Lecture 6b: MongoDB Store Practical

- Use the MongoDB in the CST labs or you can try these links:
  - https://docs.mongodb.com/manual/tutorial/query-documents/
- Create a Database
  - MongoDB, the first basic step is to have a database and a collection in place
  - The database is used to store all of the collections, and the collection in turn is used to store all of the documents. The documents store the relevant field name and field values

    {“employeeid”:1,
    “employeename”:”smith”}

  - The field names of the document are employeeid and employeename and the field value are 1 and Smith respectively.
  - **USE** command - To create a collection
    - USE studentem; ---- creates a collection called studentem
    - Semicolon is optional
    - The output of the use command will be “switched to db studentem” which means that MongoDB has automatically switched to using the collection studentem.
    - If the collection does not exist, it will create it.
    - If the collection exists, the system will switch to this collection and its documents
- Create a documents
  - **INSERT** command – to insert a single document into a collection, if the collection does not exist, a new collection will be created
  - you don’t have to use the double quotes around the field names (for example, studentid, lname, fname, city)
    - Db.studentem.insert ({“studentid”:1, “lname”:”Smith”,”fname”:”Rose”, “city”:”New York”});
    - Db.studentem.insert ({“studentid”:2, “lname”:”Jones”,”fname”:”Frank”,”city”:”New Haven”});
  - **FIND** command – used to show the documents in the collection
    - Db.studentem.find().forEach(printjson);
    - This command uses JSON (JavaScript Object Notation) to format the output in an easy-to-read manner
    - **NOTE**: The E must be capital in the forEach
  - **VAR** command –
    - Used to insert more than one document into a collection
    - First must create a variable and assign values for the documents
    - Then use the variable along with the INSERT command to insert the stored documents into the collection
    - Var mystudent=[{“studentid”:3, “lname”:”Simone”,”fname”:”Tom”,”city”:”New York”},{ “studentid”:4, “lname”:”Cush”, “fname”:”Ed”,”city”:”Brooklyn”},];

    - Db.studentem.insert(mystudent);
    - Db.studentem.find().forEach(printjson);
- Index
    - By default when inserting a document in a collection, if you do not add a filed name with the _id in the field name, MongoDB automatically adds an Object id field which is used as the index for the document
    - The ObjectId field is used as the primary key for the collection so that each document can be uniquely identified in the collection.
    - If you want to create your own id as the _id of the collection, then you need to explicitly define this while creating the collection
        - Db.studentem.insert({_id:10,
          "studentid":5,"lname":"Kraft","fname":"Joan","city":"Staten Island"})
        - Db.studentem.find().forEach(printjson);
- Update a document
    - Update() is used to update the documents of a collection
    - To update only the documents you want to update, you can add a condition to the update statement so that only documents that meet the condition are updated
    - The basic parameters in the command is 1) the condition for which document will need to meet in order to be updated, and 2) the update needed
        - Db.studentem.update(
          {"studentid":1},
          {$set:{"lname":"Smithe"}});
        - Db.studentem.find().forEach(printjson);
- Update multiple values
    - You can update two values in one document
        - Db.studentem.update(
          {"studentid":2},
          {$set:{"lname":"Jonesey",
          "fname":"Frances"}});
        - Db.studentem.find().forEach(printjson);
- Deleting Document
    - The db.collection.remove() command is used to remove documents from a collection
    - Either all the documents can be removed from a collection or only those which match a specific condition
    - To remove a specific document you must provide a condition
        - Db.studentem.remove({studentid:10});
        - Db.studentem.find().forEach(printjson);
        - This will remove only the documents which have the id as 10
        - You can remove a document based on any condition
        - Db.studentem.remove({"lname":"Smithe"});
    - If you issue the db.studentem.remove() with no condition, all the documents will be removed from the studentem collection
- Count of documents in a collection

- The db.studentem.count() returns the number of documents in the collection studentem

- Query the collection
  - You can query a collection and get all the documents
    - Db.studentem.find({})
    - This is the same as Select * from studentem
- Query based on condition
  - Search by exact value of a stated criteria
    - Db.studentem.find({“lname”:”Jonesey”}).forEach(printjson);
  - Search based on greater than value
    - Db.studentem.find({“studentid”:{$gt:2}}).forEach(printjson);
  - Search based on less than value
    - Db.studentem.find({“studentid”:{$lt:3}}).forEach(printjson);
- Queries with limits
  - The limit modifier is used to limit the number of documents which are returned in the result set for a query
    - Db.studentem.find().limit(1).forEach(printjson);
    - The function will return only one document from the collection studentem
- Queries with order
  - The order of documents to be returned can be specified based on descending order of any key in the collection
    - Db.studentem.find().sort({“studentid”:-1}).forEach(printjson)
    - The function will return all documents in the collection in descending order
  - The order of documents to be returned can be specified based on ascending order of any key in the collection
    - Db.studentem.find().sort({“studentid”:+1}).forEach(printjson)
    - The function will return all documents in the collection in ascending order
- String search
    - MongoDB supports query operations that perform a text search of string content
    - To perform text search, MongoDB uses a text index and the $text operator
    - A collection can only have one text search index, but that index can cover multiple fields
    - Db.studentem.createIndex({“city”:”text”, “lname”:”text”})
    - This makes both the city and the lname values string searchable
    - Lets search the city field for any city that has the word “New”
    - Db.studentem.find({$text:{$search:”New”}})