**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.

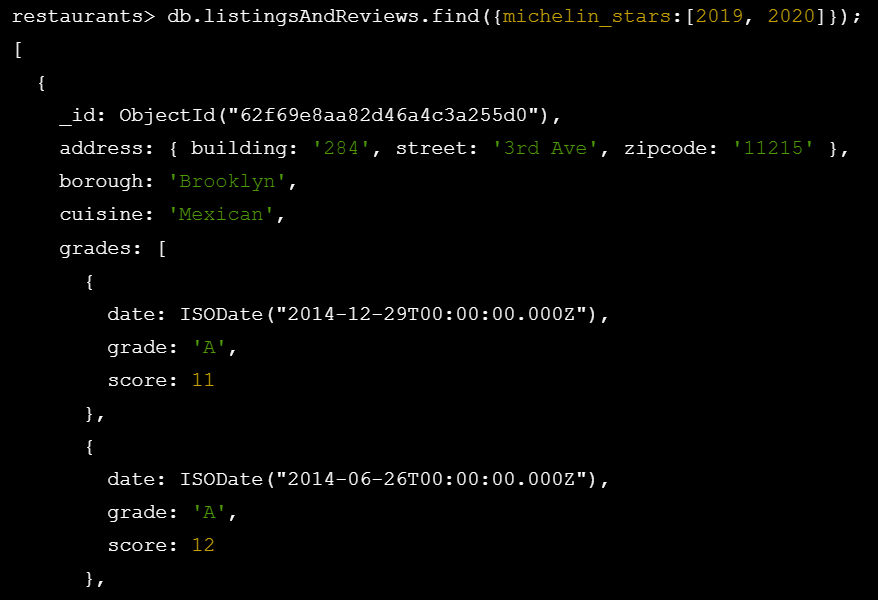
Connect to the restaurants database. Then, query the listingsAndReviews collection for all restaurants that earned exactly two michelin\_stars in the years 2019 and 2020.

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>, … ] } })



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

**Querying on Compound Filter Conditions**

In addition to querying array fields for exact matches and individual elements, we can use comparison operators to search for documents where elements in an array field meet some condition or range of conditions.

For example, imagine we have a collection of tennis athletes, called tennis\_players, with each document having a similar structure:

{  
  \_id: ObjectId(...),  
  name: "Serena Williams",  
  country: "United States",  
  wimbledon\_singles\_wins: [2002, 2003, 2009, 2010, 2012, 2015, 2016]  
}

If we wanted to search this collection for all athletes who have been Wimbledon Singles Champions from the year 2000 onward, we could run the following query:

db.tennis\_players.find({ wimbledon\_singles\_wins: { $gte: 2000 } });

This would return all documents where the wimbledon\_singles\_wins array has at least one element with a value of 2000 or greater. Our query result might look something like this:

{  
  \_id: ObjectId(...),  
  name: "Serena Williams",  
  country: "United States",  
  wimbledon\_singles\_wins: [2002, 2003, 2009, 2010, 2012, 2015, 2016]  
},  
{  
  \_id: ObjectId(...),  
  name: "Venus Williams",  
  country: "United States",  
  wimbledon\_singles\_wins: [2000, 2001, 2005, 2007, 2008]  
},  
{  
  \_id: ObjectId(...),  
  name: "Roger Federer",  
  country: "Switzerland",  
  wimbledon\_singles\_wins: [2003, 2004, 2005, 2006, 2007, 2009, 2012, 2017]  
},

We can also query based on compound conditions. Let’s consider that we want to search our tennis\_players collection to find all athletes who won a Wimbledon Singles Championship either before 1935, in the first 50 years of the championship, or after 1971, in the 50 most recent years of the tournament. We could achieve this with the following query:

db.tennis\_players.find({ wimbledon\_singles\_wins: { $gt: 1971, $lt: 1935 } })

This might return the following set of documents:

{  
  \_id: ObjectId(...),  
  name: "Suzanna Lenglen",  
  country: "United States",  
  wimbledon\_singles\_wins: [1919, 1920, 1921, 1922, 1923, 1925]  
},  
{  
  \_id: ObjectId(...),  
  name: "Roger Federer",  
  country: "Switzerland",  
  wimbledon\_singles\_wins: [2003, 2004, 2005, 2006, 2007, 2009, 2012, 2017]  
},  
…

Note that this query would match documents where the array contains elements that satisfy the query conditions in some combination. One element could satisfy the greater than 1971 condition, while another could satisfy the less than 1935 condition. And if the ranges overlapped, a single element could satisfy both conditions. However, using this syntax, it is not necessary that a single element satisfies all conditions.

In the next exercise we’ll learn how to filter our queries such that the matching documents have at least one array element that satisfies *all* the specified criteria.

Before moving on, let’s practice querying with comparison operators on compound filter conditions!

**Instructions**

**1.**

Connect to the restaurants database. Then, search the listingsAndReviews collection for restaurants that received a Michelin star after the year 2010.

Checkpoint 2 Passed

Hint

Be sure to run the command use restaurants to connect to the restaurants database. To query on an array field based on a single condition, you can use the following syntax:

db.<collection>.find({ <field>: { <operator>: <value> } })

**2.**

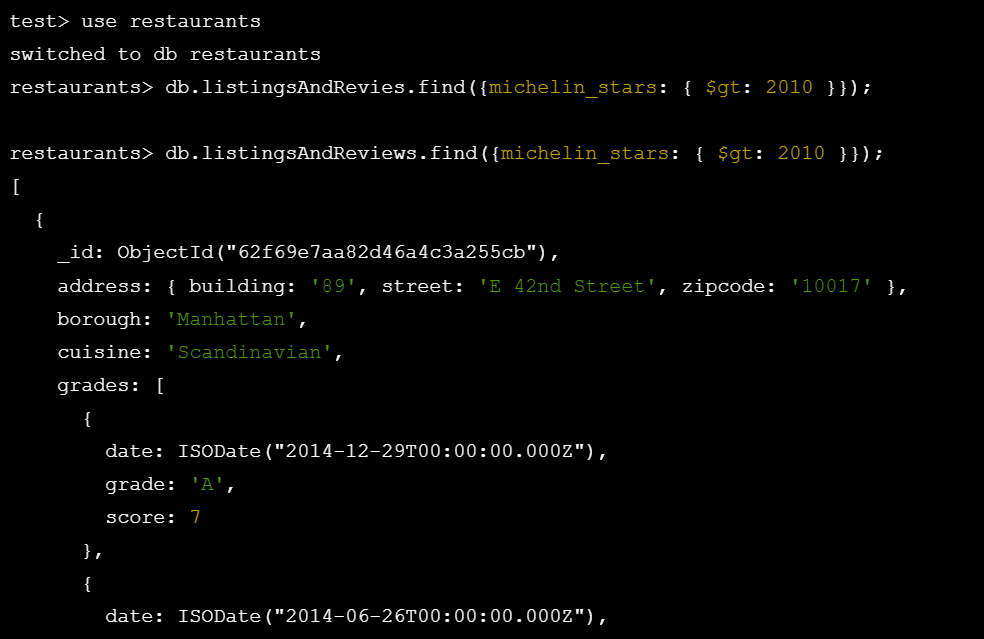
In the same listingsAndReviews collection, search for restaurants that have a michelin\_stars array field containing values that are greater than 2015 or less than 2010.

Checkpoint 3 Passed

Hint

To query on an array field based on multiple conditions, you can use the following syntax:

db.<collection>.find({ <field>: { <operator>: <value>, <operator2>: <value2>, … } })



**Querying for all conditions with $elemMatch**

More often than not, when we specify multiple query conditions for an array field, we’ll want to match at least one array element that meets *all* the filter criteria. We can accomplish this by using another operator, $elemMatch.

The [$elemMatch](https://www.mongodb.com/docs/manual/reference/operator/query/elemMatch/?utm_campaign=academia_partners&utm_source=codecademy&utm_medium=referral) operator is used in queries to specify multiple criteria on the elements of an array field, such that any returned documents have at least one array element that satisfies all the specified criteria.

Let’s reconsider our previous example about professional tennis players to see $elemMatch in action. Recall that documents from this collection have the following structure:

{  
  \_id: ObjectId(...),  
  name: "Serena Williams",  
  country: "United States",  
  wimbledon\_singles\_wins: [2002, 2003, 2009, 2010, 2012, 2015, 2016]  
}

Imagine that we want to search our collection again, this time, for any athletes who have won the Wimbledon Singles Championship in the current millennium, between the years 2000 and 2019.

Our query would look something like this:

db.tennis\_players.find({   
 wimbledon\_singles\_wins: { $elemMatch: { $gte: 2000, $lt: 2020 } }   
})

This would only return documents whose wimbledon\_singles\_wins field is an array containing at least one element that is both greater than or equal to 2000 *and* less than 2020. Our resulting cursor might contain the following documents:

{  
  \_id: ObjectId(...),  
  name: "Pete Sampras",  
  country: "United States",  
  wimbledon\_singles\_wins: [1993, 1994, 1995, 1997, 1998, 1999, 2000]  
},  
{  
  \_id: ObjectId(...),  
  name: "Serena Williams",  
  country: "United States",  
  wimbledon\_singles\_wins: [2002, 2003, 2009, 2010, 2012, 2015, 2016]  
},  
{  
  \_id: ObjectId(...),  
  name: "Roger Federer",  
  country: "Switzerland",  
  wimbledon\_singles\_wins: [2003, 2004, 2005, 2006, 2007, 2009, 2012, 2017]  
}

Note that while any matching documents *must* contain at least one value in the wimbledon\_singles\_wins array that is between the range of 2000 and 2020, this array can also include values that fall outside that range.

Let’s practice querying arrays using the $elemMatch operator!

**Instructions**

**1.**

Connect to the restaurants database. Then, search the listingsAndReviews collection for any restaurants that were awarded a Michelin star between the years 2005 and 2010, inclusive.

Checkpoint 2 Passed

Hint

To match at least one array element that meets *all* the filter criteria in your query, you can use the following syntax:



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