Objective: The objective of this lab is to gain some familiarity with a non-relational database engine, more specifically MongoDB.

Instructions:

Step 1: The MongoDB documentation is a great place to start. Install the MongoDB 4.4 Community Edition, based on your operating system. You can decide whether or not to install MongoDB as a Windows Service or not (for those of you running Windows).

Step 2: According to the MongoDB documentation the mongo shell is an interactive JavaScript interface to MongoDB. You can use the mongo shell to query and update data as well as perform administrative operations. Download and install the mongo shell. You do not have to run this step if you use another method to interface with MongoDB.

Step 3: The MongoDB documentation contains a great tutorial on CRUD (Create, Read, Update, Delete) operations. You can use any data you would like for this step. The important thing is that you run through at least one create (insert) operation, one read operation, one update operation, and one delete operation.

Step 4: In two paragraphs (maximum 300 words), explain the differences between relational and non relational databases based on your experience and research so far.

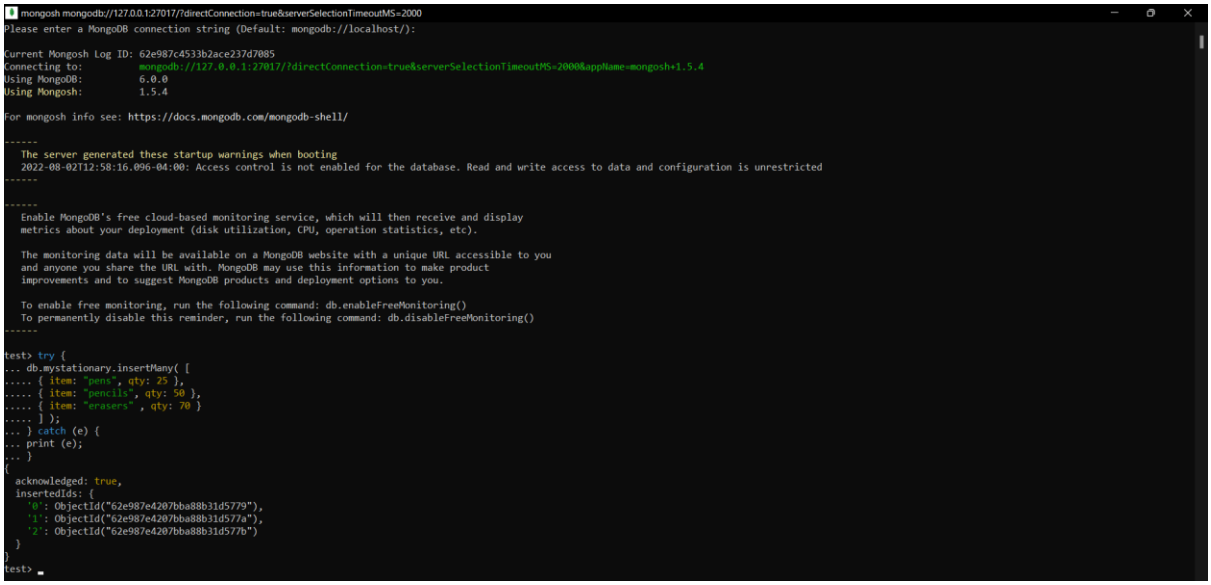
CRUD (Create, Read, Update, Delete) operations using *Mongosh* shell:

Create operation:

Creating records of stationary items and respective quantities.

```
try {
db.mystationary.insertMany( [
{ item: "pens", qty: 25 },
{ item: "pencils", qty: 50 },
{ item: "erasers" , qty: 70 }
] );
} catch (e) {
print (e);
}
```

Screenshot:



Read operation:

Reading records from stationary database with quantity greater than 30 and also limiting the result to 3 records.

db.mystationary.find({qty:{$gt:15}}).limit(3)

Screenshot:

```
test> db.mystationary.find({qty:{$gt:15}}).limit(3)
[
  { _id: ObjectId("62e987e4207bba88b31d5779"), item: 'pens', qty: 25 },
  {
    _id: ObjectId("62e987e4207bba88b31d577a"),
    item: 'pencils',
    qty: 50
  },
  {
    _id: ObjectId("62e987e4207bba88b31d577b"),
    item: 'erasers',
    qty: 70
  }
]
test>
```

Update operation:

Updating records of stationary database where item is ‘erasers’ and modifying the new quantity to 20.

db.mystationary.updateOne({ item: 'erasers' }, { $set: { qty: 20} })

Screenshot:

```
test> db.mystationary.updateOne({ item: 'erasers' }, { $set: { qty: 20} })
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
test>
```

Delete operation:

Deleting records from stationary database where item is ‘pens’.

db. mystationary.deleteOne({ item: 'pens' })

Screenshot:

```
test> db. mystationary.deleteOne({ item: 'pens' })
{ acknowledged: true, deletedCount: 1 }
test>
```

DB after CRUD operations:

Displaying final values stored in our database ‘stationary’ after performing CRUD operations.

db.mystationary.find()

Screenshot after performing CRUD operations:

```
test> db.mystationary.find()
[
  {
    _id: ObjectId("62e987e4207bba88b31d577a"),
    item: 'pencils',
    qty: 50
  },
  {
    _id: ObjectId("62e987e4207bba88b31d577b"),
    item: 'erasers',
    qty: 20
  }
]
test>
```

It can be observed that the quantity of the ‘erasers’ item has been dropped from 70 to 20 while the item ‘pens’ has been deleted.

Differences between relational and non relational databases based on your experience and research so far.

Due to the fact that relational databases rely on correctly organized information to store information, they require an increased attention to detail. There is always a clear relationship between tables and fields in relational databases, known as a schema. A relational database always allows new relationships between tables to be added without compromising the current structure. In addition, it reduces CPU usage and improves database efficiency by preventing duplication. A relational database provides a standardized method for organizing and storing data. A uniform set of rules should be applied to all DBMS. SQL breaks down data using the same principles; data that does not meet these requirements is not stored. It is essential for developers to create a logical and well-organized relation structure when working with complicated data structures with unstructured information.

In contrast to relational databases, non-relational databases are less dependent on order. When using a non-relational database, dealing with big data may be easier. In this way, users can rapidly change documents and provide keys to connected data without switching between tables. Non-relational database solutions typically have free versions available. Contrary to relational databases, which employ the same structure and language as relational systems, non-relational databases rely on the capabilities of a DBMS. While SQL has developed steadily over many years, non-relational database advances have come from several sources. As a result, customization options are greatly reduced. In comparison to SQL software, non-relational database systems typically lack advanced features and an intuitive user interface due to their recent development.

Relational databases emphasize structure, while non-relational databases emphasize flexibility. Therefore, it is important to ask the right questions and keep in mind the type of data one uses when choosing between a relational and non-relational database.
