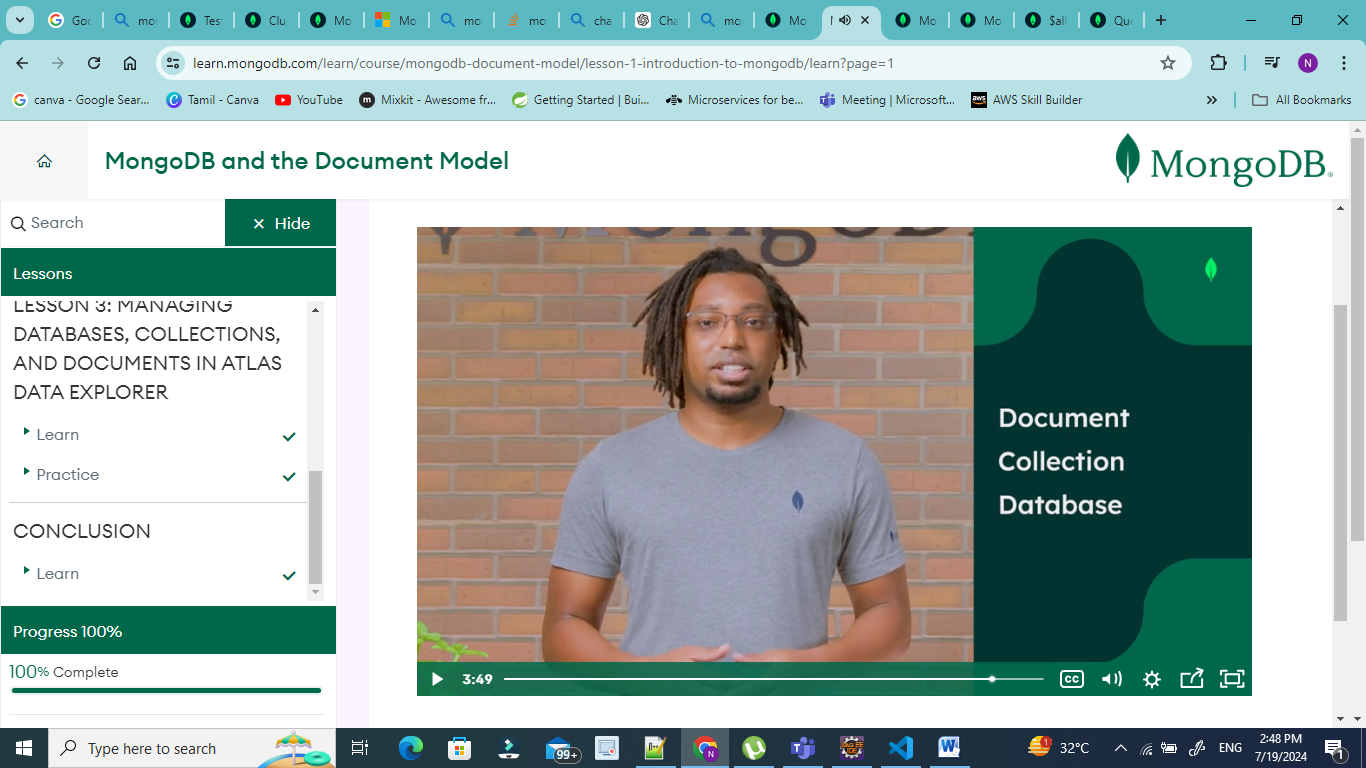
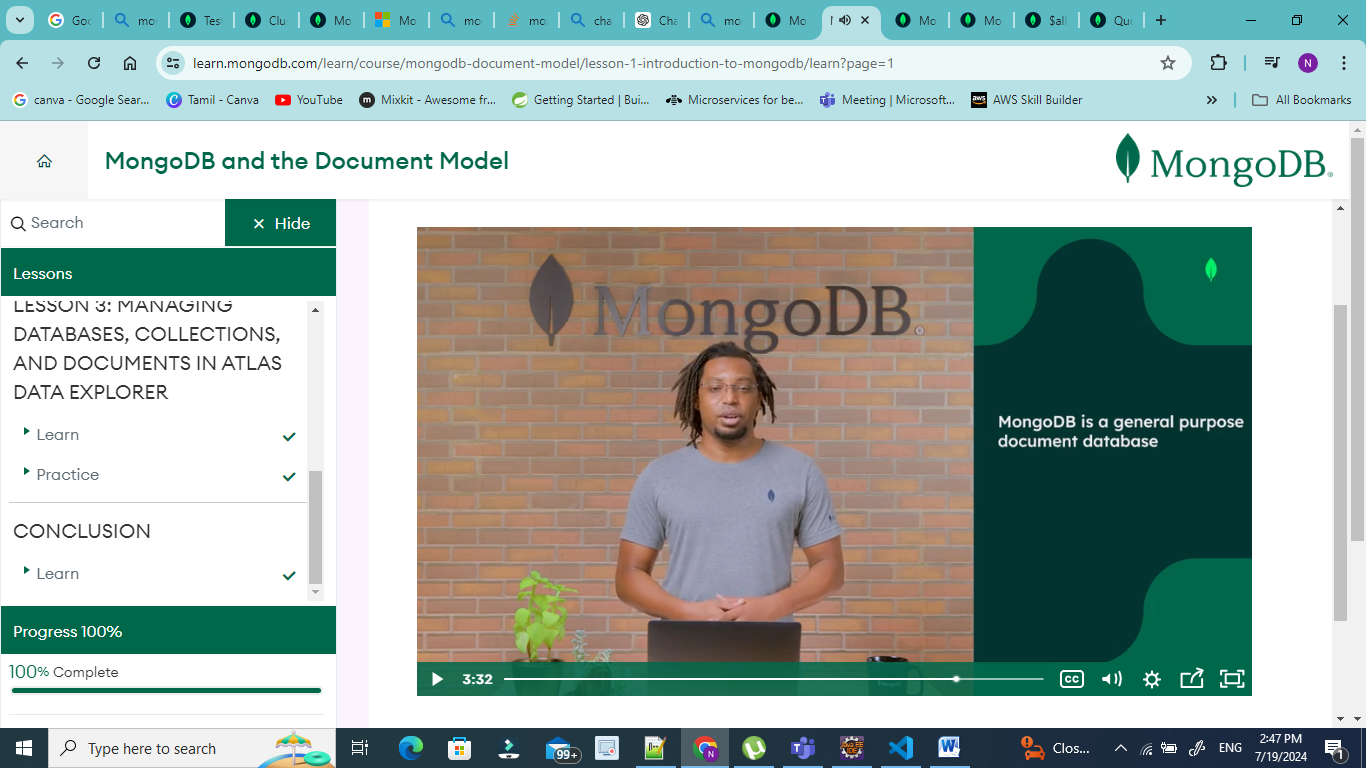
# Overview of MongoDB and the Document Model

# BSON Types

[BSON](https://www.mongodb.com/docs/manual/reference/glossary/#std-term-BSON) is a binary serialization format used to store documents and make remote procedure calls in MongoDB. The BSON specification is located at [bsonspec.org](http://bsonspec.org/).

Each BSON type has both integer and string identifiers as listed in the following table:

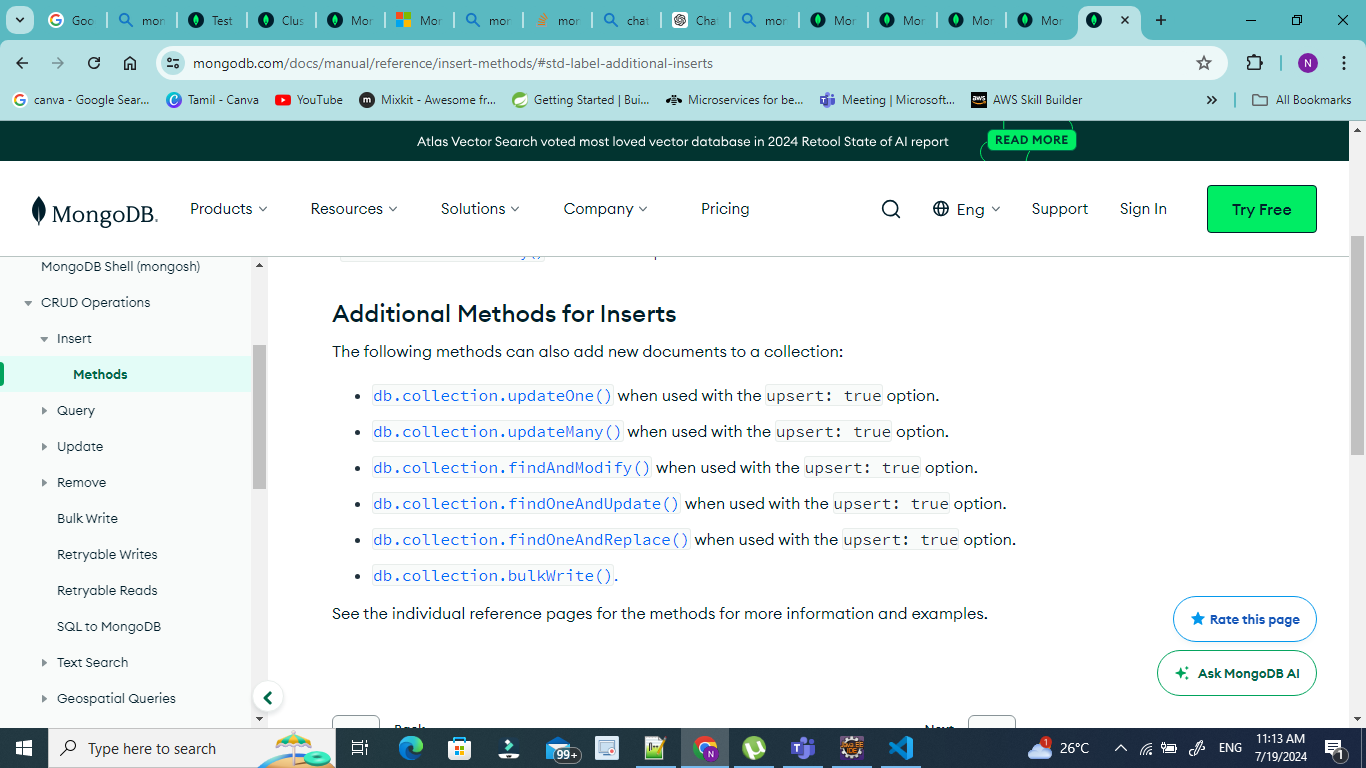
| **Type** | **Number** | **Alias** | **Notes** |
| --- | --- | --- | --- |
| Double | 1 | "double" |  |
| String | 2 | "string" |  |
| Object | 3 | "object" |  |
| Array | 4 | "array" |  |
| Binary data | 5 | "binData" |  |
| Undefined | 6 | "undefined" | Deprecated. |
| ObjectId | 7 | "objectId" |  |
| Boolean | 8 | "bool" |  |
| Date | 9 | "date" |  |
| Null | 10 | "null" |  |
| Regular Expression | 11 | "regex" |  |
| DBPointer | 12 | "dbPointer" | Deprecated. |
| JavaScript | 13 | "javascript" |  |
| Symbol | 14 | "symbol" | Deprecated. |
| 32-bit integer | 16 | "int" |  |
| Timestamp | 17 | "timestamp" |  |
| 64-bit integer | 18 | "long" |  |
| Decimal128 | 19 | "decimal" |  |
| Min key | -1 | "minKey" |  |
| Max key | 127 | "maxKey" |  |



In this unit, you learned about the relationship between MongoDB and Atlas. MongoDB is a general-purpose database that can be used for a variety of use cases. It’s part of Atlas, the developer data platform. You also learned about the MongoDB document model. Specifically, you learned about these key features:

* Data is organized into documents, collections, and databases.
* Documents are stored in BSON, which supports a large range of data types, including all JSON data types, dates, numbers, and ObjectIds.
* Every document requires an \_id field, which acts as a primary key or unique identifier. If an inserted document doesn’t have an \_id field, MongoDB automatically generates one.
* MongoDB has a flexible schema, which means that documents with different structures can be stored in the same collection.

Finally, you learned how to use Atlas Data Explorer to create a database, collection, and a document and begin storing your own data.

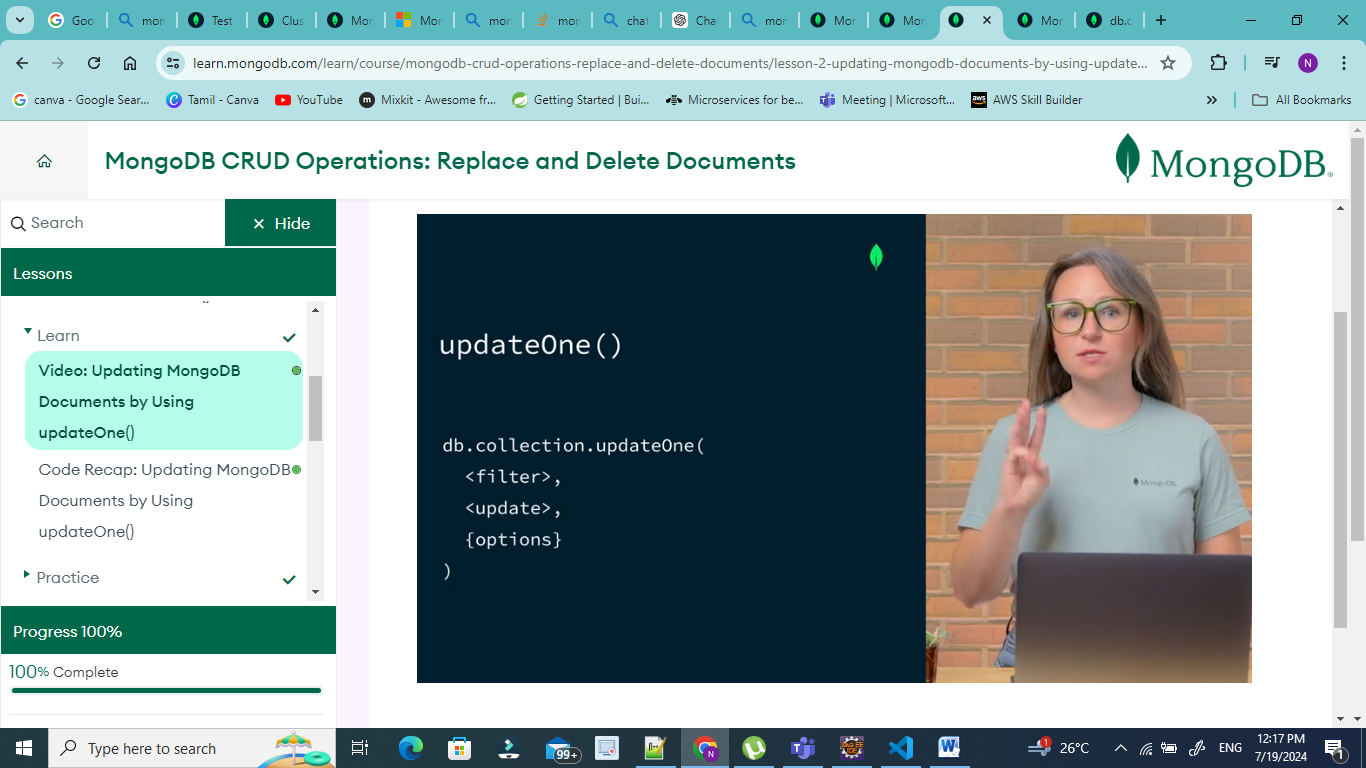


The following methods can also add new documents to a collection:

* [db.collection.updateOne()](https://www.mongodb.com/docs/manual/reference/method/db.collection.updateOne/#mongodb-method-db.collection.updateOne) when used with the upsert: true option.
* [db.collection.updateMany()](https://www.mongodb.com/docs/manual/reference/method/db.collection.updateMany/#mongodb-method-db.collection.updateMany) when used with the upsert: true option.
* [db.collection.findAndModify()](https://www.mongodb.com/docs/manual/reference/method/db.collection.findAndModify/#mongodb-method-db.collection.findAndModify) when used with the upsert: true option.
* [db.collection.findOneAndUpdate()](https://www.mongodb.com/docs/manual/reference/method/db.collection.findOneAndUpdate/#mongodb-method-db.collection.findOneAndUpdate) when used with the upsert: true option.
* [db.collection.findOneAndReplace()](https://www.mongodb.com/docs/manual/reference/method/db.collection.findOneAndReplace/#mongodb-method-db.collection.findOneAndReplace) when used with the upsert: true option.
* [db.collection.bulkWrite().](https://www.mongodb.com/docs/manual/reference/method/db.collection.bulkWrite/#mongodb-method-db.collection.bulkWrite)

With the update operations, the aggregation pipeline can consist of the following stages:

* [$addFields](https://www.mongodb.com/docs/manual/reference/operator/aggregation/addFields/#mongodb-pipeline-pipe.-addFields)
* [$set](https://www.mongodb.com/docs/manual/reference/operator/aggregation/set/#mongodb-pipeline-pipe.-set)
* [$project](https://www.mongodb.com/docs/manual/reference/operator/aggregation/project/#mongodb-pipeline-pipe.-project)
* [$unset](https://www.mongodb.com/docs/manual/reference/operator/aggregation/unset/#mongodb-pipeline-pipe.-unset)
* [$replaceRoot](https://www.mongodb.com/docs/manual/reference/operator/aggregation/replaceRoot/#mongodb-pipeline-pipe.-replaceRoot)
* [$replaceWith](https://www.mongodb.com/docs/manual/reference/operator/aggregation/replaceWith/#mongodb-pipeline-pipe.-replaceWith)



Filter doc for selection criteria, update document for selection criteria, options objects – optional one.

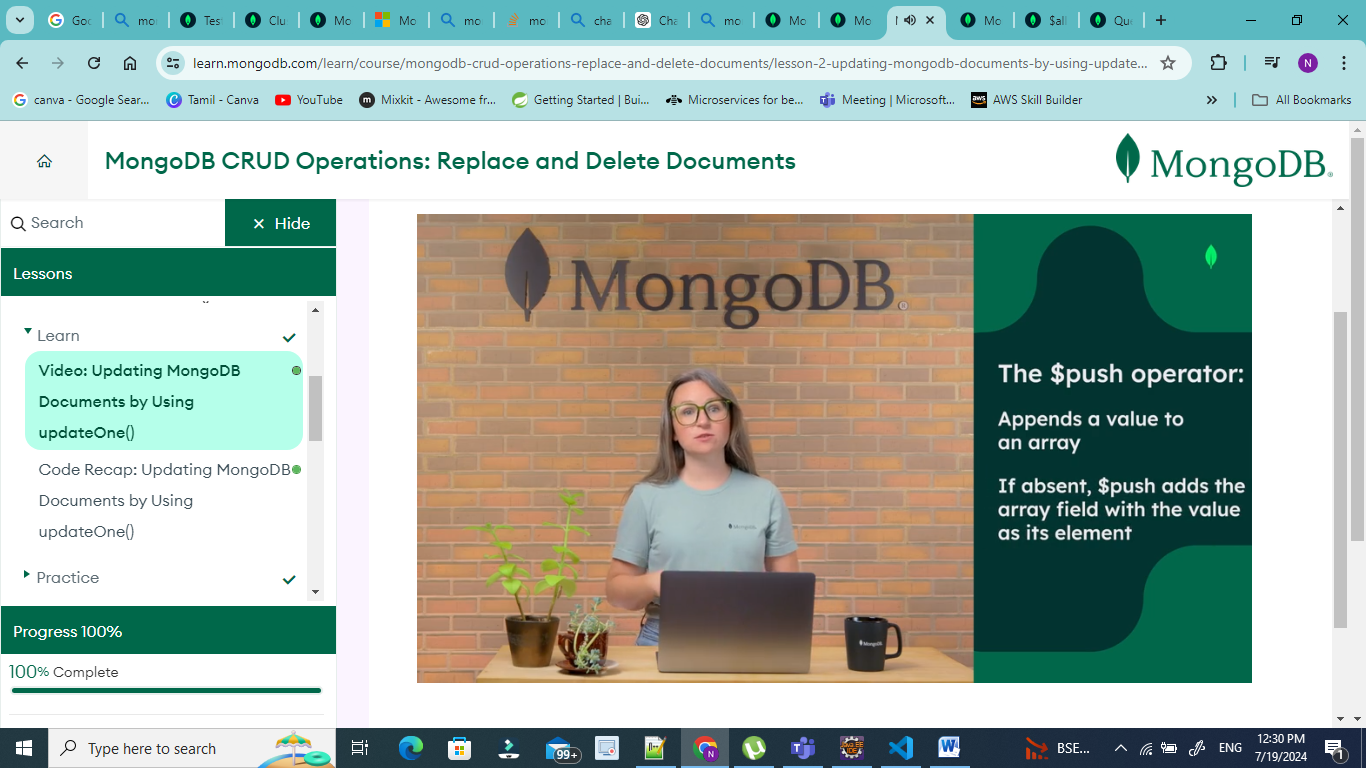
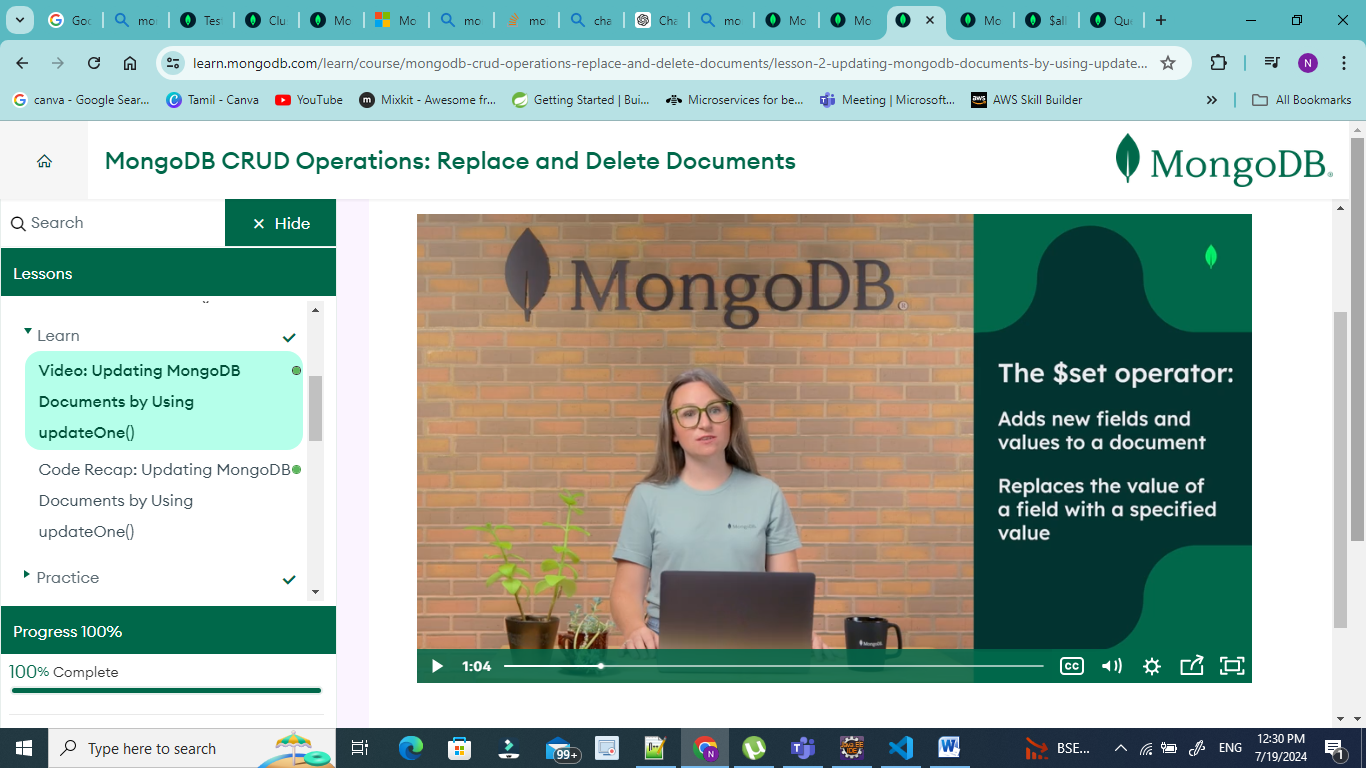
Query on Nested Field with Dot Notation

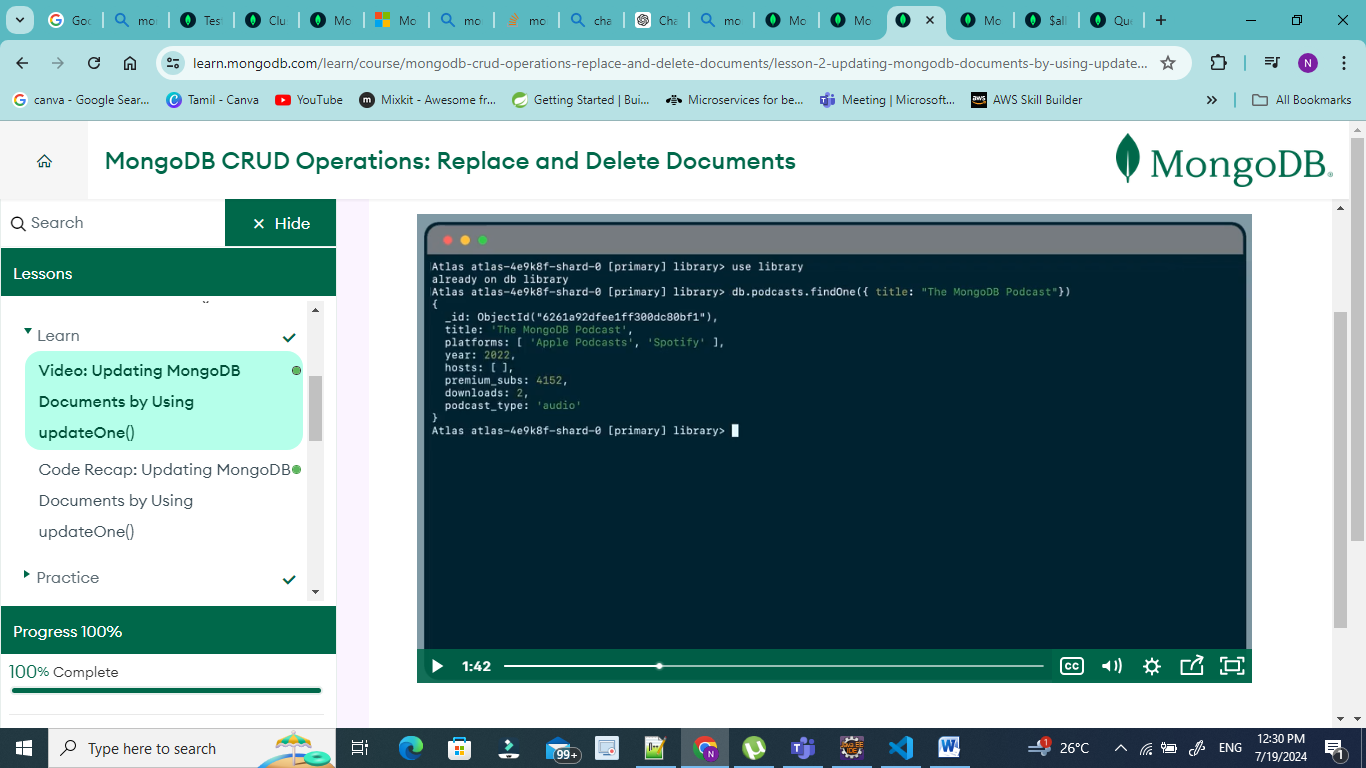
To specify a query condition on fields in an embedded/nested document, use [dot notation](https://www.mongodb.com/docs/manual/reference/glossary/#std-term-dot-notation) ("field.nestedField").

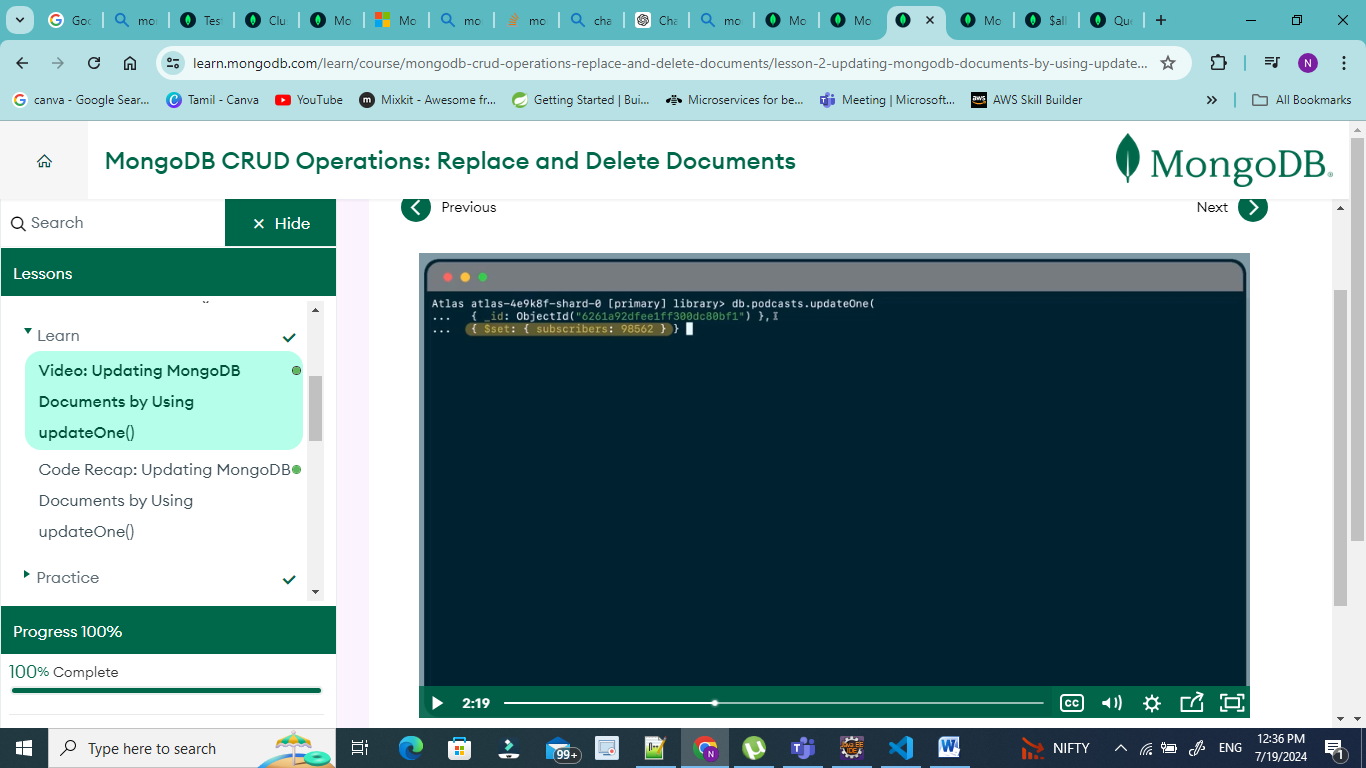
|  |
| --- |
| const cursor = db.collection(**'inventory'**).find({ |
| size: { h: 14, w: 21, uom: **'cm'** } |
| }); |

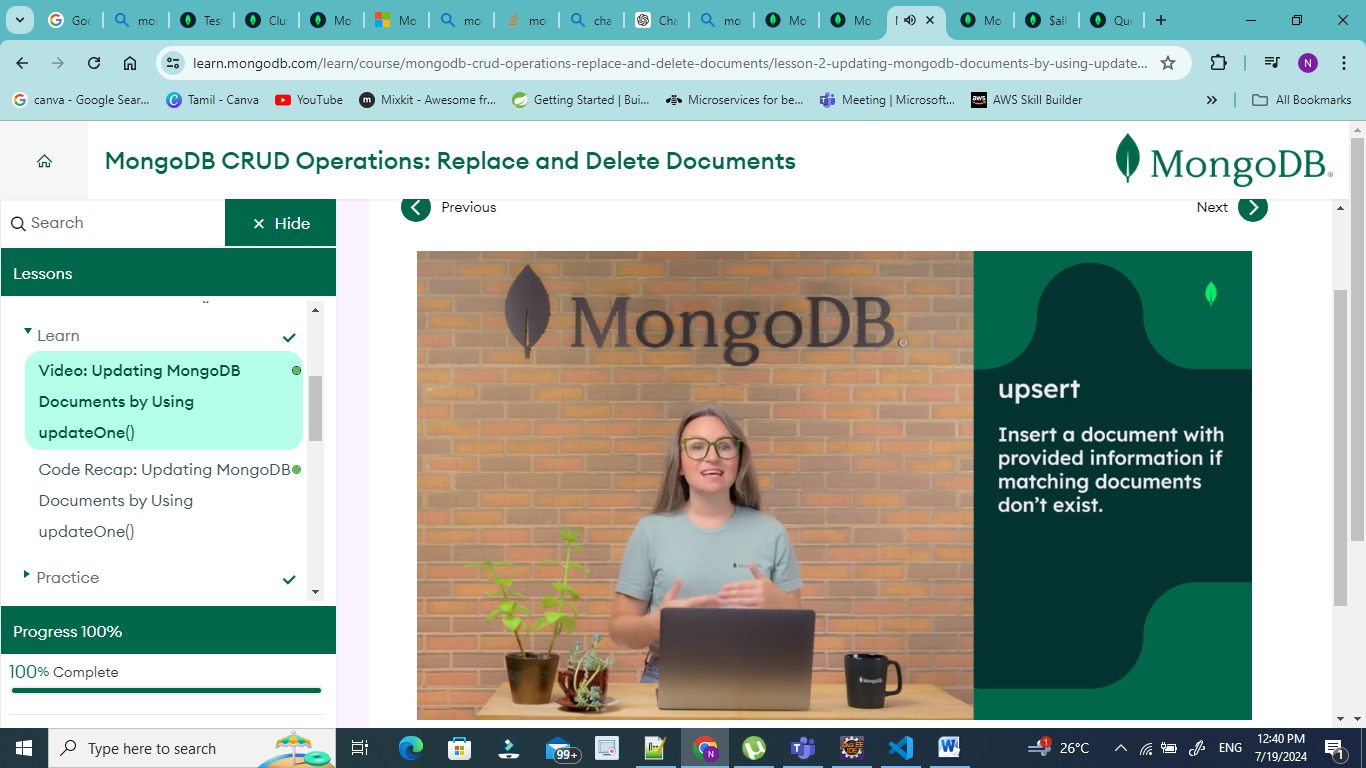
Unpredictable results:

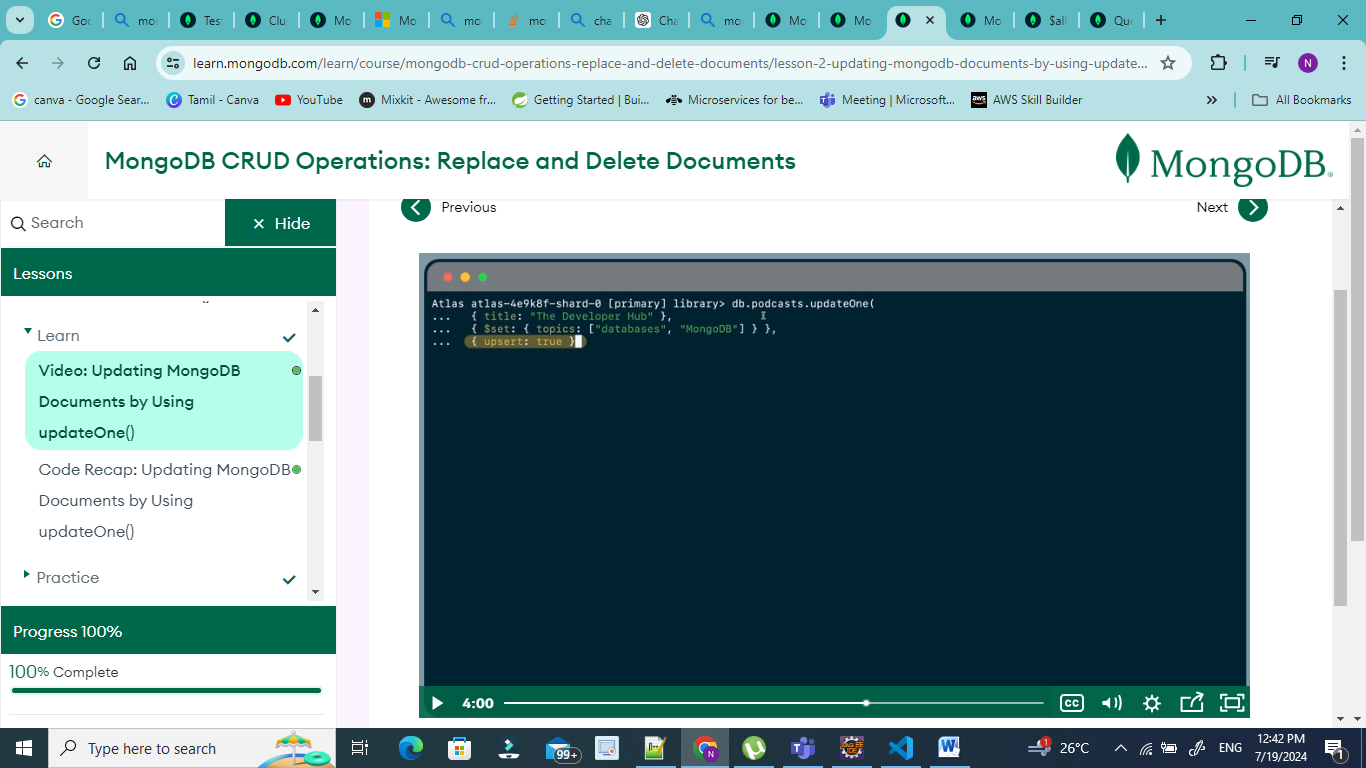
|  |
| --- |
| size: { w: 21, h: 14, uom: **'cm'** } |
|  |

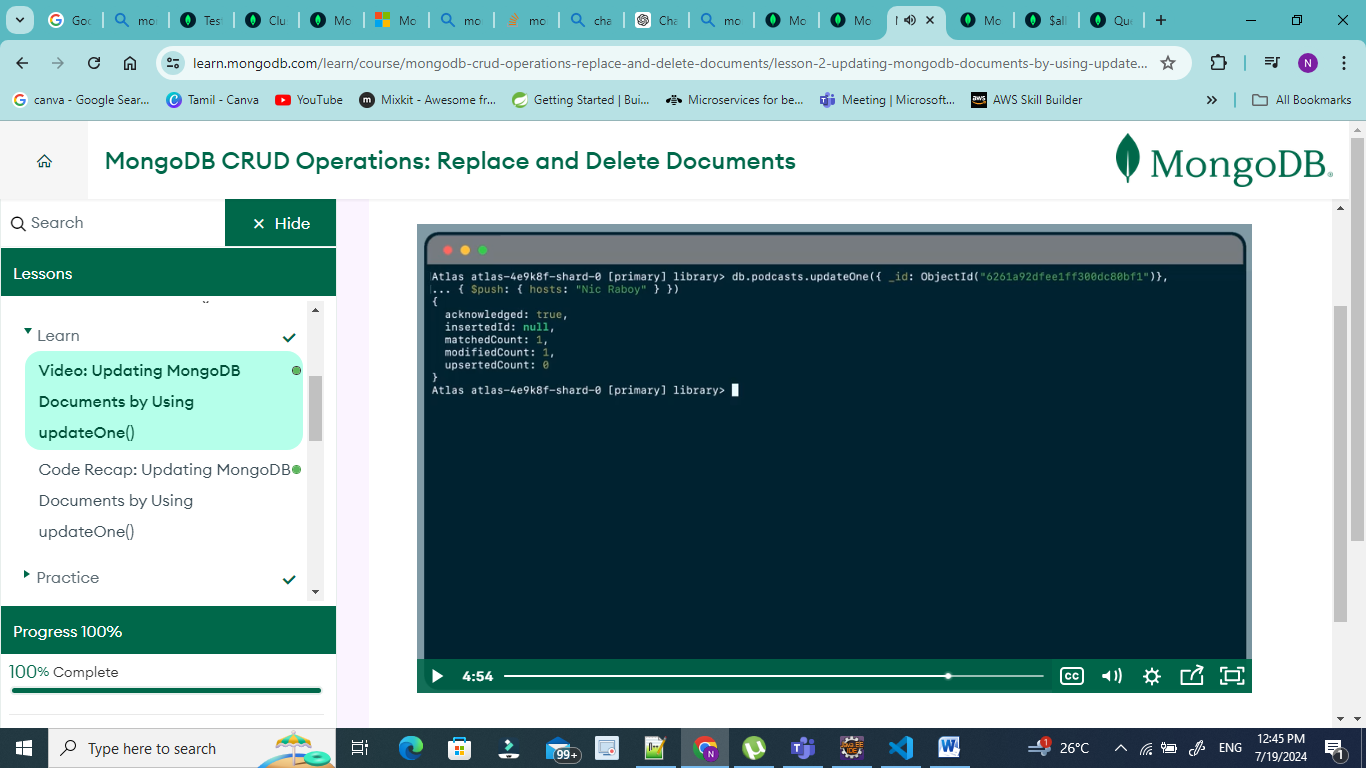


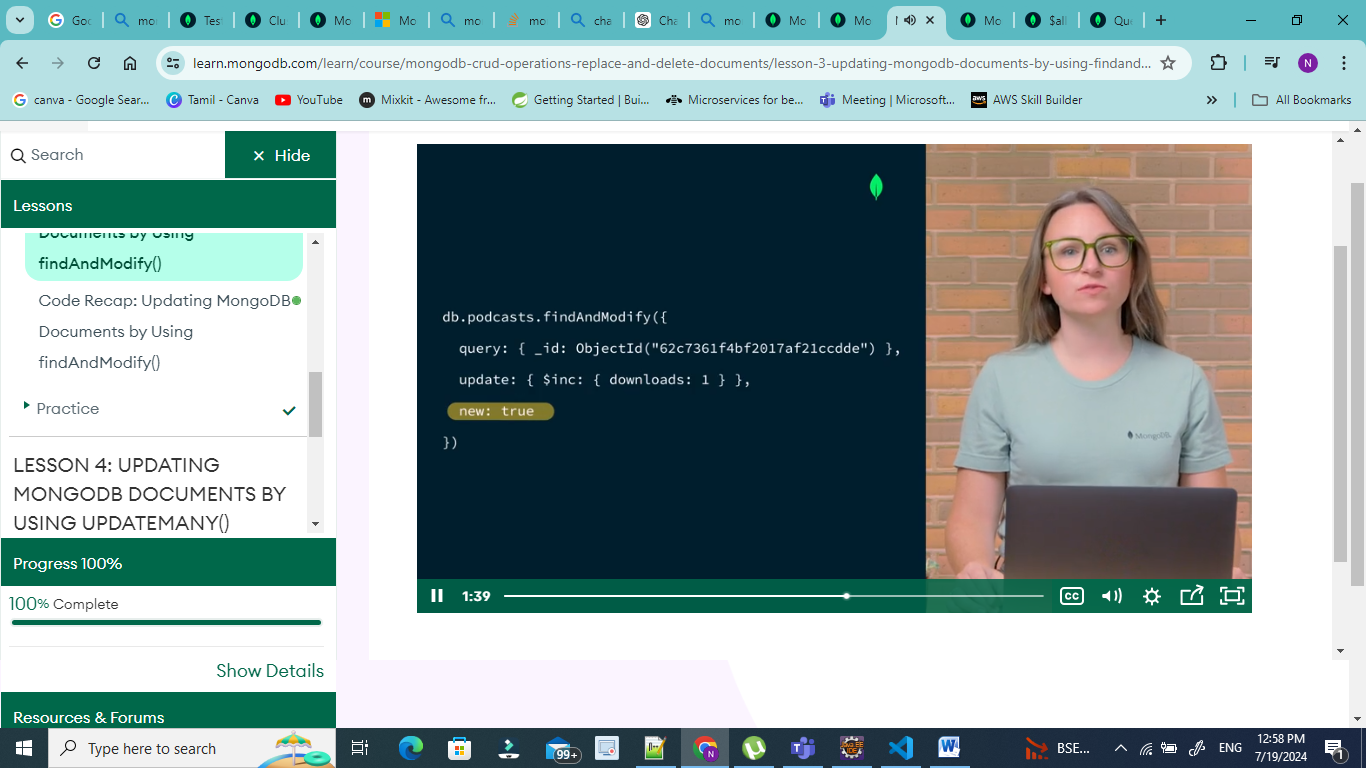
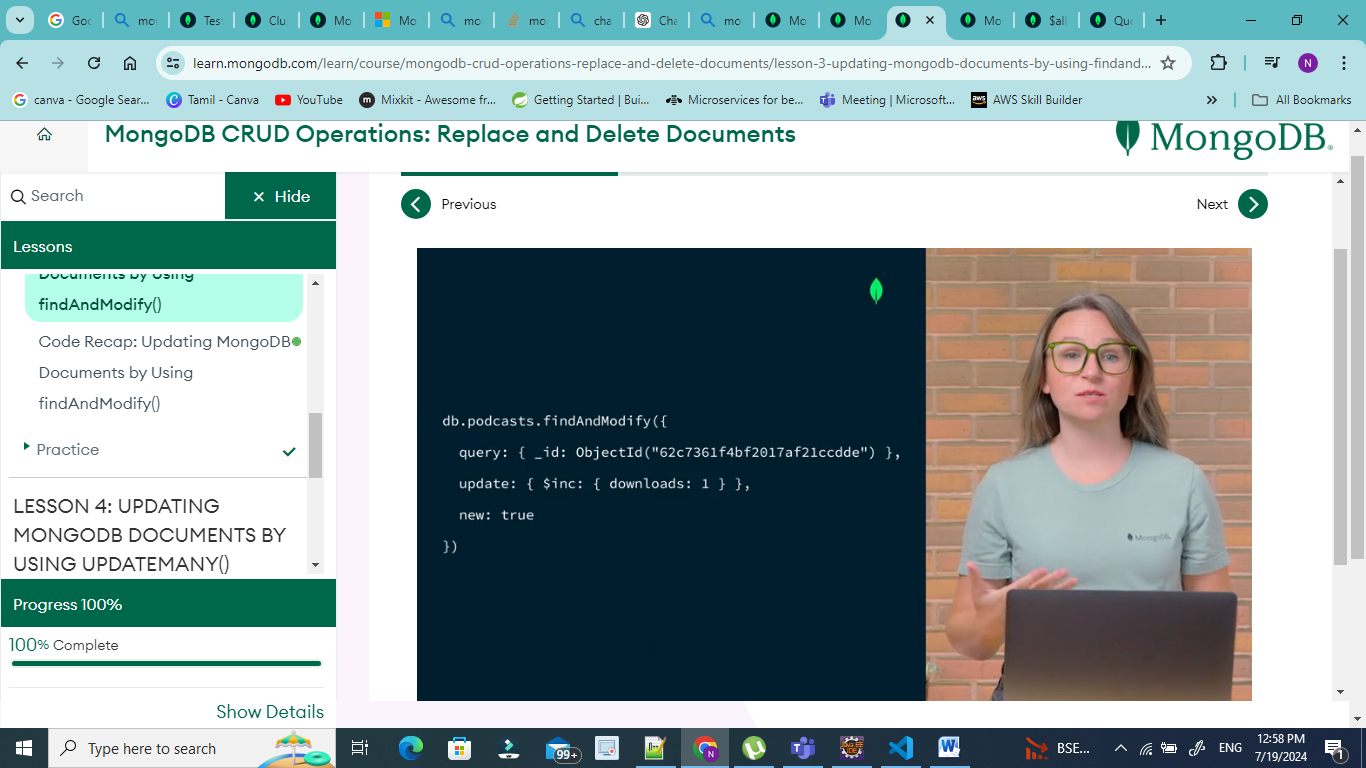


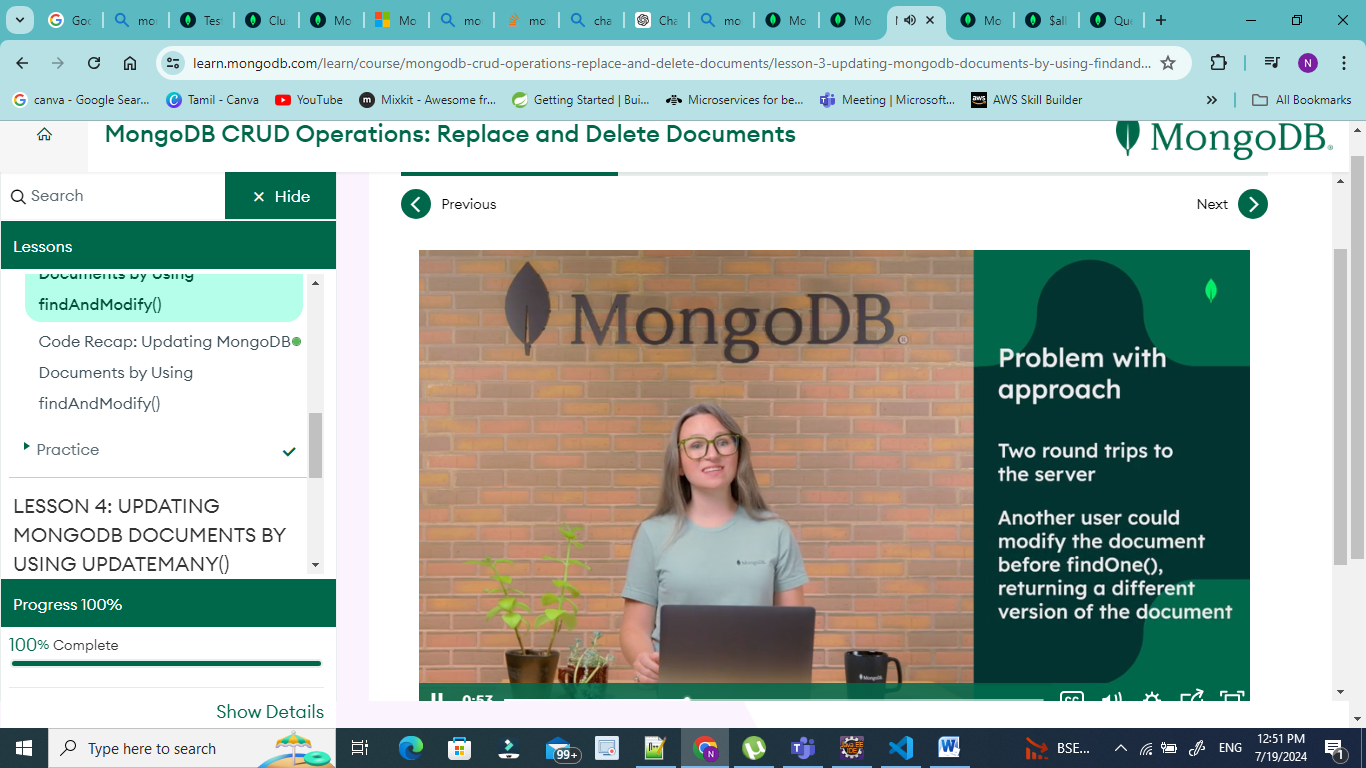
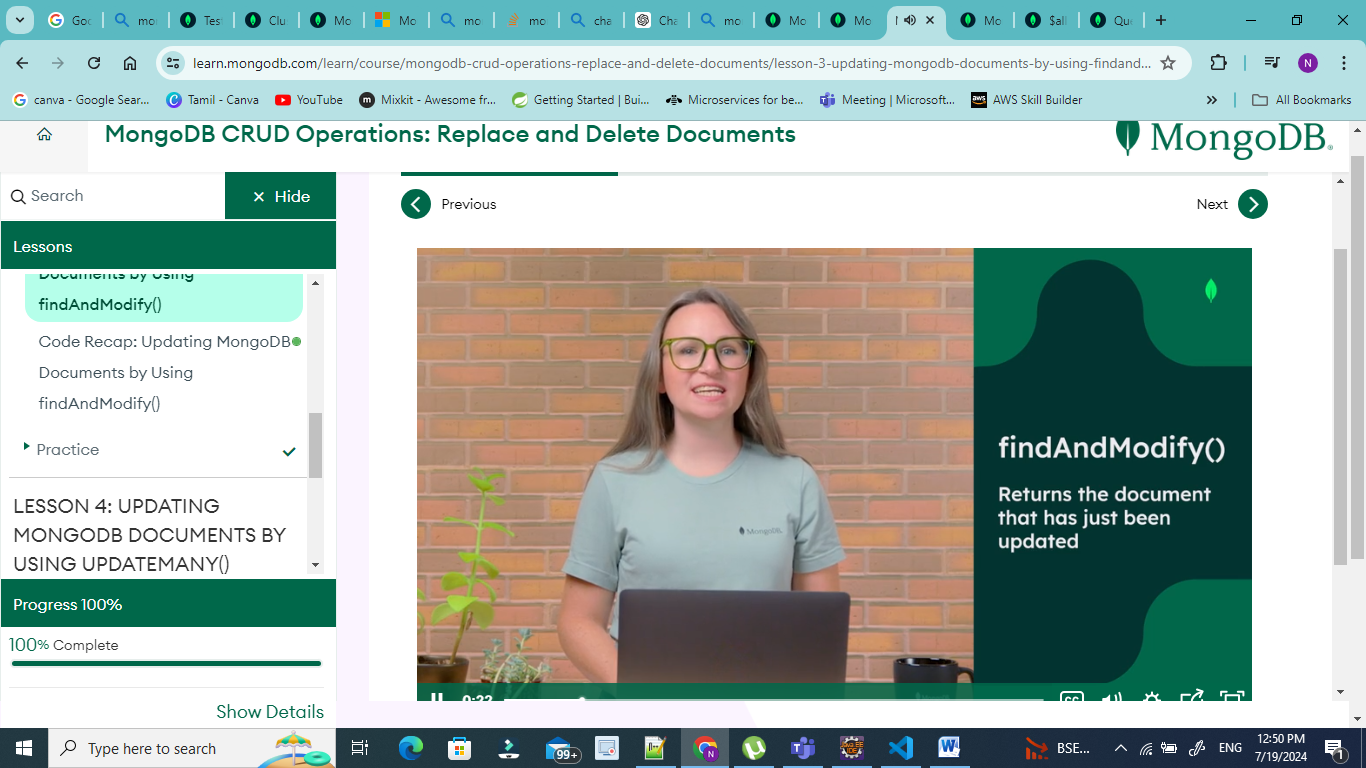
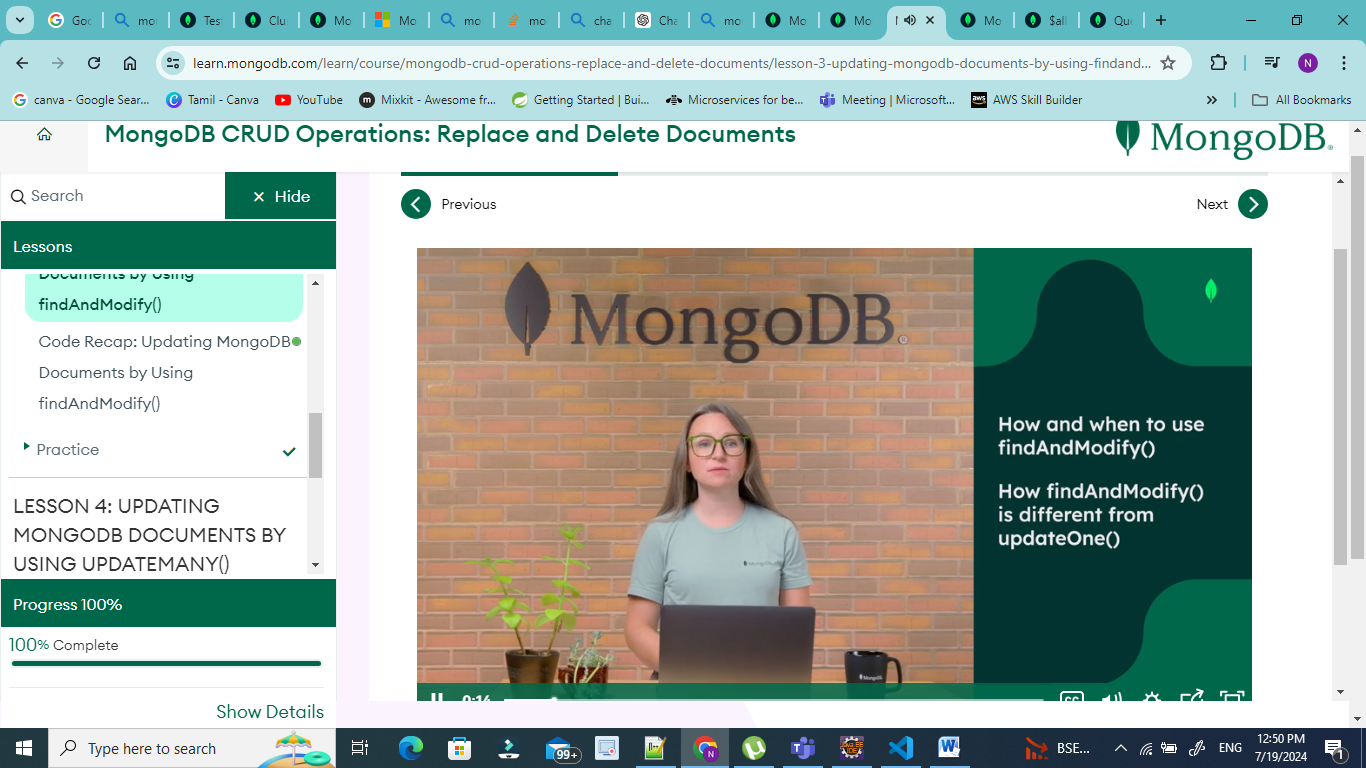




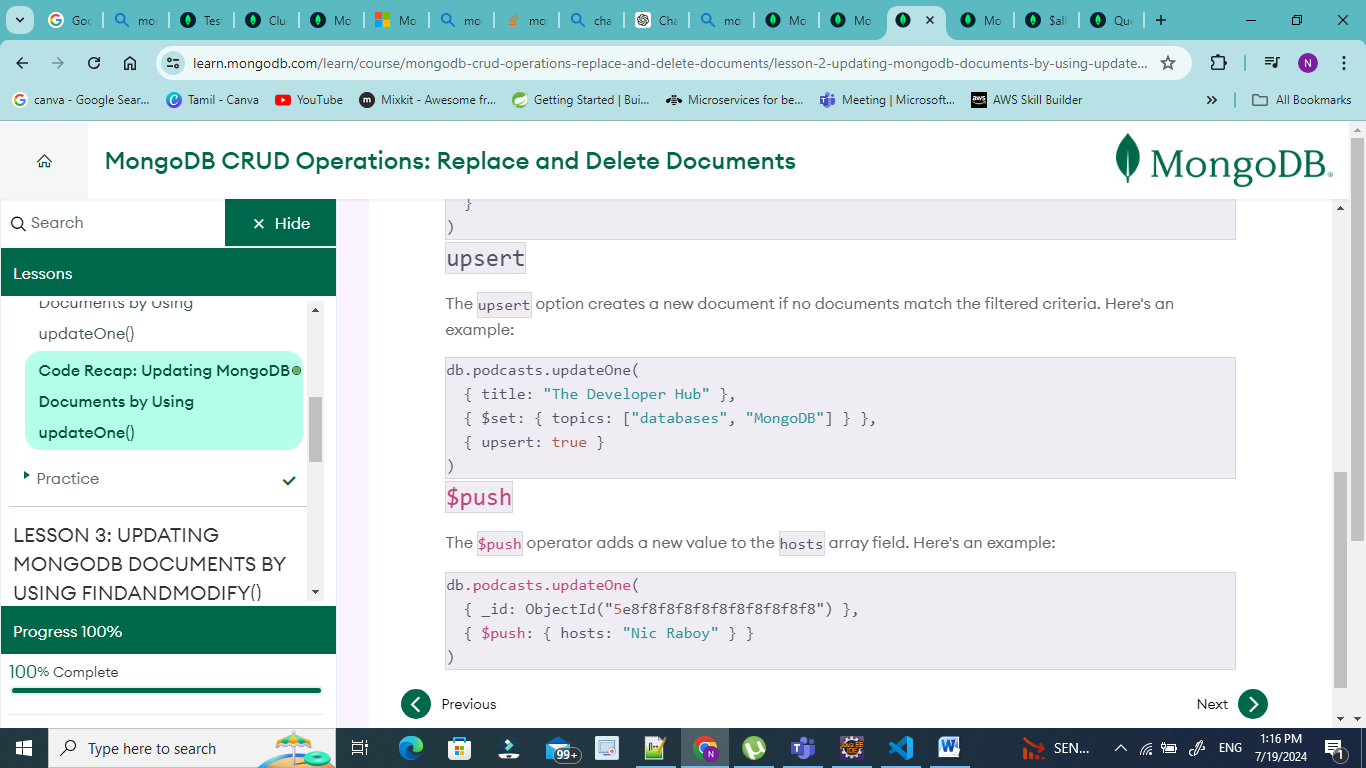
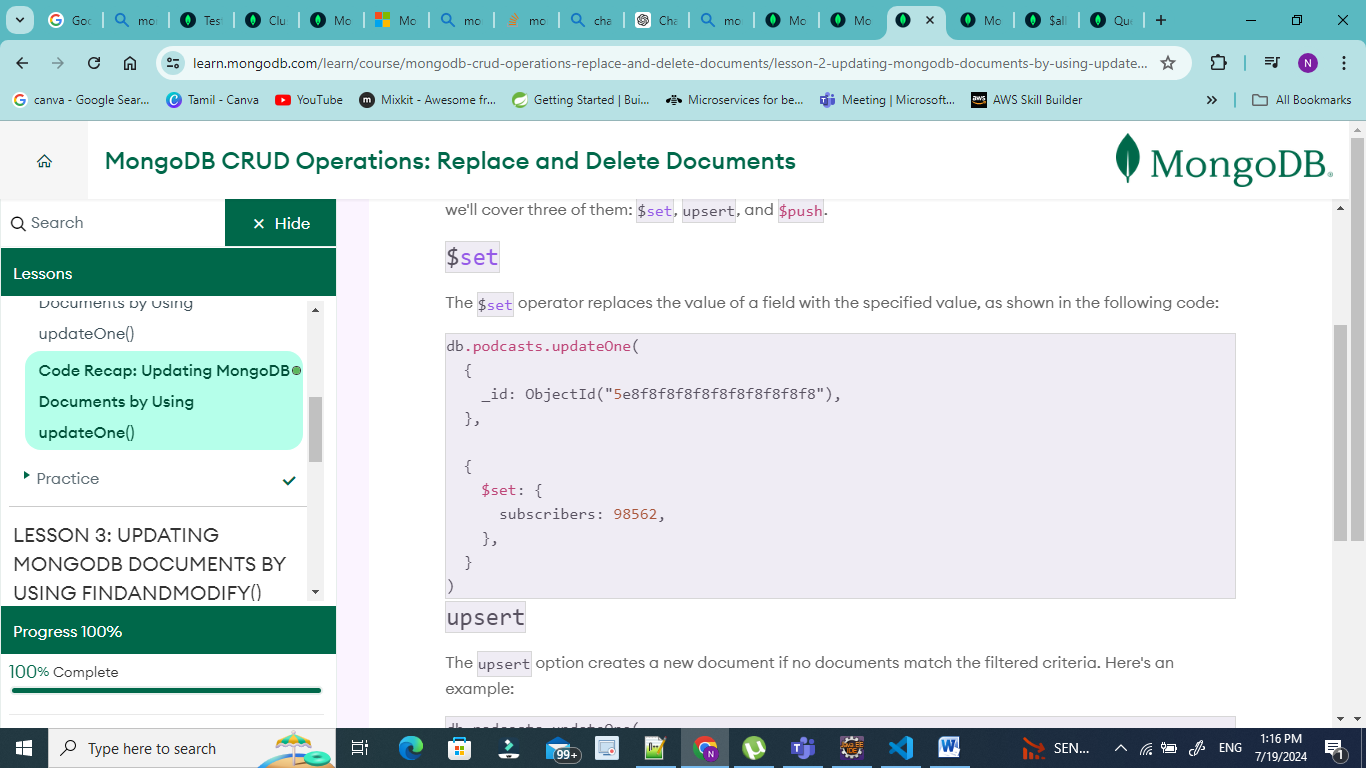
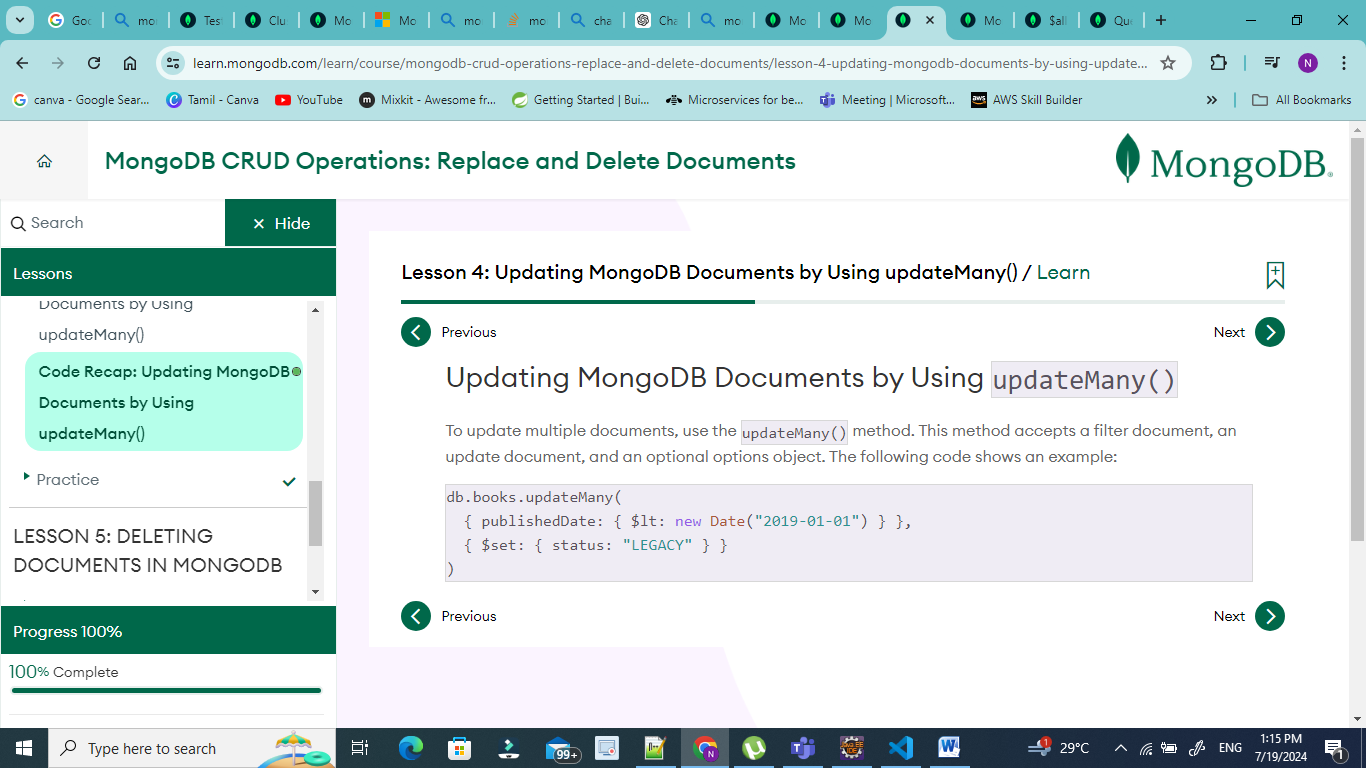








The findAndModify() method is used to find and replace a single document in MongoDB. It accepts a filter document, a replacement document, and an optional options object. The following code shows an example:



# The MongoDB Document Model

Review the following code, which demonstrates the structure of a document.

## Document Structure

The values in a document can be any data type, including strings, objects, arrays, booleans, nulls, dates, ObjectIds, and more. Here's the syntax for a MongoDB document, followed by an example:

Syntax:

{

"key": value,

"key": value,

"key" : value

}

Example:

{

"\_id": 1,

"name": "AC3 Phone",

"colors" : ["black", "silver"],

"price" : 200,

"available" : true

}

