# MongoDB Step-By-Step

# Introduction

* Document oriented NO-SQL Database
* Schema FREE database
* Composed of Binay JSON
* It has Collection (Group of docs)
* Open Source

# MongoDB vs RDBMS

* Collection versus Table
* Document versus rows
* Fields versus columns
* No restrictions on type of collection as its schema-free

# Setup on Windows

* Download mongodb.msi
* Install in C:mongo
* Route to C:\mongo\bin
* Open command prompt
* Run mongod command (mongo daemon thread)
* Open another terminal then run mongo command which will take us to mongodb terminal
* Run show dbs to see all the databases previously installed on our machines

# Create Database and Drop Database

* **show dbs** : list all databases
* **use mydb** : creates new database if not present and takes us inside this database mydb
* **db** : let us know in which database we currently reside in
* **db.mycollection.insert({“key” : “value”})** : creates new collection mycollection if not present already
* use **mydb** and then **db.dropDatabse()** : will drop the current database

# Create Collection and Drop Collection

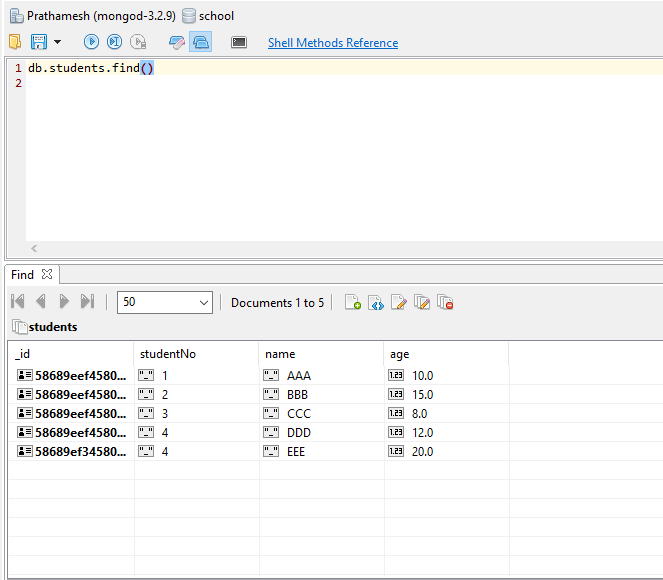
* **use mydb** : creates new database mydb and switched into mydb
* **db.createCollection(“myCollection”, <options/optional>)** : creates new collection as myCollection
* **show collections** : list all collections in db
* **db.mycollection2.insert({“key” : “value”}) :** another method to create new collection as mycollection2
* **show collections :** list both the collections inside db
* **db.myCollection.drop() :** drops the collection myCollection
* **show collections :** would now list only 1 collection and i.e mycollection2

# Insert and Update Query

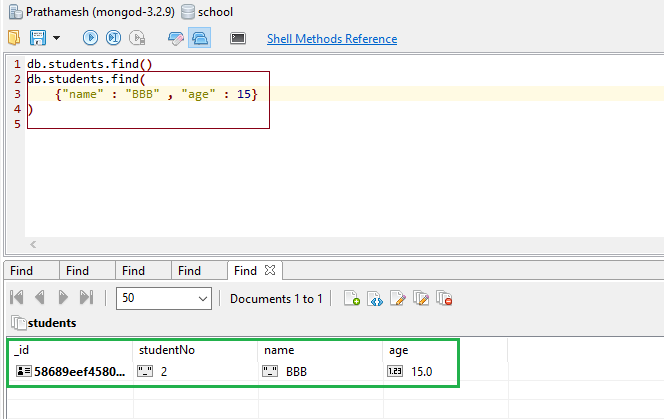
* **use school** : create a new database named as school
* **db.students.insert**([
* {
* "studentNo": "1",
* "name": "AAA",
* "age": 10
* },
* {
* "studentNo": "2",
* "name": "BBB",
* "age": 15
* },
* {
* "studentNo": "3",
* "name": "CCC",
* "age": 8
* },
* {
* "studentNo": "4",
* "name": "DDD",
* "age": 12
* },
* {
* "studentNo": "4",
* "name": "EEE",
* "age": 20
* }
* ])
* **show collections** : would list students collections
* **db.students.find()** : would give collection table/json view
* **db.students.findOne()** : would return first object in students collection
* {
* "studentNo": "1",
* "name": "AAA",
* "age": 10
* }
* **db.students.find({“studentNo” : “2”})** would return
* {
* "studentNo": "2",
* "name": "BBB",
* "age": 15
* },
* **db.students.find({"age" : {$gte : 12}})** : would give all the students whose age is greater than or equal to 12

# Query using AND OR conditions

* db.students.find() : would list all the collection

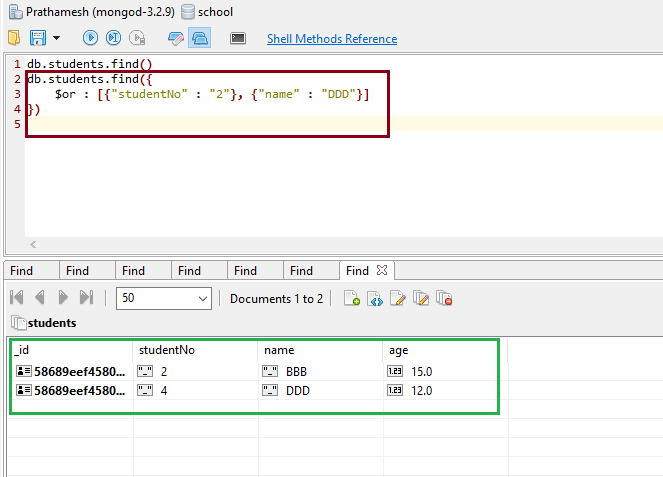


* **AND operation :**
* **db.students.find(**
* **{"name" : "BBB" , "age" : 15}**
* **)** comma in the above example would act as AND operation and can be read as Find a student whose name is BBB AND age is 15



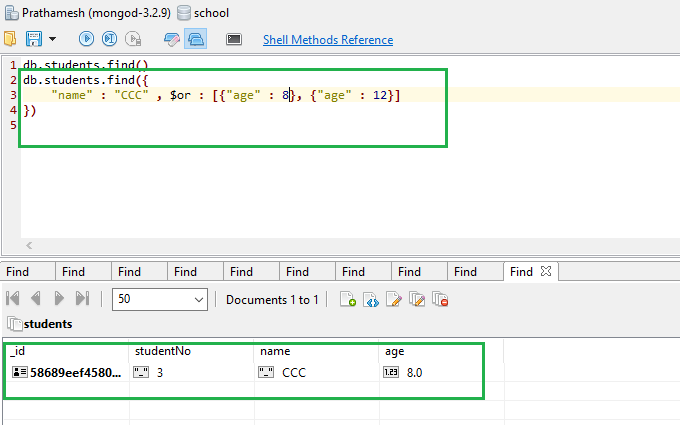
**OR operation:**

* **db.students.find({**
* **$or : [{"studentNo" : "2"}, {"name" : "DDD"}]**
* **})**
* **$or :** used for OR operator and followed by : [{} , {}] list of params to ORed with
* Can be read as get Query for students whose studentNo is 2 OR name is DDD

****

**AND OR operations**

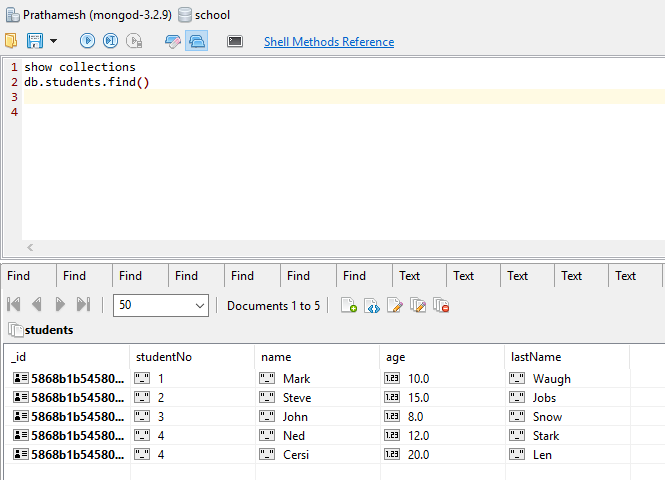
* **Query for student whose name is CCC AND whose age is 8 OR 12**

****

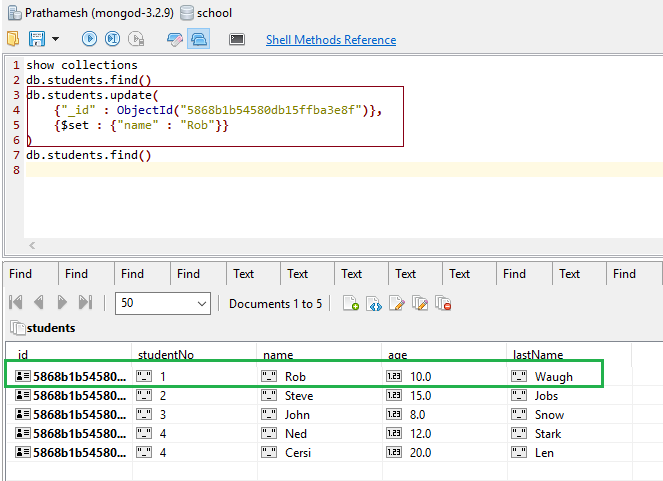
# Update Document

* Update command can be used while edit in particular data

**Before update :**

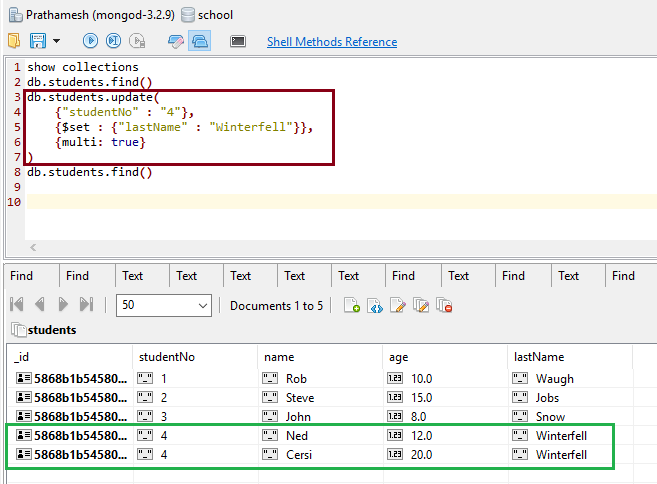


* db.students.update(
* { "\_id" : ObjectId("5868b1b54580db15ffba3e8f")},
* {$set : {"name" : "Rob"}}
* )
* In the above query we use $set to modify/update the record which matches to the id
* Can be read as update the name of student whose name was mark to Rob whose id is 5868b1b54580db15ffba3e8f – this id is auto-generated by mongodb to assign unique id to every record



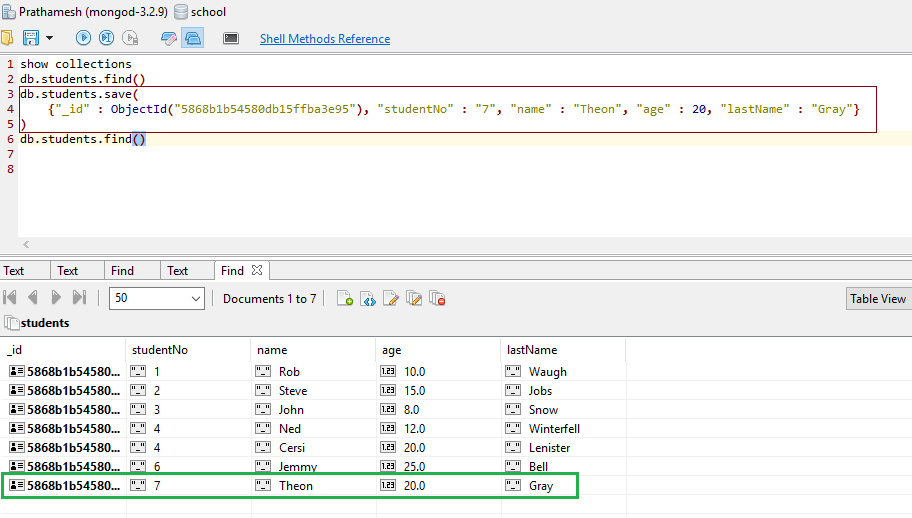
**Update multiple**

* Will update multiple records at a time
* Since last 2 records have same studentNo and we need to edit/update their lastNames to Winterfell, this can be achieved by setting **{multi: true}** and can be viewed as

****

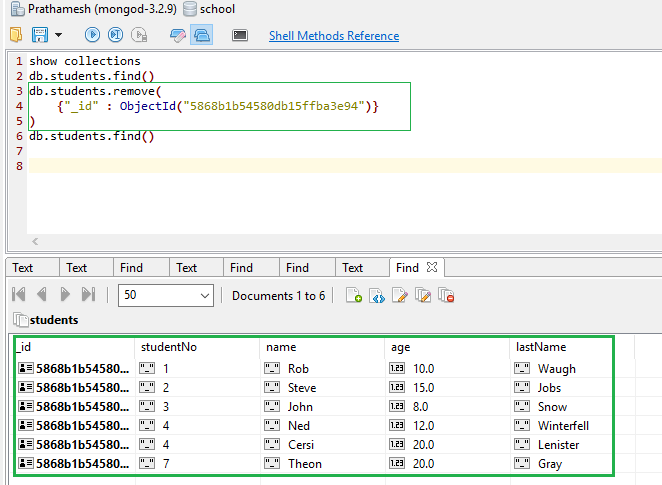
**Create New Record:**

* **db.students.save(**
* **{"\_id" : ObjectId("5868b1b54580db15ffba3e95"), "studentNo" : "7", "name" : "Theon", "age" : 20, "lastName" : "Gray"}**
* **)**
* When we give new id and followed by all the new data, this will create/update collection with new entry

****

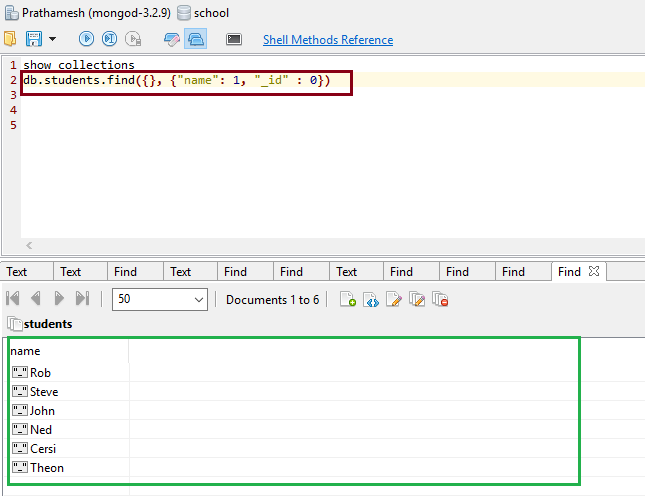
# Delete Document

* db.students.remove(<enter particular parameter id or something>)
* if we do not enter parameters inside remove(), this would remove all the entries and empty the collection
* in this case , db.students.remove(
* {"\_id" : ObjectId("5868b1b54580db15ffba3e94")}
* ) will remove data with id 5868b1b54580db15ffba3e94
* If we want to remove multiple entries with same parameters based on their ages we can specify that with {“age”, 20} and so on



# Mongodb Projection

* Projection means show only necessary data from the document rather than showing the entire document
* db.students.find({}, {"name": 1}), in this case, inside find() keep the first argument as {} and second argument would be {KEY\_From\_document : Boolean }
* Boolean 1 means we want to show
* Boolean 0 means we want to hide
* In this example it would be db.students.find({}, {"name": 1, “\_id” : 0}) and output can be viewed as



# Limit, Skip and Sort collection