**IT8701 Introduction to Programming for Data Science**

**Lab 06 – Creating and manipulating data in NoSQL databases**

**What you will learn / do in this lab**

1. How to store and manipulate data stored in NoSQL databases such as MongoDB
2. https://docs.mongodb.com/manual/crud/

**Table of Contents**

[1. Overview 3](#_Toc534549675)

[A. What you will do for this lab 3](#_Toc534549676)

[B. Intro to MongoDB 3](#_Toc534549677)

[2. Install MongoDB on your laptop 4](#_Toc534549678)

[A. Windows Installation 4](#_Toc534549679)

[3. Start MongoDB Server 5](#_Toc534549680)

[A. Set up the MongoDB environment 5](#_Toc534549681)

[B. Start MongoDB with mongod.exe 5](#_Toc534549682)

[C. Verify that MongoDB started successfully 6](#_Toc534549683)

[4. Use MongoDB with Mongo Shell 7](#_Toc534549684)

[A. Start mongo.exe 7](#_Toc534549685)

[B. View the databases in Mongo 7](#_Toc534549686)

[C. Switch to a specific database 9](#_Toc534549687)

[D. View collections 9](#_Toc534549688)

[E. Insert new documents into a collection 10](#_Toc534549689)

[Add one document 10](#_Toc534549690)

[Add multiple documents 10](#_Toc534549691)

[F. View documents in collection 11](#_Toc534549692)

[View all documents 11](#_Toc534549693)

[View document with criteria, projection and limit 12](#_Toc534549694)

[Sort documents 13](#_Toc534549695)

[G. Update documents in collection 14](#_Toc534549696)

[Update all documents 14](#_Toc534549697)

[Update specific documents with specific fields 14](#_Toc534549698)

[H. Delete documents in collection 17](#_Toc534549699)

[Delete specific documents with specific criteria 17](#_Toc534549700)

[Delete all documents 18](#_Toc534549701)

[5. Write Python code to connect to Mongo 20](#_Toc534549702)

[A. Install pymongo 20](#_Toc534549703)

[Use conda to install the library 20](#_Toc534549704)

[B. Display records from Mongo with Python 21](#_Toc534549705)

[C. Insert records into Mongo with Python 21](#_Toc534549706)

[D. Update records in Mongo with Python 22](#_Toc534549707)

[E. Delete records in Mongo with Python 23](#_Toc534549708)

# Overview

## What you will do for this lab

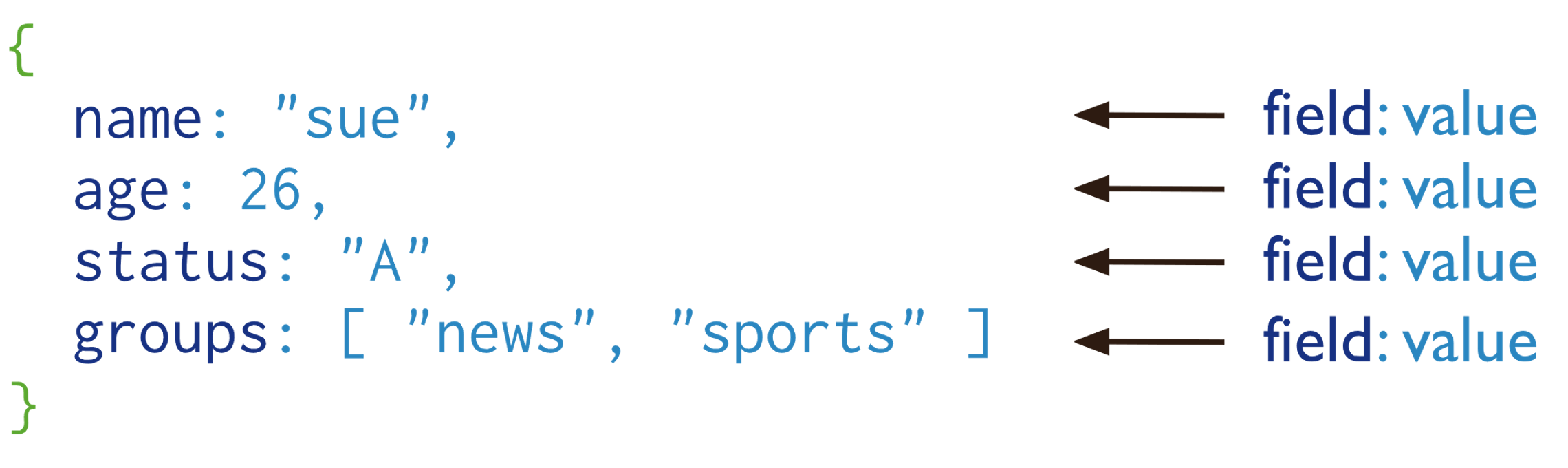
In this lab, you will learn how to set up your own NoSQL database server (MongoDB) on your laptop and manipulate the data stored in the database using two methods

1. MongoDB command-line tool (mongo.exe)
2. via Python code.

## Intro to MongoDB

MongoDB is an open-source NoSQL database that provides high performance, high availability, and automatic scaling.

A record in MongoDB is a document, which is a data structure composed of field and value pairs. For example, the following shows a record which stores user information.



MongoDB stores its documents in collections; for example, the document above can be stored in a collection called “users”. If you have documents that contain product information, you may store them in a collection called “products”. You can think of collections as the equivalent to tables in relational databases. Finally, collections are stored in databases. So the users and products collections could be stored inside a database called mydatabase.

# Install MongoDB on your laptop

## Windows Installation

| No | Task |
| --- | --- |
|  | Download the latest production release of **MongoDB Community Server** from the MongoDB Download Center (180 MB file)  <https://www.mongodb.com/download-center/community> |
|  | * Locate the downloaded MongoDB .msi file, which typically is located in the default Downloads folder and double-click the .msi file * A set of screens will appear to guide you through the installation process * Choose “Full Installation” which will nstall MongoDB to C:\Program Files\MongoDB\Server\verx.x\ by default * During the installation process you will be given the option to install MongoDB Compass in addition to MongoDB Server, select Yes |

# Start MongoDB Server

## Set up the MongoDB environment

| No | Task |
| --- | --- |
|  | MongoDB requires a data directory to store all data.  MongoDB’s default data directory path is the absolute path \data\db on the drive from which you start MongoDB  Create this folder by running the following command in a Command Prompt  mkdir C:\data  mkdir C:\data\db |

## Start MongoDB with mongod.exe

| No | Task |
| --- | --- |
|  | To start MongoDB, run mongod.exe. For example, from the Command Prompt:  "C:\Program Files\MongoDB\Server\3.6\bin\mongod.exe"  This starts the main MongoDB database process. The waiting for connections message in the console output indicates that the mongod.exe process is running successfully.  Depending on the security level of your system, Windows may pop up a Security Alert dialog box about blocking “some features” of C:\Program Files\MongoDB\Server\3.6\bin\mongod.exe from communicating on networks. All users should select Private Networks, such as my home or work network and click Allow access. |

## Verify that MongoDB started successfully

| No | Task |
| --- | --- |
|  | Verify that MongoDB has started successfully by checking the process output for the following line:  [initandlisten] waiting for connections on port 27017  The output should be visible in the terminal or shell window.  You may see non-critical warnings in the process output. As long as you see the log line shown above, you can safely ignore these warnings during your initial evaluation of MongoDB. |

# Use MongoDB with Mongo Shell

## Start mongo.exe

| No | Task |
| --- | --- |
|  | We can connect to our Mongo Server using a command-line tool called mongo.exe.  <https://docs.mongodb.com/manual/reference/method/#collection>  <https://docs.mongodb.com/manual/reference/method/#cursor> |
|  | Open a new Command prompt window and type the following command  "C:\Program Files\MongoDB\Server\3.6\bin\mongo.exe" |
|  | You should see output similar to that below with the following starting lines as shown in screen 1 and ending lines in screen 2 |
|  |  |
|  |  |

## View the databases in Mongo

| No | Task |
| --- | --- |
|  | We can know what databases are there already in Mongo Server by typing this command at the > symbol  >show dbs |
|  | For example, this is what I see in my MongoDB server.  You will not see the same output on your screen, as you probably have less databases than me since yours is a fresh installation while I already have been creating many other databases since I first installed. |

## Switch to a specific database

| No | Task |
| --- | --- |
|  | We can switch to a particular database by typing the use command e.g.  >use doradatabase |
|  | This is the output you will see |

## View collections

| No | Task |
| --- | --- |
|  | MongoDB uses the concept of “collections” to represent tables in a database. |
|  | We can view the collections in a Mongo database using the show collection command  >show collections |
|  | For example, I already have a database called mycompanydata and as you can see from the screen below, once I switch over to this database using the use command, I can see there are 2 collections in it, named inventory and users respectively. |

## Insert new documents into a collection

### Add one document

| No | Task |
| --- | --- |
|  | In this section, let’s practise how to create a new database, a new collection and a new document all at once. |
|  | First, let’s start a new database  >use it8701\_database |
|  | let’s insert one new document into this database with the following command  > db.users.insertOne({name: "dora",age:44, status: "pending"}) |
|  | You should see the following output if the insertion was successful |

### Add multiple documents

| No | Task |
| --- | --- |
|  | Now that you know how to insert a single document, let’s practise how to insert multiple documents. |
|  | Ensure you are in the correct database  >use it8701\_database |
|  | Next, let’s insert a few new documents into this database with the following command  > db.users.insertMany([{name: "sue",age: 26,status: "confirmed"},{name: "zan",age: 18,status: "pending"},{name: "don",age: 20,status: "confirmed"}]) |
|  | You should see the following output if the insertion was successful |

## View documents in collection

### View all documents

| No | Task |
| --- | --- |
|  | In this section, let’s practise how to view the newly created documents you added in the previous section. |
|  | First, make sure you are in the correct database  >use it8701\_database |
|  | Type the following command to view ALL documents  > db.users.find({}) |
|  | You should see the following output if the query was successful |

### View document with criteria, projection and limit

| No | Task |
| --- | --- |
|  | In this section, let’s practise how to view documents using specified criteria, projection and limit.  We can control the documents that are returned in a query by specifying equality conditions. E.g. db.users.find( { status: "Pending" } ) or db.users.find( { age: {$gt:40}})  We can also control the fields that are returned in a query by using projection by setting the <field> to 1. E.g. db.users.find( { status: "Pending" }, {name:1})  We can also control the number of documents that are returned by using the limit command. E.g. db.users.find({age: {$lt:45}}, {name:1,age:2}).limit(3) |
|  | First, make sure you are in the correct database  >use it8701\_database |
|  | Type the following command to view the names of the first 3 users who have age less than 45  > db.users.find({age: {$lt:45}}, {name:1,age:2}).limit(3) |
|  | You should see the following output if the query was successful |

### Sort documents

| No | Task |
| --- | --- |
|  | In this section, let’s practise how we can modify the previous query with a sort instruction |
|  | First, make sure you are in the correct database  >use it8701\_database |
|  | Type the following command to view the names of the first 3 users who have age less than 45, sorted in ascending order (youngest first)  > db.users.find({age: {$lt:45}}, {name:1,age:2}).limit(3) .sort( { age: 1 } ) |
|  | You should see the following output if the query was successful |

## Update documents in collection

### Update all documents

| No | Task |
| --- | --- |
|  | In this section, let’s practise how to update **all** documents to have a status of pending |
|  | First, make sure you are in the correct database  >use it8701\_database |
|  | Type the following command to update ALL documents  > db.users.updateMany({}, {$set: {status: 'pending'}}) |
|  | You should see the following output if the query was successful |

### Update specific documents with specific fields

| No | Task |
| --- | --- |
|  | In this section, let’s practise how to update **specific** documents to have a status of pending as well as |
|  | First, make sure you are in the correct database  >use it8701\_database |
|  | Next, let’s insert some documents in a new collection so that we can update them later.  db.inventory.insertMany( [  { item: "canvas", qty: 100, size: { h: 28, w: 35.5, uom: "cm" }, status: "A" },  { item: "journal", qty: 25, size: { h: 14, w: 21, uom: "cm" }, status: "A" },  { item: "mat", qty: 85, size: { h: 27.9, w: 35.5, uom: "cm" }, status: "A" },  { item: "mousepad", qty: 25, size: { h: 19, w: 22.85, uom: "cm" }, status: "P" },  { item: "notebook", qty: 50, size: { h: 8.5, w: 11, uom: "in" }, status: "P" },  { item: "paper", qty: 100, size: { h: 8.5, w: 11, uom: "in" }, status: "D" },  { item: "planner", qty: 75, size: { h: 22.85, w: 30, uom: "cm" }, status: "D" },  { item: "postcard", qty: 45, size: { h: 10, w: 15.25, uom: "cm" }, status: "A" },  { item: "sketchbook", qty: 80, size: { h: 14, w: 21, uom: "cm" }, status: "A" },  { item: "sketch pad", qty: 95, size: { h: 22.85, w: 30.5, uom: "cm" }, status: "A" }  ] ); |
|  | Type the following command to update only those documents that have quantity less than 50.  db.inventory.updateMany(  { "qty": { $lt: 50 } },  {  $set: { "size.uom": "in", status: "P" },  $currentDate: { lastModified: true }  }  ) |
|  | You should see the following output if the query was successful |
|  | Let’s view the updated documents to verify  db.inventory.find({}) |
|  | You should see output similar to that below |

## Delete documents in collection

### Delete specific documents with specific criteria

| No | Task |
| --- | --- |
|  | In this section, let’s practise how to delete **specific** documents that fit a given criteria |
|  | First, make sure you are in the correct database  >use it8701\_database |
|  | Let’s view the current inventory collection to see the existing documents inside it.  db.inventory.find({ status: "D" }) |
|  | Note that there are 2 documents with status D inside this collection |
|  | Type the following command to delete the first documentt that has status equal to D.  db.inventory.deleteOne( { status: "D" } ) |
|  | You should see the following output if the operation was successful |
|  | Let’s view the updated collection to verify  db.inventory.find({ status: "D" }) |
|  | Note that there is now only one document left with status D inside this collection since the other one has been deleted. |

### Delete all documents

| No | Task |
| --- | --- |
|  | In this section, let’s practise how to delete **all** documents in the inventory collection. |
|  | First, make sure you are in the correct database  >use it8701\_database |
|  | First, let’s check how many documents are there in the entire inventory collection.  db.inventory.count({}) |
|  | You should see similar output as that below. |
|  | Type the following command to delete ALL documents in the inventory collection  > db.inventory.deleteMany({}) |
|  | You should see the following output if the query was successful |

# Write Python code to connect to Mongo

Now that you have mastered the basics of using SQL statements to view and manipulate the data in a relational database such as MySQL, let’s hone your skills further by writing Python code that reads data from various CSV file, stores the data into your MySQL database and then retrieves it and shows it as a visualization based on the user’s inputs.

## Install pymongo

In order to write Python code that can interface with the Mongo database, you need to install the pymongo library first. Follow the instructions below to install it on your laptop.

### Use conda to install the library

|  |  |
| --- | --- |
|  | Run the following command in a terminal window. If the command runs successfully, Anaconda will start to install the library and your package will be ready for use in less than 60 seconds.  conda install pymongo |

## Display records from Mongo with Python

Once you have installed the pymongo package successfully, you may start to connect to your Mongo database using Python code.

Execute the example below to connect to the users collection in IT8701 and display the records inside the collection.

|  |  |
| --- | --- |
|  | Type the following code in a file named “mongo\_lab\_display.py” |
|  | You should see the following output or similar |

|  |
| --- |
| ## SELECT  import pymongo  from pymongo import MongoClient  client = MongoClient()  db = client.it8701\_database  collection = db.users  for item in collection.find():  print("Name: {}, Age: {:.0f}, Status: {}".format(item['name'], item['age'], item['status'])) |

## Insert records into Mongo with Python

Let’s try to insert documents into Mongo with this exercise

|  |  |
| --- | --- |
|  | Type the following code in a file named “mongo\_lab\_insert.py” |
|  | You should see the following output or similar |

|  |
| --- |
| ## INSERT  import pymongo  from pymongo import MongoClient  client = MongoClient()  db = client.it8701\_database  collection = db.users  collection.insert\_many([  {"name": "mary", "age":25,"status": "pending"},  {"name": "john", "age":33,"status": "confirmed"},  {"name": "robert", "age":55,"status": "confirmed"}])  for item in collection.find():  print("Name: {}, Age: {:.0f}, Status: {}".format(item['name'], item['age'], item['status'])) |

## Update records in Mongo with Python

Let’s try to update existing documents in Mongo with this exercise

|  |  |
| --- | --- |
|  | Type the following code in a file named “mongo\_lab\_update.py” |
|  | You should see the following output or similar |

|  |
| --- |
| ## UPDATE  import pymongo  from pymongo import MongoClient  client = MongoClient()  db = client.it8701\_database  collection = db.users  collection.update\_many(  {"age": {"$lt": 20}} ,  {"$set": {"status": "pending"}})  for item in collection.find():  print("Name: {}, Age: {:.0f}, Status: {}".format(item['name'], item['age'], item['status'])) |

## Delete records in Mongo with Python

This last exercise will let you practise how to delete a Mongo document

|  |  |
| --- | --- |
|  | Type the following code in a file named “mongo\_lab\_delete.py” |
|  | You should see the following output or similar |

|  |
| --- |
| ## DELETE  import pymongo  from pymongo import MongoClient  client = MongoClient()  db = client.it8701\_database  collection = db.users  collection.delete\_many({"age": {"$lt": 20}})  for item in collection.find():  print("Name: {}, Age: {:.0f}, Status: {}".format(item['name'], item['age'], item['status'])) |

**-- End of Lab --**