PG-DBDA September 2023
C-DAC THIRUVANANTHAPURAM

MongoDB – Lab 2

1. a) Create a database named **college** and create a collection named **student.**
Ans:
test> use college
db.createCollection("student")

b) Insert some documents to the collection with fields

studentid, name, batch(Science ,Commerce etc), **age, status**(present/absent).

Ans:

db.student.insertMany([

{studentid:1,name:"John Doe",batch:"Science",age:20,status:"present"},

{studentid:2,name:"Jane Smith",batch:"Commerce",age:19,status:"absent"},

{studentid:3,name:"Bob Johnson",batch:"Science",age:21,status:"present"},

{studentid:4,name:"Alice Brown",batch:"Commerce",age:22,status:"present"}

])

```
test> use college
switched to db college
college> db.createCollection("student")
{ ok: 1 }
college> db.student.insertMany([
...   { studentid: 1, name: "John Doe", batch: "Science", age: 20, status: "present" },
...   { studentid: 2, name: "Jane Smith", batch: "Commerce", age: 19, status: "absent" },
...   { studentid: 3, name: "Bob Johnson", batch: "Science", age: 21, status: "present" }
...   { studentid: 4, name: "Alice Brown", batch: "Commerce", age: 22, status: "present"
... ])
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId("6517e7953896bd758d0264c0"),
    '1': ObjectId("6517e7953896bd758d0264c1"),
    '2': ObjectId("6517e7953896bd758d0264c2"),
    '3': ObjectId("6517e7953896bd758d0264c3")
  }
}
```

c) Display the students details in descending order based on their age.

Ans:

db.student.find().sort({ age: -1 })

```
college> db.student.find().sort({ age: -1 })
[
  {
    _id: ObjectId("6517e7953896bd758d0264c3"),
    studentid: 4,
    name: 'Alice Brown',
    batch: 'Commerce',
    age: 22,
    status: 'present'
  },
  {
    _id: ObjectId("6517e7953896bd758d0264c2"),
    studentid: 3,
    name: 'Bob Johnson',
    batch: 'Science',
    age: 21,
    status: 'present'
  },
  {
    _id: ObjectId("6517e7953896bd758d0264c0"),
    studentid: 1,
    name: 'John Doe',
    batch: 'Science',
    age: 20,
    status: 'present'
  },
  {
    _id: ObjectId("6517e7953896bd758d0264c1"),
    studentid: 2,
    name: 'Jane Smith',
    batch: 'Commerce',
    age: 19,
    status: 'absent'
  }
]
```

d) Update the batch-name science to science and technology

Ans:

db.student.updateMany({ batch: "Science" },{ $set: { batch: "Science and Technology" } })

```
...   { batch: "Science" },
...   { $set: { batch: "Science and Technology" } }
...   { batch: "Science" },
...   { $set: { batch: "Science and Technology" } }
... )
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 2,
  modifiedCount: 2,
  upsertedCount: 0
}
```

e) Count the number of students who are present.

Ans:

db.student.aggregate([{$match:{ status: "present"}},{$group:{_id: null,count: { $sum: 1 }}}])

```
college> db.student.aggregate([{$match:{ status: "present"}},{$group:{_id: null,count: { $sum: 1 }}}
[ { _id: null, count: 3 } ]
college>
```

f) Remove the status field.

Ans:

db.student.updateMany({}, { $unset: { status: "" } })

```
college> db.student.updateMany({}, { $unset: { status: "" } })
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 4,
  modifiedCount: 4,
  upsertedCount: 0
}
college> db.student.find()
[
  {
    _id: ObjectId("6517e7953896bd758d0264c0"),
    studentid: 1,
    name: 'John Doe',
    batch: 'Science and Technology',
    age: 20
  },
  {
    _id: ObjectId("6517e7953896bd758d0264c1"),
    studentid: 2,
    name: 'Jane Smith',
    batch: 'Commerce',
    age: 19
  },
  {
    _id: ObjectId("6517e7953896bd758d0264c2"),
    studentid: 3,
    name: 'Bob Johnson',
    batch: 'Science and Technology',
    age: 21
  },
  {
    _id: ObjectId("6517e7953896bd758d0264c3"),
    studentid: 4,
    name: 'Alice Brown',
    batch: 'Commerce',
    age: 22
  }
]
college>
```

g) Remove all students from commerce batch.

Ans:

db.student.deleteMany({ batch: "Commerce" })

```
college> db.student.deleteMany({ batch: "Commerce" })
{ acknowledged: true, deletedCount: 2 }
```

```
college> db.student.find()
[ acknowledged: true, deletedCount: 2 }
  {
    _id: ObjectId("6517e7953896bd758d0264c0"),
    studentid: 1,
    name: 'John Doe',
    batch: 'Science and Technology',
    age: 20
  },
  {
    _id: ObjectId("6517e7953896bd758d0264c2"),
    studentid: 3,
    name: 'Bob Johnson',
    batch: 'Science and Technology',
    age: 21
  }
]
college>
```

2. a) Create database named **company** and create a collection named **employee.**
Ans:
use company
db.createCollection(“employee”)

b) Insert some documents to the collection with fields **empid, name, address, email, salary and designation.**

Ans:

db.employee.insertMany([{empid:1,name:"ken",address:"123 main st",email:"ken@example.com",salary:50000,designation:"Manager" },{empid:2,name:"smith",address:"456 abc st",email:"smith@example.com",salary:60000,designation:"Developer" },{empid:3,name:"bob",address:"789 pqr st", email:"bob@example.com",salary:55000,designation:"Salesperson"},{empid:4,name:"Alice",address:"101 ghi st",email:"alice@example.com",salary:70000,designation:"Manager}])

```
company> db.employee.drop()
true
... {empid:2,name:"smith",address:"456 abc st",email:"smith@example.com",salary:60000,designation:"Developer"
... {empid:3,name:"bob",address:"789 pqr st", email:"bob@example.com",salary:55000,designation:"Salesperson"}
company>
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId("6517f0dbc9fc3b768b601a3f"),
    '1': ObjectId("6517f0dbc9fc3b768b601a40"),
    '2': ObjectId("6517f0dbc9fc3b768b601a41"),
    '3': ObjectId("6517f0dbc9fc3b768b601a42")
  }
}
```

c) Display all the employee details.

Ans:

Db.employee.find()

```
company> db.employee.find()
[
  {
    _id: ObjectId("6517f0dbc9fc3b768b601a3f"),
    empid: 1,ctId("6517f0dbc9fc3b768b601a3f"),
    name: 'ken',d("6517f0dbc9fc3b768b601a40"),
    address: '123 main st',c9fc3b768b601a41"),
    email: 'ken@example.com',fc3b768b601a42")
    salary: 50000,
    designation: 'Manager'
  },
  {
    _id: ObjectId("6517f0dbc9fc3b768b601a40"),
    empid: 2,
    name: 'smith',
    address: '456 abc st',
    email: 'smith@example.com',
    salary: 60000,
    designation: 'Developer'
  },
  {
    _id: ObjectId("6517f0dbc9fc3b768b601a41"),
    empid: 3,
    name: 'bob',
    address: '789 pqr st',
    email: 'bob@example.com',
    salary: 55000,
    designation: 'Salesperson'
  },
  {
    _id: ObjectId("6517f0dbc9fc3b768b601a42"),
    empid: 4,
    name: 'Alice',
    address: '101 ghi st',
    email: 'alice@example.com',
    salary: 70000,
    designation: 'Manager'
  }
]
company>
```

d)

e) Update salary of a particular employee.

Ans:

db.employee.updateOne({empid:2},{$set:{salary:65000}})

```
company> db.employee.updateOne({empid:2},{$set:{salary:65000}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
company> db.employee.find({empid:2})
[
  {
    _id: ObjectId("6517f0dbc9fc3b768b601a40"),
    empid: 2,
    name: 'smith',
    address: '456 abc st',
    email: 'smith@example.com',
    salary: 65000,
    designation: 'Developer'
  }
]
company>
```

f) Add one more field department to the collection.

Ans: db.employee.updateMany({},{$set:{department:""}})

```
company> db.employee.updateMany({},{$set:{department:""}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 4,
  modifiedCount: 4,
  upsertedCount: 0
}
```

```
company> db.employee.find()
[
  {cknowledged: true,
    _id: ObjectId("6517f0dbc9fc3b768b601a3f"),
    empid: 1,t: 4,
    name: 'ken', 4,
    address: '123 main st',
    email: 'ken@example.com',
    salary: 50000,
    designation: 'Manager',
    department: ''
  },
  {
    _id: ObjectId("6517f0dbc9fc3b768b601a40"),
    empid: 2,
    name: 'smith',
    address: '456 abc st',
    email: 'smith@example.com',
    salary: 65000,
    designation: 'Developer',
    department: ''
  },
  {
    _id: ObjectId("6517f0dbc9fc3b768b601a41"),
    empid: 3,
    name: 'bob',
    address: '789 pqr st',
    email: 'bob@example.com',
    salary: 55000,
    designation: 'Salesperson',
    department: ''
  },
  {
    _id: ObjectId("6517f0dbc9fc3b768b601a42"),
    empid: 4,
    name: 'Alice',
    address: '101 ghi st',
    email: 'alice@example.com',
    salary: 70000,
    designation: 'Manager',
    department: ''
  }
]
company>
```

g) Display the fields name, salary and designation for all the documents.

Ans:

db.employee.find({},{name:1,salary:1,designation:1})

```
company> db.employee.find({},{name:1,salary:1,designation:1})
[
  { name: 'ken', salary: 50000, designation: 'Manager' },
    _id: ObjectId("6517f0dbc9fc3b768b601a3f"), 'Developer' },
    name: 'ken', salary: 55000, designation: 'Salesperson' },
    salary: 50000, salary: 70000, designation: 'Manager' }
    designation: 'Manager'
  },
  {
    _id: ObjectId("6517f0dbc9fc3b768b601a40"),
    name: 'smith',
    salary: 65000,
    designation: 'Developer'
  },
  {
    _id: ObjectId("6517f0dbc9fc3b768b601a41"),
    name: 'bob',
    salary: 55000,
    designation: 'Salesperson'
  },
  {
    _id: ObjectId("6517f0dbc9fc3b768b601a42"),
    name: 'Alice',
    salary: 70000,
    designation: 'Manager'
  }
]
company>
```

h) Display the fields name, email and designation for all the documents but exclude the field _id.

Ans:

db.employee.find({},{name:1,salary:1,designation:1,_id:0})

```
company> db.employee.find({},{name:1,salary:1,designation:1,_id:0})
[
  { name: 'ken', salary: 50000, designation: 'Manager' },
  { name: 'smith', salary: 65000, designation: 'Developer' },
  { name: 'bob', salary: 55000, designation: 'Salesperson' },
  { name: 'Alice', salary: 70000, designation: 'Manager' }
]
```

i) Display all employee details whose salary is greater than a specified value.

Ans:

db.employee.find({salary:{$gt:60000}})

```
company> db.employee.find({salary:{$gt:60000}})
[
  {
    _id: ObjectId("6517f0dbc9fc3b768b601a40"),
    empid: 2,
    name: 'smith',
    address: '456 abc st',
    email: 'smith@example.com',
    salary: 65000,
    designation: 'Developer',
    department: ''
  },
  {
    _id: ObjectId("6517f0dbc9fc3b768b601a42"),
    empid: 4,
    name: 'Alice',
    address: '101 ghi st',
    email: 'alice@example.com',
    salary: 70000,
    designation: 'Manager',
    department: ''
  }
]
company>
```

j)

k) Find department wise total salary of employees.

Ans:

db.employee.aggregate([{$group:{_id:"$department",totalSalary:{$sum:"$salary"}}}])

```
company> db.employee.aggregate([{$group:{_id:"$department",totalSalary:{$sum:"$salary"}}}])
[
  { _id: 'IT', totalSalary: 105000 },
  { _id: 'sales', totalSalary: 135000 }
]
```

l) Create an index for department field.

Ans:

db.employee.createIndex({department:1})

```
company> db.employee.createIndex({department:1})
department_1
```

m) Display the no: of employees belonging to each department sorted in ascending order.

Ans:

db.employee.aggregate([{$group:{_id:"$department",count:{ $sum:1}}},{$sort:{count:1}}])

```
company> db.employee.aggregate([{$group:{_id:"$department",count:{ $sum:1}}},{$sort:{count:1}}])
[ { _id: 'IT', count: 2 }, { _id: 'sales', count: 2 } ]
company>
```

n) Remove all indexes from employee collection.

Ans:

db.employee.dropIndexes()

```
company> db.employee.dropIndexes()
{
  nIndexesWas: 2,
  msg: 'non-_id indexes dropped for collection',
  ok: 1
}
company>
```

o) Display only the first 3 employee details whose designation is given.

Ans:

db.employee.find({designation:"Manager"}).limit(3)

```
company> db.employee.find({designation:"Manager"}).limit(3)
[
  {
    _id: ObjectId("6517f0dbc9fc3b768b601a3f"),
    empid: 1,
    name: 'ken',
    address: '123 main st',
    email: 'ken@example.com',
    salary: 50000,
    designation: 'Manager',
    department: 'IT'
  },
  {
    _id: ObjectId("6517f0dbc9fc3b768b601a42"),
    empid: 4,
    name: 'Alice',
    address: '101 ghi st',
    email: 'alice@example.com',
    salary: 70000,
    designation: 'Manager',
    department: 'sales'
  },
  {
    _id: ObjectId("6517f8a5c9fc3b768b601a43"),
    empid: 5,
    name: 'scott',
    address: '123 Dun',
    email: 'sco@example.com',
    salary: 70000,
    designation: 'Manager',
    department: 'sales'
  }
]
company>
```