MongoDB – lab2

Part1

1. Download the following json file and import it into a collection named “zips” into “iti” database
    - mongoimport --db iti --collection zips --file D:\ITI-content\mongodb\labs\lab2\zips.json

```
C:\Users\mera>mongoimport --db iti --collection zips --file D:\ITI-content\mongodb\labs\lab2\zips.json
2023-02-24T13:35:28.299+0200    connected to: mongodb://localhost/
2023-02-24T13:35:30.360+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"})

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

3. find all zip codes whose population is greater than or equal to 100
    - db.zips.find({pop:{$gte:1000}})

```
iti> db.zips.find({pop:{$gte:1000}})
[
  {
    _id: '01011',
    city: 'CHESTER',
    loc: [ -72.988761, 42.279421 ],
    pop: 1688,
    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'
  },
```

4. add a new boolean field called “check” and set its value to true for “PA” and “VA” state
    - db.zips.update({},{$set:{check:false}})
    - db.zips.updateMany({$or:[{state:"PA"},{state:"VA"}]},{$set:{check:true}})

```
iti> db.zips.update({},{$set:{check:false}})
DeprecationWarning: Collection.update() is deprecated. Use updateOne, updateMany, or bulkWrite.
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

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

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})

```
iti> 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: 481 },   { pop: 18356 },
  { pop: 285 },   { pop: 29857 },
  { pop: 1186 },  { pop: 185 },
  { pop: 20128 }, { pop: 352 },
  { pop: 17094 }, { pop: 1698 },
  { pop: 296 },   { pop: 8116 },
  { pop: 15192 }, { pop: 7188 }
]
```

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})
    - db.zips.getIndexes()

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

```
iti> 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}})

```
iti> db.zips.updateMany({state:{$nin:["AK","NY"]}},{$inc:{pop:0.2}})
{
  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.
    - db.zips.updateOne ({ $and: [{ pop:{$lt:200}},{state:{$ne:"MA"}},{"loc.0":{$lt:-71}}]},{$set:{pop:0}})

```
iti>  db.zips.updateOne ({ $and: [{ pop:{$lt:200}},{state:{$ne:"MA"}},{"loc.0":{$lt:-71}}]},{$set:{pop:0}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  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.
    - db.zips.updateMany({},{$rename:{'city':'country'}})

part2

1. Get sum of population that state in PA, KA
    - db.zips.aggregate([{$match:{ state:{$in:["PA","KA "]}}},{$group:{_id :0, sum :{$sum :"$pop"}}}])

```
iti> db.zips.aggregate([{$match:{ state:{$in:["PA","KA "]}}},{$group:{_id :0, sum :{$sum :"$pop"}}}])
[ { _id: 0, sum: 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:0,sum:{$sum :"$pop"}}}])

```
[ { _id: 0, sum: 524570 } ]
```

4. Sort Population of document that state in AK, PA and skip first 7 document
    - db.zips.aggregate([{$match:{state:{$in:["KA","PA"]}}},{$sort:{pop:1}},{$skip:7}])

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

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_pop :{$max :"$pop"},min_pop :{$min :"$pop"}}},{$out :'mypop'}])

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

iti> db.mypop.find()
[
  { _id: 'AK', max_pop: 32383, min_pop: 0 },
  { _id: null, max_pop: null, min_pop: null },
  { _id: 'RI', max_pop: 53733.2, min_pop: 45.2 },
  { _id: 'ME', max_pop: 40434.2, min_pop: 0.2 },
  { _id: 'NJ', max_pop: 69646.2, min_pop: 17.2 },
  { _id: 'MT', max_pop: 40121.2, min_pop: 7.2 },
  { _id: 'MS', max_pop: 46968.2, min_pop: 0.2 },
  { _id: 'IL', max_pop: 112047.2, min_pop: 0.2 },
  { _id: 'OK', max_pop: 45542.2, min_pop: 8.2 },
  { _id: 'ND', max_pop: 42195.2, min_pop: 12.2 },
  { _id: 'IN', max_pop: 56543.2, min_pop: 75.2 },
  { _id: 'NM', max_pop: 57502.2, min_pop: 0.2 },
  { _id: 'MO', max_pop: 54994.2, min_pop: 0.2 },
  { _id: 'MD', max_pop: 76002.2, min_pop: 1.2 },
  { _id: 'NC', max_pop: 69179.2, min_pop: 0.2 },
  { _id: 'NY', max_pop: 111396, min_pop: 0 },
  { _id: 'LA', max_pop: 58905.2, min_pop: 0.2 },
  { _id: 'PA', max_pop: 80454.2, min_pop: 0.2 },
  { _id: 'SC', max_pop: 66990.2, min_pop: 0.2 },
  { _id: 'UT', max_pop: 55999.2, min_pop: 9.2 }
]
```

6. Write an aggregation expression to calculate the average population of a zip code (postal code) by state
    - db.zips.aggregate([{$group:{_id :'$state',avg_pop:{$avg :"$pop"}}}])

```
iti> db.zips.aggregate([{$group:{_id :'$state',avg_pop:{$avg :"$pop"}}}])
[
  { _id: 'AK', avg_pop: 2792.374358974359 },
  { _id: null, avg_pop: null },
  { _id: 'RI', avg_pop: 14539.591304347827 },
  { _id: 'ME', avg_pop: 2992.0243902439024 },
  { _id: 'NJ', avg_pop: 14315.362962962963 },
  { _id: 'MT', avg_pop: 2544.620382165605 },
  { _id: 'MS', avg_pop: 7088.949311294766 },
  { _id: 'IL', avg_pop: 9238.33742926435 },
  { _id: 'OK', avg_pop: 5368.092491467577 },
  { _id: 'ND', avg_pop: 1632.6092071611251 },
  { _id: 'IN', avg_pop: 8201.584615384616 },
  { _id: 'NM', avg_pop: 5489.580434782609 },
  { _id: 'MO', avg_pop: 5141.696981891348 },
  { _id: 'MD', avg_pop: 11384.435714285713 },
  { _id: 'NC', avg_pop: 9402.521985815603 },
  { _id: 'NY', avg_pop: 11279.248902821317 },
  { _id: 'LA', avg_pop: 9089.844396551724 },
  { _id: 'PA', avg_pop: 8149.475034293552 },
  { _id: 'SC', avg_pop: 9962.208571428571 },
  { _id: 'UT', avg_pop: 8404.346341463415 }
]
```

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}}])

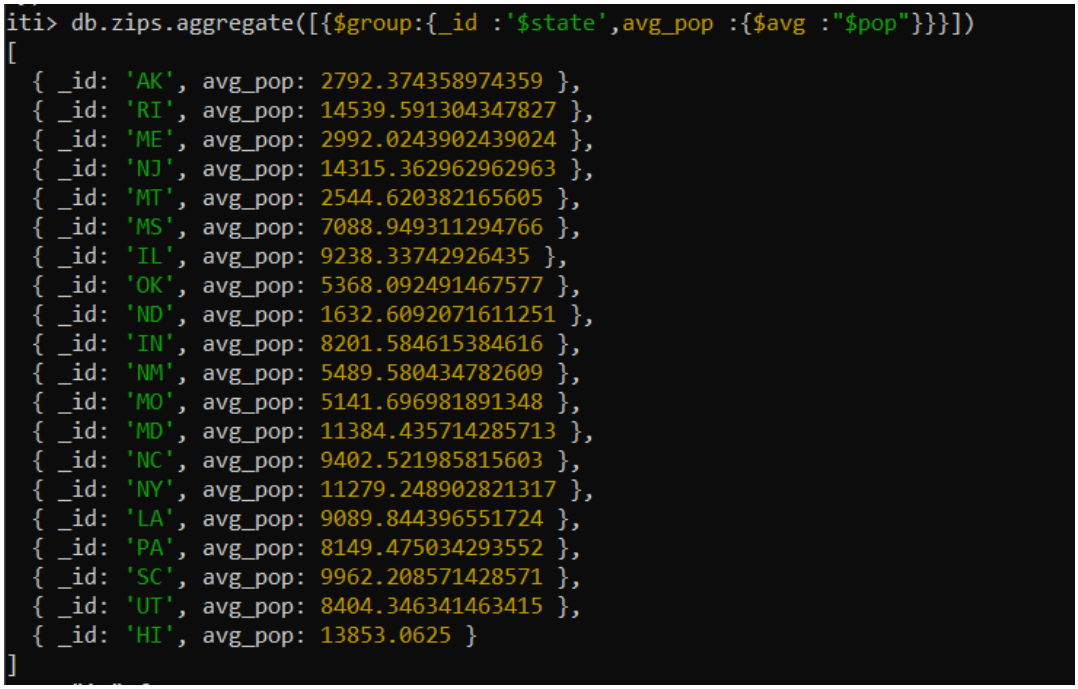
8. Write an aggregation query with just a sort stage to sort by (state, city), both descending
    - db.zips.aggregate([{$sort:{state :-1,country :-1}}])

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 :'$state',avg_pop :{$avg :"$pop"}}}])

```
iti> db.zips.aggregate([{$match:{ state:{$in:["CA","NY"]},pop:{$gt:25000}}},{$group:{_id :'$state',avg_pop :{$avg :"$pop"}}}])
[
  { _id: 'CA', avg_pop: 41498.58888888889 },
  { _id: 'NY', avg_pop: 44494.818930041154 }
]
```

10. Return the average populations for cities in each state

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



```
iti> db.zips.aggregate([{$group:{_id :'$state',avg_pop :{$avg :"$pop"}}}])
[
  { _id: 'AK', avg_pop: 2792.374358974359 },
  { _id: 'RI', avg_pop: 14539.591304347827 },
  { _id: 'ME', avg_pop: 2992.0243902439024 },
  { _id: 'NJ', avg_pop: 14315.362962962963 },
  { _id: 'MT', avg_pop: 2544.620382165605 },
  { _id: 'MS', avg_pop: 7088.949311294766 },
  { _id: 'IL', avg_pop: 9238.33742926435 },
  { _id: 'OK', avg_pop: 5368.092491467577 },
  { _id: 'ND', avg_pop: 1632.6092071611251 },
  { _id: 'IN', avg_pop: 8201.584615384616 },
  { _id: 'NM', avg_pop: 5489.580434782609 },
  { _id: 'MO', avg_pop: 5141.696981891348 },
  { _id: 'MD', avg_pop: 11384.435714285713 },
  { _id: 'NC', avg_pop: 9402.521985815603 },
  { _id: 'NY', avg_pop: 11279.248902821317 },
  { _id: 'LA', avg_pop: 9089.844396551724 },
  { _id: 'PA', avg_pop: 8149.475034293552 },
  { _id: 'SC', avg_pop: 9962.208571428571 },
  { _id: 'UT', avg_pop: 8404.346341463415 },
  { _id: 'HI', avg_pop: 13853.0625 }
]
```