Mongo DB:

Terminologies:

RDBMS	Mongo
DB	DB
Table	Collection
Tuple/Row	Document
Attribute/Column	Field
Table Join	Embedded Documents
Primary Key	Primary Key (Default key **_id**) it is provided by MongoDB itself

DataBase Server And Client

RDBMS	Mongo
Mysql	Mongod
Mysql	Mongo

1) NoSQL Database:

1) Non-relational database
2) Does not use tables for storage
3) Provides mechanism for data storage and retrieval other than tabular relationships
4) Generally used to store big-data and real-time web-apps

2) Mongo DB:

1) Def:

1) Cross-platform, document-oriented database
2) Provides:
    1) High Performance
    2) High Availability
    3) Easy Scalability
3) Works on the concept of Collections and Documents

2) **Collections:**
    1) It is a group of mongoldb documents
    2) Eq. Of RDBMS table
    3) Documents within a collection can have different fields but they should be similar or of related purpose
3) **DataTypes:**
    1) String, Integer, Boolean, Double, Min/Max Keys, Arrays, Objects, Null, Symbols, Date
4) **Mongo Commands:**
    1) **DB Related:**
        1) There is no create-db command:
            1) use <<db-name>>:
                1) If that that particular DB is not present then it will create a new DB, otherwise it will return already created DB
                2) It returns the DB and it stores it in ‘db’ variable, now we can use this ‘db’ variable to execute commands inside that database
            2) Show Databases/ show dbs:
                1) List all the databases in the server that has any data in them
            3) Inserting:
                1) Listing Collections:
                    1) Show collections;
                2) Ways of Creating a collection:
                    1) Use createCollection() command
                        1) Syntax: db.createCollection(name, options):
                            1) Name: It is a string type, defining the name of the collection
                            2) options(Optional Field):

Options Field Name	Field Data Type	Description
Capped	Boolean	If it is set to true -> Capped Collection -> It is a fixed size collection that automatically overwrites its oldest entries once the maximum size is reached
AutoIndexID	Boolean	If it is set to true -> automatically create index on ID field. Default value = false
Size	Number	Specifies maximum size(in bytes) for a capped collection. If capped=true then we need to specify this field

Options Field Name	Field Data Type	Description
Max	Number	It specifies the maximum number of documents that are allowed in a capped collection

2) db.<<collection-name>>.insert({}):
    1) db.emp.insert({“name”:”Uma”})
        1) The above will insert the document {“name”:” Uma”} in emp collection. If emp does not exist then it will create a new collection and insert the document in it, otherwise it will insert the document in the already created emp collection
4) Dropping Database:
    1) db.dropDatabase()
5) Checking which Database we are using:
    1) db

2) Collections:

1) show collections
2) db.createCollection(“<<collection-name”,option)
3) db.<<collection-name>>.insert(<<document>>)
4) db.<<collection-name>>.drop() : it will drop that particular collection

3) Document Related:

1) db.<<collection-name>>.find() -> fetch all the data from the respective collection
2) Inserting documents:
    1) Single:
        1) db.<<collection-name>>.insert(<<document>>)
    2) Multiple:
        1) db.<<collection-name>>.insertMany()
        2) db.<<collection-name>>.insert(<<array/list>>)
3) Making documents more readable:
    1) db.<<collection-name>>.find().pretty()
4) Limiting the Number of documents we want to display:
    1) db.<<collection-name>>.find().limit(<<no-of-records>>)
    2) EG: db.student.find().limit(3)
5) Skipping the documents:
    1) db.<<collection-name>>.find().limit(<<no-of-records>>).skip(<<no=of-records>>)
    2) It is used with find() and limit() methods

3) It will skip the specified number of records and display the limited records after that
4) EG: db.student.find().limit(2).skip(1)

6) Sorting:
1) Db.<<collection-name>.find().sort({KEY:1/-1})
1) 1-> ascending order
2) -1 -> descending order
2) EG: db.student.find().sort({“name":1})

7) Fetching a particular document:
1) db.student.findOne({“FirstName”:”Radhika”}) -> single record
2) db.student.find({“FirstName”:”Radhika”}) -> it return all the matched records

8) Updating a particular field:
1) db.emp.update({“FirstName”:”Uma”},{$set:{“name”:” Vivek”}})
2) If column is not there it will insert otherwise it will update

9) Updating the entire document:
1) db.<<collection-name>>.save(<<document>>)
2) If inside the document “_id” is not passed:
1) Insert New Record
3) Else:
1) Overwrite the data in that _id

10) Deleting the documents:
1) Db.<<collection-name>>.remove(<<criteria>>)
2) Deleting Particular Record:
1) db.emp.remove({“FirstName”:”Raj”}) -> all the matched records will get deleted
3) Deleting all:
1) db.emp.remove({})

4) Projections:

1) The find() method -> it displays all the fields even if we don’t want them to be displayed
2) To limit this we need to set list of fields with value:
1) 1-> we need to show
2) 0-> no-need to show
3) Eg:
1) db.emp.find({},{“name":1,"emp_id":1}) -> In this case _id will be printed
2) db.emp.find({},{“name”:1,”emp_id":1, “_id”:0}) -> we have projected _id -> 0 which implies only name and emp_id will be printed

Assignment:

1) Find the record details with name James
2) display all employees with emp_id<200
3) display all employees with emp_id<=201
4) display all employees with emp_id>201
5) display all employees with emp_id>=201
6) display all employees with emp_id!=201
7) display details of employees with name -> {"Jacob","Maya"}
8) display details of employye whose id is not equals to -> {111,205,191}
9) display detalis of employee whose name is James and course is Java
10) display detalis of employee whose name is James or course is Java
11) display detalis of employee whose name is James nor course is Java
12) display employee with id not greater than 200