



INTRODUCTION

MongoDB is an open-source document-oriented database that is designed to store a large scale of data and also allows you to work with that data very efficiently. It is categorized under the NoSQL (Not only SQL) database because the storage and retrieval of data in the MongoDB are not in the form of tables.

The MongoDB database is developed and managed by MongoDB .Inc under SSPL(Server Side Public License) and initially released in February 2009. It also provides official driver support for all the popular languages like C, C++, C#, and .Net, Go, Java, Node.js, Perl, PHP, Python, Motor, Ruby, Scala, Swift, Mongoid. So, that you can create an application using any of these languages. Nowadays there are so many companies that used MongoDB like Facebook, Nokia, eBay, Adobe, Google, etc. to store their large amount of data.

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.



Hence DB Software like Oracle, Mongo



1. Database softwares are like rack builders
2. Oracle, Mongo builds and gives you just use it

SQL VS NO-SQL

Excel spreadsheet showing a Student Database table with columns: Std ID, Std Name, Marks, Percentage, Grade, Remark. The formula bar shows: =IF(F5>70,"Pass","Fail").

Std ID	Std Name	Marks	Percentage	Grade	Remark
GA-001	Jhon	285	71.25	A	Pass
GA-002	Smith	248	62	B	
GA-003	Emma	329	82.25	A+	
GA-004	Sandy	249	62.25	B	
GA-005	Olivia	150	37.5	F	
GA-006	James	187	46.75	D	
GA-007	Eva	352	88	A+	
GA-008	Noah	279	69.75	B	
GA-009	Charlotte	282	70.5	A	
GA-010	Elijah	301	75.25	A	

Grade Description table:

Marks	Grade
80-100	A+
70-80	A
60-70	B
50-60	C
40-50	D
Below 40	F

Total Marks: 400

• Now, drag the Fill Handle downward to repeat the formula across the range. double-click on the Plus (+) sign to AutoFill the range.

Excel spreadsheet showing the Student Database table with the Remark column filled with 'Pass' for all rows.

Std ID	Std Name	Marks	Percentage	Grade	Remark
GA-001	Jhon	285	71.25	A	Pass
GA-002	Smith	248	62	B	
GA-003	Emma	329	82.25	A+	
GA-004	Sandy	249	62.25	B	
GA-005	Olivia	150	37.5	F	
GA-006	James	187	46.75	D	
GA-007	Eva	352	88	A+	
GA-008	Noah	279	69.75	B	
GA-009	Charlotte	282	70.5	A	
GA-010	Elijah	301	75.25	A	

```
JSON
{
  "name": "John Doe",
  "title": "Software Engineer",
  "info": "Highly skilled developer with 5+ years of experience. Proven"
}
```

Use code with caution.

This example uses a single object with key-value pairs. It includes basic employee information like name, title, and a brief description in the "info" field.

Example 2: Employee Object with Nested Data

```
JSON
{
  "name": "Jane Smith",
  "department": "Marketing",
  "contact": {
    "email": "jane.smith@company.com",
    "phone": "+1 (555) 555-1234"
  },
  "skills": ["Social Media Marketing", "Content Marketing", "SEO"]
}
```

Use code with caution.

SQL databases are used to store structured data while NoSQL databases like MongoDB are used to save unstructured data. MongoDB is used to save unstructured data in JSON format. MongoDB does not support advanced analytics and joins like SQL databases support.



No-SQL VS SQL



What is MongoDB?

MongoDB is a document-oriented NoSQL database system that provides high scalability, flexibility, and performance. Unlike standard relational databases, MongoDB stores data in a JSON document structure form. This makes it easy to operate with dynamic and unstructured data and MongoDB is an open-source and cross-platform database System.

Database

- Database is a container for collections.
- Each database gets its own set of files.
- A single MongoDB server can has multiple databases.

Collection

- Collection is a group of documents.
- Collection is equivalent to RDBMS table.
- A collection consist inside a single database.
- Collections do not enforce a schema.
- A Collection can have different fields within a Documents.

Where to Use MongoDB?

1. Mobile and Social Infrastructure
2. Data Hub
3. Previous Pag
4. Big Data
5. User Data Management
6. Content Management and Delivery



Why Use MongoDB?

Document Oriented Storage – Data is stored in the form of JSON documents.

- **Index on any attribute:** Indexing in MongoDB allows for faster data retrieval by creating a searchable structure on selected attributes, optimizing query performance.
- **Replication and high availability:** MongoDB's replica sets ensure data redundancy by maintaining multiple copies of the data, providing fault tolerance and continuous availability even in case of server failures.
- **Auto-Sharding:** Auto-sharding in MongoDB automatically distributes data across multiple servers, enabling horizontal scaling and efficient handling of large datasets.
- **Big Data and Real-time Application:** When dealing with massive datasets or applications requiring real-time data updates, MongoDB's flexibility and scalability prove advantageous.
- **Rich queries:** MongoDB supports complex queries with a variety of operators, allowing you to retrieve, filter, and manipulate data in a flexible and powerful manner.
- **Fast in-place updates:** MongoDB efficiently updates documents directly in their place, minimizing data movement and reducing write overhead.
- **Professional support by MongoDB:** MongoDB offers expert technical support and resources to help users with any issues or challenges they may encounter during their database operations.
- **Internet of Things (IoT) Applications:** Storing and analyzing sensor data with its diverse formats often aligns well with MongoDB's document structure.



INSTALLATION

MongoDB is an open-source document-oriented database. It is categorized under the **NoSQL(Not only SQL)** database because the storage and retrieval of data in MongoDB are not in the form of tables. This is the general introduction to MongoDB now we will learn **how to install MongoDB in Windows**.

You can install MongoDB using MSI. Let's see a step-by-step instruction guide for installing MongoDB in Windows using MSI. Let's see the MongoDB requirements for the installation of Windows.

Requirements to Install MongoDB on Windows

MongoDB 4.4 and later only support **64-bit** versions of Windows.

MongoDB 7.0 Community Edition supports the following 64-bit versions of Windows on x86_64 architecture:

Windows Server 2022

Windows Server 2019

Windows 11

Ensure that the user is running **mongod** and **mongos** has the necessary permissions from the following groups:

Performance Monitor Users

Performance Log Users

Follow steps here :

https://www.geeksforgeeks.org/how-to-install-mongodb-on-windows/?ref=ml_lbp

<https://www.mongodb.com/docs/manual/tutorial/install-mongodb-on-windows/#std-label-install-mdb-community-windows>



How to import data into collection in MongoDB compass

You can use MongoDB Compass to import and export data to and from collections. Compass supports import and export for both JSON and CSV files. To import or export data to or from a collection, navigate to the detailed collection view by either selecting the collection from the Databases tab or clicking the collection in the left-side navigation.

MongoDB Compass can import data into a collection from either a **JSON** or **CSV** file.

Limitations

- Importing data into a collection is not permitted in **MongoDB Compass Readonly Edition**.
- Importing data is not available if you are connected to a [Data Lake](#).

Format Your Data

Before you can import your data into MongoDB Compass you must first ensure that it is formatted correctly.

JSON

CSV

When importing data from a **JSON** file, you can format your data as:

- Newline-delimited documents, or
- Comma-separated documents in an array.



Procedure

To import your formatted data into a collection:

1. Connect to the deployment containing the [collection](#) you wish to import data into.

To learn how to connect to a deployment, see [Connect to MongoDB](#).

2. Navigate to your target collection.

You can either select the collection from the [Collections](#) tab or click the collection in the left-hand pane.

3. Click the **Add Data** dropdown and select **Import JSON or CSV file**.

4. Select the appropriate file type.

Select either a JSON or CSV file to import and click **Select**.

If you are importing a CSV file, you may specify fields to import and the types of those fields under **Specify Fields and Types**.

Import

To collection test.flights

☐ Stop on errors

Specify Fields and Types [Learn more about data types](#)

<input checked="" type="checkbox"/> _id	<input checked="" type="checkbox"/> airline	<input checked="" type="checkbox"/> name	<input checked="" type="checkbox"/> alias	<input checked="" type="checkbox"/> ia	
ObjectId	Int32	Mixed	Mixed	Mix	
1	56e9b497732b6122f8790280	4	2 Sqn No 1 Elementary Flying Training Sch...	empty string	WYT
2	56e9b497732b6122f8790281	2	135 Airways	empty string	GNL
3	56e9b497732b6122f8790282	5	213 Flight Unit	empty string	TFU
4	56e9b497732b6122f8790283	3	1Time Airline	IT	RNX
5	56e9b497732b6122f8790284	6	223 Flight Unit State Airline	empty string	CHD
6	56e9b497732b6122f8790285	1	Private flight	-	N/A
7	56e9b497732b6122f8790286	7	224th Flight Unit	empty string	TTF
8	56e9b497732b6122f8790287	8	247 Jet Ltd	empty string	TWF
9	56e9b497732b6122f8790288	9	3D Aviation	empty string	SEC
10	56e9b497732b6122f8790289	10	40-Mile Air	Q5	MLA

Cancel
Import



To exclude a field from a CSV file you are importing, uncheck the checkbox next to that field name. To select a type for a field, use the dropdown menu below that field name.

5. Configure import options.

Under **Options**, configure the import options for your use case.

If you are importing a CSV file, you may select how your data is delimited.

For both JSON and CSV file imports, you can toggle **Ignore empty strings** and **Stop on errors**:

- If checked, **Ignore empty strings** drops fields with empty string values from your imported documents. The document is still imported with all other fields.
- If checked, **Stop on errors** prevents any data from being imported in the event of an error. If unchecked, data is inserted until an error is encountered and successful inserts are not rolled back. The import operation will not continue after encountering an error in either case.

6. Click *Import*.

A progress bar displays the status of the import. If an error occurs during import, the progress bar turns red and an error message appears in the dialog. To see all errors, click **View Log**.

After successful import, the dialog closes and Compass displays the collection page containing the newly imported documents.



Lets Load the document

- Download the student csv from this [link](#)
- Import the data to the collection created [link](#)
- You should be able to see the uploaded data in mongo compass

Installation of Mongo Shell OR Studio3t

- Mongo Shell download [link](#)
- All the work is expected to do it in mongo shell not in mongo compass

OR

- You can also install [Studio3T](#)
- Connect to mongodb://localhost:27017



Few Commands to test after connections

Command	Expected Output	Notes
show dbs	admin 40.00 KiB config 72.00 KiB db 48.00 KiB ds 56.00 KiB laps 8.00 KiB local 72.00 KiB	All Databases are shown
use db	switched to db db	Connect and use db
show collections	student	Show all tables
db.foo.insert({"bar" : "baz"})		Insert a record to collection. Create Collection if not exists
db.foo.batchInsert([{"_id" : 0}, {"_id" : 1}, {"_id" : 2}])		Insert more than one document
db.foo.find()		Print all rows
db.foo.remove()		Remove foo table



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.

```
{"greeting" : "Hello, world!"}
```

Collections:

Collections A collection is a group of documents.

If a document is the MongoDB analog of a row in a relational database, then a collection can be thought of as the analog to a table.

Database:

MongoDB groups collections into databases.

A single instance of MongoDB can host several databases, each grouping together zero or more collections.

A database has its own permissions, and each database is stored in separate files on disk.

A good rule of thumb is to store all data for a single application in the same database.



Datatype

Basically each document will be in JSON format which will be as follows. Where each attributes inside can be of multiple data types.

1.String: This is the most commonly used data type in MongoDB to store data, BSON strings are of UTF-8. So, the drivers for each programming language convert from the string format of the language to UTF-8 while serializing and de-serializing BSON. The string must be a valid UTF-8.

2. Integer: In MongoDB, the integer data type is used to store an integer value. We can store integer data type in two forms 32-bit signed integer and 64 – bit signed integer.

3. Double: The double data type is used to store the floating-point values.

4. Boolean: The boolean data type is used to store either true or false.

5.Null: The null data type is used to store the null value.

6.Array: The Array is the set of values. It can store the same or different data types values in it. In MongoDB, the array is created using square brackets([]).

7.Object: Object data type stores embedded documents. Embedded documents are also known as nested documents. Embedded document or nested documents are those types of documents which contain a document inside another document.

8. Object Id: Whenever we create a new document in the collection MongoDB automatically creates a unique [object id](#) for that document(if the document does not have it). There is an `_id` field in MongoDB for each document. The data which is stored in Id is of hexadecimal format and the length of the id is 12 bytes which consist:

- 4-bytes for Timestamp value.



- 5-bytes for Random values. i.e., 3-bytes for machine Id and 2-bytes for process Id.
- 3- bytes for Counter

You can also create your own id field, but make sure that the value of that id field must be unique.

9.Undefined: This data type stores the undefined values.

10.Binary Data: This datatype is used to store binary data.

11. Date: Date data type stores date. It is a 64-bit integer which represents the number of milliseconds. BSON data type generally supports UTC datetime and it is signed. If the value of the date data type is negative then it represents the dates before 1970. There are various methods to return date, it can be returned either as a string or as a date object. Some method for the date:

- **Date():** It returns the current date in string format.
- **new Date():** Returns a date object. Uses the ISODate() wrapper.
- **new ISODate():** It also returns a date object. Uses the ISODate() wrapper.

12. Min & Max key: Min key compares the value of the lowest BSON element and Max key compares the value against the highest BSON

13.Symbol: This data type similar to the string data type. It is generally not supported by a mongo shell, but if the shell gets a symbol from the database, then it converts this type into a string type.element. Both are internal data types.

14.Regular Expression: This datatype is used to store regular expressions.

15.JavaScript: This datatype is used to store JavaScript code into the document without the scope.



16. JavaScript with Scope: This MongoDB data type store JavaScript data with a scope. This data type is deprecated in MongoDB 4.4.

17. Timestamp: In MongoDB, this data type is used to store a timestamp. It is useful when we modify our data to keep a record and the value of this data type is 64-bit. The value of the timestamp data type is always unique.

18. Decimal: This MongoDB data type store 128-bit decimal-based floating-point value. This data type was introduced in MongoDB version 3.4

“This course was packed with amazing and well-organized content! The project-based approach of this course made it even better to understand concepts faster. Also the instructor in the live classes is really good and knowledgeable.”



Experiment:1

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.

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

Output

```
mongosh mongodb://127.0.0.1
db> db.student.find({home_city:"City 3"});
[
  {
    _id: ObjectId('6666f7312298233688181995'),
    name: 'Student 930',
    age: 25,
    courses: "['English', 'Computer Science', 'Mathematics', 'History']",
    gpa: 3.63,
    home_city: 'City 3',
    blood_group: 'A-',
    is_hotel_resident: true
  },
  {
    _id: ObjectId('6666f731229823368818199a'),
    name: 'Student 536',
    age: 20,
    courses: "['History', 'Physics', 'English', 'Mathematics']",
    gpa: 2.87,
    home_city: 'City 3',
    blood_group: 'O-',
    is_hotel_resident: false
  },
  {
    _id: ObjectId('6666f731229823368818199e'),
    name: 'Student 487',
    age: 21,
    courses: "['History', 'Physics', 'Computer Science']",
    gpa: 2.1,
    home_city: 'City 3',
    blood_group: 'B-',
    is_hotel_resident: true
  },
  {
    _id: ObjectId('6666f73122982336881819a2'),
    name: 'Student 172',
    age: 25,
    courses: "['English', 'History', 'Physics', 'Mathematics']",
    gpa: 2.46,
    home_city: 'City 3',
    blood_group: 'A+',
    is_hotel_resident: false
  },
  {
    _id: ObjectId('6666f73122982336881819c5'),
    name: 'Student 165',
    age: 20,
    courses: "['English', 'History', 'Mathematics', 'Computer Science']",
    gpa: 2.92,
    home_city: 'City 3',
    blood_group: 'B+',
    is_hotel_resident: true
  },
  {
    _id: ObjectId('6666f73122982336881819c6'),
    name: 'Student 959',
    age: 24,
    courses: "['History', 'Computer Science']",
    gpa: 3.43,
    home_city: 'City 3',
    blood_group: 'A+',
    is_hotel_resident: true
  },
  {
    _id: ObjectId('6666f73122982336881819c9'),
    name: 'Student 457',
    age: 18,
```

```
mongosh mongodb://127.0.0.1
{
  _id: ObjectId('6666f7312298233688181a56'),
  name: 'Student 918',
  age: 19,
  courses: "['Physics', 'Computer Science']",
  gpa: 3.92,
  home_city: 'City 3',
  blood_group: 'A+',
  is_hotel_resident: true
},
{
  _id: ObjectId('6666f7312298233688181a68'),
  name: 'Student 654',
  age: 21,
  courses: "['English', 'Mathematics', 'Physics', 'Computer Science']",
  gpa: 3.4,
  home_city: 'City 3',
  blood_group: 'AB-',
  is_hotel_resident: false
},
{
  _id: ObjectId('6666f7312298233688181a6e'),
  name: 'Student 913',
  age: 20,
  courses: "['Computer Science', 'History']",
  gpa: 2.24,
  home_city: 'City 3',
  blood_group: 'AB+',
  is_hotel_resident: true
},
{
  _id: ObjectId('6666f7312298233688181a70'),
  name: 'Student 199',
  age: 21,
  courses: "['Computer Science', 'Mathematics', 'Physics', 'History']",
  gpa: 2.21,
  home_city: 'City 3',
  blood_group: 'O-',
  is_hotel_resident: false
},
{
  _id: ObjectId('6666f7312298233688181a9d'),
  name: 'Student 728',
  age: 24,
  courses: "['Mathematics', 'Physics', 'English']",
  gpa: 3.95,
  home_city: 'City 3',
  blood_group: 'A+',
  is_hotel_resident: true
},
{
  _id: ObjectId('6666f7312298233688181ac0'),
  name: 'Student 917',
  age: 19,
  courses: "['Computer Science', 'Physics', 'English']",
  gpa: 2.89,
  home_city: 'City 3',
  blood_group: 'A-',
  is_hotel_resident: true
},
{
  _id: ObjectId('6666f7312298233688181ael'),
  name: 'Student 289',
  age: 18,
  courses: "['English', 'Mathematics', 'Physics']",
  gpa: 3.64,
```



```

mongosh mongod://127.0.0.
{
  _id: ObjectId('6666f73122982336881819c9'),
  name: 'Student 457',
  age: 18,
  courses: "['Mathematics', 'History', 'English', 'Physics']",
  gpa: 3.97,
  home_city: 'City 3',
  blood_group: 'AB-',
  is_hotel_resident: true
},
{
  _id: ObjectId('6666f73122982336881819d9'),
  name: 'Student 722',
  age: 23,
  courses: "['Physics', 'History', 'Computer Science', 'Mathematics']",
  gpa: 3.75,
  home_city: 'City 3',
  blood_group: 'O+',
  is_hotel_resident: false
},
{
  _id: ObjectId('6666f73122982336881819fe'),
  name: 'Student 422',
  age: 22,
  courses: "['Physics', 'Mathematics']",
  gpa: 2.09,
  home_city: 'City 3',
  blood_group: 'AB+',
  is_hotel_resident: true
},
{
  _id: ObjectId('6666f7312298233688181a3f'),
  name: 'Student 868',
  age: 20,
  courses: "['Computer Science', 'Mathematics']",
  gpa: 2.4,
  home_city: 'City 3',
  blood_group: 'B-',
  is_hotel_resident: false
},
{
  _id: ObjectId('6666f7312298233688181a4d'),
  name: 'Student 859',
  age: 23,
  courses: "['Computer Science', 'Mathematics']",
  gpa: 2.42,
  home_city: 'City 3',
  blood_group: 'O-',
  is_hotel_resident: true
},
{
  _id: ObjectId('6666f7312298233688181a52'),
  name: 'Student 763',
  age: 23,
  courses: "['Mathematics', 'Physics', 'English', 'Computer Science']",
  gpa: 2.37,
  home_city: 'City 3',
  blood_group: 'AB+',
  is_hotel_resident: true
},
{
  _id: ObjectId('6666f7312298233688181a56'),
  name: 'Student 918',
  age: 19,
  courses: "['Physics', 'Computer Science']",
  gpa: 3.92,
}
]

```

```

mongosh mongod://127.0.0.
name: 'Student 917',
age: 19,
courses: "['Computer Science', 'Physics', 'English']",
gpa: 2.89,
home_city: 'City 3',
blood_group: 'A-',
is_hotel_resident: true
},
{
  _id: ObjectId('6666f7312298233688181ae1'),
  name: 'Student 289',
  age: 18,
  courses: "['English', 'Mathematics', 'Physics']",
  gpa: 3.64,
  home_city: 'City 3',
  blood_group: 'B-',
  is_hotel_resident: false
},
{
  _id: ObjectId('6666f7312298233688181ae7'),
  name: 'Student 707',
  age: 18,
  courses: "['Computer Science', 'Physics', 'Mathematics', 'History']",
  gpa: 2.21,
  home_city: 'City 3',
  blood_group: 'O-',
  is_hotel_resident: true
}
]
db> |

```



AND

- Given a Collection you want to FILTER a subset based on multiple conditions.

```
// 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 } }
  ]
});
```

Output :

```
db> db.student.find({ $and:[ {home_city:"City 5"},{blood_group:"A+"}]});
[
  {
    _id: ObjectId('6666f73122982336881819c8'),
    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('6666f7312298233688181ae8'),
    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('6666f7312298233688181b5a'),
    name: 'Student 567',
    age: 22,
    courses: "['Computer Science', 'History', 'English', 'Mathematics']",
    gpa: 2.01,
    home_city: 'City 5',
    blood_group: 'A+',
    is_hotel_resident: true
  }
]
db> |
```



OR

- Given a Collection you want to FILTER a subset based on multiple conditions but Any One is Sufficient

```
// 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 } }
  ]
});
```

Output

```
mongosh mongodb://127.0.0.1
{
  "_id": ObjectId("6666f731229823368818199a"),
  "name": "Student 487",
  "age": 21,
  "courses": ["History", "Physics", "Computer Science"],
  "gpa": 2.1,
  "home_city": "City 3",
  "blood_group": "B-",
  "is_hotel_resident": true
},
{
  "_id": ObjectId("6666f731229823368818199f"),
  "name": "Student 213",
  "age": 18,
  "courses": ["English", "History"],
  "gpa": 2.39,
  "blood_group": "B-",
  "is_hotel_resident": true
},
{
  "_id": ObjectId("6666f73122982336881819a0"),
  "name": "Student 690",
  "age": 22,
  "courses": ["History", "Physics", "Mathematics"],
  "gpa": 2.25,
  "home_city": "City 7",
  "blood_group": "AB-",
  "is_hotel_resident": true
},
{
  "_id": ObjectId("6666f73122982336881819a2"),
  "name": "Student 172",
  "age": 25,
  "courses": ["English", "History", "Physics", "Mathematics"],
  "gpa": 2.46,
  "home_city": "City 3",
  "blood_group": "A+",
  "is_hotel_resident": false
},
{
  "_id": ObjectId("6666f73122982336881819a4"),
  "name": "Student 232",
  "age": 18,
  "courses": ["Computer Science", "Physics", "History", "Mathematics"],
  "gpa": 2.54,
  "home_city": "City 1",
  "blood_group": "B-",
  "is_hotel_resident": true
},
{
  "_id": ObjectId("6666f73122982336881819a5"),
  "name": "Student 328",
  "age": 21,
  "courses": ["Physics", "Computer Science", "English"],
  "gpa": 2.92,
  "home_city": "City 2",
  "blood_group": "AB+",
  "is_hotel_resident": true
},
{
  "_id": ObjectId("6666f73122982336881819a6"),
  "name": "Student 690",
  "age": 24,
  "courses": ["Computer Science", "English", "History"],
  "gpa": 2.71,
  "blood_group": "AB+",
  "is_hotel_resident": false
},
{
  "_id": ObjectId("6666f73122982336881819a7"),
  "name": "Student 499",
  "age": 25,
  "courses": ["Mathematics", "English", "Computer Science", "Physics"],
  "gpa": 2.04,
  "blood_group": "A+",
  "is_hotel_resident": false
},
}
```

```
mongosh mongodb://127.0.0.1
db> db.student.find({'$or':{'is_hotel_resident':true},{gpa:{$lt:3.0}}})
{
  "_id": ObjectId("6666f7312298233688181992"),
  "name": "Student 157",
  "age": 20,
  "courses": ["Physics", "English"],
  "gpa": 2.27,
  "home_city": "City 4",
  "blood_group": "O-",
  "is_hotel_resident": true
},
{
  "_id": ObjectId("6666f7312298233688181993"),
  "name": "Student 316",
  "age": 20,
  "courses": ["Physics", "Computer Science", "Mathematics", "History"],
  "gpa": 2.32,
  "blood_group": "B+",
  "is_hotel_resident": true
},
{
  "_id": ObjectId("6666f7312298233688181998"),
  "name": "Student 563",
  "age": 18,
  "courses": ["Mathematics", "English"],
  "gpa": 2.25,
  "blood_group": "AB+",
  "is_hotel_resident": false
},
{
  "_id": ObjectId("6666f7312298233688181999"),
  "name": "Student 440",
  "age": 21,
  "courses": ["History", "Physics", "Computer Science"],
  "gpa": 2.06,
  "home_city": "City 10",
  "blood_group": "O-",
  "is_hotel_resident": true
},
{
  "_id": ObjectId("6666f731229823368818199a"),
  "name": "Student 536",
  "age": 20,
  "courses": ["History", "Physics", "English", "Mathematics"],
  "gpa": 2.87,
  "home_city": "City 3",
  "blood_group": "O-",
  "is_hotel_resident": false
},
{
  "_id": ObjectId("6666f731229823368818199b"),
  "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
},
{
  "_id": ObjectId("6666f731229823368818199c"),
  "name": "Student 177",
  "age": 23,
  "courses": ["Mathematics", "Computer Science", "Physics"],
  "gpa": 2.52,
  "home_city": "City 10",
  "blood_group": "A+",
  "is_hotel_resident": true
},
{
  "_id": ObjectId("6666f731229823368818199e"),
  "name": "Student 487",
  "age": 21,
  "courses": ["History", "Physics", "Computer Science"],
  "gpa": 2.1,
  "home_city": "City 3",
}
```



```

C:\ mongosh mongodb://127.0.0.1:27021 > use student
> db.students.insertMany([
  {
    blood_group: 'AB-',
    is_hotel_resident: true
  },
  {
    _id: ObjectId('6666f73122982336881819a6'),
    name: 'Student 690',
    age: 24,
    courses: "['Computer Science', 'English', 'History']",
    gpa: 2.71,
    blood_group: 'AB+',
    is_hotel_resident: false
  },
  {
    _id: ObjectId('6666f73122982336881819a7'),
    name: 'Student 499',
    age: 25,
    courses: "['Mathematics', 'English', 'Computer Science', 'Physics']",
    gpa: 2.04,
    blood_group: 'A+',
    is_hotel_resident: false
  },
  {
    _id: ObjectId('6666f73122982336881819a9'),
    name: 'Student 504',
    age: 21,
    courses: "['Physics', 'Computer Science', 'English', 'Mathematics']",
    gpa: 2.42,
    home_city: 'City 2',
    blood_group: 'B+',
    is_hotel_resident: true
  },
  {
    _id: ObjectId('6666f73122982336881819ab'),
    name: 'Student 915',
    age: 22,
    courses: "['Computer Science', 'History', 'Physics', 'English']",
    gpa: 2.87,
    blood_group: 'AB+',
    is_hotel_resident: true
  },
  {
    _id: ObjectId('6666f73122982336881819ac'),
    name: 'Student 127',
    age: 19,
    courses: "['History', 'English', 'Computer Science', 'Mathematics']",
    gpa: 2.56,
    home_city: 'City 10',
    blood_group: 'AB+',
    is_hotel_resident: false
  },
  {
    _id: ObjectId('6666f73122982336881819ae'),
    name: 'Student 770',
    age: 24,
    courses: "['History', 'Computer Science', 'Mathematics', 'English']",
    gpa: 2.98,
    blood_group: 'B-',
    is_hotel_resident: false
  },
  {
    _id: ObjectId('6666f73122982336881819af'),
    name: 'Student 367',
    age: 25,
    courses: "['History', 'Physics', 'Computer Science']",
    gpa: 2.61,
    home_city: 'City 1',
    blood_group: 'AB+',
    is_hotel_resident: true
  }
])
db>

```



CRUD

- C - Create / Insert
- R - Remove
- U - update
- D - Delete

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

Update

Given a collection you want to update new student update is used.

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

Output :

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



Delete

To delete particular detail delete is used.

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

Output :

```
db> db.student.deleteOne({name:"Jone Doe"});
{ acknowledged: true, deletedCount: 0 }
db> |
```

Update Many

To update many items at a time this updateMany is used.

```
// 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 } });
```

Output :

```
db> db.studdent.updateMany({gpa:{$lt:3.0}},{$inc:{gpa:0.5}});
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 0,
  modifiedCount: 0,
  upsertedCount: 0
}
db> |
```



Delete Many

To update delete many items at a time this deleteMany is used.

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

Output :

```
db> db.student.deleteMany({ is_hostel_resident:false});  
{ acknowledged: true, deletedCount: 0 }  
db> |
```



Experiment:2

Projection

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

```
// Get only the name and gpa for all students
db.students.find({}, { name: 1, gpa: 1 });

// Exclude the "_id" field from all queries by default
db.students.find({}, { _id: 0 });
```

Output :

```
db> db.student.find({}, {name:1,gpa:1});
[
  {
    _id: ObjectId('6666f7312298233688181991'),
    name: 'Student 948',
    gpa: 3.44
  },
  {
    _id: ObjectId('6666f7312298233688181992'),
    name: 'Student 157',
    gpa: 2.27
  },
  {
    _id: ObjectId('6666f7312298233688181993'),
    name: 'Student 316',
    gpa: 2.32
  },
  {
    _id: ObjectId('6666f7312298233688181994'),
    name: 'Student 346',
    gpa: 3.31
  },
  {
    _id: ObjectId('6666f7312298233688181995'),
    name: 'Student 930',
    gpa: 3.63
  },
  {
    _id: ObjectId('6666f7312298233688181996'),
    name: 'Student 305',
    gpa: 3.4
  },
  {
    _id: ObjectId('6666f7312298233688181997'),
    name: 'Student 268',
    gpa: 3.98
  },
  {
    _id: ObjectId('6666f7312298233688181998'),
    name: 'Student 563',
    gpa: 2.25
  },
  {
    _id: ObjectId('6666f7312298233688181999'),
```

```
{
  _id: ObjectId('6666f731229823368818199d'),
  name: 'Student 871',
  gpa: 3.2
},
{
  _id: ObjectId('6666f731229823368818199e'),
  name: 'Student 487',
  gpa: 2.1
},
{
  _id: ObjectId('6666f731229823368818199f'),
  name: 'Student 213',
  gpa: 2.39
},
{
  _id: ObjectId('6666f73122982336881819a0'),
  name: 'Student 690',
  gpa: 2.25
},
{
  _id: ObjectId('6666f73122982336881819a1'),
  name: 'Student 368',
  gpa: 3.91
},
{
  _id: ObjectId('6666f73122982336881819a2'),
  name: 'Student 172',
  gpa: 2.46
},
{
  _id: ObjectId('6666f73122982336881819a3'),
  name: 'Student 647',
  gpa: 3.43
},
{
  _id: ObjectId('6666f73122982336881819a4'),
  name: 'Student 232',
  gpa: 2.54
}
]
Type "it" for more
db> $|
```

- Use the projection document as the second argument to the `find` method.
- Include field names with a value of `1` to specify fields to be returned.
- Omit fields or set them to `0` to exclude them from the results.



Get Selected Attributes

- Given a Collection you want to FILTER a subset of attributes. That is the place Projection is used.

```
// Get only the name and age for all students
db.students.find({}, { name: 1, age: 1 });
```

Output :

```
db> db.candidates.find({}, {name:1,age:1,gpa:1});
[
  {
    _id: ObjectId('6668764273cc8ecde68c2e66'),
    name: 'Alice Smith',
    age: 20,
    gpa: 3.4
  },
  {
    _id: ObjectId('6668764273cc8ecde68c2e67'),
    name: 'Bob Johnson',
    age: 22,
    gpa: 3.8
  },
  {
    _id: ObjectId('6668764273cc8ecde68c2e68'),
    name: 'Charlie Lee',
    age: 19,
    gpa: 3.2
  },
  {
    _id: ObjectId('6668764273cc8ecde68c2e69'),
    name: 'Emily Jones',
    age: 21,
    gpa: 3.6
  },
  {
    _id: ObjectId('6668764273cc8ecde68c2e6a'),
    name: 'David Williams',
    age: 23,
    gpa: 3
  },
  {
    _id: ObjectId('6668764273cc8ecde68c2e6b'),
    name: 'Fatima Brown',
    age: 18,
    gpa: 3.5
  },
  {
    _id: ObjectId('6668764273cc8ecde68c2e6c'),
    name: 'Gabriel Miller',
    age: 24,
    gpa: 3.9
  },
  {
    _id: ObjectId('6668764273cc8ecde68c2e6d'),
    name: 'Hannah Garcia',
    age: 20,
    gpa: 3.3
  },
  {
    _id: ObjectId('6668764273cc8ecde68c2e6e'),
    name: 'Isaac Clark',
    age: 22,
    gpa: 3.7
  },
  {
    _id: ObjectId('6668764273cc8ecde68c2e6f'),
    name: 'Jessica Moore',
    age: 19,
    gpa: 3.1
  },
  {
    _id: ObjectId('6668764273cc8ecde68c2e70'),
    name: 'Kevin Lewis',
    age: 21,
    gpa: 4
  },
  {
    _id: ObjectId('6668764273cc8ecde68c2e71'),
    name: 'Lily Robinson',
    age: 23,
    gpa: 3.5
  }
]
```

```
db> db.student.find({}, {name:1,age:1});
[
  {
    _id: ObjectId('6666f7312298233688181991'),
    name: 'Student 948',
    age: 19
  },
  {
    _id: ObjectId('6666f7312298233688181992'),
    name: 'Student 157',
    age: 20
  },
  {
    _id: ObjectId('6666f7312298233688181993'),
    name: 'Student 316',
    age: 20
  },
  {
    _id: ObjectId('6666f7312298233688181994'),
    name: 'Student 346',
    age: 25
  },
  {
    _id: ObjectId('6666f7312298233688181995'),
    name: 'Student 930',
    age: 25
  },
  {
    _id: ObjectId('6666f7312298233688181996'),
    name: 'Student 385',
    age: 24
  },
  {
    _id: ObjectId('6666f7312298233688181997'),
    name: 'Student 268',
    age: 21
  },
  {
    _id: ObjectId('6666f7312298233688181998'),
    name: 'Student 563',
    age: 18
  },
  {
    _id: ObjectId('6666f7312298233688181999'),
    name: 'Student 440',
    age: 21
  },
  {
    _id: ObjectId('6666f731229823368818199a'),
    name: 'Student 536',
    age: 20
  },
  {
    _id: ObjectId('6666f731229823368818199b'),
    name: 'Student 256',
    age: 19
  },
  {
    _id: ObjectId('6666f731229823368818199c'),
    name: 'Student 177',
    age: 23
  },
  {
    _id: ObjectId('6666f731229823368818199d'),
    name: 'Student 871',
    age: 22
  },
  {
    _id: ObjectId('6666f731229823368818199e'),
    name: 'Student 487',
    age: 21
  },
  {
    _id: ObjectId('6666f731229823368818199f'),
    name: 'Student 213',
    age: 18
  },
  {
    _id: ObjectId('6666f73122982336881819a0'),
    name: 'Student 690',
    age: 22
  },
  {
    _id: ObjectId('6666f73122982336881819a1'),
    name: 'Student 368',
    age: 20
  },
  {
    _id: ObjectId('6666f73122982336881819a2'),
    name: 'Student 172',
    age: 25
  },
  {
    _id: ObjectId('6666f73122982336881819a3'),
    name: 'Student 647',
    age: 21
  },
  {
    _id: ObjectId('6666f73122982336881819a4'),
    name: 'Student 232',
    age: 18
  }
]
```

```
db> db.candidates.find({}, {name:1,age:1,gpa:1});
[
  {
    _id: ObjectId('6668764273cc8ecde68c2e66'),
    name: 'Alice Smith',
    age: 20,
    gpa: 3.4
  },
  {
    _id: ObjectId('6668764273cc8ecde68c2e67'),
    name: 'Bob Johnson',
    age: 22,
    gpa: 3.8
  },
  {
    _id: ObjectId('6668764273cc8ecde68c2e68'),
    name: 'Charlie Lee',
    age: 19,
    gpa: 3.2
  },
  {
    _id: ObjectId('6668764273cc8ecde68c2e69'),
    name: 'Emily Jones',
    age: 21,
    gpa: 3.6
  },
  {
    _id: ObjectId('6668764273cc8ecde68c2e6a'),
    name: 'David Williams',
    age: 23,
    gpa: 3
  },
  {
    _id: ObjectId('6668764273cc8ecde68c2e6b'),
    name: 'Fatima Brown',
    age: 18,
    gpa: 3.5
  },
  {
    _id: ObjectId('6668764273cc8ecde68c2e6c'),
    name: 'Gabriel Miller',
    age: 24,
    gpa: 3.9
  },
  {
    _id: ObjectId('6668764273cc8ecde68c2e6d'),
    name: 'Hannah Garcia',
    age: 20,
    gpa: 3.3
  },
  {
    _id: ObjectId('6668764273cc8ecde68c2e6e'),
    name: 'Isaac Clark',
    age: 22,
    gpa: 3.7
  },
  {
    _id: ObjectId('6668764273cc8ecde68c2e6f'),
    name: 'Jessica Moore',
    age: 19,
    gpa: 3.1
  },
  {
    _id: ObjectId('6668764273cc8ecde68c2e70'),
    name: 'Kevin Lewis',
    age: 21,
    gpa: 4
  },
  {
    _id: ObjectId('6668764273cc8ecde68c2e71'),
    name: 'Lily Robinson',
    age: 23,
    gpa: 3.5
  }
]
```

```
{
  _id: ObjectId('6666f73122982336881819a1'),
  name: 'Student 368',
  age: 20
},
{
  _id: ObjectId('6666f73122982336881819a2'),
  name: 'Student 172',
  age: 25
},
{
  _id: ObjectId('6666f73122982336881819a3'),
  name: 'Student 647',
  age: 21
},
{
  _id: ObjectId('6666f73122982336881819a4'),
  name: 'Student 232',
  age: 18
}
]
Type "it" for more
db> |
```



Ignore Attributes

In MongoDB, the "ignore" attribute is not a built-in feature or keyword. However, you might be referring to the concept of ignoring fields during document validation or during query projections.

```
// Get all student data but exclude the _id field
db.students.find({}, { _id: 0 });
```

Output :

```
db> db.student.find({}, {_id:0});
{
  name: 'Student 900',
  age: 19,
  courses: ['English', 'Computer Science', 'Physics', 'Mathematics'],
  gpa: 3.46,
  home_city: 'City 2',
  blood_group: 'B+',
  is_hotel_resident: true
},
{
  name: 'Student 157',
  age: 20,
  courses: ['Physics', 'English'],
  gpa: 2.27,
  home_city: 'City 4',
  blood_group: 'B-',
  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 365',
  age: 20,
  courses: ['Mathematics', 'History', 'English'],
  gpa: 3.21,
  home_city: 'City 8',
  blood_group: 'O+',
  is_hotel_resident: true
},
{
  name: 'Student 530',
  age: 25,
  courses: ['English', 'Computer Science', 'Mathematics', 'History'],
  gpa: 3.63,
  home_city: 'City 3',
  blood_group: 'A+',
  is_hotel_resident: true
},
{
  name: 'Student 305',
  age: 24,
  courses: ['History', 'Physics', 'Computer Science', 'Mathematics'],
  gpa: 3.4,
  home_city: 'City 6',
  blood_group: 'O+',
  is_hotel_resident: true
},
{
  name: 'Student 268',
  age: 21,
  courses: ['Mathematics', 'History', 'Physics'],
  gpa: 3.86,
  blood_group: 'A+',
  is_hotel_resident: false
},
{
  name: 'Student 563',
  age: 18,
  courses: ['Mathematics', 'English'],
  gpa: 2.25,
  blood_group: 'A+',
  is_hotel_resident: false
},
{
  name: 'Student 690',
  age: 21,
  courses: ['History', 'Physics', 'Computer Science'],
  gpa: 2.86,
  home_city: 'City 10',
  blood_group: 'B-',
  is_hotel_resident: true
},
{
  name: 'Student 536',
  age: 20,
  courses: ['History', 'Physics', 'English', 'Mathematics'],
  gpa: 2.87,
  home_city: 'City 3',
  blood_group: 'O+',
  is_hotel_resident: false
},
{
  name: 'Student 256',
  age: 19,
  courses: ['Computer Science', 'Mathematics', 'History', 'English'],
  gpa: 2.96,
  home_city: 'City 1',
  blood_group: 'B+',
  is_hotel_resident: true
},
{
  name: 'Student 177',
  age: 23,
  courses: ['Mathematics', 'Computer Science', 'Physics'],
  gpa: 2.52,
  home_city: 'City 10',
  blood_group: 'A+',
  is_hotel_resident: true
},
{
  name: 'Student 871',
  age: 22,
  courses: ['Mathematics', 'Computer Science'],
  gpa: 3.2,
  blood_group: 'A+',
  is_hotel_resident: false
},
{
  name: 'Student 487',
  age: 21,
  courses: ['History', 'Physics', 'Computer Science'],
  gpa: 2.1,
  home_city: 'City 3',
  blood_group: 'A+',
  is_hotel_resident: true
},
{
  name: 'Student 213',
  age: 18,
  courses: ['English', 'History'],
  gpa: 2.39,
  blood_group: 'B-',
  is_hotel_resident: true
},
{
  name: 'Student 699',
  age: 22,
  courses: ['History', 'Physics', 'Mathematics'],
  gpa: 2.25,
  home_city: 'City 7',
  blood_group: 'A-',
  is_hotel_resident: true
},
{
  name: 'Student 368',
  age: 20,
  courses: ['English', 'History', 'Physics', 'Computer Science'],
  gpa: 3.91,
  home_city: 'City 9',
  blood_group: 'B-',
  is_hotel_resident: false
},
{
  name: 'Student 172',
  age: 20,
  courses: ['English', 'History', 'Physics', 'Mathematics'],
  gpa: 2.46,
  home_city: 'City 3',
  blood_group: 'A+',
  is_hotel_resident: false
},
{
  name: 'Student 607',
  age: 21,
  courses: ['English', 'Physics'],
  gpa: 3.49,
  home_city: 'City 4',
  blood_group: 'A+',
  is_hotel_resident: false
}
```

```
{
  name: 'Student 268',
  age: 21,
  courses: ['Mathematics', 'History', 'Physics'],
  gpa: 3.86,
  blood_group: 'A+',
  is_hotel_resident: false
},
{
  name: 'Student 563',
  age: 18,
  courses: ['Mathematics', 'English'],
  gpa: 2.25,
  blood_group: 'A+',
  is_hotel_resident: false
},
{
  name: 'Student 690',
  age: 21,
  courses: ['History', 'Physics', 'Computer Science'],
  gpa: 2.86,
  home_city: 'City 10',
  blood_group: 'B-',
  is_hotel_resident: true
},
{
  name: 'Student 536',
  age: 20,
  courses: ['History', 'Physics', 'English', 'Mathematics'],
  gpa: 2.87,
  home_city: 'City 3',
  blood_group: 'O+',
  is_hotel_resident: false
},
{
  name: 'Student 256',
  age: 19,
  courses: ['Computer Science', 'Mathematics', 'History', 'English'],
  gpa: 2.96,
  home_city: 'City 1',
  blood_group: 'B+',
  is_hotel_resident: true
},
{
  name: 'Student 177',
  age: 23,
  courses: ['Mathematics', 'Computer Science', 'Physics'],
  gpa: 2.52,
  home_city: 'City 10',
  blood_group: 'A+',
  is_hotel_resident: true
},
{
  name: 'Student 871',
  age: 22,
  courses: ['Mathematics', 'Computer Science'],
  gpa: 3.2,
  blood_group: 'A+',
  is_hotel_resident: false
},
{
  name: 'Student 487',
  age: 21,
  courses: ['History', 'Physics', 'Computer Science'],
  gpa: 2.1,
  home_city: 'City 3',
  blood_group: 'A+',
  is_hotel_resident: true
},
{
  name: 'Student 213',
  age: 18,
  courses: ['English', 'History'],
  gpa: 2.39,
  blood_group: 'B-',
  is_hotel_resident: true
},
{
  name: 'Student 699',
  age: 22,
  courses: ['History', 'Physics', 'Mathematics'],
  gpa: 2.25,
  home_city: 'City 7',
  blood_group: 'A-',
  is_hotel_resident: true
},
{
  name: 'Student 368',
  age: 20,
  courses: ['English', 'History', 'Physics', 'Computer Science'],
  gpa: 3.91,
  home_city: 'City 9',
  blood_group: 'B-',
  is_hotel_resident: false
},
{
  name: 'Student 172',
  age: 20,
  courses: ['English', 'History', 'Physics', 'Mathematics'],
  gpa: 2.46,
  home_city: 'City 3',
  blood_group: 'A+',
  is_hotel_resident: false
},
{
  name: 'Student 607',
  age: 21,
  courses: ['English', 'Physics'],
  gpa: 3.49,
  home_city: 'City 4',
  blood_group: 'A+',
  is_hotel_resident: false
}
```

```
{
  name: 'Student 871',
  age: 22,
  courses: ['Mathematics', 'Computer Science'],
  gpa: 3.2,
  blood_group: 'A+',
  is_hotel_resident: false
},
{
  name: 'Student 487',
  age: 21,
  courses: ['History', 'Physics', 'Computer Science'],
  gpa: 2.1,
  home_city: 'City 3',
  blood_group: 'A+',
  is_hotel_resident: true
},
{
  name: 'Student 213',
  age: 18,
  courses: ['English', 'History'],
  gpa: 2.39,
  blood_group: 'B-',
  is_hotel_resident: true
},
{
  name: 'Student 699',
  age: 22,
  courses: ['History', 'Physics', 'Mathematics'],
  gpa: 2.25,
  home_city: 'City 7',
  blood_group: 'A-',
  is_hotel_resident: true
},
{
  name: 'Student 368',
  age: 20,
  courses: ['English', 'History', 'Physics', 'Computer Science'],
  gpa: 3.91,
  home_city: 'City 9',
  blood_group: 'B-',
  is_hotel_resident: false
},
{
  name: 'Student 172',
  age: 20,
  courses: ['English', 'History', 'Physics', 'Mathematics'],
  gpa: 2.46,
  home_city: 'City 3',
  blood_group: 'A+',
  is_hotel_resident: false
},
{
  name: 'Student 607',
  age: 21,
  courses: ['English', 'Physics'],
  gpa: 3.49,
  home_city: 'City 4',
  blood_group: 'A+',
  is_hotel_resident: false
}
```

```
{
  name: 'Student 647',
  age: 21,
  courses: "['English', 'Physics']",
  gpa: 3.43,
  home_city: 'City 6',
  blood_group: 'A+',
  is_hotel_resident: true
},
{
  name: 'Student 232',
  age: 18,
  courses: "['Computer Science', 'Physics', 'History', 'Mathematics']",
  gpa: 2.54,
  home_city: 'City 1',
  blood_group: 'B-',
  is_hotel_resident: true
}
]
Type "it" for more
db> |
```



Retrieving Specific Fields from Nested Objects

```
// Get student name and only the first course from the courses array
db.students.find({}, {
  name: 1,
  courses: { $slice: 1 }
});
```

Output :

```
db> db.student.find({}, {name:1,courses:{$slice:1}});
[
  {
    _id: ObjectId('6666f7312298233688181991'),
    name: 'Student 948',
    courses: ["'English'", 'Computer Science', 'Physics', 'Mathematics']
  },
  {
    _id: ObjectId('6666f7312298233688181992'),
    name: 'Student 157',
    courses: ["'Physics'", 'English']
  },
  {
    _id: ObjectId('6666f7312298233688181993'),
    name: 'Student 316',
    courses: ["'Physics'", 'Computer Science', 'Mathematics', 'History']
  },
  {
    _id: ObjectId('6666f7312298233688181994'),
    name: 'Student 346',
    courses: ["'Mathematics'", 'History', 'English']
  },
  {
    _id: ObjectId('6666f7312298233688181995'),
    name: 'Student 930',
    courses: ["'English'", 'Computer Science', 'Mathematics', 'History']
  },
  {
    _id: ObjectId('6666f7312298233688181996'),
    name: 'Student 305',
    courses: ["'History'", 'Physics', 'Computer Science', 'Mathematics']
  },
  {
    _id: ObjectId('6666f7312298233688181997'),
    name: 'Student 268',
    courses: ["'Mathematics'", 'History', 'Physics']
  },
  {
    _id: ObjectId('6666f7312298233688181998'),
    name: 'Student 563',
    courses: ["'Mathematics'", 'English']
  },
  {
    _id: ObjectId('6666f7312298233688181999'),
    name: 'Student 440',
    courses: ["'History'", 'Physics', 'Computer Science']
  },
  {
    _id: ObjectId('6666f731229823368818199a'),
    name: 'Student 536',
    courses: ["'History'", 'Physics', 'English', 'Mathematics']
  },
  {
    _id: ObjectId('6666f731229823368818199b'),
    name: 'Student 256',
    courses: ["'Computer Science'", 'Mathematics', 'History', 'English']
  },
  {
    _id: ObjectId('6666f731229823368818199c'),
    name: 'Student 177',
    courses: ["'Mathematics'", 'Computer Science', 'Physics']
  },
  {
    _id: ObjectId('6666f731229823368818199d'),
    name: 'Student 871',
    courses: ["'Mathematics'", 'Computer Science']
  },
  {
    _id: ObjectId('6666f731229823368818199e'),
    name: 'Student 487',
    courses: ["'History'", 'Physics', 'Computer Science']
  },
  {
    _id: ObjectId('6666f731229823368818199f'),
    name: 'Student 213',
    courses: ["'English'", 'History']
  },
  {
    _id: ObjectId('6666f731229823368818199a0'),
    name: 'Student 690',
    courses: ["'History'", 'Physics', 'Mathematics']
  },
  {
    _id: ObjectId('6666f731229823368818199a1'),
    name: 'Student 368',
    courses: ["'English'", 'History', 'Physics', 'Computer Science']
  },
  {
    _id: ObjectId('6666f731229823368818199a2'),
    name: 'Student 172',
    courses: ["'English'", 'History', 'Physics', 'Mathematics']
  },
  {
    _id: ObjectId('6666f731229823368818199a3'),
    name: 'Student 647',
    courses: ["'English'", 'Physics']
  },
  {
    _id: ObjectId('6666f731229823368818199a4'),
    name: 'Student 232',
    courses: ["'Computer Science'", 'Physics', 'History', 'Mathematics']
  }
]
```

```
Type "it" for more
db> |
```

Benefits of Projection

- Reduces 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)`

Get First 5 document

```
// Assuming you have already executed a query on the student collection
// Limit the results to the first 5 documents
db.students.find({}, { _id: 0 }).limit(5);
```

Output :

```
db> db.student.find({}, {_id:0}).limit(5);
[
  {
    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
  },
  {
    name: 'Student 930',
    age: 25,
    courses: "['English', 'Computer Science', 'Mathematics', 'History']",
    gpa: 3.63,
    home_city: 'City 3',
    blood_group: 'A-',
    is_hotel_resident: true
  }
]
db> |
```



Limiting Results

```
// Find all students with GPA greater than 3.5 and limit to 2 documents
db.students.find({ gpa: { $gt: 3.5 } }, { _id: 0 }).limit(2);
```

Output :

```
db> db.student.find({gpa:{$gt:3.5}}, {_id:0}).limit(2);
[
  {
    name: 'Student 930',
    age: 25,
    courses: "['English', 'Computer Science', 'Mathematics', 'History']",
    gpa: 3.63,
    home_city: 'City 3',
    blood_group: 'A-',
    is_hotel_resident: true
  },
  {
    name: 'Student 268',
    age: 21,
    courses: "['Mathematics', 'History', 'Physics']",
    gpa: 3.98,
    blood_group: 'A+',
    is_hotel_resident: false
  }
]
db> |
```

To get Top 5 Results

```
// Sort documents in descending order by _id and limit to 5
db.students.find({}, { _id: 0 }).sort({ _id: -1 }).limit(5);
```

Output :

```
db> db.student.find({}, {_id:0}).sort({_id:1}).limit(5);
[
  {
    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
  },
  {
    name: 'Student 930',
    age: 25,
    courses: "['English', 'Computer Science', 'Mathematics', 'History']",
    gpa: 3.63,
    home_city: 'City 3',
    blood_group: 'A-',
    is_hotel_resident: true
  }
]
db> |
```



Experiment:3

Comparison gt lt

```
// Find all students with age greater than 20
db.students.find({ age: { $gt: 20 } });
```

Output :

```
db> db.student.find({}, [_id:0]).sort({_id:1}).limit(5);
{
  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 318',
  age: 20,
  courses: ['Physics', 'Computer Science', 'Mathematics', 'History'],
  gpa: 2.32,
  blood_group: 'B+',
  is_hotel_resident: true
},
{
  name: 'Student 945',
  age: 25,
  courses: ['Mathematics', 'History', 'English'],
  gpa: 3.31,
  home_city: 'City 8',
  blood_group: 'O+',
  is_hotel_resident: true
},
{
  name: 'Student 938',
  age: 25,
  courses: ['English', 'Computer Science', 'Mathematics', 'History'],
  gpa: 3.63,
  home_city: 'City 3',
  blood_group: 'A+',
  is_hotel_resident: true
}
db> db.student.find({age:{$gt:20}});
{
  _id: ObjectId('6666f7312298233688181994'),
  name: 'Student 348',
  age: 20,
  courses: ['Mathematics', 'History', 'English'],
  gpa: 3.31,
  home_city: 'City 8',
  blood_group: 'O+',
  is_hotel_resident: true
}
```

```
{
  _id: ObjectId('6666f7312298233688181995'),
  name: 'Student 930',
  age: 25,
  courses: ['English', 'Computer Science', 'Mathematics', 'History'],
  gpa: 3.63,
  home_city: 'City 3',
  blood_group: 'A+',
  is_hotel_resident: true
},
{
  _id: ObjectId('6666f7312298233688181996'),
  name: 'Student 305',
  age: 24,
  courses: ['History', 'Physics', 'Computer Science', 'Mathematics'],
  gpa: 3.44,
  home_city: 'City 6',
  blood_group: 'O+',
  is_hotel_resident: true
},
{
  _id: ObjectId('6666f7312298233688181997'),
  name: 'Student 268',
  age: 21,
  courses: ['Mathematics', 'History', 'Physics'],
  gpa: 3.98,
  blood_group: 'A+',
  is_hotel_resident: false
},
{
  _id: ObjectId('6666f7312298233688181999'),
  name: 'Student 488',
  age: 21,
  courses: ['History', 'Physics', 'Computer Science'],
  gpa: 2.86,
  home_city: 'City 10',
  blood_group: 'O+',
  is_hotel_resident: true
},
{
  _id: ObjectId('6666f731229823368818199c'),
  name: 'Student 177',
  age: 23,
  courses: ['Mathematics', 'Computer Science', 'Physics'],
  gpa: 2.52,
  home_city: 'City 18',
  blood_group: 'A+',
  is_hotel_resident: true
},
{
  _id: ObjectId('6666f731229823368818199d'),
  name: 'Student 171',
  age: 22,
  courses: ['Mathematics', 'Computer Science'],
  blood_group: 'A+',
  is_hotel_resident: false
}
```

```
{
  _id: ObjectId('6666f731229823368818199e'),
  name: 'Student 487',
  age: 21,
  courses: ['History', 'Physics', 'Computer Science'],
  gpa: 2.1,
  home_city: 'City 3',
  blood_group: 'B+',
  is_hotel_resident: true
},
{
  _id: ObjectId('6666f73122982336881819a'),
  name: 'Student 690',
  age: 22,
  courses: ['History', 'Physics', 'Mathematics'],
  gpa: 2.25,
  home_city: 'City 7',
  blood_group: 'AB+',
  is_hotel_resident: true
},
{
  _id: ObjectId('6666f73122982336881819a2'),
  name: 'Student 172',
  age: 25,
  courses: ['English', 'History', 'Physics', 'Mathematics'],
  gpa: 2.86,
  home_city: 'City 3',
  blood_group: 'A+',
  is_hotel_resident: false
},
{
  _id: ObjectId('6666f73122982336881819a3'),
  name: 'Student 687',
  age: 21,
  courses: ['English', 'Physics'],
  gpa: 3.43,
  home_city: 'City 6',
  blood_group: 'A+',
  is_hotel_resident: true
},
{
  _id: ObjectId('6666f73122982336881819a5'),
  name: 'Student 328',
  age: 21,
  courses: ['Physics', 'Computer Science', 'English'],
  gpa: 2.92,
  home_city: 'City 2',
  blood_group: 'AB+',
  is_hotel_resident: true
},
{
  _id: ObjectId('6666f73122982336881819a6'),
  name: 'Student 690',
  age: 24,
  courses: ['Computer Science', 'English', 'History'],
  gpa: 2.71,
  blood_group: 'AB+',
  is_hotel_resident: false
}
```

```
{
  age: 21,
  courses: ['Computer Science', 'Physics', 'Mathematics', 'History'],
  gpa: 3.97,
  blood_group: 'A+',
  is_hotel_resident: true
},
{
  _id: ObjectId('6666f73122982336881819a9'),
  name: 'Student 584',
  age: 21,
  courses: ['Physics', 'Computer Science', 'English', 'Mathematics'],
  gpa: 2.42,
  home_city: 'City 3',
  blood_group: 'B+',
  is_hotel_resident: true
},
{
  _id: ObjectId('6666f73122982336881819ab'),
  name: 'Student 815',
  age: 22,
  courses: ['Computer Science', 'History', 'Physics', 'English'],
  gpa: 2.87,
  blood_group: 'AB+',
  is_hotel_resident: true
},
{
  _id: ObjectId('6666f73122982336881819ad'),
  name: 'Student 239',
  age: 21,
  courses: ['English', 'Mathematics', 'Computer Science', 'Physics'],
  gpa: 3.75,
  home_city: 'City 7',
  blood_group: 'A+',
  is_hotel_resident: false
},
{
  _id: ObjectId('6666f73122982336881819ae'),
  name: 'Student 778',
  age: 20,
  courses: ['History', 'Computer Science', 'Mathematics', 'English'],
  gpa: 2.98,
  blood_group: 'B+',
  is_hotel_resident: false
},
{
  _id: ObjectId('6666f73122982336881819af'),
  name: 'Student 387',
  age: 25,
  courses: ['History', 'Physics', 'Computer Science'],
  gpa: 2.61,
  home_city: 'City 1',
  blood_group: 'AB+',
  is_hotel_resident: true
}
Type "it" for more
```



AND operator

```
// Find students from "City 2" with blood group "B+"
db.students.find({
  $and: [
    { home_city: "City 2" },
    { blood_group: "B+" }
  ]
});
```

Output :

```
db> db.student.find({$and:[{home_city:"City 2"},{blood_group:"B+"}]});
[
  {
    _id: ObjectId('6666f7312298236881819a9'),
    name: 'Student 584',
    age: 21,
    courses: ['Physics', 'Computer Science', 'English', 'Mathematics'],
    gpa: 2.42,
    home_city: 'City 2',
    blood_group: 'B+',
    is_hotel_resident: true
  },
  {
    _id: ObjectId('6666f7312298236881819a8'),
    name: 'Student 367',
    age: 19,
    courses: ['English', 'Physics', 'History', 'Mathematics'],
    gpa: 2.81,
    home_city: 'City 2',
    blood_group: 'B+',
    is_hotel_resident: false
  },
  {
    _id: ObjectId('6666f731229823688181a6c'),
    name: 'Student 255',
    age: 21,
    courses: ['English', 'Physics'],
    gpa: 1.85,
    home_city: 'City 2',
    blood_group: 'B+',
    is_hotel_resident: false
  },
  {
    _id: ObjectId('6666f731229823688181a70'),
    name: 'Student 281',
    age: 18,
    courses: ['History', 'Mathematics', 'Physics', 'Computer Science'],
    gpa: 2.2,
    home_city: 'City 2',
    blood_group: 'B+',
    is_hotel_resident: false
  },
  {
    _id: ObjectId('6666f731229823688181a6c'),
    name: 'Student 229',
    age: 18,
    courses: ['History', 'Physics'],
    gpa: 2.89,
    home_city: 'City 2',
    blood_group: 'B+',
    is_hotel_resident: false
  },
  {
    _id: ObjectId('6666f731229823688181af1'),
    name: 'Student 303',
    age: 20,
    courses: ['Physics', 'English'],
    gpa: 3.88,
    home_city: 'City 2',
    blood_group: 'B+',
    is_hotel_resident: false
  },
  {
    _id: ObjectId('6666f731229823688181b57'),
    name: 'Student 872',
    age: 20,
    courses: ['English', 'Mathematics', 'History'],
    gpa: 3.36,
    home_city: 'City 2',
    blood_group: 'B+',
    is_hotel_resident: true
  }
]
db> |
```



OR operator

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

Output :

```
db> db.student.find({$or:[{is_hotel_resident:true},{gpa:{$lt:3.0}}]});
{
  _id: ObjectId('6666f7312298233688181992'),
  name: 'Student 157',
  age: 20,
  courses: ['Physics', 'English'],
  gpa: 2.27,
  home_city: 'City 4',
  blood_group: 'O-',
  is_hotel_resident: true
},
{
  _id: ObjectId('6666f7312298233688181993'),
  name: 'Student 316',
  age: 20,
  courses: ['Physics', 'Computer Science', 'Mathematics', 'History'],
  gpa: 2.32,
  blood_group: 'B+',
  is_hotel_resident: true
},
{
  _id: ObjectId('6666f7312298233688181998'),
  name: 'Student 563',
  age: 18,
  courses: ['Mathematics', 'English'],
  gpa: 2.25,
  blood_group: 'AB+',
  is_hotel_resident: false
},
{
  _id: ObjectId('6666f7312298233688181999'),
  name: 'Student 440',
  age: 21,
  courses: ['History', 'Physics', 'Computer Science'],
  gpa: 2.86,
  home_city: 'City 10',
  blood_group: 'O-',
  is_hotel_resident: true
},
{
  _id: ObjectId('6666f731229823368818199a'),
  name: 'Student 536',
  age: 20,
  courses: ['History', 'Physics', 'English', 'Mathematics'],
  gpa: 2.87,
  home_city: 'City 3',
  blood_group: 'O-',
  is_hotel_resident: false
},
{
  _id: ObjectId('6666f731229823368818199b'),
  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
},
{
  _id: ObjectId('6666f731229823368818199c'),
  name: 'Student 177',
  age: 23,
  courses: ['Mathematics', 'Computer Science', 'Physics'],
  gpa: 2.52,
  home_city: 'City 10',
  blood_group: 'A+',
  is_hotel_resident: true
},
{
  _id: ObjectId('6666f731229823368818199e'),
  name: 'Student 487',
  age: 21,
  courses: ['History', 'Physics', 'Computer Science'],
  gpa: 2.11,
  home_city: 'City 3',
  blood_group: 'B-',
  is_hotel_resident: true
},
{
  _id: ObjectId('6666f731229823368818199f'),
  name: 'Student 213',
  age: 18,
  courses: ['English', 'History'],
  gpa: 2.39,
  blood_group: 'B-',
  is_hotel_resident: true
},
{
  _id: ObjectId('6666f73122982336881819a0'),
  name: 'Student 690',
  age: 22,
  courses: ['History', 'Physics', 'Mathematics'],
  gpa: 2.25,
  home_city: 'City 7',
  blood_group: 'AB-',
  is_hotel_resident: true
},
{
  _id: ObjectId('6666f73122982336881819a2'),
  name: 'Student 172',
  age: 25,
```

```
,
  _id: ObjectId('6666f73122982336881819a2'),
  name: 'Student 172',
  age: 25,
  courses: ['English', 'History', 'Physics', 'Mathematics'],
  gpa: 2.46,
  home_city: 'City 3',
  blood_group: 'A+',
  is_hotel_resident: false
},
{
  _id: ObjectId('6666f73122982336881819a4'),
  name: 'Student 232',
  age: 19,
  courses: ['Computer Science', 'Physics', 'History', 'Mathematics'],
  gpa: 2.54,
  home_city: 'City 1',
  blood_group: 'B-',
  is_hotel_resident: true
},
{
  _id: ObjectId('6666f73122982336881819a5'),
  name: 'Student 328',
  age: 21,
  courses: ['Physics', 'Computer Science', 'English'],
  gpa: 2.82,
  home_city: 'City 2',
  blood_group: 'AB-',
  is_hotel_resident: true
},
{
  _id: ObjectId('6666f73122982336881819a6'),
  name: 'Student 690',
  age: 20,
  courses: ['Computer Science', 'English', 'History'],
  gpa: 2.71,
  blood_group: 'AB+',
  is_hotel_resident: false
},
{
  _id: ObjectId('6666f73122982336881819a7'),
  name: 'Student 499',
  age: 25,
  courses: ['Mathematics', 'English', 'Computer Science', 'Physics'],
  gpa: 2.84,
  blood_group: 'A+',
  is_hotel_resident: false
},
{
  _id: ObjectId('6666f73122982336881819a9'),
  name: 'Student 504',
  age: 21,
  courses: ['Physics', 'Computer Science', 'English', 'Mathematics'],
  gpa: 2.42,
  home_city: 'City 2',
  blood_group: 'B+',
  is_hotel_resident: true
},
{
  _id: ObjectId('6666f73122982336881819ab'),
  name: 'Student 915',
  age: 22,
  courses: ['Computer Science', 'History', 'Physics', 'English'],
  gpa: 2.87,
  blood_group: 'AB+',
  is_hotel_resident: true
},
{
  _id: ObjectId('6666f73122982336881819ac'),
  name: 'Student 127',
  age: 19,
  courses: ['History', 'English', 'Computer Science', 'Mathematics'],
  gpa: 2.56,
  home_city: 'City 10',
  blood_group: 'AB+',
  is_hotel_resident: false
},
{
  _id: ObjectId('6666f73122982336881819ae'),
  name: 'Student 770',
  age: 20,
  courses: ['History', 'Computer Science', 'Mathematics', 'English'],
  gpa: 2.98,
  blood_group: 'B-',
  is_hotel_resident: false
},
{
  _id: ObjectId('6666f73122982336881819af'),
  name: 'Student 367',
  age: 25,
  courses: ['History', 'Physics', 'Computer Science'],
  gpa: 2.61,
  home_city: 'City 1',
  blood_group: 'AB+',
  is_hotel_resident: true
}
Type "it" for more
db> |
```



Let's Take new Data set

- New Students Permission dataset [link](#)
- **Explanation: Collection name: students_permission**
- **name:** Student's name (string)
- **age:** Student's age (number)
- **permissions:** Bitmask representing user permissions (number)

Bitwise Value

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

Bit 3	Bit 2	Bit 1
cafe	campus	lobby

Bitwise Types

Bitwise

Name	Description
<code>\$bitsAllClear</code>	Matches numeric or binary values in which a set of bit positions <i>all</i> have a value of 0.
<code>\$bitsAllSet</code>	Matches numeric or binary values in which a set of bit positions <i>all</i> have a value of 1.
<code>\$bitsAnyClear</code>	Matches numeric or binary values in which <i>any</i> bit from a set of bit positions has a value of 0.
<code>\$bitsAnySet</code>	Matches numeric or binary values in which <i>any</i> bit from a set of bit positions has a value of 1.



Query

```
// Define bit positions for permissions
const LOBBY_PERMISSION = 1;
const CAMPUS_PERMISSION = 2;

// Query to find students with both lobby and campus permissions using
db.students_permission.find({
  permissions: { $bitsAllSet: [LOBBY_PERMISSION, CAMPUS_PERMISSION] }
});
```

Output :

```
db> const LOBBY_PERMISSION=1;
db> const CAMPUS_PERMISSION=2;
db> db.student_permission.find({permissions:{$bitsAllSet:[LOBBY_PERMISSION,CAMPUS_PERMISSION]}});
[
  {
    _id: ObjectId('6650c5cbf86b0d030ac88b'),
    name: 'George',
    age: 21,
    permissions: 6
  },
  {
    _id: ObjectId('6650c5cbf86b0d030ac88c'),
    name: 'Henry',
    age: 27,
    permissions: 7
  },
  {
    _id: ObjectId('6650c5cbf86b0d030ac88d'),
    name: 'Isla',
    age: 18,
    permissions: 6
  }
]
db> |
```



Geospatial

- Official Documentation [link](#)
- Create collection called “locations”
- Upload the dataset using json [link](#)

```

    _id: 1
    name: "Coffee Shop A"
    location: Object
      type: "Point"
      coordinates: Array (2)

```

Query:

```

db.locations.find({
  location: {
    $geoWithin: {
      $centerSphere: [[-74.005, 40.712], 0.00621376] // 1 kilometer in
    }
  }
});

```

Output :

```

db> db.location.find({location:{$geoWithin:{$centerSphere:[[-74.005,40.712],0.00621376]]}});
[
  {
    _id: 1,
    name: 'Coffee Shop A',
    location: { type: 'Point', coordinates: [ -73.985, 40.748 ] }
  },
  {
    _id: 2,
    name: 'Restaurant B',
    location: { type: 'Point', coordinates: [ -74.009, 40.712 ] }
  },
  {
    _id: 5,
    name: 'Park E',
    location: { type: 'Point', coordinates: [ -74.006, 40.705 ] }
  }
]
db> |

```



Data types and Operations

- **DataType**
 - Point
 - Line String
 - Polygon

Data types and Operations

Name	Description
<code>\$geoIntersects</code>	Selects geometries that intersect with a GeoJSON geometry. The <code>2dsphere</code> index supports <code>\$geoIntersects</code> .
<code>\$geoWithin</code>	Selects geometries within a bounding GeoJSON geometry. The <code>2dsphere</code> and <code>2d</code> indexes support <code>\$geoWithin</code> .
<code>\$near</code>	Returns geospatial objects in proximity to a point. Requires a geospatial index. The <code>2dsphere</code> and <code>2d</code> indexes support <code>\$near</code> .
<code>\$nearSphere</code>	Returns geospatial objects in proximity to a point on a sphere. Requires a geospatial index. The <code>2dsphere</code> and <code>2d</code> indexes support <code>\$nearSphere</code> .



Experiment:4

Projection Operators

Create and demonstrate how projection operators (\$, \$elematch and \$slice) would be used in the MongoDB.

Projection

- Create new collection called candidates.
- Upload the dataset [link](#).

```
_id: ObjectId('665752830959f4120ac93d06')
name : "Emily Jones"
age : 21
▸ courses : Array (3)
gpa : 3.6
home_city : "Houston"
blood_group : "AB-"
is_hotel_resident : false
```

Query:

Example 1: Retrieve Name, Age, and GPA

JavaScript

```
db.candidates.find({}, { name: 1, age: 1, gpa: 1 });
```



Output :

```
db> db.candidates.find({}, {name:1, age:1, gpa:1})
[
  {
    _id: ObjectId('6668764273cc8ecde68c2e66'),
    name: 'Alice Smith',
    age: 20,
    gpa: 3.4
  },
  {
    _id: ObjectId('6668764273cc8ecde68c2e67'),
    name: 'Bob Johnson',
    age: 22,
    gpa: 3.8
  },
  {
    _id: ObjectId('6668764273cc8ecde68c2e68'),
    name: 'Charlie Lee',
    age: 19,
    gpa: 3.2
  },
  {
    _id: ObjectId('6668764273cc8ecde68c2e69'),
    name: 'Emily Jones',
    age: 21,
    gpa: 3.6
  },
  {
    _id: ObjectId('6668764273cc8ecde68c2e6a'),
    name: 'David Williams',
    age: 23,
    gpa: 3
  },
  {
    _id: ObjectId('6668764273cc8ecde68c2e6b'),
    name: 'Fatima Brown',
    age: 18,
    gpa: 3.5
  },
  {
    _id: ObjectId('6668764273cc8ecde68c2e6c'),
    name: 'Gabriel Miller',
    age: 24,
    gpa: 3.9
  },
  {
    _id: ObjectId('6668764273cc8ecde68c2e6d'),
    name: 'Hannah Garcia',
```

```

  {
    _id: ObjectId('6668764273cc8ecde68c2e6d'),
    name: 'Hannah Garcia',
    age: 20,
    gpa: 3.3
  },
  {
    _id: ObjectId('6668764273cc8ecde68c2e6e'),
    name: 'Isaac Clark',
    age: 22,
    gpa: 3.7
  },
  {
    _id: ObjectId('6668764273cc8ecde68c2e6f'),
    name: 'Jessica Moore',
    age: 19,
    gpa: 3.1
  },
  {
    _id: ObjectId('6668764273cc8ecde68c2e70'),
    name: 'Kevin Lewis',
    age: 21,
    gpa: 4
  },
  {
    _id: ObjectId('6668764273cc8ecde68c2e71'),
    name: 'Lily Robinson',
    age: 23,
    gpa: 3.5
  }
]
db> |
```



Variation: Exclude Fields

JavaScript

```
db.candidates.find({}, { _id: 0, courses: 0 }); // Exclude _id and courses
```

Output :

```
Please enter a MongoDB connection string (Default: db=candidates.find({}, {_id:0,courses:0})):
{
  name: 'Alice Smith',
  age: 20,
  gpa: 3.4,
  home_city: 'New York City',
  blood_group: 'B+',
  is_hotel_resident: true
},
{
  name: 'Bob Johnson',
  age: 22,
  gpa: 3.4,
  home_city: 'Los Angeles',
  blood_group: 'B+',
  is_hotel_resident: false
},
{
  name: 'Charlie Lee',
  age: 18,
  gpa: 3.2,
  home_city: 'Chicago',
  blood_group: 'B+',
  is_hotel_resident: true
},
{
  name: 'Emily Jones',
  age: 21,
  gpa: 3.6,
  home_city: 'Houston',
  blood_group: 'AB+',
  is_hotel_resident: false
},
{
  name: 'David Williams',
  age: 23,
  gpa: 3.1,
  home_city: 'Phoenix',
  blood_group: 'B+',
  is_hotel_resident: true
},
{
  name: 'Fatima Brown',
  age: 19,
  gpa: 3.8,
  home_city: 'San Antonio',
  blood_group: 'B+',
  is_hotel_resident: false
},
{
  name: 'Gabriel Miller',
  age: 24,
  gpa: 3.9,
  home_city: 'San Diego',
  blood_group: 'O+',
  is_hotel_resident: true
},
{
  name: 'Hannah Garcia',
  age: 20,
  gpa: 3.3,
  home_city: 'Dallas',
  blood_group: 'AB+',
  is_hotel_resident: false
},
{
  name: 'Isaac Clark',
  age: 22,
  gpa: 3.7,
  home_city: 'San Jose',
  blood_group: 'A-',
  is_hotel_resident: false
},
{
  name: 'Jessica Moore',
  age: 19,
  gpa: 3.1,
  home_city: 'Austin',
  blood_group: 'B-',
  is_hotel_resident: true
},
{
  name: 'Kevin Lewis',
  age: 21,
  gpa: 4,
  is_hotel_resident: false
},
{
  name: 'Lily Robinson',
  age: 23,
  gpa: 3.5,
  home_city: 'Jacksonville',
  blood_group: 'AB-',
  is_hotel_resident: false
},
{
  name: 'Mia Adams',
  age: 17,
  gpa: 3.0,
  home_city: 'Portland',
  blood_group: 'O-',
  is_hotel_resident: true
}
```

```
{
  name: 'Isaac Clark',
  age: 22,
  gpa: 3.7,
  home_city: 'San Jose',
  blood_group: 'A-',
  is_hotel_resident: false
},
{
  name: 'Jessica Moore',
  age: 19,
  gpa: 3.1,
  home_city: 'Austin',
  blood_group: 'B-',
  is_hotel_resident: true
},
{
  name: 'Kevin Lewis',
  age: 21,
  gpa: 4,
  is_hotel_resident: false
},
{
  name: 'Lily Robinson',
  age: 23,
  gpa: 3.5,
  home_city: 'Jacksonville',
  blood_group: 'AB-',
  is_hotel_resident: false
},
{
  name: 'Mia Adams',
  age: 17,
  gpa: 3.0,
  home_city: 'Portland',
  blood_group: 'O-',
  is_hotel_resident: true
}
db> |
```



\$elemMatch: [link](#)

3. Projection Operator (\$slice):

Example 3: Retrieve All Candidates with First Two Courses

JavaScript

```
db.candidates.find({}, { name: 1, courses: { $slice: 2 } });
```

Output :

```
db> db.candidates.find({}, {name:1,courses:$slice:2});
[
  {
    _id: ObjectId('6668764273cc8ecde68c2e66'),
    name: 'Alice Smith',
    courses: [ 'English', 'Biology' ]
  },
  {
    _id: ObjectId('6668764273cc8ecde68c2e67'),
    name: 'Bob Johnson',
    courses: [ 'Computer Science', 'Mathematics' ]
  },
  {
    _id: ObjectId('6668764273cc8ecde68c2e68'),
    name: 'Charlie Lee',
    courses: [ 'History', 'English' ]
  },
  {
    _id: ObjectId('6668764273cc8ecde68c2e69'),
    name: 'Emily Jones',
    courses: [ 'Mathematics', 'Physics' ]
  },
  {
    _id: ObjectId('6668764273cc8ecde68c2e6a'),
    name: 'David Williams',
    courses: [ 'English', 'Literature' ]
  },
  {
    _id: ObjectId('6668764273cc8ecde68c2e6b'),
    name: 'Fatima Brown',
    courses: [ 'Biology', 'Chemistry' ]
  },
  {
    _id: ObjectId('6668764273cc8ecde68c2e6c'),
    name: 'Gabriel Miller',
    courses: [ 'Computer Science', 'Engineering' ]
  },
  {
    _id: ObjectId('6668764273cc8ecde68c2e6d'),
    name: 'Hannah Garcia',
    courses: [ 'History', 'Art History' ]
  }
]
```

```

    name: 'Fatima Brown',
    courses: [ 'Biology', 'Chemistry' ]
  },
  {
    _id: ObjectId('6668764273cc8ecde68c2e6c'),
    name: 'Gabriel Miller',
    courses: [ 'Computer Science', 'Engineering' ]
  },
  {
    _id: ObjectId('6668764273cc8ecde68c2e6d'),
    name: 'Hannah Garcia',
    courses: [ 'History', 'Political Science' ]
  },
  {
    _id: ObjectId('6668764273cc8ecde68c2e6e'),
    name: 'Isaac Clark',
    courses: [ 'English', 'Creative Writing' ]
  },
  {
    _id: ObjectId('6668764273cc8ecde68c2e6f'),
    name: 'Jessica Moore',
    courses: [ 'Biology', 'Ecology' ]
  },
  {
    _id: ObjectId('6668764273cc8ecde68c2e70'),
    name: 'Kevin Lewis',
    courses: [ 'Computer Science', 'Artificial Intelligence' ]
  },
  {
    _id: ObjectId('6668764273cc8ecde68c2e71'),
    name: 'Lily Robinson',
    courses: [ 'History', 'Art History' ]
  }
]
db> |
```

