
UNITAU
Universidade de Taubaté

MongoDB Agregação

Agregação

O MongoDB apresenta algumas formas distintas de agregação:

Agregação de Propósito Único - Aplicada uma única operação de agregação na coleção com os métodos **count** e **distinct**.

Agregação em Pipeline – Aplica uma série de estágios de agregação em pipeline para transformar os documentos da coleção.

Map-Reduce – Aplica funções em JavaScript sobre a coleção para executar uma operação de map-reduce. Geralmente, operações de map-reduce tem duas fases, a fase **map** que processa cada documento e gera um ou mais objetos para cada documento e uma fase **reduce** que combina a saída da fase map.

count

O método db.collection.count() permite contar os documentos de uma coleção.

```
db.collection.count(query, options)
```

```
> db.megasena.count()
2071
```

count

Pode ser especificada um filtro para selecionar os documentos a contar.

```
> db.megasena.count( {sena_ganhadores: {$gt: 0}} )
486
```

distinct

O método **db.collection.distinct()** permite obter os valores distintos de um campo.

```
db.collection.distinct(field, query, options)
```

```
> db.megasena.distinct( "sena_ganhadores" )
[ 0, 1, 2, 4, 3, 5, 15, 7, 6, 17 ]
```

distinct

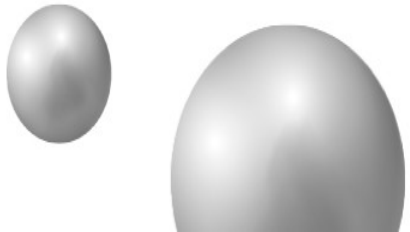
O método **db.collection.distinct()** pode ser utilizado com um campo de documentos embutidos.

```
> db.acervo.distinct( "sede.estado" )
[ "SP", "CA", "RJ" ]
```

distinct

Também pode ser utilizado com arrays, retornando todos os valores distintos dos elementos do array.

```
> db.alunos.distinct( "notas" )
[ 5, 8, 9, 7, 10, 6, 4 ]
```



distinct

Pode ser usado um filtro para selecionar os documentos pesquisados.

```
> db.acervo.distinct( "sede.estado", {"sede.pais": "Brasil"} )
[ "SP", "RJ" ]
```

Agregação em Pipeline

No MongoDB a agregação em pipeline transforma os documentos de uma coleção. Os documentos são submetidos a um pipeline de estágios de agregação.

```
db.collection.aggregate(pipeline, options)
```

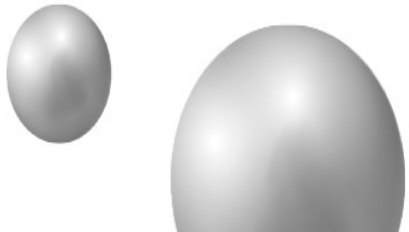
Agregação em Pipeline

```
> db.megasena.aggregate(
    {$match: {sena_ganhadores: {$gt: 0}}},
    {$group: {_id : "$sena_ganhadores",
              sorteios: {$sum: 1},
              ganhadores: { $sum:
"$sena_ganhadores"}}},
    {$sort: {_id: 1}} )
{ "_id":1, "sorteios":358, "ganhadores":358 }
{ "_id":2, "sorteios":87, "ganhadores":174 }
{ "_id":3, "sorteios":23, "ganhadores":69 }
{ "_id":4, "sorteios":11, "ganhadores":44 }
{ "_id":5, "sorteios":2, "ganhadores":10 }
{ "_id":6, "sorteios":2, "ganhadores":12 }
{ "_id":7, "sorteios":1, "ganhadores":7 }
{ "_id":15, "sorteios":1, "ganhadores":15 }
{ "_id":17, "sorteios":1, "ganhadores":17 }
```

Estágios de Agregação: $count

$count retorna o número de documentos na entrada do estágio.

```
{ $count: <string> }
> db.megasena.aggregate(
    {$count: "sorteios"}
  )
{ "sorteios" : 2071 }
```



Estágios de Agregação: $match

$match filtra os documentos que correspondem ao critério de seleção.

```
{ $match: { <query> } }
```

```
> db.megasena.aggregate(
    {$match: {sena_ganhadores: {$gt: 0}}},
    {$count: "sorteios"}
  )
{ "sorteios" : 486 }
```

Estágios de Agregação: $sort

$sort ordena os documentos da entrada do estágio.

```
{ $sort: { field1: value1, field2:
value2, ... } }
```

```
> db.livro.aggregate(
    {$match: {editora_id: "oreilly"}},
    {$sort: {ano: -1}}
  )
{ "_id" : 3, "titulo" : "Programming With
QT", "editora_id" : "oreilly", "isbn" :
"9781449390938", "ano" : "2010", "edicao" :
"2" }
{ "_id" : 4, "titulo" : "SQL Tuning",
"editora_id" : "oreilly", "isbn" :
"9780596552367", "ano" : "2009", "edicao" :
"1" }
```

Estágios de Agregação: $limit

$limit determina o número máximo de documentos a serem passados para o próximo estágio.

```
{ $limit: <number> }
```

```
> db.livro.aggregate(
    {$sort: {ano: -1}},
    {$limit: 2}
  )
{ "_id" : 1, "titulo" : "Introducao ao
MongoDB", "editora_id" : "novatec", "isbn" :
"8575224220", "ano" : "2015", "edicao" :
"1" }
{ "_id" : 2, "titulo" : "Arduino Basico",
"editora_id" : "novatec", "isbn" :
"8575224042", "ano" : "2015", "edicao" :
"2" }
```

Estágios de Agregação: $skip

$skip determina o número de documentos da entrada do estágio que devem ser ignorados.

```
{ $skip: <offset> }
```

```
> db.livro.aggregate(
    {$sort: {ano: -1}},
    {$skip: 3}
  )
{ "_id" : 5, "titulo" : "Programacao em
Linguagem C", "editora_id" : "cm", "isbn" :
"8573939494", "ano" : "2010", "edicao" :
"1" }
{ "_id" : 4, "titulo" : "SQL Tuning",
"editora_id" : "oreilly", "isbn" :
"9780596552367", "ano" : "2009", "edicao" :
"1" }
```

Estágios de Agregação: $sample

$sample determina um número de documentos aleatórios a serem passados para o próximo estágio.

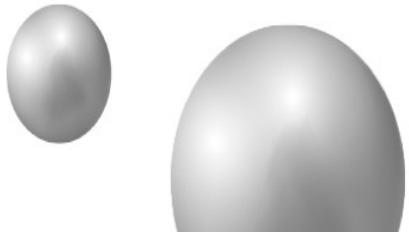
```
{ $sample: { size: <number> } }
```

```
> db.livro.aggregate(
    {$sort: {ano: -1}},
    {$sample: {size: 2}}
  )
{ "_id" : 5, "titulo" : "Programacao em
Linguagem C", "editora_id" : "cm", "isbn" :
"8573939494", "ano" : "2010", "edicao" :
"1" }
{ "_id" : 4, "titulo" : "SQL Tuning",
"editora_id" : "oreilly", "isbn" :
"9780596552367", "ano" : "2009", "edicao" :
"1" }
```

Estágios de Agregação: $group

$group agrupa os documentos segundo a expressão de agrupamento do campo **_id**, gerando um documento para cada valor distinto dessa expressão. Os operadores chamados acumuladores são utilizados nesse estágio para calcular o valor de expressões para os grupos de documentos. Alguns acumuladores também podem ser utilizados nos estágios **$project** e **$addFields**.

```
{ $group: { _id: <expression>, <field1>:
{ <accumulator1> : <expression1> }, ... } }
```



Estágios de Agregação: $group

```
> db.megasena.aggregate(
    {$match: {sena_ganhadores: {$gte: 5}}},
    {$group: {_id : "$sena_ganhadores",
              sorteios: {$sum: 1},
              ganhadores: {$sum:
"$sena_ganhadores"}}} )
{ "_id" : 17, "sorteios" : 1, "ganhadores" :
17 }
{ "_id" : 5, "sorteios" : 2, "ganhadores" :
10 }
{ "_id" : 7, "sorteios" : 1, "ganhadores" : 7
}
{ "_id" : 15, "sorteios" : 1, "ganhadores" :
15 }
{ "_id" : 6, "sorteios" : 2, "ganhadores" :
12 }
```

Estágios de Agregação: $group

Se o valor do _id for null ou outro valor constante, a expressões de acumulação será calculada para todos os documentos da entrada do estágio.

```
> db.produtos.aggregate(
    {$group: {_id: null,
              soma: {$sum: "$quantidade"}}}
  )
{ "_id" : null, "soma" : 160 }

> db.produtos.aggregate(
    {$group: {_id: "a",
              soma: {$sum: "$quantidade"}}}
  )
{ "_id" : "a", "soma" : 160 }
```

Acumuladores: $sum

$sum calcula a soma de valores numéricos. **$sum** ignora valores não numéricos. Quando usado no estágio **$group**, **$sum** trata arrays como valores não numéricos.

```
{ $sum: <expression> }
```

Acumuladores: $sum

```
> db.megasena.aggregate(
    {$match: {sena_ganhadores: {$gte: 5}}},
    {$group: {_id : "$sena_ganhadores",
              sorteios: {$sum: 1},
              ganhadores: {$sum:
"$sena_ganhadores"}}} )
{ "_id" : 17, "sorteios" : 1, "ganhadores" :
17 }
{ "_id" : 5, "sorteios" : 2, "ganhadores" :
10 }
{ "_id" : 7, "sorteios" : 1, "ganhadores" : 7
}
{ "_id" : 15, "sorteios" : 1, "ganhadores" :
15 }
{ "_id" : 6, "sorteios" : 2, "ganhadores" :
12 }
```

Acumuladores: $avg

$avg calcula a média de valores numéricos. **$avg** ignora valores não numéricos. Quando usado no estágio **$group**, **$avg** trata arrays como valores não numéricos.

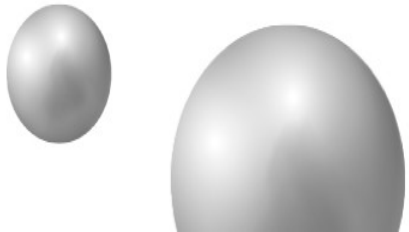
```
{ $avg: <expression> }
```

```
> db.produtos.aggregate(
    {$group: {_id: null,
              media: {$avg: "$quantidade"}}}
  )
{ "_id" : null, "media" : 26.666666666666668}
```

Acumuladores: $first

$first retorna o valor da expressão para o primeiro documento de cada grupo. Os documentos devem estar ordenados para dar significado ao resultado.

```
{ $first: <expression> }
```



Acumuladores: $first

```
> db.megasena.aggregate(
    {$match: {sena_ganhadores: {$gt: 0}}},
    {$sort: {_id: 1}},
    {$group: {_id : "$sena_ganhadores",
              primeiro: {$first:
"$sena_rateio"}}}
  )
{ "_id" : 6, "primeiro" : 41088919.05 }
{ "_id" : 15, "primeiro" : 348732.75 }
{ "_id" : 5, "primeiro" : 3196547.03 }
{ "_id" : 7, "primeiro" : 13217564.89 }
{ "_id" : 1, "primeiro" : 2307162.23 }
{ "_id" : 2, "primeiro" : 391192.51 }
{ "_id" : 3, "primeiro" : 7089194.54 }
{ "_id" : 17, "primeiro" : 18042279.04 }
{ "_id" : 4, "primeiro" : 801057.55 }
```

Acumuladores: $last

$last retorna o valor da expressão para o último documento de cada grupo. Os documentos devem estar ordenados para dar significado ao resultado.

```
{ $last: <expression> }
```

Acumuladores: $last

```
> db.megasena.aggregate(
    {$match: {sena_ganhadores: {$gt: 0}}},
    {$sort: {_id: 1}},
    {$group: {_id : "$sena_ganhadores",
              ultimo: {$last:
"$sena_rateio"}}}
  )
{ "_id" : 6, "ultimo" : 36824758.22 }
{ "_id" : 15, "ultimo" : 348732.75 }
{ "_id" : 5, "ultimo" : 35523497.52 }
{ "_id" : 7, "ultimo" : 13217564.89 }
{ "_id" : 1, "ultimo" : 35155103.28 }
{ "_id" : 2, "ultimo" : 29864355.95 }
{ "_id" : 3, "ultimo" : 4095573.45 }
{ "_id" : 17, "ultimo" : 18042279.04 }
{ "_id" : 4, "ultimo" : 9627559.21 }
```

Acumuladores: $max

$max retorna o maior valor da expressão para os documento de cada grupo.

```
{ $max: <expression> }
```

Acumuladores: $max

```
> db.megasena.aggregate(
    {$match: {sena_ganhadores: {$gt: 0}}},
    {$sort: {_id: 1}},
    {$group: {_id : "$sena_ganhadores",
              maior: {$max: "$sena_rateio"}}}
  )
{ "_id" : 6, "maior" : 41088919.05 }
{ "_id" : 15, "maior" : 348732.75 }
{ "_id" : 5, "maior" : 35523497.52 }
{ "_id" : 7, "maior" : 13217564.89 }
{ "_id" : 1, "maior" : 205329753.89 }
{ "_id" : 2, "maior" : 98688974.76 }
{ "_id" : 3, "maior" : 81594699.72 }
{ "_id" : 17, "maior" : 18042279.04 }
{ "_id" : 4, "maior" : 65823888.16 }
```

Acumuladores: $min

$min retorna o menor valor da expressão para os documento de cada grupo.

```
{ $min: <expression> }
```

Acumuladores: $min

```
> db.megasena.aggregate(
    {$match: {sena_ganhadores: {$gt: 0}}},
    {$sort: {_id: 1}},
    {$group: {_id : "$sena_ganhadores",
              menor: {$min: "$sena_rateio"}}}
  )
{ "_id" : 6, "menor" : 36824758.22 }
{ "_id" : 15, "menor" : 348732.75 }
{ "_id" : 5, "menor" : 3196547.03 }
{ "_id" : 7, "menor" : 13217564.89 }
{ "_id" : 1, "menor" : 367173.22 }
{ "_id" : 2, "menor" : 391192.51 }
{ "_id" : 3, "menor" : 361199.61 }
{ "_id" : 17, "menor" : 18042279.04 }
{ "_id" : 4, "menor" : 773932.3 }
```

Acumuladores: $stdDevSamp

$stdDevSamp retorna o desvio padrão para amostragem para o valor da expressão dos documento de cada grupo. Quando usado no estágio **$group**, **$stdDevSamp** trata arrays como valores não numéricos.

```
{ $stdDevSamp: <expression> }
```

```
> db.megasena.aggregate(
    {$group: {_id : null,
                desvio: {$stdDevSamp:
"$sena_rateio"}}}
  )
{ "_id" : null, "desvio" : 127542l5.288648907
}
```

Acumuladores: $stdDevPop

$stdDevPop retorna o desvio padrão para população para o valor da expressão dos documento de cada grupo. Quando usado no estágio **$group**, **$stdDevPop** trata arrays como valores não numéricos.

```
{ $stdDevPop: <expression> }
```

```
> db.megasena.aggregate(
    {$group: {_id : null,
              desvio: {$stdDevPop:
"$sena_rateio"}}}
  )
{ "_id" : null, "desvio" : 12751135.676075064
}
```

Acumuladores: $addToSet

$addToSet retorna um array com todos os valores distintos da expressão para os documento de cada grupo.

```
{ $addToSet: <expression> }
```

```
> db.produtos.aggregate(
    {$group: {_id: null,
              teste: {$addToSet:
"$quantidade"}}}
  )
{ "_id" : null, "teste" : [ 25, 15, 50, 30,
20 ] }
```

Acumuladores: $push

$push retorna um array com todos os valores da expressão para os documento de cada grupo.

```
{ $push: <expression> }
```

```
> db.produtos.aggregate(
    {$group: {_id: null,
              teste: {$push: "$quantidade"}}}
  )
{ "_id" : null, "teste" : [ 20, 15, 20, 30,
50, 25 ] }
```

Acumuladores: $push

$push pode ser usado para pivoteamento dos dados

```
> db.livro.aggregate(
      {$group: {_id: "$editora_id",
                livros: {$push: "$titulo"}}}
   )
{ "_id" : "oreilly", "livros" :
[ "Programming With QT", "SQL Tuning" ] }
{ "_id" : "cm", "livros" : [ "Programacao em
Linguagem C" ] }
{ "_id" : "novatec", "livros" : [ "Introducao
ao MongoDB", "Arduino Basico" ] }
```

Acumuladores: $push

A variável **ROOT** referencia o documento superior sendo processado no estágio da agregação.

```
> db.livro.aggregate(
    {$match: {editora_id: "oreilly"}},
    {$group : {_id: "$editora_id",
               livros: {$push: "$$ROOT"}}}
  ).pretty()
```

Acumuladores: $push

```
{"_id" : "oreilly",
 "livros" : [{"_id" : 3,
              "titulo" : "Programming With
QT",
              "editora_id" : "oreilly",
              "isbn" : "9781449390938",
              "ano" : "2010",
              "edicao" : "2"},
             {"_id" : 4,
              "titulo" : "SQL Tuning",
              "editora_id" : "oreilly",
              "isbn" : "9780596552367",
              "ano" : "2009",
              "edicao" : "1"}
            ]
}
```

Estágios de Agregação: $unwind

$unwind decompõe um array, gerando um documento para cada elemento do array com o valor do elemento substituindo o array.

```
{ $unwind: <field path> }
```

Ou

```
{
  $unwind:
    {
      path: <field path>,
      includeArrayIndex: <string>,
      preserveNullAndEmptyArrays: <boolean>
    }
}
```

Estágios de Agregação: $unwind

```
> db.alunos.aggregate(
    {$match: {nome: "Marcia"}},
    {$unwind: "$notas"}
  )
{ "_id" : 4, "nome" : "Marcia", "notas" : 7 }
{ "_id" : 4, "nome" : "Marcia", "notas" : 6 }
{ "_id" : 4, "nome" : "Marcia", "notas" : 9 }
```

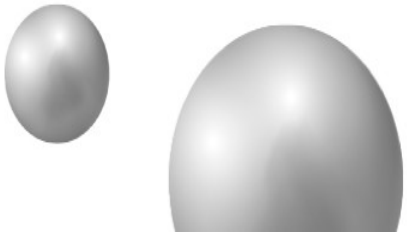
Estágios de Agregação: $sortByCount

$sortByCount agrupa os documentos pela expressão indicada, conta os documentos em cada grupo e ordena o resultado pelo valor da contagem em ordem decrescente.

```
{ $sortByCount:  <expression> }
```

É equivalente a:

```
{ $group: { _id: <expression>, count: { $sum:
1 } } },
{ $sort: { count: -1 } }
```

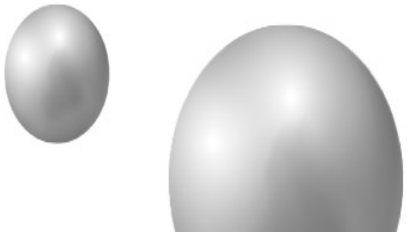


Estágios de Agregação: $sortByCount

```
> db.produtos.aggregate( {$sortByCount:
"$quantidade"} )
{ "_id" : 20, "count" : 2 }
{ "_id" : 25, "count" : 1 }
{ "_id" : 15, "count" : 1 }
{ "_id" : 30, "count" : 1 }
{ "_id" : 50, "count" : 1 }
```

Estágios de Agregação: $bucket

$bucket categoriza os documentos da entrada do estágio em grupos especificados. A cláusula **boundaries** especifica o array dos limites dos grupos. Cada par de valores adjacentes especifica o limite inferior (incluso no grupo) e o limite superior (excluído do grupo). É obrigatório especificar ao menos dois limites. O campo **_id** de cada documento indica o limite inferior do grupo. Se não for especificada a cláusula **output**, será incluído automaticamente o campo **count** com o número de documentos do grupo. A cláusula **default** especifica o **_id** para o grupo dos documentos que não se enquadrarem nos limites especificados na cláusula **boundaries**.



Estágios de Agregação: $bucket

```
{
  $bucket: {
      groupBy: <expression>,
      boundaries: [ <lowerbound1>,
<lowerbound2>, ... ],
      default: <literal>,
      output: {
         <output1>: { <$accumulator
expression> },
         ...
         <outputN>: { <$accumulator
expression> }
      }
   }
}
```

Estágios de Agregação: $bucket

```
> db.produtos.aggregate(
    {$bucket: {groupBy: "$quantidade",
               boundaries: [20, 25, 50],
               default: "Outros",
               output: {"count": {$sum: 1},
                        "titles" : {$push:
"$item"}}}}
   )
{ "_id" : 20, "count" : 2, "titles" :
[ "caderno", "envelope" ] }
{ "_id" : 25, "count" : 2, "titles" :
[ "selos", "borracha" ] }
{ "_id" : "Outros", "count" : 2, "titles" : [
"cartao", "lapis" ] }
```

Estágios de Agregação: $bucketAuto

$bucketAuto categoriza os documentos da entrada do estágio no número de grupos especificados. Os limites de cada grupo são calculados automaticamente para obter o número de grupos especificado. O campo **_id** de cada documento indica o limite inferior (incluso no grupo) e o limite superior (excluído do grupo). Se não for especificada a cláusula **output**, será incluído automaticamente o campo **count** com o número de documentos do grupo.

Estágios de Agregação: $bucketAuto

```
{
  $bucketAuto: {
      groupBy: <expression>,
      buckets: <number>,
      output: {
         <output1>: { <$accumulator
expression> },
         ...
      }
      granularity: <string>
  }
}
```

Estágios de Agregação: $bucketAuto

```
> db.megasena.aggregate(
    {$match: {sena_ganhadores: {$gt: 0}}},
    {$group: {_id : "$sena_ganhadores",
              sorteios: {$sum: 1},
              ganhadores:
{$sum:"$sena_ganhadores"}}},
    {$bucketAuto: {groupBy: "$_id",
                   buckets: 4,
                   output: {sorteios:
{$sum:"$sorteios"},
                   ganhadores: {$sum:
"$ganhadores"}}}}
    )
```

Estágios de Agregação: $bucketAuto

```
{ "_id" : { "min" : 1, "max" : 3 },
"sorteios" : 445, "ganhadores" : 532 }
{ "_id" : { "min" : 3, "max" : 5 },
"sorteios" : 34, "ganhadores" : 113 }
{ "_id" : { "min" : 5, "max" : 7 },
"sorteios" : 4, "ganhadores" : 22 }
{ "_id" : { "min" : 7, "max" : 17 },
"sorteios" : 3, "ganhadores" : 39 }
```

Exercícios

1) Obter o número de clientes que compraram pelo menos 3 cadernos

```
3
```

2) Obter um array com as notas de Português

```
[ 89, 90, 77, 85 ]
```

3) Obter um array com o nome dos produtos vendidos no dia 18/02/2020

```
[ "Borracha", "Lapis", "Marca Texto",
"Caderno", "Caneta", "Post-it" ]
```

4) Obter um array com o nome das clientes do sexo feminino

```
[ "Laura", "Renata", "Marta" ]
```

Exercícios

5) Relacionar o id da venda e a quantidade total de unidades de cada venda, ordenado pelo id da venda

```
{ "_id" : "001", "total" : 5 }
{ "_id" : "002", "total" : 8 }
{ "_id" : "003", "total" : 7 }
{ "_id" : "004", "total" : 6 }
{ "_id" : "005", "total" : 14 }
{ "_id" : "006", "total" : 24 }
```

6) Relacionar o nome do produto e a quantidade total de unidades vendidas dos 3 produtos com a maior quantidade de unidades vendidas

```
{ "_id" : "Caderno", "total" : 13 }
{ "_id" : "Caneta", "total" : 12 }
{ "_id" : "Borracha", "total" : 8 }
```

Exercícios

7) Relacionar o nome do produto e a quantidade total de unidades compradas pela cliente Laura, ordenado pela quantidade de unidades compradas

```
{ "_id" : "Caderno", "total" : 5 }
{ "_id" : "Post-it", "total" : 4 }
{ "_id" : "Caneta", "total" : 4 }
{ "_id" : "Lapis", "total" : 3 }
{ "_id" : "Fichario", "total" : 2 }
{ "_id" : "Corretivo", "total" : 1 }
```

8) Relacionar o nome do produto e a quantidade total de unidades comprados pela cliente Laura, para produtos que tenham pelo menos 4 unidades compradas

```
{ "_id" : "Post-it", "total" : 4 }
{ "_id" : "Caneta", "total" : 4 }
{ "_id" : "Caderno", "total" : 5 }
```

Exercícios

9) Relacionar o nome do produto e a quantidade de unidades vendidas na venda com maior quantidade de unidades vendidas do produto, ordenado pelo nome do produto

```
{ "_id" : "Borracha", "maior" : 5 }
{ "_id" : "Caderno", "maior" : 4 }
{ "_id" : "Caneta", "maior" : 6 }
{ "_id" : "Corretivo", "maior" : 2 }
{ "_id" : "Fichario", "maior" : 3 }
{ "_id" : "Lapis", "maior" : 4 }
{ "_id" : "Lapiseira", "maior" : 4 }
{ "_id" : "Marca Texto", "maior" : 2 }
{ "_id" : "Post-it", "maior" : 4 }
```

Exercícios

10) Relacionar o nome do cliente e um array com o nome dos produtos comprados pelo cliente, ordenado pelo nome do cliente

```
{ "_id" : "Henrique", "produtos" : [ "Marca Texto", "Borracha", "Lapis" ] }
{ "_id" : "Laura", "produtos" : [ "Post-it", "Corretivo", "Fichario", "Caneta", "Lapis", "Caderno" ] }
{ "_id" : "Marta", "produtos" : [ "Borracha", "Lapiseira", "Caderno" ] }
{ "_id" : "Paulo", "produtos" : [ "Post-it", "Caneta", "Caderno" ] }
{ "_id" : "Renata", "produtos" :
[ "Corretivo", "Lapiseira", "Borracha", "Caneta", "Caderno", "Fichario" ] }
```

Exercícios

11) Relacionar o nome do produto e um array com o nome dos cliente que compraram o produto, ordenado pelo nome do produto

```
{ "_id" : "Borracha", "clientes" : [ "Marta",
"Renata", "Henrique" ] }
{ "_id" : "Caderno", "clientes" : [ "Marta",
"Renata", "Paulo", "Laura" ] }
{ "_id" : "Caneta", "clientes" : [ "Renata",
"Paulo", "Laura" ] }
{ "_id" : "Corretivo", "clientes" :
[ "Renata", "Laura" ] }
{ "_id" : "Fichario", "clientes" :
[ "Renata", "Laura" ] }
{ "_id" : "Lapis", "clientes" : [ "Henrique",
"Laura" ] }
```

Exercícios

```
{ "_id" : "Lapiseira", "clientes" :
[ "Marta", "Renata" ] }
{ "_id" : "Marca Texto", "clientes" :
[ "Henrique" ] }
{ "_id" : "Post-it", "clientes" : [ "Paulo",
"Laura" ] }
```

Exercícios

12) Relacionar a data e o número de vendas na data, ordenado pela data

```
{ "_id" : "2020-02-14", "vendas" : 1 }
{ "_id" : "2020-02-18", "vendas" : 2 }
{ "_id" : "2020-02-22", "vendas" : 1 }
{ "_id" : "2020-03-02", "vendas" : 1 }
{ "_id" : "2020-03-06", "vendas" : 1 }
```

13) Relacionar a data e o número de vendas na data, ordenado pelo número de vendas

```
{ "_id" : "2020-02-18", "count" : 2 }
{ "_id" : "2020-02-22", "count" : 1 }
{ "_id" : "2020-03-06", "count" : 1 }
{ "_id" : "2020-03-02", "count" : 1 }
{ "_id" : "2020-02-14", "count" : 1 }
```

Exercícios

14) Relacionar o nome do produto e o número de vendas onde o produto foi vendido

```
{ "_id" : "Lapiseira", "vendas" : 2 }
{ "_id" : "Post-it", "vendas" : 2 }
{ "_id" : "Corretivo", "vendas" : 2 }
{ "_id" : "Fichario", "vendas" : 2 }
{ "_id" : "Caneta", "vendas" : 3 }
{ "_id" : "Marca Texto", "vendas" : 1 }
{ "_id" : "Borracha", "vendas" : 3 }
{ "_id" : "Lapis", "vendas" : 2 }
{ "_id" : "Caderno", "vendas" : 5 }
```

Exercícios

15) Relacionar o nome do produto e o número de vendas onde o produto foi vendido para os 3 produtos com o maior número de vendas

```
{ "_id" : "Caderno", "count" : 5 }
{ "_id" : "Borracha", "count" : 3 }
{ "_id" : "Caneta", "count" : 3 }
```

Exercícios

16) Separar os produtos em produtos que venderam ate 5 unidades, ate 10 unidades e mais de 10 unidades, com o total de unidades vendidas em cada grupo

```
{ "_id" : 0, "produtos" : [ "Corretivo",
"Fichario", "Marca Texto" ], "total" : 10 }
{ "_id" : 6, "produtos" : [ "Lapiseira",
"Post-it", "Borracha", "Lapis" ], "total" :
29 }
{ "_id" : "Acima de 10", "produtos" :
[ "Caneta", "Caderno" ], "total" : 25 }
```

Exercícios

17) Separar os produtos em melhores, intermediários e piores pelo número de unidades vendidas, com o número de unidades vendidas de cada produto

```
{ "_id" : { "min" : 2, "max" : 7 },
"produtos" : [ "Marca Texto", "Corretivo",
"Fichario" ], "quantidades" : [ 2, 3, 5 ] }
{ "_id" : { "min" : 7, "max" : 8 },
"produtos" : [ "Lapiseira", "Post-it",
"Lapis" ], "quantidades" : [ 7, 7, 7 ] }
{ "_id" : { "min" : 8, "max" : 13 },
"produtos" : [ "Borracha", "Caneta",
"Caderno" ], "quantidades" : [ 8, 12, 13 ] }
```