MongoDB Lab2
Mariam khaled (open source)

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_OpenSource/zips.json"

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

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

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

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}})

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},{name:"query for inventory"})

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

db.zips.updateMany({state:{$in:["AK","NY"]}},{$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.

db.zips.update({"loc.0":{$lt:71},state:{$ne:"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.

db.zips.updateMany({city:{$type:"string"}},{$rename:{"city":"country"}})

Hint: use Variables
****

part2

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

2. Get only 5 documents that state not equal to PA, KA
```
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
```
db.zips.aggregate({$match:{"loc.1":{$gt:55,$lt:65},state:"AK"}},{$group:{_id:"$state",Totalpop:{$sum:"$pop"}}})
```

4. Sort Population of document that state in AK, PA and skip first 7 document
```
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
```
db.zips.aggregate({$group:{_id:"$state",maxPop:{$max:"$pop"},minPop:{$min:"$pop"}}})
```

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"}}})
```

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({$match:{pop:{$gt:25000}}},{$match:{state:{$in:["CA","NY"]}}},
{$group:{_id:"$state",AvgPop:{$avg:"$pop"}}})

10. Return the average populations for cities in each state

db.zips.aggregate({$group:{_id:"$state",AvgPop:{$avg:"$pop"}}})