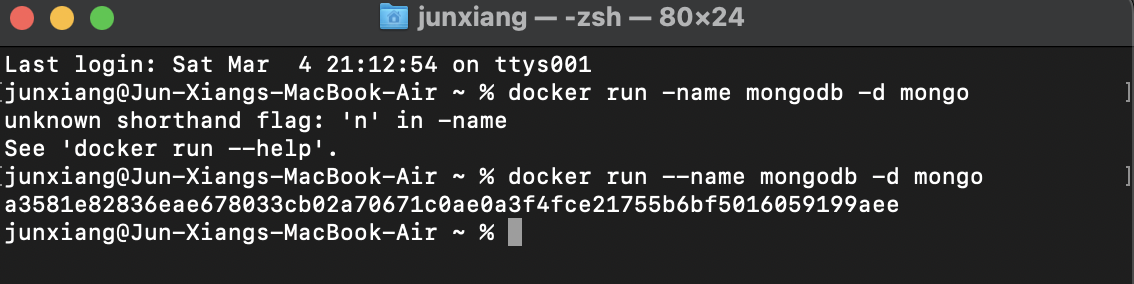
**Question 1:** Activate/run Docker and install/pull MongoDB in Docker. You are

suggested to use command-line tools, e.g., PowerShell in Windows.

(Although not the focus of this module, you are expected to be able

to develop efficiently in Linux.)

Command used:

1. docker run –name mongodb -d mongo  
   

Question 2: Start a Docker container listening on port 27019 (default MongoDB

port). Specify the name as mongo\_<student\_id>, e.g., mongo\_2101234.

Commands used:

1. docker run -d -p 27019:27019 –name mongo\_2100582 mongo  
   Text

   Description automatically generated

Question 3: Enter the docker container with MongoDB just installed. You may see

the account (root) is no longer the one for your computer.

Commands used:

1. docker exec -it mongo\_2100582 bash  
   Text

   Description automatically generated

Question 4: Enter MongoDB by typing mongo. You should be in the MongoDB environment already.

Commands used:

1. mongosh  
   Text

   Description automatically generated

Question 5: Create a database called DB23<student\_name>, e.g., DB23Wei. (Do you have to check if it exists in MongoDB?)

Command Used:

1. use DB23LauJunXiang  
   Text

   Description automatically generated

Question 6: Create a collection Students.

Command used:

1. db.createCollection(“Students”)  
   Graphical user interface

   Description automatically generated

Question 7: Insert the below student data into the Students collection. Each row  
corresponds to three attributes, namely sname, syear, and major.  
Iron Man 2019 engineering  
Deadpool 2019 biology  
Wolverine 2020 physics  
Hulk 2020 physics  
Thor 2021 physics  
Rocket 2021 engineering  
<student\_name> 2021 engineering

Command used:

1. db.Students.insertOne({sname: "Iron Man", syear: 2019, major: "engineering"})
2. db.Students.insertOne({sname: "Deadpool", syear: 2019, major: "biology"})
3. db.Students.insertOne({sname: "Wolverine", syear: 2020, major: "physics"})
4. db.Students.insertOne({sname: "Hulk", syear: 2020, major: "physics"})
5. db.Students.insertOne({sname: "Thor", syear: 2021, major: "physics"})
6. db.Students.insertOne({sname: "Rocket", syear: 2021, major: "engineering"})
7. db.Students.insertOne({sname: "Lau Jun Xiang", syear: 2021, major: "engineering"})  
   Text

   Description automatically generated

Question 8: Display data in the Students collection. Observe if we get new data besides what we just inserted.

Commands used:

1. db.students.find()  
   Text

   Description automatically generated

Question 9: Deadpool wants to switch major from biology to art, please update  
the collection accordingly.

Command used:

1. db.Students.updateOne({sname:”Deadpool”},{$set: {major:”art”}})  
   db.Students.find({sname:”Deadpool”})  
   Text

   Description automatically generated

Question 10: Add a new field campus with the default value “SIT@NYP” for all  
students. (You may display the collection to verify. If the update  
did not apply to all students, why and how to solve it?)

Commands used:

1. db.Students.updateMany({}, {$set: {campus: “SIT@NYP”}})  
   Text

   Description automatically generated
2. db.Students.find()  
   Text

   Description automatically generated
3. If it did not update, it probably is because campus has already existed in one of them. We can solve it by using   
   db.Students.updateMany({}, {$set: {campus: “SIT@NYP”}}, {upsert:true}). The command will insert if it does not exists or update if it exists.

Question 11: Add a new filed GPA with a default value of 4.0 for all students

Commands used:

1. db.Students.updateMany( {} , {$set: {GPA:4.0}})  
   Text

   Description automatically generated
2. db.Students.find()  
   Text

   Description automatically generated

Question 12:

Commands used:

1. db.Students.updateOne({sname: “Rocket”} , {$set: {GPA:5.0}})

db.Students.find({sname:”Rocket”})  


Question 13: Find the students whose name has a character “a”.

Commands used:

1. db.Students.find({sname: /a/})  
   Text

   Description automatically generated

Question 14: Find engineering students.

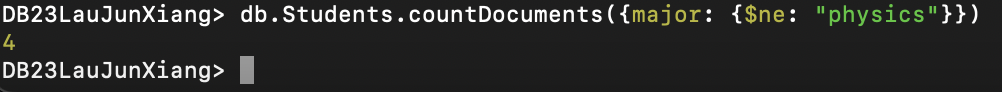
Commands used:

1. db.Students.find({major: “engineering”})  
   Text

   Description automatically generated

Question 15: Return the number of non-physics students

Command used:

1. db.Students.countDocuments({major: {$ne: “physics”}})  
   

Question 16: Find the students whose syear > 2020 and major in engineering

Command used:

1. db.Students.find({syear: {$gt: 2020}, major: “engineering”})  
   Text

   Description automatically generated

Question 17:

Commands used:

1. db.Students.find().sort({syear: -1})  
   Text

   Description automatically generated Text

   Description automatically generated

Question 18: Display the number of students for each major and display them in descending order of the number of students.

Command used:

1. db.Students.aggregate([  
   {$group: {\_id: “$major”, count: {$sum: 1}}},  
   {$sort: {count: -1}}  
   ])  
   Text

   Description automatically generated