MongoDB Shell
- Install
- Connect
- Configure
- Run Commands
- Perform CRUD Operations
- Run Aggregation Pipelines
- Client-Side Field Level Encryption
- Write Scripts
- Snippets
- Reference
- Release Notes
- Help

Run Aggregation Pipelines

On this page
- Understand the Aggregation Syntax
- Example

You can run aggregation pipelines on your collections using the MongoDB Shell. Aggregation pipelines transform your documents into aggregated results based on selected pipeline stages.

Common uses for aggregation include:

- Grouping data by a given expression.
- Calculating results based on multiple fields and storing those results in a new field.
- Filtering data to return a subset that matches a given criteria.
- Sorting data.

When you run an aggregation, MongoDB Shell outputs the results directly to the terminal.

Understand the Aggregation Syntax

The MongoDB aggregation pipeline consists of stages. Each stage transforms the documents as they pass through the pipeline. Pipeline stages do not need to produce one output document for every input document; e.g., some stages may generate new documents or filter out documents.

To create an aggregation pipeline, use the following syntax in the MongoDB Shell:

```
db.<collection>.aggregate([
  {
    <$stage1>
  },
  {
    <$stage2>
  }
  ...
])
```

Example

The examples on this page reference the Atlas sample dataset. You can create a free Atlas cluster and populate that cluster with sample data to follow along with these examples. To learn more, see Get Started with Atlas.

The example below uses the `movies` collection in the Atlas sample_mflix sample dataset.

Example Document

Each document in the `movies` collection describes a movie:

```
{
  _id: 573a1397f29313caabce8347,
  fullplot: 'In a cyberpunk vision of the future, man has developed the techn
  imdb: { rating: 8.2, votes: 419589, id: 83658 },
  year: 1982,
  plot: 'A blade runner must pursue and try to terminate four replicants who
  genres: [ 'Sci-Fi', 'Thriller' ],
  rated: 'R',
  metacritic: 88,
  title: 'Blade Runner',
  lastupdated: '2015-09-04 00:05:51.990000000',
  languages: [ 'English', 'German', 'Cantonese', 'Japanese', 'Hungarian' ],
  writers: [
    'Hampton Fancher (screenplay)',
    'David Webb Peoples (screenplay)',
    'Philip K. Dick (novel)'
  ],
  type: 'movie',
  tomatoes: {
    viewer: { rating: 4, numReviews: 331213, meter: 91 },
    dvd: 1997-08-27T00:00:00.000Z,
    critic: { rating: 8.5, numReviews: 102, meter: 90 },
    lastUpdated: 2015-09-12T17:48:21.000Z,
    consensus: "Misunderstood when it first hit theaters, the influence of Ri
    rotten: 10,
    production: 'Warner Bros. Pictures',
    fresh: 92
  },
  poster: 'https://m.media-amazon.com/images/M/MV5BNzQzMzJhZTEtOWM4NS00MTdhLT
  num_mflix_comments: 1,
  released: 1982-06-25T00:00:00.000Z,
  awards: {
    wins: 13,
    nominations: 15,
    text: 'Nominated for 2 Oscars. Another 11 wins & 15 nominations.'
  },
  countries: [ 'USA', 'Hong Kong', 'UK' ],
  cast: [
    'Harrison Ford',
    'Rutger Hauer',
    'Sean Young',
    'Edward James Olmos'
  ],
  directors: [ 'Ridley Scott' ],
  runtime: 117
}
```

The documents aggregated in this tutorial reside in the `sample_mflix.movies` collection. Use the following command to switch to the database that contains this collection:

```
use sample_mflix
```

Example Aggregation Pipeline

Consider the following pipeline:

```
db.movies.aggregate([

  // First Stage

  { $project: { _id: 0, genres: 1, imdb: 1, title: 1 } },

  // Second Stage

  { $unwind: "$genres" },

  // Third Stage

  { $group:
    { _id: "$genres",
      averageGenreRating: { $avg: "$imdb.rating" }
    }
  },

   // Fourth Stage

   { $sort: { averageGenreRating: -1 } }
] )
```

This pipeline performs an aggregation in four stages:

First Stage

The `$project` stage passes documents that contain the following fields to the next pipeline stage:

- `genres`
- `imdb`
- `title`

Documents from the collection that don't include all of these fields are not passed to the next pipeline stage.

> **NOTE**
>
> The `$project` stage specifies `_id: 0` to suppress the `_id` field from the documents it passes to the next stage.
>
> For more information, see the MongoDB Manual.

The `$project` stage transforms the example document and passes the following output to the next pipeline stage:

```
{
  imdb: { rating: 8.2, votes: 419589, id: 83658 },
  genres: [ 'Sci-Fi', 'Thriller' ],
  title: 'Blade Runner'
}
```

Second Stage

The `$unwind` stage passes a document for each element in the `genres` array to the next pipeline stage.

The `$unwind` stage generates the following two documents from the original example document, then passes them to the next pipeline stage:

```
{
  imdb: { rating: 8.2, votes: 419589, id: 83658 },
  genres: 'Sci-Fi',
  title: 'Blade Runner'
}
```

```
{
  imdb: { rating: 8.2, votes: 419589, id: 83658 },
  genres: 'Thriller',
  title: 'Blade Runner'
}
```

Third Stage

The `$group` stage:

- Retrieves the distinct genre values from the documents it receives from the previous pipeline stage,
- Creates a document for each distinct genre where the `_id` is the genre name,
- Adds a field, `averageGenreRating`, to each new document that contains the average `imdb.rating` of all documents that match the genre, and
- Passes the new documents to the next pipeline stage.

This stage sends documents that look similar to the following to the next pipeline stage:

```
{ _id: 'Sport', averageGenreRating: 6.781233933161954 },
{ _id: 'History', averageGenreRating: 7.202306920762287 },
{ _id: 'Biography', averageGenreRating: 7.097142857142857 },
{ _id: 'Adventure', averageGenreRating: 6.527788649706458 },
{ _id: 'Family', averageGenreRating: 6.36096256684492 },
{ _id: 'Crime', averageGenreRating: 6.730478683620045 },
{ _id: 'Western', averageGenreRating: 6.879197080291971 },
{ _id: 'Fantasy', averageGenreRating: 6.42495652173913 },
{ _id: 'Talk-Show', averageGenreRating: 7 },
{ _id: 'Documentary', averageGenreRating: 7.365266635205286 },
{ _id: 'War', averageGenreRating: 7.183944374209861 },
{ _id: 'Short', averageGenreRating: 7.355813953488372 },
{ _id: 'Horror', averageGenreRating: 5.84110718492344 },
{ _id: 'Film-Noir', averageGenreRating: 7.503809523809523 },
{ _id: 'News', averageGenreRating: 7.254901960784314 },
{ _id: 'Thriller', averageGenreRating: 6.322121555303888 },
{ _id: 'Action', averageGenreRating: 6.3774842271293375 },
{ _id: 'Music', averageGenreRating: 6.923452380952381 },
{ _id: 'Animation', averageGenreRating: 6.917993795243019 },
{ _id: 'Drama', averageGenreRating: 6.830528688822631 }
```

Fourth Stage

The `$sort` stage sorts the documents it receives from the previous stage in descending order based on the value of the `averageGenreRating` field.

When you run the example pipeline, the MongoDB Shell prints documents similar to the following to the terminal:

```
[
  { _id: 'Film-Noir', averageGenreRating: 7.503809523809523 },
  { _id: 'Documentary', averageGenreRating: 7.365266635205286 },
  { _id: 'Short', averageGenreRating: 7.355813953488372 },
  { _id: 'News', averageGenreRating: 7.254901960784314 },
  { _id: 'History', averageGenreRating: 7.202306920762287 },
  { _id: 'War', averageGenreRating: 7.183944374209861 },
  { _id: 'Biography', averageGenreRating: 7.097142857142857 },
  { _id: 'Talk-Show', averageGenreRating: 7 },
  { _id: 'Music', averageGenreRating: 6.923452380952381 },
  { _id: 'Animation', averageGenreRating: 6.917993795243019 },
  { _id: 'Western', averageGenreRating: 6.879197080291971 },
  { _id: 'Drama', averageGenreRating: 6.830528688822631 },
  { _id: 'Sport', averageGenreRating: 6.781233933161954 },
  { _id: 'Crime', averageGenreRating: 6.730478683620045 },
  { _id: 'Musical', averageGenreRating: 6.696913580246913 },
  { _id: 'Romance', averageGenreRating: 6.695711554220159 },
  { _id: 'Mystery', averageGenreRating: 6.563317384370015 },
  { _id: 'Adventure', averageGenreRating: 6.527788649706458 },
  { _id: 'Comedy', averageGenreRating: 6.479626461362988 },
  { _id: 'Fantasy', averageGenreRating: 6.42495652173913 }
]
```

See also:

- To learn more about the available aggregation stages, see Aggregation Pipeline Stages.
- To learn more about the available aggregation operators you can use within stages, see Aggregation Pipeline Operators.