Bài 1 - MongoDB Query Exercises: 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-folder/]/[file.bson]
```

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:

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

Ví dụ:

1. `mongobackup --db Test`
   → *Backup databaseTest ra thư mục hiện hành*
2. `mongobackup --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,
	{
	_id: ObjectId("620a8bff7c3b0f4af1d79d98"),
	Name: 'DHKTPM14',
	Subject: 'Programming Application',
	Hours: 5
	}

lienLac: { email: 'no@gmail.com', phone: '0999.987.222' } }	

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 dbTest có collection restaurants thực hiện một số câu truy vấn sau:

1. Hiển thị tất cả các documents có trong collection restaurants.

```
db.restaurants.find();
```

2. Hiển thị tất cả các documents có trong collection restaurants, tuy nhiên chỉ xuất các fields *restaurant_id, name, borough and cuisine*

```
db.restaurants.find({},{"restaurant_id" : 1,"name":1,"borough":1,"cuisine" :1});
```

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.*

```
db.restaurants.find({},{"restaurant_id" : 1,"name":1,"borough":1,"cuisine" :1,"_id":0});
```

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.*

```
db.restaurants.find({},{"restaurant_id" : 1,"name":1,"borough":1,"address.zipcode"
:1,"_id":0});
```

5. Hiển thị tất cả các documents có trong collection restaurants với field borough có giá trị là Bronx.

```
db.restaurants.find({"borough": "Bronx"});
```

6. Hiển thị 5 documents đầu tiên có trong collection restaurants với field borough có giá trị là Bronx.

```
db.restaurants.find({"borough": "Bronx"}).limit(5);
```

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.

```
db.restaurants.find({"borough": "Bronx"}).skip(5).limit(5);
```

8. Hiển thị tất cả các documents có trong collection restaurants với điều kiện score trong field grades lớn hơn 90.

```
db.restaurants.find({grades : { $elemMatch:{"score":{$gt : 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.

```
db.restaurants.find({grades : { $elemMatch:{"score":{$gt : 80 , $lt :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.

```
db.restaurants.find({"address.coord" : {$lt : -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. ******

```
db.restaurants.find({$and:[{"cuisine" : {$ne :"American "}},{"grades.score" : {$gt :
70}},{"address.coord" : {$lt : -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.

```
db.restaurants.find({"cuisine" : {$ne : "American "},"grades.score" :{$gt: 70},
"address.coord" : {$lt : -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.

```
db.restaurants.find( {"cuisine" : {$ne : "American "},
 "grades.grade" :"A","borough": {$ne : "Brooklyn"}}).sort({"cuisine":-1});
```

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'.

```
db.restaurants.find({name: /^Wil/},
                    {"restaurant_id" : 1,"name":1,"borough":1,"cuisine" :1});
```

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'.

```
db.restaurants.find({name: /ces$/},
                    {"restaurant_id" : 1,"name":1,"borough":1,"cuisine" :1});
```

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'.

```
db.restaurants.find({"name": /.*Reg.*/},

                    {"restaurant_id" : 1,"name":1,"borough":1,"cuisine" :1});
```

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.

```
db.restaurants.find({ "borough": "Bronx" ,

                      $or : [{ "cuisine" : "American " },{ "cuisine" : "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

```
db.restaurants.find(

{"borough" :{$in :["Staten Island","Queens","Bronx","Brooklyn"]}},

{"restaurant_id" : 1,"name":1,"borough":1,"cuisine" :1});
```

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.

```
db.restaurants.find(

{"borough" :{$nin :["Staten Island","Queens","Bronx","Brooklyn"]}},

{"restaurant_id" : 1,"name":1,"borough":1,"cuisine" :1});
```

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.

```
db.restaurants.find(

{"grades.score" : { $not: {$gt : 10}}},

{"restaurant_id" : 1,"name":1,"borough":1,"cuisine" :1});
```

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 dsXe 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.

Tập mẫu Xe.json

```
{ _id: ObjectId("6211e7a0a0f09845396dca07"),
  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

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: *(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.

Sinh viên

```
{
  _id: ObjectId("6131a2a278b3264add832589"),
  diemTB: 8.5,
  dsDienthoai: [ '0338618310' ],
  email: 'henry412@gmail.com',
  gioitinh: 'Nam',
  ho: 'Nguyễn Huỳnh Chí',
  malop: 'DHKTPM15B',
  mssv: '19405011',
  ngaysinh: ISODate("2001-09-19T00:00:00.000Z"),
  ten: 'Bảo'
}
```

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

Lớp học

```
{
  _id: ObjectId("6131e5e97cf137f22a235ce3"),
  macn: 'KTPM',
  mslop: 'DHKTPM12BCLC',
  sisoDukien: 45,
  tenlop: 'Đại học Kỹ thuật Phần mềm 12B Chất lượng cao'
}
```

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. Thông tin bao gồm mã lớp và tổng số sinh viên.
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.
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.