MongoDB is an open-source, document-oriented, and one of the most popular NoSQL databases.

It stores data in **BSON** format, which is very much similar to JSON.

This lab session at providing in-depth information about MongoDB with the help of a huge dataset containing basic commands like insert, update, delete. In further sessions, we can explore advance features like authentication, backup, and storage, connecting projects with MongoDB, etc.

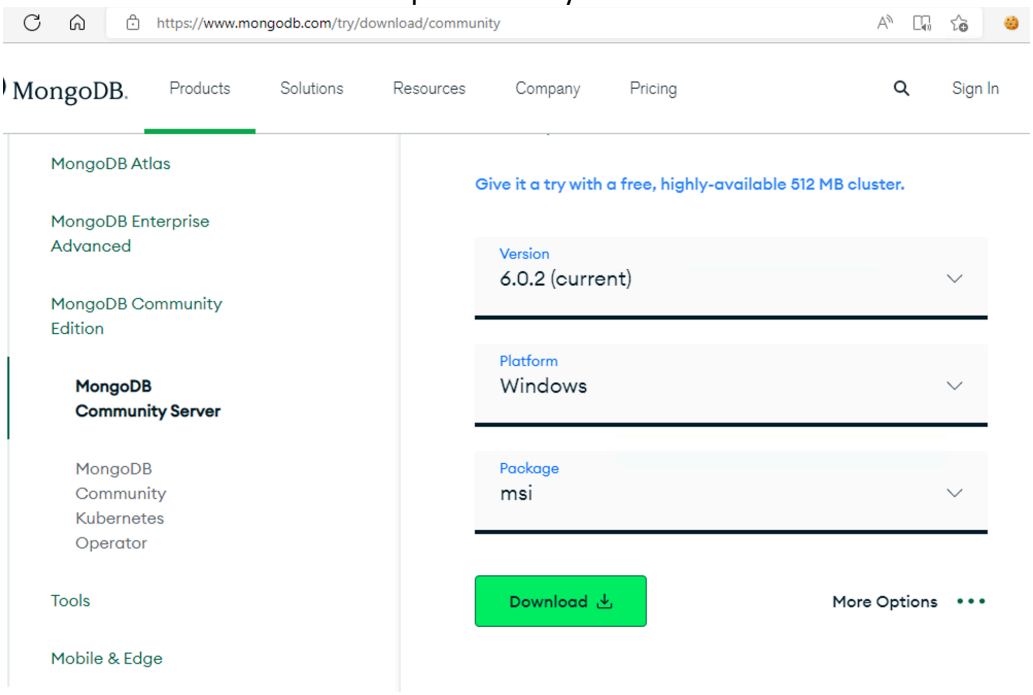
Tasks:

- **Task 1. Configure MongoDB**
- **Task 2. Working with databases**
- **Task 3. Working with collections**
- **Task 4. Insert data**
- **Task 5. Query data**
- **Task 6. Update data**
- **Task 7. Delete data**

Task 1. Configure MongoDB
MongoDB is available in two server editions: Community and Enterprise.

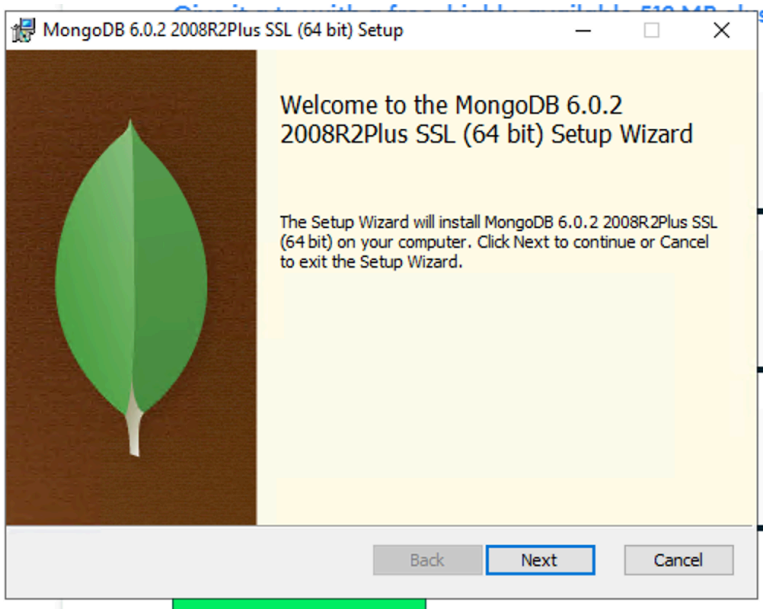
Visit **MongoDB Download Center** to download MongoDB Community Server.
https://www.mongodb.com/try/download/community

Choose the installer that corresponds with your OS:

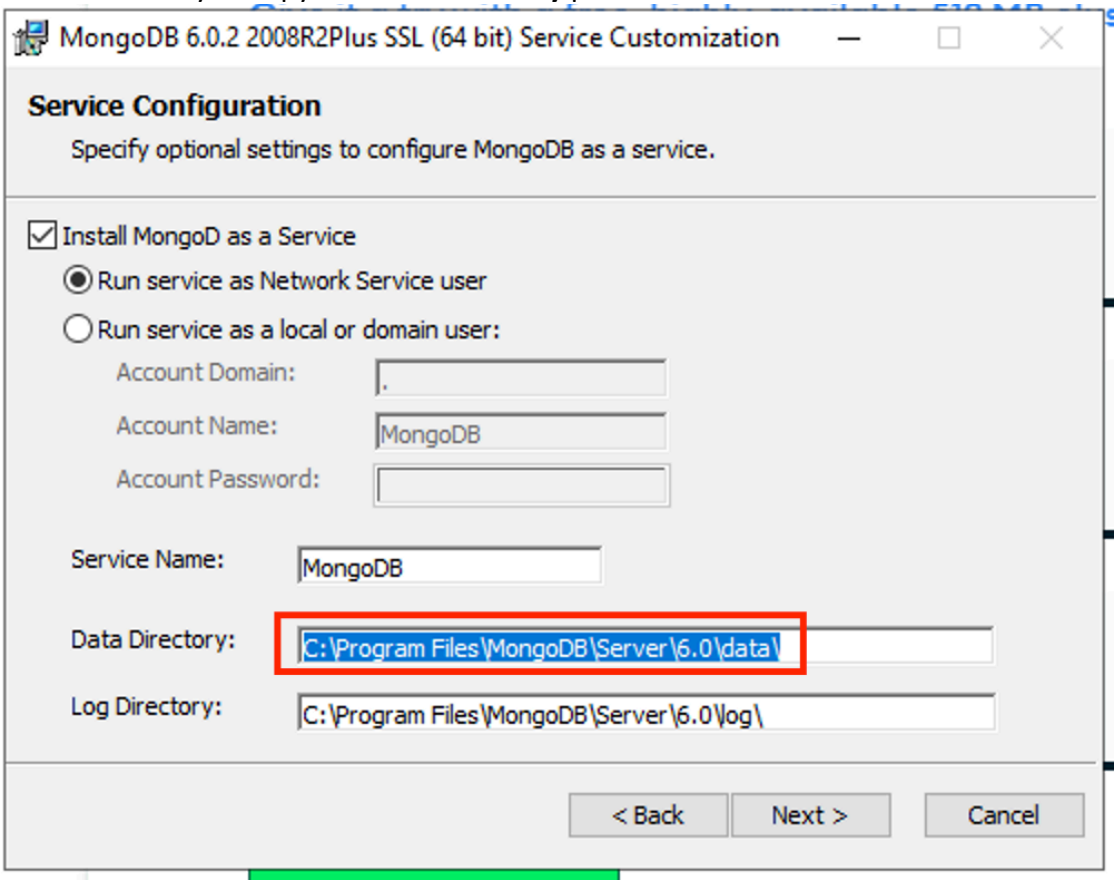


Here you can find instructions on how to proceed depending on the version that you downloaded: https://www.mongodb.com/docs/manual/administration/install-community/

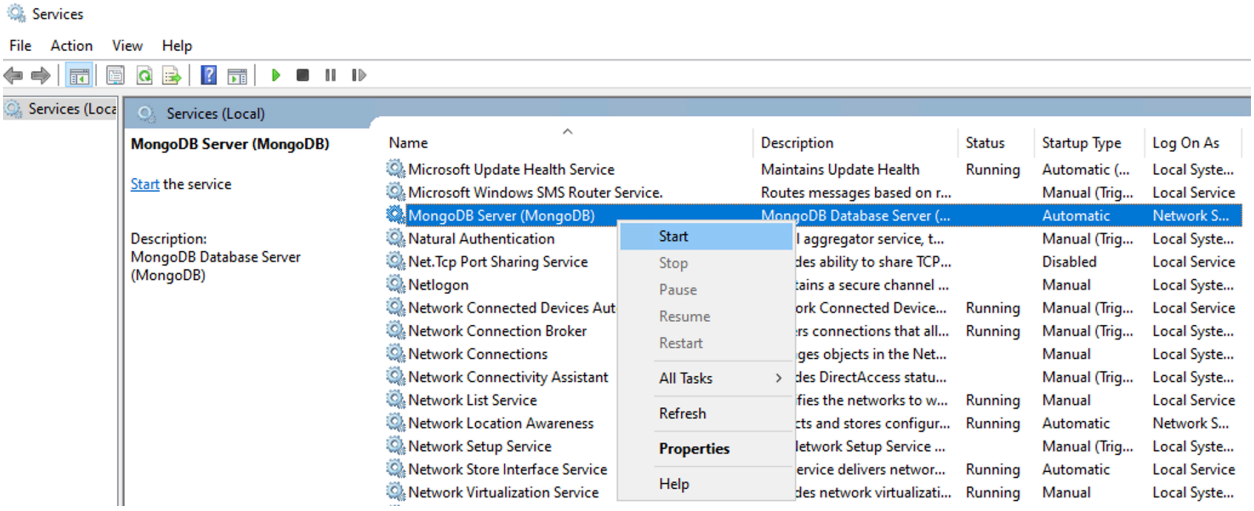
Proceed to install it



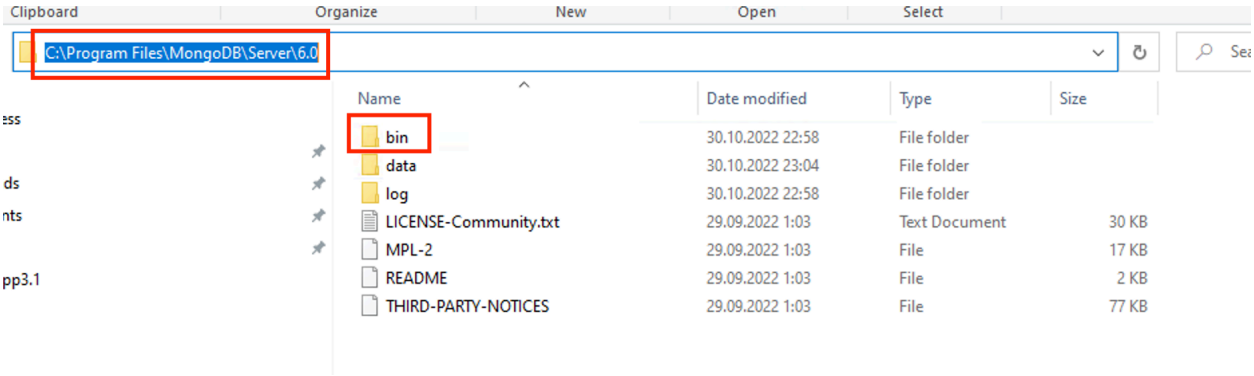
Make sure that you copy the **Data Directory** path



After MongoDB is installed, make sure the MongoDB Server service is running from the Services panel. **Start** it in case it is Stopped.

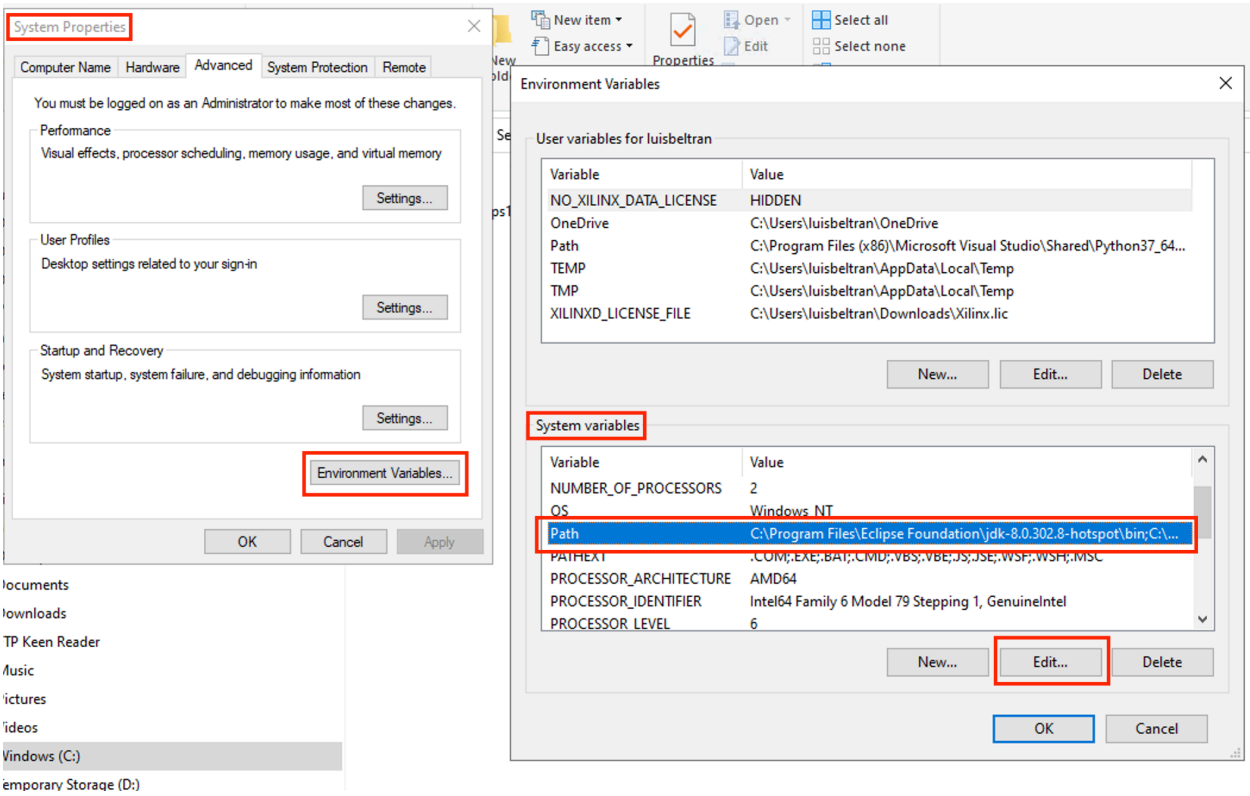


Now proceed to the following path (the Data Directory minus -data)
C:\Program Files\MongoDB\Server\6.0 and **go to the bin folder**

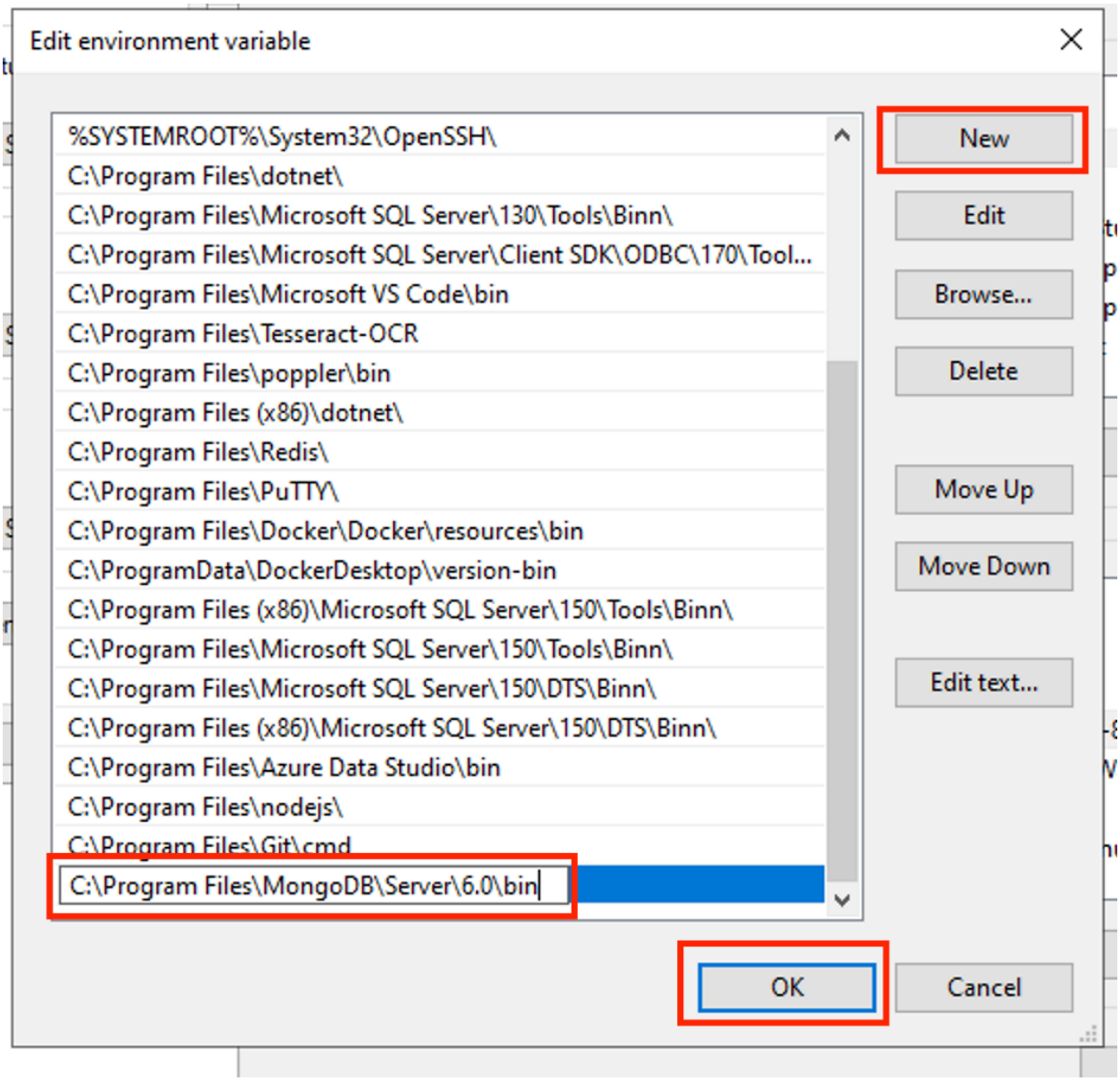


Copy this path **C:\Program Files\MongoDB\Server\6.0\bin**

Create an environment variable: System Properties → Environment Variable → System variable → PATH → Edit Environment variable.



Click on New, paste the path and click ok 3 times to confirm the new value



Now open a Command Prompt and type **mongod**

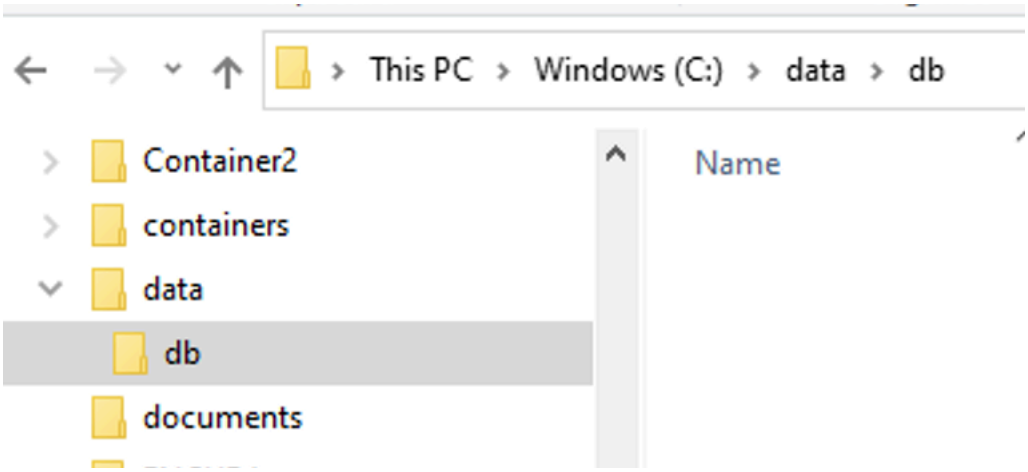
```
Select Command Prompt
Microsoft Windows [Version 10.0.19044.2130]
(c) Microsoft Corporation. All rights reserved.

C:\Users\luisbeltran>mongod
{"t":{"$date":"2022-10-30T23:14:52.815-12:00"},"s":"I",  "c":"NETWORK",  "id":49157
01, "ctx":"-","msg":"Initialized wire specification","attr":{"spec":{"incomingExter
nalClient":{"minWireVersion":0,"maxWireVersion":17},"incomingInternalClient":{"minW
ireVersion":0,"maxWireVersion":17},"outgoing":{"minWireVersion":6,"maxWireVersion":
17},"isInternalClient":true}}}
{"t":{"$date":"2022-10-30T23:14:52.823-12:00"},"s":"I",  "c":"CONTROL",  "id":23285
,   "ctx":"thread1","msg":"Automatically disabling TLS 1.0, to force-enable TLS 1.0
 specify --sslDisabledProtocols 'none'"}
{"t":{"$date":"2022-10-30T23:14:54.747-12:00"},"s":"I",  "c":"NETWORK",  "id":46486
02, "ctx":"thread1","msg":"Implicit TCP FastOpen in use."}
```

You will receive an error

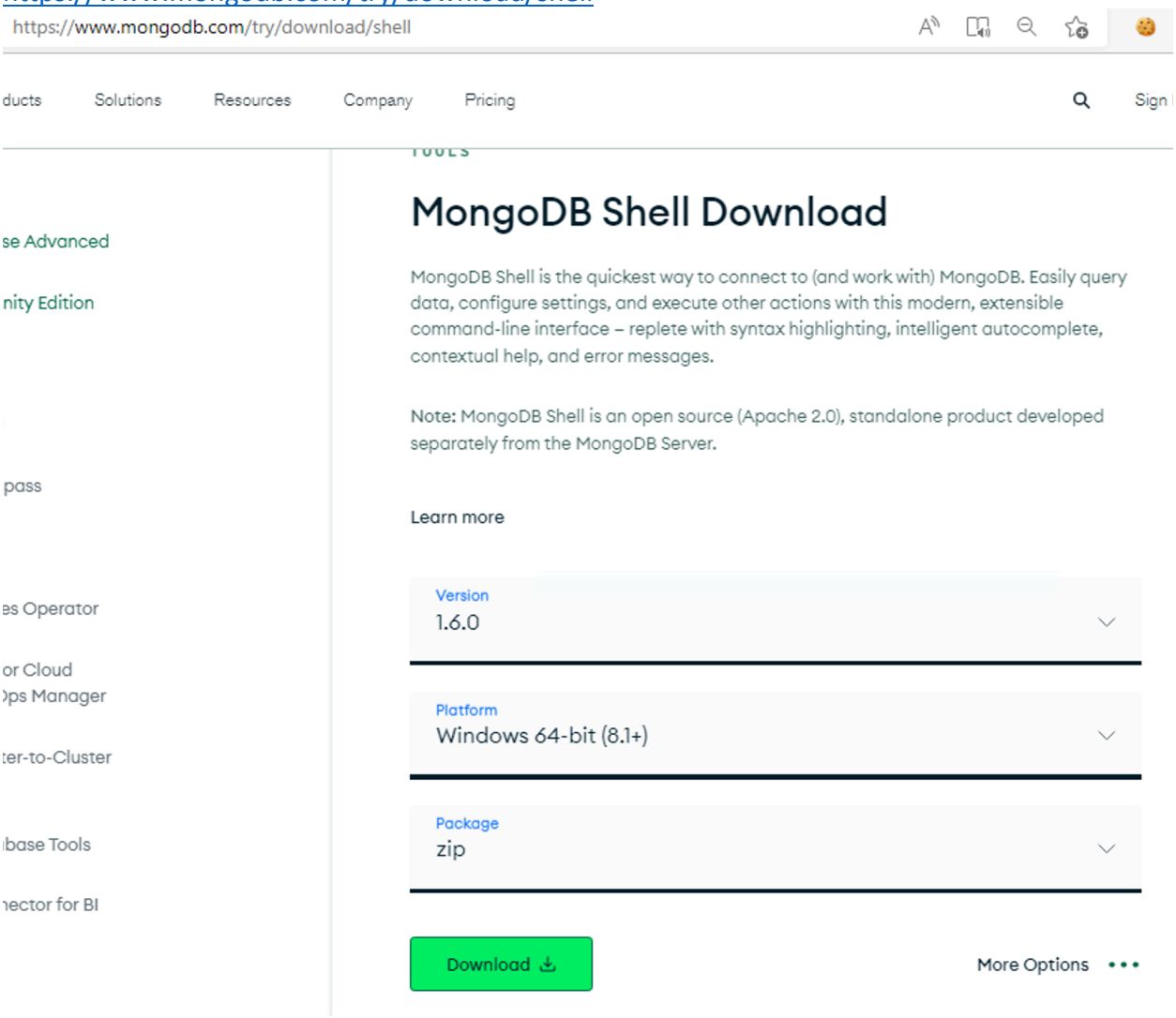
```
{"t":{"$date":"2022-10-30T23:14:54.755-12:00"},"s":"I",  "c":"CONTROL",  "id":21951
,   "ctx":"initandlisten","msg":"Options set by command line","attr":{"options":{}}
}
{"t":{"$date":"2022-10-30T23:14:54.761-12:00"},"s":"E",  "c":"CONTROL",  "id":20557
,   "ctx":"initandlisten","msg":"DBException in initAndListen, terminating","attr":
{"error":"NonExistentPath: Data directory C:\\data\\db\\ not found. Create the miss
ing directory or specify another path using (1) the --dbpath command line option, o
r (2) by adding the 'storage.dbPath' option in the configuration file."}}
{"t":{"$date":"2022-10-30T23:14:54.761-12:00"},"s":"I",  "c":"REPL",     "id":47849
00, "ctx":"initandlisten","msg":"Stepping down the ReplicationCoordinator for shutd
own","attr":{"waitTimeMillis":15000}}
{"t":{"$date":"2022-10-30T23:14:54.762-12:00"},"s":"I",  "c":"REPL",     "id":47946
```

Proceed to create a data folder and then db folder inside in C:



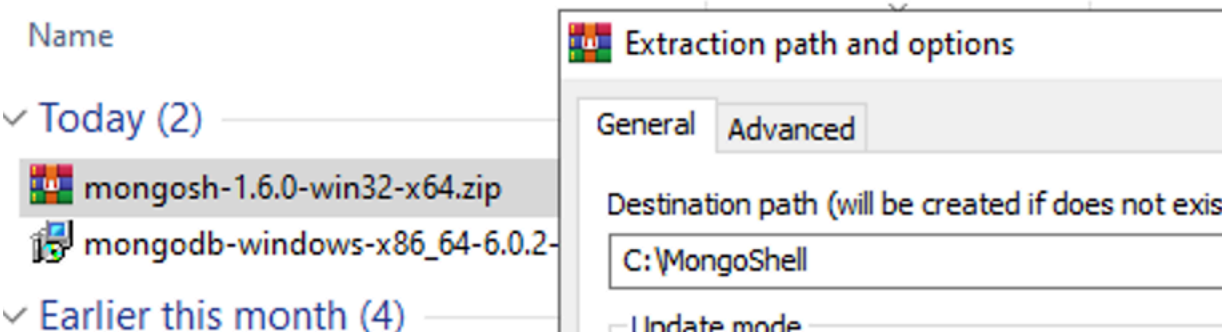
Go back to command prompt and try the **mongod** command again. This time, no error should appear.

Next, download and install the Mongo Shell
https://www.mongodb.com/try/download/shell

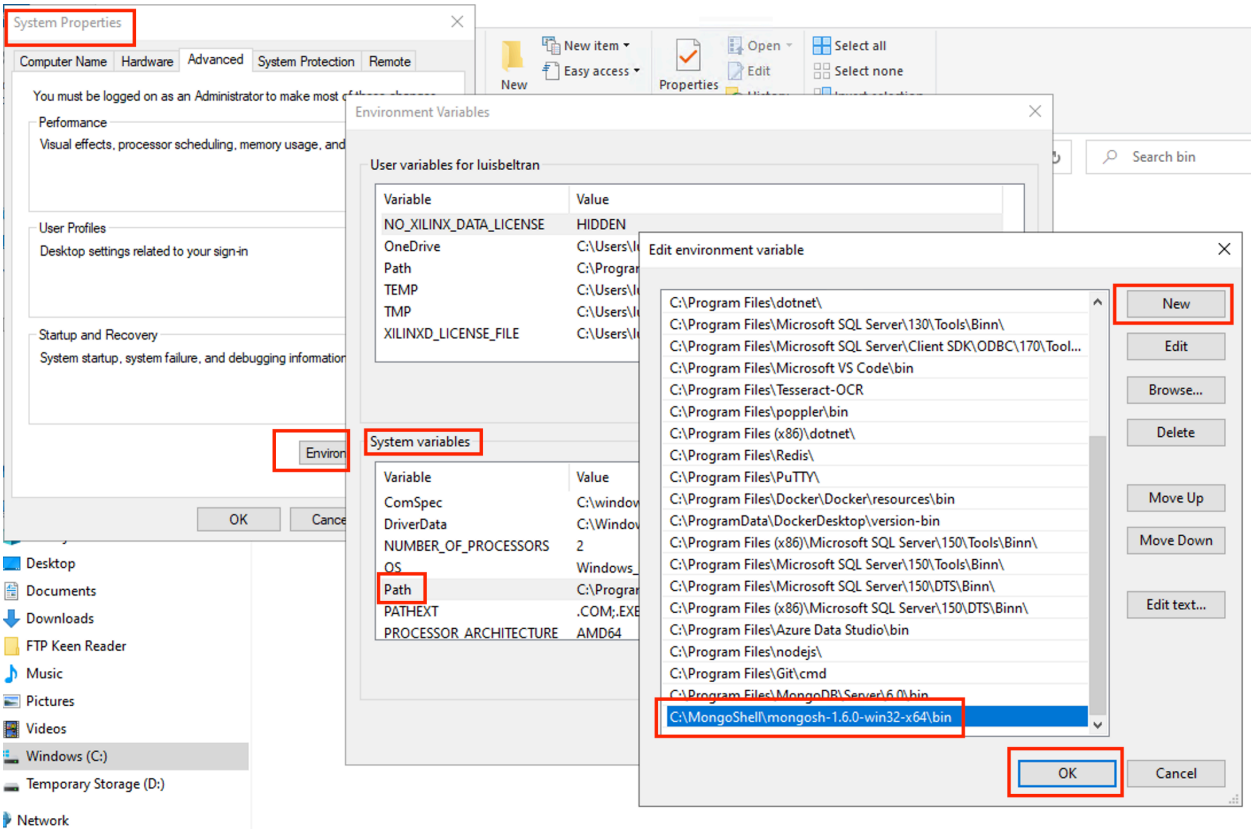


Instructions for installation and configuration of Mongo Shell
https://www.mongodb.com/docs/mongodb-shell/install/

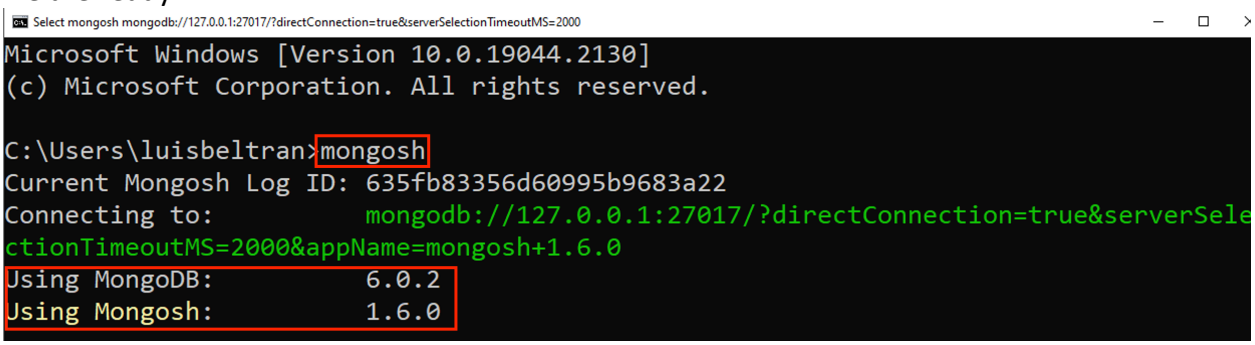
Extract the contents to any folder you want, for example, C:/MongoShell



It is recommended that you add the path to **bin** folder of MongoDB Shell to Environment Variables. For example, if your path is **C:\MongoShell\mongosh-1.6.0-win32-x64\bin**



Now, open a new command prompt and enter the command **mongosh.** If you receive no error, we are ready.



Task 2. Working with databases

The command **db** returns the current database

```
test> db
test
```

The command **use <db_name>** creates a new database with the db_name identifier (or returns a connection if db_name already exists)

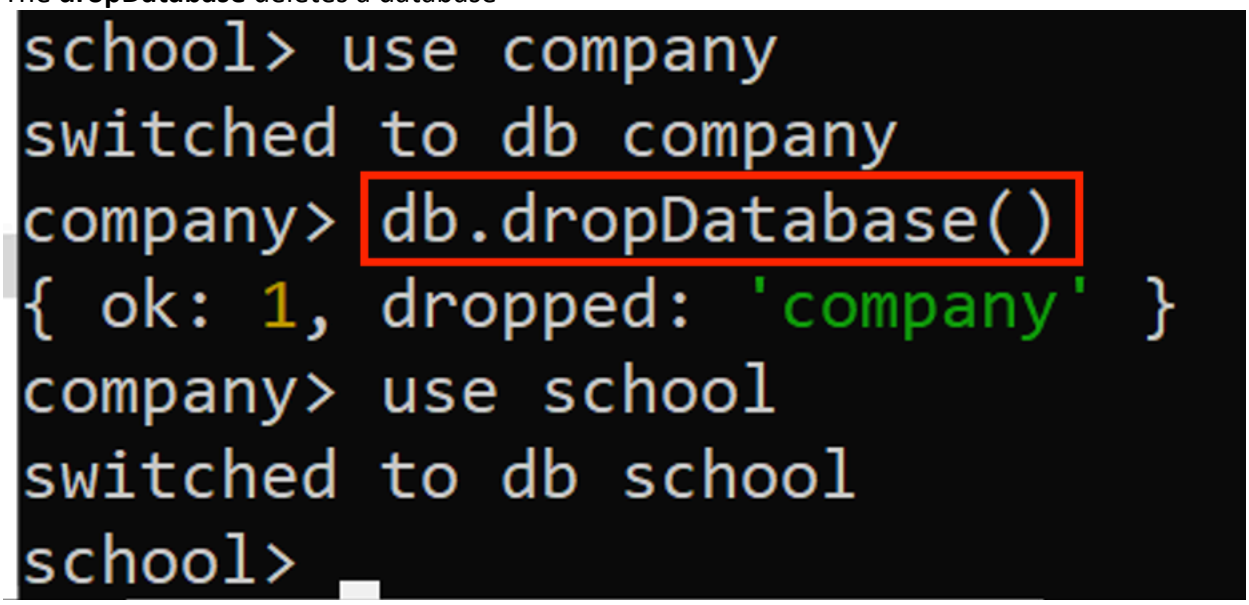
```
test> use school
switched to db school
school> db
school
school>
```

To display all available databases in your server use the following command:
db.adminCommand({listDatabases: 1})

```
school> db.adminCommand({listDatabases: 1})
{
  databases: [
    { name: 'admin', sizeOnDisk: Long("40960"), empty: false },
    { name: 'config', sizeOnDisk: Long("94208"), empty: false },
    { name: 'local', sizeOnDisk: Long("40960"), empty: false },
    { name: 'school', sizeOnDisk: Long("49152"), empty: false }
  ],
  totalSize: Long("225280"),
  totalSizeMb: Long("0"),
  ok: 1
}
```

NOTE: Empty databases might not appear

The **dropDatabase** deletes a database



Task 3. Working with collections

The **createCollection** database method creates a new collection in your database

```
school> db.createCollection("students")
{ ok: 1 }
school>
```

A second option is to create a collection during the insert process (if the collection doesn't exist, it creates it) by using **insertOne** method from a collection

```
school> db.professors.insertOne({name:"Luis"})
{
  acknowledged: true,
  insertedId: ObjectId("635fbbf20f3fdd0991403f0a")
}
```

The **getCollectionInfos** method lists all collections in the current database

```
school> db.getCollectionInfos()
[
  {
    name: 'students',
    type: 'collection',
    options: {},
    info: {
      readOnly: false,
      uuid: new UUID("724e4f83-662a-4b47-b605-15a2d5c242f2")
    },
    idIndex: { v: 2, key: { _id: 1 }, name: '_id_' }
  },
  {
    name: 'professors',
    type: 'collection',
    options: {},
    info: {
      readOnly: false,
      uuid: new UUID("d10bc546-f425-4b3a-90d3-c336221e3023")
    },
    idIndex: { v: 2, key: { _id: 1 }, name: '_id_' }
```

Another option is to use the **show collections** command

```
school> show collections
professors
students
```

To delete a collection, use the **drop** method

```
school> show collections
professors
students
school> db.professors.drop()
true
school> show collections
students
```

Assignment #1: Create a courses collection

Task 4. Insert data

To insert a single document, use the **insertOne** method.

db.students.insertOne({ name: "Ann", faculty: "FAI", semester: 7, lastAccess: new Date("2022-10-28T09:00:12"), languages: ["English", "Spanish"]})

```
school> db.students.insertOne({ name: "Ann", faculty:
 "FAI", semester: 7, lastAccess: new Date("2022-10-28
T09:00:12"), languages: ["English", "Spanish"]})
{
  acknowledged: true,
  insertedId: ObjectId("635fc15d0f3fdd0991403f0b")
}
```

To insert several documents at once, use the **insertMany** method

```
school> db.students.insertMany([
... {
... name: "Paul",
... faculty: "FT",
... semester: 4
... },
... {
... name: "Sara",
... faculty: "FAI",
... semester: 1
... }
... ]
...
... )
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId("635fc1ec0f3fdd0991403f0c"),
    '1': ObjectId("635fc1ec0f3fdd0991403f0d")
  }
```

There is a third method, **insert**, which you can use to add one or several documents to the collection. However, its use is not recommended (deprecated method)

```
school> db.students.insert([
... {
... name: "Tom",
... semester: 6,
... faculty: "FT"
... },
... {
... name: "Sam",
... semester: 9,
... faculty: "FAME"
... }
... ]
... )
DeprecationWarning: Collection.insert() is deprecated
. Use insertOne, insertMany, or bulkWrite.
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId("635fc2d20f3fdd0991403f0e"),
    '1': ObjectId("635fc2d20f3fdd0991403f0f")
  }
}
```

Assignment #2: Insert 3 documents in the courses collection

Task 5. Query data

In MongoDB, each document stored in a collection requires a unique **_id** field that acts as a primary key. If an inserted document omits the **_id** field, the MongoDB driver automatically generates an ObjectId for the _id field.

Use the **find** method to query and filter documents from a collection. Without any argument, you will retrieve all documents from the collection.

```
school> db.students.find()
[
  {
    _id: ObjectId("635fc15d0f3fdd0991403f0b"),
    name: 'Ann',
    faculty: 'FAI',
    semester: 7,
    lastAccess: ISODate("2022-10-28T21:00:12.000Z"),
    languages: [ 'English', 'Spanish' ]
  },
  {
    _id: ObjectId("635fc1ec0f3fdd0991403f0c"),
    name: 'Paul',
    faculty: 'FT',
    semester: 4
  },
  {
    _id: ObjectId("635fc1ec0f3fdd0991403f0d"),
    name: 'Sara',
    faculty: 'FAI',
    semester: 1
  },
```

You can pass a key-value pair to filter documents that contain the property and value:

```
school> db.students.find({ faculty: "FAI" })
[
  {
    _id: ObjectId("635fc15d0f3fdd0991403f0b"),
    name: 'Ann',
    faculty: 'FAI',
    semester: 7,
    lastAccess: ISODate("2022-10-28T21:00:12.000Z"),
    languages: [ 'English', 'Spanish' ]
  },
  {
    _id: ObjectId("635fc1ec0f3fdd0991403f0d"),
    name: 'Sara',
    faculty: 'FAI',
    semester: 1
  }
]
```

You can specify more properties at once to retrieve documents that meet ALL conditions

```
school> db.students.find({ faculty: "FAI", semester: 1 })
[
  {
    _id: ObjectId("635fc1ec0f3fdd0991403f0d"),
    name: 'Sara',
    faculty: 'FAI',
    semester: 1
  }
]
```

Using the **$or** operator, you can specify a compound query that joins each clause with a logical OR conjunction so that the query selects the documents in the collection that match at least one condition.

{ $or: [{ <expression1> }, { <expression2> }, ... , { <expressionN> }] }

```
school> db.students.find({ $or: [ { faculty: "FAI" }, { semester: 4 } ] })
[
  {
    _id: ObjectId("635fc15d0f3fdd0991403f0b"),
    name: 'Ann',
    faculty: 'FAI',
    semester: 7,
    lastAccess: ISODate("2022-10-28T21:00:12.000Z"),
    languages: [ 'English', 'Spanish' ]
  },
  {
    _id: ObjectId("635fc1ec0f3fdd0991403f0c"),
    name: 'Paul',
    faculty: 'FT',
school>
  },
  {
    _id: ObjectId("635fc1ec0f3fdd0991403f0d"),
    name: 'Sara',
    faculty: 'FAI',
    semester: 1
  }
]
```

Use the **$in** operator when performing equality checks on the same field.

```
school> db.students.find({ faculty: { $in: [ "FAME", "FT" ] } } )
[
  {
    _id: ObjectId("635fc1ec0f3fdd0991403f0c"),
    name: 'Paul',
    faculty: 'FT',
    semester: 4
  },
  {
    _id: ObjectId("635fc2d20f3fdd0991403f0e"),
    name: 'Tom',
    semester: 6,
    faculty: 'FT'
  },
  {
    _id: ObjectId("635fc2d20f3fdd0991403f0f"),
    name: 'Sam',
    semester: 9,
    faculty: 'FAME'
  }
]
```

$not performs a logical NOT operation on the specified <operator-expression> and selects the documents that do not match it. This includes documents that do not contain the field.

```
school> db.students.find({ faculty: { $not: { $eq: "FAI" } } } )
[
  {
    _id: ObjectId("635fc1ec0f3fdd0991403f0c"),
    name: 'Paul',
    faculty: 'FT',
    semester: 4
  },
  {
    _id: ObjectId("635fc2d20f3fdd0991403f0e"),
    name: 'Tom',
    semester: 6,
    faculty: 'FT'
  },
  {
    _id: ObjectId("635fc2d20f3fdd0991403f0f"),
    name: 'Sam',
    semester: 9,
    faculty: 'FAME'
  }
]
```

Comparison operators:
https://www.mongodb.com/docs/manual/reference/operator/query-comparison/

The following query returns all students from a semester greater than 5

```
school> db.students.find({ semester: { $gt: 5 } } )
[
  {
    _id: ObjectId("635fc15d0f3fdd0991403f0b"),
    name: 'Ann',
    faculty: 'FAI',
    semester: 7,
    lastAccess: ISODate("2022-10-28T21:00:12.000Z"),
    languages: [ 'English', 'Spanish' ]
  },
  {
    _id: ObjectId("635fc2d20f3fdd0991403f0e"),
    name: 'Tom',
    semester: 6,
    faculty: 'FT'
  },
  {
    _id: ObjectId("635fc2d20f3fdd0991403f0f"),
    name: 'Sam',
    semester: 9,
    faculty: 'FAME'
  }
]
```

Assignment #3: Query the following data

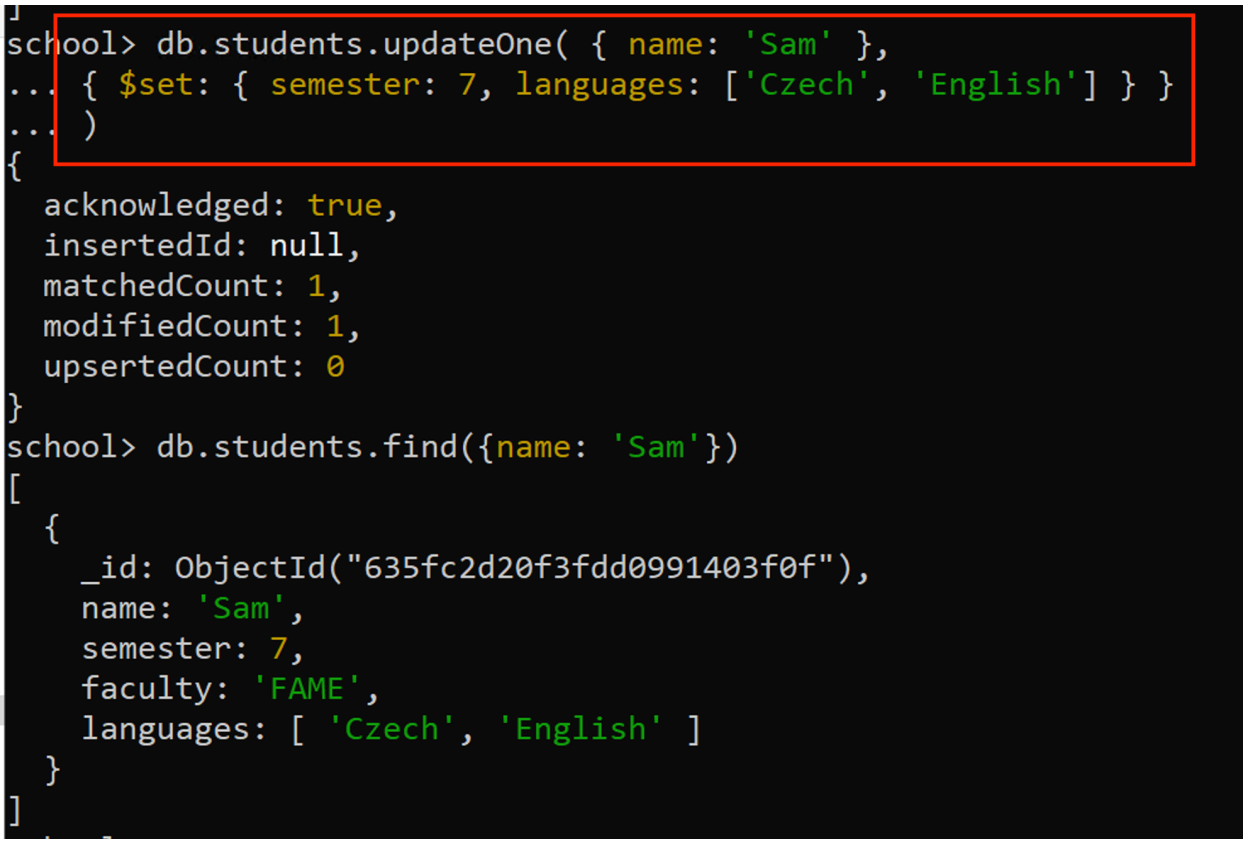
- Students from all faculties but FT
- Students Tom or Sara
- Students from FT faculty and sixth semester
- Students between 3rd and 6th semester (including them)

Task 6. Update data

There are several options to update documents:

- **db.collection.updateOne(<filter>, <update>, <options>)**
- **db.collection.updateMany(<filter>, <update>, <options>)**
- **db.collection.replaceOne(<filter>, <update>, <options>)**

The **db.collection.updateOne()** method on a collection updates the first document that meets the filter condition:



The **updateMany()** method updates all the documents that matches the given filter.

```
school> db.students.updateMany( { semester: { $lt: 5 } },
... { $set: { scholarship : 300 } } )
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 2,
  modifiedCount: 2,
  upsertedCount: 0
}
school> db.students.find({semester: { $lt: 5 } } )
[
  {
    _id: ObjectId("635fc1ec0f3fdd0991403f0c"),
    name: 'Paul',
    faculty: 'FT',
    semester: 4,
    scholarship: 300
  },
  {
    _id: ObjectId("635fc1ec0f3fdd0991403f0d"),
    name: 'Sara',
    faculty: 'FAI',
    semester: 1,
    scholarship: 300
```

The **replaceOne** method replaces a single document within the collection based on the filter.

```
school> db.students.replaceOne({ name: 'Sam' },
... { name: 'Samantha', semester: 8, languages: ['Czech', 'English'], faculty
: 'FAME' } )
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
school> db.students.find({name: 'Samantha'})
[
  {
    _id: ObjectId("635fc2d20f3fdd0991403f0f"),
    name: 'Samantha',
    semester: 8,
    languages: [ 'Czech', 'English' ],
    faculty: 'FAME'
  }
]
```

Use the **$upsert** argument to insert the document if no match was found.

```
school> db.students.replaceOne({ name: 'Luis' },
... { name: 'Luis', semester: 7, languages: ['Spanish', 'English'], faculty:
'FAI' },
... { upsert: true } )
{
  acknowledged: true,
  insertedId: ObjectId("635fd4c22a3eb8ec9eda1a56"),
  matchedCount: 0,
  modifiedCount: 0,
  upsertedCount: 1
}
school> db.students.find({name: 'Luis'})
[
  {
    _id: ObjectId("635fd4c22a3eb8ec9eda1a56"),
    name: 'Luis',
    semester: 7,
    languages: [ 'Spanish', 'English' ],
    faculty: 'FAI'
  }
]
school>
```

Update operators
https://www.mongodb.com/docs/manual/reference/operator/update/#update-operators-1

Let's increase the semester of all students that are in semesters less than 5

```
school> db.students.updateMany(
... { semester: { $lt: 5 } },
... { $inc: { semester: 1 } }
... )
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 2,
  modifiedCount: 2,
  upsertedCount: 0
}
school> db.students.find()
[
```

Assignment #4: Update the following documents:

- Ann's new faculty is FMK
- Replace the information of a student named Alice. If it does not exist, insert a new document with the information.
- Reduce by one the semester of all students from FAI faculty
- All scholarships will be increased by 30%

Task 7. Delete data

There are two methods that you can use to remove documents from a collection

- db.collection.deleteMany()
- db.collection.deleteOne()

To delete all documents that match a deletion criteria, pass a filter parameter to the **deleteMany()** method. The method returns a document with the status of the operation. For more information and examples, see **deleteMany()**.

```
school> db.students.deleteMany( { semester: 7 } )
{ acknowledged: true, deletedCount: 2 }
school> db.students.find( { semester: 7 } )

school>
```

To delete at most a single document that matches a specified filter (even though multiple documents may match the specified filter) use the **db.collection.deleteOne()** method.

```
school> db.students.deleteOne( { faculty: { $not : { $eq: "FAI" } } })
{ acknowledged: true, deletedCount: 1 }
school>
```

Assignment #5: Remote the following documents:

- All documents from FAME faculty
- One student with scholarship lower than 5000