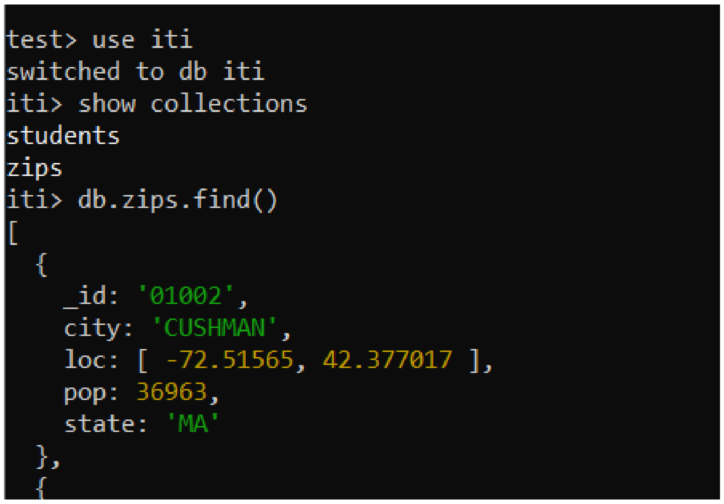
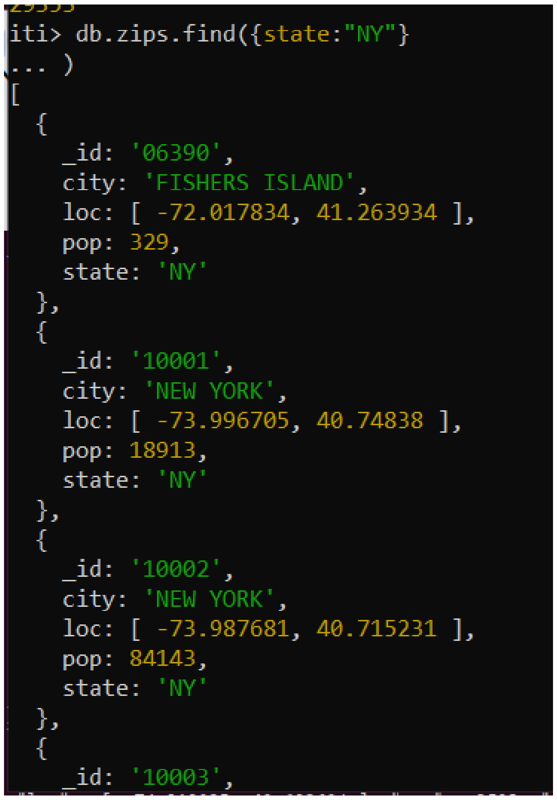
MongoDB Lab2

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



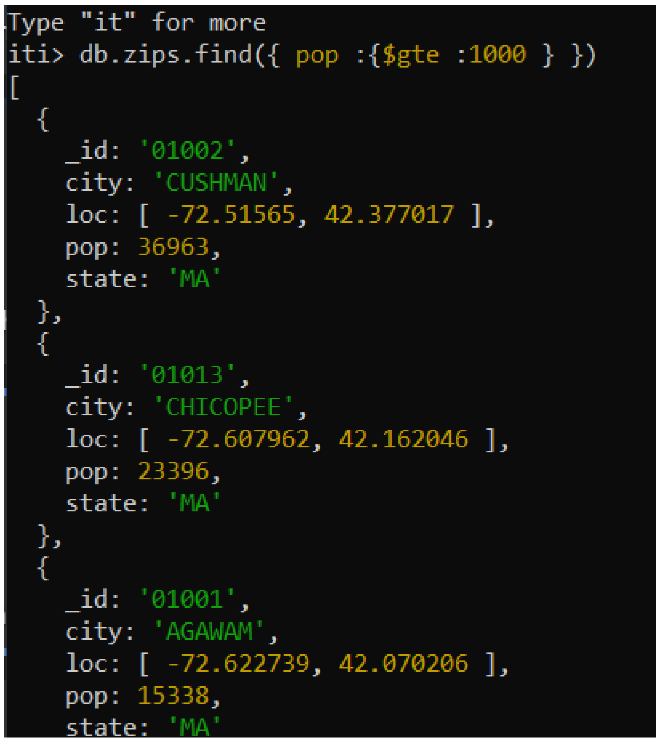
```
test> use iti
switched to db iti
iti> show collections
students
zips
iti> db.zips.find()
[
  {
    _id: '01002',
    city: 'CUSHMAN',
    loc: [ -72.51565, 42.377017 ],
    pop: 36963,
    state: 'MA'
  },
  {
```

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



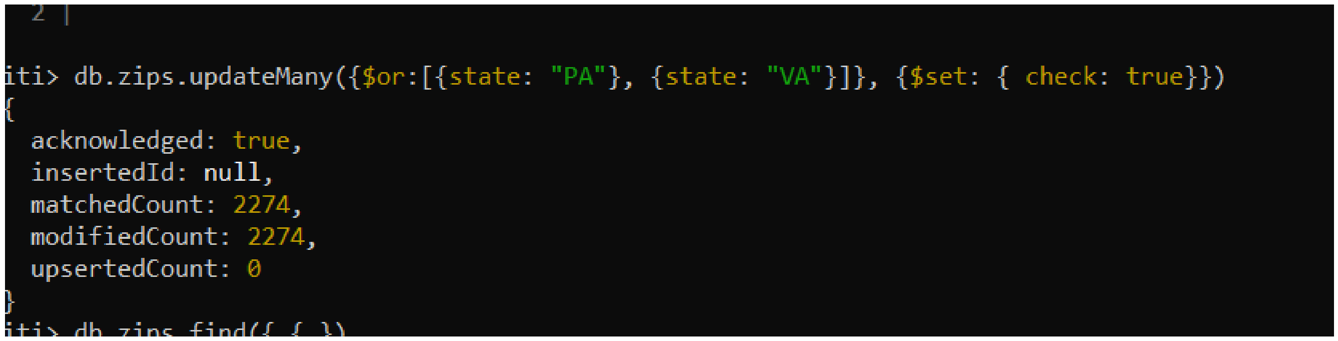
```
iti> db.zips.find({state:"NY"}
... )
[
  {
    _id: '06390',
    city: 'FISHERS ISLAND',
    loc: [ -72.017834, 41.263934 ],
    pop: 329,
    state: 'NY'
  },
  {
    _id: '10001',
    city: 'NEW YORK',
    loc: [ -73.996705, 40.74838 ],
    pop: 18913,
    state: 'NY'
  },
  {
    _id: '10002',
    city: 'NEW YORK',
    loc: [ -73.987681, 40.715231 ],
    pop: 84143,
    state: 'NY'
  },
  {
    _id: '10003',
```

3 – find <u>all zip codes</u> whose population is <u>greater than or equal to 1000</u>



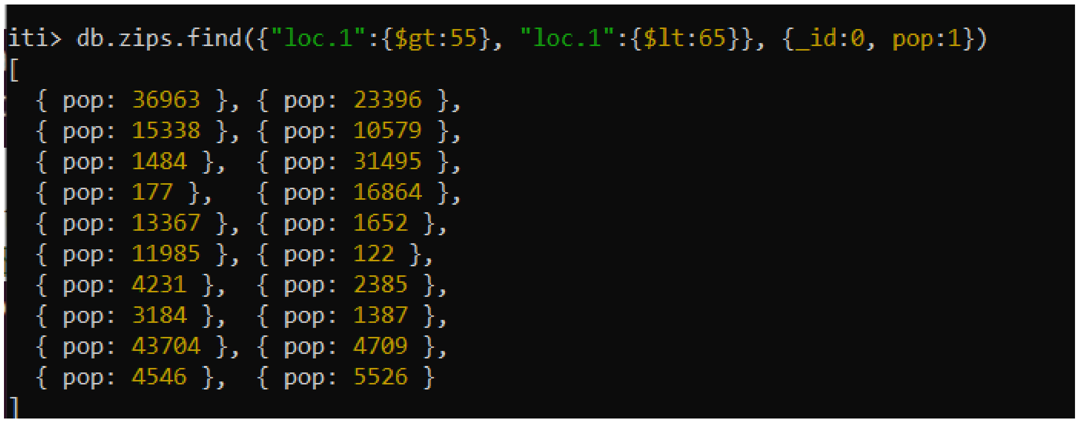
```
Type "it" for more
iti> db.zips.find({ pop :{$gte :1000 } })
[
  {
    _id: '01002',
    city: 'CUSHMAN',
    loc: [ -72.51565, 42.377017 ],
    pop: 36963,
    state: 'MA'
  },
  {
    _id: '01013',
    city: 'CHICOPEE',
    loc: [ -72.607962, 42.162046 ],
    pop: 23396,
    state: 'MA'
  },
  {
    _id: '01001',
    city: 'AGAWAM',
    loc: [ -72.622739, 42.070206 ],
    pop: 15338,
    state: 'MA'
```

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



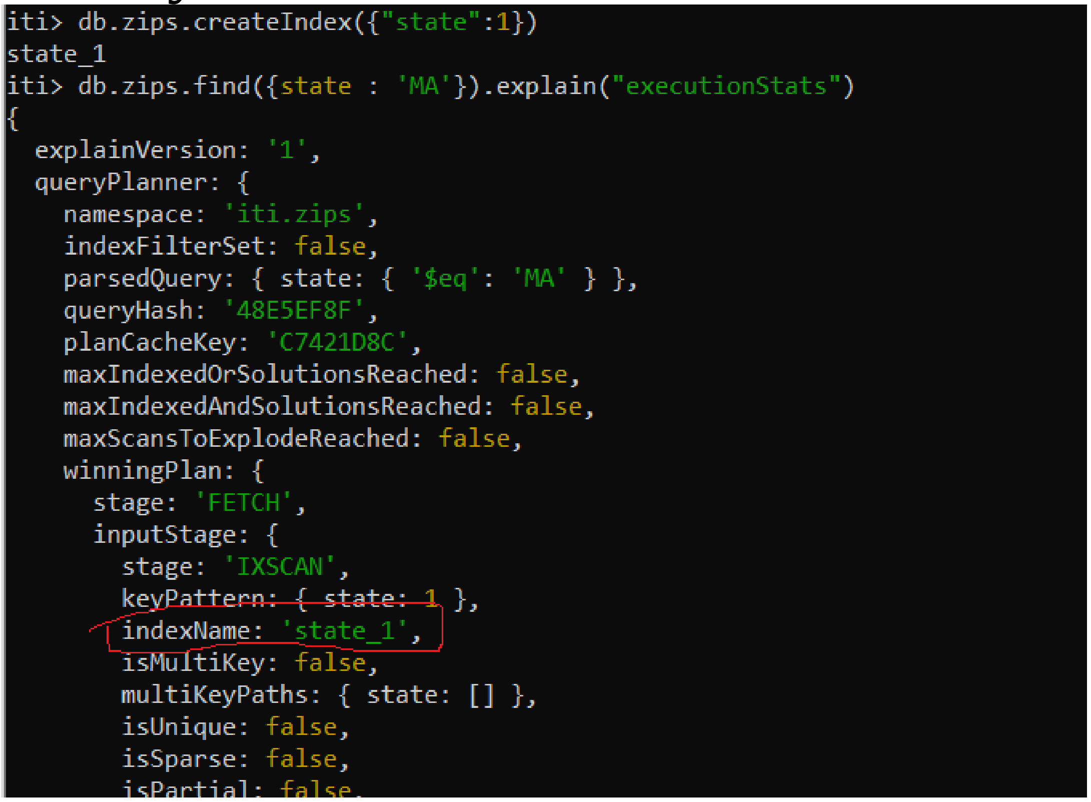
```
iti> db.zips.updateMany({$or:[{state: "PA"}, {state: "VA"}]}, {$set: { check: true}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 2274,
  modifiedCount: 2274,
  upsertedCount: 0
}
iti> db.zips.find({ { })
```

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



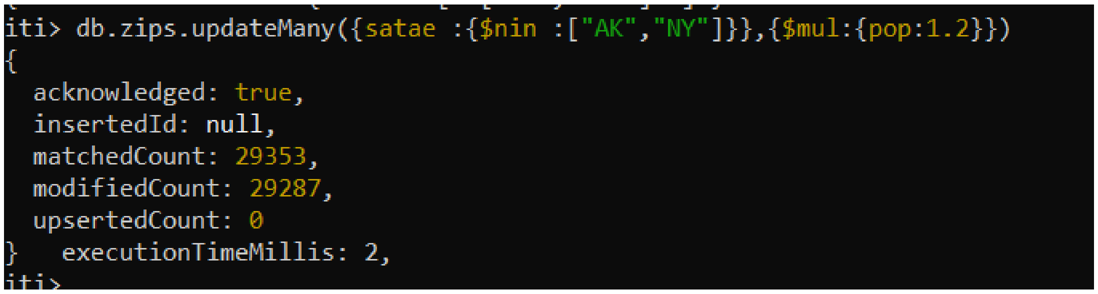
```
iti> db.zips.find({"loc.1":{$gt:55}, "loc.1":{$lt:65}}, {_id:0, pop:1})
[
  { pop: 36963 }, { pop: 23396 },
  { pop: 15338 }, { pop: 10579 },
  { pop: 1484 },  { pop: 31495 },
  { pop: 177 },   { pop: 16864 },
  { pop: 13367 }, { pop: 1652 },
  { pop: 11985 }, { pop: 122 },
  { pop: 4231 },  { pop: 2385 },
  { pop: 3184 },  { pop: 1387 },
  { pop: 43704 }, { pop: 4709 },
  { pop: 4546 },  { pop: 5526 }
]
```

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



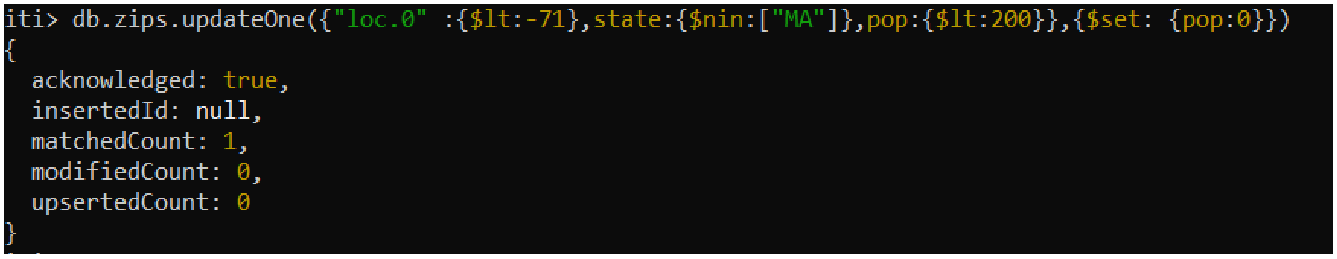
```
iti> db.zips.createIndex({"state":1})
state_1
iti> db.zips.find({state : 'MA'}).explain("executionStats")
{
  explainVersion: '1',
  queryPlanner: {
    namespace: 'iti.zips',
    indexFilterSet: false,
    parsedQuery: { state: { '$eq': 'MA' } },
    queryHash: '48E5EF8F',
    planCacheKey: 'C7421D8C',
    maxIndexedOrSolutionsReached: false,
    maxIndexedAndSolutionsReached: false,
    maxScansToExplodeReached: false,
    winningPlan: {
      stage: 'FETCH',
      inputStage: {
        stage: 'IXSCAN',
        keyPattern: { state: 1 },
        indexName: 'state_1',
        isMultiKey: false,
        multiKeyPaths: { state: [] },
        isUnique: false,
        isSparse: false,
        isPartial: false,
```

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



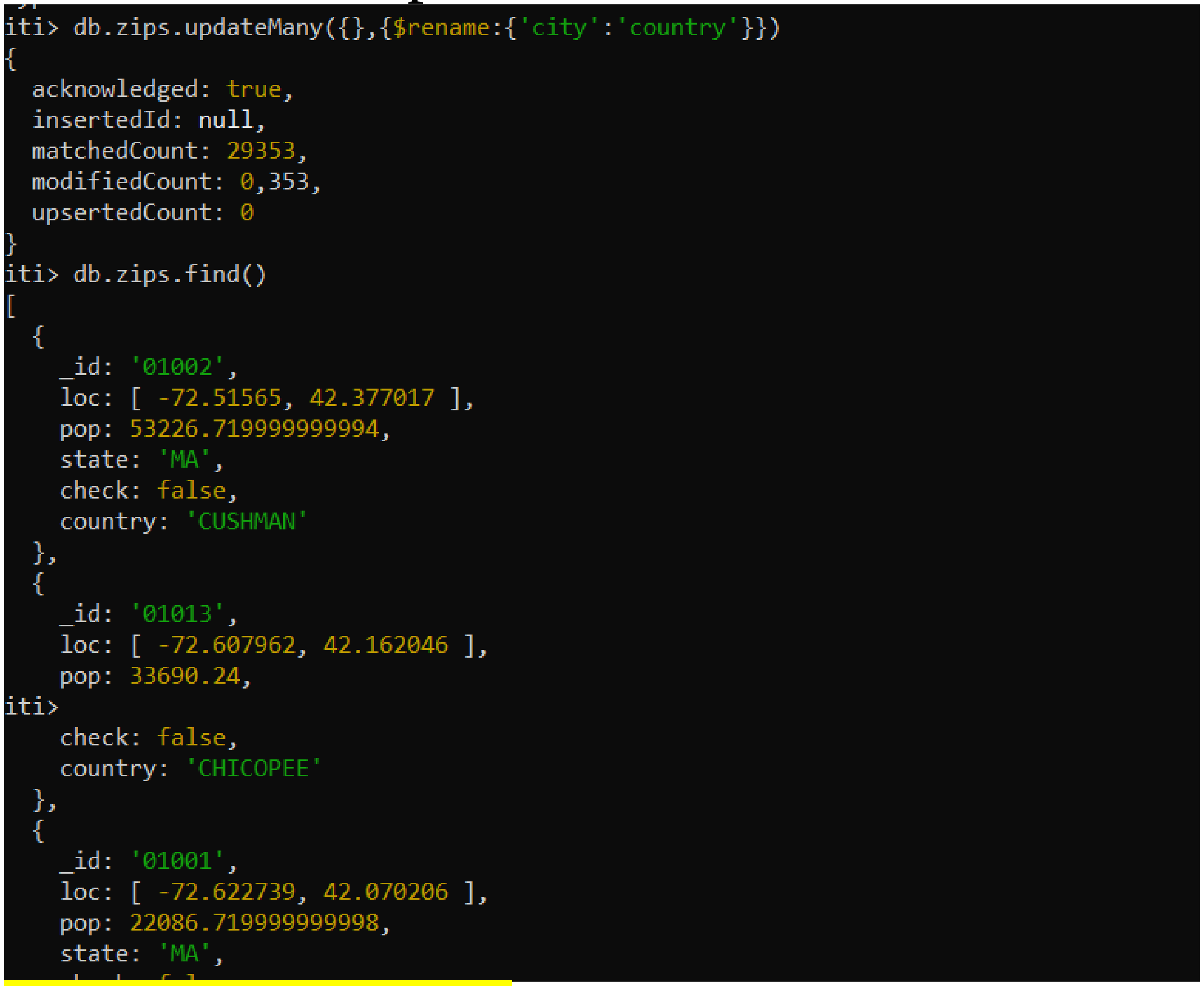
```
iti> db.zips.updateMany({satae :{$nin :["AK","NY"]}},{$mul:{pop:1.2}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 29353,
  modifiedCount: 29287,
  upsertedCount: 0
}   executionTimeMillis: 2,
iti>
```

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: 0,
  upsertedCount: 0
}
```

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.



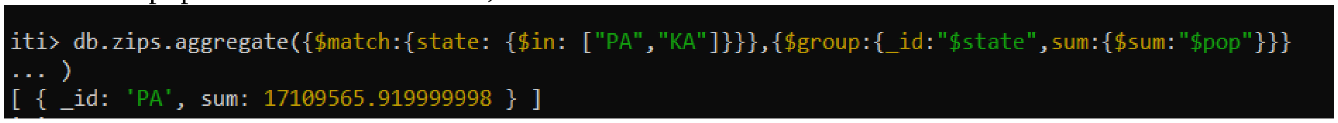
```
iti> db.zips.updateMany({},{$rename:{'city':'country'}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 29353,
  modifiedCount: 0,353,
  upsertedCount: 0
}
iti> db.zips.find()
[
  {
    _id: '01002',
    loc: [ -72.51565, 42.377017 ],
    pop: 53226.719999999994,
    state: 'MA',
    check: false,
    country: 'CUSHMAN'
  },
  {
    _id: '01013',
    loc: [ -72.607962, 42.162046 ],
    pop: 33690.24,
iti>
    check: false,
    country: 'CHICOPEE'
  },
  {
    _id: '01001',
    loc: [ -72.622739, 42.070206 ],
    pop: 22086.719999999998,
    state: 'MA',
```

Hint: use Variables

****

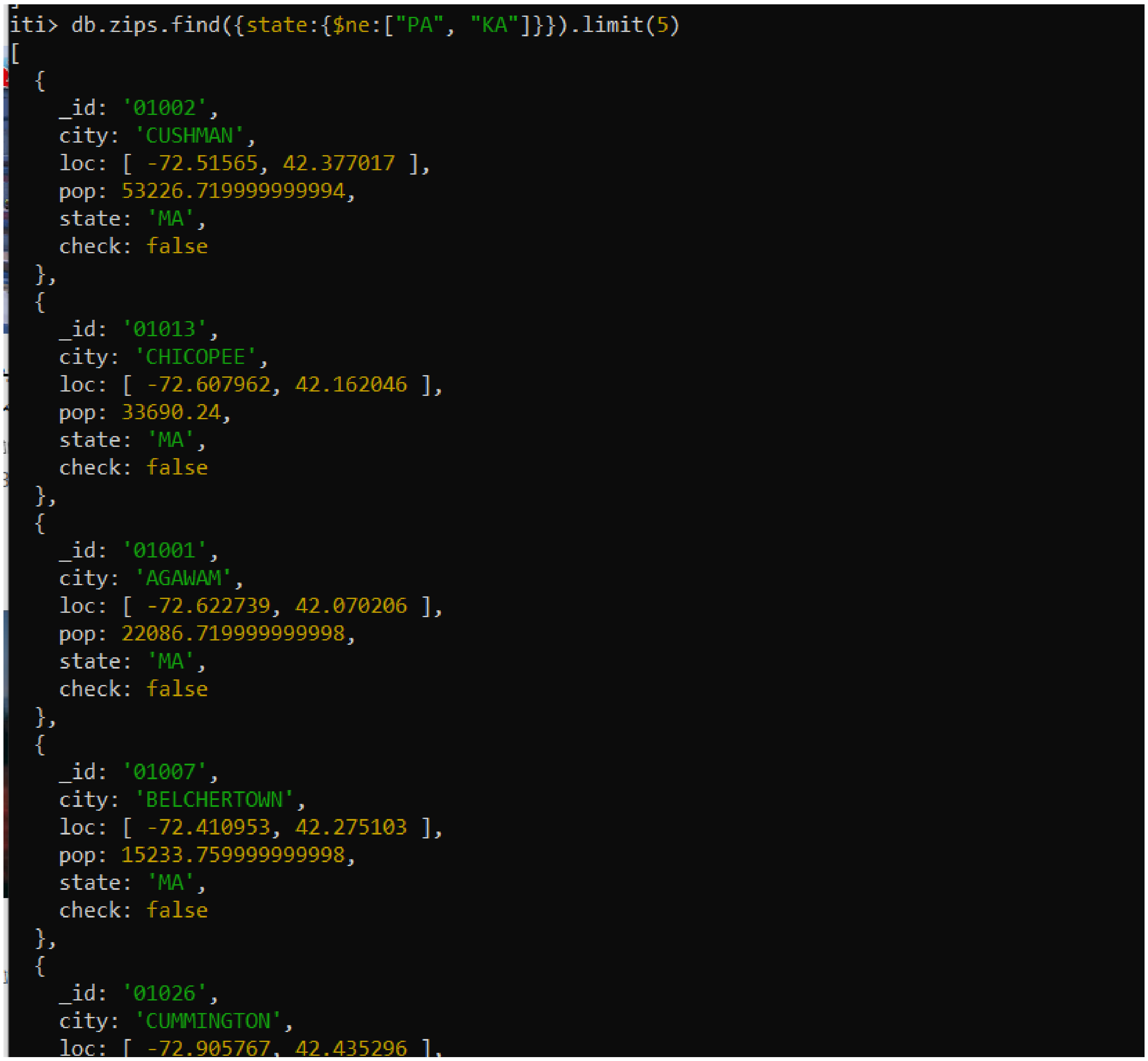
part2

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



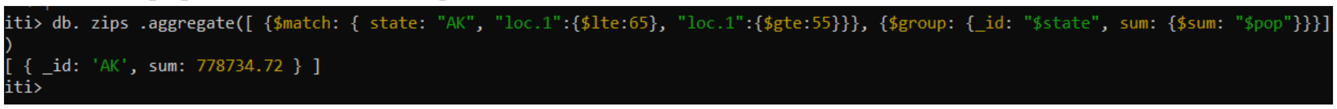
```
iti> db.zips.aggregate({$match:{state: {$in: ["PA","KA"]}}},{$group:{_id:"$state",sum:{$sum:"$pop"}}}
... )
[ { _id: 'PA', sum: 17109565.919999998 } ]
```

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



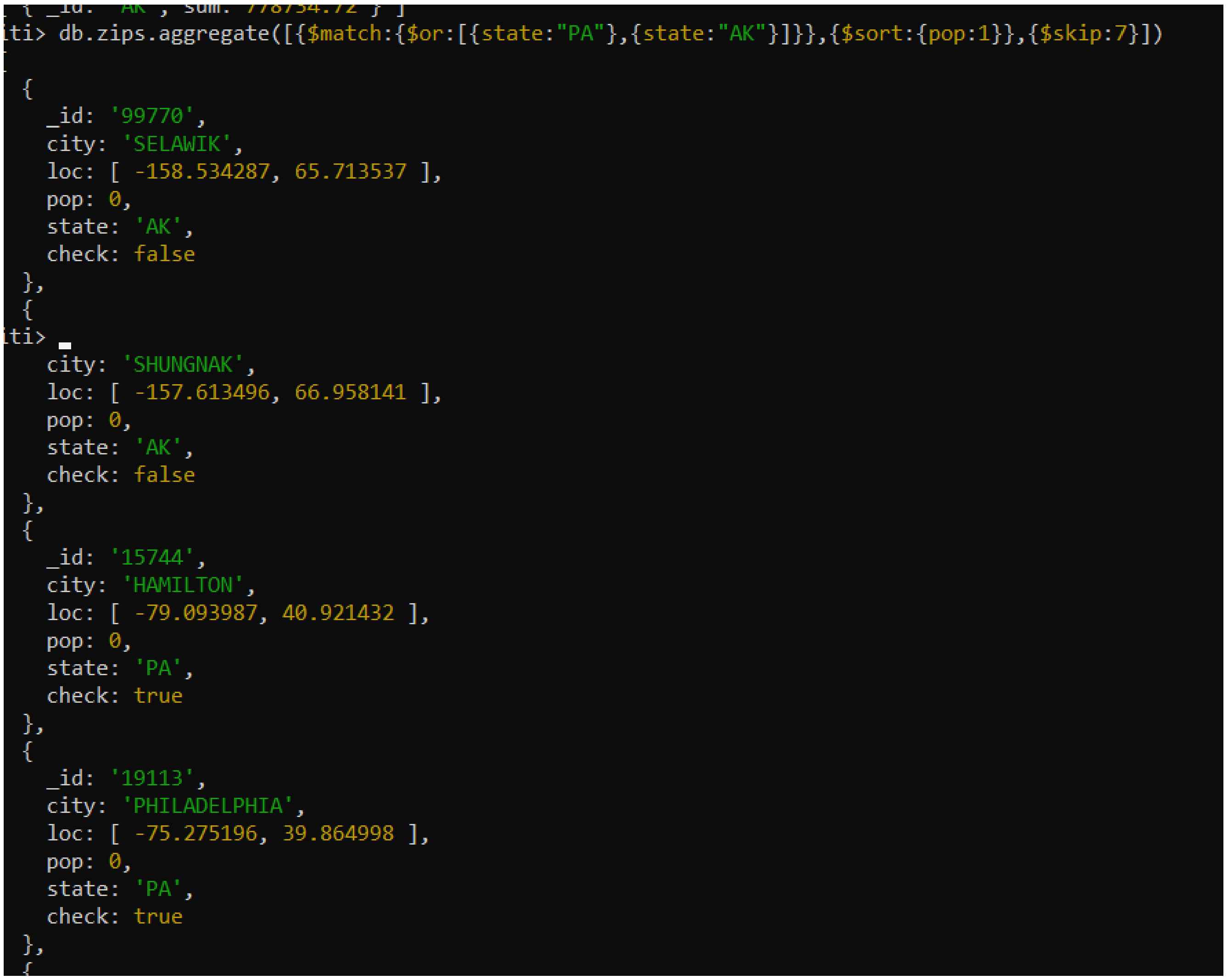
```
iti> db.zips.find({state:{$ne:["PA", "KA"]}}).limit(5)
[
  {
    _id: '01002',
    city: 'CUSHMAN',
    loc: [ -72.51565, 42.377017 ],
    pop: 53226.719999999994,
    state: 'MA',
    check: false
  },
  {
    _id: '01013',
    city: 'CHICOPEE',
    loc: [ -72.607962, 42.162046 ],
    pop: 33690.24,
    state: 'MA',
    check: false
  },
  {
    _id: '01001',
    city: 'AGAWAM',
    loc: [ -72.622739, 42.070206 ],
    pop: 22086.719999999998,
    state: 'MA',
    check: false
  },
  {
    _id: '01007',
    city: 'BELCHERTOWN',
    loc: [ -72.410953, 42.275103 ],
    pop: 15233.759999999998,
    state: 'MA',
    check: false
  },
  {
    _id: '01026',
    city: 'CUMMINGTON',
    loc: [ -72.905767, 42.435296 ],
```

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



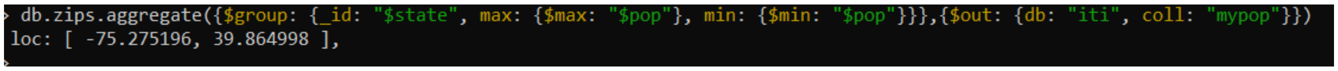
```
iti> db. zips .aggregate([ {$match: { state: "AK", "loc.1":{$lte:65}, "loc.1":{$gte:55}}}, {$group: {_id: "$state", sum: {$sum: "$pop"}}}]
)
[ { _id: 'AK', sum: 778734.72 } ]
iti>
```

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



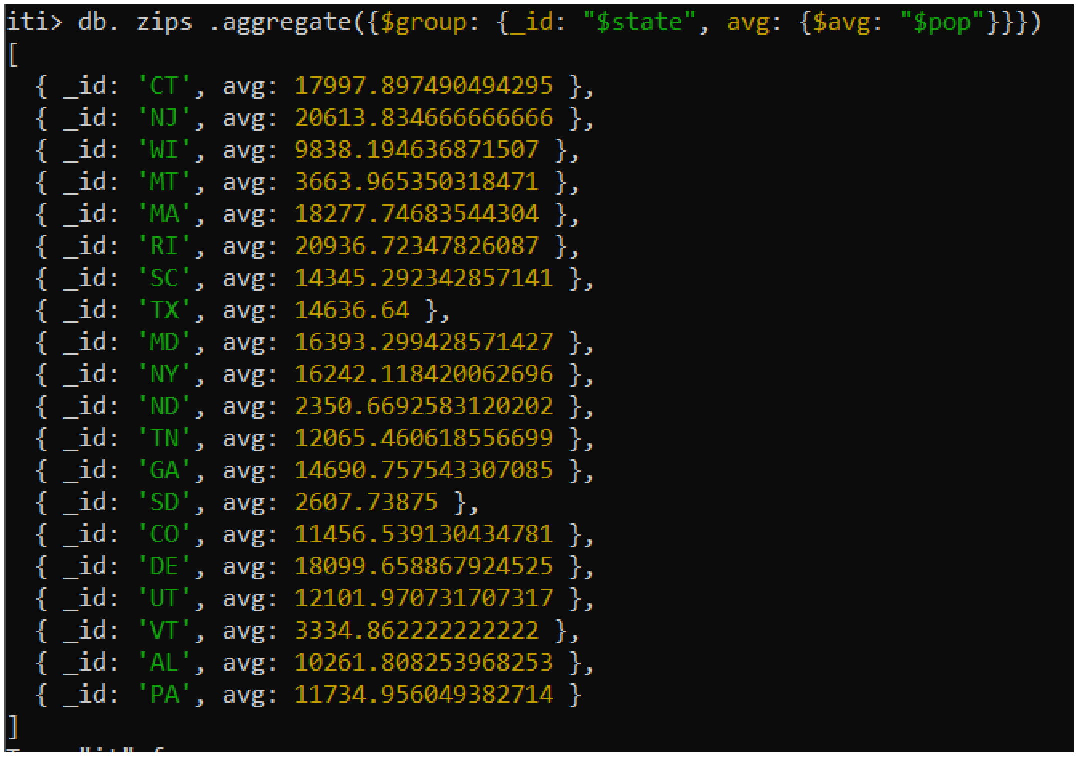
```
  { _id: 'AK', sum: 778734.72 } ]
iti> db.zips.aggregate([{$match:{$or:[{state:"PA"},{state:"AK"}]}},{$sort:{pop:1}},{$skip:7}])
[
  {
    _id: '99770',
    city: 'SELAWIK',
    loc: [ -158.534287, 65.713537 ],
    pop: 0,
    state: 'AK',
    check: false
  },
  {
iti> _
    city: 'SHUNGNAK',
    loc: [ -157.613496, 66.958141 ],
    pop: 0,
    state: 'AK',
    check: false
  },
  {
    _id: '15744',
    city: 'HAMILTON',
    loc: [ -79.093987, 40.921432 ],
    pop: 0,
    state: 'PA',
    check: true
  },
  {
    _id: '19113',
    city: 'PHILADELPHIA',
    loc: [ -75.275196, 39.864998 ],
    pop: 0,
    state: 'PA',
    check: true
  },
  {
```

5. Get smallest population and greatest population of each state and save the result in collection named "mypop" on your machine colleague



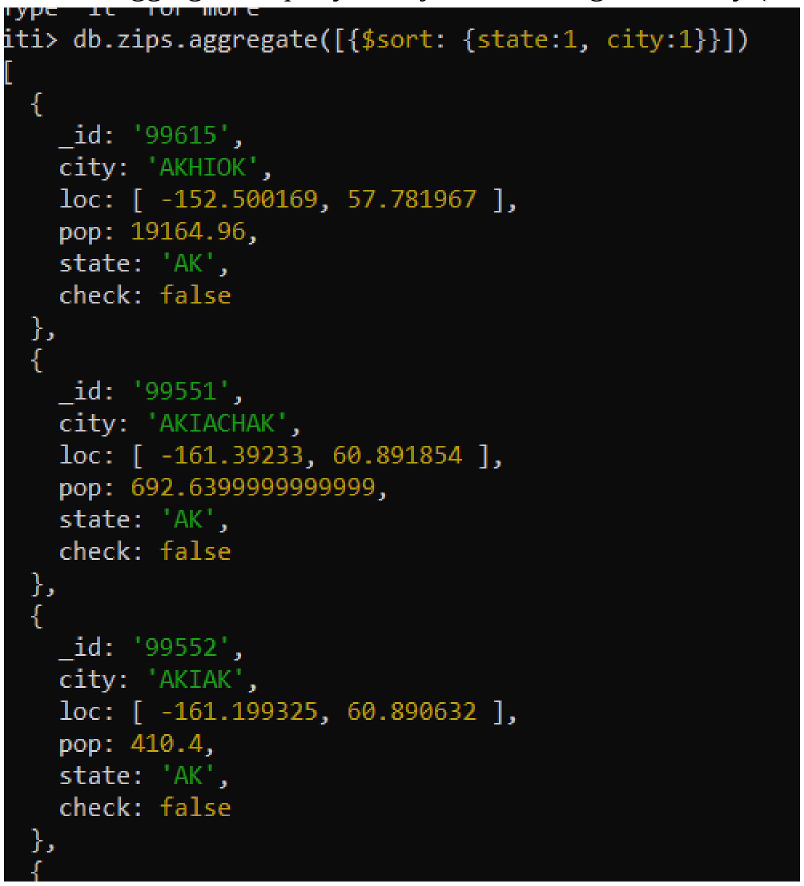
```
> db.zips.aggregate({$group: {_id: "$state", max: {$max: "$pop"}, min: {$min: "$pop"}}},{$out: {db: "iti", coll: "mypop"}})
 loc: [ -75.275196, 39.864998 ],
>
```

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



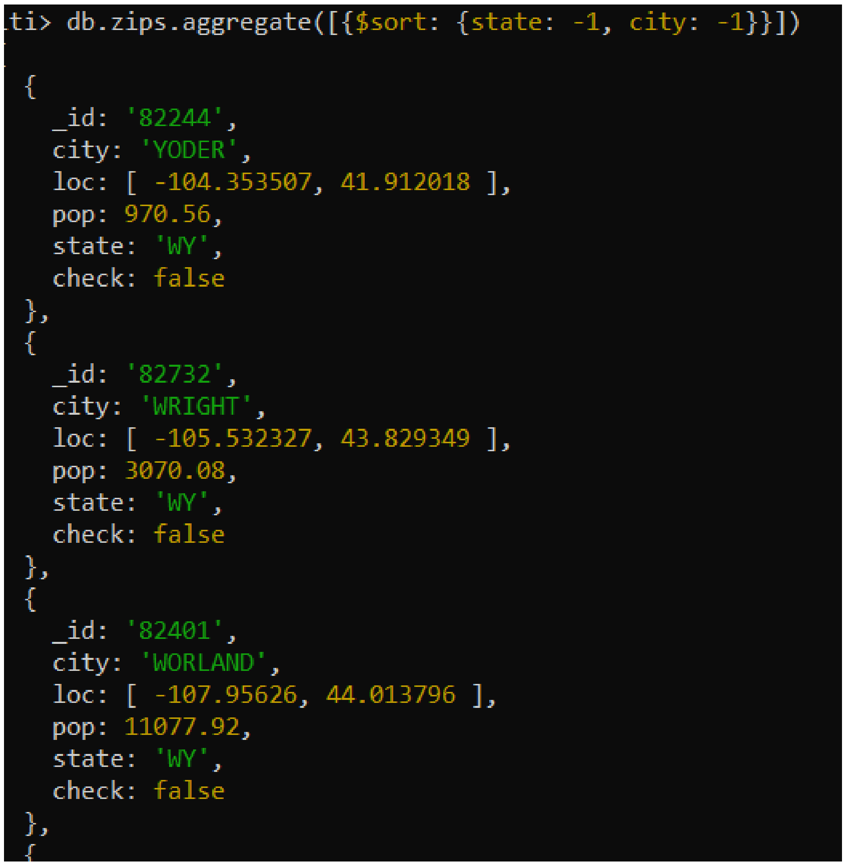
```
iti> db. zips .aggregate({$group: {_id: "$state", avg: {$avg: "$pop"}}})
[
  { _id: 'CT', avg: 17997.897490494295 },
  { _id: 'NJ', avg: 20613.834666666666 },
  { _id: 'WI', avg: 9838.194636871507 },
  { _id: 'MT', avg: 3663.965350318471 },
  { _id: 'MA', avg: 18277.74683544304 },
  { _id: 'RI', avg: 20936.72347826087 },
  { _id: 'SC', avg: 14345.292342857141 },
  { _id: 'TX', avg: 14636.64 },
  { _id: 'MD', avg: 16393.299428571427 },
  { _id: 'NY', avg: 16242.118420062696 },
  { _id: 'ND', avg: 2350.6692583120202 },
  { _id: 'TN', avg: 12065.460618556699 },
  { _id: 'GA', avg: 14690.757543307085 },
  { _id: 'SD', avg: 2607.73875 },
  { _id: 'CO', avg: 11456.539130434781 },
  { _id: 'DE', avg: 18099.658867924525 },
  { _id: 'UT', avg: 12101.970731707317 },
  { _id: 'VT', avg: 3334.862222222222 },
  { _id: 'AL', avg: 10261.808253968253 },
  { _id: 'PA', avg: 11734.956049382714 }
]
```

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



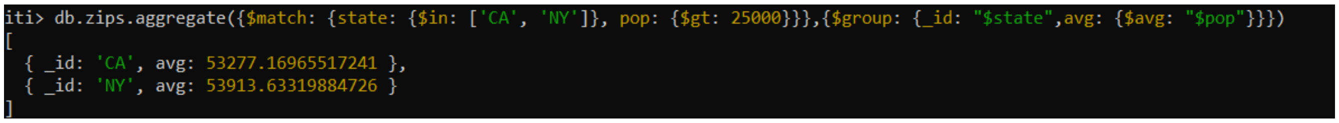
```
iti> db.zips.aggregate([{$sort: {state:1, city:1}}])
[
  {
    _id: '99615',
    city: 'AKHIOK',
    loc: [ -152.500169, 57.781967 ],
    pop: 19164.96,
    state: 'AK',
    check: false
  },
  {
    _id: '99551',
    city: 'AKIACHAK',
    loc: [ -161.39233, 60.891854 ],
    pop: 692.6399999999999,
    state: 'AK',
    check: false
  },
  {
    _id: '99552',
    city: 'AKIAK',
    loc: [ -161.199325, 60.890632 ],
    pop: 410.4,
    state: 'AK',
    check: false
  },
  {
```

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



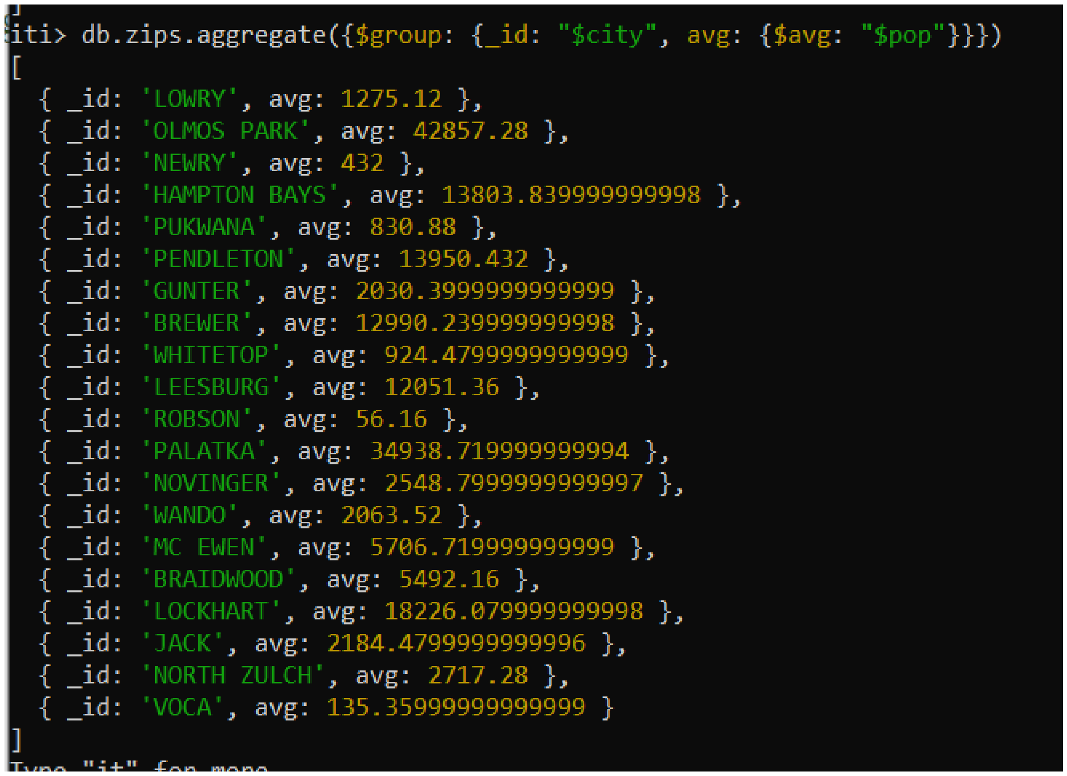
```
ti> db.zips.aggregate([{$sort: {state: -1, city: -1}}])

 {
   _id: '82244',
   city: 'YODER',
   loc: [ -104.353507, 41.912018 ],
   pop: 970.56,
   state: 'WY',
   check: false
 },
 {
   _id: '82732',
   city: 'WRIGHT',
   loc: [ -105.532327, 43.829349 ],
   pop: 3070.08,
   state: 'WY',
   check: false
 },
 {
   _id: '82401',
   city: 'WORLAND',
   loc: [ -107.95626, 44.013796 ],
   pop: 11077.92,
   state: 'WY',
   check: false
 },
```

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: 'CA', avg: 53277.16965517241 },
  { _id: 'NY', avg: 53913.63319884726 }
]
```

10. Return the average populations for cities in each state



```
iti> db.zips.aggregate({$group: {_id: "$city", avg: {$avg: "$pop"}}})
[
  { _id: 'LOWRY', avg: 1275.12 },
  { _id: 'OLMOS PARK', avg: 42857.28 },
  { _id: 'NEWRY', avg: 432 },
  { _id: 'HAMPTON BAYS', avg: 13803.839999999998 },
  { _id: 'PUKWANA', avg: 830.88 },
  { _id: 'PENDLETON', avg: 13950.432 },
  { _id: 'GUNTER', avg: 2030.3999999999999 },
  { _id: 'BREWER', avg: 12990.239999999998 },
  { _id: 'WHITETOP', avg: 924.4799999999999 },
  { _id: 'LEESBURG', avg: 12051.36 },
  { _id: 'ROBSON', avg: 56.16 },
  { _id: 'PALATKA', avg: 34938.719999999994 },
  { _id: 'NOVINGER', avg: 2548.7999999999997 },
  { _id: 'WANDO', avg: 2063.52 },
  { _id: 'MC EWEN', avg: 5706.719999999999 },
  { _id: 'BRAIDWOOD', avg: 5492.16 },
  { _id: 'LOCKHART', avg: 18226.079999999998 },
  { _id: 'JACK', avg: 2184.4799999999996 },
  { _id: 'NORTH ZULCH', avg: 2717.28 },
  { _id: 'VOCA', avg: 135.35999999999999 }
]
Type "it" for more
```