**#MONGO DB CRUD**

**#Creating a New Database**

use userdb;

**#Creating Collection**

db.createCollection("posts")

show collections

**#Creating Documents**

#insertOne():

db.users.insertOne({ name: "Angela", age: 27 });

#insertMany():

db.users.insertMany([{name: "User1", age: 22}, {name: "User2", age: 25}]);

**#Read All the Documents**

db.users.find();

**#Read Document with specific query**

db.users.find({name: "User1"});

**#Read Document with query & projection**

db.users.find({name: "User1"}, {\_id:0, age:0});

**#findOne() : Returns a single document object**

db.users.findOne({name: "User1"}, {\_id:0, age:0});

**#Update Operations**

**#updateOne()**

db.users.updateOne({ age: { $lt: 23 } }, { $set: { status: "active" } });

**#updateMany()**

db.users.updateMany({ age: { $gt: 23 } }, { $set: { status: "inactive" } });

**#Delete Operations**

**#deleteOne()**

**db.**users**.deleteOne({ name: "User2" });**

**#deleteMany()**

db.users.deleteMany({name:"User1"})

**#Delete Collection**

db.users.drop()

**#Delete Database**

db.dropDatabase()

**# Importing Data into MongoDB using mongoimport**

#Use mongoimport to import JSON data into a MongoDB collection.

mongoimport --db <database\_name> --collection <collection\_name> --file <path\_to\_json\_file>

#Use mongoimport to import CSV data into a MongoDB collection:

mongoimport --db <database\_name> --collection <collection\_name> --type csv --headerline --file <path\_to\_csv>

**#MongoDB Operators**

**Dataset Name: bank.sales.csv**

**Comparison Operators**

* $eq: Matches values that are equal to a specified value.
* $gt: Matches values that are greater than a specified value.
* $lt: Matches values that are less than a specified value.
* $gte: Matches values that are greater than or equal to a specified value.
* $lte: Matches values that are less than or equal to a specified value.
* $ne: Matches all values that are not equal to a specified value.
* $in : Matches any of the values specified in an array.

#Find one document where Value is greater than 100000

db.sales.findOne({ Value: { $gt: 100000 } })

#Find Number of documents where Transaction\_count is less than or equal to 1000

db.sales.find({ Transaction\_count: { $lte: 1000 } }).count()

# Find Number of documents where Location is an array containing 'Mumbai'

db.sales.find({ Location: { $in: ['Mumbai' , 'Pune'] } }).count()

**Logical Operators**

* $and: Joins query clauses with a logical AND and returns all documents that match the conditions of both clauses.
* $or: Joins query clauses with a logical OR and returns all documents that match the conditions of either clause.
* $not: Inverts the effect of a query expression and returns documents that do not match the query expression.
* $nor: Joins query clauses with a logical NOR and returns all documents that fail to match both clauses.

# Find documents where Location is 'Mumbai' or 'Delhi' using logical operators

db.sales.find({ $or: [ { Location: 'Mumbai' }, { Location: 'Delhi' } ] })

# Find documents where Domain is 'RETAIL' and Location is 'Pune' using logical operators

db.sales.find({ $and: [ { Domain: 'RETAIL' }, { Location: 'Pune' } ] })

**Element Operators**

* $exists: Matches documents that contain the specified field.
* $type: Matches documents based on the BSON type of the field.

#Find documents where Transaction\_count does not exist

db.sales.find({ Value: { $type: 'double' } })

#Find Number of documents where Domain exists

db.sales.find({ Domain: { $exists: true } }).count()

**Array Operators**

* $elemMatch: Matches documents that contain an array field with at least one element that matches all the specified query criteria.
* $size: Matches documents where the array field is a specific size.

**#Indexing**

#Phele humko mongo campass

#Waha pe jake data import krna hai

#Phir cmd pe jake mongosh

Use div **(where collection name is (test))**

Show collections

db.test.findOne()

db.test.find({ Location: 'Bhuj' }).explain("executionStats");

#upr wala without indexing hai

#Create index

db.test.createIndex({ Location: 1 });

#to show Indexes present in collection

db.test.getIndexes()

#aab same command with indexing run krna hai

db.test.find({ Location: 'Bhuj' }).explain("executionStats");

**#Replication**

#Phele three folder banna hai

Primary Replica1 Replica2

1.CMD

mongod --port 27070 --dbpath "C:\Users\MANOJ JHA\OneDrive\Documents\replications\primary" --replSet rs

2. CMD

mongod --port 27071 --dbpath "C:\Users\MANOJ JHA\OneDrive\Documents\replications\repl1" --replSet rs

3. CMD

mongod --port 27072 --dbpath "C:\Users\MANOJ JHA\OneDrive\Documents\replications\repl2" --replSet rs

4. CMD

mongosh --port 207070

rs.initiate({\_id:"rs",members:[{\_id:0,host:"localhost:27070"},{\_id:1,host:"localhost:27071"},{\_id:2,host:"localhost:27072"}]})

rs.status()

use div

db.test.insertOne({name:"divya"})

5. CMD  
mongosh --port 27072 (SECONDARY KA NUMBER)

show dbs

use div

db.test.find()

db.getMongo().setReadPref("secondary")

db.test.find()

**#Sharding**

Create 7 folder

Primary shard1 shard2 config1 config2 replica1 replica2

#Start Instance for Config Server & ReplicaSet

1. CMD

mongod --configsvr --port 37017 --replSet configReplSet --dbpath "C:\MongoDB\_Sharding\cfg"

1. CMD

mongod --configsvr --port 37018 --replSet configReplSet --dbpath "C:\MongoDB\_Sharding\cfg\_repl"

# Start Instance for Shard 1 & ReplicaSet1

1. CMD

mongod --shardsvr --port 47017 --replSet shard1 --dbpath "C:\MongoDB\_Sharding\shd1"

1. CMD

mongod --shardsvr --port 47018 --replSet shard1 --dbpath "C:\MongoDB\_Sharding\shd1\_repl"

#Start Instance for Shard 2 & ReplicaSet2

1. CMD

mongod --shardsvr --port 57017 --replSet shard2 --dbpath "C:\MongoDB\_Sharding\shd2"

1. CMD

mongod --shardsvr --port 57018 --replSet shard2 --dbpath "C:\MongoDB\_Sharding\shd2\_repl"

1. CMD

#Configure ReplicaSet for ConfigDB

mongosh --port 37017 #config 1 ka number

rs.initiate({\_id:"configReplSet",members:[{\_id:0,host:"localhost:37017"},{\_id:1,host:"localhost:37018"}]})

exit

#Configure ReplicaSet for Shard1

mongosh --port 47017 #shard1 ka number

rs.initiate({\_id:"shard1",members:[{\_id:0,host:"localhost:47017"},{\_id:1,host:"localhost:47018"}]})

exit

#Configure ReplicaSet for Shard2

mongosh --port 57017 #shard2 ka number

rs.initiate({\_id:"shard2",members:[{\_id:0,host:"localhost:57017"},{\_id:1,host:"localhost:57018"}]})

exit

1. CMD  
   mongos --configdb configReplSet/localhost:37017,localhost:37018 --port 27017

#Next step mongo campass mai jake apne router se connect krne ka jaise --port 27017 yeh daalne ka link mai

#Then next step mai database banane ka and usme collection banana ka usme data import krne ka

1. CMD

mongosh --port 27017

sh.addShard("shard1/localhost:47017,localhost:47018")

sh.addShard("shard2/localhost:57017,localhost:57018")

sh.enableSharding(demo) #database name is demo

sh.status()

#Create Index of Collection

db.zip.createIndex({Pincode : 1}) # zip is collection name

sh.shardCollection("demo.zip", { Pincode : 1} )

db.zip.getShardDistribution()

#Manually Splitting Chunk (Optional Step)

sh.splitAt("demo.zip", { Pincode : 200000})

sh.splitAt("demo.zip", { Pincode : 400000})

sh.splitAt("demo.zip", { Pincode : 600000})

#Move Chunk Manually From One Shard to Other

sh.moveChunk("demo.zip", { Pincode : 400000}, "shard2")

sh.moveChunk("demo.zip", { Pincode : 600000}, "shard2")

**#NEO4J**

#Crud Operation in Neo4j

#Creating nodes with labele Person

CREATE (:Person {name: 'Joe'})

CREATE (:Person {name: 'Jhon'})

CREATE (:College {name: 'Mithibai'})

CREATE (:College {name: 'Wilson'})

# Creating Relationship between Nodes

match (a:Person),(b:College) where a.name="Joe" and b.name="Mithibai" create(a)-[:studyin]->(b)

match (a:Person),(b:College) where a.name="Jhon" and b.name="Wilson" create(a)-[:studyin]->(b)

match (a:Person),(b:Person) where a.name="Jhon" and b.name="Joe" create(a)-[:friend]->(b)

# Findig/Retreving Nodes

match (a:Person) return a

match (a:Person) return a.name

# Updating Node Properties

match (a:Person{name:"Joe"}) set a.age=20

match (a:Person{name:"Jhon"}) set a.age=21

# Finding Age of Particular Node with Property Name

match (a:Person) where a.name="Jhon" return a.age #table view

# Deleting Node & Relationship

# Detaching Relationship & Delteting Node

match (a:Person) where a.name="Jhon" detach delete a

#Detaching Relationship using label

MATCH (n:Person {name: 'Joe'})-[r:studyin]->() DELETE r #detaching relation ship