

MongoDB Lab2

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

- `mongoimport --db iti --collection zips --file C:\Users\CARNIVAL\Downloads\day2_Mongo\day2\zips.json`

```
C:\Users\CARNIVAL>mongoimport --db iti --collection zips --file C:\Users\CARNIVAL\Downloads\day2_Mongo\day2\zips.json
2023-02-23T20:26:34.263+0200    connected to: mongodb://localhost/
2023-02-23T20:26:35.230+0200    29353 document(s) imported successfully. 0 document(s) failed to import.
```

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

- `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'
  },
]
```

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

- `db.zips.find({pop:{$gte:1000}})`

```
{
  _id: '01002',
  city: 'CUSHMAN',
  loc: [ -72.51565, 42.377017 ],
  pop: 36963,
  state: 'MA'
},
{
  _id: '01008',
  city: 'BLANDFORD',
  loc: [ -72.936114, 42.182949 ],
  pop: 1240,
  state: 'MA'
},
{
  _id: '01011',
  city: 'CHESTER',
  loc: [ -72.988761, 42.279421 ],
  pop: 1688,
  state: 'MA'
},
]
```

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

- `db.zips.updateMany({},{$set:{check:false}})`
- `db.zips.updateMany({state:{$in:["PA","VA"]}},{$set:{check:true}})`

```
{
  _id: '15031',
  city: 'CUDDY',
  loc: [ -80.164432, 40.360069 ],
  pop: 1037,
  state: 'PA',
  check: true
}
```

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

```
➤ db.zips.find({"loc.1":
                {$gt:55,$lt:65}},{_id:0,pop:1})
```

```
it> db.zips.find({"loc.1": {$gt:55,$lt:65}},{_id:0,pop:1})
[
  { pop: 14436 }, { pop: 15891 },
  { pop: 12534 }, { pop: 32383 },
  { pop: 7907 }, { pop: 7979 },
  { pop: 20128 }, { pop: 18356 },
  { pop: 17094 }, { pop: 29857 },
  { pop: 15192 }, { pop: 8116 },
  { pop: 119 }, { pop: 481 },
  { pop: 285 }, { pop: 1186 },
  { pop: 1698 }, { pop: 352 },
  { pop: 296 }, { pop: 7188 }
]
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.

```
➤ db.zips.createIndex( { state: 1 } )
```

```
ti> db.zips.getIndexes()
{ v: 2, key: { _id: 1 }, name: '_id_' },
{ v: 2, key: { state: 1 }, name: 'state_1' }
```

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

```
➤ db.zips.updateMany({state:{$nin:["AK","NY"]}},{$inc:{pop:0.2}})
```

```
{
  _id: '01035',
  city: 'HADLEY',
  loc: [ -72.571499, 42.36062 ],
  pop: 4231.2,
  state: 'MA',
  check: false
}
```

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

```
➤ db.zips.updateOne ({ $and: [
  {state:{$ne:"MA"}},
  { pop:{$lt:200}},
  {"loc.0":{$lt:-71}}]],
  {$set:{pop:0}})
```

```
{
  _id: '99549',
  loc: [ -158.566367, 56.964333 ],
  pop: 119,
  state: 'AK',
  check: false,
  country: 'PORT HEIDEN'
}
```

```
ti> db.zips.findOne ({ $and: [{state:{$ne:"MA"}},
  { pop:{$lt:200}},
  {"loc.0":{$lt:-71}}]]
{
  _id: '99549',
  loc: [ -158.566367, 56.964333 ],
  pop: 0,
  state: 'AK',
  check: false,
  country: 'PORT HEIDEN'
}
```

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 document in the database but change the _id to avoid duplications.

```
➤ Var first_doc=db.zips.findOne();
➤ first_doc='5683'
➤ db.zips.updateMany(
  { },
  {$rename: {'city':
    'country'},
  $setOnInsert:{first_doc} },
  {upsert:true})
```

```
ti> db.zips.findOne();
{
  _id: '01005',
  loc: [ -72.108354, 42.409698 ],
  pop: 4546.2,
  state: 'MA',
  check: false,
  country: 'BARRE'
}
```

part2

//sum of population

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

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

```
[ { _id: 'PA', totalPop: 11881934.6 } ]
```

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

```
> db.zips.aggregate([
  {
    $match : { state :{
      $nin:["PA","KA"]} }},
  { $limit : 5}
])
```

```
[
  {
    id: '99991',
    loc: [ -149.876677, 61.211571 ],
    pop: 14436,
    state: 'AK',
    check: false,
    country: 'ANCHORAGE'
  },
  {
    id: '99992',
    loc: [ -150.093943, 61.096163 ],
    pop: 15893,
    state: 'AK',
    check: false,
    country: 'ANCHORAGE'
  },
  {
    id: '99993',
    loc: [ -149.593844, 61.189953 ],
    pop: 12534,
    state: 'AK',
    check: false,
    country: 'ANCHORAGE'
  },
  {
    id: '99994',
    loc: [ -149.74467, 61.203696 ],
    pop: 12383,
    state: 'AK',
    check: false,
    country: 'ANCHORAGE'
  },
  {
    id: '99995',
    loc: [ -149.512667, 61.251531 ],
    pop: 7987,
    state: 'AK',
    check: false,
    country: 'ELMENDORF AFB'
  }
]
```

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

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

```
[ { _id: 'AK', totalPop: 524636 } ]
```

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

```
> db.zips.aggregate([
  {$match:{ state:{$in:["PA","KA"]}}},
  {$sort:{pop:1}},
  {$skip:7}
])
```

```
[
  {
    id: '16334',
    loc: [ -79.445929, 41.326077 ],
    pop: 27.2,
    state: 'PA',
    check: true,
    country: 'MARBLE'
  },
  {
    id: '16871',
    loc: [ -78.034056, 41.186798 ],
    pop: 34.2,
    state: 'PA',
    check: true,
    country: 'POTTERSDALE'
  },
  {
    id: '16217',
    loc: [ -79.19708, 41.338366 ],
    pop: 36.2,
    state: 'PA',
    check: true,
    country: 'COOKSBURG'
  }
]
```

- ```
> db.zips.aggregate(
 [
 {$group:
 {
 _id : '$state',
 MaxPopCountry : {$max : "$pop"},
 MinPopCountry : {$min : "$pop"}
 }
 },
 {$out : 'mypop'}
]
)
```

```

iti> db.mypop.find().limit(2)
[ypop
 { _id: 'MN', MaxPopCounty: 51421.2, MinPopCounty: 0.2 },
 { _id: 'OK', MaxPopCounty: 45542.2, MinPopCounty: 8.2 }
]

```

- ```
> db.zips.aggregate([
  { $group:
    {
      _id : '$state',
      avgPop : { $avg : "$pop" }
    }
  })
```

```
{
  -id: 'MN', avgPop: 4958.22947845805
  -id: 'OK', avgPop: 7368.092491467577
  -id: 'CO', avgPop: 5956.129951698821
  -id: 'NV', avgPop: 4155.761539484154
  -id: 'CA', avgPop: 19627.436147757527
  -id: 'AZ', avgPop: 13575.118518518519
  -id: 'KY', avgPop: 4543.443510560799
  -id: 'CT', avgPop: 12498.73992354374
  -id: 'MO', avgPop: 5141.696981891348
  -id: 'NJ', avgPop: 14315.362962962963
  -id: 'RI', avgPop: 10611.269634703196
  -id: 'TX', avgPop: 10164.533333333333
  -id: 'WY', avgPop: 3239.685714285714
  -id: 'DE', avgPop: 12569.48745716981
  -id: 'MA', avgPop: 12693.079746835443
  -id: 'MD', avgPop: 11384.435714285713
  -id: 'MT', avgPop: 2544.62038215695
  -id: 'ID', avgPop: 4126.2204918823785
  -id: 'WV', avgPop: 2733.654260293883
  -id: 'TN', avgPop: 8378.992096219932
}
```

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

- ```
➤ db.zips.aggregate(
 [
 {$sort:
 {
 state :1,
 country :1}
 }
]
)
```

```
{
 _id: '99615',
 loc: [-152.500169, 57.781967],
 pop: 13309,
 state: 'AK',
 check: false,
 country: 'AKHIOK'
},
{
 _id: '99551',
 loc: [-161.39233, 60.891854],
 pop: 481,
 state: 'AK',
 check: false,
 country: 'AKIACHAK'
},
```

- ```
➤ db.zips.aggregate(
  [
    {$sort:
      {
        state :-1,
        country :-1}
      }
  ]
)
```

```
{
  _id: '82244',
  loc: [ -104.353507, 41.912018 ],
  pop: 674.2,
  state: 'WY',
  check: false,
  country: 'YODER'
},
{
  _id: '82732',
  loc: [ -105.532327, 43.829349 ],
  pop: 2132.2,
  state: 'WY',
  check: false,
  country: 'WRIGHT'
},
```

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

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

```
[
  { _id: 'CA', avgPop: 41498.58888888889 },
  { _id: 'NY', avgPop: 44494.818930041154 }
]
```

10. Return the average populations for cities in each state

```
➤ db.zips.aggregate([
  { $group:
    {
      _id: '$state',
      avgPop: { $avg: "$pop" }
    }
  }
])
```

```
[
  { _id: 'MN', avgPop: 4958.22947845805 },
  { _id: 'OK', avgPop: 5368.092491467577 },
  { _id: 'CO', avgPop: 7956.129951698821 },
  { _id: 'NV', avgPop: 11556.28653846154 },
  { _id: 'CA', avgPop: 19627.436147757257 },
  { _id: 'AZ', avgPop: 13575.118518518519 },
  { _id: 'KY', avgPop: 4543.443510506799 },
  { _id: 'CT', avgPop: 12498.739923954374 },
  { _id: 'MO', avgPop: 5141.696981891348 },
  { _id: 'NJ', avgPop: 14315.362962962963 },
  { _id: 'MI', avgPop: 10611.269634703196 },
  { _id: 'TX', avgPop: 40164.53333333333 },
  { _id: 'WY', avgPop: 3239.6857142857143 },
  { _id: 'DE', avgPop: 12569.40754716981 },
  { _id: 'MA', avgPop: 12693.079746835443 },
  { _id: 'MD', avgPop: 11384.435714285713 },
  { _id: 'MT', avgPop: 2544.620382165605 },
  { _id: 'ID', avgPop: 4126.2204918032785 },
  { _id: 'WV', avgPop: 2733.654268292683 },
  { _id: 'TN', avgPop: 8378.992096219932 }
]
```

Type "it" for more