**Aggregation Pipeline**

An aggregation pipeline consists of one or more [stages](https://www.mongodb.com/docs/manual/reference/operator/aggregation-pipeline/#std-label-aggregation-pipeline-operator-reference) that process documents:

* Each stage performs an operation on the input documents. For example, a stage can filter documents, group documents, and calculate values.
* The documents that are output from a stage are passed to the next stage.
* An aggregation pipeline can return results for groups of documents. For example, return the total, average, maximum, and minimum values.

|  |
| --- |
| db.orders.insertMany( [ |
|  |
| { \_id: 0, name: **"Pepperoni"**, size: **"small"**, price: 19, |
| quantity: 10, date: ISODate( **"2021-03-13T08:14:30Z"** ) }, |
| { \_id: 1, name: **"Pepperoni"**, size: **"medium"**, price: 20, |
| quantity: 20, date : ISODate( **"2021-03-13T09:13:24Z"** ) }, |
| { \_id: 2, name: **"Pepperoni"**, size: **"large"**, price: 21, |
| quantity: 30, date : ISODate( **"2021-03-17T09:22:12Z"** ) }, |
| { \_id: 3, name: **"Cheese"**, size: **"small"**, price: 12, |
| quantity: 15, date : ISODate( **"2021-03-13T11:21:39.736Z"** ) }, |
| { \_id: 4, name: **"Cheese"**, size: **"medium"**, price: 13, |
| quantity:50, date : ISODate( **"2022-01-12T21:23:13.331Z"** ) }, |
| { \_id: 5, name: **"Cheese"**, size: **"large"**, price: 14, |
| quantity: 10, date : ISODate( **"2022-01-12T05:08:13Z"** ) }, |
| { \_id: 6, name: **"Vegan"**, size: **"small"**, price: 17, |
| quantity: 10, date : ISODate( **"2021-01-13T05:08:13Z"** ) }, |
| { \_id: 7, name: **"Vegan"**, size: **"medium"**, price: 18, |
| quantity: 10, date : ISODate( **"2021-01-13T05:10:13Z"** ) } |
| ] ) |

Calculate Total Order Quantity

|  |
| --- |
| db.orders.aggregate( [ |
|  |
| *// Stage 1: Filter pizza order documents by pizza size* |
| { |
| $match: { size: **"medium"** } |
| }, |
|  |
| *// Stage 2: Group remaining documents by pizza name and calculate total quantity* |
| { |
| $group: { \_id: **"$name"**, totalQuantity: { $sum: **"$quantity"** } } |
| } |
|  |
| ] ) |

Update Documents Using an Aggregation Pipeline

To update documents with an aggregation pipeline, use:

| **Command** | **mongosh Methods** |
| --- | --- |
| [findAndModify](https://www.mongodb.com/docs/manual/reference/command/findAndModify/#mongodb-dbcommand-dbcmd.findAndModify) | [db.collection.findOneAndUpdate()](https://www.mongodb.com/docs/manual/reference/method/db.collection.findOneAndUpdate/#std-label-findOneAndUpdate-agg-pipeline)  [db.collection.findAndModify()](https://www.mongodb.com/docs/manual/reference/method/db.collection.findAndModify/#std-label-findAndModify-agg-pipeline) |
| [update](https://www.mongodb.com/docs/manual/reference/command/update/#mongodb-dbcommand-dbcmd.update) | [db.collection.updateOne()](https://www.mongodb.com/docs/manual/reference/method/db.collection.updateOne/#std-label-updateOne-example-agg)  [db.collection.updateMany()](https://www.mongodb.com/docs/manual/reference/method/db.collection.updateMany/#std-label-updateMany-example-agg)  [Bulk.find.update()](https://www.mongodb.com/docs/manual/reference/method/Bulk.find.update/#std-label-example-bulk-find-update-agg)  [Bulk.find.updateOne()](https://www.mongodb.com/docs/manual/reference/method/Bulk.find.updateOne/#std-label-example-bulk-find-update-one-agg)  [Bulk.find.upsert()](https://www.mongodb.com/docs/manual/reference/method/Bulk.find.upsert/#std-label-bulk-find-upsert-update-agg-example) |

Group

db. orders.aggregate(

[ { $group : { \_id : "$property\_type" } } ]

)

limit

db. orders.aggregate([ { $limit: 1 } ])

project

db. orders.aggregate([

{

$project: {

"name": 1,

"cuisine": 1,

"address": 1

}

},

{

$limit: 5

}

])

This will return the documents but only include the specified fields.

Notice that the \_id field is also included. This field is always included unless specifically excluded.

We use a 1 to include a field and 0 to exclude a field

Sort

db.listingsAndReviews.aggregate([

{

$sort: { "accommodates": -1 }

},

{

$project: {

"name": 1,

"accommodates": 1

}

},

{

$limit: 5

}

])

This will return the documents sorted in descending order by the accommodates field.

The sort order can be chosen by using 1 or -1. 1 is ascending and -1 is descending.

db.listingsAndReviews.aggregate([

{ $match : { property\_type : "House" } },

{ $limit: 2 },

{ $project: {

"name": 1,

"bedrooms": 1,

"price": 1

}}

])

This will only return documents that have the property\_type of "House".

db.restaurants.aggregate([

{

$match: { "cuisine": "Chinese" }

},

{

$count: "totalChinese"

}

])

This will return the number of documents at the $count stage as a field called "totalChinese".