몽고디비 랩

몽고DB

MongoDB 서버에 연결

랩

Docker 빌드 수정

- 일부 Javascript 파일을 사용하여 MongoDB 컬렉션 및 문서를 부트스트랩하고 생성합니다.
- 이러한 파일은 MongoDB 서버 도커 컨테이너 내에서 액세스할 수 있어야 합니다.
- 가장 쉬운 방법은 모든 Javascript 파일을 mongo 디렉토리에 넣고 mongo 디렉토리의 모든 파일이 컨테이너 환경에 복사되도록 빌드를 수정하는 것입니다.

- 도전:
  - 어떤 변화가 필요합니까?

랩

Dockerfile 수정

- 1 단계
  - 강사가 제공한 모든 Javascript 파일을 ./mongo 디렉토리에 복사합니다.
- 2 단계
  - ./mongo에서 Dockerfile을 수정합니다.
  - 이 파일은 docker-composite에 mongo 컨테이너를 빌드하는 방법을 지시합니다.
  - 다음을 추가합니다.
- 3단계
  - docker-compose down 실행합니다.
  - docker-compose up --build 실행합니다

```
FROM mongo

RUN apt-get update \
    && apt-get install -y mongodb-org-shell \
    && rm -rf /var/lib/apt/lists/*


# Copy local code to the container image.
COPY . .
```

랩

터미널에서 MongoDB 컨테이너에 연결

- `docker ps` 사용 컨테이너가 작동하는지 확인하고 컨테이너 ID를 얻으려면

```
hwpark@wiken2 myapp % docker ps
CONTAINER ID   IMAGE         COMMAND                  CREATED         STATUS         PORTS
NAMES
029ca2f407eb   myapp-app     "docker-entrypoint.s…"   9 seconds ago   Up 8 seconds   0.0.0.0:3000->3000/tcp
myapp-app-1
b8c638e25355   myapp-mongo   "docker-entrypoint.s…"   9 seconds ago   Up 8 seconds   0.0.0.0:27017->27017/tcp
myapp-mongo-1
7a4a8ecd1cdf   redis         "docker-entrypoint.s…"   9 seconds ago   Up 8 seconds   0.0.0.0:6379->6379/tcp
myapp-redis-1
```

- 아래 명령과 함께 컨테이너 ID를 사용하여 bash 셸을 엽니다.

```
docker exec -it <container_id> bash
```

```
hwpark@wiken2 myapp % docker exec -it b8c638e25355 bash
root@b8c638e25355:/#
```

- 힌트: 종종 컨테이너 이름을 사용하는 것이 더 쉽습니다. 우리의 경우: `docker exec -it myapp-mongo-1 bash`

MongoDB 서버에 로그인

- mongosh를 사용하여 MongoDB 서버에 로그인
  - 자격 증명을 제공해야 합니다.
    - 사용자 = admin
    - 비밀번호 = password
  - 호스트 및 포트 이름
    - 호스트 = localhost
    - 포트 = 27017
  - 데이터베이스 인증
    - 인증에 사용된 DB 이름 = admin

```
mongosh --host localhost --port 27017 --authenticationDatabase
admin -u admin -p password
```

- 서버 컨테이너에 있으므로 기본값 사용

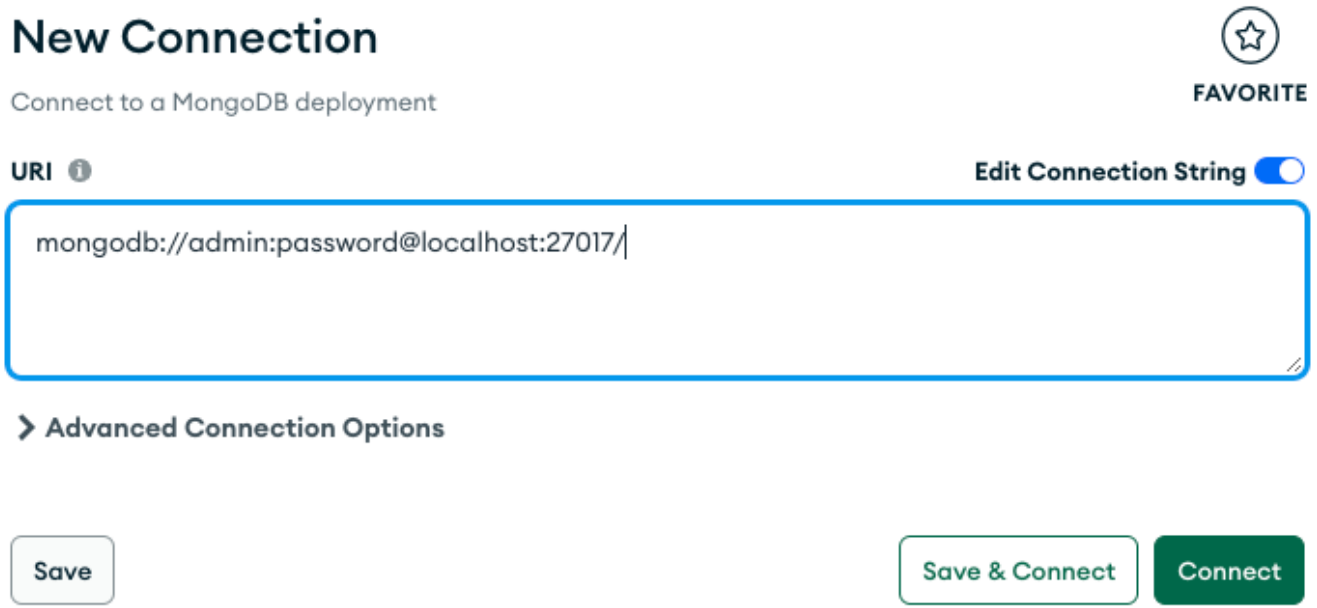
```
mongosh -u admin -p password
```

MongoDB 서버에 대한 연결 확인

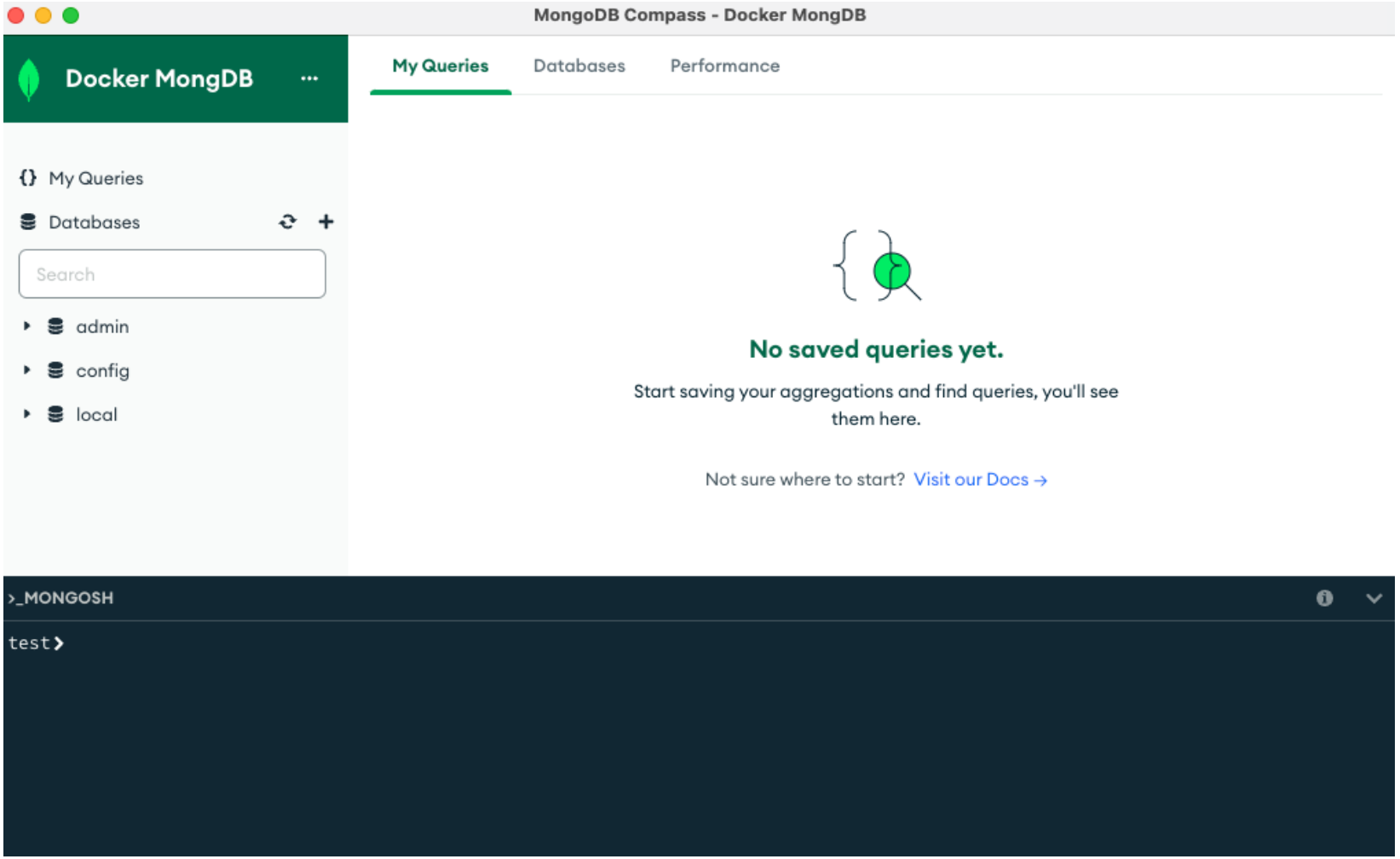
```
root@b8c638e25355:/# mongosh -u admin -p password
Current Mongosh Log ID: 648dd4831207b2f168fa906a
Connecting to:          mongodb://<credentials>@127.0.0.1:27017/?directConnection=tru
e&serverSelectionTimeoutMS=2000&appName=mongosh+1.9.1
Using MongoDB:          6.0.6
Using Mongosh:          1.9.1

For mongosh info see: https://docs.mongodb.com/mongodb-shell/

------
   The server generated these startup warnings when booting
   2023-06-17T10:53:42.604+00:00: Using the XFS filesystem is strongly recommended wi
th the WiredTiger storage engine. See http://dochub.mongodb.org/core/prodnotes-filesy
stem
   2023-06-17T10:53:43.504+00:00: vm.max_map_count is too low
------

test>
```

Compass에서 MongoDB 컨테이너에 연결

- MongoDB Compass 시작 및 새 연결 추가
- 다음 연결 문자열 사용
  - mongdb://admin:password@localhost:27017/
- 저장 및 연결



MongoDb에 대한 연결 확인



몽고DB

CRUD 소개
(만들기, 읽기, 업데이트, 삭제)

몽고DB

만들다

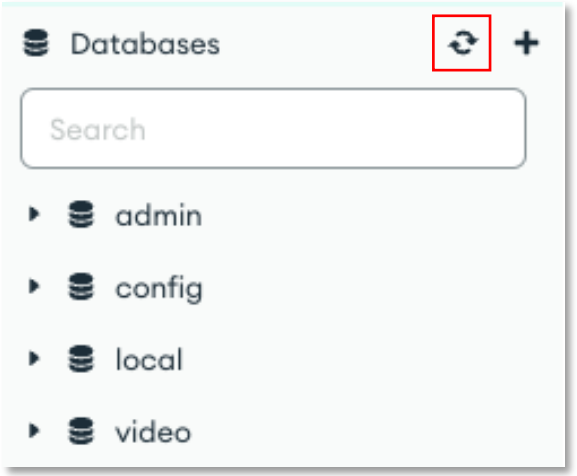
loadMovieDetail 데이터세트 생성

- loadMovieDetailsDataset.js
  - db.getSiblingDB (<데이터베이스>)
    - 서버에 대한 새로운 연결을 설정하지 않고 다른 데이터베이스로 변경할 수 있습니다.
  - db.<컬렉션>.drop
    - 컬렉션 삭제
  - db.<컬렉션>. insertMany
    - 컬렉션에 여러 문서 삽입
    - 문서는 목록에서 { key:value } 개체로 표시됩니다.
- load(" myscript.js ")를 사용하여 실행
- 삽입 방법 검토
  - https://www.mongodb.com/docs/manual/reference/insert-methods/

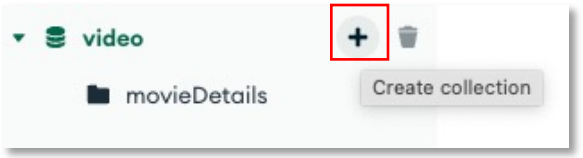
```
test> show dbs
admin   100.00 KiB
config  108.00 KiB
local    80.00 KiB
test> load("loadMovieDetailsDataset.js")
true
video> show dbs
admin   100.00 KiB
config  108.00 KiB
local    80.00 KiB
video     8.00 KiB
video> use video
already on db video
video> show collections
movieDetails
video>
```

나침반에서 컬렉션 만들기

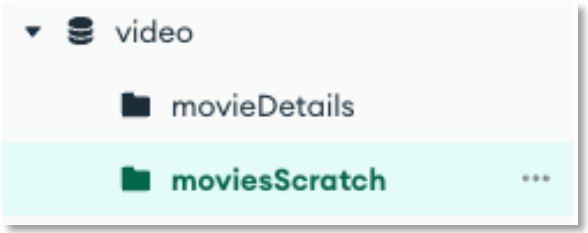
- 새로 생성된 비디오 DB가 보이지 않으면 데이터베이스 새로 고침



- 비디오 데이터베이스 선택

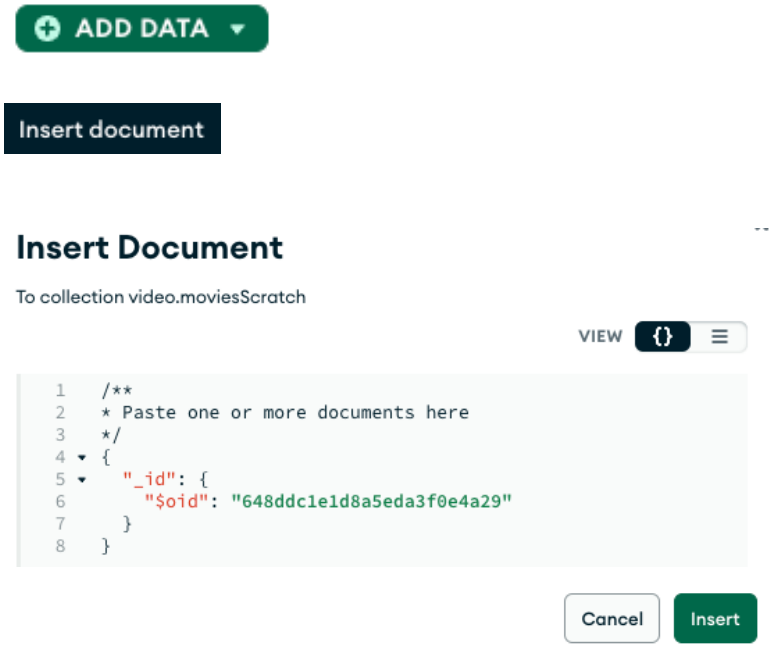


- 새 컬렉션 만들기
  - 무비스크래치
  - 시계열을 선택하지 마세요.
  - 모든 기본값 유지



Compass에서 문서 추가

- 선택하여 문서 삽입
- 문서 삽입을 선택합니다.
- 새 문서 삽입
  - 문서는 JSON 형식입니다.



- 다음 문서 추가
  - {"title":"Rocky", "year":1976, "imdb":"tt0075148"}
  - {"title" :"Creed", "year" : 2015, "imdb":"tt3076658"}

셸에서 문서 추가

- moviesScratch 컬렉션에 새 문서 추가
  - insertOne() API 사용

```
video> show collections
movieDetails
moviesScratch
video> db.moviesScratch.insertOne({"title":"Star Trek: The Wrath of Khan", "year":198
2, "imdb":"tt0084726"})
{
  acknowledged: true,
  insertedId: ObjectId("648de0471207b2f168fa9962")
}
```

  - Compass에서는 키를 따옴표로 묶어야 하지만 Shell에서는 그럴 필요가 없습니다.

insertMany – 여러 문서 삽입

- insertMany () 작업에는 두 가지 유형이 있습니다.
  - 주문됨 - 오류가 발생하면 중지
  - 순서가 지정된 삽입이 기본값입니다.

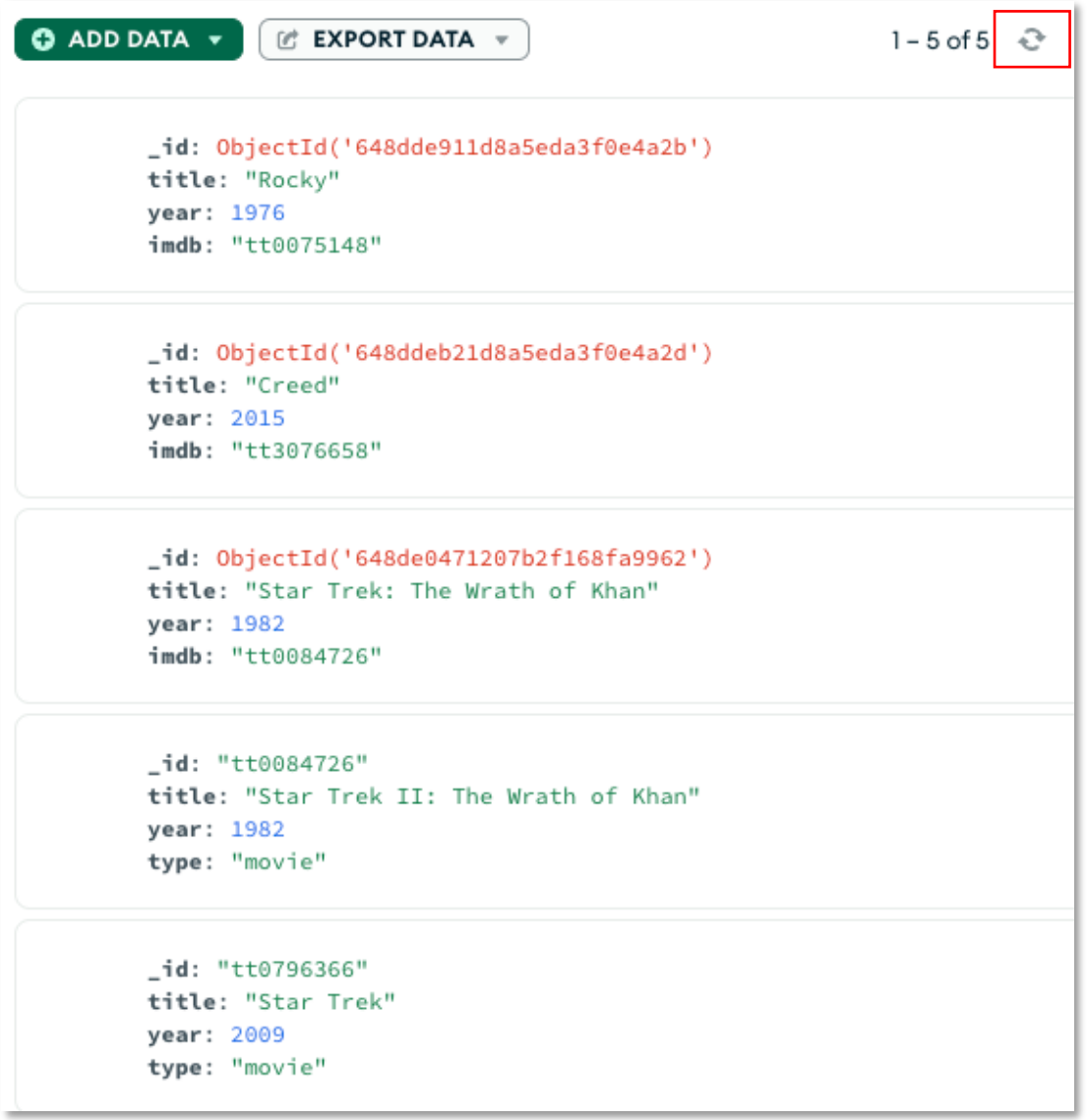
```
db.moviesScratch.insertMany (
[
{
"_id": "tt0084726",
"제목": "스타트렉 II: 칸의 분노",
"연도": 1982,
"유형": "영화"
},
{
"_id": "tt0796366",
"제목": "스타트렉",
"연도": 2009,
"유형": "영화"
},
{
"_id": "tt0084726",
"제목": "스타트렉 II: 칸의 분노",
"연도": 1982,
"유형": "영화"
},
{
"_id": "tt1408101",
"제목": "스타트렉 인투 다크니스",
"연도": 2013,
"유형": "영화"
},
{
"_id": "tt0117731",
"제목": "스타트렉: 퍼스트 컨택트",
"연도": 1996,
"유형": "영화"
}
]
);
```

```
Uncaught:
MongoBulkWriteError: E11000 duplicate key er
ror collection: video.moviesScratch index: _
id_ dup key: { _id: "tt0084726" }
Result: BulkWriteResult {
  insertedCount: 2,
  matchedCount: 0,
  modifiedCount: 0,
  deletedCount: 0,
  upsertedCount: 0,
  upsertedIds: {},
  insertedIds: {
    '0': 'tt0084726',
    '1': 'tt0796366',
    '2': 'tt0084726',
    '3': 'tt1408101',
    '4': 'tt0117731'
  }
}
Write Errors: [
  WriteError {
    err: {
      index: 2,
      code: 11000,
      errmsg: 'E11000 duplicate key error co
llection: video.moviesScratch index: _id_ du
p key: { _id: "tt0084726" }',
      errInfo: undefined,
      op: {
        _id: 'tt0084726',
        title: 'Star Trek II: The Wrath of K
han',
        year: 1982,
        type: 'movie'
      }
```

추가된 문서 검사

- 동영상 새로 고침스크래치 컬렉션

- 문서가 2개만 추가되었습니다.
  - _id를 지정하지 않은 나침반에서 추가한 문서에는 ObjectId가 할당되었습니다.



insertMany – 여러 문서 삽입

- insertMany () 작업에는 두 가지 유형이 있습니다.
  - Unordered – 오류가 발생한 후에도 나머지와 함께 계속 진행
  - 추가 {"ordered": false }

```
db.moviesScratch.insertMany (
[
{
"_id": "tt0084726",
"제목": "스타트렉 II: 칸의 분노",
"연도": 1982,
"유형": "영화"
},
{
"_id": "tt0796366",
"제목": "스타트렉",
"연도": 2009,
"유형": "영화"
},
{
"_id": "tt0084726",
"제목": "스타트렉 II: 칸의 분노",
"연도": 1982,
"유형": "영화"
},
{
"_id": "tt1408101",
"제목": "스타트렉 인투 다크니스",
"연도": 2013,
"유형": "영화"
},
{
"_id": "tt0117731",
"제목": "스타트렉: 퍼스트 컨택트",
"연도": 1996,
"유형": "영화"
}
],
{
"주문": 거짓
}
);
```

```
Uncaught:
MongoBulkWriteError: E11000 duplicate key error coll
ection: video.moviesScratch index: _id_ dup key: { _
id: "tt0084726" }
Result: BulkWriteResult {
  insertedCount: 2,
  matchedCount: 0,
  modifiedCount: 0,
  deletedCount: 0,
  upsertedCount: 0,
  upsertedIds: {},
  insertedIds: {
    '0': 'tt0084726',
    '1': 'tt0796366',
    '2': 'tt0084726',
    '3': 'tt1408101',
    '4': 'tt0117731'
  }
}
Write Errors: [
  WriteError {
    err: {
      index: 0,
      code: 11000,
      errmsg: 'E11000 duplicate key error collection
: video.moviesScratch index: _id_ dup key: { _id: "t
t0084726" }',
      errInfo: undefined,
      op: {
        _id: 'tt0084726',
        title: 'Star Trek II: The Wrath of Khan',
        year: 1982,
        type: 'movie'
      }
    }
  },
  WriteError {
    err: {
      index: 1,
      code: 11000,
```

추가된 문서 검사

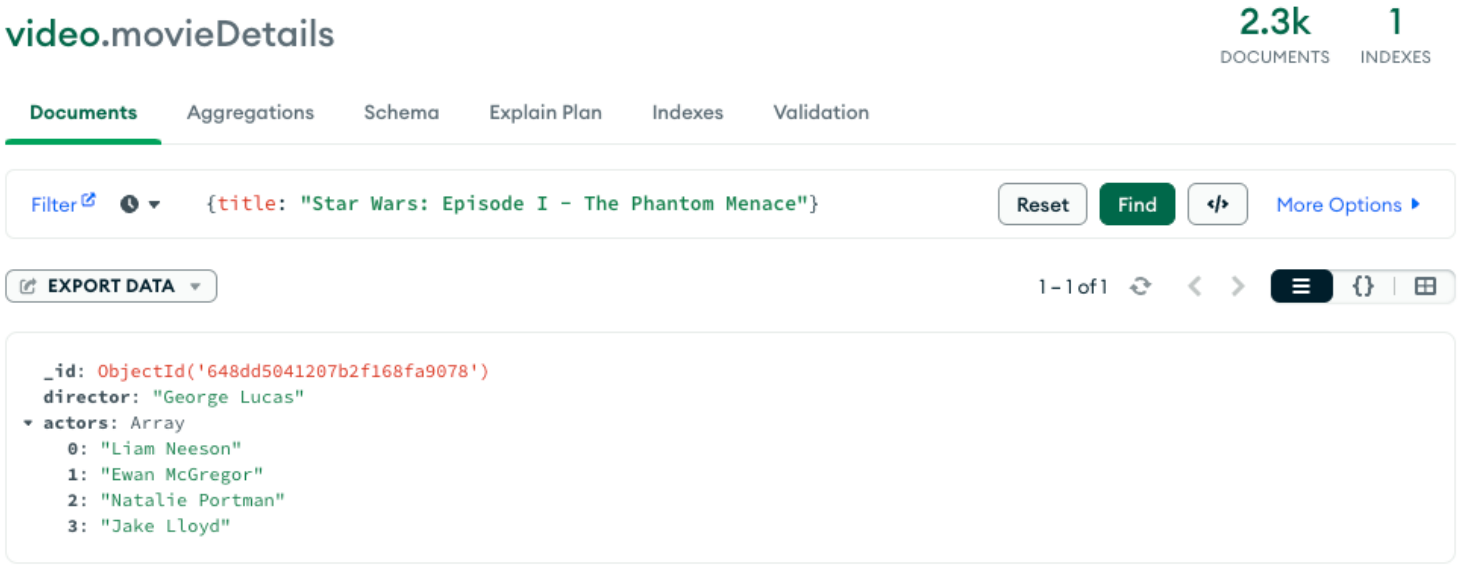
- 오류가 발생한 후에도 insertMany()는 계속해서 새 문서를 추가했습니다.
- 초기 오류 이후 두 개의 추가 문서가 추가되었습니다.
  - 스타 트렉 인투 다크니스
  - 스타 트렉: 퍼스트 컨택트

몽고DB

읽다

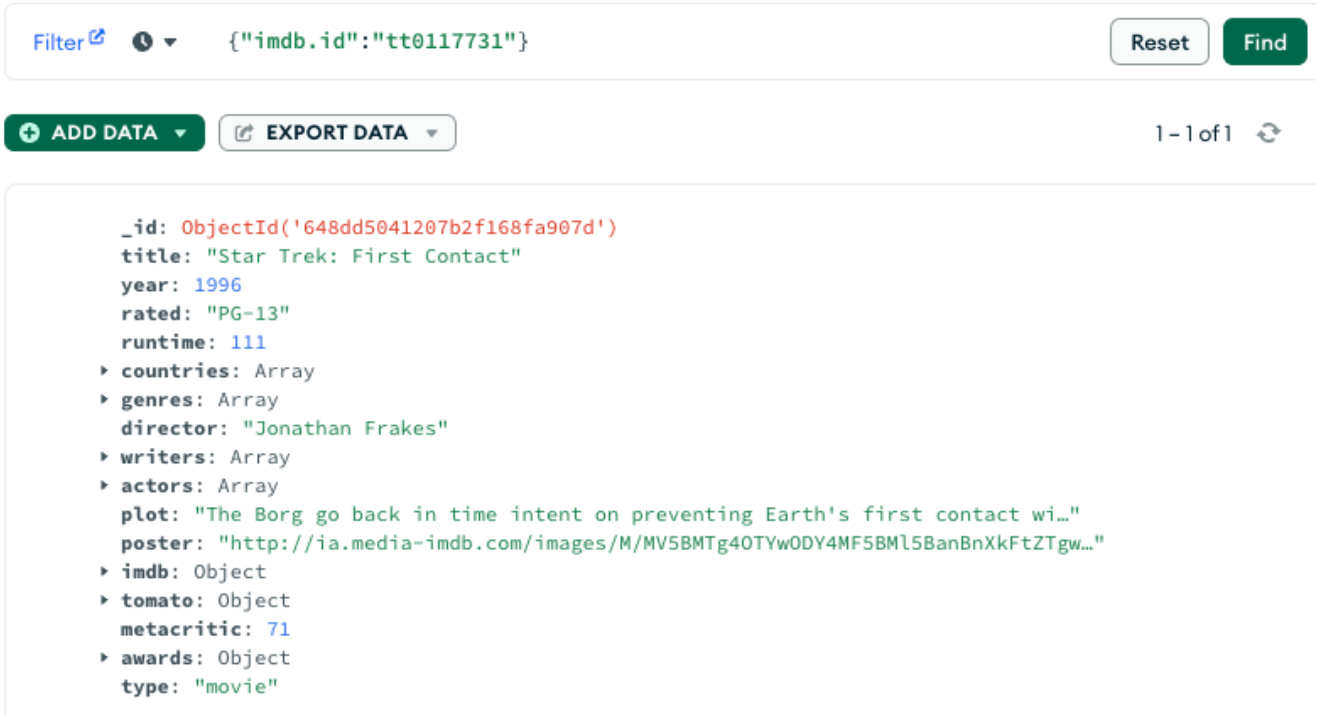
find()를 사용한 기본 쿼리

- 쿼리 방법 검토
  - https://www.mongodb.com/docs/manual/tutorial/query-documents/
- 가장 간단한 쿼리의 경우 검색할 키:값을 제공합니다.
  - 제목으로 영화 찾기
    - "스타워즈: 에피소드 I – 보이지 않는 위협"
- Compass에서 필터 조건을 추가하고 [찾기]를 클릭합니다.



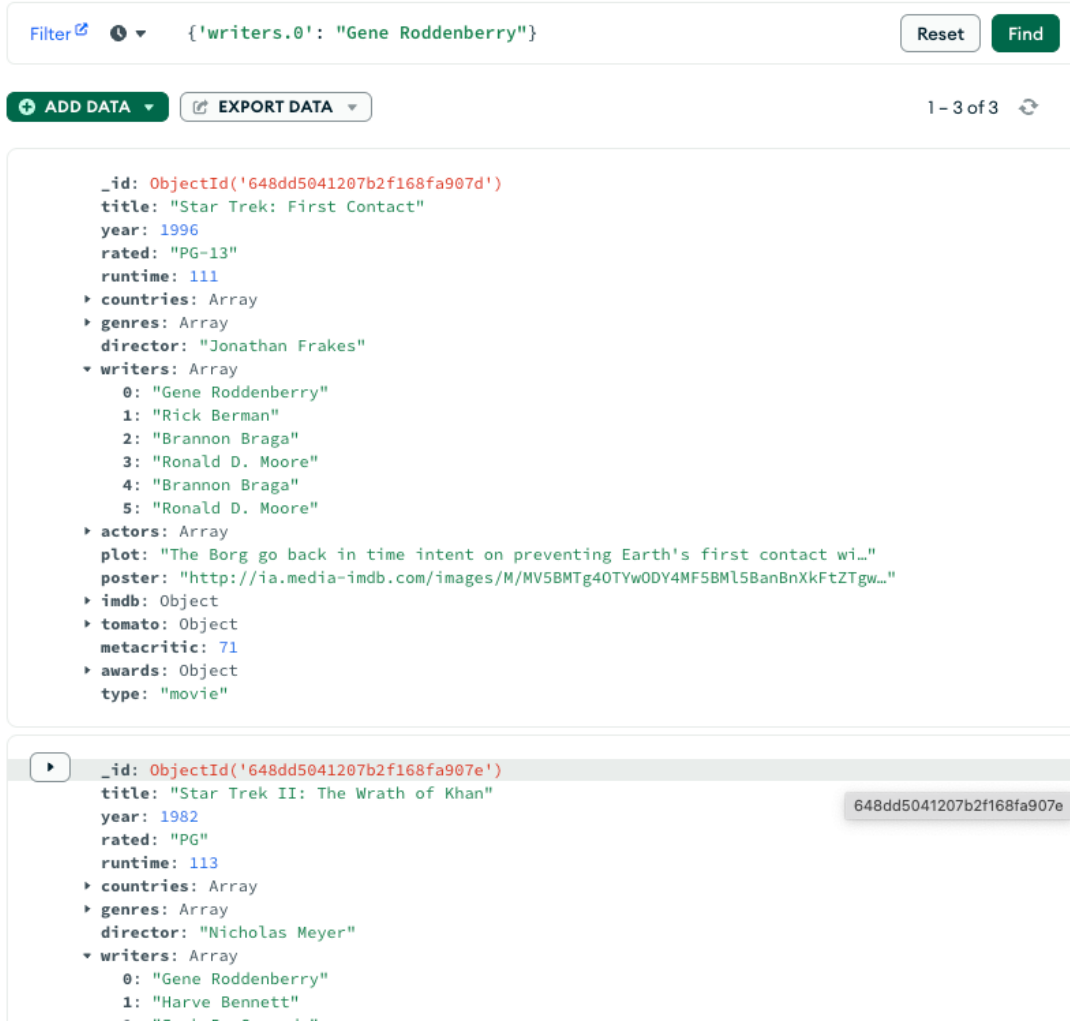
중첩 필드 쿼리

- 문서는 중첩된 JSON 필드를 포함할 수 있습니다.
  - 중첩 필드에 액세스하려면 DOT 표기법을 사용하십시오.
- IMDB – 영화 정보 사이트는 imdb.id를 사용하여 영화를 추적합니다.
  - imdb.i



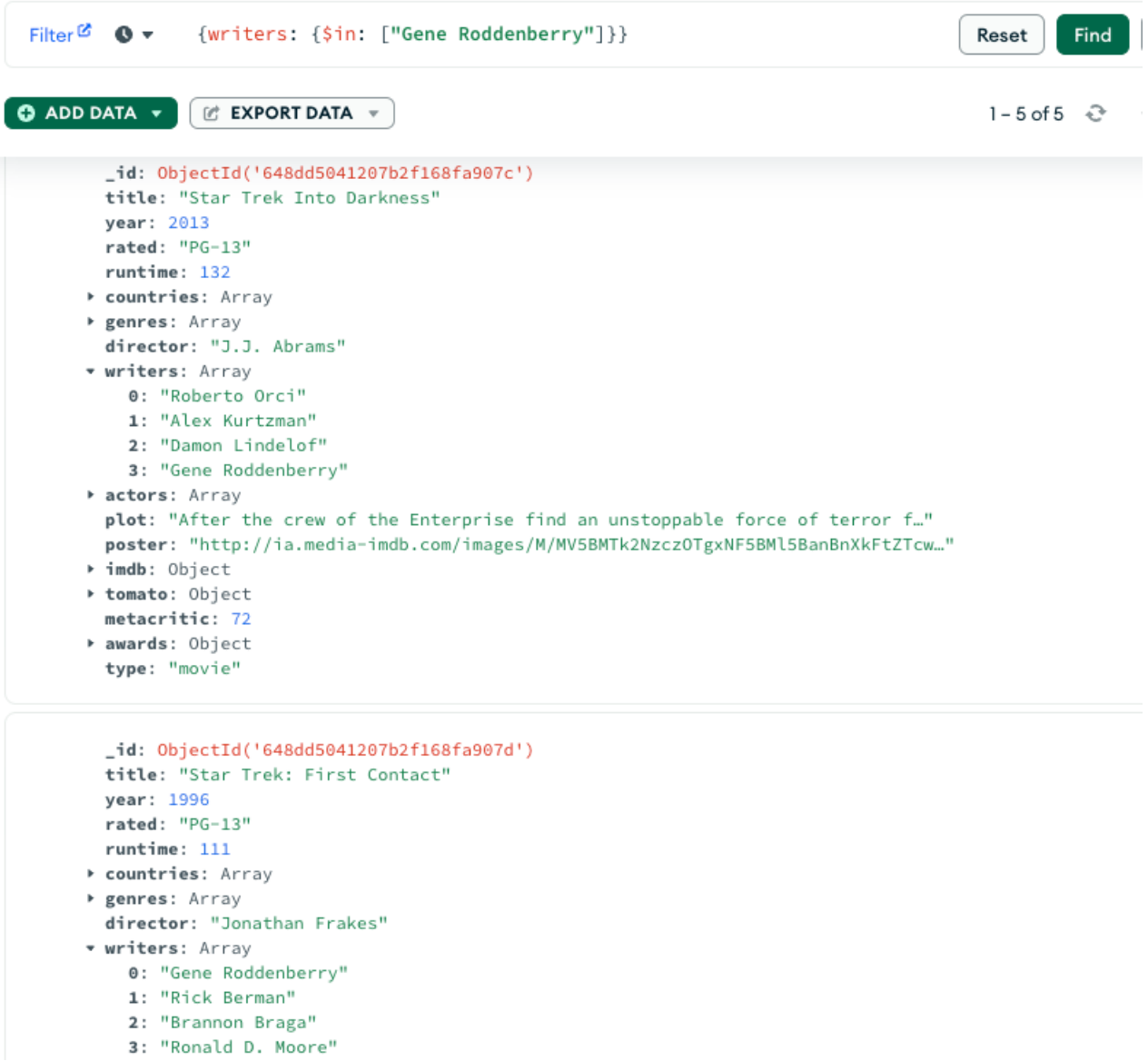
쿼리 배열 요소 필드

- 문서는 배열 필드를 포함할 수 있습니다.
  - 요소에 액세스하려면 field[index] 표기법을 사용하십시오.
  - 인덱스는 0에서 시작합니다.
  - 작가 분야에서는 첫 번째 작가가 메인 작가입니다.
  - "Gene Roddenberry"가 메인 작가였던 영화 검색



배열 필드의 쿼리 구성원

- 몇몇 영화에서 진 로든베리는 메인 작가가 아니었지만, 그는 작사 스태프의 일원이었다.
- $in 연산자를 사용하여 이것을 찾으십시오.



CRUD에 MongoDB Shell 사용

- Compass를 사용하는 것은 MongoDB의 쿼리 구문에 익숙해지는 좋은 연습 방법입니다.
- 쿼리에 대한 필터를 Mongo Shell로 쉽게 전송할 수 있습니다.
  - mongoose 또는 mongodb 클라이언트를 사용하여 거의 똑같이 쉽습니다.
  - loadMovieDetailsDataset.js 스크립트를
- "비디오" 데이터베이스 사용 시작

```
비디오 사용;
```

```
db = db.getSiblingDB("동영상")
```

  - ... 두 ... 를 달성할 수 있습니다.
- 다음 ...

```
db.<collection_name>.<operation_api>(매개변수)
```

랩

직접 이 쿼리를 사용해 보십시오 – Shell 및 Compass

- movieDetails 컬렉션을 탐색 한 후 쿼리를 실행하여 다음 질문에 답하십시오.
  - movieDetails 컬렉션에서 정확히 2개의 수상작과 2개의 수상 후보작이 있는 영화는 몇 편입니까 ?
  - 10개의 수상 후보에 오른 PG 영화는 어떻습니까?
    - mongo 쉘을 사용하여 이 질문에 답하는 데 유용한 count() 메서드를 찾을 수 있습니다.
- 이를 위해 Compass와 Mongo Shell을 사용해 보십시오.

랩

직접 이 쿼리를 사용해 보십시오 – Shell 및 Compass

- movieDetails 컬렉션을 탐색 한 후 쿼리를 실행하여 다음 질문에 답하십시오.
  - movieDetails 컬렉션에서 정확히 2개의 수상작과 2개의 수상 후보작이 있는 영화는 몇 편입니까 ?
  - 10개의 수상 후보에 오른 PG 영화는 어떻습니까?
    - mongo 쉘을 사용하여 이 질문에 답하는 데 유용한 count() 메서드를 찾을 수 있습니다.

```
사용 동영상
DB . 영화 세부 정보 . ( { ' Awards.nominations ' : 2 , '상.승리' : 2 }).
카운트 ()
DB . 영화 세부 정보 . 찾기 ({ 평가: "PG" , " Awards.nominations " : 10 }).
카운트 ()
```

배열 필드 쿼리 – 세부 정보 및 바로 가기

- movieDetails 컬렉션

```
사용 동영상
DB . movieDetails. ({ 배우들 : [ " 제프 브리지스" , "기네스 팰트로" ]})
```

- 이렇게 하면 정확히 일치하는 문서가 선택됩니다.
  - 배우의 순서도 일치해야 한다.
  - 추가 출연진이 있어서 실제 영화도 일치하지 않는다.

```
DB . 영화 세부 정보 . ( { 제목 : "아이언맨" })
```

- Jeff Bridges가 출연하는 영화를 선택하고 싶다면 어떻게 합니까? 바로 가기를 확인하십시오($in을 사용하지 않음).

```
DB . 영화 세부 정보 . ({ 배우 들 : "제프 브리지스" })
```

- 아니면 그가 스타인 영화에서

```
DB . 영화 세부 정보 . 찾기 ({ 'actors.0' : "제프 브리지스" })
```

랩

직접 이 쿼리를 사용해 보십시오 – Shell 및 Compass

- movieDetails 를 탐색한 후 쿼리를 실행하여 다음 질문에 답하십시오.
  - "Ethan Coen"과 "Joel Coen"의 두 작가를 순서대로 나열한 문서는 몇 개입니까?
  - 가족 중심 영화는 몇 편의 영화입니까?
  - 두 번째 출품작으로 서부영화로 등재된 영화는 몇 편인가요?
    - mongo 셸을 사용하여 이 질문에 답하는 데 유용한 count() 메서드를 찾을 수 있습니다 .

랩

직접 이 쿼리를 사용해 보십시오 – Shell 및 Compass

- movieDetails 컬렉션을 탐색한 다음 쿼리를 실행하여 다음 질문에 답하십시오.
  - "Ethan Coen"과 "Joel Coen"의 두 작가를 순서대로 나열한 문서는 몇 개입니까?
  - 가족 중심 영화는 몇 편의 영화입니까?
  - 두 번째 출품작으로 서부영화로 등재된 영화는 몇 편인가요?
    - mongo 셸을 사용하여 이 질문에 답하는 데 유용한 count() 메서드를 찾을 수 있습니다 .

사용 동영상

```
DB . 영화 세부 정보 . ({ writers : [ "Ethan Coen" , "Joel Coen" ] }). 카운트 ()
DB . 영화 세부 정보 . ({ 장르 : '가족' }). 카운트 ()

DB . 영화 세부 정보 . 찾기 ({ '장르.1' : '서양' }). 카운트 ()
```

커서 소개

- find（） 연산자는 커서를 반환합니다.
  - 셸에서 20개의 문서가 자동으로 인쇄됩니다.
  - 쉘에서 다음 20개를 얻기 위해 입력 할 수 있습니다.
- 셸에서 모든 문서를 반환하고 이를 사용하여 계속합니다.

```
db.movieDetails.find()
```


```
        "actors" : [
                "Peter Cushing",
                "Bernard Cribbins",
                "Ray Brooks",
                "Andrew Keir"
        ],
        "plot" : "The Daleks' fiendish plot in 2150 against Earth and its people is foiled when Dr. Who and friends arrive from the 20th century and
figure it out.",
        "poster" : "http://ia.media-imdb.com/images/M/MV5BMTg0OTc0MDc0M15BMl5BanBnXkFtZTgwMTg5ODYxMTE@._V1_SX300.jpg",
        "imdb" : {
                "id" : "tt0060278",
                "rating" : 6,
                "votes" : 2236
        },
        "awards" : {
                "wins" : 0,
                "nominations" : 0,
                "text" : ""
        },
        "type" : "movie"
}
Type "it" for more
MongoDB Enterprise >
```

예측

- 프로젝션을 사용하여 find() 쿼리에서 반환할 특정 필드만 선택
- find() 연산자에 대한 두 번째 인수
  - 해당 필드를 명시적으로 선택하려면 1을 사용하십시오.
  - 해당 필드를 명시적으로 제거하려면 0을 사용하십시오.
- 쉘에서 다음 명령을 시도하십시오.

```
db.movieDetails.find (
{ 장르:
{ $in: ["액션", "모험"]}
},
// 제목 필드만 선택하는 프로젝션
{ 제목: 1}
)
```

예측

- 기본 필드 _id가 자동으로 포함됩니다.
  - 이 필드를 어떻게 명시적으로 제외할 수 있습니까?

예측

- 기본 필드 _id가 자동으로 포함됩니다.
  - 이 필드를 어떻게 명시적으로 제외할 수 있습니까?

```
db.movieDetails.find (
{ 장르:
{ $in: ["액션", "모험"]}
},
// 제목 필드만 선택하는 프로젝션
// 명시적으로 _id 필드 제거
{ 제목: 1, _id: 0}
)
```

- 과제: 영화가 find() 필터에 포함된 이유를 알 수 있도록 장르 포함

몽고DB

업데이트

동적이고 유연한 스키마

- MongoDB의 핵심 장점 중 하나는 스키마가 동적이고 유연하다는 것입니다.
  - 모든 문서에 동일한 필드가 있어야 하는 것은 아닙니다.
  - 즉, NULL 값이 있는 필드가 필요하지 않습니다.
- movieDetails를 살펴보겠습니다 . 많은 문서에서 어워드 및 포스터와 같은 필드가 누락되었음을 알 수 있습니다.
- movieDetails 에서 영화 제목 "The Martian"을 찾습니다.

```
사용 동영상
DB . 영화 세부 정보 . ({ 제목 : "화성인" })
```

- 이 문서에는 포스터 사진에 대한 링크가 없습니다.

updateOne () 으로 문서 업데이트

- 영화에 포스터 필드를 추가하기 위해 updateOne () 연산자를 사용할 것입니다 – The Martian
- 업데이트 연산자를 사용하면 일치하는 문서에서 필드를 수정하는 방법을 지정할 수 있습니다.
- 첫 번째 매개변수는 필터를 지정합니다.
- 두 번째 매개변수는 업데이트 유형을 지정합니다.
- 여기서는 $set 연산자를 사용합니다.
  - 쓰기 또는 바꾸기로 생각하십시오.
  - 필드가 존재하지 않는 경우 작성합니다.
  - 존재하는 경우 $set은 새 데이터로 덮어씁니다.

```
DB . 영화 세부 정보 . 업데이트하나 (
{ 제목: "화성인" },
{ $세트:
{ 포스터: "https:// www.imdb.com /title/tt3659388/ mediaviewer /rm1391324160?ref_= tt_ov_i " }
}
)
```

MongoDB 업데이트 연산자 - $ inc

- 업데이트 연산자를 검토하십시오.
  - https://docs.mongodb.com/manual/reference/operator/update/
- tomato.reviews 및 tomato.userReviews
- 하나를 시도해 봅시다 - $ inc
  - $set을 사용하여 변경만 할 수도 있지만 오류가 발생하기 쉬울 수 있습니다.

```
DB . 영화 세부 정보 . 업데이트하나 (
{ 제목: "화성인" },
{ $ 포함 :
{
      " 토마토.리뷰 " : 3 ,
      " 토마토.사용자 리뷰 " : 25
}
}
)
```

MongoDB 업데이트 연산자 - $push

- 배열에서 작동하는 다른 것을 시도해 봅시다.
  - join()은 모든 요소를 단일 문자열로 결합합니다.

```
허락하다 리뷰텍스트1 = [
    "마션은 슬픈 드라마 영화가 될 수도 있었지만 ,
    "약간의 드라마가 추가된 재미있는 영화. 화성인은 무엇입니까" ,
    "모두가 우주 모험을 원합니다. Ridley Scott은 여전히 훌륭하게 만들 수 있습니다. " ,
    "영화와 이것은 최고 중 하나입니다"
]. 가입 ()

DB . 영화 세부 정보 . 업데이트하나 (
{ 제목: "화성인" },
{ 푸시:
{ 리뷰:
{
        평가: 4.5 ,
        날짜: ISODate ( "2016-01-12T09:00:00Z" ),
        검토자: "스펜서 H." ,
        텍스트: reviewText1
}
}
}
)
```

방금 추가한 리뷰를 찾습니다.

- 도전:

- 방금 추가한 리뷰를 찾습니다.
  - 내 제목을 검색할 수 있습니다.
  - 리뷰 이름을 검색해 보세요.
  - 리뷰 텍스트만 볼 수 있도록 프로젝션 추가

랩

방금 추가한 리뷰를 찾습니다.

- 도전:

- 방금 추가한 리뷰를 찾습니다.
  - 내 제목을 검색할 수 있습니다.
  - 리뷰 이름을 검색해 보세요.
  - 리뷰 텍스트만 볼 수 있도록 프로젝션 추가

```
db.movieDetails.find ({" reviews.reviewer ": "Spencer H."});

db.movieDetails.find (
{" reviews.reviewer ": "스펜서 H."},
{" 리뷰.텍스트 ": 1, _id: 0}
);
```

랩

updateMany () 로 여러 문서 업데이트

- updateMany () 를 사용하면 필터 기준과 일치하는 여러 문서를 업데이트할 수 있습니다.
- 많은 문서에 null 등급 필드가 포함되어 있음을 관찰하십시오.
- 제거하자

```
DB . 영화 세부 정보 .
updateMany (
// 이것은 필터입니다.
{ 평가: 널 },
// 이것은 업데이트입니다
{ $미설정:
{ 평가: "" }
}
)
```

- 과제: 모든 빈 포스터 필드 제거

랩

updateMany () 로 여러 문서 업데이트

- 당신은 시도: 모든 빈 포스터 필드를 제거

```
DB . 영화 세부 정보 .
updateMany (
// 이것은 필터입니다.
{ 포스터: 널 },
// 이것은 업데이트입니다
{ $미설정:
   { 포스터: "" }
}
)
```

findOne ()을 사용하여 메모리에 복사

- findOne ()을 사용하여 문서를 메모리에 넣습니다.
  - find()는 커서를 반환하고 findOne ()은 첫 번째로 일치하는 문서를 반환합니다.
  - 메모리의 문서 업데이트
- 문서를 다시 "찾아보기"
  - 실제 문서에는 업데이트된 내용이 없습니다.

```
detailDoc = db . 영화 세부 정보 . findOne ({ " imdb.id " :
"tt4368814" });
detailDoc . 포스터 ;
detailDoc . 포스터 = "https:// www.imdb.com /title/tt4368814/
mediaviewer /rm2926634240" ;
detailDoc . 장르 ;
detailDoc . 장르 . 푸시 ( "다큐멘터리" );

DB . 영화 세부 정보 . findOne ({ " imdb.id " : "tt4368814" });
```

replaceOne () 으로 바꾸기

- 내 메모리 내 문서로 대체하려면 replaceOne ()을 사용하십시오

```
DB . 영화 세부 정보 . 교체 하나 (
{ " imdb.id " : detailDoc . imdb . 아이디 },
  detailDoc
);

DB . 영화 세부 정보 . findOne ({ " imdb.id " : "tt4368814" });
```

- 지금 업데이트되니 참고하세

```
video> db.movieDetails.findOne({"imdb.id": "tt4368814"});
{
  _id: ObjectId("648dd5041207b2f168fa98bc"),
  title: 'Louis C.K.: Live at the Comedy Store',
  year: 2015,
  rated: null,
  runtime: 66,
  countries: [ 'USA' ],
  genres: [ 'Comedy', 'Documentary' ],
  director: 'Louis C.K.',
  writers: [ 'Louis C.K.', 'Jay London' ],
  actors: [ 'Louis C.K.', 'Jay London' ],
  plot: 'Comedian Louis C.K. performs live at the Comedy store in LA.',
  poster: 'https://www.imdb.com/title/tt4368814/mediaviewer/rm2926634240',
  imdb: { id: 'tt4368814', rating: 7.8, votes: 1402 },
  awards: {
    wins: 0,
    nominations: 1,
    text: 'Won 1 Primetime Emmy. Another 1 nomination.'
  },
  type: 'movie'
}
```

몽고DB

삭제

스크래치 데이터 세트 로드

- 도전:
  - `loadReviewsDataset.js`
  - 이렇게 하면 비디오 데이터베이스에 리뷰 컬렉션이 생성됩니다.
  - 새 컬렉션이 있는지 확인

- 삭제 방법 검토
  - https://www.mongodb.com/docs/manual/reference/delete-methods/

스크래치 데이터 세트 로드

- 도전:
  - loadReviewsDataset.js
  - 이렇게 하면 비디오 데이터베이스에 리뷰 컬렉션이 생성됩니다.
  - 새 컬렉션이 있는지 확인

```
video> load("loadReviewsDataset.js")
true
video> show collections;
movieDetails
moviesScratch
reviews
```

랩

MongoDB deleteOne () 작업

- deleteOne () 메서드를 사용 하면 문서를 삭제할 수 있습니다.
- deleteOne () 에 대한 인수는 필터입니다.
- 나침반 필터 또는 find() 메서드를 사용하여 리뷰 컬렉션 검토
  - ObjectId
  - ObjectId를 필터로 사용

```
DB . 리뷰 . deleteOne ({ _id: ObjectId ( "< your_object_id >
" )})
```

```
video> db.reviews.deleteOne({_id: ObjectId("648ef8ea1207b2f168fa996f")})
{ acknowledged: true, deletedCount: 1 }
```

랩

MongoDB deleteMany () 작업

- 에서 검토할 문서를 찾습니다.
  - `{ reviewer_id : 759723314 }`
- 먼저 deleteOne ()을 시도하여 모든 리뷰를 삭제합니다.
- find()를 사용하여 삭제했는지 확인하십시오.
  - 리뷰 중 1개만 삭제되었습니다.
  - 처음 발견된 것은 삭제
- 이번에는 deleteMany ()를 사용합니다.
  - 이번에 모두 삭제되었는지 확인

```
DB . 리뷰 . deleteOne ({ _id: 객체 ID ( "5c754f064ad41406ab794dfa
" )})
DB . 리뷰 . deleteMany ({ reviewer_id : 759723314 })
```

MongoDB deleteMany () 작업

- 에서 검토할 문서를 찾습니다.
  - { reviewer_id : 759723314 }
- 먼저 deleteOne ()을 시도하여 모든 리뷰를 삭제합니다.
- find()를 사용하여 삭제했는지 확인하십시오.
  - 리뷰 중 1개만 삭제되었습니다.
  - 처음 발견된 것은 삭제
- 이번에는 deleteMany ()를 사용합니다.
  - 이번에 모두 삭제되었는지 확인

```
DB . 리뷰 . deleteOne ({ _id: 객체 ID ( "5c754f064ad41406ab794dfa
" )})
DB . 리뷰 . deleteMany ({ reviewer_id : 759723314 })
```

MongoDB deleteMany () 작업

```
video> db.reviews.find({reviewer_id: 759723314}).count()
3
video> db.reviews.deleteOne({reviewer_id: 759723314})
{ acknowledged: true, deletedCount: 1 }
video> db.reviews.find({reviewer_id: 759723314}).count()
2
video> db.reviews.deleteMany({reviewer_id: 759723314})
{ acknowledged: true, deletedCount: 2 }
video> db.reviews.find({reviewer_id: 759723314}).count()
0
```

몽고DB

쿼리 연산자

쿼리 연산자

- 모든 CRUD 작업은 필터를 첫 번째 매개변수로 사용합니다.
- 쿼리 연산자를 사용하면 이러한 필터에 사용할 수 있는 복잡한 쿼리를 작성할 수 있습니다.
- 여기에서 사용할 수 있는 많은 쿼리 연산자를 검토하십시오.
  - https://docs.mongodb.com/manual/reference/operator/query/