# MongoDB manual

## 1. Creating database:/ Switch to current database

**use** database name

```
> use studentdb;
switched to db studentdb
```

## 2. To confirm the existence of database/To report the name of the current database

**db**

```
> db;
studentdb
```

## 3. To get list of all databases

**show dbs**

```
> show dbs;
local  0.000GB
```

**Note: The newly created data base is not listed in the result. The reason is that the database needs to have at least one document to shown up in the list.**

## 4. To drop database

**db.dropDatabase();**

```
> db.dropDatabase();
{ "ok" : 1 }
```

**To drop the datadase first ensure that your currently placed in that database using use command.**

## 5. To Display the version of MongoDB server

**db.version();**

```
> db.version();
3.2.10
```

## 6. To display the statistics that refelect the state of a database

**db.stats();**

```
> db.stats()
{
        "db" : "test",
        "collections" : 0,
```

```
        "objects" : 0,
        "avgObjSize" : 0,
        "dataSize" : 0,
        "storageSize" : 0,
        "numExtents" : 0,
        "indexes" : 0,
        "indexSize" : 0,
        "fileSize" : 0,
        "ok" : 1
}
```

**Note: Here all the details are coming Zero because i havent enter any documents and collection in database.**

**7. Help command**

**db.help();**

## MongoDB Query language(CRUD operations)

**Collections**

**Note: make sure your in the required database**

**1. To display the collections in database**

**show collections**

**2. Create collection**

**db.createCollection (“name of the collection”)**

```
> db.createCollection("Person");
{ "ok" : 1 }

> show collections
Person
```

**3. To drop collection**

**db.Name of the collection.drop()**

```
> db.Person.drop();
true
```

**Inserting Documents:**

**db.Collection Name.insert({doc id, fields.....})**

1. **db.student.insert({_id:1,Name:"xxx",GPA:"9.5",Hobbies:"surfing"});**

WriteResult({ "nInserted" : 1 })

**2. To insert documents without id**

**db.student.insert({Name:"ZZZ",GPA:"7.5",Hobbies:"chess"});**

**WriteResult({ "nInserted" : 1 })**

> **db.student.find();**

**{ "_id" : 1, "Name" : "xxx", "GPA" : "9.5", "Hobbies" : "surfing" }**
**{ "_id" : 2, "Name" : "YYY", "GPA" : "8.5", "Hobbies" : "cricket" }**
**{ "_id" : ObjectId("580eff3d9e9cce5d45a2af1b"), "Name" : "ZZZ", "GPA" : "7.5", "Hobbies" : "chess" }**

**3. To display the documents/To check whether document iserted properly**

**db.collectionname.find();**

**db.student.find();**

{ "_id" : 1, "Name" : "xxx", "GPA" : "9.5", "Hobbies" : "surfing" }

**4.Pretty method : To display results in a formatted manner.**

**db.collectionname.find().pretty();**

Inserting 2 documents

**db.student.insert({_id:1,Name:"xxx",GPA:"9.5",Hobbies:"surfing"});**
**db.student.insert({_id:1,Name:"xxx",GPA:"9.5",Hobbies:"surfing"});**

**db.student.find().pretty();**
**{ "_id" : 1, "Name" : "xxx", "GPA" : "9.5", "Hobbies" : "surfing" }**
**{ "_id" : 2, "Name" : "YYY", "GPA" : "8.5", "Hobbies" : "cricket" }**

**4. Update Method**

Update method it checks whether the docment with id is already exists is so it will update the required field otherwise it will insert a new document with new id.

Here UPSERT method is very imporant:

If UPSERT =false (default value) then new record are not inserted
If UPSERT =true then new records are inserted

**when using Update method its required to use $set operator to update the fields**

**suppose i will try to upadte the docment with ID3 it si not there in the collection student.**

**{ "_id" : 1, "Name" : "xxx", "GPA" : "9.5", "Hobbies" : "surfing" }**
**{ "_id" : 2, "Name" : "YYY", "GPA" : "8.5", "Hobbies" : "cricket" }**
**{ "_id" : ObjectId("580eff3d9e9cce5d45a2af1b"), "Name" : "ZZZ", "GPA" : "7.5", "Hobbies" : "chess" }**

```
db.student.update({_id:3,Name:"FFF",GPA:"6.5"},{$set:{Hobbies:"reading"}},{upsert:true}
);
```

**here id 3 document is not there so it will try to insert the new document with id:3**

**the output will be:**

WriteResult({ "nMatched" : 0, "nUpserted" : 1, "nModified" : 0, "_id" : 3 })

**Now to verify the document gets updated:**

```
db.student.find({_id:3});
```

{ "_id" : 3, "GPA" : "6.5", "Name" : "FFF", "Hobbies" : "reading" }

Here you can Note Find method . That is conditions can be applied in find method

***Now we will try to update the alreday existing document with new data***

**Changing the id:3 document hobbies as listening music**

```
db.student.update({_id:3},{$set:{Hobbies:"listening music"}},{upsert:true});
```

```
db.student.find({_id:3});
```

**{ "_id" : 3, "GPA" : "6.5", "Name" : "FFF", "Hobbies" : "listening music" }**

**when setting UPSERT:false**

```
db.student.update({_id:4, Name:"GGG",GPA:"7.5"},{$set:{Hobbies:"playing
games"}},{upsert:false});
```

**WriteResult({ "nMatched" : 0, "nUpserted" : 0, "nModified" : 0 })**

**No documents are inserted when setting upsert:false**

**db.student.find();**

{ "_id" : 1, "Name" : "xxx", "GPA" : "9.5", "Hobbies" : "surfing" }
{ "_id" : 2, "Name" : "YYY", "GPA" : "8.5", "Hobbies" : "cricket" }
{ "_id" : ObjectId("580eff3d9e9cce5d45a2af1b"), "Name" : "ZZZ", "GPA" : "7.5", "Hobbies" : "chess" }
{ "_id" : 3, "GPA" : "6.5", "Name" : "FFF", "Hobbies" : "listening music" }

**> db.student.update({_id:4, Name:"GGG",GPA:"7.5"},{$set:{Hobbies:"playing games"}},{upsert:true});**

WriteResult({ "nMatched" : 0, "nUpserted" : 1, "nModified" : 0, "_id" : 4 })

**> db.student.find();**

{ "_id" : 1, "Name" : "xxx", "GPA" : "9.5", "Hobbies" : "surfing" }
{ "_id" : 2, "Name" : "YYY", "GPA" : "8.5", "Hobbies" : "cricket" }
{ "_id" : ObjectId("580eff3d9e9cce5d45a2af1b"), "Name" : "ZZZ", "GPA" : "7.5", "Hobbies" : "chess" }
{ "_id" : 3, "GPA" : "6.5", "Name" : "FFF", "Hobbies" : "listening music" }
{ "_id" : 4, "GPA" : "7.5", "Name" : "GGG", "Hobbies" : "playing games" }

**Trying to insert new field in the Document using Update**

**db.student.update({_id:4, Name:"GGG",GPA:"7.5"},{$set:{Hobbies:"playing games",location:"india"}},{upsert:true});**

WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })

**> db.student.find();**
{ "_id" : 1, "Name" : "xxx", "GPA" : "9.5", "Hobbies" : "surfing" }
{ "_id" : 2, "Name" : "YYY", "GPA" : "8.5", "Hobbies" : "cricket" }
{ "_id" : ObjectId("580eff3d9e9cce5d45a2af1b"), "Name" : "ZZZ", "GPA" : "7.5", "Hobbies" : "chess" }
{ "_id" : 3, "GPA" : "6.5", "Name" : "FFF", "Hobbies" : "listening music" }
{ "_id" : 4, "GPA" : "7.5", "Name" : "GGG", "Hobbies" : "playing games", "location" : "india" }

**5. Save method**

save method will insert a new document if the document with the specified _id does not exist.
If adocument with the specified id exists it replaces the existing document with new one.

Now i am saving the document with id :5 this is not in collection so svae method creates new one.

**db.student.save({_id:5, Name:"jjj",GPA:"8.0"});**

**db.student.find();**

{ "_id" : 1, "Name" : "xxx", "GPA" : "9.5", "Hobbies" : "surfing" }
{ "_id" : 2, "Name" : "YYY", "GPA" : "8.5", "Hobbies" : "cricket" }
{ "_id" : ObjectId("580eff3d9e9cce5d45a2af1b"), "Name" : "ZZZ", "GPA" : "7.5", "Hobbies" : "chess" }
{ "_id" : 3, "GPA" : "6.5", "Name" : "FFF", "Hobbies" : "listening music" }
{ "_id" : 4, "GPA" : "7.5", "Name" : "GGG", "Hobbies" : "playing games", "location" : "india" }
**{ "_id" : 5, "Name" : "jjj", "GPA" : "8.0"}**

**Now trying save already existing document**

**db.student.save({_id:5,Name:"jjj",GPA:"7.5",Hobbies:"cricket"});**

**db.student.find();**

{ "_id" : 1, "Name" : "xxx", "GPA" : "9.5", "Hobbies" : "surfing" }
{ "_id" : ObjectId("580f2464e32bbc76b4975e17"), "Name" : "ZZZ", "GPA" : "7.5", "Hobbies" : "chess" }
{ "_id" : 3, "GPA" : "6.5", "Name" : "FFF", "Hobbies" : "listening music" }
{ "_id" : 4, "Name" : "jjj", "GPA" : "8.0" }
**{ "_id" : 5, "Name" : "jjj", "GPA" : "7.5", "Hobbies" : "cricket" }**

## 6. Removing a document

**db.student.remove({_id:3});**

**WriteResult({ "nRemoved" : 1 })**

**db.student.find();**

{ "_id" : 1, "Name" : "xxx", "GPA" : "9.5", "Hobbies" : "surfing" }
{ "_id" : ObjectId("580f2464e32bbc76b4975e17"), "Name" : "ZZZ", "GPA" : "7.5", "Hobbies" : "chess" }
{ "_id" : 4, "Name" : "jjj", "GPA" : "8.0" }
{ "_id" : 5, "Name" : "jjj", "GPA" : "7.5", "Hobbies" : "cricket" }

## Removing existing field in a document

**For example we are trying to remove the field gpa from a document using update and with unset method**

**db.student.update({_id:5},{$unset:{GPA:"7.5"}});**
**WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })**

**> db.student.find();**
{ "_id" : ObjectId("580f2464e32bbc76b4975e17"), "Name" : "ZZZ", "GPA" : "7.5", "Hobbies" : "chess" }
{ "_id" : 4, "Name" : "jjj", "GPA" : "8.0" }
**{ "_id" : 5, "Name" : "jjj", "Hobbies" : "cricket" }**

**Searching A document baed on condition with Find () method**

find() method is used for condition checking

1. Finding a document with id= 4

**db.student.find({_id:4});**

{ "_id" : 4, "Name" : "jjj", "GPA" : "8.0" }

2. To find the document with hobbies=cricket

**db.student.find({Hobbies:"cricket"});**

{ "_id" : 5, "Name" : "jjj", "Hobbies" : "cricket" }

3. To find Name=zzz and GPA=7.5

**db.student.find({Name:"ZZZ",GPA:"7.5"});**

{ "_id" : ObjectId("580f2464e32bbc76b4975e17"), "Name" : "ZZZ", "GPA" : "7.5", "Hobbies" : "chess" }

**4. To supress the id in the result**

**db.student.find({},{Name:"ZZZ",GPA:"7.5",_id:0});**

**Set _id:0 to supress the id field**
**Set_id:1 to display the id field**

5. To display only names from all documents

**db.RankList.find({},{Name:1,_id:0});** **it supress the id of all documents.**

{ "Name" : "Abilash" }
{ "Name" : "Babu" }
{ "Name" : "Chithra" }
{ "Name" : "sindhu" }
{ "Name" : "Manu" }
{ "Name" : "Dinesh" }
{ "Name" : "sheela" }
{ "Name" : "vignesh" }
{ "Name" : "Kala" }
{ "Name" : "Kiran" }

6. To display only names and GPA form all documents

**db.RankList.find({},{Name:1,GPA:1,_id:0});**

{ "Name" : "Abilash", "GPA" : "6.5" }
{ "Name" : "Babu", "GPA" : "8.5" }
{ "Name" : "Chithra", "GPA" : "7.5" }
{ "Name" : "sindhu", "GPA" : "7.9" }
{ "Name" : "Manu", "GPA" : "9.9" }
{ "Name" : "Dinesh", "GPA" : "5.9" }
{ "Name" : "sheela", "GPA" : "6.7" }
{ "Name" : "vignesh", "GPA" : "7.7" }
{ "Name" : "Kala", "GPA" : "8.1" }
{ "Name" : "Kiran", "GPA" : "8.8" }
>

## Conditional operators

Assume we have the set of documents:

{ "_id" : 1, "Name" : "Abilash", "GPA" : "6.5" }
{ "_id" : 2, "Name" : "Babu", "GPA" : "8.5" }
{ "_id" : 3, "Name" : "Chithra", "GPA" : "7.5" }
{ "_id" : 4, "Name" : "sindhu", "GPA" : "7.9" }
{ "_id" : 5, "Name" : "Manu", "GPA" : "9.9" }
{ "_id" : 6, "Name" : "Dinesh", "GPA" : "5.9" }
{ "_id" : 7, "Name" : "sheela", "GPA" : "6.7" }
{ "_id" : 8, "Name" : "vignesh", "GPA" : "7.7" }
{ "_id" : 9, "Name" : "Kala", "GPA" : "8.1" }
{ "_id" : 10, "Name" : "Kiran", "GPA" : "8.8" }

1. To find the student list whos GPA is greater than 6.7

**db.RankList.find({GPA:{$gt:"6.7"}});**

{ "_id" : 2, "Name" : "Babu", "GPA" : "8.5" }
{ "_id" : 3, "Name" : "Chithra", "GPA" : "7.5" }
{ "_id" : 4, "Name" : "sindhu", "GPA" : "7.9" }
{ "_id" : 5, "Name" : "Manu", "GPA" : "9.9" }
{ "_id" : 8, "Name" : "vignesh", "GPA" : "7.7" }
{ "_id" : 9, "Name" : "Kala", "GPA" : "8.1" }
{ "_id" : 10, "Name" : "Kiran", "GPA" : "8.8" }

2. Student list GPA not equal to 6.7

**db.RankList.find({GPA:{$ne:"6.7"}});**

{ "_id" : 1, "Name" : "Abilash", "GPA" : "6.5" }

{ "_id" : 2, "Name" : "Babu", "GPA" : "8.5" }
{ "_id" : 3, "Name" : "Chithra", "GPA" : "7.5" }
{ "_id" : 4, "Name" : "sindhu", "GPA" : "7.9" }
{ "_id" : 5, "Name" : "Manu", "GPA" : "9.9" }
{ "_id" : 6, "Name" : "Dinesh", "GPA" : "5.9" }
{ "_id" : 8, "Name" : "vignesh", "GPA" : "7.7" }
{ "_id" : 9, "Name" : "Kala", "GPA" : "8.1" }
{ "_id" : 10, "Name" : "Kiran", "GPA" : "8.8" }

**3.Either or operator :$in**

**db.student.find({Hobbies:{$in:['chess','surfing']}});**

It is displays the documents having chess and surfing as hobbies

{ "_id" : ObjectId("580f2464e32bbc76b4975e17"), "Name" : "ZZZ", "GPA" : "7.5", "Hobbies" : "chess" }
{ "_id" : 1, "Name" : "xxx", "GPA" : "9.5", "Hobbies" : "surfing" }

**4.Neither nor operator:$nin**

**db.student.find({Hobbies:{$nin:['chess','surfing']}});**

{ "_id" : 4, "Name" : "jjj", "GPA" : "8.0" }
{ "_id" : 5, "Name" : "jjj", "Hobbies" : "cricket" }

The following conditional opeartors used in MongoDB

| Name | Description |
|---|---|
| $eq | Matches values that are equal to a specified value. |
| $gt | Matches values that are greater than a specified value. |
| $gte | Matches values that are greater than or equal to a specified value. |
| $lt | Matches values that are less than a specified value. |
| $lte | Matches values that are less than or equal to a specified value. |
| $ne | Matches all values that are not equal to a specified value. |
| $in | Matches any of the values specified in an array. |
| $nin | Matches none of the values specified in an array. |

**String Operations**

Assume we have the following documents in the collections:

{ "_id" : 1, "Name" : "xxx", "GPA" : "9.5", "Hobbies" : "surfing" }
{ "_id" : 2, "Name" : "ZZZ", "GPA" : "7.5", "Hobbies" : "chess" }
{ "_id" : 3, "Name" : "yyy", "GPA" : "7.5", "Hobbies" : "cricket" }

1. To find the document where the name starts with "y"

**String begins with "/^char/"**

**db.student.find({Name:/^y/});**

**Or**

**db.student.find({Name:{$regex:"^y"}});**

{ "_id" : 3, "Name" : "yyy", "GPA" : "7.5", "Hobbies" : "cricket" }

2. To find the document where the hobbies ends with "t"

**String ends with "/char$/"**

**db.student.find({Hobbies :/t$/});**

**or**

**db.student.find({Hobbies :{$regex:"t$"}});**

{ "_id" : 3, "Name" : "yyy", "GPA" : "7.5", "Hobbies" : "cricket" }

3. To find the document where the hobbies has an "s" in any postion

**Any poistion /char/ or /.*char.*/ or $regex: "char"**

**db.student.find({Hobbies:/s/});**

{ "_id" : 1, "Name" : "xxx", "GPA" : "9.5", "Hobbies" : "surfing" }
{ "_id" : 2, "Name" : "ZZZ", "GPA" : "7.5", "Hobbies" : "chess" }

**db.student.find({Hobbies:/.*s.*/});**

**db.student.find({Hobbies:{$regex:"s"}});**

## Dealing with Null values;

A Null is a missing or unknown value. When we place aNULL value for afield ,it implies that currently we do not know the value or the value is missing.

We will try to insert the new field location with NULL value in document 3

**db.student. find({$or:[{_id:2},{_id:3}]});**

{ "_id" : 2, "Name" : "ZZZ", "GPA" : "7.5", "Hobbies" : "chess" }
{ "_id" : 3, "Name" : "yyy", "GPA" : "7.5", "Hobbies" : "cricket" }

Updating document 3 with b=null values
**db.student.update({_id:3},{$set:{Location:null}});**

**Displaying the documents with null values.**

**db.student.find({Location:{$eq:null}});**

{ "_id" : 1, "Name" : "xxx", "GPA" : "9.5", "Hobbies" : "surfing" }
{ "_id" : 2, "Name" : "ZZZ", "GPA" : "7.5", "Hobbies" : "chess" }
{ "_id" : 3, "Name" : "yyy", "GPA" : "7.5", "Hobbies" : "cricket", "Location" : null }

This results the document having either null or not values.

**To remove null values**

**db.student.update({_id:3},{$unset:{Loaction:null}});**

WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 0 })

## Count:

To find the number documents in acollection count is used.

**db.student.count();**

3

To find the number of documents contains hobbies as chess

**db.student.count({Hobbies:"chess"});**
2

## Limit

To retrive the first particular  number of documents

1. To retrive the first 2  number of documents

**db.student.find().limit(2);**

{ "_id" : 1, "Name" : "xxx", "GPA" : "9.5", "Hobbies" : "surfing" }
{ "_id" : 2, "Name" : "ZZZ", "GPA" : "7.5", "Hobbies" : "chess" }

**db.student.find({Hobbies:"chess"}).limit(1);**

{ "_id" : 2, "Name" : "ZZZ", "GPA" : "7.5", "Hobbies" : "chess" }

## Sort

To sort the documents in ascending or descending order.

1. To sort in  names ascending order:  **sort({Field name:1})**

**db.student.find().sort({Name:1});**

{ "_id" : 3, "Name" : "Anu", "GPA" : "9.5", "Hobbies" : "Music" }
{ "_id" : 1, "Name" : "Babu", "GPA" : "7.5", "Hobbies" : "chess" }
{ "_id" : 4, "Name" : "Deepak", "GPA" : "7.5", "Hobbies" : "Dance" }
{ "_id" : 2, "Name" : "Suriya", "GPA" : "8.5", "Hobbies" : "chess" }

2.1. To sort in  names Desending order:  **sort({Field name:-1})**

**db.student.find().sort({Name:-1});**

{ "_id" : 2, "Name" : "Suriya", "GPA" : "8.5", "Hobbies" : "chess" }
{ "_id" : 4, "Name" : "Deepak", "GPA" : "7.5", "Hobbies" : "Dance" }
{ "_id" : 1, "Name" : "Babu", "GPA" : "7.5", "Hobbies" : "chess" }
{ "_id" : 3, "Name" : "Anu", "GPA" : "9.5", "Hobbies" : "Music" }

We can sort two field at the same time

**db.student.find().sort({Name:1,GPA:-1}) ( sorting names in ascendind ang gardes in desending order)**

## Skip

To skip first particular number of documents

1. To skip first 2 documents

**db.student.find().skip(2);**

{ "_id" : 3, "Name" : "Anu", "GPA" : "9.5", "Hobbies" : "Music" }
{ "_id" : 4, "Name" : "Deepak", "GPA" : "7.5", "Hobbies" : "Dance" }

2. To sort the documents from and skip first document

**db.student.find().skip(1).sort({Name:1});**

{ "_id" : 1, "Name" : "Babu", "GPA" : "7.5", "Hobbies" : "chess" }
{ "_id" : 4, "Name" : "Deepak", "GPA" : "7.5", "Hobbies" : "Dance" }
{ "_id" : 2, "Name" : "Suriya", "GPA" : "8.5", "Hobbies" : "chess" }

3. To display the last two records from the collection student

**db.student.find().skip(db.student.count()-2);**

4. To retrive third, fourth documents

**db.student.find().skip(1).limit(3);**

## Arrays

To create a collection by the name “Food”. Ecah documents have fruits array

1. To creaye collection Food

**db.createCollection("Food");**

2.Inserting values

```
 db.Food.insert({_id:1,fruits:['banana','apple','cherry']});
WriteResult({ "nInserted" : 1 })
> db.Food.insert({_id:2,fruits:['orange','butterfruit','mango']});
WriteResult({ "nInserted" : 1 })
> db.Food.insert({_id:3,fruits:['pineapple','strawberry','grapes']});
WriteResult({ "nInserted" : 1 })
> db.Food.insert({_id:4,fruits:['banana','strawberry','grapes']});
WriteResult({ "nInserted" : 1 })
> db.Food.insert({_id:5,fruits:['orange','grapes']});
WriteResult({ "nInserted" : 1 })
```

**3. To display**

**db.Food.find();**

{ "_id" : 1, "fruits" : [ "banana", "apple", "cherry" ] }
{ "_id" : 2, "fruits" : [ "orange", "butterfruit", "mango" ] }
{ "_id" : 3, "fruits" : [ "pineapple", "strawberry", "grapes" ] }
{ "_id" : 4, "fruits" : [ "banana", "strawberry", "grapes" ] }
{ "_id" : 5, "fruits" : [ "orange", "grapes" ] }

Array opeartions

1. To find documents contains fruits banana,apple andc herry

**db.Food.find({fruits:['banana', 'apple', 'cherry' ] });**

2. To find document which has fruit banana **as an element**

**db.Food.find({fruits:'banana'});**

3. To find documents those have the fruits array having"garpes" in the **first index postion**

**db.Food.find({'fruits.1':'grapes'}); (index starts with 0)**

**4.**To find documents those have the fruits array having"garpes" in the second **index postion**

**db.Food.find({'fruits.2':'grapes'});**

5. **size** of array (to find the document having the fruit arry size as 2)

**db.Food.find({"fruits":{$size:2}});**

6. To display the first two elemnts of the array from document 1 using **Slice**

**db.Food.find({_id:1},{"fruits":{$slice:2}});**

7. To find documents which have elements"orange" and "grapes" in the array fruit.**($all)**

**db.Food.find({fruits:{$all:["orange","grapes"]}});**

8. To dipslay only two elements from the arry starting with 0 th index poistion from document1

**db.Food.find({_id:1},{"fruits":{$slice:[0,2]}});**

9. To dipslay only two elements from the arry starting with 1 st index poistion from document1

**db.Food.find({_id:1},{"fruits":{$slice:[1,2]}});**

**Update on arrays**

we have collection food with the following documents;

{ "_id" : 1, "fruits" : [ "banana", "apple", "cherry" ] }
{ "_id" : 2, "fruits" : [ "orange", "butterfruit", "mango" ] }
{ "_id" : 3, "fruits" : [ "pineapple", "strawberry", "grapes" ] }
{ "_id" : 4, "fruits" : [ "banana", "strawberry", "grapes" ] }
{ "_id" : 5, "fruits" : [ "orange", "grapes" ] }

1. To **update** the document with id:4 and **replace** the element present in the **1 st index position** of the fruits array with apple

**db.Food.update({_id:4},{$set:{'fruits.1':'apple'}});**

2. To updtae the document id:1 and replace the element “apple “ with “An apple”

**db.Food.update({_id:1,'fruits':'apple'},{$set:{'fruits.$':'An apple'}});**

3. To upadte the document with _id:2 and **push a new key value pair in the fruit array**

**db.Food.update({_id:2},{$push:{price:{orange:60,butterfruit:200,mango:120}}});**

**db.Food.find({_id:2});**

{ "_id" : 2, "fruits" : [ "orange", "butterfruit", "mango" ], "price" : [ { "orange" : 60, "butterfruit" : 200, "mango" : 120 } ] }

4. To **add an element** in array: **($addToSet)**

**db.Food.update({_id:4},{$addToSet:{fruits:"orange"}});**

This will add an new arry element in an dic 4.

5. To **pop an existing element** from an array **($pop)**

**db.food.update({_id:4},{$pop:{fruits:1}});**

**This will remove the element from arry poistioned in 1 st index.**

6. Poping element form the beginning of the array

**db.food.update({_id:4},{$pop:{fruits:-1}});**

## Aggregate functions

1. creating a collection “customer”

**db.createCollection("customer");**

2. Inserting values

**db.customer.insert([{Custid:"C123",AccBal:500,AccType:"S"},{Custid:"C123",AccBal:900,AccType:"S"},{Custid:"C111",AccBal:1200,AccType:"S"},{Custid:"C123",AccBal:1500,AccType:"C"}]);**

3. to display

**db.customer.find();**

{ "_id" : ObjectId("581840d812ad3161e8f54738"), "Custid" : "C123", "AccBal" : 500, "AccType" : "S" }
{ "_id" : ObjectId("581840d812ad3161e8f54739"), "Custid" : "C123", "AccBal" : 900, "AccType" : "S" }
{ "_id" : ObjectId("581840d812ad3161e8f5473a"), "Custid" : "C111", "AccBal" : 1200, "AccType" : "S" }
{ "_id" : ObjectId("581840d812ad3161e8f5473b"), "Custid" : "C123", "AccBal" : 1500, "AccType" : "C" }

4. to group based on id and find the sum of total balance

**db.customer.aggregate({$group:{_id:"$Custid",TotBal:{$sum:"$AccBal"}}});**

{ "_id" : "C111", "TotBal" : 1200 }
{ "_id" : "C123", "TotBal" : 2900 }

5. To filter on AccType:"S" and then groupit based on Custid and then compute sum of Accbal .

**db.customer.aggregate({$match:{AccType:"S"}},{$group:{_id:"$Custid",TotBal:{$sum:"$AccBal"}}});**

{ "_id" : "C111", "TotBal" : 1200 }
{ "_id" : "C123", "TotBal" : 1400 }

6.To filter on AccType:"S" and then groupit based on Custid and then compute sum of Accbal and filter those documents where in the "TotBal" is graeter than 1200.

**db.customer.aggregate({$match:{AccType:"S"}},{$group:{_id:"$Custid",TotBal:{$sum:"$AccBal"}}},{$match:{TotBal:{$gt:1200}}});**

{ "_id" : "C123", "TotBal" : 1400 }

7. To group id and compute average salary of accBal for each type;

**db.customer.aggregate({$group:{_id:"$Custid",TotBal:{$avg:"$AccBal"}}});**

{ "_id" : "C111", "TotBal" : 1200 }
{ "_id" : "C123", "TotBal" : 966.6666666666666 }

8. To group id and determine the amx and min accbal for each type;

**db.customer.aggregate({$group:{_id:"$Custid",TotBal:{$max:"$AccBal"}}});**
{ "_id" : "C111", "TotBal" : 1200 }
{ "_id" : "C123", "TotBal" : 1500 }
**> db.customer.aggregate({$group:{_id:"$Custid",TotBal:{$min:"$AccBal"}}});**
{ "_id" : "C111", "TotBal" : 1200 }
{ "_id" : "C123", "TotBal" : 500 }
>

## MapReduce

**To compute the total accBal of each AccType**

1. map function

**var map = function(){ emit(this.Custid,this.AccBal);}**

2. reduce function

**var reduce = function(key,values){ return Array.sum(values);}**

3. To execute mapreduce

**db.customer.mapReduce(map,reduce,{out:"CustomerTotal",query:{AccType:"S"}});**


```
{
        "result" : "CustomerTotal",
        "timeMillis" : 436,
        "counts" : {
          "input" : 3,
          "emit" : 3,
          "reduce" : 1,
          "output" : 2
        },
        "ok" : 1
}
```


4.To display the result:

**db.CustomerTotal.find();**


```
{ "_id" : "C111", "value" : 1200 }
{ "_id" : "C123", "value" : 1400 }
```


## Java script programming

**To execute jav ascripts in mongoDB we can use the method system.js**

**1. To find the java scripts in machine**

**> db.system.js.find();**

**when you execute this command it displays the prompt only it means there is no js program in machine.**

**2. To add java script programming for factorial**

**db.system.js.insert({_id:"fact",value:function(n) {if(n==1) return 1;  else return n*fact(n-1);}});**

**WriteResult({ "nInserted" : 1 })**

**3. To verfiy the JS program is added or not**

**db.system.js.find();**

**{ "_id" : "fact", "value" : function (n) {if(n==1) return 1;  else return n*fact(n-1);} }**

**4. To execute the factorial : eval function is used**

**> db.eval("fact(3)");**

**WARNING: db.eval is deprecated**

**6**

## Cursors in MongoDB

1. create a collection Alphabets

**> db.createCollection("Alphabets");**

**{ "ok" : 1 }**

2. Inserting all 26 letters as docments

db.Alphabets.insert({_id:1,alpha:"a"});
WriteResult({ "nInserted" : 1 })

db.Alphabets.insert({_id:2,alpha:"b"});
WriteResult({ "nInserted" : 1 })

> db.Alphabets.insert({_id:3,alpha:"c"});
WriteResult({ "nInserted" : 1 })

> db.Alphabets.insert({_id:4,alpha:"d"});
WriteResult({ "nInserted" : 1 })

> db.Alphabets.insert({_id:5,alpha:"e"});
WriteResult({ "nInserted" : 1 })

> db.Alphabets.find();

**{ "_id" : 1, "alpha" : "a" }**
**{ "_id" : 2, "alpha" : "b" }**
**{ "_id" : 3, "alpha" : "c" }**
**{ "_id" : 4, "alpha" : "d" }**
**{ "_id" : 5, "alpha" : "e" }**

**I inserted only 5 characters.**

**Find is the primary method to read documents.**

**When using cursor it iterates the read opeartion upto 20 documents by default.**

3. Creating a cursor

**var myc = db.Alphabets.find();**
**> myc;**
**{ "_id" : 1, "alpha" : "a" }**
**{ "_id" : 2, "alpha" : "b" }**
**{ "_id" : 3, "alpha" : "c" }**
**{ "_id" : 4, "alpha" : "d" }**
**{ "_id" : 5, "alpha" : "e" }**

**Manual cursors:**

**Using hasNext () and next() methods**

**> var myc = db.Alphabets.find({});**

**> while(myc.hasNext()){ var myrec=myc.next();**
**... print("the alphabet is :" + myrec.alpha);}**

**the alphabet is :a**
**the alphabet is :b**
**the alphabet is :c**
**the alphabet is :d**
**the alphabet is :e**

**Using forEach loop**

**> var myc = db.Alphabets.find({});**
**> var myrec;**
**> myc.forEach(function(myrec) {**
**... print("The alphabet is: " + myrec.alpha);});**

**The alphabet is: a**
**The alphabet is: b**
**The alphabet is: c**

**The alphabet is: d**
**The alphabet is: e**

## Indexes

**1. Creating acollection called “books”**

```
> db.createCollection("books");
{ "ok" : 1 }
```

**2. Inserting Documents**

```
> db.books.insert({_id:6,Type:"ML",BName:"ML for Hackers",Qty:25,price:400});
WriteResult({ "nInserted" : 1 })

> db.books.insert({_id:7,Type:"webmining",BName:"Mining social data",Qty:15,price:500});
WriteResult({ "nInserted" : 1 })

> db.books.insert({_id:8,Type:"Programming",BName:"Python for data
analysis",Qty:25,price:700});
WriteResult({ "nInserted" : 1 })

> db.books.insert({_id:9,Type:"Visualization",BName:"Visualizing Data",Qty:25,price:200});
WriteResult({ "nInserted" : 1 })

> db.books.insert({_id:10,Type:"ML",BName:"ML for Bigdata",Qty:25,price:600});
WriteResult({ "nInserted" : 1 })
```

**3. Displaying documents**

```
> db.books.find().pretty();
{
        "_id" : 6,
        "Type" : "ML",
        "BName" : "ML for Hackers",
        "Qty" : 25,
        "price" : 400
}
{
        "_id" : 7,
        "Type" : "webmining",
        "BName" : "Mining social data",
        "Qty" : 15,
        "price" : 500
}
{
        "_id" : 8,
        "Type" : "Programming",
        "BName" : "Python for data analysis",
        "Qty" : 25,
        "price" : 700
```

```
}
{
        "_id" : 9,
        "Type" : "Visualization",
        "BName" : "Visualizing Data",
        "Qty" : 25,
        "price" : 200
}
{
        "_id" : 10,
        "Type" : "ML",
        "BName" : "ML for Bigdata",
        "Qty" : 25,
        "price" : 600
}
```

**4. Creating an index for "Type" field in the books collection using ensureIndex method**

```
> db.books.ensureIndex({"Type":1});

{
        "createdCollectionAutomatically" : false,
        "numIndexesBefore" : 1,
        "numIndexesAfter" : 2,
        "ok" : 1
}
```

**5. Check the status of index (name and number)**

```
> db.books.stats();

{ "ns" : "cursorexample.books",
        "count" : 5,
        "size" : 468,
        "avgObjSize" : 93,
        "storageSize" : 36864,
        "capped" : false,
        "wiredTiger" : {
          "metadata" : {
            "formatVersion" : 1
          },

..................      },
        "nindexes" : 2,
        "totalIndexSize" : 53248,
        "indexSizes" : {
          "_id_" : 36864,
          "Type_1" : 16384
        },
        "ok" : 1
}
```

**6. Get the list of all indexes on the books collection using getIndexes() method**

```
> db.books.getIndexes();
[
        {
          "v" : 1,
          "key" : {
           "_id" : 1
          },
          "name" : "_id_",
          "ns" : "cursorexample.books"
        },
        {
          "v" : 1,
          "key" : {
           "Type" : 1
          },
          "name" : "Type_1",
          "ns" : "cursorexample.books"
        }
]
```

**7. To use the index (using hint method)**

```
> db.books.find({"Type":"ML"}).pretty().hint({"Type":1});
{
        "_id" : 6,
        "Type" : "ML",
        "BName" : "ML for Hackers",
        "Qty" : 25,
        "price" : 400
}
{
        "_id" : 10,
        "Type" : "ML",
        "BName" : "ML for Bigdata",
        "Qty" : 25,
        "price" : 600
}
```

**8. Check the explain plan of index**

```
> db.books.find({"Type":"ML"}).pretty().hint({"Type":1}).explain();


{
        "queryPlanner" : {
          "plannerVersion" : 1,
          "namespace" : "cursorexample.books",
          "indexFilterSet" : false,
          "parsedQuery" : {
           "Type" : {
               "$eq" : "ML"
           }
```

```
                },
                "winningPlan" : {
                 "stage" : "FETCH",
                 "inputStage" : {
                     "stage" : "IXSCAN",
                     "keyPattern" : {
                             "Type" : 1
                     },
                     "indexName" : "Type_1",
                     "isMultiKey" : false,
                     "isUnique" : false,
                     "isSparse" : false,
                     "isPartial" : false,
                     "indexVersion" : 1,
                     "direction" : "forward",
                     "indexBounds" : {
                             "Type" : [
                                     "[\"ML\", \"ML\"]"
                             ]
                     }
                 }
                },
                "rejectedPlans" : [ ]
               },
               "serverInfo" : {
                "host" : "200809JUNC0287-A",
                "port" : 27017,
                "version" : "3.2.10",
                "gitVersion" : "79d9b3ab5ce20f51c272b4411202710a082d0317"
               },
               "ok" : 1
}
```


## MongoImport and MongoExport

**To import CSV file as JSON document**

**1. In a Command Prompt create a .CSV file**

**baskar@200809JUNC0287-A:~$ cat > Namelist.csv**

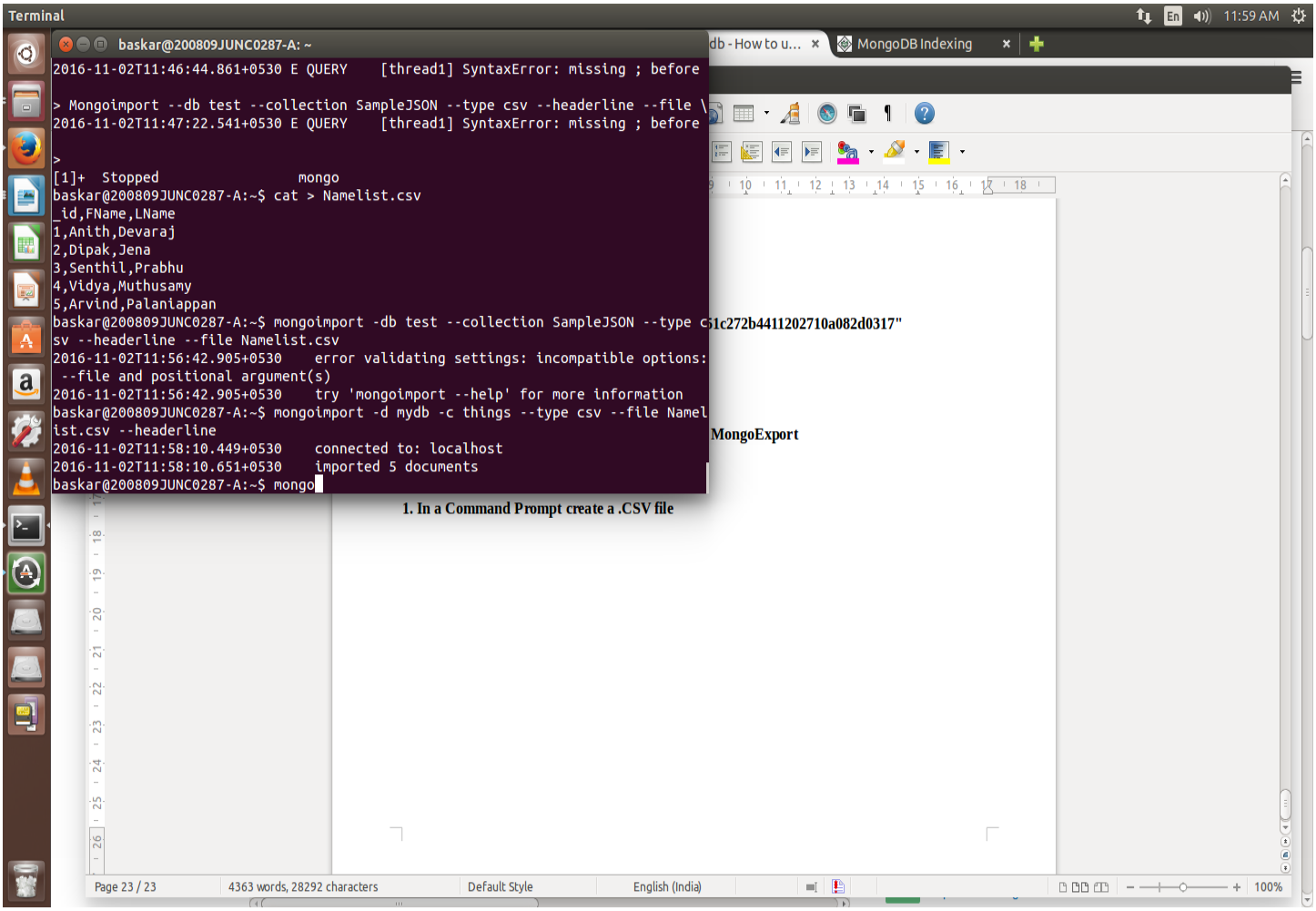
```
_id,FName,LName
1,Anith,Devaraj
2,Dipak,Jena
3,Senthil,Prabhu
4,Vidya,Muthusamy
5,Arvind,Palaniappan
```

**Enter ctrl d to exit from categorization**

**baskar@200809JUNC0287-A:~$ mongoimport -d mydb -c things --type csv --file Namelist.csv –headerline**

**If the connection estabished properly you will get the following output**

**2016-11-02T11:58:10.449+0530 connected to: localhost**
**2016-11-02T11:58:10.651+0530 imported 5 documents**




**To confirm the output**

**Start mongo DB**

**> use mydb**
**switched to db mydb**

**> show collections;**
**things**

**> db.things.find();**
**{ "_id" : 2, "FName" : "Dipak", "LName" : "Jena" }**
**{ "_id" : 3, "FName" : "Senthil", "LName" : "Prabhu" }**
**{ "_id" : 4, "FName" : "Vidya", "LName" : "Muthusamy" }**
**{ "_id" : 5, "FName" : "Arvind", "LName" : "Palaniappan" }**

**{ "_id" : 1, "FName" : "Anith", "LName" : "Devaraj" }**
**>**

**MongoExport**

**Export JSON document into csv file**

**we have things collections in mydb**

**> use mydb;**
**switched to db mydb**

**> show collections;**
**things**

**> db.things.find();**
**{ "_id" : 2, "FName" : "Dipak", "LName" : "Jena" }**
**{ "_id" : 3, "FName" : "Senthil", "LName" : "Prabhu" }**
**{ "_id" : 4, "FName" : "Vidya", "LName" : "Muthusamy" }**
**{ "_id" : 5, "FName" : "Arvind", "LName" : "Palaniappan" }**
**{ "_id" : 1, "FName" : "Anith", "LName" : "Devaraj" }**

**Now this things JSON document is exported to output.csv file.**

**In command prompt**

**baskar@200809JUNC0287-A:~$ mongoexport --csv -d mydb -c things -f "id","FName","LName" -o output.csv**

**To confirm**

**Cat output.csv**

**Note:**

**In Both Import and Export**

| | |
|---|---|
| **-d :** | **database name** |
| **-c :** | **Collection name** |
| **-type:** | **file type** |
| **-f:** | **Fields** |
| **-o** | **output** |