MongoDB – Practice Notebook - Summary Project

Part 1:

1) Switch to the 'demo' database

2) Enter these documents into the collection 'animals'

db.animals.save({ \_id: 1, name: 'cat', tags:['land','cute'] ,info:{type:'mammal', color:'red' }})

db.animals.save({ \_id: 2, name: 'rabbit', tags:['land','cute'],info:{type:'mammal', color:'white' } })

db.animals.save({ \_id: 3, name: 'shark', tags:['ocean'] ,info:{type:'fish', color:'grey' }})

db.animals.save({ \_id: 4, name: 'dolphin', tags:['ocean','cute'] ,info:{type:'mammal', color:'grey', canFly: null }})

db.animals.save({ \_id: 5, name: 'penguin', tags:['land','ocean','cute'] ,info:{type:'bird', canFly:false }})

db.animals.save({ \_id: 6, name: 'duck', tags:['land','cute'], info: {type:'bird', canFly: true} })

1. Find all ocean animals

-- db.animals.find ({tags:["ocean"]}).pretty()

1. Find all grey animals. This time, only return their name.

-- db.animals.find ({"info.color":"grey"},{name:1}).pretty()

1. Find all animals that that can fly.

-- db.animals.find ({"info.canFly":true}).pretty()

1. Find all animal that don't have an 'info.canFly' field at all

-- db.animals.find ({"info.canFly":{$exists:false}},{name:1,tags:1}).pretty()

1. Find all animals that have an 'info.canFly' field set to null or that don't have any 'info.canFly' field

-- db.animals.find({"info.canFly":null}).pretty()

1. Find all documents, sorted by their \_id in descending order. Return their \_id field only

-- db.animals.find({}, {\_id:1}).sort({\_id:-1})

1. Repeat the last query, this time return only the second and third documents

-- db.animals.find({}, {\_id:1}).sort({\_id:-1}).skip(1).limit(2)

1. Run the Command : db.animals.find({}).forEach(function(r){ print(r.\_id)})
2. Change the above command display the animal’s name concatenate with its color.

--db.animals.find({}).forEach(function(r){print(r.name +" "+ r.info.color)})

1. Find all the animals where all the tags are exactlt 'ocean' and 'cute'

--db.animals.find({}).forEach(function(r){print(r.name +" "+ r.tags)})

1. Find all the animals that have exactly 3 tags

--db.animals.find({tags:{$size:3}},{tags:1}).pretty()

Part 2: Import aircraft JSON , Practice Operators, Near.

1. Select all the aircraft where the model field is equal to 'Boeing 737-900'

--db.aircraft.find({model:"Boeing 737-900"})

1. Select all the aircraft where the range field is 5600

-- db.aircraft.find({range:5600}).pretty()

1. Select all the aircraft where the under Maintenance field is true

--no Data

1. Select a single document by \_id

--db.aircraft.find({\_id: ObjectId("617ee3c0a0a58bfdb7fca2b6")})

1. Select all the aircraft where the model field is not 'Boeing 737-900'

--db.aircraft.find({model:{$nin:["Boeing 737-900"]}}).pretty()

1. Select all the aircraft where the capacity field is strictly greater than 200

-- db.aircraft.find({capacity:{$gt:200}}).pretty()

1. Select all the aircraft where the nextMaintenance field is greater than (after) 2020-02-20

--no Data

1. Select all the aircraft where the capacity field is greater or equal than 200

--db.aircraft.find({capacity:{$gte:200}}).pretty()

1. Select all the aircraft where the capacity field is strictly less than 200

--db.aircraft.find({capacity:{$lt:200}}).pretty()

1. Select all the aircraft where the nextMaintenance field is less then (before) 2020-02-20

---no Data

1. Select all the aircraft where the capacity field is less than or equal to 200

--db.aircraft.find({capacity:{$lt:200}}).pretty()

1. Select all aircraft where the model field has one of the values: 'Airbus A350', 'Boeing 747'

--db.aircraft.find({model:{$in:["Boeing 747","Airbus A350"]}}).pretty()

1. Select all aircraft where the model field is different than: 'Airbus A350', 'Boeing 747'

--db.aircraft.find({model:{$nin:["Airbus A350","Boeing 747"]}}).pretty()

1. Select all the aircraft where the model field does not matche a regular expression

-- db.aircraft.find({ model: { $nin: [/^B/] } }).pretty()

1. NEAR - Find all the aircraft within 10 km of longitude=26.2, latitude=44.4

--no Data

Part 3: Import Flights JSON , Practice Sub-Documents, Free Test Search

1. Select all flights where departure city is Paris

-- db.flights.find({"departure.city" : "Paris"}).pretty().limit(1)

1. Find all flights where the “destination.runwaylength” is less than 3000

-- db.flights.find({"destination.runwayLength" :{$lt:3000}}).pretty()

1. Select all the flights where the duration is between 2 and 3 hours (short query version because the same field is targeted)

-- db.flights.find({duration: {$lt: 180, $gt: 120}}).pretty()

1. Select all the flights where the depart OR land in Germany

-- db.flights.find({"departure.country":"Germany","destination.country":"Germany" }).pretty()

1. Find all the flights where the duration is greater than 100 AND less than 200

-- db.flights.find({duration: {$lt: 200, $gt: 100}}).pretty()

1. Combine AND and OR: Select all the flights where the 'runwayLength' is less than 3000 AND either the duration is greater than 100 OR the distanceKm is greater than 1730 –
2. Find 'aircraftCode' field and id where the 'aircraftCode' field exists  
   -- db.flights.find({aircraftCode: {$exists: true}}, {aircraftCode: 1}).pretty()
3. Find 'aircraftCode' field and id where the 'aircraftCode' field not exists

-- db.flights.find({aircraftCode: {$exists: false}},{type:1,delayed:1}).pretty()

1. Find 'aircraftCode' field and id where the 'aircraftCode' fiels is null

--db.flights.find({aircraftCode:null},{type:1,delayed:1,aircraftCode:1}).pretty()

1. Find all the flights where the 'aircraftCode' field exists and it's type is 'string'

-- db.flights.find({aircraftCode: {$exists: true, $type: "string"}}, {type:1,delayed:1,aircraftCode: 1}).pretty()

1. Prints the duration, departure.city and destination.city, omit the \_id to the console in a readable JSON format. Sort the display in descending order by the duration field.

-- db.flights.find({},{duration: 1,"departure.city":1,"destination.city": 1, \_id: 0}).sort({duration: -1}).pretty()

### Free Text Search

1. Create text index on departure and destination fields
2. Find all the documents by free text 'Paris Portugal' and also print the text score (relevance)

--db.flights.find({$text: {$search: "Paris Portugal"}}, {"destination.city":1, "departure.city":1,"destination.country":1,score: {$meta: "textScore"}}).pretty()