**Task - MongoDB - CRUD Operations #45**

[akash-coded](https://github.com/akash-coded) started this conversation in [Tasks](https://github.com/akash-coded/mern/discussions/categories/tasks)

https://github.com/akash-coded/mern/discussions/45

# Start form JS - Classes and Objects - Class Declaration & Object Creation #160

[akash-coded](https://github.com/akash-coded) started this conversation in [Tasks](https://github.com/akash-coded/mern/discussions/categories/tasks)

[on Apr 22](https://github.com/akash-coded/mern/discussions/160#discussion-5118377) **2023**

to

https://github.com/akash-coded/mern/discussions/205

**Guided Capstone Project - MERN - Meeting App - Meet in Middle #205**

[akash-coded](https://github.com/akash-coded) started this conversation in [Tasks](https://github.com/akash-coded/mern/discussions/categories/tasks)

edited

[**on Oct 1**](https://github.com/akash-coded/mern/discussions/205#discussion-5690333) **2023**

edited

**[[A person sitting at a desk with a computer

Description automatically generated](https://github.com/akash-coded)](https://github.com/akash-coded)**

**[akash-coded](https://github.com/akash-coded)**

[**on Jun 12, 2022**](https://github.com/akash-coded/mern/discussions/45#discussion-4138422)

**Maintainer**

|  |
| --- |
| **MongoDB Practice**  MongoDB Exercise in mongo shell  Connect to a running mongo instance, use a database named mongo\_practice.  Document all your queries in a javascript file to use as a reference.  **Insert Documents**  Insert the following documents into a movies collection.  title : Fight Club  writer : Chuck Palahniuk  year : 1999  actors : [  Brad Pitt  Edward Norton  ]  title : Pulp Fiction  writer : Quentin Tarantino  year : 1994  actors : [  John Travolta  Uma Thurman  ]  title : Inglorious Basterds  writer : Quentin Tarantino  year : 2009  actors : [  Brad Pitt  Diane Kruger  Eli Roth  ]  title : The Hobbit: An Unexpected Journey  writer : J.R.R. Tolkein  year : 2012  franchise : The Hobbit  title : The Hobbit: The Desolation of Smaug  writer : J.R.R. Tolkein  year : 2013  franchise : The Hobbit  title : The Hobbit: The Battle of the Five Armies  writer : J.R.R. Tolkein  year : 2012  franchise : The Hobbit  synopsis : Bilbo and Company are forced to engage in a war against an array of combatants and keep the Lonely Mountain from falling into the hands of a rising darkness.  title : Pee Wee Herman's Big Adventure  title : Avatar  **Query / Find Documents**  query the movies collection to   1. get all documents 2. get all documents with writer set to "Quentin Tarantino" 3. get all documents where actors include "Brad Pitt" 4. get all documents with franchise set to "The Hobbit" 5. get all movies released in the 90s 6. get all movies released before the year 2000 or after 2010   **Update Documents**   1. add a synopsis to "The Hobbit: An Unexpected Journey" : "A reluctant hobbit, Bilbo Baggins, sets out to the Lonely Mountain with a spirited group of dwarves to reclaim their mountain home - and the gold within it - from the dragon Smaug." 2. add a synopsis to "The Hobbit: The Desolation of Smaug" : "The dwarves, along with Bilbo Baggins and Gandalf the Grey, continue their quest to reclaim Erebor, their homeland, from Smaug. Bilbo Baggins is in possession of a mysterious and magical ring." 3. add an actor named "Samuel L. Jackson" to the movie "Pulp Fiction"   **Text Search**   1. find all movies that have a synopsis that contains the word "Bilbo" 2. find all movies that have a synopsis that contains the word "Gandalf" 3. find all movies that have a synopsis that contains the word "Bilbo" and not the word "Gandalf" 4. find all movies that have a synopsis that contains the word "dwarves" or "hobbit" 5. find all movies that have a synopsis that contains the word "gold" and "dragon"   **Delete Documents**   1. delete the movie "Pee Wee Herman's Big Adventure" 2. delete the movie "Avatar"   **Relationships**  **Insert the following documents into a users collection**  username : GoodGuyGreg  first\_name : "Good Guy"  last\_name : "Greg"  username : ScumbagSteve  full\_name :  first : "Scumbag"  last : "Steve"  **Insert the following documents into a posts collection**  username : GoodGuyGreg  title : Passes out at party  body : Wakes up early and cleans house  username : GoodGuyGreg  title : Steals your identity  body : Raises your credit score  username : GoodGuyGreg  title : Reports a bug in your code  body : Sends you a Pull Request  username : ScumbagSteve  title : Borrows something  body : Sells it  username : ScumbagSteve  title : Borrows everything  body : The end  username : ScumbagSteve  title : Forks your repo on github  body : Sets to private  **Insert the following documents into a comments collection**  username : GoodGuyGreg  comment : Hope you got a good deal!  post : [post\_obj\_id]  where [post\_obj\_id] is the ObjectId of the posts document: "Borrows something"  username : GoodGuyGreg  comment : What's mine is yours!  post : [post\_obj\_id]  where [post\_obj\_id] is the ObjectId of the posts document: "Borrows everything"  username : GoodGuyGreg  comment : Don't violate the licensing agreement!  post : [post\_obj\_id]  where [post\_obj\_id] is the ObjectId of the posts document: "Forks your repo on github  username : ScumbagSteve  comment : It still isn't clean  post : [post\_obj\_id]  where [post\_obj\_id] is the ObjectId of the posts document: "Passes out at party"  username : ScumbagSteve  comment : Denied your PR cause I found a hack  post : [post\_obj\_id]  where [post\_obj\_id] is the ObjectId of the posts document: "Reports a bug in your code"  **Querying related collections**   1. find all users 2. find all posts 3. find all posts that was authored by "GoodGuyGreg" 4. find all posts that was authored by "ScumbagSteve" 5. find all comments 6. find all comments that was authored by "GoodGuyGreg" 7. find all comments that was authored by "ScumbagSteve" 8. find all comments belonging to the post "Reports a bug in your code" |

> show dbs

admin 0.000GB

config 0.000GB

local 0.000GB

> use mongo\_practice

switched to db mongo\_practice

> db.movies.insertMany([{title: "Fight Club", writer: "Chuck Palahniuk", year: 1999, actors: ["Brad Pitt", "EEdward Norton"]},

{title: "Pulp Fiction", writer: "Quentin Tarantino", year: 1994, actors: ["John Travolta", "Uma Thurman"]},

{title: "Inglorius Basterds", writer: "Quentin Tarantino", year: 2009, actors: ["Brad Pitt", "Diane Kruger", "Eli Roth"]},

{title: "The Hobbit: An Unexpected Journey", writer: "J.R.R. Tolkein", year: 2012, franchise: "The Hobbit"},

{title: "The Hobbit: The Desolation of smug", writer: "J.R.R. Tolkein", year: 2013, franchise: "The Hobbit"},

{title: "The Hobbit: The Battle of the Five Armies", writer: "J.R.R. Tolkein", year: 2012, franchise: "The Hobbit",

synopsis: "Bilbo and Company are forced to engage in a war against an array of combatants and keep the the lonely mountains from falling into the hands of a rising darkness."},

{title: "Pee Wee Herman's Big Adventure"}, {title: "Avatar"}]);

{

"acknowledged" : true,

"insertedIds" : [

ObjectId("62a78b4b7960f45ec00b89bb"),

ObjectId("62a78b4b7960f45ec00b89bc"),

ObjectId("62a78b4b7960f45ec00b89bd"),

ObjectId("62a78b4b7960f45ec00b89be"),

ObjectId("62a78b4b7960f45ec00b89bf"),

ObjectId("62a78b4b7960f45ec00b89c0"),

ObjectId("62a78b4b7960f45ec00b89c1"),

ObjectId("62a78b4b7960f45ec00b89c2")

]

}

> db.movies.find({}).pretty();

{

"\_id" : ObjectId("62a78b4b7960f45ec00b89bb"),

"title" : "Fight Club",

"writer" : "Chuck Palahniuk",

"year" : 1999,

"actors" : [

"Brad Pitt",

"EEdward Norton"

]

}

{

"\_id" : ObjectId("62a78b4b7960f45ec00b89bc"),

"title" : "Pulp Fiction",

"writer" : "Quentin Tarantino",

"year" : 1994,

"actors" : [

"John Travolta",

"Uma Thurman"

]

}

{

"\_id" : ObjectId("62a78b4b7960f45ec00b89bd"),

"title" : "Inglorius Basterds",

"writer" : "Quentin Tarantino",

"year" : 2009,

"actors" : [

"Brad Pitt",

"Diane Kruger",

"Eli Roth"

]

}

{

"\_id" : ObjectId("62a78b4b7960f45ec00b89be"),

"title" : "The Hobbit: An Unexpected Journey",

"writer" : "J.R.R. Tolkein",

"year" : 2012,

"franchise" : "The Hobbit"

}

{

"\_id" : ObjectId("62a78b4b7960f45ec00b89bf"),

"title" : "The Hobbit: The Desolation of smug",

"writer" : "J.R.R. Tolkein",

"year" : 2013,

"franchise" : "The Hobbit"

}

{

"\_id" : ObjectId("62a78b4b7960f45ec00b89c0"),

"title" : "The Hobbit: The Battle of the Five Armies",

"writer" : "J.R.R. Tolkein",

"year" : 2012,

"franchise" : "The Hobbit",

"synopsis" : "Bilbo and Company are forced to engage in a war against an array of combatants and keep the the lonely mountains from falling into the hands of a rising darkness."

}

{

"\_id" : ObjectId("62a78b4b7960f45ec00b89c1"),

"title" : "Pee Wee Herman's Big Adventure"

}

{ "\_id" : ObjectId("62a78b4b7960f45ec00b89c2"),

"title" : "Avatar"

}

> db.movies.find( { writer : { $eq : "Quentin Tarantino" }}).pretty();

{

"\_id" : ObjectId("62a78b4b7960f45ec00b89bc"),

"title" : "Pulp Fiction",

"writer" : "Quentin Tarantino",

"year" : 1994,

"actors" : [

"John Travolta",

"Uma Thurman"

]

}

{

"\_id" : ObjectId("62a78b4b7960f45ec00b89bd"),

"title" : "Inglorius Basterds",

"writer" : "Quentin Tarantino",

"year" : 2009,

"actors" : [

"Brad Pitt",

"Diane Kruger",

"Eli Roth"

]

}

> db.movies.find( { actors : { $in : ["Brad Pitt"]}}).pretty();

{

"\_id" : ObjectId("62a78b4b7960f45ec00b89bb"),

"title" : "Fight Club",

"writer" : "Chuck Palahniuk",

"year" : 1999,

"actors" : [

"Brad Pitt",

"EEdward Norton"

]

}

{

"\_id" : ObjectId("62a78b4b7960f45ec00b89bd"),

"title" : "Inglorius Basterds",

"writer" : "Quentin Tarantino",

"year" : 2009,

"actors" : [

"Brad Pitt",

"Diane Kruger",

"Eli Roth"

]

}

> db.movies.find( { franchise : { $eq : "The Hobbit" }}).pretty();

{

"\_id" : ObjectId("62a78b4b7960f45ec00b89be"),

"title" : "The Hobbit: An Unexpected Journey",

"writer" : "J.R.R. Tolkein",

"year" : 2012,

"franchise" : "The Hobbit"

}

{

"\_id" : ObjectId("62a78b4b7960f45ec00b89bf"),

"title" : "The Hobbit: The Desolation of smug",

"writer" : "J.R.R. Tolkein",

"year" : 2013,

"franchise" : "The Hobbit"

}

{

"\_id" : ObjectId("62a78b4b7960f45ec00b89c0"),

"title" : "The Hobbit: The Battle of the Five Armies",

"writer" : "J.R.R. Tolkein",

"year" : 2012,

"franchise" : "The Hobbit",

"synopsis" : "Bilbo and Company are forced to engage in a war against an array of combatants

and keep the the lonely mountains from falling into the hands of a rising darkness."

}

> db.movies.find({ $and : [ {year : { $gte :1990 }}, { year :{ $lte : 1999 }} ] }).pretty();

{

"\_id" : ObjectId("62a78b4b7960f45ec00b89bb"),

"title" : "Fight Club",

"writer" : "Chuck Palahniuk",

"year" : 1999,

"actors" : [

"Brad Pitt",

"EEdward Norton"

]

}

{

"\_id" : ObjectId("62a78b4b7960f45ec00b89bc"),

"title" : "Pulp Fiction",

"writer" : "Quentin Tarantino",

"year" : 1994,

"actors" : [

"John Travolta",

"Uma Thurman"

]

}

> db.movies.find({ $or : [ {year : { $gt :2010 }}, { year :{ $lt : 2000 }} ] }).pretty();

{

"\_id" : ObjectId("62a78b4b7960f45ec00b89bb"),

"title" : "Fight Club",

"writer" : "Chuck Palahniuk",

"year" : 1999,

"actors" : [

"Brad Pitt",

"EEdward Norton"

]

}

{

"\_id" : ObjectId("62a78b4b7960f45ec00b89bc"),

"title" : "Pulp Fiction",

"writer" : "Quentin Tarantino",

"year" : 1994,

"actors" : [

"John Travolta",

"Uma Thurman"

]

}

{

"\_id" : ObjectId("62a78b4b7960f45ec00b89be"),

"title" : "The Hobbit: An Unexpected Journey",

"writer" : "J.R.R. Tolkein",

"year" : 2012,

"franchise" : "The Hobbit"

}

{

"\_id" : ObjectId("62a78b4b7960f45ec00b89bf"),

"title" : "The Hobbit: The Desolation of smug",

"writer" : "J.R.R. Tolkein",

"year" : 2013,

"franchise" : "The Hobbit"

}

{

"\_id" : ObjectId("62a78b4b7960f45ec00b89c0"),

"title" : "The Hobbit: The Battle of the Five Armies",

"writer" : "J.R.R. Tolkein",

"year" : 2012,

"franchise" : "The Hobbit",

"synopsis" : "Bilbo and Company are forced to engage in a war against an array of combatants

and keep the the lonely mountains from falling into the hands of a rising darkness."

}

## Update Documents

> db.movies.updateOne({"title":"The Hobbit: An Unexpected Journey"},{$set:{"synopsis":"A reluctant hobbit, Bilbo Baggins, sets out to the Lonely Mountain with a spirited group of dwarves to reclaim their mountain home - and the gold within it - from the dragon Smaug."}});

{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }

> db.movies.updateOne({"title":"The Hobbit: The Desolation of smug"},{$set:{"synopsis":"The dwarves, along with Bilbo Baggins and Gandalf the Grey, continue their quest to reclaim Erebor, their homeland, from Smaug. Bilbo Baggins is in possession of a mysterious and magical ring."}});

{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }

> db.movies.updateOne({"title":"Pulp Fiction"},{$push:{"actors":"Samuel L. Jackson"}});

{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }

## Text Search

> db.movies.find({"synopsis":{"$regex":"Bilbo"}}).pretty();

{

"\_id" : ObjectId("62a78b4b7960f45ec00b89be"),

"title" : "The Hobbit: An Unexpected Journey",

"writer" : "J.R.R. Tolkein",

"year" : 2012,

"franchise" : "The Hobbit",

"synopsis" : "A reluctant hobbit, Bilbo Baggins, sets out to the Lonely Mountain with a spirited group of dwarves to reclaim their mountain home - and the gold within it - from the dragon Smaug."

}

{

"\_id" : ObjectId("62a78b4b7960f45ec00b89bf"),

"title" : "The Hobbit: The Desolation of smug",

"writer" : "J.R.R. Tolkein",

"year" : 2013,

"franchise" : "The Hobbit",

"synopsis" : "The dwarves, along with Bilbo Baggins and Gandalf the Grey, continue their quest to reclaim Erebor, their homeland, from Smaug. Bilbo Baggins is in possession of a mysterious and magical ring."

}

{

"\_id" : ObjectId("62a78b4b7960f45ec00b89c0"),

"title" : "The Hobbit: The Battle of the Five Armies",

"writer" : "J.R.R. Tolkein",

"year" : 2012,

"franchise" : "The Hobbit",

"synopsis" : "Bilbo and Company are forced to engage in a war against an array of combatants and keep the the lonely mountains from falling into the hands of a rising darkness."

}

> db.movies.find({"synopsis":{"$regex":"Gandalf"}}).pretty();

{

"\_id" : ObjectId("62a78b4b7960f45ec00b89bf"),

"title" : "The Hobbit: The Desolation of smug",

"writer" : "J.R.R. Tolkein",

"year" : 2013,

"franchise" : "The Hobbit",

"synopsis" : "The dwarves, along with Bilbo Baggins and Gandalf the Grey, continue their quest to reclaim Erebor, their homeland, from Smaug. Bilbo Baggins is in possession of a mysterious and magical ring."

}

> db.movies.find({"synopsis":{"$regex":"Bilbo",$not:{"$regex":"Gandalf"}}}).pretty();

{

"\_id" : ObjectId("62a78b4b7960f45ec00b89be"),

"title" : "The Hobbit: An Unexpected Journey",

"writer" : "J.R.R. Tolkein",

"year" : 2012,

"franchise" : "The Hobbit",

"synopsis" : "A reluctant hobbit, Bilbo Baggins, sets out to the Lonely Mountain with a spirited group of dwarves to reclaim their mountain home - and the gold within it - from the dragon Smaug."

}

{

"\_id" : ObjectId("62a78b4b7960f45ec00b89c0"),

"title" : "The Hobbit: The Battle of the Five Armies",

"writer" : "J.R.R. Tolkein",

"year" : 2012,

"franchise" : "The Hobbit",

"synopsis" : "Bilbo and Company are forced to engage in a war against an array of combatants and keep the the lonely mountains from falling into the hands of a rising darkness."

}

> db.movies.find({"$or":[{"synopsis":{"$regex":"dwarves"}},{"synopsis":{"$regex":"hobbit"}}]}).pretty();

{

"\_id" : ObjectId("62a78b4b7960f45ec00b89be"),

"title" : "The Hobbit: An Unexpected Journey",

"writer" : "J.R.R. Tolkein",

"year" : 2012,

"franchise" : "The Hobbit",

"synopsis" : "A reluctant hobbit, Bilbo Baggins, sets out to the Lonely Mountain with a spirited group of dwarves to reclaim their mountain home - and the gold within it - from the dragon Smaug."

}

{

"\_id" : ObjectId("62a78b4b7960f45ec00b89bf"),

"title" : "The Hobbit: The Desolation of smug",

"writer" : "J.R.R. Tolkein",

"year" : 2013,

"franchise" : "The Hobbit",

"synopsis" : "The dwarves, along with Bilbo Baggins and Gandalf the Grey, continue their quest to reclaim Erebor, their homeland, from Smaug. Bilbo Baggins is in possession of a mysterious and magical ring."

}

> db.movies.find({"$and":[{"synopsis":{"$regex":"gold"}},{"synopsis":{"$regex":"dragon"}}]}).pretty();

{

"\_id" : ObjectId("62a78b4b7960f45ec00b89be"),

"title" : "The Hobbit: An Unexpected Journey",

"writer" : "J.R.R. Tolkein",

"year" : 2012,

"franchise" : "The Hobbit",

"synopsis" : "A reluctant hobbit, Bilbo Baggins, sets out to the Lonely Mountain with a spirited group of dwarves to reclaim their mountain home - and the gold within it - from the dragon Smaug."

}

## Delete Documents

> db.movies.deleteOne({"title":"Pee Wee Herman's Big Adventure"});

{ "acknowledged" : true, "deletedCount" : 1 }

> db.movies.deleteOne({"title":"Avatar"});

{ "acknowledged" : true, "deletedCount" : 1 }

> db.movies.find({},{\_id:0,title:1}).pretty();

{ "title" : "Fight Club" }

{ "title" : "Pulp Fiction" }

{ "title" : "Inglorius Basterds" }

{ "title" : "The Hobbit: An Unexpected Journey" }

{ "title" : "The Hobbit: The Desolation of smug" }

{ "title" : "The Hobbit: The Battle of the Five Armies" }

## Relationships

> db.users.insertMany([{"username":"GoodGuyGreg","first\_name":"Good Guy", "last\_name" : "Greg"},{ "username" : "ScumbagSteve", "full\_name" : {"first" : "Scumbag" , "last" : "Steve"}}]);

{

"acknowledged" : true,

"insertedIds" : [

ObjectId("62b2127458e2771bd28580eb"),

ObjectId("62b2127458e2771bd28580ec")

]

}

### Insert the following documents into a posts collection

> db.posts.insertMany([{"username":"GoodGuyGreg","title" :" Passes out at party", "body" :" Wakes up early and cleans house"},{"username":"GoodGuyGreg","title" :"Steals your identity", "body" :"Raises your credit score"},{"username":"GoodGuyGreg","title" :" Reports a bug in your code", "body" :" Sends you a Pull Request"},{"username":"ScumbagSteve","title" :" Borrows something", "body" :"Sells it"},{"username":"ScumbagSteve","title" :"Borrows everything", "body" :"The end"},{"username":"ScumbagSteve","title" :"Forks your repo on github", "body" :"Sets to private"}]);

{

"acknowledged" : true,

"insertedIds" : [

ObjectId("62b2131f58e2771bd28580ed"),

ObjectId("62b2131f58e2771bd28580ee"),

ObjectId("62b2131f58e2771bd28580ef"),

ObjectId("62b2131f58e2771bd28580f0"),

ObjectId("62b2131f58e2771bd28580f1"),

ObjectId("62b2131f58e2771bd28580f2")

]

}

### Insert the following documents into a comments collection

> db.comments.insertMany([

... {

... username: "GoodGuyGreg",

... comment: "Hope you got a good deal!",

... post: ObjectId("62b2131f58e2771bd28580f0")

... },

... {

... username: "GoodGuyGreg",

... comment: "Don't violate the licensing agreement!",

... post: ObjectId("62b2131f58e2771bd28580f1")

... },

... {

... username: "GoodGuyGreg",

... comment: "Don't violate the licensing agreement!",

... post: ObjectId("62b2131f58e2771bd28580f2")

... },

... {

... username: "ScumbagSteve",

... comment: "It still isn't clean",

... post: ObjectId("62b2131f58e2771bd28580ed")

... },

... {

... username: "ScumbagSteve",

... comment: "Denied your PR cause I found a hack",

... post: ObjectId("62b2131f58e2771bd28580ef")

... }

... ]);

{

"acknowledged" : true,

"insertedIds" : [

ObjectId("62b2164158e2771bd28580f3"),

ObjectId("62b2164158e2771bd28580f4"),

ObjectId("62b2164158e2771bd28580f5"),

ObjectId("62b2164158e2771bd28580f6"),

ObjectId("62b2164158e2771bd28580f7")

]

}

## Querying related collections

> db.users.find().pretty();

{

"\_id" : ObjectId("62b2127458e2771bd28580eb"),

"username" : "GoodGuyGreg",

"first\_name" : "Good Guy",

"last\_name" : "Greg"

}

{

"\_id" : ObjectId("62b2127458e2771bd28580ec"),

"username" : "ScumbagSteve",

"full\_name" : {

"first" : "Scumbag",

"last" : "Steve"

}

}

> db.posts.find().pretty();

{

"\_id" : ObjectId("62b2131f58e2771bd28580ed"),

"username" : "GoodGuyGreg",

"title" : " Passes out at party",

"body" : " Wakes up early and cleans house"

}

{

"\_id" : ObjectId("62b2131f58e2771bd28580ee"),

"username" : "GoodGuyGreg",

"title" : "Steals your identity",

"body" : "Raises your credit score"

}

{

"\_id" : ObjectId("62b2131f58e2771bd28580ef"),

"username" : "GoodGuyGreg",

"title" : " Reports a bug in your code",

"body" : " Sends you a Pull Request"

}

{

"\_id" : ObjectId("62b2131f58e2771bd28580f0"),

"username" : "ScumbagSteve",

"title" : " Borrows something",

"body" : "Sells it"

}

{

"\_id" : ObjectId("62b2131f58e2771bd28580f1"),

"username" : "ScumbagSteve",

"title" : "Borrows everything",

"body" : "The end"

}

{

"\_id" : ObjectId("62b2131f58e2771bd28580f2"),

"username" : "ScumbagSteve",

"title" : "Forks your repo on github",

"body" : "Sets to private"

}

> db.posts.find({ username: 'GoodGuyGreg' }).pretty();

{

"\_id" : ObjectId("62b2131f58e2771bd28580ed"),

"username" : "GoodGuyGreg",

"title" : " Passes out at party",

"body" : " Wakes up early and cleans house"

}

{

"\_id" : ObjectId("62b2131f58e2771bd28580ee"),

"username" : "GoodGuyGreg",

"title" : "Steals your identity",

"body" : "Raises your credit score"

}

{

"\_id" : ObjectId("62b2131f58e2771bd28580ef"),

"username" : "GoodGuyGreg",

"title" : " Reports a bug in your code",

"body" : " Sends you a Pull Request"

}

> db.posts.find({ username: 'ScumbagSteve' }).pretty();

{

"\_id" : ObjectId("62b2131f58e2771bd28580f0"),

"username" : "ScumbagSteve",

"title" : " Borrows something",

"body" : "Sells it"

}

{

"\_id" : ObjectId("62b2131f58e2771bd28580f1"),

"username" : "ScumbagSteve",

"title" : "Borrows everything",

"body" : "The end"

}

{

"\_id" : ObjectId("62b2131f58e2771bd28580f2"),

"username" : "ScumbagSteve",

"title" : "Forks your repo on github",

"body" : "Sets to private"

}

> db.comments.find().pretty();

{

"\_id" : ObjectId("62b2164158e2771bd28580f3"),

"username" : "GoodGuyGreg",

"comment" : "Hope you got a good deal!",

"post" : ObjectId("62b2131f58e2771bd28580f0")

}

{

"\_id" : ObjectId("62b2164158e2771bd28580f4"),

"username" : "GoodGuyGreg",

"comment" : "Don't violate the licensing agreement!",

"post" : ObjectId("62b2131f58e2771bd28580f1")

}

{

"\_id" : ObjectId("62b2164158e2771bd28580f5"),

"username" : "GoodGuyGreg",

"comment" : "Don't violate the licensing agreement!",

"post" : ObjectId("62b2131f58e2771bd28580f2")

}

{

"\_id" : ObjectId("62b2164158e2771bd28580f6"),

"username" : "ScumbagSteve",

"comment" : "It still isn't clean",

"post" : ObjectId("62b2131f58e2771bd28580ed")

}

{

"\_id" : ObjectId("62b2164158e2771bd28580f7"),

"username" : "ScumbagSteve",

"comment" : "Denied your PR cause I found a hack",

"post" : ObjectId("62b2131f58e2771bd28580ef")

}

> db.comments.find({ username: 'GoodGuyGreg' }).pretty();

{

"\_id" : ObjectId("62b2164158e2771bd28580f3"),

"username" : "GoodGuyGreg",

"comment" : "Hope you got a good deal!",

"post" : ObjectId("62b2131f58e2771bd28580f0")

}

{

"\_id" : ObjectId("62b2164158e2771bd28580f4"),

"username" : "GoodGuyGreg",

"comment" : "Don't violate the licensing agreement!",

"post" : ObjectId("62b2131f58e2771bd28580f1")

}

{

"\_id" : ObjectId("62b2164158e2771bd28580f5"),

"username" : "GoodGuyGreg",

"comment" : "Don't violate the licensing agreement!",

"post" : ObjectId("62b2131f58e2771bd28580f2")

}

> db.comments.find({ username: 'ScumbagSteve' }).pretty();

{

"\_id" : ObjectId("62b2164158e2771bd28580f6"),

"username" : "ScumbagSteve",

"comment" : "It still isn't clean",

"post" : ObjectId("62b2131f58e2771bd28580ed")

}

{

"\_id" : ObjectId("62b2164158e2771bd28580f7"),

"username" : "ScumbagSteve",

"comment" : "Denied your PR cause I found a hack",

"post" : ObjectId("62b2131f58e2771bd28580ef")

}

CURD:

# insertOne.mongodb

// Inserting a document without an \_id field

db.books.insertOne({

title: "MongoDB insertOne",

isbn: "0-7617-5657-3",

});

// Inserting a document with an \_id field

db.books.insertOne({

\_id: 1,

title: "Mastering Big Data",

isbn: "0-9270-4986-4",

});

// Inserting another document whose \_id field already exists

db.books.insertOne({

\_id: 1,

title: "MongoDB for JS Developers",

isbn: "0-4925-3790-9",

});

### [insertMany.mongodb](https://github.com/akash-coded/mern/blob/main/database/mongodb/src/crud-operations/insert/insertMany.mongodb)

// Inseting multiple documents without the \_id fields

db.books.insertMany([

{ title: "NoSQL Distilled", isbn: "0-4696-7030-4" },

{ title: "NoSQL in 7 Days", isbn: "0-4086-6859-8" },

{ title: "NoSQL Database", isbn: "0-2504-6932-4" },

]);

// Inseting multiple documents with the \_id fields

db.books.insertMany([

{ \_id: 3, title: "SQL Basics", isbn: "0-7925-6962-8" },

{ \_id: 4, title: "SQL Advanced", isbn: "0-1184-7778-1" },

]);

// Attempting to insert documents whose \_id values already exist

db.books.insertMany([

{ \_id: 3, title: "SQL Performance Tuning", isbn: "0-6799-2974-6" },

{ \_id: 4, title: "SQL Index", isbn: "0-5097-1723-3" },

]);

// Performing an unordered insert

db.books.insertMany(

[

{ \_id: 5, title: "SQL Performance Tuning", isbn: "0-6799-2974-6" },

{ \_id: 5, title: "SQL Trees", isbn: "0-6998-1556-8" },

{ \_id: 6, title: "SQL Graph", isbn: "0-6426-4996-0" },

{ \_id: 7, title: "NoSQL Pros", isbn: "0-9602-9886-X" },

],

{ ordered: false }

);

db.posts.insertMany([

{

\_id: 1,

title: "Bagels are not croissants.",

comments: [

{ comment: "0. true", description: "Test" },

{ comment: "1. croissants aren't bagels.", description: "Speculative" },

],

},

{

\_id: 2,

title: "Coffee please.",

comments: [

{ comment: "2. iced coffee", description: "Cold" },

{ comment: "3. cappuccino", description: "Hot" },

{ comment: "4. whatever", description: "Cold" },

],

},

]);

### [updateOne.mongodb](https://github.com/akash-coded/mern/blob/main/database/mongodb/src/crud-operations/update/updateOne.mongodb)

// Inserting some records to demonstrate update operations

db.products.insertMany([

{

\_id: 1,

name: "xPhone",

price: 799,

releaseDate: ISODate("2011-05-14"),

spec: { ram: 4, screen: 6.5, cpu: 2.66 },

color: ["white", "black"],

storage: [64, 128, 256],

},

{

\_id: 2,

name: "xTablet",

price: 899,

releaseDate: ISODate("2011-09-01"),

spec: { ram: 16, screen: 9.5, cpu: 3.66 },

color: ["white", "black", "purple"],

storage: [128, 256, 512],

},

{

\_id: 3,

name: "SmartTablet",

price: 899,

releaseDate: ISODate("2015-01-14"),

spec: { ram: 12, screen: 9.7, cpu: 3.66 },

color: ["blue"],

storage: [16, 64, 128],

},

{

\_id: 4,

name: "SmartPad",

price: 699,

releaseDate: ISODate("2020-05-14"),

spec: { ram: 8, screen: 9.7, cpu: 1.66 },

color: ["white", "orange", "gold", "gray"],

storage: [128, 256, 1024],

},

{

\_id: 5,

name: "SmartPhone",

price: 599,

releaseDate: ISODate("2022-09-14"),

spec: { ram: 4, screen: 5.7, cpu: 1.66 },

color: ["white", "orange", "gold", "gray"],

storage: [128, 256],

},

]);

// Update the field of a document with a given \_id

db.products.updateOne(

{

\_id: 1,

},

{

$set: {

price: 899,

},

}

);

// UPDATE products SET PRICE = 899 WHERE \_id = 1;

**deleteOne.mongodb**

// Inserting mutilple documents for delete demonstration

db.products.insertMany([

{

\_id: 1,

name: "xPhone",

price: 799,

releaseDate: ISODate("2011-05-14"),

spec: { ram: 4, screen: 6.5, cpu: 2.66 },

color: ["white", "black"],

storage: [64, 128, 256],

},

{

\_id: 2,

name: "xTablet",

price: 899,

releaseDate: ISODate("2011-09-01"),

spec: { ram: 16, screen: 9.5, cpu: 3.66 },

color: ["white", "black", "purple"],

storage: [128, 256, 512],

},

{

\_id: 3,

name: "SmartTablet",

price: 899,

releaseDate: ISODate("2015-01-14"),

spec: { ram: 12, screen: 9.7, cpu: 3.66 },

color: ["blue"],

storage: [16, 64, 128],

},

{

\_id: 4,

name: "SmartPad",

price: 699,

releaseDate: ISODate("2020-05-14"),

spec: { ram: 8, screen: 9.7, cpu: 1.66 },

color: ["white", "orange", "gold", "gray"],

storage: [128, 256, 1024],

},

{

\_id: 5,

name: "SmartPhone",

price: 599,

releaseDate: ISODate("2022-09-14"),

spec: { ram: 4, screen: 9.7, cpu: 1.66 },

color: ["white", "orange", "gold", "gray"],

storage: [128, 256],

},

]);

// Deleting a document with a given \_id

db.products.deleteOne({ \_id: 1 });

// Deleting the first document in a collection

db.products.deleteOne({});

# deleteMany.mongodb

// Inserting some documents for demonstrating deleteMany

db.products.insertMany([

{

\_id: 6,

name: "xPhone",

price: 799,

releaseDate: ISODate("2011-05-14"),

spec: { ram: 4, screen: 6.5, cpu: 2.66 },

color: ["white", "black"],

storage: [64, 128, 256],

},

{

\_id: 7,

name: "xTablet",

price: 899,

releaseDate: ISODate("2011-09-01"),

spec: { ram: 16, screen: 9.5, cpu: 3.66 },

color: ["white", "black", "purple"],

storage: [128, 256, 512],

},

{

\_id: 8,

name: "SmartTablet",

price: 899,

releaseDate: ISODate("2015-01-14"),

spec: { ram: 12, screen: 9.7, cpu: 3.66 },

color: ["blue"],

storage: [16, 64, 128],

},

{

\_id: 9,

name: "SmartPad",

price: 699,

releaseDate: ISODate("2020-05-14"),

spec: { ram: 8, screen: 9.7, cpu: 1.66 },

color: ["white", "orange", "gold", "gray"],

storage: [128, 256, 1024],

},

{

\_id: 10,

name: "SmartPhone",

price: 599,

releaseDate: ISODate("2022-09-14"),

spec: { ram: 4, screen: 9.7, cpu: 1.66 },

color: ["white", "orange", "gold", "gray"],

storage: [128, 256],

},

]);

// Delete all documents which match a filter

db.products.deleteMany({ price: 899 });

### [limit.mongodb](https://github.com/akash-coded/mern/blob/main/database/mongodb/src/crud-operations/find/limit.mongodb)

db.products.insertMany([

{

\_id: 1,

name: "xPhone",

price: 799,

releaseDate: ISODate("2011-05-14T00:00:00Z"),

spec: { ram: 4, screen: 6.5, cpu: 2.66 },

color: ["white", "black"],

storage: [64, 128, 256],

},

{

\_id: 2,

name: "xTablet",

price: 899,

releaseDate: ISODate("2011-09-01T00:00:00Z"),

spec: { ram: 16, screen: 9.5, cpu: 3.66 },

color: ["white", "black", "purple"],

storage: [128, 256, 512],

},

{

\_id: 3,

name: "SmartTablet",

price: 899,

releaseDate: ISODate("2015-01-14T00:00:00Z"),

spec: { ram: 12, screen: 9.7, cpu: 3.66 },

color: ["blue"],

storage: [16, 64, 128],

},

{

\_id: 4,

name: "SmartPad",

price: 699,

releaseDate: ISODate("2020-05-14T00:00:00Z"),

spec: { ram: 8, screen: 9.7, cpu: 1.66 },

color: ["white", "orange", "gold", "gray"],

storage: [128, 256, 1024],

},

{

\_id: 5,

name: "SmartPhone",

price: 599,

releaseDate: ISODate("2022-09-14T00:00:00Z"),

spec: { ram: 4, screen: 9.7, cpu: 1.66 },

color: ["white", "orange", "gold", "gray"],

storage: [128, 256],

},

{

\_id: 6,

name: "xWidget",

spec: { ram: 64, screen: 9.7, cpu: 3.66 },

color: ["black"],

storage: [1024],

},

{

\_id: 7,

name: "xReader",

price: null,

spec: { ram: 64, screen: 6.7, cpu: 3.66 },

color: ["black", "white"],

storage: [128],

},

]);

db.products

.find(

{},

{

name: 1,

price: 1,

}

)

.sort({

price: -1,

})

.limit(1);

db.products

.find(

{},

{

name: 1,

price: 1,

}

)

.sort({

price: -1,

name: 1,

})

.skip(2)

.limit(2);

// SELECT name, price FROM products ORDER BY price DESC, name ASC OFFSET 2 LIMIT 2;

### [logical-or.mongodb](https://github.com/akash-coded/mern/blob/main/database/mongodb/src/crud-operations/find/logical-or.mongodb)

db.products

.find({

$or: [

{

price: 599,

},

{

price: 699,

},

],

})

.pretty();

// SELECT \* FROM products WHERE price = 599 OR price = 699 \G

db.products

.find(

{

$or: [

{

price: 599,

},

{

price: 699,

},

],

},

{

name: 1,

price: 1,

}

)

.pretty();

// SELECT \_id, name, price FROM products WHERE price = 599 OR price = 699 \G

db.products

.find({

price: {

$in: [599, 699],

},

})

.pretty();

// SELECT \* FROM products WHERE price IN (599, 699) \G

**sort.mongodb**

use crudOperationsExampleDb

db.products.insertMany([

{

\_id: 1,

name: "xPhone",

price: 799,

releaseDate: ISODate("2011-05-14T00:00:00Z"),

spec: { ram: 4, screen: 6.5, cpu: 2.66 },

color: ["white", "black"],

storage: [64, 128, 256],

},

{

\_id: 2,

name: "xTablet",

price: 899,

releaseDate: ISODate("2011-09-01T00:00:00Z"),

spec: { ram: 16, screen: 9.5, cpu: 3.66 },

color: ["white", "black", "purple"],

storage: [128, 256, 512],

},

{

\_id: 3,

name: "SmartTablet",

price: 899,

releaseDate: ISODate("2015-01-14T00:00:00Z"),

spec: { ram: 12, screen: 9.7, cpu: 3.66 },

color: ["blue"],

storage: [16, 64, 128],

},

{

\_id: 4,

name: "SmartPad",

price: 699,

releaseDate: ISODate("2020-05-14T00:00:00Z"),

spec: { ram: 8, screen: 9.7, cpu: 1.66 },

color: ["white", "orange", "gold", "gray"],

storage: [128, 256, 1024],

},

{

\_id: 5,

name: "SmartPhone",

price: 599,

releaseDate: ISODate("2022-09-14T00:00:00Z"),

spec: { ram: 4, screen: 9.7, cpu: 1.66 },

color: ["white", "orange", "gold", "gray"],

storage: [128, 256],

},

{

\_id: 6,

name: "xWidget",

spec: { ram: 64, screen: 9.7, cpu: 3.66 },

color: ["black"],

storage: [1024],

},

{

\_id: 7,

name: "xReader",

price: null,

spec: { ram: 64, screen: 6.7, cpu: 3.66 },

color: ["black", "white"],

storage: [128],

},

]);

db.products.find(

{

price: {

$exists: 1,

},

},

{

name: 1,

price: 1,

}

);

db.products

.find(

{

price: {

$exists: 1,

},

},

{

name: 1,

price: 1,

}

)

.sort({

price: 1,

});

db.products

.find(

{

price: {

$exists: 1,

},

},

{

name: 1,

price: 1,

}

)

.sort({

price: -1,

});

db.products

.find(

{

price: {

$exists: 1,

},

},

{

name: 1,

price: 1,

}

)

.sort({

name: 1,

price: 1,

});

db.products

.find(

{

price: {

$exists: 1,

},

},

{

name: 1,

price: 1,

}

)

.sort({

name: 1,

price: -1,

});

db.products

.find(

{},

{

name: 1,

spec: 1,

}

)

.sort({

"spec.ram": 1,

});