**Documents**

Documents are how MongoDB stores data.

Create

**Inserting a document** using the insertOne() function on the collection you wish to add to and pass in the JSON object you want to store:

**db**.collectionName.**insertOne** ({

“name”: “Ridwan”,

“age”: 27,

})

**Inserting multiple documents:**

**db**.collectionName.**insertMany** ([

{

“name”: “Ridwan”,

},

{

“name”: “Riyad”,

}

])

Documents in MongoDB are given a primary key; **\_id** field.

Read

Displays all documents from a collection using **find ()** function:

**db**.collectionName.**find ()**

Displaying certain fields from a document using a **projection** object into a **find ()** function:

**db**.collectionName.**find** (

{ },

{

“\_id”: false,

“name”: true,

“age”: true

}

)

The above query finds all documents because of { }, but only fields that show are name, age.

Id is displayed by default unless it’s set to false to exclude it from query results.

GENERAL RULE: in a single projection, its only possible to include fields or exclude fields, but not both. the \_id field is an exception to this rule.

Writing queries to display partial objects:

**db**.collectionName.**find** (

{

“name”: “Ridwan”

}

)

Operators

**Equals : $eq** gives the same output as the example above ^

**Not Equals : $ne**

**db**.collectionName.**find** (

{

“name” : { “$eq”: “Ridwan” }

“name” : { “$ne”: “Ridwan” }

}

)

**Greater/Less than :**

**$gt** (Greater than), **$gte** (Greater than or Equal to), **$lt** (Less than), **$lte** (Less than or Equal to)

**db**.collectionName.**find** (

{

“age” : { “$gt”: “30” },

“age” : { “$gte”: “30” },

“age” : { “$lt”: “30” },

“age” : { “$lte”: “30” }

}

)

**IN/NIN:**

**$in** (in), **$nin** (Not in)

**db**.collectionName.**find (**

{

“specialisation” : {

“$nin”: [

“Software Dev”

]

}

}

**)**

Update

**Updating a document using a updateOne () function:**

**db**.collectionName.**updateOne (**

{

“name”: “Ridwan”,

“surname”: “Kawsar”

},

{

“$set” : {

“specialisation” : “Software Development”

}

}

**)**

^ $ differentiates between the set operator and the field called set.

**Updating multiple documents using a updateMany () function:**

**db**.collectionName.**updateMany (**

{ }

{

“$set” : {

“reportsTo” : “Jim Gordon”

}

}

**)**

**Replacing an existing document with a new document:**

**db**.collectionName.**replaceOne ( )**

**Arrays:**

Putting Values into Arrays; **$push** operator

Removing Values from Arrays; **$pull** operator

**db**.collectionName.**updateOne (**

{

“name”: “Ridwan”,

“surname”: “Kawsar”

},

{

“$push” : {

“subjects” : “MongoDB”

}

“$pull” : {

“subjects” : { “$nin”: “Scala” }

}

}

**)**

Delete

**Deleting a document: deleteOne ( )** Deletes the first document that matches the query

**db**.collectionName.**deleteOne (**

{

“name”: “Ridwan”

},

**)**

**Deleting multiple documents: deleteMany ( )** Deletes all document that matches the query

**db**.collectionName.**deleteMany (**

{

“name”: “Ridwan”

},

**)**

**Embedded Documents** – Putting a document inside of another document.

Used to represent an entity belonging to another entity, e.g., a person’s job can be defined as an entity and can be embedded inside the person entity.

{

“name”: “Ridwan”,

“age”: 27,

“hobbies”: “Football”

“job”: {

“title”: “Tech Specialist”,

“salary”: “£20000”

}

}