# Experiment 04

## Create and demonstrate how projection operators ($, Selematch and $slice) would be used in the MongaDB

Projections in MongoDB are a powerful tool for controlling the data retrieved from queries. They allow you to specify exactly which fields you want to return from documents in a collection, offering several advantages:

- **Reduced Data Transfer:** By fetching only the necessary fields, you minimize the amount of data transferred between the client and server, leading to faster queries and improved network efficiency.
- **Enhanced Performance:** Less data to process translates to faster query execution times on both the client application and MongoDB server.
- **Focused Results:** Projections help you retrieve only the relevant data for your specific needs. This is especially beneficial when working with large documents containing numerous fields, many of which might not be essential for your current task.
- **Data Confidentiality:** Projections can be used to safeguard sensitive information. By excluding specific fields containing confidential data from the projection document, you can ensure they are not unintentionally returned in query results.

Here's a breakdown of the key aspects of projections in MongoDB:

### Using Projections with `find` and `findOne` methods:

Projections are typically used with the `find` and `findOne` methods in MongoDB. These methods accept two optional arguments:

1. **Query Document:** This document specifies the criteria for selecting documents (e.g., filtering based on specific field values).
2. **Projection Document:** This document defines which fields to include or exclude in the returned results.

The projection document is the second argument passed to these methods.

## Specifying Fields for Inclusion/Exclusion:

The projection document is a key-value structure where:

- **Keys:** Represent the field names in the documents.
- **Values:** Specify whether to include or exclude the corresponding field.
    - **Value of 1 (or `true`):** Includes the field in the returned documents.
    - **Value of 0 (or `false`):** Excludes the field from the returned documents.

**Important Note:** By default, the `_id` field is always returned unless you explicitly set it to 0 in the projection document.

## Projection Operators for Advanced Control:

While including/excluding fields is the core functionality, MongoDB offers several projection operators for more complex scenarios:

- **`$` ( positional operator ):** Selects the first element from an array that matches a specific condition.
- **`$elemMatch`:** Filters and includes elements from an array that satisfy a specified criteria.
- **`$slice`:** Returns a limited portion (subset) of an array.

Upload the new dataset link

## Introduction to projection operators

| Operator | Operator Description | Operator Use and Syntax |
|---|---|---|
| $ | This operator projects the first element in the array type field that matches the query condition. | { <query> },{ <array type field>.$: 1 } |
| $elemMatch | This operator projects the first element in the array type field that matches the condition given in the **$elemMatch** | { <array type field>: { $elemMatch: { <query> } }} |

| | | |
|---|---|---|
| $slice | This operator limits the number of elements projected from an array. We can also use skip and limit in this operator**.** | { <array type field>: { $slice: <number of elements> }} |

**Example 1**: Retrieve Name, Age, and GPA

```
db.collections.find({},{ name: 1, age: 1, gpa: 1});
```

```
db> db.collections.find({},{name:1, age:1,gpa:1}).count()
12
db> db.collections.find({},{name:1, age:1,gpa:1})
[
  {
    _id: ObjectId('667c21d4fef4401250753918'),
    name: 'Alice Smith',
    age: 20,
    gpa: 3.4
  },
  {
    _id: ObjectId('667c21d4fef4401250753919'),
    name: 'Bob Johnson',
    age: 22,
    gpa: 3.8
  },
  {
    _id: ObjectId('667c21d4fef440125075391a'),
    name: 'Charlie Lee',
    age: 19,
    gpa: 3.2
  },
  {
    _id: ObjectId('667c21d4fef440125075391b'),
    name: 'Emily Jones',
    age: 21,
    gpa: 3.6
  },
  {
    _id: ObjectId('667c21d4fef440125075391c'),
    name: 'David Williams',
    age: 23,
    gpa: 3
  },
  {
    _id: ObjectId('667c21d4fef440125075391d'),
    name: 'Fatima Brown',
    age: 18,
    gpa: 3.5
  },
  {
    _id: ObjectId('667c21d4fef440125075391e'),
    name: 'Gabriel Miller',
    age: 24,
    gpa: 3.9
  },
  {
    _id: ObjectId('667c21d4fef440125075391f'),
    name: 'Hannah Garcia',
    age: 20,
    gpa: 3.3
  },
  {
    _id: ObjectId('667c21d4fef4401250753920'),
    name: 'Isaac Clark',
    age: 22,
    gpa: 3.7
  },
  {
    _id: ObjectId('667c21d4fef4401250753921'),
    name: 'Jessica Moore',
    age: 19,
    gpa: 3.1
  },
  {
    _id: ObjectId('667c21d4fef4401250753922'),
    name: 'Kevin Lewis',
    age: 21,
    gpa: 4
  },
  {
    _id: ObjectId('667c21d4fef4401250753923'),
    name: 'Lily Robinson',
```

**Example 02**: Variation: Exclude Fields

```
db.collections.find({}, { _id: 0, courses: 0})
```

```
db> db.collections.find({},{_id:0,courses:0})
[
  {
    name: 'Alice Smith',
    age: 20,
    gpa: 3.4,
    home_city: 'New York City',
    blood_group: 'A+',
    is_hotel_resident: true
  },
  {
    name: 'Bob Johnson',
    age: 22,
    gpa: 3.8,
    home_city: 'Los Angeles',
    blood_group: 'O-',
    is_hotel_resident: false
  },
  {
    name: 'Charlie Lee',
    age: 19,
    gpa: 3.2,
    home_city: 'Chicago',
    blood_group: 'B+',
    is_hotel_resident: true
  },
  {
    name: 'Emily Jones',
    age: 21,
    gpa: 3.6,
    home_city: 'Houston',
    blood_group: 'AB-',
    is_hotel_resident: false
  },
  {
    name: 'David Williams',
    age: 23,
    gpa: 3,
    home_city: 'Phoenix',
    blood_group: 'A-',
    is_hotel_resident: true
  },
  {
    name: 'Fatima Brown',
    age: 18,
    gpa: 3.5,
    home_city: 'San Antonio',
    blood_group: 'B+',
    is_hotel_resident: false
  },
  {
    name: 'Gabriel Miller',
    age: 24,
    gpa: 3.9,
    home_city: 'San Diego',
    blood_group: 'O+',
    is_hotel_resident: true
  },
  {
    name: 'Hannah Garcia',
    age: 20,
```

**Example 03:** Projection Operator($elemMatch)

Find Candidates enrolled in “Computer Science” with Specific Projection

```
db> db.collections.find({ courses:{$elemMatch:{$eq:"Computer Science"}}},{name: 1,"courses.$":1})
[
  {
    _id: ObjectId('667c21d4fef4401250753919'),
    name: 'Bob Johnson',
    courses: [ 'Computer Science' ]
  },
  {
    _id: ObjectId('667c21d4fef440125075391e'),
    name: 'Gabriel Miller',
    courses: [ 'Computer Science' ]
  },
  {
    _id: ObjectId('667c21d4fef4401250753922'),
    name: 'Kevin Lewis',
    courses: [ 'Computer Science' ]
  }
]
```

**Example 04:** Projection Operator($slice)

Retrieve all candidates with first two courses

```
db> db.collections.find({},{name: 1,courses:{$slice:2}})
[
  {
    _id: ObjectId('667c21d4fef4401250753918'),
    name: 'Alice Smith',
    courses: [ 'English', 'Biology' ]
  },
  {
    _id: ObjectId('667c21d4fef4401250753919'),
    name: 'Bob Johnson',
    courses: [ 'Computer Science', 'Mathematics' ]
  },
  {
    _id: ObjectId('667c21d4fef440125075391a'),
    name: 'Charlie Lee',
    courses: [ 'History', 'English' ]
  },
  {
    _id: ObjectId('667c21d4fef440125075391b'),
    name: 'Emily Jones',
    courses: [ 'Mathematics', 'Physics' ]
  },
  {
    _id: ObjectId('667c21d4fef440125075391c'),
    name: 'David Williams',
    courses: [ 'English', 'Literature' ]
  },
  {
    _id: ObjectId('667c21d4fef440125075391d'),
    name: 'Fatima Brown',
    courses: [ 'Biology', 'Chemistry' ]
  },
  {
    _id: ObjectId('667c21d4fef440125075391e'),
    name: 'Gabriel Miller',
    courses: [ 'Computer Science', 'Engineering' ]
  },
  {
    _id: ObjectId('667c21d4fef440125075391f'),
    name: 'Hannah Garcia',
    courses: [ 'History', 'Political Science' ]
  },
  {
    _id: ObjectId('667c21d4fef4401250753920'),
    name: 'Isaac Clark',
    courses: [ 'English', 'Creative Writing' ]
  },
  {
    _id: ObjectId('667c21d4fef4401250753921'),
    name: 'Jessica Moore',
    courses: [ 'Biology', 'Ecology' ]
  },
  {
    _id: ObjectId('667c21d4fef4401250753922'),
    name: 'Kevin Lewis',
    courses: [ 'Computer Science', 'Artificial Intelligence' ]
  },
  {
    _id: ObjectId('667c21d4fef4401250753923'),
    name: 'Lily Robinson',
    courses: [ 'History', 'Art History' ]
  }
]
db>
```

Here are some resources for further exploration:

- MongoDB Projection documentation: https://www.mongodb.com/docs/manual/reference/operator/projection/
- MongoDB Projection tutorial: https://www.mongodb.com/docs/manual/reference/operator/projection/