Cheatsheet

DATABASE COMMANDS

1.View all databases

show dbs

2.Create a new or switch databases

use dbName

3.View current Database

db

4.Delete Database

db.dropDatabase()

COLLECTION COMMANDS

1.Show Collections

show collections

2.Create a collection named 'comments'

db.createCollection('comments')

3.Drop a collection named 'comments'

db.comments.drop()

ROW COMMANDS

1.Insert One Row

db.comments.insert({

'name': 'Harry',

'lang': 'JavaScript',

'member\_since': 5

})

2.Insert many Rows(array of rows thta is objects)

db.comments.insert([{},{},{}]);

FIND

Syntax:db.collection.find(query, projection)

query: Specifies selection filter using query operators.

projection:Specifies the fields to return in the documents that match the query filter.

1.Show all Rows in a Collection

db.comments.find()

2.Show all Rows in a Collection (Prettified)

db.comments.find().pretty()

3.Find the first row matching the object(If you don't specify one it will return all the rows)

db.comments.findOne({name: 'Harry'})

4.Limit the number of rows in output

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

5.Count the number of rows in the output

db.comments.find().count()

6.here in find field we have specified the filter in the first field and projection in second. What the query will do is from the collection it will filter the document with the specifies filter and then return the attribute that is present in the second field

db.Customer.find({"order.product.product\_id":203},{\_id:2}).pretty();

{ "\_id" : 2 }

7.db.bios.find().skip( 5 )

skip first documents and return remaining documents

8.gt

db.employee.find({salary:{$gt:5000}}).pretty();

db.Customer.find({"order.product.product\_id":203}).pretty();

this returns the whole document

UPDATE

a)this updates only one row.

db.collection.update(query, update, options)

db.coolection.update({condition},{the condition that you have to update})

db.employee.update({name:"RS"},{$inc:{"salary":+8000}});

b)this will update many rows. if you do not specify the condition it will update all the rows.that is if there is attribute that is present in more than one row it will update all those attributes

db.employee.updateMany({},{$inc:{"salary":+4000}});

i)increment

1.db.employee.update({name:"RS"},{$inc:{"salary":+8000}});

basically this query will find the name rs and increment its salary by 8000

ii)set

db.employee.update({role:"MSD"},{$set:{role:["C","WK"]}});

it sets the value of the attribute.

set has a 3rd attribute upsert

(//The term upsert is a portmanteau â€“ a combination of the words â€œupdateâ€ and â€œinsert.â€ In the context of relational databases, an upsert is a database operation that will update an existing row if a specified value already exists in a table, and insert a new row if the specified value doesnâ€™t already exist.)

>db.employee.update({no:11},{$set:{no:11,name:"AK",salary:10000,role:"coach"}},{upsert:true});

iii)unset

v)mul

db.employee.update({name:"RD"},{$mul:{salary:2}});

OPERATORS

In mongoDB when you provide multiple fields within single query, mongoDB treats them as an implicit AND operator

1.Same syntax for $and,$or

db.collection\_name.find({$or:[{},{}]}).pretty();

eg:

db.film.find({$or:[{title:/^s/},{year\_rel:2022}]}).pretty();

you can use nested as well

db.collection\_name.find({$and:[{$or:[{},{}]},{}]}).pretty();

2.$in -> mutliple or

syntax:

db.collection\_name.find({$in:[,,,]}).pretty();

eg:

db.film.find({year\_rel:{$in:[2015,2019]}}).pretty();

3.exits

db.bios.find( {

birth: { $gt: new Date('1920-01-01') },

death: { $exists: false }

} );

now this will return all the documents where the death attribute won't exits

4. implicit AND

db.collectionName.find({{title:/^s/},year\_rel:2022})

FINDING WITH START,END,IN MIDDLE

1. name begins with S

db.employee.find({name:/^S/}).pretty();

2.ends with K

db.employee.find({name:/K$/}).pretty();

3.S in any position

db.employee.find({name:/S/}).pretty();

4.begins with S (using regular expression)

db.employee.find({name:{$regex:"^S"}}).pretty();

5.To Find Document From the empl collection where name

begins with S (using regular expression)

db.employee.find({name:{$regex:"^S",$options:"i"}}).pretty();

SORT

Sort the actors in ascending order according to their age.

db.Actor.aggregate({$sort:{age:1}}).pretty();

1:big to small

-1:small to big

QUERY ON ARRAYS

USe bios collection from mongodb manual

1. db.bios.find( { contribs: "UNIX" } ).pretty();

"contribs" : [

"UNIX",

"C"

]

2.db.bios.find( { contribs: { $size: 4 } } )

here in the document the contribs is the array and it will return all the elements where contribs array size is 4

DIFFERNCE BETWEEN PUSH AND SET

1.db.Customer.update({\_id:2},{$set:{"order.0.product.products\_use":"very useful"}});

Customer collection has a field as order which is a array, inside that array we have to access the 0th element of the array, the 0th element here is product and inside product we have to insert product\_use, if we do using set it will be inserted as a string and if we do the same thing using push it will be inserted as a array

2.db.Customer.update({"order.order\_id":102},{$set:{"order.$.product.products\_use":"very useful"}});

DELETE

1.Delete the film " Kismat Konection".

db.film.remove({"title":"Kisamt konnection"});

2.Delete all actors from Actor who have age greater than 50

db.Actor.update({},{$unset:{age:{$gt:50}}});

USE OF DOLLAR

whenever we want to access the values we use dollar

here we want to access the values and hence we are using $ to access the values

e.g. find all

