Ejercicio

Crea una BD inventario con la siguiente colección:

```
inventario> db.productos.find()
[
  {
    _id: '000001',
    genero: 'camisa',
    talla: 'L',
    precio: 100,
    existencia: [ { tienda: 'SE', cantidad: 8 }, { tienda: 'CA', cantidad: 12 } ],
    tipo: 'hombre'
  },
  {
    _id: '000011',
    genero: 'camisa',
    talla: 'M',
    precio: 50,
    existencia: [ { tienda: 'SE', cantidad: 15 }, { tienda: 'CA', cantidad: 10 } ],
    tipo: 'mujer'
  },
  {
    _id: '000002',
    genero: 'traje',
    talla: 'M',
    precio: 450,
    existencia: [ { tienda: 'SE', cantidad: 5 }, { tienda: 'CA', cantidad: 15 } ],
    tipo: 'mujer'
  },
  {
    _id: '000009',
    genero: 'pantalon',
inventario>
```

Realiza las siguientes operaciones sobre la BD:

1. Abrimos una nueva tienda “CO”, donde solo se va a vender ropa de hombre y va a comenzar con un stock (cantidad) de cada producto de 10 prendas. Actualiza la BD.

```
db.productos.updateOne({_id:"000001"},{$push:
{existencia:{tienda:"CO",cantidad:10}}})
db.productos.updateOne({_id:"000011"},{$push:
{existencia:{tienda:"CO",cantidad:10}}})
db.productos.updateOne({_id:"000002"},{$push:
{existencia:{tienda:"CO",cantidad:10}}})
db.productos.updateOne({_id:"000009"},{$push:
{existencia:{tienda:"CO",cantidad:10}}})
```

```
inventario> db.productos.find()
[
  {
    _id: '000001',
    genero: 'camisa',
    talla: 'L',
    precio: 100,
    existencia: [
      { tienda: 'SE', cantidad: 8 },
      { tienda: 'CA', cantidad: 12 },
      { tienda: 'CO', cantidad: 10 }
    ],
    tipo: 'hombre'
  },
  {
    _id: '000011',
    genero: 'camisa',
    talla: 'M',
    precio: 50,
    existencia: [
      { tienda: 'SE', cantidad: 15 },
      { tienda: 'CA', cantidad: 10 },
      { tienda: 'CO', cantidad: 10 }
    ],
    tipo: 'mujer'
  },
  {
    _id: '000002',
    genero: 'traje',
    talla: 'M',
    precio: 450,
    existencia: [
      { tienda: 'SE', cantidad: 5 },
      { tienda: 'CA', cantidad: 15 },
      { tienda: 'CO', cantidad: 10 }
    ],
    tipo: 'mujer'
  },
  {
    _id: '000009',
    genero: 'pantalon',
    talla: 'L',
    precio: 50,
    existencia: [
      { tienda: 'SE', cantidad: 100 },
      { tienda: 'CA', cantidad: 50 },
      { tienda: 'CO', cantidad: 10 }
    ],
    tipo: 'hombre'
```

2. Consulta la información de los dos productos de menor precio, muestra el sku, género y precio.

```
db.productos.find({},{genero:1, precio:1}).sort({precio:1}).limit(2)
```

```
inventario> db.productos.find({},{genero:1, precio:1}).sort({precio:1}).limit(2)
[
  { _id: '000009', genero: 'pantalon', precio: 50 },
  { _id: '000011', genero: 'camisa', precio: 50 }
]
inventario>
```

3. Mostrar los datos sku y precio de todos los productos que no se venden en la tienda “CO” o cuyo precio sea superior a 50, ordenados por precio descendente.

```
db.productos.find({$or:[{existencia:{"CO":{$exists:false}}},{precio:{$gt:50}}]},{precio:1}).sort({precio:-1})
```

```
inventario> db.productos.find({$or:[{existencia:{"CO":{$exists:false}}},{precio:{$gt:50}}]},{ precio:1}).sort({precio:-1})
[ { _id: '000002', precio: 450 }, { _id: '000001', precio: 100 } ]
inventario>
```

4. Mostrar los datos sku, genero, talla y tipo de aquellos productos que tengan un stock inferior a 10 prendas en alguna tienda y su precio sea mayor de 100, ordenados por talla y sku.

```
db.productos.find({$and:[{"existencia.cantidad":{$lt:10}},{precio:{$gt:100}}]},{genero:1,talla:1,tipo:1}).sort({talla:1, _id:1})
```

```
inventario> db.productos.find({$and:[{"existencia.cantidad":{$lt:10}},{precio:{$gt:100}}]},{genero:1,talla:1,tipo:1}).sort({talla:1, _id:1})
[ { _id: '000002', genero: 'traje', talla: 'M', tipo: 'mujer' } ]
```

Diseño

5. Añadir en la BD los datos sobre las ventas (tienda, sku, cantidad y fecha) qué diseño elegirías? razónalo y modifica la BD con un ejemplo.

Para añadir los datos de ventas, he creado una nueva colección que está relacionada con cada tienda. De esta manera cada documento está organizado por las tiendas, y en cada uno de ellos se observan los productos por su sku y la cantidad que hay disponible en esa tienda. He planteado este diseño porque considero que estos datos son más útiles para el control de cada tienda, saber qué tienen. Otra opción que me he planteado ha sido el hacer la referencia por el sku y tener un array de las tiendas con sus cantidades, pero creo que esta perspectiva es más útil cuando en el producto se quiere saber dónde y cuánta es la disponibilidad.

```
db.ventas.insertOne({tienda:"SE",
productos:[{sku:"000001",cantidad:8},{sku:"000011",cantidad:15},{sku:"0
```

```
00002",cantidad:5},{sku:"000009",cantidad:100}],fecha: new Date(2022,
11, 27)})
db.ventas.insertOne({tienda:"CA",
productos:[{sku:"000001",cantidad:12},{sku:"000011",cantidad:10},{sku:"
000002",cantidad:15},{sku:"000009",cantidad:50}],fecha: new Date(2022,
11, 27)})
db.ventas.insertOne({tienda:"CO",
productos:[{sku:"000001",cantidad:10},{sku:"000011",cantidad:10},{sku:"
000002",cantidad:10},{sku:"000009",cantidad:10}],fecha: new Date(2022,
11, 27)})
```

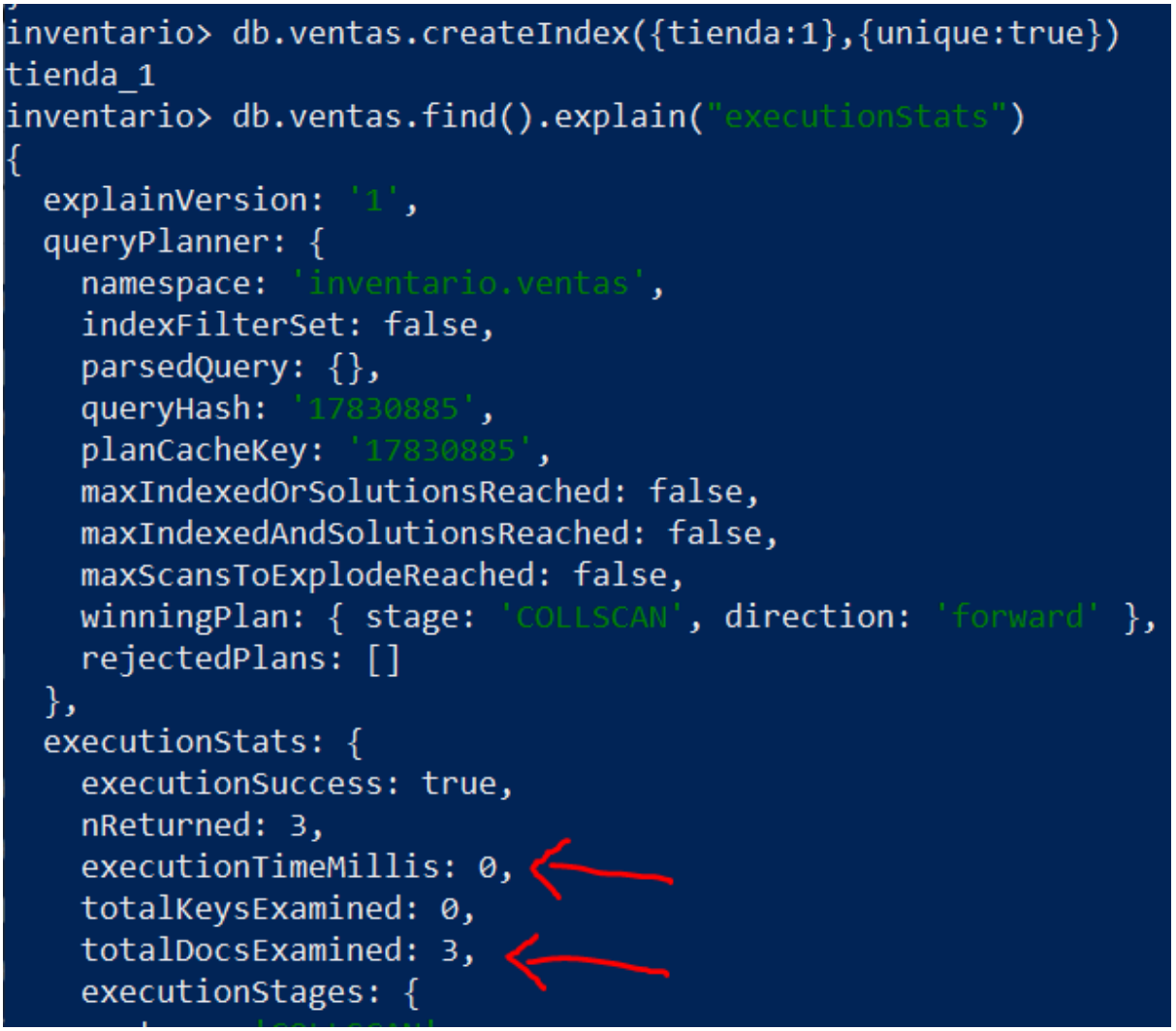
Índices

6. Qué índices crearías, suponiendo que el volumen de productos es muy grande (999999), Crea los índices en la BD y razona el motivo por el que has creado cada uno de ellos. ¿Cómo comprobarías si ha mejorado el rendimiento de la BD?.

En primer lugar, he creado un índice en la colección ventas, por el nombre de la tienda que es único, pues es la primera búsqueda que se realiza. Se observa que el tiempo de ejecución se ha reducido, aunque la cantidad de documentos examinados sea el mismo.

```
inventario> db.ventas.find().explain("executionStats")
{
  explainVersion: '1',
  queryPlanner: {
    namespace: 'inventario.ventas',
    indexFilterSet: false,
    parsedQuery: {},
    queryHash: '17830885',
    planCacheKey: '17830885',
    maxIndexedOrSolutionsReached: false,
    maxIndexedAndSolutionsReached: false,
    maxScansToExplodeReached: false,
    winningPlan: { stage: 'COLLSCAN', direction: 'forward' },
    rejectedPlans: []
  },
  executionStats: {
    executionSuccess: true,
    nReturned: 3,
    executionTimeMillis: 9,
    totalKeysExamined: 0,
    totalDocsExamined: 3,
    executionStages: {
```

```
inventario> db.ventas.createIndex({tienda:1},{unique:true})
tienda_1
```



En segundo lugar, para la colección de productos, he probado primero crea un índice sobre la talla y he consultado los de la talla L, y aunque con el índice se han examinado menos documentos, los milisegundos han salido en mayor cantidad.

```
inventario> db.productos.find().explain("executionStats")
{
  explainVersion: '1',
  queryPlanner: {
    namespace: 'inventario.productos',
    indexFilterSet: false,
    parsedQuery: {},
    queryHash: '17830885',
    planCacheKey: '17830885',
    maxIndexedOrSolutionsReached: false,
    maxIndexedAndSolutionsReached: false,
    maxScansToExplodeReached: false,
    winningPlan: { stage: 'COLLSCAN', direction: 'forward' },
    rejectedPlans: []
  },
  executionStats: {
    executionSuccess: true,
    nReturned: 4,
    executionTimeMillis: 0,
    totalKeysExamined: 0,
    totalDocsExamined: 4,
    executionStages: {
```

```
inventario> db.productos.createIndex({talla:1})
talla_1
```

```
inventario> db.productos.find({talla:"L"}).explain("executionStats")
{
  explainVersion: '1',
  queryPlanner: {
    namespace: 'inventario.productos',
    indexFilterSet: false,
    parsedQuery: { talla: { '$eq': 'L' } },
    queryHash: 'B8A0F9E4',
    planCacheKey: 'E7CB8A8B',
    maxIndexedOrSolutionsReached: false,
    maxIndexedAndSolutionsReached: false,
    maxScansToExplodeReached: false,
    winningPlan: {
      stage: 'FETCH',
      inputStage: {
        stage: 'IXSCAN',
        keyPattern: { talla: 1 },
        indexName: 'talla_1',
        isMultiKey: false,
        multiKeyPaths: { talla: [] },
        isUnique: false,
        isSparse: false,
        isPartial: false,
        indexVersion: 2,
        direction: 'forward',
        indexBounds: { talla: [ '["L", "L"]' ] }
      }
    },
    rejectedPlans: []
  },
  executionStats: {
    executionSuccess: true,
    nReturned: 2,
    executionTimeMillis: 19,
    totalKeysExamined: 2,
    totalDocsExamined: 2,
    executionStages: {
```

```
inventario> db.productos.find({talla:"L"}).explain("executionStats")
{
  explainVersion: '1',
  queryPlanner: {
    namespace: 'inventario.productos',
    indexFilterSet: false,
    parsedQuery: { talla: { '$eq': 'L' } },
    queryHash: 'B8A0F9E4',
    planCacheKey: 'B8A0F9E4',
    maxIndexedOrSolutionsReached: false,
    maxIndexedAndSolutionsReached: false,
    maxScansToExplodeReached: false,
    winningPlan: {
      stage: 'COLLSCAN',
      filter: { talla: { '$eq': 'L' } },
      direction: 'forward'
    },
    rejectedPlans: []
  },
  executionStats: {
    executionSuccess: true,
    nReturned: 2,
    executionTimeMillis: 7,
    totalKeysExamined: 0,
    totalDocsExamined: 4,
    executionStages: {
      stage: 'COLLSCAN',
      filter: { talla: { '$eq': 'L' } },
      nReturned: 2,
      executionTimeMillisEstimate: 0,
      works: 6,
      advanced: 2,
      needTime: 3,
      needYield: 0,
```

También he considerado importante tener un sistema de indexamiento por género y precio, pues son los filtros más significativos y los que se aplican en cualquier e-commerce.

```
inventario> db.productos.createIndex({genero:1})
genero_1
inventario> db.productos.createIndex({precio:1})
precio_1
inventario>
```

7. Cuántos productos hay por tipo, su precio máximo y mínimo.

```
db.productos.aggregate([{$group:{ _id:{genero:"$genero"},
"cantidad":{$sum:1}, precioMaximo:{$max:'$precio'},
precioMinimo:{$min:'$precio'}}}])
```

```
inventario> db.productos.aggregate([{$group:{ _id:{genero:"$genero"}, "cantidad":{$sum:1},
precioMaximo:{$max:'$precio'}, precioMinimo:{$min:'$precio'}}}])
[
  {
    _id: { genero: 'traje' },
    cantidad: 1,
    precioMaximo: 450,
    precioMinimo: 450
  },
  {
    _id: { genero: 'pantalon' },
    cantidad: 1,
    precioMaximo: 50,
    precioMinimo: 50
  },
  {
    _id: { genero: 'camisa' },
    cantidad: 2,
    precioMaximo: 100,
    precioMinimo: 50
  }
]
inventario>
```

8. Cuántos productos hay de cada género y su precio medio.

```
db.productos.aggregate([{$group:{ _id:{genero:"$genero"},
"cantidad":{$sum:1}, mediaPrecio:{$avg:"$precio"}}}])
```

```
inventario> db.productos.aggregate([{$group:{ _id:{genero:"$genero"}, "cantidad":{$sum:1},
mediaPrecio:{$avg:"$precio"}}}])
[
  { _id: { genero: 'traje' }, cantidad: 1, mediaPrecio: 450 },
  { _id: { genero: 'camisa' }, cantidad: 2, mediaPrecio: 75 },
  { _id: { genero: 'pantalon' }, cantidad: 1, mediaPrecio: 50 }
]
inventario>
```

9. Muestra el stock total existente (entre todas las tiendas) de cada producto.

```
db.productos.aggregate([{$unwind:"$existencia"},{$group:{
_id:{producto:"$_id"}, totalStock:{$sum:"$existencia.cantidad"}}}])
```

```
inventario> db.productos.aggregate([{$unwind:"$existencia"},{$group:{ _id:{producto:"$_id"}, totalStock:{$sum:"$existencia.cantidad"}}}])
[
  { _id: { producto: '000002' }, totalStock: 30 },
  { _id: { producto: '000009' }, totalStock: 160 },
  { _id: { producto: '000001' }, totalStock: 30 },
  { _id: { producto: '000011' }, totalStock: 35 }
]
inventario>
```

10. Visualización, de mayor a menor por tienda y producto, de la valoración de las existencias.

No entiendo el enunciado. Así que lo he interpretado de estas dos maneras.

```
db.productos.aggregate([{$group:{ _id:{producto:"$_id",
tienda:"$existencia.tienda", totalStock:"$existencia.cantidad"}}},
{$sort:{"_id.tienda":-1,"_id.producto":-1}}])
```

```
inventario> db.productos.aggregate([{$group:{ _id:{producto:"$_id", tienda:"$existencia.tienda", totalStock:"$existencia.cantidad"}}}, {$sort:{"_id.tienda":-1,"_id.producto":-1}}])
[
  {
    _id: {
      producto: '000011',
      tienda: [ 'SE', 'CA', 'CO' ],
      totalStock: [ 15, 10, 10 ]
    }
  },
  {
    _id: {
      producto: '000009',
      tienda: [ 'SE', 'CA', 'CO' ],
      totalStock: [ 100, 50, 10 ]
    }
  },
  {
    _id: {
      producto: '000002',
      tienda: [ 'SE', 'CA', 'CO' ],
      totalStock: [ 5, 15, 10 ]
    }
  },
  {
    _id: {
      producto: '000001',
      tienda: [ 'SE', 'CA', 'CO' ],
      totalStock: [ 8, 12, 10 ]
    }
  }
]
inventario>
```

```
db.productos.aggregate([{$unwind:"$existencia"},{$group:{
_id:{producto:"$_id", tienda:"$existencia.tienda"},
totalStock:{$sum:"$existencia.cantidad"}}},
{$sort:{"_id.tienda":-1,"_id.producto":-1}}])
```

```
inventario> db.productos.aggregate([{$unwind:"$existencia"},{$group:{ _id:{producto:"$_id", tienda:"$existencia.tienda"}
, totalStock:{$sum:"$existencia.cantidad"}}}, {$sort:{"_id.tienda":-1,"_id.producto":-1}}])
[
  { _id: { producto: '000011', tienda: 'SE' }, totalStock: 15 },
  { _id: { producto: '000009', tienda: 'SE' }, totalStock: 100 },
  { _id: { producto: '000002', tienda: 'SE' }, totalStock: 5 },
  { _id: { producto: '000001', tienda: 'SE' }, totalStock: 8 },
  { _id: { producto: '000011', tienda: 'CO' }, totalStock: 10 },
  { _id: { producto: '000009', tienda: 'CO' }, totalStock: 10 },
  { _id: { producto: '000002', tienda: 'CO' }, totalStock: 10 },
  { _id: { producto: '000001', tienda: 'CO' }, totalStock: 10 },
  { _id: { producto: '000011', tienda: 'CA' }, totalStock: 10 },
  { _id: { producto: '000009', tienda: 'CA' }, totalStock: 50 },
  { _id: { producto: '000002', tienda: 'CA' }, totalStock: 15 },
  { _id: { producto: '000001', tienda: 'CA' }, totalStock: 12 }
]
inventario>
```