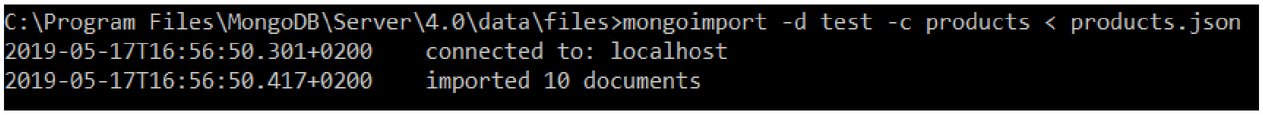
# Aggregation Framework

```
> mongoimport -d test -c products < products.json
connected to 127.0.0.1
2014-04-27T08:03:06.506+0200 imported 10 objects
```

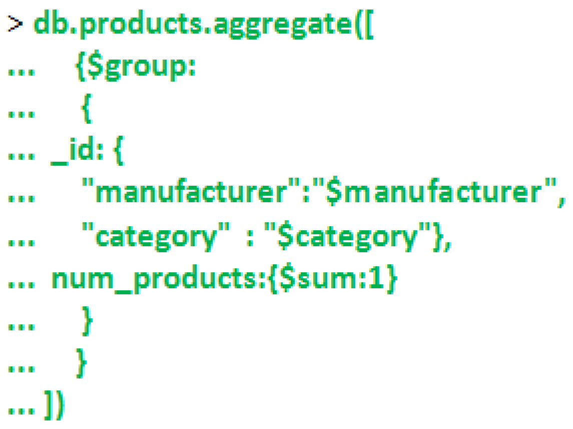


```
C:\Program Files\MongoDB\Server\4.0\data\files>mongoimport -d test -c products < products.json
2019-05-17T16:56:50.301+0200    connected to: localhost
2019-05-17T16:56:50.417+0200    imported 10 documents
```

## Aggregation Pipeline Stages

### Group

Nombre de products de chaque manufacturer pour chaque catégorie.



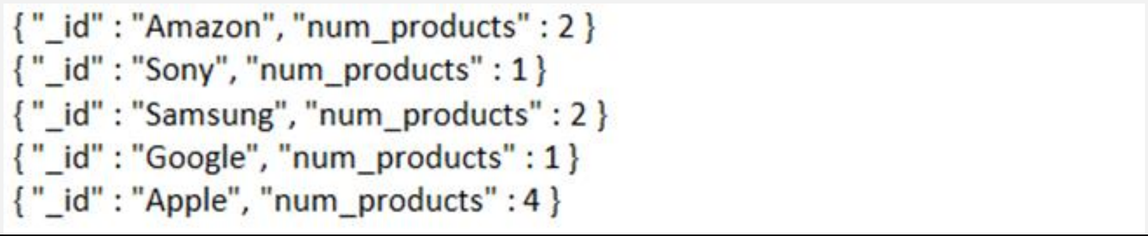
```
> db.products.aggregate([
...    {$group:
...     {
... _id: {
...     "manufacturer":"$manufacturer",
...     "category" : "$category"},
... num_products:{$sum:1}
...     }
...    }
... ])
```

```
{ "_id" : { "manufacturer" : "Amazon", "category" : "Tablets" }, "num_products" : 2 }
{ "_id" : { "manufacturer" : "Apple", "category" : "Laptops" }, "num_products" : 1 }
{ "_id" : { "manufacturer" : "Google", "category" : "Tablets" }, "num_products" : 1 }
{ "_id" : { "manufacturer" : "Sony", "category" : "Laptops" }, "num_products" : 1 }
{ "_id" : { "manufacturer" : "Samsung", "category" : "Tablets" }, "num_products" : 1 }
{ "_id" : { "manufacturer" : "Samsung", "category" : "Cell Phones" }, "num_products" : 1 }
{ "_id" : { "manufacturer" : "Apple", "category" : "Tablets" }, "num_products" : 3 }
```

## Group

Trouver le nombre de produit par manufacturer.

```
> db.products.aggregate([  {$group:  {  _id:"$manufacturer",  num_products:{$sum:1}   }  } ])
```



```
{ "_id" : "Amazon", "num_products" : 2 }
{ "_id" : "Sony", "num_products" : 1 }
{ "_id" : "Samsung", "num_products" : 2 }
{ "_id" : "Google", "num_products" : 1 }
{ "_id" : "Apple", "num_products" : 4 }
```

## Group

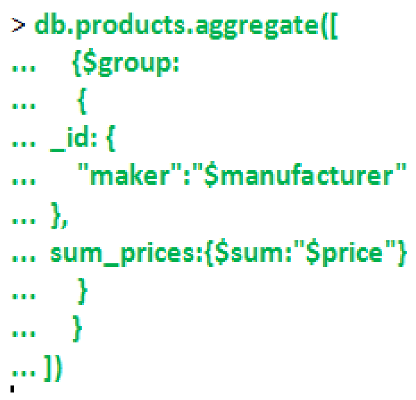
Nombre de products de chaque manufacturer pour chaque catégorie.

```
> db.products.aggregate([
...     {$group:
...      {
... _id: {
...      "manufacturer":"$manufacturer",
...      "category"  : "$category"},
...  num_products:{$sum:1}
...      }
...     }
... ])
```

```
{ "_id" : { "manufacturer" : "Amazon", "category" : "Tablets" }, "num_products" : 2 }
{ "_id" : { "manufacturer" : "Apple", "category" : "Laptops" }, "num_products" : 1 }
{ "_id" : { "manufacturer" : "Google", "category" : "Tablets" }, "num_products" : 1 }
{ "_id" : { "manufacturer" : "Sony", "category" : "Laptops" }, "num_products" : 1 }
{ "_id" : { "manufacturer" : "Samsung", "category" : "Tablets" }, "num_products" : 1 }
{ "_id" : { "manufacturer" : "Samsung", "category" : "Cell Phones" }, "num_products" : 1 }
{ "_id" : { "manufacturer" : "Apple", "category" : "Tablets" }, "num_products" : 3 }
```

## Group

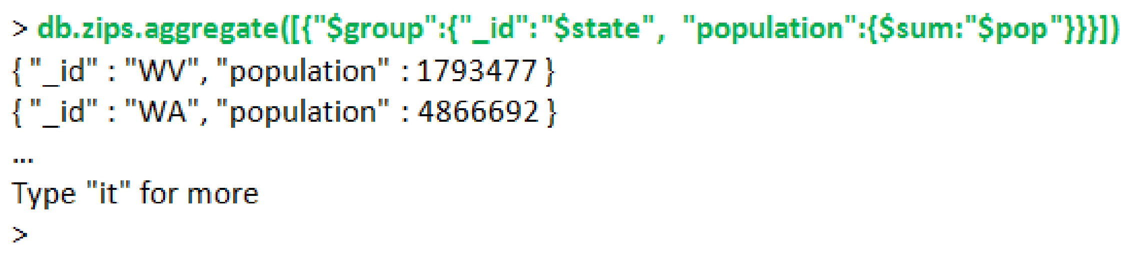
Sum: La somme des prix pour chaque manufacturer.



```
> db.products.aggregate([
...    {$group:
...     {
... _id: {
...     "maker":"$manufacturer"
... },
... sum_prices:{$sum:"$price"}
...     }
...    }
... ])
```

```
{ "_id" : { "maker" : "Amazon" }, "sum_prices" : 328 }
{ "_id" : { "maker" : "Sony" }, "sum_prices" : 499 }
{ "_id" : { "maker" : "Samsung" }, "sum_prices" : 1014.98 }
{ "_id" : { "maker" : "Google" }, "sum_prices" : 199 }
{ "_id" : { "maker" : "Apple" }, "sum_prices" : 2296 }
```

Ex: Calculer la somme de la population (pop) par state et afficher le résultat dans un champ nommé : population.



```
> db.zips.aggregate([{"$group":{"_id":"$state",  "population":{$sum:"$pop"}}}])
{ "_id" : "WV", "population" : 1793477 }
{ "_id" : "WA", "population" : 4866692 }
...
Type "it" for more
>
```

## Group

Avg: Trouver le prix moyen par category.

```
> db.products.aggregate([
...    {$group:
...     {
... _id: {
...     "category":"$category"
... },
... avg_price:{$avg:"$price"}
...     }
...    }
... ])
```

```
{ "_id" : { "category" : "Laptops" }, "avg_price" : 499 }
{ "_id" : { "category" : "Cell Phones" }, "avg_price" : 563.99 }
{ "_id" : { "category" : "Tablets" }, "avg_price" : 396.4271428571428 }
```

## Group

Lister les « category » par « manufacturer ». Cette liste contient des éléments uniques.

```
> db.products.aggregate([
...    {$group:
...     {
... _id: {
...     "maker":"$manufacturer"
... },
... categories:{$addToSet:"$category"}
...     }
...    }
... ])
```

```
{ "_id" : { "maker" : "Amazon" }, "categories" : [ "Tablets" ] }
{ "_id" : { "maker" : "Sony" }, "categories" : [ "Laptops" ] }
{ "_id" : { "maker" : "Samsung" }, "categories" : [ "Tablets", "Cell Phones" ] }
{ "_id" : { "maker" : "Google" }, "categories" : [ "Tablets" ] }
{ "_id" : { "maker" : "Apple" }, "categories" : [ "Laptops", "Tablets" ] }
```

## Group

Push : Lister les « category » par « manufacturer »

```
> db.products.aggregate([
...     {$group:
...      {
... _id: {
...      "maker":"$manufacturer"
... },
... categories:{$push:"$category"}
...      }
...     }
... ])
```

```
{ "_id" : { "maker" : "Amazon" }, "categories" : [ "Tablets", "Tablets" ] }
{ "_id" : { "maker" : "Sony" }, "categories" : [ "Laptops" ] }
{ "_id" : { "maker" : "Samsung" }, "categories" : [ "Cell Phones", "Tablets" ] }
{ "_id" : { "maker" : "Google" }, "categories" : [ "Tablets" ] }
{ "_id" : { "maker" : "Apple" }, "categories" : [ "Tablets", "Tablets", "Tablets", "Laptops" ] }
```

## Group

$max et $min : Déterminer le prix « price » maximum de chaque « manufacturer ».

```
> db.products.aggregate([
...    {$group:
...     {
... _id: {
...     "maker":"$manufacturer"
... },
... maxprice:{$max:"$price"}
...     }
...    }
... ])
```

```
{ "_id" : { "maker" : "Amazon" }, "maxprice" : 199 }
{ "_id" : { "maker" : "Sony" }, "maxprice" : 499 }
{ "_id" : { "maker" : "Samsung" }, "maxprice" : 563.99 }
{ "_id" : { "maker" : "Google" }, "maxprice" : 199 }
{ "_id" : { "maker" : "Apple" }, "maxprice" : 699 }
```

## Project

$multiply

```
> db.products.findOne()
{
"_id" : ObjectId("535c9d9a3a9816733480ee86"),
"name" : "iPad 16GB Wifi",
"manufacturer" : "Apple",
"category" : "Tablets",
"price" : 499
}
```

```
db.products.aggregate([
    {$project:
     {
        _id:0,
        'maker': {$toLower:"$manufacturer"},
        'details': {'category': "$category",
                    'price' : {"$multiply":["$price",10]}
                   },
        'item':'$name'
     }
    }
])
```

```
{ "maker" : "apple", "details" : { "category" : "Tablets", "price" : 4990 }, "item" : "iPad 16GB Wifi" }
{ "maker" : "apple", "details" : { "category" : "Tablets", "price" : 5990 }, "item" : "iPad 32GB Wifi" }
  ...
```

## Importer le fichier : zips.json

## Match

Rechercher tous les documents avec « state » égal à « NY »

```
use test

db.zips.aggregate([
    {$match:
     {
        state:"NY"
     }
    }
])
```

```
{ "_id" : "06390", "city" : "FISHERS ISLAND", "loc" : [ -72.017834, 41.263934 ], "pop" : 329, "state" : "NY" }
{ "_id" : "10001", "city" : "NEW YORK", "loc" : [ -73.996705, 40.74838 ], "pop" : 18913, "state" : "NY" }
{ "_id" : "10002", "city" : "NEW YORK", "loc" : [ -73.987681, 40.715231 ], "pop" : 84143, "state" :"NY" }
  ...
```

## Match

Rechercher la population totale par « city » de l’état « state » de « NY »

```
db.zips.aggregate([
   {$match:  {  state:"NY"   } },
   {$group:  { _id: "$city",  population:  {$sum:"$pop"} }  }
])
```

```
{ "_id" : "ELMIRA HEIGHTS", "population" : 7918 }
{ "_id" : "WELLSVILLE", "population" : 9645 }
{ "_id" : "WATKINS GLEN", "population" : 4584 }
{ "_id" : "VAN ETTEN", "population" : 1477 }
  …
```

## Match

Sort : Tri

```
db.zips.aggregate([
   {$match:  {  state:"NY"   } },
   {$group:  { _id: "$city",  population:  {$sum:"$pop"} }  },
   {$sort:  { population:-1  } }
])
```

```
{ "_id" : "BROOKLYN", "population" : 2300504 }
{ "_id" : "NEW YORK", "population" : 1476790 }
{ "_id" : "BRONX", "population" : 1209548 }
{ "_id" : "ROCHESTER", "population" : 396013 }
…
```

## Match

limit et skip : A utiliser avec sort sinon résultat non définie

```
db.zips.aggregate([
   {$match:  {  state:"NY"   } },
   {$group:  { _id: "$city",  population:  {$sum:"$pop"} }  },
   {$sort:  { population:-1  } },
   {$skip: 10},
   {$limit: 5}
])
```

```
{ "_id" : "ASTORIA", "population" : 165629 }
{ "_id" : "JACKSON HEIGHTS", "population" : 145967 }
{ "_id" : "FAR ROCKAWAY", "population" : 100646 }
{ "_id" : "RIDGEWOOD", "population" : 85732 }
{ "_id" : "BINGHAMTON", "population" : 83017 }
```

## Match

first et last : Find the largest city in every state. **Phase 1**

```
db.zips.aggregate([
    /* Trouver la population de chaque ville de chaque */
    {$group:
     {
        _id: {state:"$state", city:"$city"},
        population: {$sum:"$pop"},
     }
    }
])
```

```
{ "_id" : { "state" : "WY", "city" : "THAYNE" }, "population" : 505 }
{ "_id" : { "state" : "WY", "city" : "SMOOT" }, "population" : 414 }
....
```

## Phase 2

```
db.zips.aggregate([
    /* Trouver la population de chaque ville de chaque */
    {$group:
     {
        _id: {state:"$state", city:"$city"},
        population: {$sum:"$pop"},
     }
    },
     /* trier comme sur population */
    {$sort:
     {"_id.state":1, "population":-1}
    }
])
```

## Phase 3

```
db.zips.aggregate([
    /* get the population of every city in every state */
    {$group:
     {
        _id: {state:"$state", city:"$city"},
        population: {$sum:"$pop"},
     }
    },
     /* trier comme sur population */
    {$sort:
     {"_id.state":1, "population":-1}
    },
    /* grouper par etat, prendre le premier de chaque groupe */
    {$group:
     {
        _id:"$_id.state",
        city: {$first: "$_id.city"},
        population: {$first:"$population"}
     }
    }
])
```

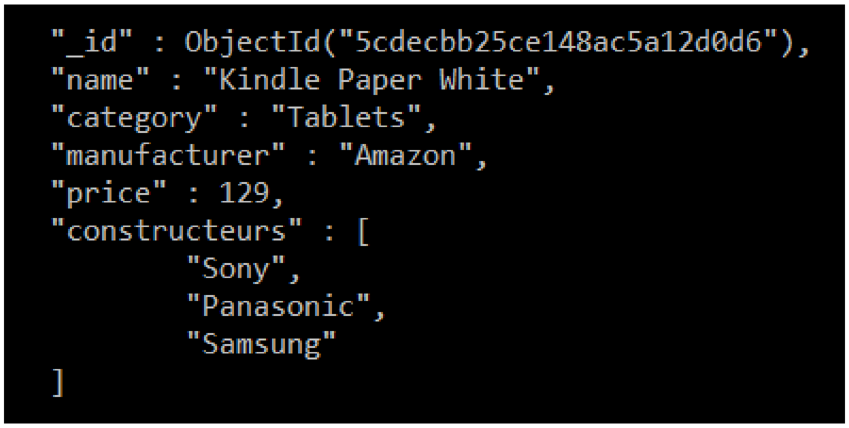
```
{ "_id" : "WV", "city" : "HUNTINGTON", "population" : 75343 }
{ "_id" : "WA", "city" : "SEATTLE", "population" : 520096 }
{ "_id" : "VT", "city" : "BURLINGTON", "population" : 39127 }
{ "_id" : "VA", "city" : "VIRGINIA BEACH", "population" : 385080 }
{ "_id" : "UT", "city" : "SALT LAKE CITY", "population" : 186346 }
```

## Unwind

Deconstruction d’un champ de type (array) et construction d’un document pour chaque élément de l’array

**Préparation :**

**> db.products.update({_id:ObjectId("5cdecbb25ce148ac5a12d0d6")},{$push: {"constructeurs": {$each : ["Sony","Panasonic","Samsung" ]} }})**



```
"_id" : ObjectId("5cdecbb25ce148ac5a12d0d6"),
"name" : "Kindle Paper White",
"category" : "Tablets",
"manufacturer" : "Amazon",
"price" : 129,
"constructeurs" : [
        "Sony",
        "Panasonic",
        "Samsung"
]
```

**Exécution :**

**> db.products.aggregate([{$match: {_id:ObjectId("5cdecbb25ce148ac5a12d0d6")}}, {$unwind : "$constructeurs"} ])**

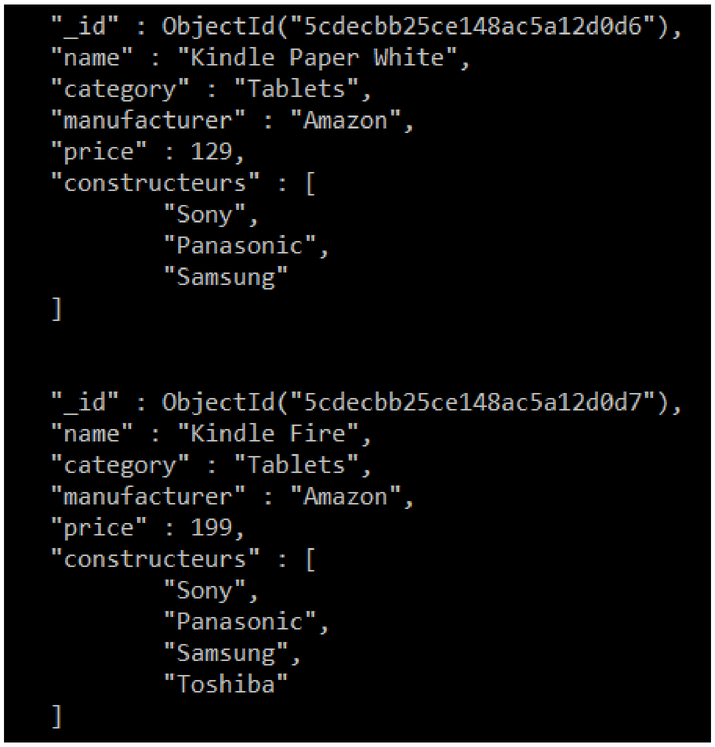
```
{ "_id" : ObjectId("5cdecbb25ce148ac5a12d0d6"), "name" : "Kindle Paper White", "category" : "Tablets",
"manufacturer" : "Amazon", "price" : 129, "constructeurs" : "Sony" }
{ "_id" : ObjectId("5cdecbb25ce148ac5a12d0d6"), "name" : "Kindle Paper White", "category" : "Tablets",
"manufacturer" : "Amazon", "price" : 129, "constructeurs" : "Panasonic" }
{ "_id" : ObjectId("5cdecbb25ce148ac5a12d0d6"), "name" : "Kindle Paper White", "category" : "Tablets",
"manufacturer" : "Amazon", "price" : 129, "constructeurs" : "Samsung" }
```

## sortByCount

Regrouper tout les documents selon une valeur spécifique, puis compter le nombre d’éléments au sein de chaque groupe distincs.

**Préparation :**

**> db.products.update({_id:ObjectId("5cdecbb25ce148ac5a12d0d7")},{$push: {"constructeurs": {$each : ["Sony","Panasonic","Samsung","Toshiba" ]} }})**



```
"_id" : ObjectId("5cdecbb25ce148ac5a12d0d6"),
"name" : "Kindle Paper White",
"category" : "Tablets",
"manufacturer" : "Amazon",
"price" : 129,
"constructeurs" : [
        "Sony",
        "Panasonic",
        "Samsung"
]


"_id" : ObjectId("5cdecbb25ce148ac5a12d0d7"),
"name" : "Kindle Fire",
"category" : "Tablets",
"manufacturer" : "Amazon",
"price" : 199,
"constructeurs" : [
        "Sony",
        "Panasonic",
        "Samsung",
        "Toshiba"
]
```

**Exécution :**

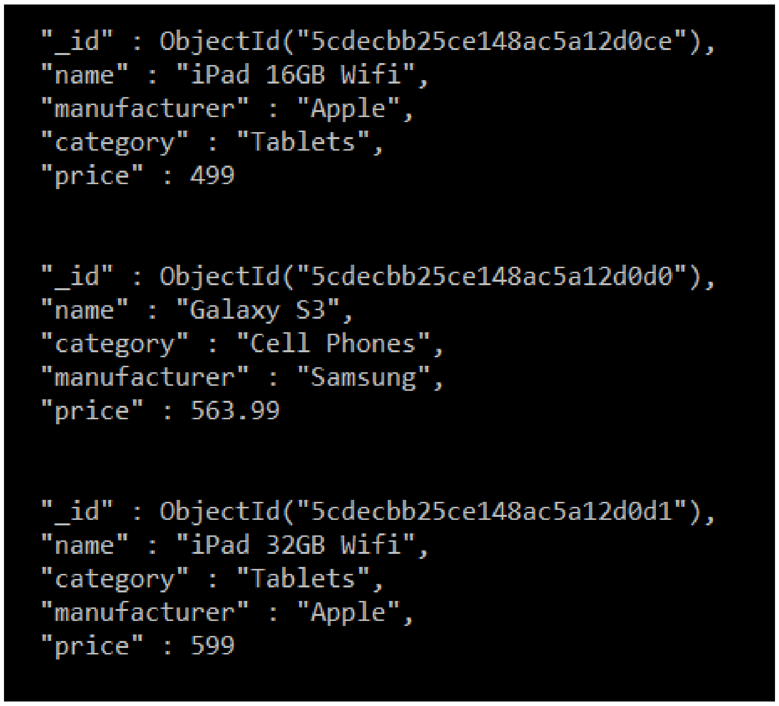
**> db.products.aggregate([{ $unwind: "$constructeurs" }, { $sortByCount: "$constructeurs" }])**

```
{ "_id" : "Samsung", "count" : 2 }
{ "_id" : "Panasonic", "count" : 2 }
{ "_id" : "Sony", "count" : 2 }
{ "_id" : "Toshiba", "count" : 1 }
```

## sample

Echantillonnage

**> db.products.aggregate([{$sample : {size: 3}}]).pretty()**



```
"_id" : ObjectId("5cdecbb25ce148ac5a12d0ce"),
"name" : "iPad 16GB Wifi",
"manufacturer" : "Apple",
"category" : "Tablets",
"price" : 499


"_id" : ObjectId("5cdecbb25ce148ac5a12d0d0"),
"name" : "Galaxy S3",
"category" : "Cell Phones",
"manufacturer" : "Samsung",
"price" : 563.99


"_id" : ObjectId("5cdecbb25ce148ac5a12d0d1"),
"name" : "iPad 32GB Wifi",
"category" : "Tablets",
"manufacturer" : "Apple",
"price" : 599
```

## Lookup

### Single Equality Join

Il permet de faire des jointures “gauches”. Pour chaque document source, il vérifie l’existance de la clé de jointure dans une collection externe. Si c’est le cas, il imbrique le(s) document(s) correspondant(s) dans une liste imbriquée dans le document (array)

**Préparation :**

```
> db.commandes.insert([
  { "_id" : 1, "nom" : "tee-shirt", "price" : 100, "couleur" : "blanc"   },
  { "_id" : 2, "nom" : "chaussure", "price" : 300, "couleur" : "blanc" },
  { "_id" : 3, "nom" : "chaussette", "price" : 20, "couleur" : "rouge" }
])

> db.stocks.insert([
  { "_id" : 1, "designation" : "tee-shirt", "qte" : 120, "etat" : "neuf" },
  { "_id" : 2, "designation" : "chaussure", "qte" : 80, "etat" : "neuf" },
  { "_id" : 3, "designation" : "chaussette", "qte" : 60, "etat" : "neuf" },
  { "_id" : 4, "designation" : "casquette", "qte" : 70, "etat" : "neuf" },
])
```

**Exécution :**

```
> db.commandes.aggregate([
  { $lookup: {
       from: "stocks",
       localField: "nom",
       foreignField: "designation",
       as: "stocks_produits_documents"
  } } ] )
```

```
{ "_id" : 1, "nom" : "tee-shirt", "price" : 100, "couleur" : "blanc", "stocks_produits_documents" : [ { "_id" : 1,
"designation" : "tee-shirt", "qte" : 120, "etat" : "neuf" } ] }
{ "_id" : 2, "nom" : "chaussure", "price" : 300, "couleur" : "blanc", "stocks_produits_documents" : [ { "_id" : 2,
"designation" : "chaussure", "qte" : 80, "etat" : "neuf" } ] }
...
```

## Lookup

### With an Array

**Préparation :**

```
> db.commandes.insert([
  { "_id" : 5, "nom" : "sandale", "price" : 200, "couleur" : "noir", production :
    ["rue montaigne", "paris", "france"] , "type" : "cuir" } ,
  { "_id" : 6, "nom" : "basket", "price" : 400, "couleur" : "noir", production :
    ["rue petit", "nantes", "france"] , "type" : "daim" }
  ])

> db.stocks.insert([
    { "_id" : 5, "designation" : "sandale", "qte" : 100, "etat" : "neuf",  "ville" : "paris" },
    { "_id" : 6, "designation" : "basket", "qte" : 50, "etat" : "neuf",  "ville" : "nantes" }
  ])
```

**Exécution :**

```
> db.commandes.aggregate([
  {
    $unwind: "$production"
  },
  {$lookup: {
          from: "stocks",
          localField: "production",
          foreignField: "ville",
          as: "stocks_produits_documents"
      }
  },
  {
    $match: { "stocks_produits_documents": { $ne: [] } }
  } ])
```

```
{ "_id" : 5, "nom" : "sandale", "price" : 200, "couleur" : "noir", "production" : "paris", "type" : "cuir",
"stocks_produits_documents" : [ { "_id" : 5, "designation" : "sandale", "qte" : 100, "etat" : "neuf", "ville" : "paris" }
] }
{ "_id" : 6, "nom" : "basket", "price" : 400, "couleur" : "noir", "production" : "nantes", "type" : "daim",
"stocks_produits_documents" : [ { "_id" : 6, "designation" : "basket", "qte" : 50, "etat" : "neuf", "ville" : "nantes" }
 ] }
```