Q.a) Create a database named college and create a collection named student.

```
pets> use college
switched to db college
```

```
college> db.createCollection("student")
{ ok: 1 }
```

b) Insert some documents to the collection with fields

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

```
college> db.student.insertMany([{studentid:1,name:"hari",batch:"science",age
:20,status:"present"},{studentid:2,name:"john",batch:"commerce",age:25,statu
s:"absent"},{studentid:3,name:"diya",batch:"computer",age:23,status:"present
"},{studentid:4,name:"ruhi",batch:"electronics",age:21,status:"present"}])
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId("6517e0f06a90424444701c67"),
    '1': ObjectId("6517e0f06a90424444701c68"),
    '2': ObjectId("6517e0f06a90424444701c69"),
    '3': ObjectId("6517e0f06a90424444701c6a")
  }
}
```

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

```
college> db.student.find().sort({age:-1})
[
  {
    _id: ObjectId("6517e0f06a90424444701c68"),
    studentid: 2,
    name: 'john',
    batch: 'commerce',
    age: 25,
    status: 'absent'
  },
  {
    _id: ObjectId("6517e0f06a90424444701c69"),
    studentid: 3,
    name: 'diya',
    batch: 'computer',
    age: 23,
    status: 'present'
  },
  {
    _id: ObjectId("6517e0f06a90424444701c6a"),
    studentid: 4,
    name: 'ruhi',
    batch: 'electronics',
    age: 21,
    status: 'present'
  },
  {
    _id: ObjectId("6517e0f06a90424444701c67"),
    studentid: 1,
    name: 'hari',
    batch: 'science',
    age: 20,
    status: 'present'
  }
]
```

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

```
college> db.student.updateMany({batch:"science"},{$set:{batch:"science and t
echnology"}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

```
college> db.student.find()
[
  {
    _id: ObjectId("6517e0f06a90424444701c67"),
    studentid: 1,
    name: 'hari',
    batch: 'science and technology',
    age: 20,
    status: 'present'
  },
```

e) Count the number of students who are present.

```
college> db.student.countDocuments({"status":"present"})
3
```

f) Remove the status field.

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

```
college> db.student.updateMany({},{$unset:{status:"absent"}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 4,
  modifiedCount: 3,
  upsertedCount: 0
}
college> db.student.find()
[
  {
    _id: ObjectId("6517e0f06a90424444701c67"),
    studentid: 1,
    name: 'hari',
    batch: 'science and technology',
    age: 20
  },
  {
    _id: ObjectId("6517e0f06a90424444701c68"),
    studentid: 2,
    name: 'john',
    batch: 'commerce',
    age: 25
  },
  {
    _id: ObjectId("6517e0f06a90424444701c69"),
    studentid: 3,
    name: 'diya',
    batch: 'computer',
    age: 23
  },
  {
    _id: ObjectId("6517e0f06a90424444701c6a"),
    studentid: 4,
    name: 'ruhi',
    batch: 'electronics',
    age: 21
  }
```

g) Remove all students from commerce batch.

```
college> db.student.deleteMany({batch:"commerce"})
{ acknowledged: true, deletedCount: 1 }
college> db.student.find()
[
  {
    _id: ObjectId("6517e0f06a90424444701c67"),
    studentid: 1,
    name: 'hari',
    batch: 'science and technology',
    age: 20
  },
  {
    _id: ObjectId("6517e0f06a90424444701c69"),
    studentid: 3,
    name: 'diya',
    batch: 'computer',
    age: 23
  },
  {
    _id: ObjectId("6517e0f06a90424444701c6a"),
    studentid: 4,
    name: 'ruhi',
    batch: 'electronics',
    age: 21
  }
]
```

Q2.

a) Create database named company and create a collection named employee.

```
college> use company
switched to db company
company> db.createCollection("employee")
{ ok: 1 }
```

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

```
company> db.employee.insertMany([{empid:1,name:"raghav",address:"abc",email:"raghaw@.com",sa
lary:50000,designation:"administrator"},{empid:2,name:"aadi",address:"efg",email:"aadi@.com"
,salary:60000,designation:"software engg"},{empid:3,name:"anna",address:"hij",email:"anna@.c
om",salary:65000,designation:"developer"},{empid:4,name:"roy",address:"klm",email:"roy@.com"
,salary:75000,designation:"developer"}])
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId("6517f04d6a90424444701c6b"),
    '1': ObjectId("6517f04d6a90424444701c6c"),
    '2': ObjectId("6517f04d6a90424444701c6d"),
    '3': ObjectId("6517f04d6a90424444701c6e")
  }
```

c)Display all the employee details.

```
company> db.employee.find()
[
  {
    _id: ObjectId("6517f04d6a90424444701c6b"),
    empid: 1,
    name: 'raghav',
    address: 'abc',
    email: 'raghaw@.com',
    salary: 50000,
    designation: 'administrator'
  },
  {
    _id: ObjectId("6517f04d6a90424444701c6c"),
    empid: 2,
    name: 'aadi',
    address: 'efg',
    email: 'aadi@.com',
    salary: 60000,
    designation: 'software engg'
  },
  {
    _id: ObjectId("6517f04d6a90424444701c6d"),
    empid: 3,
    name: 'anna',
    address: 'hij',
    email: 'anna@.com',
    salary: 65000,
    designation: 'developer'
  },
  {
    _id: ObjectId("6517f04d6a90424444701c6e"),
    empid: 4,
    name: 'roy',
    address: 'klm',
    email: 'roy@.com',
    salary: 75000,
    designation: 'developer'
  }
]
```

d) Update salary of a particular employee.

```
company> db.employee.updateOne({empid:1},{$set:{salary:55000}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
company> db.employee.find()
[
  {
    _id: ObjectId("6517f04d6a90424444701c6b"),
    empid: 1,
    name: 'raghav',
    address: 'abc',
    email: 'raghaw@.com',
    salary: 55000,
    designation: 'administrator'
  },
```

e) Add one more field department to the collection.

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

```
company> db.employee.find()
[
  {
    _id: ObjectId("6517f04d6a90424444701c6b"),
    empid: 1,
    name: 'raghav',
    address: 'abc',
    email: 'raghaw@.com',
    salary: 55000,
    designation: 'administrator',
    department: 'IT'
  },
  {
    _id: ObjectId("6517f04d6a90424444701c6c"),
    empid: 2,
    name: 'aadi',
    address: 'efg',
    email: 'aadi@.com',
    salary: 60000,
    designation: 'software engg',
    department: 'IT'
  },
  {
    _id: ObjectId("6517f04d6a90424444701c6d"),
    empid: 3,
    name: 'anna',
    address: 'hij',
    email: 'anna@.com',
    salary: 65000,
    designation: 'developer',
    department: 'IT'
  },
  {
    _id: ObjectId("6517f04d6a90424444701c6e"),
    empid: 4,
    name: 'roy',
    address: 'klm',
    email: 'roy@.com',
    salary: 75000,
    designation: 'developer',
    department: 'IT'
  }
]
```

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

```
company> db.employee.find({},{_id:0,name:1,salary:1,designation:1})
[
  { name: 'raghav', salary: 55000, designation: 'administrator' },
  { name: 'aadi', salary: 60000, designation: 'software engg' },
  { name: 'anna', salary: 65000, designation: 'developer' },
  { name: 'roy', salary: 75000, designation: 'developer' }
]
```

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

```
company> db.employee.find({},{_id:0,name:1,email:1,designation:1})
[
  {
    name: 'raghav',
    email: 'raghaw@.com',
    designation: 'administrator'
  },
  { name: 'aadi', email: 'aadi@.com', designation: 'software engg' },
  { name: 'anna', email: 'anna@.com', designation: 'developer' },
  { name: 'roy', email: 'roy@.com', designation: 'developer' }
]
```

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

```
company> db.employee.find({salary:{ $gt :60000}})
[
  {
    _id: ObjectId("6517f04d6a90424444701c6d"),
    empid: 3,
    name: 'anna',
    address: 'hij',
    email: 'anna@.com',
    salary: 65000,
    designation: 'developer',
    department: 'IT'
  },
  {
    _id: ObjectId("6517f04d6a90424444701c6e"),
    empid: 4,
    name: 'roy',
    address: 'klm',
    email: 'roy@.com',
    salary: 75000,
    designation: 'developer',
    department: 'IT'
  }
```

i) Find department wise total salary of employees.

```
company> db.employee.aggregate([{$group:{_id:"$department",totalsalary:{"$sum":"$salary"}}}])
[ { _id: 'IT', totalsalary: 255000 } ]
```

j) Create an index for department field.

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

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

```
company> db.employee.aggregate([{$group:{_id:"department",count:{$sum:1}}},{$sort:{_id:1}}])
[ { _id: 'department', count: 4 } ]
```

l) Remove all indexes from employee collection.

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

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

```
company> db.employee.find({designation:"administrator"}).limit(3)
[
  {
    _id: ObjectId("6517f04d6a90424444701c6b"),
    empid: 1,
    name: 'raghav',
    address: 'abc',
    email: 'raghaw@.com',
    salary: 55000,
    designation: 'administrator',
    department: 'IT'
  }
]
```