MongoDB Assignment 2

Name: **PRIYESHWAR M D**

Mail: **priyesh2664@gamil.com**

Projections in MongoDB

In MongoDB, projections are used to control which fields are returned in the result of a query. By default, find() returns all fields of a document — but you can use a projection to include or exclude specific fields.

What is an Embedded Document?

An embedded document is a document nested inside another document.

Example:

```
{_id: 1, name: "John Doe",
  address: {
    street: "123 Main St",
    city: "New York",
    zip: "10001"
  }}
```

Benefits:

- Data locality: related data is stored together, reducing joins.
- Atomic updates: you can update the embedded document as part of the parent.
- Simplifies reads: fetching one document brings in all related information.

◆ **What is an Array of Embedded Documents?**

An array of embedded documents is a field that contains multiple embedded documents inside an array.

Example:

```
{ _id: 2, name: "Alice",
  orders: [
    { product: "Book", quantity: 2 },
```

{ product: "Pen", quantity: 10 }]}

- orders is an array of embedded documents.

Benefits:

- Useful for lists: tags, orders, comments, etc.
- Eliminates the need for separate collections and joins.

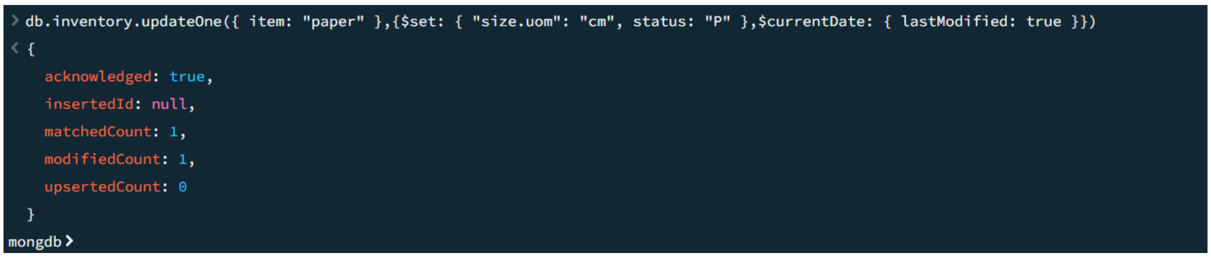
Atomicity and Transactions

In MongoDB, a write operation is atomic on the level of a single document, even if the operation modifies multiple values. When multiple update commands happen in parallel, each individual command ensures that the query condition still matches.

To guarantee that concurrent update commands do not conflict with each other, you can specify the expected current value of a field in the update filter.

Update Inventory

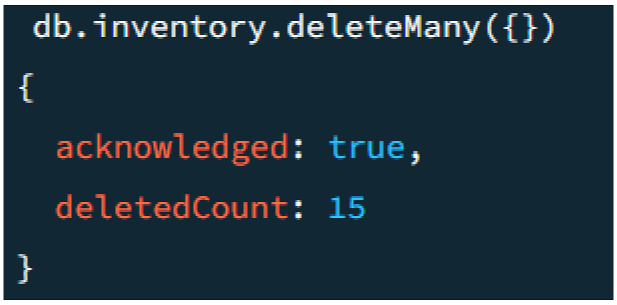
db.inventory.updateOne({ item: "paper" },{$set: { "size.uom": "cm", status: "P" },$currentDate: { lastModified: true }})



```
> db.inventory.updateOne({ item: "paper" },{$set: { "size.uom": "cm", status: "P" },$currentDate: { lastModified: true }})
< {
    acknowledged: true,
    insertedId: null,
    matchedCount: 1,
    modifiedCount: 1,
    upsertedCount: 0
  }
mongdb >
```

Delete Inventory

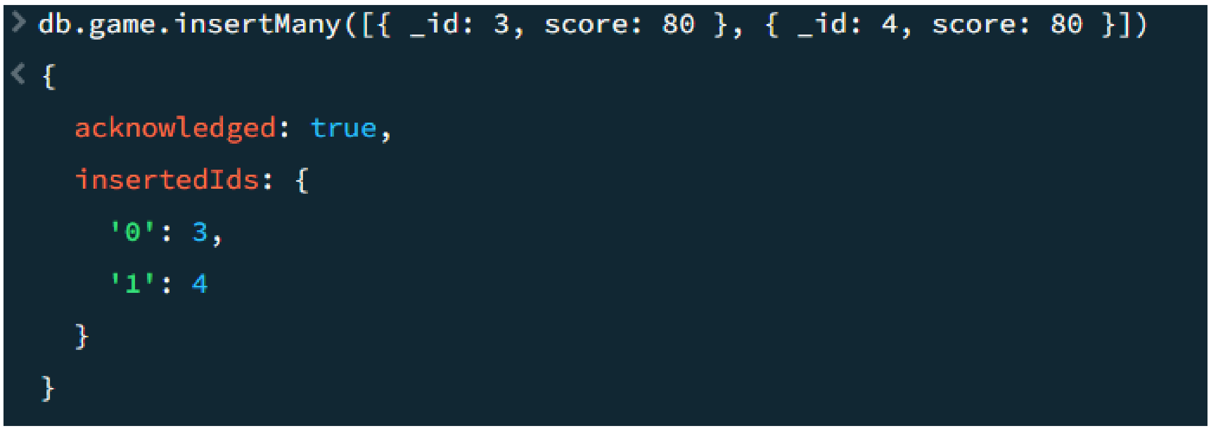
db.inventory.deleteMany({})



```
 db.inventory.deleteMany({})
{
  acknowledged: true,
  deletedCount: 15
}
```

Insert multiple documents:

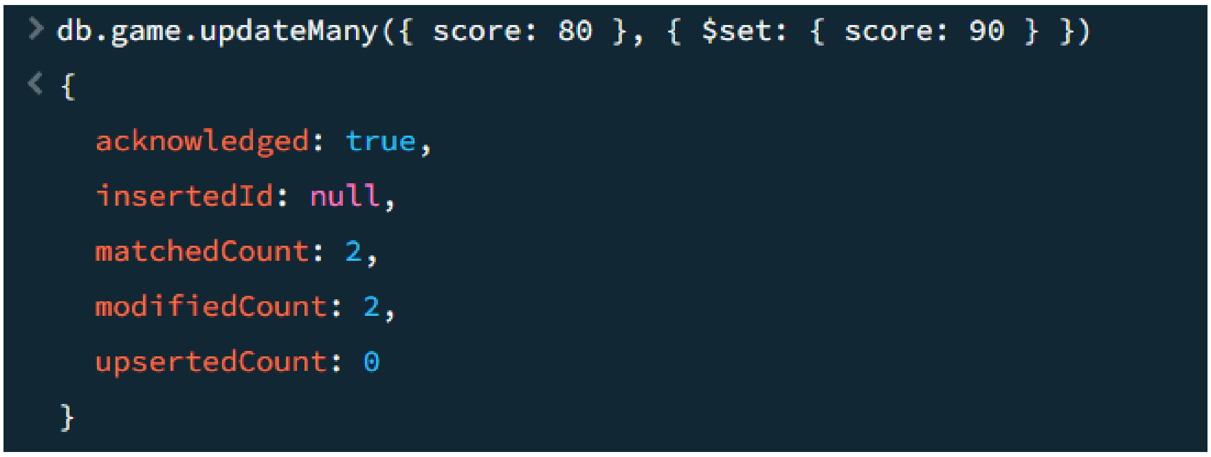
db.game.insertMany([{ _id: 3, score: 80 }, { _id: 4, score: 80 }])



```
> db.game.insertMany([{ _id: 3, score: 80 }, { _id: 4, score: 80 }])
< {
    acknowledged: true,
    insertedIds: {
      '0': 3,
      '1': 4
    }
  }
```

Update multiple documents:

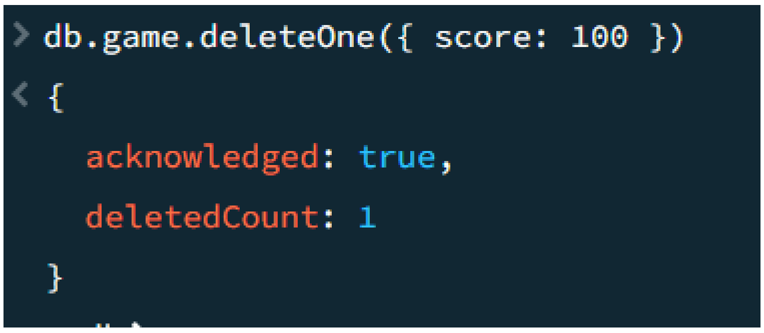
db.game.updateMany({ score: 80 }, { $set: { score: 90 } })



```
> db.game.updateMany({ score: 80 }, { $set: { score: 90 } })
< {
    acknowledged: true,
    insertedId: null,
    matchedCount: 2,
    modifiedCount: 2,
    upsertedCount: 0
  }
```

Delete Operations in Particular document:

db.game.deleteOne({ score: 100 })

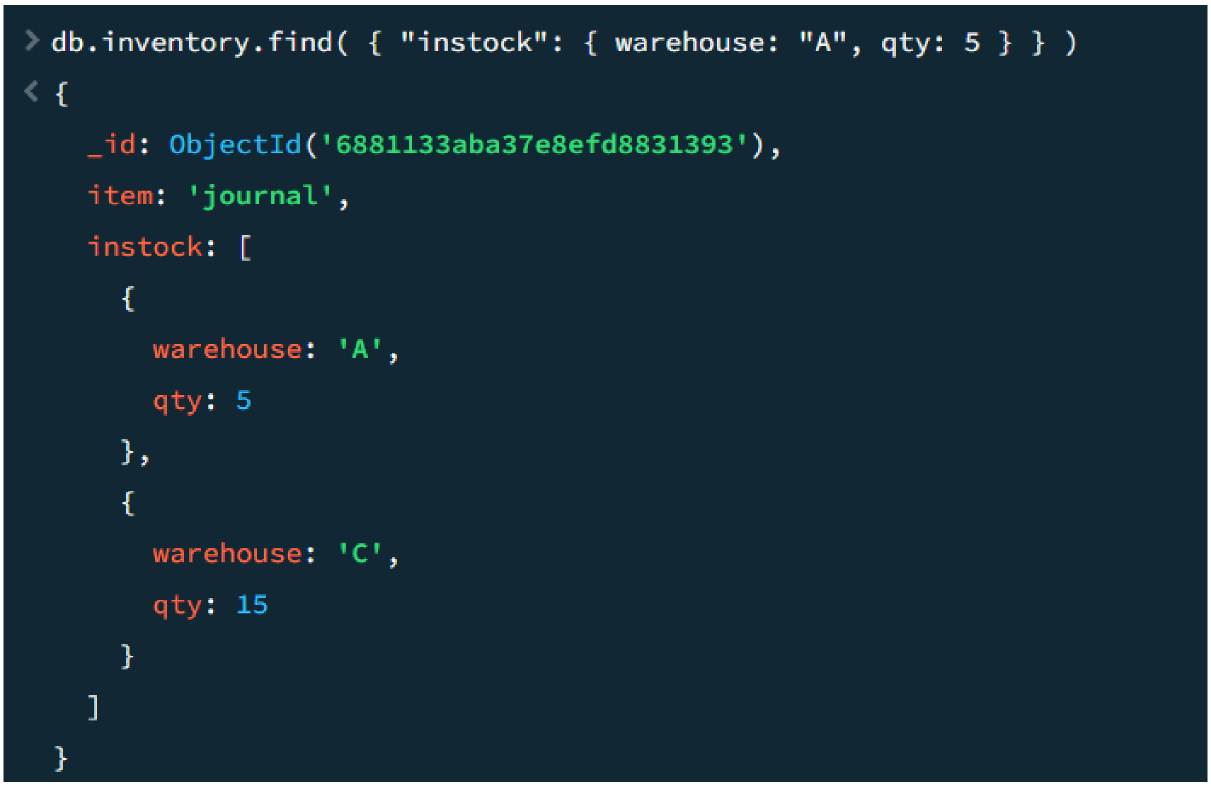


```
> db.game.deleteOne({ score: 100 })
< {
    acknowledged: true,
    deletedCount: 1
  }
```

Query Embedded Fields

Query for a Document Nested in an Array

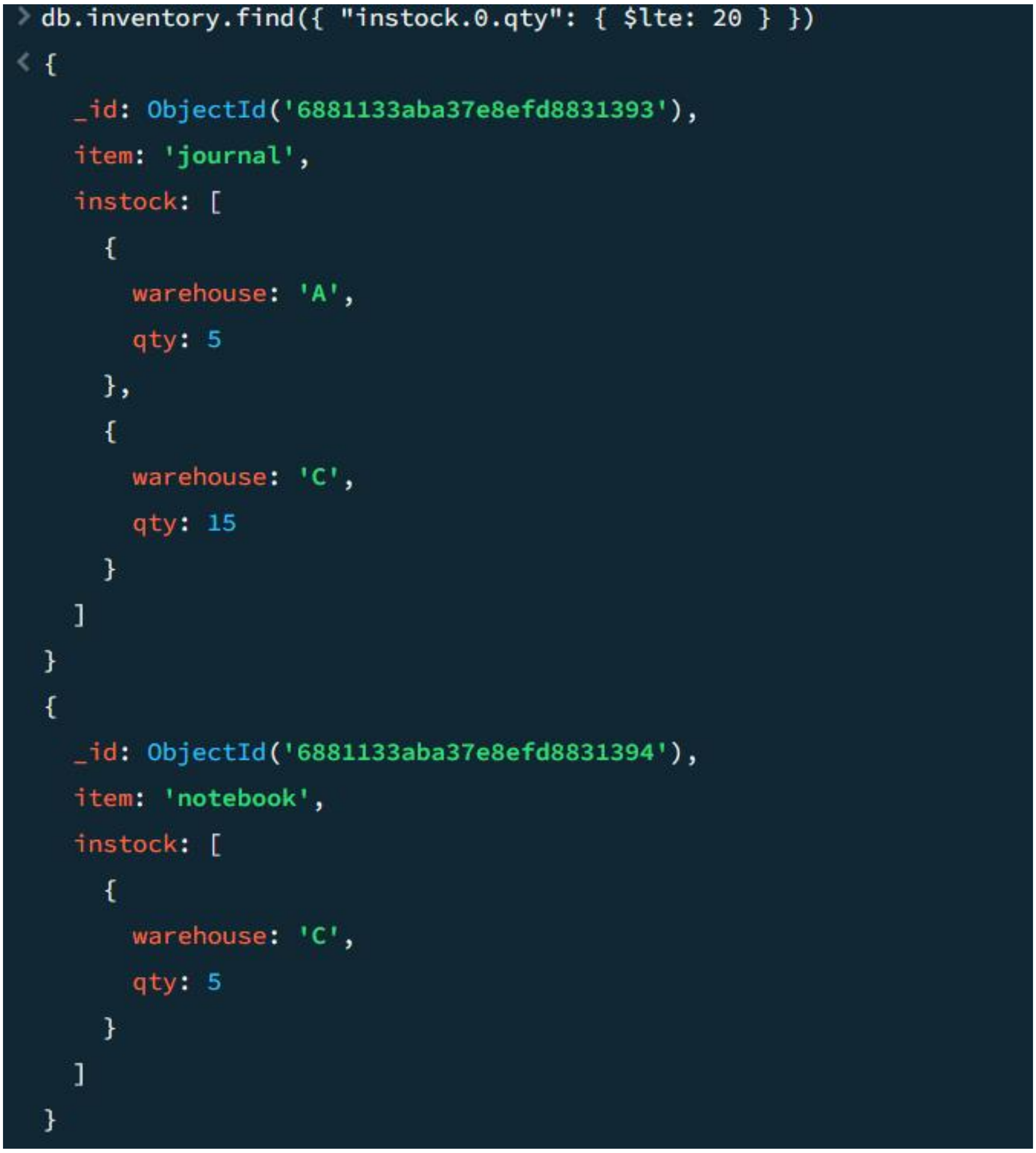
db.inventory.find({ instock: { warehouse: "A", qty: 5 } })



```
> db.inventory.find( { "instock": { warehouse: "A", qty: 5 } } )
< {
    _id: ObjectId('6881133aba37e8efd8831393'),
    item: 'journal',
    instock: [
      {
        warehouse: 'A',
        qty: 5
      },
      {
        warehouse: 'C',
        qty: 15
      }
    ]
  }
```

Specify a Query Condition on a Field in an Array of Documents

db.inventory.find({ "instock.0.qty": { $lte: 20 } })



```
> db.inventory.find({ "instock.0.qty": { $lte: 20 } })
< {
    _id: ObjectId('6881133aba37e8efd8831393'),
    item: 'journal',
    instock: [
      {
        warehouse: 'A',
        qty: 5
      },
      {
        warehouse: 'C',
        qty: 15
      }
    ]
  }
  {
    _id: ObjectId('6881133aba37e8efd8831394'),
    item: 'notebook',
    instock: [
      {
        warehouse: 'C',
        qty: 5
      }
    ]
  }
```