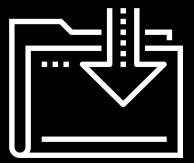
Mastering MongoDB

Data Boot Camp

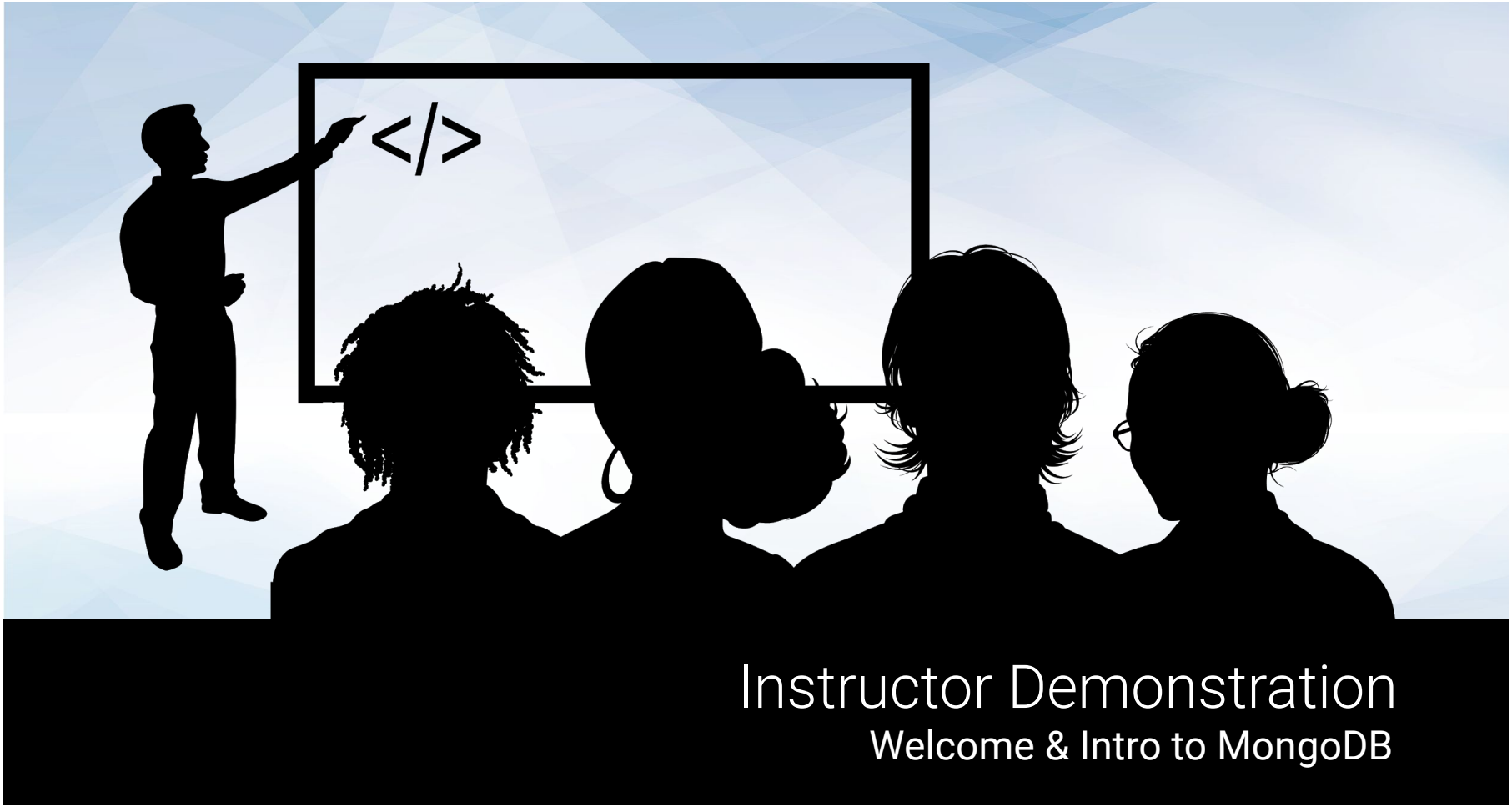
Lesson 12.1



Class Objectives

By the end of today's class you will be able to:

- Create and connect to local MongoDB databases.
- Create, read, update, and delete MongoDB documents using the Mongo Shell.
- Create simple Python applications that connects to and modify MongoDB databases using PyMongo library.



Instructor Demonstration

Welcome & Intro to MongoDB

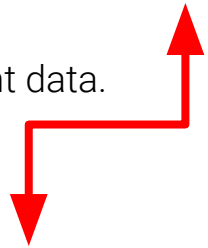
What's Welcome & Intro to MongoDB ?

- MongoDB is a very popular noSQL database.
- It uses a document-oriented model as opposed to a table-based relational model (SQL).
- MongoDB stores data in BSON format (effectively, compressed JSONs).
- MongoDB has tons of drivers and packages for connecting to Node, C++, Java, etc.

Welcome & Intro to MongoDB

ID	Title	Author	Published
1	The History of Blah	Blah Matic	2010
2	The Chronicles of Blahrnia	Sir Blahston	2011
3	Love in the Time of Blah	Gabriel Garcia Blah	2013

SQL relies on **joins** to combine relevant data.



Author	Email	Phone Number
Blah Matic	blahston@gmail.com	911-546-5454
Sir Blahston	blahby@gmail.com	911-544-5112
Gabriel Garcia Blah	blahby231@gmail.com	125-215-5645

Welcome & Intro to MongoDB

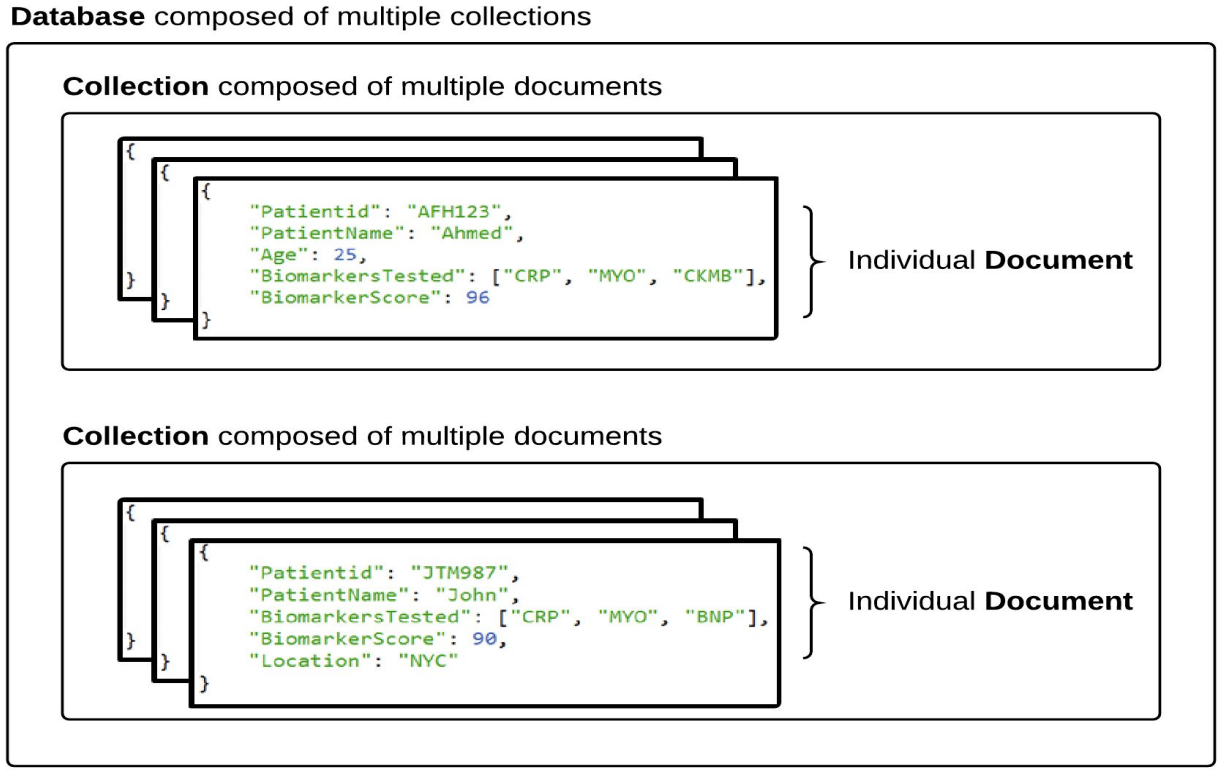
- noSQL databases, on the other hand, are effectively JSON's.
- They excel at heterogeneous data formats and are easy to implement.

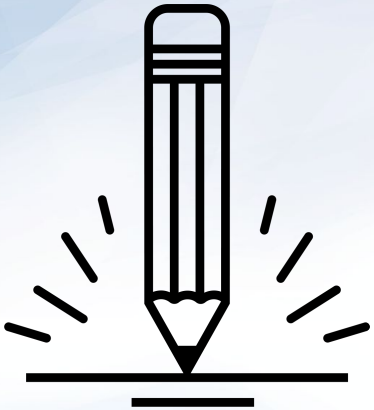
Welcome & Intro to MongoDB

- Terms are slightly difference in the noSQL context. **Take note!**

SQL (RDBMS)	MongoDB
Database	**Database**
Table	**Collection**
Row	**Document**
Column	**Field**
Table Join	**Embedded Documents**
Primary Key	**Primary Key (Default key _id provided by MongoDB itself**

Welcome & Intro to MongoDB





Activity: Quick Mongo Research

In this activity, you and your partner will answer the following questions:

Suggested Time:
5 Minutes

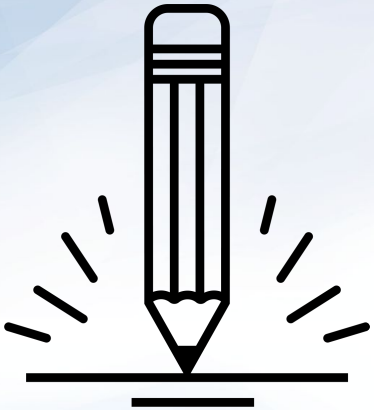
Answer the following questions:

Activity: Quick Mongo Research

- What are the advantages of using a noSQL database like MongoDB according to the MongoDB website?
- What are the advantages of using a noSQL database like MongoDB according to the web (places like Quora)?
- What are the disadvantages of using a noSQL database like MongoDB according to the web (places like Quora)?

Everyone Do: Quick Mongo Research Review

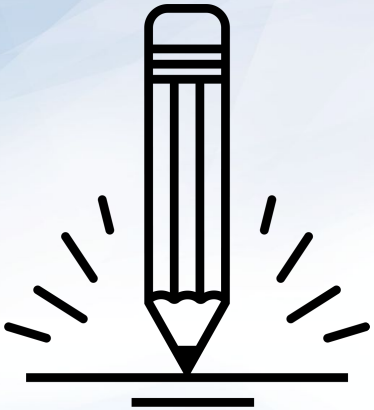
- ***What are the advantages of using a noSQL database like MongoDB according to the MongoDB website?***
  - "Relational databases require that schemas be defined before you can add data. For example, you might want to store data about your customers such as phone numbers, first and last name, address, city and state – a SQL database needs to know what you are storing in advance."
  - "Object-oriented programming that is easy to use and flexible."
- ***What are the advantages of using a noSQL database like MongoDB according to the web (places like Quora)?***
  - Deep query-ability. MongoDB supports dynamic queries on documents using a document-based query language that's nearly as powerful as SQL.
  - No schema migrations. Since MongoDB is schema-free, your code defines your schema.
- ***What are the disadvantages of using a noSQL database like MongoDB according to the web (places like Quora)?***
  - Sometimes, using joins and having strict schemas is actually preferable to MongoDB.
  - "If your database has a lot of relations and normalization, it might make little sense to use something like MongoDB. It's all about finding the right tool for the job."



Everyone Do: Quick Mongo Research Review

In this review activity, everyone will review the answers for the previous activity:

Suggested Time:
5 Minutes



Activity: Installing MongoDB

In this activity, you will download and install MongoDB into your machine.

Suggested Time:
20 Minutes

Instructions:

Activity: Installing MongoDB

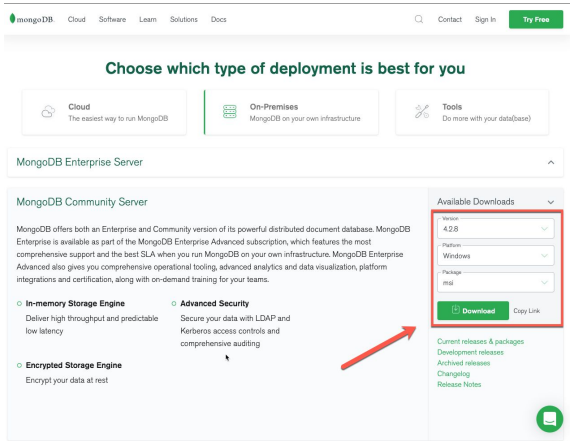
Windows

- Navigate to MongoDB Download Center and follow the instructions below:

1. Click on ‘MongoDB Community Server’

2. On the dropdown menu select:

Version: **4.2.8**
Platform: **Windows**
Package: **msi**



- Open your terminal and follow the instructions below:

1. In case you do not have **Homebrew** run the following command:

```
# Installs Homebrew
/usr/bin/ruby -e "$(curl -fsSL
https://raw.githubusercontent.com/Homebrew/install/master
/install)"
```

2. If you have Homebrew run the following command to find the MongoDB tab.

```
~
❯ brew tap mongodb/brew
```

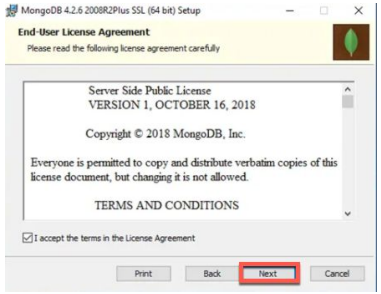
Instructions:

Activity: Installing MongoDB

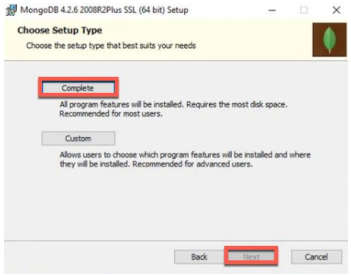
Windows

3. Run the MongoDB installer and follow the installation wizard.

1.



2.



3. **Click *Complete* and *Next***

3. Initialize installation by running the following command :

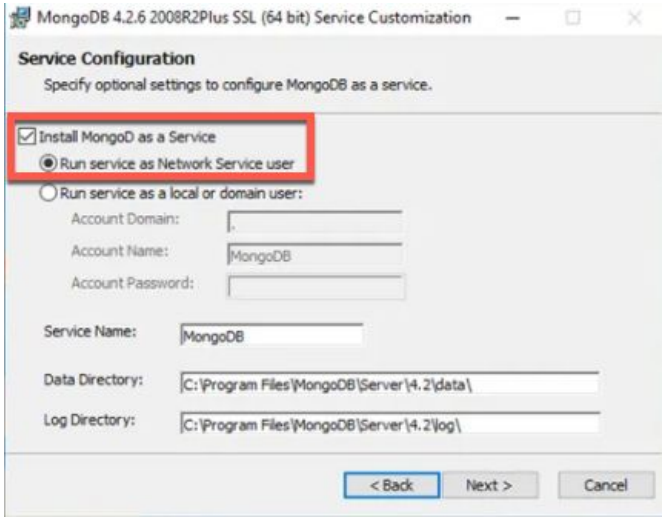
Note: Homebrew will automatically do install the latest version of MongoDB

Instructions:

Activity: Installing MongoDB

Windows

4. On Service Configuration:
   A. Check '***Install MongoDB as a Service***'
   B. Click '***Run service as Network Service User***'



4. Create a folder designated for MongoDB usage:
If you are running MacOS Catalina run:

```
❯ sudo mkdir -p /System/Volumes/Data/data/db
```

Then run the following to grant access:

```
❯ sudo chown -R `id -un` /System/Volumes/Data/data/db
```

If you are running a version prior to MacOS Catalina:

```
❯ sudo mkdir -p /data/db
```

Then run the following to grant access:

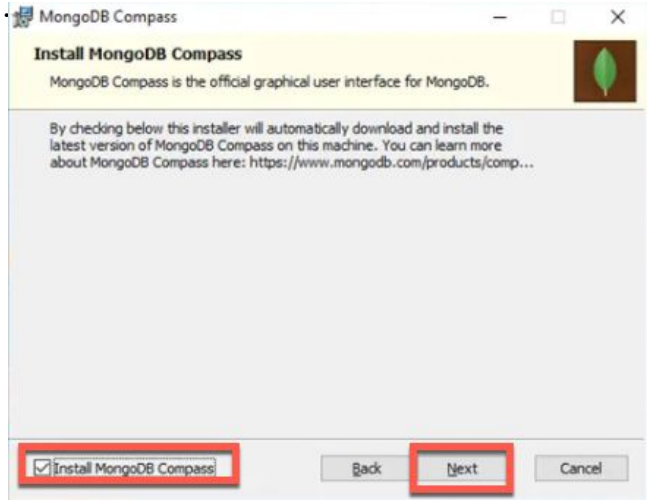
```
❯ sudo chown -R `id -un` /data/db
```

Instructions:

Activity: Installing MongoDB

Windows

5. Make sure ***"Install MongoDB Compass"*** is checked and click ***"next"***.



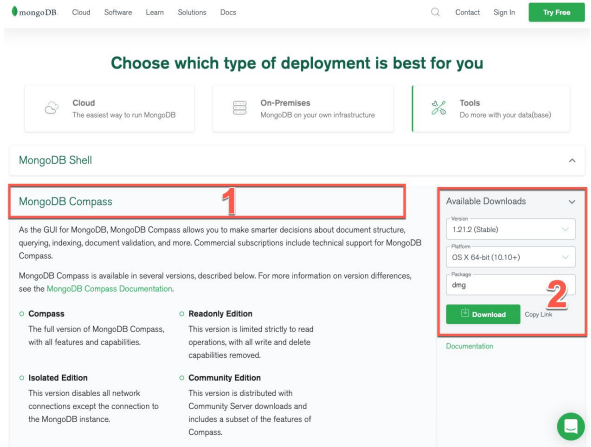
Note: The instructions from this point onwards is referent to the installation of the *MongoDB Compass* ***as a continuation of the*** *MongoDB* ***installation process.*** ***MongoDB Compass*** ***is the GUI for*** ***MongoDB****.*

- Navigate to the download page *(link sent to your slack)* and follow the instructions below:

1. Click on 'MongoDB Compass'

2. On the dropdown menu select:

Version: **1.21.2(Stable)**
Platform: **OS X 64-bit(10.10+)**

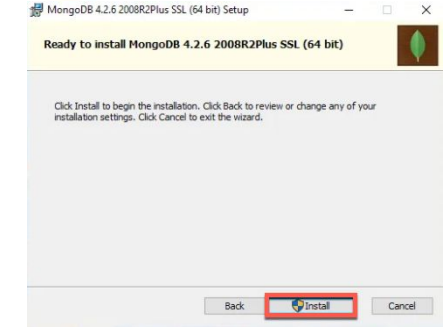


Note: The *MongoDB Compass installation process in a mac is* ***a separate installation process utilizing the GUI and NOT the CLI as*** ***MongoDB*** *was installed up to this point.*

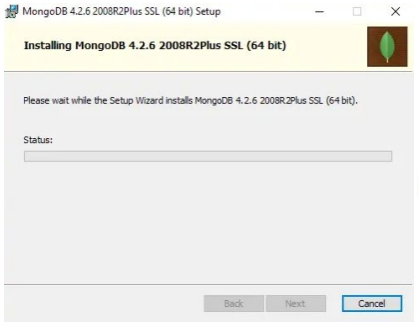
Instructions:

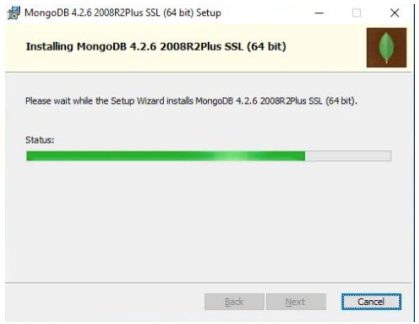
Activity: Installing MongoDB

Windows



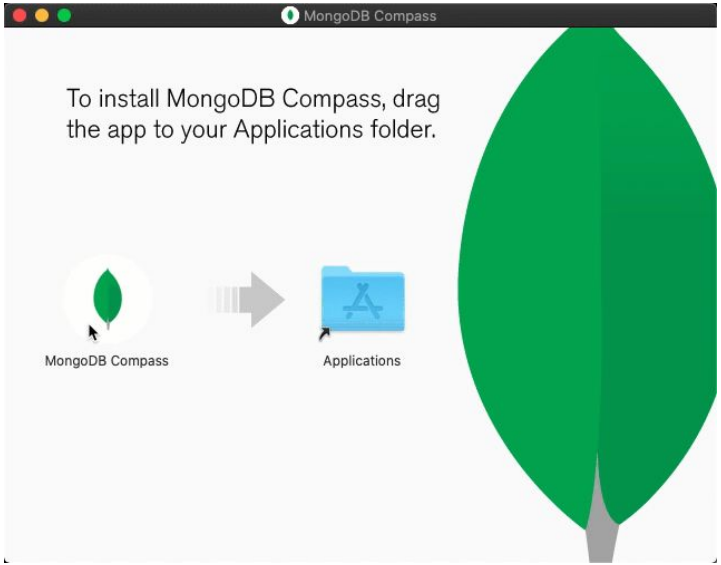
6. Click ***"Install"***





3. Once you have downloaded Compass, double-click on the `.dmg` file to open the disk image within the macOS Finder.

4. Drag the ***MongoDB Compass*** application to your **Applications** folder.

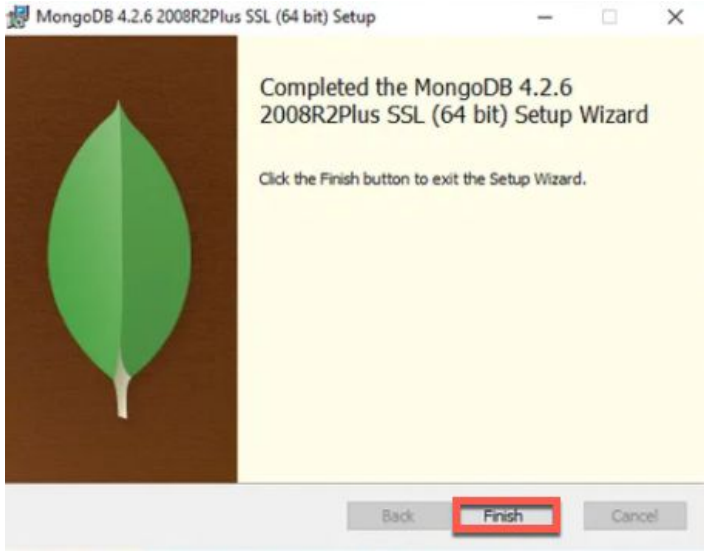


Instructions:

Activity: Installing MongoDB

Windows

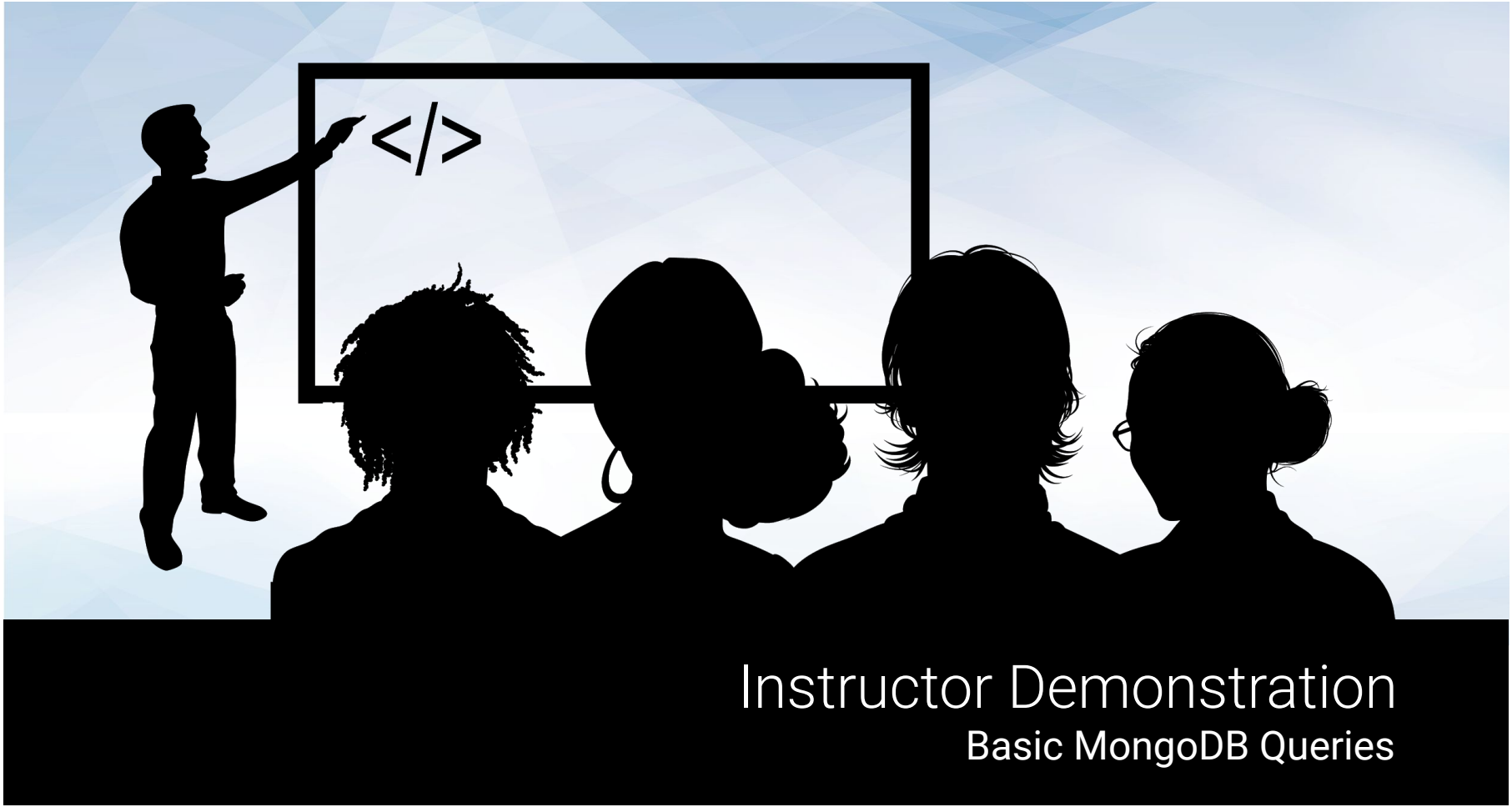
7. Click ***"Finish"*** and restart your computer.



5. In Finder, eject the disk image.
6. From the **Applications** folder, double-click on the Compass icon to start the application.
7. Allow macOS to trust Compass. If you receive a security error when starting Compass indicating that the developer could not be identified or verified, perform the following actions to allow Compass to run:

   A. Open *System Preferences*.
   B. Select the *Security and Privacy* pane.
   C. Under the *General* tab, click the button to the right of the message about Compass, labelled either **Open Anyway** or **Allow Anyway** depending on your version of macOS.
   D. If necessary, re-open Compass.

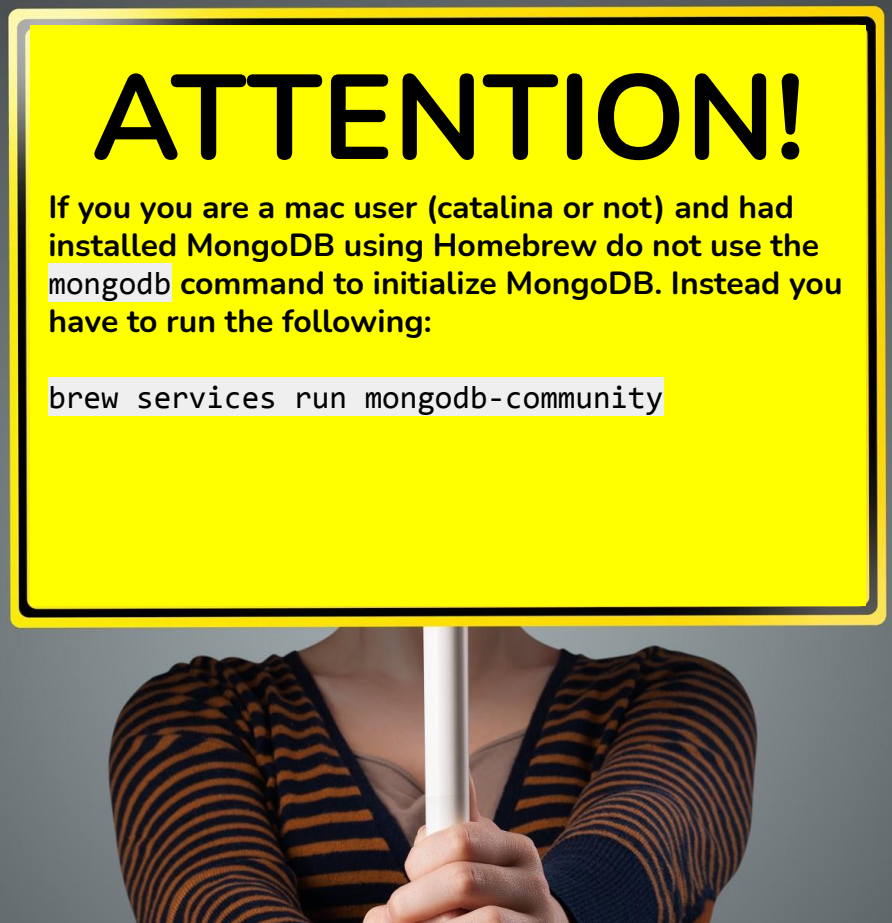
8. When you open MongoDB Compass for the first time, you may receive a notice stating that it is an application downloaded from the internet, requiring you to confirm you want to open it. Click **Open** to continue and launch Compass.



Instructor Demonstration

Basic MongoDB Queries

Basic MongoDB Queries



Basic MongoDB Queries

```
>
```

>> The `mongo` command initiates the mongo shell.

```
>
```

```
>
```

>> The `use <database_name>` command not only create a database but switch to it in case the database already exist.

```
>
```

```
>
```

>> The `db` command shows what database is current in use.

```
> 
```

Basic MongoDB Queries

```
>
```

>> The `show dbs` command lists all current databases.

```
>
```

```
>
```

>> The `show collections` command lists all collections within the current database.

```
> show collections
>
```

Basic MongoDB Queries

```
>
```

>> To input data into a document (row) we use the `db.collectionName.insert({key:value})`.

```
> db.destinations.insert({"continent": "Europe", "country": "Italy",
 "major_cities": ["Milan", "Rome", "Florence", "Turin", "Rome"]})
WriteResult({ "nInserted" : 1 })
> 
```

```
>
```

>> This function return values within specific collection in more readable way.

```
> db.destinations.find().pretty()
{
        "_id" : ObjectId("5f199b6eb3f77d515acb553c"),
        "continent" : "Europe",
        "country" : "Italy",
        "major_cities" : [
                "Milan",
                "Rome",
                "Florence",
                "Turin",
                "Rome"
        ]
}
> 
```

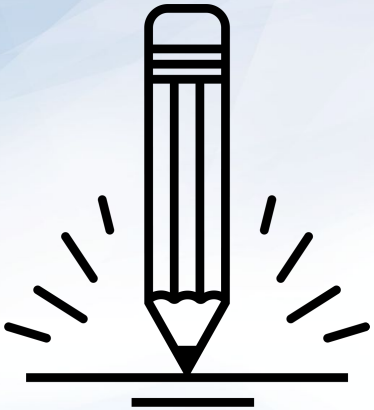
Basic MongoDB Queries

```
>
```

>> The syntax used to find specific documents within a collection is `db.collectionName.find({key:value})`.

```
>
```

```
>
```



Activity: Mongo Class

In this activity, you will familiarize with the basic query operations in MongoDB. Specifically, inserting and finding documents.

Suggested Time:
15 Minutes

Instructions:

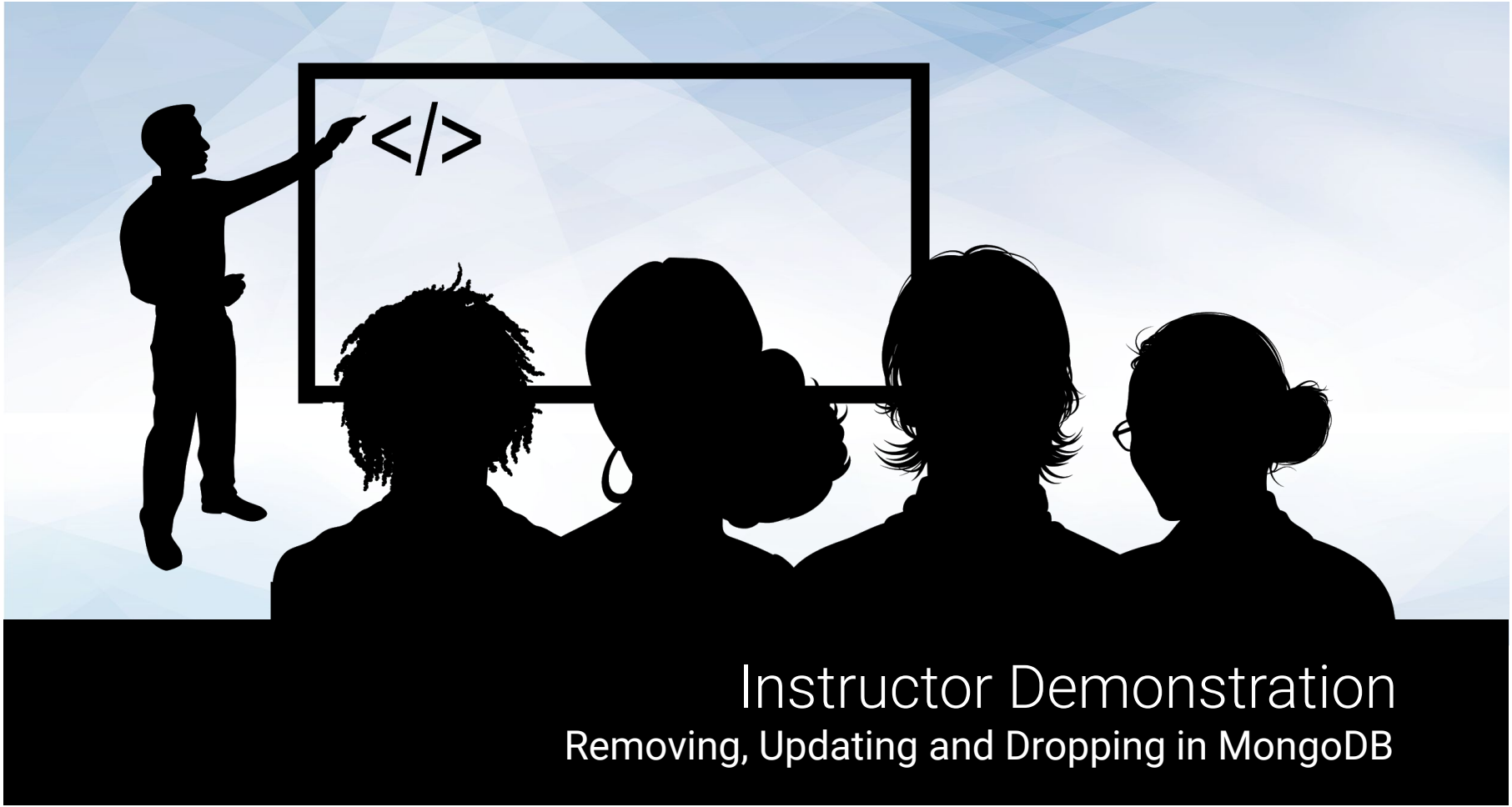
Activity: Mongo Class

- Use the command line to create a `ClassDB` database.
- Insert entries into this database for yourself and the people around you within a collection called Students.
- Each document should have a field of name with the person's name, a field of `favorite_python_library` for the person's favorite Python library, a field of `age` for the. person's age, and a field of hobbies which will hold a list of that person's hobbies.

- Use the `find()` commands to get a list of everyone of a specific age before using name to collect the entry for a single person.

- **Bonus**:
  - Check out the MongoDB Documentation and figure out how to find users by an entry within an array.

Time’s Up! Let’s Review.



Instructor Demonstration

Removing, Updating and Dropping in MongoDB

now that we know how to create and read elements within a

mongoDB.

Removing, Updating and Dropping in MongoDB

```
> db.collectionName.update()
```

>> The method `update()` takes in two objects as its parameters, and it will only update the first entry that matches.

First object: What document(s) to search from.
Second object: What values to change.

```
> db.destinations.update({"country": "USA"}, {$set: {"continent": "Antarctica"}})
```

>> The `updateMany()` method can be used to update multiple documents instead. This method will update all of the records that meet the given criterion.

```
> db.destinations.updateMany({"continent": "Europe"}, {$set: {"continent": "Antarctica"}})
```

Removing, Updating and Dropping in MongoDB

>> ***Question***: In a given scenario where the field `{"capital": "Rome"}` has not yet been inputted, what will happen when we run the following command?

```
> db.destinations.update({"country": "Italy"}, {$set: {"capital": "Rome"}})
```

>> In the event where the document being searched within a collection does not exist, the parameter `{upsert:true}` must be passed in order to create the nonexistent document.

```
> db.destinations.update({"country": "Brazil"}, {$set: {"capital": "Brasilia"}}, {upsert: true})
```

>> The `$push` added command will add a value into the array. That will substitute the `$set` command.

```
> db.destinations.update({"country": "Italy"}, {$push: {"major_cities": "Siena"}})
```

Removing, Updating and Dropping in MongoDB

```
> db.collectionName.remove({})
```

>> In order to delete documents from a Mongo collection simply pass an empty object into the `remove()` method. Note that this command is extremely risky as ***ALL DOCUMENTS*** from the collection will drop and ***ALL DATA*** will be lost.

```
> db.destinations.remove({})
```

>> Passing an object into `remove()` method will stipulate what `{key:value}` pairing to search for. Adding the `justOne` parameter will remove only a single document. Without passing the `justOne` parameter, all documents matching the `{key:value}` pairing will be dropped from the collection.

```
> db.destinations.remove({"country": "USA"}, {justOne: true})
```

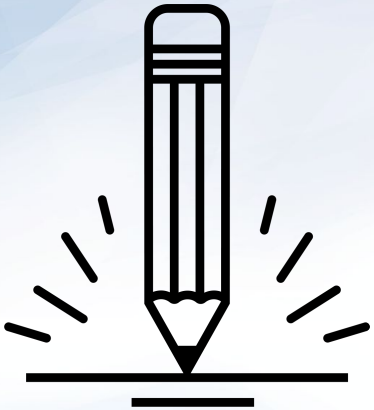
```
> db.collectionName.drop()
```

>> This method will delete the collection named from the database.

```
> db.destinations.drop()
```

>> While the following method will delete the whole database.

```
> db.dropDatabase()
```



Activity: Dumpster_DB

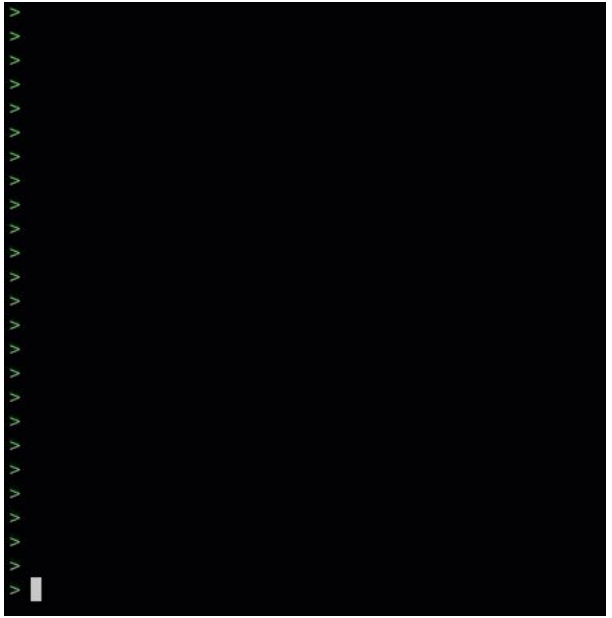
In this activity, you will gain further practice with CRUD operations in MongoDB as they create a database centered around dumpster diving.

Suggested Time:
15 Minutes

Instructions:

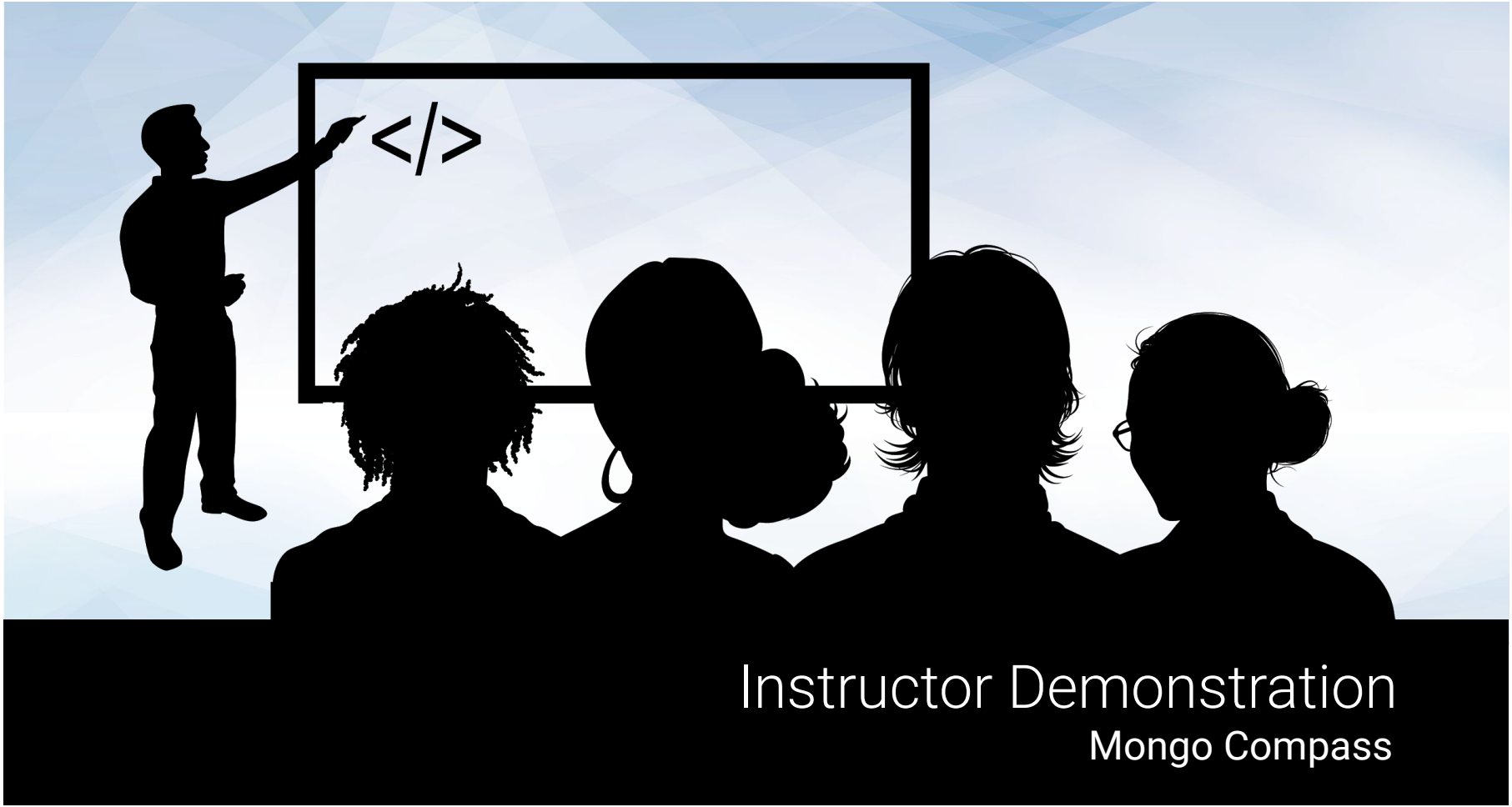
Activity: Dumpster_DB

- Create a collection called divers which will contain a string field for name, an integer field for yearsDiving, a boolean field for `stillDiving`, and an array of strings for `bestFinds`.
- Create and use a new database called `Dumpster_DB` using the Mongo shell.
- Insert three new documents into the collection. Be creative with what you put in here and have some fun with it.
- Update the `yearsDiving` fields for your documents so that they are one greater than their original values.
- Update the `stillDiving` value for one of the documents so that it is now false.
- Push a new value into the `bestFinds` array for one of the documents.



- Look through the collection, find the diver with the smallest number of `bestFinds`, and remove it from the collection.

Time’s Up! Let’s Review.



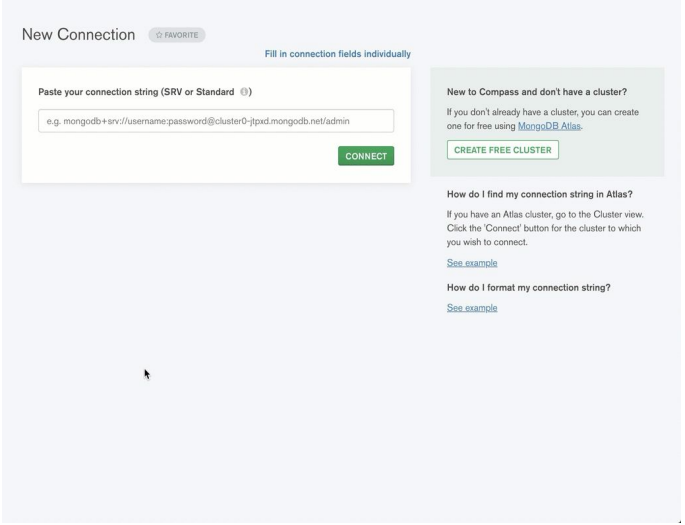
Instructor Demonstration

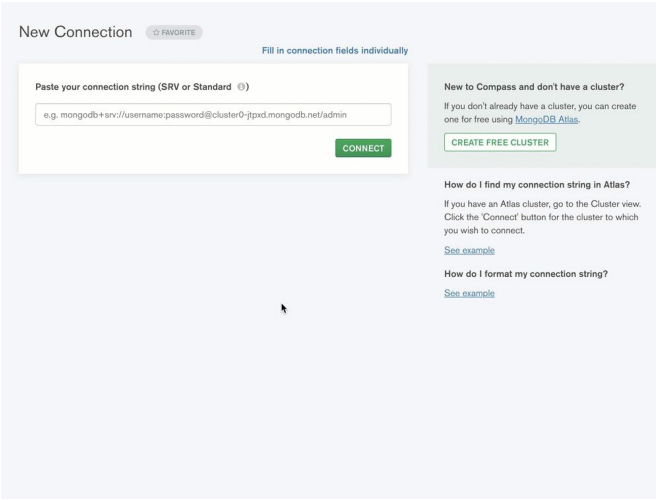
Mongo Compass

Mongo Compass

There are couple of ways to connect to localhost:

- ➔ **Connection String**: By default the string connection page will open as the initial page when you open MongoDB Compass.
- ➔ To connect using the the string method copy and paste these parameters : **mongodb://localhost:27017/?readPreference=primary&appname=MongoDB%20Compass&ssl=false** and click ***connect***.





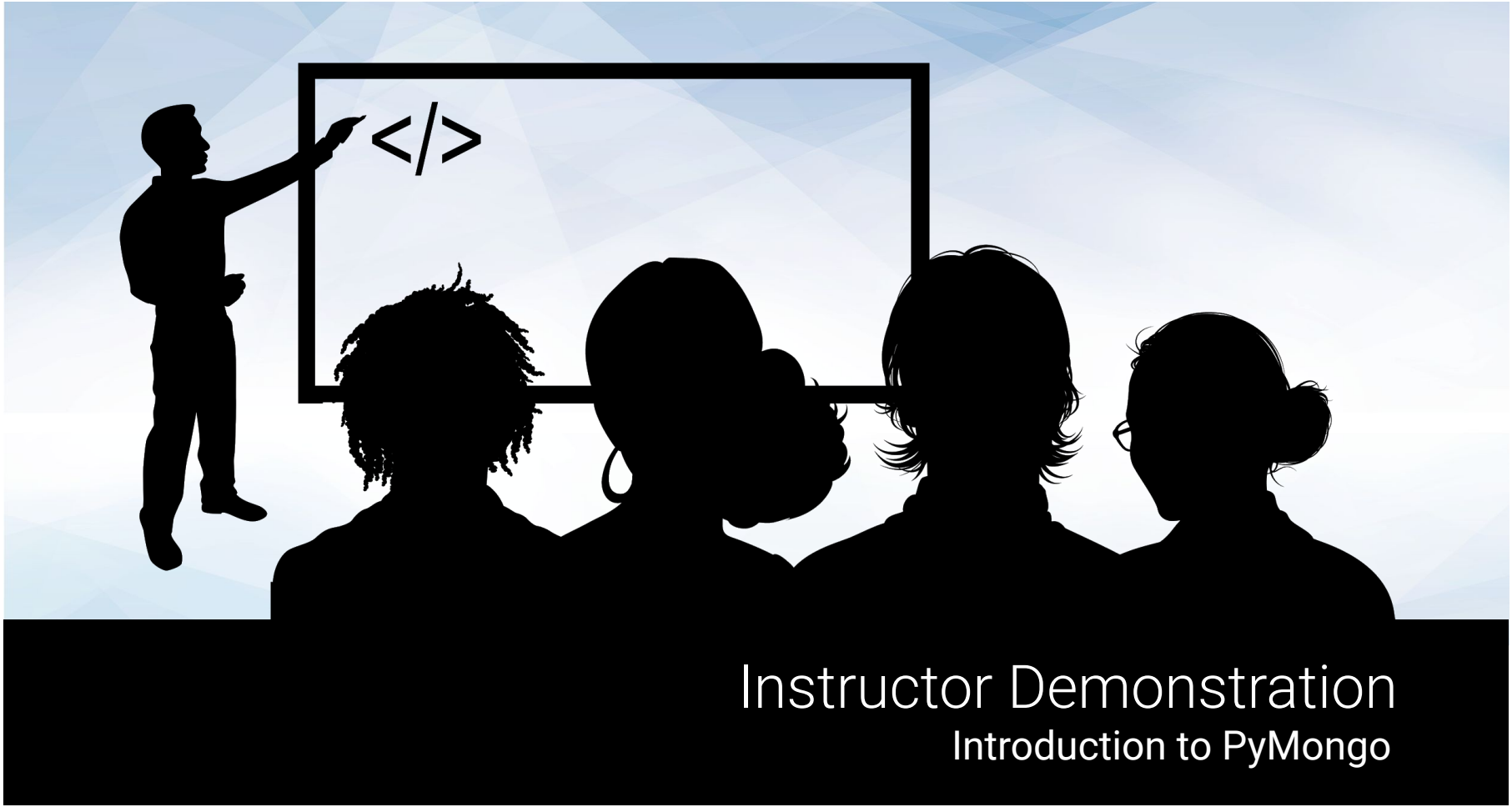
- ➔ **Fill in connection:** This is a seamless process where the default values for the connection are always set and ready to connect to localhost. Just keep in mind `mongod` for windows user and/or `brew services start mongodb-community` for macOs users must be running by the time you hit “connect” in order to take advantage of it.
- ➔ By the default the opening page will shoe the string connection method. Simply click *“Fill in connection fields individually”* followed by ***connect.***

Time to Code

Countdown timer

15:00

(with alarm)



Instructor Demonstration

Introduction to PyMongo

Introduction to PyMongo

➔ **PyMongo** is a library that allows interaction with MongoDB database through Python. It is a native driver for MongoDB.

Decomposing the basic code for PyMongo

➔ Follow the instructors commands to Install PyMongo into your environment.

➔ Once PyMongo is installed into your machine, open Jupyter Notebooks and import the module in your first line of code. In case the code executed follows by an error message, something went wrong with you installation. In that case try to find out what went wrong and remediate it. Ask for help if needed it.

➔ Creating a connection with a running instance.

```
In [4]: # import dependecy
        import pymongo
```

```
In [5]: # creating a connection

        conn = 'mongodb://localhost:27017'
        client = pymongo.MongoClient(conn)
```

Introduction to PyMongo

➔ Create a database named `classDB` and assigned to a variable called `db` using `client.classDB`. Follow by our first query.

In [8]:
```
# defining the 'classDB' database in Mongo

db = client.classDB
```

In [9]:
```
# quering all students

classroom = db.classroon.find()

# iterates through each students in the collection 'classroom'

for students in classroom:
    print(student)
```

➔ Creating and inserting our first document into a collection.

In [20]:
```
# inserting a document into the 'students' collection
db.classroom.insert_one(
    {
        'name': 'Ahmed',
        'row': 3,
        'favorite_python_library': 'Matplotlib',
        'hobbies': ['Running', 'Stargazing', 'Reading']
    }
)
```

Introduction to PyMongo

➔ Updating a document. Adding an item to a document array.

In [26]:
```
# updating a document

db.classroom.update_one(
    {'name': 'Ahmed'},
    {'$set':
        {'row': 4}
     }
)

# adding an item to a document array

db.classroom.update_one(
    {'name': 'Ahmed'},
    {'$push':
        {'hobbies': 'Listening to country music'}
     }
)
```

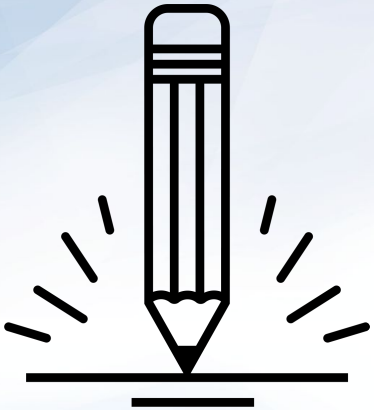
Introduction to PyMongo

➔ Deleting and removing.

In [27]:
```
# deleting a field from a document

db.classroom.update_one({'name': 'Ahmed'},
                        {'$unset':
                         {'row': ""}
                         }
                        )

# deleting a document from a collection
db.classroom.delete_one(
    {'name': 'Ahmed'}
)
```



Activity: Mongo Grove

In this activity, you will build a command-line interface application for the produce department of a supermarket using PyMongo to enable Python to interact with MongoDB.

Suggested Time:
15 Minutes

Instructions:

Activity: Mongo Grove

- Use PyMongo to create a `fruits_db` database, and a `fruits` collection.
- Into that collection, insert two documents of fruit shipments received by your supermarket. They should contain the following information: vendor name, type of fruit, quantity received, and ripeness rating (1 for unripe, 2 for ripe, 3 for over-ripe).
- Because not every supermarket employee is versed in using MongoDB, your task is to build an easy-to-use app that can be run from the console.
- Build a Python script that asks the user for the above information, then inserts a document into a MongoDB database.
- It would be good to Modify the app so that when the record is entered, the current date and time is automatically inserted into the document.

- **Hint:**
  - Consult the documentation on the `datetime` library.

Time’s Up! Let’s Review.


The End