



Victoria Delacruz, Lydia Hefel, Sadie Piianiaia, Ashley Holen  
CS200 Final Project

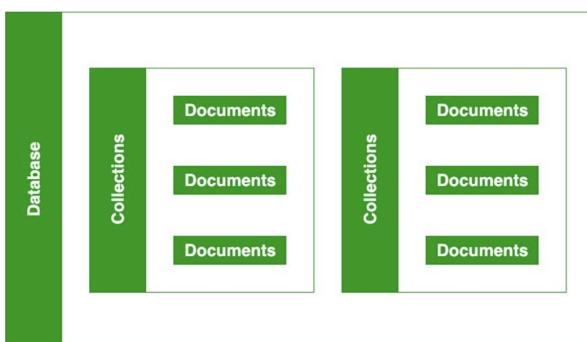
## Description of MongoDB

MongoDB is a database management system that is very popular in industry. MongoDB is an open-source NoSQL (non-relational) system that is designed to handle large volumes of unstructured or semi-structured data. It is a document-oriented database and is part of the broader family of NoSQL databases, which are known for their flexibility, scalability, and performance.

## Key Features Compared to SQL

- The primary difference between MongoDB and SQL is the structure. SQL databases are used to store structured data while NoSQL databases like MongoDB are used to save unstructured data.
- MongoDB is used to save unstructured data in JSON format, while SQL does not
- MongoDB does not support advanced analytics such as filters, joins, merge, and aggregation
- In SQL, data is saved in a row-column format known as a *Table* and can be retrieved using queries formatted in the Structured Query Language (SQL). In MongoDB, data is stored in *collections* that are analogous to MySQL tables. A collection can consist of many documents in which data is stored in JSON format.
- The SQL databases have a relational property where different tables are related to each other with foreign keys and primary keys. In MongoDB, we cannot create a relationship between collections.
- SQL requires a predefined schema (structure), while in MongoDB there is not.

## MongoDB Data Structure:



## Installation and Setup

- 1) In your primary search engine, search “install mongodb”



- 2) Go to the official MongoDB website



### Install MongoDB

This section of the manual contains information on **installing MongoDB**. For instructions on upgrading your current deployment to MongoDB 7.0, ...

[MongoDB Community](#) · [Install MongoDB Community...](#) · [Install MongoDB Enterprise](#)

Link :(<https://www.mongodb.com/docs/manual/installation/>)

- 3) Tutorials are available for Linux, macOS, Windows, and Docker platforms. Also, for both the Community Edition and Enterprise Edition of MongoDB.

### MongoDB Installation Tutorials

MongoDB installation tutorials are available for the following platforms, for both the Community Edition and the [Enterprise Edition](#):

Platform	Community Edition	Enterprise Edition
Linux	<a href="#">Install MongoDB Community Edition on Red Hat or CentOS</a> <a href="#">Install MongoDB Community Edition on Ubuntu</a> <a href="#">Install MongoDB Community Edition on Debian</a> <a href="#">Install MongoDB Community Edition on SUSE</a> <a href="#">Install MongoDB Community Edition on Amazon Linux</a>	<a href="#">Install MongoDB Enterprise Edition on Red Hat or CentOS</a> <a href="#">Install MongoDB Enterprise Edition on Ubuntu</a> <a href="#">Install MongoDB Enterprise Edition on Debian</a> <a href="#">Install MongoDB Enterprise Edition on SUSE</a> <a href="#">Install MongoDB Enterprise Edition on Amazon Linux</a>
macOS	<a href="#">Install MongoDB Community Edition on macOS</a>	<a href="#">Install MongoDB Enterprise on macOS</a>
Windows	<a href="#">Install MongoDB Community Edition on Windows</a>	<a href="#">Install MongoDB Enterprise Edition on Windows</a>
Docker	<a href="#">Install MongoDB Community with Docker</a>	<a href="#">Install MongoDB Enterprise with Docker</a>

- 4) Click on the version of MongoDB compatible with your system. For this tutorial, we will be demonstrating how to install and use MongoDB on a Mac.

macOS      [Install MongoDB Community Edition on macOS](#)      [Install MongoDB Enterprise on macOS](#)

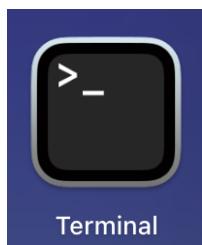
- 5) Select to download the Community edition on macOS. You will also need to select the Community edition that supports the version of your operating system. In this case, we are installing an edition that supports macOS 11 or later.

## Install MongoDB Community Edition on macOS

### Platform Support

MongoDB 7.0 Community Edition supports macOS 11 or later.

- 6) To start the download, you will need the app “Xcode” on your mac. You can download this application through the App Store, but we will show you how to download through your computer’s terminal. Below is what the application for Terminal looks like. If you cannot find the application, look in the Launchpad of your computer.



#### Install Xcode Command-Line Tools

Homebrew requires the Xcode command-line tools from Apple's Xcode.

- Install the Xcode command-line tools by running the following command in your macOS Terminal:

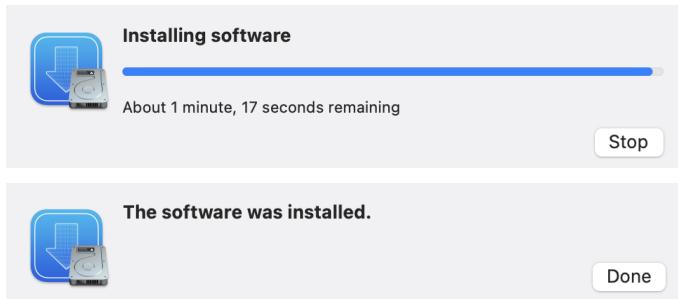
```
xcode-select --install
```



- 7) Once you open Terminal, type “xcode-select -- install”

```
Last login: Fri Nov  3 09:29:47 on ttys002
(base) lydiahefel@Lydias-MacBook-Air ~ % xcode-select --install
xcode-select: note: install requested for command line developer tools
(base) lydiahefel@Lydias-MacBook-Air ~ %
```

- 8) As shown above, there are no errors and the installation code ran through. A pop-up should appear to demonstrate that the application is installed.



- 9) We will now need to install “Homebrew”, which along with Xcode, is another prerequisite that you will need to have downloaded before MongoDB. There are instructions on the official MongoDB website. We will also be installing this application through the Terminal.

## Install Homebrew

macOS does not include the Homebrew `brew` package by default.

- Install `brew` using the official [Homebrew installation instructions](#).



Enter in the code shown above into your Terminal.

- 10) After entering in your code, you will be asked to enter the password to your computer into the Terminal. As you type, the password will not appear, but once you have typed it, press the enter key to run the code.

```
Last login: Fri Nov  3 09:29:47 on ttys002
[(base) lydiahefel@Lydias-MacBook-Air ~ % xcode-select --install
xcode-select: note: install requested for command line developer tools
[(base) lydiahefel@Lydias-MacBook-Air ~ % /bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
==> Checking for `sudo` access (which may request your password)...
Password: ]
```

- 11) Once you have entered your password, you have granted access for Homebrew to be installed. The image below will appear to show which scripts and directories will be installed.

```
==> Checking for `sudo` access (which may request your password)...
[Password:
==> This script will install:
/opt/homebrew/bin/brew
/opt/homebrew/share/doc/homebrew
/opt/homebrew/share/man/man1/brew.1
/opt/homebrew/share/zsh/site-functions/_brew
/opt/homebrew/etc/bash_completion.d/brew
/opt/homebrew
==> The following new directories will be created:
/opt/homebrew/bin
/opt/homebrew/etc
/opt/homebrew/include
/opt/homebrew/lib
/opt/homebrew/sbin
/opt/homebrew/share
/opt/homebrew/var
/opt/homebrew/opt
/opt/homebrew/share/zsh
/opt/homebrew/share/zsh/site-functions
/opt/homebrew/var/homebrew
/opt/homebrew/var/homebrew/linked
/opt/homebrew/Cellar
/opt/homebrew/Caskroom
/opt/homebrew/Frameworks
```

- 12) Next, we need to run the official Homebrew formula in the Terminal. “brew tap mongodb/brew”

Tap the [MongoDB Homebrew Tap](#) to download the official Homebrew formula for MongoDB and the Database Tools, by running the following command in your macOS Terminal:

```
brew tap mongodb/brew
```



- 13) Next, we need to update Homebrew. Type “brew update” into the terminal.

To update Homebrew and all existing formulae:

```
brew update
```



- 14) Now, we can run MongoDB as a macOS service by typing the command below into your terminal.

- To run MongoDB (i.e. the `mongod` process) **as a macOS service**, run:

```
brew services start mongodb-community@7.0
```



- 15) Install MongoDB by running the following command into the Terminal.

To install MongoDB, run the following command in your macOS Terminal application:

```
brew install mongodb-community@7.0
```



- 16) To connect and use MongoDB, run the following command in your Terminal, “mongosh”

### Connect and Use MongoDB

To begin using MongoDB, connect [mongosh](#) to the running instance. From a new terminal, issue the following:

```
mongosh
```



- 17) This will prompt an installation process, which can be seen below.

```
Press RETURN/ENTER to continue or any other key to abort:  
=> /usr/bin/sudo /usr/bin/install -d -o root -g wheel -m 0755 /opt/homebrew  
=> /usr/bin/sudo /bin/mkdir -p /opt/homebrew/bin /opt/homebrew/etc /opt/homebrew/include /opt/homebrew/lib /opt/homebrew/sbin /opt/homebrew/share /opt/homebrew/var /opt/homebrew/opt /opt/homebrew/share/zsh /opt/homebrew/share/zsh/site-functions /opt/homebrew/var/homebrew /opt/homebrew/var/homebrew/linked /opt/homebrew/Cellar /opt/homebrew/Caskroom /opt/homebrew/Frameworks  
=> /usr/bin/sudo /bin/chmod ug=rwx /opt/homebrew/bin /opt/homebrew/etc /opt/homebrew/include /opt/homebrew/lib /opt/homebrew/sbin /opt/homebrew/share /opt/homebrew/var /opt/homebrew/opt /opt/homebrew/share/zsh /opt/homebrew/share/zsh/site-functions /opt/homebrew/var/homebrew /opt/homebrew/var/homebrew/linked /opt/homebrew/Cellar /opt/homebrew/Caskroom /opt/homebrew/Frameworks  
=> /usr/bin/sudo /bin/chmod go-w /opt/homebrew/share/zsh /opt/homebrew/share/zsh/site-functions  
=> /usr/bin/sudo /usr/sbin/chown lydiahefel /opt/homebrew/bin /opt/homebrew/etc /opt/homebrew/include /opt/homebrew/lib /opt/homebrew/sbin /opt/homebrew/share /opt/homebrew/var /opt/homebrew/opt /opt/homebrew/share/zsh /opt/homebrew/share/zsh/site-functions /opt/homebrew/var/homebrew /opt/homebrew/var/homebrew/linked /opt/homebrew/Cellar /opt/homebrew/Caskroom /opt/homebrew/Frameworks  
=> /usr/bin/sudo /usr/bin/chgrp admin /opt/homebrew/bin /opt/homebrew/etc /opt/homebrew/include /opt/homebrew/lib /opt/homebrew/sbin /opt/homebrew/share /opt/homebrew/var /opt/homebrew/opt /opt/homebrew/share/zsh /opt/homebrew/share/zsh/site-functions /opt/homebrew/var/homebrew /opt/homebrew/var/homebrew/linked /opt/homebrew/Cellar /opt/homebrew/Caskroom /opt/homebrew/Frameworks  
=> /usr/bin/sudo /usr/sbin/chown -R lydiahefel:admin /opt/homebrew  
=> /usr/bin/sudo /bin/mkdir -p /Users/lydiahefel/Library/Caches/Homebrew  
=> /usr/bin/sudo /bin/chmod g+rwx /Users/lydiahefel/Library/Caches/Homebrew  
=> /usr/bin/sudo /usr/sbin/chown -R lydiahefel /Users/lydiahefel/Library/Caches/Homebrew  
=> Downloading and installing Homebrew...  
remote: Enumerating objects: 249442, done.  
remote: Counting objects: 100% (934/934), done.  
remote: Compressing objects: 100% (549/549), done.  
Receiving objects: 22% (55589/249442), 26.92 MiB | 186.00 KiB/s  
  
remote: Total 249442 (delta 414), reused 850 (delta 358), pack-reused 248508  
Receiving objects: 100% (249442/249442), 73.30 MiB | 164.00 KiB/s, done.  
Resolving deltas: 100% (181997/181997), done.  
From https://github.com/Homebrew/brew  
 * [new branch]           master      -> origin/master  
 * [new tag]              0.1         -> 0.1
```

```

remote: Enumerating objects: 18, done.
remote: Counting objects: 100% (11/11), done.
remote: Total 18 (delta 11), reused 11 (delta 11), pack-reused 7
Unpacking objects: 100% (18/18), 3.38 KiB | 173.00 KiB/s, done.
From https://github.com/Homebrew/brew
 * [new tag]      4.0.29    -> 4.0.29
 * [new tag]      4.1.9     -> 4.1.9
HEAD is now at d72ee37f6 Merge pull request #16207 from Rylan12/api-tap-migrations
Warning: /opt/homebrew/bin is not in your PATH.
  Instructions on how to configure your shell for Homebrew
  can be found in the 'Next steps' section below.
==> Installation successful!

==> Homebrew has enabled anonymous aggregate formulae and cask analytics.
Read the analytics documentation (and how to opt-out) here:
  https://docs.brew.sh/Analytics
No analytics data has been sent yet (nor will any be during this install run).

==> Homebrew is run entirely by unpaid volunteers. Please consider donating:
  https://github.com/Homebrew/brew#donations

==> Next steps:
- Run these two commands in your terminal to add Homebrew to your PATH:
  (echo; echo 'eval "$((/opt/homebrew/bin/brew shellenv))"') >> /Users/lydiahefel/.zprofile
  eval "$((/opt/homebrew/bin/brew shellenv))"
- Run brew help to get started
- Further documentation:
  https://docs.brew.sh

```

- 18) Once you see “**Installation Successful**” MongoDB should be installed! Under “**Next Steps**” in the above photo, we need to run two commands in our Terminal to add Homebrew to our Path. The name of the user for your computer should automatically appear, and you should simply be able to copy and paste the two commands.

```
(base) lydiahefel@Lydias-MacBook-Air ~ % (echo; echo 'eval "$((/opt/homebrew/bin/brew shellenv))"') >> /Users/lydiahefel/.zprofile
[ eval "$((/opt/homebrew/bin/brew shellenv)"
(base) lydiahefel@Lydias-MacBook-Air ~ % ]
```

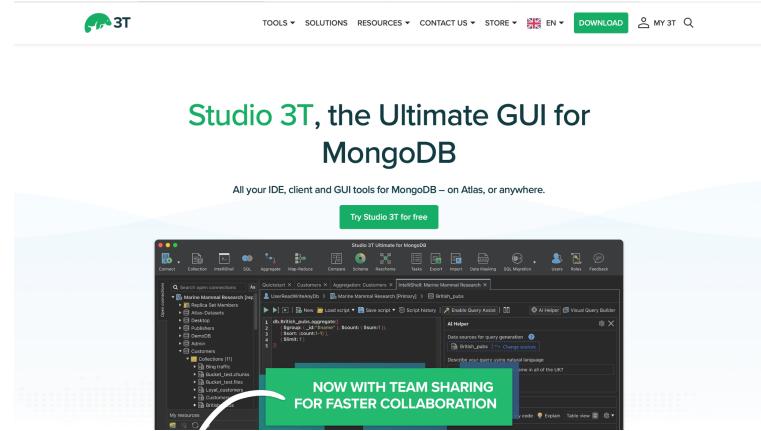
- 19) Your path should now appear, showing the name of the MongoDB you installed, the status should show “Started” in green, your computer user, and the File which the software is stored in your computer.

```
[base) lydiahefel@Lydias-MacBook-Air ~ % brew services
Name          Status  User      File
mongodb-community  started lydiahefel ~/Library/LaunchAgents/homebrew.mxcl.mongodb-community.plist
```

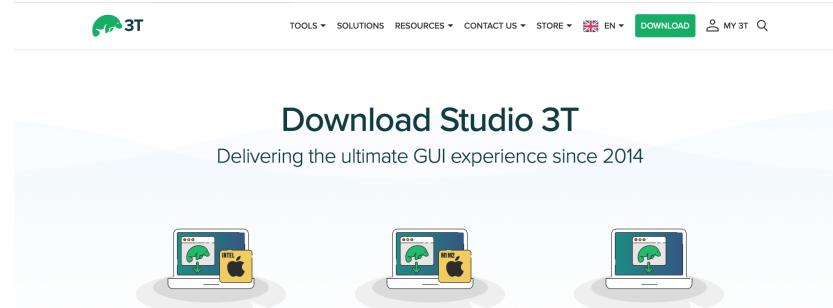
- 20) One last command that we will need to run in our Terminal will be “mongod”. The installation is shown below.

```
(base) lydiasfe10@lydias-MacBook-Air ~ % mongod
{"t":1,"$date":"2023-11-13T10:14:06.579-10:00"}, {"s":1,"c":"NETWORK", "id":4915781, "ctx":"thread1", "msg":"Initialized wire specification", "attr":{"spec":{"incomingExternalClient":{"minWireVersion":0,"maxWireVersion":22}, "outgoing":{"minWireVersion":0,"maxWireVersion":22}, "isInternalClient":true}}}, {"t":2,"$date":"2023-11-13T10:14:06.579-10:00"}, {"s":1,"c":"CONTROL", "id":23285, "ctx":"thread1", "msg":"Automatically disabling TLS 1.0, to force-enable TLS 1.0 specify --sslDisabledProtocols 'none'"}, {"t":3,"$date":"2023-11-13T10:14:06.579-10:00"}, {"s":1,"c":"NETWORK", "id":14648602, "ctx":"thread1", "msg":"Implicit TCP FastOpen in use."}, {"t":4,"$date":"2023-11-13T10:14:06.579-10:00"}, {"s":1,"c":"REPL", "id":5123088, "ctx":"thread1", "msg":"Successfully registered PrimaryOnlyService", "attr":{"service": "TenantMigrationDonorService", "name": "config-tenantMigrationDonors"}}, {"t":5,"$date":"2023-11-13T10:14:06.579-10:00"}, {"s":1,"c":"REPL", "id":5123088, "ctx":"thread1", "msg":"Successfully registered PrimaryOnlyService", "attr":{"service": "TenantMigrationRecipientService", "name": "config-tenantMigrationRecipients"}}, {"t":6,"$date":"2023-11-13T10:14:06.579-10:00"}, {"s":1,"c":"CONTROL", "id":5945603, "ctx":"thread1", "msg":"Multi threading initialized"}, {"t":7,"$date":"2023-11-13T10:14:06.579-10:00"}, {"s":1,"c":"REPL", "id":7891600, "ctx":"thread1", "msg":"Starting TenantMigrationAccessBlockerRegistry"}, {"t":8,"$date":"2023-11-13T10:14:06.579-10:00"}, {"s":1,"c":"CONTROL", "id":14615611, "ctx":"initandlisten", "msg":"MongoDB starting", "attr":{"pid":67468, "dbPath":"/data/db", "architecture": "64-bit", "host": "lydias-MacBook-Air.local"}}, {"t":9,"$date":"2023-11-13T10:14:06.579-10:00"}, {"s":1,"c":"CONTROL", "id":23403, "ctx":"initandlisten", "msg":"Build Info", "attr":{"buildInfo":{"version": "7.0.2", "gitVersion": "02b3c655e1302209ef046da6b0a3ef5749d0062", "modules": "1"}, "alloc_size": "system", "environment": "distarch=arch64", "target_arch": "arch64"}}, {"t":10,"$date":"2023-11-13T10:14:06.579-10:00"}, {"s":1,"c":"CONTROL", "id":14615615, "ctx":"initandlisten", "msg":"Operating System", "attr":{"os": "Mac OS X", "version": "22.6.0"}}, {"t":11,"$date":"2023-11-13T10:14:06.579-10:00"}, {"s":1,"c":"CONTROL", "id":14615615, "ctx":"initandlisten", "msg": "Options set by command line", "attr": {"options": []}}, {"t":12,"$date":"2023-11-13T10:14:06.579-10:00"}, {"s":1,"c":"CONTROL", "id":7891600, "ctx":"initandlisten", "msg": "Asio socket.set_option failed with std::system_error", "attr": {"note": "acceptor TCP fast open", "option": {"level": 6, "name": "261", "data": "00 04 00 00"}, "error": {"what": "set_option: Invalid argument", "message": "Invalid argument", "category": "asio::system_error", "value": 22}}}, {"t":13,"$date":"2023-11-13T10:14:06.579-10:00"}, {"s":1,"c":"CONTROL", "id":20568, "ctx":"initandlisten", "msg": "Error setting up listener", "attr": {"error": {"code": 9001, "codeName": "SocketException", "errm_sg": "setsockopt :: caused by: Address already in use"}}, {"t":14,"$date":"2023-11-13T10:14:06.579-10:00"}, {"s":1,"c":"REPL", "id":4784980, "ctx":"initandlisten", "msg": "Stepping down the ReplicationCoordinator for shutdown", "attr": {"waitTimeMillis": 15000}}, {"t":15,"$date":"2023-11-13T10:14:06.579-10:00"}, {"s":1,"c":"CONTROL", "id":14615615, "ctx":"initandlisten", "msg": "Shutting down the replSetEntirePrimary"}, {"t":16,"$date":"2023-11-13T10:14:06.579-10:00"}, {"s":1,"c":"CONTROL", "id":6374601, "ctx":"initandlisten", "msg": "Shutting down the FFL Crud thread pool"}, {"t":17,"$date":"2023-11-13T10:14:06.579-10:00"}, {"s":1,"c":"COMMAND", "id":7891600, "ctx":"initandlisten", "msg": "Shutting down the MirrorMaestro"}, {"t":18,"$date":"2023-11-13T10:14:06.579-10:00"}, {"s":1,"c":"SHARDING", "id":14784982, "ctx":"initandlisten", "msg": "Shutting down the shardForMajorityService"}, {"t":19,"$date":"2023-11-13T10:14:06.579-10:00"}, {"s":1,"c":"NETWORK", "id":14784985, "ctx":"initandlisten", "msg": "Shutting down the global connection pool"}, {"t":20,"$date":"2023-11-13T10:14:06.579-10:00"}, {"s":1,"c":"NETWORK", "id":14784918, "ctx":"initandlisten", "msg": "Shutting down the ReplicaSetMonitor"}, {"t":21,"$date":"2023-11-13T10:14:06.579-10:00"}, {"s":1,"c":"COMMAND", "id":14784921, "ctx":"initandlisten", "msg": "Shutting down the MigrationUtilExecutor"}, {"t":22,"$date":"2023-11-13T10:14:06.579-10:00"}, {"s":1,"c":"COMMAND", "id":14784932, "ctx": "MigrationUtil::TaskRouter", "msg": "Killing all outstanding egress activity."}, {"t":23,"$date":"2023-11-13T10:14:06.579-10:00"}, {"s":1,"c":"COMMAND", "id":14784933, "ctx": "MigrationUtil::TaskRouter", "msg": "Waiting for all pending point-to-point tasks to complete"}, {"t":24,"$date":"2023-11-13T10:14:06.579-10:00"}, {"s":1,"c":"CONTROL", "id":4784928, "ctx": "initandlisten", "msg": "Shutting down the TTL monitor"}, {"t":25,"$date":"2023-11-13T10:14:06.579-10:00"}, {"s":1,"c":"CONTROL", "id":16278511, "ctx": "initandlisten", "msg": "Shutting down the Change Stream Expired Pre-images Remover"}, {"t":26,"$date":"2023-11-13T10:14:06.579-10:00"}, {"s":1,"c":"CONTROL", "id":14784929, "ctx": "initandlisten", "msg": "Acquiring the global lock for shutdown"}, {"t":27,"$date":"2023-11-13T10:14:06.579-10:00"}, {"s":1,"c":"CONTROL", "id":14784931, "ctx": "initandlisten", "msg": "Dropping the scope cache for shutdown"}, {"t":28,"$date":"2023-11-13T10:14:06.579-10:00"}, {"s":1,"c":"CONTROL", "id":20565, "ctx": "initandlisten", "msg": "Now exiting"}, {"t":29,"$date":"2023-11-13T10:14:06.579-10:00"}, {"s":1,"c": "CONTROL", "id": 23138, "ctx": "initandlisten", "msg": "Shutting down", "attr": {"exitCode": 48}}
```

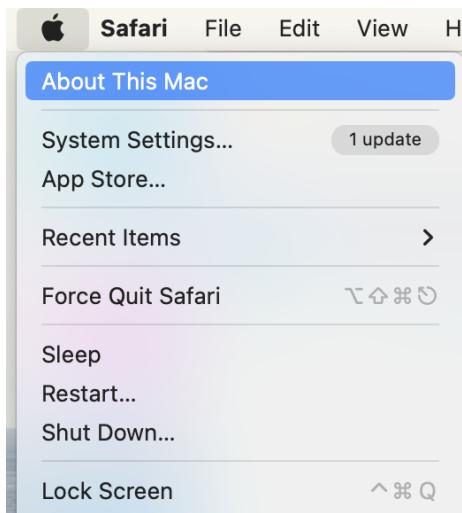
- 21) Now that the software is installed on our computer, we will need to install a user interface to actually use MongoDB. We found that the best interface to do so on Mac is called “Studio 3T”. First, go to <https://studio3t.com/>



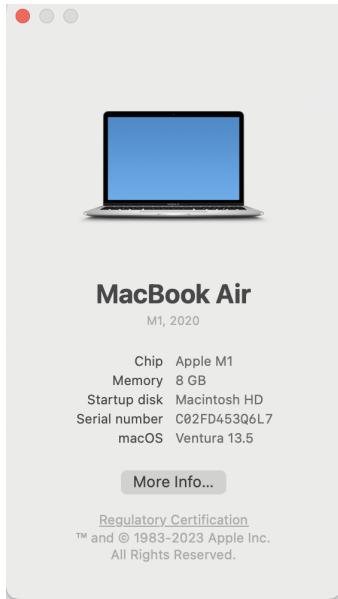
- 22) Click on the green “Download” button on the top right side of the screen. You will have the option to download versions of Studio 3T based on your system.



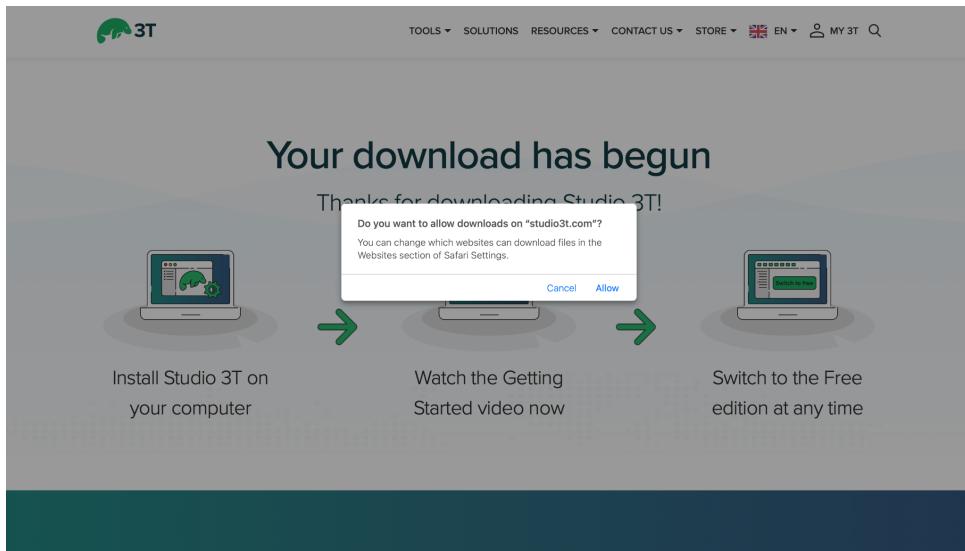
23) To check which system that you have, click the Apple logo on the top left of the screen, and “About This Mac”.



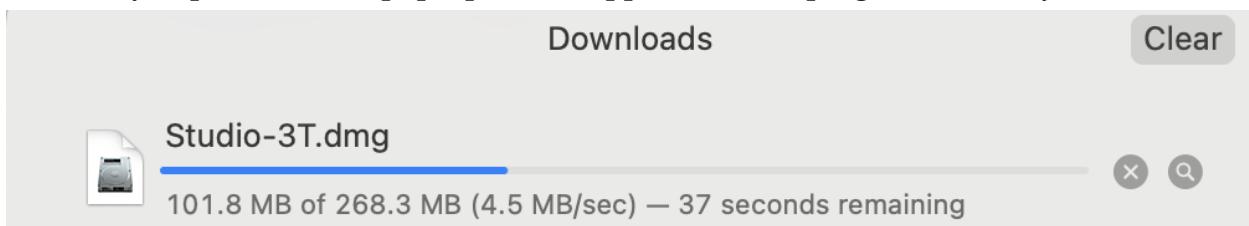
24) In this example, we can see that we have a M1 chip, so we will install the Apple Silicon version of Studio 3T.



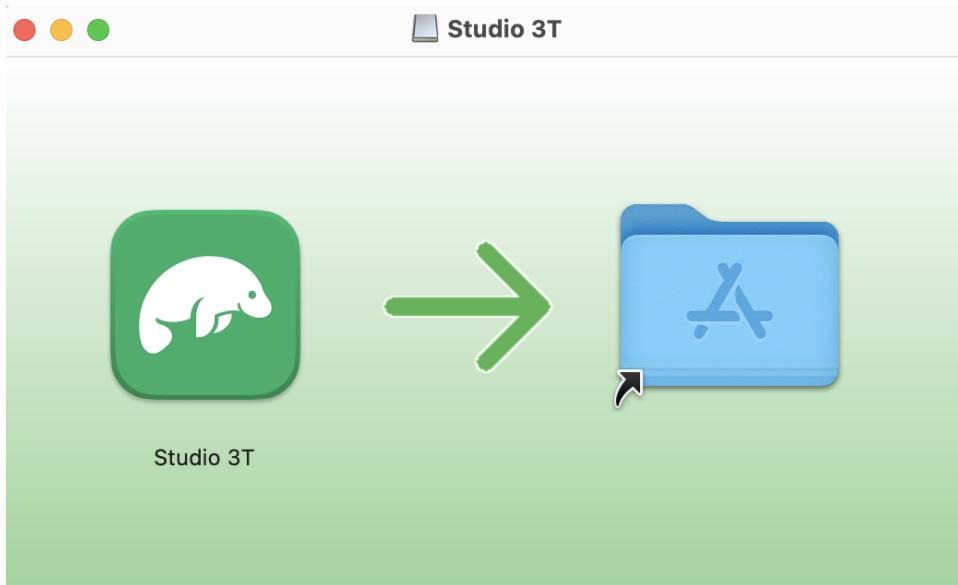
25) Click Download for the Studio 3T which corresponds to your system, and press allow downloads.



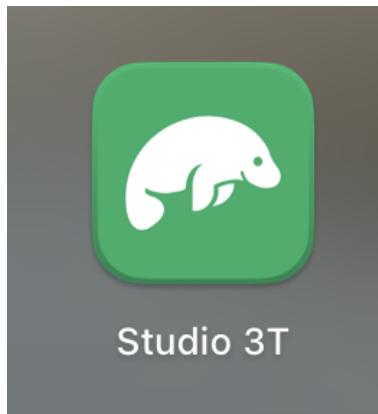
26) Once you press Allow, a pop-up should appear in the top right corner of your screen.



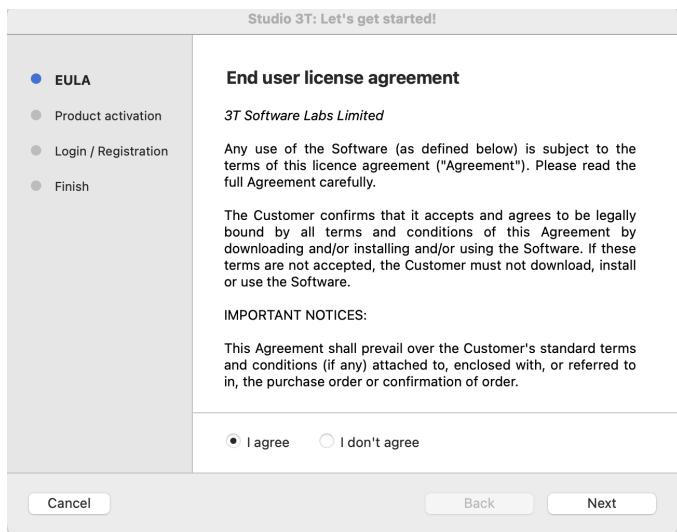
27) Once this download is completed, double click on the downloaded file. The image below should appear. Drag and drop the Studio 3T logo into your Applications folder.



28) Studio 3T should now be available in your Applications on your computer. Open the app.



29) Accept the user agreement which pops up once you first open the application.



- 80) Now it is time to activate your account. Since we are new to MongoDB, we will need to click “Sign Up”.



The screenshot shows the 3T sign-in page. At the top left is the 3T logo. Below it, a message reads "Welcome to your 3T Account". It says "Access the full range of 3T services: activate your copy of Studio 3T, join our community, and manage your license." A link "Don't have an account, yet? Sign up." is provided. The main area is titled "Sign in with your email and password". It has two input fields: "Email" (placeholder "Type your email address here") and "Password" (placeholder "Type your password here"). Below these are links "Forgot your password?" and "Sign in". At the bottom of the sign-in form is a link "Need an account? Sign up.". Below the sign-in form is a blue button with the Google logo and the text "Sign in with Google".

- 81) Create your account using your email, password, first name and last name, with your phone number being optional. Accept the 3T terms and conditions.

The screenshot shows the 3T sign-up page. At the top left is the 3T logo. Below it, a message reads "Create your new 3T account". It says "Access Studio 3T and all 3T Software Labs' services in just a few clicks." The main area is titled "To sign up, please complete the form below:". It contains five input fields: "Email \*", "Password \*", "First name \*", "Last name \*", and "Telephone number". Each field has a placeholder text. Below the telephone number field is a dropdown menu showing "USA +1". At the bottom of the form is a checkbox labeled "I accept 3T Software Labs' [Privacy Policy](#) and [Terms of Service](#) \*".

- 82) A personalized code should appear on your screen once your account is created. Copy the code by pressing the paper icon on the right of the code, and navigate back to the pop-up on your screen where you accepted the license agreement.

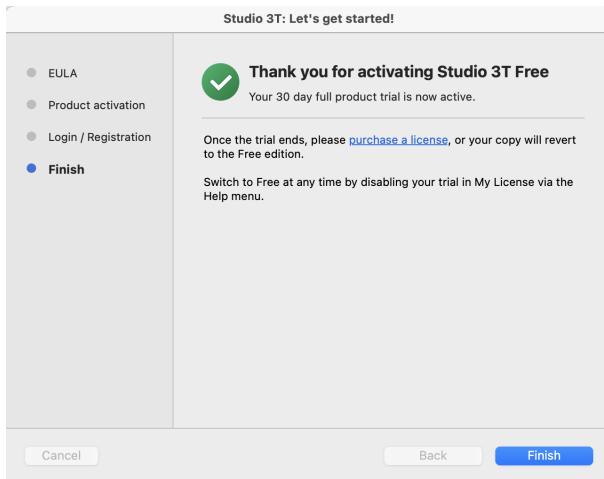
### Just one last step

We couldn't log you in automatically. Please copy the code below.

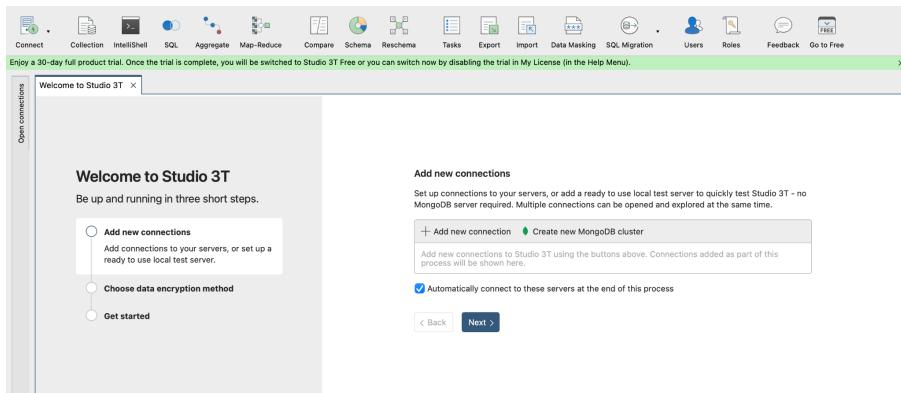
The screenshot shows a code input field with the value "04a07669-bfcf-4f30-b14c-dc". To the right of the input field is a small icon of a sheet of paper with a folded corner, used for copying text.

Then back in Studio 3T, click on the link below the sign in button, and paste the code in the box.

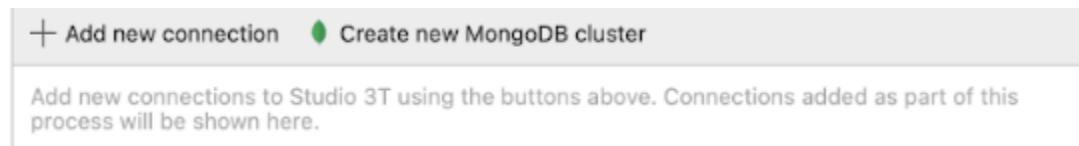
83) Copy the code and press enter in the Login/Registration screen. You should then be moved to the Finish screen, where you can begin your 30 day Free Trial. Press Finish.



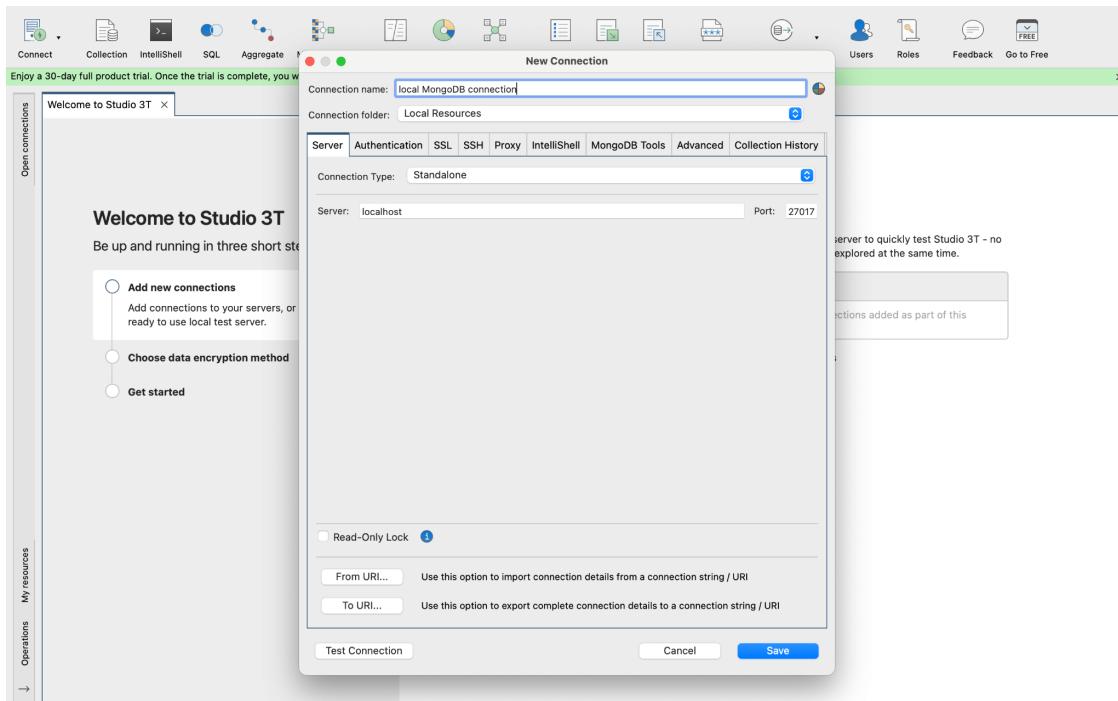
84) Studio 3T will now open and show a welcome page!



85) Click on “Add new connection”



86) This should open a new window. Under “Connection name:” enter “local MongoDB connection”. Do not change any other settings! Press Save.



- 37) After clicking Save, a screen will appear showing the connection, which should be a local connection, meaning one for your computer individually. Press the Next button.

#### Add new connections

Set up connections to your servers, or add a ready to use local test server to quickly test Studio 3T - no MongoDB server required. Multiple connections can be opened and explored at the same time.

+ Add new connection    Create new MongoDB cluster

**local MongoDB connection**  
localhost:27017

Automatically connect to these servers at the end of this process

< Back    **Next >**

A screenshot of a wizard step titled 'Add new connections'. It shows a list of connections. The first item is 'local MongoDB connection' with the details 'localhost:27017'. To the right of the list are edit and delete icons. Below the list is a checked checkbox labeled 'Automatically connect to these servers at the end of this process'. At the bottom are 'Back' and 'Next >' buttons, with 'Next >' being highlighted in dark blue.

88) Now we have to select our data encryption method. We are going to select “Use default password encryption” and then the Next button.

### Choose data encryption method

Configure Studio 3T to match your security needs.

**Use default password encryption (recommended for most users)**

The data stored on your device storage is encrypted, but Studio 3T does not require a master password on application launch.

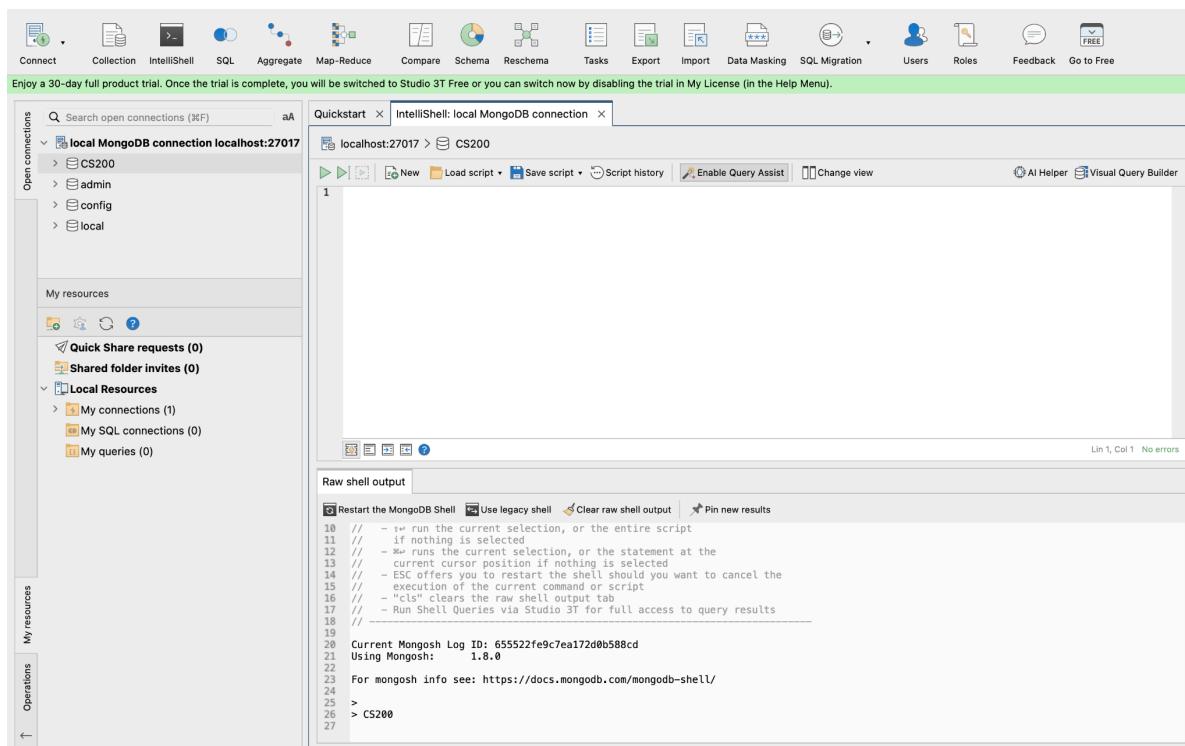
**Use cryptographic key store (for advanced security needs)**

Add extra security to Studio 3T. Your data will be protected by a master password that you will be prompted to enter every time Studio 3T launches. If the password is lost or forgotten, all locally stored data will be lost.

< Back

Next >

89) MongoDB should start. Click “IntelliShell” on the top ribbon, and your local host connection should appear bolded on the left of your screen. In this photo, I have a database already created called “CS200”, but no databases should be shown until you create one in the next section of our tutorial.



40) MongoDB is ready to use! For any troubleshooting with the software installation, visit the official MongoDB website, <https://www.mongodb.com/>

## Code Cases

Case One:

1. Create a database, for example, called **database\_cs200finalproject** with the following code:
  - a. **use database\_cs200finalproject;**
2. Create a collection, known as a table in SQL, of students.
  - a. Students will have 5 different fields, known as columns in SQL:
    - i. Name
    - ii. Age
    - iii. Major
    - iv. Year
    - v. ID
  - b. Run the following code: // this example is ran individually
    - i. **db.Students.insertOne({name: "Lydia", age: 18, major: "CS/DS", year: 01, id: 111})**
    - ii. **db.Students.insertOne({name: "Ashley", age: 21, major: "CS/DS", year: 03, id: 222})**
    - iii. **db.Students.insertOne({name: "Victoria", age: 19, major: "CS", year: 02, id: 333})**
    - iv. **db.Students.insertOne({name: "Sadie", age: 18, major: "DS", year: 01, id: 444})**
    - v. **db.Students.insertOne({name: "Sasha", age: 22, major: "ENV", year: 04, id: 555})**
    - vi. **db.Students.find()** // this will show our results

Code Input:

```
show dbs

use database_cs200finalproject

db.Students.insertOne({name: "Lydia", age: 18, major: "CS/DS", year: 01, id: 111})
db.Students.insertOne({name: "Ashley", age: 21, major: "CS/DS", year: 03, id: 222})
db.Students.insertOne({name: "Victoria", age: 19, major: "CS", year: 02, id: 333})
db.Students.insertOne({name: "Sadie", age: 18, major: "DS", year: 01, id: 444})
db.Students.insertOne({name: "Sasha", age: 22, major: "ENV", year: 04, id: 555})

db.Students.find({})
```

Result Output:

```
{
  acknowledged: true,
  insertedId: ObjectId("655bb9bd7d1797bea26588bc")
}
{
  acknowledged: true,
  insertedId: ObjectId("655bb9c07d1797bea26588bd")
}
{
  acknowledged: true,
  insertedId: ObjectId("655bb9c17d1797bea26588be")
}
{
  acknowledged: true,
  insertedId: ObjectId("655bb9c37d1797bea26588bf")
}
{
  acknowledged: true,
  insertedId: ObjectId("655bb9c57d1797bea26588c0")
}
```

The find query will be run with Query Assist.

```
{
  "_id": ObjectId("6557c2556203547c9eb69963"),
  "name": "Lydia",
  "age": NumberInt(18),
  "major": "CS/DS",
  "year": NumberInt(1),
  "id": NumberInt(111)
}
{
  "_id": ObjectId("6557c25c6203547c9eb69964"),
  "name": "Ashley",
  "age": NumberInt(21),
  "major": "CS/DS",
  "year": NumberInt(3),
  "id": NumberInt(222)
}
{
  "_id": ObjectId("6557c25f6203547c9eb69965"),
  "name": "Victoria",
  "age": NumberInt(19),
  "major": "CS",
  "year": NumberInt(2),
  "id": NumberInt(333)
}
{
  "_id": ObjectId("6557c2626203547c9eb69966"),
  "name": "Sadie",
  "age": NumberInt(18),
  "major": "DS",
  "year": NumberInt(1),
  "id": NumberInt(444)
}
{
  "_id": ObjectId("6557c2646203547c9eb69967"),
  "name": "Sasha",
  "age": NumberInt(22),
  "major": "ENV",
  "year": NumberInt(4),
  "id": NumberInt(555)
}
```

### 3. Querying

- For example, run the code `db.Students.find({year: 01})` to find which students are first year students.

Code Input:

```
db.Students.find({year: 01})
```

Result Output:

```
{
  "_id": ObjectId("6557c2556203547c9eb69963"),
  "name": "Lydia",
  "age": NumberInt(18),
  "major": "CS/DS",
  "year": NumberInt(1),
  "id": NumberInt(111)
}
{
  "_id": ObjectId("6557c2626203547c9eb69966"),
  "name": "Sadie",
  "age": NumberInt(18),
  "major": "DS",
  "year": NumberInt(1),
  "id": NumberInt(444)
}
```

### 4. Create a collection of courses.

- Courses will have 5 different fields as well.
  - ID
  - Name
  - Credit
  - Prerequisites

- v. Category
- b. Run the following code.
- i. db.Courses.insertOne({id: "EN100", name: "Intro to Expository Writing", credit: 3, preq: "None", category: "Pre-Major"})
  - ii. db.Courses.insertOne({id: "EN102", name: "Expository Writing", credit: 3, preq: "EN100", category: "Pre-Major"})
  - iii. db.Courses.insertOne({id: "EN310", name: "Types of Literature", credit: 3, preq: "EN102", category: "Major"})
  - iv. db.Courses.insertOne({id: "BU200", name: "Intro to Business", credit: 3, preq: "None", category: "Pre-Major"})
  - v. db.Courses.insertOne({id: "CS100", name: "Computers and Applications Software", credit: 3, preq: "None", category: "Pre-Major"})
  - vi. db.Courses.find()

Code Input:

```
db.Courses.insertOne({id: "EN100", name: "Intro to Expository Writing", credit: 3, preq: "None", category: "Pre-Major"})
db.Courses.insertOne({id: "EN102", name: "Expository Writing", credit: 3, preq: "EN100", category: "Pre-Major"})
db.Courses.insertOne({id: "EN310", name: "Types of Literature", credit: 3, preq: "EN102", category: "Major"})
db.Courses.insertOne({id: "BU200", name: "Intro to Business", credit: 3, preq: "None", category: "Pre-Major"})
db.Courses.insertOne({id: "CS100", name: "Computers and Applications Software", credit: 3, preq: "None", category: "Pre-Major"})
db.Courses.find({})
```

Result Output:

{ acknowledged: true, insertedId: ObjectId("655bbd497d1797bea26588c1") } { acknowledged: true, insertedId: ObjectId("655bbd4a7d1797bea26588c2") } { acknowledged: true, insertedId: ObjectId("655bbd4c7d1797bea26588c3") } { acknowledged: true, insertedId: ObjectId("655bbd4e7d1797bea26588c4") } { acknowledged: true, insertedId: ObjectId("655bbd507d1797bea26588c5") } The find query will be run with Query Assist.	{ "_id": ObjectId("6557c60e6203547c9eb69968"), "id": "EN100", "name": "Intro to Expository Writing", "credit": NumberInt(3), "preq": "None", "category": "Pre-Major" } { "_id": ObjectId("6557c6106203547c9eb69969"), "id": "EN102", "name": "Expository Writing", "credit": NumberInt(3), "preq": "EN100", "category": "Pre-Major" } { "_id": ObjectId("6557c6126203547c9eb6996a"), "id": "EN310", "name": "Types of Literature", "credit": NumberInt(3), "preq": "EN102", "category": "Major" } { "_id": ObjectId("6557c6146203547c9eb6996b"), "id": "BU200", "name": "Intro to Business", "credit": NumberInt(3), "preq": "None", "category": "Pre-Major" } { "_id": ObjectId("6557c6176203547c9eb6996c"), "id": "CS100", "name": "Computers and Applications Software", "credit": NumberInt(3), "preq": "None", "category": "Pre-Major" }
--	--

## 5. Querying

- a. For example, run the code db.Courses.find({preq: "None"}) to find which courses have no prerequisites.

Code Input:

```
db.Courses.find({preq: "None"})
```

Result Output:

```
[{"_id": ObjectId("6557c60e6203547c9eb69968"),  
 "id": "EN100",  
 "name": "Intro to Expository Writing",  
 "credit": NumberInt(3),  
 "preq": "None",  
 "category": "Pre-Major"},  
  
 {"_id": ObjectId("6557c6146203547c9eb6996b"),  
 "id": "BU200",  
 "name": "Intro to Business",  
 "credit": NumberInt(3),  
 "preq": "None",  
 "category": "Pre-Major"},  
  
 {"_id": ObjectId("6557c6176203547c9eb6996c"),  
 "id": "CS100",  
 "name": "Computers and Applications Software",  
 "credit": NumberInt(3),  
 "preq": "None",  
 "category": "Pre-Major"}]
```

6. Create a table of advisors.

- a. Again, advisors will have 5 different fields.
  - i. ID
  - ii. First Name
  - iii. Last Name
  - iv. Phone Number
  - v. Department
- b. Run the following code to create the collection
  - i. db.Advisors.insertOne({id: 10, firstname: "Justin", lastname: "Wyble", phone: "555-1111", department:"EN"})
  - ii. db.Advisors.insertOne({id: 11, firstname: "Rylan", lastname: "Chong", phone: "555-2222", department:"DSAV"})
  - iii. db.Advisors.insertOne({id: 12, firstname: "Robert", lastname: "Maruyama", phone: "555-3333", department:"CIS"})
  - iv. db.Advisors.insertOne({id: 13, firstname: "Laura", lastname: "Tipton", phone: "555-4444", department:"CIS" })
  - v. db.Advisors.insertOne({id: 14, firstname: "Mark", lastname: "Speck", phone: "555-5555", department:"DSAV" })
  - vi. db.Advisors.find({})

Code Input:

```
db.Advisors.insertOne({id: 10, firstname: "Justin", lastname: "Wyble" , phone: "555-1111", department:"EN"})  
db.Advisors.insertOne({id: 11, firstname: "Rylan", lastname: "Chong" , phone: "555-2222", department:"DSAV"})  
db.Advisors.insertOne({id: 12, firstname: "Robert", lastname: "Maruyama" , phone: "555-3333", department:"CIS"})  
db.Advisors.insertOne({id: 13, firstname: "Laura", lastname: "Tipton" , phone: "555-4444", department:"CIS" })  
db.Advisors.insertOne({id: 14, firstname: "Mark", lastname: "Speck" , phone: "555-5555", department:"DSAV" })  
  
db.Advisors.find({})
```

Results Output:

```
{
  acknowledged: true,
  insertedId: ObjectId("655bbf957d1797bea26588c6")
}

{
  acknowledged: true,
  insertedId: ObjectId("655bbf967d1797bea26588c7")
}

{
  acknowledged: true,
  insertedId: ObjectId("655bbf987d1797bea26588c8")
}

{
  acknowledged: true,
  insertedId: ObjectId("655bbf997d1797bea26588c9")
}

{
  acknowledged: true,
  insertedId: ObjectId("655bbf9a7d1797bea26588ca")
}
The find query will be run with Query Assist.
```

<code>"_id" : ObjectId("6557c8936203547c9eb6996d"),  "id" : NumberInt(10),  "firstname" : "Justin",  "lastname" : "Wyble",  "phone" : "555-1111",  "department" : "EN"</code>
<code>"_id" : ObjectId("6557c8956203547c9eb6996e"),  "id" : NumberInt(11),  "firstname" : "Rylan",  "lastname" : "Chong",  "phone" : "555-2222",  "department" : "DSAV"</code>
<code>"_id" : ObjectId("6557c8966203547c9eb6996f"),  "id" : NumberInt(12),  "firstname" : "Robert",  "lastname" : "Maruyama",  "phone" : "555-3333",  "department" : "CIS"</code>
<code>"_id" : ObjectId("6557c8986203547c9eb69970"),  "id" : NumberInt(13),  "firstname" : "Laura",  "lastname" : "Tipton",  "phone" : "555-4444",  "department" : "CIS"</code>
<code>"_id" : ObjectId("6557c89a6203547c9eb69971"),  "id" : NumberInt(14),  "firstname" : "Mark",  "lastname" : "Speck",  "phone" : "555-5555",  "department" : "DSAV"</code>

## 7. Querying

- a. For example, run the code `db.Advisors.find({department: "CIS"})` to find which advisors are in the CIS department.

Code Input:

```
db.Advisors.find({department: "CIS"})
```

Result Output:

<code>{"_id" : ObjectId("6557c8966203547c9eb6996f"),  "id" : NumberInt(12),  "firstname" : "Robert",  "lastname" : "Maruyama",  "phone" : "555-3333",  "department" : "CIS"</code>
<code>{"_id" : ObjectId("6557c8986203547c9eb69970"),  "id" : NumberInt(13),  "firstname" : "Laura",  "lastname" : "Tipton",  "phone" : "555-4444",  "department" : "CIS"</code>

Case Two:

1. First, we are going to create our database called **dbl** with the following code:
  - a. **use dbl;**
2. Let's create our collection (in SQL, you know this as table) on different ramen spots on Oahu.
  - a. In our collection, we want to show the following 5 different fields (in SQL, you know this as columns):
    - i. Name
    - ii. City
    - iii. Street
    - iv. Rating
    - v. Number
  - b. Run this code to create the collection
    - i. **db.oahuramen.insertMany([**  
          { name: "Onoya Ramen", city: "Honolulu", street: "Kapahulu", rating: 5, number: "(808)425-4415" },  
          { name: "Wagaya", city: "Honolulu", street: "King St", rating: 4, number: "(808)949-0670" },  
          { name: "Nagoya Ramen", city: "Mililani", street: "Kipapa Dr", rating: 4, number: "(808)625-9999" },  
          { name: "Rai Rai Ramen", city: "Kailua", street: "Oneawa St", rating: 5, number: "(808)230-8208" },  
          { name: "Ayame Curry Ramen House", city: "Kaneohe", street: "Kaneohe Bay Dr", rating: 5, number: "(808)236-3883" }  
**])**
    - ii. Results

```
use dbl;  
  
db.oahuramen.insertMany([  
  { name: "Onoya Ramen", city: "Honolulu", street: "Kapahulu", rating: 5, number: "(808)425-4415" },  
  { name: "Wagaya", city: "Honolulu", street: "King St", rating: 4, number: "(808)949-0670" },  
  { name: "Nagoya Ramen", city: "Mililani", street: "Kipapa Dr", rating: 4, number: "(808)625-9999" },  
  { name: "Rai Rai Ramen", city: "Kailua", street: "Oneawa St", rating: 5, number: "(808)230-8208" },  
  { name: "Ayame Curry Ramen House", city: "Kaneohe", street: "Kaneohe Bay Dr", rating: 5, number: "(808)236-3883" }  
])  
  
db.oahuramen.find()
```

3. Great job! We've created our oahuramen collection with valuable information. Let's run the code **db.oahuramen.find()** (in SQL, this is like `SELECT * FROM <table name>`) to see what our collection looks like.

Raw shell output		Find Query (line 11)											
		Documents 1 to 5											
		←	←	→	→	50	▼	Documents 1 to 5	🔒	➕	✖	✖	✖
<b>oahuramen &gt; name</b>													
_id	name	city	street	rating	number								
<code>id 6557c648f019fe</code>	<code>"Onoya Ramen</code>	<code>"Honolulu</code>	<code>"Kapahulu</code>	<code>5</code>	<code>(808)425-4415</code>								
<code>id 6557c648f019fe</code>	<code>"Wagaya</code>	<code>"Honolulu</code>	<code>"King St</code>	<code>4</code>	<code>(808)949-0670</code>								
<code>id 6557c648f019fe</code>	<code>"Nagoya Ramen</code>	<code>"Mililani</code>	<code>"Kipapa Dr</code>	<code>4</code>	<code>(808)625-9999</code>								
<code>id 6557c648f019fe</code>	<code>"Rai Rai Ramen</code>	<code>"Kailua</code>	<code>"Oneawa St</code>	<code>5</code>	<code>(808)230-8208</code>								
<code>id 6557c648f019fe</code>	<code>"Ayame Curry Rai</code>	<code>"Kaneohe</code>	<code>"Kaneohe Bay Dr</code>	<code>5</code>	<code>(808)236-3883</code>								

## 4. Querying

- a. Let's start with some simple queries with our oahuramen collection.

Querying is what we call finding specific pieces of information with our data.

- i. I want to find out which ramen spots are located in Honolulu. To do this, we're going to run the following query code

- ```
1. db.oahuramen.find(  
    { city: "Honolulu"}  
  
    db.oahuramen.find(  
        { city: "Honolulu"}  
    )
```

## 2 Results:

| Raw shell output |             | Find Query (line 13) ↗ ×           |          |                  |               |   |   |   |   |   |   |
|------------------|-------------|------------------------------------|----------|------------------|---------------|---|---|---|---|---|---|
|                  |             | Documents 1 to 2   🔒 🔍 🔎 🔎 🔎 🔎 🔎 🔎 |          |                  |               |   |   |   |   |   |   |
| ← ← → →          |             | 50                                 | ▼        | Documents 1 to 2 |               | 🔒 | 🔍 | 🔍 | 🔍 | 🔍 | 🔍 |
| <b>oahuramen</b> |             |                                    |          |                  |               |   |   |   |   |   |   |
| id               | name        | city                               | street   | rating           | number        |   |   |   |   |   |   |
| 6557c648f019fe   | Onoya Ramen | Honolulu                           | Kapahulu | 5                | (808)425-4415 |   |   |   |   |   |   |
| 6557c648f019fe   | Wagaya      | Honolulu                           | King St  | 4                | (808)949-0670 |   |   |   |   |   |   |

- ii. Now, I realized that I made an error and Wagaya actually has a rating of 3. Let's use the following code to update this document (in SQL, you know this as row)

- ```
1. db.oahuramen.updateOne(  
  { name: "Wagaya"},  
  { $set: {rating: 3}})
```

```
db.oahuramen.updateOne(  
  { name: "Wagaya"},  
  { $set: {rating: 3}}  
)
```

2. Results (to view the results, run `db.oahuramen.find()`)

oahuramen						
<code>_id</code>	<code>name</code>	<code>city</code>	<code>street</code>	<code>rating</code>	<code>number</code>	
<code>6557c648f019fe</code>	Onoya Ramen	Honolulu	Kapahulu	5	(808)425-4415	
<code>6557c648f019fe</code>	Wagaya	Honolulu	King St	3	(808)949-0670	
<code>6557c648f019fe</code>	Nagoya Ramen	Mililani	Kipapa Dr	4	(808)625-9999	
<code>6557c648f019fe</code>	Rai Rai Ramen	Kailua	Oneawa St	5	(808)230-8208	
<code>6557c648f019fe</code>	Ayame Curry Rai	Kaneohe	Kaneohe Bay Dr	5	(808)236-3883	

5. Tired of looking at ramen data? Let's create a collection of sushi places on Oahu.

- a. In our collection, we want to show the following 5 different fields (in SQL, you know this as columns):

- i. Name
- ii. Location
- iii. Top Item
- iv. Capacity
- v. Parking

- b. Run this code to create the collection

```
i. db.oahusushi.insertMany([
    { name: "Sushiya", location: "Honolulu", topitem: "California Roll",
      capacity: 10, parking: true },
    { name: "Tanuki Sushi", location: "Kaneohe", topitem: "Spicy Tuna
      Nigiri", capacity: 50, parking: false},
    { name: "SushiMan Wahiawa", location: "Wahiawa", topitem:
      "Dragon Roll", capacity: 30, parking: true },
    { name: "Genki Sushi", location: "Waipahu", topitem: "Salmon
      Avocado Roll", capacity: 10, parking: false },
    { name: "Chiba-ken", location: "Honolulu", topitem: "Spicy Salmon",
      capacity: 20, parking: true }
])
```

```
db.oahusushi.insertMany([
    { name: "Sushiya", location: "Honolulu", topitem: "California Roll", capacity: 10, parking: true },
    { name: "Tanuki Sushi", location: "Kaneohe", topitem: "Spicy Tuna Nigiri", capacity: 50, parking: false},
    { name: "SushiMan Wahiawa", location: "Wahiawa", topitem: "Dragon Roll", capacity: 30, parking: true },
    { name: "Genki Sushi", location: "Waipahu", topitem: "Salmon Avocado Roll", capacity: 10, parking: false },
    { name: "Chiba-ken", location: "Honolulu", topitem: "Spicy Salmon", capacity: 20, parking: true }
])
db.oahusushi.find()
```

ii. Results:

oahusushi > name						
_id	name	location	topitem	capacity	parking	
65586c9e96a4f	Sushiya	Honolulu	California Roll	10	true	
65586c9e96a4f	Tanuki Sushi	Kaneohe	Spicy Tuna Nigiri	50	false	
65586c9e96a4f	SushiMan Wahia	Wahiawa	Dragon Roll	30	true	
65586c9e96a4f	Genki Sushi	Waipahu	Salmon Avocado	10	false	
65586c9e96a4f	Chiba-ken	Honolulu	Spicy Salmon	20	true	

6. Querying

- a. Awesome! Let's do a couple of queries with our new collection on sushi spots on Oahu.

- i. I want to see all the sushi spots based on capacity, from least to greatest. Run this code to find out:

1. `db.oahusushi.find().sort({capacity: 1})`

2. Results

oahusushi						
_id	name	location	topitem	capacity	parking	
65586c9e96a4f	Sushiya	Honolulu	California Roll	10	true	
65586c9e96a4f	Genki Sushi	Waipahu	Salmon Avocado	10	false	
65586c9e96a4f	Chiba-ken	Honolulu	Spicy Salmon	20	true	
65586c9e96a4f	SushiMan Wahia	Wahiawa	Dragon Roll	30	true	
65586c9e96a4f	Tanuki Sushi	Kaneohe	Spicy Tuna Nigiri	50	false	

- ii. We're trying to find out which restaurants have parking. Run this code to find out:

1. `db.oahusushi.find({parking:{$eq: true}}).pretty()`

`db.oahusushi.find({parking:{$eq: true}}).pretty()`

2. Results

oahusushi > name						
_id	name	location	topitem	capacity	parking	
65586c9e96a4f	Sushiya	Honolulu	California Roll	10	true	
65586c9e96a4f	SushiMan Wahia	Wahiawa	Dragon Roll	30	false	
65586c9e96a4f	Chiba-ken	Honolulu	Spicy Salmon	20	true	

7. We're kind of hungry and want to know about the different AYCE (all you can eat) place on Oahu! Let's create a collection with this information.

- a. In our collection, we want to show the following 5 different fields (in SQL, you know this as columns):

- i. Name
  - ii. Grade (food sanitation)
  - iii. freshProduce
  - iv. foodWarmers
  - v. Timelimit

b. Run this code to create the collection

  - i. db.ohauayce.insertMany([  
    { name: "McCully Buffet", grade: "A", freshProduce: true ,  
      foodWarmers: true, timelimit: "2 hours"},  
    { name: "Pot Pot Shabu Buffet", grade: "B", freshProduce: false,  
      foodWarmers: true, timelimit: "3 hours"},  
    { name: "Kats Sushi", grade: "A", freshProduce: true, foodWarmers:  
      false, timelimit: "2 hours"},  
    { name: "Shabuya", grade: "B", freshProduce: false, foodWarmers:  
      true, timelimit: "2.5 hours"},  
    { name: "Gyu-Kaku Japanese BBQ", grade: "A", freshProduce: true,  
      foodWarmers: false, timelimit: "3 hours"}  
])

```
db.oahuayce.insertMany([
  { name: "McCully Buffet", grade: "A", freshProduce: true , foodWarmers: true, timelimit: "2 hours"}, 
  { name: "Pot Pot Shabu Buffet", grade: "B", freshProduce: false, foodWarmers: true, timelimit: "3 hours"}, 
  { name: "Kats Sushi", grade: "A", freshProduce: true, foodWarmers: false, timelimit: "2 hours"}, 
  { name: "Shabuya", grade: "B", freshProduce: false, foodWarmers: true, timelimit: "2.5 hours"}, 
  { name: "Gyu-Kaku Japanese BBQ", grade: "A", freshProduce: true, foodWarmers: false, timelimit: "3 hours"}])
db.oahuayce.find()
```

ii. Results

Raw shell output | Find Query (line 32) ⌘ X

|← ← → →| 50 ⌄ | Documents 1 to 5 | 🔒 🗑️ 🗑️ 🗑️ 🗑️ 🗑️ 🗑️

ohauayce > name						
_id	name	grade	freshProduce	foodWarmers	timelimit	
6558708e4bc3d	McCully Buffet	A	true	true	2 hours	
6558708e4bc3d	Pot Pot Shabu B	B	false	true	3 hours	
6558708e4bc3d	Kats Sushi	A	true	false	2 hours	
6558708e4bc3d	Shabuya	B	false	true	2.5 hours	
6558708e4bc3d	Gyu-Kaku Japan	A	true	false	3 hours	

## 8. Querying

- a. Let's run the following query codes to view some interesting information!
    - i. I want to see the first document (row) of the collection. Run the following code to find out

1. db.oahuayce.findOne() `db.oahuayce.findOne()`

## 2. Results

Result > name					
_id	name	grade	freshProduce	foodWarmers	timelimit
6558708e4bc3c	McCully Buffet	A	true	true	2 hours

- ii. Now, let's sort our collection by Grade, starting with A so we can see it in order

1. db.ohauayce.find().sort({grade :1}).pretty()  
`db.ohauayce.find().sort({grade :1}).pretty()`

2. Results

ohauayce > name					
_id	name	grade	freshProduce	foodWarmers	timelimit
6558708e4bc3c	McCully Buffet	A	true	true	2 hours
6558708e4bc3c	Kats Sushi	A	true	false	2 hours
6558708e4bc3c	Gyu-Kaku Japan	A	true	false	3 hours
6558708e4bc3c	Pot Pot Shabu B	B	false	true	3 hours
6558708e4bc3c	Shabuya	B	false	true	2.5 hours

## Useful Links

- What is MongoDB?:  
[https://www.mongodb.com/docs/manual/?\\_ga=2.106770563.1294327503.1699472694-109957512.1699472694](https://www.mongodb.com/docs/manual/?_ga=2.106770563.1294327503.1699472694-109957512.1699472694)
- Download/Install MongoDB and connect to Studio 3t:  
<https://studio3t.com/knowledge-base/articles/how-to-install-a-local-mongodb-on-macos/>
- Explanation/Walkthrough Youtube Video:  
<https://www.youtube.com/watch?v=MPGmnpoUEDI&list=PLRTey0Iqj9jh3WTvoupGK0aizk-Rmbz7u&index=3>
- SQL to MongoDB Concept and Terminology Conversion:  
<https://www.mongodb.com/docs/manual/reference/sql-comparison/>
- How to Query using MongoDB:  
<https://www.knowledgehut.com/blog/web-development/mongodb-query-document-using-find-with-example>

## Summary

The MongoDB language differs largely in comparison to MySQL, as it is a non-relational database. MongoDB uses collections of documents within its databases rather than tables with rows and columns like MySQL. Writing out code is also different among the two, MongoDB is a little less critical. For example, ";" aren't necessary. Ultimately, MongoDB is more flexible, whereas MySQL is a more structured, tabular database.

## References

Quach, S., & Hong, C. (2023, May 17). *MongoDB vs SQL - an in-depth comparison*. Knowi.  
<https://www.knowi.com/blog/mongodb-vs-sql/#:~:text=SQL%20databases%20are%20used%20to,join%20like%20SQL%20databases%20support>.

MongoDB. (2023). *SQL to MongoDB Mapping Chart*. MongoDB.

<https://www.mongodb.com/docs/manual/reference/sql-comparison/>

## GitHub Submissions