Activity 15

1. **Instillations** ○ https://www.mongodb.com/download-center/community
2. Download and Install MongoDB community server
   - Create a separate installation location/directory "mongodb" (for windows, c:\mongodb) and install your MongoDB in that location instead of default location. (this will be helpful later when you are starting the service)
   - Start custom installation option ○ Uncheck "Install MongoD as a Service" option and hit next.
3. Create 3 folders inside mongodb after installations ○ Create a folder "data" c:\mongodb\data
   - Create a folder "db" inside data folder created above. c:\mongodb\data\db ○ Create a folder "log" c:\mongodb\log
4. Use Command Interpreter (cmd for windows, open with admin privilege) ○ Change the path from command line to where you have your mongoDB\bin folder ○ Now type
   - mongod –directoryperdb –dbpath c:\mongodb\data\db –logpath c:\mongodb\log\mongo.log –logappend –install
5. Start the MongoDB service.
   - Type of the followings
     - net start MongoDB

---

1. Start MongoDB ○ type mongo to start mongo shell ○ Cls to clear the screen

2. To show the databases ○ show dbs
   - use <database name> will use and switch to that database. If there's no database, this command will create one.
   - db will tell you current db
   - **[Exercise]** Create a database "Company"
   - **[Exercise]** Create a database "University"

```
University> use Company
switched to db Company
Company> use University
switched to db University
University> db
University
```

3. [Exercise] To drop a database, ○ Use db to find the current database
   - db.dropDatabase();
   - **[Exercise]** Drop "University"

```
University> db.dropDatabase("University");
{ ok: 1, dropped: 'University' }
University> use Company
switched to db Company
```

4. **[Exercise]** Create user for the database “Company”

```
db.createUser(
  {
    user: "John",
pwd: "1234",
    roles: [ "readWrite", "dbAdmin" ]
  }
)
```

- MongoDB stores BSON documents, i.e. data records, in collections; the collections in databases. BSON is a binary representation of JSON documents
- Database is a physical container for collections.
- Collection is a group of MongoDB documents. It is the equivalent of an RDBMS table. A collection exists within a single database. Collections do not enforce a schema. Documents within a collection can have different fields. Typically, all documents in a collection are of similar or related purpose.

```
Company> db.createUser({user: "John", pwd: "1234", roles:["readWrite", "dbAdmin"]
{ ok: 1 }
```

5. **[Exercise]** Create a collection “customers” for Company database ○ db.createCollection(‘customers’); ○ show collections

```
Company> db.createCollection('customers')
{ ok: 1 }
Company> show collections
customers
```

6. Insert documents to collection ○ db.customers.insert({first_name:"Jon", last_name:"Doe"}); ○ **[Exercise]** Create 5 customers and the fields for their first_name and last_name:
   - John Smith, Alicia Zelaya, Jennifer Wallace, Ahmad Jabbar, James Borg

```
Company> db.customers.insert({first_name:"Jon", last_name:"Doe"});
DeprecationWarning: Collection.insert() is deprecated. Use insertOne, insertMany, or bulkWr
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('67082c0c26667e5b2086b01d') }
}
Company> db.customers.insertOne({first_name:"John", last_name:"Smith"});
{
  acknowledged: true,
  insertedId: ObjectId('67082c4f26667e5b2086b01e')
}
Company> db.customers.insertOne({first_name:"Alicia", last_name:"Zelaya"});
{
  acknowledged: true,
  insertedId: ObjectId('67082c6626667e5b2086b01f')
}
Company> db.customers.insertOne({first_name:"Jennifer", last_name:"Wallace"});
{
  acknowledged: true,
  insertedId: ObjectId('67082c7626667e5b2086b020')
}
```

```
Company> db.customers.insertOne({first_name:"Ahmad", last_name:"Jabbar"});
{
  acknowledged: true,
  insertedId: ObjectId('67082c8826667e5b2086b021')
}
Company> db.customers.insertOne({first_name:"James", last_name:"Borg"});
{
  acknowledged: true,
  insertedId: ObjectId('67082c9426667e5b2086b022')
}
```

7. find data in the customers collection ○ db.customers.find(); ○ db.customers.find().pretty();
    - **[Exercise]** Find the data for document where the first_name is Jennifer
        - https://docs.mongodb.com/manual/reference/operator/query-comparison/
        - db.customers.find({first_name:{$eq:"Ahmad"}})
        - Use regex to find partial match db.customers.find({first_name: /Ah/}) ○ Projection to whitelist fields to pass into output
        - db.customers.find({}, {first_name: true})

```
Company> db.customers.find({first_name:{$eq:"Jennifer"}})
[
  {
    _id: ObjectId('67082c7626667e5b2086b020'),
    first_name: 'Jennifer',
    last_name: 'Wallace'
  }
]
```

```
Company> db.customers.find({first_name: /Je/})
[
  {
    _id: ObjectId('67082c7626667e5b2086b020'),
    first_name: 'Jennifer',
    last_name: 'Wallace'
  }
]
```

8. Multiple documents at once using array format ○ db.customers.insert([{first_name:"Sam", last_name:"Smith"} , {first_name:"Jade", last_name:"Smith", gender:"female"}]);

```
Company> db.customers.insert([{first_name:"Sam", last_name:"Smith"}, {first_name:"Jade", last_name:"Smith",gender:"female"}]);
{
  acknowledged: true,
```

9. **[Exercise]** use an array to insert following to a database "petshop" and collection "pets"

```
use petshop
db.pets.insert({name: "Mikey", species: "Gerbil"})
db.pets.insert({name: "Davey Bungooligan", species: "Piranha"})
db.pets.insert({name: "Suzy B", species: "Cat"})
db.pets.insert({name: "Mikey", species: "Hotdog"})
db.pets.insert({name: "Terrence", species: "Sausagedog"})
db.pets.insert({name: "Philomena Jones", species: "Cat"})
```

- Add another piranha called Pete, and a naked mole rat called Henry. ○ Use find to list all the pets. Find the ID of Mikey the Gerbil.
- Use find to find all the gerbils.
- Find all the creatures named Mikey.
- Find all the creatures named Mikey who are gerbils.
- Find all the creatures with the string "dog" in their species

```
petshop> db.pets.insertOne({name: "Mikey", species: "Gerbil"})
{witched to db petshop
  acknowledged: true,
  insertedId: ObjectId('67082ebd26667e5b2086b025')
}
petshop> db.pets.insertMany([{name: "Mikey", species: "Gerbil"}, {name: "Davey Bungooligan", species: "Piranha"}
ame:"Suzy B", species: "Cat"}, {name: "Mikey", species: "Hotdog"}, {name: "Terrence", species: "Sausagedog"}, {n
 "Philomena Jones", species: "Cat"}])
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('67082fb426667e5b2086b026'),
    '1': ObjectId('67082fb426667e5b2086b027'),
    '2': ObjectId('67082fb426667e5b2086b028'),
    '3': ObjectId('67082fb426667e5b2086b029'),
    '4': ObjectId('67082fb426667e5b2086b02a'),
    '5': ObjectId('67082fb426667e5b2086b02b')
  }
}
petshop> db.pets.insertOne({name: "Pete", species: "Piranha"})
{
  acknowledged: true,
  insertedId: ObjectId('67082ffc26667e5b2086b02c')
}
petshop> db.pets.insertOne({name: "Henry", species: "Naked Mole Rat"})
{
  acknowledged: true,
  insertedId: ObjectId('6708300b26667e5b2086b02d')
}
```

```
petshop> db.pets.find({name:{$eq:"Mikey"}});
[
  {
    _id: ObjectId('67082ebd26667e5b2086b025'),
    name: 'Mikey',
    species: 'Gerbil'
  },
  {
    _id: ObjectId('67082fb426667e5b2086b026'),
    name: 'Mikey',
    species: 'Gerbil'
  },
  {
    _id: ObjectId('67082fb426667e5b2086b029'),
    name: 'Mikey',
    species: 'Hotdog'
  }
]
```

```
petshop> db.pets.find({name:{$eq:"Mikey"}, species:{$eq:"Gerbil"}});
[
  {
    _id: ObjectId('67082ebd26667e5b2086b025'),
    name: 'Mikey',
```

```
    _id: ObjectId('67082ebd26667e5b2086b025'),
    name: 'Mikey',
    species: 'Gerbil'
  },
  {
    _id: ObjectId('67082fb426667e5b2086b026'),
    name: 'Mikey',
    species: 'Gerbil'
  }
]
```

```
petshop> db.pets.find({species: /dog/})
[
  {
    _id: ObjectId('67082fb426667e5b2086b029'),
    name: 'Mikey',
    species: 'Hotdog'
  },
  {
    _id: ObjectId('67082fb426667e5b2086b02a'),
    name: 'Terrence',
    species: 'Sausagedog'
  }
]
```

10. Update ○ db.customers.update({first_name:"Sam"}, {first_name:"Sam", last_name:"Smith", gender:"male"})
    - You need to repeat all the fields with their data. Otherwise document will replace by just the fields available in the update statement. Use the $set operator instead.

○ Use the set operator for that

- db.customers.update({first_name:"Sam"}, {$set:{gender:"male"}});
- **[Exercise]** Update all the customers to include gender and age fields.

```
Company> db.customers.update( {first_name: "Sam"}, {$set:{gender:"male"}});
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
```

```
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

```
Company> db.customers.update( {first_name: "Ahmad"}, {$set:{gender:"male"}});
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
Company> db.customers.update( {first_name: "James"}, {$set:{gender:"male"}});
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
Company> db.customers.update( {first_name: "John"}, {$set:{gender:"male"}});
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
Company> db.customers.update( {first_name: "Jon"}, {$set:{gender:"male"}});
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
Company> db.customers.update( {first_name: "Alicia"}, {$set:{gender:"female"}});
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
Company> db.customers.update( {first_name: "Jennifer"}, {$set:{gender:"female"}});
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
```

```
  modifiedCount: 1,
  upsertedCount: 0
}
Company> db.customers.update( {first_name: "Jennifer"}, {$set:{age:22}});
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
Company> db.customers.update( {first_name: "Jon"}, {$set:{age:33}});
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
Company> db.customers.update( {first_name: "John"}, {$set:{age:85}});
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
```

```
  modifiedCount: 1,
  upsertedCount: 0
}
Company> db.customers.update( {first_name: "Alicia"}, {$set:{age:19}});
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
Company> db.customers.update( {first_name: "Ahmad"}, {$set:{age:40}});
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
Company> db.customers.update( {first_name: "James"}, {$set:{age:62}});
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
```

```
Company> db.customers.update( {first_name: "Sam"}, {$set:{age:26}});
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
Company> db.customers.update( {first_name: "Jade"}, {$set:{age:27}});
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
```

- Use inc operator to increment numerical values

- db.customers.update({first_name:"Sam"}, {$set:{age:40}}); db.customers.update({first_name:"Sam"}, {$inc:{age:5}});

```
Company> db.customers.update({first_name:"Sam"}, {$inc:{age:5}});
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

- Use unset to remove a field
  - {$unset: {field1:"", …}}

```
Company> db.customers.update({first_name:"Sam"}, {$unset:{age:""}});
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

  - db.customers.update({first_name:"Sam"}, {$unset:{age:""}}); ○ Use the upsert to insert if the update fails because document is not there
  - db.customers.update({first_name:"May"}, {first_name:"May", last_name:"June"}, {upsert: true});

```
Company> db.customers.update({ first_name: "May" }, {$set:{ first_name: "May", last_name: "June" }}, { upsert: true });
{
  acknowledged: true,
  insertedId: ObjectId('67095a409e8d6661d6533420'),
  matchedCount: 0,
  modifiedCount: 0,
  upsertedCount: 1
}
```

11. Remove ○ db.customers.remove({}) // remove all the documents ○ db.customers.remove({first_name:"Sam"}, {justone: true})
    - ○ justone will delete only first document it finds, otherwise it will delete all

```
Company> db.customers.remove({first_name:"Sam"}, {justone:true})
DeprecationWarning: Collection.remove() is deprecated. Use deleteOne, deleteMany, findOneAndDelete, or bulkWrite.
{ acknowledged: true, deletedCount: 1 }
```

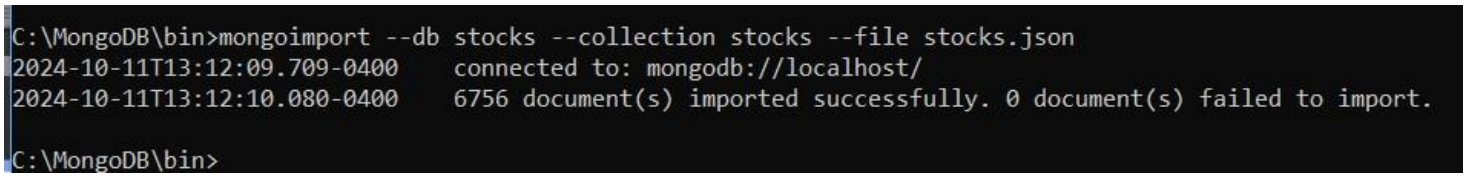
```
Company> db.customers.remove({})
{ acknowledged: true, deletedCount: 8 }
Company> db.customers.find()

Company>
```

12. Import ○ Import json files to the database

- Note that MongoDB installations 4.4 and above, they do not include the mongoimport.exe That is a part of the MongoDB Database Tools install. You can download them here: https://www.mongodb.com/try/download/database-tools?tck=docs_databasetools.

- Exit from the mongo: type "exit" and then type the following in the command line. Your path should still be mongodb\bin
  - **[Exercise]** Use the given https://www.cs.odu.edu/~sampath/courses/data/stocks.json for this exercise
  - mongoimport --db stocks --collection stocks --file stocks.json



Submit the screen capture of all your exercises to Activity 15 piazza thread.