1. Datatypes
MongoDB supports many datatypes.
1)String
-This is the most commonly used datatype to store the data.
-String in MongoDB must be UTF-8 valid.
2)Integer
-This type is used to store a numerical value.
-Integer can be 32 bit or 64 bit depending upon your server.
3)Boolean
-This type is used to store a boolean (true/ false) value.
4)Double
-This type is used to store floating point values.
5)Min/ Max keys
-This type is used to compare a value against the lowest and highest BSON elements.
6)Arrays
-This type is used to store arrays or list or multiple values into one key.
7)Timestamp
-ctimestamp.
-This can be handy for recording when a document has been modified or added.
8)Object
-This datatype is used for embedded documents.
9)Null
-This type is used to store a Null value.
10)Symbol
-This datatype is used identically to a string.
-However, it's generally reserved for languages that use a specific symbol type.
11)Date
-This datatype is used to store the current date or time in UNIX time format.
-You can specify your own date time by creating object of Date and passing day, month, year into it.
12)Object ID
-This datatype is used to store the document’s ID.
13)Binary data
-This datatype is used to store binary data.
14)Code
-This datatype is used to store JavaScript code into the document.
15)Regular expression
-This datatype is used to store regular expression.

2. Insert Document
1)The insert() Method
-To insert data into MongoDB collection, you need to use MongoDB's insert() or save() method.
-Syntax
>db.COLLECTION_NAME.insert(document)
-Example
```
>db.mycol.insert({
  _id: ObjectId(7df78ad8902c),
  title: 'MongoDB Overview',
  description: 'MongoDB is no sql database',
  by: 'tutorials point',
  url: 'http://www.tutorialspoint.com',
  tags: ['mongodb', 'database', 'NoSQL'],
  likes: 100
 })
```
-mycol : Collection name.
-If the collection doesn't exist in the database, then MongoDB will create this collection and then insert a document into it.

-In the inserted document, if we don't specify the _id parameter, then MongoDB assigns a unique ObjectId for this document.

2)_id
-is 12 bytes hexadecimal number unique for every document in a collection.
-_id: ObjectId(4 bytes timestamp, 3 bytes machine id, 2 bytes process id, 3 bytes incrementer)

3)To insert multiple documents in a single query, you can pass an array of documents in insert() command.

```
>db.post.insert([
  {
    title: 'MongoDB Overview',
    description: 'MongoDB is no sql database',
    by: 'tutorials point',
    url: 'http://www.tutorialspoint.com',
    tags: ['mongodb', 'database', 'NoSQL'],
    likes: 100
  },

  {
    title: 'NoSQL Database',
    description: 'NoSQL database doesn't have tables',
    by: 'tutorials point',
    url: 'http://www.tutorialspoint.com',
    tags: ['mongodb', 'database', 'NoSQL'],
    likes: 20,
    comments: [
      {
        user:'user1',
        message: 'My first comment',
        dateCreated: new Date(2013,11,10,2,35),
        like: 0
      }
    ]
  }
])
```

4)To insert the document you can use db.post.save(document) also.
-If you don't specify _id in the document then save() method will work same as insert() method.
0If you specify _id then it will replace whole data of document containing _id as specified in save() method.

3. Query Document
1)The find() Method
-To query data from MongoDB collection, you need to use MongoDB's find() method.
-Syntax

```
>db.COLLECTION_NAME.find()
```

-find() method will display all the documents in a non-structured way.

2)The pretty() Method
-To display the results in a formatted way, you can use pretty() method.
-Syntax

```
>db.mycol.find().pretty()
```

-Example

```
>db.mycol.find().pretty()
```

```
    {
      "_id": ObjectId(7df78ad8902c),
      "title": "MongoDB Overview",
      "description": "MongoDB is no sql database",
      "by": "tutorials point",
      "url": "http://www.tutorialspoint.com",
      "tags": ["mongodb", "database", "NoSQL"],
      "likes": "100"
    }
    >
```

3)Apart from find() method, there is findOne() method, that returns only one document.

4)RDBMS Where Clause Equivalents in MongoDB
-To query the document on the basis of some condition, you can use following operations.
-Equality
--{<key>:<value>}
--db.mycol.find({"by":"tutorials point"}).pretty()
--RDBMS -> where by = 'tutorials point'
-Less Than
--{<key>:{$lt:<value>}}
--db.mycol.find({"likes":{$lt:50}}).pretty()
--RDBMS -> where likes < 50
-Less Than Equals
--{<key>:{$lte:<value>}}
--db.mycol.find({"likes":{$lte:50}}).pretty()
--RDBMS -> where likes <= 50
-Greater Than
--{<key>:{$gt:<value>}}
--db.mycol.find({"likes":{$gt:50}}).pretty()
--RDBMS -> where likes > 50
-Greater Than Equals
--{<key>:{$gte:<value>}}
--db.mycol.find({"likes":{$gte:50}}).pretty()
--RDBMS -> where likes >= 50
-Not Equals
--{<key>:{$ne:<value>}}
--db.mycol.find({"likes":{$ne:50}}).pretty()
--RDBMS -> where likes != 50

5)AND in MongoDB
-Syntax
-In the find() method, if you pass multiple keys by separating them by ',' then MongoDB treats it as AND condition.

```
>db.mycol.find(
  {
    $and: [
      {key1: value1}, {key2:value2}
    ]
  }
).pretty()
```

-Example

```
>db.mycol.find({$and:
 [
  {"by":"tutorials point"},
  {"title": "MongoDB Overview"}
```

```
      ]}).pretty()
    {
      "_id": ObjectId(7df78ad8902c),
      "title": "MongoDB Overview",
      "description": "MongoDB is no sql database",
      "by": "tutorials point",
      "url": "http://www.tutorialspoint.com",
      "tags": ["mongodb", "database", "NoSQL"],
      "likes": "100"
    }
  -RDBMS -> ' where by = 'tutorials point' AND title = 'MongoDB Overview' '.

6)OR in MongoDB
  -Syntax
    --To query documents based on the OR condition, you need to use $or keyword.

    >db.mycol.find(
      {
        $or: [
          {key1: value1}, {key2:value2}
        ]
      }
    ).pretty()
  -Example
     >db.mycol.find({
      $or:[
        {"by":"tutorials point"},
        {"title": "MongoDB Overview"}
        ]}).pretty()
     {
       "_id": ObjectId(7df78ad8902c),
       "title": "MongoDB Overview",
       "description": "MongoDB is no sql database",
       "by": "tutorials point",
       "url": "http://www.tutorialspoint.com",
       "tags": ["mongodb", "database", "NoSQL"],
       "likes": "100"
     }
     >

7)Using AND and OR Together
  -Example
  -RDBMS -> 'where likes>10 AND (by = 'tutorials point' OR title = 'MongoDB Overview')'

    >db.mycol.find(
     {
      "likes": {$gt:10},
      $or: [
        {"by": "tutorials point"},
        {"title": "MongoDB Overview"}
           ]}).pretty()
     {
       "_id": ObjectId(7df78ad8902c),
       "title": "MongoDB Overview",
       "description": "MongoDB is no sql database",
       "by": "tutorials point",
       "url": "http://www.tutorialspoint.com",
```

```
        "tags": ["mongodb", "database", "NoSQL"],
        "likes": "100"
      }
      >


4. Update Document
 1)MongoDB's update() and save() methods are used to update document into a collection.
 2)The update() method updates the values in the existing document while the save() method replaces
 the existing document with the document passed in save() method.
 3)MongoDB Update() Method
  -The update() method updates the values in the existing document.
  -Syntax
   >db.COLLECTION_NAME.update(SELECTION_CRITERIA, UPDATED_DATA)
  -Example
   { "_id" : ObjectId(5983548781331adf45ec5), "title":"MongoDB Overview"}
   { "_id" : ObjectId(5983548781331adf45ec6), "title":"NoSQL Overview"}
   { "_id" : ObjectId(5983548781331adf45ec7), "title":"Tutorials Point Overview"}
   Following example will set the new title 'New MongoDB Tutorial' of the documents whose title is
   'MongoDB Overview'.

   >db.mycol.update(
    {'title':'MongoDB Overview'},
    {$set:{'title':'New MongoDB Tutorial'}}
   )
   >db.mycol.find()
   { "_id" : ObjectId(5983548781331adf45ec5), "title":"New MongoDB Tutorial"}
   { "_id" : ObjectId(5983548781331adf45ec6), "title":"NoSQL Overview"}
   { "_id" : ObjectId(5983548781331adf45ec7), "title":"Tutorials Point Overview"}
   >
 4)By default, MongoDB will update only a single document.
  -To update multiple documents, you need to set a parameter 'multi' to true.

  >db.mycol.update(
     {'title':'MongoDB Overview'},
     {$set:{'title':'New MongoDB Tutorial'}},
     {multi:true})
 5)MongoDB Save() Method
  -The save() method replaces the existing document with the new document passed in the save()
  method.
  -Syntax
   >db.COLLECTION_NAME.save({_id:ObjectId(),NEW_DATA})
  -Example
   >db.mycol.save(
     {
       "_id" : ObjectId(5983548781331adf45ec7),
       "title":"Tutorials Point New Topic",
       "by":"Tutorials Point"
     }
   )
   >db.mycol.find()
   { "_id" : ObjectId(5983548781331adf45ec5), "title":"Tutorials Point New Topic",
     "by":"Tutorials Point"}
   { "_id" : ObjectId(5983548781331adf45ec6), "title":"NoSQL Overview"}
   { "_id" : ObjectId(5983548781331adf45ec7), "title":"Tutorials Point Overview"}
   >

```

5. Delete Document
 1)The remove() Method
  -MongoDB's remove() method is used to remove a document from the collection.
  -remove() method accepts two parameters.
  -One is deletion criteria and second is justOne flag.
   --deletion criteria
    ---(Optional) deletion criteria according to documents will be removed.
   --justOne
    ---(Optional) if set to true or 1, then remove only one document.
  -Syntax
   >db.COLLECTION_NAME.remove(DELLETION_CRITTERIA)
  -Example
   { "_id" : ObjectId(5983548781331adf45ec5), "title":"MongoDB Overview"}
   { "_id" : ObjectId(5983548781331adf45ec6), "title":"NoSQL Overview"}
   { "_id" : ObjectId(5983548781331adf45ec7), "title":"Tutorials Point Overview"}
   Following example will remove all the documents whose title is 'MongoDB Overview'.

   >db.mycol.remove({'title':'MongoDB Overview'})
   >db.mycol.find()
   { "_id" : ObjectId(5983548781331adf45ec6), "title":"NoSQL Overview"}
   { "_id" : ObjectId(5983548781331adf45ec7), "title":"Tutorials Point Overview"}
   >

 2)Remove Only One
  -If there are multiple records and you want to delete only the first record, then set justOne parameter in remove() method.

  >db.COLLECTION_NAME.remove(DELETION_CRITERIA,1)

 3)Remove All Documents
  -If you don't specify deletion criteria, then MongoDB will delete whole documents from the collection.
  -This is equivalent of SQL's truncate command.

  >db.mycol.remove()
  >db.mycol.find()
  >