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

```
C:\Users\Rogina>mongoimport --db iti1 --collection zips --file "C:\Users\Rogina\Downloads\zips.json"
2023-04-18T19:12:06.278+0200    connected to: mongodb://localhost/
2023-04-18T19:12:08.565+0200    29353 document(s) imported successfully. 0 document(s) failed to import.
```

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

```
iti1> db.zips.aggregate([{$match:{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

```
iti1> db.zips.aggregate([{$match:{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'
  },
```

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

```
...   { state: { $in: ["PA", "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.

```
iti1> db.zips.aggregate([
...   {
...     $match: {
...       loc: {
...         $elemMatch: {
...           $gte: 55,
...           $lte: 65
...         }
...       }
...     }
...   },
...   {
...     $project: {
...       _id: 0,
...       city: 1,
...       pop: 1
...     }
...   }
... ])
[
  { city: 'ANCHORAGE', pop: 14436 },
  { city: 'ANCHORAGE', pop: 15891 },
  { city: 'ANCHORAGE', pop: 12534 },
  { city: 'ANCHORAGE', pop: 32383 },
```

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

```
iti1> db.zips.createIndex({state: 1})
state_1
```

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

```
iti1> db.zips.updateMany({$or:[{state:{$not:{$regex:"NY"}}},{state:{$not:{$regex:"AK"}}}]},{$mul:{pop:Decimal128("1.20")}})
{
  acknowledged: true,
```

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.

```
iti1> db.zips.updateOne({$and:[{"loc.0":{$lt:-71}},{state:{$not:{$regex:"MA"}}},{pop:{$lt:200}}]},{$set:{pop:0}})
{
  acknowledged: true,
```

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

```
iti1> const new_Id = new ObjectId();

iti1> const new_document = { _id: new_Id, country: "$city" };

iti1> const updateResult = db.zips.updateMany({ city: { $type: "string" } }, { $rename: { city: "country" } });

iti1> const numModified = updateResult.modifiedCount;

iti1> if (numModified === 0) {
...   db.zips.insertOne(new_document);
... } else {
...   console.log("Updated " + numModified + " documents");
... }
{
  acknowledged: true,
  insertedId: ObjectId("64403484e10ba24708822a53")
}
```

****

part2

Get the sum of the population of that state in PA, KA

```
iti1> db.zips.aggregate([
...   { $match: { state: { $in: ["PA", "KA"] } } },
...   { $group: { _id: "$state", population: { $sum: "$pop" } } }
... ])
[ { _id: 'PA', population: Decimal128("17109565.9200") } ]
```

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

```
iti1> db.zips.find({ state: { $nin: ["PA", "KA"] } }).limit(5)
[
  { _id: ObjectId("64403484e10ba24708822a53"), country: '$city' },
  {
    _id: '99501',
    loc: [ -149.876077, 61.211571 ],
    pop: Decimal128("20787.8400"),
    state: 'AK'
  },
  {
    _id: '99502',
    loc: [ -150.093943, 61.096163 ],
    pop: Decimal128("22883.0400"),
    state: 'AK'
  },
  {
    _id: '99503',
    loc: [ -149.893844, 61.189953 ],
    pop: Decimal128("18048.9600"),
    state: 'AK'
  },
  {
    _id: '99504',
    loc: [ -149.74467, 61.203696 ],
    pop: Decimal128("46631.5200"),
    state: 'AK'
  }
]
```

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

```
iti1> db.zips.aggregate([
...   {$match: {state: "AK", "loc.1": {$gt: 55, $lt: 65}}},
...   {$group: {_id: null, totalPop: {$sum: "$pop"}}}
... ])
[ { _id: null, totalPop: Decimal128("755647.2000") } ]
```

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

```
iti1> db.zips.find({state: {$in: ["AK", "PA"]}}).sort({pop: 1}).skip(7).toArray()
[
  {
    _id: '99773',
    loc: [ -157.613496, 66.958141 ],
    pop: Decimal128("0.00"),
    state: 'AK'
  },
  {
    _id: '15744',
    loc: [ -79.093987, 40.921432 ],
    pop: Decimal128("0.00"),
    state: 'PA',
    check: true
  },
```

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

```
iti1> var pipeline = [ { $group: { _id: "$state", smallestPop: { $min: "$pop" }, greatestPop: { $max: "$pop" } } }, { $project: { _id: 0, state:
"$_id", smallestPop: 1, greatestPop: 1 } }, { $out: "mypop" }];
```

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

```
iti1> db.zips.aggregate([{$group:{_id:"$state",avg_population:{$avg:"$pop"}}}])
[
  {
    _id: 'LA',
    avg_population: Decimal128("13089.087931034482758620689655172 41")
  },
  {
    _id: 'RI',
    avg_population: Decimal128("20935.784347826086956521739130434 78")
  },
  {
```

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

```
iti1>   db.zips.aggregate([{$sort:{state:1,city:1}}])
[
  { _id: ObjectId("64403484e10ba24708822a53"), country: '$city' },
  {
    _id: '99501',
```

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

```
iti1> db.zips.aggregate([{$sort:{state:-1,city:-1}}])
[
  {
```

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

```
iti1> db.zips.aggregate([{$match:{state:{$in:["NY","CA"]},pop:{$gt:25000}}},{$group:{_id:
null,avg_population:{$avg:"$pop"}}}])
[
  {
    _id: null,
    avg_population: Decimal128("53488.917353787152444870565675934 80")
  }
]
```

Return the average populations for cities in each state

```
iti1>   db.zips.aggregate([{$group:{_id:"$state",avg_population:{$avg:"$pop"}}}])
[
  {
    _id: 'LA',
    avg_population: Decimal128("13089.087931034482758620689655172 41")
  },
```