**MongoDB Demo –Introduction**

**the following commands**

[**https://www.mongodb.com/docs/manual/introduction/**](https://www.mongodb.com/docs/manual/introduction/)

**use inventory**

**db.inventory.insertOne( { item: "canvas", qty: 100, tags: ["cotton"], size: { h: 28, w: 35.5, uom: "cm" } })**

**db.inventory.insertMany([**

**{ item: "journal", qty: 25, size: { h: 14, w: 21, uom: "cm" }, status: "A" },**

**{ item: "notebook", qty: 50, size: { h: 8.5, w: 11, uom: "in" }, status: "A" },**

**{ item: "paper", qty: 100, size: { h: 8.5, w: 11, uom: "in" }, status: "D" },**

**{ item: "planner", qty: 75, size: { h: 22.85, w: 30, uom: "cm" }, status: "D" },**

**{ item: "postcard", qty: 45, size: { h: 10, w: 15.25, uom: "cm" }, status: "A" }**

**]);**

**db.inventory.find( {} )**

**SELECT \* FROM inventory**

**db.inventory.find( { status: "D" } )**

**SELECT \* FROM inventory WHERE status = "D"**

**db.inventory.find( { status: { $in: [ "A", "D" ] } } )**

**#$in filter predicate operator**

**SELECT \* FROM inventory WHERE status in ("A", "D")**

**db.inventory.find( { status: "A", qty: { $lt: 30 } } )**

**# AND operator given by,**

**# SELECT \* FROM inventory WHERE status = "A" AND qty < 30**

**db.inventory.find( { $or: [ { status: "A" }, { qty: { $lt: 30 } } ] } )**

**#OR operator**

**#SELECT \* FROM inventory WHERE status = "A" OR qty < 30**

**db.inventory.find( {**

**status: "A",**

**$or: [ { qty: { $lt: 30 } }, { item: /^p/ } ]**

**} )**

**# Combo of AND & OR**

**#SELECT \* FROM inventory WHERE status = "A" AND ( qty < 30 OR item LIKE "p%")**

**#Query on Embedded/Nested Documents**

**db.inventory.find( { size: { h: 14, w: 21, uom: "cm" } } )**

**db.inventory.find( { "size.uom": "in" } )**

**# using dot notation to access the right data**

**db.inventory.find( { "size.h": { $lt: 15 } } )**

**#Uses $lt less than operator**

**db.inventory.find( { "size.h": { $lt: 15 }, "size.uom": "in", status: "D" } )**

**#using AND, operator and $lt**

**db.inventory.find( { tags: ["red", "blank"] } )**

**#all those which contain red and blank in tags**

**db.inventory.find( { tags: { $all: ["red", "blank"] } } )**

**#Use of $all operator**

**#to find an array that contains both the elements "red" and "blank", without regard to order or other elements in the array, use the $all operator**

**db.inventory.find( { tags: "red" } )**

**#Atleast one element red**

**db.inventory.insertMany([**

**{ item: "journal", qty: 25, tags: ["blank", "red"], dim\_cm: [ 14, 21 ] },**

**{ item: "notebook", qty: 50, tags: ["red", "blank"], dim\_cm: [ 14, 21 ] },**

**{ item: "paper", qty: 100, tags: ["red", "blank", "plain"], dim\_cm: [ 14, 21 ] },**

**{ item: "planner", qty: 75, tags: ["blank", "red"], dim\_cm: [ 22.85, 30 ] },**

**{ item: "postcard", qty: 45, tags: ["blue"], dim\_cm: [ 10, 15.25 ] }**

**]);**

**#See what happens to the existing data**

**#added as additional data**

**#Data objects are unique because of MongoDB \_ID value**

**db.inventory.find( { dim\_cm: { $gt: 15, $lt: 20 } } )**

**#gt ,lt**

**db.inventory.find( { dim\_cm: { $elemMatch: { $gt: 22, $lt: 30 } } } )**

**#Use $elemMatch operator to specify multiple criteria on the elements of an array such that at least one array element satisfies all the specified criteria.The following example queries for documents where the dim\_cm array contains at least one element that is both greater than ($gt) 22 and less than ($lt) 30:**

**db.inventory.find( { "dim\_cm.1": { $gt: 25 } } )**

**#queries for all documents where the second element in the array dim\_cm is greater than 25**

**db.inventory.find( { "tags": { $size: 3 } } )**

**#Use the $size operator to query for arrays by number of elements.**

**# selects documents where the array tags has 3 elements**

**db.inventory.find( { item: null } )**

**#** Equality **Filter:**

**#The {item: null} query matches documents that either contain the item field whose value is null or that do not contain the item field.**

**Project Fields to Return from Query**

**db.inventory.insertMany( [**

**{ item: "journal", status: "A", size: { h: 14, w: 21, uom: "cm" }, instock: [ { warehouse: "A", qty: 5 } ] },**

**{ item: "notebook", status: "A", size: { h: 8.5, w: 11, uom: "in" }, instock: [ { warehouse: "C", qty: 5 } ] },**

**{ item: "paper", status: "D", size: { h: 8.5, w: 11, uom: "in" }, instock: [ { warehouse: "A", qty: 60 } ] },**

**{ item: "planner", status: "D", size: { h: 22.85, w: 30, uom: "cm" }, instock: [ { warehouse: "A", qty: 40 } ] },**

**{ item: "postcard", status: "A", size: { h: 10, w: 15.25, uom: "cm" }, instock: [ { warehouse: "B", qty: 15 }, { warehouse: "C", qty: 35 } ] }**

**]);**

**db.inventory.find( { status: "A" } )**

**#** **SELECT \* from inventory WHERE status = "A"**

**db.inventory.find( { status: "A" }, { item: 1, status: 1 } )**

**#** **SELECT item, status from inventory WHERE status = "A"**

**Use the data**

**db.inventory.insertMany( [**

**{ item: "canvas", qty: 100, size: { h: 28, w: 35.5, uom: "cm" }, status: "A" },**

**{ item: "journal", qty: 25, size: { h: 14, w: 21, uom: "cm" }, status: "A" },**

**{ item: "mat", qty: 85, size: { h: 27.9, w: 35.5, uom: "cm" }, status: "A" },**

**{ item: "mousepad", qty: 25, size: { h: 19, w: 22.85, uom: "cm" }, status: "P" },**

**{ item: "notebook", qty: 50, size: { h: 8.5, w: 11, uom: "in" }, status: "P" },**

**{ item: "paper", qty: 100, size: { h: 8.5, w: 11, uom: "in" }, status: "D" },**

**{ item: "planner", qty: 75, size: { h: 22.85, w: 30, uom: "cm" }, status: "D" },**

**{ item: "postcard", qty: 45, size: { h: 10, w: 15.25, uom: "cm" }, status: "A" },**

**{ item: "sketchbook", qty: 80, size: { h: 14, w: 21, uom: "cm" }, status: "A" },**

**{ item: "sketch pad", qty: 95, size: { h: 22.85, w: 30.5, uom: "cm" }, status: "A" }**

**] );**

**db.inventory.updateOne(**

**{ item: "paper" },**

**{**

**$set: { "size.uom": "cm", status: "P" },**

**$currentDate: { lastModified: true }**

**}**

**)**

**#** **uses the $set operator to update the value of the size.uom field to "cm" and the value of the status field to "P",**

**#uses the $currentDate operator to update the value of the lastModified field to the current date. If lastModified field does not exist, $currentDate will create the field. See $currentDate for details.**

**db.inventory.updateMany(**

**{ "qty": { $lt: 50 } },**

**{**

**$set: { "size.uom": "in", status: "P" },**

**$currentDate: { lastModified: true }**

**}**

**)**

**db.inventory.replaceOne({ item: "paper" }, { item: "paper", instock: [ { warehouse: "A", qty: 60 }, { warehouse: "B", qty: 40 } ] })**

**#** **When replacing a document, the replacement document must consist of only field/value pairs; i.e. do not include update operators expressions.**

**Delete**

**db.inventory.deleteMany({})**

**#** **deletes all documents from the inventory collection**

**db.inventory.deleteMany({ status : "A" })**

**#** **The method returns a document with the status of the operation**

**#Nothing will be displayed as you deleted all documents in the previous step!**

**db.inventory.deleteOne( { status: "D" } )**

**#** **deletes the first document where status is "D"**

**The sort() method specifies the order in which the query returns the matching documents from the given collection.**

**db.restaurants.insertMany( [**

**{ "\_id" : 1, "name" : "Central Park Cafe", "borough" : "Manhattan"},**

**{ "\_id" : 2, "name" : "Rock A Feller Bar and Grill", "borough" : "Queens"},**

**{ "\_id" : 3, "name" : "Empire State Pub", "borough" : "Brooklyn"},**

**{ "\_id" : 4, "name" : "Stan's Pizzaria", "borough" : "Manhattan"},**

**{ "\_id" : 5, "name" : "Jane's Deli", "borough" : "Brooklyn"},**

**] )**

**Try**

**db.restaurants.aggregate(**

**[**

**{ $sort : { borough : 1 } }**

**]**

**)**

**https://www.mongodb.com/docs/manual/reference/operator/aggregation/sort/**

**#** **Documents are returned in alphabetical order by borough**

**Show collections**

**db.getCollectionNames()**

**# to get the result of all the collections in the database in an array**

**db.CollectionName.drop()**

**try**

**db.restuarants.drop()**

**try show collections**

**to check whether it is dropped**

**db.dropDatabase()**

**Try show dbs**

**Check whether the database is dropped**