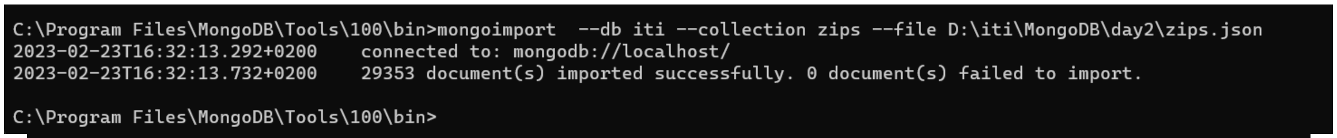
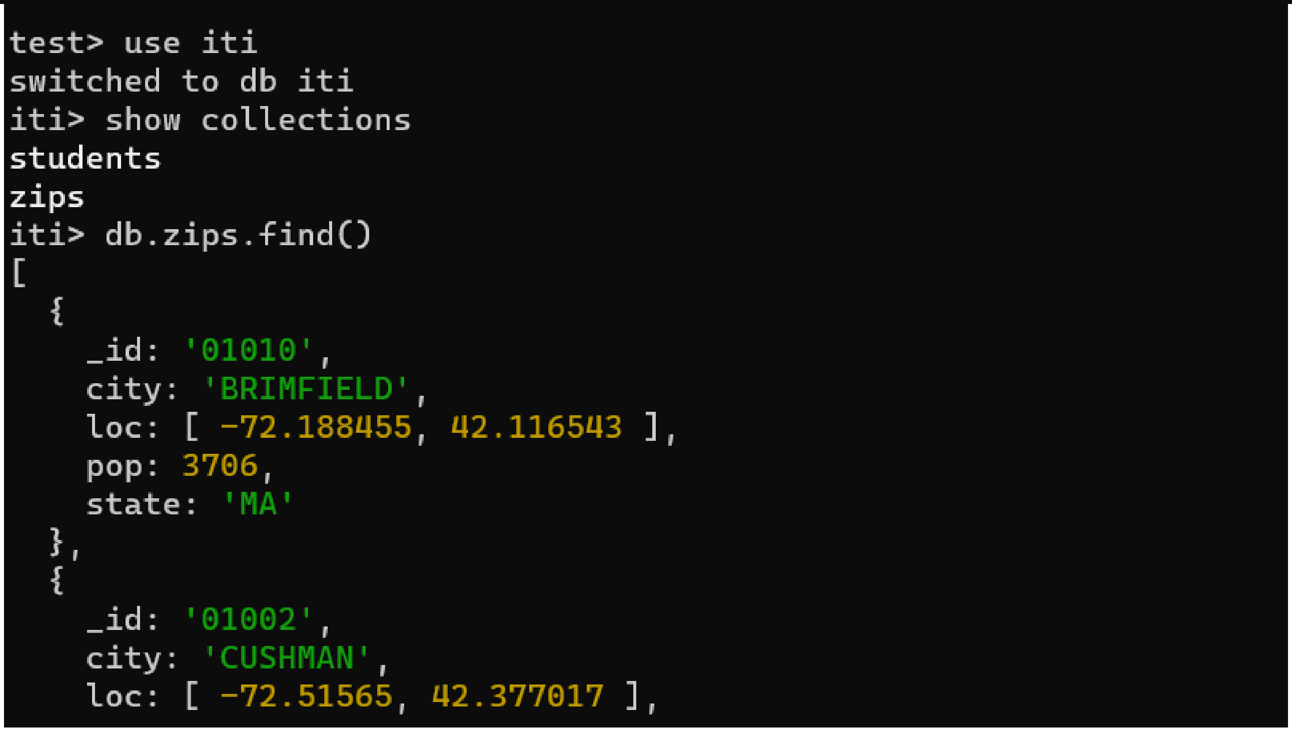
MongoDB Lab2

1 - Download the following json file and import it into a collection named “zips” into “iti” database

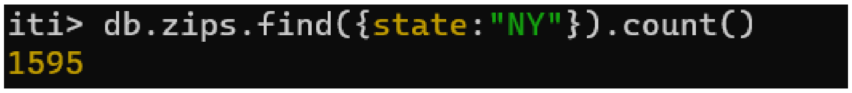


```
C:\Program Files\MongoDB\Tools\100\bin>mongoimport  --db iti --collection zips --file D:\iti\MongoDB\day2\zips.json
2023-02-23T16:32:13.292+0200    connected to: mongodb://localhost/
2023-02-23T16:32:13.732+0200    29353 document(s) imported successfully. 0 document(s) failed to import.

C:\Program Files\MongoDB\Tools\100\bin>
```



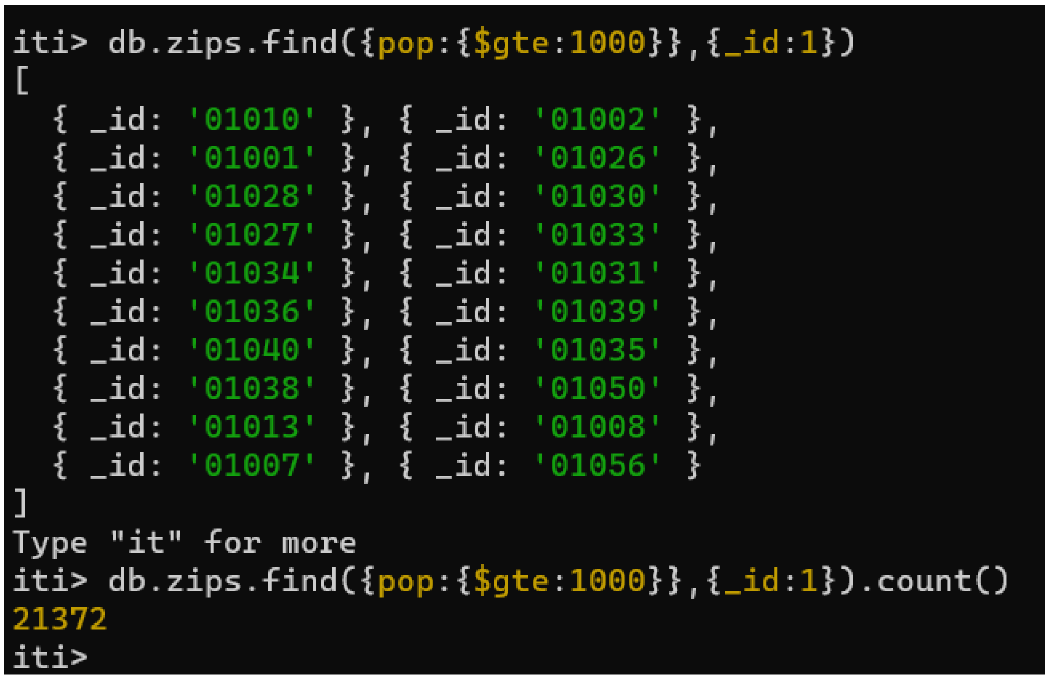
```
test> use iti
switched to db iti
iti> show collections
students
zips
iti> db.zips.find()
[
  {
    _id: '01010',
    city: 'BRIMFIELD',
    loc: [ -72.188455, 42.116543 ],
    pop: 3706,
    state: 'MA'
  },
  {
    _id: '01002',
    city: 'CUSHMAN',
    loc: [ -72.51565, 42.377017 ],
```

2 – find all documents which contains data related to “NY” state



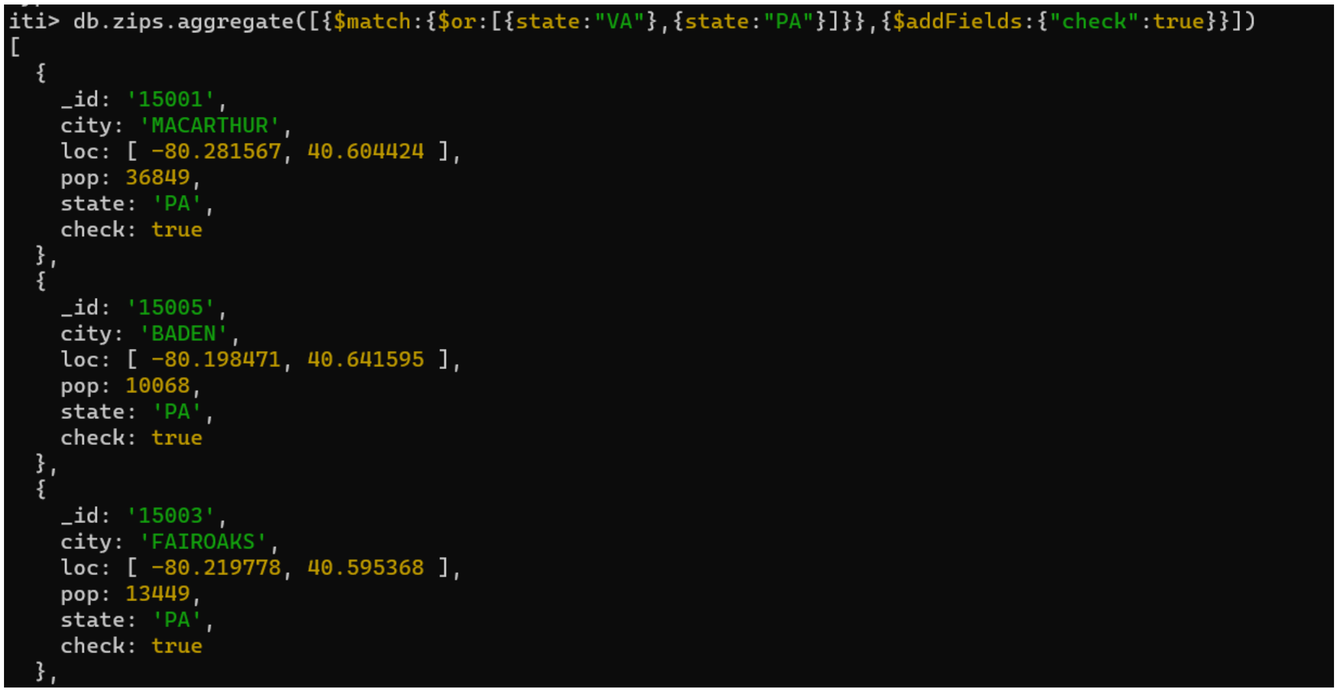
```
iti> db.zips.find({state:"NY"}).count()
1595
```

3 – find all zip codes whose population is greater than or equal to 1000



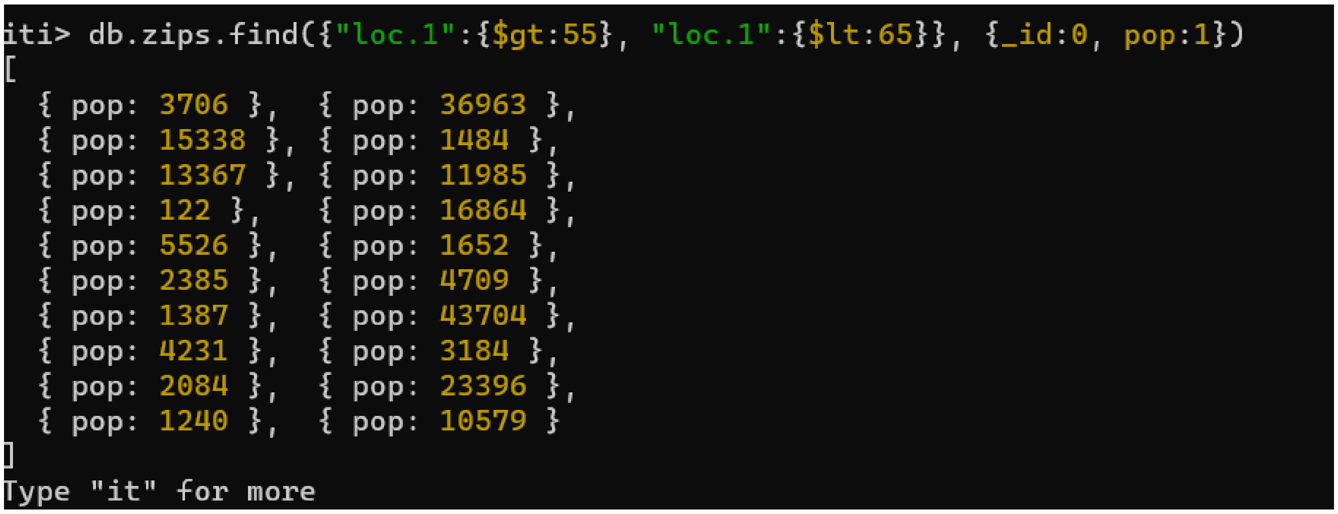
```
iti> db.zips.find({pop:{$gte:1000}},{_id:1})
[
  { _id: '01010' }, { _id: '01002' },
  { _id: '01001' }, { _id: '01026' },
  { _id: '01028' }, { _id: '01030' },
  { _id: '01027' }, { _id: '01033' },
  { _id: '01034' }, { _id: '01031' },
  { _id: '01036' }, { _id: '01039' },
  { _id: '01040' }, { _id: '01035' },
  { _id: '01038' }, { _id: '01050' },
  { _id: '01013' }, { _id: '01008' },
  { _id: '01007' }, { _id: '01056' }
]
Type "it" for more
iti> db.zips.find({pop:{$gte:1000}},{_id:1}).count()
21372
iti>
```

4 – add a new boolean field called "check" and set its value to true for "PA" and "VA" state



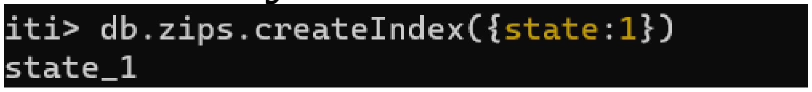
```
iti> db.zips.aggregate([{$match:{$or:[{state:"VA"},{state:"PA"}]}},{$addFields:{"check":true}}])
[
  {
    _id: '15001',
    city: 'MACARTHUR',
    loc: [ -80.281567, 40.604424 ],
    pop: 36849,
    state: 'PA',
    check: true
  },
  {
    _id: '15005',
    city: 'BADEN',
    loc: [ -80.198471, 40.641595 ],
    pop: 10068,
    state: 'PA',
    check: true
  },
  {
    _id: '15003',
    city: 'FAIROAKS',
    loc: [ -80.219778, 40.595368 ],
    pop: 13449,
    state: 'PA',
    check: true
  },
```

5 – using zip codes find all cities whose latitude is between 55 and 65 and show the population only.



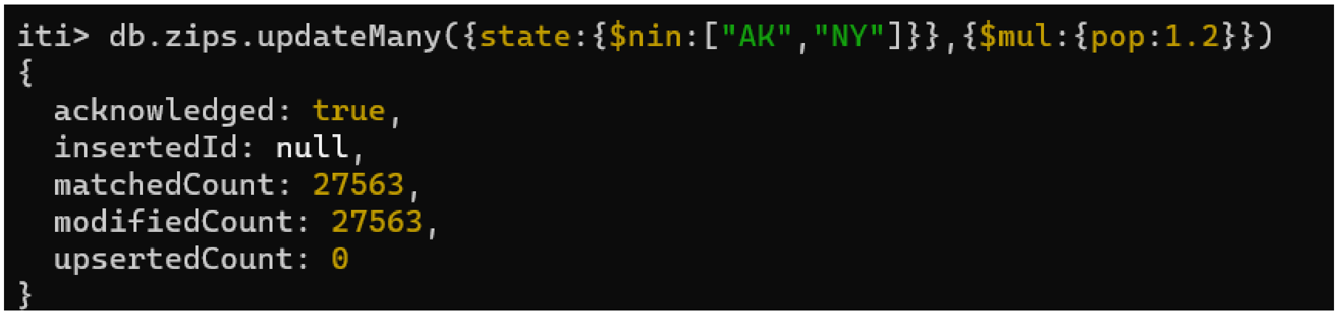
```
iti> db.zips.find({"loc.1":{$gt:55}, "loc.1":{$lt:65}}, {_id:0, pop:1})
[
  { pop: 3706 },  { pop: 36963 },
  { pop: 15338 }, { pop: 1484 },
  { pop: 13367 }, { pop: 11985 },
  { pop: 122 },   { pop: 16864 },
  { pop: 5526 },  { pop: 1652 },
  { pop: 2385 },  { pop: 4709 },
  { pop: 1387 },  { pop: 43704 },
  { pop: 4231 },  { pop: 3184 },
  { pop: 2084 },  { pop: 23396 },
  { pop: 1240 },  { pop: 10579 }
]
Type "it" for more
```

6 – create index for states to be able to select it quickly and check any query explain using the index only.



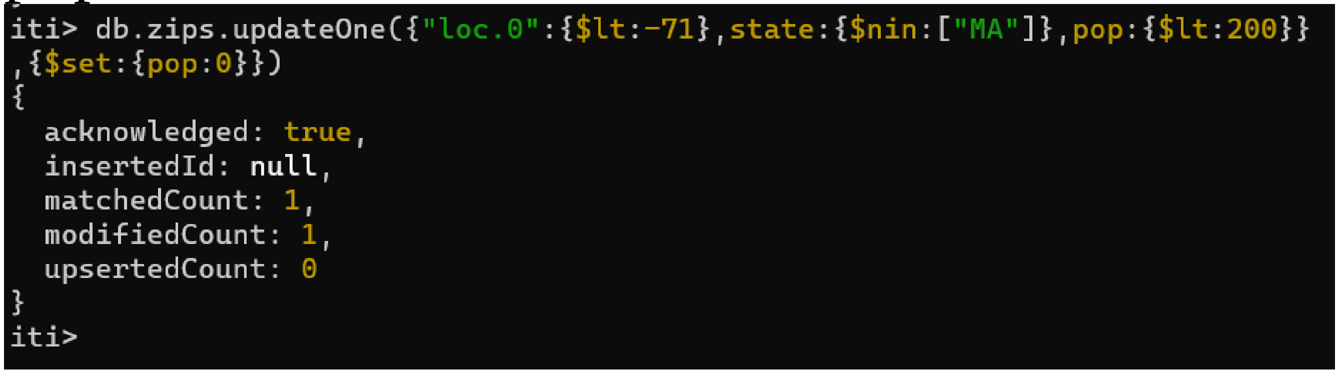
```
iti> db.zips.createIndex({state:1})
state_1
```

7 – increase the population by 0.2 for all cities which doesn't located in “AK” nor “NY”



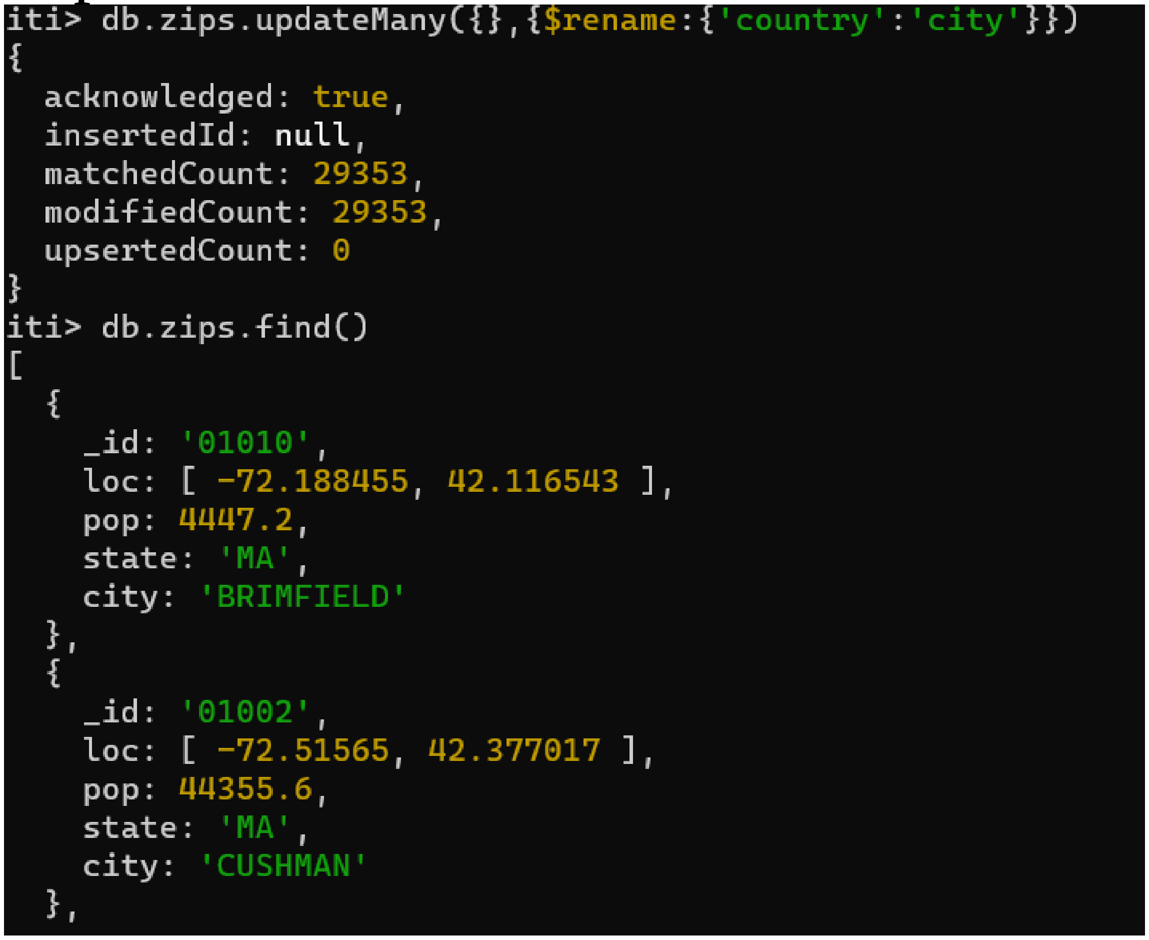
```
iti> db.zips.updateMany({state:{$nin:["AK","NY"]}},{$mul:{pop:1.2}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 27563,
  modifiedCount: 27563,
  upsertedCount: 0
}
```

8 – update only one city whose longitude is lower than -71 and is not located in “MA” state, set its population to 0 if zipcode population less than 200.



```
iti> db.zips.updateOne({"loc.0":{$lt:-71},state:{$nin:["MA"]},pop:{$lt:200}}
,{$set:{pop:0}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
iti>
```

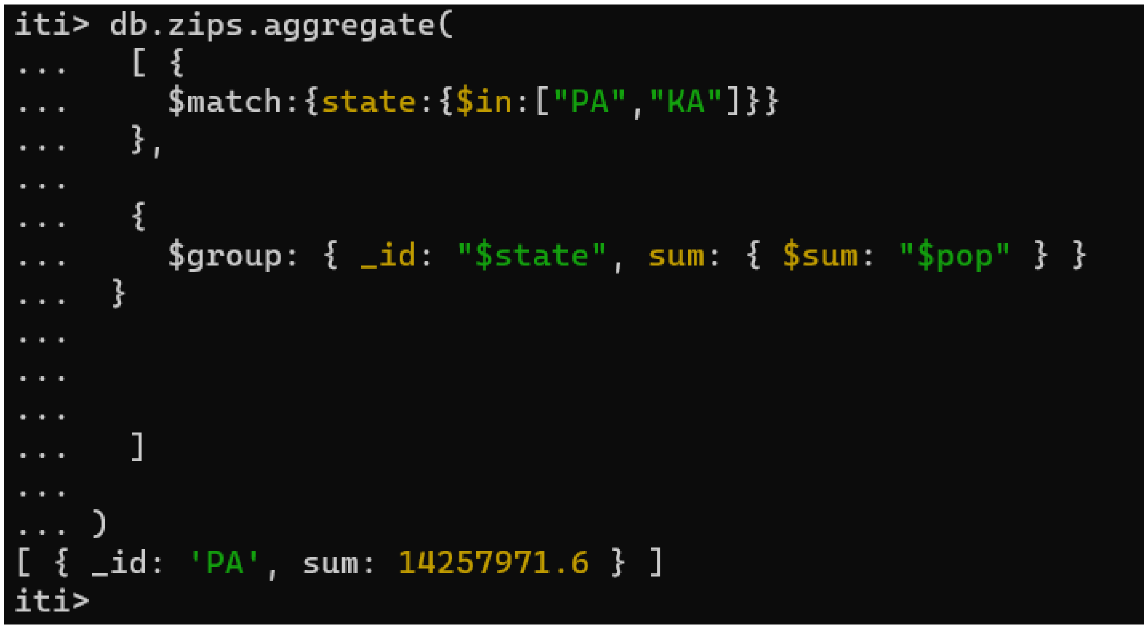
9 – update all documents whose city field is a string, rename its city field to be country and if there isn't any, add new document the same as the first documet in the database but change the _id to avoid
duplications.



```
iti> db.zips.updateMany({},{$rename:{'country':'city'}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 29353,
  modifiedCount: 29353,
  upsertedCount: 0
}
iti> db.zips.find()
[
  {
    _id: '01010',
    loc: [ -72.188455, 42.116543 ],
    pop: 4447.2,
    state: 'MA',
    city: 'BRIMFIELD'
  },
  {
    _id: '01002',
    loc: [ -72.51565, 42.377017 ],
    pop: 44355.6,
    state: 'MA',
    city: 'CUSHMAN'
  },
```

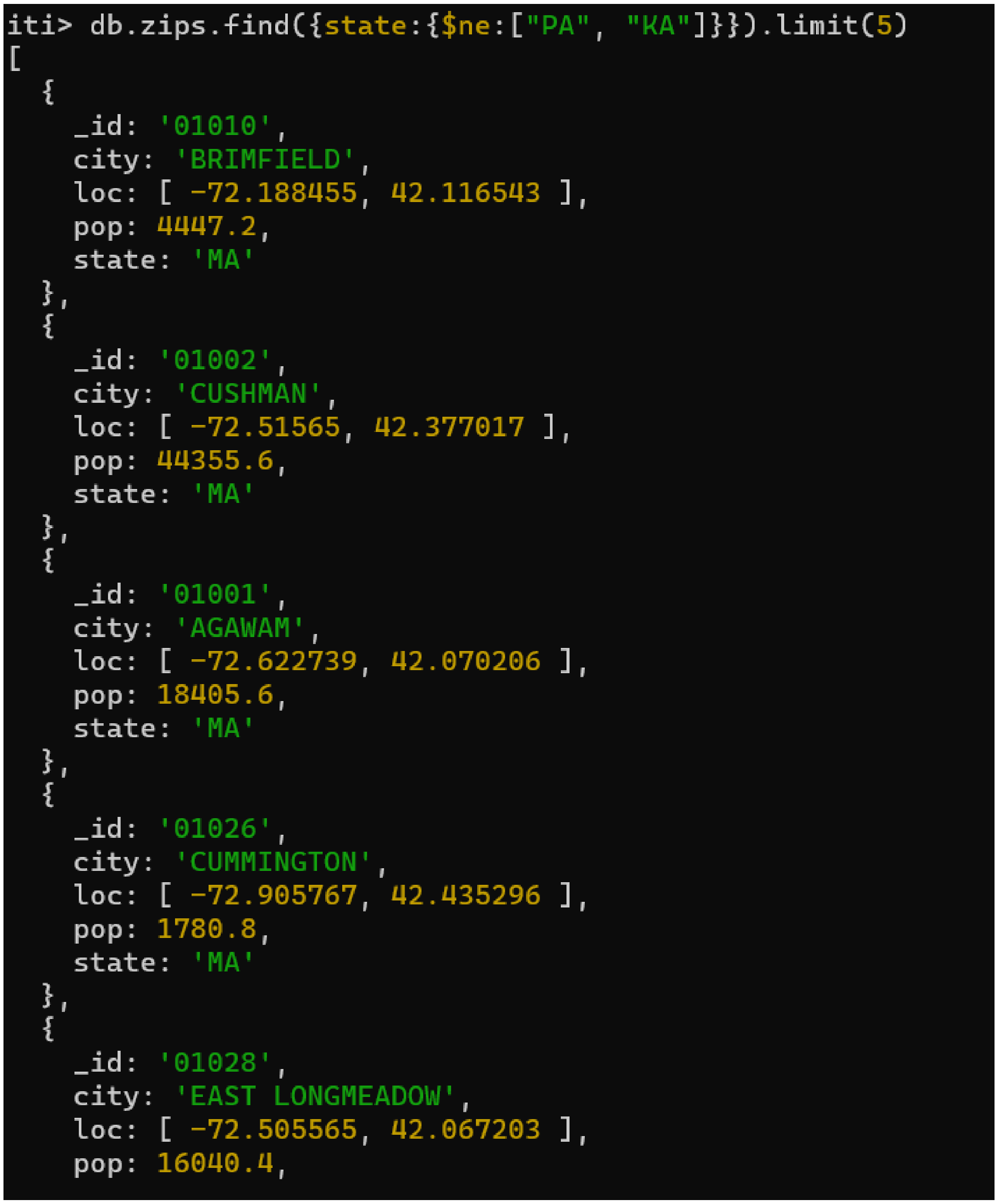
part2

1. Get sum of population that state in PA, KA



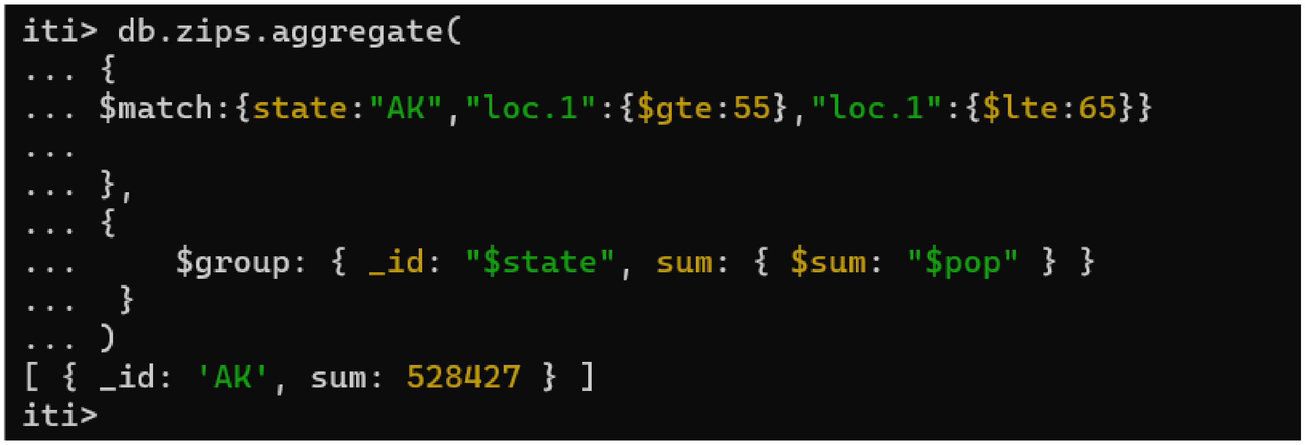
```
iti> db.zips.aggregate(
...    [ {
...      $match:{state:{$in:["PA","KA"]}}
...    },
...
...    {
...      $group: { _id: "$state", sum: { $sum: "$pop" } }
...   }
...
...
...
...    ]
...
... )
[ { _id: 'PA', sum: 14257971.6 } ]
iti>
```

2. Get only 5 documents that state not equal to PA, KA



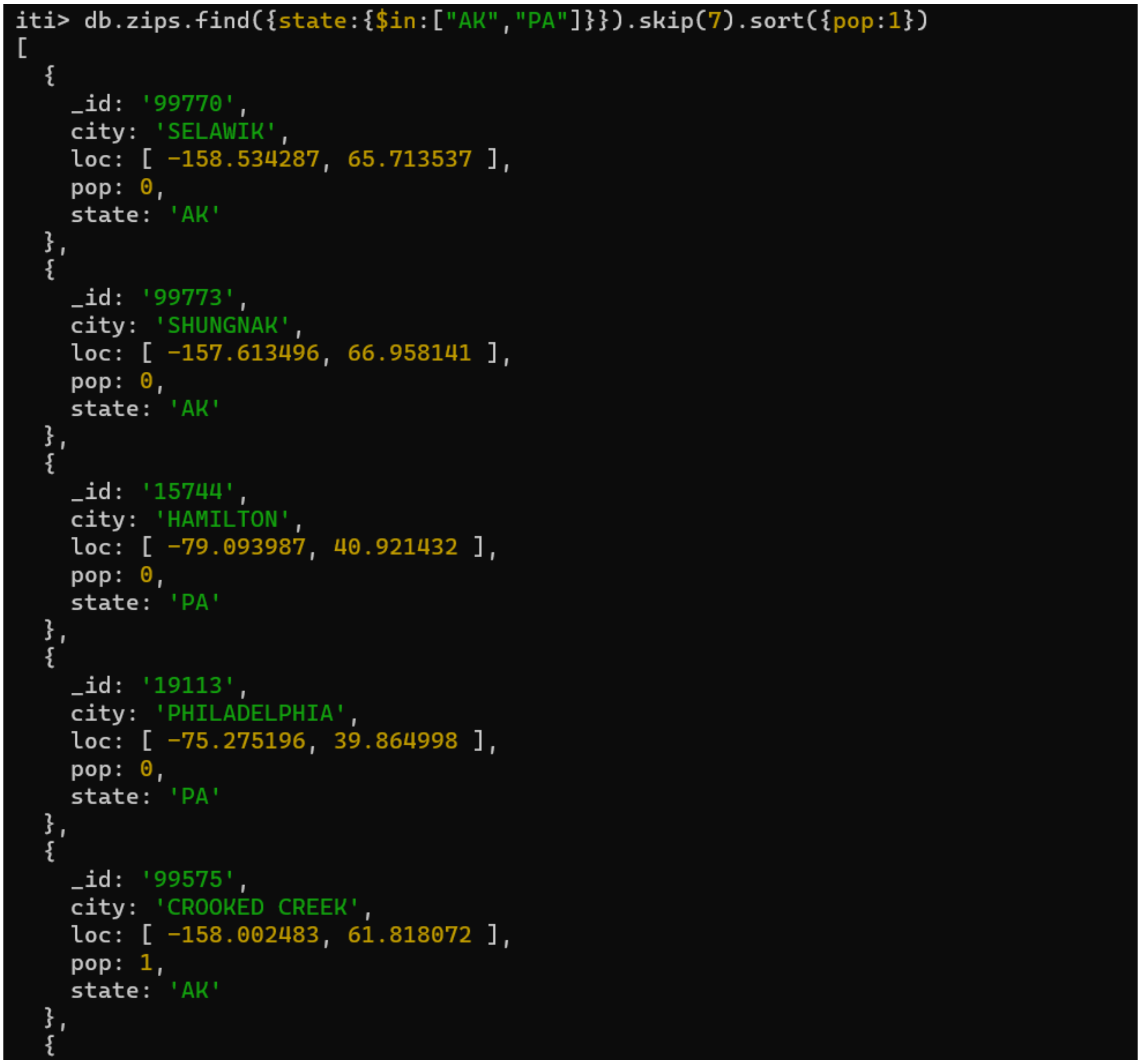
```
iti> db.zips.find({state:{$ne:["PA", "KA"]}}).limit(5)
[
  {
    _id: '01010',
    city: 'BRIMFIELD',
    loc: [ -72.188455, 42.116543 ],
    pop: 4447.2,
    state: 'MA'
  },
  {
    _id: '01002',
    city: 'CUSHMAN',
    loc: [ -72.51565, 42.377017 ],
    pop: 44355.6,
    state: 'MA'
  },
  {
    _id: '01001',
    city: 'AGAWAM',
    loc: [ -72.622739, 42.070206 ],
    pop: 18405.6,
    state: 'MA'
  },
  {
    _id: '01026',
    city: 'CUMMINGTON',
    loc: [ -72.905767, 42.435296 ],
    pop: 1780.8,
    state: 'MA'
  },
  {
    _id: '01028',
    city: 'EAST LONGMEADOW',
    loc: [ -72.505565, 42.067203 ],
    pop: 16040.4,
```

3. Get sum of population that state equal to AK and their latitude between 55, 65



```
iti> db.zips.aggregate(
... {
... $match:{state:"AK","loc.1":{$gte:55},"loc.1":{$lte:65}}
...
... },
... {
...     $group: { _id: "$state", sum: { $sum: "$pop" } }
...  }
... )
[ { _id: 'AK', sum: 528427 } ]
iti>
```

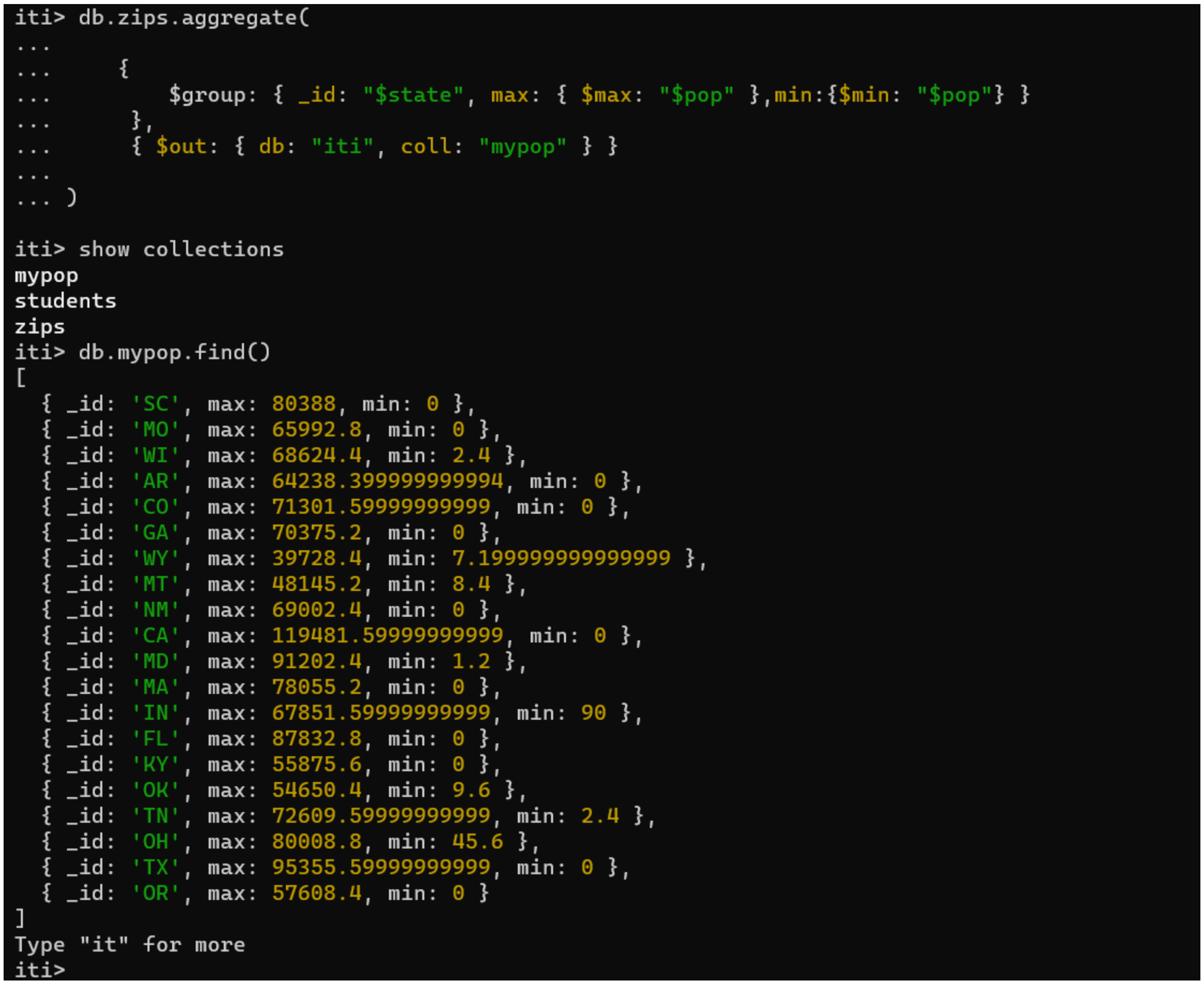
4. Sort Population of document that state in AK, PA and skip first 7 document



```
iti> db.zips.find({state:{$in:["AK","PA"]}}).skip(7).sort({pop:1})
[
  {
    _id: '99770',
    city: 'SELAWIK',
    loc: [ -158.534287, 65.713537 ],
    pop: 0,
    state: 'AK'
  },
  {
    _id: '99773',
    city: 'SHUNGNAK',
    loc: [ -157.613496, 66.958141 ],
    pop: 0,
    state: 'AK'
  },
  {
    _id: '15744',
    city: 'HAMILTON',
    loc: [ -79.093987, 40.921432 ],
    pop: 0,
    state: 'PA'
  },
  {
    _id: '19113',
    city: 'PHILADELPHIA',
    loc: [ -75.275196, 39.864998 ],
    pop: 0,
    state: 'PA'
  },
  {
    _id: '99575',
    city: 'CROOKED CREEK',
    loc: [ -158.002483, 61.818072 ],
    pop: 1,
    state: 'AK'
  },
  {
```

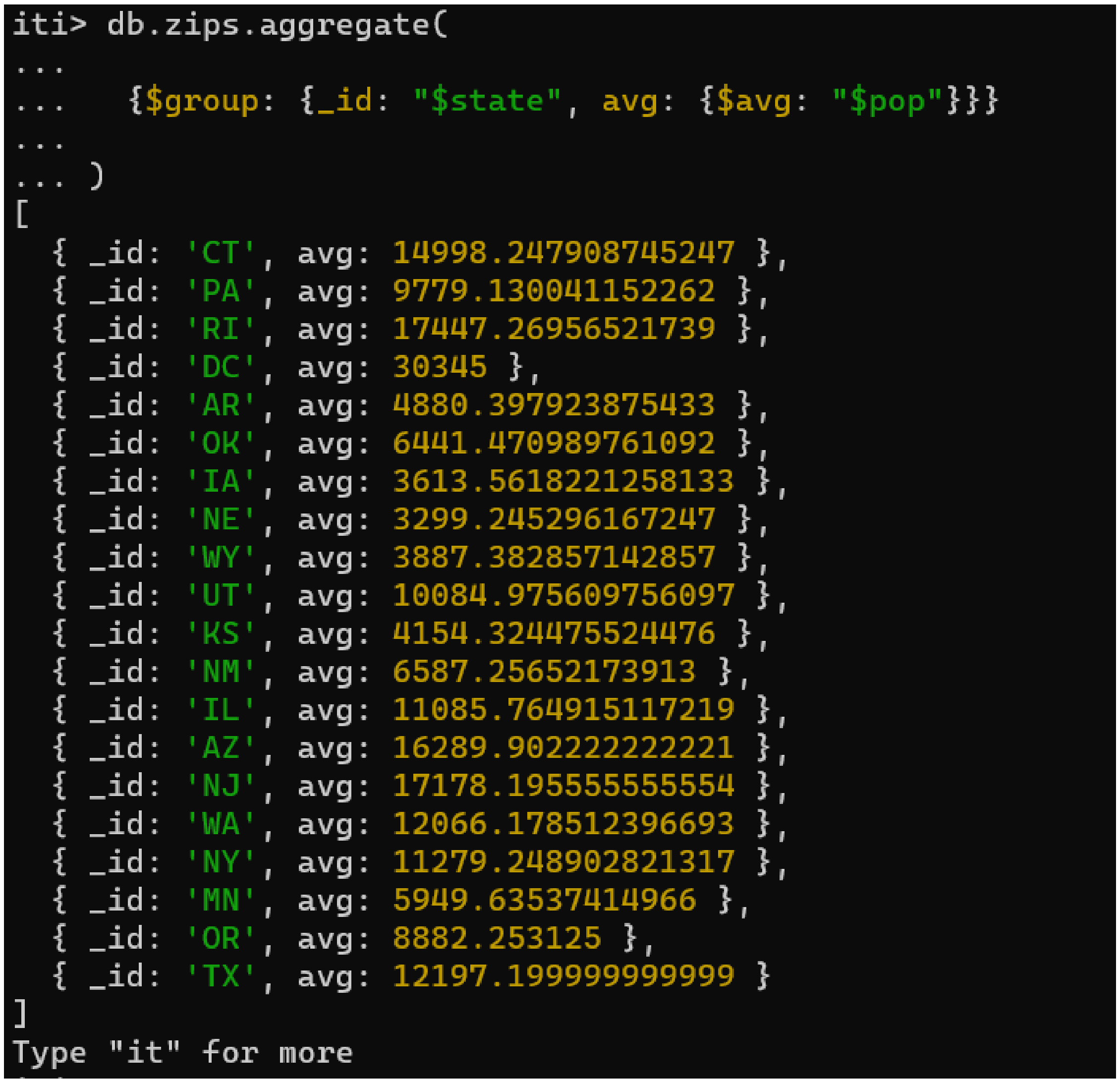
5. Get smallest population and greatest population of each state and save the result in collection

named "mypop" on your machine colleague



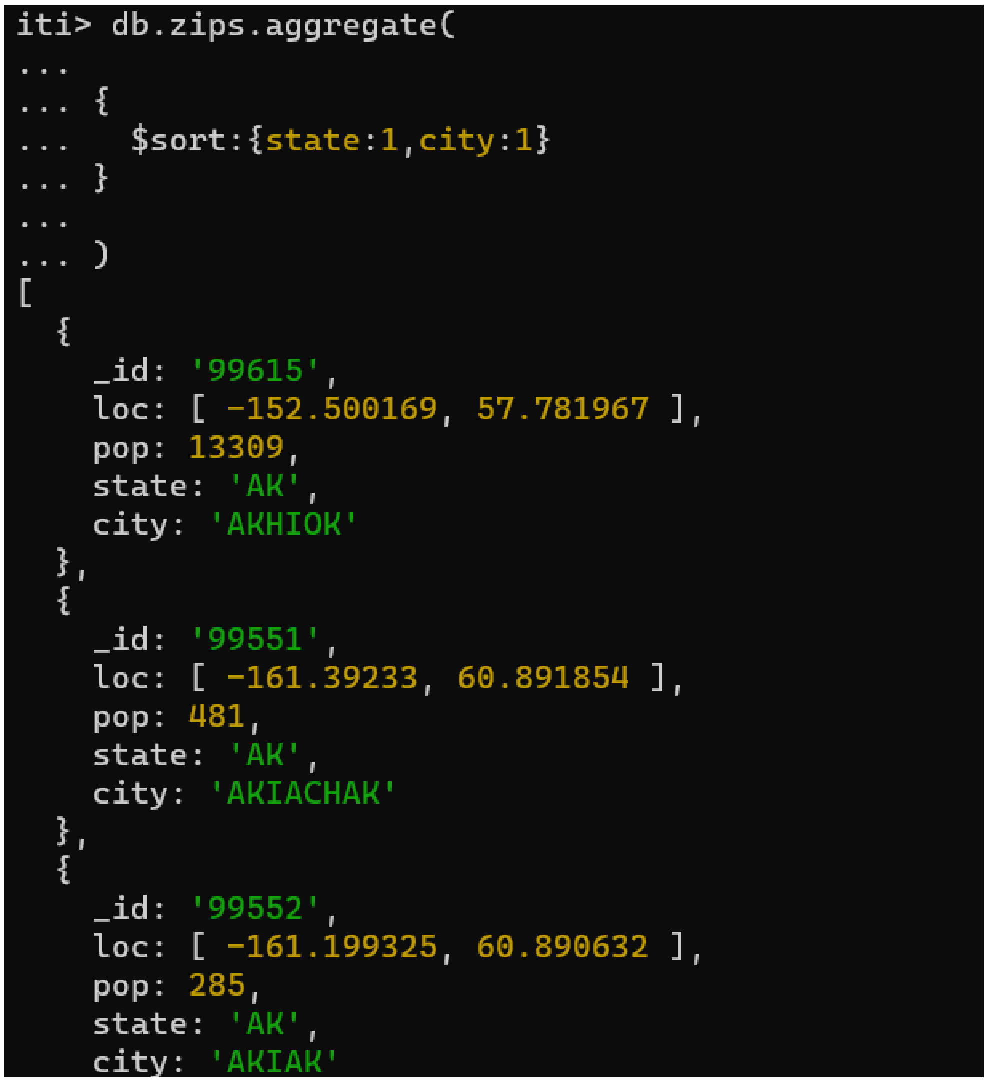
```
iti> db.zips.aggregate(
...
...     {
...         $group: { _id: "$state", max: { $max: "$pop" },min:{$min: "$pop"} }
...      },
...      { $out: { db: "iti", coll: "mypop" } }
...
... )

iti> show collections
mypop
students
zips
iti> db.mypop.find()
[
  { _id: 'SC', max: 80388, min: 0 },
  { _id: 'MO', max: 65992.8, min: 0 },
  { _id: 'WI', max: 68624.4, min: 2.4 },
  { _id: 'AR', max: 64238.399999999994, min: 0 },
  { _id: 'CO', max: 71301.59999999999, min: 0 },
  { _id: 'GA', max: 70375.2, min: 0 },
  { _id: 'WY', max: 39728.4, min: 7.199999999999999 },
  { _id: 'MT', max: 48145.2, min: 8.4 },
  { _id: 'NM', max: 69002.4, min: 0 },
  { _id: 'CA', max: 119481.59999999999, min: 0 },
  { _id: 'MD', max: 91202.4, min: 1.2 },
  { _id: 'MA', max: 78055.2, min: 0 },
  { _id: 'IN', max: 67851.59999999999, min: 90 },
  { _id: 'FL', max: 87832.8, min: 0 },
  { _id: 'KY', max: 55875.6, min: 0 },
  { _id: 'OK', max: 54650.4, min: 9.6 },
  { _id: 'TN', max: 72609.59999999999, min: 2.4 },
  { _id: 'OH', max: 80008.8, min: 45.6 },
  { _id: 'TX', max: 95355.59999999999, min: 0 },
  { _id: 'OR', max: 57608.4, min: 0 }
]
Type "it" for more
iti>
```

6. Write an aggregation expression to calculate the average population of a zip code (postal code) by state



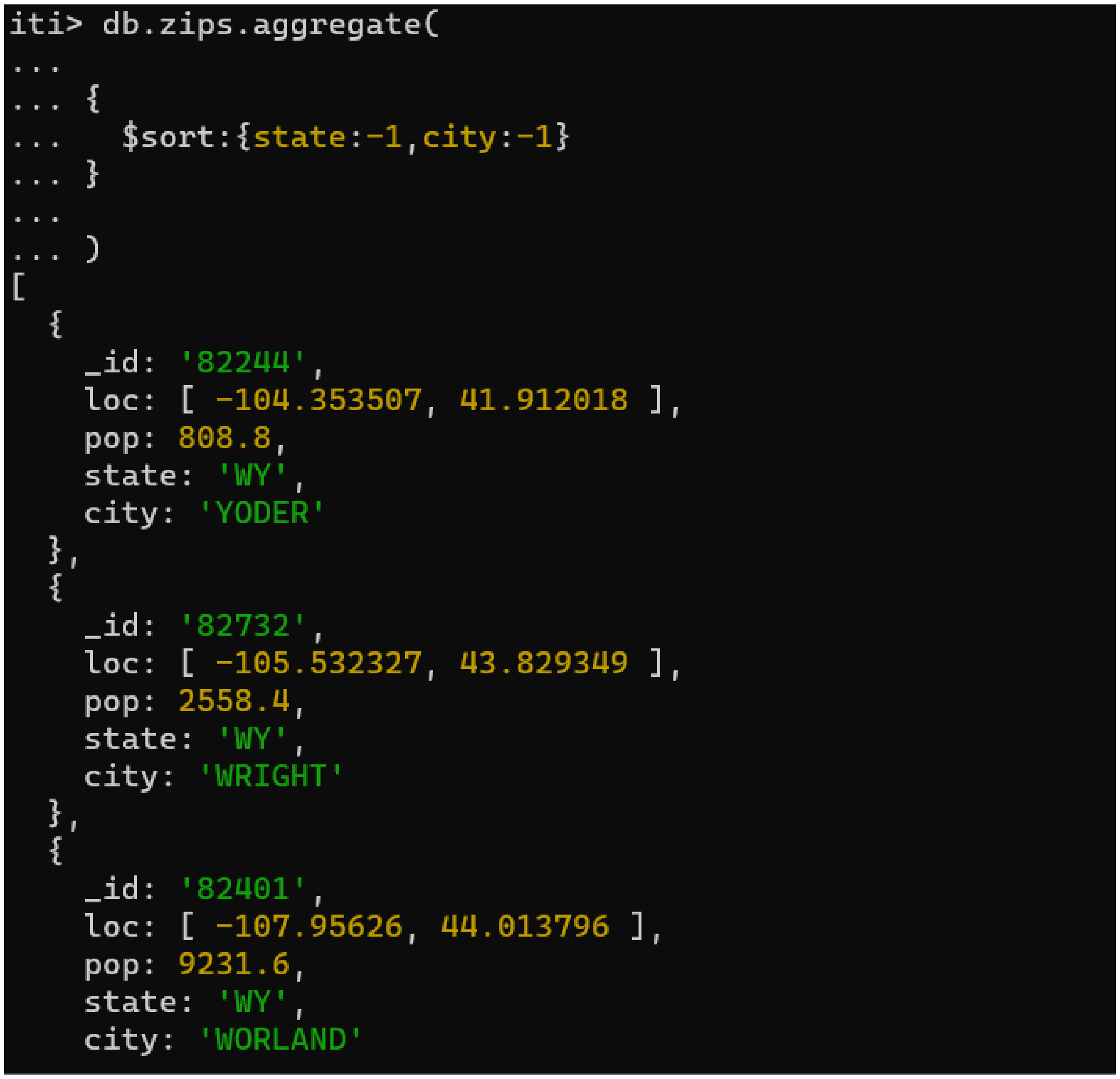
```
iti> db.zips.aggregate(
...
...    {$group: {_id: "$state", avg: {$avg: "$pop"}}}
...
... )
[
  { _id: 'CT', avg: 14998.247908745247 },
  { _id: 'PA', avg: 9779.130041152262 },
  { _id: 'RI', avg: 17447.26956521739 },
  { _id: 'DC', avg: 30345 },
  { _id: 'AR', avg: 4880.397923875433 },
  { _id: 'OK', avg: 6441.470989761092 },
  { _id: 'IA', avg: 3613.5618221258133 },
  { _id: 'NE', avg: 3299.245296167247 },
  { _id: 'WY', avg: 3887.382857142857 },
  { _id: 'UT', avg: 10084.975609756097 },
  { _id: 'KS', avg: 4154.324475524476 },
  { _id: 'NM', avg: 6587.25652173913 },
  { _id: 'IL', avg: 11085.764915117219 },
  { _id: 'AZ', avg: 16289.902222222221 },
  { _id: 'NJ', avg: 17178.195555555554 },
  { _id: 'WA', avg: 12066.178512396693 },
  { _id: 'NY', avg: 11279.248902821317 },
  { _id: 'MN', avg: 5949.63537414966 },
  { _id: 'OR', avg: 8882.253125 },
  { _id: 'TX', avg: 12197.199999999999 }
]
Type "it" for more
```

7. Write an aggregation query with just a sort stage to sort by (state, city), both ascending

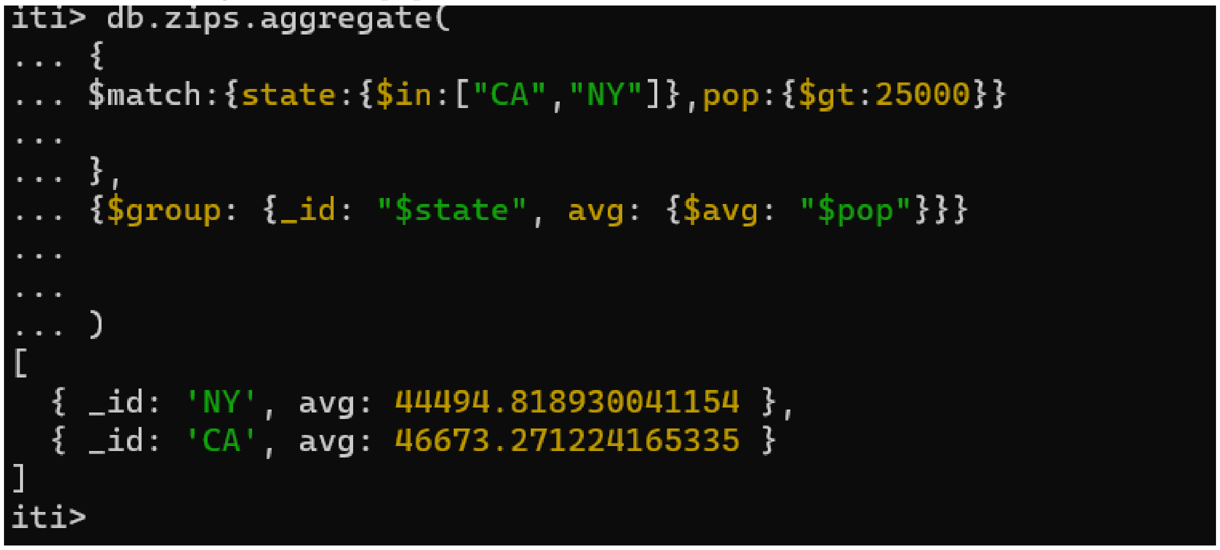


```
iti> db.zips.aggregate(
...
... {
...   $sort:{state:1,city:1}
... }
...
... )
[
  {
    _id: '99615',
    loc: [ -152.500169, 57.781967 ],
    pop: 13309,
    state: 'AK',
    city: 'AKHIOK'
  },
  {
    _id: '99551',
    loc: [ -161.39233, 60.891854 ],
    pop: 481,
    state: 'AK',
    city: 'AKIACHAK'
  },
  {
    _id: '99552',
    loc: [ -161.199325, 60.890632 ],
    pop: 285,
    state: 'AK',
    city: 'AKIAK'
```

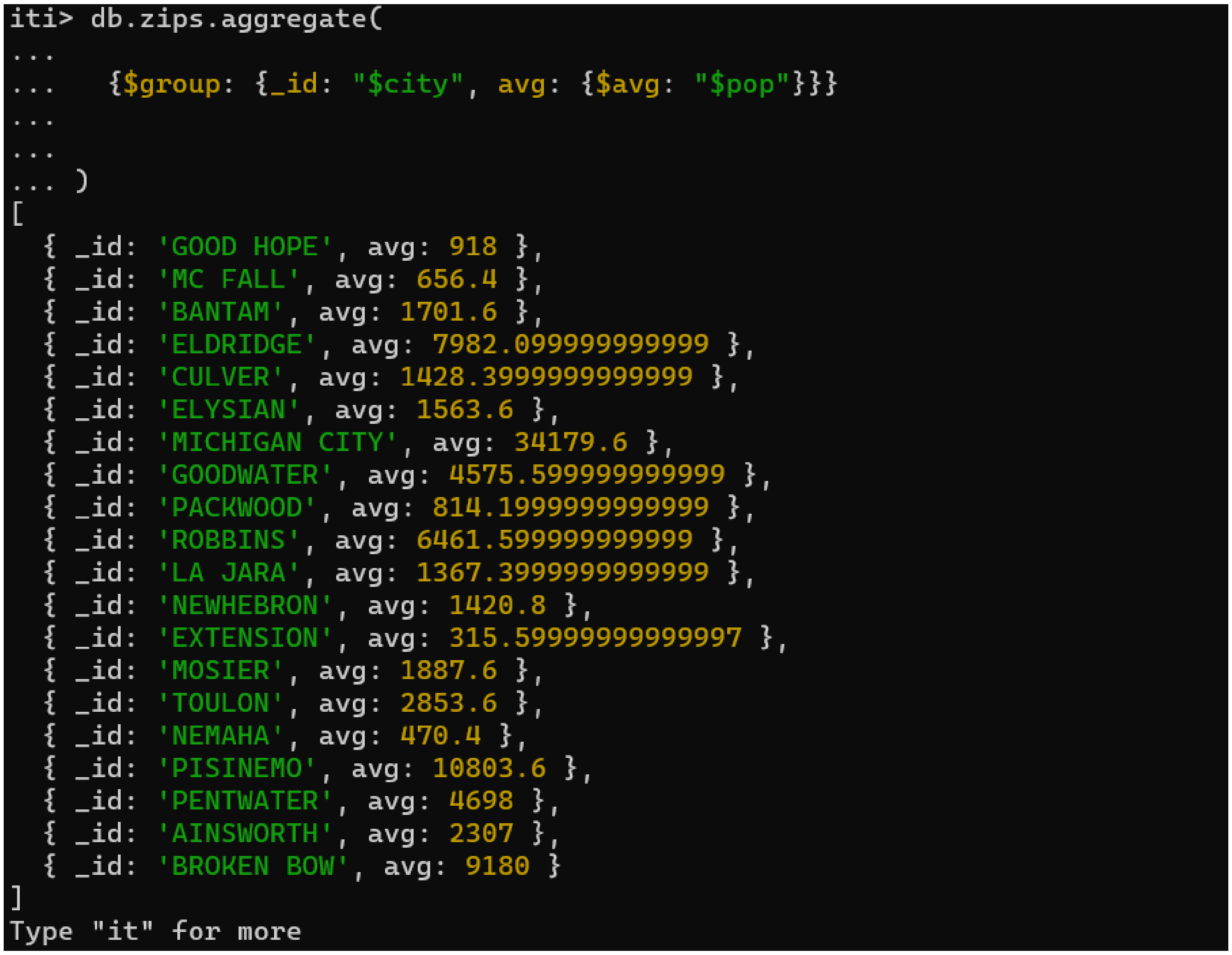
8. Write an aggregation query with just a sort stage to sort by (state, city), both descending


```
iti> db.zips.aggregate(
...
... {
...   $sort:{state:-1,city:-1}
... }
...
... )
[
  {
    _id: '82244',
    loc: [ -104.353507, 41.912018 ],
    pop: 808.8,
    state: 'WY',
    city: 'YODER'
  },
  {
    _id: '82732',
    loc: [ -105.532327, 43.829349 ],
    pop: 2558.4,
    state: 'WY',
    city: 'WRIGHT'
  },
  {
    _id: '82401',
    loc: [ -107.95626, 44.013796 ],
    pop: 9231.6,
    state: 'WY',
    city: 'WORLAND'
```

9. Calculate the average population of cities in California (abbreviation CA) and New York (NY) (taken together) with populations over 25,000


```
iti> db.zips.aggregate(
... {
... $match:{state:{$in:["CA","NY"]},pop:{$gt:25000}}
...
... },
... {$group: {_id: "$state", avg: {$avg: "$pop"}}}
...
...
... )
[
  { _id: 'NY', avg: 44494.818930041154 },
  { _id: 'CA', avg: 46673.271224165335 }
]
iti>
```

10. Return the average populations for cities in each state



```
iti> db.zips.aggregate(
...
...    {$group: {_id: "$city", avg: {$avg: "$pop"}}}
...
...
... )
[
  { _id: 'GOOD HOPE', avg: 918 },
  { _id: 'MC FALL', avg: 656.4 },
  { _id: 'BANTAM', avg: 1701.6 },
  { _id: 'ELDRIDGE', avg: 7982.099999999999 },
  { _id: 'CULVER', avg: 1428.3999999999999 },
  { _id: 'ELYSIAN', avg: 1563.6 },
  { _id: 'MICHIGAN CITY', avg: 34179.6 },
  { _id: 'GOODWATER', avg: 4575.599999999999 },
  { _id: 'PACKWOOD', avg: 814.1999999999999 },
  { _id: 'ROBBINS', avg: 6461.599999999999 },
  { _id: 'LA JARA', avg: 1367.3999999999999 },
  { _id: 'NEWHEBRON', avg: 1420.8 },
  { _id: 'EXTENSION', avg: 315.59999999999997 },
  { _id: 'MOSIER', avg: 1887.6 },
  { _id: 'TOULON', avg: 2853.6 },
  { _id: 'NEMAHA', avg: 470.4 },
  { _id: 'PISINEMO', avg: 10803.6 },
  { _id: 'PENTWATER', avg: 4698 },
  { _id: 'AINSWORTH', avg: 2307 },
  { _id: 'BROKEN BOW', avg: 9180 }
]
Type "it" for more
```