MONGODB

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

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

Nội dung:

- Arithmetic Operators
- Array Expression Operators
- Date Expression Operators
- Conditional Expression Operators
- $group

—----------

// 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, 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'}}}}
}]);
```

// Date Expression Operators

```
db.trips.aggregate([
  {$project:
     {_id:0, tripduration: 1,
     journeytime: {$divide: [{$subtract: ['$stop time', '$start time']}, 1000]}}}
]);

// $dateDiff
db.trips.aggregate([
  {$project:
     {_id:0, tripduration: 1,
     journeytime: {$dateDiff: {
```

```
                startDate: '$start time',
                endDate: '$stop time',
                unit: 'second'}
              }}}
]);

// $month, $day, $year
db.trips.aggregate([
  {$project:
    {"start time": 1,
    "month_no": {$month: '$start time'}}}
])

// calculate age of user
db.trips.aggregate([
  {$match: {'birth year': {$ne: ""}}},
  {$project: {'birth year': 1,
    'age': {$subtract: [{$year: new Date()}, '$birth year']}
  }}
])

// $addFields
db.trips.aggregate([
  {$addFields: {'tripduration_hrs': {$divide: ["$tripduration", 60]}}}
])


db.trips.aggregate([
  {$project: {'tripduration_hrs': {$divide: ["$tripduration", 60]}}}
```

```
])

db.trips.aggregate([
  {$project: {
    _id:0, tripduration: 1, bikeid: 1, usertype: 1
  }},
  {$addFields: {
     tripduration_hrs: {$divide: ["$tripduration", 60]}
  }}
])

// $count
db.trips.aggregate([
  {$count: "total docs"}
])

db.trips.aggregate([
  {$match: { $and: [{'tripduration': {$gt: 50}}, {'tripduration': {$lte: 100}}]}},
  {$count: "number of trips between 50 and 100"}
])

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

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}}}}
])

// $switch condition
db.trips.aggregate([
  {$match: {'birth year': {$ne: ""}}},
  {$addFields: {'age': {$subtract: [{$year: new Date()}, '$birth year']}}},
  {$project:
    {   _id:0, 'birth year': 1, age: 1,
      'age class': {$switch: {branches: [
        {case: {$lt: ['$age', 30]}, then: 'young'},
        {case: {$gt: ['$age', 60]}, then: 'Old'}
      ], default: 'Mid'}}
    }}
])

// $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'}
      }
} ])

// $sortByCount: returns the count of each group = {$group: }, {$sort:}
db.trips.aggregate([
  {$sortByCount: '$usertype'}
])

db.trips.aggregate([
  {$group: {_id: '$usertype',
            'count': {$sum: 1}}
  },
  {$sort: {count: -1}}
])

// In the trips collection what is the most common "start station name"?
db.trips.aggregate([
  {$sortByCount: '$start station name'},
  {$limit: 1}
])

// $bucket: {
//   groupBy: expression, // is like _id field in $group
//   boundaries: [lowerbound1, lowerbound2,...], //e.g [0, 100, 200]
//   default: string literal, // if documents do not fit any boundary
//   output: {
//      output1: {accumulator expression}, ...,
```

```
//       outputN: {accumulator expression}
//    }}
db.trips.aggregate([
  {$bucket: {
          groupBy: '$tripduration',
          boundaries: [ 0, 100, 1000, 10000, 1000000 ]
  }}])

db.trips.aggregate([
  {$bucket: {
    groupBy: '$tripduration',
    boundaries: [ 0, 100, 1000, 10000 ],
    default: "other",
    output: {
      "avg_duration": {$avg: '$tripduration'},
      "count": {$sum: 1}
    }}}
])
```

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

Database: restaurant.json

1. Hiển thị name, stars và avg_grade (trung bình grades) của mỗi document, làm tròn sau dấu phẩy 2 chữ số thập phân
2. 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
3. Hiển thị name, categories và cat_upper (viết hoa categories) của mỗi document
4. Hiển thị stars như _id và số lượng document tương ứng với mỗi giá trị stars

5. 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

Database: sales.json

1. Hiển thị ngày bán hàng đầu tiên và ngày bán hàng cuối cùng
2. Hiển thị ngày bán hàng gần nhất có số lượng items bán được nhiều nhất
3. Hiển thị tên sản phẩm và số lượng đã bán của sản phẩm có số lượng bán được nhiều nhất
4. Hiển thị 'storeLocation', số lượng khách hàng ('no_of_customers') theo từng storeLocation và từng 'purchaseMethod', sắp xếp theo storeLocation và purchaseMethod theo bảng chữ cái từ A - Z
5. Thống kê số lượng khách hàng theo độ tuổi như sau: 15-29, 30-44, 45-59, 60-74, 75+
6. Hiển thị số lượng khách hàng (no_of_customers), độ tuổi trung bình (avg_age), mức độ hài lòng trung bình (avg_satisfaction) của khách hàng theo từng khu vực. Làm tròn kết quả: avg_age: làm tròn lên, avg_satisfaction: làm tròn sau dấu phẩy 1 chữ số. Sắp xếp kết quả theo no_of_customers từ cao đến thấp
7. Hiển thị số lượng khách hàng, độ tuổi trung bình, mức độ hài lòng trung bình của khách hàng đã mua hàng ở cửa hàng 'New York' theo từng nhóm giới tính, làm tròn kết quả như câu trên
8. Hiển thị tất cả các distinct tags có trong sales collection
9. Hiển thị 'saleDate', 'items.name', 'items.price', 'items.quantity', và thêm 1 field 'items.revenue' với 'items.revenue' = 'items.price' * 'items.quantity', sort kết quả theo 'saleDate' từ cao đến thấp và chỉ hiển thị 2 kết quả đầu tiên
10. Tính tổng doanh thu (totalSalesAmount) theo từng 'items.name'. Ví dụ: binder có totalSalesAmount: 511644.57
11. Tính tổng doanh thu theo từng năm
12. Tổng số lượng đã bán được và tổng doanh thu của sản phẩm 'laptop' tại cửa hàng New York?