MONGODB

INTRODUCTION TO MONGODB:

MongoDB is an open source NoSQL database management program. NoSQL (Not only SQL) is used as an alternative to traditional relational databases. NoSQL databases are quite useful for working with large sets of distributed data. MongoDB is a tool that can manage document-oriented information, store or retrieve information.

MongoDB is used for high-volume data storage, helping organizations store large amounts of data while still performing rapidly. Organizations also use MongoDB for its ad-hoc queries, indexing, load balancing, aggregation, server-side JavaScript execution and other features.

Structured Query Language (SQL) is a standardized programming language that is used to manage relational databases.

How does MongoDB work?

MongoDB environments provide users with a server to create databases with MongoDB. MongoDB stores data as records that are made up of collections and documents.

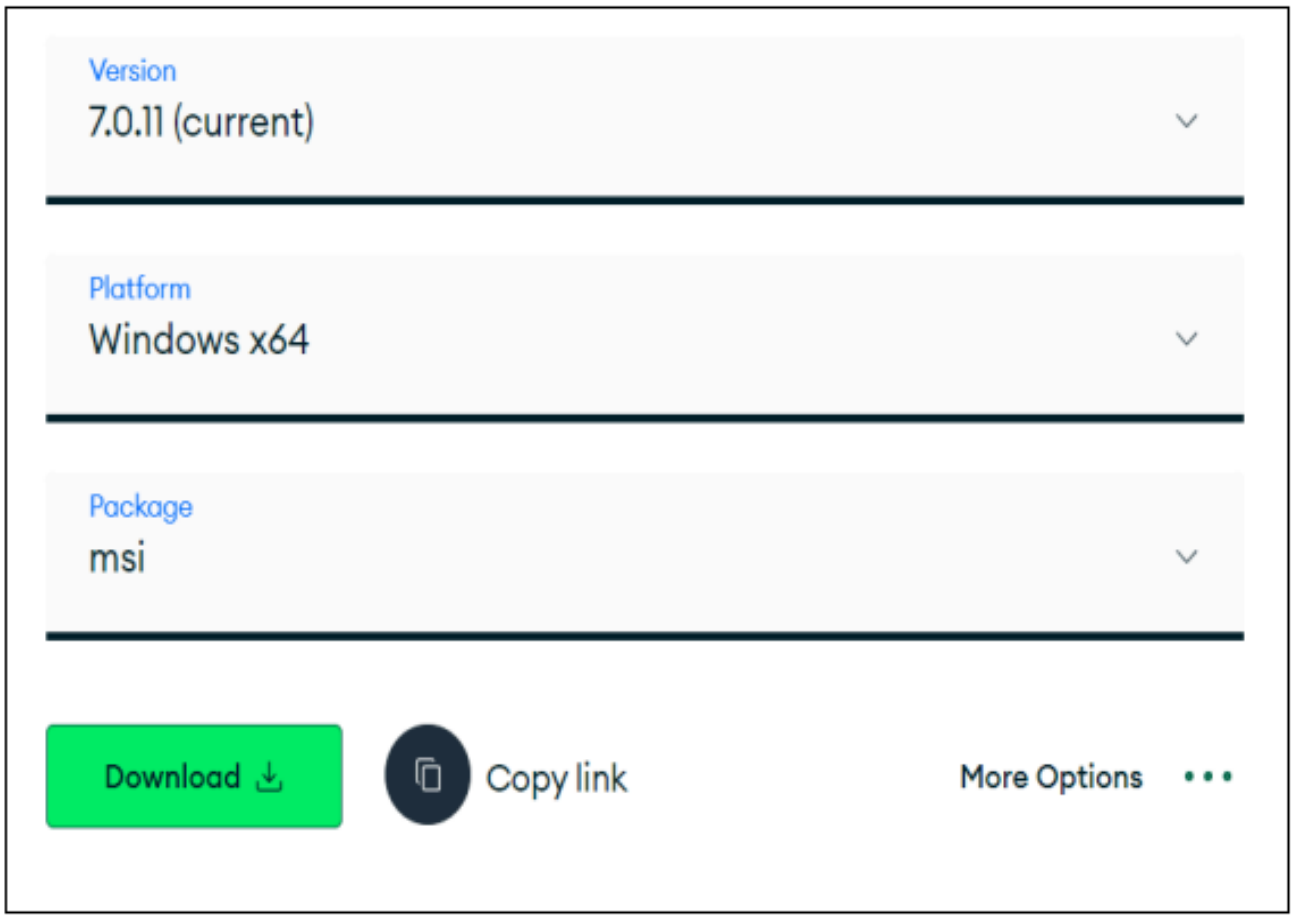
Documents contain the data the user wants to store in the MongoDB database. Documents are composed of field and value pairs. They are the basic unit of data in MongoDB. The documents are similar to JavaScript Object Notation (JSON) but use a variant called Binary JSON (BSON). The benefit of using BSON is that it accommodates more data types. The fields in these documents are like the columns in a relational database. Values contained can be a variety of data types, including other documents, arrays and arrays of documents, according to the MongoDB user manual.

INSTALLATION PROCESS:

Visit Mongodb Official Website For Downloding:

https://fastdl.mongodb.org/windows/mongodb-windows-x86_64-7.0.11-si

gned.msi



MongoDB Shell:

- Open the MongoDB Shell download page. Open the MongoDB Download Center.
- Download the mongosh installation archive for your operating system. Download mongosh from the MongoDB Download Center.
- Extract the files from the downloaded archive. ...
- Add the mongosh binary to your PATH environment variable

Mongodb shell download link:

https://downloads.mongodb.com/compass/mongosh-2.2.6-win32-x64.zip

Features of MongoDB:

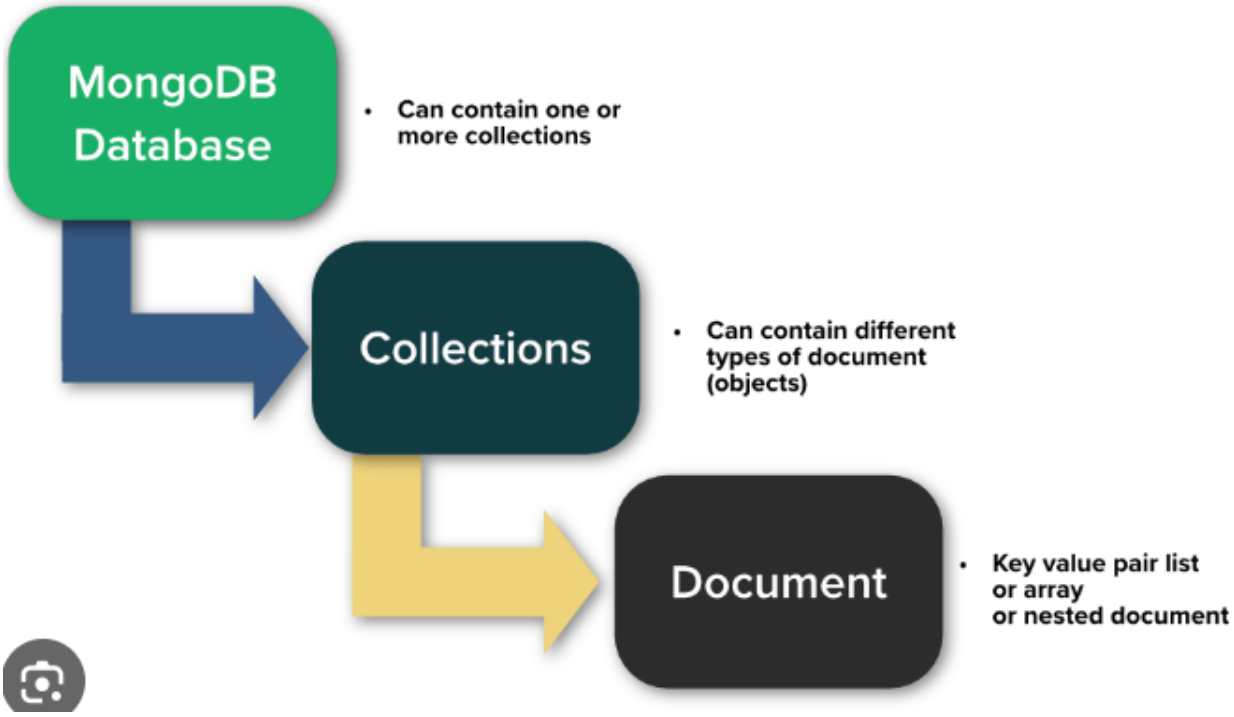
- Replication. A replica set is two or more MongoDB instances used to provide high availability. Replica sets are made of primary and secondary servers. The primary MongoDB server performs all the read and write operations, while the secondary replica keeps a copy of the data. If a primary replica fails, the secondary replica is then used.
- Scalability. MongoDB supports vertical and horizontal scaling. Vertical scaling works by adding more power to an existing machine, while horizontal scaling works by adding more machines to a user's resources.
- Load balancing. MongoDB handles load balancing without the need for a separate, dedicated load balancer, through either vertical or horizontal scaling.
- Schema-less. MongoDB is a schema-less database, which means the database can manage data without the need for a blueprint.

What is Database?

Structured Data: The information is typically organized in a specific format, often using tables with rows and columns. This makes it easier to search, filter, and analyze the data.

Database Management System (DBMS): This is the software that acts like the filing cabinet manager. It allows you to store, retrieve, update, and manage all the data within the database.

Data Types: Databases can hold various kinds of information, including text, numbers, images, videos, and more.



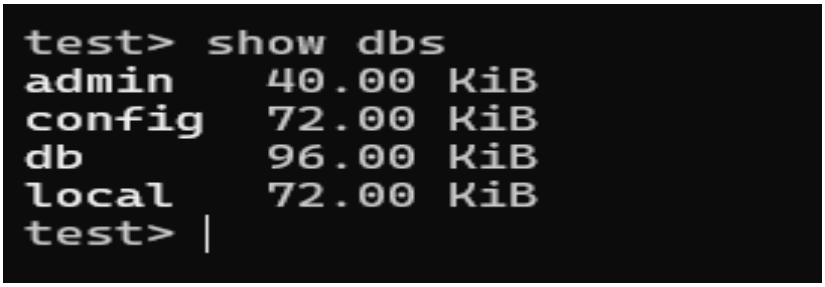
CLASS 2:
ADD , UPDATE AND DELETE

Download the student csv file and import the data to the collection
We can able to see the uploaded data in mongodb compass

Few commands to test after connection are:

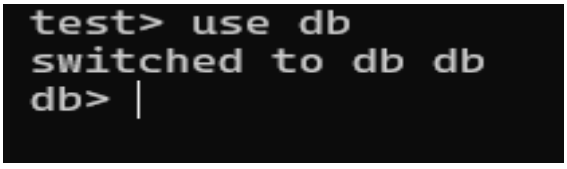
Command	Expected Output	Notes
show dbs	admin 40.00 KiB config 72.00 KiB db 128.00 KiB local 40.00 KiB	All Databases are shown
use db	switched to db db	Connect and use db
show collections	Students	Show all tables
db.foo.insert({"bar" : "baz"})		Insert a record to collection. Create Collection if not exists

1. Show dbs :-
If you want to see a list of databases in MongoDB, you can use the `show dbs` command. However, it's important to note that this command will only display databases that have data in them. If a database doesn't contain any collections, it might not show up in the list.
Here's how you can use it in the MongoDB shell:



This will display a list of databases along with their sizes. The size displayed is the total storage size of the database.
Remember that the `show dbs` command is only available in the MongoDB shell. If you're using a MongoDB client or driver in a programming language, you'll
typically use methods provided by the client to list databases and perform other Operations.

2. Use db :-
To switch to a specific database in MongoDB, you can use the `use` command followed by the name of the database you want to switch to.
Here's how you can do it in the MongoDB shell:
> use yourDatabaseName
switched to db yourDatabaseName Replace `yourDatabaseName` with the name of the database you want to switch to.
If the specified database doesn't exist, MongoDB will create it for you when you first
write data to it.



3. Show collections:-
To list the collections in the currently selected database in MongoDB, you can use the `show collections` command in the MongoDB shell.
Here's how you can do it in MongoDB shell
> show collections

This command will display a list of collections within the currently selected database.
If you haven't selected a database yet, you can first use the `use` command to switch
to the desired database, and then use `show collections`.

4. insertone():
Insert a record to collection. Create collection if not exists
Here's how you can use "`insertOne()`"to achieve the same result in the MongoDB shell:
db.foo.insert({"bar": "baz"})

5. insertmany():-
Insert the more then one document
Here's how you can use "`insertmany()`"
db.foo.insertMany([
{"bar": "baz1"},
{"bar": "baz2"},
// Add more documents as needed
])

6. find() :-
To retrieve documents from a collection in MongoDB, you can use the "`find()`" method. This method returns a cursor to the documents that match the query criteria. Here's how you can use it:
db.collectionName.find()
here,collectionName means with the name of the collection you want to guery

7. Remove() :-
Used to Remove the collection table

Here how you can use it:
db.collectionName.remove()

DOCUMENTS,COLLECTIONS,DATABASE

Documents:

At the heart of MongoDB is the document:-an ordered set of keys with associated values.
The representation of a document varies by programming language, but most languages have a data structure that is a natural fit, such as a map, hash, or dictionary

Collection:

Collection is a group of MongoDB documents. It is the equivalent of an RDBMS table. A collection exists within a single database. Collections do not enforce a schema. Documents within a collection can have different fields. Typically, all documents in a collection are of similar or related purpose.

```
db> show collections
student
```

Database:

MongoDB is a NoSQl distributed database program. Because data doesn't need to fit within the confines of a strict relationship, MongoDB can operate as a general data store. This database provides several advantages.

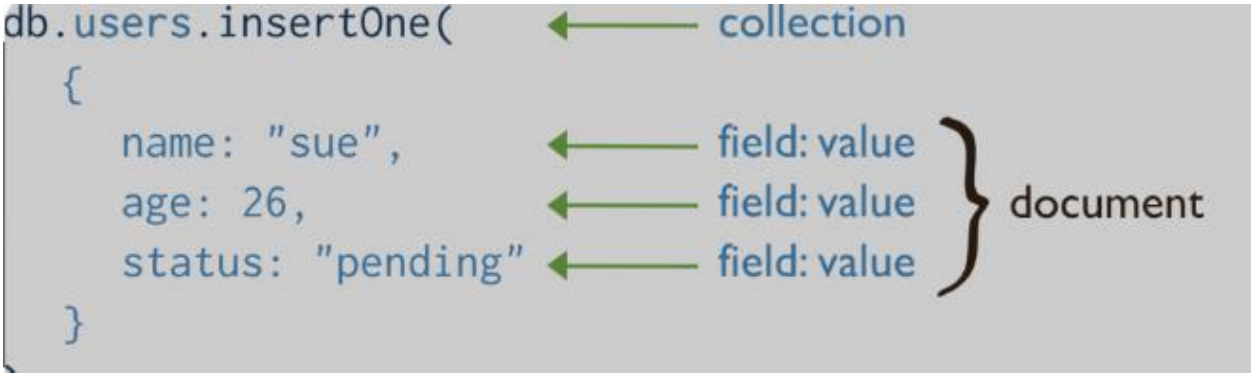
Data type:

MongoDB Server stores data using the BSON format which supports some additional data types that are not available using the JSON format. Compared to the legacy `mongo` shell, MongoDB Shell (`mongosh`) has type handling which is better aligned with the default types used by the MongoDB drivers.

Types of data types are:

- Date
- Int32
- Decimal
- Timestamp

Example:



CLASS THREE:
Where, AND, OR & CRUD

WHERE:-
Given a collection you want to filter a subset based on a condition. That is the place WHERE is used

Here how you can do it in MongoDB shell:-

MongoDB provides various comparison operators, such as $gt (greater than), $lt(less than), $gte (greater than or equal to), $lte (less than or equal to), $ne (not equal), and $in (matches any of the values specified in an array)

AND:-
Given a collection you want to FILTER a subset based on multiple conditions.

Here how you can do it in MongoDB shell:-

```
db> db.students.find({
... $and: [
... { home_city:"City 5"},
... {blood_group:"A+"}
... ]
... });
[
  {
    _id: ObjectId('666852df0851d739a08f8bdb'),
    name: 'Student 142',
    age: 24,
    courses: "['History', 'English', 'Physics', 'Computer Science']",
    gpa: 3.41,
    home_city: 'City 5',
    blood_group: 'A+',
    is_hotel_resident: false
  },
  {
    _id: ObjectId('666852df0851d739a08f8cfb'),
    name: 'Student 947',
    age: 20,
    courses: "['Physics', 'History', 'English', 'Computer Science']",
    gpa: 2.86,
    home_city: 'City 5',
    blood_group: 'A+',
    is_hotel_resident: true
  },
  {
    _id: ObjectId('666852df0851d739a08f8d6d'),
    name: 'Student 567',
    age: 22,
    courses: "['Computer Science', 'History', 'English', 'Mathematics']",
    gpa: 2.01,
    home_city: 'City 5',
    blood_group: 'A+',
```

The $and operator is not necessary in this case because the find method implicitly uses $and when you specify multiple conditions at the same level.

OR:-

Given a collection you want to filter a subset based on multiple conditions but any one is sufficient

Here how you can do it in MongoDB shell:-

```
db> db.students.find({
... $or:[
... {age:18},
... {gpa:{$lt:3.0} }
... ]
... });
[
  {
    _id: ObjectId('666852df0851d739a08f8ba5'),
    name: 'Student 157',
    age: 20,
    courses: "['Physics', 'English']",
    gpa: 2.27,
    home_city: 'City 4',
    blood_group: 'O-',
    is_hotel_resident: true
  },
  {
    _id: ObjectId('666852df0851d739a08f8ba6'),
    name: 'Student 316',
    age: 20,
    courses: "['Physics', 'Computer Science', 'Mathematics', 'History']",
    gpa: 2.32,
    blood_group: 'B+',
    is_hotel_resident: true
  },
  {
    _id: ObjectId('666852df0851d739a08f8bab'),
    name: 'Student 563',
    age: 18,
    courses: "['Mathematics', 'English']",
```

In this above example, the students database is filtered based on either “hotel_resident: true” or “gpa is less than 3.0”
If you want to use an "or" condition in MongoDB, you use the $or logical operator.
This allows you to query documents that match at least one of the specified conditions.

CRUD OPERATIONS:-

C:-Create/Insert
R:-Remove
U:-Update
D:-Delete

This is applicable for a Collection (Table) or a Document (Row)

CREATE OR INSERT:-

We can insert the single document and also multiple document into a collection

Here how you can do it in MongoDB shell:-

```
db> const studentData={ "name":"Alice", "age":20 ,"name":"Alice","home_city
":"USA"};

db> db.students.insertOne(studentData);
{
  acknowledged: true,
  insertedId: ObjectId('66685f4fe75af6778bcdcdf6')
db>
```

In this above example ,single student document is insert

DELETE OR REMOVE:

Removes/Deletes a single document from a collection.

```
db> db.student.deleteOne({name:"Bob"});
{ acknowledged: true, deletedCount: 1 }
db>
```

UPDATE:

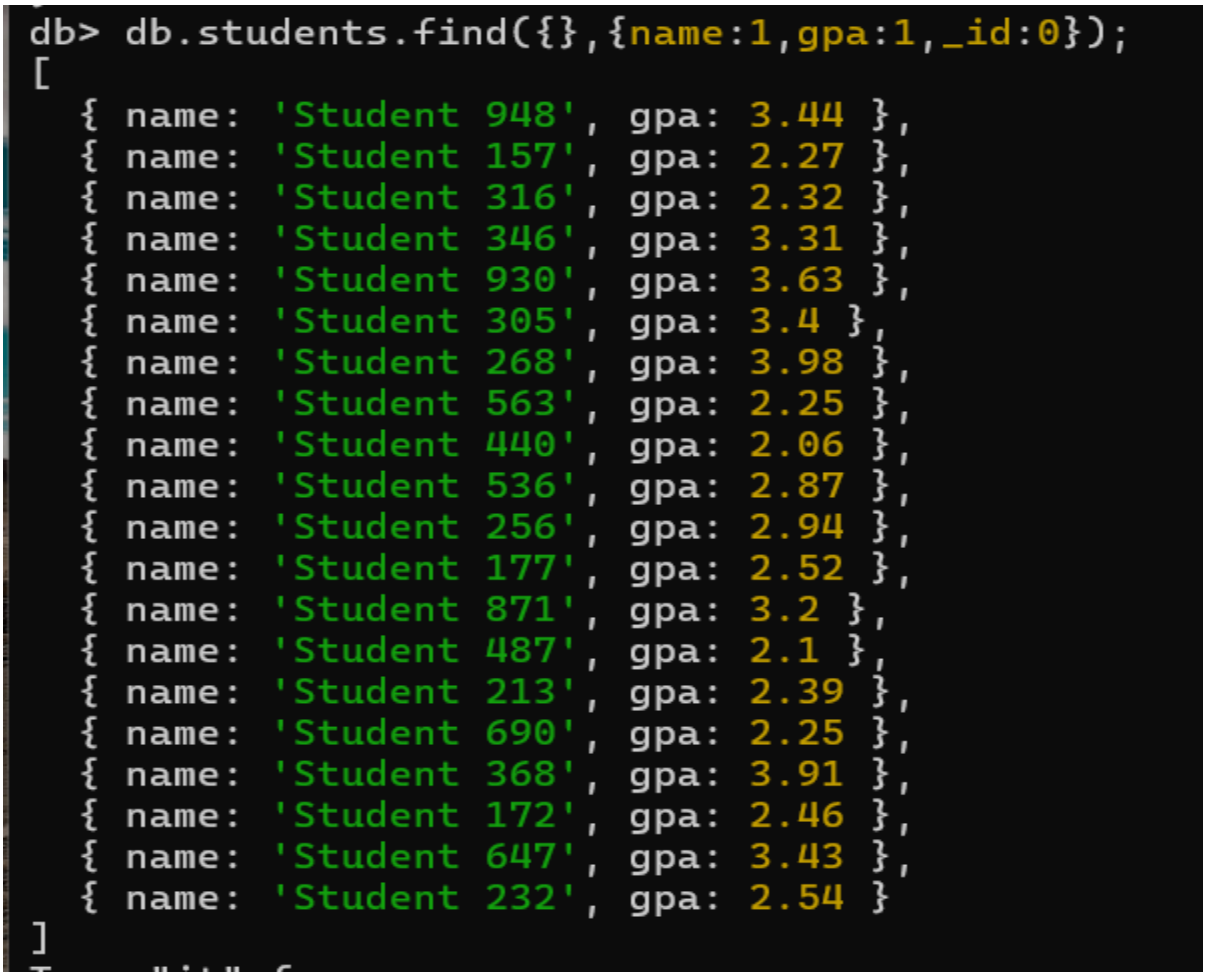
Modifies an existing document or documents in a collection. The method can modify specific fields of an existing document or documents or replace an existing document entirely, depending on the updated parameter.

```
db> db.students.updateOne({ name:"Alice Smith"},{$set:{gpa:3.8}});
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 0,
  modifiedCount: 0,
  upsertedCount: 0
}
db>
```

CLASS 4:
PROJECTION,LIMIT & SELECTORS

PROJECTION:-

This is used when we don't need all columns / attributes.



```
db> db.students.find({},{name:1,gpa:1,_id:0});
[
  { name: 'Student 948', gpa: 3.44 },
  { name: 'Student 157', gpa: 2.27 },
  { name: 'Student 316', gpa: 2.32 },
  { name: 'Student 346', gpa: 3.31 },
  { name: 'Student 930', gpa: 3.63 },
  { name: 'Student 305', gpa: 3.4 },
  { name: 'Student 268', gpa: 3.98 },
  { name: 'Student 563', gpa: 2.25 },
  { name: 'Student 440', gpa: 2.06 },
  { name: 'Student 536', gpa: 2.87 },
  { name: 'Student 256', gpa: 2.94 },
  { name: 'Student 177', gpa: 2.52 },
  { name: 'Student 871', gpa: 3.2 },
  { name: 'Student 487', gpa: 2.1 },
  { name: 'Student 213', gpa: 2.39 },
  { name: 'Student 690', gpa: 2.25 },
  { name: 'Student 368', gpa: 3.91 },
  { name: 'Student 172', gpa: 2.46 },
  { name: 'Student 647', gpa: 3.43 },
  { name: 'Student 232', gpa: 2.54 }
]
```

Benefits of Projection:-

- Reduced data transferred between the database and your application.
- Improves query performance by retrieving only necessary data.
- Simplifies your code by focusing on the specific information you need.

LIMIT :-

- The limit operator is used with the find method.
- It's chained after the filter criteria or any sorting operations.

Syntax:
db.collection.find({filter}, {projection}).limit(number)Here how you can do it in MongoDB shell:-

```
db> db.students.find({},{_id:0}).limit(4);
[
  {
    name: 'Student 948',
    age: 19,
    courses: "['English', 'Computer Science', 'Physics', 'Mathematics']",
    gpa: 3.44,
    home_city: 'City 2',
    blood_group: 'O+',
    is_hotel_resident: true
  },
  {
    name: 'Student 157',
    age: 20,
    courses: "['Physics', 'English']",
    gpa: 2.27,
    home_city: 'City 4',
    blood_group: 'O-',
    is_hotel_resident: true
  },
  {
    name: 'Student 316',
    age: 20,
    courses: "['Physics', 'Computer Science', 'Mathematics', 'History']",
    gpa: 2.32,
    blood_group: 'B+',
    is_hotel_resident: true
  },
  {
    name: 'Student 346',
    age: 25,
    courses: "['Mathematics', 'History', 'English']",
    gpa: 3.31,
    home_city: 'City 8',
    blood_group: 'O-',
    is_hotel_resident: true
  }
]
```

For getting the top 10 results we use the following command:

```
db> db.students.find({},{_id:0}).sort({_id:-1}).limit(3);
[
  { name: 'Alice', age: 20, home_city: 'USA' },
  {
    name: 'Student 591',
    age: 20,
    courses: "['Mathematics', 'History', 'English']",
    gpa: 2.27,
    home_city: 'City 4',
    blood_group: 'AB+',
    is_hotel_resident: false
  },
  {
    name: 'Student 933',
    age: 18,
    courses: "['Mathematics', 'English', 'Physics', 'History']",
    gpa: 2.54,
    home_city: 'City 10',
```

SELECTORS:

Comparing the greater than and less than operators by using the operators such as $gt,$lt.

In the below example we got data of the students age less than 20.

```
db> db.students.find({age:{$lt:20}});
[
  {
    _id: ObjectId('666852df0851d739a08f8ba4'),
    name: 'Student 948',
    age: 19,
    courses: "['English', 'Computer Science', 'Physics', 'Mathematics']",
    gpa: 3.44,
    home_city: 'City 2',
    blood_group: 'O+',
    is_hotel_resident: true
  },
  {
    _id: ObjectId('666852df0851d739a08f8bab'),
    name: 'Student 563',
    age: 18,
    courses: "['Mathematics', 'English']",
    gpa: 2.25,
    blood_group: 'AB+',
    is_hotel_resident: false
  },
  {
    _id: ObjectId('666852df0851d739a08f8bae'),
    name: 'Student 256',
    age: 19,
    courses: "['Computer Science', 'Mathematics', 'History', 'English']",
    gpa: 2.94,
    home_city: 'City 1',
    blood_group: 'B+',
    is_hotel_resident: true
```

$gt=greater than

$lt=lesser than

Bitwise Value:

- In our example its a 32 bit each bit representing different things
- Bitwise value 7 means all access 7 -> 111

BITWISE TYPES:

Bitwise

Name	Description
`$bitsAllClear`	Matches numeric or binary values in which a set of bit positions *all* have a value of `0`.
`$bitsAllSet`	Matches numeric or binary values in which a set of bit positions *all* have a value of `1`.
`$bitsAnyClear`	Matches numeric or binary values in which *any* bit from a set of bit positions has a value of `0`.
`$bitsAnySet`	Matches numeric or binary values in which *any* bit from a set of bit positions has a value of `1`.

GEOSPATIAL:

To perform a geospatial query, create a query filter with a field name and a geospatial query operator.

In MongoDB, you can store geospatial data as GeoJSON objects or as legacy coordinate pairs.

GeoJSON Objects:

To calculate geometry over an Earth-like sphere, store your location data as GeoJSON objects.

To specify GeoJSON data, use an embedded document with:

- a field named type that specifies the GeoJSON object type, and
- a field named coordinates that specifies the object's coordinates.

For example, to specify a GeoJSON Point:

```
location: {
      type: "Point",
      coordinates: [-73.856077, 40.848447]
}
```