

- **CRUD**

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

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

- **Insert**

To insert new data into collection use the below query

```
Const studentsData={  
  "name":"Alice Smith",  
  "age":12,,  
  "coueses":["Mathematics","computer science"," English" ],  
  "gpa":3.5  
  "home_city":"New York",  
  "blood_group":"A+",  
  "is_hotel_resident":false  
};
```

```
test> 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  
...   };  
  
test> db.stu.insertOne(studentData);  
{  
  acknowledged: true,  
  insertedId: ObjectId('665b529e49389824aecdcdf7')  
}  
test> |
```

In this query we want to insert student data with name, age, courses ,gpa .home_ city, blood_ group and is hotel resident.

- **Update**

To update the given students data using query

db. students. updateOne({ name: "Alice Smith"} , {\$set: {gpa:3.8} })

```
db> db.students.updateOne({name:"Alice Smith"},{$set:{gpa:3.8}});
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 0,
  upsertedCount: 0
}
```

In this query we want update one student data with name "Alice Smith" and gpa is "3.8"..

- **Delete**

Delete is used to delete one student data from the given collection.

```
db> db.students.deleteOne({ name:"Sam" })
{ acknowledged: true, deletedCount: 1 }
db> |
```

In this query we want delete one student data with name "Sam" and acknowledgment is true, and delete Count is 1.

- **Update Many**

It is used to update students data in collections.

```
db> db.students.updateMany({gpa:{$lt:3.0}},{$inc:{gpa:0.5}});
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1032,
  modifiedCount: 1032,
  upsertedCount: 0
}
db>
```

It is used to update many students with gpa in the given data collection.

- **Delete Many**

Delete many query is used to delete many student or all student data from the given collection.

```

upsertedCount: 0
> db.students.deleteMany({is_hotel_resident:false});
acknowledged: true, deletedCount: 1011 }
>

```

It is used to delete many students from the collection based on is hotel resident.

• Projection

this is used when we don't need all columns. MongoDB provides a special feature that is known as Projection,

```

mongosh mongodb://127.0.0.1:27020
db> db.students.find({}, {name:1,gpa:1});
[
  {
    _id: ObjectId('665609bde575a33e43e69479'),
    name: 'Student 328',
    gpa: 3.42
  },
  {
    _id: ObjectId('665609bde575a33e43e6947c'),
    name: 'Student 468',
    gpa: 3.97
  },
  {
    _id: ObjectId('665609bde575a33e43e6947d'),
    name: 'Student 504',
    gpa: 2.92
  },
  {
    _id: ObjectId('665609bde575a33e43e6947f'),
    name: 'Student 915',
    gpa: 3.37
  },
  {
    _id: ObjectId('665609bde575a33e43e69483'),
    name: 'Student 367',
    gpa: 3.11
  },
  {
    _id: ObjectId('665609bde575a33e43e69488'),
    name: 'Student 969',
    gpa: 3.71
  },
  {
    _id: ObjectId('665609bde575a33e43e69489'),
    name: 'Student 502',
    gpa: 3.42
  },
  {
    _id: ObjectId('665609bde575a33e43e6948a'),
    name: 'Student 504',
    gpa: 2.9
  }
]

```

Projection is used to find only student name along with gpa is 1.

PROJECTION , LIMIT AND SELECTORS

Projection:

1. projection refers to the process of selecting specific fields to return from a query, effectively controlling which fields of the documents are included in the result set. This can help improve performance by reducing the amount of data transferred over the network and processed by the application.

2. The projection work based on:

- 1 indicates that the field should be included name to be returned.
- 0 indicates that the field should be excluded them from the result.

Ex: {

"_id": 1,

"name": "Alice",

"age": 30,

"email": "alice@example.com",

"address": {

"city": "New York",

"zipcode": "10001"

}

}

Get Selected Attributes:

To select specific fields in a MongoDB query, you use the `find()` method with a projection document. The projection document specifies which fields to include (using `1`) or exclude (using `0`).

```
db. students. find ( {}, { _id: 0 } );
```

```

db> db.students.find({}, {_id:0});
[
  {
    name: 'Student 328',
    age: 21,
    courses: "['Physics', 'Computer Science', 'English']",
    gpa: 3.42,
    home_city: 'City 2',
    blood_group: 'AB-',
    is_hotel_resident: true
  },
  {
    name: 'Student 468',
    age: 21,
    courses: "['Computer Science', 'Physics', 'Mathematics', 'History']",
    gpa: 3.97,
    blood_group: 'A-',
    is_hotel_resident: true
  },
  {
    name: 'Student 504',
    age: 21,
    courses: "['Physics', 'Computer Science', 'English', 'Mathematics']",
    gpa: 2.92,
    home_city: 'City 2',
    blood_group: 'B+',
    is_hotel_resident: true
  },
  {
    name: 'Student 915',
    age: 22,
    courses: "['Computer Science', 'History', 'Physics', 'English']",
    gpa: 3.37,
    blood_group: 'AB+',
    is_hotel_resident: true
  },
  {
    name: 'Student 367',
    age: 25,
    courses: "['History', 'Physics', 'Computer Science']",
    gpa: 3.11,
  }
]

```

it is used to retrieve all documents from the `students` collection while excluding the `_id` field from the results.

Ignore attributes:

In MongoDB, ignoring an attribute (or field) in query results is done using projection. By setting the value of a field to 0 in the projection document, you can exclude that field from the returned documents.

Db. students. find({}, {_id:0});

```

db> db.students.find({}, {_id:0});
[
  {
    name: 'Student 328',
    age: 21,
    courses: "['Physics', 'Computer Science', 'English']",
    gpa: 3.42,
    home_city: 'City 2',
    blood_group: 'AB-',
    is_hotel_resident: true
  },
  {
    name: 'Student 468',
    age: 21,
    courses: "['Computer Science', 'Physics', 'Mathematics', 'History']",
    gpa: 3.97,
    blood_group: 'A-',
    is_hotel_resident: true
  },
  {
    name: 'Student 504',
    age: 21,
    courses: "['Physics', 'Computer Science', 'English', 'Mathematics']",
    gpa: 2.92,
    home_city: 'City 2',
    blood_group: 'B+',
    is_hotel_resident: true
  },
  {
    name: 'Student 915',
    age: 22,
    courses: "['Computer Science', 'History', 'Physics', 'English']",
    gpa: 3.37,
    blood_group: 'AB+',
    is_hotel_resident: true
  },
  {
    name: 'Student 367',
    age: 25,
    courses: "['History', 'Physics', 'Computer Science']",
    gpa: 3.11,
  }
]

```

In this querying the "students" collection to retrieve all documents without including the `_id` field in the results. This is useful when you want to exclude the MongoDB-generated `_id` field, which is automatically added to each document unless specified otherwise.

Retrieving specific fields from nested objects:

Retrieving specific fields from nested objects in MongoDB involves using the projection feature to specify the fields within the nested documents that you want to include or exclude.

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

```
db> db.students.find({}, {name:1, courses:{$slice:1}});
[
  {
    _id: ObjectId('665609bde575a33e43e69479'),
    name: 'Student 328',
    courses: "['Physics', 'Computer Science', 'English']"
  },
  {
    _id: ObjectId('665609bde575a33e43e6947c'),
    name: 'Student 468',
    courses: "['Computer Science', 'Physics', 'Mathematics', 'History']"
  },
  {
    _id: ObjectId('665609bde575a33e43e6947d'),
    name: 'Student 504',
    courses: "['Physics', 'Computer Science', 'English', 'Mathematics']"
  },
  {
    _id: ObjectId('665609bde575a33e43e6947f'),
    name: 'Student 915',
    courses: "['Computer Science', 'History', 'Physics', 'English']"
  },
  {
    _id: ObjectId('665609bde575a33e43e69483'),
    name: 'Student 367',
    courses: "['History', 'Physics', 'Computer Science']"
  },
  {
    _id: ObjectId('665609bde575a33e43e69488'),
    name: 'Student 969',
    courses: "['History', 'Mathematics', 'Physics', 'English']"
  },
  {
    _id: ObjectId('665609bde575a33e43e69489'),
    name: 'Student 502',
    courses: "['Computer Science', 'Mathematics', 'English', 'Physics']"
  },
  {
    _id: ObjectId('665609bde575a33e43e6948a'),
```

This command in MongoDB is used to retrieve documents from the "students" collection, including only the "name" field and the first element of the 'courses' array from each document.

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.

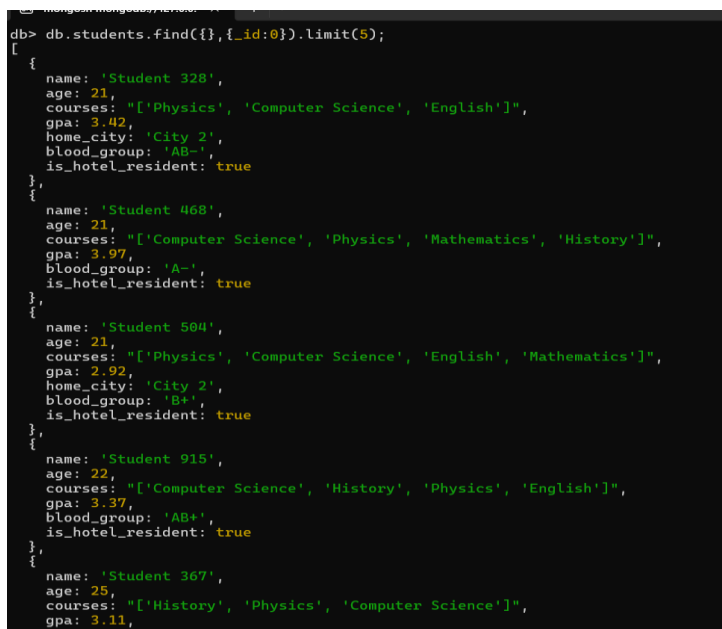
Limit

The `limit` method is used to restrict the number of documents returned by a query. This can be particularly useful when dealing with large collections, as it allows you to control the amount of data transferred and processed, thereby improving performance and reducing resource consumption.

Syntax: `db.collection.find({filter},{projection}).limit(number)`

Get first 5 document

`db.Students.find({}, {_id:0}).limit(5);`



```
mongo> db.students.find({}, {_id:0}).limit(5);
[
  {
    name: 'Student 328',
    age: 21,
    courses: "['Physics', 'Computer Science', 'English']",
    gpa: 3.42,
    home_city: 'City 2',
    blood_group: 'AB-',
    is_hotel_resident: true
  },
  {
    name: 'Student 468',
    age: 21,
    courses: "['Computer Science', 'Physics', 'Mathematics', 'History']",
    gpa: 3.97,
    blood_group: 'A-',
    is_hotel_resident: true
  },
  {
    name: 'Student 504',
    age: 21,
    courses: "['Physics', 'Computer Science', 'English', 'Mathematics']",
    gpa: 2.92,
    home_city: 'City 2',
    blood_group: 'B+',
    is_hotel_resident: true
  },
  {
    name: 'Student 915',
    age: 22,
    courses: "['Computer Science', 'History', 'Physics', 'English']",
    gpa: 3.37,
    blood_group: 'AB+',
    is_hotel_resident: true
  },
  {
    name: 'Student 367',
    age: 25,
    courses: "['History', 'Physics', 'Computer Science']",
    gpa: 3.11,
  }
]
```

This includes the `_id` field in the returned documents. Specifying `{ _id: 0 }` tells that it not to include this field. (`. limit (5)`) Restricting the number of documents returned to 5. This is useful for reducing the size of the result set.

Limiting Results

Limiting results in MongoDB is an important technique for controlling the amount of data returned by a query. This can help in optimizing performance, managing large datasets, and implementing pagination.

Syntax: `db.collection.find(<query criteria>).limit(<number>)`

- <query criteria>: Criteria to select documents.
- <number>: Maximum number of documents to return.

Ex :db. students. find({ gpa :{\$gt:3.5} },{_id:0}).limit(2);

```

db> db.students.find({gpa:{$gt:3.5}},{_id:0}).limit(2);
[
  {
    name: 'Student 468',
    age: 21,
    courses: "['Computer Science', 'Physics', 'Mathematics', 'History']",
    gpa: 3.97,
    blood_group: 'A-',
    is_hotel_resident: true
  },
  {
    name: 'Student 969',
    age: 24,
    courses: "['History', 'Mathematics', 'Physics', 'English']",
    gpa: 3.71,
    blood_group: 'B+',
    is_hotel_resident: true
  }
]
db> |

```

In this query This specifies that only documents where the gpa field is greater than 3.5 should be selected.

If we want 10 top students result with condition id=0, and limit =5

db. students. find({}, {_id:0}).sort({_id:-1}).limit(5);

```

db> db.students.find({}, {_id:0}).sort({_id:-1}).limit(5);
[
  {
    name: 'Student 933',
    age: 18,
    courses: "['Mathematics', 'English', 'Physics', 'History']",
    gpa: 3.04,
    home_city: 'City 10',
    blood_group: 'B-',
    is_hotel_resident: true
  },
  {
    name: 'Student 831',
    age: 20,
    courses: "['Mathematics', 'Computer Science']",
    gpa: 3.49,
    home_city: 'City 3',
    blood_group: 'AB+',
    is_hotel_resident: true
  },
  {
    name: 'Student 143',
    age: 21,
    courses: "['Mathematics', 'Computer Science', 'English', 'History']",
    gpa: 2.78,
    home_city: 'City 4',
    blood_group: 'O-',
    is_hotel_resident: true
  },
  {
    name: 'Student 718',
    age: 21,
    courses: "['Computer Science', 'English']",
    gpa: 2.75,
    home_city: 'City 5',
    blood_group: 'O-',
    is_hotel_resident: true
  },
  {
    name: 'Student 839',

```


This includes the `_id` field in the returned documents with Specifying `{ _id: 0 }` it tells that it not to include this field. Sorting the documents by the `_id` field in descending order. This means the documents with the highest `_id` values will come first. With limit (5).

Selectors:

Comparison between gt and lt:

Grater than: The `$gt` operator in MongoDB is used to specify a query condition where the field value must be greater than a specified value. It stands for "greater than."

Lesser than: the less than operation is performed using the `$lt` operator. This operator allows you to query documents where a specific field's value is less than a given value.

Greater Than (`$gt`):

- **Functionality:** The `$gt` operator selects documents where the value of a specified field is greater than (but not equal to) a given value.
- **Example:** `db. collection. find({ field: { $gt: value } })` selects documents where the `field` value is greater than `value`.
- **Use Cases:** Filtering documents with values exceeding a certain threshold, selecting the latest entries based on a timestamp, or retrieving documents with numerical values above a specific limit.

Less Than (`$lt`):

- **Functionality:** The `$lt` operator selects documents where the value of a specified field is less than (but not equal to) a given value.
- **Example:** `db. collection. find({ field: { $lt: value } })` selects documents where the `field` value is less than `value`.
- **Use Cases:** Filtering documents with values below a certain threshold, selecting older entries based on a timestamp, or retrieving documents with numerical values below a specific limit.

Ex: `db. students. find({age:{ $gt:20 } });`

```

db> db.students.find({age:{>20}});
[
  {
    _id: ObjectId('665609bde575a33e43e69479'),
    name: 'Student 328',
    age: 21,
    courses: "['Physics', 'Computer Science', 'English']",
    gpa: 3.42,
    home_city: 'City 2',
    blood_group: 'AB-',
    is_hotel_resident: true
  },
  {
    _id: ObjectId('665609bde575a33e43e6947c'),
    name: 'Student 468',
    age: 21,
    courses: "['Computer Science', 'Physics', 'Mathematics', 'History']",
    gpa: 3.97,
    blood_group: 'A-',
    is_hotel_resident: true
  },
  {
    _id: ObjectId('665609bde575a33e43e6947d'),
    name: 'Student 504',
    age: 21,
    courses: "['Physics', 'Computer Science', 'English', 'Mathematics']",
    gpa: 2.92,
    home_city: 'City 2',
    blood_group: 'B+',
    is_hotel_resident: true
  },
  {
    _id: ObjectId('665609bde575a33e43e6947f'),
    name: 'Student 915',
    age: 22,
    courses: "['Computer Science', 'History', 'Physics', 'English']",
    gpa: 3.37,
    blood_group: 'AB+',
    is_hotel_resident: true
  },
]

```

This query gives the students collection with age is greater than 20.

AND operator:

The logical AND operator (\$and) is used to combine multiple query conditions. The \$and operator is particularly useful when you need to match documents that satisfy all the specified conditions.

db. student. find({\$and :{{home_city: “City 2”}, {blood_group: “B+”}}});

```

Type "it" for more
db> db.students.find({$and:[{home_city:"City 2"},{blood_group:"B+"}]});
[
  {
    _id: ObjectId('665609bde575a33e43e6947d'),
    name: 'Student 504',
    age: 21,
    courses: "['Physics', 'Computer Science', 'English', 'Mathematics']",
    gpa: 2.92,
    home_city: 'City 2',
    blood_group: 'B+',
    is_hotel_resident: true
  },
  {
    _id: ObjectId('665609bde575a33e43e6962b'),
    name: 'Student 872',
    age: 24,
    courses: "['English', 'Mathematics', 'History']",
    gpa: 3.36,
    home_city: 'City 2',
    blood_group: 'B+',
    is_hotel_resident: true
  },
  {
    _id: ObjectId('665df4511655576509ad81c8'),
    name: 'Student 504',
    age: 21,
    courses: "['Physics', 'Computer Science', 'English', 'Mathematics']",
    gpa: 2.92,
    home_city: 'City 2',
    blood_group: 'B+',
    is_hotel_resident: true
  },
  {
    _id: ObjectId('665df4511655576509ad8376'),
    name: 'Student 872',
    age: 24,
    courses: "['English', 'Mathematics', 'History']",
    gpa: 3.36,
    home_city: 'City 2',
  }
]

```

In this query is used to find the students data with using \$and operator with home city “2” and bloods group is “B+”.

OR operation:

The logical OR operator (\$or) is used to combine multiple query conditions. The \$or operator returns documents that match at least one of the specified conditions.

The \$or operator takes an array of expressions and returns documents that satisfy at least one of the expressions in the array

Ex: db. student. find({\$or: [{is_hotel_resident:true},{gpa:{\$lt:3.0}}]});

```
]
db> db.students.find({$or:[{is_hotel_resident:true},{gpa:{$lt:3.0}}]});
[
  {
    _id: ObjectId('665609bde575a33e43e69479'),
    name: 'Student 328',
    age: 21,
    courses: "['Physics', 'Computer Science', 'English']",
    gpa: 3.42,
    home_city: 'City 2',
    blood_group: 'AB-',
    is_hotel_resident: true
  },
  {
    _id: ObjectId('665609bde575a33e43e6947c'),
    name: 'Student 468',
    age: 21,
    courses: "['Computer Science', 'Physics', 'Mathematics', 'History']",
    gpa: 3.97,
    blood_group: 'A-',
    is_hotel_resident: true
  },
  {
    _id: ObjectId('665609bde575a33e43e6947d'),
    name: 'Student 504',
    age: 21,
    courses: "['Physics', 'Computer Science', 'English', 'Mathematics']",
    gpa: 2.92,
    home_city: 'City 2',
    blood_group: 'B+',
    is_hotel_resident: true
  },
  {
    _id: ObjectId('665609bde575a33e43e6947f'),
    name: 'Student 915',
    age: 22,
    courses: "['Computer Science', 'History', 'Physics', 'English']",
    gpa: 3.37,
    blood_group: 'AB+',
    is_hotel_resident: true
  },
]
```

In this query is used to find the students data with using \$or operator with hotel resident and gpa is less than 3.0.

