MONGODB INTRODUCTION

Key*: All code cases and examples are written in RED

Description

MongoDB is a NoSQL, free usage document-oriented database software. It allows for users to store unstructured data while simultaneously providing full indexing support. MongoDB is popular with a variety of developers due to its scalable applications and evolving data schemas. Instead of using tables or rows, MongoDB comprises collections and documents.

KEY FEATURES - MONGODB VS MYSQL

Features	MongoDB	MySQL
Coding Languages	C++, documents with JSON format.	Structured Query Language, similar to R studio.
Schemas	Flexible	Fixed & structured Schema
Data integrity	No foreign keys or constraints	Can have foreign keys and constraints

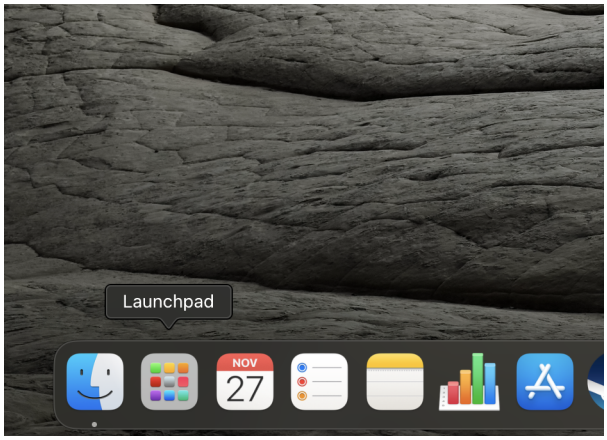
INSTALLATION SETUP

For MacOS:

https://www.mongodb.com/docs/manual/tutorial/install-mongodb-on-os-x/#install-mongodb-community-edition

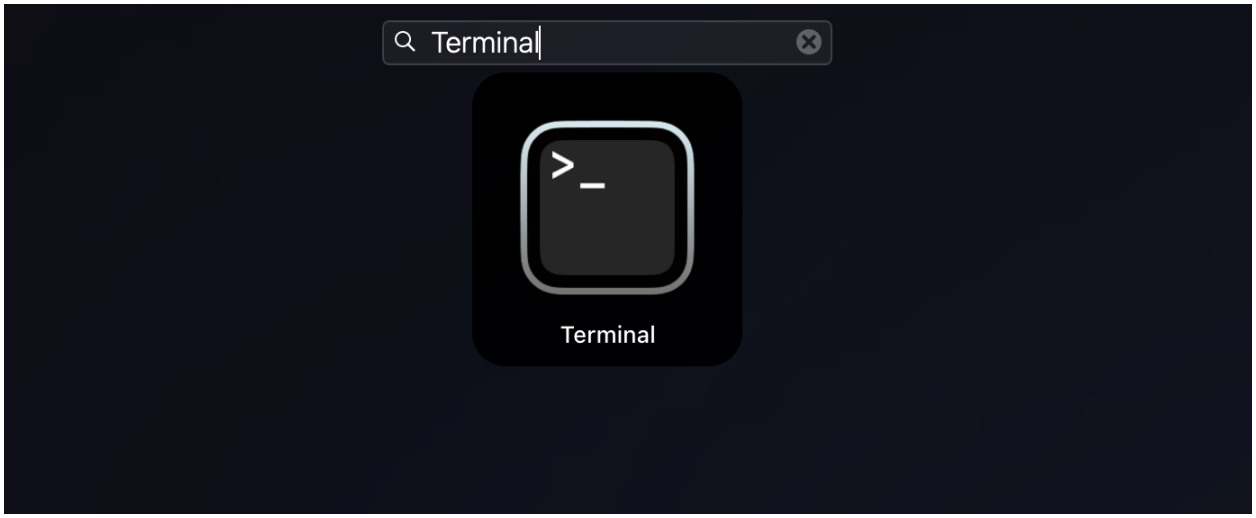
Finding the MacOS Terminal

Step 1. In your taskbar, open launchpad.



Step 2. After that, click the upper search bar.

Step 3. Search and open “Terminal.”



Installing MongoDB

Step 4. Once you're in the MacOS terminal input the following command:

xcode-select – –install

This will tell your computer to download command-line tools. These are tools needed to run MongoDB on Mac.

```
/Users/loganlasell/.zprofile:5: permission denied: /Users/loga
loganlasell@Logans-MacBook-Pro-4 ~ % xcode-select --install
```

Step 5. Go to https://brew.sh/#install Once there, copy the given link, paste it into your MacOS terminal, and follow the brew installation instructions.


```
/Users/loganlasell/.zprofile:5: permission denied: /Users/loganlasell/.zprofile
loganlasell@Logans-MacBook-Pro-4 ~ % /bin/bash -c "$(curl -fsSL https://raw.gith
ubusercontent.com/Homebrew/install/HEAD/install.sh)"
```

Step 6. Once brew is fully downloaded, run this homebrew command to download MongoDB and the database tools: brew tap mongodb/brew

```
Last login: Fri Nov 24 16:44:22 on ttys004
/Users/loganlasell/.zprofile:3: permission denied: /Users/loganlasell/.zprofile
/Users/loganlasell/.zprofile:5: permission denied: /Users/loganlasell/.zprofile
loganlasell@Logans-MacBook-Pro-4 ~ % brew tap mongodb/brew
Running `brew update --auto-update`...
==> Auto-updated Homebrew!
Updated 4 taps (homebrew/services, mongodb/brew, homebrew/core and homebrew/cask
).
==> New Formulae
action-validator           python-charset-normalizer  qbittorrent-cli
amass                      python-cycler              rapidfuzz-cpp
ansible@8                  python-dateutil            richgo
asitop                     python-hatch-vcs           scarb
awscli-local               python-hatchling           shell2http
cherrybomb                 python-idna                shellspec
flyscrape                  python-kiwisolver          skate
gdrive@2                   python-magic               sloth
geoip2fast                 python-matplotlib          snakeviz
```

Step 7. Update Homebrew with: brew update

```
/Users/loganlasell/.zprofile:3: permission denied: /User
/Users/loganlasell/.zprofile:5: permission denied: /User
loganlasell@Logans-MacBook-Pro-4 ~ % brew update
Already up-to-date.
loganlasell@Logans-MacBook-Pro-4 ~ %
```

Step 8. Input this command into your MacOS terminal to install MongoDB: brew install mongodb-community@7.0

```
loganlasell@Logans-MacBook-Pro-4 ~ % brew install mongodb-community@7.0
```

Step 9. To run MongoDB input this command in your terminal: brew services start mongodb-community@7.0

```
loganlasell@Logans-MacBook-Pro-4 ~ % brew services start mongodb-community@7.0
```

Step 10. To ensure MongoDB is up and running use this command: brew services list

And it should show the program has "started".

```
/Users/loganlasell/.zprofile:3: permission denied: /Users/loganlasell/.zprofile
/Users/loganlasell/.zprofile:5: permission denied: /Users/loganlasell/.zprofile
loganlasell@Logans-MacBook-Pro-4 ~ % brew services list
Name              Status  User        File
mongodb-community started loganlasell ~/Library/LaunchAgents/homebrew.mxcl.mongo
db-community.plist
loganlasell@Logans-MacBook-Pro-4 ~ %
```

Step 11. Finally, to begin using MongoDB, open a new terminal window and copy the following: mongosh

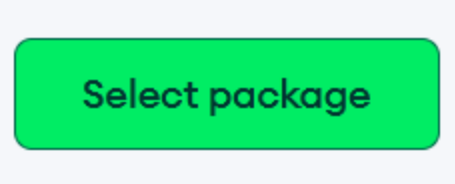
```
/Users/loganlasell/.zprofile:3: permission denied: /Users/loganlasell/.zprofile
/Users/loganlasell/.zprofile:5: permission denied: /Users/loganlasell/.zprofile
loganlasell@Logans-MacBook-Pro-4 ~ % mongosh
(node:11674) [DEP0040] DeprecationWarning: The `punycode` module is deprecated.
Please use a userland alternative instead.
(Use `node --trace-deprecation ...` to show where the warning was created)
Current Mongosh Log ID: 656160cdcc5b8a82962d1953
Connecting to:          mongodb://127.0.0.1:27017/?directConnection=true&serverS
electionTimeoutMS=2000&appName=mongosh+2.0.2
Using MongoDB:          7.0.2
Using Mongosh:          2.0.2
mongosh 2.1.0 is available for download: https://www.mongodb.com/try/download/sh
ell

For mongosh info see: https://docs.mongodb.com/mongodb-shell/

------
   The server generated these startup warnings when booting
   2023-11-15T10:55:50.766-10:00: Access control is not enabled for the database
. Read and write access to data and configuration is unrestricted
------

test>
```

Windows Installation

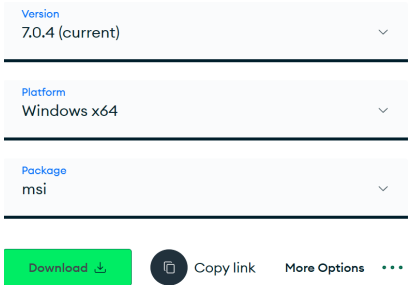
For Windows Installation:

(https://www.mongodb.com/docs/manual/tutorial/install-mongodb-on-windows/)

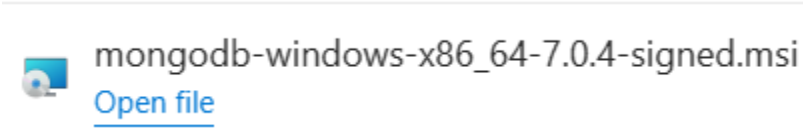
Step 1. Download MongoDB Community Server from the MongoDB Download Center.



- When selecting the platform to download, select Windows.
    - The installer can be either file, but .msi is recommended.

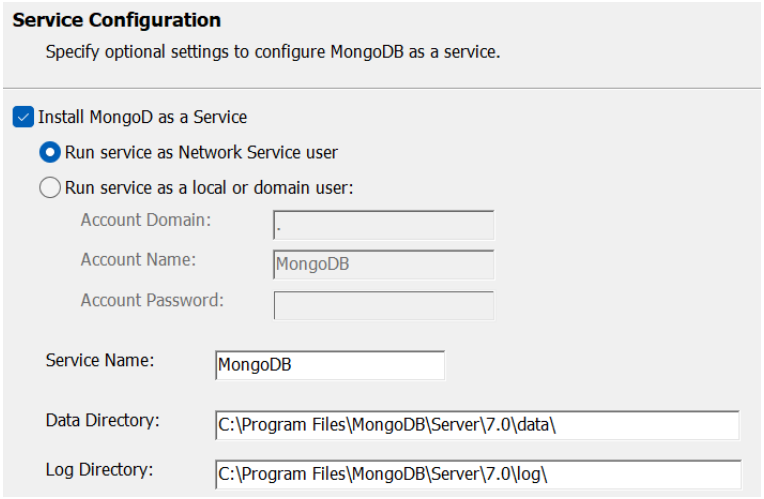


Step 2. After it has downloaded, run the installer. It should be in your downloads folder.



Step 3. Follow the installation wizard.

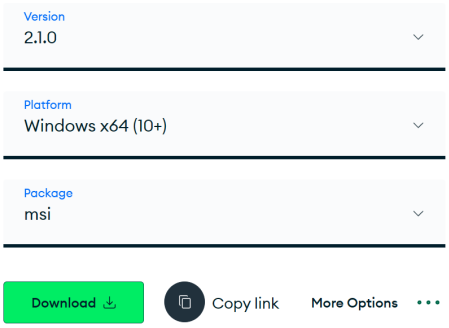
- The Complete installation is recommended.
- Install MongoD as a Service.
    - For simplicity, install as a Network Service user.



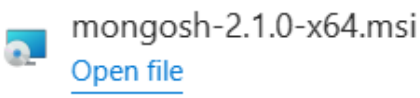
    - (Optional) Install MongoDB Compass.
- Click Install.

Step 4. Download MongoDB Shell.

- Same process as for when installing Community Server.



Step 5. After it has downloaded, run the installer. It should also be in the downloads folder.

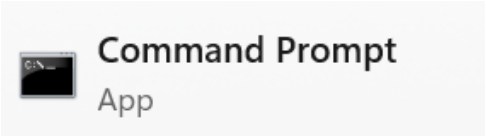


Step 6. Follow the Installation Wizard.

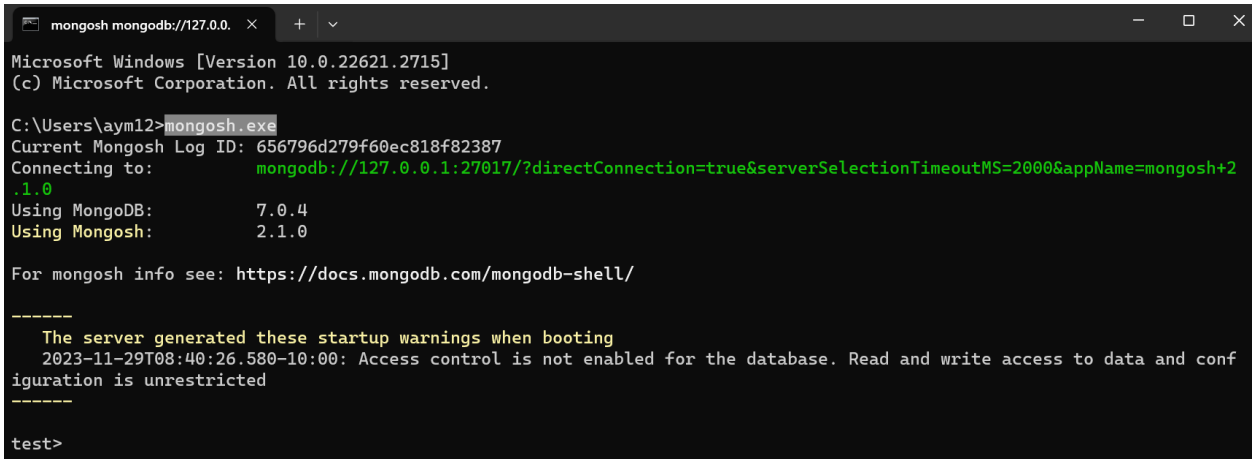
- Just hit Next and Install, there's no need to hit any other buttons or edit any text.

Step 7. Open up Command Prompt.

- You can use the search feature to find it.



Step 8. To finally begin using MongoDB, input mongosh.exe into the prompt and then hit enter.



MONGODB CASES

Description and Concepts

Knowing how to store, manage and manipulate data is key to efficiently using MongoDB. First, understanding the main ways to store data in MongoDB. Everything is housed in different databases. These databases house a diverse variety of bundled data called “Collections.” And in these collections are MongoDB’s forms of data called “documents.” These documents allow for information structure (table columns and rows) and data values to be inserted at the same time. Which you can come back to later and update. We can also use specific search methods to find and arrange data in a detailed manner, called querying.

Case 1:

1. Creating a Database

- After setting up and launching MongoDB, we will first start with creating a database.
- To create the database we will be using the statement “use <database name>”
- This statement tells MongoDB that we want to use a different database. And if we input a database name that doesn’t exist in our database collection, then MongoDB will automatically create a new one with that specific name.
- For our example we will use “school” as the database name (you can use any name for a database, choose something broad so that the collections and documents will fall under)
- In a MongoDB, input: use school
- Correct capitalization is important (school and School are differents names)
- admin> and school> indicate which database is selected.
- The code shown below can also be used to switch between dbs once the admin and school databases are made.
- Code Example: use school

```
admin> use school
switched to db school
school>
```

2. Creating Tables & Inserting Data

Collections

- Once we have the database created, we can now start making collections.
- We will use a db. method.
- These methods allow us to interact with MongoDB and our selected database, and in this interaction we can create, alter, and delete information inside the database.
- The specific method we will use is: db.createCollection(“<name of collection>”)
- This will allow us create a new collection in our database
- It will be called the name inputted in the quotes “”.
- We will be using the name “students” for one of our collections. Similar to the creation of names for databases, collections can be named anything. But they are case sensitive.
- Code Example: db.createCollection(“students”)
- Enter it and it will tell you how many collections were created.

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

Documents

- There are two main ways to create a document.
- Single Document Insertion and Multi-Document Insertion

Single Document Insertion

- The first way is to create one document at a time using the db.<collection>.insertOne({}) method.
- When inserting one document, the information that is being inserted must be surrounded by parentheses and braces like this ({info in here}). Each column name is followed by a colon, the values are entered afterwards as Integers or VarChar, and columns are separated by commas (i.e. db.<collection>.insertOne ({field name:value , field name: value}).
- Code Example: db.students.insertOne({name:"Jimmy", age:20, email:"boliviarocks@gmail.com", zodiac:"Libra", gpa:3.8})

```
school> db.createCollection("Students")
{ ok: 1 }
school> db.students.insertOne({name:"Jimmy", age:20, email:"boliviarocks@gmail.com", zodiac:"Libra", gpa:3.8})
{
  acknowledged: true,
  insertedId: ObjectId("6560ebff13ebb626aab0ea1b")
}
school>
```

Multi-Document Insertion

- The second way to create a document is to make multiple at a time using the db.<collection>.insertMany([{}, {}]) method.
- When inserting many documents, the information must be surrounded with one extra set of straight braces [ ], and each document has to be surrounded by its own set of curly braces, like so ([{1st doc in here}, {2nd doc in here}]).
- Code Example: db.students.insertMany([{name: "Name", age:__, email: "Email", zodiac: "Zodiac", gpa:#} , {name: "Name", age:__, email: "Email", zodiac: "Zodiac", gpa:#}])

```
school> db.students.insertMany([{name:"Gloria Webster", age:23, email:"ILift@hotmail.com", zodiac:"Cancer", gpa:3.5}, {name:"Barik", age:19, email:"wel0veyouBarik@sweetmail.com", zodiac:"Leo", gpa:4.0}])
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId("6560ee7d13ebb626aab0ea1c"),
    '1': ObjectId("6560ee7d13ebb626aab0ea1d")
  }
}
school>
```

Our Collections

- We created 3 collections, students, faculty, and courses.
- To query an entire collection we can use a collection-find method:
  db.<collection name>. find()
- The name inputted will be the one searched and it will show all documents within that collection
- Below is the code used to query the student collection
- Code example: db.students.find()

```
school> db.students.find()
[
  {
    _id: ObjectId("6560ebff13ebb626aab0ea1b"),
    name: 'Jimmy',
    age: 20,
    email: 'boliviarocks@gmail.com',
    zodiac: 'Libra',
    gpa: 3.8
  },
  {
    _id: ObjectId("6560ee7d13ebb626aab0ea1c"),
    name: 'Gloria Webster',
    age: 23,
    email: 'ILift@hotmail.com',
    zodiac: 'Cancer',
    gpa: 3.5
  },
  {
    _id: ObjectId("6560ee7d13ebb626aab0ea1d"),
    name: 'Barik',
    age: 19,
    email: 'wel0veyouBarik@sweetmail.com',
    zodiac: 'Leo',
    gpa: 4
  },
  {
    _id: ObjectId("6560f85313ebb626aab0ea1e"),
    name: 'Billy',
    age: 18,
    email: 'huluhaslivesports@gmail.com',
    zodiac: 'Virgo',
    gpa: 3
  },
  {
    _id: ObjectId("6560f85313ebb626aab0ea1f"),
    name: 'Olivander',
    age: 25,
    email: 'wowzerz@hackermail.com',
    zodiac: 'Capricorn',
    gpa: 3.2
  }
]
school>
```

- This is to query the faculty collection.
- Code example: db.faculty.find()

```
school> db.faculty.find()
[
  {
    _id: ObjectId("6560ff4013ebb626aab0ea20"),
    name: 'Sandy',
    age: 34,
    job: 'Professor of Psychology',
    zodiac: 'Taurus',
    email: 'iluvpsycho@crazymail.com'
  },
  {
    _id: ObjectId("6560ff4013ebb626aab0ea21"),
    name: 'Jeff',
    age: 47,
    job: 'Professor of Photography',
    zodiac: 'Gemini',
    email: 'photosbringmepeace@gmail.com'
  },
  {
    _id: ObjectId("6560ff4013ebb626aab0ea22"),
    name: 'Javale',
    age: 39,
    job: 'Professor of Music',
    zodiac: 'Libra',
    email: 'singyoursong@gmail.com'
  },
  {
    _id: ObjectId("6560ff4013ebb626aab0ea23"),
    name: 'Naomi',
    age: 40,
    job: 'Professor of English',
    zodiac: 'Scorpio',
    email: 'writerswillwrite@gmail.com'
  },
  {
    _id: ObjectId("6560ff4013ebb626aab0ea24"),
    name: 'Cindy',
    age: 30,
    job: 'Academic Adivsor',
    zodiac: 'Aquarius',
    email: 'howdidyougetintoaqua@water.com'
  }
]
school>
```

- This next one also shows what the full code for a multi-insert method would look like for one of our collections.
- And it also shows the courses collection
- Code Example: db.courses.find()

```
school> db.courses.insertMany([{name:"English 202", credit:3, category:"ENG", semester:"Fall", time:"9:00 am - 10:20 am"}, {name:"College Algebra II", credit:4, category:"MTH", semester:"Winter", time:"11:00 am - 12:00 pm"}, {name:"The Entire History of the World", credit:4, category:"HIS", semester:"Winter", time:"11:00 am - 1:00 pm"}, {name:"Computer and Database Systems", credit:3, category:"CS", semester:"Spring", time:"2:00 pm - 2:50 pm"}, {name:"African Studies", credit:4, category:"HIS", semester:"Fall", time:"9:00 am - 10:20 am"}])
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId("656117246df9ce4181ce7938"),
    '1': ObjectId("656117246df9ce4181ce7939"),
    '2': ObjectId("656117246df9ce4181ce793a"),
    '3': ObjectId("656117246df9ce4181ce793b"),
    '4': ObjectId("656117246df9ce4181ce793c")
  }
}
school> db.courses.find()
[
  {
    _id: ObjectId("656117246df9ce4181ce7938"),
    name: 'English 202',
    credit: 3,
    category: 'ENG',
    semester: 'Fall',
    time: '9:00 am - 10:20 am'
  },
  {
    _id: ObjectId("656117246df9ce4181ce7939"),
    name: 'College Algebra II',
    credit: 4,
    category: 'MTH',
    semester: 'Winter',
    time: '11:00 am - 12:00 pm'
  },
  {
    _id: ObjectId("656117246df9ce4181ce793a"),
    name: 'The Entire History of the World',
    credit: 4,
    category: 'HIS',
    semester: 'Winter',
    time: '11:00 am - 1:00 pm'
  },
  {
    _id: ObjectId("656117246df9ce4181ce793b"),
    name: 'Computer and Database Systems',
    credit: 3,
    category: 'CS',
    semester: 'Spring',
    time: '2:00 pm - 2:50 pm'
  },
  {
    _id: ObjectId("656117246df9ce4181ce793c"),
    name: 'African Studies',
    credit: 4,
    category: 'HIS',
    semester: 'Fall',
    time: '9:00 am - 10:20 am'
  }
]
school>
```

3. Querying Data

Databases and Collections

- First, querying databases. We can use a "show" query to look at all databases.
- Code Example: show dbs

```
school> show dbs
admin    40.00 KiB
config   72.00 KiB
local    72.00 KiB
school  152.00 KiB
school>
```

- We can query specific collections in a database with the same "show" method.
- Make sure you select the correct database before querying collections.
- Code Example: show collections

```
admin> use school
switched to db school
school> show collections
courses
faculty
students
school>
```

Document Query

- We can also query documents in different collections

- Using the db.<collection name>.find() method we can query all the documents within a collection.
- Code Example: db.faculty.find()

```
school> db.faculty.find()
[
  {
    _id: ObjectId("6560ff4013ebb626aab0ea20"),
    name: 'Sandy',
    age: 34,
    job: 'Professor of Psychology',
    zodiac: 'Taurus',
    email: 'iluvpsycho@crazymail.com'
  },
  {
    _id: ObjectId("6560ff4013ebb626aab0ea21"),
    name: 'Jeff',
    age: 47,
    job: 'Professor of Photography',
    zodiac: 'Gemini',
    email: 'photosbringmepeace@gmail.com'
  },
  {
    _id: ObjectId("6560ff4013ebb626aab0ea22"),
    name: 'Javale',
    age: 39,
    job: 'Professor of Music',
    zodiac: 'Libra',
    email: 'singyoursong@gmail.com'
  },
  {
    _id: ObjectId("6560ff4013ebb626aab0ea23"),
    name: 'Naomi',
    age: 40,
    job: 'Professor of English',
    zodiac: 'Scorpio',
    email: 'writerswillwrite@gmail.com'
  },
  {
    _id: ObjectId("6560ff4013ebb626aab0ea24"),
    name: 'Cindy',
    age: 30,
    job: 'Academic Adivsor',
    zodiac: 'Aquarius',
    email: 'howdidyougetintoaqua@water.com'
  }
]
school>
```

Field Search Query

- Going further, we can use that same tool to create a more specific query
- Adding a set of curly braces { } on the inside of the ( ) and inputting a field with a matching value from a document can query all the information from every document in that collection with the same field and value.
- For example: we inputted name:“Gloria Webster” as the name and field value and received all the documents with the name: Gloria Webster.
- Code example: db.students.find({name: “Gloria Webster”})

```
school> db.students.find({name:"Gloria Webster"})
[
  {
    _id: ObjectId("6560ee7d13ebb626aab0ea1c"),
    name: 'Gloria Webster',
    age: 23,
    email: 'ILift@hotmail.com',
    zodiac: 'Cancer',
    gpa: 3.5
  }
]
school>
```

Multi-Field Search Query

- Along with this, you can add multiple fields to help aid in a more specific query.
- The fields have to be a part of the same document.
- Each field is separated by a comma.
- Code example: db.students.find({age:20, gpa:3.8})

```
school> db.students.find({age:20, gpa:3.8})
[
  {
    _id: ObjectId("6560ebff13ebb626aab0ea1b"),
    name: 'Jimmy',
    age: 20,
    email: 'boliviarocks@gmail.com',
    zodiac: 'Libra',
    gpa: 3.8
  }
]
school>
```

Field Projection

- Another way to query data is to use a field projection to view specific fields in the collections by labeling them true or false.
- Still using the db.<collection name>.find() method
- Now, we will add two sets of curly braces with the ( )
- The second set of curly braces is where you input the fields you want to query.
- Code example: db.students.find({}, {name:true})

```
school> db.students.find({}, {name:true})
[
  { _id: ObjectId("6560ebff13ebb626aab0ea1b"), name: 'Jimmy' },
  { _id: ObjectId("6560ee7d13ebb626aab0ea1c"), name: 'Gloria Webster' },
  { _id: ObjectId("6560ee7d13ebb626aab0ea1d"), name: 'Barik' },
  { _id: ObjectId("6560f85313ebb626aab0ea1e"), name: 'Billy' },
  { _id: ObjectId("6560f85313ebb626aab0ea1f"), name: 'Olivander' }
]
school>
```

Multi-Field Projection

- Additionally, you can add multiple fields to the query. Each separated by commas.
- Pseudo-code: db.<collection name>.find({}, {field name:value, field name:value})

- Object id will automatically be added to the query but you can change it to false if you don't want to query it.
- Code Example: db.students.find({}, {name:true, _id:false, email:true})

```
school> db.students.find({}, {name:true, _id:false, email:true})
[
  { name: 'Jimmy', email: 'boliviarocks@gmail.com' },
  { name: 'Gloria Webster', email: 'ILift@hotmail.com' },
  { name: 'Barik', email: 'wel0veyouBarik@sweetmail.com' }
]
school>
```

Sort Query

- Adding onto the find() query. To add more specificity to your query, you can connect more than one query at a time.
- For example, the sort({}) query can be added on the original db.students.find( ) query.
- The sort({}) query can sort fields into alphabetical/reverse alphabetical order. And it can also sort them in increasing/decreasing order.
- When inputting the sort({}) query, you use 1 for alphabetical order/increasing order and -1 for reverse alphabetical order/decreasing order
- Code Example: db.students.find( ).sort({name:1}) (The "1" is for alphabetical order)

```
school> db.students.find().sort({name:1})
[
  {
    _id: ObjectId("6560ee7d13ebb626aab0ea1d"),
    name: 'Barik',
    age: 19,
    email: 'wel0veyouBarik@sweetmail.com',
    zodiac: 'Leo',
    gpa: 4
  },
  {
    _id: ObjectId("6560f85313ebb626aab0ea1e"),
    name: 'Billy',
    age: 18,
    email: 'huluhaslivesports@gmail.com',
    zodiac: 'Virgo',
    gpa: 3
  },
  {
    _id: ObjectId("6560ee7d13ebb626aab0ea1c"),
    name: 'Gloria Webster',
    age: 23,
    email: 'ILift@hotmail.com',
    zodiac: 'Cancer',
    gpa: 3.5
  },
  {
    _id: ObjectId("6560ebff13ebb626aab0ea1b"),
    name: 'Jimmy',
    age: 20,
    email: 'boliviarocks@gmail.com',
    zodiac: 'Libra',
    gpa: 3.8
  },
  {
    _id: ObjectId("6560f85313ebb626aab0ea1f"),
    name: 'Olivander',
    age: 25,
    email: 'wowzerz@hackermail.com',
    zodiac: 'Capricorn',
    gpa: 3.2
  }
]
school>
```

- Code Example: db.students.find( ).sort({name:-1}) (The "-1" is for reverse alphabetical order)

```
school> db.students.find().sort({name:-1})
[
  {
    _id: ObjectId("6560f85313ebb626aab0ea1f"),
    name: 'Olivander',
    age: 25,
    email: 'wowzerz@hackermail.com',
    zodiac: 'Capricorn',
    gpa: 3.2
  },
  {
    _id: ObjectId("6560ebff13ebb626aab0ea1b"),
    name: 'Jimmy',
    age: 20,
    email: 'boliviarocks@gmail.com',
    zodiac: 'Libra',
    gpa: 3.8
  },
  {
    _id: ObjectId("6560ee7d13ebb626aab0ea1c"),
    name: 'Gloria Webster',
    age: 23,
    email: 'ILift@hotmail.com',
    zodiac: 'Cancer',
    gpa: 3.5
  },
  {
    _id: ObjectId("6560f85313ebb626aab0ea1e"),
    name: 'Billy',
    age: 18,
    email: 'huluhaslivesports@gmail.com',
    zodiac: 'Virgo',
    gpa: 3
  },
  {
    _id: ObjectId("6560ee7d13ebb626aab0ea1d"),
    name: 'Barik',
    age: 19,
    email: 'wel0veyouBarik@sweetmail.com',
    zodiac: 'Leo',
    gpa: 4
  }
]
school>
```

Limit Query

- The other query you can connect is limit( ).
- Adding limit( ) to db.students.find( ) will limit the amount of documents queried to whatever number you put in the ( )
- Code Example: db.students.find( ).limit(2)

```
school>  db.students.find().limit(2)
[
  {
    _id: ObjectId("6560ebff13ebb626aab0ea1b"),
    name: 'Jimmy',
    age: 20,
    email: 'boliviarocks@gmail.com',
    zodiac: 'Libra',
    gpa: 3.8
  },
  {
    _id: ObjectId("6560ee7d13ebb626aab0ea1c"),
    name: 'Gloria Webster',
    age: 23,
    email: 'ILift@hotmail.com',
    zodiac: 'Cancer',
    gpa: 3.5
  }
]
school>
```

- You can also add limit( ) to db.students.find( ).sort({}) to limit the amount of sorted documents
- Code Example: db.students.find( ).sort({age:-1}).limit(3)

```
school> db.students.find().sort({age:-1}).limit(3)
[
  {
    _id: ObjectId("6560f85313ebb626aab0ea1f"),
    name: 'Olivander',
    age: 25,
    email: 'wowzerz@hackermail.com',
    zodiac: 'Capricorn',
    gpa: 3.2
  },
  {
    _id: ObjectId("6560ee7d13ebb626aab0ea1c"),
    name: 'Gloria Webster',
    age: 23,
    email: 'ILift@hotmail.com',
    zodiac: 'Cancer',
    gpa: 3.5
  },
  {
    _id: ObjectId("6560ebff13ebb626aab0ea1b"),
    name: 'Jimmy',
    age: 20,
    email: 'boliviarocks@gmail.com',
    zodiac: 'Libra',
    gpa: 3.8
  }
]
school>
```

4. Saving and Ending

- For this next section, we will be moving to exporting data to save outside of MongoDB.
- To export, we have to exit MongoDB into a regular terminal window using the statement: exit

```
school> exit
loganlasell@Logans-MacBook-Pro-4 ~ %
```

- In MongoDB we have to export each collection, one at a time.
- To do this we will use the mongoexport command.
- Pseudo-code: mongoexport --db <database name> --collection <collection name> --out User/<user's name>/<file name>

- --db is used to specify which database you want selected, --collection is used to select a collection, and --out specifies the export destination
- Code Example: mongoexport --db school --collection faculty --out User/loganlasell/Documents

```
loganlasell@Logans-MacBook-Pro-4 ~ % mongoexport --db school --collection faculty --out User/loga
nlasell/Documents
2023-11-29T10:15:57.448-1000    connected to: mongodb://localhost/
2023-11-29T10:15:57.452-1000    exported 5 records
loganlasell@Logans-MacBook-Pro-4 ~ %
```

Case 2:

For this case, we will be following the same rules and methods as Case 1, but this will be without the guiding words.

1. Creating a Database

- We will first create a database using the "use" function
- We will be naming our database "sports" for this case
- Code Example: use <database name>

```
school> use sports
switched to db sports
sports>
```

2. Creating Tables and Inserting Data

Collections

- For these collections we will be naming them "basketball", "football", and "soccer" for this case.
- We will be using the createCollection method.

- Code Example: db.createCollection("<collection name>")

```
sports> db.createCollection("basketball")
{ ok: 1 }
sports>
```

```
sports> db.createCollection("football")
{ ok: 1 }
sports>
```

```
sports> db.createCollection("soccer")
{ ok: 1 }
sports>
```

Documents

- We will now insert documents using the two different methods

Single Document Insertion

- Using the db insertOne({}) method we will insert one document.
- Code example: db.<collection>.insertOne({field name:value})

```
sports> db.basketball.insertOne({team:"Skallywags", players:14, color:"Green", wins:38, losses:1})
{
  acknowledged: true,
  insertedId: ObjectId("6566f49b439fe1763adf326a")
}
sports>
```

Multi-Document Insertion

- Using the db insertMany([{},{}]) method we will insert multiple documents
- Code Example: db.<collection>.insertMany([{},{}])

```
sports> db.football.insertMany([{team:"Hoosiers", players:53, color:"Purple", wins:12, losses:3}, {team:"Horseshoes", players:60, color:"Crimson", wins:10, losses:5}, {team:"Butlers", player
s:49, color:"Black", wins:14, losses:1}, {team:"Curb-Stompers", players:55, color:"Blood-Red", wins:3, losses:12}, {team:"Crows", players:35, color:"Midnight Purple", wins:8, losses:7}])
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId("6566f760439fe1763adf326b"),
    '1': ObjectId("6566f760439fe1763adf326c"),
    '2': ObjectId("6566f760439fe1763adf326d"),
    '3': ObjectId("6566f760439fe1763adf326e"),
    '4': ObjectId("6566f760439fe1763adf326f")
  }
}
sports>
```

Our collections

- Here are our collections of documents in our new database.
- Code Example: db.<collection>.find( )

```
sports> db.basketball.find()
[
  {
    _id: ObjectId("6566f49b439fe1763adf326a"),
    team: 'Skallywags',
    players: 14,
    color: 'Green',
    wins: 38,
    losses: 1
  },
  {
    _id: ObjectId("6566fd26439fe1763adf3270"),
    team: 'Builders',
    players: 15,
    color: 'Yellow',
    wins: 25,
    losses: 14
  },
  {
    _id: ObjectId("6566fd26439fe1763adf3271"),
    team: 'Cold Lava',
    players: 7,
    color: 'Ice White',
    wins: 30,
    losses: 9
  },
  {
    _id: ObjectId("6566fd26439fe1763adf3272"),
    team: 'Heaven Sent',
    players: 12,
    color: 'Gold',
    wins: 38,
    losses: 1
  },
  {
    _id: ObjectId("6566fd26439fe1763adf3273"),
    team: 'Wheaties',
    players: 15,
    color: 'Brown',
    wins: 37,
    losses: 2
  }
]
sports>
```

```
sports> db.football.find()
[
  {
    _id: ObjectId("6566f760439fe1763adf326b"),
    team: 'Hoosiers',
    players: 53,
    color: 'Purple',
    wins: 12,
    losses: 3
  },
  {
    _id: ObjectId("6566f760439fe1763adf326c"),
    team: 'Horseshoes',
    players: 60,
    color: 'Crimson',
    wins: 10,
    losses: 5
  },
  {
    _id: ObjectId("6566f760439fe1763adf326d"),
    team: 'Butlers',
    players: 49,
    color: 'Black',
    wins: 14,
    losses: 1
  },
  {
    _id: ObjectId("6566f760439fe1763adf326e"),
    team: 'Curb-Stompers',
    players: 55,
    color: 'Blood-Red',
    wins: 3,
    losses: 12
  },
  {
    _id: ObjectId("6566f760439fe1763adf326f"),
    team: 'Crows',
    players: 35,
    color: 'Midnight Purple',
    wins: 8,
    losses: 7
  }
]
sports>
```

```
sports> db.soccer.find()
[
  {
    _id: ObjectId("6566fff4439fe1763adf3274"),
    team: 'Pirates',
    players: 25,
    color: 'White',
    wins: 5,
    losses: 15
  },
  {
    _id: ObjectId("6566fff4439fe1763adf3275"),
    team: 'Buckeyes',
    players: 29,
    color: 'Red',
    wins: 14,
    losses: 6
  },
  {
    _id: ObjectId("6566fff4439fe1763adf3276"),
    team: 'PoBoys',
    players: 20,
    color: 'Magenta',
    wins: 12,
    losses: 8
  },
  {
    _id: ObjectId("6566fff4439fe1763adf3277"),
    team: 'Necromancers',
    players: 26,
    color: 'Light Green',
    wins: 19,
    losses: 1
  },
  {
    _id: ObjectId("6566fff4439fe1763adf3278"),
    team: 'Hitters',
    players: 30,
    color: 'Baby Blue',
    wins: 10,
    losses: 10
  }
]
sports>
```

3. Querying Data

Databases and Collections

- Now, we will start querying data.
- As before, we will start with querying databases
- Code Example: show dbs

```
sports> show dbs
admin     40.00 KiB
config    96.00 KiB
local     72.00 KiB
school   152.00 KiB
sports   152.00 KiB
sports>
```

- Next, we will query collections
- Code Example: show collections

```
sports> show collections
basketball
football
soccer
sports>
```

Document Query

- Following that, we will now do a document query
- Pseudo-Code: db.<collection name>.find( )
- Code Example: db.basketball.find( )

```
sports> db.basketball.find()
[
  {
    _id: ObjectId("6566f49b439fe1763adf326a"),
    team: 'Skallywags',
    players: 14,
    color: 'Green',
    wins: 38,
    losses: 1
  },
  {
    _id: ObjectId("6566fd26439fe1763adf3270"),
    team: 'Builders',
    players: 15,
    color: 'Yellow',
    wins: 25,
    losses: 14
  },
  {
    _id: ObjectId("6566fd26439fe1763adf3271"),
    team: 'Cold Lava',
    players: 7,
    color: 'Ice White',
    wins: 30,
    losses: 9
  },
  {
    _id: ObjectId("6566fd26439fe1763adf3272"),
    team: 'Heaven Sent',
    players: 12,
    color: 'Gold',
    wins: 38,
    losses: 1
  },
  {
    _id: ObjectId("6566fd26439fe1763adf3273"),
    team: 'Wheaties',
    players: 15,
    color: 'Brown',
    wins: 37,
    losses: 2
  }
]
sports>
```

Field Search Query

- Once again to specify our query, we will use a field query.
- Pseudo-code: db.<collection name>.find({field name:value})
- Code Example: db.basketball.find({team: “Skallywags”})

```
sports> db.basketball.find({team:"Skallywags"})
[
  {
    _id: ObjectId("6566f49b439fe1763adf326a"),
    team: 'Skallywags',
    players: 14,
    color: 'Green',
    wins: 38,
    losses: 1
  }
]
sports>
```

Multi-Field Search Query

- To continue, we will next do a multi-field query.
- Pseudo-code: db.<collection name>.find({field name:value, field name:value})
- Code Example: db.soccer.find({players:25, wins:5})

```
sports> db.soccer.find({players:25, wins:5})
[
  {
    _id: ObjectId("6566fff4439fe1763adf3274"),
    team: 'Pirates',
    players: 25,
    color: 'White',
    wins: 5,
    losses: 15
  }
]
sports>
```

Field Projection

- Next, we will use a field projection.
- Once again we will query them by labeling them true or false
- Pseudo-code: db.<collection name>.find({}, {field name:true/false})

- Code Example: db.football.find({}, {team:true})

```
sports> db.football.find({}, {team:true})
[
  { _id: ObjectId("6566f760439fe1763adf326b"), team: 'Hoosiers' },
  { _id: ObjectId("6566f760439fe1763adf326c"), team: 'Horseshoes' },
  { _id: ObjectId("6566f760439fe1763adf326d"), team: 'Butlers' },
  { _id: ObjectId("6566f760439fe1763adf326e"), team: 'Curb-Stompers' },
  { _id: ObjectId("6566f760439fe1763adf326f"), team: 'Crows' }
]
sports>
```

Multi-Field Projection

- Additionally, like we've used before we can add more specificity to this query.
- Pseudo-code: db.<collection name>.find({}, {field name:true/false, field name:true/false})
- Code Example: db.football.find({}, {team:true, _id:false, wins:true})

```
sports> db.football.find({}, {team:true, _id:false, wins:true})
[
  { team: 'Hoosiers', wins: 12 },
  { team: 'Horseshoes', wins: 10 },
  { team: 'Butlers', wins: 14 },
  { team: 'Curb-Stompers', wins: 3 },
  { team: 'Crows', wins: 8 }
]
sports>
```

Sort Query

- The next we will use to search is the sort query.
- Pseudo-code: db.<collection name>.find( ).sort({field name: 1/-1}) (do both 1 and -1 for alphabet)
- Code Example1: db.basketball.find( ).sort({team: 1})

```
sports> db.basketball.find().sort({team:1})
[
  {
    _id: ObjectId("6566fd26439fe1763adf3270"),
    team: 'Builders',
    players: 15,
    color: 'Yellow',
    wins: 25,
    losses: 14
  },
  {
    _id: ObjectId("6566fd26439fe1763adf3271"),
    team: 'Cold Lava',
    players: 7,
    color: 'Ice White',
    wins: 30,
    losses: 9
  },
  {
    _id: ObjectId("6566fd26439fe1763adf3272"),
    team: 'Heaven Sent',
    players: 12,
    color: 'Gold',
    wins: 38,
    losses: 1
  },
  {
    _id: ObjectId("6566f49b439fe1763adf326a"),
    team: 'Skallywags',
    players: 14,
    color: 'Green',
    wins: 38,
    losses: 1
  },
  {
    _id: ObjectId("6566fd26439fe1763adf3273"),
    team: 'Wheaties',
    players: 15,
    color: 'Brown',
    wins: 37,
    losses: 2
  }
]
sports>
```

- Code Example2: db.basketball.find( ).sort({team:-1})

```
sports> db.basketball.find().sort({team:-1})
[
  {
    _id: ObjectId("6566fd26439fe1763adf3273"),
    team: 'Wheaties',
    players: 15,
    color: 'Brown',
    wins: 37,
    losses: 2
  },
  {
    _id: ObjectId("6566f49b439fe1763adf326a"),
    team: 'Skallywags',
    players: 14,
    color: 'Green',
    wins: 38,
    losses: 1
  },
  {
    _id: ObjectId("6566fd26439fe1763adf3272"),
    team: 'Heaven Sent',
    players: 12,
    color: 'Gold',
    wins: 38,
    losses: 1
  },
  {
    _id: ObjectId("6566fd26439fe1763adf3271"),
    team: 'Cold Lava',
    players: 7,
    color: 'Ice White',
    wins: 30,
    losses: 9
  },
  {
    _id: ObjectId("6566fd26439fe1763adf3270"),
    team: 'Builders',
    players: 15,
    color: 'Yellow',
    wins: 25,
    losses: 14
  }
]
sports>
```

Limit Query

- The final query type we will do is the limit query.
- Pseudo-code: db.<collection name>.find( ).limit( )
- Code Example: db.football.find( ).limit(2)

```
sports> db.football.find().limit(2)
[
  {
    _id: ObjectId("6566f760439fe1763adf326b"),
    team: 'Hoosiers',
    players: 53,
    color: 'Purple',
    wins: 12,
    losses: 3
  },
  {
    _id: ObjectId("6566f760439fe1763adf326c"),
    team: 'Horseshoes',
    players: 60,
    color: 'Crimson',
    wins: 10,
    losses: 5
  }
]
sports>
```

- And just as before, we can connect a sort and find query together.
- Pseudo-code: db<collection name>. find( ).sort({}).limit( )
- Code Example: db.soccer.find( ).sort({wins:1}).limit(3)

```
sports> db.soccer.find().sort({wins:1}).limit(3)
[
  {
    _id: ObjectId("6566fff4439fe1763adf3274"),
    team: 'Pirates',
    players: 25,
    color: 'White',
    wins: 5,
    losses: 15
  },
  {
    _id: ObjectId("6566fff4439fe1763adf3278"),
    team: 'Hitters',
    players: 30,
    color: 'Baby Blue',
    wins: 10,
    losses: 10
  },
  {
    _id: ObjectId("6566fff4439fe1763adf3276"),
    team: 'PoBoys',
    players: 20,
    color: 'Magenta',
    wins: 12,
    losses: 8
  }
]
sports>
```

4.Saving and Ending

- We can now finish and save our data from this database.
- Pseudo-code: mongoexport --db <db name> --collection <collection name> --out User/<user name>/file name
- Code Example: mongoexport --db sports --collection basketball --out User/loganlasell/Downloads

```
school> exit
loganlasell@Logans-MacBook-Pro-4 ~ % mongoexport --db sports --collection basketball --out User/l
oganlasell/Downloads
2023-11-29T10:46:27.970-1000    connected to: mongodb://localhost/
2023-11-29T10:46:27.974-1000    exported 5 records
loganlasell@Logans-MacBook-Pro-4 ~ %
```

Summary

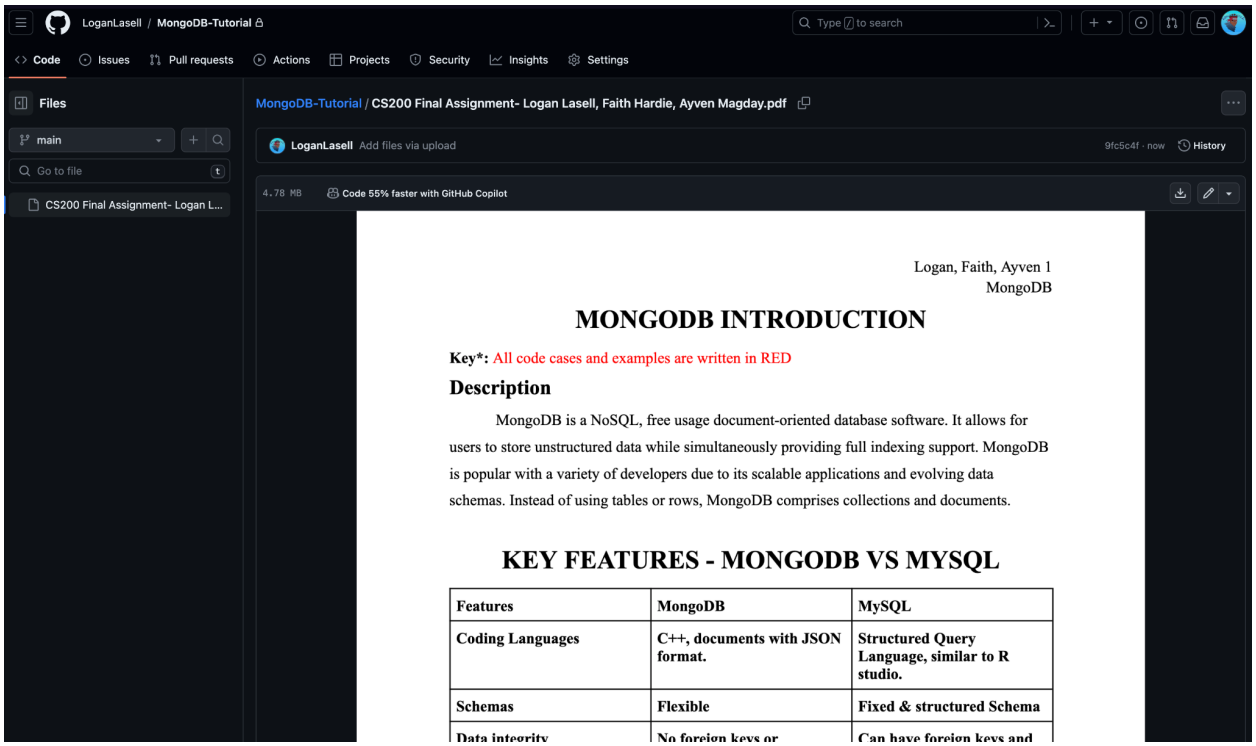
In summary, MongoDB is useful for developers who work with evolving data structures as it allows you to store and manage data without a predefined schema, unlike MySQL. MongoDB is also more practical for handling large data sets compared to MySQL because of its design to share data horizontally across servers. MongoDB uses a document format similar to JSON which allows users to work with their data similar to their codes.

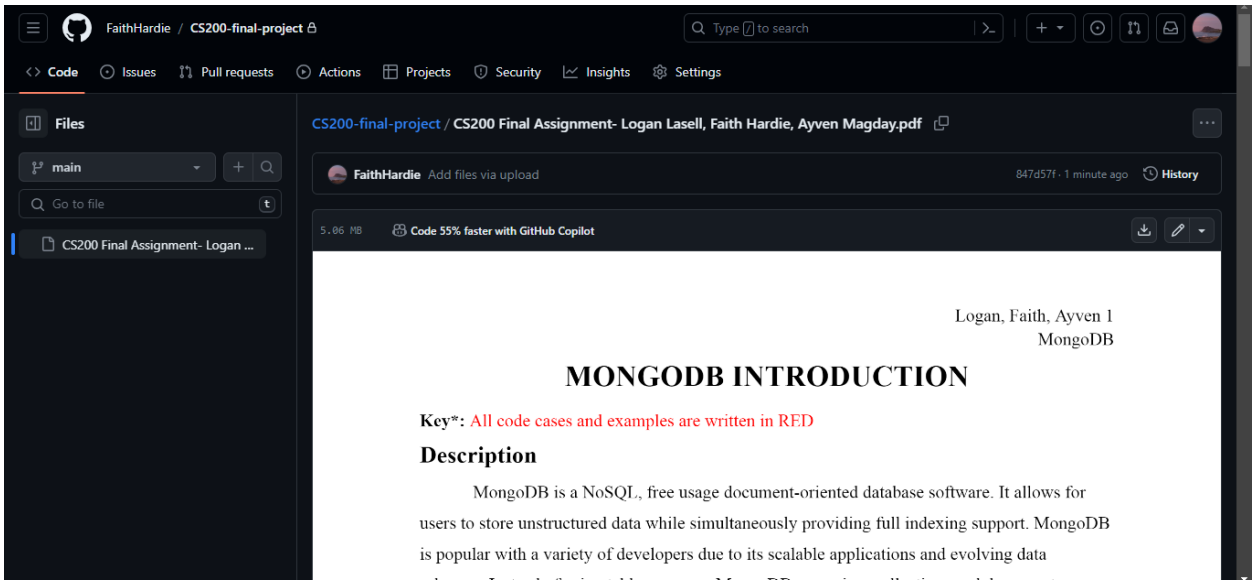
(Optional) Useful Links

(MongoDB download)

https://www.mongodb.com/try/download/community (*Download MongoDB Community Server*, n.d.)

Github:





References

Install MongoDB Community Edition on macOS — MongoDB Manual. (n.d.). MongoDB. Retrieved November 15, 2023, from https://www.mongodb.com/docs/manual/tutorial/install-mongodb-on-os-x/#install-mongodb-community-edition

Install MongoDB Community Edition on Windows — MongoDB Manual. (n.d.) MongoDB. Retrieved November 15, 2023, from https://www.mongodb.com/docs/manual/tutorial/install-mongodb-on-windows/

Learn MongoDB in 1 Hour 🍃 (2023). (2023, April 14). YouTube. Retrieved November 24, 2023, from https://www.youtube.com/watch?v=c2M-rlkkT5o

Ultimate Guide MongoDB: Definition, Advantages & Disadvantages. (n.d.). KnowledgeNile. Retrieved November 27, 2023, from https://www.knowledgenile.com/blogs/pros-and-cons-of-mongodb

What Is MongoDB? (n.d.). MongoDB. Retrieved November 27, 2023, from https://www.mongodb.com/what-is-mongodb