MONGODB COMMANDS: -

**🡺"Mongosh" is a shell interface for MongoDB, a popular NoSQL database system**

C:\Users\pmpk2>**mongosh**

Current Mongosh Log ID: 65cb3206e06243e5250a908e

Connecting to: mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongosh+2.1.3

Using MongoDB: 7.0.5

Using Mongosh: 2.1.3

mongosh 2.1.4 is available for download: https://www.mongodb.com/try/download/shell

For mongosh info see: https://docs.mongodb.com/mongodb-shell/

------

The server generated these startup warnings when booting

2024-02-08T15:46:07.556+05:30: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted

test> **show dbs**

* **show dbs is a command used in the MongoDB shell to display a list of databases present on the current MongoDB server.**

admin 40.00 KiB

config 72.00 KiB

local 72.00 KiB

test> **use studentdb**

* **In MongoDB** **You can switch one to other database (or) we can create a new database by typing use then the name of the database. (“studentdb is a database name”)**

switched to db studentdb

studentdb> **db.createCollection("student")**

* **db.createCollection is a method used to explicitly create a new collection within a specified database.**
* **createCollection is the method used to create the collection.**

{ ok: 1 } // ok is a success and 1 is true

studentdb> **show dbs**

admin 40.00 KiB

config 72.00 KiB

local 72.00 KiB

studentdb 8.00 KiB

studentdb> **db 🡺 db refers to the current database.**

studentdb

studentdb> **show collections**

* **When you execute show collections, MongoDB lists all the collections available in the current database.**

student

studentdb> **db.student.drop()**

true

* **student: This is the name of the collection that you want to drop.**
* **drop(): This is the method used to drop (or) delete the collection.**

studentdb> **show collections**

studentdb> **db.dropDatabase()**

* **db.dropDatabase() is a method used to permanently delete the current database from the MongoDB server.**

{ ok: 1, dropped: 'studentdb' }

studentdb> **show dbs**

admin 40.00 KiB

config 108.00 KiB

local 72.00 KiB

studentdb> **use admin**

switched to db admin

admin> **use studentdb**

switched to db studentdb

studentdb> **db.createCollection("student")**

{ ok: 1 }

studentdb> **show dbs**

admin 40.00 KiB

config 108.00 KiB

local 72.00 KiB

studentdb 8.00 KiB

studentdb> **show collections**

student

studentdb> **db.student.insert({sid:30001, sname:"KLU"})**

* **insert (): This method is used to insert documents into a collection within the current database. (Here, Collection is a “Student”, sid and sname are fields.)**

DeprecationWarning: Collection.insert() is deprecated. Use insertOne, insertMany, or bulkWrite.

{

acknowledged: true,

insertedIds: { '0': ObjectId('65c35f4917f49f353a12b665') }

}

studentdb> **db.student.find()**

* **The command db.student.find() in MongoDB is used to retrieve documents from the "student" collection within the current database.**
* **When we execute find() method, MongoDB will return all documents stored in a collection. If the collection is empty, the command will return an empty result set.**

[

{

\_id: ObjectId('65c35f4917f49f353a12b665'),

sid: 30001,

sname: 'KLU'

}

]

studentdb> **db.student.insertOne({sid:30002,sname:"klef"})**

* **insertOne() is a method used to insert a single document into a collection**

{

acknowledged: true,

insertedId: ObjectId('65c3607617f49f353a12b666')

}

studentdb> **db.student.insertOne({sid:30003,sname:"klef",dept:"CSE"})**

{

acknowledged: true,

insertedId: ObjectId('65c360c817f49f353a12b667')

}

studentdb> **db.student.find()**

[

{

\_id: ObjectId('65c35f4917f49f353a12b665'),

sid: 30001,

sname: 'KLU'

},

{

\_id: ObjectId('65c3607617f49f353a12b666'),

sid: 30002,

sname: 'klef'

},

{

\_id: ObjectId('65c360c817f49f353a12b667'),

sid: 30003,

sname: 'klef',

dept: 'CSE'

}

]

studentdb> **db.student.find().pretty() it is used to print the output in a more readable**

[

{

\_id: ObjectId('65c35f4917f49f353a12b665'),

sid: 30001,

sname: 'KLU'

},

{

\_id: ObjectId('65c3607617f49f353a12b666'),

sid: 30002,

sname: 'klef'

},

{

\_id: ObjectId('65c360c817f49f353a12b667'),

sid: 30003,

sname: 'klef',

dept: 'CSE'

}

]

studentdb> **db.student.insertMany([{sid:30004},{sid:30005,sname:"30865"}])**

* **insertMany() is a method used to insert multiple documents into a collection in a single operation**

{

acknowledged: true,

**insertedIds**: {

'0': ObjectId('65c3632717f49f353a12b668'),

'1': ObjectId('65c3632717f49f353a12b669')

}

}

studentdb> **db.student.find().size()**

* **size() method is used to retrieve the number of elements in an array field within a document.**

5

studentdb> **db.student.countDocuments()**

* **countDocuments() method is used to count the number of documents that match a query within a collection.**

5

studentdb> **db.student.find().limit(1)**

* **The command db.student.find().limit(1) in MongoDB is used to query the "student" collection and retrieve only one document from it.**

[

{

\_id: ObjectId('65c35f4917f49f353a12b665'),

sid: 30001,

sname: 'KLU'

}

]

studentdb> **db.student.find().limit(2)**

* **limit(2) in MongoDB is used to query the “student” collection & retrieve only two documents from it**

[

{

\_id: ObjectId('65c35f4917f49f353a12b665'),

sid: 30001,

sname: 'KLU'

},

{

\_id: ObjectId('65c3607617f49f353a12b666'),

sid: 30002,

sname: 'klef'

}

]

studentdb> **db.student.find().sort({sid:1}) // 1 represents ascending order**

[

{

\_id: ObjectId('65c35f4917f49f353a12b665'),

sid: 30001,

sname: 'KLU'

},

{

\_id: ObjectId('65c3607617f49f353a12b666'),

sid: 30002,

sname: 'klef'

},

{

\_id: ObjectId('65c360c817f49f353a12b667'),

sid: 30003,

sname: 'klef',

dept: 'CSE'

},

{ \_id: ObjectId('65c3632717f49f353a12b668'), sid: 30004 },

{

\_id: ObjectId('65c3632717f49f353a12b669'),

sid: 30005,

sname: '30865'

}

]

studentdb> **db.student.find().sort({sid:-1}) // -1 represents descending order.**

[

{

\_id: ObjectId('65c3632717f49f353a12b669'),

sid: 30005,

sname: '30865'

},

{ \_id: ObjectId('65c3632717f49f353a12b668'), sid: 30004 },

{

\_id: ObjectId('65c360c817f49f353a12b667'),

sid: 30003,

sname: 'klef',

dept: 'CSE'

},

{

\_id: ObjectId('65c3607617f49f353a12b666'),

sid: 30002,

sname: 'klef'

},

{

\_id: ObjectId('65c35f4917f49f353a12b665'),

sid: 30001,

sname: 'KLU'

}

]

studentdb> **db.student.find().sort()**

* **sort() method will sort in ascending order**

[

{

\_id: ObjectId('65c35f4917f49f353a12b665'),

sid: 30001,

sname: 'KLU'

},

{

\_id: ObjectId('65c3607617f49f353a12b666'),

sid: 30002,

sname: 'klef'

},

{

\_id: ObjectId('65c360c817f49f353a12b667'),

sid: 30003,

sname: 'klef',

dept: 'CSE'

},

{ \_id: ObjectId('65c3632717f49f353a12b668'), sid: 30004 },

{

\_id: ObjectId('65c3632717f49f353a12b669'),

sid: 30005,

sname: '30865'

}

]

studentdb> **db.student.updateOne({sid:30005},{$set:{sname:"RAMA"}})**

* **updateOne(): This method is used to update a single document in the collection that matches the specified filter criteria.**
* **It takes two parameters: a filter object that specifies which document(s) to update, and an update object that specifies how to modify the matched document(s).**
* **{sid:30005}: This is the filter criteria specifying which document to update.**
* **The $set operator in MongoDB is used within update operations to set or update the value of specific fields within a document.**
* **(The $set operator is used to set the value of the "sname" field to "RAMA". So, if a document matching the filter criteria is found, its "sname" field will be updated to "RAMA".)**

{

acknowledged: true,

insertedId: null,

matchedCount: 1,

modifiedCount: 1,

upsertedCount: 0

}

studentdb> **db.student.find()**

[

{

\_id: ObjectId('65c35f4917f49f353a12b665'),

sid: 30001,

sname: 'KLU'

},

{

\_id: ObjectId('65c3607617f49f353a12b666'),

sid: 30002,

sname: 'klef'

},

{

\_id: ObjectId('65c360c817f49f353a12b667'),

sid: 30003,

sname: 'klef',

dept: 'CSE'

},

{ \_id: ObjectId('65c3632717f49f353a12b668'), sid: 30004 },

{

\_id: ObjectId('65c3632717f49f353a12b669'),

sid: 30005,

sname: 'RAMA'

}

]

studentdb> **db.student.find().limit(1)**

[

{

\_id: ObjectId('65c35f4917f49f353a12b665'),

sid: 30001,

sname: 'KLU'

}

]

studentdb> **db.student.find().limit(2)**

[

{

\_id: ObjectId('65c35f4917f49f353a12b665'),

sid: 30001,

sname: 'KLU'

},

{

\_id: ObjectId('65c3607617f49f353a12b666'),

sid: 30002,

sname: 'klef'

}

]

studentdb> **db.student.find().limit(3)**

[

{

\_id: ObjectId('65c35f4917f49f353a12b665'),

sid: 30001,

sname: 'KLU'

},

{

\_id: ObjectId('65c3607617f49f353a12b666'),

sid: 30002,

sname: 'klef'

},

{

\_id: ObjectId('65c360c817f49f353a12b667'),

sid: 30003,

sname: 'klef',

dept: 'CSE'

}

]

studentdb> **db.student.find().limit(4)**

[

{

\_id: ObjectId('65c35f4917f49f353a12b665'),

sid: 30001,

sname: 'KLU'

},

{

\_id: ObjectId('65c3607617f49f353a12b666'),

sid: 30002,

sname: 'klef'

},

{

\_id: ObjectId('65c360c817f49f353a12b667'),

sid: 30003,

sname: 'klef',

dept: 'CSE'

},

{ \_id: ObjectId('65c3632717f49f353a12b668'), sid: 30004 }

]

studentdb> **db.student.find().limit(5)**

[

{

\_id: ObjectId('65c35f4917f49f353a12b665'),

sid: 30001,

sname: 'KLU'

},

{

\_id: ObjectId('65c3607617f49f353a12b666'),

sid: 30002,

sname: 'klef'

},

{

\_id: ObjectId('65c360c817f49f353a12b667'),

sid: 30003,

sname: 'klef',

dept: 'CSE'

},

{ \_id: ObjectId('65c3632717f49f353a12b668'), sid: 30004 },

{

\_id: ObjectId('65c3632717f49f353a12b669'),

sid: 30005,

sname: 'RAMA'

}

]

studentdb> **db.student.find({sid:30005})**

[

{

\_id: ObjectId('65c3632717f49f353a12b669'),

sid: 30005,

sname: 'RAMA'

}

]

studentdb> **db.student.findOne({sid:30001})**

{ \_id: ObjectId('65c35f4917f49f353a12b665'), sid: 30001, sname: 'KLU' }

studentdb> **db.student.findOne({sid:30004})**

{ \_id: ObjectId('65c3632717f49f353a12b668'), sid: 30004 }

studentdb> **db.student.updateOne({sid:30004},{$set:{sname:"manish",dept:"CSE"}})**

{

acknowledged: true,

insertedId: null,

matchedCount: 1,

modifiedCount: 1,

upsertedCount: 0

}

studentdb> **db.student.findOne({sid:30004})**

{

\_id: ObjectId('65c3632717f49f353a12b668'),

sid: 30004,

dept: 'CSE',

sname: 'manish'

}

studentdb> **db.student.findOne({dept:"CSE"})**

* **This method retrieves the first document that matches the specified query criteria.**

{

\_id: ObjectId('65c360c817f49f353a12b667'),

sid: 30003,

sname: 'klef',

dept: 'CSE'

}

studentdb> **db.student.find({dept:"CSE"})**

[

{

\_id: ObjectId('65c360c817f49f353a12b667'),

sid: 30003,

sname: 'klef',

dept: 'CSE'

},

{

\_id: ObjectId('65c3632717f49f353a12b668'),

sid: 30004,

dept: 'CSE',

sname: 'manish'

}

]

studentdb> **db.student.find({sid:30001,sname:"KLU"})**

[

{

\_id: ObjectId('65c35f4917f49f353a12b665'),

sid: 30001,

sname: 'KLU'

}

]

studentdb> **db.student.find({sid:30005})**

[

{

\_id: ObjectId('65c3632717f49f353a12b669'),

sid: 30005,

sname: 'RAMA'

}

]

studentdb> **db.student.deleteOne({sid:30005})**

* **The deleteOne() method in MongoDB is used to delete a single document that matches the specified filter criteria from a collection.**

{ acknowledged: true, deletedCount: 1 }

studentdb> **db.student.find().size()**

* **size() method is used to retrieve the number of elements in an array field within a document.**

4

studentdb> **db.student.countDocuments()**

* **countDocuments() method is used to count the number of documents that match a query within a collection.**

4

studentdb> **db.student.find({dept:"CSE"})**

[

{

\_id: ObjectId('65c360c817f49f353a12b667'),

sid: 30003,

sname: 'klef',

dept: 'CSE'

},

{

\_id: ObjectId('65c3632717f49f353a12b668'),

sid: 30004,

dept: 'CSE',

sname: 'manish'

}

]

studentdb> **db.student.deleteMany({dept:"CSE"})**

* **The deleteMany() method in MongoDB is used to delete multiple documents that match the specified filter criteria from a collection**

{ acknowledged: true, deletedCount: 2 }

studentdb> **db.student.countDocuments()**

2

studentdb> **db.student.find()**

[

{

\_id: ObjectId('65c35f4917f49f353a12b665'),

sid: 30001,

sname: 'KLU'

},

{

\_id: ObjectId('65c3607617f49f353a12b666'),

sid: 30002,

sname: 'klef'

}

]

studentdb> **db.student.find({})**

* **The command db.student.find({}) in MongoDB queries the "student" collection to retrieve all documents without any specific filter condition.**

[

{

\_id: ObjectId('65c35f4917f49f353a12b665'),

sid: 30001,

sname: 'KLU'

},

{

\_id: ObjectId('65c3607617f49f353a12b666'),

sid: 30002,

sname: 'klef'

}

]

studentdb> **db.student.find({sid:30001})**

[

{

\_id: ObjectId('65c35f4917f49f353a12b665'),

sid: 30001,

sname: 'KLU'

}

]

studentdb> **db.student.find({}).skip(1)**

[

{

\_id: ObjectId('65c3607617f49f353a12b666'),

sid: 30002,

sname: 'klef'

}

]

studentdb> **db.student.find({}).skip(3)**

studentdb> **db.student.find({}).skip(0)**

[

{

\_id: ObjectId('65c35f4917f49f353a12b665'),

sid: 30001,

sname: 'KLU'

},

{

\_id: ObjectId('65c3607617f49f353a12b666'),

sid: 30002,

sname: 'klef'

}

]

studentdb> **db.student.find({}).limit(1)**

[

{

\_id: ObjectId('65c35f4917f49f353a12b665'),

sid: 30001,

sname: 'KLU'

}

]

studentdb> **db.student.updateOne({sid:30002},{$set:{sname:"RAMA"}})**

{

acknowledged: true,

insertedId: null,

matchedCount: 1,

modifiedCount: 1,

upsertedCount: 0

}

studentdb> **db.student.updateMany({},{$set:{gender:"FEMALE"}})**

{

acknowledged: true,

insertedId: null,

matchedCount: 2,

modifiedCount: 2,

upsertedCount: 0

}

studentdb> **db.student.find({})**

[

{

\_id: ObjectId('65c35f4917f49f353a12b665'),

sid: 30001,

sname: 'KLU',

gender: 'FEMALE'

},

{

\_id: ObjectId('65c3607617f49f353a12b666'),

sid: 30002,

sname: 'RAMA',

gender: 'FEMALE'

}

]

studentdb> **db.student.find({gender:"FEMALE"})**

[

{

\_id: ObjectId('65c35f4917f49f353a12b665'),

sid: 30001,

sname: 'KLU',

gender: 'FEMALE'

},

{

\_id: ObjectId('65c3607617f49f353a12b666'),

sid: 30002,

sname: 'RAMA',

gender: 'FEMALE'

}

]

studentdb>**db.student.updateMany({sid:30003},{$set:{sname:"SRINU",gender:"MALE"}})**

{

acknowledged: true,

insertedId: null,

matchedCount: 0,

modifiedCount: 0,

upsertedCount: 0

}

studentdb> **db.student.updateMany({sid:30003},{$set:{sname:"SRINU",gender:"MALE"}}, {upsert:true})**

{

acknowledged: true,

insertedId: ObjectId('65c6f8f9b6b06ec4c2df2585'),

matchedCount: 0,

modifiedCount: 0,

upsertedCount: 1

}

studentdb> **db.student.find({})**

[

{

\_id: ObjectId('65c35f4917f49f353a12b665'),

sid: 30001,

sname: 'KLU',

gender: 'FEMALE'

},

{

\_id: ObjectId('65c3607617f49f353a12b666'),

sid: 30002,

sname: 'RAMA',

gender: 'FEMALE'

},

{

\_id: ObjectId('65c6f8f9b6b06ec4c2df2585'),

sid: 30003,

gender: 'MALE',

sname: 'SRINU'

}

]

studentdb> **db.student.find({},{sname:1})**

* **The first argument {} represents an empty query object, meaning there are no specific filter criteria, so it retrieves all documents.**
* **The second argument {sname: 1} is the projection object. It specifies which fields to include in the returned documents. In this case, it includes only the "sname" field with a value of 1, indicating that it should be included in the output.**
* **But each document will contain only the "\_id" field (which is included by default) and the "sname" field. Other fields will be excluded from the output.**

[

{ \_id: ObjectId('65c35f4917f49f353a12b665'), sname: 'KLU' },

{ \_id: ObjectId('65c3607617f49f353a12b666'), sname: 'RAMA' },

{ \_id: ObjectId('65c6f8f9b6b06ec4c2df2585'), sname: 'SRINU' }

]

studentdb> **db.student.find({},{sname:1,\_id:0})**

* **The first argument {} represents an empty query object, meaning there are no specific filter criteria, so it retrieves all documents.**
* **The second argument {sname: 1, \_id: 0} is the projection object. It specifies which fields to include or exclude in the returned documents.**
* **Here, the "sname" field is included with a value of 1, indicating it should be included in the output, and the "\_id" field is excluded with a value of 0, indicating it should not be included in the output.**

[ { sname: 'KLU' }, { sname: 'RAMA' }, { sname: 'SRINU' } ]

studentdb> **db.student.distinct("MALE")**

* **The .distinct() method in MongoDB is used to retrieve an array of distinct values for a specified field across all documents in a collection**

[]

studentdb> **db.student.distinct("gender")**

[ 'FEMALE', 'MALE' ]

studentdb> **db.student.find({}).size()**

3

studentdb> **db.student.find({gender:"MALE"}).size()**

1

studentdb> **db.student.find({gender:"FEMALE"}).size()**

2

studentdb> **db.student.countDocuments()**

3

studentdb> **db.student.countDocuments({})**

3

studentdb> **db.student.countDocuments({gender:"FEMALE"})**

2

studentdb> **db.createCollection("emp")**

* **createCollection is the method used to create the collection.**

{ ok: 1 }

studentdb> show collections

emp

student

studentdb> **db.emp.insertMany([{\_id:1,name:"A"},{\_id:2,name:"B"}])**

{ acknowledged: true, insertedIds: { '0': 1, '1': 2 } }

studentdb> **db.emp.insertMany([{\_id:3,name:"C"},{\_id:4,name:"C"}])**

{ acknowledged: true, insertedIds: { '0': 3, '1': 4 } }

studentdb> **db.emp.find({})**

[

{ \_id: 1, name: 'A' },

{ \_id: 2, name: 'B' },

{ \_id: 3, name: 'C' },

{ \_id: 4, name: 'C' },

]

studentdb> **db.emp.updateOne({\_id:1},{$set:{sal:10000.5}})**

{

acknowledged: true,

insertedId: null,

matchedCount: 1,

modifiedCount: 1,

upsertedCount: 0

}

studentdb> **db.emp.updateOne({\_id:2},{$set:{sal:20000.5}})**

{

acknowledged: true,

insertedId: null,

matchedCount: 1,

modifiedCount: 1,

upsertedCount: 0

}

studentdb> **db.emp.updateOne({\_id:3},{$set:{sal:30000.5}})**

{

acknowledged: true,

insertedId: null,

matchedCount: 1,

modifiedCount: 1,

upsertedCount: 0

}

studentdb> **db.emp.updateOne({\_id:4},{$set:{sal:40000.5}})**

{

acknowledged: true,

insertedId: null,

matchedCount: 1,

modifiedCount: 1,

upsertedCount: 0

}

studentdb> **db.emp.find({})**

[

{\_id: 1, name: 'A', sal: 10000.5 },

{\_id: 2, name: 'B', sal: 20000.5 },

{\_id: 3, name: 'C', sal: 30000.5 },

{\_id: 4, name: 'C', sal: 40000.5 }

]

studentdb> **db.emp.updateMany({},{$set:{gender:"FEMALE"}})**

{

acknowledged: true,

insertedId: null,

matchedCount: 4,

modifiedCount: 4,

upsertedCount: 0

}

studentdb> **db.emp.find({})**

[

{ \_id: 1, name: 'A', sal: 10000.5, gender: 'FEMALE' },

{ \_id: 2, name: 'B', sal: 20000.5, gender: 'FEMALE' },

{ \_id: 3, name: 'C', sal: 30000.5, gender: 'FEMALE' },

{ \_id: 4, name: 'C', sal: 40000.5, gender: 'FEMALE' }

]

studentdb> **db.emp.find({sal:{$lt:20000.0}})**

* **{$lt:20000.0}: This is a comparison operator.**
* **$lt stands for "less than", and 20000.0 is the value to compare against.**
* **So, the query will return documents where the value of the "sal" field is less than 20000.0.**

[ { \_id: 1, name: 'A', sal: 10000.5, gender: 'FEMALE' } ]

studentdb> **db.emp.find({sal:{$gt:30000.0}})**

* **{$gt:30000.0}: This is a comparison operator.**
* **$gt stands for "greater than", and 30000.0 is the value to compare against.**
* **So, the query will return documents where the value of the "sal" field is greater than 30000.0.**

[

{ \_id: 3, name: 'C', sal: 30000.5, gender: 'FEMALE' },

{ \_id: 4, name: 'C', sal: 40000.5, gender: 'FEMALE' }

]

studentdb> **db.emp.find({sal:{$lte:30000.5}}) 🡺 lte means less than or equal to**

[

{ \_id: 1, name: 'A', sal: 10000.5, gender: 'FEMALE' },

{ \_id: 2, name: 'B', sal: 20000.5, gender: 'FEMALE' },

{ \_id: 3, name: 'C', sal: 30000.5, gender: 'FEMALE' }

]

studentdb> **db.emp.find({sal:{$gte:30000.5}}) 🡺 gte means greater than or equal to**

[

{ \_id: 3, name: 'C', sal: 30000.5, gender: 'FEMALE' },

{ \_id: 4, name: 'C', sal: 40000.5, gender: 'FEMALE' }

]

studentdb> **db.emp.find({sal:{$ne:30000.5}})** 🡺 **ne means not equal to**

[

{ \_id: 1, name: 'A', sal: 10000.5, gender: 'FEMALE' },

{ \_id: 2, name: 'B', sal: 20000.5, gender: 'FEMALE' },

{ \_id: 4, name: 'C', sal: 40000.5, gender: 'FEMALE' }

]

studentdb> **db.emp.find({})**

[

{ \_id: 1, name: 'A', sal: 10000.5, gender: 'FEMALE' },

{ \_id: 2, name: 'B', sal: 20000.5, gender: 'FEMALE' },

{ \_id: 3, name: 'C', sal: 30000.5, gender: 'FEMALE' },

{ \_id: 4, name: 'C', sal: 40000.5, gender: 'FEMALE' }

]

studentdb> **db.emp.find().sort() 🡺 sort() method will sort the documents in ascending.**

[

{ \_id: 1, name: 'A', sal: 10000.5, gender: 'FEMALE' },

{ \_id: 2, name: 'B', sal: 20000.5, gender: 'FEMALE' },

{ \_id: 3, name: 'C', sal: 30000.5, gender: 'FEMALE' },

{ \_id: 4, name: 'C', sal: 40000.5, gender: 'FEMALE' }

]

studentdb> **db.emp.find().sort({sal:1}) 🡺 sort({sal:1}): Here, 1 represents in ascending.**

[

{ \_id: 1, name: 'A', sal: 10000.5, gender: 'FEMALE' },

{ \_id: 2, name: 'B', sal: 20000.5, gender: 'FEMALE' },

{ \_id: 3, name: 'C', sal: 30000.5, gender: 'FEMALE' },

{ \_id: 4, name: 'C', sal: 40000.5, gender: 'FEMALE' }

]

studentdb> **db.emp.find().sort({sal:-1})** 🡺 **sort({sal:-1}): Here, -1 represents in descending.**

[

{ \_id: 4, name: 'C', sal: 40000.5, gender: 'FEMALE' },

{ \_id: 3, name: 'C', sal: 30000.5, gender: 'FEMALE' },

{ \_id: 2, name: 'B', sal: 20000.5, gender: 'FEMALE' },

{ \_id: 1, name: 'A', sal: 10000.5, gender: 'FEMALE' }

]

studentdb> **db.emp.find({}).sort({sal:-1}).limit(1)**

[ { \_id: 4, name: 'C', sal: 40000.5, gender: 'FEMALE' } ]

studentdb> **db.emp.find({}, {name:1}).sort({sal:-1}).limit(1)**

[ { \_id: 4, name: 'C' } ]

studentdb> **db.emp.find({})**

[

{ \_id: 1, name: 'A', sal: 10000.5, gender: 'FEMALE' },

{ \_id: 2, name: 'B', sal: 20000.5, gender: 'FEMALE' },

{ \_id: 3, name: 'C', sal: 30000.5, gender: 'FEMALE' },

{ \_id: 4, name: 'C', sal: 40000.5, gender: 'FEMALE' }

]

studentdb> **db.emp.updateMany({sal:{$gt:20000}},{$set:{gender:"MALE"}})**

{

acknowledged: true,

insertedId: null,

matchedCount: 3,

modifiedCount: 3,

upsertedCount: 0

}

studentdb> **db.emp.find({})**

[

{ \_id: 1, name: 'A', sal: 10000.5, gender: 'FEMALE' },

{ \_id: 2, name: 'B', sal: 20000.5, gender: 'MALE' },

{ \_id: 3, name: 'C', sal: 30000.5, gender: 'MALE' },

{ \_id: 4, name: 'C', sal: 40000.5, gender: 'MALE' }

]

studentdb> **db.emp.find({gender:{$in:["MALE", "FEMALE"]}})**

* **The $in operator selects the documents where the value of the field is equal to any value in the specified array.**
* **The $in operator is useful when you want to filter documents based on multiple possible values for a particular field.**
* **This query will return all documents from the "emp" collection where the gender is either "MALE" or "FEMALE"**

[

{ \_id: 1, name: 'A', sal: 10000.5, gender: 'FEMALE' },

{ \_id: 2, name: 'B', sal: 20000.5, gender: 'MALE' },

{ \_id: 3, name: 'C', sal: 30000.5, gender: 'MALE' },

{ \_id: 4, name: 'C', sal: 40000.5, gender: 'MALE' }

]

studentdb> **db.emp.find({gender:{$in:["MALE"]}})**

* **This query will return all documents from the "emp" collection where the gender is only "MALE" .**

[

{ \_id: 2, name: 'B', sal: 20000.5, gender: 'MALE' },

{ \_id: 3, name: 'C', sal: 30000.5, gender: 'MALE' },

{ \_id: 4, name: 'C', sal: 40000.5, gender: 'MALE' }

]

studentdb> **db.emp.find({gender:{$nin:["MALE"]}})**

* **The $nin operator in MongoDB is the negation of the $in operator.**
* **It is used to query documents where the value of a field does not match any value in the specified array.**

[ { \_id: 1, name: 'A', sal: 10000.5, gender: 'FEMALE' } ]

studentdb> **db.emp.find({})**

[

{ \_id: 1, name: 'A', sal: 10000.5, gender: 'FEMALE' },

{ \_id: 2, name: 'B', sal: 20000.5, gender: 'MALE' },

{ \_id: 3, name: 'C', sal: 30000.5, gender: 'MALE' },

{ \_id: 4, name: 'C', sal: 40000.5, gender: 'MALE' }

]

studentdb> **db.emp.renameCollection("employee")**

* **The .renameCollection() method in MongoDB is used to rename an existing collection within a database.**

{ ok: 1 }

studentdb> **show collections**

employee

student

studentdb> **db.employee.renameCollection("emp")**

{ ok: 1 }

studentdb> **show collections**

emp

student

studentdb> **db.emp.find({$and:[{name:"C"},{sal:{$gt:30000.5}}]})**

[ { \_id: 4, name: 'C', sal: 40000.5, gender: 'MALE' } ]

studentdb> **db.emp.find({$or:[{name:"C"},{sal:{$lt:40000}}]})**

[

{ \_id: 1, name: 'A', sal: 10000.5, gender: 'FEMALE' },

{ \_id: 2, name: 'B', sal: 20000.5, gender: 'MALE' },

{ \_id: 3, name: 'C', sal: 30000.5, gender: 'MALE' },

{ \_id: 4, name: 'C', sal: 40000.5, gender: 'MALE' }

]

studentdb> **db.emp.updateMany({},{$rename:{sal:"salary"}})**

* **{$rename:{sal:"salary"}}: This is the update operation to be performed on the selected documents. It uses the $rename operator to rename the field sal to salary.**

{

acknowledged: true,

insertedId: null,

matchedCount: 4,

modifiedCount: 4,

upsertedCount: 0

}

studentdb> **show collections**

emp

student

studentdb> **db.emp.find({})**

[

{ \_id: 1, name: 'A', gender: 'FEMALE', salary: 10000.5 },

{ \_id: 2, name: 'B', gender: 'MALE', salary: 20000.5 },

{ \_id: 3, name: 'C', gender: 'MALE', salary: 30000.5 },

{ \_id: 4, name: 'C', gender: 'MALE', salary: 40000.5 }

]

studentdb> **db.emp.find({}).sort({salary:-1}).limit(1)**

[ { \_id: 4, name: 'C', gender: 'MALE', salary: 40000.5 } ]

studentdb> **db.emp.find({},{name:1}).sort({salary:-1}).limit(1)**

* **({},{name:1}):** **The first parameter {} represents the filter criteria, which is empty in this case, meaning all documents will be selected.**
* **The second parameter {name:1}:** **specifies the fields to include in the result. It includes only the name field with a value of 1, which means it will be included in the output.**
* **.sort({salary:-1}):** **This method sorts the documents based on the salary field in descending order (-1).**
* **.limit(1):** **This method limits the number of documents returned to just one, effectively returning only the document with the highest salary.**

[ { \_id: 4, name: 'C' } ]

studentdb> **db.emp.find({},{name:1,\_id:0}).sort({salary:-1}).limit(1)**

[ { name: 'C' } ]

studentdb> **// Aggregate Functions or Multi-Document Functions**

studentdb> **db.emp.aggregate({$count:"totalcount"})**

* **db.emp.aggregate: This is the syntax for initiating an aggregation operation on the "emp" collection in the MongoDB database.**
* **{ $count}: This is the aggregation pipeline stage. The $count stage counts the number of documents that pass through the pipeline and returns a single document containing the count.**
* **The "totalcount" is where the count will be stored.**

[ { totalcount: 4 } ]

studentdb> **db.emp.find({})**

[

{ \_id: 1, name: 'A', gender: 'FEMALE', salary: 10000.5 },

{ \_id: 2, name: 'B', gender: 'MALE', salary: 20000.5 },

{ \_id: 3, name: 'C', gender: 'MALE', salary: 30000.5 },

{ \_id: 4, name: 'C', gender: 'MALE', salary: 40000.5 }

]

studentdb> **db.emp.aggregate({$group:{\_id:null,totalsal:{$sum:"$salary"}}})**

* **$group: This is an aggregation pipeline stage that groups documents by a specified expression.**
* **In this case, it's grouping by a constant value (null), which means it groups all documents into a single group.**
* **$sum operator is used to calculate the sum of numeric values within documents.**

[ { \_id: null, totalsal: 100002 } ]

studentdb> **db.emp.aggregate({$group:{\_id:"$gender",totalsal:{$sum:"$salary"}}})**

[

{ \_id: 'MALE', totalsal: 90001.5 },

{ \_id: 'FEMALE', totalsal: 10000.5 }

]

studentdb> **db.emp.aggregate({$group:{\_id:"$name",totalsal:{$sum:"$salary"}}})**

[

{ \_id: 'A', totalsal: 10000.5 },

{ \_id: 'B', totalsal: 20000.5 },

{ \_id: 'C', totalsal: 70001 }

]

studentdb> **db.emp.aggregate({$group:{\_id:null,totalsal:{$sum:"$salary"}}})**

[ { \_id: null, totalsal: 100002 } ]

studentdb> **db.emp.aggregate({$group:{\_id:null,avgsal:{$avg:"$salary"}}})**

[ { \_id: null, totalsal: 25000.5 } ]

studentdb> **db.emp.aggregate({$group:{\_id:null,minsal:{$min:"$salary"}}})**

[ { \_id: null, minsal: 10000.5 } ]

studentdb> **db.emp.aggregate({$group:{\_id:null,maxsal:{$max:"$salary"}}})**

[ { \_id: null, maxsal: 40000.5 } ]

studentdb> **db.emp.aggregate({$group:{\_id:"$gender",totalsal:{$sum:"$salary"}}})**

[

{ \_id: 'MALE', totalsal: 90001.5 },

{ \_id: 'FEMALE', totalsal: 10000.5 }

]

studentdb> **db.emp.aggregate({$group:{\_id:"$gender",avgsal:{$avg:"$salary"}}})**

[

{ \_id: 'FEMALE', avgsal: 10000.5 },

{ \_id: 'MALE', avgsal: 30000.5 }

]

studentdb> **db.emp.aggregate({$group:{\_id:"$gender",minsal:{$min:"$salary"}}})**

[

{ \_id: 'MALE', minsal: 20000.5 },

{ \_id: 'FEMALE', minsal: 10000.5 }

]

studentdb> **db.emp.aggregate({$group:{\_id:"$gender",maxsal:{$max:"$salary"}}})**

[

{ \_id: 'MALE', maxsal: 40000.5 },

{ \_id: 'FEMALE', maxsal: 10000.5 }

]

studentdb> **db.emp.aggregate({$match:{gender:"MALE"}})**

[

{ \_id: 2, name: 'B', gender: 'MALE', salary: 20000.5 },

{ \_id: 3, name: 'C', gender: 'MALE', salary: 30000.5 },

{ \_id: 4, name: 'C', gender: 'MALE', salary: 40000.5 }

]

studentdb> **db.emp.aggregate([{$match:{gender:"MALE"}}, {$group:{\_id:"$gender",maxsal:{$max:"$salary"}}}])**

[ { \_id: 'MALE', maxsal: 40000.5 } ]

studentdb> **db.emp.aggregate([{$match:{gender:"MALE"}},** **{$group:{\_id:"$gender",totalsal:{$sum:"$salary"}}}])**

[ { \_id: 'MALE', totalsal: 90001.5 } ]

studentdb> **db.emp.aggregate([{$match:{gender:"MALE"}},** **{$group:{\_id:null,totalsal:{$sum:"$salary"}}}])**

[ { \_id: null, totalsal: 90001.5 } ]

studentdb> **db.emp.aggregate([{$match:{}}, {$group:{\_id:"$gender",totalsal:{$sum:"$salary"}}}])**

[

{ \_id: 'FEMALE', totalsal: 10000.5 },

{ \_id: 'MALE', totalsal: 90001.5 }

]

studentdb> **db.emp.aggregate([{$match:{}}, {$group:{\_id:"$gender",totalsal:{$sum:"$salary"}}},** **{$project:{\_id:0,}} ])**

[ { totalsal: 10000.5 }, { totalsal: 90001.5 } ]

studentdb> **db.emp.aggregate([{$match:{}}, {$group:{\_id:"$gender",totalsal:{$sum:"$salary"}}},** **{$project:{\_id:0,}} ])**

[ { totalsal: 90001.5 }, { totalsal: 10000.5 } ]

studentdb> **db.emp.aggregate([{$match:{}}, {$group:{\_id:"$gender",totalsal:{$sum:"$salary"}}}, {$project:{\_id:0}}, {$sort:{totalsal:1}} ])**

[ { totalsal: 10000.5 }, { totalsal: 90001.5 } ]

studentdb> **db.emp.aggregate([{$match:{}}, {$group:{\_id:"$gender",totalsal:{$sum:"$salary"}}}, {$project:{\_id:0}}, {$sort:{totalsal:-1}} ])**

[ { totalsal: 90001.5 }, { totalsal: 10000.5 } ]

studentdb**> db.emp.aggregate([{$match:{}}, {$group:{\_id:"$gender",totalsal:{$sum:"$salary"}}}, {$project:{\_id:0}}, {$sort:{totalsal:-1}},{$limit:1} ])**

[ { totalsal: 90001.5 } ]

studentdb> **db.emp.find({})**

[

{ \_id: 1, name: 'A', gender: 'FEMALE', salary: 10000.5 },

{ \_id: 2, name: 'B', gender: 'MALE', salary: 20000.5 },

{ \_id: 3, name: 'C', gender: 'MALE', salary: 30000.5 },

{ \_id: 4, name: 'C', gender: 'MALE', salary: 40000.5 }

]

studentdb> **db.emp.updateMany({}, {$set:{location:"VIZAG"}})**

{

acknowledged: true,

insertedId: null,

matchedCount: 4,

modifiedCount: 4,

upsertedCount: 0

}

studentdb> **db.emp.find({})**

[

{

\_id: 1,

name: 'A',

gender: 'FEMALE',

salary: 10000.5,

location: 'VIZAG'

},

{

\_id: 2,

name: 'B',

gender: 'MALE',

salary: 20000.5,

location: 'VIZAG'

},

{

\_id: 3,

name: 'C',

gender: 'MALE',

salary: 30000.5,

location: 'VIZAG'

},

{

\_id: 4,

name: 'C',

gender: 'MALE',

salary: 40000.5,

location: 'VIZAG'

}

]

studentdb> **db.emp.updateMany({}, {$unset:{location:""}})**

{

acknowledged: true,

insertedId: null,

matchedCount: 4,

modifiedCount: 4,

upsertedCount: 0

}

studentdb> **db.emp.find({})**

[

{ \_id: 1, name: 'A', gender: 'FEMALE', salary: 10000.5 },

{ \_id: 2, name: 'B', gender: 'MALE', salary: 20000.5 },

{ \_id: 3, name: 'C', gender: 'MALE', salary: 30000.5 },

{ \_id: 4, name: 'C', gender: 'MALE', salary: 40000.5 }

]

studentdb> **db.emp.find({salary:{$not:{$gt:20000}}})**

[ { \_id: 1, name: 'A', gender: 'FEMALE', salary: 10000.5 } ]

studentdb> **db.emp.aggregate([{$match:{}},{$out:"faculty"}])**

studentdb> **show collections**

emp

faculty

student

studentdb> **db.faculty.find({})**

[

{ \_id: 1, name: 'A', gender: 'FEMALE', salary: 10000.5 },

{ \_id: 2, name: 'B', gender: 'MALE', salary: 20000.5 },

{ \_id: 3, name: 'C', gender: 'MALE', salary: 30000.5 },

{ \_id: 4, name: 'C', gender: 'MALE', salary: 40000.5 }

]