# MONGODB - Tuần 4

db.collection.aggregate([{stage1},{stage2},...,{stageN}])

$match , $project , $group

https://www.mongodb.com/docs/manual/reference/operator/aggregation/

| SQL command | Aggregation framework operator |
|---|---|
| SELECT | $project |
| | $group functions: $sum, $min, $avg, etc. |
| FROM | db.collectionName.aggregate(...) |
| JOIN | $unwind |
| WHERE | $match |
| GROUP BY | $group |
| HAVING | $match |

```
// $match
db.products.find({name: /^AC3 Case/}).sort({price: -1, rating: 1});
db.products.aggregate([
  {$match: {name: /^AC3 Case/}},
  {$sort: {price: -1, rating: 1}}
]);

db.products.aggregate([
  {$match: {name: /^AC3 Case/}},
  {$sort: {price: -1, rating: 1}},
  {$skip: 1},
  {$limit: 2}
]);

db.products.find({name: {$regex: "phone service", $options: 'i'}});
db.products.aggregate([
```

```
  {$match: {name: {$regex: "phone service", $options: 'i'}}}
]);

db.products.find({'type': {$elemMatch: {$eq: "accessory", $ne: "charger"}}});
db.products.aggregate([{$match: {type: {$elemMatch: {$eq: "accessory", $ne: "charger"}}}}]);

// $project
db.products.aggregate([ {$match: {name: {$regex: "phone service", $options: 'i'}}},
                {$project: {name: 1, rating: 1, monthly_price: 1, _id: 0}}
]);

// Arithmetic Operators
// $add, $subtract, $divide, $multiply, $round ...
// {$add ($multiply): [expression1, expression2, ...]}
// {$subtract ($divide): [expression1, expression2]}
// {$round: [expression, no_of_decimals]}

db.products.aggregate([{$project: {name: 1, monthly_price: 1,
                        annual_price: {$multiply: ["$monthly_price",12, 0.9]}}}]);

db.products.aggregate([
                  {$match: {monthly_price: {$ne: null}}},
          {$project: {name: 1,
                  monthly_price: 1,
                  annual_price: {$multiply: ["$monthly_price",12, 0.9]}}}
])

// Array Expression Operators
// $isArray, $arrayElemAt, $concatArrays, $first, $last, $size, $map
```

```
// $map: {input: expression, as: string, in:expression}

// show name, type, for
db.products.aggregate([{$project: {name: 1, type: 1, for: 1}}
]);

// show name, type, and isarray_type
db.products.aggregate([
        {$project: {name: 1, type: 1,
                        isarray_type: {$isArray: "$type"}}}
]);

db.products.aggregate([
        {$project: {name: 1, type: 1,
                        element1: {$first: '$type'}}}
]);

// Show name, type and the first element of type
db.products.aggregate([
        {$match: {$expr: {$isArray: '$type'}}},
        {$project: {name: 1, type: 1, element1: {$first: '$type'}}}
]);

// Show name, type and the second element of type
db.products.aggregate([
        {$match: {$expr: {$isArray: '$type'}}},
        {$project: {name: 1, type: 1,
                        element2nd: {$arrayElemAt: ['$type',1]}}}
]);
```

```
// the second element of type array is 'charger'???
db.products.aggregate([
        {$match: {$expr: {$isArray: '$type'}}},
        {$project: {type: 1}},
        {$match:
                {$expr: {$eq: [{$arrayElemAt: ['$type',1]}, 'charger']}}}
]);

// show all types in UPPERCASE
// $map: {input: expression, as: string, in:expression}
// as: declare a variable
// in: apply toUpper to x variable - - $$: accesses the value of the variable declared here
db.products.aggregate([
  {$match: {$expr: {$isArray: '$type'}}},
  {$project: {
    type: 1,
    type_upper: {$map: {input: '$type', as: 'x', in: {$toUpper: '$$x'}}}}
}]);

// Conditional Expression Operators
// $cond: {$cond: {if: expression, then: true-case, else: false-case}}
// $ifNull: {$ifNull: [expression1, ..., expressionN, replacement-expression-if-null]}

db.products.aggregate([
  {$project: {"monthly_price": 1}}
])

// if monthly_price is null, replace it with = 0
```

```
db.products.aggregate([
  {$project: {monthly_price: {$ifNull: ["$monthly_price", 0]}}
}]);

// if monthly_price < 60, classification = 0; otherwise, classification = 1
db.products.aggregate([
  {$project: {monthly_price: {$ifNull: ["$monthly_price", 0]},
            classification:
               {$cond:{if: {$lt: ['$monthly_price',60]},
                     then: 0, else: 1}}}}
])

// $group
// group by rating and count number of documents for each rating
db.products.aggregate([{$group: {_id:'$rating', count: {$sum: 1}}}]);

// group by availabe, count number of documents, and calculate the average of rating for each
available
db.products.aggregate([
      {$group :
         {_id : '$available',
          count : {$sum : 1},
          avg_rating : {$avg : '$rating'}}
} ])

//  Cập nhật phần làm tròn avg_rating: chúng ta có thể thêm 1 stage mới để thực hiện toán tử
$round nếu trong $group không được phép sử dụng $round
db.products.aggregate([
      {$group :
```

```
        {_id : '$available',
         count : {$sum : 1},
         avg_rating: {$avg : '$rating'}}},
    {$project: {count: 1, avg_rating: {$round: ['$avg_rating', 2]}}}
]);

db.products.aggregate([
    {$match: {available: {$ne: null}}},
    {$group :
        {_id : '$available',
         count : {$sum : 1},
         avg_rating : {$avg : '$rating'},
         min_rating : {$min : '$rating'},
         max_rating : {$max : '$rating'}
        }
} ])
```

**Bài tập**

Database: restaurant.json

Sử dụng **aggregate framework** để thực hiện các câu truy vấn sau:

1. Hiển thị name, stars của 5 nhà hàng đầu tiên có stars > 3, sort theo stars thứ tự từ cao đến thấp.
2. Hiển thị name, stars của 5 nhà hàng tiếp theo sau khi skip 5 nhà hàng đầu tiên có stars > 3, sort theo stars thứ tự từ cao đến thấp và name từ thấp đến cao.
3. Hiển thị tên các nhà hàng có vị trí latitude < -95.754168
4. Hiển thị tên các nhà hàng không có món ăn Californian, điểm stars >= 4, và latitude < -65.754168, sắp xếp theo thứ tự stars giảm dần
5. Hiển thị _id, name, stars, categories, phone, và email của các nhà hàng có tên bắt đầu bằng ‘Wil'
6. Hiển thị _id, name, stars, categories, phone, và email của các nhà hàng có tên kết thúc bằng ‘ces’

7. Hiển thị _id, name, stars, categories, phone, và email của các nhà hàng có tên chứa các kí tự ‘Reg’.
8. Tìm tên, stars, categories các nhà hàng có cả ba món Russian, Chinese và Vietnamese
9. Tìm tên, stars các nhà hàng có số lượng phần tử trong grade bằng 5
10. Hiển thị tên, stars các nhà hàng có ít nhất một phần tử trong grades lớn hơn hoặc bằng 20 và nhỏ hơn 30
11. Hiển thị tên các nhà hàng có số điện thoại bắt đầu bằng 770 và email kết thúc bằng '.com'
12. Hiển thị name, stars và avg_grade (trung bình grades) của mỗi document, làm tròn sau dấu phẩy 2 chữ số thập phân
13. Hiển thị name, stars và min_grade, avg_grade, max_grade, no_of_grades (số lượng phần tử của mảng grades) của mỗi document , sắp xếp theo avg_grade từ cao đến thấp, làm tròn sau dấu phẩy 2 chữ số thập phân
14. Hiển thị name, categories và cat_upper (viết hoa categories) của mỗi document
15. Hiển thị stars như _id và số lượng document tương ứng với mỗi giá trị stars
16. Hiển thị name, stars, no_of_grades ( số lượng phần tử trong grades) và stars_mul_grades (nhân stars với no_of_grades) của mỗi document