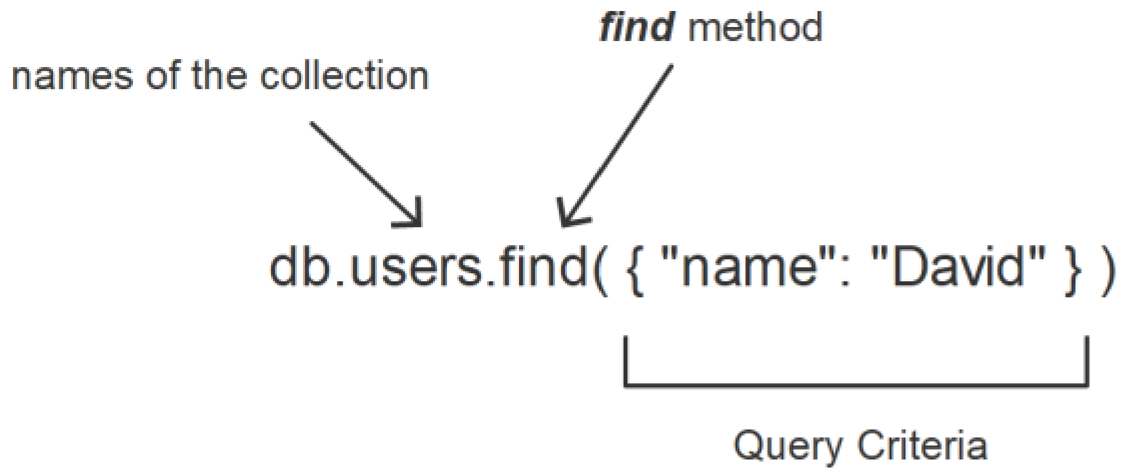
Querying Documents

Objectives

- By the end of this session, you should be able to:
  - Prepare and execute queries in MongoDB
  - Find documents from a collection
  - Limit the fields in the output
  - Usage of conditional and logical operators in a query
  - Usage of regular expressions in query
  - Limit, skip, sort records in the result set

MongoDB query structure

- MongoDB queries are based on JSON documents in which you write your criteria in the form of valid documents.
- The following diagram is an example of a simple MongoDB query that finds all the documents where the name field contains the value David:



```
db.users.find( { "name": "David" } )
```

MongoDB Basic Queries

- **Finding Documents**
  - The most basic query in MongoDB is performed with the find() function on the collection. When this function is executed without any argument, it returns all the documents in a collection.
- **Finding specific documents**
  - To return only specific documents, a condition can be provided to the find() function. When this is done, the find() function evaluates it against each and every document in the collection and returns the documents that match the condition.
- **Formatted output**
  - pretty() function can be used to print a well-formatted result
- **Using findOne()**
  - This function is very useful when you are looking to isolate a specific document. Need to pass the condition , which will be used against all document and returns only a matching document.
  - Note:If this query is executed without any condition and matches all the documents in the collection but will return only first one.

Choosing the Fields for the Output

- In MongoDB queries, you can either include or exclude specific fields from the result. This technique is called projection.
- Projection is expressed as a second argument to the find() or findOne() functions. In the projection expression, you can explicitly exclude a field by setting it to 0 or include one by setting it to 1.

Finding the Distinct Fields

- The distinct() function is used to get the distinct or unique values of a field with or without query criteria.
- The distinct() function can also be used along with a query condition.

Counting Documents

- count()
- countDocuments()
- estimatedDocumentCount()

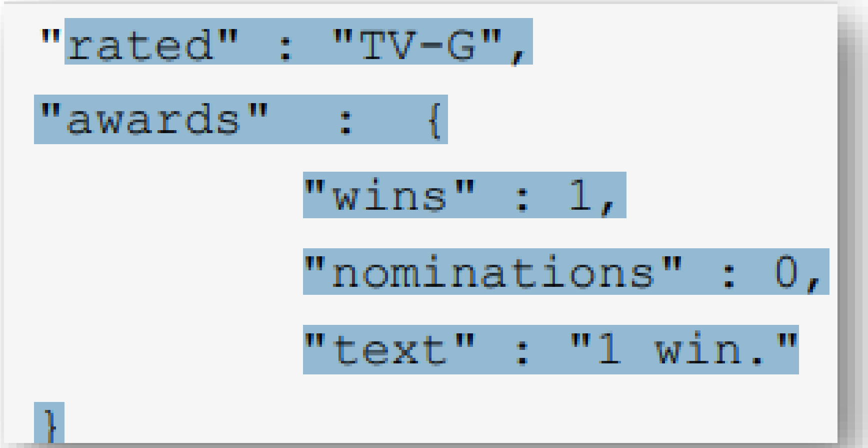
- **Conditional Operators**
  - Equals ($eq) and Not Equal To ($ne)
  - Greater Than ($gt) and Greater Than or Equal To ($gte)
  - Less Than ($lt) and Less Than or Equal To ($lte)
  - In ($in) and Not In ($nin)
- **Logical Operators**
  - $and operator, $or Operator
  - $nor Operator, $not Operator
- **Regular Expressions**
  - $regex
  - the caret (^) operator
  - the dollar ($) operator
  - Case-Insensitive Search: $options argument with a value of i, where i stands for case-insensitive

Query Arrays Documents

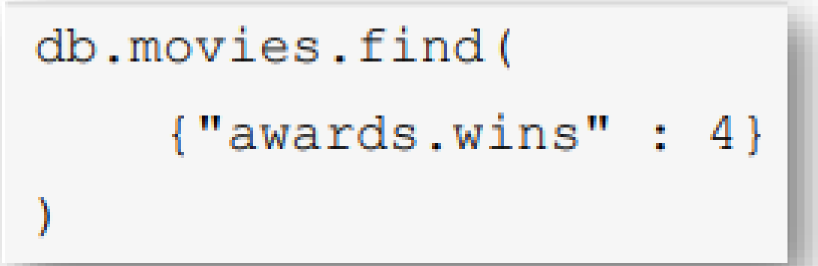
- **Finding an Array by an Element :** when an array field is queried using a value, all those documents are returned where the array field contains at least one element that satisfies the query.
- **Finding an Array by an Array:** An array fields can also be searched using array values. However, when you search an array field using an array value, the elements and their order must match.
- **Searching an Array with the $all Operator :** The $all operator finds all those documents where the value of the field contains all the elements, irrespective of their order or size.
- **Projecting Matching Elements Using ($) :** You can search an array by an element value and use projection to exclude all but the first matching element of the array using the $ operator.
- **Projecting Matching Elements by their Index Position ($slice):** It is used to limit the array elements based on their index position. This operator can be used with any array field, irrespective of the field being queried or not.
  - If $slice operator passed with negative value, It will return elements from the end.
  - The $slice operator can also be passed with two arguments, where the first argument indicates the number of elements to be skipped and the second one indicates the number of elements to be returned.
  - If the value of skip is negative, the counting starts from the end.

Query Nested Documents

- **Querying Nested Objects**
  - The fields that have other objects as their values can be searched using the complete object as a value. In the movies collection, there is a field named awards whose value is a nested object.
  - When nested object fields are searched with object values, there must be an exact match.
- **Querying Nested Object Fields**
  - Dot notation can be used to search nested objects by providing the values of its fields.
  - You can search by multiple fields and use any of the conditional or logical query operators.



```
"rated" : "TV-G",
"awards"  :  {
          "wins" : 1,
          "nominations" : 0,
          "text" : "1 win."
}
```



```
db.movies.find(
     {"awards.wins" : 4}
)
```

Limiting, Skipping, and Sorting Documents

- **Limiting the Result : The** limit() function accepts an integer and returns the same number of records, if available. For queries returning smaller records, a negative size limit is considered equivalent to the limit of a positive number.
- **Skipping Documents:** The MongoDB cursor provides the skip() function, which accepts an integer and skips the specified number of documents from the cursor, returning the rest. IJSON is an easy-to-understand key-value pair format to describe data.
- **Sorting Documents:** The MongoDB cursor provides a sort() function that accepts an argument of the document type, where the document defines a sort order for specific fields.BSON is designed to be more efficient in space than standard JSON.

Summary

- In this session, you learned about:
  - A detailed study of the structure of MongoDB queries and how different they are from SQL queries.
  - Then, we implemented these queries to find and count the documents and limit the number of fields returned in the result using various examples.
  - We also learned about the various conditional and logical operators and practiced using them in combination to notice the difference in results.
  - We then learned how to provide a text pattern using regular expressions to filter our search results and covered how to query arrays and nested objects and include their specific fields in the results.
  - Finally, we learned how to paginate large result sets by using limiting, sorting, and skipping on the documents in the result.
- In the next chapter, we will learn how to insert, update, and delete documents from MongoDB collections.