MongoDB Lab2

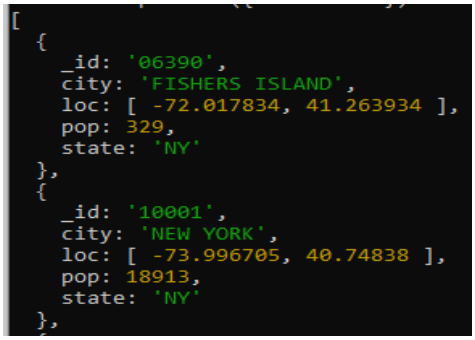
1 - Download the following json file and import it into a collection named “zips” into “iti” database

- 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.
```

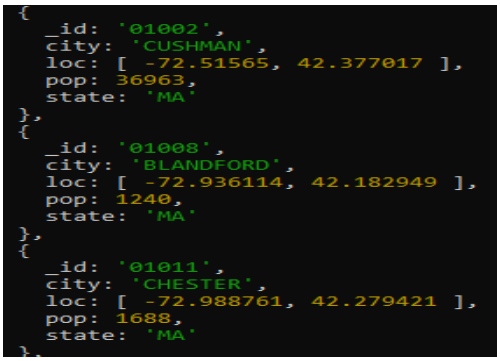
2 – find all documents which contains data related to “NY” state

- `db.zips.find({state:"NY"})`



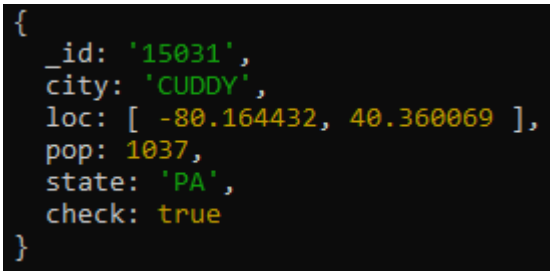
3 – find <u>all zip codes</u> whose population is <u>greater than or equal to 1000</u>

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



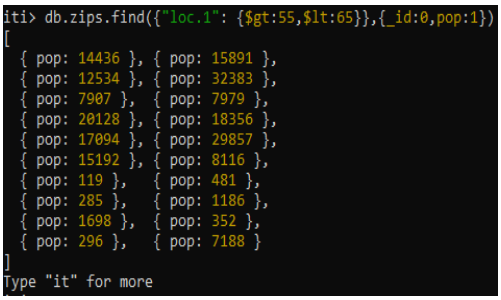
4 – add a new boolean field called “check” and set its value to true for “PA” and “VA” state

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



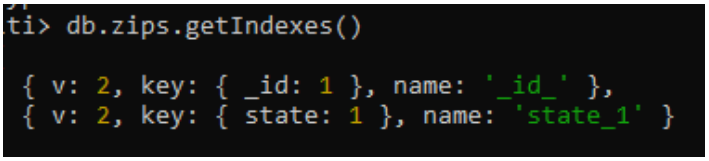
5 – using zip codes find all cities whose latitude is between 55 and 65 and show the population only.

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



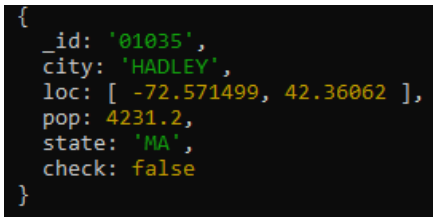
6 – create index for states to be able to select it quickly and check any query explain using the index only.

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



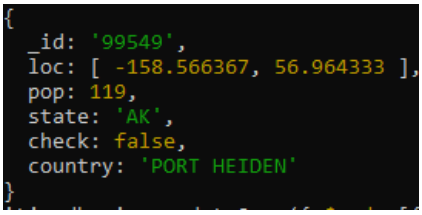
7 – increase the population by 0.2 for all cities which doesn't located in “AK” nor “NY”

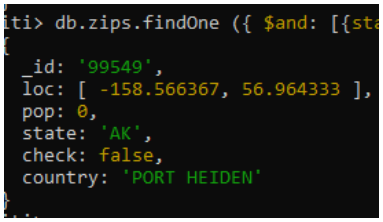
- `db.zips.updateMany({state:{$nin:["AK","NY"]}},{$inc:{pop:0.2}})`



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.

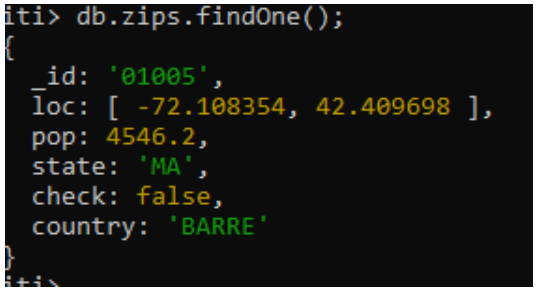
- `db.zips.updateOne ({ $and: [ {state:{$ne:"MA"}}, { pop:{$lt:200}}, {"loc.0":{$lt:-71}}]}, {$set:{pop:0}})`





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.

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



part2
//sum of population

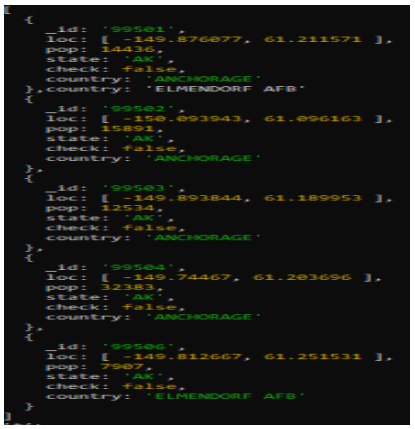
1. Get sum of population that state in PA, KA

```
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}
            ])
```



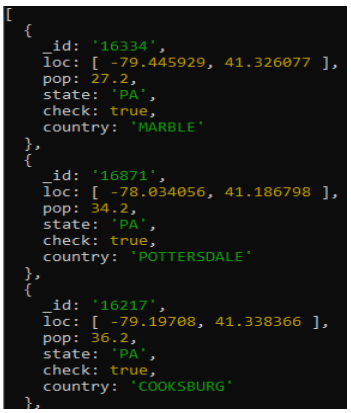
3. Get sum of population that state equal to AK and their latitude between 55, 65

```
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}
      ])
```



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',
               MaxPopCountary :{$max :"$pop"},
               MinPopCountary :{$min :"$pop"}
            }
         },
         {$out :'mypop'}
        ])
```

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

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

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

```
[
  { _id: 'MN', avgPop: 4958.22947845805 },
  { _id: 'OK', avgPop: 5368.092491467577 },
  { _id: 'CO', avgPop: 7956.129951690821 },
  { _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: 10164.533333333333 },
  { _id: 'WY', avgPop: 3239.685714285714 },
  { _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
```

```
// We change city to country in part 1
```

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'
},
```

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

```
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"}
                }
            }
            ])
```