Steps:

* Start the container on docker
* After it starts running in the command prompt run docker ps -a
* Then docker exec -it mynode\_kira bash

Importing the given json file:

* We first start the mongod using sudo service mongod start command
* Then use the mongosh command to start the shell
* Then use the following command to import zips.json

mongoimport --db usdata --collection cityinfo --drop --file zips.json

3. Show all collections of the database usdata. Copy and paste the command and its result into hw2.txt.

**Commands:**

test> use usdata switched to db usdata

usdata> show collections

**(output 1)**

cityinfo

4. Find all documents of cityinfo collection. Copy and paste the command and its result into hw2.txt.

**Commands:**

test> show dbs admin 41 kB config 111 kB local 73.7 kB usdata 1.67 MB test> use usdata switched to db usdata usdata> db.cityinfo.find({}).limit(10)

**(output 2)**

[ { \_id: '01002', city: 'CUSHMAN', loc: [ -72.51565, 42.377017 ], pop: 36963, state: 'MA' }, { \_id: '01008', city: 'BLANDFORD', loc: [ -72.936114, 42.182949 ], pop: 1240, state: 'MA' }, { \_id: '01011', city: 'CHESTER', loc: [ -72.988761, 42.279421 ], pop: 1688, state: 'MA' }, { \_id: '01013', city: 'CHICOPEE', loc: [ -72.607962, 42.162046 ], pop: 23396, state: 'MA' }, { \_id: '01020', city: 'CHICOPEE', loc: [ -72.576142, 42.176443 ], pop: 31495, state: 'MA' }, { \_id: '01026', city: 'CUMMINGTON', loc: [ -72.905767, 42.435296 ], pop: 1484, state: 'MA' }, { \_id: '01022', city: 'WESTOVER AFB', loc: [ -72.558657, 42.196672 ], pop: 1764, state: 'MA' }, { \_id: '01028', city: 'EAST LONGMEADOW', loc: [ -72.505565, 42.067203 ], pop: 13367, state: 'MA' }, { \_id: '01027', city: 'MOUNT TOM', loc: [ -72.679921, 42.264319 ], pop: 16864, state: 'MA' }, { \_id: '01010', city: 'BRIMFIELD', loc: [ -72.188455, 42.116543 ], pop: 3706, state: 'MA' } ] usdata>

5. Find all documents with \_id that contains 9503 in it. Do not include "loc" in the output. For example, expected documents in the output may include a document with "\_id":"19503" and a document with "\_id":"95037". Copy and paste the command and its result into hw2.txt.

**Command:**

usdata> db.cityinfo.find({"\_id":/9503/},{loc:0}).limit(10)

**(output 3)**

[ { \_id: '19503', city: 'BALLY', pop: 973, state: 'PA' }, { \_id: '39503', city: 'GULFPORT', pop: 26830, state: 'MS' }, { \_id: '49503', city: 'GRAND RAPIDS', pop: 32876, state: 'MI' }, { \_id: '79503', city: 'AVOCA', pop: 248, state: 'TX' }, { \_id: '89503', city: 'RENO', pop: 23955, state: 'NV' }, { \_id: '95030', city: 'MONTE SERENO', pop: 25881, state: 'CA' }, { \_id: '95032', city: 'LOS GATOS', pop: 18189, state: 'CA' }, { \_id: '95035', city: 'MILPITAS', pop: 50907, state: 'CA' }, { \_id: '95037', city: 'MORGAN HILL', pop: 31309, state: 'CA' }, { \_id: '99503', city: 'ANCHORAGE', pop: 12534, state: 'AK' } ] usdata>

6. Use the aggregate function to answer this question. Find all cities with populations between 23,000 (inclusive) and 150,000 (inclusive) and their state border the Pacific Ocean ("AK", "HI", "WA", "OR", and "CA" are the states that border the Pacific Ocean). Each line of output should present city first, state next, and then total\_population. In the output, a city should be unique for a given state and it should come with the total population of the city. For example, there should be one line of output for LOS ANGELES, CA with total population of LOS ANGELES as shown below.  
...{"city" : "LOS ANGELES", "state": "CA", total\_pop: ....} ...

Copy and paste the command and its result into hw2.txt.

**Command:**

usdata> db.cityinfo.aggregate( [{ $group: { \_id: { state: "$state", city:"$city"}, totalPop: { $sum: "$pop" } } },{ $match: {$and: [ {"\_id.state":{$in:["AK","HI","WA","OR","CA"]}}, {totalPop: { $gte:23000 , $lte:150000}} ] }}, { $project: { \_id: 0, city: "$\_id.city", state: "$\_id.state", totalPop: "$totalPop"}}] ).batchSize(10)

**(output 4)**

[ { city: 'LACEY', state: 'WA', totalPop: 68381 }, { city: 'CHIRIACO SUMMIT', state: 'CA', totalPop: 47118 }, { city: 'TEMECULA', state: 'CA', totalPop: 31175 }, { city: 'KANEOHE', state: 'HI', totalPop: 55236 }, { city: 'CHINO HILLS', state: 'CA', totalPop: 37965 }, { city: 'PORT ANGELES', state: 'WA', totalPop: 30373 }, { city: 'CHINA LAKE NWC', state: 'CA', totalPop: 34246 }, { city: 'LA MIRADA', state: 'CA', totalPop: 40452 }, { city: 'REDMOND', state: 'WA', totalPop: 59751 }, { city: 'PHILLIPS RANCH', state: 'CA', totalPop: 64056 } ] Type "it" for more usdata>

7. Find all zip codes in the cities of San Jose, New York, and Washington (that is, city: "WASHINGTON" and state: "DC") that have a population between 6,000 (inclusive) and 11,000 (inclusive). Copy and paste the command and its result into hw2.txt.

**Command:**

usdata> db.cityinfo.find( {$and:[{city:{$in:["SAN JOSE", "New York","WASHINGTON"]}} ,{pop: { $gte:6000 , $lte:11000}}]},{loc:0, \_id :0, pop:0}).limit(10)

**(output 5)**

[ { city: 'WASHINGTON', state: 'DC' }, { city: 'WASHINGTON', state: 'DC' }, { city: 'WASHINGTON', state: 'GA' }, { city: 'WASHINGTON', state: 'MI' }, { city: 'WASHINGTON', state: 'IA' }, { city: 'SAN JOSE', state: 'CA' }, { city: 'SAN JOSE', state: 'CA' }, { city: 'SAN JOSE', state: 'CA' } ] usdata> usdata>

8. Add an embedded document called "Details" into all documents with a [Santa Clara County zip code](http://www.cs.sjsu.edu/~kim/cs157c/contents/homework/santaclara_zips). In Details, add the following fields with names and values: {county:"Santa Clara", medianIncome: 93500}. Hint: You may use the readFileSync and split functions on mongosh. Copy and paste the command and its result into hw2.txt.

**Command:**

usdata> db.cityinfo.updateMany({ \_id: {$in: ['95009', '95008', '95013', '95014', '95020','94085', '95023', '94087', '94086', '94089', '94088', '95031','95030', '95033', '95032', '95035', '95037', '94301', '95042','94303', '95044', '94305', '95050', '94304', '95046', '94306','95051', '95054', '95070', '95103', '95108', '95111', '95110','95113', '95112', '95117', '95116', '95119', '95118', '95121','95120', '95123', '95122', '95125', '95124', '95127', '95126','95129', '95128', '95131', '95130', '95133', '95132', '95135', '95134', '95136', '95139', '95138', '95141', '95140', '95148','94550', '95151', '95150', '94022', '94024', '95190', '94028','94035', '94040', '94042', '94041', '94043', '95002']} }, { $set: { Details: { county: "Santa Clara", medianIncome: "93500" }}})

**(output 6)**

{ acknowledged: true, insertedId: null, matchedCount: 61, modifiedCount: 60, upsertedCount: 0 } usdata>

9. Find all documents that have an embedded document named Details without using the zip code. Copy and paste the command and its result into hw2.txt.

**Command:**

usdata> db.cityinfo.find({"Details.county": "Santa Clara"}).limit(10)

**(output 7)**  [ { \_id: '94022', city: 'LOS ALTOS', loc: [ -122.125754, 37.381432 ], pop: 17366, state: 'CA', Details: { county: 'Santa Clara', medianIncome: '93500' } }, { \_id: '94024', city: 'LOS ALTOS', loc: [ -122.086205, 37.354745 ], pop: 20795, state: 'CA', Details: { county: 'Santa Clara', medianIncome: '93500' } }, { \_id: '94028', city: 'LADERA', loc: [ -122.208131, 37.378859 ], pop: 6379, state: 'CA', Details: { county: 'Santa Clara', medianIncome: '93500' } }, { \_id: '94035', city: 'MOFFETT FIELD', loc: [ -122.051944, 37.41001 ], pop: 790, state: 'CA', Details: { county: 'Santa Clara', medianIncome: '93500' } }, { \_id: '94040', city: 'MOUNTAIN VIEW', loc: [ -122.087983, 37.385532 ], pop: 26969, state: 'CA', Details: { county: 'Santa Clara', medianIncome: '93500' } }, { \_id: '94041', city: 'MOUNTAIN VIEW', loc: [ -122.078341, 37.389347 ], pop: 13438, state: 'CA', Details: { county: 'Santa Clara', medianIncome: '93500' } }, { \_id: '94086', city: 'SUNNYVALE', loc: [ -122.023771, 37.376407 ], pop: 56215, state: 'CA', Details: { county: 'Santa Clara', medianIncome: '93500' } }, { \_id: '94087', city: 'SUNNYVALE', loc: [ -122.034859, 37.350214 ], pop: 47813, state: 'CA', Details: { county: 'Santa Clara', medianIncome: '93500' } }, { \_id: '94089', city: 'SUNNYVALE', loc: [ -122.000637, 37.398255 ], pop: 13522, state: 'CA', Details: { county: 'Santa Clara', medianIncome: '93500' } }, { \_id: '94043', city: 'MOUNTAIN VIEW', loc: [ -122.077468, 37.405567 ], pop: 28592, state: 'CA', Details: { county: 'Santa Clara', medianIncome: '93500' } } ] Type "it" for more usdata>

10. Use a MongoDB mapreduce function to find the population of every state and show the populations in the output. (Ignore the warning about "deprecated".) Copy and paste the command and its result into hw2.txt.

**Commands:**

usdata> var mapFunction = function(){emit(this.state,this.pop)}

usdata> var reduceFunction = function(key,values){return Array.sum(values)}

usdata> db.cityinfo.mapReduce(mapFunction, reduceFunction, {'out':'mapReduce\_output'})

{ result: 'mapReduce\_output', ok: 1 }

usdata> db.mapReduce\_output.find({})

**(output 8)**

[

{ \_id: 'CA', value: 29754890 },

{ \_id: 'ND', value: 638272 },

{ \_id: 'NY', value: 17990402 },

{ \_id: 'ME', value: 1226648 },

{ \_id: 'WV', value: 1793146 },

{ \_id: 'MA', value: 6016425 },

{ \_id: 'FL', value: 12686644 },

{ \_id: 'IA', value: 2776420 },

{ \_id: 'VA', value: 6181479 },

{ \_id: 'NH', value: 1109252 }

]

Type "it" for more

usdata>

11. Use a MongoDB aggregate function to find the population of every state and show the populations in the output. Copy and paste the command and its result into hw2.txt.

**Command:**

usdata> db.cityinfo.aggregate( [{ $group: { \_id: { state: "$state"}, totalPop: { $sum: "$pop" } } }, {$project: {\_id:0, state:"$\_id.state", totalPopulation: "$totalPop"}}] )

**(output 9)**

[

{ state: 'HI', totalPopulation: 1108229 },

{ state: 'SD', totalPopulation: 695397 },

{ state: 'WA', totalPopulation: 4866692 },

{ state: 'KY', totalPopulation: 3675484 },

{ state: 'DC', totalPopulation: 606900 },

{ state: 'NV', totalPopulation: 1201833 },

{ state: 'AK', totalPopulation: 544698 },

{ state: 'WY', totalPopulation: 453528 },

{ state: 'MI', totalPopulation: 9295297 },

{ state: 'MO', totalPopulation: 5110648 }

]

Type "it" for more

usdata>

12. Compare the execution times of the mapreduce and the aggregate functions to find the population of every state. Follow the instruction below:

* Measure the times before and after a call to the mapreduce function, and get their difference. Copy and paste the command and its result into hw2.txt.

**Commands:**

usdata> var before = new Date();db.cityinfo.mapReduce(mapFunction, reduceFunction, {'out':'mapReduce\_output'}); var after = new Date();

{ result: 'mapReduce\_output', ok: 1 }

usdata> after-before

**(output 10)**

739

* Measure the times before and after a call to the aggregate function, and get their difference. Copy and paste the command and its result into hw2.txt.

**Commands:**

usdata> var before = new Date(); db.cityinfo.aggregate( [{ $group: { \_id: { state: "$state"}, totalPop: { $sum: "$pop" } } }, {$project: {\_id:0, state:"$\_id.state", totalPopulation: "$totalPop"}}] ); var after = new Date();

[

{ state: 'SD', totalPopulation: 695397 },

{ state: 'CO', totalPopulation: 3293755 },

{ state: 'KY', totalPopulation: 3675484 },

{ state: 'WA', totalPopulation: 4866692 },

{ state: 'AK', totalPopulation: 544698 },

{ state: 'DC', totalPopulation: 606900 },

{ state: 'NV', totalPopulation: 1201833 },

{ state: 'WY', totalPopulation: 453528 },

{ state: 'MI', totalPopulation: 9295297 },

{ state: 'MO', totalPopulation: 5110648 }

]

Type "it" for more

usdata> after-before

**(output 11)**

8

* Get the executionTimeMillis" from db.cityinfo.explain("executionStats").aggregate(...) Copy and paste the command and its result into hw2.txt.

**Command:**

usdata> db.cityinfo.explain("executionStats").aggregate( [{ $group: { \_id: { state: "$state"}, totalPop: { $sum: "$pop" } } }, {$project: {\_id:0, state:"$\_id.state", totalPopulation: "$totalPop"}}] )

**(output 12)**

{

explainVersion: '1',

stages: [

{

'$cursor': {

queryPlanner: {

namespace: 'usdata.cityinfo',

indexFilterSet: false,

parsedQuery: {},

queryHash: '530D6DA9',

planCacheKey: 'AFA24CEA',

maxIndexedOrSolutionsReached: false,

maxIndexedAndSolutionsReached: false,

maxScansToExplodeReached: false,

winningPlan: {

stage: 'PROJECTION\_SIMPLE',

transformBy: { pop: 1, state: 1, \_id: 0 },

inputStage: { stage: 'COLLSCAN', direction: 'forward' }

},

rejectedPlans: []

},

executionStats: {

executionSuccess: true,

nReturned: 29353,

executionTimeMillis: 34,

totalKeysExamined: 0,

totalDocsExamined: 29353,

executionStages: {

stage: 'PROJECTION\_SIMPLE',

nReturned: 29353,

executionTimeMillisEstimate: 1,

works: 29355,

advanced: 29353,

needTime: 1,

needYield: 0,

saveState: 30,

restoreState: 30,

isEOF: 1,

transformBy: { pop: 1, state: 1, \_id: 0 },

inputStage: {

stage: 'COLLSCAN',

nReturned: 29353,

executionTimeMillisEstimate: 0,

works: 29355,

advanced: 29353,

needTime: 1,

needYield: 0,

saveState: 30,

restoreState: 30,

isEOF: 1,

direction: 'forward',

docsExamined: 29353

}

}

}

},

nReturned: Long("29353"),

executionTimeMillisEstimate: Long("19")

},

{

'$group': { \_id: { state: '$state' }, totalPop: { '$sum': '$pop' } },

maxAccumulatorMemoryUsageBytes: { totalPop: Long("3672") },

totalOutputDataSizeBytes: Long("23358"),

usedDisk: false,

nReturned: Long("51"),

executionTimeMillisEstimate: Long("29")

},

{

'$project': { state: '$\_id.state', totalPopulation: '$totalPop', \_id: false },

nReturned: Long("51"),

executionTimeMillisEstimate: Long("29")

}

],

serverInfo: {

host: '6129edca66ba',

port: 27017,

version: '5.0.6',

gitVersion: '212a8dbb47f07427dae194a9c75baec1d81d9259'

},

serverParameters: {

internalQueryFacetBufferSizeBytes: 104857600,

internalQueryFacetMaxOutputDocSizeBytes: 104857600,

internalLookupStageIntermediateDocumentMaxSizeBytes: 104857600,

internalDocumentSourceGroupMaxMemoryBytes: 104857600,

internalQueryMaxBlockingSortMemoryUsageBytes: 104857600,

internalQueryProhibitBlockingMergeOnMongoS: 0,

internalQueryMaxAddToSetBytes: 104857600,

internalDocumentSourceSetWindowFieldsMaxMemoryBytes: 104857600

},

command: {

aggregate: 'cityinfo',

pipeline: [

{

'$group': { \_id: { state: '$state' }, totalPop: { '$sum': '$pop' } }

},

{

'$project': { \_id: 0, state: '$\_id.state', totalPopulation: '$totalPop' }

}

],

cursor: {},

'$db': 'usdata'

},

ok: 1

}

usdata>

* + Compare A and B. Write your analysis briefly.

**(output 13)**

In “A” we get the time for execution as 739 whereas for “B” we get the time for execution as 8. So we can say that Aggregate is much faster than MapReduce. This might be because in MapReduce the function first maps the values and then reduces them.

* + Compare B and C. Write your analysis briefly.

**(output 14)**

In “B” we get the time for execution as 8whereas for “C” we get the time for execution as 34. So we can say that Aggregate is much faster without explanation. This might be because when we use explain the function tries to detail everything.