MongoDB OPERATIONS

- BY Prem Vishwanath Patil

● *Start CMD WITH mongo in UBANTU*

sl@sl-HP-280-G3-SFF-Business-PC:~$ sudo systemctl start mongodb
[sudo] password for sl:
sl@sl-HP-280-G3-SFF-Business-PC:~$ sudo systemctl status mongodb
● mongodb.service - An object/document-oriented database
Loaded: loaded (/lib/systemd/system/mongodb.service; enabled; vendor preset: enabled)
Active: ***active (running)*** since Tue 2025-03-04 11:07:11 IST; 19min ago
Docs: man:mongod(1)
Main PID: 801 (mongod)
Tasks: 23 (limit: 9214)
Memory: 178.7M
CGroup: /system.slice/mongodb.service
└─801 /usr/bin/mongod --unixSocketPrefix=/run/mongodb --config /etc/mongodb.conf

Mar 04 11:07:11 sl-HP-280-G3-SFF-Business-PC systemd[1]: Started An object/document-oriented databas>

[1]+ Stopped sudo systemctl status mongodb
sl@sl-HP-280-G3-SFF-Business-PC:~$ mongo
MongoDB shell version v3.6.8
connecting to: mongodb://127.0.0.1:27017
Implicit session: session { "id" : UUID("a868c453-aac8-4631-a904-766b469e5c4a") }
MongoDB server version: 3.6.8
Server has startup warnings:
2025-03-04T11:07:11.808+0530 I STORAGE [initandlisten]
2025-03-04T11:07:11.808+0530 I STORAGE [initandlisten] ** WARNING: Using the XFS filesystem is strongly recommended with the WiredTiger storage engine

● *Start CMD WITH mongo in Window*

C:\Users\User\mongosh

● Shows databases

```
> show dbs;
RBTL24CS153  0.000GB

Students     0.000GB
admin        0.000GB
blog         0.000GB
config       0.000GB
local        0.000GB
test         0.000GB
```

● Switch/Create new databases

```
> use RBTL24CS153
switched to db RBTL24CS153
> use Students
switched to db Students
```

● Shows Tables

```
> use RBTL24CS153
switched to db RBTL24CS153
> show tables
RBTL24CS153
posts
> use Students
switched to db Students
> show tables
Syco
Tyco
```

● Create new Tables

```
> db.createCollection("posts")
{ "ok" : 1 }
> db.createCollection("RBTL24CS153")
{ "ok" : 1 }
> use Students
switched to db Students
> db.createCollection("Fyco")
{ "ok" : 1 }
> show tables
Fyco
Syco
```

Tyco

● Insert data in database Students

> use Students
switched to db Students

> db.Syco.insertMany([
{stud_id: 01, name: 'Prem', roll_no: 'CS2177', address: 'Pune', branch: 'Computer', college: 'JSPM'},
{stud_id: 02, name: 'Revti', roll_no: 'CS2171', address: 'bavdan', branch: 'A&R', college: 'Sinhgad'},
{stud_id: 03, name: 'Prathamesh', roll_no: 'CS2155', address: 'Dhankarvadi', branch: 'Computer', college: 'VIIT'},
{stud_id: 04, name: 'Dhanesh', roll_no: 'CS2285', address: 'Pune', branch: 'data science', college: 'Alard'}])

```
{
        "acknowledged" : true,
        "insertedIds" : [
                ObjectId("67cbd43c5144be28bfcc6383"),
                ObjectId("67cbd43c5144be28bfcc6384"),
                ObjectId("67cbd43c5144be28bfcc6385"),
                ObjectId("67cbd43c5144be28bfcc6386")
        ]
}
```

> db.Tyco.insertMany([
{stud_id: 01, name: 'Prem', roll_no: 'CS2177', address: 'Pune', branch: 'Computer', college: 'JSPM'},
{stud_id: 02, name: 'Revti', roll_no: 'CS2171', address: 'bavdan', branch: 'A&R', college: 'Sinhgad'},
{stud_id: 03, name: 'Prathamesh', roll_no: 'CS2155', address: 'Dhankarvadi', branch: 'Computer', college: 'VIIT'},
{stud_id: 04, name: 'Dhanesh', roll_no: 'CS2285', address: 'Pune', branch: 'data science', college: 'Alard'}])

```
{
        "acknowledged" : true,
        "insertedIds" : [
                ObjectId("67cbd45a5144be28bfcc6387"),
                ObjectId("67cbd45a5144be28bfcc6388"),
                ObjectId("67cbd45a5144be28bfcc6389"),
                ObjectId("67cbd45a5144be28bfcc638a")
        ]

}
```

● Drop Database

```
> show dbs;
RBTL24CS153  0.000GB
Students     0.000GB
admin        0.000GB
blog         0.000GB
config       0.000GB
local        0.000GB
test         0.000GB
> use Students;
switched to db Students

> db.dropDatabase()
{ "dropped" : "Students", "ok" : 1 }
> show dbs;
RBTL24CS153  0.000GB
admin        0.000GB
blog         0.000GB
config       0.000GB
local        0.000GB
test         0.000GB
```

● Drop Tables

```
> use RBTL24CS153
switched to db RBTL24CS153
> show tables
Syco
mytable
posts
> db.posts.drop()
true
> show tables
Syco
mytable
```

● Insert data in database RBTL24CS153

```
> db.RBTL24CS153.insertOne({'Title': 'Post 1 Prem'})
{
        "acknowledged" : true,
        "insertedId" : ObjectId("67c6975af1c4902fb8322f9f")
```

```
}
> db.RBTL24CS153.insertMany([{'Title1':'Prem', 'Organization':'Engineer'},
{'Title2':'Piyush', 'Organization':'MBBS'}])
{
        "acknowledged" : true,
        "insertedIds" : [
                ObjectId("67c69875f1c4902fb8322fa0"),
                ObjectId("67c69875f1c4902fb8322fa1")
        ]
}
```

● To display or find

```
> db.RBTL24CS153.find()
{ "_id" : ObjectId("67c6975af1c4902fb8322f9f"), "Title" : "Post 1 Prem" }
{ "_id" : ObjectId("67c69875f1c4902fb8322fa0"), "Title1" : "Prem", "Organization" :
"Engineer" }
{ "_id" : ObjectId("67c69875f1c4902fb8322fa1"), "Title2" : "Piyush", "Organization"
: "MBBS" }
> db.Syco.find().pretty()
{
        "_id" : ObjectId("67cbead7729909eac347944a"),
        "stud_id" : 2,
        "name" : "Revti",
        "roll_no" : "CS2171",
        "address" : "bavdan",
        "branch" : "A&R",
        "college" : "Sinhgad"
}
{
        "_id" : ObjectId("67cbead7729909eac347944b"),
        "stud_id" : 3,
        "name" : "Prathamesh",
        "roll_no" : "CS2155",
        "address" : "Dhankarvadi",
        "branch" : "Computer",
        "college" : "VIIT"
}
{
        "_id" : ObjectId("67cbf3686d72ae29dd6e21e3"),
        "stud_id" : 1,
        "name" : "Prem",
        "roll_no" : "CS2177",
```

"address" : "Pune",
"branch" : "Computer",

"college" : "JSPM"
}
> db.RBTL24CS153.findOne()
{ "_id" : ObjectId("67c6975af1c4902fb8322f9f"), "Title" : "Post 1 Prem" }
> db.RBTL24CS153.find({Organization: 'MBBS'})
{ "_id" : ObjectId("67c69875f1c4902fb8322fa1"), "Title2" : "Piyush", "Organization" : "MBBS" }
> db.RBTL24CS153.find({Organization: 'Engineer'})
{ "_id" : ObjectId("67c69875f1c4902fb8322fa0"), "Title1" : "Prem", "Organization" : "Engineer" }
> db.RBTL24CS153.find({})
{ "_id" : ObjectId("67c6975af1c4902fb8322f9f"), "Title" : "Post 1 Prem" }
{ "_id" : ObjectId("67c69875f1c4902fb8322fa0"), "Title1" : "Prem", "Organization" : "Engineer" }
{ "_id" : ObjectId("67c69875f1c4902fb8322fa1"), "Title2" : "Piyush", "Organization" : "MBBS" }

● Update Records

> db.RBTL24CS153.updateOne({Title:'Prem'}, { $set: { Organization: 77 } }, {upsert: true})
{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }
> db.RBTL24CS153.find({})
{ "_id" : ObjectId("67c6975af1c4902fb8322f9f"), "Title" : "Post 1 Prem" }
{ "_id" : ObjectId("67c69875f1c4902fb8322fa0"), "Title1" : "Prem", "Organization" : "Engineer" }
{ "_id" : ObjectId("67c69875f1c4902fb8322fa1"), "Title2" : "Piyush", "Organization" : "MBBS" }
{ "_id" : ObjectId("67c69c71ff0d03e74141b816"), "Title" : "Prem", "Organization" : 77 }
> db.RBTL24CS153.updateOne({Title1:'Prem'}, { $set: { Organization: 'CSEngineering' } }, {upsert: true})
{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }
> db.RBTL24CS153.find({Title1:'Prem'})
{ "_id" : ObjectId("67c69875f1c4902fb8322fa0"), "Title1" : "Prem", "Organization" : "CSEngineering" }
> db.RBTL24CS153.updateOne({Title1:'Prem'}, { $set: { Organization: 'CS Engineering' } })
{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }
> db.RBTL24CS153.find({Title1:'Prem'})

{ "_id" : ObjectId("67c69875f1c4902fb8322fa0"), "Title1" : "Prem", "Organization" : "CS Engineering" }

● Delete Records

> db.RBTL24CS153.deleteOne({ Title : "Prem" })
{ "acknowledged" : true, "deletedCount" : 1 }
> db.RBTL24CS153.find({})
{ "_id" : ObjectId("67c6975af1c4902fb8322f9f"), "Title" : "Post 1 Prem" }
{ "_id" : ObjectId("67c69875f1c4902fb8322fa0"), "Title1" : "Prem", "Organization" : "CS Engineering" }
{ "_id" : ObjectId("67c69875f1c4902fb8322fa1"), "Title2" : "Piyush", "Organization" : "MBBS" }

> use RBTL24CS153
switched to db RBTL24CS153
> db.dropDatabase()
{ "dropped" : "RBTL24CS153", "ok" : 1 }

● Refresing/Re-creating table

> use RBTL24CS153
switched to db RBTL24CS153
> db.Syco.insertMany([{stud_id: 01, name: 'Prem', roll_no: 'CS2177', address: 'Pune', branch: 'Computer', college: 'JSPM'}, {stud_id: 02, name: 'Revti', roll_no: 'CS2171', address: 'bavdan', branch: 'A&R', college: 'Sinhgad'}, {stud_id: 03, name: 'Prathamesh', roll_no: 'CS2155', address: 'Dhankarvadi', branch: 'Computer', college: 'VIIT'}, {stud_id: 04, name: 'Dhanesh', roll_no: 'CS2285', address: 'Pune', branch: 'data science', college: 'Alard'}])
{
        "acknowledged" : true,
        "insertedIds" : [
                ObjectId("67cbead7729909eac3479449"),
                ObjectId("67cbead7729909eac347944a"),
                ObjectId("67cbead7729909eac347944b"),
                ObjectId("67cbead7729909eac347944c")
        ]
}
> show tables
Syco

● Delete Many Records

> db.Syco.find()
{ "_id" : ObjectId("67cbead7729909eac3479449"), "stud_id" : 1, "name" : "Prem", "roll_no" : "CS2177", "address" : "Pune", "branch" : "Computer", "college" : "JSPM" }
{ "_id" : ObjectId("67cbead7729909eac347944a"), "stud_id" : 2, "name" : "Revti", "roll_no" : "CS2171", "address" : "bavdan", "branch" : "A&R", "college" : "Sinhgad" }
{ "_id" : ObjectId("67cbead7729909eac347944b"), "stud_id" : 3, "name" : "Prathamesh", "roll_no" : "CS2155", "address" : "Dhankarvadi", "branch" : "Computer", "college" : "VIIT" }
{ "_id" : ObjectId("67cbead7729909eac347944c"), "stud_id" : 4, "name" : "Dhanesh", "roll_no" : "CS2285", "address" : "Pune", "branch" : "data science", "college" : "Alard" }
> db.Syco.deleteMany({ address: "Pune" })
{ "acknowledged" : true, "deletedCount" : 2 }
> db.Syco.find()
{ "_id" : ObjectId("67cbead7729909eac347944a"), "stud_id" : 2, "name" : "Revti", "roll_no" : "CS2171", "address" : "bavdan", "branch" : "A&R", "college" : "Sinhgad" }
{ "_id" : ObjectId("67cbead7729909eac347944b"), "stud_id" : 3, "name" : "Prathamesh", "roll_no" : "CS2155", "address" : "Dhankarvadi", "branch" : "Computer", "college" : "VIIT" }

● MongoDB Query Operator

Comparison

The following operators can be used in queries to compare values:

- `$eq`: Values are equal
- `$ne`: Values are not equal
- `$gt`: Value is greater than another value
- `$gte`: Value is greater than or equal to another value
- `$lt`: Value is less than another value
- `$lte`: Value is less than or equal to another value
- `$in`: Value is matched within an array

Logical

The following operators can logically compare multiple queries.

- `$and`: Returns documents where both queries match
- `$or`: Returns documents where either query matches
- `$nor`: Returns documents where both queries fail to match
- `$not`: Returns documents where the query does not match

Evaluation

The following operators assist in evaluating documents.

- `$regex`: Allows the use of regular expressions when evaluating field values
- `$text`: Performs a text search
- `$where`: Uses a JavaScript expression to match documents

● MongoDB Update Operator

Fields

The following operators can be used to update fields:

- `$currentDate`: Sets the field value to the current date
- `$inc`: Increments the field value
- `$rename`: Renames the field
- `$set`: Sets the value of a field
- `$unset`: Removes the field from the document

Array

The following operators assist with updating arrays.

- `$addToSet`: Adds distinct elements to an array
- `$pop`: Removes the first or last element of an array
- `$pull`: Removes all elements from an array that match the query
- `$push`: Adds an element to an array
-

● Update Many Records

> db.mytable.updateMany({ category: 'A' }, { $set: { application: "Youtube" } })

```
{ "acknowledged" : true, "matchedCount" : 4, "modifiedCount" : 4 }
```

> db.mytable.find().pretty()

```
{
        "_id" : ObjectId("67ce9d98a725866d51b8f8fc"),
        "title" : "Post 1",
        "category" : "A",
        "likes" : 5,
        "application" : "Youtube"
}
{
        "_id" : ObjectId("67ce9d98a725866d51b8f8fd"),
        "title" : "Post 2",
        "category" : "B",
        "likes" : 3
}
```

```
{
        "_id" : ObjectId("67ce9d98a725866d51b8f8fe"),
        "title" : "Post 3",
        "category" : "A",
        "likes" : 2,
        "application" : "Youtube"
}
{
        "_id" : ObjectId("67ce9d98a725866d51b8f8ff"),
        "title" : "Post 4",
        "category" : "B",
        "likes" : 1
}
{
        "_id" : ObjectId("67ce9d98a725866d51b8f900"),
        "title" : "Post 5",
        "category" : "C",
        "likes" : 6
}
{
        "_id" : ObjectId("67ce9d98a725866d51b8f901"),
        "title" : "Post 6",
        "category" : "A",
        "likes" : 0,
        "application" : "Youtube"
}
```


● Update Many columns

```
> db.mytable.updateMany({},{ $set: {customer_id: "customer123"}});
{ "acknowledged" : true, "matchedCount" : 7, "modifiedCount" : 7 }
> db.mytable.find().pretty()
{
        "_id" : ObjectId("67ce9d98a725866d51b8f8fc"),
        "title" : "Post 1",
        "category" : "A",
        "likes" : 5,
        "application" : "Linkedin",
        "customer_id" : "customer123"
}
{
        "_id" : ObjectId("67ce9d98a725866d51b8f8fd"),
        "title" : "Post 2",
        "category" : "B",
```

```
        "likes" : 3,
        "customer_id" : "customer123"
}
.
.
.
```

● Insert using object

```
> db.mytable.drop()
true

const newPosts = [
   {
      _id: ObjectId("67ce9d98a725866d51b8f8fc"),
      title: "Post 1",
      category: "A",
      likes: 5,
      application: "Linkedin",
      customer_id: "customer123"
   },
   {
      _id: ObjectId("67ce9d98a725866d51b8f8fd"),
      title: "Post 2",
      category: "B",
      likes: 3,
      customer_id: "customer123"
   },
   {
      _id: ObjectId("67ce9d98a725866d51b8f8fe"),
      title: "Post 3",
      category: "A",
      likes: 2,
      application: "Youtube",
      customer_id: "customer123"
   },
   {
      _id: ObjectId("67ce9d98a725866d51b8f8ff"),
      title: "Post 4",
      category: "B",
      likes: 1,
      customer_id: "customer123"
   },
```

```
    {
        _id: ObjectId("67ce9d98a725866d51b8f900"),
        title: "Post 5",
        category: "C",
        likes: 6,
        customer_id: "customer123"
    },
    {
        _id: ObjectId("67ce9d98a725866d51b8f901"),
        title: "Post 6",
        category: "A",
        likes: 0,
        application: "Youtube",
        customer_id: "customer123"
    },
    {
        _id: ObjectId("67d7b6c12611696349e31aa9"),
        title: "Post 7",  // Corrected key from 'tittle' to 'title'
        category: "A",
        likes: 3,
        application: "Youtube",
        customer_id: "customer123"
    }
];

db.mytable.insertMany(newPosts);
```

● UpdateOne Records

```
> db.mytable.updateOne({ likes: 5 }, { $set: { application: "Linkedin" } })
{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }

> db.mytable.find({likes: 5}).pretty()
{
        "_id" : ObjectId("67ce9d98a725866d51b8f8fc"),
        "title" : "Post 1",
        "category" : "A",
        "likes" : 5,
        "application" : "Linkedin"
}
```

● MongoDB Aggregration Ppelines

> db.mytable.insertMany([{ title: "Post 1", category: "A", likes: 5 }, { title: "Post 2", category: "B", likes: 3 }, { title: "Post 3", category: "A", likes: 2 }, { title: "Post 4", category: "B", likes: 1 }, { title: "Post 5", category: "C", likes: 6 }, { title: "Post 6", category: "A", likes: 0 }])

```
{
        "acknowledged" : true,
        "insertedIds" : [
                ObjectId("67ce9a83a725866d51b8f8f6"),
                ObjectId("67ce9a83a725866d51b8f8f7"),
                ObjectId("67ce9a83a725866d51b8f8f8"),
                ObjectId("67ce9a83a725866d51b8f8f9"),
                ObjectId("67ce9a83a725866d51b8f8fa"),
                ObjectId("67ce9a83a725866d51b8f8fb")
        ]
}
```

> db.mytable.find().pretty()

```
{
        "_id" : ObjectId("67ce9d98a725866d51b8f8fc"),
        "title" : "Post 1",
        "category" : "A",
        "likes" : 5
}
{
        "_id" : ObjectId("67ce9d98a725866d51b8f8fd"),
        "title" : "Post 2",
        "category" : "B",
        "likes" : 3
}
{
        "_id" : ObjectId("67ce9d98a725866d51b8f8fe"),
        "title" : "Post 3",
        "category" : "A",
        "likes" : 2
}
{
        "_id" : ObjectId("67ce9d98a725866d51b8f8ff"),
        "title" : "Post 4",
        "category" : "B",
        "likes" : 1
}
{
        "_id" : ObjectId("67ce9d98a725866d51b8f900"),
```

```
        "title" : "Post 5",
        "category" : "C",
        "likes" : 6
}
{
        "_id" : ObjectId("67ce9d98a725866d51b8f901"),
        "title" : "Post 6",
        "category" : "A",
        "likes" : 0
}
```

//$Comparission

> db.mytable.aggregate({$match: {likes: {$gt:3}}})

```
{ "_id" : ObjectId("67ce9d98a725866d51b8f8fc"), "title" : "Post 1", "category" : "A", "likes" : 5 }
{ "_id" : ObjectId("67ce9d98a725866d51b8f900"), "title" : "Post 5", "category" : "C", "likes" : 6 }
```

//Group : This aggregation stage groups documents by the unique `_id` expression provided.

> db.mytable.aggregate([{$group: {_id: '$category'}}])

```
{ "_id" : "C" }
{ "_id" : "B" }
{ "_id" : "A" }
```

//$limit : This aggregation stage limits the number of documents passed to the next stage.

> db.mytable.aggregate([{$limit: 1}])

```
{ "_id" : ObjectId("67ce9d98a725866d51b8f8fc"), "title" : "Post 1", "category" : "A", "likes" : 5 }
```

> db.mytable.aggregate([{$limit: 2}])

```
{ "_id" : ObjectId("67ce9d98a725866d51b8f8fc"), "title" : "Post 1", "category" : "A", "likes" : 5 }
{ "_id" : ObjectId("67ce9d98a725866d51b8f8fd"), "title" : "Post 2", "category" : "B", "likes" : 3 }
```

//$project : This stage specifies which fields to include in the output documents.

> db.mytable.aggregate([{$project: {'title':1, 'category':1} }, {$limit: 2}])

```
{ "_id" : ObjectId("67ce9d98a725866d51b8f8fc"), "title" : "Post 1", "category" : "A" }
{ "_id" : ObjectId("67ce9d98a725866d51b8f8fd"), "title" : "Post 2", "category" : "B" }
```

//$sort : This aggregation stage groups sorts all documents in the specified sort order.
The sort order can be chosen by using `1` or `-1`

> db.mytable.aggregate([{$sort: {'likes': -1}}, {$project: {'title': 1}}, {$limit: 3}])

{ "_id" : ObjectId("67ce9d98a725866d51b8f900"), "title" : "Post 5" }
{ "_id" : ObjectId("67ce9d98a725866d51b8f8fc"), "title" : "Post 1" }
{ "_id" : ObjectId("67ce9d98a725866d51b8f8fd"), "title" : "Post 2" }

***Post 5** has the most likes (6)*

Post 1 follows with 5 likes.

Post 2 comes next with 3 likes.

> db.mytable.aggregate([{$sort: {'likes': 1}}, {$project: {'title': 1}}, {$limit: 3}])
{ "_id" : ObjectId("67ce9a83a725866d51b8f8fb"), "title" : "Post 6" }
{ "_id" : ObjectId("67ce9a83a725866d51b8f8f9"), "title" : "Post 4" }
{ "_id" : ObjectId("67ce9a83a725866d51b8f8f8"), "title" : "Post 3" }

***Post 6** has the less likes (0)*

Post 4 follows with 1 likes.

Post 3 comes next with 2 likes.

//$match : *You want to filter documents based on a property type.*
> db.mytable.aggregate([{ $match: { likes: { $gt: 1 } } }, { $group: { _id: '$category', sum_Likes: { $sum: '$likes' } } }])
{ "_id" : "C", "sum_Likes" : 6 }
{ "_id" : "B", "sum_Likes" : 3 }
{ "_id" : "A", "sum_Likes" : 7 }
> db.mytable.aggregate([{$match: {category: 'A'}}, {$limit: 2}, {$project: {'title':1, 'category':1}}])
{ "_id" : ObjectId("67ce9a83a725866d51b8f8f6"), "title" : "Post 1", "category" : "A" }
{ "_id" : ObjectId("67ce9a83a725866d51b8f8f8"), "title" : "Post 3", "category" : "A" }

//$addFields : *This aggregation stage adds new fields to documents.*
> db.mytable.aggregate([{$addFields: {application: "Youtube"}}, {$match: {category:{$eq: 'A'}}}])
{ "_id" : ObjectId("67ce9a83a725866d51b8f8f6"), "title" : "Post 1", "category" : "A", "likes" : 5, "application" : "Youtube" }
{ "_id" : ObjectId("67ce9a83a725866d51b8f8f8"), "title" : "Post 3", "category" : "A", "likes" : 2, "application" : "Youtube" }
{ "_id" : ObjectId("67ce9a83a725866d51b8f8fb"), "title" : "Post 6", "category" : "A", "likes" : 0, "application" : "Youtube" }

//$count : *This aggregation stage counts the total amount of documents passed from the previous stage.*
> db.mytable.aggregate([{ $group: { _id: "$category", count: { $sum: 1 } } }])

```
{ "_id" : "C", "count" : 1 }
{ "_id" : "B", "count" : 2 }
{ "_id" : "A", "count" : 4 }
```

//$lookup : This aggregation stage performs a left outer join to a collection in the same database.

1. insert another table

```
db.customers.insertMany([
   { "_id": "customer001", "name": "Alice", "email": "alice@example.com" },
   { "_id": "customer002", "name": "Bob", "email": "bob@example.com" },
   { "_id": "customer003", "name": "Charlie", "email": "char@example.com" },
   { "_id": "customer004", "name": "David", "email": "david@example.com" },
   { "_id": "customer005", "name": "Eve", "email": "eve@example.com" },
   { "_id": "customer006", "name": "Frank", "email": "frank@example.com" },
   { "_id": "customer007", "name": "Grace", "email": "grace@example.com" }
]);
```

1. now perform operation with previou table

```
db.mytable.aggregate([
   {
      $lookup: {
         from: "customers",
         localField: "customer_id",
         foreignField: "_id",
         as: "customer_info"
      }
   }
]);
```

```
>[
   {
      "_id": ObjectId("67ce9d98a725866d51b8f8fc"),
      "title": "Post 1",
      "category": "A",
      "likes": 5,
      "application": "Linkedin",
      "customer_id": "customer001",
      "customer_info": [
         {
            "_id": "customer001",
            "name": "Alice",
            "email": "alice@example.com"
         }
      ]
   },
   {
```

```
    "_id": ObjectId("67ce9d98a725866d51b8f8fd"),
    "title": "Post 2",
    "category": "B",
    "likes": 3,
    "customer_id": "customer002",
    "customer_info": [
      {
        "_id": "customer002",
        "name": "Bob",
        "email": "bob@example.com"
      }
    ]
  }                    // Other posts with customer information...
]
```

//$out : In this case, the results will be stored in a new collection named Table

```
db.mytable.aggregate([
  {
    $lookup: {
      from: "customers",
      localField: "customer_id",
      foreignField: "_id",
      as: "customer_info"
    }
  },
  {
    $out: "mytable_aggregate" // The name of the new collection
  }
]);
```

```
> db.mytable_aggregate.find().pretty()
{
        "_id" : ObjectId("67ce9d98a725866d51b8f8fc"),
        "title" : "Post 1",
        "category" : "A",
        "likes" : 5,
        "application" : "Linkedin",
        "customer_id" : "customer001",
        "customer_info" : [
                {
                        "_id" : "customer001",
                        "name" : "Alice",
                        "email" : "alice@example.com"
                }
        ]
```

```
}
{
        "_id" : ObjectId("67ce9d98a725866d51b8f8fd"),
        "title" : "Post 2",
        "category" : "B",
        "likes" : 3,
        "customer_id" : "customer002",
        "customer_info" : [
                {
                        "_id" : "customer002",
                        "name" : "Bob",
                        "email" : "bob@example.com"
                }
        ]
}               // Other posts with customer information...
//$const :

const customerIds = [
   { _id: ObjectId("67ce9d98a725866d51b8f8fc"), customer_id: "customer001" },
   { _id: ObjectId("67ce9d98a725866d51b8f8fd"), customer_id: "customer002" },
   { _id: ObjectId("67ce9d98a725866d51b8f8fe"), customer_id: "customer003" },
   { _id: ObjectId("67ce9d98a725866d51b8f8ff"), customer_id: "customer004" },
   { _id: ObjectId("67ce9d98a725866d51b8f900"), customer_id: "customer005" },
   { _id: ObjectId("67ce9d98a725866d51b8f901"), customer_id: "customer006" },
   { _id: ObjectId("67d7b6c12611696349e31aa9"), customer_id: "customer007" }
];

customerIds.forEach(entry => {
   db.mytable.updateOne(
      { _id: entry._id },
      { $set: { customer_id: entry.customer_id } }
   );
});

> db.mytable.find().pretty()
{
        "_id" : ObjectId("67ce9d98a725866d51b8f8fc"),
        "title" : "Post 1",
        "category" : "A",
        "likes" : 5,
        "application" : "Linkedin",
        "customer_id" : "customer001"
}
{
```

```
        "_id" : ObjectId("67ce9d98a725866d51b8f8fd"),
        "title" : "Post 2",
        "category" : "B",
        "likes" : 3,
        "customer_id" : "customer002"
}
{
        "_id" : ObjectId("67ce9d98a725866d51b8f8fe"),
        "title" : "Post 3",
        "category" : "A",
        "likes" : 2,
        "application" : "Youtube",
        "customer_id" : "customer003"
}
{
        "_id" : ObjectId("67ce9d98a725866d51b8f8ff"),
        "title" : "Post 4",
        "category" : "B",
        "likes" : 1,
        "customer_id" : "customer004"
}
{
        "_id" : ObjectId("67ce9d98a725866d51b8f900"),
        "title" : "Post 5",
        "category" : "C",
        "likes" : 6,
        "customer_id" : "customer005"
}
{
        "_id" : ObjectId("67ce9d98a725866d51b8f901"),
        "title" : "Post 6",
        "category" : "A",
        "likes" : 0,
        "application" : "Youtube",
        "customer_id" : "customer006"
}
{
        "_id" : ObjectId("67d7b6c12611696349e31aa9"),
        "title" : "Post 7",
        "category" : "A",
        "likes" : 3,
        "application" : "Youtube",
        "customer_id" : "customer007"
}
```

//$Indexing/Search : comes with a full-text search engine that can be used to search for documents in a collection.

> db.mytable.createIndex({"application": "text"})

```
{
        "createdCollectionAutomatically" : false,
        "numIndexesBefore" : 1,
        "numIndexesAfter" : 2,
        "ok" : 1
}
```

> db.mytable.getIndexes()

```
[
        {
                "v" : 2,
                "key" : {
                        "_id" : 1
                },
                "name" : "_id_",
                "ns" : "mytable.mytable"
        },
        {
                "v" : 2,
                "key" : {
                        "_fts" : "text",
                        "_ftsx" : 1
                },
                "name" : "application_text",
                "ns" : "mytable.mytable",
                "weights" : {
                        "application" : 1
                },
                "default_language" : "english",
                "language_override" : "language",
                "textIndexVersion" : 3
        }
]
```

> db.mytable.find({$text: {$search: "Youtube"}})

```
{ "_id" : ObjectId("67d7b6c12611696349e31aa9"), "title" : "Post 7", "category" : "A", "likes" : 3, "application" : "Youtube", "customer_id" : "customer007" }
{ "_id" : ObjectId("67ce9d98a725866d51b8f901"), "title" : "Post 6", "category" : "A", "likes" : 0, "application" : "Youtube", "customer_id" : "customer006" }
{ "_id" : ObjectId("67ce9d98a725866d51b8f8fe"), "title" : "Post 3", "category" : "A", "likes" : 2, "application" : "Youtube", "customer_id" : "customer003" }
```

//$Validation : After applying the validation, any attempts to insert documents that do not conform to the schema will be rejected.

```
db.createCollection("mytable", {
   validator: {
      $jsonSchema: {
         bsonType: "object",
         required: ["title", "category", "likes", "application", "customer_id"],
         properties: {
            title: {
               bsonType: "string",
               description: "must be a string and is required"
            },
            category: {
               bsonType: "string",
               description: "must be a string and is required"
            },
            likes: {
               bsonType: "int",
               minimum: 0,
               description: "must be an int greater than or equal to 0 and is
required"
            },
            application: {
               bsonType: "string",
               description: "must be a string and is required"
            },
            customer_id: {
               bsonType: "string",
               description: "must be a string and is required"
            }
         }
      }
   }
});
```

1. Valid Insert Example

```
db.mytable.insert({
   title: "Valid Post",
   category: "General",
   likes: 10, // Valid positive integer
   application: "LinkedIn",
   customer_id: "customer001"
});
{
   "acknowledged" : true,
```

```
    "insertedId" : ObjectId("your_inserted_object_id")
}
```

2. *IN*-Valid Insert Example

```
db.mytable.insert({
    title: "Invalid Post",
    category: "General",
    likes: -5, // Invalid because likes must be >= 0
    application: "LinkedIn"
    // customer_id is missing
});
```