MongoDB: Advanced Queries (Aggregation Framework)

Prepared by Jeong-Hun Kim

Table of Content

In the last lecture

- Query embedded documents
- Query an array
- Query an array of embedded documents
- Query for null and missing fields
- Regular Expressions

In the last lecture

- Big data process

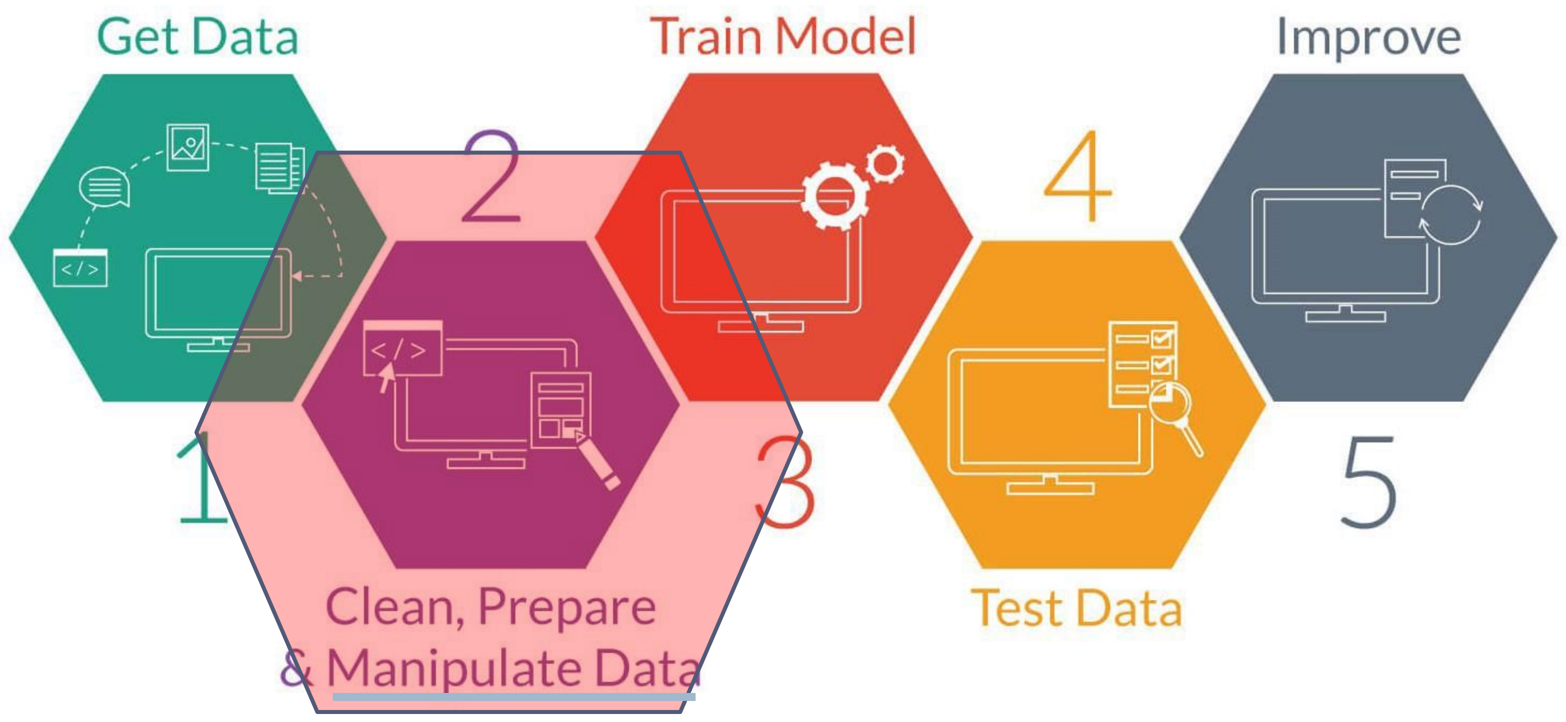


Table of Content

Aggregation Framework

- Traditional SQL queries
  - Fixed order (not flexible) and thus no optimization can be achieved

```
SELECT        [DISTINCT] 애트리뷰트(들)               (1) ⎫
FROM          릴레이션(들)                            (2) ⎭ 필수
[WHERE        조건                                    (3) ⎫
                     [중첩 질의]]                     (4) ⎪
[GROUP BY     애트리뷰트(들)]                         (5) ⎬ 선택
[HAVING       조건]                                   (6) ⎪
[ORDER BY     애트리뷰트(들) [ASC|DESC]];             (7) ⎭
```

[그림 4.9] SELECT문의 형식

Aggregation Framework

- MongoDB aggregation framework
  - Operation pipeline
    - Obtains results in step-by-step
    - Flexible order

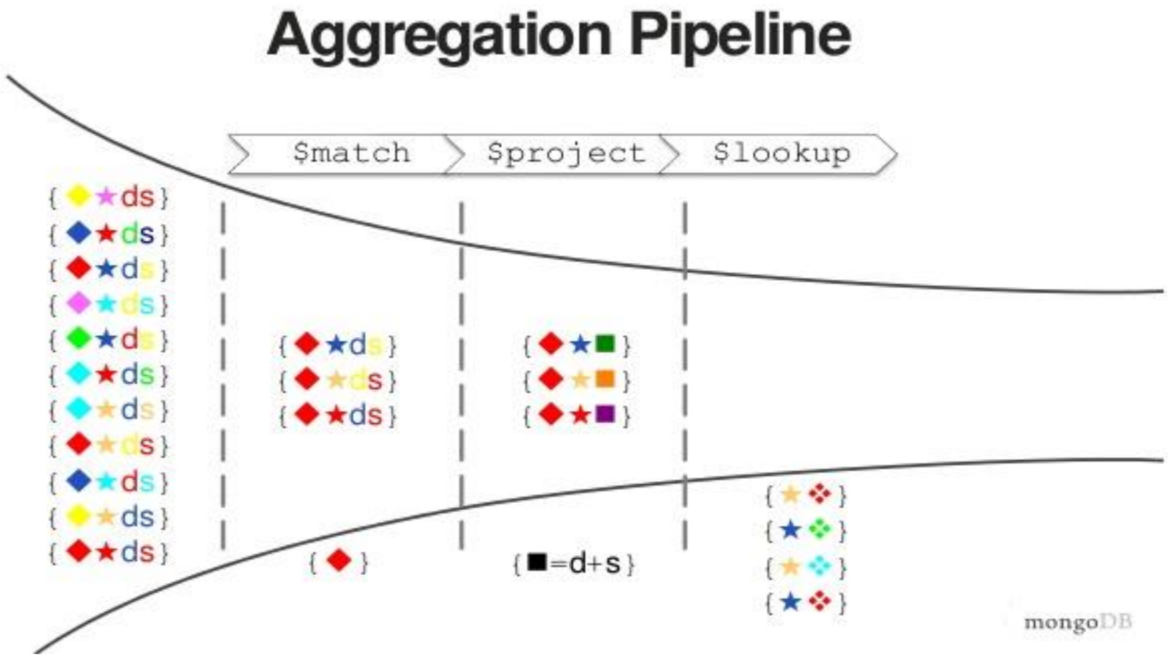
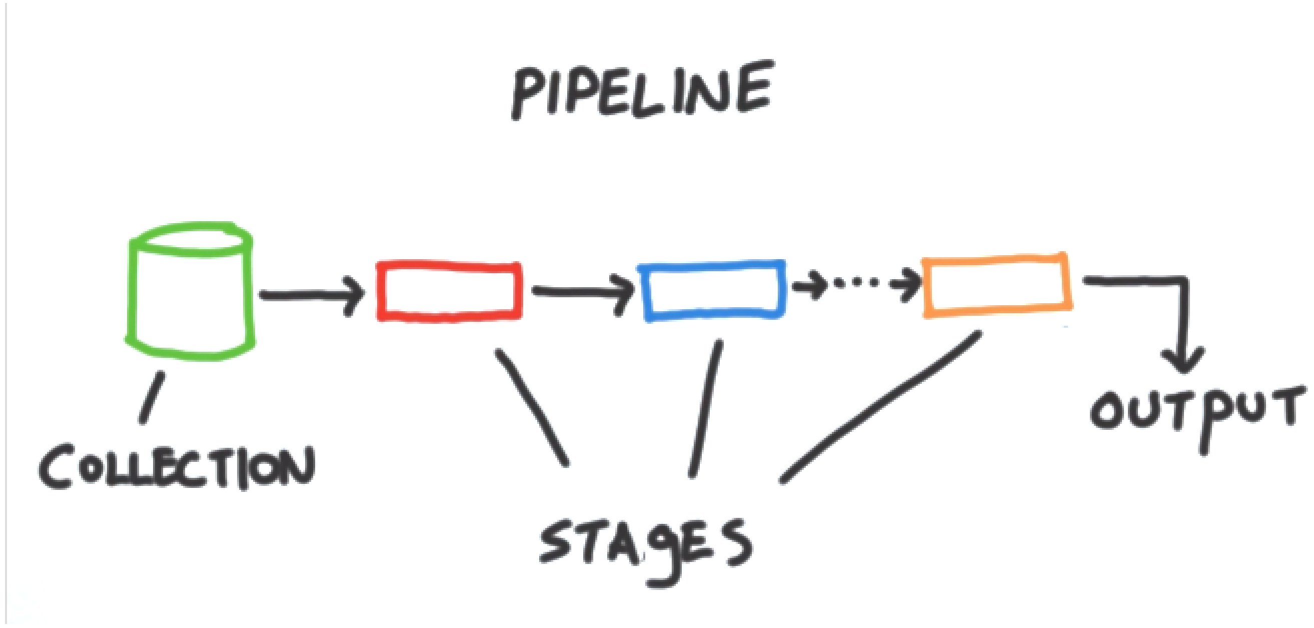


Table of Content

Aggregation Pipeline

- Documents enter a multi-stage pipeline that transforms the documents into an aggregated result

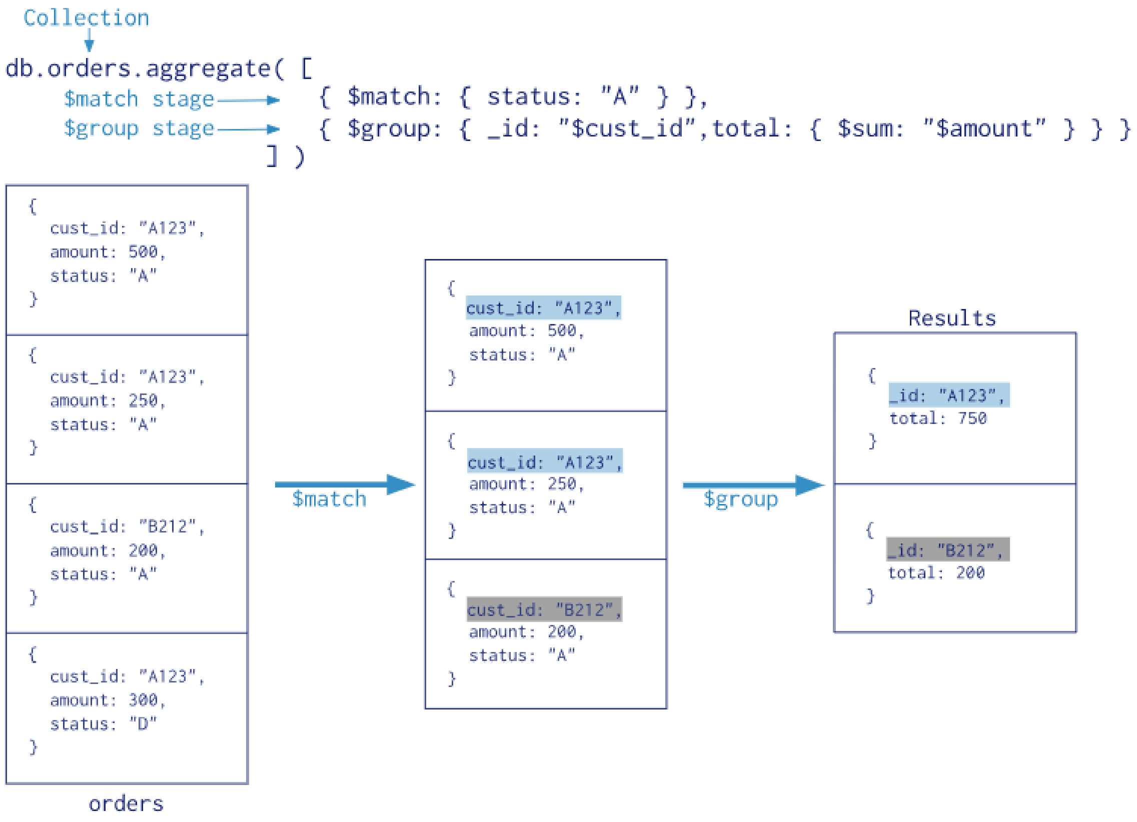


Aggregation Framework

- Aggregate() function is used for creating pipelines
  - *db.collection.aggregate(pipeline, options)*
  - Pipeline is an array that can have multiple stages
    - Each stage here is a document
  - Pipeline is a sequence of data aggregation operations or stages
  - Optional
    - Additional options that aggregate() passes to the aggregate command

Aggregation Pipeline

- Documents enter a multi-stage pipeline that transforms the documents into an aggregated result



```
db.orders.aggregate( [
      { $match: { status: "A" } },
      { $group: { _id: "$cust_id",total: { $sum: "$amount" } } }
   ] )
```

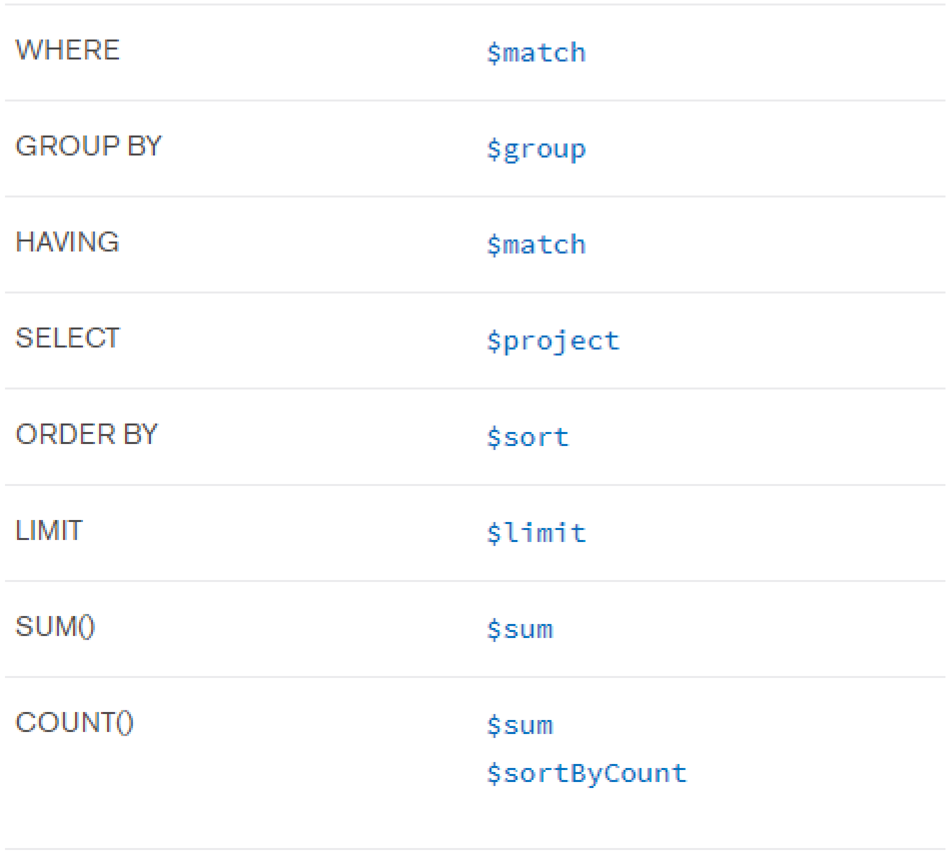
Collection
$match stage
$group stage
{ cust_id: "A123", amount: 500, status: "A" }
{ cust_id: "A123", amount: 250, status: "A" }
{ cust_id: "B212", amount: 200, status: "A" }
{ cust_id: "A123", amount: 300, status: "D" }
orders
$match
$group
Results
{ _id: "A123", total: 750 }
{ _id: "B212", total: 200 }

Aggregation Pipeline

- Pipeline stages
  - $match
    - Filter documents
  - $project
    - Reshape documents
  - $group
    - Summarize documents
  - $sort
    - Order documents
  - $limit and $skip
    - Paginate document
  - $unwind
    - Create documents from array elements

Aggregation Framework

- SQL to Aggregation Mapping Chart

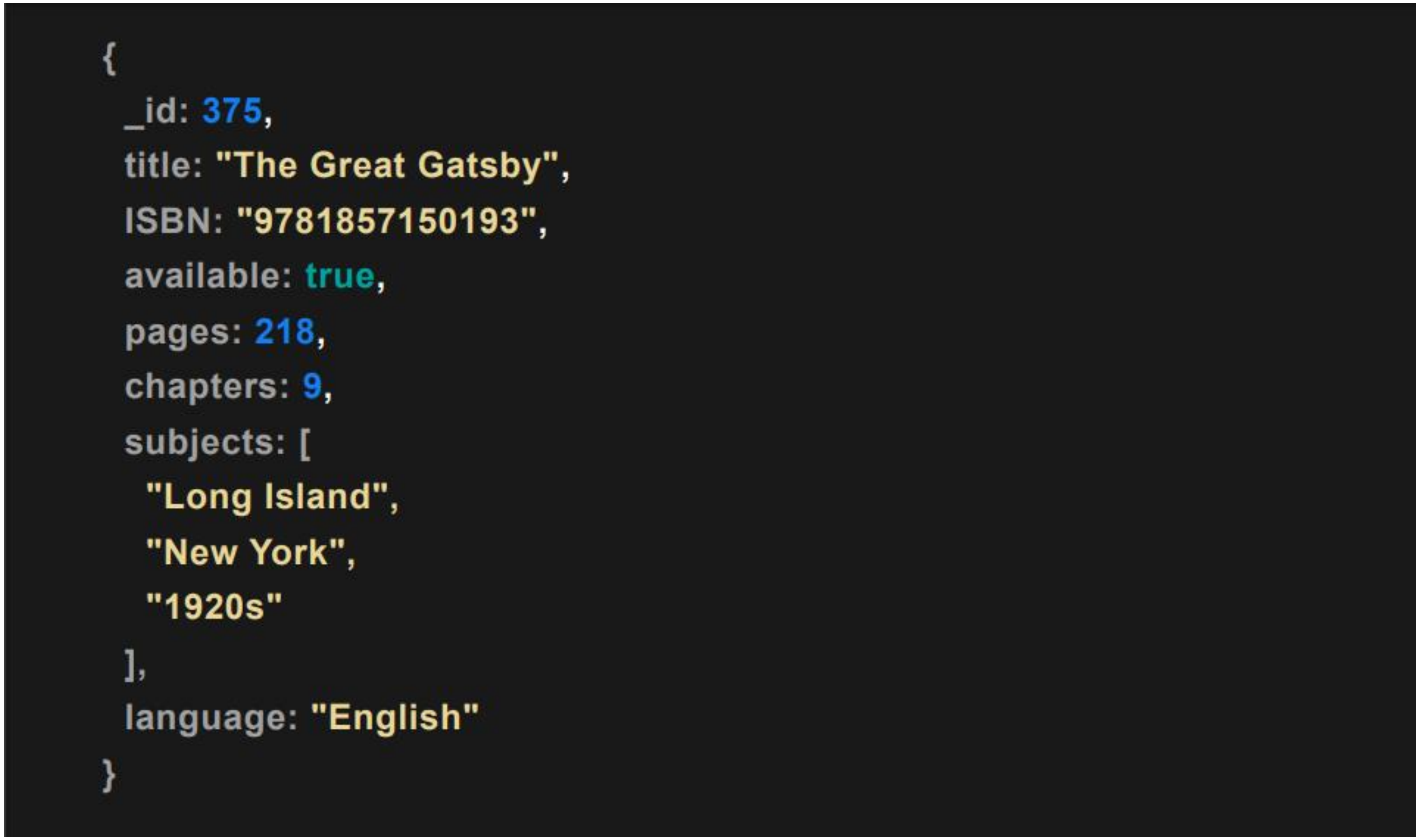


WHERE	`$match`
GROUP BY	`$group`
HAVING	`$match`
SELECT	`$project`
ORDER BY	`$sort`
LIMIT	`$limit`
SUM()	`$sum`
COUNT()	`$sum` `$sortByCount`

Table of Content

Aggregation Pipeline Stages

- Example dataset



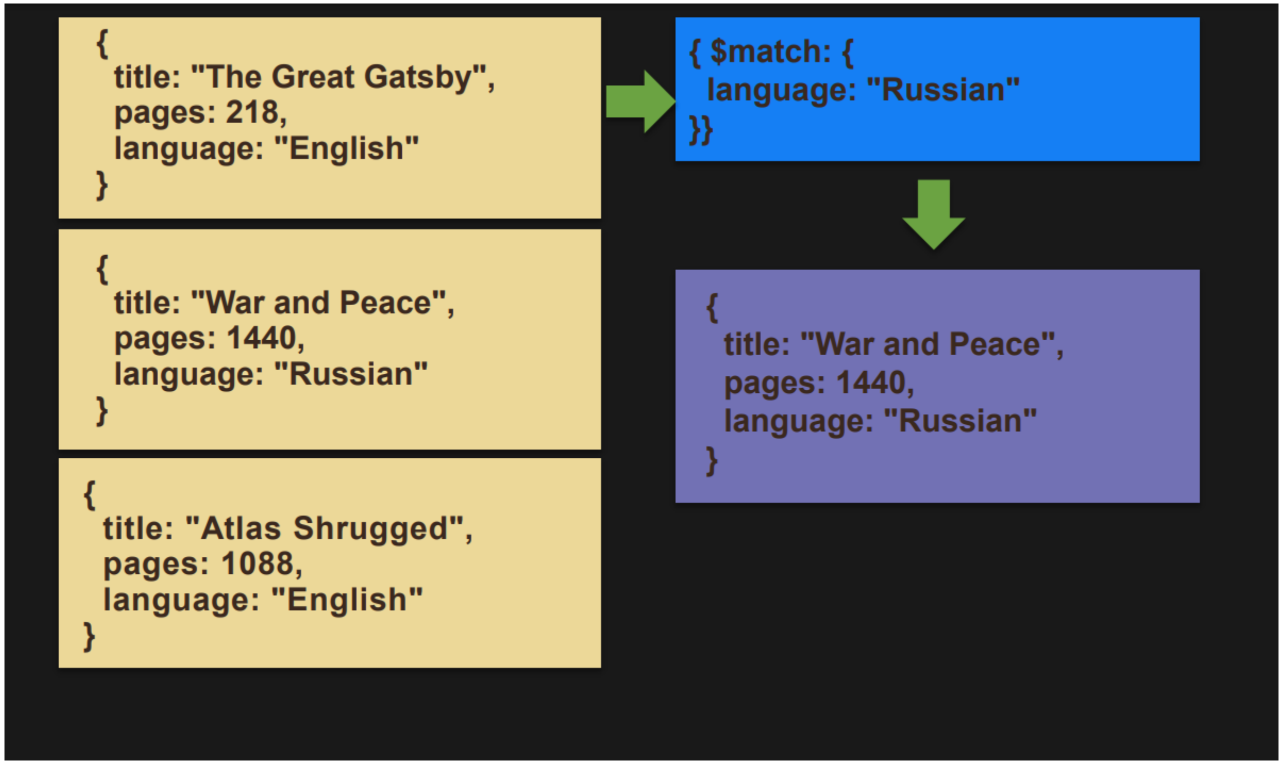
```
{
  _id: 375,
  title: "The Great Gatsby",
  ISBN: "9781857150193",
  available: true,
  pages: 218,
  chapters: 9,
  subjects: [
    "Long Island",
    "New York",
    "1920s"
  ],
  language: "English"
}
```

Aggregation Pipeline Stages

- $match
  - Filters the documents to pass only the documents that match the specified condition(s) to the next pipeline stage
  - The $match stage has the following prototype form
    - { *$match:* { <*query*> } }
  - Place the $match as early in the aggregation pipeline as possible
    - limits the total number of documents in the aggregation pipeline
    - minimizes the amount of processing down the pipe

Aggregation Pipeline Stages

- $match
  - Matching field values



```
{
  title: "The Great Gatsby",
  pages: 218,
  language: "English"
}
```

```
{
  title: "War and Peace",
  pages: 1440,
  language: "Russian"
}
```

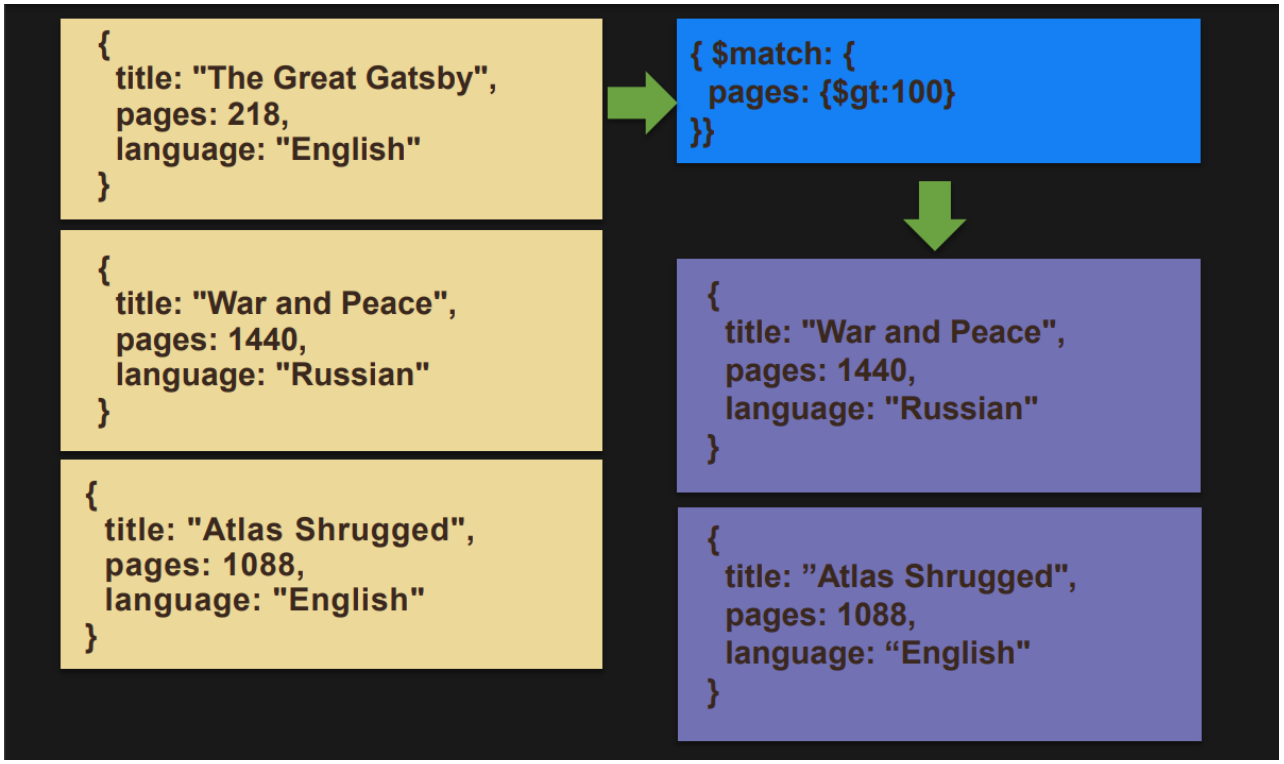
```
{
  title: "Atlas Shrugged",
  pages: 1088,
  language: "English"
}
```

```
{ $match: {
  language: "Russian"
}}
```

```
{
  title: "War and Peace",
  pages: 1440,
  language: "Russian"
}
```

Aggregation Pipeline Stages

- $match
  - Matching with query operators



```
{
  title: "The Great Gatsby",
  pages: 218,
  language: "English"
}
```

```
{
  title: "War and Peace",
  pages: 1440,
  language: "Russian"
}
```

```
{
  title: "Atlas Shrugged",
  pages: 1088,
  language: "English"
}
```

```
{ $match: {
  pages: {$gt:100}
}}
```

```
{
  title: "War and Peace",
  pages: 1440,
  language: "Russian"
}
```

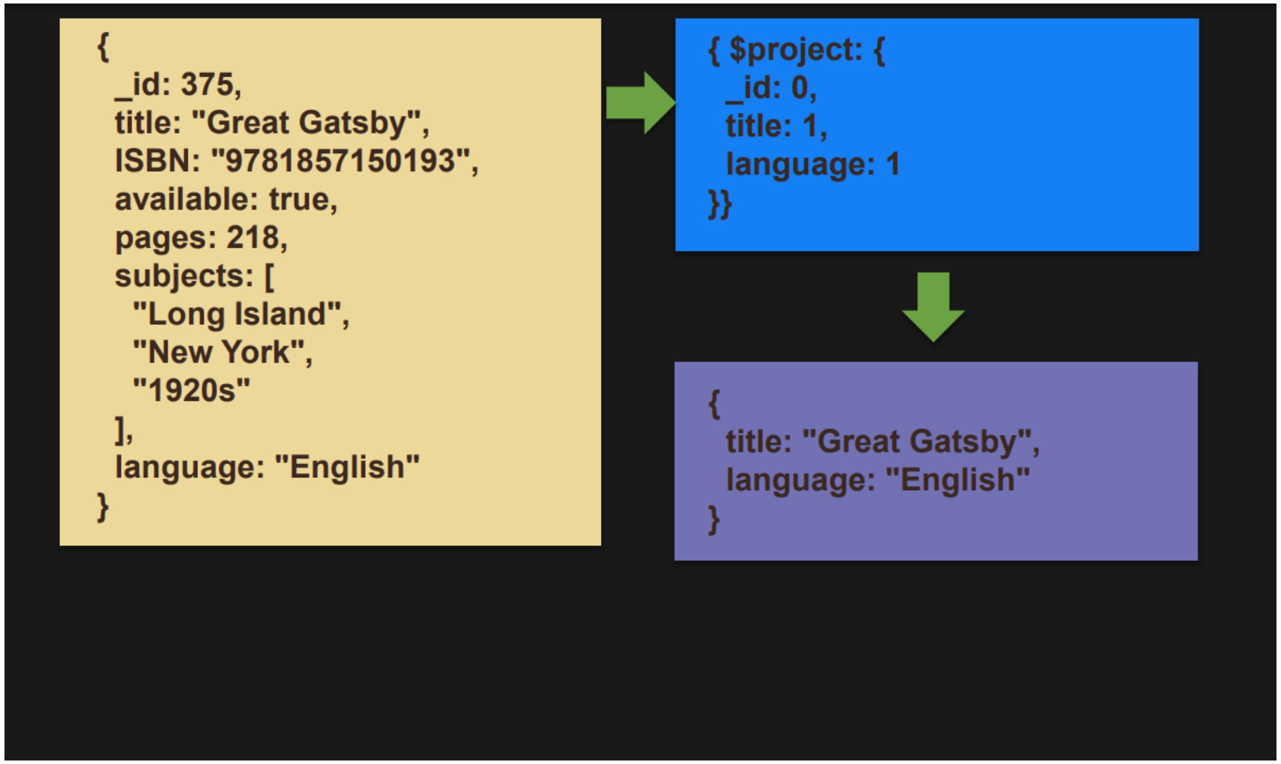
```
{
  title: ”Atlas Shrugged",
  pages: 1088,
  language: “English"
}
```

Aggregation Pipeline Stages

- $project
  - Passes along the documents with the requested fields to the next stage in the pipeline
  - The specified fields can be existing fields from the input documents or newly computed fields
    - Include, exclude or rename fields
  - The $project stage has the following prototype form
    - { *$project:* { *<specification(s)>* } }
  - You can use the $project as projection operator

Aggregation Pipeline Stages

- $project
  - Including and Excluding Fields



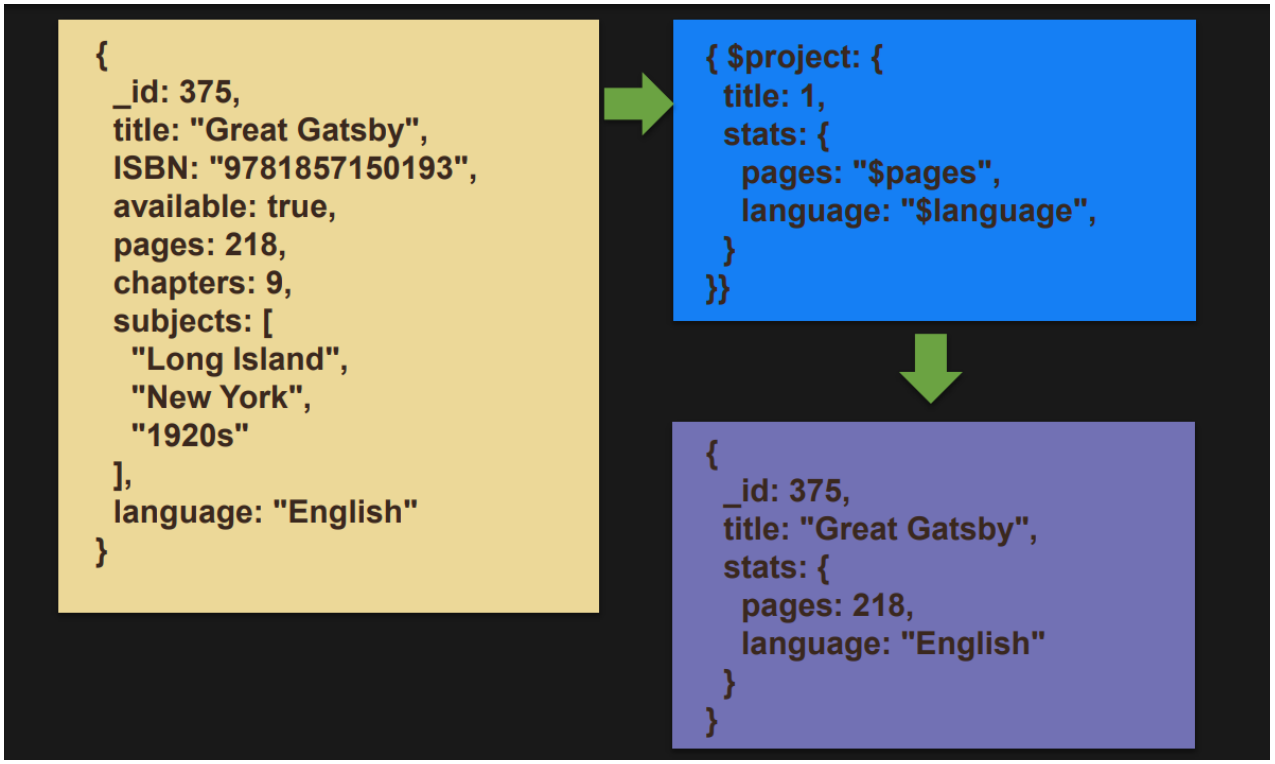
```
{
  _id: 375,
  title: "Great Gatsby",
  ISBN: "9781857150193",
  available: true,
  pages: 218,
  subjects: [
    "Long Island",
    "New York",
    "1920s"
  ],
  language: "English"
}
```

```
{ $project: {
  _id: 0,
  title: 1,
  language: 1
}}
```

```
{
  title: "Great Gatsby",
  language: "English"
}
```

Aggregation Pipeline Stages

- $project
  - Creating Sub-Document Fields



```
{
  _id: 375,
  title: "Great Gatsby",
  ISBN: "9781857150193",
  available: true,
  pages: 218,
  chapters: 9,
  subjects: [
    "Long Island",
    "New York",
    "1920s"
  ],
  language: "English"
}
```

```
{ $project: {
  title: 1,
  stats: {
    pages: "$pages",
    language: "$language",
  }
}}
```

```
{
  _id: 375,
  title: "Great Gatsby",
  stats: {
    pages: 218,
    language: "English"
  }
}
```

Aggregation Pipeline Stages

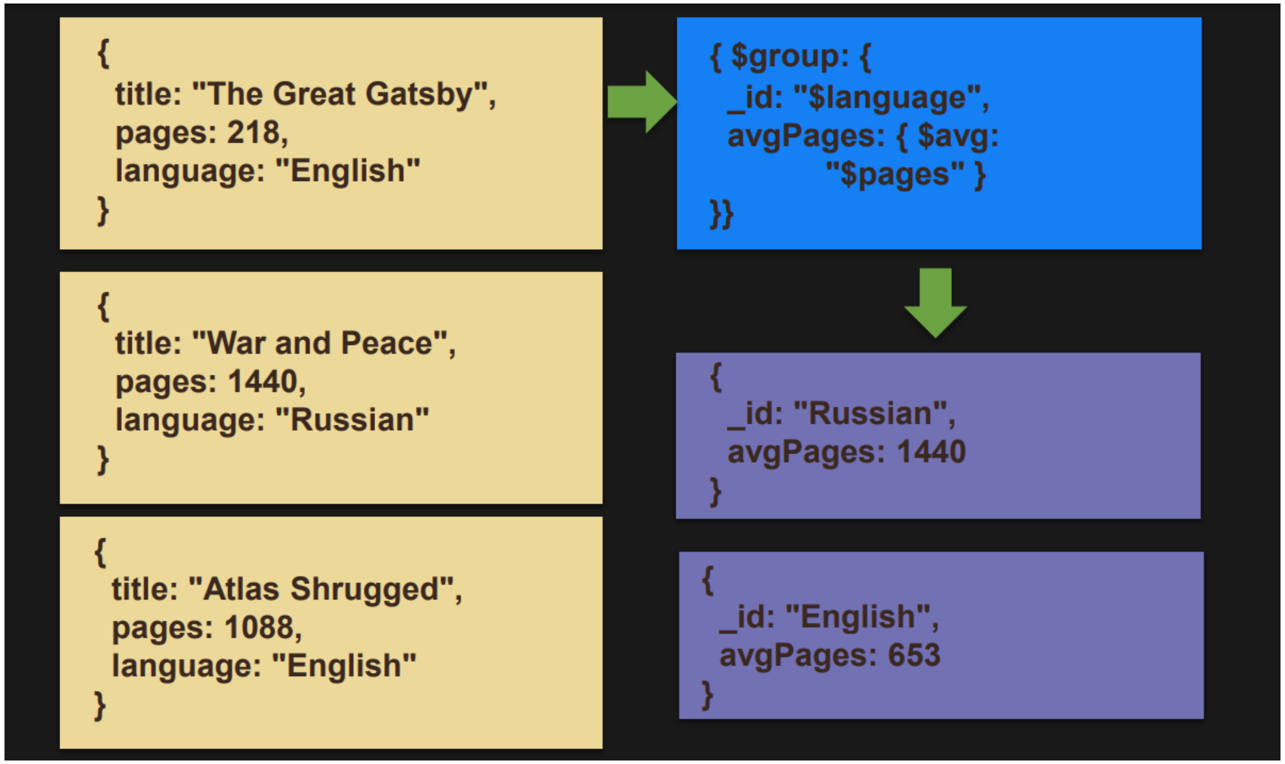
- $group
  - Group documents by value

  - The $group stage has the following prototype form
    - *{ $group: { _id: <expression>, <field1>: { <accumulator1> : <expression1> }, ... } }*
      - _id field is mandatory

  - Accumulators
    - $max, $min, $avg, $sum
    - $addToSet, $push
    - $first, $last

Aggregation Pipeline Stages

- $group
  - Calculating average



```
{
  title: "The Great Gatsby",
  pages: 218,
  language: "English"
}
```

```
{
  title: "War and Peace",
  pages: 1440,
  language: "Russian"
}
```

```
{
  title: "Atlas Shrugged",
  pages: 1088,
  language: "English"
}
```

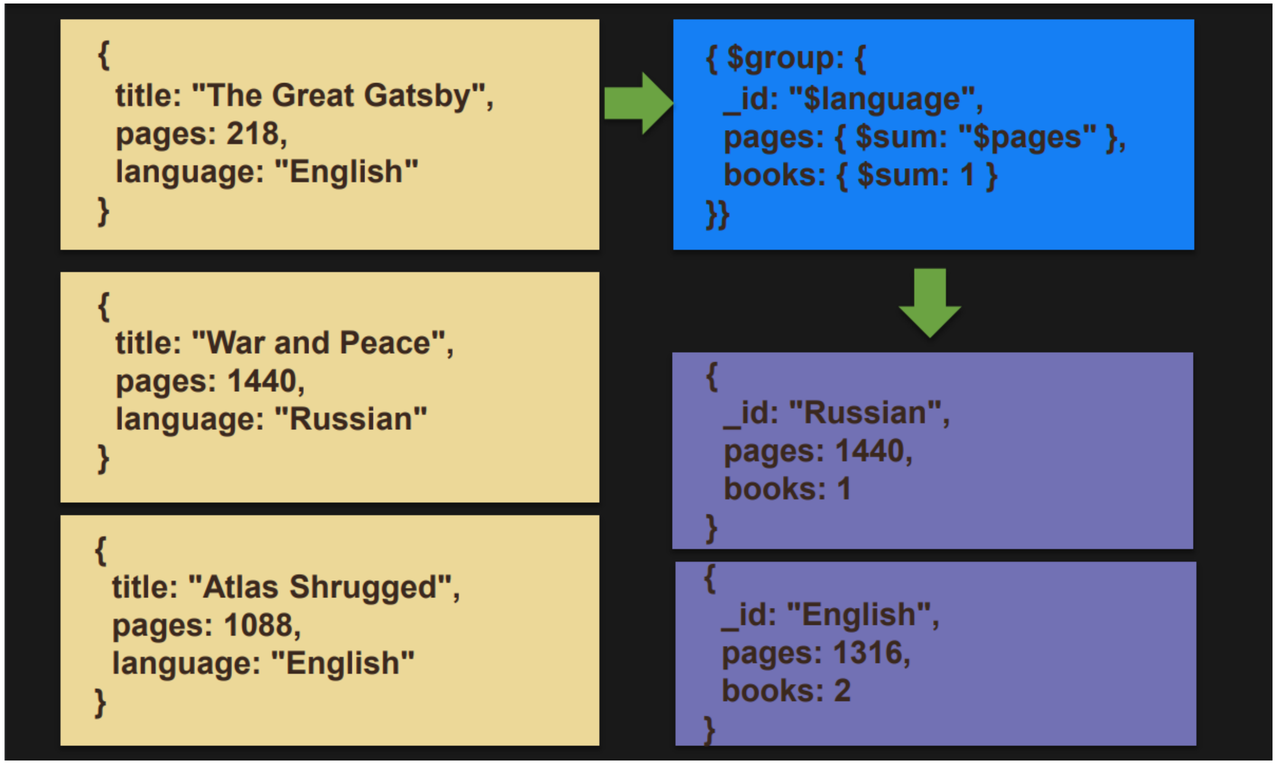
```
{ $group: {
  _id: "$language",
  avgPages: { $avg:
           "$pages" }
}}
```

```
{
  _id: "Russian",
  avgPages: 1440
}
```

```
{
  _id: "English",
  avgPages: 653
}
```

Aggregation Pipeline Stages

- $group
  - Summing Fields and Counting



```
{
  title: "The Great Gatsby",
  pages: 218,
  language: "English"
}
```

```
{
  title: "War and Peace",
  pages: 1440,
  language: "Russian"
}
```

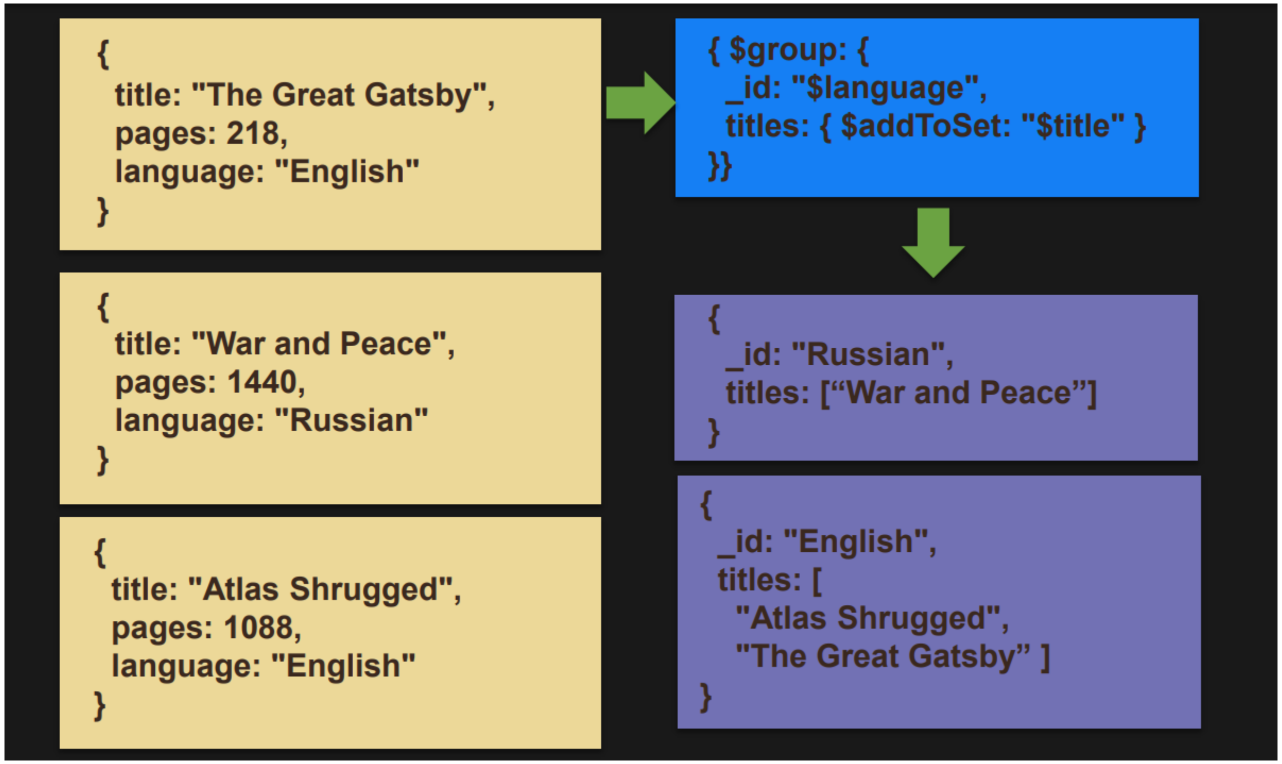
```
{
  title: "Atlas Shrugged",
  pages: 1088,
  language: "English"
}
```

```
{ $group: {
  _id: "$language",
  pages: { $sum: "$pages" },
  books: { $sum: 1 }
}}
```

```
{
  _id: "Russian",
  pages: 1440,
  books: 1
}
{
  _id: "English",
  pages: 1316,
  books: 2
}
```

Aggregation Pipeline Stages

- $group
  - Collecting Distinct Values



```
{
  title: "The Great Gatsby",
  pages: 218,
  language: "English"
}
```

```
{
  title: "War and Peace",
  pages: 1440,
  language: "Russian"
}
```

```
{
  title: "Atlas Shrugged",
  pages: 1088,
  language: "English"
}
```

```
{ $group: {
  _id: "$language",
  titles: { $addToSet: "$title" }
}}
```

```
{
  _id: "Russian",
  titles: ["War and Peace"]
}
```

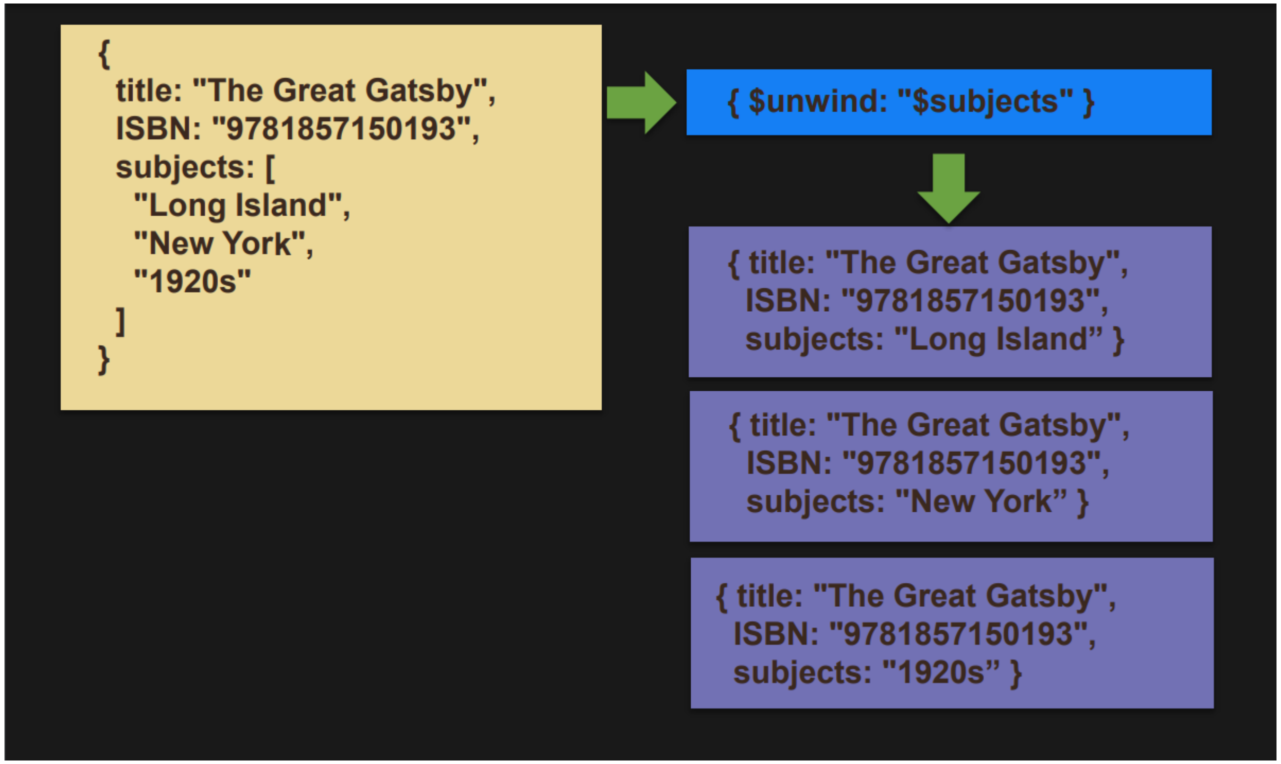
```
{
  _id: "English",
  titles: [
    "Atlas Shrugged",
    "The Great Gatsby" ]
}
```

Aggregation Pipeline Stages

- $unwind
  - Deconstructs an array field from the input documents to output a document for each element
    - Array replaced by element value
    - Missing/empty fields → no output
  - The $unwind stage has the following syntax
    - { $unwind: <field path> }

Aggregation Pipeline Stages

- $unwind
  - Deconstructing (unwind) the array



```
{
  title: "The Great Gatsby",
  ISBN: "9781857150193",
  subjects: [
    "Long Island",
    "New York",
    "1920s"
  ]
}
```

```
{ $unwind: "$subjects" }
```

```
{ title: "The Great Gatsby",
  ISBN: "9781857150193",
  subjects: "Long Island" }
```

```
{ title: "The Great Gatsby",
  ISBN: "9781857150193",
  subjects: "New York" }
```

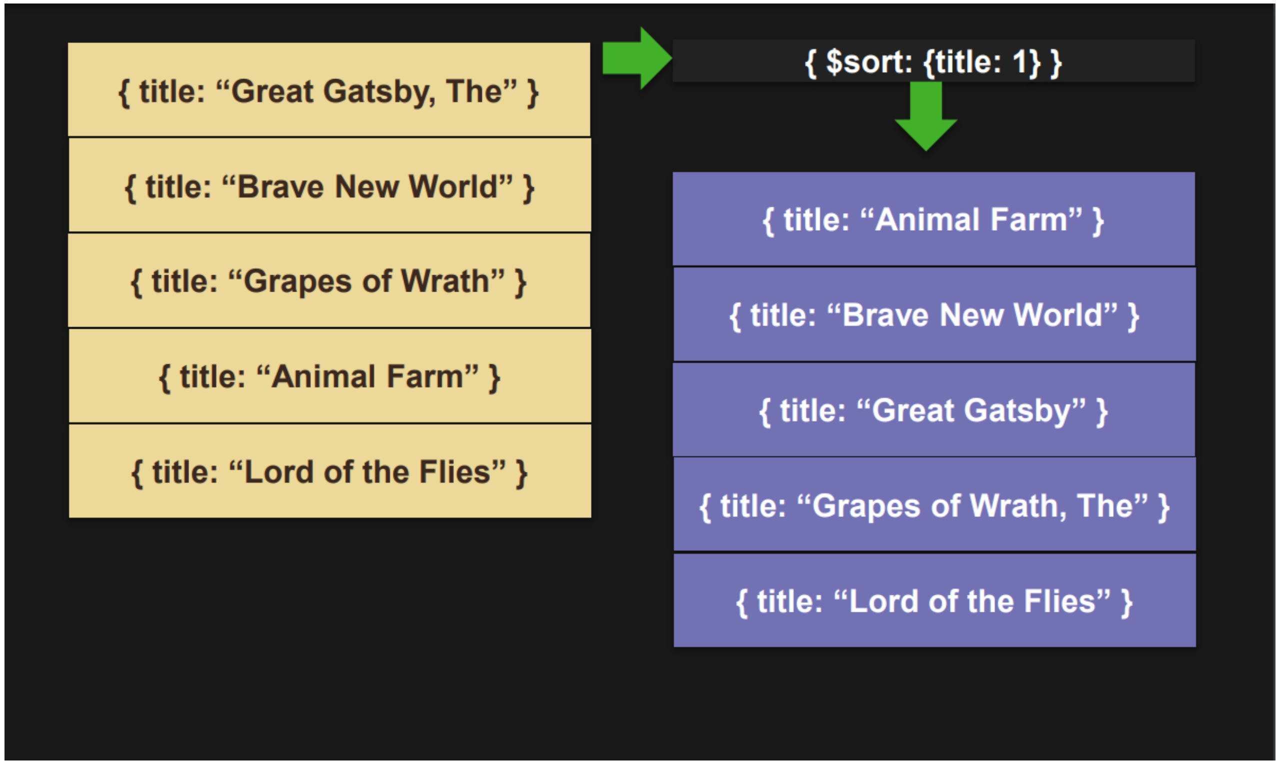
```
{ title: "The Great Gatsby",
  ISBN: "9781857150193",
  subjects: "1920s" }
```

Aggregation Pipeline Stages

- $sort, $limit, $skip
  - Sorts all input documents and returns them to the pipeline in sorted order
  - The $sort stage has the following prototype form
    - *{ $sort: { <field1>: <sort order>, <field2>: <sort order> ... } }*
      - *1 to specify ascending order.*
      - *-1 to specify descending order.*
  - $limit and $skip limits or skip the number of documents passed to the next stage in the pipeline
  - The $limit stage has the following prototype form
    - { $limit: <positive integer> }

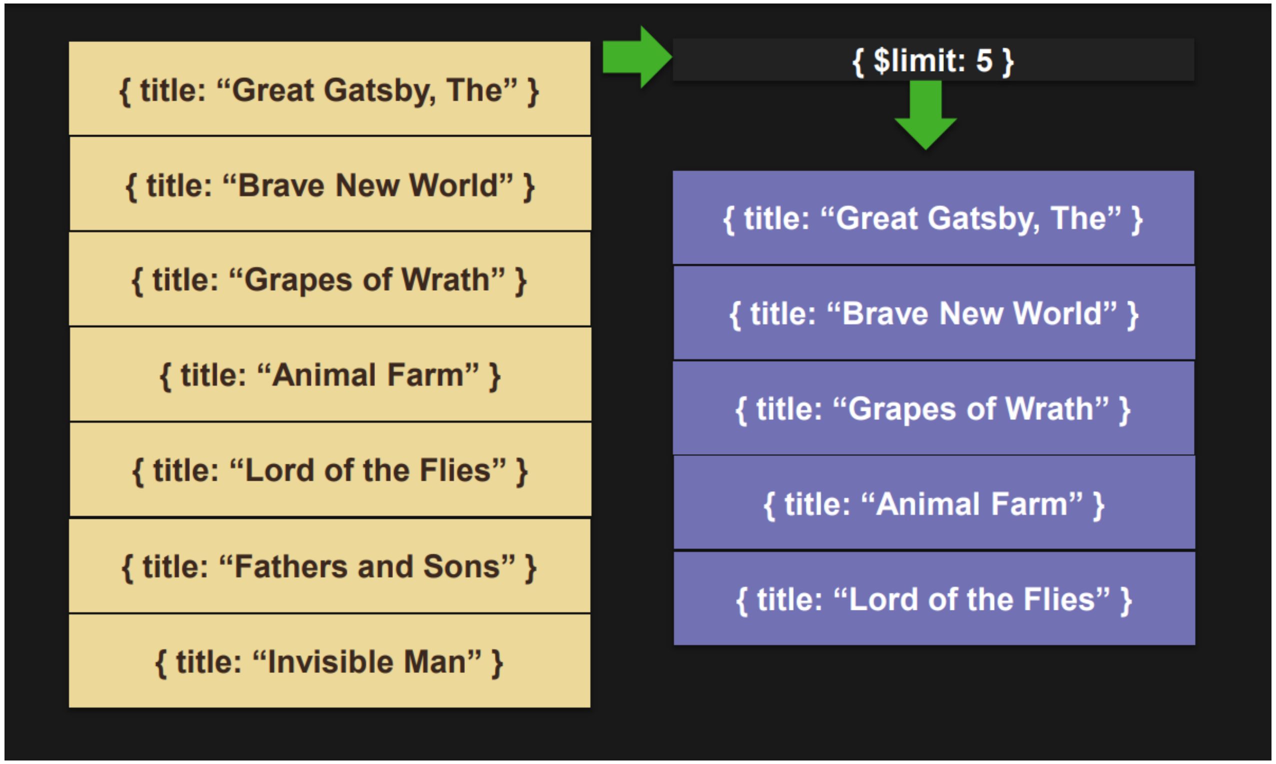
Aggregation Pipeline Stages

- $sort, $limit, $skip
  - Sort All the Documents in the Pipeline


{ title: "Great Gatsby, The" }
{ title: "Brave New World" }
{ title: "Grapes of Wrath" }
{ title: "Animal Farm" }
{ title: "Lord of the Flies" }
{ $sort: {title: 1} }
{ title: "Animal Farm" }
{ title: "Brave New World" }
{ title: "Great Gatsby" }
{ title: "Grapes of Wrath, The" }
{ title: "Lord of the Flies" }

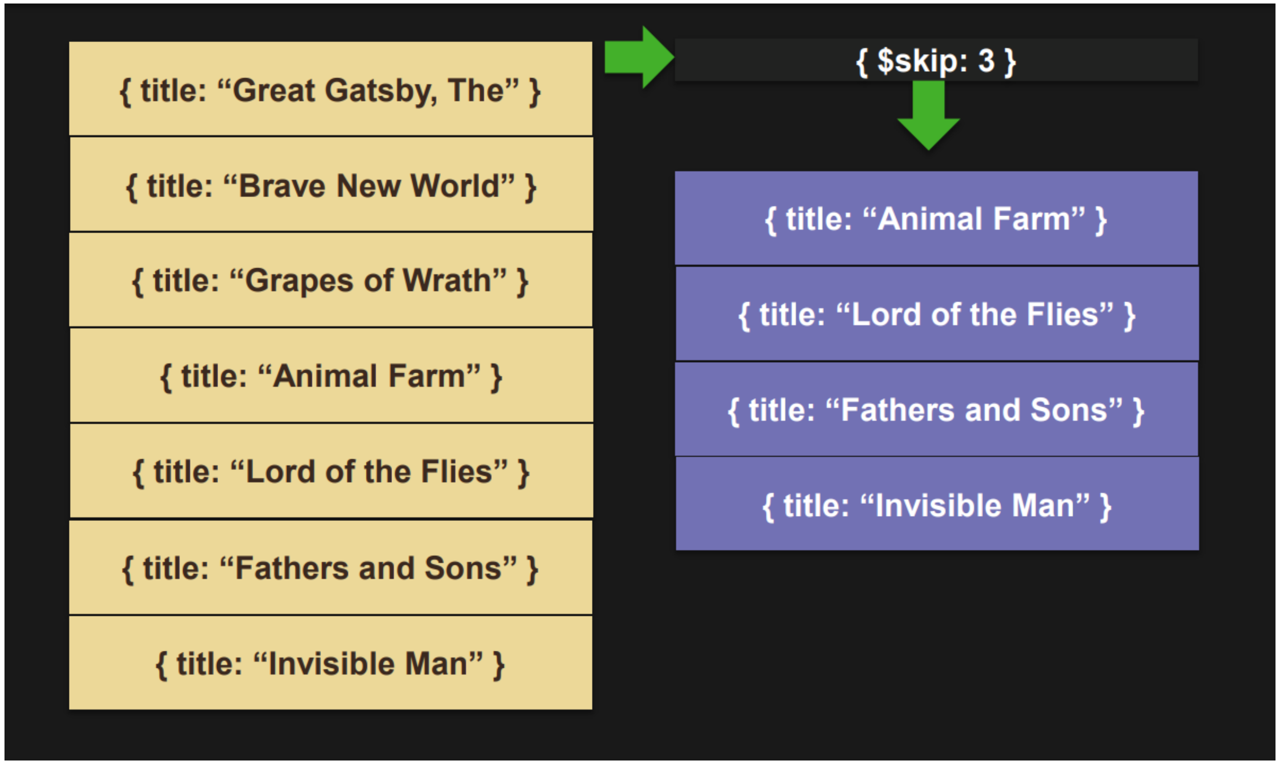
Aggregation Pipeline Stages

- $sort, $limit, $skip
  - Limit Documents Through the Pipeline


{ title: "Great Gatsby, The" }
{ title: "Brave New World" }
{ title: "Grapes of Wrath" }
{ title: "Animal Farm" }
{ title: "Lord of the Flies" }
{ title: "Fathers and Sons" }
{ title: "Invisible Man" }
{ $limit: 5 }
{ title: "Great Gatsby, The" }
{ title: "Brave New World" }
{ title: "Grapes of Wrath" }
{ title: "Animal Farm" }
{ title: "Lord of the Flies" }

Aggregation Pipeline Stages

- $sort, $limit, $skip
  - Skip documents in the pipeline


{ title: "Great Gatsby, The" }
{ title: "Brave New World" }
{ title: "Grapes of Wrath" }
{ title: "Animal Farm" }
{ title: "Lord of the Flies" }
{ title: "Fathers and Sons" }
{ title: "Invisible Man" }
{ $skip: 3 }
{ title: "Animal Farm" }
{ title: "Lord of the Flies" }
{ title: "Fathers and Sons" }
{ title: "Invisible Man" }