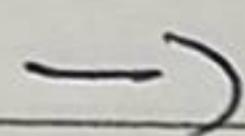


Mongo DB



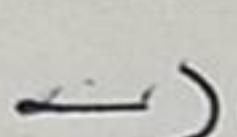
Create database:



→ Use database_name



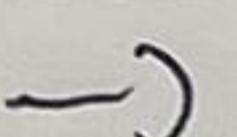
Create collection:



→ db.createCollection('n')



Remove collection



→ db.N.drop()



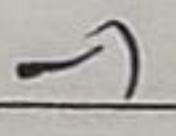
Remove database



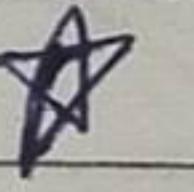
→ db.dropDatabase



Show all database



→ show dbs



Show all collection

show collection

* execute one:

document db. N. insertOne (

{ {

 "make": "Tata",

 "model": "Nexon",

 "engine":

 { "type": "Turbocharged",

 "cc": 1025

 },

 "features":

 ["Touch screen", "ABC", "ABD"]

 "airbags": 2

})

InserMany : { [{ }, { }, { }] }

db. N. insertMany ([

{

 "make": "Laksh",

 "model": "Nexon",

 "engine":

 { "type": "ABC",

 "cc": ["xyz", "DEF", "ABC"] } }

 "airbags": 2

},

{

new document

},

])

find, ~~select~~

Date _____
Page _____

find many :-

* db. N. find ({key: value}) {makes: 1} {key: value}

Output :-

{_id: 'ABC',
maker: 'Tata' } {key: value}

{_id: 'XYZ',
maker: 'Laksh' } {key: value}

* In above output if we do not want
ID then,

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

Output :-

{ maker: 'Tata' } {key: value}

{ maker: 'Tata' } {key: value}

{ maker: 'Hyundai' } {key: value}

* Peculiar duty finds whole details.

* db. ca- find ({key: value}) {key: value}

maker: 'Tata'

* Select whole data finding array value

* db.N.find({c : classynme : "values find" y})

;

classynme : 'values find'

;

;

* select on nested document whole data

* db.N.find({c : "assn. min" : 1025 y})

;

;

;

;

;

;

;

* if we want to find first matching
value only,

db.N.findOne({c : "muker" : 'Tata' y})

Some us find just little bit
change.

Update

Date _____

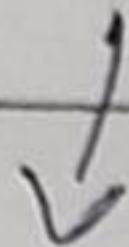
Page _____

```
db.cars.updateOne (
```

```
  { model : "Neon" },
```

```
  { $set : { color : "Red" } }
```

```
);
```



find first neon model and ~~create~~
insert new value color if key already
exists then just update value

```
db.cars.updateOne (
```

```
  { model : "Neon" },
```

```
  { $push : { features : "Heated Seats" } }
```

```
);
```



for ~~append~~ something in array

```
db.cars.updateOne (
```

```
  { model : "Neon" },
```

```
  { $pull : { features : "Heated seats" } }
```

```
);
```

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

db. cars.updateOne({
 model: "Creta"},
 { \$set: { "engine.torque": "270 nm" } }
)

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

db. cars.updateOne({
 model: "nexon",
 \$push: { features: { \$each: ["wireless
 charges", "voice control"] } } }
)

else search when we want to add or update multiple values in array.

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

this is use for remove value from documents.

it's not deleting one value it is just
check key and remove it.

Update many

Date _____
Page _____

db. cars. updateMany (

{
 \$y,

 {\$set : {color : "Blue"}},
 })

↓

Update all rows.

This code add color column in all documents.

Upsert → Update + Insert

db. cars. updateMany (

 { model : "Venue"},
 {\$set : {maker : "Hyundai"}},
 {upsert : true},
 })

↓

If data match then update them
and when data not found then
create that data is now
document.

delete one cmd delete many

db. cars. deleteOne ({fuel_Type : "Petal"})

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() })

* \Rightarrow mapping:

Operations

{ \$eq : '\$' }
& \$lt : '\$ <' }
& \$gt : '\$ >' }
& \$lte : '\$ <=' }
& \$gte : '\$ >=' }
& \$ne : '\$!=' }
& \$in : '\$ in' }

Comparison Operations

{ age : { \$lt : 18 } }

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

* \$in :

db. cars find { "engine.cc": { \$in: [1498, 2179] }}

give direct value instead of use or

→ Not in \$in point value that not in document

\$and
\$or
\$nor
\$not

↳ logical operator

→ \$in only use particular 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 choose multiple
condition in logical operator we
need to use : \$and

\$and : \$or; \$nor;

db.car.find({})

\$and : [

{ "fuel-type": "Diesel" },

{ "engine-type": "Turbocharged" },

{ "sunroof": true }

])

\$and : Returns documents where both queries match.

\$or : Returns documents where either query matches.

\$nor : Returns documents where both query fail to match.

\$not : Returns documents where the query does not match.

Aggregations

Evaluation :-

\$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 }`.

If particular documents have that field than point where document.

\$type :-

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

→ This can be useful to field with null values.

Ex: `name : { $type : "string" }`

\$size :-

Return all documents that match specified array size.

→ db.N. find ({ hobbies : { \$size : 4 } })

→ All the hobbies is 4, point their whole documents.

\$all :-

Return all documents that match the pattern, particular value.

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

Approach:-Update selected operatorsfield\$currentDate:Delete

sets the field value to the current date.

* db.Std.UpdateOne({\$id: 13,lastUpdated: {\$currentDate: {lastUpdated: true}}Output:-- id : 1lastUpdated: ISODate("2025...")~~current Date only~~* db.Std.UpdateOne({\$id: 14,lastUpdated: \$currentDate, \$type: "date"~~current Date only type~~* db.Std.UpdateOne({\$id: 13,lastUpdated: \$currentDate,lastUpdate:lastType: "date"}yyy->

new column currentDate datatype is now date.

* \$inc : increment :-

db.N.updateOne({

{_id: 1},

{\$inc: {marks: -5}}

↓

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

* \$Pop : Remove the last element of an array

db.N.updateOne({

{_id: 1}, → must be array

{\$pop: {score: 1}}

↳

can be -1

-1 → remove right end

-1 → remove left end

* \$pull :- Remove elements from array

\$push :- add elements

db. N. updateOne({ _id : 13 }, { \$pull : { score : { \$gt : 80 } } })

3)

\$push :- for multiple use array and each.

\$addToSet :

(only for use on arrays)

→ main difference between push and addToSet is set is not add already in array.

→ So, No Duplicate value.

\$rename :-

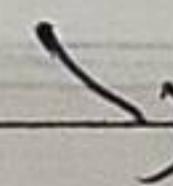
to rename change field name.

→ db. students.updateOne({

{ _id : 13 },

{ \$rename : { "fullname" : "name" } }

4)



can add multiple documents

in nested :-

info.city → info.location



must use info

cursor methods

Date _____
Page _____

* Count() → finding number of document.

db.cursor().find().count()

db.cursor().find({fuel-type : "Petrol"}).count()

* Sort() → sort in ascending or descending order.

db.cursor().find().sort({model : 1})

↓

-1 for

Descending

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

db.cursor().find().limit(2)

* skip(3) :- not print first 3

db.cursor().find().skip(3)