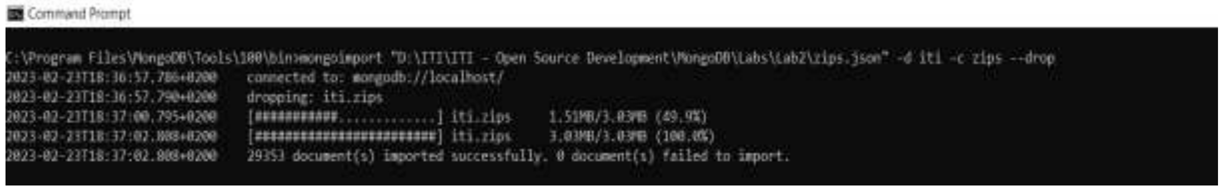
MongoDB Lab-2

PART 1

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

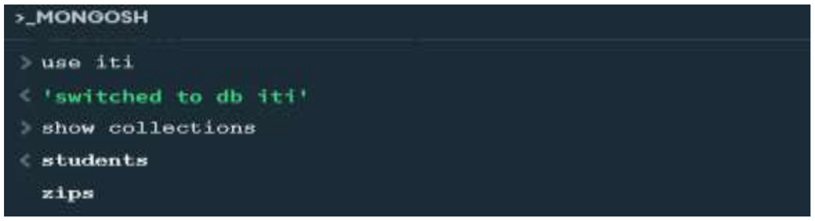
COMMAND

```
>>  cd “C:\Program Files\MongoDB\Tools\100\bin”
>> mongoimport "D:\ITI\ITI - Open Source Development\MongoDB\Labs\Lab2\zips.json" -d iti -c zips --drop
```



Command Prompt

```
C:\Program Files\MongoDB\Tools\100\bin>mongoimport "D:\ITI\ITI - Open Source Development\MongoDB\Labs\Lab2\zips.json" -d iti -c zips --drop
2023-02-23T18:36:57.786+0200    connected to: mongodb://localhost/
2023-02-23T18:36:57.790+0200    dropping: iti.zips
2023-02-23T18:37:00.795+0200    [###########.............] iti.zips     1.51MB/3.03MB (49.9%)
2023-02-23T18:37:02.808+0200    [########################] iti.zips     3.03MB/3.03MB (100.0%)
2023-02-23T18:37:02.808+0200    29353 document(s) imported successfully. 0 document(s) failed to import.
```



```
>_MONGOSH
> use iti
< 'switched to db iti'
> show collections
< students
  zips
```

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

COMMAND >> **db.zips.find({state:'NY'})**

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

COMMANDS >> **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

COMMAND

```
>>   db.zips.updateMany({},{$set:{check:false}})

>> db.zips.updateMany({$or:[{state:'PA'},{state:'VA'}]},{$set:{check:true}})
```

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

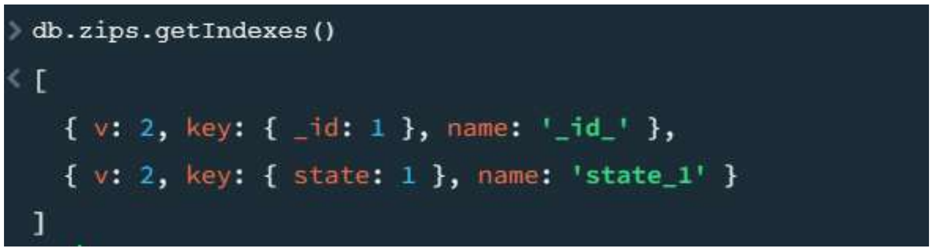
COMMANDS

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

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

COMMANDS

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



```
> 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”

COMMANDS

>> db.zips.updateMany({state:{$nin:['AK','NY']}},{$mul:{pop:1.2}})

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

COMMANDS

>> db.zips.updateOne({'loc.0':{$lt:-71},state:{$nin:['MA']},pop:{$lt:200}},{$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. **Hint**: use Variables

COMMANDS

```
>>  db.zips.updateMany({},{$rename:{'city':'country'}})
```

PART 2

1 Get sum of population that state in PA, KA

COMMAND >>

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

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

COMMAND >> `db.zips.find({state:{$nin: ['PA','KA']}}).limit(5)`

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

COMMAND >>

```
db.zips.aggregate([
        {$match:{state:{$in:['AK']},'loc.1':{$gt:55,$lt:65}}},
        {$group:{_id:0,totalNum:{$sum:'$pop'}}}
])
```

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

COMMAND

```
>>  db.zips.find({state:{$in:['AK','PA']}}).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

COMMAND >>

```
db.zips.aggregate(
{$group:{_id:'$state',minPop:{$min:'$pop'},maxPop:{$max:'$pop'}}},{$out:'mypo
p'})
```

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

COMMAND >>

```
db.zips.aggregate(
      {$group:{_id:'$state',average:{$avg:'$pop'}}}
)
```

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

COMMAND >>

```
db.zips.aggregate(
   {$sort:{state:1,city:1}}
)
```

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

COMMAND >>

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

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

COMMAND >>

```
db.zips.aggregate(
      {$match:{state:{$in:['CA','NY']},pop:{$gt:25000}}},
      {$group:{_id:0,average:{$avg:'$pop'}}}
)
```

10	Return the average populations for cities in each state

COMMAND >>

```
db.zips.aggregate(
    {$group:{_id:'$state',average:{$avg:'$pop'}}}
)
```