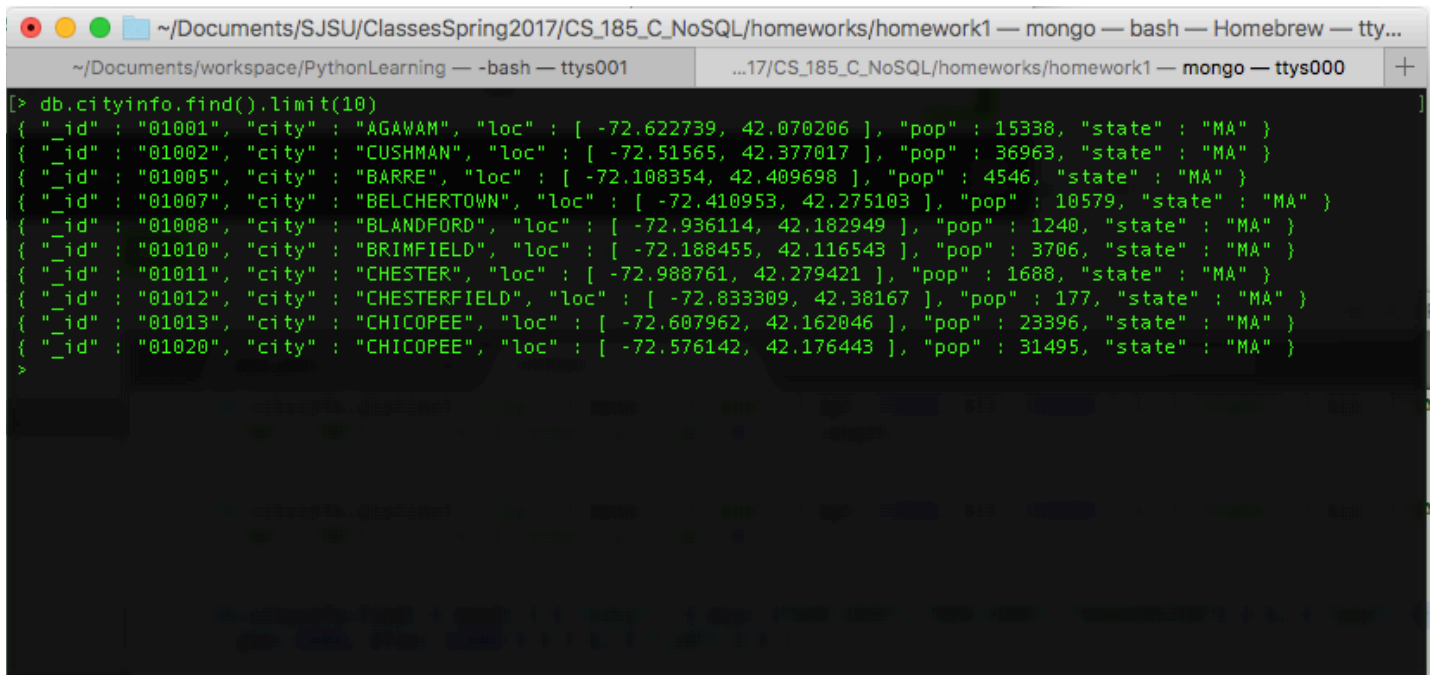


The screenshot shows a terminal window with the following content:

```
~/Documents/SJSU/ClassesSpring2017/CS_185_C_NoSQL/homeworks/ho  
~/Documents/workspace/PythonLearning — -bash — ttys001    ...17/CS_185_C_NoS  
[> show dbs  
admin    0.000GB  
local    0.000GB  
test     0.002GB  
usdata   0.002GB  
[> use usdata  
switched to db usdata  
[> db.getCollectionNames()  
[ "cityinfo" ]  
>
```

// 4. Find all documents of cityinfo collection. (screen 2)

```
db.cityinfo.find().limit(10)
```



The screenshot shows a terminal window with a dark background. The title bar at the top indicates the current directory is `~/Documents/SJSU/ClassesSpring2017/CS_185_C_NoSQL/homeworks/homework1` and the active application is `mongo`. The prompt is `ttys000`. The user has entered the command `db.cityinfo.find().limit(10)`, and the terminal displays the following JSON array of documents:

```
[{"_id": "01001", "city": "AGAWAM", "loc": [ -72.622739, 42.070206 ], "pop": 15338, "state": "MA" }, {"_id": "01002", "city": "CUSHMAN", "loc": [ -72.51565, 42.377017 ], "pop": 36963, "state": "MA" }, {"_id": "01005", "city": "BARRE", "loc": [ -72.108354, 42.409698 ], "pop": 4546, "state": "MA" }, {"_id": "01007", "city": "BELCHERTOWN", "loc": [ -72.410953, 42.275103 ], "pop": 10579, "state": "MA" }, {"_id": "01008", "city": "BLANDFORD", "loc": [ -72.936114, 42.182949 ], "pop": 1240, "state": "MA" }, {"_id": "01010", "city": "BRIMFIELD", "loc": [ -72.188455, 42.116543 ], "pop": 3706, "state": "MA" }, {"_id": "01011", "city": "CHESTER", "loc": [ -72.988761, 42.279421 ], "pop": 1688, "state": "MA" }, {"_id": "01012", "city": "CHESTERFIELD", "loc": [ -72.833309, 42.38167 ], "pop": 177, "state": "MA" }, {"_id": "01013", "city": "CHICOPEE", "loc": [ -72.607962, 42.162046 ], "pop": 23396, "state": "MA" }, {"_id": "01020", "city": "CHICOPEE", "loc": [ -72.576142, 42.176443 ], "pop": 31495, "state": "MA" }]
```

// 5. Find all documents with `_id` that contains 9503 in it. Do not include "loc" in the output.
// For example, expected documents in the output may include a document with `"_id": "19503"` and a document with `"_id": "95037"`. (screen 3)

```
db.cityinfo.find( {
  "_id": {
    $regex: /. *9503.*/i
  },
  {
    "loc": 0
  }
})
```

```
~/Documents/SJSU/ClassesSpring2017/CS_185_C_NoSQL/homeworks/homework1 — mongo — bash — Homebrew — tty...
~/Documents/workspace/PythonLearning — -bash — ttys001    ...17/CS_185_C_NoSQL/homeworks/homework1 — mongo — ttys000 +
[> db.cityinfo.find({"_id":{"regex: /.9503./i}}, {"loc":0})
{ "_id" : "19503", "city" : "BALLY", "pop" : 973, "state" : "PA" }
{ "_id" : "39503", "city" : "GULFPORT", "pop" : 26830, "state" : "MS" }
{ "_id" : "49503", "city" : "GRAND RAPIDS", "pop" : 32876, "state" : "MI" }
{ "_id" : "79503", "city" : "AVOCA", "pop" : 248, "state" : "TX" }
{ "_id" : "89503", "city" : "RENO", "pop" : 23955, "state" : "NV" }
{ "_id" : "95030", "city" : "MONTE SERENO", "pop" : 25881, "state" : "CA" }
{ "_id" : "95032", "city" : "LOS GATOS", "pop" : 18189, "state" : "CA" }
{ "_id" : "95035", "city" : "MILPITAS", "pop" : 50907, "state" : "CA" }
{ "_id" : "95037", "city" : "MORGAN HILL", "pop" : 31309, "state" : "CA" }
{ "_id" : "99503", "city" : "ANCHORAGE", "pop" : 12534, "state" : "AK" }
>
```

// 6. Find all cities with populations between 23,000 and 150,000 where the state they are in borders the pacific ocean. (screen 4)

```
db.cityinfo.distinct(
  "city",
  {
    $and: [
      {
        "pop": {
          $gte: 23000,
          $lte: 150000
        }
      },
      {
        "state": {
          $in: [
            "CA",
            "OR",
            "WA"
          ]
        }
      }
    ]
  }
)
```

```
}  
]  
}  
)
```

```
~/Documents/SJSU/ClassesSpring2017/CS_185_C_NoSQL/homeworks/homework1 — mongo — bash — Homebrew — tty...  
~/Documents/workspace/PythonLearning — -bash — ttys001 ...17/CS_185_C_NoSQL/homeworks/homework1 — mongo — ttys000 +  
[> db.cityinfo.distinct( "city", { $and: [ { "pop": { $gte: 23000, $lte: 150000 } }, { "state" : { $in: [ "CA", "OR"]  
, "WA" ] } } ] } )  
[  
  "LOS ANGELES",  
  "EAST LOS ANGELES",  
  "COLE",  
  "HAZARD",  
  "BELL GARDENS",  
  "RANCHO DOMINGUEZ",  
  "EAST RANCHO DOMI",  
  "ROSEWOOD",  
  "CULVER CITY",  
  "DOWNEY",  
  "GARDENA",  
  "HOLLY PARK",  
  "HUNTINGTON PARK",  
  "LAWNDALE",  
  "LYNWOOD",  
  "MANHATTAN BEACH",  
  "MAYWOOD",  
  "PALOS VERDES EST",  
  "REDONDO BEACH",  
  "SOUTH GATE",  
  "VENICE",  
  "INGLEWOOD",  
  "LENNOX",  
  "SANTA MONICA",  
  "TORRANCE",  
  "WHITTIER",  
  "LOS NIETOS",  
  "BUENA PARK",  
  "CYPRESS",  
  "LA HABRA HEIGHTS",  
  "LA MIRADA",  
  "MONTEBELLO",
```

```
// get the length of the array returned by distinct()  
db.cityinfo.distinct(  
  "city",  
  {  
    $and: [  
      {  
        "pop": {
```

```

    $gte: 23000,
    $lte: 150000
  }
},
{
  "state": {
    $in: [
      "CA",
      "OR",
      "WA"
    ]
  }
}
]
}
).length

```

The screenshot shows a terminal window with a dark background. At the top, there are window titles: `~/Documents/SJSU/ClassesSpring2017/CS_185_C_NoSQL/homeworks/homework1 — mongo — bash — Homebrew — tty...`, `~/Documents/workspace/PythonLearning — -bash — ttys001`, and `...17/CS_185_C_NoSQL/homeworks/homework1 — mongo — ttys000`. The terminal content shows a MongoDB query being executed in the `mongo` shell:

```
[> db.cityinfo.distinct( "city", { $and: [ { "pop": { $gte: 23000, $lte: 150000 } }, { "state" : { $in: [ "CA", "OR" ] , "WA" ] } } ] } ).length
```

The result of the query is displayed as `408` on the next line. Below the result, there is a faint, dark rectangular area that appears to be a blurred screenshot of a table or document, but its content is illegible.

```

// 7. Find all zip code in San Jose, New Work, or Washington that have a population between
6,000 and 11,000. (screen 5)
// question 7 - find zipcodes
// this output is not that great
// > db.getCollectionNames()

```

```

// [ "cityinfo", "zip_codes" ]
// > db.zip_codes.find()
// { "_id" : "10044", "value" : "10044" }
// { "_id" : "20005", "value" : "20005" }
// { "_id" : "20336", "value" : "20336" }
// { "_id" : "30673", "value" : "30673" }
// { "_id" : "48094", "value" : "48094" }
// { "_id" : "52353", "value" : "52353" }
// { "_id" : "95119", "value" : "95119" }
// { "_id" : "95135", "value" : "95135" }
// { "_id" : "95139", "value" : "95139" }

var zipcodes = db.cityinfo.mapReduce(
  function() { emit( this._id, this._id ) },
  function(key, values) { return { "zipcode": key } },
  {
    query: {
      $and:[
        {
          "city": {
            $in: [
              "SAN JOSE",
              "WASHINGTON",
              "NEW YORK"
            ]
          }
        },
        {
          "pop": {
            $gte: 6000,
            $lte: 11000
          }
        }
      ]
    },
    out: "zip_codes"
  }
)

```

```

// 7. Find all zip code in San Jose, New Work, or Washington that have a population between
6,000 and 11,000. (screen 5)
// question 7 - find zipcodes
// > db.zip_codes.find()
// { "_id" : "zipcode", "value" : { "zipcodes" : [ "10044", "20005", "20336", "30673", "48094",
"52353", "95119", "95135", "95139" ] } }

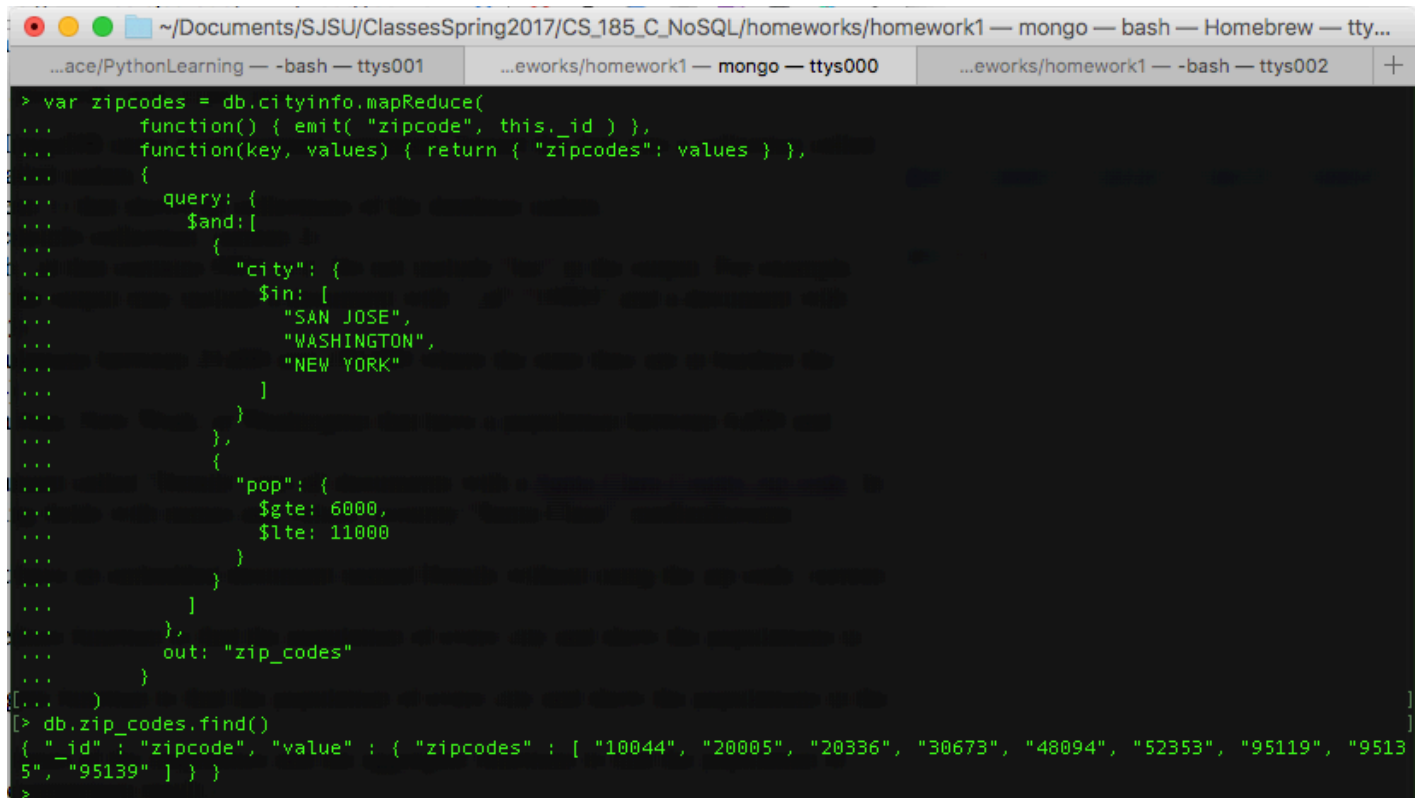
var zipcodes = db.cityinfo.mapReduce(

```

```

function() { emit( "zipcode", this._id ) },
function(key, values) { return { "zipcodes": values } },
{
  query: {
    $and:[
      {
        "city": {
          $in: [
            "SAN JOSE",
            "WASHINGTON",
            "NEW YORK"
          ]
        }
      },
      {
        "pop": {
          $gte: 6000,
          $lte: 11000
        }
      }
    ]
  },
  out: "zip_codes"
}
)

```



The screenshot shows a terminal window with a dark background and green text. The terminal title bar indicates the current directory is ~/Documents/SJSU/ClassesSpring2017/CS_185_C_NoSQL/homeworks/homework1. The terminal shows the execution of a MongoDB mapReduce command on the cityinfo collection, followed by a find() query on the newly created zip_codes collection. The output of the find() query is displayed as a JSON document.

```

> var zipcodes = db.cityinfo.mapReduce(
...   function() { emit( "zipcode", this._id ) },
...   function(key, values) { return { "zipcodes": values } },
...   {
...     query: {
...       $and:[
...         {
...           "city": {
...             $in: [
...               "SAN JOSE",
...               "WASHINGTON",
...               "NEW YORK"
...             ]
...           }
...         },
...         {
...           "pop": {
...             $gte: 6000,
...             $lte: 11000
...           }
...         }
...       ]
...     },
...     out: "zip_codes"
...   }
... )
[... ]
[> db.zip_codes.find()
{ "_id" : "zipcode", "value" : { "zipcodes" : [ "10044", "20005", "20336", "30673", "48094", "52353", "95119", "95135", "95139" ] } }
>

```

```
// using the aggregation pipeline gives a better output
// so I'm thinking of keeping this
```

```
db.cityinfo.aggregate(
{
  $match: {
    $and:[
      {
        "city": {
          $in: [
            "SAN JOSE",
            "WASHINGTON",
            "NEW YORK"
          ]
        }
      },
      {
        "pop": {
          $gte: 6000,
          $lte: 11000
        }
      }
    ]
  },
  {
    $group: {
      "_id": "$city",
      "zipcode": {
        $push: "$_id"
      }
    }
  }
})
```



```
~/Documents/SJSU/ClassesSpring2017/CS_185_C_NoSQL/homeworks/homework1 — mongo — bash — Homebrew — ttys001 — 135x43 — %1
.../workspace/PythonLearning — -bash — ttys000
...QL/homeworks/homework1 — mongo — ttys001
...SQL/homeworks/homework1 — -bash — ttys002
+

> db.cityinfo.aggregate(
... {
...   $match: {
...     $and:[
...       {
...         "city": {
...           $in: [
...             "SAN JOSE",
...             "WASHINGTON",
...             "NEW YORK"
...           ]
...         },
...       },
...       {
...         "pop": {
...           $gte: 6000,
...           $lte: 11000
...         }
...       }
...     ]
...   },
...   {
...     $group: {
...       "_id": "$city",
...       "zipcode": {
...         $push: "$_id"
...       }
...     }
...   }
... }
[... ]
{ "_id" : "SAN JOSE", "zipcode" : [ "95119", "95135", "95139" ] }
{ "_id" : "WASHINGTON", "zipcode" : [ "20005", "20336", "30673", "40094", "52353" ] }
{ "_id" : "NEW YORK", "zipcode" : [ "10044" ] }
>
```

// 8. Add an embedded document called "Details" into all documents with a Santa Clara County zip code.

// In Details, add the following fields with names and values: {county:"Santa Clara", medianIncome: 93500}. (screen 6)

```
db.cityinfo.updateMany(
{
  "_id": {
    $in: [
      "95009",
      "95008",
      "95013",
      "95014",
      "95020",
```

"94085",
"95023",
"94087",
"94086",
"94089",
"94088",
"95031",
"95030",
"95033",
"95032",
"95035",
"95037",
"94301",
"95042",
"94303",
"95044",
"94305",
"95050",
"94304",
"95046",
"94306",
"95051",
"95054",
"95070",
"95103",
"95108",
"95111",
"95110",
"95113",
"95112",
"95117",
"95116",
"95119",
"95118",
"95121",
"95120",
"95123",
"95122",
"95125",
"95124",
"95127",
"95126",
"95129",
"95128",
"95131",
"95130",
"95133",
"95132",
"95135",

```
"95134",
"95136",
"95139",
"95138",
"95141",
"95140",
"95148",
"94550",
"95151",
"95150",
"94022",
"94024",
"95190",
"94028",
"94035",
"94040",
"94042",
"94041",
"94043",
"95002"
]
}
},
{
  $set: {
    "Details": {
      "county": "Santa Clara",
      "medianIncome": 93500
    }
  }
},
{
  upsert: true
}
)
```

```
~/Documents/SJSU/ClassesSpring2017/CS_185_C_NoSQL/homeworks/homework1 — mongo — bash — Homebrew — tty...
...ace/PythonLearning — -bash — ttys000  ...eworks/homework1 — -bash — ttys001  ...eworks/homework1 — mongo — ttys002  +
> db.cityinfo.updateMany(
... {
...   "_id": {
...     "$in": [
...       "95009",
...       "95008",
...       "95013",
...       "95014",
...       "95020",
...       "94085",
...       "95023",
...       "94087",
...       "94086",
...       "94089",
...       "94088",
...       "95031",
...       "95030",
...       "95033",
...       "95032",
...       "95035",
...       "95037",
...       "94301",
...       "95042",
...       "94303",
...       "95044",
...       "94305",
...       "95050",
...       "94304",
...       "95046",
...       "94306",
...       "95051",
...       "95054",
...       "95070",
...       "95103",
...       "95108",
...     ]
...   }
... }
```

```
~/Documents/SJSU/ClassesSpring2017/CS_185_C_NoSQL/homeworks/homework1 — mongo — bash — Homebrew — tty...
...ace/PythonLearning — -bash — ttys000  ...eworks/homework1 — -bash — ttys001  ...eworks/homework1 — mongo — ttys002  +
...   "95132",
...   "95135",
...   "95134",
...   "95136",
...   "95139",
...   "95138",
...   "95141",
...   "95140",
...   "95148",
...   "94550",
...   "95151",
...   "95150",
...   "94022",
...   "94024",
...   "95190",
...   "94028",
...   "94035",
...   "94040",
...   "94042",
...   "94041",
...   "94043",
...   "95002"
... ]
... }
... },
... {
...   $set: {
...     "Details": {
...       "county": "Santa Clara",
...       "medianIncome": 93500
...     }
...   }
... }
[... ]
{ "acknowledged" : true, "matchedCount" : 61, "modifiedCount" : 60 }
```

```
~/Documents/SJSU/ClassesSpring2017/CS_185_C_NoSQL/homeworks/homework1 — mongo — bash — Homebrew — tty...
...ace/PythonLearning — -bash — ttys000  ...eworks/homework1 — -bash — ttys001  ...eworks/homework1 — mongo — ttys002  +
[> db.cityinfo.find( { "_id": { "$in": [ "95009", "95008", "95013", "95014", ]
    "95020", "94085", "95023", "94087", "94086", "94089", "94088",
    "95031", "95030", "95033", "95032", "95035", "95037", "94
301", "95042", "94303", "95044", "94305", "95050", "94304",
"95046", "94306", "95051", "95054", "95070", "95103", "95108",
"95111", "95110", "95113", "95112", "95117", "95116", "95119",
"95118", "95121", "95120", "95123", "95122", "95125", "95124",
"95127", "95126", "95129", "95128", "95131", "95130", "9513
3", "95132", "95135", "95134", "95136", "95139", "95138", "9
5141", "95140", "95148", "94550", "95151", "95150", "94022",
"94024", "95190", "94028", "94035", "94040", "94042", "94041",
"94043", "95002" ] } }).limit(10)
{ "_id": "94022", "city": "LOS ALTOS", "loc": [ -122.125754, 37.381432 ], "pop": 17366, "state": "CA", "Details
": { "county": "Santa Clara", "medianIncome": 93500 } }
{ "_id": "94024", "city": "LOS ALTOS", "loc": [ -122.086205, 37.354745 ], "pop": 20795, "state": "CA", "Details
": { "county": "Santa Clara", "medianIncome": 93500 } }
{ "_id": "94028", "city": "LADERA", "loc": [ -122.208131, 37.378859 ], "pop": 6379, "state": "CA", "Details":
{ "county": "Santa Clara", "medianIncome": 93500 } }
{ "_id": "94035", "city": "MOFFETT FIELD", "loc": [ -122.051944, 37.41001 ], "pop": 790, "state": "CA", "Detail
s": { "county": "Santa Clara", "medianIncome": 93500 } }
{ "_id": "94040", "city": "MOUNTAIN VIEW", "loc": [ -122.087983, 37.385532 ], "pop": 26969, "state": "CA", "Det
ails": { "county": "Santa Clara", "medianIncome": 93500 } }
{ "_id": "94041", "city": "MOUNTAIN VIEW", "loc": [ -122.078341, 37.389347 ], "pop": 13438, "state": "CA", "Det
ails": { "county": "Santa Clara", "medianIncome": 93500 } }
{ "_id": "94043", "city": "MOUNTAIN VIEW", "loc": [ -122.077468, 37.405567 ], "pop": 28592, "state": "CA", "Det
ails": { "county": "Santa Clara", "medianIncome": 93500 } }
{ "_id": "94086", "city": "SUNNYVALE", "loc": [ -122.023771, 37.376407 ], "pop": 56215, "state": "CA", "Details
": { "county": "Santa Clara", "medianIncome": 93500 } }
{ "_id": "94087", "city": "SUNNYVALE", "loc": [ -122.034859, 37.350214 ], "pop": 47813, "state": "CA", "Details
": { "county": "Santa Clara", "medianIncome": 93500 } }
{ "_id": "94089", "city": "SUNNYVALE", "loc": [ -122.000637, 37.398255 ], "pop": 13522, "state": "CA", "Details
": { "county": "Santa Clara", "medianIncome": 93500 } }
>
```

// 9. Find all documents that have an embedded document named Details without using the zip code. (screen 7)

```
db.cityinfo.find(
{
  "Details": {
    $exists: true
  }
})
```

// limiting to 10

```
db.cityinfo.find(
```

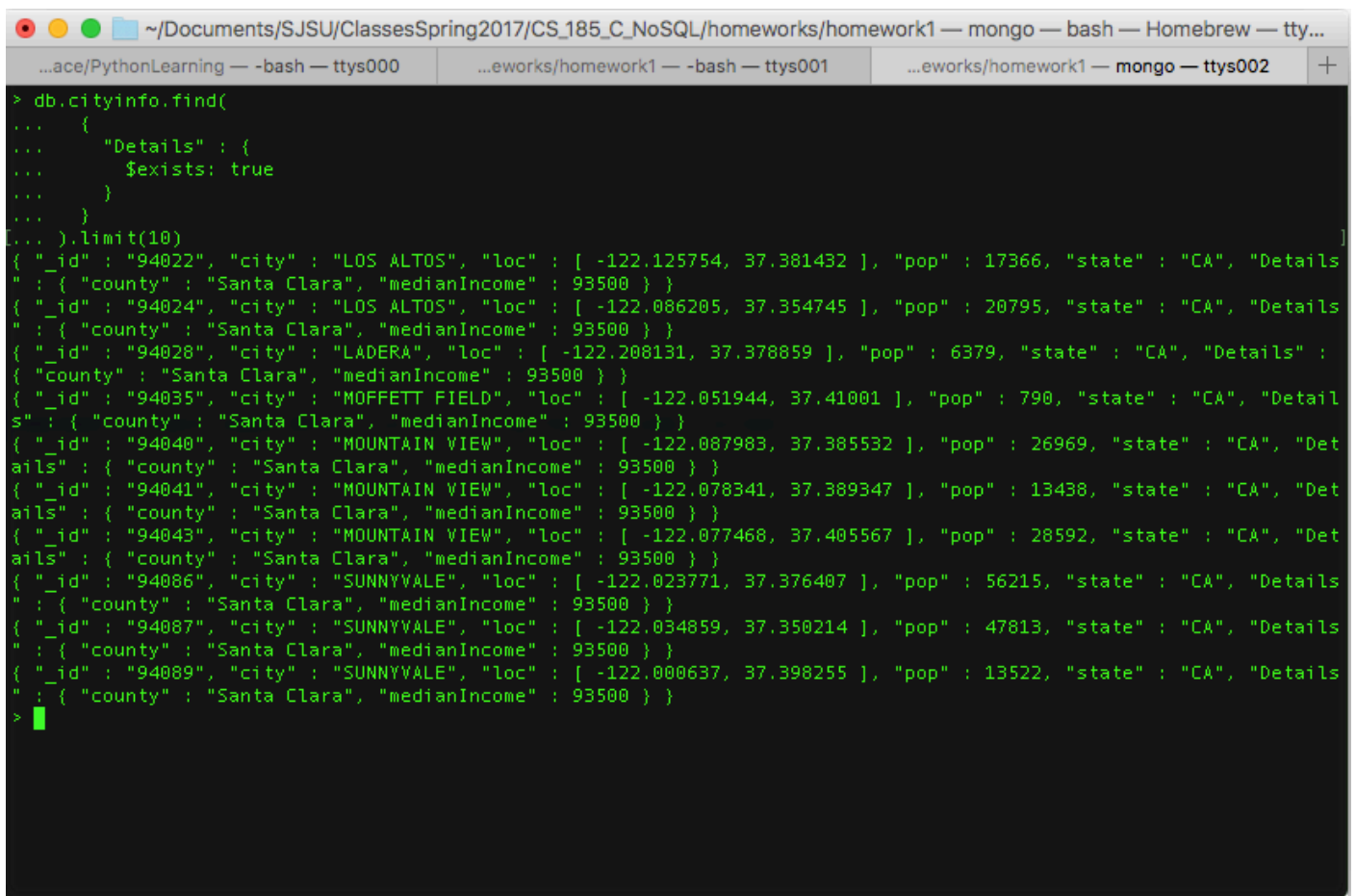
```

{
  "Details" : {
    $exists: true
  }
}
).limit(10)

// verifying length

db.cityinfo.find(
{
  "Details" : {
    $exists: true
  }
}
).toArray().length

```



The screenshot shows a terminal window with three tabs: `...ace/PythonLearning — -bash — ttys000`, `...eworks/homework1 — -bash — ttys001`, and `...eworks/homework1 — mongo — ttys002`. The active tab is `mongo`. The user has entered the following commands:

```

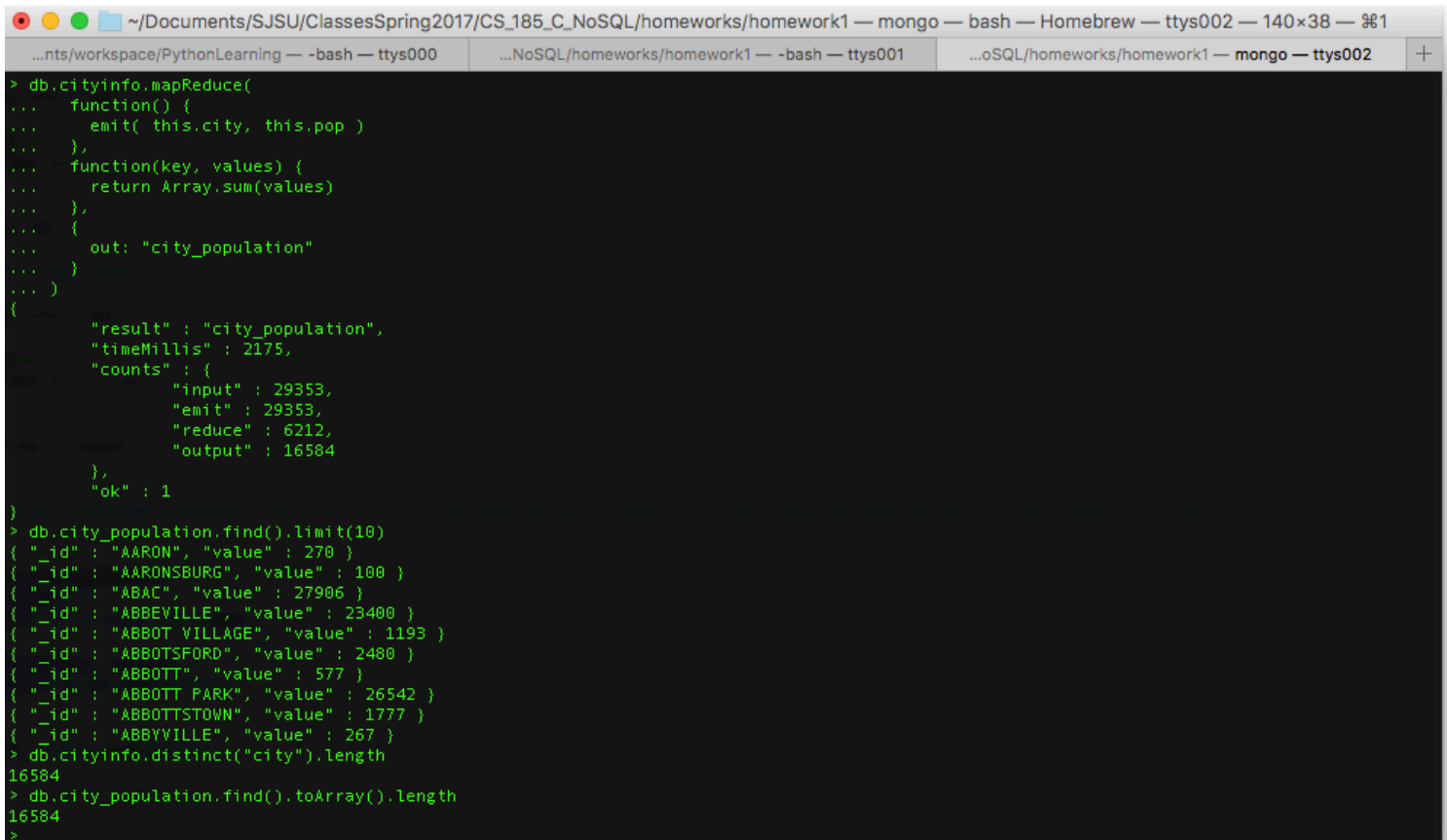
> db.cityinfo.find(
...   {
...     "Details" : {
...       $exists: true
...     }
...   }
... ).limit(10)

```

The output shows 10 documents from the `cityinfo` collection, each containing fields like `_id`, `county`, `city`, `loc`, `pop`, `state`, and `Details`. The documents are truncated in the image, but the structure is consistent across all 10 results.

// 10. Use a MongoDB mapreduce function to find the population of every city and show the populations in the output. (screen 8)

```
db.cityinfo.mapReduce(
  function() {
    emit( this.city, this.pop )
  },
  function(key, values) {
    return Array.sum(values)
  },
  {
    out: "city_population"
  }
)
```



The screenshot shows a terminal window with a dark background and light green text. The terminal title bar indicates the current directory is ~/Documents/SJSU/ClassesSpring2017/CS_185_C_NoSQL/homeworks/homework1 and the shell is mongo. The user has executed a mapreduce operation on the cityinfo collection, emitting city names and populations, and reducing them by summing the populations for each city. The result is stored in a collection named city_population. The terminal output shows the mapreduce command, its execution details (including input, emit, reduce, and output counts), and a list of the resulting documents. The documents are in the form { "_id": "city_name", "value": population }. The cities listed include AARON, AARONSBURG, ABAC, ABBEVILLE, ABBOT VILLAGE, ABBOTSFORD, ABBOTT, ABBOTT PARK, ABBOTTSTOWN, and ABBYVILLE. The total number of documents is 16584, which is also confirmed by the length of the distinct city names and the length of the result array.

```
> db.cityinfo.mapReduce(
...   function() {
...     emit( this.city, this.pop )
...   },
...   function(key, values) {
...     return Array.sum(values)
...   },
...   {
...     out: "city_population"
...   }
... )
{
  "result" : "city_population",
  "timeMillis" : 2175,
  "counts" : {
    "input" : 29353,
    "emit" : 29353,
    "reduce" : 6212,
    "output" : 16584
  },
  "ok" : 1
}
> db.city_population.find().limit(10)
{ "_id" : "AARON", "value" : 270 }
{ "_id" : "AARONSBURG", "value" : 100 }
{ "_id" : "ABAC", "value" : 27906 }
{ "_id" : "ABBEVILLE", "value" : 23400 }
{ "_id" : "ABBOT VILLAGE", "value" : 1193 }
{ "_id" : "ABBOTSFORD", "value" : 2480 }
{ "_id" : "ABBOTT", "value" : 577 }
{ "_id" : "ABBOTT PARK", "value" : 26542 }
{ "_id" : "ABBOTTSTOWN", "value" : 1777 }
{ "_id" : "ABBYVILLE", "value" : 267 }
> db.cityinfo.distinct("city").length
16584
> db.city_population.find().toArray().length
16584
>
```

// 11. Use a MongoDB aggregate function to find the population of every city and show the populations in the output. (screen 9)
 // used \$sort for getting similar output as my mapReduce() query
 // to compare
 // map reduce had sorted result by default

```
db.cityinfo.aggregate(
[
  {
    $sort: {
      "city": -1
    }
  },
  {
    $group: {
      "_id": "$city",
      "population": {
        $sum: "$pop"
      }
    }
  }
]
)
```

The screenshot shows a terminal window with a dark background. At the top, there are window title bars for a file explorer and three terminal tabs. The active terminal tab is titled 'mongo — ttys002'. The terminal displays the execution of a MongoDB aggregation command and its output. The output is a list of JSON objects, each representing a city and its population. The cities are sorted in descending order of population. At the bottom, the terminal shows the command to convert the aggregation result to an array and count the number of elements, which is 16584.

```
> db.cityinfo.aggregate(
... [
...   {
...     $sort: {
...       "city": -1
...     }
...   },
...   {
...     $group: {
...       "_id": "$city",
...       "population": {
...         $sum: "$pop"
...       }
...     }
...   }
... ]
... )
{ "_id": "AARON", "population": 270 }
{ "_id": "AARONSBURG", "population": 100 }
{ "_id": "ABAC", "population": 27906 }
{ "_id": "ABBEVILLE", "population": 23400 }
{ "_id": "ABBOT VILLAGE", "population": 1193 }
{ "_id": "ABBOTT", "population": 577 }
{ "_id": "ABBOTT PARK", "population": 26542 }
{ "_id": "ABBOTTSTOWN", "population": 1777 }
{ "_id": "ABELL", "population": 601 }
{ "_id": "ABERDEEN PROVING", "population": 5294 }
{ "_id": "ABIE", "population": 282 }
{ "_id": "ABINGDON", "population": 42334 }
{ "_id": "ABITA SPRINGS", "population": 2659 }
{ "_id": "ABRAMS", "population": 1712 }
{ "_id": "ABSARAKA", "population": 124 }
{ "_id": "ABSAROKEE", "population": 1330 }
{ "_id": "ACADEMY", "population": 2425 }
{ "_id": "ACCOMAC", "population": 2562 }
{ "_id": "ACCORD", "population": 2695 }
{ "_id": "ACEQUIA", "population": 9761 }
Type "it" for more
> db.cityinfo.aggregate( [ { $sort: { "city": -1 } }, { $group: { "_id": "$city",
"population": { $sum: "$pop" } } } ].toArray().length
16584
>
```


// 12. Compare the execution times of the mapreduce and the aggregate functions to find the population of every city. (Write your comparison result.)

// The mapReduce() took 1633 milliseconds (from the query execution explain document)

```
// {
//   "result" : "city_population",
//   "timeMillis" : 1633,
//   "counts" : {
//     "input" : 29353,
//     "emit" : 29353,
//     "reduce" : 6212,
//     "output" : 16584
//   },
//   "ok" : 1
// }
```

// and the aggregation pipeline or the aggregate() took 70 milliseconds (from the snapshot of the profiler)

// > db.system.profile.find().limit(1).pretty()

```
// {
//   "op" : "command",
//   "ns" : "usdata.cityinfo",
//   "command" : {
//     "aggregate" : "cityinfo",
//     "pipeline" : [
//       {
//         "$sort" : {
//           "city" : -1
//         }
//       },
//       {
//         "$group" : {
//           "_id" : "$city",
//           "population" : {
//             "$sum" : "$pop"
//           }
//         }
//       }
//     ],
//     "cursor" : {
//       }
//     },
//     "cursorid" : 36591943087,
//     "keysExamined" : 0,
```

```
// "docsExamined" : 29353,
// "hasSortStage" : true,
// "numYield" : 229,
// "locks" : {
//   "Global" : {
//     "acquireCount" : {
//       "r" : NumberLong(468)
//     }
//   },
//   "Database" : {
//     "acquireCount" : {
//       "r" : NumberLong(234)
//     }
//   },
//   "Collection" : {
//     "acquireCount" : {
//       "r" : NumberLong(233)
//     }
//   }
// },
// "nreturned" : 101,
// "responseLength" : 4346,
// "protocol" : "op_command",
// "millis" : 70,
// "planSummary" : "COLLSCAN",
// "ts" : ISODate("2017-02-25T08:04:14.647Z"),
// "client" : "127.0.0.1",
// "appName" : "MongoDB Shell",
// "allUsers" : [ ],
// "user" : ""
// }
```

// Looking at the results, aggregation pipeline is definitely faster, considering I even did a sort!