KB금융그룹 | 국민의 평생 금융파트너

2025년 상반기 K-디지털 트레이닝

쿼리 작성하기

[KB] IT's Your Life

KB 국민은행

상품, 카테고리, 리뷰

```
product = db.products.findOne({'slug': 'wheel-barrow-9092'})
db.categories.findOne({'_id': product['main_cat_id']})
db.reviews.find({'product_id': product['_id']})
```

전자상거래 쿼리

skip, limit 그리고 쿼리 옵션

- skip(n): 건너뛸 데이터 개수
- limit(m): 추출할 데이터 개수
- sort({키1:1, 키2: -1, ... }): 정렬. 1(오름차순), -1(내림차순)

→ 연결 순서는 상관 없음

```
db.reviews.find({'product_id': product['_id']}).skip(0).limit(12)

product  = db.products.findOne({'slug': 'wheel-barrow-9092'})
db.reviews.find({'product_id': product['_id']})
                .sort({'helpful_votes': -1})
                .limit(12)
```

skip, limit 그리고 쿼리 옵션

페이지네이션

```
page_number = 1
product  = db.products.findOne({'slug': 'wheel-barrow-9092'})
category = db.categories.findOne({'_id': product['main_cat_id']})
reviews_count = db.reviews.count({'product_id': product['_id']})
reviews = db.reviews.find({'product_id': product['_id']})
                    .skip((page_number - 1) * 12)
                    .limit(12)
                    .sort({'helpful_votes': -1})
```

전자상거래 쿼리

상품 리스트 페이지

```
page_number = 1
category = db.categories.findOne({'slug': 'outdoors'})
siblings = db.categories.find({'parent_id': category['_id']}) //  'outdoors' 상품과 같은 카테고리에 있는 상품 얻기
products = db.products.find({'category_id': category['_id']})
                      .skip((page_number - 1) * 12)
                      .limit(12)
                      .sort({'helpful_votes': -1})
```

최상위 상품 카테고리 얻기

```
categories = db.categories.find({'parent_id': null})
```

MongoDB의 쿼리 언어

질의 조건과 셀렉터

AND 조건

```
db.users.find({'last_name': "Banker"})
db.users.find({'first_name': "Smith", birth_year: 1975})
```

범위

연산자	설명
$lt	~보다 작은
$gt	~보다 큰
$lte	~보다 작거나 같은
$gte	~보다 크거나 같은

```
db.users.find({'birth_year': {'$gte': 1985}, 'birth_year': {'$lte': 2015}})

db.users.find({'birth_year': {'$gte': 1985, '$lte': 2015}})
```

MongoDB의 쿼리 언어

질의 조건과 셀렉터

범위

- 테스트 데이터

```
db.items.insert({ "_id" : ObjectId("4caf82011b0978483ea29ada"), "value" : 97 })
db.items.insert({ "_id" : ObjectId("4caf82031b0978483ea29adb"), "value" : 98 })
db.items.insert({ "_id" : ObjectId("4caf82051b0978483ea29adc"), "value" : 99 })
db.items.insert({ "_id" : ObjectId("4caf820d1b0978483ea29ade"), "value" : "a" })
db.items.insert({ "_id" : ObjectId("4caf820f1b0978483ea29adf"), "value" : "b" })
db.items.insert({ "_id" : ObjectId("4caf82101b0978483ea29ae0"), "value" : "c" })
```

- 범위 질의

```
db.items.find({'value': {'$gte': 97}})  // 정수에 대한 검사이므로 정수 값 데이터만 출력

db.items.find({'value': {'$gte': "a"}}) // 문자열ㄹ에 대한 검사이므로 문자열 데이터만 출력
```

MongoDB의 쿼리 언어

집합 연산자

연산자	설명
$in	어떤 인수든 하나라도 참고 집합에 있는 경우 일치
$all	모든 인수가 참고 집합에 있고 배열이 포함된 도큐먼트에서 사용되는 경우 일치
$nin	그 어떤 인수도 참고 집합에 있지 않을 경우 일치

```
db.products.find({
    'main_cat_id': {
      '$in': [
        ObjectId("6a5b1476238d3b4dd5000048"),
        ObjectId("6a5b1476238d3b4dd5000051"),
        ObjectId("6a5b1476238d3b4dd5000057")
      ]
    }
  })
```

MongoDB의 쿼리 언어

부울 연산자

연산자	설명
$ne	인수가 요소와 같지 않은 경우 일치
$not	일치 결과를 반전시킴(반대로 만듦)
$or	제공된 검색어 집합 중 하나라도 TRUE인 경우 일치
$nor	제공된 검색어 집합 중 그 어떤 것도 TRUE가 아닌 경우 일치
$and	제공된 검색어 집합이 모두 TRUE인 경우 일치
$exists	요소가 도큐먼트 안에 존재할 경우 일치

MongoDB의 쿼리 언어

부울 연산자

```
db.products.find({
    '$or': [
      {'details.color': 'blue'},
      {'details.manufacturer': 'ACME'}
    ]
  })

db.products.find({
    $and: [
      {
        tags: {$in: ['gift', 'holiday']}
      },
      {
        tags: {$in: ['gardening', 'landscaping']}
      }
    ]
  })
```

MongoDB의 쿼리 언어

부울 연산자

```
db.products.find({'details.color': {$exists: false}})

db.products.find({'details.color': {$exists: true}})
```

MongoDB의 쿼리 언어

배열

연산자	설명
$elemMatch	제공된 모든 조건이 동일한 하위 도큐먼트에 있는 경우 일치
$size	배열 하위 도큐먼트의 크기가 제공된 리터럴 값과 같으면 일치

```
{
    _id: ObjectId("4c4b1476238d3b4dd5003981"),
    slug: "wheel-barrow-9092",
    sku: "9092",
    tags: ["tools", "equipment", "soil"]
}
```

db.products.find({tags: "soil"})
db.products.find({'tags.0': "soil"})

MongoDB의 쿼리 언어

배열

```
{
    _id: ObjectId("4c4b1476238d3b4dd5000001")
    username: "kbanker",
    addresses: [
        {
            name: "home",
            street: "588 5th Street",
            city: "Brooklyn",
            state: "NY",
            zip: 11215
        },
        {
            name: "work",
            street: "1 E. 23rd Street",
            city: "New York",
            state: "NY",
            zip: 10010
        },
    ]
}
```

```
db.users.find({'addresses.0.state': "NY"})

db.users.find({'addresses.state': "NY"})

db.users.find({'addresses.name': 'home', 'addresses.state': 'NY'})

db.users.find({
    'addresses': {
      '$elemMatch': {
        'name': 'home',
        'state': 'NY'
      }
    }
  })
```