

Laboratorio 2 – Data transformation

Ejercicio 1

1. Conexión a la base de datos a partir de node js

The screenshot shows the Visual Studio Code interface. The left pane displays the code file `connection.js` which connects to a MongoDB database using the `mongodb` module. The right pane shows the Explorer view with the project structure `DB2/Ejercicio1` containing files like `connection.js`, `.gitignore`, `Lab 02 - Data Transformati...`, `package-lock.json`, `package.json`, and `README.md`. The bottom pane shows the terminal output:

```
diggsy@diggsstop:~/UVG/DB2/Ejercicio1$ node connection.js
Databases:
- lab01
- query_performance
- admin
- local
```

- 1.1. Generación de Data Random: Apoyándose del comando `insert`, cree una colección de datos llamada `usuarios` con al menos 100,000 documentos que tenga los siguientes campos (se sugiere leer primero las consultas antes de ponerse a generar la data)
Crear la colección de `usuarios`

The screenshot shows the MongoDB Cloud Atlas interface. In the top window, a 'Create Database' dialog is open, prompting for a 'Database name' (lab2) and a 'Collection name' (usuarios). Below this, the main 'Databases' page is visible, showing 'Cluster0' with two databases: 'lab01' and 'lab2'. Under 'lab2', the 'usuarios' collection is selected. The bottom window shows the 'Collections' tab for 'lab2.usuarios', displaying storage metrics (STORAGE SIZE: 4KB, LOGICAL DATA SIZE: 0B, TOTAL DOCUMENTS: 0, INDEXES TOTAL SIZE: 4KB), and a query interface with a 'Find' button and a search bar.

Se ejecuto en node js

```

connection.js - DB2 - Visual Studio Code

File Edit Selection View Go Run Terminal Help
Welcome connection.js (1 exercise)
You, 2 seconds ago | Author (You) | Click here to ask Blackbox to help you code faster!
1 | const { MongoClient } = require('mongodb');
2 | 
3 | function getRandomDate(startDate, endDate) {
4 |   const startTimestamp = startDate.getTime();
5 |   const endTimestamp = endDate.getTime();
6 | 
7 |   // Generate a random timestamp within the range
8 |   const randomTimestamp = Math.floor(Math.random() * (endTimestamp - startTimestamp)) + startTimestamp;
9 | 
10 |   // Create a new Date object using the random timestamp
11 |   const randomDate = new Date(randomTimestamp);
12 | 
13 |   return randomDate;
14 | }
15 | 
16 | Comment Code
17 | async function listDatabases(client) {
18 |   databaseList = await Client.db().admin().listDatabase();
19 | 
20 |   console.log("Databases:");
21 |   databaseList.databases.forEach(db => console.log(` - ${db.name}`));
22 | }
23 | 
24 | Comment Code
25 | const generateRandomArray = (length) => {
26 |   const randomArray = [];
27 |   for (let i = 0; i < length; i++) {
28 |     const randomNumber = Math.floor(Math.random() * 1000);
29 |     randomArray.push(randomNumber);
30 |   }
31 |   return randomArray;
32 | }
33 | 
34 | Comment Code
35 | async function createUser(client, user) {
36 |   const result = await client.db("lab02").collection("usuarios").insertOne(user);
37 |   return result;
38 | }

PROBLEMS OUTPUT TERMINAL PORTS XPOSE SEARCH ERROR GITLEN COMMENTS DEBUG CONSOLE
dipygondi@dygondi-OptiPlex-5090:~/WGU/002/Ejercicios$ node connection.js

```

Y así es como se ve en atlas mientras se ejecuta

lab2.usuarios

STORAGE SIZE: 3.38MB LOGICAL DATA SIZE: 9.75MB TOTAL DOCUMENTS: 18378 INDEXES TOTAL SIZE: 620KB

[Find](#)

[Indexes](#)

[Schema Anti-Patterns](#)

[Aggregation](#)

[Search Indexes](#)

[INSERT DOCUMENT](#)

[Filter](#)

Type a query: { field: 'value' }

[Reset](#)

[Apply](#)

[Options](#)

QUERY RESULTS: 1-20 OF MANY

```

_id: ObjectId('65c15c5fffc440f4353896e7')
nombre: "nombre0"
email: "nombre@gmail.com"
fecha_registro: 2024-12-29T14:50:20.745+00:00
puntos: 6386
▶ historial_compras: Array (3)
▶ dirección: Object
tags: "tag0"
archivo: true

```

[PREVIOUS](#)

1-20 of many results

[NEXT](#)

```

const url = "mongodb://127.0.0.1:27017/Manejador1230/cluster0.sabth1r.mongodb.net";
const client = new MongoClient(url);
const dbName = "cluster0";
const enddate = new Date("2024-12-31");
try {
    await client.connect();
    console.log("Connected to the MongoDB cluster");
    const db = client.db(dbName);
    const collection = db.collection("usuarios");
    let i = 1;
    for (let k = 1912; k < 1000000; k++) {
        const user = {
            nombre: `Nombre${k}`,
            email: `nombre${k}@gmail.com`,
            fecha_registro: new Date(`2024-01-01T${Math.floor(Math.random() * 1000)}Z`),
            puntos: Math.floor(Math.random() * 1000),
            historial_compras: [
                {
                    producto: `Producto ${k}`,
                    color: `Color ${k}`,
                    fecha: getRndDate(startDate, endDate),
                    cantidad: `Cantidad ${k} K`,
                    codigo_postal: `CP ${k}150`,
                    direccion: `Domicilio ${k}150`,
                    tags: `Tags ${k}150`,
                    archivo: Math.random() > 0.5,
                    notas: `Nota ${k}150`,
                    etiquetas: `Etiquetas ${k}150`,
                    etiquetas_color: `Etiquetas Color ${k}150`,
                    etiquetas_precio: `Etiquetas Precio ${k}150`
                }
            ],
            preferences: {
                color: `Color ${k}150`,
                idiomas: [
                    `Idioma ${k}150`,
                    `Idioma ${k}150`
                ],
                temas: [
                    `Tema ${k}150`,
                    `Tema ${k}150`
                ]
            }
        };
        await collection.insertOne(user);
        i++;
    }
}
catch (e) {
    console.error(e);
}

```

Ya se terminó de ejecutar.

_id	nombre	email	fecha_registro	puntos	historial_compras	dirección	tags	archivo
ObjectId("65c166f40c3fb1550b1e66a2")	Nombre1	nombre1@gmail.com	2024-01-06T14:25:12.651+00:00	1544	(array of 3)	Object	Tags1	True
ObjectId("65c166f40c3fb1550b1e66a3")	Nombre2	nombre2@gmail.com	2024-01-06T14:25:12.652+00:00	1545	(array of 3)	Object	Tags2	True
ObjectId("65c166f40c3fb1550b1e66a4")	Nombre3	nombre3@gmail.com	2024-01-06T14:25:12.653+00:00	1546	(array of 3)	Object	Tags3	True
ObjectId("65c166f40c3fb1550b1e66a5")	Nombre4	nombre4@gmail.com	2024-01-06T14:25:12.654+00:00	1547	(array of 3)	Object	Tags4	True
ObjectId("65c166f40c3fb1550b1e66a6")	Nombre5	nombre5@gmail.com	2024-01-06T14:25:12.655+00:00	1548	(array of 3)	Object	Tags5	True
ObjectId("65c166f40c3fb1550b1e66a7")	Nombre6	nombre6@gmail.com	2024-01-06T14:25:12.656+00:00	1549	(array of 3)	Object	Tags6	True
ObjectId("65c166f40c3fb1550b1e66a8")	Nombre7	nombre7@gmail.com	2024-01-06T14:25:12.657+00:00	1550	(array of 3)	Object	Tags7	True
ObjectId("65c166f40c3fb1550b1e66a9")	Nombre8	nombre8@gmail.com	2024-01-06T14:25:12.658+00:00	1551	(array of 3)	Object	Tags8	True
ObjectId("65c166f40c3fb1550b1e66aa")	Nombre9	nombre9@gmail.com	2024-01-06T14:25:12.659+00:00	1552	(array of 3)	Object	Tags9	True
ObjectId("65c166f40c3fb1550b1e66ab")	Nombre10	nombre10@gmail.com	2024-01-06T14:25:12.660+00:00	1553	(array of 3)	Object	Tags10	True
ObjectId("65c166f40c3fb1550b1e66ac")	Nombre11	nombre11@gmail.com	2024-01-06T14:25:12.661+00:00	1554	(array of 3)	Object	Tags11	True
ObjectId("65c166f40c3fb1550b1e66ad")	Nombre12	nombre12@gmail.com	2024-01-06T14:25:12.662+00:00	1555	(array of 3)	Object	Tags12	True
ObjectId("65c166f40c3fb1550b1e66ae")	Nombre13	nombre13@gmail.com	2024-01-06T14:25:12.663+00:00	1556	(array of 3)	Object	Tags13	True
ObjectId("65c166f40c3fb1550b1e66af")	Nombre14	nombre14@gmail.com	2024-01-06T14:25:12.664+00:00	1557	(array of 3)	Object	Tags14	True
ObjectId("65c166f40c3fb1550b1e66ag")	Nombre15	nombre15@gmail.com	2024-01-06T14:25:12.665+00:00	1558	(array of 3)	Object	Tags15	True
ObjectId("65c166f40c3fb1550b1e66ah")	Nombre16	nombre16@gmail.com	2024-01-06T14:25:12.666+00:00	1559	(array of 3)	Object	Tags16	True
ObjectId("65c166f40c3fb1550b1e66ai")	Nombre17	nombre17@gmail.com	2024-01-06T14:25:12.667+00:00	1560	(array of 3)	Object	Tags17	True
ObjectId("65c166f40c3fb1550b1e66aj")	Nombre18	nombre18@gmail.com	2024-01-06T14:25:12.668+00:00	1561	(array of 3)	Object	Tags18	True
ObjectId("65c166f40c3fb1550b1e66ak")	Nombre19	nombre19@gmail.com	2024-01-06T14:25:12.669+00:00	1562	(array of 3)	Object	Tags19	True
ObjectId("65c166f40c3fb1550b1e66al")	Nombre20	nombre20@gmail.com	2024-01-06T14:25:12.670+00:00	1563	(array of 3)	Object	Tags20	True

Se insertaron 100k de documentos.

1.2. Realizar las consultas

1.2.1. Usuarios activos con más de 500 puntos

```
2 db.usuarios.find({"puntos":{$gt:500}, "archivo":true});
```

Result x Find (1) x

usuarios | 0.022 s | 47.312 Docs

1 /* 1 createdAt:2/8/2024, 7:30:52 AM*/
2 {
3 "_id" : ObjectId("65c4d78c86c94f08402c4430"),
4 "nombre" : "nombre36698",
5 "email" : "nombre36698@gmail.com",
6 "fecha_registro" : ISODate("2023-07-19T15:53:20.902-06:00"),
7 "puntos" : 501,
8 "historial_compras" : [
9 {
10 "producto" : "Producto 1",
11 "fecha" : ISODate("2022-01-26T11:04:52.140-06:00")
12 },
13 {
14 "producto" : "Producto 1",
15 "fecha" : ISODate("2022-09-04T19:26:09.226-06:00")
16 },
17 {
18 "producto" : "Producto 7",
19 "fecha" : ISODate("2022-08-14T23:21:40.729-06:00")
20 }
21],
22 "direccion" : {
23 "calle" : "calle8",
24 "ciudad" : "ciudad4",
25 "codigo_postal" : 98
26 },
27 "tags" : "tag3",
28 "archivo" : true,
29 "notas" : "nota2",
30 "visitas" : 8854,
31 "amigos" : [
32 67942,
33 16348,
34 31144,
35 32661,
36 25217,
37 16640,
38 44629,
39 15921,
40 32052,
41 49385,
42 68531,
43 46200,
44 49978,
45 22818,
46 98371,
47 81982,
48 78704,
49 61908,
50 498,
51 30134,

```
3 db.usuarios.find({"puntos":{$gt:500}, "archivo":true}).count();
```

Error x Find x Find (1) x Find (2) x Find (3) x Result x

0.130 s

1 47719

De manera que son 47719 usuarios con puntos mayores a 500 y activos.

1.2.2. Usuarios que han comprado el producto 1 en la última semana

The screenshot shows the MongoDB Compass interface with a results table. The query is:

```
6 db.usuarios.find({ "historial_compras": { $elemMatch: { "fecha": { $gt: ISODate("2024-01-31T16:00:00Z"), $lt: ISODate("2024-02-07T16:00:00Z") }, "producto": "Producto 1" } } }).projection({"_id":1, "historial_compras":1})
```

The results table has two columns: Key and Value. The Key column shows the user ID, and the Value column shows the historical_compras array. The array contains documents for each purchase, with fields like 'fecha' (date) and 'producto' (product name). There are 182 documents in total.

Y esta sería la cantidad total de usuarios que compraron:

The screenshot shows the MongoDB Compass interface with a results table. The query is:

```
7 db.usuarios.find({ "historial_compras": { $elemMatch: { "fecha": { $gt: ISODate("2024-01-31T16:00:00Z"), $lt: ISODate("2024-02-07T16:00:00Z") }, "producto": "Producto 1" } } }).projection({"_id":1, "historial_compras":1}).count()
```

The results table shows a single row with the value 209.

Son 209 que realizaron compras esta y la semana pasada.

Y esta sería la query para conocer quiénes compraron la semana pasada y compraron el producto 1.

1.2.3. Usuarios con la etiqueta tag 2 y que tienen más de 100 visitas

The screenshot shows the MongoDB Compass interface with a results table. The query is:

```
10 db.usuarios.find({ "tags": "tag2", "visitas": {$gt:100} })
```

The results table shows two user documents. The first user has 102 visits and the second has 3050 visits. Both users have the tag 'tag2'.

Así mismo, son 33050 registros así.

The screenshot shows the MongoDB Compass interface with a results table. The query is:

```
11 db.usuarios.find({ "tags": "tag2", "visitas": {$gt:100} }).count()
```

The results table shows a single row with the value 33050.

1.2.4. Usuarios con preferencias de color azul y que tienen entre 1000 y 2000 amigos.

```

14 db.usuarios.find({ "preferencias.color": "azul" , $expr: { $gt: [ { $size: "$amigos" } , 999 ] } });
15 db.usuarios.find({ "preferencias.color": "azul" , $expr: { $gt: [ { $size: "$amigos" } , 999 ] } }).count();

```

Result x Find x Find (1) x Result (1) x Find (2) x

usuarios | 0.453 s | 33 Docs

Key	Type
_id	ObjectID
nombre	String
email	String
fecha_registro	Date
puntos	Int32
historial_compras	Array
direccion	Object
tags	Array
archivo	Bool
notas	String
visitas	Int32
amigos	Array
preferencias	Object
(1) 65c50561d9649e54f6102bf9	Document
(2) 65c50561d9649e54f6102ea8	Document
(3) 65c50561d9649e54f6102fb4	Document
(4) 65c50562d9649e54f6103f1	Document
(5) 65c50564d9649e54f61048c4	Document
(6) 65c50564d9649e54f61049f7	Document
(7) 65c50567d9649e54f61090ce	Document
(8) 65c50567d9649e54f61090f1	Document
(9) 65c50567d9649e54f610937a	Document
(10) 65c50568d9649e54f6109724	Document
(11) 65c5056dd9649e54f6108cb7	Document
(12) 65c5056ed9649e54f61090b0	Document
(13) 65c50570d9649e54f6109eb6	Document
(14) 65c50573d9649e54f610b2ce	Document
(15) 65c50577d9649e54f610bb0	Document
(16) 65c50578d9649e54f610d1d	Document
(17) 65c50579d9649e54f610d7ae	Document
(18) 65c5057bd9649e54f610df2	Document
(19) 65c5057bd9649e54f610f037	Document
(20) 65c5057ed9649e54f610f58a	Document
(21) 65c5057ed9649e54f610f8dd	Document
(22) 65c50582d9649e54f6112256	Document
(23) 65c50588d9649e54f6112b50	Document

```

15 db.usuarios.find({ "preferencias.color": "azul" , $expr: { $gt: [ { $size: "$amigos" } , 999 ] } }).count();
16 db.usuarios.explain("executionStats").find({ "preferencias": { $elemMatch: { "color": "azul" } } , "amigos": { $gt: [ { $size: "$amigos" } , 999 ] } })

```

Result x Find x Find (1) x Result (1) x

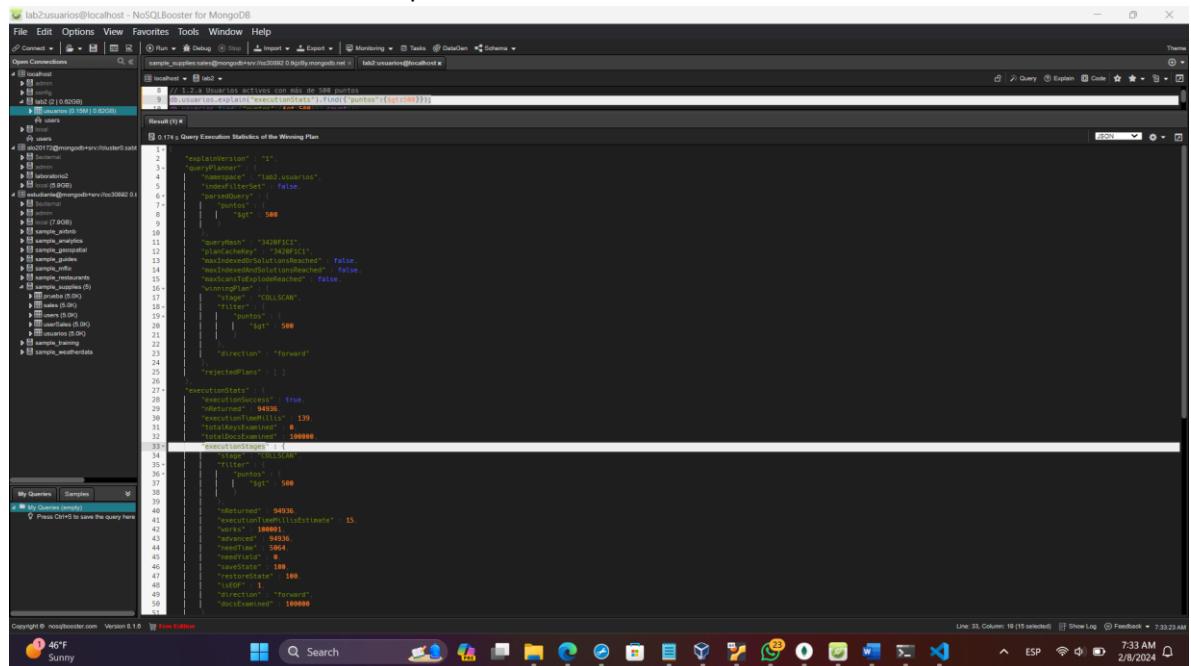
0.480 s

1 33

Son 33 usuarios con azul y amigos entre 1000 y 2000

1.3 Realice una evaluación del rendimiento de cada consulta, apóyese del comando visto en clase, executionStats. Explique qué puede destacar de cada consulta realizada y el rendimiento asociado a la misma.

1.3.1 Usuarios activos con más de 500 puntos.



The screenshot shows the NoSQLBooster for MongoDB interface. In the top navigation bar, the URL is `sample_usuarios.sane@mongodbsrv:20200/test/lab2-usuarios@localhost`. The main window displays the results of a query:

```
sample_usuarios.sane@mongodbsrv:20200/test.lab2-usuarios@localhost
{
    "_id": "5f3e0d0a0000000000000000",
    "username": "jose",
    "password": "1234567890",
    "email": "jose@correo.com",
    "puntos": 500
}
```

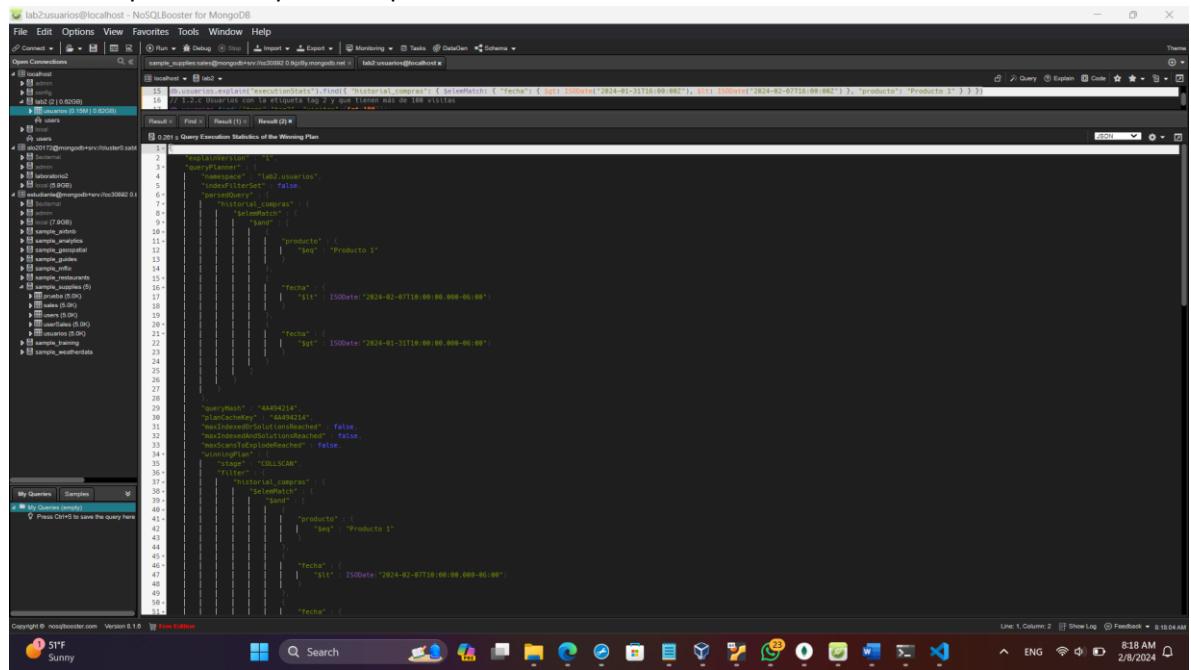
Below the results, the "Result (1)" panel shows the "0.0176s Query Execution Statistics of the Winning Plan". The plan details the following steps:

- 1. "explainVersion": 1
- 2. "queryPlanner": { "maxMemoryLimit": 1024000000 }
- 3. "maxIndexBounds": false
- 4. "parseVerbosity": 0
- 5. "planCacheHit": false
- 6. "parsedQuery": { "puntos": { "\$gt": 500 } }
- 7. "sort": { "puntos": -1 }
- 8. "direction": "forward"
- 9. "rejectedPlans": []
- 10. "executionStats": { "executionTime": 112, "returning": true, "returned": 47719, "executionTimeMillis": 139, "totalKeysExamined": 100000, "totalDocsExamined": 100000 }
- 11. "executionStages": { "stage": "COLLSCAN", "filter": { "puntos": { "\$gt": 500 } }, "returned": 47719, "executionTime": 112, "executionTimeEstimate": 35, "work": 100000, "needTime": 9436, "needTimeEstimate": 5864, "needSize": 0, "restSize": 100000, "restTime": 1, "restTimeEstimate": 100, "restStage": "forward", "docsExamined": 100000 }

The bottom status bar indicates "Copyright © nosqlbooster.com Version 8.1.6" and the system status "46°F Sunny".

Se realizó el query sobre la colección de usuarios en la base de datos lab2. Se aplicó el plan de COLLSCAN de manera que se buscan todos los puntos mayores a 500 y se escanean todos los documentos es por ello que en el total de docs examined aparece 100000, se retornaron 47719 documentos, nótese que no hubieron planes rechazados, así mismo se tardó 112 ms. También se debe de destacar que no se examinó ninguna llave de índice dado que no existe ninguna y que se utilizó collscan, en las etapas de ejecución únicamente se realizó el collscan, así mismo, se destaca que el batchSize o la cantidad de documentos que se retornaron por batch (que es el grupo de documentos que se retornan o que se procesan y que son tratados como una sola unidad) es de 1000.

1.3.2 Usuarios que han comprador el producto 1 en la última semana.



The screenshot shows the NoSQLBooster for MongoDB interface. The left sidebar displays a tree view of databases and collections, including 'sample_usuarios' and 'sample_usuarios.\$index'. The main window shows the results of a query execution plan. The plan details the following stages:

```

1: {
    "stage": "COLLSCAN",
    "filter": {
        "selector": {
            "$and": [
                {
                    "products": {
                        "$eq": "Producto 1"
                    }
                },
                {
                    "fecha": {
                        "$lt": ISODate("2024-02-07T18:00:00.000-06:00")
                    }
                }
            ]
        }
    }
}

```

The execution plan indicates that 209 documents were found, and it took 140ms to execute. The batch size was set to 1000. The results are displayed in a table at the bottom of the interface.

Se puede observar que también se realizó un COLLSCAN de manera que se examinaron todos los documentos para ver si tenían esos filtros (100000 documentos), no se rechazó ningún plan, se retornaron 209 registros, se ejecutó en 140 ms, el batchSize también fue de 1000 documentos. No hubieron índices que se utilizaran de manera que esto no permitió que hubiese un filtro por índices.

1.3.3 Usuarios con la etiqueta tag 2 y que tienen más de 100 visitas

```
{  
    "explainVersion" : "1",  
    "queryPlanner" : {  
        "namespace" : "lab2.usuarios",  
        "indexFilterSet" : false,  
        "parsedQuery" : {  
            "$and" : [  
                {  
                    "tags" : {  
                        "$eq" : "tag2"  
                    }  
                },  
                {  
                    "visitas" : {  
                        "$gt" : 100  
                    }  
                }  
            ]  
        },  
        "queryHash" : "D98E5563",  
        "planCacheKey" : "D98E5563",  
        "maxIndexedOrSolutionsReached" : false,  
        "maxIndexedAndSolutionsReached" : false,  
        "maxScansToExplodeReached" : false,  
        "winningPlan" : {  
            "stage" : "COLLSCAN",  
            "filter" : {  
                "$and" : [  
                    {  
                        "tags" : {  
                            "$eq" : "tag2"  
                        }  
                    },  
                    {  
                        "visitas" : {  
                            "$gt" : 100  
                        }  
                    }  
                ]  
            },  
            "direction" : "forward"  
        },  
        "rejectedPlans" : [ ]  
    },  
}
```

Se utilizó el query plan de COLLSCAN de manera que se examinaron 100000 documentos, también se retornaron 33050 documentos y se tardó 105 ms, no se utilizaron llaves o índices y el batchSize nuevamente fue de 1000.

1.3.4 Usuarios con preferencias de color azul y que tienen entre 1000 y 2000 amigos.

```

1 [
2   "explainVersion" : "1",
3   "queryPlanner" : {
4     "namespace" : "lab2.usuarios",
5     "indexFilterSet" : false,
6     "parsedQuery" : {
7       "$and" : [
8         {
9           "preferencias" : {
10             "$elemMatch" : {
11               "color" : {
12                 "$eq" : "azul"
13               }
14             }
15           }
16         },
17         {
18           "$expr" : {
19             "$gt" : [
20               {
21                 "$size" : [ "$amigos" ]
22               },
23               {
24                 "$const" : 999
25               }
26             ]
27           }
28         }
29       ],
30     },
31     "queryHash" : "92983880",
32     "planCacheKey" : "92983880",
33     "maxIndexedOrSolutionsReached" : false,
34     "maxIndexedAndSolutionsReached" : false,
35     "maxScansToExplodeReached" : false,
36     "winningPlan" : {
37       "stage" : "COLLSCAN",
38       "filter" : {
39         "$and" : [
40           {
41             "preferencias" : {
42               "$elemMatch" : {
43                 "color" : {
44                   "$eq" : "azul"
45                 }
46               }
47             },
48             {
49               "$expr" : {
50

```

Aquí se observa que el plan utilizado fue COLLSCAN nuevamente, se retornaron 33 filas, y se tardó en ejecutarse 458 ms, así mismo se examinaron los 100000 documentos y el batchSize fue de 1000.

1.4 Diseño de índices

- 1.4.1 Cree un índice compuesto sobre esta colección en los campos activo y puntos.

```
25 // 1.4.a Cree un índice compuesto sobre esta colección en los campos activo y puntos.
26 db.usuarios.createIndex({ "activo": 1, "puntos": 1 });
27
```

Result x Find x Result (1) x Result (2) x Result (3) x Result (4) x Result (5) x

0.752 s Tree

Key	Type
4 (1)	Object
numIndexesBefore	Int32
numIndexesAfter	Int32
createdCollectionAutomatically	Bool
ok	Int32

- 1.4.2 Cree un índice compuesto sobre los campos embebidos producto (historial_compras) y fecha (historial_compras).

```
27 // 1.4.b Cree un índice compuesto sobre los campos embebidos producto (historial_compras) y fecha (hi
28 db.usuarios.createIndex({ "historial_compras.fecha":1, "historial_compras.producto":1 });
```

Result x Find x Result (1) x Result (2) x Result (3) x Result (4) x Result (5) x Result (6) x

1.400 s Tree

Key	Type
4 (1)	Object
numIndexesBefore	Int32
numIndexesAfter	Int32
createdCollectionAutomatically	Bool
ok	Int32

- 1.4.3 Cree un índice compuesto sobre los campos de tags y visitas.

```
29 // 1.4.3 Cree un índice compuesto sobre los campos de tags y visitas.
30 db.usuarios.createIndex({ "tags":1, "visitas":1 });
```

Result x Find x Result (1) x Result (2) x Result (3) x Result (4) x Result (5) x Result (6) x Result (7) x

0.584 s Tree

Key	Type
4 (1)	Object
numIndexesBefore	Int32
numIndexesAfter	Int32
createdCollectionAutomatically	Bool
ok	Int32

- 1.4.4 Cree un índice compuesto sobre el campo embebido color (preferencias) y amigos.

```
26 db.usuarios.createIndex({ "preferencias.color":1, "amigos":1 });
```

144.328 s Tree

Key	Type
4 (1)	Object
numIndexesBefore	Int32
numIndexesAfter	Int32
createdCollectionAutomatically	Bool
ok	Int32

- 1.5 Realice una evaluación del rendimiento de cada consulta ahora con los índices creados.

Explique y argumente qué cambios observó comparando los resultados y su análisis del inciso

1.3.

1.5.1 Usuarios activos con más de 500 puntos.



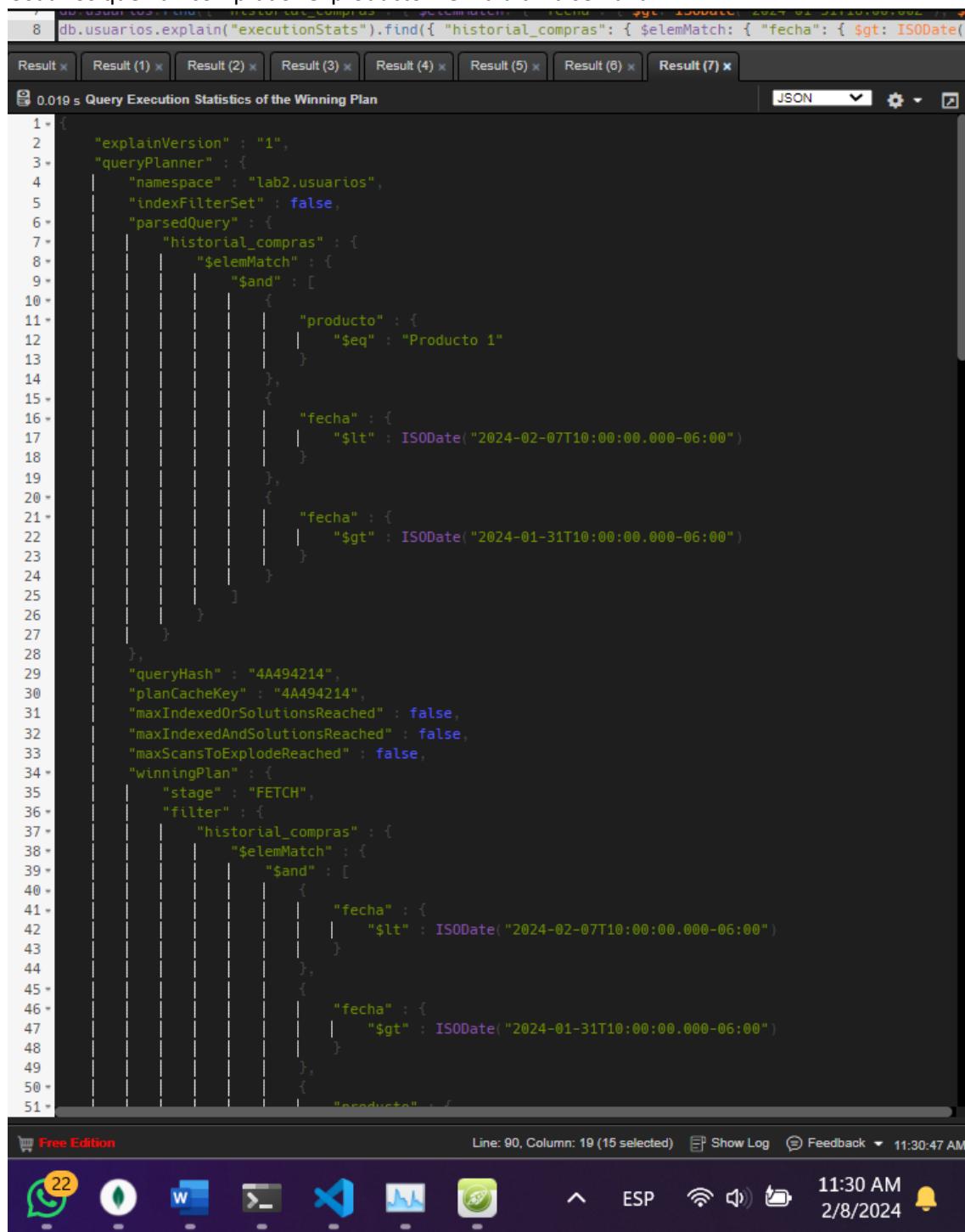
```

4 db.usuarios.explain("executionStats").find({$puntos:{$gt:500}, "archivo":true});
Find < Result < Result (1) <
0.186s Query Execution Statistics of the Winning Plan
1- {
2-   "explainVersion" : "1",
3-   "queryPlanner" : {
4-     "namespace" : "lab2.usuarios",
5-     "indexFilterSet" : false,
6-     "parsedQuery" : {
7-       "$and" : [
8-         {
9-           "archivo" : {
10-             "$eq" : true
11-           }
12-         },
13-         {
14-           "puntos" : {
15-             "$gt" : 500
16-           }
17-         }
18-       ],
19-     },
20-     "queryHash" : "8994A4AA",
21-     "planCacheKey" : "94520262",
22-     "maxIndexedOrSolutionsReached" : false,
23-     "maxIndexedAndSolutionsReached" : false,
24-     "maxScansToExplodeReached" : false,
25-     "winningPlan" : {
26-       "stage" : "FETCH",
27-       "inputStage" : {
28-         "stage" : "IXSCAN",
29-         "keyPattern" : {
30-           "archivo" : 1,
31-           "puntos" : 1
32-         },
33-         "indexName" : "archivo_1_puntos_1",
34-         "isMultiKey" : false,
35-         "multiKeyPaths" : {
36-           "archivo" : [ ],
37-           "puntos" : [ ]
38-         },
39-         "isUnique" : false,
40-         "isSparse" : false,
41-         "isPartial" : false,
42-         "indexVersion" : 2,
43-         "direction" : "forward",
44-         "indexBounds" : {
45-           "archivo" : [ "[true, true]" ],
46-           "puntos" : [ "(500, inf.0]" ]
47-         }
48-       },
49-       "rejectedPlans" : []
50-     },
51-     "executionStats" : {
52-       ...
53-     }
54-   }
55- }

```

En primer lugar el plan utilizado fue FETCH de manera que es aquel plan que es responsable de adquirir los documentos basados en el precedente IXSCAN que es el index scan, esto implica que utilizó el índice para ir a recoger los datos, a su vez examino 47719 documentos en vez de los 100000 que había examinado la vez pasada, así mismo su tiempo de ejecución fue de 156 mili segundos. El tamaño del batch se mantiene igual y nótese que examinó la misma cantidad de documentos que de llaves, esto implica que sí o sí fue más rápido, a pesar de que en el anterior marca que este fue más lento dado que se tardó 156 ms en comparación de los 112 ms es posible que se haya tardado un poco más en este caso porque ya había ejecutado previamente el query de manera que ya estaba guardado y por ende estaba actualizado mientras que en este caso dado que fue la primera vez que se ejecutó el query con índices se pudo tardar algo más.

1.5.2 Usuarios que han comprado el producto 1 en la última semana.



```

8 db.usuarios.explain("executionStats").find({ "historial_compras": { $elemMatch: { "fecha": { $gt: ISODate("2024-01-31T10:00:00.000-06:00"), $lt: ISODate("2024-02-07T10:00:00.000-06:00") } }, "producto": { $eq: "Producto 1" } } })
Result x Result (1) x Result (2) x Result (3) x Result (4) x Result (5) x Result (6) x Result (7) x
0.019 s Query Execution Statistics of the Winning Plan
1 * {
  "explainVersion" : "1",
  "queryPlanner" : {
    "namespace" : "lab2.usuarios",
    "indexFilterSet" : false,
    "parsedQuery" : {
      "historial_compras" : {
        "$elemMatch" : {
          "$and" : [
            {
              "fecha" : {
                "$gt" : ISODate("2024-01-31T10:00:00.000-06:00"),
                "$lt" : ISODate("2024-02-07T10:00:00.000-06:00")
              }
            },
            {
              "fecha" : {
                "$gt" : ISODate("2024-01-31T10:00:00.000-06:00"),
                "$lt" : ISODate("2024-02-07T10:00:00.000-06:00")
              }
            }
          ]
        }
      }
    }
  }
  "queryHash" : "4A494214",
  "planCacheKey" : "4A494214",
  "maxIndexedOrSolutionsReached" : false,
  "maxIndexedAndSolutionsReached" : false,
  "maxScansToExplodeReached" : false,
  "winningPlan" : {
    "stage" : "FETCH",
    "filter" : {
      "historial_compras" : {
        "$elemMatch" : {
          "$and" : [
            {
              "fecha" : {
                "$lt" : ISODate("2024-02-07T10:00:00.000-06:00")
              }
            },
            {
              "fecha" : {
                "$gt" : ISODate("2024-01-31T10:00:00.000-06:00")
              }
            },
            {
              "producto" : {
                "$eq" : "Producto 1"
              }
            }
          ]
        }
      }
    }
  }
}
Line: 90, Column: 19 (15 selected) Show Log Feedback 11:30:47 AM
Free Edition 11:30 AM 2/8/2024

```

Para este caso también se utilizó el plan de FETCH utilizando las llaves creadas de las cuales únicamente se examinaron 1957 llaves e implicando que solo se examinaran 209 documentos, significando una optimización bastante alta en contraste a los 100000 documentos examinados en la primera parte, a su vez retornó 209 documentos. Y esto se pudo notar con el hecho de que el tiempo de ejecución fue de 3 ms. Siendo muy superior al visto previamente.

1.5.3 Usuarios con la etiqueta tag 2 y que tienen más de 100 visitas



```

12 | db.usuarios.explain("executionStats").find({"tags": "tag2", "visitas": {$gt: 100}});
| Result x | Result (1) x | Result (2) x | Result (3) x | Result (4) x | Result (5) x | Result (6) x | Result (7) x | Fin
| 0.119 s Query Execution Statistics of the Winning Plan
1 | {
2 |   "explainVersion" : "1",
3 |   "queryPlanner" : {
4 |     "namespace" : "lab2.usuarios",
5 |     "indexFilterSet" : false,
6 |     "parsedQuery" : {
7 |       "$and" : [
8 |         {
9 |           "tags" : {
10 |             "$eq" : "tag2"
11 |           }
12 |         },
13 |         {
14 |           "visitas" : {
15 |             "$gt" : 100
16 |           }
17 |         }
18 |       ]
19 |     },
20 |     "queryHash" : "D98E5563",
21 |     "planCacheKey" : "A4F8E3D2",
22 |     "maxIndexedOrSolutionsReached" : false,
23 |     "maxIndexedAndSolutionsReached" : false,
24 |     "maxScansToExplodeReached" : false,
25 |     "winningPlan" : {
26 |       "stage" : "FETCH",
27 |       "inputStage" : {
28 |         "stage" : "IXSCAN",
29 |         "keyPattern" : {
30 |           "tags" : 1,
31 |           "visitas" : 1
32 |         },
33 |         "indexName" : "tags_1_visitas_1",
34 |         "isMultiKey" : true,
35 |         "multiKeyPaths" : {
36 |           "tags" : [ "tags" ],
37 |           "visitas" : [ ]
38 |         },
39 |         "isUnique" : false,
40 |         "isSparse" : false,
41 |         "isPartial" : false,
42 |         "indexVersion" : 2,
43 |         "direction" : "forward",
44 |         "indexBounds" : {
45 |           "tags" : [ "\\"tag2\\", \\"tag2\\"] ],
46 |           "visitas" : [ "(100, inf.0]" ]
47 |         }
48 |       },
49 |       "rejectedPlans" : [ ]
50 |     }
51 |   }
52 |
53 | Free Edition
54 | Line: 1, Column: 2 Show Log

```

En este también se utilizó el plan de FETCH y el IXSCAN para los índices, el índice creado, esto implicó que se retornaran los 33050 documentos, y que se tardara 13 ms siendo

superior a la query sin dicho índice, a su vez se examinaron únicamente esa cantidad de índices y esas fueron las que se retornaron en contraste con las 100000 buscadas para la parte 1. El tiempo estimado fue de 5 ms, siendo más veloz que el tiempo de la parte anterior.

1.5.4 Usuarios con preferencias de color azul y que tienen entre 1000 y 2000 amigos.

```

16 db.usuarios.explain("executionStats").find({ "preferencias.color": "azul" , $expr: { $gt: [ { $size: "$amigos" } , { $const: 999 } ] } })
17 {
18     "queryPlanner": {
19         "namespace": "lab2.usuarios",
20         "indexFilterSet": false,
21         "parsedQuery": {
22             "$and": [
23                 {
24                     "preferencias.color": {
25                         "$eq": "azul"
26                     }
27                 },
28                 {
29                     "$expr": {
30                         "$gt": [
31                             {
32                                 "$size": [ "$amigos" ]
33                             },
34                             {
35                                 "$const": 999
36                             }
37                         ]
38                     }
39                 }
40             ]
41         }
42     }
43     "queryHash": "E38AE453",
44     "planCacheKey": "4E2AAD34",
45     "maxIndexedOrSolutionsReached": false,
46     "maxIndexedAndSolutionsReached": false,
47     "maxScansToExplodeReached": false,
48     "winningPlan": {
49         "stage": "FETCH",
50         "filter": {
51             "$expr": {
52                 "$gt": [
53                     {
54                         "$size": [ "$amigos" ]
55                     },
56                     {
57                         "$const": 999
58                     }
59                 ]
60             }
61         },
62         "inputStage": {
63             "stage": "IXSCAN",
64             "keyPattern": {
65                 "preferencias.color": 1,
66                 "amigos": 1
67             }
68         }
69     }
70 }

```

En este caso el índice es sustancialmente peor que en el caso anterior, utiliza el plan de FETCH y el índice sin embargo, dicho índice genera que sea más lenta la ejecución, produciendo que se tarde 9679 ms en comparación del anterior, y es que se realizó una

examinación de 8199052 llaves distintas esto debido a que se realizaba una llave por cada amigo que se tuviera, generando que fuera extremadamente más lento, a diferencia del de la parte 1 que realiza una búsqueda por cada documento, significando una búsqueda de 100000 vs una de 8199052, por ello es que en el inciso anterior es extremadamente mucho más efectivo.

- 1.6 Vuelva a ejecutar su script realizado en el inciso 1.1, insertando al menos 50,000 documentos más y vuelva a ejecutar el inciso 1.5. ¿Nota algo diferente en cuanto al rendimiento de las consultas? Explique sus observaciones.

```
C:\Users\User\Documents>node coso.js
```

Inserción de las nuevas 50000 consultas.

The screenshot shows the MongoDB Compass interface. On the left, the database structure is visible with 'localhost:27017/lab2' selected. Under 'Databases', there are 'admin', 'config', 'lab2' (which is selected), and 'local'. The 'lab2' database contains a single collection named 'usuarios'. The details for this collection are shown on the right: Storage size: 646.43 MB, Documents: 150 K, Avg. document size: 4.91 kB, Indexes: 5, and Total index size: 1.21 GB. The interface also includes a search bar, a 'Create collection' button, and a 'Refresh' button. The bottom of the screen shows a Windows taskbar with various icons and system status.

Ya se insertaron las 50000 extras

1.6.1 Usuarios activos con más de 500 puntos.

En primer lugar el plan utilizado fue FETCH de manera que es aquel plan que es responsable de adquirir los documentos basados en el precedente IXSCAN que es el index scan, esto implica que utilizó el índice para ir a recoger los datos, a su vez examino 47719 documentos en vez de los 100000 que había examinado la vez pasada, así mismo su tiempo de ejecución fue de 156 mili segundos. El tamaño del batch se mantiene igual y nótese que examinó la misma cantidad de documentos que de llaves, esto implica que sí o sí fue más rápido, a pesar de que en el anterior marca que este fue más lento dado que se tardó 156 ms en comparación de los 112 ms es posible que se haya tardado un poco más en este caso porque ya había ejecutado previamente el query de manera que ya estaba guardado y por ende estaba actualizado mientras que en este caso dado que fue la primera vez que se ejecutó el query con índices se pudo tardar algo más.

1.6.2 Usuarios que han comprado el producto 1 en la última semana.

The screenshot shows the MongoDB Query Execution Statistics interface. The top bar displays the query: `8 : ISODate("2024-01-31T16:00:00Z"), $lt: ISODate("2024-02-07T16:00:00Z") }, "producto": "Producto 1"`. Below this, there are two tabs: "Result (1)" and "Result x". The main area is titled "0.032 s Query Execution Statistics of the Winning Plan" and contains the following JSON output:

```
1: [
  2:   "explainVersion": "1",
  3:   "queryPlanner": {
  4:     "namespace": "lab2.usuarios",
  5:     "indexFilterSet": false,
  6:     "parsedQuery": {
  7:       "historial_compras": {
  8:         "$elemMatch": {
  9:           "$and": [
 10:             {
 11:               "producto": {
 12:                 "$eq": "Producto 1"
 13:               }
 14:             },
 15:             {
 16:               "fecha": {
 17:                 "$lt": ISODate("2024-02-07T10:00:00.000-06:00")
 18:               }
 19:             },
 20:             {
 21:               "fecha": {
 22:                 "$gt": ISODate("2024-01-31T10:00:00.000-06:00")
 23:               }
 24:             }
 25:           ]
 26:         }
 27:       }
 28:     },
 29:     "queryHash": "4A494214",
 30:     "planCacheKey": "4A494214",
 31:     "maxIndexedOrSolutionsReached": false,
 32:     "maxIndexedAndSolutionsReached": false,
 33:     "maxScansToExplodeReached": false,
 34:     "winningPlan": {
 35:       "stage": "FETCH",
 36:       "filter": {
 37:         "historial_compras": {
 38:           "$elemMatch": {
 39:             "$and": [
 40:               {
 41:                 "fecha": {
 42:                   "$lt": ISODate("2024-02-07T10:00:00.000-06:00")
 43:                 }
 44:               },
 45:               {
 46:                 "fecha": {
 47:                   "$gt": ISODate("2024-01-31T10:00:00.000-06:00")
 48:                 }
 49:               }
 50:             ]
 51:           }
 52:         }
 53:       }
 54:     }
 55:   }
 56: ]
```

Aquí se observó que también utilizó el plan de FETCH, utilizando el índice creado a su vez se incrementó la cantidad de usuarios que cumplen siendo 296, y el tiempo de ejecución también se incrementó a 8 milisegundos, también se examinaron únicamente 2935 llaves de índices siendo significativamente menos que vs el sin índices, aunque es peor tiempo que el anterior debido a que es más data. A su vez, el estimado de tiempo de ejecución es de 2 ms.

1.6.3 Usuarios con la etiqueta tag 2 y que tienen más de 100 visitas

```

12 db.usuarios.explain("executionStats").find({"tags": "tag2", "visitas": {$gt: 100}});
Result (1) x Result x Result (2) x
0.310 s Query Execution Statistics of the Winning Plan
1 [
2   {
3     "explainVersion" : "1",
4     "queryPlanner" : {
5       "namespace" : "lab2.usuarios",
6       "indexFilterSet" : false,
7       "parsedQuery" : {
8         "$and" : [
9           {
10             "tags" : {
11               "$eq" : "tag2"
12             }
13           },
14           {
15             "visitas" : {
16               "$gt" : 100
17             }
18           ]
19         },
20       "queryHash" : "D98E5563",
21       "planCacheKey" : "A4F8E3D2",
22       "maxIndexedOrSolutionsReached" : false,
23       "maxIndexedAndSolutionsReached" : false,
24       "maxScansToExplodeReached" : false,
25       "winningPlan" : {
26         "stage" : "FETCH",
27         "inputStage" : {
28           "stage" : "IXSCAN",
29           "keyPattern" : {
30             "tags" : 1,
31             "visitas" : 1
32           },
33           "indexName" : "tags_1_visitas_1",
34           "isMultiKey" : true,
35           "multiKeyPaths" : {
36             "tags" : [ "tags" ],
37             "visitas" : [ ]
38           },
39           "isUnique" : false,
40           "isSparse" : false,
41           "isPartial" : false,
42           "indexVersion" : 2,
43           "direction" : "forward",
44           "indexBounds" : {
45             "tags" : [ "[\"tag2\", \"tag2\"]" ],
46             "visitas" : [ "(100, inf.0]" ]
47           }
48         },
49         "rejectedPlans" : [ ]
50       }
51     }
52   ]

```

Free Edition Line: 1, Column: 2 Show L

Así mismo, aquí también se utilizó el fetch como plan, con un uso de los índices, la cantidad de retornos fue de 49454 que tiene sentido dado que se incrementaron la cantidad de documentos, a su vez también se incrementó el tiempo de ejecución a 295 ms, la cantidad de llaves es igual a la de documentos examinados y el estimado de ejecución del tiempo es de 85 ms.

1.6.4 Usuarios con preferencias de color azul y que tienen entre 1000 y 2000 amigos.

Así mismo, aquí también se utilizó el `fetch` como plan, con un uso de los índices, la cantidad de retornos fue de 49454 que tiene sentido dado que se incrementaron la cantidad de documentos, a su vez también se incrementó el tiempo de ejecución a 295 ms, la cantidad de llaves es igual a la de [documentos examinados](#) y el estimado de ejecución del tiempo es de 85 ms.

1.6.4 Usuarios con preferencias de color azul y que tienen entre 1000 y 2000 amigos.

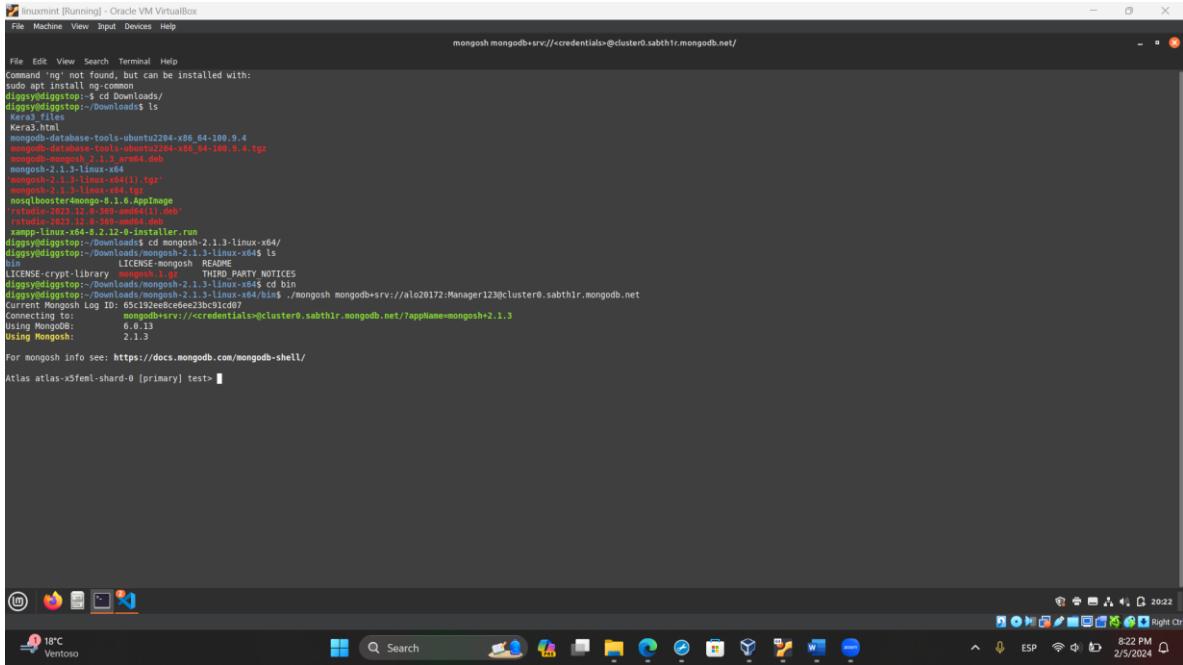
En este caso el índice es sustancialmente peor que en el caso anterior, utiliza el plan de `FETCH` y el índice sin embargo, dicho índice genera que sea más lenta la ejecución, produciendo que se tarde 9679 ms en comparación del anterior, y es que realizó una examinación de 8199052 llaves distintas esto debido a que se realizaba una llave por cada amigo que se tuviera, generando que fuera extremadamente más lento, a diferencia del de la parte 1 que realiza una búsqueda por cada documento, significando una búsqueda de 100000 vs una de 8199052, por ello es que en el inicio anterior es extremadamente mucho más efectivo.

1.6.5

En este caso el índice se mantiene siendo peor que en el caso sin índices, utiliza nuevamente el plan de `FETCH`, generando que sea más lenta la ejecución debido a que se crean más llaves distintas que cantidad de documentos, retornando más documentos siendo 58, así mismo se tardó más porque eran más documentos 13361 ms, así mismo, no se examinaron tantos documentos ya que fueron solo 24964 pero son muchas más llaves (12338016).

Ejercicio 2

2. Conexión en la mongo Shell



```

linuxmint [Running] - Oracle VM VirtualBox
File Machine View Search Terminal Help
File Edit View Search Terminal Help
Command 'ng' not found, but can be installed with:
sudo apt install ng-common
diggy@diggytop:~$ cd Downloads/
diggy@diggytop:~/Downloads$ ls
Kera3.html
mongodatabase-tools-ubuntu2204-x86_64-100.9.4
mongodatabase-tools-ubuntu2204-x86_64-100.9.4.tgz
mongosh-2.1.3-arm64.deb
mongosh-2.1.3-Linux-x64.deb
mongosh-2.1.3-Linux-x64.tgz
mongosbooster-mongo-0.1.6.AppImage
rstudio-2023.12.0-309-amd64.deb
rstudio-2023.12.0-309-amd64.tgz
xampp-linux-x64-8.2.12-0-installer.run
diggy@diggytop:~/Downloads$ cd mongosh-2.1.3-Linux-x64/
diggy@diggytop:~/Downloads/mongosh-2.1.3-Linux-x64$ ls
bin LICENSE-mongosh README
LICENSE-crypt-library mongosh.lgr THIRDPARTY_NOTICES
diggy@diggytop:~/Downloads$ ./mongosh.lgr
Current Mongosh Log ID: 65c192ee8ce66e23bc91cd07
Connecting to: mongodbsrv://<credentials>@cluster0.sabthir.mongodb.net/?appName=mongosh+2.1.3
Using MongoDB: 0.0.13
Using Mongosh: 2.1.3

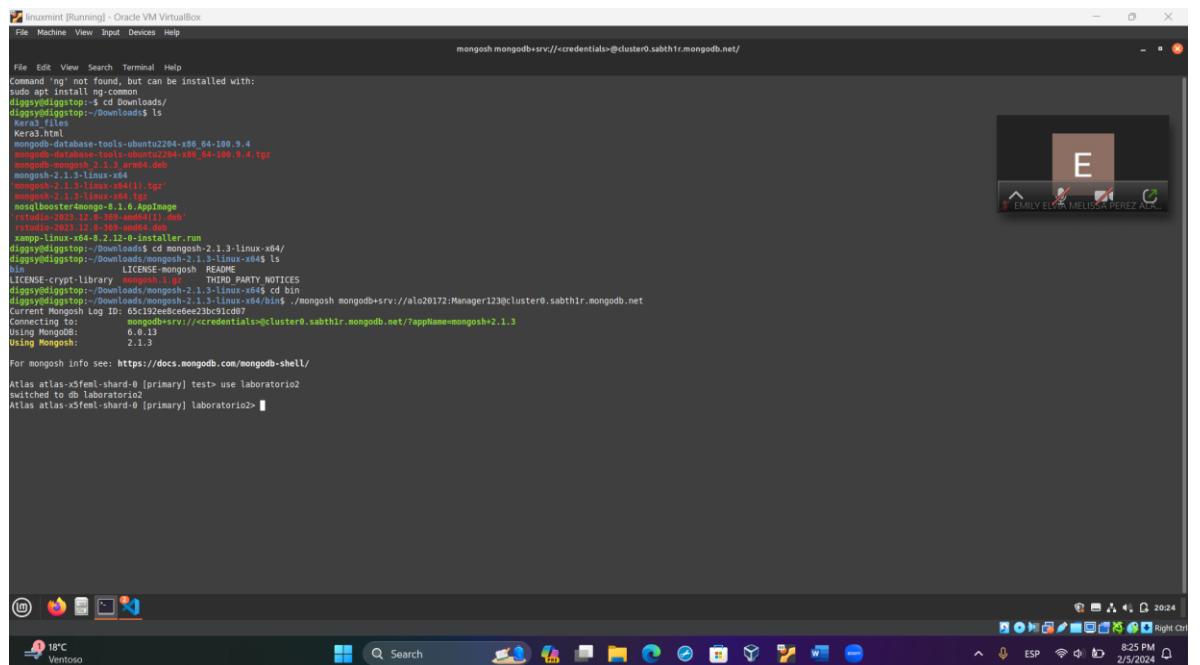
For mongosh info see: https://docs.mongodb.com/mongodb-shell/
Atlas atlas-x5fem1-shard-0 [primary] test>

```

Se logró conectar a la mongo shell

2.1. Crear la base de datos laboratorio 2

2.1.1.

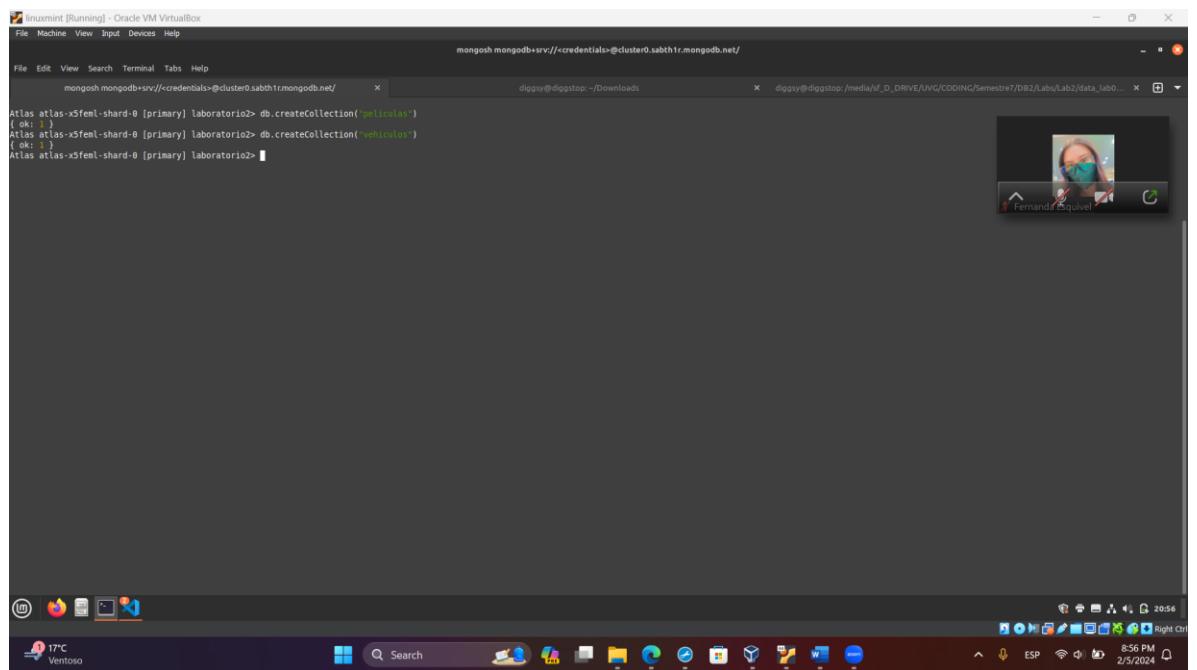


```

linuxmint [Running] - Oracle VM VirtualBox
File Machine View Search Terminal Help
File Edit View Search Terminal Help
Command 'ng' not found, but can be installed with:
sudo apt install ng-common
diggy@diggytop:~$ cd Downloads/
diggy@diggytop:~/Downloads$ ls
Kera3.html
mongodatabase-tools-ubuntu2204-x86_64-100.9.4
mongodatabase-tools-ubuntu2204-x86_64-100.9.4.tgz
mongosh-2.1.3-arm64.deb
mongosh-2.1.3-Linux-x64.deb
mongosh-2.1.3-Linux-x64.tgz
mongosbooster-mongo-0.1.6.AppImage
rstudio-2023.12.0-309-amd64.deb
rstudio-2023.12.0-309-amd64.tgz
xampp-linux-x64-8.2.12-0-installer.run
diggy@diggytop:~/Downloads$ cd mongosh-2.1.3-Linux-x64/
diggy@diggytop:~/Downloads/mongosh-2.1.3-Linux-x64$ ls
bin LICENSE-mongosh README
LICENSE-crypt-library mongosh.lgr THIRDPARTY_NOTICES
diggy@diggytop:~/Downloads$ ./mongosh.lgr
Current Mongosh Log ID: 65c192ee8ce66e23bc91cd07
Connecting to: mongodbsrv://<credentials>@cluster0.sabthir.mongodb.net/?appName=mongosh+2.1.3
Using MongoDB: 0.0.13
Using Mongosh: 2.1.3

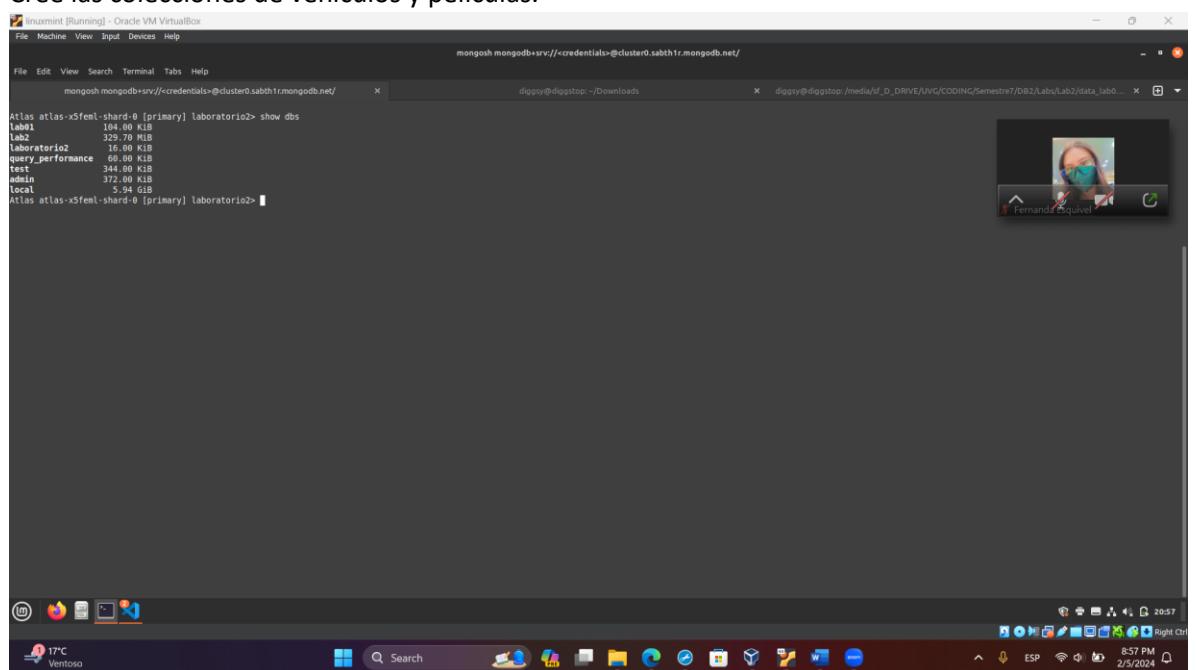
For mongosh info see: https://docs.mongodb.com/mongodb-shell/
Atlas atlas-x5fem1-shard-0 [primary] test> use laboratorio2
switched to db laboratorio2
Atlas atlas-x5fem1-shard-0 [primary] laboratorio2>

```



```
mongosh mongodb+srv://<credentials>@cluster0.sabth1r.mongodb.net/
File Edit View Search Terminal Tabs Help
mongosh mongodb+srv://<credentials>@cluster0.sabth1r.mongodb.net/
Atlas atlas-x5fem-shard-0 [primary] laboratorio2> db.createCollection("vehiculos")
{
  "ok": 1
}
Atlas atlas-x5fem-shard-0 [primary] laboratorio2> db.createCollection("peliculas")
{
  "ok": 1
}
Atlas atlas-x5fem-shard-0 [primary] laboratorio2>
```

Cree las colecciones de vehículos y películas.



```
mongosh mongodb+srv://<credentials>@cluster0.sabth1r.mongodb.net/
File Edit View Search Terminal Tabs Help
mongosh mongodb+srv://<credentials>@cluster0.sabth1r.mongodb.net/
Atlas atlas-x5fem-shard-0 [primary] laboratorio2> show dbs
label1          104.00 kB
lab2           329.70 kB
laboratorio2      11.00 kB
query_performance   60.00 kB
test            344.00 kB
admin            372.00 kB
local            5.94 GB
Atlas atlas-x5fem-shard-0 [primary] laboratorio2>
```

Ya aparece mi base de datos creada.

Y también aparece en el clúster.

Añadir vehículos a la base de datos

Daba error porque el file tenía problemas de manera que existía una coma al final por lo que se lo tuve que quitar y ya me importó.

```
diggy@digstop:~/Downloads/mongodb-database-tools-ubuntu2204-x86_64-100.9.4/bin$ mongorestore -d laboratorio2 --collection vehiculos --file /media/sf_D_DRIVE/UVG/CODING/Sebastián/semestre7/D02/Labs/Lab2/data_lab0/vehiculos.json
2024-02-05T21:03:04.225-0600 continuing through error: E11000 duplicate key error collection: laboratorio2.vehiculos index: id dup key: { id: 979 }
2024-02-05T21:03:04.225-0600 continuing through error: E11000 duplicate key error collection: laboratorio2.vehiculos index: id dup key: { id: 958 }
2024-02-05T21:03:04.225-0600 continuing through error: E11000 duplicate key error collection: laboratorio2.vehiculos index: id dup key: { id: 911 }
2024-02-05T21:03:04.225-0600 continuing through error: E11000 duplicate key error collection: laboratorio2.vehiculos index: id dup key: { id: 962 }
2024-02-05T21:03:04.225-0600 continuing through error: E11000 duplicate key error collection: laboratorio2.vehiculos index: id dup key: { id: 974 }
2024-02-05T21:03:04.225-0600 continuing through error: E11000 duplicate key error collection: laboratorio2.vehiculos index: id dup key: { id: 975 }
2024-02-05T21:03:04.225-0600 continuing through error: E11000 duplicate key error collection: laboratorio2.vehiculos index: id dup key: { id: 959 }
2024-02-05T21:03:04.225-0600 continuing through error: E11000 duplicate key error collection: laboratorio2.vehiculos index: id dup key: { id: 988 }
2024-02-05T21:03:04.225-0600 continuing through error: E11000 duplicate key error collection: laboratorio2.vehiculos index: id dup key: { id: 979 }
2024-02-05T21:03:04.225-0600 continuing through error: E11000 duplicate key error collection: laboratorio2.vehiculos index: id dup key: { id: 998 }
2024-02-05T21:03:04.225-0600 continuing through error: E11000 duplicate key error collection: laboratorio2.vehiculos index: id dup key: { id: 991 }
2024-02-05T21:03:04.225-0600 continuing through error: E11000 duplicate key error collection: laboratorio2.vehiculos index: id dup key: { id: 992 }
2024-02-05T21:03:04.225-0600 continuing through error: E11000 duplicate key error collection: laboratorio2.vehiculos index: id dup key: { id: 973 }
2024-02-05T21:03:04.225-0600 continuing through error: E11000 duplicate key error collection: laboratorio2.vehiculos index: id dup key: { id: 993 }
2024-02-05T21:03:04.225-0600 continuing through error: E11000 duplicate key error collection: laboratorio2.vehiculos index: id dup key: { id: 995 }
2024-02-05T21:03:04.225-0600 continuing through error: E11000 duplicate key error collection: laboratorio2.vehiculos index: id dup key: { id: 954 }
2024-02-05T21:03:04.225-0600 continuing through error: E11000 duplicate key error collection: laboratorio2.vehiculos index: id dup key: { id: 989 }
2024-02-05T21:03:04.225-0600 continuing through error: E11000 duplicate key error collection: laboratorio2.vehiculos index: id dup key: { id: 998 }
2024-02-05T21:03:04.225-0600 continuing through error: E11000 duplicate key error collection: laboratorio2.vehiculos index: id dup key: { id: 999 }
2024-02-05T21:03:04.225-0600 continuing through error: E11000 duplicate key error collection: laboratorio2.vehiculos index: id dup key: { id: 1000 }
2024-02-05T21:03:04.225-0600 continuing through error: E11000 duplicate key error collection: laboratorio2.vehiculos index: id dup key: { id: 972 }
2024-02-05T21:03:04.225-0600 continuing through error: E11000 duplicate key error collection: laboratorio2.vehiculos index: id dup key: { id: 980 }
2024-02-05T21:03:04.225-0600 continuing through error: E11000 duplicate key error collection: laboratorio2.vehiculos index: id dup key: { id: 981 }
2024-02-05T21:03:04.225-0600 continuing through error: E11000 duplicate key error collection: laboratorio2.vehiculos index: id dup key: { id: 982 }
2024-02-05T21:03:04.225-0600 continuing through error: E11000 duplicate key error collection: laboratorio2.vehiculos index: id dup key: { id: 984 }
2024-02-05T21:03:04.225-0600 continuing through error: E11000 duplicate key error collection: laboratorio2.vehiculos index: id dup key: { id: 986 }
2024-02-05T21:03:04.225-0600 continuing through error: E11000 duplicate key error collection: laboratorio2.vehiculos index: id dup key: { id: 997 }
2024-02-05T21:03:04.225-0600 ##### document(s) imported successfully. 0 document(s) failed to import.
2024-02-05T21:03:04.225-0600 #####
diggy@digstop:~/Downloads/mongodb-database-tools-ubuntu2204-x86_64-100.9.4/bin$ mongorestore -d laboratorio2 --collection vehiculos --file /media/sf_D_DRIVE/UVG/CODING/Sebastián/semestre7/D02/Labs/Lab2/data_lab0/vehiculos.json
2024-02-05T21:06:38.430-0600 connected to: mongodb+srv://cluster0.sabthir.mongodb.net/laboratorio2
2024-02-05T21:06:39.405-0600 Failed: error processing document #0#0: invalid character '!' looking for beginning of value
2024-02-05T21:06:39.427-0600 0 document(s) imported successfully. 0 document(s) failed to import.
diggy@digstop:~/Downloads/mongodb-database-tools-ubuntu2204-x86_64-100.9.4/bin$ mongorestore -d laboratorio2 --collection vehiculos --file /media/sf_D_DRIVE/UVG/CODING/Sebastián/semestre7/D02/Labs/Lab2/data_lab0/vehiculos.json
2024-02-05T21:06:39.430-0600 connected to: mongodb+srv://cluster0.sabthir.mongodb.net/laboratorio2
2024-02-05T21:06:39.430-0600 Failed: error processing document #0#0: invalid character '!' looking for beginning of value
2024-02-05T21:06:39.427-0600 0 document(s) imported successfully. 0 document(s) failed to import.
diggy@digstop:~/Downloads/mongodb-database-tools-ubuntu2204-x86_64-100.9.4/bin$ mongorestore -d laboratorio2 --collection vehiculos --file /media/sf_D_DRIVE/UVG/CODING/Sebastián/semestre7/D02/Labs/Lab2/data_lab0/vehiculos.json
2024-02-05T21:06:39.430-0600 connected to: mongodb+srv://cluster0.sabthir.mongodb.net/laboratorio2
2024-02-05T21:06:39.430-0600 #####
2024-02-05T21:06:39.430-0600 1808 document(s) imported successfully. 0 document(s) failed to import.
diggy@digstop:~/Downloads/mongodb-database-tools-ubuntu2204-x86_64-100.9.4/bin$
```

Se logró observar que sí se insertaron porque:

The screenshot shows a Linux desktop environment with several windows open:

- Terminal Window:** Shows the command `mongosh mongodb+srv://<credentials>@cluster0.sabth1r.mongodb.net/` and the output of a MongoDB query. The query retrieves documents from the `vehiculos` collection, filtering by `Año-del-Cache` greater than or equal to 1990. The output includes fields like `_id`, `Nombre-de-la-pelicula`, `Año-de-la-pelicula`, `Marca-del-Cache`, `Modelo-del-Cache`, and `Precio-del-Cache`.
- Web Browser:** Displays the MongoDB Atlas interface for the `laboratorio2.vehiculos` collection. The left sidebar shows the project structure and deployment details. The main panel shows the collection overview, storage metrics (STORAGE SIZE: 80KB, LOGICAL DATA SIZE: 146.77KB), and document counts (TOTAL DOCUMENTS: 1000, INDEXES TOTAL SIZE: 64KB). A search bar and a "Find" button are visible. Below the search bar, a "QUERY RESULTS: 1-20 OF MANY" section displays the first few documents from the collection.

2.1.2. Añadir películas a la base de datos

```
linuzem [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
diggy@digstop: ~/Downloads/mongodb-database-tools-ubuntu2204-x86_64-100.9.4/bin
diggy@digstop: ~/Downloads/mongodb-database-tools-ubuntu2204-x86_64-100.9.4/bin$ mongorestore --uri "mongodb://alo20172:Manager123@cluster0.sabthir.mongodb.net/laboratorio2" --collection peliculas --file /media/sf_D_DRIVE/UVG/COING/Semestre2023/Labs/Abd/data/lab02/peliculas.json
mongorestore command not found

diggy@digstop: ~/Downloads/mongodb-database-tools-ubuntu2204-x86_64-100.9.4/bin$ ls
bsounding mongomock mongoprotocol mongoprotosec mongostat mongotop
diggy@digstop: ~/Downloads/mongodb-database-tools-ubuntu2204-x86_64-100.9.4/bin$ mongoimport --uri "mongodb://alo20172:Manager123@cluster0.sabthir.mongodb.net/laboratorio2" --collection peliculas --file /media/sf_D_DRIVE/UVG/COING/Semestre2023/Labs/Abd/data/lab02/peliculas.json
2024-02-05T21:02:58.944+0000 connected to: mongodb://***REDACTED***@cluster0.sabthir.mongodb.net/laboratorio2
2024-02-05T21:03:03.846+0000 continuing through error: E10000 duplicate key error collection: laboratorio2.peliculas index: _id dup key: { _id: 1 }
2024-02-05T21:03:03.846+0000 continuing through error: E10000 duplicate key error collection: laboratorio2.peliculas index: _id dup key: { _id: 9 }
2024-02-05T21:03:03.846+0000 continuing through error: E10000 duplicate key error collection: laboratorio2.peliculas index: _id dup key: { _id: 10 }
2024-02-05T21:03:03.846+0000 continuing through error: E10000 duplicate key error collection: laboratorio2.peliculas index: _id dup key: { _id: 11 }
2024-02-05T21:03:03.846+0000 continuing through error: E10000 duplicate key error collection: laboratorio2.peliculas index: _id dup key: { _id: 12 }
2024-02-05T21:03:03.846+0000 continuing through error: E10000 duplicate key error collection: laboratorio2.peliculas index: _id dup key: { _id: 13 }
2024-02-05T21:03:03.846+0000 continuing through error: E10000 duplicate key error collection: laboratorio2.peliculas index: _id dup key: { _id: 14 }
2024-02-05T21:03:03.846+0000 continuing through error: E10000 duplicate key error collection: laboratorio2.peliculas index: _id dup key: { _id: 15 }
2024-02-05T21:03:03.846+0000 continuing through error: E10000 duplicate key error collection: laboratorio2.peliculas index: _id dup key: { _id: 16 }
2024-02-05T21:03:03.846+0000 continuing through error: E10000 duplicate key error collection: laboratorio2.peliculas index: _id dup key: { _id: 17 }
2024-02-05T21:03:03.846+0000 continuing through error: E10000 duplicate key error collection: laboratorio2.peliculas index: _id dup key: { _id: 18 }
2024-02-05T21:03:03.846+0000 continuing through error: E10000 duplicate key error collection: laboratorio2.peliculas index: _id dup key: { _id: 19 }
2024-02-05T21:03:03.846+0000 continuing through error: E10000 duplicate key error collection: laboratorio2.peliculas index: _id dup key: { _id: 20 }
2024-02-05T21:03:03.846+0000 continuing through error: E10000 duplicate key error collection: laboratorio2.peliculas index: _id dup key: { _id: 21 }
2024-02-05T21:03:03.846+0000 continuing through error: E10000 duplicate key error collection: laboratorio2.peliculas index: _id dup key: { _id: 22 }
2024-02-05T21:03:03.846+0000 continuing through error: E10000 duplicate key error collection: laboratorio2.peliculas index: _id dup key: { _id: 23 }
2024-02-05T21:03:03.846+0000 continuing through error: E10000 duplicate key error collection: laboratorio2.peliculas index: _id dup key: { _id: 6 }
2024-02-05T21:03:03.846+0000 continuing through error: E10000 duplicate key error collection: laboratorio2.peliculas index: _id dup key: { _id: 3 }
2024-02-05T21:03:03.846+0000 continuing through error: E10000 duplicate key error collection: laboratorio2.peliculas index: _id dup key: { _id: 4 }
2024-02-05T21:03:03.846+0000 continuing through error: E10000 duplicate key error collection: laboratorio2.peliculas index: _id dup key: { _id: 5 }
2024-02-05T21:03:03.846+0000 continuing through error: E10000 duplicate key error collection: laboratorio2.peliculas index: _id dup key: { _id: 19 }
2024-02-05T21:03:03.846+0000 continuing through error: E10000 duplicate key error collection: laboratorio2.peliculas index: _id dup key: { _id: 24 }
2024-02-05T21:03:03.846+0000 continuing through error: E10000 duplicate key error collection: laboratorio2.peliculas index: _id dup key: { _id: 8 }
2024-02-05T21:03:03.846+0000 continuing through error: E10000 duplicate key error collection: laboratorio2.peliculas index: _id dup key: { _id: 25 }
2024-02-05T21:03:03.846+0000 continuing through error: E10000 duplicate key error collection: laboratorio2.peliculas index: _id dup key: { _id: 26 }
2024-02-05T21:03:03.846+0000 continuing through error: E10000 duplicate key error collection: laboratorio2.peliculas index: _id dup key: { _id: 27 }
2024-02-05T21:03:03.846+0000 continuing through error: E10000 duplicate key error collection: laboratorio2.peliculas index: _id dup key: { _id: 28 }
2024-02-05T21:03:03.846+0000 continuing through error: E10000 duplicate key error collection: laboratorio2.peliculas index: _id dup key: { _id: 29 }
2024-02-05T21:03:03.846+0000 continuing through error: E10000 duplicate key error collection: laboratorio2.peliculas index: _id dup key: { _id: 30 }
2024-02-05T21:03:03.846+0000 continuing through error: E10000 duplicate key error collection: laboratorio2.peliculas index: _id dup key: { _id: 31 }
2024-02-05T21:03:03.846+0000 continuing through error: E10000 duplicate key error collection: laboratorio2.peliculas index: _id dup key: { _id: 32 }
2024-02-05T21:03:03.846+0000 continuing through error: E10000 duplicate key error collection: laboratorio2.peliculas index: _id dup key: { _id: 33 }
2024-02-05T21:03:03.846+0000 continuing through error: E10000 duplicate key error collection: laboratorio2.peliculas index: _id dup key: { _id: 41 }
2024-02-05T21:03:03.846+0000 continuing through error: E10000 duplicate key error collection: laboratorio2.peliculas index: _id dup key: { _id: 42 }
2024-02-05T21:03:03.846+0000 continuing through error: E10000 duplicate key error collection: laboratorio2.peliculas index: _id dup key: { _id: 43 }
2024-02-05T21:03:03.846+0000 continuing through error: E10000 duplicate key error collection: laboratorio2.peliculas index: _id dup key: { _id: 44 }
2024-02-05T21:03:03.846+0000 continuing through error: E10000 duplicate key error collection: laboratorio2.peliculas index: _id dup key: { _id: 45 }




E  
EMILY ELVIA MELISSA PÉREZ AGUILAR


```

Si se lograron añadir algunas películas, sin embargo, dio problema y por ello solo se lograron insertar algunas películas ya que habían índices duplicados.

```
linuxum (Running) - Oracle VM VirtualBox
File Machine View Input Devices Help
File Edit View Search Terminal Tabs Help
mongosh mongodb+srv://<credentials>@cluster0.sabthir.mongodb.net/
diggy@digstop:~/Downloads/mongodb-database-tools-ubuntu2004x86_64-100.9.4/bin$ ./mongosh mongodb+srv://alo20172:Manager123@cluster0.sabthir.mongodb.net/test
Connecting To mongodb+srv://<credentials>@cluster0.sabthir.mongodb.net/?appName=mongosh+2.1.3
Using Mongos:
Using Mongod:
2.1.3

For mongosh info see: https://docs.mongodb.com/mongosh-shell/

Atlas atlas->x5feml-shard-0 [primary] test> use laboratorio2
switched to db laboratorio2
Atlas atlas->x5feml-shard-0 [primary] laboratorio2> db.peliculas.find()

Atlas atlas->x5feml-shard-0 [primary] laboratorio2> mongodump --uri "mongodb+srv://alo20172:Manager123@cluster0.sabthir.mongodb.net/laboratorio2" --out dump
Uncought:
SyntaxError: Missing semicolon. (1:12)
Atlas atlas->x5feml-shard-0 [primary] laboratorio2> db.createCollection("peliculas")
{ ok: 1 }
Atlas atlas->x5feml-shard-0 [primary] laboratorio2> db.createCollection("vehiculos")
Atlas atlas->x5feml-shard-0 [primary] laboratorio2> db.peliculas.find()
{
  id: 8,
  "Nombre-de-la-pelicula": "Solo",
  "Año-de-la-pelicula": 1998,
  "Genero-de-la-pelicula": "Action|Sci-Fi|Thriller",
  "Precio-de-la-entrada-de-la-pelicula": "$125.92"
},
{
  id: 9,
  "Nombre-de-la-pelicula": "Silent Night, Deadly Night 3: The Toy Maker",
  "Año-de-la-pelicula": 1992,
  "Genero-de-la-pelicula": "Horror|Sci-Fi",
  "Precio-de-la-entrada-de-la-pelicula": "$130.98"
},
{
  id: 10,
  "Nombre-de-la-pelicula": "Monster X Strikes Back: Attack the 68 Summit, The (Girare no gyakushū: Tōya-ko Sannito kikippatsu)",
  "Año-de-la-pelicula": 1989,
  "Genero-de-la-pelicula": "Comedy|Sci-Fi",
  "Precio-de-la-entrada-de-la-pelicula": "$119.02"
},
{
  id: 11,
  "Nombre-de-la-pelicula": "My Little Business (Ma petite entreprise)",
  "Año-de-la-pelicula": 2000,
  "Genero-de-la-pelicula": "Comedy|Romantic|Drama|Thriller"
}
EMILY ELVIA MELISSA PÉREZ ALA...
21:44
9:04 PM
2/25/2024
Right Ctrl
17°C Ventoso
Search
Windows Taskbar
```

Aquí se observan las películas que sí se lograron insertar en la colección de películas.

The screenshot shows the MongoDB Atlas interface. On the left, the sidebar shows 'Project 0' with 'DEPLOYMENT' selected, displaying 'lab01', 'lab2', and 'laboratorio2' databases. Under 'laboratorio2', the 'peliculas' collection is selected, showing document counts for 'vehiculos', 'query_performance', and 'test'. The main panel displays the 'laboratorio2.peliculas' collection with 1000 documents. A specific document is highlighted with the following data:

```

_id: 8
Nombre-de-la-pelicula: "Solo"
Año-de-la-pelicula: 1998
Genero-de-la-pelicula: "Action|Sci-Fi|Thriller"
Precio-de-la-entrada-de-la-pelicula: "$125.92"

```

2.1.3. Consultar la información de todas las colecciones (db.getCollectionInfos()).

```

mongosh mongoDB+srv://<credentials>@cluster0.sabch1r.mongodb.net/
File Machine View Input Devices Help
mongosh mongoDB+srv://<credentials>@cluster0.sabch1r.mongodb.net/  mongos@diggstop: ~ /Downloads/mongodb-database-tools/ubuntu2204x86_64-100.9.4/bin/  digpsy@digstop:/media/sf_D_DRIVE/IVG/CODING/Cementre2/D82/als0ah2/data_lab0...
File Edit View Search Terminal Tabs Help
mongosh mongoDB+srv://<credentials>@cluster0.sabch1r.mongodb.net/
[1] :0> db.getCollectionInfos()
{
  "name": "peliculas",
  "type": "collection",
  "options": {},
  "info": {
    "readOnly": false,
    "uuid": UUID("19c9e48-d533-468e-b605-43f35da8bfcd")
  },
  "idIndex": { v: 2, key: { _id: 1 }, name: "_id" }
},
{
  "name": "vehiculos",
  "type": "collection",
  "options": {},
  "info": {
    "readOnly": false,
    "uuid": UUID("a9abdf4b-ccb2-4a1e-a2c2-cb65a9f23b68")
  },
  "idIndex": { v: 2, key: { _id: 1 }, name: "_id" }
}
Atlas atlas-x5fem1-shard-0 [primary] laboratorio2>

```

2.2. Transformando los resultados de la colección Vehículos

2.2.1. Actualizar los nombres de las columnas de vehículos

```

Invenment [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
mongosh mongodb+srv://<credentials>@cluster0.sabth1r.mongodb.net/
File Edit View Search Terminal Help
88537, 89428, 83392, 63169, 29883, 50499, 69777, 7933, 53295, 3095,
13773, 40811, 77229, 59810, 61513, 80941, 52231, 41555,
4812, 75758, 3179, 74318, 46818, 62012, 58982, 75102, 9410, 80245,
34986, 9835, 78160, 84691, 9614, 37516, 12393, 96613, 81634, 62988,
7996, 40849, 55610, 84537, 30927, 30397, 72729, 44795, 17668, 26707,
37224, 44449, 55611, 84538, 30928, 30398, 72730, 44796, 17669, 26708,
48068, 83825, 78279, 78114, 83223, 248211, 20121, 2386, 67076, 64417,
85121, 44084, 34740, 24059, 91105, 63537, 77260, 76711, 42378, 2216,
39145, 27386, 88623, 12789, 5852, 31948, 13993, 75191, 82785, 68818,
9884, 60582, 19730, 58343, 891, 28599, 61983, 72744, 46164, 94681,
4681, 96582, 19730, 58343, 891, 28599, 61983, 72744, 46164, 94681,
54598, 64493, 22933, 67767, 20248, 61609, 70583, 40652, 53681, 67106
],
preferencias: [
  { color: "azul", idioma: null, tema: "tema1" },
  { color: "naranja", idioma: null, tema: "tema1" }
]
}
Type 'exit' for more
Atlas atlas-x5feml-shard-0 [primary] lab> db.usuarios.find({ { "preferencias": { $elemMatch: { "color": "azul" } } }, { $expr: { $gt: [{ $size: "$preferencias", 999 } ] } } }).count()
0
Atlas atlas-x5feml-shard-0 [primary] lab> db.usuarios.find({ { "preferencias": { $elemMatch: { "color": "azul" } } }, { $expr: { $gt: [{ $size: "$preferencias", 999 } ] } } }).count()
0
Atlas atlas-x5feml-shard-0 [primary] lab> db.usuarios.find({ "preferencias": { $elemMatch: { "color": "azul" } } }, { $expr: { $gt: [{ $size: "$preferencias", 999 } ] } }).count()
0
Atlas atlas-x5feml-shard-0 [primary] lab> use laboratorio2
switched to db laboratorio2
Atlas atlas-x5feml-shard-0 [primary] laboratorio2> 2.2.1. db.vehiculos.updateMany({$rename:{'id-del-Coché':'id'}},{$rename:{'Marca-del-Coché':'brand'}},{$rename:{'Modelo-del-Coché':'model'}},{$rename:{'Año-del-Coché':'year'}},{$ren
ame:{'Precio-del-Coché':'price'}})
Atlas atlas-x5feml-shard-0 [primary] laboratorio2> db.vehiculos.updateMany(),{$rename:{'id-del-Coché': '_id', 'Marca-del-Coché': 'brand', 'Modelo-del-Coché': 'model', 'Año-del-Coché': 'year', 'Precio-del-Coché': 'price'}})
MongoServerError: you are over your space quota, using 513 MB of 512 MB
Atlas atlas-x5feml-shard-0 [primary] laboratorio2> db.vehiculos.updateMany(),{$rename:{'id-del-Coché': '_id', 'Marca-del-Coché': 'brand', 'Modelo-del-Coché': 'model', 'Año-del-Coché': 'year', 'Precio-del-Coché': 'price'}})
MongoServerError: Performing an update on the path '$_id' would modify the immutable field '_id'
Atlas atlas-x5feml-shard-0 [primary] laboratorio2> db.vehiculos.updateMany(),{$rename:{'id-del-Coché': 'id', 'Marca-del-Coché': 'brand', 'Modelo-del-Coché': 'model', 'Año-del-Coché': 'year', 'Precio-del-Coché': 'price'}})
Atlas atlas-x5feml-shard-0 [primary] laboratorio2>
{
  acknowledged: true,
  insertedId: null,
  insertedCount: 0,
  modifiedCount: 18000,
  upsertedCount: 0
}
Atlas atlas-x5feml-shard-0 [primary] laboratorio2>

```

2.2.2. Calcular la cantidad de modelos y precio promedio por año por marca. Apóyese de la instrucción aggregate. Exporte su resultado a un JSONArray llamado vehiculos_brand_stats.json.

```

Atlas atlas-x5feml-shard-0 [primary] laboratorio2> db.vehiculos.aggregate([
  {
    $addFields: { priceD: {$toDouble: {$substrBytes: ["$price", 1, -1]} } },
    { $group: {
      _id: { brand: "$brand", year: "$year" },
      amountModel: { $sum: 1 },
      avgPrice: { $avg: "$priceD" }
    }},
    { $out: "vehiclesStats" }
  }
])

Atlas atlas-x5feml-shard-0 [primary] laboratorio2> show collections
peliculas
top20in90s
vehiclesStats
vehiculos

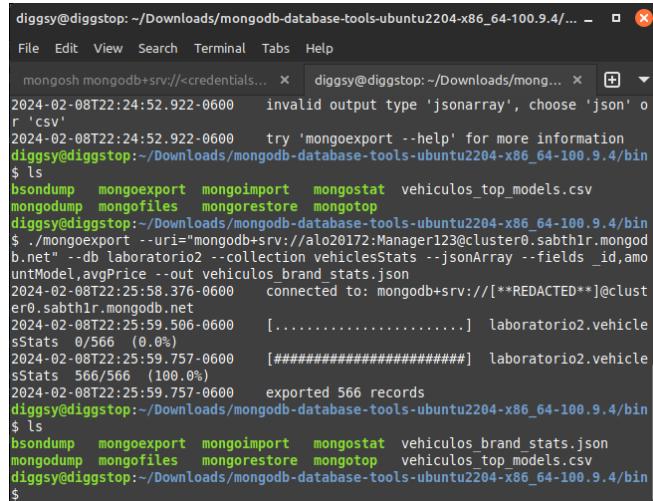
```

Se genero la colección para generar el json

```

diggsy@diggstop: ~/Downloads/mongodb-database-tools-ubuntu2204-x86_64-100.9.4/bin$ ./mongoexport --uri="mongodb+srv://al020172:Manager123@cluster0.sabth1r.mongodb.net" --db laboratorio2 --collection vehiclesStats --type=jsonArray --fields _id,amountModel,avgPrice --out vehiculos_brand_stats.json
2024-02-08T22:24:52.922-0600      invalid output type 'jsonarray', choose 'json' or 'csv'
2024-02-08T22:24:52.922-0600      try "mongoexport --help" for more information
diggsy@diggstop: ~/Downloads/mongodb-database-tools-ubuntu2204-x86_64-100.9.4/bin$ ls
bsodump  mongoexport  mongoimport  mongostat  vehiculos_top_models.csv
mongodump  mongofiles  mongorestore  mongotop
diggsy@diggstop: ~/Downloads/mongodb-database-tools-ubuntu2204-x86_64-100.9.4/bin$ ./mongoexport --uri="mongodb+srv://al020172:Manager123@cluster0.sabth1r.mongodb.net" --db laboratorio2 --collection vehiclesStats --jsonArray --fields _id,amountModel,avgPrice --out vehiculos_brand_stats.json
2024-02-08T22:25:58.376-0600      connected to: mongodb+srv://[**REDACTED**]@cluster0.sabth1r.mongodb.net
2024-02-08T22:25:59.506-0600      [.....] laboratorio2.vehicle
ssStats 0/566 (0.0%)
2024-02-08T22:25:59.757-0600      [#####] laboratorio2.vehicle
ssStats 566/566 (100.0%)
2024-02-08T22:25:59.757-0600      exported 566 records
diggsy@diggstop: ~/Downloads/mongodb-database-tools-ubuntu2204-x86_64-100.9.4/bin$

```



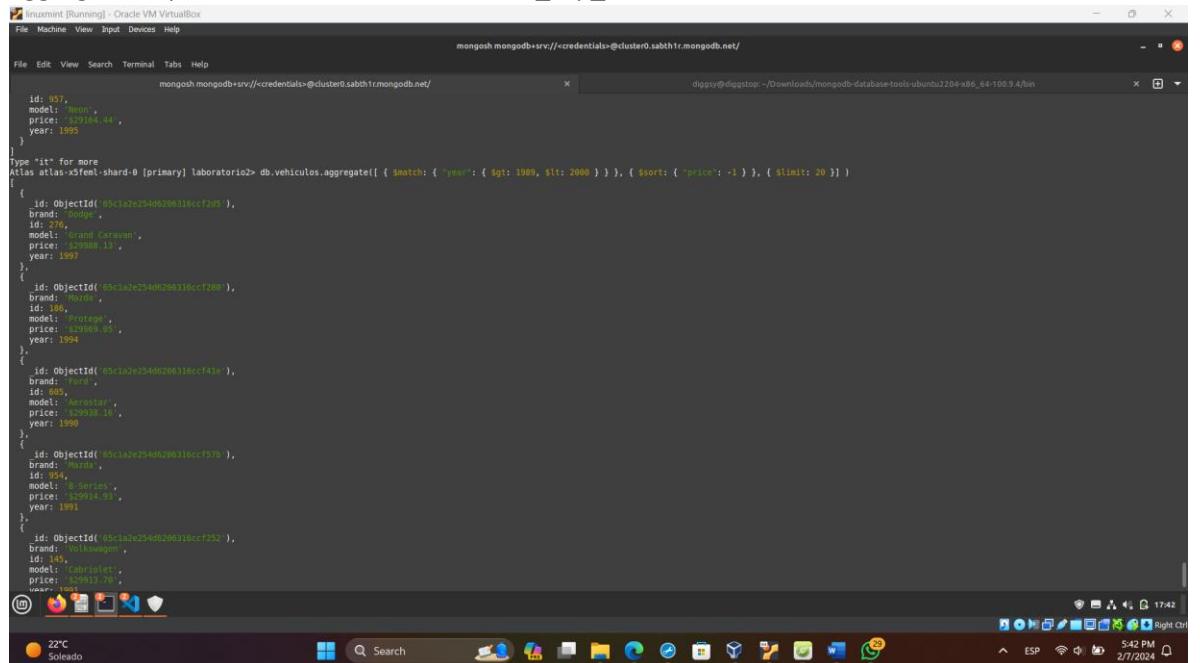
```

diggsy@diggstop:~/Downloads/mongodb-database-tools-ubuntu2204-x86_64-100.9.4/bin$ mongoexport mongoimport mongostat vehiculos_top_models.csv
mongodump mongofiles mongorestore mongotop
diggsy@diggstop:~/Downloads/mongodb-database-tools-ubuntu2204-x86_64-100.9.4/bin$ ./mongoexport --uri="mongodb+srv://alo20172:Manager123@cluster0.sabth1r.mongodb.net" --db laboratorio2 --collection vehiclesStats --jsonArray --fields _id,amounModel,avgPrice --out vehiculos_brand_stats.json
2024-02-08T22:25:58.376-0600      connected to: mongodb+srv://[*REDACTED*]@cluster0.sabth1r.mongodb.net
2024-02-08T22:25:59.506-0600      [.....] laboratorio2.vehicle
sStats 0/566 (0.0%)
2024-02-08T22:25:59.757-0600      [#####] laboratorio2.vehicle
sStats 566/566 (100 %)
2024-02-08T22:25:59.757-0600      exported 566 records
diggsy@diggstop:~/Downloads/mongodb-database-tools-ubuntu2204-x86_64-100.9.4/bin$ ls
bsondump mongoexport mongoimport mongostat vehiculos_top_models.csv
diggsy@diggstop:~/Downloads/mongodb-database-tools-ubuntu2204-x86_64-100.9.4/bin$ 

```

De manera que se generó el json.

2.2.3. Buscar los 20 vehículos más caros de la década de los 90s. Apóyese de la instrucción aggregate. Exportar a CSV llamado vehiculos_top_models.csv

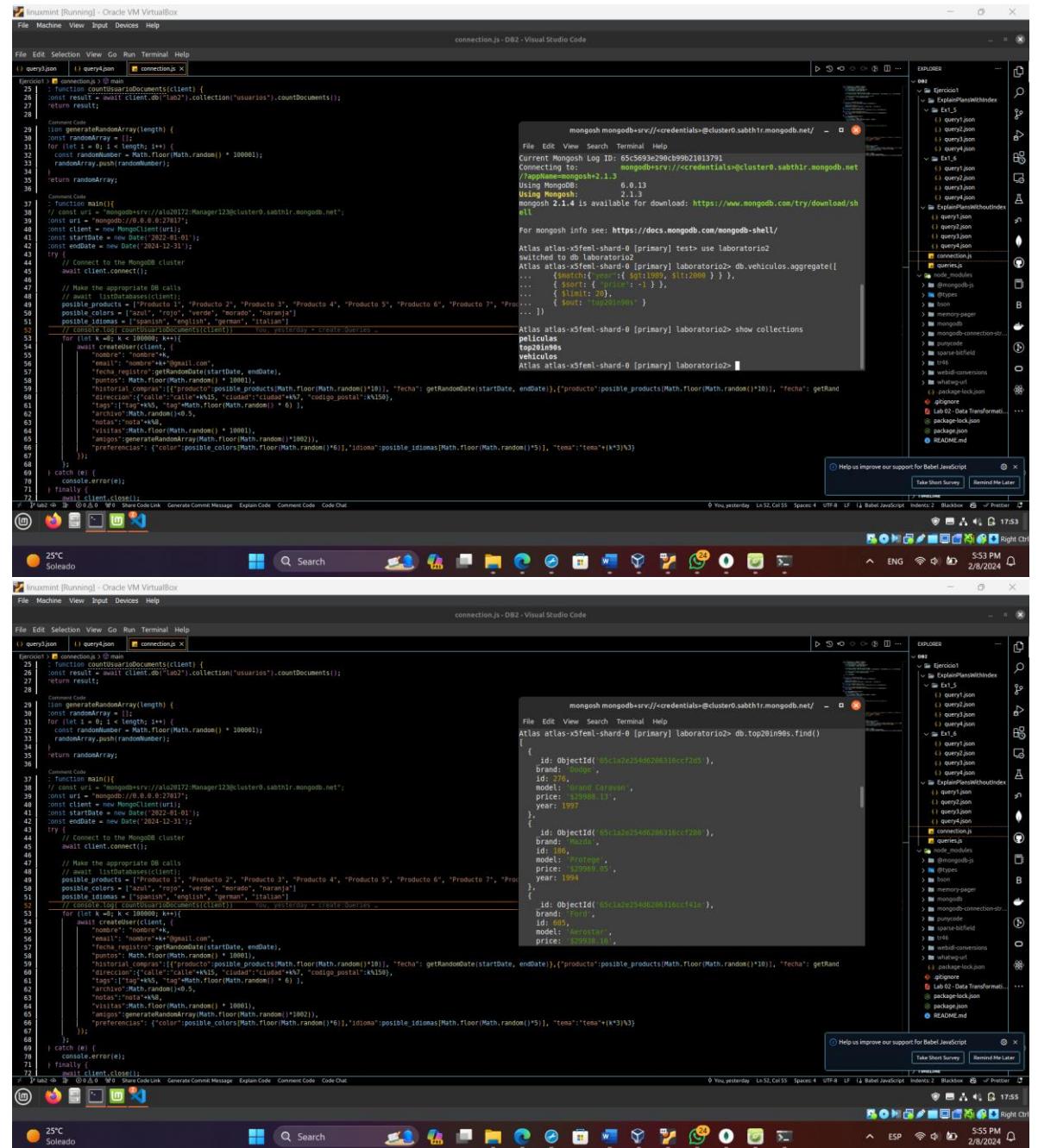


```

diggsy@diggstop:~/Downloads/mongodb-database-tools-ubuntu2204-x86_64-100.9.4/bin$ mongo
mongosh mongoimport://<credentials>@cluster0.sabth1r.mongodb.net/
File Edit View Search Terminal Tabs Help
mongosh mongoimport://<credentials>@cluster0.sabth1r.mongodb.net/ x diggsy@diggstop:~/Downloads/mongodb-database-tools-ubuntu2204-x86_64-100.9.4/bin x
{
  "_id": "057",
  "model": "Neon",
  "price": 159164.44,
  "year": 1995
}
Type "it" for more
atlas atlas-x5fem1-shard-0 [primary] laboratorio2: db.vehiculos.aggregate([ { $match: { "year": { $gt: 1989, $lt: 2000 } } }, { $sort: { "price": -1 } }, { $limit: 20 } ])
{
  "_id": ObjectId("65c1a2e25406200316ccf2d5"),
  "brand": "Dodge",
  "id": 776,
  "model": "Grand Caravan",
  "price": 129988.13,
  "year": 1997
{
  "_id": ObjectId("65c1a2e25406200316ccf280"),
  "brand": "Audi",
  "id": 106,
  "model": "Protege",
  "price": 129969.05,
  "year": 1994
{
  "_id": ObjectId("65c1a2e25406200316ccf41e"),
  "brand": "Ford",
  "id": 605,
  "model": "Aerostar",
  "price": 129938.16,
  "year": 1999
{
  "_id": ObjectId("65c1a2e25406200316ccf57b"),
  "brand": "Mazda",
  "id": 934,
  "model": "8-Series",
  "price": 129914.83,
  "year": 1991
{
  "_id": ObjectId("65c1a2e25406200316ccf252"),
  "brand": "Vauxhall",
  "id": 145,
  "model": "Castrilete",
  "price": 129913.78,
  "year": 1990
}
}
}

```

Genero un out para generar una colección y que esta sea la que pueda exportar a .CSV.



```

linuxmint [Running] : Oracle VM VirtualBox
File Edit Selection View Go Run Terminal Help
connection.js - DB2 - Visual Studio Code
File Edit Selection View Go Run Terminal Help
connection.js connections x
connections x
main
const countUsuariosDocuments = (client) => {
  const result = await client.db("lm02").collection("usuarios").countDocuments();
  return result;
}

Comment Code
function generateRandomArray(length) {
  const randomArray = [];
  for (let i = 0; i < length; i++) {
    const randomNumber = Math.floor(Math.random()) * 1000001;
    randomArray.push(randomNumber);
  }
  return randomArray;
}

Comment Code
main()
// const url = "mongodb://120.0.0.0:27017";
const url = "mongodb://0.0.0.0:27017";
const client = new MongoClient(url);
const startDate = new Date("2022-01-01");
const endDate = new Date("2024-12-31");
try {
  // Connect to the MongoDB cluster
  await client.connect();
  // Make the appropriate DB calls
  // await listDatabases(client);
  await client.db("lm02").collection("usuarios").insertOne({
    nombre: "Nombre"+Math.floor(Math.random()*1000),
    fecha_registro: getRandomDate(startDate, endDate),
    puntos: Math.floor(Math.random() * 1000),
    historial_compras: [
      {
        producto: "Producto 1", "Producto 2", "Producto 3", "Producto 4", "Producto 5", "Producto 6", "Producto 7", "Producto 8", "Producto 9", "Producto 10"
      },
      {
        color: ["azul", "rojo", "verde", "marrón", "naranja"]
      },
      {
        posible_idiomas: ["spanish", "english", "german", "italian"]
      }
    ],
    ultima_compra: "2023-01-01"
  });
  const result = await client.db("lm02").collection("usuarios").countDocuments();
  console.log(`You, yesterday created ${result} documents.`);
}

for (let k = 0; k < 100000; k++) {
  const client = new MongoClient(url);
  const result = await client.db("lm02").collection("usuarios").insertOne({
    nombre: "Nombre"+k,
    email: "Nombre"+k+"@gmail.com",
    fecha_registro: getRandomDate(startDate, endDate),
    puntos: Math.floor(Math.random() * 1000),
    historial_compras: [
      {
        producto: "Producto 1", "Producto 2", "Producto 3", "Producto 4", "Producto 5", "Producto 6", "Producto 7", "Producto 8", "Producto 9", "Producto 10"
      },
      {
        color: ["azul", "rojo", "verde", "marrón", "naranja"]
      },
      {
        posible_idiomas: ["spanish", "english", "german", "italian"]
      }
    ],
    ultima_compra: "2023-01-01"
  });
  const result = await client.db("lm02").collection("usuarios").countDocuments();
  console.log(`You, yesterday created ${result} documents.`);
}

try {
  await client.close();
} catch (e) {
  console.error(e);
}
finally {
  client.close();
}
}

Help us improve our support for Babel/JavaScript
Take Short Survey | Remind Me Later
File Edit Selection View Go Run Terminal Help
connection.js - DB2 - Visual Studio Code
File Edit Selection View Go Run Terminal Help
connection.js connections x
connections x
main
const countUsuariosDocuments = (client) => {
  const result = await client.db("lm02").collection("usuarios").countDocuments();
  return result;
}

Comment Code
function generateRandomArray(length) {
  const randomArray = [];
  for (let i = 0; i < length; i++) {
    const randomNumber = Math.floor(Math.random()) * 1000001;
    randomArray.push(randomNumber);
  }
  return randomArray;
}

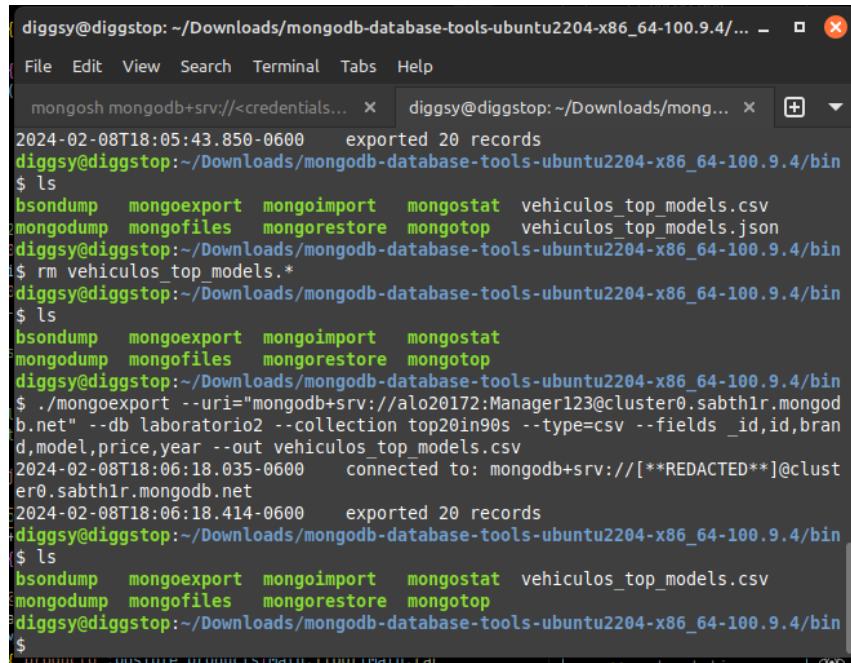
Comment Code
main()
// const url = "mongodb://120.0.0.0:27017";
const url = "mongodb://0.0.0.0:27017";
const client = new MongoClient(url);
const startDate = new Date("2022-01-01");
const endDate = new Date("2024-12-31");
try {
  // Connect to the MongoDB cluster
  await client.connect();
  // Make the appropriate DB calls
  // await listDatabases(client);
  await client.db("lm02").collection("usuarios").insertOne({
    nombre: "Nombre"+Math.floor(Math.random()*1000),
    fecha_registro: getRandomDate(startDate, endDate),
    puntos: Math.floor(Math.random() * 1000),
    historial_compras: [
      {
        producto: "Producto 1", "Producto 2", "Producto 3", "Producto 4", "Producto 5", "Producto 6", "Producto 7", "Producto 8", "Producto 9", "Producto 10"
      },
      {
        color: ["azul", "rojo", "verde", "marrón", "naranja"]
      },
      {
        posible_idiomas: ["spanish", "english", "german", "italian"]
      }
    ],
    ultima_compra: "2023-01-01"
  });
  const result = await client.db("lm02").collection("usuarios").countDocuments();
  console.log(`You, yesterday created ${result} documents.`);
}

for (let k = 0; k < 100000; k++) {
  const client = new MongoClient(url);
  const result = await client.db("lm02").collection("usuarios").insertOne({
    nombre: "Nombre"+k,
    email: "Nombre"+k+"@gmail.com",
    fecha_registro: getRandomDate(startDate, endDate),
    puntos: Math.floor(Math.random() * 1000),
    historial_compras: [
      {
        producto: "Producto 1", "Producto 2", "Producto 3", "Producto 4", "Producto 5", "Producto 6", "Producto 7", "Producto 8", "Producto 9", "Producto 10"
      },
      {
        color: ["azul", "rojo", "verde", "marrón", "naranja"]
      },
      {
        posible_idiomas: ["spanish", "english", "german", "italian"]
      }
    ],
    ultima_compra: "2023-01-01"
  });
  const result = await client.db("lm02").collection("usuarios").countDocuments();
  console.log(`You, yesterday created ${result} documents.`);
}

try {
  await client.close();
} catch (e) {
  console.error(e);
}
finally {
  client.close();
}
}

Help us improve our support for Babel/JavaScript
Take Short Survey | Remind Me Later

```



The screenshot shows a terminal window with two tabs. The left tab is titled 'mongosh mongodb+srv://<credentials...>' and the right tab is titled 'diggsy@diggstop:~/Downloads/mong...'. The terminal output is as follows:

```
2024-02-08T18:05:43.850-0600    exported 20 records
diggsy@diggstop:~/Downloads/mongodb-database-tools-ubuntu2204-x86_64-100.9.4/bin
$ ls
bsondump  mongoexport  mongoimport  mongostat  vehiculos_top_models.csv
mongodump  mongofiles   mongorestore  mongotop  vehiculos_top_models.json
diggsy@diggstop:~/Downloads/mongodb-database-tools-ubuntu2204-x86_64-100.9.4/bin
$ rm vehiculos_top_models.*
diggsy@diggstop:~/Downloads/mongodb-database-tools-ubuntu2204-x86_64-100.9.4/bin
$ ls
bsondump  mongoexport  mongoimport  mongostat
mongodump  mongofiles   mongorestore  mongotop
diggsy@diggstop:~/Downloads/mongodb-database-tools-ubuntu2204-x86_64-100.9.4/bin
$ ./mongoexport --uri="mongodb+srv://alo20172:Manager123@cluster0.sabth1r.mongodb.net" --db laboratorio2 --collection top20in90s --type=csv --fields _id,id,brand,model,price,year --out vehiculos_top_models.csv
2024-02-08T18:06:18.035-0600    connected to: mongodb+srv://[*REDACTED*]@cluster0.sabth1r.mongodb.net
2024-02-08T18:06:18.414-0600    exported 20 records
diggsy@diggstop:~/Downloads/mongodb-database-tools-ubuntu2204-x86_64-100.9.4/bin
$ ls
bsondump  mongoexport  mongoimport  mongostat  vehiculos_top_models.csv
mongodump  mongofiles   mongorestore  mongotop
diggsy@diggstop:~/Downloads/mongodb-database-tools-ubuntu2204-x86_64-100.9.4/bin
$
```

De manera que se generó el .csv y ahora solo lo copiaré a mi branch.

```
db.vehiculos.aggregate([
  {$match:{"year":{ $gt:1989, $lt:2000 } } },
  { $sort: { "price": -1 } },
  { $limit: 20},
  { $out: "top20in90s" }
])
```

```
db.top20in90s.find();
```