SBD02-02

Practica Consultas en MongoDB

DNI: 39465322J

Alumno : Bieito Sousa Barreiro

Url Entrega : Consultas en MongoDB

descendentemente por número de elementos vendidos. Debería quedar como en esta captura.
        - Q12 Resolución
            - Q12 vemos la estructura del primer documento
            - Q12 output stage 1
            - Q12 output stage 2
            - Q12 output stage 3
            - Q12 output final stage 4
    - 13.- Indica la consulta que muestre las 3 tiendas que más han facturado donde se vea su nombre y la cantidad facturada.
- Anotaciones ejemplo de uso

Objetivo

• Destacar la importancia de tener un usuario para cada responsabilidad, en este caso uno para la gestión de mongoDB a nivel de sistema operativo.
• Personalizar y configurar las rutas de datos y logs (y en general el archivo de configuración) en un servidor mongoDB
• Conocer las rutas y, especialmente el puerto predeterminado.
• Interactuar con las aplicaciones a través de sus servicios para autoarrancarlos, pararlos, reiniciarlos,…
• Comprender el papel de servidor y el papel de cliente incluso en la misma máquina.
• Configurar un cliente MongoDB con IP y puerto

Entorno

install docker

```
cd ~
# instalar llave publica
wget https://download.docker.com/linux/ubuntu/gpg
# desarmarla pasar de texto plano a
sudo gpg -o /usr/share/keyrings/docker.gpg --dearmor gpg
# ver version de linux
lsb_release -a
# No LSB modules are available.
# Distributor ID: Ubuntu
# Description:    Ubuntu 20.04.6 LTS
# Release:        20.04
# Codename:       focal
sudo nano /etc/apt/sources.list.d
##### añadimos a /etc/apt/sources.list.d
deb [ arch=amd64 signed-by=/usr/share/keyrings/docker.gpg] https://download.docker.co
#####
sudo apt update # forzar a apt que lea este nuevo repo
sudo apt update && sudo apt upgrade && sudo apt install docker-ce
sudo systemctl start docker  # activar servicio
sudo systemctl status docker.service  #verificar servicio
sudo usermod -aG docker bieitostudies  # añadimos usuario bieitostudies a grupo docke
sudo usermod -aG docker bieito # añadimos usuario bieito a grupo docker
```

docker imagen mongodb

```
docker ps
docker run --name mongo -d mongodb/mongodb-community-server
docker exec -it mongo bash
# copiamos el archivo de configuración a local
docker cp mongo:/etc/mongod.conf.orig .
# editamos el archivo de configuración para que se puedan conectar todas las ips 0.0.
nano mongod.conf.orig
# # network interfaces
# net:
#   port: 27017
#   bindIp: 0.0.0.0

# paramos contenedor
docker stop mongo
# eliminamos el contenedor
docker rm mongo
# vuelvo a crear el contenedor redirigiendo los puertos
docker run -p 27017:27017 --name mongo -d mongodb/mongodb-community-server
# sobre escribo la configuración de mongo
docker cp ./mongod.conf.orig mongo:/etc/mongod.conf.orig
# actualizo el contenedor
docker stop mongo
docker start mongo
# me conecto
mongo "mongodb://localhost:27017"
```

Instrucciones de entrega

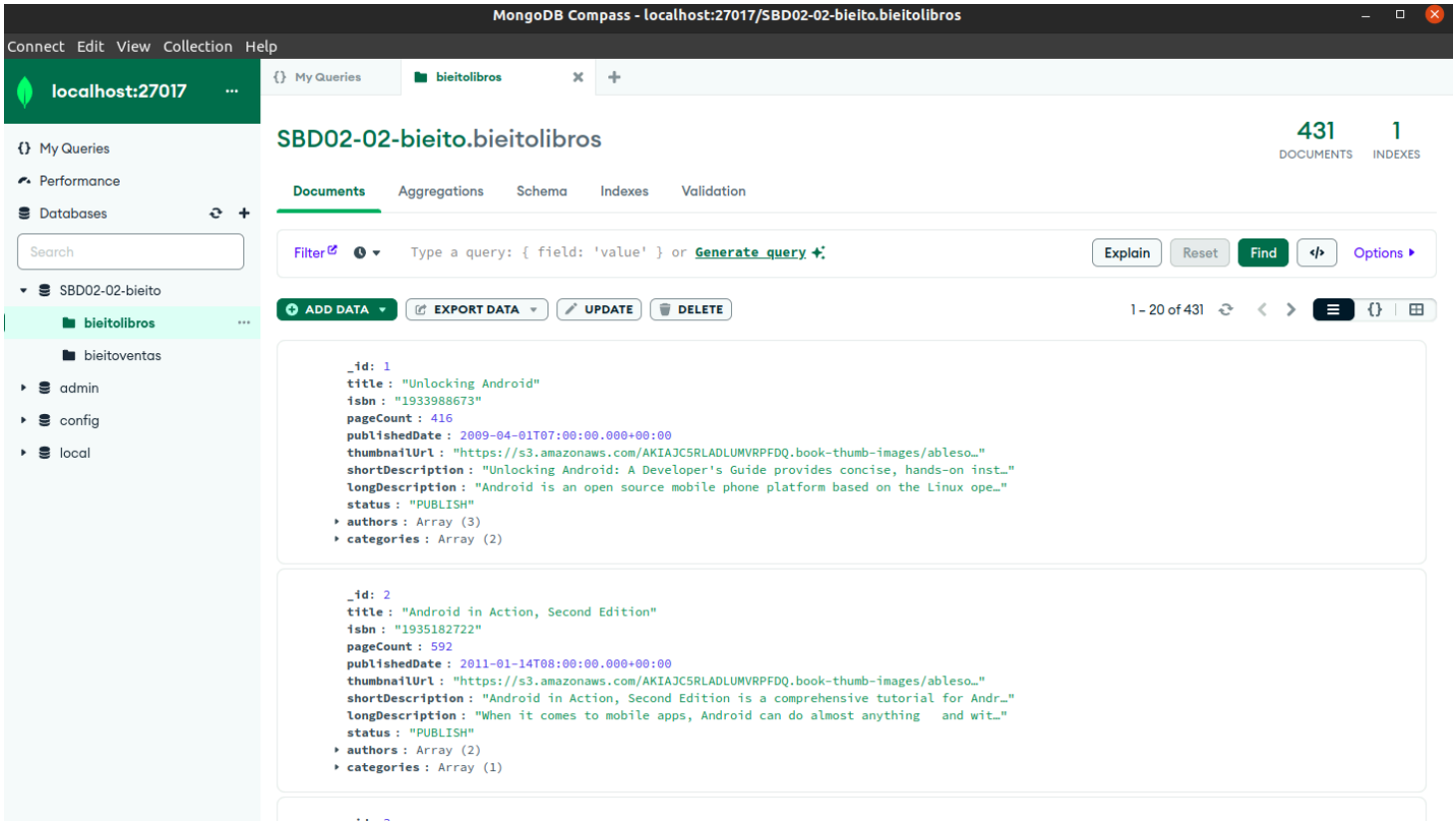
Elabora y envía un documento PDF con tu nombre completo y DNI en la portada. Usa una nueva hoja por cada apartado, copia el enunciado y contesta. Las capturas de pantalla deben ser de “pantalla completa”, no solo del detalle.

Enunciado

1.- Importa en una colección llamada “bieitolibros” el contenido del archivo books.json

```
# copiamos los archivos a /tmp
docker cp ./sales.json mongo:/tmp/sales.json
# Successfully copied 4.87MB to mongo:/tmp/sales.json
docker cp ./books.json mongo:/tmp/books.json
# Successfully copied 539kB to mongo:/tmp/books.json
# nos conectamos a un terminal
docker exec -it mongo bash
# importamos books.json
mongoimport --db=SBD02-02-bieito --collection=bieitolibros --file=/tmp/books.json
```

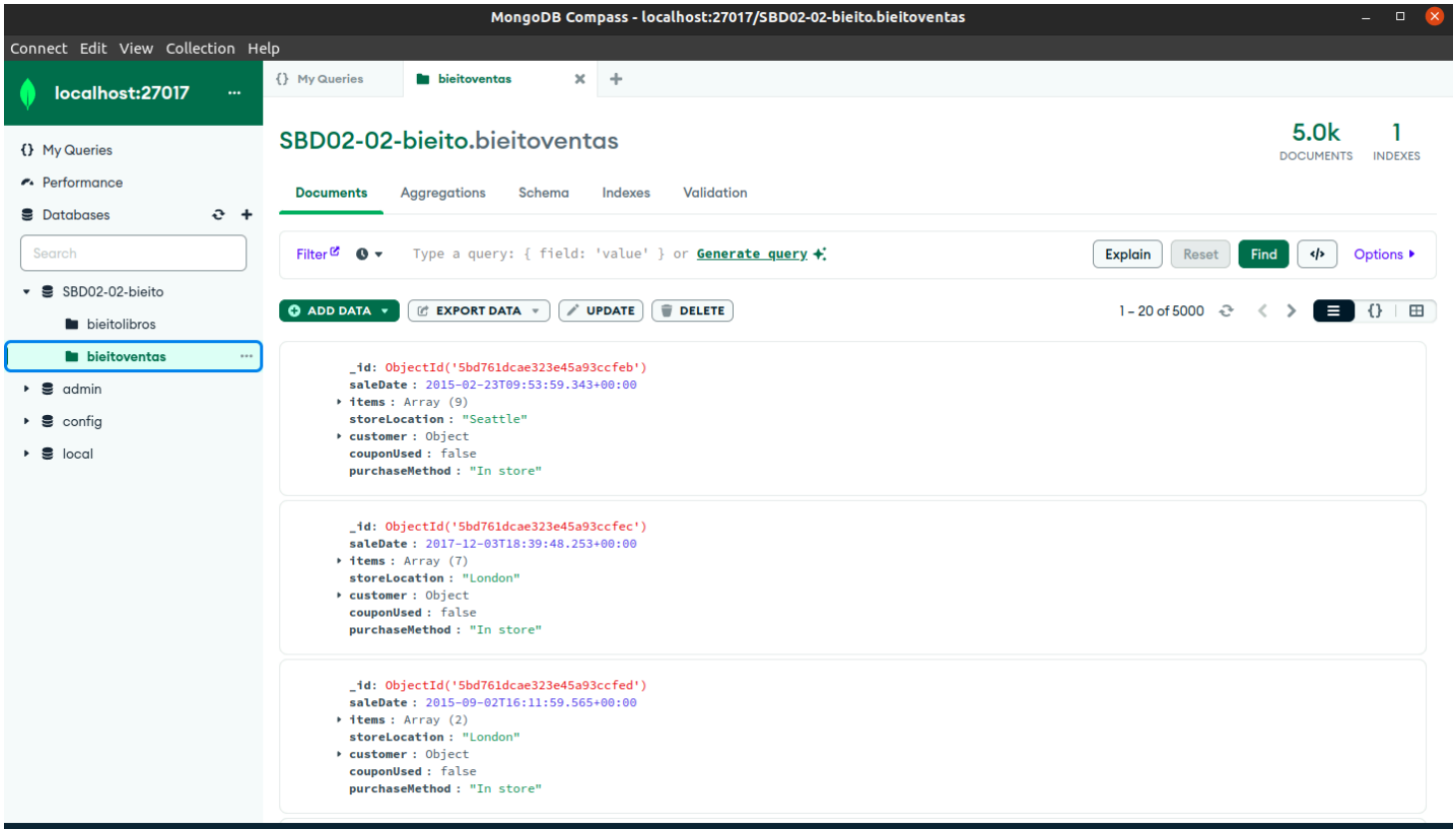
```
mongodb@627d186231c4:/$ mongoimport --db=SBD02-02-bieito --collection=bieitolibros --file=/tmp/books.json
2024-02-29T23:01:50.474+0000    connected to: mongodb://localhost/
2024-02-29T23:01:50.510+0000    431 document(s) imported successfully. 0 document(s) failed to import.
mongodb@627d186231c4:/$
```



2.- Importa en una colección llamada “bieitoventas” el contenido del archivo sales.json

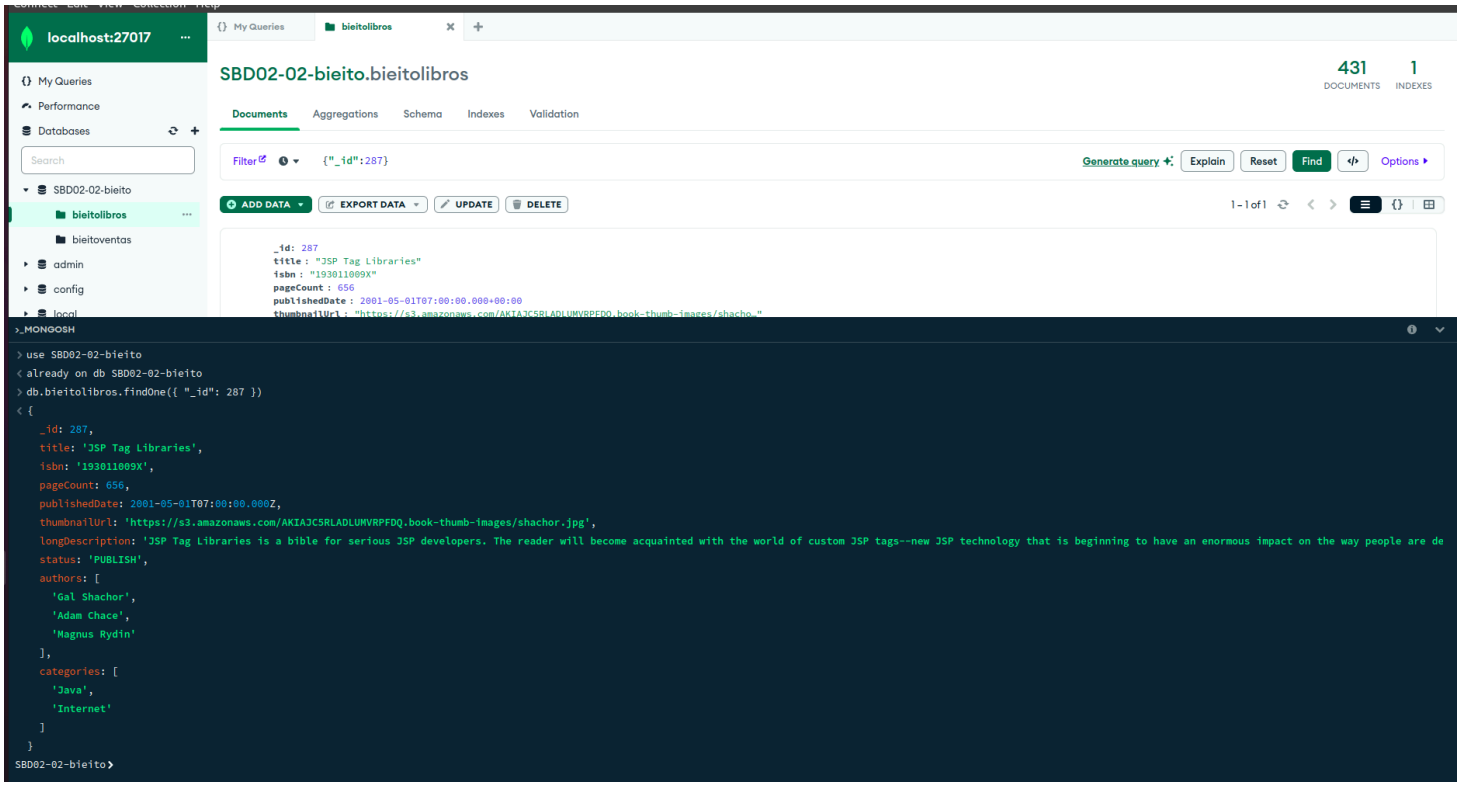
```
# importamos sales.json
mongoimport --db=SBD02-02-bieito --collection=bieitoventas --file=/tmp/sales.json
```

```
mongodb@627d186231c4:/$ mongoimport --db=SBD02-02-bieito --collection=bieitoventas --file=/tmp/sales.json
2024-02-29T23:01:37.492+0000    connected to: mongodb://localhost/
2024-02-29T23:01:37.934+0000    5000 document(s) imported successfully. 0 document(s) failed to import.
```



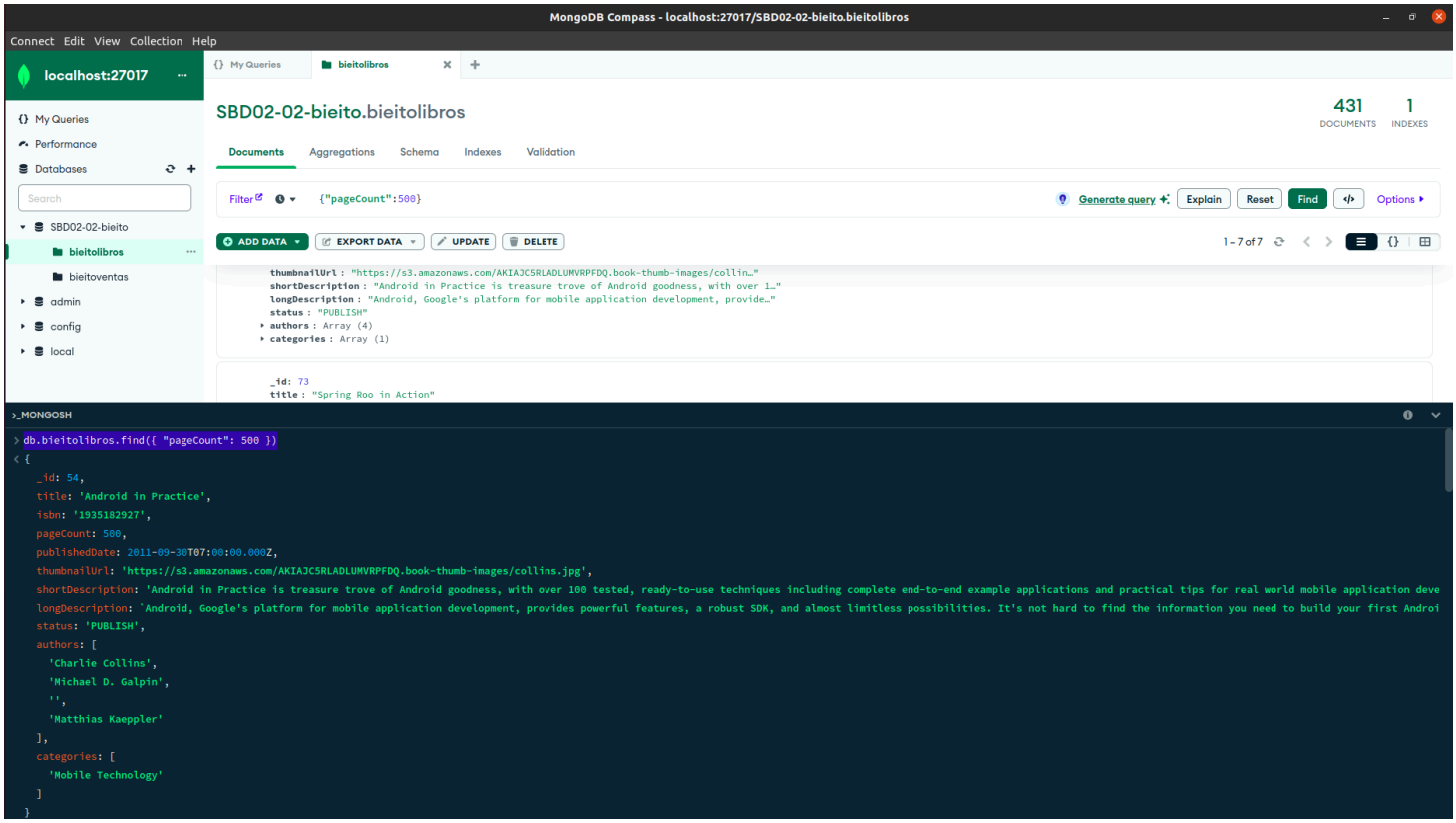
3.- Usando la colección libros, escribe la consulta que necesitas para obtener el documento con _id igual a 287.

```
// nota verificar que los id sean un string o un objeto tipo `ObjectId()`
use SBD02-02-bieito
db.bieitolibros.findOne({
    "_id": 287
 })
```



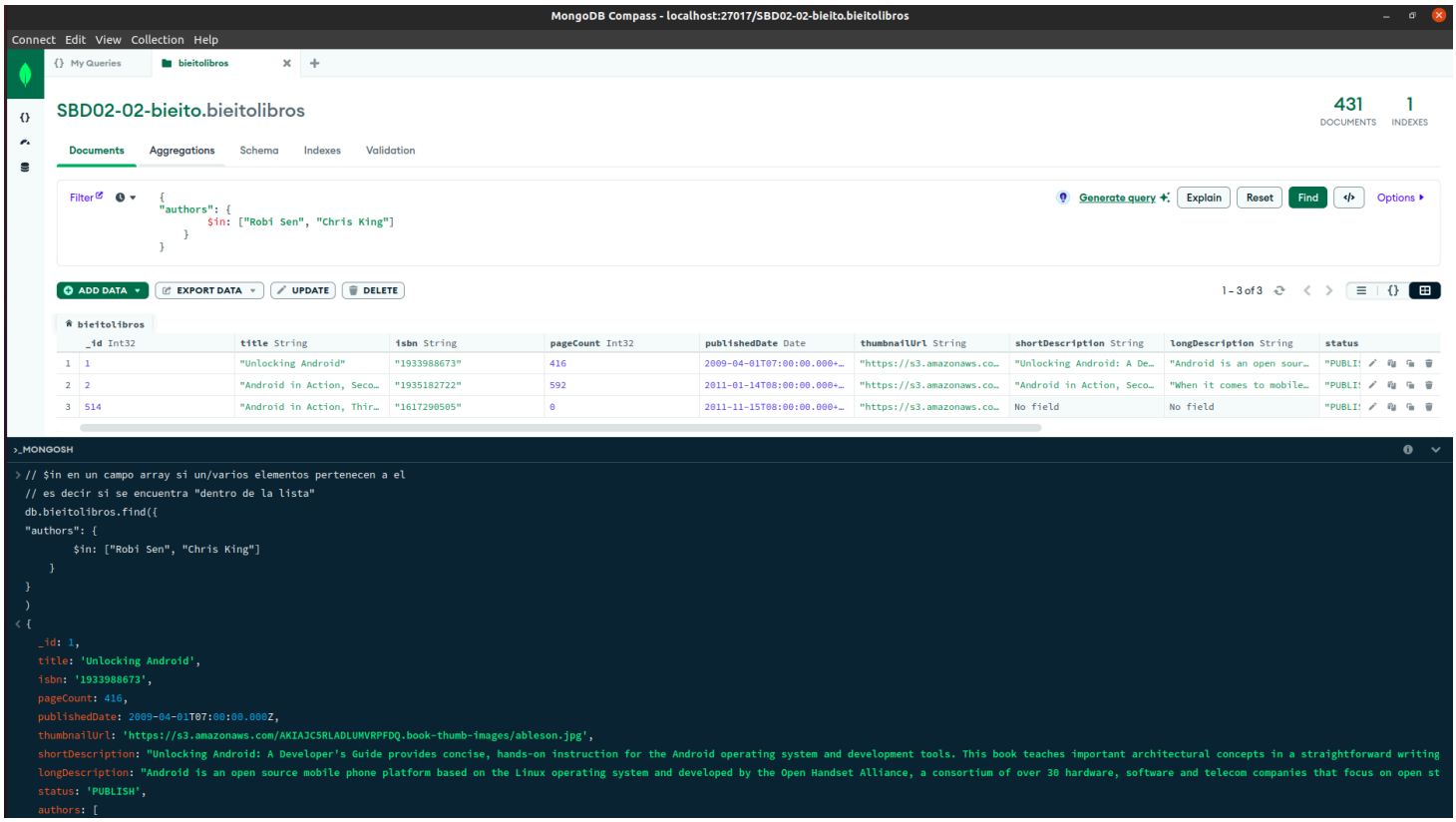
4.- Usando la colección libros, escribe una consulta que devuelva los libros con exactamente 500 páginas

```
db.bieitolibros.find({
  pageCount: 500,
});
```



5.- Usando la colección libros, escribe la consulta que devuelva los libros escritos por “Robi Sen” o por “Chris King”

```
// $in para búsquedas en u campo array
// es decir si se encuentra "dentro de la lista"
db.bieitolibros.find({
  authors: {
    $in: ["Robi Sen", "Chris King"],
  },
});
```



6.- Explica con tus palabras qué hace la siguiente consulta:

`db.javilibros.find ( { "_id" : { $gt : 10 }, "_id" : { $lt : 15 } } )`

busca en la colección `javilibros` los documentos cuyo `id` es `> 10` y `< 15`

Operadores relacionales

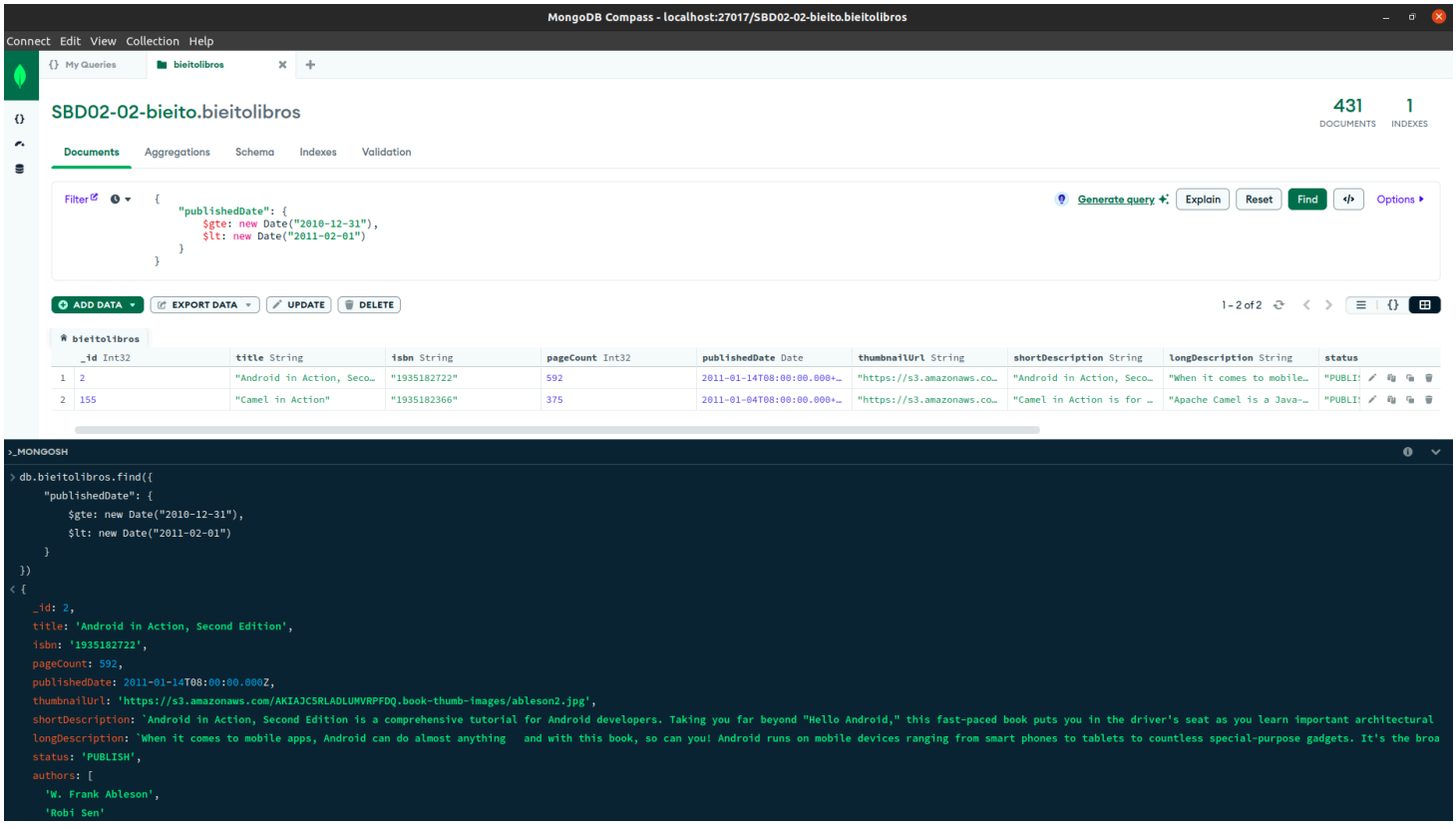
- $eq - equal - igual
- $lt - low than - menor que
- $lte - low than equal - menor o igual que
- $gt - greater than - mayor que
- $gte - greater than equal - mayor o igual que
- $ne - not equal - distinto
- $in - in - dentro de
- $nin - not in - no dentro de

```
db.bieitolibros.find({
    //#  , concatena
    "_id" : { $gt : 10 }, //# id mayor que 10
    "_id" : { $lt : 15 }  //#  id menor que 15
})
```

7.- Usando la colección libros, escribe la consulta que devuelva los libros publicados en enero del 2.011

```
// para trabajar con fechas hay que ue pasarlas a objeto date
// desde el 2010-12-31 hasta 2011-02-01 no incluidos
db.bieitolibros.find({
  publishedDate: {
    $gte: new Date("2010-12-31"),
    $lt: new Date("2011-02-01"),
  },
});


db.bieitolibros.find({
  publishedDate: {
    $gte: ISODate("2010-12-31"),
    $lt: ISODate("2011-02-01"),
  },
});
```



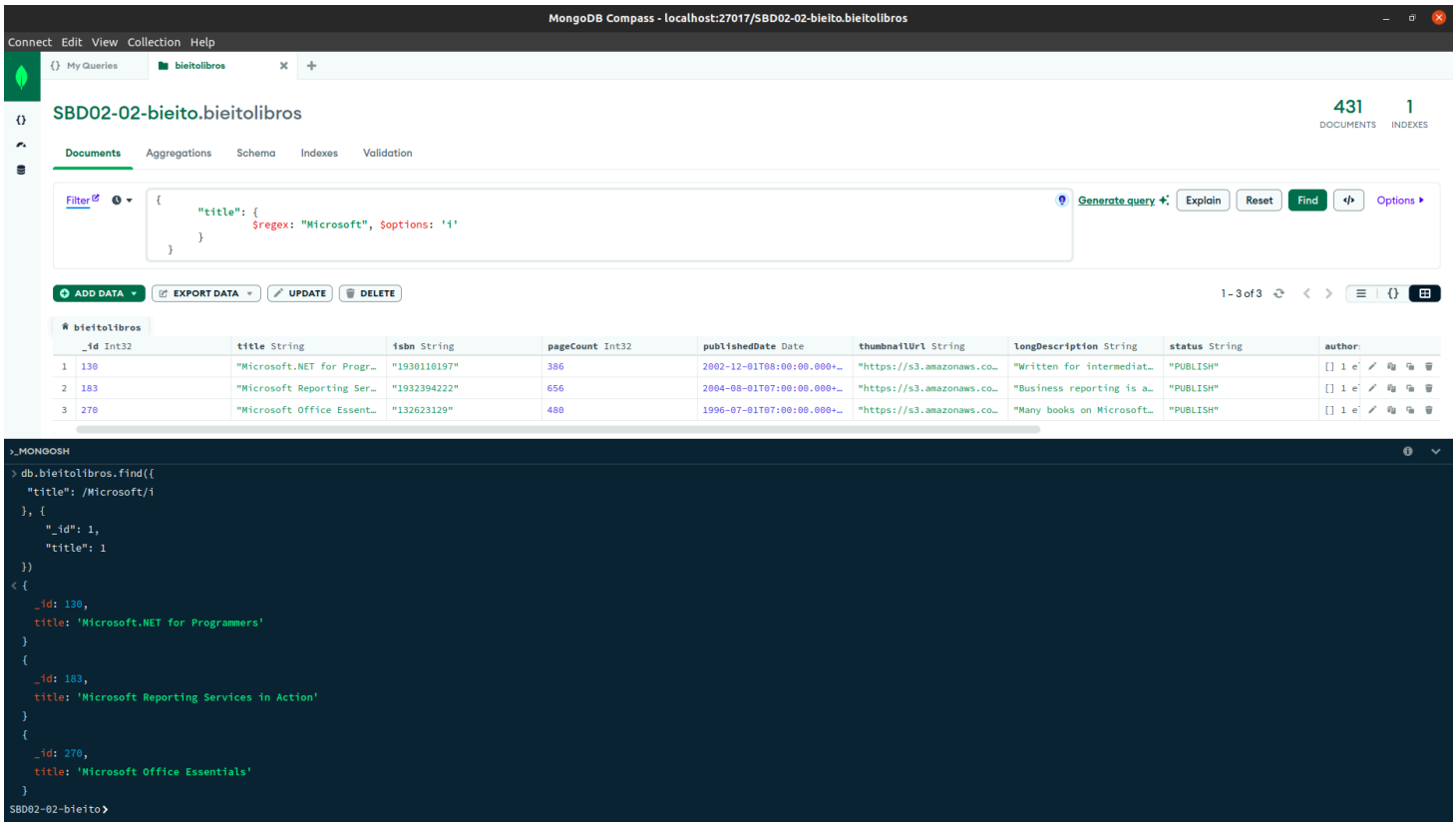
8.- Usando la colección libros, escribe la consulta que devuelve todos los libros en cuyo título podemos encontrar la palabra “Microsoft”

```
/* # options
    # i toggles case insensitivity,
    # m toggles multiline regular expression
    # x toggles an “extended” capability.
*/
db.bieitolibros.find({
  title: {
    $regex: "Microsoft",
    $options: "i",
  },
});


// otra forma de escribirlo  /pattern/
db.bieitolibros.find({
  title: "/Microsoft/i",
});
```

nota si queremos que solo muestre x cabeceras las indicamos como 2 parámetro en el find con nombre y valor 1

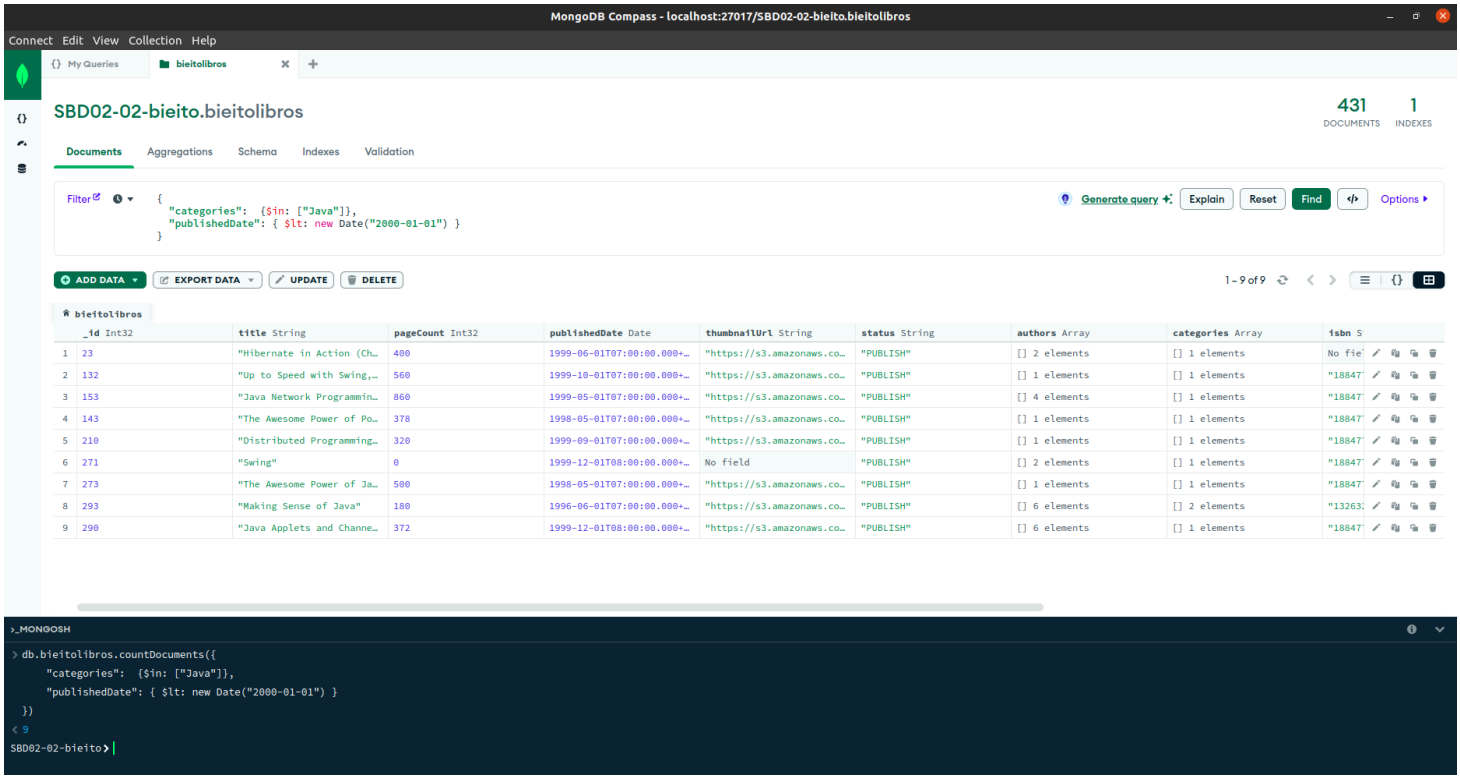
```
db.bieitolibros.find(
  {
    title: /Microsoft/i,
  },
  {
    _id: 1,
    title: 1,
  }
);
```




```
>_MONGOSH
> db.bieitolibros.find({
  "title": /Microsoft/i
}, {
    "_id": 1,
    "title": 1
})
< {
  _id: 130,
  title: 'Microsoft.NET for Programmers'
}
{
  _id: 183,
  title: 'Microsoft Reporting Services in Action'
}
{
  _id: 270,
  title: 'Microsoft Office Essentials'
}
SBD02-02-bieito>
```

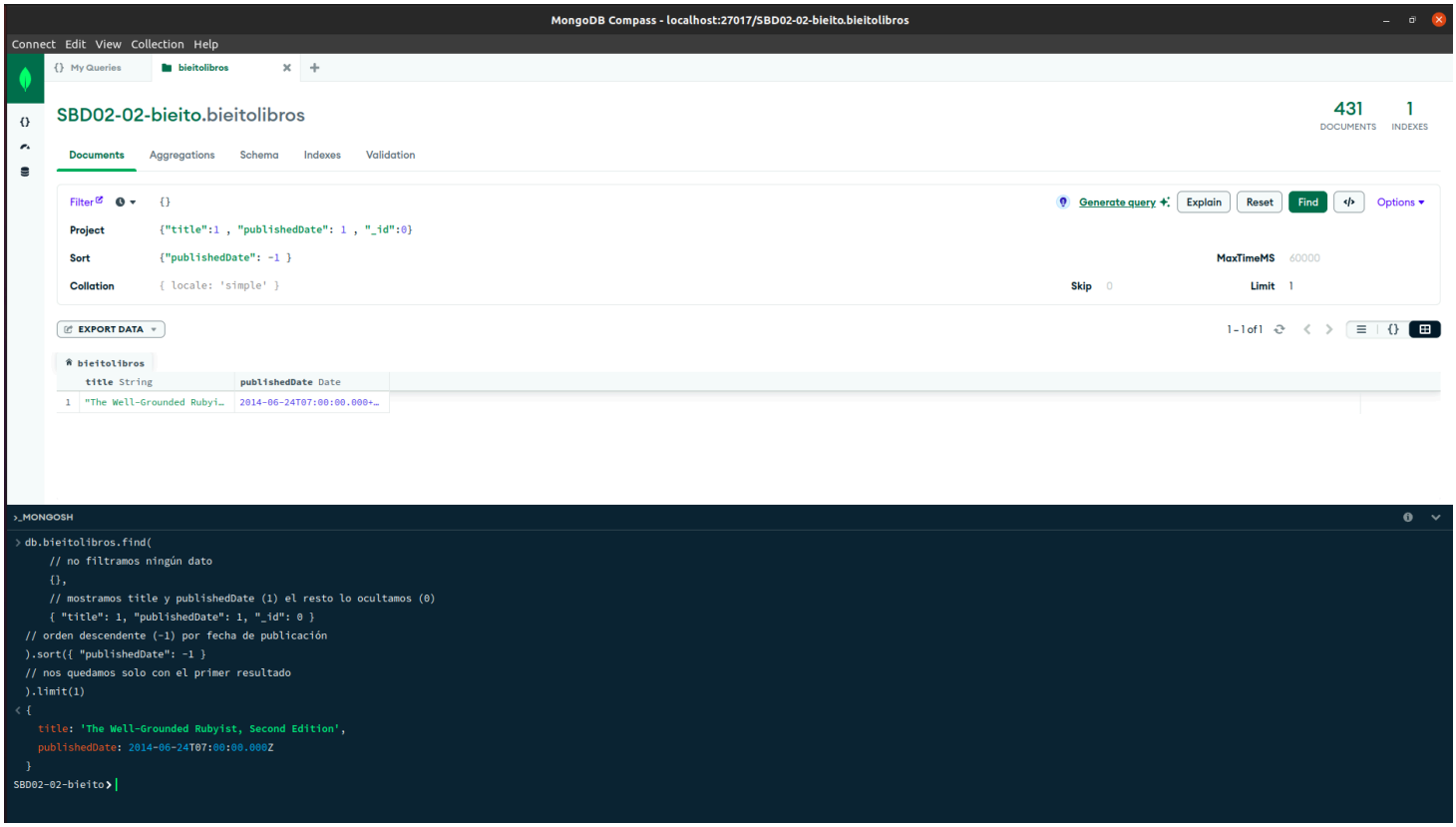
9.- Usando la colección libros, escribe una consulta que indique cuantos libros en la categoría de “Java” se han publicado antes del año 2.000.

```
db.bieitolibros.count({
    "categories":  {$in: ["Java"]}
    "publishedDate": { $lt: new Date("2000-01-01") }
})
```



10.- Usando la colección libros, escribe una consulta que indique solamente el título y la fecha de publicación del libro más recientemente publicado.

```
// find({$1},{$2})
//  - parm $1 filtros
//  - param $2 headers mostrados (1) ocultos (0)
// soft({name:value})
// - orden descendente (-1)
// - orden ascendente (+1)
db.bieitolibros
  .find(
    // no filtramos ningún dato
    {},
    // mostramos title y publishedDate (1) el resto lo ocultamos (0)
    { title: 1, publishedDate: 1, _id: 0 }
    // orden descendente (-1) por fecha de publicación
  )
  .sort(
    { publishedDate: -1 }
    // nos quedamos solo con el primer resultado
  )
  .limit(1);
```



11.- En la colección ventas, explica dónde está el problema en la segunda consulta de agregados de esta captura

indica el error de la siguiente consulta

```
use SBD02-02-bieito

db.bieitoventas.aggregate([
  { // Stage 1
    $group: { _id: "$storeLocation" },
  },
  { // Stage 2 : ordenar
    // ⚠️ tenemos de entrada la salida de Stage1
    // existe ´_id´ , ´storeLocation´ ya no existe
    $sort: { storeLocation: 1 },
  },
]);
```

Explicación

- **Se ejecuta Stage 1**

  ```
  db.bieitoventas.aggregate([
    { // Stage 1
      $group: { _id: "$storeLocation" },
    },
    [...]
  ```

- **Salida Stage 1**

  ```
  [
    { _id: "Denver" },
    { _id: "New York" },
    { _id: "San Diego" },
    { _id: "Seattle" },
    { _id: "Austin" },
    { _id: "London" },
  ];
  ```

- **Se ejecuta Stage 2**
  - recordemos que tiene como entrada el output de Stage1

```
  [...]
  { // Stage 2 : ordenar
      // ⚠️ tenemos de entrada la salida de Stage1
      // existe ´_id´ , ´storeLocation´ ya no existe
      $sort: { storeLocation: 1 },
    },
  [...]
```

- cuando el motor intenta asignar la operación de `$soft` , no puede interpretar `storeLocation` puesto que no forma parte de la salida de Stage 1

Corrección

```
db.bieitoventas.aggregate([
  {
    $group: { _id: "$storeLocation" },
  },
  {
    // para que funcione debemos ordenar por `_id`
    $sort: { _id: 1 },
  },
]);
```

```
>_MONGOSH
> db.bieitoventas.aggregate([
    {
      $group: {"_id" : "$storeLocation"}
    },
    {
      $sort:{"_id" : 1}
    }

  ])
< {
    _id: 'Austin'
  }
  {
    _id: 'Denver'
  }
  {
    _id: 'London'
  }
  {
    _id: 'New York'
  }
  {
    _id: 'San Diego'
  }
  {
    _id: 'Seattle'
  }
SBD02-02-bieito>
```

```
>_MONGOSH
> db.bieitoventas.aggregate([
    {
      $group: {"_id" : "$storeLocation"}
    },
    {
      $sort:{"_id" : -1}
    }

  ])
< {
    _id: 'Seattle'
  }
  {
    _id: 'San Diego'
  }
  {
    _id: 'New York'
  }
  {
    _id: 'London'
  }
  {
    _id: 'Denver'
  }
  {
    _id: 'Austin'
  }
SBD02-02-bieito>
```

12.- En la colección de ventas, indica la consulta que usarías para obtener un listado en el que aparezca cuantas unidades de cada elemento se han vendido en cada tienda y que aparezca ordenado alfabéticamente por tienda

y descendentemente por número de elementos vendidos. Debería quedar como en esta captura.

Q12 Resolución

```
db.bieitoventas.aggregate([
  {
    // Stage 1
    // desglosamos el documento por el array $items
    $unwind: "$items",
  },
  {
    // Stage 2: Agrupamos y campo calculado
    $group: {
      _id: {
        // agrupo por tienda y elemento
        tienda: "$storeLocation",
        elemento: "$items.name",
      },
      // campo calculado suma de unidades vendidas
      totalUnidadesVendidas: { $sum: "$items.quantity" },
    },
  },

  {
    // Stage 3: Ordenamos
    $sort: {
      // ordenación 1 : alfabéticamente (1) por tienda (_id.tienda)
      "_id.tienda": 1,
      // ordenación 2: descendentemente (-1) por número de elementos vendidos (totalU
      totalUnidadesVendidas: -1,
    },
  },
  {
    // Stage 4: Mostramos
    $project: {
      _id: 0, // no mostramos el _id
      tienda: "$_id.tienda", // renombramos $_id.tienda a tienda
      elemento: "$_id.elemento", // renombramos $_id.elemento a elemento
      totalUnidadesVendidas: 1, // mostramos unidades vendidas
    },
  },
]);
```

Q12 vemos la estructura del primer documento

```
db.bieitoventas.aggregate([{$limit:1}])
```

```
[
  {
    _id: ObjectId("5bd761dcae323e45a93ccfeb"),
    saleDate: ISODate("2015-02-23T09:53:59.343Z"),
    items: [
      {
        name: "binder",
        tags: ["school", "general", "organization"],
        price: Decimal128("20.08"),
        quantity: 1,
      },
      {
        name: "pens",
        tags: ["writing", "office", "school", "stationary"],
        price: Decimal128("23.08"),
        quantity: 4,
      },
      {
        name: "backpack",
        tags: ["school", "travel", "kids"],
        price: Decimal128("82.73"),
        quantity: 2,
      },
      {
        name: "printer paper",
        tags: ["office", "stationary"],
        price: Decimal128("15.98"),
        quantity: 3,
      },
      {
        name: "notepad",
        tags: ["office", "writing", "school"],
        price: Decimal128("27.24"),
        quantity: 4,
      },
      {
        name: "notepad",
        tags: ["office", "writing", "school"],
        price: Decimal128("27.7"),
        quantity: 5,
      },
      {
        name: "pens",
```

```
        tags: ["writing", "office", "school", "stationary"],
        price: Decimal128("59.86"),
        quantity: 5,
      },
      {
        name: "binder",
        tags: ["school", "general", "organization"],
        price: Decimal128("27.33"),
        quantity: 9,
      },
      {
        name: "notepad",
        tags: ["office", "writing", "school"],
        price: Decimal128("13.59"),
        quantity: 1,
      },
    ],
    storeLocation: "Seattle",
    customer: { gender: "F", age: 45, email: "vatires@ta.pe", satisfaction: 3 },
    couponUsed: false,
    purchaseMethod: "In store",
  },
];
```

Q12 output stage 1

desglosamos el array `items`

```
db.bieitoventas.aggregate([{$unwind: "$items"},{$limit:2}])
```

```
[
  {
    _id: ObjectId("5bd761dcae323e45a93ccfeb"),
    saleDate: ISODate("2015-02-23T09:53:59.343Z"),
    items: {
      name: "binder",
      tags: ["school", "general", "organization"],
      price: Decimal128("20.08"),
      quantity: 1,
    },
    storeLocation: "Seattle",
    customer: { gender: "F", age: 45, email: "vatires@ta.pe", satisfaction: 3 },
    couponUsed: false,
    purchaseMethod: "In store",
  },
  {
    _id: ObjectId("5bd761dcae323e45a93ccfeb"),
    saleDate: ISODate("2015-02-23T09:53:59.343Z"),
    items: {
      name: "pens",
      tags: ["writing", "office", "school", "stationary"],
      price: Decimal128("23.08"),
      quantity: 4,
    },
    storeLocation: "Seattle",
    customer: { gender: "F", age: 45, email: "vatires@ta.pe", satisfaction: 3 },
    couponUsed: false,
    purchaseMethod: "In store",
  },
];
```

Q12 output stage 2

agrupamos y creamos el campo calculado

```
db.bieitoventas.aggregate([
  {
    $unwind: "$items",
  },
  {
    $group: {
      _id: {
        tienda: "$storeLocation",
        elemento: "$items.name",
      },
      totalUnidadesVendidas: { $sum: "$items.quantity" },
    },
  },
  {
    $limit: 1,
  },
]);
```

output

```
[
  {
    _id: { tienda: "Denver", elemento: "laptop" },
    totalUnidadesVendidas: 2025,
  },
];
```

Q12 output stage 3

ordenamos

```
db.bieitoventas.aggregate([
  {
    $unwind: "$items",
  },
  {
    $group: {
      _id: {
        tienda: "$storeLocation",
        elemento: "$items.name",
      },
      totalUnidadesVendidas: { $sum: "$items.quantity" },
    },
  },
  {
    $sort: {
      "_id.tienda": 1,
      totalUnidadesVendidas: -1,
    },
  },
  {
    $limit: 1,
  },
]);
```

output

```
[
  {
    _id: { tienda: "Austin", elemento: "envelopes" },
    totalUnidadesVendidas: 3490,
  },
];
```

Q12 output final stage 4

seleccionamos los campos que queremos mostrar y los nombramos

```
db.bieitoventas.aggregate([
  {
    $unwind: "$items",
  },
  {
    $group: {
      _id: {
        tienda: "$storeLocation",
        elemento: "$items.name",
      },
      totalUnidadesVendidas: { $sum: "$items.quantity" },
    },
  },
  {
    $sort: {
      "_id.tienda": 1,
      totalUnidadesVendidas: -1,
    },
  },
  {
    $project: {
      _id: 0,
      tienda: "$_id.tienda",
      elemento: "$_id.elemento",
      totalUnidadesVendidas: 1,
    },
  },
]);
```

output

```
[
  {
    totalUnidadesVendidas: 3490,
    tienda: "Austin",
    elemento: "envelopes",
  },
  { totalUnidadesVendidas: 3331, tienda: "Austin", elemento: "binder" },
  {
    totalUnidadesVendidas: 3128,
    tienda: "Austin",
    elemento: "notepad",
  },
  { totalUnidadesVendidas: 1821, tienda: "Austin", elemento: "pens" },
  {
    totalUnidadesVendidas: 1597,
    tienda: "Austin",
    elemento: "printer paper",
  },
  { totalUnidadesVendidas: 1035, tienda: "Austin", elemento: "laptop" },
  {
    totalUnidadesVendidas: 949,
    tienda: "Austin",
    elemento: "backpack",
  },
  { totalUnidadesVendidas: 7986, tienda: "Denver", elemento: "binder" },
  {
    totalUnidadesVendidas: 7832,
    tienda: "Denver",
    elemento: "envelopes",
  },
  {
    totalUnidadesVendidas: 6276,
    tienda: "Denver",
    elemento: "notepad",
  },
  { totalUnidadesVendidas: 4188, tienda: "Denver", elemento: "pens" },
  {
    totalUnidadesVendidas: 3727,
    tienda: "Denver",
    elemento: "printer paper",
  },
  {
    totalUnidadesVendidas: 2121,
```

```
    tienda: "Denver",
    elemento: "backpack",
  },
  { totalUnidadesVendidas: 2025, tienda: "Denver", elemento: "laptop" },
  {
    totalUnidadesVendidas: 4037,
    tienda: "London",
    elemento: "envelopes",
  },
  { totalUnidadesVendidas: 3871, tienda: "London", elemento: "binder" },
  {
    totalUnidadesVendidas: 3278,
    tienda: "London",
    elemento: "notepad",
  },
  { totalUnidadesVendidas: 2232, tienda: "London", elemento: "pens" },
  {
    totalUnidadesVendidas: 2042,
    tienda: "London",
    elemento: "printer paper",
  },
  {
    totalUnidadesVendidas: 1177,
    tienda: "London",
    elemento: "backpack",
  },
];
```

13.- Indica la consulta que muestre las 3 tiendas que más han facturado donde se vea su nombre y la cantidad facturada.

Será necesaria la siguiente expresión para calcular el precio por cantidad.

```
{$multiply: [“$campo1”,”$campo2”] }
```

```
db.bieitoventas.aggregate([
  {
    // Stage 1
    // desglosamos el documento por el array $items
    $unwind: "$items",
  },
  {
    // Stage 2
    // como desglosamos es obligatorio agrupar
    // para obligar a que los id sean únicos
    $group: {
      // agrupamos por tienda
      _id: "$storeLocation",
      totalFacturado: {
        // creamos un campo calculado
        $sum: {
          // creamos un contador que acumule
          // en cada iteracción acumulamos:
          // los datos de precio por cantidad
          $multiply: ["$items.price", "$items.quantity"],
        },
      },
    },
  },
  {
    // Stage 3 : Mostramos datos
    $project: {
      tienda: "$_id", // renombramos _id a tienda
      totalFacturado: 1, // mostramos total calculado
      _id: 0, // no mostramos _id
    },
  },
  {
    // Stage 4 : Ordenamos Datos
    // como necesitamos los que más facturamos
    // ordenamos desc y mostramos los 3 últimos resultados
    $sort: {
      // ordenamos desc por totalFacturado
      totalFacturado: -1,
    },
  },
  {
    // Stage 4 : Limitamos Datos a mostrar
```

```
    $limit: 3, // mostramos los 3 primeros resultados
  },
]);
```

Anotaciones ejemplo de uso

```
// ejemplo de secuencia de uso
db.bieitoventas.aggregate([
  // nota se puede acceder a los arrays y objetos :
  // array.obj1.atrb1.
  // nota los valores se pasan por referencia `$`

  {// Stage 1
   // aplana el documento
   // `desagregamos` un array creando las copias necesarias del documento
   // siendo el elemento del array un elemento individual
   // ⚠️ hay que tener cuidado dado que la salida genera un copia de documentos
   // por tanto varios documentos van a tener el mismo id
    $unwind: {"key":"arraykey"} // desagregamos por el array
  },
  {// Stage 2
    $match: {"key":"value"} // hace lo mismo que un .find()
  },
  {// Stage 3 , como entrada ya tengo los datos filtrados
   // $group: {_id:{key:value}, ...} estructura obligatoria
    $group: {
        _id: { //nota el valor es una referencia de ahi`$`
          "key":"$value"
        },
      // si necesitamos `acumular` valores hacemos un `count de 1`
      // es decir cada vez que se itere el valor por el cual agrupamos
      // incrementa en 1 el contador
      countkey: {$sum:1}
      // campo calculados
      // atribuimos a un campo un valor calculado
      // la secuencia depende del operador y los argumentos que necesite
      // si necesitamos varios argumentos se declaran dentro de un array
      // no olvidar que si hay etapas anteriores los datos son por referencia `$`
      key1: {{ <operator>: [ <argument1>, <argument2> ... ] }},
      key2: {{ <operator>:  <argument1> }},

   }
  },
  {// Stage 4 ,
    $project{
      // capos que quiero ver
      // 1 visible
```

```
      // 0 oculto
        "key": 1 | 0
    }
  },

  {// Stage 5
    // limita el numero de resultados
    $limit: 3
  }
])
```