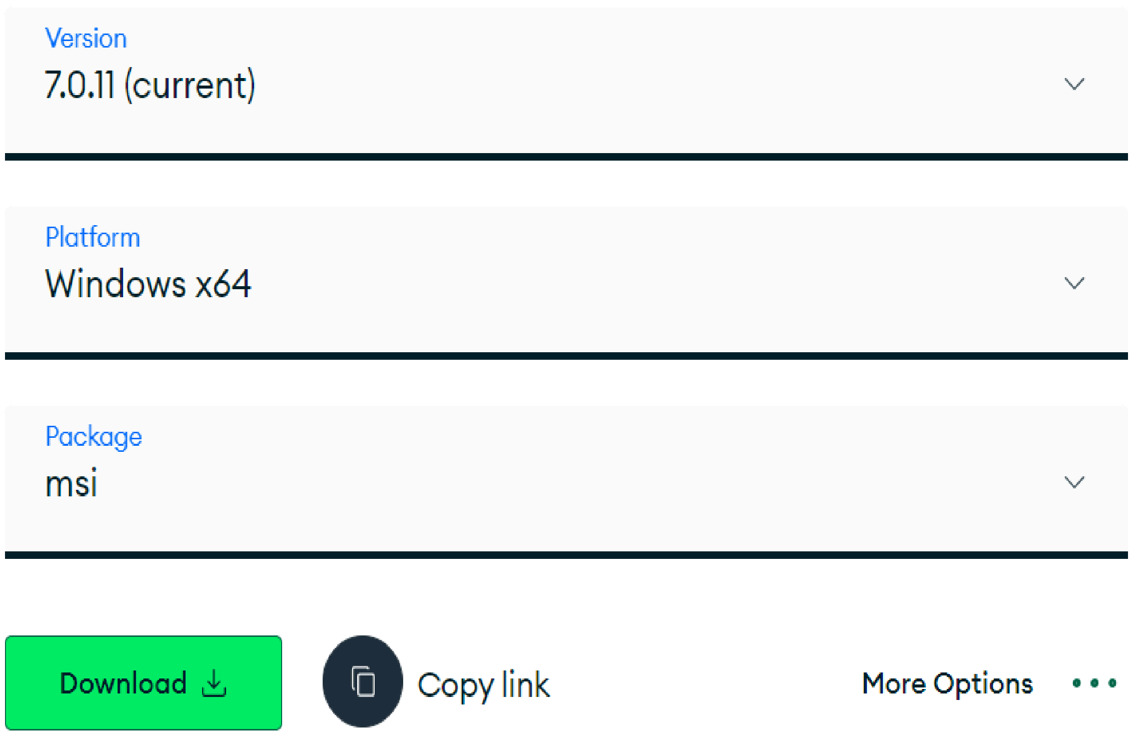
# MongoDB

## INSTALLATION PROCESS:

Visit Mongodb Official Website For Downloding:
**https://fastdl.mongodb.org/windows/mongodb-windows-x86_64-7.0.11-signed.msi**



Version
7.0.11 (current)

Platform
Windows x64

Package
msi

## MongoDB Shell

**MongoDB Shell is the quickest way to connect to (and work with) MongoDB. Easily query data, configure settings, and execute other actions with this modern, extensible command-line interface — replete with syntax highlighting, intelligent autocomplete, contextual help, and error messages.**

Download link :
[https://downloads.mongodb.com/compass/mongosh-2.2.6-win32-x64.zip](https://downloads.mongodb.com/compass/mongosh-2.2.6-win32-x64.zip)

# INTRODUCTION

The Introduction to MongoDB course guides you through the foundational skills and knowledge you need to get started with MongoDB. This includes connecting to a MongoDB database, how to conduct simple CRUD operations, and key topics such as aggregation, indexing, data modeling, and transactions.

## Brief History of MongoDB

MongoDB was developed by Eliot Horowitz and Dwight Merriman in the year 2007, when they experienced some scalability issues with the relational database while developing enterprise web applications at their company DoubleClick. According to Dwight Merriman, one of the developers of MongoDB, this name of the database was derived from the word *humongous* to support the idea of processing large amount of data.

In 2009, MongoDB was made as an open source project, while the company offered commercial support services. Many companies started using MongoDB for its amazing features. The New York Times newspaper used MongoDB to build a web based application to submit the photos. In 2013, the company was officially named to MongoDB Inc.

## Key Features of MongoDB

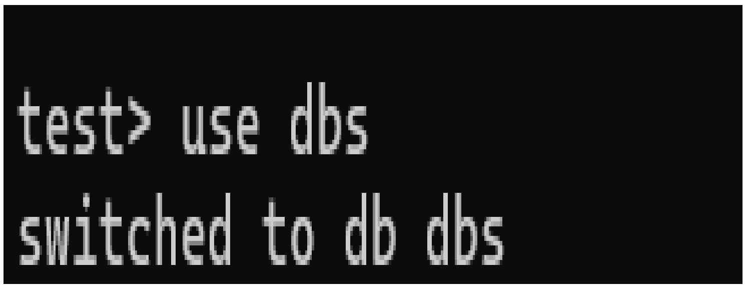
Apart from most of the NoSQL default features, MongoDB does bring in some more, very important and useful features :

1. MongoDB provides high performance. Input/Output operations are lesser than relational databases due to support of embedded documents(data models) and Select queries are also faster as Indexes in MongoDB supports faster queries.
2. MongoDB has a rich Query Language, supporting all the major CRUD operations. The Query Language also provides good Text Search and Aggregation features.

3. Auto Replication feature of MongoDB leads to High Availability. It provides an automatic failover mechanism, as data is restored through backup(replica) copy if server fails.
4. Sharding is a major feature of MongoDB. Horizontal Scalability is possible due to sharding.
5. MongoDB supports multiple Storage Engines. When we save data in form of documents(NoSQL) or tables(RDBMS) who saves the data? It's the Storage Engine. Storage Engines manages how data is saved in memory and on disk.

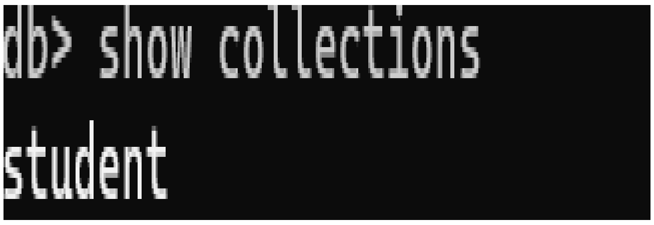
MongoDB is a cross-platform, document oriented database that provides, high performance, high availability, and easy scalability. MongoDB works on concept of collection and document.

**Database:** Database is a physical container for collections. Each database gets its own set of files on the file system. A single MongoDB server typically has multiple databases.



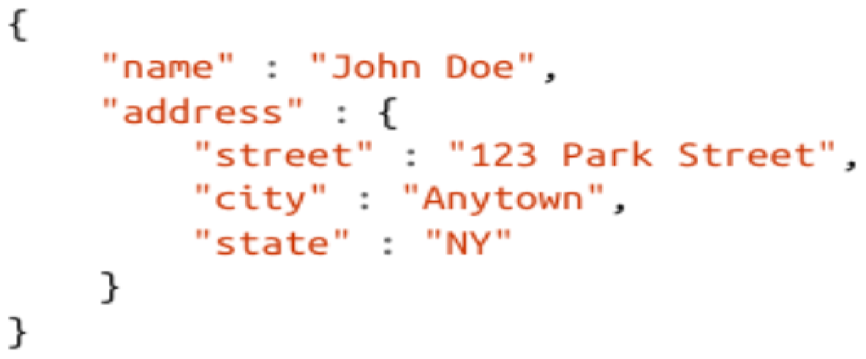
```
test> use dbs
switched to db dbs
```

**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
```

**Datatype:** A document is a set of key-value pairs. Documents have dynamic schema.

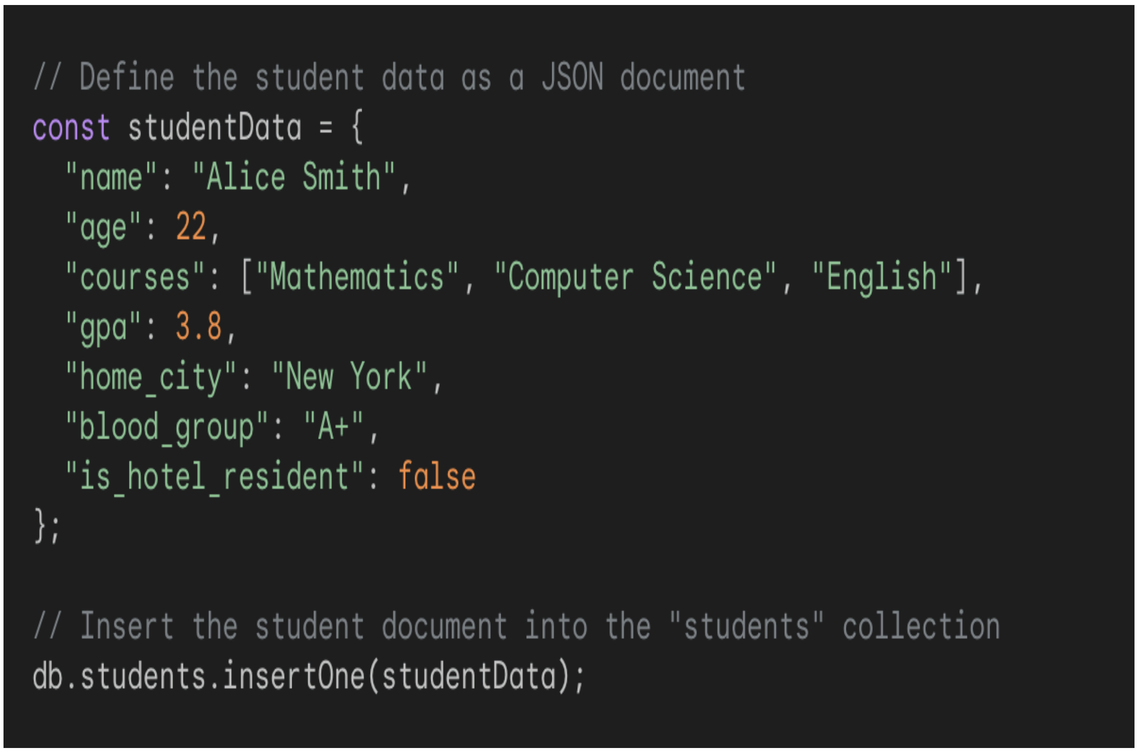


```
{
    "name" : "John Doe",
    "address" : {
        "street" : "123 Park Street",
        "city" : "Anytown",
        "state" : "NY"
    }
}
```

## How To Insert, Update, and Delete,Delete Many Records in MongoDB :

### Insert

The insert() method in MongoDB inserts documents in the MongoDB collection. This method is also used to create a new collection by inserting documents.



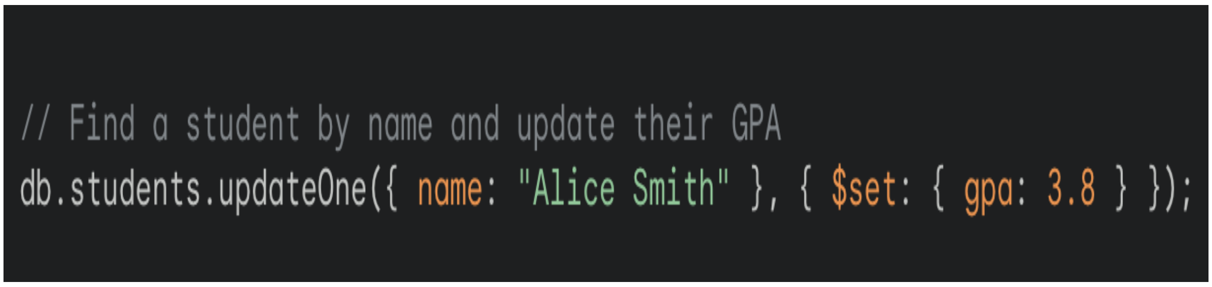
```
// Define the student data as a JSON document
const studentData = {
  "name": "Alice Smith",
  "age": 22,
  "courses": ["Mathematics", "Computer Science", "English"],
  "gpa": 3.8,
  "home_city": "New York",
  "blood_group": "A+",
  "is_hotel_resident": false
};

// Insert the student document into the "students" collection
db.students.insertOne(studentData);
```

### Updating Data

The update() method in MongoDB updates a document or multiple documents in the collection. When the document is updated the _id field remains unchanged.

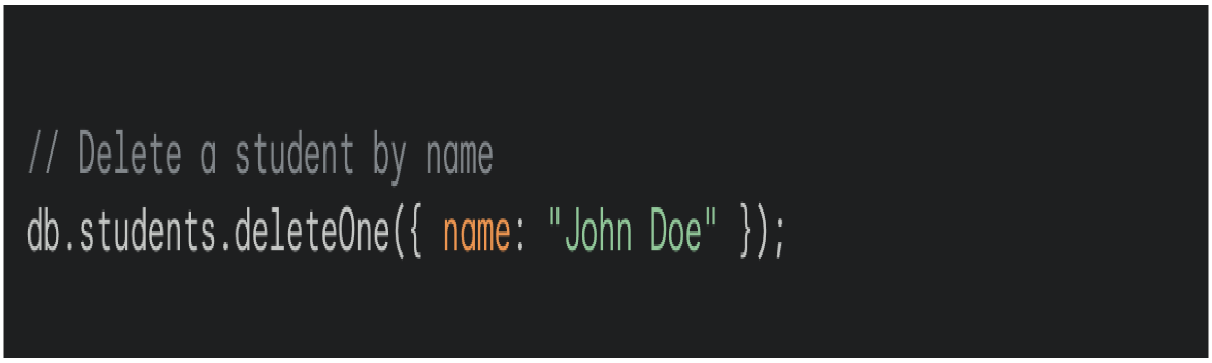
This method can be used for a single updating of documents as well as multiple documents. By default, the db.collection.update() method updates a single document. To update all documents that match the given query. Include the option “multi: true“.



```
// Find a student by name and update their GPA
db.students.updateOne({ name: "Alice Smith" }, { $set: { gpa: 3.8 } });
```

## Deleting Data (D)

**The `delete` command removes documents from a collection. A single `delete` command can contain multiple delete specifications. The delete methods provided by the MongoDB drivers use this command internally.**



```
// Delete a student by name
db.students.deleteOne({ name: "John Doe" });
```

## Update Many

The updateMany() method allows you to **update all documents that satisfy a condition**.

```
// Update all students with a GPA less than 3.0 by increasing it by 0.5
db.students.updateMany({ gpa: { $lt: 3.0 } }, { $inc: { gpa: 0.5 } });
```

## Delete Many

**The `deleteMany()` method allows you to remove all documents that match a condition from a collection.**

```
// Delete all students who are not hotel residents
db.students.deleteMany({ is_hotel_resident: false });
```

# Projection Limit and Selectors

## Projection Limit :

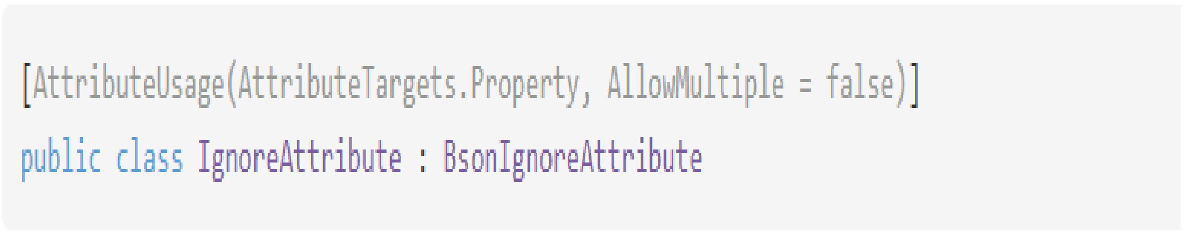
### Selected attributes

**Selects a subset of an array to return based on the specified condition. Returns an array with only those elements that match the condition. The returned elements are in the original order.**

**{ $filter: { input: <array>, as: <string>, cond: <expression> } }**

### Ignore attributes

**Use this attribute to ignore a property when persisting an entity to the database.**



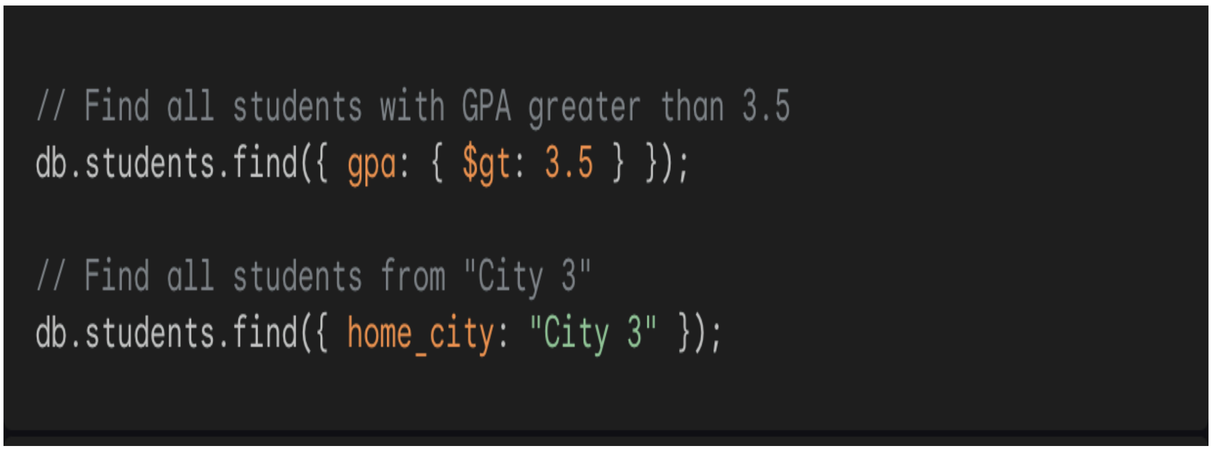
```
[AttributeUsage(AttributeTargets.Property, AllowMultiple = false)]
public class IgnoreAttribute : BsonIgnoreAttribute
```

## Selectors :

**Selects** documents in a collection or view and returns a cursor to the selected documents.

### WHERE Condition

**let's say you want to fetch data based on a `firstName` key from the user collection.**

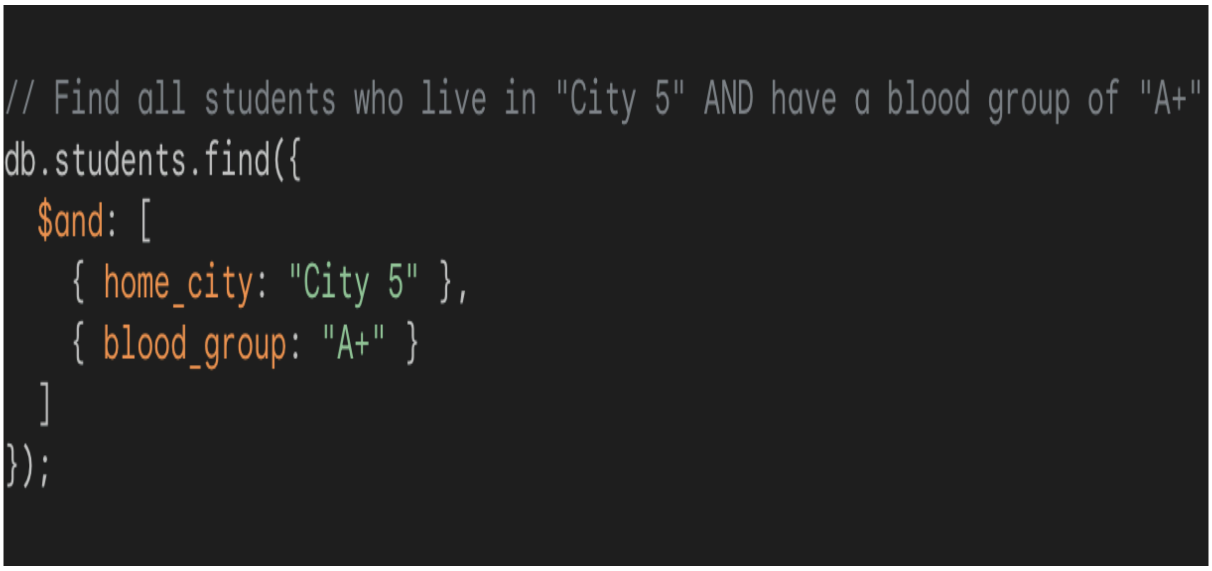


```
// Find all students with GPA greater than 3.5
db.students.find({ gpa: { $gt: 3.5 } });

// Find all students from "City 3"
db.students.find({ home_city: "City 3" });
```

### AND Condition

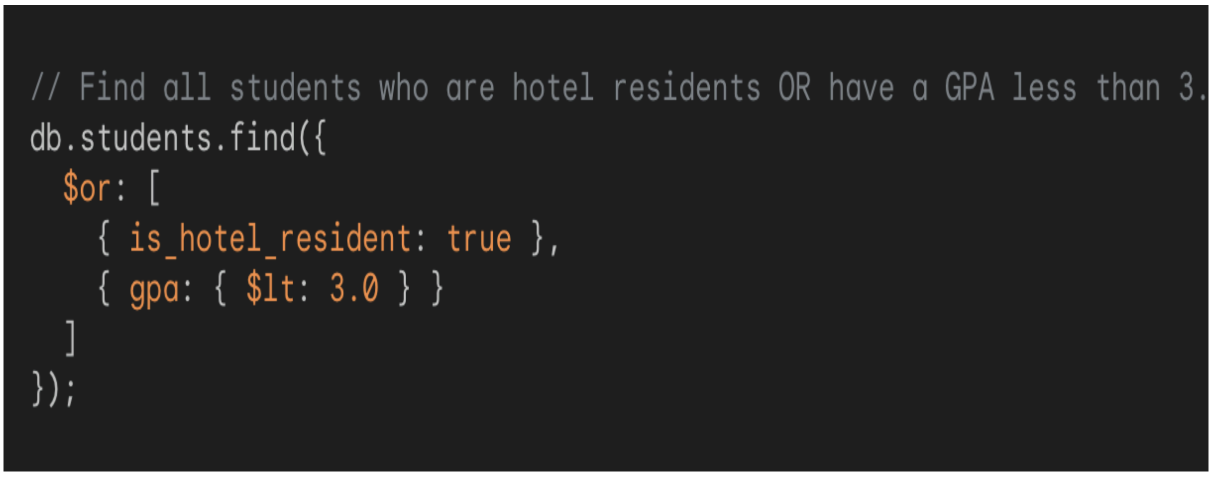
**AND condition will fetch data if both conditions match. Let's check out the following query.**



```
// Find all students who live in "City 5" AND have a blood group of "A+"
db.students.find({
  $and: [
    { home_city: "City 5" },
    { blood_group: "A+" }
  ]
});
```

## OR Condition

Matching any one of the given conditions will fetch records from database. In the AND condition you did not have to mention any AND operator but for OR condition you need to specify OR operator.



```
// Find all students who are hotel residents OR have a GPA less than 3.
db.students.find({
  $or: [
    { is_hotel_resident: true },
    { gpa: { $lt: 3.0 } }
  ]
});
```

# Introduction to the MongoDB $elemMatch, $Slice operator

## $elemMatch operator :

**The `$elemMatch` is an array query operator that matches documents that contain an array field and the array field has at least one element that satisfies all the specified queries**

**{ <arrayField>: {$elemMatch: { <query1>, <query2>, ...} } }**

## $Slice operator :

**Returns a subset of an array.**

**`$slice` has one of two syntax forms:**

**The following syntax returns elements from either the start or end of the array:**

**{ $slice: [ <array>, <n> ] }**

## Conclusion

**MongoDB was founded with the objective of creating a better database for programmers and developers. Today, MongoDB plays an integral role in managing unstructured data for thousands of companies as a leading NoSQL database.**

**MongoDB's scalability makes it great for any company in need of fast results on large datasets. Its tools such as Atlas and Compass also limit the load on local servers by using cloud servers to host their data in an easily accessible form. MongoDB has spread fast and leading corporations such as Adobe and Lyft are already using the platform to host all things data. Its scalability and versatility give MongoDB a decisive edge over many other NoSQL databases.**

**In conclusion, MongoDB is a great option for both on-premise and cloud work, and with the help of Knowi, companies can seamlessly query their data and turn data into action.**