Chapter 3: Core Aggregation - Combining Information

Lab - $unwind

Back to the Question

The solution we used is below.

```
db.movies.aggregate([
  {
    $match: {
      languages: "English"
    }
  },
  {
    $project: { _id: 0, cast: 1, "imdb.rating": 1 }
  },
  {
    $unwind: "$cast"
  },
  {
    $group: {
      _id: "$cast",
      numFilms: { $sum: 1 },
      average: { $avg: "$imdb.rating" }
    }
  },
  {
    $project: {
      numFilms: 1,
      average: {
        $divide: [{ $trunc: { $multiply: ["$average", 10] } }, 10]
      }
    }
  },
  {
    $sort: { numFilms: -1 }
  },
  {
    $limit: 1
  }
])
```

We start with a familiar `$match` stage, looking for movies that include "English" as a language

```
{
  $match: {
    languages: "English"
  }
},
```

Next, we use a `$project` stage, keeping only the data necessary for the aggregation stages that follow

COPY

```
{
  $project: { _id: 0, cast: 1, "imdb.rating": 1 }
}
```

`$unwind` follows next, extracting every entry in the **cast** array and creating a document for each one

COPY

```
{
  $unwind: "$cast"
}
```

Our `$group` stage groups cast members together by their name, totals the number of documents, and gets the average **imdb.rating**

COPY

```
{
  $group: {
    _id: "$cast",
    numFilms: { $sum: 1 },
    average: { $avg: "$imdb.rating" }
  }
}
```

We then use a `$project` stage to truncate the `imdb.rating` to one decimal. This is done by first multiplying by 10, truncating the number, then dividing by 10

COPY

```
{
  $project: {
    numFilms: 1,
    average: {
      $divide: [
        { $trunc: { $multiply: ["$average", 10] } }
        , 10
      ]
    }
  }
}
```

Lastly, we `$sort` in descending order so the result with the greatest number of movies comes first, and then `$limit` our result to 1 document, giving the expected answer

```
{ "_id" : "John Wayne", "numFilms" : 107, "average" : 6.4 }
```

COPY

Proceed to next section