**CRUD I: QUERYING ON ARRAY FIELDS**

**Querying for An Entire Array**

At this point, we should be familiar with querying in MongoDB using the .find() method. Let’s take things a step further by learning how we can use this same method to filter documents based on array fields.

Consider a collection called books where each document shares a similar structure to the following:

{  
  \_id: ObjectId(...),  
  title: "Alice in Wonderland",  
  author: "Lewis Carroll",  
  year\_published: 1865,  
  genres: ["childrens", "fantasy", "literary nonsense"]  
}

Imagine we are looking for a new book to dive into – specifically, one that spans the young adult, fantasy, and adventure genres. We can query the collection for an array containing these exact values by using the .find() method and passing in a query argument that includes the field and array we want to match:

db.books.find({ genres: ["young adult", "fantasy", "adventure"] })

This query would return documents where the genres field is an array containing exactly three values, "young adult", "fantasy", and "adventure". For example, we would get a result that might look like this:

{  
  \_id: ObjectId(...),  
  title: "Harry Potter and The Sorcerer's Stone",  
  author: "JK Rowling",  
  year\_published: 1997,  
  genres: ["young adult", "fantasy", "adventure"]  
},  
{  
  \_id: ObjectId(...),  
  title: "The Gilded Ones",  
  author: "Namina Forna",  
  year\_published: 2021,  
  genres: ["young adult", "fantasy", "adventure"]  
}

Note that this query would only return documents where the array field contains precisely the values included in the query in the specified order. The following document contains the same values as mentioned in our query, but it wouldn’t be matched by our search because these values are in a different order:

{  
  \_id: ObjectId(...),  
  title: "Children of Blood and Bone",  
  author: "Tomi Adeyemi",  
  year\_published: 2018,  
  genres: ["fantasy", "young adult", "adventure"]  
}

The following document would also not be matched because it contains an additional value not specified by our query:

{  
  \_id: ObjectId(...),  
  title: "Eragon",  
  author: "Christopher Paolini,  
  year\_published: 2002,  
  genres: ["young adult", "fantasy", "adventure", "science fiction"]  
}

Before moving on, let’s practice querying array fields for exact matches!

**Instructions**

**1.**

We recently upgraded our database. We assigned some restaurants a new field named michelin\_stars which contains an array of years (e.g., 2019) that they received a Michelin star for their outstanding cuisine.

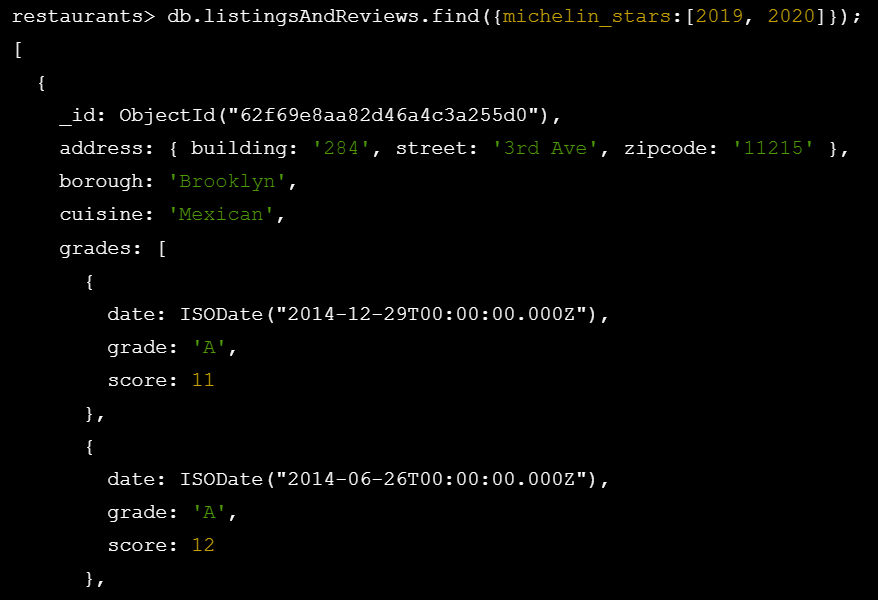
Connect to the restaurants database. Then, query the listingsAndReviews collection for all restaurants that earned exactly two michelin\_stars in the years 2019 and 2020.

Checkpoint 2 Passed

Hint

To query a collection for an array containing values for a specific field, you can use the following syntax:

db.<collection>.find({ <field>: [<value1>, <value2>, ....] });

****

**Matching Multiple Array Elements with $all**

So far, we’ve learned to query an array for exact matches, or individual elements. These are two extremes: searching for a specific ordering of elements, or only matching a single element. MongoDB offers us a middle ground. We can use the [$all](https://www.mongodb.com/docs/manual/reference/operator/query/all/?utm_campaign=academia_partners&utm_source=codecademy&utm_medium=referral) operator to match documents for an array field that includes all the specified elements, without regard for the order of the elements or additional elements in the array.

For example, let’s say we’ve finished our young adult novel and are ready to move on to something that spans the science fiction and adventure genres. We could use the $all operator to perform this query, like so:

db.books.find({ genres: { $all: [ "science fiction", "adventure" ] } })

This query might return the following documents:

{  
  \_id: ObjectId(...),  
  title: "Jurassic Park",  
  author: "Michael Crichton",  
  year\_published: 1990,  
  genres: ["science fiction", "adventure", "fantasy", "thriller"]  
},  
{  
  \_id: ObjectId(...),  
  title: "A Wrinkle in Time",  
  author: "Madeleine L'Engle",  
  year\_published: 1962,  
  genres: ["young adult", "fantasy", "science fiction", "adventure"]  
},  
{  
  \_id: ObjectId(...),  
  title: "Dune",  
  author: "Frank Herbert",  
  year\_published: 1965,  
  genres: ["science fiction", "fantasy", "adventure"]  
},  
…

Notice that using the $all operator will match documents where the given array field contains all the specified elements in *any* order, not necessarily the order provided in the query. Furthermore, our search returns documents where the genres array includes other elements, in addition to the ones specified in our query.

Let’s practice writing queries with the $all operator!

**Instructions**

**1.**

Connect to the restaurants database. Then, search the listingsAndReviews collection for any restaurants where the michelin\_stars field has at least two award years: 2018 and 2019.

Checkpoint 2 Passed

Hint

Be sure to run the command use restaurants to connect to the restaurants collection first. To match multiple array elements in a query, you can use the following syntax:

db.<collection>.find({ <field>: { $all: [ <value1>, <value2>, … ] } })



\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_