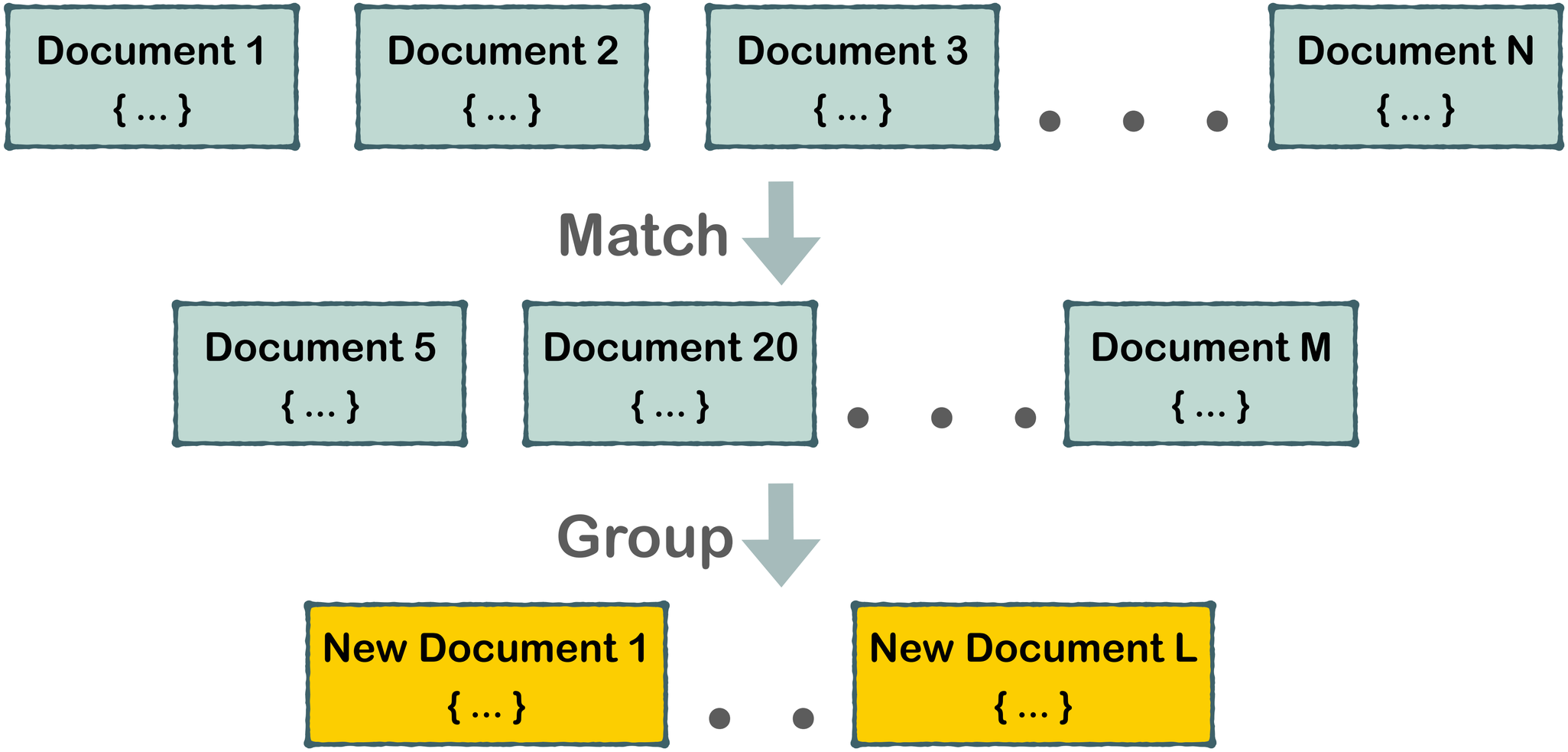
AGGREGATE

Aggregation Process



aggregate()

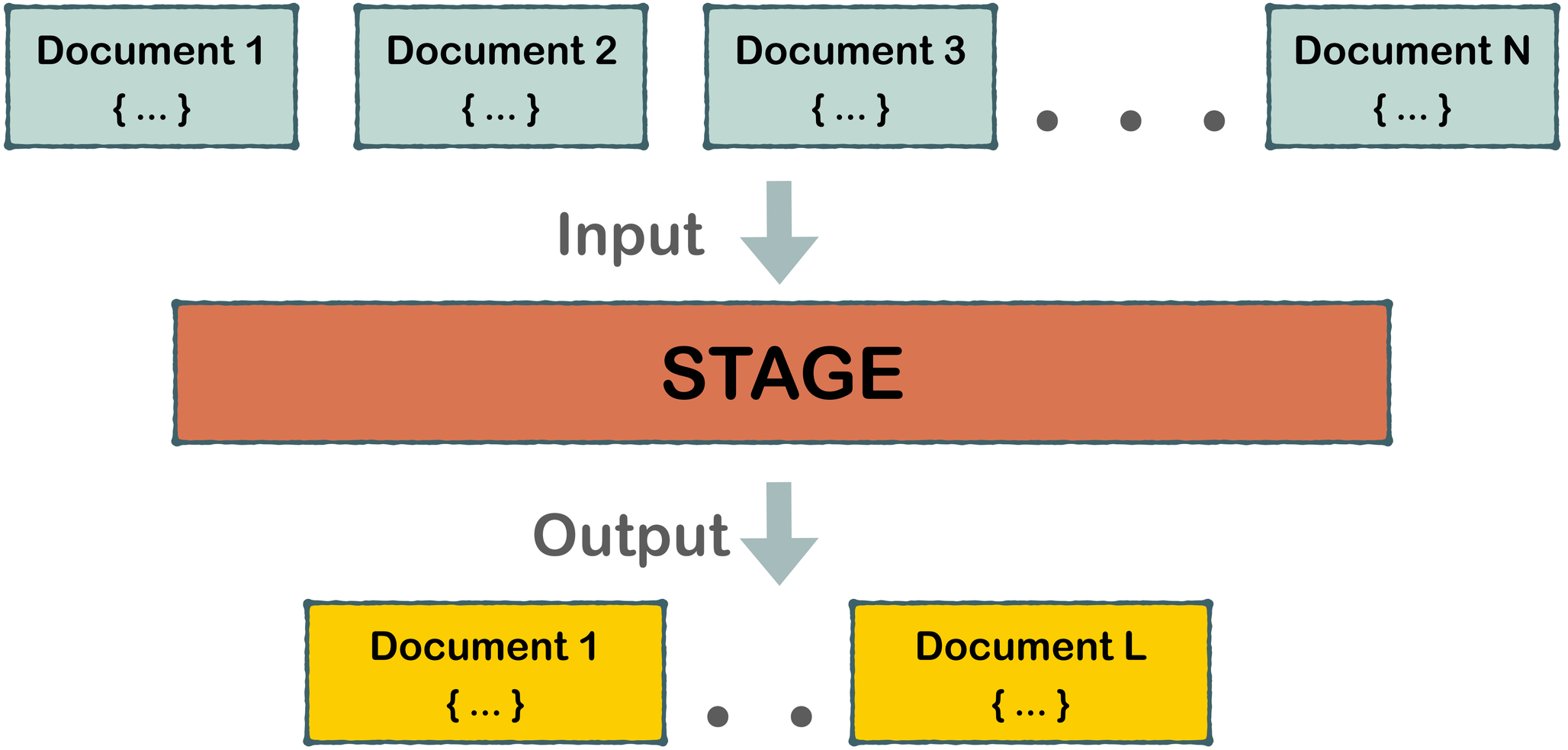
- Documents during aggregation process pass through the stages

```
db.<collection>.aggregate([
  <stage1>,
  <stage2>,
  ...
  <stageN>
])
```

Note

Aggregation request returns cursor from the server

Aggregation Stage



Aggregation Stage Operators

- Each stage starts from the stage operator

```
{ $<stageOperator> : {} }
```

- Examples

```
{ $match: { age: {$gt: 20} } }
{ $group: { _id: "$age"} }
{ $sort: { count: -1 } }
```

Aggregation Stages Overview

$match	$group	$project
$sort	$count	$limit
$skip	$out	

Aggregation Expressions

- Expression refers to the name of the field in input documents

```
"$<fieldName>"
```

- Examples

```
{ $group: { _id: "$age"} }
{ $group: { _id: "$company.location.country"} }
{ $group: { _id: "$name", total: {$sum:
"$price"}}}
```

$match Stage

- **Match specific documents using query**

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

Note

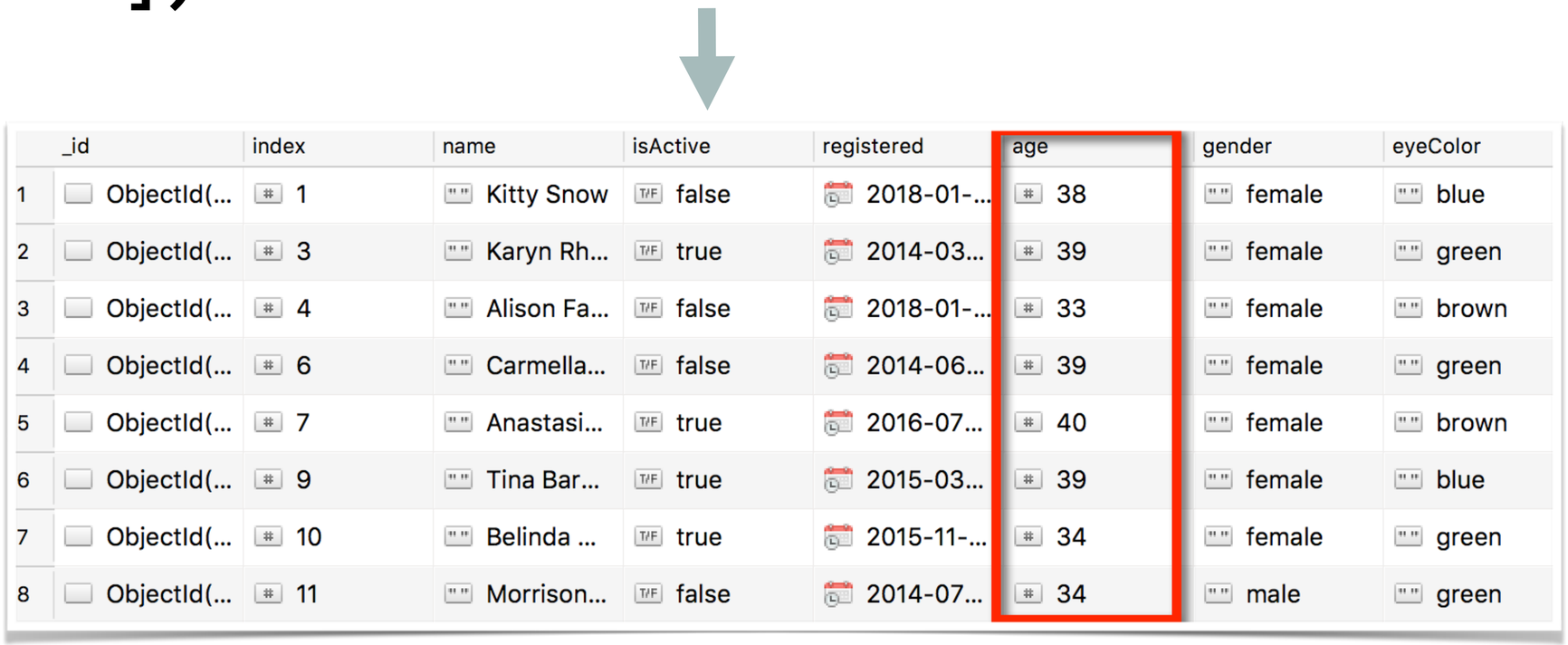
Match uses standard MongoDB queries and supports all query operators

- **Examples**

```
{ $match: { city: "New York" } }
{ $match: {age: {$gt: 25} } }
{ $match: {$and: [ {gender: "female"} ,
{age: {$gt: 25}} ] } }
```

Example 1: $match

```
db.persons.aggregate([
   { $match: { age: {$gt: 25} } }
])
```



	_id	index	name	isActive	registered	age	gender	eyeColor
1	ObjectId(...	1	Kitty Snow	false	2018-01-...	38	female	blue
2	ObjectId(...	3	Karyn Rh...	true	2014-03...	39	female	green
3	ObjectId(...	4	Alison Fa...	false	2018-01-...	33	female	brown
4	ObjectId(...	6	Carmella...	false	2014-06...	39	female	green
5	ObjectId(...	7	Anastasi...	true	2016-07...	40	female	brown
6	ObjectId(...	9	Tina Bar...	true	2015-03...	39	female	blue
7	ObjectId(...	10	Belinda ...	true	2015-11-...	34	female	green
8	ObjectId(...	11	Morrison...	false	2014-07...	34	male	green

$group Stage

- Groups input documents by certain expressions

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

Note

_id is Mandatory field

- Examples

```
{ $group: { _id: "$age"} }
```

```
{ $group: { _id: {age: "$age", gender:
"$gender"} } }
```

Example 2: $group

```
db.persons.aggregate([
   { $group: { _id: "$age" } }
])
```

```
{ "_id" : 27 }
{ "_id" : 30 }
{ "_id" : 26 }
{ "_id" : 31 }
{ "_id" : 23 }
{ "_id" : 37 }
      ...
```

Example 3: $group by nested Fields

```
db.persons.aggregate([
  { $group: { _id: "$company.location.country" } }
])
```

```
{ "_id" : "Germany" }
{ "_id" : "France" }
{ "_id" : "Italy" }
{ "_id" : "USA" }
```

Example 4: $group by multiple fields

```
db.persons.aggregate([
  { $group: { _id: {age: "$age", gender: "$gender"} } }
])
```

```
{ "_id" : { "age" : 27, "gender" : "male" } }
{ "_id" : { "age" : 37, "gender" : "female" } }
{ "_id" : { "age" : 22, "gender" : "female" } }
{ "_id" : { "age" : 26, "gender" : "male" } }
{ "_id" : { "age" : 27, "gender" : "female" } }
{ "_id" : { "age" : 38, "gender" : "male" } }
                    ...
```

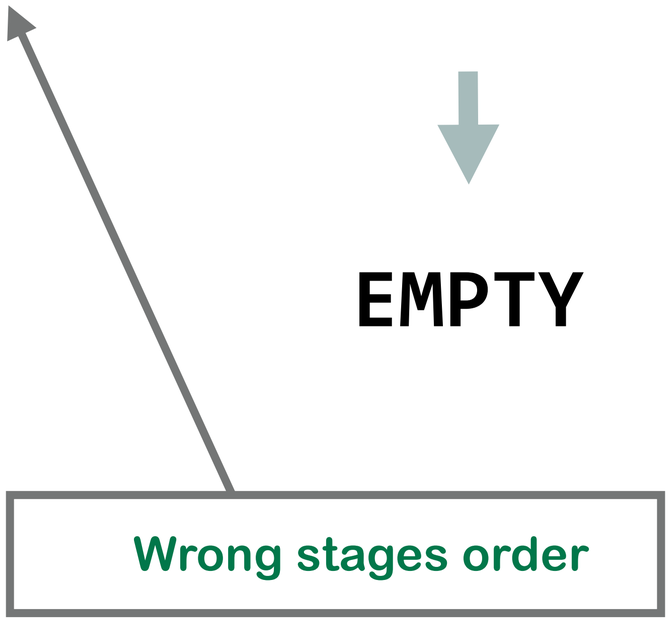
Example 5: $match and $group

```
db.persons.aggregate([
  { $match: { favoriteFruit: "banana" } },
  { $group: { _id: {age: "$age", eyeColor: "$eyeColor"} } }
])
```

```
{ "_id" : { "age" : 27, "eyeColor" : "blue" } }
{ "_id" : { "age" : 21, "eyeColor" : "brown" } }
{ "_id" : { "age" : 38, "eyeColor" : "brown" } }
{ "_id" : { "age" : 25, "eyeColor" : "blue" } }
{ "_id" : { "age" : 39, "eyeColor" : "blue" } }
{ "_id" : { "age" : 27, "eyeColor" : "green" }
                        ...
```

Example 6: $group and $match

```
db.persons.aggregate([
  { $group: { _id: {age: "$age", eyeColor: "$eyeColor"} } },
  { $match: { favoriteFruit: "banana" } }
])
```



Example 7: $group and $match

```
db.persons.aggregate([
  { $group: { _id: {age: "$age", eyeColor: "$eyeColor"} } },
  { $match: { "_id.age": {$gt: 30} } }
])
```

```
{ "_id" : { "age" : 38, "eyeColor" : "brown" } }
{ "_id" : { "age" : 33, "eyeColor" : "green" } }
{ "_id" : { "age" : 35, "eyeColor" : "brown" } }
{ "_id" : { "age" : 37, "eyeColor" : "green" } }
{ "_id" : { "age" : 39, "eyeColor" : "brown" } }
                    ...
```

$count Stage

- Counts number of the input documents

```
{ $count: "<title>" }
```

- Example

```
{ $count: "countries" }
```

```
{ "countries" : 4 }
```

Example 8: $count

```
db.persons.aggregate([
  { $count: "allDocumentsCount"}
])
```

```
{ "allDocumentsCount" : 1000 }
```

Different count methods

```
db.persons.aggregate([]).toArray().length
```

1,7 Sec → 1000

Client-side Count

```
db.persons.aggregate([]).itcount()
```

1,4 Sec → 1000

Server-side Count

```
db.persons.aggregate([{$count: "total"}])
```

0,21 Sec → `{ "total" : 1000 }`

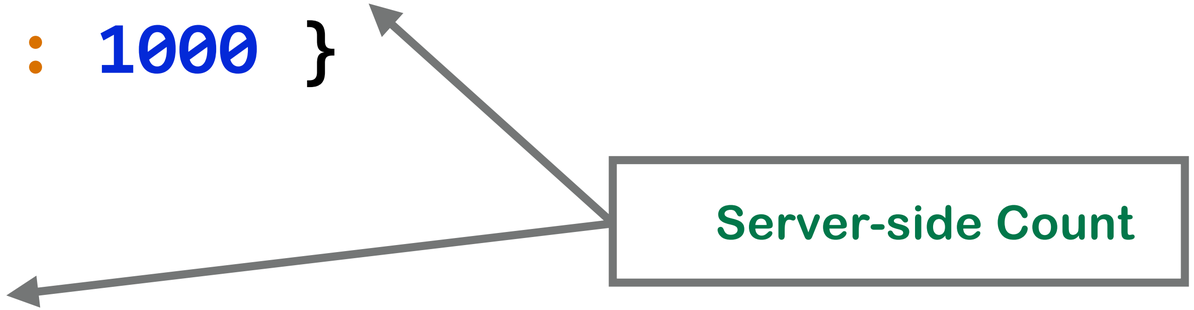
Compare Aggregate and Find count

```
db.persons.aggregate([{$count: "total"}])
```

0,21 Sec → `{ "total" : 1000 }`

```
db.persons.find({}).count()
```

0,21 Sec → `1000`



Note

Find count() is wrapper of the Aggregate $count

Example 9: $group and $count

```
db.persons.aggregate([
  { $group: { _id: "$company.location.country" } },
  { $count: "countriesCount"}
])
```

```
{ "countriesCount" : 4 }
```

$sort Stage

- Sorts input documents by certain field(s)

```
{ $sort: { <field1>: <-1 | 1>, <field2>: <-1 | 1> ... } }
```

- Examples

```
{ $sort: {score: -1} }
```

```
{ $sort: {age: 1 , country: 1} }
```

Note

<field>: 1 Ascending Order
<field>: -1 Descending Order

Example 10: $sort

```
db.persons.aggregate([
   { $sort: {name: 1}}
])
```

_id	index	name	isActive	registered	age	gender	eyeColor
ObjectId(...	29	Abby Wallace	false	2016-07...	40	female	green
ObjectId(...	989	Acevedo Wagner	true	2014-10-...	27	male	blue
ObjectId(...	658	Acosta Walter	false	2016-11-...	24	male	blue
ObjectId(...	988	Ada Hoover	false	2017-01-...	34	female	blue
ObjectId(...	497	Adams Hernandez	false	2014-09...	32	male	blue
ObjectId(...	113	Adeline Brewer	true	2016-10-...	24	female	green
ObjectId(...	258	Adrian Whitney	false	2015-10-...	34	female	brown
ObjectId(...	22	Agnes West	true	2014-03...	39	female	blue
ObjectId(...	510	Aguirre Cabrera	false	2017-06...	35	male	blue

Example 11: $group and $sort

```
db.persons.aggregate([
  { $group: { _id: "$favoriteFruit" } },
  { $sort: {_id: 1}}
])
```

```
{ "_id" : "apple" }
{ "_id" : "banana" }
{ "_id" : "strawberry" }
```

$project Stage

- Includes, Excludes or adds new field(s)

```
{ $project: { <field1>: <1>, <field2>: <0>, <newField1>:
<expression> ... } }
```

- Examples

```
{ $project: {name: 1, "company.title": 1} }
{ $project: {_id: 0 , name: 1 , age: 1} }
{ $project: {eyeColor: 0 , age: 0} }
{ $project: {name: 1 , newAge: "$age"} }
```

Example 12: $project

```
db.persons.aggregate([
  { $project: {name: 1, "company.location.country": 1} }
])
```

```
{
    "_id" : ObjectId("5ad4cbde2edbf6ddeec71741"),
    "name" : "Aurelia Gonzales",
    "company" : {
        "location" : {
            "country" : "USA"
        }
    }
}
...
```

Example 13: $project with new fields

```
db.persons.aggregate([
  {$project: {
    _id: 0,
    name: 1,
    info: {
      eyes: "$eyeColor",
      fruit: "$favoriteFruit",
      country:
"$company.location.country"
    }
  }}])
```

```
/* 1 */
{
    "name" : "Aurelia Gonzales",
    "info" : {
        "eyes" : "green",
        "fruit" : "banana",
        "country" : "USA"
    }
}
...
```

$limit Stage

- Outputs first N documents from the input

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

- Examples

```
{ $limit: 100 }
```

```
{ $limit: 1000 }
```

Note

$limit is usually used in:

1. **Sampled aggregation requests with $limit as first stage**
2. **After $sort to produce topN results**

Example 14: $limit, $match and $group

```
db.persons.aggregate([
   { $limit: 100},
   { $match: { age: {$gt: 27}} },
   { $group: { _id: "$company.location.country" } }
])
```

```
{ "_id" : "Germany" }
{ "_id" : "France" }
{ "_id" : "Italy" }
{ "_id" : "USA" }
```

$unwind Stage

- Splits each document with specified array to several documents - one document per array element

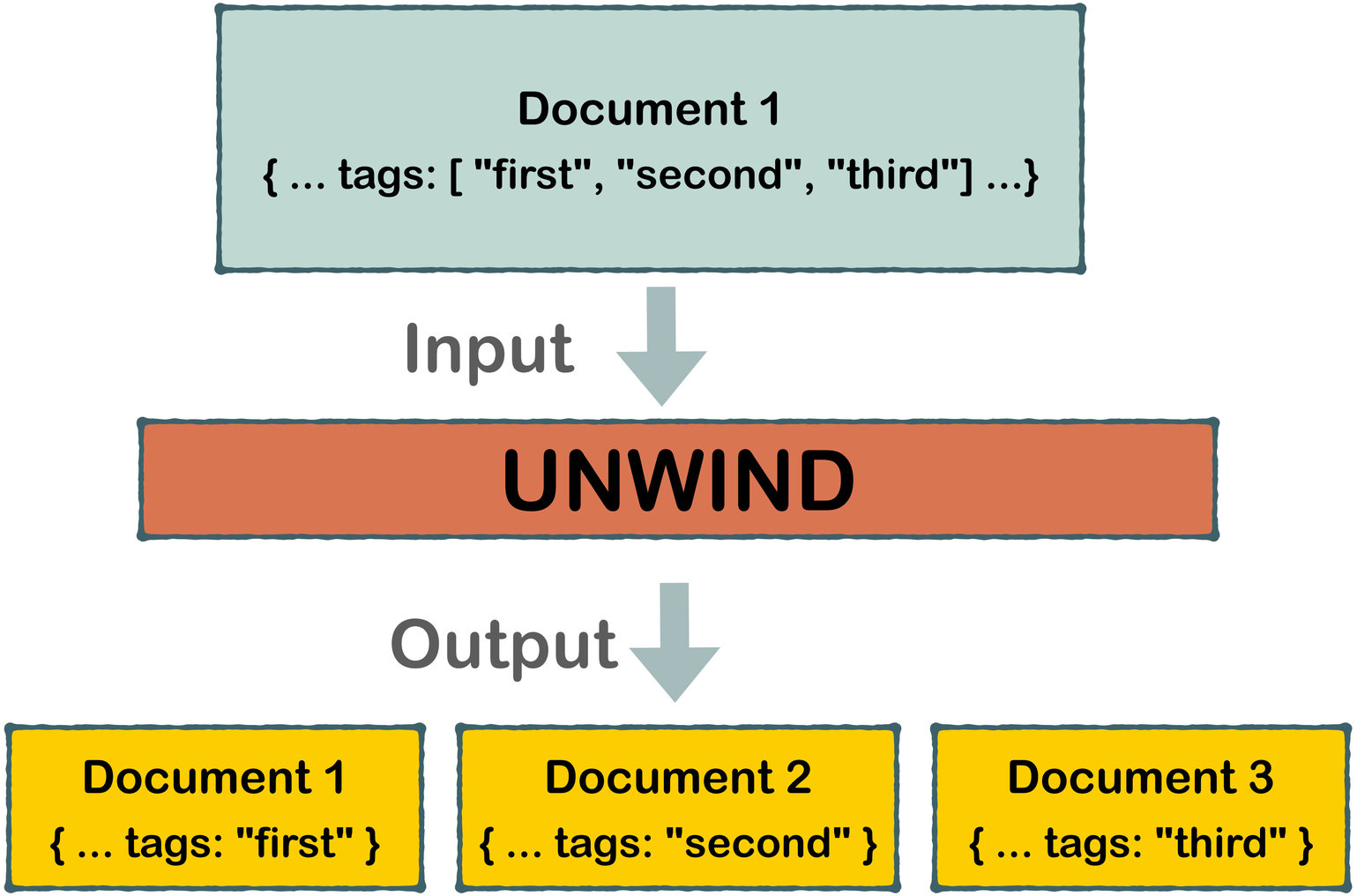
```
{ $unwind: <arrayReferenceExpression> }
```

- Examples

```
{ $unwind: "$tags" }
```

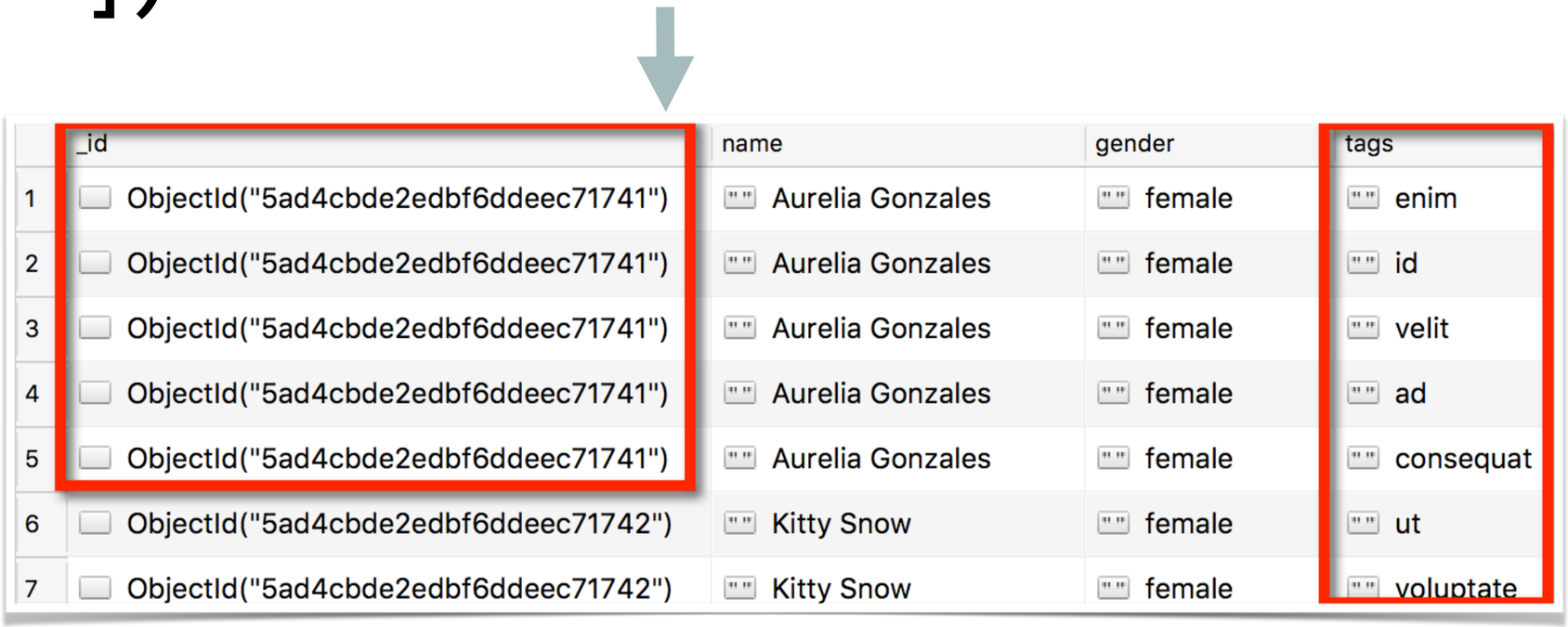
```
{ $unwind: "$hobbies" }
```

Unwind Stage Logic



Example 15: $unwind and $project

```
db.persons.aggregate([
  { $unwind: "$tags"},
  { $project: {name: 1, gender: 1, tags: 1}}
])
```



	_id	name	gender	tags
1	ObjectId("5ad4cbde2edbf6ddeec71741")	Aurelia Gonzales	female	enim
2	ObjectId("5ad4cbde2edbf6ddeec71741")	Aurelia Gonzales	female	id
3	ObjectId("5ad4cbde2edbf6ddeec71741")	Aurelia Gonzales	female	velit
4	ObjectId("5ad4cbde2edbf6ddeec71741")	Aurelia Gonzales	female	ad
5	ObjectId("5ad4cbde2edbf6ddeec71741")	Aurelia Gonzales	female	consequat
6	ObjectId("5ad4cbde2edbf6ddeec71742")	Kitty Snow	female	ut
7	ObjectId("5ad4cbde2edbf6ddeec71742")	Kitty Snow	female	voluptate

Example 16: $unwind and $group

```
db.persons.aggregate([
   { $unwind: "$tags"},
   { $group: { _id: "$tags" } }
])
```

```
{ "_id" : "nulla" }
{ "_id" : "reprehenderit" }
{ "_id" : "laboris" }
{ "_id" : "anim" }
{ "_id" : "consectetur" }
{ "_id" : "sit" }
                ...
```

Accumulators

$sum

$avg

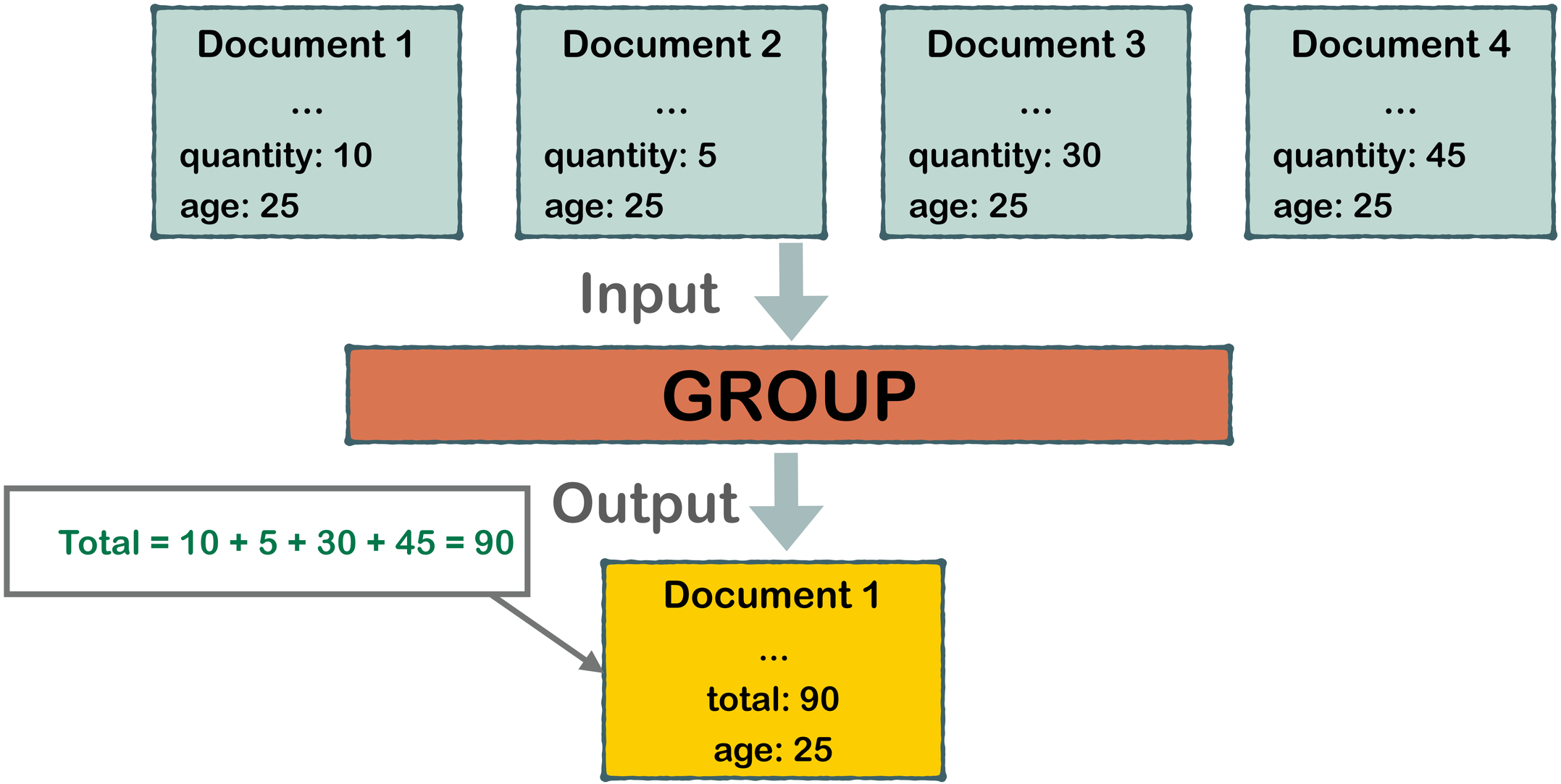
$max

$min

Note

Most accumulators are used only in the $group stage

Accumulators Logic ($sum Example)



Accumulators Syntax

- Accumulators maintain state for each group of the documents

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

- Examples

```
{ $sum: "$quantity" }
{ $avg: "$age" }
{ $max: "$spentMoney"}
```

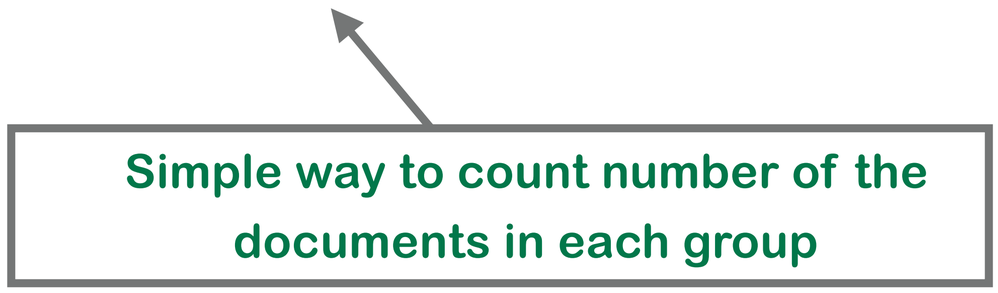
$sum Accumulator

- Sums numeric values for the documents in each group

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

- Examples

```
{ total: { $sum: "$quantity" } }
{ count: { $sum: 1 } }
```



Example 17: $sum and $group

```
db.persons.aggregate([
  {
    $group: {
      _id: "$age",
      count: { $sum: 1 }
    }
  }
])
```

```
{ "_id" : 27, "count" : 42 }
{ "_id" : 30, "count" : 38 }
{ "_id" : 26, "count" : 51 }
{ "_id" : 31, "count" : 53 }
{ "_id" : 23, "count" : 57 }
{ "_id" : 37, "count" : 49 }
{ "_id" : 32, "count" : 38 }
                ...
```

Example 18: $sum, $unwind and $group

```
db.persons.aggregate([
  { $unwind: "$tags" },
  {
    $group: {
      _id: "$tags",
      count: { $sum: 1 }
    }
  }
])
```



```
{ "_id" : "nulla", "count" : 65 }
{ "_id" : "reprehenderit", "count" : 57 }
{ "_id" : "laboris", "count" : 52 }
{ "_id" : "anim", "count" : 35 }
{ "_id" : "consectetur", "count" : 62 }
{ "_id" : "sit", "count" : 56 }
                    ...
```

$avg Accumulator

- Calculates average value of the certain values in the documents for each group

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

- Example

```
{ avgAge: { $avg: "$age" } }
```

Example 19: $avg and $group

```
db.persons.aggregate([
   {
      $group: {
         _id: "$eyeColor",
         avgAge: { $avg: "$age" }
      }
   }
])
```

```
{ "_id" : "brown", "avgAge" : 29.816023738872403 }
{ "_id" : "blue", "avgAge" : 30.033033033033032 }
{ "_id" : "green", "avgAge" : 29.654545454545456 }
                              ...
```

Unary Operators

$type

$or

$lt

$gt

$and

$multiply

Note

1. Unary Operators are usually used in the $project stage
2. In the $group stage Unary Operators can be used only in conjunction with Accumulators

$type Unary Operator

- Returns BSON type of the field's value

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

- Examples

```
{ $type: "$age" }
{ $type: "$name" }
```

Example 20: $type and $project

```
db.persons.aggregate([
  {
    $project: {
      name: 1,
      eyeColorType: { $type: "$eyeColor" },
      ageType: { $type: "$age" }
    }
  }
])
```

```
{
    "_id" : ObjectId("5ad4cbde2edbf6ddeec71741"),
    "name" : "Aurelia Gonzales",
    "eyeColorType" : "string",
    "ageType" : "int"
}
```

...

$out Stage

- Writes resulting documents to the MongoDB collection

```
{ $out: "<outputCollectionName>" }
```

- Example

```
{ $out: "newCollection" }
```

Notes

1. $out MUST be last stage in the pipeline
2. If output collection doesn't exist, it will be created automatically

Example 21: $out

```
db.persons.aggregate([
  { $group: { _id: {age: "$age", eyeColor: "$eyeColor"} }},
  { $out: "aggregationResults"}
])
```

Documents from the $group stage will be written to the collection "aggregationResults"

allowDiskUse: true

- All aggregation stages can use maximum 100 MB of RAM
- Server will return error if RAM limit is exceeded
- Following option will enable MongoDB to write stages data to the temporal files

```
{ allowDiskUse: true }
```

- Example

```
db.persons.aggregate([] , {allowDiskUse: true})
```

SUMMARY

- Aggregation Stages
  - `$group`
  - `$match`
  - `$sort`
  - `$project`
  - `$out`
- Stages chaining
- Accumulator Operators
  - `$sum`
  - `$avg`
- Unary Operators