Mongo DB

☆ Create database:

→ Use database_name

☆ Create collection.

→ db.CreateCollection('v')

☆ Remove collection

→ db.v.drop()

☆ Remove database.

→ db.dropDatabase

☆ Show all database

→ show dbs

☆ Show all collection

Show collection

CRUd

Date
Page

* Execute one :

document db.N.insertOne (
{
"maker" : "Tata",
"model" : "Nexon",
"engin" :
{"type" : "Turbochanged",
"cc" : 1025
},
"fetures" :
["Touch screen", "ABC", "ABD"]
"airbags" : 2
})

Insert many : {[{},{},{}]}

db.N.insertMany ([
{
"maker" : "Laksh",
"model" : "Nexon",
"engine :
{"type" : "ABC",
"cc" : ["xyz", "DEF", "AAB"]},
"airbags" : 2
},
{
# new document
}
])

find, xcld

Date
Page

find many :-

* db.N.find ({ }, { maker : 1 })

* output :

{ _Id : ABC,
maker : 'Tatu' }

{ _id : xyz,
maker : 'laksh' }

* In above output if we are not want Id then,

* db.N.find ({ }, { maker : 1, _id : 0 })

* output :-

{ maker : 'Tatu' }
{ maker : "Tatu" }
{ maker : "Hyundai" }

* Perticular data find whole details.

* db.ca.find ({ maker : "Tata" })

maker : 'Tata'

* Select whole data finding array value.

* db.N.find ({ arrayname : "values find" })

arrayname : 'values find'

* select on nested document whole data

* db.N.find ({"assdn.main" : loas })

assdn {

main : loas

}

* if we want to find first matching value only,

db.N.findone ({ marks : 'TWTD' })

Same as find just toggle bit change.

Update

Date
Page

```
db.cars.updateOne(
    { model : "Nexon" },
    { $set : { color : "Red" }
);
```

↓

find first nexon model and ~~creat~~ insert new value color if key already exists then just update value.

```
db.cars.updateone (
    { model : "Nexon" },
    { $Push : { fetures : "Heated Seats" } }
);
```

↓

for add/end something in array.

```
db.cars.updutone (
    { model : "Nexon" }
    { $PULL : { fetures : "Heated seats" } }
);
```

↓

for remove element from array
we can use ~~model~~, $PULL,

db. cars. update one (
{ model : "creta" },
{ $set : { "engine. torque" : "270 nm" } }
)

↓

we need to write in double qute in use.
for merge then we can change it.

---

db. cars. update One (
{ model : "nexon" },
{ $Push : { features : { $each : ["wireless
charges", "voice control"] } } }
);

→ use each when we want
add or append multiple
values in
array.

---

db. cars. Updateone (
{ model : "nexon" },
{ $unset :
{ price : anything } }
)

this is use for remove value from
documents.

→ its not defending on value it is just
check Key and remove it.

Update many

Date
Page

```
db.cars.updateMany (
    { },
    { $set : { color : "Blue" } }
)
```

↓

Update all rows.
this code add color column in all documents.

Upsert ←) Update + Insert

, one both

```
db.cars.updateMany (
    { model : "Venue" }
    { $set : { maker : "Hyundai" } },
    { Upsert : True }
)
```

↓

If data match then update them and when data not founed that time creat that docis new document.

---

delete one and delete many

```
db.cars.deleteone ({fuel_Type : "Petrol"})
```

↓

only for need to give selected data ..

Date

Date
Page

* Save live time ~~Dataset~~
and Date.

db.N.insertOne ({"Date" : new Date})

* write TimeStamp :-

db.N.insertone ({"TS" : new Timestamp()})

* Querying Operators

Comparison Operators

{$eq : =}
{$lt : <}
{$gt : >}
{$lte : <=}
{$gte : >=}
{$ne : !=}
{$in : in}
$in : not in

{age : {$lt : 18}}

{"engine.cc" : {$gt : 2506}}

* $in :-

db.car.find ({"engine.cc" : {$in : [1498, 2179]}})

give direct value instend of use or or

-> not in $nin Point value thett not in document

$and
$or
$nor
$not

} logical operator

-) $in only use perticuler on one columns but or we can use multiple column.

-> in logical operator we need write all conditions.

-) when we need to add many things. i mean when we doe use multiple condition in logical operator we muse need to use array.

$and : $or ; $nor :

```
db.car.find ({
    $and : [
        {fuel_type : "Diesel"},
        {"engine.type" : "Turbocharged"},
        {"sunroof" : true}
    ]
})
```

. $and : Returns documents where both queryes match
. $or : Returns documents where either query matches.
. $nor : Returns documents where both query fail to match
. $not : Returns documents where the quer does not match.

$regex :-

Allows the use of regular expressions evaluating field values.

$text :-

Performs a text search

$where :-

Uses a Javascript expression to match documents.

$exists :

ex: {age : {$exists : true}}

↓

Particular data have that field than Print whole documen

$type :-

Here we can filter the content based on BASON types like string, bool etc

→ This can be useful to field with null values.

→ {name : {$type : "string"}}

$size :-

Return all documents that match specified array size.

→ db.N.find ({hobbies : {$size : 4}})

↓

whoever hobbies is 4, Point their all whole documents.

$all :-

return all documents that match the pattern particular value.

→ db.N.find ({hobbies : {$all : ["play", "read"]}})

~~Assay~~:-

Update releted operators

* field

· **$ currentDate:** ~~Date~~

sets the field value to the current date.

* db. Std. updateOne ({ _id : 1},
{ $currentDate : (lastUpdated : true)}
})

output :-

_id : 1
lastupdate : IsoDate ("2025 ....")

~~current Date only~~

* ~~db. Std. updateOne ({ _id : 1},~~
~~{ $current Date : {lastupdate : { $type :~~

current Date only type

* db. Std. updateOne ({ _id : 1},
{ $currentDate :
{ "last update" :
{ $type : "date" }}
} }
})

-) new column currentDate datatype is now date.

* $inc : increment :-

db.N.updateOne(
{_id: 1},
{$inc : {marks :-5}}
)

if document don't there then it's create.

* $pop : Remove the last element of an array

db.N.updateone(
{_id: 1}, → must be array
{$pop {score : 1}
})

1 -> remove right end
-1 -> remove left end.

* $pull :- Remove elements from array

$push :- add elements.

Date
Page

db. N. UpdateOne ({ _id : 1},
{ $pull : { score : { $lt : 80}}}
)

---

* $ Push :- for multiple use array and each.

---

* $ addToSet :
only for use on array.

-> main differce between push and addToset is set is not add already in array.

-> So, No Duplicate value.

---

* $ rename :-
to ~~rename~~ change field name.

=> db. students. UpdateOne ({
{ _id : 1},
{ $rename : { "fullname" : "name" }}
)

↳ can add multiple documents

* in nested :-
info. city ——> info. location
↳ must use info

Cursor methods

Date
Page

* Count() → finding number of document.

db.car.find().count()

db.car.find({fuel_type: "Petrol"}).count()

---

* Sort() :→ sort in assending or desending order

db.car.find({},{model:1}).sort({model:1})

↓
-1 for Dessending

---

* limit(2) :- Show first 2 documents only.

db.car.find().limit(2)

---

* Skip(3) :- not Print first 3

db.car.find().skip(3)