The Aggregation Pipeline

Sarah Evans: Demo Lesson

Defining Terms and Assumptions

Assumptions

Audience:

- Developers who are familiar with MongoDB and MongoDB Atlas, but new to the aggregation pipeline
- Users already have a cluster set up in MongoDB Atlas and know how to access sample data

Demo Lesson Plan

Lesson Objective

By the end of this lesson, you will be able to:

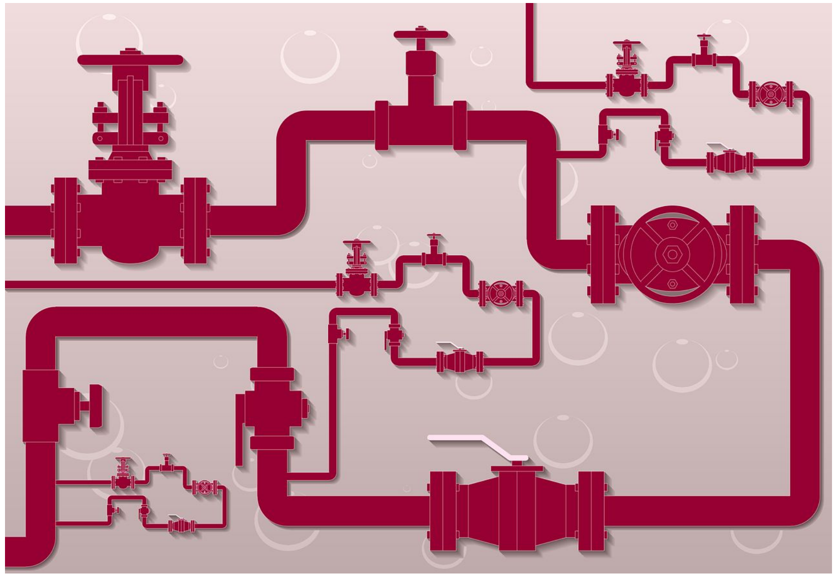
- **Explain the aggregation pipeline and how it differs from the MQL**

Key Terms

- Aggregation Framework
- Aggregation pipeline
- Stage
- Aggregate operator
- $match
- $project

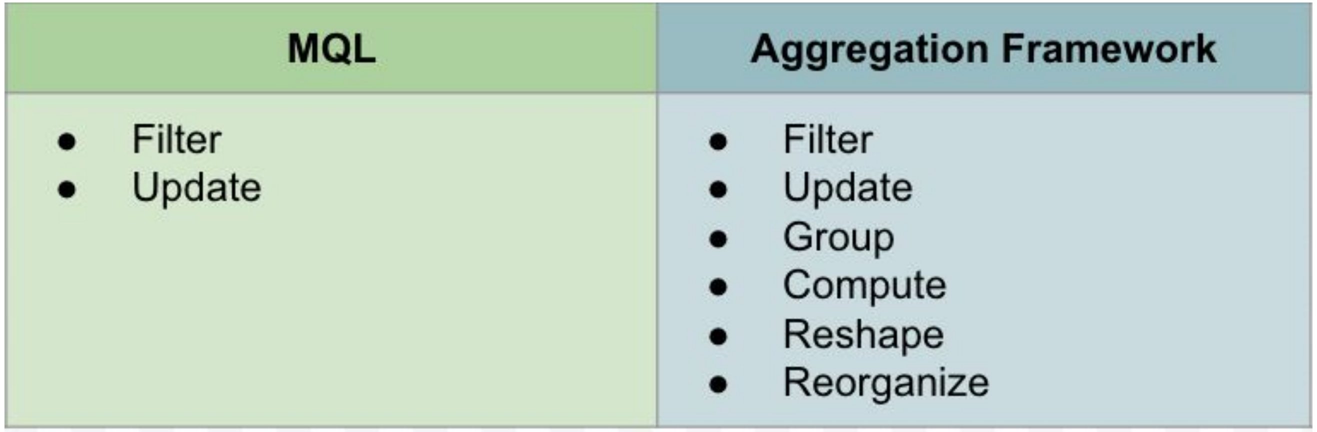
The Aggregation Pipeline

- MongoDB's **Aggregation Framework** was created as a powerful and efficient solution to the limitations of MQL
- An **aggregation pipeline** can be configured to perform all kinds of complex tasks against the data in a database
- But at its core, it is just another way to query and process data!



Aggregation Pipeline

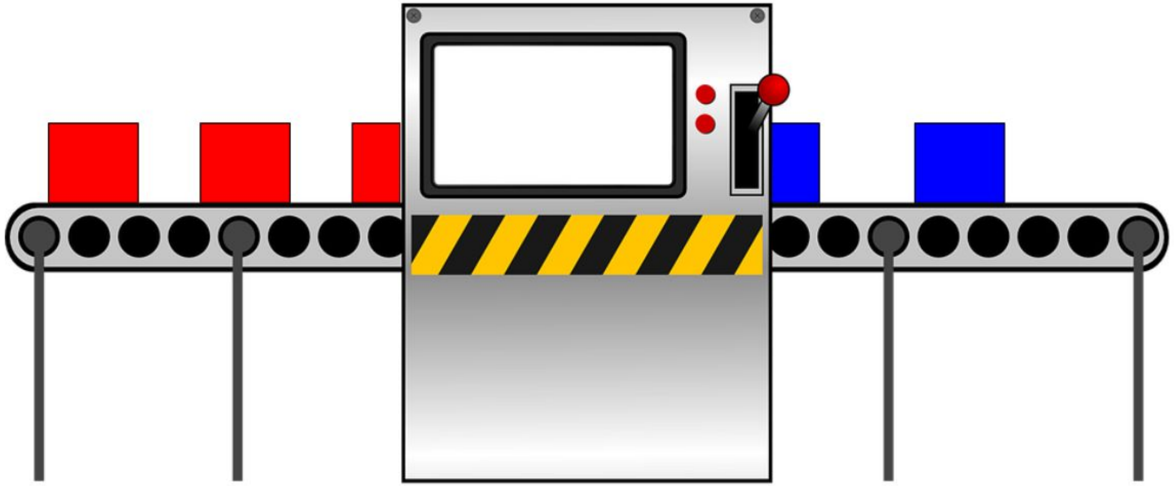
- Like MQL, an aggregation pipeline can be used to filter and update data in MongoDB
- But it can be used to do much more than is possible with MQL



MQL	Aggregation Framework
• Filter • Update	• Filter • Update • Group • Compute • Reshape • Reorganize

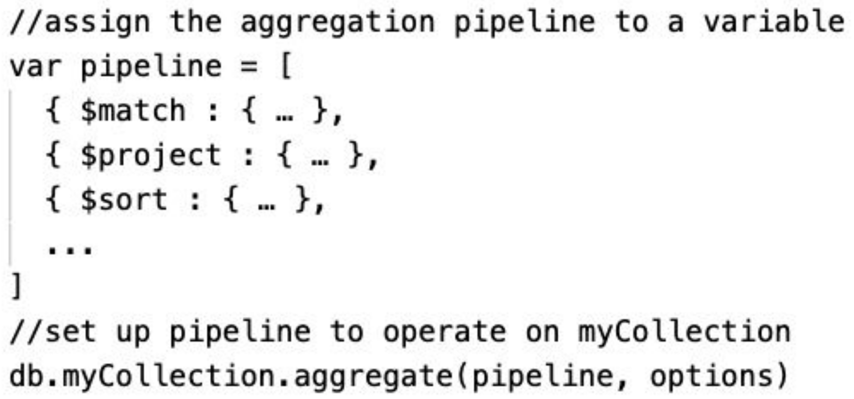
How does it work?

We can think of a pipeline as an assembly line that operates on data from a collection, transforms the data in a specific sequence of **stages**, and then emits the transformed data



Stages

- Aggregation pipelines are arrays that contain one or more stages
- Each **stage** is configured to process or transform data from a collection
- Stages are set up as a JSON object with key/value pairs, where the key is an **aggregate operator**
- The purpose of the **$match** stage is to filter documents - it uses MQL query operators

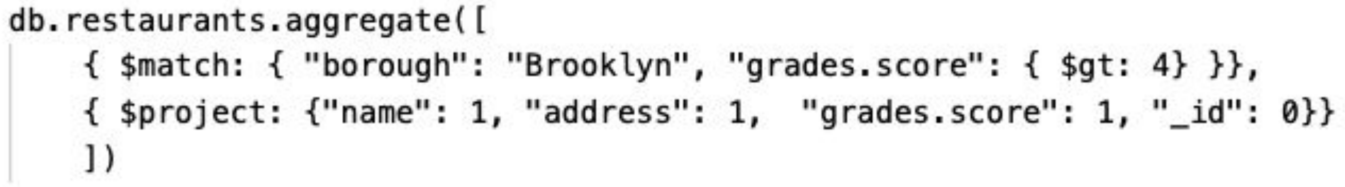


```
//assign the aggregation pipeline to a variable
var pipeline = [
  { $match : { … },
  { $project : { … },
  { $sort : { … },
  ...
]
//set up pipeline to operate on myCollection
db.myCollection.aggregate(pipeline, options)
```

Let’s Compare

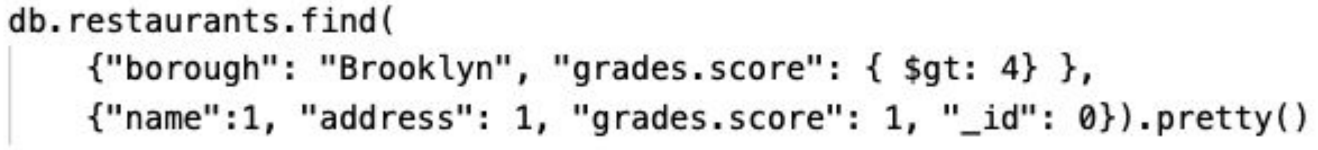
Aggregation

Pipeline



```
db.restaurants.aggregate([
    { $match: { "borough": "Brooklyn", "grades.score": { $gt: 4} }},
    { $project: {"name": 1, "address": 1,  "grades.score": 1, "_id": 0}}
    ])
```

MQL



```
db.restaurants.find(
    {"borough": "Brooklyn", "grades.score": { $gt: 4} },
    {"name":1, "address": 1, "grades.score": 1, "_id": 0}).pretty()
```

Knowledge Check

Iman is working with a large collection of documents that contain book data. She needs to find all of the books in the collection that have the word “mountain” in the title. Would you recommend that she set up an aggregation pipeline for this task?

- No, a simple query with MQL would work in this case.
- If, after finding all books with “mountain” in the title, Iman wished to sort the results by genre and calculate the average rating, then setting up a pipeline for this task would be a good idea!

Key Takeaways:

- An aggregation pipeline is just another way to query and update data in MongoDB
- A pipeline is an array of stages that transform data in a specific order
- Both MQL and aggregation pipelines can filter and update data, but a pipeline can also perform many other complex tasks