Todo : Insert documents using both Compass

Export a small dataset from Postgres and import it into MongoDB for comparison.

Mongodb

Doc -> collections -> databases

db.dbname.insertOne({ name: “tanmaie”})

db.dbname.insertMany

db.dbname.find()

db.dbname.limit()

db.dbname.sort()

db.dbname.skip().limit(2)

where -> db.dbname.find({name: “tanmaie”})

db.dbname.find({name:”tanmaie”}{name: 1, age:1}}

db.dbname.find({name: {$eq:”sally”})

db.dbname.find({name:{$ne

$gt

$gte

$lte

$lt less than

$in: [12,18] this or that

$nin

{age: {$

Show dbs

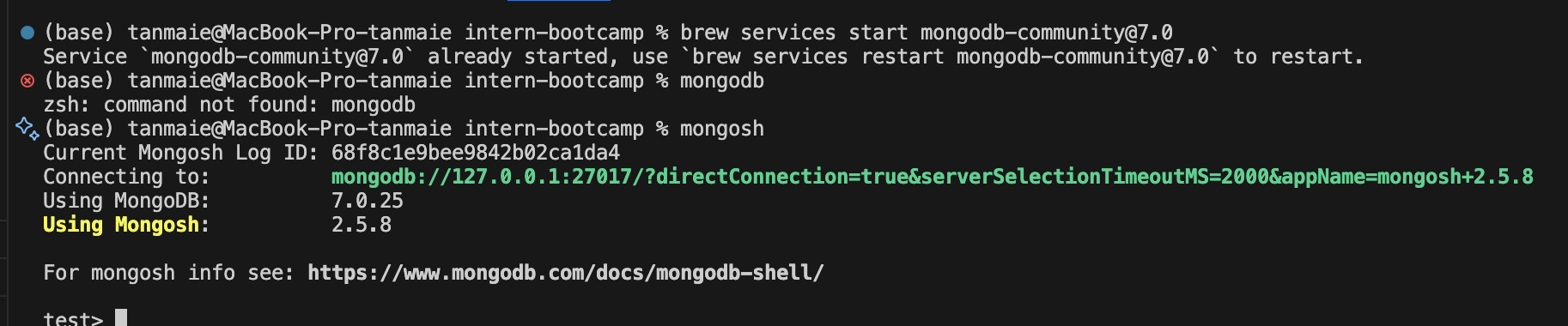
db.createCollection(“dbname”)

What update property can be set to true in order to insert a document if it's not found? **Upsert**

**.**deleteMany

Which update operator will replace the current value of a field with a new, specified value? $SET

.aggregate



Monogodb

A screenshot of a computer

Description automatically generated

client = MongoClient(" ") db = client["dbname"]

client = mongoClient(“”)

db = client[“”]

customers = d

from pymongo import MongoClient

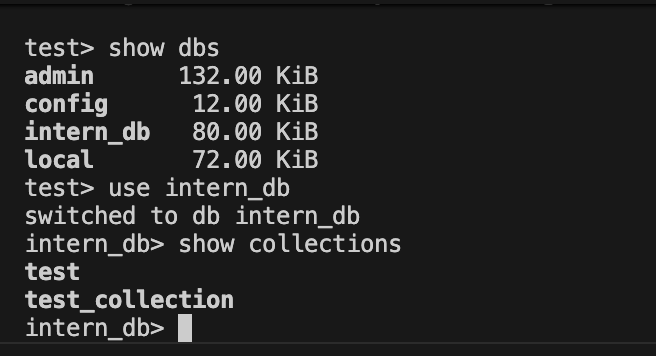
/Users/tanmaie/Downloads/intern-bootcamp/03\_mongo/mongo-data

Mongod - - port 27017 --db /Users/tanmaie/Downloads/intern-bootcamp/03\_mongo/mongo-data - - noauth

ls /opt/homebrew/etc/mongod.conf - check path

nano /opt/homebrew/etc/mongod.conf – change disabled here

control o, enter, control x



**MongoDB Shell Practice — Basic CRUD**

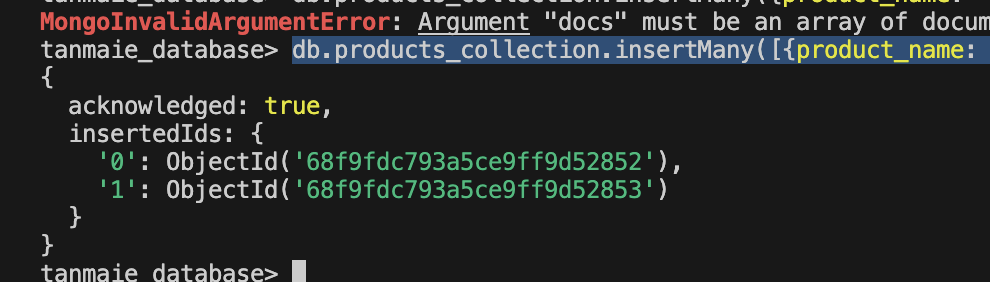
**1️⃣ Create**

1. Insert a new customer into customers\_collection with:
   * user\_name: "John Doe"
   * age: 35
   * address: "456 Main Street"
   * email: [johndoe@example.com](mailto:johndoe@example.com)

db. customers\_collection.insertOne({username:'john doe',age:35,address:'43 street,us',email:'johndoe@example.com'})

1. Insert 3 new products into products\_collection at once with your choice of:
   * product\_name, price, quantity\_prod

db.products\_collection.insertMany([{product\_name: 'apple',price:40},{product\_name:'charger', quantity\_prod: 40, price:50}])



**2️⃣ Read**

1. Show all customers in customers\_collection with all fields.

db.customers\_collection.find().limit(2)

1. Show only user\_name and email for all customers, hiding \_id.

db.customers\_collection.find({},{user\_name:1,email:1,\_id:0})

1. Find all products with price greater than 500.

db.products\_collection.find({price: {$gt: 500}})

1. Find all orders where customer\_name starts with the letter "A".

db.order\_collection.find({customer\_name:{$regex:"^A"}})

**3️⃣ Update**

1. Update the age of "John Doe" to 36.

db.customers\_collection.updateOne({ user\_name: "john doe" }, { $set: { age: 40 } })

1. Increase the price of all products with price less than 50 by 10%.

db.products\_collection.updateMany({price: {$lt:50}},{$mul:{price: 1.1}})

**4️ Delete**

1. Delete one customer with user\_name "'Ashley Maddox'".

db.customers\_collection.deleteOne({user\_name: 'Ashley Maddox'})

1. Delete all orders where order\_id is greater than 25.

db.order\_collection.deleteMany({order\_id: {$gt: 25}})

**5️ Bonus (Optional)**

1. Count how many customers are older than 40.

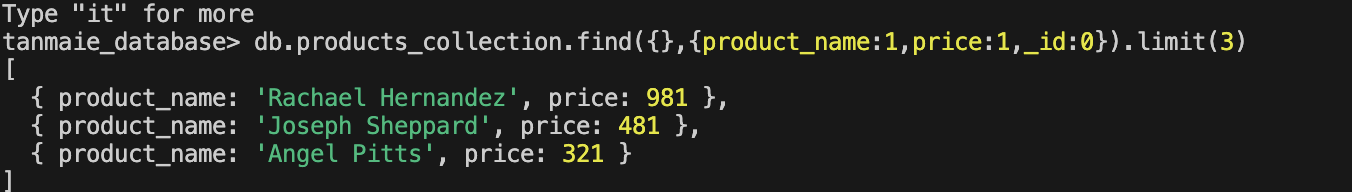
db.customers\_collection.countDocuments({age:{$gt:40}})

1. Show all collections in the tanmaie\_database.

Show collections

1. Find all products but only show product\_name and price.

db.products\_collection.find({},{product\_name:1,price:1,\_id:0}).limit(3)



1. Find **all products priced over 500**, show only product\_name and price.

db.products\_collection.find({price:{$gt: 500}},{product\_name:1,\_id:0,price:1})

1. Find **all orders for customers whose name starts with “A”**, show customer\_name and product\_name.

db.customers\_collection.find({ user\_name: { $regex: "^J" } })

1. List **top 5 youngest customers**.

db.customers\_collection.find({},{user\_name:1,age:1,\_id:0}).sort({age:1}).limit(5)

A screenshot of a computer screen

Description automatically generated

1. Sort **products by quantity descending** and show top 10.

db.products\_collection.find({},{quantity\_prod:1,product\_name:1,\_id:0}).sort({quantity\_prod:-1}).limit(5)

