Creating two collections employee and department in company database

```
> use company;
switched to db company
> db.createCollection("employee");
{ "ok" : 1 }
> db.createCollection("department");
{ "ok" : 1 }
> show collections;
department
employee
```

Inserting 5 documents into employee collection

```
> db.employee.insert({_id:1,name:"Prateek Aryan", designation:"Manager"});
WriteResult({ "nInserted" : 1 })
> db.employee.insert({_id:2,name:"Saifur Rahman",desination:"CTO"});
WriteResult({ "nInserted" : 1 })
> db.employee.insert({_id:3,name:"Mayank Wali",designation:"CEO"});
WriteResult({ "nInserted" : 1 })
> db.employee.insert({_id:4,name:"Srishti Rathi",designation:"Assistant to Manager"});
WriteResult({ "nInserted" : 1 })
> db.employee.insert({_id:5,name:"Harshwardhan",designation:"Assosciate Manager"});
WriteResult({ "nInserted" : 1 })
> db.employee.find().pretty();
{ "_id" : 1, "name" : "Prateek Aryan", "designation" : "Manager" }
{ "_id" : 2, "name" : "Saifur Rahman", "desination" : "CTO" }
{ "_id" : 3, "name" : "Mayank Wali", "designation" : "CEO" }
{
        "_id" : 4,
        "name" : "Srishti Rathi",
        "designation" : "Assistant to Manager"
}
{ "_id" : 5, "name" : "Harshwardhan", "designation" : "Assosciate Manager" }
```

Insertion using save

```
> db.employee.save({name:"Satyam",designation:"CFO"});
WriteResult({ "nInserted" : 1 })
> db.employee.find()
{ "_id" : 1, "name" : "Prateek Aryan", "designation" : "Manager", "dob" : "22.10.1998" }
{ "_id" : 2, "name" : "Saifur Rahman", "desination" : "CTO" }
{ "_id" : 3, "name" : "Mayank Wali", "designation" : "CEO" }
{ "_id" : 4, "name" : "Srishti Rathi", "designation" : "Assistant to Manager" }
{ "_id" : 5, "name" : "Harshwardhan", "designation" : "Assosciate Manager" }
{ "_id" : ObjectId("5f75a202193825c1a0449b4b"), "name" : "Satyam", "designation" : "CFO" }
```

Insertion using setting upsert: true in update parameters

```
> db.employee.update({_id:6},{$set:{name:"neha",designation:"xyz"}},{upsert:false});
WriteResult({ "nMatched" : 0, "nUpserted" : 0, "nModified" : 0 })
```

```
> db.employee.update({_id:6},{$set:{name:"neha",designation:"xyz"}},{upsert:true});
WriteResult({ "nMatched" : 0, "nUpserted" : 1, "nModified" : 0, "_id" : 6 })
> db.employee.find().pretty()
{
        "_id" : 1,
        "name" : "Prateek Aryan",
        "designation" : "Manager",
        "dob" : "22.10.1998"
}
{ "_id" : 2, "name" : "Saifur Rahman", "desination" : "CTO" }
{ "_id" : 3, "name" : "Mayank Wali", "designation" : "CEO" }
{
        "_id" : 4,
        "name" : "Srishti Rathi",
        "designation" : "Assistant to Manager"
}
{ "_id" : 5, "name" : "Harshwardhan", "designation" : "Assosciate Manager" }
{
        "_id" : ObjectId("5f75a202193825c1a0449b4b"),
        "name" : "Satyam",
        "designation" : "CFO"
}
{ "_id" : 6, "designation" : "xyz", "name" : "neha" }
```

Adding an extra field in already existing document in employee table

```
> db.employee.update({_id:1},{$set:{dob:"22.10.1998"}},{upsert:true});
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.employee.find().pretty();
{
        "_id" : 1,
        "name" : "Prateek Aryan",
        "designation" : "Manager",
        "dob" : "22.10.1998"
}
```

Removing an existing field from document

```
> db.employee.update({_id:1},{$unset:{dob:"22.10.1998"}});
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.employee.find(_id:1);
uncaught exception: SyntaxError: missing ) after argument list :
@(shell):1:20
> db.employee.find({_id:1});
{ "_id" : 1, "name" : "Prateek Aryan", "designation" : "Manager" }
```

Employees collection

```
> db.employee.find().pretty()
{ "_id" : 1, "name" : "Prateek Aryan", "designation" : "Manager" }
{ "_id" : 2, "name" : "Saifur Rahman", "desination" : "CTO" }
{ "_id" : 3, "name" : "Mayank Wali", "designation" : "CEO" }
{
        "_id" : 4,
        "name" : "Srishti Rathi",
        "designation" : "Assistant to Manager"
}
{ "_id" : 5, "name" : "Harshwardhan", "designation" : "Assosciate Manager" }
{
        "_id" : ObjectId("5f75a202193825c1a0449b4b"),
        "name" : "Satyam",
        "designation" : "CFO"
}
{ "_id" : 6, "designation" : "xyz", "name" : "neha" }
```

Department collection

```
> db.department.find().pretty()
{
        "_id" : ObjectId("5f759995193825c1a0449b4a"),
        "id" : 1,
        "name" : "Computer Science and Engineering",
        "code" : "CSE"
}
{
        "_id" : 2,
        "name" : "Information Science and Engineering",
        "code" : "ISE",
        "hod" : "XYZ"
}
{
        "_id" : 3,
        "name" : "Electronics and Communication and Engineering",
        "code" : "ECE",
        "hod" : "XYZ"
}
{
        "_id" : 4,
        "name" : "Mechanical Engineering",
        "code" : "ME",
        "hod" : "XYZ"
}
{ "_id" : 5, "name" : "Medical Electronis", "code" : "MI", "hod" : "XYZ" }
```

5. Select only employee name and department number whose department number falls between 1001 to 1005

3 to 5

```
> db.employee.find({_id:{$gte:3,$lte:5}}).pretty();
{ "_id" : 3, "name" : "Mayank Wali", "designation" : "CEO" }
{
        "_id" : 4,
        "name" : "Srishti Rathi",
        "designation" : "Assistant to Manager"
}
{ "_id" : 5, "name" : "Harshwardhan", "designation" : "Assosciate Manager" }
```

6. Select employee documents whose name begins with a.

```
> db.employee.find({name:/^n/});
{ "_id" : 6, "designation" : "xyz", "name" : "neha" }
```

7.Select employee document s whose age is greater than 30

```
> db.employee.find({_id:{$gte:3}}).pretty();
{ "_id" : 3, "name" : "Mayank Wali", "designation" : "CEO" }
{
        "_id" : 4,
        "name" : "Srishti Rathi",
        "designation" : "Assistant to Manager"
}
{ "_id" : 5, "name" : "Harshwardhan", "designation" : "Assosciate Manager" }
{ "_id" : 6, "designation" : "xyz", "name" : "neha" }
```