

# Query Documents

find() method is used to query the document. It displays the documents in non structured way.

```
db.collectionname.find({});
```

> db.stu3.find({}); - non formatted output

> db.stu3.find({}).pretty(); - formatted output



# Query Documents

Specifying the keys:

```
> db.salary.find({"DEPT" : "HR"});
```

\$and: operator – used to specify multiple keys

```
>db.collectionname.find({  
    $and: [  
        {key : value1}, {key : value2}]  
    });
```



# Query Documents

Specifying the keys:

```
>db.salary.find( {$and:
```

```
    [ {"DEPT" : "HR"},
```

```
      {"DESI" : "ASSOCIATE"}]
```

```
});
```

\$or: operator



# Query Documents

Specifying the keys:

\$or: operator

```
>db.collectionname.find({
```

```
    $or: [
```

```
        {key : value1}, {key : value2}]
```

```
    });
```

```
>db.salary.find( {
```

```
    $or: [
```

```
        {"DEPT" : "HR"},
```

```
        {"DESI" : "ASSOCIATE"}]
```

```
    }
```

```
});
```



# Update Documents

```
>db.collectionname.updateOne(<filter>,<update>,<options>);
```

```
>db.collectionname.updateMany(<filter>,<update>,<options>);
```

```
>db.collectionname.replaceOne(<filter>,<update>,<options>);
```

```
> db.emp.updateOne({"EID" : 1025},  
                    {$set: {"ADDRESS" : "B302 PRAGYA  
APARTMENTS, DWARKA, DELHI", "PHONE" : 9899245970} }  
                    );
```

```
> db.salary.updateMany({$and: [{"DEPT" : "HR"}, {"DESI" :  
"ASSOCIATE"}]}}, {$set: {"DESI" : "SR. ASSOCIATE"}});
```



# Update Documents

```
>db.collectionname.updateMany(<filter>,<update>,<options>);
```

```
> db.salary.updateMany(  
    {$and: [{"DEPT" : "HR"},  
            {"DESI" : "ASSOCIATE"}]},  
    {$set: {"DESI" : "SR. ASSOCIATE"}}  
);
```

```
> db.stu2.updateMany(  
    {},  
    {"$set" : {"class" : "8th"}});
```



# Update Documents

```
>db.collectionname.replaceOne(<filter>,<update>,<options>);
```

```
> db.stu.replaceOne(  
    {"rno" : 2},  
    {"rno" : 3 ,"name" : "ajay kumar",  
    "age" : 16}  
    );
```

Note: \_id remains same the document has been replaced



# ASSIGNMENT



- Create a database demo
- Create a EMP collection.
- Insert 8 documents containing in EMP collection containing eid, name, city, doj, dept, desi
- Display the documents in formatted manner.
- Show the documents containing “HR” dept.
- Show the documents for “OPS” managers.
- Promote all the associates as Sr.Associates



# MongoDb CRUD Operations

CRUD operations create, read, update, and delete operations on documents

Create Operations – to create a new document

```
db.collection.insertOne()
```

```
db.collection.insertMany()
```

Read Operations – to retrieve the document from collection

```
db.collection.find()
```

```
db.collection.findOne()
```



# MongoDb CRUD Operations

Update Operations – to modify existing documents in a collection.

```
db.collection.updateOne()
```

```
db.collection.updateMany()
```

```
db.collection.replaceOne()
```

Delete Operations – to remove documents from a collection.

```
db.collection.deleteOne()
```

```
db.collection.deleteMany()
```



# Inserting Documents

`db.collection.save()` can also be used to insert document into a collection.

If you don't specify `_id` in the document then **`save()`** method will work same as **`insert()`** method. If you specify `_id` then it will replace whole data of document containing `_id` as specified in `save()` method.

```
> db.stu2.save({"rno" : 7, "name" : "inderjeet singh", "age" : 15});
```

```
> db.stu2.save({"_id" :
```

```
    ObjectId("608fb168ea90e1f913d52714"),
```

```
    "rno" : 8,
```

```
    "name" : "Inder kumar singh",
```

```
    "age" : 16}
```

```
);
```



# Logical Operator in MongoDB

To query documents based on the AND condition, we need to use \$and keyword.

```
>db.collection.find({ $and: [ {<key1>:<value1>}, { <key2>:<value2>} ] })
```

```
> db.sal.find({  
    $and:[  
        {"DEPT" : "HR"},  
        {"DESI" : "ASSOCIATE"}  
    ]  
});
```

Note: and is the default logical operator when working with multiple conditions.



```
.sal.find({"DEPT" : "HR", "DESI" : "ASSOCIATE"});
```

# Logical Operator in MongoDB

To query documents based on the OR condition, we need to use \$or keyword.

```
>db.collection.find({ $or: [ {<key1>:<value1>}, { <key2>:<value2>} ] })
```

```
> db.sal.find(  
    {$or:[  
        {"DEPT" : "HR"},  
        {"DESI" : "ASSOCIATE"}  
    ]  
});
```



# Logical Operator in MongoDB

To query documents based on the NOT condition, we need to use \$not keyword.

```
> db.salary.find(  
    {"DEPT" :  
      {$not:  
        {$eq : "HR"}}  
    }  
  });
```



# Logical Operator in MongoDB

NOR is the combination of NOT and OR, we need to use \$nor keyword.

```
> db.salary.find({$nor:  
    [  
        { "DESI" : "ASSOCIATE"},  
        { "DEPT" : "HR"}  
    ]  
});
```



# Comparison Operator in MongoDB

**\$eq:** Equality Operator same as where dept = 'hr' in SQL


Syntax : {<key>:{\$eq:<value>}}

Eg1: > db.sal.find({"DEPT" :{\$eq : "HR"}});

Eg2: > db.sal.find({"DEPT" : "HR"});

**\$ne:** Not equal to Operator same as where dept <> 'hr' in SQL

Syntax : {<key>:{\$ne:<value>}}

 > db.sal.find({"DEPT" :{\$ne : "HR"}});



# Comparison Operator in MongoDB

**\$lt:** less than Operator same as where salary < 100000 in SQL

Syntax: {<key>:{\$lt:<value>}}

Eg: > db.sal.find({"SALARY" :{\$lt : 50000}});

**\$lte:** less than or equal to Operator same as where salary <= 50000 in SQL

Syntax: {<key>:{\$lte:<value>}}

Eg: > db.sal.find({"SALARY" :{\$lte : 50000}});



# Comparison Operator in MongoDB

**\$gt:** greater than or equal to Operator same as where salary >= 50000 in SQL

Syntax: {<key>:{\$gt:<value>}}

Eg: > db.sal.find({"SALARY" :{\$gt : 50000}});

**\$gte:** greater than or equal to Operator same as where salary >= 50000 in SQL

Syntax: {<key>:{\$gte:<value>}}

Eg: > db.sal.find({"SALARY" :{\$gte : 50000}});



# Comparison Operator in MongoDB

**\$in:** represents values in an array same as where dept in ('hr', 'it', 'admin')

Syntax: {<key>:{\$in:[<value1>, <value2>,.....<valueN>]}}

```
Eg: > db.sal.find({"DEPT" :  
                    {$in :  
                      ["HR", "IT", "TEMP"]  
                    }  
                });
```

**\$nin:** represents values not in an array same as where NOT dept in ('hr', 'it', 'admin')

Syntax: {<key>:{\$nin:[<value1>, <value2>,.....<valueN>]}}

```
Eg: > db.sal.find({"DEPT" :{$nin : ["HR", "IT", "TEMP","ADMIN"]}});
```



# Operators Examples

```
> db.orders.find({"Category" : {"$eq" : "Technology"},"Sub-Category" : {"$eq" : "Phones"}}).count()
```

```
> db.salary.find({"$and" : [{"DEPT" : {"$eq" : "HR"}}, {"DESI" : {"$eq" : "ASSOCIATE"}}]})
```

```
> db.salary.find({"$and" : [{"DEPT" : "HR"}, {"DESI" : "ASSOCIATE"}]})
```

```
> db.salary.find({"$and" : [{"$or" : [{"DEPT" : "HR"}, {"DEPT" : "MIS"}]}, {"$or" : [{"DESI" : "MANAGER"}, {"DESI" : "SR. ASSOCIATE"}]}]})
```

```
> db.salary.find({"SALARY" : {"$not" : {"$gt" : 100000}}})
```

