mongoDB Documentation Search Documentation

Reference > Operators > Aggregation Pipeline Operators > Pipeline Aggregation Stages

Pipeline Aggregation Stages

On this page

> **NOTE:**
> For details on specific operator, including syntax and examples, click on the specific operator to go to its reference page.

Name	Description
`$collStats`	Returns statistics regarding a collection or view.
`$project`	Reshapes each document in the stream, such as by adding new fields or removing existing fields. For each input document, outputs one document.
`$match`	Filters the document stream to allow only matching documents to pass unmodified into the next pipeline stage. `$match` uses standard MongoDB queries. For each input document, outputs either one document (a match) or zero documents (no match).
`$redact`	Reshapes each document in the stream by restricting the content for each document based on information stored in the documents themselves. Incorporates the functionality of `$project` and `$match`. Can be used to implement field level redaction. For each input document, outputs either one or zero documents.
`$limit`	Passes the first *n* documents unmodified to the pipeline where *n* is the specified limit. For each input document, outputs either one document (for the first *n* documents) or zero documents (after the first *n* documents).
`$skip`	Skips the first *n* documents where *n* is the specified skip number and passes the remaining documents unmodified to the pipeline. For each input document, outputs either zero documents (for the first *n* documents) or one document (if after the first *n* documents).
`$unwind`	Deconstructs an array field from the input documents to output a document for *each* element. Each output document replaces the array with an element value. For each input document, outputs *n* documents where *n* is the number of array elements and can be zero for an empty array.
`$group`	Groups input documents by a specified identifier expression and applies the accumulator expression(s), if specified, to each group. Consumes all input documents and outputs one document per each distinct group. The output documents only contain the identifier field and, if specified, accumulated fields.

Name	Description
`$sample`	Randomly selects the specified number of documents from its input.
`$sort`	Reorders the document stream by a specified sort key. Only the order changes; the documents remain unmodified. For each input document, outputs one document.
`$geoNear`	Returns an ordered stream of documents based on the proximity to a geospatial point. Incorporates the functionality of `$match`, `$sort`, and `$limit` for geospatial data. The output documents include an additional distance field and can include a location identifier field.
`$lookup`	Performs a left outer join to another collection in the *same* database to filter in documents from the “joined” collection for processing.
`$out`	Writes the resulting documents of the aggregation pipeline to a collection. To use the `$out` stage, it must be the last stage in the pipeline.
`$indexStats`	Returns statistics regarding the use of each index for the collection.
`$facet`	Processes multiple aggregation pipelines within a single stage on the same set of input documents. Enables the creation of multi-faceted aggregations capable of characterizing data across multiple dimensions, or facets, in a single stage.
`$bucket`	Categorizes incoming documents into groups, called buckets, based on a specified expression and bucket boundaries.
`$bucketAuto`	Categorizes incoming documents into a specific number of groups, called buckets, based on a specified expression. Bucket boundaries are automatically determined in an attempt to evenly distribute the documents into the specified number of buckets.
`$sortByCount`	Groups incoming documents based on the value of a specified expression, then computes the count of documents in each distinct group.
`$addFields`	Adds new fields to documents. Outputs documents that contain all existing fields from the input documents and newly added fields.
`$replaceRoot`	Replaces a document with the specified embedded document. The operation replaces all existing fields in the input document, including the `_id` field. Specify a document embedded in the input document to promote the embedded document to the top level.
`$count`	Returns a count of the number of documents at this stage of the aggregation pipeline.
`$graphLookup`	Performs a recursive search on a collection. To each output document, adds a new array field that contains the traversal results of the recursive search for that document.

Additional Resources

- MongoDB Analytics: Learn Aggregation by Example: Exploratory Analytics and Visualization Using Flight Data
- MongoDB for Time Series Data: Analyzing Time Series Data Using the Aggregation Framework and Hadoop
- The Aggregation Framework
- Webinar: Exploring the Aggregation Framework
- Quick Reference Cards