**Practical 1: MongoDB Basics**

**a. Write a MongoDB query to create and drop database.**

> use Shivam

> db.dropDatabase()

**b. Write a MongoDB query to create, display and drop collection.**

**Create a Collection**

> db.createCollection("myCollection")

**Display Collections**

> show collections

**Drop a Collection**

> db.myCollection.drop()

**c. Write a MongoDB query to insert, query, update and delete a document.**

**Insert a Document**

> db.myCollection.insert({“Sid” : ”t.22.122”, “name” : “Shivam”, “age” : 20, “Profession” : “Software Engineer”})

**Query a Document**

> db.myCollection.find({ age: 25 })

**Update a Document**

> db.mycollection.update({“name”: “Shvam”}, {“$set : {“profession” : “senior developer”}})

**Delete a Document**

> db.myCollection.deleteOne({ name: "Shivam" })

> db.myCollection.deleteMany({ age: { $lt: 30 } })

**Practical 2: Simple Queries with MongoDB**

**Practical 3: Implementing Aggregation**

**a. Write a MongoDB query to use sum, avg, min and max expression.**

**First Create Employee Collection**

> db.employee.insertMany([{ "\_id": 11,"name": "Alice", "age": 25, "salary":3000 },{ "\_id": 12, "name": "Bob", "age": 30, "salary": 4000 },{ "\_id": 21, "name": "Charlie", "age": 35, "salary": 5000 },{ "\_id": 19, "name": "David", "age": 40, "salary": 6000 },{ "\_id": 20, "name": "Eve","age": 45, "salary": 7000 }])

**1. Sum of Salaries:**

**>** db.employee.aggregate([ { $group: { \_id: null, totalSalary: { $sum: "$salary" } } }] )

**2. Average Age:**

**>** db.cemployee.aggregate([ { $group: { \_id: null, averageAge: { $avg: "$salary" }}}] )

**3. Minimum Age:**

**>** db.employee.aggregate([ { $group: { \_id: null, minAge: { $min: "$age" } } }] )

**4. Maximum Salary:**

**>** db.employee.aggregate([ { $group: { \_id: null, maxSalary: { $max: "$salary" } } }] )

**Practical 4: Replication, Backup and Restore**

**a. Write a MongoDB query to create Replica of existing database.**

Create a folder name data(optional) inside that create three more folder such as

db1, db2, db3 -------------------🡪 remember the folder location

ex- C:\data\db1 --replSet rs1

ex- C:\data\db2 --replSet rs1

ex- C:\data\db3 --replSet rs1

note: default MongoDB port is 27017 better to Go with 27018 and and so on

**open the Window terminal**

* start mongod --port 27018 --dbpath C:\data\db1 --replSet rs1
* start mongod --port 27019 --dbpath C:\data\db2 --replSet rs1
* start mongod --port 27020 --dbpath C:\data\db3 --replSet rs1

**Here:** 3 different terminals will start those are your instances which are listening on ports

27018, 27019, 27020. In this one will be primary and two will be secondary.

**open new terminal:**

* **mongosh --port 27018**
* **test>** rs.initiate({ id:"rs1",members: [{ \_id:0,host:"localhost:27018"},

{\_id:1, host:"localhost:27019"},

{\_id:2, host:"localhost:27020"}

]})

// this will initialize all the members

**Find which one is Primary and Secondary(log):**

* **rs.status()**

**perform all in port 27018 (primary)**

* rs1 [direct: primary] test> show dbs
* rs1 [direct: primary] test> use College
* rs1 [direct: primary] College> db.createCollection("Student")
* rs1 [direct: primary] College> db.Student.insert({name:"Shivam Prajapati", age: 20, departement:"IT"})
* rs1 [direct: primary] College> db.Student.find()

**Now Check whether database and data are replicated in all secondary or not**

**Open New terminal:**

* mongosh --port 27019
* rs1 [direct: secondary] test> show dbs
* rs1 [direct: secondary] admin> use College
* db.Student.find()

**Error:** MongoServerError: not primary and secondaryOk=false - consider using db.getMongo().setReadPref() or readPreference in the connection string

* db.getMongo().setSecondaryOk()
* rs1 [direct: secondary] College> db.Student.find()

**all set you will be now able to see all data from secondary but cant perform write operations from Secondary**

**(Same with port 27020)**

**b) Write a MongoDB query to create a backup of existing database.**

**Using mongodump:** this is an part of MongoDB database used to create backup of existing database rather than writing a query.

* mongodump --db yourDatabaseName --out /path/to/backup

/path/to/backup: this is my backup file location my data will store here

**c) Write a MongoDB query to restore database from the backup.**

**Using mongorestore:** this is an another tool used for backup.the mongorestore command restores data from the BSON files generated by mongodump.

* mongorestore --db yourDatabaseName /path/to/backup/yourDatabaseName

path/to/backup/yourDatabaseName : this is the location where you had taken the backup the data base.

**Practical 5. Java and MongoDB**

a. Connecting Java with MongoDB and inserting, retrieving, updating and deleting.

**ConnectingJavaToMongo:**