**Parte 1 Bases de Datos NoSQL y Relacionales**

1) Los conceptos de RDBMS que existen en MongoDB es la Base de Datos, los demás conceptos tienen otra alternativa, estas son :

Tabla/Relación -> Colección

Fila/Tupla -> Documento

Columna -> Campo

2) Las transacciones de MongoDB antes de la versión 4.0 garantizaba transacciones ACID a nivel de documento, esto quiere decir que las operaciones realizadas en los subdocumentos dentro de un documento cumplían con la condición de que si un error ocurría la base de datos se encargaba de hacer rollback al estado anterior al inicio de la operación.

En la versión MongoDB 4.0 se implementó un modelo transaccional ACID (Atomicidad, Consistencia, Aislamiento y Durabilidad.) sobre diferentes y hace una interfaz agradable para implementar esta funcionalidad dentro de aplicaciones existentes.

Métodos Transacciones MongoDB:

Por un lado, tenemos métodos para gestionar la sesión:

* startSession (), las transacciones están asociadas a una sesión, es por ello que siempre necesitaremos abrir una sesión.
* endSession (), cuando cerremos la sesión, si hay alguna transacción ejecutándose, esta se abortará.

Y por otro los métodos para controlar la transacción:

Session.startTransaction(), este método nos sirve para empezar la transacción multi-documento dentro de una sesión.

Session.commitTransaction(), con este método confirmaremos los cambios que hayamos realizado dentro de la transacción.

Session.abortTransaction(), aborta la ejecución de la transacción y por lo tanto no graba las modificaciones hechas a lo largo de la transacción.

3) MongoDB permite soportar indicies por cualquier campo o subcampo del documento, los tipos de índices son:

● Single field: Además del \_id índice definido por MongoDB, se admite la creación

de índices ascendentes/descendentes definidos por el usuario en un solo campo de

un documento. Para un índice de campo único y operaciones de clasificación, el

orden de clasificación (es decir, ascendente o descendente) de la clave no importa,

porque MongoDB puede atravesar el índice en cualquier dirección.

● Compound index: MongoDB también admite índices definidos por el usuario en múltiples campos. El orden de los campos que figuran en un índice compuesto tiene

importancia.

● Multikey index: MongoDB utiliza índices de múltiples claves para indexar el

contenido almacenado en matrices. Si indexa un campo que contiene un valor de

matriz, MongoDB crea entradas de índice separadas para cada elemento de la

matriz.

● Geoespacial index: Para admitir consultas eficientes de datos de coordenadas

geoespaciales, MongoDB proporciona dos índices especiales: índices 2d que utilizan

geometría plana al devolver resultados e índices 2dsphere que utilizan geometría

esférica para devolver resultados.

● Text index: MongoDB proporciona un text de índice que admite la búsqueda

de contenido de un String en una colección. Estos índices no almacenan palabras de

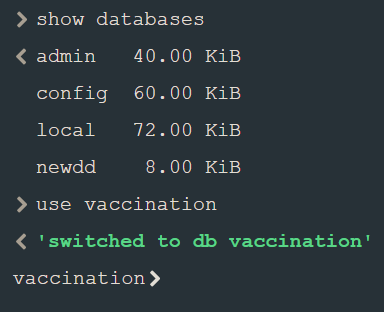
detecciones específicas del idioma (por ejemplo, "the", "a", "or").

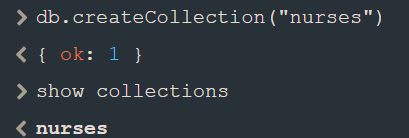
● Hashed index: Para admitir sharding basada en hash, MongoDB proporciona un Hashed Index, que indexa el hash del valor de un campo.

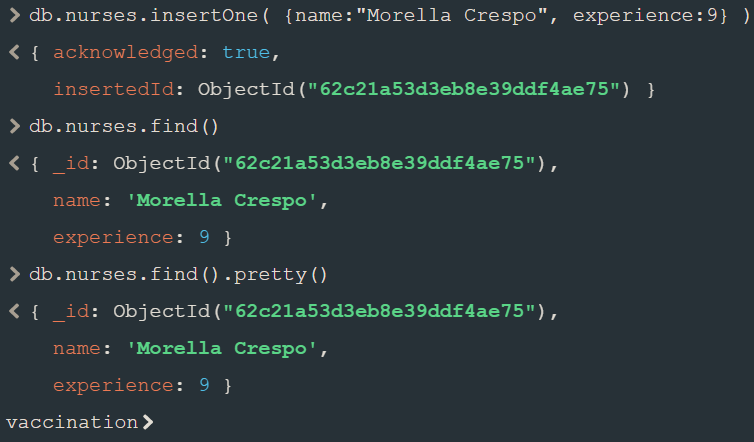
4) No, se remplaza la clave foránea por una referencia. Esta referencia es el id del documento que queramos guardar, se crea un campo nuevo en el documento que tiene esa referencia. Además, MongoDB tiene un Norma DBRef que ayuda a estandarizar la creación de estas referencias.

**Parte 2 Primeros pasos con MongoDB**

5)







use vaccination

db.nurses.insertOne( {name:"Morella Crespo", experience:9} )

db.nurses.find()

db.nurses.find().pretty()

El find ademas de traer los atributos agregados con sus valores correspondientes, te trae el campo id que se genera al insertar.

6)

db.nurses.insertMany (

[

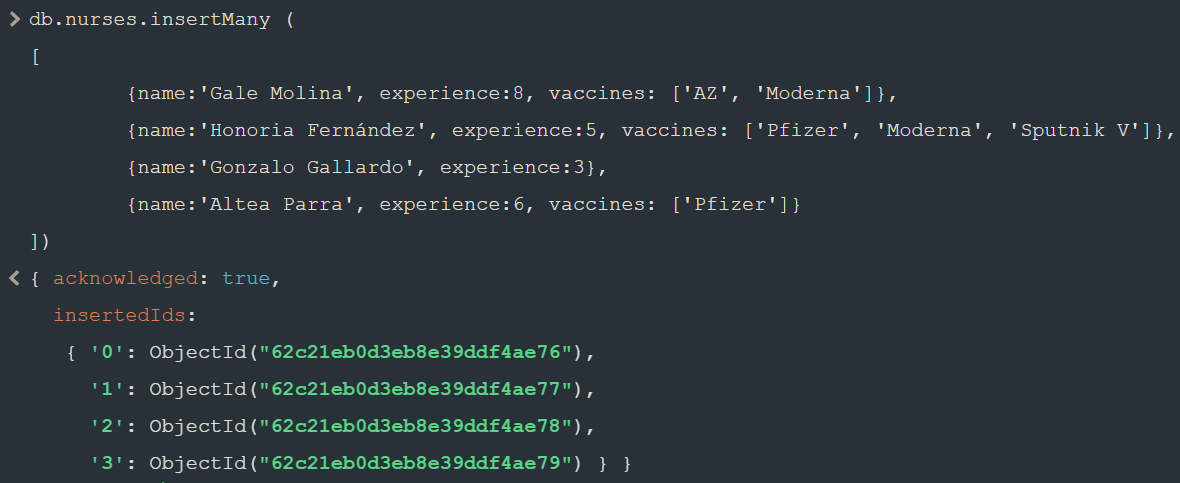
{name:'Gale Molina', experience:8, vaccines: ['AZ', 'Moderna']},

{name:'Honoria Fernández', experience:5, vaccines: ['Pfizer', 'Moderna', 'Sputnik V']},

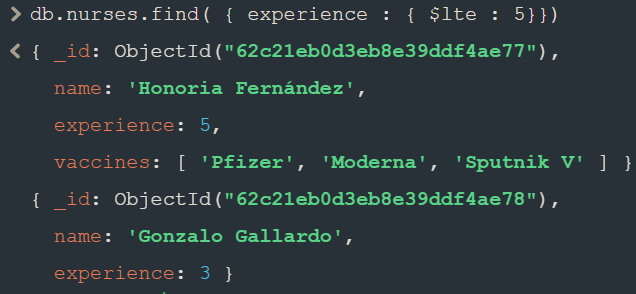
{name:'Gonzalo Gallardo', experience:3},

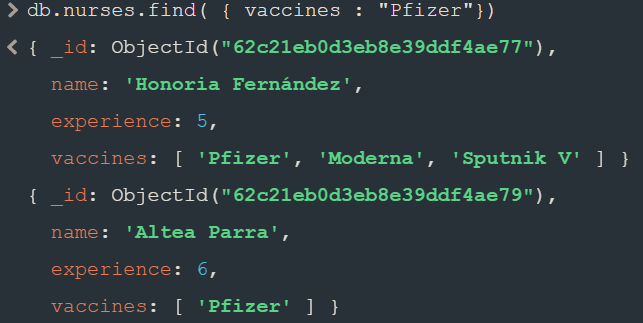
{name:'Altea Parra', experience:6, vaccines: ['Pfizer']}

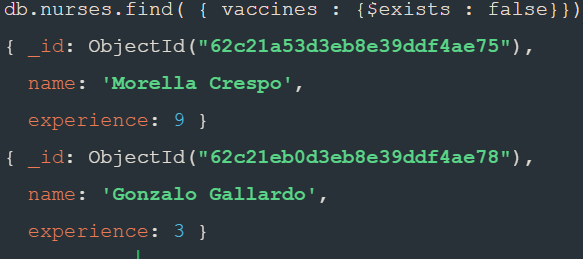
])



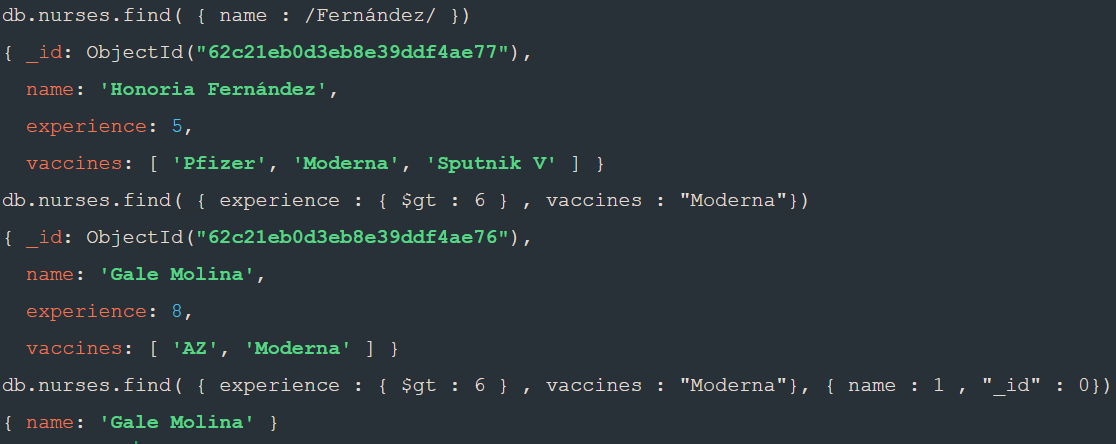
* db.nurses.find( { experience : { $lte : 5}})



* db.nurses.find( { vaccines : "Pfizer"})
* db.nurses.find( { vaccines : {$exists : false}})

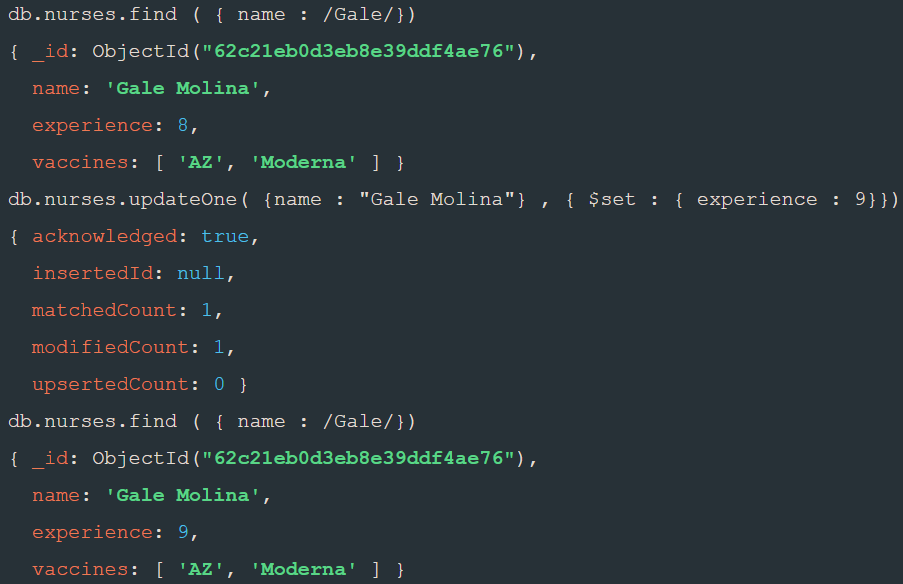


* db.nurses.find( { name : /Fernández/ })
* db.nurses.find( { experience : { $gt : 6 } , vaccines : "Moderna"})
* db.nurses.find( { experience : { $gt : 6 } , vaccines : "Moderna"}, { name : 1 , "\_id" : 0})



7)

* db.nurses.updateOne( {name : "Gale Molina"} , { $set : { experience : 9}})



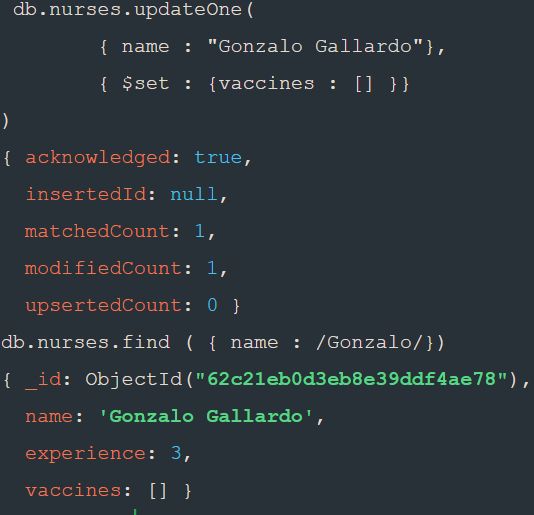
8)

* db.nurses.updateOne(

{ name : "Gonzalo Gallardo"},

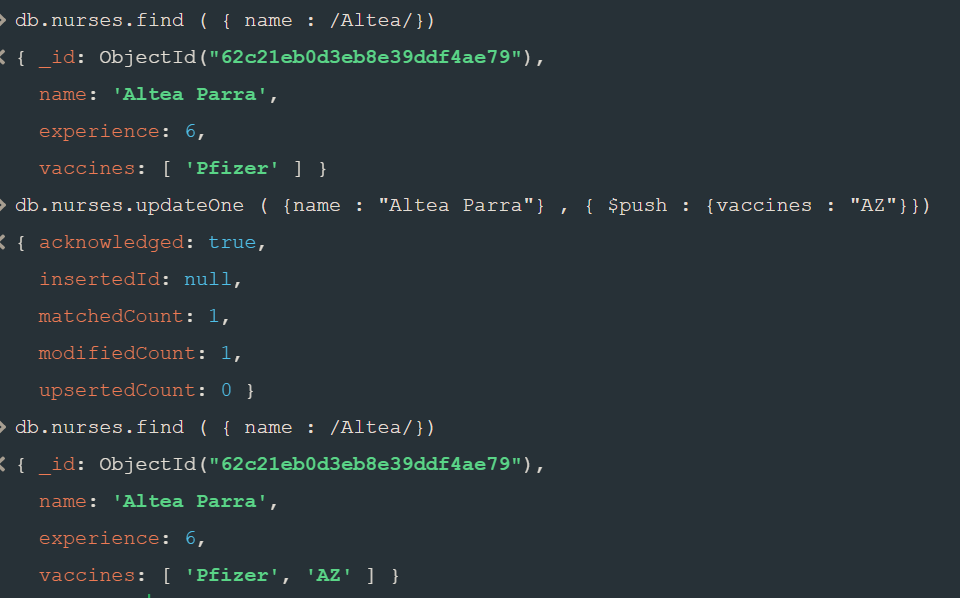
{ $set : {vaccines : [] }}

)



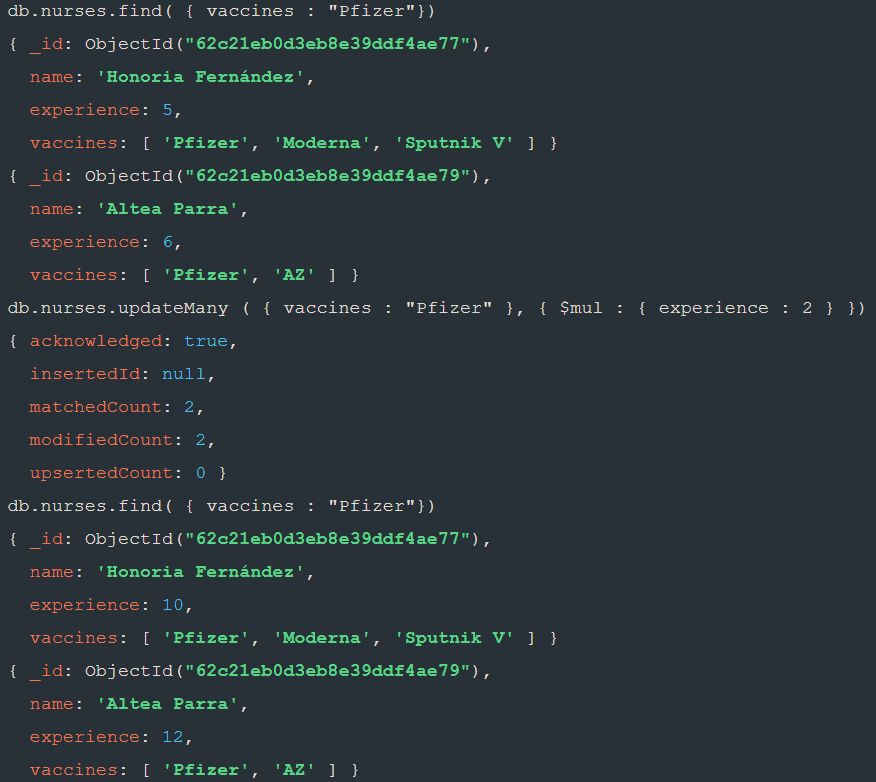
9)

* db.nurses.updateOne ( {name : "Altea Parra"} , { $push : {vaccines : "AZ"}})



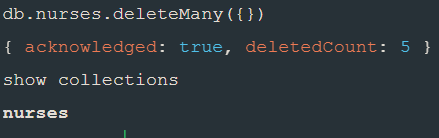
10)

* db.nurses.updateMany ( { vaccines : "Pfizer" }, { $mul : { experience : 2 } })

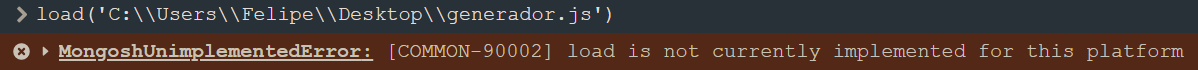


**Parte 3 Índices**

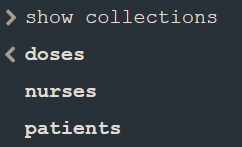
* db.nurses.deleteMany({})



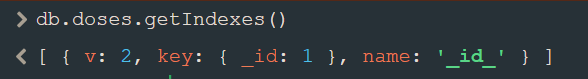
* load('C:\\Users\\Felipe\\Desktop\\generador.js')



// Ejecutamos este comando en Studio 3D ya que en MongoDB Compass o en el Shell de MongoDB no reconoce la instrucción load ()

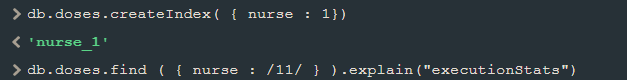


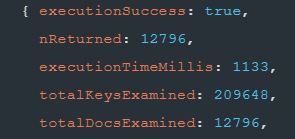
11)

* db.doses.getIndexes()
* 

12)

* db.doses.createIndex( { nurse : 1})
* db.doses.find ( { nurse : /11/ } ).explain("executionStats")

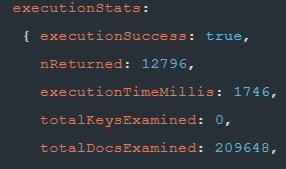




* db.doses.dropIndex( "nurse\_1" )



* db.doses.find ( { nurse : /11/ } ).explain("executionStats")



13)

var caba = {

"type":"MultiPolygon",

"coordinates":[[[

[-58.46305847167969,-34.53456089748654],

[-58.49979400634765,-34.54983198845187],

[-58.532066345214844,-34.614561581608186],

[-58.528633117675774,-34.6538270014492],

[-58.48674774169922,-34.68742794931483],

[-58.479881286621094,-34.68206400648744],

[-58.46855163574218,-34.65297974261105],

[-58.465118408203125,-34.64733112904415],

[-58.4585952758789,-34.63998735602951],

[-58.45344543457032,-34.63603274732642],

[-58.447265625,-34.63575026806082],

[-58.438339233398445,-34.63038297923296],

[-58.38100433349609,-34.62162507826766],

[-58.38237762451171,-34.59251960889388],

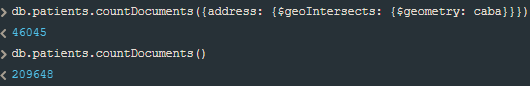
[-58.378944396972656,-34.5843230246475],

[-58.46305847167969,-34.53456089748654]

]]]

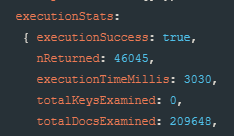
}

* db.patients.find({address: {$geoIntersects: {$geometry: caba}}})
* db.patients.countDocuments({address: {$geoIntersects: {$geometry: caba}}})



(sin indice)

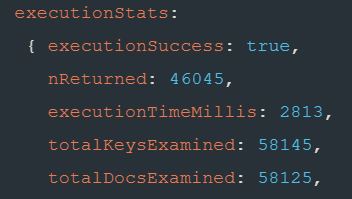
* db.patients.find({address: {$geoIntersects: {$geometry: caba}}}).explain("executionStats")



(con indice)

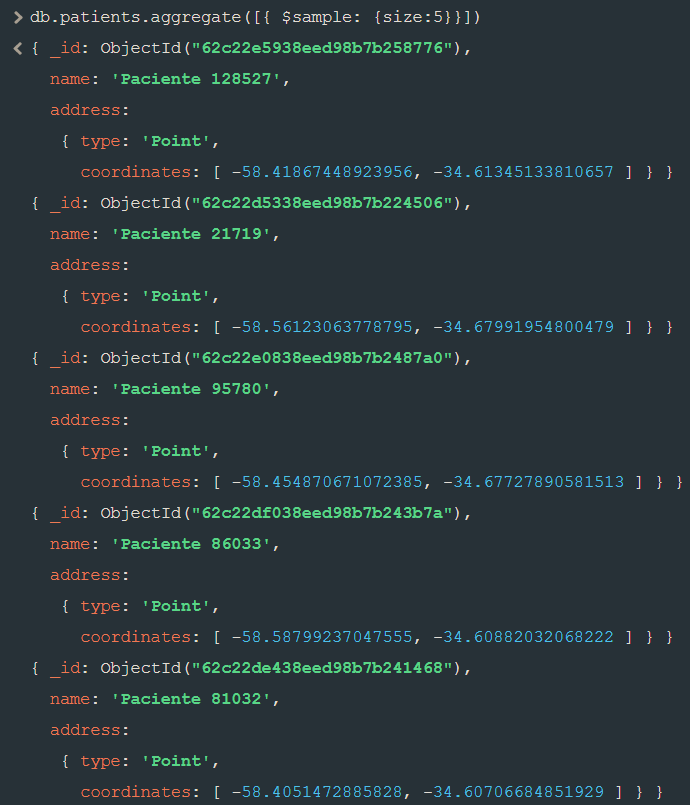
* db.patients.createIndex({address:’2dsphere’})



* db.patients.find({address: {$geoIntersects: {$geometry: caba}}}).explain("executionStats")
* 

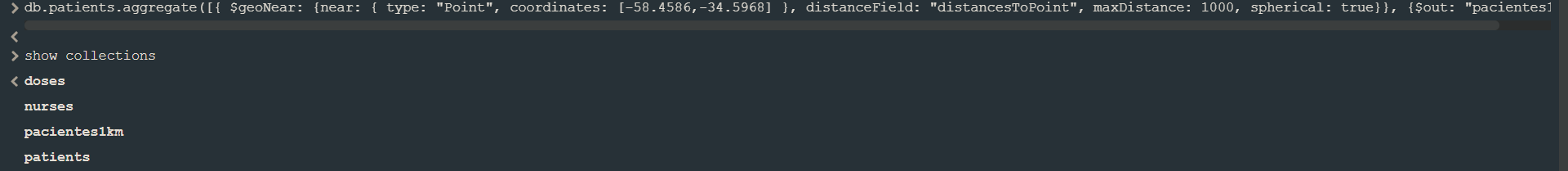
**Parte 4 Aggregation Framework**

**14)**

* db.patients.aggregate([{ $sample: {size:5}}])
* 

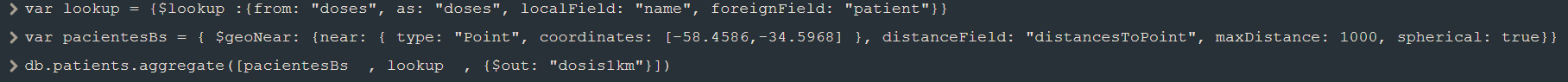
15)

* db.patients.aggregate([{ $geoNear: {near: { type: "Point", coordinates: [-58.4586,-34.5968] }, distanceField: "distancesToPoint", maxDistance: 1000, spherical: true}}, {$out: "pacientes1km"}])



16)

* var lookup = {$lookup :{from: "doses", as: "doses", localField: "name", foreignField: "patient"}}
* var pacientesBs = { $geoNear: {near: { type: "Point", coordinates: [-58.4586,-34.5968] }, distanceField: "distancesToPoint", maxDistance: 1000, spherical: true}}
* db.patients.aggregate([pacientesBs , lookup , {$out: "dosis1km"}])



17)

* db.nurses.aggregate([{ $match: { name: { $regex: ”111” } } }, { $addFields: { doses: [] } }, { $lookup: { from: ”doses”, localField: ”name”, foreignField: “nurse”, as: “doses”, pipeline: [ { $match: { date: { $gt: ISODate(“2021-05-01”) } } } ] } }, { $out: ”pacientes111” }])

