[1] MongoDB 기본 및 Spring Boot 연동

- MongoDB CRUD 실습 (insert, find, update, delete)

- 필터링, 정렬, distinct, aggregate 를 활용한 다양한 쿼리

- 조건문, 루프문, 정규표현식을 이용한 JavaScript 문법 MongoDB 쉘 활용

- Spring Boot + MongoDB 연동 (MongoRepository, @Document 등)

- Postman 을 활용한 API 테스트 및 실습

[2] 날짜 처리 및 고급 Aggregation

- 날짜 데이터 ISODate 및 new Date() 활용법

- 날짜 범위 필터링 ($gte, $lte), 정렬, 업데이트

- 날짜 기반 Aggregation ($year, $month, $dayOfMonth)

- $dateAdd, $dateDiff, $dateFromParts, $dateToParts 등 날짜 연산자 실습

- React 와 연동될 JSON 기반 API 설계 및 MongoDB 날짜 관련 필드 처리 실습

[3] 배열 연산자 및 조인 연산자 실습

- 배열 연산자: $push, $addToSet, $map, $filter, $slice, $unwind 등

- 배열 필드의 다양한 조작과 Aggregation 연습

- $lookup, $merge, $unionWith 을 활용한 조인과 셀프 조인

- 조건부 조인, Cross Join, Inner Join, Outer Join, Equi Join 실습

- React 에서 배열 데이터를 처리하는 데 필요한 쿼리 포맷 정리

[4] GridFS 를 활용한 파일 업로드 및 다운로드

- mongofiles CLI 사용법 실습: put, get, delete, search, list

- GridFS 내부 구조 이해 (fs.files, fs.chunks)

- 파일 ID 기반 업로드 및 다운로드, 쉘 스크립트 활용 자동화

- mongoimport, mongoexport 를 활용한 JSON/CSV 백업 및 복원 실습

[5] 인덱스, 성능 최적화, explain(), 샤딩 개념

- 단일 및 복합 인덱스 생성/삭제 및 성능 확인

- db.collection.aggregate([{ $indexStats: {} }]) 활용

- explain("executionStats")로 쿼리 성능 분석

- 샤딩 및 복제 개념 소개 및 구성 원리 이해

MongoDB 도구

No.	도구 이름	2024 년 기준 버전	2025 년 6 월 최신 버전	비고
1	MongoDB Community Server	8.0	8.0.10	최신 패치 업데이트 포함
2	mongosh (Shell)	1.11.0	2.5.2	대규모 기능 개선
3	MongoDB Compass (GUI)	1.40.3	1.46.3	UI 및 UX 개선
4	MongoDB Atlas	–	상시 최신	클라우드 서비스
5	MongoDB Ops Manager	6.0	확인 필요	Enterprise 전용
6	MongoDB Charts	2.6.1	확인 필요	Atlas 에서 관리
7	MongoDB BI Connector	2.14.0	확인 필요	BI 툴 연동 도구

설치 및 실행 참고

- 실행 파일 위치

  C:₩Program Files₩MongoDB₩Server₩8.0₩bin

- 서비스 실행 명령:

  "C:₩Program Files₩MongoDB₩Server₩8.0₩bin₩mongod.exe" --config
  "C:₩Program Files₩MongoDB₩Server₩8.0₩bin₩mongod.cfg" --service

- 다운로드 링크:
  - MongoDB Community Server: https://www.mongodb.com/try/download/community
- mongosh Shell: https://www.mongodb.com/try/download/shell

MongoDB 란?

MongoDB 는 대표적인 NoSQL(Non-Relational Database) 데이터베이스 시스템으로, **문서(document) 지향(Document-Oriented)** 저장 방식을 사용한다.

◇ 주요 특징

- **관계형 DB 와 달리 테이블, 행, 열이 없음**
- BSON (Binary JSON) 형식으로 데이터를 저장
- **스키마가 유연**하여, 각 문서마다 다른 구조의 데이터를 가질 수 있음
- 수평 확장(Sharding) 용이 → 대용량 데이터 처리에 적합

정형 데이터 (Structured Data)

항목	설명
정의	미리 정의된 스키마(열 구조)를 따르는 데이터
예시	SQL, 엑셀 표, 고객 정보(이름, 전화번호, 주소 등)
저장 방식	관계형 데이터베이스(RDBMS), CSV, Excel 등
특징	검색·정렬이 빠르고 체계적이지만 유연성이 낮음

MySQL, Oracle, PostgreSQL 등은 정형 데이터 저장에 적합

비정형 데이터 (Unstructured Data)

항목	설명
정의	사전에 정의된 구조가 없는 자유로운 형태의 데이터
예시	이미지, 동영상, 로그파일, 센서 데이터, JSON
저장 방식	파일 시스템, NoSQL DB(MongoDB 등), Object Storage
특징	유연성과 확장성 뛰어남. 구조 없이도 검색 가능하도록 설계 필요

MongoDB 와 정형/비정형 데이터 비교

기준	관계형 DB (MySQL 등)	MongoDB
스키마 구조	고정 (정형)	유연함 (비정형/반정형)
데이터 표현 방식	행(Row) / 열(Column)	문서(Document) / JSON
사용 목적	일관된 형식의 데이터 저장	유연한 구조, 대규모 데이터, 빠른 개발
예시	은행거래, 회원 DB 등	IoT 데이터, 로그, 채팅, CMS, 분석용 데이터

[실습 01] - MongoDB CRUD 실습 (insert, find, update, delete)

```
db.Score.insert({name:"aaa",kor:90,eng:80,mat:98,test:"midterm"})
db.Score.insert({name:"bbb",kor:100,eng:100,mat:76,test:"final"})
db.Score.insert({name:"ccc",kor:90,eng:55,mat:67,test:"midterm"})
db.Score.insert({name:"ddd",kor:70,eng:69,mat:89,test:"midterm"})
db.Score.insert({name:"eee",kor:60,eng:80,mat:78,test:"final"})
db.Score.insert({name:"fff",kor:90,eng:65,mat:98,test:"midterm"})
db.Score.insert({name:"ggg",kor:75,eng:100,mat:98,test:"final"})
```

분류	메소드 / 연산자	설명
삽입	insert, insertOne	새 문서를 컬렉션에 추가
조회	find, findOne	문서 조회
반복	forEach, while, hasNext, next	자바스크립트 기반 루프 처리
집계	aggregate	문서들을 그룹화/변환/계산
그룹화	$group	특정 필드를 기준으로 그룹 생성
계산	$sum, $avg, $max, $min, $size	합계, 평균, 최대, 최소, 배열 크기 계산
조건	$match	조건에 맞는 문서 필터링 (aggregate 내부)
필터링	$gte, $lte, $or, $regex, $type	비교, 정규식, 논리 조건, 타입 필터링
정렬	sort()	문서 정렬 (1: 오름차순, -1: 내림차순)
제한	limit(), skip()	결과 문서 개수 제한 및 건너뛰기
중복 제거	distinct()	중복 없이 특정 필드 값만 추출
출력	print()	쉘에서 결과값 출력
조건문	if, $cond, $isArray	조건 판단 처리
필드선택	$project	필드 가공 및 결과 필드 선택
정규표현식	/^a/, /^e/	이름 패턴으로 시작하는 값 필터링

 # Q1) Score 의 전체 출력 해보자.

```
db.Score.find().count();
```

 # Q2) Score 의 이름과 수학 점수만 출력 해보자.

```
db.Score.find({}, {name:1,mat:1, _id:0})
```

 # Q3) Score 의 수학 점수 중 70 점 이상만 출력 해보자.

```
db.Score.find({mat:{$gte : 70}}).count()
```

 # Q4) Score 의 이름과 국어점수를 출력하되 국어점수가 80 점 이상만 출력하자.

```
var sr = db.Score.find({kor: {$gte:80}},{name:1,kor:1,_id:0})

var tot = 0

while(sr.hasNext()){

	res = sr.next()

	print(res.name + ":" + res.kor);

	tot += res.kor;

	print("tot =" + tot);

}

print("tot =" + tot);
```

 # Q5) Score 의 전체 출력하자 .이름과 test 를 출력하자.

```
var sm = db.Score.find();

sm.forEach(function(x) {

    if(x.test == "final")

            print(x.name +":"+ x.test)

})
```

 # Q6) Score 의 전체 출력하자

```
# 조건 1 : test 가  midterm 만  출력하자.

# 조건 2 : midterm 의  수학만 합을 구하자.

var sm = db.Score.find();

var tot = 0;

sm.forEach(function(x) {

    if(x.test == "midterm")

        tot += x.mat;

})

print("mat_tot =" + tot);
```

 # Q7) Score 에서 이름이 a 로 시작되는 데이터를 출력하자

db.Score.find({name : /^a/})

 # Q8) Score 에서 이름이 a 로 시작하거나 e 로 시작하는 애들을 출력

db.Score.find({"$or": [{"name" : /^a/} , {"name" :/^e/}] });

 # Q9) Score 에서 test 가 m 으로 시작하는 사람들의 개수를 구해보자.

db.Score.find({test : /^m/}).count()

 # Q10) 중복데이터 제거

 # 이름을 출력하되 중복 되지 않도록 출력하자

db.Score.distinct("name");

 # Q11) distinct 를 이용해서 영어점수가 80 점 이상인 학생의 이름을 출력 해보자.

db.Score.distinct("name" , { eng : {$gte :80}});

 # Q12) 이름과 test 를 출력하되 이름을 내림차순으로 정렬해보자.

db.Score.find({} ,{name:1, test:1 }).sort({name: -1})

 # Q13) 영어의 점수가 가장 높은 레코드를 출력해라.

db.Score.find({}).sort({eng:-1}).limit(1);

 # Q14) 국어 점수가 가장 낮은 레코드를 출력해라

```
db.Score.find({}).sort({kor:1}).limit(1);
```

 # Q15) 2 개를 건너 뛰고 3 줄을 출력하라

```
db.Score.find({}).skip(2).limit(3)
```

 # Q16) 2 개를 건너 뛰고 3 줄을 출력하되 수학점수를 내림차순으로 출력하라

```
db.Score.find({}).skip(2).limit(3).sort({mat:-1})
```

 # Q17) test 를 그룹핑 해보자.

```
db.Score.aggregate([
   { $group: { _id: "$test", count: { $sum: 1 } } }
])
```

 # Q18) test 를 그룹핑 하되, 국어 점수가 90 점 이상인 학생만 출력하자.

```
db.Score.aggregate([
   { $match: { kor: { $gte: 90 } } },
   { $group: { _id: "$test", count: { $sum: 1 } } }
])
```

 # Q19) test 로 그룹핑하되, 수학 70 점 이상 합계 출력

```
db.Score.aggregate([
   { $match: { mat: { $gte: 70 } } },
   { $group: { _id: "$test", totalMath: { $sum: "$mat" } } }
])
```

 # Q20) test 로 그룹화 하되, kor, eng, mat 의 총점을 구하자.

```
db.Score.aggregate([
   { $group: {
       _id: "$test",
       sumkor: { $sum: "$kor" },
       sumeng: { $sum: "$eng" },
       summat: { $sum: "$mat" }
   }}
])
```

 # Q21) test 로 그룹화 하되 국어는 최고 점수, 영어는 최저 점수를 출력하자.

```
db.Score.aggregate([
   { $group: {
        _id: "$test",
        maxKor: { $max: "$kor" },
        minEng: { $min: "$eng" }
   }}
])
```

 # Q22) test 로 그룹화 하되 국어의 개수를 구해보자.

```
db.Score.insert({name:"hhh",kor:[100,20,40,100],eng:100,mat:98,test:"final"})
db.Score.aggregate([
   { $group: {
        _id: "$test",
        countkor: { $sum: { $cond: { if: { $isArray: "$kor" }, then: { $size: "$kor" }, else:
1 } } } }
   }}
])
```

 # Q23) Score 에서 국어 평균을 구하자.

```
db.Score.aggregate([
   { $group: {
        _id: null,
        sumkor: { $sum: "$kor" },
        countkor: { $sum: 1 }
   }},
   { $project: {
        _id: 0,
        averagekor: { $divide: ["$sumkor", "$countkor"] }
   }}
])
```

 # Q24) Score 에서 국어점수를 출력하되 배열의 값을 가진 값만 출력하자.

```
db.Score.find( {kor :{$type:"array"}}, {kor :1, _id :0})
```

 # Q25) Score 에서 국어점수를 출력하되 배열의 값을 가진 값만 합을 구하자.

```
var tot = 0;
var r = db.Score.find( {kor :{$type:"array"}}, {kor :1, _id :0}).toArray();
r[0]["kor"].forEach( function( val) { tot += val } )
tot
```