WHERE, AND , OR & CRUD:

# **WHERE:** Use the where operator to pass either a string containing a JavaScript expression or a full JavaScript function to the query system. The [where](https://www.mongodb.com/docs/manual/reference/operator/query/where/#mongodb-query-op.-where) provides greater flexibility, but requires that the database processes the JavaScript expression or function for each document in the collection. Reference the document in the JavaScript expression or function using either this or obj .

# ex:

# db.players.insertMany([

# { \_id: 12378, name: "Steve", username: "steveisawesome", first\_login: "2017-01-01" },

# { \_id: 2, name: "Anya", username: "anya", first\_login: "2001-02-02" }

# ])

# **AND**: [and](https://www.mongodb.com/docs/manual/reference/operator/query/and/#mongodb-query-op.-and) performs a logical AND operation on an array of one or more expressions (<expression1>, <expression2>, and so on) and selects the documents that satisfy all the expressions.

# ex:

# db.inventory.find( { $and: [ { price: { $ne: 1.99 } }, { price: { $exists: true } } ] } )

# **OR:** The or operator performs a logical OR operation on an array of one or more <expressions> and selects the documents that satisfy at least one of the <expressions>.

# ex:

# db.inventory.find( { $or: [ { quantity: { $lt: 20 } }, { price: 10 } ] } )

CRUD OPERATIONS:

# 1.insert :

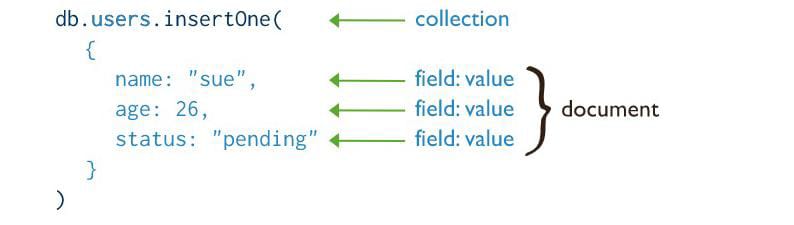
# Create or insert operations add new documents to a [collection](https://www.mongodb.com/docs/manual/core/databases-and-collections/#std-label-collections). If the collection does not currently exist, insert operations will create the collection.

# MongoDB provides the following methods to insert documents into a collection:

# [db.collection.insertOne()](https://www.mongodb.com/docs/manual/reference/method/db.collection.insertOne/#mongodb-method-db.collection.insertOne)

# [db.collection.insertMany()](https://www.mongodb.com/docs/manual/reference/method/db.collection.insertMany/#mongodb-method-db.collection.insertMany)

# In MongoDB, insert operations target a single [collection](https://www.mongodb.com/docs/manual/reference/glossary/#std-term-collection). All write operations in MongoDB are [atomic](https://www.mongodb.com/docs/manual/core/write-operations-atomicity/) on the level of a single [document.](https://www.mongodb.com/docs/manual/core/document/)



# 2.update and update many:

# Update operations modify existing [documents](https://www.mongodb.com/docs/manual/core/document/#std-label-bson-document-format) in a [collection](https://www.mongodb.com/docs/manual/core/databases-and-collections/#std-label-collections). MongoDB provides the following methods to update documents of a collection:

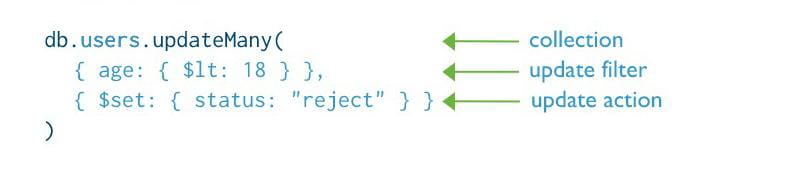
# [db.collection.updateOne()](https://www.mongodb.com/docs/manual/reference/method/db.collection.updateOne/#mongodb-method-db.collection.updateOne)

# [db.collection.updateMany()](https://www.mongodb.com/docs/manual/reference/method/db.collection.updateMany/#mongodb-method-db.collection.updateMany)

# [db.collection.replaceOne()](https://www.mongodb.com/docs/manual/reference/method/db.collection.replaceOne/#mongodb-method-db.collection.replaceOne)

# In MongoDB, update operations target a single collection. All write operations in MongoDB are [atomic](https://www.mongodb.com/docs/manual/core/write-operations-atomicity/) on the level of a single document.

# You can specify criteria, or filters, that identify the documents to update. These [filters](https://www.mongodb.com/docs/manual/core/document/#std-label-document-query-filter) use the same syntax as read operations.



# 3.delete and delete many:

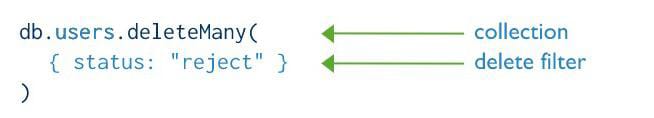
# Delete operations remove documents from a collection. MongoDB provides the following methods to delete documents of a collection:

# [db.collection.deleteOne()](https://www.mongodb.com/docs/manual/reference/method/db.collection.deleteOne/#mongodb-method-db.collection.deleteOne)

# [db.collection.deleteMany()](https://www.mongodb.com/docs/manual/reference/method/db.collection.deleteMany/#mongodb-method-db.collection.deleteMany)

# In MongoDB, delete operations target a single [collection](https://www.mongodb.com/docs/manual/reference/glossary/#std-term-collection). All write operations in MongoDB are [atomic](https://www.mongodb.com/docs/manual/core/write-operations-atomicity/) on the level of a single document.

# You can specify criteria, or filters, that identify the documents to remove. These [filters](https://www.mongodb.com/docs/manual/core/document/#std-label-document-query-filter) use the same syntax as read operations.



PROJECTIONS :

# The positional [$](https://www.mongodb.com/docs/manual/reference/operator/projection/positional/#mongodb-projection-proj.-) operator limits the contents of an <array> to return the first element that matches the query condition on the array.

# Use [$](https://www.mongodb.com/docs/manual/reference/operator/projection/positional/#mongodb-projection-proj.-) in the projection document of the find() method or the [findOne()](https://www.mongodb.com/docs/manual/reference/method/db.collection.findOne/" \l "mongodb-method-db.collection.findOne) method when you only need one particular array element in selected documents.

# See the aggregation operator [$filter](https://www.mongodb.com/docs/manual/reference/operator/aggregation/filter/#mongodb-expression-exp.-filter) to return an array with only those elements that match the specified condition.

# Ex:

|  |
| --- |
| { "\_id" : 1, "semester" : 1, "grades" : [ 70, 87, 90 ] } |
| { "\_id" : 2, "semester" : 1, "grades" : [ 90, 88, 92 ] } |
| { "\_id" : 3, "semester" : 1, "grades" : [ 85, 100, 90 ] } |
| { "\_id" : 4, "semester" : 2, "grades" : [ 79, 85, 80 ] } |
| { "\_id" : 5, "semester" : 2, "grades" : [ 88, 88, 92 ] } |
| { "\_id" : 6, "semester" : 2, "grades" : [ 95, 90, 96 ] } |