**MongoDB Cheat Sheet**

**View all databases**

=> show dbs

**Create a new or switch databases**

=> use dbName

**View current database**

=>db

**Delete Database**

=>db.dropDatabase();

**View all collection**

=> show collections;

**Create collection**

=> db.createCollection(‘comments’);

**Delete a collection**

=> db.collectionName.drop();

**Commands for rows:**

**Insert one row**

=>

db.comments.insert({

'name': 'Harry',

'lang': 'JavaScript',

'member\_since': 5

})

### **Insert many Rows:**

=>

db.comments.insertMany([{

'name': 'Harry',

'lang': 'JavaScript',

'member\_since': 5

},

{'name': 'Rohan',

'lang': 'Python',

'member\_since': 3

},

{'name': 'Lovish',

'lang': 'Java',

'member\_since': 4

}])

**Show all rows**

=> db.comments.find();

### **Limit the number of rows in output**

=> db.comments.find().limit(2)

### **Count the number of rows in the output**

=> db.comments.find().count()

**Search in mongoDb:**

=> db.comments.find({ ‘name’ : “Neha”});

=> db.persons.findOne({ index : 0});

=> db.persons.find({ eyeColor : "green"}).size();

=> db.persons.find({ eyeColor : { $ne : "green"} }).size();

=> db.persons.find({ "company.location.country" : "USA"}).size();

**Update a row**

=>

db.comments.updateOne({name: 'Shubham'},

{$set: {'name': 'Harry',

'lang': 'JavaScript',

'member\_since': 51

}}, {upsert: true})

**Delete a row**

=> db.comments.remove({name: 'Harry'})

=> db.persons.deleteOne({ index : 0});

**Delete many**

=> db.persons.deleteMany({index : 0});

**Counts the documents**

=> db.persons.countDocuments()

**Operators in MongoDB:**

1. **Query and Projection Operators**

Query Selectors

Comparison

Logical

Element

1. **Update Operators**

Set

Fields

**========================================================================**

**Neha - Comparison and Logical**

**find() =>** To query documents from collections

**db.collection.find(query, projection);**

Query => selection criteria

Projection => specifies which fields to include and exclude

**Query Selectors:**

| { <field>: { $eq: <value> } } |
| --- |

**Comparison Operators: (Works with numbers, strings and Date)**

* **Order matters**

1. **$eq => equal to**

**isActive = false**

db.persons.find({"isActive" : {$eq : false}}).size();

Embedded document =>

db.persons.find({"company.title" : {$eq : "YURTURE"}});

Array element equals a value=>

db.persons.find({tags : { $eq : "ad"}}).size();

Equals an array value =>

db.persons.find({tags : { $eq : ["enim",

"id",

"velit",

"ad",

"consequat"]}}).size();

1. **$gt => greater than**

Age > 20

db.persons.find({ age : {$gt : 20}}).size();

1. **$gte => greater than equal to**

db.persons.find({ age : {$gte : 20}}).size();

1. **$in => multiple or conditions (it needs an array)**

db.persons.find({favoriteFruit : {$in : ["banana", "strawberry"]}}).size();

Update and in =>

db.persons.updateOne({favoriteFruit : {$in : ["apple"]}}, {$set : {isActive : true}});

1. **$lt => less than**
2. **$lte => less than equal to**
3. **$ne => not equal to**
4. **$nin => not in**

db.persons.find({favoriteFruit : {$nin : ["banana", "strawberry"]}}).size();

**Logical Operators:**

1. **$and => conditions written inside [ ]**

**Gender is female and country is USA**

db.persons.find({$and : [ {gender : { $eq : "female"}} , { "company.location.country" : {$eq : "USA"} }] } ).size();

**Implicit AND => having the same field will result in the first condition overridden by the second condition.**

db.persons.find({age : {$ne : 25}, age : {$gte : 26}}).size();

**It will override**

db.persons.find({age : {$gte : 26}, age : {$ne : 25}}).size();

1. **$or => conditions written inside [ ]**

**Eye Color is green OR gender is female**

db.persons.find({$or : [ {"eyeColor" : "green"}, {"gender" : "female"} ]}).size();

**If want to use or operator on field then use in**

1. **$not => requires regex or document**

**Find all users who are either 20 years old or have the word "velit" in their tags, and their favorite fruit is not an apple:**

db.persons.find({

"$or": [

{ "age": 20 },

{ "tags": "velit" }

],

"favoriteFruit": { "$ne": "apple" }

})

**========================================================================**

**Shreyas - Array, Evaluation**

### **Query Operators:**

### **Array**

| **Name** | **Description** |
| --- | --- |
| [$all](https://www.mongodb.com/docs/manual/reference/operator/query/all/#mongodb-query-op.-all) | Matches arrays that contain all elements specified in the query.  **db.inventory.find({tags : {$all : ["books", "bags"]}});** |
| [$elemMatch](https://www.mongodb.com/docs/manual/reference/operator/query/elemMatch/#mongodb-query-op.-elemMatch) | Selects documents if the element in the array field matches all the specified $elemMatch conditions.  **db.inventory.find({qty : {$elemMatch : {"num" : {$lte : 10}}}});** |
| [$size](https://www.mongodb.com/docs/manual/reference/operator/query/size/#mongodb-query-op.-size) | Selects documents if the array field is a specified size. |

**Syntax:**

* **$all -** db.<collectionname>.find({ <Array Name> : { $all : [<value1>, <value2> ]}})
* **$size -** db.<collectionname>.find({ <Array Name> : { $size : <value>}})
* **$elemMatch -** db.<collectionname>.find(<Array Name> : {$elemMatch : {<condition1>, <condition2>}})

### **Evaluation**

| **Name** | **Description** |
| --- | --- |
| [$expr](https://www.mongodb.com/docs/manual/reference/operator/query/expr/#mongodb-query-op.-expr) | Allows use of aggregation expressions within the query language. |
| [$jsonSchema](https://www.mongodb.com/docs/manual/reference/operator/query/jsonSchema/#mongodb-query-op.-jsonSchema) | Validate documents against the given JSON Schema. |
| [$mod](https://www.mongodb.com/docs/manual/reference/operator/query/mod/#mongodb-query-op.-mod) | Performs a modulo operation on the value of a field and selects documents with a specified result. |
| [$regex](https://www.mongodb.com/docs/manual/reference/operator/query/regex/#mongodb-query-op.-regex) | Selects documents where values match a specified regular expression. |
| [$text](https://www.mongodb.com/docs/manual/reference/operator/query/text/#mongodb-query-op.-text) | Performs text search. |
| [$where](https://www.mongodb.com/docs/manual/reference/operator/query/where/#mongodb-query-op.-where) | Matches documents that satisfy a JavaScript expression. |

**Syntax:**

* **$expr -** db.collection\_name.find( { $expr: { <expression> } } )
* **$jsonSchema -** db.collection\_name.find( { $jsonSchema: <schema\_name> } )
* **$mod -** db.collection\_name.find( { <Property Name> : { $mod : [<value1>, <value2>] } } )
* **$regex -** db.collection\_name.find( { <Property Name> : { $regex : /pattern/ } } )
* **$where -** db.collection\_name.find( { $where: <string|JavaScript Code> } )
* **$text -** db.collection\_name.find( { $text: { $search: <string> } } )

**Update Operators**

**db.collectionName.update({filter}, {update}, {optional});**

***Syntax:***

***db.collection.updateMany(***

***<filter>,***

***<update>,***

***{***

***upsert: <boolean>,***

***writeConcern: <document>,***

***collation: <document>,***

***arrayFilters: [ <filterdocument1>, <filterdocument2>... ],***

***hint: <document|string>***

***}***

***)***

## **Syntax**

Specify the operator expression in a document of the form:

| { |
| --- |
| <operator1>: { <field1>: <value1>, ... }, |
| <operator2>: { <field2>: <value2>, ... }, |
| ... |
| } |

### **Array Operators**

| **Name** | **Description** |
| --- | --- |
| [$](https://www.mongodb.com/docs/manual/reference/operator/update/positional/#mongodb-update-up.-) | Acts as a placeholder to update the first element that matches the query condition.  **db.bloggers.updateOne(**  **{username : "user5", "posts.post\_id" : 5},**  **{$set : {"posts.$.content" : "Doesn't mind, doesn't bother"}}**  **);** |
| [$[]](https://www.mongodb.com/docs/manual/reference/operator/update/positional-all/#mongodb-update-up.---) | Acts as a placeholder to update all elements in an array for the documents that match the query condition.  **db.bloggers.updateMany(**  **{username : {$in : ["user1", "user4"] }},**  **{$addToSet : {"posts.$[].likes" : { $each : ["user2" , "user5"] }}}**  **);** |
| [$[<identifier>]](https://www.mongodb.com/docs/manual/reference/operator/update/positional-filtered/#mongodb-update-up.---identifier--) | Acts as a placeholder to update all elements that match the arrayFilters condition for the documents that match the query condition.  **db.bloggers.updateOne(**  **{username : "user1", "posts.post\_id" : 1},**  **{$set : {"posts.$.comments.$[tippani].comment" : "You are not amazing !!!"}},**  **{arrayFilters : [{"tippani.user" : "user2"}]}**  **);** |
| [$addToSet](https://www.mongodb.com/docs/manual/reference/operator/update/addToSet/#mongodb-update-up.-addToSet) | Adds elements to an array only if they do not already exist in the set.  **db.inventory.updateMany({}, {$addToSet : {tags : {$each : ["school", "bag" ]}}});** |
| [$pop](https://www.mongodb.com/docs/manual/reference/operator/update/pop/#mongodb-update-up.-pop) | Removes the first or last item of an array.  1 => last, -1 => first  **db.survey.updateMany({\_id : 1}, {$pop : {results : -1}});** |
| [$pull](https://www.mongodb.com/docs/manual/reference/operator/update/pull/#mongodb-update-up.-pull) | Removes all array elements that match a specified query.  (Removes all instances of the value from the existing array)  **db.inventory.updateMany({}, {$pull : {tags : "electronics"}});** |
| [$push](https://www.mongodb.com/docs/manual/reference/operator/update/push/#mongodb-update-up.-push) | Adds an item to an array.  **db.inventory.updateMany({}, {$push : {tags : {$each : ["school", "books"]}}});** |
| [$pullAll](https://www.mongodb.com/docs/manual/reference/operator/update/pullAll/#mongodb-update-up.-pullAll) | Removes all matching values from an array. (uses list of values to remove) i.e. [ values to remove ]  **db.inventory.updateMany({}, {$pullAll : {tags : ["school", "book"]}});** |

**Aggregation Operators**

### **Array Expression Operators**

| **Name** | **Description** |
| --- | --- |
| [$arrayElemAt](https://www.mongodb.com/docs/manual/reference/operator/aggregation/arrayElemAt/#mongodb-expression-exp.-arrayElemAt) | Returns the element at the specified array index. |
| [$arrayToObject](https://www.mongodb.com/docs/manual/reference/operator/aggregation/arrayToObject/#mongodb-expression-exp.-arrayToObject) | Converts an array of key value pairs to a document. |
| [$concatArrays](https://www.mongodb.com/docs/manual/reference/operator/aggregation/concatArrays/#mongodb-expression-exp.-concatArrays) | Concatenates arrays to return the concatenated array. |
| [$filter](https://www.mongodb.com/docs/manual/reference/operator/aggregation/filter/#mongodb-expression-exp.-filter) | Selects a subset of the array to return an array with only the elements that match the filter condition. |
| [$firstN](https://www.mongodb.com/docs/manual/reference/operator/aggregation/firstN-array-element/#mongodb-expression-exp.-firstN) | Returns a specified number of elements from the beginning of an array. Distinct from the [$firstN](https://www.mongodb.com/docs/manual/reference/operator/aggregation/firstN/#mongodb-group-grp.-firstN) accumulator. |
| [$in](https://www.mongodb.com/docs/manual/reference/operator/aggregation/in/#mongodb-expression-exp.-in) | Returns a boolean indicating whether a specified value is in an array. |
| [$indexOfArray](https://www.mongodb.com/docs/manual/reference/operator/aggregation/indexOfArray/#mongodb-expression-exp.-indexOfArray) | Searches an array for an occurrence of a specified value and returns the array index of the first occurrence. Array indexes start at zero. |
| [$isArray](https://www.mongodb.com/docs/manual/reference/operator/aggregation/isArray/#mongodb-expression-exp.-isArray) | Determines if the operand is an array. Returns a boolean. |
| [$lastN](https://www.mongodb.com/docs/manual/reference/operator/aggregation/lastN-array-element/#mongodb-expression-exp.-lastN) | Returns a specified number of elements from the end of an array. Distinct from the [$lastN](https://www.mongodb.com/docs/manual/reference/operator/aggregation/lastN/#mongodb-group-grp.-lastN) accumulator. |
| [$map](https://www.mongodb.com/docs/manual/reference/operator/aggregation/map/#mongodb-expression-exp.-map) | Applies a subexpression to each element of an array and returns the array of resulting values in order. Accepts named parameters. |
| [$maxN](https://www.mongodb.com/docs/manual/reference/operator/aggregation/maxN-array-element/#mongodb-expression-exp.-maxN) | Returns the n largest values in an array. Distinct from the [$maxN](https://www.mongodb.com/docs/manual/reference/operator/aggregation/maxN/#mongodb-group-grp.-maxN) accumulator. |
| [$minN](https://www.mongodb.com/docs/manual/reference/operator/aggregation/minN-array-element/#mongodb-expression-exp.-minN) | Returns the n smallest values in an array. Distinct from the [$minN](https://www.mongodb.com/docs/manual/reference/operator/aggregation/minN/#mongodb-group-grp.-minN) accumulator. |
| [$objectToArray](https://www.mongodb.com/docs/manual/reference/operator/aggregation/objectToArray/#mongodb-expression-exp.-objectToArray) | Converts a document to an array of documents representing key-value pairs. |
| [$range](https://www.mongodb.com/docs/manual/reference/operator/aggregation/range/#mongodb-expression-exp.-range) | Outputs an array containing a sequence of integers according to user-defined inputs. |
| [$reduce](https://www.mongodb.com/docs/manual/reference/operator/aggregation/reduce/#mongodb-expression-exp.-reduce) | Applies an expression to each element in an array and combines them into a single value. |
| [$reverseArray](https://www.mongodb.com/docs/manual/reference/operator/aggregation/reverseArray/#mongodb-expression-exp.-reverseArray) | Returns an array with the elements in reverse order. |
| [$size](https://www.mongodb.com/docs/manual/reference/operator/aggregation/size/#mongodb-expression-exp.-size) | Returns the number of elements in the array. Accepts a single expression as an argument. |
| [$slice](https://www.mongodb.com/docs/manual/reference/operator/aggregation/slice/#mongodb-expression-exp.-slice) | Returns a subset of an array. |
| [$sortArray](https://www.mongodb.com/docs/manual/reference/operator/aggregation/sortArray/#mongodb-expression-exp.-sortArray) | Sorts the elements of an array. |
| [$zip](https://www.mongodb.com/docs/manual/reference/operator/aggregation/zip/#mongodb-expression-exp.-zip) | Merge two arrays together. |

**========================================================================**

**Dhananjay - Elements, Bitwise, Update**

### **Element Operators**

| **Name** | **Description** |
| --- | --- |
| [$](https://www.mongodb.com/docs/manual/reference/operator/update/positional/#mongodb-update-up.-)exists | It is use to check whether in collection, in documents, there exists specific field or not  \*In MySQL, the EXISTS operator is used in a SELECT statement to test whether a subquery returns any rows.  { field: { $exists:Boolean } } |
| [$](https://www.mongodb.com/docs/manual/reference/operator/update/positional-all/#mongodb-update-up.---)type | It is use to filter out the specific field, having value of specific data type  { field: { $type: <BSON type> } } |

**Update operators**

| **Name** | **Description** |
| --- | --- |
| [$](https://www.mongodb.com/docs/manual/reference/operator/update/positional/#mongodb-update-up.-)rename | Renames a field.  Syntax: {$rename: { <field1>: <newName1>, <field2>: <newName2>, ... } }  **db.products.updateMany({}, {$rename : {price : "amount"}});** |
| [$](https://www.mongodb.com/docs/manual/reference/operator/update/positional-all/#mongodb-update-up.---)set | Sets the value of a field in a document  Syntax: { $set: { <field1>: <value1>, ... } }  **db.products.updateOne({name : "a"}, {$set : {quantity : 200}});** |
| $unset | Removes the specified field from a document.  The [$unset](https://www.mongodb.com/docs/manual/reference/operator/update/unset/#mongodb-update-up.-unset) operator deletes a particular field.  { $unset: { <field1>: "", ... } }  **db.products.updateOne({name : "d"}, {$unset : {quantity : "", amount : ""}});** |
| $mul | Multiplies the value of the field by the specified amount.  { $mul: { <field1>: <number1>, ... } }  **db.newDb.updateOne({index : 0}, {$mul : {quantity : 2}});** |
| $inc | Increments the value of the field by the specified amount  { $inc: { <field1>: <amount1>, <field2>: <amount2>, ... } }  **db.newDb.updateOne({index : 1}, {$inc : {age : 1 }});** |
| $min | **db.newDb.updateOne({index : 1}, {$min : {age : 25}});** |
| $setOnInsert | If the record is not present in collection then it will insert new records otherwise it won’t add new record if already same record is present  **db.products.updateOne({name : "e"}, {$setOnInsert : {quantity : 60}}, {upsert : true});** |
| $currentDate | **db.products.updateMany({}, {$currentDate : {time : {$type : 'date'}}});** |

| **Name** | **Description** |
| --- | --- |
| [$](https://www.mongodb.com/docs/manual/reference/operator/update/positional/#mongodb-update-up.-) (positional update) | **db.bloggers.updateOne(**  **{username : "user5", "posts.post\_id" : 5},**  **{$set : {"posts.$.content" : "Doesn't mind, doesn't bother"}}**  **);** |
| $[] (all positional update) | **db.bloggers.updateMany(**  **{username : {$in : ["user1", "user4"] }},**  **{$addToSet : {"posts.$[].likes" : { $each : ["user2" , "user5"] }}}**  **);** |
| $[<identifier>] | **db.bloggers.updateOne(**  **{username : "user1", "posts.post\_id" : 1},**  **{$set : {"posts.$.comments.$[comment].comment" : "Updated comment by user 2"}},**  **{$arrayFilters : [{"comment.user" : "user2"}]}**  **);** |