Mongodb :

The $set operator replaces the value of a field with the specified value

The upsert option creates a new document if no documents match the filtered criteria

The $push operator adds a new value to an array field

The $each  modifier to add multiple elements to the array.

* The updateMany() method accepts a filter, an update, and an optional options object.

MongoDB CRUD Operations: Replace and Delete Documents

In this unit, you learned how to modify query results with MongoDB. Specifically, you:

* Replaced a single document by using db.collection.replaceOne().
* Updated a field value by using the $set update operator in db.collection.updateOne().
* Added a value to an array by using the $push update operator in db.collection.updateOne().
* Added a new field value to a document by using the upsert option in db.collection.updateOne().
* Found and modified a document by using db.collection.findAndModify().
* Updated multiple documents by using db.collection.updateMany().
* Deleted a document by using db.collection.deleteOne().
* Return the data on the three most recent sales made from the London store that included one or more of the following items: a laptop, a backpack or printer paper.(*Forgot the command? Check the hints below!*)

db.sales.find({ "items.name": { $in: ["laptop", "backpack", "printer paper"] }, "storeLocation": "London", }).sort({ saleDate: -1, }).limit(3)

# MongoDB CRUD Operations: Modifying Query Results

In this unit, you learned how to modify query results with MongoDB. Specifically, you learned how to:

* Return query results in a specified order by using cursor.sort().
* Constrained the number of results returned by using cursor.limit().
* Specified fields to return by adding a projection document parameter in calls to db.collection.find().
* Counted the number of documents that match a query by using db.collection.countDocuments().

# Introduction to MongoDB Aggregation

This section contains key definitions for this lesson, as well as the code for an aggregation pipeline.

## **Definitions**

* **Aggregation**: Collection and summary of data
* **Stage**: One of the built-in methods that can be completed on the data, but does not permanently alter it
* **Aggregation pipeline**: A series of stages completed on the data in order

## **Structure of an Aggregation Pipeline**

db.collection.aggregate([

{

$stage1: {

{ expression1 },

{ expression2 }...

},

$stage2: {

{ expression1 }...

}

}

])

The $match stage filters for documents that match specified conditions

The $group stage groups documents by a group key.

The $sort stage sorts all input documents and returns them to the pipeline in sorted order. We use 1 to represent ascending order, and -1 to represent descending order.

The $limit stage returns only a specified number of records.