arrayElemAt

```
db.restaurants.find().limit(2)
```

restaurants 0.064 sec.


```
/* 1 */
{
    "_id" : ObjectId("5f040a018b6d6655e5d47266"),
    "address" : {
        "building" : "469",
        "coord" : [
            -73.961704,
            40.662942
        ],
        "street" : "Flatbush Avenue",
        "zipcode" : "11225"
    },
    "borough" : "Brooklyn",
    "cuisine" : "Hamburgers",
    "grades" : [
        {
            "date" : ISODate("2014-12-30T00:00:00.000Z"),
            "grade" : "A",
            "score" : 8
        },
        {
            "date" : ISODate("2014-07-01T00:00:00.000Z"),
            "grade" : "B",
            "score" : 23
        },
        {
            "date" : ISODate("2013-04-30T00:00:00.000Z"),
            "grade" : "A",
            "score" : 12
        },
        {
            "date" : ISODate("2012-05-08T00:00:00.000Z"),
            "grade" : "A",
            "score" : 12
        }
    ],
    "name" : "Wendy'S",
    "restaurant_id" : "30112340"
```


> 마지막 도큐먼트는 가장 오래된(최초) 평가, 첫 번째 도큐먼트는 가장 최신 평가

```
db.restaurants.aggregate([
{$match: {"cuisine": "Italian", "borough": "Brooklyn"}},
{$project: {
    _id: 0,
    cuisine: 1,
    name: 1,
    last_award: {$arrayElemAt: ["$grades", 0]},
    first_award: {$arrayElemAt: ["$grades", -1]}
    }}
])
```


```
/* 1 */
{
    "cuisine" : "Italian",
    "name" : "Philadelhia Grille Express",
    "last_award" : {
        "date" : ISODate("2014-02-25T00:00:00.000Z"),
        "grade" : "A",
        "score" : 12
    },
    "first_award" : {
        "date" : ISODate("2011-11-09T00:00:00.000Z"),
        "grade" : "A",
        "score" : 12
    }
}

/* 2 */
{
    "cuisine" : "Italian",
    "name" : "New Corner",
    "last_award" : {
        "date" : ISODate("2014-12-04T00:00:00.000Z"),
        "grade" : "A",
        "score" : 11
    },
    "first_award" : {
        "date" : ISODate("2011-12-19T00:00:00.000Z"),
        "grade" : "A",
        "score" : 12
    }
}
```

> -1은 배열의 마지막 번 째 지칭

```
db.restaurants.aggregate([
{$match: {"cuisine": "Italian", "borough": "Brooklyn"}},
{$project: {
    _id: 0,
    cuisine: 1,
    name: 1,
    last_award: {$slice: ["$grades", 0, 2]}
    }}
])
```

restaurants 0,039 sec.

```
/* 1 */
{
    "cuisine" : "Italian",
    "name" : "Philadelhia Grille Express",
    "last_award" : [
        {
            "date" : ISODate("2014-02-25T00:00:00.000Z"),
            "grade" : "A",
            "score" : 12
        },
        {
            "date" : ISODate("2013-06-27T00:00:00.000Z"),
            "grade" : "A",
            "score" : 7
        }
    ]
}

/* 2 */
{
    "cuisine" : "Italian",
    "name" : "New Corner",
    "last_award" : [
        {
            "date" : ISODate("2014-12-04T00:00:00.000Z"),
            "grade" : "A",
            "score" : 11
        },
        {
            "date" : ISODate("2014-02-19T00:00:00.000Z"),
            "grade" : "A",
            "score" : 10
        }
```

> slice로 최신 2개 평가 출력

```
db.restaurants.aggregate([
{$match: {"cuisine": "Italian", "borough": "Brooklyn"}},
{$project: {
    _id: 0,
    cuisine: 1,
    name: 1,
    total_award: {$size: "$grades"}
    }}
])
```

restaurants 0.037 sec.

```
/* 1 */
{
    "cuisine" : "Italian",
    "name" : "Philadelhia Grille Express",
    "total_award" : 4
}

/* 2 */
{
    "cuisine" : "Italian",
    "name" : "New Corner",
    "total_award" : 5
}
```

> size로 평가를 총 몇 번 받았는 지 출력

accumulator 연산자

- sum
- max / min
- avg

```
db.restaurants.aggregate([
{$match: {"cuisine": "Italian", "borough": "Brooklyn"}},
{$project: {
    _id: 0,
    cuisine: 1,
    name: 1,
    max_award: {$max: "$grades.score"},
    }}
    ])
```

restaurants 0.042 sec.

```
/* 1 */
{
    "cuisine" : "Italian",
    "name" : "Philadelhia Grille Express",
    "max_award" : 12
}

/* 2 */
{
    "cuisine" : "Italian",
    "name" : "New Corner",
    "max_award" : 12
}

/* 3 */
{
    "cuisine" : "Italian",
    "name" : "Gargiulo'S Restaurant",
    "max_award" : 13
}
```

> max

```
db.restaurants.aggregate([
{$match: {"cuisine": "Italian", "borough": "Brooklyn"}},
{$project: {
    _id: 0,
    cuisine: 1,
    name: 1,
    max_award: {$sum: "$grades.score"},
    }}
    ])
```

restaurants 0,051 sec.

```
/* 1 */
{
    "cuisine" : "Italian",
    "name" : "Philadelhia Grille Express",
    "max_award" : 41
}

/* 2 */
{
    "cuisine" : "Italian",
    "name" : "New Corner",
    "max_award" : 52
}

/* 3 */
{
    "cuisine" : "Italian",
    "name" : "Gargiulo'S Restaurant",
    "max_award" : 43
}
```

> sum

```
db.restaurants.aggregate([
{$match: {"cuisine": "Italian", "borough": "Brooklyn"}},
{$project: {
    _id: 0,
    cuisine: 1,
    name: 1,
    max_award: {$avg: "$grades.score"},
    }}
    ])
```

restaurants 0.039 sec.

```
/* 1 */
{
    "cuisine" : "Italian",
    "name" : "Philadelhia Grille Express",
    "max_award" : 10.25
}

/* 2 */
{
    "cuisine" : "Italian",
    "name" : "New Corner",
    "max_award" : 10.4
}

/* 3 */
{
    "cuisine" : "Italian",
    "name" : "Gargiulo'S Restaurant",
    "max_award" : 10.75
}
```

> avg

grouping

```
(c) 2019 Microsoft Corporation. All rights reserved.

C:₩Users₩BIT>d:

D:₩>cd khy

D:₩KHY>cd MongoDB

D:₩KHY₩MongoDB>mongoimport --db test --collection companies --drop --file D:₩KHY₩MongoDB_work₩companies.json
2020-07-08T09:36:12.016+0900    Failed: open D:₩KHY₩MongoDB_work₩companies.json: The system cannot find the file specified.

2020-07-08T09:36:12.168+0900    0 document(s) imported successfully. 0 document(s) failed to import.

D:₩KHY₩MongoDB>mongoimport --db test --collection companies --drop --file D:₩KHY₩MongoDB_work₩companies.json
2020-07-08T09:38:28.665+0900    connected to: mongodb://localhost/
2020-07-08T09:38:28.740+0900    dropping: test.companies
2020-07-08T09:38:31.431+0900    18801 document(s) imported successfully. 0 document(s) failed to import.

D:₩KHY₩MongoDB>
```

```
db.companies.find()
```

companies 0.011 sec.


```
/* 1 */
{
    "_id" : ObjectId("52cdef7c4bab8bd675297d8b"),
    "name" : "AdventNet",
    "permalink" : "abc3",
    "crunchbase_url" : "http://www.crunchbase.com/company/adventnet",
    "homepage_url" : "http://adventnet.com",
    "blog_url" : "",
    "blog_feed_url" : "",
    "twitter_username" : "manageengine",
    "category_code" : "enterprise",
    "number_of_employees" : 600,
    "founded_year" : 1996,
    "deadpooled_year" : 2,
    "tag_list" : "",
    "alias_list" : "Zoho ManageEngine ",
    "email_address" : "pr@adventnet.com",
    "phone_number" : "925-924-9500",
    "description" : "Server Management Software",
    "created_at" : ISODate("2007-05-25T19:24:22.000Z"),
    "updated_at" : "Wed Oct 31 18:26:09 UTC 2012",
    "overview" : "<p>AdventNet is now <a href=\"/company/zoho-manageengine\" title=\"Zoho ManageEngine\" rel=\"nof
    "image" : {
        "available_sizes" : [
```


```
db.companies.aggregate([
{$group: {
    _id: {founded_year: "$founded_year"},
    average_number_of_employees: {$avg: "$number_of_employees"}
    }},
{$sort: {average_number_of_employees: -1}}
])
```

companies 0.056 sec.


```
/* 1 */
{
    "_id" : {
        "founded_year" : 1847
    },
    "average_number_of_employees" : 405000.0
}

/* 2 */
{
    "_id" : {
        "founded_year" : 1896
    },
    "average_number_of_employees" : 388000.0
}

/* 3 */
{
    "_id" : {
        "founded_year" : 1933
    },
    "average_number_of_employees" : 320000.0
}
```


> 그룹을 작성하려면 들어가면 무조건 아이디 필드가 들어가야 함

```
db.companies.aggregate([
{$match: {"relationships.person": {$ne: null}}},
{$project: {relationships: 1, _id: 0}},
{$unwind: "$relationships"},
{$group: {
    _id: "$relationships.person",
    count: {$sum: 1}
    }},
{$sort: {count: -1}}
])
```

companies 0.463 sec.

```
/* 1 */
{
    "_id" : {
        "first_name" : "Tim",
        "last_name" : "Hanlon",
        "permalink" : "tim-hanlon"
    },
    "count" : 28.0
}

/* 2 */
{
    "_id" : {
        "first_name" : "David S.",
        "last_name" : "Rose",
        "permalink" : "david-s-rose"
    },
    "count" : 24.0
}

/* 3 */
{
    "_id" : {
        "first_name" : "Saul",
        "last_name" : "Klein",
        "permalink" : "saul-klein"
    },
    "count" : 24.0
```

> ne : not equal
>
> unwind : 배열[]로 한 항목으로 되어있는 것을 개별의 도큐먼트로 쪼개기
>
> count: {$sum: 1}은 카운트을 하나씩 하겠다는 코드
>
> $sort: {count: -1} 카운트를 내림차순으로 정렬하겠다는 코드

```
db.companies.aggregate([
{$match: {founded_year: {$gte: 2010}}},
{$group: {
    _id: {founded: "$founded_year"},
    companies: {$push: "$name"}
    }},
{$sort: {"_id.founded": -1}}
])
```

companies 0.044 sec.

```
/* 1 */
{
    "_id" : {
        "founded" : 2013
    },
    "companies" : [
        "Fixya",
        "Wamba",
        "Advaliant",
        "Fluc",
        "iBazar",
        "Gimigo",
        "SEOGroup",
        "Clowdy",
        "WhosCall",
        "Pikk",
        "Tongxue",
        "Shopseen",
        "VistaGen Therapeutics"
    ]
}

/* 2 */
{
    "_id" : {
        "founded" : 2012
    },
    "companies" : [
        "PeekYou",
        "headr",
        "Pinger",
        "Widgetbox",
        "Mobiluck",
        "Skydeck",
        "Simplicant",
        "Springleap",
        "Jumbuck Entertainment",
        "Carfeine",
        "Bling Easy",
        "FoodCare",
```

> 컬럼 기준으로 오른쪽에 필드명을 넣을 땐 $ 사용

> push : 배열을 생성해서 값을 하나씩 집어 넣음

> grouping 하는 과정에서 projection을 같이 해주는 코드라고 보면 됨

```
db.companies.aggregate([
{$match: {founded_year: {$gte: 2010}}},
{$group: {
    _id: {founded_year: "$founded_year", category_code: "$category_code"},
    companies: {$push: "$name"}
    }},
{$sort: {"_id.founded_year": -1}}
])
```

companies 0.034 sec.

```
/* 1 */
{
    "_id" : {
        "founded_year" : 2013,
        "category_code" : "ecommerce"
    },
    "companies" : [
        "iBazar",
        "Shopseen"
    ]
}

/* 2 */
{
    "_id" : {
        "founded_year" : 2013,
        "category_code" : "advertising"
    },
    "companies" : [
        "Advaliant",
        "SEOGroup"
    ]
}

/* 3 */
{
    "_id" : {
        "founded_year" : 2013,
        "category_code" : "mobile"
    },
    "companies" : [
        "WhosCall"
    ]
}
```

> 그룹핑 한 필드가 2개

```
db.companies.findOne({"ipo": {"$exists": true, "$ne": null}})
```

0.002 sec.


```
/* 1 */
{
    "_id" : ObjectId("52cdef7c4bab8bd675297d94"),
    "name" : "Twitter",
    "permalink" : "twitter",
    "crunchbase_url" : "http://www.crunchbase.com/company/twitter",
    "homepage_url" : "http://twitter.com",
    "blog_url" : "http://twitter.com/blog",
    "blog_feed_url" : "http://feeds.feedburner.com/TwitterBlog",
    "twitter_username" : "twitter",
    "category_code" : "social",
    "number_of_employees" : 1300,
    "founded_year" : 2006,
    "founded_month" : 3,
    "founded_day" : 21,
    "deadpooled_year" : null,
    "deadpooled_month" : null,
    "deadpooled_day" : null,
    "deadpooled_url" : "",
    "tag_list" : "text, messaging, social, community, twitter, tweet, twttr, microb
    "alias_list" : "",
    "email_address" : "press@twitter.com",
    "phone_number" : "",
    "description" : "Real time communication platform",
    "created_at" : "Fri Jun 01 08:42:34 UTC 2007",
    "updated_at" : "Sat Dec 07 16:07:56 UTC 2013",
    "overview" : "<p>Created in 2006, Twitter is a global real-time communications
    "image" : {
        "available_sizes" : [
            [
                [
                    150,
                    150
                ],
                "assets/images/resized/0000/2755/2755v33-max-150x150.png"
            ],
            [
                [
                    250,
                    250
                ],
                "assets/images/resized/0000/2755/2755v33-max-250x250.png"
            ],
            [
                [
                    300
```


> 필드가 존재 하는 지 찾기
>
> ipo 필드를 갖고 있는 도큐먼트를 출력한 모습

```
db.companies.aggregate([
    {$group: {
        _id: {ipo_year: "$ipo.pub_year"},
        companies: {$push: "$name"}
        }},
    {$sort: {"_id.ipo_year": 1}}
    ])
```

companies 0.063 sec.


```
/* 1 */
{
    "_id" : {
        "ipo_year" : null
    },
    "companies" : [
        "AdventNet",
        "Wetpaint",
        "Zoho",
        "Omnidrive",
        "Postini",
        "Flektor",
        "Geni",
        "Digg",
        "Fox Interactive Media",
        "Gizmoz",
        "StumbleUpon",
        "Scribd",
        "Slacker",
        "Lala",
        "Helio",
        "MeetMoi",
        "Joost",
        "CBS",
        "Babelgum",
        "Viacom",
        "Plaxo",
        "Powerset",
        "Technorati",
        "SpinVox",
        "AddThis",
```


> null인 값이 있어서 출력 또한 ipo_year가 null로 출력되는 모습

```
db.companies.aggregate([
    {$match: {"ipo.pub_year": {"$exists": true, "$ne": null}}},
    {$group: {
        _id: {ipo_year: "$ipo.pub_year"},
        companies: {$push: "$name"}
        }},
    {$sort: {"_id.ipo_year": 1}}
    ])
```

companies 0.052 sec.

```
/* 1 */
{
    "_id" : {
        "ipo_year" : 1969
    },
    "companies" : [
        "The Walt Disney Company"
    ]
}

/* 2 */
{
    "_id" : {
        "ipo_year" : 1971
    },
    "companies" : [
        "Waste Management"
    ]
}

/* 3 */
{
    "_id" : {
        "ipo_year" : 1978
    },
    "companies" : [
        "Gannett",
        "Intel",
        "Sony",
        "Hewlett-Packard",
        "General Electric",
        "IBM",
        "Texas Instruments",
        "GlaxoSmithKline",
        "Hitachi",
        "Hitachi",
        "Corning",
        "3M"
    ]
```

> match ~ exists ~ ne 구문으로 null값을 제외한 도큐먼트들을 출력한 화면

```
db.companies.aggregate([
    {$group: {
        _id: null,
        count: {$sum: 1}}}
])
```

companies 0.04 sec.

```
/* 1 */
{
    "_id" : null,
    "count" : 18801.0
}
```

> aggregation으로 companies의 도큐먼트 갯수 구하기

> id를 null로 주고 sum을 사용

```
db.companies.aggregate([
        {$group: {
            _id: "$category_code",
            }}
        ])
```

companies 0.054 sec.

```
/* 1 */
{
    "_id" : "enterprise"
}

/* 2 */
{
    "_id" : "social"
}

/* 3 */
{
    "_id" : "other"
}

/* 4 */
{
    "_id" : "nanotech"
}

/* 5 */
{
    "_id" : "finance"
}
```

> 카테고리 코드(카테고리)의 종류들을 알고 싶을 때의 코드와 출력 화면

```
db.companies.aggregate([
        {$match: {funding_rounds: {$ne: []}}},
        {$unwind: "$funding_rounds"},
        {$sort: {"funding_rounds.funded_year": 1,
            "funding_rounds.funded_month": 1,
            "funding_rounds.funded_day": 1}},
        {$group: {
            _id: {company: "$name"},
            funding:{
                $push: {
                    amount: "$funding_rounds.raised_amount",
                    year: "$funding_rounds.funded_year"
                    }
                }
            }}
        ])
```

companies 0.163 sec.

```
/* 1 */
{
    "_id" : {
        "company" : "ReverbNation"
    },
    "funding" : [
        {
            "amount" : 2000000,
            "year" : 2006
        },
        {
            "amount" : 3000000,
            "year" : 2008
        }
    ]
}

/* 2 */
{
    "_id" : {
        "company" : "ibeatyou"
    },
    "funding" : [
        {
            "amount" : 1000000,
            "year" : 2008
        }
    ]
}
```

> project에서 못하는 find를 처리하기 위한 것이 group

```
db.companies.aggregate([
        {$match: {funding_rounds: {$exists: true, $ne: []}}},
        {$unwind: "$funding_rounds"},
        {$sort: {"funding_rounds.funded_year": 1,
            "funding_rounds.funded_month": 1,
            "funding_rounds.funded_day": 1}},
        {$group: {
            _id: {company: "$name"},
            first_round: {$first: "&funding_rounds"},
            last_round: {$last: "$funding_rounds"},
            num_rounds: {$sum: 1},
            total_raised: {$sum: "$funding_rounds.raised_amount"}
            }},
        {$project: {
            _id: 0,
            company: "$_id.company",
            first_round: {
                amount: "$first_round.raised_amount",
```

```
                article: "$first_round.source_url",
                year: "$first_round.funded_year"
                },
            last_round: {
                amount: "$last_round.raised_amount",
                article: "$last_round.source_url",
                year: "$last_round.funded_year"
                },
            num_rounds: 1,
            total_raised: 1
            }},
        {$sort: {total_raised: -1}}
        ])
```

> 프로젝트에서 _id : 0은 출력하지 말라는 뜻

> first_round 부분도 출력 되어야 하는데 코드 오류인 듯

임베디드에서 컬렉션을 나누어 조인 시키기

- 임베디드를 하면 조인을 안 해도 되는 장점이 있음
- 하지만 조인이 필요할 때가 있을 땐 컬렉션을 분리해서 조인을 해줘야 함

```
db.orders.insert([
    {"_id": 1, "item": "almonds", "price": 12, "quantity": 2},
    {"_id": 2, "item": "pecans", "price": 20, "quantity": 1},
    {"_id": 3}
])
db.orders.find()
```

orders 0.002 sec.

```
/* 1 */
{
    "_id" : 1.0,
    "item" : "almonds",
    "price" : 12.0,
    "quantity" : 2.0
}

/* 2 */
{
    "_id" : 2.0,
    "item" : "pecans",
    "price" : 20.0,
    "quantity" : 1.0
}

/* 3 */
{
    "_id" : 3.0
}
```

> insertMany로 사용해야 올바른 작성법

```
db.inventory.insertMany([
    {"_id": 1, "sku": "almonds", description: "product 1", "instock": 120},
    {"_id": 2, "sku": "bread", description: "product 2", "instock": 80},
    {"_id": 3, "sku": "cashews", description: "product 3", "instock": 60},
    {"_id": 4, "sku": "pecans", description: "product 4", "instock": 70},
    {"_id": 5, "sku": null, description: "Incomplete"},
    {"_id": 6}
])
db.inventory.find()
```

inventory 0.001 sec.

```
/* 1 */
{
    "_id" : 1.0,
    "sku" : "almonds",
    "description" : "product 1",
    "instock" : 120.0
}

/* 2 */
{
    "_id" : 2.0,
    "sku" : "bread",
    "description" : "product 2",
    "instock" : 80.0
}

/* 3 */
{
    "_id" : 3.0,
    "sku" : "cashews",
    "description" : "product 3",
    "instock" : 60.0
}
```

lookup

```
db.orders.aggregate([
    {
        $lookup:{
            from: "inventory",
            localField: "item",
            foreignField: "sku",
            as: "inventory_docs"
            }
        }
])
```

orders 0.005 sec.

```
/* 1 */
{
    "_id" : 1.0,
    "item" : "almonds",
    "price" : 12.0,
    "quantity" : 2.0,
    "inventory_docs" : [
        {
            "_id" : 1.0,
            "sku" : "almonds",
            "description" : "product 1",
            "instock" : 120.0
        }
    ]
}

/* 2 */
{
    "_id" : 2.0,
    "item" : "pecans",
    "price" : 20.0,
    "quantity" : 1.0,
    "inventory_docs" : [
        {
            "_id" : 4.0,
            "sku" : "pecans",
            "description" : "product 4",
            "instock" : 70.0
        }
    ]
}
```

> from ~ : 조인 할 컬렉션
>
> localField : 조인 주체 컬렉션의 필드
>
> foreignField : 조인 당하는 컬렉션의 필드
>
> as : 알리아스

> 임베디드 도큐먼트 형식으로 조인이 되는 것을 확인 가능(해당 컬렉션에서 매칭되는 조건의 도큐먼트를 통째로 갖고 조인(임베디드)시켜버림)

array가 있을 때의 lookup

```
db.classes.insertMany([
{_id: 1, title: "Reading is ...", enrollmentlist: ["giraffe2", "pandabear", "artie"], days: ["M", "W", "F"]},
{_id: 2, title: "But Writing ...", enrollmentlist: ["giraffe1", "artie"], days: ["T", "F"]}
])

db.members.insertMany([
{_id: 1, name: "artie", joined: new Date("2016-05-01"), status: "A"},
{_id: 2, name: "giraffe", joined: new Date("2017-05-01"), status: "D"},
{_id: 3, name: "giraffe1", joined: new Date("2017-10-01"), status: "A"},
{_id: 4, name: "panda", joined: new Date("2018-10-11"), status: "A"},
{_id: 5, name: "pandabear", joined: new Date("2018-12-01"), status: "A"},
{_id: 6, name: "giraffe2", joined: new Date("2018-12-01"), status: "D"}
])
```

```
db.classes.aggregate([
{$lookup: {
    from: "members",
    localField: "enrollmentlist",
    foreignField: "name",
    as: "enrollee_info"
    }
    }
])
```

classes 0,003 sec.

```
/* 1 */
{
    "_id" : 1.0,
    "title" : "Reading is ...",
    "enrollmentlist" : [
        "giraffe2",
        "pandabear",
        "artie"
    ],
    "days" : [
        "M",
        "W",
        "F"
    ],
    "enrollee_info" : [
        {
            "_id" : 1.0,
            "name" : "artie",
            "joined" : ISODate("2016-05-01T00:00:00.000Z"),
            "status" : "A"
        },
        {
            "_id" : 5.0,
            "name" : "pandabear",
            "joined" : ISODate("2018-12-01T00:00:00.000Z"),
            "status" : "A"
        },
```