PONTIFICIA UNIVERSIDAD CATÓLICA MADRE Y MAESTRA

FACULTAD DE CIENCIAS E INGENIERÍA

Tecnologías Emergentes.

ISC-573

Tarea – índices.

Presentado por: Junior Hernández 2018-0999.

Entregado a: Alfredo Vásquez.

Fecha de entrega: 4 de julio del 2023.

Santiago de los caballeros; República Dominicana.

Desarrollo

1. Utilizando el script GenerarDatosIndices.js, generar un collection llamado facturacion, el cual debe contener 700,000 documentos de facturas con fecha desde 01-Enero-2021 hasta 30-Junio- 2022. Estos datos serán generados con datos aleatorios, utilizando las siguientes funciones:

- generarFecha(annio)
- obtenerCliente()
- obtenerCiudad()
- obtenerProducto()
- obtenerSucursal()

La estructura que debe tener cada documento de la colección es:

```
var documentoFactura =  { codigoFactura:1012,
                          sucursal:"SUCURSAL SANTIAGO",
                          codigoCliente:"1010",
                          fechaFactura: ISODate("2016-10-09 00:00:00" ),
                          nombreCliente:"OLIVER RODRIGUEZ",
                          ciudadFactura:"SANTIAGO",
                          ciudadDespacho:"LA VEGA",
                          totalFactura: 2394.00,
                          articulos:[ {codigoProducto:"39",descripcion:"PRODUCTO 39",precioVenta:14.40,cantidadFacturada:10.00,importe:144.00},
                                      {codigoProducto:"33",descripcion:"PRODUCTO 33",precioVenta:150.00,cantidadFacturada:15.00,importe:2250.00}
                                    ]

                        }
```

Para poder lograr esta parte, tomé el código y lo traduje a Python. Implemente el uso de sus bibliotecas para generar datos y combinaciones de ellos de manera aleatoria. El código fue el siguiente:

```python
import datetime
import random

from pymongo import MongoClient

# Conexión a la base de datos
client = MongoClient()
db = client['Facturacion']
facturacion = db['DocumentoFactura']

# Función para generar una fecha aleatoria
def generarFecha(annio):
    import datetime
    import random

    if annio is None:
        annio = 2016

    mes = random.randint(1, 12)
    if mes == 2:  # February
        dia = random.randint(1, 28)
    elif mes in [4, 6, 9, 11]:  # April, June, September, November
        dia = random.randint(1, 30)
```

```
    else:  # Other months
        dia = random.randint(1, 31)

    return datetime.datetime(annio, mes, dia, 0, 0)

# Función para obtener un cliente aleatorio
def obtenerCliente():
    listadoClientes = [
        {"ID": "1001", "NOMBRE": "MARIA CASTRO"},
        {"ID": "1002", "NOMBRE": "ROBERTO SANTANA"},
        {"ID": "1003", "NOMBRE": "MARIO RAMIREZ"},
        {"ID": "1004", "NOMBRE": "ALEXANDRA VASQUEZ"},
        {"ID": "1005", "NOMBRE": "JOSE RAMON RAMIREZ"},
        {"ID": "1006", "NOMBRE": "CARME GUTIERREZ"},
        {"ID": "1007", "NOMBRE": "ANGELA RODRIGUEZ"},
        {"ID": "1008", "NOMBRE": "MANUEL CEPEDA"},
        {"ID": "1009", "NOMBRE": "FRANCISCO PEREZ"},
        {"ID": "1010", "NOMBRE": "OLIVER RODRIGUEZ"}
    ]

    import random
    indice = random.randint(0, 9)
    return listadoClientes[indice]

# Función para obtener una ciudad aleatoria
def obtenerCiudad():
    listadoCiudad = [
        "SANTIAGO", "MOCA", "LA VEGA", "SANTO DOMINGO",
        "DAJABON", "MONTECRISTI", "HIGUEY", "MAO VALVERDE"
    ]

    import random
    indice = random.randint(0, 7)
    return listadoCiudad[indice]

# Función para obtener un producto aleatorio
def obtenerProducto():
    listadoProducto = [
        {"ID": "11", "DESCRIPCION": "PRODUCTO 11", "PRECIO": 14.00},
        {"ID": "14", "DESCRIPCION": "PRODUCTO 14", "PRECIO": 10.00},
        {"ID": "42", "DESCRIPCION": "PRODUCTO 42", "PRECIO": 9.80},
        {"ID": "72", "DESCRIPCION": "PRODUCTO 72", "PRECIO": 34.80},
        {"ID": "51", "DESCRIPCION": "PRODUCTO 51", "PRECIO": 42.40},
        {"ID": "41", "DESCRIPCION": "PRODUCTO 41", "PRECIO": 7.70},
        {"ID": "65", "DESCRIPCION": "PRODUCTO 65", "PRECIO": 16.80},
        {"ID": "20", "DESCRIPCION": "PRODUCTO 20", "PRECIO": 64.80},
        {"ID": "33", "DESCRIPCION": "PRODUCTO 33", "PRECIO": 150.00},
        {"ID": "60", "DESCRIPCION": "PRODUCTO 60", "PRECIO": 27.20},
        {"ID": "39", "DESCRIPCION": "PRODUCTO 39", "PRECIO": 14.40},
        {"ID": "49", "DESCRIPCION": "PRODUCTO 49", "PRECIO": 16.00}
    ]

    import random
    indice = random.randint(0, 11)
    return listadoProducto[indice]

# Función para obtener una sucursal aleatoria
```

```
def obtenerSucursal():
    listadoSucursal = [None, "SUCURSAL SANTIAGO", "SUCURSAL SANTO DOMINGO"]

    import random
    indice = random.randint(0, 2)
    return listadoSucursal[indice]

# Generar documentos de facturación
for i in range(700000):
    documentoFactura = {
        "codigoFactura": i + 1,
        "sucursal": obtenerSucursal(),
        "codigoCliente": obtenerCliente()["ID"],
        "fechaFactura": generarFecha(2021),
        "nombreCliente": obtenerCliente()["NOMBRE"],
        "ciudadFactura": obtenerCiudad(),
        "ciudadDespacho": obtenerCiudad(),
        "totalFactura": round(random.uniform(100, 5000), 2),
        "articulos": [
            {
                "codigoProducto": obtenerProducto()["ID"],
                "descripcion": obtenerProducto()["DESCRIPCION"],
                "precioVenta": obtenerProducto()["PRECIO"],
                "cantidadFacturada": round(random.uniform(1, 20), 2),
                "importe": round(random.uniform(10, 500), 2)
            },
            {
                "codigoProducto": obtenerProducto()["ID"],
                "descripcion": obtenerProducto()["DESCRIPCION"],
                "precioVenta": obtenerProducto()["PRECIO"],
                "cantidadFacturada": round(random.uniform(1, 20), 2),
                "importe": round(random.uniform(10, 500), 2)
            }
        ]
    }

    facturacion.insert_one(documentoFactura)
```

2. Luego de generados los documentos de la colección, desarrollar y hacer un análisis de las siguientes consultas, así como determinar si es o no necesario crear índices para que las mismas sean realizadas de forma eficiente. Deben explicar por qué razón fue necesario crear dicho índice y qué dato del análisis realizado lo llevo a tomar esa decisión.

a. Obtener un listado de las Facturas realizadas entre Entre 01/01/2021 y 02/01/2022, las mismas deben ser organizadas descendentemente por los atributos: “Ciudad”, “fechaFactura”. Los atributos de salida seran:

Ciudad, Fecha Factura, Codigo Cliente, TotalFactura

```
db.DocumentoFactura.find({"fechaFactura": {"$gte": ISODate("2021-01-01"),
"$lt": ISODate("2022-01-02")}},{"ciudadFactura": 1, "fechaFactura": 1,
"codigoCliente": 1, "totalFactura": 1, "_id": 0}).sort({"ciudadFactura": -1,
"fechaFactura": -1}).explain("executionStats")
```

Resultado


```
{
    explainVersion: '1',
    queryPlanner: {
        namespace: 'Facturacion.DocumentoFactura',
        indexFilterSet: false,
        parsedQuery: {
            '$and': [
                {
                    fechaFactura: { '$lt': ISODate("2022-01-
02T00:00:00.000Z") }
                },
                {
                    fechaFactura: { '$gte': ISODate("2021-01-
01T00:00:00.000Z") }
                }
            ]
        },
        queryHash: '214838C7',
        planCacheKey: '214838C7',
        maxIndexedOrSolutionsReached: false,
        maxIndexedAndSolutionsReached: false,
        maxScansToExplodeReached: false,
        winningPlan: {
            stage: 'SORT',
            sortPattern: { ciudadFactura: -1, fechaFactura: -1 },
            memLimit: 104857600,
            type: 'simple',
            inputStage: {
                stage: 'PROJECTION_SIMPLE',
                transformBy: {
                    ciudadFactura: 1,
                    fechaFactura: 1,
                    codigoCliente: 1,
                    totalFactura: 1,
                    _id: 0
                },
                inputStage: {
                    stage: 'COLLSCAN',
                    filter: {
                        '$and': [
                            {
                                fechaFactura: { '$lt': ISODate("2022-01-
02T00:00:00.000Z") }
                            },
                            {
                                fechaFactura: { '$gte': ISODate("2021-01-
01T00:00:00.000Z") }
                            }
                        ]
                    },
                    direction: 'forward'
                }
            }
        },
        rejectedPlans: []
```

```
    },
    executionStats: {
        executionSuccess: true,
        nReturned: 700000,
        executionTimeMillis: 9225,
        totalKeysExamined: 0,
        totalDocsExamined: 700000,
        executionStages: {
            stage: 'SORT',
            nReturned: 700000,
            executionTimeMillisEstimate: 5277,
            works: 1400003,
            advanced: 700000,
            needTime: 700002,
            needYield: 0,
            saveState: 1402,
            restoreState: 1402,
            isEOF: 1,
            sortPattern: { ciudadFactura: -1, fechaFactura: -1 },
            memLimit: 104857600,
            type: 'simple',
            totalDataSizeSorted: 147075401,
            usedDisk: true,
            spills: 2,
            inputStage: {
                stage: 'PROJECTION_SIMPLE',
                nReturned: 700000,
                executionTimeMillisEstimate: 1621,
                works: 700002,
                advanced: 700000,
                needTime: 1,
                needYield: 0,
                saveState: 1402,
                restoreState: 1402,
                isEOF: 1,
                transformBy: {
                    ciudadFactura: 1,
                    fechaFactura: 1,
                    codigoCliente: 1,
                    totalFactura: 1,
                    _id: 0
                },
                inputStage: {
                    stage: 'COLLSCAN',
                    filter: {
                        '$and': [
                            {
                                fechaFactura: { '$lt': ISODate("2022-01-
02T00:00:00.000Z") }
                            },
                            {
                                fechaFactura: { '$gte': ISODate("2021-01-
01T00:00:00.000Z") }
                            }
                        ]
                    },
                    nReturned: 700000,
```

```
                    executionTimeMillisEstimate: 1303,
                    works: 700002,
                    advanced: 700000,
                    needTime: 1,
                    needYield: 0,
                    saveState: 1402,
                    restoreState: 1402,
                    isEOF: 1,
                    direction: 'forward',
                    docsExamined: 700000
                }
            }
        }
    },
    command: {
        find: 'DocumentoFactura',
        filter: {
            fechaFactura: {
                '$gte': ISODate("2021-01-01T00:00:00.000Z"),
                '$lt': ISODate("2022-01-02T00:00:00.000Z")
            }
        },
        sort: { ciudadFactura: -1, fechaFactura: -1 },
        projection: {
            ciudadFactura: 1,
            fechaFactura: 1,
            codigoCliente: 1,
            totalFactura: 1,
            _id: 0
        },
        '$db': 'Facturacion'
    },
    serverInfo: {
        host: 'DESKTOP-LSTENG7',
        port: 27017,
        version: '6.0.6',
        gitVersion: '26b4851a412cc8b9b4a18cdb6cd0f9f642e06aa7'
    },
    serverParameters: {
        internalQueryFacetBufferSizeBytes: 104857600,
        internalQueryFacetMaxOutputDocSizeBytes: 104857600,
        internalLookupStageIntermediateDocumentMaxSizeBytes: 104857600,
        internalDocumentSourceGroupMaxMemoryBytes: 104857600,
        internalQueryMaxBlockingSortMemoryUsageBytes: 104857600,
        internalQueryProhibitBlockingMergeOnMongoS: 0,
        internalQueryMaxAddToSetBytes: 104857600,
        internalDocumentSourceSetWindowFieldsMaxMemoryBytes: 104857600
    },
    ok: 1
}
```


Conclusión:

Al observar la consulta se revela que se toma un tiempo considerable para completarla, con un total de 9225 milisegundos. La etapa más costosa es el "SORT" con un tiempo estimado de 5277 milisegundos y

in needtime 700002. Para mejorar la eficiencia de la consulta, se sugiere crear un índice que priorice los campos utilizados en el "sort", es decir, "ciudadFactura" y "fechaFactura" con un orden descendente (-1). Al hacer esto, se espera reducir significativamente el tiempo necesario para ordenar los resultados y, por ende, mejorar la eficiencia general de la consulta.

Crear el índice mencionado permitirá al sistema realizar búsquedas más rápidas y optimizadas basadas en los atributos de ordenamiento, lo que resultará en una mejora notable en el rendimiento de la consulta.

```
db.DocumentoFactura.createIndex({ "ciudadFactura": -1, "fechaFactura": -1 },
{ name: "ciudadFechaIndex" })
```

Resultado después de aplicar el índice:

```
{
    explainVersion: '1',
    queryPlanner: {
        namespace: 'Facturacion.DocumentoFactura',
        indexFilterSet: false,
        parsedQuery: {
            '$and': [
                {
                    fechaFactura: { '$lt': ISODate("2022-01-
02T00:00:00.000Z") }
                },
                {
                    fechaFactura: { '$gte': ISODate("2021-01-
01T00:00:00.000Z") }
                }
            ]
        },
        queryHash: '214838C7',
        planCacheKey: 'B1AEA4F8',
        maxIndexedOrSolutionsReached: false,
        maxIndexedAndSolutionsReached: false,
        maxScansToExplodeReached: false,
        winningPlan: {
            stage: 'PROJECTION_SIMPLE',
            transformBy: {
                ciudadFactura: 1,
                fechaFactura: 1,
                codigoCliente: 1,
                totalFactura: 1,
                _id: 0
            },
            inputStage: {
                stage: 'FETCH',
                filter: {
                    '$and': [
                        {
                            fechaFactura: { '$lt': ISODate("2022-01-
02T00:00:00.000Z") }
                        },
                        {
```

```
                            fechaFactura: { '$gte': ISODate("2021-01-
01T00:00:00.000Z") }
                        }
                    ]
                },
                inputStage: {
                    stage: 'IXSCAN',
                    keyPattern: { ciudadFactura: -1, fechaFactura: -1 },
                    indexName: 'ciudadFechaIndex',
                    isMultiKey: false,
                    multiKeyPaths: { ciudadFactura: [], fechaFactura: [] },
                    isUnique: false,
                    isSparse: false,
                    isPartial: false,
                    indexVersion: 2,
                    direction: 'forward',
                    indexBounds: {
                        ciudadFactura: [ '[MaxKey, MinKey]' ],
                        fechaFactura: [ '[MaxKey, MinKey]' ]
                    }
                }
            }
        },
        rejectedPlans: []
    },
    executionStats: {
        executionSuccess: true,
        nReturned: 700000,
        executionTimeMillis: 4839,
        totalKeysExamined: 700000,
        totalDocsExamined: 700000,
        executionStages: {
            stage: 'PROJECTION_SIMPLE',
            nReturned: 700000,
            executionTimeMillisEstimate: 1779,
            works: 700001,
            advanced: 700000,
            needTime: 0,
            needYield: 0,
            saveState: 700,
            restoreState: 700,
            isEOF: 1,
            transformBy: {
                ciudadFactura: 1,
                fechaFactura: 1,
                codigoCliente: 1,
                totalFactura: 1,
                _id: 0
            },
            inputStage: {
                stage: 'FETCH',
                filter: {
                    '$and': [
                        {
                            fechaFactura: { '$lt': ISODate("2022-01-
02T00:00:00.000Z") }
                        },
```

```
                        {
                            fechaFactura: { '$gte': ISODate("2021-01-
01T00:00:00.000Z") }
                        }
                    ]
                },
                nReturned: 700000,
                executionTimeMillisEstimate: 1538,
                works: 700001,
                advanced: 700000,
                needTime: 0,
                needYield: 0,
                saveState: 700,
                restoreState: 700,
                isEOF: 1,
                docsExamined: 700000,
                alreadyHasObj: 0,
                inputStage: {
                    stage: 'IXSCAN',
                    nReturned: 700000,
                    executionTimeMillisEstimate: 353,
                    works: 700001,
                    advanced: 700000,
                    needTime: 0,
                    needYield: 0,
                    saveState: 700,
                    restoreState: 700,
                    isEOF: 1,
                    keyPattern: { ciudadFactura: -1, fechaFactura: -1 },
                    indexName: 'ciudadFechaIndex',
                    isMultiKey: false,
                    multiKeyPaths: { ciudadFactura: [], fechaFactura: [] },
                    isUnique: false,
                    isSparse: false,
                    isPartial: false,
                    indexVersion: 2,
                    direction: 'forward',
                    indexBounds: {
                        ciudadFactura: [ '[MaxKey, MinKey]' ],
                        fechaFactura: [ '[MaxKey, MinKey]' ]
                    },
                    keysExamined: 700000,
                    seeks: 1,
                    dupsTested: 0,
                    dupsDropped: 0
                }
            }
        }
    },
    command: {
        find: 'DocumentoFactura',
        filter: {
            fechaFactura: {
                '$gte': ISODate("2021-01-01T00:00:00.000Z"),
                '$lt': ISODate("2022-01-02T00:00:00.000Z")
            }
        },
```

```
        sort: { ciudadFactura: -1, fechaFactura: -1 },
        projection: {
            ciudadFactura: 1,
            fechaFactura: 1,
            codigoCliente: 1,
            totalFactura: 1,
            _id: 0
        },
        '$db': 'Facturacion'
    },
    serverInfo: {
        host: 'DESKTOP-LSTENG7',
        port: 27017,
        version: '6.0.6',
        gitVersion: '26b4851a412cc8b9b4a18cdb6cd0f9f642e06aa7'
    },
    serverParameters: {
        internalQueryFacetBufferSizeBytes: 104857600,
        internalQueryFacetMaxOutputDocSizeBytes: 104857600,
        internalLookupStageIntermediateDocumentMaxSizeBytes: 104857600,
        internalDocumentSourceGroupMaxMemoryBytes: 104857600,
        internalQueryMaxBlockingSortMemoryUsageBytes: 104857600,
        internalQueryProhibitBlockingMergeOnMongoS: 0,
        internalQueryMaxAddToSetBytes: 104857600,
        internalDocumentSourceSetWindowFieldsMaxMemoryBytes: 104857600
    },
    ok: 1
}
```


Como se puede ver hubo una una baja significativa del tiempo de ejecución ahora siendo de 4839 milisegundos.

b. Desarrollar una consulta que muestre por cada ciudad y artículo el total vendido para los códigos clientes: 1001,1002,1003 y 1004, cuya fechaFactura sea >=01/05/2021 y <= 31/08/2022. Datos a mostrar:

Ciudad, Producto, Monto Total Producto

```
db.DocumentoFactura.aggregate([{$match: {codigoCliente: {$in: ["1001",
"1002", "1003", "1004"]}, fechaFactura: {$gte: ISODate("2021-05-01"), $lte:
ISODate("2022-08-31")}}}, {$unwind: "$articulos"}, {$group: {_id: {ciudad:
"$ciudadFactura", producto: "$articulos.descripcion"}, montoTotal: {$sum:
"$articulos.importe"}}}, {$project: {_id: 0, Ciudad: "$_id.ciudad", Producto:
"$_id.producto", "Monto Total Producto":
"$montoTotal"}}]).explain("executionStats")
```

Resultado:

```
{
    explainVersion: '1',
    stages: [
        {
            '$cursor': {
                queryPlanner: {
                    namespace: 'Facturacion.DocumentoFactura',
                    indexFilterSet: false,
                    parsedQuery: {
                        '$and': [
                            {
                                fechaFactura: { '$lte': ISODate("2022-08-
31T00:00:00.000Z") }
                            },
                            {
                                fechaFactura: { '$gte': ISODate("2021-05-
01T00:00:00.000Z") }
                            },
                            {
                                codigoCliente: { '$in': [ '1001', '1002',
'1003', '1004' ] }
                            }
                        ]
                    },
                    queryHash: '3F4932BF',
                    planCacheKey: '3F4932BF',
                    maxIndexedOrSolutionsReached: false,
                    maxIndexedAndSolutionsReached: false,
                    maxScansToExplodeReached: false,
                    winningPlan: {
                        stage: 'PROJECTION_SIMPLE',
                        transformBy: { articulos: 1, ciudadFactura: 1, _id: 0
},
                        inputStage: {
                            stage: 'COLLSCAN',
                            filter: {
                                '$and': [
                                    {
                                        fechaFactura: { '$lte':
ISODate("2022-08-31T00:00:00.000Z") }
```

```
                                    },
                                    {
                                        fechaFactura: { '$gte':
ISODate("2021-05-01T00:00:00.000Z") }
                                    },
                                    {
                                        codigoCliente: { '$in': [ '1001',
'1002', '1003', '1004' ] }
                                    }
                                ]
                            },
                            direction: 'forward'
                        }
                    },
                    rejectedPlans: []
                },
                executionStats: {
                    executionSuccess: true,
                    nReturned: 186799,
                    executionTimeMillis: 2527,
                    totalKeysExamined: 0,
                    totalDocsExamined: 700000,
                    executionStages: {
                        stage: 'PROJECTION_SIMPLE',
                        nReturned: 186799,
                        executionTimeMillisEstimate: 158,
                        works: 700002,
                        advanced: 186799,
                        needTime: 513202,
                        needYield: 0,
                        saveState: 726,
                        restoreState: 726,
                        isEOF: 1,
                        transformBy: { articulos: 1, ciudadFactura: 1, _id: 0
},
                        inputStage: {
                            stage: 'COLLSCAN',
                            filter: {
                                '$and': [
                                    {
                                        fechaFactura: { '$lte':
ISODate("2022-08-31T00:00:00.000Z") }
                                    },
                                    {
                                        fechaFactura: { '$gte':
ISODate("2021-05-01T00:00:00.000Z") }
                                    },
                                    {
                                        codigoCliente: { '$in': [ '1001',
'1002', '1003', '1004' ] }
                                    }
                                ]
                            },
                            nReturned: 186799,
                            executionTimeMillisEstimate: 137,
                            works: 700002,
                            advanced: 186799,
```

```
                        needTime: 513202,
                        needYield: 0,
                        saveState: 726,
                        restoreState: 726,
                        isEOF: 1,
                        direction: 'forward',
                        docsExamined: 700000
                    }
                }
            }
        },
        nReturned: Long("186799"),
        executionTimeMillisEstimate: Long("1630")
    },
    {
        '$unwind': { path: '$articulos' },
        nReturned: Long("373598"),
        executionTimeMillisEstimate: Long("1887")
    },
    {
        '$group': {
            _id: {
                ciudad: '$ciudadFactura',
                producto: '$articulos.descripcion'
            },
            montoTotal: { '$sum': '$articulos.importe' }
        },
        maxAccumulatorMemoryUsageBytes: { montoTotal: Long("7680") },
        totalOutputDataSizeBytes: Long("45948"),
        usedDisk: false,
        spills: Long("0"),
        nReturned: Long("96"),
        executionTimeMillisEstimate: Long("2527")
    },
    {
        '$project': {
            Ciudad: '$_id.ciudad',
            Producto: '$_id.producto',
            'Monto Total Producto': '$montoTotal',
            _id: false
        },
        nReturned: Long("96"),
        executionTimeMillisEstimate: Long("2527")
    }
],
serverInfo: {
    host: 'DESKTOP-LSTENG7',
    port: 27017,
    version: '6.0.6',
    gitVersion: '26b4851a412cc8b9b4a18cdb6cd0f9f642e06aa7'
},
serverParameters: {
    internalQueryFacetBufferSizeBytes: 104857600,
    internalQueryFacetMaxOutputDocSizeBytes: 104857600,
    internalLookupStageIntermediateDocumentMaxSizeBytes: 104857600,
    internalDocumentSourceGroupMaxMemoryBytes: 104857600,
    internalQueryMaxBlockingSortMemoryUsageBytes: 104857600,
```

```
        internalQueryProhibitBlockingMergeOnMongoS: 0,
        internalQueryMaxAddToSetBytes: 104857600,
        internalDocumentSourceSetWindowFieldsMaxMemoryBytes: 104857600
    },
    command: {
        aggregate: 'DocumentoFactura',
        pipeline: [
            {
                '$match': {
                    codigoCliente: { '$in': [ '1001', '1002', '1003', '1004'
] },
                    fechaFactura: {
                        '$gte': ISODate("2021-05-01T00:00:00.000Z"),
                        '$lte': ISODate("2022-08-31T00:00:00.000Z")
                    }
                }
            },
            { '$unwind': '$articulos' },
            {
                '$group': {
                    _id: {
                        ciudad: '$ciudadFactura',
                        producto: '$articulos.descripcion'
                    },
                    montoTotal: { '$sum': '$articulos.importe' }
                }
            },
            {
                '$project': {
                    _id: 0,
                    Ciudad: '$_id.ciudad',
                    Producto: '$_id.producto',
                    'Monto Total Producto': '$montoTotal'
                }
            }
        ],
        cursor: {},
        '$db': 'Facturacion'
    },
    ok: 1
}
```

Conclusión:

Al analizar la consulta presentada, se observa que la fase que requiere más tiempo de procesamiento es la etapa del "match" con un needTime de 513202, donde se aplican filtros a los campos "fechaFactura" y "codigoCliente". Para mejorar la eficiencia de la consulta, se recomienda crear un índice basado en el campo "fechaFactura", ya que este atributo abarca una mayor cantidad de datos. Además, el campo "codigoCliente" también puede beneficiarse de un índice secundario. Estos índices permitirían una búsqueda más rápida y eficiente, reduciendo el tiempo necesario para ejecutar la consulta, y si se puede inplmentar el uso de uno compuesto conformado por estos dos atributos se supone que su mejora seria aún mejor.

```
db.DocumentoFactura.createIndex({ codigoCliente: 1, fechaFactura: 1 }, {
name: "clienteFechaIndex" })
```

Resultado después de aplicar el índice:

```
{
    explainVersion: '1',
    stages: [
        {
            '$cursor': {
                queryPlanner: {
                    namespace: 'Facturacion.DocumentoFactura',
                    indexFilterSet: false,
                    parsedQuery: {
                        '$and': [
                            {
                                fechaFactura: { '$lte': ISODate("2022-08-
31T00:00:00.000Z") }
                            },
                            {
                                fechaFactura: { '$gte': ISODate("2021-05-
01T00:00:00.000Z") }
                            },
                            {
                                codigoCliente: { '$in': [ '1001', '1002',
'1003', '1004' ] }
                            }
                        ]
                    },
                    queryHash: '3F4932BF',
                    planCacheKey: '898C4CA7',
                    maxIndexedOrSolutionsReached: false,
                    maxIndexedAndSolutionsReached: false,
                    maxScansToExplodeReached: false,
                    winningPlan: {
                        stage: 'PROJECTION_SIMPLE',
                        transformBy: { articulos: 1, ciudadFactura: 1, _id: 0
},
                        inputStage: {
                            stage: 'FETCH',
                            inputStage: {
                                stage: 'IXSCAN',
                                keyPattern: { codigoCliente: 1, fechaFactura:
1 },
                                indexName: 'clienteFechaIndex',
                                isMultiKey: false,
                                multiKeyPaths: { codigoCliente: [],
fechaFactura: [] },
                                isUnique: false,
                                isSparse: false,
                                isPartial: false,
                                indexVersion: 2,
                                direction: 'forward',
                                indexBounds: {
                                    codigoCliente: [
                                        '["1001", "1001"]',
```

```
                                    '["1002", "1002"]',
                                    '["1003", "1003"]',
                                    '["1004", "1004"]'
                                ],
                                fechaFactura: [
                                    '[new Date(1619827200000), new
Date(1661904000000)]'
                                ]
                            }
                        }
                    }
                },
                rejectedPlans: []
            },
            executionStats: {
                executionSuccess: true,
                nReturned: 186799,
                executionTimeMillis: 2391,
                totalKeysExamined: 186803,
                totalDocsExamined: 186799,
                executionStages: {
                    stage: 'PROJECTION_SIMPLE',
                    nReturned: 186799,
                    executionTimeMillisEstimate: 487,
                    works: 186803,
                    advanced: 186799,
                    needTime: 3,
                    needYield: 0,
                    saveState: 216,
                    restoreState: 216,
                    isEOF: 1,
                    transformBy: { articulos: 1, ciudadFactura: 1, _id: 0
},
                    inputStage: {
                        stage: 'FETCH',
                        nReturned: 186799,
                        executionTimeMillisEstimate: 421,
                        works: 186803,
                        advanced: 186799,
                        needTime: 3,
                        needYield: 0,
                        saveState: 216,
                        restoreState: 216,
                        isEOF: 1,
                        docsExamined: 186799,
                        alreadyHasObj: 0,
                        inputStage: {
                            stage: 'IXSCAN',
                            nReturned: 186799,
                            executionTimeMillisEstimate: 176,
                            works: 186803,
                            advanced: 186799,
                            needTime: 3,
                            needYield: 0,
                            saveState: 216,
                            restoreState: 216,
                            isEOF: 1,
```

```
                                keyPattern: { codigoCliente: 1, fechaFactura:
1 },
                                indexName: 'clienteFechaIndex',
                                isMultiKey: false,
                                multiKeyPaths: { codigoCliente: [],
fechaFactura: [] },
                                isUnique: false,
                                isSparse: false,
                                isPartial: false,
                                indexVersion: 2,
                                direction: 'forward',
                                indexBounds: {
                                    codigoCliente: [
                                        '["1001", "1001"]',
                                        '["1002", "1002"]',
                                        '["1003", "1003"]',
                                        '["1004", "1004"]'
                                    ],
                                    fechaFactura: [
                                        '[new Date(1619827200000), new
Date(1661904000000)]'
                                    ]
                                },
                                keysExamined: 186803,
                                seeks: 4,
                                dupsTested: 0,
                                dupsDropped: 0
                            }
                        }
                    }
                }
            },
            nReturned: Long("186799"),
            executionTimeMillisEstimate: Long("1407")
        },
        {
            '$unwind': { path: '$articulos' },
            nReturned: Long("373598"),
            executionTimeMillisEstimate: Long("1706")
        },
        {
            '$group': {
                _id: {
                    ciudad: '$ciudadFactura',
                    producto: '$articulos.descripcion'
                },
                montoTotal: { '$sum': '$articulos.importe' }
            },
            maxAccumulatorMemoryUsageBytes: { montoTotal: Long("7680") },
            totalOutputDataSizeBytes: Long("45948"),
            usedDisk: false,
            spills: Long("0"),
            nReturned: Long("96"),
            executionTimeMillisEstimate: Long("2383")
        },
        {
            '$project': {
```

```
                Ciudad: '$_id.ciudad',
                Producto: '$_id.producto',
                'Monto Total Producto': '$montoTotal',
                _id: false
            },
            nReturned: Long("96"),
            executionTimeMillisEstimate: Long("2383")
        }
    ],
    serverInfo: {
        host: 'DESKTOP-LSTENG7',
        port: 27017,
        version: '6.0.6',
        gitVersion: '26b4851a412cc8b9b4a18cdb6cd0f9f642e06aa7'
    },
    serverParameters: {
        internalQueryFacetBufferSizeBytes: 104857600,
        internalQueryFacetMaxOutputDocSizeBytes: 104857600,
        internalLookupStageIntermediateDocumentMaxSizeBytes: 104857600,
        internalDocumentSourceGroupMaxMemoryBytes: 104857600,
        internalQueryMaxBlockingSortMemoryUsageBytes: 104857600,
        internalQueryProhibitBlockingMergeOnMongoS: 0,
        internalQueryMaxAddToSetBytes: 104857600,
        internalDocumentSourceSetWindowFieldsMaxMemoryBytes: 104857600
    },
    command: {
        aggregate: 'DocumentoFactura',
        pipeline: [
            {
                '$match': {
                    codigoCliente: { '$in': [ '1001', '1002', '1003', '1004'
] },
                    fechaFactura: {
                        '$gte': ISODate("2021-05-01T00:00:00.000Z"),
                        '$lte': ISODate("2022-08-31T00:00:00.000Z")
                    }
                }
            },
            { '$unwind': '$articulos' },
            {
                '$group': {
                    _id: {
                        ciudad: '$ciudadFactura',
                        producto: '$articulos.descripcion'
                    },
                    montoTotal: { '$sum': '$articulos.importe' }
                }
            },
            {
                '$project': {
                    _id: 0,
                    Ciudad: '$_id.ciudad',
                    Producto: '$_id.producto',
                    'Monto Total Producto': '$montoTotal'
                }
            }
        ],
```

```
        cursor: {},
        '$db': 'Facturacion'
    },
    ok: 1
}
```

Como se puede ver el uso de índices redujo significativamente el needtime llegando a ser este solo 3.

c. Obtener un listado de las facturas generadas en “LA VEGA” y despachadas hacia “SANTIAGO”. Los datos a mostrar son:

CodigoFactura, CodigoCliente, CiudadFactura, CiudadDespacho.

```
db.DocumentoFactura.find({"ciudadFactura": "LA VEGA", "ciudadDespacho":
"SANTIAGO"}, {"codigoFactura": 1, "codigoCliente": 1, "ciudadFactura": 1,
"ciudadDespacho": 1, "_id": 0}).explain("executionStats")
```

Resultado:


```
{
    explainVersion: '1',
    queryPlanner: {
        namespace: 'Facturacion.DocumentoFactura',
        indexFilterSet: false,
        parsedQuery: {
            '$and': [
                { ciudadDespacho: { '$eq': 'SANTIAGO' } },
                { ciudadFactura: { '$eq': 'LA VEGA' } }
            ]
        },
        queryHash: '86EE4CAA',
        planCacheKey: '86EE4CAA',
        maxIndexedOrSolutionsReached: false,
        maxIndexedAndSolutionsReached: false,
        maxScansToExplodeReached: false,
        winningPlan: {
            stage: 'PROJECTION_SIMPLE',
            transformBy: {
                codigoFactura: 1,
                codigoCliente: 1,
                ciudadFactura: 1,
                ciudadDespacho: 1,
                _id: 0
            },
            inputStage: {
                stage: 'COLLSCAN',
                filter: {
                    '$and': [
                        { ciudadDespacho: { '$eq': 'SANTIAGO' } },
                        { ciudadFactura: { '$eq': 'LA VEGA' } }
                    ]
                },
                direction: 'forward'
            }
        },
        rejectedPlans: []
    },
    executionStats: {
        executionSuccess: true,
        nReturned: 10951,
        executionTimeMillis: 1173,
        totalKeysExamined: 0,
        totalDocsExamined: 700000,
```

```
        executionStages: {
            stage: 'PROJECTION_SIMPLE',
            nReturned: 10951,
            executionTimeMillisEstimate: 98,
            works: 700002,
            advanced: 10951,
            needTime: 689050,
            needYield: 0,
            saveState: 700,
            restoreState: 700,
            isEOF: 1,
            transformBy: {
                codigoFactura: 1,
                codigoCliente: 1,
                ciudadFactura: 1,
                ciudadDespacho: 1,
                _id: 0
            },
            inputStage: {
                stage: 'COLLSCAN',
                filter: {
                    '$and': [
                        { ciudadDespacho: { '$eq': 'SANTIAGO' } },
                        { ciudadFactura: { '$eq': 'LA VEGA' } }
                    ]
                },
                nReturned: 10951,
                executionTimeMillisEstimate: 88,
                works: 700002,
                advanced: 10951,
                needTime: 689050,
                needYield: 0,
                saveState: 700,
                restoreState: 700,
                isEOF: 1,
                direction: 'forward',
                docsExamined: 700000
            }
        }
    },
    command: {
        find: 'DocumentoFactura',
        filter: { ciudadFactura: 'LA VEGA', ciudadDespacho: 'SANTIAGO' },
        projection: {
            codigoFactura: 1,
            codigoCliente: 1,
            ciudadFactura: 1,
            ciudadDespacho: 1,
            _id: 0
        },
        '$db': 'Facturacion'
    },
    serverInfo: {
        host: 'DESKTOP-LSTENG7',
        port: 27017,
        version: '6.0.6',
        gitVersion: '26b4851a412cc8b9b4a18cdb6cd0f9f642e06aa7'
```

```
    },
    serverParameters: {
        internalQueryFacetBufferSizeBytes: 104857600,
        internalQueryFacetMaxOutputDocSizeBytes: 104857600,
        internalLookupStageIntermediateDocumentMaxSizeBytes: 104857600,
        internalDocumentSourceGroupMaxMemoryBytes: 104857600,
        internalQueryMaxBlockingSortMemoryUsageBytes: 104857600,
        internalQueryProhibitBlockingMergeOnMongoS: 0,
        internalQueryMaxAddToSetBytes: 104857600,
        internalDocumentSourceSetWindowFieldsMaxMemoryBytes: 104857600
    },
    ok: 1
}
```

Conclusión:

Al analizar la consulta muestra que, aunque actualmente se está realizando un COLLSCAN, lo cual implica que no hay un índice específico para acelerar la búsqueda, el needtime (tiempo necesario) es alto, lo que sugiere que la consulta podría beneficiarse de un índice adecuado.

La consulta busca un listado de facturas generadas en "LA VEGA" y despachadas hacia "SANTIAGO", y los campos que se desean mostrar son "CodigoFactura", "CodigoCliente", "CiudadFactura" y "CiudadDespacho". Dado que se filtra por dos campos, "CiudadFactura" y "CiudadDespacho", la creación de un índice compuesto en estos dos atributos (CiudadDespacho: 1, CiudadFactura: 1) podría eficientizar considerablemente la búsqueda. Al crear un índice compuesto con los campos más utilizados en las consultas, se optimiza la búsqueda y se reduce el tiempo de ejecución, lo que mejorará el rendimiento general de la consulta. Por lo tanto, se recomienda crear un índice compuesto en "CiudadDespacho" y "CiudadFactura" para esta consulta en particular.

```
db.DocumentoFactura.createIndex({ "ciudadDespacho": 1, "ciudadFactura": 1 },
{ name: "ciudadesIndex" })
```

Resultado después de aplicar el índice:

```
{
    explainVersion: '1',
    queryPlanner: {
        namespace: 'Facturacion.DocumentoFactura',
        indexFilterSet: false,
        parsedQuery: {
            '$and': [
                { ciudadDespacho: { '$eq': 'SANTIAGO' } },
                { ciudadFactura: { '$eq': 'LA VEGA' } }
            ]
        },
        queryHash: '86EE4CAA',
        planCacheKey: 'ABCCFA56',
        maxIndexedOrSolutionsReached: false,
        maxIndexedAndSolutionsReached: false,
        maxScansToExplodeReached: false,
        winningPlan: {
            stage: 'PROJECTION_SIMPLE',
```

```
            transformBy: {
                codigoFactura: 1,
                codigoCliente: 1,
                ciudadFactura: 1,
                ciudadDespacho: 1,
                _id: 0
            },
            inputStage: {
                stage: 'FETCH',
                inputStage: {
                    stage: 'IXSCAN',
                    keyPattern: { ciudadDespacho: 1, ciudadFactura: 1 },
                    indexName: 'ciudadesIndex',
                    isMultiKey: false,
                    multiKeyPaths: { ciudadDespacho: [], ciudadFactura: [] },
                    isUnique: false,
                    isSparse: false,
                    isPartial: false,
                    indexVersion: 2,
                    direction: 'forward',
                    indexBounds: {
                        ciudadDespacho: [ '["SANTIAGO", "SANTIAGO"]' ],
                        ciudadFactura: [ '["LA VEGA", "LA VEGA"]' ]
                    }
                }
            }
        },
        rejectedPlans: []
    },
    executionStats: {
        executionSuccess: true,
        nReturned: 10951,
        executionTimeMillis: 88,
        totalKeysExamined: 10951,
        totalDocsExamined: 10951,
        executionStages: {
            stage: 'PROJECTION_SIMPLE',
            nReturned: 10951,
            executionTimeMillisEstimate: 41,
            works: 10952,
            advanced: 10951,
            needTime: 0,
            needYield: 0,
            saveState: 11,
            restoreState: 11,
            isEOF: 1,
            transformBy: {
                codigoFactura: 1,
                codigoCliente: 1,
                ciudadFactura: 1,
                ciudadDespacho: 1,
                _id: 0
            },
            inputStage: {
                stage: 'FETCH',
                nReturned: 10951,
                executionTimeMillisEstimate: 38,
```

```
            works: 10952,
            advanced: 10951,
            needTime: 0,
            needYield: 0,
            saveState: 11,
            restoreState: 11,
            isEOF: 1,
            docsExamined: 10951,
            alreadyHasObj: 0,
            inputStage: {
                stage: 'IXSCAN',
                nReturned: 10951,
                executionTimeMillisEstimate: 10,
                works: 10952,
                advanced: 10951,
                needTime: 0,
                needYield: 0,
                saveState: 11,
                restoreState: 11,
                isEOF: 1,
                keyPattern: { ciudadDespacho: 1, ciudadFactura: 1 },
                indexName: 'ciudadesIndex',
                isMultiKey: false,
                multiKeyPaths: { ciudadDespacho: [], ciudadFactura: [] },
                isUnique: false,
                isSparse: false,
                isPartial: false,
                indexVersion: 2,
                direction: 'forward',
                indexBounds: {
                    ciudadDespacho: [ '["SANTIAGO", "SANTIAGO"]' ],
                    ciudadFactura: [ '["LA VEGA", "LA VEGA"]' ]
                },
                keysExamined: 10951,
                seeks: 1,
                dupsTested: 0,
                dupsDropped: 0
            }
        }
    }
},
command: {
    find: 'DocumentoFactura',
    filter: { ciudadFactura: 'LA VEGA', ciudadDespacho: 'SANTIAGO' },
    projection: {
        codigoFactura: 1,
        codigoCliente: 1,
        ciudadFactura: 1,
        ciudadDespacho: 1,
        _id: 0
    },
    '$db': 'Facturacion'
},
serverInfo: {
    host: 'DESKTOP-LSTENG7',
    port: 27017,
    version: '6.0.6',
```

```
        gitVersion: '26b4851a412cc8b9b4a18cdb6cd0f9f642e06aa7'
    },
    serverParameters: {
        internalQueryFacetBufferSizeBytes: 104857600,
        internalQueryFacetMaxOutputDocSizeBytes: 104857600,
        internalLookupStageIntermediateDocumentMaxSizeBytes: 104857600,
        internalDocumentSourceGroupMaxMemoryBytes: 104857600,
        internalQueryMaxBlockingSortMemoryUsageBytes: 104857600,
        internalQueryProhibitBlockingMergeOnMongoS: 0,
        internalQueryMaxAddToSetBytes: 104857600,
        internalDocumentSourceSetWindowFieldsMaxMemoryBytes: 104857600
    },
    ok: 1
}
```

Nuevamente se puede ver la ventaja del uso de los índices en donde se puede observar claramente la notable reducción en el needtime siendo este 0.

d. Mostrar un listado de las ventas realizadas a los clientes por sucursal. Esto es mostrar por cada cliente en cuales sucursal ha comprado y qué monto. Los datos a mostrar son: CodigoCliente, NombreCliente, Sucursal y VentaTotal.

```
db.DocumentoFactura.aggregate([{ $group: { _id: { codigoCliente:
"$codigoCliente", nombreCliente: "$nombreCliente", sucursal: "$sucursal" },
ventaTotal: { $sum: "$totalFactura" } } }, { $project: { _id: 0,
CodigoCliente: "$_id.codigoCliente", NombreCliente: "$_id.nombreCliente",
Sucursal: "$_id.sucursal", VentaTotal: "$ventaTotal" }
}]).explain("executionStats")
```

Resultado:

```
{
    explainVersion: '2',
    stages: [
        {
            '$cursor': {
                queryPlanner: {
                    namespace: 'Facturacion.DocumentoFactura',
                    indexFilterSet: false,
                    parsedQuery: {},
                    queryHash: 'C7D869A3',
                    planCacheKey: 'C7D869A3',
                    maxIndexedOrSolutionsReached: false,
                    maxIndexedAndSolutionsReached: false,
                    maxScansToExplodeReached: false,
                    winningPlan: {
                        queryPlan: {
                            stage: 'GROUP',
                            planNodeId: 2,
                            inputStage: {
                                stage: 'COLLSCAN',
                                planNodeId: 1,
                                filter: {},
                                direction: 'forward'
                            }
                        },
                        slotBasedPlan: {
                            slots: '$$RESULT=s14 env: { s2 = Nothing
(SEARCH_META), s3 = 1688325868542 (NOW), s1 =
TimeZoneDatabase(America/Santiago...America/Sao_Paulo) (timeZoneDB) }',
                            stages: '[2] mkobj s14 [_id = s12, ventaTotal =
s13] true false \n' +
                                    '[2] project [s12 = newObj
("codigoCliente", s6, "nombreCliente", s7, "sucursal", s8), s13 =
doubleDoubleSumFinalize (s10)] \n' +
                                    '[2] group [s6, s7, s8] [s10 =
aggDoubleDoubleSum (s9)] spillSlots[s11]
mergingExprs[aggMergeDoubleDoubleSums (s11)] \n' +
                                    '[2] project [s9 = getField (s4,
"totalFactura")] \n' +
                                    '[2] project [s8 = getField (s4,
"sucursal")] \n' +
                                    '[2] project [s7 = getField (s4,
```

```
"nombreCliente")] \n' +
                                    '[2] project [s6 = getField (s4,
"codigoCliente")] \n' +
                                    '[1] scan s4 s5 none none none none []
@"a7d630ef-b286-4d32-a4d3-5f8f94e33678" true false '
                            }
                        },
                        rejectedPlans: []
                    },
                    executionStats: {
                        executionSuccess: true,
                        nReturned: 300,
                        executionTimeMillis: 1659,
                        totalKeysExamined: 0,
                        totalDocsExamined: 700000,
                        executionStages: {
                            stage: 'mkobj',
                            planNodeId: 2,
                            nReturned: 300,
                            executionTimeMillisEstimate: 1652,
                            opens: 1,
                            closes: 1,
                            saveState: 701,
                            restoreState: 701,
                            isEOF: 1,
                            objSlot: 14,
                            fields: [],
                            projectFields: [ '_id', 'ventaTotal' ],
                            projectSlots: [ Long("12"), Long("13") ],
                            forceNewObject: true,
                            returnOldObject: false,
                            inputStage: {
                                stage: 'project',
                                planNodeId: 2,
                                nReturned: 300,
                                executionTimeMillisEstimate: 1652,
                                opens: 1,
                                closes: 1,
                                saveState: 701,
                                restoreState: 701,
                                isEOF: 1,
                                projections: {
                                    '12': 'newObj ("codigoCliente", s6,
"nombreCliente", s7, "sucursal", s8) ',
                                    '13': 'doubleDoubleSumFinalize (s10) '
                                },
                                inputStage: {
                                    stage: 'group',
                                    planNodeId: 2,
                                    nReturned: 300,
                                    executionTimeMillisEstimate: 1652,
                                    opens: 1,
                                    closes: 1,
                                    saveState: 701,
                                    restoreState: 701,
                                    isEOF: 1,
                                    groupBySlots: [ Long("6"), Long("7"),
```

```
Long("8") ],
                                expressions: { '10': 'aggDoubleDoubleSum (s9)
' },
                                mergingExprs: { '11':
'aggMergeDoubleDoubleSums (s11) ' },
                                usedDisk: false,
                                numSpills: 0,
                                spilledRecords: 0,
                                spilledDataStorageSize: 0,
                                inputStage: {
                                    stage: 'project',
                                    planNodeId: 2,
                                    nReturned: 700000,
                                    executionTimeMillisEstimate: 1567,
                                    opens: 1,
                                    closes: 1,
                                    saveState: 701,
                                    restoreState: 701,
                                    isEOF: 1,
                                    projections: { '9': 'getField (s4,
"totalFactura") ' },
                                    inputStage: {
                                        stage: 'project',
                                        planNodeId: 2,
                                        nReturned: 700000,
                                        executionTimeMillisEstimate: 1541,
                                        opens: 1,
                                        closes: 1,
                                        saveState: 701,
                                        restoreState: 701,
                                        isEOF: 1,
                                        projections: { '8': 'getField (s4,
"sucursal") ' },
                                        inputStage: {
                                            stage: 'project',
                                            planNodeId: 2,
                                            nReturned: 700000,
                                            executionTimeMillisEstimate:
1520,
                                            opens: 1,
                                            closes: 1,
                                            saveState: 701,
                                            restoreState: 701,
                                            isEOF: 1,
                                            projections: { '7': 'getField
(s4, "nombreCliente") ' },
                                            inputStage: {
                                                stage: 'project',
                                                planNodeId: 2,
                                                nReturned: 700000,
                                                executionTimeMillisEstimate:
1495,
                                                opens: 1,
                                                closes: 1,
                                                saveState: 701,
                                                restoreState: 701,
                                                isEOF: 1,
```

```
                                                projections: { '6': 'getField
(s4, "codigoCliente") ' },
                                                inputStage: {
                                                    stage: 'scan',
                                                    planNodeId: 1,
                                                    nReturned: 700000,

executionTimeMillisEstimate: 1456,
                                                    opens: 1,
                                                    closes: 1,
                                                    saveState: 701,
                                                    restoreState: 701,
                                                    isEOF: 1,
                                                    numReads: 700000,
                                                    recordSlot: 4,
                                                    recordIdSlot: 5,
                                                    fields: [],
                                                    outputSlots: []
                                                }
                                            }
                                        }
                                    }
                                }
                            }
                        }
                    }
                },
                nReturned: Long("300"),
                executionTimeMillisEstimate: Long("1652")
            },
            {
                '$project': {
                    CodigoCliente: '$_id.codigoCliente',
                    NombreCliente: '$_id.nombreCliente',
                    Sucursal: '$_id.sucursal',
                    VentaTotal: '$ventaTotal',
                    _id: false
                },
                nReturned: Long("300"),
                executionTimeMillisEstimate: Long("1652")
            }
        ],
        serverInfo: {
            host: 'DESKTOP-LSTENG7',
            port: 27017,
            version: '6.0.6',
            gitVersion: '26b4851a412cc8b9b4a18cdb6cd0f9f642e06aa7'
        },
        serverParameters: {
            internalQueryFacetBufferSizeBytes: 104857600,
            internalQueryFacetMaxOutputDocSizeBytes: 104857600,
            internalLookupStageIntermediateDocumentMaxSizeBytes: 104857600,
            internalDocumentSourceGroupMaxMemoryBytes: 104857600,
            internalQueryMaxBlockingSortMemoryUsageBytes: 104857600,
            internalQueryProhibitBlockingMergeOnMongoS: 0,
            internalQueryMaxAddToSetBytes: 104857600,
```

```
        internalDocumentSourceSetWindowFieldsMaxMemoryBytes: 104857600
    },
    command: {
        aggregate: 'DocumentoFactura',
        pipeline: [
            {
                '$group': {
                    _id: {
                        codigoCliente: '$codigoCliente',
                        nombreCliente: '$nombreCliente',
                        sucursal: '$sucursal'
                    },
                    ventaTotal: { '$sum': '$totalFactura' }
                }
            },
            {
                '$project': {
                    _id: 0,
                    CodigoCliente: '$_id.codigoCliente',
                    NombreCliente: '$_id.nombreCliente',
                    Sucursal: '$_id.sucursal',
                    VentaTotal: '$ventaTotal'
                }
            }
        ],
        cursor: {},
        '$db': 'Facturacion'
    },
    ok: 1
}
```

Conclusión:

En esta consulta, el "$group" utilizará los datos para realizar una agrupación por sucursal y calcular el total. Dado que se trata de una agrupación por cliente y sucursal, se necesitan los datos correspondientes. Según la documentación, la posible acción a tomar sería no hacer nada, ya que el operador "$group" no utiliza índices al momento de agrupar los elementos. Por lo tanto, la creación de un índice no tendría ningún efecto. En conclusión, esta consulta no requiere el uso de índices, ya que cuenta con un operador de agrupación (que no emplea índices) y un operador de proyección (que simplemente toma la salida y la transforma según los nombres de los atributos deseados).