1.

a. db.towns.insertOne({name: "Punxsutawney ", populatiuon: 6200, last\_sensus: ISODate("2022-01-31"), famous\_for: [""], mayor: {name: "Jim Wehrle" }})

db.towns.insertOne({name: "New York",

populatiuon: 22200000,

last\_sensus: ISODate("2022-07-31"),

famous\_for: ["status of liberty", "food"],

mayor: {

name: "Michael Bloomberg",

party: "I"}}

)

db.towns.insertOne({name: "Portland",

populatiuon: 528000,

last\_sensus: ISODate("2022-07-20"),

famous\_for: ["beer", "food"],

mayor: {

name: "Sam Adams",

party: "D"}}

)

B. db.towns.find({"mayor.party":"I"},{name:true, mayor:true, \_id:0})

{ name: 'New York',

mayor: { name: 'Michael Bloomberg', party: 'I' } }

c. db.towns.find({"mayor.party":{$exists:false}},{name:true, mayor:true, \_id:0})

{ name: 'Punxsutawney ', mayor: { name: 'Jim Wehrle' } }

2.

A. fn = function() { return this.gender=="m"; }

db.unicorns.find(fn)

b. var cursor = db.unicorns.find({gender:"m"});null;cursor.sort({name:1}).limit(2);null;

c. cursor.forEach(function(obj){ print(obj.name); })

3. db.unicorns.find({$and:[{gender:"f"}, {weight: {$gt : 500, $lt: 700}}]}.count()

4. db.unicorns.find({}, {loves: 1},{ "\_id":0 })

5. db.unicorns.aggregate({"$group":{\_id:"$gender",count:{$sum:1}}})

6. db.unicorns.save({name: 'Barny', loves: ['grape'], weight: 340, gender: 'm'})

7. db.unicorns.updateOne({name: "Ayna"},{$set: {weight: 700, vampires: 51}})

8. db.unicorns.updateOne({name: "Raleigh"},{$set: {loves: ["redbool"]}})

9. db.unicorns.updateMany({gender: "m"},{$inc: {vampires: 5}}, {multi: true})

10. db.towns.updateOne({name: "Portland"},{$set: {"mayor.party": "No"}})

11. db.unicorns.updateOne({name: "Pilot"},{$push: {loves:"chocolate"}})

12. db.unicorns.updateOne({name: "Aurora"},{$addToSet:{ loves:{$each:["shugar","lemon"]}}})

13. db.towns.insert ({name: "Punxsutawney ",

popujatiuon: 6200,

last\_sensus: ISODate("2022-01-31"),

famous\_for: ["phil the groundhog"],

mayor: {

name: "Jim Wehrle"

}})

db.towns.insert ({name: "New York",

popujatiuon: 22200000,

last\_sensus: ISODate("2022-07-31"),

famous\_for: ["status of liberty", "food"],

mayor: {

name: "Michael Bloomberg",

party: "I"}})

db.towns.insert ({name: "Portland",

popujatiuon: 528000,

last\_sensus: ISODate("2022-07-20"),

famous\_for: ["beer", "food"],

mayor: {

name: "Sam Adams",

party: "D"}})

ЧАСТЬ2

1.

А. db.place.insert({\_id:"tw", name:"town", description:"Cold and dirty"})

db.place.insert({\_id:"fr", name:"forest", description:"Green and beautifull"})

db.place.insert({\_id:"sa", name:"sea", description:"Blue and warm"})

b. db.unicorns.update({\_id:ObjectId("636e3bdc3c6dae7bb1e6b5ee")},{$set:

{place:{$ref:"place", $id: "sa"}}})

db.unicorns.update({\_id:ObjectId("636e3bdc3c6dae7bb1e6b5ef")},{$set:

{place:{$ref:"place", $id: "fr"}}})

2. db.unicorns.ensureIndex({"name" : 1}, {"unique" : true})

[ 'name\_1' ]

3.db.unicorns.getIndexes()

[

{ v: 2, key: { \_id: 1 }, name: '\_id\_' },

{ v: 2, key: { name: 1 }, name: 'name\_1', unique: true }

]

db.unicorns.dropIndex("name\_1")

{ nIndexesWas: 2, ok: 1 }

db.unicorns.getIndexes()

[ { v: 2, key: { \_id: 1 }, name: '\_id\_' } ]

db.unicorns.dropIndex("\_id\_")

MongoServerError: cannot drop \_id index

4. var cursor = db.numbers.find();null;

for(i = 0; i < 100000; i++){db.numbers.insert({value: i})};

db.numbers.find().sort({value: -1}).limit(4)

{ \_id: ObjectId("637649543c6dae7bb1e83c9d"), value: 99999 }

{ \_id: ObjectId("637649543c6dae7bb1e83c9c"), value: 99998 }

{ \_id: ObjectId("637649543c6dae7bb1e83c9b"), value: 99997 }

{ \_id: ObjectId("637649543c6dae7bb1e83c9a"), value: 99996 }

db.numbers.explain("executionStats").find({})

{ explainVersion: '1',

queryPlanner:

{ namespace: 'learn.numbers',

indexFilterSet: false,

parsedQuery: {},

queryHash: '17830885',

planCacheKey: '17830885',

maxIndexedOrSolutionsReached: false,

maxIndexedAndSolutionsReached: false,

maxScansToExplodeReached: false,

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

rejectedPlans: [] },

executionStats:

{ executionSuccess: true,

nReturned: 100000,

executionTimeMillis: 63,

totalKeysExamined: 0,

totalDocsExamined: 100000,

executionStages:

{ stage: 'COLLSCAN',

nReturned: 100000,

executionTimeMillisEstimate: 0,

works: 100002,

advanced: 100000,

needTime: 1,

needYield: 0,

saveState: 100,

restoreState: 100,

isEOF: 1,

direction: 'forward',

docsExamined: 100000 } },

command: { find: 'numbers', filter: {}, '$db': 'learn' },

serverInfo:

{ host: 'DESKTOP-3SVOTEM',

port: 27017,

version: '6.0.2',

gitVersion: '94fb7dfc8b974f1f5343e7ea394d0d9deedba50e' },

serverParameters:

{ internalQueryFacetBufferSizeBytes: 104857600,

internalQueryFacetMaxOutputDocSizeBytes: 104857600,

internalLookupStageIntermediateDocumentMaxSizeBytes: 104857600,

internalDocumentSourceGroupMaxMemoryBytes: 104857600,

internalQueryMaxBlockingSortMemoryUsageBytes: 104857600,

internalQueryProhibitBlockingMergeOnMongoS: 0,

internalQueryMaxAddToSetBytes: 104857600,

internalDocumentSourceSetWindowFieldsMaxMemoryBytes: 104857600 },

ok: 1 }

db.numbers.ensureIndex({"value" : 1}, {"unique" : true})

[ 'value\_1' ]

db.numbers.getIndexes()

[

{ v: 2, key: { \_id: 1 }, name: '\_id\_' },

{ v: 2, key: { value: 1 }, name: 'value\_1', unique: true }

]

db.numbers.find().sort({value: -1}).limit(4)

{ \_id: ObjectId("637649543c6dae7bb1e83c9d"), value: 99999 }

{ \_id: ObjectId("637649543c6dae7bb1e83c9c"), value: 99998 }

{ \_id: ObjectId("637649543c6dae7bb1e83c9b"), value: 99997 }

{ \_id: ObjectId("637649543c6dae7bb1e83c9a"), value: 99996 }

db.numbers.explain("executionStats").find({})

{ explainVersion: '1',

queryPlanner:

{ namespace: 'learn.numbers',

indexFilterSet: false,

parsedQuery: {},

queryHash: '17830885',

planCacheKey: '17830885',

maxIndexedOrSolutionsReached: false,

maxIndexedAndSolutionsReached: false,

maxScansToExplodeReached: false,

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

rejectedPlans: [] },

executionStats:

{ executionSuccess: true,

nReturned: 100000,

executionTimeMillis: 56,

totalKeysExamined: 0,

totalDocsExamined: 100000,

executionStages:

{ stage: 'COLLSCAN',

nReturned: 100000,

executionTimeMillisEstimate: 2,

works: 100002,

advanced: 100000,

needTime: 1,

needYield: 0,

saveState: 100,

restoreState: 100,

isEOF: 1,

direction: 'forward',

docsExamined: 100000 } },

command: { find: 'numbers', filter: {}, '$db': 'learn' },

serverInfo:

{ host: 'DESKTOP-3SVOTEM',

port: 27017,

version: '6.0.2',

gitVersion: '94fb7dfc8b974f1f5343e7ea394d0d9deedba50e' },

serverParameters:

{ internalQueryFacetBufferSizeBytes: 104857600,

internalQueryFacetMaxOutputDocSizeBytes: 104857600,

internalLookupStageIntermediateDocumentMaxSizeBytes: 104857600,

internalDocumentSourceGroupMaxMemoryBytes: 104857600,

internalQueryMaxBlockingSortMemoryUsageBytes: 104857600,

internalQueryProhibitBlockingMergeOnMongoS: 0,

internalQueryMaxAddToSetBytes: 104857600,

internalDocumentSourceSetWindowFieldsMaxMemoryBytes: 104857600 },

ok: 1 }

