**Miscellaneous Examples**

**( For logical Reference only)**

**Example 1**

**create collections named demo having fields name,age,dep,salary,gender**

1.insert 7 records into demo collection

2. find name and age of all person who are female

db.demo.find({gender:'female'},{name:true,age:true,\_id:false})

3.update the sal of all by 5000

db.demo.updateMany({},{$inc:{sal:5000}})

4.arrange all records in asscending order of name

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

5.update all person who are male into female

db.demo.updateMany({gender:'male'},{$set:{gender:"female"}})

6.delete all records whose salary are gt 70000

db.deleteMany({sal:{$gt:70000}})

**Example 2**

**create a collection faculty having fields fname, lname, sub, exp, age, gender, sal**

1. enter 10 records into fac table

2. find all fnames where abc occurs as a substring or as a seperate word

db.fac.find({fname:/abc/},{fname:1,lname:1,\_id:0})

3. update document if lname beginning with P update their sal by half

db.fac.updateMany({lname:/^P/i},{$mul:{sal:0.5}})

4. find all records whose age are end with 0/5

db.fac.find({age:/0|5$/})

5. update first record by gender to others

db.fac.updateOne({gender:'others'})

6. count records who are teaching fsd-2 sub

db.fac.find({sub:'fsd-2'}).count()

7. update all record and insert current date to that record

db.fac.updateMany({},{$set:{date:new Date()}})

8. update sal of person whose sal less than 20000 to double

db.fac.updateMany({sal:{$lt:20000}},{$mul:{sal:2}})

9. update subject java to fsd-2 if java is not found insert using updateOne

db.fac.updateOne({sub:'java'},{$set:{sub:'fsd-2'}},{upsert:true})

10. delete all records and table

db.fac.deleteMany({}) or db.fac.drop()

**Example 3:**

**Write commands to perform following tasks on employee collection having fields**

**having name,age & joiningDate:**

**(1) Update the name="Senior citizen" having age>60 years.**

**(2) Update the name="JKL" having age=20 years. Insert this record, if it is not found.**

**(3) Retire all employees by deleting senior citizens from collection.**

>db.employee.updateMany({age:{$gt:60}},{$set:{name:”senior citizen”}})

>db.employee.updateMany({age:20},{$set:{name:”JKL”}},{upsert:true})

>db.employee.deleteMany({name:”senior citizen”})

**Example 4 :**

**Write commands to perform following tasks on employee collection having fields**

**name,age & joiningDate:**

**(1) Insert 3-4 records in collection.**

**(2) List all employees who joined before 1st January, 2010.**

**(3) Update the name of employee to "WWW" whose joiningDate is "05-05-2015"**

>db.employee.insertMany({name:"xyz",age:27,DOJ:ISODate("2014-07-03")},{name:"pqr",age:35,DOJ:ISODate("2011-02-10")},{name:"def",age:28,DOJ:ISODate("2009-08-25")},{name:"abc",age:30,DOJ:ISODate("2008-05-15")})

>db.employee.find({DOJ:{$lt:ISODate("2010-01-01")}})

>db.employee.updateOne({DOj:ISODate("2015-05-05")},{$set:{name:"WWW"}})

**Example 5:**

**Write commands to perform following tasks on employee collection having fields**

**having name,age & joiningDate:**

**(1) Delete all records having joiningDate before 1st January, 2010.**

**(2) List all employees having age>50 years.**

**(3) List only 1st employee having age>60 years**

>db.employee.deleteMany({ joiningDate: { $lt: ISODate("2010-01-01") } });

>db.employee.find({ age: { $gt: 50 } });

>db.employee.findOne({age:{$gt:60}})Top of Form

**Example 6:**

**Write commands to perform following tasks on employee collection having fields**

**having name,age & joiningDate:**

**(1) Count no. of employees having age>=60 years.**

**(2) List all employees in descending order of names having names "ABC", "PQR", "XYZ".**

**(3) List all employees whose age lies between 25 to 50 years excluding all rest of the fields.**

>db.employee.count({ age: { $gte: 60 } });

>db.employee.find({ name: { $in: ["ABC", "PQR", "XYZ"] } }).sort({ name: -1 });

>db.employee.find({ age: { $gte: 25, $lte: 50 } }, { \_id: 0, name: 1, age: 1 });

**Example:7**

**Consider following student collection:**

**[**

**{\_id:123433,name: "2DD", surname:"GGG", age:22},**

**{\_id:123434,name: "LLL", surname:"RRR", age:2},**

**{\_id:123435,name: "KKK", surname:"III", age:32},**

**{\_id:123436,name: "ZZZ", surname:"TTTT", age:9}**

**]**

**Do as directed:**

**(1) List all students whose name starts by digit only.**

**(2) List all students whose surname has exactly 4 letters only.**

**(3) List only names of students from youngest to oldest.**

**(4) List all students whose name has 3-10 letters only. Don't allow digits & underscore**

> db.student.find({ name: /^[0-9]/ });

> db.student.find({ surname: /^[A-Za-z]{4}$/ });

> db.student.find({}, { \_id: 0, name: 1 }).sort({ age: 1 });

> db.student.find({ name: /^[A-Za-z]{3,10}$/ });

**Example:8**

**Map following SQL queries to MongoDB query:**

**(1) alter table people add joiningDate datetime**

**(2) alter table people drop column joiningDate**

**(3) select age,name from people where status="PH"**

**(4) select \* from people where status!="PH"**

**(5) select name from people order by age desc**

> db.people.insertMany({ name: "John", age: 30, joiningDate: ISODate("2022-01-01") });

> db.people.updateMany({}, { $unset: { joiningDate: "" } });

>db.people.find({ status: "PH" }, { \_id: 0, age: 1, name: 1 });

> db.people.find({ status: { $ne: "PH" } });

> db.people.find({}, { \_id: 0, name: 1 }).sort({ age: -1 });

**Example:9**

**Map following SQL queries to MongoDB query:**

**(1) update employee set name="TTT" where age not in {12,33,44,55}**

**(2) select count(\*) from employee where age<>23**

**(3) update employee set age=age+10**

>db.employee.updateMany({ age: { $nin: [12, 33, 44, 55] } }, { $set: { name: "TTT" } });

> db.employee.count ({ age: { $ne: 23 } });

> db.employee.updateMany({}, { $inc: { age: 10 } });

**Example:10**

Write a MongoDB query to upsert a document in the products collection. The document should have productId (a string), name (a string), and price (a number). If a document with productId "P123" already exists, update its price to 29.99. If it does not exist, insert a new document with productId "P123", name "Gadget", and price 29.99.

db.products.updateOne(

{ productId: "P123" },

{

$set: { name: "Gadget", price: 29.99 },

},

{ upsert: true }

);