**Day-1**

**Essential Assignment:-**

**Create a new MongoDB database called “TEACHER”.**

**Within “TEACHER” database, create a collection named “TEACHER\_MASTER”.**

**Assume an appropriate Schema consisting of fields like Name, Age, Subject(array),**

**DOB, Gender, Salary, City.**

**1. Insert 7 documents into the above collection.**

db.Teacher\_Master.insert({\_id:1,name:"Akhil­=",age:21,subject:["hindi","gujarati","maths","eng"],dob:"11

/12/2002",gender:"female",salary:20000,city:"ahemadabad"}) db.Teacher\_Master.insert({\_id:2,name:"Ravi",age:24,subject:["sanskrit","gujarati","maths","eng"],dob: "11/12/2001",gender:"female",salary:30000,city:"surat"}) db.Teacher\_Master.insert({\_id:3,name:"jay",age:22,subject:["s.s","gujarati","maths","eng"],dob:"19/ 1/2000",gender:"male",salary:40000,city:"ahemadabad"}) db.Teacher\_Master.insert({\_id:4,name:"hit",age:25,subject:["hindi","phy","maths","che"],dob:"21/2

/2007",gender:"male",salary:60000,city:"baroda"}) db.Teacher\_Master.insert({\_id:5,name:"kajal",age:25,subject:["hindi","phy","maths","che"],dob:"24/7/2 004",gender:"female",salary:60000,city:"amerali"}) db.Teacher\_Master.insert({\_id:6,name:"samir",age:27,subject:["che","phy","maths","che"],dob:"26/8/2 007",gender:"female",salary:50000,city:"jamnager"}) db.Teacher\_Master.insert({\_id:7,name:"jenil",age:30,subject:["hindi","phy","eng","che"],dob:"28/2/2 003",gender:"male",salary:55000,city:"navsari"})

**2. Display Name, Subject, Gender and Salary.**

db.Teacher\_Master.find({},{\_id:0,name:1,gender:1,salary:1})

{ name: 'Akhil', gender: 'female', salary: 20000 },

{ name: 'Ravi', gender: 'female', salary: 30000 },

{ name: 'jay', gender: 'male', salary: 40000 },

{ name: 'hit', gender: 'male', salary: 60000 },

{ name: 'kajal', gender: 'female', salary: 60000 },

{ name: 'samir', gender: 'female', salary: 50000 },

{ name: 'jenil', gender: 'male', salary: 55000 }

**3. Display teacher, which are from the city “Ahmedabad”.**

db.Teacher\_Master.find({city:"ahemadabad"}) [

{

\_id: 1,

name: 'Akhil', age: 21,

subject: [ 'hindi', 'gujarati', 'maths', 'eng' ], dob: '11/12/2002',

gender: 'female', salary: 20000,

city: 'ahemadabad'

},

{

\_id: 3,

name: 'jay', age: 22,

subject: [ 's.s', 'gujarati', 'maths', 'eng' ], dob: '19/1/2000',

gender: 'male', salary: 40000,

city: 'ahemadabad'

}

]

**4. Display the teacher id, name, city and DOB.**

db.Teacher\_Master.find({},{name:1,city:1,dob:1})

{ \_id: 1, name: 'Akhil', dob: '11/12/2002', city: 'ahemadabad' },

{ \_id: 2, name: 'Ravi', dob: '11/12/2001', city: 'surat' },

{ \_id: 3, name: 'jay', dob: '19/1/2000', city: 'ahemadabad' },

{ \_id: 4, name: 'hit', dob: '21/2/2007', city: 'baroda' },

{ \_id: 5, name: 'kajal', dob: '24/7/2004', city: 'amerali' },

{ \_id: 6, name: 'samir', dob: '26/8/2007', city: 'jamnager' },

{ \_id: 7, name: 'jenil', dob: '28/2/2003', city: 'navsari' }

**5. Display the teachers whose gender is female and teach either “Hindi” or “English” subject.**

db.Teacher\_Master.find({$and:[{gender:"female"},{subject:{$in:["hindi","eng"]}},{subject:"eng"}]}) [

{

\_id: 1,

name: 'Akhil', age: 21,

subject: [ 'hindi', 'gujarati', 'maths', 'eng' ], dob: '11/12/2002',

gender: 'female', salary: 20000,

city: 'ahemadabad'

},

{

\_id: 2,

name: 'Ravi', age: 24,

subject: [ 'sanskrit', 'gujarati', 'maths', 'eng' ], dob: '11/12/2001',

gender: 'female', salary: 30000, city: 'surat'

}

]

**6. Update all those documents where name of teacher is “Anil” with the new value of subject as “English”.**

db.Teacher\_Master.updateMany({name:"kajal"},{$push:{"subject":"English"}})

{

\_id: 5,

name: 'kajal', age: 25,

subject: [ 'hindi', 'phy', 'maths', 'che', 'English' ], dob: '24/7/2004',

gender: 'female', salary: 60000, city: 'amerali'

},

**7. Delete data of all those teachers who were born after 1st January 1980.**

db.Teacher\_Master.remove({dob:{$gt:"11/12/2002"}})

**8. Remove field age.**

db.Teacher\_Master.updateMany({},{$unset:{age:""}})

{

\_id: 1,

name: 'Akhil',

subject: [ 'hindi', 'gujarati', 'maths', 'eng' ], dob: '11/12/2002',

gender: 'female', salary: 20000,

city: 'ahemadabad'

},

{

\_id: 2,

name: 'Ravi',

subject: [ 'sanskrit', 'gujarati', 'maths', 'eng' ], dob: '11/12/2001',

gender: 'female', salary: 30000, city: 'surat'

}

**9. Display the teachers that do not teach “English” subject and their salary is more than 30000.**

db.Teacher\_Master.find({subject:{$nin:["eng"]},salary:{$gt:50000}}) [

{

\_id: 4,

name: 'hit', age: 25,

subject: [ 'hindi', 'phy', 'maths', 'che' ], dob: '21/2/2007',

gender: 'male', salary: 60000, city: 'baroda'

},

{

\_id: 5,

name: 'kajal', age: 25,

subject: [ 'hindi', 'phy', 'maths', 'che', 'English' ], dob: '24/7/2004',

gender: 'female', salary: 60000, city: 'amerali'

}

]

**10. Find all the teachers having gender “Male” and display salary for them.**

db.Teacher\_Master.find({gender:"male"},{salary:1,name:1}) [

{ \_id: 3, name: 'jay', salary: 40000 },

{ \_id: 4, name: 'hit', salary: 60000 },

{ \_id: 7, name: 'jenil', salary: 55000 }

]

**Desirable:-**

'use EMPLOY db.createCollection("EMPLOYEE\_MASTER");

**Create a new MongoDB database called “EMPLOYEE”.**

**Within “EMPLOYEE” database, create a collection named EMPLOYEE\_MASTER” assume an appropriate schema consisting of fields like Empno, Name, Designation, DOJ, Department, Salary, Gender, Skills(array)**

1. **Insert 7 documents into the above collection**

db.EMPLOYEE\_MASTER.insert({\_id:101, Empno:"E101",

Name:"Akash",

Designation:"Teacher",

DOJ:"21/2/2003",

Department:"clerk",

Salary:34000,

Gender:"Male",

Skills:['PHP','JAVA','PHYTHON']})

db.EMPLOYEE\_MASTER.insert({\_id:102, Empno:"E102",

Name:"Amit",

Designation:"Engineer",

DOJ:"21/2/1998",

Department:"Computer",

Salary:40000,

Gender:"Male",

Skills:['MACHINELEARNING','JAVA','.NET']})

db.EMPLOYEE\_MASTER.insert({\_id:103, Empno:"E103",

Name:"Akhil",

Designation:"Accountant",

DOJ:"21/2/2002",

Department:"Ca",

Salary:50000,

Gender:"Female",

Skills:['TELLY','JAVA','PHYTHON']})

db.EMPLOYEE\_MASTER.insert({\_id:104, Empno:"E104",

Name:"samir",

Designation:"Teacher",

DOJ:"21/2/2003",

Department:"HR",

Salary:12000,

Gender:"Female",

Skills:['CS','MUSIC','READING']})

db.EMPLOYEE\_MASTER.insert({\_id:105, Empno:"E105",

Name:"hit",

Designation:"Scientist",

DOJ:"21/2/1990",

Department:"Arynotic",

Salary:100000,

Gender:"Male",

Skills:['C','JAVA','DSA']})

db.EMPLOYEE\_MASTER.insert({\_id:106, Empno:"E106",

Name:"jenil",

Designation:"Artist",

DOJ:"21/2/2009",

Department:"Painter",

Salary:5000, Gender:"Male",

Skills:['DRAWING','LISTING','READING']})

db.EMPLOYEE\_MASTER.insert({\_id:107, Empno:"E107",

Name:"Rah",

Designation:"IT",

DOJ:"21/2/2006",

Department:"HR",

Salary:34000,

Gender:"Male",

Skills:['DRAWING','LISTING','READING']})

1. **Display Name, Department, Gender and Salary.**

db.EMPLOYEE\_MASTER.find({},{Name:1,Department:1,Gender:1,Salary:1,\_id:0}).pretty()

[

{ Name: 'Akash', Department: 'clerk', Salary: 34000, Gender: 'Male' },

{ Name: 'Amit', Department: 'Computer',Salary: 40000, Gender: 'Male'},

{ Name: 'Akhil', Department: 'Ca', Salary: 50000, Gender: 'Female' },

{ Name: 'samir', Department: 'HR', Salary: 12000, Gender: 'Female' },

{Name: 'jenil', Department: 'Painter', Salary: 5000,Gender: 'Male'},

{Name: 'hit',Department: 'Arynotic',Salary: 100000, Gender: 'Male'}

]

1. **Display the list of employees from the department “HR”.**

db.EMPLOYEE\_MASTER.find({Department:"HR"},{Name:1})

[ { \_id: 104, Name: 'samir' },

{ \_id: 107, Name: 'Rah' } ]

1. **Display the employee id, name, department and DOB. db.EMPLOYEE\_MASTER.updateMany({"DOJ":{$ne:null}},{$rename:{"DOJ":"DOB"}}**

db.EMPLOYEE\_MASTER.find({},{Name:1,Department:1,DOB:1}).pretty()

[

{ \_id: 101, Name: 'Akash', Department: 'clerk', DOB: '21/2/2003' },

{ \_id: 102, Name: 'Amit', Department: 'Computer', DOB: '21/2/1998' },

{ \_id: 103, Name: 'Akhil', Department: 'Ca', DOB: '21/2/2002' },

{ \_id: 104, Name: 'samir', Department: 'HR', DOB: '21/2/2003' },

{ \_id: 106, Name: 'jenil', Department: 'Painter', DOB: '21/2/2009' },

{\_id: 105,Name: 'hit',Department: 'Arynotic',DOB: '21/2/1990'},

{ \_id: 107, Name: 'Rah', Department: 'HR', DOB: '21/2/2006' }

]

1. **Display the employees whose gender is female and designation either**

db.EMPLOYEE\_MASTER.find({ $and:[{Gender:"Female"},{Designation:{$in:["Accountant","Teacher"]}}]})

[

{

\_id: 103, Empno: 'E103',

Name: 'Akhil', Designation: 'Accountant', Department: 'Ca',

Salary: 50000, Gender: 'Female',

Skills: [ 'TELLY', 'JAVA', 'PHYTHON' ], DOB: '21/2/2002'

},

{

\_id: 104, Empno: 'E104',

Name: 'samir', Designation: 'Teacher', Department: 'HR', Salary: 12000,

Gender: 'Female',

Skills: [ 'CS', 'MUSIC', 'READING' ], DOB: '21/2/2003'

}

]

1. **Update all those documents where name of employee is “Akash” with the new value of designation as “Engineer”.**

db.EMPLOYEE\_MASTER.updateMany({Name:"Akash"},{$set:{Designation:}})

1. **Delete all those documents where DOJ is before 1st January 1999.**

db.EMPLOYEE\_MASTER.deleteMany({DOB:{$gt:{11/12/2000}}})

1. **Display the employees whose salary is greater than 25000 and have skills Java or PHP.**

db.EMPLOYEE\_MASTER.find({ $and:[{Salary:{$gt:25000}},{Skills:{$in:["PHP","JAVA"]}}]}) [

{

\_id: 101, Empno: 'E101',

Name: 'Akash', Designation: 'Teacher', Department: 'clerk', Salary: 34000,

Gender: 'Male',

Skills: [ 'PHP', 'JAVA', 'PHYTHON' ], DOB: '21/2/2003'

},

{

\_id: 102, Empno: 'E102',

Name: 'Amit', Designation: 'Engineer', Department: 'Computer', Salary: 40000,

Gender: 'Male',

Skills: [ 'MACHINELEARNING', 'JAVA', '.NET' ], DOB: '21/2/1998'

},

{

\_id: 103, Empno: 'E103',

Name: 'Akhil', Designation: 'Accountant', Department: 'Ca',

Salary: 50000, Gender: 'Female',

Skills: [ 'TELLY', 'JAVA', 'PHYTHON' ], DOB: '21/2/2002'

},

{

\_id: 105, Empno: 'E105', Name: 'hit',

Designation: 'Scientist', Department: 'Arynotic', Salary: 100000, Gender: 'Male',

Skills: [ 'C', 'JAVA', 'DSA' ], DOB: '21/2/1990'

}

]

1. **Find all the employees having designation “Engineer” and display salary for them.**

db.EMPLOYEE\_MASTER.find({Designation:"Engineer"},{Salary:1}) [ { \_id: 102, Salary: 40000 } ]

1. **Display only those documents where the name of employee is “Amit” and designation is “Accountant”**

db.EMPLOYEE\_MASTER.find({$or:[{Name:"Amit"},{Designation:"Accountant"}]}) [

{

\_id: 102, Empno: 'E102',

Name: 'Amit', Designation: 'Engineer', Department: 'Computer', Salary: 40000,

Gender: 'Male',

Skills: [ 'MACHINELEARNING', 'JAVA', '.NET' ], DOB: '21/2/1998'

},

{

\_id: 103, Empno: 'E103',

Name: 'Akhil', Designation: 'Accountant', Department: 'Ca',

Salary: 50000, Gender: 'Female',

Skills: [ 'TELLY', 'JAVA', 'PHYTHON' ], DOB: '21/2/2002'

}

]

**Day\_2**

**Essential Assignment:-**

**Create a Student Master database with a collection called “Student” containing**

**documents with some or all of the following fields: StudentRollNo, StudentName,**

**Grade, Hobbies, and DOJ.**

**Perform the following operations on the database:**

db.student.insert({\_id:1,rollno:1,name:"Akhil",grade:"VII",hobby:['chess','dancing'],doj:"2015-02-15"})

db.student.insert({\_id:2,rollno:2,name:"Ravi",grade:"VI",hobby:['Playing','music'],doj:"2011-12-25"})

db.student.insert({\_id:3,rollno:3,name:"jay",grade:"X",hobby:['cricket','tennis'],doj:"2001-05-27"})

db.student.insert({\_id:4,rollno:4,name:"dev",grade:"I",hobby:['chess','adminton'],doj:"2012-11-30"}) db.student.insert({\_id:5,rollno:5,name:"rutvik",grade:"VII",hobby:['foodball','playing'],doj:"2000-12- 31"})

db.student.insert({\_id:6,rollno:6,name:"bhargavi",grade:"II",hobby:['chess','dancing'],doj:"1995-12-05"})

db.student.insert({\_id:7,rollno:7,name:"Akhil",grade:"VI",hobby:['Playing','music'],doj:"2011-12-25"})

db.student.insert({\_id:8,rollno:8,name:"jinal",grade:"XI",hobby:['cricket','tennis'],doj:"2001-03-23"})

db.student.insert({\_id:9,rollno:9,name:"Anil",grade:"VII",hobby:['chess','dancing'],doj:"2015-05-05"}) db.student.insert({\_id:10,rollno:10,name:"Aman",grade:"XI",hobby:['chess','admintion'],doj:"2011-03- 12"})

1. **Insert 10 Records in the database.**

{

"\_id" : 1,

"rollno" : 1,

"name" : "Akhil",

"grade" : "VII", "hobby" : ["chess", "dancing"],

"doj" : "2015-02-15"

}

{

"\_id" : 2,

"rollno" : 2,

"name" : "Ravi",

"grade" : "VI",

"hobby" : ["Playing", "music"],

"doj" : "2011-12-25"

}

{

"\_id" : 3,

"rollno" : 3,

"name" : "jay",

"grade" : "X",

"hobby" : ["cricket", "tennis"],

"doj" : "2001-05-27"

}

{

"\_id" : 4,

"rollno" : 4,

"name" : "dev",

"grade" : "I",

"hobby" : ["chess", "adminton"],

"doj" : "2012-11-30"

}

{

"\_id" : 5,

"rollno" : 5,

"name" : "rutvik",

"grade" : "VII",

"hobby" : ["foodball","playing"],

"doj" : "2000-12-31"

}

{

"\_id" : 6,

"rollno" : 6,

"name" : "bhargavi",

"grade" : "II",

"hobby" : ["chess", "dancing"],

"doj" : "1995-12-05"

}

{

"\_id" : 7,

"rollno" : 7,

"name" : "Mohit",

"grade" : "VI",

"hobby" : ["Playing", "music"],

"doj" : "2011-12-25"

}

{

"\_id" : 8,

"rollno" : 8,

"name" : "jinal",

"grade" : "XI",

"hobby" : ["cricket", "tennis"],

"doj" : "2001-03-23"

}

{

"\_id" : 9,

"rollno" : 9,

"name" : "Anil",

"grade" : "VII",

"hobby" : ["chess", "dancing"],

"doj" : "2015-05-05"

}

{

"\_id" : 10,

"rollno" : 10,

"name" : "Aman",

"grade" : "XI",

"hobby" : ["chess", "admintion"],

"doj" : "2011-03-12"

}

1. **Find the document where in the “StudName” has value “Akhil”. db.student.find({name:"Akhil"})**

{ "\_id" : 1, "rollno" : 1, "name" : "Akhil", "grade" : "VII", "hobby" : [ "chess", "dancing" ], "doj" : "2015-02-15" }

1. **Find all documents in proper (like tabular) format. (Without \_Id field)**

db.student.find({},{\_id:0,rollno:1,name:1,grade:1,hobby:1,doj:1})

{ "rollno" : 1, "name" : "Akhil", "grade" : "VII", "hobby" : [ "chess", "dancing" ], "doj" : "2015-02-15" }

{ "rollno" : 2, "name" : "Ravi", "grade" : "VI", "hobby" : [ "Playing", "music" ], "doj" : "2011-12-25" }

{ "rollno" : 3, "name" : "jay", "grade" : "X", "hobby" : [ "cricket", "tennis" ], "doj" : "2001-05-27" }

{ "rollno" : 4, "name" : "dev", "grade" : "I", "hobby" : [ "chess", "adminton" ], "doj" : "2012-11-30" }

{ "rollno" : 5, "name" : "rutvik", "grade" : "VII", "hobby" : [ "foodball", "playing" ], "doj" : "2000-12-31" }

{ "rollno" : 6, "name" : "bhargavi", "grade" : "II", "hobby" : [ "chess", "dancing" ], "doj" : "1995-12-05" }

{ "rollno" : 7, "name" : "Mohit", "grade" : "VI", "hobby" : [ "Playing", "music" ], "doj" : "2011-12-25" }

{ "rollno" : 8, "name" : "jinal", "grade" : "XI", "hobby" : [ "cricket", "tennis" ], "doj" : "2001-03-23" }

{ "rollno" : 9, "name" : "Anil", "grade" : "VII", "hobby" : [ "chess", "dancing" ], "doj" : "2015-05-05" }

{ "rollno" : 10, "name" : "Aman", "grade" : "XI", "hobby" : [ "chess", "admintion" ], "doj" : "2011-03-12" }

1. **Retrieve only Student Name and Grade.**

db.student.find({},{\_id:0,name:1,grade:1})

{ "name" : "Akhil", "grade" : "VII" }

{ "name" : "Ravi", "grade" : "VI" }

{ "name" : "jay", "grade" : "X" }

{ "name" : "dev", "grade" : "I" }

{ "name" : "rutvik", "grade" : "VII" }

{ "name" : "bhargavi", "grade" : "II" }

{ "name" : "Mohit", "grade" : "VI" }

{ "name" : "jinal", "grade" : "XI" }

{ "name" : "Anil", "grade" : "VII" }

{ "name" : "Aman", "grade" : "XI" }

1. **Retrieve Student Name and Grade of student who is having \_id column is 1.**

db.student.find({\_id:1},{name:1,grade:1})

{ "\_id" : 1, "name" : "Akhil", "grade" : "VII" }

1. **Add new field “Address” in Student collection.**

db.student.updateMany({grade:"VII"},{$set:{Address:"vandematrm"}})

"\_id" : 1,

"rollno" : 1, "name" : "Akhil",

"grade" : "VII", "hobby" : ["chess", "dancing"],

"doj" : "2015-02-15",

"Address" : "vandematrm"

"\_id" : 5,

"rollno" : 5, "name" : "rutvik",

"grade" : "VII", "hobby" : ["foodball", "playing"],

"doj" : "2000-12-31",

"Address" : "vandematrm

1. **Find those documents where the Grade is set to ‘VII’.**

db.student.find({grade:"VII"})

{ "\_id" : 1, "rollno" : 1, "name" : "Akhil", "grade" : "VII", "hobby" : [ "chess", "dancing" ], "doj" : "2015- 02-15", "Address" : "vandematrm" }

{ "\_id" : 5, "rollno" : 5, "name" : "rutvik", "grade" : "VII", "hobby" : [ "foodball", "playing" ], "doj" : "2000- 12-31", "Address" : "vandematrm" }

{ "\_id" : 9, "rollno" : 9, "name" : "Anil", "grade" : "VII", "hobby" : [ "chess", "dancing" ], "doj" : "2015-05- 05", "Address" : "vandematrm" }

1. **Find those documents where the Grade is not set to ‘VII’. not eqval=ne**

db.student.find({grade:{$ne:"VII"}})

{ "\_id" : 2, "rollno" : 2, "name" : "Ravi", "grade" : "VI", "hobby" : [ "Playing", "music" ], "doj" : "2011-12- 25"}

{ "\_id" : 3, "rollno" : 3, "name" : "jay", "grade" : "X", "hobby" : [ "cricket", "tennis" ], "doj" : "2001-05-27"}

{ "\_id" : 4, "rollno" : 4, "name" : "dev", "grade" : "I", "hobby" : [ "chess", "adminton" ], "doj" : "2012-11- 30"}

{ "\_id" : 6, "rollno" : 6, "name" : "bhargavi", "grade" : "II", "hobby" : [ "chess", "dancing" ], "doj" : "1995- 12-05" }

{ "\_id" : 7, "rollno" : 7, "name" : "Akhil", "grade" : "VI", "hobby" : [ "Playing", "music" ], "doj" : "2011-12- 25"}

{ "\_id" : 8, "rollno" : 8, "name" : "jinal", "grade" : "XI", "hobby" : [ "cricket", "tennis" ], "doj" : "2001-03- 23"}

{ "\_id" : 10, "rollno" : 10, "name" : "Aman", "grade" : "XI", "hobby" : [ "chess", "admintion" ], "doj" : "2011-03-12" }

1. **Find those documents where the Hobbies is set to either ‘Chess’ or is set to ‘Dancing”.**

db.student.find({$or:[{hobby:"chess"},{hobby:"dancing"}]})

{ "\_id" : 1, "rollno" : 1, "name" : "Akhil", "grade" : "VII", "hobby" : [ "chess", "dancing" ], "doj" : "2015- 02-15", "Address" : "vandematrm" }

{ "\_id" : 4, "rollno" : 4, "name" : "dev", "grade" : "I", "hobby" : [ "chess", "adminton" ], "doj" : "2012-11- 30" }

{ "\_id" : 6, "rollno" : 6, "name" : "bhargavi", "grade" : "II", "hobby" : [ "chess", "dancing" ], "doj" : "1995- 12-05" }

{ "\_id" : 9, "rollno" : 9, "name" : "Anil", "grade" : "VII", "hobby" : [ "chess", "dancing" ], "doj" : "2015-05- 05", "Address" : "vandematrm" }

{ "\_id" : 10, "rollno" : 10, "name" : "Aman", "grade" : "XI", "hobby" : [ "chess", "admintion" ], "doj" : "2011-03-12" }

1. **Find those documents where the Hobbies is set neither to ‘Chess’ nor is set to ‘Dancing”**

db.student.find({$nor:[{hobby:"chess"},{hobby:"dancing"}]})

{ "\_id" : 2, "rollno" : 2, "name" : "Ravi", "grade" : "VI", "hobby" : [ "Playing", "music" ], "doj" : "2011-12- 25" }

{ "\_id" : 3, "rollno" : 3, "name" : "jay", "grade" : "X", "hobby" : [ "cricket", "tennis" ], "doj" : "2001-05-27"

}

{ "\_id" : 5, "rollno" : 5, "name" : "rutvik", "grade" : "VII", "hobby" : [ "foodball", "playing" ], "doj" : "2000- 12-31", "Address" : "vandematrm" }

{ "\_id" : 7, "rollno" : 7, "name" : "Mohit", "grade" : "VI", "hobby" : [ "Playing", "music" ], "doj" : "2011-12- 25" }

{ "\_id" : 8, "rollno" : 8, "name" : "jinal", "grade" : "XI", "hobby" : [ "cricket", "tennis" ], "doj" : "2001-03- 23" }

**Question 2**

**Create a Movie\_Maker database with a collection called “Movie“ containing**

**documents with some or all of the following fields: titles, directors, years, actors.**

**Perform the following operations on the database:**

db.Movie.insertMany([

{ "\_id" : 1, "title" : "Fight Club", "writer" : "Chuck Palahniuk", "year" : "1999", "actors" : [ "Brad Pitt", "Edward Norton" ] },

{ "\_id" : 2, "title" : "Pulp Fiction", "writer" : "Quentin Tarantino", "year" : "2009", "actors" : [ "John Travolta", "Uma Thurman" ] },

{ "\_id" : 3, "title" : "Inglorious Hero", "writer" : "Quentin Tarantino", "year" : "2009", "actors" : [ "Brad Pitt", "Diane Kruger", "Eli Roth" ] },

{ "\_id" : 4, "title" : "The Hobbit: An unexpected Journey", "writer" : "J.R.R. Tolkein", "year" : "2012", "franchise" : "The Hobbit" },

{ "\_id" : 5, "title" : "The Hobbit: The Desolation of Smaug", "writer" : "J.R.R Tolkien", "year" : "2013", "franchise" : "The Hobbit" },

{ "\_id" : 6, "title" : "The Hobbit: The Battle of the Five Armies", "writer" : "J.R.R Tolkien", "year" : "2002", "franchise" : "The Hobbit", "synopsis" : "Bilbo and Company are forced to engage in a war against an array of combatants and keep the Lonely Mountain from falling into the hands of a rising darkness." },

{ "\_id" : 7, "title" : "Pee Wee Herman's Big Adventures" },

{ "\_id" : 8, "title" : "Avatar" }

])

db.Movie.find().pretty()

{

"\_id" : 1,

"title" : "Fight Club", "writer" : "Chuck Palahniuk", "year" : "1999",

"actors" : [

"Brad Pitt", "Edward Norton"

]

}

{

"\_id" : 2,

"title" : "Pulp Fiction",

"writer" : "Quentin Tarantino", "year" : "2009",

"actors" : [

"John Travolta", "Uma Thurman"

]

}

{

"\_id" : 3,

"title" : "Inglorious Hero", "writer" : "Quentin Tarantino", "year" : "2009",

"actors" : [

"Brad Pitt", "Diane Kruger", "Eli Roth"

]

}

{

"\_id" : 4,

"title" : "The Hobbit: An unexpected Journey", "writer" : "J.R.R. Tolkein",

"year" : "2012",

"franchise" : "The Hobbit"

}

{

"\_id" : 5,

"title" : "The Hobbit: The Desolation of Smaug", "writer" : "J.R.R Tolkien",

"year" : "2013",

"franchise" : "The Hobbit"

}

{

"\_id" : 6,

"title" : "The Hobbit: The Battle of the Five Armies", "writer" : "J.R.R Tolkien",

"year" : "2002",

"franchise" : "The Hobbit",

"synopsis" : "Bilbo and Company are forced to engage in a war against an array of combatants and

keep the Lonely Mountain from falling into the hands of a rising darkness."

}

{ "\_id" : 7, "title" : "Pee Wee Herman's Big Adventures" }

{ "\_id" : 8, "title" : "Avatar" }

1. **Retrieve all documents. db.Movie.find({})**

{ "\_id" : 1, "title" : "Fight Club", "writer" : "Chuck Palahniuk", "year" : "1999", "actors" : [ "Brad Pitt", "Edward Norton" ] }

{ "\_id" : 2, "title" : "Pulp Fiction", "writer" : "Quentin Tarantino", "year" : "2009", "actors" : [ "John Travolta", "Uma Thurman" ] }

{ "\_id" : 3, "title" : "Inglorious Hero", "writer" : "Quentin Tarantino", "year" : "2009", "actors" : [ "Brad Pitt", "Diane Kruger", "Eli Roth" ] }

{ "\_id" : 4, "title" : "The Hobbit: An unexpected Journey", "writer" : "J.R.R. Tolkein", "year" : "2012", "franchise" : "The Hobbit" }

{ "\_id" : 5, "title" : "The Hobbit: The Desolation of Smaug", "writer" : "J.R.R Tolkien", "year" : "2013", "franchise" : "The Hobbit" }

{ "\_id" : 6, "title" : "The Hobbit: The Battle of the Five Armies", "writer" : "J.R.R Tolkien", "year" : "2002", "franchise" : "The Hobbit", "synopsis" : "Bilbo and Company are forced to engage in a war against an array of combatants and keep the Lonely Mountain from falling into the hands of a rising darkness." }

{ "\_id" : 7, "title" : "Pee Wee Herman's Big Adventures" }

{ "\_id" : 8, "title" : "Avatar" }

1. **Retrieve all documents with Director set to "Quentin Tarantino"**.

db.Movie.find({writer:"Quentin Tarantino"})

{ "\_id" : 2, "title" : "Pulp Fiction", "writer" : "Quentin Tarantino", "year" : "2009", "actors" : [ "John Travolta", "Uma Thurman" ] }

{ "\_id" : 3, "title" : "Inglorious Hero", "writer" : "Quentin Tarantino", "year" : "2009", "actors" : [ "Brad Pitt", "Diane Kruger", "Eli Roth" ] }

1. **Retrieve all documents where actors include "Brad Pitt".**

db.Movie.find({actors:"Brad Pitt"})

{ "\_id" : 1, "title" : "Fight Club", "writer" : "Chuck Palahniuk", "year" : "1999", "actors" : [ "Brad Pitt", "Edward Norton" ] }

{ "\_id" : 3, "title" : "Inglorious Hero", "writer" : "Quentin Tarantino", "year" : "2009", "actors" : [ "Brad Pitt", "Diane Kruger", "Eli Roth" ] }

1. **Retrieve all movies released before the year 2000 or after 2010.**

db.Movie.find({ $or:[{year:{$lt:"2000"}},{year:{$gt:"2010"}}]})

{ "\_id" : 1, "title" : "Fight Club", "writer" : "Chuck Palahniuk", "year" : "1999", "actors" : [ "Brad Pitt", "Edward Norton" ] }

{ "\_id" : 4, "title" : "The Hobbit: An unexpected Journey", "writer" : "J.R.R. Tolkein", "year" : "2012", "franchise" : "The Hobbit" }

{ "\_id" : 5, "title" : "The Hobbit: The Desolation of Smaug", "writer" : "J.R.R Tolkien", "year" : "2013", "franchise" : "The Hobbit" }

1. **Add a synopsis to "The Hobbit: An Unexpected Journey”: "A reluctant hobbit, Bilbo Baggins, sets out to the Lonely Mountain with a spirited group of dwarves to reclaim their mountain home - and the gold within it - from the dragon Smaug."**

db.Movie.update({title: "Avatar"},{$set:{synopsis:"The Hobbit An Unexpected Journey A reluctant hobbit,Bilbo Baggins, sets out to the Lonely Mountain with a spirited group of dwarves to reclaim their mountain home and the gold within itfrom the dragon Smaug."}})

{ "\_id" : 1, "title" : "Fight Club", "writer" : "Chuck Palahniuk", "year" : "1999", "actors" : [ "Brad Pitt", "Edward Norton" ] }

{ "\_id" : 2, "title" : "Pulp Fiction", "writer" : "Quentin Tarantino", "year" : "2009", "actors" : [ "John Travolta", "Uma Thurman" ] }

{ "\_id" : 3, "title" : "Inglorious Hero", "writer" : "Quentin Tarantino", "year" : "2009", "actors" : [ "Brad Pitt", "Diane Kruger", "Eli Roth" ] }

{ "\_id" : 4, "title" : "The Hobbit: An unexpected Journey", "writer" : "J.R.R. Tolkein", "year" : "2012", "franchise" : "The Hobbit" }

{ "\_id" : 5, "title" : "The Hobbit: The Desolation of Smaug", "writer" : "J.R.R Tolkien", "year" : "2013", "franchise" : "The Hobbit" }

{ "\_id" : 6, "title" : "The Hobbit: The Battle of the Five Armies", "writer" : "J.R.R Tolkien", "year" : "2002", "franchise" : "The Hobbit", "synopsis" : "Bilbo and Company are forced to engage in a war against an array of combatants and keep the Lonely Mountain from falling into the hands of a rising darkness." }

{ "\_id" : 7, "title" : "Pee Wee Herman's Big Adventures" }

{ "\_id" : 8, "title" : "Avatar", "synopsis" : "The Hobbit An Unexpected Journey A reluctant hobbit,Bilbo Baggins, sets out to the Lonely Mountain with a spirited group of dwarves to reclaim their mountain home and the gold within itfrom the dragon Smaug." }

1. **Add a synopsis to "The Hobbit: The Desolation of Smaug”: "The dwarves, along with Bilbo Baggins and Gandalf the Grey, continue their quest to reclaim Erebor, their homeland, from Smaug. Bilbo Baggins is in possession of a mysterious and magical ring."**

db.Movie.update({title:"The Hobbit: The Desolation of Smaug"},{$set:{synopsis:"The dwarves along with Bilbo Baggins and Gandalf the Grey continue their quest to reclaim Erebortheir homeland from Smaug Bilbo Baggins is in possession of a mysterious and magical ring."}})

{

\_id: 5,

title: 'The Hobbit: The Desolation of Smaug', writer: 'J.R.R Tolkien',

year: '2013',

franchise: 'The Hobbit',

synopsis: 'The dwarves along with Bilbo Baggins and Gandalf the Grey continue their quest to reclaim

Erebortheir homeland from Smaug Bilbo Baggins is in possession of a mysterious and magical ring.'

},

1. **Add an actor named "Samuel L. Jackson" to the movie "Pulp Fiction"**

db.Movie.update({"title":"Pulp Fiction"},{$push:{"actors": "Samuel L. Jackson"}})

{

\_id: 2,

title: 'Pulp Fiction',

writer: 'Quentin Tarantino', year: '2009',

actors: [ 'John Travolta', 'Uma Thurman', 'Samuel L. Jackson' ]

1. **Find all movies that have a synopsis that contains the word "Bilbo".**

db.Movie.find({synopsis:/Bilbo/}) [

{

\_id: 5,

title: 'The Hobbit: The Desolation of Smaug', writer: 'J.R.R Tolkien',

year: '2013',

franchise: 'The Hobbit',

synopsis: 'The dwarves along with Bilbo Baggins and Gandalf the Grey continue their quest to reclaim

Erebortheir homeland from Smaug Bilbo Baggins is in possession of a mysterious and magical ring.'

},

{

\_id: 6,

title: 'The Hobbit: The Battle of the Five Armies', writer: 'J.R.R Tolkien',

year: '2002',

franchise: 'The Hobbit',

synopsis: 'Bilbo and Company are forced to engage in a war against an array of combatants and keep the Lonely Mountain from falling into the hands of a rising darkness.'

},

{

\_id: 8,

title: 'Avatar',

synopsis: 'The Hobbit An Unexpected Journey A reluctant hobbit,Bilbo Baggins, sets out to the Lonely Mountain with a spirited group of dwarves to reclaim their mountain home and the gold within itfrom the dragon Smaug.'

}

]

1. **Find all movies that have a synopsis that contains the word "Gandalf".**

db.Movie.find({synopsis:/Gandalf/})

[

{

\_id: 5,

title: 'The Hobbit: The Desolation of Smaug', writer: 'J.R.R Tolkien',

year: '2013',

franchise: 'The Hobbit',

synopsis: 'The dwarves along with Bilbo Baggins and Gandalf the Grey continue their quest to reclaim

Erebortheir homeland from Smaug Bilbo Baggins is in possession of a mysterious and magical ring.'

}

]

1. **Find all movies that have a synopsis that contains the word "Bilbo" and not the word "Gandalf".**

db.Movie.find({$and:[{synopsis:/Bilbo/},{synopsis:{$not:/Gandalf/}}]}) [

{

\_id: 6,

title: 'The Hobbit: The Battle of the Five Armies', writer: 'J.R.R Tolkien',

year: '2002',

franchise: 'The Hobbit',

synopsis: 'Bilbo and Company are forced to engage in a war against an array of combatants and keep

the Lonely Mountain from falling into the hands of a rising darkness.'

},

{

\_id: 8,

title: 'Avatar',

synopsis: 'The Hobbit An Unexpected Journey A reluctant hobbit,Bilbo Baggins, sets out to the Lonely Mountain with a spirited group of dwarves to reclaim their mountain home and the gold within itfrom the dragon Smaug.'

}

]

1. **Find all movies that have a synopsis that contains the word "dwarves" or "hobbit"**

db.Movie.find({$or:[{synopsis:/dwarves/},{synopsis:/hobbit/}]}) [

{

\_id: 5,

title: 'The Hobbit: The Desolation of Smaug', writer: 'J.R.R Tolkien',

year: '2013',

franchise: 'The Hobbit',

synopsis: 'The dwarves along with Bilbo Baggins and Gandalf the Grey continue their quest to reclaim

Erebortheir homeland from Smaug Bilbo Baggins is in possession of a mysterious and magical ring.'

},

{

\_id: 8,

title: 'Avatar',

synopsis: 'The Hobbit An Unexpected Journey A reluctant hobbit,Bilbo Baggins, sets out to the Lonely Mountain with a spirited group of dwarves to reclaim their mountain home and the gold within itfrom the dragon Smaug.'

}

]

1. **Find all movies that have a synopsis that contains the word "gold" and "dragon".**

db.Movie.find({$and:[{synopsis:/gold/},{synopsis:/dragon/}]}) [

{

\_id: 8,

title: 'Avatar',

synopsis: 'The Hobbit An Unexpected Journey A reluctant hobbit,Bilbo Baggins, sets out to the Lonely

Mountain with a spirited group of dwarves to reclaim their mountain home and the gold within itfrom the dragon Smaug.'

}

]

1. **Delete the movie "Pee Wee Herman's Big Adventure**

db.Movie.deleteOne({"title":"Pee Wee Herman's Big Adventure"})

db.student.insert({\_id:1,rollno:1,name:"Akhil",grade:"VII",hobby:['chess','dancing'],doj:"2015-02-15"})

db.student.insert({\_id:2,rollno:2,name:"Ravi",grade:"VI",hobby:['Playing','music'],doj:"2011-12-25"})

db.student.insert({\_id:3,rollno:3,name:"jay",grade:"X",hobby:['cricket','tennis'],doj:"2001-05-27"})

db.student.insert({\_id:4,rollno:4,name:"dev",grade:"I",hobby:['chess','adminton'],doj:"2012-11-30"}) db.student.insert({\_id:5,rollno:5,name:"rutvik",grade:"VII",hobby:['foodball','playing'],doj:"2000-12- 31"})

db.student.insert({\_id:6,rollno:6,name:"bhargavi",grade:"II",hobby:['chess','dancing'],doj:"1995-12-05"})

db.student.insert({\_id:7,rollno:7,name:"Mohit",grade:"VI",hobby:['Playing','music'],doj:"2011-12-25"})

db.student.insert({\_id:8,rollno:8,name:"jinal",grade:"XI",hobby:['cricket','tennis'],doj:"2001-03-23"})

db.student.insert({\_id:9,rollno:9,name:"Anil",grade:"VII",hobby:['chess','dancing'],doj:"2015-05-05"})

db.student.insert({\_id:10,rollno:10,name:"Aman",grade:"XI",hobby:['chess','admintion'],doj:"2011-03- 12"})

[

{

\_id: 1,

rollno: 1, name: 'Akhil',

grade: 'VII',

hobby: [ 'chess', 'dancing' ], doj: '2015-02-15',

address: 'vandematrm'

},

{

\_id: 2,

rollno: 2, name: 'Ravi',

grade: 'VI',

hobby: [ 'Playing', 'music' ], doj: '2011-12-25',

address: 'vandematrm'

},

{

\_id: 3,

rollno: 3, name: 'jay',

grade: 'X',

hobby: [ 'cricket', 'tennis' ], doj: '2001-05-27',

address: 'vandematrm'

},

{

\_id: 4,

rollno: 4, name: 'dev',

grade: 'I',

hobby: [ 'chess', 'adminton' ], doj: '2012-11-30',

address: 'vandematrm'

},

{

\_id: 5,

rollno: 5, name: 'rutvik', grade: 'VII',

hobby: [ 'foodball', 'playing' ], doj: '2000-12-31',

address: 'vandematrm'

},

{

\_id: 6,

rollno: 6,

name: 'bhargavi', grade: 'II',

hobby: [ 'chess', 'dancing' ], doj: '1995-12-05',

address: 'vandematrm'

},

{

\_id: 7,

rollno: 7, name: 'Mohit',

grade: 'VI',

hobby: [ 'Playing', 'music' ], doj: '2011-12-25',

address: 'vandematrm'

},

{

\_id: 8,

rollno: 8, name: 'jinal',

grade: 'XI',

hobby: [ 'cricket', 'tennis' ], doj: '2001-03-23',

address: 'vandematrm'

},

{

\_id: 9,

rollno: 9, name: 'Anil',

grade: 'VII',

hobby: [ 'chess', 'dancing' ], doj: '2015-05-05',

address: 'vandematrm'

},

{

\_id: 10,

rollno: 10, name: 'Aman',

grade: 'XI',

hobby: [ 'chess', 'admintion' ], doj: '2011-03-12',

address: 'vandematrm'

},

{

\_id: ObjectId('65fad903835292a1c547cccd'),

rollno: 11, name: 'Sanjay', grade: 'XII',

hobby: [ 'playing', 'tennis' ], doj: '2001-03-30'

},

{

\_id: 12, rollno: null,

name: 'Sunial', grade: 'VI',

hobby: [ 'chess', 'dancing' ], doj: '2018-05-12'

},

{

\_id: 13, rollno: null, name: 'Amit',

grade: 'XII',

hobby: [ 'volleybal', 'admintion' ], doj: '2019-06-21'

},

{

\_id: 14, rollno: null,

name: 'Rajesh', grade: 'XIII',

hobby: [ 'cricket', 'tennis' ], doj: '2024-03-26'

}

]

**Day 3**

**Essential Assignment:-**

**Question 1:-**

1. **Find those documents where the student name begins with ‘M’.**

db.student.find({name:/^A/}) [

{

\_id: 1,

rollno: 1, name: 'Akhil',

grade: 'VII',

hobby: [ 'chess', 'dancing' ], doj: '2015-02-15',

address: 'vandematrm'

},

{

\_id: 9,

rollno: 9, name: 'Anil',

grade: 'VII',

hobby: [ 'chess', 'dancing' ], doj: '2015-05-05',

address: 'vandematrm'

},

{

\_id: 10,

rollno: 10, name: 'Aman',

grade: 'XI',

hobby: [ 'chess', 'admintion' ], doj: '2011-03-12',

address: 'vandematrm'

},

{

\_id: 13, rollno: null, name: 'Amit',

grade: 'XII',

hobby: [ 'volleybal', 'admintion' ], doj: '2019-06-21'

}

]

1. **Find those documents where the student name has an “e” in any position.**

db.student.find({name:/e/})

[

{

\_id: 4,

rollno: 4, name: 'dev',

grade: 'I',

hobby: [ 'chess', 'adminton' ], doj: '2012-11-30',

address: 'vandematrm'

},

{

\_id: 14, rollno: null,

name: 'Rajesh', grade: 'XIII',

hobby: [ 'cricket', 'tennis' ], doj: '2024-03-26'

}

]

1. **Find those documents where the student name ends in “a”**.

db.student.find({name:/l$/}) [

{

\_id: 8,

rollno: 8, name: 'jinal',

grade: 'XI',

hobby: [ 'cricket', 'tennis' ], doj: '2001-03-23',

address: 'vandematrm'

},

{

\_id: 9,

rollno: 9, name: 'Anil',

grade: 'VII',

hobby: [ 'chess', 'dancing' ], doj: '2015-05-05',

address: 'vandematrm'

},

{

\_id: 12, rollno: null,

name: 'Sunial', grade: 'VI',

hobby: [ 'chess', 'dancing' ], doj: '2018-05-12'

}

]

1. **Find total number of documents.**

db.student.find({},{\_id:1}).count() 10

1. **Find total number of documents where Grade is ‘VII’.**

db.student.find({grade:'VII'}).count() 3

1. **Sort the documents in ascending order of student name.**

db.student.find().sort({name:1})

[

{

\_id: 1,

rollno: 1, name: 'Akhil',

grade: 'VII',

hobby: [ 'chess', 'dancing' ], doj: '2015-02-15',

Address: 'vandematrm'

},

{

\_id: 10,

rollno: 10, name: 'Aman',

grade: 'XI',

hobby: [ 'chess', 'admintion' ], doj: '2011-03-12',

Address: 'vandematrm'

},

{

\_id: 9,

rollno: 9, name: 'Anil',

grade: 'VII',

hobby: [ 'chess', 'dancing' ], doj: '2015-05-05',

Address: 'vandematrm'

},

{

\_id: 7,

rollno: 7, name: 'Mohit',

grade: 'VI',

hobby: [ 'Playing', 'music' ], doj: '2011-12-25',

Address: 'vandematrm'

},

{

\_id: 2,

rollno: 2, name: 'Ravi',

grade: 'VI',

hobby: [ 'Playing', 'music' ], doj: '2011-12-25',

Address: 'vandematrm'

},

{

\_id: 6,

rollno: 6,

name: 'bhargavi', grade: 'II',

hobby: [ 'chess', 'dancing' ], doj: '1995-12-05',

Address: 'vandematrm'

},

{

\_id: 4,

rollno: 4, name: 'dev',

grade: 'I',

hobby: [ 'chess', 'adminton' ], doj: '2012-11-30',

Address: 'vandematrm'

},

{

\_id: 3,

rollno: 3, name: 'jay',

grade: 'X',

hobby: [ 'cricket', 'tennis' ], doj: '2001-05-27',

Address: 'vandematrm'

},

{

\_id: 8,

rollno: 8, name: 'jinal',

grade: 'XI',

hobby: [ 'cricket', 'tennis' ], doj: '2001-03-23',

Address: 'vandematrm'

},

{

\_id: 5,

rollno: 5, name: 'rutvik', grade: 'VII',

hobby: [ 'foodball', 'playing' ], doj: '2000-12-31',

Address: 'vandematrm'

}

]

1. **Display the last two records**

db.student.find().sort({\_id:-1}).limit(2) [

{

\_id: 10,

rollno: 10, name: 'Aman',

grade: 'XI',

hobby: [ 'chess', 'admintion' ], doj: '2011-03-12',

Address: 'vandematrm'

},

{

\_id: 9,

rollno: 9, name: 'Anil',

grade: 'VII',

hobby: [ 'chess', 'dancing' ], doj: '2015-05-05',

Address: 'vandematrm'

}

]

**Question 2:-**

**Create database EMP and Make Collection With name "EMPL" and Follow Queries Emp> use Empl**

**switched to db Empl**

db.createCollection("EMP")

**Insert Records Into EMP Collection.**

**db.Emp.insertMany([{\_id:1,no:1,name:"ST",salary:2000,role:"OB"},**

**{\_id:2,no:2,name:"MSD",salary:1500,role:"WK"},**

**{\_id:3,no:3,name:"YS",salary:1000,role:"ALR"},**

**{\_id:4,no:4,name:"RD",salary:1000,role:"MOB"},**

**{\_id:5,no:5,name:"RS",salary:500,role:"OB"},**

**{\_id:6,no:6,name:"BK",salary:500,role:"MOB"},**

**{\_id:7,no:7,name:"VK",salary:300,role:"BW"},**

**{\_id:8,no:8,name:"JB",salary:400,role:"BW"},**

**{\_id:9,no:9,name:"HP",salary:400,role:"ALR"},**

**{\_id:10,no:10,name:"VS",salary:300,role:"OB"}])**

1. **Display data in tabular format**.

db.Emp.find({},{\_id:0,name:1,no:1,salary:1,role:1}) [

{ no: 1, name: 'ST', salary: 2000, role: 'OB' },

{ no: 2, name: 'MSD', salary: 1500, role: 'WK' },

{ no: 3, name: 'YS', salary: 1000, role: 'ALR' },

{ no: 4, name: 'RD', salary: 1000, role: 'MOB' },

{ no: 5, name: 'RS', salary: 500, role: 'OB' },

{ no: 6, name: 'BK', salary: 500, role: 'MOB' },

{ no: 7, name: 'VK', salary: 300, role: 'BW' },

{ no: 8, name: 'JB', salary: 400, role: 'BW' },

{ no: 9, name: 'HP', salary: 400, role: 'ALR' },

{ no: 10, name: 'VS', salary: 300, role: 'OB' }

]

1. **Update salary of employee where name is "ST" also increase salary by +8000.**

db.Emp.update({name:"ST"},{$inc:{salary:8000}})

([{\_id:1,no:1,name:"ST",salary:2000,role:"OB"}

{ \_id: 1, no: 1, name: 'ST', salary: 10000, role: 'OB' }

1. **Update salary of all employee by giving an increment of +4000 each employee.**

db.Emp.updateMany({},{$inc:{salary:4000}})

[

{ \_id: 1, no: 1, name: 'ST', salary: 14000, role: 'OB' },

{ \_id: 2, no: 2, name: 'MSD', salary: 5500, role: 'WK' },

{ \_id: 3, no: 3, name: 'YS', salary: 5000, role: 'ALR' },

{ \_id: 4, no: 4, name: 'RD', salary: 5000, role: 'MOB' },

{ \_id: 5, no: 5, name: 'RS', salary: 4500, role: 'OB' },

{ \_id: 6, no: 6, name: 'BK', salary: 4500, role: 'MOB' },

{ \_id: 7, no: 7, name: 'VK', salary: 4300, role: 'BW' },

{ \_id: 8, no: 8, name: 'JB', salary: 4400, role: 'BW' },

{ \_id: 9, no: 9, name: 'HP', salary: 4400, role: 'ALR' },

{ \_id: 10, no: 10, name: 'VS', salary: 4300, role: 'OB' }

]

1. **Update role of employee whose name is "MSD" as "C and WK"**

db.Emp.update({name:"MSD"},{$set:{role:["C","WK"]}})

{ \_id: 2, no: 2, name: 'MSD', salary: 5500, role: [ 'C', 'WK' ] },

1. **Add a new field remark to document with name "RS" set remark as WC**

db.Emp.update({name:"RS"},{$set:{remark:"WC"}})

{ \_id: 5, no: 5, name: 'RS', salary: 4500, role: 'OB', remark: 'WC' },

1. **Add a new field as number “11”, name “AK”, Salary “10000”, role “coach” without using insert statement, but for doing so you should have a record added number 11.**

db.Emp.update({"\_id":11},{$set:{"\_id":11,"no": 11, "name": 'AK', "salary": 10000, "role": 'coach'}},{upsert:true})

{ \_id: 11, name: 'AK', no: 11, role: 'coach', salary: 10000 }

1. **Remove added new field(number “11”).**

db.Emp.deleteOne({\_id:11})

1. **Update the field "RD" by multiplying with salary by 2.**

db.Emp.update({name:"RD"},{$mul:{salary:2}})

{ \_id: 4, no: 4, name: 'RD', salary: 10000, role: 'MOB' },

1. **Find document from the empl collection where name begins with ‘S’**

db.Emp.find({name:/^S/})

[ { \_id: 1, no: 1, name: 'ST', salary: 14000, role: 'OB' } ]

1. **Find document from the empl collection where name begins with ‘R’**

db.Emp.find({name:/^R/}) [

{ \_id: 4, no: 4, name: 'RD', salary: 10000, role: 'MOB' },

{ \_id: 5, no: 5, name: 'RS', salary: 4500, role: 'OB', remark: 'WC' }

]

1. **Find document from the empl collection where name ends with ‘K’ db.Emp.find({name:/K$/})**

[

{ \_id: 6, no: 6, name: 'BK', salary: 4500, role: 'MOB' },

{ \_id: 7, no: 7, name: 'VK', salary: 4300, role: 'BW' },

{ \_id: 11, name: 'AK', no: 11, role: 'coach', salary: 10000 }

]

1. **Find document from the empl collection where name ends with ‘D’ db.Emp.find({name:/D$/})**

db.EMP.find({name:{$regex:/D$/}})

[

{ \_id: 2, no: 2, name: 'MSD', salary: 5500, role: [ 'C', 'WK' ] },

{ \_id: 4, no: 4, name: 'RD', salary: 10000, role: 'MOB' }

]

1. **Find document from the empl collection where name has S in any position db.Emp.find({name:/S/})**

db.Emp.find({name:{$regex:/S/}}) [

{ \_id: 1, no: 1, name: 'ST', salary: 14000, role: 'OB' },

{ \_id: 2, no: 2, name: 'MSD', salary: 5500, role: [ 'C', 'WK' ] },

{ \_id: 3, no: 3, name: 'YS', salary: 5000, role: 'ALR' },

{ \_id: 5, no: 5, name: 'RS', salary: 4500, role: 'OB', remark: 'WC' },

{ \_id: 10, no: 10, name: 'VS', salary: 4300, role: 'OB' }

]

**Regular Expression (Note: Use Case sensitive allow For that write in Option: "i")**

1. **Find document from the empl collection where name begins with ‘S’**

db.Emp.find({name:{$regex:/^s/,$options:"i"}})

[ { \_id: 1, no: 1, name: 'ST', salary: 14000, role: 'OB' } ]

1. **Find document from the empl collection where name begins with ‘S’**

db.Emp.find({name:{$regex:/^S/,$options:"i"}})

[ { \_id: 1, no: 1, name: 'ST', salary: 14000, role: 'OB' } ]

**Use of $in and $nin (in and notin)**

**(Note: There will not use {} braces in that $in and $nin)**

1. **Display documents where in empl collection field have OB,MOB**

db.Emp.find({role:{$in:["OB","MOB"]}})

[

{ \_id: 1, no: 1, name: 'ST', salary: 14000, role: 'OB' },

{ \_id: 4, no: 4, name: 'RD', salary: 10000, role: 'MOB' },

{ \_id: 5, no: 5, name: 'RS', salary: 4500, role: 'OB', remark: 'WC' },

{ \_id: 6, no: 6, name: 'BK', salary: 4500, role: 'MOB' },

{ \_id: 10, no: 10, name: 'VS', salary: 4300, role: 'OB' }

]

1. **Display documents where in empl collection field not have OB,MO**

db.Emp.find({role:{$nin:["OB","MOB"]}})

[

{ \_id: 2, no: 2, name: 'MSD', salary: 5500, role: [ 'C', 'WK' ] },

{ \_id: 3, no: 3, name: 'YS', salary: 5000, role: 'ALR' },

{ \_id: 7, no: 7, name: 'VK', salary: 4300, role: 'BW' },

{ \_id: 8, no: 8, name: 'JB', salary: 4400, role: 'BW' },

{ \_id: 9, no: 9, name: 'HP', salary: 4400, role: 'ALR' },

{ \_id: 11, name: 'AK', no: 11, role: 'coach', salary: 10000 }

]

**Day - 4**

**Create a database named “Store” in MongoDB with a collection called “Sales”**

**containing documents with some or all of the following fields:**

**customerId, customerName, gender, DOB, contactNumber,**

**address (containing fields: houseNo, street, area, city, pincode),**

**orders (containing fields: orderId, orderDate, items (containing fields: itemId,**

**itemName, itemPrice, quantityOrdered, discount)).**

use inventory db.createCollection("sales")

db.sales.insert([{\_id:1,proid:111,proname:"tv",saledate:"2021-06- 11",saleprice:12000,salequa:65,purdate:"2022-07-12",purprice:500,purqua:200},

{\_id:2,proid:221,proname:"ac",saledate:"2021-04-11",saleprice:30000,salequa:100,purdate:"2021-07- 10",purprice:20000,purqua:20},

{\_id:3,proid:555,proname:"lg",saledate:"2021-02-11",saleprice:50000,salequa:200,purdate:"2022-08- 12",purprice:50000,purqua:350},

{\_id:4,proid:862,proname:"sony",saledate:"2021-03-11",saleprice:10000,salequa:276,purdate:"2020-07- 03",purprice:34400,purqua:100},

{\_id:5,proid:550,proname:"iphone",saledate:"2020-09-11",saleprice:90000,salequa:30,purdate:"2022- 08-12",purprice:40000,purqua:150},

{\_id:6,proid:940,proname:"lava",saledate:"2002-03-11",saleprice:80000,salequa:100,purdate:"2021-07- 12",purprice:45200,purqua:200},

{\_id:7,proid:666,proname:"oppo",saledate:"2021-04-11",saleprice:8000,salequa:300,purdate:"2021-04- 10",purprice:50000,purqua:20},

{\_id:8,proid:333,proname:"charger",saledate:"2003-06-11",saleprice:90000,salequa:67,purdate:"2021- 07-12",purprice:50000,purqua:10},

{\_id:9,proid:552,proname:"mobile",saledate:"2021-03-10",saleprice:30000,salequa:400,purdate:"2021- 08-12",purprice:78000,purqua:230},

{\_id:10,proid:444,proname:"oppo",saledate:"2021-04-01",saleprice:10000,salequa:450,purdate:"2002- 07-12",purprice:5000,purqua:100}])

1. **Display only the productName, where salePrice is less than 150**

db.sales.find({saleprice:{$lt:20000}},{proname:1})

{ "\_id" : 4, "proname" : "sony" }

{ "\_id" : 7, "proname" : "oppo" }

{ "\_id" : 10, "proname" : "oppo" }

1. **Change the product name of TV to Television.**

db.sales.update({proname:"tv"},{proname:"television"})

db.sales.find()

{ "\_id" : 1, "proname" : "television" }

1. **Add a field productsize (an array) with values “small”, “middle” and “big” for document with productId “Pid0023” and “Pid0231**

db.sales.updateMany({proid:{$in:[221,555]}},{$set:{prosize:["small","middle","big"]}})

db.sales.updateMany({proid:{$in:[221,555]}},{$set:{prosize:["small","middle","big"]}})

{ "acknowledged" : true, "matchedCount" : 2, "modifiedCount" : 2 }

1. **Group on productId and compute the maximum of salePrice.**

db.sales.aggregate({$group:{\_id:"$proid",max\_saleprice:{$max:"$saleprice"}}}) db.sales.aggregate({$group:{\_id:"$proid",max\_saleprice:{$max:"$saleprice"}}})

{ "\_id" : null, "max\_saleprice" : null }

{ "\_id" : 862, "max\_saleprice" : 10000 }

{ "\_id" : 940, "max\_saleprice" : 80000 }

{ "\_id" : 555, "max\_saleprice" : 50000 }

{ "\_id" : 333, "max\_saleprice" : 90000 }

{ "\_id" : 666, "max\_saleprice" : 8000 }

{ "\_id" : 444, "max\_saleprice" : 10000 }

{ "\_id" : 550, "max\_saleprice" : 90000 }

{ "\_id" : 552, "max\_saleprice" : 30000 }

{ "\_id" : 221, "max\_saleprice" : 30000 }

1. **Display the productName and saleDate, whose saleQuantity is greater than 100 and less than 165.**

db.sales.find({$and:[{salequa:{$gt:165}},{salequa:{$lt:400}}]},{proname:1},{saledate:1})

{ "\_id" : 3, "proname" : "lg" }

{ "\_id" : 4, "proname" : "sony" }

{ "\_id" : 7, "proname" : "oppo" }

1. **Add a field called “Descriptor”, which describes the product for documents with Model No, is “Pid0044”, “Pid0231” and “Pid3211**

db.sales.updateMany({proid:{$in:[333,111]}},{$set:{describes:"healthy fruit"}})

1. **Search the documents where the Descriptor field contains the word “healthy” and “fruit”.**

db.sales.find({describes:/healthy fruit/})

{ "\_id" : 8, "proid" : 333, "proname" : "charger", "saledate" : "2003-06-11", "saleprice" : 90000, "salequa" : 67, "purdate" : "2021-07-12", "purprice" : 50000, "purqua" : 10, "describes" : "healthy fruit" }

=====================================================================================

1. **Create Index on productID.**

db.sales.createIndex({proid:1}) db.sales.getIndexes()

[

{

"v" : 2,

"key" : {

"\_id" : 1

},

"name" : "\_id\_",

"ns" : "inventory.sales"

},

{

"v" : 2,

"key" : {

"proid" : 1

},

"name" : "proid\_1", "ns" : "inventory.sales"

}

]

1. **Remove field salesDate where productID is “Pid0555**

db.sales.remove({proid:940})

]

db.sales.remove({proid:940}) WriteResult({ "nRemoved" : 1 })

**Question 2**

**Create a database named “BookStore” in MongoDB with a collection called**

**“Books” containing documents with some or all of the following fields: bookId,**

**bookTitle, authors (containing fields: authorName), publicationYear, publisher,**

**Orders (containing fields: OrderedId, orderDate, customerName, price,**

**quantityOrdered, discount).**

**Note that a book may have one or more authors and orders. Also, the same Ordered**

**can be present in one or more books.**

**Perform the following operations on the**

**database (either in the console or using any programming language):**

db.Books.insert([

{\_id:1,bookId:"b101",bookTitle:"The Secret 1",authors:["Rhonda Byrne"],publicationYear:2006,publisher:"Atria Publishing Group",

orders:[{OrderedId:"o101", orderDate:new Date("2020-02-11"), customerName:"Jainam", price:1000, quantityOrdered:1, discount:100},{OrderedId:"o102", orderDate:new Date("2020-02-12"), customerName:"Rahil", price:1000,

quantityOrdered:2, discount:50},{OrderedId:"o103", orderDate:new Date("2020-02-13"), customerName:"Gautam", price:1000,

quantityOrdered:2, discount:150},{OrderedId:"o104", orderDate:new Date("2020-02-14"), customerName:"Darshan", price:1000,

quantityOrdered:1, discount:100}]},

{\_id:2,bookId:"b102",bookTitle:"The Secret 2",authors:["Rhonda Byrne","Bob Proctor"],publicationYear:2006,publisher:"Atria Publishing Group",

orders:[{OrderedId:"o101", orderDate:new Date("2020-02-11"), customerName:"Jainam", price:1000, quantityOrdered:1, discount:100},{OrderedId:"o102", orderDate:new Date("2020-02-12"), customerName:"Rahil", price:1000,

quantityOrdered:2, discount:50},{OrderedId:"o103", orderDate:new Date("2020-02-13"), customerName:"Gautam", price:1000,

quantityOrdered:2, discount:150}]},

{\_id:3,bookId:"b103",bookTitle:"The Secret 3",authors:["Rhonda Byrne","Esther Hicks"],publicationYear:2006,publisher:"Atria Publishing Group",

orders:[{OrderedId:"o101", orderDate:new Date("2020-02-11"), customerName:"Jainam", price:1000, quantityOrdered:1, discount:100},{OrderedId:"o102", orderDate:new Date("2020-02-12"), customerName:"Rahil", price:1000,

quantityOrdered:2, discount:50},{OrderedId:"o103", orderDate:new Date("2020-02-13"), customerName:"Gautam", price:1000,

quantityOrdered:2, discount:150},{OrderedId:"o104", orderDate:new Date("2020-02-14"), customerName:"Darshan", price:1000,

quantityOrdered:1, discount:100}]},

{\_id:4,bookId:"b104",bookTitle:"The Secret 4",authors:["Rhonda Byrne","Bob Proctor"],publicationYear:2006,publisher:"Beyond Words Publishing",

orders:[{OrderedId:"o101", orderDate:new Date("2020-02-11"), customerName:"Jainam", price:1000, quantityOrdered:1, discount:100},{OrderedId:"o102", orderDate:new Date("2020-02-12"), customerName:"Rahil", price:1000,

quantityOrdered:2, discount:50},{OrderedId:"o103", orderDate:new Date("2020-02-13"), customerName:"Gautam", price:1000,

quantityOrdered:2, discount:150}]},

{\_id:5,bookId:"b105",bookTitle:"The Secret 5",authors:["Rhonda Byrne"],publicationYear:2006,publisher:"Atria Publishing Group",

orders:[{OrderedId:"o101", orderDate:new Date("2020-02-11"), customerName:"Jainam", price:1000, quantityOrdered:1, discount:100},{OrderedId:"o102", orderDate:new Date("2020-02-12"), customerName:"Rahil", price:1000,

quantityOrdered:2, discount:50}]},

{\_id:6,bookId:"b106",bookTitle:"The Secret 6",authors:["Rhonda Byrne","EstherHicks","Esther Hicks"],publicationYear:2006,publisher:"Beyond Words Publishing",

orders:[{OrderedId:"o101", orderDate:new Date("2020-02-11"), customerName:"Jainam", price:1000, quantityOrdered:1, discount:100},{OrderedId:"o102", orderDate:new Date("2020-02-12"), customerName:"Rahil", price:1000,

quantityOrdered:2, discount:50},{OrderedId:"o103", orderDate:new Date("2020-02-13"), customerName:"Gautam", price:1000,

quantityOrdered:2, discount:150}]},

{\_id:7,bookId:"b107",bookTitle:"The Secret 7",authors:["Rhonda Byrne","BobProctor","Esther Hicks"],publicationYear:2006,publisher:"Atria Publishing Group",

orders:[{OrderedId:"o103", orderDate:new Date("2020-02-13"), customerName:"Gautam", price:1000, quantityOrdered:2, discount:150},{OrderedId:"o104", orderDate:new Date("2020-02-14"), customerName:"Darshan", price:1000,

quantityOrdered:1, discount:100}]},

{\_id:8,bookId:"b108",bookTitle:"The Secret 8",authors:["Rhonda Byrne"],publicationYear:2006,publisher:"Beyond Words Publishing",

orders:[{OrderedId:"o101", orderDate:new Date("2020-02-11"), customerName:"Jainam", price:1000, quantityOrdered:1, discount:100},{OrderedId:"o102", orderDate:new Date("2020-02-12"), customerName:"Rahil", price:1000,

quantityOrdered:2, discount:50},{OrderedId:"o103", orderDate:new Date("2020-02-13"), customerName:"Gautam", price:1000,

quantityOrdered:2, discount:150}]},

{\_id:9,bookId:"b109",bookTitle:"The Secret 9",authors:["Rhonda Byrne","Esther Hicks"],publicationYear:2006,publisher:"Atria Publishing Group",

orders:[{OrderedId:"o101", orderDate:new Date("2020-02-11"), customerName:"Jainam", price:1000, quantityOrdered:1, discount:100},{OrderedId:"o102", orderDate:new Date("2020-02-12"), customerName:"Rahil", price:1000,

quantityOrdered:2, discount:50},{OrderedId:"o104", orderDate:new Date("2020-02-14"), customerName:"Darshan", price:1000,

quantityOrdered:1, discount:100}]},

{\_id:10,bookId:"b110",bookTitle:"The Secret 10",authors:["Rhonda Byrne","BobProctor","Esther Hicks"],publicationYear:2006,publisher:"Beyond Words Publishing",

orders:[{OrderedId:"o102", orderDate:new Date("2020-02-12"), customerName:"Rahil", price:1000, quantityOrdered:2, discount:50},{OrderedId:"o103", orderDate:new Date("2020-02-13"), customerName:"Gautam", price:1000,

quantityOrdered:2, discount:150},{OrderedId:"o104", orderDate:new Date("2020-02-14"), customerName:"Darshan", price:1000,

quantityOrdered:1, discount:100}]},

])

1. **Insert records for 10 books from 5 authors, and at least 20 orders in total.**
2. **Update the title of a particular book.**

{\_id:10,bookId:"b110",bookTitle:"The Secret 10"

db.Books.update({\_id:10},{$set:{bookTitle:"THE SECRECT 10"}},{upsert:true})

{ "\_id" : 10, "bookId" : "b110", "bookTitle" : "THE SECRECT 10",

1. **Display all the books having less than 3 authors and sort by book name.**

db.Books.aggregate({$project:{no\_authors:{$size:"$authors"}}})

{ "\_id" : 1, "no\_of\_authors" : 1 }

{ "\_id" : 2, "no\_of\_authors" : 2 }

{ "\_id" : 3, "no\_of\_authors" : 2 }

{ "\_id" : 4, "no\_of\_authors" : 2 }

{ "\_id" : 5, "no\_of\_authors" : 1 }

{ "\_id" : 6, "no\_of\_authors" : 3 }

{ "\_id" : 7, "no\_of\_authors" : 3 }

{ "\_id" : 8, "no\_of\_authors" : 1 }

{ "\_id" : 9, "no\_of\_authors" : 2 }

{ "\_id" : 10, "no\_of\_authors" : 3 }

db.Books.aggregate({$project:{no\_authors:{$size:"$authors"},bookTitle:1}})

{ "\_id" : 1, "bookTitle" : "The Secret 1", "no\_of\_authors" : 1 }

{ "\_id" : 2, "bookTitle" : "The Secret 2", "no\_of\_authors" : 2 }

{ "\_id" : 3, "bookTitle" : "The Secret 3", "no\_of\_authors" : 2 }

{ "\_id" : 4, "bookTitle" : "The Secret 4", "no\_of\_authors" : 2 }

{ "\_id" : 5, "bookTitle" : "The Secret 5", "no\_of\_authors" : 1 }

{ "\_id" : 6, "bookTitle" : "The Secret 6", "no\_of\_authors" : 3 }

{ "\_id" : 7, "bookTitle" : "The Secret 7", "no\_of\_authors" : 3 }

{ "\_id" : 8, "bookTitle" : "The Secret 8", "no\_of\_authors" : 1 }

{ "\_id" : 9, "bookTitle" : "The Secret 9", "no\_of\_authors" : 2 }

{ "\_id" : 10, "bookTitle" : "THE SECRECT 10", "no\_of\_authors" : 3 }

db.Books.aggregate({$project:{no\_authors:{$size:"$authors"},bookTitle:1}},{$out:"no\_authors"})

{ "\_id" : 1, "bookTitle" : "The Secret 1", "no\_of\_authors" : 1 }

{ "\_id" : 2, "bookTitle" : "The Secret 2", "no\_of\_authors" : 2 }

{ "\_id" : 3, "bookTitle" : "The Secret 3", "no\_of\_authors" : 2 }

{ "\_id" : 4, "bookTitle" : "The Secret 4", "no\_of\_authors" : 2 }

{ "\_id" : 5, "bookTitle" : "The Secret 5", "no\_of\_authors" : 1 }

{ "\_id" : 6, "bookTitle" : "The Secret 6", "no\_of\_authors" : 3 }

{ "\_id" : 7, "bookTitle" : "The Secret 7", "no\_of\_authors" : 3 }

{ "\_id" : 8, "bookTitle" : "The Secret 8", "no\_of\_authors" : 1 }

{ "\_id" : 9, "bookTitle" : "The Secret 9", "no\_of\_authors" : 2 }

{ "\_id" : 10, "bookTitle" : "THE SECRECT 10", "no\_of\_authors" : 3 }

db.no\_authors.find({no\_authors:{$lt:3}}).

{ "\_id" : 1, "bookTitle" : "The Secret 1", "no\_authors" : 1 }

{ "\_id" : 2, "bookTitle" : "The Secret 2", "no\_authors" : 2 }

{ "\_id" : 3, "bookTitle" : "The Secret 3", "no\_authors" : 2 }

{ "\_id" : 4, "bookTitle" : "The Secret 4", "no\_authors" : 2 }

{ "\_id" : 5, "bookTitle" : "The Secret 5", "no\_authors" : 1 }

{ "\_id" : 8, "bookTitle" : "The Secret 8", "no\_authors" : 1 }

{ "\_id" : 9, "bookTitle" : "The Secret 9", "no\_authors" : 2 }

db.no\_authors.find({no\_authors:{$lt:3}}).sort({bookTitle:1})

{ "\_id" : 1, "bookTitle" : "The Secret 1", "no\_authors" : 1 }

{ "\_id" : 2, "bookTitle" : "The Secret 2", "no\_authors" : 2 }

{ "\_id" : 3, "bookTitle" : "The Secret 3", "no\_authors" : 2 }

{ "\_id" : 4, "bookTitle" : "The Secret 4", "no\_authors" : 2 }

{ "\_id" : 5, "bookTitle" : "The Secret 5", "no\_authors" : 1 }

{ "\_id" : 8, "bookTitle" : "The Secret 8", "no\_authors" : 1 }

{ "\_id" : 9, "bookTitle" : "The Secret 9", "no\_authors" : 2 }

========================================================

1. Display the number of books from each publisher

db.Books.aggregate({$group:{\_id:"$publisher",total:{$sum:1}}})

{ "\_id" : "Atria Publishing Group", "total" : 6 }

{ "\_id" : "Beyond Words Publishing", "total" : 4 }

**Desirable**

**Create “Mymenu” database with a collection called “Restaurants”, containing**

**documents with some or all of the following fields: Restaurant Id, Restaurant**

**Name, Grades (Note: An array is expected), Cuisine, Address (Note: Must**

**include Building Name, Street, Area, City, ZipCode), and Date of**

**Establishment (Note: Use Proper Date format), Score and Rating. Perform the**

**following operations on the database. (Insert at least 10 documents)**

use MYmenu db.createCollection("Restaurants")

db.Restaurants.insert({\_id:1, RestaurantId:1, RestaurantsName:"Mirch Masala", Grades:["A","A++"],

Cuisine:"Kitchen king", Address:[{BuildgName:"Himalaya",Street:"vastrapur",Area:"vasrapur Lake",City:"Ahmedabad",ZipCode:"380025"}], DateofEstablishment:"22-Mar-2021",

Score:90, Rating:4.9

})

db.Restaurants.insert({\_id:2, RestaurantId:2, RestaurantsName:"Sabar", Grades:["A","B++"],

Cuisine:"lilivadi", Address:[{BuildgName:"yariyann",Street:"nikol",Area:"Nikol",City:"Ahmedabad",ZipCode:"3800256"}], DateofEstablishment:Date(),

Score:30, Rating:5.6

})

db.Restaurants.insert({\_id:3, RestaurantId:3, RestaurantsName:"Barbeque Nation", Grades:["A","B"],

Cuisine:"American", Address:[{BuildgName:"sonalika",Street:"iskon",Area:"ISKON",City:"Ahmedabad",ZipCode:"380024"}], DateofEstablishment:"6-january-2007",

Score:40, Rating:4.6

})

db.Restaurants.insert({\_id:4, RestaurantId:4, RestaurantsName:"Honest", Grades:["C","B++"],

Cuisine:"Chiness", Address:[{BuildgName:"ETC",Street:"Jayapprtment",Area:"bhavnager",City:"Bhavnager",ZipCode:"3870 256"}],

DateofEstablishment:"22-4-2009", Score:60,

Rating:7.9

})

db.Restaurants.insert({\_id:5, RestaurantId:5, RestaurantsName:"jenil", Grades:["A++","B++"],

Cuisine:"Chiness", Address:[{BuildgName:"Vatica",Street:"umiyapark",Area:"rajmahel",City:"rajkot",ZipCode:"3870256"}], DateofEstablishment:"26-4-2024",

Score:60, Rating:5.7

})

1. **Find the Restaurant Names, who have established after January 2010.**

db.Restaurants.find({DateofEstablishment:{$gt:"2010"}},{RestaurantsName:1})

db.Restaurants.find({DateofEstablishment:{$gt:'1-1-2010'}},{RestaurantsName:1})

{ "\_id" : 1, "RestaurantsName" : "Mirch Masala" }

{ "\_id" : 2, "RestaurantsName" : "Barbeque Nation" }

{ "\_id" : 3, "RestaurantsName" : "Sabar" }

1. **Find the restaurants that do not prepare Cuisine of “American”, and their Score is more than 70.**

db.Restaurants.find({$and:[{Cuisine:{$nin:["American"]}},{Score:{$gt:70}}]})

{ "\_id" : 1, "RestaurantId" : 1, "RestaurantsName" : "Mirch Masala", "Grades" : [ "A", "A++" ], "Cuisine" : "Kitchen king", "Address" : [ { "BuildgName" : "Himalaya", "Street" : "vastrapur", "Area" : "vasrapur Lake", "City" : "Ahmedabad", "ZipCode" : "380025" } ], "DateofEstablishment" : "22-Mar-2021", "Score" : 90, "Rating" : 4.9}

1. **Update the Rating of the restaurant “Mirch Masala”**.

db.Restaurants.update({RestaurantsName:"Mirch Masala"},{$set:{Rating:6}})

{ "\_id" : 1, "RestaurantId" : 1, "RestaurantsName" : "Mirch Masala", "Grades" : [ "A", "A++" ], "Cuisine" : "Kitchen king", "Address" : [ { "BuildgName" : "Himalaya", "Street" : "vastrapur", "Area" : "vasrapur Lake", "City" : "Ahmedabad", "ZipCode" : "380025" } ], "DateofEstablishment" : "22-Mar-2021", "Score" : 90, "Rating" : 6 }

1. **Display the restaurants, which are located in “Ahmedabad”.**

db.Restaurants.find({"Address.City":"Ahmedabad"},{RestaurantsName:1})

{ "\_id" : 1, "RestaurantsName" : "Mirch Masala" }

{ "\_id" : 2, "RestaurantsName" : "Barbeque Nation" }

db.Restaurants.find({"Address.City":"Ahmedabad"})

1. **Find the Restaurant Names and Cuisine, for those restaurants which contain ‘chen’ (Example: “Kitchen”) as the last three letters.**

db.Restaurants.find({Cuisine:/chen/})

{ "\_id" : 1, "RestaurantId" : 1, "RestaurantsName" : "Mirch Masala", "Grades" : [ "A", "A++" ], "Cuisine" : "Kitchen king", "Address" : [ { "BuildgName" : "Himalaya", "Street" : "vastrapur", "Area" : "vasrapur Lake", "City" : "Ahmedabad", "ZipCode" : "380025" } ], "DateofEstablishment" : "22-Mar-2021", "Score" : 90, "Rating" : 6 }

1. **Find the Restaurant Id’s and Restaurant Names of those restaurants, which are situated in “Ahmedabad” (City) but not in “ISKON” (Area).**

Address:[{BuildgName:"ETC",Street:"Jayapprtment",Area:"bhavnager"

db.Restaurants.find({

$and:[{"Address.City":{$in:["Ahmedabad"]}},{"Address.Area":{$nin:["ISKON"]}}]},{RestaurantId:1,Restaur antsName:1})

[

{ \_id: 1, RestaurantId: 1, RestaurantsName: 'Mirch Masala' },

{ \_id: 2, RestaurantId: 2, RestaurantsName: 'Sabar' }

]

1. **Add a field “Borough” with value “Bronx”, for restaurants with \_id: 3 and 4.**

db.Restaurants.updateMany({RestaurantId:{$in:[1,3]}},{$push:{"Borough":"Bronx"}})

{

\_id: 1,

RestaurantId: 1, RestaurantsName: 'Mirch Masala', Grades: [ 'A', 'A++' ],

Cuisine: 'Kitchen king', Address: [

{

BuildgName: 'Himalaya', Street: 'vastrapur',

Area: 'vasrapur Lake', City: 'Ahmedabad', ZipCode: '380025'

}

],

DateofEstablishment: '22-Mar-2021', Score: 90,

Rating: 6,

Borough: [ 'Bronx' ]

1. **Remove the field Cuisine for restaurants whose name is “Jassi De Paratha”.**

db.Restaurants.update({RestaurantsName:"Honest"},{$unset:{Cuisine:""}})

{

\_id: 4,

RestaurantId: 4, RestaurantsName: 'Honest', Grades: [ 'C', 'B++' ],

Address: [

{

BuildgName: 'ETC', Street: 'Jayapprtment', Area: 'bhavnager', City: 'Bhavnager', ZipCode: '3870256'

}

],

1. **Remove the document, with restaurant named “Barbeque Nation”**

db.Restaurants.deleteOne({RestaurantsName:"jenil"})

{ "\_id" : 1, "RestaurantId" : 1, "RestaurantsName" : "Mirch Masala", "Grades" : [ "A", "A++" ], "Cuisine" : "Kitchen king", "Address" : [ { "BuildgName" : "Himalaya", "Street" : "vastrapur", "Area" : "vasrapur Lake", "City" : "Ahmedabad", "ZipCode" : "380025" } ], "DateofEstablishment" : "22-Mar-2021", "Score" : 90, "Rating" : 4.9 }

{ "\_id" : 3, "RestaurantId" : 3, "RestaurantsName" : "Sabar", "Grades" : [ "A", "B++" ], "Cuisine" :

"lilivadi", "Address" : [ { "BuildgName" : "yariyann", "Street" : "nikol", "Area" : "Nikol", "City" : "Ahmedabade", "ZipCode" : "3800256" } ], "DateofEstablishment" : "Sat Mar 23 2024 11:46:17 GMT+0530 (India Standard Time)", "Score" : 30, "Rating" : 5.6 }

{ "\_id" : 2, "RestaurantId" : 1, "RestaurantsName" : "Barbeque Nation", "Grades" : [ "A", "B" ], "Cuisine" :

"American", "Address" : [ { "BuildgName" : "sonalika", "Street" : "iskon", "Area" : "ISKON", "City" : "Ahmedabad", "ZipCode" : "380024" } ], "DateofEstablishment" : "6-january-2007", "Score" : 40, "Rating" : 4.6 }

{ "\_id" : 4, "RestaurantId" : 5, "RestaurantsName" : "Honest", "Grades" : [ "C", "B++" ], "Cuisine" :

"Chiness", "Address" : [ { "BuildgName" : "ETC", "Street" : "Jayapprtment", "Area" : "bhavnager", "City" : "Bhavnager", "ZipCode" : "3870256" } ], "DateofEstablishment" : "22-4-2009", "Score" : 60, "Rating" : 7.9

}

**Day-5**

**Question-2**

**Find Out total number of posts by each user whose status is active. use Blogs**

db.createCollection("Blogs") db.Blogs.insertMany([

{\_id:1,userId:"u101",post\_text:"Python MapReduce", status:"Active"},

{\_id:2,userId:"u101",post\_text:"Python Socket", status:"Active"},

{\_id:3,userId:"u102",post\_text:"Java Script", status:"Active"},

{\_id:4,userId:"u102",post\_text:"Css&bootstrap", status:"Active"},

{\_id:5,userId:"u101",post\_text:"Python ADT", status:"Passive"},

])

var map=function(){emit(this.userId,1);}

var reduce =function(userId,count){return Array.sum(count)}

db.Blogs.mapReduce(map,reduce,{out:"total\_post"}); db.total\_post.find()

{ "\_id" : "u101", "value" : 3 }

{ "\_id" : "u102", "value" : 2 }

db.Blogs.mapReduce(map,reduce,{out:"total\_post",query:{status:"Active"}})

{ "\_id" : "u101", "value" : 2 }

{ "\_id" : "u102", "value" : 2 }

**Question-3**

**db.Customers.insert({CustID:"C123", AccBal:500, AccType:"S"})**

**db.Customers.insert({CustID:"C123", AccBal:900, AccType:"S"})**

**db.Customers.insert({CustID:"C123", AccBal:1500, AccType:"C"})**

**db.Customers.insert({CustID:"C111", AccBal:1200, AccType:"S"})**

**db.Customers.insert({CustID:"C111", AccBal:1000, AccType:"C"})**

**db.Customers.insert({CustID:"C111", AccBal:5000, AccType:"S"})**

**Find Out total balance, Average balance of the customer whose AccType is “S”.**

db.Customers.insert({CustID:"C123", AccBal:500, AccType:"S"})

db.Customers.insert({CustID:"C123", AccBal:900, AccType:"S"})

db.Customers.insert({CustID:"C123", AccBal:1500, AccType:"C"})

db.Customers.insert({CustID:"C123", AccBal:1200, AccType:"S"})

db.Customers.insert({CustID:"C111", AccBal:1200, AccType:"S"})

db.Customers.insert({CustID:"C111", AccBal:1000, AccType:"C"})

db.Customers.insert({CustID:"C111", AccBal:5000, AccType:"S"})

db.Customers.insert({CustID:"C112", AccBal:5000, AccType:"S"})

db.Customers.insert({CustID:"C112", AccBal:200, AccType:"S"})

db.createCollection("Customers")

var map=function(){emit(this.CustID,this.AccBal);}

var reduce=function(key,values){return Array.sum(values);}

db.Customers.mapReduce(map,reduce,{out:"Customers\_Totals"})

db.Customers\_Totals.find().pretty()

{ "\_id" : "C111", "value" : 7200 }

{ "\_id" : "C112", "value" : 5200 }

{ "\_id" : "C123", "value" : 4100 }

db.Customers.mapReduce(map,reduce,{out:"Customers\_Totals",query:{AccType:"S"}})

db.Customers\_Totals.find().pretty()

{ "\_id" : "C111", "value" : 6200 }

{ "\_id" : "C112", "value" : 5200 }

{ "\_id" : "C123", "value" : 2600 }

db.Customers.mapReduce(map,reduce,{out:"Customers\_Totals",query:{AccType:"C"}})

db.Customers\_Totals.find().pretty()

{ "\_id" : "C111", "value" : 1000 }

{ "\_id" : "C123", "value" : 1500 }

**Day-6**

**Create a database named “BookStore” in MongoDB with a**

**collection called “Books” containing documents with some or all of the following**

**fields:**

**bookId, bookTitle,authors(containing fields: authorName),**

**publicationYear, publisher,**

**Orders(containing fields: OrderedId, orderDate, customerName,**

**price,quantityOrdered, discount).**

**1. Use MapReduce function to display the total quantity of books ordered for**

**each date.**

**2. Use Map Reduce function to display the discount offered to a particular**

**customer.**

db.Books.insertMany([

{\_id:1,bookId:"b101",bookTitle:"The Secret 1",authors:["Rhonda Byrne"],publicationYear:2006,publisher:"Atria Publishing Group",

orders:[{OrderedId:"o101", orderDate:new Date("2020-02-11"), customerName:"Jainam", price:1000, quantityOrdered:1, discount:100},{OrderedId:"o102", orderDate:new Date("2020-02-12"), customerName:"Rahil", price:1000,

quantityOrdered:2, discount:50},{OrderedId:"o103", orderDate:new Date("2020-02-13"), customerName:"Gautam", price:1000,

quantityOrdered:2, discount:150},{OrderedId:"o104", orderDate:new Date("2020-02-14"), customerName:"Darshan", price:1000,

quantityOrdered:1, discount:100}]},

{\_id:2,bookId:"b102",bookTitle:"The Secret 2",authors:["Rhonda Byrne","Bob Proctor"],publicationYear:2006,publisher:"Atria Publishing Group",

orders:[{OrderedId:"o101", orderDate:new Date("2020-02-11"), customerName:"Jainam", price:1000, quantityOrdered:1, discount:100},{OrderedId:"o102", orderDate:new Date("2020-02-12"), customerName:"Rahil", price:1000,

quantityOrdered:2, discount:50},{OrderedId:"o103", orderDate:new Date("2020-02-13"), customerName:"Gautam", price:1000,

quantityOrdered:2, discount:150}]},

{\_id:3,bookId:"b103",bookTitle:"The Secret 3",authors:["Rhonda Byrne","Esther Hicks"],publicationYear:2006,publisher:"Atria Publishing Group",

orders:[{OrderedId:"o101", orderDate:new Date("2020-02-11"), customerName:"Jainam", price:1000, quantityOrdered:1, discount:100},{OrderedId:"o102", orderDate:new Date("2020-02-12"), customerName:"Rahil", price:1000,

quantityOrdered:2, discount:50},{OrderedId:"o103", orderDate:new Date("2020-02-13"), customerName:"Gautam", price:1000,

quantityOrdered:2, discount:150},{OrderedId:"o104", orderDate:new Date("2020-02-14"), customerName:"Darshan", price:1000,

quantityOrdered:1, discount:100}]},

{\_id:4,bookId:"b104",bookTitle:"The Secret 4",authors:["Rhonda Byrne","Bob Proctor"],publicationYear:2006,publisher:"Beyond Words Publishing",

orders:[{OrderedId:"o101", orderDate:new Date("2020-02-11"), customerName:"Jainam", price:1000, quantityOrdered:1, discount:100},{OrderedId:"o102", orderDate:new Date("2020-02-12"), customerName:"Rahil", price:1000,

quantityOrdered:2, discount:50},{OrderedId:"o103", orderDate:new Date("2020-02-13"), customerName:"Gautam", price:1000,

quantityOrdered:2, discount:150}]},

{\_id:5,bookId:"b105",bookTitle:"The Secret 5",authors:["Rhonda Byrne"],publicationYear:2006,publisher:"Atria Publishing Group",

orders:[{OrderedId:"o101", orderDate:new Date("2020-02-11"), customerName:"Jainam", price:1000, quantityOrdered:1, discount:100},{OrderedId:"o102", orderDate:new Date("2020-02-12"), customerName:"Rahil", price:1000,

quantityOrdered:2, discount:50}]},

{\_id:6,bookId:"b106",bookTitle:"The Secret 6",authors:["Rhonda Byrne","EstherHicks","Esther Hicks"],publicationYear:2006,publisher:"Beyond Words Publishing",

orders:[{OrderedId:"o101", orderDate:new Date("2020-02-11"), customerName:"Jainam", price:1000, quantityOrdered:1, discount:100},{OrderedId:"o102", orderDate:new Date("2020-02-12"), customerName:"Rahil", price:1000,

quantityOrdered:2, discount:50},{OrderedId:"o103", orderDate:new Date("2020-02-13"), customerName:"Gautam", price:1000,

quantityOrdered:2, discount:150}]},

{\_id:7,bookId:"b107",bookTitle:"The Secret 7",authors:["Rhonda Byrne","BobProctor","Esther Hicks"],publicationYear:2006,publisher:"Atria Publishing Group",

orders:[{OrderedId:"o103", orderDate:new Date("2020-02-13"), customerName:"Gautam", price:1000, quantityOrdered:2, discount:150},{OrderedId:"o104", orderDate:new Date("2020-02-14"), customerName:"Darshan", price:1000,

quantityOrdered:1, discount:100}]},

{\_id:8,bookId:"b108",bookTitle:"The Secret 8",authors:["Rhonda Byrne"],publicationYear:2006,publisher:"Beyond Words Publishing",

orders:[{OrderedId:"o101", orderDate:new Date("2020-02-11"), customerName:"Jainam", price:1000, quantityOrdered:1, discount:100},{OrderedId:"o102", orderDate:new Date("2020-02-12"), customerName:"Rahil", price:1000,

quantityOrdered:2, discount:50},{OrderedId:"o103", orderDate:new Date("2020-02-13"), customerName:"Gautam", price:1000,

quantityOrdered:2, discount:150}]},

{\_id:9,bookId:"b109",bookTitle:"The Secret 9",authors:["Rhonda Byrne","Esther Hicks"],publicationYear:2006,publisher:"Atria Publishing Group",

orders:[{OrderedId:"o101", orderDate:new Date("2020-02-11"), customerName:"Jainam", price:1000, quantityOrdered:1, discount:100},{OrderedId:"o102", orderDate:new Date("2020-02-12"), customerName:"Rahil", price:1000,

quantityOrdered:2, discount:50},{OrderedId:"o104", orderDate:new Date("2020-02-14"), customerName:"Darshan", price:1000,

quantityOrdered:1, discount:100}]},

{\_id:10,bookId:"b110",bookTitle:"The Secret 10",authors:["Rhonda Byrne","BobProctor","Esther Hicks"],publicationYear:2006,publisher:"Beyond Words Publishing",

orders:[{OrderedId:"o102", orderDate:new Date("2020-02-12"), customerName:"Rahil", price:1000, quantityOrdered:2, discount:50},{OrderedId:"o103", orderDate:new Date("2020-02-13"), customerName:"Gautam", price:1000,

quantityOrdered:2, discount:150},{OrderedId:"o104", orderDate:new Date("2020-02-14"), customerName:"Darshan", price:1000,

quantityOrdered:1, discount:100}]},

])

1. **Use MapReduce function to display the total quantity of books ordered for each date**

var map = function(){

for(var i=0; i<this.orders.length; i++)

{

var key=this.orders[i].orderDate;

var value={count:1,qty:this.orders[i].quantityOrdered}; emit(key,value);}

}

var reduce = function(key,value){ reduceval = {count:0,qty:0}; for(var i=0; i<value.length; i++)

{

reduceval.count=reduceval.count+value[i].count; reduceval.qty=reduceval.qty+value[i].qty;

}

return reduceval;

}

db.Books.mapReduce(map,reduce,{out:"TOTAL"}) db.TOTAL.find()

1. **Use Map Reduce function to display the discount offered to a particular customer**

var map = function(){

for(i=0; i<this.orders.length; i++)

{

var key = this.orders[i].customerName;

var values = {count:1,total\_dis:this.orders[i].discount}; emit(key,values);

}

}

var reduce = function(key,value){ reduceval={count:0,total\_dis:0}; for(var i=0; i<value.length; i++)

{

reduceval.count+=value[i].count; reduceval.total\_dis+=value[i].total\_dis;

}

return reduceval;

}

db.Books.mapReduce(map,reduce,{out:"TOTAL\_DISCOUNT"}) db.TOTAL\_DISCOUNT.find()