M1: Design and Develop MongoDB Queries using CRUD operations:
Create Employee collection by considering following Fields:
i. Name: Embedded Doc (FName, LName)
ii. Company Name: String
iii. Salary: Number
iv. Designation: String
v. Age: Number
vi. Expertise: Array
vii. DOB: String or Date
viii. Email id: String
ix. Contact: String
x. Address: Array of Embedded Doc (PAddr, LAddr)
Insert at least 5 documents in collection by considering above attribute and execute following queries:

```
db.Employee.insertMany([
     {
         Name: { FName: "Apoorva", LName: "Karne" },
         CompanyName: "Infosys",
         Salary: 15000000,
         Designation: "Web Developer",
         Age: 20,
         Expertise: ["HTML", "CSS", "Python", "Android"],
         DOB: "30-06-2004",
         Email_id: "apoorva@infosys.com",
         Contact: 7776543211,
         Address: [
             { PAddr: { city: "Pune", pin: 411014 }},
             { LAddr: { city: "Delhi", pin: 411005 }}
         ]
     },
     {
         Name: { FName: "Aditya", LName: "Konde" },
         CompanyName: "TCS",
         Salary: 3000000,
         Designation: "Programmer",
         Age: 21,
         Expertise: ["Python", "CPP", "Java", "Mongodb", "Mysql", "Cassandra"],
         DOB: "02-01-2003",
         Email_id: "aadi@tcs.com",
         Contact: 777777888888,
         Address: [
             { PAddr: { city: "Mumbai", pin: 411002 }},
             { LAddr: { city: "Pune", pin: 411014 }}
         ]
     },
     {
         Name: { FName: "Sara", LName: "Sayyad" },
         CompanyName: "TCS",
         Salary: 50000,
         Designation: "Programmer",
         Age: 21,
         Expertise: ["Python", "CPP", "Java", "Mongodb", "Mysql", "Cassandra"],
         DOB: "11-107-2002",
         Email_id: "sara@tcs.com",
         Contact: 98989888888,
         Address: [
             { PAddr: { city: "Delhi", pin: 411005 }},
             { LAddr: { city: "Pune", pin: 411014 }}
         ]
     },
 ]);

 db.Employee.insertMany([
```

```
{
    Name: { FName: "Ravi", LName: "Sharma" },
    CompanyName: "Wipro",
    Salary: 4500000,
    Designation: "Senior Developer",
    Age: 28,
    Expertise: ["Java", "Spring", "Hibernate", "SQL", "Python"],
    DOB: "15-05-1995",
    Email_id: "ravi@wipro.com",
    Contact: 9999887777,
    Address: [
        { PAddr: { city: "Bangalore", pin: 411003 }},
        { LAddr: { city: "Chennai", pin: 411007 }}
    ]
},
{
    Name: { FName: "Simran", LName: "Singh" },
    CompanyName: "Infosys",
    Salary: 2500000,
    Designation: "QA Engineer",
    Age: 24,
    Expertise: ["Testing", "Selenium", "Java", "Automation"],
    DOB: "10-11-1999",
    Email_id: "simran@infosys.com",
    Contact: 8887776666,
    Address: [
        { PAddr: { city: "Hyderabad", pin: 411009 }},
        { LAddr: { city: "Bangalore", pin: 411008 }}
    ]
},
{
    Name: { FName: "Nikita", LName: "Joshi" },
    CompanyName: "Tech Mahindra",
    Salary: 5500000,
    Designation: "Software Engineer",
    Age: 26,
    Expertise: ["JavaScript", "Node.js", "MongoDB", "React", "Express"],
    DOB: "20-08-1997",
    Email_id: "nikita@techm.com",
    Contact: 7778889999,
    Address: [
        { PAddr: { city: "Pune", pin: 411010 }},
        { LAddr: { city: "Mumbai", pin: 411006 }}
    ]
},
{
    Name: { FName: "Vishal", LName: "Patel" },
    CompanyName: "Accenture",
    Salary: 6500000,
    Designation: "Architect",
    Age: 32,
    Expertise: ["Cloud", "AWS", "Azure", "Terraform", "DevOps"],
    DOB: "01-03-1991",
    Email_id: "vishal@accenture.com",
    Contact: 7777774444,
    Address: [
        { PAddr: { city: "Mumbai", pin: 411013 }},
        { LAddr: { city: "Hyderabad", pin: 411004 }}
    ]
},
{
    Name: { FName: "Priya", LName: "Verma" },
    CompanyName: "Capgemini",
    Salary: 7000000,
```

```
        Designation: "Senior Consultant",
        Age: 29,
        Expertise: ["C#", ".NET", "SQL Server", "Angular", "JavaScript"],
        DOB: "12-02-1994",
        Email_id: "priya@capgemini.com",
        Contact: 9999777777,
        Address: [
            { PAddr: { city: "Gurgaon", pin: 411012 }},
            { LAddr: { city: "Noida", pin: 411011 }}
        ]
    }
]);
```

1. Select all documents where the Designation field has the value "Programmer" and the salary is greater than 30000:

```
db.Employee.find({
    Designation: "Programmer",
    Salary: { $gt: 30000 }
});
```

2. Create a new document if no document in the Employee collection contains {Designation: "Tester", Company_name: "TCS", Age: 25}:

```
db.Employee.updateOne(
    { Designation: "Tester", Company_name: "TCS", Age: 25 },
    { $setOnInsert: {
        Name: { FName: "New", LName: "Tester" },
        Company_name: "TCS",
        Salary: 35000,
        Designation: "Tester",
        Age: 25,
        Expertise: ["Manual Testing", "Automation"],
        DOB: new Date("1999-01-01"),
        Email: "new.tester@tcs.com",
        Contact: "6789012345",
        Address: [{ PAddr: "New Addr, Pune", LAddr: "Another Addr, Mumbai" }]
    }},
    { upsert: true }
);  //no $ for upsert
```

3. Increase the salary of each Employee working with â œInfosysâ by 10000:

```
db.Employee.updateMany(
    { Company_name: "Infosys" },
    { $inc: { Salary: 10000 } }
);
```

4. Find all employees working with "TCS" and reduce their salary by 5000:

```
db.Employee.updateMany(
    { Company_name: "TCS" },
    { $inc: { Salary: -5000 } }
);
```

5.Return documents where Designation is not equal to "Tester":

```
db.Employee.find({
    Designation: { $ne: "Tester" }
});
```

6. Find all employees with an exact match on an array having Expertise: ['Mongodb','Mysql','Cassandra']:

```
db.Employee.find({
    Expertise: ["Mongodb", "Mysql", "Cassandra"]
});
```

M2:

1. Find the full name (FName and LName) of employees where age is less than 30 and salary is more than 50000:

```
db.Employee.find(
    { Age: { $lt: 30 }, Salary: { $gt: 50000 } },
    { "Name.FName": 1, "Name.LName": 1, _id: 0 }
);
```

2. Insert a new document if no document in the collection contains {Designation: "Tester", Company_name: "TCS", Age: 25}:

```
db.Employee.updateOne(
    { Designation: "Tester", Company_name: "TCS", Age: 25 },
    { $setOnInsert: {
        Name: { FName: "New", LName: "Tester" },
        Company_name: "TCS",
        Salary: 30000,
        Designation: "Tester",
        Age: 25,
        Expertise: ["Manual Testing"],
        DOB: new Date("1999-01-01"),
        Email_id: "new.tester@tcs.com",
        Contact: "1234567890",
        Address: [{ PAddr: { city: "Pune", pin: 411014 } }]
    }},
    { upsert: true }
);
```

3. Select all documents where age is less than 30 or salary is greater than 40000:

```
db.Employee.find({
    $or: [
        { Age: { $lt: 30 } },
        { Salary: { $gt: 40000 } }
    ]
});
```

4. Find documents where Designation is not equal to "Developer":

```
db.Employee.find({
    Designation: { $ne: "Developer" }
});
```

5. Find _id, Designation, Address, and Name of all documents where Company_name is "Infosys":

```
db.Employee.find(
    { Company_name: "Infosys" },
    { _id: 1, Designation: 1, Address: 1, Name: 1 }
);
```

6. Display only FName and LName of all employees:

```
db.Employee.find(
    {},
    { "Name.FName": 1, "Name.LName": 1, _id: 0 }
);
```

M3:

1. Creates a new document if no document in the employee collection contains
{Designation: "Tester", Company_name: "TCS", Age: 25}

```
db.Employee.updateOne(
  { Designation: "Tester", CompanyName: "TCS", Age: 25 },
  { $setOnInsert: { Designation: "Tester", CompanyName: "TCS", Age: 25, Name: "New Tester", Salary: 3000000, Expertise: ["Testing", "Automation"] } },
  { upsert: true }
);
```

2. Finds all employees working with Company_name: "TCS" and
increase their salary by 2000.

```
db.Employee.updateMany(
  { CompanyName: "TCS" },
  { $inc: { Salary: 2000 } }
);
```

3. Matches all documents where the value of the field Address is an
embedded document that contains only the field city with the
value "Pune" and the field Pin_code with the value "411001".

```
db.Employee.find({
  "Address": {
    $elemMatch: {
      $or: [
        { PAddr: { city: "Pune", pin: "411001" } },
        { LAddr: { city: "Pune", pin: "411001" } }
      ]
    }
  }
});
```

4. Find employee details who are working as "Developer" or
"Tester".

```
db.Employee.find({
  Designation: { $in: ["Developer", "Tester"] }
});
```

5. Drop Single documents where designation="Developer".

```
db.Employee.deleteOne({ Designation: "Developer" });
```

6. Count number of documents in employee collection.

```
db.Employee.countDocuments();
```

M4:
Insert at least 5 documents in collection by considering above
attribute and execute following:
1. Using aggregation Return Designation with Total Salary is Above
200000.

```
db.Employee.aggregate([
  {
    $group: {
      _id: "$Designation",
      totalSalary: { $sum: "$Salary" }
    }
  },
  {
    $match: { totalSalary: { $gt: 200000 } }
  }
]);
```

2. Using Aggregate method returns names and _id in upper case and
in alphabetical order.

```
db.Employee.aggregate([
   {
     $project: {
       _id: 0,
       name: { $concat: ["$Name.FName", " ", "$Name.LName"] }
     }
   },
   { $sort: { name: 1 } }
 ]);
```

3. Using aggregation method find Employee with Total Salary for
Each City with Designation="DBA".

```
db.Employee.aggregate([ { $match: { Designation: "Programmer" } }, { $group:
{ _id: { $or: ["$Address.PAddr.city", "$Address.LAddr.city"] }, totalSalary:
{ $sum: "$Salary" } } }])
```

4. Create Single Field Indexes on Designation field of employee collection

```
db.Employee.createIndex({ Designation: 1 });
```

5. To Create Multikey Indexes on Expertise field of employee collection.

```
db.Employee.createIndex({ Expertise: 1 });
```

6. Create an Index on Emp_id field, compare the time require to search Emp_id before and after creating an index. (Hint Add at least 10000 Documents)

a. // You can use a loop to insert 10,000 documents

```
for (let i = 0; i < 10000; i++) {
  db.Employee.insertOne({
    Name: { FName: `Emp${i}`, LName: `Test` },
    CompanyName: "TestCompany",
    Salary: 500000,
    Designation: "Developer",
    Age: 25,
    Expertise: ["JavaScript", "Node.js"],
    DOB: "01-01-1997",
    Email_id: `emp${i}@test.com`,
    Contact: 999888777,
    Address: [
      { PAddr: { city: "City", pin: 123456 }},
      { LAddr: { city: "City", pin: 123456 }}
    ]
  });
}
```

b. `db.Employee.find({}).explain("executionStats")`

c. `db.Employee.createIndex({ Emp_id: 1 });`

d. `db.Employee.find({}).explain("executionStats")`

7. Return a List of Indexes on created on employee Collection.

```
db.Employee.getIndexes();
```

M5:

1. Using aggregation Return separates value in the Expertise array and return sum of each element of array.

```
db.Employee.aggregate([
{
     $unwind: "$Expertise"
   },
   {
     $group: {
       _id: "$Expertise",
       totalSalary: { $sum: "$Salary" }
     }
   }
 ]);
```

2. Using Aggregate method return Max and Min Salary for each company.

```
db.Employee.aggregate([
   {
     $group: {
       _id: "$CompanyName",
       maxSalary: { $max: "$Salary" },
       minSalary: { $min: "$Salary" }
     }
   }
```

```
 ]);
```
3. Using Aggregate method find Employee with Total Salary for Each City with Designation="DBA".
```
db.Employee.aggregate([
   {
     $match: {
       Designation: "Programmer"
     }
   },
   {
     $group: {
       _id: "$Address.PAddr.city",  // Correct the field name here
       totalSalary: { $sum: "$Salary" }
     }
   }
 ]);
```
4. Using aggregation method Return separates value in the Expertise array for employee name where Swapnil Jadhav
```
db.Employee.aggregate([
   {
     $match: {
       "Name.FName": "Sara",
       "Name.LName": "Sayyad"
     }
   },
   {
     $unwind: "$Expertise"
   },
   {
     $project: {
       _id: 0,
       expertise: "$Expertise"
     }
   }
 ]);
```
5. To Create Compound Indexes on Name: 1, Age: -1
```
db.Employee.createIndex({ "Name.FName": 1, "Age": -1 });
```
6. Create an Index on Emp_id field, compare the time require to search Emp_id before and after creating an index. (Hint Add at least 10000 Documents)
```
a. db.Employee.find({ Emp_id: 12345 }).explain("executionStats");
b. db.Employee.createIndex({ Emp_id: 1 });
c. db.Employee.find({ Emp_id: 12345 }).explain("executionStats");
```

7. Return a List of Indexes on created on employee Collection.
```
db.Employee.getIndexes();
```