Perform the following queries using update, delete, rename and createcollection method:

Part – A (Use collection “Student” created in Lab-9)

{ "\_id": 1, "name": "John", "age": 30, "city": "New York", "isActive": true }

{ "\_id": 2, "name": "Jane", "age": 25, "city": "Los Angeles", "isActive": false }

{ "\_id": 3, "name": "Tom", "age": 35, "city": "Chicago", "isActive": true }

{ "\_id": 4, "name": "Lucy", "age": 28, "city": "San Francisco", "isActive": true }

{ "\_id": 5, "name": "David", "age": 40, "city": "Miami", "isActive": false }

{ "\_id": 6, "name": "Eva", "age": 23, "city": "Boston", "isActive": true }

{ "\_id": 7, "name": "Nick", "age": 38, "city": "Seattle", "isActive": false }

{ "\_id": 8, "name": "Sophia", "age": 27, "city": "New York", "isActive": true }

{ "\_id": 9, "name": "Liam", "age": 32, "city": "Los Angeles", "isActive": false }

{ "\_id": 10, "name": "Olivia", "age": 29, "city": "San Diego", "isActive": true }

1. Update the age of John's to 31.

db.Student.updateOne({name:'John'}, {$set:{age:31}})

1. Update the city of all students from 'New York' to 'New Jersey'.

db.Student.updateMany({city:'New York'}, {$set:{city:'New Jersey'}})

1. Set isActive to false for every student older than 35.

db.Student.updateMany({age:{$gt:35}}, {$set:{isActive:false}})

1. Increment the age of all students by 1 year.

db.Student.updateMany({}, {$inc:{age:1}})

1. Set the city of 'Eva' to 'Cambridge'.

db.Student.updateOne({name:'Eva'}, {$set:{city:'Cambridge'}})

1. Update 'Sophia's isActive status to false.

db.Student.updateOne({name:'Sophia'}, {$set:{isActive:false}})

1. Update the city field of student aged below 30 to 'San Diego'.

db.Student.updateMany({age:{$lt:30}}, {$set:{city:'San Diego'}})

1. Rename the age field to years for all documents.

db.Student.updateMany({}, {$rename:{'age':'years'}})

1. Update 'Nick' to make him active (isActive = true).

db.Student.updateOne({name:'Nick'}, {$set:{isActive:true}})

1. Update all documents to add a new field country with the value 'USA'.

db.Student.updateMany({}, {$set:{country:'USA'}})

1. Update 'David's city to 'Orlando'.

db.Student.updateOne({name:'David'}, {$set:{city:'Orlando'}})

1. Multiply the age of all students by 2.

db.Student.updateMany({}, {$mul:{years:2}})

1. Unset (remove) the city field for 'Tom'.

db.Student.updateOne({name:'Tom'}, {$unset:{city:''}})

1. Add a new field premiumUser and to true for users older than 30.

db.Student.updateMany({years:{$gt:30}}, {$set:{premiumUser:true}})

1. Set isActive to true for 'Jane'.

db.Student.updateOne({name:'Jane'}, {$set:{isActive:true}})

1. Update isActive field of 'Lucy' to false.

db.Student.updateOne({name:'Lucy'}, {$set:{isActive:false}})

1. Delete a document of 'Nick' from the collection.

db.Student.deleteOne({name:'Nick'})

1. Delete all students who are inactive (isActive = false).

db.Student.deleteMany({isActive:false})

1. Delete all students who live in 'New York'.

db.Student.deleteMany({city:'New York'})

1. Delete all the students aged above 35.

db.Student.deleteMany({years:{$gt:35}})

1. Delete a student named ''Olivia'' from the collection.

db.Student.deleteOne({name:'Olivia'})

1. Delete all the students whose age is below 25.

db.Student.deleteMany({years:{$lt:25}})

1. Delete the first student whose isActive field is true.

db.Student.deleteOne({isActive:true})

1. Delete all students from 'Los Angeles'.

db.Student.deleteMany({city:'Los Angeles'})

1. Delete all students who have city field missing.

db.Student.deleteMany({city:{$exists:false}})

1. Rename 'city' field to 'location' for all documents.

db.Student.updateMany({}, {$rename:{'city':'location'}})

1. Rename the name field to FullName for 'John'.

db.Student.updateMany({name:'John'}, {$rename:{'name':'FullName'}})

1. Rename the isActive field to status for all documents.

db.Student.updateMany({}, {$rename:{'isActive':'status'}})

1. Rename age to yearsOld for everyone from 'San Francisco' student only.

db.Student.updateMany({location:'San Francisco'}, {$rename:{'years':'yearsOld'}})

30. Create a Capped Collection named “Employee” as per follows:

a. Ecode and Ename are compulsory fields

b. Datatype of EID is int, Ename is string, Age is int and City is string Insert following documents into above “Employee” collection.

{"Ecode": 1, "Ename": "John"}

{"Ecode ": 2, "Ename": "Jane", "age": 25, "city": "Los Angeles"}

{"Ecode ": 3, "Ename": "Tom", "age": 35}

{"Ecode ": 4, "Ename": "Lucy", "age": 28, "city": "San Francisco", "isActive": true}

{"Ename": "Dino"}

db.createCollection('Employee',

{

capped: true,

size: 5120,

max: 100,

validator: {

$jsonSchema: {

bsonType: 'object',

required: ['Ecode', 'Ename'],

properties: {

Ecode: {bsonType: 'int'},

Ename: {bsonType: 'string'},

Age: {bsonType: 'int'},

City: {bsonType: 'string'}

}

}

}

})

db.Employee.insertMany([

{"Ecode": 1, "Ename": "John"},

{"Ecode": 2, "Ename": "Jane", "age": 25, "city": "Los Angeles"},

{"Ecode": 3, "Ename": "Tom", "age": 35},

{"Ecode": 4, "Ename": "Lucy", "age": 28, "city": "San Francisco", "isActive": true},

])

Part – B Create collection named “Student\_data” and insert following 10 documents into it.

* db.createCollection('Student\_data')

db.Student\_data.insertMany([

{

"ROLLNO": 101,

"SNAME": "Vina",

"DEPARTMENT": "CE",

"FEES": 15000,

"SEM": 3,

"GENDER": "Female",

"CITY": "Rajkot"

},

{

"ROLLNO": 102,

"SNAME": "Krisha",

"DEPARTMENT": "EC",

"FEES": 8000,

"SEM": 5,

"GENDER": "Female",

"CITY": "Ahmedabad"

},

{

"ROLLNO": 103,

"SNAME": "Priti",

"DEPARTMENT": "Civil",

"FEES": 12000,

"SEM": 7,

"GENDER": "Female",

"CITY": "Baroda"

},

{

"ROLLNO": 104,

"SNAME": "Mitul",

"DEPARTMENT": "CE",

"FEES": 15000,

"SEM": 3,

"GENDER": "Male",

"CITY": "Rajkot"

},

{

"ROLLNO": 105,

"SNAME": "Keshav",

"DEPARTMENT": "CE",

"FEES": 15000,

"SEM": 3,

"GENDER": "Male",

"CITY": "Jamnagar"

},

{

"ROLLNO": 106,

"SNAME": "Zarna",

"DEPARTMENT": "Civil",

"FEES": 12000,

"SEM": 5,

"GENDER": "Female",

"CITY": "Ahmedabad"

},

{

"ROLLNO": 107,

"SNAME": "Nima",

"DEPARTMENT": "EE",

"FEES": 9000,

"SEM": 5,

"GENDER": "Female",

"CITY": "Rajkot"

},

{

"ROLLNO": 108,

"SNAME": "Dhruv",

"DEPARTMENT": "Mechanical",

"FEES": 10000,

"SEM": 5,

"GENDER": "Male",

"CITY": "Rajkot"

},

{

"ROLLNO": 109,

"SNAME": "Krish",

"DEPARTMENT": "Mechanical",

"FEES": 10000,

"SEM": 7,

"GENDER": "Male",

"CITY": "Baroda"

},

{

"ROLLNO": 110,

"SNAME": "Zeel",

"DEPARTMENT": "EE",

"FEES": 9000,

"SEM": 3,

"GENDER": "Female",

"CITY": "Jamnagar"

}

])

From the above given “Student\_data” collection perform the following queries:

1. Display Female students and belong to Rajkot city.

db.Student\_data.find({$and:[{GENDER:'Female'},{CITY:'Rajkot'}]})

1. Display students not studying in 3rd sem.

db.Student\_data.find({SEM:{$ne:3}})

1. Display students whose city is Jamnagar or Baroda. (use: IN)

db.Student\_data.find({CITY:{$in:['Jamnagar','Baroda']}})

1. Display first 2 students names who lives in Baroda.

db.Student\_data.find({CITY:'Rajkot'},{\_id:0,SNAME:1}).limit(2)

1. Display Male students who studying in 3rd sem.

db.Student\_data.find({$and:[{GENDER:'Male'},{SEM:3}]})

1. Display sname and city and fees of those students whose roll no is less than 105.

db.Student\_data.find({ROLLNO:{$lt:105}},{\_id:0,SNAME:1,CITY:1,FEES:1})

1. Update City of all students from 'Jamnagar' City and Department as 'CE' to 'Surat'.

db.Student\_data.updateMany({$and:[{CITY:'Jamnagar'},{DEPARTMENT:'CE'}]},{$set:{CITY:'Surat'}})

1. Increase Fees by 500 where the Gender is not 'Female'. (Use: Not)

db.Student\_data.updateMany({GENDER:{$not:{$eq:"Female"}}},{$inc:{FEES:500}})

1. Set the Department of all students from 'EE' and in Sem 3 to 'Electrical'.

db.Student\_data.updateMany({$and:[{SEM:3},{DEPARTMENT:'EE'}]},{$set:{DEPARTMENT:'Electrical'}})

1. Update the Fees of students in 'Rajkot' who are male.

db.Student\_data.updateMany({$and:[{CITY:'Rajkot'},{GENDER:'Male'}]},{$inc:{FEES:500}})

1. Change City to 'Vadodara' for students in Sem 5 and with fees less than 10000.

db.Student\_data.updateMany({$and:[{SEM:5},{FEES:{$lt:10000}}]},{$set:{CITY:'Vadodara'}})

1. Delete all students where the City is 'Ahmedabad' or GENDER is 'Male'.

db.Student\_data.deleteMany({$or:[{CITY:'Ahmedabad'},{GENDER:'Male'}]})

1. Delete students whose Rollno is not in the list [101, 105, 110].

db.Student\_data.deleteMany({ROLLNO:{$nin:[101,105,110]}})

1. Delete students from the 'Civil' department who are in Sem 5 or Sem 7.

db.Student\_data.deleteMany({$and:[{DEPARTMENT:'Civil'},{SEM:{$in:[5,7]}}]})

1. Delete all students who are not in the cities 'Rajkot', 'Baroda', or 'Jamnagar'.

db.Student\_data.deleteMany({CITY:{$nin:['Rajkot','Baroda','Jamnagar']}})

1. Delete students whose Rollno is between 105 and 108.

db.Student\_data.deleteMany({$and:[{ROLLNO:{$gte:105}},{ROLLNO:{$lte:108}}]})

1. Rename the City field to LOCATION for all students.

db.Student\_data.updateMany({},{$rename:{'CITY':'LOCATION'}})

1. Rename the Department field to Branch where the Fees is less than 10000.

db.Student\_data.updateMany({FEES:{$lt:10000}},{$rename:{'DEPARTMENT':'BRANCH'}})

1. Rename Sname to Fullname for students with Rollno in [106, 107, 108].

db.Student\_data.updateMany({ROLLNO:{$in:[106,107,108]}},{$rename:{'SNAME':'FULLNAME'}})

1. Rename Fees to Tuition\_Fees for all students with Fees greater than 9000.

db.Student\_data.updateMany({FEES:{$gt:9000}},{$rename:{'FEES':'TUTION\_FEES'}})

1. Rename Department to Major where the Fees is less than 15000 and Gender is 'Female'.

db.Student\_data.updateMany({$and:[{FEES:{$lt:15000}},{GENDER:'Female'}]},{$rename:{'BRANCH':'MAJOR'}})

1. Rename City to Hometown for all students whose SEM is 3 and Department is not 'Mechanical'.

db.Student\_data.updateMany({$and:[{SEM:3},{BRANCH:{$not:{$eq:'Mechanical'}}}]},{$rename:{'LOCATION':'HOMETOWN'}})

Part – C

1. Create a capped collection named” logs” with a maximum size of 100 KB and a maximum of 10 documents.

db.createCollection("logs", {

capped: true,

size: 100 \* 1024,

max: 10

})

1. Insert below 12 log entries into the “logs” collection. Each entry should contain a message, level (e.g., "info", "warning", "error"), and a timestamp field. Use the insertMany() method.

{ message: "System started", level: "info", timestamp: new Date() }

{ message: "Disk space low", level: "warning", timestamp: new Date() }

{ message: "User login", level: "info", timestamp: new Date() }

{ message: "System reboot", level: "info", timestamp: new Date() }

{ message: "Error in module", level: "error", timestamp: new Date() }

{ message: "Memory usage high", level: "warning", timestamp: new Date() }

{ message: "User logout", level: "info", timestamp: new Date() }

{ message: "File uploaded", level: "info", timestamp: new Date() }

{ message: "Network error", level: "error", timestamp: new Date() }

{ message: "Backup completed", level: "info", timestamp: new Date() }

{ message: "Database error", level: "error", timestamp: new Date() }

{ message: "Service started", level: "info", timestamp: new Date() }

* db.logs.insertMany([

{ message: "System started", level: "info", timestamp: new Date() },

{ message: "Disk space low", level: "warning", timestamp: new Date() },

{ message: "User login", level: "info", timestamp: new Date() },

{ message: "System reboot", level: "info", timestamp: new Date() },

{ message: "Error in module", level: "error", timestamp: new Date() },

{ message: "Memory usage high", level: "warning", timestamp: new Date() },

{ message: "User logout", level: "info", timestamp: new Date() },

{ message: "File uploaded", level: "info", timestamp: new Date() },

{ message: "Network error", level: "error", timestamp: new Date() },

{ message: "Backup completed", level: "info", timestamp: new Date() },

{ message: "Database error", level: "error", timestamp: new Date() },

{ message: "Service started", level: "info", timestamp: new Date() }

])

1. Perform find method on “logs” collection to ensure only the last 10 documents are retained (even though you inserted 12).

db.logs.find()

1. Insert below 5 more documents and check if the oldest ones are automatically removed.

{ message: "New log entry 1", level: "info", timestamp: new Date() }

{ message: "New log entry 2", level: "info", timestamp: new Date() }

{ message: "New log entry 3", level: "info", timestamp: new Date() }

{ message: "New log entry 4", level: "warning", timestamp: new Date() }

{ message: "New log entry 5", level: "error", timestamp: new Date() }

* db.logs.insertMany([

{ message: "New log entry 1", level: "info", timestamp: new Date() },

{ message: "New log entry 2", level: "info", timestamp: new Date() },

{ message: "New log entry 3", level: "info", timestamp: new Date() },

{ message: "New log entry 4", level: "warning", timestamp: new Date() },

{ message: "New log entry 5", level: "error", timestamp: new Date() }

])