# EXPERIMENT 04

## Projection Operators

4.Create and demonstrate how projection operators ($, $elematch and $slice) would be used in the MondoDB.

Projection operators in MongoDB are used to shape the documents returned in the query results by including, excluding, or manipulating fields. Here are some commonly used projection operators:

### 1. Include Specific Fields:

- `{ field1: 1, field2: 1 }`: Includes only the specified fields in the result.

### 2. Exclude Specific Fields:

- `{ field1: 0, field2: 0 }`: Excludes the specified fields from the result.

### 3. Include the `_id` Field:

- `{ _id: 1 }`: Includes the `_id` field in the result. This is included by default unless explicitly excluded.

### 4. Exclude the `_id` Field:

- `{ _id: 0 }`: Excludes the `_id` field from the result.

### 5.Array Projection Operators:

- `$`: Projects the first element of an array.

- `$slice`: Projects a subset of elements of an array.

- `$elemMatch`: Projects only the first element that matches the specified condition in an array.

### 6. Meta-projection Operators:

- `$meta`: Projects metadata associated with each matching document, such as the score in text search queries.

### 7. Computed Fields:

- `$addFields`: Adds new fields to the documents in the result.

- `$project`: Reshapes documents by including, excluding, or renaming fields.

**Here's an example of using projection operators in MongoDB:** //

Include only the "name" and "age" fields, excluding "_id"

```
db.users.find({}, { name: 1, age: 1, _id: 0 });
```

// Exclude the "password" field from the query result db.users.find({},

{ password: 0 });

// Project the first three elements of the "scores" array field db.students.find({},

{ scores: { $slice: 3 } });

// Include only the first matching element in the "scores" array field greater than 70

```
db.students.find({}, { scores: { $elemMatch: { $gt: 70 } } });
```

// Include metadata associated with the text search score db.articles.find({ $text: { $search:

"mongodb" } }, { _id: 0, score: { $meta: "textScore" } });

These projection operators provide flexibility in shaping query results according to specific requirements.

## ADDING NEW DATASET CALLED PRODUCTS:

```
test> use db
switched to db db
db> show collections
candidates
locations
products
std
students_permission
```

## RETRIVING NAME AND RATINGS OF THE PRODUCTS:

```
db> db.products.find({},{name:1,rating:1});
[
  { _id: 'ac3', name: 'AC3 Phone', rating: 3.8 },
  { _id: 'ac7', name: 'AC7 Phone', rating: 4 },
  {
    _id: ObjectId('507d95d5719dbef170f15bf9'),
    name: 'AC3 Series Charger',
    rating: 2.8
  },
  {
    _id: ObjectId('507d95d5719dbef170f15bfa'),
    name: 'AC3 Case Green',
    rating: 1
  },
  {
    _id: ObjectId('507d95d5719dbef170f15bfb'),
    name: 'Phone Extended Warranty',
    rating: 5
  },
  {
    _id: ObjectId('507d95d5719dbef170f15bfc'),
    name: 'AC3 Case Black',
    rating: 2
  },
  {
    _id: ObjectId('507d95d5719dbef170f15bfd'),
    name: 'AC3 Case Red',
    rating: 4
  },
  {
    _id: ObjectId('507d95d5719dbef170f15bfe'),
    name: 'Phone Service Basic Plan',
    rating: 3
  },
  {
    _id: ObjectId('507d95d5719dbef170f15bff'),
    name: 'Phone Service Core Plan',
    rating: 3
  },
  {
    _id: ObjectId('507d95d5719dbef170f15c00'),
    name: 'Phone Service Family Plan',
    rating: 4
  },
  {
    _id: ObjectId('507d95d5719dbef170f15c01'),
    name: 'Cable TV Basic Service Package',
    rating: 3.9
  }
]
db>
```

- db: This refers to the current database connection.

- .products: This refers to the collection named products within the current database.
- .find({}): This is the find method which is used to find documents in a collection. The empty curly braces {} specify that all documents in the collection should be returned.

- ({name: 1, rating: 1}): This is the projection document which specifies which fields to include in the returned documents. In this case, only the name and rating fields are included by setting their values to 1.

The output of the query shows seven documents from the products collection. Each document has an _id field, which is the unique identifier for the document, and the two fields specified in the projection, name and rating.

## Excluding fields:

```
db> db.products.find({},{_id:0,type:0,limits:0,data:0});
[
  {
    name: 'AC3 Phone',
    brand: 'ACME',
    price: 200,
    rating: 3.8,
    warranty_years: 1,
    available: true
  },
  {
    name: 'AC7 Phone',
    brand: 'ACME',
    price: 320,
    rating: 4,
    warranty_years: 1,
    available: false
  },
  {
    name: 'AC3 Series Charger',
    price: 19,
    rating: 2.8,
    warranty_years: 0.25,
    for: [ 'ac3', 'ac7', 'ac9' ]
  },
  {
    name: 'AC3 Case Green',
    color: 'green',
    price: 12,
    rating: 1,
    warranty_years: 0
  },
  {
    name: 'Phone Extended Warranty',
    price: 38,
    rating: 5,
    warranty_years: 2,
    for: [ 'ac3', 'ac7', 'ac9', 'qp7', 'qp8', 'qp9' ]
  },
  {
    name: 'AC3 Case Black',
    color: 'black',
    price: 12.5,
    rating: 2,
    warranty_years: 0.25,
    available: false,
    for: 'ac3'
  },
  {
    name: 'AC3 Case Red',
    color: 'red',
    price: 12,
    rating: 4,
    warranty_years: 0.25,
    available: true,
    for: 'ac3'
  },
```

```
mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000
  },
  {
    name: 'AC3 Case Red',
    color: 'red',
    price: 12,
    rating: 4,
    warranty_years: 0.25,
    available: true,
    for: 'ac3'
  },
  {
    name: 'Phone Service Basic Plan',
    monthly_price: 40,
    rating: 3,
    term_years: 2
  },
  {
    name: 'Phone Service Core Plan',
    monthly_price: 60,
    rating: 3,
    term_years: 1
  },
  {
    name: 'Phone Service Family Plan',
    monthly_price: 90,
    rating: 4,
    sales_tax: true,
    term_years: 2
  },
  {
    name: 'Cable TV Basic Service Package',
    monthly_price: 50,
    rating: 3.9,
    term_years: 2,
    cancel_penalty: 25,
    sales_tax: true,
    additional_tarriffs: [
      { kind: 'federal tarriff', amount: { percent_of_service: 0.06 } },
      { kind: 'misc tarriff', amount: 2.25 }
    ]
  }
```

- ({id: 0, type: 0, limits: 0, data: 0}): This is the projection document which specifies which fields to exclude from the returned documents. In this case, the following fields are excluded by setting their values to 0: id, type, limits, and data.

The output of the query shows seven documents from the products collection. However, it only displays specific fields for each document based on what wasn't excluded in the projection. For instance, the first document shows:

- name: "AC3 Phone"
- brand: "ACME"
- price: 200
- rating: 3.8
- warranty_years: 1
- available: true

Since some fields were excluded, it might not show all the information available for each product in the collection.

## To get total number of counts for the above operation:

```
  }
]
db> db.products.find({},{_id:0,type:0,limits:0,data:0}).count();
11
db>
```

## Now lets use candidates.json dataset:

```
db> db.candidates.find({courses:{$elemMatch:{$eq:"Computer Science"}}},{name:1,"courses.$":1});
[
  { _id: ObjectId('6657ff95946a866dbb971e60'), name: 'Bob Johnson' },
  { _id: ObjectId('6657ff95946a866dbb971e65'), name: 'Gabriel Miller' },
  { _id: ObjectId('6657ff95946a866dbb971e69'), name: 'Kevin Lewis' }
]
```

- db: This refers to the current database connection.
- db.candidates: This refers to the collection named "candidates" within the current database.
- .find({courses: {$elemMatch: {$eq:"Computer Science"}}}): This is the find method with a filter expression. The filter expression searches for documents in the "courses" field that contain an element matching the value "Computer Science".
    - courses: This refers to the field in the collection documents that contains information about the courses taken by the candidate.
    - $elemMatch: This is a MongoDB operator that allows you to find documents within an array that contain an element that matches a specified condition.
    - $eq: This is a MongoDB comparison operator that checks for equality.
- ({name: 1, "courses, $": 1}): This is the projection document which specifies which fields to include in the returned documents.
    - name: 1: This includes the "name" field in the output.
    - "courses, $": This includes the "courses" field but uses the positional operator $ to only return the matching element (the course named "Computer Science" in this case).

The output of the query shows three documents where the "courses" field contains the value "Computer Science". The output only shows the "name" field and the matched course ("Computer Science") from the "courses" field for each document.

## $elemMatch

The $elemMatch operator limits the contents of an <array> field from the query results to contain only the **first** element matching the $elemMatch condition.

```
db> db.candidates.find({courses:{$elemMatch:{$eq:"Physics"}}},{name:1,"courses,$":1});
[
  { _id: ObjectId('6657ff95946a866dbb971e60'), name: 'Bob Johnson' },
  { _id: ObjectId('6657ff95946a866dbb971e62'), name: 'Emily Jones' }
]
db>
```

- .find({courses: {$elemMatch: {$eq:"Physics"}}}): This is the find method with a filter expression that searches for documents in the "courses" field that contain an element matching the value "Physics".
    - courses: This refers to the field in the collection documents that contains information about the courses taken by the candidate.
    - $elemMatch: This is a MongoDB operator that allows you to find documents within an array that contain an element that matches a specified condition.
    - $eq: This is a MongoDB comparison operator that checks for equality.
- ({name: 1, "courses.$": 1}): This is the projection document which specifies which fields to include in the returned documents.
    - name: 1: This includes the "name" field in the output.
    - "courses.$": This includes the "courses" field but uses the positional operator $ to only return the matching element (the course named "Physics" in this case).

The output of the query shows two documents where the "courses" field contains the value "Physics". The output only shows the "name" field and the matched course ("Physics") from the "courses" field for each document.

# $slice

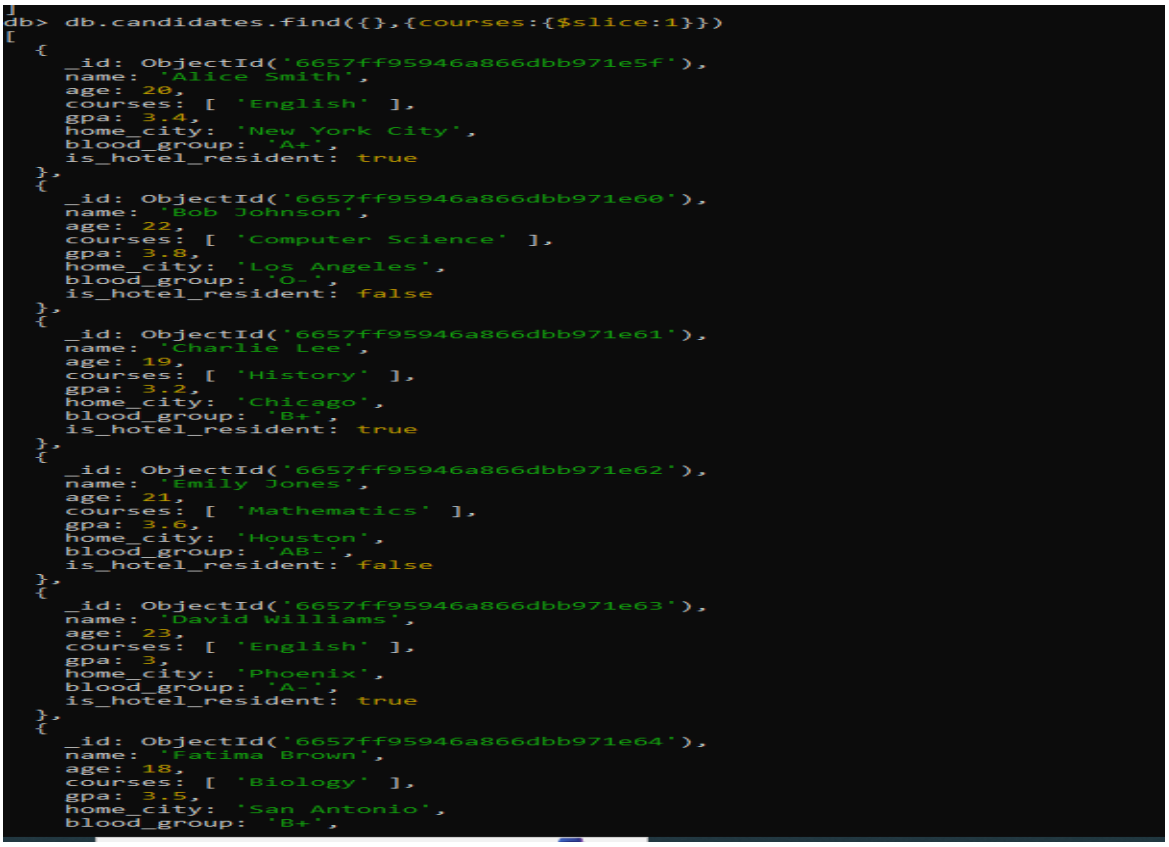
The $slice projection operator specifies the number of elements in an array to return in the query result.

***Syntax:***

```
db.collection.find(
 <query>,
{ <arrayField>: { $slice: <number> } }
 );
```

| Value | Description |
|---|---|
| `$slice: <number>` | Specifies the number of elements to return in the `<arrayField>`. For `<number>`:<br>• Specify a positive number `n` to return the first `n` elements.<br>• Specify a negative number `n` to return the last `n` elements.<br>If the `<number>` is greater than the number of array elements, the query returns all array elements. |
| `$slice: [ <number to skip>, <number to return> ]` | Specifies the number of elements to return in the `<arrayField>` after skipping the specified number of elements starting from the first element. You must specify both elements.<br>For the `<number to skip>`:<br>• Specify a positive number `n` to skip `n` elements from the start of the array; i.e. 0th index position. Based on a zero-based array index, `1` indicates the starting position of the 2nd element, etc. If `n` is greater than the number of array elements, the query returns an empty array for the `<arrayField>`.<br>• Specify a negative number `n` to skip backward `n` elements from the start of the array; i.e. 0th index position Based on a zero-based array index (i.e. the first element is at index 0), `-1` indicates the starting position of the last element, etc. If the absolute value of the negative number is greater than the number of array elements, the starting position is the start of the array.<br>For the `<number to return>`, you must specify a *positive* number `n` to return the next `n` elements, starting after skipping the specified number. |

**$slice operation**



```
db> db.candidates.find({},{courses:{$slice:1}})
[
  {
    _id: ObjectId('6657ff95946a866dbb971e5f'),
    name: 'Alice Smith',
    age: 20,
    courses: [ 'English' ],
    gpa: 3.4,
    home_city: 'New York City',
    blood_group: 'A+',
    is_hotel_resident: true
  },
  {
    _id: ObjectId('6657ff95946a866dbb971e60'),
    name: 'Bob Johnson',
    age: 22,
    courses: [ 'Computer Science' ],
    gpa: 3.8,
    home_city: 'Los Angeles',
    blood_group: 'O-',
    is_hotel_resident: false
  },
  {
    _id: ObjectId('6657ff95946a866dbb971e61'),
    name: 'Charlie Lee',
    age: 19,
    courses: [ 'History' ],
    gpa: 3.2,
    home_city: 'Chicago',
    blood_group: 'B+',
    is_hotel_resident: true
  },
  {
    _id: ObjectId('6657ff95946a866dbb971e62'),
    name: 'Emily Jones',
    age: 21,
    courses: [ 'Mathematics' ],
    gpa: 3.6,
    home_city: 'Houston',
    blood_group: 'AB-',
    is_hotel_resident: false
  },
  {
    _id: ObjectId('6657ff95946a866dbb971e63'),
    name: 'David Williams',
    age: 23,
    courses: [ 'English' ],
    gpa: 3,
    home_city: 'Phoenix',
    blood_group: 'A-',
    is_hotel_resident: true
  },
  {
    _id: ObjectId('6657ff95946a866dbb971e64'),
    name: 'Fatima Brown',
    age: 18,
    courses: [ 'Biology' ],
    gpa: 3.5,
    home_city: 'San Antonio',
    blood_group: 'B+',
```

- `({courses: {$slice: 1}})`: This part defines a projection document using curly braces `{}`. The projection document specifies which fields to include or exclude when returning results. In this case, it includes only the `courses` field but uses the `$slice` operator to limit the number of courses returned to the first element (1) in the array.

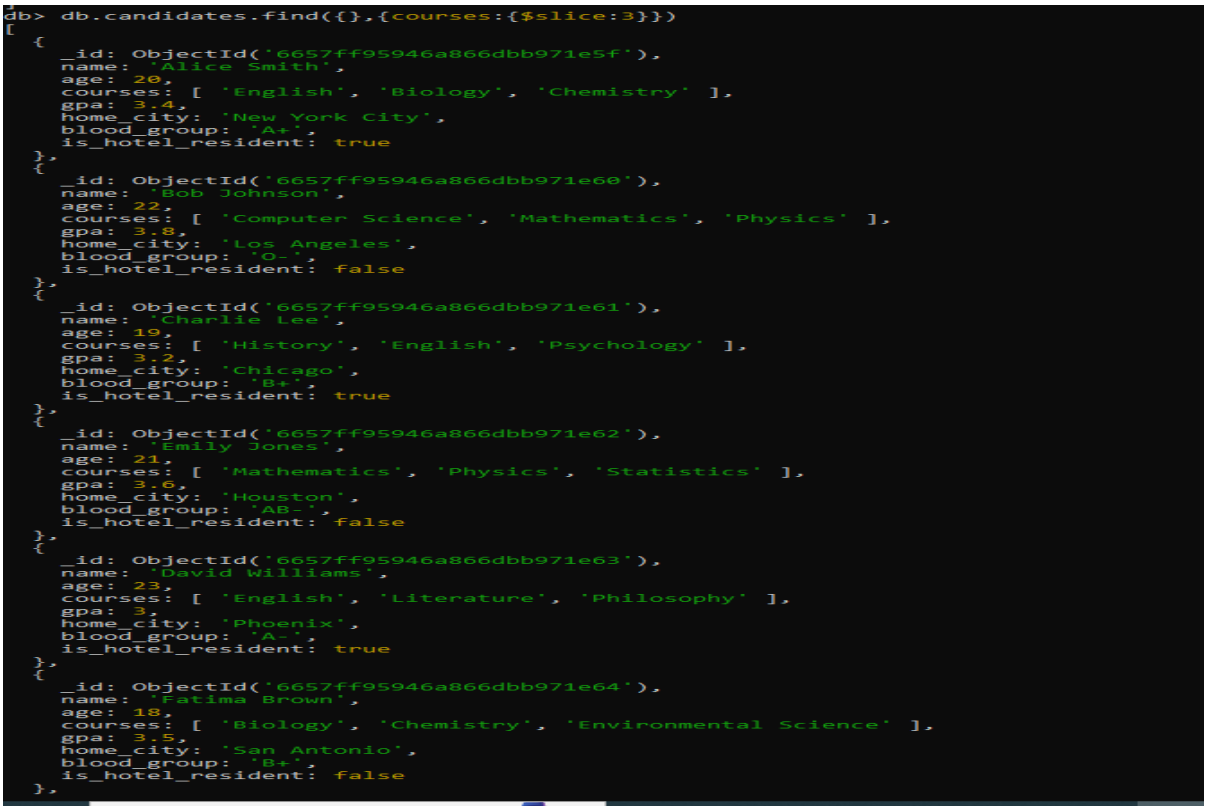
- `$slice`: This operator limits the number of elements returned from an array. It takes two arguments:
    - The first argument specifies the starting index of the slice (zero-based).
    - The second argument (optional) specifies the number of elements to return. If omitted, it returns all elements from the starting index to the end of the array.

So, the query fetches all documents from the `candidates` collection and projects only the first course from the `courses` field for each candidate.

```
db> db.candidates.find({},{courses:{$slice:[1,3]}})
[
  {
    _id: ObjectId('6657ff95946a866dbb971e5f'),
    name: 'Alice Smith',
    age: 20,
    courses: [ 'Biology', 'Chemistry' ],
    gpa: 3.4,
    home_city: 'New York City',
    blood_group: 'A+',
    is_hotel_resident: true
  },
  {
    _id: ObjectId('6657ff95946a866dbb971e60'),
    name: 'Bob Johnson',
    age: 22,
    courses: [ 'Mathematics', 'Physics' ],
    gpa: 3.8,
    home_city: 'Los Angeles',
    blood_group: 'O-',
    is_hotel_resident: false
  },
  {
    _id: ObjectId('6657ff95946a866dbb971e61'),
    name: 'Charlie Lee',
    age: 19,
    courses: [ 'English', 'Psychology' ],
    gpa: 3.2,
    home_city: 'Chicago',
    blood_group: 'B+',
    is_hotel_resident: true
  },
  {
    _id: ObjectId('6657ff95946a866dbb971e62'),
    name: 'Emily Jones',
    age: 21,
    courses: [ 'Physics', 'Statistics' ],
    gpa: 3.6,
    home_city: 'Houston',
    blood_group: 'AB-',
    is_hotel_resident: false
  },
  {
    _id: ObjectId('6657ff95946a866dbb971e63'),
    name: 'David Williams',
    age: 23,
    courses: [ 'Literature', 'Philosophy' ],
    gpa: 3,
    home_city: 'Phoenix',
    blood_group: 'A-',
    is_hotel_resident: true
  },
  {
    _id: ObjectId('6657ff95946a866dbb971e64'),
    name: 'Fatima Brown',
    age: 18,
    courses: [ 'Chemistry', 'Environmental Science' ],
    gpa: 3.5,
    home_city: 'San Antonio',
    blood_group: 'B+',
    is_hotel_resident: false
  },
```

`({courses: {$slice: 3}})`: This part defines a projection document using curly braces `{}`. The projection document specifies which fields to include or exclude when returning results. In this case, it includes only the `courses` field but uses the `$slice` operator to limit the number of courses returned to the first 3 elements (3) in the array.

**$slice operation [1:3]**



```
db> db.candidates.find({},{courses:{$slice:3}})
[
  {
    _id: ObjectId('6657ff95946a866dbb971e5f'),
    name: 'Alice Smith',
    age: 20,
    courses: [ 'English', 'Biology', 'Chemistry' ],
    gpa: 3.4,
    home_city: 'New York City',
    blood_group: 'A+',
    is_hotel_resident: true
  },
  {
    _id: ObjectId('6657ff95946a866dbb971e60'),
    name: 'Bob Johnson',
    age: 22,
    courses: [ 'Computer Science', 'Mathematics', 'Physics' ],
    gpa: 3.8,
    home_city: 'Los Angeles',
    blood_group: 'O-',
    is_hotel_resident: false
  },
  {
    _id: ObjectId('6657ff95946a866dbb971e61'),
    name: 'Charlie Lee',
    age: 19,
    courses: [ 'History', 'English', 'Psychology' ],
    gpa: 3.2,
    home_city: 'Chicago',
    blood_group: 'B+',
    is_hotel_resident: true
  },
  {
    _id: ObjectId('6657ff95946a866dbb971e62'),
    name: 'Emily Jones',
    age: 21,
    courses: [ 'Mathematics', 'Physics', 'Statistics' ],
    gpa: 3.6,
    home_city: 'Houston',
    blood_group: 'AB-',
    is_hotel_resident: false
  },
  {
    _id: ObjectId('6657ff95946a866dbb971e63'),
    name: 'David Williams',
    age: 23,
    courses: [ 'English', 'Literature', 'Philosophy' ],
    gpa: 3,
    home_city: 'Phoenix',
    blood_group: 'A-',
    is_hotel_resident: true
  },
  {
    _id: ObjectId('6657ff95946a866dbb971e64'),
    name: 'Fatima Brown',
    age: 18,
    courses: [ 'Biology', 'Chemistry', 'Environmental Science' ],
    gpa: 3.5,
    home_city: 'San Antonio',
    blood_group: 'B+',
    is_hotel_resident: false
  },
```

- `db.candidates.find({}, {courses:{$slice: [1,3]}})`: This line uses the `find` method with two arguments to retrieve documents from the `candidates` collection and apply a projection to the output.
- `db.candidates.find({})`: This part finds all documents in the `candidates` collection. The empty curly braces `{}` specify that no filtering criteria are applied to the find operation.
- `{courses:{$slice: [1,3]}}`: This is the projection document that specifies how to modify the retrieved documents before returning them. It includes only the `courses` field using the `$slice` operator to limit the number of courses returned.
- `courses`: This specifies the field to be included in the output.
- `$slice`: This operator limits the elements returned from an array field. It takes two arguments in this case:

The first argument `[1]` specifies the starting index of the slice (which is 1, since arrays are zero-based in MongoDB). So it starts from the second element (index 1).The second argument `3` specifies the number of elements to return (which is 3).

So, the query retrieves all documents from the `candidates` collection and returns only a portion of the `courses` field. It starts from the second element (index 1) and includes the next 3 elements, resulting in a maximum of 3 courses being returned for each candidate.