Tuần 1-2-3: MongoDB Query Exercises

Bài 1 - Tạo Database {import/export}

Yêu cầu:

Tạo file MSWord theo định dạng ***STT_MSSV_Hoten_Tuan2***, lưu toàn bộ các câu lệnh, chụp màn hình kết quả(có thể) vào file MSWord và nộp lại vào cuối buổi TH

Mục tiêu:

- *SV tạo được tài khoản Atlas trên Cloud MongoDB*
- *Cài đặt MongoDB Server trên Windows máy tính cá nhân*
- *Cài đặt Mongo Compass trên máy tính cá nhân để thực thi database bằng UI (User Interface)*
- *Cài đặt thêm Mongo tools thực hiện việc import và export database.*
- *Thực hành trên Mongo hoặc Mongosh thực thi các lệnh xem databases, collections*
- *Thực hiện các câu truy vấn vào database trên MongoDB và thực thi các lệnh trên Collections như: thêm, xóa, cập nhật documents trong collection (CRUD)*

Lý thuyết:

Thực hiện các bài tập, ví dụ trong slide chương 2

Bài 1.1: restore/backup database MongoDB

Dữ liệu đưa lên MongoDB ở 2 định dạng Bson/Json

BSon:

1. Restore:

```
mongorestore –drop –nsInclude=<dbName.coName> [des-folder/]>
     hay
mongorestore –drop –d <dbName> –c <coName> [des-
```

Ví dụ:

1. `mongorestore –drop dump/`
   → *Xóa và restore tất cả các collections của các database có trong thư mục dump*
2. `mongorestore –drop –nsInclude=test.* dump/`
   →Xóa và restore toàn bộ collection trong database test
3. `mongorestore –drop –nsInclude=test.purchaseorders dump/`
   hay
   `mongorestore –drop –db=test –collection=purchaseorders dump/test/purchaseorders.bson`
   `hay`

```
mongorestore --drop -d test -c purchaseorders
dump/test/purchaseorders.bson
```

→ *Xóa và restore collections* `purchaseorders` *của database* `test` *có trong thư mục* `dump`

2. Backup:

```
mongodump  --db <dbName> -o [des-folder/]>
```

hay

```
mongodump  --db <dbName> --colections <colectionName> -o [des-
folder/]>
```

Ví dụ:

1. `mongodump --db Test`
   → *Backup databaseTest ra thư mục hiện hành*
2. `mongodump --db Test -c dsSinhvien -o sinhvien.json`
   → *Backup collection dsSinhvien ra thư mục hiện hành là file sinhvien.json*

JSon:

1. Export:

```
mongoexport --db=<dbName> --collection=<coName>  --out=[file.json]
```

hay

```
mongoexport -d <dbName> -c <coName> -o [file.json]
```

2. Import:

```
mongoimport --drop -db=<dbName> --collection=<coName> --
file=[file.json]
```

hay

```
mongoimport --drop -d <dbName> -c <coName> [file.json]
```

Một số lệnh cơ bản

`use {database_name}: chuyển sang database_name (nếu không có tạo mới)`

`show dbs: hiển thị toàn bộ database đã có`

`show collections: hiển thị toàn bộ collections của database hiện hành`

`db.{collection_name}.find(): hiển thị toàn bộ document có trong collection_name`

`db.{collection_name}.find(projection): hiển thị document có trong collection_name theo tham số của projection`

`db.{collection_name}.find().limit(n): hiển thị n document`

`db.{collection_name}.find().skip(n):hiển thị các document bỏ qua document thứ n`

…

Mô tả database QLSinhvien gồm 2 collection sinhvien, lophoc tương ứng file json *sinhvien.json, lophoc.json* chứa các document với danh sách sinh viên và lớp học

sinhvien.json	***Lophoc.json***
{ _id: ObjectId("620a8bbb2c96dd44ef22230a"), ten: 'Nở', tuoi: 22, diem: 9, monHoc: ['Toan', 'Ngu Van', 'Tin Hoc'], totNghiep: null, lienLac: { email: 'no@gmail.com', phone: '0999.987.222' } }	{ _id: ObjectId("620a8bff7c3b0f4af1d79d98"), Name: 'DHKTPM14', Subject: 'Programming Application', Hours: 5 }

1. Thực hiện restore database QLSinhvien lên server Mongodb
2. Thực hiện các câu truy vấn các document trong collection sinhvien, lophoc như sau: *(Xem kết quả và chụp lại kết quả, nêu ý nghĩa câu query)*

```
db.sinhvien.find()
db.lophoc.find()
db.sinhvien.find().count()
db.lophoc.find().count()
db.sinhvien.find().limit(2)
db.lophoc.find().skip(2)
db.sinhvien.find().skip(2).limit(2)
db.sinhvien.find({"ten":"Tí"})
db.sinhvien.find({"tuoi":{$gt:20}})
db.sinhvien.find({"lienLac.email":"ti@gmail.com"})
db.sinhvien.find({"monHoc":"Tin Hoc"})
db.lophoc.find({"Name":"DHKTPM16"})
```

Bài 1.2: Bài tập các câu Query

Sử dụng database collection **restaurants.json** thực hiện một số câu truy vấn sau:

restaurants.json

```
{
        address: {
        building: '7715',
        coord: [ -73.9973325, 40.61174889999999 ],
        street: '18 Avenue',
        zipcode: '11214' },
        borough: 'Brooklyn',
        cuisine: 'American ',
        grades:
        [
                {date: ISODate("2014-04-16T00:00:00.000Z"),grade: 'A',score: 5},
                {date: ISODate("2013-04-23T00:00:00.000Z"),grade: 'A', score: 2},
                { date: ISODate("2012-04-24T00:00:00.000Z"),grade: 'A', score: 5},
                {date: ISODate("2011-12-16T00:00:00.000Z"), grade: 'A',score: 2}
        ],
        name: 'C & C Catering Service',
        restaurant_id: '40357437'
}
```

1. Hiển thị tất cả các documents có trong collection restaurants.
2. Hiển thị tất cả các documents có trong collection restaurants, tuy nhiên chỉ xuất các fields *restaurant_id, name, borough and cuisine*
3. Hiển thị tất cả các documents có trong collection restaurants, tuy nhiên chỉ xuất các fields *restaurant_id, name, borough and cuisine* và không xuất field *_id.*
4. Hiển thị tất cả các documents có trong collection restaurants, tuy nhiên chỉ xuất các fields restaurant_id, name, borough and zip code và không xuất field *_id.*
5. Hiển thị tất cả các documents có trong collection restaurants với field borough có giá trị là Bronx.
6. Hiển thị 5 documents đầu tiên có trong collection restaurants với field borough có giá trị là Bronx.
7. Hiển thị 5 documents tiếp theo sau khi bỏ qua 5 documents đầu tiên có trong collection restaurants với field borough có giá trị là Bronx.
8. Hiển thị tất cả các documents có trong collection restaurants với điều kiện score trong field grades lớn hơn 90.
9. Hiển thị tất cả các documents có trong collection restaurants với điều kiện score trong field grades lớn hơn 80 và nhỏ hơn 100.
10. Hiển thị tất cả các documents có trong collection restaurants với điều kiện giá trị coord trong field address nhỏ hơn -95.754168.

11. Hiển thị tất cả các documents có trong collection restaurants với điều kiện field cuisine không là 'American' và score của field grade lớn hơn 70 và giá trị coord trong field address nhỏ hơn -65.754168. ******
12. Hiển thị tất cả các documents có trong collection restaurants với điều kiện field cuisine không là 'American' và score của field grade lớn hơn 70 và giá trị coord trong field address nhỏ hơn -65.754168.
13. Hiển thị tất cả các documents có trong collection restaurants với điều kiện field cuisine không là 'American' và giá trị grade của field grade là 'A', và fiels borough không là Brooklyn. Sau đó sắp xếp các document theo thứ tự tăng dần của field cuisine.

14. Hiển thị tất cả các documents có trong collection restaurants, tuy nhiên chỉ xuất các fields restaurant Id, name, borough, cuisine với name có chứa 3 ký tự bắt đầu là 'Wil'.

15. Hiển thị tất cả các documents có trong collection restaurants, tuy nhiên chỉ xuất các fields restaurant Id, name, borough, cuisine với name có chứa 3 ký tự cuối cùng là 'ces'.

16. Hiển thị tất cả các documents có trong collection restaurants, tuy nhiên chỉ xuất các fields restaurant Id, name, borough, cuisine với name có chứa 3 ký tự 'Reg'.

17. Hiển thị tất cả các documents có trong collection restaurants với field borough có giá trị là Bronx và field cusine có giá trị là American hoặc Chinese.

18. Hiển thị tất cả các documents có trong collection restaurants với field borough có giá trị Staten Island or Queens or Bronxor Brooklyn, chỉ xuất các field restaurant Id, name, borough, cuisine

19. Hiển thị tất cả các documents có trong collection restaurants với field borough có không là các giá trị Staten Island or Queens or Bronxor Brooklyn, chỉ xuất các field restaurant Id, name, borough, cuisine.

20. Hiển thị tất cả các documents có trong collection restaurants với giá trị score của field grades không lớn hơn 10, chỉ xuất các field restaurant Id, name, borough, cuisine.

Bài 2: MongoDB Query Exercises:

Lý thuyết:

1. Tạo database:

```
use [database]
```

2. Thêm document

```
db.collection.insertOne([{filtr}])
db.collection.insertMany([{filter }])
db.collection.insert([{filter }])
```

3. Xóa document

```
db.collection.deleteOne({filter }) : xóa 1 document theo điều kiện.
db.collection.deleteMany({filter }): xóa tất cả
```

4. Cập nhật document

```
db.collection.updateOne({filter}): cập nhật 1 document theo điều kiện
fileter
db.collection.updateMany({filter}): cập nhật 1 document theo điều kiện
fileter
```

Bài 2.1: Tạo database dbQLXe với collection Xe theo cấu trúc

Thông tin mỗi xe gồm có: mã xe, tên, năm sản xuất, giá, hình ảnh, loại xe. Trong đó loại xe chứa thông tin về mã loại và tên loại.

Xe.json

```
{
  ma: 'Ya001',
  ten: 'Yamaha',
  namsx: 1992,
  gia: 12000000,
  hinhanh: 'h2.jpg',
  loai:
       { maloai: '001Xemay', tenloai: 'Xe máy' }
 }
```

{Sử dụng file dữ liệu QLXe.txt}

Thực hiện các câu truy vấn trên collections Xe

1. Xuất toàn bộ danh sách xe máy.
2. Xuất ra danh sách xe là ô tô
3. Xuất danh sách xe máy Yamaha
4. Xuất danh sách xe máy Honda
5. Xuất danh sách xe máy Yamaha sản xuất năm 1992
6. Xuất danh sách xe máy Honda có giá từ 15.000.000 trở lên
7. Xuất ra các danh sách các xe có Namsx là 1999
8. Xuất ra danh sách xe ô tô có sản xuất từ năm 2000
9. Đếm có bao nhiêu xe có năm sản xuất 1992
10. Xuất ra các xe có giá trên 25000000 dưới 655000000
11. Cập nhật mã loại của ô tô thành '001CAR'
12. Cập nhật giá của tất cả giá xe máy tăng thêm 1000000
13. Cập nhật giá của tất cả giá ô tô lên 0.5 lần
14. Cập nhật các document có "ten":"Honda" với các Namsx:2000
15. Thêm 1 thuộc tính màu xe có các giá trị đỏ, đen, trắng cho tất cả xe ô tô

16. Xuất ra tên và giá của tất cả xe ‘Yamaha’

Bài 2.2: Query *(dữ liệu mẫu sinhviendb3)*

Thông tin sinh viên gồm: Mã số sinh viên, họ, tên, giới tính, ngày sinh, email, các số điện thoại và điểm trung bình.

sinhvien

```
{
      _id: ObjectId("6131a2a278b3264add832588"),
       diemTB: 9.5,
      dsDienthoai: [ '0947536844' ],
       email: 'Chuot@gmail.com',
      gioitinh: 'Nam',
      ho: 'Nguyễn Việt',
      malop: 'DHKTPM15B',
      mssv: '19296011',
      ngaysinh: ISODate("2001-09-15T00:00:00.000Z"),
      ten: 'Anh'
}
```

Thông tin lớp học gồm: Mã lớp, tên lớp và sĩ số dự kiến.

lop

```
{
    _id: ObjectId("6131e5e97cf137f22a235cd0"),
    macn: 'KHMT',
    mslop: 'DHKHMT15A',
    sisoDukien: 80,
    tenlop: 'Đại học Khoa học Máy tính 15A'
  }
```

Thực hiện các câu truy vấn sau:

1. Thêm 1 document, thêm nhiều document: insertOne, insertMany.
2. Cập nhật giá trị thuộc tính của một hoặc nhiều document: updateOne, updateMany.
3. Tìm kiếm và thay thế: findOneAndUpdate.
4. Xóa một hoặc nhiều document: deleteOne, deleteMany, findOneAndDelete
5. Thêm/xóa thuộc tính của document.
6. Tìm kiếm theo mã: Tìm lớp khi biết mã lớp, tìm sinh viên khi biết mã sinh viên…
7. Xử lý kết quả tìm kiếm find với: count, size, limit, skip, explain, sort…
8. Đếm số sinh viên có điểm trung bình từ 5 trở lên.
9. Đếm số sinh viên có điểm trung bình từ 3.0 đến nhỏ hơn 6.5.
10. Liệt kê danh sách sinh viên không có số điện thoại hoặc email.
11. Liệt kê danh sách sinh viên có từ 2 số điện thoại trở lên.
12. Liệt kê danh sách sinh viên có lót chữ “Văn”, không phân biệt chữ hoa chữ thường.
13. Liệt kê danh sách sinh viên có họ tên chứa chữ “Minh”, không phân biệt chữ hoa chữ thường.
14. Thêm/xóa/cập nhật số điện thoại cho sinh viên khi biết mã số sinh viên.
15. Tính tỷ lệ phần trăm số lượng sinh viên theo năm sinh. ***
16. Đếm số sinh viên thực theo từng lớp.
17. Đếm số sinh viên thực theo từng lớp. Thông tin bao gồm thông tin của lớp học và tổng số **sinh viên thực.**

18. Tìm lớp học có tổng số sinh viên thực tế cao nhất. Thông tin bao gồm thông tin của lớp học và tổng số sinh viên. [{ _id: 'DHKTPM15B', soluong: 59 }]
19. Đếm số sinh viên có điểm trung bình từ 9.0 trở lên theo từng lớp, sắp xếp theo tên sinh viên. Thông tin bao gồm thông tin của lớp học và tổng số sinh viên.
20. Xuất danh sách sinh viên có điểm trung bình từ 9.0 trở lên theo từng lớp, sắp xếp theo tên sinh viên ra collection riêng biệt. Thông tin bao gồm thông tin của sinh viên.
21. Tính tổng số sinh viên theo từng chuyên ngành. Thông tin bao gồm thông tin của chuyên ngành và tổng số sinh viên.

Bài 2.3: Insert/Update/Delete Document.

Cú pháp:

```
db.collection.insertMany([{filter }])
db.collection.update({<filter>},{<update>},{<optional>})
db.collection.updateOne({<filter>},{<update>},{<optional>})
db.collection.updateMany({<filter>},{<update>},{<optional>})
```

Sử dụng collection Xe trong database dbQLXe của bài 2.1 thực hiện các câu sau:

1. Thêm vào 3 document với các thuộc tính là xe của sv *(tùy ý).*
2. Tăng giá của tất cả xe Yamaha lên 1200000.
3. Cập nhật xe máy Honda có giá thấp nhất 24000000
4. Cập nhật giá xe ô tô Ford có giá cao nhất 700000000.
5. Cập nhật thêm thuộc tính số lượng là 12 cho xe yamaha
6. Cập nhật số lượng của xe Ya004 số lượng tăng gấp 3 lần.
7. Cập nhật giá tăng 20% cho xe máy Honda sản xuất từ năm 1990 cho đến nay.
8. Tất cả xe cập nhật thêm thuộc tính mauxe(black,white,gold).
9. Tất cả xe Ford có năm sản xuất từ trước năm 2000 được cập nhật ngày bán là ngày hiện hành.
10. Đổi tất cả tên thuộc tính của tất cả các document trong collection Xe sang English *(Ví dụ: ten→name,namsx→year, loai →categories, …)*
11. Các xe Ford được cập nhật thêm thuộc tính như sau *soluong(20,30,50)*

Bài 2.4: Insert/Update/Delete Document

restaurants

```
{
    address: {  building: '97-22',  coord: [ -73.8601152, 40.7311739 ], street: '63
Road', zipcode: '11374'},
    borough: 'Queens',
    cuisine: 'Jewish/Kosher',
    grades: [ {date: ISODate("2014-11-24T00:00:00.000Z"), grade: 'Z', score: 20},
                   { date: ISODate("2013-01-17T00:00:00.000Z"), grade: 'A', score: 13},
                   {date: ISODate("2012-08-02T00:00:00.000Z"), grade: 'A', score: 13},
                  {date: ISODate("2011-12-15T00:00:00.000Z"), grade: 'B', score: 25}],
    name: 'Tov Kosher Kitchen',
    restaurant_id: '40356068'
}
```

zipcode

```
{
    _id: '01012',
    city: 'CHESTERFIELD',
    loc: [ -72.833309, 42.38167 ],
    pop: 177,
    state: 'MA'
  }
```

1. Thêm 2 documents restaurants với thông tin như sau:
   db.restaurants.insertMany(

```
[{
    address: {street: '63 Ky Dong', zipcode: '11374'},
    cuisine: 'Vietnamese',
    name: 'Lobster Bay',
    restaurant_id: '40356666',
    like:23000,
    share:4000
},
{
    address: {street: '73rd Floor Landmark 81', zipcode: '11374'},
    cuisine: 'Vietnamese',
    name: 'Truffle',
    restaurant_id: '40357777',
    like:29000,
    share:3000
}])
```

   Operators:

- Fields: $currentDate,$inc,$min,$max,mul,$rename,$set,$unset
- Array: $, $[],$[<identifier>],$each,

2. Cập nhật thuộc tính assigndate là ngày hiện hành cho các document có zipcode:’11374’. {$currentDate}
3. Cập nhật giá trị ‘Euro’ cho thuộc tính cusine của các document có thuộc tính borough là 'Queens'. {$set}
4. Tăng lượt like lên 100, lượt share lên 200 cho các nhà hàng Vietnamese. {$inc}
5. Cập nhật điểm đánh giá cao nhất là 28 cho tất cả các documnent có zipcode là ‘10002’
6. Trong document zipcode, cập nhật dân số (pop) tăng 10% của các city HADLEY thuộc state MA. {$mul}

Tuần 4 – 5 – 6 – 7 : Aggregation

Mục tiêu:
- *Hiểu rõ về quy trình (stage) trong nội dung aggregation pipelines*
- *Thực hiện các stage match, project, group,…*
- *Hiểu rõ về các accumulators và expression trong mỗi stage*

Cú pháp: Aggregation

```
db.collectionname.agrregate([
                                    {<Stage 1>},
                                    {<Stage 2>},
                                    ……
                                    {<Stage n>},
                                ])
```

$match:

Dùng để lọc các document theo một số điều kiện.

Link tham khảo
https://www.mongodb.com/docs/manual/reference/operator/aggregation/match/

```
db.collectionname.aggregate([{$match:{<expression>}}])
```

$project:

Cho định nghĩa các thuộc tính cần lấy từ collection ban đầu.

```
db.collectionname.aggregate([{$project:{'field1':1,'field':0,…}}])
```

$group:

Cho phép gom nhóm các document theo các field của collection ban đầu.

```
db.collectionname.aggregate([{$group:{_id:'$field',
accumulator:<expression>}}
])
```

Bài 1: Tạo database ***dbtuan5*** import collection ***customers.json***, tìm hiểu ý nghĩa của document sau đó thực hiện yêu cầu:

Import dữ liệu từ file Customers.json vào CSDL. Và sử dụng aggregate để thực hiện các truy vấn sau.

1. Xuất danh sách khách hàng với các thông tin address, city, state.
2. Xuất các khách hàng có city ở "Woburn" và state là "MA".
3. Xuất thông tin khách hàng bao gồm individual, first_name, last_name của officer.

4. Nhóm danh sách khách hàng theo city và cho biết số lượng khách hàng tương ứng với điều kiện là các khách hàng có city ở “Salem”.
5. Xuất danh sách 5 khách hàng đầu tiên được sắp xếp theo thứ tự tăng dần của postal_code và giảm dần của fed_id.
6. Lọc các khách hàng có state là “MA”. Sau đó, thực hiện nhóm các khách hàng vừa tìm được theo city và đếm số lượng khách hàng tương ứng với danh sách city vừa nhóm. Tiếp theo thực hiện xuất 03 dữ liệu cuối cùng được sắp xếp kết quả giảm dần theo postal_code. ***{city}
7. Xuất danh sách khách hàng có tồn tại field business.
8. Xuất danh sách khách hàng có tồn tại field business với các thông tin fed_id, cust_type_cd, state, khuyenMai.
   Trong đó field khuyenMai được tính như sau: nếu field state = “MA” thì sẽ được hưởng khuyến mãi (khuyenMai: true), ngược lại sẽ không được hưởng khuyến mãi (khuyenMai: false).
9. tiếp theo câu 9. Nếu field khuyenMai là true thì sẽ hiển thị là “Được giảm 10%”, ngược lại sẽ hiển thị “Không được hưởng khuyến mãi”.

Quay lại làm bài 2.2 từ câu 11 (tuần 1-2-3)

Bài 2: Tạo database ***dbtuan5*** import collection ***zips.json***, tìm hiểu ý nghĩa của document sau đó thực hiện yêu cầu:

---

zipcodes

```
{
_id: '01001',
city: 'AGAWAM',
loc: [ -72.622739, 42.070206 ],
pop: 15338,
state: 'MA'
}
```

1. Hiển thị n documents từ document thứ k. *(n, k tùy ý)*
2. Chèn thêm 1 document mới. (tùy ý)
3. Cập nhật thông tin của một document khi biết id bất kỳ.
4. Tìm dân số của thành phố (city) PALMER.
5. Tìm các document có dân số (pop) >100000
6. Tìm dân số của thành phố (city) FISHERS ISLAND
7. Tìm các thành phố có dân số từ 10 – 50
8. Tìm tất cả các thành phố của bang MA có dân số trên 500
9. Tìm tất cả các bang (không trùng)
10. Tìm tất cả các bang mà có chứa ít nhất 1 thành phố có dân số trên 100000
11. Tính tổng số dân (pop) theo từng bang (state).
12. Tìm tất cả các bang có tổng dân số trên 10.000.000
13. Tính dân số trung bình (các thành phố) theo từng bang (state).
14. Tìm những document của bang 'CT' và thành phố 'WATERBURY'
15. Bang WA có bao nhiêu city (nếu trùng chỉ tính 1 lần)**
16. Tính số city của mỗi bang (nếu trùng chỉ tính 1 lần), kết quả giảm dần theo số city
17. Tìm ra các thành phố có dân số (pop) lớn (nhỏ) nhất.

18. Tìm bang có dân số (pop) lớn (nhỏ) nhất.
19. Xuất những document có dân số dưới dân số trung bình của mỗi city

Bài 3: Collection **movies.** Đọc và tìm hiểu các field trong collection

Movies

{

_id: ObjectId("573a1390f29313caabcd418c"),

title: 'The House of the Devil',

year: 1896,

runtime: 3,

cast: ["Jeanne d'Alcy", 'Georges M�li�s'],

plot: 'A bat flies into an ancient castle and transforms itself into Mephistopheles himself. Producing a cauldron, Mephistopheles conjures up a young girl and various supernatural creatures, one ...',

fullplot: 'A bat flies into an ancient castle and transforms itself into Mephistopheles himself. Producing a cauldron, Mephistopheles conjures up a young girl and various supernatural creatures, one of which brandishes a crucifix in an effort to force the devil-vampire to vanish.',

lastupdated: '2015-08-26 00:06:16.697000000',

type: 'movie',

directors: ['Georges M�li�s'],

writers: ['Georges M�li�s'],

imdb: { rating: 6.8, votes: 1135, id: 91 },

countries: ['France'],

genres: ['Short', 'Horror'],

tomatoes: {

viewer: { rating: 5, numReviews: 23 },

lastUpdated: ISODate("2015-06-02T19:48:08.000Z")

},

num_mflix_comments: 1,

comments: [

{

name: 'Oscar Sanchez',

```
        email: 'oscar_sanchez@fakegmail.com',
        movie_id: ObjectId("573a1390f29313caabcd418c"),
        text: 'Non repellat atque in ipsa accusantium. Assumenda modi magni quis.\n' +
          'Recusandae recusandae dicta repellat ad reprehenderit mollitia quam. Itaque voluptate asperiores quia alias.',
        date: ISODate("2017-02-11T07:00:52.000Z")
      }
    ]
  }
```

Thực hiện các phép truy vấn aggregation trên collection movies:

1. Đếm tổng số các document movies
2. Xuất các document movies theo năm, tính tổng số film trong mỗi năm
3. Xuất các document movies theo năm, tính tổng số film trong mỗi năm, sau đó sắp xếp tăng.
4. Xuất các document movies theo năm, sau đó sắp xếp theo thứ tự giảm dần dựa trên số lượng.
5. Xuất các document movies theo số lượng film mỗi đạo diễn có được,
6. Xuất các document movies theo số lượng film từng year, type, title. Sau đó sắp xếp giảm dần theo số lượng đếm được.
7. Liệt kê danh sách các đạo diễn có tham gia từ 30 bộ phim trở lên. Thông tin bao gồm: Tên đạo diễn (director) và số bộ phim.

```
Hướng dẫn: Đạo diễn dựa vào thuộc tính directors; output dạng: [ { director: 'Takashi Miike','number of movies': 34 }, …]
```

8. Tìm tất cả các đạo diễn có tham gia đạo diễn nhiều bộ phim nhất

```
Hướng dẫn: Output là: [ { director: 'Woody Allen' } ]
```

9. Liệt kê tựa phim (title) theo từng đạo diễn. Thông tin bao gồm: tên đạo diễn (director) và danh sách tựa phim

```
Hướng dẫn: Dùng Pivot Data; output dạng : output:[ { movies: [ 'Lost in America', 'Defending Your Life', 'Mother' ], director: 'Albert Brooks'}, …]
```

10. Thống kê số bộ phim đã phát hành theo từng năm, sắp xếp giảm dần theo năm

```
Hướng dẫn: năm dựa vào field released; output dạng:[ { 'number of movies': 10, year: 2016 },]
```

11. Tìm năm phát hành nhiều bộ phim nhất.

```
Hướng dẫn: Năm dựa vào field released; output: [ { year: 2014 } ]
```

12. Liệt kê danh sách các tựa phim (title) theo từng quốc gia. Thông tin bao gồm: tên quốc gia và danh sách tựa phim

Hướng dẫn: Tên quốc gia dựa vào thuộc tính countries; Dùng Pivot Data; output dạng:[{ movies: ['Bitter Sugar', 'Red Passport', 'Sugar', 'Jean Gentil', 'Kidnapped for Christ', 'CèdigoPaz'], country: 'Dominican Republic' }, …]

13. Đếm số bộ phim theo từng quốc gia, sắp xếp giảm dần theo số bộ phim. Thông tin bao gồm:Tên quốc gia và số bộ phim

Hướng dẫn: Tên quốc gia dựa vào thuộc tính countries; output dạng:[{ country: 'USA', 'number of movies': 11855 }, ….]

14. Tìm những tựa phim (title) phát hành trong tháng 03 năm 2016

Hướng dẫn: Tháng và năm dựa vào thuộc tính released; output là: [{ title: 'Knight of Cups'}, { title: 'Sand Castles' }, { title: 'The Treasure' }]

15. Liệt kê những tựa phim (title) do diễn viên “Frank Powell” hoặc “Charles Wellesley” đóng

Hướng dẫn: Diễn viên dựa vào thuộc tính cast; output là : [{ title: 'A Corner in Wheat' }, {title: 'The Poor Little Rich Girl' }]

16. Tìm những quốc gia phát hành nhiều bộ phim nhất

Hướng dẫn: Tên quốc gia dựa vào thuộc tính countries; output là [{ country: 'USA' }]

Tuần 9: Dựng mô hình ReplicaSet

Chuẩn bị:

1. *Vào ổ T, tạo thư mục mới có tên "STT_HoTen";*
2. *Giải nén file "Tuan09.rar" vào đường dẫn "T:\STT_HoTen";*

Phần 1: trong file config của các node: node1, node2, node3, node4, arbiter. Thực hiện:

- Câu 1: cập nhật lại đường dẫn thư mục chứa file data và file log cùng cấp với thư mục chứa file config của các node.

- Câu 2: cập nhật lại đường dẫn chứa "keyFile" đúng với thực tế.

- Câu 3: cập nhật tên replicaset theo cú pháp "repMSSV".

Phần 2: dựng mô hình replica set.

- Câu 4: sử dụng lệnh mongod chạy các server: "node1", "node2", "node3".

- Câu 5: sử dụng mongosh kết nối vào server "node1". Chạy lệnh khởi tạo mô hình replica set. Sau đó tạo user admin trong database admin với role root. Thực hiện chứng thực user sau khi tạo.

- Câu 6: thực hiện add các server "node2" và "node3" vào replica set vừa khởi tạo.

- Câu 7: thực hiện add thêm server "node4" vào replica set.

- Câu 8: thực hiện add thêm server "arbiter" với vai trò là trọng tài vào replica set.

- Câu 9: thực hiện xóa server "node4" ra khỏi replica set/

Câu 10: sử dụng lệnh isMaster() để kiểm tra thông tin của replica set.

Tuần 10: Dựng mô hình Shard Cluster

Chuẩn bị:

1. *Vào ổ T, tạo thư mục mới có tên "STT_HoTen";*
2. *Giải nén file "Tuan10.rar" vào đường dẫn "T:\STT_HoTen";*

Phần 1: trong file config của các node: csrs1, csrs2, csrs3, mongos. Thực hiện:

- Câu 1: cập nhật lại đường dẫn thư mục chứa file data và file log cùng cấp với thư mục chứa file config của các node.

- Câu 2: cập nhật lại đường dẫn chứa "keyFile" đúng với thực tế.

- Câu 3: cập nhật tên replicaset theo cú pháp "repConfigServer". Riêng server "mongos" thì cập nhật lại đường dẫn của "configDB".

Phần 2: dựng mô hình shard cluster.

- Câu 4: sử dụng lệnh mongod chạy các server: "csrs1", "csrs2", "csrs3".

- Câu 5: sử dụng mongosh kết nối vào server "csrs1". Chạy lệnh khởi tạo mô hình replica set. Sau đó tạo user admin trong database admin với role root và chứng thực user sau khi tạo. Sau đó add các server "csrs2" và "csrs3" vào replica set vừa khởi tạo

- Câu 6: chạy server "mongos". Kết nối vào server "mongos" và chứng thực user.

- Câu 7: cập nhật file config của các "node1", "node2", "node3": đường dẫn chứa file dữ liệu, file log cùng cấp với thư mục chứa file config và cập nhật tên replica set là "repMSSV".

- Câu 8: Dựng mô hình replica set với 3 node vừa tạo.

- Câu 9: Thêm replica set "repMSSV" vào shard cluster.

- Câu 10: Chạy lệnh sh.status() để kiểm tra trạng thái.