|  |  |
| --- | --- |
| RELATIONAL DATABASE | NO SQL(MONGO DB EQUIVALENT) |
| Database | Database |
| Table | Collection |
| Row | Document |
| Column | Field |

**NOTE:**

1. MongoDB is a NO SQL Database.
2. Collections do not follow any schema.
3. Every document in a collection can contain different fields.
4. It stores data in the form of BSON files.
5. Strings in Mongo queries can be surrounded by either single or double quotes.

---------------------------------------------

Commands for database:

1. View all database:

show dbs

2. Create database or switch to pre-existing database:

use *db\_name*

eg. Create database named Shop:

use Shop

eg. Switch to pre-existing database Admin:

use Admin

3. View name of database you are currently in:

db

4. Drop a database

db.dropDatabase()

eg. Drop database named Shop:

db.dropDatabase()

----------------------------------------------

Commands for Collections:

1. View collections in a database:

show collections

2. Add collection in database:

db.createCollection('*collection\_name*')

eg. Create collection named Product:

db.createCollection('Product')

3. Drop collection:

db.*collection\_name*.drop()

eg. Drop collection named Product:

db.Product.drop()

4. Rename collections: db.*collection\_name*.renameCollection("*new\_collection\_name*")

eg. db.Product.renameCollection("product")

---------------------------------------------------

Commands for Documents:

* **Insertion Queries:**

1. Insert 1 Document: db.*collection\_name*.insertOne({“field1”:“data1”, “field2”:“data2”})

eg. Insert a document with name,id,price of product in Product collection:

db.Product.insertOne({

'Name':'Clothes',

'Id':12345,

'Price':300})

2. Insert Multiple Documents:

db.*collection\_name*.insertMany([{//perform query where},{//perform query where}])

eg. db.Product.insertMany([

{'Name':'Mobile', 'Brand':'One Plus', 'Id':909, 'Price':25000},

{'Name':'Laptop','Id':100, 'Price':125000},

{‘Name’: ‘Charger’, //nested object ‘Specifications’:{‘Power’: ‘80W’, ‘Brand’: ‘OnePlus’},//array ‘Variants’: [‘30W’ , ‘50W’]}

])

* **Searching Queries:**

3. View all documents in a Collection:

db.*collection\_name*.find() OR

db.*collection\_name*.find().pretty()

eg. View all documents in Collection Product:

db.Product.find()

4. Search a document in collection with specified field:

db.*collection\_name*.find({//perform query where})

eg. db.Product.find({Id:100})

5. Limit:

db.*collection\_name*({//perform query where}).limit(//val)

eg. db.Product.find({Price:500,Category:“Electronics”}.limit(3)

6. View in sorted form (ascending):

db.*collection\_name*.find().sort({*field*:1})

eg. db.Product.find().sort({Price:1})

7. View in sorted form (descending)

db.*collection\_name*.find().sort({*field*:-1})

eg. db.Product.find().sort({Price:-1})

8. Count no of rows:

db.*collection\_name*.find(//perform query where).count()

eg. db.Product.find({Category:“Electronics”}).count()

9. Find only 1 document with specified field:

db.*collection\_name*.findOne(//perform query where)

eg. db.Product.findOne({Price:1000})

1. Include Specific field in Result:

db.*collection\_name.*find({“*search\_field*”: “data”},

{“*include\_field*”:1})

eg. db.Product.find({Name:"Clothes"},{Name:1})

1. Exclude Specific Field in Result:

db.*collection\_name.*find({“*search\_field*”: “data”},

{“ex*clude\_field*”:1})

eg. db.Product.find({Name:"Clothes"},{Name:0})

**NOTE:** Inclusion exclusion cannot be done in same projection command:

eg. db.Product.find({Name:"Clothes"},{Price:1},{Id:0})

This will give the following error:

MongoServerError: Cannot do exclusion on field Id in inclusion projection

1. Finding field from an object within the main object:

eg. db.movies.find({"Specifications.Power":80W)

1. Finding data greater than equal to a value:

db.*collection\_name.*find*({field\_name:{*$gte*:val}})*

*eg.* db.Product*.*find*({Price:{*$gte*:500}})*

1. Finding data greater than a value:

db.*collection\_name.*find*({field\_name:{*$gt*:val}})*

*eg.* db.Product*.*find*({Price:{*$gt*:500}})*

1. Finding data less than equal to a value:

db.*collection\_name.*find*({field\_name:{*$lte*:val}})*

*eg.* db.Product*.*find*({Price:{*$lte*:500}})*

1. Finding data lesser than a value:

db.*collection\_name.*find*({field\_name:{*$lt*:val}})*

*eg.* db.Product*.*find*({Price:{*$lt*:500}})*

* **Update Queries:**

1. Update 1 pre-existing document:

db.*collection\_name*.updateOne({//perform query where},

{$set:{update\_field: “*new\_data*”}})

eg. db.Product.updateOne({ Name: 'Charger' },

{ $set:{Name: 'Charger', Id: 805 }})

**NOTE**: $set operator is used in update queries. If you forget to write $set while updating then error will be shown:

eg: db.Product.updateOne({Name:'Clothes',Id:12},{Id:123})

error:MongoInvalidArgumentError: Update document requires atomic operators

1. Update non-existing document:

db.*collection\_name*.updateOne({//perform query where},

{$set:{update\_field: “*new\_data*”}},

{upsert:true}//this updates+inserts if specified document doesn’t already exist

)

eg. db.Product.update({ Name: 'Charger' },

{ $set:{Name: 'Charger', Id: 805 }},

{upsert:true})

**NOTE:**If you write update then only one field is updated with there are multiple same name fields,

so it is better to use updateOne or updateMany

1. Update Many:

db.*collection\_name*.updateMany({//perform query where},{$set:{update\_field: *“new\_data”*}})

eg. db.Product.updateMany({Name:'Clothes'},{$set:{Price:1000}})

1. Increment Value:

db.*collection\_name*.updateMany({//perform query where},

{$inc:{update\_field: *“incr\_by”*}})

eg. db.Product.updateMany({Name:'Clothes'},{$inc:{Price:20}})

1. Decrement Value:

db.*collection\_name*.updateOne(

{ //perform query where}, { $inc: { update\_field: *“decr\_by”*} }

)

eg. db.Product.updateOne({Name:'Clothes'},{$inc:{Price:-20}})

* Delete Document:

22. Delete One Document:

db.*collection\_name*.deleteOne({*delete\_field*:'*new\_data*'})

eg. db.Product.deleteOne({Name:'Clothes'})

1. Delete Many Documents:

db.*collection\_name*.deleteMany({*delete\_field*:'*new\_data*'})

eg. db.Product.deleteMany({Name:'Clothes'})

---------------------------------------------------

Commands for Field:

1. Delete Field:

db.c*ollection\_name*.updateMany({},

{$unset:{*field\_name*:"*data*"}})

eg. db.veg.updateMany({},{$unset:{name:"Clothes"}})

24. Rename Field:

db.collection\_name.updateMany({},

{$rename:{"oldField\_name":"newField\_name"}})

eg. db.veg.updateMany({},{$rename:{"name":"dish\_name"}})