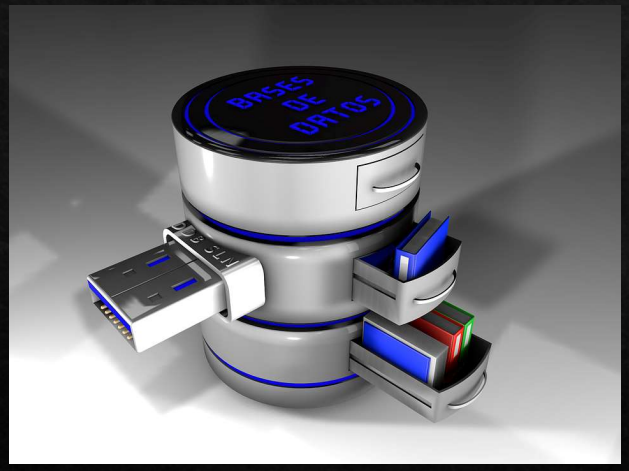
DOCUMENT DATABASES

Basic Data Storage:

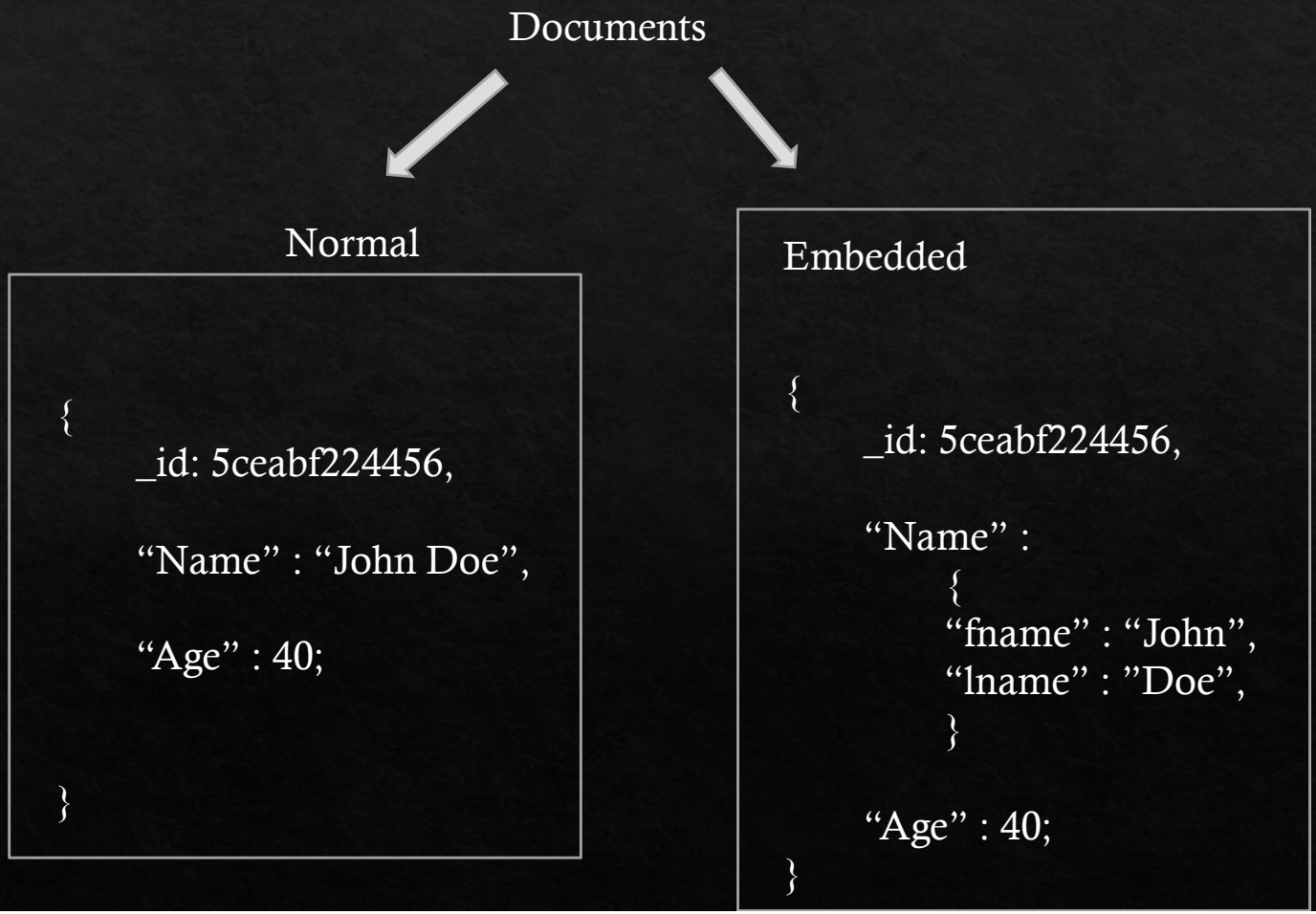
Database, that contains
All collections.



Collection that contains
Set of documents

Document
That contains
Data.

Types Of Documents in document databases:

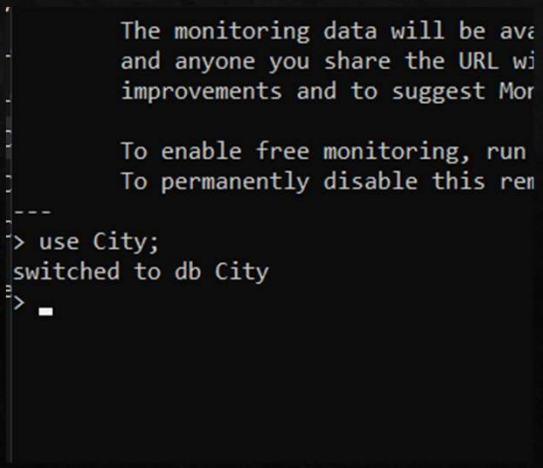


TO Create a database:

- One of the most popular databases are mongo db.
- Syntax to create a database:

  Use <database_name>

- The above syntax switches to the database user specifies if already created, and if not then creates a new one.



Create collection in a database:

- To create a collection we use Syntax:

db.createCollection(name)

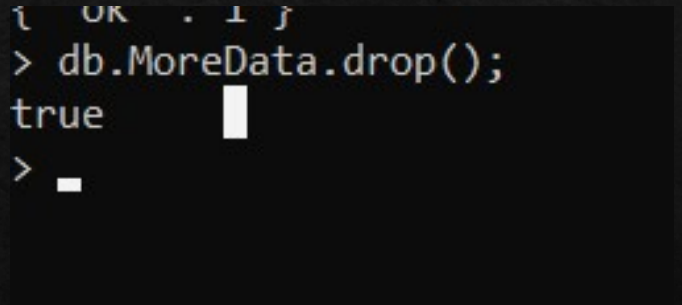
```
> db.createCollection("MoreData");
{ "ok" : 1 }
```

Drop Collection:

- To remove a collection from database we use following syntax:

  db.collection_name.drop();

This will completely remove a collection from the database as follows:



As seen here this returns
True if collection removed
False if any error occurs.

INSERTION of a Document:

- To insert document a simple insert method is used as follows:

```
db.Collection_name.insert({
                        //the data in key : value, format,
});
```

In this case if the collection doesn't exists then mongo will create it automatically.

How to actually insert document:

```
Db.Citizen.insert({
        "_id" : ObjectId("5ff99c886da1c8983c52cfbf"),
        "username" : "Grahm",
        "Basic" : {
                "name" : "Grahm De Villeres",
                "Age" : 82,
                "permanantaddress" : "Goa",
                "Occupation" : "Retired Army"
        },
        "Service" : {
                "Saddress" : "N.A",
                "Post" : "Batallion Cheif",
                "Salary" : "500000"
        },
        "AdharInfo" : {
                "number" : "16544646"
        },
        "PanInifo" : {
                "number" : "HBDASFG454"
        }
});
```

How to View a Document:

- To view a document there is a method called find() in mongo, syntax for that is:

  db.<collection_name>.find();

Above syntax finds all the documents in that database→collection.

How to View particular Document:

db.<collection_name>.find({“key” : “Value”});

Output Of Find method:

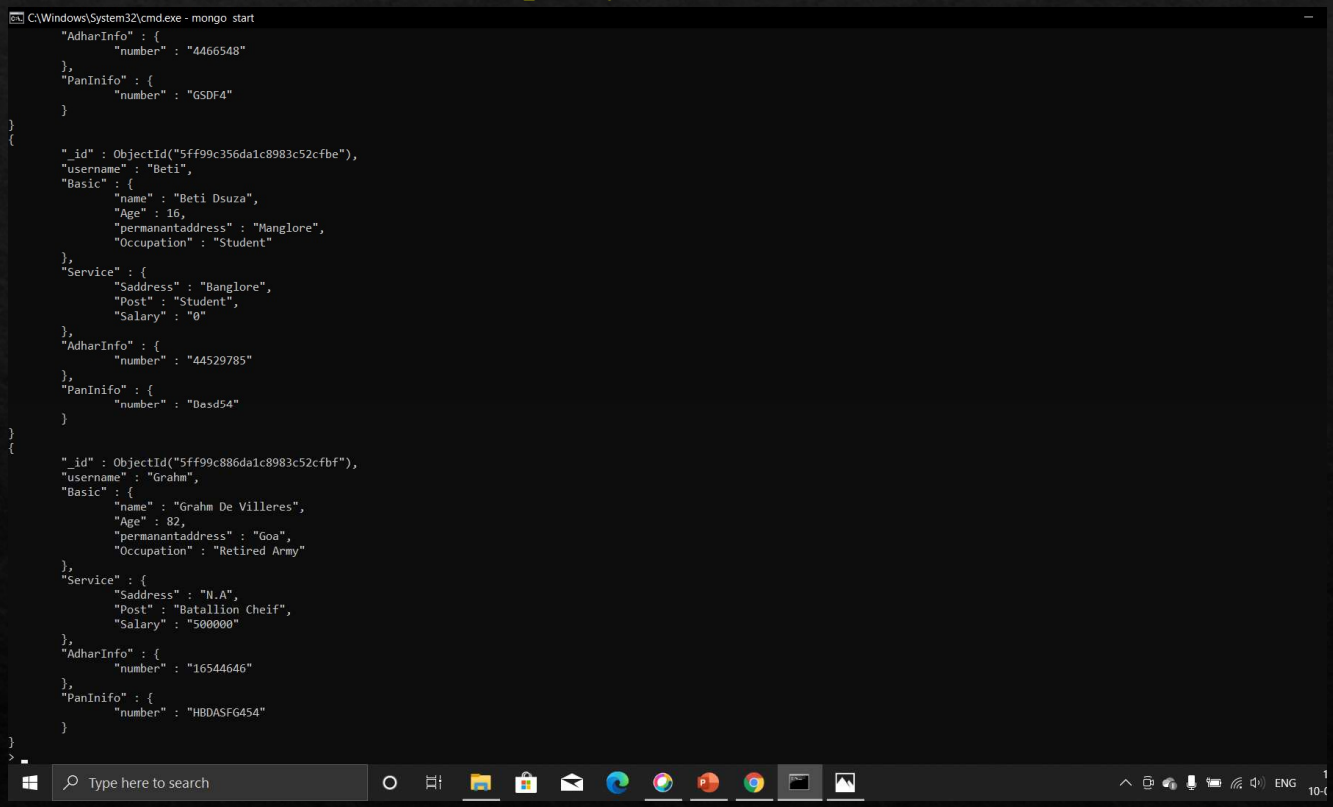
```
}
> db.Citizen.find();
{ "_id" : ObjectId("5ff949f6aa642cd039e5156b"), "username" : "Hanny", "Basic" : { "name" : "Hanny McManhon", "Age" : 33, "permanantaddress" : "Virginia", "Occupation" : "Bussiness" }, "Service" : { "Saddress" :
"India", "Post" : "CEO", "Salary" : "90000120" }, "AdharInfo" : { "number" : "7788996655" }, "PanInifo" : { "number" : "ABCD78C" } }
{ "_id" : ObjectId("5ff9989f6da1c8983c52cfba"), "username" : "Manny", "Basic" : { "name" : "Manny Patricia", "Age" : 31, "permanantaddress" : "Palo Alto", "Occupation" : "SDE" }, "Service" : { "Saddress" : "Cali
fornia", "Post" : "Software Developer", "Salary" : "890000" }, "AdharInfo" : { "number" : "78945444" }, "PanInifo" : { "number" : "ABCD778a" } }
{ "_id" : ObjectId("5ff998e06da1c8983c52cfbb"), "username" : "Panny", "Basic" : { "name" : "Siraj ", "Age" : 23, "permanantaddress" : "Aurangabad", "Occupation" : "Cricketer" }, "Service" : { "Saddress" : "N.A",
 "Post" : "Bowler", "Salary" : "415111000" }, "AdharInfo" : { "number" : "45628791" }, "PanInifo" : { "number" : "Pamsd4455" } }
{ "_id" : ObjectId("5ff999466da1c8983c52cfbc"), "username" : "Sachin", "Basic" : { "name" : "Sachin Tendehar", "Age" : 49, "permanantaddress" : "Mumbai", "Occupation" : "Cricketer" }, "Service" : { "Saddress" :
"N.A", "Post" : "Batsman", "Salary" : "4562200" }, "AdharInfo" : { "number" : "8844552" }, "PanInifo" : { "number" : "ABcd78" } }
{ "_id" : ObjectId("5ff99ac06da1c8983c52cfbd"), "username" : "Dainel", "Basic" : { "name" : "Dainel de Santa", "Age" : 34, "permanantaddress" : "Banglore", "Occupation" : "bussiness" }, "Service" : { "Saddress"
: "Banglore", "Post" : "Owner", "Salary" : "875422200" }, "AdharInfo" : { "number" : "4466548" }, "PanInifo" : { "number" : "GSDF4" } }
{ "_id" : ObjectId("5ff99c356da1c8983c52cfbe"), "username" : "Beti", "Basic" : { "name" : "Beti Dsuza", "Age" : 16, "permanantaddress" : "Manglore", "Occupation" : "Student" }, "Service" : { "Saddress" : "Banglo
re", "Post" : "Student", "Salary" : "0" }, "AdharInfo" : { "number" : "44529785" }, "PanInifo" : { "number" : "Basd54" } }
{ "_id" : ObjectId("5ff99c886da1c8983c52cfbf"), "username" : "Grahm", "Basic" : { "name" : "Grahm De Villeres", "Age" : 82, "permanantaddress" : "Goa", "Occupation" : "Retired Army" }, "Service" : { "Saddress" :
 "N.A", "Post" : "Batallion Cheif", "Salary" : "500000" }, "AdharInfo" : { "number" : "16544646" }, "PanInifo" : { "number" : "HBDASFG454" } }
>
```

This is not a feasible form to view a document:

Pretty() method:

Syntax for Pretty()

db.<collection_name>.find().pretty();



Findone() method:

- Findone() helps to retrieve a single document from the collection.
- In addition to this we can also limit the output by syntax:

```
db.COLLECTION_NAME.find().limit(NUMBER)
```

Updating and Deleting Docs:

- The Syntax for Update is simple where
- db.Collectionname.update({key:value to be changed}, {$set:{New Values}})

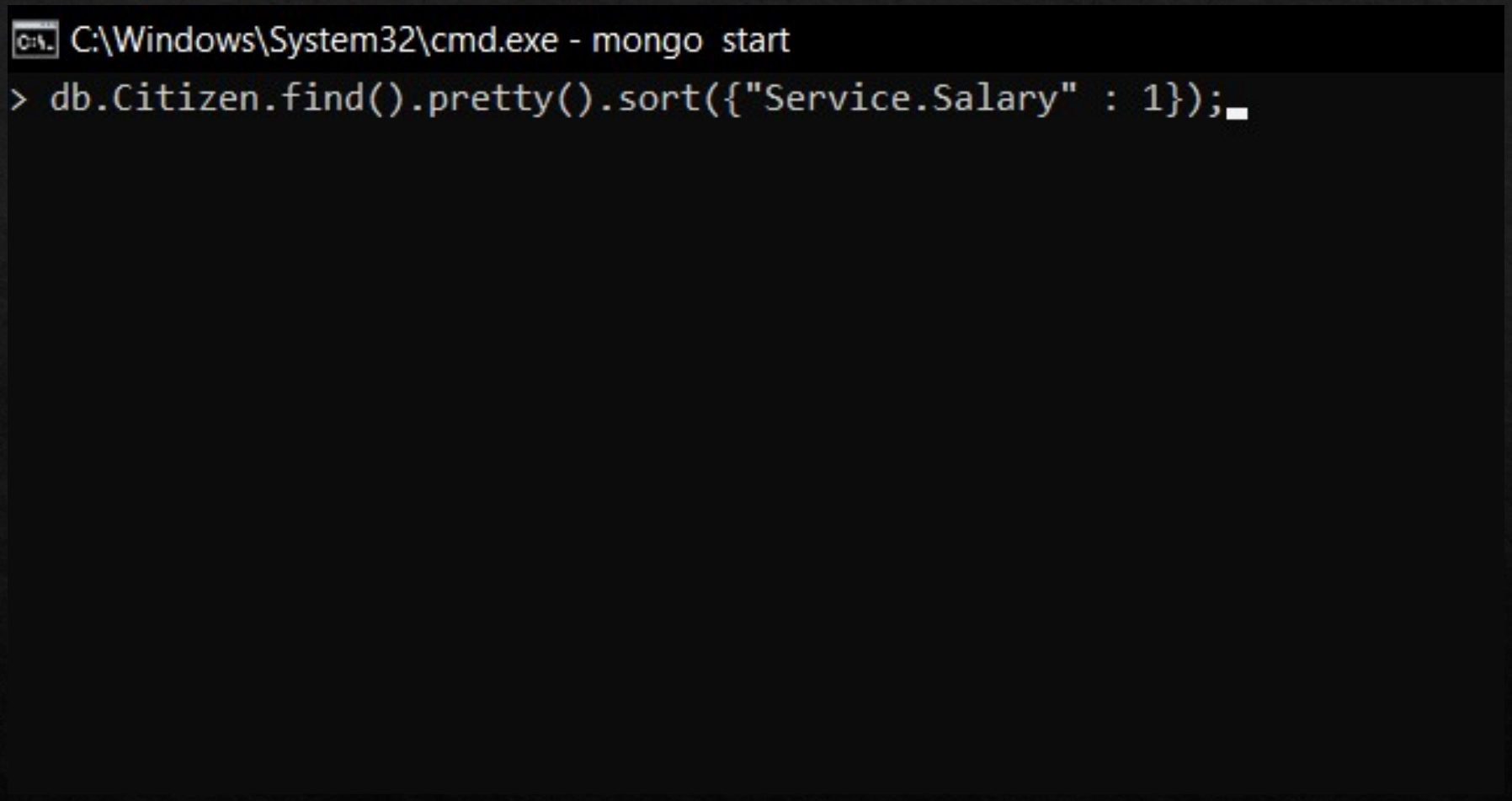
- To Delete Also we can simply do:

  db.Collectionname.remove({Key : value});

SORTING

- We can sort the documents in mongo db in ascending or descending order.

```
db.collectioname.find().sort({key : 1/-1})
```

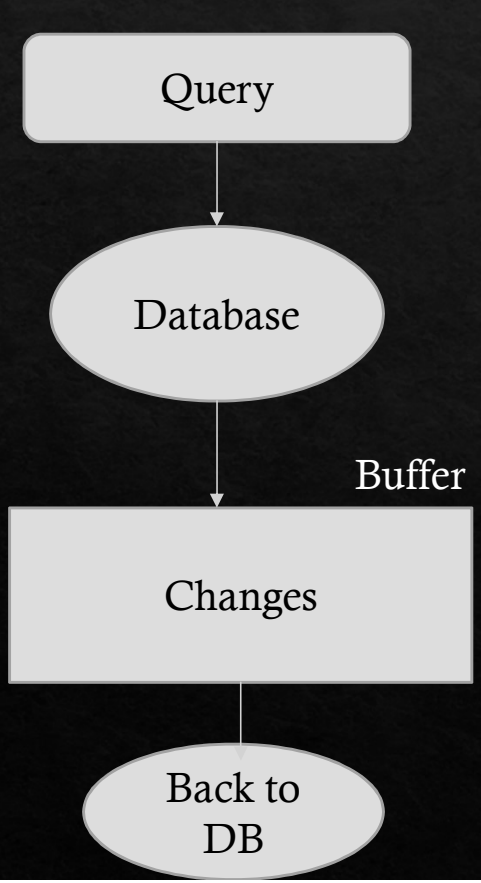

C:\Windows\System32\cmd.exe - mongo start
> db.Citizen.find().pretty().sort({"Service.Salary" : 1});

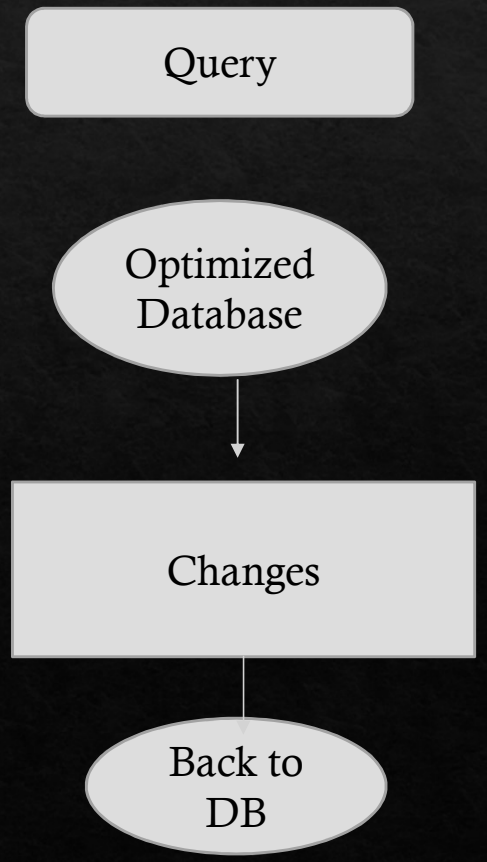
C:\Windows\System32\cmd.exe - mongo start

```
> db.Citizen.find().pretty().sort({"Service.Salary" : 1});
{
        "_id" : ObjectId("5ff99c356da1c8983c52cfbe"),
        "username" : "Beti",
        "Basic" : {
                "name" : "Beti Dsuza",
                "Age" : 16,
                "permanantaddress" : "Manglore",
                "Occupation" : "Student"
        },
        "Service" : {
                "Saddress" : "Banglore",
                "Post" : "Student",
                "Salary" : "0"
        },
        "AdharInfo" : {
                "number" : "44529785"
        },
        "PanInifo" : {
                "number" : "Basd54"
        }
}
{
        "_id" : ObjectId("5ff998e06da1c8983c52cfbb"),
        "username" : "Panny",
        "Basic" : {
                "name" : "Siraj ",
                "Age" : 23,
                "permanantaddress" : "Aurangabad",
                "Occupation" : "Cricketer"
        },
        "Service" : {
```

INDEXING

It is a data Structure to improve speed of data retrieval





What an index actually contain?

Values/Data from a field or set of fields:

Factors to choose index:

- Access type
- Access time
- Insertion time
- Deletion time
- Space overhead

- Examples: Btree , Hashmaps etc.

AGGREGATION

Expression	Description	Example
$sum	Sums up the defined value from all documents in the collection.	db.mycol.aggregate([{$group : {_id : "$by_user", num_tutorial : {$sum : "$likes"}}}])
$avg	Calculates the average of all given values from all documents in the collection.	db.mycol.aggregate([{$group : {_id : "$by_user", num_tutorial : {$avg : "$likes"}}}])
$min	Gets the minimum of the corresponding values from all documents in the collection.	db.mycol.aggregate([{$group : {_id : "$by_user", num_tutorial : {$min : "$likes"}}}])
$max	Gets the maximum of the corresponding values from all documents in the collection.	db.mycol.aggregate([{$group : {_id : "$by_user", num_tutorial : {$max : "$likes"}}}])
$push	Inserts the value to an array in the resulting document.	db.mycol.aggregate([{$group : {_id : "$by_user", url : {$push: "$url"}}}])
$addToSet	Inserts the value to an array in the resulting document but does not create duplicates.	db.mycol.aggregate([{$group : {_id : "$by_user", url : {$addToSet : "$url"}}}])
$first	Gets the first document from the source documents according to the grouping. Typically this makes only sense together with some previously applied “$sort”-stage.	db.mycol.aggregate([{$group : {_id : "$by_user", first_url : {$first : "$url"}}}])
$last	Gets the last document from the source documents according to the grouping. Typically this makes only sense together with some previously applied “$sort”-stage.	db.mycol.aggregate([{$group : {_id : "$by_user", last_url : {$last : "$url"}}}])

SOME ADVANCED FUNCTIONS

- Creating own 12bit ObjectId:

  Syntax: obID = OBjectID(“12bit hexadecimal ID”);

- Getting Timestamp:

  objectId().getTimestamp();

TEXT INDEXES:

Db.collectioname.createIndex(key : “text”);

Creates a text index to the collection.

Db.collectioname.find({$text:{search: “text_to_search”}})

Use Cases

- Aadhar
- MetLife
- eBay
- EA Sports

THANK YOU

References: mongo dB documentation