**AUTM2023 CSCI235 Week7 Lab7**

**MongoDB CRUD1: Create and Read**

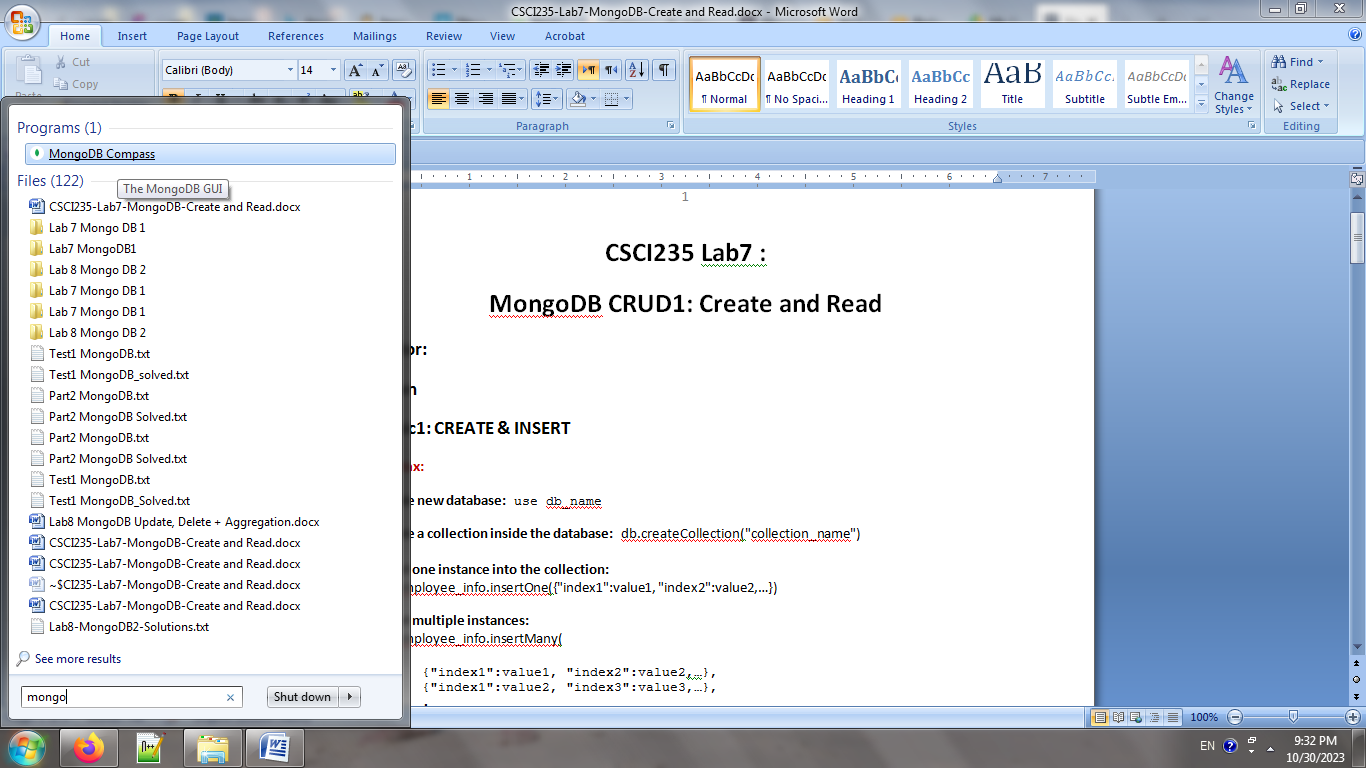
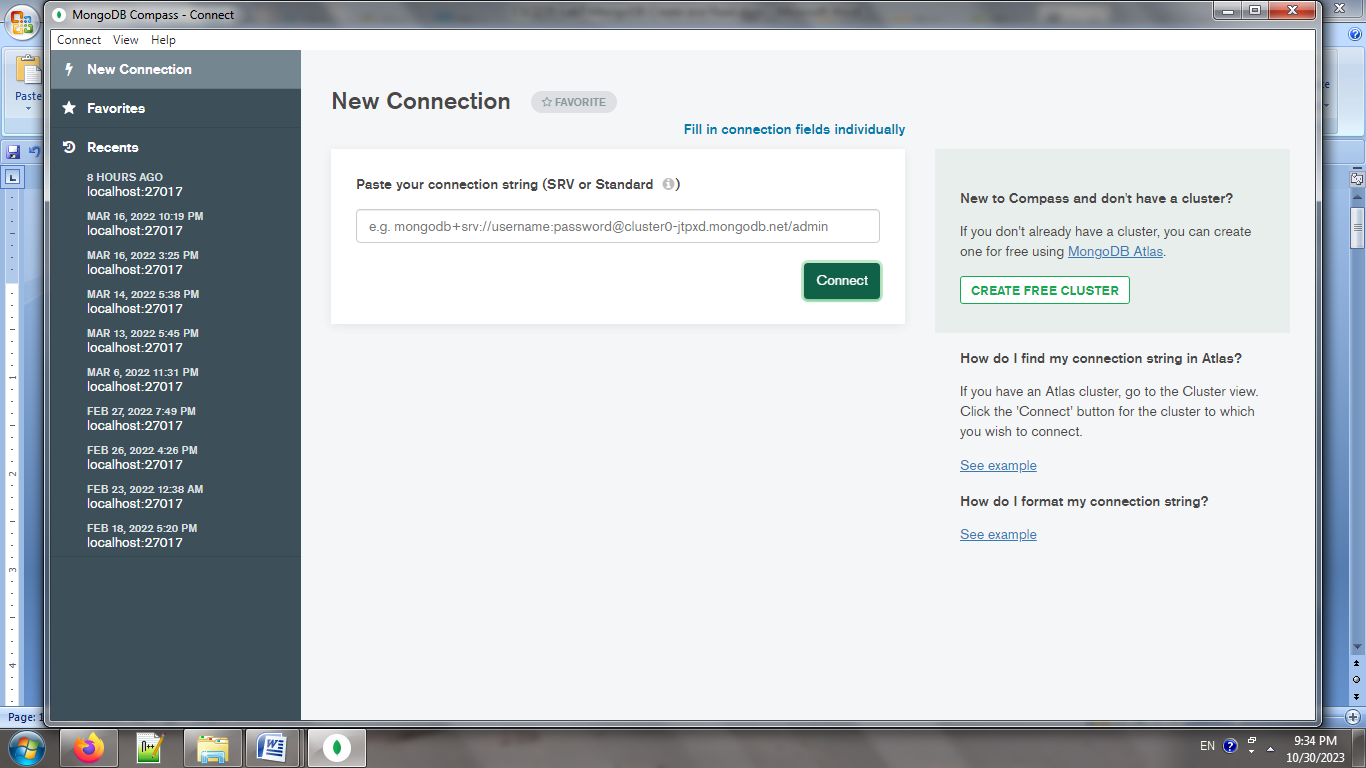
**MongoDB community installation:**

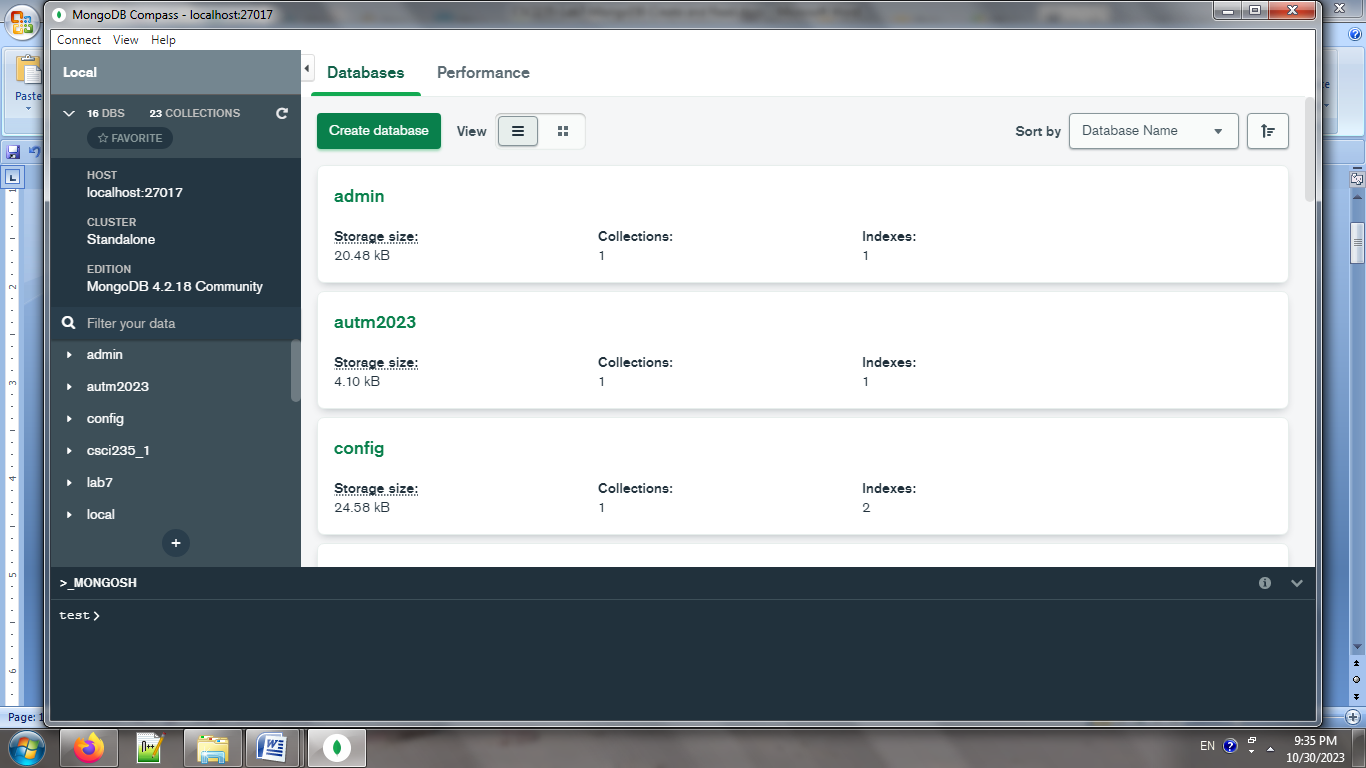
* Visit <https://www.mongodb.com/try/download/community>
* Select package based on your operating system
* Run the installation file to get MongoDB server installed

**MongoDB Compass installation:**

* Visit <https://www.mongodb.com/try/download/compass>
* Select package based on your operating system

**Editor:**

* Open MongoDB Compass****
* Click on “Connect” ****
* We will use MONGOSH (mongo shell) as the bottom of the compass as the editor

****

**Topic1: CREATE & INSERT**

**Syntax:**

**Create new database:** use db\_name

**Create a collection inside the database:** db.createCollection("collection\_name")

**Insert one instance into the collection:**

db.employee\_info.insertOne({"index1":value1, "index2":value2,…})

**Insert multiple instances:**

db.employee\_info.insertMany(

[

{"index1":value1, "index2":value2,…},

{"index1":value2, "index3":value3,…},

.

.

{"index100":value100, "index2":value2,…}

]

)

**NOTE: VALUE COULD BE STRING, NUMBER, AN ARRAY, A DOCUMENT, AN ARRAY OF DOCUMENT, OR EVEN A NESTED DOCUMENT**

**TASK0:**

1. If using MongoDB on Lab Terminals then run below to delete all existing collections:

db.getCollectionNames().forEach(

function(collection\_name) {

db[collection\_name].remove()

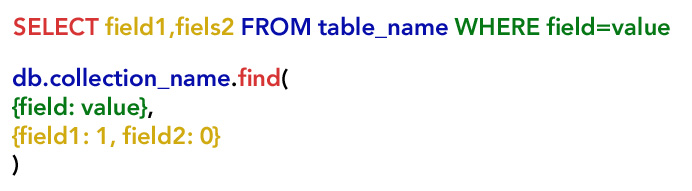
}

);

1. Create a new database called Lab7
2. Create “Employee” collection
3. Insert all the values from employee.txt into the database

**Topic 2: READ/QUERY**

1. ***Mapping read in mongodb with read in sql***



|  |  |
| --- | --- |
| Select \* from employee | db.Employee.find({}) |
| Select \* from employee where position= 'Teacher' | db.Employee.find(  { "Position": "Teacher"}  ) |
| Select name, gender, department from employee where position= 'Teacher' | db.employee.find(  {"Position":"Teacher"},  {"Name":1,"Gender":1,  "Department":1}) |
| Select name, gender, department from employee | db.employee.find(  {},  {"Name":1,"Gender":1,  "Department":1, "\_id":0}) |
| Select \* from employee where position <> 'Teacher' | db.Employee.find(  { "Position": { $ne:"Teacher"}}) |
| Select \* from employee where position= 'Teacher' AND gender= 'F' | db.Employee.find(  { "Position": "Teacher",  "Gender ": "F " }  )  NOTE:, is used for AND |
| Select \* from employee where position= 'Teacher' OR position= 'Technical Support' | db.Employee.find({$or: [{"Position":"Teacher"}, {"Gender":"F"}]})  NOTE:  $or is followed by an array []  Each condition needs to be inside a {} separated by , |
| Select \* from employee where position IN ('Teacher','Technical Support') | db.Employee.find({"Position": {$in:["Teacher","Technical Support"]}})  NOTE: $nin for not in |
| Select \* from employee where height>170 | db.Employee.find ({"Height":{$gt:170}})  NOTE:  $gt: >, $lt: <, $gte >=, $lte <=, $eq: = |

1. ***Projection 1: read from embedded document***

**Syntax:** "field.nested\_field:value"

**EXAMPLE:** find all employees whose first name is Asma

db.employee.find({ "Name.Fname": "Asma "})

1. ***Find value in array field***

|  |  |  |
| --- | --- | --- |
| **Explanation** | **Syntax** | **Example** |
| Search for *specific value* in the array | {array\_field:value} | db.employee.find({"Courses":"CSCI218"}) |
| Find if an array is equal to a given array  *All the values must be matched in the exact same order* | {array\_field:[value1,value1]} | db.employee.find({"Courses":  ["CSCI235","CSCI218","CSIT115"]}) |
| Search for *a list of* values in the array | {array\_field:{$all:[value1,value1]}} | db.employee.find({"Courses":{$all:["CSCI235","CSIT115"]}}) |
| Find an array that at least one element/documents satisfies at least one of the conditions | {array\_field: { <operator1>: <value1>,<operator1>: <value1} } | db.employee.find({"TestArray":{$lt:100,$gt:80}}) |
| Return an array that at least one of the elements/documents satisfies all of the conditions | { array\_field: { $elemMatch: { <operator>: value1, <Operator>: value2 } } }  If list of documents:  { array\_field: { $elemMatch: { doc\_field:{<operator>: value1, <Operator>: value2 } } }} | db.employee.find({"TestArray":{$elemMatch:{$lt:100,$gt:80}}) |
| Return documents with array of specific size | { "array\_field": { $size: value } } | db.employee.find({"Courses":{$size:2}}) |
| Check array value based on the element index | { "array\_field.index": value } | db.employee.find({"Courses.0":"CSCI235"}) |

1. ***Access field in array of nested documents***

***Access Specific Field:*** {Array\_field.document\_field: value}

show matching array :db.employee.find({"Projects.Duration":15},{"\_id":0,"Projects":1})

show matching document: db.employee.find({"Projects.Duration":15},{"\_id":0,"Projects.$":1})

***Access All Fiels:*** {Array\_field:{document\_field:value}}

db.employee.find({"Projects":{"Name":"New Website","Duration":15}})

1. ***$Exist Operator***

{field:{$exists : 0}} means return documents where the specified field does not exist

1. ***CountDocument***

db.Collectio\_name.countDocuments({Conditions})

db.employee.countDocuments() return number of all documents

**TASKS**

**TASK1:**

1. How many female employees are working in the company?
2. How many female employees are working in IT Department
3. How many female employees are working in any department except IT, HR or Marketing

**TASK2:**

1. show name, gender, position and department of non-teaching staff
2. show all details except salary, date of birth, height and default id of non-teaching staffs

**TASK3:**

1. Show all details of all employees who are working in IT **and** are responsible for Daily Support
2. Show ONLY department and salary details of all employees who are working in IT **or** are responsible for Daily Support
3. Show ONLY department and salary details of all employees who are working in IT or are responsible for Daily Support or Accounting

**TASK4:**

1. Show hourly employees who are paid between 100 and 200 (including 100 and 200)
2. Show hourly employees who are paid between 50 and 200 (including 50 and 200) and has a job list
3. Show hourly employees who are paid between 50 and 200 (including 20 and 200) and has **at least** 2 jobs in their job lists

**TASK5:**

For all full time employees hired before 2022, show their Hired Date and net salary. (HINT use new Date)

**TASK6:**

1. Show Employee Name, department and project list of employees working on projects with budget between 100000 and 500000 (**you** **must use $elemMatch**)
2. Change above query to show only the projects with matching budget range