



APAC

Simplify Data Processing with the MongoDB Aggregation Framework

Environment Setup:
mdb.link/instruqt-jedee

MongoDB
Build & Learn Workshops Series



Hi, I'm Wen Jie!



- Senior Developer Advocate at MongoDB
- Software Engineer / Tech Lead in both public and private sectors
- Full-stack but hates frontend work
- > 50 Design Reviews in a year

[Teo Wen Jie](#)

Connect with me on LinkedIn





Skill Badge Benefits

Formal Validation

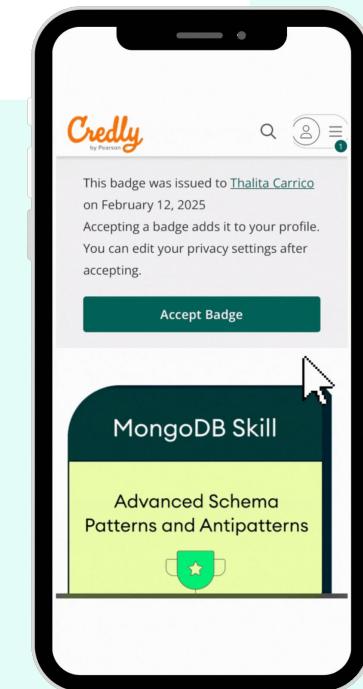
Provide a way to quickly learn and validate specific MongoDB skills

Elevate in current roles and increase appeal for future positions.

Career Advancement

Digital Badges

Showcase achievements with a Credly badge and inclusion in the Talent Directory.

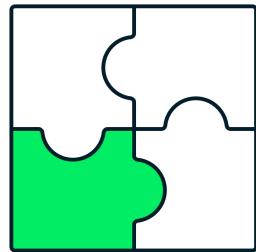


Agenda

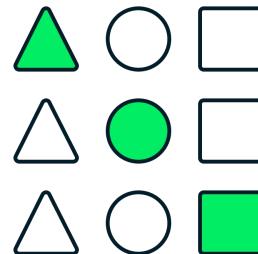
- A brief overview of MongoDB Document Model
- MongoDB Query API
 - CRUD
 - Aggregation Framework
- Common Aggregation Stages
 - 👉👉 Simple pipelines
 - 👉👉 Arrays, counting, and sorting
 - 👉👉 Joins & working with results
- ⭐ Skill Badge Challenge



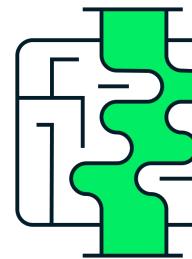
Core MongoDB design concepts



Embedding



Arrays



Polymorphism

MongoDB document

```
{  
  _id: "9781602833364",  
  title: "Brave New World",  
  category: "books",  
  price: 14.99,  
  author: "Aldous Huxley",  
  genres: [ "science fiction", "novel" ],  
  reviews: [  
    { name: "John", text: "Amazing!" },  
    { name: "Jane", text: "Incredible book!" }  
  ]  
}
```





Related documents are stored together in **collections**.

```
{  
  _id: "9781602833364",  
  title: "Brave New World",  
  category: "books",  
  price: 14.99,  
  author: "Aldous Huxley",  
  genres: [ "science fiction", "novel" ],  
  reviews: [  
    { user: "John", text: "Amazing!" },  
    { user: "Jane", text: "Incredible book!" }  
  ]  
}
```



Related
documents are
stored together
in **collections**.

```
{  
  _id: ObjectId("686a190d137d5873152d7da6")  
  title: "Brave New World",  
  category: "books",  
  price: 14.99,  
  author: "Aldous Huxley",  
  genres: [ "science fiction", "novel" ],  
  reviews: [  
    { user: "John", text: "Amazing!" },  
    { user: "Jane", text: "Incredible book!" }  
  ]  
}
```



Collections can be **polymorphic**—storing documents with different shapes.

```
{  
  _id: "9781602833364",  
  title: "Brave New World",  
  author: "Aldous Huxley",  
  type: "print",  
  condition: "excellent"  
}
```

```
{  
  _id: "9798354502691",  
  title: "Frankenstein",  
  author: "Mary Shelley",  
  type: "e-book",  
  format: "epub",  
  fileSizeMB: 2.5,  
  supportedDevices: [  
    "Kindle",  
    "iPad",  
    "Android",  
    "PC"  
  ]  
}
```



When modeling data for
MongoDB, your workload
should drive your data
model.

Agenda

- A brief overview of last week's session
- **MongoDB Query API**
 - CRUD
 - Aggregation Framework
- Common Aggregation Stages
 - 👉 Simple pipelines
 - 👉 Arrays, counting, and sorting
 - 👉 Joins & working with results
- ⭐ Skill Badge Challenge

Agenda

- A brief overview of last week's session
- **MongoDB Query API**
 - CRUD**
 - Aggregation Framework
 - Common Aggregation Stages
 - 👉 Simple pipelines
 - 👉 Arrays, counting, and sorting
 - 👉 Joins & working with results
 - Skill Badge Challenge



MongoDB Query API Syntax Structure

Keyword

Collection Operator
db.authors.insertOne()

Document

```
{  
  "name": "Jane Austen",  
  "books": [ "0141439688", "0375757813", "1551114798" ],  
  "aliases": [ "Austen, Jane", "Jane Austen" ]  
}  
)
```



Create

```
db.authors.insertOne({  
    "name": "Jane Austen",  
    "books": [ "0141439688", "0375757813", "1551114798" ],  
    "aliases": [ "Austen, Jane", "Jane Austen" ]  
})
```

```
db.authors.insertMany([{...}, {...}, ...])
```



Read

```
db.authors.find({  
    "name": "Jane Austen"  
})
```

```
db.authors.findOne({...})
```



Update

```
db.authors.updateOne(  
  { name: "Jane Austin" },  
  { $set:  
    { name: "Jane Austen" }  
  }  
)  
  
db.authors.update({...},{...})
```



Delete

```
db.authors.deleteOne(  
  { name: "Jane Austen" }  
)
```

```
db.authors.deleteMany({...})
```



MongoDB Query API Syntax Structure

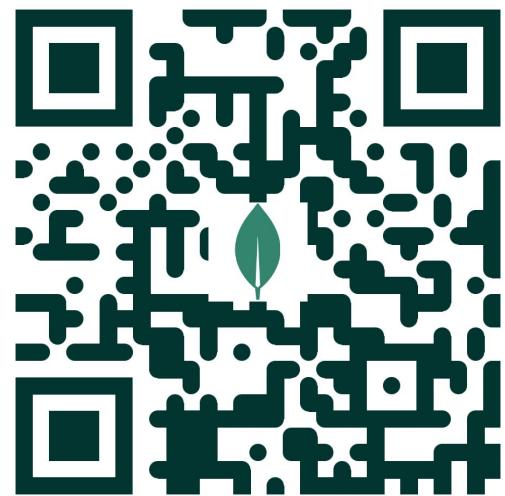


```
// Example MongoDB find command
```

```
db.collection.find(  
  { age: { $gte: 25 } }, // query or filter: where age is greater than or equal to 25  
  {  
    projection: { name: 1, age: 1 }, // options  
    sort: { age: -1 },  
    limit: 10  
  }  
);
```

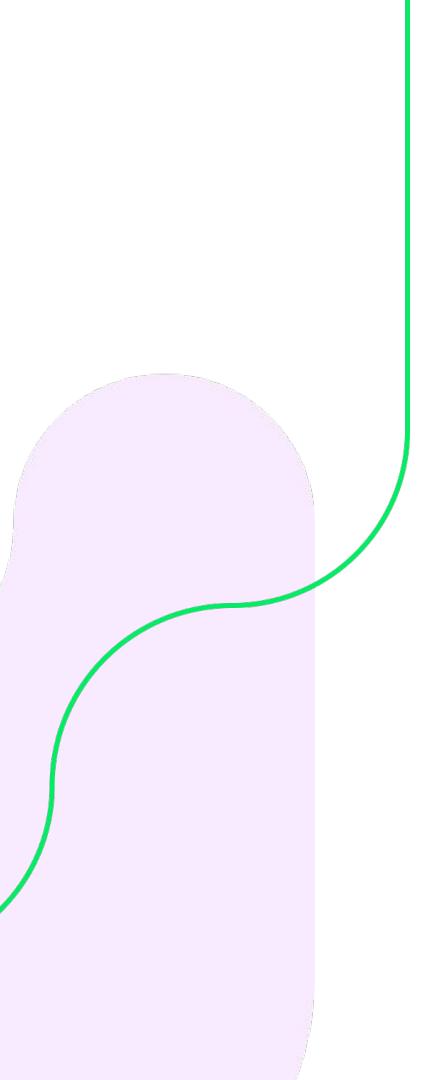


Check out the
MongoDB docs
for more
details



Agenda

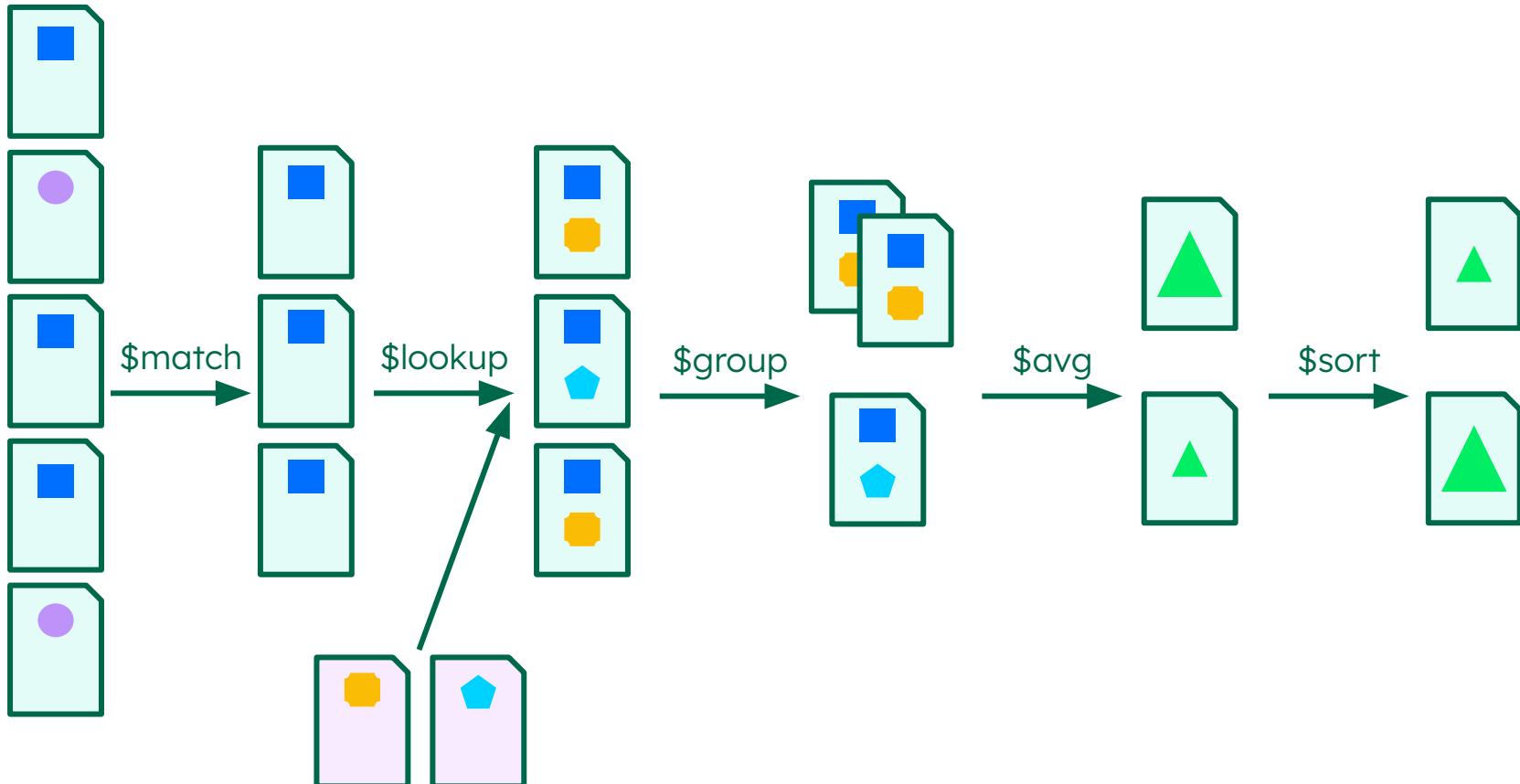
- A brief overview of last week's session
- **MongoDB Query API**
 - CRUD
 - Aggregation Framework**
 - Common Aggregation Stages
 - 👉 Simple pipelines
 - 👉 Arrays, counting, and sorting
 - 👉 Joins & working with results
 - Skill Badge Challenge



The aggregation framework allows you to transform and analyze your data in real time.

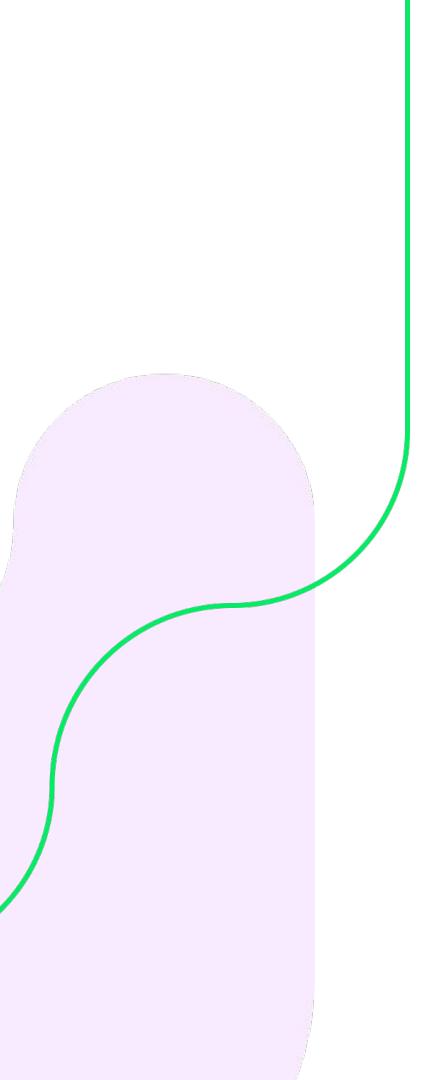


Multi-stage aggregation pipelines



SQL vs Aggregation Pipelines





Get authors' bios
with books that
have an average
rating greater
than 4

Get authors' bios with books that have an average rating greater than 4-star

```
SELECT
    authors.name,
    authors.bio
FROM
    library.authors authors
JOIN library.author_book author_book_join
ON authors.id = author_book_join.author_id
JOIN (
    SELECT books.id
    FROM library.books books
    JOIN library.reviews reviews ON books.id = reviews.book_id
    GROUP BY books.id
    HAVING AVG(reviews.rating) > 4
) five_star_books ON author_book_join.book_id = five_star_books.id;
```





Comparing SQL JOINS and aggregation

```
db.reviews.aggregate([
  { $group: {
      _id: "$bookId",
      averageRating: {
        $avg: "$rating",
      },
    },
  },
  { $match: { averageRating: {$gt: 4} } },
  { $lookup: {
      from: "authors",
      localField: "_id",
      foreignField: "books",
      as: "author",
    },
  },
  {$addFields: {
    bio: "$author.bio"
  }},
])
])
```

Structure of an Aggregation Pipeline



MongoDB Aggregation Syntax Structure

Keyword

Collection Command

```
db.reviews.aggregate()
```

```
[
```

```
{ $group: {  
    _id: "$bookId",  
    averageRating: { $avg: "$rating" } } },
```

options

```
{ $match: {  
    averageRating: { $gt: 4 } } },
```

options

```
{ $lookup: {  
    from: "books",  
    localField: "_id",  
    foreignField: "_id",  
    as: "bookDetails" } },
```

options

```
]
```

```
)
```



Stage

Stage

Stage

Agenda

- A brief overview of last week's session
- MongoDB Query API
 - CRUD
 - Aggregation Framework
- **Common Aggregation Stages**
 - 👉👉 **Simple pipelines**
 - 👉👉 Arrays, counting, and sorting
 - 👉👉 Joins & working with results
- ⭐ Skill Badge Challenge



\$match

```
db.books.aggregate([
  {
    $match: {
      pages: 100
    }
  }
]);
```

```
_id: "0393037959"
title : "Stalking the Shark: Pressure and Passion on the Pro Golf Tour"
pages : 100
```

```
_id: "0757300383"
title : "Chicken Soup for the Soul Christmas Treasury for Kids: A Story a Day f..."
pages : 100
```

```
_id: "0867193751"
title : "DORI STORIES"
pages : 100
```

```
_id: "088961184X"
title : "Bat Had Blue Eyes, The"
pages : 100
```

```
_id: "1880284545"
title : "ANOTHER FINE MESS"
pages : 100
```

```
_id: "1881943135"
title : "COMPLETE SLAVE"
pages : 100
```



\$project

```
db.books.aggregate([
  {
    $project: {
      title: 1,
      cover: 1
    }
  }
]);
```

```
_id: "0002005018"
title: "Clara Callan: A novel"
cover: "https://images.isbndb.com/covers/50/12/9780002005012.jpg"
```

```
_id: "0002244705"
title: "Charity"
cover: "https://images.isbndb.com/covers/47/01/9780002244701.jpg"
```

```
_id: "0002251000"
title: "NBA: The Official Fan's Guide"
cover: "https://images.isbndb.com/covers/10/06/9780002251006.jpg"
```

```
_id: "0002253402"
title: "Anita and me"
cover: "https://images.isbndb.com/covers/34/06/9780002253406.jpg"
```

```
_id: "0002554062"
title: "Shadow maker: The life of Gwendolyn MacEwen"
cover: "https://images.isbndb.com/covers/40/60/9780002554060.jpg"
```

```
_id: "0002554119"
title: "Labyrinth of Desire: Women, Passion, and Romantic Obsession"
cover: "https://images.isbndb.com/covers/41/14/9780002554114.jpg"
```



\$project

```
db.books.aggregate([
  {
    $project: {
      title: 1,
      cover: 1,
      _id: 0
    }
  }
]);
```

```
title: "Clara Callan: A novel"
cover: "https://images.isbndb.com/covers/50/12/9780002005012.jpg"
```

```
title: "Charity"
cover: "https://images.isbndb.com/covers/47/01/9780002244701.jpg"
```

```
title: "NBA: The Official Fan's Guide"
cover: "https://images.isbndb.com/covers/10/06/9780002251006.jpg"
```

```
title: "Anita and me"
cover: "https://images.isbndb.com/covers/34/06/9780002253406.jpg"
```

```
title: "Shadow maker: The life of Gwendolyn MacEwen"
cover: "https://images.isbndb.com/covers/40/60/9780002554060.jpg"
```

```
title: "Labyrinth of Desire: Women, Passion, and Romantic Obsession"
cover: "https://images.isbndb.com/covers/41/14/9780002554114.jpg"
```



\$limit

```
db.books.aggregate([
  {
    $limit: 3
  }
]);
```

```
_id: "0002005018"
title : "Clara Callan: A novel"
cover : "https://images.isbndb.com/covers/50/12/9780002005012.jpg"
```

```
_id: "0002244705"
title : "Charity"
cover : "https://images.isbndb.com/covers/47/01/9780002244701.jpg"
```

```
_id: "0002251000"
title : "NBA: The Official Fan's Guide"
cover : "https://images.isbndb.com/covers/10/06/9780002251006.jpg"
```



Quick Demo

mdb.link/instruqt-jedee

Exercise 1.

Simple pipelines



Environment:
mdb.link/instruqt-jedee

Lab:
mdb.link/agg-lab-simple

**Unsure? Need more
information?
Ask the instructors!**

The screenshot shows a web browser displaying the MongoDB Aggregation Pipeline Lab at mongodb-developer.github.io/aggregation-pipeline-lab/docs/what-is-aggregation/what-is-aggregation. The page title is "What is an Aggregation Pipeline?". On the left, a sidebar menu lists topics: Intro, Prerequisites, The Aggregation Pipeline, Simple Pipelines (which is highlighted with a green box), Using Arrays, Counting and Sorting, Lookups / Joins, Grouping Results, Modifying Results, Exporting Data, Lecture material, and Summary. The main content area contains a diagram illustrating an aggregation pipeline. It starts with an input array of documents (Orders) containing four items. The first item has cust_id: "A123", amount: 500, and status: "A". The second item has cust_id: "A232", amount: 250, and status: "A". The third item has cust_id: "B232", amount: 200, and status: "A". The fourth item has cust_id: "A123", amount: 300, and status: "W". An arrow labeled "\$match" points to the second item. Another arrow labeled "\$group" points to the result, which is a single document with id: "A123" and total: 750. Below the diagram, text explains that an aggregation pipeline is similar to unix commands connected using pipes and that it consists of one or more stages that process documents. It also notes that each stage performs an operation on the input documents.

Agenda

- A brief overview of last week's session
- MongoDB Query API
 - CRUD
 - Aggregation Framework
- **Common Aggregation Stages**
 - 👉 Simple pipelines
 - 👉 **Arrays, counting, and sorting**
 - 👉 Joins & working with results
- ⭐ Skill Badge Challenge

Array Matching





Arrays of Strings

Document A

```
title: "Dune",  
genres: [  
    "Sci-fi", 🚀  
    "Space", 🌟  
    "Epic" 😍  
]
```

Document B

```
title: "LOTR",  
genres: [  
    "Fantasy", 🗡️  
    "Fiction", 🦄  
    "Epic" 😍  
]
```

Document C

```
title: "4h workweek",  
genres: [  
    "Fiction", 🦄  
    "Fantasy", 🗡️  
    "Self-improvement"  
]
```

You may want to find documents with an array based on:

- **a single element**
- **multiple elements**
- **exact match of an array**

and so on...



Guess which documents would be returned?

Document A

title: "Dune",
genres: [
 "Sci-fi", 
 "Space", 
 "Epic" 
]

Document B

title: "LOTR",
genres: [
 "Fantasy", 
 "Fiction", 
 "Epic" 
]

Document C

title: "4h workweek",
genres: [
 "Fiction", 
 "Fantasy", 
 "Self-improvement"
]

```
db.books.aggregate([  
  $match: {  
    genres: ["Sci-fi", "Space", "Epic"]  
  }  
]);
```



Find books with exactly those genres

Document A

```
title: "Dune",  
genres: [  
  "Sci-fi", 🚀  
  "Space", 🌎  
  "Epic" 😍  
]
```

Document B

```
title: "LOTR",  
genres: [  
  "Fantasy", 🗡️  
  "Fiction", 🦄  
  "Epic" 😍  
]
```

Document C

```
title: "4h workweek",  
genres: [  
  "Fiction", 🦄  
  "Fantasy", 🗡️  
  "Self-improvement"  
]
```

```
db.books.aggregate([  
  {$match: {  
    genres: ["Sci-fi", "Space", "Epic"]  
  }  
}] );
```



Guess which documents would be returned?

Document A

```
title: "Dune",  
genres: [  
    "Sci-fi", 🚀  
    "Space", 🌟  
    "Epic" 😍  
]
```

Document B

```
title: "LOTR",  
genres: [  
    "Fantasy", 🗡️  
    "Fiction", 🦄  
    "Epic" 😍  
]
```

Document C

```
title: "4h workweek",  
genres: [  
    "Fiction", 🦄  
    "Fantasy", 🗡️  
    "Self-improvement"  
]
```

```
db.books.aggregate([  
    {$match: {  
        genres: 'Epic'  
    }  
}] );
```



Find books with one genre

Document A

```
title: "Dune",  
genres: [  
    "Sci-fi", 🚀  
    "Space", 🌟  
    "Epic" 😍  
]
```

Document B

```
title: "LOTR",  
genres: [  
    "Fantasy", 🗡️  
    "Fiction", 🦄  
    "Epic" 😍  
]
```

Document C

```
title: "4h workweek",  
genres: [  
    "Fiction", 🦄  
    "Fantasy", 🗡️  
    "Self-improvement"  
]
```

```
db.books.aggregate([  
    {$match: {  
        genres: 'Epic'  
    }  
}] );
```



Guess which documents would be returned?

Document A

title: "Dune",
genres: [
 "Sci-fi", 🚀
 "Space", 🌟
 "Epic" 😍
]

Document B

title: "LOTR",
genres: [
 "Fantasy", 🗡️
 "Fiction", 🦄
 "Epic" 😍
]

Document C

title: "4h workweek",
genres: [
 "Fiction", 🦄
 "Fantasy", 🗡️
 "Self-improvement"
]

```
db.books.aggregate([  
  {$match: {  
    genres: { $all: ["Fantasy", "Fiction"] }  
  }  
});
```



Find books with all two genres

Document A

```
title: "Dune",  
genres: [  
    "Sci-fi", 🚀  
    "Space", 🌟  
    "Epic" 😍  
]
```

Document B

```
title: "LOTR",  
genres: [  
    "Fantasy", 🗡️  
    "Fiction", 🦄  
    "Epic" 😍  
]
```

Document C

```
title: "4h workweek",  
genres: [  
    "Fiction", 🦄  
    "Fantasy", 🗡️  
    "Self-improvement"  
]
```

```
db.books.aggregate([  
    {  
        $match: {  
            genres: { $all: ["Fantasy", "Fiction"] }  
        }  
    }  
]);
```



Guess which documents would be returned?

Document A

```
title: "Dune",  
genres: [  
    "Sci-fi", 🚀  
    "Space", 🌟  
    "Epic" 😍  
]
```

Document B

```
title: "LOTR",  
genres: [  
    "Fantasy", 🗡️  
    "Fiction", 🦄  
    "Epic" 😍  
]
```

Document C

```
title: "4h workweek",  
genres: [  
    "Fiction", 🦄  
    "Fantasy", 🗡️  
    "Self-improvement"  
]
```

```
db.books.aggregate([  
    {$match: {  
        genres: { $in: ["Fantasy", "Epic"] }  
    }  
}] );
```



Find books with either genre

Document A

```
title: "Dune",  
genres: [  
  "Sci-fi", 🚀  
  "Space", 🌟  
  "Epic" 😍  
]
```

Document B

```
title: "LOTR",  
genres: [  
  "Fantasy", 🗡️  
  "Fiction", 🦄  
  "Epic" 😍  
]
```

Document C

```
title: "4h workweek",  
genres: [  
  "Fiction", 🦄  
  "Fantasy", 🗡️  
  "Self-improvement"  
]
```

```
db.books.aggregate([  
  {$match: {  
    genres: { $in: ["Fantasy", "Epic"] }  
  }  
}] );
```

Search Inside Objects

Aggregation
Stages



```
attr: [
  {
    key: 'edition',
    value: '1st'
  },
  {
    key: 'msrp',
    value: 6.99
  },
  {
    key: 'isbn',
    value: '0765342502'
  },
  ...
]
```

The Attribute Pattern in action!



```
title: "Dune",  
attr: [  
  {  
    key: "edition",  
    value: "1st"  
  },  
  ...  
]
```

```
title: "LOTR",  
attr: [  
  {  
    key: "edition",  
    value: "10th"  
  },  
  ...  
]
```

```
title: "4h workweek",  
attr: [  
  {  
    key: "msrp",  
    value: "10"  
  },  
  ...  
]
```

Find all books that have an "edition" attribute



```
title: "Dune",  
attr: [  
  {  
    key: "edition",  
    value: "1st"  
  },  
  ...  
]
```

```
title: "LOTR",  
attr: [  
  {  
    key: "edition",  
    value: "10th"  
  },  
  ...  
]
```

```
title: "4h workweek",  
attr: [  
  {  
    key: "msrp",  
    value: "10"  
  },  
  ...  
]
```

```
db.books.aggregate([  
  $match: {  
    "attributes.key": "edition"  
  }  
]);
```

Find all books that are 1st edition (simplified)



```
title: "Dune",
attr: [
  {
    key: "edition",
    value: "1st"
  },
  ...
]
```

```
title: "LOTR",
attr: [
  {
    key: "edition",
    value: "10th"
  },
  ...
]
```

```
title: "4h workweek",
attr: [
  {
    key: "msrp",
    value: "10"
  },
  ...
]
```

```
db.books.aggregate([
  $match: {
    "attributes.keyattributes.value
```



Aggregation Stages

This is OK,
but we get
all
attributes!

```
{  
  "_id": "0002005018",  
  "title": "Clara Callan: A novel",  
  "attributes": [  
    {  
      "key": "edition",  
      "value": "1st"  
    },  
    {  
      "key": "dimensions",  
      "value": "Height: 11.11 Inches, Length: 6.11 Inches,  
Weight: 1 Pounds, Width: 1.11 Inches"  
    },  
    {  
      "key": "isbn13",  
      "value": "9780002005012"  
    },  
    {  
      "key": "msrp",  
      "value": "0.00"  
    },  
    {  
      "key": "isbn",  
      "value": "0002005018"  
    },  
    {  
      "key": "isbn10",  
      "value": "0002005018"  
    }  
  ]  
}
```

```
title: "1984",
attri: [
{
  "key": "msrp",
  "value": "10",
},
{
  "key": "edition",
  "value": "1st",
},
{
  "key": "isbn",
  "value": "82828722",
},
]
```

\$unwind

\$unwind

```
title: "1984",
attr: {
  "key": "msrp",
  "value": "10",
},
```

```
title: "1984",
attr: {
  "key": "edition",
  "value": "first",
},
```

```
title: "1984",
attr: {
  "key": "isbn",
  "value": "82828722",
},
```



Aggregation Stages

We'll get only
the attribute
we're looking
for!

```
db.books.aggregate([  
  { $unwind : "$attr" },  
  { $match:  
    {"attr.key": "edition",  
     "attr.value": "1st"}  
  }  
]);
```

Counting and Sorting





\$count

```
db.books.aggregate([
  {
    $count: "numberOfBooks"
  }
]);
```



\$sorting

```
db.books.aggregate([
  {
    $sort: {
      pages: -1
    }
  }
]);
```

```
_id: "0393970876"
title : "The Norton Shakespeare: Based on the Oxford Edition"
pages : 3420
```

```
_id: "0060655267"
title : "HarperCollins Study Bible: New Revised Standard Version (with the Apoc..."
pages : 2368
```

```
_id: "0443045607"
title : "Gray's Anatomy: The Anatomical Basis of Medicine and Surgery"
pages : 2092
```

```
_id: "0192835254"
title : "The Bible: Authorized King James Version with Apocrypha"
pages : 1824
```

```
_id: "0198600461"
title : "The Oxford English Reference Dictionary"
pages : 1804
```

```
_id: "0312280874"
title : "Microsoft Encarta College Dictionary: The First Dictionary For The Int..."
pages : 1728
```

Exercise 2.

Counting and sorting



Environment:
mdb.link/instruqt-jedee

Lab:
mdb.link/agg-lab-grouping

**Unsure? Need more
information?
Ask the instructors!**

What is an Aggregation Pipeline?

```
graph LR; Input[Orders] --> Match{$match}; Match --> Group{$group}; Group --> Output[{"id": "A123", "total": 750}, {"id": "B232", "total": 200}];
```

An aggregation pipeline is similar to unix commands connected using pipes. We can construct modular, composable processing pipelines.

An aggregation pipeline consists of one or more stages that process documents:

Each stage performs an operation on the input documents. For example, a stage can filter documents, group documents, and calculate values.

Agenda

- A brief overview of last week's session
- MongoDB Query API
 - CRUD
 - Aggregation Framework
- **Common Aggregation Stages**
 - 👉 Simple pipelines
 - 👉 Arrays, counting, and sorting
 - 👉 **Joins & working with results**
- ⭐ Skill Badge Challenge

Joins





\$lookup

```
db.authors.aggregate([
  {
    $lookup: {
      from: "books",
      localField: "books",
      foreignField: "_id",
      as: "booksWritten"
    }
  }
])
```

```
_id: ObjectId('64cc2db4830ba29148da4c3b')
name : "Richard Bruce Wright"
▼ booksWritten : Array (1)
  ▼ 0: Object
    title : "Clara Callan: A novel"
```

```
_id: ObjectId('64cc2db4830ba29148da4c3f')
name : "Amy Tan"
▼ booksWritten : Array (3)
  ▼ 0: Object
    title : "The Joy Luck Club"
  ▼ 1: Object
    title : "La Hija Del Curandero / the Bonesetter's Daughter (Spanish Edition)"
  ▼ 2: Object
    title : "The Hundred Secret Senses"
```

```
_id: ObjectId('64cc2db4830ba29148da4c40')
name : "Scott Turow"
▼ booksWritten : Array (2)
  ▼ 0: Object
    title : "Personal Injuries"
  ▼ 1: Object
    title : "Pleading Guilty (G K Hall Large Print Book Series)"
```

Grouping Data





\$group

```
db.books.aggregate([
  {
    $group: {
      _id: "$year",
      totalPages: {
        $sum: "$pages"
      }
    }
  }
])
```

```
_id: 1959
totalPages : 256
```

```
_id: 1971
totalPages : 5177
```

```
_id: 1993
totalPages : 66571
```

```
_id: 1977
totalPages : 7507
```

```
_id: 2006
totalPages : 3869
```

```
_id: 1990
totalPages : 45080
```

Modifying Results





\$set

```
db.books.aggregate([
  {
    $set: {
      readingTimeHours: {
        $divide: [
          { $multiply: ["$pages", 2] },
          60
        ]
      }
    }
  }
])
```

```
_id: "0002005018"
title: "Clara Callan: A novel"
pages: 414
readingTimeHours: 13.8
```

```
_id: "0002244705"
title: "Charity"
pages: 320
readingTimeHours: 10.666666666666666
```

```
_id: "0002251000"
title: "NBA: The Official Fan's Guide"
pages: 160
readingTimeHours: 5.333333333333333
```

```
_id: "0002253402"
title: "Anita and me"
pages: 328
readingTimeHours: 10.933333333333334
```

```
_id: "0002554062"
title: "Shadow maker: The life of Gwendolyn MacEwen"
pages: 416
readingTimeHours: 13.866666666666667
```

Exporting Data





\$out

```
db.books.aggregate([
  {
    $match: {
      pages: 100
    }
  },
  {
    $out: {
      db: "reporting",
      coll: "authors"
    }
  }
])
```

Exercise 3.

Grouping results & modifying results



Environment:
mdb.link/instruqt-jedee

Lab:
mdb.link/agg-lab-grouping

**Unsure? Need more information?
Ask the instructors!**

The screenshot shows a web browser displaying the MongoDB Aggregation Pipeline Lab at mongodb-developer.github.io/aggregation-pipeline-lab/docs/what-is-aggregation/what-is-aggregation. The page title is "What is an Aggregation Pipeline?" and the sub-page title is "What is an Aggregation Pipeline?".

The left sidebar lists navigation options: Intro, Prerequisites, The Aggregation Pipeline, Simple Pipelines, Using Arrays, Counting and Sorting, Lookups / Joins, Grouping Results (highlighted with a green box), Modifying Results (highlighted with a green box), Exporting Data, Lecture material, and Summary.

The main content area shows a diagram of an aggregation pipeline. It starts with an input collection named "Orders" containing five documents. The documents are:

```
[{"cust_id": "A123", "amount": 500, "status": "A"}, {"cust_id": "A123", "amount": 250, "status": "A"}, {"cust_id": "B212", "amount": 200, "status": "A"}, {"cust_id": "B212", "amount": 300, "status": "B"}]
```

The pipeline consists of two stages: "\$match" and "\$group".

- \$match:** This stage filters the input documents. The output contains only the first two documents from the input.
- \$group:** This stage groups the documents by their "cust_id" and calculates the total amount. The output is a single document with two entries:

```
[{"id": "A123", "total": 750}, {"id": "B212", "total": 200}]
```

Below the diagram, there is explanatory text:

An aggregation pipeline is similar to unix commands connected using pipes. We can construct modular, composable processing pipelines.

An aggregation pipeline consists of one or more stages that process documents:

Each stage performs an operation on the input documents. For example, a stage can filter documents, group documents, and calculate values.



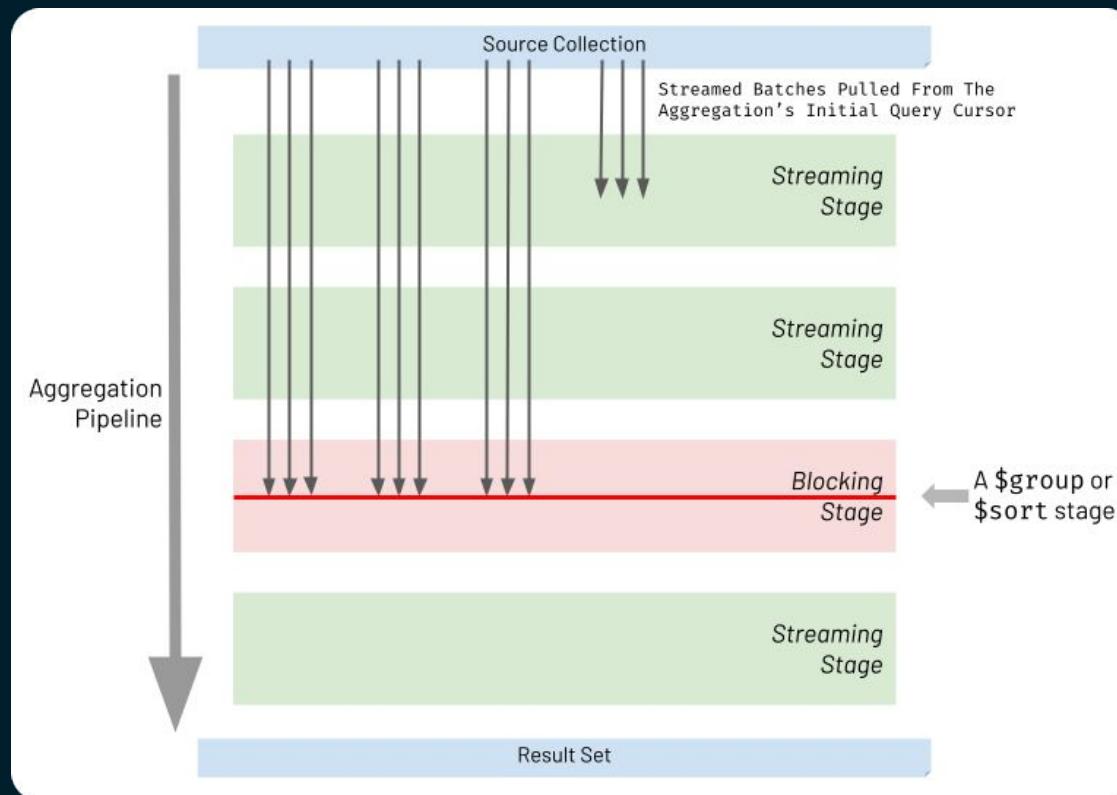
Bonus Question:

Using aggregation on *authors* collection:

For each author, find the book he/she has written with the most number of pages.



Streaming VS Blocking Stages





Best Practices

Avoid using aggregation only for transformation

Avoid block stages

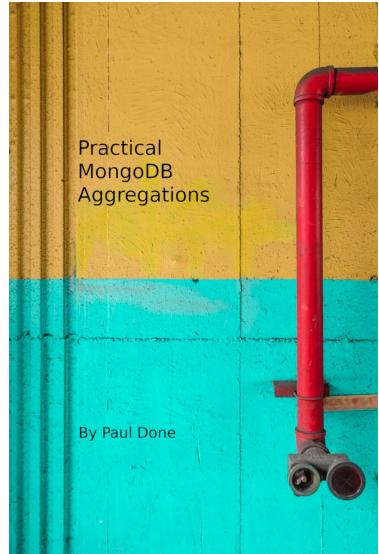
Use explain plan

Embrace compositability



E-BOOK

Practical MongoDB Aggregations



mdb.link/agg-ebook



Compass. The GUI for MongoDB.

<https://mdb.link/compass>



Session survey

Agenda

- A brief overview of last week's session
- MongoDB Query API
 - CRUD
 - Aggregation Framework
- Common Aggregation Stages
 - 👉👉 Simple pipelines
 - 👉👉 Arrays, counting, and sorting
 - 👉👉 Joins & working with results
- ⭐ **Skill Badge Challenge**



SKILL BADGE knowledge check - 10 MINUTES

Fundamentals of Data Transformation

MongoDB Skill

Fundamentals of Data
Transformation



mdb.link/apac-workshop/badge-3

Questions?



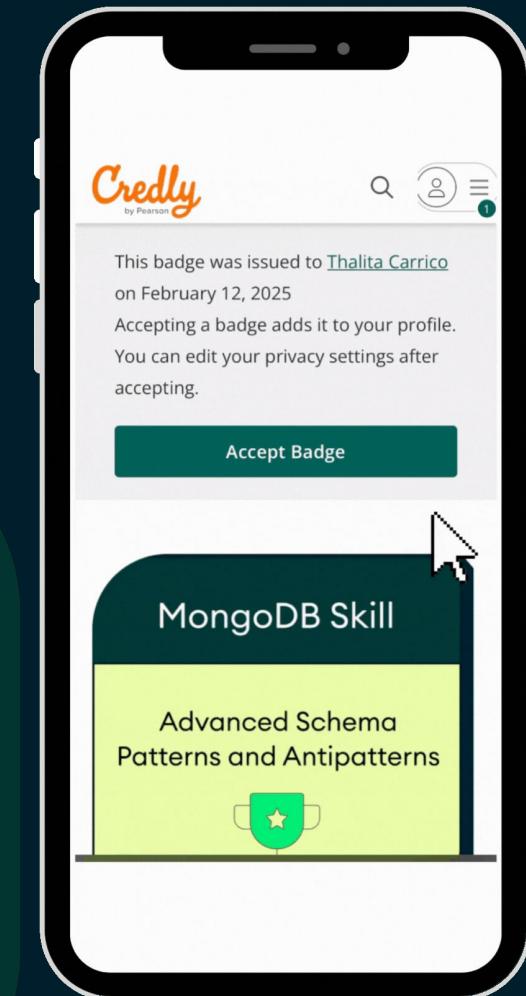
Ask your neighbor



Raise your hand

Share Your Badge

Use #MDBDayAPAC





Thank you!



Teo Wen Jie
Senior Developer Advocate
[in/wjteo](https://www.linkedin.com/in/wjteo)



Lets connect!